

# Package ‘sClust’

October 14, 2022

**Type** Package

**Title** R Toolbox for Unsupervised Spectral Clustering

**Version** 1.0

**Date** 2021-08-20

**Author** Emilie Poisson-Caillault [aut, cre, cph],  
Alain Lefebvre [ctb],  
Erwan Vincent [aut],  
Pierre-Alexandre Hebert [ctb]

**Description** Toolbox containing a variety of spectral clustering tools functions. Among the tools available are the hierarchical spectral clustering algorithm, the Shi and Malik clustering algorithm, the Perona and Freeman algorithm, the non-normalized clustering, the Von Luxburg algorithm, the Partition Around Medoids clustering algorithm, a multi-level clustering algorithm, recursive clustering and the fast method for all clustering algorithm. As well as other tools needed to run these algorithms or useful for unsupervised spectral clustering. This toolbox aims to gather the main tools for unsupervised spectral classification. See <<http://mawenzi.univ-littoral.fr/>> for more information and documentation.

**Depends** R (>= 3.0.0)

**Imports** cluster, stats, grDevices, class

**License** GPL (>= 2)

**RoxygenNote** 6.0.1

**NeedsCompilation** no

**Maintainer** Emilie Poisson-Caillault <emilie.caillault@univ-littoral.fr>

**Repository** CRAN

**Date/Publication** 2021-08-23 18:50:02 UTC

## R topics documented:

checking.gram.similarityMatrix . . . . .	2
compute.kclust . . . . .	3
compute.kclust2 . . . . .	4
compute.laplacian.NJW . . . . .	5

compute.nbCluster.gap . . . . .	6
compute.similarity.gaussien . . . . .	7
compute.similarity.ZP . . . . .	7
fastClustering . . . . .	8
fastMSC . . . . .	9
HierarchicalClust . . . . .	11
HierarchicalSC . . . . .	12
kmeansQuantization . . . . .	13
MSC . . . . .	14
PeronaFreemanSC . . . . .	15
recursClust . . . . .	17
search.neighbor . . . . .	18
ShiMalikSC . . . . .	19
spectralPAM . . . . .	20
UnnormalizedSC . . . . .	21
VonLuxburgSC . . . . .	22
<b>Index</b>	<b>24</b>

---

checking.gram.similarityMatrix  
*Gram similarity matrix checker*

---

## Description

Function to check if a similarity matrix is Gram or not

## Usage

```
checking.gram.similarityMatrix(W, flagDiagZero = FALSE, verbose = FALSE)
```

## Arguments

W                    Gram Similarity Matrix or not.  
flagDiagZero        if True, Put zero on the similarity matrix W.  
verbose              To output the verbose in the terminal.

## Value

a Gram similarity matrix

## Author(s)

Emilie Poisson Caillault and Erwan Vincent

**Examples**

```

### Example 1: 2 disks of the same size
n<-100 ; r1<-1
x<-(runif(n)-0.5)*2;
y<-(runif(n)-0.5)*2
keep1<-which((x*2+y*2)<(r1*2))
disk1<-data.frame(x+3*r1,y)[keep1,]
disk2 <-data.frame(x-3*r1,y)[keep1,]
sameTwoDisks <- rbind(disk1,disk2)
W <- compute.similarity.ZP(scale(sameTwoDisks))
W <- checking.gram.similarityMatrix(W)

### Example 2: Speed and Stopping Distances of Cars
W <- compute.similarity.ZP(scale(cars))
W <- checking.gram.similarityMatrix(W)

```

compute.kclust

*Gram similarity matrix checker***Description**

Function which select the number of cluster to compute thanks to a selected method

**Usage**

```

compute.kclust(
  eigenValues,
  method = "default",
  Kmax = 20,
  tolerance = 1,
  threshold = 0.9,
  verbose = FALSE
)

```

**Arguments**

eigenValues	The eigenvalues of the laplacian matrix.
method	The method that will be used. "default" to let the function choose the most suitable method. "PEV" for the Principal EigenValue method. "GAP" for the GAP method.
Kmax	The maximum number of cluster which is allowed.
tolerance	The tolerance allowed for the Principal EigenValue method.
threshold	The threshold to select the dominant eigenvalue for the GAP method.
verbose	To output the verbose in the terminal.

**Value**

a vector which contain the number of cluster to compute.

**Author(s)**

Emilie Poisson Caillault and Erwan Vincent

**Examples**

```
### Example 1: 2 disks of the same size
n<-100 ; r1<-1
x<-(runif(n)-0.5)*2;
y<-(runif(n)-0.5)*2
keep1<-which((x*2+y*2)<(r1*2))
disk1<-data.frame(x+3*r1,y)[keep1,]
disk2 <-data.frame(x-3*r1,y)[keep1,]
sameTwoDisks <- rbind(disk1,disk2)
W <- compute.similarity.ZP(scale(sameTwoDisks))
W <- checking.gram.similarityMatrix(W)
eigVal <- compute.laplacian.NJW(W,verbose = TRUE)$eigen$values
K <- compute.kclust(eigVal, method="default", Kmax=20, tolerance=0.99, threshold=0.9, verbose=TRUE)

### Example 2: Speed and Stopping Distances of Cars
W <- compute.similarity.ZP(scale(cars))
W <- checking.gram.similarityMatrix(W)
eigVal <- compute.laplacian.NJW(W,verbose = TRUE)$eigen$values
K <- compute.kclust(eigVal, method="default", Kmax=20, tolerance=0.99, threshold=0.9, verbose=TRUE)
```

---

compute.kclust2

*K clust compute selection V2*

---

**Description**

Function which select the number of cluster to compute thanks to a selected method

**Usage**

```
compute.kclust2(
  eigenValues,
  method = "default",
  Kmax = 20,
  tolerance = 1,
  threshold = 0.9,
  verbose = FALSE
)
```

**Arguments**

eigenValues	The eigenvalues of the laplacian matrix.
method	The method that will be used. "default" to let the function choose the most suitable method. "PEV" for the Principal EigenValue method. "GAP" for the GAP method.
Kmax	The maximum number of cluster which is allowed.
tolerance	The tolerance allowed for the Principal EigenValue method.
threshold	The threshold to select the dominant eigenvalue for the GAP method.
verbose	To output the verbose in the terminal.

**Value**

a vector which contain the number of cluster to compute.

**Author(s)**

Emilie Poisson Caillault and Erwan Vincent

---

compute.laplacian.NJW *Gram similarity matrix checker*

---

**Description**

Function which select the number of cluster to compute thanks to a selected method

**Usage**

```
compute.laplacian.NJW(W, verbose = FALSE)
```

**Arguments**

W	Gram Similarity Matrix.
verbose	To output the verbose in the terminal.

**Value**

returns a list containing the following elements:

- Lsym: a NJW laplacian matrix
- eigen: a list that contain the eigenvectors ans eigenvalues
- diag: a diagonal matrix used for the laplacian matrix

**Author(s)**

Emilie Poisson Caillault and Erwan Vincent

## Examples

```
### Example 1: 2 disks of the same size
n<-100 ; r1<-1
x<-(runif(n)-0.5)*2;
y<-(runif(n)-0.5)*2
keep1<-which((x*2+y*2)<(r1*2))
disk1<-data.frame(x+3*r1,y)[keep1,]
disk2 <-data.frame(x-3*r1,y)[keep1,]
sameTwoDisks <- rbind(disk1,disk2)
W <- compute.similarity.ZP(scale(sameTwoDisks))
W <- checking.gram.similarityMatrix(W)
res <- compute.laplacian.NJW(W,verbose = TRUE)

### Example 2: Speed and Stopping Distances of Cars
W <- compute.similarity.ZP(scale(cars))
W <- checking.gram.similarityMatrix(W)
res <- compute.laplacian.NJW(W,verbose = TRUE)
```

---

compute.nbCluster.gap *Recherche du nb de cluster par selon le critere du gap*

---

## Description

Recherche du nb de cluster par selon le critere du gap

## Usage

```
compute.nbCluster.gap(val, seuil = 0, fig = FALSE)
```

## Arguments

val	#valeur propre d'une matrice de similarite
seuil	seuil
fig	booleen

## Value

Kli

## Author(s)

Emilie Poisson Caillault v13/10/2015

---

compute.similarity.gaussien  
*Calcule matrice de similarite gaussienn*

---

### **Description**

Calcule matrice de similarite gaussienn

### **Usage**

```
compute.similarity.gaussien(points, sigma)
```

### **Arguments**

points	matrice pointsxattributs
sigma	sigma

### **Value**

mat

### **Author(s)**

Emilie Poisson Caillault v13/10/2015

---

compute.similarity.ZP *Calcule matrice de similarite gaussienne selon Zelnik-Manor et Perona*

---

### **Description**

sigma local, attention risque matrice non semi-def positive

### **Usage**

```
compute.similarity.ZP(points, vois = 7)
```

### **Arguments**

points	matrice pointsxattributs
vois	nombre de voisin qui seront selectionnes

### **Value**

mat

**Author(s)**

Emilie Poisson Caillault v13/10/2015

---

fastClustering      *Fast Spectral Clustering*

---

**Description**

This function will sample the data before performing a classification function on the samples and then applying K nearest neighbours.

**Usage**

```
fastClustering(  
  dataframe,  
  smp1Point,  
  stopCriteria = 0.99,  
  neighbours = 7,  
  similarity = TRUE,  
  clustFunction,  
  ...  
)
```

**Arguments**

dataFrame	The dataframe.
smp1Point	maximum of sample number for reduction.
stopCriteria	criterion for minimizing intra-group distance and select final smp1Point.
neighbours	number of points that will be selected for the similarity computation.
similarity	if True, will use the similarity matrix for the clustering function.
clustFunction	the clustering function to apply on data.
...	additional arguments for the clustering function.

**Value**

returns a list containing the following elements:

- results: clustering results
- sample: dataframe containing the sample used
- quantLabels: quantization labels
- clustLabels: results labels
- kmeans: kmeans quantization results



**Author(s)**

Emilie Poisson Caillault and Erwan Vincent

**Examples**

```
### Example 1: 2 disks of the same size
n<-100 ; r1<-1
x<-(runif(n)-0.5)*2;
y<-(runif(n)-0.5)*2
keep1<-which((x*2+y*2)<(r1*2))
disk1<-data.frame(x+3*r1,y)[keep1,]
disk2 <-data.frame(x-3*r1,y)[keep1,]
sameTwoDisks <- rbind(disk1,disk2)
res <- fastClustering(scale(sameTwoDisks),smplPoint = 500,
                      stopCriteria = 0.99, neighbours = 7, similarity = TRUE,
                      clustFunction = UnnormalizedSC, K = 2)
plot(sameTwoDisks, col = as.factor(res$clustLabels))

### Example 2: Speed and Stopping Distances of Cars
res <- fastClustering(scale(iris[,-5]),smplPoint = 500,
                      stopCriteria = 0.99, neighbours = 7, similarity = TRUE,
                      clustFunction = spectralPAM, K = 3)
plot(iris, col = as.factor(res$clustLabels))
table(res$clustLabels,iris$Species)
```

---

fastMSC

*Fast Multi-Level Spectral Clustering*


---

**Description**

The function, for a given dataFrame, will separate the data using the Fast NJW clustering in several levels.

**Usage**

```
fastMSC(
  X,
  levelMax,
  silMin = 0.7,
  vois = 7,
  flagDiagZero = FALSE,
  method = "default",
  Kmax = 20,
  tolerance = 0.99,
  threshold = 0.7,
  minPoint = 7,
  verbose = FALSE
)
```

**Arguments**

X	The dataFrame.
levelMax	The maximum depth level.
silMin	The minimal silhouette allowed. Below this value, the cluster will be cut again.
vois	number of points that will be selected for the similarity computation.
flagDiagZero	if True, Put zero on the similarity matrix W.
method	The method that will be used. "default" to let the function choose the most suitable method. "PEV" for the Principal EigenValue method. "GAP" for the GAP method.
Kmax	The maximum number of cluster which is allowed.
tolerance	The tolerance allowed for the Principal EigenValue method.
threshold	The threshold to select the dominant eigenvalue for the GAP method.
minPoint	The minimum number of points required to compute a cluster.
verbose	To output the verbose in the terminal.

**Value**

a dataframe containing the results labels of each levels

**Author(s)**

Emilie Poisson Caillault and Erwan Vincent

**Examples**

```
### Example 1: 2 disks of the same size
n<-100 ; r1<-1
x<-(runif(n)-0.5)*2;
y<-(runif(n)-0.5)*2
keep1<-which((x*2+y*2)<(r1*2))
disk1<-data.frame(x+3*r1,y)[keep1,]
disk2 <-data.frame(x-3*r1,y)[keep1,]
sameTwoDisks <- rbind(disk1,disk2)
res <- fastMSC(scale(sameTwoDisks),levelMax=5, silMin=0.7, vois=7,
               flagDiagZero=TRUE, method = "PEV", Kmax = 20,
               tolerance = 0.99,threshold = 0.7, minPoint = 7, verbose = TRUE)
plot(sameTwoDisks, col = as.factor(res[,ncol(res)]))

### Example 2: Speed and Stopping Distances of Cars
res <- fastMSC(scale(iris[,-5]),levelMax=5, silMin=0.7, vois=7,
               flagDiagZero=TRUE, method = "PEV", Kmax = 20,
               tolerance = 0.99,threshold = 0.9, minPoint = 7, verbose = TRUE)
plot(iris, col = as.factor(res[,ncol(res)]))
table(res[,ncol(res)],iris$Species)
```

**Description**

Hierarchical Clustering

**Usage**

```
HierarchicalClust(  
  W,  
  K = 5,  
  method = "ward.D2",  
  flagDiagZero = FALSE,  
  verbose = FALSE,  
  ...  
)
```

**Arguments**

W	Gram Similarity Matrix.
K	number of cluster to obtain.
method	method that will be used in the hierarchical clustering.
flagDiagZero	if True, Put zero on the similarity matrix W.
verbose	To output the verbose in the terminal.
...	Additional parameter for the hclust function.

**Value**

returns a list containing the following elements:

- cluster: a vector containing the cluster

**Author(s)**

Emilie Poisson Caillault and Erwan Vincent

**Examples**

```
### Example 1: 2 disks of the same size  
n<-100 ; r1<-1  
x<-(runif(n)-0.5)*2;  
y<-(runif(n)-0.5)*2  
keep1<-which((x*2+y*2)<(r1*2))  
disk1<-data.frame(x+3*r1,y)[keep1,]  
disk2 <-data.frame(x-3*r1,y)[keep1,]  
sameTwoDisks <- rbind(disk1,disk2)
```

```
W <- compute.similarity.ZP(scale(sameTwoDisks))
res <- HierarchicalClust(W,K=2,method="ward.D2",flagDiagZero=TRUE,verbose=TRUE)
plot(sameTwoDisks, col = res$cluster)

### Example 2: Speed and Stopping Distances of Cars
W <- compute.similarity.ZP(scale(iris[,-5]))
res <- HierarchicalClust(W,K=2,method="ward.D2",flagDiagZero=TRUE,verbose=TRUE)
plot(iris, col = res$cluster)
```

---

HierarchicalSC

*Hierarchical Spectral Clustering*

---

## Description

Hierarchical Spectral Clustering

## Usage

```
HierarchicalSC(
  W,
  K = 5,
  method = "ward.D2",
  flagDiagZero = FALSE,
  verbose = FALSE
)
```

## Arguments

W	Gram Similarity Matrix.
K	number of cluster to obtain.
method	method that will be used in the hierarchical clustering.
flagDiagZero	if True, Put zero on the similarity matrix W.
verbose	To output the verbose in the terminal.

## Value

returns a list containing the following elements:

- cluster: a vector containing the cluster
- eigenVect: a vector containing the eigenvectors
- eigenVal: a vector containing the eigenvalues

## Author(s)

Emilie Poisson Caillault and Erwan Vincent

## References

Sanchez-Garcia, R., Fernnelly, M. and al. (2014). Hierarchical Spectral Clustering of Power Grids. In IEEE Transaction on Power Systems 29.5, pages 2229-2237. ISSN : 0885-8950.

## Examples

```
### Example 1: 2 disks of the same size
n<-100 ; r1<-1
x<-(runif(n)-0.5)*2;
y<-(runif(n)-0.5)*2
keep1<-which((x*2+y*2)<(r1*2))
disk1<-data.frame(x+3*r1,y)[keep1,]
disk2 <-data.frame(x-3*r1,y)[keep1,]
sameTwoDisks <- rbind(disk1,disk2)
W <- compute.similarity.ZP(scale(sameTwoDisks))
res <- HierarchicalSC(W,K=2,method = "ward.D2",flagDiagZero=TRUE,verbose=TRUE)
plot(sameTwoDisks, col = res$cluster)
plot(res$eigenVect[,1:2], col = res$cluster, main="spectral space",
      xlim=c(-1,1),ylim=c(-1,1)); points(0,0,pch='+');
plot(res$eigenVal, main="Laplacian eigenvalues",pch='+');

### Example 2: Speed and Stopping Distances of Cars
W <- compute.similarity.ZP(scale(iris[,-5]))
res <- HierarchicalSC(W,K=2,method="ward.D2",flagDiagZero=TRUE,verbose=TRUE)
plot(iris, col = res$cluster)
plot(res$eigenVect[,1:2], col = res$cluster, main="spectral space",
      xlim=c(-1,1),ylim=c(-1,1)); points(0,0,pch='+');
plot(res$eigenVal, main="Laplacian eigenvalues",pch='+');
```

---

kmeansQuantization      *Data quantization*

---

## Description

The function use kmeans algorithm to perform data quantization.

## Usage

```
kmeansQuantization(dataFrame, maxData, stopCriteria = 0.99)
```

## Arguments

dataFrame	The dataFrame.
maxData	maximum of sample number for reduction.
stopCriteria	criterion for minimizing intra-group distance and select final smplPoint.

## Value

kmeans result

**Author(s)**

Emilie Poisson Caillault and Erwan Vincent

---

MSC

*Multi-Level Spectral Clustering*

---

**Description**

The function, for a given `dataFrame`, will separate the data using the NJW clustering in several levels.

**Usage**

```
MSC(  
  X,  
  levelMax,  
  silMin = 0.7,  
  vois = 7,  
  flagDiagZero = FALSE,  
  method = "default",  
  Kmax = 20,  
  tolerance = 0.99,  
  threshold = 0.7,  
  minPoint = 7,  
  verbose = FALSE  
)
```

**Arguments**

<code>X</code>	The <code>dataFrame</code> .
<code>levelMax</code>	The maximum depth level.
<code>silMin</code>	The minimal silhouette allowed. Below this value, the cluster will be cut again.
<code>vois</code>	number of points that will be selected for the similarity computation.
<code>flagDiagZero</code>	if True, Put zero on the similarity matrix <code>W</code> .
<code>method</code>	The method that will be used. "default" to let the function choose the most suitable method. "PEV" for the Principal EigenValue method. "GAP" for the GAP method.
<code>Kmax</code>	The maximum number of cluster which is allowed.
<code>tolerance</code>	The tolerance allowed for the Principal EigenValue method.
<code>threshold</code>	The threshold to select the dominant eigenvalue for the GAP method.
<code>minPoint</code>	The minimum number of points required to compute a cluster.
<code>verbose</code>	To output the verbose in the terminal.

**Value**

returns a list containing the following elements:

- cluster: a vector containing the cluster
- eigenVect: a vector containing the eigenvectors
- eigenVal: a vector containing the eigenvalues

**Author(s)**

Emilie Poisson Caillault and Erwan Vincent

**References**

Grassi, K. (2020) Definition multivariee et multi-echelle d'etats environnementaux par Machine Learning : Caracterisation de la dynamique phytoplantonique.

**Examples**

```
### Example 1: 2 disks of the same size
n<-100 ; r1<-1
x<-(runif(n)-0.5)*2;
y<-(runif(n)-0.5)*2
keep1<-which((x*2+y*2)<(r1*2))
disk1<-data.frame(x+3*r1,y)[keep1,]
disk2 <-data.frame(x-3*r1,y)[keep1,]
sameTwoDisks <- rbind(disk1,disk2)
res <- MSC(scale(sameTwoDisks),levelMax=5, silMin=0.7, vois=7,
           flagDiagZero=TRUE, method = "default", Kmax = 20,
           tolerance = 0.99,threshold = 0.7, minPoint = 7, verbose = TRUE)
plot(sameTwoDisks, col = as.factor(res[,ncol(res)]))

### Example 2: Speed and Stopping Distances of Cars
res <- MSC(scale(iris[,-5]),levelMax=5, silMin=0.7, vois=7,
           flagDiagZero=TRUE, method = "default", Kmax = 20,
           tolerance = 0.99,threshold = 0.9, minPoint = 7, verbose = TRUE)
plot(iris, col = as.factor(res[,ncol(res)]))
table(res[,ncol(res)],iris$Species)
```

---

PeronaFreemanSC

*Bi-parted Spectral Clustering. Peronna and Freeman.*

---

**Description**

Bi-parted spectral clustering based on Peronna and Freeman algorithm, which separates the data into two distinct clusters

**Usage**

```
PeronaFreemanSC(W, flagDiagZero = FALSE, verbose = FALSE)
```

**Arguments**

W                    Gram Similarity Matrix.  
 flagDiagZero      if True, Put zero on the similarity matrix W.  
 verbose            To output the verbose in the terminal.

**Value**

returns a list containing the following elements:

- cluster: a vector containing the cluster
- eigenVect: a vector containing the eigenvectors
- eigenVal: a vector containing the eigenvalues

**Author(s)**

Emilie Poisson Caillault and Erwan Vincent

**References**

Perona, P. and Freeman, W. (1998). A factorization approach to grouping. In European Conference on Computer Vision, pages 655-670

**Examples**

```
### Example 1: 2 disks of the same size
n<-100 ; r1<-1
x<-(runif(n)-0.5)*2;
y<-(runif(n)-0.5)*2
keep1<-which((x*2+y*2)<(r1*2))
disk1<-data.frame(x+3*r1,y)[keep1,]
disk2 <-data.frame(x-3*r1,y)[keep1,]
sameTwoDisks <- rbind(disk1,disk2)
W <- compute.similarity.ZP(scale(sameTwoDisks))
res <- PeronaFreemanSC(W,flagDiagZero=TRUE,verbose=TRUE)
plot(sameTwoDisks, col = res$cluster)
plot(res$eigenVect[,1:2], col = res$cluster, main="spectral space",
      xlim=c(-1,1),ylim=c(-1,1)); points(0,0,pch='+');
plot(res$eigenVal, main="Laplacian eigenvalues",pch='+');
```

```
### Example 2: Speed and Stopping Distances of Cars
W <- compute.similarity.ZP(scale(iris[,-5]))
res <- PeronaFreemanSC(W,flagDiagZero=TRUE,verbose=TRUE)
plot(iris, col = res$cluster)
plot(res$eigenVect[,1:2], col = res$cluster, main="spectral space",
      xlim=c(-1,1),ylim=c(-1,1)); points(0,0,pch='+');
plot(res$eigenVal, main="Laplacian eigenvalues",pch='+');
```



---

recursClust	<i>Perform a multi level clustering</i>
-------------	---

---

### Description

The function, for a given dataFrame, will separate the data using the input clustering method in several levels.

### Usage

```
recursClust(
  dataframe,
  levelMax = 2,
  clustFunction,
  similarity = TRUE,
  vois = 7,
  flagDiagZero = FALSE,
  biparted = FALSE,
  method = "default",
  tolerance = 0.99,
  threshold = 0.9,
  minPoint = 7,
  verbose = FALSE,
  ...
)
```

### Arguments

dataFrame	The dataFrame.
levelMax	The maximum depth level.
clustFunction	the clustering function to apply on data.
similarity	if True, will use the similarity matrix for the clustering function.
vois	number of points that will be selected for the similarity computation.
flagDiagZero	if True, Put zero on the similarity matrix W.
biparted	if True, the function will not automatically choose the number of clusters to compute.
method	The method that will be used. "default" to let the function choose the most suitable method. "PEV" for the Principal EigenValue method. "GAP" for the GAP method.
tolerance	The tolerance allowed for the Principal EigenValue method.
threshold	The threshold to select the dominant eigenvalue for the GAP method.
minPoint	The minimum number of points required to compute a cluster.
verbose	To output the verbose in the terminal.
...	additional arguments for the clustering function.

**Value**

returns a list containing the following elements:

- cluster: vector that contain the result of the last level
- allLevels: dataframe containing the clustering results of each levels
- nbLevels: the number of computed levels

**Author(s)**

Emilie Poisson Caillault and Erwan Vincent

**Examples**

```
### Example 1: 2 disks of the same size
n<-100 ; r1<-1
x<-(runif(n)-0.5)*2;
y<-(runif(n)-0.5)*2
keep1<-which((x*2+y*2)<(r1*2))
disk1<-data.frame(x+3*r1,y)[keep1,]
disk2 <-data.frame(x-3*r1,y)[keep1,]
sameTwoDisks <- rbind(disk1,disk2)
res <- recursClust(scale(sameTwoDisks),levelMax=3, clustFunction =ShiMalikSC,
                    similarity = TRUE, vois = 7, flagDiagZero = FALSE,
                    biparted = TRUE, verbose = TRUE)
plot(sameTwoDisks, col = as.factor(res$cluster))

### Example 2: Speed and Stopping Distances of Cars
res <- recursClust(scale(iris[,-5]),levelMax=4, clustFunction = spectralPAM,
                    similarity = TRUE, vois = 7, flagDiagZero = FALSE,
                    biparted = FALSE, method = "PEV", tolerance = 0.99,
                    threshold = 0.9, verbose = TRUE)
plot(iris, col = as.factor(res$cluster))
```

---

search.neighbor

*Recherche du voisin num id le plus proche*

---

**Description**

Recherche du voisin num id le plus proche

**Usage**

```
search.neighbor(vdist, vois)
```

**Arguments**

vdist                    vecteur de distance du point avec d'autres points  
vois                      nombre de voisin a selectionner

**Value**

id

**Author(s)**

Emilie Poisson Caillault v13/10/2015

---

ShiMalikSC*Bi-parted Spectral Clustering. Shi and Malik.*

---

**Description**

Bi-parted spectral clustering based on Shi and Malik algorithm, which separates the data into two distinct clusters

**Usage**

```
ShiMalikSC(W, flagDiagZero = FALSE, verbose = FALSE)
```

**Arguments**

W	Gram Similarity Matrix.
flagDiagZero	if True, Put zero on the similarity matrix W.
verbose	To output the verbose in the terminal.

**Value**

returns a list containing the following elements:

- cluster: a vector containing the cluster
- eigenVect: a vector containing the eigenvectors
- eigenVal: a vector containing the eigenvalues

**Author(s)**

Emilie Poisson Caillault and Erwan Vincent

**References**

Shi, J and Malik, J. (2000). Normalized cuts and image segmentation. In PAMI, Transactions on Pattern Analysis and Machine Intelligence, pages 888-905

## Examples

```
### Example 1: 2 disks of the same size
n<-100 ; r1<-1
x<-(runif(n)-0.5)*2;
y<-(runif(n)-0.5)*2
keep1<-which((x*2+y*2)<(r1*2))
disk1<-data.frame(x+3*r1,y)[keep1,]
disk2 <-data.frame(x-3*r1,y)[keep1,]
sameTwoDisks <- rbind(disk1,disk2)
W <- compute.similarity.ZP(scale(sameTwoDisks))
res <- ShiMalikSC(W,flagDiagZero=TRUE,verbose=FALSE)
plot(sameTwoDisks, col = res$cluster)
plot(res$eigenVect[,1:2], col = res$cluster, main="spectral space",
      xlim=c(-1,1),ylim=c(-1,1)); points(0,0,pch='+');
plot(res$eigenVal, main="Laplacian eigenvalues",pch='+');

### Example 2: Speed and Stopping Distances of Cars
W <- compute.similarity.ZP(scale(iris[,-5]))
res <- ShiMalikSC(W,flagDiagZero=TRUE,verbose=TRUE)
plot(iris, col = res$cluster)
plot(res$eigenVect[,1:2], col = res$cluster, main="spectral space",
      xlim=c(-1,1),ylim=c(-1,1)); points(0,0,pch='+');
plot(res$eigenVal, main="Laplacian eigenvalues",pch='+');
```

---

spectralPAM

*Spectral-PAM clustering*

---

## Description

The function, for a given similarity matrix, will separate the data using a spectral space. It is based on the Jordan and Weiss algorithm. This version uses K-medoid to split the clusters.

## Usage

```
spectralPAM(W, K, flagDiagZero = FALSE, verbose = FALSE)
```

## Arguments

W	Gram Similarity Matrix.
K	number of cluster to obtain.
flagDiagZero	if True, Put zero on the similarity matrix W.
verbose	To output the verbose in the terminal.

## Value

returns a list containing the following elements:

- cluster: a vector containing the cluster
- eigenVect: a vector containing the eigenvectors
- eigenVal: a vector containing the eigenvalues

**Author(s)**

Emilie Poisson Caillault and Erwan Vincent

**Examples**

```
### Example 1: 2 disks of the same size
n<-100 ; r1<-1
x<-(runif(n)-0.5)*2;
y<-(runif(n)-0.5)*2
keep1<-which((x*2+y*2)<(r1*2))
disk1<-data.frame(x+3*r1,y)[keep1,]
disk2 <-data.frame(x-3*r1,y)[keep1,]
sameTwoDisks <- rbind(disk1,disk2)
W <- compute.similarity.ZP(scale(sameTwoDisks))
res <- spectralPAM(W,K=2,flagDiagZero=TRUE,verbose=TRUE)
plot(sameTwoDisks, col = res$cluster)
plot(res$eigenVect[,1:2], col = res$cluster, main="spectral space",
      xlim=c(-1,1),ylim=c(-1,1)); points(0,0,pch='+');
plot(res$eigenVal, main="Laplacian eigenvalues",pch='+');
abline(h=1,lty="dashed",col="red")

### Example 2: Speed and Stopping Distances of Cars
W <- compute.similarity.ZP(scale(iris[-5]))
res <- spectralPAM(W,K=2,flagDiagZero=TRUE,verbose=TRUE)
plot(iris, col = res$cluster)
plot(res$eigenVect[,1:2], col = res$cluster, main="spectral space",
      xlim=c(-1,1),ylim=c(-1,1)); points(0,0,pch='+');
plot(res$eigenVal, main="Laplacian eigenvalues",pch='+');
abline(h=1,lty="dashed",col="red")
```

---

 UnnormalizedSC

*Unnormalized Spectral Clustering Ng.*


---

**Description**

The function, for a given similarity matrix, will separate the data using a spectral space. It does not normalize the Laplacian matrix compared to other algorithms

**Usage**

```
UnnormalizedSC(W, K = 5, flagDiagZero = FALSE, verbose = FALSE)
```

**Arguments**

W	Gram Similarity Matrix.
K	number of cluster to obtain.
flagDiagZero	if True, Put zero on the similarity matrix W.
verbose	To output the verbose in the terminal.

**Value**

returns a list containing the following elements:

- cluster: a vector containing the cluster
- eigenVect: a vector containing the eigenvectors
- eigenVal: a vector containing the eigenvalues

**Author(s)**

Emilie Poisson Caillaud and Erwan Vincent

**Examples**

```
### Example 1: 2 disks of the same size
n<-100 ; r1<-1
x<-(runif(n)-0.5)*2;
y<-(runif(n)-0.5)*2
keep1<-which((x*2+y*2)<(r1*2))
disk1<-data.frame(x+3*r1,y)[keep1,]
disk2 <-data.frame(x-3*r1,y)[keep1,]
sameTwoDisks <- rbind(disk1,disk2)
W <- compute.similarity.ZP(scale(sameTwoDisks))
res <- UnormalizedSC(W,K=2,flagDiagZero=TRUE,verbose=TRUE)
plot(sameTwoDisks, col = res$cluster)
plot(res$eigenVect[,1:2], col = res$cluster, main="spectral space",
      xlim=c(-1,1),ylim=c(-1,1)); points(0,0,pch='+');
plot(res$eigenVal, main="Laplacian eigenvalues",pch='+');

### Example 2: Speed and Stopping Distances of Cars
W <- compute.similarity.ZP(scale(iris[,-5]))
res <- UnormalizedSC(W,K=2,flagDiagZero=TRUE,verbose=TRUE)
plot(iris, col = res$cluster)
plot(res$eigenVect[,1:2], col = res$cluster, main="spectral space",
      xlim=c(-1,1),ylim=c(-1,1)); points(0,0,pch='+');
plot(res$eigenVal, main="Laplacian eigenvalues",pch='+');
```

**Description**

The function, for a given similarity matrix, will separate the data using a spectral space. It uses the Von Luxburg algorithm to do this

**Usage**

```
VonLuxburgSC(W, K = 5, flagDiagZero = FALSE, verbose = FALSE)
```

**Arguments**

W	Gram Similarity Matrix.
K	number of cluster to obtain.
flagDiagZero	if True, Put zero on the similarity matrix W.
verbose	To output the verbose in the terminal.

**Value**

returns a list containing the following elements:

- cluster: a vector containing the cluster
- eigenVect: a vector containing the eigenvectors
- eigenVal: a vector containing the eigenvalues

**Author(s)**

Emilie Poisson Caillault and Erwan Vincent

**References**

Von Luxburg, U. (2007). A Tutorial on Spectral Clustering. *Statistics and Computing*, Volume 17(4), pages 395-416

**Examples**

```
### Example 1: 2 disks of the same size
n<-100 ; r1<-1
x<-(runif(n)-0.5)*2;
y<-(runif(n)-0.5)*2
keep1<-which((x*2+y*2)<(r1*2))
disk1<-data.frame(x+3*r1,y)[keep1,]
disk2 <-data.frame(x-3*r1,y)[keep1,]
sameTwoDisks <- rbind(disk1,disk2)
W <- compute.similarity.ZP(scale(sameTwoDisks))
res <- VonLuxburgSC(W,K=2,flagDiagZero=TRUE,verbose=TRUE)
plot(sameTwoDisks, col = res$cluster)
plot(res$eigenVect[,1:2], col = res$cluster, main="spectral space",
      xlim=c(-1,1),ylim=c(-1,1)); points(0,0,pch='+');
plot(res$eigenVal, main="Laplacian eigenvalues",pch='+');

### Example 2: Speed and Stopping Distances of Cars
W <- compute.similarity.ZP(scale(iris[,-5]))
res <- VonLuxburgSC(W,K=2,flagDiagZero=TRUE,verbose=TRUE)
plot(iris, col = res$cluster)
plot(res$eigenVect[,1:2], col = res$cluster, main="spectral space",
      xlim=c(-1,1),ylim=c(-1,1)); points(0,0,pch='+');
plot(res$eigenVal, main="Laplacian eigenvalues",pch='+');
```

# Index

checking.gram.similarityMatrix, [2](#)  
compute.kclust, [3](#)  
compute.kclust2, [4](#)  
compute.laplacian.NJW, [5](#)  
compute.nbCluster.gap, [6](#)  
compute.similarity.gaussien, [7](#)  
compute.similarity.ZP, [7](#)

fastClustering, [8](#)  
fastMSC, [9](#)

HierarchicalClust, [11](#)  
HierarchicalSC, [12](#)

kmeansQuantization, [13](#)

MSC, [14](#)

PeronaFreemanSC, [15](#)

recursClust, [17](#)

search.neighbor, [18](#)  
ShiMalikSC, [19](#)  
spectralPAM, [20](#)

UnnormalizedSC, [21](#)

VonLuxburgSC, [22](#)