Package ‘SNFtool’

April 24, 2018

Type Package

Title Similarity Network Fusion

Version 2.3.0

Date 2018-04-24

Author Bo Wang, Aziz Mezlini, Feyyaz Demir, Marc Fiume, Zhuowen Tu, Michael Brudno, Benjamin Haibe-Kains, Anna Goldenberg

Maintainer Daniel Cole <goldenberglab@gmail.com>

Imports heatmap.plus, ExPosition, alluvial

Description Similarity Network Fusion takes multiple views of a network and fuses them together to construct an overall status matrix. The input to our algorithm can be feature vectors, pairwise distances, or pairwise similarities. The learned status matrix can then be used for retrieval, clustering, and classification.

License GPL

NeedsCompilation no

Repository CRAN

Date/Publication 2018-04-24 16:23:07 UTC

R topics documented:

`affinityMatrix` *Affinity matrix calculation*

Description

Computes affinity matrix from a generic distance matrix

Usage

```
affinityMatrix(diff, K = 20, sigma = 0.5)
```

Arguments

`diff`	Distance matrix
`K`	Number of nearest neighbors
`sigma`	Variance for local model

Value

Returns an affinity matrix that represents the neighborhood graph of the data points.

Author(s)

Dr. Anna Goldenberg, Bo Wang, Aziz Mezlini, Feyyaz Demir

References

B Wang, A Mezlini, F Demir, M Fiume, T Zu, M Brudno, B Haibe-Kains, A Goldenberg (2014) Similarity Network Fusion: a fast and effective method to aggregate multiple data types on a genome wide scale. Nature Methods. Online. Jan 26, 2014

Examples

```
## First, set all the parameters:
K = 20; ##number of neighbors, must be greater than 1. usually (10~30)
alpha = 0.5; ##hyperparameter, usually (0.3~0.8)
T = 20; ###Number of Iterations, usually (10~50)

## Data1 is of size n x d_1,
## where n is the number of patients, d_1 is the number of genes,
## Data2 is of size n x d_2,
## where n is the number of patients, d_2 is the number of methylation
data(Data1)
data(Data2)

## Calculate distance matrices(here we calculate Euclidean Distance,
## you can use other distance, e.g. correlation)
Dist1 = (dist2(as.matrix(Data1),as.matrix(Data1)))^(1/2)
Dist2 = (dist2(as.matrix(Data2),as.matrix(Data2)))^(1/2)

## Next, construct similarity graphs
W1 = affinityMatrix(Dist1, K, alpha)
W2 = affinityMatrix(Dist2, K, alpha)
```

`calNMI` *Mutual Information calculation*

Description

Calculate the mutual information between vectors x and y.

Usage

```
calNMI(x, y)
```

Arguments

`x`	a vector
`y`	a vector

Value

Returns the mutual information between vectors x and y.

Author(s)

Dr. Anna Goldenberg, Bo Wang, Aziz Mezlini, Feyyaz Demir

References

B Wang, A Mezlini, F Demir, M Fiume, T Zu, M Brudno, B Haibe-Kains, A Goldenberg (2014) Similarity Network Fusion: a fast and effective method to aggregate multiple data types on a genome wide scale. Nature Methods. Online. Jan 26, 2014

Examples

```
# How to use SNF with multiple views

# Load views into list "dataL"
data(dataL)
data(label)

# Set the other parameters
K = 20 # number of neighbours
alpha = 0.5 # hyperparameter in affinityMatrix
T = 20 # number of iterations of SNF

# Normalize the features in each of the views if necessary
# dataL = lapply(dataL, standardNormalization)

# Calculate the distances for each view
distL = lapply(dataL, function(x) (dist2(x, x))^(1/2))

# Construct the similarity graphs
affinityL = lapply(distL, function(x) affinityMatrix(x, K, alpha))

# Example of how to use SNF to perform subtyping
# Construct the fused network
W = SNF(affinityL, K, T)
# Perform clustering on the fused network.
clustering = spectralClustering(W,3);
# Use NMI to measure the goodness of the obtained labels.
NMI = calNMI(clustering,label);
```

`chiDist2` *Pairwise Chi-squared distances*

Description

Wrapper function chi2Dist imported from 'ExPosition' package. Computes the Chi-squared distances between all pairs of data point given

Usage

```
chiDist2(A)
```

Arguments

`A`	A data matrix where each row is a different data point

Value

Returns an N x N matrix where N is the number of rows in X. element (i,j) is the squared Chi-squared distance between ith data point in X and jth data point in X.

Author(s)

Dr. Anna Goldenberg, Bo Wang, Aziz Mezlini, Feyyaz Demir

Examples

```
## Data1 is of size n x d_1,
## where n is the number of patients, d_1 is the number of genes,
## Data2 is of size n x d_2,
## where n is the number of patients, d_2 is the number of methylation
data(Data1)
data(Data2)

## Calculate distance matrices(here we calculate Euclidean Distance,
## you can use other distance, e.g. correlation)
Dist1 = chiDist2(as.matrix(Data1))
Dist2 = chiDist2(as.matrix(Data2))
```

`concordanceNetworkNMI` *Concordance Network NMI calculation*

Description

Given a list of affinity matrices, Wall, the number of clusters, return a matrix containing the NMIs between cluster assignments made with spectral clustering on all matrices provided.

Usage

```
concordanceNetworkNMI(Wall, C)
```

Arguments

`Wall`	List of matrices. Each element of the list is a square, symmetric matrix that shows affinities of the data points from a certain view.
`C`	Number of clusters

Value

Returns an affinity matrix that represents the neighborhood graph of the data points.

Author(s)

Dr. Anna Goldenberg, Bo Wang, Aziz Mezlini, Feyyaz Demir

Examples

```
# How to use SNF with multiple views

# Load views into list "dataL"
data(dataL)
data(label)

# Set the other parameters
K = 20 # number of neighbours
alpha = 0.5 # hyperparameter in affinityMatrix
T = 20 # number of iterations of SNF
# Normalize the features in each of the views.
#dataL = lapply(dataL, standardNormalization)

# Calculate the distances for each view
distL = lapply(dataL, function(x) (dist2(x, x)^(1/2)))

# Construct the similarity graphs
affinityL = lapply(distL, function(x) affinityMatrix(x, K, alpha))

# an example of how to use concordanceNetworkNMI
Concordance_matrix = concordanceNetworkNMI(affinityL, 3);

## The output, Concordance_matrix,
## shows the concordance between the fused network and each individual network.
```

`Data1` *Data1*

Description

Data1 dataset used to demonstrate the use of SNFtool.

Usage

```
data(Data1)
```

Format

A data frame with 200 observations on the following 2 variables.

`V1` a numeric vector

`V2` a numeric vector

Examples

```
data(Data1)
```

`Data2` *Data2*

Description

Data2 dataset used to demonstrate the use of SNFtool.

Usage

```
data(Data2)
```

Format

A data frame with 200 observations on the following 2 variables.

`V3` a numeric vector

`V4` a numeric vector

Examples

```
data(Data2)
```

`dataL` *dataL*

Description

Dataset used to provide an example of predicting the new labels with label propagation.

Usage

```
data(dataL)
```

Format

The format is: List of 2 $: num [1:600, 1:76] 0.0659 0.0491 0.0342 0.0623 0.062 ... ..- attr(*, "dimnames")=List of 2 .. ..$: chr [1:600] "V1" "V2" "V3" "V4" ... .$: NULL $: int [1:600, 1:240] 0 0 0 0 0 0 0 0 0 0 ... ..- attr(*, "dimnames")=List of 2 .. ..$: chr [1:600] "V1" "V2" "V3" "V4" ... .$: NULL

Examples

```
data(dataL)
```

displayClusters *Plot given similarity matrix by clusters*

Description

Visualize the clusters in given similarity matrix

Usage

```
displayClusters(W, group)
```

Arguments

`W`	Similarity matrix
`group`	A vector containing the labels for each sample in W.

Value

Plots given similarity matrix with patients ordered to form clusters.

Author(s)

Dr. Anna Goldenberg, Bo Wang, Aziz Mezlini, Feyyaz Demir

Examples

```
## First, set all the parameters:
K = 20; # number of neighbors, usually (10~30)
alpha = 0.5;   # hyperparameter, usually (0.3~0.8)
T = 10;  # Number of Iterations, usually (10~20)

## Data1 is of size n x d_1,
## where n is the number of patients, d_1 is the number of genes,
## Data2 is of size n x d_2,
## where n is the number of patients, d_2 is the number of methylation
data(Data1)
data(Data2)

## Here, the simulation data (SNFdata) has two data types. They are complementary to each other.
## And two data types have the same number of points.
## The first half data belongs to the first cluster; the rest belongs to the second cluster.
truelabel = c(matrix(1,100,1),matrix(2,100,1)); ## the ground truth of the simulated data

## Calculate distance matrices
## (here we calculate Euclidean Distance, you can use other distance, e.g,correlation)

## If the data are all continuous values, we recommend the users to perform
## standard normalization before using SNF,
## though it is optional depending on the data the users want to use.
# Data1 = standardNormalization(Data1);
# Data2 = standardNormalization(Data2);


## Calculate the pair-wise distance;
## If the data is continuous, we recommend to use the function "dist2" as follows
Dist1 = (dist2(as.matrix(Data1),as.matrix(Data1)))^(1/2)
Dist2 = (dist2(as.matrix(Data2),as.matrix(Data2)))^(1/2)

## next, construct similarity graphs
W1 = affinityMatrix(Dist1, K, alpha)
W2 = affinityMatrix(Dist2, K, alpha)

## These similarity graphs have complementary information about clusters.
displayClusters(W1, truelabel);
displayClusters(W2, truelabel);
```

---

displayClustersWithHeatmap

Display the similarity matrix by clusters with some sample information

---

Description

Visualize the clusters present in the given similarity matrix as well as some sample information.

Usage

```
displayClustersWithHeatmap(W, group, ColSideColors=NULL, ...)
```

Arguments

`W`	Similarity matrix
`group`	A numeric vector containing the groups information for each sample in W such as the result of the spectralClustering function. The order should correspond to the sample order in W.
`ColSideColors`	(optional) character vector of length ncol(x) containing the color names for a horizontal side bar that may be used to annotate the columns of x, used by the heatmap function, OR a character matrix with number of rows matching number of rows in x. Each column is plotted as a row similar to heatmap()'s ColSide-Colors by the heatmap.plus function.
`...`	other paramater that can be pass on to the heatmap (if ColSideColor is a NULL or a vector) or heatmap.plus function (if ColSideColors is matrix)

Details

Using the heatmap or heatmap.plus function to display the similarity matrix For representation purpose, the similarity matrix diagonal is set to the median value of W, the matrix is normalised and W = W + t(W) is applied In this presentation no clustering method is ran the samples are ordered in function of their group label present in the group arguments.

Value

Plots the similarity matrix using the heatmap function. Samples are ordered by the clusters provided by the argument groups with sample information displayed with a color bar if the ColSideColors argument is informed.

Author(s)

Florence Cavalli

Examples

```
## First, set all the parameters:
K = 20;    # number of neighbors, usually (10~30)
alpha = 0.5;    # hyperparameter, usually (0.3~0.8)
T = 20;   # Number of Iterations, usually (10~20)

## Data1 is of size n x d_1,
## where n is the number of patients, d_1 is the number of genes,
## Data2 is of size n x d_2,
## where n is the number of patients, d_2 is the number of methylation
data(Data1)
data(Data2)

## Here, the simulation data (SNFdata) has two data types. They are complementary to each other.
## And two data types have the same number of points.
## The first half data belongs to the first cluster; the rest belongs to the second cluster.
truelabel = c(matrix(1,100,1),matrix(2,100,1)); ## the ground truth of the simulated data

## Calculate distance matrices
## (here we calculate Euclidean Distance, you can use other distance, e.g,correlation)

## If the data are all continuous values, we recommend the users to perform
## standard normalization before using SNF,
## though it is optional depending on the data the users want to use.
# Data1 = standardNormalization(Data1);
# Data2 = standardNormalization(Data2);

## Calculate the pair-wise distance;
## If the data is continuous, we recommend to use the function "dist2" as follows
Dist1 = (dist2(as.matrix(Data1),as.matrix(Data1)))^(1/2)
Dist2 = (dist2(as.matrix(Data2),as.matrix(Data2)))^(1/2)

## next, construct similarity graphs
W1 = affinityMatrix(Dist1, K, alpha)
```

```
W2 = affinityMatrix(Dist2, K, alpha)

## next, we fuse all the graphs
## then the overall matrix can be computed by similarity network fusion(SNF):
W = SNF(list(W1,W2), K, T)

## With this unified graph W of size n x n,
## you can do either spectral clustering or Kernel NMF.
## If you need help with further clustering, please let us know.

## You can display clusters in the data by the following function
## where C is the number of clusters.
C = 2    # number of clusters
group = spectralClustering(W,C);  # the final subtypes information

## Get a matrix containing the group information
## for the samples such as the SpectralClustering result and the True label
M_label=cbind(group,truelabel)
colnames(M_label)=c("spectralClustering","TrueLabel")

## ****
## Comments
## rownames(M_label)=names(spectralClustering) To add if the spectralClustering function
## pass the sample ID as names.
## or rownames(M_label)=rownames(W) Having W with rownames and colmanes
## with smaple ID would help as well.
## ***

## Use the getColorsForGroups function to assign a color to each group
## NB is more than 8 groups, you will have to input a vector
## of colors into the getColorsForGroups function
M_label_colors=t(apply(M_label,1,getColorsForGroups))
## or choose you own colors for each label, for example:
M_label_colors=cbind("spectralClustering"=getColorsForGroups(M_label[,"spectralClustering"],
colors=c("blue","green")),"TrueLabel"=getColorsForGroups(M_label[,"TrueLabel"],
colors=c("orange","cyan")))

## Visualize the clusters present in the given similarity matrix
## as well as some sample information
## In this presentation no clustering method is ran the samples
## are ordered in function of their group label present in the group arguments
displayClustersWithHeatmap(W, group, M_label_colors[,"spectralClustering"])
displayClustersWithHeatmap(W, group, M_label_colors)
```

dist2 *Pairwise squared Euclidean distances*

Description

Computes the squared Euclidean distances between all pairs of data point given

Usage

```
dist2(X, C)
```

Arguments

`X`	A data matrix where each row is a different data point
`C`	A data matrix where each row is a different data point. If this matrix is the same as X, pairwise distances for all data points are computed.

Value

Returns an N x M matrix where N is the number of rows in X and M is the number of rows in M. element (n,m) is the squared Euclidean distance between nth data point in X and mth data point in C

Author(s)

Dr. Anna Goldenberg, Bo Wang, Aziz Mezlini, Feyyaz Demir

Examples

```
## Data1 is of size n x d_1,
## where n is the number of patients, d_1 is the number of genes,
## Data2 is of size n x d_2,
## where n is the number of patients, d_2 is the number of methylation
data(Data1)
data(Data2)

## Calculate distance matrices(here we calculate Euclidean Distance,
## you can use other distance, e.g. correlation)
Dist1 = dist2(as.matrix(Data1), as.matrix(Data1))
Dist2 = dist2(as.matrix(Data2), as.matrix(Data2))
```

`estimateNumberOfClustersGivenGraph`
Estimate Number Of Clusters Given Graph

Description

This function estimates the number of clusters given the two huristics given in the supplementary materials of our nature method paper W is the similarity graph NUMC is a vector which contains the possible choices of number of clusters.

Usage

```
estimateNumberOfClustersGivenGraph(W, NUMC=2:5)
```

Arguments

`W`	List of matrices. Each element of the list is a square, symmetric matrix that shows affinities of the data points from a certain view.
`NUMC`	A vector which contains the possible choices of number of clusters.

Value

K1 is the estimated best number of clusters according to eigen-gaps K12 is the estimated SECOND best number of clusters according to eigen-gaps K2 is the estimated number of clusters according to rotation cost K22 is the estimated SECOND number of clusters according to rotation cost

Author(s)

Dr. Anna Goldenberg, Bo Wang, Aziz Mezlini, Feyyaz Demir

References

B Wang, A Mezlini, F Demir, M Fiume, T Zu, M Brudno, B Haibe-Kains, A Goldenberg (2014) Similarity Network Fusion: a fast and effective method to aggregate multiple data types on a genome wide scale. Nature Methods. Online. Jan 26, 2014

Concise description can be found here: http://compbio.cs.toronto.edu/SNF/SNF/Software.html

Examples

```
## First, set all the parameters:
K = 20;   # number of neighbors, usually (10~30)
alpha = 0.5;   # hyperparameter, usually (0.3~0.8)
T = 20;  # Number of Iterations, usually (10~20)

## Data1 is of size n x d_1,
## where n is the number of patients, d_1 is the number of genes,
## Data2 is of size n x d_2,
## where n is the number of patients, d_2 is the number of methylation
data(Data1)
data(Data2)

## Here, the simulation data (SNFdata) has two data types. They are complementary to each other.
## And two data types have the same number of points.
## The first half data belongs to the first cluster; the rest belongs to the second cluster.
truelabel = c(matrix(1,100,1),matrix(2,100,1)); ## the ground truth of the simulated data

## Calculate distance matrices
## (here we calculate Euclidean Distance, you can use other distance, e.g,correlation)

## If the data are all continuous values, we recommend the users to perform
## standard normalization before using SNF,
## though it is optional depending on the data the users want to use.
# Data1 = standardNormalization(Data1);
# Data2 = standardNormalization(Data2);
```

```
## Calculate the pair-wise distance;
## If the data is continuous, we recommend to use the function "dist2" as follows
Dist1 = (dist2(as.matrix(Data1),as.matrix(Data1)))^(1/2)
Dist2 = (dist2(as.matrix(Data2),as.matrix(Data2)))^(1/2)

## next, construct similarity graphs
W1 = affinityMatrix(Dist1, K, alpha)
W2 = affinityMatrix(Dist2, K, alpha)

## These similarity graphs have complementary information about clusters.
displayClusters(W1,truelabel);
displayClusters(W2,truelabel);

## next, we fuse all the graphs
## then the overall matrix can be computed by similarity network fusion(SNF):
W = SNF(list(W1,W2), K, T)

## With this unified graph W of size n x n,
## you can do either spectral clustering or Kernel NMF.
## If you need help with further clustering, please let us know.

## You can display clusters in the data by the following function
## where C is the number of clusters.
C = 2  # number of clusters
group = spectralClustering(W,C);  # the final subtypes information
displayClusters(W, group)

## You can get cluster labels for each data point by spectral clustering
labels = spectralClustering(W, C)

plot(Data1, col=labels, main='Data type 1')
plot(Data2, col=labels, main='Data type 2')

## Here we provide two ways to estimate the number of clusters. Note that,
## these two methods cannot guarantee the accuracy of esstimated number of
## clusters, but just to offer two insights about the datasets.

estimationResult = estimateNumberOfClustersGivenGraph(W, 2:5);
```

`getColorsForGroups` *Obtaining a vector of colors from a numeric vector of group*

Description

Convert a numeric vector containing group information to a vector of colors

Usage

```
getColorsForGroups(group, colors)
```

Arguments

`group`	A numeric vector containing the groups information such as the result of the spectralClustering function.
`colors`	a vector of colors to be used for the different groups. If the number of group is > 8, the user will have to use the colors argument and give a vector of colors with length at least equal to the number of groups.

Details

Essentially used to construct a vector or a matrix with colors used as for the ColSideColors argument in the displayClustersWithHeatmap function. See the displayClustersWithHeatmap()'s example.

Value

A character vector of colors, corresponding to the given vector of group, keeping the same order.

Author(s)

Florence Cavalli

Examples

```
## Example 1
gp=c(rep(1,10),rep(2,4),rep(1,3),rep(3,6))
## Using the default colors
gp_colors=getColorsForGroups(gp)
gp_colors
## Specifying the colors
gp_colors=getColorsForGroups(gp,colors=c("cyan","purple","orange"))
gp_colors

## Example 2: Part of SNF
## First, set all the parameters:
K = 20;    # number of neighbors, usually (10~30)
alpha = 0.5;    # hyperparameter, usually (0.3~0.8)
T = 20;   # Number of Iterations, usually (10~20)

## Data1 is of size n x d_1,
## where n is the number of patients, d_1 is the number of genes,
## Data2 is of size n x d_2,
## where n is the number of patients, d_2 is the number of methylation
data(Data1)
data(Data2)

## Here, the simulation data (SNFdata) has two data types. They are complementary to each other.
## And two data types have the same number of points.
## The first half data belongs to the first cluster; the rest belongs to the second cluster.
```

```
truelabel = c(matrix(1,100,1),matrix(2,100,1)); ## the ground truth of the simulated data

## Calculate distance matrices
## (here we calculate Euclidean Distance, you can use other distance, e.g,correlation)

## If the data are all continuous values, we recommend the users to perform
## standard normalization before using SNF,
## though it is optional depending on the data the users want to use.
# Data1 = standardNormalization(Data1);
# Data2 = standardNormalization(Data2);

## Calculate the pair-wise distance;
## If the data is continuous, we recommend to use the function "dist2" as follows
Dist1 = dist2(as.matrix(Data1),as.matrix(Data1));
Dist2 = dist2(as.matrix(Data2),as.matrix(Data2));

## next, construct similarity graphs
W1 = affinityMatrix(Dist1, K, alpha)
W2 = affinityMatrix(Dist2, K, alpha)

## next, we fuse all the graphs
## then the overall matrix can be computed by similarity network fusion(SNF):
W = SNF(list(W1,W2), K, T)

## With this unified graph W of size n x n,
## you can do either spectral clustering or Kernel NMF.
## If you need help with further clustering, please let us know.

## You can display clusters in the data by the following function
## where C is the number of clusters.
C = 2      # number of clusters
group = spectralClustering(W,C);  # the final subtypes information

## Get a matrix containing the group information
## for the samples such as the SpectralClustering result and the True label
M_label=cbind(group,truelabel)
colnames(M_label)=c("spectralClustering","TrueLabel")

## ****
## Comments
## rownames(M_label)=names(spectralClustering) To add if the spectralClustering function
## pass the sample ID as names.
## or rownames(M_label)=rownames(W) Having W with rownames and colmanes
## with smaple ID would help as well.
## ***

## Use the getColorsForGroups function to assign a color to each group
## NB is more than 8 groups, you will have to input a vector
## of colors into the getColorsForGroups function
M_label_colors=t(apply(M_label,1,getColorsForGroups))
## or choose you own colors for each label, for example:
M_label_colors=cbind("spectralClustering"=getColorsForGroups(M_label[,"spectralClustering"],
colors=c("blue","green")),"TrueLabel"=getColorsForGroups(M_label[,"TrueLabel"],
colors=c("orange","cyan")))

## Visualize the clusters present in the given similarity matrix
## as well as some sample information
## In this presentation no clustering method is ran the samples
## are ordered in function of their group label present in the group arguments
displayClustersWithHeatmap(W, group, M_label_colors[,"spectralClustering"])
displayClustersWithHeatmap(W, group, M_label_colors)
```

`groupPredict` *Group Predict*

Description

This function is used to predict the subtype of new patients.

Usage

```
groupPredict(train, test, groups, K=20, alpha=0.5, t=20, method=1)
```

Arguments

`train`	Training data. Has the same number of view and columns as test data.
`test`	Test data. Has the same number of view and columns as training data.
`groups`	The label for the training data.
`K`	Number of neighbors.
`alpha`	Hyperparameter used in constructing similarity network.
`t`	Number of iterations.
`method`	A indicator of which method to use to predict the label. method = 0 means to use local and global consistency; method = 1 means to use label propagation.

Value

Returns the prediction of which group the test data belongs to.

Author(s)

Dr. Anna Goldenberg, Bo Wang, Aziz Mezlini, Feyyaz Demir

Examples

```
# Provide an example of predicting the new labels with label propagation

# Load views into list "dataL" and the cluster assignment into vector "label"
data(dataL)
data(label)

# Create the training and test data
n = floor(0.8*length(label)) # number of training cases
trainSample = sample.int(length(label), n)
train = lapply(dataL, function(x) x[trainSample, ]) # Use the first 150 samples for training
test = lapply(dataL, function(x) x[-trainSample, ]) # Test the rest of the data set
groups = label[trainSample]

# Set the other
K = 20
alpha = 0.5
t = 20
method = TRUE

# Apply the prediction function to the data
newLabel = groupPredict(train,test,groups,K,alpha,t,method)

# Compare the prediction accuracy
accuracy = sum(label[-trainSample] == newLabel[-c(1:n)])/(length(label) - n)
```

`label` *Labels for dataL dataset*

Description

The ground truth for dataL dataset

Usage

```
data(label)
```

Format

The format is: int [1:600] 1 1 1 1 1 1 1 1 1 1 ...

Examples

```
data(label)
```

`plotAlluvial` *Plot Alluvial*

Description

This function plots an alluvial (Parallel coordinate plot) of sample clusterings for a specified number of clusters. Samples can be coloured by providing a vector of colours, allowing for the visualization of sample properties over a range of clustering number choices.

*This is a wrapper function calling the Alluvial Package (Bojanowski M. & Edwards R)

Usage

```
plotAlluvial(W, clust.range, color.vect)
```

Arguments

`W`	Affinity matrix of dimension n.samples by n.samples
`clust.range`	Integer vector specifying the number of clusters for each clustering
`color.vect`	A vector of color's of length n.samples to colour the samples

Value

Plots an alluvial plot for range of clustering choices.

Author(s)

Daniel Cole

See Also

More information on Alluvial Package

Examples

```
K <- 20
alpha <- 0.5
iter <- 20

data(Data1)
data(Data2)

dist1 <- (dist2(as.matrix(Data1), as.matrix(Data1)))^(1/2)
dist2 <- (dist2(as.matrix(Data2), as.matrix(Data2)))^(1/2)

W1 <- affinityMatrix(dist1, K, alpha)
W2 <- affinityMatrix(dist2, K, alpha)

W <- SNF(list(W1, W2), K, iter)
```

```
#Plots the alluvial with no colouring
plotAlluvial(W, 2:5)

#Change the colour of all samples a single colour
plotAlluvial(W, 2:5, col="red")

colour.breaks <- 30
#This will assign each sample to one of colour.breaks colour bins between green and red.
colFunc <- colorRampPalette(c("green", "red"))
colours <- colFunc(colour.breaks)[as.numeric(cut(Data1[,1],breaks=colour.breaks))]
plotAlluvial(W, 2:5, col=colours)
```

`rankFeaturesByNMI` *Rank Features by NMI*

Description

Ranks each features by NMI based on their clustering assingments

Usage

```
rankFeaturesByNMI(data, W)
```

Arguments

`data`	List containing all the data types.
`W`	Target Matrix for which the NMI is calculated against.

Value

List containing the NMI and rank based on NMI for each feature.

Author(s)

Dr. Anna Goldenberg, Bo Wang, Aziz Mezlini, Feyyaz Demir

Examples

```
## First, set all the parameters:
K = 20; # number of neighbors, usually (10~30)
alpha = 0.5;   # hyperparameter, usually (0.3~0.8)
T = 20;  # Number of Iterations, usually (10~20)

## Data1 is of size n x d_1,
## where n is the number of patients, d_1 is the number of genes,
## Data2 is of size n x d_2,
## where n is the number of patients, d_2 is the number of methylation
```

```
data(Data1)
data(Data2)

## Here, the simulation data (SNFdata) has two data types. They are complementary to each other.
## And two data types have the same number of points.
## The first half data belongs to the first cluster; the rest belongs to the second cluster.
truelabel = c(matrix(1,100,1),matrix(2,100,1)); ## the ground truth of the simulated data

## Calculate distance matrices
## (here we calculate Euclidean Distance, you can use other distance, e.g,correlation)

## If the data are all continuous values, we recommend the users to perform
## standard normalization before using SNF,
## though it is optional depending on the data the users want to use.
# Data1 = standardNormalization(Data1);
# Data2 = standardNormalization(Data2);



## Calculate the pair-wise distance;
## If the data is continuous, we recommend to use the function "dist2" as follows
Dist1 = (dist2(as.matrix(Data1),as.matrix(Data1)))^(1/2)
Dist2 = (dist2(as.matrix(Data2),as.matrix(Data2)))^(1/2)

## next, construct similarity graphs
W1 = affinityMatrix(Dist1, K, alpha)
W2 = affinityMatrix(Dist2, K, alpha)

## next, we fuse all the graphs
## then the overall matrix can be computed by similarity network fusion(SNF):
W = SNF(list(W1,W2), K, T)

NMI_scores <- rankFeaturesByNMI(list(Data1, Data2), W)
```

SNF *Similarity Network Fusion*

Description

Similarity Network Fusion takes multiple views of a network and fuses them together to construct an overall status matrix. The input to our algorithm can be feature vectors, pairwise distances, or pairwise similarities. The learned status matrix can then be used for retrieval, clustering, and classification.

Usage

```
SNF(Wall, K, t)
```

Arguments

`Wall`	List of matrices. Each element of the list is a square, symmetric matrix that shows affinities of the data points from a certain view.
`K`	Number of neighbors in K-nearest neighbors part of the algorithm.
`t`	Number of iterations for the diffusion process.

Value

W is the overall status matrix derived

Author(s)

Dr. Anna Goldenberg, Bo Wang, Aziz Mezlini, Feyyaz Demir

References

B Wang, A Mezlini, F Demir, M Fiume, T Zu, M Brudno, B Haibe-Kains, A Goldenberg (2014) Similarity Network Fusion: a fast and effective method to aggregate multiple data types on a genome wide scale. Nature Methods. Online. Jan 26, 2014

Concise description can be found here: http://compbio.cs.toronto.edu/SNF/SNF/Software.html

Examples

```
## First, set all the parameters:
K = 20; # number of neighbors, usually (10~30)
alpha = 0.5;   # hyperparameter, usually (0.3~0.8)
T = 20;  # Number of Iterations, usually (10~20)

## Data1 is of size n x d_1,
## where n is the number of patients, d_1 is the number of genes,
## Data2 is of size n x d_2,
## where n is the number of patients, d_2 is the number of methylation
data(Data1)
data(Data2)

## Here, the simulation data (SNFdata) has two data types. They are complementary to each other.
## And two data types have the same number of points.
## The first half data belongs to the first cluster; the rest belongs to the second cluster.
truelabel = c(matrix(1,100,1),matrix(2,100,1)); ## the ground truth of the simulated data

## Calculate distance matrices
## (here we calculate Euclidean Distance, you can use other distance, e.g,correlation)

## If the data are all continuous values, we recommend the users to perform
## standard normalization before using SNF,
## though it is optional depending on the data the users want to use.
# Data1 = standardNormalization(Data1);
# Data2 = standardNormalization(Data2);
```

```
## Calculate the pair-wise distance;
## If the data is continuous, we recommend to use the function "dist2" as follows
Dist1 = (dist2(as.matrix(Data1),as.matrix(Data1)))^(1/2)
Dist2 = (dist2(as.matrix(Data2),as.matrix(Data2)))^(1/2)

## next, construct similarity graphs
W1 = affinityMatrix(Dist1, K, alpha)
W2 = affinityMatrix(Dist2, K, alpha)

## These similarity graphs have complementary information about clusters.
displayClusters(W1,truelabel);
displayClusters(W2,truelabel);

## next, we fuse all the graphs
## then the overall matrix can be computed by similarity network fusion(SNF):
W = SNF(list(W1,W2), K, T)

## With this unified graph W of size n x n,
## you can do either spectral clustering or Kernel NMF.
## If you need help with further clustering, please let us know.

## You can display clusters in the data by the following function
## where C is the number of clusters.
C = 2  # number of clusters
group = spectralClustering(W,C);  # the final subtypes information
displayClusters(W, group)

## You can get cluster labels for each data point by spectral clustering
labels = spectralClustering(W, C)

plot(Data1, col=labels, main='Data type 1')
plot(Data2, col=labels, main='Data type 2')
```

spectralClustering *Spectral Clustering*

Description

Perform the famous spectral clustering algorithms. There are three variants. The default one is the third type.

Usage

```
spectralClustering(affinity, K, type = 3)
```

Arguments

`affinity`	Similarity matrix
`K`	Number of clusters
`type`	The variants of spectral clustering to use.

Value

A vector consisting of cluster labels of each sample.

Author(s)

Dr. Anna Goldenberg, Bo Wang, Aziz Mezlini, Feyyaz Demir

Examples

```
## First, set all the parameters:
K = 20;##number of neighbors, usually (10~30)
alpha = 0.5; ##hyperparameter, usually (0.3~0.8)
T = 20; ###Number of Iterations, usually (10~50)

## Data1 is of size n x d_1,
## where n is the number of patients, d_1 is the number of genes,
## Data2 is of size n x d_2,
## where n is the number of patients, d_2 is the number of methylation
data(Data1)
data(Data2)

## Calculate distance matrices (here we calculate Euclidean Distance,
## you can use other distance, e.g. correlation)
Dist1 = (dist2(as.matrix(Data1),as.matrix(Data1)))^(1/2)
Dist2 = (dist2(as.matrix(Data2),as.matrix(Data2)))^(1/2)

## Next, construct similarity graphs
W1 = affinityMatrix(Dist1, K, alpha)
W2 = affinityMatrix(Dist2, K, alpha)

# Next, we fuse all the graphs
# then the overall matrix can be computed by
W = SNF(list(W1,W2), K, T)

## With this unified graph W of size n x n,
## you can do either spectral clustering or Kernel NMF.
## If you need help with further clustering, please let us know.

## You can display clusters in the data by the following function
## where C is the number of clusters.
C = 2

## You can get cluster labels for each data point by spectral clustering
labels = spectralClustering(W, C)
```

`standardNormalization` *Standard Normalization*

Description

Normalize each column of the input data to have mean 0 and standard deviation 1.

Usage

```
standardNormalization(x)
```

Arguments

`x`	The unnormalized data.

Value

The data normalized.

Author(s)

Dr. Anna Goldenberg, Bo Wang, Aziz Mezlini, Feyyaz Demir

Examples

```
## Data1 is of size n x d_1,
## where n is the number of patients, d_1 is the number of genes,
## Data2 is of size n x d_2,
## where n is the number of patients, d_2 is the number of methylation
data(Data1)
data(Data2)

Data1 = standardNormalization(Data1);
Data2 = standardNormalization(Data2);
```

Index