Package 'SGCP'

September 16, 2024

Type Package

Title SGCP: a spectral self-learning method for clustering genes in co-expression networks

Version 1.4.4

Description SGC is a semi-supervised pipeline for gene clustering in gene co-expression networks. SGC consists of multiple novel steps that enable the computation of highly enriched modules in an unsupervised manner. But unlike all existing frameworks, it further incorporates a novel step that leverages Gene Ontology information in a semi-supervised clustering method that further improves the quality of the computed modules.

License GPL-3

Encoding UTF-8

Imports ggplot2, expm, caret, plyr, dplyr, GO.db, annotate, SummarizedExperiment, genefilter, GOstats, RColorBrewer, xtable, Rgraphviz, reshape2, openxlsx, ggridges, DescTools, org.Hs.eg.db, methods, grDevices, stats, RSpectra, graph

Suggests knitr, rmarkdown, BiocManager, devtools, BiocStyle

Depends R (>= 4.2.0)

biocViews GeneExpression, GeneSetEnrichment, NetworkEnrichment, SystemsBiology, Classification, Clustering, DimensionReduction, GraphAndNetwork, NeuralNetwork, Network, mRNAMicroarray, RNASeq, Visualization

VignetteBuilder knitr

NeedsCompilation no

URL https://github.com/na396/SGCP

RoxygenNote 7.2.1

LazyData true

git_url https://git.bioconductor.org/packages/SGCP

git_branch RELEASE_3_19

git_last_commit 01d2332

git_last_commit_date 2024-08-19

Repository Bioconductor 3.19

Date/Publication 2024-09-15

Author Niloofar AghaieAbiane [aut, cre]
(<https://orcid.org/0000-0003-1096-7592>),
Ioannis Koutis [aut]

Maintainer Niloofar AghaieAbiane <niloofar.abiane@gmail.com>

Contents

`adjacencyMatrix` *Performs netwrok construction step in the SGCP pipeline*

Description

It creates the adjacency matrix of the gene co-expression network in the SGCP pipeline. Users can specify steps in the following order: calibration, norm, Gaussian kernel, and tom. If `calibration` is set to `TRUE`, SGCP performs calibration as the first step (refer to the manuscript for details). If `norm` is `TRUE`, each gene is normalized by its L2 norm. The Gaussian kernel metric is then calculated as a mandatory step to determine pairwise gene similarity values. If `tom` is `TRUE`, SGCP incorporates second-order node neighborhood information into the network. The pipeline concludes by returning a symmetric adjacency matrix `adja` of size m**n, where n is the number of genes. All values in the

adjacency matrix range from 0 to 1, with 1 indicating maximum similarity. The diagonal elements of the matrix are set to zero.

Usage

```
adjacencyMatrix(expData, calibration = FALSE, norm = TRUE,
                tom = TRUE, saveAdja = FALSE,
                adjaNameFile = "adjacency.RData",
                hm = "adjaHeatMap.png")
```

Arguments

expData	A dataframe or matrix containing the expression data, where rows correspond to genes and columns to samples.
calibration	Logical, default FALSE. If TRUE, performs calibration step.
norm	Logical, default TRUE. If TRUE, divides each gene (row) by its norm2.
tom	Logical, default TRUE. If TRUE, adds TOM to the network.
saveAdja	Logical, default FALSE. If TRUE, saves the adjacency matrix.
adjaNameFile	String indicating the name of the file for saving the adjacency matrix.
hm	String indicating the name of the file for saving the adjacency matrix heatmap.

Value

adja	A symmetric matrix of dimension n * n representing the adjacency matrix, where n is the number of genes. Values range in (0, 1) with a zero diagonal.

References

Aghaieabiane, N and Koutis, I (2024) SGCP: a spectral self-learning method for clustering genes in co-expression networks

See Also

SGCP Toturial calibration step information

Examples

```
## create an adjcency matrix
GeneExpression <- matrix(runif(1000, 0,1), nrow = 200, ncol = 5)
diag(GeneExpression) <- 0

## call the function
adja <- adjacencyMatrix(GeneExpression, hm= NULL)
head(adja)
```

cheng *Normalized gene expression data from Cheng et al.'s publication on ischemic cardiomyopathy (ICM).*

Description

This dataset contains normalized gene expression data for 1500 genes across 5 samples. It is a subset of a larger dataset related to ischemic cardiomyopathy (ICM), which includes 5000 genes and 57 samples. The normalization was performed using the DESeq method, which utilizes the median ratio of gene counts to achieve normalization.

Usage

```
data(cheng)
```

Format

An object of class `SummarizedExperiment`.

Details

assays contains the gene expression data and rowData field includes the corresponding gene Entrez IDs. Sample names also are avialable in colData.

Source

https://www.sciencedirect.com/science/article/pii/S0010482520303061?via%3Dihub

Examples

```
## load cheng dataset
library(SGCP)
library(SummarizedExperiment)

data(cheng)
expData <- assay(cheng)
geneID <- rowData(cheng)
geneID <- geneID$ENTREZID
```

clustering *Perform network clustering step in the SGCP pipeline*

Description

It performs clustering on the adjacency network of gene co-expression network in SGCP pipeline. Initially, it transforms the n*n adjacency matrix into a new dimension Y. Subsequently, it determines the number of clusters k using three methods: "relativeGap", "secondOrderGap", and "additiveGap". For each method, k-means clustering is applied to Y with the determined k as input. Conductance indices are computed for the clusters within each method, and the cluster with the smallest conductance index is selected for further analysis. Following this, gene ontology enrichment analysis is performed on the selected clusters to finalize the optimal k. The pipeline concludes by returning the result of k-means clustering based on the selected method, along with the transformed matrix Y and additional information. This step produces the initial clusters.

Usage

```
clustering(adjaMat, geneID , annotation_db ,
            kopt = NULL, method = NULL,
            func.GO = sum, func.conduct = min,
            maxIter = 1e8, numStart = 1000, eff.egs = TRUE,
            saveOrig = TRUE, n_egvec = 200, sil = FALSE)
```

Arguments

adjaMat	A squared symmetric matrix of size n*n with values in (0, 1) and 0 diagonal. This is the output of the adjacencyMatrix function in SGCP.
geneID	A vector containing gene IDs of size n, where n is the number of genes.
annotation_db	A string indicating the genomic-wide annotation database.
kopt	An integer denoting the optimal number of clusters \(k \) chosen by the user (default: NULL).
method	Method for identifying the number of clusters \(k \) (default: NULL). Options include "relativeGap", "secondOrderGap", "additiveGap", or NULL.
func.GO	A function for gene ontology validation (default: sum).
func.conduct	A function for conductance validation (default: min).
maxIter	An integer specifying the maximum number of iterations for k-means clustering.
numStart	An integer indicating the number of starts for k-means clustering.
eff.egs	Boolean (default: TRUE). If TRUE, uses eigs_sym to calculate eigenvalues and eigenvectors, which is more efficient than R's default function.
saveOrig	Boolean (default: TRUE). If TRUE, keeps the transformation matrix.
n_egvec	An integer (default: 200) specifying the number of columns of the transformation matrix to retain. Should be less than 200.
sil	Boolean (default: FALSE). If TRUE, calculates silhouette index for each cluster.

Details

If kopt is not null, SGCP will determine clusters based on the specified kopt. Otherwise, if method is not NULL, SGCP will select k using the specified method. If both geneID and annotation_db are NULL, SGCP will determine the optimal method and its corresponding number of clusters based on conductance validation. It selects a method where the conductance, evaluated by func.conduct, is minimized. Alternatively, SGCP defaults to gene ontology validation to find the optimal method and its corresponding clusters. It performs gene ontology enrichment on clusters, selecting the method where the cluster with the minimum conductance index yeilds the highest func.GO over log10 of the p-values.

Value

- dropped.indices A vector of dropped gene indices.
- geneID A vector of gene IDs.
- method Indicates the selected method for number of clusters.
- k Selected number of clusters.
- Y Transformed matrix with 2*k columns.
- X Eigenvalues corresponding to 2*k columns in Y.
- cluster An object of class kmeans.
- clusterLabels A vector containing the cluster label per gene, with a 1-to-1 correspondence to geneID.
- conductance A list containing mean and median conductance indices for clusters per method. The index in clusterConductance field denotes the cluster label and the value shows the conductance index.
- cvGOdf A dataframe used for gene ontology validation. For each method, it returns the gene ontology enrichment result on the cluster with the minimum conductance index.
- cv A string indicating the validation method for number of clusters:
  - "cvGO": Gene ontology validation used.
  - "cvConductance": Conductance validation used.
  - "userMethod": User-defined method.
  - "userkopt": User-defined kopt.
- clusterNumberPlot An object of class ggplot2 for relativeGap, secondOrderGap, and additiveGap.
- silhouette A dataframe indicating the silhouette index for genes.
- original A list with matrix transformation, corresponding eigenvalues, and n_egvec, where the top n_egvec columns of the transformation are retained.

References

Aghaieabiane, N and Koutis, I (2024) SGCP: a spectral self-learning method for clustering genes in co-expression networks

See Also

adjacencyMatrix SGCP Toturial

Examples

```
## load cheng dataset
library(SGCP)
library(SummarizedExperiment)

data(cheng)
expData <- assay(cheng)
geneID <- rowData(cheng)
geneID <- geneID$ENTREZID


# to create the adjacency matrix un comment the following
## resAdja <- adjacencyMatrix(expData = expData, hm = NULL)
## resAdja[0:10, 0:5]

# to perform clustering
## library(org.Hs.eg.db)
annotation_db = "org.Hs.eg.db"
## resClus = clustering(adjaMat = resAdja, geneID = geneID,
##              annotation_db = annotation_db)
```

---

ezSGCP *Integrated execution of the SGCP pipeline*

Description

The SGCP pipeline for gene co-expression network construction and analysis integrates multiple steps into a single function. It begins with network construction, where gene expression data and gene IDs are utilized alongside an annotation database to build an adjacency matrix. Next, network clustering identifies initial clusters. Gene ontology enrichment distinguishes genes into remarkable and unremarkable sets, enabling semi-labeling to convert the problem into semi-supervised learning. Remarkable genes serve as the training set for a supervised model, predicting labels for unremarkable genes and producing final modules. Finally, another gene ontology step evaluates module enrichment.

Usage

```
ezSGCP(expData, geneID, annotation_db, semilabel = TRUE,
    calib = FALSE, norm = TRUE, tom = TRUE,
    saveAdja = FALSE, adjaNameFile = "adjacency.Rdata",
    hm = "adjaHeatMap.png",
    kopt = NULL, method_k = NULL, f.GO = sum, f.conduct = min,
    maxIteration = 1e8, numberStart = 1000, eff.egs = TRUE,
    saveOrig = TRUE, n_egvec = 100, sil = FALSE,
    dir = c("over", "under"), onto = c("BP", "CC", "MF"),
    hgCut = NULL, condTest = TRUE,
```

```
          cutoff = NULL, percent = 0.10, stp = 0.01,
          model = "knn", kn = NULL)
```

Arguments

expData	A dataframe or matrix containing the expression data, where rows correspond to genes and columns to samples.
geneID	A vector containing the gene IDs of size n, where n is the number of genes.
annotation_db	A string indicating the genomic-wide annotation database.
semilabel	Logical, default TRUE. If TRUE, performs semilabeling step.
calib	Logical, default FALSE. If TRUE, performs calibration step.
norm	Logical, default TRUE. If TRUE, divides each gene (row) by its norm2.
tom	Logical, default TRUE. If TRUE, adds TOM to the network.
saveAdja	Logical, default FALSE. If TRUE, saves the adjacency matrix.
adjaNameFile	String indicating the name of the file for saving the adjacency matrix.
hm	String indicating the name of the file for saving the adjacency matrix heatmap.
kopt	An integer indicating the optimal number of clusters k chosen by the user, default is NULL.
method_k	Method for identifying the number of clusters k, default NULL. Options are "relativeGap", "secondOrderGap", "additiveGap", or NULL.
f.GO	A function for gene ontology validation, default is sum.
f.conduct	A function for conductance validation, default is min.
maxIteration	An integer indicating the maximum number of iterations for kmeans.
numberStart	An integer indicating the number of starts for kmeans.
eff.egs	Boolean, default TRUE. If TRUE, uses eigs_sym to calculate eigenvalues and eigenvectors, which is more efficient than R's default function.
saveOrig	Boolean, default TRUE. If TRUE, keeps the transformation matrix.
n_egvec	Either "all" or an integer indicating the number of columns of the transformation matrix to keep, default is 100.
sil	Logical, default FALSE. If TRUE, calculates silhouette index for each cluster.
dir	Test direction for GO terms, default c("over", "under").
onto	GO ontologies to consider, default c("BP", "CC", "MF").
hgCut	Numeric value in (0,1) as the p-value cutoff for GO terms, default 0.05.
condTest	Logical, default TRUE. If TRUE, performs conditional hypergeometric test.
cutoff	Numeric in (0, 1) default NULL, baseline for GO term significance.
percent	Numeric in (0,1) default 0.1, percentile for finding top GO terms.
stp	Numeric in (0,1) default 0.01, increasing value for percent parameter.
model	Type of classification model, either "knn" (k nearest neighbors) or "lr" (logistic regression).
kn	Integer indicating the number of neighbors in knn, default NULL.

Details

For clustering step; If kopt is not NULL, SGCP finds clusters based on kopt. If method_k is not NULL, SGCP picks k based on the selected method ("relativeGap", "secondOrderGap", "additiveGap"). If geneID or annotation_db is NULL, SGCP determines the optimal method and corresponding number of clusters based on conductance validation. It selects the method where func.conduct on its clusters is minimized. Otherwise, SGCP uses gene ontology validation (by default) to find the optimal method and its corresponding number of clusters. It performs gene ontology enrichment on the cluster with the minimum conductance index per method and selects the method that maximizes func.GO over -log10 of p-values.

For semilabeling step; Genes associated with GO terms more significant than cutoff value are considered remarkable. If cutoff value is NULL, SGCP determines the cutoff based on the significance level of GO terms. SGCP selects the top percent (default 0.1) GO terms from all clusters collectively and considers genes associated with those as remarkable. If all remarkable genes come from a single cluster, SGCP increases the percent by 0.01 until remarkable genes come from at least two clusters.

For semi-supervise step; Remarkable clusters are those that have at least one remarkable gene. SGCP performs semi-supervised classification using the transformed matrix from clustering and gene semilabels from semilabeling function. It uses remarkable genes as the training set to train either a "k nearest neighbor" (knn) or "logistic regression" (lr) model and makes predictions for unremarkable genes to produce the final modules.

Value

Returns a list with the following fields, depending on the initial call:

- semilabel Boolean indicating if semilabeling step was performed.
- clusterLabels DataFrame with geneID and its corresponding initial and final labels.
- clustering List containing clustering information:
  - dropped.indices Vector of dropped gene indices.
  - geneID Vector of geneIDs.
  - method Method selected for number of clusters.
  - k Selected number of clusters.
  - Y Transformed matrix with 2*k columns.
  - X Eigenvalues corresponding to 2*k columns in Y.
  - cluster Object of class kmeans.
  - clusterLabels Vector containing cluster labels for each gene.
  - conductance List with mean, median, and individual cluster conductance indices. clusterConductance field denotes the cluster label and its conductance index.
  - cvGOdf DataFrame used for gene ontology validation. For each method, shows GO enrichment on the cluster with smallest conductance index.
  - cv String indicating validation method for number of clusters: "cvGO", "cvConductance", "userMethod", or "userkopt".
  - clusterNumberPlot Object of class ggplot2 for visualizing cluster number selection methods.
  - silhouette DataFrame indicating silhouette indices for genes.

  - original List with matrix transformation, corresponding eigenvalues, and n_egvec top columns of transformation matrix kept.
- initial.GO List containing initial gene ontology (GO) information:
  - GOresults DataFrame summarizing GO term information. Includes clusterNum, GOtype, GOID, Pvalue, OddsRatio, ExpCount, Count, Size, and Term.
  - FinalGOTermGenes List of geneIDs associated with each GO term per cluster.
- semiLabeling List containing semilabeling information:
  - cutoff Numeric (0,1) indicating selected cutoff for significant GO terms.
  - geneLabel DataFrame with geneID and its corresponding cluster label if remarkable, otherwise NA.
- semiSupervised List containing semi-supervised classification information:
  - semiSupervised Object of classification result.
  - prediction Vector of predicted labels for unremarkable genes.
  - FinalLabeling DataFrame of geneID with its corresponding semilabel and final label.
- final.GO List containing final gene ontology (GO) information:
  - GOresults DataFrame summarizing GO term information. Includes clusterNum, GOtype, GOID, Pvalue, OddsRatio, ExpCount, Count, Size, and Term.
  - FinalGOTermGenesList of geneIDs associated with each GO term per cluster.

References

Aghaieabiane, N and Koutis, I (2024) SGCP: a spectral self-learning method for clustering genes in co-expression networks

See Also

SGCP Toturial

Examples

```
## load cheng dataset
library(SGCP)
library(SummarizedExperiment)
data(cheng)
expData <- assay(cheng)
geneID <- rowData(cheng)
geneID <- geneID$ENTREZID


library(org.Hs.eg.db)

# to call the function uncomment the following
## res <- ezSGCP(expData = expData, geneID = geneID, annotation_db = "org.Hs.eg.db")
## summary(res)
## summary(res$clustering)
## summary(res$initial.GO)
## summary(res$semiLabeling)
```

```
## summary(res$semiSupervised)
## summary(res$final.GO)
```

---

geneOntology *Performs gene ontology enrichment step in the SGCP pipeline.*

Description

It performs gene ontology enrichment step GOstat package in SGCP pipeline. It takes the entire genes in the input with their labels, along with annotation_db to perform gene ontology enrichment for each set of genes that have similar label.

Usage

```
geneOntology(geneUniv, clusLab, annotation_db,
             direction = c("over", "under"),
             ontology = c("BP", "CC", "MF"), hgCutoff = NULL,
             cond = TRUE)
```

Arguments

geneUniv	a vector of all the geneIDs in the expression dataset.
clusLab	a vector of cluster label for each geneID.
annotation_db	a string indicating the genomic wide annotation database.
direction	test direction, default c("over", "under"), for over-represented, or under-represented GO terms.
ontology	GO ontologies, default c("BP", "CC", "MF"), BP: Biological Process, CC: Cellular Component, MF: Molecular Function.
hgCutoff	a numeric value in (0,1) as the p-value cutoff, default 0.05, GO terms smaller than hgCutoff value are kept.
cond	Boolean, default TRUE, if TRUE conditional hypergeometric test is performed.

Value

GOresults	a dataframe containing the summary of the information of GOTerms, clusterNum: indicates the cluster label, GOtype: indicates the test directions plut ontology, GOID: unique GO term id, Pvalue: the p-value of hypergeometric test for the GO term, OddsRatio: the odds ratio of the GO term, ExpCount: expected count value for genes associated the GO term, Count: actual count of the genes associated to the GO term in the cluster, Size: actual size of the genes associated to the GO term in the entire geneIDs, Term: description of the GO term.
FinalGOTermGenes	a list containing the geneIDs of each GOTerms per cluster.

References

Aghaieabiane, N and Koutis, I (2024) SGCP: a spectral self-learning method for clustering genes in co-expression networks

See Also

SGCP Toturial GOstat Toturial

Examples

```
library(SGCP)
# load the output of clustering function
data(resClus)

# call the function
library(org.Hs.eg.db)

# to call the geneOntology uncomment the following
## res <- geneOntology(geneUniv = resClus$geneID, clusLab = resClus$clusterLabels,
##                     annotation_db = "org.Hs.eg.db")
## summary(res$GOresults)
## summary(res$FinalGOTermGenes)
```

`resClus` *An example of the output from* `clustering` *function in the SGCP pipeline*

Description

This is an example of the output from the `clustering` function, representing the network clustering step in the SGCP pipeline. Initially, the adjacency matrix is generated using the `adjacencyMatrix` function within the SGCP framework applied to the `cheng` dataset. This adjacency matrix serves as input to the `clustering` function, resulting in the clustering outcome stored in `resClus`

Usage

```
data(resClus)
```

Format

An object of clas `list` containing the clustering information.

Details

resClus is a list containing the following clustering information:

- dropped.indices: A vector of dropped gene indices.
- geneID: A vector of gene IDs.
- method: Indicates the selected method for determining the number of clusters.
- k: The selected number of clusters.
- Y: Transformed matrix with 2*k columns.
- X: Eigenvalues corresponding to the 2*k columns in Y.
- cluster: An object of class kmeans.
- clusterLabels: A vector containing the cluster label for each gene. There is a 1-to-1 correspondence between geneID and clusterLabels.
- conductance: A list containing the mean, median, and individual cluster conductance index for clusters per method. The index in the clusterConductance field denotes the method.
- cvGOdf: A dataframe used for gene ontology validation. For each method, it returns the gene ontology enrichment result on the cluster with the minimum conductance index.
- cv: A string indicating the validation method for the number of clusters; "cvGO" means gene ontology validation was used.
- clusterNumberPlot: An object of class ggplot2 for relativeGap, secondOrderGap, and additiveGap.
- silhouette: A dataframe indicating the silhouette values for genes.
- original: A list with matrix transformation, corresponding eigenvalues, and n_egvec, where the top
- n_egvec columns of the transformation are kept.

See Also

SGCP Toturial adjacencyMatrix clustering

Examples

```
library(SGCP)
data(resClus)
summary(resClus)
resClus
```

resFinalGO *An example of the output from* geneOntololgy *function in the SGCP pipeline*

Description

This is an example of the output from the geneOntology function, representing the final step in the SGCP pipeline. Initially, the adjacency matrix is generated using the adjacencyMatrix function within the SGCP framework applied to the cheng dataset. This adjacency matrix serves as input to the clustering function, resulting in the clustering outcome stored in resClus. The clustering result, resClus, is subsequently utilized in the geneOntology function to derive resInitialGO, which captures the initial gene ontology (GO) enrichment results. The resInitialGO output is then processed through the semiLabeling function to produce resSemiLabel, indicating the semi-labeled genes based on their clustering characteristics. This semi-labeled information is further employed in the semiSupervised function, yielding resSemiSupervised, which includes the final supervised classification outcomes for the unremarkable genes. Finally, the results from resSemiSupervised are fed into the geneOntology function once more to generate resFinalGO, which represents the final GO enrichment analysis

Usage

```
data(resFinalGO)
```

Format

An object of class list containing the gene ontology information for final gene ontology.

Details

coderesFinalGO is a list containing the following information:

- GOresults: A dataframe of significant gene ontology terms and their corresponding test statistics.
- FinalGOTermGenes: A list of genes belonging to significant gene ontology terms per cluster..

See Also

SGCP Toturial geneOntology

Examples

```
library(SGCP)
data(resFinalGO)
summary(resFinalGO)

# dataframe of significant gene ontology terms
head(resFinalGO$GOresults)
```

```
# a list of genes belong to significant gene ontology term for cluster 1
head(resFinalGO$FinalGOTermGenes$Cluster1_GOTermGenes)

# a list of genes belong to significant gene ontology term for cluster 2
head(resFinalGO$FinalGOTermGenes$Cluster2_GOTermGenes)
```

`resInitialGO` *An example of the output from the* `geneOntololgy` *function in the SGCP pipeline*

Description

This is an example of the output from the `geneOntology` function, representing the third step in the SGCP pipeline. Initially, the adjacency matrix is generated using the `adjacencyMatrix` function within the SGCP framework applied to the `cheng` dataset. This adjacency matrix serves as input to the `clustering` function, resulting in the clustering outcome stored in `resClus`. The clustering result, `resClus`, is subsequently utilized in the `geneOntology` function to derive `resInitialGO`, which captures the initial gene ontology (GO) enrichment results.

Usage

```
data(resInitialGO)
```

Format

An object of class `list` containing the gene ontology information for initial gene ontology.

Details

`resInitialGO` is a list containing the following information.

- `GOresults`: a dataframe of significant gene ontology terms and their corresponding test statistics information.
- `FinalGOTermGenes`: a list of the genes belong to significant gene ontology terms per cluster.

See Also

SGCP Toturial `geneOntology`

Examples

```
library(SGCP)
data(resInitialGO)
summary(resInitialGO)

# dataframe of significant gene ontology terms
head(resInitialGO$GOresults)

# a list of genes belong to significant gene ontology term for cluster 1
```

```
head(resInitialGO$FinalGOTermGenes$Cluster1_GOTermGenes)

# a list of genes belong to significant gene ontology term for cluster 2
head(resInitialGO$FinalGOTermGenes$Cluster2_GOTermGenes)
```

`resSemiLabel` *An example of the output from* `semiLabeling` *function in the SGCP pipeline*

Description

This is an example of the output from the `semiLabeling` function, representing the semi-label step in the SGCP pipeline. Initially, the adjacency matrix is generated using the `adjacencyMatrix` function within the SGCP framework applied to the `cheng` dataset. This adjacency matrix serves as input to the `clustering` function, resulting in the clustering outcome stored in `resClus`. The clustering result, `resClus`, is subsequently utilized in the `geneOntology` function to derive `resInitialGO`, which captures the initial gene ontology (GO) enrichment results. The `resInitialGO` output is then processed through the `semiLabeling` function to produce `resSemiLabel`, indicating the semi-labeled genes based on their clustering characteristics.

Usage

```
data(resSemiLabel)
```

Format

An object of class `list` containing the semi-labeling information.

Details

`resSemiLabel` is a list containing the following information.

- `cutoff`: a numeric in (0,1) that shows the base line for identifying remarkable genes.
- `geneLabel`: a dataframe of geneIDs and its corresponding label, NA labels means that correpsonding genes are unremarkable.

See Also

SGCP Toturial `semiLabeling`

Examples

```
library(SGCP)
data(resSemiLabel)
summary(resSemiLabel)

# cutoff value
head(resSemiLabel$cutoff)
```

```
# gene semi-label
head(resSemiLabel$geneLabel)
```

`resSemiSupervised` *An example of the output from* `semiSupervised` *function in the SGCP pipeline*

Description

This is an example of the output from the `semiSupervised` function, representing the semi-supervised step in the SGCP pipeline. Initially, the adjacency matrix is generated using the `adjacencyMatrix` function within the SGCP framework applied to the `cheng` dataset. This adjacency matrix serves as input to the `clustering` function, resulting in the clustering outcome stored in `resClus`. The clustering result, `resClus`, is subsequently utilized in the `geneOntology` function to derive `resInitialGO`, which captures the initial gene ontology (GO) enrichment results. The `resInitialGO` output is then processed through the `semiLabeling` function to produce `resSemiLabel`, indicating the semi-labeled genes based on their clustering characteristics. This semi-labeled information is further employed in the `semiSupervised` function, yielding `resSemiSupervised`, which includes the final supervised classification outcomes for the unremarkable genes.

Usage

```
data(resSemiSupervised)
```

Format

An object of class `list` containing the semi-supervised information.

Details

`resSemiSupervised` is a list containin the following information.

- `semiSupervised`: an object of `caret` for the training model.
- `prediction`: A vector of predicted labels for unremakable genes.
- `FinalLabeling`: a dataframe gene semil-label and final predicted labels.

See Also

SGCP Toturial `semiLabeling`

Examples

```
library(SGCP)
data(resSemiSupervised)

# supervised model information
summary(resSemiSupervised$semiSupervised)

# predicted label for unremarkable genes
head(resSemiSupervised$prediction)

# gene semi and final labeling
head(resSemiSupervised$FinalLabeling)
```

`semiLabeling` *Performs gene semi-labeling step in the SGCP pipeline*

Description

Performs the Semi-labeling step in the SGCP pipeline to identify remarkable and unremarkable genes. This step involves collecting all gene ontology (GO) terms from all clusters and selecting terms in the top 0.1 percent. Genes associated with these terms are considered remarkable, while the remaining genes are categorized as unremarkable.

Usage

```
semiLabeling(geneID, df_GO, GOgenes, cutoff = NULL,
             percent = 0.10, stp = 0.01)
```

Arguments

`geneID`	A vector containing gene IDs, where n is the number of genes.
`df_GO`	The `GOresults` dataframe returned by the `geneOntology` function, containing information on GO terms in the clusters.
`GOgenes`	The `FinalGOTermGenes` list returned by the `geneOntology` function, listing genes associated with GO terms for each cluster.
`cutoff`	A numeric value in (0, 1) (default: NULL), serving as a baseline for GO term significance.
`percent`	A numeric value in (0, 1) (default: 0.1), indicating the percentile for selecting top GO terms.
`stp`	A numeric value in (0, 1) (default: 0.01), increment added to the `percent` parameter for stepwise selection of top GO terms.

Details

Genes associated with GO terms more significant than the cutoff value are considered remarkable. If the cutoff value is NULL, SGCP determines the cutoff based on the significance level of the GO terms. Otherwise, SGCP selects the top percent (default: 0.1) of GO terms from all clusters combined, considering genes associated with these terms as remarkable. If all remarkable genes originate from a single cluster, SGCP incrementally increases the percent parameter by 0.01 to identify both remarkable and unremarkable genes. This process continues until remarkable genes originate from at least two clusters.

Value

cutoff	a numeric in (0,1) which indicates the selected cutoff.
geneLabel	a dataframe containing the information of geneID and its corresponding cluster label if is remarkable otherwise NA.

References

Aghaieabiane, N and Koutis, I (2024) SGCP: a spectral self-learning method for clustering genes in co-expression networks

See Also

geneOntology SGCP Toturial

Examples

```
library(SGCP)
# load the output of clustering, gene ontology function

data(resClus)
data(resInitialGO)

# call the function

res <- semiLabeling(geneID = resClus$geneID, df_GO = resInitialGO$GOresults,
                GOgenes = resInitialGO$FinalGOTermGenes)
# cutoff value
res$cutoff

# gene semi-labeling information
head(res$geneLabel)
```

`semiSupervised` *Performs the semi-supervised step in the SGCP pipeline*

Description

Performs the semi-supervised classification step in the SGCP pipeline. It utilizes the transformed matrix from the `clustering` function along with gene semi-labels from the `semiLabeling` function. The labeled (remarkable) genes serve as the training set to train either a "k-nearest neighbor" or "logistic regression" model. The trained model then predicts labels for unlabeled (unremarkable) genes, resulting in the final modules.

Usage

```
semiSupervised(specExp, geneLab, model = "knn", kn = NULL)
```

Arguments

`specExp`	Matrix or dataframe where genes are in rows and features are in columns, representing the Y matrix from `clustering` function output.
`geneLab`	A dataframe returned by the `semiLabeling` function, containing geneIDs and their corresponding labels (remarkable or NA).
`model`	Classification model type: "knn" for k-nearest neighbors or "lr" for logistic regression.
`kn`	An integer (default: NULL) indicating the number of neighbors in k-nearest neighbors (knn) model. If kn is `NULL`, the default value is determined by: kn = 20 if 2 * k < 30, otherwise kn = 20 : 30, where k is the number of remarkable clusters.

Details

Remarkable clusters are defined as clusters that contain at least one remarkable gene.

Value

\itemsemiSupervisedAn object of the caret train class representing the semi-supervised classification model.

`prediction`	A vector containing predicted labels for unremarkable genes.
`FinalLabeling`	A dataframe containing geneIDs along with their corresponding semi-labels and final labels.

References

Aghaieabiane, N and Koutis, I (2024) SGCP: a spectral self-learning method for clustering genes in co-expression networks

See Also

clustering semiLabeling SGCP Toturial

Examples

```
library(SGCP)
# load the output of clustering, gene ontology function

data(resClus)
data(resSemiLabel)

# call the function

res <- semiSupervised(specExp = resClus$Y, geneLab = resSemiLabel$geneLabel)

# model summary
summary(res$semiSupervised)

# prediction label for unremarkable genes
head(res$prediction)

# semi and final gene labels
head(res$FinalLabeling)
```

---

sgcp *An example of the output of* ezSGCP *function in the SGCP pipeline*

Description

This is an example of the output from the ezSGCP function, representing the entire SGCP pipeline.Initially, the adjacency matrix is generated using the adjacencyMatrix function within the SGCP framework applied to the cheng dataset. This adjacency matrix serves as input to the clustering function, resulting in the clustering outcome stored in resClus. The clustering result, resClus, is subsequently utilized in the geneOntology function to derive resInitialGO, which captures the initial gene ontology (GO) enrichment results. The resInitialGO output is then processed through the semiLabeling function to produce resSemiLabel, indicating the semi-labeled genes based on their clustering characteristics. This semi-labeled information is further employed in the semiSupervised function, yielding resSemiSupervised, which includes the final supervised classification outcomes for the unremarkable genes. Finally, the results from resSemiSupervised are fed into the geneOntology function once more to generate resFinalGO, which represents the final GO enrichment analysis.

Usage

```
data(sgcp)
```

Format

An object of class `list` containing the `ezSGCP` function information.

Details

`sgcp` contains a list with the following fields:

`clustering`: List of clustering

- `dropped.indices`: Dropped gene indices.
- `geneID`: Vector of geneIDs.
- `method`: Selected method for determining the number of clusters.
- `k`: Selected number of clusters.
- `Y`: Transformed matrix with 2*k columns.
- `X`: Eigenvalues corresponding to the 2*k columns in Y.
- `cluster`: Object of class `kmeans`.
- `clusterLabels`: Vector containing cluster labels for each gene.
- `conductance`: List containing mean, median, and individual cluster conductance indices. Each method's `clusterConductance` field denotes the cluster label with its corresponding conductance index.
- `cvGOdf`: DataFrame used for gene ontology validation. For each method, it shows gene ontology enrichment on the cluster with the smallest conductance index.
- `cv`: String indicating the validation method for the number of clusters (`"cvGO"` for gene ontology validation).
- `clusterNumberPlot`: Object of class `ggplot2` for displaying relativeGap, secondOrderGap, and additiveGap.
- `silhouette`: DataFrame indicating silhouette values for genes.
- `original`: List with matrix transformation, eigenvalues, and `n_egvec`, retaining the top columns of transformation.

`initial.GO`: List of GO term analysis results for initial clusters

- `GOresults`: DataFrame summarizing GO term information.
- `FinalGOTermGenes`: List containing geneIDs of each GO term per cluster.

`semiLabeling`: List of semi-labeling results

- `cutoff`: Numeric indicating selected cutoff.
- `geneLabel`: DataFrame with geneID and corresponding cluster label (or NA if unremarkable).

`semiSupervised`: List of semi-supervised learning results

- `semiSupervised`: Object of classification result.
- `prediction`: Vector of predicted labels for unremarkable genes.
- `FinalLabeling`: DataFrame of geneID with corresponding semi-label and final label.

`final.GO`

: List of GO term analysis results for final modules

- `GOresults`: DataFrame summarizing GO term information.
- `FinalGOTermGenes`: List containing geneIDs of each GO term per cluster.

See Also

SGCP Toturial ezSGCP

Examples

```
library(SGCP)
data(sgcp)
summary(sgcp)

# clustering step
summary(sgcp$clustering)

# intial gene ontology step
summary(sgcp$initial.GO)

# semilabeling step
summary(sgcp$semiLabeling)

# semi-supervised step
summary(sgcp$semiSupervised)

# final gene ontology step
summary(sgcp$final.GO)
```

SGCP_ezPLOT *Comprehensive SGCP plotting in one execution*

Description

The plotting function for ezSGCP results visualizes various aspects. It accepts the ezSGCP output and expression data, generating the following plots: PCA of transformed and expression data, cluster conductance, gene silhouette index, method for determining the number of clusters, distribution and density of gene ontology terms, and cluster performance metrics for both initial clusters and final modules.

Usage

```
SGCP_ezPLOT(sgcp, expreData, keep = FALSE,
        pdf.file = TRUE, pdfname = "ezSGCP.pdf",
        excel.file = TRUE, xlsxname = "ezSGCP.xlsx",
        w = 6, h = 6, sr = 2, sc = 2, ftype = "png",  uni = "in",
        expressionPCA = TRUE, pointSize1 = .5,
        exprePCATitle0 = "Expression Data PCA Without Labels",
        exprePCATitle1 = "Expression Data PCA With Initial Labels",
        exprePCATitle2 = "Expression Data PCA With Final Labels",
        transformedPCA = TRUE, pointSize2 = 0.5,
        transformedTitle0 = "Transformed Data PCA Without Labels",
```

```
transformedTitle1 = "Transformed Data PCA Initial Labels",
transformedTitle2 = "Transformed Data PCA Final Labels",
conduct = TRUE,
conductanceTitle = "Cluster Conductance Index",
conductx = "clusterLabel", conducty = "conductance index",
clus_num = TRUE,
silhouette_index = FALSE,
silTitle = "Gene Silhouette Index",
silx = "genes", sily = "silhouette index",
jitt1 = TRUE,
jittTitle1 = "Initial GO p-values", jps1 = 3,
jittx1 = "cluster", jitty1 = "-log10 p-value",
jitt2 = TRUE,
jittTitle2 = "Final GO p-values", jps2 = 3,
jittx2 = "module", jitty2 = "-log10 p-value",
density1 = TRUE,
densTitle1 = "Initial GO p-values Density",
densx1 = "cluster", densy1 = "-log10 p-value",
density2 = TRUE,
densTitle2 = "Final GO p-values Density",
densx2 = "module", densy2 = "-log10 p-value",
mean1 = TRUE,
meanTitle1 = "Cluster Performance",
meanx1 = "cluster", meany1 = "mean -log10 p-value",
mean2 = TRUE,
meanTitle2 = "Module Performance",
meanx2 = "module", meany2 = "mean -log10 p-value",
pie1 = TRUE, pieTitle1 = "Initial GO Analysis",
piex1 = "cluster", piey1 = "count", posx1 = 1.8,
pie2 = TRUE, pieTitle2 = "Final GO Analysis",
piex2 = "module", piey2 = "count", posx2 = 1.8)
```

Arguments

sgcp	Result from the SGCP pipeline, typically generated by the ezSGCP function.
expreData	Matrix containing the initial gene expression dataset.
keep	Logical, default FALSE. If TRUE, retains plotting objects.
pdf.file	Logical, default TRUE. If TRUE, saves plots in a PDF file.
pdfname	Character string, default "ezSGCP.pdf", name of the PDF file for plots.
excel.file	Logical, default TRUE. If TRUE, saves plots in an Excel file.
xlsxname	Character string, default "ezSGCP.xlsx", name of the Excel file for plots.
w	Numeric, width of plot images in Excel, default 6.
h	Numeric, height of plot images in Excel, default 6.
sr	Numeric, starting row in the Excel sheet, default 2.
sc	Numeric, starting column in the Excel sheet, default 2.
ftype	Character string, plot image type, default "png".

`uni` Character string, plot image units, default "in" for inches.

`expressionPCA` Logical, default `TRUE`. If `TRUE`, plots PCA of gene expression data.

`pointSize1` Numeric, point size for expression PCA, default 0.5.

`exprePCATitle0` Character string, title for expression PCA plot without labels, default "Expression Data PCA Without Labels".

`exprePCATitle1` Character string, title for expression PCA plot with initial cluster labels, default "Expression Data PCA With Initial Labels".

`exprePCATitle2` Character string, title for expression PCA plot with final module labels, default "Expression Data PCA With Final Labels".

`transformedPCA` Logical, default `TRUE`. If `TRUE`, plots PCA of transformed data.

`pointSize2` Numeric, point size for transformed PCA, default 0.5.

`transformedTitle0` Character string, title for PCA plot without labels on transformed data, default "Transformed Data PCA Without Labels".

`transformedTitle1` Character string, title for PCA plot with initial cluster labels on transformed data, default "Transformed Data PCA Initial Labels".

`transformedTitle2` Character string, title for PCA plot with final labels on transformed data, default "Transformed Data PCA Final Labels".

`conduct` Logical, default `TRUE`. If `TRUE`, plots conductance indices for clusters.

`conductanceTitle` Character string, title for conductance indices plot, default "Cluster Conductance Index".

`conductx` Character string, x-axis title in conductance indices plot, default "clusterLabel".

`conducty` Character string, y-axis title in conductance indices plot, default "conductance index".

`clus_num` Logical, default `TRUE`. If `TRUE`, plots cluster number selection methods.

`silhouette_index` Logical, default `FALSE`. If `TRUE`, plots silhouette indices for genes.

`silTitle` Character string, title for silhouette indices plot, default "Gene Silhouette Index".

`silx` Character string, x-axis title in silhouette plot, default "genes".

`sily` Character string, y-axis title in silhouette indices plot, default "silhouette index".

`jitt1` Logical, default `TRUE`. If `TRUE`, plots jitter plot of -log10 p-values of GO terms in initial clusters.

`jps1` Numeric, point size in jitter plot for initial clusters, default 3.

`jittTitle1` Character string, title for jitter plot for initial clusters, default "Initial GO p-values".

`jittx1` Character string, legend for jitter plot for initial clusters, default "cluster".

`jitty1` Character string, y-axis title in jitter plot for initial clusters, default "-log10 p-value".

jitt2	Logical, default TRUE. If TRUE, plots jitter plot of -log10 p-values of GO terms in final modules.
jps2	Numeric, point size in jitter plot for final modules, default 3.
jittTitle2	Character string, title for jitter plot for final modules, default "Final GO p-values".
jittx2	Character string, legend for jitter plot for final modules, default "module".
jitty2	Character string, y-axis title in jitter plot for final modules, default "-log10 p-value".
density1	Logical, default TRUE. If TRUE, plots density plot of p-values of GO terms in initial clusters.
densTitle1	Character string, title for density plot for initial clusters, default "Initial GO p-values Density".
densx1	Character string, legend for density plot for initial clusters, default "cluster".
densy1	Character string, y-axis title in density plot for initial clusters, default "-log10 p-value".
density2	Logical, default TRUE. If TRUE, plots density plot of p-values of GO terms in final modules.
densTitle2	Character string, title for density plot for final modules, default "Final GO p-values Density".
densx2	Character string, legend for density plot for final modules, default "module".
densy2	Character string, y-axis title in density plot for final modules, default "-log10 p-value".
mean1	Logical, default TRUE. If TRUE, plots mean over p-values of GO terms in initial clusters.
meanTitle1	Character string, title for mean plot for initial clusters, default "Cluster Performance".
meanx1	Character string, legend for mean plot for initial clusters, default "cluster".
meany1	Character string, y-axis title in mean plot for initial clusters, default "mean -log10 p-value".
mean2	Logical, default TRUE. If TRUE, plots mean over p-values of GO terms in final modules.
meanTitle2	Character string, title for mean plot for final modules, default "Module Performance".
meanx2	Character string, legend for mean plot for final modules, default "module".
meany2	Character string, y-axis title in mean plot for final modules, default "mean -log10 p-value".
pie1	Logical, default TRUE. If TRUE, plots pie chart of direction and ontology of GO terms for initial clusters.
pieTitle1	Character string, title for pie plot for initial clusters, default "Initial GO Analysis".
piex1	Character string, x-axis title for pie plot for initial clusters, default "cluster".

piey1	Character string, y-axis title in pie plot for initial clusters, default "count".
posx1	Numeric, position of label of -log10 p-value of the most significant term, default 1.8.
pie2	Logical, default TRUE. If TRUE, plots pie chart of direction and ontology of GO terms for final modules.
pieTitle2	Character string, title for pie plot for final modules, default "Final GO Analysis".
piex2	Character string, x-axis title for pie plot for final modules, default "module".
piey2	Character string, y-axis title in pie plot for final modules, default "count".
posx2	Numeric, position of label of -log10 p-value of the most significant term, default 1.8.

Value

Returns the plotting object for each plot if keep is TRUE.

References

Aghaieabiane, N and Koutis, I (2024) SGCP: a spectral self-learning method for clustering genes in co-expression networks

See Also

SGCP Toturial

Examples

```
library(SGCP)
library(SummarizedExperiment)

# load the result of ezSGCP function
data(sgcp)

# load the input expression dataset
data(cheng)
expData <- assay(cheng)

# to call the function uncomment the following
## plt <- SGCP_ezPLOT(sgcp = sgcp, expreData = cheng, keep = TRUE)

## print(plt)
```

SGCP_plot_bar *Mean p-value bar chart for gene ontology enrichment in the SGCP pipeline*

Description

Generates a bar chart illustrating the average p-values from gene ontology enrichment across the SGCP pipeline.

Usage

```
SGCP_plot_bar(df, tit = "mean -log10 p-values",
              xname = "module", yname = "-log10 p-value")
```

Arguments

df	The GOresults dataframe returned by the geneOntology function in the SGCP pipeline.
tit	Plot title (default: "Mean -log10 p-values")
xname	X-axis title (default: "module")
yname	Y-axis title (default: "-log10 p-value")

Value

returns the plot, an object of class ggplot2.

References

Aghaieabiane, N and Koutis, I (2024) SGCP: a spectral self-learning method for clustering genes in co-expression networks

See Also

geneOntology SGCP_ezPLOT SGCP Toturial

Examples

```
library(SGCP)
# load the output of geneOntology function
data(resInitialGO)

# call the function

plt <- SGCP_plot_bar(df = resInitialGO$GOresults)
print(plt)
```

SGCP_plot_conductance *Cluster conductance index bar chart in the SGCP Pipeline*

Description

Generates a bar chart displaying the cluster conductance index in the SGCP pipeline.

Usage

```
SGCP_plot_conductance(conduct, tit = "Clustering Conductance Index",
                      xname = "cluster", yname = "conductance")
```

Arguments

conduct	The conductance field returned by the clustering function in the SGCP pipeline.
tit	Plot title (default: "Clustering Conductance Index")
xname	X-axis title (default: "cluster")
yname	Y-axis title (default: "conductance")

Value

returns the plot, an object of class ggplot2.

References

Aghaieabiane, N and Koutis, I (2024) SGCP: a spectral self-learning method for clustering genes in co-expression networks

See Also

clustering SGCP_ezPLOT SGCP Toturial

Examples

```
library(SGCP)
# load the output of geneOntology function
data(resClus)

# call the function

plt <- SGCP_plot_conductance(conduct = resClus$conductance)
print(plt)
```

SGCP_plot_density *Visualization of gene ontology term p-value distribution in the SGCP pipeline*

Description

Generates a density chart displaying p-values of gene ontology terms in the SGCP pipeline.

Usage

```
SGCP_plot_density(df, tit = "p-values Density",
                  xname = "module", yname = "-log10 p-value")
```

Arguments

df	The GOresults dataframe returned by the geneOntology function in the SGCP pipeline.
tit	Plot title (default: "p-values Density")
xname	X-axis title (default: "module")
yname	Y-axis title (default: "-log10 p-value")

Value

returns the plot, an object of class ggplot2.

References

Aghaieabiane, N and Koutis, I (2024) SGCP: a spectral self-learning method for clustering genes in co-expression networks

See Also

geneOntology SGCP_ezPLOT SGCP Toturial

Examples

```
library(SGCP)
# load the output of geneOntology function
data(resInitialGO)

# call the function

plt <- SGCP_plot_density(df = resInitialGO$GOresults)
print(plt)
```

SGCP_plot_heatMap *Adjacency matrix heatmap in the SGCP pipeline*

Description

Generates a Heatmap of the the adjacency matrix (network) in the SGCP pipeline.

Usage

```
SGCP_plot_heatMap(m, tit = "Adjacency Heatmap",
    xname = "genes", yname = "genes")
```

Arguments

m	An adjacency matrix returned by the adjacencyMatrix function in the SGCP pipeline, or any symmetric matrix with values in (0, 1) excluding the diagonal.
tit	Plot title (default: "Adjacency Heatmap")
xname	X-axis title (default: "genes")
yname	Y-axis title (default: "genes")

Value

returns the plot, an object of class ggplot2.

References

Aghaieabiane, N and Koutis, I (2024) SGCP: a spectral self-learning method for clustering genes in co-expression networks

See Also

adjacencyMatrix SGCP_ezPLOT SGCP Toturial

Examples

```
library(SGCP)
GeneExpression <- matrix(runif(200, 0,1), nrow = 40, ncol = 5)
diag(GeneExpression) <- 0

## call the function
adja <- adjacencyMatrix(GeneExpression)

plt <- SGCP_plot_heatMap(m =  adja)
print(plt)
```

SGCP_plot_jitter *P-value jitter chart for gene ontology enrichment in the SGCP pipeline.*

Description

Generates a jitter chart illustrating the cluster gene ontology enrichment in the SGCP pipeline.

Usage

```
SGCP_plot_jitter(df, tit = "p-values Distribution",
                 xname = "module", yname = "-log10 p-value", ps = 3)
```

Arguments

df	The GOresults dataframe returned by the geneOntology function in the SGCP pipeline.
tit	Plot title (default: "p-values Distribution")
xname	X-axis title (default: "module")
yname	Y-axis title (default: "-log10 p-value")
ps	Numeric value for point size (default: 3)

Value

returns the plot, an object of class ggplot2.

References

Aghaieabiane, N and Koutis, I (2024) SGCP: a spectral self-learning method for clustering genes in co-expression networks

See Also

geneOntology SGCP_ezPLOT SGCP Toturial

Examples

```
library(SGCP)
# load the output of geneOntology function
data(resInitialGO)

# call the function

plt <- SGCP_plot_jitter(df = resInitialGO$GOresults)
print(plt)
```

SGCP_plot_pca *PCA visualization in the SGCP Pipeline*

Description

Generates Principal Component Analysis (PCA) of the gene expression and transformed gene expression; comparision with and without labels.

Usage

```
SGCP_plot_pca(m, clusLabs, tit = "PCA plot", ps = .5)
```

Arguments

m	A numeric matrix of size n*m.
clusLabs	Either NULL or a vector of size n indicating cluster labels. There is a 1-to-1 correspondence between the rows in m and clusLabs.
tit	Plot title (default: "PCA plot")
ps	Point size (default: 0.5)

Value

returns the plot, an object of class ggplot2.

References

Aghaieabiane, N and Koutis, I (2024) SGCP: a spectral self-learning method for clustering genes in co-expression networks

See Also

SGCP_ezPLOT SGCP Toturial

Examples

```
library(SGCP)
GeneExpression <- matrix(runif(200, 0,1), nrow = 40, ncol = 5)
diag(GeneExpression) <- 0

## call the function
plt <- SGCP_plot_pca(m = GeneExpression, clusLabs = NULL)

print(plt)
```

SGCP_plot_pie *Gene ontology analysis pie chart in the SGCP pipeline*

Description

Generate a pie chart illustrating the ontology and test direction of gene ontology terms across the SGCP pipeline

Usage

```
SGCP_plot_pie(df, tit = "GO Analysis",
                xname = "module", yname = "count", posx = 1.9)
```

Arguments

df	The GOresults dataframe returned by the geneOntology function in the SGCP pipeline.
tit	Plot title (default: "GO Analysis")
xname	X-axis title (default: "module")
yname	Y-axis title (default: "count")
posx	Numeric value for label position in the pie chart. A higher number places labels further from the pie chart.

Value

returns the plot, an object of class ggplot2.

References

Aghaieabiane, N and Koutis, I (2024) SGCP: a spectral self-learning method for clustering genes in co-expression networks

See Also

geneOntology SGCP_ezPLOT SGCP Toturial

Examples

```
library(SGCP)
# load the output of geneOntology function
data(resInitialGO)

# call the function

plt <- SGCP_plot_pie(df = resInitialGO$GOresults)
print(plt)
```

SGCP_plot_silhouette *Cluster silhouette index chart in the SGCP Pipeline*

Description

Generates a chart displaying the cluster silhouette index in the SGCP pipeline.

Usage

```
SGCP_plot_silhouette(df, tit = "Gene Silhouette Index",
                     xname = "genes", yname = "silhouette index")
```

Arguments

df	The silhouette dataframe returned by the clustering function in the SGCP pipeline.
tit	Plot title (default: "Gene Silhouette Index")
xname	X-axis title (default: "genes")
yname	Y-axis title (default: "silhouette index")

Details

In order to plot silhouette index, sil argument in the clustering function must be set to TRUE.

Value

returns the plot, an object of class ggplot2.

References

Aghaieabiane, N and Koutis, I (2024) SGCP: a spectral self-learning method for clustering genes in co-expression networks

See Also

clustering SGCP_ezPLOT SGCP Toturial

Examples

```
library(SGCP)
data(resClus)

## call the function
plt <- SGCP_plot_silhouette(df = resClus$silhouette)

print(plt)
```

Index