Package ‘SGCP’

October 16, 2023

Type Package

Title SGCP: A semi-supervised pipeline for gene clustering using self-training approach in gene co-expression networks

Version 1.0.0

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, BiocManager

Depends R (>= 4.3.0)

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

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

git_branch RELEASE_3_17

git_last_commit 2c0861e

git_last_commit_date 2023-04-25

Date/Publication 2023-10-15

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

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

R topics documented:

adjacencyMatrix 2
cheng 4
clustering 5
ezSGCP 7
geneOntology 11
resClus 13
resFinalGO 14
resInitialGO 15
resSemiLabel 16
resSemiSupervised 17
semiLabeling 18
semiSupervised 19
sgcp 20
SGCP_ezPLOT 22
SGCP_plot_bar 26
SGCP_plot_conductance 27
SGCP_plot_density 28
SGCP_plot_heatMap 29
SGCP_plot_jitter 30
SGCP_plot_pca 31
SGCP_plot_pie 32
SGCP_plot_silhouette 33

Index **35**

`adjacencyMatrix` *Performs Netwrok Construction step In SGCP Pipeline*

Description

It creates the adjanecy matrix of gene co-expression network in SGCP pipeline. Here, users can select steps. The order of steps are calibration, norm, Gaussian kernel, and tom. If calibration is TRUE, then SGCP will perform calibration at first (please see the manuscript for more information). Next, if norm is TRUE, SGCP will divide each gene by its norm2. Then, SGCP will calculate Gaussian kernel metric as the similarity function to calculate the pairwise gene similarity values, this step is mandatory and user cannot change it. If tom is TRUE, SGCP will add the information of the second order of the node neighborhood to the network. At the end, SGCP returns a symmetric adjacency matrix adja of size n*n where n is the number of genes. All values in the adjancency matrix range from 0 to 1, where 1 is the most similar. The diagonal of the returning matrix is 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, rows correspond to genes and columns to samples.
calibration	boolean, default FALSE, if TRUE it performs calibration step.
norm	boolean, default TRUE, if TRUE will divide each genes (rows) by its norm2.
tom	boolean, default TRUE, if TRUE it adds TOM to the network.
saveAdja	boolean, default FALSE, if TRUE, the adjancency matrix will be saved.
adjaNameFile	string indicates the name of file for saving adjancency matrix.
hm	string indicates the name of file for saving adjancency matrix heat map.

Value

adja	a matrix of dimension n * n of the adjacency matrix where n is the number of the genes. This matrix is symmetric and entry values are in (0,1) with 0 diagonal

References

Aghaieabiane, N and Koutis, I (2022) SGCP: A semi-supervised pipeline for gene clustering using self-training approach in gene 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 Of ischemic cardiomyopathy (ICM) from a publication by Cheng et al.*

Description

This is a normalized gene expression data of 1500 genes * 5 samples. This data is a subset of a larger gene expression of ischemic cardiomyopathy (ICM) with 5000 genes and 57 samples. Gene expression data is normalized using the DESeq method, based on median ratio of gene counts.

Usage

```
data(cheng)
```

Format

An object of class `SummarizedExperiment`.

Details

assays contains the gene expression data and rowData field contains 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 SGCP Pipeline*

Description

It performs clustering on the adjacency network of gene co-expression network in SGCP pipeline. To this end, it firstly transforms the adjacency matrix of size n*n into new dimension Y of n*n using gaph Laplacian. It then, calculates number of cluster k based on three methods "relativeGap", "secondOrderGap", and "additiveGap". For each method, it performs kmeans on Y and the corresponding k as the input. Then it calculates conductance index for the clusters in the methods, and for each method, it picks the cluster that has smallest conductance index. Finally it performs gene ontology enrichment on those selected clusters to finalize k. At the end, it returns the result of the kmeans clustering on the selected method along with the transformed matrix Y, and some additional information. At this step, initial clusters are produced.

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	adjacency matrix of n*n where n is the number of the genes, this matrix is squared symmetric with values in (0, 1) and 0 diagonal. It is the output of adjacencyMatrix function of SGCP.
geneID	a vector containing the genes IDs of size n where n is the number of genes.
annotation_db	a string indicating the genomic wide annotation database.
kopt	an integer denotes the optimal number of clusters k by the user, default is NULL.
method	method for identifying the number of clusters k, default NULL, either "relativeGap", "secondOrderGap", "additiveGap", or NULL.
func.GO	a function for gene ontology validation, default is sum.
func.conduct	a function for conductance validation, default is min.
maxIter	an integer, identifies the maximum number of iteration for kmeans.
numStart	an integer, identifies the number of start for kmeans.
eff.egs	boolean, default TRUE, if TRUE, used `eigs_sym` ti calculate the eigenvalues and eigenvectors, more efficient than R default function
saveOrig	boolean, default TRUE, if TRUE, keeps the transformation matrix.
n_egvec	an integer less than 200, default = 200, indicates the number of columns of transformation matrix to be kept
sil	boolean, default FALSE, if TRUE, calculates silhouette index for each cluster.

Details

If `kopt` is not null, SGCP will find clusters based on `kopt`. Otherwise, if method is not null, SGCP will pick k based on the selected method. Otherwise, if `geneID` or `annotation_db` is null, SGCP will determine the optimal method and its corresponding number of cluster based on condutance validation. It picks a method that `func.conduct` on its cluster is minimum. Otherwise, SGCP will use gene ontology validation (by default) to find the optimal method and its corresponding number of clusters. To this end, it will perform gene ontology enrichment on the cluster with minimum conductanc index per method and pick the one that has the maximum `func.GO` over -log10 of p-values.

Value

a list containing some of the following depending on the initial call

`dropped.indices`
: a vector of dropped gene indices.

`geneID`
: a vector of geneIDs.

`method`
: indicates the selected method for number of cluster.

`k`
: selected number of clusters.

`Y`
: transformed matrix with 2*k columns.

`X`
: eigenvalues correspond to 2*k columns in Y.

`cluster`
: an object of class `kmeans`.

`clusterLabels`
: a vector containing the cluster label per gene, there is a 1-to-1 correspondence between geneID and clusterLabes

`conductance`
: a list containing mean and median, and individual cluster conductance index for clusters per method. 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 minimum conductance index.

`cv`
: an string indicates the validation method for number of cluster, "cvGO": if gene ontology validation used, "cvConductance": if conductance validation used, "userMethod": if user defined the method, "userkopt": if user defines the kopt.

`clusterNumberPlot`
: an object of class `ggplot2` for relativeGap"", "secondOrderGap", and "additiveGap".

`silhouette`
: a dataframe that indicates the silhouette for genes.

`original`
: a list with matrix transformation and corresponding eigenvalues and n_egvec, where n_egvec top columns of transformation is kept.

References

Aghaieabiane, N and Koutis, I (2022) SGCP: A semi-supervised pipeline for gene clustering using self-training approach in gene 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 *Performs All SGCP pipeline In One Step*

Description

On step SGCP pipeline for gene co-expression network construction and analysis. It takes the gene expression and gene IDs, along with annotation_db and performs all steps of SGCP in a single function. It firstly perform network construction step and build the adjacency matrix. It then apply network clustering step using to find the intial clusters. Using gene ontology enrichent step, it finds and divides the genes into set of remarkable and unremarkable and use this for the next step to semi-label the data and convert the problem into smei-supervised. It used the remarkable genes as the training set to train a supervised model and make prediction for unremarakable gene and produce the final modules. Finaly, it performs one more gene ontology step to see the 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, rows correspond to genes and columns to samples.
geneID	a vector containing the genes IDs of size n where n is the number of genes.
annotation_db	a string indicating the genomic wide annotation database.
semilabel	Boolean, default TRUE, if TRUE, semilabeling step will be performed.
calib	boolean, default FALSE, if TRUE it performs calibration step.
norm	boolean, default TRUE, if TRUE will divide each genes (rows) by its norm2.
tom	boolean, default TRUE, if TRUE it adds TOM to the network.
saveAdja	boolean, default FALSE, if TRUE, the adjancency matrix will be saved.
adjaNameFile	string indicates the name of file for saving adjancency matrix.
hm	string indicates the name of file for saving adjancency matrix heat map.
kopt	an integer denotes the optimal number of clusters k by the user, default is NULL.
method_k	method for identifying the number of clusters k, default NULL, either "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, identifies the maximum number of iteration for kmeans.
numberStart	an integer, identifies the number of start for kmeans.
eff.egs	a boolean, default TRUE, if TRUE it uses `eigs_sym` to calculate the eigenvalues and eigenvectors, more efficient than R default function
saveOrig	boolean, default TRUE, if TRUE, keeps the transformation matrix.
n_egvec	either "all" or an integer, default = 100, indicates the number of columns of transformation matrix to be kept
sil	boolean, default FALSE, if TRUE, calculates silhouette index for each cluster.
dir	test direction, default c("over", "under"), for over-represented, or under-represented GO terms.
onto	GO ontologies, default c("BP", "CC", "MF"), BP: Biological Process, CC: Cellular Component, MF: Molecular Function.
hgCut	a numeric value in (0,1) as the p-value cutoff, default 0.05, GO terms smaller than hgCutoff value are kept.
condTest	Boolean, default TRUE, if TRUE conditional hypergeometric test is performed.
cutoff	a numeric in (0, 1) default NULL, is a base line for GO term significancy.

`percent` a number in (0,1) default 0.1, indicate the percentile for finding top GO terms.

`stp` a number in (0,1) default 0.01, indicates increasing value to be added to percent parameter.

`model` either "knn" or "lr" for classification model, knn: k nearest neighbors, lr: logistic regression.

`kn` an integer default NULL indicating the number of neighbors in knn, if kn is NULL, then kn = 20 : (20 + 2 * k) if 2 * k < 30 otherwise 20 : 30, where k is the numbber of remarkable cluster

Details

For clustering step, if kopt is not null, SGCP will find clusters based on kopt. Otherwise, if method is not null, SGCP will pick k based on the selected method. Otherwise, if geneID or annotation_db is null, SGCP will determine the optimal method and its corresponding number of cluster based on condutance validation. It picks a method that func.conduct on its cluster is minimum. Otherwise, SGCP will use gene ontology validation (by default) to find the optimal method and its corresponding number of clusters. To this end, it will perform gene ontology enrichment on the cluster with minimum conductanc index per method and pick the one that has the maximum func.GO over -log10 of p-values. In semilabeling step, gene associated to the GO terms more significant than cutoff value are remarkable. If cutoff value is NULL, SGCP will find the cutoff depend on the GO terms significant level. Otherwise, SGCP picks the top percent (by default 0.1) GO terms from all clusters collectively, and consider the genes associated to those as remakable. If all remarkable genes come from a single cluster, then SGCP will increase the precent by 0.01 to find the remarkable and unremarkable genes. It repeats this process until all remarkable genes come from at least two clusters. In semi-supervise step, remarkable clusters are the clusters that have at least one remarkable gene.

Value

It returns a list of clustering, initial.GO, semiLabeling, semiSupervised, final.GO fields, which contains the infromation of corresponding step.

`semilabel` Boolean, indicates if semilabeling step is performed.

`clusterLabels` a dataframe with geneID and its corresponding initial and final labels.

clustering field, a list of

`dropped.indices` dropped gene indices.

`geneID` a vector of geneIDs.

`method` indicates the selected method for number of cluster.

`k` selected number of clusters.

`Y` transformed matrix with 2*k columns.

`X` eigenvalues correspond to 2*k columns in Y.

`cluster` object rof class `kmeans`.

`clusterLabels` a vector containing the cluster label, for each gene, there is a 1-to-1 correspondance between geneID and clusterLabes.

`conductance` a list containing mean and median, and individual cluster conductance index for clusters. In each method, the clusterConductance field denote the cluster label with its corresponding conductance index.

`cvGOdf` a dataframe used for gene ontology validation, for each method, it shows the gene ontology enrichment on the cluster with smallest conductance index.

`cv` an string indicates the validation method for number of cluster, "cvGO": if gene ontology validation used, "cvConductance": if conductance validation used, "userMethod": if user defined the method, "userkopt": if user defines the kopt

`clusterNumberPlot` an objet of class `ggplot2` for relativeGap"", "secondOrderGap", and "additiveGap".

`silhouette` a dataframe that indicates the silhouette for genes.

`original` a list with matrix transformation and corresponding eigenvalues and n_egvec, where n_egvec top columns of tranformation is kept.

initial.GO field, a list of

`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.

semiLabeling field, a list of

`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.

semiSupervised field which is a list of

`semiSupervised` an object of classification result.

`prediction` a vector of predicted labels for unremarkable genes.

`FinalLabeling` a dataframe of geneID with its corresponding semilabel and final label.

final.GO field, a list of

`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 (2022) SGCP: A semi-supervised pipeline for gene clustering using self-training approach in gene 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 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 (2022) SGCP: A semi-supervised pipeline for gene clustering using self-training approach in gene 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 output of* `clustering` *function in SGCP pipeline.*

Description

This is an example of `clustering` function output. Firstly, the adjacency matrix is produced using `adjacencyMatrix` function in SGCP over cheng dataset. The matrix is then used in~`clustering` function to produce the clustering result.

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 geneIDs, `method`:indicates the selected method for number of cluster,k:selected number of clusters, `Y`: transformed matrix with 2*k columns, `X`:eigenvalues correspond to 2*k columns in Y, `cluster`:an object of class `kmeans`, `clusterLabels`:a vector containing the cluster label per gene, there is a 1-to-1 correspondence between geneID and clusterLabes, `conductance`:a list containing mean and median, and individual cluster conductance index for clusters per method. Index in ‘clusterConductance‘ field denotes the `cvGOdf`:a dataframe used for gene ontology validation. For each method, it returns the gene ontology enrichment result on the cluster with minimum conductance index, `cv`: an string indicates the validation method for number of cluster, "cvGO" means gene ontology validation used, `clusterNumberPlot`:an object of class `ggplot2` for relativeGap"", "secondOrderGap", and "additiveGap", `silhouette`:a dataframe that indicates the silhouette for genes,`original`:a list with matrix transformation and corresponding eigenvalues and n_egvec, where n_egvec top columns of transformation is kept.

See Also

SGCP Toturial `adjacencyMatrix` `clustering`

Examples

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

resFinalGO *An example of output of* geneOntololgy *function in SGCP pipeline.*

Description

This is an example of geneOntology function output as the last step in SGCP pipeline. Firstly, the adjacency matrix is produced using adjacencyMatrix function in SGCP over cheng dataset. The matrix is then used in~clustering function to produce the clustering result resClus. resClus is then used in geneOnology to produce~resInitialGO. This result is fed to semiLabeling to produce~resSemiLabel. This is result is used as input to~semiSupervised function to produce resSemiSupervised. At the end this is used in~geneOntology funtion to produce resFinalGO.

Usage

```
data(resFinalGO)
```

Format

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

Details

resFinalGO is a list containin 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 ignificant 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 output of* `geneOntololgy` *function in SGCP pipeline.*

Description

This is an example of `geneOntology` function output as the third step in SGCP pipeline. Firstly, the adjacency matrix is produced using `adjacencyMatrix` function in SGCP over cheng dataset. The matrix is then used in~`clustering` function to produce the clustering result `resClus`. resClus is then used in `geneOnology` to produce~`resInitialGO`.

Usage

```
data(resInitialGO)
```

Format

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

Details

`resInitialGO` is a list containin 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 ignificant 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 output of* `semiLabeling` *function in SGCP pipeline.*

Description

This is an example of `geneOntology` function output as the last step in SGCP pipeline. Firstly, the adjacency matrix is produced using `adjacencyMatrix` function in SGCP over cheng dataset. The matrix is then used in~`clustering` function to produce the clustering result `resClus`. resClus is then used in `geneOnology` to produce~`resInitialGO`. This result is fed to `semiLabeling` to produce~`resSemiLabel`.

Usage

```
data(resSemiLabel)
```

Format

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

Details

`resSemiLabel` is a list containin 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 output of* semiSupervised *function in SGCP pipeline.*

Description

This is an example of geneOntology function output as the last step in SGCP pipeline. Firstly, the adjacency matrix is produced using adjacencyMatrix function in SGCP over cheng dataset. The matrix is then used in~clustering function to produce the clustering result resClus. resClus is then used in geneOnology to produce~resInitialGO. This result is fed to semiLabeling to produce~resSemiLabel. This is result is used as input to~semiSupervised function to produce resSemiSupervised.

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 SGCP Pipeline*

Description

Peforms Semi-labeling step and indentifies remarkable and unremarkable genes in SGCP pipeline. It collects all gene ontology (GO) terms from all clusters and pick the terms in top 0.1 percent. It considered the genes associated to those terms as remarkable, and the remaining as unremarkable.

Usage

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

Arguments

geneID	a vector containing the genes IDs of size n, where n is the number of genes.
df_GO	GOresults dataframe returned by geneOntology function, consists the information of GO terms the clusters.
GOgenes	FinalGOTermGenes list returedn by geneOntology function, is a list of genes associated to the GO Terms per each cluster.
cutoff	a numeric in (0, 1) default NULL, is a base line for GO term significancy.
percent	a number in (0,1) default 0.1, indicate the percentile for finding top GO terms.
stp	a number in (0,1) default 0.01, indicates increasing value to be added to percent parameter.

Details

Gene associated to the GO terms more significant than cutoff value are remarkable. If cutoff value is NULL, SGCP will find the cutoff depend on the GO terms significant level. Otherwise, SGCP picks the top percent (by default 0.1) GO terms from all clusters collectively, and consider the genes associated to those as remakable. If all remarkable genes come from a single cluster, then SGCP will increase the precent by 0.01 to find the remarkable and unremarkable genes. It repeats this process until all remarkable genes come 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 (2022) SGCP: A semi-supervised pipeline for gene clustering using self-training approach in gene 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 Semi-supervised step In SGCP Pipeline*

Description

Performs semi-supervised classification step in SGCP pipline. It takes the transformed matrix from clustering function along with gene semi-labels from semiLabeling function, and use the labeled (remarkable) genes as the training set to trian either "k nearest neighbor" or "logistic regression" model and make prediction for unlabeled (unremarkable genes). At this step, final modules are produced.

Usage

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

Arguments

specExp	matrix or dataframe with genes in rows and features in columns, this is Y matrix from clustering function output.
geneLab	a dataframe returned by semiLabeling function, contains the geneID and its corresponding label if is remarkable otherwise NA.
model	either "knn" or "lr" for classification model, knn: k nearest neighbors, lr: logistic regression.
kn	an integer default NULL indicating the number of neighbors in knn, if kn is NULL, then kn = 20 : (20 + 2 * k) if 2 * k < 30 otherwise 20 : 30, where k is the numbber of remarkable cluster

Details

remarkable clusters are the clusters that have at least one remarkable gene.

Value

semiSupervised	an object of caret train class.
prediction	a vector of predicted labels for unremarkable genes.
FinalLabeling	a dataframe of geneID with its corresponding semilabel and final label.

References

Aghaieabiane, N and Koutis, I (2022) SGCP: A semi-supervised pipeline for gene clustering using self-training approach in gene 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 output of* zSGCP *function in SGCP pipeline.*

Description

This is an example of ezSGPC function output. This function is the automatic SGCP.

Usage

```
data(sgcp)
```

Format

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

Details

`sgcp` contains a list of clustering, initial.GO, semiLabeling, semiSupervised, final.GO fields, which contains the infromation of corresponding step. `semilabel`:Boolean, indicates if semilabeling step is performed, `clusterLabels`:a dataframe with geneID and its corresponding initial and final labels. In the clustering field, following information is provided; `dropped.indices`:dropped gene indices, `geneID`:a vector of geneIDs, `method`:indicates the selected method for number of cluster, `k`:selected number of clusters, `Y`: transformed matrix with 2*k columns, `X`:eigenvalues correspond to 2*k columns in Y, `cluster`:object rof class `kmeans`, `clusterLabels`:a vector containing the cluster label, for each gene, there is a 1-to-1 correspondance between geneID and clusterLabes, `conductance`:a list containing mean and median, andindividual cluster conductance index for clusters. In each method, the clusterConductance field denote the cluster label with its corresponding conductance index, `cvGOdf`:a dataframe used for gene ontology validation, for each method, it shows the gene ontology enrichment on the cluster with smallest conductance index, `cv`:an string indicates the validation method for number of cluster, "cvGO" shows that gene ontology validation used, `clusterNumberPlot`:an objet of class `ggplot2` for relativeGap"", "secondOrderGap", and "additiveGap", `silhouette`:a dataframe that indicates the silhouette for genes, `original`:a list with matrix transformation and corresponding eigenvalues and n_egvec, where n_egvec top columns of tranformation is kept, initial.GO field, a list of `GOresults`:a dataframe containing the summary of the information of GOTerms, `FinalGOTermGenes`:a list containing the geneIDs of each GOTerms per cluster, semiLabeling field, a list of `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, semiSupervised field which is a list of `semiSupervised`:an object of classification result, `prediction`:a vector of predicted labels for unremarkable genes, `FinalLabeling`:a dataframe of geneID with its corresponding semilabel and final label final.GO field, a list of `GOresults`:a dataframe containing the summary of the information of GOTerms, `FinalGOTermGenes`:a list containing the geneIDs of each GOTerms 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 *Performs All SGCP Plots In One Step*

Description

On step plotting function for ezSGCP result. It takes the result from ezSGCP along with the expression data, and plot PCA of transformed and expression data, cluster conductance, gene silhouette index, method for number of clusters, distribution of gene ontology terms, density of gene ontology terms, cluster performance for both intial 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	a returning result from ezSGCP function.
expreData	a matrix of initial gene expression dataset.
keep	Boolean, default FALSE. If TRUE plotting objects are kept.
pdf.file	Boolean, default TRUE, if TRUE it stores the plots in a pdf file.
pdfname	name of pdf file, default "ezSGCP.pdf".
excel.file	Boolean, default TRUE, if TRUE it stores the plots in a excel file.
xlsxname	name of pdf file, default "ezSGCP.xlsx".
w	width of plot images in excel, default 6.
h	height of plot images in excel, default 6.
sr	starting row in an excel sheet, default 2.
sc	starting column in an excel sheet, default 2.
ftype	plot image type, default "png".
uni	plot image units , default "in" for inch.
expressionPCA	Boolean, default TRUE, if TRUE PCA of gene expression data is plotted.
pointSize1	point size in for expression PCA, default 0.5.
exprePCATitle0	a string for expression PCA plot title without labels, default "Expression Data PCA Without Labels".
exprePCATitle1	a string for expression PCA plot title with initial cluster labels, default "Expression Data PCA With Initial Labels".
exprePCATitle2	a string for expression PCA plot title with final module labels, default "Expression Data PCA With Final Labels".
transformedPCA	Boolean, default TRUE, if TRUE PCA of transformed data is plotted.
pointSize2	point size in for transformed PCA, default 0.5.

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

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

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

`conduct`
: Boolean, default TRUE, if TRUE conductance indices for clusters are plotted.

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

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

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

`clus_num`
: Boolean, default TRUE, if TRUE cluster numbers method are plotted.

`silhouette_index`
: Boolean, default FALSE, if TRUE silhouette indices for genes are plotted.

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

`silx`
: a string for x-axis title in silhouette plot , default "genes".

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

`jitt1`
: Boolean, default TRUE, if TRUE jitter plot of p-values of GO terms in intial clusters are plotted.

`jps1`
: point size in jiiter plot for initial clusters, default 3.

`jittTitle1`
: a string for jiiter plot title for initial clusters, default "Initial GO p-values".

`jittx1`
: a string for jiiter plot for initial clusters legend , default "cluster".

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

`jitt2`
: Boolean, default TRUE, if TRUE jitter plot of p-values of GO terms in final modules are plotted.

`jps2`
: point size in jiiter plot for final modules, default 3.

`jittTitle2`
: a string for jiiter plot title for final modules, default "Final GO p-values".

`jittx2`
: a string for jiiter plot for final modules legend , default "module".

`jitty2`
: string for y-axis title in jiiter plot for final modules, default "-log10 p-value".

`density1`
: Boolean, default TRUE, if TRUE density plot of p-values of GO terms in initial clusters are plotted.

`densTitle1`
: a string for density plot title for initial clusters, default "Initial GO p-values Density".

`densx1`
: a string for density plot for initial clusters legend , default "cluster".

`densy1`
: string for y-axis title in desnity plot for initial clusters, default "-log10 p-value".

`density2`
: Boolean, default TRUE, if TRUE density plot of p-values of GO terms in final modules are plotted.

`densTitle2`
: a string for density plot title for final modules, default "Final GO p-values Density".

`densx2`	a string for density plot for final modules legend , default "module".
`densy2`	string for y-axis title in desnity plot for final modules, default "-log10 p-value".
`mean1`	Boolean, default TRUE, if TRUE mean over p-values of GO terms in initial clusters are plotted.
`meanTitle1`	a string for mean plot title for initial clusters, default "Cluster Performance".
`meanx1`	a string for mean plot for initial clusters legend , default "cluster".
`meany1`	string for y-axis title in mean plot for initial clusters, default "mean -log10 p-value".
`mean2`	Boolean, default TRUE, if TRUE mean over p-values of GO terms in final modules are plotted.
`meanTitle2`	a string for mean plot title for final modules, default "Module Performance".
`meanx2`	a string for mean plot for initial clusters legend , default "module".
`meany2`	string for y-axis title in mean plot for final modules, default "mean -log10 p-value".
`pie1`	Boolean, default TRUE, if TRUE pie chart of direction and ontology of GO terms for initial clusters are plotted.
`pieTitle1`	a string for pie plot title for initial clusters, default "Initial GO Analysis".
`piex1`	a string for pie plot x-axis title for initial clusters, default "cluster".
`piey1`	string for y-axis title in pie plot for initial clusters, default "count".
`posx1`	a numeric, default 1.8, position of label of -log10 p-value of the most significant term.
`pie2`	Boolean, default TRUE, if TRUE pie chart of direction and ontology of GO terms for final modules are plotted.
`pieTitle2`	a string for pie plot title for initial clusters, default Final GO Analysis".
`piex2`	a string for pie plot x-axis title for final modules, default "module".
`piey2`	string for y-axis title in pie plot for final modules, default "count".
`posx2`	a numeric, default 1.8, position of label of -log10 p-value of the most significant term.

Value

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

References

Aghaieabiane, N and Koutis, I (2022) SGCP: A semi-supervised pipeline for gene clustering using self-training approach in gene 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 Over Gene Ontology Enrichment p-values In SGCP Pipeline*

Description

Plots the mean over gene ontology enrchment p-values in 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 geneOntology function in SGCP pipeline
tit	plot title, default"mean -log10 p-values"
xname	x-axis title, default "module"
yname	y-axis title, default either "-log10 p-value"

Value

returns the plot, an object of class ggplot2.

References

Aghaieabiane, N and Koutis, I (2022) SGCP: A semi-supervised pipeline for gene clustering using self-training approach in gene 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 *Plots Cluster Conductance Index In SGCP Pipeline*

Description

Plots the cluster conductance index per cluster in SGCP pipeline.

Usage

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

Arguments

conduct	conductance field returned by clustering function in 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 (2022) SGCP: A semi-supervised pipeline for gene clustering using self-training approach in gene 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` *Density Plot Of Gene Ontology Enrichment p-values In SGCP Pipeline*

Description

Plots the density plot of gene ontology enrchment p-values in SGCP pipeline.

Usage

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

Arguments

`df`	the ‘GOresults‘ dataframe returned by geneOntology function in SGCP pipeline
`tit`	plot title, default "p-values Density"
`xname`	x-axis title, default "module"
`yname`	y-axis title, default either "-log10 p-value"

Value

returns the plot, an object of class `ggplot2`.

References

Aghaieabiane, N and Koutis, I (2022) SGCP: A semi-supervised pipeline for gene clustering using self-training approach in gene 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 *Plots Adjacency Matrix HeatMap In SGCP Pipeline*

Description

Plots the HeatMap of Adjacency Matrix (Network) in SGCP pipeline.

Usage

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

Arguments

m	an adjacency matrix returned by adjacencyMatrix function in SGCP pipeline, or must be a matrix, symmetric, with values in (0, 1) and zero 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 (2022) SGCP: A semi-supervised pipeline for gene clustering using self-training approach in gene 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 *Jitter Plot Of Gene Ontology Enrichment p-values In SGCP Pipeline*

Description

Plots the jitter plot of gene ontology enrchment p-values in 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 geneOntology function in SGCP pipeline
tit	plot title, default "p-values Distribution"
xname	x-axis title, default "module"
yname	y-axis title, default either "-log10 p-value"
ps	a numeric for point size, default 3

Value

returns the plot, an object of class ggplot2.

References

Aghaieabiane, N and Koutis, I (2022) SGCP: A semi-supervised pipeline for gene clustering using self-training approach in gene 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 *Plots PCA Of The Data In SGCP Pipeline*

Description

Plots PCA with and without labels in SGCP pipeline.

Usage

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

Arguments

m	a numeric matrix of n*m
clusLabs	NULL or a vector of size n showing cluster labels, there 1-to-1 correspondance between the rows in m and clusLabs
tit	plot title, default "PCA plot"
ps	point size, default .5

Value

returns the plot, an object of class ggplot2.

References

Aghaieabiane, N and Koutis, I (2022) SGCP: A semi-supervised pipeline for gene clustering using self-training approach in gene 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 *Pie Chart of Gene Ontology Terms In SGCP Pipeline*

Description

Plots the test direction plus ontology of gene ontology terms in SGCP pipeline.

Usage

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

Arguments

df	the ‘GOresults‘ dataframe returned by geneOntology function in SGCP pipeline
tit	plot title, default "GO Analysis"
xname	x-axis title, default "module"
yname	y-axis title, default "count"
posx	a numeric for label position in pie chart, the higher the number the futhur the label will be from pie chart.

Value

returns the plot, an object of class ggplot2.

References

Aghaieabiane, N and Koutis, I (2022) SGCP: A semi-supervised pipeline for gene clustering using self-training approach in gene 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 *Plots Gene Silhouette Index In SGCP Pipeline*

Description

Plots the Silhouette index of genes in SGCP pipeline.

Usage

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

Arguments

df	the silhouette dataframe returned by clustering function in 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‘ parameter in clustering function must be set to TRUE.

Value

returns the plot, an object of class ggplot2.

References

Aghaieabiane, N and Koutis, I (2022) SGCP: A semi-supervised pipeline for gene clustering using self-training approach in gene 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

∗ **classification**
semiSupervised, 19
∗ **clustering**
clustering, 5
∗ **datasets**
cheng, 4
∗ **graphs**
adjacencyMatrix, 2

adjacencyMatrix, 2, *7*, *13*, *29*

cheng, 4
clustering, 5, *13*, *20*, *27*, *33*

ezSGCP, 7, *21*

geneOntology, 11, *14, 15*, *19*, *26*, *28*, *30*, *32*

resClus, 13
resFinalGO, 14
resInitialGO, 15
resSemiLabel, 16
resSemiSupervised, 17

semiLabeling, *16, 17*, 18, *20*
semiSupervised, 19
sgcp, 20
SGCP_ezPLOT, 22, *26–33*
SGCP_plot_bar, 26
SGCP_plot_conductance, 27
SGCP_plot_density, 28
SGCP_plot_heatMap, 29
SGCP_plot_jitter, 30
SGCP_plot_pca, 31
SGCP_plot_pie, 32
SGCP_plot_silhouette, 33