Package 'Pigengene'

April 18, 2025

Type Package

Title Infers biological signatures from gene expression data

Version 1.35.0

Date 2016-03-31

Author Habil Zare, Amir Foroushani, Rupesh Agrahari, Meghan Short, Isha Mehta, Neda Emami, and Sogand Sajedi

Maintainer Habil Zare <zare@u.washington.edu>

biocViews GeneExpression, RNASeq, NetworkInference, Network, GraphAndNetwork, BiomedicalInformatics, SystemsBiology, Transcriptomics, Classification, Clustering, DecisionTree, DimensionReduction, PrincipalComponent, Microarray, Normalization, ImmunoOncology

Depends R (>= 4.0.3), graph, BiocStyle (>= 2.28.0)

Description Pigengene package provides an efficient way to infer biological signatures from gene expression profiles. The signatures are independent from the underlying platform, e.g., the input can be microarray or RNA Seq data. It can even infer the signatures using data from one platform, and evaluate them on the other. Pigengene identifies the modules (clusters) of highly coexpressed genes using coexpression network analysis, summarizes the biological information of each module in an eigengene, learns a Bayesian network that models the probabilistic dependencies between modules, and builds a decision tree based on the expression of eigengenes.

License GPL (>=2)

Imports bnlearn (>= 4.7), C50 (>= 0.1.2), MASS, matrixStats, partykit, Rgraphviz, WGCNA, GO.db, impute, preprocessCore, grDevices, graphics, stats, utils, parallel, pheatmap (>= 1.0.8), dplyr, gdata, clusterProfiler, ReactomePA, ggplot2, openxlsx, DBI, DOSE

Suggests org.Hs.eg.db (>= 3.7.0), org.Mm.eg.db (>= 3.7.0), biomaRt (>= 2.30.0), knitr, AnnotationDbi, energy

VignetteBuilder knitr

NeedsCompilation no

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

git_branch devel

git_last_commit ab1017c

git_last_commit_date 2025-04-15

Repository Bioconductor 3.22

Date/Publication 2025-04-17

Contents

Pigengene-package *Infers robust biological signatures from gene expression data*

Description

Pigengene identifies gene modules (clusters), computes an eigengene for each module, and uses these biological signatures as features for classification. The resulting biological signatures are very robust with respect to the profiling platform. For instance, if Pigenegene computes a biological signature using a microarray dataset, it can infer the same signature in an RNA Seq dataset such that it is directly comparable across the two datasets.

Details

Package:	Pigengene
Type:	Package
Version:	0.99.0
Date:	2016-04-25
License:	GPL (>= 2)

The main function is `one.step.pigengene` which requires a gene expression profile and the corresponding conditions (types). Individual functions are provided to facilitate running the pipeline in a customized way. Also, the inferred biological signatures (computed eigengenes) are useful for other supervised or unsupervised analyses.

In most functions of this package, eigenegenes are computed or used as robust biological signatures. Briefly, each eigengene is a weighted average of the expression of all genes in a module (cluster), where the weights are adjusted in a way that the explained variance is maximized.

Author(s)

Amir Foroushani, Habil Zare, and Rupesh Agrahari

Maintainer: Habil Zare <zare@txstate.edu>

References

Foroushani, Amir, et al. "Large-scale gene network analysis reveals the significance of extracellular matrix pathway and homeobox genes in acute myeloid leukemia: an introduction to the Pigengene package and its applications." BMC medical genomics 10.1 (2017): 1-15.

See Also

`Pigengene-package`, `one.step.pigengene`, `compute.pigengene`, `project.eigen`, `WGCNA::blockwiseModules`

Examples

```
data(aml)
data(mds)
d1 <- rbind(aml,mds)
Labels <- c(rep("AML",nrow(aml)),rep("MDS",nrow(mds)))
names(Labels) <- rownames(d1)
```

```
p1 <- one.step.pigengene(Data=d1,saveDir='pigengene', bnNum=10, verbose=1,
   seed=1, Labels=Labels, toCompact=FALSE, doHeat=FALSE)
plot(p1$c5treeRes$c5Trees[["34"]])
## See pigengene for results.
```

---

aml *AML gene expression profile*

Description

Gene expression profile of 202 acute myeloid leukemia (AML) cases from Mills et al. study. The profile was compared with the profile of 164 myelodysplastic syndromes (MDS) cases and only the 1000 most differentially expressed genes are included.

Usage

```
data("aml")
```

Format

A numeric matrix

Details

The columns and rows are named according to the genes Entrez, and patient IDs, respectively. The original data was produced using Affymetrix Human Genome U133 Plus 2.0 Miccoaray. Mills et al. study is part of the MILE Study (Microarray Innovations In LEukemia) program, and aimed at prediction of AML transformation in MDS.

Value

It is a `202*1000` numeric matrix.

Source

http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE15061

References

Mills, Ken I., et al. (2009). Microarray-based classifiers and prognosis models identify subgroups with distinct clinical outcomes and high risk of AML transformation of myelodysplastic syndrome. Blood 114.5: 1063-1072.

See Also

Pigengene-package, one.step.pigengene, mds, pigengene

Examples

```
library(pheatmap)
data(aml)
pheatmap(aml[,1:20],show_rownames=FALSE)
```

apply.filter *Applies a given filter on the data*

Description

Takes as input gamma and epsilon values and a filter graph, which is represented by an adjacency matrix named filt. Applies the filter on the data in either of the two ways: a) with normalization of the filter by degrees in the graph, b) without normalization.

Usage

```
apply.filter(gamma, filt, Data, doNormalize=FALSE)
```

Arguments

gamma	This value is in the [0,1] range and determines the weight of the filter data. Setting to 0 will result in not filtering at all.
filt	It is a binary matrix computed by the make.filter function.
Data	A matrix or data frame (or list of matrices or data frames) containing the expression data, with genes corresponding to columns and rows corresponding to samples. Rows and columns must be named. For example, for RNA-Seq data, log(RPKM+1) can be used.
doNormalize	If TRUE, the filter will be normalized by the degree in the graph using the filt * D^(-1), where D is a diagonal matrix with degrees of filt on its diagonal.

Value

filtered	A filtered matrix computed using the gamma*sData formula, where sData is the scaled Data and filtN is the normalized or unormalized filter.

Author(s)

Habil Zare and Neda Emami.

See Also

make.filter, determine.modules

Examples

```
data(aml)
data(mds)
d1 <- rbind(aml,mds)[, 1:200]
Labels <- c(rep("AML",nrow(aml)),rep("MDS",nrow(mds)))
names(Labels) <- rownames(d1)

p0 <- one.step.pigengene(Data=d1, saveDir=".", verbose=1,
                         seed=1, Labels=Labels, naTolerance=0.5,
    RsquaredCut=0.8, doNetOnly=TRUE)

##Making the filter
made <- make.filter(network=p0$Network, epsilon=0.7, outPath=".")
```

```
##Applying the filter
f1 <- apply.filter(gamma=0.5, filt=made$filt, Data=d1)
```

`balance` *Balances the number of samples*

Description

Oversamples data by repeating rows such that each condition has roughly the same number of samples.

Usage

```
balance(Data, Labels, amplification = 5, verbose = 0, naTolerance=0.05)
```

Arguments

`Data`	A matrix or data frame containing the expression data, with genes corresponding to columns and rows corresponding to samples. Rows and columns must be named.
`Labels`	A (preferably named) vector containing the Labels (condition types) for `Data`. Names must agree with rows of `Data`.
`amplification`	An integer that controls the number of repeats for each condition. The number of all samples roughly will be multiplied by this factor after oversampling.
`verbose`	The integer level of verbosity. 0 means silent and higher values produce more details of computation.
`naTolerance`	Upper threshold on the fraction of entries per gene that can be missing. Genes with a larger fraction of missing entries are ignored. For genes with smaller fraction of NA entries, the missing values are imputed from their average expression in the other samples. See `check.pigengene.input`.

Value

A list of:

`balanced`	The matrix of oversampled data
`Reptimes`	A vector of integers named by conditions reporting the number of repeats for each condition.
`origSampleInds`	The indices of rows in `balanced` that correspond to the original samples before oversampling

Author(s)

Habil Zare

See Also

`Pigengene-package`, `one.step.pigengene`, `wgcna.one.step`, `compute.pigengene`

Examples

```
data(aml)
data(mds)
d1 <- rbind(aml,mds)
Labels <- c(rep("AML",nrow(aml)),rep("MDS",nrow(mds)))
names(Labels) <- rownames(d1)
b1 <- balance(Data=d1, Labels=Labels)
d2 <- b1$balanced
```

`calculate.beta` *Estimates an appropriate power value*

Description

The WGCNA package assumes that in the coexpression network the genes are connected with a power-law distribution. Therefore, it need a soft-thresholding power for network construction, which is estimated by this auxiliary function.

Usage

```
calculate.beta(saveFile = NULL, RsquaredCut = 0.8, Data,  doThreads=FALSE,
  verbose = 0)
```

Arguments

`saveFile`	The file to save the results in. Set to `NULL` to disable.
`RsquaredCut`	A threshold in the range [0,1] used to estimate the power. A higher value can increase power. For technical use only. See `pickSoftThreshold` for more details.
`Data`	A matrix or data frame containing the expression data, with genes corresponding to columns and rows corresponding to samples. Rows and columns must be named.
`doThreads`	Boolean. Allows WGCNA to run a little faster using multi-threading but might not work on all systems.
`verbose`	The integer level of verbosity. 0 means silent and higher values produce more details of computation.

Value

A list of:

`sft`	The full output of `pickSoftThreshold` function
`power`	The estimated power (beta) value
`powers`	The numeric vector of all tried powers
`RsquaredCut`	The value of input argument `RsquaredCut`

References

Langfelder P and Horvath S, WGCNA: an R package for weighted correlation network analysis. BMC Bioinformatics 2008, 9:559

See Also

pickSoftThreshold, blockwiseModules, one.step.pigengene, wgcna.one.step

Examples

```
data(aml)
p1 <- calculate.beta(Data=aml[,1:200])
```

---

check.nas *Removes* NA*s from a data matrix*

Description

Checks Data for NA values.

Usage

```
check.nas(Data, naTolerance=0.05, na.rm=TRUE)
```

Arguments

Data	A matrix or data frame containing the expression data, with genes corresponding to columns and rows corresponding to samples. Rows and columns must be named.
naTolerance	A number in the 0-1 range. If the frequency of NAs in a column of Data is more than this threshold , then that column will be removed.
na.rm	If TRUE, NAs in the Data will be replaces with the average of the column, however, if the frequency of NAs in the column is too high (i.e., more than naTolerance), the whole column will be removed.

Value

A list of:

cleaned	The cleaned data with no NA value. Rows are the same as Data, but some columns may be deleted.
tooNaGenes	A character vector of those genes (i.e., column names of Data) that had too many NAs, and therefore were removed.
replacedNaNum	The number of NA entries in the matrix that were replaced with the average of the corresponding column (gene).

Author(s)

Habil Zare

See Also

check.pigengene.input, Pigengene-package

Examples

```
data(aml)
dim(aml)
aml[1:410]<-NA
c1 <- check.nas(Data=aml)
dim(c1$cleaned)
c1$tooNaGenes
rm(aml)
```

`check.pigengene.input` *Quality check on the imput*

Description

Checks `Data` and `Labels` for `NA` values, row and column names, etc.

Usage

```
check.pigengene.input(Data, Labels, na.rm = FALSE, naTolerance=0.05)
```

Arguments

`Data`	A matrix or data frame containing the expression data, with genes corresponding to columns and rows corresponding to samples. Rows and columns must be named.
`Labels`	A (preferably named) vector containing the Labels (condition types) for `Data`. Names must agree with rows of `Data`.
`na.rm`	If `TRUE`, `NA`s in the Data will be replaces with the average of the column, however, if the frequency of `NA`s in the column is too high, the whole column will be removed.
`naTolerance`	Upper threshold on the fraction of entries per gene that can be missing. Genes with a larger fraction of missing entries are ignored. For genes with smaller fraction of NA entries, the missing values are imputed from their average expression in the other samples. See `check.pigengene.input`.

Value

A list of:

`Data`	The checked `Data` matrix, `NA` possibly removed and rows are ordered as names of `Labels`.
`Labels`	The checked vector of `Labels`

Author(s)

Habil Zare

See Also

`check.nas`, `one.step.pigengene`, `Pigengene-package`

Examples

```
data(aml)
Labels <- c(rep("AML",nrow(aml)))
names(Labels) <- rownames(aml)
c1 <- check.pigengene.input(Data=aml, Labels=Labels,na.rm=TRUE)
Data <- c1$Data
Labels <- c1$Labels
```

`combine.networks` *Combines two or more networks*

Description

Takes as input two or more adjacency matrices, and the corresponding contributions. Computes a combined network (weighted graph) in which the weight on an edge between two nodes is an average of the weights on the same edge in the input networks.

Usage

```
combine.networks(nets, contributions, outPath, midfix="",
    powerVector=1:20, verbose=1, RsquaredCut=0.75, minModuleSize=5,
   doRemoveTOM=TRUE, datExpr, doReturNetworks=FALSE, doSave=FALSE, doIdentifyModule=TRUE)
```

Arguments

`nets`	A list of adjacency matrices (networks), which can be generated using e.g., the `WGCNA::adjacency` function. Rows and columns must be named.
`contributions`	A numeric vector with the same length as nets. In computing the average weight on each edge in the combined network, first the edge weights from individual networks are multiplied by their corresponding contributions, then the result will be divided by the sum of weights of all networks containing this edge.
`outPath`	A string to the path where plots and results will be saved.
`midfix`	An optional string used in the output file names.
`powerVector`	A numeric vector of power values that are tried to find the best one. See `WGCNA::pickSoftThreshold` documentation.
`verbose`	The integer level of verbosity. 0 means silent and higher values produce more details of computation.
`RsquaredCut`	A threshold in the range [0,1] used to estimate the power. A higher value can increase power. For technical use only. See `pickSoftThreshold` for more details.
`minModuleSize`	The value that controls the minimum number of genes per module. See `WGCNA::blockwiseModules`.
`doRemoveTOM`	A boolean determining the big TOM file must remove or not.
`datExpr`	The expression matrix that `WGCNA::blockwiseModules` uses for fine-tuning and removing genes from modules. This is not an ideal behavior by WGCNA.
`doReturNetworks`	A boolean value to determine whether to return `Network`, which is relatively a big matrix (typically GBs). Set to `FALSE` not to waste memory.

doSave — A boolean value to determine whether the whole output of this function (typically 1-2 GBs) should be saved as combinedNetwork. Set to FALSE not to waste disk space.

doIdentifyModule — A boolean value to determine whether modules should be identified. Set it to FALSE if you just need the network, not the modules.

Value

A list with following components

call — The command that created the results

midfix — The input argument

Network — The adjacency matrix of the combined network

denominators — A matrix, each cell of which is the sum of weights of all networks contributing to the edge corresponding to that cell

power — The power (beta) value used for the combined network

fits — The fit indices calculated for the combined network

net — The output of WGCNA::blockwiseModules containing the module information in its colors field

modules — The output of WGCNA::blockwiseModules

combinedNetworkFile — The path to the saved file containing combinedNetwork

Note

If the networks have different node sets, the combined network will be computed on the union of nodes.

See Also

WGCNA::blockwiseModules, WGCNA::TOMsimilarity, and WGCNA::pickSoftThreshold.fromSimilarity

Examples

```
data(aml)
data(mds)
nets <- list()
## Make the coexpression networks:
nets[["aml"]] <- abs(stats::cor(aml[,1:200]))
nets[["mds"]] <- abs(stats::cor(mds[,1:200]))
## Combine them:
combined <- combine.networks(nets=nets, contributions=c(nrow(aml), nrow(mds)),
                outPath=".", datExpr=rbind(aml, mds)[,1:200])
print(table(combined$modules))
```

compact.tree *Reduces the number of genes in a decision tree*

Description

In a greedy way, this function removes the genes with smaller weight one-by-one, while assessing the accuracy of the predictions of the resulting trees.

Usage

```
compact.tree(c5Tree, pigengene, Data=pigengene$Data, Labels=pigengene$Labels,
  testD=NULL, testL=NULL, saveDir=".", verbose=0)
```

Arguments

c5Tree	A decision tree of class C50 that uses module eigengenes, or NULL. If NULL, If NULL, expression plots for all modules are created.
pigengene	A object of pigengene-class, output of compute.pigengene
Data	A matrix or data frame containing the expression data, with genes corresponding to columns and rows corresponding to samples. Rows and columns must be named.
Labels	Labels (condition types) for the (training) expression data. It is a named vector of characters. Data will be subset according to these names.
testD	The test expression data, for example, from an independent dataset. Optional.
testL	Labels (condition types) for the (test) expression data. Optional.
saveDir	Where to save the plots of the tree(s)
verbose	Integer level of verbosity. 0 means silent and higher values produce more details of computation.

Value

A list with following elements is invisibly returned:

call	The call that created the results
predTrain	Prediction using projected data without compacting
predTrainCompact	Prediction after compacting
genes	A character vector of all genes in the full tree before compacting
genesCompacted	A character vector of all genes in the compacted tree
trainErrors	A matrix reporting errors on the train data. The rows are named according to the number of removed genes. Each column reports the number of misclassified samples in one condition (type) except the last column that reports the total.
testErrors	A matrix reporting errors on the test data similar to trainErrors
queue	A numeric vector named by all genes contributing to the full tree before compacting. The numeric values are weights increasingly ordered by absolute value.
pos	The number of removed genes
txtFile	Confusion matrices and other details on compacting are reported in this text file

References

Large-scale gene network analysis reveals the significance of extracellular matrix pathway and homeobox genes in acute myeloid leukemia, Foroushani A, Agrahari R, Docking R, Karsan A, and Zare H. In preparation.

Gene shaving as a method for identifying distinct sets of genes with similar expression patterns, Hastie, Trevor, et al. Genome Biol 1.2 (2000): 1-0003.

See Also

Pigengene-package, compute.pigengene, make.decision.tree, C5.0, Pigengene-package

Examples

```
## Data:
data(aml)
data(mds)
data(pigengene)
d1 <- rbind(aml,mds)

## Fiting the trees:
trees <- make.decision.tree(pigengene=pigengene, Data=d1,
saveDir="trees", minPerLeaf=14:15, doHeat=FALSE,verbose=3,
toCompact=FALSE)
c1 <- compact.tree(c5Tree=trees$c5Trees[["15"]], pigengene=pigengene,
    saveDir="compacted", verbose=1)
```

---

compute.pigengene *Computes the eigengenes*

Description

This function takes as input the expression data and module assignments, and computes an eigengene for each module using PCA. If you already have a Pigengene object, you can use the `project.eigen` function to infer the values of your eigengenes in a new expression dataset.

Usage

```
compute.pigengene(Data, Labels, modules, saveFile = "pigengene.RData",
    selectedModules = "All", amplification = 5, doPlot = TRUE,
   verbose = 0, dOrderByW = TRUE, naTolerance=0.05, doWgcna=FALSE, doMinimize=FALSE)
```

Arguments

`Data`	A matrix or data frame containing the training expression data, with genes corresponding to columns and rows corresponding to samples. Rows and columns must be named.
`Labels`	A (preferably named) vector containing the Labels (condition types) for the training Data. Names must agree with rows of `Data`.
`modules`	A numeric vector, named by genes, that reports the module (clustering) assignments.

saveFile	The file to save the results. NULL will disable saving, and thus requires doPlot to be FALSE.
selectedModules	A numeric vector determining which modules to use, or set to "All" (default) to include every module.
amplification	An integer that controls the number of repeats for each condition. The number of all samples roughly will be multiplied by this factor after oversampling. See balance.
doPlot	Boolean determining whether heatmaps of expression of eigengenes should be ploted and saved. Set it to FALSE for large data to avoid memory exhaustion.
verbose	The integer level of verbosity. 0 means silent and higher values produce more details of computation.
dOrderByW	If TRUE, the genes will be ordered in the csv file based on their absolute weight in the corresponding module.
naTolerance	Upper threshold on the fraction of entries per gene that can be missing. Genes with a larger fraction of missing entries are ignored. For genes with smaller fraction of NA entries, the missing values are imputed from their average expression in the other samples. See check.pigengene.input.
doWgcna	If FALSE, prcomp will be used to compute PCA. Otherwise, WGCNA::blockwiseModules will be used leading to consuming more memory with no advantages.
doMinimize	If TRUE, only the minimal elements essential for the project.eigen function will be included, leading to an order of magnitude smaller pigengene object.

Details

Rows of Data are oversampled using balance so that each condition has roughly the same number of samples. For each module, an eigengene is computed using PCA.

Value

An object of pigengene-class.

Author(s)

Habil Zare and Amir Foroushani

References

Large-scale gene network analysis reveals the significance of extracellular matrix pathway and homeobox genes in acute myeloid leukemia, Foroushani A, Agrahari R, Docking R, Karsan A, and Zare H. In preparation.

See Also

Pigengene-package, one.step.pigengene, wgcna.one.step, project.eigen, make.decision.tree, moduleEigengenes

Examples

```
## Data:
data(aml)
data(mds)
data(eigengenes33)
d1 <- rbind(aml,mds)
Labels <- c(rep("AML",nrow(aml)),rep("MDS",nrow(mds)))
names(Labels) <- rownames(d1)
modules33 <- eigengenes33$modules[colnames(d1)]
## Computing:
pigengene <- compute.pigengene(Data=d1, Labels=Labels, modules=modules33,
   saveFile="pigengene.RData", doPlot=TRUE, verbose=3)
class(pigengene)
plot(pigengene, fontsize=12)


## If you need the pigengene object only to compute eigengenes
## in a new   dataset, you can make is much smaller.
pigengeneM <- compute.pigengene(Data=d1, Labels=Labels, modules=modules33,
   saveFile="pigengene.RData", doPlot=TRUE, verbose=1, doMinimize=TRUE)
object.size(pigengene)/10^6 ## MB
object.size(pigengeneM)/10^6 ## MB
```

`dcor.matrix` *Computes distance correlation for give matrix*

Description

This function computes the distance correlation between every pair of columns of the input data matrix.

Usage

```
dcor.matrix(Data)
```

Arguments

`Data`	A matrix containing the data

Details

Using for loops, all pairs of columns are passed to `link[energy]{dcor}` function from `link[energy]{energy-package`

Value

A numeric square matrix. The number of rows and columns is equal to the number of columns of `Data` and they are named accordingly.

Note

This function uses for loops, which are not efficient for an input matrix with too many columns.

Author(s)

Habil Zare

References

Szekely, G.J., Rizzo, M.L., and Bakirov, N.K. (2007), Measuring and Testing Dependence by Correlation of Distances, _Annals of Statistics_, Vol. 35 No. 6, pp. 2769-2794.

<URL: http://dx.doi.org/10.1214/009053607000000505>

Szekely, G.J. and Rizzo, M.L. (2009), Brownian Distance Covariance, _Annals of Applied Statistics_, Vol. 3, No. 4, 1236-1265.

<URL: http://dx.doi.org/10.1214/09-AOAS312>

Szekely, G.J. and Rizzo, M.L. (2009), Rejoinder: Brownian Distance Covariance, _Annals of Applied Statistics_, Vol. 3, No. 4, 1303-1308.

See Also

`link[energy]{dcor}`

Examples

```
## Data:
data(aml)
dcor1 <- dcor.matrix(Data=aml[,1:5])
dcor1

## Comparison with Pearson:
cor1 <- abs(stats::cor(aml[,1:5]))
## With 202 samples, distance and Pearson correlations do not differ much:
dcor1-cor1
dcor2 <- dcor.matrix(Data=aml[1:20,1:5])
cor2 <- abs(stats::cor(aml[1:20,1:5]))
## Distance correlation is more robust if fewer samples are available:
dcor2-cor2
plot(dcor2-cor1,cor1-cor2,xlim=c(-0.5,0.5),ylim=c(-0.5,0.5))
```

`determine.modules` *Identifies modules of the network*

Description

Takes as input a `network` (i.e., weighted graph) and identifies modules (i.e., clusters of similar genes) using `WGCNA::blockwiseModules`. It also produces a plot showing the number of genes in each module.

Usage

```
determine.modules(network, outPath, midfix="", powerVector=1:20,
  verbose=1, RsquaredCut=0.75, minModuleSize=5, doRemoveTOM=FALSE,
               datExpr, doSave=FALSE)
```

Arguments

network	An adjacency matrix of the network that is built using combine.networks.
outPath	A string to the path where plots and results will be saved.
midfix	An optional string used in the output file names.
powerVector	A numeric vector of integer values that are tried to find the best power. See WGCNA::pickSoftThreshold.
verbose	The integer level of verbosity, where 0 means silent and higher values produce more details.
RsquaredCut	A threshold in the range [0,1] used to estimate the power. A higher value can increase power. For technical use only. See pickSoftThreshold for more details.
minModuleSize	The value that controls the minimum number of genes per module. See WGCNA::blockwiseModules.
doRemoveTOM	A boolean determining whether the big TOM file must remove or not.
datExpr	The expression matrix that WGCNA::blockwiseModules uses for fine-tuning and removing genes from modules. This is not an ideal behavior by WGCNA.
doSave	A boolean value to determine whether the whole output of this function (typically 1-2 GBs) should be saved as combinedNetwork. Set to FALSE not to waste disk space.

Value

A list with the following components:

call	The call that created the results.
midfix	The midfix input.
power	The integer value of the estimated power computed by pickSoftThreshold.fromSimilarity.
fits	The fitIndices output from pickSoftThreshold.fromSimilarity.
modules	A vector that representing the identified modules. Its length is equal to the number of nodes in the network, named by node names (i.e., row names of network), and values are the corresponding module numbers.
net	The full output of the blockwiseModules function.

Author(s)

Neda Emami and Habil Zare.

See Also

apply.filter, combine.networks, make.filter

Examples

```
data(aml)

##Making the coexpression network
network <- abs(stats::cor(aml[,1:200]))

##Identifying modules
identifiedMod <- determine.modules(network=network, outPath=".", datExpr=aml[,1:200])
print(table(identifiedMod$modules))
```

draw.bn *Draws a Bayesian network*

Description

Draws the BN using appropriate colors and font size.

Usage

```
draw.bn(BN, plotFile = NULL, inputType = "ENTREZIDat", edgeColor = "blue",
  DiseaseCol = "darkgreen", DiseaseFill = "red", DiseaseChildFill = "pink",
  nodeCol = "darkgreen", nodeFill = "yellow", moduleNamesFile = NULL,
  mainText = NULL, nodeFontSize = 14 * 1.1, verbose = 0)
```

Arguments

BN	An object of bn-class
plotFile	If not NULL, the plot will be saved here.
inputType	The type of gene IDs in BN
edgeColor	The color of edges
DiseaseCol	The color of the border of the Disease node
DiseaseFill	The color of the area inside the Disease node
DiseaseChildFill	The color of the area inside the children of the Disease node
nodeCol	The color of the border of the usual nodes excluding Disease and its children
nodeFill	The color of the area inside the usual nodes
moduleNamesFile	An optional csv file including the information to rename the nodes name. See coderename.node.
mainText	The main text shown at the top of the plot
nodeFontSize	Adjusts the size of nodes
verbose	The integer level of verbosity. 0 means silent and higher values produce more details of computation.

Value

A list with following components:

call	The call that created the results
BN	An echo of input BN argument
renamedBN	An object of bn-class when moduleNamesFile is provided
gr	The full output of graphviz.plot function

Author(s)

Habil Zare

See Also

bnlearn-package, Pigengene-package, learn.bn, graph-class

Examples

```
## See lear.bn function.
```

---

eigengenes33 *Eigengenes of 33 modules*

Description

This list contains partial eigengenes computed from AML and MDS gene expression profiles provided by Mills et al. These data are included to illustrate how to use Pigengene-package and also to facilitate reproducing the results presented in the corresponding paper.

Usage

```
data(eigengenes33)
```

Format

A list

Details

The top 9166 differentially expressed genes were identified and their expressions in AML were used for identifying 33 modules. The first column, ME0, corresponds to module 0 (outliers) and is usually ignored. The eigengene for each module was obtained using compute.pigengene function. Oversampling was performed with `amplification=5` to adjust for unbalanced sample-size.

Value

It is a list of 3 objects:

`aml` A 202 by 34 matrix. Each column reports the values of a module eigengene for AML cases.

`mds` A 164 by 34 matrix for MDS cases with columns similar to aml.

`modules` A numeric vector of length 9166 labeling members of each module. Named by Entrez ID.

Source

http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE15061

References

Mills, Ken I., et al. (2009). Microarray-based classifiers and prognosis models identify subgroups with distinct clinical outcomes and high risk of AML transformation of myelodysplastic syndrome. Blood 114.5: 1063-1072.

See Also

Pigengene-package, compute.pigengene, aml, mds, learn.bn

Examples

```
library(pheatmap)
data(eigengenes33)
pheatmap(eigengenes33$aml,show_rownames=FALSE)
## See Pigengene::learn.bn() documentation for more examples.
```

---

gene.mapping *Maps gene IDs*

Description

Takse as input gene IDs in a convention, say REFSEQ, and converts them to another convention.

Usage

```
gene.mapping(ids, inputType = "REFSEQ", outputType = "SYMBOL",
  leaveNA = FALSE, inputDb = "Human", outputDb = inputDb,
  verbose = 0)
```

Arguments

ids	A character vector of input gene IDs
inputType	The type of input IDs.
outputType	The type of output IDs. If it is a character vector, mapping will be done for each element.
leaveNA	If TRUE,the IDs that were not matched are left with NAs in the second column of the output, otherwise (i.e., default) the input IDs are returned.
inputDb	The input data base. Use org.Hs.eg.db for human and org.Mm.eg.db for mouse. The default "Human" character uses the former.
outputDb	The output data base. If it is a list, mapping will be done for each element.
verbose	The integer level of verbosity. 0 means silent and higher values produce more details of computation.

Details

It can map homologous genes between species e.g. from mouse to human. If more than 1 ID found for an input gene, only one of them is returned.

Value

A matrix of characters with 3 columns: input, output1, and output2. The last one is guaranteed not to be NA if leaveNA=FALSE.

Author(s)

Amir Foroushani, Habil Zare, and Rupesh Agrahari

References

Pages H, Carlson M, Falcon S and Li N. AnnotationDbi: Annotation Database Interface. R package version 1.32.3.

See Also

AnnotationDb-class, org.Hs.eg.db org.Mm.eg.db

Examples

```
library(org.Hs.eg.db)
g1 <- gene.mapping(ids="NM_001159995")
print(g1)

## Mapping to multiple convention
library(org.Mm.eg.db)
g2 <- gene.mapping(ids=c("NM_170730", "NM_001013580"),
   inputType="REFSEQ", inputDb=org.Mm.eg.db,
   outputType=c("SYMBOL","ENTREZID"),
   outputDb=list(org.Hs.eg.db,org.Mm.eg.db), verbose=1)
print(g2)
```

---

get.enriched.pw *Performs pathway over representation analysis*

Description

Takes as input a vector or list of gene IDs in any convention, and performs over representation analysis.

Usage

```
get.enriched.pw(genes, idType, pathwayDb, ont=c("BP", "MF", "CC"),
                Org="Human", OrgDb=NULL, outPath, pvalueCutoff=0.05,
                pAdjustMethod="BH", fontSize=14, verbose=0)
```

Arguments

genes	A character vector or a named list of genes for which pathway over representation analysis to be done.
idType	A string describing the type of input gene ID e.g., "ENTREZID", "REFSEQ", "SYMBOL".
pathwayDb	A character vector determining which enrichment database to be used e.g., "GO", "KEGG", "REACTOME", or "NCG".
ont	GO ontology terms to be analysed e.g., "BP", "MF" or "CC". Default is all three.
Org	A character string equal to "Human" or "Mouse" determining the reference organism to be used. For "Human" and "Mouse" org.Hg.eg.db and org.Mm.eg.db will be used, respectively. If Org is not NULL, OrgDb must be NULL.
OrgDb	The reference data base to be used. Use e.g. org.Ce.eg.db for 'Celegans' when analysing Celegans data. If OrgDb is not NULL, Org must be NULL.
outPath	A file path where results will be saved.
pvalueCutoff	A numerical value that determines a cutoff of adjusted pValue.
pAdjustMethod	A string passed to the clusterProfiler::enrichGO function to determine the method for adjusting the p-value. Options include "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none".

fontSize	A numerical value that determines the font size of the y-axis and the title in the plot.
verbose	The integer level of verbosity. 0 means silent and higher values produce more details of computation.

Value

A list:

enriched	A list of output of enrichment analysis for different database analyzed.
noEnrichment	A vector of database names in which no enriched pathways were found.

The output is saved for each selected module under the moduleName_enrichment folder. There is a subfolder that includes an excel file and plot(s). Each sheet in the excel file corresponds to a pathway database (KEGG in the below example). Each row is an overrepresented pathway.

Author(s)

Isha Mehta, Habil Zare, and Sogand Sajedi

References

Guangchuang Yu, Li-Gen Wang, Yanyan Han and Qing-Yu He, clusterProfiler: an R package for comparing biological themes among gene clusters. OMICS: A Journal of Integrative Biology 2012, 16(5):284-287

Guangchuang Yu, Qing-Yu He. ReactomePA: an R/Bioconductor package for reactome pathway analysis and visualization. Molecular BioSystems 2016, 12(2):477-479

See Also

enrichGO, enrichKEGG, enrichNCG, enrichPathway

Examples

```
library(org.Hs.eg.db)
genes <- c("NM_170730", "NM_001013580", "NM_002142", "NM_003417", "NM_000082",
           "NM_006158", "NM_006047", "NM_022356", "NM_003979", "NM_001030", "NM_022872")
p1 <- get.enriched.pw(genes=genes, idType="REFSEQ", pathwayDb="GO", Org="Human",
                      outPath=getwd(), verbose=1)
```

---

get.fitted.leaf *Returs the leaf for each sample*

Description

Taking as input a tree and data, this function determines the leaf each sample will fall in.

Usage

```
get.fitted.leaf(c5Tree, inpDTemp, epsi = 10^(-7))
```

Arguments

c5Tree	A decision tree of class C50 that uses module eigengenes, or NULL. If NULL, expression plots for all modules are created.
inpDTemp	The possibly new data matrix with samples on rows
epsi	A small perturbation to resolve the boundary issue

Value

A numeric vector of node indices named by samples (rows of inpDTemp)

Note

This function is tricky because C50 uses a global variable.

Author(s)

Amir Foroushani

See Also

Pigengene-package, make.decision.tree, compact.tree, compute.pigengene, module.heatmap, get.used.features, preds.at

Examples

```
## Data:
data(aml)
data(mds)
data(pigengene)
d1 <- rbind(aml,mds)

## Fiting the trees:
trees <- make.decision.tree(pigengene=pigengene, Data=d1,
saveDir="trees", minPerLeaf=15, doHeat=FALSE,verbose=3,
  toCompact=FALSE)
f1 <- get.fitted.leaf(c5Tree=trees$c5Trees[["15"]],
  inpDTemp=pigengene$eigengenes)
```

---

get.genes *List the (most relevant) genes for a decision tree.*

Description

This function returns all genes that are left after shrinking (compacting) a given tree. If enhance is set to TRUE, it makes sure that the output contains at least two genes from each used module.

Usage

```
get.genes(c5Tree = NULL, pigengene = NULL, queue = NULL, modules = NULL, pos=0,
  enhance = TRUE)
```

Arguments

queue	A character vector. The membership queue for a decsision tree.
pos	Number of genes that are considered from removal. Same interpretation as in preds.at
enhance	If enhance is set to TRUE, the function makes sure that the output contains at least two genes from each used module. Otherwise, exactly the pos first elements of the queue are removed from consideration.
modules	Named character vector listing the module assignments.
c5Tree	A decision tree of class C50.
pigengene	An object in pigengene-class, usually created by compute.pigengene.

Details

This function needs modules and queue, or alternatively, c5Tree and pigengene.

Value

A character vector containing the names of the genes involved in the modules whose eigengenes are used in the tree. If pos > 0, the first pos such genes with lowest absolute membership in their respective modules are filtered.

See Also

Pigengene-package, compact.tree,preds.at, get.used.features, make.decision.tree

Examples

```
## Data:
data(aml)
data(mds)
data(pigengene)
d1 <- rbind(aml,mds)

## Fiting the trees:
trees <- make.decision.tree(pigengene=pigengene, Data=d1,
saveDir="trees", minPerLeaf=15, doHeat=FALSE,verbose=3,
  toCompact=FALSE)
g1 <- get.genes(c5Tree=trees$c5Trees[["15"]],pigengene=pigengene)
```

---

get.used.features *Return the features used in a tree*

Description

Only some of the features will be automatically selected and used in a decision tree. However, an object of class C5.0 does not have the selected feature names explicitly. This function parses the tree component and extracts the names of features contributing to the tree.

Usage

```
get.used.features(c5Tree)
```

Arguments

c5Tree	A decision tree of class 50

Value

A character vector of the names of features (module eigengenes) contributing to the input decision tree.

Author(s)

Amir Foroushani

See Also

Pigengene-package, make.decision.tree, compact.tree, compute.pigengene, module.heatmap, get.fitted.leaf, preds.at, Pigengene-package

Examples

```
## Data:
data(aml)
data(mds)
data(pigengene)
d1 <- rbind(aml,mds)

## Fiting the trees:
trees <- make.decision.tree(pigengene=pigengene, Data=d1,
  saveDir="trees", minPerLeaf=15, doHeat=FALSE,verbose=3,
  toCompact=FALSE)
get.used.features(c5Tree=trees$c5Trees[["15"]])
```

---

learn.bn *Learns a Bayesian network*

Description

This function takes as input the eigengenes of all modules and learns a Bayesian network using bnlearn package. It builds several individual networks from random staring networks by optimizing their score. Then, it infers a consensus network from the ones with relatively "higher" scores. The default hyper-parameters and arguments should be fine for most applications.

Usage

```
learn.bn(pigengene=NULL, Data=NULL, Labels=NULL, bnPath = "bn", bnNum = 100,
  consensusRatio = 1/3, consensusThresh = "Auto", doME0 = FALSE,
  selectedFeatures = NULL, trainingCases = "All", algo = "hc", scoring = "bde",
  restart = 0, pertFrac = 0.1, doShuffle = TRUE, use.Hartemink = TRUE,
  bnStartFile = "None", use.Disease = TRUE, use.Effect = FALSE, dummies = NULL,
  tasks = "All", onCluster = !(which.cluster()$cluster == "local"),
  inds = 1:ceiling(bnNum/perJob), perJob = 2, maxSeconds = 5 * 60,
  timeJob = "00:10:00", bnCalculationJob = NULL, seed = NULL, verbose = 0,
  naTolerance=0.05)
```

Arguments

pigengene	An object from pigengene-class. The output of compute.pigengene function.
Data	A matrix or data frame containing the training data with eigengenes corresponding to columns and rows corresponding to samples. Rows and columns must be named.
Labels	A (preferably named) vector containing the Labels (condition types) for the training data. Names must agree with rows of Data.
bnPath	The path to save the results
bnNum	The total number of individual networks. In practice, the number of learnt networks can be less than bnNum because some jobs may take too long and be terminated.
consensusRatio	A numeric in the range 0-1 that determines what portion of highly scored networks should be used to build the consensus network
consensusThresh	A vector of thresholds in the range 0-1. For each threshold t, a consensus network will be build by considering the arcs that are present in at least a fraction of t of the individual networks. Alternatively, if it is "Auto" (the default), the threshold will be automatically set to the mean plus the standard deviation of the frequencies (strengths) of all arcs in the individual networks.
doME0	If TRUE, module 0 (the outliers) will be considered in learning the Bayesian network.
selectedFeatures	A character vector. If not NULL, only these features (eigengenes) will be used.
trainingCases	A character vector that determines which cases (samples) should be considered for learning the network.
algo	The algorithm that bnlean uses for optimizing the score. The default is "hc" (hill climbing). See arc.strength for other options and more details.
scoring	A character determining the scoring criteria. Use 'bde' and 'bic' for the Bayesian Dirichlet equivalent and Bayesian Information Criterion scores, respectively. See score for technical details.
restart	The number of random restarts. For technical use only. See hc.
pertFrac	A numeric in the range 0-1 that determines the number of attempts to randomly insert/remove/reverse an arc on every random restart. For technical use only.
doShuffle	The ordering of the features (eigengenes) is important in making the initial network. If doShuffle=TRUE, they will be shuffled before making every initial network.
use.Hartemink	If TRUE, Hartemink algorithm will be used to discretize data. Otherwise, interval discretization will be applied. See bnlearn:discretize.
bnStartFile	Optionally, learning can start from a Bayesian network instead of a random network. bnStartFile should contain a list called selected and selected$BN should be an object of bn-class. Non-technical users can set to "None" to disable.
use.Disease	If TRUE, the condition variable Disease will be included in the network, which cannot be the child of any other variable.
use.Effect	If TRUE, the condition variable beAML will be included in the network, which cannot be the parent of any other variable.

dummies — A vector of numeric values in the range 0-1. Dummy random variables will be added to the Bayesian network to check whether the learning process is effective. For development purposes only.

tasks — A character vector and a subset of c("learn","harvest","consensus","graph") that identifies the tasks to be done. Useful if part of the analysis was done previously, otherwise set to "All".

onCluster — A Boolean variable that is FALSE if the learning is not done on a computer cluster.

inds — The indices of the jobs that are included in the analysis.

perJob — The number of individual networks that are learnt by 1 job.

maxSeconds — An integer limiting computation time for each training job that runs locally, i.e., when oncluster=FALSE.

timeJob — The time in "hh:mm:ss" format requested for each job if they are running on a computer cluster.

bnCalculationJob — An R script used to submit jobs to the cluster. Set to NULL if not using a cluster. An example is provided at system.file("script", "bn.calculation.job.R", package="Pigengene")

seed — The random seed that can be set to an integer to reproduce the same results.

verbose — Integer level of verbosity. 0 means silent and higher values produce more details of computation.

naTolerance — Upper threshold on the fraction of entries per gene that can be missing. Genes with a larger fraction of missing entries are ignored. For genes with smaller fraction of NA entries, the missing values are imputed from their average expression in the other samples. See check.pigengene.input.

Details

For learning a Bayesian network with tens of nodes (eigengenes), bnNum=1000 or higher is recommended. Increasing consensusThresh generally results in a network with fewer arcs. Nagarajan et al. proposed a fundamental approach that determines this hyper-parameter based on the background noise. They use non-parametric bootstrapping, which is not implemented in the current package yet.

The default values for the rest of the hyper-parameters should be fine for most applications.

Value

A list of:

consensusThresh — The vector of thresholds as described in the arguments.

indvPath — The path where the individual networks were saved.

moduleFile — The file containing data in appropriate format for bnlearn package and the black-list arcs.

scoreFile — The file containing the record of the successively jobs and the scores of the corresponding individual networks.

consensusFile — The file containing the consensus network and its BDe and BIC scores.

bnModuleRes — The result of bn.module function. Useful mostly for development.

runs — A list containing the record of successful jobs.

`scores` The list saved in `scoreFile`.

`consensusThreshRes`
The full output of `consensus.thresh()` function.

`consensus1` The consensus Bayesian network corresponding to the first threshold. It is the output of `consensus` function and `consensus1$BN` is an object of `bn-class`.

`scorePlot` The output of `plot.scores` functions, containing the scores of individual networks.

`graphs` The output of `plot.graphS` function, containing the BDe score of the consensus network.

`timeTaken` An object of `difftime-class` recording the learning wall-time.

`use.Disease`, `use.Effect`, `use.Hartemink`
Some of the input arguments.

Note

Running the jobs on a cluster needs a proper `bnCalculationJob` script. Also, the unexported function `sbatch()` is adopted for a particular cluster and may need generalization on other clusters.

Author(s)

Amir Foroushani, Habil Zare, and Rupesh Agrahari

References

Hartemink A (2001). Principled Computational Methods for the Validation and Discovery of Genetic Regulatory Networks. Ph.D. thesis, School of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

Nagarajan, Radhakrishnan, et al. (2010) Functional relationships between genes associated with differentiation potential of aged myogenic progenitors. Frontiers in Physiology 1.

See Also

`bnlearn-package`, `Pigengene-package`, `compute.pigengene`

Examples

```
data(eigengenes33)
ms <- 10:20 ## A subset of modules for quick demonstration
amlE <- eigengenes33$aml[,ms]
mdsE <- eigengenes33$mds[,ms]
eigengenes <- rbind(amlE,mdsE)
Labels <- c(rep("AML",nrow(amlE)),rep("MDS",nrow(mdsE)))
names(Labels) <- rownames(eigengenes)
learnt <- learn.bn(Data=eigengenes, Labels=Labels,
  bnPath="bnExample", bnNum=10, seed=1)
bn <- learnt$consensus1$BN

## Visualize:
d1 <- draw.bn(BN=bn,nodeFontSize=14)

## What are the children of the Disease node?
childrenD <- bnlearn::children(x=bn, node="Disease")
print(childrenD)
```

```
## Fit the parameters of the Bayesian network:
fit <- bnlearn::bn.fit(x=bn, data=learnt$consensus1$Data, method="bayes",iss=10)

## The conditional probability table for a child of the Disease node:
fit[[childrenD[1]]]

## The fitted Bayesian network can be used for predicting the labels
## (i.e., values of the Disease node).
l2 <- predict(object=fit, node="Disease", data=learnt$consensus1$Data, method="bayes-lw")
table(Labels, l2)
```

`make.decision.tree` *Creates a decision tree to classify samples using the eigengenes values*

Description

A decision tree in Pigengene-package uses module eigengenes to build a classifier that distinguishes the different classes. Briefly, each eigengene is a weighted average of the expression of all genes in the module, where the weight of each gene corresponds to its membership in the module.

Usage

```
make.decision.tree(pigengene, Data,
  Labels = structure(pigengene$annotation[rownames(pigengene$eigengenes),
          1], names = rownames(pigengene$eigengenes)),
  testD = NULL, testL = NULL, selectedFeatures = NULL,
  saveDir = "C5Trees", minPerLeaf = NULL, useMod0 = FALSE,
  costRatio = 1, toCompact = NULL, noise = 0, noiseRepNum = 10, doHeat=TRUE,
  verbose = 0, naTolerance=0.05)
```

Arguments

`pigengene`	The pigengene object that is used to build the decision tree. See pigengene-class.
`Data`	The training expression data
`Labels`	Labels (condition types) for the (training) expression data. It is a named vector of characters. `Data` and `pigengene` will be subset according to these names.
`testD`	The test expression data, for example, from an independent dataset. Optional.
`testL`	Labels (condition types) for the (test) expression data. Optional.
`selectedFeatures`	A numeric vector determining the subset of eigengenes that should be used as potential predictors. By default ("All"), eigengenes for all modules are considered. See also `useMod0`.
`saveDir`	Where to save the plots of the tree(s).
`minPerLeaf`	Vector of integers. For each value, a tree will be built requiring at least that many nodes on each leaf. By default (`NULL`), several trees are built, one for each possible value between 2 and 10 percent of the number of samples.
`useMod0`	Boolean. Wether to allow the tree(s) to use the eigengene of module 0, which corresponds to the set of outlier, as a proper predictor.

costRatio	A numeric value effective only for 2 groups classification. The default value (1) considers the misclassification of both conditions as equally disadvatageous. Change this value to a larger or smaller value if you are more interested in the specificity of predictions for condition 1 or condition 2, respectively.
toCompact	An integer. The tree with this minPerLeaf value will be compacted (shrunk). Compacting in this context means reducing the number of required genes for the calculation of the relevant eigengenes and making the predictions using the tree. If TRUE or NULL (default), the (persumably) most general proper tree (corresponding to the largest value in the minPerLeaf vector for which a tree could be constructed) is compacted. Set to FALSE to turn off compacting.
noise, noiseRepNum	For development purposes only. These parameters allow investigating the effect of gaussian noise in the expression data on the accurracy of the tree for test data.
doHeat	Boolean. Set to FALSE not to plot the heatmaps for faster comoutation.
verbose	The integer level of verbosity. 0 means silent and higher values produce more details of computation.
naTolerance	Upper threshold on the fraction of entries per gene that can be missing. Genes with a larger fraction of missing entries are ignored. For genes with smaller fraction of NA entries, the missing values are imputed from their average expression in the other samples. See check.pigengene.input.

Details

This function passes the inut eigengenes and appropriate arguments to C5.0 function from C50 package.

The effect of test data: Only when both testD and testL are provided, the test data will be used for a) compacting the trees, b) plotting heatmaps of expression of genes in the compacted and full trees, and c) the noise analysis. If either of testD or testL is NULL, then Data and Labels are instead used for these purposes.

Value

A list with following elements:

call	The call that created the results
c5Trees	A list, with one element of class C5.0 for each attempted minNodesperleaf value. The list is named with the corresponding values as characters. An extra info element is added that includes information on the performance of the tree.
minPerLeaf	A numeric vector enumerating all of the attempted minPerLeaf values.
compacted	The full output of compact.tree function if toCompact is not FALSE
heat	The output of module.heatmap function for the full tree if doHeat is not FALSE
heatCompact	The output of module.heatmap function for the compacted tree if toCompact is not FALSE
noisy	The full output of noise.analysiy function if noise is not 0. For development and evaluation purposes only.
leafLocs	A matrix reporting the leaf for each sample on 1 row. The columns are named according to the correspoding minNodesperleaf value.
toCompact	Echos the toCompact input argument
costs	The cost matrix
saveDir	The directory where plots are saved in

Note

For faster computation in an initial, explanatory run, turn off compacting, which can take a few minutes, with `toCompact=FALSE`.

See Also

`Pigengene-package`, `compute.pigengene`, `compact.tree`, `C5.0`, `Pigengene-package`

Examples

```
## Data:
data(aml)
data(mds)
data(pigengene)
d1 <- rbind(aml,mds)

## Fiting the trees:
trees <- make.decision.tree(pigengene=pigengene, Data=d1,
  saveDir="trees", minPerLeaf=14:15, doHeat=FALSE,verbose=3,
  toCompact=15)
```

`make.filter` *Computes the filter based on a similarity network*

Description

Takes as input the similarity matrix of a graph (i.e., `network`) and an `epsilon` value. It computes a filter graph using the `epsilon` threshold. The dimention of the output filter matrix is the same as the input similarity network. It also produces two plots showing the weighted degrees of the input graph and degrees of the filter, respectively.

Usage

```
make.filter(network, epsilon, outPath=NULL)
```

Arguments

`network`	A matrix of similarity for the network.
`epsilon`	A threshold for deciding which edges to keep. If the similarity is less than 1/`epsilon` (i.e., distance > `epsilon`), the edge will be removed, and it will be kept in the filter graph otherwise.
`outPath`	A string determining the path where plots and results will be saved.

Value

A list with the following components:

`filt`	A matrix representing adjacency matrix of the computed filter graph. If the distance between two nodes in the similarity matrix is higher than epsilon, those nodes are connected in the filter graph (i.e., the corresponding entry in the adjacency matrix is 1). Otherwise, the corresponding entry is 0.
`epsilon`	The `epsilon` input.

Author(s)

Habil Zare and Neda Emami.

See Also

one.step.pigengene, apply.filter

Examples

```
data(aml)
data(mds)
d1 <- rbind(aml,mds)[, 1:200]
Labels <- c(rep("AML",nrow(aml)),rep("MDS",nrow(mds)))
names(Labels) <- rownames(d1)

p0 <- one.step.pigengene(Data=d1, saveDir=".", verbose=1,
                         seed=1, Labels=Labels, naTolerance=0.5,
    RsquaredCut=0.8, doNetOnly=TRUE)

##making the filter
made <- make.filter(network=p0$Network, epsilon=0.7, outPath=".")
```

---

mds *MDS gene expression profile*

Description

Gene expression profile of 164 myelodysplastic syndromes (MDS) cases from Mills et al. study. The profile was compared with the profile of 202 acute myeloid leukemia (AML) cases and only the 1000 most differentially expressed genes are included.

Usage

```
data("mds")
```

Format

A numeric matrix

Details

The columns and rows are named according to the genes Entrez, and patient IDs, respectively. The original data was produced using Affymetrix Human Genome U133 Plus 2.0 Miccoaray.Mills et al. study is part of the MILE Study (Microarray Innovations In LEukemia) program, and aimed at prediction of AML transformation in MDS.

Value

It is a `164*1000` numeric matrix.

Note

This profile includes data of the 25 chronic myelomonocytic leukemia (CMLL) cases that can have different expression signatures according to Mills et al.

Source

http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE15061

References

Mills, Ken I., et al. (2009). Microarray-based classifiers and prognosis models identify subgroups with distinct clinical outcomes and high risk of AML transformation of myelodysplastic syndrome. Blood 114.5: 1063-1072.

See Also

Pigengene-package, one.step.pigengene, aml, compute.pigengene

Examples

```
library(pheatmap)
data(mds)
pheatmap(mds[,1:20],show_rownames=FALSE)
```

---

message.if *Conditional messaging.*

Description

Messages only if verbose is more than 0 and write in a text file if provided.

Usage

```
message.if(me=NULL, verbose=0, txtFile=NULL, append=TRUE, ...)
```

Arguments

me	The Message. Can be a character vector.
verbose	A integer.
txtFile	The text file in which the message will be written. Set to NULL to disable.
append	logical. Set to FALSE to overwrite txtFile.
...	Arguments to be passed to capture.output.

Value

NULL

Author(s)

Amir Foroushani

Examples

```
message.if("Hello world!", verbose=1)
```

module.heatmap *Plots heatmaps for modules*

Description

This function takes as input a tree and an object from pigengene-class and per any module used in the tree, it plots one gene expression heatmap. Alternatively, it can plot a heatmap for every module in the given pigengene object.

Usage

```
module.heatmap(c5Tree=NULL, pigengene, mes=NULL, saveDir, testD = NULL,
  testL = NULL, pos = 0, verbose=0, doAddEigengene=TRUE, scalePngs=1, ...)
```

Arguments

c5Tree	A decision tree of class C50 that uses module eigengenes, or NULL. If NULL, expression plots for all modules are created.
pigengene	A object of pigengene-class, output of compute.pigengene
mes	A character vector that determines which modules to plot, e.g., c("ME3","ME5"). Set it to NULL to plot a heatmap for every module. This argument will be ignored if c5Tree is not NULL.
saveDir	Directory to save the plots
testD, testL	Optional. The matrix of (independent) test expression data, and the corresponding vector of labels. testL must be named according to the row names of testD.
pos	Number of genes to discard. Interpreted the same way as in compact.tree and preds.at
verbose	The integer level of verbosity. 0 means silent and higher values produce more details of computation.
doAddEigengene	If TRUE, the eigengene of each module will be added to the corresponding heatmap.
scalePngs	If not 1, the size of pngs will be adjusted using this parameter. A typical value would be 7.
...	Additional arguments. Passed to pheatmap.type

Value

A list of:

call	The call that created the results
saveDir	An echo of the input argument determining where the plots are saved

See Also

Pigengene-package, make.decision.tree, compact.tree, compute.pigengene

Examples

```
## Data:
data(aml)
data(mds)
data(pigengene)
d1 <- rbind(aml,mds)

## Plotting the heatmaps of all modules:
module.heatmap(pigengene=pigengene, saveDir="heatmaps", pos=0, verbose=1)

## Fiting the trees:
trees <- make.decision.tree(pigengene=pigengene, Data=d1,
  saveDir="trees", minPerLeaf=14:15, doHeat=FALSE,verbose=3,
  toCompact=15)

## Plotting the heatmaps of only the modules in the tree:
module.heatmap(c5Tree=trees$c5Trees[["15"]], pigengene=pigengene,
    saveDir="treeHeatmaps", pos=0, verbose=1)
```

`one.step.pigengene` *Runs the entire Pigengene pipeline*

Description

Runs the entire Pigengene pipeline, from gene expression to compact decision trees in a single function. It identifies the gene modules using coexpression network analysis, computes eigengenes, learns a Bayesian network, fits decision trees, and compact them.

Usage

```
one.step.pigengene(Data, saveDir="Pigengene", Labels, testD=NULL,
  testLabels=NULL, doBalance=TRUE, RsquaredCut=0.8, costRatio=1,
  toCompact=FALSE, bnNum=0, bnArgs=NULL, useMod0=FALSE, mit="All",
  verbose=0, doHeat=TRUE, seed=NULL, dOrderByW=TRUE, naTolerance=0.05,
  doNetOnly=FALSE, doReturNetworks=doNetOnly, idType="ENTREZID",
  pathwayDb=NULL, OrgDb=org.Hs.eg.db)
```

Arguments

`Data`	A matrix or data frame (or list of matrices or data frames) containing the training expression data, with genes corresponding to columns and rows corresponding to samples. Rows and columns must be named. For example, from RNA-Seq data, log(RPKM+1) can be used.
`Labels`	A (preferably named) vector containing the Labels (condition types) for the training Data. Or, if Data is a list, a list of label vectors corresponding to the data sets in Data. Names must agree with rows of `Data`.
`saveDir`	Directory to save the results.
`testD`	Test expression data with syntax similar to `Data`, possibly with different rows and columns. This argument is optional and can be set to `NULL` if test data are not available.

testLabels — A (preferably named) vector containing the Labels (condition types) for the test Data. This argument is optional and can be set to NULL if test data are not available.

doBalance — Boolean. Whether the data should be oversampled before identifying the modules so that each condition contribute roughly the same number of samples to clustering.

RsquaredCut — A threshold in the range [0,1] used to estimate the power. A higher value can increase power. For technical use only. See pickSoftThreshold for more details. A larger value generally leads to more modules.

costRatio — A numeric value, the relative cost of misclassifying a sample from the first condition vs. misclassifying a sample from the second condition.

toCompact — An integer value determining which decision tree to shrink. It is the minimum number of genes per leaf imposed when fitting the tree. Set to FALSE to skip compacting, to NULL to automatically select the maximum value. See make.decision.tree.

bnNum — Desired number of bootstraped Baysian networks. Set to 0 to skip BN learning.

bnArgs — A list of arguments passed to learn.bn function.

useMod0 — Boolean, whether to allow module zero (the set of outliers) to be used as a predictor in the decision tree(s).

mit — The "module identification type", a character vector determining the reference conditions for clustering. If 'All' (default), clustering is performed using the entire data regardless of condition.

verbose — The integer level of verbosity. 0 means silent and higher values produce more details of computation.

doHeat — If TRUE the heatmap of expression of genes in the modules that contribute to the the tree will be plotted.

seed — Random seed to ensure reproducibility.

dOrderByW — If TRUE, the genes will be ordered in the csv file based on their absolute weight in the corresponding module.

naTolerance — Upper threshold on the fraction of entries per gene that can be missing. Genes with a larger fraction of missing entries are ignored. For genes with smaller fraction of NA entries, the missing values are imputed from their average expression in the other samples. See check.pigengene.input.

doNetOnly — If TRUE, the pipeline does not continue after making the network and identifying the modules, e.g., eigengenes will not be computed.

doReturNetworks — A boolean value to determine whether to return Network, which is relatively a big matrix (typically GBs). Set to FALSE not to waste memory.

idType — A string describing the type of input gene ID e.g., "ENTREZID", "REFSEQ", "SYMBOL".

pathwayDb — A character vector determining which enrichment database to be used by the get.enriched.pw function e.g., "GO", "KEGG", "REACTOME", or "NCG". Set to NULL to skip the pathway enrichment analysis.

OrgDb — The reference data base to be used. Use e.g. org.Ce.eg.db for 'Celegans' when analysing Celegans data. If OrgDb is not NULL, Org must be NULL.

Details

This is the main function of the package Pigengene and performs several steps: First, modules are identified in the training expression data, according to mit argument i.e. based on coexpression behaviour in the corresponding conditions. Set it to "All" to use all training data for this step regardless of the condition. If a list of data frames is provided in Data, similarity networks on the data sets are computed and combined into one similarity network for the union of nodes across data sets.

Then, the eigengenes for each module and each sample are calculated, where the expression of an eigengene of a module in a sample is the weighted average of the expression of the genes in that module in the sample. Technically, an eigengene is the first principal component of the gene expression in a module. PCA ensures that the maximum variance accross all the training samples is explained by the eigengene.

Next, (optionally –if bnNum is set to a value greater than 0), several bootstrapped Bayesian networks are learned and combined into a consensus network, in order to detect and illustrate the probabilistic dependencies between the eigengenes and the disease subtype.

Next, decisision tree(s) are built that use the module eigengenes, or a subset of them, to distinguish the classes (`Labels`). The accurracy of trees is assessed on the train and (if provided) test data. Finally, the number of required genes for the calculation of the relevant eigengenes is reduced (the tree is 'compacted'). The accuracy of the tree is reassessed after removal of each gene.

Along the way, several self explanatory directories, heatmaps and plots are created and stored under `saveDir`. See `make.decision.tree` for the effect of test data in the process.

Value

A list with the following components:

`call`	The call that created the results.
`modules`	A named vector. Names are genes IDs and values are the corresponding module number.
`wgRes`	A list. The results of WGCNA clustering of the Data by `wgcna.one.step` if `Data` is one matrix.
`betaRes`	A list. The automatically selected beta (power) parameter which was used for the WGCNA clustering. It is the result of the call to `calculate.beta` using the expression data of `mit` conditions(s).
`pigengene`	The pigengene object computed for the clusters, result of `compute.pigengene`.
`leanrtBn`	A list. The results of `learn.bn` call for learning a Bayesian network using the eigengenes.
`selectedFeatures`	A vector of the names of module eigengenes that were considered during the construction of decision trees. If `bnNum` >0, this corresponds to the immediate neighbors of the Disease or Effect variable in the consensus network.
`c5treeRes`	A list. The results of `make.decision.tree` call for learning decision trees that use the eigengenes as features.

Note

The individual functions are exported to facilitated running the pipeline step-by-step in a customized way.

Author(s)

Amir Foroushani, Habil Zare, Rupesh Agrahari, and Meghan Short

References

Large-scale gene network analysis reveals the significance of extracellular matrix pathway and homeobox genes in acute myeloid leukemia, Foroushani A, Agrahari R, Docking R, Karsan A, and Zare H. In preparation.

See Also

check.pigengene.input, balance, calculate.beta, wgcna.one.step, compute.pigengene, project.eigen, learn.bn, make.decision.tree, blockwiseModules

Examples

```
library(org.Hs.eg.db)
data(aml)
data(mds)
d1 <- rbind(aml,mds)
Labels <- c(rep("AML",nrow(aml)),rep("MDS",nrow(mds)))
names(Labels) <- rownames(d1)
p1 <- one.step.pigengene(Data=d1,saveDir=".", bnNum=10, verbose=1, seed=1,
      Labels=Labels, toCompact=FALSE, doHeat=FALSE)
plot(p1$c5treeRes$c5Trees[["34"]])
```

---

pheatmap.type *Plots heatmap with clustering only within types.*

Description

This function first performs hierarchical clustering on samples (rows of data) within each condition. Then, plots a heatmap without further clustering of rows.

Usage

```
pheatmap.type(Data, annRow, type = colnames(annRow)[1],
doTranspose=FALSE, conditions="Auto",...)
```

Arguments

Data	A matrix with samples on rows and features (genes) on columns.
annRow	A data frame with 1 column or more. Row names must be the same as row names of Data.
type	The column of annRow used for determining the condition
doTranspose	If TRUE, the matrix will be transposed for the final plot. E.g., if the genes are on the columns of Data, they will be shown on rows of the heatmap.
conditions	A character vector that determines the conditions, and their order, to be included in the heatmap. By default ("Auto"), an alphabetical order of all available conditions in annRow will be used.
...	Additional arguments passed to pheatmap function.

Value

A list of:

pheatmapS	The results of pheatmap function for each condition
pheat	The output of final pheatmap function applied on all data
ordering	The ordering of the rows in the final heatmap
annRowAll	The row annotation used in the final heatmap

Note

If type is not determined, by default the first column of annRow is used.

Author(s)

Habil Zare

See Also

eigengenes33, pheatmap

Examples

```
data(eigengenes33)
d1 <- eigengenes33$aml
d2 <- eigengenes33$mds
Disease <- c(rep("AML",nrow(d1)), rep("MDS",nrow(d2)))
Disease <- as.data.frame(Disease)
rownames(Disease) <- c(rownames(d1), rownames(d2))
p1 <- pheatmap.type(Data=rbind(d1,d2),annRow=Disease,show_rownames=FALSE)
```

---

pigengene *An object of class* Pigengene

Description

This is a toy example object of class pigengene-class. It is used in examples of Pigengene-package. Gene expression profile of 202 acute myeloid leukemia (AML) cases from Mills et al. study. The profile was compared with the profile of 164 myelodysplastic syndromes (MDS) cases and only the 1000 most differentially expressed genes are included.

Usage

```
data("aml")
```

Format

An object of pigengene-class.

Details

The object is made using compute.pigengene function from aml and mds data as shown in the examples. The R CMD build --resave-data trick was used to reduce the size of saved object from 3.1 MB to 1.4 MB.

Value

It is an object of pigengene-class.

Source

http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE15061

References

Mills, Ken I., et al. (2009). Microarray-based classifiers and prognosis models identify subgroups with distinct clinical outcomes and high risk of AML transformation of myelodysplastic syndrome. Blood 114.5: 1063-1072.

See Also

Pigengene-package, pigengene-class, one.step.pigengene, mds, aml, compute.pigengene, project.eigen

Examples

```
library(pheatmap)
data(pigengene)
plot(pigengene,fontsize=12)

## To reproduce:
data(aml)
data(mds)
data(eigengenes33)
d1 <- rbind(aml,mds)
Labels <- c(rep("AML",nrow(aml)),rep("MDS",nrow(mds)))
names(Labels) <- rownames(d1)
modules33 <- eigengenes33$modules[colnames(d1)]
## Computing:
computed <- compute.pigengene(Data=d1, Labels=Labels, modules=modules33,
   saveFile="pigengene.RData", doPlot=FALSE, verbose=3)
class(computed)
plot(computed, fontsize=12, main="Reproduced")
```

---

pigengene-class *The pigengene class*

Description

A pigengene object holds the eigengenes, weights (memberships) and other related information.

Details

A object of class pigengene is the output of compute.pigengene function. It is a list. If doMinimize=TRUE, only the minimal elements needed to project eigengenes in a new dataset are included (i.e., see project.eigen(pigengene=NULL)[["projectionaries"]]). Otherwise, it contains at least the following components:

- call The call that created the results.

- Reptimes A named vector reporting the number of repeats for each condition in the oversampling process, which is done by the balance function.
- eigenResults A list including at least eigengenes and varExplained. If doWgcna=TRUE, then this list will be the full output of moduleEigengenes function with some fixes, e.g., we change eigengenes to a matrix, and use genes as its row names. Also, varExplained is named according to modules. Setting doWgcna=TRUE leads to more memory usage and a larger Pigengene object likely, with no advantage.
- Data The data matrix of gene expression.
- Labels A character vector giving the condition (type) for each sample (row of Data).
- eigengenes The matrix of eigengenes ordered based on selectedModules if provided. Rows correspond to samples.
- membership The matrix of weights of genes (rows) in all modules (columns).
- orderedModules The module assignment numeric vector named with genes and ordered based on module number.
- annotation A data frame containing labeling information useful in plotting. It has a column named "Condition". Rows have sample names.
- saveFile The file where the pigengene object is saved.
- weightsCsvFile The file containing the weights in csv format. See dOrderByW=TRUE.
- weights The weight matrix, which is also saved in csv format. It has more columns than membership but rows may be in a different order if dOrderByW=TRUE.
- heavyToLow If dOrderByW=TRUE, this will be the ordering of genes according to the modules the belong to, where the genes in each module are ordered based on the absolute value of the weights in that module. Also, the genes in the csv file are in this order.

For 2 or more groups (conditions), additional (optional) components include:

- pvalues A numeric matrix with columns "pValue", "FDR", and "Bonferroni". Rows correspond to modules. The null hypothesis is that the eigengene is expressed with the same distribution in all groups (conditions).
- log.pvalues A data frame with 1 column containing the logarithm of Bonferroni-adjusted pvalues in base 10.

See Also

Pigengene-package, plot.pigengene, wgcna.one.step, compute.pigengene, learn.bn, make.decision.tree

---

plot.pigengene *Plots and saves a* pigengene *object*

Description

Plots a couple of heatmaps of expression of the eigengenes, weights (memberships), and so on. Saves the plots in png format.

Usage

```
## S3 method for class 'pigengene'
plot(x, saveDir = NULL,
  DiseaseColors="Auto",
  fontsize = 35, doShowColnames = TRUE, fontsizeCol = 25,
  doClusterCols = ncol(pigengene$eigengenes) > 1,
  verbose = 2, doShowRownames = "Auto",
  pngfactor = max(2, ncol(pigengene$eigengenes)/16), do0Mem = FALSE, ...)
```

Arguments

x	The object from pigengene-class computed by compute.pigengene.
saveDir	The dirctory for saving the plots
DiseaseColors	A vector of characters determining color for each disease. Names should match the values in the first column of x$annotation.
fontsize	Passd to pheatmap.type
doShowColnames	Boolean
fontsizeCol	Numeric
doClusterCols	Boolean
verbose	The integer level of verbosity. 0 means silent and higher values produce more details of computation.
doShowRownames	Boolean
pngfactor	A numeric adjusting the size of the png files
do0Mem	If TRUE, module 0 genes are included in the membership heatmap.
...	Passd to pheatmap.type function

Details

Many of the arguments are passed to pheatmap.

Value

A list of:

heat	The full output of pheatmap functionion
heatNotRows	The full output of pheatmap.type function

Author(s)

Habil Zare ad Amir Foroushani

References

Large-scale gene network analysis reveals the significance of extracellular matrix pathway and homeobox genes in acute myeloid leukemia, Foroushani A, Agrahari R, Docking R, Karsan A, and Zare H. In preparation.

See Also

Pigengene-package, compute.pigengene, pheatmap.type

Examples

```
## Data:
data(aml)
data(mds)
data(eigengenes33)
d1 <- rbind(aml,mds)
Labels <- c(rep("AML",nrow(aml)),rep("MDS",nrow(mds)))
names(Labels) <- rownames(d1)
Labels <- c(rep("AML",nrow(eigengenes33$aml)),rep("MDS",nrow(eigengenes33$mds)))
names(Labels) <- rownames(d1)
toyModules <- eigengenes33$modules[colnames(d1)]
## Computing:
p1 <- compute.pigengene(Data=d1, Labels=Labels, modules=toyModules,
   saveFile="pigengene.RData", doPlot=TRUE, verbose=3)
plot(p1,saveDir="plots")
```

---

preds.at *Prediction using a possibly compacted tree*

Description

A decision tree in Pigengene uses module eigengenes to build a classifier that distincuishes two or more classes. Each eigengene is a weighted average of the expression of all genes in the module, where the weight of each gene corresponds to its membership in the module. Each modules might contain dozens to hundreds of genes, and hence the final classifier might depend on the expression of a large number of genes. In practice, it can be desireable to reduce the number of necessary genes used by a decision tree. This function is helpful in observing changes to the classification output after removing genes with lower weights membership. It determines how a given decision tree would classify the expression data after removing a certain number of genes from consideration.

Usage

```
preds.at(c5Tree, pigengene, pos=0, Data)
```

Arguments

c5Tree	A decision tree that uses eigengenes from the pigengene object to classify the samples from the expression data.
pigengene	A object of pigengene-class, output of compute.pigengene
pos	Number of genes to be removed from the consideration. Genes are removed in ascending order of their absolute weight in the relevant modules. If 0 (default), the prediction will be done without compacting.
Data	The expression possibly new data used for classification

Value

A list with following components:

predictions	The vector of predictions after neglecting pos number of genes
eigengenes	The values for the eigenges after neglecting pos number of genes

See Also

Pigengene-package, pigengene-class, make.decision.tree, compact.tree, compute.pigengene, module.heatmap, get.used.features, get.fitted.leaf, Pigengene-package

Examples

```
## Data:
data(aml)
data(mds)
data(pigengene)
d1 <- rbind(aml,mds)

## Fiting the trees:
trees <- make.decision.tree(pigengene=pigengene, Data=d1,
  saveDir="trees", minPerLeaf=15, doHeat=FALSE,verbose=3,
  toCompact=FALSE)
preds1 <- preds.at(c5Tree=trees$c5Trees[["15"]], pigengene=pigengene,
  pos=0, Data=d1)
```

`project.eigen` *Infers eigengenes for given expression data*

Description

This function projects (new) expression data onto the eigengenes of modules from another dataset. It is useful for comparing the expression behaviour of modules accross (biologically related yet independent) datasets, for evaluating the performance of a classifier on new datasets, and for examining the robustness of a pattern with regards to missing genes.

Usage

```
project.eigen(Data, saveFile = NULL, pigengene, naTolerance = 0.05,
  verbose = 0, ignoreModules = c())
```

Arguments

`Data`	A matrix or data frame of expression data to be projected. Genes correspond to columns, and rows correspond to samples. Rows and columns must be named. It is OK to miss a few genes originally used to compute the eigengenes, thereby, projection is robust to choose of platform.
`saveFile`	If not `NULL`, where to save the results in `.RData` format.
`pigengene`	An object of pigengene-class, usually created by compute.pigengene. If `NULL`, only `projectionaries` will be returned.
`naTolerance`	Upper threshold on the fraction of entries per gene that can be missing. Genes with a larger fraction of missing entries are ignored. For genes with smaller fraction of NA entries, the missing values are imputed from their average expression in the other samples. See check.pigengene.input.
`verbose`	The integer level of verbosity. 0 means silent and higher values produce more details of computation.

`ignoreModules` A vector of integers. In order to speed up the projection, it may be desirable to focus only on the eigengenes of a few interesting modules. In that case, the remaining modules can be listed here and will be ignored during projection (Optional).

Details

For each module, from the pigengene object, the weight (`membership`) of each gene is retrieved. The eigengene is computed (inferred) on the new data as alinear combination using the corresponding weights. The inferred eigengene vector will be normalized so that it has the same Euclidean norm as the original eigengene vector.

Value

A list of:

`projectionaries` The character vector of names of minimal elements needed to be in the `pigengene` object

`projected` The matrix of inferred (projected) eigengenes

`replacedNaNum` The number of `NA` entries in the input `Data` that were replaced with the the average expression of the corresponding gene

`tooNaGenes` A character vector of genes that were ignored because they had too many `NA`s

`notMatched` A character vector of genes in the original eigengene that could not be matched in the given input `Data`

Note

The new data should use the same type of biolocal identifiers (e.g. Gene Symbols or ENTREZIDs) as the original data for which the pigengene was constructed. It is, however, not required that the new data originate from the same type of technology, e.g. the eigengenes can be based on microarray experiments, whereas the new data comes from an RNA-Seq experiment. Nor is it necessary that the new datset contains measurements for all of the genes from the original modules.

See Also

`Pigengene-package`, `compute.pigengene` `moduleEigengenes`

Examples

```
## Data:
data(aml)
data(mds)
data(eigengenes33)
d1 <- rbind(aml,mds)
Labels <- c(rep("AML",nrow(aml)),rep("MDS",nrow(mds)))
names(Labels) <- rownames(d1)
toyModules <- eigengenes33$modules[colnames(d1)]
## Computing:
p1 <- compute.pigengene(Data=d1, Labels=Labels, modules=toyModules,
   saveFile="pigengene.RData", doPlot=TRUE, verbose=3, doMinimize=TRUE)

## How robust projecting is?
p2 <- project.eigen(Data=d1, pigengene = p1, verbose = 1)
```

```
plot(p1$eigengenes[,"ME1"],p2$projected[,"ME1"])
stats::cor(p1$eigengenes[,"ME1"],p2$projected[,"ME1"])
```

`pvalues.manova` *Computes pvalues for multi-class differential expression*

Description

Passes the arguments to `manova`, which performs multi-class analysis of variance.

Usage

```
pvalues.manova(Data, Labels)
```

Arguments

`Data`	A matrix or data frame containing the (expression) data, with genes corresponding to columns and rows corresponding to samples. Rows and columns must be named.
`Labels`	A (preferably named) vector containing the Labels (condition types). Names must agree with rows of `Data`

Value

A list with following elements:

`call`	The call that created the results
`pvals`	The matrix of pvalues with columns "pValue", "FDR", "Bonferroni". Rows are named according to genes, the columns of `Data`.
`manovaFit`	The full output of `manova` function.

Note

`oneway.test` function is a better generalizatoion to Welch's t-tst from 2-calsses to multi-class because it dose not assume that the variaces are necessarly equal. However, in practice, with "enough number of samples", the two approaches will lead to similar p-values.

Author(s)

Amir Foroushani

References

Krzanowski, W. J. (1988) _Principles of Multivariate Analysis. A User's Perspective._ Oxford.

Hand, D. J. and Taylor, C. C. (1987) _Multivariate Analysis of Variance and Repeated Measures._ Chapman and Hall.

B. L. Welch (1951), On the comparison of several mean values: an alternative approach.

See Also

`oneway.test`, `manova`, `compute.pigengene`

Examples

```
data(eigengenes33)
d1 <- rbind(eigengenes33$aml,eigengenes33$mds)
Labels <- c(rep("AML",nrow(eigengenes33$aml)),rep("MDS",nrow(eigengenes33$mds)))
names(Labels) <- rownames(d1)
ps <- pvalues.manova(Data=d1, Labels=Labels)
plot(log10(ps$pvals[,"Bonferroni"]))
```

`save.if` *Saves an object verbosely.*

Description

Saves an R object, and reports the size of the saved object in memory and on file.

Usage

```
save.if(x1, file, compress=TRUE, verbose=1, ...)
```

Arguments

`x1`	The object to be saved.
`file`	Where to save. If `NULL`, nothing will be saved.
`compress`	A Boolean or character sent to the `save` function. The default `TRUE` leads to compression using gzip. With "xz", maximum compression is obtained in expense of more save and load time.
`verbose`	A numeric determining how much detail will be printed.
`...`	Optional arguments to be passed to the `save` function.

Value

A list including file, and a vector of sizes of the object in memory and on file.

Author(s)

Amir Foroushani, and Habil Zare

See Also

`message.if`, `save`

Examples

```
m1 <- matrix(0, nrow=1000, ncol=1000)
save.if(m1, file="./m1.RData", verbose=3)
```

`wgcna.one.step` *Module identification*

Description

This function is a wrapper function for WGCNA::`blockwiseModules` and passes its arguments to it. Some other arguments are fixed.

Usage

```
wgcna.one.step(Data, power, saveDir=".", blockSize = "All", saveTOMs = FALSE,
   doThreads=FALSE, verbose = 0, seed = NULL)
```

Arguments

`Data`	A matrix or data frame containing the expression data, with genes corresponding to columns and rows corresponding to samples. Rows and columns must be named.
`power`	Soft-thresholding power for network construction
`saveDir`	The directory to save the results and plots. `NULL` will disable saving.
`blockSize`	The size of block when the data is too big. If not "All" (default) may introduce artifacts.
`saveTOMs`	Boolean determining if the TOM data should be saved, which can be hundreds of MBs and useful for identifying hubs.
`doThreads`	Boolean. Allows WGCNA to run a little faster using multi-threading but might not work on all systems.
`verbose`	The integer level of verbosity. 0 means silent and higher values produce more details of computation.
`seed`	Random seed to ensure reproducibility.

Details

`Data`, `power`, `blockSize`, `saveTOMs`, `verbose`, and `seed` are passed to WGCNA::`blockwiseModules`.

Value

A list with following components

`call`	The command that created the results
`genes`	The names of `Data` columns
`modules`	A numeric vector, named by `genes`, that reports the module (clustering) assignments.
`moduleColors`	A character vector, named by `genes`, that reports the color of each gene according to its module assignment
`net`	The full output of `blockwiseModules` function
`netFile`	The file in which the net object is saved
`power`	An echo of the `power` argument.

References

Langfelder P and Horvath S, WGCNA: an R package for weighted correlation network analysis. BMC Bioinformatics 2008, 9:559

See Also

blockwiseModules, pickSoftThreshold, calculate.beta

Examples

```
data(aml)
wgRes <- wgcna.one.step(Data=aml[,1:200], seed=1, power=7,
                            saveDir="wgcna", verbose=1)
```

Index