# Package 'SmCCNet'

January 13, 2024

**Title** Sparse Multiple Canonical Correlation Network Analysis Tool

**Version** 2.0.2

**Date** 2024-1-13

**Author** Weixuan Liu [aut, cre],
Yonghua Zhuang [aut, cre],
W. Jenny Shi [aut, cre],
Thao Vu [aut],
Iain Konigsberg [aut],
Katherine Pratte [aut],
Laura Saba [aut],
Katerina Kechris [aut]

**Maintainer** Weixuan Liu <weixuan.liu@cuanschutz.edu>

**Description** A canonical correlation based framework (SmCCNet) designed for the construction of phenotype-specific multi-omics networks. This framework adeptly integrates single or multiple omics data types along with a quantitative or binary phenotype of interest. It offers a streamlined setup process that can be tailored manually or configured automatically, ensuring a flexible and user-friendly experience.

**URL** https://github.com/KechrisLab/SmCCNet

**Depends** R (>= 3.5)

**Imports** EnvStats, future, pROC, spls, Matrix, pbapply, igraph, magrittr, rlist, furrr, purrr, pracma

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**biocViews** Network

**RoxygenNote** 7.2.3

**NeedsCompilation** no

**VignetteBuilder** knitr

**Repository** CRAN

**Date/Publication** 2024-01-13 21:50:17 UTC

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0), dplyr, reshape2, shadowtext, tidyverse, parallel, mltools, caret,

**Config/testthat/edition** 3

# R topics documented:



---

`aggregateCVSingle` *Aggregate and Save Cross-validation Result for Single-omics Analysis*

---

## Description

Saves cross-validation results in a table with the user-defined directory and outputs penalty term with the highest testing canonical correlation, lowest prediction error, and lowest scaled prediction error.

## Usage

```
aggregateCVSingle(CVDir, SCCAmethod = "SmCCA", K = 5, NumSubsamp = 500)
```

## Arguments

| | |
|---|---|
| `CVDir` | A directory where the result is stored. |
| `SCCAmethod` | The canonical correlation analysis method that is used in the model, used to name cross-validation table file, default is set to 'SmCCA'. |
| `K` | number of folds for cross-validation. |
| `NumSubsamp` | Number of subsampling used. |

## Value

A vector of length 3 with indices of the penalty term that (1) maximize the testing canonical correlation, (2) minimize the prediction error and (3) minimize the scaled prediction error.

---

`classifierEval` *Evaluation of Binary Classifier with Different Evaluation Metrics*

---

## Description

Evaluate binary classifier's performance with respect to user-selected metric (accuracy, auc score, precision, recall, f1 score) for binary phenotype.

## Usage

```
classifierEval(
  obs,
  pred,
  EvalMethod = "accuracy",
  BinarizeThreshold = 0.5,
  print_score = TRUE
)
```

## Arguments

| | |
|---|---|
| `obs` | Observed phenotype, vector consists of 0, 1. |
| `pred` | Predicted probability of the phenotype, vector consists of any value between 0 and 1 |
| `EvalMethod` | Binary classifier evaluation method, should be one of the following: 'accuracy' (default), 'auc', 'precision', 'recall', and 'f1'. |
| `BinarizeThreshold` | Cutoff threshold to binarize the predicted probability, default is set to 0.5. |
| `print_score` | Whether to print out the evaluation score, default is set to `TRUE`. |

## Value

An evaluation score corresponding to the selected metric.

## Examples

```
# simulate observed binary phenotype
obs <- rbinom(100,1,0.5)
# simulate predicted probability
pred <- runif(100, 0,1)
# calculate the score
pred_score <- classifierEval(obs, pred, EvalMethod = 'f1', print_score = FALSE)
```

`dataPreprocess` *preprocess a omics dataset before running omics SmCCNet*

## Description

Data preprocess pipeline to: (1) filter by coefficient of variation (cv), (2) center or scale data and (3) adjust for clinical covariates.

## Usage

```
dataPreprocess(
  X,
  covariates = NULL,
  is_cv = FALSE,
  cv_quantile = 0,
  center = TRUE,
  scale = TRUE
)
```

## Arguments

| | |
|---|---|
| `X` | dataframe with the size of $n$ by $p$, where $n$ is the sample size and $p$ is the feature size. |
| `covariates` | dataframe with covariates to be adjusted for. |
| `is_cv` | Whether to use coefficient of variation filter (small cv filter out). |
| `cv_quantile` | CV filtering quantile. |
| `center` | Whether to center the dataset X. |
| `scale` | Whether to scale the dataset X. |

## Value

Processed omics data with the size of nxp.

## Examples

```
X1 <- as.data.frame(matrix(rnorm(600, 0, 1), nrow = 60))
covar <- as.data.frame(matrix(rnorm(120, 0, 1), nrow = 60))
processed_data <- dataPreprocess(X = X1, covariates = covar, is_cv = TRUE,
cv_quantile = 0.5, center = TRUE, scale = TRUE)
```

`fastAutoSmCCNet` *Automated SmCCNet to Streamline the SmCCNet Pipeline*

## Description

Automated SmCCNet automatically identifies the project problem (single-omics vs multi-omics), and type of analysis (CCA for quantitative phenotype vs. PLS for binary phenotype) based on the input data that is provided. This method automatically preprocesses data, chooses scaling factors, subsampling percentage, and optimal penalty terms, then runs through the complete SmCCNet pipeline without the requirement for users to provide additional information. This function will store all the subnetwork information to a user-defined directory, as well as return all the global network and evaluation information. Refer to the automated SmCCNet vignette for more information.

## Usage

```
fastAutoSmCCNet(
  X,
  Y,
  AdjustedCovar = NULL,
  preprocess = FALSE,
  Kfold = 5,
  EvalMethod = "accuracy",
  subSampNum = 100,
  DataType,
  BetweenShrinkage = 2,
  ScalingPen = c(0.1, 0.1),
  CutHeight = 1 - 0.1^10,
  min_size = 10,
  max_size = 100,
  summarization = "NetSHy",
  saving_dir = getwd(),
  ncomp_pls = 3,
  tuneLength = 5,
  tuneRangeCCA = c(0.1, 0.5),
  tuneRangePLS = c(0.5, 0.9),
  seed = 123
)
```

## Arguments

| | |
|---|---|
| `X` | A list of matrices with same set and order of subjects ($n$). |
| `Y` | Phenotype variable of either numeric or binary, for binary variable, for binary $Y$, it should be binarized to 0,1 before running this function. |
| `AdjustedCovar` | A data frame of covariates of interest to be adjusted for through regressing-out approach, argument preprocess need to be set to TRUE if adjusting covariates are supplied. |

| | |
|---|---|
| preprocess | Whether the data preprocessing step should be conducted, default is set to FALSE. If regressing out covariates is needed, provide corresponding covariates to AdjustCovar argument. |
| Kfold | Number of folds for cross-validation, default is set to 5. |
| EvalMethod | The evaluation methods used to selected the optimal penalty parameter(s) when binary phenotype is given. The selections is among 'accuracy', 'auc', 'precision', 'recall', and 'f1', default is set to 'accuracy'. |
| subSampNum | Number of subsampling to run, the higher the better in terms of accuracy, but at a cost of computational time, we generally recommend 500-1000 to increase robustness for larger data, default is set to 100. |
| DataType | A vector indicating annotation of each dataset of $X$, example would be `c('gene', 'miRNA')`. |
| BetweenShrinkage | A real number > 0 that helps shrink the importance of omics-omics correlation component, the larger this number is, the greater the shrinkage it is, default is set to 2. |
| ScalingPen | A numeric vector of length 2 used as the penalty terms for scaling factor determination method: default set to 0.1 for both datasets, and should be between 0 and 1. |
| CutHeight | A numeric value specifying the cut height for hierarchical clustering, should be between 0 and 1, default is set to 1 - 0.1^10. |
| min_size | Minimally possible subnetwork size after network pruning, default set to 10. |
| max_size | Maximally possible subnetwork size after network pruning, default set to 100. |
| summarization | Summarization method used for network pruning and summarization, should be either 'NetSHy' or 'PCA'. |
| saving_dir | Directory where user would like to store the subnetwork results, default is set to the current working directory. |
| ncomp_pls | Number of components for PLS algorithm, only used when binary phenotype is given, default is set to 3. |
| tuneLength | The total number of candidate penalty term values for each omics data, default is set to 5. |
| tuneRangeCCA | A vector of length 2 that represents the range of candidate penalty term values for each omics data based on canonical correlation analysis, default is set to `c(0.1,0.5)`. |
| tuneRangePLS | A vector of length 2 that represents the range of candidate penalty term values for each omics data based on partial least squared discriminant analysis, default is set to `c(0.5,0.9)`. |
| seed | Random seed for result reproducibility, default is set to 123. |

## Value

This function returns the global adjacency matrix, omics data details, network clustering outcomes, and cross-validation results. Pruned subnetwork modules are saved in the directory specified by the user.

## Examples

```
# library(SmCCNet)
# set.seed(123)
# data("ExampleData")
# Y_binary <- ifelse(Y > quantile(Y, 0.5), 1, 0)
## single-omics PLS
# result <- fastAutoSmCCNet(X = list(X1), Y = as.factor(Y_binary), Kfold = 3,
#                           subSampNum = 100, DataType = c('Gene'),
#                           saving_dir = getwd(), EvalMethod = 'auc',
#                           summarization = 'NetSHy',
#                           CutHeight = 1 - 0.1^10, ncomp_pls = 5)
## single-omics CCA
# result <- fastAutoSmCCNet(X = list(X1), Y = Y, Kfold = 3, preprocess = FALSE,
#                           subSampNum = 50, DataType = c('Gene'),
#                           saving_dir = getwd(), summarization = 'NetSHy',
#                           CutHeight = 1 - 0.1^10)
## multi-omics PLS
# result <- fastAutoSmCCNet(X = list(X1,X2), Y = as.factor(Y_binary),
#                           Kfold = 3, subSampNum = 50,
#                           DataType = c('Gene', 'miRNA'),
#                           CutHeight = 1 - 0.1^10,
#                           saving_dir = getwd(), EvalMethod = 'auc',
#                           summarization = 'NetSHy',
#                           BetweenShrinkage = 5, ncomp_pls = 3)
## multi-omics CCA
# result <- fastAutoSmCCNet(X = list(X1,X2), Y = Y,
#                           K = 3, subSampNum = 50, DataType = c('Gene', 'miRNA'),
#                           CutHeight = 1 - 0.1^10,
#                           saving_dir = getwd(),
#                           summarization = 'NetSHy',
#                           BetweenShrinkage = 5)
```

---

# getAbar *Calculate similarity matrix based on canonical weights.*

---

## Description

Compute the similarity matrix based on the outer products of absolute canonical correlation weights, can be used for both single and multi-omics setting.

## Usage

```
getAbar(Ws, FeatureLabel = NULL)
```

### Arguments

| | |
|---|---|
| Ws | A canonical correlation weight vector or matrix. If Ws is a matrix, then each column corresponds to one weight vector. |
| FeatureLabel | A vector of feature labels for each feature in the adjacency matrix |

### Value

A $p \times p$ symmetric non-negative matrix.

### Examples

```
w <- matrix(rnorm(6), nrow = 3)
Ws <- apply(w, 2, function(x)return(x/sqrt(sum(x^2))))
abar <- getAbar(Ws,  FeatureLabel = c('omics1', 'omics2', 'omics3'))
```

---

## getCanCorMulti *Canonical Correlation Value for SmCCA*

---

### Description

Calculate canonical correlation value for SmCCA given canonical weight vectors and scaling factor

### Usage

```
getCanCorMulti(X, CCcoef, CCWeight, Y)
```

### Arguments

| | |
|---|---|
| X | A list of data each with same number of subjects. |
| CCcoef | A vector of scaling factors indicating weights for each pairwise canonical correlation. |
| CCWeight | A list of canonical weight vectors corresponds to each data in $X$. |
| Y | A phenotype matrix, should have only one column. |

### Value

A numeric value of the total canonical correlation

### Examples

```
library(SmCCNet)
data("ExampleData")
getCanCorMulti(list(X1,X2), CCcoef = c(1,1,1),
CCWeight = list(rnorm(500,0,1), rnorm(100,0,1)), Y = Y)
```

getCanWeightsMulti *Get Canonical Weight SmCCA Algorithm (No Subsampling)*

## Description

Run Sparse multiple Canonical Correlation Analysis (SmCCA) and return canonical weight vectors.

## Usage

```
getCanWeightsMulti(
  X,
  Trait = NULL,
  Lambda,
  CCcoef = NULL,
  NoTrait = TRUE,
  trace = FALSE,
  TraitWeight = FALSE
)
```

## Arguments

| | |
|---|---|
| X | A list of omics data each with n subjects. |
| Trait | An $n$ by 1 trait (phenotype) data for the same samples. |
| Lambda | Lasso penalty vector with length equals to the number of omics data ($X$). Lambda needs to be between 0 and 1. |
| CCcoef | Optional scaling factors for the SmCCA pairwise canonical correlations. If CCcoef = NULL (default), then the objective function is the total sum of all pairwise canonical correlations. It follows the column order of combn(T+1, 2), where T is the total number of omics data. |
| NoTrait | Whether or not trait (phenotype) information is provided, default is set to TRUE. |
| trace | Whether to display CCA algorithm trace, default is set to FALSE. |
| TraitWeight | Whether to return canonical weight for trait (phenotype), default is set to FALSE. |

## Value

A canonical weight vector with size of $p$ by 1.

## Examples

```
# This function is typically used as an internal function.
# It is also used when performing cross-validation,
# refer to multi-omics vignette for more detail.
# X <- list(X1,X2)
# result <- getCanWeightsMulti(X, Trait = as.matrix(Y), Lambda = c(0.5,0.5), NoTrait = FALSE)
# result <- getCanWeightsMulti(X, Trait = NULL, Lambda = c(0.5,0.5), NoTrait = TRUE)
# cccoef <- c(1,10,10)
```

```
# result <- getCanWeightsMulti(X, Trait = as.matrix(Y), CCcoef = cccoef,
#                              Lambda = c(0.5,0.5), NoTrait = FALSE)
```

---

`getOmicsModules` *Extract Omics Modules based on Similarity Matrix.*

---

## Description

Apply hierarchical tree cutting to the similarity matrix and extract multi/single-omics network modules.

## Usage

```
getOmicsModules(Abar, CutHeight = 1 - 0.1^10, PlotTree = TRUE)
```

## Arguments

| | |
|---|---|
| `Abar` | A similary matrix for all features (all omics data types). |
| `CutHeight` | Height threshold for the hierarchical tree cutting. Default is $1 - 0.1^{10}$. |
| `PlotTree` | Logical. Whether to create a hierarchical tree plot, default is set to `TRUE`. |

## Value

A list of multi/single-omics modules.

## Examples

```
set.seed(123)
w <- rnorm(5)
w <- w/sqrt(sum(w^2))
feature_name <- paste0('feature_', 1:5)
abar <- getAbar(w, FeatureLabel = feature_name)
modules <- getOmicsModules(abar, CutHeight = 0.5)
```

---

`getRobustWeightsMulti` *Run Sparse multiple Canonical Correlation Analysis and Obtain Canonical Weights (with Subsampling)*

---

## Description

SmCCNet algorithm with multi-omics data and quantitative phenotype. Calculate the canonical weights for SmCCA.

## Usage

```
getRobustWeightsMulti(
  X,
  Trait,
  Lambda,
  s = NULL,
  NoTrait = FALSE,
  SubsamplingNum = 1000,
  CCcoef = NULL,
  trace = FALSE,
  TraitWeight = FALSE
)
```

## Arguments

| | |
|---|---|
| X | A list of omics data each with n subjects. |
| Trait | An $n \times 1$ trait (phenotype) data matrix for the same n subjects. |
| Lambda | Lasso penalty vector with length equals to the number of omics data ($X$). Lambda needs to be between 0 and 1. |
| s | A vector with length equals to the number of omics data ($X$), specifying the percentage of omics feature being subsampled at each subsampling iteration. |
| NoTrait | Logical, default is FALSE. Whether trait information is provided. |
| SubsamplingNum | Number of feature subsamples. Default is 1000. Larger number leads to more accurate results, but at a higher computational cost. |
| CCcoef | Optional scaling factors for the SmCCA pairwise canonical correlations. If CCcoef = NULL (default), then the objective function is the total sum of all pairwise canonical correlations. This coefficient vector follows the column order of combn(T+1, 2) assuming there are T omics data and a phenotype data. |
| trace | Whether to display the CCA algorithm trace, default is set to FALSE. |
| TraitWeight | Whether to return canonical weight for trait (phenotype), default is set to FALSE. |

## Value

A canonical correlation weight matrix with $p = \sum_i p_i$ rows, where $p_i$ is the number of features for the $i$th omics. Each column is the canonical correlation weights based on subsampled features. The number of columns is SubsamplingNum.

## Examples

```
## For illustration, we only subsample 5 times.
set.seed(123)
X1 <- matrix(rnorm(600,0,1), nrow = 60)
X2 <- matrix(rnorm(600,0,1), nrow = 60)
Y <- matrix(rnorm(60,0,1), nrow = 60)
# Unweighted SmCCA
```

```
result <- getRobustWeightsMulti(X = list(X1, X2), Trait = Y, NoTrait = FALSE,
Lambda = c(0.5, 0.5),s = c(0.7, 0.7), SubsamplingNum = 20)
```

---

`getRobustWeightsMultiBinary` *Run Sparse multiple Canonical Correlation Analysis and Obtain Canonical Weights (with Subsampling)*

---

## Description

SmCCNet algorithm with multi-omics data and binary phenotype. This is a stepwise approach (1) use SmCCA to identify relationship between omics (exlude phenotype), (2) within highly connected omics features selected in step 1, identify relationship between these selected omics features and phenotype of interest with sparse PLS. First, it computes PLSDA by assuming outcome is continuous to extract multiple latent factors, then uses latent factors to fit logistic regression, and weight latent factor by regression parameters. Refer to multi-omics vignette for more detail.

## Usage

```
getRobustWeightsMultiBinary(
  X,
  Y,
  Between_Discriminate_Ratio = c(1, 1),
  SubsamplingPercent = NULL,
  CCcoef = NULL,
  LambdaBetween,
  LambdaPheno = NULL,
  SubsamplingNum = 1000,
  ncomp_pls = 3,
  EvalClassifier = FALSE,
  testData = NULL
)
```

## Arguments

| | |
|---|---|
| `X` | A list of omics data each with n subjects. |
| `Y` | A vector of binary variable, user needs to set the level of this variable to 0 and 1. |
| `Between_Discriminate_Ratio` | A vector with length 2 specifying the relative importance of between-omics relationship and omics-phenotype relationship. For instance a ratio of 1:1 (c(1,1) in the argument) means between-omics relationship and omics-phenotype relationship contribute equally to the canonical weights extraction. |
| `SubsamplingPercent` | A vector with length equal to the number of omics data (`X`), specifying the percentage of omics feature being subsampled at each subsampling iteration. |

| | |
|---|---|
| CCcoef | A vector of scaling factors only for between-omics relationship (exclude omics-phenotype). This coefficient vector follows the column order of combn(T, 2) when there are T omics data. |
| LambdaBetween | A vector of sparsity penalty value for each omics data to run the between-omics SmCCA, each penalty term should be within the range of 0 and 1. |
| LambdaPheno | A penalty term when running the sparse PLS with phenotype, penalty term should be within the range of 0 and 1. |
| SubsamplingNum | Number of feature subsamples. Default is 1000. Larger number leads to more accurate results, but at a higher computational cost, default is set to 1000. |
| ncomp_pls | Number of latent components for PLS, default set to 3. |
| EvalClassifier | If TRUE, the algorithm is at the phase of evaluating classification performance, and the latent factors from SPLSDA will be returned; if FALSE, the algorithm is at the phase of constructing multi-omics network, canonical weight will be returned. Default is set to FALSE. |
| testData | A list of testing omics data matrix, should have the exact same order as data list X, only used when EvalClassifier is set to TRUE for performing cross-validation, refer to multi-omics vignette for detail. |

## Value

If EvalClassifier is set to FALSE, a canonical correlation weight matrix is returned with combined omics data. Each column is the canonical correlation weights based on subsampled X features. The number of columns is SubsamplingNum. If EvalClassifier is set to TRUE, then latent factors from training and testing data will be returned for classifier evaluation.

## Examples

```
## For illustration, we only subsample 5 times.
set.seed(123)
X1 <- matrix(rnorm(600,0,1), nrow = 60)
X2 <- matrix(rnorm(600,0,1), nrow = 60)
Y_binary <- rbinom(60,1,0.5)

Ws <- getRobustWeightsMultiBinary(list(X1,X2), Y_binary,
      SubsamplingPercent = c(0.8,0.8), CCcoef = NULL,
      LambdaBetween = c(0.5,0.5), LambdaPheno = 0.1, SubsamplingNum = 10)
```

---

# getRobustWeightsSingle

*Single-omics SmCCA with Quantitative Phenotype*

---

## Description

Compute aggregated (SmCCA) canonical weights for single omics data with quantitative phenotype (subampling enabled).

## Usage

```
getRobustWeightsSingle(
  X1,
  Trait,
  Lambda1,
  s1 = 0.7,
  SubsamplingNum = 1000,
  trace = FALSE
)
```

## Arguments

| | |
|---|---|
| X1 | An $n \times p_1$ data matrix (e.g. mRNA) with $p_1$ features and $n$ subjects. |
| Trait | An $n \times 1$ trait (phenotype) data matrix for the same $n$ subjects. |
| Lambda1 | LASSO penalty parameter for X1. Lambda1 needs to be between 0 and 1. |
| s1 | Proportion of features in X1 to be included, default at s1 = 0.7. s1 needs to be between 0 and 1, default is set to 0.7. |
| SubsamplingNum | Number of feature subsamples. Default is 1000. Larger number leads to more accurate results, but at a higher computational cost. |
| trace | Whether to display the CCA algorithm trace, default is set to FALSE. |

## Value

A canonical correlation weight matrix with $p_1$ rows. Each column is the canonical correlation weights based on subsampled X1 features. The number of columns is SubsamplingNum.

## Examples

```
## For illustration, we only subsample 5 times.
set.seed(123)

# Single Omics SmCCA
W1 <- getRobustWeightsSingle(X1, Trait = Y, Lambda1 = 0.05,
  s1 = 0.7,
  SubsamplingNum = 5, trace = FALSE)
```

---

# getRobustWeightsSingleBinary

*Single-omics SmCCA with Binary Phenotype*

---

## Description

Compute aggregated (SmCCA) canonical weights for single omics data with quantitative phenotype (subampling enabled).

## Usage

```
getRobustWeightsSingleBinary(
  X1,
  Trait,
  Lambda1,
  s1 = 0.7,
  SubsamplingNum = 1000,
  K = 3
)
```

## Arguments

| | |
|---|---|
| X1 | An $n \times p_1$ data matrix (e.g. mRNA) with $p_1$ features and $n$ subjects. |
| Trait | An $n \times 1$ trait (phenotype) data matrix for the same $n$ subjects. |
| Lambda1 | LASSO penalty parameter for X1. Lambda1 needs to be between 0 and 1. |
| s1 | Proportion of mRNA features to be included, default at s1 = 0.7. s1 needs to be between 0 and 1, default is set to 0.7. |
| SubsamplingNum | Number of feature subsamples. Default is 1000. Larger number leads to more accurate results, but at a higher computational cost. |
| K | Number of hidden components for PLSDA, default is set to 3. |

## Value

A partial least squared weight matrix with $p_1$ rows. Each column is the canonical correlation weights based on subsampled X1 features. The number of columns is SubsamplingNum.

## Examples

```
X <- matrix(rnorm(600,0,1), nrow = 60)
Y <- rbinom(60,1,0.5)
Ws <- getRobustWeightsSingleBinary(X1 = X, Trait = as.matrix(Y), Lambda1 = 0.8,
0.7, SubsamplingNum = 10)
```

networkPruning *Prunes Subnetwork and Return Final Pruned Subnetwork Module*

## Description

Prunes subnetworks with network pruning algorithm (see multi-omics vignette for detail), and save the final pruned subnetwork to the user-defined directory. The final subnetwork is an .Rdata file with a name 'size_m_net_ind.Rdata', where $m$ is the final pruned network size, and ind is the index of the subnetwork module after hierarchical clustering.

## Usage

```
networkPruning(
  Abar,
  CorrMatrix,
  data,
  Pheno,
  type,
  ModuleIdx,
  min_mod_size = 10,
  max_mod_size,
  damping = 0.9,
  method = "NetSHy",
  saving_dir
)
```

## Arguments

| | |
|---|---|
| Abar | Adjacency matrix of subnetwork with size $m^*$ by $m^*$ after hierarchical clustering. |
| CorrMatrix | The correlation matrix of features in Abar, it should be $m^*$ by $m^*$ as well. |
| data | The omics data for the subnetwork. |
| Pheno | The trait (phenotype) data used for network pruning. |
| type | A vector with length equal to total number of features in the adjacency matrix indicating the type of data for each feature. For instance, for a subnetwork with 2 genes and a protein, the type argument should be set to c('gene', 'gene', 'protein'), see multi-omics vignette for more information. |
| ModuleIdx | The index of the network module that summarization score is intended to be stored, this is used for naming the subnetwork file in user-defined directory. |
| min_mod_size | The minimally possible subnetwork size for the pruned network module, should be an integer from 1 to the largest possible size of the subnetwork, default is set to 10. |
| max_mod_size | the maximally possible subnetwork size for the pruned network module, should be an integer from 1 to the largest possible size of the subnetwork, and it needs to be greater than the value specified in min_mod_size. |

| | |
|---|---|
| damping | damping parameter for the PageRank algorithm, default is set to 0.9, see igraph package for more detail. |
| method | Selection between NetSHy' and 'PCA', specifying the network summarization method used for network pruning, default is set to NetSHy. |
| saving_dir | User-defined directory to store pruned subnetwork. |

## Value

A file stored in the user-defined directory, which contains the following: (1) correlation_sub: correlation matrix for the subnetwork. (2) M: adjacency matrix for the subnetwork. (3) omics_corelation_data: individual molecular feature correlation with phenotype. (4) pc_correlation: first 3 PCs correlation with phenotype. (5) pc_loading: principal component loadings. (6) pca_x1_score: principal component score and phenotype data. (7) mod_size: number of molecular features in the subnetwork. (8) sub_type: type of feature for each molecular features.

## Examples

```
library(SmCCNet)
set.seed(123)
w <- rnorm(20)
w <- w/sqrt(sum(w^2))
labels <- paste0('feature_', 1:20)
abar <- getAbar(w, FeatureLabel = labels)
modules <- getOmicsModules(abar, CutHeight = 0.1)
x <- X1[ ,seq_len(20)]
corr <- stats::cor(x)
# display only example
# networkPruning(abar, corr, data = x, Pheno = Y,
# ModuleIdx = 1,  min_mod_size = 3, max_mod_size = 10, method = 'NetSHy', saving_dir =
# )
```

---

# scalingFactorInput *Scaling Factor Input Prompt*

---

## Description

Input the vector of the annotation of each type of dataset in the data list X (e.g., `c('gene', 'protein')`), and return prompt ask the user to supply the scaling factor for SmCCNet algorithm to prioritize the correlation structures of interest. All scaling factor values supplied should be numeric and nonnegative.

## Usage

```
scalingFactorInput(DataType = NULL)
```

### Arguments

| | |
|---|---|
| DataType | A character vector that contains the annotation of each type of omics dataset in X. |

### Value

A numeric vector of scaling factors.

### Examples

```
# not run
# scalingFactorInput(c('gene','mirna', 'phenotype'))
```

---

## summarizeNetSHy *NetSHy Summarization Score*

---

### Description

Implement NetSHy network summarization via a hybrid approach (Vu et al.,) to summarize network by considering the network topology with Laplacian matrix.

### Usage

```
summarizeNetSHy(X, A, npc = 1)
```

### Arguments

| | |
|---|---|
| X | An $n \times m$ data matrix with $m$ features and $n$ subjects. |
| A | Corresponding adjacency matrix of size $p$ by $p$. |
| npc | Number of principal components used to summarize the network, default is set to 1. |

### Value

A list consists of (1) subject-level network summarization score, (2) principal component importance information: standard deviation, percent of variance explained, and cumulative proportion of variance explained, and (3) principal component feature-level loadings.

### References

Vu, Thao, Elizabeth M. Litkowski, Weixuan Liu, Katherine A. Pratte, Leslie Lange, Russell P. Bowler, Farnoush Banaei-Kashani, and Katerina J. Kechris. "NetSHy: network summarization via a hybrid approach leveraging topological properties." Bioinformatics 39, no. 1 (2023): btac818.

## Examples

```
# simulate omics data
OmicsData <- matrix(rnorm(200,0,1), nrow = 10, ncol = 20)
# simulate omics adjacency matrix
set.seed(123)
w <- rnorm(20)
w <- w/sqrt(sum(w^2))
featurelabel <- paste0('omics',1:20)
abar <- getAbar(w, FeatureLabel = featurelabel)
# extract NetSHy summarization score
netshy_score <- summarizeNetSHy(OmicsData, abar)
```

---

# X1 *A synthetic mRNA expression dataset.*

---

## Description

A matrix containing simulated mRNA expression levels for 358 subjects (rows) and 500 features (columns).

## Usage

```
X1
```

## Format

An object of class `matrix` (inherits from `array`) with 358 rows and 500 columns.

---

# X2 *A synthetic miRNA expression dataset.*

---

## Description

A matrix containing simulated miRNA expression levels for 358 subjects (rows) and 100 features (columns).

## Usage

```
X2
```

## Format

An object of class `matrix` (inherits from `array`) with 358 rows and 100 columns.

Y *A synthetic phenotype dataset.*

## Description

A matrix containing simulated quantitative phenotype measures for 358 subjects (rows).

## Usage

```
Y
```

## Format

An object of class `matrix` (inherits from `array`) with 358 rows and 1 columns.

# Index