Manual of NetSAM

Jing Wang, Bing Zhang

March 1, 2017

1 Introduction

The last decade of systems biology research has demonstrated that networks rather than individual genes govern the onset and progression of complex diseases. Meanwhile, real-world complex networks usually exhibit hierarchical organization, in which nodes can be combined into groups that can be further combined into larger groups, and so on over multiple scales. Thus, identifying the hierarchical organization of a network becomes indispensable in complex disease studies. A traditional and useful method for revealing hierarchical architecture of network is hierarchical clustering, which groups data over a variety of scales by creating a hierarchical tree. However, hierarchical clustering has three major limitations. First, there are many different leaf node orderings consistent with the structure of a hierarchical tree, and hierarchical clustering does not optimize the ordering. Secondly, hierarchical clustering does not assess the statistical significance of the modular organization of a network. Finally, it does not specify relevant hierarchical levels and modules at different scales. To address these limitations, we developed the NetSAM (Network Seriation and Modularization) package which will identify the hierarchical modules from a network (network modularization) and find a suitable linear order for all leaves of the identified hierarchical organization (network seriation). NetSAM takes an edge-list representation of a weighted or unweighted network as an input and generates as files that can be used as an input for the one-dimensional network visualization tool NetGestalt (http://www.netgestalt.org) or other network analysis. NetSAM uses random walk distance-based hierarchical clustering to identify the hierarchical modules of the network and then uses the optimal leaf ordering (OLO) method to optimize the one-dimensional ordering of the genes in each module by minimizing the sum of the pair-wise random walk distance of adjacent genes in the ordering. The detailed description of the NetSAM method can be found in our published Nature Methods paper ”NetGestalt: integrating multidimensional omics data over biological networks” (http://www.nature.com/nmeth/journal/v10/n7/full/nmeth.2517.html).

The NetSAM package can also generate correlation network (e.g. co-expression network) based on the input matrix data, perform seriation and modularization analysis for correlation network and calculate the associations between the sample features and modules or identify the associated GO terms for the modules.

2 Environment

NetSAM requires R version 3.0.0 or later, which can be downloaded from the website http://www.r-project.org/. Because the seriation step requires pair-wise distance between all nodes, NetSAM is memory consuming. We recommend to use the 64 bit version of R to run the NetSAM. For networks with less than 10,000 nodes, we recommend to use a computer with 8GB memory. Using our computer with 2.7 GHz Intel Core i5 processor and 8 GB 1333 MHz DDR3 memory, NetSAM took 402 seconds to analyze the HPRD network (http://www.hprd.org) with 9198 nodes. For networks with more than 10,000 nodes, a computer with at least 16GB memory is recommended. NetSAM package requires the following packages: igraph (>=0.6-1), seriation (>=1.0-6), WGCNA (>=1.34.0), snow (>=0.4-1), doSNOW (>=1.0.6), foreach (>=1.4.0), tools (>=3.0.0), biomaRt (>=2.18.0), GO.db (>=2.10.0), R2HTML (>=2.2.0) and survival (>=2.37-7), which can be installed as follows.

```
>install.packages(”igraph”)

>install.packages(”seriation”)

>install.packages(”WGCNA”)
```

>install.packages(”snow”)

>install.packages(”doSNOW”)

>install.packages(”foreach”)

>source(”http://bioconductor.org/biocLite.R”)

>biocLite(”biomaRt”)

>biocLite(”GO.db”)

>install.packages(”R2HTML”)

>install.packages(”survival”)

3 Network Seriation and Modularization

After building up the basic environment mentioned above, the users can install the NetSAM package and use it to analyze networks.

```
> library("NetSAM")
```

```
==========================================================================
*
*  Package WGCNA 1.51 loaded.
*
*    Important note: It appears that your system supports multi-threading,
*    but it is not enabled within WGCNA in R.
*    To allow multi-threading within WGCNA with all available cores, use
*
*          allowWGCNAThreads()
*
*    within R. Use disableWGCNAThreads() to disable threading if necessary.
*    Alternatively, set the following environment variable on your system:
*
*          ALLOW_WGCNA_THREADS=<number_of_processors>
*
*    for example
*
*          ALLOW_WGCNA_THREADS=8
*
*    To set the environment variable in linux bash shell, type
*
*           export ALLOW_WGCNA_THREADS=8
*
*     before running R. Other operating systems or shells will
*     have a similar command to achieve the same aim.
*
==========================================================================
```

```
> inputNetworkDir <- system.file("extdata","exampleNetwork.net",package="NetSAM")
> outputFileName <- paste(getwd(),"/NetSAM",sep="")
> result <- NetSAM(inputNetwork=inputNetworkDir, outputFileName=outputFileName, outputFormat="nsm",
+ edgeType="unweighted", map_to_genesymbol=FALSE, organism="hsapiens", idType="auto", minModule=0.00
+ stepIte=FALSE, maxStep=4, moduleSigMethod="cutoff", modularityThr=0.2, ZRanNum=10,
+ PerRanNum=100, ranSig=0.05, edgeThr=(-1), nodeThr=(-1), nThreads=3)
```

```
Allowing parallel execution with up to 7 working processes.
```

```
Identifying the hierarchical modules of the network...
Starting to analysis connected component 1!
Evaluating networks in Level 1 ...
Network modularity: 0.5512183

Evaluating networks in Level 2 ...
Modularity of network 1: 0.2083333

Modularity of network 2: 0.2915519

Modularity of network 3: 0.377551

Modularity of network 4: 0.4114896

Modularity of network 5: 0.3669114

Modularity of network 6: 0.4228597

Modularity of network 7: 0.25

Modularity of network 8: 0.1985731

Modularity of network 9: 0.21875

Modularity of network 10: 0.0798611

Modularity of network 11: 0

Evaluating networks in Level 3 ...
Modularity of network 1: 0

Modularity of network 2: 0.2040816

Modularity of network 3: 0.1417769

Modularity of network 4: 0.3047337

Modularity of network 5: 0.3584808

Modularity of network 6: 0.1725207

Modularity of network 7: 0.1982249

Modularity of network 8: 0

Modularity of network 9: 0

Modularity of network 10: 0.1942149

Modularity of network 11: 0.2904

Modularity of network 12: 0.2366864

Modularity of network 13: 0.3010204

Modularity of network 14: 0.04195011

Modularity of network 15: 0.1938775
Modularity of network 16: 0.1064815

Modularity of network 17: 0.21875

Evaluating networks in Level 4 ...
Modularity of network 1: 0.03061226

Modularity of network 2: 0.1577778

Modularity of network 3: 0.1342593

Modularity of network 4: 0.08000001

Modularity of network 5: 0.2167969

Modularity of network 6: 0.21875

Modularity of network 7: 0

Modularity of network 8: 0

Modularity of network 9: 0.08000001

Evaluating networks in Level 5 ...
Modularity of network 1: 0



Reordering the genes in the one dimentional layout...
NetSAM identified 39 modules in 5 levels!
Processing completed!

>
```

3.1 Input

This section describes the arguments of the NetSAM function:

1. *inputNetwork* is the network under analysis. *inputNetwork* can be the directory of the input network file including the file name with "net" extension. If *edgeType* is "unweighted", each row represents an edge with two node names separated by a tab or space. If *edgeType* is "weighted", each row represents an edge with two node names and edge weight separated by a tab or space. *inputNetwork* can also be a data object in R (data object must be igraph, graphNEL, matrix or data.frame class).

2. *outputFileName* is the name of the output file.

3. *outputFormat* is the format of the output file. "nsm" format can be used as an input in NetGestalt (http://www.netgestalt.org), "gmt" format can be used to do other network analysis (e.g. as an input in GSEA (Gene Set Enrichment Analysis) to do module enrichment analysis) and "none" means the function will not output a file.

4. Because pathway enrichment analysis in NetGestalt is based on gene symbol, setting *map_to_genesymbol* as TRUE can transform other ids in the network into gene symbols and thus allow users to do functional analysis based on the identified modules. If the input network is not a biology network or users do not plan to do enrichment analysis in the NetGestalt, users can set *map_to_genesymbol* as FALSE. The default is FALSE.

5. *organism* is the organism of the input network. Currently, the package supports the following nine organisms: hsapiens, mmusculus, rnorvegicus, drerio, celegans, scerevisiae, cfamiliaris, dmelanogaster and athaliana. The default is "hsapiens".

6. *idType* is the id type of the input network. MatSAM will use BiomaRt package to transform the input ids to gene symbols based on *idType*. The users can also set *idType* as "auto" that means MatSAM will automatically search the id type of the input data. However, this may take 10 minutes based on the users' internet speed. The default is "auto".

7. *minModule* is the minimum percentage of nodes in a module. The minimum size of a module is calculated by multiplying *minModule* by the number of nodes in the whole network. If the size of a module identified by the function is less than the minimum size, the module will not be further partitioned into sub-modules. The default is 0.003 which means the minimum module size is 30 if there are 10,000 nodes in the whole network. If the minimum module size is less than 5, the minimum module size will be set as 5. The *minModule* should be less than 0.2.

8. Because NetSAM uses random walk distance-based hierarchical clustering to reveal the hierarchical organization of an input network, it requires a specified length of the random walks. If *stepIte* is TRUE, the function will test a range of lengths ranging from 2 to *maxStep* to get the optimal length. Otherwise, the function will directly use *maxStep* as the length of the random walks. The default *maxStep* is 4. Because optimizing the length of the random walks will take a long time, if the network is too big (e.g. the number of edges is over 200,000), we suggest to set *stepIte* as FALSE.

9. To test whether a network under consideration has a non-random internal modular organization, the function provides three options for parameter *moduleSigMethod*: "cutoff", "zscore" and "permutation". "cutoff" means if the modularity score of the network is above a specified cutoff value, the network will be considered to have internal organization and will be further partitioned. For "zscore" and "permutation", the function will first generate a set of random modularity scores. Based on a unweighted network, the function uses the edge switching method to generate a given number of random networks with the same number of nodes and an identical degree sequence and calculates the modularity scores for these random networks. Based on a weighted network, the function shuffles the weights of all edges and calculate the modularity scores for network with random weights. Then, "zscore" method will transform the real modularity score to a z score based on the random modularity scores and then transform the z score to a p value assuming a standard normal distribution. The "permutation" method will compare the real modularity score with the random ones to calculate a p value. Finally, under a specified significance level, the function determines whether the network can be further partitioned. The default is "cutoff".

10. *modularityThr* is the threshold of modularity score for the "cutoff" method. The default is 0.2.

11. *ZRanNum* is the number of random networks that will be generated for the "zscore" calculation. The default is 10.

12. *PerRanNum* is the number of random networks that will be generated for the "permutation" p value calculation. The default is 100.

13. *ranSig* is the significance level for determining whether a network has non-random internal modular organization for the "zscore" or "permutation" methods.

14. If the network is too big, it will take a long time to identify the modules. Thus, the function provides the option to set the threshold of number of edges and nodes as *edgeThr* and *nodeThr*. If the size of network is over the threshold, the function will stop and the users should change the parameters and re-run the function. We suggest to set the threshold for node as 12,000 and the threshold for edge as 300,000. The default is -1 which means there is no limitation for the network.

15. *nThreads* is the number of cores for parallel computing. The default is 3.

3.2 Output

If output format is "nsm", the function will output not only a "nsm" file but also a list object containing module information, gene order information and network information. If output format is "gmt", the function will output the "gmt" file and a matrix object containing the module and annotation information.

4 Network Analyzer

The *NetAnalyzer* function will calculate the degree, clustering coefficient, betweeness and closeness centrality for each node and the shortest path distance for each pair of nodes. The function can also plot the distributions for these measurements.

```
> library("NetSAM")
> inputNetworkDir <- system.file("extdata","exampleNetwork.net",package="NetSAM")
> outputFileName <- paste(getwd(),"/NetSAM",sep="")
> NetAnalyzer(inputNetworkDir,outputFileName,"unweighted")
```

```
[1] "Alpha = 1.389"
[1] "R square = 0.861"
null device
          1
```

4.1 Input

This section describes the arguments of the *mergeDuplicate* function: 1. *inputNetwork* is the network under analysis. *inputNetwork* can be the directory of the input network file including the file name with "net" extension. If *edgeType* is "unweighted", each row represents an edge with two node names separated by a tab or space. If *edgeType* is "weighted", each row represents an edge with two node names and edge weight separated by a tab or space. *inputNetwork* can also be a data object in R (data object must be igraph, graphNEL, matrix or data.frame class).

2. *outputFileName* is the name of the output file.

4.2 Output

The *NetAnalyzer* function will output two "txt" files and five "pdf" files. Two "txt" files contain degree, clustering coefficient, betweeness and closeness centrality for each node and the shortest path distance for each pair of nodes. Five "pdf" files are the distributions of these measurements.

5 mergeDuplicate

The *mergeDuplicate* function will merge the duplicate Ids in the matrix and return the matrix with unique Ids. This function can also used to merge the duplicate mapped Ids when transforming the Ids of data matrix to other Ids.

```
> library("NetSAM")
> inputMatDir <- system.file("extdata","exampleExpressionData_nonsymbol.cct",package="NetSAM")
> inputMat <- read.table(inputMatDir,header=TRUE,sep="\t",stringsAsFactors=FALSE,check.names=FALSE)
> mergedData <- mergeDuplicate(id=inputMat[,1],data=inputMat[,2:ncol(inputMat)],collapse_mode="maxSD
```

5.1 Input

This section describes the arguments of the *mergeDuplicate* function: 1. *id* is the duplicate Ids that should be a vector object in R.

2. *data* is the corresponding data matrix that has the same number of rows with *id* and should be a matrix or data.frame object in R.

3. *collapse_mode* is the method to collapse duplicate ids. "mean", "median", "maxSD", "maxIQR", "max" and "min" represent the mean, median, max standard deviation, max interquartile range, maximum and minimum of values for ids in each sample. The default is "maxSD".

5.2 Output

The *mergeDuplicate* function will return the data matrix with unique Ids.

6 Mapping other ids to gene symbols

To perform enrichment analysis in NetGestalt, the gene ids in each module should be gene symbols. The mapToSymbol function can transform other ids from a gene list, network, matrix, sbt file or sct file to gene symbols.

```
> library("NetSAM")
> print("transform ids from a gene list to gene symbols...")

[1] "transform ids from a gene list to gene symbols..."

> geneListDir <- system.file("extdata","exampleGeneList.txt",package="NetSAM")
> geneList <- read.table(geneListDir,header=FALSE,sep="\t",stringsAsFactors=FALSE)
> geneList <- as.vector(as.matrix(geneList))
> geneList_symbol <- mapToSymbol(inputData=geneList, organism="hsapiens", inputType="genelist",idType
> print("transform ids in the input network to gene symbols...")

[1] "transform ids in the input network to gene symbols..."

> inputNetwork <- system.file("extdata","exampleNetwork_nonsymbol.net",package="NetSAM")
> network_symbol <- mapToSymbol(inputData=inputNetwork,organism="hsapiens",inputType="network",idType

Transforming the ids in the input network to gene symbols...

> print("transform ids in the input matrix to gene symbols...")

[1] "transform ids in the input matrix to gene symbols..."

> inputMatDir <- system.file("extdata","exampleExpressionData_nonsymbol.cct",package="NetSAM")
> matrix_symbol <- mapToSymbol(inputData=inputMatDir,organism="hsapiens",inputType="matrix",idType="a
> print("transform ids in the sbt file to gene symbols...")

[1] "transform ids in the sbt file to gene symbols..."

> inputSBTDir <- system.file("extdata","exampleSBT.sbt",package="NetSAM")
> sbt_symbol <- mapToSymbol(inputData= inputSBTDir,organism="hsapiens",inputType="sbt",idType="affy_h
> print("transform ids in the sct file to gene symbols...")

[1] "transform ids in the sct file to gene symbols..."

> inputSCTDir <- system.file("extdata","exampleSCT.sct",package="NetSAM")
> sct_symbol <- mapToSymbol(inputData= inputSCTDir,organism="hsapiens",inputType="sct",idType="affy_h
```

6.1 Input

This section describes the arguments of the *mapToSymbol* function: 1. *inputData*: mapToSymbol function supports five different types of data: "genelist", "network", "matrix", "sbt" and "sct". For "genelist" type, *inputData* should be a vector containing gene ids. For "network" type, *inputData* can be the directory of the input network file including the file name with "net" extension. If *edgeType* is "unweighted", each row represents an edge with two node names separated by a tab or space. If *edgeType* is ”weighted”, each row represents an edge with two node names and edge weight separated by a tab or space. *inputNetwork* can also be a data object in R (data object must be igraph, graphNEL, matrix or data.frame class). For ”matrix” type, *inputData* should be a directory containing a file name with extension ”cct” or ”cbt” or a matrix or data.frame object in R. The detail information of ”cct” or ”cbt” format can be found in the manual of NetGestalt (www.netgestalt.org). The data should have column names. If the data do not have any row name, the first column of the data should be the ids. For ”sbt” type, *inputData* should be a directory containing a file name with extension ”sbt” . The detail information of ”sbt” format can be found in the manual of NetGestalt (www.netgestalt.org). For ”sct” type, *inputData* should be a directory containing a file name with extension ”sct” . The detail information of ”sct” format can be found in the manual of NetGestalt (www.netgestalt.org).

2. *inputType*: the type of the input data. see detailed information in *inputData* parameter.

3. *idType*: see *idType* in *NetSAM* function.

4. *collapse_mode* is the method to collapse duplicate ids. See details in *mergeDuplicate* section. For SCT file, we suggest to use ”max” or ”min” to collapse duplicate ids in the statistic data.

5. *is_outputFile*: If *is_outputFile* is TRUE, the function will output the transformed data to a file. The default is FALSE.

6. *outputFileName*: the output file name.

7. *verbose* is to identify whether the function report the extra information on progress. The default is TRUE.

6.2 Output

The *mapToSymbol* function will output a object with transformed data. If the ids in the input data can not be transformed to gene symbols, the function will output NULL. If *outputFileName* is TRUE, the function will output the transformed data to a file.

7 Construction of correlation network

The *MatNet* function can be used to construct a correlation network based on the input matrix.

```
> library("NetSAM")
> inputMatDir <- system.file("extdata","exampleExpressionData.cct",package="NetSAM")
> matNetwork <- MatNet(inputMat=inputMatDir, collapse_mode="maxSD", naPer=0.7, meanPer=0.8, varPer=0.
+ corrType="spearman", matNetMethod="rank", valueThr=0.6, rankBest=0.003, networkType="signed",
+ netFDRMethod="BH", netFDRThr=0.05, idNumThr=(-1), nThreads=3)
```

```
The input data contain 3899 ids and 90 samples. No missing values in the data.
After removing ids based on parameters 'naPer', 'meanPer' and 'varPer', 2495 ids are remained!
Allowing parallel execution with up to 7 working processes.
Calculating spearman correlation for each pair of ids...
Base on rank-based method, an undirected network with 1993 nodes and 3048 edges was identified when s
```

7.1 Input

This section describes the arguments of the *MatNet* function:

1. *inputMat* should be a directory containing a file name with extension ”cct” or a matrix or data.frame object in R. The detail information of ”cct” format can be found in the manual of NetGestalt (www.netgestalt.org). The data should have column names. If the data do not have any row name, the first column of the data should be the ids.

2. If the input matrix data contains the duplicate ids, the function will collapse duplicate ids based on the emphcollapse_mode. ”mean”, ”median”, ”maxSD” and ”maxIQR” represent the mean, median, max standard deviation or max interquartile range of id values in each sample. The default is ”maxSD”.

3. To remove ids with missing values in most of samples, the function calculates the percentage of missing values in all samples for each id and removes ids with over *naPer* missing values in all samples. The default *naPer* is 0.7.

4. To remove ids with low values, the function calculates the mean of values for each id in all samples and remains top *meanPer* ids based on the mean. The default *meanPer* is 0.8.

5. Based on the remained ids filtered by *meanPer*, the function can also remove less variable ids by calculating the standard deviation of values for each id in all samples and remaining top *varPer* ids based on the standard deviation. The default *varPer* is 0.8.

6. *corrType* is a character string indicating which correlation coefficient is to be computed for each pair of ids. The function supports ”spearman” (default) or ”pearson” method.

7. *MatNet* function supports three methods to construct correlation network: ”value”, ”rank” and ”directed”. 1. ”value” method: the correlation network only remains id pairs with correlations over cutoff threshold *valueThr*; 2. ”rank” method: for each id A, the function first selects ids that significantly correlate with id A and then extracts a set of candidate neighbors (the number of ids is *rankBest*) from the significant set that are most similar to id A. Then, for each id B in the candidate neighbors of id A, the function also extracts the same number of ids that are significant correlated and most similar to id B. If id A is also the candidate neighbors of id B, there will be an edge between id A and id B. Combining all edges can construct a correlation network; 3. ”directed” method: the function will only remain the best significant id for each id as the edge.Combining all edges can construct a directed correlation network.

8. *valueThr* is the correlation cutoff threshold for ”value” method.

9. *rankBest* is the percentage of ids that are most similar to one id for ”rank” method. The default is 0.003 that means the ”rank” method will select top 30 most similar ids for each id if the number of ids in the matrix is 10,000.

10. If *networkType* is ”unsigned”, the correlation of all pairs of i ids will be changed to absolute values. The default is ”signed”.

11. *netFDRMethod* is the p value adjustment methods for ”rank” and ”directed” methods. The default is ”BH”.

12. *netFDRThr* is fdr threshold for identifying significant pairs for ”rank” and ”directed” methods.

13. *idNumThr*: If the matrix contains too many ids, it will take a long time and use a lot of memory to identify the modules. Thus, the function provides the option to set the threshold of number of ids for further analysis. After filtering by meanPer and varPer, if the number of ids is still larger than *idNumThr*, the function will select top *idNumThr* ids with the largest variance. The default is -1, which means there is no limitation for the matrix.

7.2 Output

The *MatNet* function will output a matrix with two columns.

8 Test input data format

The *testFileFormat* function will test the format of the input data matrix and annotation data and return the standardized data matrix and sample annotation data.

```
> library("NetSAM")
> inputMatDir <- system.file("extdata","exampleExpressionData.cct",package="NetSAM")
> sampleAnnDir <- system.file("extdata","sampleAnnotation.tsi",package="NetSAM")
> formatedData <- testFileFormat(inputMat=inputMatDir,sampleAnn=sampleAnnDir,collapse_mode="maxSD")
```

8.1 Input

This section describes the arguments of the *testFileFormat* function:

1. *inputMat* is a file name or a matrix or data.frame object in R.

2. *sampleAnn* is a file name or a data.frame object in R.

If the users set inputMat as "", the testFileFormat function only test format of sample annotation data. If the users set sampleAnn as "", the testFileFormat function only test format of data matrix.

3. If the input data contains the duplicate ids, the function will collapse duplicate ids based on the *collapse_mode*. "mean", "median", "maxSD" and "maxIQR" represent the mean, median, max standard deviation or max interquartile range of values for ids in each sample. The default is "maxSD".

8.2 Output

If there is no format error, the function will return the standardized data matrix and sample annotation data. Otherwise, the function will output the detailed position of the errors.

9 Identification of the associations between sample features and modules

The *featureAssociation* function can be used to calculate the associations between sample features in the input sample annotation data and the modules identified by *NetSAM* or *MatSAM* functions.

```
> library("NetSAM")
> inputMatDir <- system.file("extdata","exampleExpressionData.cct",package="NetSAM")
> sampleAnnDir <- system.file("extdata","sampleAnnotation.tsi",package="NetSAM")
> data(NetSAMOutput_Example)
> outputHtmlFile <- paste(getwd(),"/featureAsso_HTML",sep="")
> featureAsso <- featureAssociation(inputMat=inputMatDir, sampleAnn=sampleAnnDir, NetSAMOutput=netsa
```

```
The assocaitaions between modules and features can be found at /private/var/folders/7r/3czb3xbj4b59zw
```

9.1 Input

This section describes the arguments of the *featureAssociation* function:

1. *inputMat* should be a directory containing a file name with extension "cct" or a matrix or data.frame object in R. The detail information of "cct" format can be found in the manual of NetGestalt (www.netgestalt.org). The data should have column names. If the data do not have any row name, the first column of the data should be the ids.

2. *sampleAnn* should be a directory containing a file name with "tsi" extension or a data.frame object in R. The detail information of "tsi" format can be found in the manual of NetGestalt (www.netgestalt.org). The first row of the sample annotation data is the feature names. The second row is the feature types. The function supports four types: BIN (binary feature, such as male and female), CAT (category feature, such as stage i, stage ii and stage iii), CON (continuous feature, such as age) and SUR (survival data, such as overall survival). The third row is the feature categories. If there is no category information for the features, the sample information will start from the third row . The first column is the sample names.

3. *NetSAMOutput* is the list object outputted from the *NetSAM* or *MatSAM* functions.

4. *outputHtmlFile* is the output directory of the HTML file.

5. *CONMethod* is the method to calculate the associations between modules and continuous features. The function provides two methods: "spearman" and "pearson" and the default is "spearman".

6. *CATMethod* is the method to calculate the associations between modules and category features. The function provides two methods: "anova" and "kruskal" and the default is "kruskal".

7. *BINMethod* is the method to calculate the associations between modules and binary features. The function provides two methods: "test" and "rankest" and the default is "rankest".

8. *fdrmethod* is the FDR method for identifying the significantly associated GO terms. The default is "BH".

9. *pth* is the threshold of the p values to identify the significant associations.

10. If the input matrix data contains the duplicate ids, the function will collapse duplicate ids based on the *collapse_mode*. "mean", "median", "maxSD" and "maxIQR" represent the mean, median, max standard deviation or max interquartile range of id values in each sample. The default is "maxSD".

9.2 Output

The function will output a data.frame object and a HTML file to show the significant associations.

10 Identification of the associated GO terms for the modules

The *GOAssociation* function can be used to identify the associated GO terms for the modules identified by *NetSAM* or *MatSAM* functions.

```
> library("NetSAM")
> data(NetSAMOutput_Example)
> outputHtmlFile <- paste(getwd(),"/GOAsso_HTML",sep="")
> GOAsso <- GOAssociation(NetSAMOutput=netsam_output, outputHtmlFile=outputHtmlFile, organism="hsapie
```

```
The associated GO terms for each module can be found at /private/var/folders/7r/3czb3xbj4b59zwhxtzh_0
```

10.1 Input

This section describes the arguments of the *GOAssociation* function:

1. *NetSAMOutput* is the list object outputted from the *NetSAM* or *MatSAM* functions.

2. *outputHtmlFile* is the output directory of the HTML file.

3. *organism* is the organism of the input data matrix that has been used to identify the modules. Currently, the package supports the following nine organisms: hsapiens, mmusculus, rnorvegicus, drerio, celegans, scerevisiae, cfamiliaris, dmelanogaster and athaliana. The default is "hsapiens".

4. *outputType* is the type for the output associated GO terms . The function supports two types of output results. 1. "significant" represents all associated GO terms should be significant under a certain FDR threshold; 2. "top" represents the function first sorts all GO terms based on their hypergeometric test p values and then selects top GO terms as the associated terms. The default is "significant'.

5. *fdrmethod* is the FDR method for identifying the significantly associated GO terms. The default is "BH".

6. *fdrth* is the FDR threshold.

7. *topNum* is the number of the selected top GO terms.

10.2 Output

The function will output a data.frame object and a HTML file to show the associated GO terms for each module.

11 Identification of correlation modules

The *MatSAM* function can identify the hierarchical correlation modules by calling *MatToSymbol*, *MatNet* and *NetSAM* functions.

```
> library("NetSAM")
> inputMatDir <- system.file("extdata","exampleExpressionData.cct",package="NetSAM")
> sampleAnnDir <- system.file("extdata","sampleAnnotation.tsi",package="NetSAM")
> outputFileName <- paste(getwd(),"/MatSAM",sep="")
> matModule <- MatSAM(inputMat=inputMatDir, sampleAnn=sampleAnnDir, outputFileName=outputFileName, ou
+ map_to_symbol=FALSE, idType="auto", collapse_mode="maxSD", naPer=0.7, meanPer=0.8, varPer=0.8, corr
+ valueThr=0.6, rankBest=0.003, networkType="signed", netFDRMethod="BH", netFDRThr=0.05, minModule=0.
+ maxStep=4, moduleSigMethod="cutoff", modularityThr=0.2, ZRanNum=10, PerRanNum=100, ranSig=0.05, idN

The input data contain 3899 ids and 90 samples. No missing values in the data.
After removing ids based on parameters 'naPer', 'meanPer' and 'varPer', 2495 ids are remained!
Allowing parallel execution with up to 7 working processes.
Calculating spearman correlation for each pair of ids...
Base on rank-based method, an undirected network with 1993 nodes and 3048 edges was identified when s
Allowing parallel execution with up to 7 working processes.

Identifying the hierarchical modules of the network...
Starting to analysis connected component 1!
Evaluating networks in Level 1 ...
Network modularity: 0.8002245

Evaluating networks in Level 2 ...
Modularity of network 1: 0.7213026

Modularity of network 2: 0.5231162

Modularity of network 3: 0.566358

Modularity of network 4: 0.6766937

Modularity of network 5: 0.505102

Modularity of network 6: 0.4201184

Modularity of network 7: 0.3163266

Modularity of network 8: 0.465374

Modularity of network 9: 0.5625

Modularity of network 10: 0.2956104

Modularity of network 11: 0.6538086

Modularity of network 12: 0.4965277

Modularity of network 13: 0.4756944

Modularity of network 14: 0.6963683

Modularity of network 15: 0.5321181

Modularity of network 16: 0.5478317

Modularity of network 17: 0.6276706

Modularity of network 18: 0.5371094

Modularity of network 19: 0.355
Modularity of network 20: 0.4669422

Modularity of network 21: 0.3417609

Modularity of network 22: 0.4002268

Modularity of network 23: 0.2205679

Modularity of network 24: 0.4244445

Modularity of network 25: 0.4387755

Modularity of network 26: 0.3571429

Modularity of network 27: 0.5117188

Modularity of network 28: 0.3671875

Modularity of network 29: 0.4584775

Modularity of network 30: 0.3571429

Modularity of network 31: 0.3571429

Modularity of network 32: 0.2578125

Modularity of network 33: 0.4012346

Modularity of network 34: 0.5040171

Modularity of network 35: 0.4171598

Modularity of network 36: 0.4822485

Modularity of network 37: 0.06568879

Modularity of network 38: 0.26

Modularity of network 39: 0.3

Modularity of network 40: 0.5006511

Modularity of network 41: 0.26

Modularity of network 42: 0.26

Modularity of network 43: 0.165

Modularity of network 44: 0.4380165

Modularity of network 45: 0.3571429

Modularity of network 46: 0.26

Modularity of network 47: 0.2109375

Modularity of network 48: 0.26
Modularity of network 49: 0.2083333

Modularity of network 50: 0.2916667

Modularity of network 51: 0.26

Modularity of network 52: 0.3671875

Modularity of network 53: 0.4467456

Modularity of network 54: 0.3571429

Modularity of network 55: 0.2916667

Modularity of network 56: 0.3571429

Modularity of network 57: 0.3264463

Modularity of network 58: 0.3671875

Modularity of network 59: 0.2577161

Modularity of network 60: 0.3644444

Modularity of network 61: 0.3828125

Modularity of network 62: 0.4723184

Modularity of network 63: 0.2844445

Modularity of network 64: 0.26

Modularity of network 65: 0.3571429

Modularity of network 66: 0.2301587

Modularity of network 67: 0.3571429

Modularity of network 68: 0.2901234

Modularity of network 69: 0.1938775

Modularity of network 70: 0.3194444

Modularity of network 71: 0.2083333

Modularity of network 72: 0.2959184

Modularity of network 73: 0.3194444

Modularity of network 74: 0.3194444

Modularity of network 75: 0.3

Modularity of network 76: 0.1938775

Modularity of network 77: 0.3194444

Modularity of network 78: 0.1122449
Modularity of network 79: 0.15625

Modularity of network 80: 0.1694215

Modularity of network 81: 0.06720001

Modularity of network 82: 0.1234568

Modularity of network 83: 0.2541091

Modularity of network 84: 0.0925926

Evaluating networks in Level 3 ...
Modularity of network 1: 0.2916667

Modularity of network 2: 0.3222656

Modularity of network 3: 0.4157986

Modularity of network 4: 0.4158951

Modularity of network 5: 0.2142857

Modularity of network 6: 0.1631944

Modularity of network 7: 0.002835566

Modularity of network 8: 0

Modularity of network 9: 0

Modularity of network 10: 0.1770833

Modularity of network 11: 0.125

Modularity of network 12: 0.1626276

Modularity of network 13: 0.3298611

Modularity of network 14: 0.5918913

Modularity of network 15: 0.395

Modularity of network 16: 0.5145682

Modularity of network 17: 0.5367461

Modularity of network 18: 0.26

Modularity of network 19: 0.2530864

Modularity of network 20: 0.3111111

Modularity of network 21: 0.3535156

Modularity of network 22: 0.26

Modularity of network 23: 0.2361111
Modularity of network 24: 0.2603306

Modularity of network 25: 0.2407848

Modularity of network 26: 0.316609

Modularity of network 27: 0.125

Modularity of network 28: 0.3194444

Modularity of network 29: 0.1157025

Modularity of network 30: 0.0798611

Modularity of network 31: 0.3061225

Modularity of network 32: 0

Modularity of network 33: 0.26

Modularity of network 34: 0.26

Modularity of network 35: 0.396605

Modularity of network 36: 0.5149177

Modularity of network 37: 0.345

Modularity of network 38: 0.271605

Modularity of network 39: 0.1666667

Modularity of network 40: 0.25

Modularity of network 41: 0.2831633

Modularity of network 42: 0.09375

Modularity of network 43: 0.1666667

Modularity of network 44: 0.0007999986

Modularity of network 45: 0.1796875

Modularity of network 46: 0.1656805

Modularity of network 47: 0.2947846

Modularity of network 48: 0.1088

Modularity of network 49: 0.1371191

Modularity of network 50: 0.1471894

Modularity of network 51: 0.3

Modularity of network 52: 0
Modularity of network 53: 0.05293007

Modularity of network 54: 0

Modularity of network 55: 0.01785716

Modularity of network 56: 0.1065089

Modularity of network 57: 0.26

Modularity of network 58: 0.271605

Modularity of network 59: 0.1975

Modularity of network 60: 0

Modularity of network 61: 0.01785716

Modularity of network 62: 0.01384084

Evaluating networks in Level 4 ...
Modularity of network 1: 0.125

Modularity of network 2: 0

Modularity of network 3: 0.122449

Modularity of network 4: 0.125

Modularity of network 5: 0.2083333

Modularity of network 6: 0.271605

Modularity of network 7: 0.2911111

Modularity of network 8: 0.04938272

Modularity of network 9: 0.2142857

Modularity of network 10: 0.2048611

Modularity of network 11: 0.125

Modularity of network 12: 0.26

Modularity of network 13: 0.2040816

Modularity of network 14: 0.2167969

Modularity of network 15: 0.2083333

Modularity of network 16: 0.2395833

Modularity of network 17: 0.3016

Modularity of network 18: 0.1666667

Modularity of network 19: 0
Modularity of network 20: 0

Modularity of network 21: 0.2421875

Modularity of network 22: 0

Modularity of network 23: 0.04492188


Starting to analysis connected component 2!
Evaluating networks in Level 1 ...
Network modularity: 0.347699

Evaluating networks in Level 2 ...
Modularity of network 1: 0

Modularity of network 2: 0


Starting to analysis connected component 3!
Evaluating networks in Level 1 ...
Network modularity: 0.4259259


Starting to analysis connected component 4!
Evaluating networks in Level 1 ...
Network modularity: 0.3


Starting to analysis connected component 5!
Evaluating networks in Level 1 ...
Network modularity: 0.3


Starting to analysis connected component 6!
Evaluating networks in Level 1 ...
Network modularity: 0



Reordering the genes in the one dimentional layout...
NetSAM identified 177 modules in 4 levels!
Processing completed!

Calculate the associations between modules and annotations...
The assocaitaions between modules and features can be found at /private/var/folders/7r/3czb3xbj4b59zw
Identify the associated GO Terms for each module...
The associated GO terms for each module can be found at /private/var/folders/7r/3czb3xbj4b59zwhxtzh_(
```

11.1 Input

This section describes the arguments of the *MatSAM* function:

1. *inputMat* should contain a file name with extension ”cct” (the detail information of ”cct” format can be found in the manual of NetGestalt (www.netgestalt.org)) or a matrix or data.frame object in R. The data should have column names. If the data do not have row names, the first column of the data should be the ids.

2. *sampleAnn* should contain a file name with ”tsi” extension (the detail information of ”tsi” format can be

found in the manual of NetGestalt (www.netgestalt.org)) or a data.frame object in R. If the data does not have sample annotation, this argument can be ignored. The first row of the data is the name of sample features. The second row is the type of each feature. The third row is the category of each feature. If there is no category information for the features, the sample information will start from the third row . The first column is the sample name.

3. *outputFormat* is the format of the output file. ”msm” format can be used as an input in NetGestalt; ”gmt” format can be used to do other network analysis (e.g. as an input in GSEA (Gene Set Enrichment Analysis) to do module enrichment analysis); ”multiple” represents the MatSAM function will output five files: ruler file containing gene order information, hmi file containing module information, net file containing correlation network information, cct file containing the filtered data matrix, and tsi file containing the sample annotation with standardized format; and ”none” represents the function will not output any file.

4. If *map_to_symbol* is TRUE, the function will first change the input id to gene symbol and collapse multiple ids with the same gene symbol based on the *collapse_mode* method before identifying correlation network. The default is FALSE.

5. *matNetMethod* is the method to construct correlation network, which has two options ”value” and ”rank”. Details can be found in the argument description of the *MatNet* function.

The description of other arguments can be found in the argument description of the above functions.

11.2 Output

Including a ”msm” file or other type of file (see *outputFormat*), the function will output a list object containing module information, gene order information, correlation network and filtered matrix based on the ids in the network. The function will also output two HTML files that contain the significant associations between sample features and modules identified from *featureAssociation* function and associated GO terms for the modules identified from *GOAssociation* function.

Note: When calling the *featureAssociation* function, MatSAM only uses the default parameters. When calling the *GOAssociation* function, MatSAM sets ”ouputType” as ”top” and ”topNum” as 1. The users can use the list object returned by MatSAM as the input to these two functions to perform some further analysis based on the different parameters.