NetRankR R Package User Guide

alexander.shmelkin@tu-dresden.de

June 12, 2022

Contents

1 Introduction **1**

2 Functions **2**
2.1 RunNetRank() 2
2.2 fetchStringDB() 3
2.3 CoexpressionNetwork_Preprocessing() 4
2.4 CoexpressionNetwork_BuildNetwork() 4
2.5 CoexpressionNetwork_Postprocessing() 5
2.6 CoexpressionNetwork_Clusteranalysis() 6
2.7 SignatureClusters() 7

3 Workflow Examples **7**
3.1 Preparation 7
3.2 First Workflow: Gene Co-Expression Network 9
3.2.1 Preprocessing 9
3.2.2 Constructing the Weighted Gene Co-Expression Network . . . 9
3.2.3 Extracting Cluster Information from the Network 10
3.2.4 Postprocessing 10
3.2.5 Applying NetRank 11
3.2.6 Interpret Results 12
3.3 Second Workflow: Protein-Protein Interaction Network 12
3.3.1 Fetching the Protein-Protein Interaction Network 12
3.3.2 Applying NetRank 13

1 Introduction

This is a user guide for the R package "NetRankR". The package provides tools for creating a protein-protein interaction network or a gene co-expression network from gene expression data and applying the NetRank algorithm on those networks. Additionally, the package contains functionality for identifying clusters and hub genes in gene co-expression networks, as well as the 10 most relevant Gene Ontology terms for each cluster.

2 Functions

The current version of the R package contains 7 functions:

- RunNetRank()
- fetchStringDB()
- CoexpressionNetwork_Preprocessing()
- CoexpressionNetwork_BuildNetwork()
- CoexpressionNetwork_Postprocessing()
- CoexpressionNetwork_Clusteranalysis()
- SignatureClusters()

2.1 RunNetRank()

This function applies the NetRank algorithm to a previously computed genetic interaction network, which can either be a protein-protein interaction network (fetched from StringDB) or a gene co-expression network (computed from gene co-expression data using the WGCNA package). NetRank updates are computed using the formula

$$r_j^n = (1-d)s_j + \max(d\sum_{i=1}^{N}\frac{m_{ij}r_i^{n-1}}{\text{degree}_i},\ 1),\ 1 \leq j \leq N$$

with
r: the node (gene) ranking score
n: iteration
j: index of the current node
d: damping factor (ranging between 0 and 1)
s: Pearson correlation coefficients
degree: the sum of the output connectivities for connected nodes
N: number of the total nodes
m: connectivity of connected nodes, $m_{ij} = 1$, if i and j are connected and 0 otherwise.

For more information see this publication.

This function requires a previoulsy build network from either fetchStringDB() or CoexpressionNetwork_BuildNetwork().

The parameters for this function are:

- **network**: Specify between "StringDB" or "Coexpression" to decide on which (previously constructed) network NetRank will be applied.
- **datasetDir**: Name of the dataset directory, the dataset directory has to be directly within the working directory (workingDir).

- **workingDir**: Name of the working directory .
- **d**: Dampening factor between 0 and 1. This factor weights the two parts of the NetRank formula against each other (node correlation vs. node connectivity). A d value close to 0 increases the importance of node correlation, a value close to 1 increases the importance of node connectivity. Default = 0.5.
- **cores**: Number of CPU cores used for the computation. An increased number of cores significantly speeds up computation time.
- **minError**: Threshold for finishing the main Netrank computation as soon as the root mean square deviation between two consecutive iterations is below minError. Default = 0.01.
- **dataFolder**: Name of the directory where the NetRank results will be stored, must be directly within workingDir/datasetDir/.
- **NetWorkCutThreshold**: Must be the value previously chosen threshold for cutting edges with low coexpression values when using the function CoexpressionNetwork_Postprocessing(). Only relevant if network == "Coexpression".

2.2 fetchStringDB()

This function takes previously computed correlation values for each gene towards a phenotype, fetches information from STRING DB and creates a protein-protein interaction network for NetRank.

A file containing each genes correlation with specified phenotype is required. One such file is created during CoexpressionNetwork_Clusteranalysis() and can be used here.

The arguments for this function are:

- **datasetDir**: As before.
- **workingDir**: As before.
- **standardCorr_column**: Name of the column containing the gene correlation information towards a phenotype. This column has to be in the file "corrFilename" and has to be provided by the user. Default = "PearsonCorrelation".
- **genename_col**: Name of the column containing the gene names in the correlation file. This column has to be in the file "corrFilename" and has to be provided by the user. Default = "GeneName".
- **dataFolder**: Name of the directory where the correlation file ("corrFilename") is stored, must be directly in workingDir/datasetDir/.
- **corrfilename**: Name of the file containing the correlation information of each gene towards a phenotype. The file must be provided by the user and stored within workingDir/datasetDir/dataFolder/.
- **species**: StringDB species id. Default = 9606 for human.

2.3 CoexpressionNetwork_Preprocessing()

This function is applied to preprocessed and normalized gene expression data, as well as the phenotype data. This step ensures data quality by removing duplicates, samples without labels, and samples with too many missing entries by utilizing the WGCNA function "goodSamplesGenes". Additionally, in this step, a PCA and hierarchical clustering is performed to plot outliers to allow the user to choose a cut height for culling outliers, as well as plots provided to help the user choose a power threshold for gene co-expression construction that results in a scale-free network.

This function requires preprocessed and normalized gene expression data, as well as the corresponding phenotype data. Example files are provided alongside the source code on GitHub.

The arguments for this function are:

- **networkType**: Type of network. It can either be "none", "unsigned", "signed", "signed Nowick", "unsigned 2", "signed 2", and "signed Nowick 2". "Unsigned" treats negative and positive correlation as absolute values while "signed" keeps the sign. For further information see the documentation in this link. Default = "unsigned".
- **datasetDir**: As before.
- **workingDir**: As before.
- **pathToExprData**: Full path to the gene expression data file.
- **pathToTraitData**: Full path to the phenotype data file.
- **sampleIDCol**: Column name of the column containing the sample ids in the phenotype data file. Default = 'sampleName'.
- **phenoCol**: Column name of the column containing the phenotype id in the phenotype data file. Default = 'pheno'.

2.4 CoexpressionNetwork_BuildNetwork()

This function creates a weighted gene co-expression network using the WGCNA package. It cuts outliers above a user-defined cut height and builds a weighted gene co-expression network out of the data using the WGCNA function "blockwiseModules". In the resulting network the edge value a_{ij} between two nodes i and j is computed using the formula $a_{ij} = s_{ij}^{\beta}$, with β being the soft thresholding power chosen by the user with the parameter "power_thre" and s_{ij} being the co-expression similarity as the absolute value of the correlation coefficient between the expression profiles of nodes i and j.

This function requires "CoexpressionNetwork_Preprocessing()" to have previously been called.

The arguments for this function are:

- **networkType**: Use the same networkType as used during CoexpressionNetwork_Preprocessing.

- **power_thre**: Power threshold for soft thresholding during network construction. For more information see the WGCNA publication.
- **maxBlockSize**: Maximum block size. The current version of the package supports only the creation of one TOM-Block with "CoexpressionNetwork BuildNetwork". If more than one TOM-Block is created by the function call, the user should increase the parameter "maxBlockSize" and rerun the function, it is recommended to have maxBlockSize set to around 20000 on a workstation with 16GB RAM or around 30000 on a 32GB workstation, large datasets require a higher value for maxBlockSize. If more than one TOM-block is created then the maxBlockSize should be increased and the computation repeated.
- **cutheight**: Cut height for cutting outliers. The plots created using CoexpressionNetwork_Preprocessing() help for deciding on a cut height. For more information see the WGCNA publication.
- **workingDir**: As before.
- **datasetDir**: As before.
- **cores**: Number of CPU cores used for the computation. An increased number of cores significantly speeds up computation time.

2.5 CoexpressionNetwork_Postprocessing()

This function takes the previously computed coexpression network, as well as a file containing phenotype correlation values and prepares it in the format needed for applying NetRank to it. It is also worth mentioning that this function requires a lot of memory, therefore a workstation with at least 32GB RAM is recommended.

This function requires "CoexpressionNetwork_BuildNetwork()" to have previously been called and a file containing each genes correlation with specified phenotype. One such file is created during CoexpressionNetwork_Clusteranalysis() and can be used here.

The arguments for this function are:

- **datasetDir**: As before.
- **workingDir**: As before.
- **networkType**: Must use same as before.
- **standardCorr_column**: Name of the column containing the gene correlation information towards a phenotype. This column has to be in the file "corrFilename" and has to be provided by the user. Default = "PearsonCorrelation".
- **genename_col**: Name of the column containing the gene names in the correlation file. This column has to be in the file "corrFilename" and has to be provided by the user. Default = "GeneName".

- **dataFolder**: Name of the directory where the correlation file ("corrFilename") is stored, must be directly in workingDir/datasetDir/.
- **NetWorkCutThreshold**: The parameter "NetWorkCutThreshold" servers as a threshold, removing all edges smaller or equal to the threshold from the network. This is crucial for controlling the size of the network, as otherwise the number of edges and therefore the memory consumption would be far too large to handle. The choice of the threshold highly depends on the user's intention and the underlying data, as well as the choice of the parameter "power_thre" for network construction, therefore no value can be recommended and the user is encouraged to find one that best fits his needs and the data used.
- **corrFilename**: Name of the file containing the correlation information of each gene towards a phenotype. The file must be provided by the user and stored within workingDir/datasetDir/dataFolder/.

2.6 CoexpressionNetwork_Clusteranalysis()

This function takes the previously computed gene co-expression network and outputs a list of hub genes ("HubGenes.csv"), a list containing the cluster membership for each gene ("GeneModules.csv"), a list containing detailed information regarding the most common Gene Ontology terms for each cluster ("GOEnrichmentTable.csv"), and a list of each gene's correlation towards the specific phenotype ("PhenoCorr.csv"). This information is computed using the WGCNA R package. Hereby the clusters are determined by first pre-clustering nodes into large clusters, referred to as blocks, using a variant of k-means clustering, afterwards hierarchical clustering is applied to each block, and clusters are defined as branches of the resulting dendrogram. Hub genes are defined as the most connected gene within each cluster. The most common GO terms are computed for each cluster by calculating all enrichments in the specified ontologies and collecting information about the terms with the highest enrichment. The enrichment p-value is calculated using Fisher's exact test. Please note that this function uses organism-specific annotation and is set to human, to analyze data from a different organism, the user should change the organism parameter within the source code and install the corresponding annotation package. The weighted Pearson correlation of each gene towards the phenotype can be used as the correlation file for NetRank initialization by the functions "fetchStringDB" and "CoexpressionNetwork_Postprocessing".

This function requires "CoexpressionNetwork_BuildNetwork()" to have previously been called.

The arguments for this function are:

- **datasetDir**: As before.
- **workingDir**: As before.
- **networkType**: As before.
- **power_thre**: As before.

- **numOfGOterms**: Number of best fitting GO terms for each cluster to be saved, default = 10.

2.7 SignatureClusters()

Here the user can input a ranked list of genes, i.e., NetRank results, a p-value threshold, and the desired number of genes. The function then outputs a gene signature with the desired number of genes ranked highest, that have a phenotype correlation p-value below the chosen threshold. Additionally, SignatureClusters outputs the detailed cluster information for the signature ("signature_with_clusters.csv"), the hub genes found in the signature ("signature_hubs.csv"), as well as all the clusters found in the signature, their total cluster size, and the number of genes from the signature in each cluster ("signatureclusters.csv"). This helps the user to interpret the results.

This function requires a ranked list of genes and "CoexpressionNetwork_Clusteranalysis" to have previously been called.

The arguments for this function are

- **p_threshold**: P-value threshold for the gene signature. Genes with a p-value greater or equal to p_treshold are discarded.
- **pathToSignatureFile**: Path to the file that contains a ranked list of genes, i.e. the NetRank output file.
- **workingDir**: As before.
- **datasetDir**: As before.
- **rankCol**: Name of the column cintaining the gene ranking.
- **networkType**: As before.

3 Workflow Examples

In this section we go through both possible workflows using the example data uploaded alongside the source code on GitHub.

3.1 Preparation

We start the gene expression analysis by installing the R package using "devtools", the two possible methods are installing it directly from GitHub with

```
devtools::install_github('BiotecNetRank/NetRankR')
```

or by downloading the package as a .zip and using

```
devtools::install_local('path/to/package.zip')
```

Now that we have installed the package, we make sure to have the directory structure that we need for applying our package. First, we need to define our working directory, in this guide's example case it is "D:\\ExampleProject\\" for Windows. Within our working directory, we need a directory containing the dataset files, as we use the NCBI dataset GSE11121 we name the directory "GSE11121". Within this dataset directory, we put our already preprocessed expression and phenotype data. The files used in this example are uploaded on GitHub alongside the source code. We named the expression file "ExpressionData" and the phenotype file "PhenoData".

The gene expression data must have the genes as gene symbols as column names and the sample ids as the first row as shown below.

```
        X     MELK     UBE2C    CCNB2    KIF2C    LMNB1   TRIP13    ...
1 GSM282373 7.752413 10.025228 7.505244 8.094909 6.832414 7.726518  ...
2 GSM282374 6.723572  8.096600 6.704696 6.720870 5.542061 6.706176  ...
3 GSM282375 6.691311  8.380379 6.542921 6.401457 5.725244 6.273672  ...
4 GSM282376 7.009169  8.045753 6.238003 6.673168 6.077715 6.615858  ...
5 GSM282377 7.075874  8.709428 7.257200 7.042989 5.976952 6.915200  ...
...
```

The phenotype data must at least have two columns, one for the sample ids and one for the phenotypes. The column names can be chosen freely as they are given to the function as parameters. The first 5 rows of our example file are shown below.

```
  sampleName pheno
1  GSM282373     2
2  GSM282374     3
3  GSM282375     3
4  GSM282376     2
5  GSM282377     2
...
```

Within the directory GSE11121, we need another directory where we store our gene correlation file. In this example, the directory is called "GeneData" and the correlation file is called "stdCorrResults_GSE11121_stdcor.csv". The file must have one column containing the gene names and one column containing the correlation values. One such file is created in a later step during the workflow which we could also use instead if we didn't already have one. Those are the first 5 rows of our correlation file:

```
  GeneName PearsonCorrelation   p.Standard   q.Standard
1     CCNB2          0.5703789 9.346538e-20 5.122458e-16
2     FOXM1          0.5567175 1.193699e-18 3.271089e-15
3     UBE2C          0.5534896 2.137126e-18 3.904241e-15
4     KIF2C          0.5333113 6.950435e-17 7.188277e-14
5      MELK          0.5329985 7.320681e-17 7.188277e-14
...
```

Now let's load the package:

```
library(NetRankR)
```

This concludes the preparation.

3.2 First Workflow: Gene Co-Expression Network

3.2.1 Preprocessing

Now that we have all the required files, we can start with the first step of our workflow for constructing a gene co-expression network. We call the function

```
CoexpressionNetwork_Preprocessing(
networkType = 'unsigned',
datasetDir = 'GSE11121',
workingDir = 'D:\\ExampleProject\\',
pathToExprData = 'D:\\ExampleProject\\GSE11121\\ExpressionData.csv',
pathToTraitData = 'D:\\ExampleProject\\GSE11121\\PhenoData.csv',
sampleIDCol = 'sampleName',
phenoCol = 'pheno').
```

This function call created within our dataset directory a new directory named "unsigned" and within that a new directory named "Plots". Here we find 3 plots to detect outliers. "1_sampleClustering.pdf" and "1_sampleClustering_pheno.pdf" were clustered using the build-in R base function "hclust" for hierarchical clustering with "method" set on "average", the first having the sample ids as labels while the latter has the phenotype as labels." 1_sampleClustering_complete.pdf" has the method set on "complete" for hierarchical clustering. Those files are created to help decide on a cut height for culling outliers. Another two plots are created for PCA analysis, "1_PCA_eigenvalues.pdf" visualizing the PCA eigenvalues, showing the percentage of variances explained by each principal component, and "1_PCA_individuals.pdf" showing the PCA of individuals, groping together individuals with similar profiles. "2_Mean_connectivity.pdf" and "2_scale_independence.pdf" help the user choose a power threshold that will result in a scale-free network. For more details see the WGCNA publication.

3.2.2 Constructing the Weighted Gene Co-Expression Network

In this example we don't want to cull any outliers, so we set cutHeight = 1000 for our next step. For the soft thresholding power β, we choose 1, to not change the connectivity, as it is computed using the formula $a_{ij} = s_{ij}^{\beta}$, with a_{ij} being the resulting edge value between nodes i and j, and s_{ij} being the correlation in the expression profiles between genes i and j.

With this we use the next function, setting the parameter cores to as many cores as our workstation allows, as it greatly increases computation time.

```
CoexpressionNetwork_BuildNetwork (
networkType = 'unsigned',
power_thre = 1,
maxBlockSize = 25000,
cutHeight = 1000,
workingDir = 'D:\\ExampleProject\\',
datasetDir = 'GSE11121',
```

```
cores = 6)
```

Once the program has finished we take a look inside the GSE11121 directory and see a file named "CoexpressionNetwork_TOM-block.1", if more than this one TOM-block has been created, the program should be restarted with an increased value for "maxBlockSize". This is due to the limitation of the current version of NetRankR only supporting one TOM-block. Additionally, we find new plots showcasing the cut and the dataset after removing outliers.

3.2.3 Extracting Cluster Information from the Network

Next, we want to run

```
CoexpressionNetwork_Clusteranalysis(
datasetDir = 'GSE11121',
workingDir = 'D:\\ExampleProject\\',
networkType = 'unsigned',
power_thre = 1,
numOfGOterms = 10).
```

It is important to choose the same parameters as were chosen for building the network. It is important to note that this function uses organism-specific annotation and is set to human, to analyze data from a different organism, the user should change the organism parameter within the code (WGCNA::GOenrichmentAnalysis) and install the corresponding annotation package.

This creates within the 'unsigned' directory four .csv files:

- **GeneModules**: a list of all genes with their assigned clusters/modules (modules are listed both as an id and as a color).
- **GOEnrichmentTable**: a list containing the top 10 (numOfGOterms) Gene Ontology terms for each cluster/module. For a detailed description, see this link.
- **HubGenes**: a list of hub genes.
- **PhenoCorr**: a list of genes with their weighted Pearson correlation towards the phenotype, as well as the p-value for this correlation. This file can be used as the correlation file for NetRank.

3.2.4 Postprocessing

Now we want to post-process the weighted gene co-expression network so that we can apply NetRank to it. We do this by calling

```
CoexpressionNetwork_Postprocessing(
datasetDir = 'GSE11121',
workingDir = 'D:\\ExampleProject\\',
networkType = 'unsigned',
```

```
standardCorr_column = 'PearsonCorrelation',
geneName_col = 'GeneName',
dataFolder = 'GeneData',
NetWorkCutThreshold = 0.15,
corrFilename = 'stdCorrResults_GSE11121_stdcor.csv').
```

It is important to choose the same networkType as before. "standardCorr_column" is the name of the column which contains the correlation for each gene towards the phenotype. "geneName_col" is the name of the column containing the gene names. "dataFolder" is the name of the directory containing the correlation file (it has to be a directory within the dataset directory) and "corrFilename" the name of the correlation file. "NetWorkCutThreshold" servers as a threshold, removing all edges smaller or equal to the threshold from the network. This is crucial for controlling the size of the network, as otherwise the number of edges and therefore the memory consumption would be far too large to handle. The choice of the threshold highly depends on the user's intention and the underlying data, as well as the parameter "power_thre" during network construction, therefore no value can be recommended, and the user is encouraged to find one that best fits his needs and the data used. In this example we choose a NetWorkCutThreshold of 0.15. This step requires a lot of memory, so the workstation should have at least 16GB RAM if no other processes take up much memory, 32GB or more are recommended.

3.2.5 Applying NetRank

Now that we have the data in the right format, we can apply NetRank using

```
RunNetRank(
network = 'Coexpression',
datasetDir = 'GSE11121',
workingDir =  'D:\\ExampleProject\\',
cores = 6,
d = 0.5,
minError = 0.01,
dataFolder = 'GeneData',
NetWorkCutThreshold = 0.15).
```

The parameter "cores" specifies the number of cores used. As NetRank is optimized for parallel processing having many cores greatly increases computation time. "d" is the dampening factor weighting both parts of the NetRank formula, the node connectivity, and the node value, against each other; a value of d = 0.5 weights both parts equally. Having a value closer to 0 increases the impact of each gene's correlation towards the phenotype, while a value closer to 1 increases the impact of each gene's connectivity towards the final ranking. "minError" serves as a threshold to break the iterative update of the Network if the difference between two updates (measured by a root mean square deviation) is smaller or equal to the minError. For "NetWorkCutThreshold" it is important to take the same value used in the previous steps.

After the program has finished, we find within our "GeneData" directory the results in form of a .csv file.

3.2.6 Interpret Results

One last optional step is to use the function

```
SignatureClusters(
p_threshold = 0.05,
numberOfGenes = 50,
pathToSignatureFile = 'D:\\ExampleProject\\GSE11121\\GeneData\\netRankResults_WGCNA
workingDir = 'D:\\ExampleProject\\',
datasetDir = 'GSE11121',
rankCol = 'rank_NetRank_score',
networkType = 'unsigned').
```

This function call takes our previously computed NetRank results ("pathToSignatureFile") and outputs a gene signature consisting of the top 50 ranked genes that have a p-value below 0.05. Additionally, this function saves the detailed cluster information for the signature ("signature_with_clusters.csv"), the hub genes found in the signature ("signature_hubs.csv"), as well as all the clusters found in the signature, their total cluster size and the number of genes from the signature in each cluster ("Signatureclusters.csv"). This information can be used to easily interpret the result from our gene expression data analysis.

This concludes the first workflow.

3.3 Second Workflow: Protein-Protein Interaction Network

The second workflow constructs a protein-protein interaction network and applies NetRank to it, the whole workflow is executed with only two steps.

3.3.1 Fetching the Protein-Protein Interaction Network

We fetch the protein-protein interaction network from StringDB by calling

```
fetchStringDB(
datasetDir = 'D:\\GSE11121\\',
workingDir = 'D:\\ExampleProject\\',
standardCorr_column = 'PeasonCorrelation',
geneName_col = 'GeneName',
dataFolder = 'GeneData',
corrFilename = 'stdCorrResults_GSE11121_stdcor.csv',
species=9606).
```

Most of the parameters are already known from the previous workflow, the new parameter "species" is the species code for StringDB, and 9606 is the code for human.

3.3.2 Applying NetRank

Now we can already apply NetRank with

```
RunNetRank(
network = 'StringDB',
datasetDir = 'GSE11121',
workingDir = 'D:\\ExampleProject\\',
cores = 6,
d = 0.5,
minError = 0.01,
dataFolder = 'GeneData',
NetWorkCutThreshold = 0.15).
```

This saves our results as .csv files within our "GeneData" directory and concludes our example for both workflows.