**4-MAT USER MANUAL**

**Introduction :**

The 4-MAT program (standing for *4-Methods Analyzing Tool / 4-Methods Analysis of Transcriptome*) has been developed with the purpose of finding relevant associations between genes via transcriptomic data.

Initially conceived for associating genes of interest (from now on called *anchor genes*) who play a known role in given metabolic pathways and others *candidate genes* who are not yet known to have a role in those pathways, 4-MAT can also do a broader unfocused analysis and find associations between any couple of genes in a global pool.

**Usage rules :**

4-MAT is written in the Python language. Make sure a Python interpreter (like IDLE) is installed on the computer you're using to run the program as well as the following Python modules : time, collections, sys, copy, NumPy, matplotlib.pyplot, sklearn (scikit-learn), pandas, SciPy, networkx, pickle, warnings.

NOTE : The underlined modules are installed by default with Python IDLE.

When writing the command line to run the program, be sure to put the program file first, the arguments file in second, and then at least one dataset csv file (unless you set the argument file to only run the Consensus Step, in which case no dataset are required).

When using multiple datasets, make sure all of them contains the same genes and the same number of genes (more detailed on the datasets' structure below).

When customizing the settings in the arguments file, make sure that all settings that are number-of-dataset dependent are correctly set according to the number of datasets you are using. Same goes for settings that are number-of-list-of-genes-of-interest dependent.

The program is composed of 4 Analysis Steps plus a Consensus Step. You can customize the argument file to specify which steps you want to run or not. All steps that you'll run will produce one or several result files (detailed below).

All datasets must be structured in the following way :

- The first column must be called 'ID\_REF' and contain the names of all genes to be analyzed. The order doesn't have to be the same in all datasets but all datasets must contain the entirety of the global gene pool, no more no less.

- The next columns can be called whatever you want. They must contain finite float values (no NaN or inf). The number of value columns can change from one dataset to another. As a general rule of thumb, the more columns there are, the more reliable the results will be.

**Initial Data and Settings :**

By giving 4-MAT one or several datasets, each containing transcriptomic values for a same single gene pool, 4-MAT will apply up to 4 different analysis methods to find relevant associations between genes from the pool.

If one or several lists of specific *anchor genes* are given, 4-MAT will create a consensus around those genes and sort all associations between them and all other *candidate genes* found by each method. If no such list of *anchor genes* is provided, 4-MAT will sort all associations between any couple of genes found by each method. The more methods a specific association is found by, the more relevant it ought to be.

The initial settings are the following :

* The **Global gene pool** is the list of all genes whose transcriptomic data are in the datasets you will provide. This list is used twice during the whole running of 4-MAT : once when sorting the anchor genes and once during the P-CEN method.
* The **Anchor gene lists** are the optional lists of specific genes whose associations you want the program to focus on. For each of these lists, 4-MAT will intersect it with the global gene pool and any gene present in the intersection will be labeled as an *anchor*.
* The **Anchor Genes labels** are optional names you can give to the *anchor genes*. By default, the program labels each anchor gene lists as "Anchor-1", "Anchor-2" and so on. If no list of *anchor genes* is provided, all genes will be labeled "Candidate".
* The **Anchor colors** are optional colors for the program to use when building the resulting networks for each methods. By default, all genes, be they anchors or not, will be colored black. These colors are to be given in an RGB format (i.e. sets of 3 values between 0 and 1), and there has to be as many colors as you provided lists of anchor genes. If colors are given, only the *candidate genes* will be colored black.
* The **Step list** is the list of methods you want to activate when launching a 4-MAT run. You can ask for a run with no methods activated by explicitly set this argument to 0.
* The **Consensus step** setting is to indicate if you want a consensus of results to be created.
* The **Linkage dictionary** is the name of a preexisting result dictionary. By default, the program will create a new empty dictionary, initiate it's keys with all focused genes (i.e. *anchor genes* or all genes, depending on the providing of lists of the formers) and it's values with empty sub-dictionaries. For each activated method, the dictionary will fill each key's sub-dictionary value with the code name of the method as the sub-key and the list of associated genes the method has found for this key as the sub-value. If the name of a preexisting dictionary is provided, any activated method will overwrite the previous results found for this method.

The linkage dictionary's general structure looks like this :

* Anchor gene 1 :
  + Method A : [candidate 1.1a , candidate 1.2a , candidate 1.3a , …]
  + Method B : [candidate 1.1b , candidate 1.2b , candidate 1.3b , …]
  + Method C : [candidate 1.1c , candidate 1.2c , candidate 1.3c , …]
  + Method D : [candidate 1.1d , candidate 1.2d , candidate 1.3d , …]
* Anchor gene 2 :
  + Method A : [candidate 2.1a , candidate 2.2a , candidate 2.3a , …]
  + Method B : [candidate 2.1b , candidate 2.2b , candidate 2.3b , …]
  + Method C : [candidate 2.1c , candidate 2.2c , candidate 2.3c , …]
  + Method D : [candidate 2.1d , candidate 2.2d , candidate 2.3d , …]
* Anchor gene 3 : …

NOTE : Some methods give more information than just the list of candidates found for each *anchor gene*. Those information are specified in the description of each method.

**Method 1 – P-CEIN (*Pearson based Co-Expression Network*)**

The *Pearson based CoExpression Intersected Network* method uses the Pearson's Correlation Coefficient (form now on abbreviated PCC) calculation to evaluate the association between genes.

This method's settings are the following :

* The **Anchor centered** boolean indicates if you want the method to focus on the correlation between the anchors and the rest of the global pool (other anchors included) or if you want it to calculate the PCC between all couples of genes from the global pool. WARNING : deactivating this setting makes the P-CEIN method the single longest method to run.
* The **Pearson principal threshold** is an optional threshold used when first evaluating the PCCs. It is by default set to 0.5 (in absolute value) for all datasets and can go up to 1 (perfect correlations only) and down to 0 (all correlations). You can either give a single threshold that will be used for all dataset or give as many thresholds as the number of dataset you provided.
* The **Neighborhood threshold** is an optional value that lightens the network after all couples have been analyzed. If customized to N, it will sort for each gene it's correlations and only keep the N highest. By default, all correlations are kept. **ATTENTION** : This setting is documented here for transparency but in practice it is highly unadvised to use, as it is dependent of the order in which the genes have been put into the network.
* The **Pearson dynamic threshold** boolean is used after the initial evaluation. If activated, the average PCC and standard deviation (respectively abbreviated Avg and Std) of all kept couples will be calculated and a new threshold will be set. Any couple with a PCC lower than this threshold will be filtered out.
* The **Dynamic threshold factor** is the multiplicative value used in the formula to calculate the dynamic threshold. It is by default set to 1.
* The **Minimum neighborhood 1** is an optional setting used to filter out any candidate gene with less than a certain number of anchors linked to it. It is by default set to keep all candidates. If you want to activate this filter, you must give a minimum number of neighbors for each anchor list you have provided.
* The **Research time announcement** and **Global time announcement** are quality of life settings that write on the console respectively the run time for each analyzed gene and the run time for each sub-step of the method.
* The **New Network** boolean indicates if you want to calculate the PCCs of your genes from scratch or if you want to load a preexisting network created by a previous run of the method. Be sure not to change the file's name between it's creation and it's subsequent loading.

The P-CEIN method's execution is as following :

1. It either load a preexisting network or calculates a new one. When calculating a new network, it first sorts for each provided dataset the value vectors of each gene based on the global gene pool order. If the **Anchor centered** setting is activated, P-CEIN calculates for each *anchor gene* it's correlation to all the other genes. Otherwise, it calculates for each gene it's correlation to all the remaining genes of the pool (because the PCC between two vectors is symmetrical, it is pointless to go through the entire list for each next gene). For each couple of gene, P-CEIN calculates the PCC of the couple in each of the provided datasets. The couple is then saved in a network if it clears two conditions : it's PCCs are all higher than the **principal threshold** in absolute value, and they are all of the same sign. The edge representing the couple in the network is given a weight equal to the average of the couple's PCCs and a color based on the PCC's sign.
2. Once all targeted couples have been analyzed, if the **Neighborhood threshold** setting has been customized to a N value, the program will go through each gene in the network, sort all it's edges from the highest weight to the lowest and then delete all edges below the Nth highest. This substep is dependent on the order in which the genes have been add to the network. As such, it carries a risk of giving two different results if two same global gene pools don't have their genes in the same order. For this reason, it is highly unadvised to customize this setting to anything else than the default.
3. The newly built network is saved in a file and the general data of each gene are saved as well.
4. If the **Pearson dynamic threshold** is activated, the average weight of all edges in the network is calculated as well as their standard deviation and a second threshold is calculated via the formula "P = Avg + **factor**\*Std". All edges with a weight lower than this new threshold are filtered out. If after this filtering, any gene ends up with no neighbors, it is deleted from the network as well. The resulting network and it's general data are then saved.
5. If the **Minimum neighborhood 1** is customized, the program will go through all *candidate genes* and delete any who doesn't have enough *anchor genes* among it's neighbors. When several lists of *anchor genes* are provided, a *candidate gene* only needs to respect the quota of neighbors of one list to be kept in the network. It is only deleted if it doesn't respect the quota for all anchor lists. The resulting network and it's general data are then saved.
6. Finally, the last state of the network is used to fill the result **Linkage dictionary**. For each gene referenced in the dictionary at the launching or the program, a list of it's neighbors along the values of their respective PCCs (in real value) is add to the sub-dictionary of the referenced gene.

The linkage dictionary's specific structure for the P-CEIN method looks like this :

* Anchor gene 1 :
  + P-CEIN : [(candidate 1.1 , average PCC 1.1) , (candidate 1.2 , average PCC 1.2) , …]
* Anchor gene 2 :
  + P-CEIN : [(candidate 2.1 , average PCC 2.1) , (candidate 2.2 , average PCC 2.2) , …]

**Method 2 – KNN-RBH (*****K-Nearest Neighbors enhanced with Reciprocal Best Hit*)**

The *K-Nearest Neighbors enhanced with Reciprocal Best Hit* method combines the ideas of the K-Nearest Neighbors algorithm (form now on abbreviated KNN) and the Reciprocal Best Hit concept (from now on abbreviated RBH) to evaluate the association between genes.

This method's settings are the following :

* The **KNN version** allows to choose between two KNN calculations : Version *KNN\_1* is the classic KNN algorithm that looks for the same number of k neighbors for all genes. Version *KNN\_2* is a dynamic variant of the algorithm where a number of k neighbors is chosen for each gene separately.
* The **RBH sub-step** boolean indicates if you want the calculated neighborhoods to be reciprocal or not. If activated, any couple of genes where one considers the other a neighbor but the other way around is not true will not be kept as an association.
* The **Number of neighbors** is an optional value only used with Version *KNN\_1*. It is the value of k neighbors to look for each gene. The default value is 2.
* The **Threshold factor** is the multiplicative value only used with Version *KNN\_2* to calculate the dynamic k value. It is by default set to 1.

The KNN-RBH method's execution is as following :

1. It calculates each gene's neighborhood according the chosen KNN version. In version 1, all genes end up with the same number of k neighbors given by the **Number of neighbors** parameter. In version 2, the program calculate for each gene the Euclidean distance between it and all other genes, get the average distance and the standard deviation, then determines a threshold distance using the formula "D = Avg - **factor**\*Std". A given gene's neighborhood consists then in all genes whose distances are shorter than the threshold distance.
2. Once all neighborhoods are calculated, if the **RBH sub-step** is activated, the program goes through each gene's neighborhood and looks if the current gene is also in the neighborhood of each of it's neighbors. If both genes consider each other neighbors, they stay in their respective neighborhood. If not, they are deleted from the respective neighborhood.

NOTE : Step 1 and 2 run once for each provided dataset.

1. Once all datasets have been analyzed, KNN-RBH crosses for each gene it's neighborhoods from all datasets. A network is then built with an edge linking two genes that consider each other a nearest neighbor in all provided datasets. The edges are weighted with the average distance calculated between it's two genes in all datasets.
2. Once fully built, the network is saved and used to fill the **Linkage dictionary**. For each gene referenced in the dictionary at the launching or the program, a list of it's neighbors along the values of their respective average distances is add to the sub-dictionary of the referenced gene.

The linkage dictionary's specific structure for the KNN-RBH method looks like this :

* Anchor gene 1 :
  + KNN-RBH : [candidate 1.1 , candidate 1.2 , candidate 1.3 , …]
* Anchor gene 2 :
  + KNN-RBH : [candidate 2.1 , candidate 2.2 , candidate 2.3 , …]

**Method 3 – NPC (*****Network Properties Closeness*)**

The *Network Properties Closeness* method uses network properties to evaluate the association between genes. This method's core execution has been developed by Gabriel Dominico, currently in a PhD at the Federal University of Rio Grande do Sul in Brazil.

This method's settings are the following :

* The **Minimum neighborhood 3** is used to filter out any *candidate gene* with less than a certain number of anchors linked to it. It is by default set to keep all candidates. If you want to activate this filter, you must give a minimum number of neighbors for each anchor list you have provided.
* The **Minimum redundancy** is an optional value that indicates the minimum number of datasets a candidate must be found in for it to be considered relevant from the method's point of view. By default, candidates are kept only if they are found in all provided datasets for at least one *anchor gene* list.

NOTE : This method is the only one among all 4-MAT methods that explicitly needs *anchor genes* lists. It cannot run in the context of an unfocused analysis.

The NPC method's execution is as following :

1. It generates a network based on Gabriel's code. The step ends by giving a list of *candidate genes* that are considered relevant by virtue of being linked to enough *anchor genes* as specified by the **Minimum neighborhood 3** parameter.

NOTE : This step runs (n\*m) times, with "n" being the number of provided datasets and "m" the number of *anchor genes* lists. If multiple *anchor genes* lists are provided, keep in mind that for each run, *anchor genes* from another list than the one actively used in the current run will be considered *candidate genes* from thee method's point of view. This inconvenience is corrected later in the program and as such, is of no direct consequence.

1. Once all datasets and all *anchor genes* lists have been analyzed, NPC crosses the results and dresses for each of the method's candidates the list of *anchor genes* it has been found associated to, filtering out all candidates found in less than the **Minimum redundancy** threshold. All kept candidates (and their associations) are then used to build a network where edges between two genes are more or less wide based on the redundancy of the corresponding associations.
2. Once fully built, the network is saved and used to fill the **Linkage dictionary**. For each gene referenced in the dictionary at the launching or the program, sub-lists of it's neighbors are add to the sub-dictionary of the referenced gene. Each sub-list contains candidates with the same redundancy. There are as many sub-lists as there are provided datasets.

The linkage dictionary's specific structure for the NPC method looks like this :

* Anchor gene 1 :
  + NPC : [ [candidate 1.1 , candidate 1.2] , [candidate 1.3] , …]
* Anchor gene 2 :
  + NPC : [ [candidate 2.1] , [candidate 2.2] , [candidate 2.3] , …]

**Method 4 – Cluster Path**

The *Cluster Path* method uses several K-Means clustering instances to evaluate the association between genes.

This method's setting is the following :

* The **Number of clusters** is used to indicate how many clusters must be calculated by the K-Means algorithm. It is by default set to 3.

The Cluster Path method's execution is as following :

1. For each timepoint of a dataset, a 1-dimension K-Means algorithm is used to sort the genes according to the **Number of clusters** parameter. The clusters are then arranged in ascending order of their centers' values. Each gene is then associated with a vector where each value represents one of the clusters, a so-called "cluster path".
2. Once all timepoints have been analyzed, the genes are regrouped by common "cluster path".

NOTE : Sub-steeps 1 and 2 run once for each provided datasets. The results from each run is saved for the following sub-steps.

1. Once all datasets have been analyzed, the genes are regrouped common "cluster paths" in all datasets, meaning that genes that stay together in a common path, regardless of dataset and even if the path differs from a dataset to another, are grouped together.
2. Once all groups are formed, a network is built with edges between genes that are in the same group. Once fully built, the network is saved and used to fill the **Linkage dictionary**. For each gene referenced in the dictionary at the launching or the program, a list of it's neighbors is add to the sub-dictionary of the referenced gene.

The linkage dictionary's specific structure for the Cluster Path method looks like this :

* Anchor gene 1 :
  + ClusterPath : [candidate 1.1 , candidate 1.2 , candidate 1.3 , …]
* Anchor gene 2 :
  + ClusterPath : [candidate 2.1 , candidate 2.2 , candidate 2.3 , …]

**Methods Consensus**

When given a non-empty **Linkage dictionary**, the Consensus part of 4-MAT crosses the results from all methods for all genes acting as keys in the dictionary. For each couple of associated genes, a consensus level is determined based on how many methods the association has been found by. Three files are then written to sum up the final results :

* An anchor-based file, where for each *anchor gene* are sorted it's associated candidates by descending Consensus level order. This file's data are non-redundant, meaning that if a first *anchor* has a second *anchor* among it's candidate, the second *anchor* doesn't have the first *anchor* among it's. At the end of the file, the total number of associations of each Consensus level is written.
* A candidate-based file, where all associations are described. Each line informs about the two genes concerned by the association, the label of the *anchor gene* (only if *anchor genes* have been provided), the Consensus level, and for each method a boolean indicating if a given method has found the association.
* A custom graph file, firstly listing all genes along with a color corresponding to their nature (candidate or anchor), and secondly listing all edges (representing the associations) along with the Consensus level, and for each method, either a 'NO' if the method didn't find the association or the value of finding if the method did find it (values being the PCC for the P-CEIN method, the distance for the KNN-RBH method, a number of datasets for the NPC method and a 'YES' for the Cluster Path method).