


# Introduction to WGCNA

## Correlation Networks

Correlation networks are commonly used in the bioinformatic analysis of high-dimensional biological data sets. Weighted correlation network analysis can be used to find clusters, or modules of genes with correlated patterns of expression.


Gene co-expression networks can be used to increase the power of functional analysis. Co-expression networks are one of many ways to characterize the functionality of genes detected in a transcriptomic analysis.

In [correlation networks](https://peterlangfelder.com/2018/11/25/when-can-correlation-network-analysis-be-useful-and-when-you-are-better-off-not-using-it/)  (<https://peterlangfelder.com/2018/11/25/when-can-correlation-network-analysis-be-useful-and-when-you-are-better-off-not-using-it/>) each node represents a variable (feature) and links represent correlations among the variables. Networks are used to determine the variables and groups (modules) of variables that are potentially important for a property of the system that is represented by the network.


Some assumptions are made about the system and its properties, such as the:

- collective behaviour of multiple nodes in the network should be relevant for the property under investigation
- correlations should reflect, to some extent, functional relationships
- functional relationships reflected in the correlations should be relevant for the property under investigation
- calculation of correlations on the data should make sense

## WGCNA


The weighted gene co-expression network analysis ([WGCNA](https://cran.r-project.org/web/packages/WGCNA/index.html)  (<https://cran.r-project.org/web/packages/WGCNA/index.html>)) R package can be used to describe the patterns of correlation between gene expression profiles, image data, genetic marker data, proteomics data, and other high-dimensional data.



The WGCNA package constructs either [signed or unsigned networks](https://peterlangfelder.com/2018/11/25/signed-or-unsigned-which-network-type-is-preferable/)  (<https://peterlangfelder.com/2018/11/25/signed-or-unsigned-which-network-type-is-preferable/>). Signed networks take into consideration the sign of correlations to determine the connectedness of

pairs of nodes (e.g., genes), where strongly negatively correlated nodes are considered unconnected. Alternatively, unsigned networks can be used to detect genes that have mixed directions of expression. Unsigned networks are the default method of network construction in WGCNA.

## Installation

The WGCNA app can be run on a computer locally using RStudio. To run the app locally we need to download the WGCNA R Shiny app script. The script is in a GitHub repository and can be downloaded [HERE](https://github.com/ElizabethBrooks/DGEAnalysis_ShinyApps/)  ([https://github.com/ElizabethBrooks/DGEAnalysis\\_ShinyApps/](https://github.com/ElizabethBrooks/DGEAnalysis_ShinyApps/)).

**First**, download the GitHub repository using the git clone command in the terminal, for example:

```
git clone https://github.com/ElizabethBrooks/DGEAnalysis_ShinyApps.git
```

**Next**, if running the app locally, we will need to install or update [R and RStudio](https://posit.co/download/rstudio-desktop/)  (<https://posit.co/download/rstudio-desktop/>).

**Lastly**, we need to install all of the necessary R packages with the software needed to run WGCNA and create plots with the results:

```
packageList <- c("BiocManager", "shiny", "shinythemes", "dplyr", "matrixStats", "Hmisc", "splines", "foreach", "doParallel", "fastcluster", "dynamicTreeCut", "survival")


biocList <- c("WGCNA", "GO.db", "impute", "preprocessCore")

newPackages <- packageList[!(packageList %in% installed.packages()[,"Package"])]

newBioc <- biocList[!(biocList %in% installed.packages()[,"Package"])]

if(length(newPackages)){
  install.packages(newPackages)
}

if(length(newBioc)){
  BiocManager::install(newBioc)
}
```

The above code can also be found at the top of the R script for the WGCNA app, which is located in the *apps* directory of the  repository that we downloaded.

## Data Format

Before running the WGCNA app, make sure to have ready the two .csv files with the **normalized gene counts** and **experimental design**. Example data for the WGCNA app can be found in the *data/WGCNA* directory of the GitHub repository.

The normalized gene counts can be produced using the edgeR app. The following is a small example normalized gene counts table:

```
Gene,SampleOne,SampleTwo,SampleThree,SampleFour,SampleFive,SampleSix
gene-1,55.47,60.63,41.28,169.00,21.93,1.29
gene-2,55.49,58.63,77.48,187.42,24.08,3.14
gene-3,50.83,66.88,73.12,135.54,33.89,1.78
gene-4,49.06,55.19,52.74,199.91,34.34,1.67
gene-5,35.01,47.52,59.19,132.54,26.68,0.90
```

Keep in mind the *tables of gene counts with missing data may produce an error*. Recall the assumption of correlation networks that calculating correlations on the data should make sense.

The experimental design file must also be in the format required by WGCNA, for example:

```
Sample,Group
SampleOne,1
SampleTwo,1
SampleThree,1
SampleFour,2
SampleFive,2
SampleSix,2
```

## Analysis Workflow

To run the WGCNA app, open the R script **app\_shiny\_DGE\_WGCNA.R** in RStudio and press the "Run App" button in the upper right corner of the [source pane](https://docs.posit.co/ide/user/ide/guide/ui/ui-panes.html)  (<https://docs.posit.co/ide/user/ide/guide/ui/ui-panes.html>).

### Part One: Getting Started

Start in the left-hand sidebar by:

1. uploading .csv files with the normalized gene counts and experimental design



2. clicking the *Upload* button to check that the inputs are valid, which appears after the format of the inputs are checked
3. clicking the *Run Analysis* button, which appears after the input files are verified as valid for analysis

## Part Two: Data Input and Cleaning

After uploading the normalized gene counts and experimental design, an appropriate minimum branch cluster and branch cut height needs to be selected. We can use the dendrogram of sample clustering based on Euclidean distance to view outliers.

**Set** the branch cut height to remove apparent sample outliers.

Keep in mind that *errors can result from a high minimum branch cluster size or low branch cut height*, since this can result in the removal of a significant amount of data.

## Part Three: Network Construction and Module Detection

Co-expression networks need to be manually constructed by specifying a minimum module size and soft thresholding power.

**First**, select an upper value for the range of candidate soft thresholding powers, which will adjust the range of values displayed in the scale independence and mean connectivity plots.

**Next**, choose a soft thresholding power to which co-expression similarity is raised to calculate adjacency. Also, choose a minimum module size for the gene clusters.

The minimum module size and soft thresholding power should be selected where the scale free topology model fit is above 0.8 and mean connectivity under the hundreds.

Keep in mind that *errors can result from a combination of high soft thresholding power or minimum module size values*, as this will create a small number of very large modules that may not be strongly correlated.

**Lastly**, select a cut height to merge close modules. It is recommended to select a cut height of 0.25, which corresponds to a 0.75 correlation. The dendrogram that displays the clustering of module eigengenes (ME) shows the selected cut height.

The closeness of modules is measured by the correlation of the MEs. Eigengenes can be thought of as a weighted average expression profile. Modules are clustered by their calculated eigengene correlations to determine the co-expression similarity of modules.



The final clustering dendrogram of genes shows dissimilarity based on topological overlap, together with assigned merged module colors and the original module colors.

## Part Four: Network Analysis Results

There are two tables of data that should be downloaded before proceeding with additional downstream analyses, which are the:

- genes and their associated module colors and numbers
- calculated ME expression values for each module

The table of **genes and their associated module colors and numbers** can be input into the topGO app to determine the potential function of the sets of genes placed in different network modules.

The **calculated ME expression values for each module** can be visualized (e.g., box plots) or further tested for significant association to factors of interest (e.g., exact tests or ANOVAs).

## Example Script

An example R script named **WGCNA\_script\_example.R** with code for performing the same analysis as in the WGCNA R Shiny app can be found in the *scripts* directory of the GitHub repository. The script is hard coded to work for an example data set and will need to be customized for use with other data.

