# WGCNA package FAQ

Peter Langfelder and Steve Horvath

Dept. of Human Genetics, UC Los Ageles (PL, SH), Dept. of Biostatistics, UC Los Ageles (SH)

Peter (dot) Langfelder (at) gmail (dot) com, SHorvath (at) mednet (dot) ucla (dot) edu

This page provides a list of Frequently Asked Questions and our frequently given answers. Please read these before emailing us about a problem. This FAQ was last updated on December 24, 2017.

## Data analysis questions

1. **How many samples do I need?**

   We do not recommend attempting WGCNA on a data set consisting of fewer than 15 samples. In a typical high-throughput setting, correlations on fewer than 15 samples will simply be too noisy for the network to be biologically meaningful. If at all possible, one should have at least 20 samples; as with any analysis methods, more samples usually lead to more robust and refined results.

2. **Should I filter probesets or genes?**

   Probesets or genes may be filtered by mean expression or variance (or their robust analogs such as median and median absolute deviation, MAD) since low-expressed or non-varying genes usually represent noise. Whether it is better to filter by mean expression or variance is a matter of debate; both have advantages and disadvantages, but more importantly, they tend to filter out similar sets of genes since mean and variance are usually related.

   **We do not recommend filtering genes by differential expression.** WGCNA is designed to be an unsupervised analysis method that clusters genes based on their expression profiles. Filtering genes by differential expression will lead to a set of correlated genes that will essentially form a single (or a few highly correlated) modules. It also completely invalidates the scale-free topology assumption, so choosing soft thresholding power by scale-free topology fit will fail.

3. **What argument (option) settings are recommended?**

   In general, we attempt to select suitable defaults that work well in multiple applications. However, in certain cases we keep 'simple' or historical default settings for backward compatibility and reproducibility, while for new calculations we recommend settings that differ from the defaults. Some of the settings are listed below.

   - **Signed networks.** The choice of signed vs. unsigned networks is complex, but in general we prefer signed (or "signed hybrid") networks to unsigned networks. To construct signed networks, use argument `type = "signed"` or `type = "signed hybrid"` in functions such as `accuracyMeasures, adjacency, chooseOneHubInEachModule, chooseTopHubInEachModule, nearestNeighborConnectivity, nearestNeighborConnectivityMS, orderBranchesUsingHubGenes, softConnectivity` and possibly others (please see the help file for each function if in doubt). Some functions use the argument **`networkType`** to select network type; notable examples are `blockwiseModules, blockwiseConsensusModules, blockwiseIndividualTOMs, consensusTOM, intramodularConnectivity, modulePreservation, pickSoftThreshold, TOMsimilarityFromExpr, vectorTOM` but there are others as well. Again, please read the help file if in doubt.
   - **Robust correlation.** The default correlation method in all functions in WGCNA is standard Pearson correlation. In general, unless there is good reason to believe that there are no outlier measurements, we recommend (and use ourselves) the biweight mid-correlation as a robust alternative. This is implemented in WGCNA function `bicor`. Many WGCNA functions take the argument `corFnc` that allows one to specify an alternative correlation function to the standard `cor` and `bicor` is one option. Additional arguments to the correlation function can be specified using the argument `corOptions` (depending on function, this argument may require one of two alternate forms, please see the help for each function for details). In certain functions, notably the of the `blockwise` family, correlation function cannot be specified directly as a function; rather, one must use the argument `corType` to specify either Pearson or biweight mid-correlation.

     **Important cautionary notes regarding the use of bicor.** The biweight mid-correlation works very well in a variety of settings but in some situations it will produce unwanted results.

- **Restricting the number of excluded outliers: argument `maxPOutliers`.** The default version of the biweight mid-correlation, described in Langfelder and Horvath (2011) (link to article), can produce unwanted results when the data have a bi-modal distribution (e.g., when a gene expression depends heavily on a binary variable such as disease status or genotype) or when one of the variables entering the correlation is itself binary (or ordinal). For this reason, we strongly recommend using the argument `maxPOutliers = 0.05` or `0.10` whenever the biweight midcorrelation is used. This argument essentially forces `bicor` to never regard more than the specified proportion of samples as outliers.
- **Dealing with binary data.** When relating high-throughput data `x` to binary variable `y` such as sample traits, one can use argument `robustY = FALSE` to turn off the robust treatment for the `y` argment of `bicor`. This results in a hybrid robust-Pearson correlation as described in Langfelder and Horvath (2011). The hybrid correlation can also be used when one of the inputs is numeric but known to not have any outliers.

4. **Can WGCNA be used to analyze RNA-Seq data?**

   Yes. As far as WGCNA is concerned, working with (properly normalized) RNA-seq data isn't really any different from working with (properly normalized) microarray data.

   We suggest removing features whose counts are consistently low (for example, removing all features that have a count of less than say 10 in more than 90% of the samples) because such low-expressed features tend to reflect noise and correlations based on counts that are mostly zero aren't really meaningful. The actual thresholds should be based on experimental design, sequencing depth and sample counts.

   We then recommend a variance-stabilizing transformation. For example, package DESeq2 implements the function `varianceStabilizingTransformation` which we have found useful, but one could also start with normalized counts (or RPKM/FPKM data) and log-transform them using `log2(x+1)`. For highly expressed features, the differences between full variance stabilization and a simple log transformation are small.

   Whether one uses RPKM, FPKM, or simply normalized counts doesn't make a whole lot of difference for WGCNA analysis as long as all samples were processed **the same way**. These normalization methods make a big difference if one wants to compare expression of gene A to expression of gene B; but WGCNA calculates correlations for which gene-wise scaling factors make no difference. (Sample-wise scaling factors of course do, so samples do need to be normalized.)

   If data come from different batches, we recommend to check for batch effects and, if needed, adjust for them. We use ComBat for batch effect removal but other methods should also work.

   Finally, we usually check quantile scatterplots to make sure there are no systematic shifts between samples; if sample quantiles show correlations (which they usually do), quantile normalization can be used to remove this effect.

5. **My data are heterogeneous. Can I still use WGCNA?**

   Data heterogeneity may affect any statistical analysis, and even more so an unsupervised one such as WGCNA. What, if any, modifications should be made to the analysis depends crucially on whether the heterogeneity (or its underlying driver) is considered "interesting" for the question the analyst is trying to answer, or not. If one is lucky, the main driver of sample differences is the treatment/condition one studies, in which case WGCNA can be applied to the data as is. Unfortunately, often the heterogeneity drivers are uninteresting and should be adjusted for. Such factors can be technical (batch effects, technical variables such as post-mortem interval etc.) or biological (e.g., sex, tissue, or species differences).

   If one has a categorical source of variation (e.g., sex or tissue differences) and the number of samples in each category is large enough (at least 30, say) to construct a network in each category separately, it may be worthwhile to carry out a consensus module analysis (Tutorial II, see WGCNA Tutorials). Because this analysis constructs a network in each category separately, the between-category variation does not affect the analysis.

   If it is desired to construct a single network for all samples, the unwanted or uninteresting sources of large variation in the data should be adjusted for. For categorical (ordinal) factors we recommend using the function `ComBat` (from the package `sva`). Users who have never used ComBat before should read the help file for `ComBat` and work through the `sva` vignette (type `vignette("sva")` at the R prompt) to make sure they use `ComBat` correctly.

   For continuous sources of variation (e.g., postmortem interval), one can use simple linear regression to adjust the data. There may be more advanced methods out there that also allow the use of covariates and protect from over-correction.

   Whichever method is used, we caution the user that removal of unwanted sources of variation is never perfect and it can, in some cases, lead to removal of true interesting signal, and in rare cases it may introduce spurious association signal. Thus, only sources of relatively large variation should be removed.

6. **I can't get a good scale-free topology index no matter how high I set the soft-thresholding power.**

   First, the user should ensure that variables (probesets, genes etc.) have **not** been filtered by differential expression with respect to a sample trait. See item 2 above for details about beneficial and detrimental filtering genes or probesets.

If the scale-free topology fit index fails to reach values above 0.8 for reasonable powers (less than 15 for unsigned or signed hybrid networks, and less than 30 for signed networks) and the mean connectivity remains relatively high (in the hundreds or above), chances are that the data exhibit a strong driver that makes a subset of the samples globally different from the rest. The difference causes high correlation among large groups of genes which invalidates the assumption of the scale-free topology approximation.

Lack of scale-free topology fit by itself does not invalidate the data, but should be looked into carefully. It always helps to plot the sample clustering tree and any technical or biological sample information below it as in Figure 2 of Tutorial I, section 1; strong clusters in the clustering tree indicate globally different groups of samples. It could be the result a technical effect such as a batch effect, biological heterogeneity (e.g., a data set consisting of samples from 2 different tissues), or strong changes between conditions (say in a time series). One should investigate carefully whether there is sample heterogeneity, what drives the heterogeneity, and whether the data should be adjusted (see previous point).

If the lack of scale-free topology fit turns out to be caused by an interesting biological variable that one does not want to remove (i.e., adjust the data for), the appropriate soft-thresholding power can be chosen based on the number of samples as in the table below. This table has been updated in December 2017 to make the resulting networks conservative.

| Number of samples | Unsigned and signed hybrid networks | Signed networks |
|---|---|---|
| Less than 20 | 9 | 18 |
| 20-30 | 8 | 16 |
| 30-40 | 7 | 14 |
| more than 40 | 6 | 12 |

7. **The functions take many arguments! Are default settings always appropriate?**

   Many of the WGCNA functions take multiple arguments that control various subtleties in network construction and module identification. In general we attempt to provide defaults that work reasonably well in most common situations. However, in some cases we, over time, find that a different setting is more appropriate. In most cases we keep the old default for reproducibility.

8. **Functions TOMsimilarity and TOMsimilarityFromExpr give slightly different results!**

   The function `TOMsimilarityFromExpr` uses a slightly different default setting for TOM calculation in *unsigned* networks. This should produce TOM that's slightly easier to interpret but is slightly different from what one gets by calculating a standard unsigned adjacency and then TOM using `TOMsimilarity`. To get the same result, use the argument `TOMType="unsigned"` when calling `TOMsimilarityFromExpr`.

## Errors while running tutorial examples

1. **I get an error saying *could not find function "..."***

   This error nearly always occurs because R was not able to load the WGCNA package. In R, type

   ```
   library(WGCNA)
   ```

   and read the output carefully; if you see any errors (usually about a missing package needed for WGCNA or the inability to load a such a package), these need to be fixed before you can execute any WGCNA code.

2. **I get an error running your tutorials exactly as described.**

   If you get an error in function `pickSoftThreshold`, please see item 1 in Runtime errors. Otherwise, please send Peter Langfelder an email. It is certainly possible that the tutorials still contain undiscovered bugs, or that our changes to the package or changes to R have broken the offending tutorial.

3. **The tutorials run fine on example data, but produce an error on my own data.**

   The most likely culprit is the size of your data set. In particular, Sections 2.a of Tutorials I and II assume that you have less than 5000 probes in your data set. If you have more than that, please look at the corresponding section 2.c (Dealing with large data sets). Modify the argument maxBlockSize to suit the capabilities of your computer; the details are described in Section 2.c of the corresponding tutorial.

## Runtime errors

1. **I get an error when running parallel calculations - function pickSoftThreshold**

   Multiple user reports indicate that parallel (multi-threaded or cluster) calculations fail when run from third-party R GUI environments such as RStudio. If you use RStudio or other environments not supplied by R Core team, please disable parallel execution by `disableWGCNAThreads()` before running the calculations.

2. **I get errors when WGCNA code tries to start multiple threads.**

   Although most modern processsors have multiple cores, some environments, most notably clusters (such the Sun Grid Engine clusters), make only one processor core available to each process. Attempting to start multiple threads often leads to error messages similar to this example: `thread 0 could not be started successfully. Error code: 11.`

   If this happens, please disable threading (for example, using the function disableWGCNAThreads()) before calling the offending function. If this does not solve the problem, please contact Peter Langfelder

3. **I keep getting a malloc error.**

   Several Mac users have reported malloc errors such as

   ```
   R(9073,0xa013dfa0) malloc: *** mmap(size=95006720) failed (error code=12)
   *** error: can't allocate region
   *** set a breakpoint in malloc_error_break to debug
   ```

   From all we know this is a spurious and harmless message, and can be ignored. For all C- and gcc-savvy Mac users out there, if you find a cause and solution, please let us know.

4. **The code crashes with a mysterious error message.**

   Such crashes are often the result of unusual data that cause a condition we have not thought about, or subtle abnormalities in the data. Here are a few situations in which we have seen our code crash:

   - The data contains too many missing entries. We attempt to make the code resistant to missing data, but sometimes we miss something. As a test, try imputing the missing data and run a test run with imputed data.
   - The data may contain too few samples or probes.
   - The input data is not numeric. This often a subtle and hard to catch problem. When reading data in from text tables (tab-separated .txt or comma-separated .csv files), R may convert a data frame to a list and/or convert some columns to a character or a factor. This may happen, for example, because missing data in your data file are encoded as `NULL` or `N/A`, while R expects `NA` and everything else is treated as a character string. It is best to sort such problems out at the source, but it may not always be possible, and the user will need to run a version of

     ```
     charData = apply(as.matrix(data), 2, as.character);
     numData = apply(charData, 2, as.numeric);
     ```

   - Last but not least, your data may be perfectly fine but lead to no modules, or perhaps just one module, and such cases may not be handled correctly by the code (i.e., the code is buggy). Please send Peter Langfelder an email if you encounter such a situation.

5. **Another mysterious crash - data.frame vs. matrix.**

   Some errors may be caused by conversion between matrices and data frames. For most purposes, a `data.frame` and a `matrix` are equivalent and many functions will run with both types of arguments. One big exception is the handling of column names. While column names of matrices are fairly arbitrary column names of data frames must begin with a letter, underscore, or a dot. This can be a problem if, for example, the expression data are stored in a `matrix` and the probe set idetifiers begin with a number, for example, `"1552612_at"`. When converted to a data frame, R will prepend an "X" to each invalid column name, making the example `"X1552612_at"`. Such changes may cause errors, for example, in the function `plotNetworkHeatmap` and others.

6. **Code crashes when q-values are calculated**

   Some of our screening functions calculate q-values associated with a family of statistical tests. It is not uncommon that the q-value calculation fails because the p-values have an unusual distribution. We will attempt to catch such errors and prevent them from derailing the entire screening calculation.

7. **(Updated) WGCNA crashes R on my Mac!**

   We have received scattered reports of crashes on Mac OSX 10.6.x (10.5.x systems do not exhibit this error). The symptoms are various hard crashes (freeze, segmentation faults and similar) that happen when trying to execute almost

anything after loading the WGCNA package version 1.00 and below. The culprit turned out to be wrong and/or incorrectly installed Tcl/Tk. For this reason we have removed the Tcl/Tk dependency starting with WGCNA version 1.10. Please install the new version. If your problems still persist, please contact Peter Langfelder.

## Installation problems

1. **Most installation problems can be solved by using CRAN!**

   Before you spend time trying to solve an installation problem with the downloaded package, please consider installing the package from CRAN. We understand that upgrading R can seem like a bit of a hassle, but in the end it's worth it.

2. **Package impute is not available**

   As of R version 2.14.0, the package `impute` has been withdrawn from CRAN and is now available exclusively from Bioconductor. To install it, type the following lines in an R session:

   ```
   source("http://bioconductor.org/biocLite.R")
   biocLite("impute")
   ```

   This should install the package but watch for any errors that may come up.

3. **R complains about wrong package type.**

   A common cause of this error is that when the user saves the file, the operating system will uncompress or unzip it. Typically, this means the .zip or .tar.gz bundle will be decompressed and extracted, which renders the file unusable for R. For example, Mac OS X seems to automatically decompress the gzipped file. The solution is to save the file to disk as is, without letting any program such as WinZip touch it. R will decompress and unpack the package itself. On a Mac, you may have to open a terminal, change to the directory where you saved the file, and type

   ```
   gzip WGCNA_*.tar
   ```

4. **The package won't install on my Mac.**

   The best solution is to update your R to the newest version, then simply run R and use the command `install.packages("WGCNA")`. If you for some reason cannot or do not want to update your R, please look at the installation instructions and make sure you have the required XCode tools installed.

5. **I have Xcode tools installed and the package still won't install.**

   Some users have reported that a package named `gfortran.pkg` is also necessary. This may be a new feature of R as of version 2.9.0.

6. **I cannot install package qvalue.**

   **Update:** Starting with version 1.10, WGCNA does not require qvalue. Please install the newest version of WGCNA; this should cure all Tcl/Tk and qvalue installation problems.

7. **I would like to use a 64-bit version of WGCNA for Windows.**

   Installing the package using CRAN will automatically provide the 64-bit version if appropriate.

## Version compatibility issues

1. **I installed a newer WGCNA version and now I'm getting different results!**

   As we continue to develop and improve the methods contained in this package, from time to time the default calculations methods and arguments may change. We do our best to preserve options that will allow the user to replicate old results

using a new package version, and we document the changes in the changelog. If you cannot figure out the necessary arguments to reproduce an old result, please send Peter Langfelder an email.

## General questions

1. **How can I make network construction execute faster?**

   When constructing a network from a data set of a typical genomic size (i.e., between 10 000 and 30 000 genes or other variables), the most time consuming step is the calculation of Topological Overlap Matrix which involves multiplying matrices with tens of thousands of rows and columns. With a standard R distribution, this may take multiple hours even on a modern workstation since matrix multiplication in standard R does not take advantage of multi-threading (parallel execution). It is possible to speed up this process by a factor of 10-100 by installing a speed-optimized Basic Linear Algebra Subprograms (BLAS) library and compiling R against it. The process of compiling R against an enhanced BLAS library is described in the R installation and adminitration manual. Compiling R on Linux and Unix flavors is usually relatively simple and straightforward. On Mac OSX and (more so) on Windows it requires installing additional tools and packages. Although it is helpful to have administrator privileges to compile and install R, it is usually not necessary. See the R installation and adminitration manual for full details.

2. **Can WGCNA use multiple processors or cores (parallel execution)?**

   Some of the WGCNA code is written to take advantage of parallel execution to speed up the calculations. There are two main parallel computation mechanisms used by WGCNA: the compiled functions cor and bicor use POSIX threading for a part but not all of the calculation. This parallel code is only available on platforms that have POSIX threads available (various Linux and Unix flavors and Mac OS). It is not available on Windows. Some functions (such as pickSoftThreshold) are able to use POSIX threads where they are available and use SNOW clusters where multi-threading is not available. The users should be aware that POSIX and cluster parallel execution are very different and are not interchangeable; in fact, many cluster environments only allocate a single core to each process ("job") and attempting to start additional threads leads to errors (see Runtime errors below).

   Even on systems where threading is available **it is disabled by default**. This is a conservative setting that may slow down the calculations but will also prevent WGCNA from grabbing all available processor cores. There are two ways to enable multi-threading:

   - Within R, call the function

     ```
     allowWGCNAThreads()
     ```

     to allow threading from within WGCNA. Multi-threading can be turned off again using the function `disableWGCNAThreads()`. See the help file for `allowWGCNAThreads()` for more details.

   - Set the environment variable

     ```
     ALLOW_WGCNA_THREADS=<number_of_processors>,
     ```

     for example, if you have 2 cores (or want to use 2 cores),

     ```
     ALLOW_WGCNA_THREADS=2
     ```

     .

     If you don't know what an environment variable is, please use the above `allowWGCNAThreads()` within R.

   Please note that this setting does not affect the multi-threading status of the underlying BLAS library.

3. **Is there a GUI interface to WGCNA?**

   In short, no. There used to be one but it is hopelessly broken and out of date.

---