WGCNA Pipeline - part one: Correlating mRNA

Rachel Richardson

January 29, 2019

Weighted Gene Correlation Network Analysis (WGCNA) is a method often used for correlating microarray or RNA-seq data to external data from the same group of subjects. It is advantegous when trying to avoid multiple testing problems when comparing against thousands of genes, as it groups similarly transcribed genes into groups called modules. The overall transcriptional trend of each model (an “eigengene”, generated on the same principals as a principal component analysis) is correlated against external data.

The WGCNA as an analysis method is described in

Zhang B and Horvath S (2005) A General Framework for Weighted Gene Co-Expression Network Analysis, Statistical Applications in Genetics and Molecular Biology: Vol. 4: No. 1, Article 17 PMID: 16646834 The package implementation is described in the article

Langfelder P, Horvath S (2008) WGCNA: an R package for weighted correlation network analysis. BMC Bioinformatics 2008, 9:559 (link to paper)

Peter Langfelder and Steve Horvath with help of many other contributors have a website with associated information and tutorials at: https://horvath.genetics.ucla.edu/html/CoexpressionNetwork/Rpackages/WGCNA/ (https://horvath.genetics.ucla.edu/html/CoexpressionNetwork/Rpackages/WGCNA/)

Our analysis uses WGCNA to correlate mRNA-seq data to lipid species that are significantly different in offspring whose mothers were fed either a high-fat diet or a healthy control diet.

For readability, we've split the code for our analysis into three parts. The first step in our WGCNA is processing the mRNA data. But before that, we need to load a few packages.

WGCNA, S4Vectors, and DESeq2 are used in part one.

```
#Normalizing data function vst
library(DESeq2)

#Install S4 for sumCols function
library(S4Vectors)

#WGCNA package install
library(WGCNA)
```

```
## ================================================================================
## *
## *  Package WGCNA 1.66 loaded.
## *
## ================================================================================
```

```
#Use multithreading for increased speeds:
allowWGCNAThreads()
```

```
## Allowing multi-threading with up to 8 threads.
```

We next need to read in our data. For continuity with other parts of the WGCNA, we will upload RNA-seq and lipid abundance datasets. Datasets are availible in the github repo.

Import RNA-seq data

```
#We read in our counts and add the column names beloging to their respective samples
GeneCounts = read.table("GeneCountsTotal", header = F, row.names = 1, sep = "\t", chec
k.names=FALSE)
GeneCounts = rbind(c("T100","T101","T102","T103","T104","T105","T106","T107","T108","T
109","T201","T202","T203","T204","T205","T300","T304","T305","T306","T307","T308","T30
9","T311","T312","T313","T400","T401","T402","T404","T405","T406","T407","T408","T50
0","T501","T502","T503","T504","T505","T506"), GeneCounts)
```

Import Lipid Data

```
#Read in Lipid Data

DGs1.2 <- read.csv("C:/Users/rachel/McCurdyPro/LipidData/LipidData/CSVs from given dat
a/Processed CSVs (format and LOQ)/Modified LOQ/1.2DGs.changed LOQ.csv",
                   head = TRUE, row.names = 1)
AC <- read.csv("C:/Users/rachel/McCurdyPro/LipidData/LipidData/CSVs from given data/Pr
ocessed CSVs (format and LOQ)/Modified LOQ/AC.formatted.noloq.csv",
               head = TRUE)
Cer <- read.csv("C:/Users/rachel/McCurdyPro/LipidData/LipidData/CSVs from given data/P
rocessed CSVs (format and LOQ)/Modified LOQ/Cerimides.changedLOQ.csv",
                head = TRUE, row.names = 1)

DGs1.3 <- read.csv("C:/Users/rachel/McCurdyPro/LipidData/LipidData/CSVs from given dat
a/Processed CSVs (format and LOQ)/Modified LOQ/CM033018 1-3DGs.changedLOQ.csv",
                   head = TRUE,  row.names = 1)
names(DGs1.3) <- paste(names(DGs1.3), ".1.3DGs", sep = "")  #Accounts for poor headers

dh <- read.csv("C:/Users/rachel/McCurdyPro/LipidData/LipidData/CSVs from given data/Pr
ocessed CSVs (format and LOQ)/Modified LOQ/dhCer.changedLOQ.csv",
               head = TRUE,  row.names = 1)
Glu <- read.csv("C:/Users/rachel/McCurdyPro/LipidData/LipidData/CSVs from given data/P
rocessed CSVs (format and LOQ)/Modified LOQ/GluCer.changedLOQ.csv",
                head = TRUE,  row.names = 1)
hex <- read.csv("C:/Users/rachel/McCurdyPro/LipidData/LipidData/CSVs from given data/P
rocessed CSVs (format and LOQ)/Modified LOQ/hexosylCer.changedLOQ.csv",
                head = TRUE,  row.names = 1)
Lac <- read.csv("C:/Users/rachel/McCurdyPro/LipidData/LipidData/CSVs from given data/P
rocessed CSVs (format and LOQ)/Modified LOQ/LacCer.changedLOQ.csv",
                head = TRUE,  row.names = 1)
mye <- read.csv("C:/Users/rachel/McCurdyPro/LipidData/LipidData/CSVs from given data/P
rocessed CSVs (format and LOQ)/Modified LOQ/Sphingomyelins.formatted.noloq.csv",
                header = TRUE,  row.names = 1)
sine <- read.csv("C:/Users/rachel/McCurdyPro/LipidData/LipidData/CSVs from given data/
Processed CSVs (format and LOQ)/Modified LOQ/Sphingosine.formatted.noloq.editfordismat
rix.csv",
                 header = TRUE, row.names = 1)
TAG <- read.csv("C:/Users/rachel/McCurdyPro/LipidData/LipidData/CSVs from given data/P
rocessed CSVs (format and LOQ)/Modified LOQ/TAG.changedLOQ.csv",
                head = TRUE,  row.names = 1)
```

We will next process this data into one dataframe. This can be done later or avoided all together, but will require modification of function parameters later on.

```
#Bind the lipid data into one dataframe

ALL <- cbind(AC, Cer, DGs1.2, DGs1.3, dh, Glu, hex, Lac, mye, sine, TAG)

#Translate the RNA-seq data set for processing with the lipid data then create a dataf
rame for all of the Data
Gene <- t(GeneCounts)

#Use subject ID to merge genes and lipid data
Fulldata <- merge(ALL, Gene, by.x = 1, by.y = 1)

#Remove subject ID as row and make into row names
row.names(Fulldata) <- Fulldata[,1]
Fulldata <- Fulldata[2:32508]

#Convert to numeric for RNA-seq data processing
cvt <- sapply(Fulldata, is.factor)
Fulldata[cvt] <- lapply(Fulldata[cvt], function(x) as.numeric(as.character(x)))

#Rename columns in a cleaner format than the given excel sheet
Labels <- gsub("..pmol.", "", colnames(Fulldata))
Labels <- gsub("X", "", Labels)
colnames(Fulldata) <- Labels
```

WGCNA works best with normalized RNA data that is free from the noise of low level transcripts.

We filter out any genes with sums below 10 (consistent with the filtering used in differential expression methods), use Goodsamps to ensure low-varience genes or genes with many missing values are removed, than normalize with vst.

```
#Remove RNA-seq data with low counts
keep <- colSums(Fulldata) >= 10
Fulldata <- Fulldata[,keep]

#Checks which samples are low-varience and complete (not missing values) for the major
ity of samples.
Goodsamps <- goodSamplesGenes(Fulldata)
```

```
##  Flagging genes and samples with too many missing values...
##   ..step 1
```

```
Goodsamps <- Fulldata[Goodsamps$goodGenes]

#Normalize data of genes, in our case all columns after the 121st column.
#(vst requires application on genes in rows while our data is in columns, so t() is us
ed to tranform data.)
Goodsamps<- data.frame(Goodsamps[1:121],t(vst(t(Goodsamps[122:length(Goodsamps)]))))
```

```
## converting counts to integer mode
```

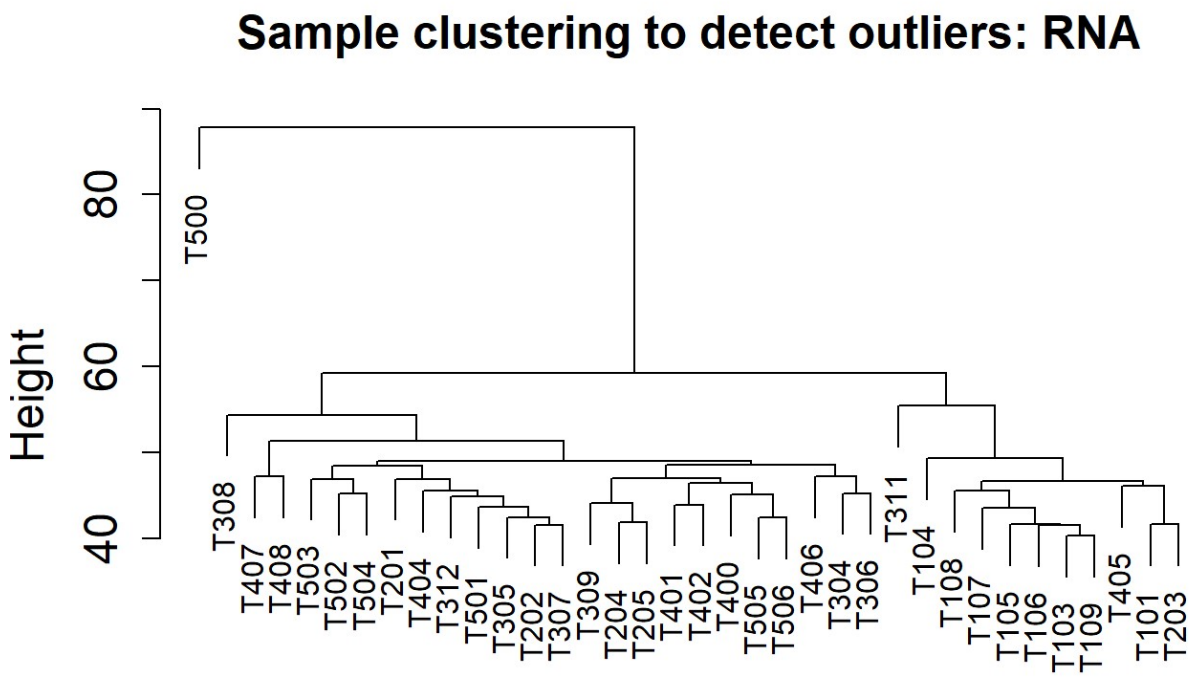
```
#Remove "X" from t() transformations
colnames(Goodsamps) <- gsub("X", "", colnames(Goodsamps))
```

One final cleaning procedure for the data would be to handle outliers.

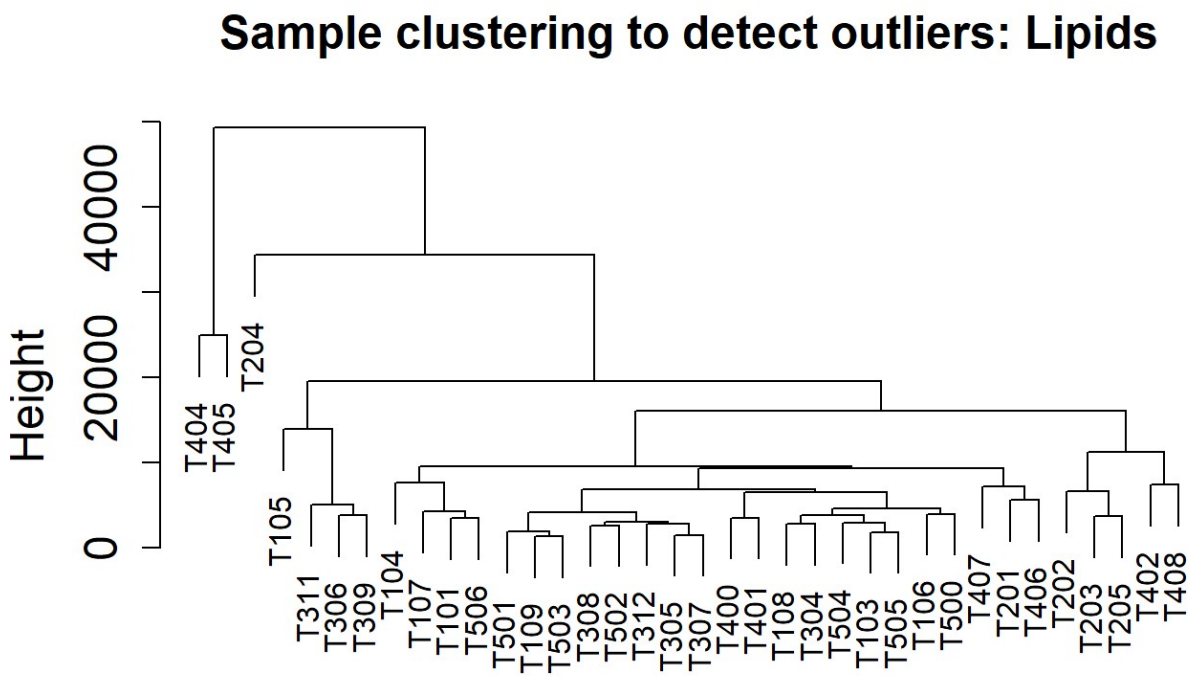
We can detect outliers in our samples by plotting a hierarchical cluster analysis of the RNA data, and lipid data.

```
#Displays the similarity of the samples based on RNA-seq data
sampleTree1 <- hclust(dist(Goodsamps[122:length(Goodsamps)]), method = "average")
sampleTree2 <- hclust(dist(Goodsamps[1:121]), method = "average")


plot(sampleTree1, main = "Sample clustering to detect outliers: RNA",
     sub="", xlab="", cex.lab = 1.5, cex.axis = 1.5, cex.main = 1.5)
```




```
plot(sampleTree2, main = "Sample clustering to detect outliers: Lipids",
     sub="", xlab="", cex.lab = 1.5, cex.axis = 1.5, cex.main = 1.5)
```



We avoid comparing the overall data since the abundance data is scaled differently than the normalized RNA-seq data, but we can see that the outliers are not consistant between groups. Outlier removal is a common approach to avoid the effects of outliers, but our analysis instead used methods robust to outliers (specifically biweight midcorrelation). Pick appropriate methods based on the dataset.

Later on, we can compare these plots to external data as well.

Continuing the analysis, this dataset is suitable for continuing with WGCNA. Lets see how it looks:

```
head(Goodsamps[,1:7])
```

```
##      10.0.AC 12.0.AC 14.0.AC 16.0.AC 16.1.AC 18.0.AC 18.1.AC
## T101    58.9    38.1   154.3  1195.3   203.9   646.1  1359.3
## T103    36.6    26.5    71.7   425.4    78.5   274.9   411.4
## T104    40.5    31.8   129.7  1177.7   156.0   518.7  1171.6
## T105    37.7    16.7    58.5   250.2    71.7   145.0   271.2
## T106    58.8    34.1   123.9   953.2   169.9   442.4  1058.8
## T107    22.2    22.9    83.0   636.6   132.0   379.0   821.4
```

The next step in the analysis is about building a network adjacency matrix and a topology overlap matrix to get the most informative gene modules. Integral to this step is the establishing of a soft threshold to approximate scale-free topology. This process reduces noise in the data by raising the similarity values to a power (the soft threshold) and emphasizes strong correlations. The scale free topology index and network connectivity data is calculated from the values raise to the soft-threshold.

There are other considerations to take into account regarding soft thresholding; signed networks are typically strictly concerned with correlation, while unsigned takes into account inverse correlation as well. Different soft thresholds are expected based on the network type. Here, we generate an unsigned network.

This analysis picks a soft threshold using the pickSoftThreshold function of WGCNA using the biweight midcorrelation method mentioned above.

```
#Creates a list of powers to test for fitting the data, determines a soft threshold fr
om these numbers. Unsigned soft thresholds are typically < 15
powers <- c(1:14)


#Generate soft threshold values from the data
#bicor is Biweight Midcorrelation, used to reduce influence of outliers in matrix

sft <- pickSoftThreshold(Goodsamps[122:length(Goodsamps)], powerVector = powers, verbo
se = 5, networkType = "unsigned", corFnc = "bicor")
```

```
## pickSoftThreshold: will use block size 2586.
##  pickSoftThreshold: calculating connectivity for given powers...
##    ..working on genes 1 through 2586 of 17300
##    ..working on genes 2587 through 5172 of 17300
##    ..working on genes 5173 through 7758 of 17300
##    ..working on genes 7759 through 10344 of 17300
##    ..working on genes 10345 through 12930 of 17300
##    ..working on genes 12931 through 15516 of 17300
##    ..working on genes 15517 through 17300 of 17300
##    Power SFT.R.sq slope truncated.R.sq mean.k. median.k. max.k.
## 1      1    0.755 -2.55          0.842 3110.00  2.85e+03   5550
## 2      2    0.847 -2.13          0.882  904.00  7.05e+02   2700
## 3      3    0.852 -1.93          0.885  341.00  2.14e+02   1580
## 4      4    0.886 -1.76          0.920  154.00  7.45e+01   1030
## 5      5    0.900 -1.65          0.941   78.50  2.85e+01    711
## 6      6    0.914 -1.58          0.960   44.00  1.18e+01    517
## 7      7    0.911 -1.54          0.962   26.50  5.14e+00    392
## 8      8    0.912 -1.52          0.971   16.90  2.38e+00    308
## 9      9    0.913 -1.49          0.977   11.30  1.14e+00    247
## 10    10    0.904 -1.49          0.976    7.76  5.71e-01    202
## 11    11    0.897 -1.49          0.975    5.51  2.97e-01    167
## 12    12    0.891 -1.49          0.974    4.01  1.58e-01    140
## 13    13    0.900 -1.48          0.981    2.99  8.67e-02    119
## 14    14    0.895 -1.49          0.981    2.26  4.85e-02    102
```

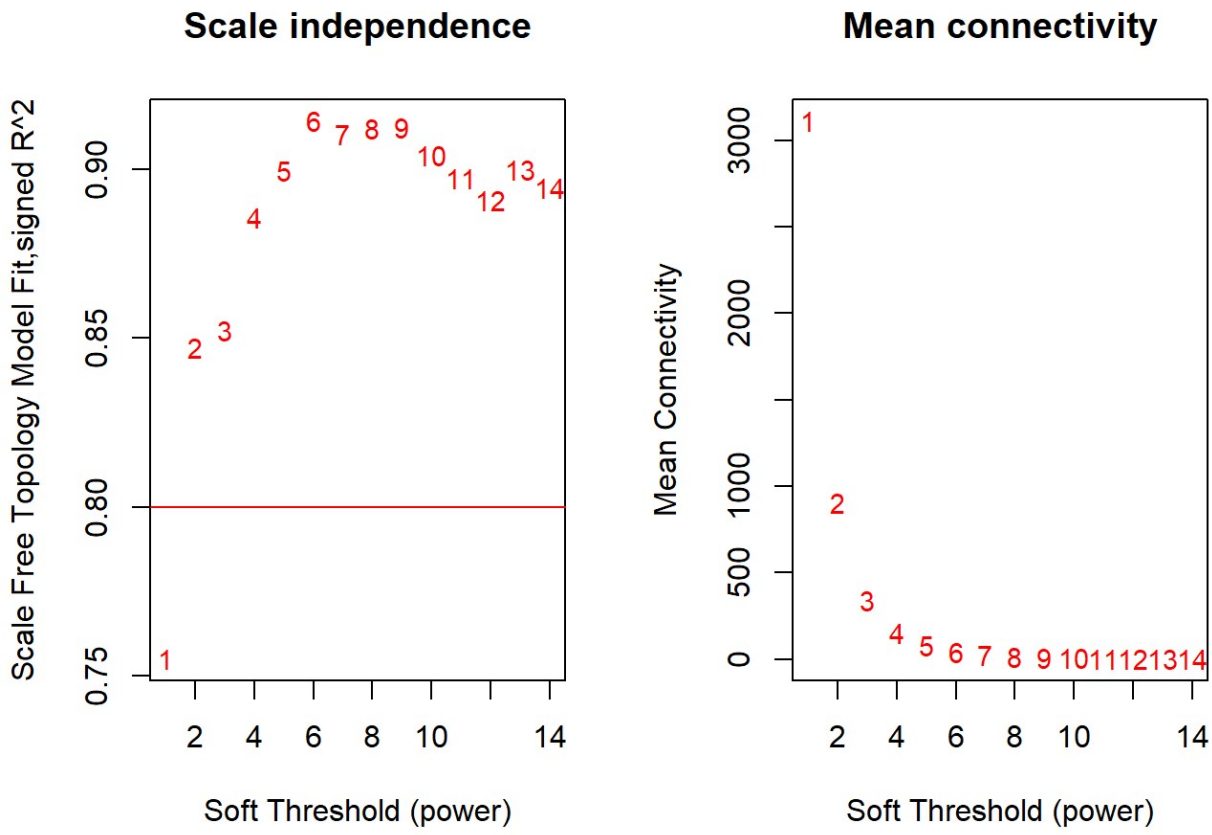
We can plot the soft threshold calculations in order to determine which threshold is best for our data.

```
#Plots results from soft threshold determination above

par(mfrow=c(1,2))

plot(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
xlab="Soft Threshold (power)",ylab="Scale Free Topology Model Fit,signed R^2",type
="n",
main = "Scale independence");
text(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
labels=powers,cex=0.9,col="red");
abline(h=0.80,col="red")

plot(sft$fitIndices[,1], sft$fitIndices[,5],
xlab="Soft Threshold (power)",ylab="Mean Connectivity", type="n",
main = "Mean connectivity")
text(sft$fitIndices[,1], sft$fitIndices[,5], labels=powers, cex=0.9,col="red")
```



Thresholds are typically between 5 and 8 based on the number of samples in the analysis and picked above the 0.80 line for scale free topology fit. We use 6 as the soft threshold, noting that model fit is reduced at higher powers.

We can now calculate our adjacency and topology overlap matrixes. NOTE: This is a time intensive step and may require increased memory in R. For reloading purposes, it may be beneficial to save the final matrix as an R object to avoid needing to re-run this step.

(While this step is running, you can move on to part two in the workflow!)

Generate matrixes

```
#Adjust for memory limit
memory.limit(20000)

#Calculates adjacency based on soft threshold, biweight correlation calculations
softPower <- 6
adjacency <- adjacency(Goodsamps[122:length(Goodsamps)], type = "unsigned", power = so
ftPower, corFnc = "bicor")

#Topology overlap matrix silimarity comparison. Essentially, under the theory that th
e data concerning the relationship between two genes may be incomplete, comparison to
adjacent nodes helps inform the true similarity in different genes.

TOM <- TOMsimilarity(adjacency, TOMType = "unsigned")
dissTOM <- 1-TOM
```

Save final matrix as an R object; you can aslo save the good samples dataframe as well.

```
saveRDS(dissTOM, file = "dissTOM.rds")
saveRDS(Goodsamps, file = "Goodsamps.rds")
```

Having generated our matrix of relationships between our genes, we are now ready to move on to part two of the WGCNA workflow, located in the github repository.