Weighted gene coexpression network analysis (WGCNA) practical

Integrative Genomics module

Michael Inouye
Centre for Systems Genomics
University of Melbourne, Australia

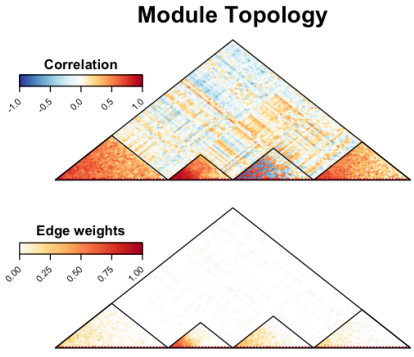
Summer Institute in Statistical Genetics 2016
Seattle, USA

@minouye271
inouyelab.org

THE UNIVERSITY OF MELBOURNE

Gene co-expression networks

- **Weighted, undirected complete gene network**
  - **Nodes**: genes/probes
  - **Edges**: $|cor(node\_i, node\_j)|^{\nu}$
    - Scale-free assumption and [0,1]
- **Identify subnets (modules/ clusters)**
  - Typically subnets represent known biological pathways
  - Various methods and tools for clustering



What we're doing today

- Data management and filtering
- Network construction
- Module detection
- Module association analysis

Getting started
(if you haven't already done so)

Setup

First, we installed the WGCNA package. Some its dependencies are in the BioConductor repository rathern than CRAN. We needed to install these dependencies manually, because `install.packages` will not do it for us:

```
source("https://bioconductor.org/biocLite.R")
# You can answer no ('n') to any prompt that asks to update old packages.
biocLite(c("impute", "preprocessCore", "GO.db", "AnnotationDbi"))

# Now we can install WGCNA from CRAN.
install.packages("WGCNA")
```

Data filtering

First, we will loaded in mouse adipose gene expression data (adapted from Yang X et al, *Genome Res.* 2006 Aug; 16(8): 995–1004, full multi-tissue set freely available from Sage BioNetworks).

```
read.matrix <- function(file) {
  df <- read.table(file, header=TRUE, row.names=1, sep="\t",
                   check.names=FALSE)
  mat <- as.matrix(df)
  # Avoid having numeric IDs
  rownames(mat) <- paste0("Probe_", rownames(mat))
  colnames(mat) <- paste0("Sample_", colnames(mat))
  return(mat)
}

setwd("/Users/minouye/Documents/Courses_Workshops/SISG/2016/prac_WGCNA/
curatedExpressionLiver")

liver_ge <- read.matrix("expression_head2500.txt")
```

Next, we removed remove probes that failed for >5% of samples, and then samples where >5% of their assays failed.

```
# Removes probes that have more the 5% of their observations missing
filter_probes <- function(x) {
  # How many samples are missing for each probe?
  nMissing <- apply(x, 2, function(probe) {
    sum(is.na(probe))
  })
  missingness <- nMissing/nrow(x)
  return(x[, missingness <= 0.05])
}

# Removes samples that failed > 5% of their assays
filter_samples <- function(x) {
  # How many samples are missing for each probe?
  nMissing <- apply(x, 1, function(sample) {
    sum(is.na(sample))
  })
  missingness <- nMissing/ncol(x)
  return(x[missingness <= 0.05,])
}

liver_ge_fil <- filter_samples(filter_probes(liver_ge))
```

Then we imputed the remaining missing observations using the K-nearest neighbours algorithm in the *impute* package:

```
# This package is a dependency of WGCNA so we have installed it already.
library(impute)

if (any(is.na(liver_ge_fil)))
  liver_ge_fil_imp <- impute.knn(liver_ge_fil)$data

anyNA(liver_ge_fil)
anyNA(liver_ge_fil_imp)
```

To reduce the burden of computation for the purposes of software testing we will only analyse the top few thousand most variable and most connected probes. This is standard practice when performing weighted gene coexpression network analysis on computers with limited resources (Ghazalpour *et al.*, 2006).

Following Ghazalpour *et al.*, first we first get the top 1,000 most varying probes in the liver tissue:

```
most_varying <- function(ge, topN=1000) {
  standard_deviation <- apply(ge, 1, sd)
  sorted <- sort(standard_deviation, decreasing=TRUE)
  sorted_names <- names(sorted)
  topN_names <- sorted_names[seq_len(topN)]
  return(topN_names)
}

top_varying <- most_varying(liver_ge_fil_imp)
liver_ge_fil_imp_top1000 <- liver_ge_fil_imp[top_varying,]
```

Network inference

Next we will infer the interaction networks. If we want to use *NetRep*, we need to save both the correlation structure (coexpression), as well as the interaction network (adjacency matrix) inferred through *WGCNA*:

```
library(WGCNA)
## Warning: package 'WGCNA' was built under R version 3.2.3
calculate_coexpression <- function(ge) {
  coexpression <- cor(t(ge), method="pearson")
}

infer_network <- function(coexpression) {
  # First pick the soft threshold to use to define the interaction network
  sft <- WGCNA::pickSoftThreshold(abs(coexpression), dataIsExpr=FALSE)
  if (is.na(sft$powerEstimate)) {
    sft$powerEstimate <- 1
    warning("Could not satisfy the scale-free topology criterion")
  }
  network <- abs(coexpression)^sft$powerEstimate
}

liver_coexpression <- calculate_coexpression(liver_ge_fil_imp_top1000)
liver_network <- infer_network(liver_coexpression)
```

What's it doing?

- Calculate Pearson correlation coefficients between all pairs of genes
- Use a power transform to satisfy scale-free topology criteria (select soft power threshold)
- Infer a network where
  - Nodes: Genes
  - Edges: Pearson correlations raised to the selected power

Next we will identify tightly coexpressed modules in the liver tissue network.

```
detect_modules <- function(ge, network) {
  # Calculate the distance between probes based on their topological similarity:
  # i.e. the strength of their coexpression as well as the similarity of their
  # patterns of coexpression to all other probes
  probe_dist <- WGCNA::TOMdist(network)
  dimnames(probe_dist) <- dimnames(network)

  # Hierarchically cluster based on this distance metric
  dendro <- hclust(as.dist(probe_dist), method="average")

  # Detect modules. `cutreeDynamic` is a function in the `dynamicTreeCut`
  # package, which is loaded in by the `WGCNA` package.
  module_labels <- cutreeDynamic(dendro, distM = probe_dist)
  names(module_labels) <- colnames(network)

  # Merge similar modules
  merged <- mergeCloseModules(t(ge), module_labels, relabel=TRUE)
  module_labels <- merged$colors

  return(module_labels)
}

liver_modules <- detect_modules(liver_ge_fil_imp_top1000, liver_network)
```

What's it doing?

- Goal: Get the most coherent gene subnetworks as possible
- Instead of using the correlation-based edges, WGCNA is calculating a distance measure called topological similarity (TOM):

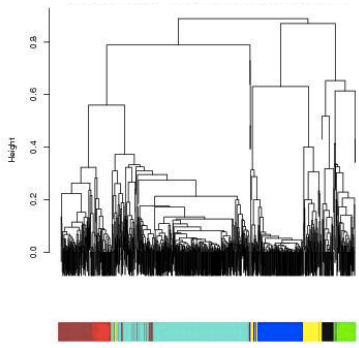
$$t_{ij} = \begin{cases} \frac{|N_1(i) \cap N_1(j)| + a_{ij}}{\min\{|N_1(i)|, |N_1(j)|\} + 1 - a_{ij}} & \text{if } i \neq j \\ 1 & \text{if } i = j. \end{cases} \quad (1)$$

where $N_1(i)$ denotes the set of direct neighbors of i excluding i itself and $|\cdot|$ denotes the number of elements (cardinality) in its argument. The quantity $|N_1(i) \cap N_1(j)|$ measures the number of common neighbors that nodes i and j share whereas $|N_1(i)|$ gives the number of neighbors of i. The topological overlap t_{ij} assumes a minimal value of 0 if there is no direct linkage between the two nodes and if they share no common direct neighbors. It assumes a maximum value of 1 if there is a direct link between the two nodes and if one set of direct neighbors is a subset of the other. The fact that t_{ij} is bounded between 0 and 1 is used in the definition of the topological overlap based dissimilarity measure (see Eq. 4).

Yip & Horvath, BMC Bioinf 2007

What's it doing?

- Hierarchical clustering of TOM matrix
- Move through the dendrogram with a dynamic cutting algorithm



Yip & Horvath, BMC Bioinf 2007

Each probe is now assigned a (numeric) module label:

```
table(liver_modules)
## liver_modules
##   0   1   2   3   4   5   6   7
## 395 168 166  93  61  59  32  26
```

Where "0" corresponds to the network "background": all genes that did not cluster into coexpression module.

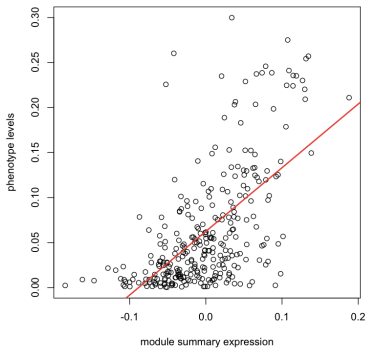
Saving the data

Finally, we will save the data for processing with *NetRep*:

```
# Create the directory to store the data in.
dir.create("test_data")

write.matrix <- function(x, file) {
  write.csv(x, file, quote=FALSE)
}

write.vector <- function(x, file, col.names) {
  column_matrix <- t(t(x))
  colnames(column_matrix) <- col.names
  write.csv(column_matrix, file, quote=FALSE)
}

# NetRep requires the probes to be columns, so we will transpose when
# saving
write.matrix(t(liver_ge_fil_imp_top1000), "test_data/liver_expression.csv")

write.matrix(liver_coexpression, "test_data/liver_coexpression.csv")

write.matrix(liver_network, "test_data/liver_network.csv")

write.vector(liver_modules, "test_data/liver_modules.csv", col.names="Module")
```

Phenotype association analysis

```
MEs <- moduleEigengenes(t(liver_ge_fil_imp_vary), liver_modules)

names(MEs$varExplained) <- colnames(MEs$eigengenes)
MEs$varExplained

# add in sample names to the eigengenes.
rownames(MEs$eigengenes) <- colnames(liver_ge_fil_imp_vary)

pheno<-read.table("pheno.txt", header=TRUE)
data<-cbind(pheno,MEs$eigengenes)

plot(data$ME0,data$pheno)

plot(data$ME6,data$pheno,xlab="module summary expression",ylab="phenotype levels",pch=20)
summary(lm(pheno ~ ME6, data=data))
abline(intercept,slope,col="red",lwd=2)
```



References

Ghazalpour, A., Doss, S., Zhang, B., Wang, S., Plaisier, C., Castellanos, R., Brozell, A., Schadt, E.E., Drake, T.A., Lusis, A.J., et al. (2006). Integrating genetic and network analysis to characterize genes related to mouse weight. PLoS Genet. *2*, 1182–1192.

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

Yang, X., Schadt, E.E., Wang, S., Wang, H., Arnold, A.P., Ingram-Drake, L., Drake, T.A., and Lusis, A.J. (2006). Tissue-specific expression and regulation of sexually dimorphic genes in mice. Genome Res. *16*, 995– 1004.

Special thanks to Scott Ritchie
Network inference adapted from his script