RNA-Seq Data Analysis 3

Lauren Vanderlinden

BIOS 6660

Spring 2019

Overview From Last Time

- Multiple Testing Issue in ‘Omics Analyses
- How to Visualize Results
- Enrichment Analyses
- Transformations

- Quick Note on Yampa Calling Programs:

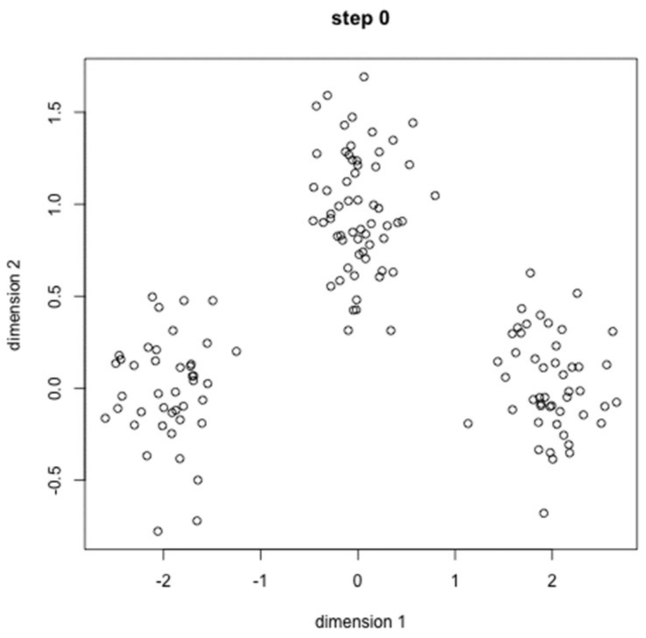
```
export PATH=/usr/local/bin/samtools-1.3:$PATH
export PATH=/usr/local/bin/bowtie2:$PATH
```

Network Analyses

- Data driven network analyses
- Nodes are genes
- Edges are the relationship between the genes
- Application is for when you want to know what genes are working together in your system
- Common Models:
  - K-means clustering
  - Hierarchical clustering
  - WGCNA

K-means Clustering

- First define how many groups you have
  - This is NOT ideal
  - Look at PC plots to get an idea
- Randomly initialize the groups center points
- Compute the distances between each point and each group center
  - A node becomes a member of a group based on smallest distance
- Re-compute cluster center by taking group means
  - Iterative process till group centers don't change much



https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

WGCNA: Weighted Gene Co-expression Network Analysis

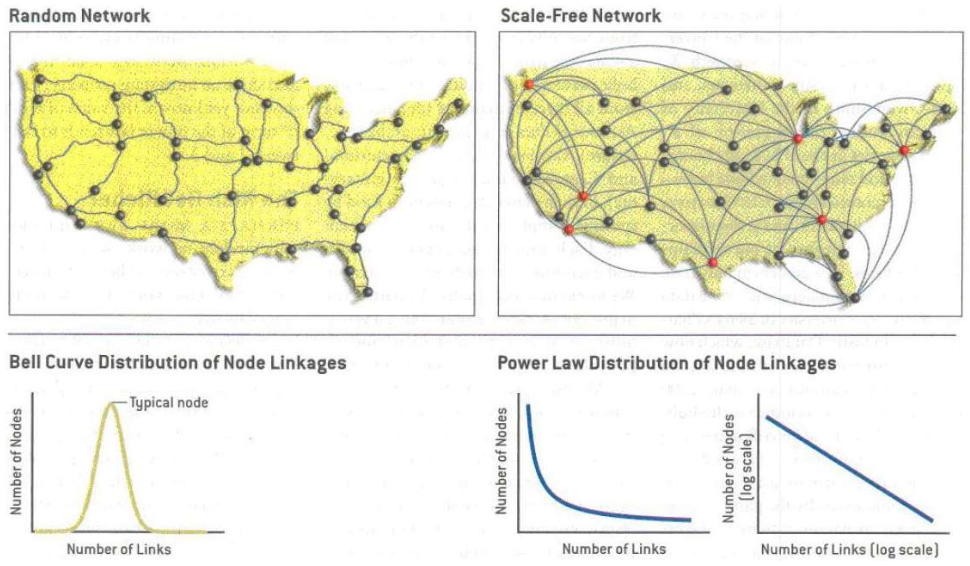
- Popular method, well documented and very easy to use in R
  - Optimized for microarrays
  - This was my first time working in R!
- Original Paper:
  - Bin Zhang and Steve Horvath *(2005) "A General Framework for Weighted Gene Co-Expression Network Analysis",* Statistical Applications in Genetics and Molecular Biology: Vol. 4: No. 1, Article 17 PMID: 16646834
  - http://dibernardo.tigem.it/files/papers/2008/zhangbin-statappsgeneticsmolbio.pdf
- WGCNA links:
  - https://labs.genetics.ucla.edu/horvath/CoexpressionNetwork/Rpackages/WGCNA/Tutorials/
  - https://labs.genetics.ucla.edu/horvath/CoexpressionNetwork/OverviewWGCNA.pdf

WGCNA: More than Correlation

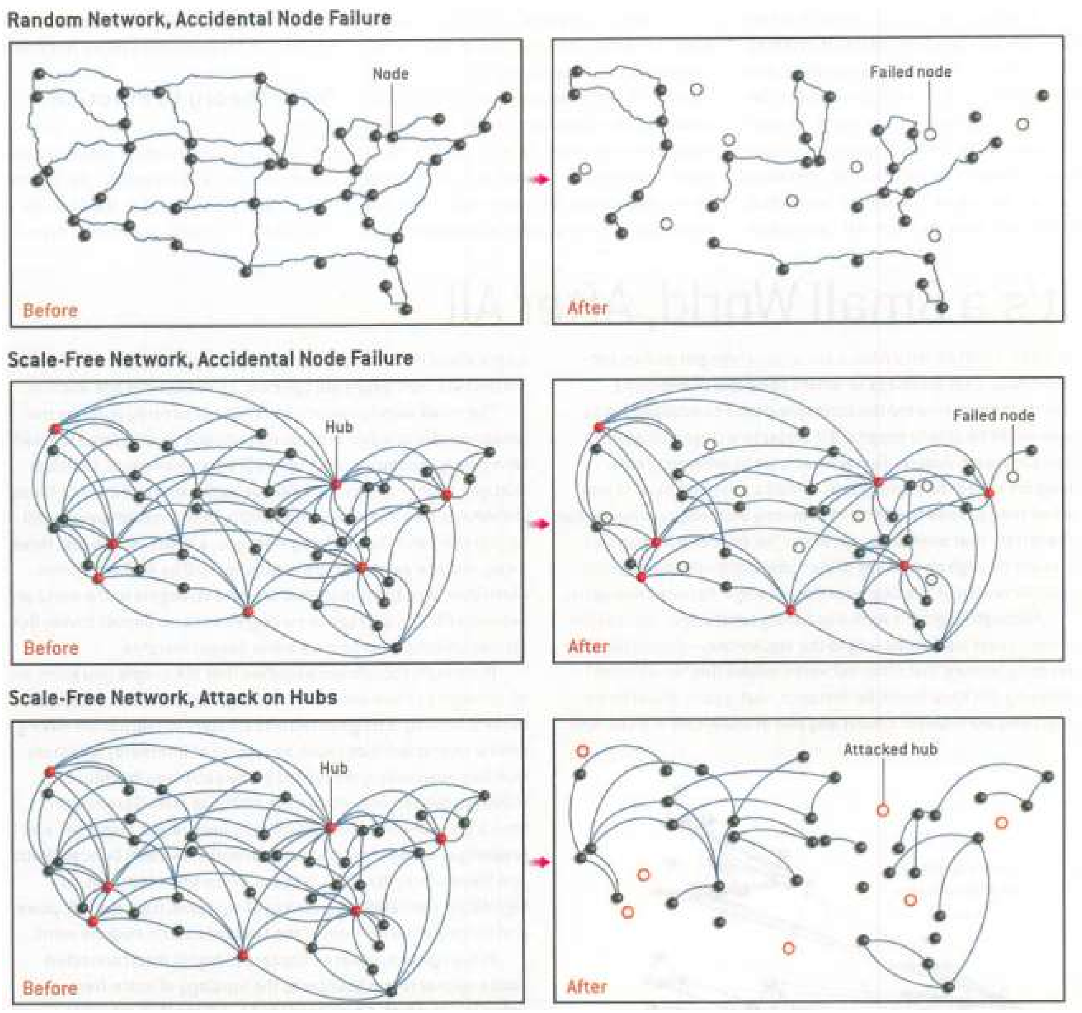
1. Simple correlation does not give connectivity
2. How are we measuring co-expression?
   - Scale-Free Network
     - Most biological networks have been defined as scale-free
     - System is robust to failure at any 1 gene
3. How do we get a robust measurement of connectivity to define modules?
   - Topological Overlap Measure
     - Includes a measure of how many “friends” two genes have in common
     - Protects against spurious correlations among genes

Scale-Free Networks

- Some nodes are "hubs" and have more connection than others
  - Probability of having many connections is low
- Asymptotically follows a power law
  $$P(m) \sim m^{-\gamma}$$
  - Where γ is the tuning parameter
- Gained interest in '99 when looking at the topology of the world wide web
- K-mean clustering assumes a random network

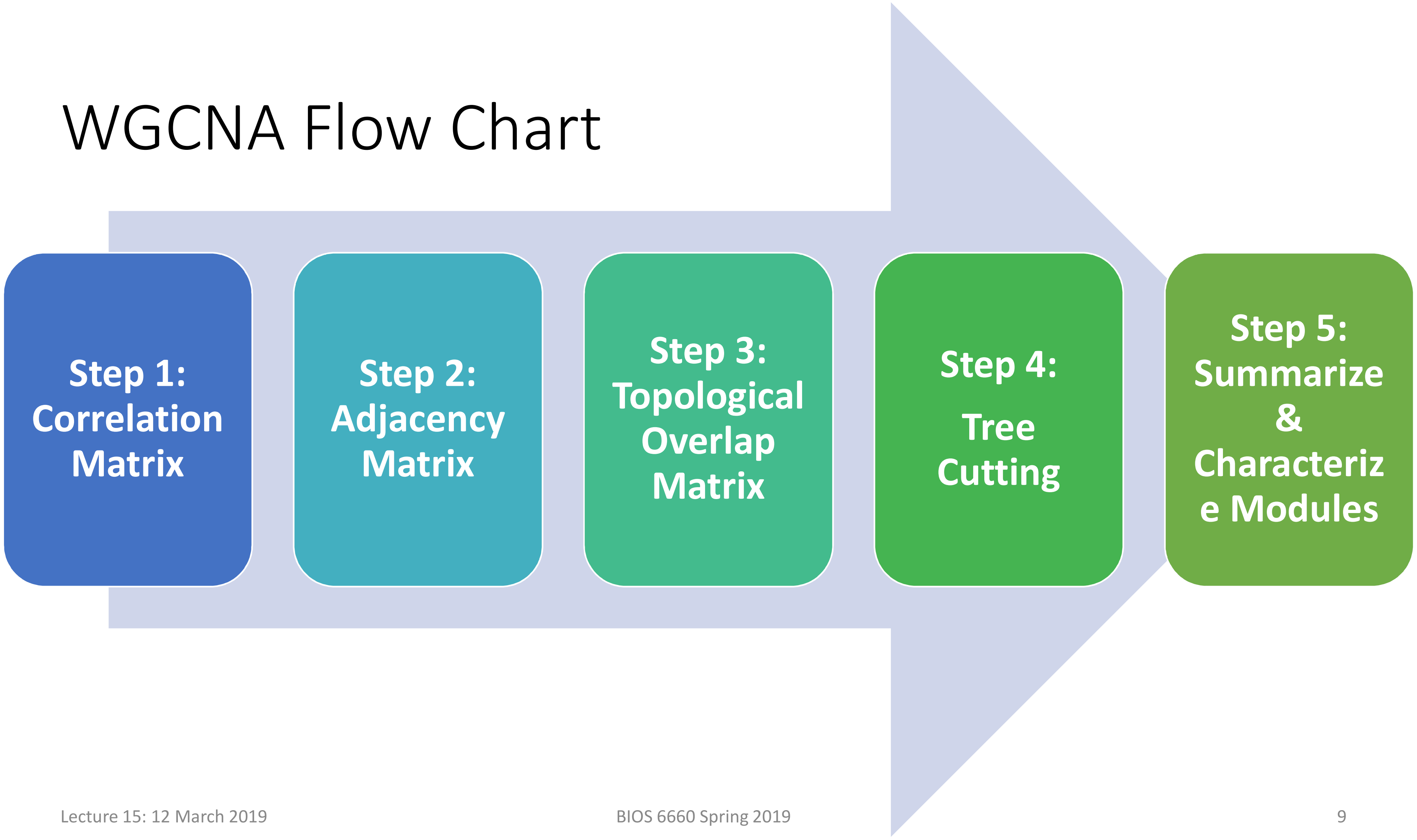

Random Network
Scale-Free Network
Bell Curve Distribution of Node Linkages
Typical node
Number of Nodes
Number of Links
Power Law Distribution of Node Linkages
Number of Nodes
Number of Links
Number of Nodes (log scale)
Number of Links (log scale)

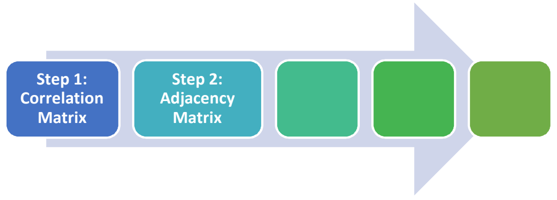
Barabási, AL. Bonabeau, E. **2003**. Scale-Free Networks. *Scientific American* **288** (5): 50–9.


Random Network, Accidental Node Failure
Node
Before
Failed node
After
Scale-Free Network, Accidental Node Failure
Hub
Before
Failed node
After
Scale-Free Network, Attack on Hubs
Hub
Before
Attacked hub
After

Barabási, AL. Bonabeau, E. **2003**. Scale-Free Networks. *Scientific American* **288** (5): 50–9.

WGCNA Flow Chart


Step 1: Correlation Matrix
Step 2: Adjacency Matrix
Step 3: Topological Overlap Matrix
Step 4: Tree Cutting
Step 5: Summarize & Characterize Modules



Constructing an Adjacency Matrix

1. Correlate your genes
   - **corType** options in package: “pearson” (default), “bicor” (non-parametric option)
   - Biweight midcorrelation (bicor) is median based (rather than mean based)
2. Determine your network type
   - **networkType** options in package: “unsigned” (default), “signed”, or “signed hybrid”
   - Signed networks allows you to create module with only positively associated genes
3. Choose a soft-thresholding power (β)
   - Choose a **beta** so you have a scale-free network

`adjacency()`

$$\alpha_{ij} = \left|cor(x_i, x_j)\right|^{\beta}$$

Unsigned Network

$$\alpha_{ij} = \left|\left(0.5 + 0.5 * cor\left(x_i, x_j\right)\right|^{\beta}$$

Signed Network

$$\begin{cases} \alpha_{ij} = cor(x_i, x_j)^{\beta} & if\ cor > 0 \\ \alpha_{ij} = 0 & if\ cor \le 0 \end{cases}$$

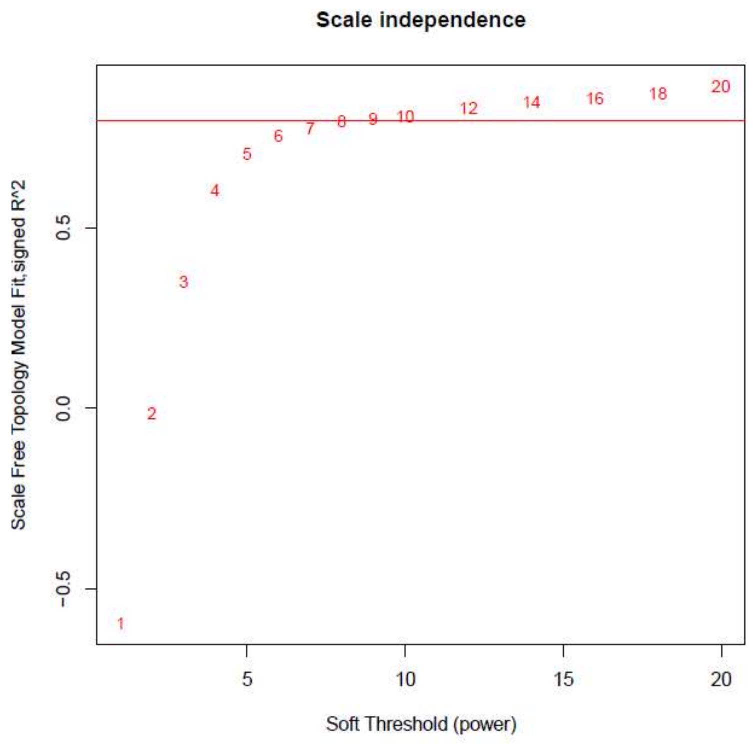
Signed Hybrid Network

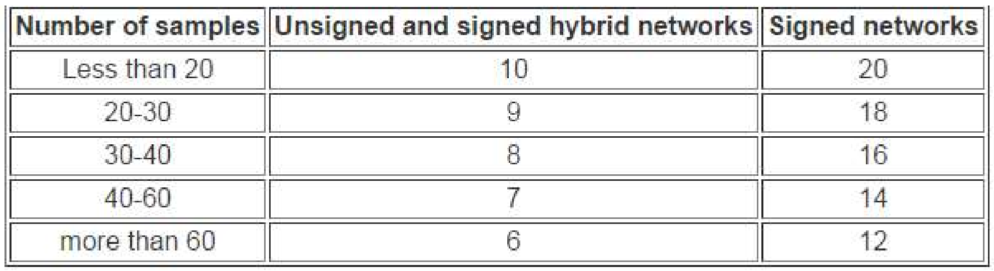
Choosing a β Parameter

- Connectivity (k) is defined as

$$k_i = \sum_{j=1}^{N} \alpha_{ij}$$

- P(k) is the probability of k
- Should have a linear model for log(P(k)) vs. log(k)
- An $R^2 \geq 0.8$ is considered to be scale-free
- Default β based on sample size
- Beware this takes some computational time…



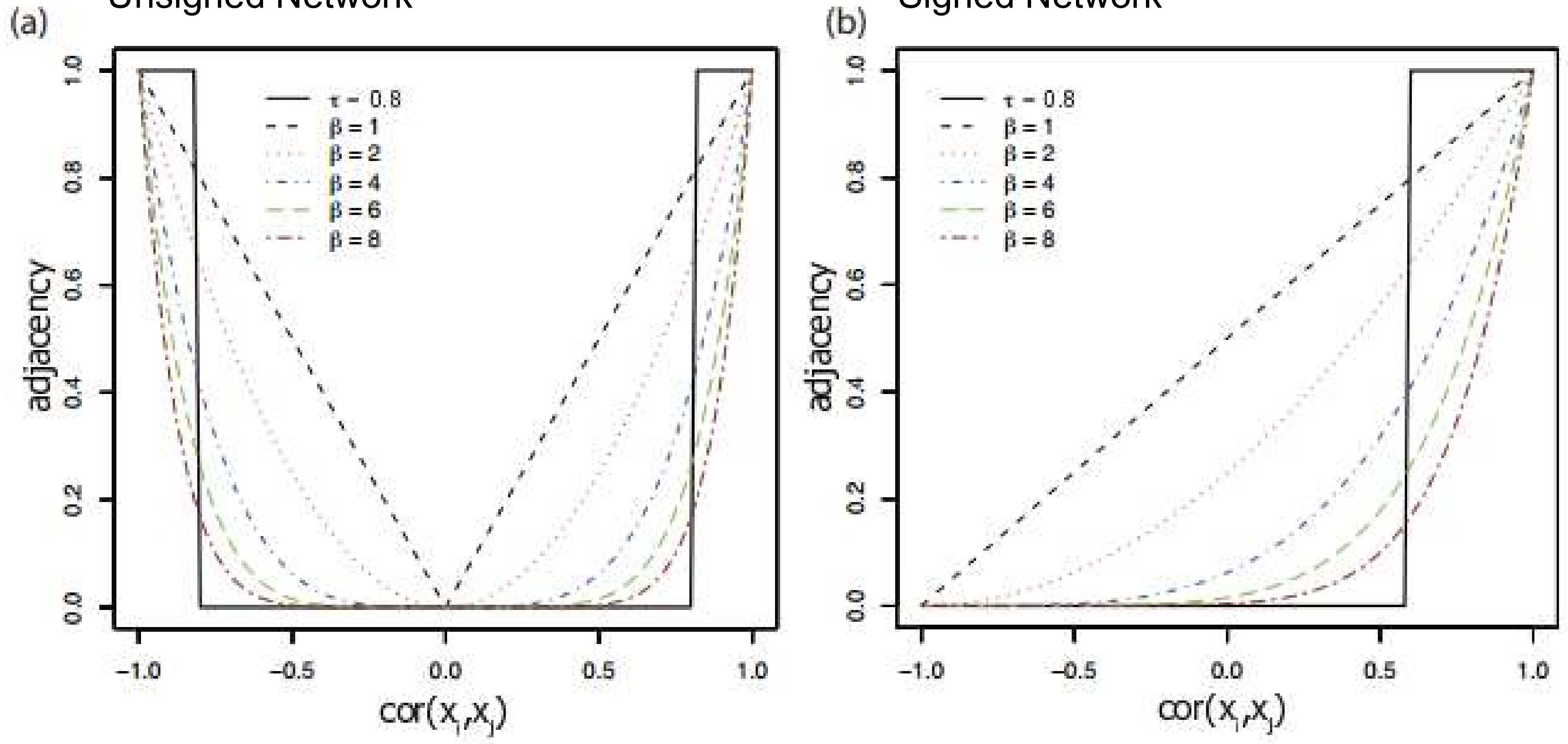


Number of samples	Unsigned and signed hybrid networks	Signed networks
Less than 20	10	20
20-30	9	18
30-40	8	16
40-60	7	14
more than 60	6	12

Power Adjacency vs Step Function

Unsigned Network

Signed Network



https://labs.genetics.ucla.edu/horvath/CoexpressionNetwork/WORKSHOP/2012/Talk1OverviewWGCNAHorvath.pdf

Topological Overlap Matrix (TOM)

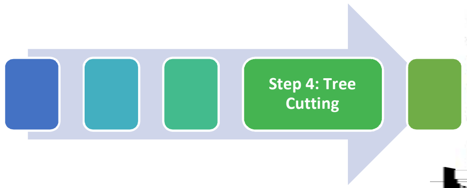
- Transform adjacency matrix to topological overlap matrix (TOM)

$$TOM_{ij} = \frac{\sum_u a_{iu}a_{uj} + a_{ij}}{\min(k_i, k_j) + 1 - a_{ij}}$$

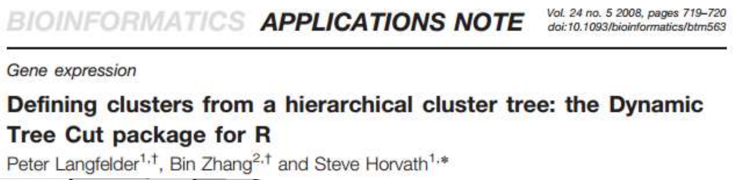
- *k* is connectivity and *a* is adjacency matrix
- Includes the direct relationship between 2 transcripts (*i, j*)
- Also includes their indirect interactions by comparing their relationships with all other transcripts in network
- TOM close to 1 for two genes signifies high connectivity and co-expression
- Can technically be considered a type of adjacency matrix

```
TOMsimilarity()
```

Defining Modules



- Perform hierarchical clustering with the topological overlap dissimilarity (1 – TOM) measure
- Define modules as branches
- 2 types of branch cutting methods:
  1. Static
     - Constant cut height
     - `cutreeStatic()`
  2. Dynamic Hybrid
     - Not all defined branches are at a specific cut height
     - Combines static and partition around medoids (PAM)clustering
     - `cutreeDynamic()`

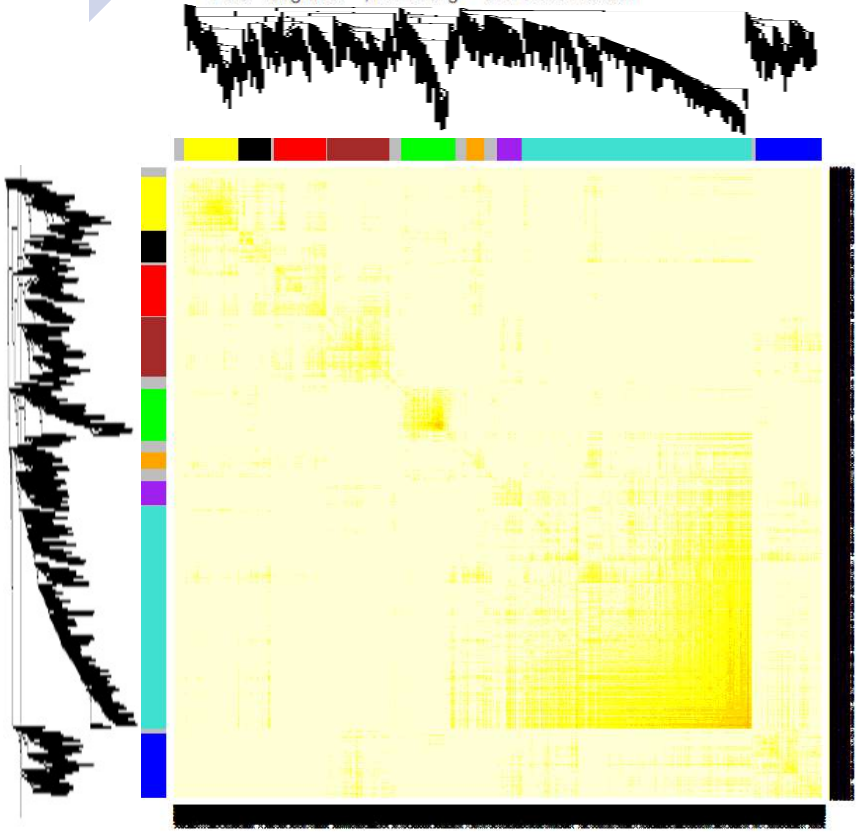


BIOINFORMATICS APPLICATIONS NOTE Vol. 24 no. 5 2008, pages 719–720 doi:10.1093/bioinformatics/btm563

Gene expression

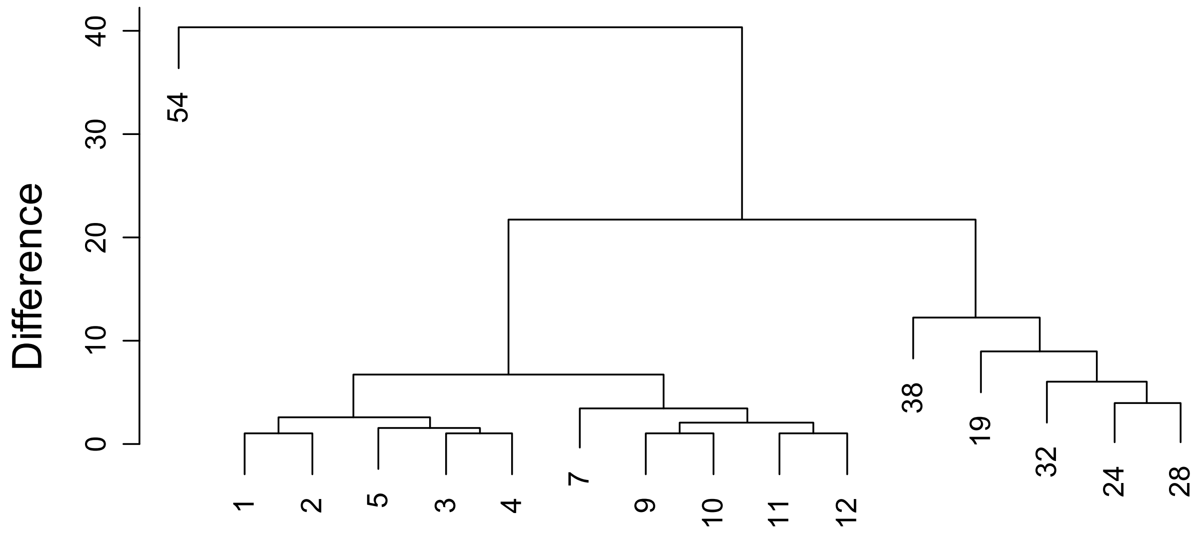
Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R

Peter Langfelder[1,†], Bin Zhang[2,†] and Steve Horvath[1,*]



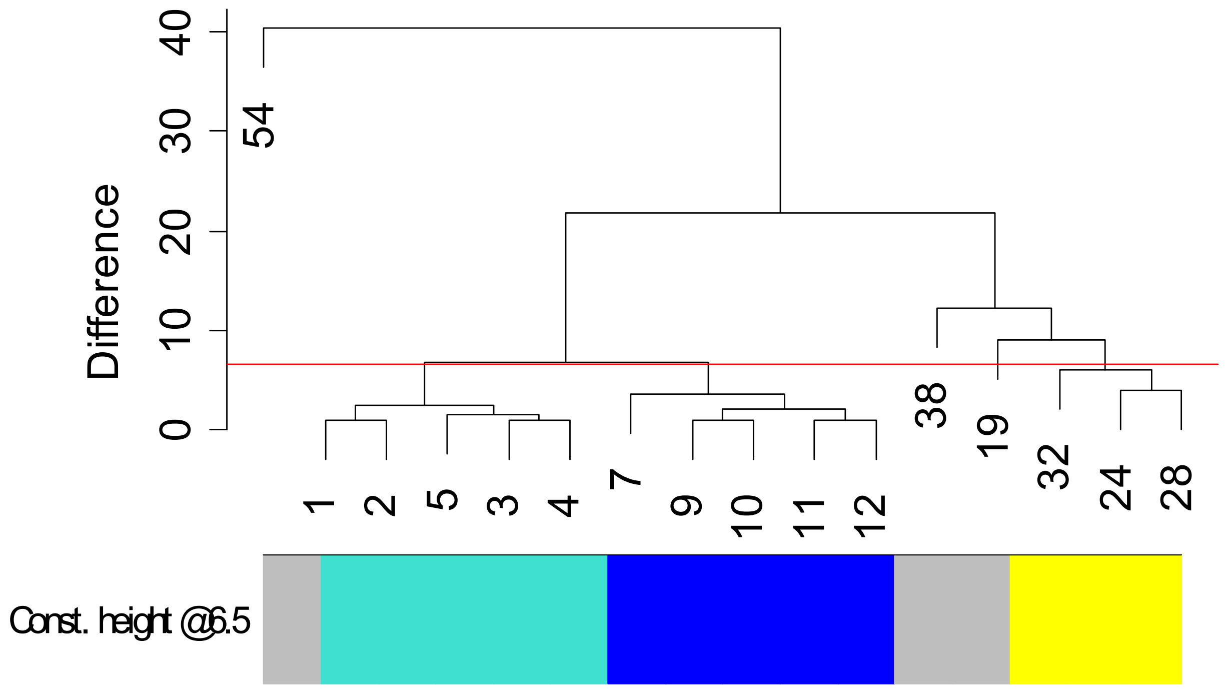
https://labs.genetics.ucla.edu/horvath/CoexpressionNetwork/WORKSHOP/2012/Talk1OverviewWGCNAHorvath.pdf

Tree Cutting Example



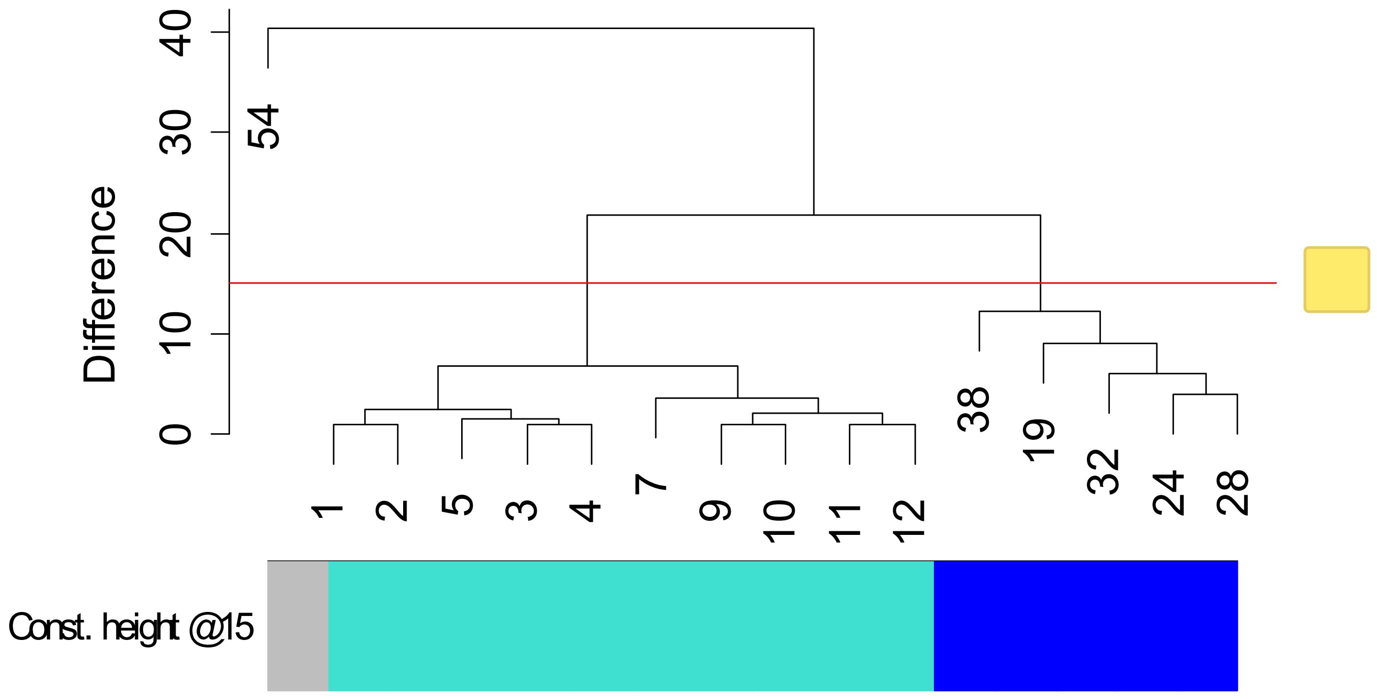
https://labs.genetics.ucla.edu/horvath/CoexpressionNetwork/WORKSHOP/2012/Talk1OverviewWGCNAHorvath.pdf

Tree Cutting Example



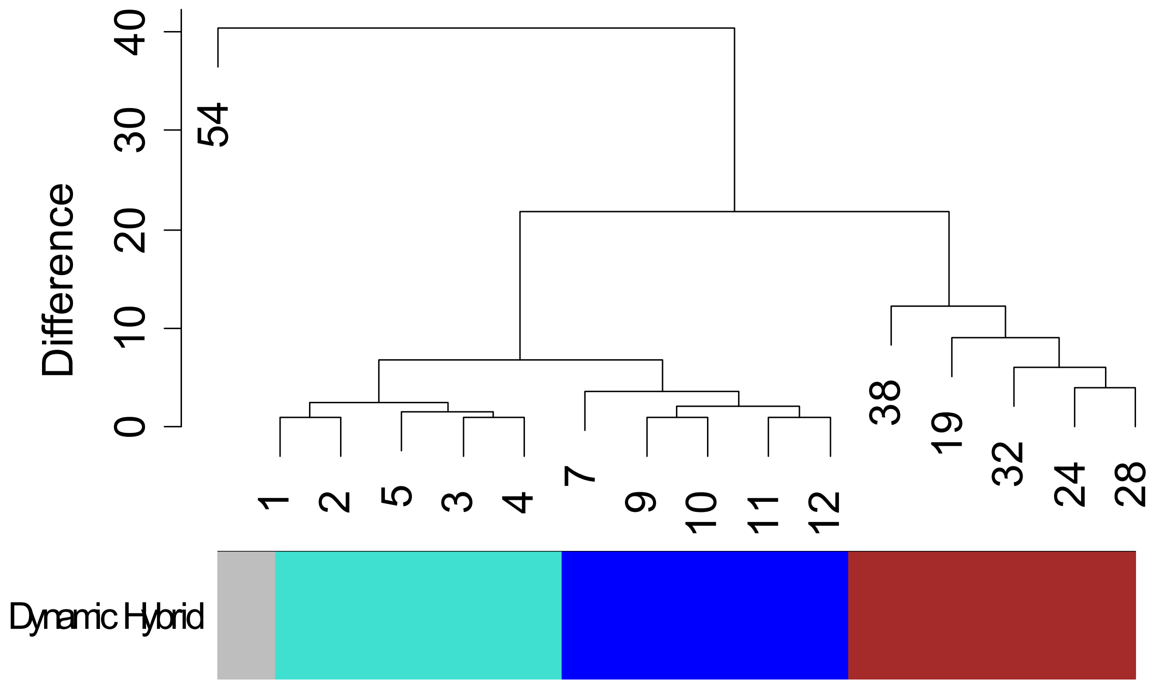
https://labs.genetics.ucla.edu/horvath/CoexpressionNetwork/WORKSHOP/2012/Talk1OverviewWGCNAHorvath.pdf

Tree Cutting Example



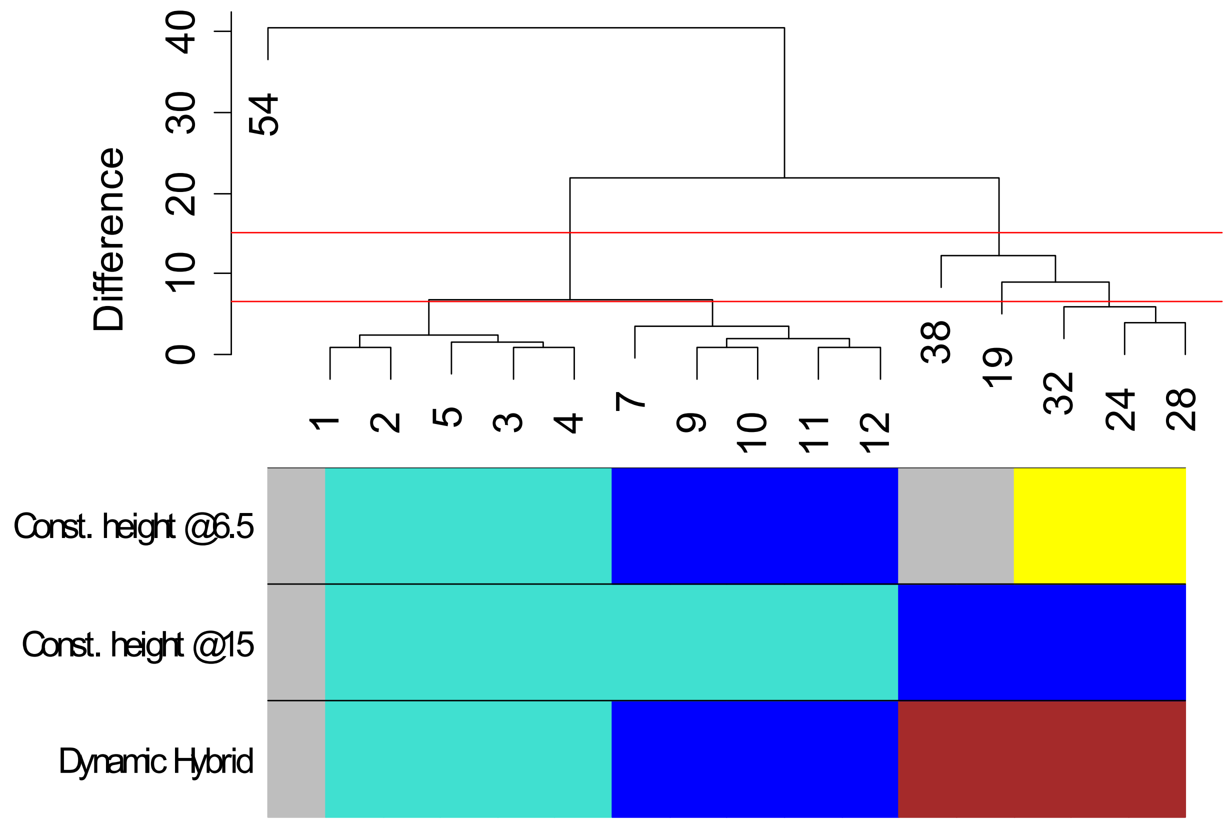
https://labs.genetics.ucla.edu/horvath/CoexpressionNetwork/WORKSHOP/2012/Talk1OverviewWGCNAHorvath.pdf

Tree Cutting Example



https://labs.genetics.ucla.edu/horvath/CoexpressionNetwork/WORKSHOP/2012/Talk1OverviewWGCNAHorvath.pdf

Tree Cutting Example



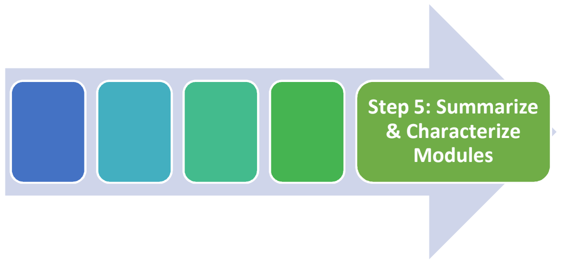
https://labs.genetics.ucla.edu/horvath/CoexpressionNetwork/WORKSHOP/2012/Talk1OverviewWGCNAHorvath.pdf

Settings for Defining Modules blockwiseModules()

- **mergeCutHeight** (default = 0.15)
  - Static cut height to define modules
- **minModuleSize** (default = 20)
  - Minimum size of a module you will accept, we used 5 alcohol related traits very complex and we are expecting smaller modules
  - If you are doing enrichment analysis you want a minimum of 20 genes
- **deepSplit** (default = 2)
  - Integer value between 0 (least sensitive) and 4 (most sensitive) for a simplified control on how the module detection should be module splitting
  - We used 4 because we wanted smaller modules
- **pamRespectsDendro** (default = TRUE)
  - Only use if using dynamic cut
  - where PAM respects the dendrogram and it won't group objects into clusters unless they are from the same branch

Example Summary Results Whole Brain Network

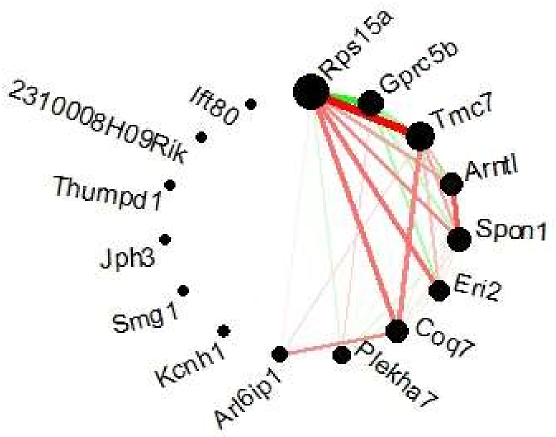
Number of Probesets Placed Into Modules	28,867
Number of Modules Identified	499
Minimum Module Size	5
Maximum Module Size	2,881
Median Module Size	9



Summarizing Modules

1. HUB gene
   - Most connected gene in your module
   - Example: for hotpink3 hub gene is Rps15a

2. Eigengene
   - 1st principal component of module
   - Check to see how much variance the eigengene explains



Hotpink3 module

Characterization of Modules

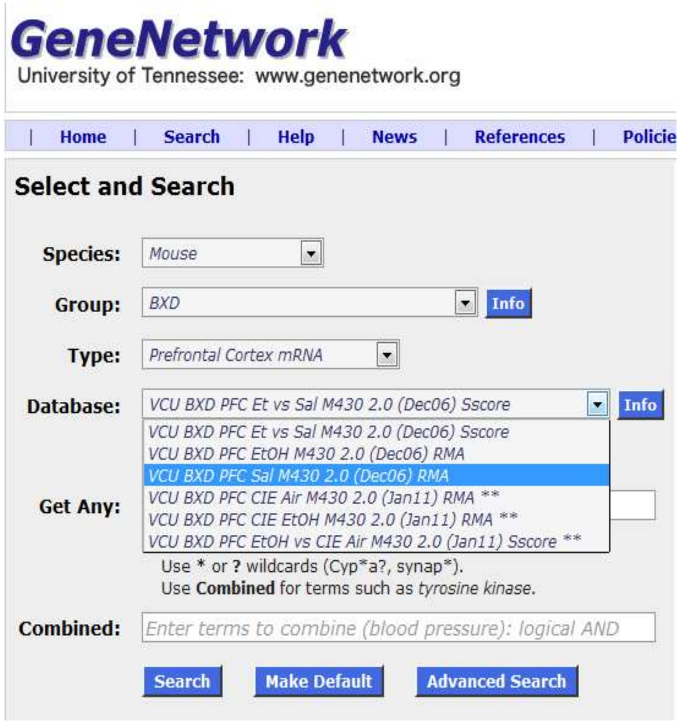
- Statistical Analysis of eigengene with physiological or behavioral trait
- Gene Ontology and Pathway Enrichment
- Cell-type specific expression
- Source of expression control
    - Transcription factors
    - microRNA
    - Common eQTL
    - Module eigengene QTL (E-QTL)

Other Useful WGCNA functions

Function	What it does
chooseTopHubInEachModule()	identifies the hub gene in each module
moduleEigengenes()	calculates the module's eigengenes
propVarExplained()	calculates the proportion of variance explained by eigengenes
exportNetworkToCytoscape()	exports a network in edge and node list files in a format suitable for importing to Cytoscape
plotNetworkHeatmap()	network heatmap plot
plotDendroAndColors()	dendrogram plot with color annotation of objects
plotMEpairs()	pairwise scatter plots of eigengenes
circlePlot()*	visualize connectivity within a module
GOenrichmentAnalysis()	calculation of GO enrichment (experimental)
networkScreening()	identifies genes related to a trait by taking into account both the network and gene-centric info (experimental)

Comparing Networks

- Example: Whole Brain vs Brain Region
  - Treatment vs control network
- 6 brain regions:
  - Cerebellum (CER)
  - Hippocampus (HIP)
  - Nucleus Accumbens (NA)
  - Prefrontal Cortex (PFC)
  - Striatum (STR)
  - Ventral Tegmental Area (VTA)
- Created a network for each area using same probesets and same parameters as whole brain
  - Need to have same nodes (i.e. genes) going into network


GeneNetwork
University of Tennessee: www.genenetwork.org

| Home | Search | Help | News | References | Policie

Select and Search

Species: Mouse
Group: BXD
Type: Prefrontal Cortex mRNA
Database: VCU BXD PFC Et vs Sal M430 2.0 (Dec06) Sscore

VCU BXD PFC Et vs Sal M430 2.0 (Dec06) Sscore
VCU BXD PFC EtOH M430 2.0 (Dec06) RMA
VCU BXD PFC Sal M430 2.0 (Dec06) RMA
VCU BXD PFC CIE Air M430 2.0 (Jan11) RMA **
VCU BXD PFC CIE EtOH M430 2.0 (Jan11) RMA **
VCU BXD PFC EtOH vs CIE Air M430 2.0 (Jan11) Sscore **

Get Any:
Use * or ? wildcards (Cyp*a?, synap*).
Use **Combined** for terms such as *tyrosine kinase*.

Combined: Enter terms to combine (blood pressure): logical AND

Z summary statistics

- Composite statistic telling how well a module is preserved
  - 4 statistics related to density
    - Highly connected nodes maintain that level of connectivity
  - 3 statistics related to connectivity
    - Connectivity pattern between specific genes is maintained
- Bias: Larger modules tend to have larger Z summary scores

`modulePreservation()`

OPEN ACCESS Freely available online PLoS COMPUTATIONAL BIOLOGY

Is My Network Module Preserved and Reproducible?

Peter Langfelder[1], Rui Luo[1], Michael C. Oldham[1], Steve Horvath[2*]

$$Z_{density} = \text{median}(Z_{meanCor}, Z_{meanAdj}, Z_{propVarExpl}, Z_{meanKME})$$

$$Z_{connectivity} = median(Z_{cor.kIM}, Z_{cor.kME}, Z_{cor.cor})$$

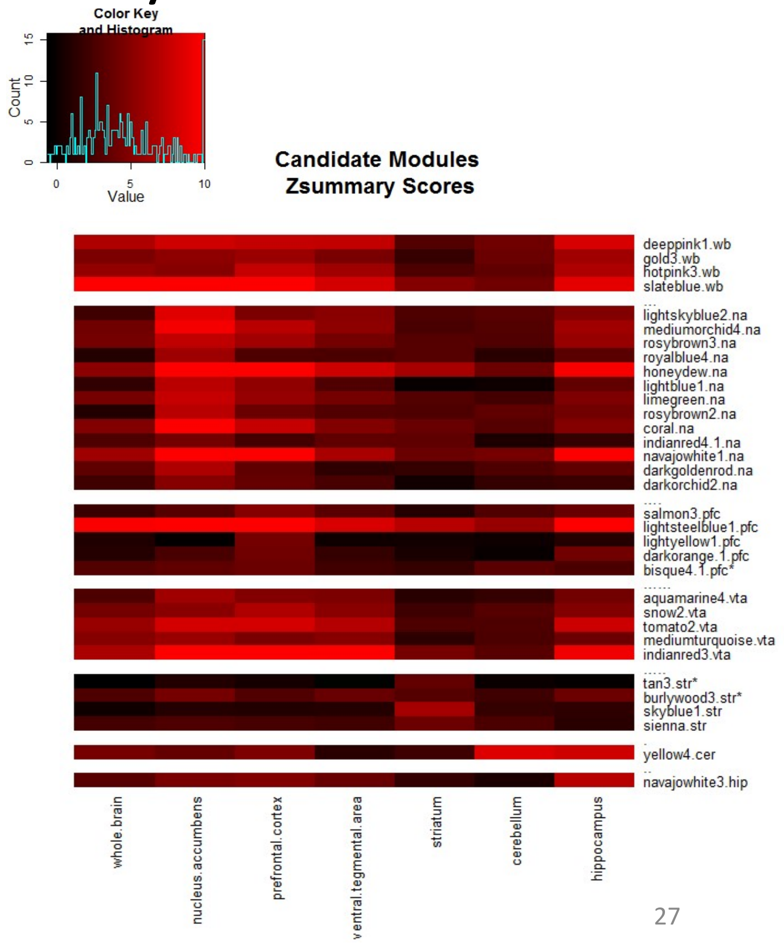
$$Z_{summary} = \frac{Z_{density} + Z_{connectivity}}{2}$$

Z summary	Interpretation
< 2	Not Preserved
2-10	Moderately Preserved
> 10	Strongly Preserved

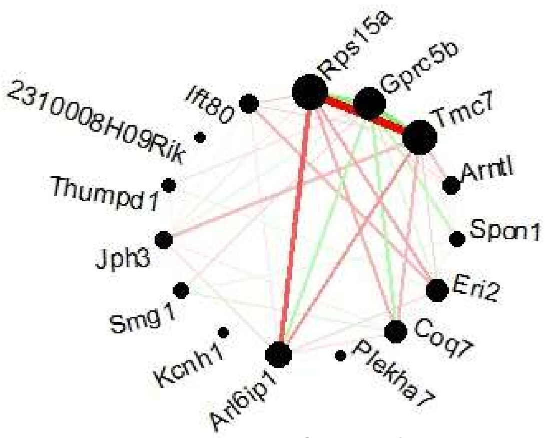
Methods to Generate Z summary Scores

1. Reproducibility WITHIN dataset
   - Verify candidate modules were of high quality and not generated by chance
   - Used 100 bootstrap samples, calculated Z summary for each bootstrap sample compared to original dataset
2. Reproducibility BETWEEN datasets
   - Determine if candidate modules are preserved in other brain regions/whole brain
   - Calculated Z summary by setting the dataset module originated from as reference and comparison region as test set
   - **Note:** you need to have same gene identifiers!

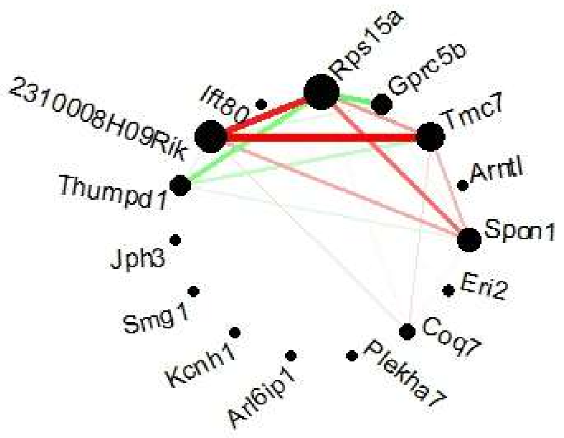
- Mean Within Z scores range: 3.05 to 16.15
- Only 3 modules have lower 95% CI bound < 2 (noted with *)
- Between Z scores range: -0.67 to 17.05



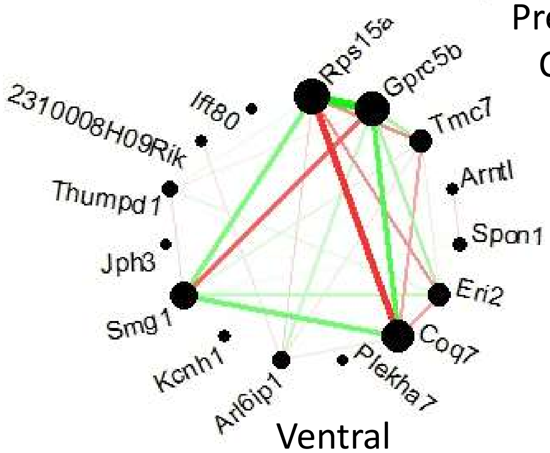
Hotpink3 from Whole Brain



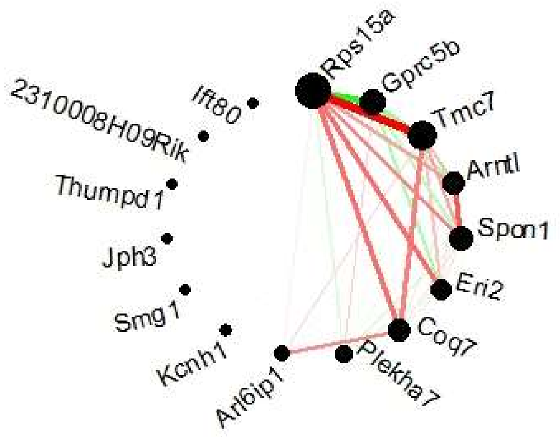
Prefrontal Cortex



Striatum

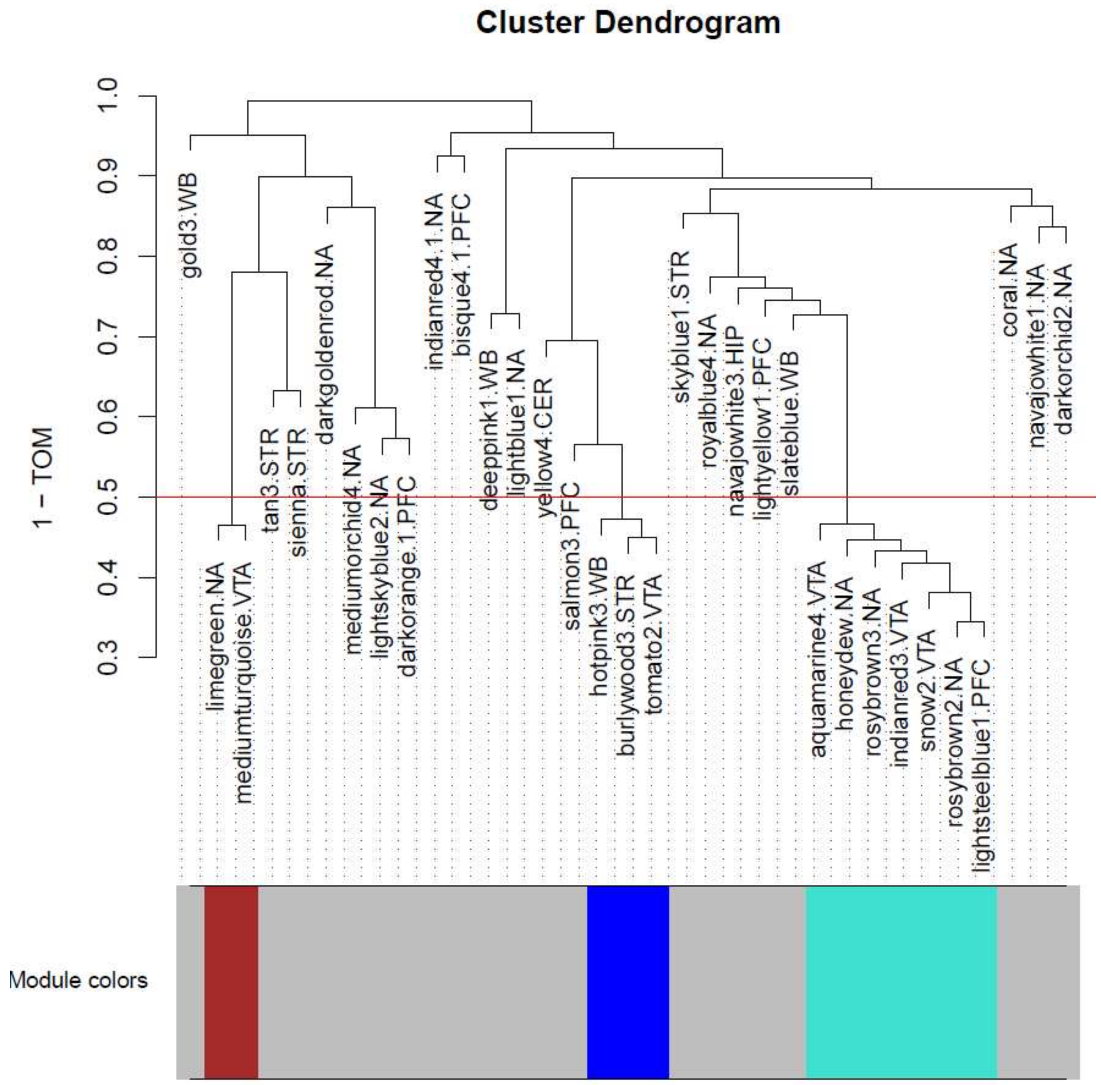


Ventral Tegmental Area



Whole Brain

Eigengene Network



Modules Can Be Useful To…

- Add biological context to unannotated or under annotated genes
  - Especially useful in organisms with little annotation (e.g. rats)
- Explaining biomarkers
  - Make sense of biomarker given the other genes within module
- How do module structure changes in different environments
  - Treatment
  - Tissue
  - Exposure
- Finding target genes for pharmaceutical therapies

Pro and Con list for WGCNA

PRO's:

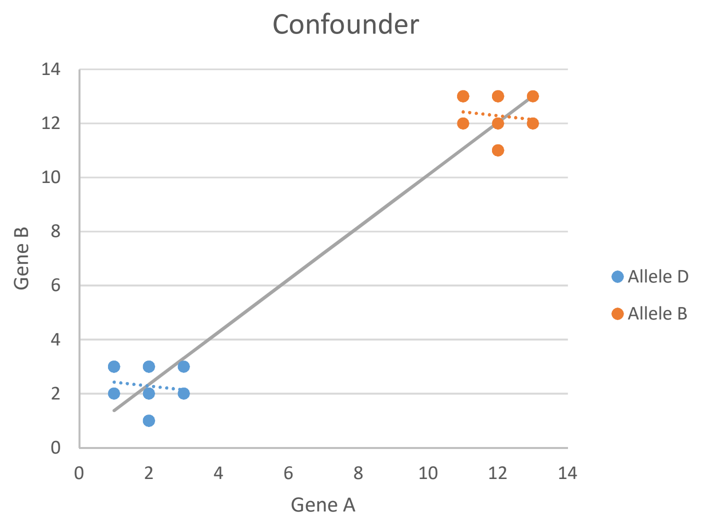
- Easy to use and understand
  - Peter Langfelder normally answers all questions in online forums directly.
- Robust network
  - Scale-free
  - Indirect & Direct Interactions
- Accepted by community
  - Most common form for network analyses I've seen for gene expression data (both microarray and sequencing)

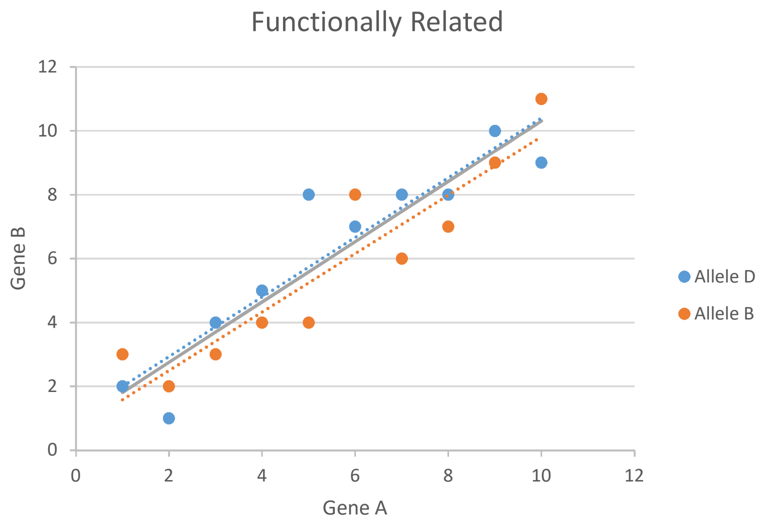
CON's:

- Missing non-linear relationships
- How do we choose which genes to include in the module?
  - I find the tree cutting the most difficult evaluate
- Not easy to perform differential networks
  - How are my edges changing given a specific circumstance?
- Are these modules truly clustered together functionally?

Something to Consider: Functional Results or Driven by Confounder

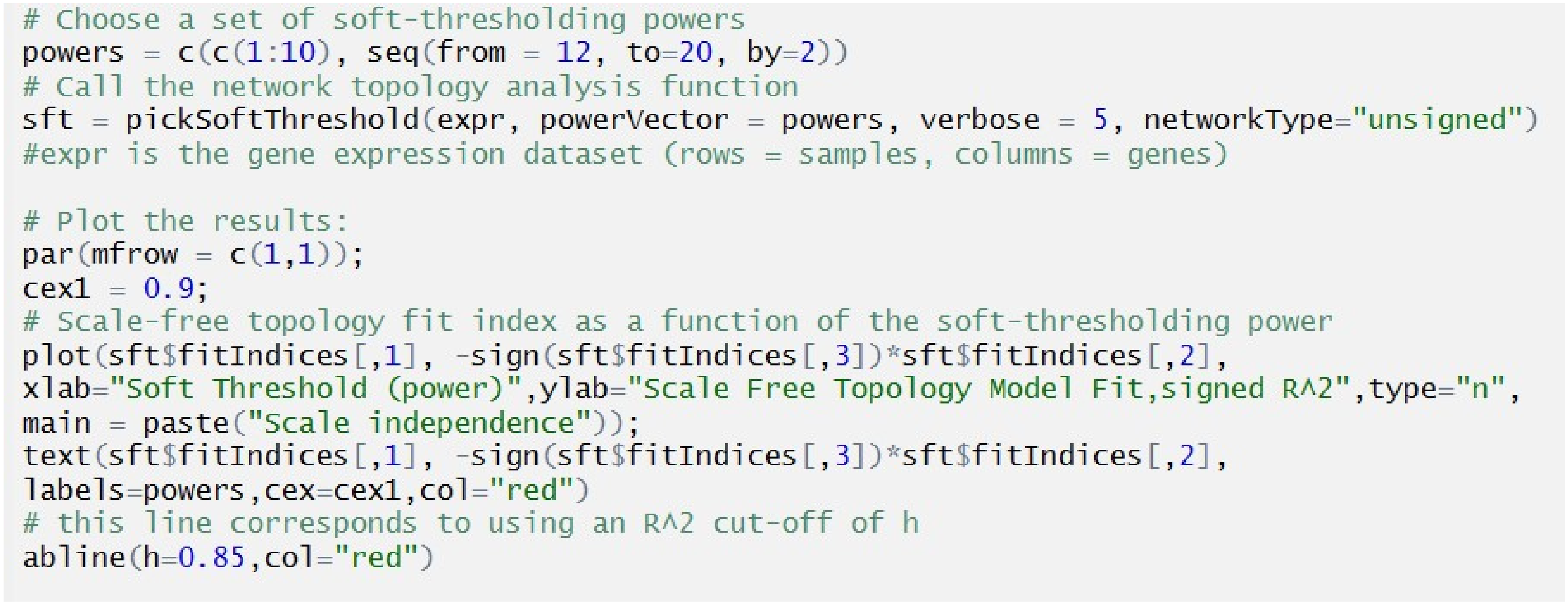
- Some modules have genes physically in close proximity to one another and have extremely strong cis-QTL's
- Partial correlation adjusting for allele in the cis-QTL
- Our candidate modules stayed highly correlated
- Confounding issue: many genes functionally related are also physically located near each other





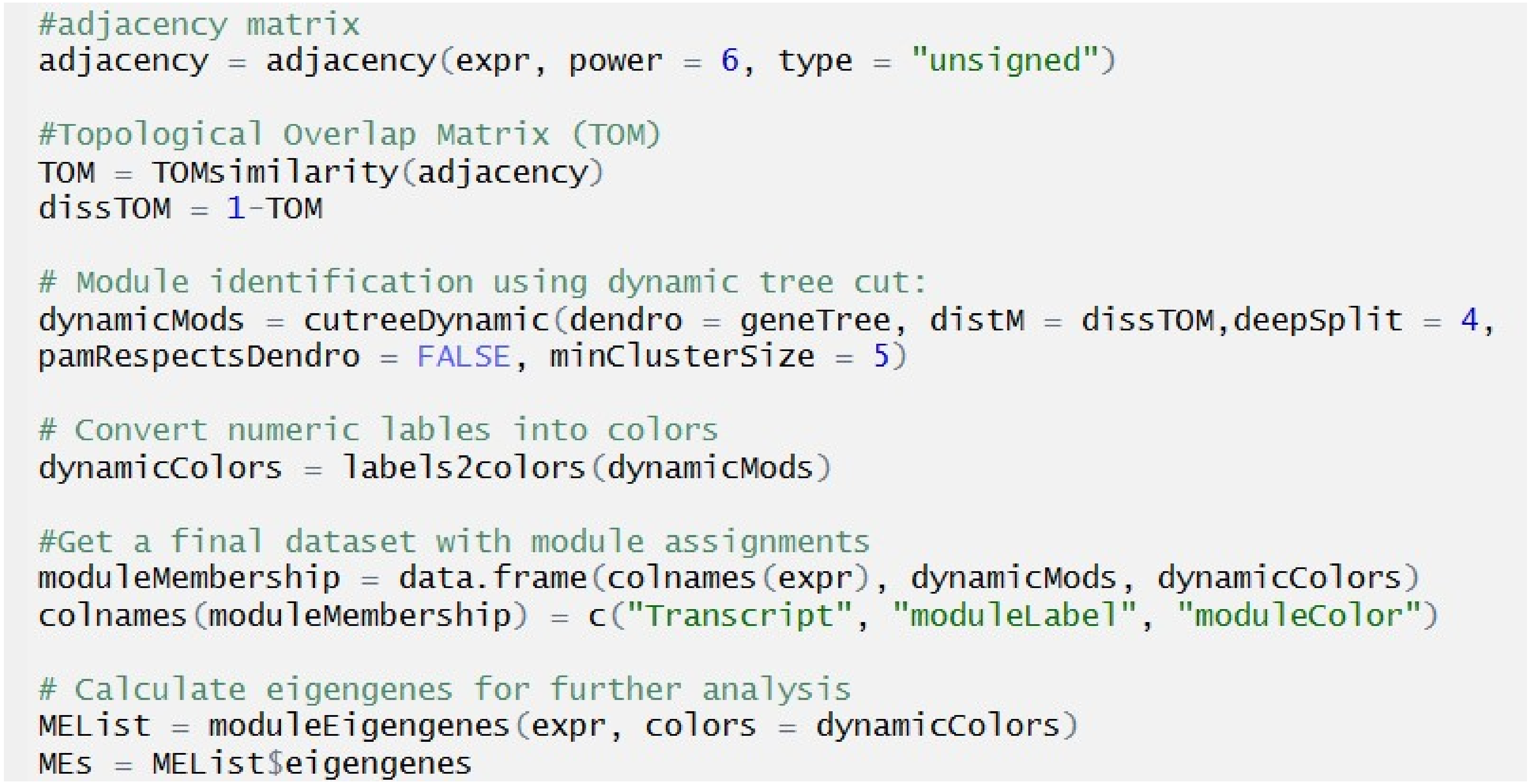
WGCNA Code

library(WGCNA)



```r
# Choose a set of soft-thresholding powers
powers = c(c(1:10), seq(from = 12, to=20, by=2))
# Call the network topology analysis function
sft = pickSoftThreshold(expr, powerVector = powers, verbose = 5, networkType="unsigned")
#expr is the gene expression dataset (rows = samples, columns = genes)

# Plot the results:
par(mfrow = c(1,1));
cex1 = 0.9;
# Scale-free topology fit index as a function of the soft-thresholding power
plot(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
xlab="Soft Threshold (power)",ylab="Scale Free Topology Model Fit,signed R^2",type="n",
main = paste("Scale independence"));
text(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
labels=powers,cex=cex1,col="red")
# this line corresponds to using an R^2 cut-off of h
abline(h=0.85,col="red")
```

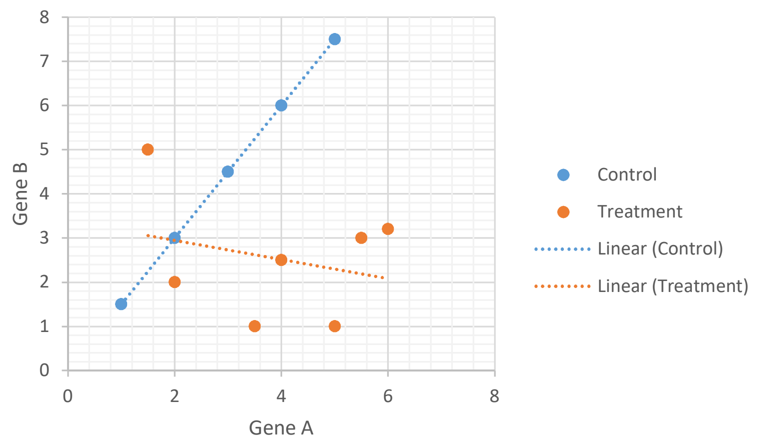
WGCNA Code Continued

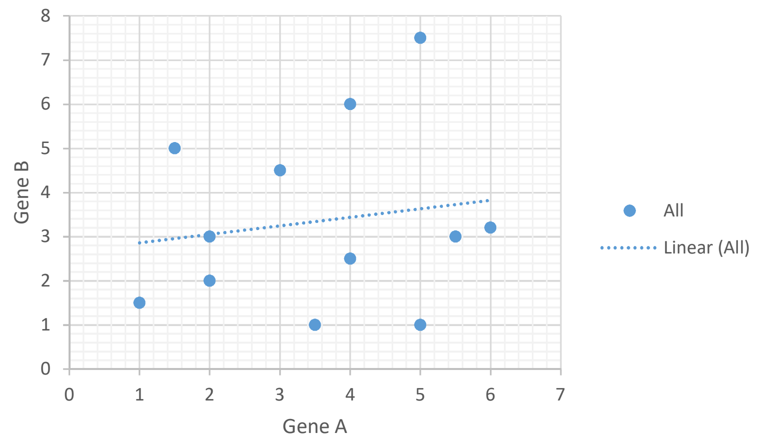


```
#adjacency matrix
adjacency = adjacency(expr, power = 6, type = "unsigned")

#Topological Overlap Matrix (TOM)
TOM = TOMsimilarity(adjacency)
dissTOM = 1-TOM

# Module identification using dynamic tree cut:
dynamicMods = cutreeDynamic(dendro = geneTree, distM = dissTOM,deepSplit = 4,
pamRespectsDendro = FALSE, minClusterSize = 5)

# Convert numeric lables into colors
dynamicColors = labels2colors(dynamicMods)

#Get a final dataset with module assignments
moduleMembership = data.frame(colnames(expr), dynamicMods, dynamicColors)
colnames(moduleMembership) = c("Transcript", "moduleLabel", "moduleColor")

# Calculate eigengenes for further analysis
MEList = moduleEigengenes(expr, colors = dynamicColors)
MEs = MEList$eigengenes
```

Differential Co-expression

- What modules are disrupted by a specific treatment or exposure?
- R/Discordant
  - Looks at the edges and what edges change over the 2-groups
  - Perform WGCNA on posterior probabilities from discordant (Liz Litkowski)
- R/DCGL: Differential Co-expression Analysis and Differential Regulation Analysis of Gene Expression Microarray Data





Prediction Modeling

- Want to know best predictor for an outcome
    - What gene (or combination of genes) predicts disease the best?
- Could just plug into an ordinary least squares (OLS) regression and perform backward selection
- Major OLS concerns
    - Collinearity (expected in a biological system)
    - Using OLS, variance explodes with collinearity and makes your estimates unstable
    - Help reduce this inflated variance
- Common Methods
    - Ridge Regression
    - LASSO
    - Random Forest

Overview of Ridge Regression & Lasso

- LASSO (least absolute shrinkage and selection operator) regression
- Reign in estimates by applying a penalty
    - Estimates need to be in the circle (ridge regression) or diamond (LASSO)
    - LASSO allows beta coefficients to be exactly 0 (feature selection)
    - Ridge: L2 penalty
    - LASSO: L1 penalty

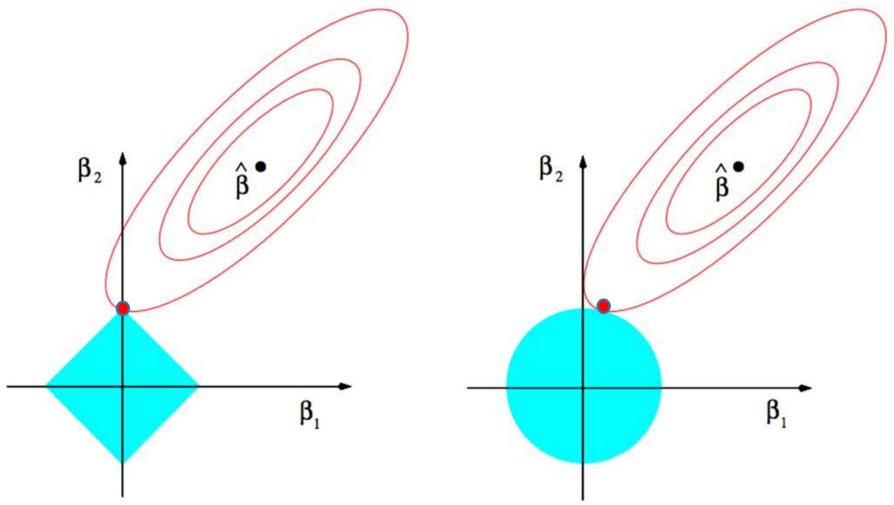


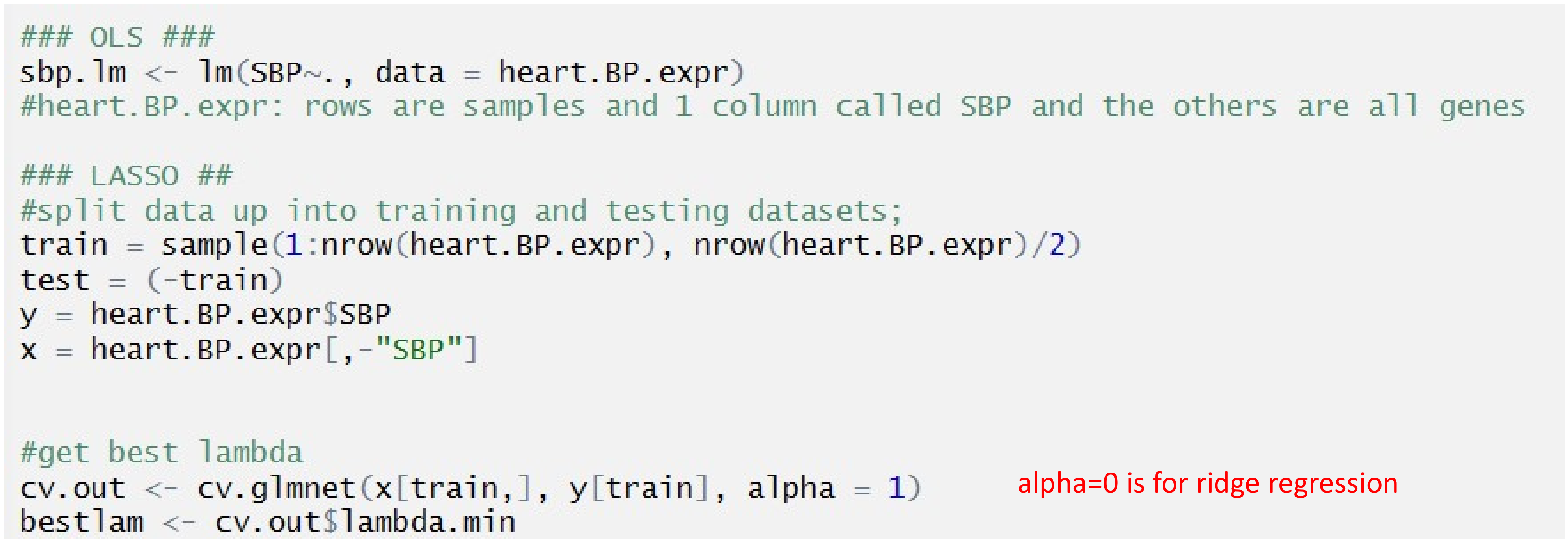
FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions $|\beta_1| + |\beta_2| \leq t$ and $\beta_1^2 + \beta_2^2 \leq t^2$, respectively, while the red ellipses are the contours of the least squares error function.*

https://bmcgenomics.biomedcentral.com/articles/10.1186/1471-2164-13-452

OLS: $(X\beta - y)'(X\beta - y)$
Ridge: $(X\beta - y)'(X\beta - y) + \lambda \|\beta\|_2$
LASSO: $(X\beta - y)'(X\beta - y) + \lambda \|\beta\|_1$

LASSO Code

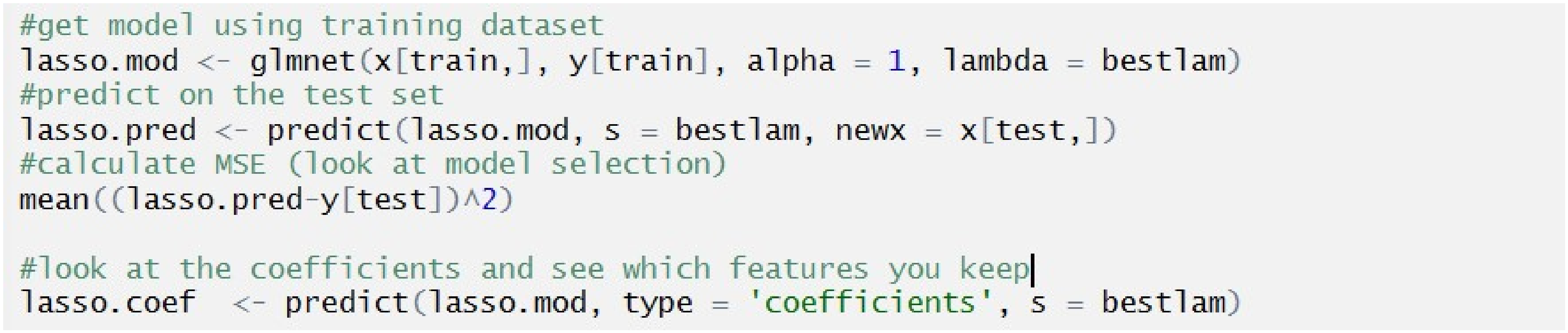
library(glmnet)



```
### OLS ###
sbp.lm <- lm(SBP~., data = heart.BP.expr)
#heart.BP.expr: rows are samples and 1 column called SBP and the others are all genes

### LASSO ##
#split data up into training and testing datasets;
train = sample(1:nrow(heart.BP.expr), nrow(heart.BP.expr)/2)
test = (-train)
y = heart.BP.expr$SBP
x = heart.BP.expr[,-"SBP"]


#get best lambda
cv.out <- cv.glmnet(x[train,], y[train], alpha = 1)
bestlam <- cv.out$lambda.min
```

alpha=0 is for ridge regression

LASSO Code



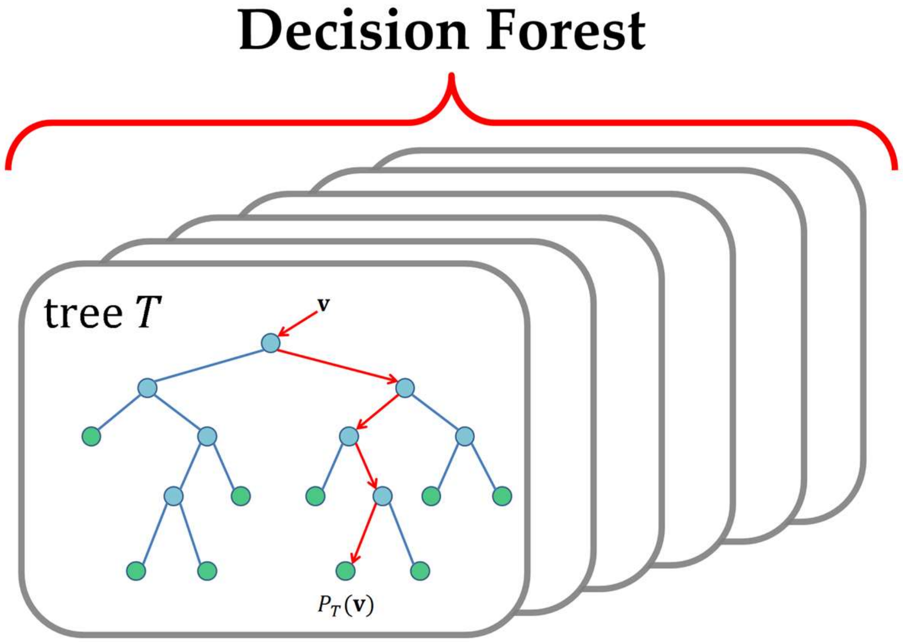
```
#get model using training dataset
lasso.mod <- glmnet(x[train,], y[train], alpha = 1, lambda = bestlam)
#predict on the test set
lasso.pred <- predict(lasso.mod, s = bestlam, newx = x[test,])
#calculate MSE (look at model selection)
mean((lasso.pred-y[test])^2)

#look at the coefficients and see which features you keep
lasso.coef  <- predict(lasso.mod, type = 'coefficients', s = bestlam)
```

Normally plug the features you are keeping back into an OLS to get estimates now that you are selecting a subset of your features you won't have the collinearity issues.

Random Forest

- Just like a forest is made up of trees, random forest is made up may decision trees

Source: BBC



Source: dimensionless.in

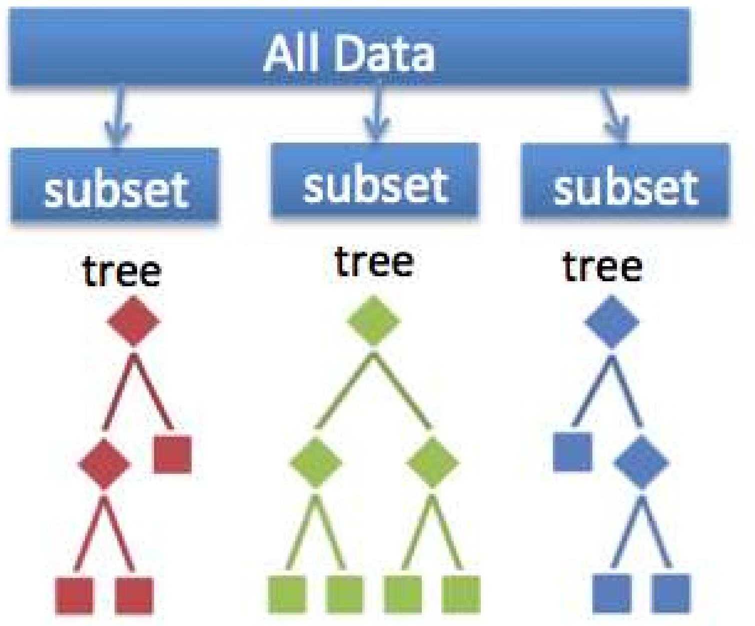
Constructing a Single Tree

1. Random Sample Selection: Take 2/3 samples with replacement
2. Random Predictor Selection: Take sqrt(m) predictors
   - 20K genes ~ 141 predictors for each tree
3. Start with which predictor separates out the best and move on down
   - What predictor explains the most variance?
4. Keep moving down the predictors till you get a formal tree

https://www.youtube.com/watch?v=LDRbO9a6XPU

Random Forest

- Collection of these decision trees
- Avoid overfitting due to the "randomness" in each of your trees
- Aggregate all the trees to get the importance of each predictor
  - Try to get a minimum amount of leaves (node with no children)
  - Like taking a poll of all the trees
- R/randomforest
- https://www.r-bloggers.com/how-to-implement-random-forests-in-r/



Source: towardsdatascience.com