Yuyue Wang
yw1384
12-18-2016
TCGA project report

Introduction

The aim of this project is to study the relationship between different types of cancer on the level of gene expression. To do that, I created cancer gene co-expression modules using data of from The Cancer Genome Atlas (TCGA) that was filtered by Catalog of Somatic Mutations in Cancer(COSMIC). There were in total two types of cancers whose related genes were studied: lung cancer and colorectal cancer. Each individual data file was from one patient who was diagnosed with one of the two cancers. R package "WGCNA" (Weighted Gene Co-Expression Network Analysis)and "cluster" were used to calculate the correlation between genes and generate clusters. The results showed two modules by different ways of tree cutting although they were not that salient due to the small size of data. Further comparison between different cutting method will be discussed later in the report.

Methods and Results

[Data Collection]

I chose to study Lung Squamous Cell Carcinoma(TCGA-LUSC) and Colon Adenocarcinoma(TCGA-COAD) for this project. These cancer types were chosen so that their cancer related gene numbers differ less than a fold of 5. Firstly, the symbols of cancer related genes of these two types of cancer were collected respectively from COSMIC(http://cancer.sanger.ac.uk/cosmic). Then the gene symbols of those cancer related genes were inverted to gene IDs using BioMart(http://www.ensembl.org/biomart/martview/bc2409af2c8ab01273ef375f3e7e9668), which would match with the gene IDs in the patient's data files later. Next, I downloaded the gene expression level data of 10 patients(5 for each cancer) from TCGA(https://gdc-portal.nci.nih.gov/projects/TCGA-LUSC and https://gdc-portal.nci.nih.gov/projects/TCGA-COAD). The patients were chosen so that a reasonable number of cutHeight can be applied to generate clusters in the Method and Results part..Note that only those files that were listed as "open" were accessible.

[Matrix construction]

Using Python code(see attachment), I filtered all gene expression levels data of the 10 patients by the cancer related genes, only keeping the genes that were present in either of the two cancer related gene lists. In other word, calculate the union of the two types of cancer genes. After calculation, 49 cancer related genes were included in the matrix. So a 49*10 matrix was constructed to store the filtered data where each row showed one cancer related gene, and each column means the data of one patient. (See the first several rows in the figure below)

Gene_name	lung1	lung2	lung3	lung4	lung5	colon1	colon2	colon3	colon4	colon5					
ENSG00000273686															
ENSG00000146374	7367.54559328	0.651344164531	0.33258650813	0.169500216136	3644.93677904	8404.39045357	38224.7388773	2963.54146012	1.63923681167	0.127384170761					
ENSG00000145675	46798.2906807	5.04653163841	2.11257329688	5.86224034666	126061.759296	61326.7710143	110155.16605	124618.0679	4.72391462932	5.35655379043					
ENSG00000196090	167215.212691	0.770479119765	7.54844649294	1.49013353565	32043.8678699	589.785041046	310.366418788	727.891393164	0.0133098112303	0.0312875128525					
ENSG00000166710	2969449.04223	504.861855213	134.047176977	753.895200903	16211780.7751	20127240.114	18333550.6374	14057961.9932	786.219395505	604.264139779					
ENSG00000135679	189431.318193	8.24841568061	8.55132823415	8.51412668163	183087.987678	82529.9365123	142094.143461	93281.8997783	6.09359167714	4.00960729256					
ENSG00000100644	2464701.94062	75.2635458806	111.261830909	73.6664104249	1584124.28526	229704.155867	365560.621576	279449.553688	15.6767696885	12.0117940461					
ENSG00000141646	67000.9179584	4.91675072677	3.02456239505	5.41578103367	116461.087348	168691.645052	87853.3737905	68994.6758762	3.76752042202	2.96565095907					
ENSG00000164754	1320307.52584	28.2205344552	59.6014594167	50.4098341688	1084014.30261	433334.319585	751417.036618	941805.531132	32.223907959	40.4823479665					
ENSG00000100393	583282.006399	21.4984557906	26.3305768939	12.9919724998	279379.693292	391833.614068	148780.465125	210909.687704	6.38032913901	9.06569252875					
ENSG00000133703	268392.737116	12.9431463058	12.1158128056	16.1277280849	346811.057823	90950.7172562	164489.824014	176962.203393	7.05401220751	7.60650182851					
ENSG00000103126	247472.260459	11.6919161923	11.1714184762	6.08024441622	130749.724121	267129.691465	204555.751687	272129.887748	8.77220690192	11.6971672428					
ENSG00000157764	92390.4144488	3.61360485914	4.17069768176	3.70182570897	79604.1502709	30887.3494242	38022.868869	51113.3515001	1.63057978068	2.19704430773					
ENSG00000273993															
ENSG00000151532	42838.1985431	3.01183188614	1.93380640644	2.62755574107	56503.0226979	40401.36257	61154.2920399	66580.8251705	2.6225520343	2.86189456672					

At the same of constructing gene expression level matrix, another 49*2 matrix of truemodule colors were made as well. Genes that only appeared in colon cancer were marked as color “turquoise”; Genes that only appeared in lung cancer were marked as color “blue”; Gene that appeared in both cancers were marked as color “brown”. (See the first several rows in the figure below)

ENSG00000273686	turquoise
ENSG00000146374	turquoise
ENSG00000145675	turquoise
ENSG00000196090	brown
ENSG00000166710	turquoise
ENSG00000135679	turquoise
ENSG00000100644	brown
ENSG00000141646	turquoise
ENSG00000164754	brown
ENSG00000100393	turquoise
ENSG00000133703	brown
ENSG00000103126	turquoise

To make the R code from the provided tutorials work, I manually deleted three rows in both matrices where no information was found in patients’ gene expression level files or the expression levels were all 0 in all the patients. I also deleted the title row in expression level matrix. So the resulted two matrices both contained 46 rows. (See the first several rows in the figures below. Left is gene expression level matrix; Right is truemodule matrix)

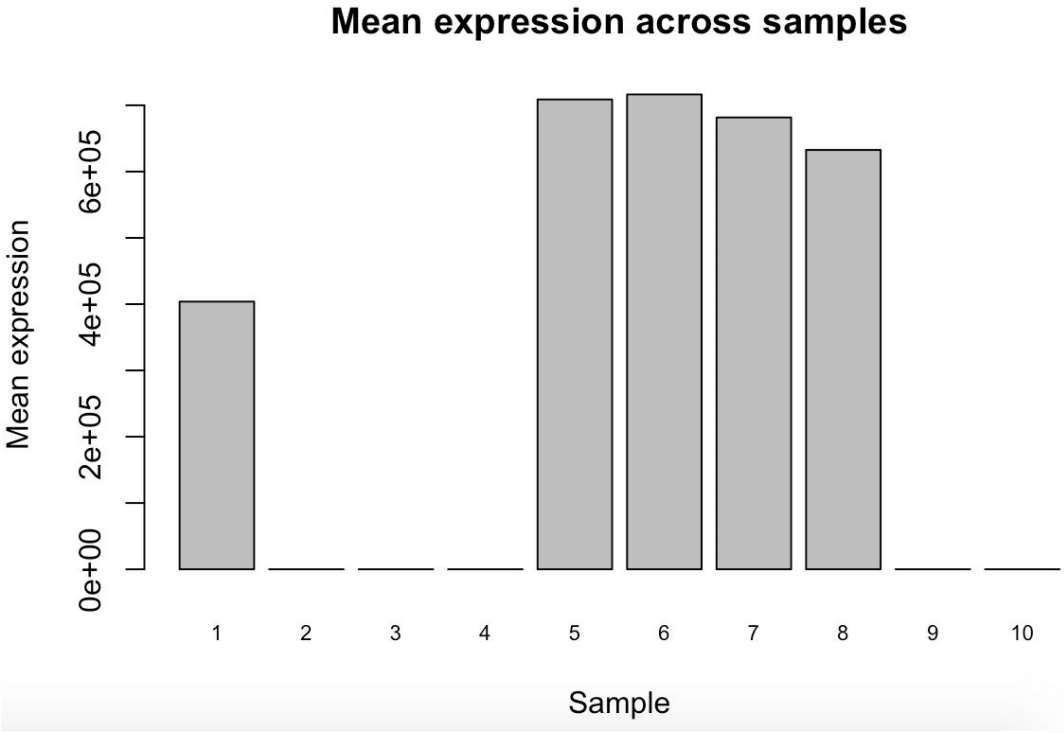
ENSG00000142208	437244.1381	15.6401135539	19.7381202803	15.1532333414	325855.499105	690387.642916
ENSG00000104517	234821.404329	14.1200963794	10.6003322153	10.3148969814	221811.796096	105592.616684
ENSG00000116062	264249.068884	10.3618730749	11.928758904	10.1092831489	217390.271229	78120.4972213
ENSG00000177565	848626.640413	29.4431882906	38.3087919128	71.1199516554		
ENSG00000065559	225996.42405	6.90599233625	10.2019540392	10.893497983		
ENSG00000169032	534498.182365	23.20510058	24.1283724442	27.2556925966		
ENSG00000121879	195802.425801	6.78273425648	8.8389334353	14.3161064185		
ENSG00000109670	41843.866596	3.42416573824	1.88892017045	1.73579100773		
ENSG00000148737	80168.6363299	5.30020097089	3.61898090496	5.07910953703		
ENSG00000147655	0.0	0.0520540436346	0.0	0.0135460976351	291.295613707	
ENSG00000104408	960006.029685	27.5615397719	43.3366918676	48.9565894067		
ENSG00000168646	15746.5560314	3.03309927806	0.710832667302	1.75283202844		
ENSG00000152894	289637.952559	12.9737891748	13.074866527	10.3329507334		
ENSG00000112531	153223.201667	5.76917997359	6.9168176785	8.21519059171		
ENSG00000183454	39.6003473544	0.0223336867434	0.0017876429919	0.017435780972		
ENSG00000141510	61918.0507361	15.9091713788	2.79511107516	55.6855431996		
ENSG00000175387	46023.2784663	2.78671095324	2.07758761503	2.39425022115		
ENSG00000095002	360310.39816	7.65456693786	16.2651694041	10.9884285472		
ENSG00000134982	85741.5165512	2.52734330657	3.87055244253	2.76470517051		
ENSG00000177084	107591.76426	4.03955376375	4.85691859328	4.380886326		

ENSG00000146374	turquoise
ENSG00000145675	turquoise
ENSG00000196090	brown
ENSG00000166710	turquoise
ENSG00000135679	turquoise
ENSG00000100644	brown
ENSG00000141646	turquoise
ENSG00000164754	brown
ENSG00000100393	turquoise
ENSG00000133703	brown
ENSG00000103126	turquoise
ENSG00000157764	turquoise
ENSG00000151532	turquoise
ENSG00000163513	turquoise
ENSG00000257923	turquoise

[Data cleaning & Pre-Processing]

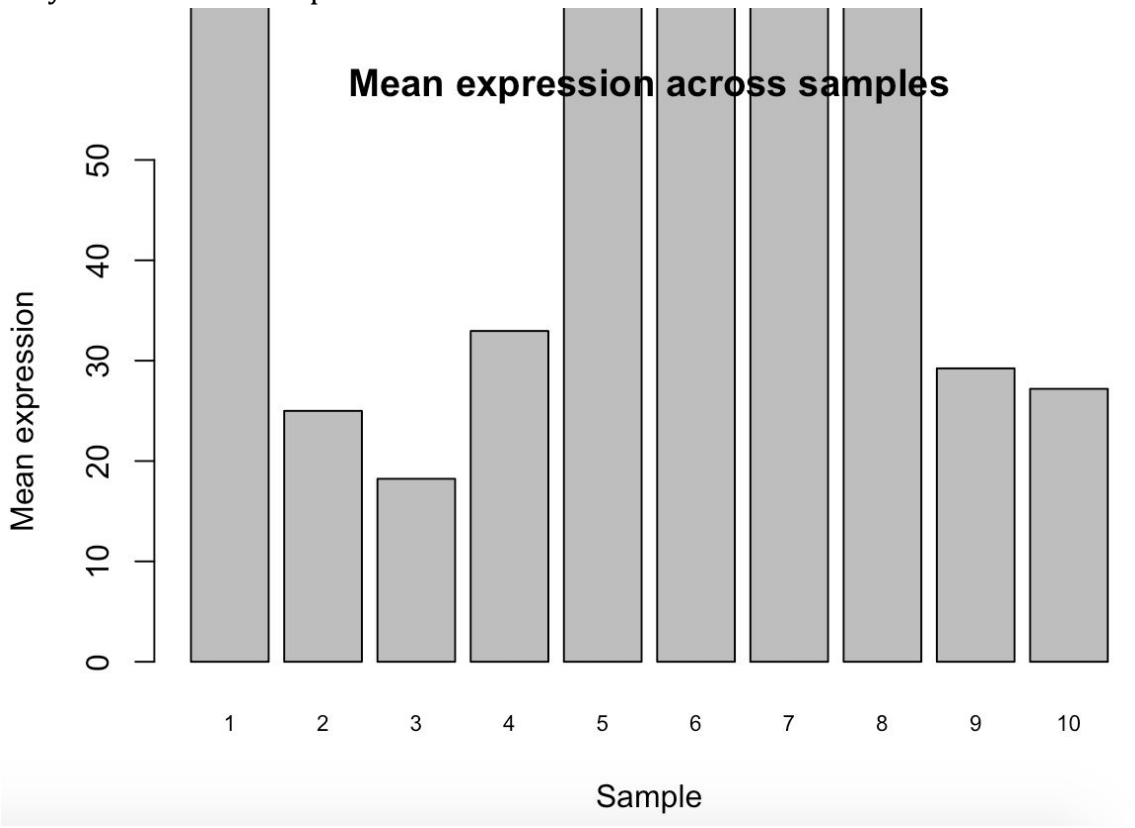
Using R and following “Simulated Tutorial 02/03”, the constructed gene expression level matrix were read as tables and restructured in the way we desire for making clusters. Then a barplot of mean expression (y-axis) of all probes in each sample/patient (x-axis). (See the figures below for the R codes(upper) and mean expression plot(lower))

```
library(WGCNA)
datMicroarrays=read.table("cancer_genes_com.txt")
ArrayName= names(data.frame(datMicroarrays[,-1]))
GeneName= datMicroarrays$GeneName
datExpr=data.frame(t(datMicroarrays[,-1]))
names(datExpr)=datMicroarrays[,1]
dimnames(datExpr)[[1]]=names(data.frame(datMicroarrays[,-1]))
meanExpressionByArray=apply( datExpr,1,mean, na.rm=T)
NumberMissingByArray=apply( is.na(data.frame(datExpr)),1, sum)
sizeGrWindow(9, 5)
barplot(meanExpressionByArray,
        xlab = "Sample", ylab = "Mean expression",
        main ="Mean expression across samples",
        names.arg = c(1:10), cex.names = 0.7)
KeepArray= NumberMissingByArray<500
table(KeepArray)
datExpr=datExpr[KeepArray,]
ArrayName[KeepArray]
NumberMissingByGene =apply( is.na(data.frame(datExpr)),2, sum)
# One could do a barplot(NumberMissingByGene), but the barplot is empty in this case.
# It may be better to look at the numbers of missing samples using the summary method:
summary(NumberMissingByGene)
# Calculate the variances of the probes and the number of present entries
variancedatExpr=as.vector(apply(as.matrix(datExpr),2,var, na.rm=T))
no.presentdatExpr=as.vector(apply(!is.na(as.matrix(datExpr)),2, sum) )
# Another way of summarizing the number of pressent entries
table(no.presentdatExpr)
# Keep only genes whose variance is non-zero and have at least 4 present entries
KeepGenes= variancedatExpr>0 & no.presentdatExpr>=4
table(KeepGenes)
datExpr=datExpr[, KeepGenes]
GeneName=GeneName[KeepGenes]
```



Note that there are some “0”s for some samples. Actually they are not really no expression but just relatively smaller so that it’s hard to observe. To illustrate this statement, I changed

the y limit value and replotted it:



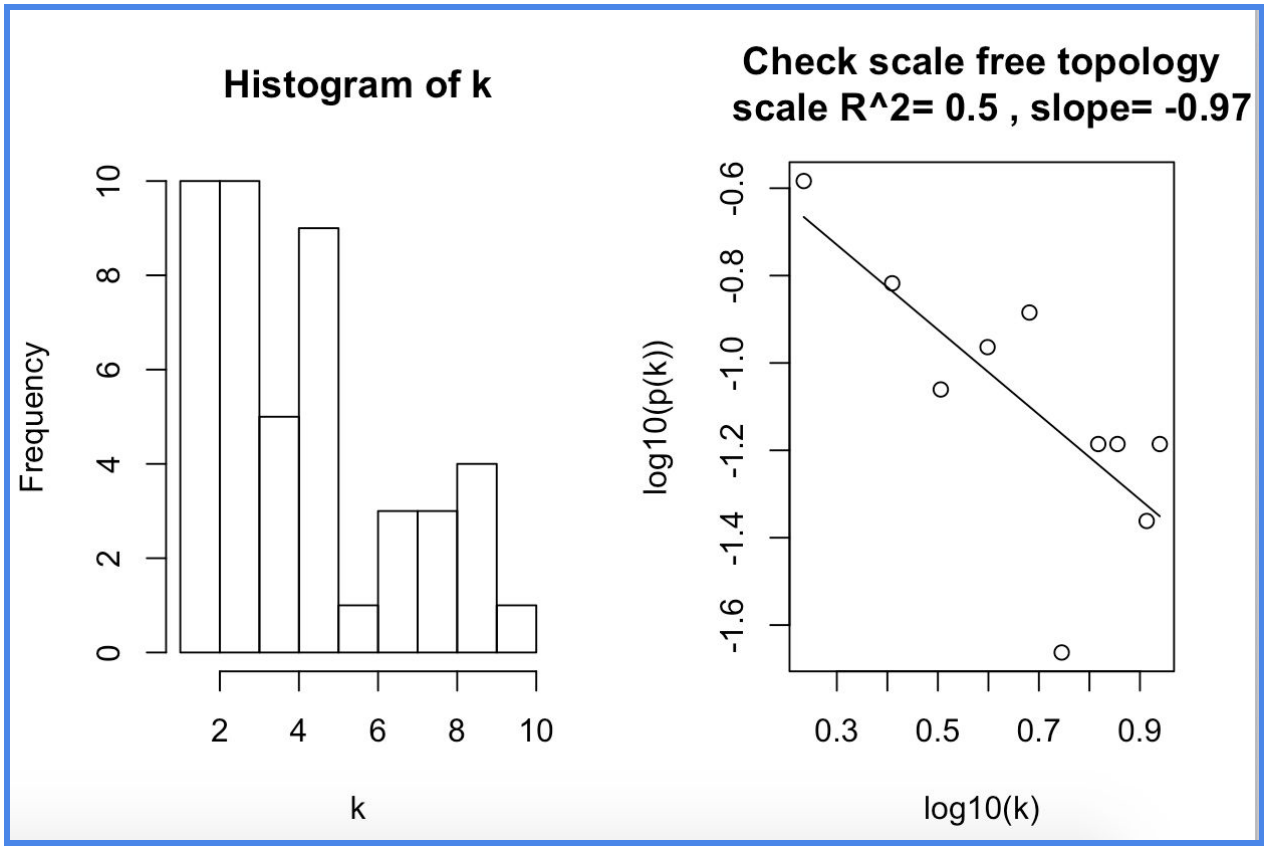
In this figure above, it's clear that all samples have some number of expression. Although the scale of data across samples differs a lot (e+01 VS. e+05), since there are almost half of the samples in both scales, I chose not to dismiss any samples. So no arrays in the plot were treated as an outlying mean expression value.

[Construction of WGCNA and network modules]

(the code in this section is following "simulated tutorial 05")

Firstly, I defined a weighted gene co-expression network(WGCN) and plotted the evaluation of scale free topology..(R code and plot is shown below).

```
library(cluster)
truemodule = read.table('color.txt')
truemodule=truemodule$V2
# here we define the adjacency matrix using soft thresholding with beta=6
ADJ1=abs(cor(datExpr,use="p"))^20
# When you have relatively few genes (<5000) use the following code
k=as.vector(apply(ADJ1,2,sum, na.rm=T))
# When you have a lot of genes use the following code
# k=softConnectivity(datE=datExpr,power=50)
# Plot a histogram of k and a scale free topology plot
sizeGrWindow(10,5)
par(mfrow=c(1,2))
hist(k)
scaleFreePlot(k, main="Check scale free topology\n")
```



I used a high power of 20 because in most applications it's found that scale free topology is at least approximately satisfied when a high power is chosen. It turned out that the network satisfied scale free topology(R^2 value in the right panel is relatively large). However, I think this power of 20 is a little bit too big. I propose that it's probably because the data are comprised of globally distinct groups of samples(different organs where cancer was detected) or because the expression scales differ too much as shown in the figure before.

[Comparing various module detection methods]

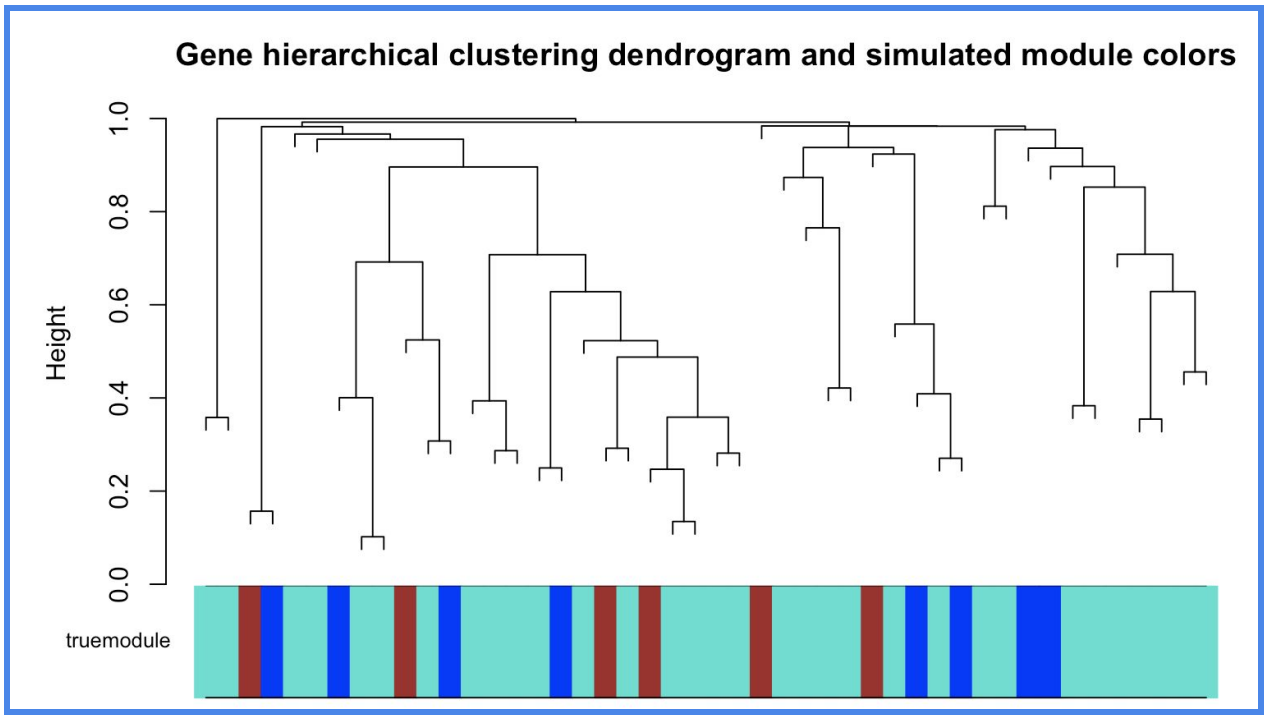
- Before detecting modules, I first defined the clustering dissimilarity from adjacency because many clustering procedures required a dissimilarity matrix as input. Then I used topological overlap to define dissimilarity. (shown in the code below)

```
dissADJ=1-ADJ1
dissTOM=TOMdist(ADJ1)
collectGarbage()
```

- Then I started with "Average linkage hierarchical clustering" by running the R codes below:

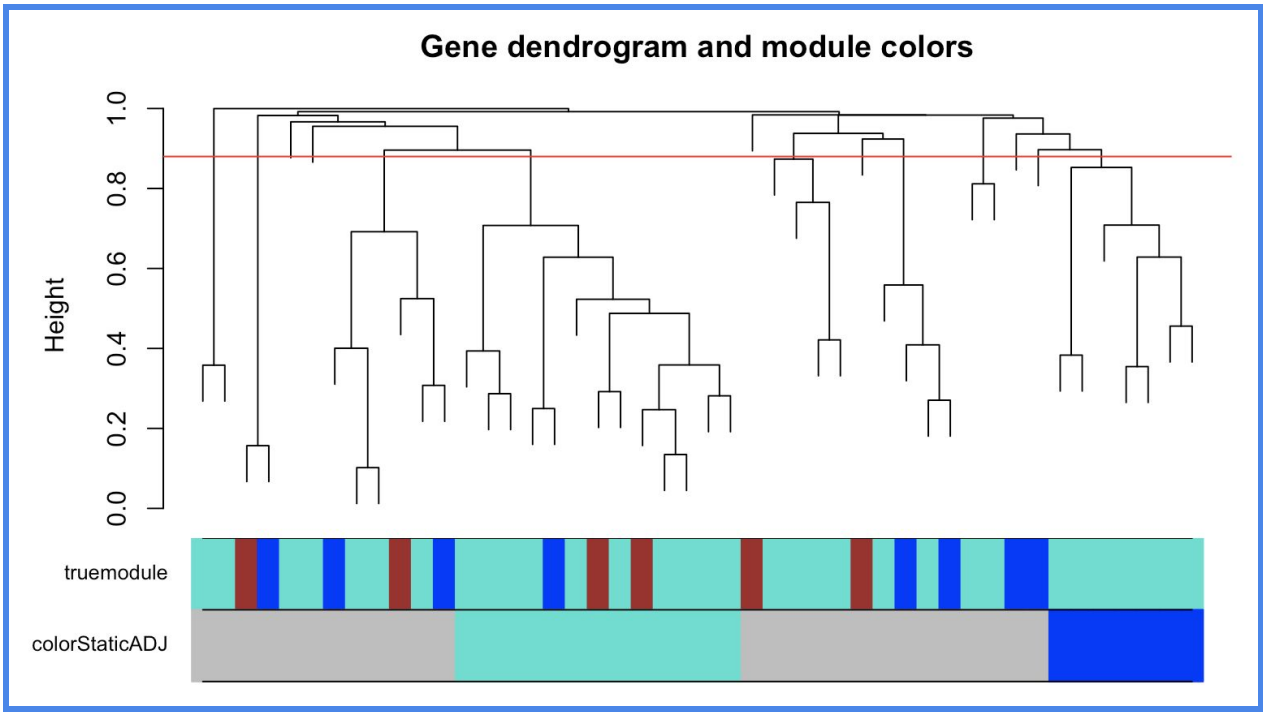
```
hierADJ=hclust(as.dist(dissADJ), method="average" )
# Plot the resulting clustering tree together with the true color assignment
sizeGrWindow(10,5);
plotDendroAndColors(hierADJ, colors = data.frame(truemodule), dendroLabels = FALSE, hang = 0.03,
                    main = "Gene hierarchical clustering dendrogram and simulated module colors" )
```

The result is shown in the figure below. Different colors in the band were defined in the truemodule matrix mentioned before.



- After that, I first used "static height cut-off" method to cut the tree. The function cutreeStaticColor colors each gene by the branches that result from choosing 0.88 as height cut-off. The label "grey" is reserved to color genes that are not part of any module. Here we only consider modules that contain at least minSize(in my case, 7)genes. The R code and result is shown below:

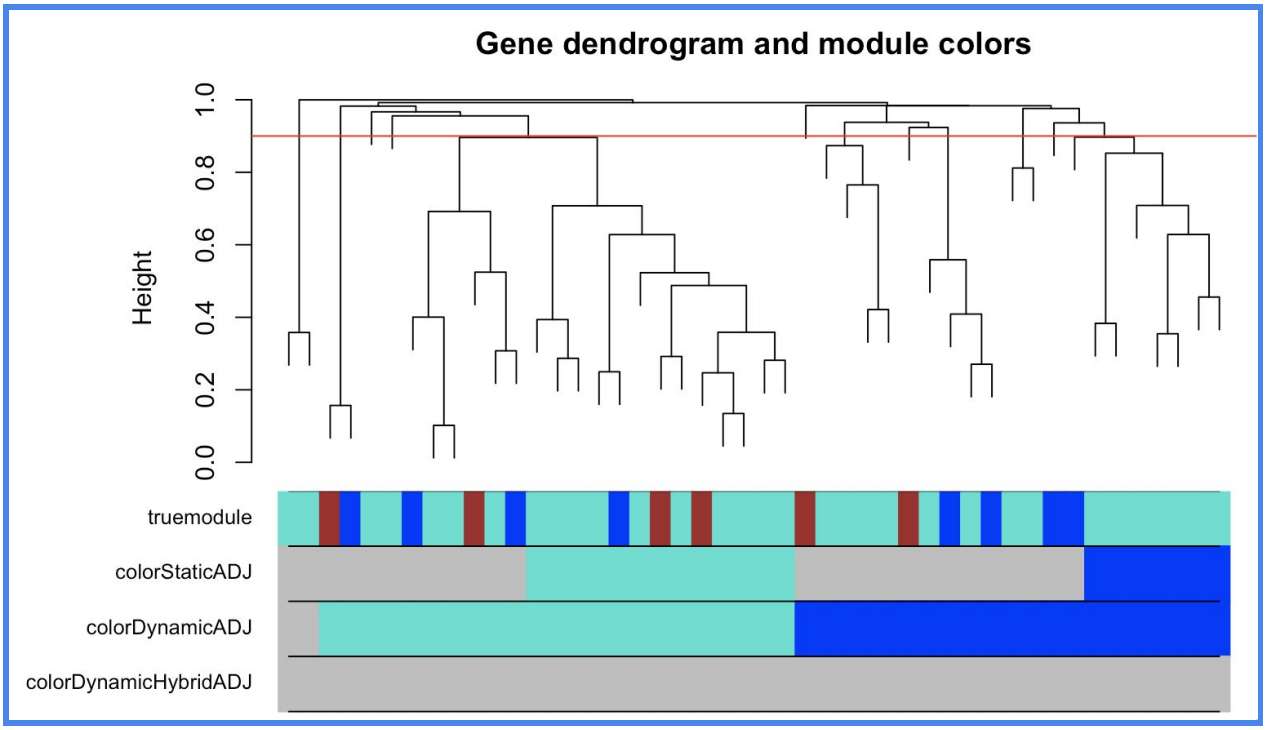
```
colorStaticADJ=as.character(cutreeStaticColor(hierADJ, cutHeight=.88, minSize=7))
# Plot the dendrogram with module colors
sizeGrWindow(10,5);
plotDendroAndColors(hierADJ, colors = data.frame(truemodule, colorStaticADJ),
                    dendroLabels = FALSE, abHeight = 0.88,
                    main = "Gene dendrogram and module colors")
```



The static height cut-off method works not bad at retrieving the true modules. At lease we can see the two modules here. However, it misses a lot of genes at the fringes of the modules.

- Then I used "dynamic branch cutting" methods to cut the tree. I used two Dynamic Tree Cut methods in which the height cut-off played a minor role. The first method is called the "tree" method and only uses the dendrogram as input. The second method is called "hybrid" and is a hybrid between hclust and pam. As input it requires both a dendrogram and the dissimilarity that was used to create the dendrogram(cited from tutorial). Here I used cutHeight of 0.9. The R code and result of all methods together is shown below:

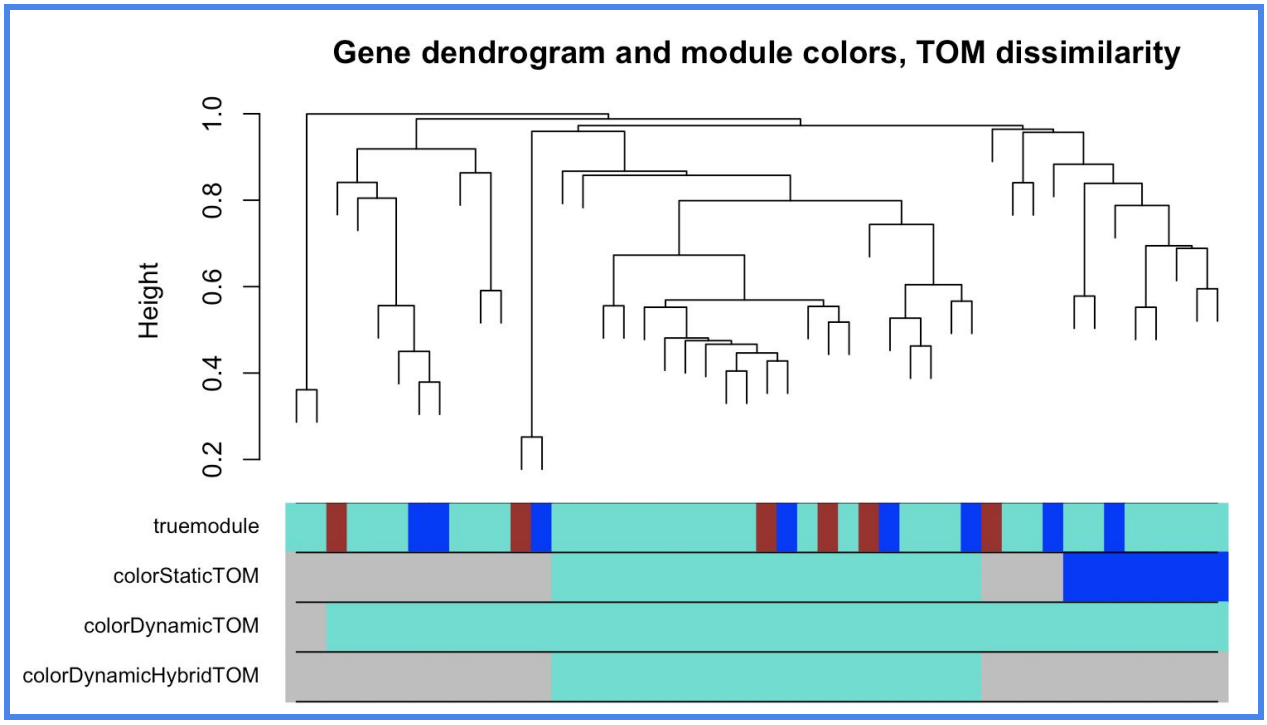
```
branch.number=cutreeDynamic(hierADJ,method="tree")
# This function transforms the branch numbers into colors
colorDynamicADJ=labels2colors(branch.number )
colorDynamicHybridADJ=labels2colors(cutreeDynamic(hierADJ,distM= dissADJ,
                                                  cutHeight = 0.9, deepSplit=2, pamRespectsDendro = FALSE))
# Plot results of all module detection methods together:
sizeGrWindow(10,5)
plotDendroAndColors(dendro = hierADJ,
                    colors=data.frame(truemodule, colorStaticADJ,
                                      colorDynamicADJ, colorDynamicHybridADJ),
                    dendroLabels = FALSE, marAll = c(0.2, 8, 2.7, 0.2),abHeight = 0.9,
                    main = "Gene dendrogram and module colors")
```



The static method (2nd band) has high specificity but low sensitivity, i.e. its module membership assignment was relatively accurate but it missed a lot of genes (too many grey genes). In contrast, the dynamic method (first band) had high sensitivity but low specificity. And the dynamic hybrid method didn't work in this task.

- Then I used "topological overlap" as input to the clustering methods used above. The cutHeight and minSize were set the same as the corresponding clustering methods shown above. The code and result is shown below:

```
# Calculate the dendrogram
hierTOM = hclust(as.dist(dissTOM),method="average");
# The reader should vary the height cut-off parameter h1
# (related to the y-axis of dendrogram) in the following
colorStaticTOM = as.character(cutreeStaticColor(hierTOM, cutHeight=.88, minSize=7))
colorDynamicTOM = labels2colors (cutreeDynamic(hierTOM,method="tree"))
colorDynamicHybridTOM = labels2colors(cutreeDynamic(hierTOM, distM= dissTOM , cutHeight = 0.9,
                                                    deepSplit=2, pamRespectsDendro = FALSE))
# Now we plot the results
sizeGrWindow(10,5)
plotDendroAndColors(hierTOM,
                    colors=data.frame(truemodule, colorStaticTOM,
                                      colorDynamicTOM, colorDynamicHybridTOM),
                    dendroLabels = FALSE, marAll = c(1, 8, 3, 1),
                    main = "Gene dendrogram and module colors, TOM dissimilarity")
```



We can see from the figure above aht only colorStatic method gives a result of distint two modules.

Discussion

- Which dissimilarity measure and which branch cutting method performs best in this data set?

To answer this question, I created tables for relating the different module assignments to the true module colors. Next, I used the (unadjusted) Rand index to measure agreement which computes the Rand index for each table.(cited from Tutorial 05) The code and returned Rand indices are as follows:

```
tabStaticADJ=table(colorStaticADJ,truemodule)
tabStaticTOM=table(colorStaticTOM,truemodule)
tabDynamicADJ=table(colorDynamicADJ, truemodule)
tabDynamicTOM=table(colorDynamicTOM,truemodule)
tabDynamicHybridADJ =table(colorDynamicHybridADJ,truemodule)
tabDynamicHybridTOM =table(colorDynamicHybridTOM,truemodule)
randIndex(tabStaticADJ,adjust=F)
randIndex(tabStaticTOM,adjust=F)
randIndex(tabDynamicADJ,adjust=F)
randIndex(tabDynamicTOM,adjust=F)
randIndex(tabDynamicHybridADJ ,adjust=F)
randIndex(tabDynamicHybridTOM ,adjust=F)
```

```
> randIndex(tabStaticADJ,adjust=F)
[1] 0.4541063
> randIndex(tabStaticTOM,adjust=F)
[1] 0.4772947
> randIndex(tabDynamicADJ,adjust=F)
[1] 0.4753623
> randIndex(tabDynamicTOM,adjust=F)
[1] 0.4898551
> randIndex(tabDynamicHybridADJ ,adjust=F)
[1] 0.5207729
> randIndex(tabDynamicHybridTOM ,adjust=F)
[1] 0.4888889
```

In my particular data set, dissTOM performs better than dissADJ for the first two branch cutting method. And dissTOM performs worse than dissADJ in the DynamicHybrid method. It's hard to decide which to choose to do subsequent analysis because although the results for Dynamic methods are relatively similar, Dynamic method didn't even give two distinct modules. On the other hand, although Static method gave two relatively beautiful

modules, since my minSize was set as 7, which was really small so that in most cases wouldn't be meaningful, I can't claim that Static is better method as well.

In conclusion, the results of this project are not salient enough to make and reasonable choice among the mentioned clustering method. However, it did give me a taste of cancer related gene study and the using of strength of R packages "WGCNA" and "cluster". To make the result more salient, I think I should collect data from more patients and select more genes besides those in the cancer gene consensus of COSMIC. Future work would include studying one type of cancer and detecting if any mutation hotspots or studying one type of cancer and detecting if any hyper- or hypo-methylation genes can be found.

References

Simulated-02-dataLoading.pdf (uploaded by Prof. Fang)
Simulated-03-Preprocessing.pdf (uploaded by Prof. Fang)
Simulated-05-NetworkConstruction.pdf (uploaded by Prof. Fang)