

WGCNA Analysis

Asad

2023-05-12

Call libraries

```
library(WGCNA)
```

```
## Loading required package: dynamicTreeCut
```

```
## Loading required package: fastcluster
```

```
##  
## Attaching package: 'fastcluster'
```

```
## The following object is masked from 'package:stats':  
##  
##      hclust
```

```
## Warning: replacing previous import 'utils::findMatches' by  
## 'S4Vectors::findMatches' when loading 'AnnotationDbi'
```

```
##
```

```
##  
## Attaching package: 'WGCNA'
```

```
## The following object is masked from 'package:stats':  
##  
##      cor
```

```
library(DESeq2)
```

```
## Loading required package: S4Vectors
```

```
## Loading required package: stats4
```

```
## Loading required package: BiocGenerics
```

```
##  
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:stats':  
##  
##    IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':  
##  
##    anyDuplicated, append, as.data.frame, basename, cbind, colnames,  
##    dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,  
##    grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,  
##    order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,  
##    rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,  
##    union, unique, unsplit, which.max, which.min
```

```
##  
## Attaching package: 'S4Vectors'
```

```
## The following object is masked from 'package:utils':  
##  
##    findMatches
```

```
## The following objects are masked from 'package:base':  
##  
##    expand.grid, I, unname
```

```
## Loading required package: IRanges
```

```
##  
## Attaching package: 'IRanges'
```

```
## The following object is masked from 'package:grDevices':  
##  
##    windows
```

```
## Loading required package: GenomicRanges
```

```
## Loading required package: GenomeInfoDb
```

```
## Loading required package: SummarizedExperiment
```

```
## Loading required package: MatrixGenerics
```

```
## Loading required package: matrixStats
```

```
##  
## Attaching package: 'MatrixGenerics'
```

```
## The following objects are masked from 'package:matrixStats':  
##  
##   colAlls, colAnyNAs, colAnys, colAvgPerRowSet, colCollapse,  
##   colCounts, colCummaxs, colCummins, colCumprods, colCumsums,  
##   colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,  
##   colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,  
##   colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,  
##   colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,  
##   colWeightedMeans, colWeightedMedians, colWeightedSds,  
##   colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgPerColSet,  
##   rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,  
##   rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,  
##   rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,  
##   rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,  
##   rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,  
##   rowWeightedMads, rowWeightedMeans, rowWeightedMedians,  
##   rowWeightedSds, rowWeightedVars
```

```
## Loading required package: Biobase
```

```
## Welcome to Bioconductor  
##  
##   Vignettes contain introductory material; view with  
##   'browseVignettes()'. To cite Bioconductor, see  
##   'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
##  
## Attaching package: 'Biobase'
```

```
## The following object is masked from 'package:MatrixGenerics':  
##  
##   rowMedians
```

```
## The following objects are masked from 'package:matrixStats':
##
##   anyMissing, rowMedians
```

```
library(tidyverse)
```

```
## — Attaching packages ————— tidyverse 1.3.1 —
```

```
## ✓ ggplot2 3.4.2    ✓ purrr  1.0.1
## ✓ tibble  3.2.1    ✓ dplyr  1.1.1
## ✓ tidyr   1.3.0    ✓ stringr 1.5.0
## ✓ readr   2.1.2    ✓ forcats 1.0.0
```

```
## — Conflicts ————— tidyverse_conflicts() —
## ✗ dplyr::collapse() masks IRanges::collapse()
## ✗ dplyr::combine()  masks Biobase::combine(), BiocGenerics::combine()
## ✗ dplyr::count()    masks matrixStats::count()
## ✗ dplyr::desc()     masks IRanges::desc()
## ✗ tidyr::expand()   masks S4Vectors::expand()
## ✗ dplyr::filter()   masks stats::filter()
## ✗ dplyr::first()    masks S4Vectors::first()
## ✗ dplyr::lag()      masks stats::lag()
## ✗ ggplot2::Position() masks BiocGenerics::Position(), base::Position()
## ✗ purrr::reduce()   masks GenomicRanges::reduce(), IRanges::reduce()
## ✗ dplyr::rename()   masks S4Vectors::rename()
## ✗ dplyr::slice()    masks IRanges::slice()
```

```
library(CorLevelPlot)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##   combine
```

```
## The following object is masked from 'package:Biobase':
##
##   combine
```

```
## The following object is masked from 'package:BiocGenerics':  
##  
##      combine
```

```
library(magrittr)
```

```
##  
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:purrr':  
##  
##      set_names
```

```
## The following object is masked from 'package:tidyr':  
##  
##      extract
```

```
library(ggplot2)  
  
allowWGCNAThreads()
```

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

```
setwd('E:/WGCNA')
```

1. Read data

```
data<- read.csv('Counts.csv', row.names = 1)  
Sample_info<- read.csv('Sample_info.csv', row.names = 1)
```

2. Quality control

Step1: Detect outlier genes with WGCNA package

```
gg<- goodSamplesGenes(t(data))
```

```
## Flagging genes and samples with too many missing values...  
## ..step 1  
## ..step 2
```

```
summary(gg)
```

```
##           Length Class  Mode
## goodGenes  28089  -none- logical
## goodSamples    52  -none- logical
## allOK         1  -none- logical
```

```
gg$allOK
```

```
## [1] FALSE
```

#Since it indicates tat all genes are not good genes let's quantify the outlier genes #We'll do it for samples as well

```
table(gg$goodGenes)
```

```
##
## FALSE  TRUE
##  3576 24513
```

```
table(gg$goodSamples)
```

```
##
## TRUE
##   52
```

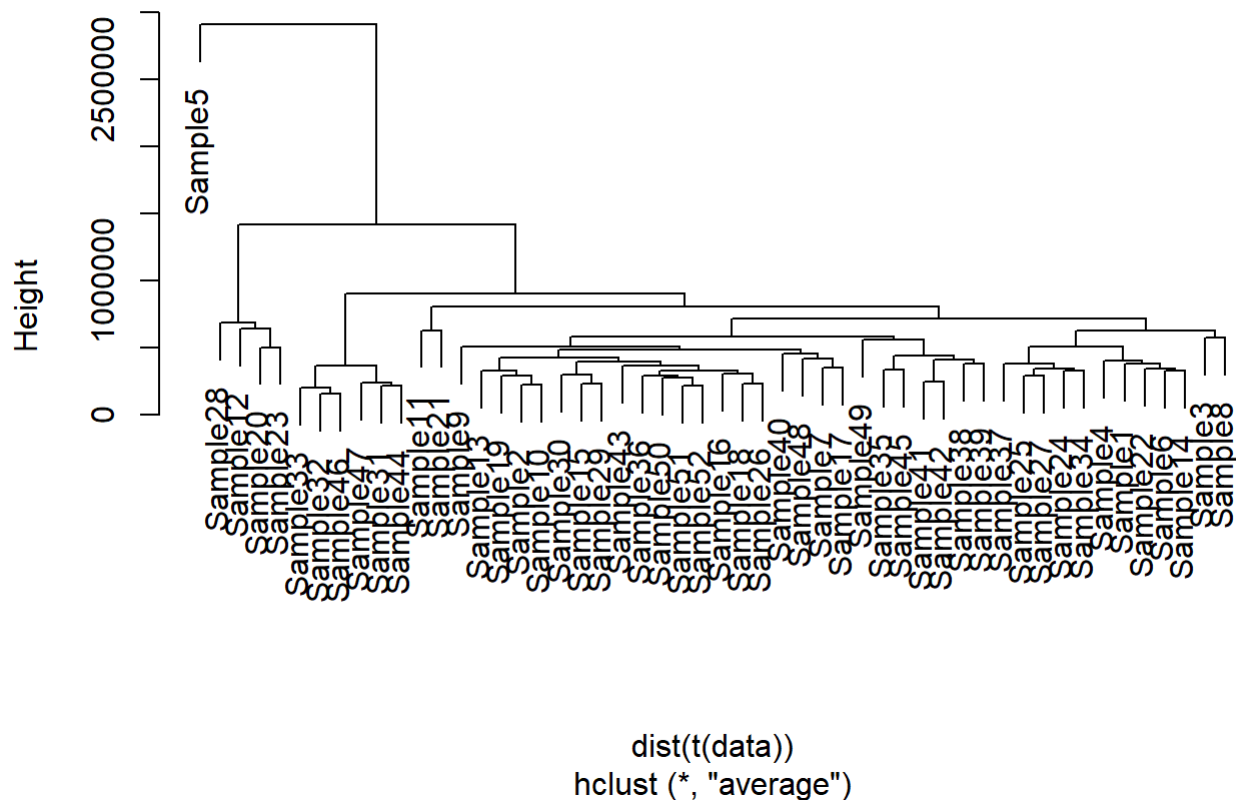
#Around 3500 outlier genes but no outlier samples. Let's remove those genes

```
data<- data[gg$goodGenes=='TRUE',]
```

Step2: Identify outlier samples with hierachical clustering

```
htree <- hclust(dist(t(data)), method = "average")
plot(htree)
```

Cluster Dendrogram



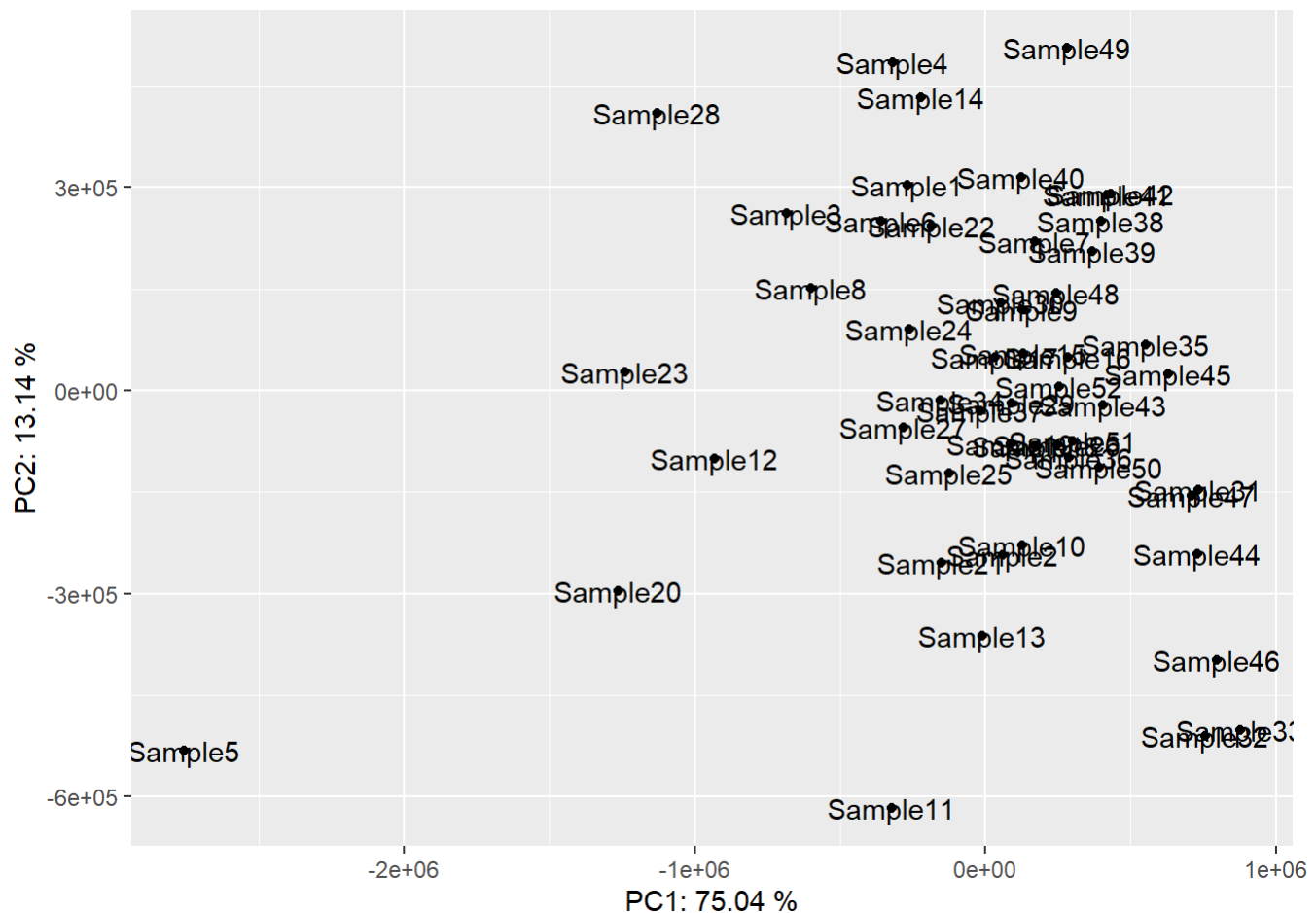
Step3: Identify outlier samples with PCA

```
pca <- prcomp(t(data))
pca.dat <- pca$x

pca.var <- pca$sdev^2
pca.var.percent <- round(pca.var/sum(pca.var)*100, digits = 2)

pca.dat <- as.data.frame(pca.dat)

ggplot(pca.dat, aes(PC1, PC2)) +
  geom_point() +
  geom_text(label = rownames(pca.dat)) +
  labs(x = paste0('PC1: ', pca.var.percent[1], ' %'),
       y = paste0('PC2: ', pca.var.percent[2], ' %'))
```



#N.B. We need to remove batch effect if there's any at this stage before we proceed further

#In both these methods we found Sample5 is distantly related

#So we'll exclude them

```
discard_samples<- c('Sample5')
```

```
data.final<- data[,!colnames(data) %in% discard_samples]
```

```
Sample_info_final<- Sample_info[!row.names(Sample_info) %in% discard_samples,]
```

4. Perform VST normalization with DESeq2 since it's a count matrix

#N.B. FPKM or RPKM matrices need to be log transformed #Before performing normalization make sure if the rownames in matrix and colnames in #Sample info are identical

```
all(rownames(Sample_info_final)%in%colnames(data.final)) #If present
```

```
## [1] TRUE
```

```
all(rownames(Sample_info_final)==colnames(data.final)) #If present in same order
```



```
## [1] TRUE
```

Create dseq2 object

```
dds <- DESeqDataSetFromMatrix(countData = data.final,  
                              colData = Sample_info_final,  
                              design = ~ 1) #Specifying no design  
  
nrow(dds)
```

```
## [1] 24513
```

Remove samples that have <15 counts in more than 75% of the samples

#Recommended by WGCNA #However, the downstream analysis with higher # of genes will depend largely on #the available ram of PC. So, we'll consider gene numbers based on our PC capacity.

```
keep <- dds[rowSums(counts(dds) >= 15)>=39,] #75% of 52 samples is 52*0.75=39  
nrow(keep)
```

```
## [1] 12226
```

Perform VST transformation

```
dds.vst<- vst(keep)  
dds.vst<- dds.vst[1:5000,] #We reduced the # of genes to 8000 because of low RAM  
nrow(dds.vst)
```

```
## [1] 5000
```

Get normalized counts

```
norm.counts <- assay(dds.vst) %>%  
  t()
```

5. Constructing network

Set softthreshold power

```
power <- c(c(1:10), seq(from = 12, to = 50, by = 2))
```

Set the network topology analysis function

```
sft <- pickSoftThreshold(norm.counts,
                          powerVector = power,
                          networkType = "signed",
                          verbose = 5)
```

```
## pickSoftThreshold: will use block size 5000.
## pickSoftThreshold: calculating connectivity for given powers...
## ..working on genes 1 through 5000 of 5000
##   Power SFT.R.sq slope truncated.R.sq mean.k. median.k. max.k.
## 1      1    0.1310 32.20          0.974 2.50e+03 2.50e+03 2580.00
## 2      2    0.0456 -5.56          0.951 1.34e+03 1.33e+03 1470.00
## 3      3    0.2070 -5.36          0.962 7.52e+02 7.47e+02 931.00
## 4      4    0.3250 -4.12          0.966 4.43e+02 4.36e+02 627.00
## 5      5    0.3900 -3.25          0.960 2.71e+02 2.64e+02 443.00
## 6      6    0.4500 -2.67          0.962 1.72e+02 1.64e+02 324.00
## 7      7    0.5190 -2.36          0.966 1.12e+02 1.05e+02 244.00
## 8      8    0.5990 -2.33          0.975 7.56e+01 6.88e+01 191.00
## 9      9    0.6550 -2.37          0.977 5.21e+01 4.59e+01 153.00
## 10     10   0.7020 -2.35          0.980 3.67e+01 3.12e+01 124.00
## 11     12   0.7560 -2.39          0.983 1.92e+01 1.51e+01 85.80
## 12     14   0.7910 -2.47          0.982 1.08e+01 7.73e+00 61.50
## 13     16   0.8010 -2.47          0.965 6.33e+00 4.13e+00 45.40
## 14     18   0.8090 -2.55          0.977 3.89e+00 2.27e+00 34.20
## 15     20   0.8280 -2.47          0.982 2.48e+00 1.28e+00 26.30
## 16     22   0.8380 -2.42          0.975 1.63e+00 7.48e-01 20.50
## 17     24   0.8280 -2.41          0.948 1.10e+00 4.44e-01 16.20
## 18     26   0.8410 -2.35          0.949 7.63e-01 2.70e-01 13.00
## 19     28   0.8500 -2.29          0.947 5.39e-01 1.66e-01 10.50
## 20     30   0.3950 -2.99          0.321 3.87e-01 1.04e-01 8.52
## 21     32   0.3950 -2.91          0.321 2.83e-01 6.61e-02 6.99
## 22     34   0.4190 -2.90          0.426 2.10e-01 4.29e-02 5.78
## 23     36   0.4150 -2.82          0.407 1.58e-01 2.79e-02 4.81
## 24     38   0.9330 -1.95          0.994 1.20e-01 1.83e-02 4.03
## 25     40   0.9280 -1.90          0.962 9.27e-02 1.23e-02 3.39
## 26     42   0.9340 -1.88          0.956 7.23e-02 8.30e-03 3.11
## 27     44   0.9460 -1.86          0.957 5.69e-02 5.56e-03 2.92
## 28     46   0.9470 -1.85          0.950 4.52e-02 3.73e-03 2.74
## 29     48   0.9400 -1.82          0.932 3.63e-02 2.54e-03 2.58
## 30     50   0.4160 -2.32          0.276 2.94e-02 1.75e-03 2.43
```

#N.B. unsigned -> nodes with positive & negative correlation are treated equally #N.B. signed -> nodes with negative correlation are considered *unconnected*, treated as zero

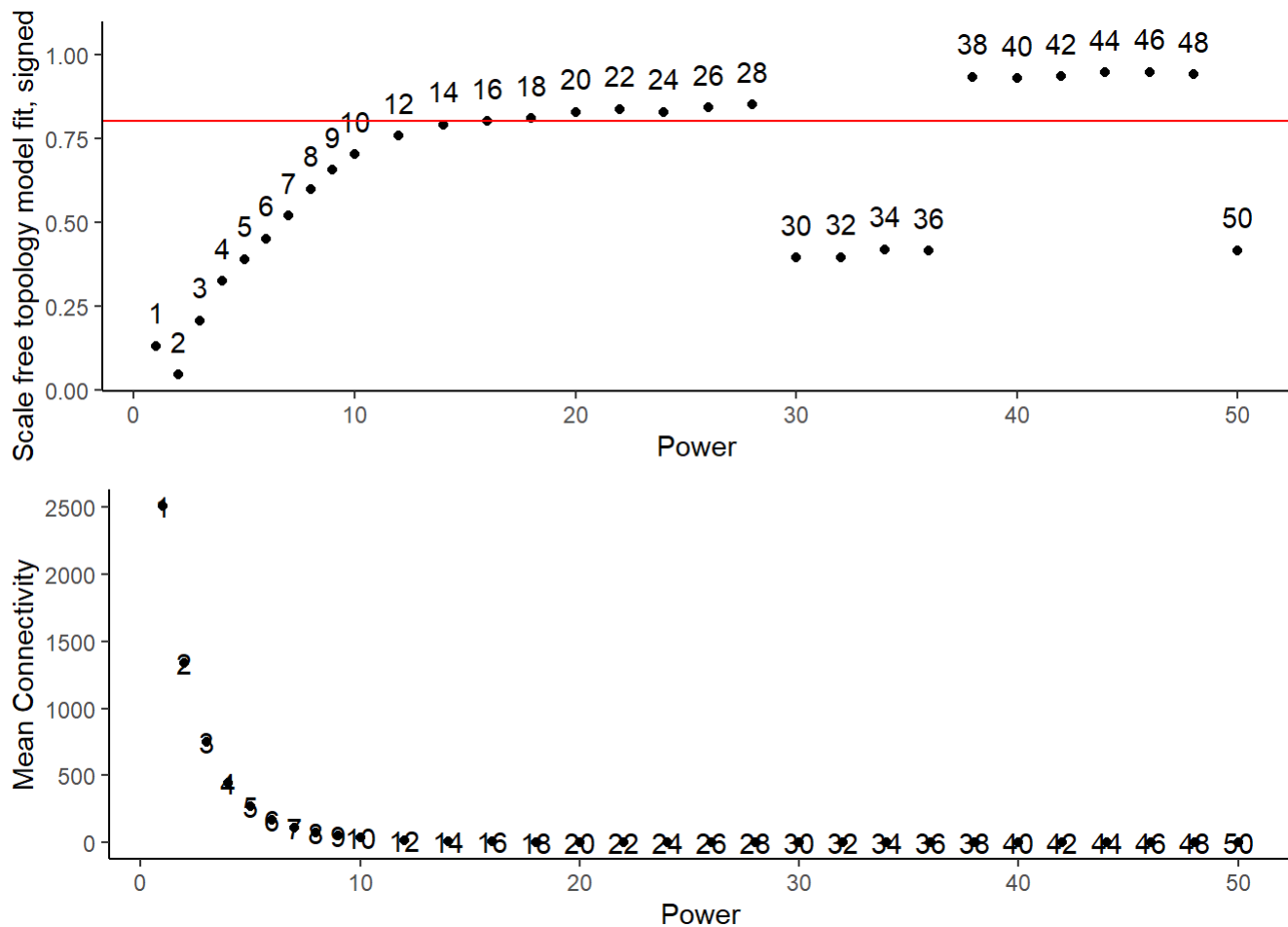
```
sft.data <- sft$fitIndices
```

visualization to pick power

```
Plot1 <- ggplot(sft.data, aes(Power, SFT.R.sq, label = Power)) +
  geom_point() +
  geom_text(nudge_y = 0.1) +
  geom_hline(yintercept = 0.8, color = 'red') +
  labs(x = 'Power', y = 'Scale free topology model fit, signed R^2') +
  theme_classic()
```

```
Plot2 <- ggplot(sft.data, aes(Power, mean.k., label = Power)) +
  geom_point() +
  geom_text(nudge_y = 0.1) +
  labs(x = 'Power', y = 'Mean Connectivity') +
  theme_classic()
```

```
grid.arrange(Plot1, Plot2, nrow = 2)
```



convert matrix to numeric and assign power

```
norm.counts[] <- sapply(norm.counts, as.numeric)
soft_power <- 14
temp_cor <- cor
cor <- WGCNA::cor
```

Estimate w.r.t blocksize memory depends on available ram of PC

```
bnwnet <- blockwiseModules(norm.counts,
                           maxBlockSize = 5000,
                           TOMType = "signed",
                           power = soft_power,
                           mergeCutHeight = 0.25,
                           numericLabels = FALSE,
                           randomSeed = 1234,
                           verbose = 3)
```

```
## Calculating module eigengenes block-wise from all genes
##   Flagging genes and samples with too many missing values...
##   ..step 1
##   ..Working on block 1 .
##   TOM calculation: adjacency..
##   ..will not use multithreading.
##   Fraction of slow calculations: 0.000000
##   ..connectivity..
##   ..matrix multiplication (system BLAS)..
##   ..normalization..
##   ..done.
##   ....clustering..
##   ....detecting modules..
##   ....calculating module eigengenes..
##   ....checking kME in modules..
##   ..reassigning 1 genes from module 3 to modules with higher KME.
##   ..merging modules that are too close..
##   mergeCloseModules: Merging modules whose distance is less than 0.25
##   Calculating new MEs...
```

```
cor <- temp_cor
```

6. Identify Module Eigengenes

```
module_eigengenes <- bwnet$MEs
head(module_eigengenes)
```

```
##           MEturquoise    MEbrown    MEgreen    MEblack    MEdred
## Sample1 -0.10375879    0.21685398    0.020542538 -0.09222291 -0.100192258
## Sample2 -0.01255866    0.12839304    0.032163029 -0.07031597 -0.221029610
## Sample3  0.04519388    0.03717920    0.066635498 -0.05908388  0.017316170
## Sample4  0.06835863   -0.08730094   -0.003943952 -0.06046904  0.074580135
## Sample6  0.15559583   -0.04275675    0.087524806 -0.07252025  0.041854527
## Sample7  0.05497992    0.13879174   -0.144092037 -0.04763668  -0.004967255
##           MEblue      MEyellow      MEgrey
## Sample1 -0.06997022   -0.167852659   -0.0823958709
## Sample2 -0.12854895   -0.149656308    0.0066448007
## Sample3 -0.02317673   -0.073541884   -0.0006775664
## Sample4 -0.06173939   -0.047079887    0.0677114636
## Sample6 -0.14128791   -0.004358101    0.1408660417
## Sample7 -0.13322219   -0.090824750    0.0951961513
```

Check the number of genes for each module

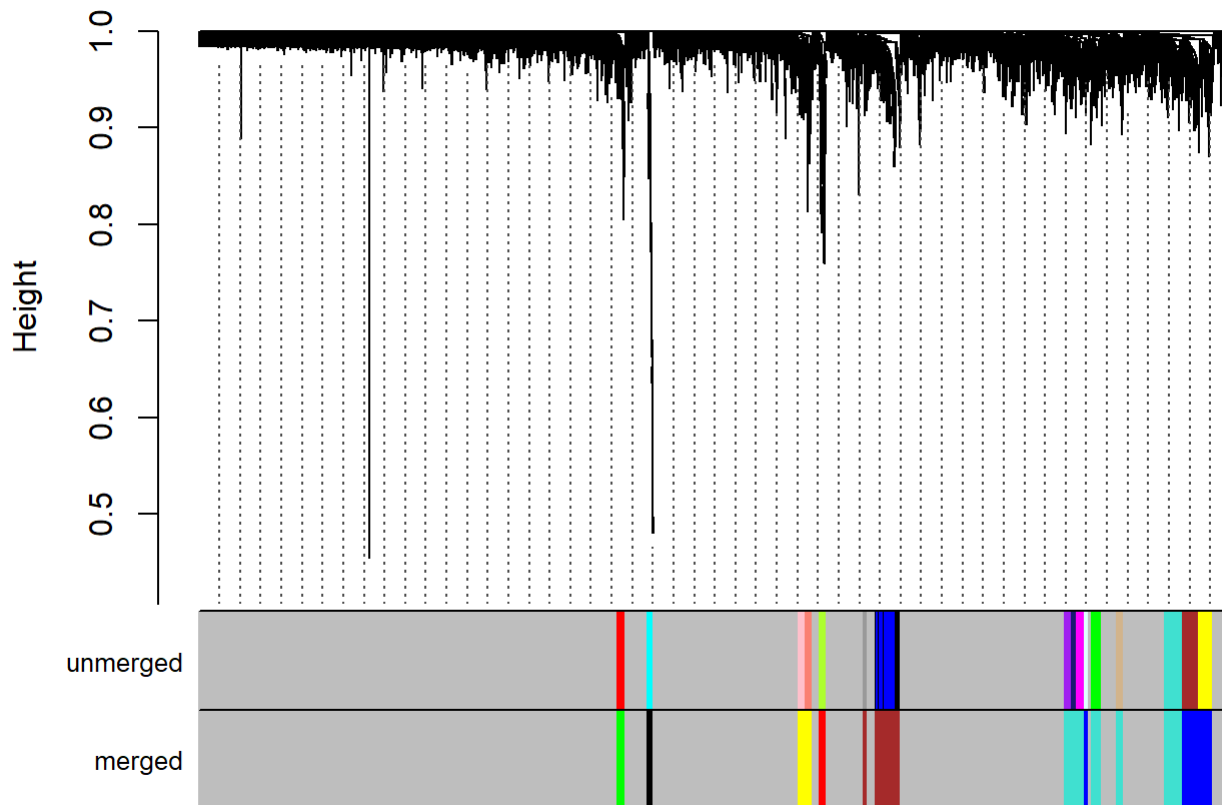
```
table(bwnet$colors)
```

```
##
##      black      blue      brown      green      grey      red turquoise      yellow
##         26        164        141         39       4262         32         267         69
```

Plot the dendrogram and the module colors before and after merging underneath

```
plotDendroAndColors(bwnet$dendrograms[[1]], cbind(bwnet$unmergedColors, bwnet$colors),
  c("unmerged", "merged"),
  dendroLabels = FALSE,
  addGuide = TRUE,
  hang= 0.03,
  guideHang = 0.05)
```

Cluster Dendrogram



grey module = all genes that doesn't fall into other modules were assigned to the grey module

7. Extract genes and their associated color modules and save them

```
mergedColors<- labels2colors(bwnet$colors)
module_df <- data.frame(
  gene_id <- names(bwnet$colors),
  colors <- labels2colors(bwnet$colors)
)

module_df[1:5,]
```

```
##  gene_id....names.bwnet.colors.  colors....labels2colors.bwnet.colors.
## 1          ENSG00000000419                green
## 2          ENSG00000000457                green
## 3          ENSG00000000460                green
## 4          ENSG00000000938                green
## 5          ENSG00000001036                green
```

```
write_delim(module_df,  
            file = "gene_modules.txt",  
            delim = "\t")
```

8. Merging and clustering modules from eigengenes

Calculate eigengenes

```
MEList <- moduleEigengenes(norm.counts, colors = mergedColors)  
MEs <- MEList$eigengenes
```

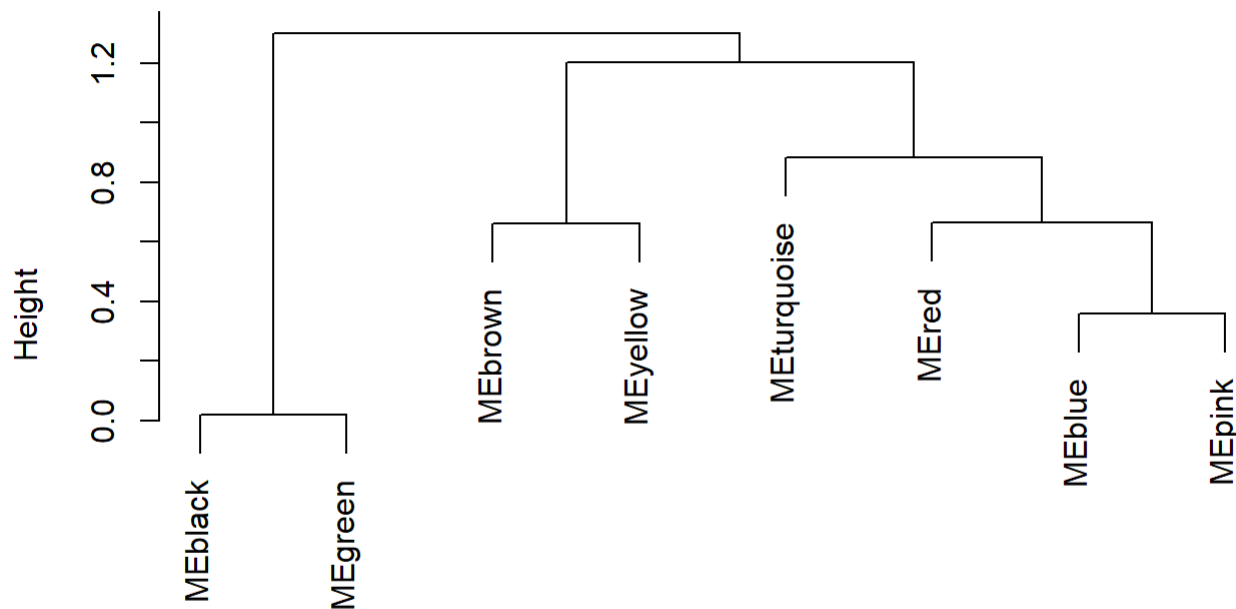
Calculate dissimilarity of module eigengenes

```
MEDiss <- 1-cor(MEs)  
METree <- hclust(as.dist(MEDiss), method = "average")
```

Plot the result

```
plot(METree, main = "Clustering of Module eigengenes",  
     xlab = "", sub = "")
```

Clustering of Module eigengenes



9. Show the correlation between modules and each sample

Get Module Eigengenes per cluster

```
MEs0 <- moduleEigengenes(norm.counts, mergedColors)$eigengenes
```

#Reorder modules so similar modules are next to each other

```
MEs0 <- orderMEs(MEs0)
module_order <- names(MEs0) %>% gsub("ME","", .)
```

Add the treatment names

```
MEs0$treatment <- row.names(MEs0)
```

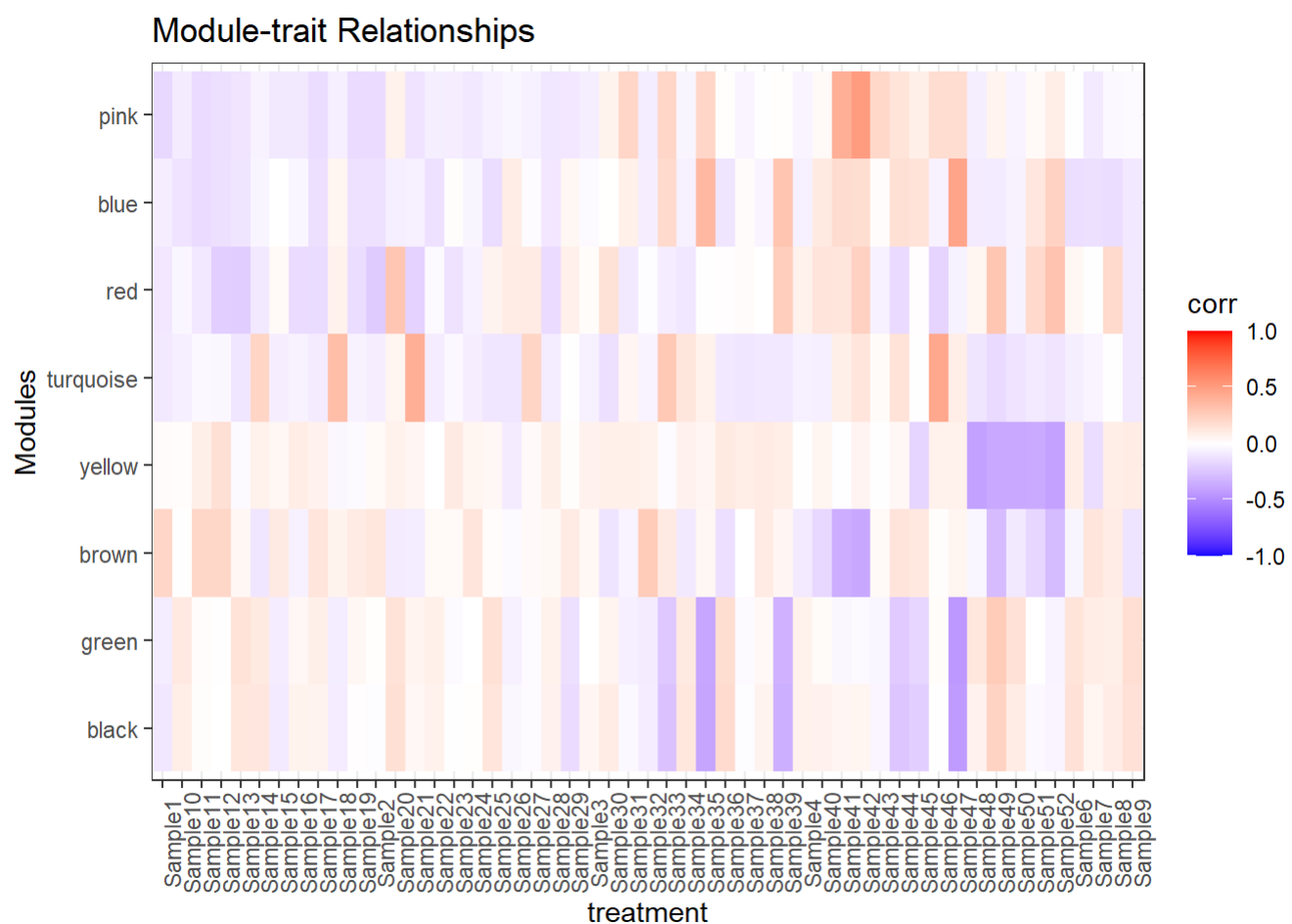
Tidy & plot data


```

mME = MEs0 %>%
  pivot_longer(-treatment) %>%
  mutate(
    name = gsub("ME", "", name),
    name = factor(name, levels = module_order)
  )

mME %>% ggplot(., aes(x=treatment, y=name, fill=value)) +
  geom_tile() +
  theme_bw() +
  scale_fill_gradient2(
    low = "blue",
    high = "red",
    mid = "white",
    midpoint = 0,
    limit = c(-1,1)) +
  theme(axis.text.x = element_text(angle=90)) +
  labs(title = "Module-trait Relationships", y = "Modules", fill="corr")

```



10. Showing gene expression level in specific modules

```
modules_of_interest = c("green", "turquoise", "red")
```

Pull out list of genes in that module

```
submod = module_df %>%
  subset(colors %in% modules_of_interest)

row.names(module_df) = module_df$gene_id
```

Get normalized expression for those genes

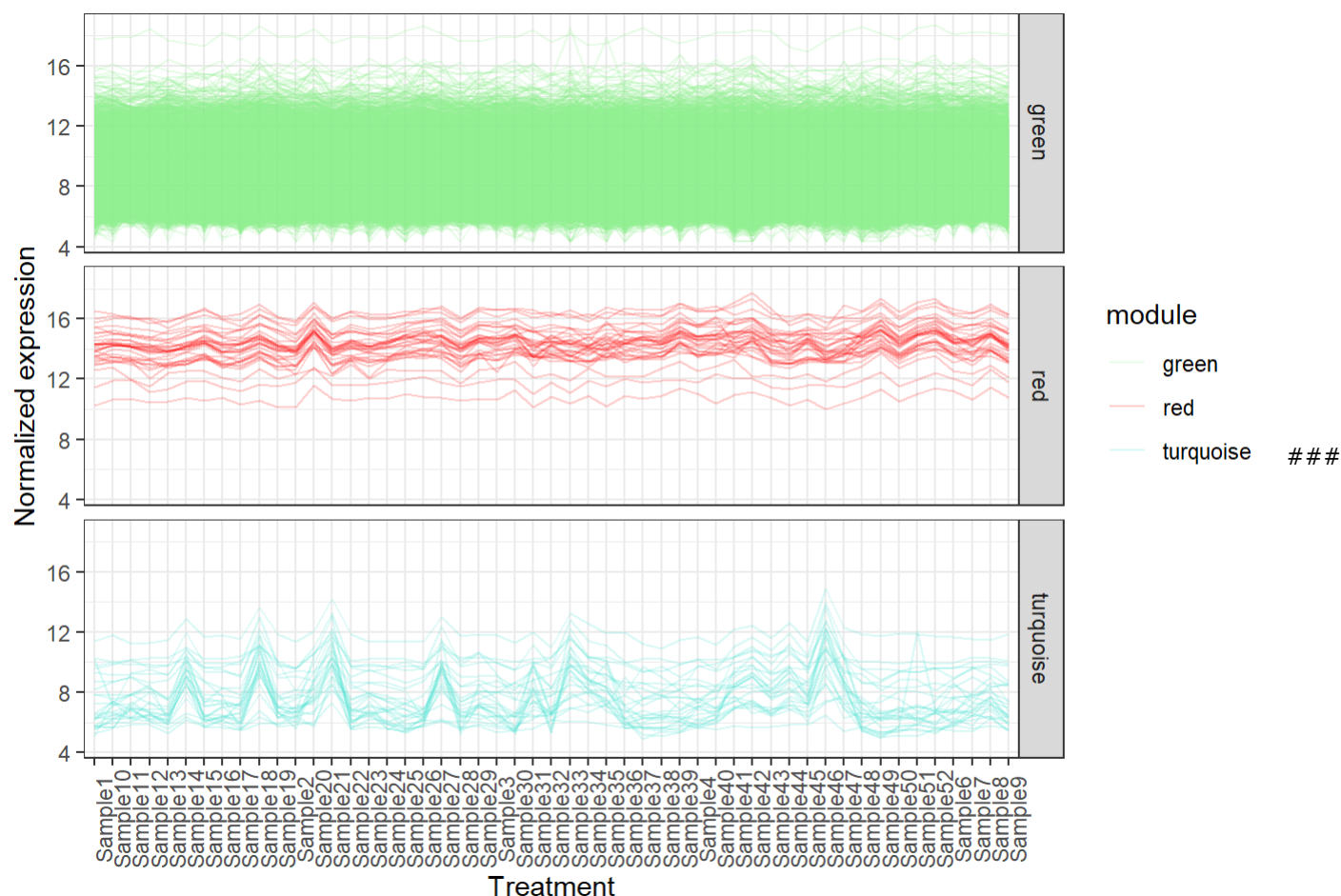
```
norm.counts[1:5,1:10]
```

```
##          ENSG00000000419 ENSG00000000457 ENSG00000000460 ENSG00000000938
## Sample1          7.817808          9.010102          8.278295          13.14962
## Sample2          7.860089          9.828239          8.520372          13.56801
## Sample3          8.152240          8.428289          8.494442          11.96361
## Sample4          8.430191          9.276900          8.532905          12.65407
## Sample6          8.316706          8.996154          8.293994          12.79312
##          ENSG00000001036 ENSG00000001084 ENSG00000001167 ENSG00000001460
## Sample1          7.925833          9.816597          9.853574          7.460652
## Sample2          8.382598          9.414906          9.628366          7.404398
## Sample3          8.657241          9.982769          9.601348          6.924517
## Sample4          8.300237          10.903511          10.299930          7.517321
## Sample6          8.890079          10.308196          9.615493          7.567784
##          ENSG00000001461 ENSG00000001497
## Sample1          9.763363          10.061901
## Sample2          10.019682          9.610658
## Sample3          9.858863          10.066467
## Sample4          9.777715          10.017410
## Sample6          9.932600          10.083488
```

```
norm.exprs<- data.frame(t(norm.counts))
norm.exprs$gene_id<- row.names(norm.exprs)
norm.exprs<- norm.exprs %>% select(gene_id,everything())
subexpr <- norm.exprs[submod$gene_id,]

submod_df <- data.frame(subexpr) %>%
  mutate(
    gene_id = row.names(.)
  ) %>%
  pivot_longer(-gene_id) %>%
  mutate(
    module = module_df[gene_id,]$colors
  )

submod_df %>% ggplot(., aes(x=name, y=value, group=gene_id, colour=module)) +
  geom_line(alpha=0.2) +
  theme_bw() +
  theme(
    axis.text.x = element_text(angle = 90)
  ) +
  facet_grid(rows = vars(module)) +
  labs(x = "Treatment",
       y = "Normalized expression") +
  scale_color_manual(values = c("lightgreen","red","turquoise"))
```



11. Finding the association between module and traits # Prepare and manipulate data for association analysis # Converting categorical variable into binary group variables

```
type <- Sample_info_final %>%
  mutate(type = ifelse(grepl('Lung cancer', Group), 1, 0)) %>%
  select(4)
```

```
Sample_info_final$Status <- factor(Sample_info_final$Status,
  levels = c("None", "Newly diagnosed",
    "Post Chemotherapy", "On Doxorubicin"))
```

```
Status<- binarizeCategoricalColumns(Sample_info_final$Status,
  includePairwise = FALSE,
  includeLevelVsAll = TRUE,
  minCount = 1)
```

```
Status.final<- cbind(type, Status)
```

Define numbers of genes and samples and draw correlation

```
nSamples <- nrow(norm.counts)
nGenes <- ncol(norm.counts)
status.module.corr <- cor(module_eigengenes, Status.final, use = 'p')
status.module.corr.pvals <- corPvalueStudent(status.module.corr , nSamples)
nrow(module_eigengenes)
```

```
## [1] 51
```

Visualize module-trait association as a heatmap

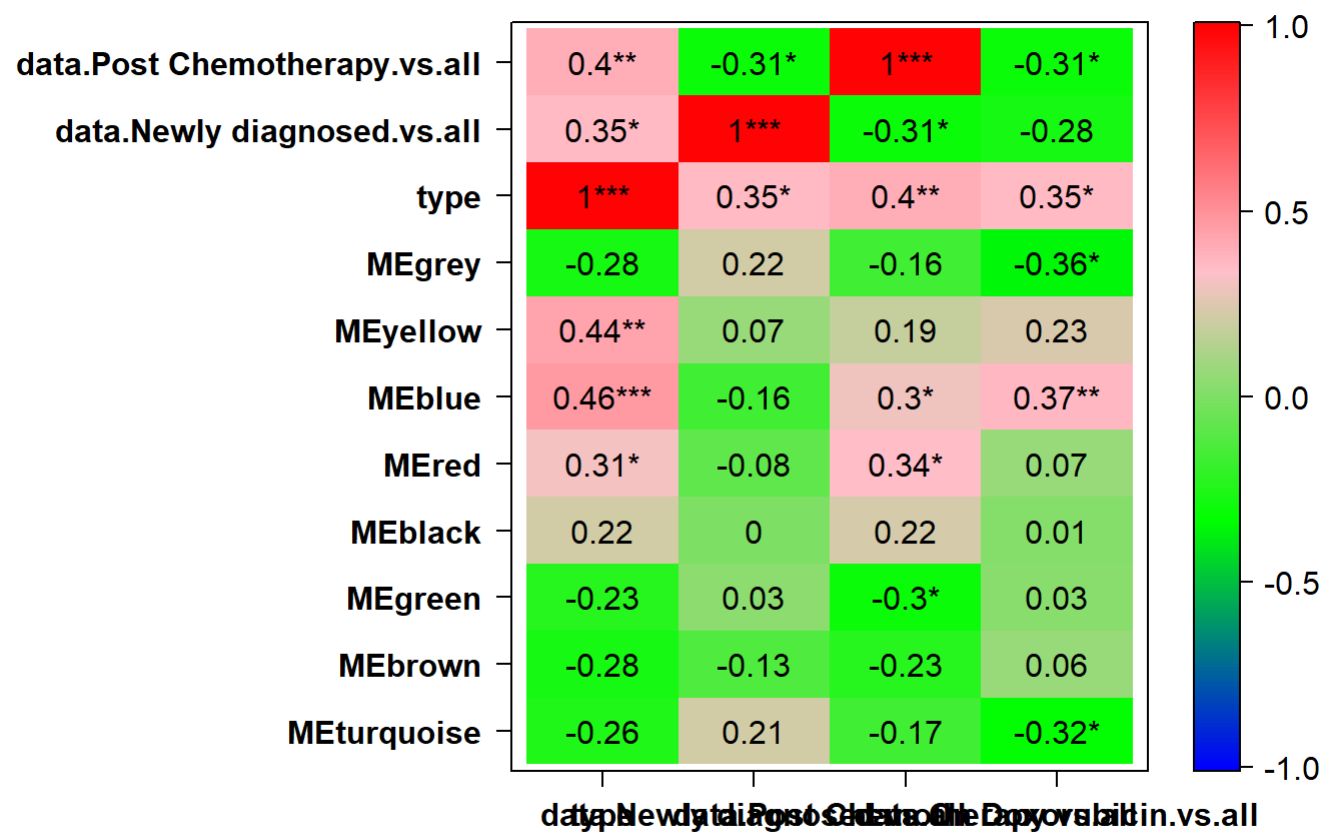
```
heatmap.data <- merge(module_eigengenes, Status.final, by = 'row.names')

head(heatmap.data)
```

```
## Row.names MEturquoise MEbrown MEgreen MEblack MERed
## 1 Sample1 -0.1037587873 0.21685398 0.02054254 -0.09222291 -0.10019226
## 2 Sample10 0.0895341492 0.00293838 0.01367637 -0.06928702 -0.03836467
## 3 Sample11 0.0099912350 0.20925609 0.08402187 -0.02660053 -0.09621342
## 4 Sample12 0.0006960986 0.20967381 0.16100147 -0.03369510 -0.20749830
## 5 Sample13 0.1272433032 0.04066019 -0.02093506 -0.10491016 -0.21409801
## 6 Sample14 0.1288343520 -0.10802289 0.07607466 0.22150429 -0.09486251
## MEblue MEyellow MEgrey type data.Newly diagnosed.vs.all
## 1 -0.06997022 -0.16785266 -0.082395871 0 0
## 2 -0.11979305 -0.08319146 0.116674013 0 0
## 3 -0.15188447 -0.15462433 0.014037255 0 0
## 4 -0.13145467 -0.13190268 -0.003316252 0 0
## 5 -0.10896400 -0.11249850 0.148243795 0 0
## 6 -0.04248159 -0.05396714 0.110840769 0 0
## data.Post Chemotherapy.vs.all data.On Doxorubicin.vs.all
## 1 0 0
## 2 0 0
## 3 0 0
## 4 0 0
## 5 0 0
## 6 0 0
```

```
heatmap.data <- heatmap.data %>%
  column_to_rownames(var = 'Row.names')

CorLevelPlot(heatmap.data,
  x = names(heatmap.data)[9:12],
  y = names(heatmap.data)[1:11],
  col = c("blue1", "green", "pink", "red"))
```



```
module.gene.mapping <- as.data.frame(bwnet$colors)
module.gene.mapping %>%
  filter(`bwnet$colors` == 'blue') %>%
  rownames()
```

```

## [1] "ENSG0000005483" "ENSG0000006607" "ENSG0000010072" "ENSG0000010165"
## [5] "ENSG0000029363" "ENSG0000035141" "ENSG0000043093" "ENSG0000044574"
## [9] "ENSG0000048649" "ENSG0000051382" "ENSG0000051620" "ENSG0000053254"
## [13] "ENSG0000053770" "ENSG0000055070" "ENSG0000056586" "ENSG0000059769"
## [17] "ENSG0000067064" "ENSG0000067082" "ENSG0000069345" "ENSG0000069956"
## [21] "ENSG0000070831" "ENSG0000072415" "ENSG0000075568" "ENSG0000076053"
## [25] "ENSG0000076248" "ENSG0000077152" "ENSG0000079332" "ENSG0000080371"
## [29] "ENSG0000080824" "ENSG0000081320" "ENSG0000084463" "ENSG0000087074"
## [33] "ENSG0000088682" "ENSG0000089597" "ENSG0000090432" "ENSG0000090863"
## [37] "ENSG0000091527" "ENSG0000092094" "ENSG0000092871" "ENSG0000095002"
## [41] "ENSG0000095370" "ENSG0000095787" "ENSG0000099246" "ENSG0000099797"
## [45] "ENSG0000100221" "ENSG0000100354" "ENSG0000100393" "ENSG0000100395"
## [49] "ENSG0000100401" "ENSG0000100532" "ENSG0000100603" "ENSG0000100614"
## [53] "ENSG0000100982" "ENSG0000101166" "ENSG0000101337" "ENSG0000101367"
## [57] "ENSG0000103061" "ENSG0000103415" "ENSG0000103429" "ENSG0000104164"
## [61] "ENSG0000104312" "ENSG0000104889" "ENSG0000105835" "ENSG0000105856"
## [65] "ENSG0000106245" "ENSG0000106263" "ENSG0000106615" "ENSG0000106682"
## [69] "ENSG0000107560" "ENSG0000107897" "ENSG0000107937" "ENSG0000108061"
## [73] "ENSG0000108106" "ENSG0000108829" "ENSG0000108960" "ENSG0000109332"
## [77] "ENSG0000110696" "ENSG0000111802" "ENSG0000112081" "ENSG0000112118"
## [81] "ENSG0000112149" "ENSG0000112245" "ENSG0000113163" "ENSG0000113575"
## [85] "ENSG0000113734" "ENSG0000113811" "ENSG0000114125" "ENSG0000114796"
## [89] "ENSG0000114988" "ENSG0000115165" "ENSG0000115520" "ENSG0000115840"
## [93] "ENSG0000116455" "ENSG0000116731" "ENSG0000116747" "ENSG0000116752"
## [97] "ENSG0000116830" "ENSG0000116906" "ENSG0000117036" "ENSG0000117505"
## [101] "ENSG0000117614" "ENSG0000118894" "ENSG0000119392" "ENSG0000119559"
## [105] "ENSG0000119725" "ENSG0000119801" "ENSG0000119899" "ENSG0000119929"
## [109] "ENSG0000119950" "ENSG0000119953" "ENSG0000120690" "ENSG0000120705"
## [113] "ENSG0000120709" "ENSG0000120727" "ENSG0000121578" "ENSG0000121864"
## [117] "ENSG0000122068" "ENSG0000122257" "ENSG0000122882" "ENSG0000123091"
## [121] "ENSG0000123505" "ENSG0000123728" "ENSG0000124209" "ENSG0000124380"
## [125] "ENSG0000125835" "ENSG0000126524" "ENSG0000126945" "ENSG0000128245"
## [129] "ENSG0000128590" "ENSG0000128881" "ENSG0000128989" "ENSG0000129235"
## [133] "ENSG0000129315" "ENSG0000129691" "ENSG0000130311" "ENSG0000130522"
## [137] "ENSG0000130803" "ENSG0000130844" "ENSG0000131263" "ENSG0000131323"
## [141] "ENSG0000131381" "ENSG0000132507" "ENSG0000132603" "ENSG0000132661"
## [145] "ENSG0000132823" "ENSG0000133026" "ENSG0000133606" "ENSG0000133773"
## [149] "ENSG0000134375" "ENSG0000134644" "ENSG0000134686" "ENSG0000134970"
## [153] "ENSG0000135334" "ENSG0000136527" "ENSG0000136819" "ENSG0000137075"
## [157] "ENSG0000137502" "ENSG0000137575" "ENSG0000137876" "ENSG0000137947"
## [161] "ENSG0000138032" "ENSG0000138069" "ENSG0000138166" "ENSG0000138433"

```

12. Identifying significant genes against different features of data

Calculating the module membership and the associated p-values

The module membership/intramodular connectivity is calculated as the correlation of the eigengene and the gene expression profile.

This quantifies the similarity of all genes on the array to every module.

```
module.membership.measure <- cor(module_eigengenes, norm.counts, use = 'p')
module.membership.measure.pvals <- corPvalueStudent(module.membership.measure, nSamples)

module.membership.measure.pvals[1:5,1:10]
```

```
##          ENSG00000000419 ENSG00000000457 ENSG00000000460 ENSG00000000938
## MEturquoise      0.08179877    3.935824e-06    7.873861e-01    0.16687163
## MEbrown          0.01031164    3.801793e-01    1.015938e-01    0.02125898
## MEgreen          0.27024680    2.434798e-02    4.255289e-05    0.07970018
## MEblack          0.95418281    7.291502e-01    3.064485e-01    0.46636890
## MERed            0.06741831    1.364833e-01    6.076142e-02    0.03317124
##          ENSG00000001036 ENSG00000001084 ENSG00000001167 ENSG00000001460
## MEturquoise      1.066509e-05    1.211570e-02    0.1373691154    0.03960014
## MEbrown          1.158562e-02    7.932893e-01    0.0255369951    0.02068358
## MEgreen          3.976450e-02    2.249371e-07    0.0006352417    0.77101215
## MEblack          3.100530e-01    6.265870e-02    0.2118802323    0.83164735
## MERed            7.442618e-01    8.543609e-02    0.4044932254    0.89784399
##          ENSG00000001461 ENSG00000001497
## MEturquoise      2.468861e-05    0.0001716473
## MEbrown          8.252186e-01    0.7602053664
## MEgreen          4.901483e-01    0.1947561939
## MEblack          5.127292e-04    0.7311565906
## MERed            6.122777e-01    0.8786558002
```

Calculating the gene significance and associated p-values against a particular trait

```
gene.signf.corr <- cor(norm.counts, Status.final$'data.Post Chemotherapy.vs.all', use =
  'p')
gene.signf.corr.pvals <- corPvalueStudent(gene.signf.corr, nSamples)
```

Sub-setting based on significant association: X1:corr, X2:pval


```
Sig_chemo<-data.frame(cbind(gene.signf.corr, gene.signf.corr.pvals))  
Sig_chemo_final<- filter(Sig_chemo, Sig_chemo$X1> 0.3 | Sig_chemo$X1< -0.3  
                        & Sig_chemo$X2<0.05)  
  
write.csv(Sig_chemo_final, 'sig_genes_chemo.csv')
```