WGCNA Analyses

Run WGCNA

Setup and read in data

```
# Libraries
library(easypackages)
libraries("WGCNA","gplots","here","ggplot2")
```

```
## ======================================================================
## *
## *  Package WGCNA 1.63 loaded.
## *
## *    Important note: It appears that your system supports multi-threading,
## *    but it is not enabled within WGCNA in R.
## *    To allow multi-threading within WGCNA with all available cores, use
## *
## *          allowWGCNAThreads()
## *
## *    within R. Use disableWGCNAThreads() to disable threading if necessary.
## *    Alternatively, set the following environment variable on your system:
## *
## *          ALLOW_WGCNA_THREADS=<number_of_processors>
## *
## *    for example
## *
## *          ALLOW_WGCNA_THREADS=8
## *
## *    To set the environment variable in linux bash shell, type
## *
## *           export ALLOW_WGCNA_THREADS=8
## *
## *     before running R. Other operating systems or shells will
## *     have a similar command to achieve the same aim.
## *
## ======================================================================
```

```
# Allow multi-threading within WGCNA
allowWGCNAThreads()
```

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

```
options(stringsAsFactors = FALSE)

# WGCNA parameters
networkType = 'signed'
tomType = 'signed'
corrType = 'bicor'
maxBlockSize = 30000
minModSize = 100
modMergeCutHeight = 0.20
```

```
deepSplit = 4

resultpath = here("WGCNAresults")
dir.create(resultpath)

# Read in data
load(here("data","processed","exprDataAdj.Rdata"))

datExpr = t(exprDataAdj)
datTraits = as.data.frame(labelData[,c("subgrp2",
                                       "gex_age",
                                       "sex",
                                       "batch",
                                       "RIN")])
rownames(datTraits) = labelData$exprColNames
datTraits$sex = as.numeric(factor(datTraits$sex))
datTraits$subgrp2 = as.numeric(factor(datTraits$subgrp2))
datTraits$batch = as.numeric(factor(datTraits$batch))
```

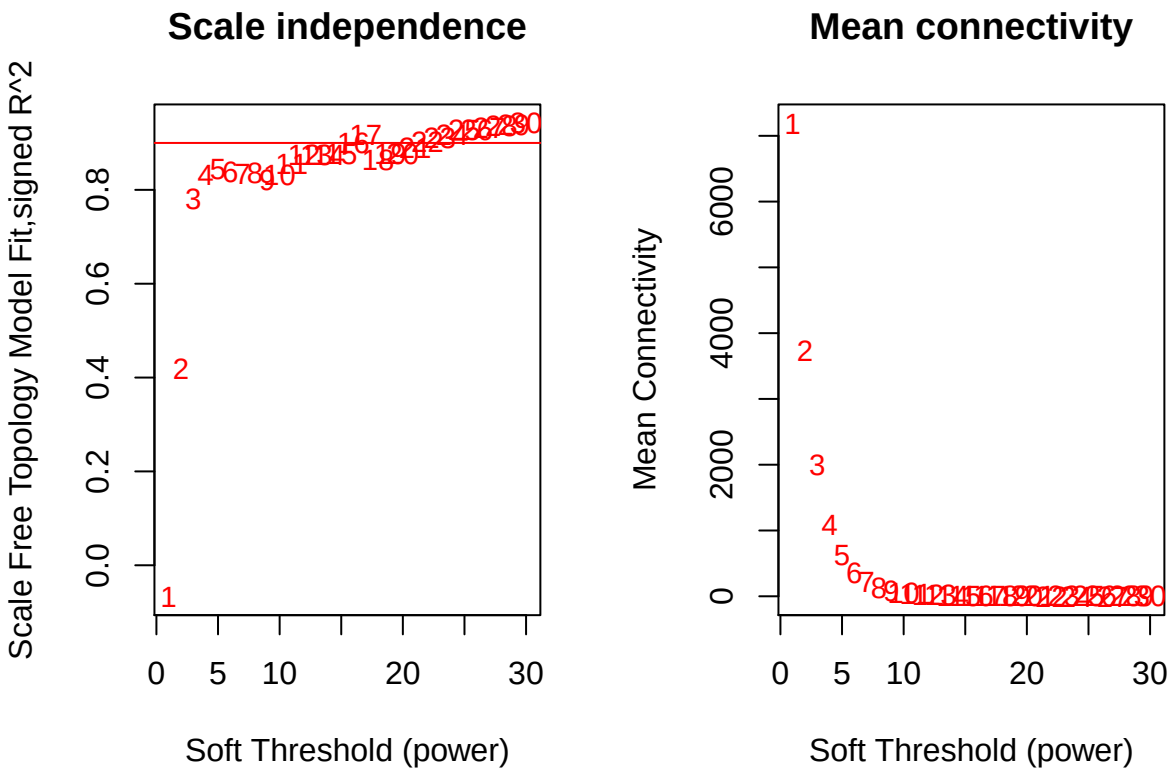
Choose soft-threshold power

```
powers = c(1:30)
if (corrType=="pearson"){
  corFnc2use = "cor"
}else if (corrType=="bicor"){
  corFnc2use = corrType
}
sft = pickSoftThreshold(datExpr,
                        powerVector = powers,
                        verbose = 5,
                        networkType = networkType,
                        corFnc = corrType)
```

```
## pickSoftThreshold: will use block size 3125.
##  pickSoftThreshold: calculating connectivity for given powers...
##    ..working on genes 1 through 3125 of 14313
##    ..working on genes 3126 through 6250 of 14313
##    ..working on genes 6251 through 9375 of 14313
##    ..working on genes 9376 through 12500 of 14313
##    ..working on genes 12501 through 14313 of 14313
##    Power SFT.R.sq  slope truncated.R.sq  mean.k. median.k. max.k.
## 1      1    0.066   8.57          0.958 7190.000  7.19e+03 7550.0
## 2      2    0.420 -12.20          0.833 3730.000  3.68e+03 4300.0
## 3      3    0.780 -10.90          0.935 1990.000  1.95e+03 2610.0
## 4      4    0.833  -7.76          0.972 1090.000  1.06e+03 1680.0
## 5      5    0.843  -5.94          0.984  617.000  5.88e+02 1120.0
## 6      6    0.839  -4.80          0.983  358.000  3.34e+02  771.0
## 7      7    0.834  -4.06          0.980  214.000  1.93e+02  547.0
## 8      8    0.836  -3.49          0.978  131.000  1.14e+02  397.0
## 9      9    0.822  -3.11          0.965   82.500  6.86e+01  294.0
## 10    10    0.831  -2.75          0.963   53.300  4.20e+01  221.0
## 11    11    0.856  -2.45          0.974   35.400  2.62e+01  169.0
```

```
## 12    12    0.876  -2.25          0.984   24.000  1.66e+01  132.0
## 13    13    0.874  -2.20          0.984   16.700  1.06e+01  109.0
## 14    14    0.875  -2.14          0.981   11.900  6.89e+00   90.5
## 15    15    0.876  -2.07          0.975    8.640  4.52e+00   75.9
## 16    16    0.900  -1.95          0.985    6.400  2.99e+00   64.1
## 17    17    0.917  -1.89          0.990    4.820  2.01e+00   55.8
## 18    18    0.865  -1.99          0.956    3.690  1.36e+00   51.7
## 19    19    0.876  -1.99          0.965    2.870  9.34e-01   48.1
## 20    20    0.876  -1.99          0.965    2.270  6.43e-01   44.9
## 21    21    0.889  -1.96          0.968    1.810  4.47e-01   41.9
## 22    22    0.902  -1.93          0.972    1.460  3.14e-01   39.2
## 23    23    0.910  -1.89          0.971    1.200  2.22e-01   36.7
## 24    24    0.918  -1.85          0.969    0.987  1.57e-01   34.5
## 25    25    0.927  -1.81          0.974    0.822  1.12e-01   32.4
## 26    26    0.927  -1.78          0.973    0.691  8.02e-02   30.4
## 27    27    0.932  -1.75          0.972    0.586  5.76e-02   28.6
## 28    28    0.935  -1.71          0.970    0.500  4.18e-02   27.0
## 29    29    0.939  -1.68          0.969    0.430  3.03e-02   25.4
## 30    30    0.942  -1.65          0.967    0.372  2.22e-02   23.9
```

```r
makeSoftPowerPlot <- function(sft, powers, cex1 = 0.9){
  # Scale-free topology fit index as a function of the soft-thresholding power
  par(mfrow = c(1,2))
  plot(sft$fitIndices[,1],
       -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
       xlab = "Soft Threshold (power)",
       ylab = "Scale Free Topology Model Fit,signed R^2",
       type = "n",
       main = paste("Scale independence"))
  text(sft$fitIndices[,1],
       -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
       labels = powers,
       cex = cex1,
       col = "red")
  abline(h = 0.90, col = "red")
  ## Mean connectivity as a function of the soft-thresholding power
  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=cex1,
       col="red")
}

makeSoftPowerPlot(sft = sft, powers = powers)
```



Run blockwiseModules

```
softPower = 16

## Run an automated network analysis
net3 = blockwiseModules(datExpr,
                        power = softPower,
                        deepSplit = deepSplit,
                        minModuleSize = minModSize,
                        mergeCutHeight = modMergeCutHeight,
                        detectCutHeight = 0.9999,
                        corType = corrType,
                        networkType = networkType,
                        pamStage = FALSE,
                        pamRespectsDendro = TRUE,
                        verbose = 3,
                        saveTOMs = FALSE,
                        maxBlockSize = maxBlockSize,
                        numericLabels = TRUE)
```

```
##  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 use 8 parallel threads.
##      Fraction of slow calculations: 0.000000
##     ..connectivity..
```

```
##     ..matrix multiplication (system BLAS)..
##     ..normalization..
##     ..done.
##  ....clustering..
##  ....detecting modules..
##  ....calculating module eigengenes..
##  ....checking kME in modules..
##      ..removing 62 genes from module 1 because their KME is too low.
##      ..removing 226 genes from module 2 because their KME is too low.
##      ..removing 1 genes from module 3 because their KME is too low.
##      ..removing 49 genes from module 4 because their KME is too low.
##      ..removing 7 genes from module 5 because their KME is too low.
##      ..removing 46 genes from module 6 because their KME is too low.
##      ..removing 1 genes from module 7 because their KME is too low.
##      ..removing 27 genes from module 8 because their KME is too low.
##      ..removing 4 genes from module 9 because their KME is too low.
##      ..removing 15 genes from module 11 because their KME is too low.
##      ..removing 27 genes from module 13 because their KME is too low.
##      ..removing 10 genes from module 14 because their KME is too low.
##      ..removing 1 genes from module 18 because their KME is too low.
##   ..reassigning 42 genes from module 1 to modules with higher KME.
##   ..reassigning 2 genes from module 2 to modules with higher KME.
##   ..reassigning 18 genes from module 3 to modules with higher KME.
##   ..reassigning 9 genes from module 4 to modules with higher KME.
##   ..reassigning 13 genes from module 5 to modules with higher KME.
##   ..reassigning 3 genes from module 6 to modules with higher KME.
##   ..reassigning 3 genes from module 7 to modules with higher KME.
##   ..reassigning 9 genes from module 8 to modules with higher KME.
##   ..reassigning 8 genes from module 9 to modules with higher KME.
##   ..reassigning 8 genes from module 10 to modules with higher KME.
##   ..reassigning 7 genes from module 11 to modules with higher KME.
##   ..reassigning 1 genes from module 12 to modules with higher KME.
##   ..reassigning 2 genes from module 14 to modules with higher KME.
##   ..reassigning 2 genes from module 15 to modules with higher KME.
##   ..reassigning 5 genes from module 16 to modules with higher KME.
##   ..reassigning 3 genes from module 17 to modules with higher KME.
##   ..reassigning 1 genes from module 19 to modules with higher KME.
##   ..reassigning 4 genes from module 20 to modules with higher KME.
##   ..reassigning 3 genes from module 22 to modules with higher KME.
##  ..merging modules that are too close..
##      mergeCloseModules: Merging modules whose distance is less than 0.2
##        Calculating new MEs...
```

```
net3$moduleNumbers = net3$colors
net3$colors = labels2colors(net3$moduleNumbers)

moduleLabels = net3$moduleNumbers
moduleColors = net3$colors
modNum_tab = data.frame(table(moduleLabels))
modCol_tab = data.frame(table(moduleColors))
modColNum_tab = cbind(moduleLabels = modNum_tab$moduleLabels,
                      modCol_tab[order(-modCol_tab$Freq),])
knitr::kable(modColNum_tab)
```

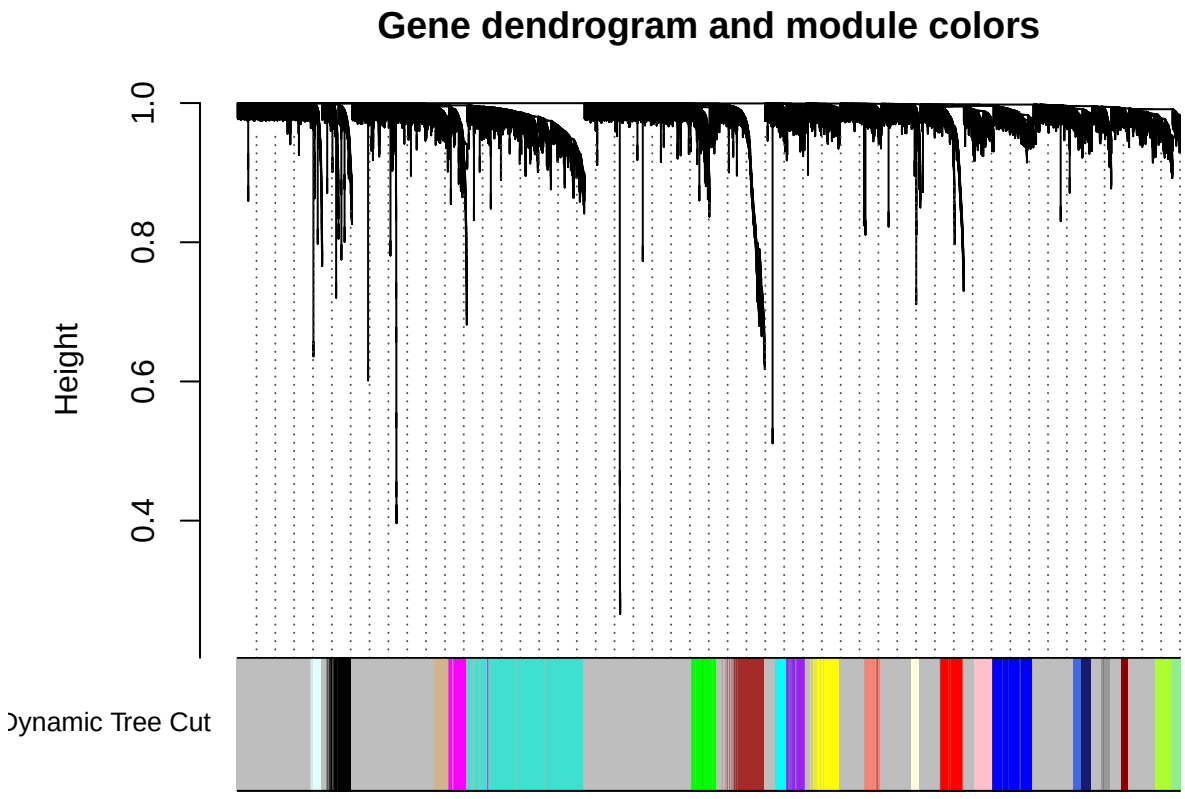
	moduleLabels	moduleColors	Freq
8	0	grey	7523
21	1	turquoise	1678
2	2	blue	590
3	3	brown	495
22	4	yellow	383
6	5	green	367
17	6	red	344
1	7	black	323
15	8	pink	292
13	9	magenta	264
16	10	purple	262
7	11	greenyellow	244
20	12	tan	241
19	13	salmon	231
4	14	cyan	159
14	15	midnightblue	147
9	16	grey60	140
10	17	lightcyan	140
11	18	lightgreen	128
12	19	lightyellow	126
18	20	royalblue	123
5	21	darkred	113

```
rownames(net3$MEs) = labelData$subjectId
tmp_MEs = net3$MEs
# rename columns in net3$MEs
for (i in 1:dim(tmp_MEs)[2]){
  tmp_mnum = substr(colnames(tmp_MEs)[i], 3, nchar(colnames(tmp_MEs)[i]))
  if (nchar(tmp_mnum)==1){
    new_mnum = sprintf("M0%s",tmp_mnum)
  } else if (nchar(tmp_mnum)==2){
    new_mnum = sprintf("M%s",tmp_mnum)
  }
  colnames(tmp_MEs)[i] = new_mnum
}
net3$MEs_colreordered = tmp_MEs[,sort(colnames(tmp_MEs))]
nums2use = 0:(dim(tmp_MEs)[2]-1)
for (i in 1:length(nums2use)){
  colnames(net3$MEs_colreordered)[i] = sprintf("M%d",nums2use[i])
}
if (sum(colnames(net3$MEs_colreordered)=="M0")>0){
  net3$MEs_colreordered = net3$MEs_colreordered[,2:ncol(net3$MEs_colreordered)]
}

# order rows by subgrp2 and then subjectId
new_label_data = labelData
new_label_data2 = new_label_data[order(new_label_data$subgrp2,
                                       order(new_label_data$subjectId)),]
net3$MEs_colreordered = net3$MEs_colreordered[new_label_data2$subjectId,]

# rename columns in net3$MEs
```

```
for (i in 1:dim(net3$MEs)[2]){
  tmp_mnum = substr(colnames(net3$MEs)[i], 3, nchar(colnames(net3$MEs)[i]))
  new_mnum = sprintf("M%s",tmp_mnum)
  colnames(net3$MEs)[i] = new_mnum
}
```

Make TOM plot

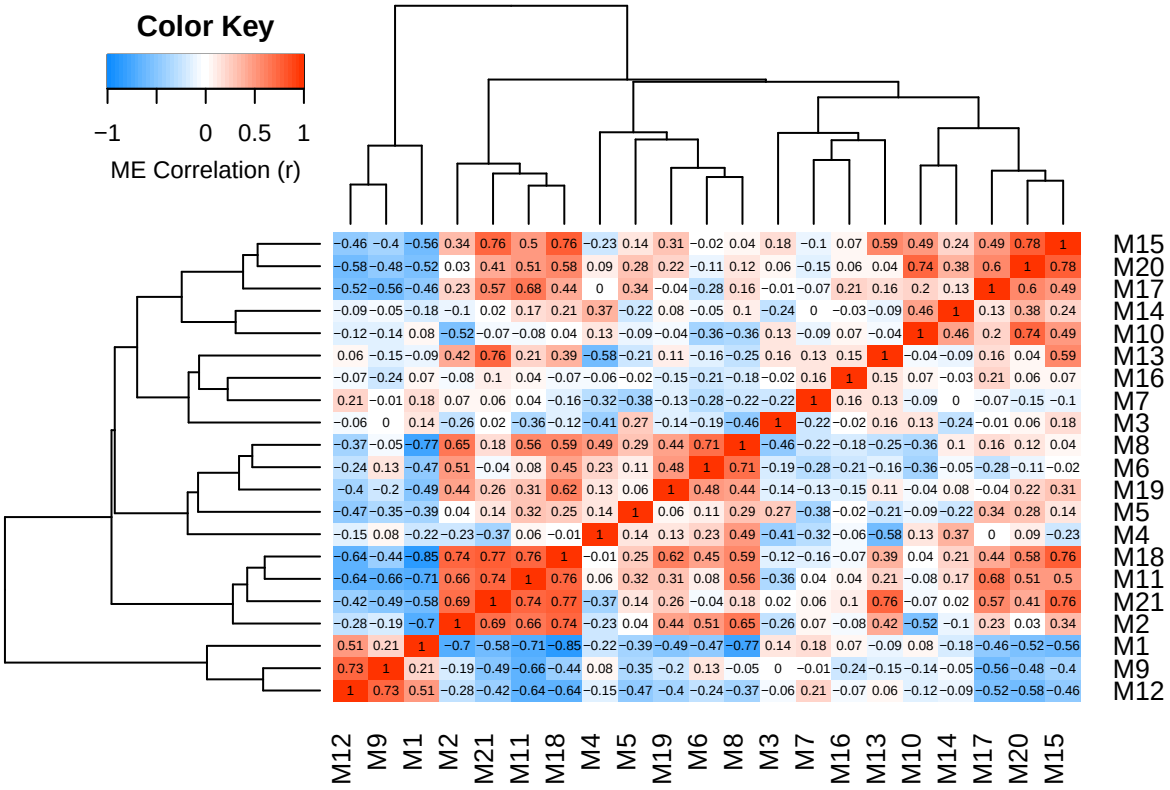
```
plotDendroAndColors(net3$dendrograms[[1]],
                    net3$colors, "Dynamic Tree Cut",
                    dendroLabels = FALSE,
                    hang = 0.03,
                    addGuide = TRUE,
                    guideHang = 0.05,
                    main = "Gene dendrogram and module colors")
```



Make eigengene network plot

```
MEcorMat = data.frame(cor(net3$MEs))
MEcorMat = MEcorMat[!(names(MEcorMat) %in% "MO"),
                    !(names(MEcorMat) %in% "MO")]

heatmap.2(as.matrix(MEcorMat),
          col = blueWhiteRed(50),
          Rowv = TRUE,
```

```
        trace = "none",
        cellnote= round(as.matrix(MEcorMat),digits = 2),
        notecol = "black",
        notecex=0.5,
        density.info = "none",
        key.xlab ="ME Correlation (r)")
```



Write out results files

```
# write ME files
fname2write = here("WGCNAresults","module_eigengenes.csv")
write.csv(net3$MEs,file = fname2write)
write.table(net3$MEs_colreordered,
            file = here("WGCNAresults","data4pls.txt"),
            quote = FALSE,
            row.names = FALSE,
            sep="\t",
            col.names = FALSE)

# compute module membership and write out to file
kme_data = signedKME(datExpr = datExpr, datME = net3$MEs)
for (i in 1:dim(net3$MEs)[2]){
  colnames(kme_data)[i] = sprintf("kME.%s",colnames(net3$MEs)[i])
}
geneInfo2use = geneInfo[,c("PROBE_ID","geneSymbol")]

wgcna_res_data = cbind(geneInfo2use,
```

```
                        data.frame(moduleLabels, moduleColors),
                        kme_data)
write.csv(wgcna_res_data,
          file = here("WGCNAresults", "wgcna_results_summary.csv"),
          quote = FALSE,
          col.names = FALSE,
          row.names = FALSE)
```

Run DE analysis on module eigengenes

```
medata = net3$MEs
nmods = dim(medata)[2]
medata$group = factor(labelData$subgrp2)
medata$sex = factor(labelData$sex)
medata$age = labelData$age
medata$Dx = factor(labelData$Dx)
medata$batch = factor(labelData$batch)
medata$RIN = labelData$RIN
covs2use = "group"

res_colnames = c("Module",
                 "Group.Fstat",
                 "Group.pval",
                 "Group.fdr",
                 "TD_vs_ASDPoor.tstat",
                 "TD_vs_ASDPoor.pval",
                 "TD_vs_ASDPoor.fdr",
                 "TD_vs_ASDGood.tstat",
                 "TD_vs_ASDGood.pval",
                 "TD_vs_ASDGood.fdr",
                 "ASDGood_vs_ASDPoor.tstat",
                 "ASDGood_vs_ASDPoor.pval",
                 "ASDGood_vs_ASDPoor.fdr")
group_diff_res = data.frame(matrix(nrow = nmods-1,
                                   ncol = length(res_colnames)))
colnames(group_diff_res) = res_colnames

for (imod in 1:(nmods-1)){
  module2use = sprintf("M%d",imod)
  group_diff_res$Module[imod] = module2use

  # test for subgroup effect
  form2use = as.formula(sprintf("%s ~ %s",module2use,covs2use))
  mod2use = lm(formula = form2use, data = medata)
  res = anova(mod2use)
  group_diff_res$Group.Fstat[imod] = res["group","F value"]
  group_diff_res$Group.pval[imod] = res["group","Pr(>F)"]

  # test for TD vs ASD Poor effect
  mask = medata$group=="TD" | medata$group=="Poor"
  tmp_data = subset(medata, mask)
  form2use = as.formula(sprintf("%s ~ %s",module2use,covs2use))
```

```
  mod2use = t.test(formula = form2use, data = tmp_data)
  group_diff_res$TD_vs_ASDPoor.tstat[imod] = mod2use$statistic
  group_diff_res$TD_vs_ASDPoor.pval[imod] = mod2use$p.value

  # test for TD vs ASD Good effect
  mask = medata$group=="TD" | medata$group=="Good"
  tmp_data = subset(medata, mask)
  form2use = as.formula(sprintf("%s ~ %s",module2use,covs2use))
  mod2use = t.test(formula = form2use, data = tmp_data)
  group_diff_res$TD_vs_ASDGood.tstat[imod] = mod2use$statistic
  group_diff_res$TD_vs_ASDGood.pval[imod] = mod2use$p.value

  # test for ASD Good vs ASD Poor effect
  mask = medata$group=="Good" | medata$group=="Poor"
  tmp_data = subset(medata, mask)
  form2use = as.formula(sprintf("%s ~ %s",module2use,covs2use))
  mod2use = t.test(formula = form2use, data = tmp_data)
  group_diff_res$ASDGood_vs_ASDPoor.tstat[imod] = mod2use$statistic
  group_diff_res$ASDGood_vs_ASDPoor.pval[imod] = mod2use$p.value
}#for (imod in 1:(nmods-1)){
rownames(group_diff_res) = group_diff_res$Module

# compute FDR
group_diff_res$Group.fdr = p.adjust(group_diff_res$Group.pval,
                                    method = "fdr")
group_diff_res$TD_vs_ASDPoor.fdr = p.adjust(group_diff_res$TD_vs_ASDPoor.pval,
                                            method = "fdr")
group_diff_res$TD_vs_ASDGood.fdr = p.adjust(group_diff_res$TD_vs_ASDGood.pval,
                                            method = "fdr")
group_diff_res$ASDGood_vs_ASDPoor.fdr = p.adjust(group_diff_res$ASDGood_vs_ASDPoor.pval,
                                                 method = "fdr")
group_diff_res
```

```
##     Module Group.Fstat Group.pval Group.fdr TD_vs_ASDPoor.tstat
## M1      M1  0.85466832 0.42810844 0.5993518          -1.3077840
## M2      M2  2.21274318 0.11403738 0.3204154           1.9583960
## M3      M3  3.48452314 0.03394472 0.1425678          -2.3282604
## M4      M4  0.99199585 0.37398634 0.5609795          -0.7802731
## M5      M5  0.15349680 0.85787894 0.9007729          -0.3953875
## M6      M6  1.23334291 0.29514108 0.5164969           1.0178269
## M7      M7  0.60950713 0.54536534 0.6673585           1.0717799
## M8      M8  0.56130056 0.57202155 0.6673585           0.8370422
## M9      M9  4.67901867 0.01112702 0.1366466          -1.9749269
## M10    M10  0.09927062 0.90557513 0.9055751           0.4028410
## M11    M11  4.50985599 0.01301396 0.1366466           2.5044641
## M12    M12  1.77808444 0.17357485 0.3645072          -1.5280577
## M13    M13  1.12424957 0.32844036 0.5305575           1.4686285
## M14    M14  0.45444935 0.63593248 0.7028727           0.6495977
## M15    M15  1.80609348 0.16892359 0.3645072           1.6993953
## M16    M16  0.63403431 0.53229143 0.6673585           1.0884492
## M17    M17  2.14668001 0.12153188 0.3204154           1.3515302
## M18    M18  3.67200087 0.02845258 0.1425678           2.4916716
## M19    M19  2.14215664 0.12206301 0.3204154           2.0033869
## M20    M20  1.26492125 0.28615870 0.5164969           1.5494964
```

```
## M21    M21  3.69217556 0.02791818 0.1425678          2.3337542
##     TD_vs_ASDPoor.pval TD_vs_ASDPoor.fdr TD_vs_ASDGood.tstat
## M1          0.19490314         0.3148435          -0.7133109
## M2          0.05391260         0.1617378           1.4702005
## M3          0.02257705         0.1185295          -2.3035095
## M4          0.43766928         0.5106142           0.5774464
## M5          0.69367516         0.6936752          -0.5418886
## M6          0.31201017         0.4095134          -0.3542066
## M7          0.28723281         0.4021259           0.5502798
## M8          0.40521200         0.5005560           0.9952571
## M9          0.05299963         0.1617378          -2.6683263
## M10         0.68823574         0.6936752           0.1001046
## M11         0.01459805         0.1185295           2.6205969
## M12         0.13148716         0.2761230          -1.5304563
## M13         0.14633954         0.2793755           1.0096012
## M14         0.51803237         0.5725621           1.0172965
## M15         0.09334369         0.2450272           0.7945324
## M16         0.27984237         0.4021259           0.3609575
## M17         0.18065763         0.3148435           2.0288055
## M18         0.01492681         0.1185295           1.9561098
## M19         0.04870371         0.1617378           1.7747514
## M20         0.12567776         0.2761230           0.9181648
## M21         0.02225434         0.1185295           2.3564695
##     TD_vs_ASDGood.pval TD_vs_ASDGood.fdr ASDGood_vs_ASDPoor.tstat
## M1          0.47787535         0.6690255              0.605344288
## M2          0.14594251         0.3405325             -0.631508970
## M3          0.02403018         0.1261584              0.007879917
## M4          0.56554160         0.6878272              1.394010742
## M5          0.58956615         0.6878272             -0.128465624
## M6          0.72440050         0.7606205             -1.587045747
## M7          0.58380490         0.6878272             -0.577615462
## M8          0.32287520         0.5650316              0.132805503
## M9          0.01005985         0.1115294             -0.914398070
## M10         0.92054642         0.9205464             -0.343384180
## M11         0.01062185         0.1115294              0.431908102
## M12         0.13071826         0.3405325             -0.042657185
## M13         0.31593678         0.5650316             -0.581530532
## M14         0.31247684         0.5650316              0.313983191
## M15         0.42985018         0.6447753             -1.208229109
## M16         0.71914725         0.7606205             -0.742428674
## M17         0.04632505         0.1905484              0.649485776
## M18         0.05444241         0.1905484             -0.743318261
## M19         0.08015987         0.2404796             -0.045094533
## M20         0.36158682         0.5841018             -0.670069422
## M21         0.02133914         0.1261584             -0.357707051
##     ASDGood_vs_ASDPoor.pval ASDGood_vs_ASDPoor.fdr
## M1                0.5466854              0.9890597
## M2                0.5295575              0.9890597
## M3                0.9937327              0.9937327
## M4                0.1677955              0.9890597
## M5                0.8981105              0.9937327
## M6                0.1171290              0.9890597
## M7                0.5651770              0.9890597
## M8                0.8946860              0.9937327
```

```
## M9                0.3633331             0.9890597
## M10               0.7322268             0.9900994
## M11               0.6670684             0.9900994
## M12               0.9660831             0.9937327
## M13               0.5626257             0.9890597
## M14               0.7543615             0.9900994
## M15               0.2310813             0.9890597
## M16               0.4600424             0.9890597
## M17               0.5179341             0.9890597
## M18               0.4595420             0.9890597
## M19               0.9641492             0.9937327
## M20               0.5047740             0.9890597
## M21               0.7216176             0.9900994
```

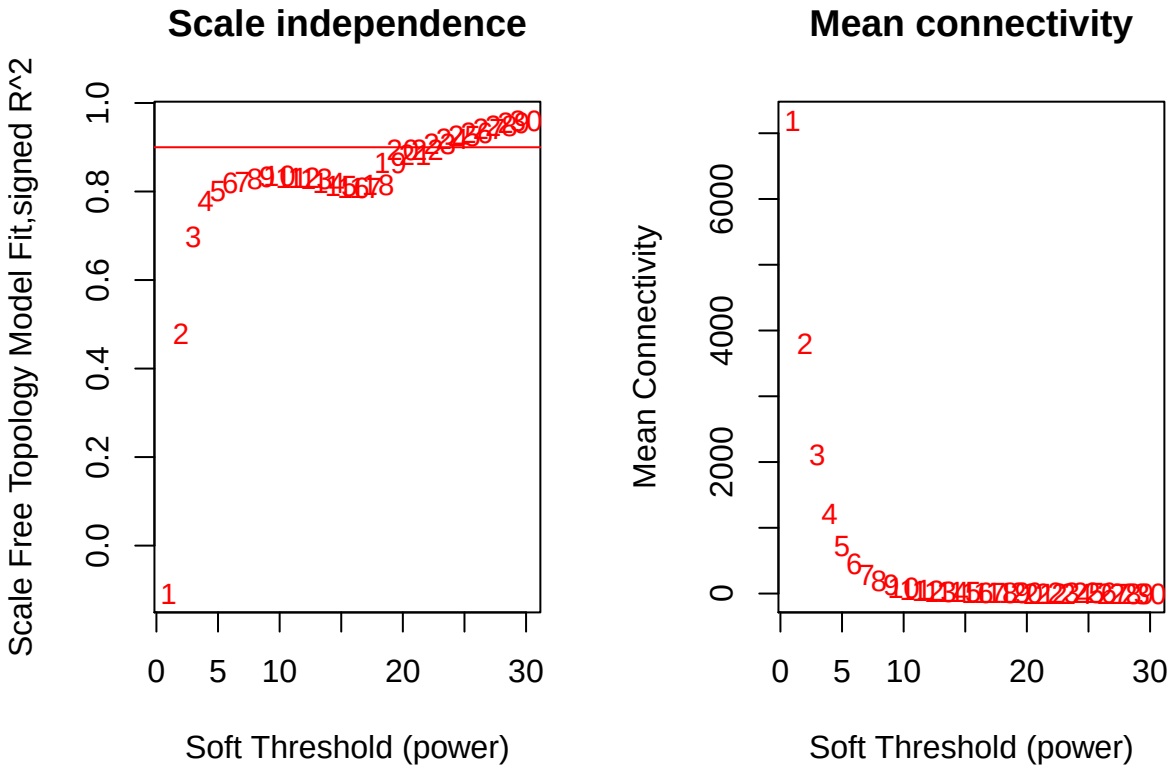
Run WGCNA on TD only

```
# run on TD
grp2use = "TD"
datExpr_grp = datExpr[labelData$subgrp2==grp2use,]
datTraits_grp = datTraits[labelData$subgrp2==grp2use,]

# Choose a soft-threshold power
powers = c(1:30)
if (corrType=="pearson"){
  corFnc2use = "cor"
}else if (corrType=="bicor"){
  corFnc2use = corrType
}
sft = pickSoftThreshold(datExpr_grp,
                        powerVector = powers,
                        verbose = 5,
                        networkType = networkType,
                        corFnc = corrType)
```

```
## pickSoftThreshold: will use block size 3125.
##  pickSoftThreshold: calculating connectivity for given powers...
##    ..working on genes 1 through 3125 of 14313
##    ..working on genes 3126 through 6250 of 14313
##    ..working on genes 6251 through 9375 of 14313
##    ..working on genes 9376 through 12500 of 14313
##    ..working on genes 12501 through 14313 of 14313
##    Power SFT.R.sq  slope truncated.R.sq  mean.k. median.k. max.k.
## 1      1    0.108   8.77          0.937 7190.000  7200.000 7580.0
## 2      2    0.479 -10.80          0.858 3800.000  3760.000 4440.0
## 3      3    0.696  -8.20          0.905 2100.000  2050.000 2800.0
## 4      4    0.779  -6.09          0.950 1210.000  1170.000 1860.0
## 5      5    0.800  -4.79          0.970  723.000   688.000 1290.0
## 6      6    0.819  -3.95          0.986  446.000   416.000  929.0
## 7      7    0.821  -3.50          0.986  284.000   257.000  687.0
## 8      8    0.828  -3.16          0.989  185.000   163.000  522.0
## 9      9    0.834  -2.91          0.990  124.000   105.000  404.0
## 10    10    0.836  -2.73          0.988   85.600    69.400  318.0
## 11    11    0.832  -2.60          0.987   60.100    46.500  253.0
```

```
## 12    12    0.832  -2.49          0.983   43.100    31.600  205.0
## 13    13    0.829  -2.40          0.980   31.500    21.900  167.0
## 14    14    0.820  -2.34          0.976   23.500    15.300  138.0
## 15    15    0.813  -2.28          0.973   17.700    10.800  115.0
## 16    16    0.808  -2.22          0.969   13.600     7.700   96.7
## 17    17    0.808  -2.14          0.965   10.600     5.560   81.8
## 18    18    0.815  -2.05          0.961    8.320     4.040   69.6
## 19    19    0.865  -1.90          0.978    6.630     2.960   59.6
## 20    20    0.894  -1.84          0.991    5.340     2.180   52.7
## 21    21    0.884  -1.90          0.990    4.340     1.620   49.3
## 22    22    0.895  -1.91          0.996    3.560     1.220   46.3
## 23    23    0.907  -1.91          0.998    2.950     0.916   43.6
## 24    24    0.919  -1.91          0.999    2.460     0.696   41.1
## 25    25    0.927  -1.90          0.998    2.070     0.530   38.8
## 26    26    0.933  -1.89          0.997    1.750     0.406   36.7
## 27    27    0.942  -1.86          0.996    1.490     0.313   34.7
## 28    28    0.947  -1.84          0.995    1.280     0.241   32.9
## 29    29    0.955  -1.82          0.997    1.100     0.189   31.2
## 30    30    0.960  -1.80          0.996    0.958     0.147   29.6
```

```
makeSoftPowerPlot(sft = sft, powers = powers)
```




```
softPower = 16

## Run an automated network analysis
net_tmp = blockwiseModules(datExpr,
                           power = softPower,
                           deepSplit = deepSplit,
                           minModuleSize = minModSize,
                           mergeCutHeight = modMergeCutHeight,
```

```
                            detectCutHeight = 0.9999,
                            corType = corrType,
                            networkType = networkType,
                            pamStage = FALSE,
                            pamRespectsDendro = TRUE,
                            verbose = 3,
                            saveTOMs = FALSE,
                            maxBlockSize = maxBlockSize,
                            numericLabels = TRUE)
```

```
##  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 use 8 parallel threads.
##      Fraction of slow calculations: 0.000000
##     ..connectivity..
##     ..matrix multiplication (system BLAS)..
##     ..normalization..
##     ..done.
##  ....clustering..
##  ....detecting modules..
##  ....calculating module eigengenes..
##  ....checking kME in modules..
##      ..removing 62 genes from module 1 because their KME is too low.
##      ..removing 226 genes from module 2 because their KME is too low.
##      ..removing 1 genes from module 3 because their KME is too low.
##      ..removing 49 genes from module 4 because their KME is too low.
##      ..removing 7 genes from module 5 because their KME is too low.
##      ..removing 46 genes from module 6 because their KME is too low.
##      ..removing 1 genes from module 7 because their KME is too low.
##      ..removing 27 genes from module 8 because their KME is too low.
##      ..removing 4 genes from module 9 because their KME is too low.
##      ..removing 15 genes from module 11 because their KME is too low.
##      ..removing 27 genes from module 13 because their KME is too low.
##      ..removing 10 genes from module 14 because their KME is too low.
##      ..removing 1 genes from module 18 because their KME is too low.
##    ..reassigning 42 genes from module 1 to modules with higher KME.
##    ..reassigning 2 genes from module 2 to modules with higher KME.
##    ..reassigning 18 genes from module 3 to modules with higher KME.
##    ..reassigning 9 genes from module 4 to modules with higher KME.
##    ..reassigning 13 genes from module 5 to modules with higher KME.
##    ..reassigning 3 genes from module 6 to modules with higher KME.
##    ..reassigning 3 genes from module 7 to modules with higher KME.
##    ..reassigning 9 genes from module 8 to modules with higher KME.
##    ..reassigning 8 genes from module 9 to modules with higher KME.
##    ..reassigning 8 genes from module 10 to modules with higher KME.
##    ..reassigning 7 genes from module 11 to modules with higher KME.
##    ..reassigning 1 genes from module 12 to modules with higher KME.
##    ..reassigning 2 genes from module 14 to modules with higher KME.
##    ..reassigning 2 genes from module 15 to modules with higher KME.
##    ..reassigning 5 genes from module 16 to modules with higher KME.
##    ..reassigning 3 genes from module 17 to modules with higher KME.
```

```
##   ..reassigning 1 genes from module 19 to modules with higher KME.
##   ..reassigning 4 genes from module 20 to modules with higher KME.
##   ..reassigning 3 genes from module 22 to modules with higher KME.
##  ..merging modules that are too close..
##      mergeCloseModules: Merging modules whose distance is less than 0.2
##        Calculating new MEs...
```

```
td_colors = labels2colors(net_tmp$colors)
datExpr_td = datExpr_grp
```

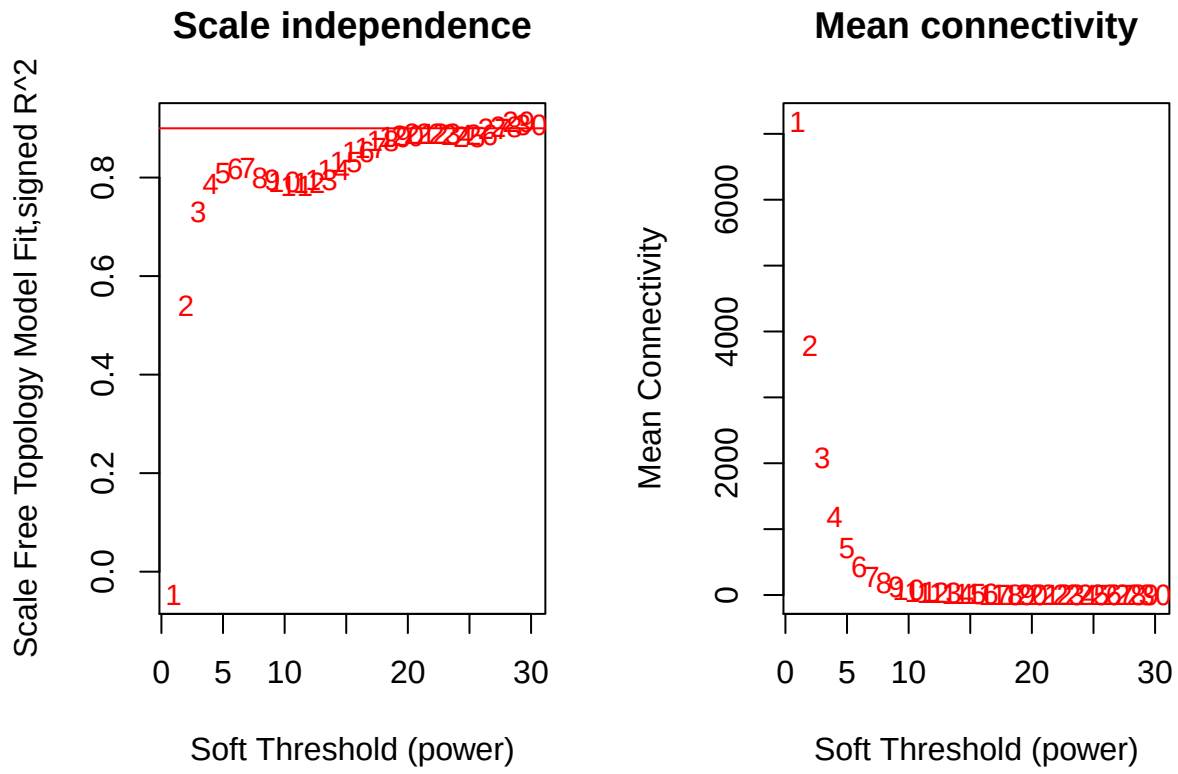
Run WGCNA on ASD Good only

```
# run on ASD Good
grp2use = "Good"
datExpr_grp = datExpr[labelData$subgrp2==grp2use,]
datTraits_grp = datTraits[labelData$subgrp2==grp2use,]

# Choose a soft-threshold power
powers = c(1:30)
if (corrType=="pearson"){
  corFnc2use = "cor"
}else if (corrType=="bicor"){
  corFnc2use = corrType
}
sft = pickSoftThreshold(datExpr_grp,
                        powerVector = powers,
                        verbose = 5,
                        networkType = networkType,
                        corFnc = corrType)
```

```
## pickSoftThreshold: will use block size 3125.
##  pickSoftThreshold: calculating connectivity for given powers...
##    ..working on genes 1 through 3125 of 14313
##    ..working on genes 3126 through 6250 of 14313
##    ..working on genes 6251 through 9375 of 14313
##    ..working on genes 9376 through 12500 of 14313
##    ..working on genes 12501 through 14313 of 14313
##    Power SFT.R.sq  slope truncated.R.sq  mean.k. median.k. max.k.
## 1      1   0.0473   7.65          0.929 7180.000  7.18e+03 7490.0
## 2      2   0.5400 -15.70          0.823 3780.000  3.73e+03 4380.0
## 3      3   0.7300 -10.50          0.905 2080.000  2.03e+03 2780.0
## 4      4   0.7880  -7.49          0.937 1190.000  1.14e+03 1880.0
## 5      5   0.8090  -5.81          0.950  704.000  6.59e+02 1320.0
## 6      6   0.8170  -4.68          0.957  430.000  3.92e+02  965.0
## 7      7   0.8190  -4.03          0.958  271.000  2.39e+02  726.0
## 8      8   0.7980  -3.68          0.948  176.000  1.48e+02  559.0
## 9      9   0.7960  -3.37          0.947  117.000  9.42e+01  439.0
## 10    10   0.7910  -3.15          0.944   79.500  6.07e+01  350.0
## 11    11   0.7840  -2.98          0.945   55.300  3.99e+01  282.0
## 12    12   0.7890  -2.79          0.945   39.300  2.66e+01  231.0
## 13    13   0.7950  -2.63          0.946   28.500  1.80e+01  190.0
## 14    14   0.8160  -2.47          0.953   21.000  1.23e+01  158.0
## 15    15   0.8320  -2.35          0.961   15.700  8.52e+00  133.0
```

```
## 16    16   0.8520  -2.23          0.972   11.900  5.98e+00  112.0
## 17    17   0.8610  -2.15          0.977    9.200  4.22e+00   95.1
## 18    18   0.8750  -2.06          0.981    7.170  3.02e+00   81.3
## 19    19   0.8830  -2.02          0.984    5.660  2.16e+00   71.0
## 20    20   0.8850  -1.99          0.988    4.520  1.57e+00   62.6
## 21    21   0.8890  -1.96          0.990    3.640  1.14e+00   55.4
## 22    22   0.8900  -1.93          0.990    2.960  8.38e-01   49.2
## 23    23   0.8880  -1.91          0.989    2.420  6.23e-01   43.8
## 24    24   0.8870  -1.89          0.988    2.000  4.66e-01   39.1
## 25    25   0.8820  -1.88          0.985    1.670  3.49e-01   35.0
## 26    26   0.8870  -1.85          0.986    1.390  2.64e-01   31.4
## 27    27   0.9000  -1.80          0.988    1.180  2.00e-01   28.3
## 28    28   0.9030  -1.78          0.987    0.997  1.52e-01   25.5
## 29    29   0.9130  -1.74          0.989    0.850  1.16e-01   23.1
## 30    30   0.9070  -1.75          0.989    0.729  8.92e-02   21.8
```

```
makeSoftPowerPlot(sft = sft, powers = powers)
```




```
softPower = 16

## Run an automated network analysis
net_tmp = blockwiseModules(datExpr,
                           power = softPower,
                           deepSplit = deepSplit,
                           minModuleSize = minModSize,
                           mergeCutHeight = modMergeCutHeight,
                           detectCutHeight = 0.9999,
                           corType = corrType,
                           networkType = networkType,
                           pamStage = FALSE,
```

```
                            pamRespectsDendro = TRUE,
                            verbose = 3,
                            saveTOMs = FALSE,
                            maxBlockSize = maxBlockSize,
                            numericLabels = TRUE)
```

```
##  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 use 8 parallel threads.
##      Fraction of slow calculations: 0.000000
##     ..connectivity..
##     ..matrix multiplication (system BLAS)..
##     ..normalization..
##     ..done.
##  ....clustering..
##  ....detecting modules..
##  ....calculating module eigengenes..
##  ....checking kME in modules..
##      ..removing 62 genes from module 1 because their KME is too low.
##      ..removing 226 genes from module 2 because their KME is too low.
##      ..removing 1 genes from module 3 because their KME is too low.
##      ..removing 49 genes from module 4 because their KME is too low.
##      ..removing 7 genes from module 5 because their KME is too low.
##      ..removing 46 genes from module 6 because their KME is too low.
##      ..removing 1 genes from module 7 because their KME is too low.
##      ..removing 27 genes from module 8 because their KME is too low.
##      ..removing 4 genes from module 9 because their KME is too low.
##      ..removing 15 genes from module 11 because their KME is too low.
##      ..removing 27 genes from module 13 because their KME is too low.
##      ..removing 10 genes from module 14 because their KME is too low.
##      ..removing 1 genes from module 18 because their KME is too low.
##    ..reassigning 42 genes from module 1 to modules with higher KME.
##    ..reassigning 2 genes from module 2 to modules with higher KME.
##    ..reassigning 18 genes from module 3 to modules with higher KME.
##    ..reassigning 9 genes from module 4 to modules with higher KME.
##    ..reassigning 13 genes from module 5 to modules with higher KME.
##    ..reassigning 3 genes from module 6 to modules with higher KME.
##    ..reassigning 3 genes from module 7 to modules with higher KME.
##    ..reassigning 9 genes from module 8 to modules with higher KME.
##    ..reassigning 8 genes from module 9 to modules with higher KME.
##    ..reassigning 8 genes from module 10 to modules with higher KME.
##    ..reassigning 7 genes from module 11 to modules with higher KME.
##    ..reassigning 1 genes from module 12 to modules with higher KME.
##    ..reassigning 2 genes from module 14 to modules with higher KME.
##    ..reassigning 2 genes from module 15 to modules with higher KME.
##    ..reassigning 5 genes from module 16 to modules with higher KME.
##    ..reassigning 3 genes from module 17 to modules with higher KME.
##    ..reassigning 1 genes from module 19 to modules with higher KME.
##    ..reassigning 4 genes from module 20 to modules with higher KME.
##    ..reassigning 3 genes from module 22 to modules with higher KME.
##  ..merging modules that are too close..
```

```
##       mergeCloseModules: Merging modules whose distance is less than 0.2
##         Calculating new MEs...
```

```
asdgood_colors = labels2colors(net_tmp$colors)
datExpr_asdgood = datExpr_grp
```

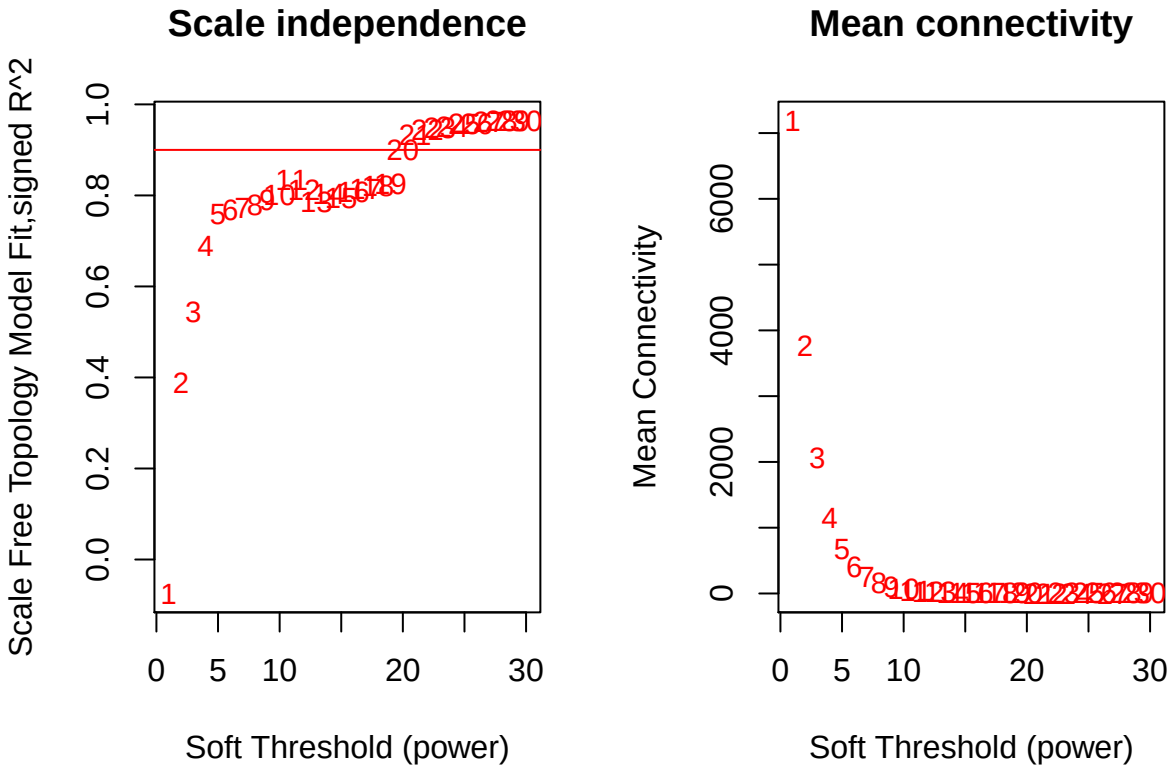
Run WGCNA on ASD Poor only

```
# run on ASD Poor
grp2use = "Poor"
datExpr_grp = datExpr[labelData$subgrp2==grp2use,]
datTraits_grp = datTraits[labelData$subgrp2==grp2use,]

# Choose a soft-threshold power
powers = c(1:30)
if (corrType=="pearson"){
  corFnc2use = "cor"
}else if (corrType=="bicor"){
  corFnc2use = corrType
}
sft = pickSoftThreshold(datExpr_grp,
                        powerVector = powers,
                        verbose = 5,
                        networkType = networkType,
                        corFnc = corrType)
```

```
## pickSoftThreshold: will use block size 3125.
##  pickSoftThreshold: calculating connectivity for given powers...
##    ..working on genes 1 through 3125 of 14313
##    ..working on genes 3126 through 6250 of 14313
##    ..working on genes 6251 through 9375 of 14313
##    ..working on genes 9376 through 12500 of 14313
##    ..working on genes 12501 through 14313 of 14313
##    Power SFT.R.sq  slope truncated.R.sq  mean.k. median.k. max.k.
## 1      1   0.0746   9.00          0.957 7190.000  7.19e+03 7560.0
## 2      2   0.3890 -13.00          0.902 3770.000  3.74e+03 4330.0
## 3      3   0.5440 -10.00          0.900 2050.000  2.02e+03 2650.0
## 4      4   0.6890  -8.20          0.952 1160.000  1.13e+03 1690.0
## 5      5   0.7600  -6.84          0.977  674.000  6.51e+02 1130.0
## 6      6   0.7670  -5.52          0.985  404.000  3.85e+02  776.0
## 7      7   0.7740  -4.63          0.987  249.000  2.34e+02  548.0
## 8      8   0.7800  -4.04          0.986  158.000  1.45e+02  396.0
## 9      9   0.7900  -3.57          0.983  102.000  9.15e+01  292.0
## 10    10   0.8010  -3.19          0.982   67.800  5.88e+01  219.0
## 11    11   0.8350  -2.81          0.988   46.000  3.84e+01  167.0
## 12    12   0.8130  -2.89          0.970   31.800  2.56e+01  137.0
## 13    13   0.7860  -2.96          0.953   22.400  1.72e+01  114.0
## 14    14   0.8050  -2.85          0.964   16.100  1.17e+01   95.7
## 15    15   0.7950  -2.83          0.963   11.800  8.08e+00   81.0
## 16    16   0.8070  -2.73          0.966    8.760  5.64e+00   69.1
## 17    17   0.8150  -2.62          0.963    6.600  3.98e+00   59.3
## 18    18   0.8210  -2.51          0.956    5.050  2.83e+00   51.2
## 19    19   0.8250  -2.39          0.945    3.910  2.03e+00   44.4
```

```
## 20    20   0.9010  -2.16          0.975    3.070  1.46e+00   38.7
## 21    21   0.9340  -2.03          0.986    2.440  1.07e+00   33.9
## 22    22   0.9440  -1.97          0.987    1.960  7.85e-01   30.7
## 23    23   0.9490  -1.95          0.988    1.590  5.82e-01   28.6
## 24    24   0.9510  -1.92          0.987    1.300  4.33e-01   26.7
## 25    25   0.9570  -1.88          0.990    1.070  3.25e-01   24.9
## 26    26   0.9560  -1.86          0.988    0.893  2.45e-01   23.3
## 27    27   0.9630  -1.83          0.991    0.750  1.85e-01   21.8
## 28    28   0.9640  -1.80          0.990    0.634  1.41e-01   20.4
## 29    29   0.9630  -1.77          0.987    0.540  1.08e-01   19.1
## 30    30   0.9640  -1.74          0.986    0.463  8.28e-02   18.0
```

```
makeSoftPowerPlot(sft = sft, powers = powers)
```




```
softPower = 16

## Run an automated network analysis
net_tmp = blockwiseModules(datExpr,
                           power = softPower,
                           deepSplit = deepSplit,
                           minModuleSize = minModSize,
                           mergeCutHeight = modMergeCutHeight,
                           detectCutHeight = 0.9999,
                           corType = corrType,
                           networkType = networkType,
                           pamStage = FALSE,
                           pamRespectsDendro = TRUE,
                           verbose = 3,
                           saveTOMs = FALSE,
                           maxBlockSize = maxBlockSize,
```

```
                            numericLabels = TRUE)
```

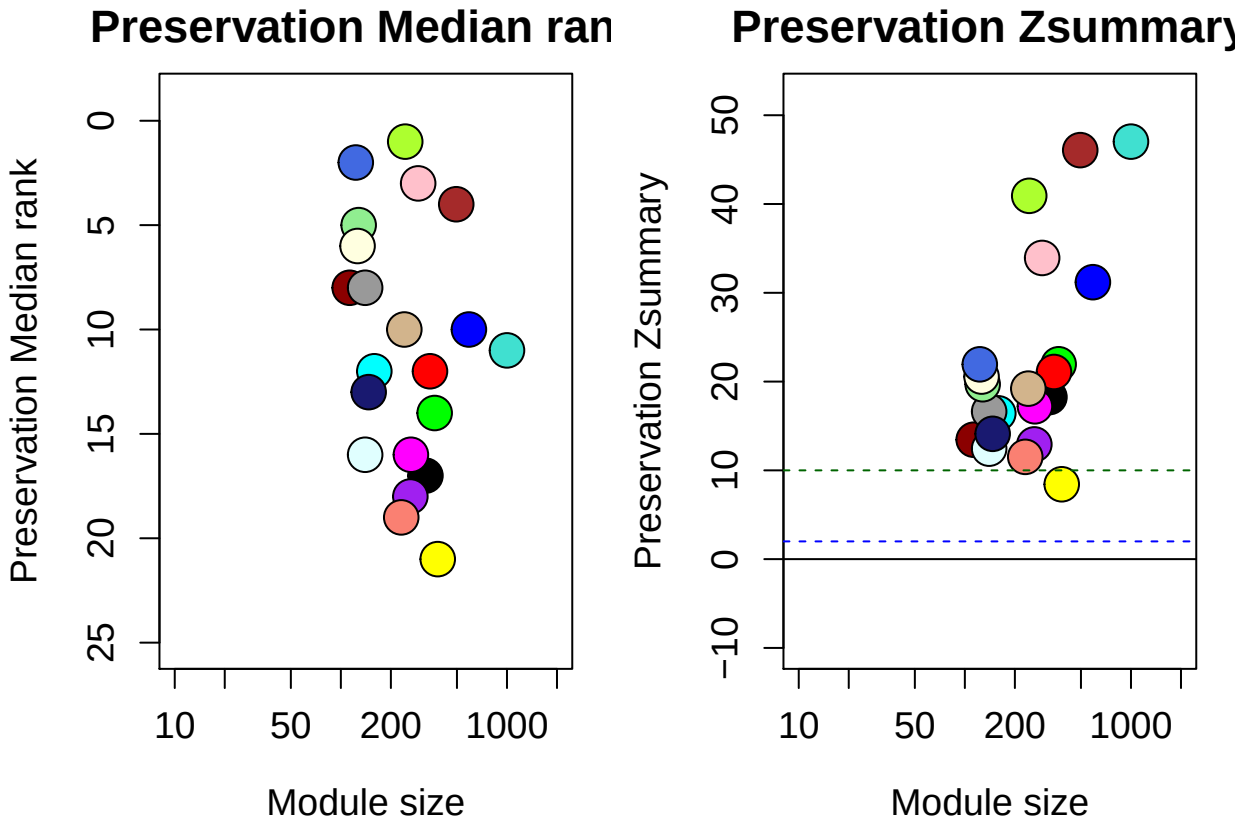
```
##  Calculating module eigengenes block-wise from all genes
##    Flagging genes and samples with too many missing values...
##     ..step 1
##  ..Working on block 1 .
##     TOM calculation: adjacency..
##     ..will use 8 parallel threads.
##      Fraction of slow calculations: 0.000000
##     ..connectivity..
##     ..matrix multiplication (system BLAS)..
##     ..normalization..
##     ..done.
##  ....clustering..
##  ....detecting modules..
##  ....calculating module eigengenes..
##  ....checking kME in modules..
##      ..removing 62 genes from module 1 because their KME is too low.
##      ..removing 226 genes from module 2 because their KME is too low.
##      ..removing 1 genes from module 3 because their KME is too low.
##      ..removing 49 genes from module 4 because their KME is too low.
##      ..removing 7 genes from module 5 because their KME is too low.
##      ..removing 46 genes from module 6 because their KME is too low.
##      ..removing 1 genes from module 7 because their KME is too low.
##      ..removing 27 genes from module 8 because their KME is too low.
##      ..removing 4 genes from module 9 because their KME is too low.
##      ..removing 15 genes from module 11 because their KME is too low.
##      ..removing 27 genes from module 13 because their KME is too low.
##      ..removing 10 genes from module 14 because their KME is too low.
##      ..removing 1 genes from module 18 because their KME is too low.
##   ..reassigning 42 genes from module 1 to modules with higher KME.
##   ..reassigning 2 genes from module 2 to modules with higher KME.
##   ..reassigning 18 genes from module 3 to modules with higher KME.
##   ..reassigning 9 genes from module 4 to modules with higher KME.
##   ..reassigning 13 genes from module 5 to modules with higher KME.
##   ..reassigning 3 genes from module 6 to modules with higher KME.
##   ..reassigning 3 genes from module 7 to modules with higher KME.
##   ..reassigning 9 genes from module 8 to modules with higher KME.
##   ..reassigning 8 genes from module 9 to modules with higher KME.
##   ..reassigning 8 genes from module 10 to modules with higher KME.
##   ..reassigning 7 genes from module 11 to modules with higher KME.
##   ..reassigning 1 genes from module 12 to modules with higher KME.
##   ..reassigning 2 genes from module 14 to modules with higher KME.
##   ..reassigning 2 genes from module 15 to modules with higher KME.
##   ..reassigning 5 genes from module 16 to modules with higher KME.
##   ..reassigning 3 genes from module 17 to modules with higher KME.
##   ..reassigning 1 genes from module 19 to modules with higher KME.
##   ..reassigning 4 genes from module 20 to modules with higher KME.
##   ..reassigning 3 genes from module 22 to modules with higher KME.
##  ..merging modules that are too close..
##      mergeCloseModules: Merging modules whose distance is less than 0.2
##        Calculating new MEs...
```

```
asdpoor_colors = labels2colors(net_tmp$colors)
datExpr_asdpoor = datExpr_grp
```

Function for reporting module preservation results

TD - ASD Good module preservation results

```
nperm = 1000
rand_seed = 1

setLabels = c("TD","ASDGood")
multiExpr = list(TD = list(data = datExpr_td),
                 ASDGood = list(data = datExpr_asdgood))
multiColor = list(TD = td_colors)

mp_td_asdgood = modulePreservation(multiExpr,
                                   multiColor,
                                   networkType = networkType,
                                   corFnc = corFnc2use,
                                   referenceNetworks = 1,
                                   nPermutations = nperm,
                                   randomSeed = rand_seed,
                                   quickCor = 0,
                                   verbose = 0)

mp_res = modulePreservationReport(mp_td_asdgood)
```

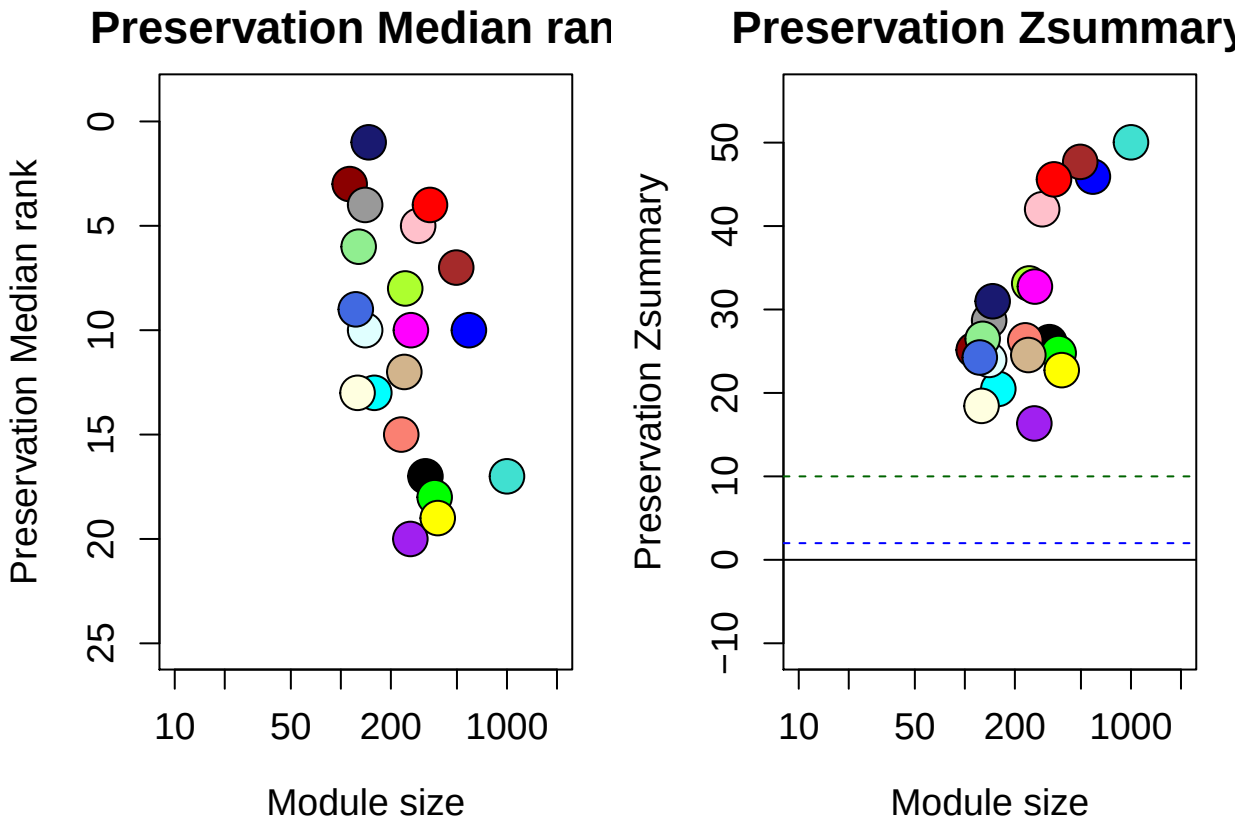



```
mp_res
```

```
##                medianRank.pres medianRank.qual Zsummary.pres Zsummary.qual
## black                       17            18.0           18.0         28.00
## blue                        10            11.0           31.0         68.00
## brown                        4            13.0           46.0         74.00
## cyan                        12            20.5           16.0          7.40
## darkred                      8             7.0           13.0         45.00
## gold                        22            22.0           22.0          0.34
## green                       14            13.5           22.0         41.00
## greenyellow                  1             1.0           41.0         81.00
## grey                        23            23.0           11.0        -15.00
## grey60                       8             6.0           17.0         54.00
## lightcyan                   16            16.0           12.0         23.00
## lightgreen                   5             4.5           20.0         54.00
## lightyellow                  6            12.5           21.0         27.00
## magenta                     16             5.5           17.0         76.00
## midnightblue                13             4.0           14.0         57.00
## pink                         3             8.5           34.0         70.00
## purple                      18            15.0           13.0         25.00
## red                         12            10.5           21.0         64.00
## royalblue                    2             2.5           22.0         55.00
## salmon                      19            20.5           12.0          8.90
## tan                         10             5.5           19.0         74.00
## turquoise                   11            16.5           47.0         56.00
## yellow                      21            19.0            8.4         23.00
```

TD - ASD Poor module preservation results

```
nperm = 1000
rand_seed = 1

setLabels = c("TD","ASDPoor");
multiExpr = list(TD = list(data = datExpr_td),
                 ASDPoor = list(data = datExpr_asdpoor))
multiColor = list(TD = td_colors)

# Calculate module preservation stats
mp_td_asdpoor = modulePreservation(multiExpr,
                                   multiColor,
                                   networkType = networkType,
                                   corFnc = corFnc2use,
                                   referenceNetworks = 1,
                                   nPermutations = nperm,
                                   randomSeed = rand_seed,
                                   quickCor = 0,
                                   verbose = 0)

mp_res = modulePreservationReport(mp_td_asdpoor)
```



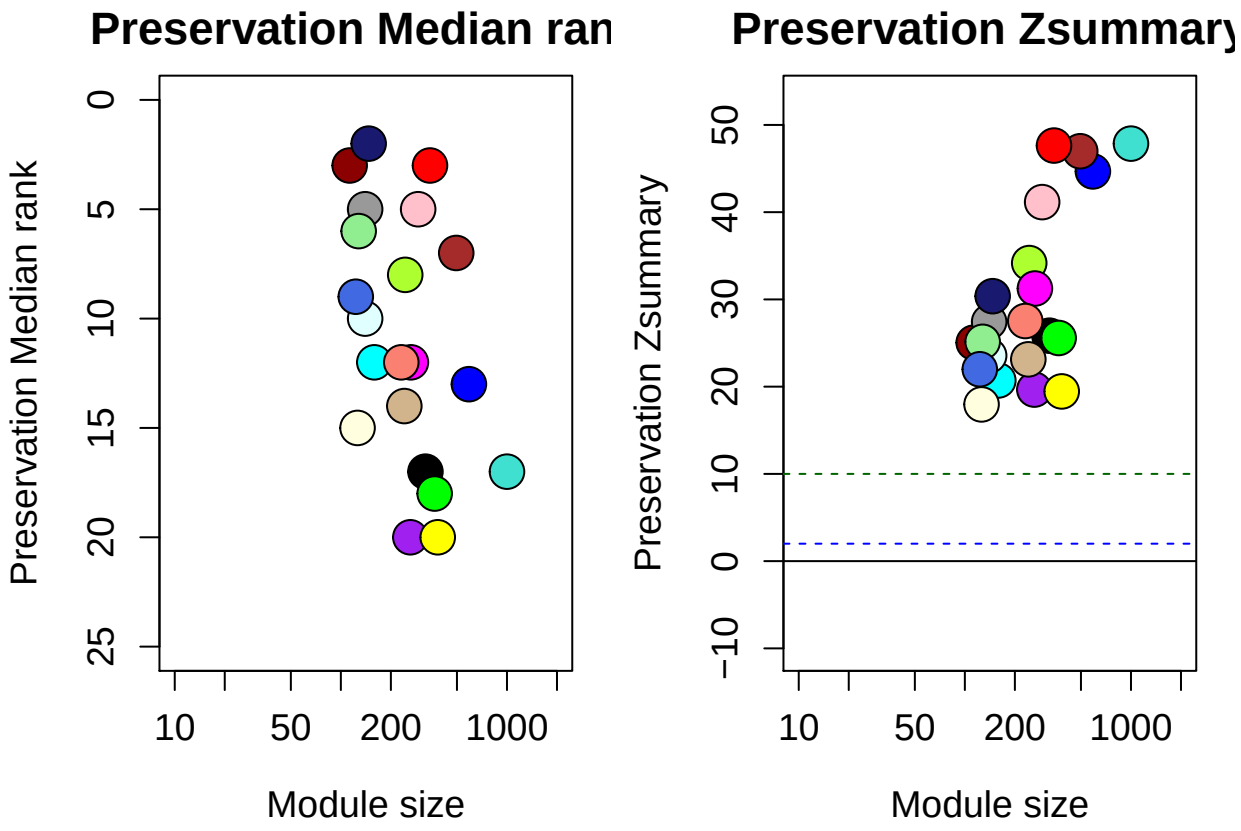

```
mp_res
```

```
##             medianRank.pres medianRank.qual Zsummary.pres Zsummary.qual
## black                    17            18.0          26.0         28.00
## blue                     10            11.0          46.0         68.00
```

```
## brown                  7           13.0          48.0         74.00
## cyan                  13           20.5          20.0          7.40
## darkred                3            7.0          25.0         45.00
## gold                  22           22.0          24.0          0.34
## green                 18           13.5          25.0         41.00
## greenyellow            8            1.0          33.0         81.00
## grey                  23           23.0           9.9        -15.00
## grey60                 4            6.0          29.0         54.00
## lightcyan             10           16.0          24.0         23.00
## lightgreen             6            4.5          26.0         54.00
## lightyellow           13           12.5          18.0         27.00
## magenta               10            5.5          33.0         76.00
## midnightblue           1            4.0          31.0         57.00
## pink                   5            8.5          42.0         70.00
## purple                20           15.0          16.0         25.00
## red                    4           10.5          46.0         64.00
## royalblue              9            2.5          24.0         55.00
## salmon                15           20.5          26.0          8.90
## tan                   12            5.5          25.0         74.00
## turquoise             17           16.5          50.0         56.00
## yellow                19           19.0          23.0         23.00
```

ASD Good - ASD Poor module preservation results

```
nperm = 1000
rand_seed = 1

setLabels = c("ASDGood","ASDPoor")
multiExpr = list(ASDGood = list(data = datExpr_asdgood),
                 ASDPoor = list(data = datExpr_asdpoor))
multiColor = list(ASDGood = asdgood_colors)

# Calculate module preservation stats
mp_asdgood_asdpoor = modulePreservation(multiExpr,
                                        multiColor,
                                        networkType = networkType,
                                        corFnc = corFnc2use,
                                        referenceNetworks = 1,
                                        nPermutations = nperm,
                                        randomSeed = rand_seed,
                                        quickCor = 0,
                                        verbose = 0)

mp_res = modulePreservationReport(mp_asdgood_asdpoor)
```




```
mp_res
```

```
##                medianRank.pres medianRank.qual Zsummary.pres Zsummary.qual
## black                       17            17.5          26.0         23.00
## blue                        13            10.5          45.0         55.00
## brown                        7            11.0          47.0         84.00
## cyan                        12             8.0          21.0         32.00
## darkred                      3             5.5          25.0         27.00
## gold                        22            22.0          26.0         -0.61
## green                       18            14.0          26.0         35.00
## greenyellow                  8             1.0          34.0        110.00
## grey                        23            23.0           7.5        -13.00
## grey60                       5             8.0          27.0         29.00
## lightcyan                   10            16.5          24.0         17.00
## lightgreen                   6             4.0          25.0         43.00
## lightyellow                 15             9.0          18.0         38.00
## magenta                     12            18.5          31.0         24.00
## midnightblue                 2            11.0          30.0         25.00
## pink                         5             3.0          41.0         79.00
## purple                      20            19.0          20.0         18.00
## red                          3            17.0          48.0         30.00
## royalblue                    9             2.0          22.0         56.00
## salmon                      12            15.5          28.0         20.00
## tan                         14             6.5          23.0         40.00
## turquoise                   17            11.0          48.0         78.00
## yellow                      20            21.0          19.0         15.00
```