Tutorial for the WGCNA package for R: III. Using simulated data to evaluate different module detection methods and gene screening approaches

5. Construction of a weighted gene co-expression network and network modules

Steve Horvath and Peter Langfelder

December 7, 2011

Contents

0 Setting up the R session

Before starting, the user should choose a working directory, preferably a directory devoted exclusively for this tutorial. After starting an R session, change working directory, load the requisite packages, set standard options, and load the results of previous sections:

```
# Display the current working directory
getwd();
# If necessary, change the path below to the directory where the data files are stored.
# "." means current directory. On Windows use a forward slash / instead of the usual \.
workingDir = ".";
setwd(workingDir);
# Load WGCNA package
library(WGCNA)
# Load additional necessary packages
library(cluster)
```

```
# The following setting is important, do not omit.
options(stringsAsFactors = FALSE);
# Load the previously saved data
load("Simulated-StandardScreening.RData");
attach(ModuleEigengeneNetwork1)
```

5 Construction of a weighted gene co-expression network and network modules

In this section we provide a step-by-step overview of gene network construction and module detection.

5.a Defining a weighted gene co-expression network

We illustrate network construction and evaluation of scale free topology. Recall that the genes correspond to the columns of `datExpr`.

```
# here we define the adjacency matrix using soft thresholding with beta=6
ADJ1=abs(cor(datExpr,use="p"))^6
# 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=6)
# 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")
```

The resulting plot is shown in Fig. 1. The approximate straight line relationship (high R^2 value, right panel in Fig. 1) shows approximate scale free topology. In most applications we find that scale free topology is at least approximately satisfied when a high power is chosen for defining the adjacency matrix. We should point out that is not necessary that a network satisfies scale free topology; scale free topology may not be satisfied if the data are comprised of globally very distinct groups of samples (e.g. different tissues types). Poor fit to scale free topology may also indicate the presence of array outliers.

5.b Computational restrictions on the number of genes

For computational reasons, we restrict the network analysis to 3600 most connected genes.

```
datExpr=datExpr[, rank(-k,ties.method="first" )<=3600]
```

This restriction is only necessary when constructing the network in a step-by-step, "manual" way. The user can also use the automatic one-step function `blockwiseModules` that is able to handle large data sets. We refer the reader to Tutorials I, II (mouse data analysis) for examples of using the automatic function on large data sets.

5.c Comparing various module detection methods

5.c.1 Definition of clustering dissimilarity from adjacency

Many clustering procedures require a dissimilarity matrix as input. We define a dissimilarity based on adjacency:

```
# Turn adjacency into a measure of dissimilarity
dissADJ=1-ADJ1
```

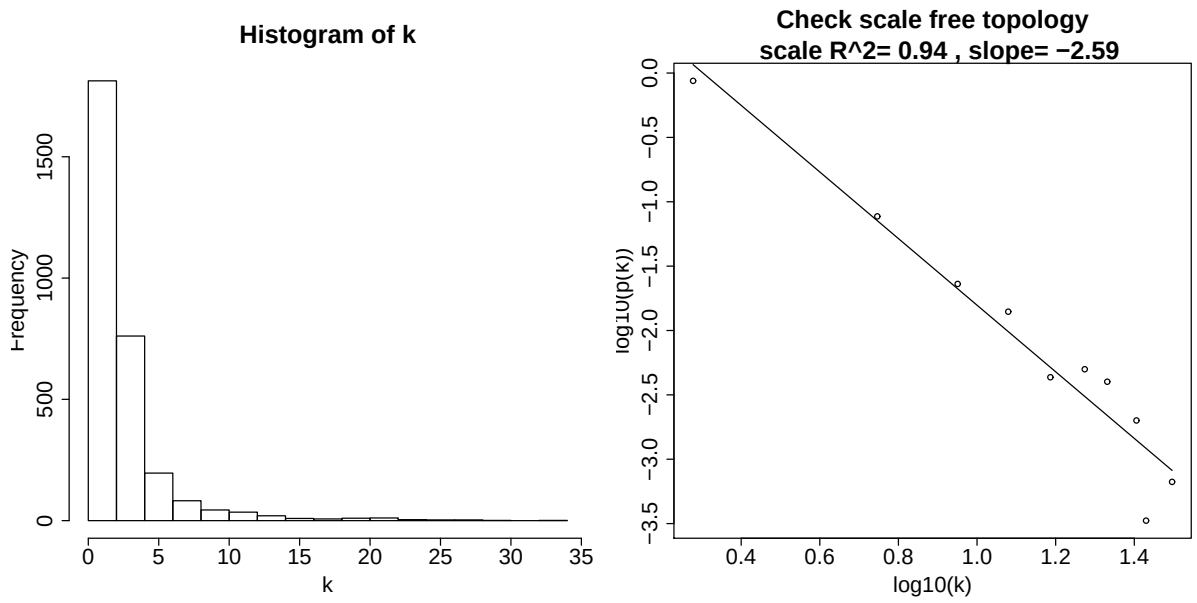

Figure 1: The left panel shows a histogram of network connectivities. The right panel shows a log-log plot of the same histogram. The approximate straight line relationship (high R^2 value) shows approximate scale free topology. In most applications we find that scale free topology is at least approximately satisfied when a high power is chosen for defining the adjacency matrix. We should point out that is not necessary that a network satisfies scale free topology; scale free topology may not be satisfied if the data are comprised of globally very distinct groups of samples (e.g. different tissues types). Poor fit to scale free topology may also indicate the presence of array outliers.

5.c.2 Use of topologial overlap to define dissimilarity

Adjacency can be used to define a separate measure of similarity, the Topological Overlap Matrix(TOM) [2, 1]:

```
dissTOM=TOMdist(ADJ1)
collectGarbage()
```

In the following subsections we show that Partitioning Around Medoids (PAM) is not a good choice for clustering genes because it forces all genes into a module. We then illustrate the use of hierarchical clustering and several branch cut methods.

5.c.3 Partitioning Around Medoids based on adjacency

We use PAM with 3 different numbers of clusters:

```
pam4=pam(as.dist(dissADJ), 4)
pam5=pam(as.dist(dissADJ), 5)
pam6=pam(as.dist(dissADJ), 6)
# Cross-tabulte the detected and the true (simulated) module membership:
table(pam4$clustering, truemodule)
table(pam5$clustering, truemodule)
table(pam6$clustering, truemodule)
```

The results are

```
> table(pam4$clustering, truemodule)
   truemodule
    blue brown green grey turquoise yellow
  1   51     8    17  792       530      1
```

```
  2  380    10    34  261        37      6
  3   12     4    17  167        17    172
  4    7   218    52  190        16      1
> table(pam5$clustering, truemodule)
   truemodule
    blue brown green grey turquoise yellow
  1   50     5     2  742       524      1
  2  373    10     3  234        35      6
  3   11     3     2  142        15    170
  4    9     7   107  133        16      2
  5    7   215     6  159        10      1
> table(pam6$clustering, truemodule)
   truemodule
    blue brown green grey turquoise yellow
  1   39     3     1  440       310      1
  2   21     5     3  412       235      1
  3  363    10     3  186        26      6
  4   11     3     2  119         8    169
  5    9     6   106  119        14      2
  6    7   213     5  134         7      1
```

The rows correspond to clusters found by PAM, while the columns correspond to simulated modules. Numbers in the table are gene counts in the intersection of the respective simulated and found modules. PAM performs terribly on this data set since the grey, non-module genes are "forced" into the observed modules (corresponding to the rows). In general, partitioning methods that dont allow for unclustered objects will have a problem on this simulated data set.

5.c.4 Partitioning Around Medoids based on topological overlap

We again use PAM with the same three different numbers of clusters:

```
pamTOM4=pam(as.dist(dissTOM), 4)
pamTOM5=pam(as.dist(dissTOM), 5)
pamTOM6=pam(as.dist(dissTOM), 6)
# Cross-tabulte the detected and the true (simulated) module membership:
table(pamTOM4$clustering, truemodule)
table(pamTOM5$clustering, truemodule)
table(pamTOM6$clustering, truemodule)
```

The results of the cross-tabulation are

```
> table(pamTOM4$clustering, truemodule)
   truemodule
    blue brown green grey turquoise yellow
  1   58    15    34 1100       576     10
  2  384    11    48  197        21      9
  3    3     2     4   42         2    159
  4    5   212    34   71         1      2
> table(pamTOM5$clustering, truemodule)
   truemodule
    blue brown green grey turquoise yellow
  1   58    13     9 1087       574     10
  2  384    10     7  190        20      9
  3    0     4    97   28         3      0
  4    3     2     0   40         2    159
  5    5   211     7   65         1      2
```

```
> table(pamTOM6$clustering, truemodule)
   truemodule
    blue brown green grey turquoise yellow
  1   49    11     7  808       475      8
  2    9     2     2  285        99      2
  3  384    10     7  185        20      9
  4    0     4    97   28         3      0
  5    3     2     0   40         2    159
  6    5   211     7   64         1      2
```

As above, the rows correspond to clusters found by PAM, while the columns correspond to simulated modules. Numbers in the table are gene counts in the intersection of the respective simulated and found modules. The performance of pam is not good. This is why we will use hierarchical clustering.

5.c.5 Average linkage hierachical clustering with adjacency-based dissimilarity

Here we illustrate the use of average linkage hierachical clustering.

```
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 Fig. 2. Note that the branches correspond to the true modules. The question now is how should we cut the branches of the dendrogram to determine module membership?

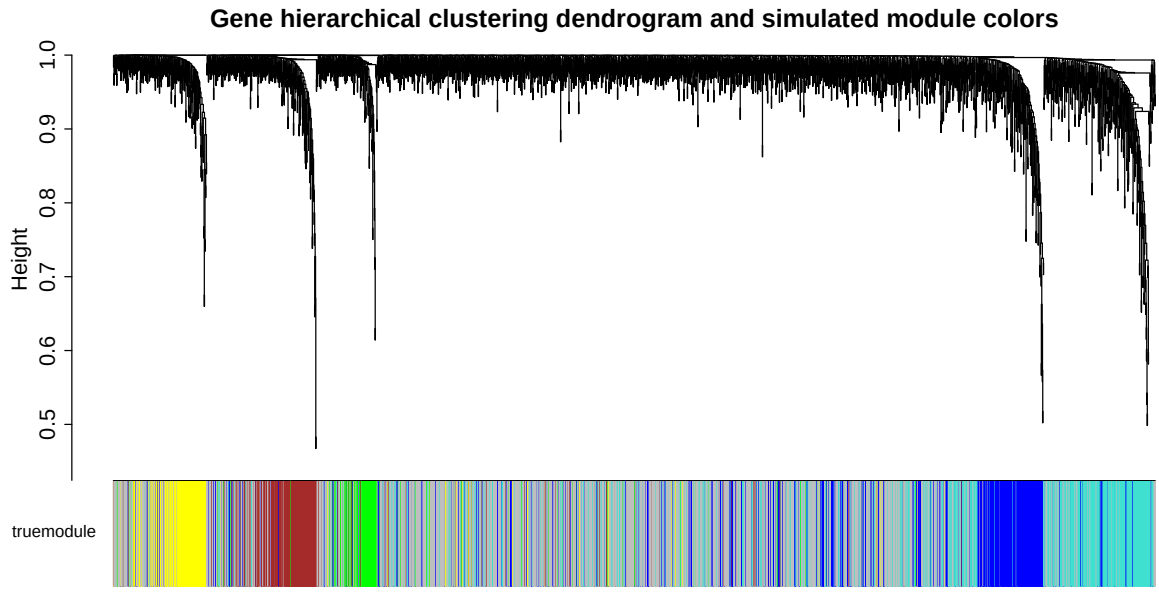

Figure 2: Gene clustering tree (dendrogram) obtained by hierarchical clustering of adjacency-based dissimilarity. The color row below the dendrogram indicates module membership.

5.c.6 Module definition via static (fixed) height cut-off

By our definition, modules correspond to branches of the tree. The question is what height cut-off should be used? This depends on the biology. Large heigth values lead to big modules, small values lead to small but tight modules.

In reality, the user should use different thresholds `h1` (red below) to assess how robust the findings are. The function `cutreeStaticColor` colors each gene by the branches that result from choosing a particular 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` genes.

```
colorStaticADJ=as.character(cutreeStaticColor(hierADJ, cutHeight=.99, minSize=20))
# Plot the dendrogram with module colors
sizeGrWindow(10,5);
plotDendroAndColors(hierADJ, colors = data.frame(truemodule, colorStaticADJ),
                    dendroLabels = FALSE, abHeight = 0.99,
                    main = "Gene dendrogram and module colors")
```

The resulting plot is shown in Fig. 3. The static height cut-off method works quite well at retrieving the true modules. More precisely, it works well at retrieving highly connected intramodular hub genes in the modules. However, it misses a lot of genes at the fringes of the modules.

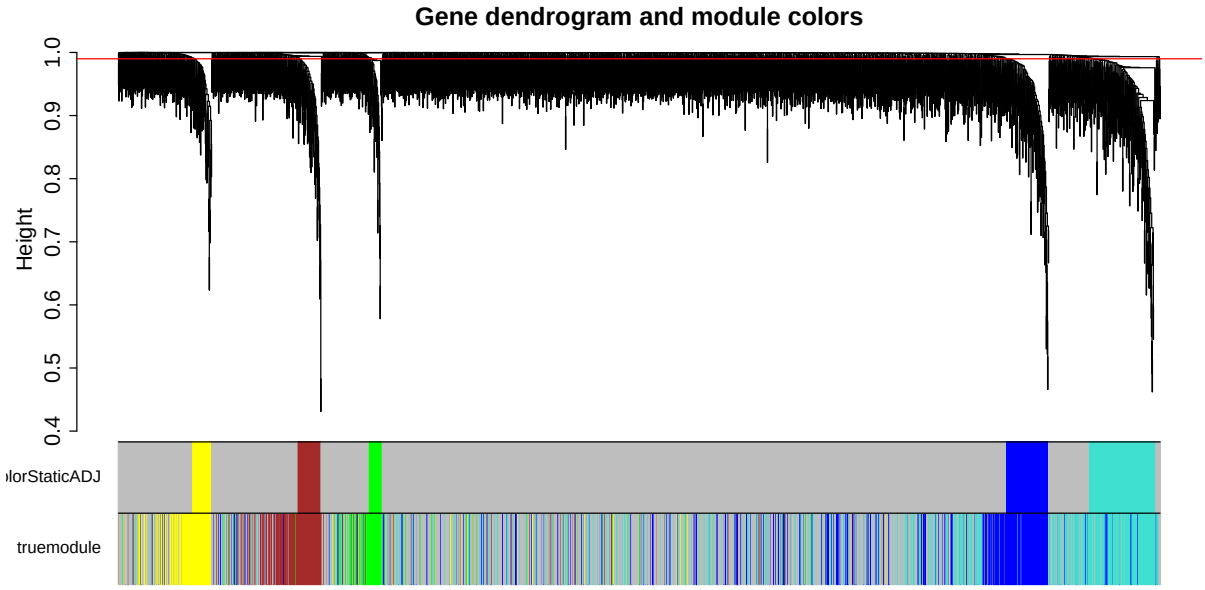

Figure 3: Gene clustering tree (dendrogram) obtained by hierarchical clustering of adjacency-based dissimilarity. The color rows below the dendrogram indicate identified and simulated module membership. The static height cut-off method works quite well at retrieving the true modules. More precisely, it works well at retrieving highly connected intramodular hub genes in the modules. However, it misses a lot of genes at the fringes of the modules.

5.c.7 Module definition via dynamic branch cutting methods

We now use two Dynamic Tree Cut methods in which the height cut-off plays a minor role. The first method is called the "tree" method and only uses the dendrogram as input.

```
branch.number=cutreeDynamic(hierADJ,method="tree")
# This function transforms the branch numbers into colors
colorDynamicADJ=labels2colors(branch.number )
```

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.

```
colorDynamicHybridADJ=labels2colors(cutreeDynamic(hierADJ,distM= dissADJ,
                            cutHeight = 0.998, 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),
                    main = "Gene dendrogram and module colors")
```

The plot is shown in Fig. 4. The static method (3rd band) has high specificity but low sensitivity, i.e. its module membership assignment is very accurate but it misses a lot of genes (too many grey genes). In contrast, the dynamic hybrid method (first band) has high sensitivity but low specificity. Actually, we simulated the grey genes so that they would have a weak "background" correlation with the turquoise module genes. Therefore, it comes as no surprise that the hybrid method assigns turquoise to many grey module genes. The "tree" method (second color band) does not perform very well since it splits some of the branches into two sub-branches. As we demonstrate below, one should always look at the tree and the correlations between the module eigengenes to determine whether two modules should be merged.

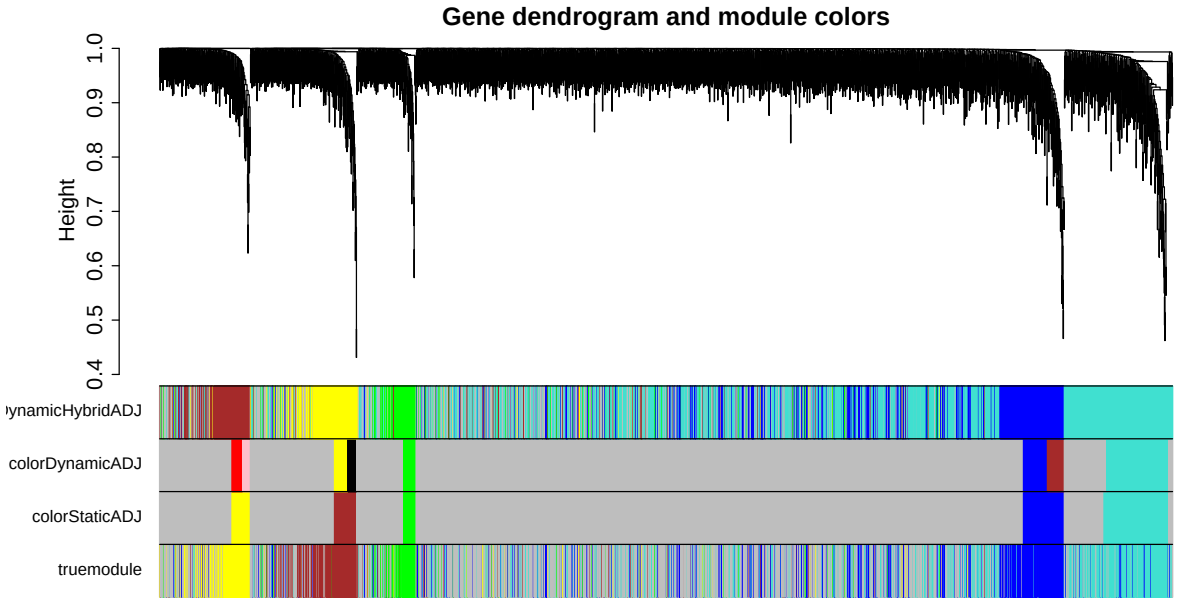

Figure 4: Gene clustering tree (dendrogram) obtained by hierarchical clustering of adjacency-based dissimilarity. The color rows below the dendrogram indicate module membership identified by the three methods discussed in text and the simulated module membership. The static method (3rd band) has high specificity but low sensitivity, i.e. its module membership assignment is very accurate but it misses a lot of genes (too many grey genes). In contrast, the dynamic hybrid method (first band) has high sensitivity but low specificity. Actually, we simulated the grey genes so that they would have a weak "background" correlation with the turquoise module genes. Therefore, it comes as no surprise that the hybrid method assigns turquoise to many grey module genes. The "tree" method (second color band) does not perform very well since it splits some of the branches into two sub-branches. As we demonstrate below, one should always look at the tree and the correlations between the module eigengenes to determine whether two modules should be merged.

5.c.8 Module definition using the topological overlap based dissimilarity

We now use the topological overlap based dissimilarity as input to the clustering methods we used above.

```
# 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=.99, minSize=20))
colorDynamicTOM = labels2colors (cutreeDynamic(hierTOM,method="tree"))
colorDynamicHybridTOM = labels2colors(cutreeDynamic(hierTOM, distM= dissTOM , cutHeight = 0.998,
                    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")
```

The result is shown in Fig. 5. The dynamic hybrid method leads to modules that are slightly too large. In practice this would not be a problem since one typically focuses on the tip of the branches (highly connected hub genes). Also it would have a very small effect on the definition of the module eigengenes.

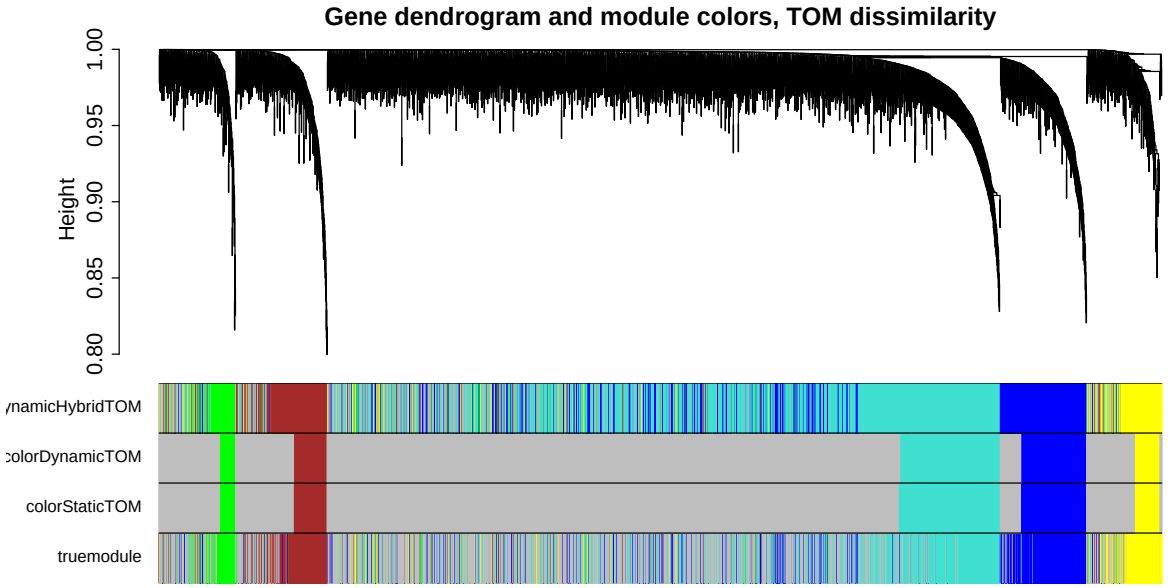

Figure 5: Gene clustering tree (dendrogram) obtained by hierarchical clustering of TOM-based dissimilarity. The color rows below the dendrogram indicate module membership identified by the three methods discussed in text and the simulated module membership. The dynamic hybrid method leads to modules that are slightly too large. In practice this would not be a problem since one typically focuses on the tip of the branches (highly connected hub genes). Also it would have a very small effect on the definition of the module eigengenes.

5.c.9 Which dissimilarity measure and which branch cutting method performs best in this data set?

The answer depends on threshold parameters used for branch cutting. For illustration, we carry out a brute force comparison. In the following, we create tables for relating the different module assignements to the true module colors.

```
tabStaticADJ=table(colorStaticADJ,truemodule)
tabStaticTOM=table(colorStaticTOM,truemodule)
tabDynamicADJ=table(colorDynamicADJ, truemodule)
tabDynamicTOM=table(colorDynamicTOM,truemodule)
```

```
tabDynamicHybridADJ =table(colorDynamicHybridADJ,truemodule)
tabDynamicHybridTOM =table(colorDynamicHybridTOM,truemodule)
```

Next, we use the (unadjusted) Rand index to measure agreement. This computes the Rand index for each table:

```
randIndex(tabStaticADJ,adjust=F)
randIndex(tabStaticTOM,adjust=F)
randIndex(tabDynamicADJ,adjust=F)
randIndex(tabDynamicTOM,adjust=F)
randIndex(tabDynamicHybridADJ ,adjust=F)
randIndex(tabDynamicHybridTOM ,adjust=F)
```

The returned Rand indices are as follows:

```
> randIndex(tabStaticADJ,adjust=F)
[1] 0.5072126
> randIndex(tabStaticTOM,adjust=F)
[1] 0.5997922
> randIndex(tabDynamicADJ,adjust=F)
[1] 0.5033845
> randIndex(tabDynamicTOM,adjust=F)
[1] 0.5980453
> randIndex(tabDynamicHybridADJ ,adjust=F)
[1] 0.7054071
> randIndex(tabDynamicHybridTOM ,adjust=F)
[1] 0.7039642
```

In this data set, dissTOM performs better than dissADJ for the first two branch cutting method. The results for the Dynamic Hybrid methods are very similar. In the following, we will proceed with the observed modules from DynamicHybridTOM. Here is the cross tabulation of colorDynamicHybridTOM with the true module assignment:

```
> tabDynamicHybridTOM
                      truemodule
colorDynamicHybridTOM blue brown green grey turquoise yellow
             blue      377     8     4  163        30      7
             brown       5   211     5   68         3      1
             green       0     4    95   35         4      0
             grey       26    11    11  326        28      6
             turquoise  35     4     4  754       530      3
             yellow      7     2     1   64         5    163
```

We define a shorter name for the module assignemnt of choice, remove unneeded variables and save the results for use in subsequent sessions:

```
colorh1= colorDynamicHybridTOM
# remove the dissimilarities, adjacency matrices etc to free up space
rm(ADJ1); rm(dissADJ);
collectGarbage()
save.image("Simulated-NetworkConstruction.RData")
```

References

[1] Andy Yip and Steve Horvath. Gene network interconnectedness and the generalized topological overlap measure. *BMC Bioinformatics*, 8(1):22, 2007.

[2] B. Zhang and S. Horvath. A general framework for weighted gene co-expression network analysis. *Statistical Applications in Genetics and Molecular Biology*, 4(1):Article 17, 2005.