An Algorithm for Identification of Preserved Networks in Multiple Datasets

The `cmdnet` Vignette

Nicholas Wisniewski
University of California, Los Angeles

October 17, 2016

Abstract

As more scientific data is shared, interest is moving beyond single experiments toward the integration of multiple public datasets. Network analysis tools are important in analysis of high-dimensional data such as genomics, but difficulties arise when faced with multiple sources of data due to large variability across datasets. Here we present an algorithm, `cmdnet`, that attempts to understand this variability by computing correlation matrix distance (CMD) statistics for an ensemble of networks. It can be used to quantify overall and node-specific differences between network datasets, and rank nodes by consistency of edge weights. After deriving the main statistics, we show an example usage, and demonstrate how to identify preserved subnetworks and reduce noise by removing highly variable nodes. Finally, we use `cmdnet` in conjunction with WGCNA to show that preserved subnetworks are unbiased with respect to modular structure, being evenly distributed across the WGNCA network. We also show that a dataset can be filtered significantly while still recovering the modular structure found by full-network reconstruction.

Contents

1 Introduction

The `cmdnet` algorithm computes correlation matrix distances (CMD) for an ensemble of networks, which can be used to quantify variability across networks, and as a noise reduction tool to reduce variability. This application is based on networks that can be represented by correlation matrices, though alternative measures of undirected adjacency (e.g. MIC, mutual information) are also compatible (using the `corFUN` option). The algorithm works by first computing an average or consensus network, and then quantifying variability by measuring deviations from the consensus.

The distance measures used by `cmdnet` to quantify deviations are based on the angular distance between vectors, or what is often referred to as cosine similarity. This measure arises from the familiar Euclidean dot product formula, relating the inner product of two vectors a and b to the cosine of the subtended angle,

$$\langle a, b \rangle = \|a\| \, \|b\| \, \cos\theta.$$

Consequently, $\cos\theta$ is generally used to measure similarity between two vectors, since $\cos\theta \to 1$ when vectors are identical, $\cos\theta \to 0$ when vectors are orthogonal, and $\cos\theta \to -1$ when vectors are anti-parallel. This relationship provides a computationally efficient way to compute similarity in very large datasets, such as when inferring a network from data. Indeed, the Pearson correlation coefficient can be understood as an application of this inner product.

But it is also necessary sometimes to work with a measure of distance rather than similarity. For these purposes, it is common to simply flip the unit interval so that similar vectors tend to zero, while orthogonal vectors tend to one,

$$d = 1 - \cos\theta.$$

However, this measure is not a distance measure in the strictest sense, as it does not satisfy the triangle equality. To recover a true distance measure, we can instead solve for the angle θ directly,

$$\theta = \cos^{-1}\left(\frac{\langle a, b\rangle}{\|a\|\,\|b\|}\right),$$

and then normalize it to get a value between zero and one. The normalization chosen depends on whether the inner product is strictly positive,

$$d_\theta = \frac{2}{\pi}\cos^{-1}\left(\frac{\langle a, b\rangle}{\|a\|\,\|b\|}\right), \qquad 1 \geq \langle a, b\rangle \geq 0$$

$$d_\theta = \frac{1}{\pi}\cos^{-1}\left(\frac{\langle a, b\rangle}{\|a\|\,\|b\|}\right), \qquad 1 \geq \langle a, b\rangle \geq -1$$

Below, we generalize this distance measure to sets of correlation matrices to provide fast measures of network differences across datasets.

1.1 Node-Specific Deviations

A node in a graph can be described by a vector of edge weights encoding its connections. When a network is represented by a correlation matrix ρ, the vector of edge weights for node j simply corresponds to the j-th row of the correlation matrix, $\rho_{j\cdot}$. Similarity between a node's connectivity in two networks $\rho_{1,j\cdot}$ and $\rho_{2,j\cdot}$ can then be summarized by the cosine similarity explained above. This leads directly to an angular distance measure,

$$d_\theta(\rho_{1,j\cdot}, \rho_{2,j\cdot}) = \frac{1}{\pi}\cos^{-1}\left(\frac{\langle \rho_{1,j\cdot}, \rho_{2,j\cdot}\rangle}{\|\rho_{1,j\cdot}\|\,\|\rho_{2,j\cdot}\|}\right).$$

With this as a distance metric between two networks, we can proceed to derive a measure of variability over several networks. Using the set of correlation matrices, we compute an average network $\bar{\rho}$, and then measure variability by computing the deviations $\delta_{i,j}$ of each node j in network i away from the average,

$$\delta_{i,j} = d_\theta(\rho_{i,j\cdot}, \bar{\rho}_{j\cdot}); \qquad \bar{\rho} = \sum_{i=1}^{m} w_i\,\rho_i; \qquad |w| = 1.$$

We then calculate the mean deviation $\bar{\delta}_j$ for each node, providing a scalar measure of variance in node connectivity across m datasets,

$$\bar{\delta}_j = \sum_{i=1}^{k} w_i\,\delta_{i,j}.$$

It follows that subsets of nodes with low mean deviation describe subnetworks that are well preserved across datasets, suggesting a fast technique for noise reduction in network analysis. Next, we create a metric to measure the reduction in noise, by quantifying overall network similarity.

1.2 Correlation Matrix Deviations

The notion of cosine similarity can be generalized to matrices in order to measure the distance between two networks. This generalization is known as the correlation matrix distance (CMD), which we write here in terms of the angular distance,

$$D_\theta(\rho_1, \rho_2) = \frac{2}{\pi} \cos^{-1}\left(\frac{\mathrm{tr}\{\rho_1 \rho_2\}}{\|\rho_1\| \, \|\rho_2\|} \right).$$

This distance is 0 if two matrices are equal up to a scaling factor, and 1 when they are maximally different. It is an attractive metric because because it can be vectorized, viewing orthogonality between $n \times n$ matrices as orthogonality in an n^2 dimensional vector space.

With this as a measure of distance between networks, we derive a measure of variability over a set of networks, analogous to the node-specific measure. With the average network $\bar{\rho}$ computed earlier, we measure variability using the deviations Δ_i from the average for each network i,,

$$\Delta_i = D_\theta(\rho_i, \bar{\rho}); \quad \bar{\rho} = \sum_{i=1}^{m} w_i \, \rho_i; \quad |w| = 1.$$

We then calculate the mean deviation $\bar{\Delta}$, providing a scalar measure of variance in network connectivity across m datasets

$$\bar{\Delta} = \sum_{i=1}^{m} w_i \, \Delta_i.$$

Successful noise reduction schemes should minimize this quantity. In our results, we show how selecting subnetworks based on the node-specific deviations can be used to filter out network noise.

1.2.1 Option: Cosine Dissimilarity

Here we note that the alternative distance measures based on cosine similarity, which do not calculate the angle directly,

$$d(\rho_{1,j\cdot}, \rho_{2,j\cdot}) = 1 - \frac{\langle \rho_{1,j\cdot}, \rho_{2,j\cdot} \rangle}{\|\rho_{1,j\cdot}\| \, \|\rho_{2,j\cdot}\|},$$

$$D(\rho_1, \rho_2) = 1 - \frac{\mathrm{tr}\{\rho_1 \rho_2\}}{\|\rho_1\| \, \|\rho_2\|},$$

are also available in `cmdnet` and can be accessed by setting the option `metric = "cosine"`.

2 Data Preparation

In this article, we show how to use `cmdnet`, discuss the different modes and options, and demonstrate how to perform noise reduction. But first, we must load some example data to illustrate the required format of the inputs. There is only one necessary input, and that is a list of data arrays in usual format (dataframe or matrix), where each row is a node, and each column is an independent measurement (e.g. expression level). In our example, this list is named `datalist`. There is a second optional input, which is a vector of nodes-of-interest. In our example, this vector is named `geneset`, and we use it to limit our analysis to only 268 genes. It’s worth noting that this set of genes was previously compiled by using a network analysis approach outside `cmdnet` to identify modules.

2.1 Load `cmdnet`

To load the functions, just source the main file.

```r
library(devtools)
install_github("nwisn/cmdnet")
```

```
## Downloading GitHub repo nwisn/cmdnet@master
## Installing cmdnet
## '/Library/Frameworks/R.framework/Resources/bin/R' --no-site-file  \
##   --no-environ --no-save --no-restore CMD INSTALL  \
##   '/private/var/folders/6m/695jqp0n6bg3zr6qhp_hw49w0000gp/T/RtmpvDeeDp/devtools6cce23b3dd2a/nwis
##   --library='/Users/drwho/Documents/cmdnet/packrat/lib/x86_64-apple-darwin14.5.0/3.2.3'  \
##   --install-tests
```

```r
library(cmdnet)
```

```
## Loading required package: psych
## Loading required package: matrixcalc
```

2.2 Import raw data

Here we load an ensemble of genomic datasets `datalist`, and `geneset` containing EnsemblID’s of genes-of-interest. The datasets have different numbers of genes, and the ordering of the genes in the data arrays are different across datasets. To handle this automatically, `cmdnet` calls a function named `cleandDatalist` that performs the matching. We show the dimensionality of each dataset, where number of columns corresponds to the sample size. Most of the datasets are around 20 samples, expect for one with n=5. The `geneset` is just a vector of gene names used to match with the rownames of the datasets, and can be either a factor from some dataframe, or characters.

```r
load("datalist.RData")
class(datalist)
```

```
## [1] "list"
```

```r
names(datalist)
```

```
## [1] "GSE52202_eset_rlt"     "GSE51684_eset_rlt"     "LudwigHuman_eset_rlt"
## [4] "TaylorStJude_eset_rlt" "GSE76220_eset_rlt"     "GSE54409_eset_rlt"
```

```r
lapply(datalist, dim)
```

```
## $GSE52202_eset_rlt
## [1] 15013    16
##
## $GSE51684_eset_rlt
## [1] 16911    24
##
## $LudwigHuman_eset_rlt
## [1] 13429    25
##
## $TaylorStJude_eset_rlt
## [1] 15655    18
##
## $GSE76220_eset_rlt
## [1] 15404    20
##
## $GSE54409_eset_rlt
## [1] 14087     5
```

```r
class(geneset)
```

```
## [1] "factor"
```

```r
head(geneset)
```

```
## [1] ENSG00000005810 ENSG00000013293 ENSG00000014641 ENSG00000014824
## [5] ENSG00000017260 ENSG00000023287
## 350 Levels: ENSG00000004766 ENSG00000005810 ... ENSG00000282984
```

3 Run `cmdnet`

The `cmdnet` function runs in three general modes, which can be computed together or separately, and specified by the option `mode = c("cmd","row","net")`. The `"cmd"` mode calculates the mean correlation matrix deviation $\bar{\Delta}$ by computing the distance Δ_i between each network i and the averaged network $\bar{\rho}$, and averaging according to sample size. The `"node"` mode calculates node-specific correlation deviations $\delta_{i,j}$, giving rise to a metric $\bar{\delta}_j$ describing the variability of each node j's connectivity across datasets. The `"net"` mode computes the correlation matrix distance $D(\rho_i, \rho_j)$ between each pair of networks, enabling clustering of networks in the ensemble, and identification of outlier networks.

To run `cmdnet`, simply pass in the `datalist` and `geneset`. By default, all three modes of `cmdnet` are computed, but any subset can be specified in the function call using the `mode` option. Other options will be detailed later.

```
result <- cmdnet(datalist, geneset, mode=c("cmd","node","net"))
class(result)
```

```
## [1] "list"
```

```
names(result)
```

```
##  [1] "cmd.dev"        "cmd.meandev"    "node.dev"       "node.meandev"
##  [5] "node.dev.cor1" "node.dev.cor2" "net"            "group.names"
##  [9] "group.r"        "group.n"        "total.n"        "r0"
```

The output of `cmdnet` is a list of 12 statistics. If all modes are selected, then all the statistics will be computed, and every element of the list will be filled. However, if only a subset of modes were selected, then some elements will be null. In addition to the statistics resulting from each mode, the result also contains statistics about each group, such as the names of each dataset or group `group.names`, the correlation matrices `group.r` of each dataset, the sample size of each dataset `group.n`, the total sample size `total.n`, and the mean correlation matrix `r0`.

3.1 Mode "cmd"

The `"cmd"` mode concerns itself with computing correlation matrix deviations around the mean network. It returns two statistics, Δ_i=`cmd.dev` and $\bar{\Delta}$=`cmd.meandev`, containing the distance from each network to the mean, and the mean correlation matrix deviation. We give more details below.

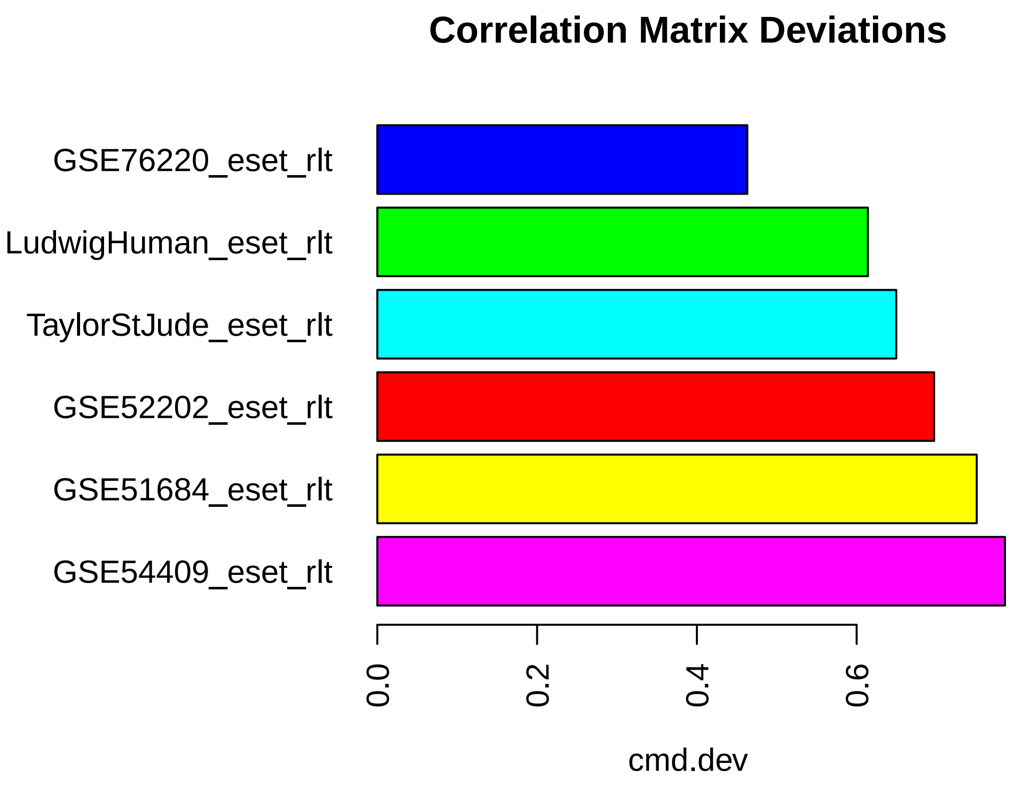
3.1.1 Correlation Matrix Deviations `cmd.dev`

The deviations Δ_i are found by computing the correlation matrix distances between networks ρ_i =`group.r` and the mean network $\bar{\rho}$ =`r0`. There is one deviation statistic for each dataset or group. Outlier datasets can be identified by looking for extreme deviations. We notice that the blue dataset is closest to the consensus network, and the magenta dataset is farthest. It’s worth noting that the magenta dataset has extremely small sample size (n=5), and is therefore highly suspect. Later when we compute node-specific, we can see if there are only certain genes that are problematic, or if the whole dataset should be dropped.

```
result$cmd.dev
```

```
##     GSE52202_eset_rlt     GSE51684_eset_rlt  LudwigHuman_eset_rlt
##             0.6969836             0.7503387             0.6140464
## TaylorStJude_eset_rlt     GSE76220_eset_rlt     GSE54409_eset_rlt
##             0.6495966             0.4631677             0.7856427
```

```r
par(mar=c(5,14,4,2))
bars <- sort(result$cmd.dev, decreasing=T, index.return=T)
barplot(bars$x, horiz=T, las=2, xlab="cmd.dev", space=.2 ,
        col= rainbow(length(names(datalist)))[bars$ix],
        main="Correlation Matrix Deviations")
```



3.1.1.1 Option: Variance Stabilization `xform = "fisher"` The mean network $\bar{\rho}$ =`r0` is computed by default using a linear weighted averaging of the correlation matrices `group.r`. By setting the option `xform = "fisher"`, the averaging of each correlation coefficient can be done in the variance stabilized space provided by Fisher's transformation $z = \text{arctanh}(r)$ to improve accuracy. However, the improved accuracy comes at a time cost, which is inconsequential for small networks, but significant for networks with thousands of nodes.

```r
system.time(result.fisher <- cmdnet(datalist, geneset, mode=c("cmd"), xform="fisher"))
```

```
##    user  system elapsed
##   0.236   0.004   0.241
```

```r
system.time(result <- cmdnet(datalist, geneset, mode=c("cmd")))
```

```
##    user  system elapsed
##   0.021   0.001   0.022
```

3.1.1.2 Option: User-Defined Consensus Network

It is also possible to give `cmdnet` a pre-computed matrix (such as a consensus network) to act as the mean correlation matrix, by passing in the variable `r0`, which is by default set to `r0=NULL`. Currently this matrix must have the exact same rows and columns as the `r0` that would been automatically computed by `cmdnet`, which occurs after some processing to identify the subset of genes common across all datasets, so it is best to first let `cmdnet` compute `r0`, and then use its structure to filter a consensus network taken from a different source.

However, more importantly, it is possible to analyze the differences between the mean network computed by `cmdnet` and the user-defined network using the `cmdist` function. For example, the distance between the two reference networks `r1` and `r2`, can be found using `cmdist(r1, r2, type="network")`, and node-specific distances found using `cmdist(r1, r2, type="node")`.

3.1.2 Mean Correlation Matrix Deviation `cmd.meandev`

The summary measure of how similar two networks are is found by averaging the individual correlation matrix deviations, using the samples sizes as weights. This measure $\bar{\Delta}$ acts to quantify how variable the networks are across datasets.

```
result$cmd.meandev
```

```
## [1] 0.6425494
```

3.2 Mode "node"

The `"node"` mode concerns itself with computing node-specific information about deviations from the mean network. It returns 4 statistics, `node.dev`, `node.meandev`, `node.dev.cor1`, and `node.dev.cor2`, containing the deviations $\delta_{i,j}$ of each node in each dataset to the mean network, the mean deviation $\bar{\delta}_j$ of each node, the correlation matrix $\rho(\delta_{i,j})$ of the deviations for the datasets, and the correlation matrix $\rho(\delta_{j,i})$ of the deviations for the nodes.

```
system.time(result <- cmdnet(datalist, geneset, mode=c("node")))
```

```
##    user  system elapsed
##   0.127   0.005   0.112
```

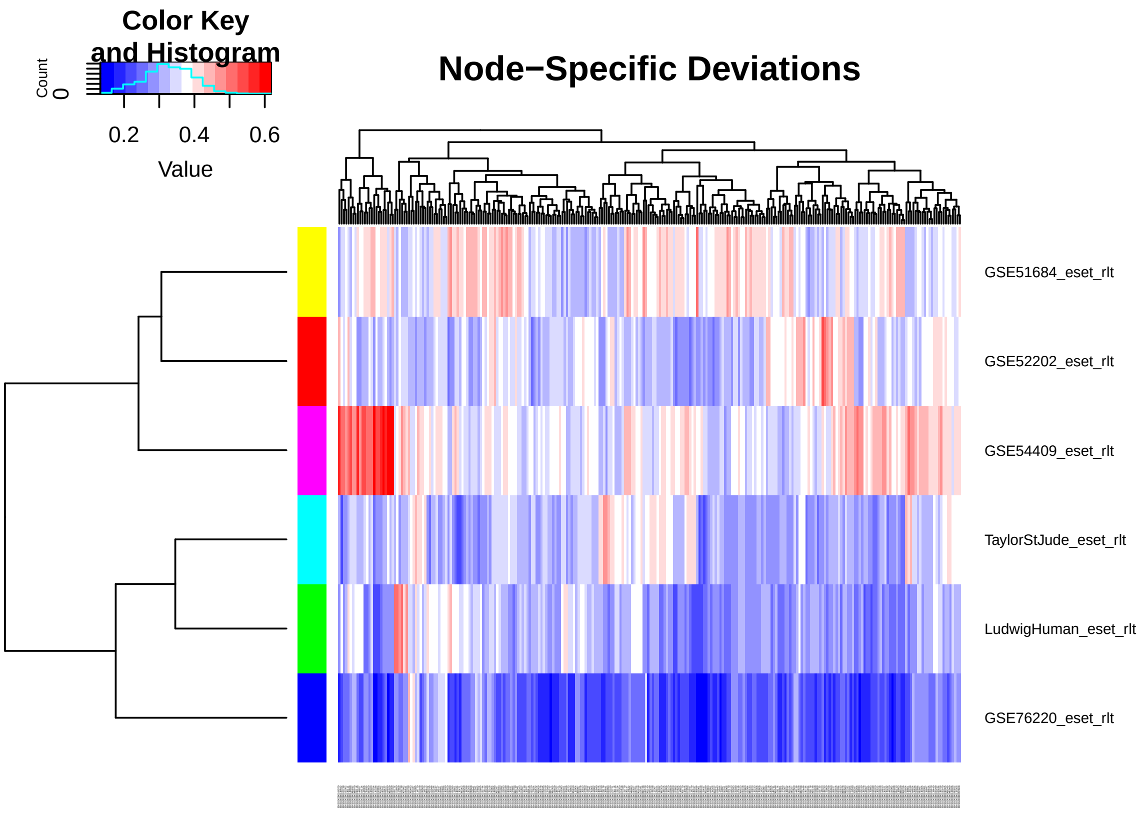
3.2.1 Node-Specific Deviations `node.dev`

The `node.dev` variable contains deviations $\delta_{i,j}$ for each of the nodes in each dataset from the mean network. We notice that the blue dataset (GSE76220) has much smaller deviations than the others. This again shows that it is most similar to the consensus network. There is also a cluster of genes in the magenta dataset that have very large deviations; this is the dataset with extremely small sample size (n=5). This suggests that only one cluster of genes is highly suspect in the magenta dataset, and that the dataset can be salvaged by removal of these genes.

```
head(result$node.dev)
```

```
##                    GSE52202_eset_rlt GSE51684_eset_rlt LudwigHuman_eset_rlt
## ENSG00000005810         0.3298853         0.3116496            0.3732977
## ENSG00000014641         0.3000437         0.4238474            0.3446232
## ENSG00000014824         0.3380262         0.3243531            0.2776487
## ENSG00000017260         0.3781585         0.3188503            0.2903541
## ENSG00000023287         0.2944509         0.3299243            0.2309700
## ENSG00000023516         0.3081468         0.3005257            0.2932052
##                    TaylorStJude_eset_rlt GSE76220_eset_rlt GSE54409_eset_rlt
## ENSG00000005810             0.2948955         0.1900783         0.3294017
## ENSG00000014641             0.2874839         0.2688502         0.3507485
## ENSG00000014824             0.3482314         0.2978118         0.4836664
## ENSG00000017260             0.3883510         0.1980214         0.3179879
## ENSG00000023287             0.2717407         0.1752339         0.4850514
## ENSG00000023516             0.2781057         0.2504491         0.3635375
```

```r
suppressMessages(require(gplots))
heatmap.2(t(result$node.dev), margins=c(2,8), cexRow=.7, cexCol=.1, scale="none",
        RowSideColors = rainbow(length(result$group.names)),
        trace="none", col="bluered", main="Node-Specific Deviations")
```



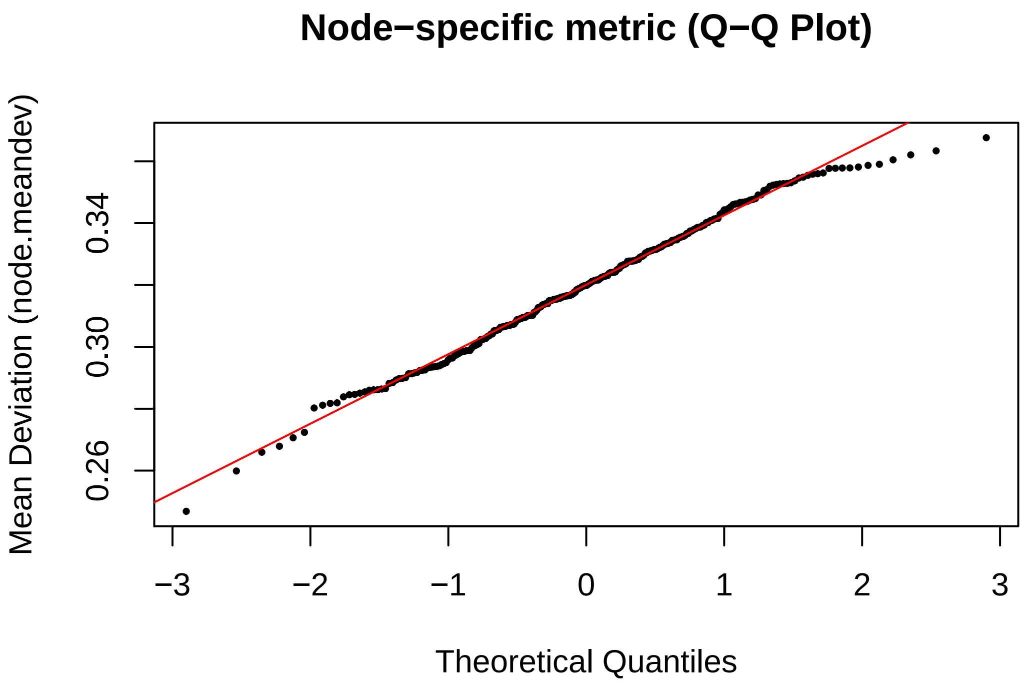
3.2.2 Node-Specific Mean Deviations `node.meandev`

The mean deviations $\bar{\delta}_j$=`node.meandev` are computed by weighted average of the node deviations across all datasets. Smaller mean deviations indicate better preservation across datasets, while larger deviations indicate changes in connectivity across datasets. We notice this score is normally distributed.

```
head(result$node.meandev)
```

```
## ENSG00000005810 ENSG00000014641 ENSG00000014824 ENSG00000017260
##       0.3041379       0.3323525       0.3220078       0.3102081
## ENSG00000023287 ENSG00000023516
##       0.2706011       0.2898673
```

```
# par(mar=c(5,3,2,2))
# barplot(sort(result$node.meandev, dec=T), horiz=T, las=2, xlab="node.meandev",
#         space=1000, cex.names=.1, main="Node-specific metric" )
# dotchart(sort(result$node.meandev, dec=T), cex=1, bg="white", color="blue",
#          main="Node-specific metric", xlab="Mean Deviation (node.meandev)",
#          ylab="Nodes", labels="")
par(mar=c(7,5,5,5))
qqnorm(result$node.meandev, pch=16, cex=.5,
       main = "Node-specific metric (Q-Q Plot)",
       ylab="Mean Deviation (node.meandev)")
qqline(result$node.meandev, col="red")
```



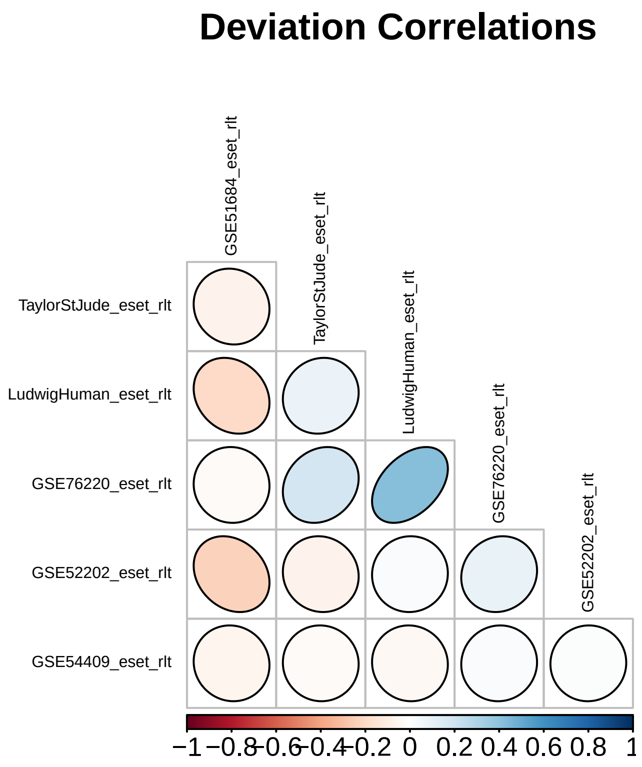
3.2.3 Node-Specific Deviation Correlations by Dataset `node.dev.cor1`

We compute correlations between deviations in the datasets by looking across nodes. We expect these deviations should be largely uncorrelated, and that correlations may indicate hidden biases among datasets. Here, we detect a moderate correlation (r=0.43) between two of the datasets, which we cannot explain.

```
result$node.dev.cor1
```

```
##                        GSE52202_eset_rlt GSE51684_eset_rlt
## GSE52202_eset_rlt             1.00000000       -0.22155078
## GSE51684_eset_rlt            -0.22155078        1.00000000
## LudwigHuman_eset_rlt          0.02203271       -0.19010958
## TaylorStJude_eset_rlt        -0.06171178       -0.06469524
## GSE76220_eset_rlt             0.09445736       -0.02566236
## GSE54409_eset_rlt             0.01645418       -0.05044505
##                        LudwigHuman_eset_rlt TaylorStJude_eset_rlt
## GSE52202_eset_rlt                0.02203271           -0.06171178
## GSE51684_eset_rlt               -0.19010958           -0.06469524
## LudwigHuman_eset_rlt             1.00000000            0.08364888
## TaylorStJude_eset_rlt            0.08364888            1.00000000
## GSE76220_eset_rlt                0.42839106            0.18022519
## GSE54409_eset_rlt               -0.03360746           -0.02010774
##                        GSE76220_eset_rlt GSE54409_eset_rlt
## GSE52202_eset_rlt             0.09445736        0.01645418
## GSE51684_eset_rlt            -0.02566236       -0.05044505
## LudwigHuman_eset_rlt          0.42839106       -0.03360746
## TaylorStJude_eset_rlt         0.18022519       -0.02010774
## GSE76220_eset_rlt             1.00000000        0.02943396
## GSE54409_eset_rlt             0.02943396        1.00000000
```

```
suppressMessages(require(corrplot))
suppressMessages(corrplot(result$node.dev.cor1, method="ellipse", diag=F, type="lower",
                          outline=T, order="hclust", hclust.method="ward",
                          tl.col="black", tl.cex=.5, mar=c(2,1,2,2),
                          main="Deviation Correlations"))
```



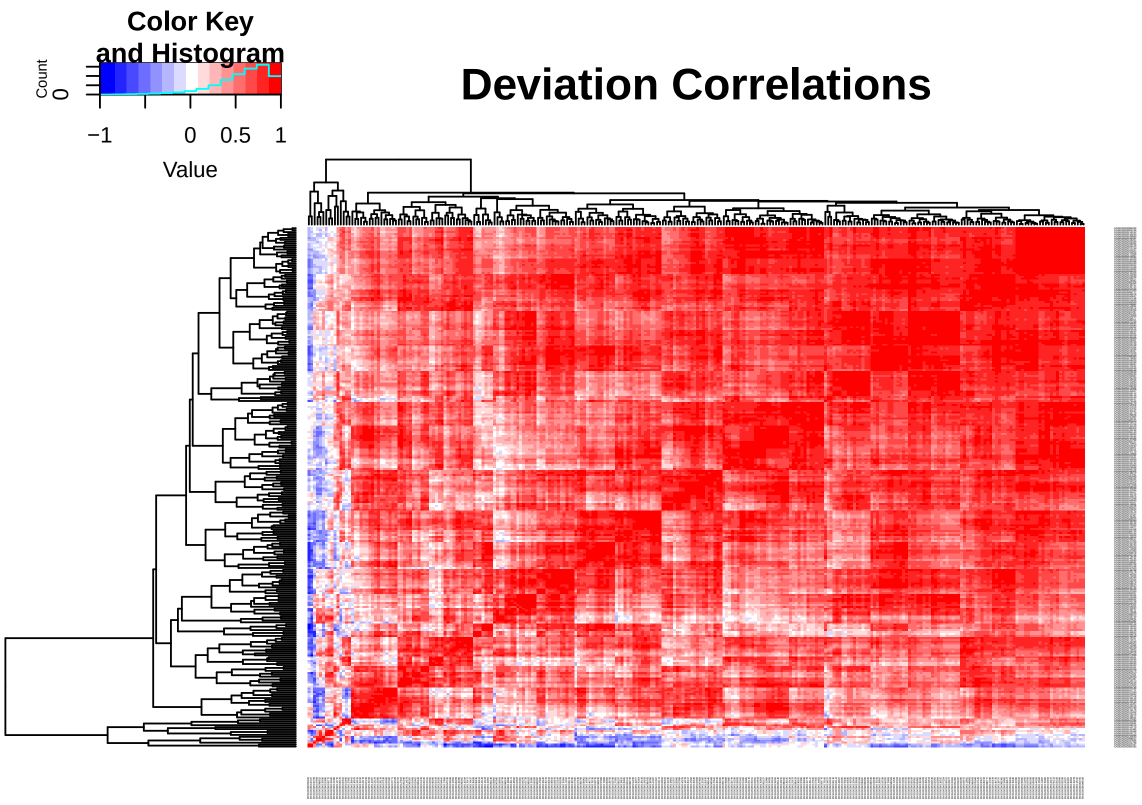
3.2.4 Node-Specific Deviation Correlations by Node `node.dev.cor2`

We also compute the correlations between node-specific deviations across datasets. Because some datasets are very much closer overall to the consensus network than others, systematic correlations in the deviations will be strongly present. Thus, we see that the deviations for most genes are highly correlated. This is tautological.

```
result$node.dev.cor2[1:5,1:5]
```

```
##                 ENSG00000005810 ENSG00000014641 ENSG00000014824
## ENSG00000005810       1.0000000       0.5056365       0.1711965
## ENSG00000014641       0.5056365       1.0000000       0.1419315
## ENSG00000014824       0.1711965       0.1419315       1.0000000
## ENSG00000017260       0.5945593       0.1137466       0.2919724
## ENSG00000023287       0.4402700       0.4890809       0.9203703
##                 ENSG00000017260 ENSG00000023287
## ENSG00000005810       0.5945593       0.4402700
## ENSG00000014641       0.1137466       0.4890809
## ENSG00000014824       0.2919724       0.9203703
## ENSG00000017260       1.0000000       0.3984033
## ENSG00000023287       0.3984033       1.0000000
```

```
heatmap.2(result$node.dev.cor2, margins=c(2,2), col="bluered",
          trace="none", cexRow=.1, cexCol=.1,
          scale="none", main="Deviation Correlations")
```



3.3 Mode "net"

The "net" mode concerns itself with computing pairwise correlation matrix distances, so that the datasets themselves may be viewed as a network. It returns only 1 statistic, net, containing the network similarity matrix. This matrix can be used to view the datasets as a network themselves.

```
system.time(result <- cmdnet(datalist, geneset, mode="net", metric="angle"))
```

```
##    user  system elapsed
##   0.046   0.008   0.054
```

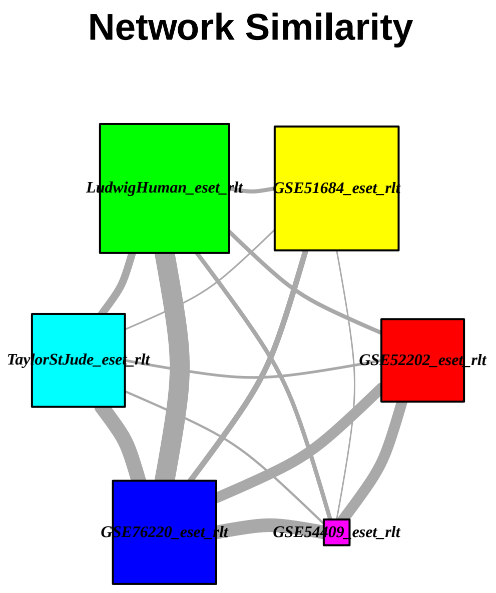
3.3.1 Network Similarity Matrix net

We view the network similarity matrix using a correlogram and a network. The size of each dataset is proportional to the sample size.

```
result$net
```

```
##                       GSE52202_eset_rlt GSE51684_eset_rlt
## GSE52202_eset_rlt            1.00000000        0.03381231
## GSE51684_eset_rlt            0.03381231        1.00000000
## LudwigHuman_eset_rlt         0.06065089        0.06050404
## TaylorStJude_eset_rlt        0.05137242        0.04462261
## GSE76220_eset_rlt            0.10294905        0.07117803
## GSE54409_eset_rlt            0.10437373        0.04362312
##                         LudwigHuman_eset_rlt TaylorStJude_eset_rlt
## GSE52202_eset_rlt                 0.06065089            0.05137242
## GSE51684_eset_rlt                 0.06050404            0.04462261
## LudwigHuman_eset_rlt              1.00000000            0.08369123
## TaylorStJude_eset_rlt             0.08369123            1.00000000
## GSE76220_eset_rlt                 0.16471004            0.13053249
## GSE54409_eset_rlt                 0.06149063            0.04860719
##                         GSE76220_eset_rlt GSE54409_eset_rlt
## GSE52202_eset_rlt              0.10294905        0.10437373
## GSE51684_eset_rlt              0.07117803        0.04362312
## LudwigHuman_eset_rlt           0.16471004        0.06149063
## TaylorStJude_eset_rlt          0.13053249        0.04860719
## GSE76220_eset_rlt              1.00000000        0.12327044
## GSE54409_eset_rlt              0.12327044        1.00000000
```

```r
adj <- result$net
diag(adj) <- NA
normalized = (adj-min(adj, na.rm=T))/(max(adj, na.rm=T)-min(adj, na.rm=T))
adj <- normalized

suppressMessages(require(igraph))
g <- graph_from_adjacency_matrix(adj,
                                 mode = c("undirected"),
                                 weighted = T, diag = F)
V(g)$color <- rainbow(length(V(g)))
plot.igraph(g,
            vertex.size = 3*result$group.n,
            vertex.shape = "square",
            vertex.label.font = 4,
            vertex.label.cex = .5,
            vertex.label.color = "black",
            edge.width = 10 * (E(g)$weight)^1,
            layout=layout.circle,
            main="Network Similarity",
            rescale = T, edge.curved=.2)
```



4 Noise Reduction

Finally, the `cmdnet` algorithm can be used to reduce noise across datasets by identifying subnetworks that are maximally preserved. Here we show that reduction in noise is possible by optimizing a subnetwork of smaller size.

4.1 Applied Post-Network Reconstruction

It is common practice to use correlation network analysis algorithms (e.g. WGCNA, MICA) to cluster full genomic datasets, compute module eigengenes, and finally identify “disease modules” related to phenotypes or traits. However, after a disease module is found, the process of understanding the module remains a difficult process that involves creative application of many bioinformatics tools, and which can still be complicated by the presence of hundreds or thousands of genes. In order to isolate even smaller gene sets, a variety of metrics or scoring systems are usually developed to indicate some property. For example, correlation with the phenotype or trait creates a kind of priority score for considering predictive genes. Additionally, metrics computed from the correlation network itself are also commonly used, such as degree connectivity or correlation with the module eigengene.

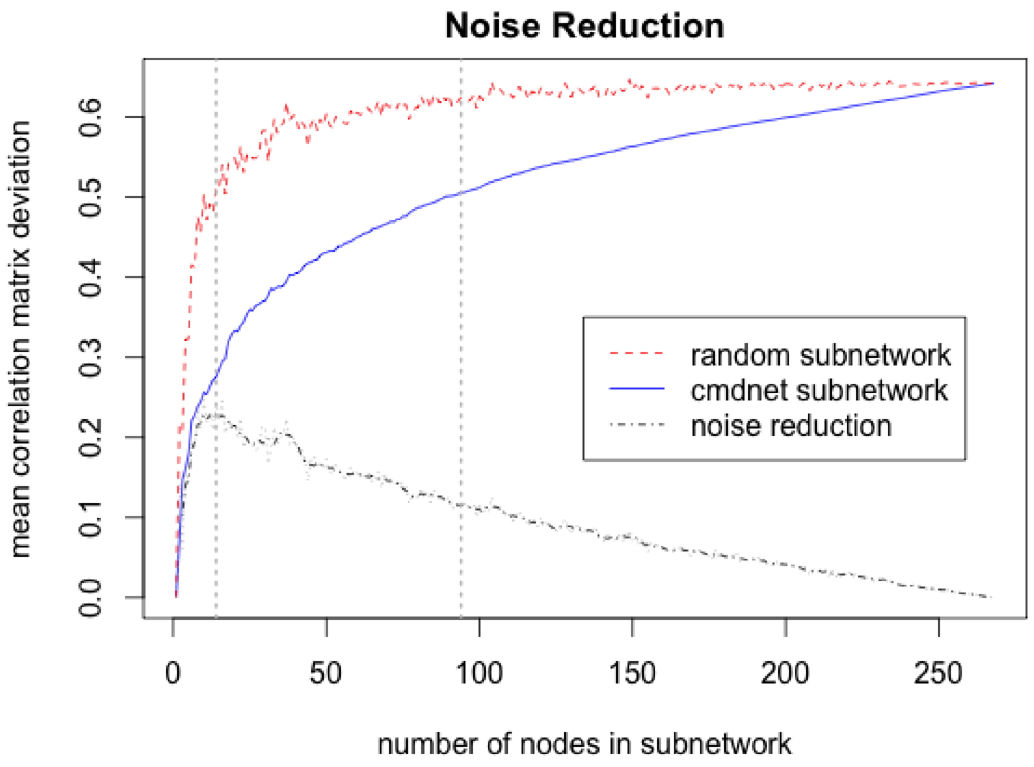
Here we propose a priority score based on the mean correlation deviation `node.meandev` for each node. Nodes with low mean correlation deviation are to be considered most reproducible when viewed across different datasets. It stands to reason that if we choose a subnetwork consisting of only nodes with the smallest mean correlation deviation, that subnetwork will be most reproducible.

Below, we demonstrate how this approach can be used to effectively reduce noise in network analysis across datasets. First, we restrict our analysis to a small gene list (268 genes) corresponding to a previously identified disease module. We build subnetworks of various sizes using this criteria, and compute the subnetwork’s mean correlation matrix deviation across datasets. To compute the effective noise reduction, we subtract this from the mean correlation matrix deviation found by creating a random subnetwork. We note that maximum noise reduction occurs for a set of about 15 nodes, and half-max at about 95 nodes.

```r
result <- cmdnet(datalist, geneset, mode=c("cmd","node"))

# sort nodes by mean deviation
node_priority <- sort(result$node.meandev, decreasing=F, index.return=T)
depth_size <- length(node_priority$x)

# compute cumulative mean deviation using cmdnet ordering
subnet_meandev <- sapply(1:depth_size,
                         function(depth) cmdnet(datalist,
                                                names(node_priority$x)[1:depth],
                                                mode="cmd")$cmd.meandev)
# compute cumulative mean deviation by select nodes at random
random_meandev <- sapply(1:depth_size,
                         function(depth) cmdnet(datalist,
                                                sample(names(node_priority$x), depth),
                                                mode="cmd")$cmd.meandev)

# compute noise reduction
NR <- random_meandev - subnet_meandev
ma <- function(x, n) filter(x,rep(1/n,n), sides=2) # moving average
maNR <- ma(NR, 5)
NRmax.val <- max(maNR, na.rm=T)
NRmax <- which(maNR==NRmax.val)
NRhalfmax <- which.min(abs(maNR - NRmax.val/2))
```

We plot the result:

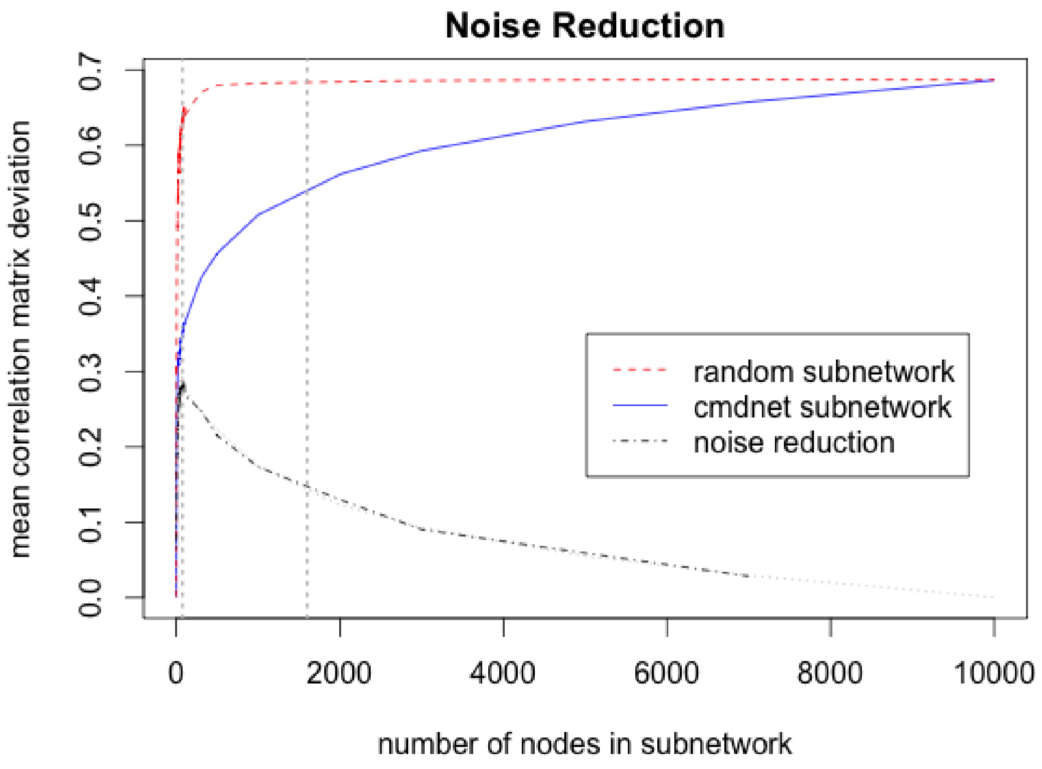
```r
par(mar=c(5,5,2,2))
matplot(1:depth_size,
        cbind(subnet_meandev, random_meandev, NR, maNR ),
        type="l", col=c("blue","red","grey","black"),
        xlab="number of nodes in subnetwork",
        ylab="mean correlation matrix deviation",
        main="Noise Reduction")
legend(depth_size/2,.35,
       legend=c("random subnetwork","cmdnet subnetwork", "noise reduction"),
       col=c("red","blue","black"), lty=c(2,1,4), cex=1)
abline(v=c(NRmax, NRhalfmax), lty="dotted", col="grey", lwd=2)
```



Next, we perform the same task, except this time on the full dataset. We build subnetworks of various sizes using this criteria, and compute the subnetwork's mean correlation matrix deviation across datasets. To compute the effective noise reduction, we subtract this from the mean correlation matrix deviation found by creating a random subnetwork. We find that maximum noise reduction occurs at around 75 nodes. The half-max is at about 1600 nodes.

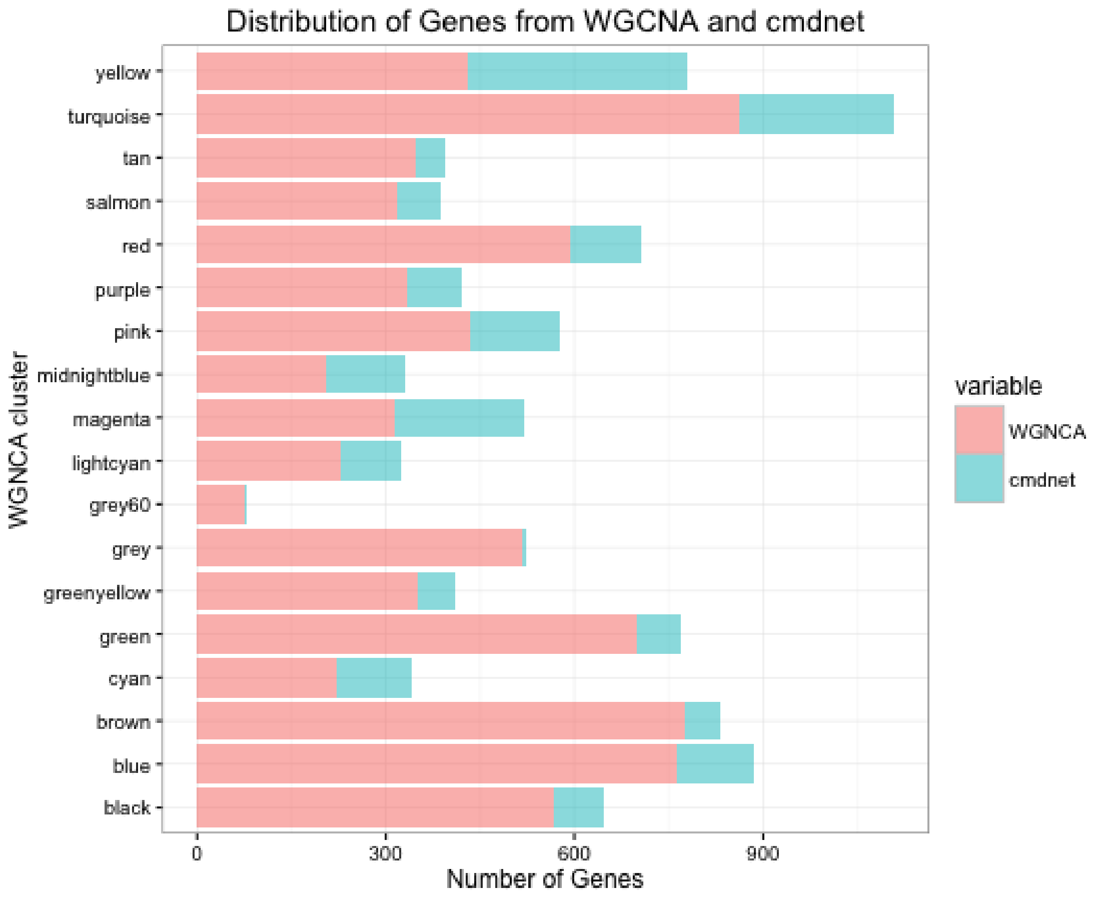
```r
result.full <- cmdnet(datalist, mode=c("cmd","node"))

# sort nodes by mean deviation
node_priority <- sort(result.full$node.meandev, decreasing=F, index.return=T)
depth_size <- length(node_priority$x)

lattice <- c(1:100,300, 500, 1000, 2000, 3000, 5000, 7000, 10000)

# THIS WILL TAKE A LONG TIME!!!
# compute cumulative mean deviation using cmdnet ordering
subnet_meandev <- sapply(lattice,
                         function(depth) cmdnet(datalist,
                                                names(node_priority$x)[1:depth],
                                                mode="cmd")$cmd.meandev)
# compute cumulative mean deviation by select nodes at random
random_meandev <- sapply(lattice,
                         function(depth) cmdnet(datalist,
                                                sample(names(node_priority$x), depth),
                                                        mode="cmd")$cmd.meandev)

# compute noise reduction
NR <- random_meandev - subnet_meandev
ma <- function(x, n) filter(x,rep(1/n,n), sides=2) # moving average
maNR <- ma(NR, 3)
NRmax.val <- max(maNR, na.rm=T)
NRmax <- which(maNR==NRmax.val)
NRhalfmax <- which.min(abs(maNR - NRmax.val/2))
NRhalfmax <- 1600
```

```
par(mar=c(5,5,2,2))
matplot(lattice,
        cbind(subnet_meandev, random_meandev, NR, maNR ),
        type="l", col=c("blue","red","grey","black"),
        xlab="number of nodes in subnetwork",
        ylab="mean correlation matrix deviation",
        main="Noise Reduction")
legend(depth_size/2,.35,
       legend=c("random subnetwork","cmdnet subnetwork", "noise reduction"),
       col=c("red","blue","black"), lty=c(2,1,4), cex=1)
abline(v=c(NRmax, NRhalfmax), lty="dotted", col="grey", lwd=2)
```



Finally, we are interested in knowing whether the least variable genes, the ones selected by `cmdnet`, are clustered in any significant way. For example, it is possible that some of the WGCNA modules have very stable connections, while most of the variability is distributed across other modules. On the other hand, if the most preserved subnetworks include nodes that are evenly distributed across modules, then the de-noising process can be seen as unbiased from the standpoint of WGCNA. This means it would be possible to apply the filter to reduce variability, while still capturing the span of functional groups that make up the systems biology.

To test this hypothesis, we select the top 2000 genes according to the `cmdnet` criteria, and identify which WGCNA modules they belong to. The WGCNA reconstruction here was performed on the average correlation matrix, which was subjected to soft-thresholding before topological overlap was computed. Hierarchical clustering on the topological overlap matrix discovered 18 modules, which were given corresponding color-labels. We then imposed those color-labels on the top 200 genes, which we show in a bar graph. Our result shows that the subnetworks identified by `cmdnet` are evenly distributed across the 18 modules, with the exception of one or two, suggesting that the de-noising is unbiased an maintains system-wide relationships. Interestingly, one of the modules that is deficient in `cmdnet` genes is the grey module, which is reserved for unclustered genes that do not have strong relationships. The other module with few `cmdnet` genes is grey60, which is very small to begin with.

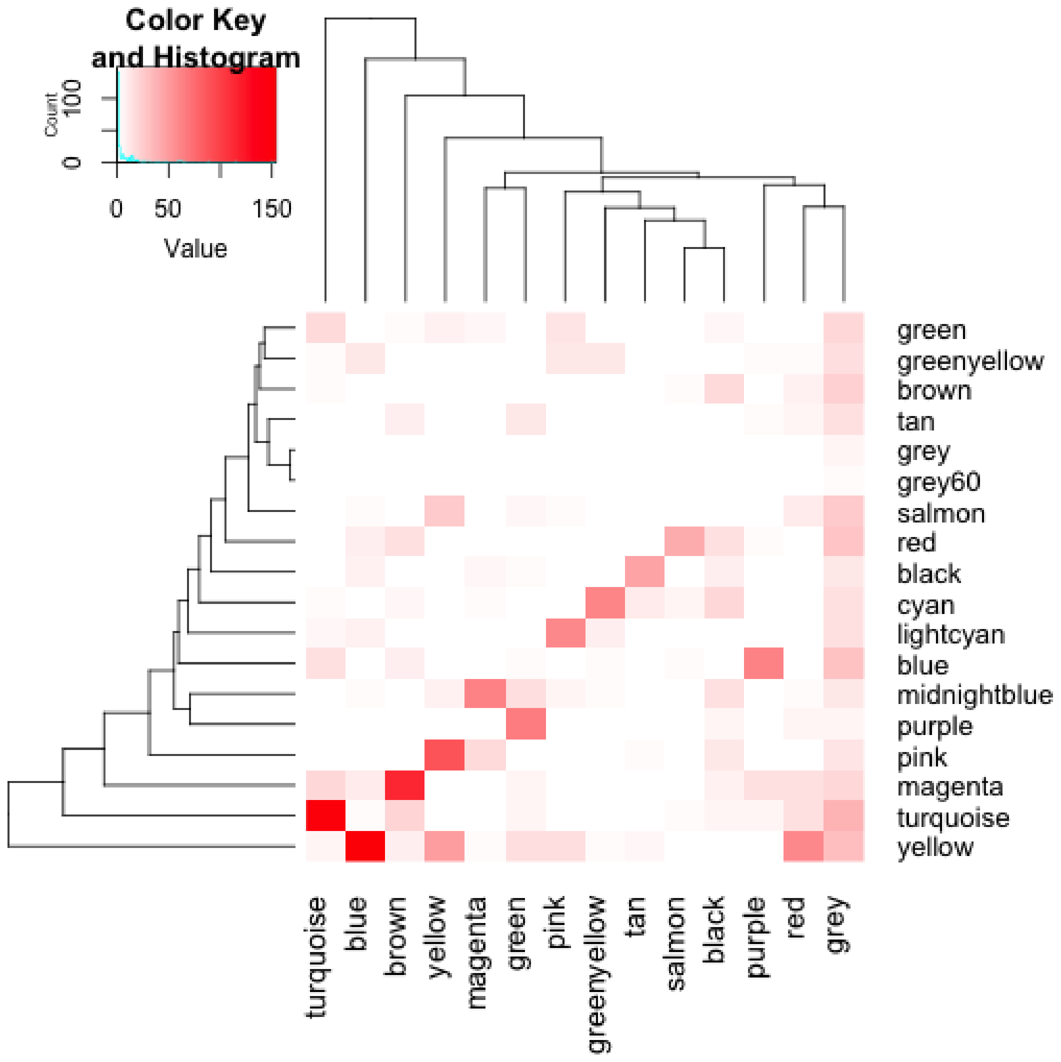


4.2 Applied Pre-Network Reconstruction

We can also use `cmdnet` on the full dataset before any network analysis algorithm is applied. Pre-conditioning filters on variability are already standard practice in genomic analyses. For example, when beginning a whole-transcriptome analysis, an initial array of >40k transcripts is always filtered for transcripts that have low variance, so that unchanging transcripts can be removed before focusing on differential expression.

Just as node expression variability can be used as a filter to improve differential expression analysis, it may be possible to improve network analysis by first filtering on node connectivity variability. When performing network analyses across several datasets, it is common practice to compute networks independently and just hope to find modules that are preserved across networks. Here we propose that `cmdnet` can be used to filter for nodes with the most consistent connectivity, thereby creating correlation matrices that are more similar, which should lead to more reproducible network analyses. Conversely, examining nodes that have high variability in connections may be of interest for identifying differential connectivity across datasets, a notion not yet explored in genomics, but which may prove important.

To examine the feasibility of such a filtering protocol, we apply the `cmdnet` filter prior to network reconstruction by WGCNA. We infer a coexpression network based on the top 2000 genes, which yields 14 modules, and ask how well these modules overlap with the 18 modules defined by full network reconstruction. To visualize the result we use a clustered heatmap, where rows are module labels found by the original full-network reconstruction, and columns are module labels found from the denoised reconstruction. The clustering is able to find a diagonal arrangement, indicating that the module identities are preserved even if denoising is applied before network inference.



5 Discussion

One of the major challenges in genomic research is integration of multiple datasets from the rapidly growing public data repositories. Large variability in protocols, from RNA extraction to data processing, has led to reproducibility concerns. Adding to the variability is an overwhelming number of small sample size experiments, typically no larger than 30 individuals. Of the machine learning methods used to make sense of such high-dimensional, low-sample size datasets, network analysis offers a number of promising tools. Firstly, they act as a means of data compression, identifying a small number of important features through use of clustering and principal component analysis. At the same time, this process provides a kind of noise reduction; since principal components or eigengenes are linear combinations of a large number of variables, they wash out individual fluctuations and keep a more robust composite feature.

Most of this ability comes from leveraging the large number of mutual correlations occuring within the data. But when comparing networks from multiple datasets, identifying stable network features still poses a problem. Understanding the sources of variability in these mutual correlations will lead to a better understanding of both the data, and of the network methods being applied. Here we presented a method for quantifying variability at the network and node-specific levels. By computing correlation matrix distance (CMD) statistics, our `cmdnet` package provides a fast algorithm for the detection of subnetworks preserved across datasets. We found that these subnetworks were evenly distributed across a modular structure inferred by WGCNA, indicating no bias with respect to the network topology. Furtherfore, we found that if the dataset is pre-filtered using `cmdnet` criteria, the resulting modular structure is approximately preserved, suggesting that noise reduction can be applied prior to network reconstruction to improve module preservation across datasets.