

# main

August 7, 2021

## 1 WGCNA

- @author = 'Apua Paquola'
- Edits by K.J. Benjamin
- Edits2 by Arthur S. Feltrin
  - New scale-free plots, export data to create network on cytoscape/igraph and format for jupyter notebook (05/2019)
  - Conversion from Rscript to jupyter notebook

Final edits by K.J. Benjamin for publication

```
[1]: PARAM_NETWORK_TYPE = 'signed'
```

### 1.1 Prepare Data and Traits Table

```
[2]: filter_outliers = function(expression, z_threshold = 2.5)
{
  # Input: an expression matrix
  # Output: an expression matrix with outliers removed
  # Remove samples with z normalized total distance from other samples >
  ↪ z_threshold

  sample_distance = dist(expression)
  dist_z = scale(colSums(as.matrix(sample_distance)))
  stopifnot(all(rownames(dist_z) == rownames(expression)))

  keepSamples = dist_z < z_threshold

  new_expression = expression[keepSamples,]
  new_expression
}
```

```
[3]: prepare_data=function()
{
  suppressMessages(library(dplyr))
  # Load sample data
```

```

sample_table = read.csv('/ceph/users/arthurfeltrin/phase3_new/
→0-annotation_data/Caudate_RiboZero_phenotype_SCHIZO-CAUC_AA-PRS_binary-table.
→csv')
#sample_table<-sample_table[-grep("BPNOS",sample_table$primarydx),]
sample_table = subset(sample_table, ageddeath >= 17)
rownames(sample_table) = sample_table[,62]
sample_table = subset(sample_table, select = c(ageddeath, sex, Race_CAUC,
→primarydx, RIN, antipsychotics))
colnames(sample_table)<-c("Age", "Sex", "Race_EA", "Primarydx", "RIN",
→"Antipsychotics_False")

#get DRD2 junction expression
load(".././differential_expression/_m/junctions/voomSVA.RData")
drd2j = v$E %>% as.data.frame %>% tibble::rownames_to_column() %>%
  filter(rowname %in% c('chr11:113412884-113415420(-)',
                        'chr11:113414462-113415420(-)',
                        'chr11:113412884-113414374(-)')) %>%
  tibble::column_to_rownames("rowname")

# Load residualized expression
load(".././differential_expression/_m/genes/voomSVA.RData")
vsd0 <- v$E %>% as.data.frame %>% replace(is.na(.), "")
vsd = rbind(vsd0, drd2j)
rm(vsd0)
dim(vsd)

# Keep only the columns and rows that are present in
# both the sample table and vsd file
samples = intersect(colnames(vsd), rownames(sample_table))
vsd = vsd[,samples]
sample_table = sample_table[samples,]

# WGCNA data import
library(WGCNA)
options(stringsAsFactors = FALSE)
datExpr0 = t(vsd)

# Remove offending genes and samples from the data
gsg = goodSamplesGenes(datExpr0, verbose = 3);
if (!gsg$allOK)
{
  datExpr0 = datExpr0[gsg$goodSamples, gsg$goodGenes]
}
datExpr=datExpr0
# Remove outliers
datExpr = filter_outliers(datExpr0, z_threshold = 2.0)
rm(datExpr0)

```

```

# Clean data
samples = intersect(rownames(datExpr), rownames(sample_table))
sample_table = sample_table[samples,]
datExpr = datExpr[samples,]
dim(datExpr)
save(datExpr, sample_table, file = '00.RData')
}

```

## 1.2 Create Sample Dendrogram Based on Distance (h)

```

[4]: prepare_traits = function()
{
  lnames = load('00.RData')
  # Associate traits with samples
  traitRows = match(rownames(datExpr), rownames(sample_table))
  datTraits = sample_table[traitRows,]
  # Diagnostic plot: Sample dendrogram and trait heatmap
  pdf(file='sample_dendrogram_and_trait_heatmap.pdf', height=16, width = 22)
  sampleTree2 = hclust(dist(datExpr), method = "average")
  # Convert traits to a color representation: white means
  # low, red means high, grey means missing entry
  traitColors = numbers2colors(traitRows, signed=FALSE);
  # Plot the sample dendrogram and the colors underneath.
  plotDendroAndColors(sampleTree2, traitColors, groupLabels="Avg. Counts",
    main = "Sample dendrogram and trait heatmap",
    cex.dendroLabels=0.7)

  dev.off()
  # Print output
  plotDendroAndColors(sampleTree2, traitColors, groupLabels="Avg. Counts",
    main = "Sample dendrogram and trait heatmap",
    cex.dendroLabels=0.75)
  save(datExpr, sample_table, datTraits, file = "01.RData")
}

```

## 1.3 Calculate Scale-Free Topology

```

[5]: plot_power_parameter=function(datExpr, plot_filename)
{
  # Choose a set of soft-thresholding powers
  powers = seq(from = 1, to=30, by=1)
  # Call the network topology analysis function
  sft = pickSoftThreshold(datExpr, networkType = PARAM_NETWORK_TYPE,
    powerVector = powers, verbose = 5)

  # Plot the results:
  pdf(file=plot_filename)
  par(mfcol = c(2,2));
  par(mar = c(4.2, 4.5 , 2.2, 0.5), oma=c(0,0,2,0))

```

```

cex1 = 0.7;
# Scale-free topology fit index as a function of the
# soft-thresholding power
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="blue");
# this line corresponds to using an R^2 cut-off of h
abline(h=0.80,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 = paste("Mean connectivity"))
text(sft$fitIndices[,1], sft$fitIndices[,5], labels=powers,
      ↪cex=cex1,col="blue")
#####
plot(sft$fitIndices[,1], sft$fitIndices[,6],
      xlab="Soft Threshold (power)", ylab="Median Connectivity",
      type="n", main = paste("Median connectivity"))
text(sft$fitIndices[,1], sft$fitIndices[,6], labels=powers,
      ↪cex=cex1,col="blue")
#####
plot(sft$fitIndices[,1], sft$fitIndices[,7],
      xlab="Soft Threshold (power)", ylab="Max Connectivity",
      type="n", main = paste("Max connectivity"))
text(sft$fitIndices[,1], sft$fitIndices[,7], labels=powers,
      ↪cex=cex1,col="blue")
dev.off()
####plot on jupyter
par(mfcol = c(2,2));
par(mar = c(4.2, 4.5 , 2.2, 0.5),oma=c(0,0,2,0))
cex1 = 0.7;
# Scale-free topology fit index as a function of the
# soft-thresholding power
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="blue");
# this line corresponds to using an R^2 cut-off of h
abline(h=0.80,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 = paste("Mean connectivity"))
    text(sft$fitIndices[,1], sft$fitIndices[,5], labels=powers,
    ↪ cex=cex1,col="blue")
    #####
    plot(sft$fitIndices[,1], sft$fitIndices[,6],
        xlab="Soft Threshold (power)", ylab="Median Connectivity",
        type="n", main = paste("Median connectivity"))
    text(sft$fitIndices[,1], sft$fitIndices[,6], labels=powers,
    ↪ cex=cex1,col="blue")
    #####
    plot(sft$fitIndices[,1], sft$fitIndices[,7],
        xlab="Soft Threshold (power)", ylab="Max Connectivity", type="n",
        main = paste("Max connectivity"))
    text(sft$fitIndices[,1], sft$fitIndices[,7], labels=powers,
    ↪ cex=cex1,col="blue")
}

```

```

[6]: figure_out_power_parameter=function()
{
    library(WGCNA)
    options(stringsAsFactors = FALSE);
    enableWGCNAThreads(nThreads=16)
    lnames = load(file = '01.RData')
    plot_power_parameter(datExpr, 'power_parameter_selection.pdf')
}

```

## 1.4 Build the Network

```

[7]: construct_network=function()
{
    library(WGCNA)
    options(stringsAsFactors = FALSE);
    enableWGCNAThreads(nThreads=16)
    lnames = load(file = "01.RData")

    # softPower value from previous plot power_parameter_selection.pdf
    softPower = 7; #check this value, it changes accordingly to your data! You
    ↪ should
    # ALWAYS choose a value equal or above (better) 0.8
    cor <- WGCNA::cor
    net = blockwiseModules(datExpr,
        power = softPower,
        networkType = PARAM_NETWORK_TYPE,
        TOMType = PARAM_NETWORK_TYPE,
        numericLabels = TRUE,
        corType = "bicor",
        saveTOMs = TRUE, saveTOMFileBase = "TOM",

```

```

        verbose = 3, maxBlockSize=30000)

moduleLabels = net$colors
moduleColors = labels2colors(net$colors)
MEs = net$MEs;
geneTree = net$dendrograms[[1]];
save(net, MEs, moduleLabels, moduleColors, geneTree, softPower, file = "02.
→RData")
}
#cyt = exportNetworkToCytoscape(modTOM,

```

### 1.5 Use Topology Overlap Matrix (TOM) to cluster the genes on the networks into different modules

```

[8]: plot_cluster_dendrogram=function()
{
  library(WGCNA)
  options(stringsAsFactors = FALSE);
  enableWGCNAThreads(nThreads=16)
  load(file = "02.RData")
  pdf(file="cluster_dendrogram.pdf",height=16,width = 22)
  mergedColors = labels2colors(net$colors)
  plotDendroAndColors(net$dendrograms[[1]], mergedColors[net$blockGenes[[1]]],
    "Module Colors", dendroLabels = FALSE, hang = 0.03,
    addGuide = TRUE, guideHang = 0.05, cex.dendroLabels=0.3)

  dev.off()
  # Print output
  plotDendroAndColors(net$dendrograms[[1]], mergedColors[net$blockGenes[[1]]],
    "Module Colors", dendroLabels = FALSE, hang = 0.03,
    addGuide = TRUE, guideHang = 0.05, cex.dendroLabels=0.3)
}

```

### 1.6 Use Pearson Correlation to measure the correlation between each module eigenvalue (kME) and the various sample traits

```

[9]: correlate_with_traits=function()
{
  library(WGCNA)
  options(stringsAsFactors = FALSE)
  enableWGCNAThreads(nThreads=16)
  lnames = load(file = "01.RData")
  lnames = load(file = "02.RData")
  # Define numbers of genes and samples
  nGenes = ncol(datExpr);
  nSamples = nrow(datExpr);
  # Recalculate MEs with color labels
  MEs0 = moduleEigengenes(datExpr, moduleColors)$eigengenes

```

```

MEs = orderMEs(MEs0)
moduleTraitCor = cor(MEs, datTraits, use = "p");
moduleTraitPvalue = corPvalueStudent(moduleTraitCor, nSamples);
# Plot
pdf(file="module_trait_relationships.pdf", height=16,width = 22)
# Will display correlations and their p-values
textMatrix = paste(signif(moduleTraitCor, 2), "\n(",
                    signif(moduleTraitPvalue, 1), ")", sep = "");
dim(textMatrix) = dim(moduleTraitCor)
par(mar = c(6, 8.5, 3, 3));
# Display the correlation values within a heatmap plot
labeledHeatmap(Matrix = moduleTraitCor,
                xLabels = names(datTraits),
                yLabels = names(MEs),
                ySymbols = names(MEs),
                colorLabels = FALSE,
                naColor = "grey",
                colors = blueWhiteRed(50),
                textMatrix = textMatrix,
                setStdMargins = FALSE,
                cex.text = 0.9,
                zlim = c(-1,1),
                main = paste("Module kME-Trait Correlation"))

dev.off()
# Print output
textMatrix = paste(signif(moduleTraitCor, 2), "\n(",
                    signif(moduleTraitPvalue, 1), ")", sep = "");
dim(textMatrix) = dim(moduleTraitCor)
par(mar = c(12, 6.5, 3, 0.5));
# Display the correlation values within a heatmap plot
labeledHeatmap(Matrix = moduleTraitCor, xLabels = names(datTraits),
                yLabels = names(MEs), ySymbols = names(MEs),
                colorLabels = FALSE, naColor = "grey",
                colors = blueWhiteRed(50), textMatrix = textMatrix,
                setStdMargins = FALSE, cex.text = 0.55, zlim = c(-1,1),
                main = paste("Module kME-Trait Correlation"))
}

```

## 1.7 Export the main results

```

[10]: export_eigengene_tables = function()
{
  library(WGCNA)
  options(stringsAsFactors = FALSE)
  lnames = load(file = "01.RData")
  lnames = load(file = "02.RData")
  # Define numbers of genes and samples

```

```

nGenes = ncol(datExpr)
nSamples = nrow(datExpr)
# Recalculate MEs with color labels
MEs0 = moduleEigengenes(datExpr, moduleColors)$eigengenes
rownames(MEs0) = rownames(datExpr)
write.csv(MEs0, 'eigengenes.csv')
# Write modules
modules = data.frame(row.names=colnames(datExpr), module=moduleColors)
write.csv(modules, 'modules.csv')
save(datExpr,softPower,moduleColors, file = "cytoscapenetwork.Rdata")
}

```

## 1.8 Run the functions and plot the results

```
[11]: prepare_data()
```

Loading required package: limma

Loading required package: dynamicTreeCut

Loading required package: fastcluster

Attaching package: ‘fastcluster’

The following object is masked from ‘package:stats’:

hclust

Attaching package: ‘WGCNA’

The following object is masked from ‘package:stats’:

cor

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

```
[12]: # 1 - Sample dendrogram and trait heatmap
prepare_traits()
```



A dendrogram illustrating hierarchical clustering of 100 samples based on their average counts. The y-axis represents 'Height' (ranging from 80 to 160), and the x-axis represents 'Avg. Counts' (ranging from 0.00 to 0.04). The samples are labeled with IDs such as R1308, R1310, R1312, R1314, R1316, R1318, R1320, R1322, R1324, R1326, R1328, R1330, R1332, R1334, R1336, R1338, R1340, R1342, R1344, R1346, R1348, R1350, R1352, R1354, R1356, R1358, R1360, R1362, R1364, R1366, R1368, R1370, R1372, R1374, R1376, R1378, R1380, R1382, R1384, R1386, R1388, R1390, R1392, R1394, R1396, R1398, R1400, R1402, R1404, R1406, R1408, R1410, R1412, R1414, R1416, R1418, R1420, R1422, R1424, R1426, R1428, R1430, R1432, R1434, R1436, R1438, R1440, R1442, R1444, R1446, R1448, R1450, R1452, R1454, R1456, R1458, R1460, R1462, R1464, R1466, R1468, R1470, R1472, R1474, R1476, R1478, R1480, R1482, R1484, R1486, R1488, R1490, R1492, R1494, R1496, R1498, R1500, R1502, R1504, R1506, R1508, R1510, R1512, R1514, R1516, R1518, R1520, R1522, R1524, R1526, R1528, R1530, R1532, R1534, R1536, R1538, R1540, R1542, R1544, R1546, R1548, R1550, R1552, R1554, R1556, R1558, R1560, R1562, R1564, R1566, R1568, R1570, R1572, R1574, R1576, R1578, R1580, R1582, R1584, R1586, R1588, R1590, R1592, R1594, R1596, R1598, R1600, R1602, R1604, R1606, R1608, R1610, R1612, R1614, R1616, R1618, R1620, R1622, R1624, R1626, R1628, R1630, R1632, R1634, R1636, R1638, R1640, R1642, R1644, R1646, R1648, R1650, R1652, R1654, R1656, R1658, R1660, R1662, R1664, R1666, R1668, R1670, R1672, R1674, R1676, R1678, R1680, R1682, R1684, R1686, R1688, R1690, R1692, R1694, R1696, R1698, R1700, R1702, R1704, R1706, R1708, R1710, R1712, R1714, R1716, R1718, R1720, R1722, R1724, R1726, R1728, R1730, R1732, R1734, R1736, R1738, R1740, R1742, R1744, R1746, R1748, R1750, R1752, R1754, R1756, R1758, R1760, R1762, R1764, R1766, R1768, R1770, R1772, R1774, R1776, R1778, R1780, R1782, R1784, R1786, R1788, R1790, R1792, R1794, R1796, R1798, R1800, R1802, R1804, R1806, R1808, R1810, R1812, R1814, R1816, R1818, R1820, R1822, R1824, R1826, R1828, R1830, R1832, R1834, R1836, R1838, R1840, R1842, R1844, R1846, R1848, R1850, R1852, R1854, R1856, R1858, R1860, R1862, R1864, R1866, R1868, R1870, R1872, R1874, R1876, R1878, R1880, R1882, R1884, R1886, R1888, R1890, R1892, R1894, R1896, R1898, R1900, R1902, R1904, R1906, R1908, R1910, R1912, R1914, R1916, R1918, R1920, R1922, R1924, R1926, R1928, R1930, R1932, R1934, R1936, R1938, R1940, R1942, R1944, R1946, R1948, R1950, R1952, R1954, R1956, R1958, R1960, R1962, R1964, R1966, R1968, R1970, R1972, R1974, R1976, R1978, R1980, R1982, R1984, R1986, R1988, R1990, R1992, R1994, R1996, R1998, R2000, R2002, R2004, R2006, R2008, R2010, R2012, R2014, R2016, R2018, R2020, R2022, R2024, R2026, R2028, R2030, R2032, R2034, R2036, R2038, R2040, R2042, R2044, R2046, R2048, R2050, R2052, R2054, R2056, R2058, R2060, R2062, R2064, R2066, R2068, R2070, R2072, R2074, R2076, R2078, R2080, R2082, R2084, R2086, R2088, R2090, R2092, R2094, R2096, R2098, R2100, R2102, R2104, R2106, R2108, R2110, R2112, R2114, R2116, R2118, R2120, R2122, R2124, R2126, R2128, R2130, R2132, R2134, R2136, R2138, R2140, R2142, R2144, R2146, R2148, R2150, R2152, R2154, R2156, R2158, R2160, R2162, R2164, R2166, R2168, R2170, R2172, R2174, R2176, R2178, R2180, R2182, R2184, R2186, R2188, R2190, R2192, R2194, R2196, R2198, R2200, R2202, R2204, R2206, R2208, R2210, R2212, R2214, R2216, R2218, R2220, R2222, R2224, R2226, R2228, R2230, R2232, R2234, R2236, R2238, R2240, R2242, R2244, R2246, R2248, R2250, R2252, R2254, R2256, R2258, R2260, R2262, R2264, R2266, R2268, R2270, R2272, R2274, R2276, R2278, R2280, R2282, R2284, R2286, R2288, R2290, R2292, R2294, R2296, R2298, R2300, R2302, R2304, R2306, R2308, R2310, R2312, R2314, R2316, R2318, R2320, R2322, R2324, R2326, R2328, R2330, R2332, R2334, R2336, R2338, R2340, R2342, R2344, R2346, R2348, R2350, R2352, R2354, R2356, R2358, R2360, R2362, R2364, R2366, R2368, R2370, R2372, R2374, R2376, R2378, R2380, R2382, R2384, R2386, R2388, R2390, R2392, R2394, R2396, R2398, R2400, R2402, R2404, R2406, R2408, R2410, R2412, R2414, R2416, R2418, R2420, R2422, R2424, R2426, R2428, R2430, R2432, R2434, R2436, R2438, R2440, R2442, R2444, R2446, R2448, R2450, R2452, R2454, R2456, R2458, R2460, R2462, R2464, R2466, R2468, R2470, R2472, R2474, R2476, R2478, R2480, R2482, R2484, R2486, R2488, R2490, R2492, R2494, R2496, R2498, R2500, R2502, R2504, R2506, R2508, R2510, R2512, R2514, R2516, R2518, R2520, R2522, R2524, R2526, R2528, R2530, R2532, R2534, R2536, R2538, R2540, R2542, R2544, R2546, R2548, R2550, R2552, R2554, R2556, R2558, R2560, R2562, R2564, R2566, R2568, R2570, R2572, R2574, R2576, R2578, R2580, R2582, R2584, R2586, R2588, R2590, R2592, R2594, R2596, R2598, R2600, R2602, R2604, R2606, R26

```
# 2 - Scale Free Topology Model Fit
figure_out_power_parameter()
```

Allowing parallel execution with up to 16 working processes.

```
pickSoftThreshold: will use block size 1948.
```

```
pickSoftThreshold: calculating connectivity for given powers...
```

```
..working on genes 1 through 1948 of 22961
```

```
..working on genes 1949 through 3896 of 22961
```

```
..working on genes 3897 through 5844 of 22961
```

```
..working on genes 5845 through 7792 of 22961
```

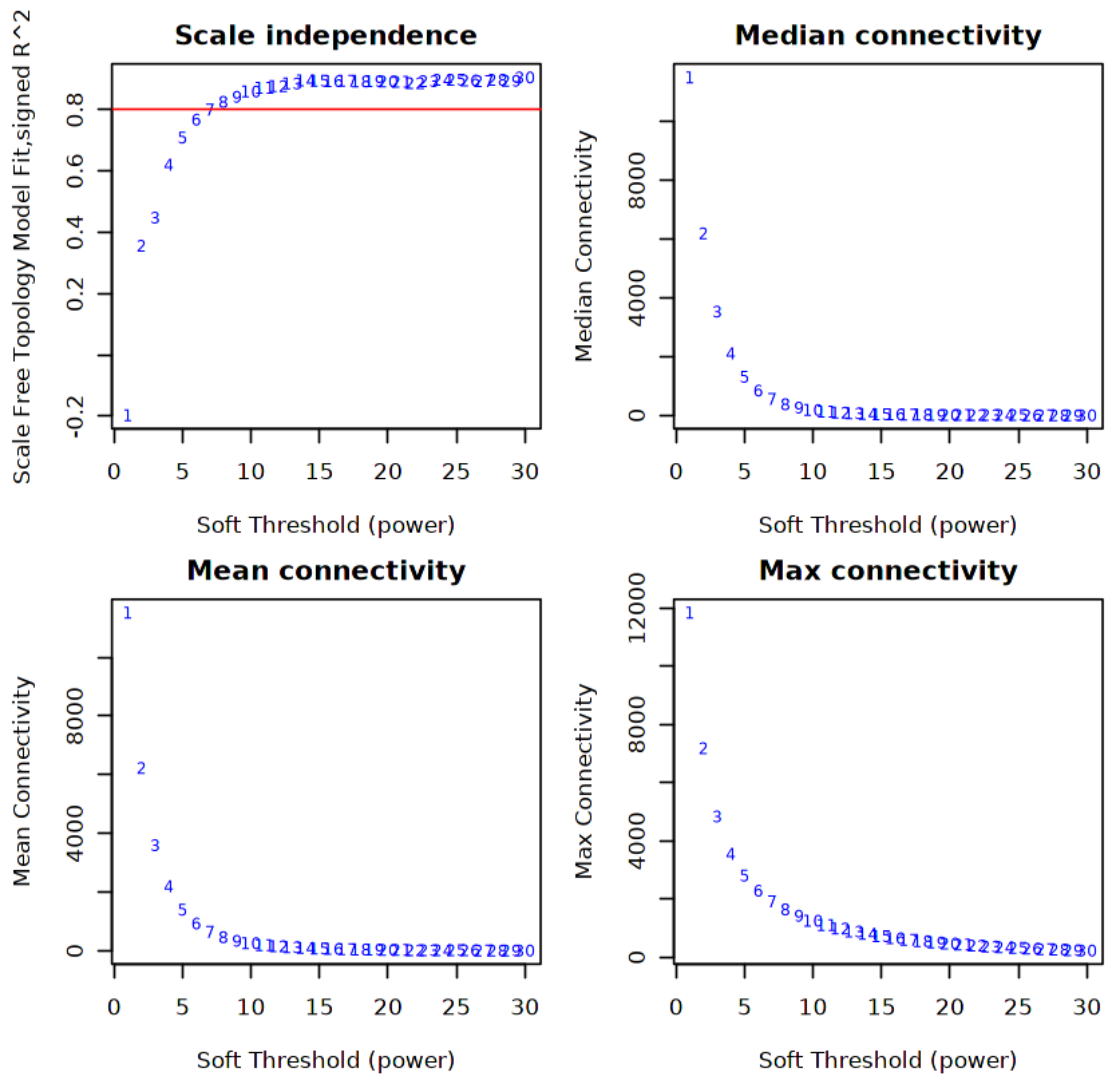
```
..working on genes 7793 through 9740 of 22961
```

```
..working on genes 9741 through 11688 of 22961
```

```
..working on genes 11689 through 13636 of 22961
```

..working on genes 13637 through 15584 of 22961  
 ..working on genes 15585 through 17532 of 22961  
 ..working on genes 17533 through 19480 of 22961  
 ..working on genes 19481 through 21428 of 22961  
 ..working on genes 21429 through 22961 of 22961

	Power	SFT.R.sq	slope	truncated.R.sq	mean.k.	median.k.	max.k.
1	1	0.196	56.00	0.951	11500.00	1.15e+04	11800
2	2	0.356	-10.60	0.933	6240.00	6.20e+03	7190
3	3	0.448	-3.90	0.864	3600.00	3.54e+03	4840
4	4	0.621	-2.50	0.924	2190.00	2.11e+03	3550
5	5	0.709	-2.21	0.941	1400.00	1.31e+03	2800
6	6	0.768	-1.98	0.957	935.00	8.40e+02	2290
7	7	0.801	-1.85	0.963	647.00	5.52e+02	1930
8	8	0.826	-1.77	0.966	462.00	3.71e+02	1650
9	9	0.842	-1.73	0.967	340.00	2.55e+02	1430
10	10	0.860	-1.69	0.970	256.00	1.78e+02	1250
11	11	0.872	-1.66	0.971	197.00	1.26e+02	1110
12	12	0.877	-1.66	0.968	155.00	9.03e+01	988
13	13	0.886	-1.65	0.970	123.00	6.55e+01	887
14	14	0.896	-1.64	0.972	99.60	4.77e+01	800
15	15	0.895	-1.64	0.966	81.50	3.52e+01	725
16	16	0.894	-1.64	0.964	67.50	2.62e+01	660
17	17	0.895	-1.63	0.963	56.40	1.96e+01	602
18	18	0.892	-1.63	0.960	47.60	1.49e+01	552
19	19	0.893	-1.61	0.959	40.40	1.13e+01	507
20	20	0.893	-1.61	0.957	34.60	8.62e+00	468
21	21	0.891	-1.60	0.955	29.80	6.64e+00	432
22	22	0.887	-1.60	0.951	25.80	5.15e+00	400
23	23	0.893	-1.59	0.954	22.50	3.99e+00	371
24	24	0.898	-1.58	0.957	19.70	3.12e+00	345
25	25	0.898	-1.57	0.957	17.30	2.45e+00	321
26	26	0.896	-1.56	0.955	15.30	1.93e+00	299
27	27	0.894	-1.56	0.954	13.60	1.52e+00	280
28	28	0.898	-1.55	0.957	12.10	1.21e+00	262
29	29	0.895	-1.55	0.954	10.80	9.61e-01	245
30	30	0.906	-1.53	0.961	9.65	7.63e-01	230



```
[14]: construct_network()
```

Allowing parallel execution with up to 16 working processes.

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 16 parallel threads.

Fraction of slow calculations: 0.000000

..connectivity..

..matrix multiplication (system BLAS)..

..normalization..

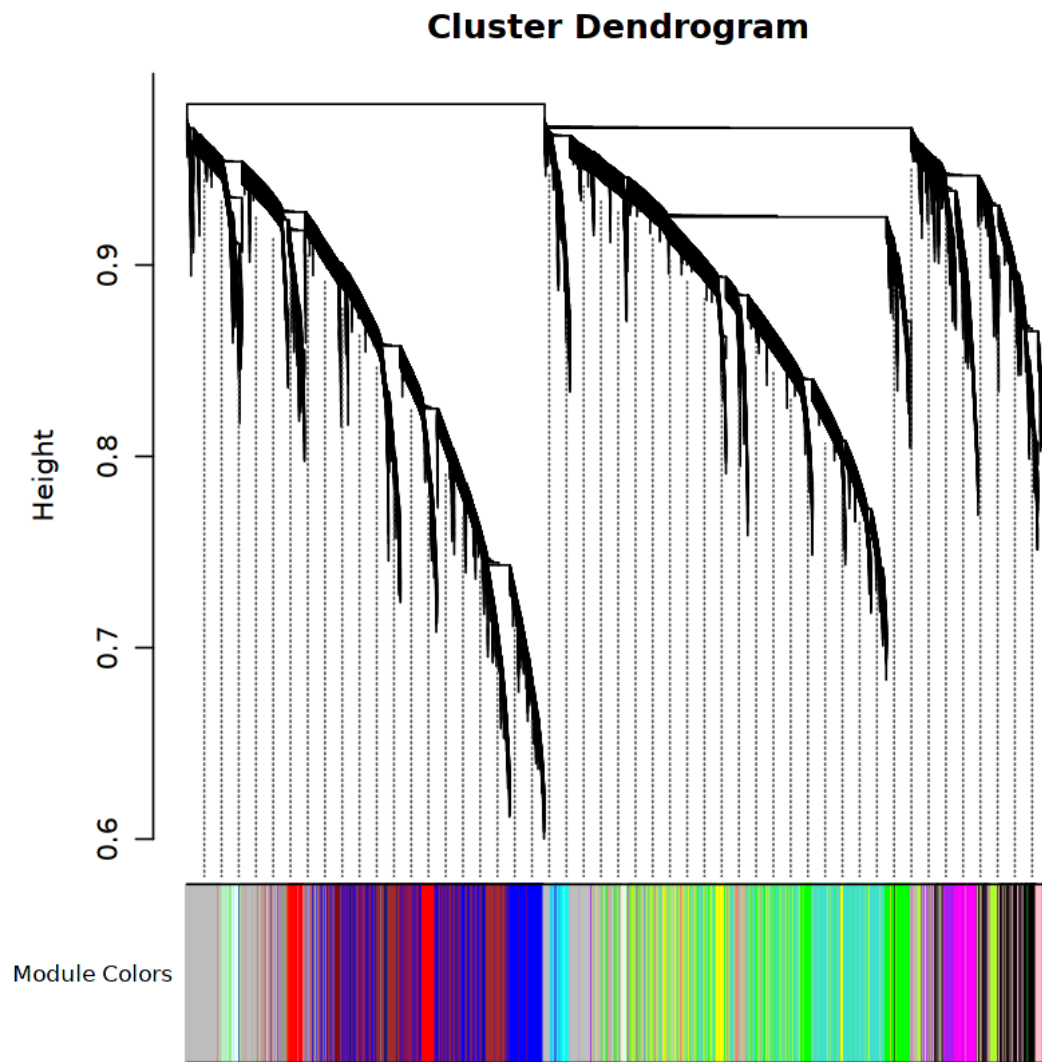
```

..done.
..saving TOM for block 1 into file TOM-block.1.RData
...clustering..
...detecting modules..
...calculating module eigengenes..
...checking kME in modules..
    ..removing 798 genes from module 1 because their KME is too low.
    ..removing 1179 genes from module 2 because their KME is too low.
    ..removing 733 genes from module 3 because their KME is too low.
    ..removing 733 genes from module 4 because their KME is too low.
    ..removing 210 genes from module 5 because their KME is too low.
    ..removing 138 genes from module 6 because their KME is too low.
    ..removing 104 genes from module 7 because their KME is too low.
    ..removing 1 genes from module 8 because their KME is too low.
    ..removing 3 genes from module 10 because their KME is too low.
    ..removing 30 genes from module 11 because their KME is too low.
    ..removing 1 genes from module 18 because their KME is too low.
    ..removing 3 genes from module 19 because their KME is too low.
    ..removing 1 genes from module 20 because their KME is too low.
    ..removing 4 genes from module 22 because their KME is too low.
    ..removing 2 genes from module 23 because their KME is too low.
..reassigning 777 genes from module 1 to modules with higher KME.
..reassigning 670 genes from module 2 to modules with higher KME.
..reassigning 525 genes from module 3 to modules with higher KME.
..reassigning 500 genes from module 4 to modules with higher KME.
..reassigning 216 genes from module 5 to modules with higher KME.
..reassigning 179 genes from module 6 to modules with higher KME.
..reassigning 101 genes from module 7 to modules with higher KME.
..reassigning 55 genes from module 8 to modules with higher KME.
..reassigning 127 genes from module 9 to modules with higher KME.
..reassigning 67 genes from module 10 to modules with higher KME.
..reassigning 44 genes from module 11 to modules with higher KME.
..reassigning 75 genes from module 12 to modules with higher KME.
..reassigning 62 genes from module 13 to modules with higher KME.
..reassigning 74 genes from module 14 to modules with higher KME.
..reassigning 19 genes from module 15 to modules with higher KME.
..reassigning 10 genes from module 16 to modules with higher KME.
..reassigning 28 genes from module 17 to modules with higher KME.
..reassigning 23 genes from module 18 to modules with higher KME.
..reassigning 3 genes from module 20 to modules with higher KME.
..reassigning 7 genes from module 21 to modules with higher KME.
..reassigning 17 genes from module 22 to modules with higher KME.
..reassigning 7 genes from module 23 to modules with higher KME.
..reassigning 1 genes from module 24 to modules with higher KME.
..reassigning 1 genes from module 25 to modules with higher KME.
..merging modules that are too close..
    mergeCloseModules: Merging modules whose distance is less than 0.15
    Calculating new MEs...

```

```
[15]: #3 - TOM Dendrogram  
plot_cluster_dendrogram()
```

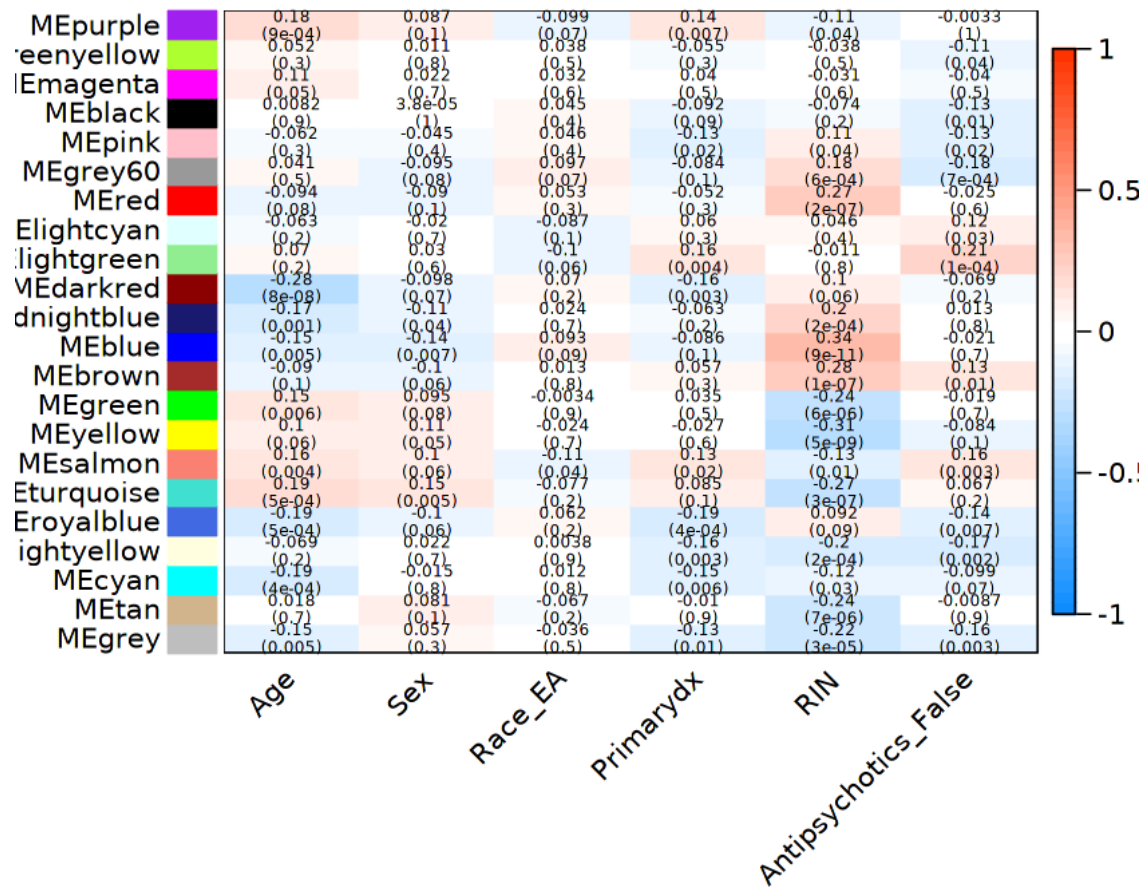
Allowing parallel execution with up to 16 working processes.



```
[16]: #4 - Module Eigenvalue Correlation with sample's traits  
correlate_with_traits()
```

Allowing parallel execution with up to 16 working processes.

## Module kME-Trait Correlation



```
[17]: export_eigengene_tables()
```

## 1.9 Reproducibility Information

```
[18]: Sys.time()
proc.time()
options(width = 120)
sessioninfo::session_info()
```

```
[1] "2021-08-07 10:06:47 EDT"
```

```
      user  system elapsed
9066.619 3673.052  760.040
```

```

Session info
setting  value
version  R version 4.0.3 (2020-10-10)
os       Arch Linux
system   x86_64, linux-gnu
ui       X11
language (EN)
collate  en_US.UTF-8
ctype    en_US.UTF-8
tz       America/New_York
date     2021-08-07

```

```

Packages
package      * version  date      lib source
AnnotationDbi 1.52.0    2020-10-27 [1] Bioconductor
assertthat    0.2.1     2019-03-21 [1] CRAN (R 4.0.2)
backports     1.2.1     2020-12-09 [1] CRAN (R 4.0.2)
base64enc     0.1-3     2015-07-28 [1] CRAN (R 4.0.2)
Biobase       2.50.0    2020-10-27 [1] Bioconductor
BiocGenerics  0.36.1    2021-04-16 [1] Bioconductor
bit           4.0.4     2020-08-04 [1] CRAN (R 4.0.2)
bit64         4.0.5     2020-08-30 [1] CRAN (R 4.0.2)
blob          1.2.1     2020-01-20 [1] CRAN (R 4.0.2)
cachem        1.0.5     2021-05-15 [1] CRAN (R 4.0.3)
Cairo         1.5-12.2  2020-07-07 [1] CRAN (R 4.0.2)
checkmate     2.0.0     2020-02-06 [1] CRAN (R 4.0.2)
cli           3.0.0     2021-06-30 [1] CRAN (R 4.0.3)
cluster       2.1.0     2019-06-19 [2] CRAN (R 4.0.3)
codetools     0.2-16    2018-12-24 [2] CRAN (R 4.0.3)
colorspace    2.0-2     2021-06-24 [1] CRAN (R 4.0.3)
crayon        1.4.1     2021-02-08 [1] CRAN (R 4.0.3)
data.table    1.14.0    2021-02-21 [1] CRAN (R 4.0.3)
DBI           1.1.1     2021-01-15 [1] CRAN (R 4.0.2)
digest        0.6.27    2020-10-24 [1] CRAN (R 4.0.2)
doParallel    1.0.16    2020-10-16 [1] CRAN (R 4.0.3)
dplyr         * 1.0.7     2021-06-18 [1] CRAN (R 4.0.3)
dynamicTreeCut * 1.63-1    2016-03-11 [1] CRAN (R 4.0.3)
ellipsis      0.3.2     2021-04-29 [1] CRAN (R 4.0.3)
evaluate      0.14      2019-05-28 [1] CRAN (R 4.0.2)
fans          0.5.0     2021-05-25 [1] CRAN (R 4.0.3)
fastcluster   * 1.2.3     2021-05-24 [1] CRAN (R 4.0.3)
fastmap       1.1.0     2021-01-25 [1] CRAN (R 4.0.2)
foreach       1.5.1     2020-10-15 [1] CRAN (R 4.0.2)
foreign       0.8-80    2020-05-24 [2] CRAN (R 4.0.3)
Formula       1.2-4     2020-10-16 [1] CRAN (R 4.0.2)
generics      0.1.0     2020-10-31 [1] CRAN (R 4.0.2)
ggplot2       3.3.5     2021-06-25 [1] CRAN (R 4.0.3)
glue          1.4.2     2020-08-27 [1] CRAN (R 4.0.2)

```

GO.db	3.12.1	2021-04-08	[1]	Bioconductor
gridExtra	2.3	2017-09-09	[1]	CRAN (R 4.0.2)
gtable	0.3.0	2019-03-25	[1]	CRAN (R 4.0.2)
Hmisc	4.5-0	2021-02-28	[1]	CRAN (R 4.0.3)
htmlTable	2.2.1	2021-05-18	[1]	CRAN (R 4.0.3)
htmltools	0.5.1.1	2021-01-22	[1]	CRAN (R 4.0.2)
htmlwidgets	1.5.3	2020-12-10	[1]	CRAN (R 4.0.2)
impute	1.64.0	2020-10-27	[1]	Bioconductor
IRanges	2.24.1	2020-12-12	[1]	Bioconductor
IRdisplay	1.0	2021-01-20	[1]	CRAN (R 4.0.2)
IRkernel	1.2	2021-05-11	[1]	CRAN (R 4.0.3)
iterators	1.0.13	2020-10-15	[1]	CRAN (R 4.0.2)
jpeg	0.1-8.1	2019-10-24	[1]	CRAN (R 4.0.2)
jsonlite	1.7.2	2020-12-09	[1]	CRAN (R 4.0.2)
knitr	1.33	2021-04-24	[1]	CRAN (R 4.0.3)
lattice	0.20-41	2020-04-02	[2]	CRAN (R 4.0.3)
latticeExtra	0.6-29	2019-12-19	[1]	CRAN (R 4.0.2)
lifecycle	1.0.0	2021-02-15	[1]	CRAN (R 4.0.3)
limma	* 3.46.0	2020-10-27	[1]	Bioconductor
magrittr	2.0.1	2020-11-17	[1]	CRAN (R 4.0.2)
Matrix	1.3-4	2021-06-01	[1]	CRAN (R 4.0.3)
matrixStats	0.59.0	2021-06-01	[1]	CRAN (R 4.0.3)
memoise	2.0.0	2021-01-26	[1]	CRAN (R 4.0.2)
munsell	0.5.0	2018-06-12	[1]	CRAN (R 4.0.2)
nnet	7.3-14	2020-04-26	[2]	CRAN (R 4.0.3)
pbdZMQ	0.3-5	2021-02-10	[1]	CRAN (R 4.0.3)
pillar	1.6.1	2021-05-16	[1]	CRAN (R 4.0.3)
pkgconfig	2.0.3	2019-09-22	[1]	CRAN (R 4.0.2)
png	0.1-7	2013-12-03	[1]	CRAN (R 4.0.2)
preprocessCore	1.52.1	2021-01-08	[1]	Bioconductor
purrr	0.3.4	2020-04-17	[1]	CRAN (R 4.0.2)
R6	2.5.0	2020-10-28	[1]	CRAN (R 4.0.2)
RColorBrewer	1.1-2	2014-12-07	[1]	CRAN (R 4.0.2)
Rcpp	1.0.7	2021-07-07	[1]	CRAN (R 4.0.3)
repr	1.1.3	2021-01-21	[1]	CRAN (R 4.0.2)
rlang	0.4.11	2021-04-30	[1]	CRAN (R 4.0.3)
rpart	4.1-15	2019-04-12	[2]	CRAN (R 4.0.3)
RSQLite	2.2.7	2021-04-22	[1]	CRAN (R 4.0.3)
rstudioapi	0.13	2020-11-12	[1]	CRAN (R 4.0.2)
S4Vectors	0.28.1	2020-12-09	[1]	Bioconductor
scales	1.1.1	2020-05-11	[1]	CRAN (R 4.0.2)
sessioninfo	1.1.1	2018-11-05	[1]	CRAN (R 4.0.2)
stringi	1.7.3	2021-07-16	[1]	CRAN (R 4.0.3)
stringr	1.4.0	2019-02-10	[1]	CRAN (R 4.0.2)
survival	3.2-7	2020-09-28	[2]	CRAN (R 4.0.3)
tibble	3.1.2	2021-05-16	[1]	CRAN (R 4.0.3)
tidyselect	1.1.1	2021-04-30	[1]	CRAN (R 4.0.3)
utf8	1.2.1	2021-03-12	[1]	CRAN (R 4.0.3)



uuid	0.1-4	2020-02-26	[1]	CRAN	(R 4.0.2)
vctrs	0.3.8	2021-04-29	[1]	CRAN	(R 4.0.3)
WGCNA	* 1.70-3	2021-02-28	[1]	CRAN	(R 4.0.3)
withr	2.4.2	2021-04-18	[1]	CRAN	(R 4.0.3)
xfun	0.24	2021-06-15	[1]	CRAN	(R 4.0.3)

[1] /home/jbenja13/R/x86\_64-pc-linux-gnu-library/4.0

[2] /usr/lib/R/library