Network Inference with WGCNA

February 27, 2019

1 Network analysis of liver expression data in female mice

Tutorial for Module 6 DUBII 2019

Costas Bouyioukos Universite Paris Diderot and Anais Baudot CNRS

1.1 1. Preliminaries and data input

In [56]:
```
#  Code chunk 1
# Display the current working directory
getwd();
# If necessary, change the path below to the directory where the data files are stored
# "." means current directory.
workingDir = ".";
setwd(workingDir);
# Load the WGCNA package
library(WGCNA);
# The following setting is important, do not omit.
options(stringsAsFactors = FALSE);
#Read in the female liver data set
femData = read.csv("LiverFemale3600.csv");
# Take a quick look at what is in the data set:
dim(femData);
names(femData);
head(femData);
```

'/home/costas/Documents/teaching/diderot/dubii/dubii_m6_s5_WGCNAtutorial_2019'
1. 3600 2. 143
1. 'substanceBXH' 2. 'gene_symbol' 3. 'LocusLinkID' 4. 'ProteomeID' 5. 'cytogeneticLoc' 6. 'CHROMOSOME' 7. 'StartPosition' 8. 'EndPosition' 9. 'F2_2' 10. 'F2_3' 11. 'F2_14' 12. 'F2_15' 13. 'F2_19' 14. 'F2_20' 15. 'F2_23' 16. 'F2_24' 17. 'F2_26' 18. 'F2_37' 19. 'F2_42' 20. 'F2_43' 21. 'F2_45' 22. 'F2_46' 23. 'F2_47' 24. 'F2_48' 25. 'F2_51' 26. 'F2_52' 27. 'F2_54' 28. 'F2_63' 29. 'F2_65' 30. 'F2_66' 31. 'F2_68' 32. 'F2_69' 33. 'F2_70' 34. 'F2_71' 35. 'F2_72' 36. 'F2_78' 37. 'F2_79' 38. 'F2_80' 39. 'F2_81' 40. 'F2_83' 41. 'F2_86' 42. 'F2_87' 43. 'F2_88' 44. 'F2_89' 45. 'F2_107' 46. 'F2_108' 47. 'F2_109' 48. 'F2_110' 49. 'F2_111' 50. 'F2_112' 51. 'F2_117' 52. 'F2_119' 53. 'F2_125' 54. 'F2_126' 55. 'F2_127' 56. 'F2_141' 57. 'F2_142' 58. 'F2_143' 59. 'F2_144' 60. 'F2_145' 61. 'F2_154' 62. 'F2_155' 63. 'F2_156' 64. 'F2_157' 65. 'F2_162' 66. 'F2_163' 67. 'F2_164' 68. 'F2_165' 69. 'F2_166' 70. 'F2_167' 71. 'F2_169'

72. 'F2_180' 73. 'F2_181' 74. 'F2_182' 75. 'F2_187' 76. 'F2_188' 77. 'F2_189' 78. 'F2_190' 79. 'F2_191' 80. 'F2_192' 81. 'F2_194' 82. 'F2_195' 83. 'F2_200' 84. 'F2_201' 85. 'F2_212' 86. 'F2_213' 87. 'F2_214' 88. 'F2_215' 89. 'F2_221' 90. 'F2_222' 91. 'F2_223' 92. 'F2_224' 93. 'F2_225' 94. 'F2_226' 95. 'F2_227' 96. 'F2_228' 97. 'F2_241' 98. 'F2_242' 99. 'F2_243' 100. 'F2_244' 101. 'F2_245' 102. 'F2_247' 103. 'F2_248' 104. 'F2_261' 105. 'F2_263' 106. 'F2_264' 107. 'F2_270' 108. 'F2_271' 109. 'F2_272' 110. 'F2_278' 111. 'F2_287' 112. 'F2_288' 113. 'F2_289' 114. 'F2_290' 115. 'F2_291' 116. 'F2_296' 117. 'F2_298' 118. 'F2_299' 119. 'F2_300' 120. 'F2_302' 121. 'F2_303' 122. 'F2_304' 123. 'F2_305' 124. 'F2_306' 125. 'F2_307' 126. 'F2_308' 127. 'F2_309' 128. 'F2_310' 129. 'F2_311' 130. 'F2_312' 131. 'F2_320' 132. 'F2_321' 133. 'F2_323' 134. 'F2_324' 135. 'F2_325' 136. 'F2_326' 137. 'F2_327' 138. 'F2_328' 139. 'F2_329' 140. 'F2_330' 141. 'F2_332' 142. 'F2_355' 143. 'F2_357'

substanceBXH	gene_symbol	LocusLinkID	ProteomeID	cytogeneticLoc	CHROMOSOME	Start
MMT00000044	1700007N18Rik	69339	286025	0	16	5091
MMT00000046	Mast2	17776	157466	0	4	1152
MMT00000051	Ankrd32	105377	321939	0	13	7494
MMT00000076	0	383154	0	0	16	4934
MMT00000080	Ldb2	16826	157383	0	5	4354
MMT00000102	Rdhs	216453	0	10_70.0_cM	10	1337

Keep only the part of the data that contains the gene expression and keep the gene names as data frame index

```
In [27]: #  Code chunk 2
         datExpr0 = as.data.frame(t(femData[, -c(1:8)]));
         names(datExpr0) = femData$substanceBXH;
         rownames(datExpr0) = names(femData)[-c(1:8)];
```

Check if there are genes with missing values.

```
In [28]: #  Code chunk 3
         gsg = goodSamplesGenes(datExpr0, verbose = 3);
         gsg$allOK
```

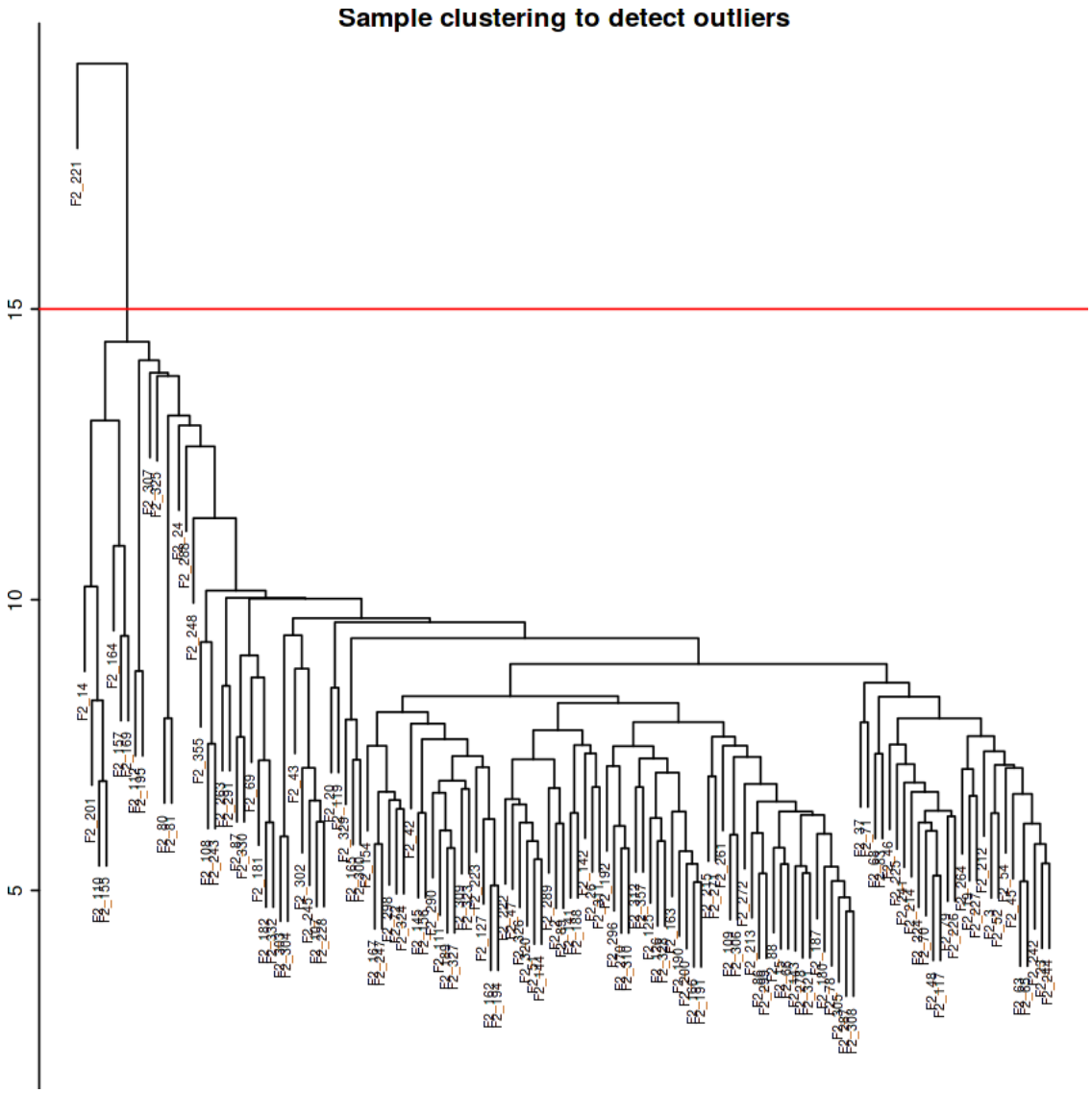
```
 Flagging genes and samples with too many missing values...
  ..step 1
```

TRUE

All genes are OK.

Cluster the transposed matrix to identify sample outliers.

```
In [69]: #  Code chunk 4
         sampleTree = hclust(dist(datExpr0), method = "average");
         # Plot the sample tree: Open a graphic output window of size 12 by 9 inches
         # The user should change the dimensions if the window is too large or too small.
         par(cex = 0.5);
         par(mar = c(0,2,1,0))
         plot(sampleTree, main = "Sample clustering to detect outliers", sub="", xlab="",
              cex.lab = 1.5, cex.axis = 1.5, cex.main = 2)
         # Plot a line to show the cut
         abline(h = 15, col = "red");
```



Identify the outlier.

```
In [70]: #  Code chunk 5
         # Determine cluster under the line
         clust = cutreeStatic(sampleTree, cutHeight = 15, minSize = 10)
         table(clust)
```

```
clust
  0   1
  1 134
```

Remove the outlier and construct the main data frame.

```
In [ ]: #  Code chunk 5
        # clust 1 contains the samples we want to keep.
        keepSamples = (clust==1)
        datExpr = datExpr0[keepSamples, ]
        nGenes = ncol(datExpr)
        nSamples = nrow(datExpr)
```

Introduce the clinical data, preapre and clean it.

```
In [71]: #  Code chunk 7
         traitData = read.csv("ClinicalTraits.csv");
         dim(traitData)
         names(traitData)
         # remove columns that hold information we do not need.
         allTraits = traitData[, -c(31, 16)];
         allTraits = allTraits[, c(2, 11:36) ];
         dim(allTraits)
         names(allTraits)
         # Form a data frame analogous to expression data that will hold the clinical traits.
         femaleSamples = rownames(datExpr);
         traitRows = match(femaleSamples, allTraits$Mice);
         datTraits = allTraits[traitRows, -1];
         rownames(datTraits) = allTraits[traitRows, 1];
```

1. 361 2. 38

1. 'X' 2. 'Mice' 3. 'Number' 4. 'Mouse_ID' 5. 'Strain' 6. 'sex' 7. 'DOB' 8. 'parents' 9. 'Western_Diet' 10. 'Sac_Date' 11. 'weight_g' 12. 'length_cm' 13. 'ab_fat' 14. 'other_fat' 15. 'total_fat' 16. 'comments' 17. 'X100xfat_weight' 18. 'Trigly' 19. 'Total_Chol' 20. 'HDL_Chol' 21. 'UC' 22. 'FFA' 23. 'Glucose' 24. 'LDL_plus_VLDL' 25. 'MCP_1_phys' 26. 'Insulin_ug_l' 27. 'Glucose_Insulin' 28. 'Leptin_pg_ml' 29. 'Adiponectin' 30. 'Aortic.lesions' 31. 'Note' 32. 'Aneurysm' 33. 'Aortic_cal_M' 34. 'Aortic_cal_L' 35. 'CoronaryArtery_Cal' 36. 'Myocardial_cal' 37. 'BMD_all_limbs' 38. 'BMD_femurs_only'

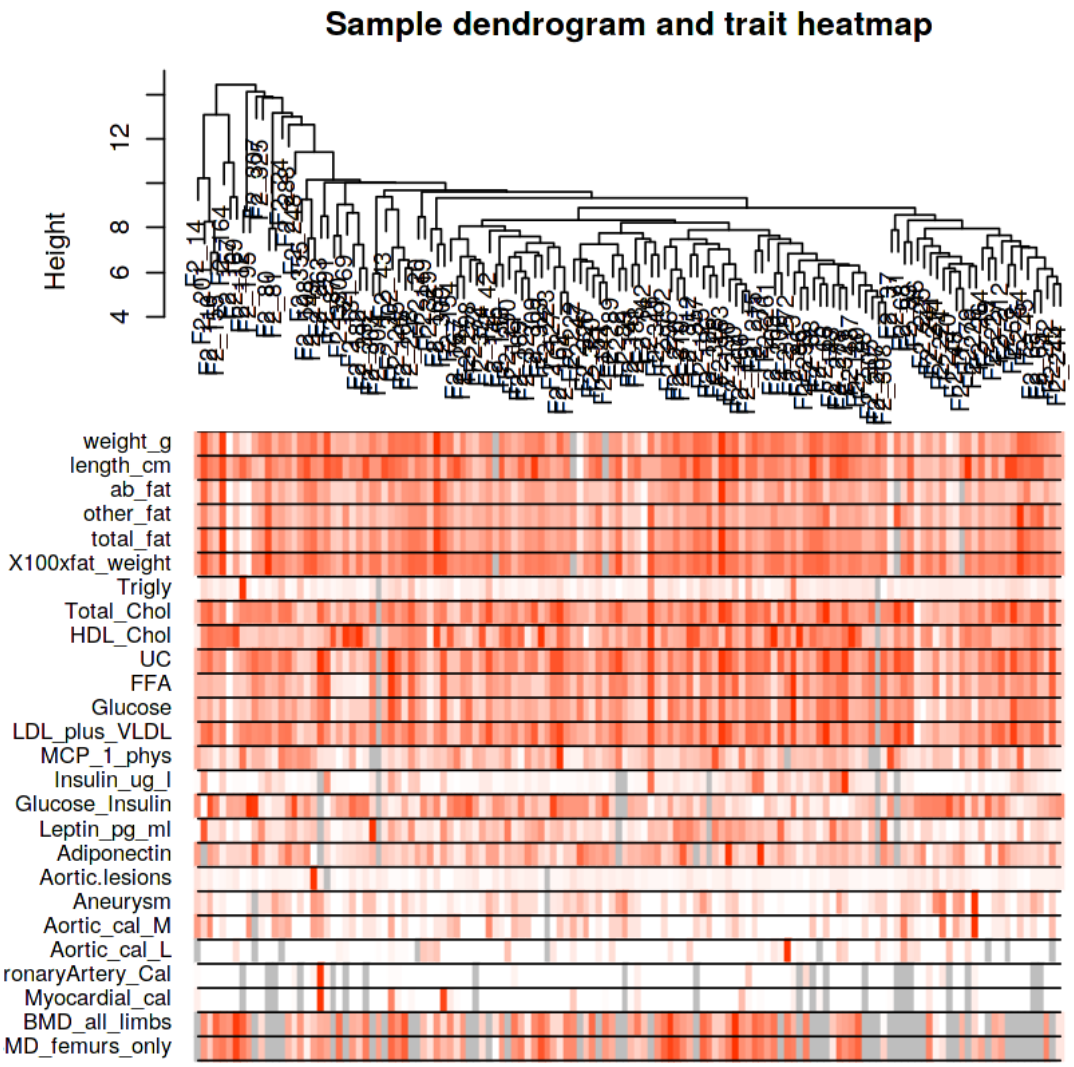
1. 361 2. 27

1. 'Mice' 2. 'weight_g' 3. 'length_cm' 4. 'ab_fat' 5. 'other_fat' 6. 'total_fat' 7. 'X100xfat_weight' 8. 'Trigly' 9. 'Total_Chol' 10. 'HDL_Chol' 11. 'UC' 12. 'FFA' 13. 'Glucose' 14. 'LDL_plus_VLDL' 15. 'MCP_1_phys' 16. 'Insulin_ug_l' 17. 'Glucose_Insulin' 18. 'Leptin_pg_ml' 19. 'Adiponectin' 20. 'Aortic.lesions' 21. 'Aneurysm' 22. 'Aortic_cal_M' 23. 'Aortic_cal_L' 24. 'CoronaryArtery_Cal' 25. 'Myocardial_cal' 26. 'BMD_all_limbs' 27. 'BMD_femurs_only'

Repeat the sample clustering together with a heat map of the phenotypic data.

```
In [72]: #  Code chunk 8
         # Re-cluster samples
         sampleTree2 = hclust(dist(datExpr), method = "average")
         # Convert traits to a color representation: white means low, red means high, grey mea
         traitColors = numbers2colors(datTraits, signed = FALSE);
         # Plot the sample dendrogram and the colors underneath.
```

```
plotDendroAndColors(sampleTree2, traitColors,
                    groupLabels = names(datTraits),
                    main = "Sample dendrogram and trait heatmap")
```



Save the analysis to an RData file.

```
In [73]: # Code chunk 9
         save(datExpr, datTraits, file = "FemaleLiver-01-dataInput.RData")
```

1.2 2. Automatic network construction and module detection

```
In [74]: #  Code chunk 10
         # Allow multi-threading within WGCNA. This helps speed up certain calculations.
         # At present this call is necessary for the code to work.
```

```
        # Any error here may be ignored but you may want to update WGCNA if you see one.
        # See note above.
        allowWGCNAThreads()
        # Load the data saved in the first part
        lnames = load(file = "FemaleLiver-01-dataInput.RData");
        #The variable lnames contains the names of loaded variables.
        lnames
```

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

1. 'datExpr' 2. 'datTraits'

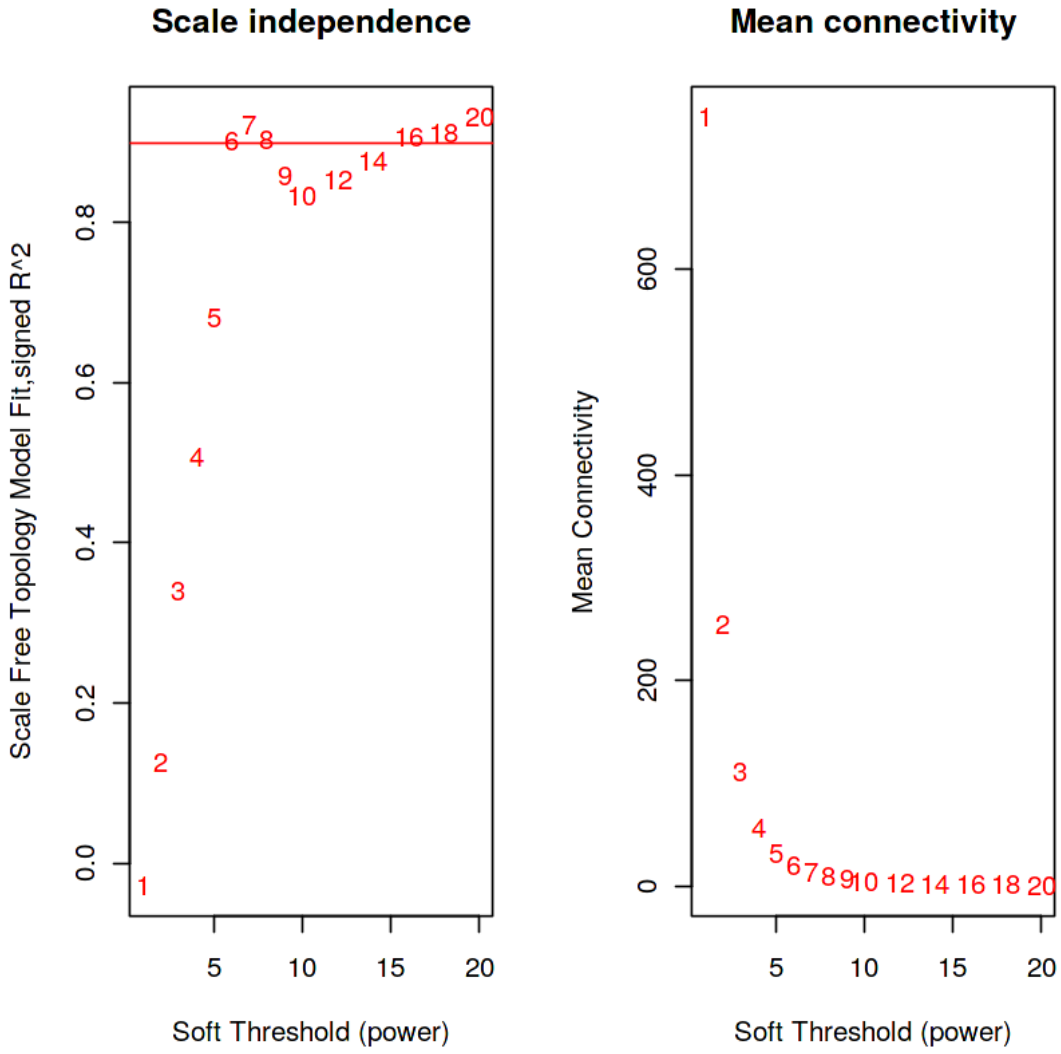
This is the most convenient and automatic way to detect modules and construct a network with WGCNA.

Here the developers of WGCNA are proposing a "soft thresholding" approach. This method identifies a power -to wich the correlation matrix is raised in order to calculate the network adjacency matrix- based on the criterion of scale-free approximation.

```
In [79]: #  Code chunk 11
         # Choose a set of soft-thresholding powers
         powers = c(c(1:10), seq(from = 12, to=20, by=2))
         # Call the network topology analysis function
         sft = pickSoftThreshold(datExpr, powerVector = powers, verbose = 5)
         # Plot the results:
         par(mfrow = c(1,2));
         # Scale-free topology fit index as a function of the soft-thresholding power
         plot(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
              xlab="Soft Threshold (power)",ylab="Scale Free Topology Model Fit,signed R^2",ty
              main = paste("Scale independence"));
         text(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
              labels=powers,cex=cex1,col="red");
         # this line corresponds to using an R^2 cut-off of h
         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 = paste("Mean connectivity"))
         text(sft$fitIndices[,1], sft$fitIndices[,5], labels=powers, cex=cex1,col="red")
```

```
pickSoftThreshold: will use block size 3600.
 pickSoftThreshold: calculating connectivity for given powers...
   ..working on genes 1 through 3600 of 3600
   Power SFT.R.sq  slope truncated.R.sq mean.k. median.k. max.k.
1      1   0.0278  0.345          0.456  747.00  762.0000 1210.0
2      2   0.1260 -0.597          0.843  254.00  251.0000  574.0
3      3   0.3400 -1.030          0.972  111.00  102.0000  324.0
4      4   0.5060 -1.420          0.973   56.50   47.2000  202.0
5      5   0.6810 -1.720          0.940   32.20   25.1000  134.0
6      6   0.9020 -1.500          0.962   19.90   14.5000   94.8
```

```
7      7   0.9210 -1.670          0.917   13.20    8.6800   84.1
8      8   0.9040 -1.720          0.876    9.25    5.3900   76.3
9      9   0.8590 -1.700          0.836    6.80    3.5600   70.5
10    10   0.8330 -1.660          0.831    5.19    2.3800   65.8
11    12   0.8530 -1.480          0.911    3.33    1.1500   58.1
12    14   0.8760 -1.380          0.949    2.35    0.5740   51.9
13    16   0.9070 -1.300          0.970    1.77    0.3090   46.8
14    18   0.9120 -1.240          0.973    1.39    0.1670   42.5
15    20   0.9310 -1.210          0.977    1.14    0.0951   38.7
```



Thisis the actual network construction step.

We choose 6 as the lowest power that constructs a scale free topology. And then we instruct the function to generate modules of size 30, merge modules which are more than 25% similar and

save the Topological Overlap Matrix in an object.

```
In [81]: #  Code chunk 12
         net = blockwiseModules(datExpr, power = 6,
                                TOMType = "unsigned", minModuleSize = 30,
                                reassignThreshold = 0, mergeCutHeight = 0.25,
                                numericLabels = TRUE, pamRespectsDendro = FALSE,
                                saveTOMs = TRUE,
                                saveTOMFileBase = "femaleMouseTOM",
                                verbose = 3)
```

```
 Calculating module eigengenes block-wise from all genes
   Flagging genes and samples with too many missing values...
    ..step 1
Cluster size 3600 broken into 2133 1467
Cluster size 2133 broken into 1221 912
Done cluster 1221
Done cluster 912
Done cluster 2133
Done cluster 1467
 ..Working on block 1 .
    TOM calculation: adjacency..
    ..will use 8 parallel threads.
     Fraction of slow calculations: 0.362314
    ..connectivity..
    ..matrix multiplication (system BLAS)..
    ..normalization..
    ..done.
   ..saving TOM for block 1 into file femaleMouseTOM-block.1.RData
 ...clustering..
 ...detecting modules..
 ...calculating module eigengenes..
 ...checking kME in modules..
     ..removing 1 genes from module 1 because their KME is too low.
     ..removing 1 genes from module 7 because their KME is too low.
     ..removing 1 genes from module 8 because their KME is too low.
     ..removing 1 genes from module 21 because their KME is too low.
 ..merging modules that are too close..
     mergeCloseModules: Merging modules whose distance is less than 0.25
       Calculating new MEs...
```

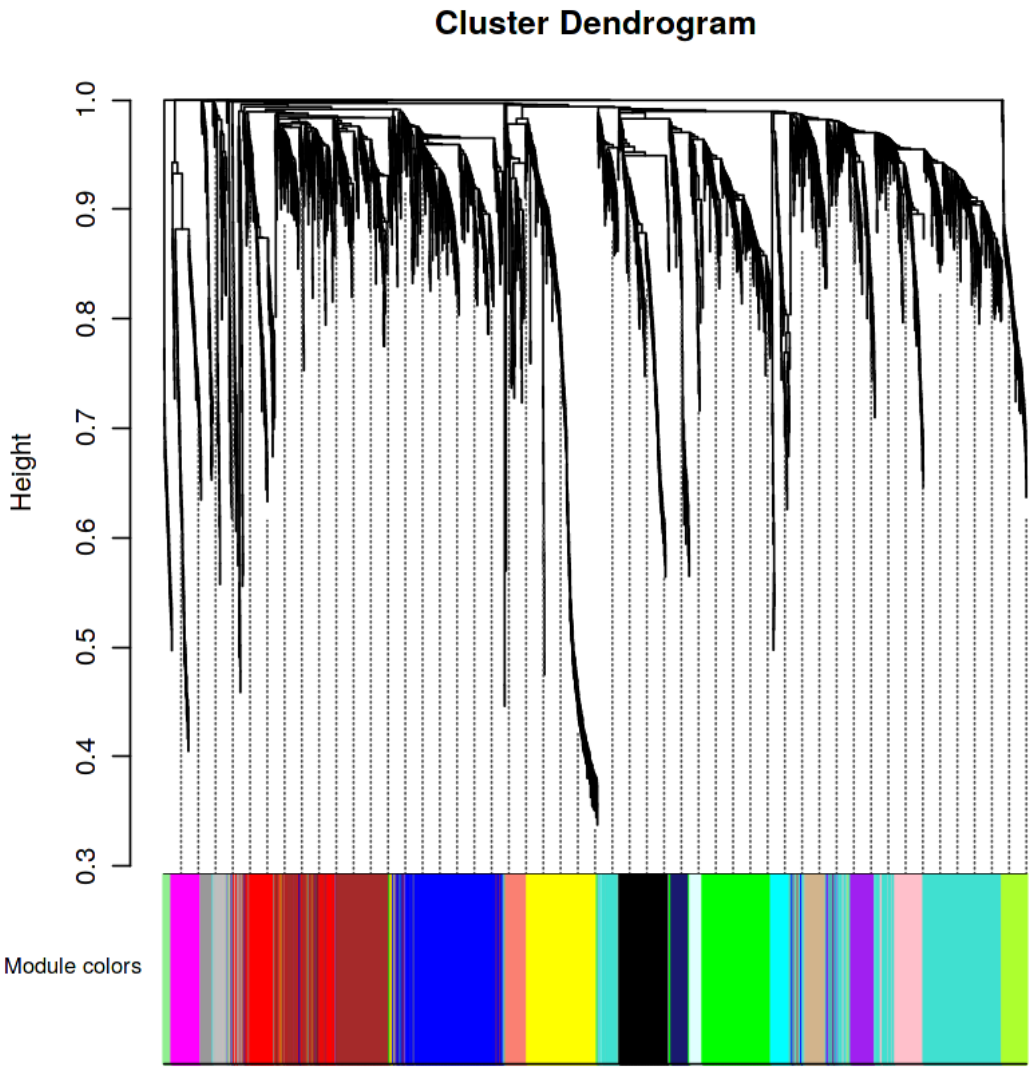
Here is the modules (as numbers and not colours yet) of each module with its size.

```
In [82]: table(net$colors)
```

```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
 99 609 460 409 316 312 221 211 157 123 106 100  94  91  77  76  58  47  34
```

Here is the resuling plot dendrogram of the module construction and the clustering of the genes

```
In [83]: #  Code chunk 13
         # Convert labels to colors for plotting
         mergedColors = labels2colors(net$colors)
         # Plot the dendrogram and the module colors underneath
         plotDendroAndColors(net$dendrograms[[1]], mergedColors[net$blockGenes[[1]]],
                             "Module colors",
                             dendroLabels = FALSE, hang = 0.03,
                             addGuide = TRUE, guideHang = 0.05)
```



Save some results of this part in an .RData file.

```
In [84]: #  Code chunk 14
         moduleLabels = net$colors
         moduleColors = labels2colors(net$colors)
         MEs = net$MEs;
         geneTree = net$dendrograms[[1]];
         save(MEs, moduleLabels, moduleColors, geneTree,
              file = "FemaleLiver-02-networkConstruction-auto.RData")
```

1.3 3. Relating modules to external information and identifying important genes

```
In [85]: #  Code chunk 15
         lnames = load(file = "FemaleLiver-01-dataInput.RData");
         #The variable lnames contains the names of loaded variables.
         lnames
         # Load network data saved in the second part.
         lnames = load(file = "FemaleLiver-02-networkConstruction-auto.RData");
         lnames
```

1. 'datExpr' 2. 'datTraits'
1. 'MEs' 2. 'moduleLabels' 3. 'moduleColors' 4. 'geneTree'

Quantifying module–trait associations Here we identify modules that are significantly associated with the measured clinical traits. We already have a computed summary profile (eigengene) for each module, so then we simply correlate eigengenes with phenotypic traits and look for the most significant associations:

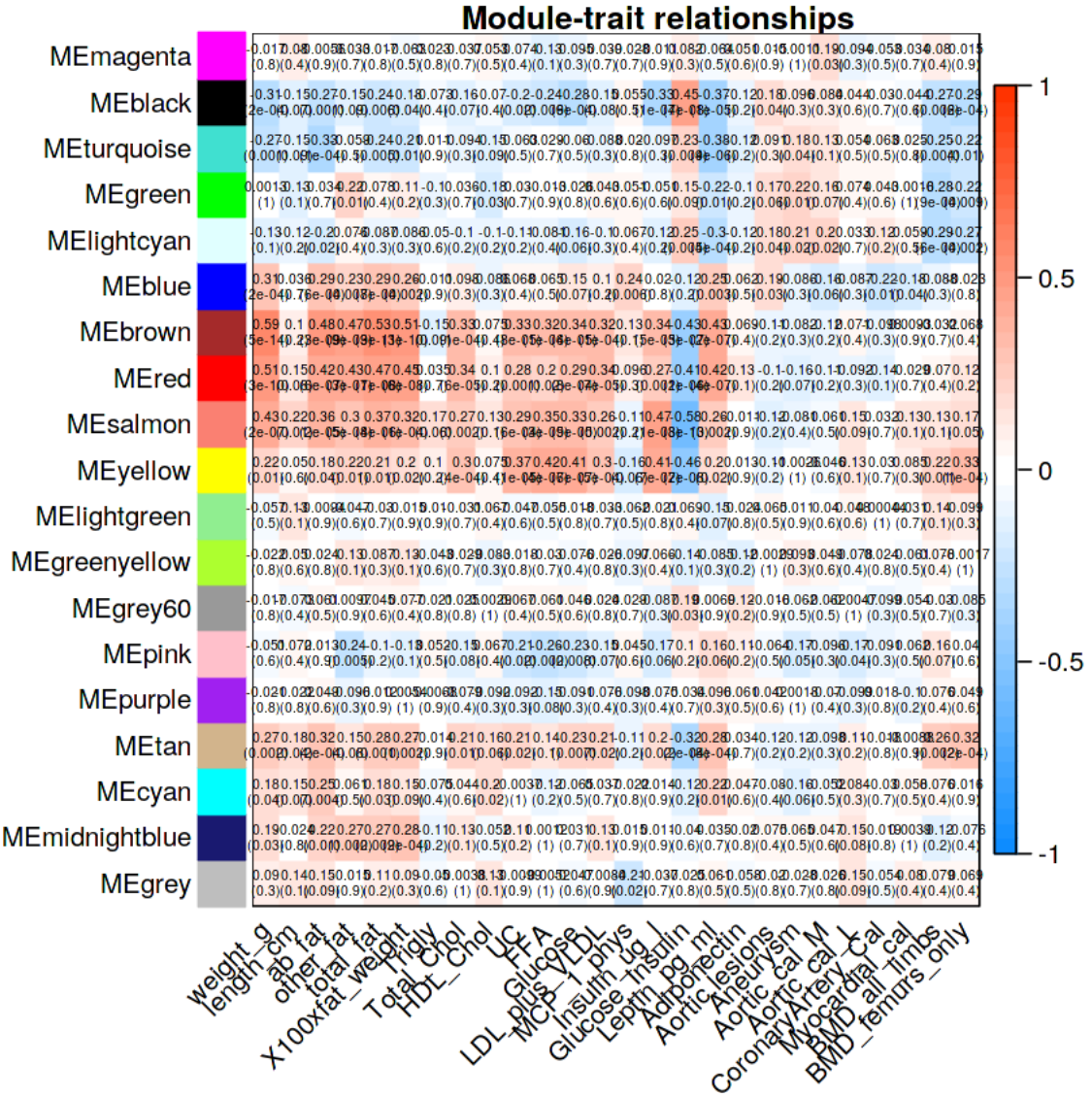
```
In [86]: #  Code chunk 16
         # Define numbers of genes and samples
         nGenes = ncol(datExpr);
         nSamples = nrow(datExpr);
         # Recalculate MEs with color labels
         MEs0 = moduleEigengenes(datExpr, moduleColors)$eigengenes
         MEs = orderMEs(MEs0)
         moduleTraitCor = cor(MEs, datTraits, use = "p");
         moduleTraitPvalue = corPvalueStudent(moduleTraitCor, nSamples);
```

Vsualise the module-trait association.

Each module eigengene and its correlation coefficient are ploted here. Since we have many a colour code aids the interprettion of the plot.

```
In [94]: #  Code chunk 17
         # 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, 1, 1));
         # Display the correlation values within a heatmap plot
         labeledHeatmap(Matrix = moduleTraitCor,
```

```
    xLabels = names(datTraits),
    yLabels = names(MEs),
    ySymbols = names(MEs),
    colorLabels = FALSE,
    colors = blueWhiteRed(50),
    textMatrix = textMatrix,
    setStdMargins = FALSE,
    cex.text = 0.5,
    zlim = c(-1,1),
    main = paste("Module-trait relationships"))
```



In [95]:
```
#  Code chunk 18
names(datExpr)[moduleColors=="magenta"]
```

1. 'MMT00000713' 2. 'MMT00001923' 3. 'MMT00003127' 4. 'MMT00003545' 5. 'MMT00003906' 6. 'MMT00003968' 7. 'MMT00004393' 8. 'MMT00004398' 9. 'MMT00004682' 10. 'MMT00005626' 11. 'MMT00006811' 12. 'MMT00008299' 13. 'MMT00008438' 14. 'MMT00010131' 15. 'MMT00012112' 16. 'MMT00012971' 17. 'MMT00013135' 18. 'MMT00014249' 19. 'MMT00014332' 20. 'MMT00014842' 21. 'MMT00015100' 22. 'MMT00015204' 23. 'MMT00017461' 24. 'MMT00018499' 25. 'MMT00018534' 26. 'MMT00018540' 27. 'MMT00020013' 28. 'MMT00020720' 29. 'MMT00021045' 30. 'MMT00021101' 31. 'MMT00022128' 32. 'MMT00022673' 33. 'MMT00022995' 34. 'MMT00023687' 35. 'MMT00023914' 36. 'MMT00024126' 37. 'MMT00024589' 38. 'MMT00024738' 39. 'MMT00025504' 40. 'MMT00025626' 41. 'MMT00027274' 42. 'MMT00027714' 43. 'MMT00027815' 44. 'MMT00027926' 45. 'MMT00028099' 46. 'MMT00028707' 47. 'MMT00029304' 48. 'MMT00029635' 49. 'MMT00030667' 50. 'MMT00031068' 51. 'MMT00031302' 52. 'MMT00031885' 53. 'MMT00032473' 54. 'MMT00033646' 55. 'MMT00034196' 56. 'MMT00034201' 57. 'MMT00036034' 58. 'MMT00036455' 59. 'MMT00037178' 60. 'MMT00037383' 61. 'MMT00038053' 62. 'MMT00038527' 63. 'MMT00040568' 64. 'MMT00040689' 65. 'MMT00040748' 66. 'MMT00041058' 67. 'MMT00041075' 68. 'MMT00043550' 69. 'MMT00044531' 70. 'MMT00045141' 71. 'MMT00045235' 72. 'MMT00046187' 73. 'MMT00046189' 74. 'MMT00046479' 75. 'MMT00048831' 76. 'MMT00050140' 77. 'MMT00050573' 78. 'MMT00050853' 79. 'MMT00050980' 80. 'MMT00051830' 81. 'MMT00051969' 82. 'MMT00052908' 83. 'MMT00053346' 84. 'MMT00053348' 85. 'MMT00053622' 86. 'MMT00053926' 87. 'MMT00054654' 88. 'MMT00056406' 89. 'MMT00056840' 90. 'MMT00057948' 91. 'MMT00058392' 92. 'MMT00058742' 93. 'MMT00059409' 94. 'MMT00059963' 95. 'MMT00060778' 96. 'MMT00061056' 97. 'MMT00061252' 98. 'MMT00063424' 99. 'MMT00064449' 100. 'MMT00065504' 101. 'MMT00065851' 102. 'MMT00067476' 103. 'MMT00067843' 104. 'MMT00068925' 105. 'MMT00069033' 106. 'MMT00069333' 107. 'MMT00070149' 108. 'MMT00070332' 109. 'MMT00070554' 110. 'MMT00071034' 111. 'MMT00071318' 112. 'MMT00072448' 113. 'MMT00072685' 114. 'MMT00074187' 115. 'MMT00074989' 116. 'MMT00075756' 117. 'MMT00075914' 118. 'MMT00076573' 119. 'MMT00078097' 120. 'MMT00078506' 121. 'MMT00078931' 122. 'MMT00080518' 123. 'MMT00081919'

Probe nnotation file provided by the manufacturer to facilitate functional annotation.

```
In [96]: #  Code chunk 19
         annot = read.csv(file = "GeneAnnotation.csv");
         dim(annot)
         names(annot)
         probes = names(datExpr)
         probes2annot = match(probes, annot$substanceBXH)
         # The following is the number or probes without annotation:
         sum(is.na(probes2annot))
         # Should return 0.
```

1. 23388 2. 34

1. 'X' 2. 'ID' 3. 'arrayname' 4. 'substanceBXH' 5. 'gene_symbol' 6. 'LocusLinkID' 7. 'OfficialGeneSymbol' 8. 'OfficialGeneName' 9. 'LocusLinkSymbol' 10. 'LocusLinkName' 11. 'ProteomeShortDescription' 12. 'UnigeneCluster' 13. 'LocusLinkCode' 14. 'ProteomeID' 15. 'ProteomeCode'

16. 'SwissprotID' 17. 'OMIMCode' 18. 'DirectedTilingPriority' 19. 'AlternateSymbols' 20. 'AlternateNames' 21. 'SpeciesID' 22. 'cytogeneticLoc' 23. 'Organism' 24. 'clustername' 25. 'reporterid' 26. 'probeid' 27. 'sequenceid' 28. 'clusterid' 29. 'chromosome' 30. 'startcoordinate' 31. 'endcoordinate' 32. 'strand' 33. 'sequence_3_to_5_prime' 34. 'sequence_5_to_3_prime'

0

Collect all the information for significant genes related to body weight.

```
In [97]: #  Code chunk 20
         # Create the starting data frame
         geneInfo0 = data.frame(substanceBXH = probes,
                                geneSymbol = annot$gene_symbol[probes2annot],
                                LocusLinkID = annot$LocusLinkID[probes2annot],
                                moduleColor = moduleColors,
                                geneTraitSignificance,
                                GSPvalue)
         # Order modules by their significance for weight
         modOrder = order(-abs(cor(MEs, weight, use = "p")));
         # Add module membership information in the chosen order
         for (mod in 1:ncol(geneModuleMembership))
         {
           oldNames = names(geneInfo0)
           geneInfo0 = data.frame(geneInfo0, geneModuleMembership[, modOrder[mod]],
                                  MMPvalue[, modOrder[mod]]);
           names(geneInfo0) = c(oldNames, paste("MM.", modNames[modOrder[mod]], sep=""),
                                paste("p.MM.", modNames[modOrder[mod]], sep=""))
         }
         # Order the genes in the geneInfo variable first by module color, then by geneTraitSi
         geneOrder = order(geneInfo0$moduleColor, -abs(geneInfo0$GS.weight));
         geneInfo = geneInfo0[geneOrder, ]
```

Save the results in an output file for further analysis.

```
In [34]: #  Code chunk 21
         write.csv(geneInfo, file = "geneInfo.csv")
```

1.4 4. Interfacing network analysis with other data such as functional annotation and gene ontology

```
In [98]: #  Code chunk 22
         # Load the expression and trait data saved in the first part
         lnames = load(file = "FemaleLiver-01-dataInput.RData");
         #The variable lnames contains the names of loaded variables.
         lnames
         # Load network data saved in the second part.
         lnames = load(file = "FemaleLiver-02-networkConstruction-auto.RData");
         lnames
```

1. 'datExpr' 2. 'datTraits'

1. 'MEs' 2. 'moduleLabels' 3. 'moduleColors' 4. 'geneTree'

In [36]:
```
#  Code chunk 23
# Read in the probe annotation
annot = read.csv(file = "GeneAnnotation.csv");
# Match probes in the data set to the probe IDs in the annotation file
probes = names(datExpr)
probes2annot = match(probes, annot$substanceBXH)
# Get the corresponding Locuis Link IDs
allLLIDs = annot$LocusLinkID[probes2annot];
# $ Choose interesting modules
intModules = c("brown", "red", "salmon")
for (module in intModules)
{
  # Select module probes
  modGenes = (moduleColors==module)
  # Get their entrez ID codes
  modLLIDs = allLLIDs[modGenes];
  # Write them into a file
  fileName = paste("LocusLinkIDs-", module, ".txt", sep="");
  write.table(as.data.frame(modLLIDs), file = fileName,
              row.names = FALSE, col.names = FALSE)
}
# As background in the enrichment analysis, we will use all probes in the analysis.
fileName = paste("LocusLinkIDs-all.txt", sep="");
write.table(as.data.frame(allLLIDs), file = fileName,
            row.names = FALSE, col.names = FALSE)
```

In [38]:
```
#  Code chunk 24
GOenr = GOenrichmentAnalysis(moduleColors, allLLIDs, organism = "mouse", nBestP = 10)
```

```
Warning message in GOenrichmentAnalysis(moduleColors, allLLIDs, organism = "mouse", :
This function is deprecated and will be removed in the near future.
We suggest using the replacement function enrichmentAnalysis
in R package anRichment, available from the following URL:
https://labs.genetics.ucla.edu/horvath/htdocs/CoexpressionNetwork/GeneAnnotation/

 GOenrichmentAnalysis: loading annotation data...
  ..of the 3038  Entrez identifiers submitted, 2829 are mapped in current GO categories.
  ..will use 2829 background genes for enrichment calculations.
  ..preparing term lists (this may take a while)..
  ..working on label set 1 ..
    ..calculating enrichments (this may also take a while)..
    ..putting together terms with highest enrichment significance..
```

In [50]:
```
#anRichment(moduleColors, allLLIDs, organism = "mouse", nBestP = 10);  # Does not wor
```

```
In [39]: #  Code chunk 25
         tab = GOenr$bestPTerms[[4]]$enrichment

In [40]: #  Code chunk 26
         names(tab)
```

1. 'module' 2. 'modSize' 3. 'bkgrModSize' 4. 'rank' 5. 'enrichmentP' 6. 'BonferoniP' 7. 'nModGenesInTerm' 8. 'fracOfBkgrModSize' 9. 'fracOfBkgrTermSize' 10. 'bkgrTermSize' 11. 'termID' 12. 'termOntology' 13. 'termName' 14. 'termDefinition'

```
In [41]: #  Code chunk 27
         write.table(tab, file = "GOEnrichmentTable.csv", sep = ",", quote = TRUE, row.names =

In [42]: #  Code chunk 28
         keepCols = c(1, 2, 5, 6, 7, 12, 13);
         screenTab = tab[, keepCols];
         # Round the numeric columns to 2 decimal places:
         numCols = c(3, 4);
         screenTab[, numCols] = signif(apply(screenTab[, numCols], 2, as.numeric), 2)
         # Truncate the the term name to at most 40 characters
         screenTab[, 7] = substring(screenTab[, 7], 1, 40)
         # Shorten the column names:
         colnames(screenTab) = c("module", "size", "p-val", "Bonf", "nInTerm", "ont", "term nam
         rownames(screenTab) = NULL;
         # Set the width of R's output. The reader should play with this number to obtain sati
         options(width=95)
         # Finally, display the enrichment table:
         screenTab
```

module	size	p-val	Bonf	nInTerm	ont	term name
black	166	3.9e-04	1.0e+00	4	BP	dopamine transport
black	166	6.5e-04	1.0e+00	5	BP	mRNA transport
black	166	8.1e-04	1.0e+00	13	MF	receptor ligand activity
black	166	9.9e-04	1.0e+00	13	MF	receptor regulator activity
black	166	1.0e-03	1.0e+00	6	BP	RNA transport
black	166	1.3e-03	1.0e+00	6	BP	RNA localization
black	166	1.6e-03	1.0e+00	6	BP	amine transport
black	166	2.4e-03	1.0e+00	6	MF	growth factor activity
black	166	2.6e-03	1.0e+00	2	BP	ventricular compact myocardium morphogen
black	166	2.6e-03	1.0e+00	2	BP	detection of chemical stimulus involved
blue	428	3.2e-33	5.7e-29	166	BP	immune system process
blue	428	3.8e-32	6.9e-28	121	BP	immune response
blue	428	3.6e-23	6.4e-19	109	BP	defense response
blue	428	1.5e-22	2.7e-18	73	BP	innate immune response
blue	428	2.2e-22	4.0e-18	101	BP	regulation of immune system process
blue	428	8.9e-22	1.6e-17	82	BP	positive regulation of immune system pro
blue	428	1.4e-21	2.5e-17	85	BP	cell activation
blue	428	1.9e-18	3.3e-14	71	BP	cytokine production
blue	428	1.4e-17	2.6e-13	64	BP	regulation of cytokine production
blue	428	1.6e-17	2.8e-13	65	BP	regulation of immune response
brown	396	7.7e-22	1.4e-17	46	CC	extracellular matrix
brown	396	2.6e-19	4.7e-15	125	CC	extracellular region
brown	396	5.9e-15	1.1e-10	30	CC	collagen-containing extracellular matrix
brown	396	1.4e-13	2.6e-09	91	CC	extracellular space
brown	396	1.2e-12	2.2e-08	57	BP	blood vessel development
brown	396	1.8e-12	3.3e-08	58	BP	vasculature development
brown	396	2.0e-12	3.6e-08	59	BP	cardiovascular system development
brown	396	6.3e-12	1.1e-07	29	BP	extracellular matrix organization
brown	396	3.3e-11	5.9e-07	49	BP	blood vessel morphogenesis
brown	396	7.9e-11	1.4e-06	69	BP	circulatory system development
tan	81	5.9e-04	1	13	CC	catalytic complex
tan	81	7.9e-04	1	37	MF	catalytic activity
tan	81	1.2e-03	1	3	CC	COPII-coated ER to Golgi transport vesic
tan	81	1.9e-03	1	2	BP	anterograde synaptic vesicle transport
tan	81	1.9e-03	1	2	BP	endoplasmic reticulum tubular network or
tan	81	3.8e-03	1	2	CC	axon cytoplasm
tan	81	4.6e-03	1	4	BP	cellular response to xenobiotic stimulus
tan	81	4.7e-03	1	14	CC	Golgi apparatus
tan	81	5.2e-03	1	4	BP	drug catabolic process
tan	81	5.5e-03	1	47	CC	cytoplasmic part
turquoise	529	6.3e-05	1	9	BP	nuclear-transcribed mRNA catabolic proce
turquoise	529	2.4e-04	1	97	MF	nucleic acid binding
turquoise	529	3.0e-04	1	74	BP	positive regulation of macromolecule bio
turquoise	529	3.2e-04	1	9	BP	translational initiation
turquoise	529	3.3e-04	1	15	BP	sensory perception of chemical stimulus
turquoise	529	4.5e-04	1	13	BP	sensory perception of smell
turquoise	529	5.3e-04	1	36	MF	transcription factor binding
turquoise	529	6.2e-04	1	64	BP	positive regulation of transcription, DN
turquoise	529	6.2e-04	1	64	BP	positive regulation of RNA biosynthetic
turquoise	529	1.0e-03	1	4	MF	DNA-directed 5′-3′ RNA polymerase activi
yellow	199	1.2e-04	1	3	MF	nickel cation binding

1.5 5. Export of networks to external software

In [43]:
```
#  Code chunk 29
# Load the expression and trait data saved in the first part
lnames = load(file = "FemaleLiver-01-dataInput.RData");
#The variable lnames contains the names of loaded variables.
lnames
# Load network data saved in the second part.
lnames = load(file = "FemaleLiver-02-networkConstruction-auto.RData");
lnames
```

1. 'datExpr' 2. 'datTraits'
1. 'MEs' 2. 'moduleLabels' 3. 'moduleColors' 4. 'geneTree'

In [99]:
```
#  Code chunk 30
# Recalculate topological overlap if needed
TOM = TOMsimilarityFromExpr(datExpr, power = 6);
# Read in the annotation file
annot = read.csv(file = "GeneAnnotation.csv");
# Select modules
modules = c("brown", "red");
# Select module probes
probes = names(datExpr)
inModule = is.finite(match(moduleColors, modules));
modProbes = probes[inModule];
modGenes = annot$gene_symbol[match(modProbes, annot$substanceBXH)];
# Select the corresponding Topological Overlap
modTOM = TOM[inModule, inModule];
dimnames(modTOM) = list(modProbes, modProbes)
# Export the network into edge and node list files Cytoscape can read
cyt = exportNetworkToCytoscape(modTOM,
  edgeFile = paste("CytoscapeInput-edges-", paste(modules, collapse="-"), ".txt", sep
  nodeFile = paste("CytoscapeInput-nodes-", paste(modules, collapse="-"), ".txt", sep
  weighted = TRUE,
  threshold = 0.5,
  nodeNames = modProbes,
  altNodeNames = modGenes,
  nodeAttr = moduleColors[inModule]);
```

```
TOM calculation: adjacency..
..will use 8 parallel threads.
 Fraction of slow calculations: 0.361682
..connectivity..
..matrix multiplication (system BLAS)..
..normalization..
..done.
```

Open these two files as node table and edge table with Cytoscape and inspect the network