main

July 12, 2021

1 Building consensus network with WGCNA

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

1.1 Functions

[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
}


prepare_data=function(setLabels)
{
    suppressMessages(library(dplyr))
    # Load sample data
    load("../../../../differential_analysis/caudate/_m/genes/voomSVA.RData")
    phenotypes = v$targets %>% as.data.frame %>% select(RNum, Race)
    sample_table0 = v$design %>% as.data.frame %>% select(-Intercept) %>%
        rename("Ancestry"="EA", "Sex"="Male")
    sample_table = phenotypes %>%
        inner_join(tibble::rownames_to_column(sample_table0, "RNum"),␣
 ↪by=c("RNum")) %>%
        mutate("V1"=RNum) %>% tibble::column_to_rownames("V1")
    ## Filter by ancestry
    aa_samples = phenotypes %>% filter(Race == "AA")
    ea_samples = phenotypes %>% filter(Race == "CAUC")
    print(dim(aa_samples))
    print(dim(ea_samples))
    # Load residualized expression
    vsd <- data.table::fread(paste0("../../../../differential_analysis/caudate/
↪",
                                    "_m/genes/residualized_expression.tsv")) %>%
        replace(is.na(.), "") %>% tibble::column_to_rownames("V1")
    print(dim(vsd))
    # Keep only the columns and rows that are present in
    # both the sample table and vsd file
    samples_aa = intersect(colnames(vsd), rownames(aa_samples))
    samples_ea = intersect(colnames(vsd), rownames(ea_samples))
    vsd_aa = vsd[,samples_aa]
    vsd_ea = vsd[,samples_ea]
    # WGCNA data import
    suppressMessages(library(WGCNA))
    nSets = 2; shortLabels = c("AA", "EA")
    multiExpr0 = vector(mode="list", length=nSets)
    multiExpr0[[1]] = list(data=as.data.frame(t(vsd_aa)))
    names(multiExpr0[[1]]$data) = rownames(vsd_aa)
    rownames(multiExpr0[[1]]$data) = colnames(vsd_aa)
    multiExpr0[[2]] = list(data=as.data.frame(t(vsd_ea)))
    names(multiExpr0[[2]]$data) = rownames(vsd_ea)
    rownames(multiExpr0[[2]]$data) = colnames(vsd_ea)
    exprSize = checkSets(multiExpr0)
    print(exprSize)
    # Remove offending genes and samples from the data
    gsg = goodSamplesGenesMS(multiExpr0, verbose = 3);
    if (!gsg$allOK)
    {
        for(set in 1:exprSize$nSets){
            multiExpr0[[set]]$data = multiExpr0[[set]]$data[gsg$goodSamples,␣
↪gsg$goodGenes]
        }
    }
    # Secondary sample filtering
    for(set in 1:exprSize$nSets){
        multiExpr0[[set]]$data = filter_outliers(multiExpr0[[set]]$data, 2.5)
    }
    multiExpr <- multiExpr0
    exprSize = checkSets(multiExpr)
    samples_aa = intersect(rownames(multiExpr[[1]]$data), rownames(aa_samples))
    samples_ea = intersect(rownames(multiExpr[[2]]$data), rownames(ea_samples))
    samples = c(samples_aa, samples_ea)
    sample_table = sample_table[samples,]
    save(multiExpr, exprSize, sample_table, shortLabels, file = '00.RData')
}
```

```
plot_sample_clustering <- function(setLabels){
    lnames = load('00.RData')
    sampleTrees = list()
    for(set in 1:exprSize$nSets){
        sampleTrees[[set]] = hclust(dist(multiExpr[[set]]$data),␣
 ↪method="average")
    }
    pdf(file='sample_clustering.pdf', height=12, width=12)
    par(mfrow=c(2,1))
    par(mar=c(0,4,2,0))
    for(set in 1:exprSize$nSets){
        plot(sampleTrees[[set]],
             main=paste("Sample clustering on all genes in ", setLabels[set]),
             xlab="", sub="", cex=0.7)
    }
    dev.off()
}
```

[3]:
```
prepare_traits = function()
{
    lnames = load('00.RData')
    Traits <- vector(mode="list", length=exprSize$nSets)
    # Associate traits with samples
    for(set in 1:exprSize$nSets){
        setSamples = rownames(multiExpr[[set]]$data)
        traitRows = match(setSamples, sample_table$RNum)
        Traits[[set]] = list(data=sample_table[traitRows, c(-1, -2)])
        rownames(Traits[[set]]$data) = sample_table[traitRows, 1]
    }
    nGenes = exprSize$nGenes
    nSamples = exprSize$nSamples
    save(multiExpr, exprSize, sample_table, shortLabels,
         Traits, nGenes, nSamples, file = "01.RData")
}


plot_power_parameter <- function(nSets, multiExpr, RsquaredCut = 0.85){
    # Choose a set of soft-thresholding powers
    powers = seq(from = 4, to=20, by=1)
    # Initialize a list to hold the results of scale-free analysis
    powerTables = vector(mode = "list", length = nSets)
    softPowerTables = vector(mode = "list", length = nSets)
    # Call the network topology analysis function for each set in turn
    for (set in 1:nSets){
        powerTables[[set]] = list(data =␣
↪pickSoftThreshold(multiExpr[[set]]$data,
                                                          powerVector=powers,␣
↪verbose = 2,
                                                         ␣
↪networkType=PARAM_NETWORK_TYPE)[[2]])
        # Calculated softpower from fitted values
        cond = powerTables[[set]]$data$`SFT.R.sq` > RsquaredCut
        softPowerTables[[set]] = min(powerTables[[set]]$data[cond,"Power"])
    }
    softpower = max(unlist(softPowerTables))
    print(softpower)
    # Plot the results:
    colors = c("black", "red")
    # Will plot these columns of the returned scale free analysis tables
    plotCols = c(2,5,6,7)
    colNames = c("Scale Free Topology Model Fit", "Mean connectivity",
                 "Median connectivity", "Max connectivity")
    # Get the minima and maxima of the plotted points
    ylim = matrix(NA, nrow = 2, ncol = 4)
    for (set in 1:nSets){
        for (col in 1:length(plotCols)){
            ylim[1, col] = min(ylim[1, col],
                               powerTables[[set]]$data[, plotCols[col]],
                               na.rm = TRUE)
            ylim[2, col] = max(ylim[2, col],
                               powerTables[[set]]$data[, plotCols[col]],
                               na.rm = TRUE)
        }
    }
    # Plot the quantities in the chosen columns vs. the soft thresholding power
    sizeGrWindow(8, 6)
    pdf(file = "power_parameter_selection.pdf", wi = 8, he = 6)
    par(mfcol = c(2,2))
    par(mar = c(4.2, 4.2 , 2.2, 0.5))
    cex1 = 0.7
    for (col in 1:length(plotCols)) for (set in 1:nSets){
        if (set==1){
            plot(powerTables[[set]]$data[,1],␣
↪-sign(powerTables[[set]]$data[,3])*powerTables[[set]]$data[,2],
                 xlab="Soft Threshold (power)",ylab=colNames[col],type="n",␣
↪ylim = ylim[, col],
                 main = colNames[col])
            addGrid()
        }
        if (col==1){
            text(powerTables[[set]]$data[,1],␣
↪-sign(powerTables[[set]]$data[,3])*powerTables[[set]]$data[,2],
                 labels=powers,cex=cex1,col=colors[set])
        } else {
            text(powerTables[[set]]$data[,1],␣
↪powerTables[[set]]$data[,plotCols[col]],
                 labels=powers,cex=cex1,col=colors[set])
        }
        if (col==1){
            legend("bottomright", legend = setLabels, col = colors, pch = 20)
        } else {
            legend("topright", legend = setLabels, col = colors, pch = 20)
        }
    }
    dev.off()
}


figure_out_power_parameter <- function()
{
    suppressMessages(library(WGCNA))
    #enableWGCNAThreads()
    lnames = load('01.RData')
    nSets = exprSize$nSets
    plot_power_parameter(nSets, multiExpr, 0.85)
}
```

[4]:
```
construct_network <- function(softPower){
    suppressMessages(library(WGCNA))
    enableWGCNAThreads()
    lnames = load("01.RData")
    # softPower value from previous plot power_parameter_selection.pdf
    cor <- WGCNA::cor
    net = blockwiseConsensusModules(multiExpr, maxBlockSize=30000,
                                    power=softPower, minModuleSize=30,
                                    deepSplit=2, pamRespectsDendro=FALSE,
                                    mergeCutHeight=0.25, numericLabels=TRUE,
                                    minKMEtoStay=0, corType="bicor",
                                    saveTOMFileBase="TOM", saveTOMs=TRUE,
                                    networkType=PARAM_NETWORK_TYPE,
                                    TOMType=PARAM_NETWORK_TYPE, verbose=3)
    consMEs = net$multiMEs
    moduleLabels = net$colors
    moduleColors = labels2colors(moduleLabels)
    consTree = net$dendrograms[[1]]
    save(net, consMEs, moduleLabels, moduleColors, consTree, file="02.RData")
}
```

```
plot_cluster_dendrogram <- function(){
    suppressMessages(library(WGCNA))
    lnames = load("02.RData")
    sizeGrWindow(8,6)
    pdf(file = "consensus_dendrogram.pdf", wi = 8, he = 6)
    plotDendroAndColors(consTree, moduleColors, "Module colors",␣
 ↪dendroLabels=FALSE,
                        hang=0.03, addGuide=TRUE, guideHang=0.05,
                        main="Consensus gene dendrogram and module colors")
    dev.off()
}
```

[5]:
```
consensus_eigengene_network <- function(){
    suppressMessages(library(WGCNA))
    lnames = load(file = "01.RData")
    lnames = load(file = "02.RData")
    nSets = exprSize$nSets
    # Create a variable weight that will hold just the body weight of mice in␣
 ↪both sets
    ancestry = vector(mode = "list", length = nSets);
    for (set in 1:nSets){
        ancestry[[set]] = list(data = as.data.
 ↪frame(Traits[[set]]$data$Ancestry))
        names(ancestry[[set]]$data) = "ancestry"
    }
    # Recalculate consMEs to give them color names
    consMEsC = multiSetMEs(multiExpr, universalColors = moduleColors)
    # Plot eigengene network
    sizeGrWindow(8,10)
    pdf(file = "eigengene_networks.pdf", width=8, height=10)
    par(cex = 0.9)
    plotEigengeneNetworks(consMEsC, setLabels, marDendro=c(0,2,2,1),
                          marHeatmap=c(3,3,2,1), xLabelsAngle=0,
                          zlimPreservation=c(0.5, 1))
    dev.off()
    # We add the weight trait to the eigengenes and order them by consesus␣
 ↪hierarchical clustering:
    MET = consensusOrderMEs(addTraitToMEs(consMEsC, ancestry))
    # Plot eigengene network
    sizeGrWindow(8,10)
    pdf(file = "eigengene_networks_ancestry.pdf", width=8, height=10)
    par(cex = 0.9)
    plotEigengeneNetworks(MET, setLabels, marDendro=c(0,2,2,1),
                          marHeatmap=c(3,3,2,1), xLabelsAngle=0,
                          zlimPreservation=c(0.5, 1))
    dev.off()
    save(MET, consMEsC, ancestry, file="03.RData")
}


export_eigengene_tables = function(){
    suppressMessages(library(WGCNA))
    lnames = load(file = "01.RData")
    lnames = load(file = "02.RData")
    lnames = load(file = "03.RData")
    nSets = exprSize$nSets
    ## Export eigengene tables
    for(set in 1:nSets){
        write.csv(consMEsC[[set]]$data,
                  paste0('eigengenes_',shortLabels[[set]],'.csv'))
    }
    # Write modules
    modules = data.frame(row.names=colnames(multiExpr[[1]]$data),
                         module=moduleColors)
    write.csv(modules, 'modules.csv')
}
```

1.2 Main

[6]:
```
setLabels = c("AA Caudate", "EA Caudate")
prepare_data(setLabels)
plot_sample_clustering(setLabels)
prepare_traits()
figure_out_power_parameter()
```

```
Loading required package: limma


[1] 122   2
[1] 117   2
```

```
Warning message in
data.table::fread(paste0("../../../../differential_analysis/caudate/", :
"Detected 239 column names but the data has 240 columns (i.e. invalid file).
Added 1 extra default column name for the first column which is guessed to be
row names or an index. Use setnames() afterwards if this guess is not correct,
or fix the file write command that created the file to create a valid file."
```

```
[1] 22374   239
$nSets
[1] 2

$nGenes
[1] 22374
```

```
$nSamples
[1] 122 117

$structureOK
[1] TRUE

 Flagging genes and samples with too many missing values…
  ..step 1
   ..bad gene count: 0, bad sample counts: 0, 0
```

png: 2

```
pickSoftThreshold: will use block size 1999.
 pickSoftThreshold: calculating connectivity for given powers…
   ..working on genes 1 through 1999 of 22374
```

```
Warning message:
"executing %dopar% sequentially: no parallel backend registered"
```

```
   ..working on genes 2000 through 3998 of 22374
   ..working on genes 3999 through 5997 of 22374
   ..working on genes 5998 through 7996 of 22374
   ..working on genes 7997 through 9995 of 22374
   ..working on genes 9996 through 11994 of 22374
   ..working on genes 11995 through 13993 of 22374
   ..working on genes 13994 through 15992 of 22374
   ..working on genes 15993 through 17991 of 22374
   ..working on genes 17992 through 19990 of 22374
   ..working on genes 19991 through 21989 of 22374
   ..working on genes 21990 through 22374 of 22374
   Power SFT.R.sq  slope truncated.R.sq mean.k. median.k. max.k.
1      4    0.452 -12.40          0.900 1570.00  1550.000 1990.0
2      5    0.685 -10.50          0.919  841.00   825.000 1240.0
3      6    0.836  -8.32          0.950  458.00   444.000  811.0
4      7    0.917  -6.77          0.973  255.00   242.000  556.0
5      8    0.955  -5.57          0.986  144.00   134.000  398.0
6      9    0.974  -4.64          0.994   83.00    74.600  295.0
7     10    0.981  -3.95          0.995   48.90    42.100  225.0
8     11    0.981  -3.46          0.993   29.50    24.100  179.0
9     12    0.977  -3.10          0.991   18.20    13.900  147.0
10    13    0.967  -2.84          0.986   11.50     8.150  123.0
11    14    0.955  -2.62          0.980    7.50     4.810  105.0
12    15    0.941  -2.44          0.973    5.02     2.880   90.6
13    16    0.937  -2.28          0.973    3.45     1.730   79.0
14    17    0.926  -2.15          0.970    2.44     1.060   69.5
15    18    0.922  -2.04          0.971    1.77     0.651   61.6
16    19    0.912  -1.95          0.966    1.32     0.405   54.9
17    20    0.918  -1.85          0.970    1.00     0.255   49.2
pickSoftThreshold: will use block size 1999.
```

```
 pickSoftThreshold: calculating connectivity for given powers…
   ..working on genes 1 through 1999 of 22374
   ..working on genes 2000 through 3998 of 22374
   ..working on genes 3999 through 5997 of 22374
   ..working on genes 5998 through 7996 of 22374
   ..working on genes 7997 through 9995 of 22374
   ..working on genes 9996 through 11994 of 22374
   ..working on genes 11995 through 13993 of 22374
   ..working on genes 13994 through 15992 of 22374
   ..working on genes 15993 through 17991 of 22374
   ..working on genes 17992 through 19990 of 22374
   ..working on genes 19991 through 21989 of 22374
   ..working on genes 21990 through 22374 of 22374
   Power SFT.R.sq slope truncated.R.sq  mean.k. median.k. max.k.
1      4    0.334 -9.21          0.874 1570.000  1550.000 1960.0
2      5    0.539 -8.48          0.890  838.000   824.000 1170.0
3      6    0.723 -7.27          0.921  456.000   443.000  727.0
4      7    0.851 -6.16          0.948  252.000   241.000  474.0
5      8    0.935 -5.06          0.973  142.000   133.000  326.0
6      9    0.963 -4.40          0.980   81.800    73.800  242.0
7     10    0.963 -3.93          0.978   48.000    41.500  191.0
8     11    0.968 -3.45          0.980   28.800    23.600  156.0
9     12    0.976 -3.04          0.985   17.700    13.600  130.0
10    13    0.982 -2.71          0.990   11.200     7.910  111.0
11    14    0.985 -2.46          0.993    7.270     4.650   96.3
12    15    0.985 -2.27          0.994    4.860     2.760   85.3
13    16    0.983 -2.11          0.994    3.350     1.660   76.3
14    17    0.977 -1.98          0.989    2.380     1.000   68.7
15    18    0.977 -1.87          0.995    1.730     0.614   62.3
16    19    0.977 -1.78          0.996    1.300     0.379   56.7
17    20    0.971 -1.71          0.994    0.998     0.236   51.8
[1] 7
```

png: 2

[7]:
```
softpower = 11 ## Based on Dentate Gyrus and Hippocampus
construct_network(softpower)
plot_cluster_dendrogram()
```

```
Allowing parallel execution with up to 63 working processes.
 Calculating consensus modules and module eigengenes block-wise from all genes
 Calculating topological overlaps block-wise from all genes
   Flagging genes and samples with too many missing values…
    ..step 1
 …Working on set 1
    TOM calculation: adjacency..
    ..will use 63 parallel threads.
     Fraction of slow calculations: 0.000000
    ..connectivity..
    ..matrix multiplication (system BLAS)..
    ..normalization..
    ..done.
 …Working on set 2
    TOM calculation: adjacency..
    ..will use 63 parallel threads.
     Fraction of slow calculations: 0.000000
    ..connectivity..
    ..matrix multiplication (system BLAS)..
    ..normalization..
    ..done.
 ..Working on block 1 .
 …Working on set 1
 …Working on set 2
 …Calculating consensus network
 ..Working on block 1 .
 …clustering and detecting modules..
 …calculating eigengenes..
 …checking consensus modules for statistical meaningfulness..
 …checking for genes that should be reassigned..
 ..merging consensus modules that are too close..
     mergeCloseModules: Merging modules whose distance is less than 0.25
       Calculating new MEs…
```

png: 2

[8]:
```
consensus_eigengene_network()
export_eigengene_tables()
```

```
multiSetMEs: Calculating module MEs.
  Working on set 1 …
  Working on set 2 …
```

1.3 Repredicibility Information

[9]:
```
Sys.time()
proc.time()
options(width = 120)
sessioninfo::session_info()
```

```
[1] "2021-07-12 11:27:25 EDT"
```

```
   user   system  elapsed
5060.883 1628.467 1264.950
```

```
 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     2021-07-12

 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)
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)
fansi            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          1.6.2   2021-05-17 [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
```