

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/dlpfc/_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/dlpfc/",
                                "_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 DLPFC", "EA DLPFC")
prepare_data(setLabels)
plot_sample_clustering(setLabels)
prepare_traits()
figure_out_power_parameter()

```

Loading required package: limma

```

[1] 123  2
[1] 88  2

```

Warning message in
data.table::fread(paste0("../../../differential_analysis/dlpfc/", :
"Detected 211 column names but the data has 212 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] 22398  211
$nSets
[1] 2

```

```

$nGenes
[1] 22398

```

```

$nSamples
[1] 123  88

```

```
$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 1997.
```

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

```
..working on genes 1 through 1997 of 22398
```

```
Warning message:
```

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

```
..working on genes 1998 through 3994 of 22398
```

```
..working on genes 3995 through 5991 of 22398
```

```
..working on genes 5992 through 7988 of 22398
```

```
..working on genes 7989 through 9985 of 22398
```

```
..working on genes 9986 through 11982 of 22398
```

```
..working on genes 11983 through 13979 of 22398
```

```
..working on genes 13980 through 15976 of 22398
```

```
..working on genes 15977 through 17973 of 22398
```

```
..working on genes 17974 through 19970 of 22398
```

```
..working on genes 19971 through 21967 of 22398
```

```
..working on genes 21968 through 22398 of 22398
```

	Power	SFT.R.sq	slope	truncated.R.sq	mean.k.	median.k.	max.k.
1	4	0.163	-5.17	0.831	1610.00	1580.000	2020.0
2	5	0.498	-6.29	0.889	875.00	852.000	1270.0
3	6	0.771	-6.07	0.952	486.00	465.000	858.0
4	7	0.901	-5.32	0.980	275.00	258.000	612.0
5	8	0.949	-4.56	0.985	160.00	145.000	458.0
6	9	0.967	-3.92	0.981	94.60	83.000	356.0
7	10	0.975	-3.45	0.981	57.50	48.000	289.0
8	11	0.979	-3.06	0.979	35.80	28.100	241.0
9	12	0.979	-2.76	0.976	22.90	16.700	205.0
10	13	0.979	-2.53	0.975	15.00	10.000	178.0
11	14	0.977	-2.34	0.973	10.10	6.090	156.0
12	15	0.977	-2.17	0.972	7.01	3.740	138.0
13	16	0.976	-2.05	0.971	4.98	2.330	123.0
14	17	0.977	-1.93	0.973	3.63	1.460	110.0
15	18	0.979	-1.84	0.976	2.71	0.930	99.1
16	19	0.977	-1.76	0.974	2.07	0.597	89.7
17	20	0.972	-1.70	0.969	1.61	0.387	81.6

```
pickSoftThreshold: will use block size 1997.
```

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

```
..working on genes 1 through 1997 of 22398
```

```
..working on genes 1998 through 3994 of 22398
```



```

..working on genes 3995 through 5991 of 22398
..working on genes 5992 through 7988 of 22398
..working on genes 7989 through 9985 of 22398
..working on genes 9986 through 11982 of 22398
..working on genes 11983 through 13979 of 22398
..working on genes 13980 through 15976 of 22398
..working on genes 15977 through 17973 of 22398
..working on genes 17974 through 19970 of 22398
..working on genes 19971 through 21967 of 22398
..working on genes 21968 through 22398 of 22398

```

	Power	SFT	R.sq	slope	truncated	R.sq	mean.k.	median.k.	max.k.
1	4	0.572	-7.84		0.911	1680.00	1640.000	2250.0	
2	5	0.741	-6.06		0.941	936.00	905.000	1460.0	
3	6	0.842	-5.00		0.965	536.00	508.000	1010.0	
4	7	0.900	-4.16		0.979	315.00	291.000	721.0	
5	8	0.929	-3.60		0.984	190.00	169.000	539.0	
6	9	0.940	-3.24		0.985	117.00	100.000	421.0	
7	10	0.940	-2.98		0.982	74.60	60.400	340.0	
8	11	0.946	-2.77		0.985	48.60	36.800	284.0	
9	12	0.952	-2.59		0.988	32.50	22.800	241.0	
10	13	0.956	-2.43		0.990	22.30	14.200	208.0	
11	14	0.946	-2.33		0.983	15.60	9.040	182.0	
12	15	0.943	-2.23		0.982	11.20	5.790	161.0	
13	16	0.941	-2.14		0.980	8.26	3.760	143.0	
14	17	0.938	-2.05		0.979	6.20	2.460	128.0	
15	18	0.940	-1.97		0.982	4.74	1.640	116.0	
16	19	0.943	-1.90		0.984	3.69	1.100	105.0	
17	20	0.944	-1.84		0.985	2.92	0.741	95.1	

```
[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 Reproducibility Information

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

```

[1] "2021-07-12 12:11:58 EDT"

      user  system elapsed
5416.852 1581.629 1381.389

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)
pbdsZMQ	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