Methods S1. WGCNA Code utilized for analysis on the Liver Data, related to GLDS-168 and the methods section called WGCNA Analysis on GLDS-168 Liver Tissue in the STAR methods.

```
#### WGCNA_ PCA _NASA LIVER STUDY #################
#11_SEPT_2019_RMELLER
#_____________________________________________________________________________
___
# first clean memory
rm(list=ls())
#set working directory
setwd("D:/R/WGCNA-PROJECTS/NASA/Project-1 Balbc/RR3")
getwd()
dir.create("Plots")

# 1. Load library WCGNA and set number of processors for multithreading
#install.packages("BiocManager")
#BiocManager::install("WGCNA")

library(WGCNA)
allowWGCNAThreads(8)

# 2. Import data
# The following setting is important, do not omit.
options(stringsAsFactors = FALSE);
#Read in the liver data set
options(stringsAsFactors = FALSE);
extdata.dir = "D:/R/WGCNA-PROJECTS/NASA/DATA"
#Read in the liver data sets
RR3Data =
read.table(paste(extdata.dir,"vsd_RR3_Liver_countdata_Deanne_Integers_April_0
8_19.txt", sep="/"), sep="\t", header = TRUE)
RR3Data[1:8,1:8]
dim(RR3Data)
# data contains ERCC data, so so only chose ENS gene IDS. First create
Dataframe with first column as rownames
temp=RR3Data[,-1]; rownames(temp)=RR3Data[,1]
filter = rownames(temp)[grep("^ENS", rownames(temp))]
LivData = as.matrix(temp[filter,])
LivData[1:8,1:8]

#Remove the BSL samples from the data
LivData=LivData[,-c(1:4)]
dim(LivData)

# The data output from DeSeq will need to be transposed (WGCNA requires genes
on cols, samples on rows) removing the first 8 columns (these contain row
name data)
datExpr0 = as.data.frame(t(LivData));
datExpr0[1:8,1:8]

#Load experimental attributes data
RR3traitData <- read.table(paste(extdata.dir, "Groups_Liver_RR3.txt", sep
="/"), header=TRUE, sep="\t")
RR3traitData <- RR3traitData[-c(1:4),]
RR3traitData
temp <- rownames(datExpr0)
temp
```

```
trait<- as.data.frame(cbind(RR3traitData$Samples, RR3traitData$Samples,
RR3traitData$condition))
colnames(trait) <-c("Tissue","Species", "Condition")
rownames(trait) = temp
trait
trait$Tissue <- "Liver"
trait$Species <- "Balbc"
traitData=trait
traitData
str(traitData)

#Form a data frame and make sure attributes match
samples = rownames(datExpr0);
samples
traitRows = match(samples, rownames(traitData));
traitRows

# now convert to numerical values for correlation
traitData$Condition <- as.numeric(factor(traitData$Condition, levels=c("FLT",
"GC")))
traitData$Tissue <- as.numeric("1")
traitData$Species <- as.numeric("1")
str(traitData)

# check for samples with too many missing variables
gsg = goodSamplesGenes(datExpr0, verbose = 3);
gsg$allOK

# if not TRUE then you need to remove genes with too many missing values
if (!gsg$allOK)
{
  # Optionally, print the gene and sample names that were removed:
  if (sum(!gsg$goodGenes)>0)
     printFlush(paste("Removing genes:",
paste(names(datExpr0)[!gsg$goodGenes], collapse = ", ")));
  if (sum(!gsg$goodSamples)>0)
     printFlush(paste("Removing samples:",
paste(rownames(datExpr0)[!gsg$goodSamples], collapse = ", ")));
  # Remove the offending genes and samples from the data:
  datExpr0 = datExpr0[gsg$goodSamples, gsg$goodGenes]
}

#2.b use PCA to test for outlier

Sample <- data.frame(t(na.omit(t(datExpr0))))
pca <- prcomp(Sample, retx=TRUE)
library(ggfortify); library (ggplot2)
sizeGrWindow(12,9)
pdf(file ="Plots/NASA_Liver_PCA.pdf", width = 12, height = 9)
autoplot(pca, label = TRUE, label.size = 3)
dev.off()

# then plot to screen
autoplot(pca, label = TRUE, label.size = 3)

#3. use Hierarchical cluster to also identify outliers (can cut if needed)
sampleTree = hclust(dist(datExpr0), method = "average");
```

```
#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.
sizeGrWindow(12,9)

# Plot the sample tree to a pdf:
pdf(file = "Plots/sampleClustering.pdf", width = 12, height = 9);
par(cex = 0.6);
par(mar = c(0,4,2,0))
plot(sampleTree, main = "Sample clustering to detect outliers", sub="",
xlab="", cex.lab = 1.5,
     cex.axis = 1.5, cex.main = 2)
dev.off()

# or print to screen
plot(sampleTree, main = "Sample clustering to detect outliers", sub="",
xlab="", cex.lab = 1.5,
     cex.axis = 1.5, cex.main = 2)

# data are split in two- batching effect

#2.e recluster data using clinical traits
sampleTree2 = hclust(dist(datExpr0), method = "average")
# Convert traits to a color representation: white means low, red means high,
grey means missing entry
traitColors = numbers2colors(traitData, signed = FALSE, centered = FALSE,
colors = blueWhiteRed(100))

# Plot the sample dendrogram and the colors underneath and save as pdf.
sizeGrWindow(12,9)
pdf(file ="Plots/NASA_Liver_clustering.pdf", width = 12, height = 9)
plotDendroAndColors(sampleTree2, traitColors,
                    groupLabels = names(traitData),
                    main = "Sample dendrogram and trait heatmap")
dev.off()

# print to screen
sizeGrWindow(12,9); plotDendroAndColors(sampleTree2, traitColors,
                    groupLabels = names(traitData),
                    main = "Sample dendrogram and trait heatmap")


#2.f save the data
save(datExpr0, traitData, file = "NASA_LIVER_RR3.RData")

######################################################################
#################################STEP TWO#############################
######################################################################

## 3 Create the network
# 3.a 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(datExpr0, powerVector = powers, verbose = 5)
# Plot the results:
sizeGrWindow(9, 5);
par(mfrow = c(1,2));
```

```
cex1 = 0.9;
# 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="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")

## use connectivity of 10

library(doParallel);cl <- makeCluster(6); registerDoParallel(cl)

net = blockwiseModules(datExpr0, power = 10, maxBlockSize = 30000,
                       TOMType = "unsigned", minModuleSize = 50,
                       reassignThreshold = 0, mergeCutHeight = 0.25,
                       numericLabels = TRUE, pamRespectsDendro = FALSE,
                       saveTOMs = TRUE,
                       saveTOMFileBase = "NASA_Expr-TOM",
                       verbose = 3)
# to list the number of modules and number of genes within
table (net$colors)

#3c Now plot a graphical representation of the networks
sizeGrWindow(12, 9)
# Convert labels to colors for plotting
mergedColors = labels2colors(net$colors)
# Plot the dendrogram and the module colors underneath as a pdf
pdf(file = "Plots/NASA_Liver_Dendroandcolors.pdf", width = 12, height = 9)
plotDendroAndColors(net$dendrograms[[1]], mergedColors[net$blockGenes[[1]]],
                    "Module colors",
                    dendroLabels = FALSE, hang = 0.03,
                    addGuide = TRUE, guideHang = 0.05)
dev.off()

# now plot on screen
plotDendroAndColors(net$dendrograms[[1]], mergedColors[net$blockGenes[[1]]],
                    "Module colors",
                    dendroLabels = FALSE, hang = 0.03,
                    addGuide = TRUE, guideHang = 0.05)

# 4 Save output for further analysis
moduleLabels = net$colors
moduleColors = labels2colors(net$colors)
MEs = net$MEs;
geneTree = net$dendrograms[[1]];

save(MEs, moduleLabels, moduleColors, geneTree,
```

```
     file = "RR3_NASA_Liver_network.RData")

#########################################################################
#################################Step THREE##############################
#########################################################################

## 3 Relating Modules to External Experimental Traits
#3.a Quantifying module trait associations
# Define numbers of genes and samples
nGenes = ncol(datExpr0);
nSamples = nrow(datExpr0);
# Recalculate MEs with color labels
MEs0 = moduleEigengenes(datExpr0, moduleColors)$eigengenes
MEs = orderMEs(MEs0)
moduleTraitCor = cor(MEs, traitData, use = "p");
moduleTraitPvalue = corPvalueStudent(moduleTraitCor, nSamples);

# Graphical representation of correlation of ME and traits
sizeGrWindow(10,12)
# 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
pdf(file = "Plots/NASA_Liver-Traitheatmap3.pdf", width = 12, height = 10)
labeledHeatmap(Matrix = moduleTraitCor,
               xLabels = names(traitData),
               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"))
dev.off()

# and on screen....
sizeGrWindow(10,10)
labeledHeatmap(Matrix = moduleTraitCor,
               xLabels = names(traitData),
               yLabels = names(MEs),
               ySymbols = names(MEs),
               colorLabels = FALSE,
               colors = greenWhiteRed(50),
               textMatrix = textMatrix,
               setStdMargins = TRUE,
               cex.text = 0.5,
               zlim = c(-1,1),
               main = paste("Module-trait relationships"))

data <- as.data.frame(cbind(names(MEs), signif(moduleTraitCor),
signif(moduleTraitPvalue)))
dim(data)
write.table(data, file="METocond.txt", sep ="\t")
```

```
#3.b Correlation of genetrait significance and Module Membership
# Define variable Condition containing the condition column of datTrait
dx = as.data.frame(traitData$Condition);
names(dx) = "dx"
# names (colors) of the modules
modNames = substring(names(MEs), 3)

geneModuleMembership = as.data.frame(cor(datExpr0, MEs, use = "p"));
MMPvalue = as.data.frame(corPvalueStudent(as.matrix(geneModuleMembership),
nSamples));

names(geneModuleMembership) = paste("MM", modNames, sep="");
names(MMPvalue) = paste("p.MM", modNames, sep="");

geneTraitSignificance = as.data.frame(cor(datExpr0, dx, use = "p"));
GSPvalue = as.data.frame(corPvalueStudent(as.matrix(geneTraitSignificance),
nSamples));

names(geneTraitSignificance) = paste("GS.", names(dx), sep="");
names(GSPvalue) = paste("p.GS.", names(dx), sep="");

# print out traits as dataframe and order from highest to lowest correlation
of modules to condition. This is easier to search
Newdata <- as.data.frame(moduleTraitCor)
Newdata2 <- as.data.frame(moduleTraitPvalue)
Newdata <- cbind(Newdata, Newdata2$Condition)
Newdata[order(-Newdata$Condition),]



#3.c Identifying genes with high GS and MM
# Identify genes in ## module, which has a significant correlation with dx
module = "skyblue"
column = match(module, modNames);
moduleGenes = moduleColors==module;

sizeGrWindow(7, 7);
par(mfrow = c(1,1));
pdf(file= paste("Plots/RR3",module,".pdf", sep=""))
verboseScatterplot(abs(geneModuleMembership[moduleGenes, column]),
                   abs(geneTraitSignificance[moduleGenes, 1]),
                   xlab = paste("Module Membership in", module, "module"),
                   ylab = "Gene significance for Condition",
                   main = paste("Module membership vs. gene significance\n"),
                   cex.main = 1.2, cex.lab = 1.2, cex.axis = 1.2, col =
"blue")
dev.off()

#3.c Identifying genes with high GS and MM
# Identify genes in ## module, which has a significant correlation with dx
module = "lightpink4"
column = match(module, modNames);
moduleGenes = moduleColors==module;

sizeGrWindow(7, 7);
```

```
par(mfrow = c(1,1));
pdf(file= paste("Plots/RR3",module,".pdf", sep=""))
verboseScatterplot(abs(geneModuleMembership[moduleGenes, column]),
                   abs(geneTraitSignificance[moduleGenes, 1]),
                   xlab = paste("Module Membership in", module, "module"),
                   ylab = "Gene significance for Condition",
                   main = paste("Module membership vs. gene significance\n"),
                   cex.main = 1.2, cex.lab = 1.2, cex.axis = 1.2, col =
"pink")
dev.off()

#########################################################################
################# change to mouse ###############################
#########################################################################

# 3.d. Annotate data and export data to cvs.

##Use the following to convert gene IDs to gene names.
# cannot match ensemblgene names as they finish with .1 etc, so not matching.
need to remove .1.
library(reshape2)
data <- data.frame(names(datExpr0))
df <- transform(data, names.datExpr0 = colsplit(data$names.datExpr0, pattern
="\\.", names = c("ensembl_gene_id", "b")))

#library(mygene)
#annot <- queryMany((df[,2]), scopes="ensembl.gene",
fields=c("ensembl.gene","symbol", "entrezgene"), species = "mouse")
#write.table(annot, file = "Mouse_Annotation.csv", sep = ",")

library(biomaRt)
mart = useMart("ensembl", dataset = "mmusculus_gene_ensembl")
mapping <-
data.frame(getBM(attributes=c('ensembl_gene_id','ensembl_transcript_id',
                              'description',
                              'chromosome_name',
                              'start_position',
                              'end_position',
                              'strand','mgi_symbol','entrezgene_id'),mart =
mart))
write.table(mapping, file = "Mouse_Annotation_mapping.csv", sep = ",")


probes = (df[,2])
probes2annot = match(probes, mapping$ensembl_gene_id)

# The following is the number or probes without annotation:
sum(is.na(probes2annot))
head(mapping)
# Should return 0.

# Create the starting data frame
geneInfo0 = data.frame(ID = probes,
                       geneSymbol = mapping$mgi_symbol[probes2annot],
                       LocusLinkID = mapping$entrezgene_id[probes2annot],
                       moduleColor = moduleColors,
                       geneTraitSignificance,
```

```
                      GSPvalue)

# Order modules by their significance for Dx
modOrder = order(-abs(cor(MEs, dx, 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
geneTraitSignificance
geneOrder = order(geneInfo0$moduleColor, -abs(geneInfo0$GS.dx));
geneInfo = geneInfo0[geneOrder, ]
write.csv(geneInfo, file = "geneInfo.csv")

#write module specific lists with locuslink IDs
allLLIDs = mapping$entrezgene_id[probes2annot];

# $ Choose interesting modules
intModules = c("lightpink4", "skyblue")
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)

# now repeat for genesymbols
allgeneIDs = mapping$mgi_symbol[probes2annot]
intModules = intModules
for (module in intModules)
{
  # Select module probes
  modGenes = (moduleColors==module)
  # Get their entrez ID codes
  modIDs = allgeneIDs[modGenes];
  # Write them into a file
  fileName = paste("GeneSymbols-", module, ".txt", sep="");
  write.table(as.data.frame(modIDs), file = fileName,
              row.names = FALSE, col.names = FALSE)
}
```

```
# As background in the enrichment analysis, we will use all probes in the
analysis.
fileName = paste("GeneSymbols-all.txt", sep="");
write.table(as.data.frame(allgeneIDs), file = fileName,
            row.names = FALSE, col.names = FALSE)

######################################################################These
lists can be ran in other GO software or used in enrichment, such as cluego
############################################################
######################################################################

#5. Visualization in R
# 3.a Visualize all
# # Calculate topological overlap anew: this could be done more efficiently
by saving the TOM
# calculated during module detection, but let us do it again here.
dissTOM = 1-TOMsimilarityFromExpr(datExpr0, power = 10);

# 3b. Visualize top 2000 genes
nSelect = 2000
# For reproducibility, we set the random seed
set.seed(10);
select = sample(nGenes, size = nSelect);
selectTOM = dissTOM[select, select];
# There's no simple way of restricting a clustering tree to a subset of
genes, so we must re-cluster.
selectTree = hclust(as.dist(selectTOM), method = "average")
selectColors = moduleColors[select];

# Open a graphical window
sizeGrWindow(9,9)
# Taking the dissimilarity to a power, say 10, makes the plot more
informative by effectively changing
# the color palette; setting the diagonal to NA also improves the clarity of
the plot
pdf(file ="Plots/RR3_Tomplot_NASA_Liver.pdf")
plotDiss = selectTOM^7;
diag(plotDiss) = NA;
TOMplot(plotDiss, selectTree, selectColors, main = "Network heatmap plot,
selected genes")
dev.off()

# on screen
TOMplot(plotDiss, selectTree, selectColors, main = "Network heatmap plot,
selected genes")

## 4.0 Visualize epigene network
# Recalculate module eigengenes
MEs = moduleEigengenes(datExpr0, moduleColors)$eigengenes
# Isolate weight from the clinical traits
Dx = as.data.frame(traitData$Condition);
names(Dx) = "Condition"
# Add the weight to existing module eigengenes
MET = orderMEs(cbind(MEs, Dx))
# Plot the relationships among the eigengenes and the trait
sizeGrWindow(12,15);
par(cex = 0.9)
```

```
pdf(file= "Plots/RR3_Final_Epigene_map_NASA_Liver.pdf")
plotEigengeneNetworks(MET, "", marDendro = c(0,4,1,2), marHeatmap =
c(3,4,1,2), cex.lab = 0.8, xLabelsAngle
= 90)
dev.off()

# Code to plot the dendrogram and heatmap seperately
sizeGrWindow(10,10);
par(cex = 1.0)
plotEigengeneNetworks(MET, "Eigengene dendrogram", marDendro = c(0,4,2,0),
                      plotHeatmaps = FALSE)
#(note: this plot will overwrite the dendrogram plot)
par(cex = 1.0)
plotEigengeneNetworks(MET, "Eigengene adjacency heatmap", marHeatmap =
c(3,4,2,2),
                      plotDendrograms = FALSE, xLabelsAngle = 90)
```