LabBook_03_06_2016

Claire Green

Tuesday

Today I got started on working on the example of conducting a meta-analysis of data from multiple microarray data sets as written by Jeremy Miller of UCLA. Link - https://labs.genetics.ucla.edu/horvath/CoexpressionNetwork/JMiller/

This is my code so far for the example:

```
setwd(dir="/Users/clairegreen/Downloads/metaAnalysisFiles")

library(WGCNA)
library(impute)
library(dynamicTreeCut)
library(qvalue)
library(flashClust)
library(lattice)
library(survival)
library(Formula)
library(ggplot2)
library(Hmisc)

load("metaAnalysisData.RData")

#  (Section will take ~5-10 minutes to run)
# source("collapseRows_NEW.R") # ONLY uncomment this line if you get an error with it commented
datExprB1g = (collapseRows(datExprB1,genesI,probesI))[[1]]
datExprB2g = (collapseRows(datExprB2,genesA,probesA))[[1]]

commonProbesA = intersect (rownames(datExprA1),rownames(datExprA2))
datExprA1p = datExprA1[commonProbesA,]
datExprA2p = datExprA2[commonProbesA,]

commonGenesB = intersect (rownames(datExprB1g),rownames(datExprB2g))
datExprB1g = datExprB1g[commonGenesB,]
datExprB2g = datExprB2g[commonGenesB,]


###Choosing soft threshold
# 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(Exp, 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$fitIndices[,2],
     labels=powers,cex=cex1,col="red");
# this line corresponds to using an R^2 cut-off of h
abline(h=0.50,col="red")
# Mean connectivity as a function of the soft-thresholding power
plot(sft$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")

#Correlating general network properties
softPower = 10 # (Read WGCNA tutorial to learn how to pick your power)
rankExprA1= rank(rowMeans(datExprA1p))
rankExprA2= rank(rowMeans(datExprA2p))
random5000= sample(commonProbesA,5000)
rankConnA1= rank(softConnectivity(t(datExprA1p[random5000,]),type="signed",power=softPower))
rankConnA2= rank(softConnectivity(t(datExprA2p[random5000,]),type="signed",power=softPower))

rankExprB1= rank(rowMeans(datExprB1g))
rankExprB2= rank(rowMeans(datExprB2g))
random5000= sample(commonGenesB,5000)
rankConnB1= rank(softConnectivity(t(datExprB1g[random5000,]),type="signed",power=softPower))
rankConnB2= rank(softConnectivity(t(datExprB2g[random5000,]),type="signed",power=softPower))

pdf("generalNetworkProperties.pdf", height=10, width=9)
par(mfrow=c(2,2))
verboseScatterplot(rankExprA1,rankExprA2, xlab="Ranked Expression (A1)",
                   ylab="Ranked Expression (A2)")
verboseScatterplot(rankConnA1,rankConnA2, xlab="Ranked Connectivity (A1)",
                   ylab="Ranked Connectivity (A2)")
verboseScatterplot(rankExprB1,rankExprB2, xlab="Ranked Expression (B1)",
                   ylab="Ranked Expression (B2)")
verboseScatterplot(rankConnB1,rankConnB2, xlab="Ranked Connectivity (B1)",
                   ylab="Ranked Connectivity (B2)")
dev.off()

##Run WGCNA on the data sets##

#calculate all of the necessary values to run WGCNA
#(this will take around 10 minutes)
adjacencyA1 = adjacency(t(datExprA1p),power=softPower,type="signed");
diag(adjacencyA1)=0
dissTOMA1   = 1-TOMsimilarity(adjacencyA1, TOMType="signed")
geneTreeA1  = flashClust(as.dist(dissTOMA1), method="average")

adjacencyA2 = adjacency(t(datExprA2p),power=softPower,type="signed");
diag(adjacencyA2)=0
dissTOMA2   = 1-TOMsimilarity(adjacencyA2, TOMType="signed")
geneTreeA2  = flashClust(as.dist(dissTOMA2), method="average")

#save.image("tutorial.RData")  #  (Section will take ~5-15 minutes to run)
```

```
#display the networks visually
pdf("dendrogram.pdf",height=6,width=16)
par(mfrow=c(1,2))
plot(geneTreeA1,xlab="",sub="",main="Gene clustering on TOM-based dissimilarity (A1)", labels=FALSE,hang
plot(geneTreeA2,xlab="",sub="",main="Gene clustering on TOM-based dissimilarity (A2)", labels=FALSE,hang
dev.off()

#determine modules based on control data set (takes approx minute)
mColorh=NULL
for (ds in 0:3){
  tree = cutreeHybrid(dendro = geneTreeA1, pamStage=FALSE,
                      minClusterSize = (30-3*ds), cutHeight = 0.99,
                      deepSplit = ds, distM = dissTOMA1)
  mColorh=cbind(mColorh,labels2colors(tree$labels));
}
pdf("Module_choices.pdf", height=10,width=25);
plotDendroAndColors(geneTreeA1, mColorh, paste("dpSplt =",0:3), main = "",dendroLabels=FALSE);
dev.off()

modulesA1 =  mColorh[,3] #choose based on deepslit values in plot

#calculate the principle components for visualizations
PCs1A    = moduleEigengenes(t(datExprA1p),  colors=modulesA1)
ME_1A    = PCs1A$eigengenes
distPC1A = 1-abs(cor(ME_1A,use="p"))
distPC1A = ifelse(is.na(distPC1A), 0, distPC1A)
pcTree1A = hclust(as.dist(distPC1A),method="a")
MDS_1A   = cmdscale(as.dist(distPC1A),2)
colorsA1 = names(table(modulesA1))
pdf("ModuleEigengeneVisualizations.pdf",height=6,width=8)
par(mfrow=c(1,1), mar=c(0, 3, 1, 1) + 0.1, cex=1)

plot(pcTree1A, xlab="",ylab="",main="",sub="")

plot(MDS_1A, col= colorsA1,  main="MDS plot", cex=2, pch=19)

ordergenes = geneTreeA1$order
plotMat(scale(log(datExprA1p[ordergenes,])) , rlabels= modulesA1[ordergenes], clabels= colnames(datExpr

for (which.module in names(table(modulesA1))){
  ME = ME_1A[, paste("ME",which.module, sep="")]
  barplot(ME, col=which.module, main="", cex.main=2,
          ylab="eigengene expression",xlab="array sample")
}

dev.off()

##Qualitatively and quantitatively measure network preservation at the module level##

#assess how well modules in network 1 are preserved in network 2
pdf("Final_modules.pdf",height=8,width=12)
plotDendroAndColors(geneTreeA1, modulesA1, "Modules", dendroLabels=FALSE, hang=0.03, addGuide=TRUE,
                    guideHang=0.05, main="Gene dendrogram and module colors (A1)")
```

```
plotDendroAndColors(geneTreeA2, modulesA1, "Modules", dendroLabels=FALSE, hang=0.03, addGuide=TRUE,
                    guideHang=0.05, main="Gene dendrogram and module colors (A2)")
dev.off()

#assess how well a module in one study is preserved in another study
# (This step will take ~10 minutes)
multiExpr  = list(A1=list(data=t(datExprA1p)),A2=list(data=t(datExprA2p)))
multiColor = list(A1 = modulesA1)
mp=modulePreservation(multiExpr,multiColor,referenceNetworks=1,verbose=3,networkType="signed",
                      nPermutations=30,maxGoldModuleSize=100,maxModuleSize=400)
stats = mp$preservation$Z$ref.A1$inColumnsAlsoPresentIn.A2
stats[order(-stats[,2]),c(1:2)]

##Module membership (kME) and its use in comparing networks##

#get the kME values
geneModuleMembership1 = signedKME(t(datExprA1p), ME_1A)
colnames(geneModuleMembership1)=paste("PC",colorsA1,".cor",sep="");

MMPvalue1=corPvalueStudent(as.matrix(geneModuleMembership1),dim(datExprA1p)[[2]]);
colnames(MMPvalue1)=paste("PC",colorsA1,".pval",sep="");

Gene       = rownames(datExprA1p)
kMEtable1  = cbind(Gene,Gene,modulesA1)
for (i in 1:length(colorsA1))
  kMEtable1 = cbind(kMEtable1, geneModuleMembership1[,i], MMPvalue1[,i])
colnames(kMEtable1)=c("PSID","Gene","Module",sort(c(colnames(geneModuleMembership1), colnames(MMPvalue1)

write.csv(kMEtable1,"kMEtable1.csv",row.names=FALSE)

# First calculate MEs for A2, since we haven't done that yet
PCs2A = moduleEigengenes(t(datExprA2p),  colors=modulesA1)
ME_2A = PCs2A$eigengenes

geneModuleMembership2 = signedKME(t(datExprA2p), ME_2A)
colnames(geneModuleMembership1)=paste("PC",colorsA1,".cor",sep="");

MMPvalue2=corPvalueStudent(as.matrix(geneModuleMembership2),dim(datExprA2p)[[2]]);
colnames(MMPvalue2)=paste("PC",colorsA1,".pval",sep="");

kMEtable2  = cbind(Gene,Gene,modulesA1)
for (i in 1:length(colorsA1))
  kMEtable2 = cbind(kMEtable2, geneModuleMembership2[,i], MMPvalue2[,i])
colnames(kMEtable2)=colnames(kMEtable1)

write.csv(kMEtable2,"kMEtable2.csv",row.names=FALSE)

#plot the kME values
pdf("all_kMEtable2_vs_kMEtable1.pdf",height=8,width=8)
for (c in 1:length(colorsA1)){
  verboseScatterplot(geneModuleMembership2[,c],geneModuleMembership1[,c],main=colorsA1[c],
                     xlab="kME in A2",ylab="kME in A1")
}; dev.off()
```

```
pdf("inModule_kMEtable2_vs_kMEtable1.pdf",height=8,width=8)
for (c in 1:length(colorsA1)){
  inMod = modulesA1== colorsA1[c]
  verboseScatterplot(geneModuleMembership2[inMod,c],geneModuleMembership1[inMod,c],main=colorsA1[c],
                     xlab="kME in A2",ylab="kME in A1")
}; dev.off()

#determine which genes are hubs in both networks
topGenesKME = NULL
for (c in 1:length(colorsA1)){
  kMErank1    = rank(-geneModuleMembership1[,c])
  kMErank2    = rank(-geneModuleMembership2[,c])
  maxKMErank  = rank(apply(cbind(kMErank1,kMErank2+.00001),1,max))
  topGenesKME = cbind(topGenesKME,Gene[maxKMErank<=10])
}; colnames(topGenesKME) = colorsA1
topGenesKME

##Comparing networks and annotating modules using programs outside of R##

#output data from our network for import into VisANT
source("tutorialFunctions.R")
#Dataset 1
for (co in colorsA1[colorsA1!="grey"])
  visantPrepOverall(modulesA1, co, t(datExprA1g), rownames(datExprA1p), 500, softPower, TRUE)
#Dataset 2
for (co in colorsA1[colorsA1!="grey"])
  visantPrepOverall(modulesA1, co, t(datExprA2g), rownames(datExprA2p), 500, softPower, TRUE)

## This function will output a lot of files into your current directory, which you should immediately pu

#Hub genes specific to one network
datExprA12g = t(cbind(datExprA1g,datExprA2g))
i1 = 1:dim(datExprA1g)[[2]];
i2 = (1:dim(datExprA2g)[[2]])+length(i1)
for (co in colorsA1[colorsA1!="grey"])
  visantPrep(modulesA1, co, i1, i2, datExprA12g, rownames(datExprA1g), 500, softPower, TRUE)

#Use EASE to annotate modules based on Gene Ontology (GO) enrichment

#Create a new folder called "geneLists_A12" then run this code:
dir.create("geneLists_A12/")
  folder = "geneLists_A12/"
for (c in colorsA1){
  fn = paste(folder, c, ".txt",sep="");
  write(Gene[modulesA1==c], fn)
};
write(Gene,paste(folder,"all.txt",sep=""))

#"all.txt" is what you should use as your population file when asked in EASE


#annotate modules based enrichment for user-defined lists
enrichments = userListEnrichment(Gene, modulesA1, c("exampleListInput.csv","exampleMMInput.csv"),
```

```
                                    c("cellType","humanModules"), "enrichment.csv")


##Using phenotypic information to determine differentially expressed genes and modules##

#find all of the modules that are phenotype related
region = rep("CA1",32);
region[c(1,4,6,11,12,15,16,17,22,24,25,26,28,29,31,32)] = "CA3"
age = c(81,72,86,90,88,90,90,74,83,73,73,70,85,85,75,90,72,70,90,84,75,85,80,
        86,85,84,81,88,80,90,83,74)

# Find the region-related modules
var     = list(region=="CA1", region=="CA3")
datRegM = t(apply(t(ME_1A),1,t.test.1))
colnames(datRegM)=c("MeanCA1","MeanCA3","SD_CA1","SD_CA3","PvalRegion")
datRegM[datRegM[,5]<0.02,]

#visualize these results
pdf("RegionAgePlots.pdf",width=16,height=4)
par(mfrow=c(1,4))
verboseBoxplot(as.numeric(datExprA1g["NRIP3",]), region,
               main="NRIP3 expression -", las=2, xlab="Region", ylab="")
verboseScatterplot(age, as.numeric(datExprA1g["DDX42",]),
                   main="DDX42 expression -", las=2, abline=TRUE, xlab="Age", ylab="")
verboseBoxplot(as.numeric(ME_1A[,"MEgreen"]), region,
               main="Green ME expr. -", las=2, xlab="Region", ylab="")
verboseScatterplot(age, as.numeric(ME_1A[,"MEmagenta"]),
                   main="Magenta ME expr. -", las=2, abline=TRUE, xlab="Age", ylab="")
dev.off()

#visually compare how a gene or module relates to phenotype across data sets
region2 = rep("CA1",31);
region2[c(1,4,7,8,9,11,15,18,20,21,22,26,28,29,30)] = "CA3"
var     = list(region2=="CA1", region2=="CA3")
datReg2 = t(apply(datExprA2g,1,t.test.1))
colnames(datReg2)=c("MeanCA1","MeanCA3","SD_CA1","SD_CA3","PvalRegion")
datRegM2 = t(apply(t(ME_2A),1,t.test.1))
colnames(datRegM2)=c("MeanCA1","MeanCA3","SD_CA1","SD_CA3","PvalRegion")

pdf("RegionAgePlots12.pdf",width=16,height=4)
par(mfrow=c(1,4))
verboseBoxplot(as.numeric(datExprA1g["NRIP3",]), region,
               main="NRIP3 expression (A1) -", las=2, xlab="Region (A1)", ylab="")
verboseBoxplot(as.numeric(datExprA2g["NRIP3",]), region2,
               main="NRIP3 expression (A2) -", las=2, xlab="Region (A2)", ylab="")
verboseBoxplot(as.numeric(ME_1A[,"MEgreen"]), region,
               main="Green ME expr. (A1) -", las=2, xlab="Region (A1)", ylab="")
verboseBoxplot(as.numeric(ME_2A[,"MEgreen"]), region2,
               main="Green ME expr. (A2) -", las=2, xlab="Region (A2)", ylab="")
dev.off()

#
```

It is also stored in the R file meta_WGCNA.R

This afternoon I attempted to work out how to run this on iceberg. I'm a little rusty but this is what I have remembered to do so far:

login is mdp15cmg@iceberg.sheffield.ac.uk. Password is the same as logging into uni account. To load R, input

module load apps/R

Wednesday

Today I attempted to add my C9orf72 and CHMP2B data sets into the meta WGCNA code. This is what I have so far:

```
setwd(dir="/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Code/Results/GeneExpression/DEG_Test2/")

library(WGCNA)
library(impute)
library(dynamicTreeCut)
library(qvalue)
library(flashClust)
library(lattice)
library(survival)
library(Formula)
library(ggplot2)
library(Hmisc)

C9Expr <- read.csv(file = "C9rankeduniqueresult.csv")
rownames(C9Expr) <- C9Expr$Probe.Set.ID
datExprA1 <- C9Expr[,49:59]
CHExpr <- read.csv(file = "CHrankeduniqueresult.csv")
rownames(CHExpr) <- CHExpr$Probe.Set.ID
datExprA2 <- CHExpr[,49:57]

# CHExpr <- read.csv(file = "CHrankeduniqueresult.csv")
# rownames(CHExpr) <- CHExpr$Probe.Set.ID
# datExprB1 <- CHExpr[,55:57] #C9 patients
# datExprB2 <- CHExpr[,49:54] #C9 controls

IDs <- read.csv("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/WGCNA/C9orf72/C9ID geneIDs.csv")
IDs <- C9Expr[,c("Probe.Set.ID", "Gene.Symbol")]
genes <- IDs$Gene.Symbol
probeID <- IDs$Probe.Set.ID

#For matching probes if from different platforms
#  (Section will take ~5-10 minutes to run)
# source("collapseRows_NEW.R") # ONLY uncomment this line if you get an error with it commented
datExprB1g = (collapseRows(datExprB1,genes,probeID))[[1]]
datExprB2g = (collapseRows(datExprB2,genes,probeID))[[1]]

#Limit analysis to common probes
commonProbesA = intersect (rownames(datExprA1),rownames(datExprA2))
datExprA1p = datExprA1[commonProbesA,]
datExprA2p = datExprA2[commonProbesA,]
```

```
# commonGenesB = intersect (rownames(datExprB1g),rownames(datExprB2g))
# datExprB1g = datExprB1g[commonGenesB,]
# datExprB2g = datExprB2g[commonGenesB,]

Exp <- t(datExprA2p)

###Choosing soft threshold
# 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(Exp, 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=0.50,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")

setwd("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/WGCNA/")
#Correlating general network properties
softPower = 6 # (Read WGCNA tutorial to learn how to pick your power)
rankExprA1= rank(rowMeans(datExprA1p))
rankExprA2= rank(rowMeans(datExprA2p))
random5000= sample(commonProbesA,5000)
rankConnA1= rank(softConnectivity(t(datExprA1p[random5000,]),type="signed",power=softPower, minNSamples
rankConnA2= rank(softConnectivity(t(datExprA2p[random5000,]),type="signed",power=softPower, minNSamples

# rankExprB1= rank(rowMeans(datExprB1g))
# rankExprB2= rank(rowMeans(datExprB2g))
# random5000= sample(commonGenesB,5000)
# rankConnB1= rank(softConnectivity(t(datExprB1g[random5000,]),type="signed",power=softPower, minNSampl
# rankConnB2= rank(softConnectivity(t(datExprB2g[random5000,]),type="signed",power=softPower, minNSampl

pdf("generalNetworkProperties.pdf", height=10, width=9)
par(mfrow=c(2,2))
verboseScatterplot(rankExprA1,rankExprA2, xlab="Ranked Expression (A1)",
                   ylab="Ranked Expression (A2)")
verboseScatterplot(rankConnA1,rankConnA2, xlab="Ranked Connectivity (A1)",
                   ylab="Ranked Connectivity (A2)")
# verboseScatterplot(rankExprB1,rankExprB2, xlab="Ranked Expression (B1)",
#                     ylab="Ranked Expression (B2)")
# verboseScatterplot(rankConnB1,rankConnB2, xlab="Ranked Connectivity (B1)",
```

```
#                     ylab="Ranked Connectivity (B2)")
dev.off()

keepGenesDups = (collapseRows(datExprA1p,genes,probeID))[[2]]
datExprA1g    = datExprA1p[keepGenesDups[,2],]
datExprA2g    = datExprA2p[keepGenesDups[,2],]
rownames(datExprA1g)<-rownames(datExprA2g)<-keepGenesDups[,1]

#calculate all of the necessary values to run WGCNA
#(this will take around 10 minutes)
adjacencyA1 = adjacency(t(datExprA1p),power=softPower,type="signed");
diag(adjacencyA1)=0
dissTOMA1   = 1-TOMsimilarity(adjacencyA1, TOMType="signed")
geneTreeA1  = flashClust(as.dist(dissTOMA1), method="average")

adjacencyA2 = adjacency(t(datExprA2p),power=softPower,type="signed");
diag(adjacencyA2)=0
dissTOMA2   = 1-TOMsimilarity(adjacencyA2, TOMType="signed")
geneTreeA2  = flashClust(as.dist(dissTOMA2), method="average")

# adjacencyB1 = adjacency(t(datExprB1g),power=softPower,type="signed");
# diag(adjacencyB1)=0
# dissTOMB1   = 1-TOMsimilarity(adjacencyB1, TOMType="signed")
# geneTreeB1  = flashClust(as.dist(dissTOMB1), method="average")
#
# adjacencyB2 = adjacency(t(datExprB2g),power=softPower,type="signed");
# diag(adjacencyB2)=0
# dissTOMB2   = 1-TOMsimilarity(adjacencyB2, TOMType="signed")
# geneTreeB2  = flashClust(as.dist(dissTOMB2), method="average")

# save.image("tutorial.RData")  #  (Section will take ~5-15 minutes to run)

pdf("dendrogram.pdf",height=6,width=16)
par(mfrow=c(1,2))
plot(geneTreeA1,xlab="",sub="",main="Gene clustering on TOM-based dissimilarity (A1)", labels=FALSE,hang
plot(geneTreeA2,xlab="",sub="",main="Gene clustering on TOM-based dissimilarity (A2)", labels=FALSE,hang
# plot(geneTreeB1,xlab="",sub="",main="Gene clustering on TOM-based dissimilarity (B1)", labels=FALSE,h
# plot(geneTreeB2,xlab="",sub="",main="Gene clustering on TOM-based dissimilarity (B2)", labels=FALSE,h
dev.off()

#determine modules based on control data set

mColorh=NULL
for (ds in 0:3){
  tree = cutreeHybrid(dendro = geneTreeA1, pamStage=FALSE,
                      minClusterSize = (30-3*ds), cutHeight = 0.99,
                      deepSplit = ds, distM = dissTOMA1)
  mColorh=cbind(mColorh,labels2colors(tree$labels));
}
pdf("Module_choices.pdf", height=10,width=25);
plotDendroAndColors(geneTreeA1, mColorh, paste("dpSplt =",0:3), main = "",dendroLabels=FALSE);
dev.off()
```

```
modulesA1 =  mColorh[,4] #choose based on deepslit values in plot

#calculate the principle components for visualizations
PCs1A    = moduleEigengenes(t(datExprA1p),  colors=modulesA1)
ME_1A    = PCs1A$eigengenes
distPC1A = 1-abs(cor(ME_1A,use="p"))
distPC1A = ifelse(is.na(distPC1A), 0, distPC1A)
pcTree1A = hclust(as.dist(distPC1A),method="a")
MDS_1A   = cmdscale(as.dist(distPC1A),2)
colorsA1 = names(table(modulesA1))
pdf("ModuleEigengeneVisualizations.pdf",height=6,width=8)
par(mfrow=c(1,1), mar=c(0, 3, 1, 1) + 0.1, cex=1)

plot(pcTree1A, xlab="",ylab="",main="",sub="")

plot(MDS_1A, col= colorsA1,  main="MDS plot", cex=2, pch=19)

ordergenes = geneTreeA1$order
plotMat(scale(log(datExprA1p[ordergenes,])) , rlabels= modulesA1[ordergenes], clabels= colnames(datExpr

for (which.module in names(table(modulesA1))){
  ME = ME_1A[, paste("ME",which.module, sep="")]
  barplot(ME, col=which.module, main="", cex.main=2,
          ylab="eigengene expression",xlab="array sample")
}

dev.off()

##Qualitatively and quantitatively measure network preservation at the module level##

#assess how well modules in network 1 are preserved in network 2
pdf("Final_modules.pdf",height=8,width=12)
plotDendroAndColors(geneTreeA1, modulesA1, "Modules", dendroLabels=FALSE, hang=0.03, addGuide=TRUE,
                    guideHang=0.05, main="Gene dendrogram and module colors (A1)")
plotDendroAndColors(geneTreeA2, modulesA1, "Modules", dendroLabels=FALSE, hang=0.03, addGuide=TRUE,
                    guideHang=0.05, main="Gene dendrogram and module colors (A2)")
# plotDendroAndColors(geneTreeB1, modulesA1, "Modules", dendroLabels=FALSE, hang=0.03, addGuide=TRUE,
#                     guideHang=0.05, main="Gene dendrogram and module colors (A1)")
# plotDendroAndColors(geneTreeB2, modulesA1, "Modules", dendroLabels=FALSE, hang=0.03, addGuide=TRUE,
#                     guideHang=0.05, main="Gene dendrogram and module colors (A2)")
# dev.off()

#assess how well a module in one study is preserved in another study
# (This step will take ~10 minutes)
multiExpr  = list(A1=list(data=t(datExprA1g)),A2=list(data=t(datExprA2g)))
multiColor = list(A1 = modulesA1)
mp=modulePreservation(multiExpr,multiColor,referenceNetworks=1,verbose=3,networkType="signed",
                      nPermutations=30,maxGoldModuleSize=100,maxModuleSize=400)
stats = mp$preservation$Z$ref.A1$inColumnsAlsoPresentIn.A2
stats[order(-stats[,2]),c(1:2)]

#get the kME values
geneModuleMembership1 = signedKME(t(datExprA1g), ME_1A)
```

```
colnames(geneModuleMembership1)=paste("PC",colorsA1,".cor",sep="");

MMPvalue1=corPvalueStudent(as.matrix(geneModuleMembership1),dim(datExprA1g)[[2]]);
colnames(MMPvalue1)=paste("PC",colorsA1,".pval",sep="");

Gene       = rownames(datExprA1g)
kMEtable1  = cbind(Gene,Gene,modulesA1)
for (i in 1:length(colorsA1))
  kMEtable1 = cbind(kMEtable1, geneModuleMembership1[,i], MMPvalue1[,i])
colnames(kMEtable1)=c("PSID","Gene","Module",sort(c(colnames(geneModuleMembership1), colnames(MMPvalue1

write.csv(kMEtable1,"kMEtable1.csv",row.names=FALSE)

# First calculate MEs for A2, since we haven't done that yet
PCs2A = moduleEigengenes(t(datExprA2g),  colors=modulesA1)
ME_2A = PCs2A$eigengenes

geneModuleMembership2 = signedKME(t(datExprA2g), ME_2A)
colnames(geneModuleMembership1)=paste("PC",colorsA1,".cor",sep="");

MMPvalue2=corPvalueStudent(as.matrix(geneModuleMembership2),dim(datExprA2g)[[2]]);
colnames(MMPvalue2)=paste("PC",colorsA1,".pval",sep="");

kMEtable2  = cbind(Gene,Gene,modulesA1)
for (i in 1:length(colorsA1))
  kMEtable2 = cbind(kMEtable2, geneModuleMembership2[,i], MMPvalue2[,i])
colnames(kMEtable2)=colnames(kMEtable1)

write.csv(kMEtable2,"kMEtable2.csv",row.names=FALSE)

#plot the kME values
pdf("all_kMEtable2_vs_kMEtable1.pdf",height=8,width=8)
for (c in 1:length(colorsA1)){
  verboseScatterplot(geneModuleMembership2[,c],geneModuleMembership1[,c],main=colorsA1[c],
                     xlab="kME in A2",ylab="kME in A1")
}; dev.off()

pdf("inModule_kMEtable2_vs_kMEtable1.pdf",height=8,width=8)
for (c in 1:length(colorsA1)){
  inMod = modulesA1== colorsA1[c]
  verboseScatterplot(geneModuleMembership2[inMod,c],geneModuleMembership1[inMod,c],main=colorsA1[c],
                     xlab="kME in A2",ylab="kME in A1")
}; dev.off()

#determine which genes are hubs in both networks
topGenesKME = NULL
for (c in 1:length(colorsA1)){
  kMErank1    = rank(-geneModuleMembership1[,c])
  kMErank2    = rank(-geneModuleMembership2[,c])
  maxKMErank  = rank(apply(cbind(kMErank1,kMErank2+.00001),1,max))
  topGenesKME = cbind(topGenesKME,Gene[maxKMErank<=10])
}; colnames(topGenesKME) = colorsA1
topGenesKME
```

```
# ##Comparing networks and annotating modules using programs outside of R##
#
# #output data from our network for import into VisANT
# source("tutorialFunctions.R")
# #Dataset 1
# for (co in colorsA1[colorsA1!="grey"])
#   visantPrepOverall(modulesA1, co, t(datExprA1g), rownames(datExprA1g), 500, softPower, TRUE)
# #Dataset 2
# for (co in colorsA1[colorsA1!="grey"])
#   visantPrepOverall(modulesA1, co, t(datExprA2g), rownames(datExprA2g), 500, softPower, TRUE)


# #annotate modules based enrichment for user-defined lists
#
enrichments = userListEnrichment(Gene, modulesA1, c("TDP43Enrichment.csv","kMEtable1.csv"),
                                 c("enrichmentlist","humanModules"), "enrichment.csv")
write.csv(x = enrichments$pValues, file = "allenrichments.csv")
#
# #compare how a gene or module relates to phenotype across data sets
# group <- c("Control","Control","Control","Patient","Patient","Patient","Patient","Patient","Patient",
# var <- list(group=="Control", group=="Patient")
# datgroupM = t(apply(t(ME_1A),1,t.test.l))
# colnames(datgroupM)=c("MeanCon","MeanPat","SD_Con","SD_Pat","PvalGroup")
# datgroupM[datgroupM[,5]<0.05,]
#
# group2 <- c("Control","Control","Control","Control","Control","Control","Patient","Patient","Patient"
# var <- list(group2=="Control", group2=="Patient")
# datgroup2 = t(apply(t(datExprA2g),1,t.test.l))
# colnames(datgroup2)=c("MeanCon","MeanPat","SD_Con","SD_Pat","PvalGroup")
# datgroupM2 = t(apply(t(ME_2A),1,t.test.l))
# colnames(datgroupM2)=c("MeanCon","MeanPat","SD_Con","SD_Pat","PvalGroup")
#
# datgroupM2[datgroupM2[,5]<0.05,]
```

What I end up with (I think) is a list of corresponding modules as denoted by C9orf72 (as one of the data sets has to be used as reference). Although there are a lot of options for visualisation and GO enrichment, I thought the most useful experiment would be to look for enrichment of disease-related genes in each of the modules. userListEnrichment allows you to identify which modules are significantly enriched with lists that you input yourself. The first like "TDP43Enrichmet" contains all the GWASCentral ALS and AD genes, the neuroX lists, subnetwork 28, the Carulli list, and the TDP-43 PPI lists from Pasterkamp and Taylor. The results unfortunately did not show any enrichment within any of the modules (results in allenrichments.csv). From here, I'm not sure what to do about carrying on - is there a point? I suppose the idea is to include the other data sets and hope for a common module to appear, rather than focusing on enrichment. It's going to be tricky though as I have to sort out using a different microarray platform and then try and work out how to include RNA-seq as well...

Also - do I need patients included or not? I'm not looking for differential expression, just commonly expressed modules. Will need to think/Ask Gabriel.

Thursday

I have emailed Gabriel but I thought I would go ahead and add the other two u133 plus 2 data sets just to see what I can make. I decided to just use the patient samples as it makes most sense to me, but it wouldn't be hard to change if Gabriel says otherwise.

Interestingly, all data set ranked expression correlates, but one pair does not correlated in ranked connectivity (C9orf72 and VCP). Not really sure what that means - does it mean that C9 is not a good one to choose as the base for creating modules?

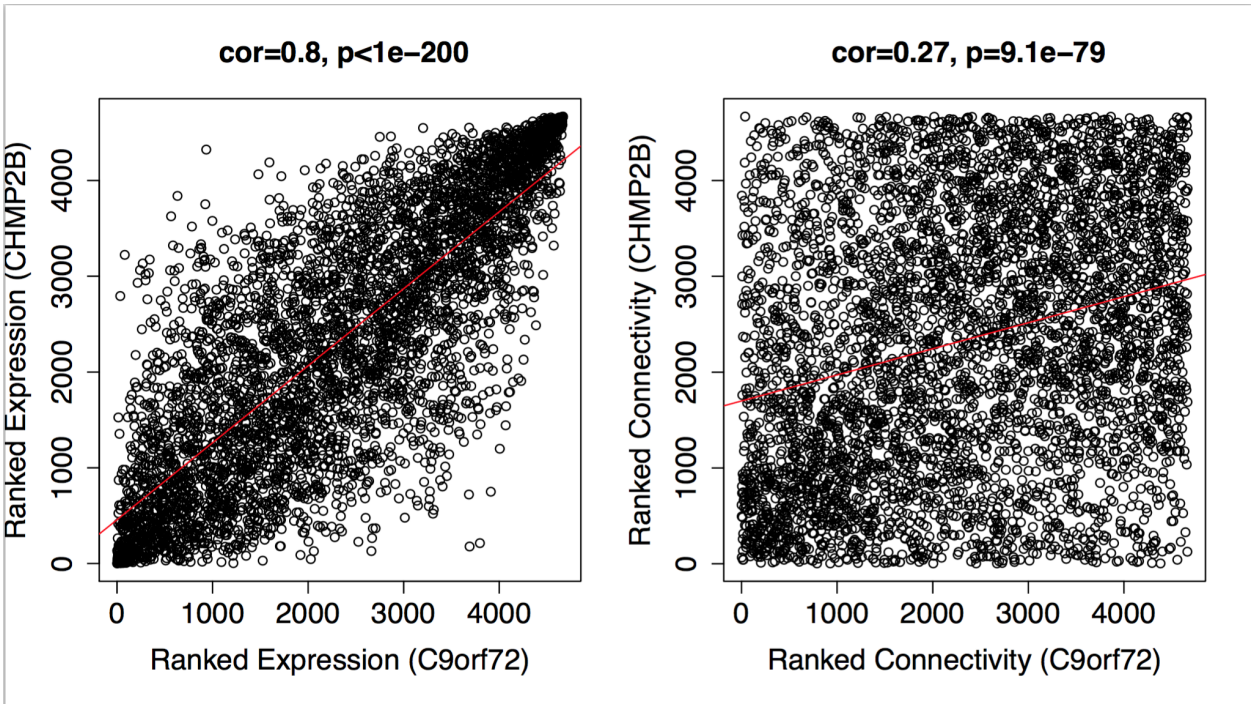

Figure 1:

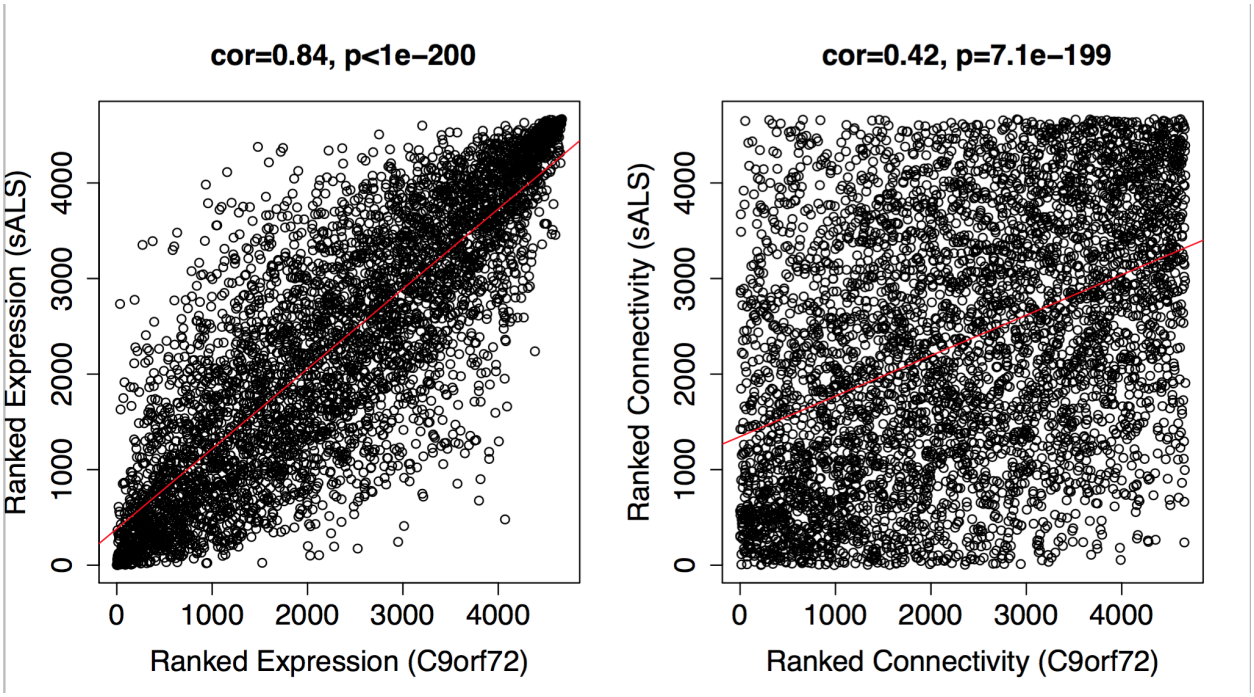

Figure 2:

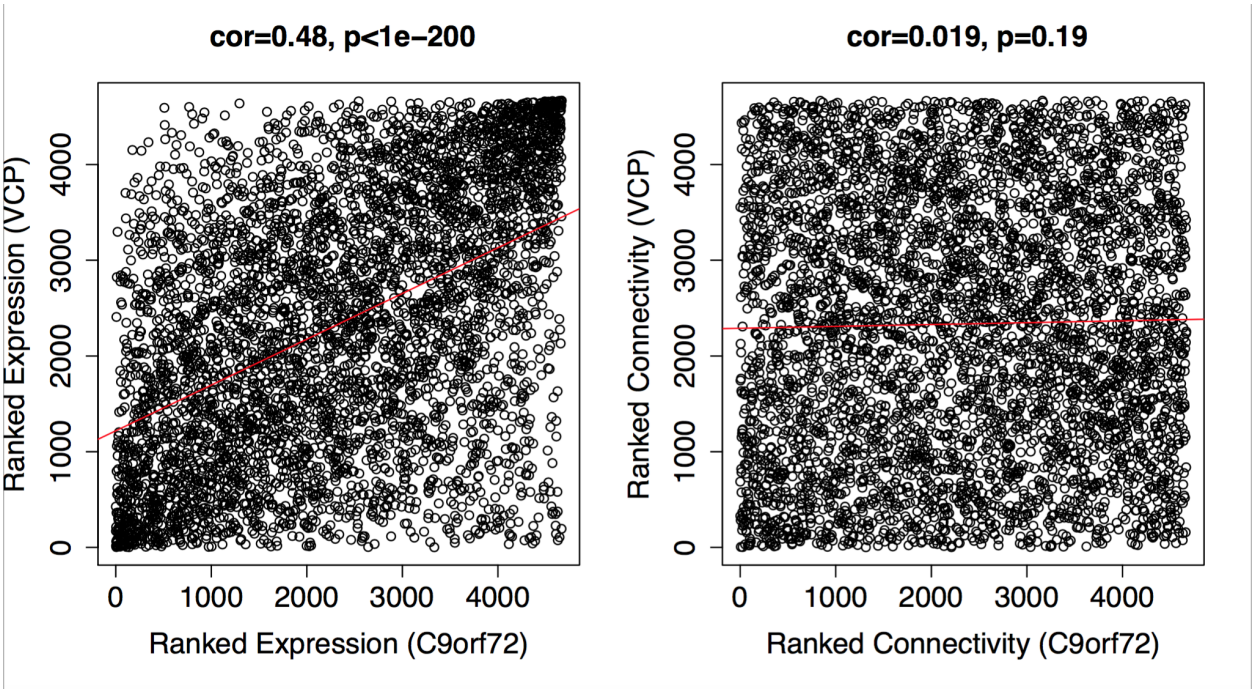

Figure 3:

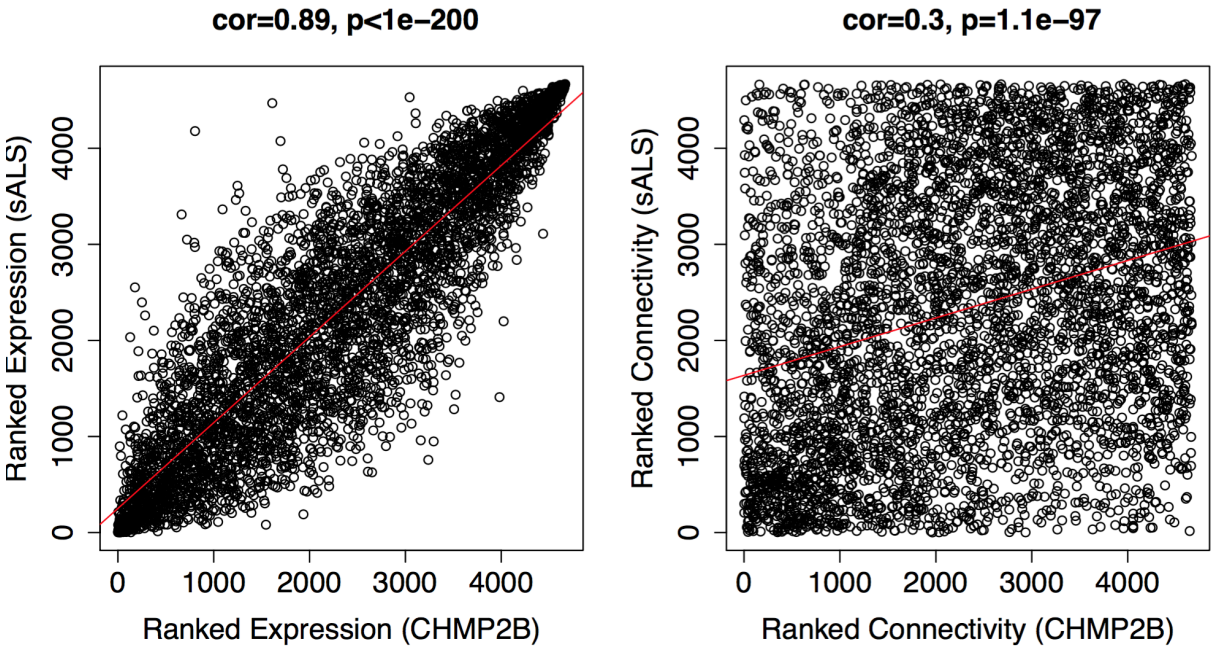

Figure 4:

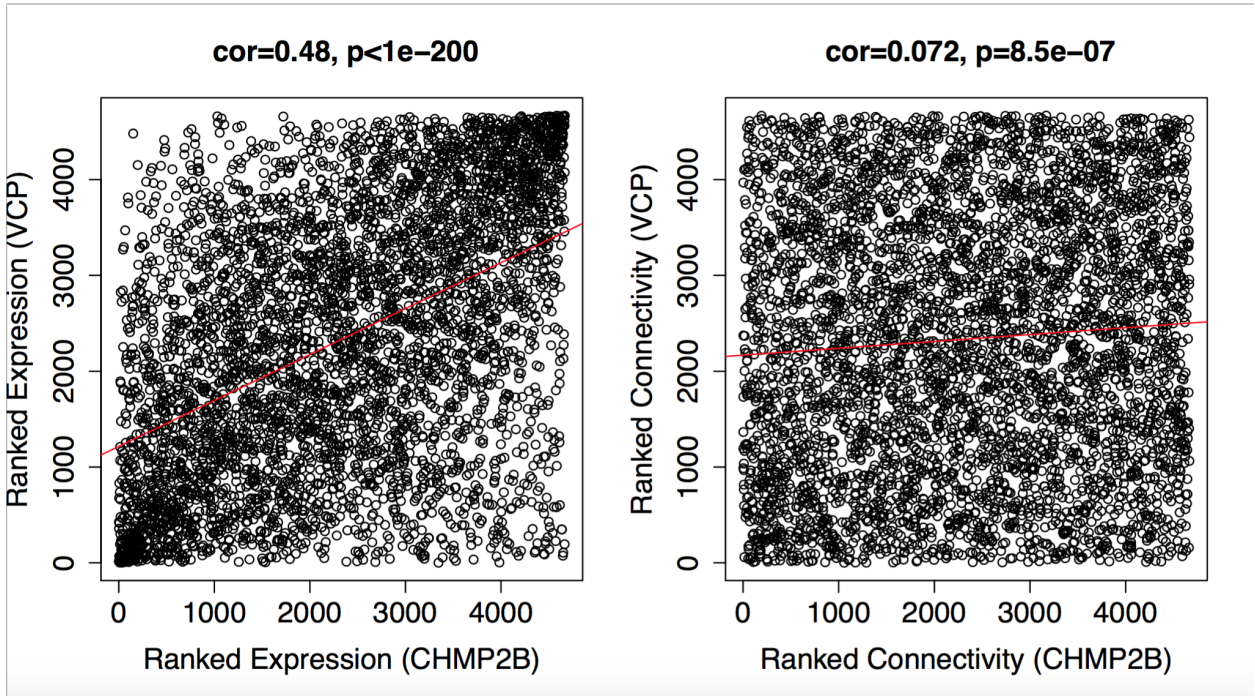

Figure 5:

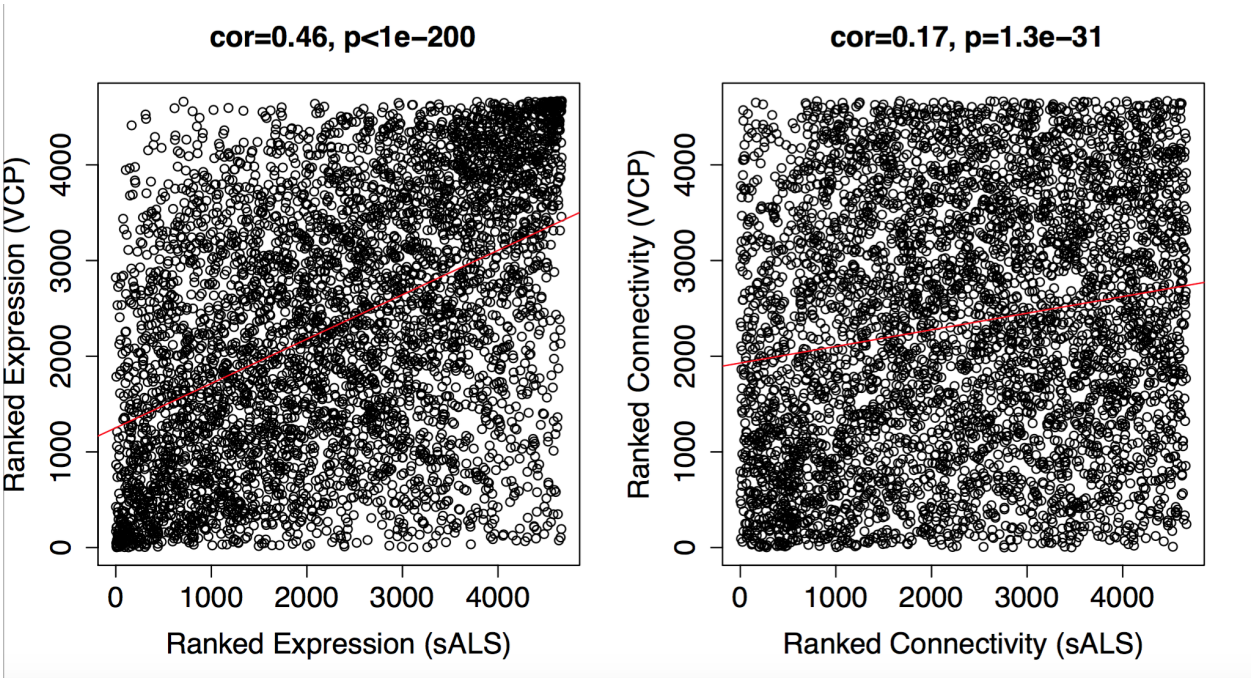
cor=0.46, p<1e−200
Ranked Expression (VCP)
0
1000
2000
3000
4000
Ranked Expression (sALS)
cor=0.17, p=1.3e−31
Ranked Connectivity (VCP)
Ranked Connectivity (sALS)

Figure 6: