Regarding the RPPA data validation: The idea behind that is as follows: In the past it has been shown in several papers (including my own ones) that NEMs can be used in principle to infer parts of non-transcriptional networks based on transcriptome data. This is what you did so far. Now the reasoning is that in case the network is correct it should in principle be in agreement with direct protein measurements. Of course we cannot do that in total, because we only have perturbations of parts of the S-genes. Nonetheless, the effects, which are predicted by your network structure, should be in agreement with these perturbations.

We already discussed in November, how to do the validation technically: We said that we have to distinguish between phospho and total protein concentrations. We said that phospho proteins can be used to validate the network on the level of S-genes. Please note that in the NEM graph when perturbing S-gene s an effect (i.e. significant dysregulation) is predicted for

a) all E-genes directly attached to s.

b) all E-genes attached to S-genes that are reachable from s, i.e. for which there exists a path from s. The latter can be checked easily by (for example) computing the transitive closure of the graph and then looking, which S-genes are connected to s in this transitively closed graph. Alternatively, one can, of course, use depth or breadth first search.

c) In addition to b) one can also check the perturbation of the reachable S-genes itself. This has to be done on the level of the phospho-proteins measured for these S-genes.

In addition to c) one can also partially validate the inferred edge types: If there is an edge S1->S2 and S2 is \*only\* influenced by S1, then in case of an activation S2 should go down and in case of an inhibition go up upon a S1 knock-down.

I hope my explanations made the whole issue a bit clearer. If not, please come back to me.

Best,

Holger

myImagePlot(mat, yLabels= colnames(mat), xLabels=colnames(mat), title=c("Adjacency matrix showing time lags for edges"))

yLabels <- colnames(net$network)

xLabels <- yLabel

# Set min and max values of rand

 min <- min(rand, na.rm=T)

 max <- max(rand, na.rm=T)

# Red and green range from 0 to 1 while Blue ranges from 1 to 0

 ColorRamp <- rgb(seq(0.95,0.99,length=50),  # Red

                  seq(0.95,0.05,length=50),  # Green

                  seq(0.95,0.05,length=50))  # Blue

 ColorLevels <- seq(min, max, length=length(ColorRamp))

myImagePlot <- function(x, ...){

min <- min(x)

max <- max(x)

yLabels <- rownames(x)

xLabels <- colnames(x)

title <-c()

# check for additional function arguments

if( length(list(...)) ){

Lst <- list(...)

if( !is.null(Lst$zlim) ){

min <- Lst$zlim[1]

max <- Lst$zlim[2]

}

if( !is.null(Lst$yLabels) ){

yLabels <- c(Lst$yLabels)

}

if( !is.null(Lst$xLabels) ){

xLabels <- c(Lst$xLabels)

}

if( !is.null(Lst$title) ){

title <- Lst$title

}

}

# check for null values

if( is.null(xLabels) ){

xLabels <- c(1:ncol(x))

}

if( is.null(yLabels) ){

yLabels <- c(1:nrow(x))

}

layout(matrix(data=c(1,2), nrow=1, ncol=2), widths=c(4,1), heights=c(1,1))

# Red and green range from 0 to 1 while Blue ranges from 1 to 0

ColorRamp <- rgb( seq(0.95,0.99,length=256), # Red

seq(0.95,0.05,length=256), # Green

seq(0.95,0.05,length=256)) # Blue

ColorLevels <- seq(min, max, length=length(ColorRamp))

# Reverse Y axis

reverse <- nrow(x) : 1

yLabels <- yLabels[reverse]

x <- x[reverse,]

# Data Map

par(mar = c(3,5,2.5,2))

image(1:length(xLabels), 1:length(yLabels), t(x), col=ColorRamp, xlab="",

ylab="", axes=FALSE, zlim=c(min,max))

if( !is.null(title) ){

title(main=title)

}

axis(BELOW<-1, at=1:length(xLabels), labels=xLabels, cex.axis=0.7, las=3)

axis(LEFT <-2, at=1:length(yLabels), labels=yLabels, las= HORIZONTAL<-1,

cex.axis=0.7)

# Color Scale

par(mar = c(3,2.5,2.5,2))

image(1, ColorLevels,

matrix(data=ColorLevels, ncol=length(ColorLevels),nrow=1),

col=ColorRamp,

xlab="",ylab="",

xaxt="n")

layout(1)

}