Networks

Filipe Russo

July 7, 2019

Introduction

At this time we will use the `R` package `igraph` to build networks based on Pearson Correlation together with the modules detected by the package `WGCNA`. We chose Partition_A and its 9 modules to guide us through the construction of the networks.

```
library(igraph)
library(scales)
library(colorspace)

proteins <- read.csv("proteins.csv", sep = ",", header = TRUE)
datExprA <- t(proteins[, c("Hete", "Mixo", "Auto")])
colnames(datExprA) <- proteins$ID
```

We believe a causal link implies a correlation, but a correlation doesn't imply a causal link. So after we construct our network based on a square correlation matrix we still need to erase misleading correlation links, for that we chose to believe higher absolute correlations are closer to a causal link than lower ones.

Degree & Edge Betweenness

In graph theory, the `degree` of a vertex (protein) of a graph (network) is the number of edges connected to it. The `degree` is a measure of how connected a given protein is to the rest of the (unweighted) network.

The `vertex betweenness` is the number of geodesiscs (shortest paths) going through a vertex (protein). The `vertex betweenness` is a measure of how connected a given (weighted) network is to a particular protein.

Minimum Spanning Tree (MST)

A minimum spanning tree is a subset of the edges (correlations) of a connected, edge-weighted undirected graph that connects all the vertices (proteins) together, without any cycles and with the minimum possible total edge weight.

If each edge (correlation) has a distinct weight then there will be only one, unique minimum spanning tree.

If the weights are positive, then a minimum spanning tree is in fact a minimum-cost subgraph connecting all vertices (proteins).

Let's explore all of those concepts below.

Our Algorithm

1. We pass an adjacency matrix based on Pearson Correlation to the `graph.adjancency()` function from the `Igraph` package.
2. We colour blue the edges from the `valid_g` graph that represent negative correlation and colour red the edges that represent positive correlation.
3. We convert from `valid_g` the edgeweights based on Pearson Correlation to a distance metric by subtracting them from 1.1.

4. We construct from the `valid_g` graph the MST (minimum spanning tree) subgraph and store it in the `valid_mst` variable.
5. We extract from the `valid_mst` subgraph its maximum edgeweight to be our `cut_value`, then we delete from the `valid_g` graph the edges that have a distance higher than the `cut_value`.
6. We define the edgewidths by restoring the values back to absolute correlation status and rescaling said values to a scale from 1 to 3, this way higher absolute correlations will be visually translated to thicker edges.
7. We define a metric to account for both degree and betweenness of the vertices by rescaling each to a scale from 0 to 1, then multiplying one by the other and yet again rescaling the product to a scale from 0 to 1. We store this result in the `new_intensity` variable we will use to lighten the vertices' colors stored in `cols` producing the `new_cols` variable, this way we are able to simultaneously highlight proteins that are highly connected to the network (high degree) and proteins that the network is highly connected to (high betweenness).
8. Finally we plot the `valid_g` graph and the `valid_mst` subgraph, in this order. The first has transparent edges (correlations) for context information, the latter has darker opaque edges and represents the backbone of the network.

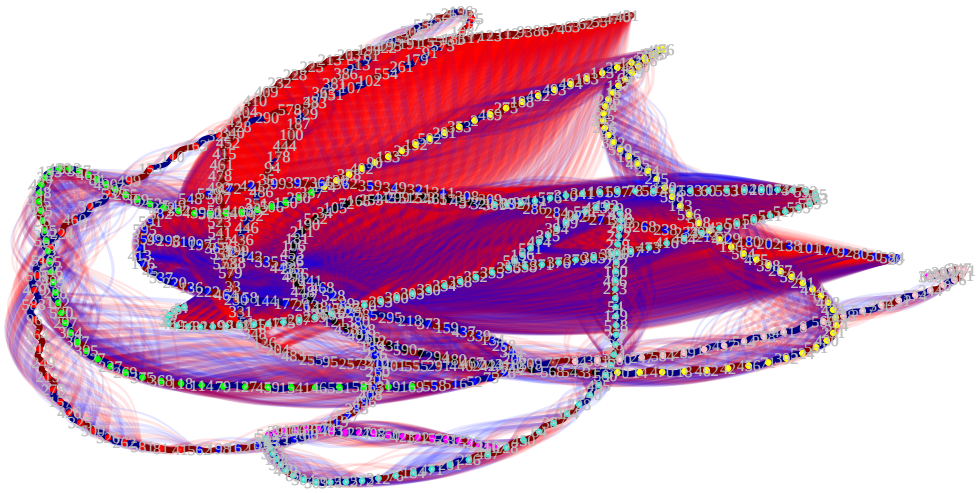
Graph Plots

600 Valid Proteins

```
valid_g <- graph.adjacency(
  as.matrix(as.dist(cor(datExprA[, ], method = "pearson"))),
  mode = "undirected",
  weighted = TRUE,
  diag = FALSE,
  add.colnames = NA,
  add.rownames = NA
)

# Simplfy the adjacency object
valid_g <- simplify(valid_g, remove.multiple = TRUE, remove.loops = TRUE)

# Colour negative correlation edges as blue
E(valid_g)[which(E(valid_g)$weight < 0)]$color <- rgb(0, 0, 1, 0.1)

# Colour positive correlation edges as red
E(valid_g)[which(E(valid_g)$weight > 0)]$color <- rgb(1, 0, 0, 0.1)

# Convert edgeweights based on Pearson correlation to a distance metric
E(valid_g)$weight <- 1.1 - abs(E(valid_g)$weight)

# Store the colors from Partition A's modules in the cols variable
cols <- as.character(proteins$Partition_A)

# Customize the valid_g's inheritable graphical attributes for consistence along plots
V(valid_g)$label <- c(1:length(cols))
V(valid_g)$frame.color <- cols
V(valid_g)$size = 1
V(valid_g)$label.color = "grey"
V(valid_g)$label.cex = 0.5
E(valid_g)$curved = TRUE
```

```r
E(valid_g)$arrow.mode = 0

# From the original graph we construct the MST (minimum spanning tree) subgraph
valid_mst <- mst(valid_g, algorithm = "prim")

# Fruchterman-Reingold is one of the most used force-directed layout algorithms out there
valid_l <- layout_with_fr(valid_mst)
valid_g$layout <- valid_l
valid_mst$layout <- valid_l

# Data derived cut value made to reduce number of edges
cut_value <- max(E(valid_mst)$weight)

# Delete edges that have a distance higher than the cut value
valid_g <- delete_edges(valid_g, E(valid_g)[which(E(valid_g)$weight > cut_value)])

# Define a metric to account for both degree and betweenness by multiplying both
new_intensity <- rescale(rescale(betweenness(valid_g,
                                             directed = FALSE),
                                 to = c(0, 1)) *
                         rescale(degree(valid_g),
                                 to = c(0, 1)),
                         to = c(0,1))

# Redefine the inner colors of the vertices according to the new connectivity metric
new_cols <- lighten(cols, new_intensity)

# Change the inner colors of the graphs' vertices
V(valid_g)$color <- new_cols
V(valid_mst)$color <- new_cols

# Make the edges of the subgraph darker and opaque
E(valid_mst)[which(E(valid_mst)$color == "#0000FF1A")]$color <- rgb(0, 0, 0.5, 1)
E(valid_mst)[which(E(valid_mst)$color == "#FF00001A")]$color <- rgb(0.5, 0, 0, 1)

# Define the edgewidths from graphs in a scale from 1 to 3
E(valid_g)$width <- rescale(abs(E(valid_g)$weight - 1.1), to = c(1, 3))
E(valid_mst)$width <- rescale(abs(E(valid_mst)$weight - 1.1), to = c(1, 3))

# Uncomment the png() function before and the dev.off() after the plot() to save the image

# png(filename = "valid.png",
#     width = 15000,
#     height = 15000,
#     units = "px",
#     res = 400)

plot(
  valid_g,
  asp = FALSE,
  main = "The 600 Valid Proteins divided in 9 Modules"
)
```

```
par(new=T)

plot(
  valid_mst,
  asp = FALSE
)
```



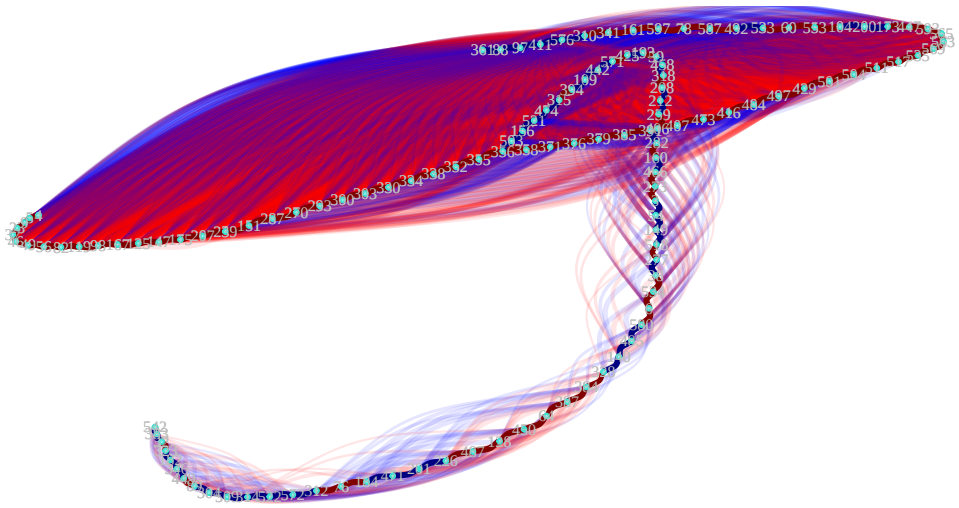

```
par(new = F)

# dev.off()
```

From the 179700 original edges we were able to reduce them to 10794 remaining edges after the MST based cut, while the MST subgraph itself has 599 edges and is the backbone of our network's construction.

Turquoise Module

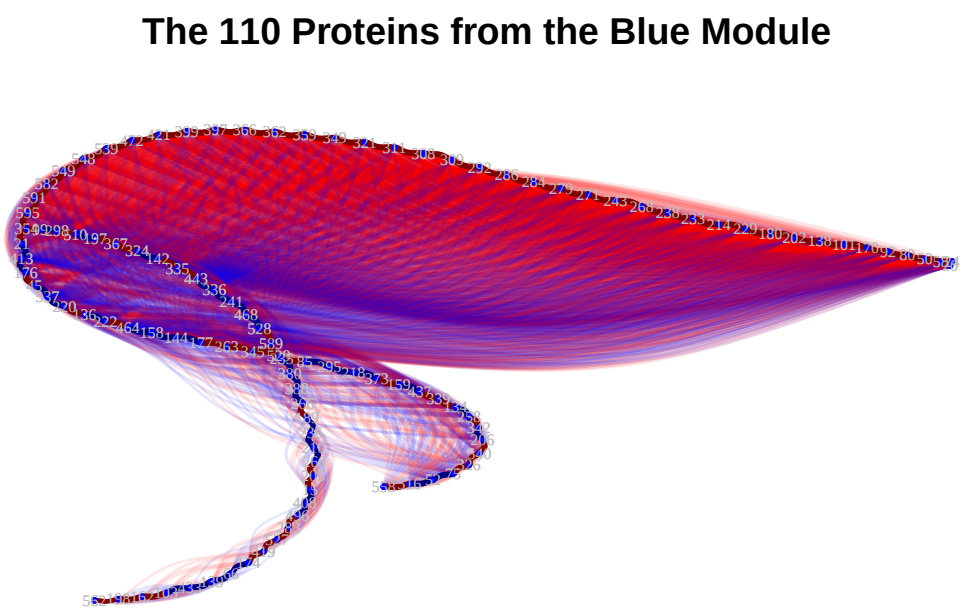
```
not_turquoise <- which(V(valid_g)$frame.color != "turquoise")

turquoise_g <- valid_g
turquoise_g <- delete_vertices(turquoise_g, V(turquoise_g)[not_turquoise])

turquoise_mst <- valid_mst
turquoise_mst <- delete_vertices(turquoise_mst, V(turquoise_mst)[not_turquoise])

turquoise_g$layout <- valid_g$layout[-not_turquoise, ]
turquoise_mst$layout <- valid_g$layout[-not_turquoise, ]

# Uncomment the png() function before and the dev.off() after the plot() to save the image

# png(filename = "turquoise.png",
#     width = 15000,
#     height = 15000,
#     units = "px",
#     res = 400)
```

```
plot(
  turquoise_g,
  asp = FALSE,
  main = "The 128 Proteins from the Turquoise Module"
)

par(new=T)

plot(
  turquoise_mst,
  asp = FALSE
)
```



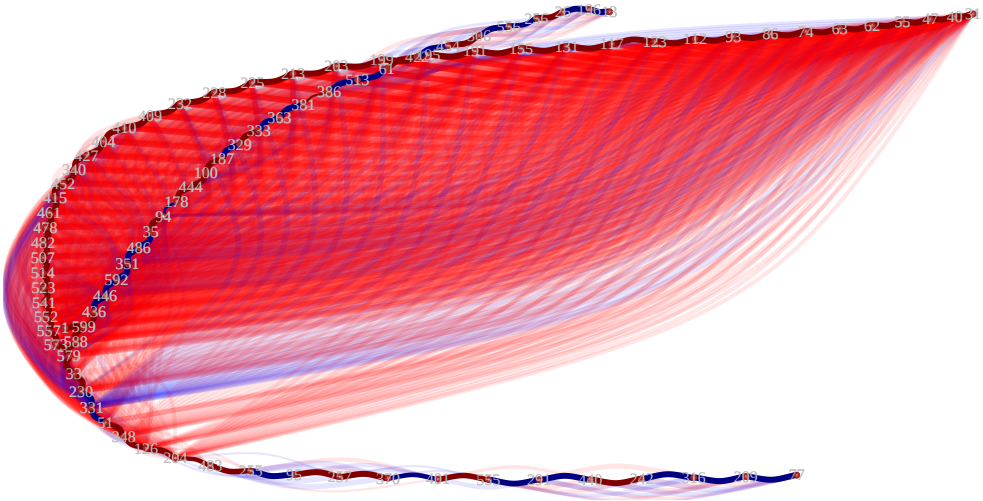

```
par(new = F)

# dev.off()
```

Blue Module



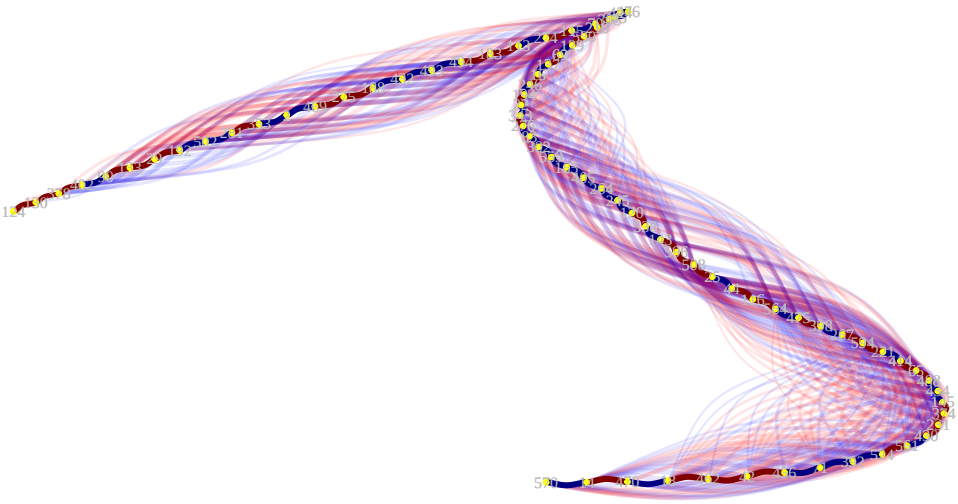
Notice we don't have a white module, so the white vertex found in the Blue Module subgraph is actualy given by our method to highlight important proteins, in this case the protein g1497.t1.

Brown Module



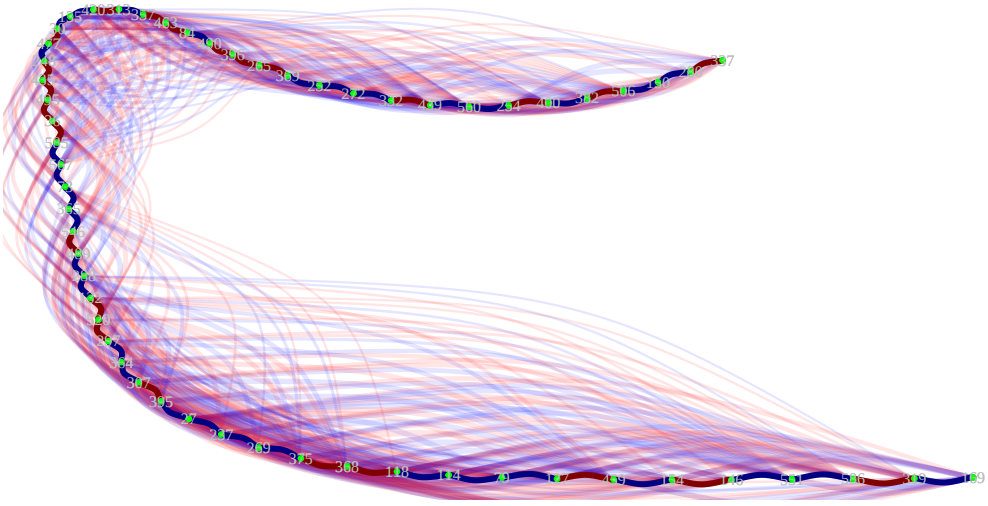
Yellow Module

The 78 Proteins from the Yellow Module

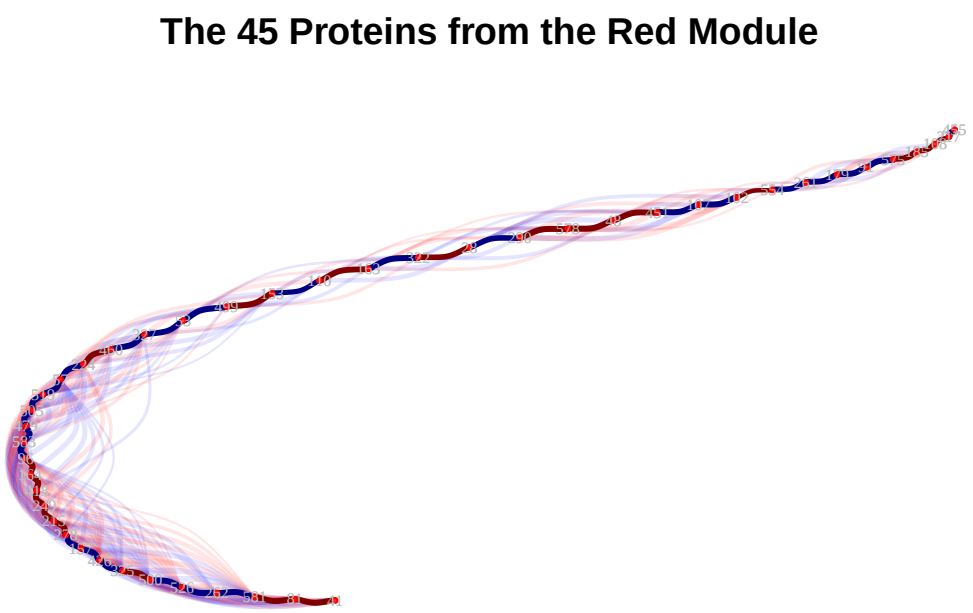


Green Module

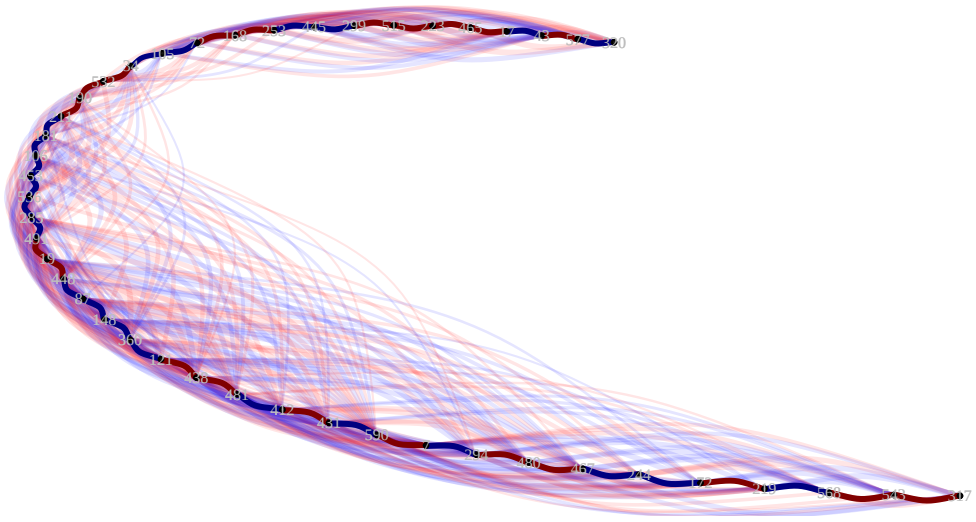
The 57 Proteins from the Green Module



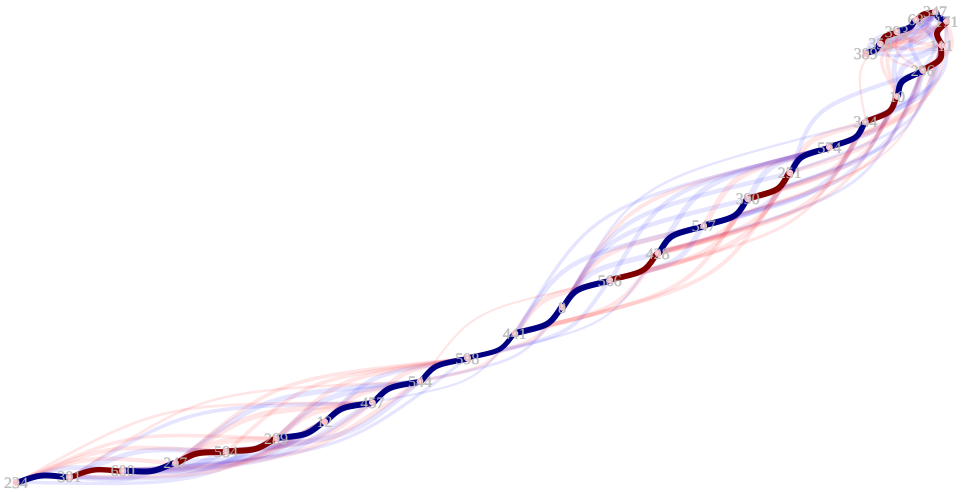
Red Module



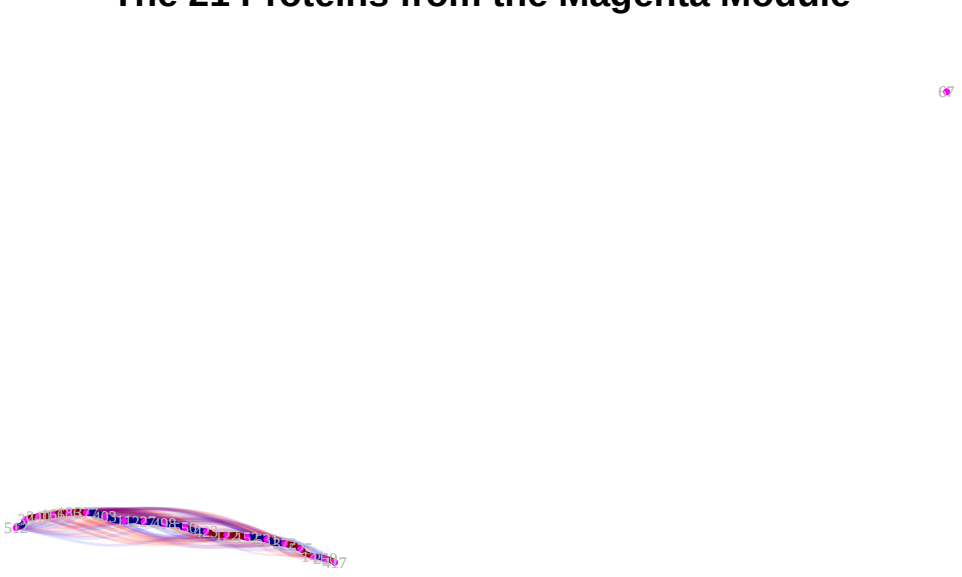
Black Module



Pink Module



Magenta Module



Conclusions

The Minimum Spanning Tree (MST) method offers a good backbone from where to span the construction of a proteomics network, WGCNA's module/cluster detection was consistent with it. The plots themselves offer a collection of insights regarding which proteins and clusters to study in a future analysis, they also provide visual clues on how to visualize and organize proteins as a coexpression network. Zipped together with this last report there is a folder entitled "annex", where one can find high definition images of all 10 plots and visually inspect the relations between the proteins, I highly recommend it.