Visualizing a co-occurence network in Cytoscape

Julia Gustavsen

Contents

1 Description of vignette **2**
- 1.1 Tara Oceans . . . 2
- 1.2 How to examine organisms that occur together: co-occurence networks . . . 2
- 1.3 What can we find out by creating co-occurence networks? . . . 2
- 1.4 What kind of data are used in this vignette? . . . 2

2 Set up Cytoscape and R connection **3**
- 2.1 Requirements . . . 3

3 Read in data **3**

4 Read in taxonomic classification **3**

5 Add the taxonomic classifications to the network and then send network to Cytoscape **4**

6 Send network to Cytoscape using RCy3 **5**

7 Colour network by prokaryotic phylum **7**
- 7.1 Set node shape to reflect virus or prokaryote . . . 9

8 Colour edges of phage nodes **11**

9 Do layout to minimize overlap of nodes. **13**

10 Look at network properties **16**

11 Size by degree **16**

12 Select an interesting node and make a subnetwork from it **18**
- 12.1 Conclusion . . . 20

References **20**

1 Description of vignette

1.1 Tara Oceans

Many projects have collected samples from different regions and from different depths of the ocean. Some, such as the pioneering study by Craig Venter (Venter et al. 2004) have pioneered metagenomic sequencing and others, such as Tara Oceans Expedition have collected large amounts of data with global ecological questions in mind. On the Tara Oceans (8th and 9th expedition for this vessel researchers used a small sailboat outfitted with a lab and filtration supplies to collect samples from many different size fractions of microorganisms in the oceans over three years. They collected these samples to look at the different kinds of microorganisms present in different parts of the oceans to look at their composition and to observe their spatial patterns and distribution.

The scientists collected the samples and then used either targeted sequencing (amplicon approach using primers for specific targets such as the ribosomal genes and then amplifying these targets using PCR) or using metagenomic sequencing (where all the genetic material in a sample is sequenced) of each of the size fractions.

After the sequencing and quality checking of the samples was done, the sequences were taxonomically classified (different approaches for the different targets, see here for the details in Brum et al. (2015) and Sunagawa et al. (2015)). After that the data could be made into a species occurrence table where the rows are different sites and the columns are the observations of the different organisms at each site (Lima-Mendez et al. 2015).

1.2 How to examine organisms that occur together: co-occurence networks

Many of these microbial species in these types of studies have not yet been characterized in the lab. Thus, to know more about the organisms and their interactions, we can observe which ones occur at the same sites or under the same kinds of environmental conditions. One way to do that is by using co-occurrence networks where you examine which organisms occur together at which sites. The more frequently that organisms co-occur at the same site, the stronger the interaction predicted among these organisms. For a review of some of the different kinds of techniques and software for creating interaction networks see: Weiss et al. (2016).

1.3 What can we find out by creating co-occurence networks?

These kinds of analyses can be useful for studies where the organisms have not yet been characterized in the lab because these analyses can provide insights about the communities and how the organisms within them are interacting. These analyses can be exploratory, so that we can see which organisms warrant further insights and perhaps experimental work . We can also learn about how the overall community is organized (community structure) by looking at some of the network properties (that is the overall way that the organisms are co-occurring and the properties of the network seen this way).

1.4 What kind of data are used in this vignette?

In this analysis we are using a Tara Ocean data and we have data from the bacterial dataset (Sunagawa et al. 2015) and also from the viral dataset (Brum et al. 2015). They have been examined in Lima-Mendez et al. (2015) and we have used the original relative abundances to visualize the data. Data were retrieved from: http://www.raeslab.org/companion/ocean-interactome.html

2 Set up Cytoscape and R connection

We will run this example using RCy3 (P. T. Shannon et al. 2013) to drive the visualization of these networks in Cytoscape (P. Shannon et al. 2003) using CyREST (Ono et al. 2015).

2.1 Requirements

```
library(RCy3)
library(igraph)
library(RJSONIO)
library(RColorBrewer)
library(httr)
```

To run this example **Cytoscape software must be running**. In Cytoscape we will also need Allegro-plugin for this example.

To begin we create a connection in R that we can use to manipulate the networks and then we will delete any windows that were already in Cytoscape so that we don't use up all of our memory.

```
cy <- CytoscapeConnection()
deleteAllWindows(cy)
```

3 Read in data

We will read in a species co-occurrence matrix that was calculated using Spearman Rank coefficient. (If interested in seeing how this was done please see scripts and the raw data in inst/data-raw)

```
## scripts for processing located in "inst/data-raw/"
prok_vir_cor <- read.delim("./data/virus_prok_cor_abundant.tsv")
```

There are many different ways to work with graphs in R. We will use both the igraph (Csardi and Nepusz 2006) and the graph (Gentleman et al. 2016) package to work with our network with Cytoscape.

The igraph package is used to convert the co-occurrence dataframe into a network that we can send to Cytoscape. In this case our graph is undirected (so "directed = FALSE") since we do not have any information about the direction of the interactions.

```
graph_vir_prok <- simplify(graph.data.frame(prok_vir_cor,
                                            directed = FALSE))
```

4 Read in taxonomic classification

Since these are data from small, microscopic organisms that were sequenced using shotgun sequencing, we rely on the classification of the sequences to know what kind of organisms are in the samples. In this case the bacterial viruses (bacteriophage), were classified by Basic Local Alignment Search Tool (BLAST http://blast.ncbi.nlm.nih.gov/Blast.cgi) by searching for their closest sequence in the RefSeq database (see methods in Brum et al. (2015)). The prokaryotic taxonomic classifications were determined using the SILVA database.

```r
phage_id_affiliation <- read.delim("./data/phage_ids_with_affiliation.tsv")
bac_id_affi <- read.delim("./data/prok_tax_from_silva.tsv")
```

5 Add the taxonomic classifications to the network and then send network to Cytoscape

In preparation for sending the networks to Cytoscape we will add in the taxonomic data. Some of the organisms do not have taxonomic classifications associated with them so we have described them as "not_class" for not classified. We do that because we have had problems sending "NA"s to Cytoscape from RCy3.

```r
genenet.nodes <- as.data.frame(vertex.attributes(graph_vir_prok))

## not all have classification, so create empty columns
genenet.nodes$phage_aff <- rep("not_class", nrow(genenet.nodes))
genenet.nodes$Tax_order <- rep("not_class", nrow(genenet.nodes))
genenet.nodes$Tax_subfamily <- rep("not_class", nrow(genenet.nodes))

for (row in seq_along(1:nrow(genenet.nodes))){
  if (genenet.nodes$name[row] %in% phage_id_affiliation$first_sheet.Phage_id_network){
    id_name <- as.character(genenet.nodes$name[row])
    aff_to_add <- unique(subset(phage_id_affiliation,
                                first_sheet.Phage_id_network == id_name,
                                select = c(phage_affiliation,
                                           Tax_order,
                                           Tax_subfamily)))
    genenet.nodes$phage_aff[row] <- as.character(aff_to_add$phage_affiliation)
    genenet.nodes$Tax_order[row] <- as.character(aff_to_add$Tax_order)
    genenet.nodes$Tax_subfamily[row] <- as.character(aff_to_add$Tax_subfamily)
  }
}
```

```r
## do the same for proks
genenet.nodes$prok_king <- rep("not_class", nrow(genenet.nodes))
genenet.nodes$prok_tax_phylum <- rep("not_class", nrow(genenet.nodes))
genenet.nodes$prok_tax_class <- rep("not_class", nrow(genenet.nodes))

for (row in seq_along(1:nrow(genenet.nodes))){
  if (genenet.nodes$name[row] %in% bac_id_affi$Accession_ID){
    aff_to_add <- unique(subset(bac_id_affi,
                                Accession_ID == as.character(genenet.nodes$name[row]),
                                select = c(Kingdom,
                                           Phylum,
                                           Class)))

    genenet.nodes$prok_king[row] <- as.character(aff_to_add$Kingdom)
    genenet.nodes$prok_tax_phylum[row] <- as.character(aff_to_add$Phylum)
    genenet.nodes$prok_tax_class[row] <- as.character(aff_to_add$Class)
  }
}
```

Add to the network the data related to the connections between the organisms, the edge data, and then prepare to send the nodes and edges to Cytoscape using the function `cyPlot()`.

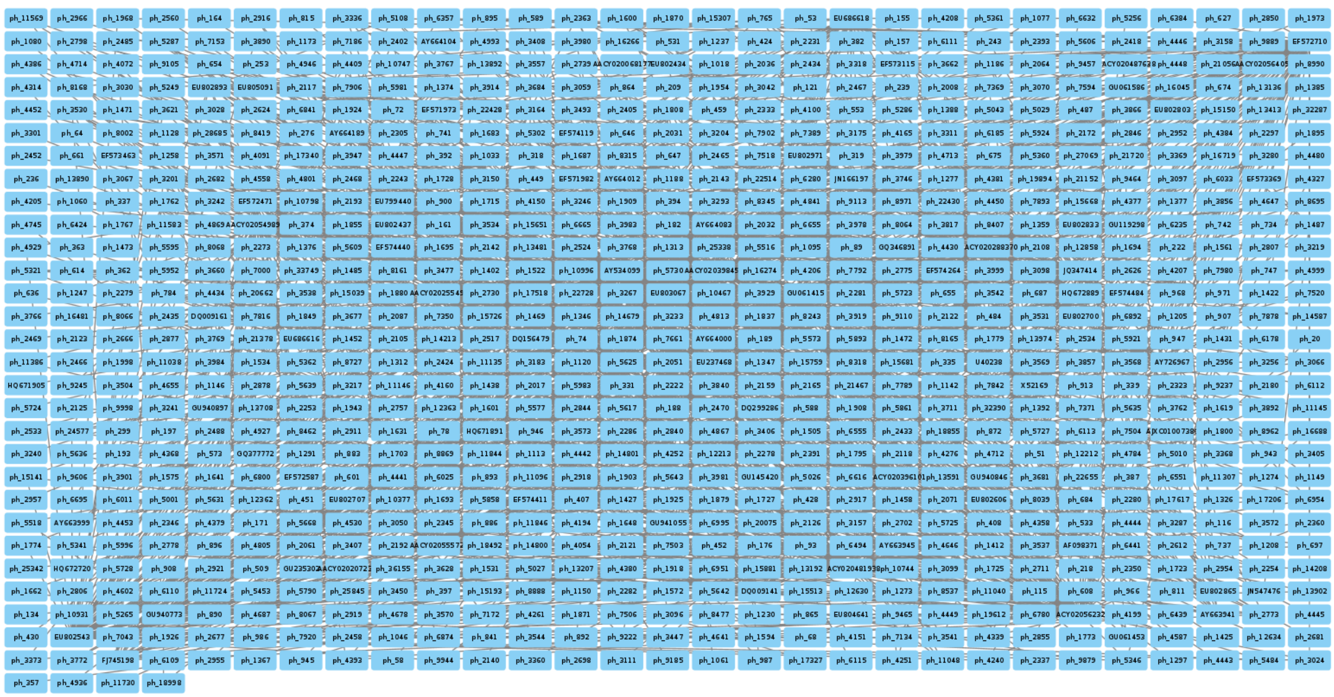
```r
genenet.edges <- data.frame(as_edgelist(graph_vir_prok))
names(genenet.edges) <- c("name.1",
                          "name.2")
genenet.edges$Weight <- edge_attr(graph_vir_prok)[[1]]

genenet.edges$name.1 <- as.character(genenet.edges$name.1)
genenet.edges$name.2 <- as.character(genenet.edges$name.2)
genenet.nodes$name <- as.character(genenet.nodes$name)

ug <- cyPlot(genenet.nodes,genenet.edges)
```

6 Send network to Cytoscape using RCy3

Now we will send the network from R to Cytoscape.

```r
cw <- CytoscapeWindow("Tara oceans",
                      graph = ug,
                      overwriteWindow = TRUE)
```

```r
displayGraph(cw)
layoutNetwork(cw)
fitContent(cw)
```



7 Colour network by prokaryotic phylum

We would like to get an overview of the different phylum of bacteria that are in the network. One way is to colour the different nodes based on their phylum classification. The package Rcolorbrewer will be used to generate a set of good colours for the nodes.

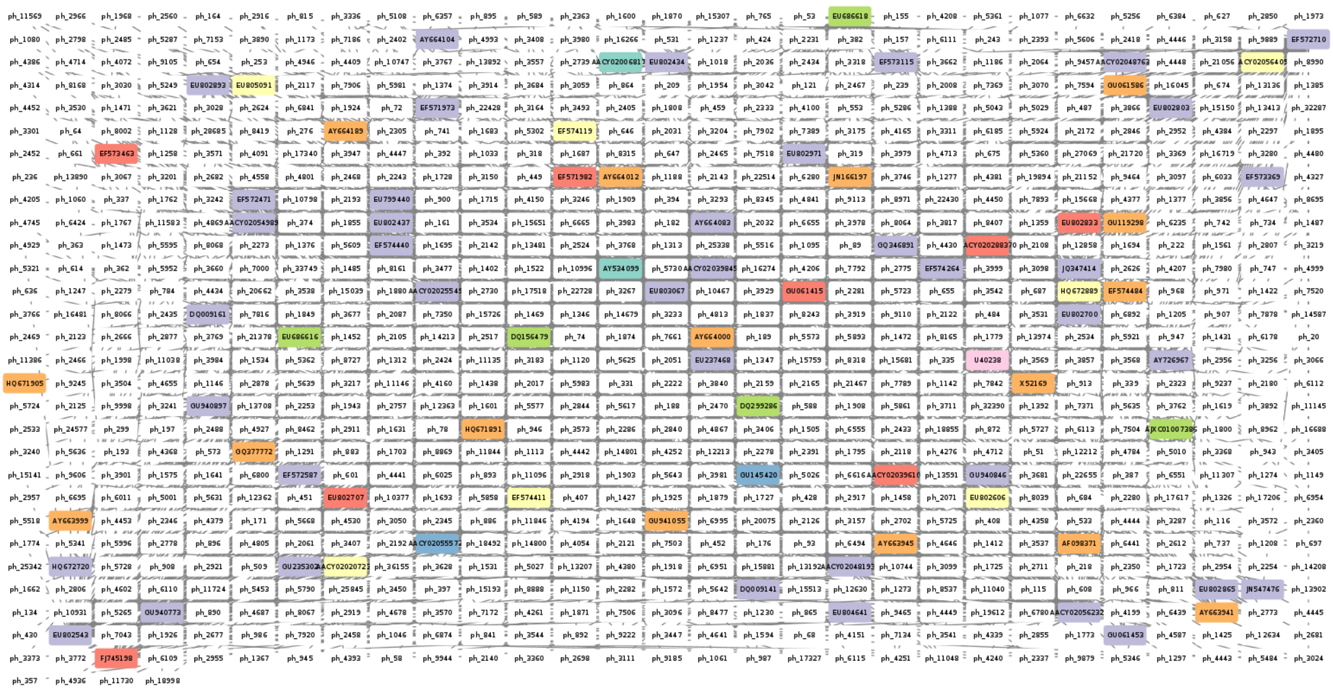
```
families_to_colour <- unique(genenet.nodes$prok_tax_phylum)
families_to_colour <- families_to_colour[!families_to_colour %in% "not_class"]
node.colour <- brewer.pal(length(families_to_colour),
                          "Set3")
```

Use the colours from Rcolorbrewer to colour the nodes in Cytoscape.

```
setNodeColorRule(cw,
                 "prok_tax_phylum",
                 families_to_colour,
                 node.colour,
                 "lookup",
                 default.color = "#ffffff")
```

```
## Successfully set rule.
```

```
displayGraph(cw)
layoutNetwork(cw)
fitContent(cw)
```



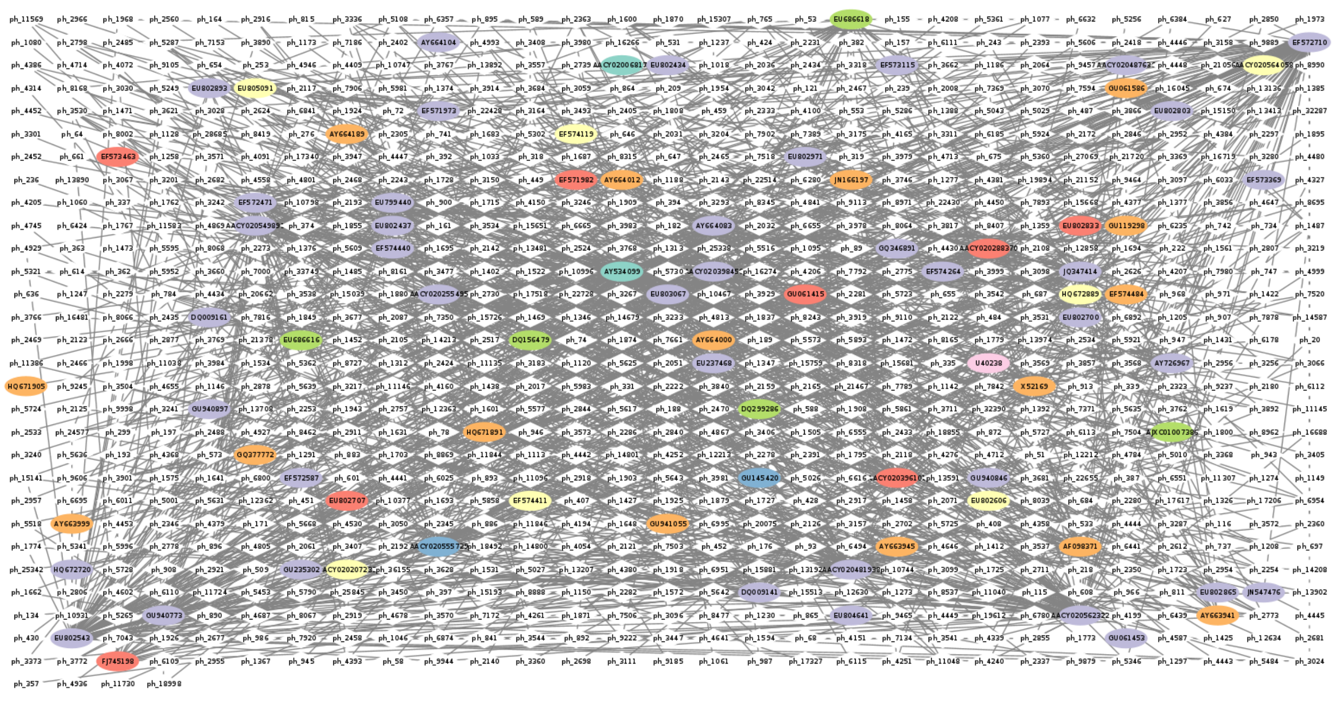
7.1 Set node shape to reflect virus or prokaryote

Next we would like to change the shape of the node to reflect whether the nodes are viral or prokaryotic in origin. In this dataset all of the viral node names start with "ph_", thus we can set the viral nodes to be diamond-shaped by looking for all the nodes that start with "ph" in the network.

```
shapes_for_nodes <- c("DIAMOND")

phage_names <- grep("ph_",
                    genenet.nodes$name,
                    value = TRUE)
setNodeShapeRule(cw,
                 "label",
                 phage_names,
                 shapes_for_nodes)
```

```
## Successfully set rule.
```

```
displayGraph(cw)
fitContent(cw)
```



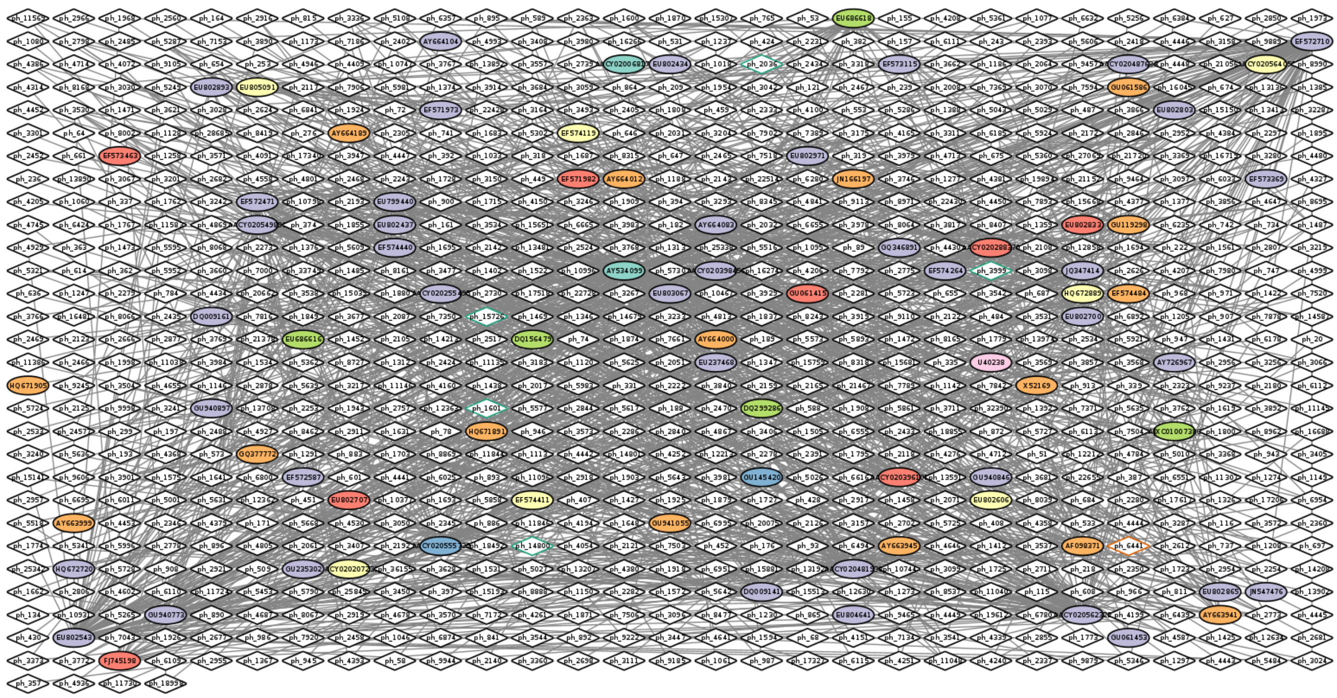
8 Colour edges of phage nodes

The classification of the viral data was done in a very conservative manner so not many of the viral nodes were identified. However, if we do want to add some of this information to our visualization we can colour the edges of the viral nodes by family. The main families that were identified in this dataset are the *Podoviridae*, the *Siphoviridae* and the *Myoviridae* (for more info see NCBI Podoviridae, NCBI Myoviridae, and NCBI Siphoviridae)

```
setDefaultNodeBorderWidth(cw, 5)
families_to_colour <- c(" Podoviridae",
                        " Siphoviridae",
                        " Myoviridae")
node.colour <- brewer.pal(length(families_to_colour),
                          "Dark2")
setNodeBorderColorRule(cw,
                       "Tax_subfamily",
                       families_to_colour,
                       node.colour,
                       "lookup",
                       default.color = "#000000")
```

```
## Successfully set rule.
```

```
displayGraph(cw)
fitContent(cw)
```



9 Do layout to minimize overlap of nodes.

After doing all of this colouring to the network we would like to layout the network in a way that allows us to more easily see which nodes are connected without overlap. To do the layout we will use the Cytoscape plugin Allegro.

When using RCy3 to drive Cytoscape, if we are not sure what the current values are for a layout or we are not sure what kinds of values are accepted for the different parameters of our layout, we can investigate using the RCy3 functions `getLayoutPropertyNames()` and then `getLayoutPropertyValue()`.

```
getLayoutNames(cw)
```

```
##  [1] "attribute-circle"
##  [2] "allegro-weak-clustering"
##  [3] "allegro-edge-repulsive-fruchterman-reingold"
##  [4] "stacked-node-layout"
##  [5] "allegro-edge-repulsive-strong-clustering"
##  [6] "allegro-strong-clustering"
##  [7] "degree-circle"
##  [8] "allegro-fruchterman-reingold"
##  [9] "allegro-edge-repulsive-spring-electric"
## [10] "circular"
## [11] "attributes-layout"
## [12] "kamada-kawai"
## [13] "force-directed"
## [14] "allegro-edge-repulsive-weak-clustering"
## [15] "grid"
## [16] "hierarchical"
## [17] "allegro-spring-electric"
## [18] "fruchterman-rheingold"
## [19] "isom"
```

```
getLayoutPropertyNames(cw, layout.name = "allegro-spring-electric")
```

```
## [1] "randomize"                    "maxIterations"
## [3] "noOverlapIterations"          "deviceSelection"
## [5] "componentProcessingSelection" "componentSorting"
## [7] "scale"                        "gravityTypeSelection"
## [9] "gravity"
```

```
getLayoutPropertyValue(cw, "allegro-spring-electric","gravity")
```

```
## [[1]]
## [1] 100
```

```
getLayoutPropertyValue(cw, "allegro-spring-electric","maxIterations")
```

```
## [[1]]
## [1] 2000
```

```r
getLayoutPropertyValue(cw, "allegro-spring-electric","noOverlapIterations")
```

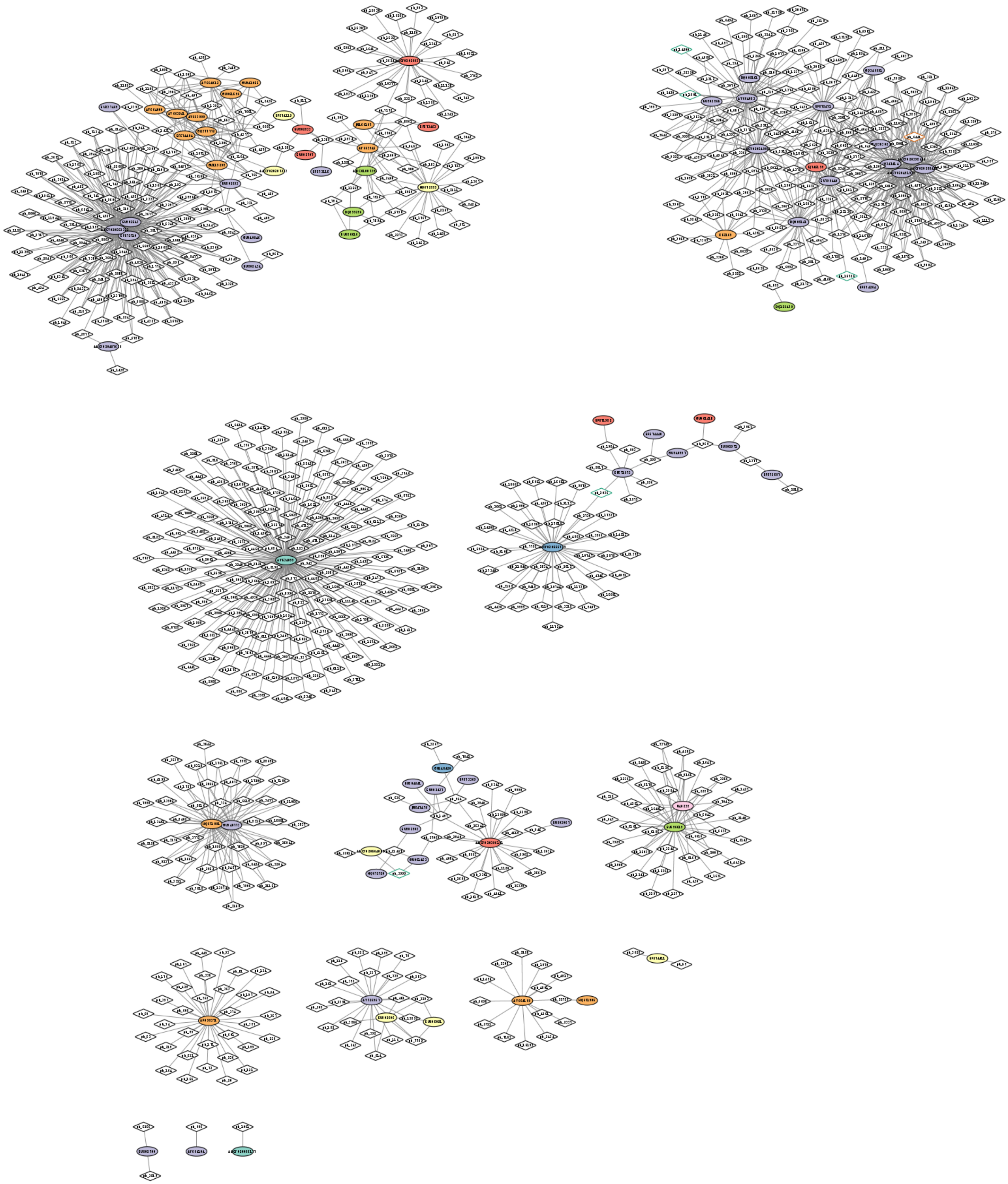
```
## [[1]]
## [1] TRUE
```

Once we decide on the properties we want, we can go ahead and set them like this:

```r
setLayoutProperties(cw,
                    layout.name = "allegro-spring-electric",
                    list(gravity = 100,
                         scale = 6))
```

```
## Successfully updated the property 'gravity'.
## Successfully updated the property 'scale'.
```

```r
layoutNetwork(cw,
              layout.name = "allegro-spring-electric")
fitContent(cw)
```



10 Look at network properties

One thing that might be interesting to visualize is nodes that are connected to many different nodes and nodes that are connected to few other nodes. The number of other nodes to which one node is connected is called degree. We can use a gradient of size to quickly visualize nodes that have high degree.

```
## initiate a new node attribute
ug2 <- initNodeAttribute(graph = ug,
                         "degree",
                         "numeric",
                         0.0)

## degree from graph package for undirected graphs not working well,
## so instead using igraph to calculate this from the original graph
nodeData(ug2, nodes(ug2), "degree") <- degree(graph_vir_prok)

cw2 <- CytoscapeWindow("Tara oceans with degree",
                      graph = ug2,
                      overwriteWindow = TRUE)
```

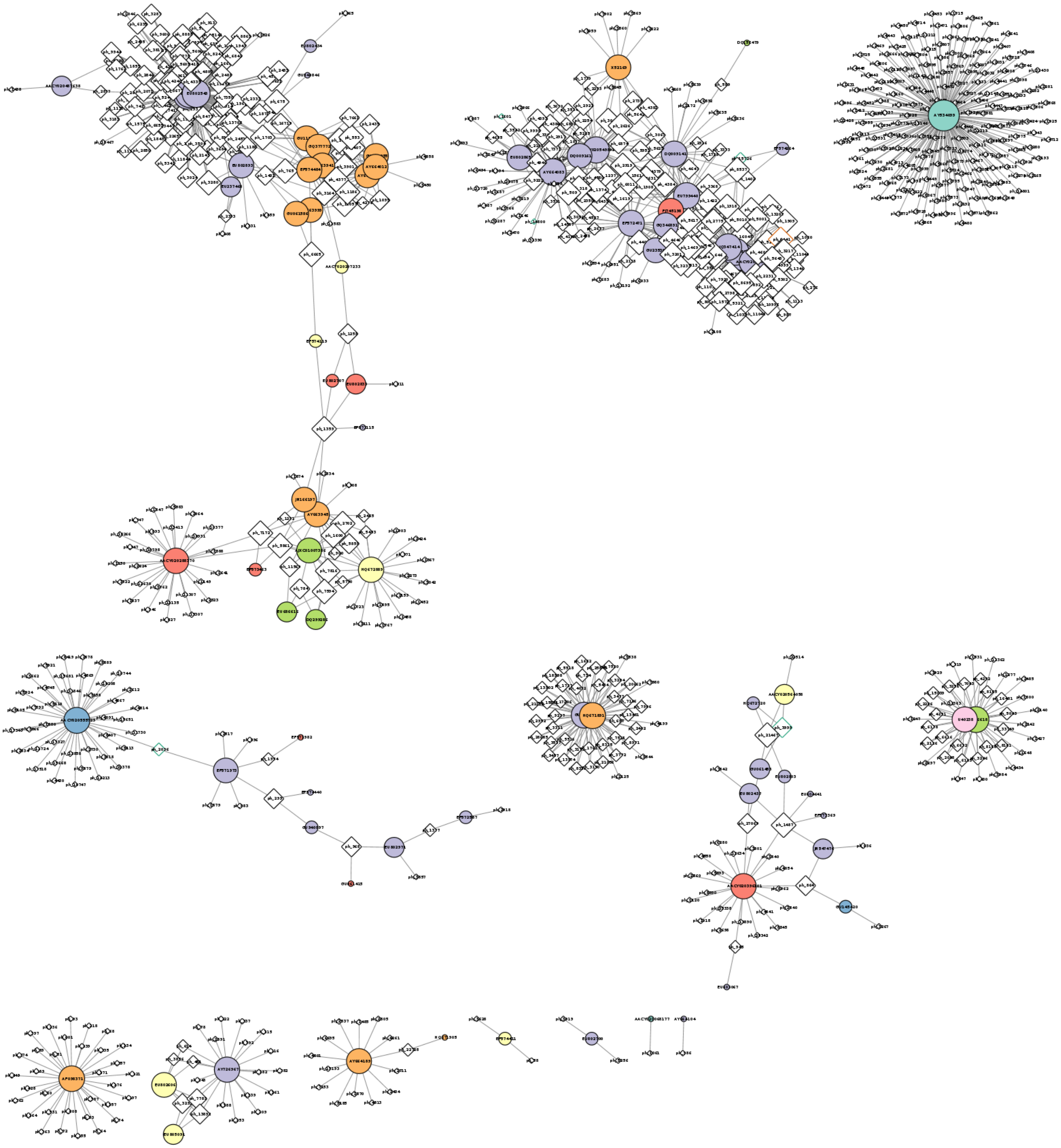
```
displayGraph(cw2)
layoutNetwork(cw2)
```

11 Size by degree

```
degree_control_points <- c(min(degree(graph_vir_prok)),
                           mean(degree(graph_vir_prok)),
                           max(degree(graph_vir_prok)))
node_sizes <- c(20,
                20,
                80,
                100,
                110) # number of control points in interpolation mode,
                     # the first and the last are for sizes "below" and "above" the attribute seen.

setNodeSizeRule(cw2,
                "degree",
                degree_control_points,
                node_sizes,
                mode = "interpolate")
```

```
## Locked node dimensions successfully even if the check box is not ticked.
## Locked node dimensions successfully even if the check box is not ticked.
## Successfully set rule.
```

```
layoutNetwork(cw2,
              "force-directed")
```



12 Select an interesting node and make a subnetwork from it

The visualization displays several different areas where there are highly connected nodes that are in the same bacterial phylum. We will select one of these nodes, all of the nodes connected to this node, its first neighbours, and then the nodes connected to the first neighbours. One node that is in a group of highly connected nodes is the cyanobacterial node “GQ377772”. We will select it and its first and second neighbours and then make a new network from these nodes and their connections.

```
selectNodes(cw2,
            "GQ377772") # selects specific nodes
getSelectedNodes(cw2)
```

```
## [1] "GQ377772"
```

```
selectFirstNeighborsOfSelectedNodes(cw2)
getSelectedNodes(cw2)
```

```
##  [1] "ph_3164"  "ph_1392"  "ph_1808"  "ph_3901"  "ph_407"   "ph_4377"
##  [7] "ph_553"   "ph_765"   "ph_7661"  "GQ377772"
```

Now select the second neighbours of node “GQ377772”.

```
selectFirstNeighborsOfSelectedNodes(cw2)
getSelectedNodes(cw2)
```

```
##  [1] "ph_3164"       "ph_1392"       "ph_1808"       "ph_3901"
##  [5] "ph_407"        "ph_4377"       "ph_553"        "ph_765"
##  [9] "ph_7661"       "AACY020207233" "AY663941"      "AY663999"
## [13] "AY664000"      "AY664012"      "EF574484"      "EU802893"
## [17] "GQ377772"      "GU061586"      "GU119298"      "GU941055"
```

This has only selected the nodes, but not the edges in Cytoscape, so we will need to select all of the edges before we make the new subnetwork.

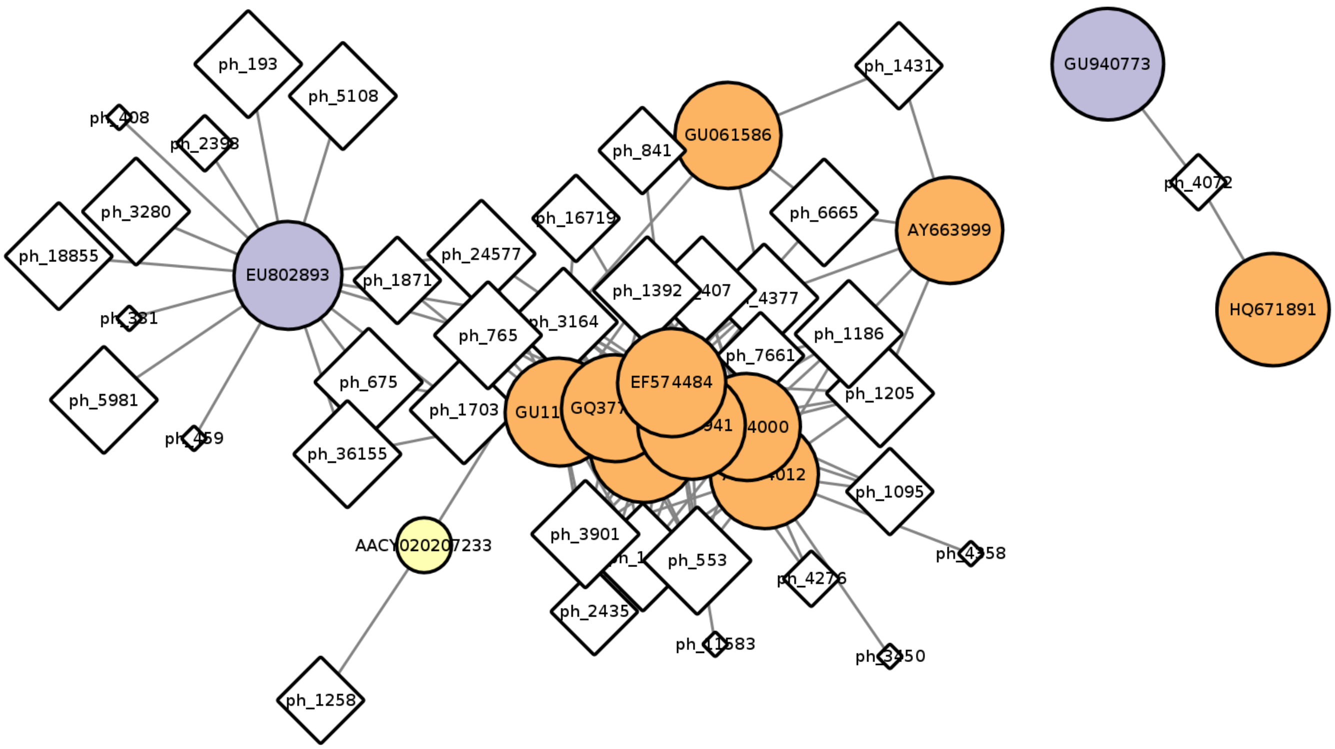
```
source("./functions_to_add_to_RCy3/subnetwork_stuff.R")
selectEdgesConnectedBySelectedNodes(cw2)
```

Now edges should appear selected in Cytoscape, in addition to the nodes.

```
newnet <- subnetwork_from_selected(cw2)
```

```
## [1] "Cytoscape window Tara oceans with degree(1) successfully connected to R session and graph copie
```

```
layoutNetwork(newnet, "force-directed")
```


ph_193
ph_5108
ph_408
ph_2398
ph_3280
ph_18855
EU802893
ph_381
ph_5981
ph_459
ph_1871
ph_24577
ph_16719
ph_841
GU061586
ph_1431
ph_6665
AY663999
GU940773
ph_4072
HQ671891
ph_1392
ph_3164
ph_765
ph_675
ph_1703
ph_36155
ph_7661
ph_1186
ph_1205
EF574484
ph_1095
ph_4358
ph_4276
ph_3450
ph_553
ph_3901
ph_2435
AACY020207233
ph_1258

12.1 Conclusion

This has been a very basic introduction to exploring co-occurence networks in Cytoscape using RCy3.

References

Brum, Jennifer R., J. Cesar Ignacio-Espinoza, Simon Roux, Guilhem Doulcier, Silvia G. Acinas, Adriana Alberti, Samuel Chaffron, et al. 2015. "Patterns and Ecological Drivers of Ocean Viral Communities." *Science* 348 (6237): 1261498. http://www.sciencemag.org/content/348/6237/1261498.short.

Csardi, Gabor, and Tamas Nepusz. 2006. "The Igraph Software Package for Complex Network Research." *InterJournal* Complex Systems: 1695. http://igraph.org.

Gentleman, R., Elizabeth Whalen, W. Huber, and S. Falcon. 2016. *Graph: Graph: A Package to Handle Graph Data Structures*.

Lima-Mendez, Gipsi, Karoline Faust, Nicolas Henry, Johan Decelle, Sébastien Colin, Fabrizio Carcillo, Samuel Chaffron, et al. 2015. "Determinants of Community Structure in the Global Plankton Interactome." *Science* 348 (6237). doi:10.1126/science.1262073.

Ono, K, T Muetze, G Kolishovski, P Shannon, and B Demchak. 2015. "CyREST: Turbocharging Cytoscape Access for External Tools via a RESTful API [Version 1; Referees: 2 Approved]." *F1000Research* 4 (478). doi:10.12688/f1000research.6767.1.

Shannon, Paul T., Mark Grimes, Burak Kutlu, Jan J. Bot, and David J. Galas. 2013. "RCytoscape: Tools for Exploratory Network Analysis." *BMC Bioinformatics* 14 (1): 1–9. doi:10.1186/1471-2105-14-217.

Shannon, Paul, Andrew Markiel, Owen Ozier, Nitin S. Baliga, Jonathan T. Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. 2003. "Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks." *Genome Research* 13 (11): 2498–2504. http://genome.cshlp.org/content/13/11/2498.short.

Sunagawa, Shinichi, Luis Pedro Coelho, Samuel Chaffron, Jens Roat Kultima, Karine Labadie, Guillem Salazar, Bardya Djahanschiri, et al. 2015. "Structure and Function of the Global Ocean Microbiome." *Science* 348 (6237): 1261359. http://www.sciencemag.org/content/348/6237/1261359.short.

Venter, J. Craig, Karin Remington, John F. Heidelberg, Aaron L. Halpern, Doug Rusch, Jonathan A. Eisen, Dongying Wu, et al. 2004. "Environmental Genome Shotgun Sequencing of the Sargasso Sea." *Science* 304 (5667): 66–74. doi:10.1126/science.1093857.

Weiss, Sophie, Will Van Treuren, Catherine Lozupone, Karoline Faust, Jonathan Friedman, Ye Deng, Li Charlie Xia, et al. 2016. "Correlation Detection Strategies in Microbial Data Sets Vary Widely in Sensitivity and Precision." *ISME J* 10 (7): 1669–81. http://dx.doi.org/10.1038/ismej.2015.235.