An introduction to network inference and mining - TP

Nathalie Villa-Vialaneix - nathalie.villa@toulouse.inra.fr
http://www.nathalievilla.org

INRA, UR 0875 MIAT

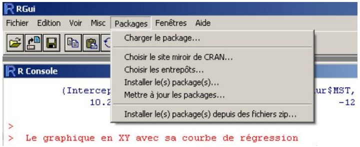
Formation INRA, Niveau 3

Packages in R

R is provided with basic functions but more than 3,000 packages are available on the CRAN (Comprehensive R Archive Network) for additionnal functions (see also the project Bioconductor).

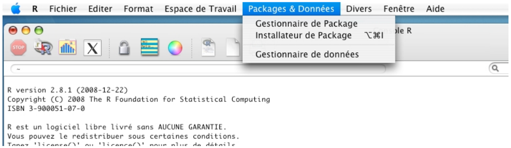
- **Installing new packages** (has to be done only once) with the command line or the menu (Windows or Mac OS X or RStudio)

```
install.packages(c("huge","igraph",
                   "mixOmics"))
```


RGui
Fichier
Edition
Voir
Misc
Packages
Fenêtres
Aide
Charger le package...
Choisir le site miroir de CRAN...
Choisir les entrepôts...
Installer le(s) package(s)...
Mettre à jour les packages...
Installer le(s) package(s) depuis des fichiers zip...
R Console

```
     (Intercep                      ur$MST,
        10.2                          -12

>
>  Le graphique en XY avec sa courbe de régression
```



```
R version 2.8.1 (2008-12-22)
Copyright (C) 2008 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails
```

- **Loading a package** (has to be done each time R is re-started)

```
library(igraph)
```

- Be sure you properly set the working directory (directory in which you put the data files) before you start

Outline

Outline

Use case description

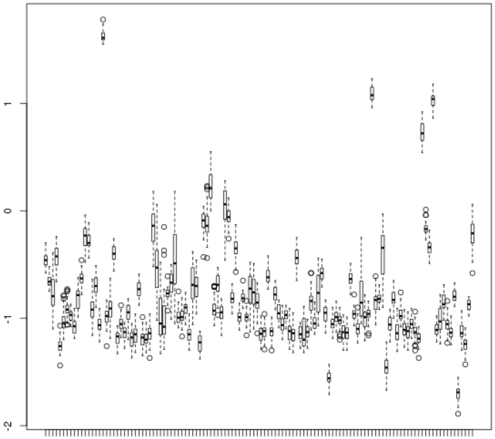
Data in the R package mixOmics

microarray data: expression of 120 selected genes potentially involved in nutritional problems on 40 mice. These data come from a nutrigenomic study [Martin et al., 2007].

```
data(nutrimouse)
summary(nutrimouse)
expr <- nutrimouse$gene
rownames(expr) <- paste(1:nrow(expr),
                       nutrimouse$genotype,
                       nutrimouse$diet,
                       sep="-")
```

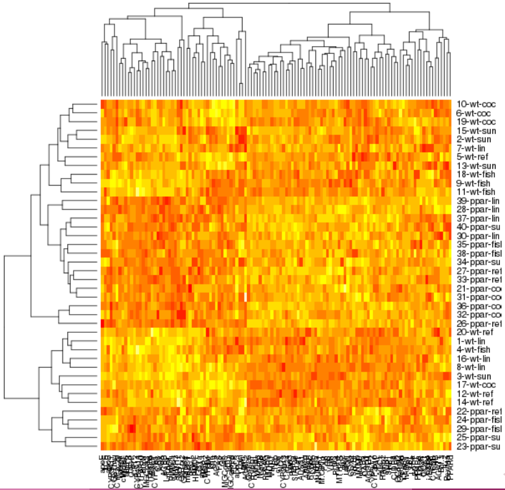
Data distribution

```
boxplot(expr, names=NULL)
```



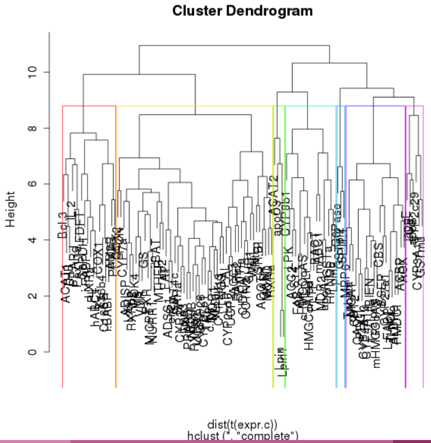
Gene correlations and clustering

```
expr.c <- scale(expr)
heatmap(as.matrix(expr.c))
```



Hierarchical clustering from raw data

```
hclust.tree <- hclust(dist(t(expr)))
plot(hclust.tree)
rect.hclust(hclust.tree, k=7, border=rainbow(7))
```



Save gene clusters

```
hclust.groups <- cutree(hclust.tree, k=7)
table(hclust.groups)
hclust.groups
# 1  2  3  4  5  6  7
# 18  4 20 52 17  6  3
```

Sparse linear regression by Maximum Likelihood

Estimation: [Friedman et al., 2008] Gaussien framework allows us to use ML optimization **with a sparse penalization**

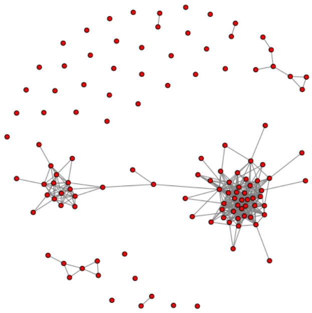
$$\mathcal{L}(S|X) + pen = \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \log \mathbb{P}(X_i^j | X_i^{-j}, \ S_j) \right) - \lambda \|S\|_1$$

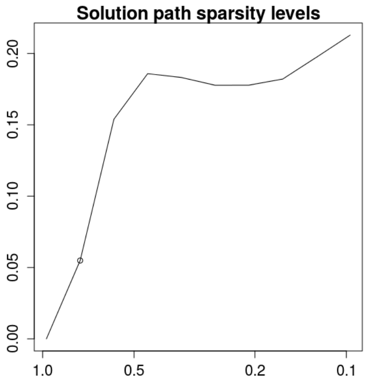
```
glasso.res <- huge(as.matrix(expr), method="glasso"
glasso.res
# Model: graphical lasso (glasso)
# Input: The Data Matrix
# Path length: 10
# Graph dimension: 120
# Sparsity level: 0 -----> 0.2128852
```

estimates of the concentration matrix S are in `glasso.res$icov[[1]]`, ..., `glasso.res$icov[[10]]`, each one corresponding to a different λ

Select λ for a targeted density with the StARS method [Liu et al., 2010]

```
glasso.sel <- huge.select(glasso.res,
                          criterion="stars")
plot(glasso.sel)
```





Using igraph to create the graph

From the binary adjacency matrix:

```
bin.mat <- as.matrix(glasso.sel$opt.icov)!=0
colnames(bin.mat) <- colnames(expr)
```

Create an undirected simple graph from the matrix:

```
nutrimouse.net <- simplify(graph.adjacency(bin.mat,
  mode="max"))
nutrimouse.net
# IGRAPH UN-- 120 392 --
#   + attr: name (v/c)
```

Connected components

The resulting network is not connected:

```
is.connected(nutrimouse.net)
# [1] FALSE
```

Connected components are extracted by:

```
components.nutrimouse <- clusters(nutrimouse.net)
components.nutrimouse
# [1]   1  2  3  2  2  4  5  2  2  2  6  2  7  2
# ...
#
# $csize
# [1]  7 67  1  6  2  1  1  1  1  1  1  1  1  2
# ...
#
# $no
# [1] 40
```

Working on a connected subgraph

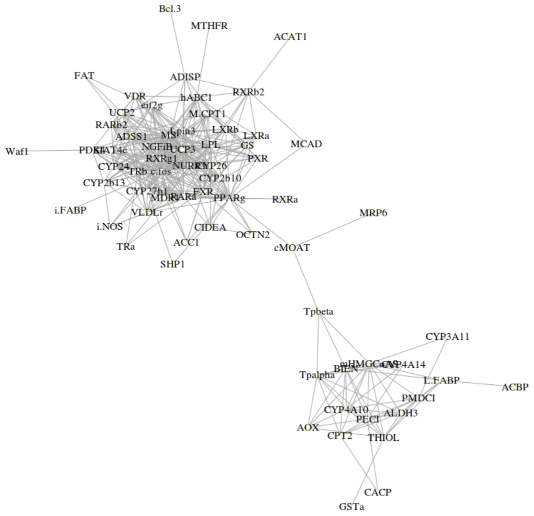
The largest connected component (with 99 nodes) can be extracted with:

```
nutrimouse.lcc <- induced.subgraph(nutrimouse.net,
  components.nutrimouse$membership==
    which.max(components.nutrimouse$csize))
nutrimouse.lcc
# IGRAPH UN-- 67 375 --
#    + attr: name (v/c)
```

and visualized with:

```
nutrimouse.lcc$layout <- layout.kamada.kawai(
  nutrimouse.lcc)
plot(nutrimouse.lcc, vertex.size=2,
     vertex.color="lightyellow",
     vertex.frame.color="lightyellow",
     vertex.label.color="black",
     vertex.label.cex=0.7)
```

Resulting network



Node clustering (compared to raw clustering)

Using one of the clustering function available in igraph:

```
set.seed(1219)
clusters.nutrimouse <- spinglass.community(
  nutrimouse.lcc)
clusters.nutrimouse
        table(clusters.nutrimouse$membership)
# 1  2  3  4
# 15 25 17 10
```

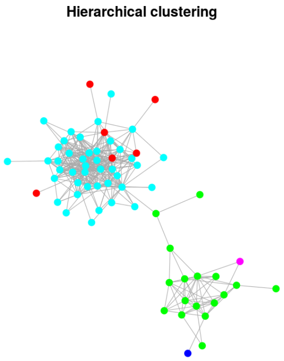
The hierarchical clustering for nodes in the largest connected component is obtained with:

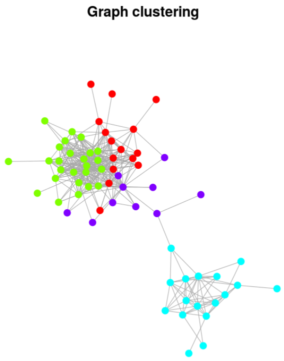
```
induced.hclust.groups <- hclust.groups[
  components.nutrimouse$membership==
    which.max(components.nutrimouse$csize)]
table(induced.hclust.groups)
# 1  3  4  5  6
# 6 17 42  1  1
```

Visual comparison

```
par(mfrow=c(1,2))
par(mar=rep(1,4))
plot(nutrimouse.lcc, vertex.size=5,
     vertex.color=rainbow(6)[induced.hclust.groups]
     vertex.frame.color=
       rainbow(6)[induced.hclust.groups],
       vertex.label=NA,
     main="Hierarchical clustering")
plot(nutrimouse.lcc, vertex.size=5,
     vertex.color=rainbow(4)[
       clusters.nutrimouse$membership],
     vertex.frame.color=rainbow(4)[
       clusters.nutrimouse$membership],
     vertex.label=NA, main="Graph clustering")
```

Visual comparison





Numeric comparison

```
compare.communities(induced.hclust.groups,
                    clusters.nutrimouse$membership,
                    method="nmi")
# [1] 0.5091568
modularity(nutrimouse.lcc, induced.hclust.groups)
# [1] 0.2368462
```

Outline

Use case description

Data are Natty's facebook network[1]

- fbnet-el.txt is the edge list;
- fbnet-name.txt are the nodes' initials.

```
edgelist <- as.matrix(read.table("fbnet-el.txt"))
vnames <- read.table("fbnet-name.txt")
vnames <- as.character(vnames[,1])
```

The graph is built with:

```
# with 'graph.edgelist'
fbnet0 <- graph.edgelist(edgelist, directed=FALSE)
fbnet0
# IGRAPH U--- 152 551 --
```

See also help(graph.edgelist) for more graph constructors

[1]Do it with yours: http://shiny.nathalievilla.org/fbs

Vertexes, vertex attributes

The graph's vertexes are accessed and counted with:

```
V(fbnet0)
vcount(fbnet0)
```

Vertexes can be described by attributes:

```
# add an attribute for vertices
V(fbnet0)$initials <- vnames
fbnet0
# IGRAPH U--- 152 551 --
# + attr: initials (v/x)
```

Edges, edge attributes

The graph's edges are accessed and counted with:

```
E(fbnet0)
# [1] 11 --  1
# [2] 41 --  1
# [3] 52 --  1
# [4] 69 --  1
# [5] 74 --  1
# [6] 75 --  1
# ...
ecount(fbnet0)
# 551
```

igraph can also handle edge attributes (and also graph attributes).

Connected components

```
is.connected(fbnet0)
# [1] FALSE
\end{Rcode}
As this network is not connected, the connected com
\begin{Rcode}
fb.components <- clusters(fbnet0)
names(fb.components)
# [1] "membership" "csize"      "no"
head(fb.components$membership, 10)
# [1] 1 1 2 2 1 1 1 1 3 1
fb.components$csize
# [1] 122   5   1   1   2   1   1   1   2   1
# [11]   1   2   1   1   2   3   1   1   1   1
# [21]   1
fb.components$no
# [1] 21
```

Largest connected component

```
fbnet.lcc <- induced.subgraph(fbnet0,
                fb.components$membership==
                which.max(fb.components$csize))
# main characteristics of the LCC
fbnet.lcc
# IGRAPH U--- 122 535 --
# + attr: initials (v/x)
is.connected(fbnet.lcc)
# [1] TRUE
```

and global characteristics

```
graph.density(fbnet.lcc)
# [1] 0.0724834
transitivity(fbnet.lcc)
# [1] 0.5604524
```

Network visualization

Different layouts are implemented in igraph to visualize the graph:

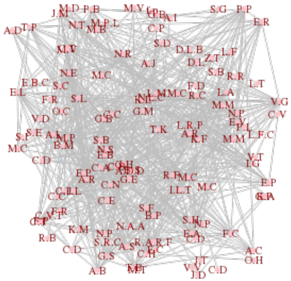
```
plot(fbnet.lcc, layout=layout.random,
     main="random layout", vertex.size=3,
     vertex.color="pink", vertex.frame.color="pink"
     vertex.label.color="darkred",
     edge.color="grey",
     vertex.label=V(fbnet.lcc)$initials)
```

Try also `layout.circle`, `layout.kamada.kawai`, `layout.fruchterman.reingold`... Network are generated with some randomness. See also `help(igraph.plotting)` for more information on network visualization

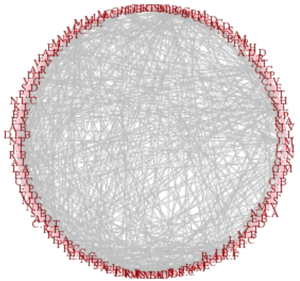
igraph integrates a pre-defined graph attribute `layout`:

```
V(fbnet.lcc)$label <- V(fbnet.lcc)$initials
plot(fbnet.lcc)
```

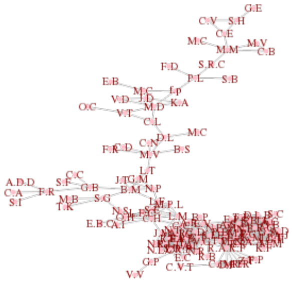
random layout



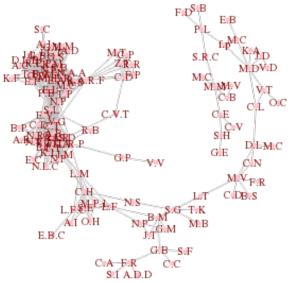
circular layout



Kamada Kawai layout



Fruchterman & Reingold layout



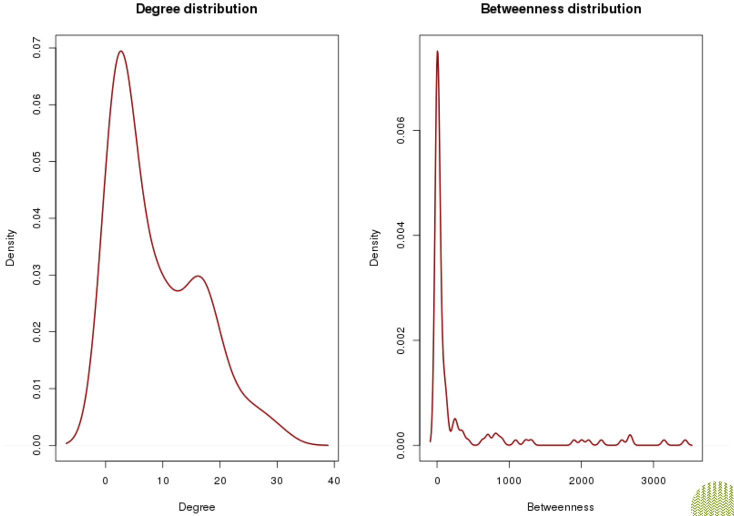
Degree and betweenness

```
fbnet.degrees <- degree(fbnet.lcc)
summary(fbnet.degrees)
#     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#     1.00     2.00     6.00     8.77    15.00    31.00
fbnet.between <- betweenness(fbnet.lcc)
summary(fbnet.between)
#     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#     0.00     0.00    14.03   301.70   123.10  3439.00
```

and their distributions:

```
par(mfrow=c(1,2))
plot(density(fbnet.degrees), lwd=2,
     main="Degree distribution", xlab="Degree",
     ylab="Density")
plot(density(fbnet.between), lwd=2,
     main="Betweenness distribution",
     xlab="Betweenness", ylab="Density")
```

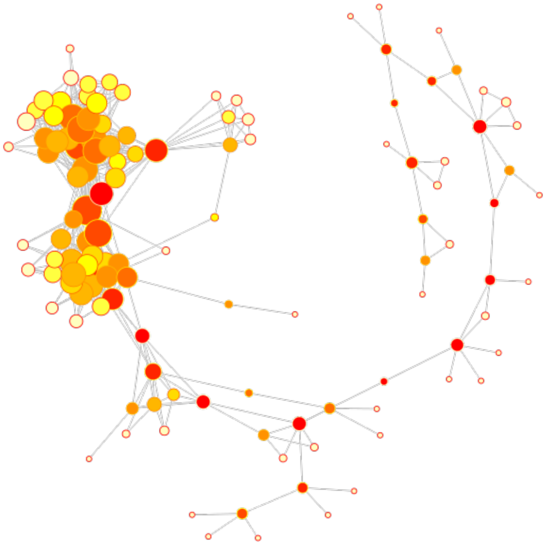
Degree and betweenness distribution



Combine visualization and individual characteristics

```r
par(mar=rep(1,4))
# set node attribute ‘size’ with degree
V(fbnet.lcc)$size <- 2*sqrt(fbnet.degrees)
# set node attribute ‘color’ with betweenness
bet.col <- cut(log(fbnet.between+1),10,
               labels=FALSE)
V(fbnet.lcc)$color <- heat.colors(10)[11-bet.col]
plot(fbnet.lcc, main="Degree and betweenness",
     vertex.frame.color=heat.colors(10)[bet.col],
     vertex.label=NA, edge.color="grey")
```

Degree and betweenness



Node clustering

One of the function to perform node clustering is spinglass.community (that prossibly produces different results each time it is used since it is based on a stochastic process):

```
fbnet.clusters <- spinglass.community(fbnet.lcc)
fbnet.clusters
# Graph community structure calculated with
#   the spinglass algorithm
# Number of communities: 9
# Modularity: 0.5654136
# Membership vector:
#   [1] 9 5 6 5 5 9 1 9 1 1 4 9 9 6 2 7 2 7 2 7 5
#  ...
table(fbnet.clusters$membership)
#  1  2  3  4  5  6  7  8  9
#  8  7  8 32 14  7 18  2 26
```

See help(communities) for more information.

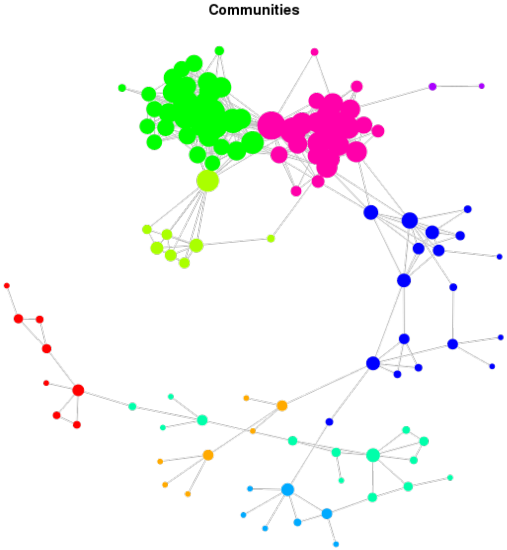
Combine clustering and visualization

```
# create a new attribute
V(fbnet.lcc)$community <- fbnet.clusters$membershi
fbnet.lcc
# IGRAPH U--- 122 535 --
#   + attr: layout (g/n), initials (v/c),
# label (v/c), size (v/n), color (v/c),
# community (v/n)
```

Display the clustering:

```
par(mfrow=c(1,1))
par(mar=rep(1,4))
plot(fbnet.lcc, main="Communities",
     vertex.frame.color=
       rainbow(9)[fbnet.clusters$membership],
     vertex.color=
       rainbow(9)[fbnet.clusters$membership],
     vertex.label=NA, edge.color="grey")
```

Combine clustering and visualization



Export the graph

The `graphml` format can be used to export the graph (it can be read by most of the graph visualization programs and includes information on node and edge attributes)

```
write.graph(fbnet.lcc, file="fblcc.graphml",
            format="graphml")
```

see `help(write.graph)` for more information of graph exportation formats

Outline

Import data

Start $\mathcal{G}\varphi$ and select “new project”.

- **Create a graph from an edge list**: file `fbnet-el.txt`

File / Open

Graph type: “undirected”

Import data

Start and select "new project".

- **Create a graph from an edge list**: file `fbnet-el.txt`

  File / Open

  Graph type: "undirected"

- **Create a graph (with nodes an edges attributes) from a GraphML file**: file `fblcc.graphml` previously created with igraph

  File / Open

  Graph type: "undirected"

On the right part of the screen, the number of nodes and edges are indicated.

Import data

Start and select "new project".

- **Create a graph from an edge list**: file `fbnet-el.txt`

  File / Open

  Graph type: "undirected"

- **Create a graph (with nodes an edges attributes) from a GraphML file**: file `fblcc.graphml` previously created with igraph

  File / Open

  Graph type: "undirected"

  On the right part of the screen, the number of nodes and edges are indicated.

- **Check nodes attributes** and define nodes labels

  Data lab

  Copy data to a column: initials

  Copy to: Label

  Other nodes or edges attributes can be imported from a csv file with

  import file

Visualization

- **Visualize the graph with Fruchterman & Reingold algorithm**

Bottom left of panel “Global view”
Spatialization: Choose “Fruchterman and Reingold”
Area: 800 - Gravity: 1

Click on “Stop” when stabilized

Visualization

- **Visualize the graph with Fruchterman & Reingold algorithm**

  Bottom left of panel "Global view"
  Spatialization: Choose "Fruchterman and Reingold"
  Area: 800 - Gravity: 1

  Click on "Stop" when stabilized

- **Customize the view**:
  - zoom or de-zoom with the mouse
  - change link width (bottom toolbox)
  - display labels and change labels size and color (bottom toolbox)
  - with the "select" tool, visualize a node neighbors (top left toolbox)
  - with the "move" tool, move a node (top left toolbox)

Visualization

- **Visualize the graph with Fruchterman & Reingold algorithm**

  Bottom left of panel "Global view"
  Spatialization: Choose "Fruchterman and Reingold"
  Area: 800 - Gravity: 1

  Click on "Stop" when stabilized

- **Customize the view**:
  - zoom or de-zoom with the mouse
  - change link width (bottom toolbox)
  - display labels and change labels size and color (bottom toolbox)
  - with the "select" tool, visualize a node neighbors (top left toolbox)
  - with the "move" tool, move a node (top left toolbox)
- **Use the view to understand the graph** (delete labels before)
  - Color a node's neighbors (middle left toolbox)
  - Visualize the shortest path between two nodes (middle left toolbox)
  - Visualize the distances from a given node to all the other nodes by nodes coloring (middle left toolbox)

Graph mining

- **Node characteristics**

Window / Statistics

On the bottom right panel, Degree: run

Check on Data Lab

On the top left panel, Clustering / Nodes / Size: Choose a clustering parameter: Degree

Size: Min size:10, Max size: 50, Apply

On the bottom right panel, Shortest path: run

Color: Choose a clustering parameter: Betweenness centrality

Default: choose a palette, Apply

Node clustering

- **Node characteristics**

On the bottom right panel, Modularity: run

Check on Data Lab

On the top left panel, Clustering / Nodes / Label color: Choose a clustering parameter: Modularity class

Color: Default: choose a palette, Apply

Node clustering

- **Node characteristics**

  On the bottom right panel, Modularity: run
  Check on Data Lab
  On the top left panel, Clustering / Nodes / Label color: Choose a clustering parameter: Modularity class
  Color: Default: choose a palette, Apply

- **Export a view**

  Preview / Default with straight links / Refresh / Export (bottom left)

Friedman, J., Hastie, T., and Tibshirani, R. (2008).
Sparse inverse covariance estimation with the graphical lasso.
Biostatistics, 9(3):432–441.

Liu, H., Roeber, K., and Wasserman, L. (2010).
Stability approach to regularization selection for high dimensional graphical models.
In *Proceedings of Neural Information Processing Systems (NIPS 2010)*, pages 1432–1440, Vancouver, Canada.

Martin, P., Guillou, H., Lasserre, F., Déjean, S., Lan, A., Pascussi, J., San Cristobal, M., Legrand, P., Besse, P., and Pineau, T. (2007).
Novel aspects of PPARα-mediated regulation of lipid and xenobiotic metabolism revealed through a multrigenomic study.
Hepatology, 54:767–777.