Supplementary File: NetBioV: An *R* package for visualizing large network data in biology and medicine

Shailesh Tripathi[1] and Matthias Dehmer[2] and Frank Emmert-Streib[1*]
[1] Computational Biology and Machine Learning Laboratory
Center for Cancer Research and Cell Biology
School of Medicine, Dentistry and Biomedical Sciences
Faculty of Medicine, Health and Life Sciences
Queen's University Belfast
97 Lisburn Road, Belfast, BT9 7BL, UK
v@bio-complexity.com
[2] Institute for Bioinformatics and Translational Research, UMIT
Eduard Wallnoefer Zentrum 1, 6060, Hall in Tyrol, Austria

Abstract

This is the supplementary file for our main manuscript containing instructions and various examples for the visualization capabilities of NetBioV - an R based software package. Due to the flexibility of NetBioV, allowing to combine different layout, color and feature styles with each other, there is a vast number of different network visualizations that can be realized.

*Corresponding author: Frank Emmert-Streib

Contents

1 Source code for Figure 1 A, B in the main manuscript

The following code allows to reproduce Figure 1 A and B in the main manuscript.

```
> ############ Figure A #############
> library("igraph")
> library("netbiov")
> data("PPI_Athalina")
> gparm <- mst.plot.mod(g1, v.size=1.5,e.size=.25,
+ colors=c("red", "orange", "yellow", "green"),
+ mst.e.size=1.2,expression=abs(runif(vcount(g1),
+ max=5, min=1)), sf=-15, v.sf=5,
+ mst.edge.col="white", layout.function=layout.fruchterman.reingold)
> ############ Figure B #############
> library("igraph")
> library("netbiov")
> data("PPI_Athalina")
> data("color_list")
> gparm<- plot.abstract.nodes(g1, nodes.color=color.list$citynight,
+ lab.cex=1, lab.color="white", v.sf=-18,
+ layout.function=layout.fruchterman.reingold)
```

2 General guidelines for using NetBioV

In the following, we provide a gallery of networks demonstrating the usage of different layout styles, color schemes and combinations of these provided by our R package NetBioV. Due to the fact that the functions we provide to visualize networks are object oriented, there is a vast combination of different visual effects one can generate from the base functions. Below, for every figure, the corresponding code is provided to reproduce the figures. We are including four example networks with the NetBioV package, two artificially created networks with $10,000$ and $5,000$ nodes respectively, and a total number of edges of $32,761$ and $23,878$ respectively. Furthermore, we provide two biological networks, a gene regulatory network of B-Cell lymphoma inferred with BC3NET and the PPI network of *Arabidopsis thaliana*; see Table 1 for details. In NetBioV the functions *plot.modules*, *split.mst*, *plot.abstract.module* and *plot.abstract.nodes* require information about their modules for plotting the networks. For this reason there are two ways to specify this information. (1) The user defines the modules in the network by specifying a list of objects, where each component of the list object is a vector of vertex ids of a module. (2) If module information is not provided in this way our plotting functions automatically predict modular information about the network using the *fastgreedy* algorithm.

Networks	number of vertices	number of edges
Artificial Network 1	10,000	32,761
Artificial Network 2	5,000	23,878
B-Cell Network	2,498	2,654
A. Thaliana	1,212	2,574

Table 1: Summary of the networks, we provide as part of the NetBioV package to demonstrate its visualization capabilities.

3 Brief introduction of networks available as example networks in NetBioV

Artificial Network: First we generate a network of 10000 and 5000 nodes using *barabasi.game* function available in *igraph*. In the second step we select n_s nodes randomly of degree greater than 5. In the third step for each node in n_s, we select n_e neighbors randomly of order 2 and 3 and draw edges between them.

B-Cell Network: This is a subnetwork of gene regulatory network of B-Cell inferred with *BC3NET [1]*.

A. Thaliana Network: This is a subnetwork of PPI network of the main network [2].

3.1 Plotting time for networks using NetBioV

The plotting time of any network using NetBioV depends on the size of the network and its edge density. To provide the user with some estimates of the expected plotting times, we compared two artificially generated networks with two biological networks on an Apple computer with an *intel i3* processor and *8GB RAM*. The estimated time of plotting these networks are given in the Table 2.

Networks \ Functions	mst.plot.mod	plot.modules	plot.abstract.nodes
Artificial Network 1	5.50 minutes	25.6 seconds	19.35 seconds
Artificial Network 2	1.44 minutes	9.94 seconds	4.31 seconds
B-Cell Network	6 seconds	2 seconds	1.1 seconds
A. Thaliana	3.30 seconds	1.2 seconds	1.1 seconds

Table 2: Comparison of estimated plotting times for four different networks using NetBioV.

3.2 Loading example data

```
> ########## Loading the artificial network with $10,000$ edges
> data("artificial1.graph")
> ########## Loading the artificial network with $5,000$ edges
> data("artificial2.graph")
> ########## Loading the B-Cell network and module information
> data("gnet_bcell")
> data("modules_bcell")
> ########## Loading the Arabidopsis Thaliana network and module information
> data("PPI_Athalina")
> data("modules_PPI_Athalina")
```

```
> ########## Loading a predefined list of colors
> data("color_list")
```

3.3 Interactive representation

In order to modify a network interactively one, first, needs to plot a network with any of the functions we are providing. Then, one uses the output of these function to call the function 'tkplot.netbiov'. Below is an example.

```
> data("artificial2.graph")
> xx<- plot.abstract.nodes(g1, layout.function=layout.fruchterman.reingold,
+ v.sf=-30, lab.color="green")
> #tkplot.netbiov(xx)
```

4 Comparison of NetBioV with other network visualization software

4.1 yEd

yEd is a multi-purpose graph visualization software that does not target any particular networks from application domains. For this reason, it offers no functionality with respect to the 'modular structure' in networks from biology or medicine, as provided by NetBioV. For instance, NetBioV allows the automatic identification of the modules within a network.

yEd allows the visualization of 5 **major graph layout styles:**

- **Hierarchical Layout**
- **Organic Layout**
- **Orthogonal Layout**
- **Tree Layout**
- **Circular Layout**

All of these layouts are available in NetBioV, except 'orthogonal layout'. The reason for this is that the resulting arrangement of the network, looks like a electronic circuit, which is for networks with more than 100 **genes difficult to apply.**

All of these layout styles can only be applied 'globally' to the network as a whole, but not to selected parts of a network, as possible with NetBioV. This limit the number of resulting graph layout styles to exactly 5**.**

Also, yEd does not provide information flow layout styles, as available in NetBioV.

yEd does not allow to identify modules by selecting an algorithm, as is available in NetBioV.

4.2 Cytoscape

Cytoscape allows the usage of a set of commercial layouts provided by yWorks, on which yEd is based too (see above for graph layout styles of yEd). In addition, Cytoscape provides the following layouts that have been implemented by the Cytoscape develops:

- **Grid Layout**
- **Spring-Embedded Layout**
- **Circle Layout**
- **Group Attributes Layout**

All of these graph layouts, except Grid Layout, are also available from NetBioV. Grid Layout show a regular arrangements of the nodes in a graph which is hard to apply to network from biology or medicine.

The Group Attributes Layout can be obtained by NetBioV if modules are defined in the network, either by the application of an algorithm or manually. In addition, NetBioV allows to specify any graph layout for each module separately. Also, Cytoscape does not provide information flow layout styles, as available in NetBioV.

4.3 visANT

visANT is a graph visualization software that has been specifically designed to visualize biological networks, like NetBioV.

The graph layouts provided by visANT are similar to yEd and Cytoscape, and limited to global graph layouts, meaning that one cannot select parts of a network and use different layout for these, as possible with NetBioV. Also, visANT does not provide information flow layout styles, as available in NetBioV.

visANT does not allow to identify modules by selecting an algorithm, as available in NetBioV.

visANT does not allow to save figures in the pdf format, as available in NetBioV. Instead, visANT allows the user to save figures in a svg format. If a pdf format is required, e.g., for publications, an external converter software needs to be used (not part of visANT).

Visualization software⇒ Features ⇓	NetBioV	igraph	Cytoscape	VisANT	yEd	RgraphViz
Application type	R (based on igraph)	R	Java	Java	Java (yWorks)	R
Input	Use formats as igraph	Tab delim, graphml, gml, graphdb	BioPax, XML, XGMML, GML, GRAPHML, Tab delim	Tab delim, BioPax, XML	YGF, GML, XGML, excel, TGF, GED	Tab delim
Graphical user interface	Yes (via tkplot in igraph)	Yes	Yes	Yes	Yes	No
Save image as a postscript or pdf	Yes	Yes	Yes	No	Yes	Yes
Save all graph features	Data frame (Tab delim) or a R object of netbiv class	R object of igraph class	Cys format	Txt or XML	Graphml, GML, TGF, XGML	R object of graph class
Modular view with different layout options for modules	Yes	No	No	No	No	No
Multiroot-tree visualization (information flow)	Yes	No	No	No	No	No
Information flow between sets of nodes or modules	Yes	No	No	No	No	No
Abstract view of a network (modular layout)	Yes	No	No	No	No	No
Global layout	Use force-based algorithm on minimum spanning tree of a network to plot nodes. Then, add remaining edges and color, based on distance.	Force-based	Force-based	Force-based	Force-based	Force-based

Table 3: Comparison of different network visualization software with NetBioV.

4.4 Overview table

The following table shows a comparison of features provided by NetBioV, yEd, Cytoscape and visANT.

5 Network gallery and example code

5.1 Global layout style of artificial network 1

```
> ###Generation of the network:
> require(netbiov)
> data("artificial1.graph")
> hc <- rgb(t(col2rgb(heat.colors(20)))/255,alpha=.2)
> cl <- rgb(r=0, b=.7, g=1, alpha=.05)
> xx <- mst.plot.mod(g1, vertex.color=cl, v.size=3, sf=-20,
+ colors=hc, e.size=.5, mst.e.size=.75,
+ layout.function=layout.fruchterman.reingold)
```

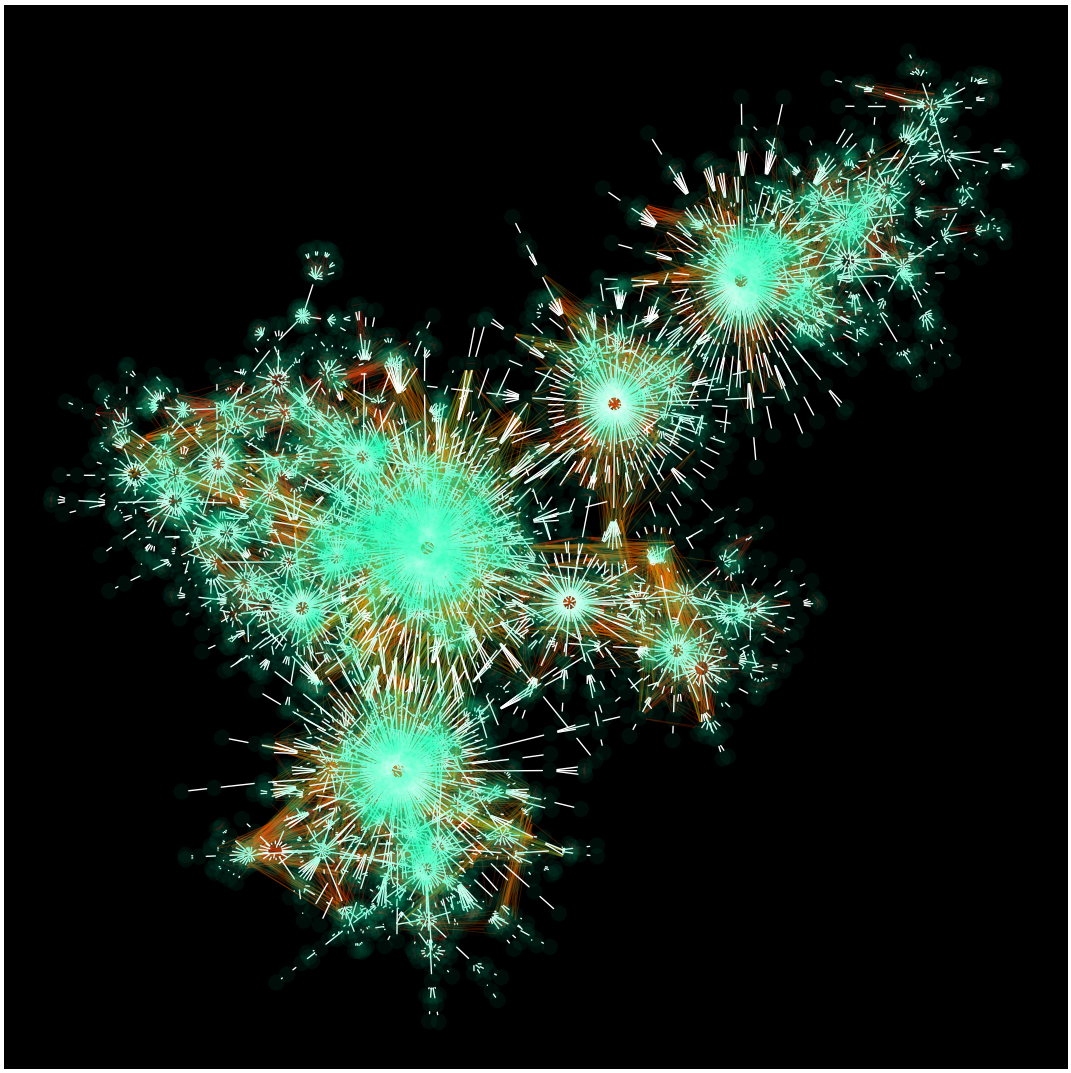

Figure 1: Global layout style of artificial network 1 ($|V| = 10,000$, $|E| = 32,761$). The positions of the nodes in the MST (minimum spanning tree) are determined by Fruchterman-Reingold algorithm. Edges found by the MST are white, other edges are shades of red to yellow.

5.2 Global layout style of artificial network 2

```
> require(netbiov)
> data("artificial2.graph")
> hc <- rgb(t(col2rgb(heat.colors(20)))/255,alpha=.2)
> cl <- rgb(r=1, b=.7, g=0, alpha=.1)
> fn <- function(x){layout.reingold.tilford(x, circular=TRUE,
+ root=which.max(degree(x)))}
> xx <- mst.plot.mod(g1, vertex.color=cl, v.size=1, sf=30,
+ colors=hc, e.size=.5, mst.e.size=.75,
+ layout.function=fn, layout.overall=layout.kamada.kawai)
```

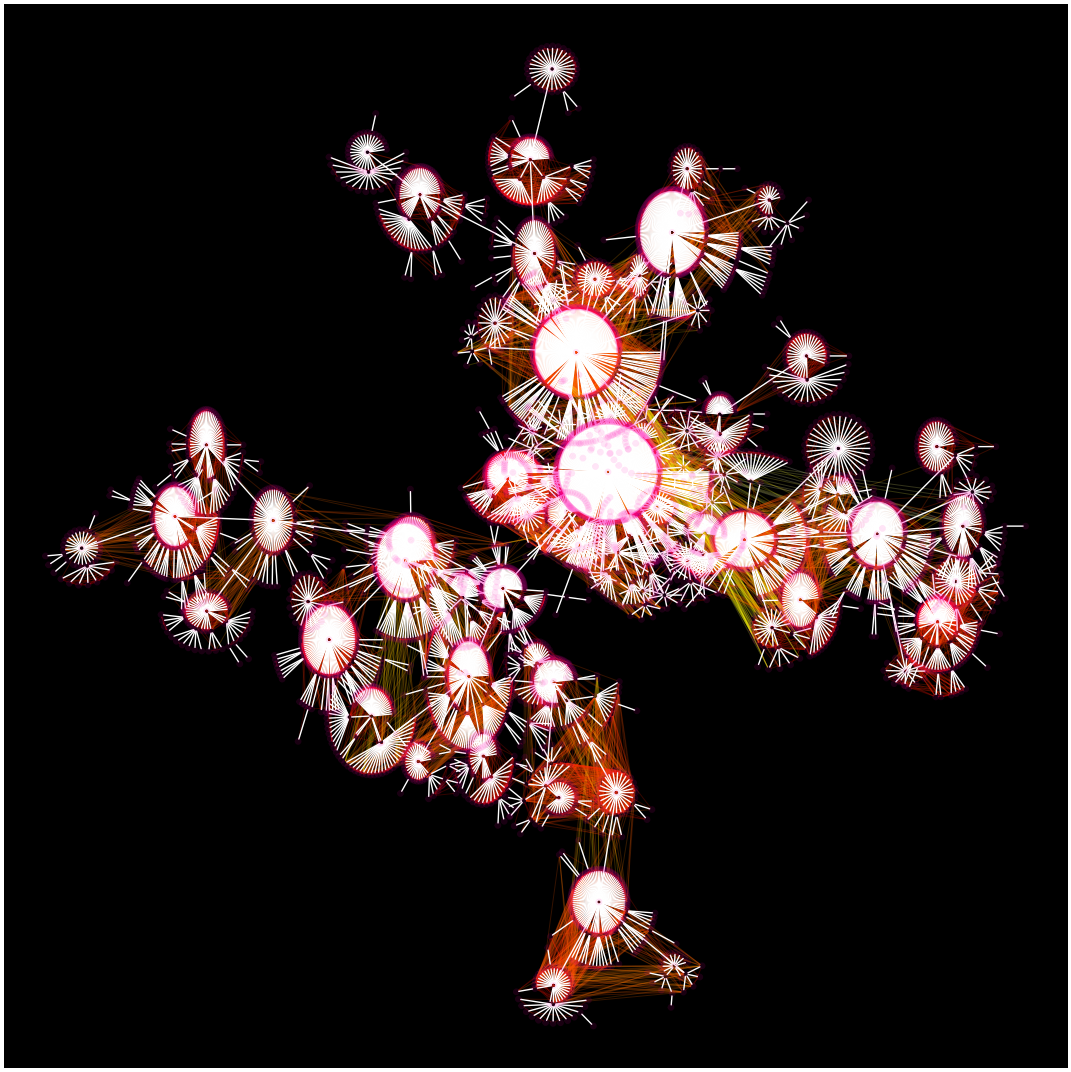

Figure 2: Global layout style of artificial network 2 ($|V| = 5,000$, $|E| = 23,878$). Position of the nodes in the MST are determined by combining the Reingold-Tilford and the Kamada-Kawai algorithm. Edges found by the MST are white, other edges are shades of red to yellow.

5.3 Modular layout style of artificial network 1

```
> data("artificial1.graph")
> xx <- plot.modules(g1, v.size=.8,
+ modules.color=c("red", "yellow"),
+ mod.edge.col=c("green", "purple"), sf=30)
```

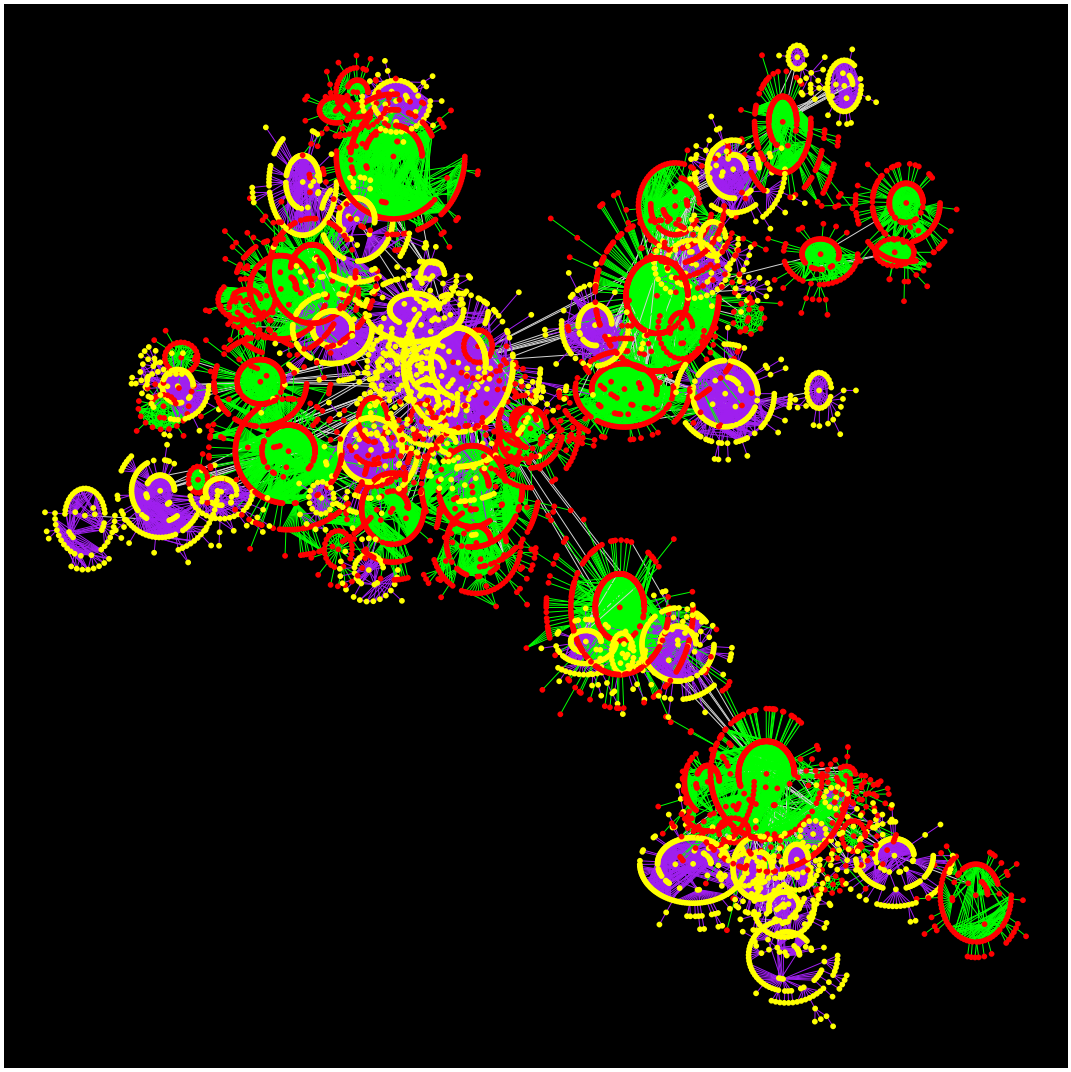

Figure 3: Modular layout style of artificial network 1 ($|V| = 10,000$, $|E| = 32,761$), modules are plotted using Reingold-Tilford and Fruchterman-Reingold algorithm. Each module is colored in red or yellow color and edges of modules are colored in purple and green, the edges connecting modules are colored in grey.

5.4 Multiroot-tree (hierarchical) layout style of artificial network 1

```
> data("artificial1.graph")
> cl <- c(rgb(r=1, b=1, g=0, alpha=.2))
> cl <- rep(cl, 3)
> ecl <-   c(rgb(r=.7, b=.7, g=.7, alpha=.2), rgb(r=.7, b=.7, g=.7, alpha=.2),
+ rgb(r=0, b=0, g=1, alpha=.2), rgb(r=.7, b=.7, g=.7, alpha=.2))
> ns <- c(1581, 1699, 4180, 4843, 4931, 5182, 5447, 5822, 6001,
+ 6313, 6321, 6532, 7379, 8697, 8847, 9342)
> xx <- level.plot(g1, tkplot=FALSE, level.spread=TRUE, v.size=1,
+ vertex.colors=cl,  edge.col=ecl, initial_nodes=ns, order_degree=NULL,
+ e.curve=.25)
```

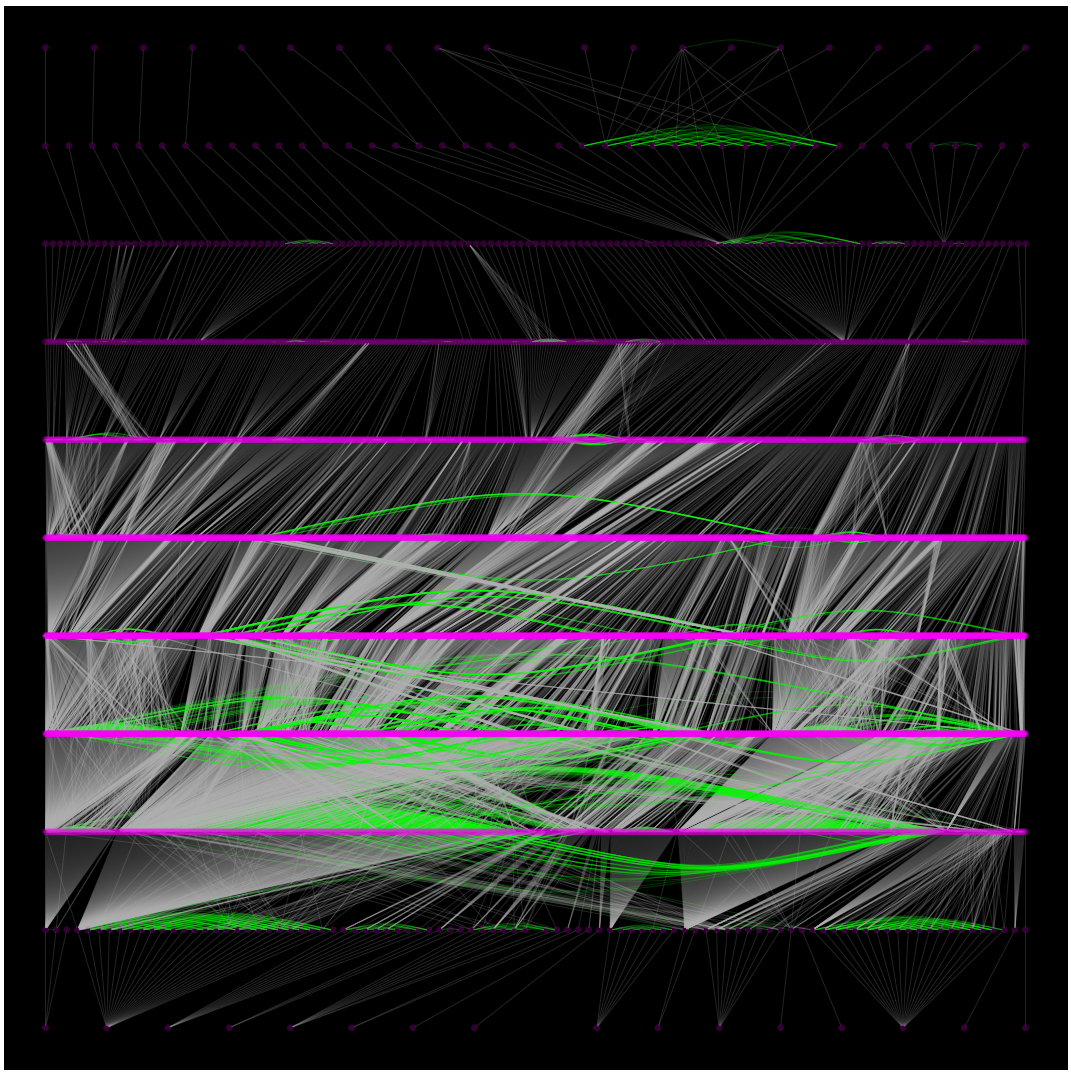

Figure 4: Multiroot-tree (hierarchical) layout style of artificial network 1 ($|V| = 10,000$, $|E| = 32,761$). Edges connecting nodes on different levels are in grey and edges connected nodes on same level are shown in green.

5.5 Modular layout style of artificial network 2

```
> data("artificial2.graph")
> xx <- plot.modules(g1,mod.lab=TRUE, color.random=TRUE, mod.edge.col="grey",
+ ed.color="gold", sf=15, v.size=.5,layout.function=layout.fruchterman.reingold,
+ lab.color="grey", modules.name.num=FALSE, lab.cex=1, lab.dist=5)
```

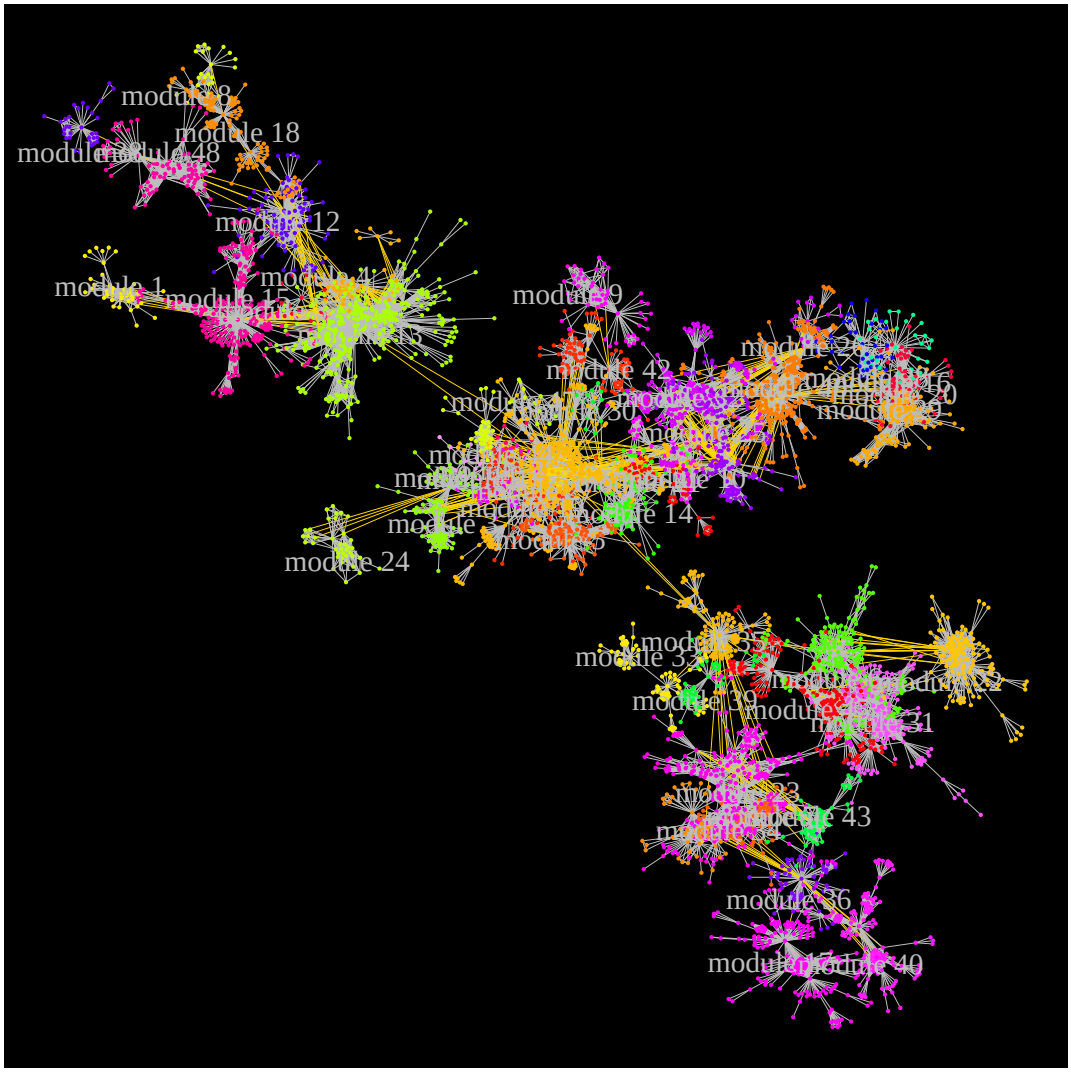

Figure 5: Modular view of artificial network 2 ($|V| = 5,000$, $|E| = 23,878$), emphasizing the labels of each module. The colors of the modules are selected randomly.

5.6 Abstract modular view of artificial network 2

```
> data("artificial2.graph")
> xx<- plot.abstract.nodes(g1, layout.function=layout.fruchterman.reingold,
+ v.sf=-30, lab.color="green")
```

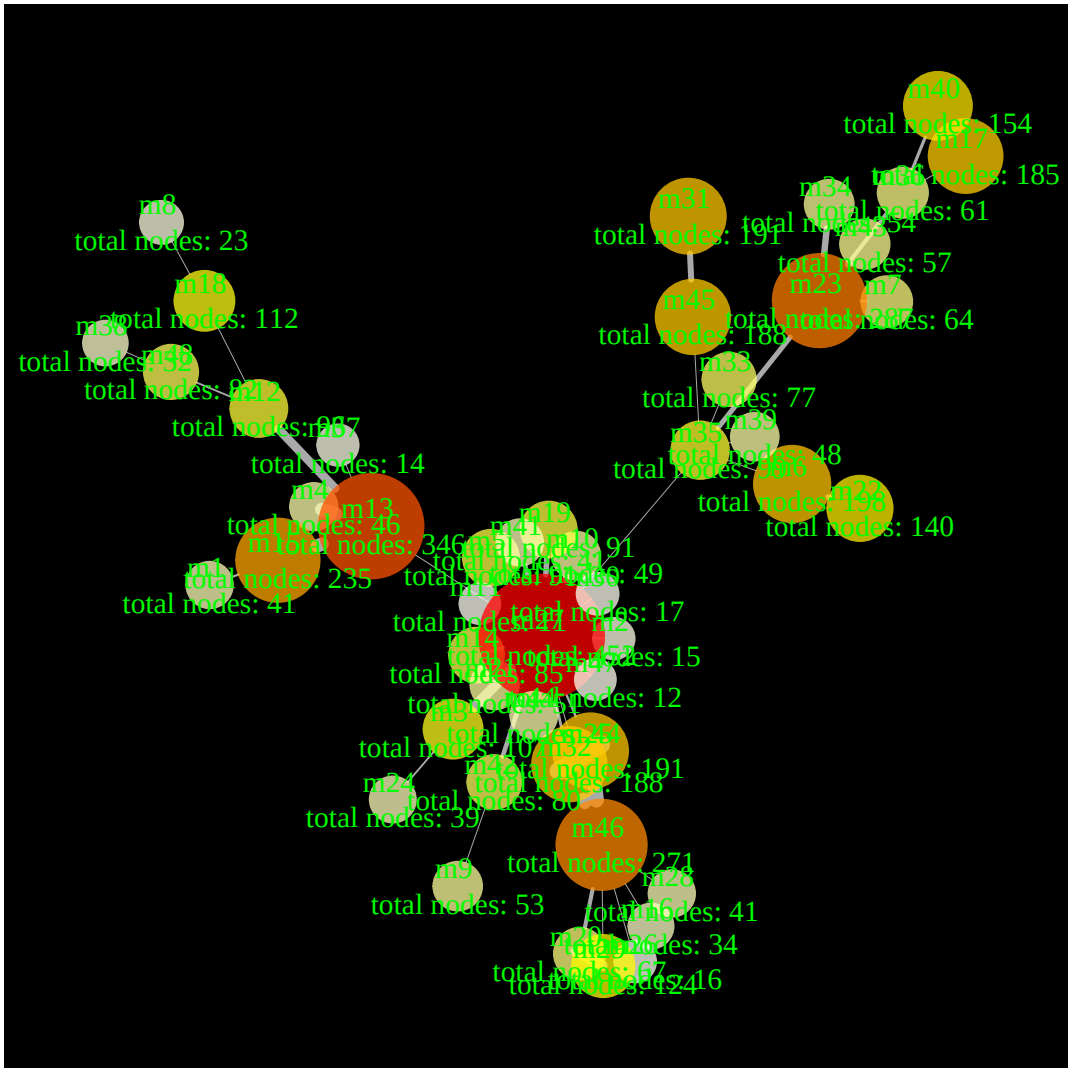

Figure 6: Abstract modular view of artificial network 2 ($|V| = 5,000$, $|E| = 23,878$) emphasizing the labels of each module, the size of each module and the total number of connections between modules. The size of each node represents the number of nodes in that module and the edge-width is proportional to the number of edges between two modules.

5.7 Global view of a component of the B-Cell network

```
> data("gnet_bcell")
> ecl <- rgb(r=0, g=1, b=1, alpha=.6)
> ppx <- mst.plot.mod(gnet, v.size=degree(gnet),e.size=.5,
+ colors=ecl,mst.e.size=1.2,expression=degree(gnet),
+ mst.edge.col="white", sf=-10, v.sf=6)
```

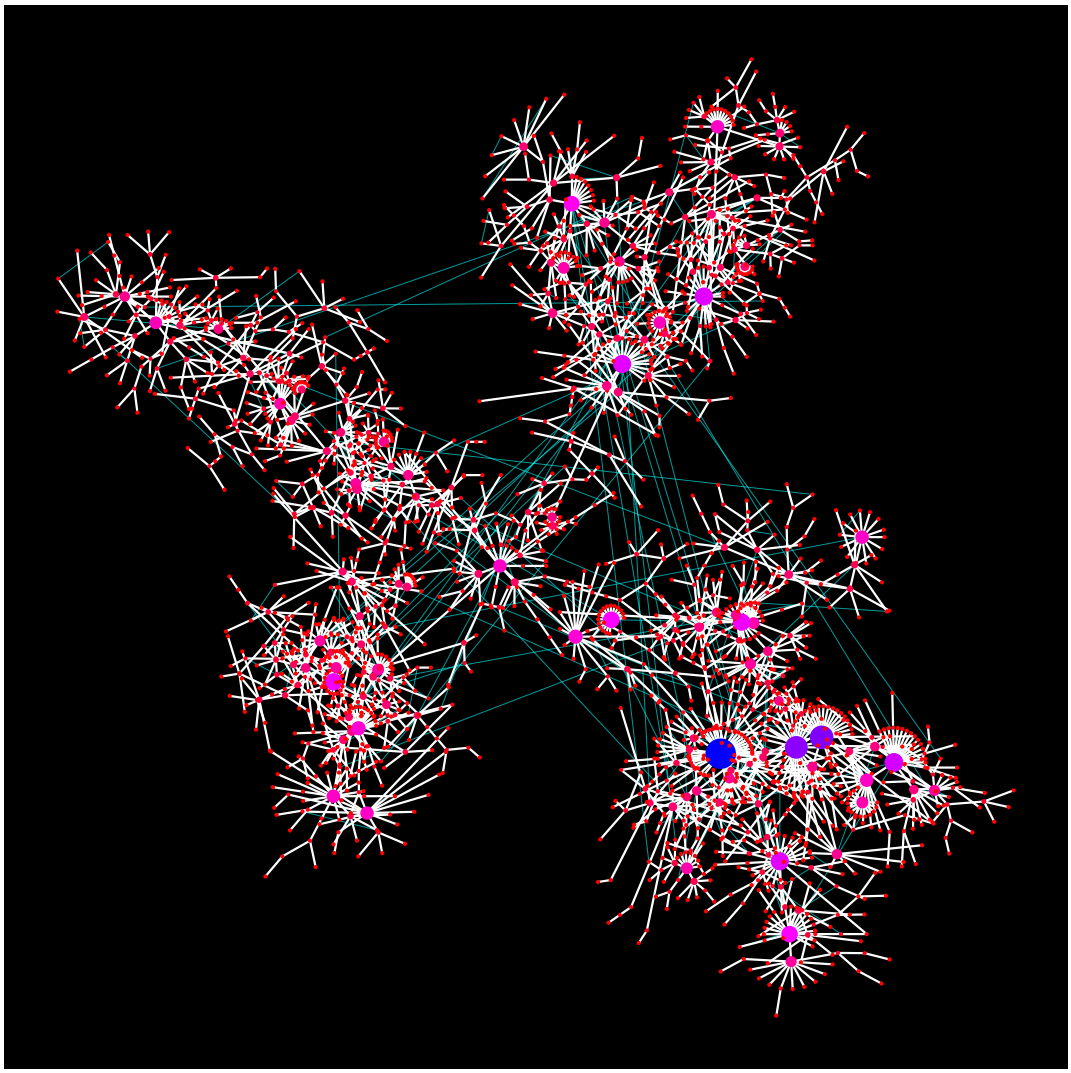

Figure 7: Global view of a component of the B-Cell network ($|V| = 2,498$, $|E| = 2,654$) emphasizing the expression of the genes. The size of each gene is proportional to the degree of the gene. The color of a gene reflects the gene expression value. The minimum spanning tree edges in the network are shown in white and all remaining edges are shown in green, in the background.

5.8 Information flow layout of a component of the B-Cell network

```
> data("gnet_bcell")
> cl <- rgb(r=.6, g=.6, b=.6, alpha=.5)
> xx <- level.plot(gnet, init_nodes=20,tkplot=FALSE, level.spread=TRUE,
+ order_degree=NULL, v.size=1, edge.col=c(cl, cl, "green", cl),
+ vertex.colors=c("red", "red", "red"), e.size=.5, e.curve=.25)
> data("gnet_bcell")
> cl <- rgb(r=.6, g=.6, b=.6, alpha=.5)
> xx <- level.plot(gnet, init_nodes=20,tkplot=FALSE, level.spread=TRUE,
+ order_degree=NULL, v.size=1, edge.col=c(cl, cl, "green", cl),
+ vertex.colors=c("red", "red", "red"), e.size=.5, e.curve=.25)
>
```

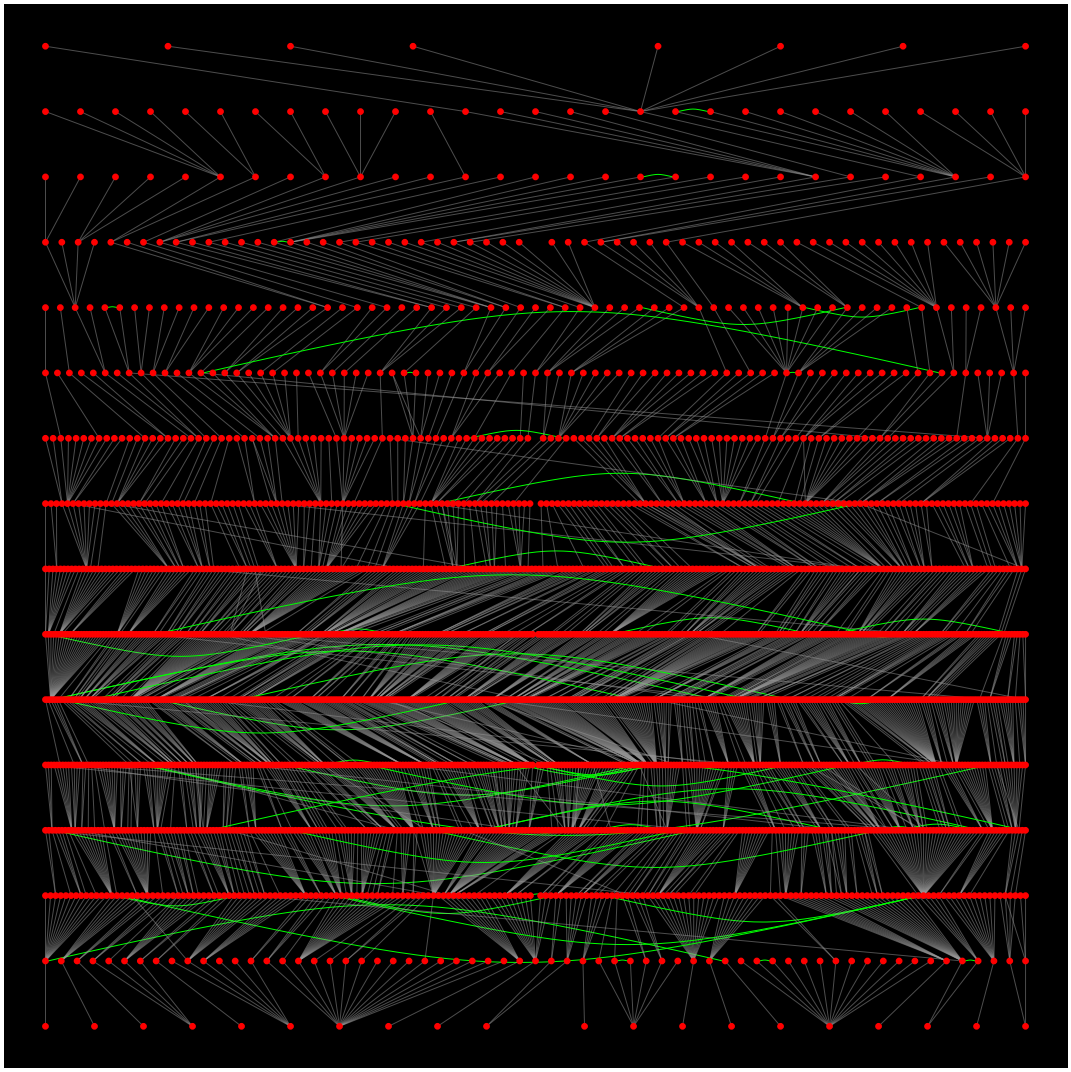

Figure 8: Multiroot-tree (hierarchical) view of a component of the B-Cell network ($|V| = 2,498$, $|E| = 2,654$).

5.9 Modular view of a component of the B-Cell network

```
> data("gnet_bcell")
> xx<-plot.modules(gnet, color.random=TRUE, v.size=1,
+ layout.function=layout.graphopt)
>
```

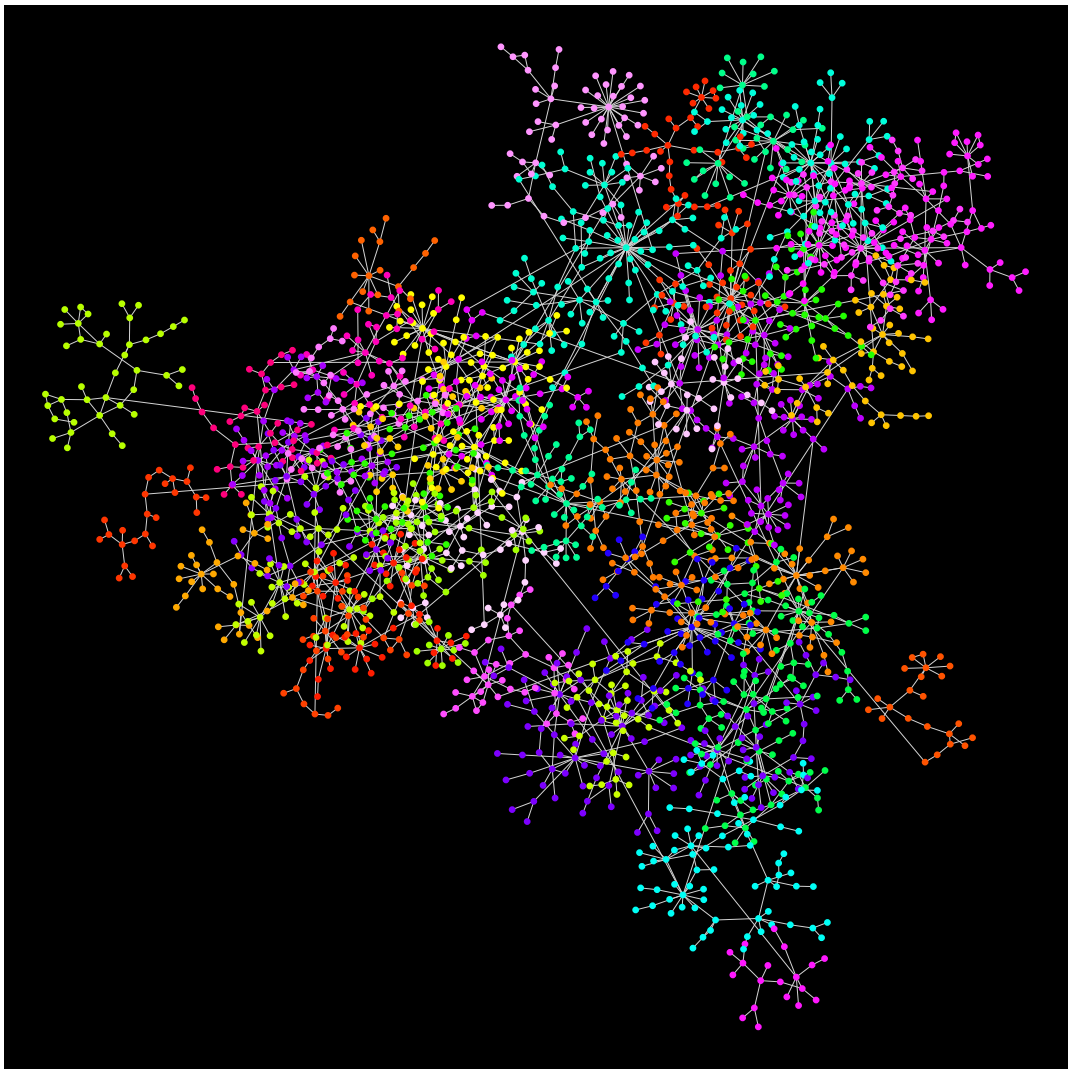

Figure 9: Modular view of a component of the B-Cell network ($|V| = 2,498$, $|E| = 2,654$). Each module is randomly colored.

5.10 Information flow layout of a component of the B-Cell network

```
> data("gnet_bcell")
> xx <- plot.modules(gnet, modules.color=cl, mod.edge.col=cl,
+ sf=5, nodeset=c(2,5,44,34),
+ mod.lab=TRUE, v.size=.9,
+ path.col=c("blue",  "purple", "green"),
+ col.s1 = c("yellow", "pink"),
+ col.s2 = c("orange", "white" ),
+ e.path.width=c(1.5,3.5), v.size.path=.9)
>
```

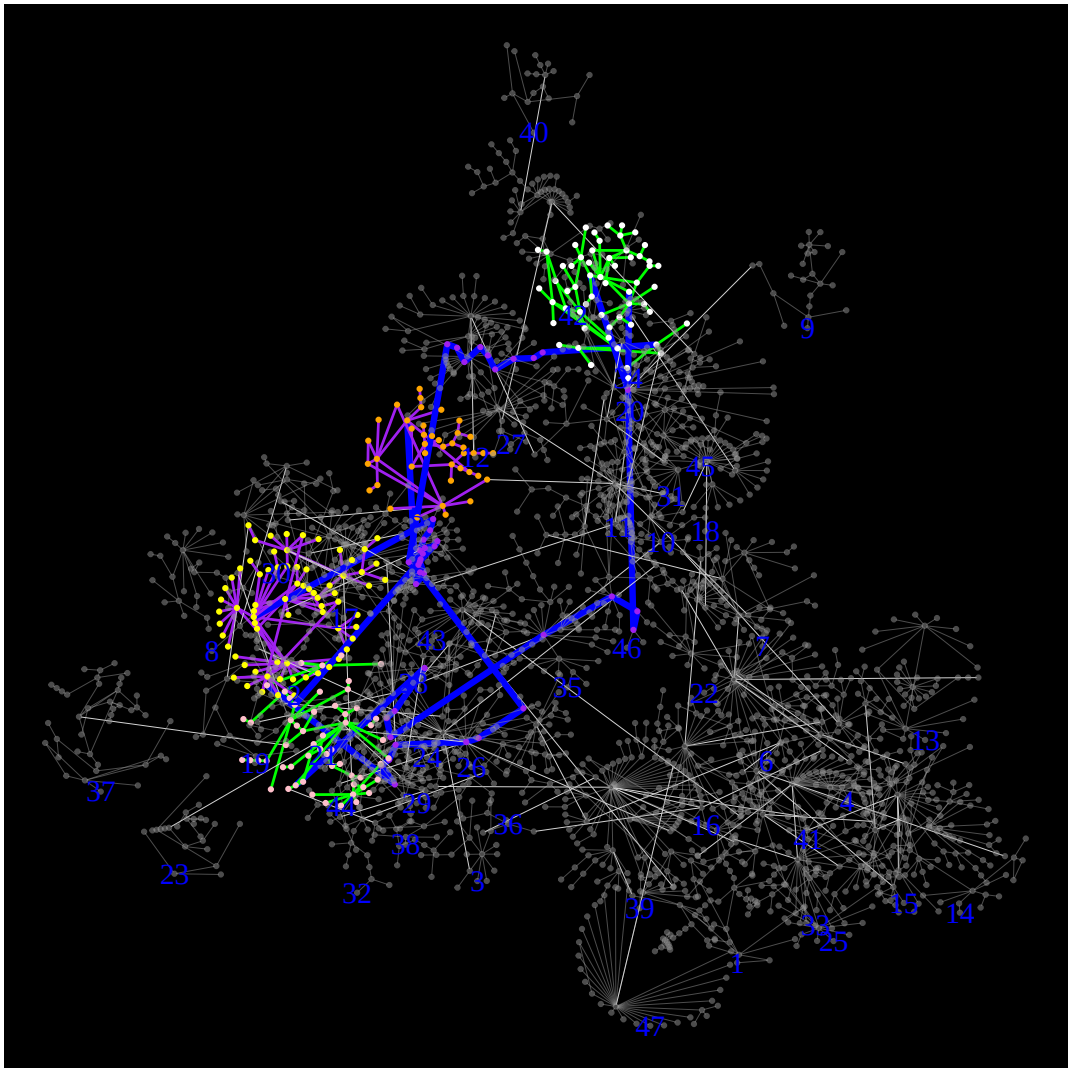

Figure 10: Information flow view of a component of the B-Cell network ($|V| = 2,498$, $|E| = 2,654$). In this figure, we emphasize the information flow between two pairs of gene sets. We show the shortest paths between modules 2, 5 and modules 34, 44. The shortest paths between modules is highlighted by a thick blue line and internal paths within the modules are shown in purple and green colors.

5.11 Abstract modular view of a component of the B-CELL network

```
> data("gnet_bcell")
> xx<-plot.abstract.nodes(gnet, v.sf=-35,
+ layout.function=layout.fruchterman.reingold, lab.color="white",lab.cex=.75)
```

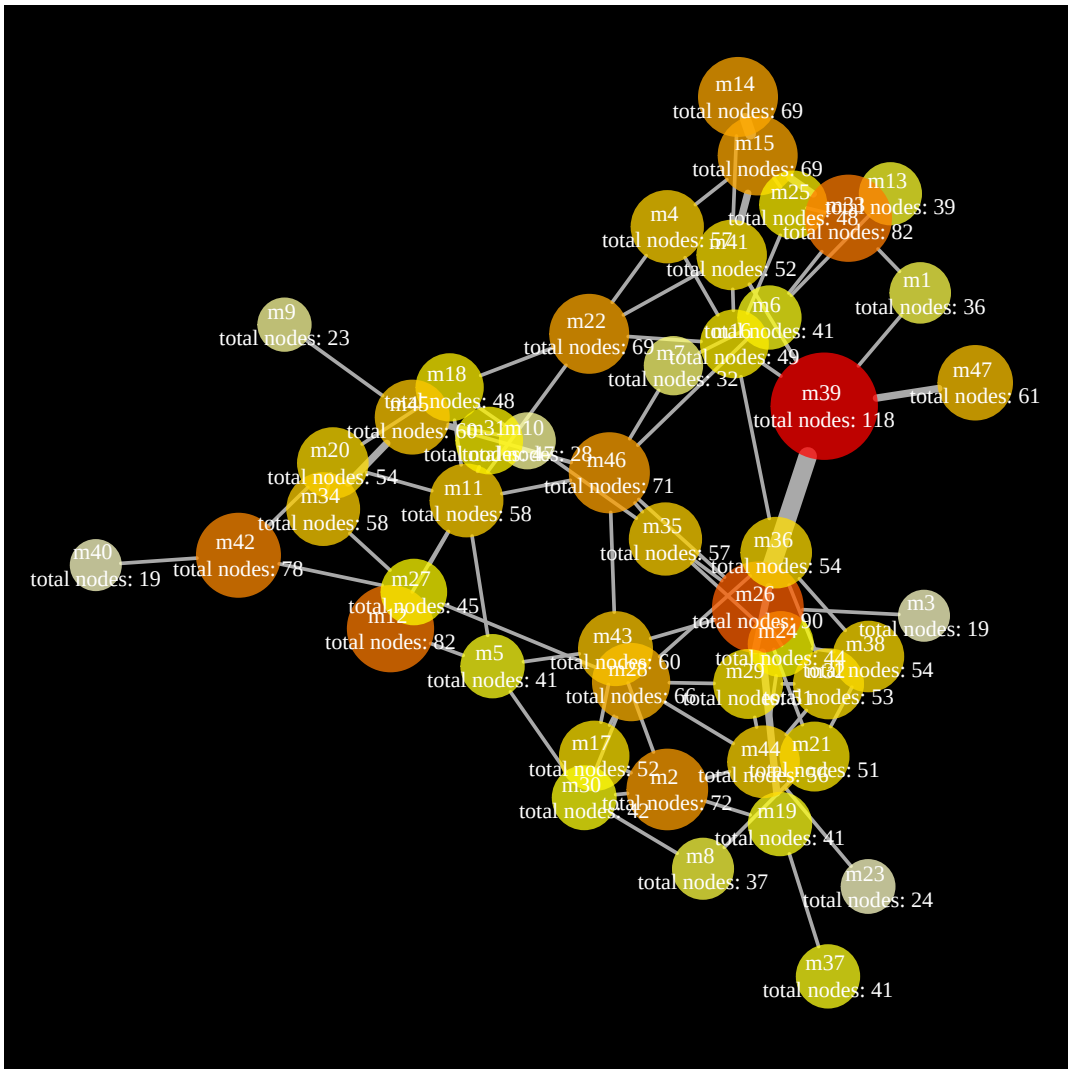

Figure 11: Abstract modular view of a component of the B-CELL network ($|V| = 2,498$, $|E| = 2,654$). Node-size is proportional to the total number of genes in a module and the edge size is proportional to the total number of connections between modules.

5.12 Global layout style of the *A. Thaliana* network: Fruchterman-Reingold

```
> data("PPI_Athalina")
> data("color_list")
> id <- mst.plot(g1, colors=c("purple4","purple"),mst.edge.col="green",
+ vertex.color = "white",tkplot=FALSE,
+ layout.function=layout.fruchterman.reingold)
>
```

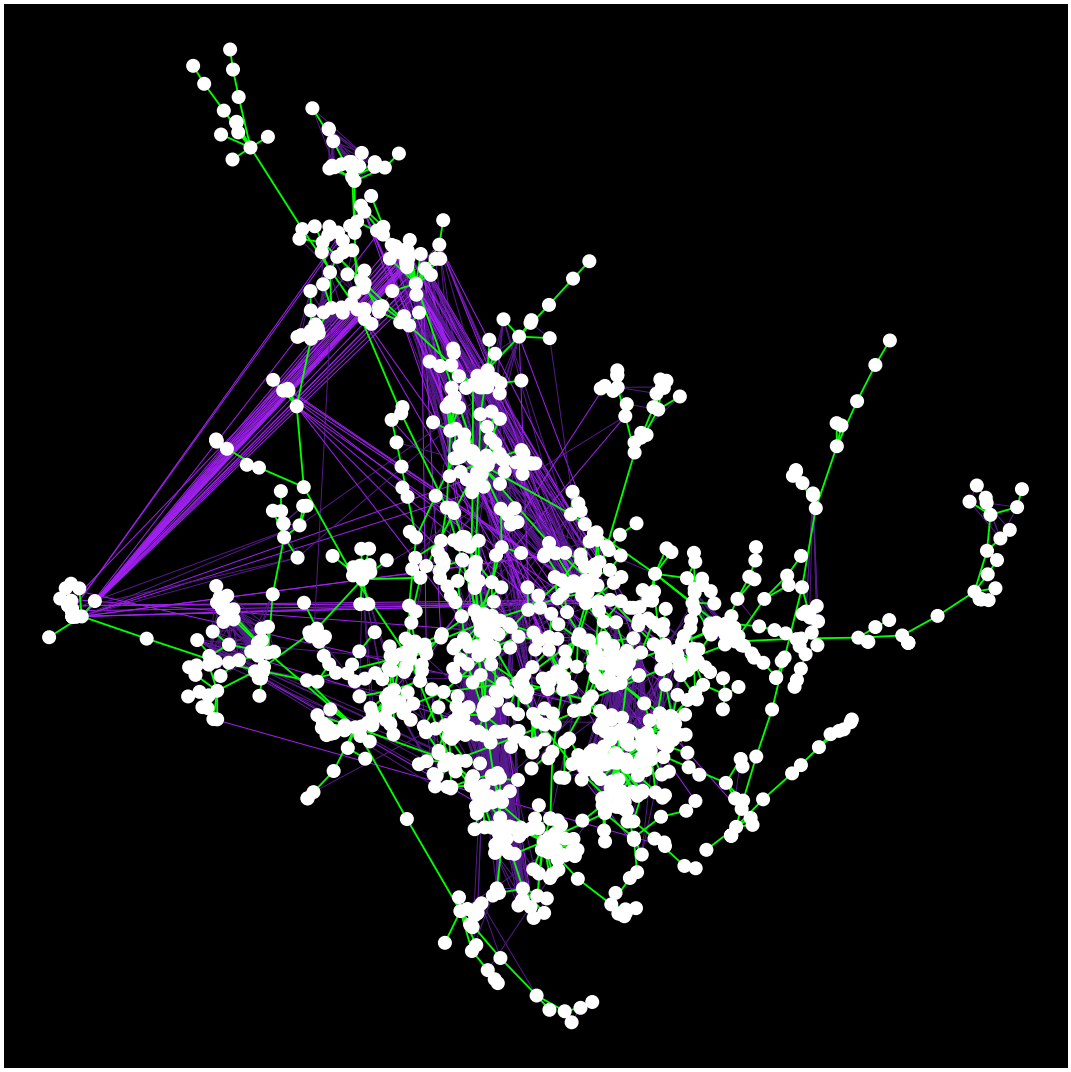

Figure 12: Global layout style of the *A. Thaliana* network. The position of the nodes in the MST are determined by Fruchterman-Reingold, edges found by the MST are shown in green, other edges are shades of purple.

5.13 Global layout style of the *A. Thaliana* network: Kamada-Kawai

```
> data("PPI_Athalina")
> data("color_list")
> id <- mst.plot(g1, colors=c("purple4","purple"),mst.edge.col="green",
+ vertex.color = "white",tkplot=FALSE, layout.function=layout.kamada.kawai)
```

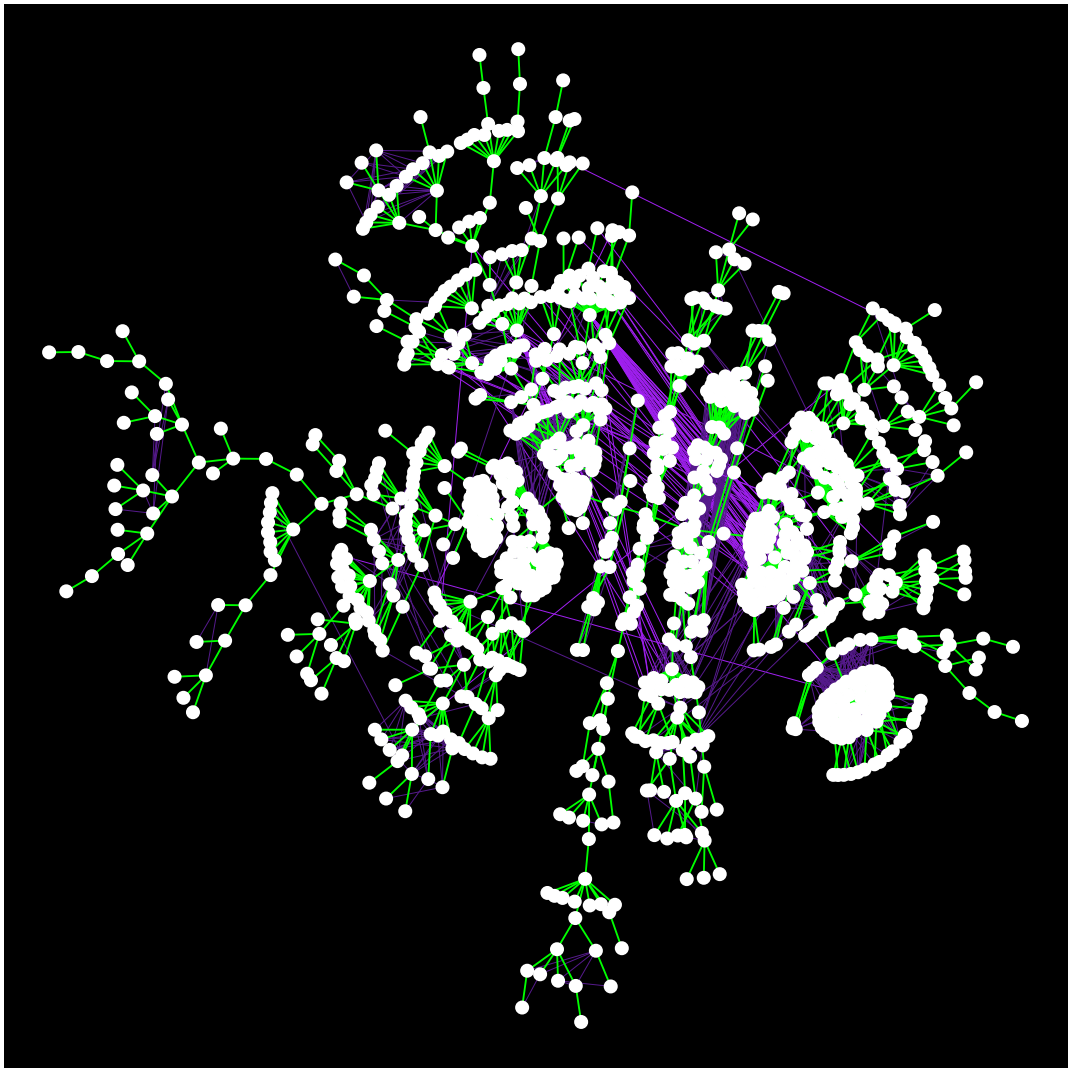

Figure 13: Global layout style of the *A. Thaliana* network. Positions of the nodes in the MST are determined by Kamada-Kawai, edges found by the MST are green, other edges are shades of purple.

5.14 Modular layout of the *A. Thaliana* network: Module highlighting

```
> data("PPI_Athalina")
> data("color_list")
> data("modules_PPI_Athalina")
> cl <- rep("blue", length(lm))
> cl[1] <- "green"
> id <- plot.modules(g1, layout.function = layout.graphopt, modules.color = cl,
+ mod.edge.col=c("green","darkgreen") , tkplot=FALSE, ed.color = c("blue"),sf=-25)
>
```

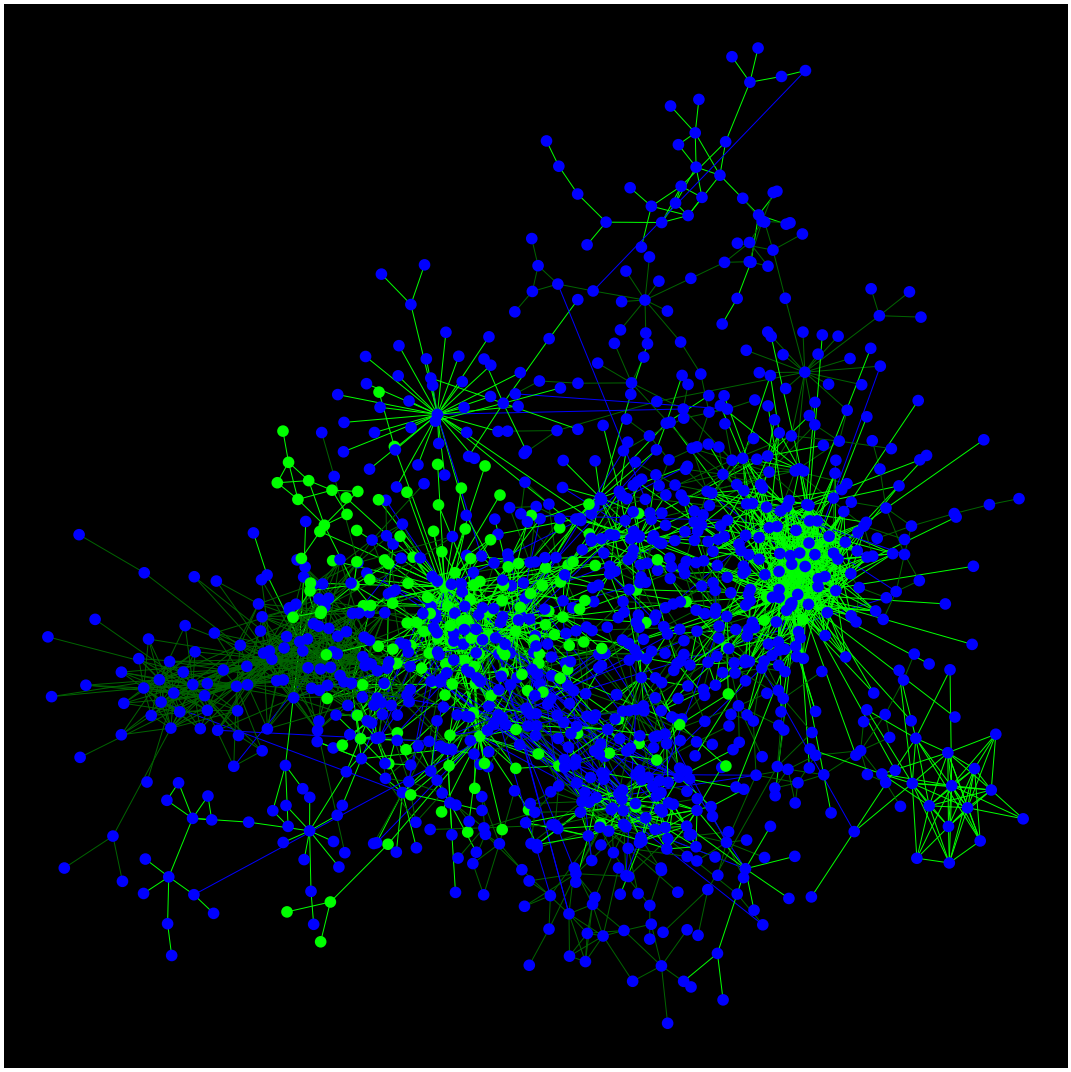

Figure 14: Modular layout of the *A. Thaliana* network. Each module is plotted separately, by using a force-based algorithm (layout.graphopt). The nodes in the modules are shown in blue, except for module 1 where all vertices are shown in green. The edge color of the modules is shown in green.

5.15 Modular layout of the *A. Thaliana* network: Colorize modules

```
> data("PPI_Athalina")
> data("color_list")
> data("modules_PPI_Athalina")
> cl <- rep("blue", length(lm))
> cl[1] <- "green"
> cl[2] <- "orange"
> cl[10] <- "red"
> id <- plot.modules(g1, mod.list = lm,
+ layout.function = layout.graphopt, modules.color = cl,
+ mod.edge.col=c("green","darkgreen") ,
+ tkplot=FALSE, ed.color = c("blue"),sf=-25)
>
```

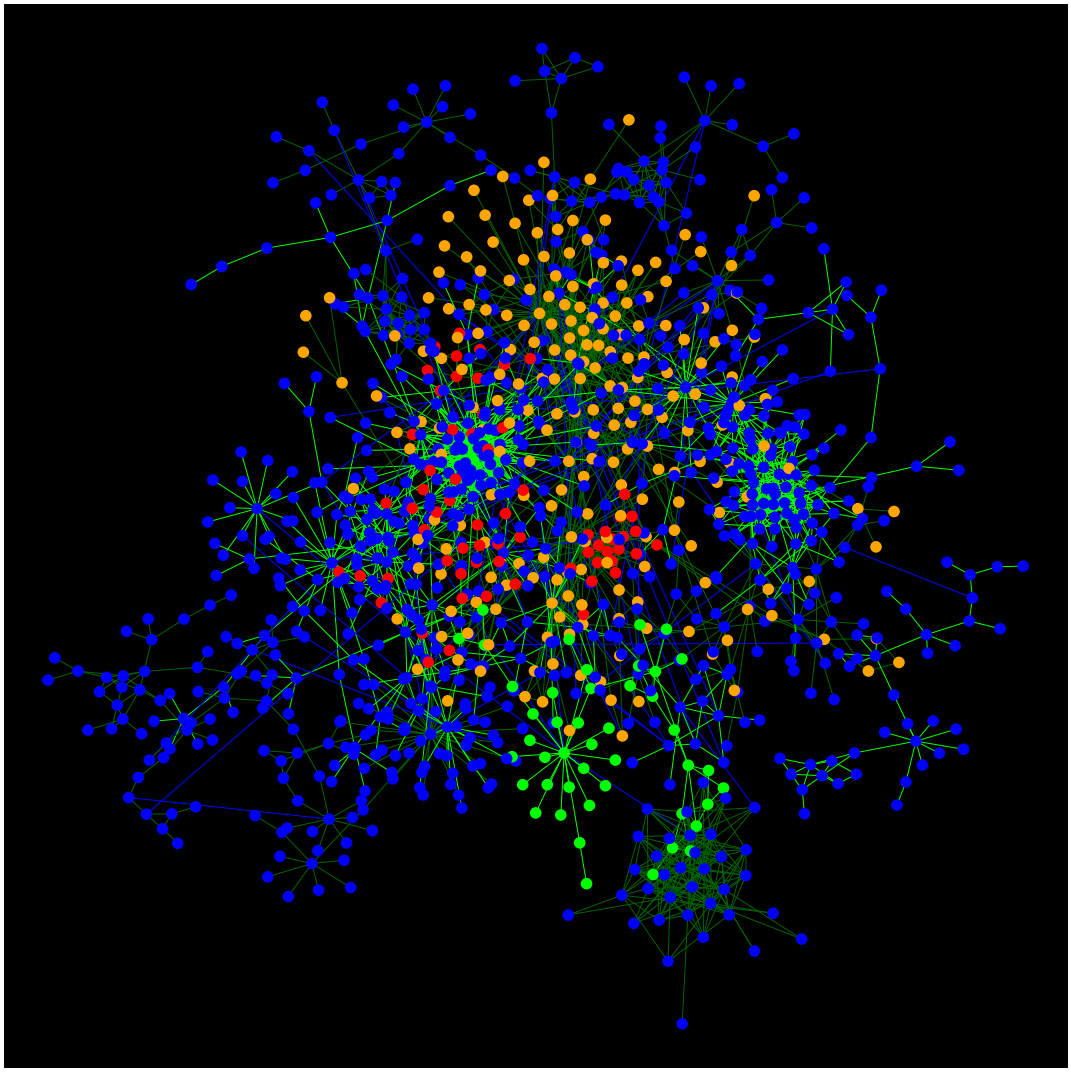

Figure 15: Modular layout of the *A. Thaliana* network. Each module is separately plotted by using a force-based algorithm (layout.graphopt). The nodes for the modules are shown in blue, except for module 1, 2 and 10 where the vertices are shown in green, orange and red. The edge color of the modules is shown in green.

5.16 Modular layout of the *A. Thaliana* network: Fruchterman-Reingold

```
> data("PPI_Athalina")
> data("modules_PPI_Athalina")
> data("color_list")
> id<-plot.modules(g1,mod.list=lm,layout.function=c(layout.fruchterman.reingold),
+ modules.color="grey", mod.edge.col = sample(color.list$bright), tkplot=FALSE)
```

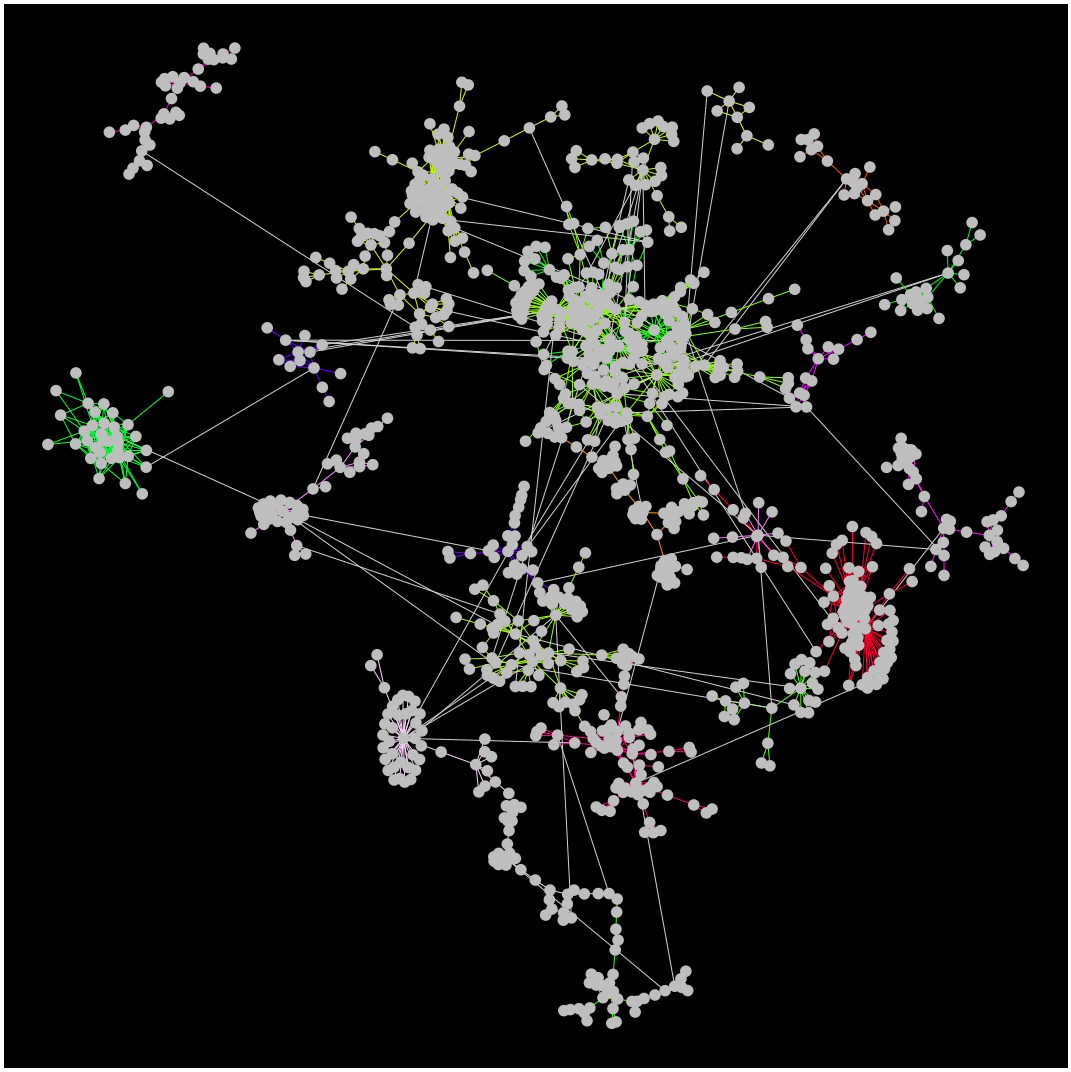

Figure 16: Modular layout style of the *A. Thaliana* network, the coordinates of the nodes in each module are determined by Fruchterman-Reingold. The edge color in each module is given by 'mod.edge.col', the edge color between modules is grey. The modules are arranged by the number of nodes. The module with the largest number of nodes is placed into the center, modules of smaller sizes are arranged in a circular manner.

5.17 Modular layout of the *A. Thaliana* network: Star layout

```
> data("PPI_Athalina")
> data("color_list")
> id <- plot.modules(g1, layout.function = c(layout.fruchterman.reingold),
+ modules.color = sample(color.list$bright),layout.overall = layout.star,
+ sf=40, tkplot=FALSE)
```

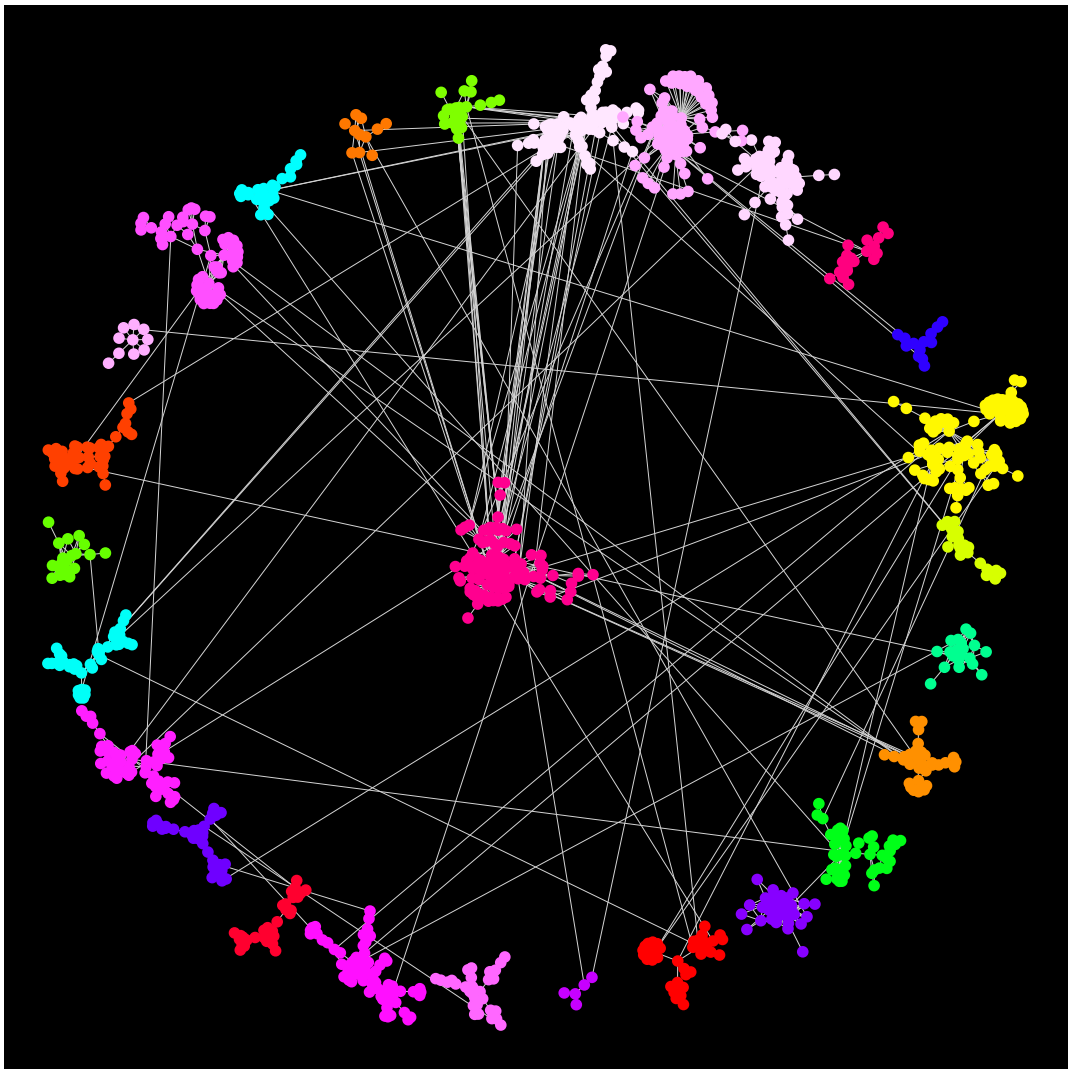

Figure 17: Modular layout style of the *A. Thaliana* network, the coordinates of the nodes in each module are determined by Fruchterman-Reingold. The module consisting of the largest number of nodes is placed in the center. The other modules are arranged in a circular manner. The colors for the modules are chosen randomly from a compiled list of bright colors.

5.18 Modular layout of the *A. Thaliana* network: Mixed layouts

```
> data("PPI_Athalina")
> data("color_list")
> id <- plot.modules(g1, layout.function = c(layout.fruchterman.reingold,
+ layout.star,layout.reingold.tilford, layout.graphopt,layout.kamada.kawai),
+ modules.color = sample(color.list$bright), sf=40, tkplot=FALSE)
```

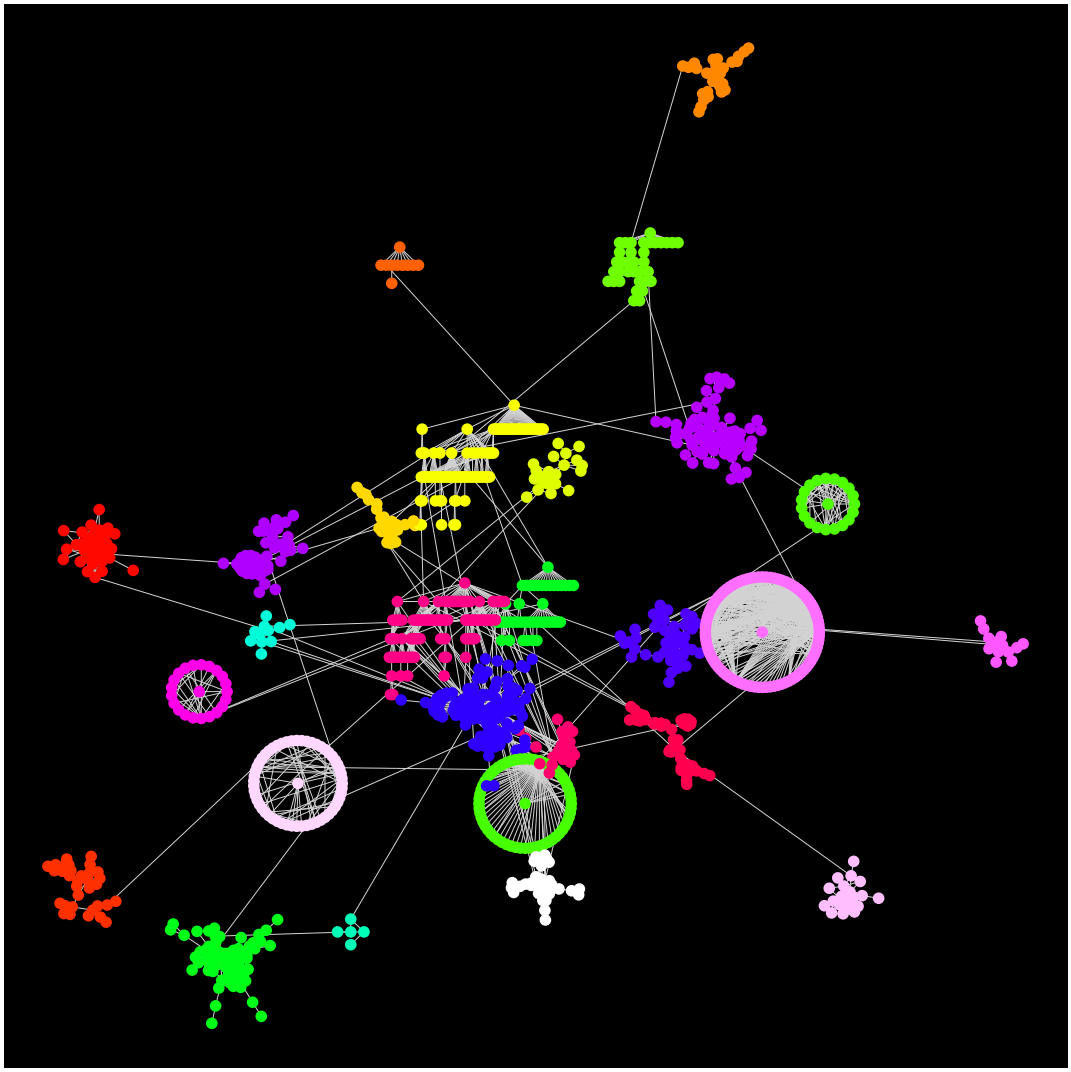

Figure 18: Modular layout style of the *A. Thaliana* network. The position of different modules is determined by different layouts given as inputs to the function. Colors for the modules are chosen randomly from a compiled list of bright colors.

5.19 Modular layout of the *A. Thaliana* network: Node coloring

```
> data("PPI_Athalina")
> data("color_list")
> cl <- list(rainbow(40),  heat.colors(40) )
> id <- plot.modules(g1,  col.grad=cl , tkplot=FALSE)
>
```

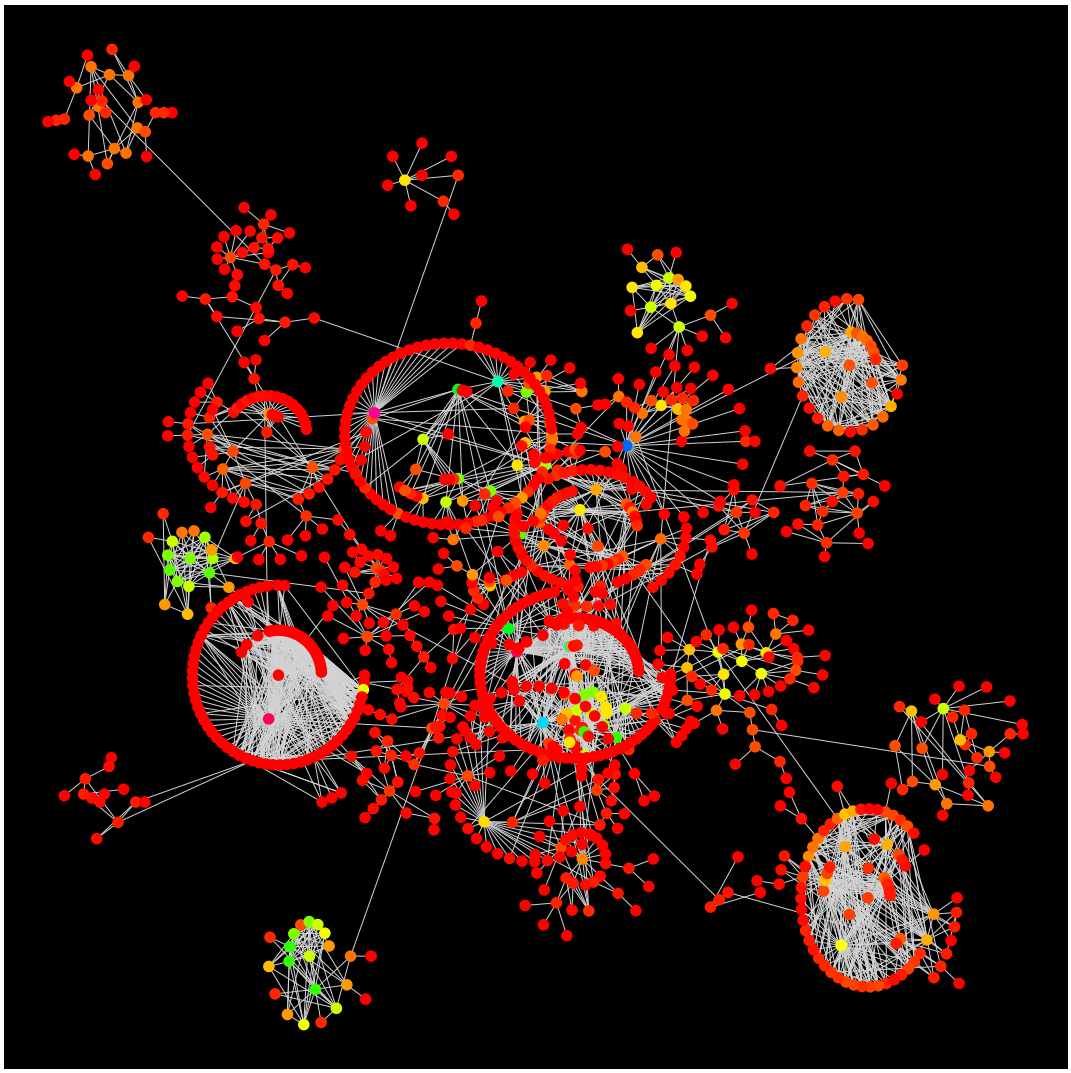

Figure 19: Modular layout of the *A. Thaliana* network. The position of the nodes for each module is determined by a circular tree view. Nodes in each modules are colored based on their degree according to a range of colors provided as an input. For example, nodes in module 2 are colored by a range of heat-colors, where dark colors indicate a high degree and light colors are indicating a low degree.

5.20 Modular layout of the *A. Thaliana* network: Node coloring

```
> data("PPI_Athalina")
> data("color_list")
> exp <- rnorm(vcount(g1))
> id <- plot.modules(g1, expression  = exp,  tkplot=FALSE)
>
```

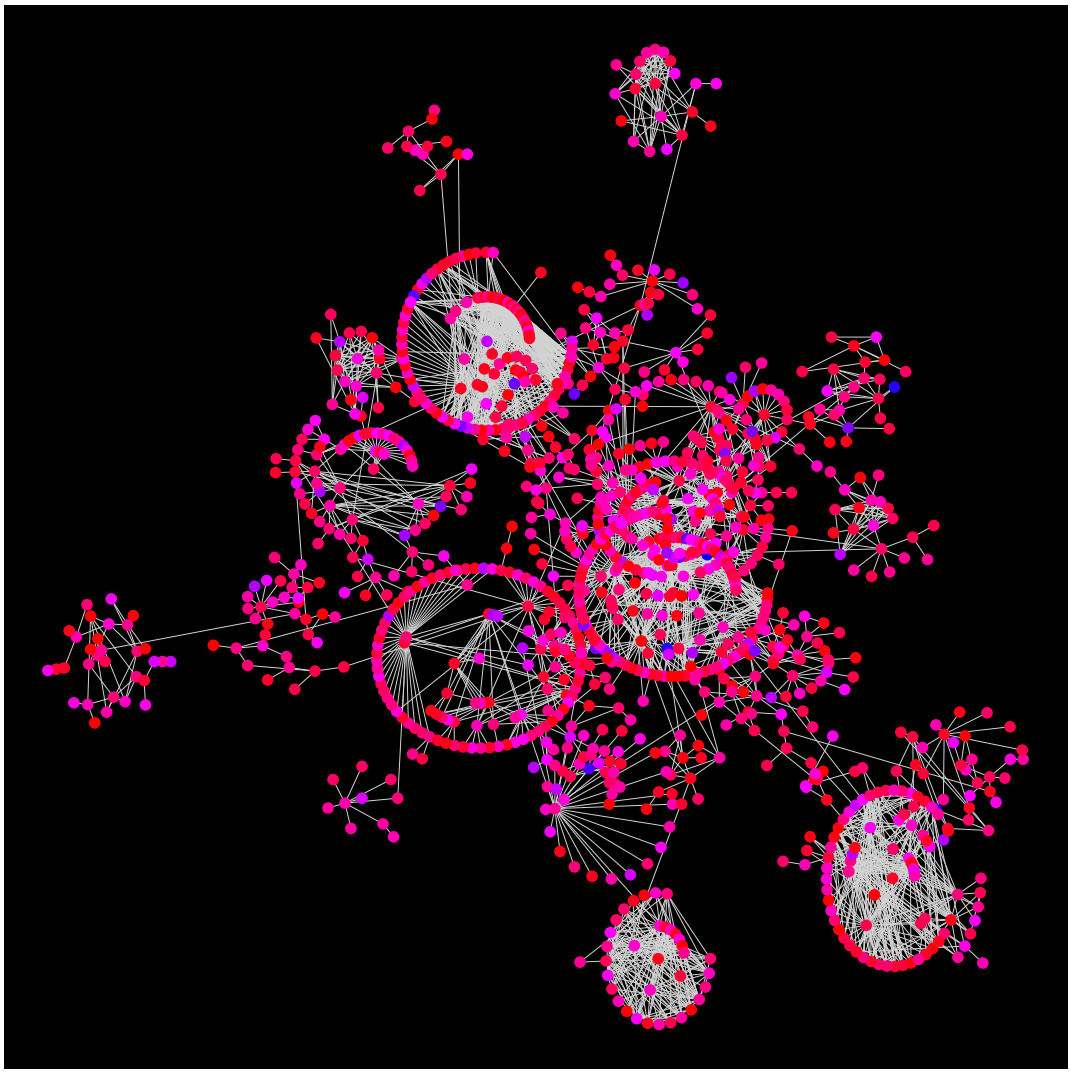

Figure 20: Modular layout of the *A. Thaliana* network. When an expression value is given for all nodes then the nodes are colored based on their expression values from the range of colors (red to blue - smaller to higher expression value).

5.21 Modular layout of the *A. Thaliana* network highlighting gene expression in selected modules

```
> data("PPI_Athalina")
> data("color_list")
> exp <- rnorm(vcount(g1))
> id <- plot.modules(g1, modules.color="grey",
+ expression  = exp, exp.by.module = c(1,2,5),  tkplot=FALSE)
>
```

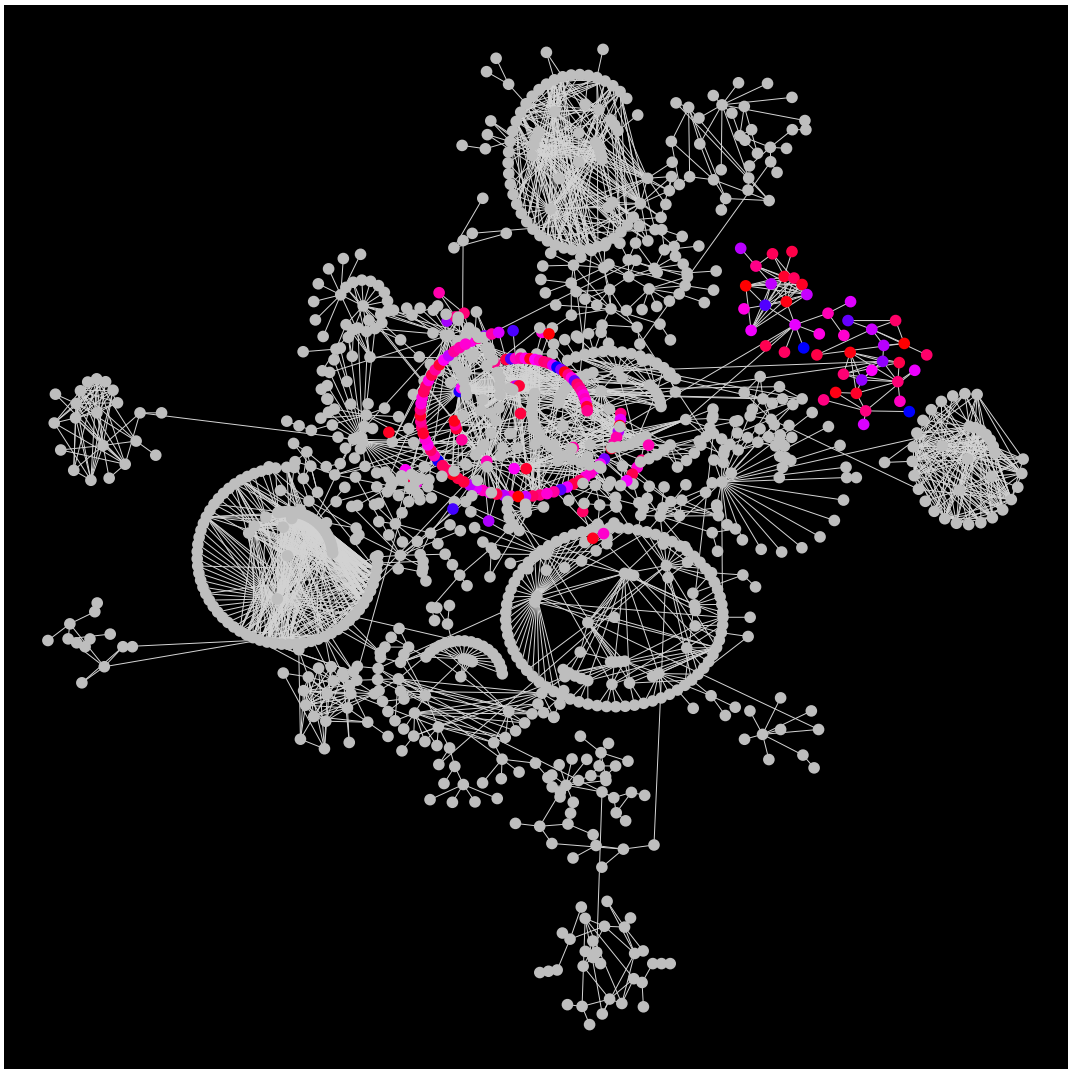

Figure 21: Modular layout of the *A. Thaliana* network highlighting the variation of the expression values of nodes in particular modules. In this example we are highlighting the variation in the modules $1, 2, 5$ (from red to blue - low to high).

5.22 Modular layout with hierarchical plots for the modules of the *A. Thaliana* network

```
> data("PPI_Athalina")
> data("color_list")
> id <- plot.modules(g1, layout.function = layout.reingold.tilford,
+ col.grad=list(color.list$citynight), tkplot=FALSE)
```

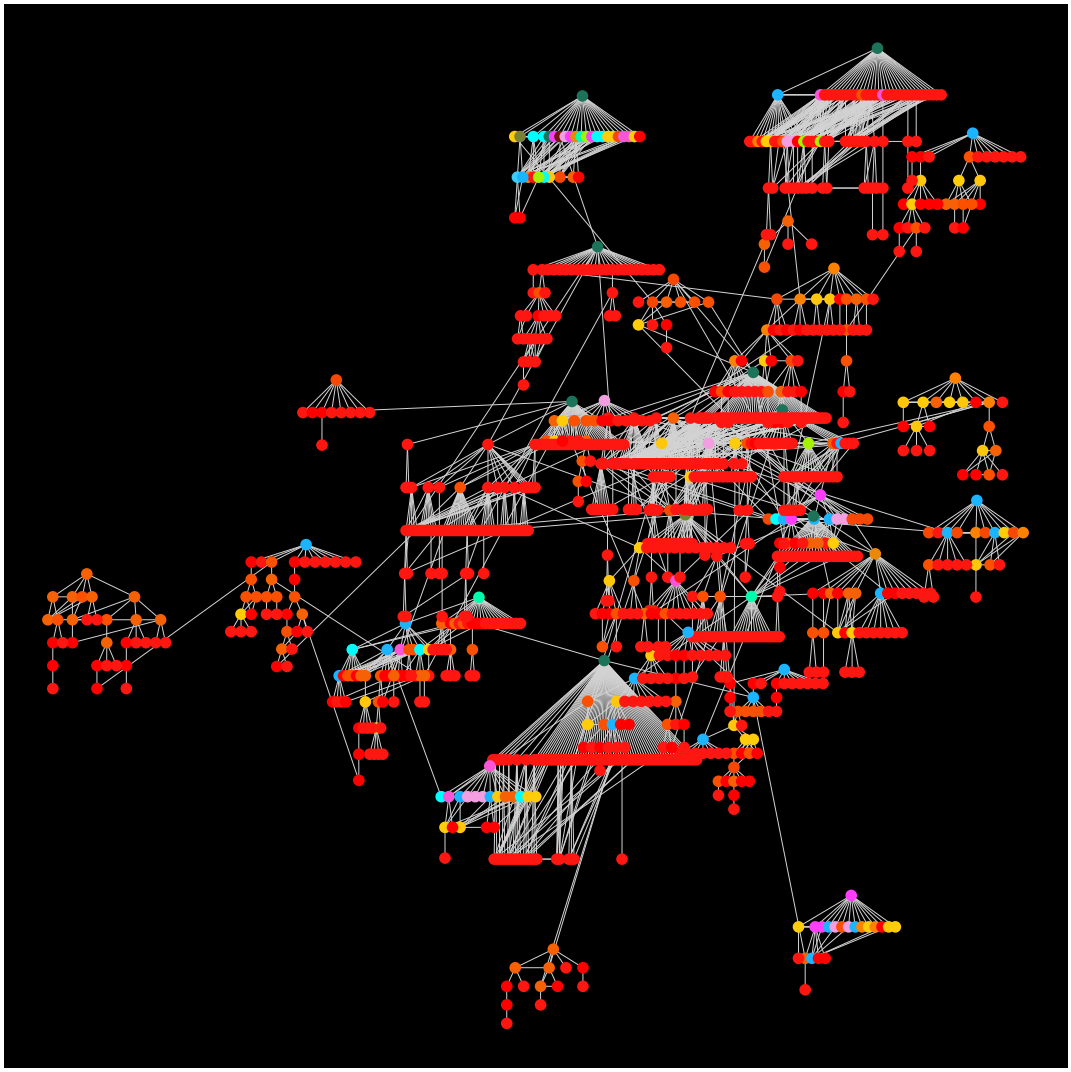

Figure 22: A multiroot-tree (hierarchical) plot of the modules of the *A. Thaliana* network. The nodes are colored in each module based on their degree by providing a range of colors.

5.23 Global view of the *A. Thaliana* network

```
> data("PPI_Athalina")
> data("color_list")
> n = vcount(g1)
> xx <- plot.NetworkSperical(g1, mo="in", v.lab=FALSE, tkplot = FALSE,v.size=1 )
```

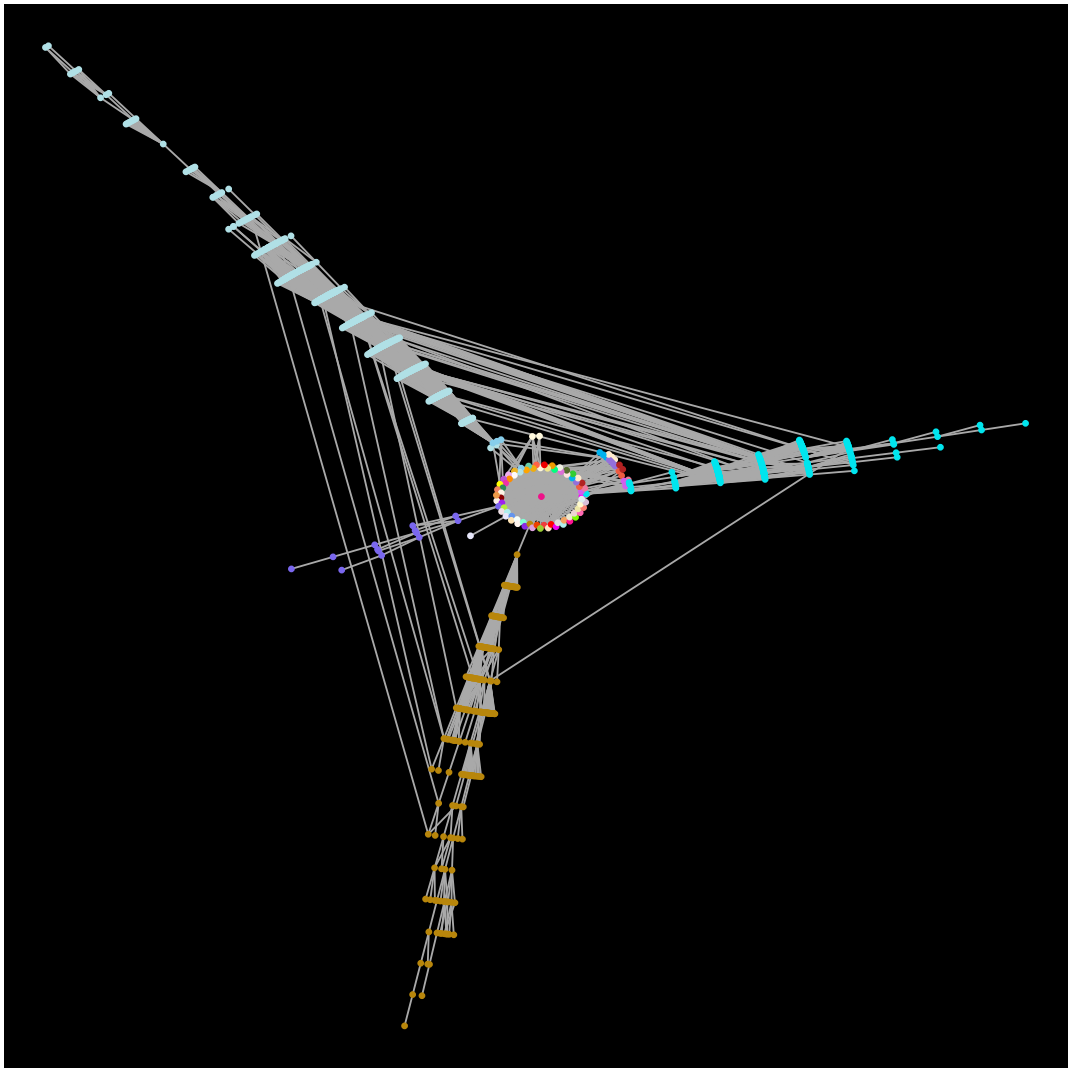

Figure 23: A global view of the *A. Thaliana* network, starting with the node of the highest degree.

5.24 A star-like global view of a scale free network starting with the five highest degree nodes

```
> g <- barabasi.game(500)
> xx <- plot.NetworkSperical.startSet(g, mo = "in", nc = 5)
```

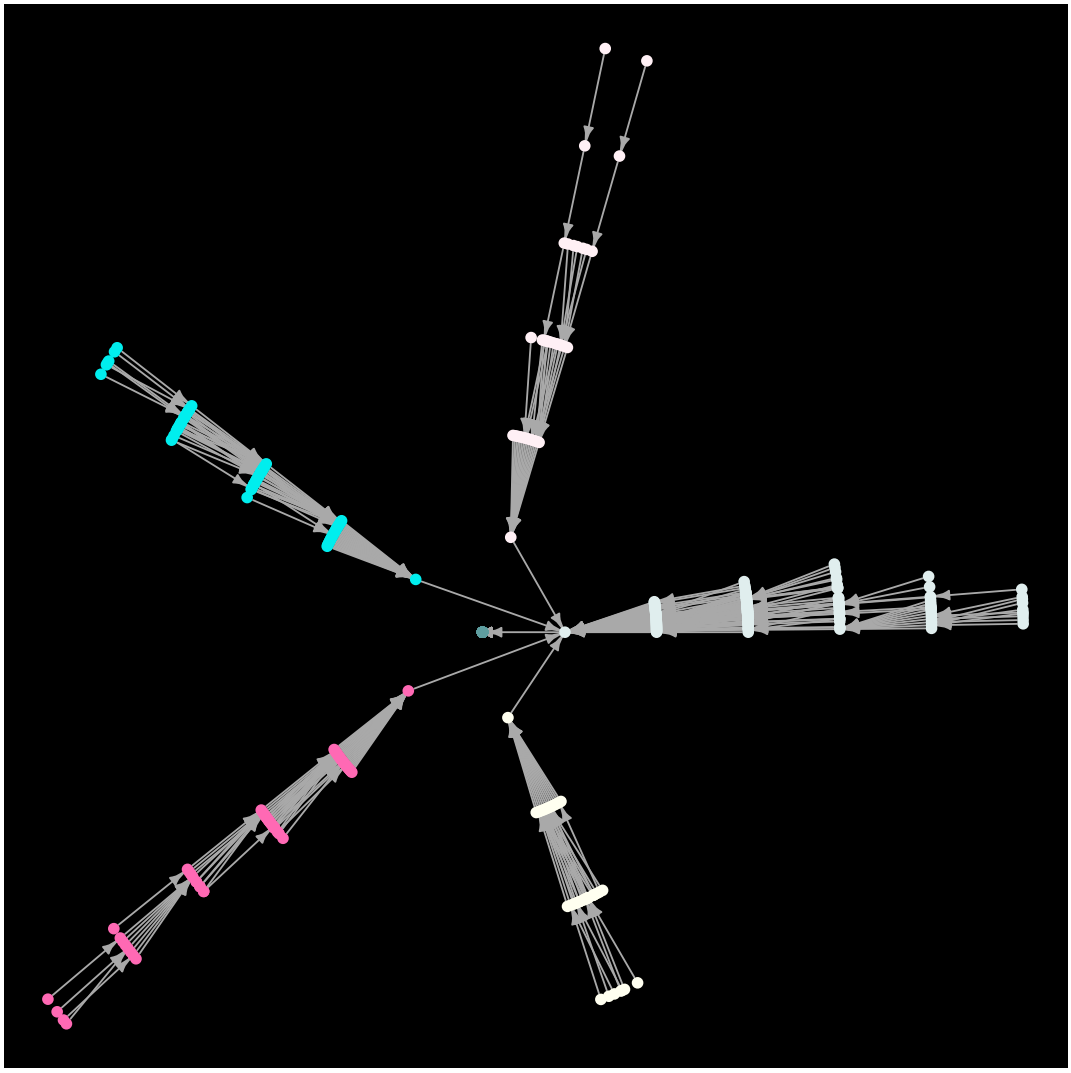

Figure 24: A star-like global view of a scale free network starting with the five highest degree nodes.

5.25 A spiral-view of modules in a network

```
> g <- barabasi.game(3000, directed=FALSE)
> fn <- function(g)plot.spiral.graph(g,60)$layout
> xx <- plot.modules(g, layout.function=fn,
+ layout.overall=layout.fruchterman.reingold,sf=20, v.size=1, color.random=TRUE)
```

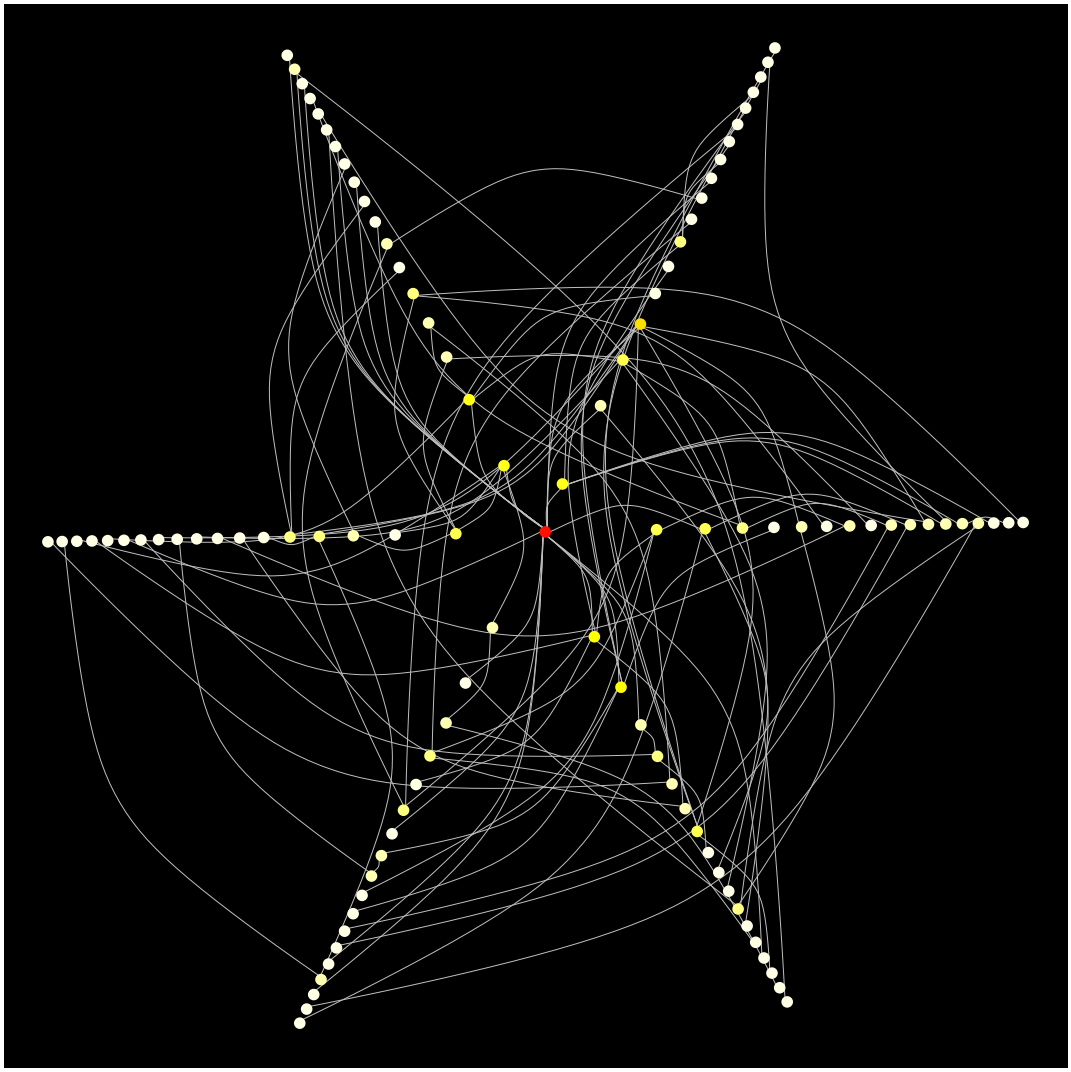

Figure 25: A spiral-view of modules in a network.

5.26 A spiral-view of modules in a network

```
> g <- barabasi.game(3000, directed=FALSE)
> fn <- function(g)plot.spiral.graph(g,12)$layout
> xx <- plot.modules(g, layout.function=fn,
+ layout.overall=layout.fruchterman.reingold,sf=20, v.size=1, color.random=TRUE)
```

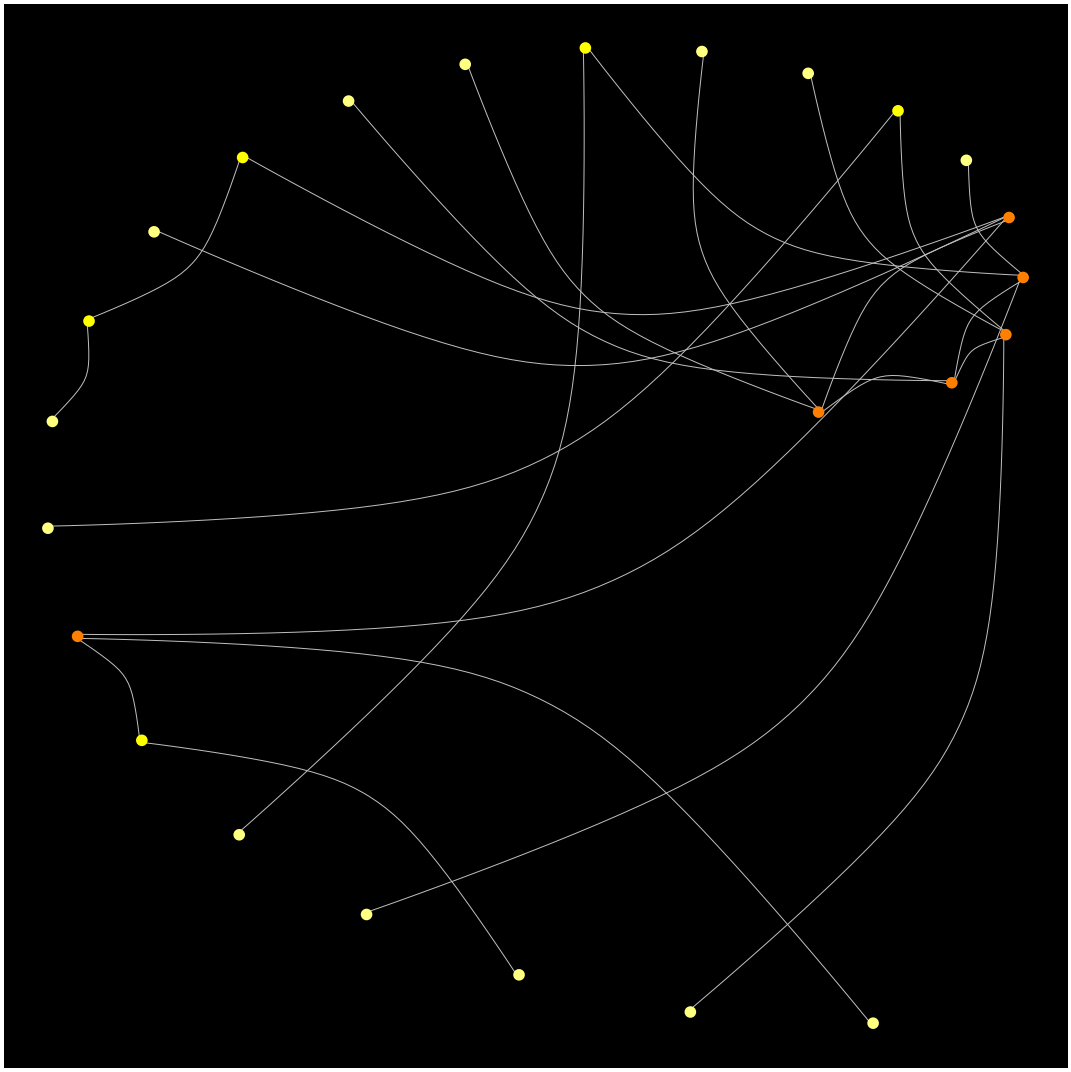

Figure 26: A spiral-view of modules in a network.

5.27 Global layout: A spiral-view of the *A. Thaliana* network starting with the highest degree node

```
> data("PPI_Athalina")
> data("color_list")
> xx <- plot.spiral.graph(g1, tp=179,vertex.color=sample(color.list$bright) )
```

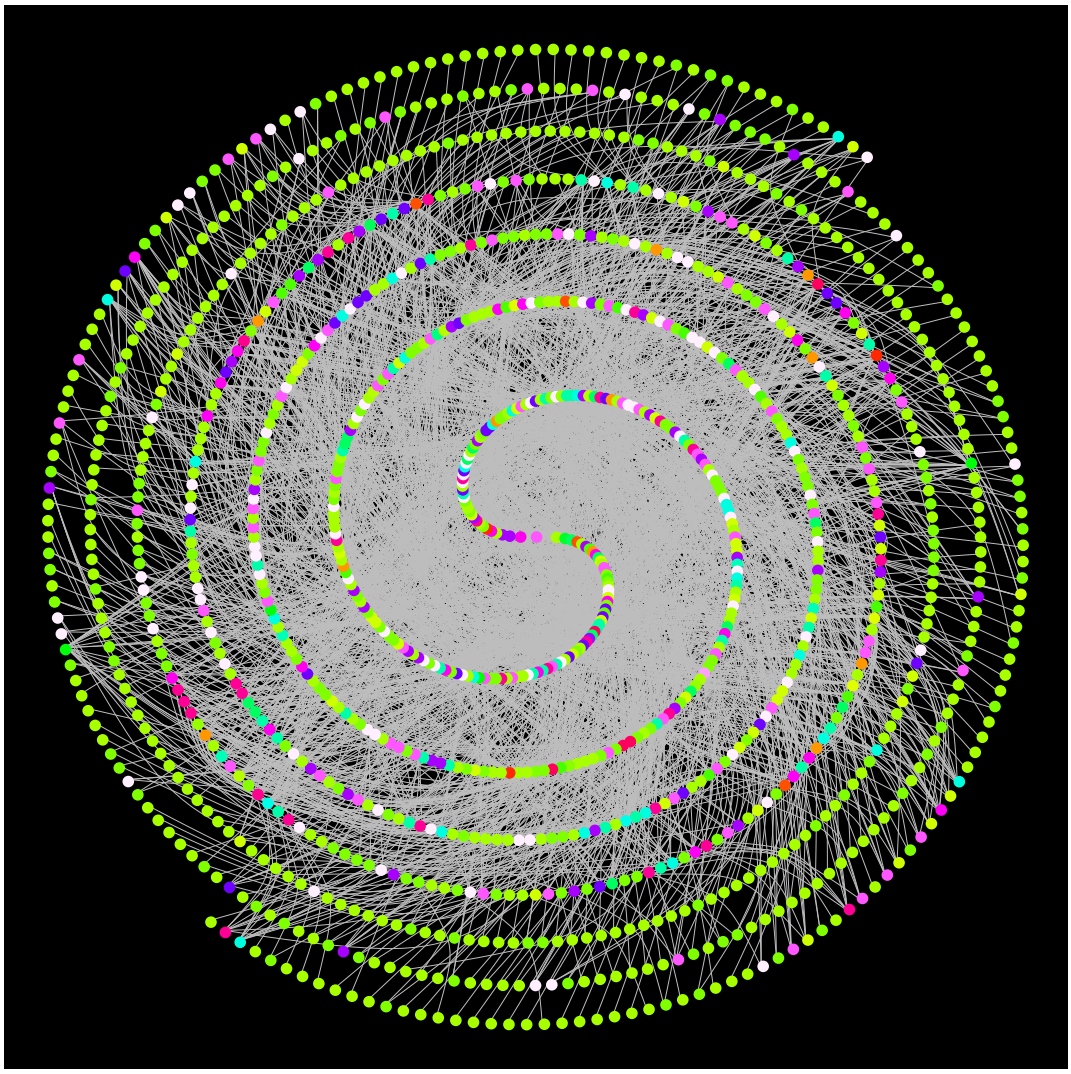

Figure 27: A spiral-view of the *A. Thaliana* network starting with the highest degree node. Vertices are uniquely colored according to their degree.

5.28 Global layout: A spiral-view of the *A. Thaliana* network starting with the highest degree node

```
> data("PPI_Athalina")
> data("color_list")
> xx <- plot.spiral.graph(g1, tp=60,vertex.color=sample(color.list$bright))
```

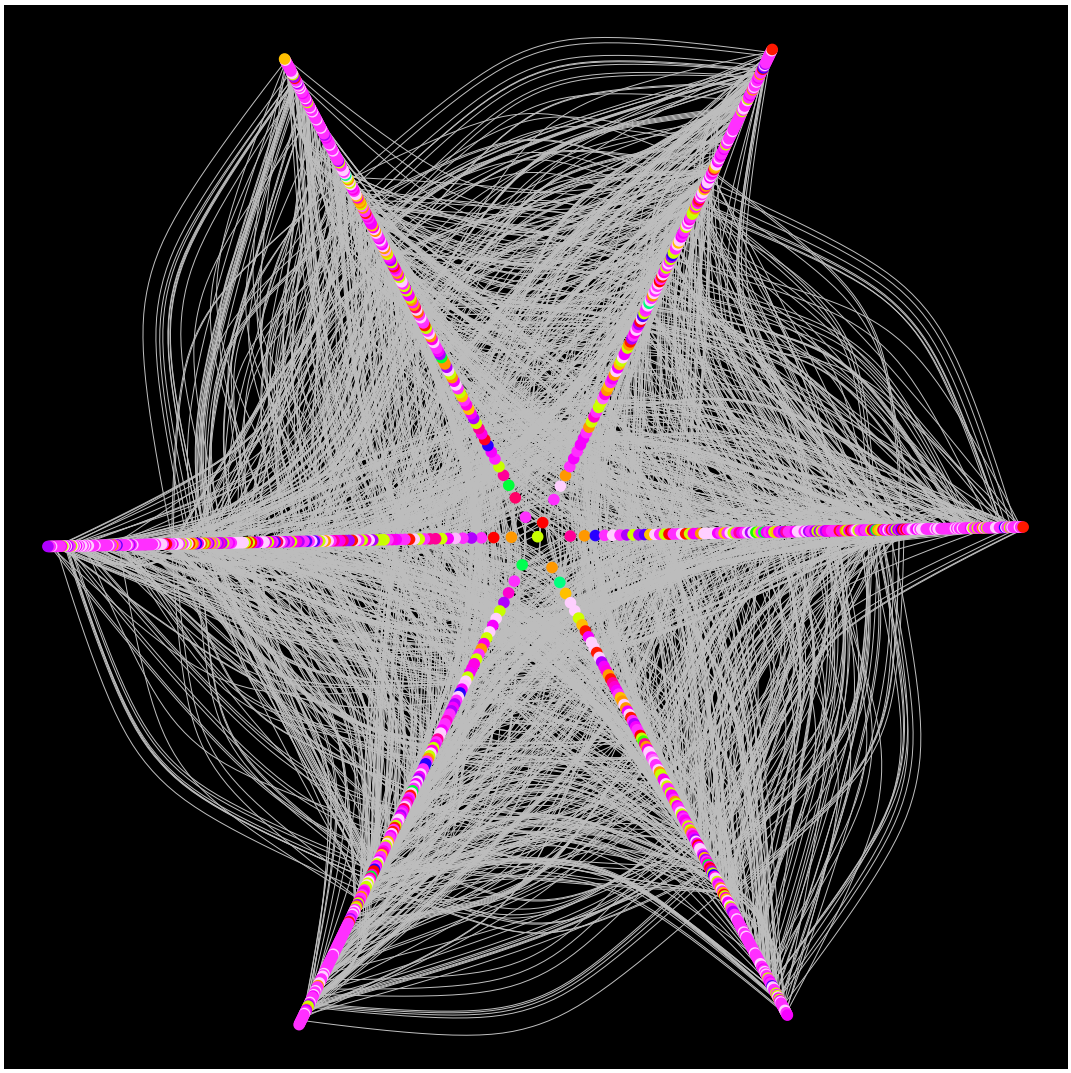

Figure 28: A spiral-view of the *A. Thaliana* network starting with the highest degree node. Vertices are uniquely colored according to their degree.

5.29 Global layout: A spiral-view of the *A. Thaliana*, nodes are ranked using reingold-tilford algorithm

```
> data("PPI_Athalina")
> data("color_list")
> xx <- plot.spiral.graph(g1, tp=90,vertex.color="blue",e.col="gold",
+ rank.function=layout.reingold.tilford)
```

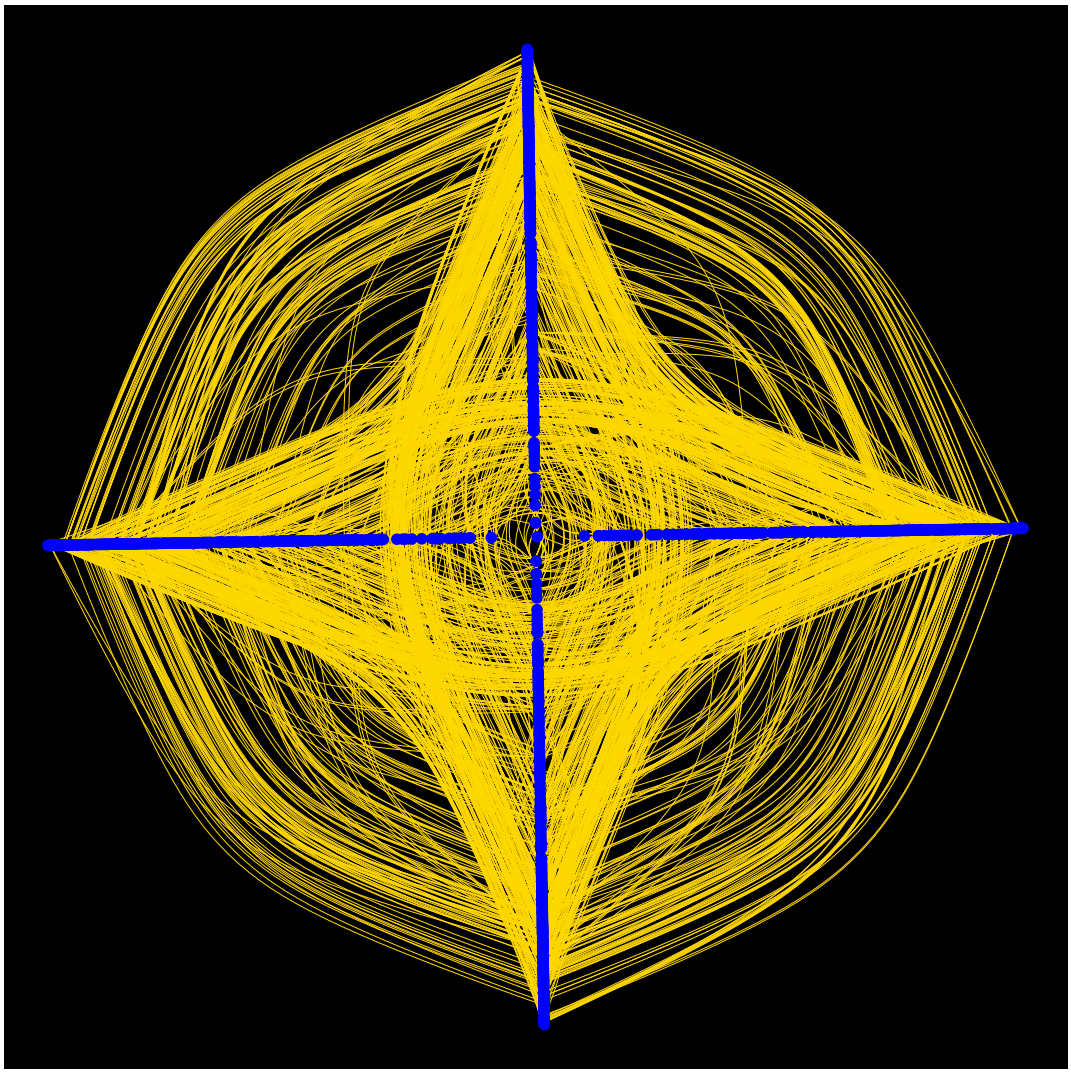

Figure 29: A spiral-view of the *A. Thaliana*. Nodes are ranked using reingold-tilford algorithm .

5.30 An abstract module view of the *A. Thaliana* network

```
> data("PPI_Athalina")
> data("color_list")
> xx <- plot.abstract.module(g1, tkplot = FALSE, layout.function=layout.star)
>
>
```

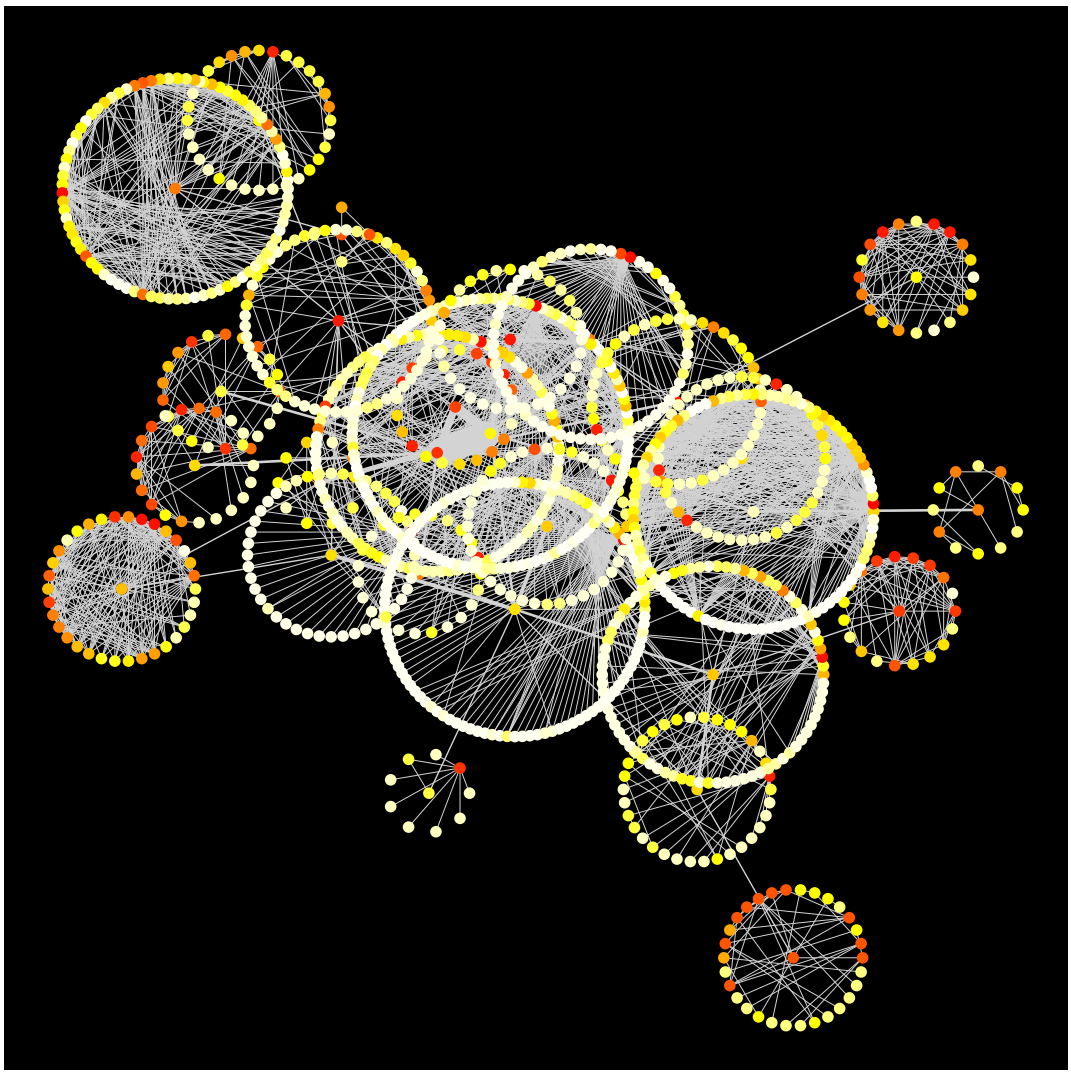

Figure 30: An abstract-view of the *A. Thaliana* network, the position of the nodes for each module is determined by a star-view layout. The edges between the modules are collapsed to single edge. The width of the edges is proportional to the total number of connection between the modules. Nodes in a module are colored by a range of heat-colors, where dark colors indicate a high degree and light colors are indicating a low degree of the node.

5.31 An abstract module of the *A. Thaliana* network

```
> data("PPI_Athalina")
> data("color_list")
> xx <- plot.abstract.nodes(g1,  nodes.color ="grey",layout.function=layout.star,
+ edge.colors= sample(color.list$bright), tkplot =FALSE,lab.color = "red")
```

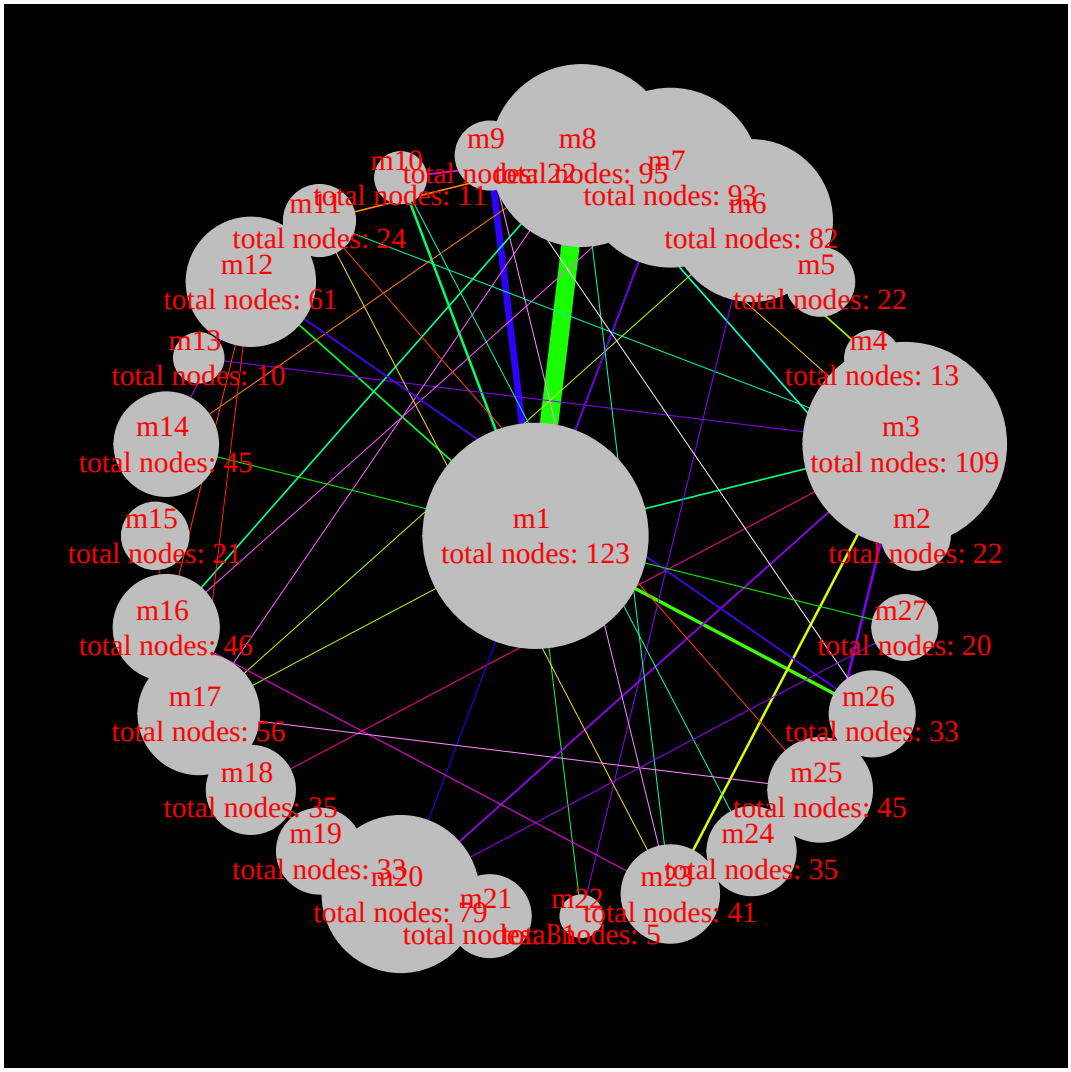

Figure 31: An abstract view of the *A. Thaliana* network where the modules are replaced by single nodes and the connections between the modules are collapsed to single edge. The size of a node is proportional to the total umber of nodes in the module and the edge width is proportional to the total number of connection between the modules. The module containing the largest number of nodes is placed in the center, the other modules are arranged in a circular manner.

5.32 A modular view of the *A. Thaliana* network

```
> data("PPI_Athalina")
> data("color_list")
> xx <- splitg.mst(g1,  vertex.color = sample(color.list$bright),
+ colors = color.list$warm[1:30], tkplot = FALSE)
```

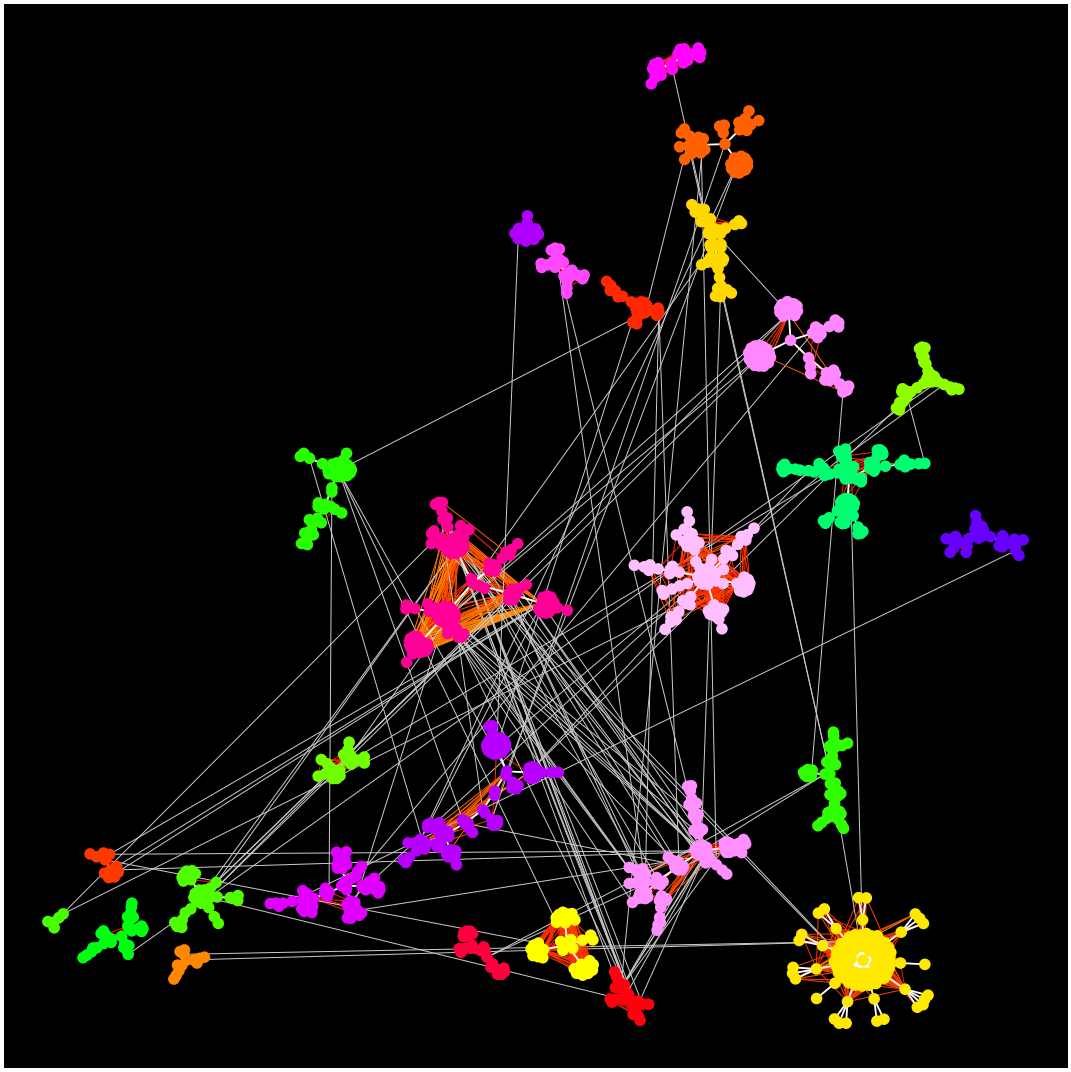

Figure 32: A modular view of the *A. Thaliana* network where the coordinates of the nodes in a module are determined by the Fruchterman-Reingold algorithm of the MST of the module. The MST edges of the modules are plotted in white color and all other edges in the module are plotted with a range of warm color. The colors for the modules are chosen randomly.

5.33 Information flow layout: Level plot of the *A. Thaliana* network

```
> data("PPI_Athalina")
> xx <- level.plot(g, tkplot=FALSE, level.spread=FALSE,
+ layout.function=layout.fruchterman.reingold)
```

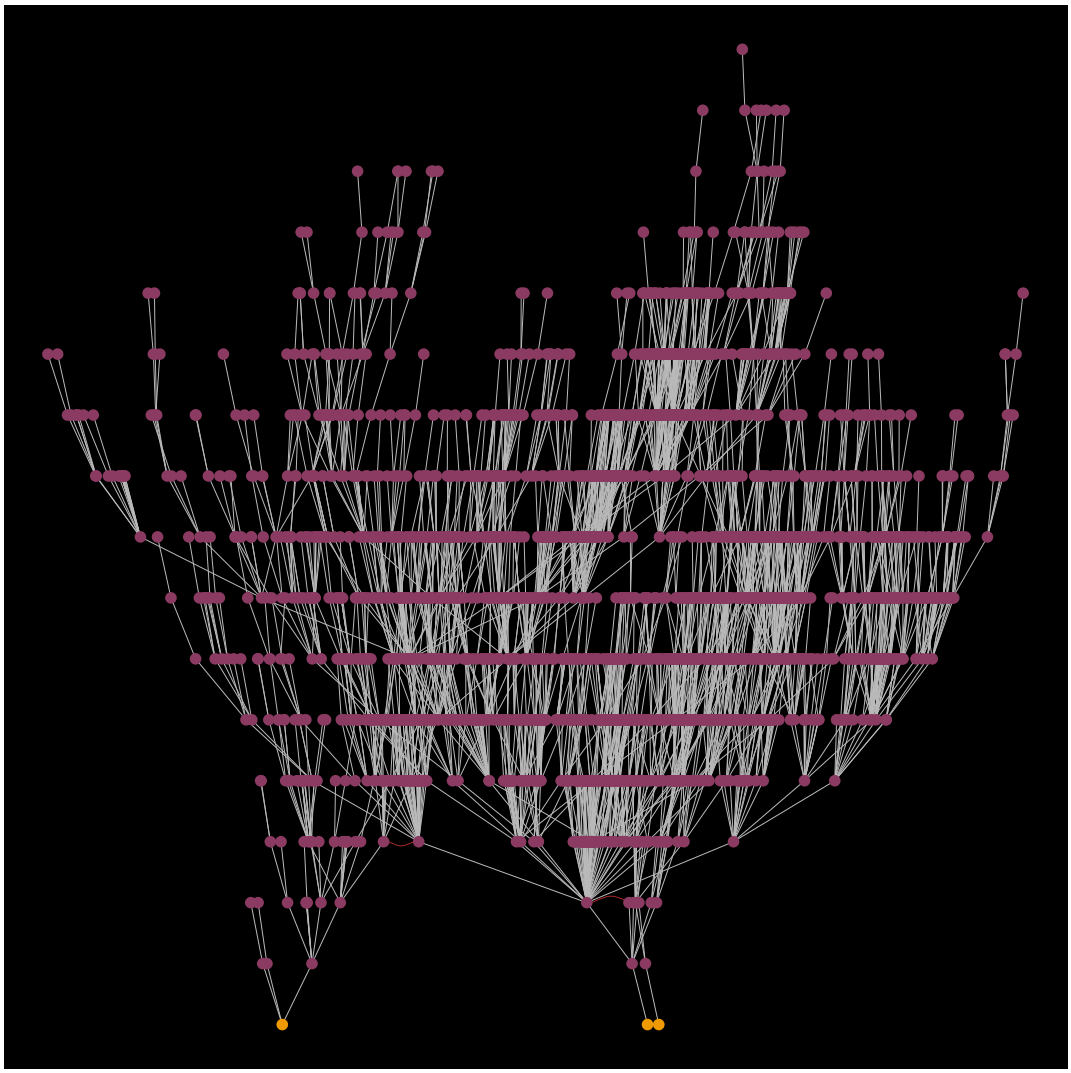

Figure 33: Level plot of the *A. Thaliana* network. The initial nodes on level 1 are picked randomly and their adjacent neighbors of these nodes are iteratively plotted on consequtive levels. The initial nodes are colored in orange, all other nodes are shown in maroon. The edges connecting nodes on the same level are shown in blue with a curved shape.

6 Algorithmic description of the main graph-layouts

6.1 Global view

Algorithm 1 Minimum spanning tree global graph layout

n := total no. of vertices.
E := total no. of edges of a graph G.
G_{mst} : minimum spanning tree of a graph G, with V vertices and E_{mst} edges.
$ecol_{mst}$:= $COLOR$ (assign a unique color to the edges (E_{mst}) of the minimum spanning tree graph).
$coord$:= apply forced based algorithm (e.g. Fruchterman-Reingold) on G_{mst}, and get the position of nodes.
E_{rest} := $E - E_{mst}$.
$color\_vector$:= a color vector with different shades of a color.
repeat
  assign a color to $E_{rest}(i)$ from $color\_vector$, based on the distance between their connecting nodes.
until all E_{rest} are colored.
plot the graph G, with $coord$ and their edge colors.

6.2 Modular view

Algorithm 2 Modular graph layout

E = total number of edges of the graph G.
M = a vector of modules of the graph G.
$COORD$:= get coordinates for location of modules M (the location of modules can be identified by standard layout or user defined algorithms on an abstract graph creating from modules where each modules is replaced by a node and edges between modules by a single edge).
$RANK$:= rank $COORD$ depending on the features of modules (for example module with maximum no of node will be plotted in a specific location).
repeat
M_k : pick k^{th} module from M based on its $RANK(k)$.
$RANK_{M_k}$: assign ranks to the nodes of the module M_k.
$COORD_{M_k}$:= get coordinates for the nodes of k^{th} module using any standard layout algorithm.
$COORD_{M_k}$:= transform each coordinate of $COORD_{M_k}$ to the coordinate location $COORD(k)$ of module k depending on its rank $RANK(k)$.
$VCOL_{M_k}$:= assign colors to the nodes of $M_k(i)$ based on their ranks $RANK_{M_k}(i)$.
$ECOL_{M_k}$:= assign colors to the edges of M_k.
until all modules are assigned their properties.
$ECOL_{rest}$:= assign colors to the edges of G which are joining two modules.
plot the graph G with the assigned color to the vertices and edges.

6.3 Multiroot Tree (Hierarchical) view

Algorithm 3 To visualize network with respect to the spread of information within the network

n = total no. of vertices of a graph $G = (V, E)$
V_k = Pick k vertices randomly from V
$level$ = 0 (initial level)
plot V_k linearly at $level$
$levelup$ = 1
$leveldown$ = -1
V^{out}_{kup} = outgoing nodes of V_k
V^{in}_{kdown} = incoming nodes of V_k
V^{in}_{kup} = NULL
V^{out}_{kdown} = NULL
plot V_{kup} linearly at $levelup$
plot V_{kdown} linearly at $leveldown$
repeat
  V^{out}_{kup} = outgoing nodes of V^{out}_{kup}
  V^{in}_{kup} = incoming nodes of V^{out}_{kup}
  V^{out}_{kdown} = outgoing nodes of V^{in}_{kdown}
  V^{in}_{kdown} = incoming nodes of V^{in}_{kdown}
  Assign $levelup$ + 1 as Y coordinates to V^{out}_{kup}
  Assign $levelup$ - 1 as Y coordinates to V^{in}_{kup}
  Assign $leveldown$ - 1 as Y coordinates to V^{in}_{kdown}
  Assign $leveldown$ - 1 as Y coordinates to $leveldown$ + 1
  $levelup$ = $levelup$ + 1
  $leveldown$ = $leveldown$ -1
until V_{kup} = 0 AND V_{kdown} = 0
Obtain X coordinates of G by using any force-based algorithm.
Plot nodes of G to their corresponding X, Y locations.

6.4 Information flow view

Algorithm 4 To visualize network with respect to the spread of information within the network

n = total no. of vertices of a graph $G = (V, E)$
$ECOL$ = is a vector of colors of edges.
$VCOL$ = is a vector of colors of nodes.
{}V_{source} and V_{dest} are source and desination nodes between which the information flow to be visualized.
V_{source} = A vector of k_1 nodes of G
V_{dest} = A vector of k_2 nodes of G
Apply shortest path distance algorithm $F_{shortest}(G, V_{source}, V_{dest})$ on G to find shortest paths between V_{source} and V_{des}
E_{sp} := $F_{shortest}(G, V_{source}, V_{dest})$ (A vector of edge objects of shortest path between V_{source} and V_{des})
V_{path} is a set of nodes which connectes V_{source} and V_{des} through E_{sp}
$ECOL_{E_{sp}}$:= assign a color to the edges of E_{sp}.
$VCOL_{V_{path}}$:= assign a color to the nodes of V_{path}.
$COORD$ = Find coordinates of G using Algorihm 2 or 3
Plot nodes of G to their corresponding locations specified by $COORD$.

```
> sessionInfo()

R version 3.4.0 (2017-04-21)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: OS X El Capitan 10.11.6

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] netbiov_1.10.0 igraph_1.0.1

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.10    lattice_0.20-35 digest_0.6.12   rprojroot_1.2   grid_3.4.0
 [6] backports_1.0.5 magrittr_1.5    evaluate_0.10   stringi_1.1.5   Matrix_1.2-9
[11] rmarkdown_1.4   BiocStyle_2.4.0 tools_3.4.0     stringr_1.2.0   yaml_2.1.14
[16] compiler_3.4.0  htmltools_0.3.5 knitr_1.15.1
```

References

[1] Ricardo de Matos Simoes and Frank Emmert-Streib. Bagging statistical network inference from large-scale gene expression data. *PLoS ONE*, 7(3):e33624, 03 2012. URL: http://dx.doi.org/10.1371%2Fjournal.pone.0033624, doi:10.1371/journal.pone.0033624.

[2] Bobby-Joe Breitkreutz, Chris Stark, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, Michael Livstone, Rose Oughtred, Daniel H. Lackner, JÃ¼rg BÃ¤hler, Valerie Wood, Kara Dolinski, and Mike Tyers. The biogrid interaction database: 2008 update. *Nucleic Acids Research*, 36(suppl 1):D637–D640, 2008. URL: http://nar.oxfordjournals.org/content/36/suppl_1/D637.abstract, arXiv:http://nar.oxfordjournals.org/content/36/suppl_1/D637.full.pdf+html, doi:10.1093/nar/gkm1001.