

Complex Networks

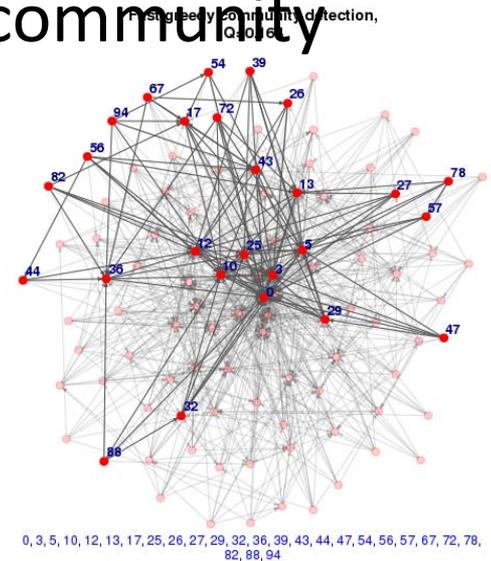
tools for analyzing networks

(R+igraph)

2011.10.17

igraph <http://cneurocv.s.rmki.kfki.hu/igraph/>

- igraph is a free software package for creating and manipulating undirected and directed graphs. It includes implementations for classic graph theory problems like minimum spanning trees and network flow, and also implements algorithms for some recent network analysis methods, like community structure search.



tutorials

- tutorials of R
 - <http://cran.r-project.org/other-docs.html> (many tutorials in English and other languages)
- tutorial of igraph
 - <http://igraph.sourceforge.net/igraphbook/> (English, under development)

procedure for analyzing network

1. create graph object
2. layout the network
3. ranking
4. metrics
5. community detection
6. export

0. starting igraph

main

graph layout

The screenshot displays the RGui interface. The R Console window on the left shows the following code and output:

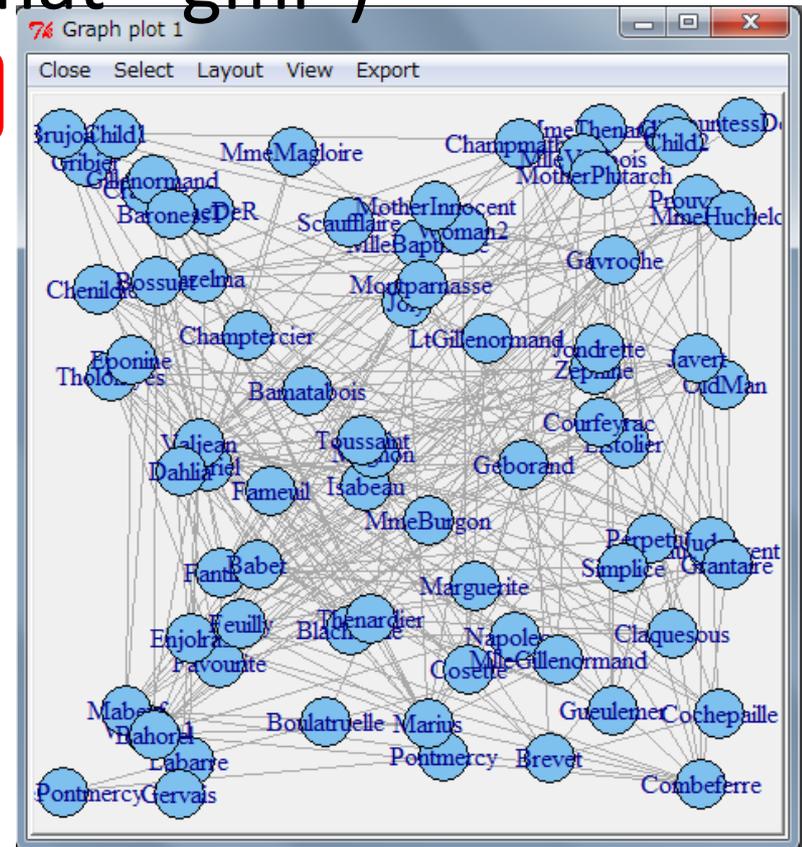
```
> setwd("R")
> library("igraph")
> g<-read.graph("lesmis.gml",format="gml")
> tkplot(g)
Loading required package: tcltk
Tcl/Tkインターフェースのロード中 終了済
[1] 1
> |
```

The graph plot window on the right, titled "Graph plot 1", shows a network graph with numerous nodes and edges. The nodes are labeled with names such as "MmeMagloire", "MmeHuchel", "MmeBurgon", and "MmeGillormand". The graph is a dense network of interconnected nodes, representing a social network.

1. create graph object

- `library("igraph")` start igraph
- `setwd("R")` set directory
- `g<-read("lesmis.gml", format="gml")` create graph
- `tkplot(g)` open graph window

lesmis.gml is available at
Mark Newman's Website



2. layout the network

set parameters

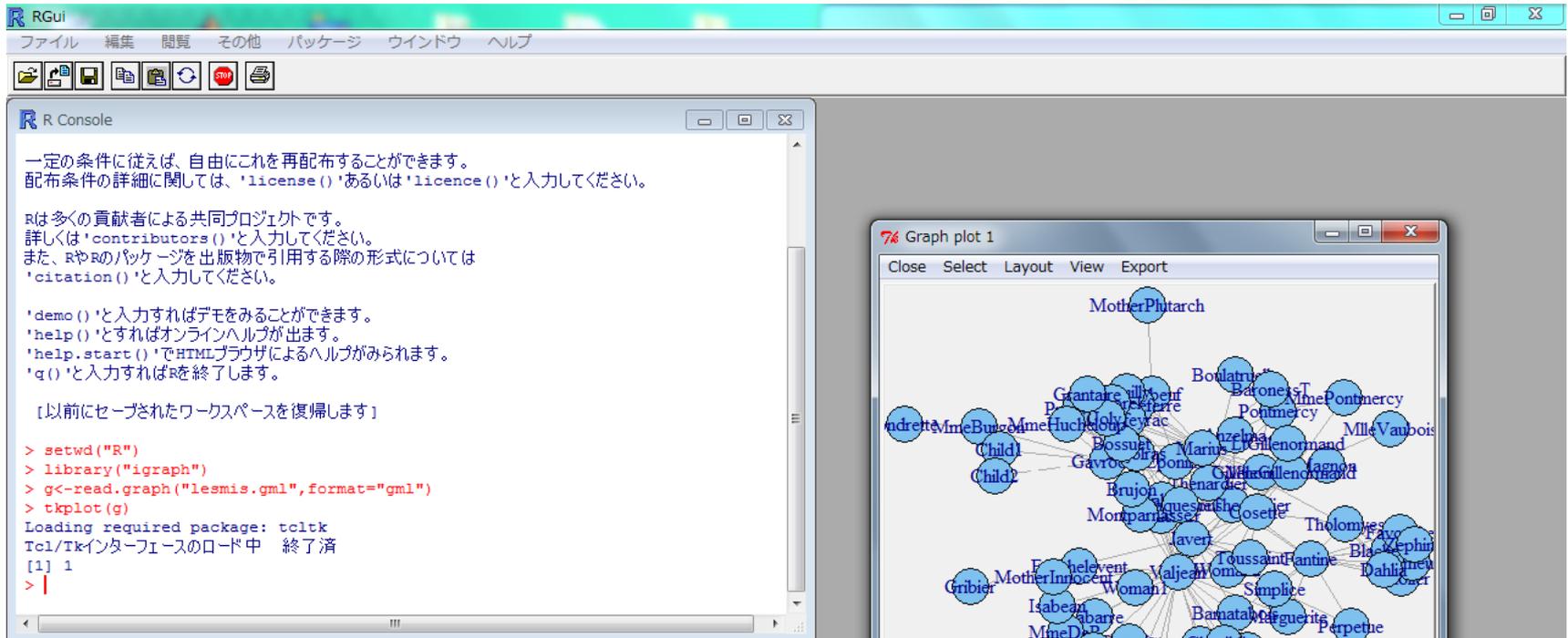
choose layout

The screenshot displays the RGui interface. On the left, a dialog box titled "Layout parameters for graph plot 1" is open, showing the "Fruchterman-Reingold layout" settings. The parameters are: Number of iterations: 500, Maximum change (n): 77, Area parameter (n^2): 5929, Cooling exponent: 3, and Cancellation radius (n^3): 456533. Below the dialog, the R console shows the following code and output:

```
> setwd("R")
> library("igraph")
> g<-read.graph("lesmis.gml", format="gml")
> tkplot(g)
Loading required package: tcltk
Tel/Tkインターフェースのロード中 終了済
[1] 1
> |
```

On the right, a window titled "Graph plot 4" shows a network graph with nodes labeled with names like "ephie", "Courfeyrac", "Bahorel", etc. A context menu is open over the graph, listing layout options: Random, Circle, Fruchterman-Reingold, Kamada-Kawai, Spring Embedder, and Reingold-Tilford. A red arrow points from the "choose layout" text to the "Fruchterman-Reingold" option in the menu.

2. layout the network



The screenshot shows the RGui environment with the R Console on the left and a network graph plot on the right.

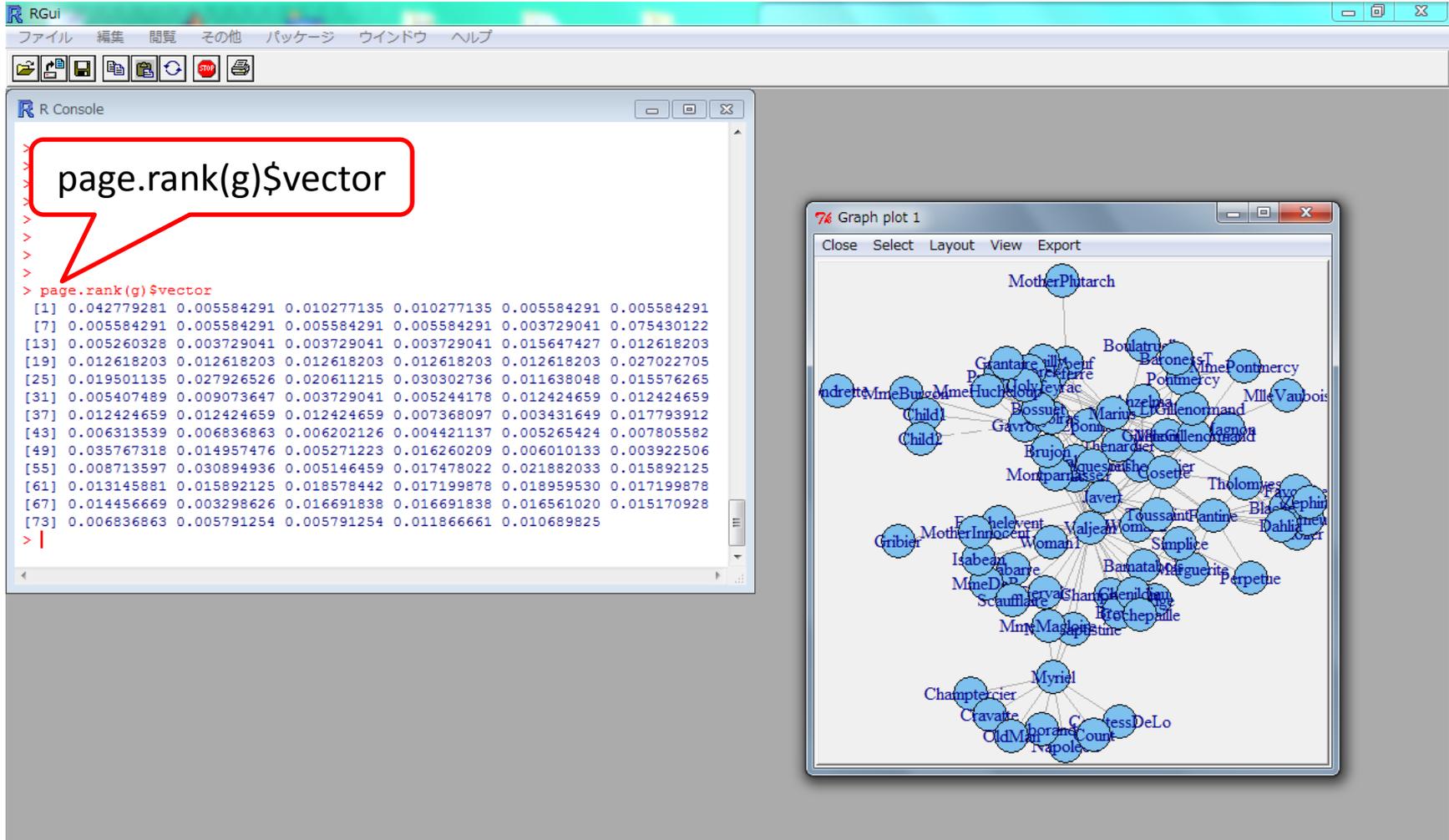
R Console Output:

```
> setwd("R")
> library("igraph")
> g<-read.graph("lesmis.gml", format="gml")
> tkplot(g)
Loading required package: tcltk
Tcl/Tkインターフェイスのロード中 終了済
[1] 1
> |
```

Network Graph Plot:

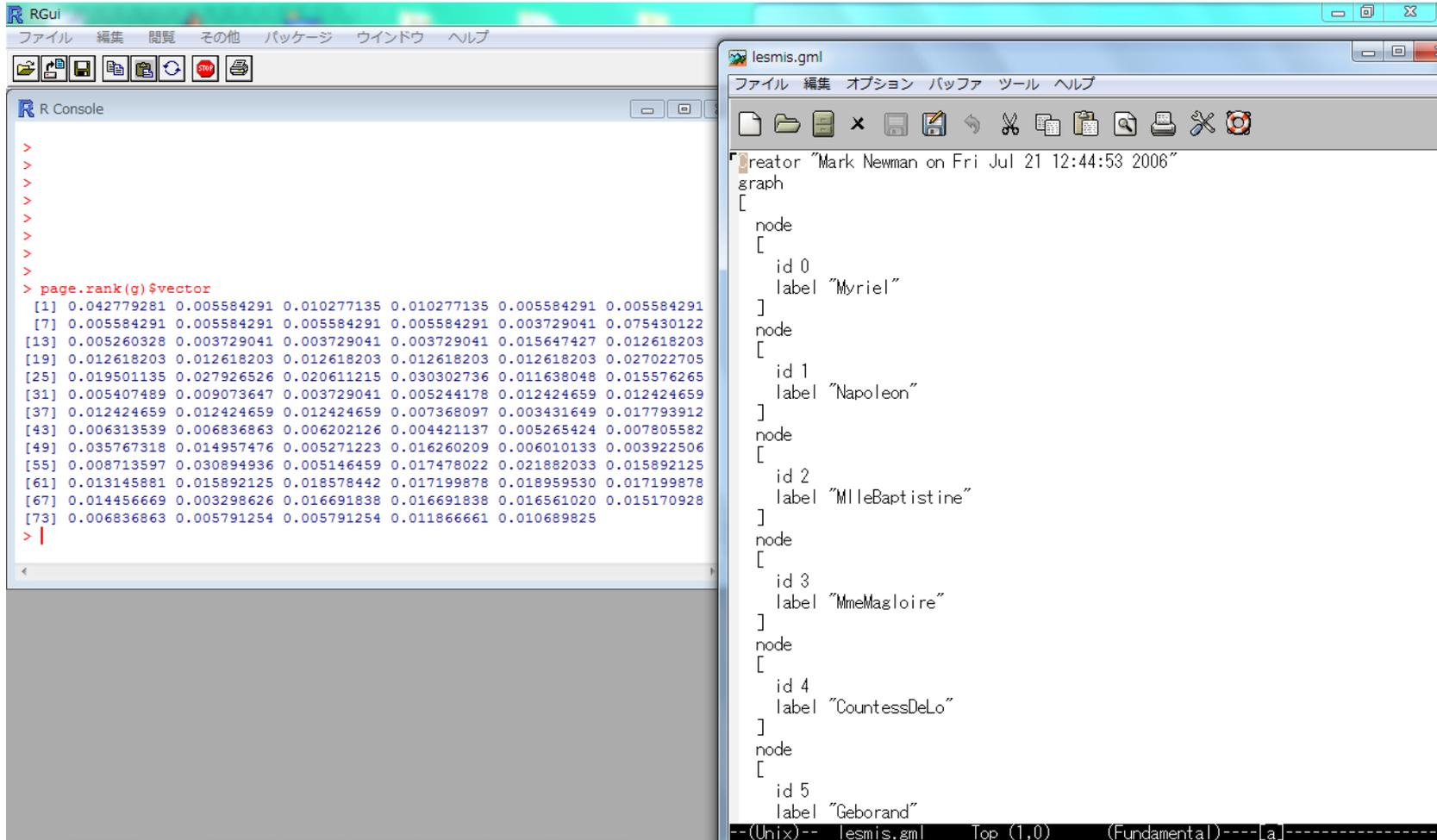
The graph plot, titled "7% Graph plot 1", displays a network of nodes and edges. The nodes are labeled with names, including MotherPhtarch, Boulatru, MmePontmercy, MlleVaubois, Child1, Child2, Gribier, MotherInnocent, Myriel, and others. The plot includes a menu bar with options: Close, Select, Layout, View, Export.

3. ranking



3.ranking

- original gml file contain labels



The image shows two windows from a computer screen. The left window is RGui, and the right window is a text editor showing a GML file named 'lesmis.gml'.

R Console Output:

```
>
>
>
>
>
>
>
> page.rank(g)$vector
[1] 0.042779281 0.005584291 0.010277135 0.010277135 0.005584291 0.005584291
[7] 0.005584291 0.005584291 0.005584291 0.005584291 0.003729041 0.075430122
[13] 0.005260328 0.003729041 0.003729041 0.003729041 0.015647427 0.012618203
[19] 0.012618203 0.012618203 0.012618203 0.012618203 0.012618203 0.027022705
[25] 0.019501135 0.027926526 0.020611215 0.030302736 0.011638048 0.015576265
[31] 0.005407489 0.009073647 0.003729041 0.005244178 0.012424659 0.012424659
[37] 0.012424659 0.012424659 0.012424659 0.007368097 0.003431649 0.017793912
[43] 0.006313539 0.006836863 0.006202126 0.004421137 0.005265424 0.007805582
[49] 0.035767318 0.014957476 0.005271223 0.016260209 0.006010133 0.003922506
[55] 0.008713597 0.030894936 0.005146459 0.017478022 0.021882033 0.015892125
[61] 0.013145881 0.015892125 0.018578442 0.017199878 0.018959530 0.017199878
[67] 0.014456669 0.003298626 0.016691838 0.016691838 0.016561020 0.015170928
[73] 0.006836863 0.005791254 0.005791254 0.011866661 0.010689825
> |
```

GML File Content (lesmis.gml):

```
creator "Mark Newman on Fri Jul 21 12:44:53 2006"
graph
[
  node
  [
    id 0
    label "Myriel"
  ]
  node
  [
    id 1
    label "Napoleon"
  ]
  node
  [
    id 2
    label "MlleBaptistine"
  ]
  node
  [
    id 3
    label "MmeMagloire"
  ]
  node
  [
    id 4
    label "CountessDeLo"
  ]
  node
  [
    id 5
    label "Geborand"
  ]

```

At the bottom of the text editor window, the status bar shows: --(Unix)-- lesmis.gml Top (1,0) (Fundamental)----[a]-----

4. metrics

- diameter(g)
- graph.density(g)
- average.path.length(g)
- transitivity(g)  clustering coefficient
- help
 - ??rank
 - help(“page.rank”)

4. metrics

The image shows the RGui interface. On the left, the R Console displays the following commands and their outputs:

```
> diameter(g)
[1] 5
> graph.density(g)
[1] 0.08680793
> average.path.length(g)
[1] 2.641148
> transitivity(g)
[1] 0.4989316
> |
```

On the right, the 'Graph plot 1' window displays a network graph with nodes and edges. The nodes are labeled with names, including: Mother, Phlarch, Boulatru, Barones, T, Mme Pontmercy, Pontmercy, Mlle Vaubois, Grantare, Villenore, P, Hohl, Yac, Bossuet, Marn, Gillemand, Gavro, Brujon, Vuesnthe, Gosette, Tholomey, Laver, Toussant, Pantine, Bl, Kephin, Dahn, Gribier, Mother Innocent, Woman, Valjean, Simple, Isabeau, Babar, Banata, Marguerite, Perpetue, Mme De, Scauffla, Ghar, Genil, Rotherpille, Myrielle, Myrielle, Champertier, Cravate, Old, Grand, Count, and Napole.

5. community detection

modularity optimization

store membership and size

```
> gr<-fastgreedy.community(g)
> m<-community.to.membership(g,gr$merges,steps=nrow(gr$merges)-1)
> m
```

```
$membership
```

```
[1] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
$csize
```

```
[1] 17 17
```

```
> V(g)$color<-m$membership
```

store membership

```
> V(g)$color
```

```
[1] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

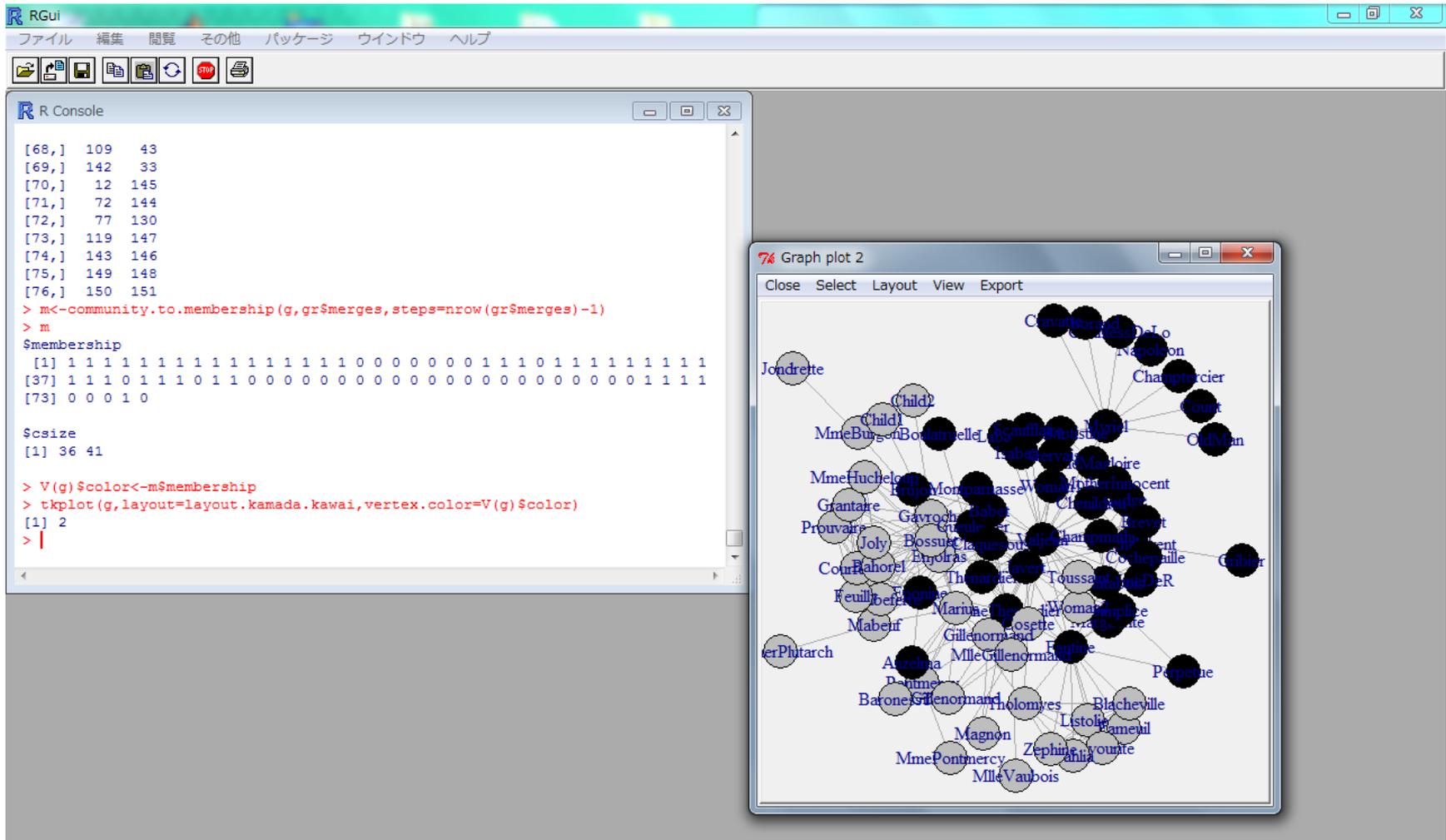
```
> tkplot(g,layout=layout.kamada.kawai,vertex.color=V(g)$color)
```

```
[1] 2
```

visualize network

```
>
```

5. community detection



6. export

The image shows the RGui interface. On the left, the R Console displays a list of data points:

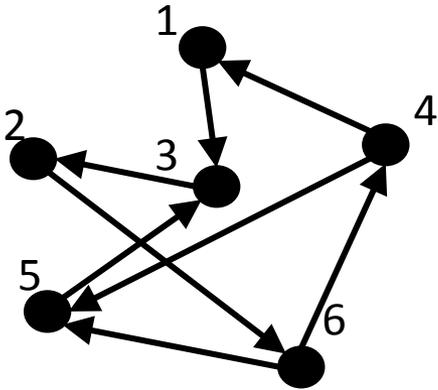
```
[232] 41 -- 71
[233] 25 -- 71
[234] 26 -- 72
[235] 27 -- 72
[236] 11 -- 72
[237] 48 -- 73
[238] 48 -- 74
[239] 73 -- 74
[240] 69 -- 75
[241] 68 -- 75
[242] 25 -- 75
[243] 48 -- 75
[244] 41 -- 75
[245] 70 -- 75
[246] 71 -- 75
[247] 64 -- 76
[248] 65 -- 76
[249] 66 -- 76
[250] 63 -- 76
[251] 62 -- 76
[252] 48 -- 76
[253] 58 -- 76
> |
```

On the right, a window titled "7% Graph plot 1" displays a network graph with blue circular nodes and connecting lines. The nodes are labeled with names such as "M...", "Gantake", "Boulatri", "Barones", "T. Pontmercy", "Ponmercy", "Mlle Vaubois", "Gribier", "Mother Innocent", "Womani", "Valjean", "Toussant", "Antine", "Blas", "Ephim", "Dahlan", "Cort", "Isabeau", "Sabary", "Mme De", "Scauffard", "Mme Magloire", "Myriel", "Champtercier", "Cravatte", "Portant", "Count", "Napole", "Cress", "DeLo". The "Export" menu is open, showing the "Postscript" option.

create from adjacency matrix(1)

- from adjacency matrix

```
> a <- matrix(c(0,0,0,1,0,0,  
                0,0,1,0,0,0,  
                1,0,0,0,1,0,  
                0,0,0,0,0,1,  
                0,0,0,1,0,1,  
                0,1,0,0,0,0),nrow=6,byrow=TRUE)
```



$$A = \begin{matrix} & & & \color{red}{j} & & \\ \color{red}{i} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{matrix}$$

create from adjacency matrix(2)

```
> ga <- graph.adjacency(t(a))
```

```
> ga
```

```
Vertices: 6
```

```
Edges: 8
```

```
Directed: TRUE
```

```
Edges:
```

```
[0] 0 -> 2
```

```
[1] 1 -> 5
```

```
[2] 2 -> 1
```

```
[3] 3 -> 0
```

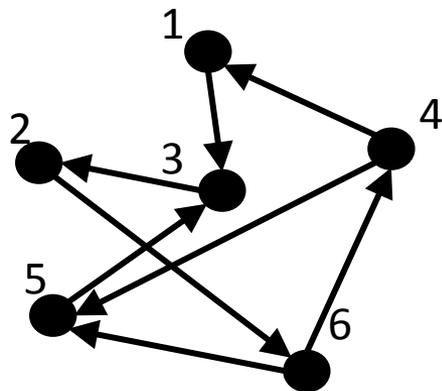
```
[4] 3 -> 4
```

```
[5] 4 -> 2
```

```
[6] 5 -> 3
```

```
[7] 5 -> 4
```

transposition



In igraph,

ID starts from 0 &

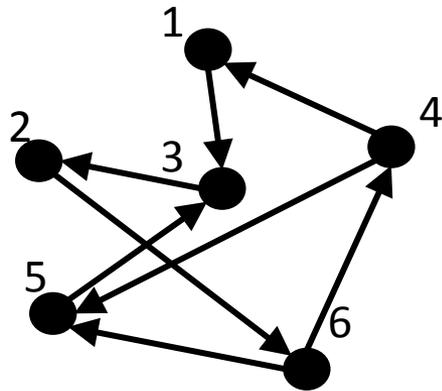
direction is from **i** to **j**

create from edge list

```
> el <-  
  matrix(c(0,2,1,5,2,1,3,0,3,4,4,2,5,  
          3,5,4),nc=2,byrow=TRUE)
```

```
> el
```

```
  [,1] [,2]  
[1,]  0  2  
[2,]  1  5  
[3,]  2  1  
[4,]  3  0  
[5,]  4  2  
[6,]  5  3  
[7,]  5  4
```



```
> gb<-graph.edgelist(el)
```

```
> gb
```

```
Vertices: 6
```

```
Edges: 8
```

```
Directed: TRUE
```

```
Edges:
```

```
[0] 0 -> 2
```

```
[1] 1 -> 5
```

```
[2] 2 -> 1
```

```
[3] 3 -> 0
```

```
[4] 3 -> 4
```

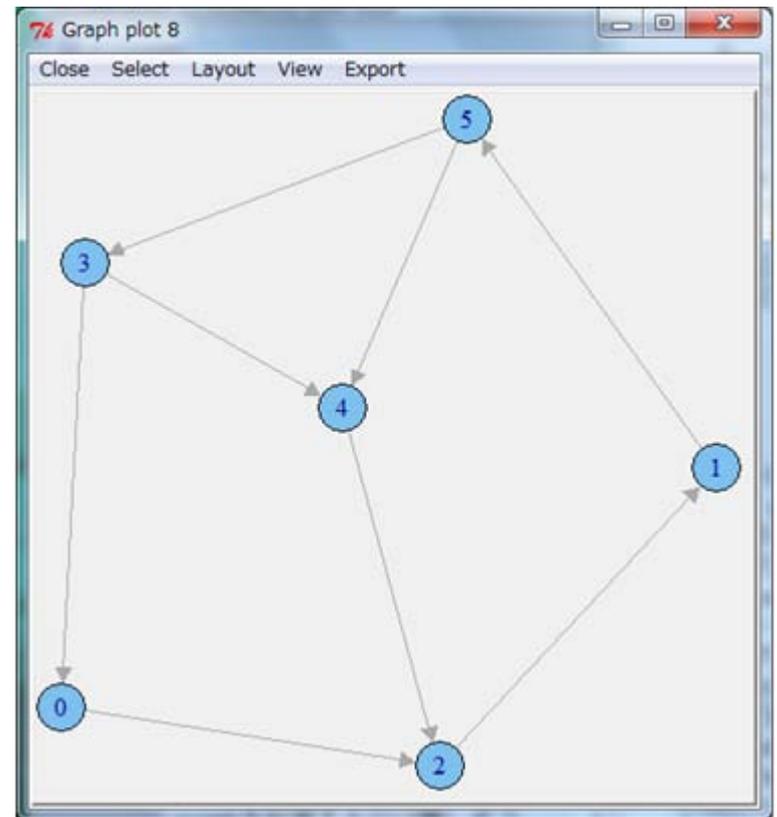
```
[5] 4 -> 2
```

```
[6] 5 -> 3
```

```
[7] 5 -> 4
```

layout the network

- > `tkplot(ga,layout=layout.kamada.kawai)`
 - choose layout (random, circle, Fruchterman-Reingold, Kamada-Kawai)
 - deform graph
 - export (Postscript)

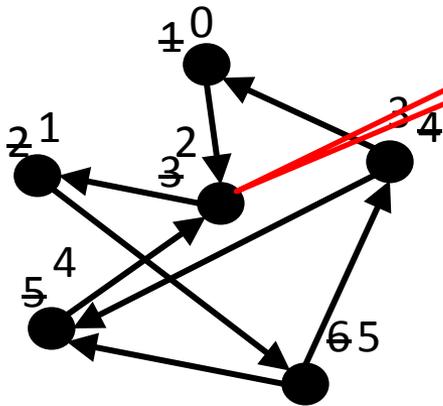


ranking

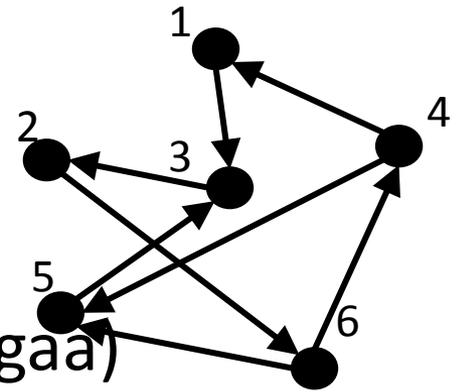
- PageRank: importance of vertices
 - the probability that a random walker will visit

```
> page.rank(ga)$vector
```

```
[1] 0.07337065 0.21643820 0.22522142  
0.11381330 0.16218395 0.20897247
```



metrics (1)



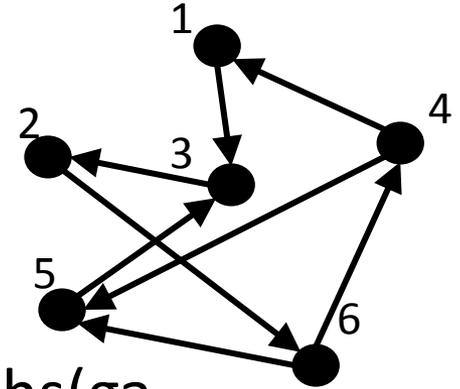
```
> cocitation(gaa)
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	0	0	0	0	1	0
[2,]	0	0	0	0	0	0
[3,]	0	0	0	0	0	0
[4,]	0	0	0	0	1	0
[5,]	1	0	0	1	0	0
[6,]	0	0	0	0	0	0

```
> bibcoupling(gaa)
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	0	0	0	0	1	0
[2,]	0	0	0	0	0	0
[3,]	0	0	0	0	0	0
[4,]	0	0	0	0	0	1
[5,]	1	0	0	0	0	0
[6,]	0	0	0	1	0	0

metrics (2)



- undirected

```
> shortest.paths(ga)
```

```
  [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  0  2  1  1  2  2
[2,]  2  0  1  2  2  1
[3,]  1  1  0  2  1  2
[4,]  1  2  2  0  1  1
[5,]  2  2  1  1  0  1
[6,]  2  1  2  1  1  0
```

- directed

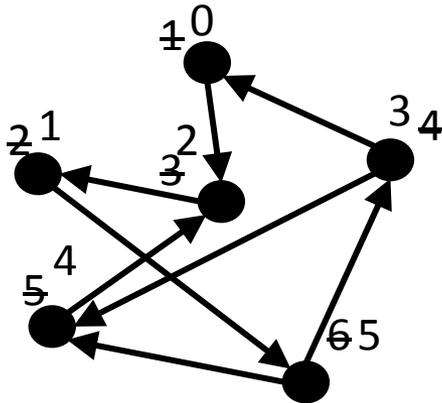
```
> shortest.paths(ga,
  mode="out")
```

```
  [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  0  2  1  4  4  3
[2,]  3  0  3  2  2  1
[3,]  4  1  0  3  3  2
[4,]  1  3  2  0  1  4
[5,]  5  2  1  4  0  3
[6,]  2  3  2  1  1  0
```

metrics (3)

```
> average.path.length(ga)
[1] 2.433333
> average.path.length(ga,directed=FALSE)
[1] 1.466667
```

directed



```
> get.all.shortest.paths(ga,0)
[[1]]
[1] 0
[[2]]
[1] 0 2 1
[[3]]
[1] 0 2
[[4]]
[1] 0 3
[[5]]
[1] 0 3 4
[[6]]
[1] 0 2 4
[[7]]
[1] 0 3 5
```

shortest paths from 0

metrics (4)

```
> is.connected(ga)
```

```
[1] TRUE
```

```
> no.clusters(ga)
```

```
[1] 1
```

```
> clusters(ga)
```

```
$membership
```

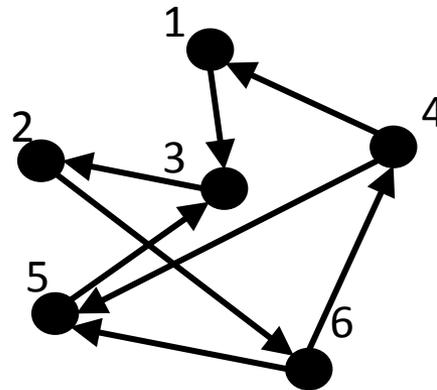
```
[1] 0 0 0 0 0 0
```

```
$size
```

```
[1] 6
```

```
$no
```

```
[1] 1
```



metrics (5)

```
> graph.density(ga)
```

```
[1] 0.2666667
```

$$\rho = \frac{m}{n(n-1)} = \frac{8}{6 \cdot 5}$$

