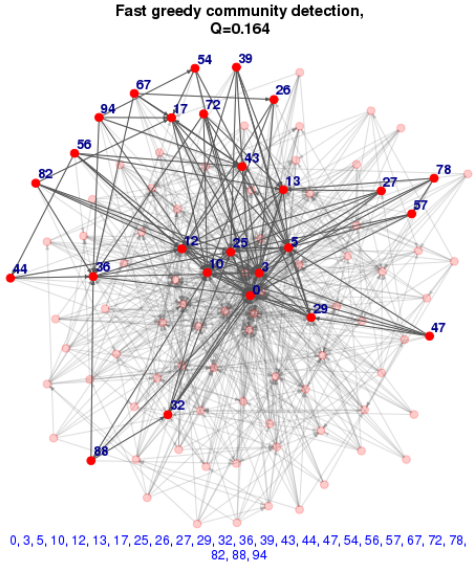
Complex Networks
tools for analyzing networks (igraph)

2018.11.29(Thu)

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.
- The following three are available:
  - igraph R package
  - python-igraph
  - igraph C library

http://igraph.org



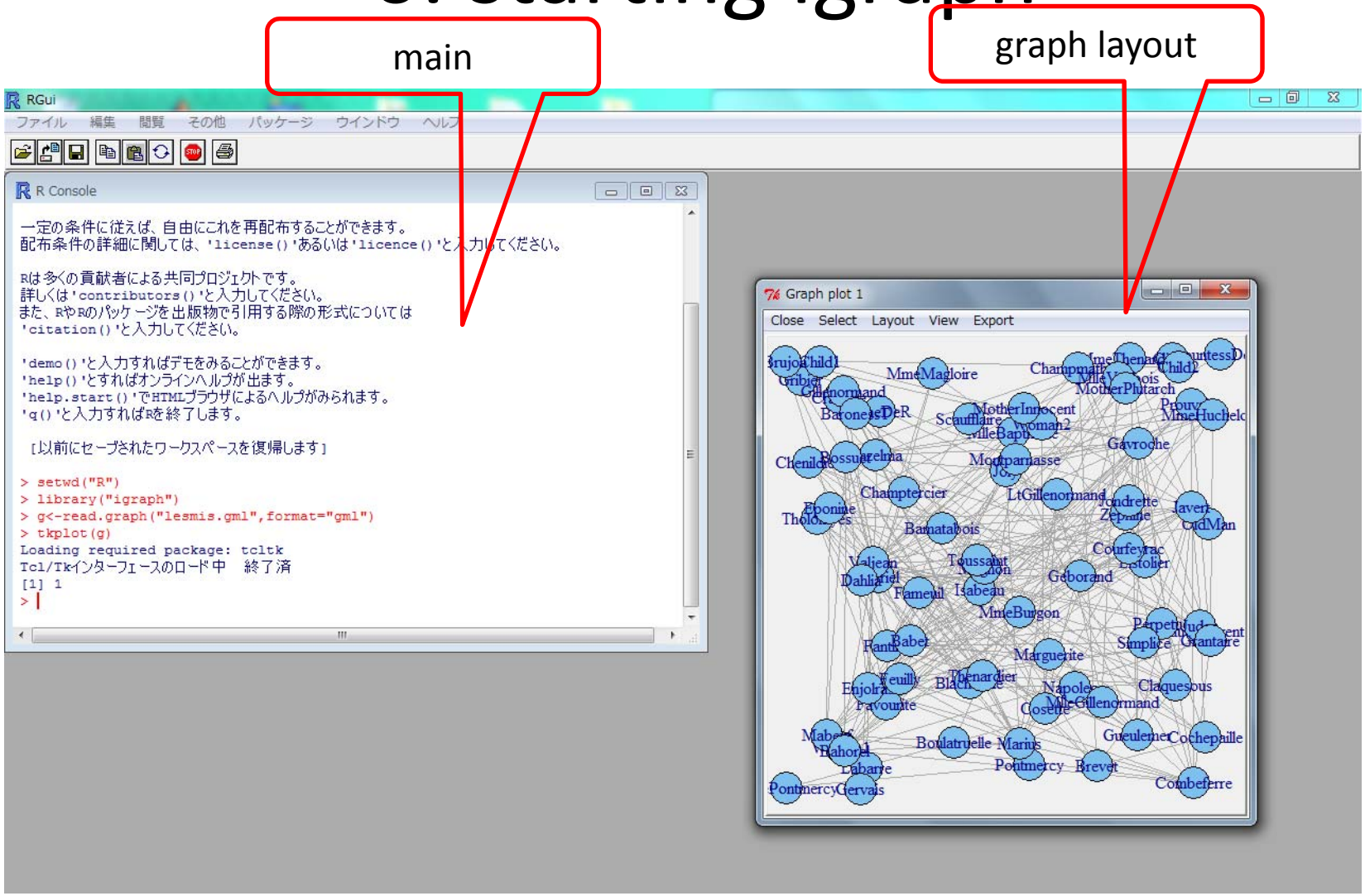
tutorials

- tutorial of python-igraph
  - http://igraph.org/python/doc/tutorial/tutorial.html
- tutorials of R
  - http://cran.r-project.org/other-docs.html (many tutorials in English and other languages)

procedure for analyzing network

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

0. starting igraph



1. create graph object

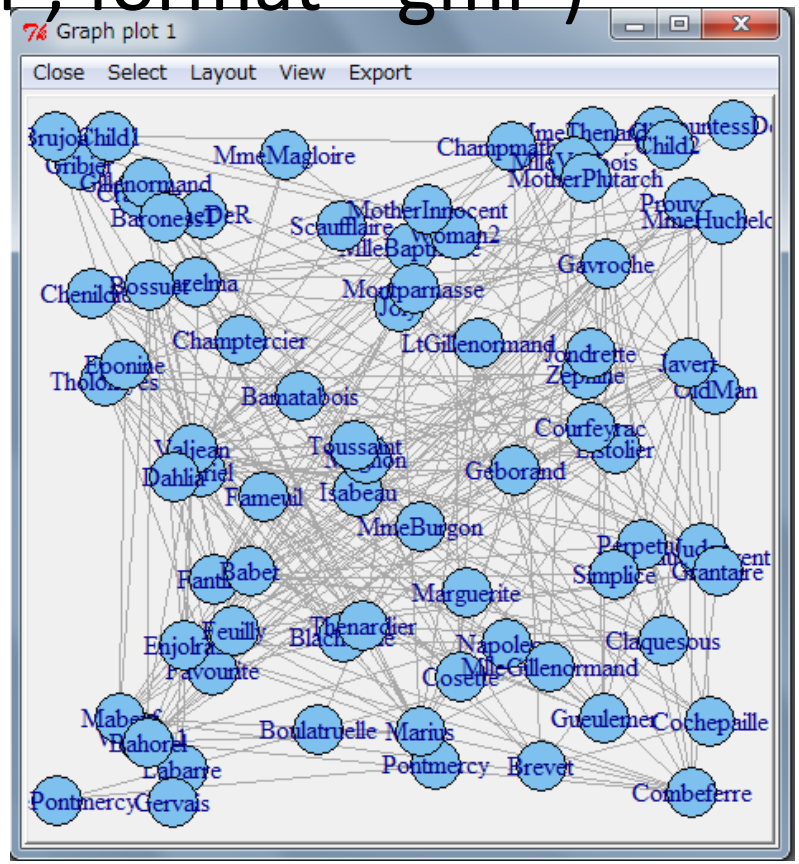
- library(“igraph”)
- setwd(“R”)
- g<-read.graph(“lesmis.gml”, format=“gml”)
- tkplot(g)

start igraph

set directory

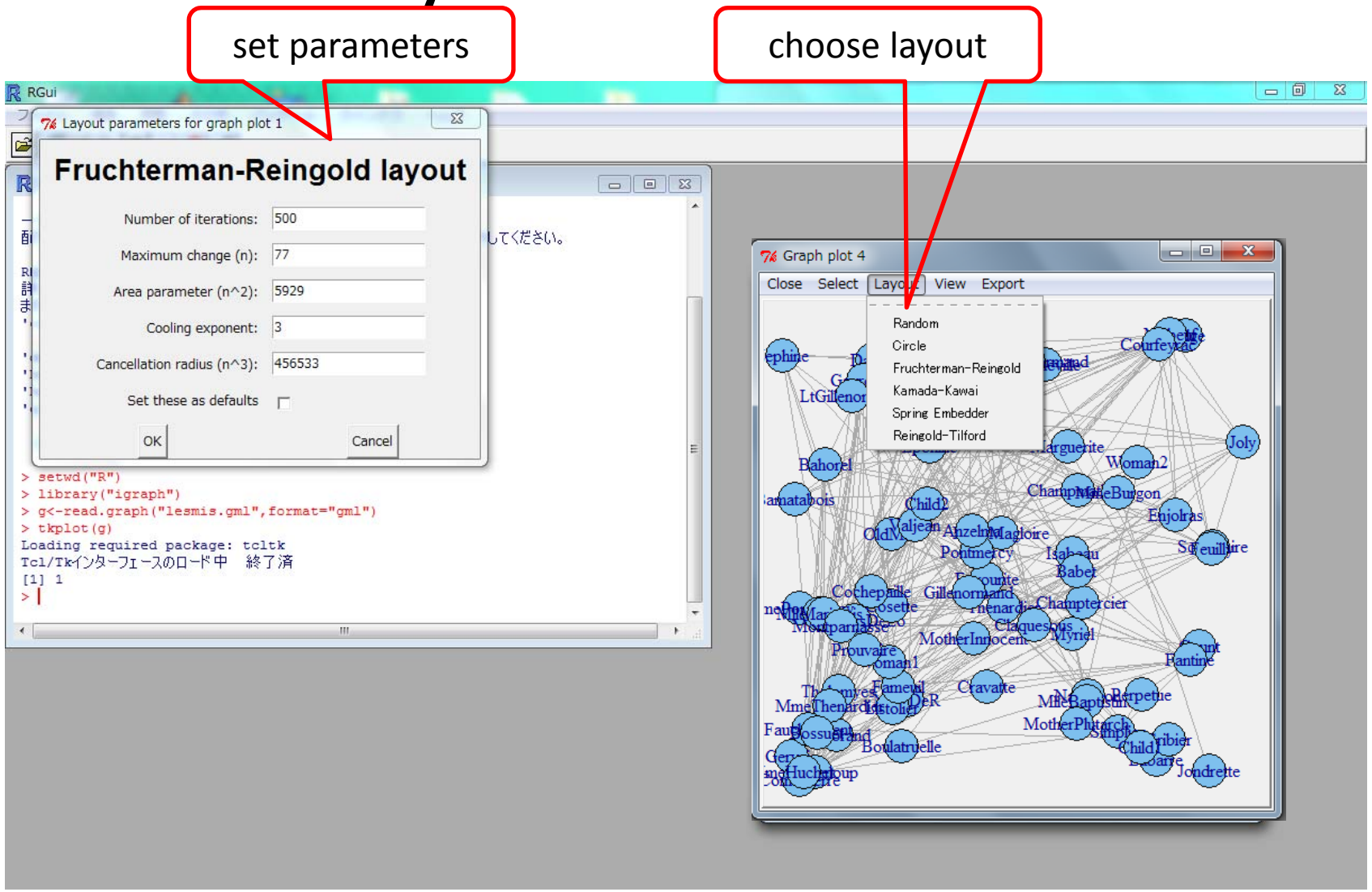
create graph

open graph window

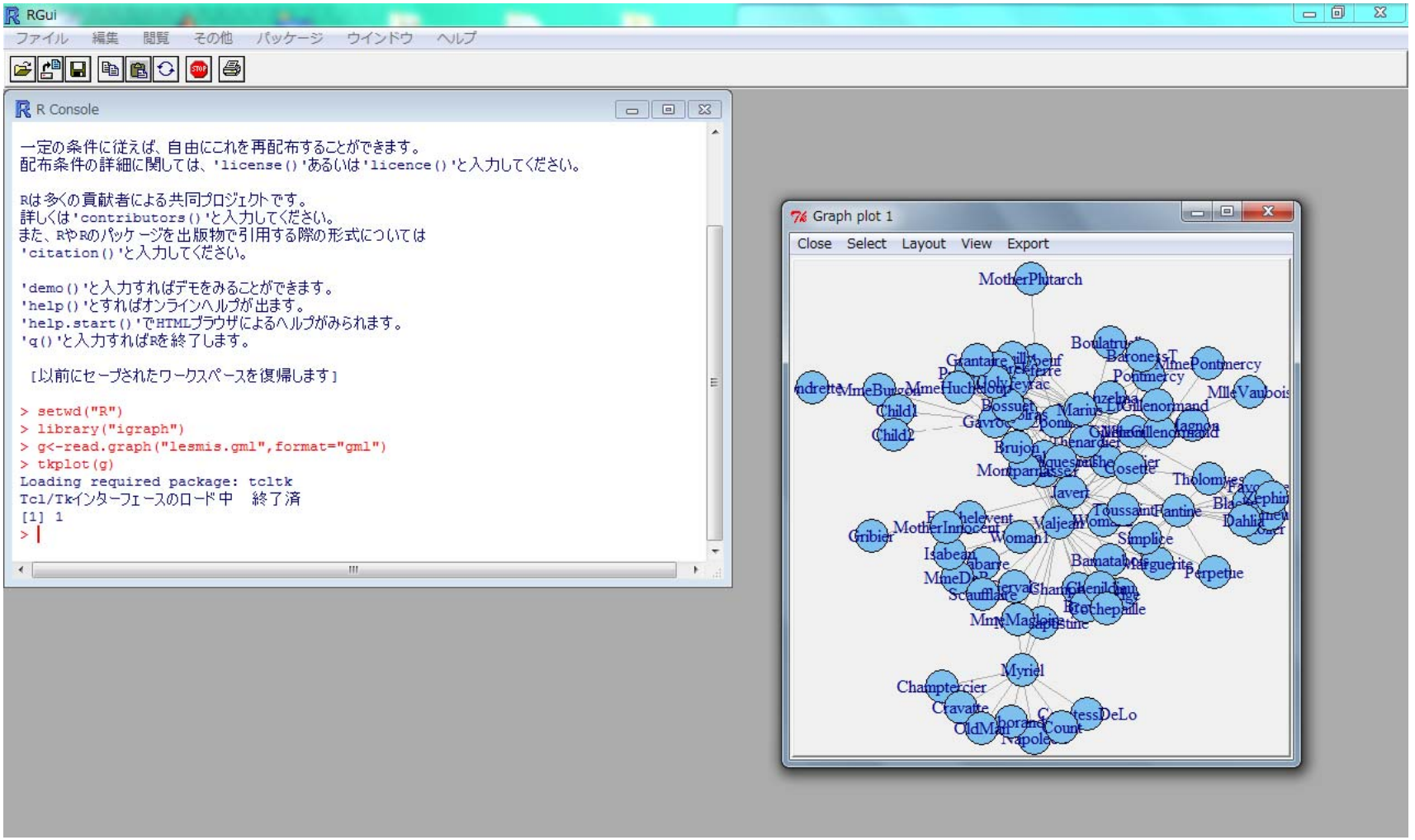


lesmis.gml is available at Mark Newman’s Website

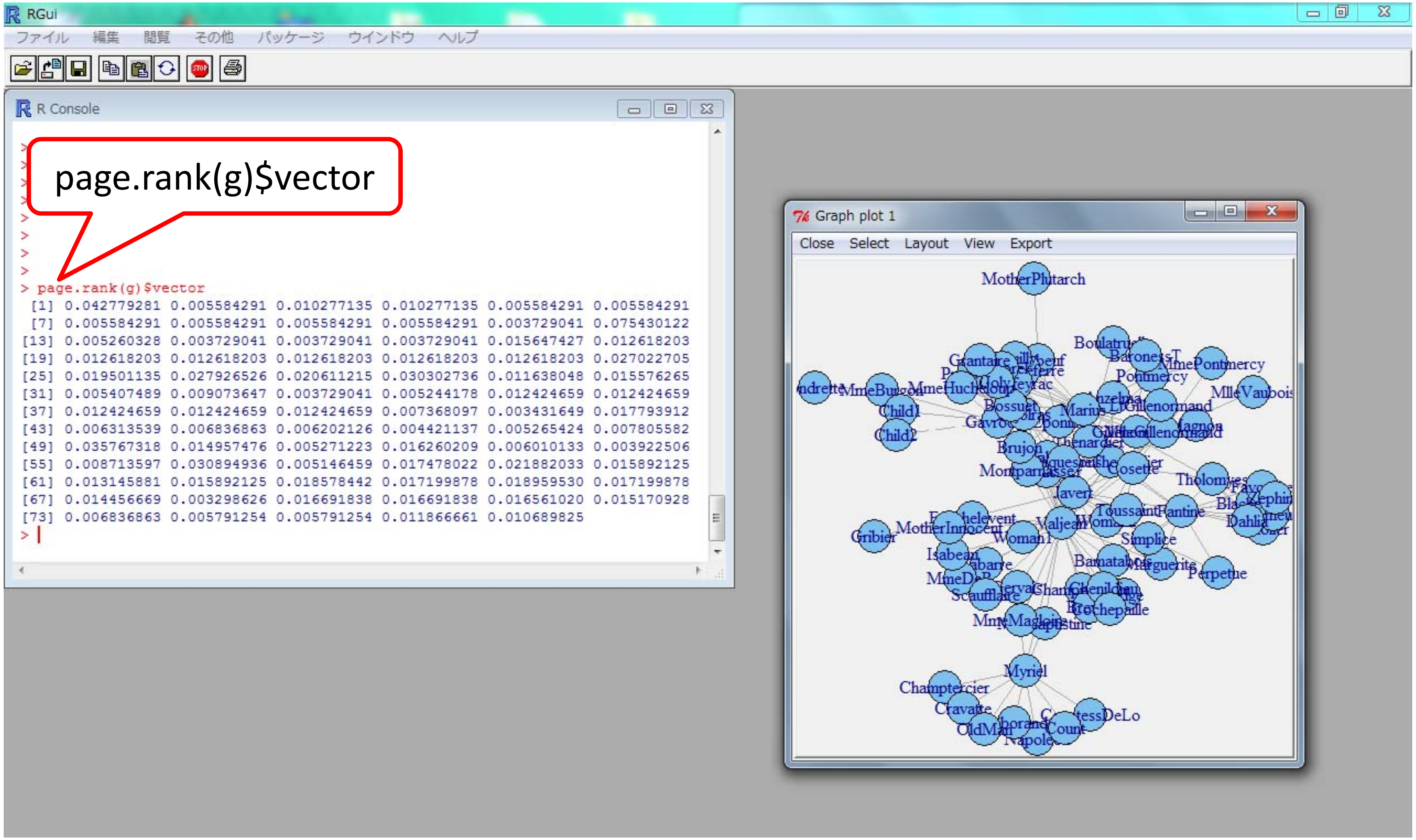
2. layout the network



2. layout the network

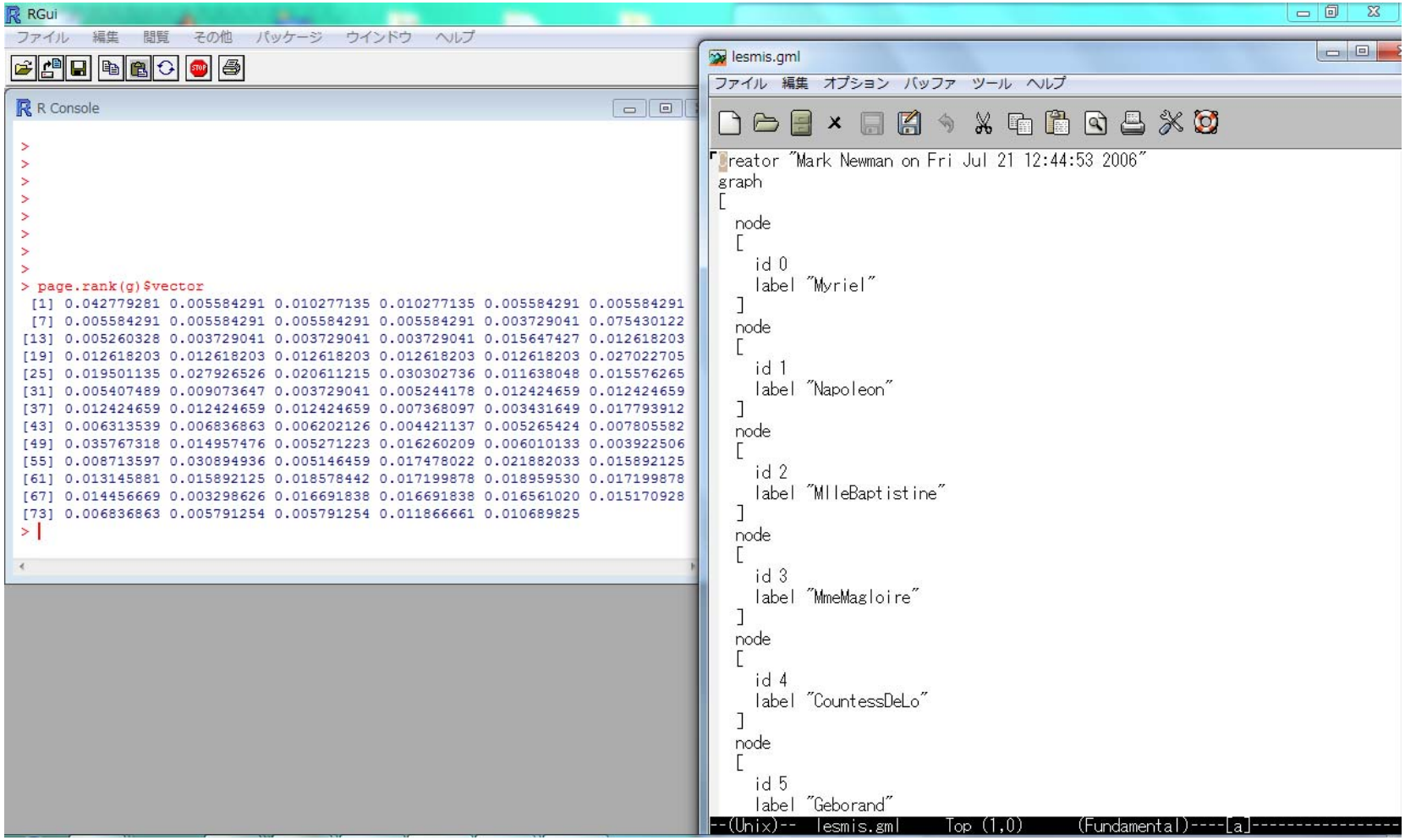


3. ranking


RGui
ファイル 編集 閲覧 その他 パッケージ ウインドウ ヘルプ
R Console
page.rank(g)$vector
> page.rank(g)$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
Graph plot 1
Close Select Layout View Export
MotherPlutarch
Myriel
Gribier
MlleVaubois

3.ranking

- original gml file contain labels



4. metrics

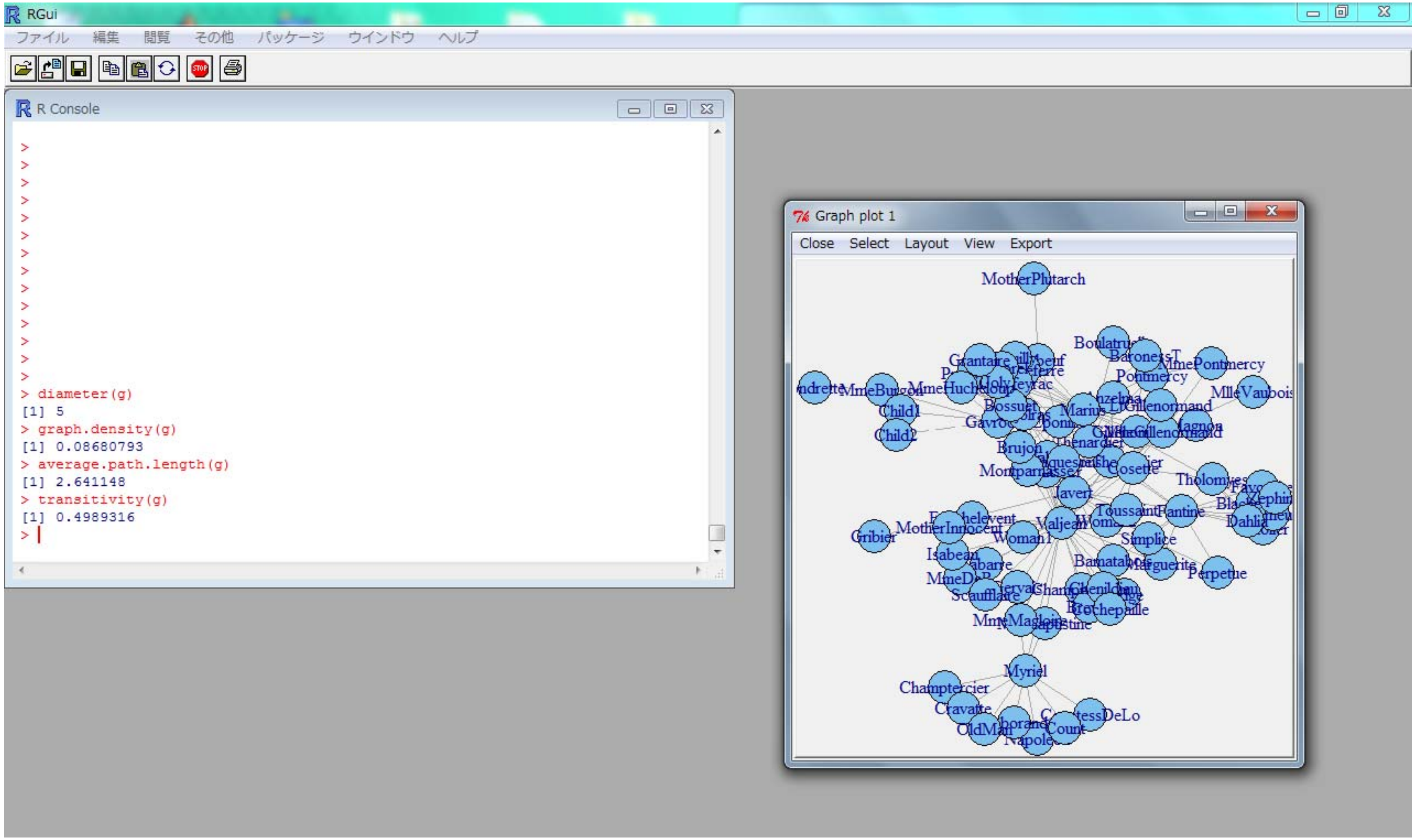
- diameter(g)
- graph.density(g)
- average.path.length(g)
- transitivity(g)

clustering coefficient

percentage that two friends are friends of each other
(percentage of triangle appearing in the network)

- help
  - ??rank
  - help("page.rank")

4. metrics


RGui
ファイル 編集 閲覧 その他 パッケージ ウインドウ ヘルプ
R Console
> diameter(g)
[1] 5
> graph.density(g)
[1] 0.08680793
> average.path.length(g)
[1] 2.641148
> transitivity(g)
[1] 0.4989316
Graph plot 1
Close Select Layout View Export
MotherPlutarch
Myriel
Valjean
Cosette
Javert
Gribier
MlleVaubois
Perpetue
Simplice
Child1
Child2
Brujon

5. community detection

modularity optimization

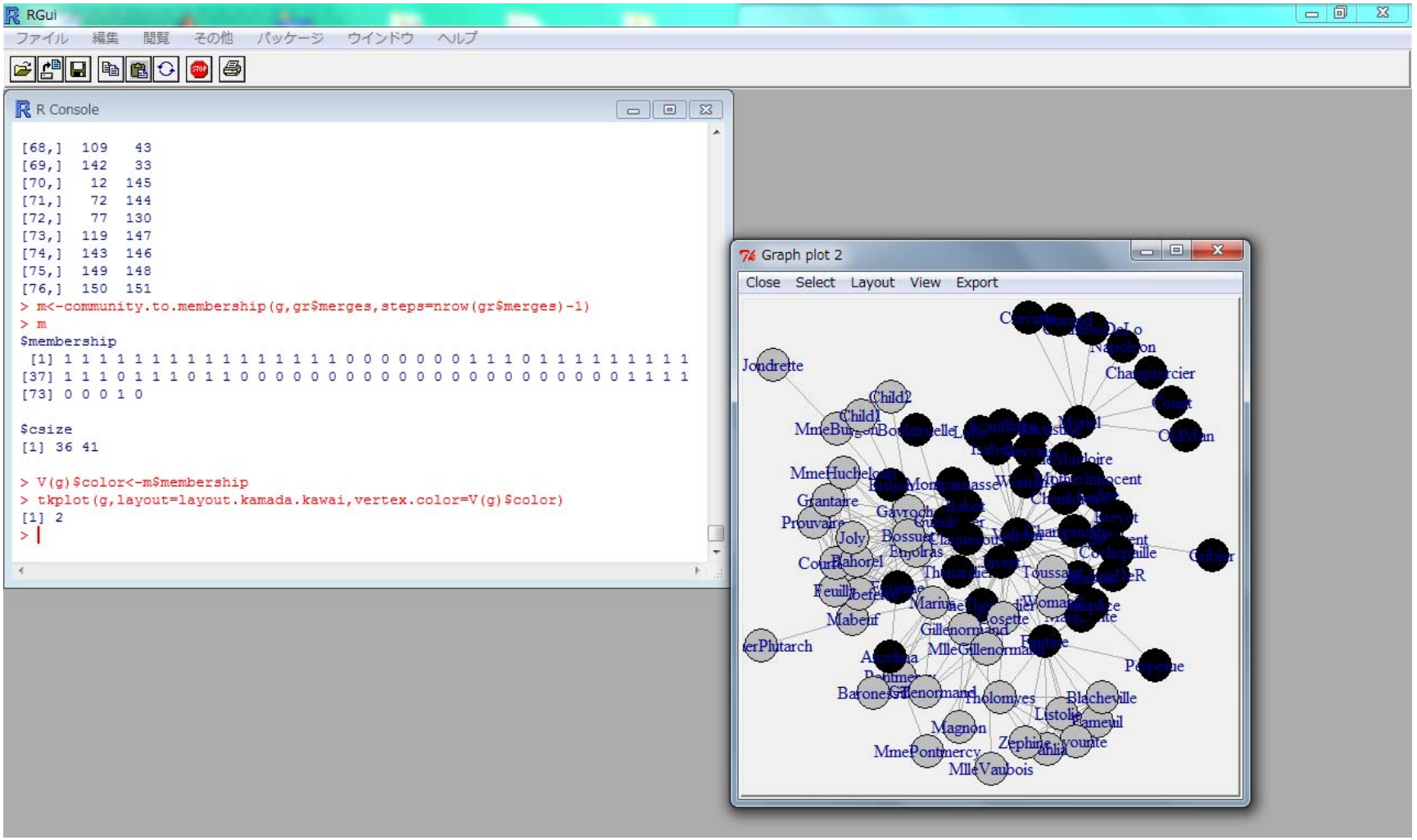
store membership and size

```
> 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
$csize
[1] 17 17
> V(g)$color<-m$membership
> V(g)$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
> tkplot(g,layout=layout.kamada.kawai,vertex.color=V(g)$color)
[1] 2
>
```

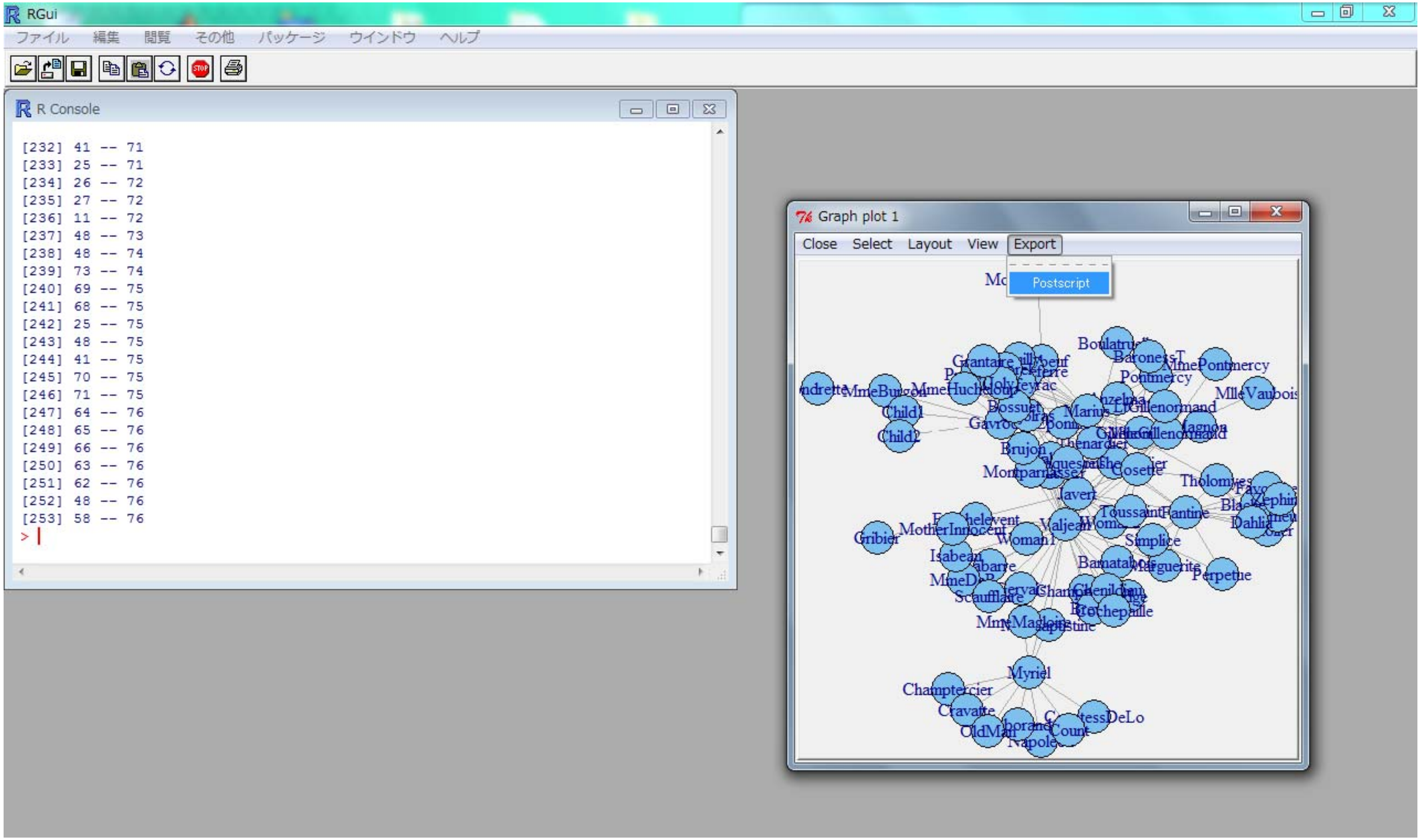
store membership

visualize network

5. community detection



6. export


RGui
ファイル 編集 閲覧 その他 パッケージ ウインドウ ヘルプ
R Console
[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
>
Graph plot 1
Close Select Layout View Export
Postscript
Myriel
Champtercier
Gribier
Perpetue
Simplice
Cosette
Tholomyes
MlleVaubois

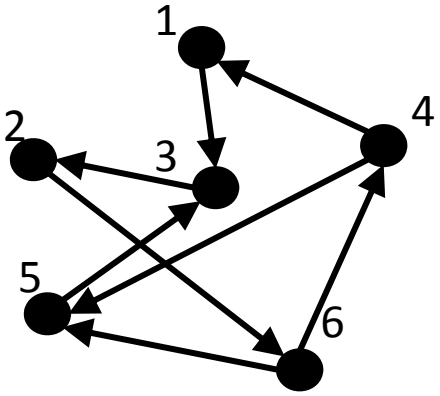
create from adjacency matrix(1)

- 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)

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

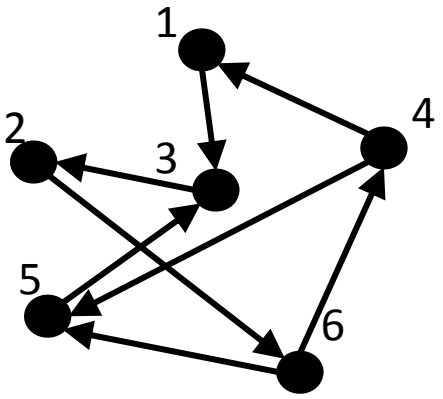
(i: row index, j: column index)



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
```

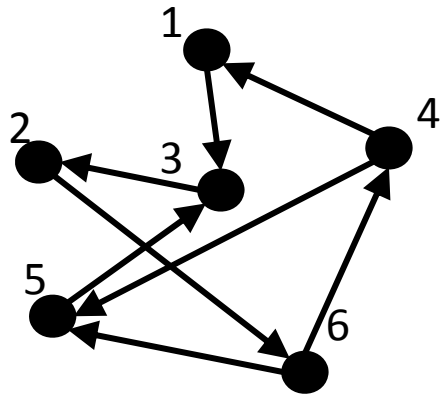
transposition



In igraph,
ID starts from 0 &
direction is from i to j

create from edge list

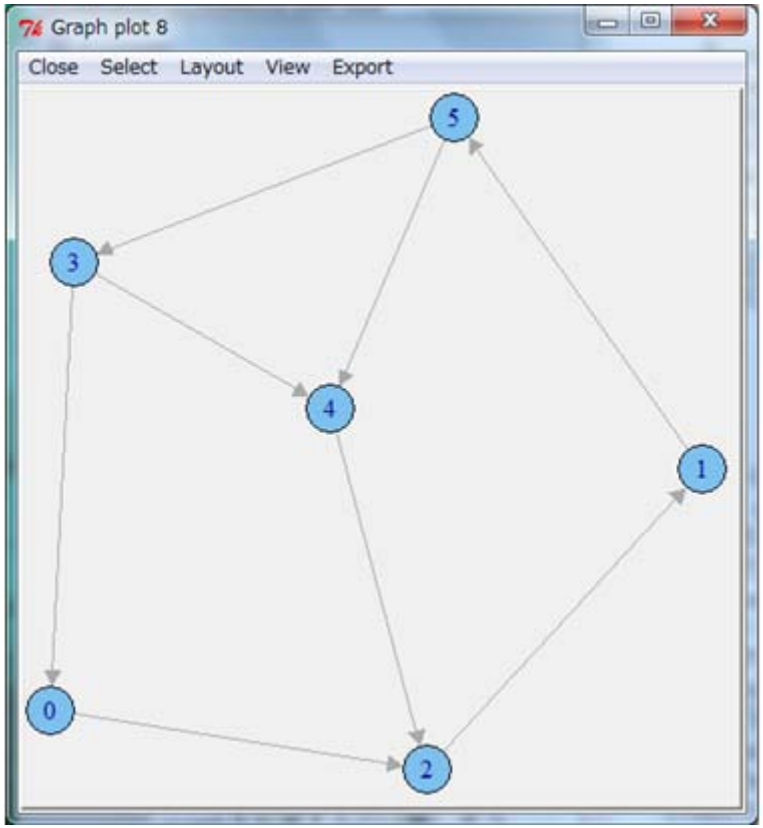
```
> el <-
    matrix(c(0,2,1,5,2,1,3,0,3,4,4,2,5,
    3,5,4),nc=2,byrow=TRUE)
> el
     [,1] [,2]
[1,]    0    2
[2,]    1    5
[3,]    2    1
[4,]    3    0
[5,]    4    2
[6,]    5    3
[7,]    5    4
```




```
> 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)

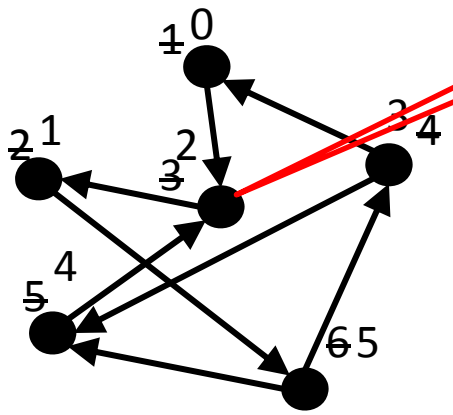


ranking

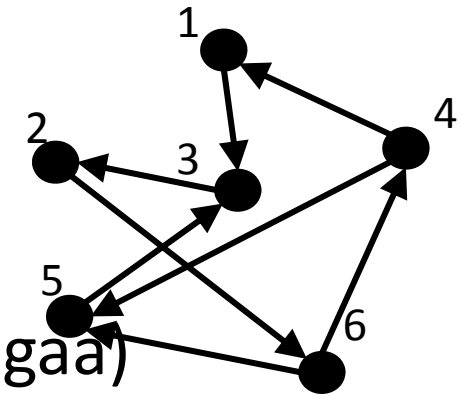
- PageRank: importance of vertices
  - the probability that a random walker will visit

> page.rank(ga)$vector

[1] 0.07337065 0.21643820 0.22522142
0.11381330 0.16218395 0.20897247



metrics (1)



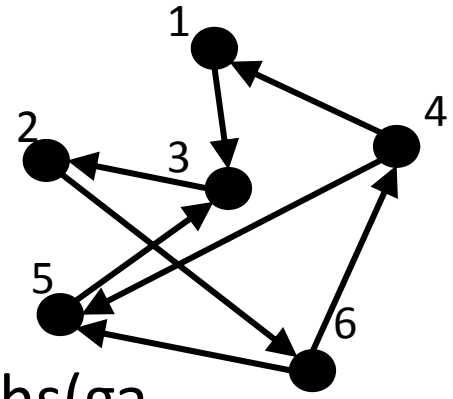
> cocitation(gaa)

	[,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
```

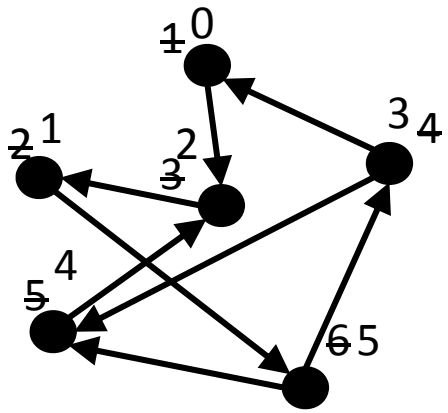
- directed

```
> shortest.paths(ga,
    mode="out")
                  to
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    0    2    1    4    4    3
from[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
```

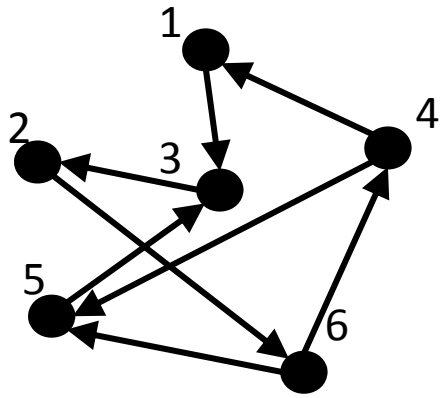
shortest paths from 0

metrics (4)

```
> is.connected(ga)
[1] TRUE

> no.clusters(ga)
[1] 1

> clusters(ga)
$membership
[1] 0 0 0 0 0 0
$csize
[1] 6
$no
[1] 1
```



metrics (5)

> graph.density(ga)

[1] 0.2666667

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