An R Lecture from Practice: Part III

Fangda Fan

2016.5

Contents

Network Analysis

Clustering

Preparation

- Download the dataset "Rlecture_data_facebook.txt"

Our Goals

- Learn network analysis in R with igraph
- Learn some basic concepts and methods to analyze a network
- Learn ERGM, a regression for network structure

Contents

Network Analysis
Introduction to Network
Summarize Network
Network Regression

Clustering

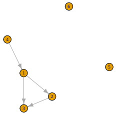
Network Analysis

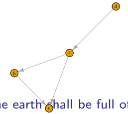
- There are many kinds of networks in the society: classmates, friends, telephones, business transactions,...
  - How they are same with and different from each other?
  - When we want to compare them, we need to represent them in a uniform way.
- Network (graph) representation in mathematics $G = (V, E)$
  - G (Graph): the whole network
  - V (Vertex): the nodes of network (people, companies, ...)
  - E (Edges): the connection between nodes (friendship, transactions, ...)

An R Network Analysis Package: igraph

- Install by install.packages("igraph")

```
library("igraph")
g = graph(edges = c(1,2, 2,3, 1,3, 4,1), n = 6)
plot(g)
g

## IGRAPH D--- 6 4 --
## + edges:
## [1] 1->2 2->3 1->3 4->1

V(g)

## + 6/6 vertices:
## [1] 1 2 3 4 5 6

E(g)

## + 4/4 edges:
## [1] 1->2 2->3 1->3 4->1

g_namev = graph(edges = c("a","b", "b","c", "a","c", "d","a"))
plot(g_namev)
```

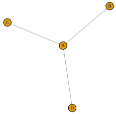
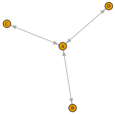




Directed and Undirected k-Stars

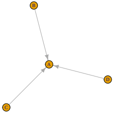
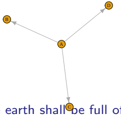
- We can also create simple graphs by graph.formula()

```
g_staru = graph.formula(B:C:D - A) # Undirected 3-Stars
plot(g_staru)
g_starb = graph.formula(B:C:D + A) # Bidirectional 3-Stars
plot(g_starb)
```

- Create one-sided directed graph with “+” on one side of “-”s

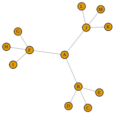
```
g_stari = graph.formula(B:C:D ---+ A) # In-directed 3-Stars
plot(g_stari)
g_staro = graph.formula(B:C:D +- A) # Out-directed 3-Stars
plot(g_staro)
```

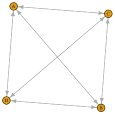



Some Other Basic Components of a Graph

- Tree graph and fully-connected graph

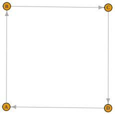
```
g_tree = graph.formula(A - B - C:D:E, A - F - G:H:I, A - J - K:M:L) # Tree Graph
plot(g_tree)
g_full = graph.formula(A:B:C:D + A:B:C:D) # Fully-connected Graph
plot(g_full)
```





- Ring graph and isolated graph

```
g_ring = graph.formula(A -+ B -+ C -+ D -+ A) # Ring Graph
plot(g_ring)
g_empty = graph.formula(A:B:C:D) # Graph of Isolated Points
plot(g_empty)
```



Demo Tutorial of Igraph

- Find and run the demo in igraph

```
demo(package = "igraph")
demo("centrality", package = "igraph")
```

- Use interactive version igraphdemo()

```
igraphdemo("centrality")
```

Contents

Network Analysis
- Introduction to Network
- Summarize Network
- Network Regression

Clustering

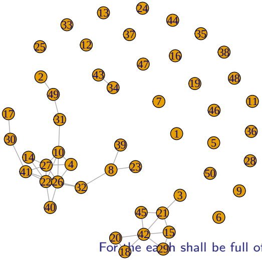
Read Network Data

- Data: A sample of 4000+ Facebook friends (mutual)

```
datag = read.graph("Rlecture_data_facebook.txt", format = "edgelist")
datag = as.undirected(datag)
datag
```

- Take a subgraph from the original graph for analysis

```
datag_sub = induced.subgraph(datag, 1:50)
datag_sub
V(datag_sub)
E(datag_sub)
plot(datag_sub, vertex.size = 10, edge.arrow.size = 0.3)
```



Neighbour and Path

- Neighbours: The set of vertices directly connected to a vertex

```
V(datag)[nei(1)] # The vertices vertex 1 is connected with

## + 2/4039 vertices:
## [1]  59 172
```

- The shortest path to pass from one vertex to another vertex

```
shortest.paths(datag, v = 4, to = 10) # The shortest path from vertex 4 to vertex 10

##      [,1]
## [1,]    1

shortest.paths(datag, v = 1:5, to = 1:10) # The shortest path from vertex 1~5 to vertex 1~10

##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    0    3    7    4    5    4    7    4    7     3
## [2,]    3    0    6    2    4    2    6    2    6     2
## [3,]    7    6    0    5    7    6    4    5    4     5
## [4,]    4    2    5    0    5    2    6    3    5     1
## [5,]    5    4    7    5    0    4    7    3    7     4
```

Adjacency Matrix

- Adjecency Matrix: the existence of edge between any two vertices

```
am = get.adjacency(datag)
am[1:10, 1:10]

## 10 x 10 sparse Matrix of class "dgCMatrix"
##
##  [1,] . . . . . . . . . .
##  [2,] . . . . . . . . . .
##  [3,] . . . . . . . . . .
##  [4,] . . . . . . . . . 1
##  [5,] . . . . . . . . . .
##  [6,] . . . . . . . . . .
##  [7,] . . . . . . . . . .
##  [8,] . . . . . . . . . .
##  [9,] . . . . . . . . . .
## [10,] . . . 1 . . . . . .
```

- Matrix power of adjecency matrix A^p: the total number of p-path between any two vertices

```
am2 = am%*%am
am2[1:10, 1:10]
```

Measurements of Network Centrality (1)

- Degree $C_D(v)$: the number of edges of a vertex

```
vdegree = degree(datag)
head(vdegree) # The degree of first 6 vertices

## [1]  2 16  9 16  9 12

which.max(vdegree) # The vertex name with the highest degree

## [1] 2544

max(vdegree) # The highest degree of the network

## [1] 293
```

- Betweenness $C_B(v) = \sum_{i,j \in V \setminus \{v\}} \frac{\# \arg_v d(i,j)}{\# \arg d(i,j)}$

```
vbetween = betweenness(datag)
head(vbetween)

## [1]    0.00 6293.45    9.72 3878.46 7836.00 4845.79

which.max(vbetween)

## [1] 1086

max(vbetween)

## [1] 1951224
```

Measurements of Network Centrality (2)

- Closeness $C(v) = \sum_{i \in V \setminus \{v\}} \frac{1}{d(i,v)}$

```
vclose = closeness(datag)
head(vclose)

## [1] 2.12e-06 2.11e-06 2.05e-06 2.10e-06 2.08e-06 2.10e-06

which.max(vclose)

## [1] 1535

max(vclose)

## [1] 2.14e-06
```

- Page Rank in Google

```
vpagerank = page.rank(datag)
head(vpagerank$vector)

## [1] 8.73e-05 2.27e-04 2.31e-04 2.23e-04 2.73e-04 2.07e-04

which.max(vpagerank$vector)

## [1] 484

max(vpagerank$vector)

## [1] 0.00136
```

Comparison between Network Centrality

- We compare 4 measurements of network centrality

```
library(data.table)
vcentral = data.table(vertex = V(datag), degree = vdegree, betweenness = vbetween,
           closeness = vclose, pagerank = vpagerank$vector) # data.table of vertex centrality
setkey(vcentral, degree) # Sort by degree centrality
vcentral[,lapply(.SD, frank), .SDcols = -1] # Comparsion by rank

##       degree betweenness closeness pagerank
##    1:   40.5         174      40.5     40.5
##    2:   40.5         174      40.5     40.5
##    3:   40.5         174      40.5     40.5
##    4:   40.5         174      40.5     40.5
##    5:   40.5         174      40.5     40.5
##   ---
## 4035: 4035.0        3860    4025.0   4000.0
## 4036: 4036.0        3608    3980.0   4015.0
## 4037: 4037.0        3605    3961.0   4017.0
## 4038: 4038.0        3992    3419.0   3980.0
## 4039: 4039.0        4023    3844.0   4002.0

cor(vcentral[, -1, with = FALSE], method = "spearman") # Correlation matrix of rank (Spearman)

##             degree betweenness closeness pagerank
## degree       1.000       0.511     0.612    0.778
## betweenness  0.511       1.000     0.500    0.635
## closeness    0.612       0.500     1.000    0.311
## pagerank     0.778       0.635     0.311    1.000
```

Contents

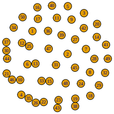
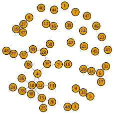
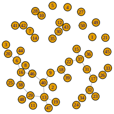
Network Analysis
Introduction to Network
Summarize Network
Network Regression

Clustering

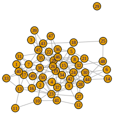
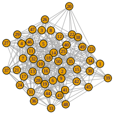
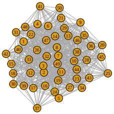
Random Graph

- Random graph: edges randomly generated with probability p

```
set.seed(1)
plot(random.graph.game(50, p = 0.01))
plot(random.graph.game(50, p = 0.03))
plot(random.graph.game(50, p = 0.05))
```

```
plot(random.graph.game(50, p = 0.1))
plot(random.graph.game(50, p = 0.3))
plot(random.graph.game(50, p = 0.5))
```

Exponential Random Graph Model (ERGM)

- Generate a new random graph based on shapes of network
- Assumption: the structure of a network y can be represent as a exponential random graph with the number of shapes $s(y) = (s_1(y), ..., s_k(y))$ as statistics and $\beta = (\beta_1, ..., \beta_k)$ as coefficient
- Let $\beta = \hat{\beta}$ maximize $P(Y = y|\beta) = \exp\left\{\frac{1}{c}\sum_{j=1}^{k} \beta_j s_j(y) - c_1(\beta) - c_2(y)\right\}$
- Linear regression: Let $\beta = \hat{\beta}$ maximize $f(Y = y|\beta, X) = \exp\left\{\frac{1}{\sigma^2}\sum_{j=1}^{p} \beta_j (x_j^T y) - c_1(\beta, X) - c_2(y)\right\}$

ERGM and Network Package

- Install by install.packages("ergm")

```
library(ergm)
data_edgelist = get.edgelist(datag)
datan = network(data_edgelist, directed = FALSE)
class(datag)

## [1] "igraph"

class(datan)

## [1] "network"
```

- We calculate the probability coefficient of edges for network

```
model_ergm1 = ergm(datan ~ edges, estimate = "MPLE")

## Evaluating log-likelihood at the estimate.

model_ergm1

##
## MPLE Coefficients:
## edges
## -4.56
```

Summary of ERGM

- All coefficients in ERGM represents an exponential magnitude in probability

```
summary(model_ergm1)

##
## ==========================
## Summary of model fit
## ==========================
##
## Formula:   datan ~ edges
##
## Iterations:  NA
##
## Maximum Likelihood Results:
##       Estimate Std. Error MCMC % p-value
## edges -4.56226    0.00346      0  <1e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## For this model, the pseudolikelihood is the same as the likelihood.
##
##      Null Deviance: 11304871  on 8154741  degrees of freedom
##  Residual Deviance:   938040  on 8154740  degrees of freedom
##
## AIC: 938042    BIC: 938056    (Smaller is better.)
```

Terms of ERGM (1)

To test the effect of a complex term, we must add all simpler terms to the model

- k-stars represent the concentration of relationships

```
model_ergm2 = ergm(datan ~ edges + kstar(2:3), estimate = "MPLE")

## Evaluating log-likelihood at the estimate.

summary(model_ergm2)

##
## ==========================
## Summary of model fit
## ==========================
##
## Formula:   datan ~ edges + kstar(2:3)
##
## Iterations:  NA
##
## Maximum Pseudolikelihood Results:
##          Estimate Std. Error MCMC % p-value
## edges  -7.53e+00   1.22e-02      0  <1e-04 ***
## kstar2  3.33e-02   1.47e-04      0  <1e-04 ***
## kstar3 -1.82e-04   1.40e-06      0  <1e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Warning:  The standard errors are based on naive pseudolikelihood and are suspect.
##
##      Null Pseudo-deviance: 11304871  on 8154741  degrees of freedom
##  Residual Pseudo-deviance:   779779  on 8154738  degrees of freedom
```

Terms of ERGM (2)

- Triangles represent a smallest clique in the network

```
model_ergm3 = ergm(datan ~ edges + kstar(2:3) + triangle, estimate = "MPLE")

## Evaluating log-likelihood at the estimate.

summary(model_ergm3)

##
## ==========================
## Summary of model fit
## ==========================
##
## Formula:   datan ~ edges + kstar(2:3) + triangle
##
## Iterations:  NA
##
## Maximum Pseudolikelihood Results:
##           Estimate Std. Error MCMC % p-value
## edges    -5.79e+00   1.20e-02      0  <1e-04 ***
## kstar2    1.54e-02   2.19e-04      0  <1e-04 ***
## kstar3   -4.30e-04   2.71e-06      0  <1e-04 ***
## triangle  1.62e-01   5.04e-04      0  <1e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Warning:  The standard errors are based on naive pseudolikelihood and are suspect.
##
##      Null Pseudo-deviance: 11304871  on 8154741  degrees of freedom
##  Residual Pseudo-deviance:   437289  on 8154737  degrees of freedom
##
## AIC: 437297    BIC: 437353    (Smaller is better.)
```

Terms of ERGM (3): More Terms

We can also add more terms to ERGM

- concurrent: vertices with degree 2 or more

```
model_ergm4 = ergm(datan ~ edges + concurrent, estimate = "MPLE")
summary(model_ergm4)
```

- threetrail: a path of 3 connected edges

```
model_ergm5 = ergm(datan ~ edges + kstar(2) + threetrail, estimate = "MPLE")
summary(model_ergm5)
```

- cycle(k): a ring of k edges. In an undirected graph, cycle(3) are triangles, and cycle(4) are squares

```
model_ergm6 = ergm(datan ~ edges + kstar(2:3) + threetrail + cycle(3:4), estimate = "MPLE")
summary(model_ergm5)
```

Contents

Network Analysis

Clustering