R を利用した社会ネットワーク分析

ネットワークの諸指標

1　はじめに

前章では隣接行列からネットワークを描画する方法について学びました。ネットワークの視覚化は見ていて楽しいですし，構造を直感的に把握できるので分析的観点からも有用ではあります。しかし，目視しているだけではそれ以上の情報は得られず，次にどうすべきかの判断ができません。より考察を進めるためには，ネットワークを特徴付ける何らかの指標を得る必要があるのです。

むしろ社会ネットワーク分析（SNA）では，描画することそのものよりも，本章以降で述べるようなネットワークの諸指標を計算し，その構造を比較・検討することが目的となります。そこで本章では SNA の諸指標の中でも，最も基本的なものについて取り上げ，『SNA で分析をする』ということはどういうことなのかを説明します。なお，本章では以下の R のパッケージを使用しますので，最初にインストールしておいて下さい。

- sna

インストール終了後，以下のコマンドを実行します。

```
library(sna)
```

これで本章で説明する分析のための準備が整いました。が，まずは本章で必要になると思われる，SNA を行うに当たっての基礎用語についてまとめておきます。

2　SNA の基礎用語

距離 (distance)：　2 つのノードをつなぐ辺の数の最小値を（最短）距離という。

完全グラフ (complete graph)：　すべてのノードから，すべての他のノードに辺が存在するグラフ。

密度 (density)：　手元のグラフ G の辺の数 m と完全グラフの辺の数の比率を，グラフ G の密度 $d(G)$ という。頂点の数を n とすると，無向グラフの場合は $G(d) = \frac{2m}{n(n-1)}$ で，有向グラフの場合は $G(d) = \frac{m}{n(n-1)}$ で定義される。

相互性（mutuality）：　有向グラフ全体において，相互に有向辺を持つ二者関係がどれくらいの割合を占めているかの指標。

構造同値（structural equivalence）：　ノード i とノード j を入れ替えても，ノードの名前以外ではネットワークに変化がない状態。

3　ネットワーク構造の諸指標

さて，前章で作図に関して一通りの説明が終わりました。今度はネットワークを特徴づける諸指標について学びましょう。ここでは『距離』『密度』『相互性』『構造同値』の概念について説明します。

説明のために図 1 と図 2 のようなネットワークを用います。まずはじめに隣接行列を R に読み込んでおきましょう。

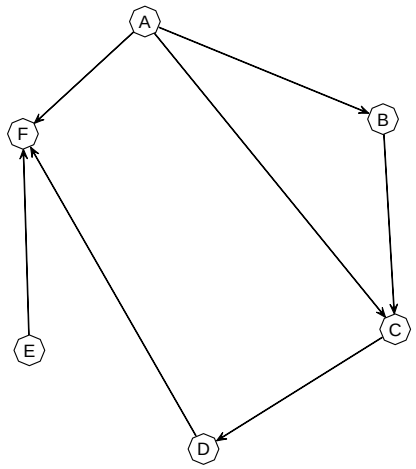

図 1　説明用の有向グラフ 1

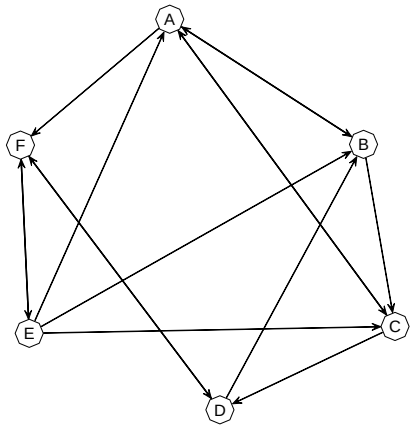

図 2　説明用の有向グラフ 2

```
# データの読み込み
dat1 <- matrix(c(0,0,0,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,
                 1,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0), 6, 6)
dat2 <- matrix(c(0,1,1,0,1,0,1,0,0,1,1,0,1,1,0,0,1,0,0,0,
                 1,0,0,1,0,0,0,0,0,1,1,0,0,1,1,0), 6, 6)
# 行名の入力
rownames(dat1) <- c("A", "B", "C", "D", "E", "F")
rownames(dat2) <- c("A", "B", "C", "D", "E", "F")
# 列名の入力
colnames(dat1) <- c("A", "B", "C", "D", "E", "F")
colnames(dat2) <- c("A", "B", "C", "D", "E", "F")
# ネットワークの描画
gplot(dat1, mode="circle", displaylabels=T,vertex.cex=3,
      vertex.col="white", label.pos=5)
gplot(dat2, mode="circle", displaylabels=T,vertex.cex=3,
      vertex.col="white", label.pos=5)
```

3.1 距離

2 つのノードをつなぐ辺の数（道の長さ）の最小値を『最短距離（shortest distance)』, あるいは単純に『距離（distance)』と言います。また，そのときの経路のことを『最短経路（shortest path)』とか『測地線（geodesic)』と言います。

geodist() という関数はノード間の測地線の数，また各々の測地線の長さ（2 つのノード間の距離）を計算してくれます。ノード間に測地線が存在しない場合，当該ノード間の距離には'Inf' が与えられます。以下に関数の詳細を示しておきます。

```
geodist(dat)
```

【引数】

dat：　分析対象となるグラフの隣接行列。1 つ以上でもよい。

この関数を使って図 1 の距離を求めると以下のようになります。

```
> geodist(dat1)
$counts
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    1    1    1    0    1
[2,]    0    1    1    1    0    1
[3,]    0    0    1    1    0    1
[4,]    0    0    0    1    0    1
[5,]    0    0    0    0    1    1
[6,]    0    0    0    0    0    1
$gdist
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    0    1    1    2  Inf    1
[2,]  Inf    0    1    2  Inf    3
[3,]  Inf  Inf    0    1  Inf    2
[4,]  Inf  Inf  Inf    0  Inf    1
[5,]  Inf  Inf  Inf  Inf    0    1
[6,]  Inf  Inf  Inf  Inf  Inf    0
```

『1 〜6』はそれぞれ『A〜F』を表しています。$counts には測地線の数が，$gdist には最短距離が示されています。$counts において値が 0 ということは，そのノードへの経路が存在しないということを意味しています。ここでは [,5]，すなわちノード E への要素が全て 0 となっていますね。これはすべてのノードからノード E へは行けないと言う

ことです。また，$gdist における 2 行 4 列目を見てみましょう。ここの値が『2』ということはノード B からノード D への最短距離が 2 ということを意味しています。図 1 をみて，値が正しいことを確認してみましょう。

3.2　密度

密度（density）とは，文字通りグラフの『込み具合』を把握することができる指標です。まず始めに金光（2003）に基づき，グラフの密度の定義を行います。

まず n 個のノードからなる（全てのノード間に辺が存在する）完全グラフを K_n とします。グラフ G の密度を $d(G)$ とし，さらにグラフ G の辺の数を $m(G)$ とすると，$d(G)$ は以下の式で表現されます。

$$d(G) = \frac{m(G)}{m(K_n)} \tag{1}$$

つまり密度は，グラフ G の辺の数と，完全グラフのときの辺の数との比率となります。このことをより詳しく書き下してみます。ノード数が n 個の有向グラフの場合，可能な有向辺の数は最大で $n(n-1)$ 本となりますから，グラフ G に含まれる有向辺の数を m 本とすると，密度 $G(d)$ は

$$G(d) = \frac{m}{n(n-1)} \tag{2}$$

となります。一方，無向グラフの場合では，可能な無向辺の数は最大で $\frac{n(n-1)}{2}$ 本となりますから，密度 $G(d)$ は

$$G(d) = \frac{m}{\frac{n(n-1)}{2}} = \frac{2m}{n(n-1)} \tag{3}$$

となります。

R を利用して密度を計算する場合，gdent という関数を利用します。以下に関数の詳細を記します。

```
gden(dat, g=NULL, diag=FALSE, mode="digraph")
```

【引数】

dat：　分析対象となるグラフの隣接行列。1 つ以上でもよい。

g：　密度を計算するグラフを指定する整数（あるいはベクトル）。'g=NULL' のとき（初期値），密度は全てのグラフで計算される。

diag：　行列の対角要素が意味を持つか否かを決定する論理値。ノードがループ構造を持っているならば，そのときのみ"TRUE"を選択する。デフォルトでは"FALSE"になっている。

mode：　グラフのタイプを識別する引数。"digraph"ではグラフに有向辺が含まれ，"graph"では無向辺のみが含まれることをそれぞれ示す。初期値は"digraph"。

図 1 と図 2 の 2 つのネットワークで密度を計算すると以下のようになります。

```
> gden(dat1, mode="digraph")
[1] 0.2333333
> gden(dat2, mode="digraph")
[1] 0.5
```

結果を見るとわかるように，図 1 よりも図 2 の方が密度が高いです。これはネットワークを目視で見ても納得のいく結果だと思います。このような単純なネットワークであれば，わざわざ密度計算をせずともよいのですが，ノード数が多い場合には目視による判断はより難しいものになります。ですので，密度のような単純な指標でも数値による計算は大切なのです。

3.3　相互性

2 つのノードをそれぞれ i，j とすると，有向辺の張り方には以下の 3 種類が存在します。

$$i \longleftrightarrow j \tag{a}$$

$$i \longrightarrow j \text{ あるいは } i \longleftarrow j \tag{b}$$

$$i \quad\quad j \tag{c}$$

相互性（reciprocity, mutuality）とは，あるグラフにおいて，全ての有向辺のうちどれくらい双方向（$i \longleftrightarrow j$）の関係性を含んでいるかの割合のことです。R で有向辺の種類と数を知りたければ `dyad.census()` という関数を，相互性スコアを求めるためには `grecip()` という関数を使用します。それぞれの関数の詳細は以下になります。

```
dyad.census(dat, g=NULL)
```

【引数】

dat：　分析対象となるグラフの隣接行列。1 つ以上でもよい。
g：　計算するグラフを指定する整数（あるいはベクトル）。

```
grecip(dat, g = NULL,
       measure = c("dyadic", "dyadic.nonnull", "edgewise", "edgewise.lrr"))
```

【引数】

dat：　分析対象となるグラフの隣接行列。1 つ以上でもよい。

g：　計算するグラフを指定する整数（あるいはベクトル）。

measure：　”dyadic”（初期値），”dyadic.nonnull”，”edgewise”あるいは”edgewise.lrr”のうちから 1 つ。

相互性の指標としては以下の 3 つがあり，引数 measure で指定します。ここで a, b, c は，上式における 2 つのノード間の関係性の種類を表しています。

1. dyadic $= \frac{(a+c)}{(a+b+c)}$
2. dyadic.nonnull $= \frac{a}{(a+b)}$
3. edgewise $= \frac{2\times a}{(2\times a+b)}$

また’edgewise.lrr’ は密度に対する edgewise 相互性の比率の対数を計算してくれます。用途にもよりますが，基本的には dyadic.nonnull を選択すると良いと思います。

実際に図 2 の相互性を調べてみると以下のようになります。

```
> dyad.census(dat2)
     Mut Asym Null
[1,]   4    7    4
> grecip(dat2, measure="dyadic.nonnull")
      Mut
0.3636364
```

このとき『Mnt』は有向辺の双方向関係（a）を，『Asym』は片方向関係（b）を，『Null』は関係がない場合（c）を表しています。

3.4　構造同値

あるグラフにおいて，特定の 2 つのノード i と j を入れ替えても，ノードの名前以外はネットワーク構造に変化がない場合，その 2 つのノードは構造同値（Structural equivalence）であると言います。まずは図 3 を見て下さい。この図の場合，ノード i と

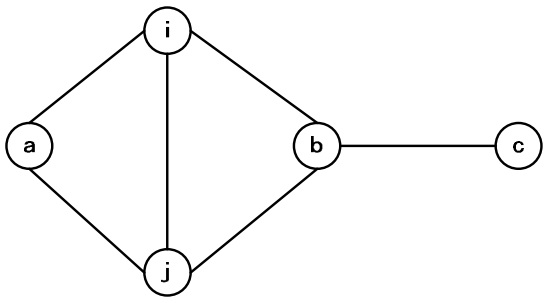

図 3　構造同値の例

ノード j を入れ替えても，ネットワークの構造自体に全く変化がないことが分かるでしょう。したがって，ノード i とノード j は構造同値の関係にあります。しかし，ノード a とノード b は構造同値ではありません。なぜならノード a とノード b を入れ替えた場合，ノード c はノード a と接続されることになり，ネットワークの構造が変わってしまうからです。

実際の社会的な場面で考えた場合，この構造同値の関係は，そのネットワークにおいての同等の役割を演じているもの同士であると考えられます。したがって，この構造同値の関係にあるものを見つけ出し，1 つにまとめることができれば有用です。なぜなら複雑なネットワーク構造をより単純な構造に写像できるので，考察がしやすくなると考えられるからです。これを構造同値性に基づくネットワーク構造の縮約あるいはブロックモデリング（blockmodeling）と言います。R でブロックモデリングを行いたい場合，`blockmodel()` という関数を使用します。この関数の詳細は以下になります。

```
blockmodel(dat, ec, k=NULL, h=NULL, block.content="density",
           plabels=NULL, glabels=NULL, rlabels=NULL,
           mode="digraph", diag=FALSE)
```

【引数】

dat：　1 つあるいは複数の隣接行列。

ec：　`equiv.clust` あるいは `hclust` のオブジェクトクラス，あるいは集団員を示すベクトルの形式としてあたえられる等質なクラスの情報。

k：　形成するクラスの数 (`cutree` を利用した場合)。

h：　クラスを分割するときのノードの数 (`cutree` を利用した場合)。

block.content：　ブロックの内容を示す文字列。

plabels：　各ノードに適用されるラベルを含んだベクトル。

glabels：　モデル化されたグラフに適用されるラベルを含んだベクトル。

rlabels：　(縮約された) ロールに適用されるラベルを含んだベクトル。

mode：　無向グラフ（`graph`）か有向グラフ（`digraph`）かを指定する文字列。

diag：　グラフのループを許すか否かを指定する論理値。

R でブロックモデリングを実行したい場合，まず始めに隣接行列に対してクラスター分析をしておかなければなりません。関数には `equiv.clust()` を使用します。この関数の詳細に関して，重要な引数についてだけ以下に示しておきます。詳しく知りたい方はヘルプを参照して下さい。

```
equiv.clust(dat, method="hamming", mode="digraph", diag=FALSE,
            cluster.method="complete")
```

【引数】

dat：　1 つあるいは複数の隣接行列。

method：　距離関数を指定する。hamming や euclidean など。

mode：　無向グラフ（graph）か有向グラフ（digraph）かを指定する文字列。

diag：　グラフのループを許すか否かを指定する論理値。

cluster.method：　クラスター分析の種類を指定する。ward，single，complete，average，mcquitty，median，centroid から選択する。

ここでは例として，図 3 の隣接行列に対してブロックモデリングを行ってみましょう。なお，隣接行列におけるクラスター分析の結果は図 4 に示します。

```
# 隣接行列を作成する
x <- c(0,0,0,1,1,0,0,1,1,1,0,1,0,0,0,1,1,0,0,1,1,1,0,1,0)
dat <- matrix(x, 5, 5)
colnames(dat) <- c("a", "b", "c", "i", "j")
rownames(dat) <- c("a", "b", "c", "i", "j")
# クラスター分析のオブジェクトを作成する
clust <- equiv.clust(dat, method="euclidean", mode="graph",
                     cluster.method="ward")
plot(clust)
# クラスター数は 2 でブロックモデリングを実行。
blockmodel(dat, ec=clust, k=2, mode="graph")
```

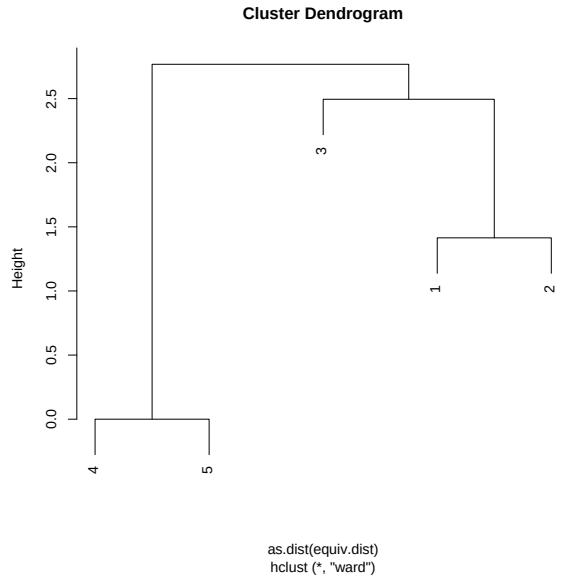

図 4　デンドログラム

結果の出力としては，以下のようなものが得られます。

```
> blockmodel(dat, ec=clust, k=2, mode="graph")

Network Blockmodel:

Block membership:

a b c i j
1 1 1 2 2

Reduced form blockmodel:

         a b c i j
          Block 1   Block 2
Block 1 0.3333333 0.6666667
Block 2 0.6666667 1.0000000
```

【Block membership:】のところを見ると，構造同値の関係にあるノード i とノード j が『2』というブロックにカテゴライズされていることが分かります。それ以外のノードはブロック『1』に分類されていますね。

また隣接行列を縮約したイメージ行列が，【Reducete form block model:】のところに表示されています。この行列の要素は 1 に近いほど，ブロック間に関係性が存在する可能性が高いと解釈されます。例えば，Block1 に所属するノード a, b, c の間には，全部で 3 本の無向辺が引けるにもかかわらず，実際には 1 本しか引かれていません。したがって $\frac{1}{3} = 0.3333333$ となり，これが Block1 と Block1 における要素の値となっています。