R を利用した社会ネットワーク分析

ネットワークの描画

1　はじめに

社会ネットワーク分析（SNA）を行うに当たって，ネットワークを描画し，視覚化することは重要です。なぜならネットワークを視覚的に表現することは，ネットワークの特徴を直感的に捉え，新たな発見につながるからです。

もちろん SNA の本道は，ネットワークから特徴となる情報を取り出し，構造を分析することであって，グラフを描画するだけでは分析したことにはなりません。しかし，多くの書籍で散見されるように最初からこれら諸指標を詳しく解説すると，退屈のため途中で挫折してしまい，結局何も身につかないということがよくあるのです。

なので教育的見地から見ると，まずはじめに視覚的にも派手な描画手法を解説することは有効だと思うのです。ここではそのような観点に立ち，まずはネットワークの描画方法を説明します。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
- rgl

インストール終了後，以下のコマンドを実行します。

```
library(sna)
library(rgl)
```

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，まずはこの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

ノード (node)：　頂点（vertex）とも言う。ネットワークの構成要素である点のこと。ミクロなネットワーク分析には個人が，マクロなネットワーク分析には集団がノードとなる。ノードが社会的単位であるような場合にはアクターという表現を用い，よりグラフ理論に即した表現をする場合にはノードという表現を用いることが多い。

辺 (edge)：　ノードとノードをつなぐネットワーク上の線。単一方向の関係の場合を有向辺，双方向の関係の場合を無向辺で表す。

有向グラフ (directed graph)：　辺が有向辺のみで構成されている場合，そのグラフは有向グラフと呼ばれる。

無向グラフ (non-directed graph)：　辺が無向辺のみで構成されている場合，そのグラフは無向グラフと呼ばれる。

連鎖グラフ（chain graph)：　ネットワークが複数の群（group）に分かれており，その群内では無向辺，群間では有向辺で結合されているとき，そのグラフを連鎖グラフという。

隣接行列 (adjacent matrix)：　ノードとノードの間に辺がある場合に 1 が，そうでない場合に 0 が立つ要素をもつ行列のこと。有向グラフの場合は非対称行列に，無向グラフの場合は対称行列になる。ノードが社会単位を表現するとき，隣接行列はソシオマトリクス (sociomatrix) と呼ばれることもある。

次数 (degree)：　あるノードが他のノードと隣接 (1 つの辺で結ばれていること) している数を次数という。

連結 (connected)・**非連結** (nonconnected)：　任意の 2 点のノードのペアを考えるとき，すべてのペアにおいて，ノード間が辺で結合されている場合，そのようなグラフは連結しているといわれる。そうでない場合は，非連結と呼ばれる。

3　2次元描画

社会ネットワーク分析では，ネットワークの図示はノード間の関係を考察する上で非常に重要です。関数 gplot() は行列形式で与えられる隣接行列からネットワークの 2 次元プロットを行います。関数 gplot() にはかなりの数の引数が準備されていますが，実際に使用する引数はごくわずかでしょう。ここでは最低限のものだけ紹介します。より詳しく知りたい方は以下のようにしてヘルプファイルを参照して下さい。

```
> ?gplot
```

3.1　関数詳細

```
gplot(dat, gmode, mode, interactive, label, displaylabels, boxed.labels,
      label.pos, label.cex, vertex.cex, vertex.col, edge.lty, edge.lwd,
      pad, diag)
```

【引数】

dat：　数値行列。隣接行列をここで指定する。

gmode：　グラフのタイプを指定する。"digraph"では有向グラフが，"graph"では無向グラフが描画される。また"twomode"は二部グラフを描画するときに利用する。デフォルトでは"digraph"が選択されている。

mode：　ノードの配置に関するアルゴリズム。円（"circle"），力学的モデルに基づくもの（"fruchtermanreingold"，"kamadakawai"，"spring"），固有ベクトルに基づくもの（"eigen"，"hall"，"princoord"），多次元尺度法（"mds"），ランダム（"random"）な

どが指定できる。

interactive：　TRUE にするとマウス操作で選択した頂点の位置を調整できる。

label：　ノードのラベルのベクトル。デフォルトではノードは番号。

displaylabels：　ノードにラベルを表示する場合に TRUE にする。

boxed.labels：　ノードのラベルをボックスで囲まないとき FALSE にする。

label.pos：　ノードに対するラベルの位置。プロット領域の中心から見て外側（0），下（1），左（2），上（3），右（4），頂点の位置（5）。

label.cex：　ラベルの文字の大きさ。

vertex.cex：　ノードの大きさ。ノードごとに指定する場合はベクトルにする。

vertex.col：　ノードの色。”red”，”white”，”blue”などのように指定する。

edge.lty：　辺における線の種類。実線（`"solid"`），破線（`"dashed"`），点線（`"dotted"`），一点破線（`"longdash"`），長い破線（`"longdash"`），長短交互の破線（`"twodash"`）。

edge.lwd：　辺の太さ。デフォルト値（1）に対する比率を正の数値で指定する。

pad：　余白の調整。

diag：　隣接行列の対角成分をデータとして使うとき TRUE にするとループが描かれる。

3.2　簡単な使い方

さっそくこの `gplot()` を使ってみましょう。まずは隣接行列を準備します。ここでは適当に，以下のような 5×5 の行列を準備します。

```
  A B C D E
A 0 0 1 1 0
B 0 0 1 0 0
C 0 0 0 1 1
D 1 1 0 0 1
E 1 0 0 0 0
```

これを R に読み込みます。読み込む方法が分からない方は簡単な R の入門書や Web を参照して下さい。

```
# データの読み込み
dat <- matrix(c(0,0,0,1,1,0,0,0,1,0,1,1,0,0,0,1,0,1,0,0,0,0,1,1,0), 5, 5)
# 行名の入力
rownames(dat) <- c("A", "B", "C", "D", "E")
# 列名の入力
colnames(dat) <- c("A", "B", "C", "D", "E")
# データを表示
dat
```

無事，上記のような行列が表示されたでしょうか？

では次に，この隣接行列（dat）をグラフで表示してみます。以下のコマンドを実行してみましょう。

```
gplot(dat)
```

すると図 1 のようなネットワーク図が別画面で出力されると思います。

しかしこのままではどの点が誰を表しているのかわかりません。そこでノードにラベルをつけて，ネットワークを描画してみましょう。使用する引数は displaylabels です。この引数はデフォルトでは FALSE に設定されていますので，これを TRUE に変更します。上記で使用した gplot(dat) という関数を以下のように変更して下さい。

```
gplot(dat, displaylabels=T)
```

すると，図 2 のようにラベル付きのグラフが描画されるでしょう。この図を見ればわかるように，隣接行列におけるある要素が『1』のとき，その行から列に向かって有向辺が接続されています。こ

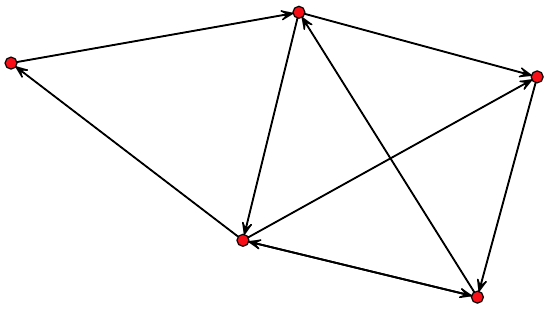

図 1　隣接行列のグラフ化（ラベルなし）

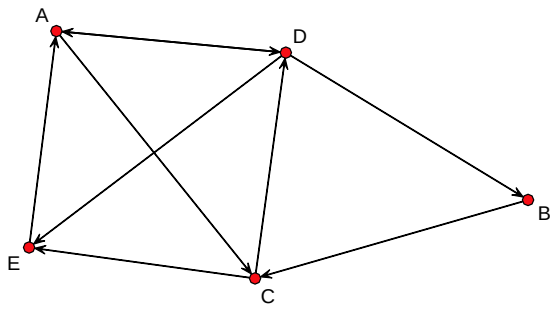

図 2　隣接行列のグラフ化（ラベルあり）

のままでも良いのですが，引数を色々指定することで，図 3 のようにもすることができます。

```
gplot(dat, displaylabels=T, vertex.cex=3, vertex.col="white", label.pos=5)
```

また mode という引数を使えば，描画の形を変えることができます。例えばノードを円環状に配置したければ以下のようなコマンドに変更します。

```
gplot(dat, mode="circle", displaylabels=T,
      vertex.cex=3, vertex.col="white", label.pos=5)
```

すると，図 4 のような形で描画されます。

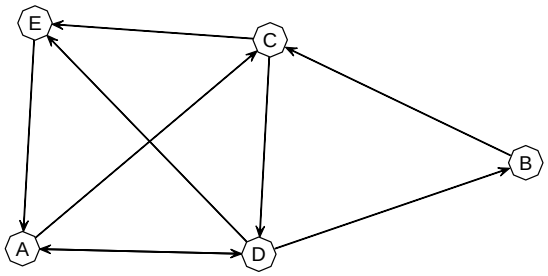

図 3　隣接行列のグラフ化（ラベルをノードの中に表示）

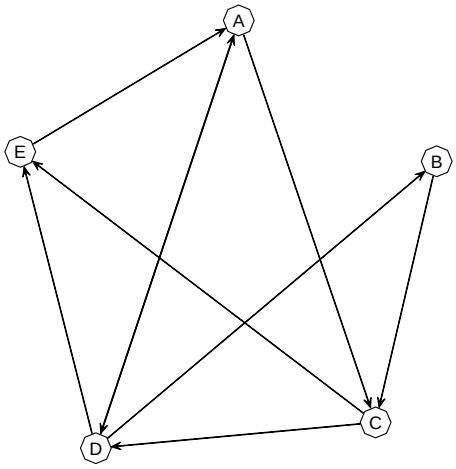

図 4　隣接行列のグラフ化（円環状に表示）

今回のように小規模なネットワークではどの図もそれほど見にくくはありませんが，ノー

ド数が増えてしまうと配置に工夫が必要になります。一般的には力学的モデルに基づくもの（"fruchtermanreingold", "kamadakawai", "spring"）が見やすいと言われています。僕自身は mode="kamadakawai"をよく利用します。

```
gplot(dat, displaylabels=T, mode="kamadakawai",
      vertex.cex=3, vertex.col="white", label.pos=5)
```

しかし，これら自動配置ではどうしても自分の思い通りには出力されてくれません。そこで interactive という引数を TRUE にします。すると出力ウィンドウにおけるネットワークのノードをマウスで自在に移動できるようになるのです。これを利用すれば，ほぼ思い通りのネットワーク配置を実現できるでしょう。念のため，以下にコマンドを記しておきます。

```
gplot(dat, displaylabels=T, interactive=T,
      vertex.cex=3, vertex.col="white", label.pos=5)
```

3.3　無向グラフの描画

無向グラフにおける無向辺は，2 つのノード間における双方向関係を表しています。したがって，無向グラフを描くために準備する隣接行列は，必然的に対称行列になります。例えば以下のような行列を考えてみましょう。

```
  A B C D E
A 0 0 1 1 0
B 0 0 1 0 0
C 1 1 0 1 1
D 1 0 1 0 1
E 0 0 1 1 0
```

これを R に読み込みます。

```
# データの読み込み
dat <- matrix(c(0,0,1,1,0,0,0,1,0,0,1,1,0,1,1,1,0,1,0,1,0,0,1,1,0), 5, 5)
# 列名の入力
rownames(dat) <- c("A", "B", "C", "D", "E")
# 行名の入力
colnames(dat) <- c("A", "B", "C", "D", "E")
# データを表示
dat
```

そしてこのオブジェクトを以下のように関数に食わせることで無向グラフを描くことができます。ポイントは引数 gmode を gmode="graph"に指定することです。これが『無向グラフを描け』という命令になります。

```
gplot(dat, gmode="graph", mode="kamadakawai", displaylabels=T, vertex.cex=3,
      vertex.col="white", label.pos=5)
```

図 5 のようなグラフが出力されたでしょうか？

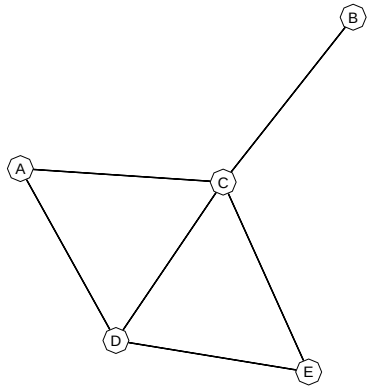

図 5　無向グラフの描画

4　3 次元描画

R ではネットワークを 3 次元で描画することもできます。が，これは分野にもよるのですが，ほとんど使うことはないと思います。少なくとも僕自身が研究などで 3 次元描画を使用したことは一切ありません。

そうは言っても，プレゼンテーション時のグラフの見栄えを良くしたり，あるいはデモンストレーションや自分自身が楽しむために役立つ（？）こともあるかもしれませんので，一応説明しておきます。

とは言っても 2 次元の時とそれほど変わりません。ただ `gplot3d()` という関数を使えば良いだけです。関数の詳細はヘルプファイルを参照して下さい。ここではさきほどの無向グラフの隣接行列を用いて，3 次元ネットワークを描画してみます。

```
gplot3d(dat, gmode="graph", mode="kamadakawai", displaylabels=T)
```

上記のコマンドを実行すると，図 6 のようなグラフが出力されるでしょう。

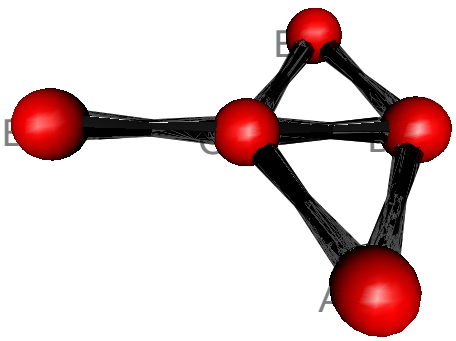

図 6　3 次元無向グラフの描画

ただし，3 次元の場合，図の保存の仕方に少し工夫がいります。と言うのも，3 次元図はマウスのクリックなどで保存することはできず，R のコマンドを利用しないといけないのです。ここでは png と ps の 2 種類のフォーマットで保存するコマンドを以下に書いておきます。

```
# nongraph3d.png というファイル名でワーキングディレクトリに保存
rgl.snapshot(filename="nongraph3d.png")
# nongraph3d.png というファイル名でワーキングディレクトリに保存
rgl.postscript(filename="nongraph3d.ps", fmt="ps")
```

上記のコマンドを実行するとワーキングディレクトリ（作業フォルダ）に図が保存されているはずです。ワーキングディレクトリの場所が分からない方は以下のコマンドを実行して保存場所を確認してみて下さい。

```
getwd()
```