SNA Basics IV
Multiple Networks

@ Sean F. Everton

Place a header at the top of your scripts that tell you what the script does, what its name is, etc.

```
##################################################
# What: Multiple Networks in R
# File: snab4.R
# Created: 02.28.14
# Revised: 06.19.18
##################################################
```

Data

The data we will use in this exercise are the Sampson Monastery network data collected by Samuel Sampson. Sampson observed and recorded the social interactions among a group of novices (men who were preparing to join a monastic order). He recorded four types of "ties": esteem (SAMPES) and disesteem (SAMPDES); liking (SAMPLK – three different time periods recorded) and disliking (SAMPDLK – one-time period recorded); positive influence (SAMPIN) and negative influence (SAMPNIN); praise (SAMPPR) and blame (SAMPNPR). Each novice only ranked his top three choices for each type of tie where a 3 indicates their first choice, a 2 their second, and a 1 their third (some subjects offered tied ranks for their top four choices).

During Sampson's period of observation, a "crisis in the cloister" occurred in response to some of the changes proposed by the Second Vatican Council (Vatican II). This led to the expulsion of four novices and the voluntary departure of several others. Based on his observations, Sampson partitioned (i.e., sorted, divided) the novices into four groups: (1) the young turks, (2) the loyal opposition, (3) the outcasts, and (4) the neutrals. The young turks arrived later and questioned some of the monastery's practices, which the loyal opposition defended. The outcasts were novices that weren't accepted by the larger group, and the neutrals were those who didn't take sides in the debate. Most of the loyal opposition had attended a seminary, "Cloisterville," prior to their arrival at the monastery.

Setup

Clear the workspace each time before beginning.

```
rm(list=ls())
```

Set your working directory to where the data are, so you don't have to include the entire path when importing and exporting data, files, etc. Of course, you'll need to set this to your own directory.

```
setwd("~/Dropbox/Networks and Religion (Book)/Website/Labs/SNA Basics 4")
```

Multiple Networks in *statnet*

Next, we need to load the libraries we plan to use. The *sna* and *network* libraries are part of the *statnet* package; *intergraph* allows us to convert a *network* object to an *igraph* object, which is an SNA library that we'll use later. Because *igraph* and *sna* conflict with one another, we can't load them at the same time.

```
# Load libraries
library(sna)
library(network)
library(intergraph)
```

To stack networks in R we need to first convert them into objects that R will understand. First load the networks as a matrix (they must have the same dimensions and node sequence) and then check the dimensions

```
# Read in data
samplike1.mat <-as.matrix.network(read.paj("SAMPLK1.net"))
samplike2.mat <-as.matrix.network(read.paj("SAMPLK2.net"))
samplike3.mat <-as.matrix.network(read.paj("SAMPLK3.net"))

# Check the dimensions (just to be sure)
dim(samplike1.mat)
## [1] 18 18
dim(samplike2.mat)
## [1] 18 18
dim(samplike3.mat)
## [1] 18 18
```

Create network object that we can later use for graph labels and convert to igraph objects; note that the Sampson data are directed.

```
samplike1.net <- as.network(samplike1.mat,directed=TRUE)
samplike2.net <- as.network(samplike2.mat,directed=TRUE)
samplike3.net <- as.network(samplike3.mat,directed=TRUE)
```

We can stack the matrices by defining them as an array, as well as assign names to each of the networks, which we can then turn into a sociomatrix. We also check to see if the dimensions of the stacked network are correct.

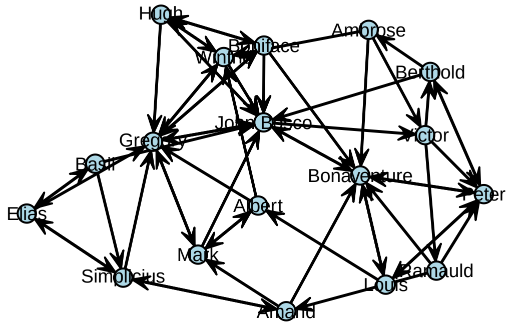
```
samplike.array <- array(data=list(samplike1.mat,samplike2.mat,samplike3.mat),
                  dimnames=list(c("samplike1", "samplike2", "samplike3")))

samplike.mat <- as.sociomatrix(samplike.array)

dim(samplike.array)
## [1] 3
dim(samplike.mat)
##             samplike1 samplike1
##          3         18        18
```

We can run the stackcount function to see how many networks are stacked together

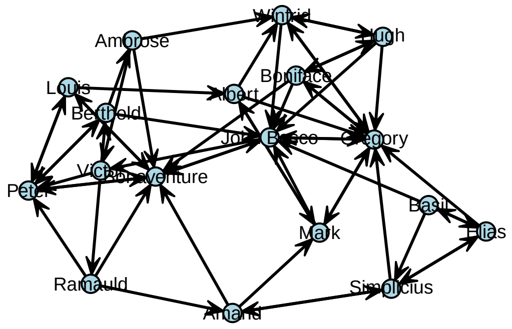
```
stackcount(samplike.array)
## [1] 3
stackcount(samplike.mat)
## [1] 3
```

We can extract individual networks if we like and plot them separately. Here we extract the Sampson liking network at time 2 two different ways and then visualize them. Remember that the size of the label is controlled by the function, “label.cex=” and the position of the label is controlled by the function, “label.pos=”

```
# If you want to extract one of the stacked matrices, you can get it from the array like this:
samp2.array <- samplike.array[2]
gplot(samp2.array,gmode="digraph",label=network.vertex.names(samplike2.net),
      vertex.col="Light Blue",label.col="black",label.cex=0.6,label.pos=5,
      usearrows=TRUE)
```

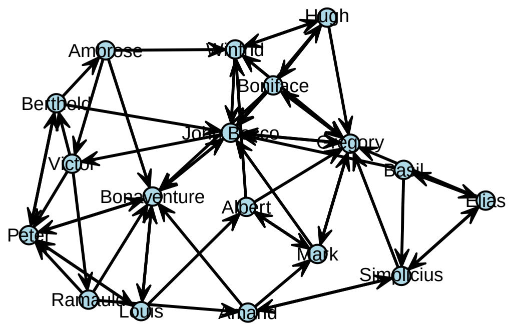


```
# Or from the sociomatrix like this where the spaces between commas refer to matrices, rows, and column
samp2.mat <- samplike.mat[2,,]
gplot(samp2.mat,gmode="digraph",label=network.vertex.names(samplike2.net),
      vertex.col="Light Blue",label.col="black",label.cex=0.6,label.pos=5,
      usearrows=TRUE)
```

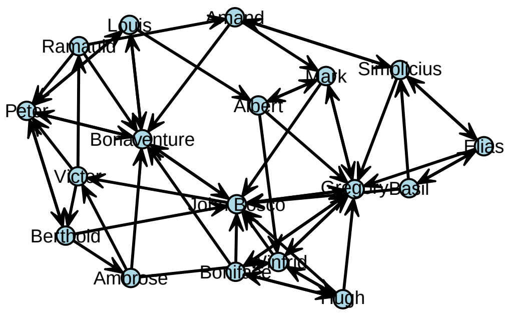


If we just wanted to visualize the networks, we don't have to go through all that. We can just do this:

```
gplot(samplike.array[2],gmode="digraph",label=network.vertex.names(samplike2.net),
      vertex.col="Light Blue",label.col="black",label.cex=0.6,label.pos=5,usearrows=TRUE)
```

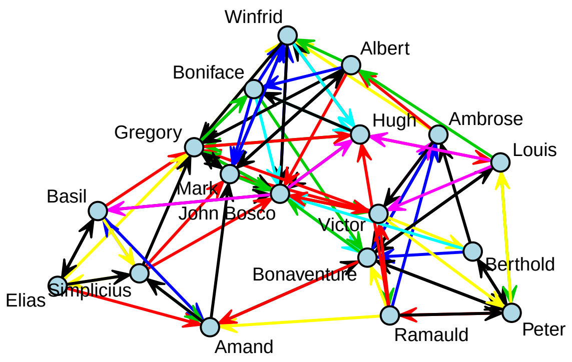


```
gplot(samplike.mat[2,,],gmode="digraph",label=network.vertex.names(samplike2.net),
      vertex.col="Light Blue",label.col="black",label.cex=0.6,label.pos=5,usearrows=TRUE)
```



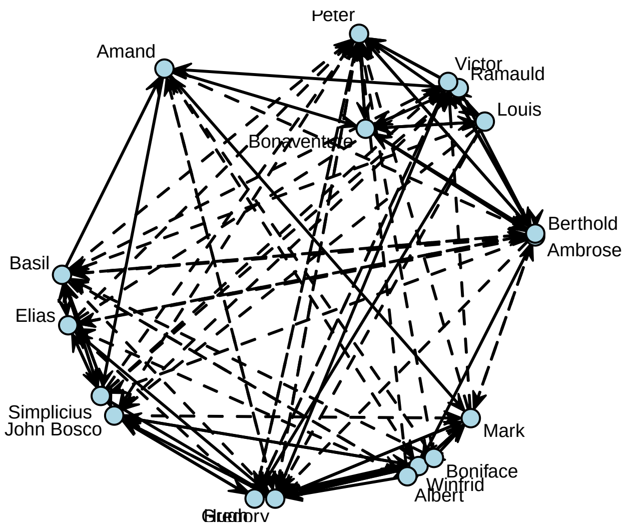
Now, let's plot the networks where the ties between the actors vary in color, based on the type of tie (liking at time 1, 2, or 3). The number folowing the brackets (e.g., "+3") indicates the color that will be generated. Note that here I didn't tell R where to place the labels, but let *sna* figure it out on its own.

```
gplot(samplike.mat[1,,]|samplike.mat[2,,]|samplike.mat[3,,],
      edge.col=2*samplike.mat[1,,]+3*samplike.mat[2,,]+4*samplike.mat[3,,],
      gmode="digraph",jitter=TRUE,label=network.vertex.names(samplike3.net),
      vertex.col="Light Blue",label.col="black",label.cex=0.6)
```



The UCINET portion of this lab created a network that combined the liking network at time 3 with the disliking network (recall that disliking was only recorded at one time point, presumably the third), where a positive number indicates that particular novice liked another and a negative tie indicates that a particular novice disliked another. This was saved as a .csv file, which we will now read into R, save as a network object, and then visualize.

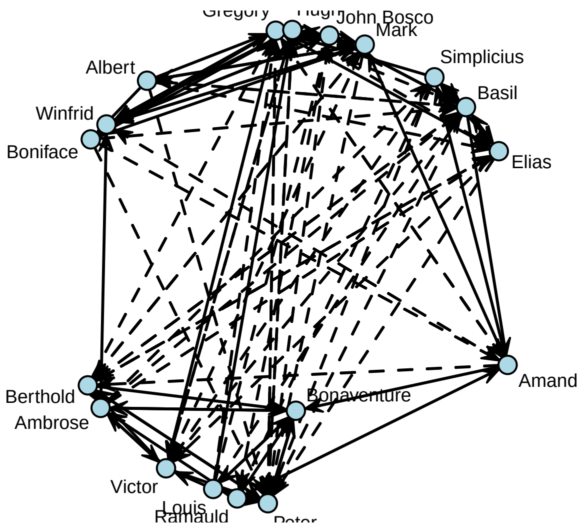
```
samplkdlk.mat <- as.matrix(read.csv(("SAMPLKDLK.csv"),header=TRUE,row.names=1,
                           check.names=FALSE))
samplkdlk.net <- as.network(samplkdlk.mat,directed=TRUE)

gplot(samplkdlk.mat,gmode="digraph",label=network.vertex.names(samplkdlk.net),
      vertex.col="Light Blue",label.col="black",label.cex=0.6,usearrows=TRUE)
```



Notice that the negative ties are drawn as dashed lines (as they are in Pajek), while the positive ties are drawn as solid lines. It also appears that negative ties push actors apart, while positive ties pull them closer (again, as in Pajek), which is what we want.

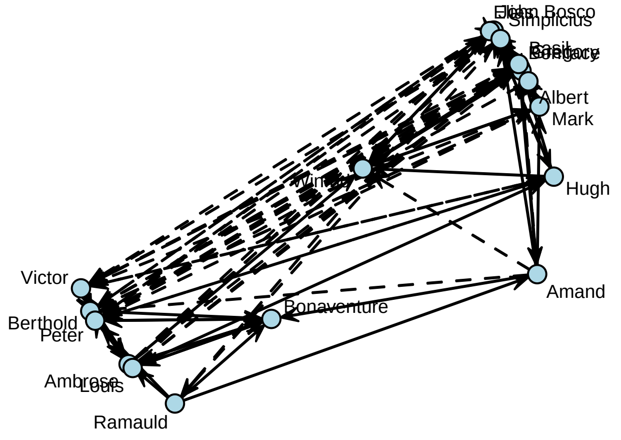
Here's a slight different way of doing the same thing.

```
samplkdlk.df <- read.csv(("SAMPLKDLK.csv"),header=TRUE,row.names=1,check.names=FALSE)
samplkdlk.net <- as.network(samplkdlk.df)
gplot(samplkdlk.df,gmode="digraph",label=network.vertex.names(samplkdlk.net),
      vertex.col="Light Blue",label.col="black",label.cex=0.6,usearrows=TRUE)
```



We can also read the liking and disliking data in directly and convert the disliking data to negatives, add the two graphs together, and then visualize

```
sampdislike.mat <-as.matrix.network(read.paj("SAMPDLK.net"))
sampdlk.mat <- sampdislike.mat * -1
samplkdlk.mat = sampdlk.mat + samplike3.mat
samplkdlk.net <- as.network(samplkdlk.mat)

gplot(samplkdlk.mat,gmode="digraph",label=network.vertex.names(samplkdlk.net),
      vertex.col="Light Blue",label.col="black",label.cex=0.6,usearrows=TRUE)
```



Multiple Networks in *igraph*

Let's move to igraph and see how we can do some of the same things in it. We need to first load *igraph* and detach *sna*.

```r
library(igraph)
detach(package:sna, unload=TRUE)
```

For *igraph*, we'll read the data in as edge lists and store them as data frames. Typically, edge lists don't contain zeros; they only include edges (ties) that actually exist (i.e., of tie strength "1" or higher). However, when "stacking" edge lists, we need to include edges of strength "0" so that all possible pairs of actors are included. Otherwise, it's impossible to merge them together.

If you are interested in what the edge lists look like, open the text files. These were created in UCINET although the headers created by UCINET were removed.

```r
samplike1.df <- read.table('SAMPLK1.txt')
samplike2.df <- read.table('SAMPLK2.txt')
samplike3.df <- read.table('SAMPLK3.txt')
```

Next, add column names to the three data frames

```r
colnames(samplike1.df) <- c('ego', 'alter', 'like1_tie')
colnames(samplike2.df) <- c('ego', 'alter', 'like2_tie')
colnames(samplike3.df) <- c('ego', 'alter', 'like3_tie')
```

Before we merge these three data frames, we need to make sure 'ego' and 'alter' are the same across data sets. We can compare each row using the == function. However, that generates a lot of output, so it's easeier to see which row entries are not equal using the syntax below:

```r
which(samplike1.df$ego != samplike2.df$ego)
## integer(0)
which(samplike1.df$ego != samplike3.df$ego)
## integer(0)
which(samplike2.df$ego != samplike3.df$ego)
## integer(0)
```

The "0s" indicate that all the rows are the same. Whew! I feel better. Now, we can combine them into a single data frame.

```r
samplikeall.df <- cbind(samplike1.df,samplike2.df$like2_tie,samplike3.df$like3_tie)
```

An alternative way to build the data frame is this way:

```
samplikeall.df <- data.frame(ego = samplike1.df[,1],alter = samplike1.df[,2],
    like1_tie = samplike1.df[,3],like2_tie = samplike2.df[,3],like3_tie = samplike3.df[,3])
```

Now, we can remove the edges of strength zero so that it only contains actual ties. We could convert the full edge list into an igraph object, but the visualizations work better (or at least some of them do) when the edges of strength zero are absent.

```
samplikeall.zero.df <- subset(samplikeall.df,(like1_tie > 0 | like2_tie > 0 | like3_tie > 0))
```

Convert the data frame to an igraph object

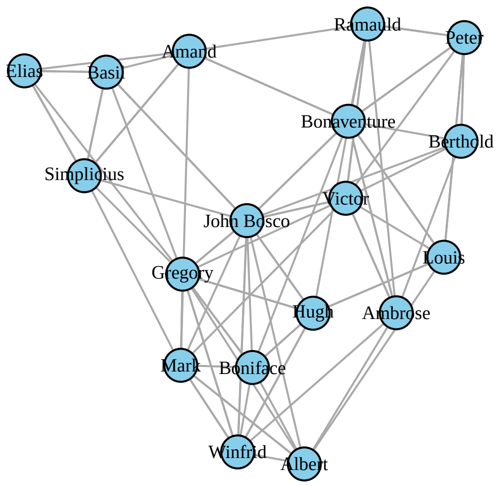
```
samplikeall.ig <- graph.data.frame(samplikeall.zero.df)
samplikeall.ig
## IGRAPH f019dbc DN-- 18 88 --
## + attr: name (v/c), like1_tie (e/n), like2_tie (e/n), like3_tie
## | (e/n)
## + edges from f019dbc (vertex names):
##  [1] 1 ->2  1 ->3  1 ->5  1 ->7  1 ->11 1 ->15 2 ->3  2 ->5  2 ->6  2 ->9
## [11] 2 ->15 3 ->2  3 ->7  3 ->8  3 ->14 4 ->2  4 ->3  4 ->5  4 ->9  5 ->1
## [21] 5 ->2  5 ->4  5 ->6  6 ->2  6 ->5  6 ->7  6 ->11 6 ->14 7 ->1  7 ->3
## [31] 7 ->4  7 ->5  7 ->9  7 ->10 8 ->9  8 ->10 8 ->11 8 ->13 9 ->2  9 ->7
## [41] 9 ->8  9 ->10 9 ->11 9 ->16 10->8  10->9  10->11 10->12 10->13 11->6
## [51] 11->8  11->9  11->10 11->12 12->2  12->8  12->9  12->10 12->11 12->13
## [61] 13->7  13->8  13->9  13->10 13->14 14->8  14->9  14->10 14->12 14->13
## + ... omitted several edges
```

Create a "names" vector and data.frame, add a column of numbers from 1 through 18, and then add the resulting data frame to Sampson data frame.

```
names <- network.vertex.names(samplike1.net)
names <- data.frame(names)

names = cbind(1:length(names[,1]),names)

samplikeall.ig <- graph.data.frame(d = samplikeall.zero.df,vertices = names)
```

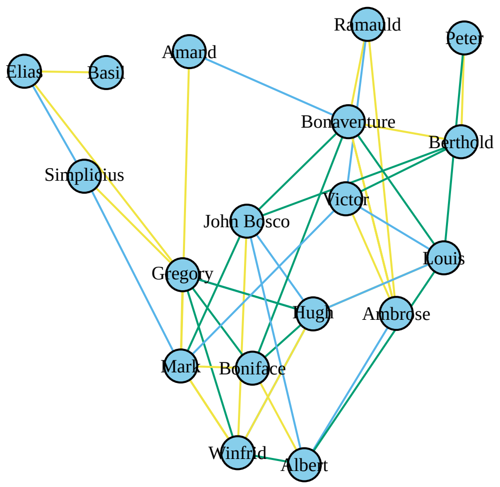
Let's set the names as vertex labels and the generate a plot to see if the names show up as labels. As we did in the previous lab, we'll save the coordinates as an attribute so that we will get the same layout each time.

```
V(samplikeall.ig)$label = V(samplikeall.ig)$names
samplikeall.ig$layout <- layout.fruchterman.reingold(samplikeall.ig)
plot(samplikeall.ig,vertex.color="Sky Blue",vertex.label.cex=.6,
     vertex.label.color="black",edge.arrow.mode=0)
```



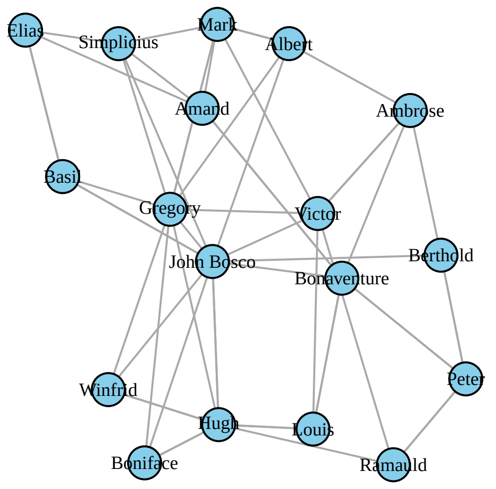
Let's set the color of the edges so that they'll be different for each type of tie. We'll set liking time 1 as red, time 2 as green, and time 3 as blue. These colors are based on the currently installed color palette in R.

```
palette()
## [1] "black"   "red"     "green3"  "blue"    "cyan"    "magenta" "yellow"
## [8] "gray"

E(samplikeall.ig)$color[E(samplikeall.ig)$like1_tie==1] = 2
E(samplikeall.ig)$color[E(samplikeall.ig)$like2_tie==1] = 3
E(samplikeall.ig)$color[E(samplikeall.ig)$like3_tie==1] = 4

plot(samplikeall.ig,vertex.color="Sky Blue",vertex.label.cex=.6,
     vertex.label.color="black",edge.arrow.mode=0)
```

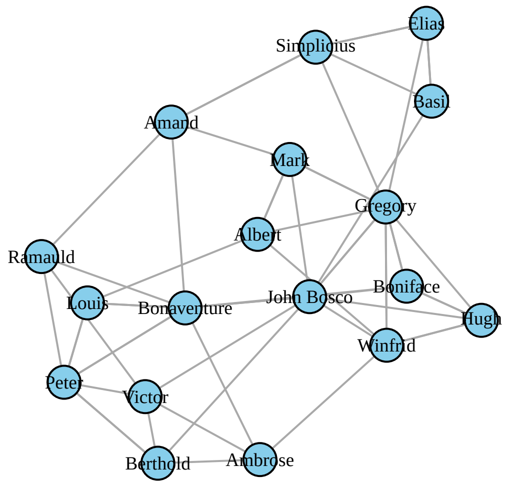


Here's how to extract the subnetworks from the larger network and then plot them in *igraph*

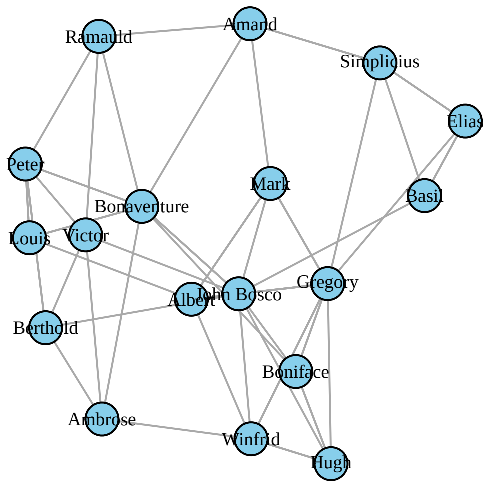
```
samplike1.ig <- delete.edges(samplikeall.ig,E(samplikeall.ig)[get.edge.attribute(samplikeall.ig,
                name = "like1_tie") == 0])
samplike2.ig <- delete.edges(samplikeall.ig,E(samplikeall.ig)[get.edge.attribute(samplikeall.ig,
                            name = "like2_tie") == 0])
samplike3.ig <- delete.edges(samplikeall.ig,E(samplikeall.ig)[get.edge.attribute(samplikeall.ig,
                             name = "like3_tie") == 0])

plot(samplike1.ig,layout=layout.fruchterman.reingold,edge.color="darkgrey",
     vertex.color="Sky Blue",vertex.label.cex=.6,vertex.label.color="black",
     edge.arrow.mode=0)
```



```
plot(samplike2.ig,layout=layout.fruchterman.reingold,edge.color="darkgrey",
     vertex.color="Sky Blue",vertex.label.cex=.6,vertex.label.color="black",
     edge.arrow.mode=0)
```



```
plot(samplike2.ig,layout=layout.fruchterman.reingold,edge.color="darkgrey",
     vertex.color="Sky Blue",vertex.label.cex=.6,vertex.label.color="black",
     edge.arrow.mode=0)
```



That's all for now