National Research University Higher School of Economics, Faculty of Social Sciences

Introduction to Network Analysis, Spring 2019

Seminar 2 Assignment, 01/25/2018

Dmitry Zaytsev, PhD and Valentina Kuskova, PhD

(with deep appreciation to all used sources; references available in text and upon request)

Welcome to the second seminar! Hopefully, you now have at least an idea of how R works. Wewill guide you every step of the way, and hopefully, the task of working in R will get easier with each passing week. Every seminar we will be adding packages to our toolchest; wewill clearly list them at the beginning of each seminar assignment. Before attempting to run any code, please make sure you have every necessary package installed.

Contents of today's seminar:

- In your Seminar 2 folder, you will find the following files:
    1. This file, "Seminar 2," in .pdf format.
    2. Dataset "flo.Rdata," already in a format, readable to R; there is a short tutorial in this Seminar about reading different dataset formats.
    3. Dataset "drugnet2.paj," in Pajek format - it is a format from a special program called Pajek, very popular in network analysis.
    4. Datasets: padgett.dat, padgw.dat, Data files in .dat format are common (come from Stata, for example), and they are also the files that UCINet, a relatively common network package, usually reads data from.
- For today's assignment, you will need the following packages (remember, R is case sensitive, make sure, when installing packages, to keep the appropriate case:
    1. rmarkdown
    2. network

    Remember that to use certain package, you need to make sure that the library is installed (using install.packages("packagename") command). To use a loaded package, you call it with "library.(packagename)" command.
- After completing today's seminar, you should be able to accomplish the following:
    1. Learn how to load network data into R.
    2. Generate simple and not-so-simple graphs out of network data.
    3. Obtain simple network characteristics out of network data.
    4. Compare and interpret obtained network results. That's right, you are already network analysts!

Seminar 2 assignment. Please reproduce the R code in this document and answer the questions that are marked as *Assignment questions*. Please include your answers right after each question in your RMarkdown file. You should submit both the .Rmd and the .pdf files with results.

Seminar 2 deadline: February 01, 2019 by 23:59.

Reading network data in R

Please make sure your working directory is set to a folder where the data files are located; otherwise, you will be forced to provide a full path to data every time.

Before we can do any manipulations on data, we have to read data into R. You have already seen that reading data means assigning it to an object of some sort by using the "<-" operator. That part was easy. But network data comes in many different forms, and we need to learn how to read different types of data.

Here is a short reference for loading data into R: http://www.statmethods.net/input/importingdata.html.

To manipulate data, we need to have a set of packages with special commands (the list is provided above). Let's start by loading the "network" package.

```
# This installs the network package
## install.packages('network')
# This loads the network package
library(network)
```

```
## network: Classes for Relational Data
## Version 1.13.0.1 created on 2015-08-31.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##                     Mark S. Handcock, University of California -- Los Angeles
##                     David R. Hunter, Penn State University
##                     Martina Morris, University of Washington
##                     Skye Bender-deMoll, University of Washington
##  For citation information, type citation("network").
##  Type help("network-package") to get started.
```

Creating a network object from the matrix data

One of the most common ways to represent a network is an *edgelist*, or a list of ties. Once we have imported it into R, we can transform it into a network object using the *network* package.

Generating a matrix of data

We could use an existing data matrix, but it could be fun to create our own - so let's go ahead and do that. We are creating a matrix of data, or sociomatrix - one of the most common types of data.

```
# We are first creating a variable that will tell matrix generator how many nodes to store.
# Let's have 15 nodes (you can choose any number).
num_nodes <- 15
# Next, generate the matrix using the build-in "matrix" command in R
my_matrix<-matrix(round(runif(num_nodes*num_nodes)), # edge values
                          nrow = num_nodes, #nrow must be same as ncol
                          ncol = num_nodes)
# Next, let's make sure there are no self-referencing loops
# (meaning, the node is not connected to itself)
diag(my_matrix) <- 0
# We can check dimensions of the new object:
dim(my_matrix)
```

```
## [1] 15 15
```

```
# You should see that matrix dimensions are equal to the number of nodes you have specified.
# You can also check the class of your new dataset:
class(my_matrix)
```

```
## [1] "matrix"
```

```
# Let's check whether any data is missing:
sum(is.na(my_matrix))
```

```
## [1] 0
```

There should not be any missing data, because we have generated the matrix ourselves. However, with real-life data, it is not always the case, and with network data especially, missing data could be a problem. We will talk more about this in lectures.

So, now let's go ahead and create an edgelist from our matrix. Command for doing so is "as.network," and you can learn more about it by reading the help files.

```
# Command below shows the help file for the as.network command.
# Uncomment it to see the info.
##?as.network
my_network<-as.network(x = my_matrix, # the network object
                  directed = TRUE, # specify whether the network is directed
                  loops = FALSE, # do we allow self ties (should not allow them)
                  matrix.type = "adjacency" # the type of input
                  )
```

And now we have the network!

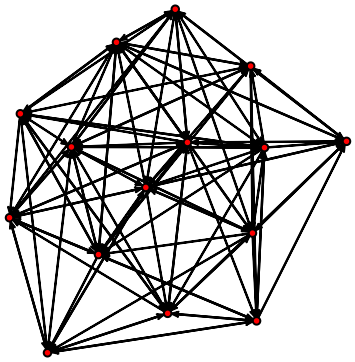
Working with edgelist data

We can do our first tests on our newly created network:

```
network.size(my_network)
```

```
## [1] 15
```

```
# Command below provides all information about the data and shows the matrix,
# from which we created the data.
# It could be helpful in case we did not start with the matrix first, but with the edgelist first.
# We commented it out, because the output takes a lot of space.
# Experiment with the command on its own in the console.
## summary(my_network)
# View your network:
par(mar=c(1,1,1,1)) # get rid of the large margins and plot the data:
plot(my_network)
```



Importing other data formats

Network data will generally come as a raw text file or as a file saved in another analysis tool. The package *foreign* allows you to load several data formats with functions that follow the same syntax as to the **read.table()** function in base R.

The following sniplet of code installs and provides information about a library "foreign," which allows you to read other data types into R. Uncomment the code to see what it does.

```
##install.packages("foreign")
##library(foreign)
##?read.dta # reads STATA files
##?read.xport  # reads SAS xport files
```

While *foreign* will help you read data saved in other statistical packages, network data usually come as either raw text or from a specialized network analysis software such as UCINet.

Reading a Pajek file

Pajek is an awesome network program, and if we have time, we'll look at it, though most of its capabailities are now incorporated in R. The package *network* includes the function **read.paj()**, which allows you to load a Pajek project file into R. We will demo **read.paj()** reading data from a respondent-driven sample (RDS) study of drug users in Hartford, CT. This redacted dataset allows us to observe how the structure network data sampled through RDS differs from the structure of both complete and personal networks. The relation measured is referral into the study. Please remember that for the code below to work, the datafile, "drugnet2.paj," has to be in the same folder as your .Rmd file or where the console reads from.

```
drugpaj <- read.paj('drugnet2.paj')
names(drugpaj) # objects in drugpaj
```

```
## [1] "networks"   "partitions"
```

```
names(drugpaj$networks)
```

```
## [1] "C:\\Users\\Ann\\Desktop\\drugnet.net"
```

```
names(drugpaj$partitions)
```

```
## [1] "C:\\Users\\Ann\\Desktop\\drugnet_gender.clu"
```

```
## [2] "C:\\Users\\Ann\\Desktop\\drugnet_ethnicity.clu"
```

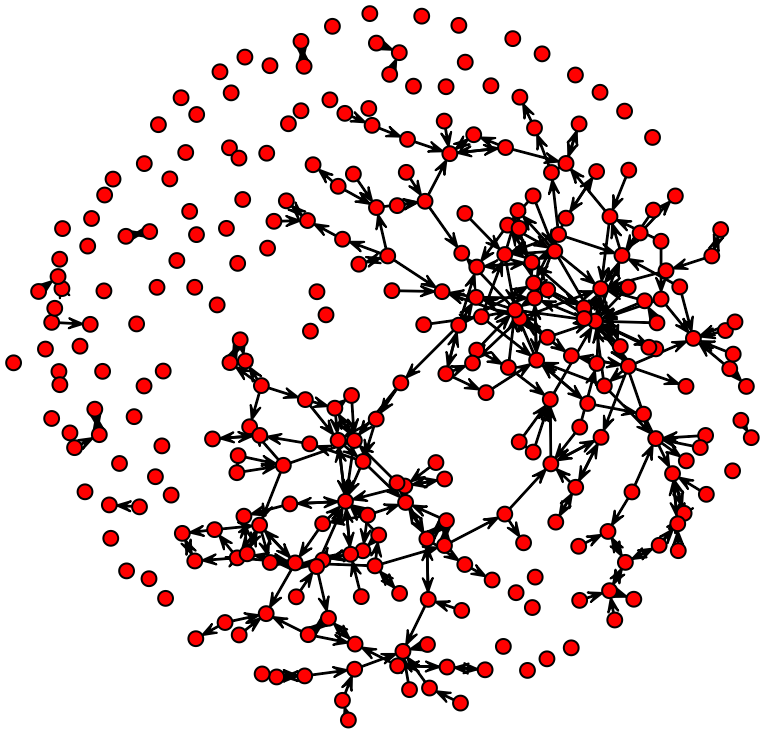
```
drug <- drugpaj$networks[[1]] # extract network
class(drug)
```

```
## [1] "network"
```

```
drug  # examine network object
```

```
##  Network attributes:
##   vertices = 293
##   directed = TRUE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   title = C:\Users\Ann\Desktop\drugnet.net
##   total edges= 337
##     missing edges= 0
##     non-missing edges= 337
##
##  Vertex attribute names:
##     vertex.names x y z
##
##  Edge attribute names:
##     C:\Users\Ann\Desktop\drugnet.net
```

```
plot(drug)
```



Assignment question: what looks strange about this network? Why?

You can further examine the network:

```
network.size(drug) # how many nodes?
```

```
## [1] 293
```

```
network.edgecount(drug) # how many edges?
```

```
## [1] 337
```

```
network.dyadcount(drug) # how many dyads?
```

```
## [1] 85556
```

Assignment question: What do the numbers above represent?

Remember, we talked that nodes can have attributes? How do we set attributes to nodes, if these attributes are provided as a separate column in the data? The code for doing so is provided below.

```
names(drugpaj$partitions) # attributes included
```

```
## [1] "C:\\Users\\Ann\\Desktop\\drugnet_gender.clu"
## [2] "C:\\Users\\Ann\\Desktop\\drugnet_ethnicity.clu"
```

```
gender <- drugpaj$partitions[[1]] # extract gender
gender #check the values assigned to each gender
```

```
##   [1] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1
##  [36] 1 2 1 1 1 1 1 1 1 1 2 2 1 1 1 1 2 1 2 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1
##  [71] 2 2 1 1 2 1 1 1 1 1 2 1 1 1 1 1 2 2 2 1 2 2 1 1 1 2 1 1 2 1 2 1 1 1 1
## [106] 1 1 1 1 1 2 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
## [141] 1 1 1 1 2 1 1 2 1 2 2 1 1 1 1 2 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
## [176] 1 1 2 1 1 1 1 1 2 1 1 1 1 0 1 1 1 1 1 2 1 1 1 0 1 2 0 2 1 2 2 1 1 2 1
## [211] 1 2 1 1 1 2 1 2 1 1 1 1 1 2 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2
## [246] 1 2 2 2 2 1 1 1 1 1 2 2 2 1 0 0 2 2 2 2 2 2 2 0 2 2 2 2 2 2 2 2 2 1 2 2
## [281] 2 1 0 2 2 2 2 2 2 2 2 2 2
```

```
table(gender) # how many women/men
```

```
## gender
##   0   1   2
##   7 200  86
```

```
# It is actually better to recode gender into 1 and 0 as opposed to 2 and 1;
# setting 0/1 variable as an attribute is cleaner that way.
female <- ifelse(gender == 2, 1, # recode
            ifelse(gender == 1, 0, NA))
## set attributes
drug <-set.vertex.attribute(drug, 'female', value=c(female))
ethnicity <- drugpaj$partitions[[2]] # extract ethnicity
table(ethnicity) # how is it coded?
```

```
## ethnicity
##   1   2   3   4
##  25  99 155  14
```

```
drug <-set.vertex.attribute(drug, 'ethnicity', value=c(ethnicity))
```

Reading a native R data file

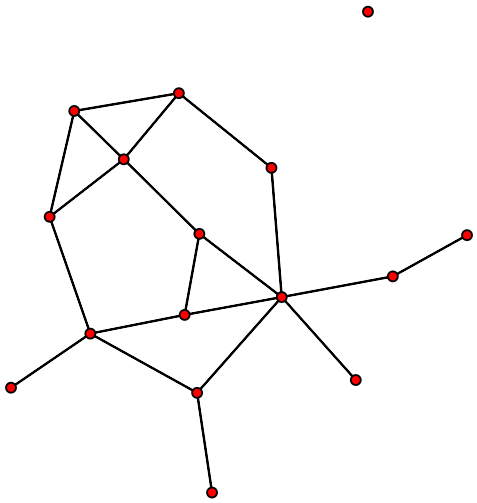
One of the data packages, the Florentine Family network, we will be working with extensively in this class. Because it comes with R package, it is already in an R format. For the purposes of this class, we have extracted it as a separate file, which is provided in your Seminar 2 folder.

With native R files, the command is simply "load."

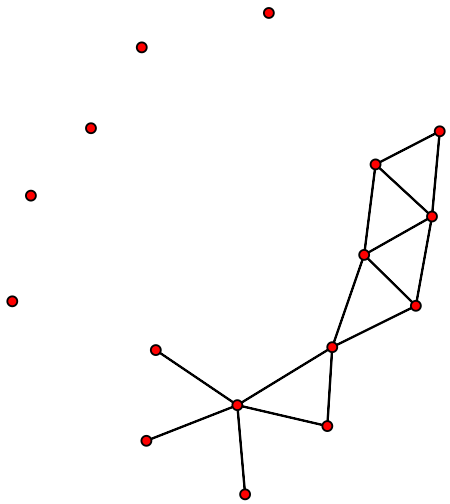
```
load('flo.Rdata')
```

Once the data is loaded, we can start manipulating it to turn it into a network and obtain all relevant network characteristics:

```
flo.marriage <- as.network(as.matrix(flo.marriage), directed=FALSE)
flo.biz <- as.network(as.matrix(flo.biz), directed=FALSE)
# Add attributes
set.vertex.attribute(flo.marriage, 'wealth', flo.att[,2])
set.vertex.attribute(flo.biz,'wealth', flo.att[,2])
# Simple plots:
par(mar=c(0,0,0,0))
plot(flo.marriage)
```



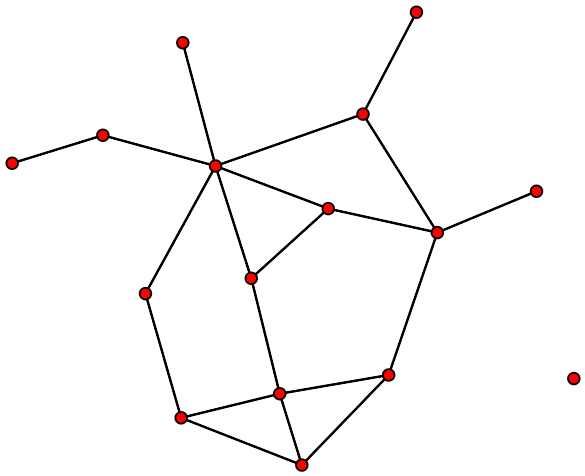
```
plot(flo.biz)
```

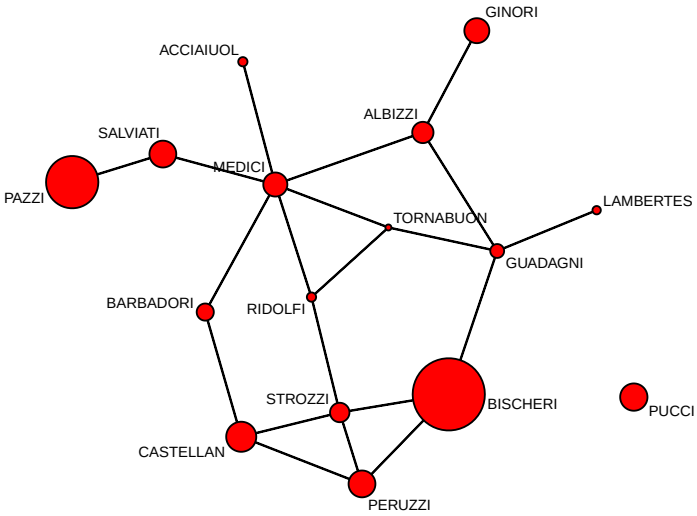


Of course, what fun would it be to have attributes and not plot them? Those attributes we have loaded we can now include in the graphs.

```
# Look at networks with node sizes proportional to wealth
# Notice that one of the option in this command is to set coordinates, so that pictures are equal.
# Otherwise, plot with attributes may differ from the original plot (compare with plot above).
par(mar=c(0,0,0,0))
plot(flo.marriage,
vertex.cex=(get.vertex.attribute(flo.marriage, 'wealth')/25 +.4),
```

```
    displaylabels=TRUE,
    label.cex=.5,
    label.pos=0,
    coord=plot(flo.marriage))
```





Assignment task: Please plot the biz network with node attributes that you've set above.

Importing UCINet files directly into R

UCINet files are space-delimited text files with a header. If you inspect the file, you can figure-out how to read it in R:

1. Open the UCINet file in R studio (as a text file)
2. Note the line number where the data actually starts
3. Read the file into R starting at that line number
4. Use information from the header to name variables

We will demo this process with Padgett's Florentine Families data, which we've seen already, and which is included as a part of a different set of data. Reading commands are different for .dat types of files, so please carefully step through the code below:

```
# Read vertex labels with scan()
flo.names <- scan('padgett.dat', what='character', skip=4, nlines=16)
# Read data with read.table()
flos <- read.table('padgett.dat', skip=41, col.names=flo.names)
# Read node attributes:
flo.att <- read.table('padgw.dat',
col.names =c('WEALTH','NUM.PRIORS','NUM.TIES'), skip=25)
flo.att
```

```
##    WEALTH NUM.PRIORS NUM.TIES
## 1      10         53        2
## 2      36         65        3
## 3      27         38        4
## 4     146         74       29
## 5      55          0       14
## 6      44         12        9
## 7      20         22       18
## 8       8         21       14
## 9      42          0       14
## 10    103         53       54
## 11     48          0        7
## 12     49         42       32
## 13     10         35        5
## 14     48          0        7
## 15     32          0        9
## 16      3          0        1
```

Please take a look at the flo.att data. As you can tell, it is simply a set of data, which are all attributes. How do we add it to network? Check the code below.

```
flo.att <-cbind(flo.names,flo.att)
# Command below provides first six rows of data, so that you can check your data without loading the en
head(flo.att)
```

```
##   flo.names WEALTH NUM.PRIORS NUM.TIES
## 1 ACCIAIUOL     10         53        2
## 2   ALBIZZI     36         65        3
## 3 BARBADORI     27         38        4
## 4  BISCHERI    146         74       29
## 5 CASTELLAN     55          0       14
## 6    GINORI     44         12        9
```

In this particula network setup, marriage network and business networks are joined in one file. It could happen and with other data you use; sometimes, you need to separate the files. Code below shows how to break up files into needed chunks.

```
# Separate adjacency matrices
flo.marriage <-flos[1:16,]  # subset of the first 16 colums is the marriage network
dim(flo.marriage)
```

```
## [1] 16 16
```

```
row.names(flo.marriage) <-flo.names # name
flo.biz <- flos[17:32,] # subset of the second 16 is the business network.
row.names(flo.biz) <-flo.names # name
dim(flo.biz)
```

```
## [1] 16 16
```

```
# Check the data by listing a couple of rows and colums from each network.
flo.marriage[1:2,1:2]
```

```
##           ACCIAIUOL ALBIZZI
## ACCIAIUOL         0       0
## ALBIZZI           0       0
```

```
flo.marriage[15:16,15:16]
```

```
##           STROZZI TORNABUON
## STROZZI         0         0
## TORNABUON       0         0
```

```
flo.biz[1:2,1:2]
```

```
##           ACCIAIUOL ALBIZZI
## ACCIAIUOL         0       0
## ALBIZZI           0       0
```

```
flo.biz[15:16,15:16]
```

```
##           STROZZI TORNABUON
## STROZZI         0         0
## TORNABUON       0         0
```

Now that matrices are separated and set up in the way we need them to be set up, let's create networks from data and add attributes to nodes.

```
flo.marriage <- as.network(as.matrix(flo.marriage),directed=FALSE)
flo.biz <- as.network(as.matrix(flo.biz),directed=FALSE)
## add attributes
set.vertex.attribute(flo.marriage, 'wealth', flo.att[,2])
set.vertex.attribute(flo.biz,'wealth', flo.att[,2])
```

Assignment task: Using the code already shown, plot both of the new networks. Add attributes if you wish.

Saving Network Data in R

If you will be using the dataset in R, it's best to save as 'filename.Rdata' but if you plan on sharing the data, then you will want to save it as text.

```
# Save several objects in the same .Rdata file to load all at once
save(flo.marriage, flo.biz, file='floNets.Rdata')
# Save network as an edgelist in a .csv file
drug.edges <- as.matrix(drug, matrix.type='edgelist')
write.csv(drug.edges, file='drugsEdgelist.csv',row.names=FALSE)
```

Assignment task: For the network "drug" that we created and loaded with attributes, create several different network plots, adding gender and ethnicity to the graph as node attributes. Consider using a variety of colors to make your plot more informative.
