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

Introduction to Network Analysis, Fall 2019

Seminar 3 and Homework 2 Assignment, 01/02/2019

Dmitry Zaytsev, PhD and Valentina Kuskova, PhD

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

Welcome to the third seminar! By now, we hope, you are getting a feel for what we do in this course. Before attempting to run any code, please make sure you have every necessary package installed; however, pay attention. Today we will learn that some packages interfere with each other, so you will need to unload some before you can use others.

Contents of today's seminar:

- In your Seminar 3 folder, you will find the following files:
  1. This file, "Seminar 3 and Homework 2," in .pdf format.
  2. Dataset "flo.Rdata," which we've seen before; also, in the same format - "madmen.Rdata" and "trade.Rdata.""
- 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. RColorBrewer
  3. network
  4. sna
  5. igraph. *Please note:* this package may conflict with "network" package, especially on graphics. Therefore, when using one, it is always wise to unload the other (using command "detach(package:packagename)", or you may get an error in execution.

  Remember that to use a 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 calculate and interpret network characteristics that we have covered in lecture.
  2. Acquire additional skills for network graphs.
  3. Compare and interpret obtained network results. That's right, you are already network analysts!

Seminar 3 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.

Homework 2 assignment is provided at the end of this document.

Both assignments are due on Friday, February 08, 23.59.

Experimenting with graphs

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.

More on plotting attributes

In Seminar 2, we learned how to assign to graphs node attributes that were based on volume of an attribute characteristic - we showed you how to change node size based on wealth. In cases where an attribute is a interval-ratio variable, changing the node is intuitive. But what about assigning attributes given by nominal-scale variables? (For those most adventurous this assginment was given as an extra credit, but it's time for all of us to learn the technique.) In this case, we can use either color or node graphic mode (circle, square, triangle, etc.).

Let's look at how to do it with yet another dataset, "madmen.Rdata," the data for the sexual network of the TV show "Mad Men," which we extracted from the R package *gcookbook*:

```
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.
```

```
load('madmen.Rdata')
```

If you check your Data tab in the upper right-hand corner you will see that 'madmen.Rdata' contains two files: "mad.att"" (obviously, the node attributes) and "mad.matrix"" (the matrix of data). We have seen such structure before and know how to extract data from these two types of files. Let's check whether the attributes and the adjacency matrix are in the same order:

```
dim(mad.att)
```

```
## [1] 45  2
```

```
head(mad.att)
```

```
##                   Names Female
## 32          Abe Drexler      0
## 12              Allison      1
## 44 Bellhop in Baltimore      0
## 13     Bethany Van Nuys      1
## 1           Betty Draper      1
## 14       Bobbie Barrett      1
```

```
mad.matrix[1:6,1:2]
```

```
##                       Abe Drexler Allison
## Abe Drexler                     0       0
## Allison                         0       0
## Bellhop in Baltimore            0       0
## Bethany Van Nuys                0       0
```

```
## Betty Draper                    0       0
## Bobbie Barrett                  0       0
```

```
sum(as.character(mad.att[,1]) == colnames(mad.matrix))
```

```
## [1] 45
```

```
# Ok, everything looks fine, let's make the network:
mad.net <- as.network(mad.matrix, directed=FALSE)
# Let's look at attributes:
mad.att
```

```
##                       Names Female
## 32              Abe Drexler      0
## 12                  Allison      1
## 44     Bellhop in Baltimore      0
## 13         Bethany Van Nuys      1
## 1              Betty Draper      1
## 14           Bobbie Barrett      1
## 33 Brooklyn College Student      0
## 15                  Candace      1
## 2                Don Draper      0
## 16                   Doris      1
## 34            Duck Phillips      0
## 17              Faye Miller      1
## 27                 Franklin      0
## 28              Greg Harris      0
## 36                  Gudrun      1
## 3               Harry Crane      0
## 10            Henry Francis      0
## 25                   Hildy      1
## 39          Ida Blankenship      1
## 40              Jane Siegel      1
## 29                   Janine      1
## 26           Jennifer Crane      1
## 4             Joan Holloway      1
## 18                     Joy      1
## 45            Kitty Romano      1
## 5               Lane Pryce      0
## 35                    Mark      0
## 19            Megan Calvet      1
## 20            Midge Daniels      1
## 41           Mirabelle Ames      1
## 42            Mona Sterling      1
## 6               Peggy Olson      1
## 7             Pete Campbell      0
## 37        Playtex bra model      1
## 21            Rachel Menken      1
## 11               Random guy      0
## 30            Rebecca Pryce      1
## 8            Roger Sterling      0
## 9                Sal Romano      0
## 22                  Shelly      1
## 23          Suzanne Farrell      1
## 31                    Toni      1
## 38           Trudy Campbell      1
```

```
## 43                    Vicky      1
## 24 Woman at the Clios party      1
```

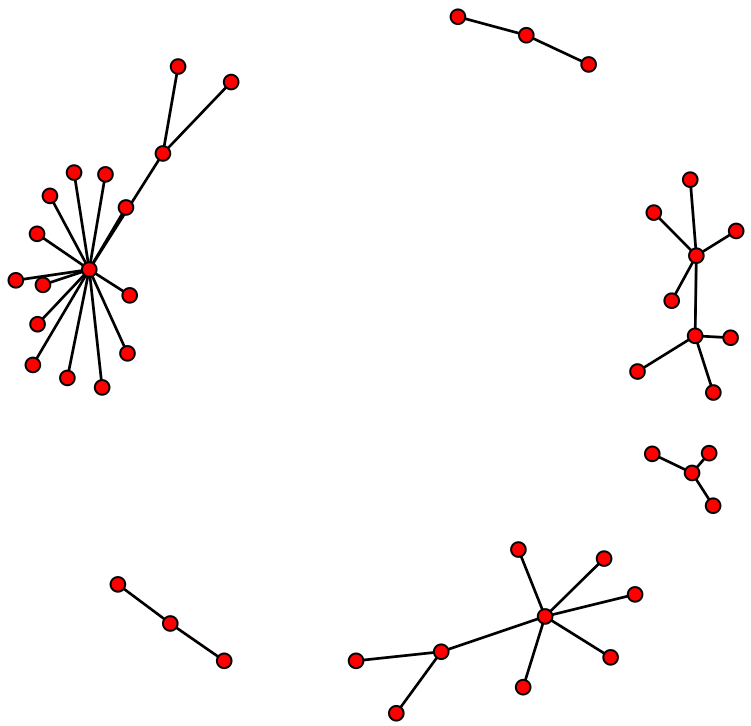
```
# So apparently, we have at least gender that is relatively intuitive.
# Now let's assign attributes to nodes, just as we did last seminar:
set.vertex.attribute(mad.net,attrname='female', value=mad.att[,2])
mad.net #check - we can see that nodes have the attribute "female"
```

```
##  Network attributes:
##   vertices = 45
##   directed = FALSE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 39
##     missing edges= 0
##     non-missing edges= 39
##
##  Vertex attribute names:
##     female vertex.names
##
## No edge attributes
```

Assignment question: Why did we use option FALSE for command "directed" above, when creating a network?

Now, let's plot the network, and then change graph structure by assigning different colors to nominal-type attributes:

```
par(mar=c(1,1,1,1)) # set margins
plot(mad.net)
```

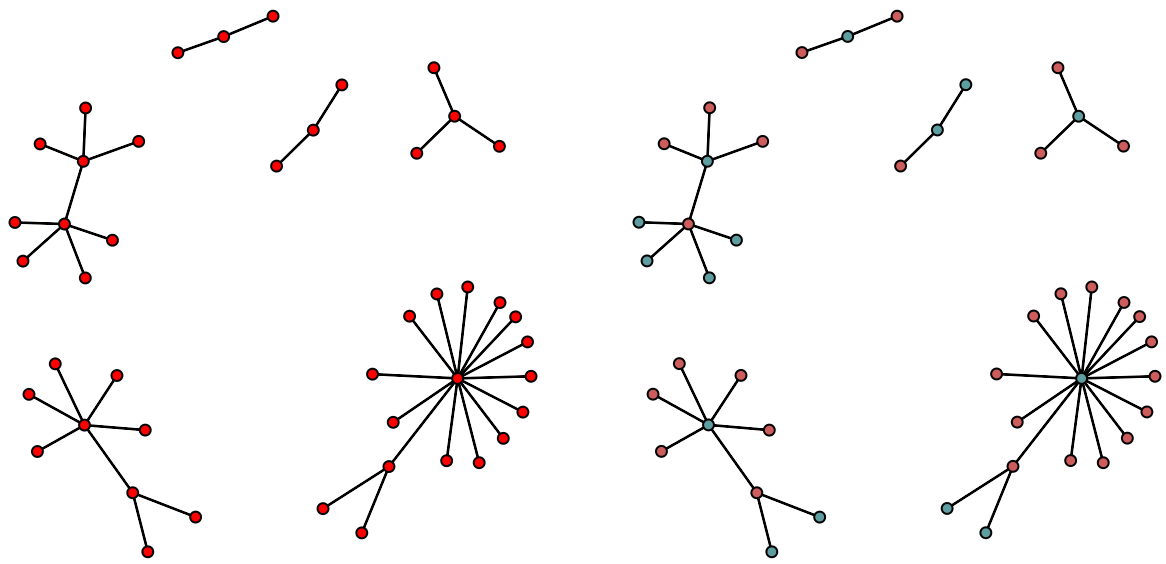


Next, set colors by gender and plot the network together with the original coordinates:

```
# We happen to know a few colors by heart and we assign them to color attributes
colors <- ifelse(mad.att$Female == 1, 'indianred', 'cadetblue')

# Setup the page by making smaller margins and putting two plots side-by-side:
par(mar=c(0,0.5,0,0.5),mfrow=c(1,2)) #divide the space into one row, two columns

#You can now see two networks side by side, one colored and one not:
plot(mad.net, vertex.col = colors,coord=plot(mad.net))
```



Saving plots as separate files

It is entirely possible that you may need to extract a plot from your work and use it as a separate file. There are several options for saving graphical data outside of your R environment. After you execute commands below, plots will be saved as separate documents in your working directory.

Saving plots as separate .pdf files

```
pdf('myplot.pdf', width=4, height=4) # width and height are in inches
par(mar=c(0,0,0,0))
plot(mad.net, vertex.col = colors, mode = 'fruchtermanreingold' )
dev.off() # this function turns off the pdf-maker, otherwise program may get confused
```

```
## pdf
##   2
```

Saving plot as .png files

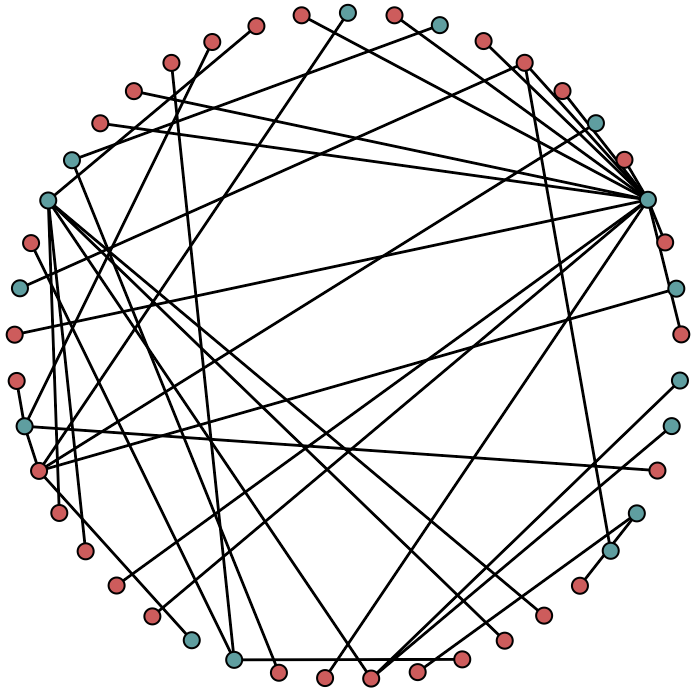
```
# Note that for png files height and width are given in pixels
png('myplot.png')
par(mar=c(0,0,0,0))
plot(mad.net, vertex.col = colors)
dev.off() #Important to turn off .png-maker as well
```

```
## pdf
##   2
```

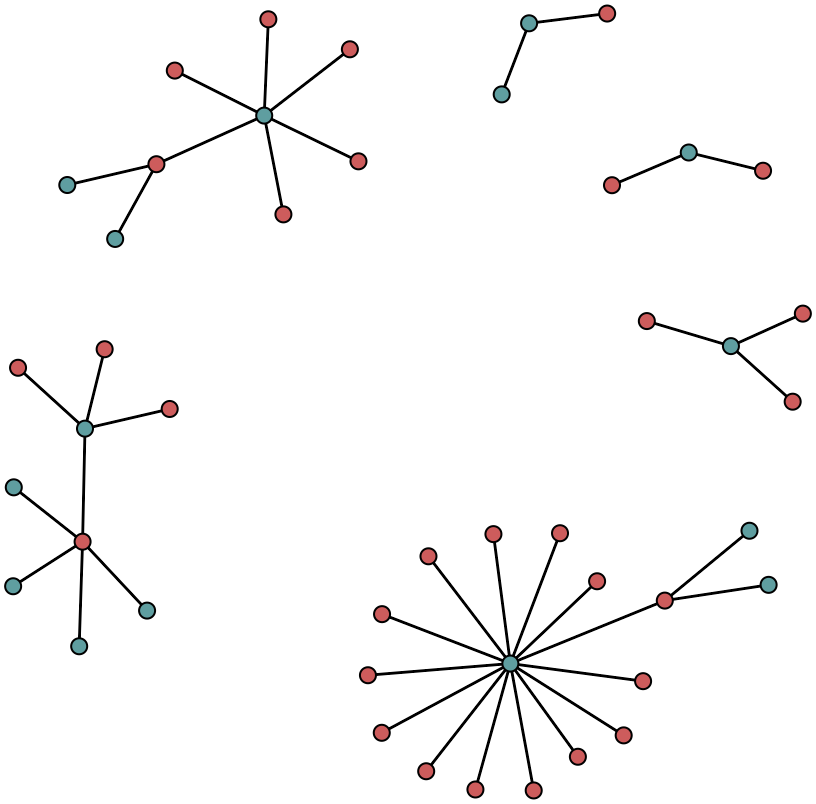
Experimenting with network layouts

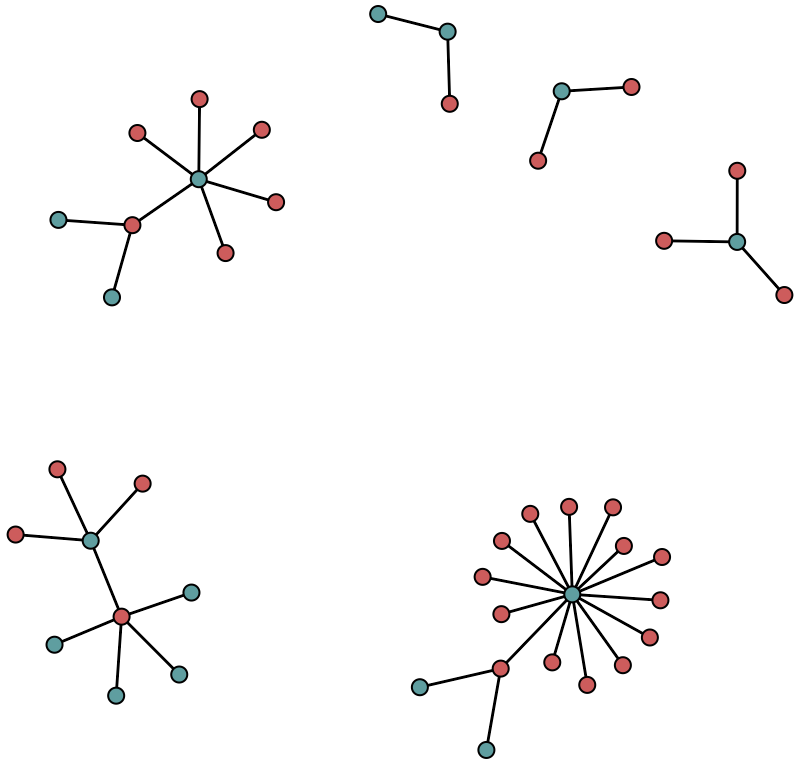
As we have talked, there is no single best way to plot data. Package *network*, which we are working with, has different built-in network layouts: "circle," where vertex attributes are "forced" to create a circle, "fruchtermanreingold," which generates a layout based on force-directed placement algorithm of Fruchterman and Reingold, and "kamadakawai," which generates a vertex layout based on a version of Kamada-Kawai force-directed placement algorithm. You can find all the arguments that the last two layouts take by using the ?network.layout command. Let's look at the difference between them:

```
par(mar=c(0,0,0,0))
plot(mad.net, vertex.col = colors, mode = 'circle')
```



```
plot(mad.net, vertex.col = colors, mode = 'fruchtermanreingold')
plot(mad.net, vertex.col = colors, mode = 'kamadakawai')
```





Experimenting with colors

Network graphs are all about making the information more clear; colors play a crucial role in this task. Fortunately, R provides us with a variety of ways to color-code our graphics. Let's explore one particular package that provides such functionality, *RColorBrewer*. Start by (installing, for those who do not have it installed and) loading the package into your workspace.

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

Next, let's set up the page with appropriate margins (command "par") and divide it into a 2x3 set of rows and columns (command "mfrow"), so that our color palettes display neatly, and then explore the contents of the *RColorBrewer* package.

Each palette many colors, and the number of them differs somewhat depending on the palette. They can be sequential (which are great for ordered data, such as "on a scale from 1 to 5," where increasing color demonstrates increasing value), diverging or qualitative. Each palette has a name, and you have to call it by name. To explore the package, use the "?brewer.all" help request.

The coolest part about this package is you can look at your colors before you start using them: it's a "display" command. Let's look at the contents of the palette named "Paired." Note that in the parentheses we use number 12 - it's the number of colors in the palette that we obtained from the help file. This number will be different depending on the package you select.

```
par(mar=c(2,2,2,2))
display.brewer.pal(12, 'Paired')
```

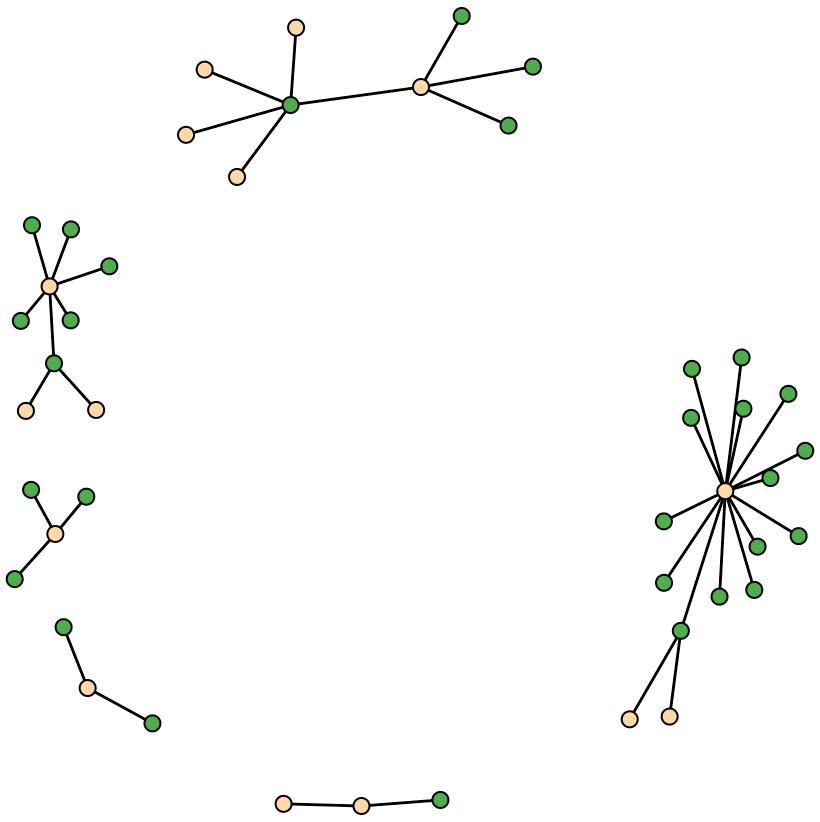
Sometimes it's helpful to see several palettes side by side. Code below allows you to do that, though to make palettes easy to see, we only selected five colors from each.

```
par(mar=c(1,1,1,1),mfrow=c(2,3))
# Palettes come as matrices, where columns correspond to colors:
display.brewer.pal(5, 'Accent')
display.brewer.pal(5, 'Dark2')
display.brewer.pal(5, 'Pastel1')
display.brewer.pal(5, 'Set1')
display.brewer.pal(5, 'Set2')
display.brewer.pal(5, 'Set3')
```

You can use one or several palettes at a time. To use the palette you want, assign it to an object that will be easier to use. To not get confused, we name my sets based on the original palette name.

```
#
col1 <- brewer.pal(5, 'Set1') #pick this set for bright colors
colPastel<-brewer.pal(5, 'Pastel1') #pick this set for pastel colors

# Assign colors from your chosen palette
# Remember that the column in a palette corresponds to a color
# We pick bright green and pale vanilla
# They correspond to columns 3 and 5 in their respective palettes
colors <- ifelse(mad.att$Female == 1, col1[3], colPastel[5])
par(mar=c(0,0,0,0))
plot(mad.net, vertex.col = colors )
```



Assignment task: Please examine the options in the "network.layout"" command and perform the following:

1. Create the madmen.net with labels.
2. Experiment with options by adding attributes, changing vertex or edge colors, finding the best position for labels. While this task may take a while, it will count as complete if you generate at least one graph that is different from the graphs I've shown you in this assignment. The more different graphs with options you generate, the better - extra-credit never hurts anyone.

Calculating network statistics

While it was a lot of fun drawing graphs (and we will continue to learn new things about them with each seminar), it is important to start generating statistical information about our networks.

Basic network measures

These are the items we talked about in detail in Lecture 2. For this assignment, we go back to our familiar dataset, Florentine Families.

```
# Load data and create networks:
load('flo.Rdata')
flo.marriage<-as.network(flo.marriage)
```

Next, install and load the package for social network analysis, "sna" (uncomment the code you need)

```
##install.packages("sna")
library(sna)
```

```
## Loading required package: statnet.common
```

```
##
## Attaching package: 'statnet.common'
```

```
## The following object is masked from 'package:base':
##
##     order
```

```
## sna: Tools for Social Network Analysis
## Version 2.4 created on 2016-07-23.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##  For citation information, type citation("sna").
##  Type help(package="sna") to get started.
```

And, onto the code for the quantities we have discussed:

Basic network statistics:

```
# The basics:
network.dyadcount(flo.marriage) #self-explanatory, dyads
```

```
## [1] 240
```

```
dyad.census(flo.marriage) # types and count of dyads
```

```
##      Mut Asym Null
## [1,]  20    0  100
```

```
network.edgecount(flo.marriage) # number of ties
```

```
## [1] 40
```

```
network.density(flo.marriage) #density of an overall network
```

```
## [1] 0.1666667
```

```
triad.census(flo.marriage) # types and counts of triads
```

```
##      003 012 102 021D 021U 021C 111D 111U 030T 030C 201 120D 120U 120C 210
## [1,] 324   0 195    0    0    0    0    0    0    0  38    0    0    0   0
##      300
## [1,]   3
```

```
gtrans(flo.marriage, measure='weak') # transitivity
```

```
## [1] 0.1914894
```

```
# Paths require a more involved command:
kpath.census(flo.marriage, mode = "digraph",
     tabulate.by.vertex = FALSE, path.comembership = "none",
              dyadic.tabulation = "none")
```

```
## $path.count
##   1   2   3
##  40  94 174
```

```
#Of course, if you set tabulate.by.vertex=TRUE, you will get
# counts of all path lengths by individual vertex.

# Same applies to vertices (including by vertex):
kcycle.census(flo.marriage, maxlen = 4, mode = "digraph",
     tabulate.by.vertex = FALSE, cycle.comembership = "none")
```

```
## $cycle.count
##  2  3  4
## 20  6  4
```

Geodesic distances

Because geodesic distance are different for every pair of nodes (you understand why, right?), we actually get a matrix of values that indicate the length of the shortest path between those nodes. Because it's a matrix, it makes sense to assign it to an object to do additional work on it:

```
# The command is simply geodist, but we will assign it to an object:
geo.dist<-geodist(flo.marriage)
class(geo.dist) # Check the structure
```

```
## [1] "list"
```

```
# Ok, it's a list, let's look at it:
summary(geo.dist)
```

```
##        Length Class  Mode
## counts 256    -none- numeric
## gdist  256    -none- numeric
```

```
# We can also summarize individual elements.
# These summaries give us all relevant information for all nodes.
summary(geo.dist$counts)
```

```
##        V1              V2              V3              V4
##  Min.   :0.00   Min.   :0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.:1.00   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000
##  Median :1.00   Median :1.000   Median :1.000   Median :1.000
##  Mean   :1.25   Mean   :1.062   Mean   :1.125   Mean   :1.562
##  3rd Qu.:1.25   3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:2.250
##  Max.   :3.00   Max.   :2.000   Max.   :2.000   Max.   :3.000
##        V5              V6              V7              V8
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000
##  Median :1.000   Median :1.000   Median :1.000   Median :1.000
##  Mean   :1.188   Mean   :1.062   Mean   :1.312   Mean   :1.312
##  3rd Qu.:1.250   3rd Qu.:1.000   3rd Qu.:2.000   3rd Qu.:2.000
##  Max.   :2.000   Max.   :2.000   Max.   :2.000   Max.   :2.000
##        V9             V10             V11             V12
##  Min.   :0.00   Min.   :0.00   Min.   :0.00   Min.   :0.0000
##  1st Qu.:1.00   1st Qu.:1.00   1st Qu.:1.00   1st Qu.:0.0000
##  Median :1.00   Median :1.00   Median :1.00   Median :0.0000
##  Mean   :1.25   Mean   :1.25   Mean   :1.25   Mean   :0.0625
##  3rd Qu.:1.25   3rd Qu.:1.25   3rd Qu.:2.00   3rd Qu.:0.0000
##  Max.   :3.00   Max.   :3.00   Max.   :2.00   Max.   :1.0000
##       V13             V14             V15             V16
##  Min.   :0.0000   Min.   :0.00   Min.   :0.000   Min.   :0.000
```

```
##  1st Qu.:1.0000   1st Qu.:1.00   1st Qu.:1.000   1st Qu.:1.000
##  Median :1.0000   Median :1.00   Median :1.000   Median :1.000
##  Mean   :0.9375   Mean   :1.25   Mean   :1.062   Mean   :1.188
##  3rd Qu.:1.0000   3rd Qu.:1.25   3rd Qu.:1.000   3rd Qu.:1.250
##  Max.   :1.0000   Max.   :3.00   Max.   :2.000   Max.   :2.000
```

```
summary(geo.dist$gdist)
```

```
##        V1             V2             V3            V4             V5
##  Min.   :0.00   Min.   :0.00   Min.   :  0   Min.   :0.00   Min.   :0.00
##  1st Qu.:2.00   1st Qu.:1.75   1st Qu.:  2   1st Qu.:1.75   1st Qu.:1.75
##  Median :3.00   Median :2.00   Median :  2   Median :2.00   Median :3.00
##  Mean   : Inf   Mean   : Inf   Mean   :Inf   Mean   : Inf   Mean   : Inf
##  3rd Qu.:3.25   3rd Qu.:3.00   3rd Qu.:  3   3rd Qu.:3.25   3rd Qu.:3.25
##  Max.   : Inf   Max.   : Inf   Max.   :Inf   Max.   : Inf   Max.   : Inf
##        V6             V7            V8            V9            V10
##  Min.   :0.00   Min.   :  0   Min.   :  0   Min.   :0.00   Min.   :0.00
##  1st Qu.:2.75   1st Qu.:  1   1st Qu.:  2   1st Qu.:1.00   1st Qu.:3.00
##  Median :3.00   Median :  2   Median :  3   Median :2.00   Median :3.50
##  Mean   : Inf   Mean   :Inf   Mean   :Inf   Mean   : Inf   Mean   : Inf
##  3rd Qu.:4.00   3rd Qu.:  3   3rd Qu.:  4   3rd Qu.:2.25   3rd Qu.:4.25
##  Max.   : Inf   Max.   :Inf   Max.   :Inf   Max.   : Inf   Max.   : Inf
##       V11            V12           V13           V14            V15
##  Min.   :0.00   Min.   :  0   Min.   :0.00   Min.   :0.00   Min.   :  0
##  1st Qu.:1.75   1st Qu.:Inf   1st Qu.:1.75   1st Qu.:2.00   1st Qu.:  1
##  Median :3.00   Median :Inf   Median :2.00   Median :2.50   Median :  2
##  Mean   : Inf   Mean   :Inf   Mean   : Inf   Mean   : Inf   Mean   :Inf
##  3rd Qu.:4.00   3rd Qu.:Inf   3rd Qu.:2.25   3rd Qu.:3.25   3rd Qu.:  3
##  Max.   : Inf   Max.   :Inf   Max.   : Inf   Max.   : Inf   Max.   :Inf
##       V16
##  Min.   :0.00
##  1st Qu.:1.75
##  Median :2.00
##  Mean   : Inf
##  3rd Qu.:3.00
##  Max.   : Inf
```

Now, that's a lot of information, and for large networks especially, it makes reading the output difficult. Therefore, we can run the code for one node at a time:

```
summary(geo.dist$counts[,3])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   1.000   1.125   1.000   2.000
```

```
summary(geo.dist$counts[3,])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   1.000   1.125   1.000   2.000
```

Assignment question: What can you say about the last two commands? Why is the result what it is?

Reciprocity

Here, we show you how to calculate this measure and its different types, but the information you get will be

meaningless. Do you understand why?

```
grecip(flo.marriage, measure = 'dyadic') # why not 0?
```

```
## Mut
##   1
```

```
grecip(flo.marriage, measure = 'dyadic.nonnull')
```

```
## Mut
##   1
```

```
grecip(flo.marriage, measure = 'edgewise')
```

```
## Mut
##   1
```

```
grecip(flo.marriage, measure = 'edgewise.lrr')
```

```
##      Mut
## 1.791759
```

```
grecip(flo.marriage, measure = 'correlation')
```

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

Now, that was a lot of new information. We realize that you might start feeling overwhelmed, but please trust us, in time it will get better. Just do not give up and keep at it.

Homework 2

As we have told you, homework assignments are somewhat more involved than regular seminars. They will require you to explore commands and data on your own, and go beyond simply duplicating the data. Usually the assignment consists of the following parts:

1. Demonstrating new packages or commands.
2. Asking you to provide interpretive information for the commands you execute.
3. Requiring you to perform the same or similar analytic/interpretive work on a different set of data.

Today's homework has several tasks:

1. To explore the capabilities of a new package, *igraph*. Because this is a graphical package, it utilizes some of the same R functions as "network" package, and the two may conflict. Therefore, while not required, it is always advisable to unload the *network* package before attempting to install and load *igraph*.
2. To use previously learned (in other courses) theoretical material to explore the possibility of changing network ties types (called dichotomization).

The data

For this task, we will be working with yet another dataset, "trade." This is a Smith and White world trade dataset, described in Wasserman and Faust (1994, our text book) and also available through UCINet. The data records interaction of the countries with respect to trade of manufactured goods, food and live animals, crude materials (not food), and minerals and fuels. An additional matrix records exchange of diplomats between countries. All trade (including the diplomants) flows from the row to the column for the matrix.

The trade attribute data lists average population growth between 1970 and 1981, average GNP growth per capita over the same period, secondary school enrollment ratio in 1981, and energy consumption in 1981 (in kilo coal equivalents per capita). Load the data in order to explore it:

```
# Load data and perform a few checks we already know:
load('trade.Rdata')
```

The following objects should have loaded into your workspace:

- **trade.all** Sum of trade ties;
- **manufacture** Manufacture trade matrix;
- **food** Food trade matrix;
- **crude** Crude materials (not food, not oil, not minerals) trade matrix;
- **minerals** (Including oil and natural gas) trade matrix;
- **diplomacy** Diplomatic ties.

Please note that the first data matrix, trade.all, is simply a sum of all other matrices - in other words, it is a valued matrix that contains in the cells the number, indicating how many different flows (manufacture, food, crude, etc.) go from country A to country B. Other matrices are 0-1 matrices, indicating whether the ties are present.

Assignment questions. Please examine available matrices and answer the following questions:

1. Are the matrices symmetric?
2. What does that mean for resulting networks? Would they be directed or undirected?

Dichotomizing valued data

Many network analysis measures can only be calculated in dichotomous (tie present or absent) networks. Sometimes it will make sense to retain the valued data and sometimes you will need to reduce your data to a dichotomous network. For example, calcuating geodesic distances in some cases assumes that the network is binary; you can't find this measure on a valued network. There are, of course, many other measures that can only be applied to a 0-1 network, but which provide us with the wealth of data. So dichotomizing network data becomes an important issue.

The valued network data that we will use for this example is the "trade.all" matrix. As said above, the cells of the matrix contain the sum of all other available trades. In a theoretical sense, this matrix contains information on whether trade ties are available, and the strength of the trade relationship between two countries can be inferred by the value of the cell - theoretically, the more, the higher.

However, this network also provides a quick glance at whether any trade is present between two countries at all. Instead of looking for this information in all other matrices one at a time, we can quickly determine the tie presence from this matrix. Usually, for this purpose the data is dichotomized (cell contents are turned into 0-1 values).

The process of dichotomizing the network is easy, but selecting the correct level at which to dichotomize is not. Everything is clear with a zero - there is no flow. But what if the value is anywhere from 1 to 5, with 5 indicating that trades in all five categories are present? Should each value be equal to 1, or should we make some of them zero? This is called selecting the correct level for dichotomizing, and we should keep in mind:

1. Whether there is a theoretical justification for the cut-off we selected, and
2. What are the impirical implications of this cut-off.

Let's create several networks, each with a different level of dichotomy.

```
# First, check what data we are dealing with
class(trade.all)
```

```
## [1] "data.frame"
```

```
# we need a matrix
trade.all<-as.matrix(trade.all)
# First dichotomy is with any tie present,
# meaning 0 is a 0, and everything else is a 1:
trade.any <- ifelse(trade.all > 0, 1, 0)
# Second dichotomy is with any value greater than 1,
# so 0/1 become a 0, and everything else - a 1:
trade.2 <- ifelse(trade.all > 1, 1, 0)
# We can go on, but we will do the max,
# dichotomizing at 5 ties:
trade.max <- ifelse(trade.all == 5, 1, 0)
```

Assignment questions. With respect to the above actions, please answer the following:

1. How would you justify any of these choices? Please refer to specific social theories to make your answer more legitimate.
2. What are the impirical implication of these choices?

Package igraph

Let's plot these networks we've created by dichotomizing the trade.all data. We will be using the package *igraph*, and you can find more information about the plotting options here: http://igraph.sourceforge.net/doc/R/plot.common.html.

```
# Uncomment the code you need
##install.packages("igraph")
##detach(package:sna)
##detach(package:network)
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:sna':
##
##     betweenness, bonpow, closeness, components, degree,
##     dyad.census, evcent, hierarchy, is.connected, neighborhood,
##     triad.census
```

```
## The following objects are masked from 'package:network':
##
##     %c%, %s%, add.edges, add.vertices, delete.edges,
##     delete.vertices, get.edge.attribute, get.edges,
##     get.vertex.attribute, is.bipartite, is.directed,
##     list.edge.attributes, list.vertex.attributes,
##     set.edge.attribute, set.vertex.attribute
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

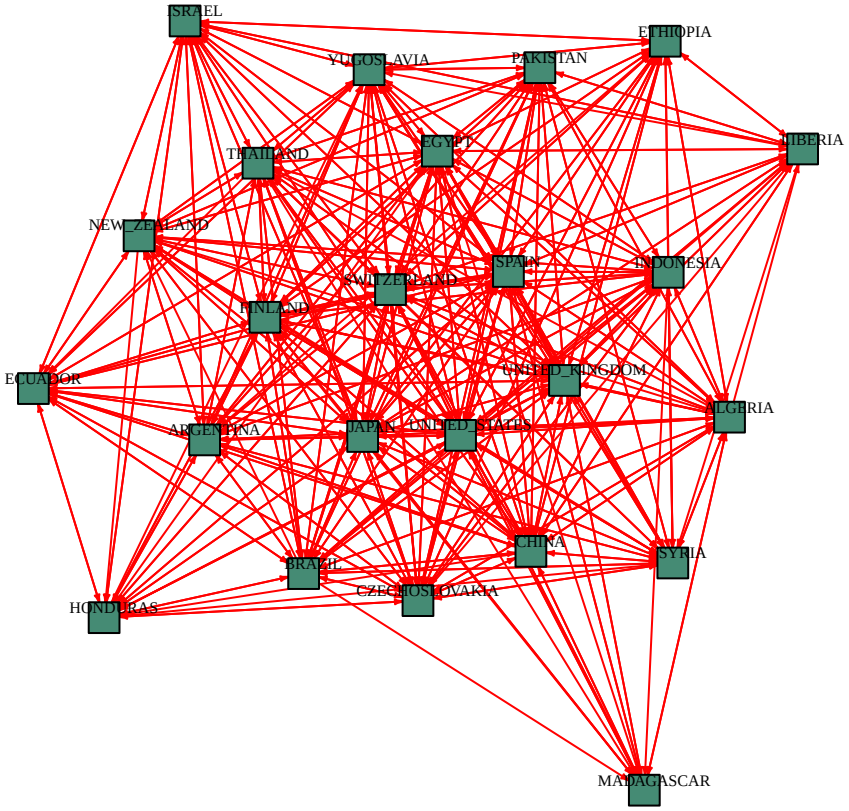
igraph requires that we convert matrices into graph adjacency matrices as follows:

```
tradegraph.any <-graph.adjacency(trade.any,
    mode=c("directed"),
    weighted=NULL,
    diag=FALSE)
tradegraph.2 <-graph.adjacency(trade.2,
    mode=c("directed"),
    weighted=NULL,
    diag=FALSE)
tradegraph.5 <-graph.adjacency(trade.max,
    mode=c("directed"),
    weighted=NULL,
    diag=FALSE)
```

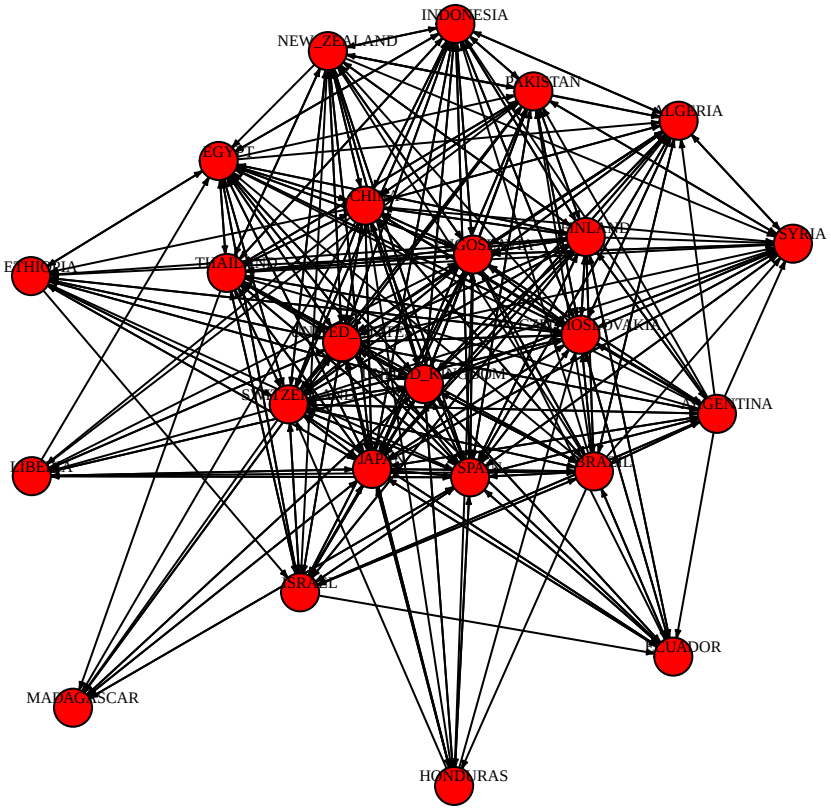
Now, let's plot these networks, at the same time exploring the various capabilities of the *igraph* package:

```
par(mar=c(0,0,0,0)) # you should know what this means
# And the graphs themselves:
plot(tradegraph.any,
    vertex.size = 8,
```

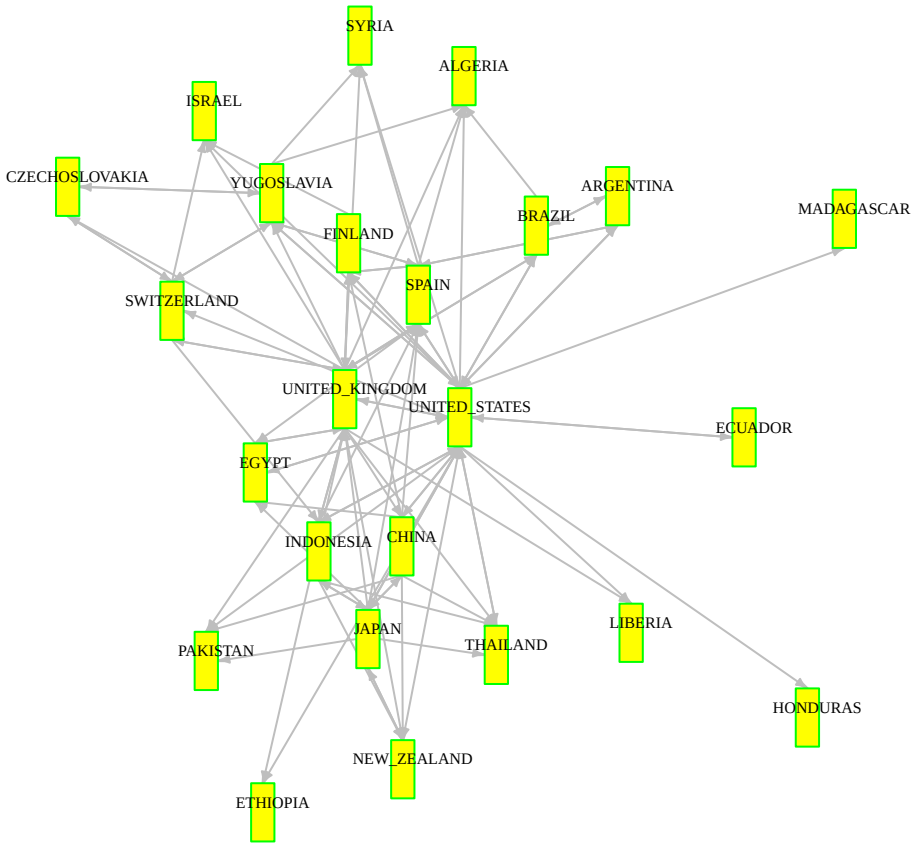
```
    edge.arrow.size = .2,
    vertex.label.cex = .5,
    vertex.color = 'aquamarine4',
    edge.color='red',
    vertex.shape = 'square',
    vertex.label.dist = .5,
    vertex.label.color = 'black')
```




```
plot(tradegraph.2,
    vertex.size = 10,
    edge.arrow.size = .2,
    vertex.label.cex = .5,
    vertex.color = 'red',
    edge.color='black',
    vertex.shape = 'circle',
    vertex.label.dist = .5,
    vertex.label.color = 'black')
```




```
plot(tradegraph.5,
    vertex.size = 6,
    edge.arrow.size = .3,
    edge.color='gray',
    vertex.label.cex = .5,
    vertex.color = 'yellow',
    vertex.shape = 'crectangle',
    vertex.frame.color = 'green',
    vertex.label.dist = .5,
    vertex.label.color = 'black')
```



Assignment questions. Irrespective of all the color/shape variations that are hurting your eyes (but at the same time show you the capabilities of the package), please answer the following questions:

1. What differences do you observe between the graphs where the cutpoint is any tie, at least two ties, and all ties present?
2. What information can you gather from these observed differences to help you expand on your earlier theoretical justification of ties? Alternatively, does your theoretical justification seem reasonable in light of new information obtained from these graphs?

Directed to undirected network

Another theoretical choice that we might often face is changing directed network to undirected network. While we lose some information, we build a simpler model that only shows presence or absense of ties. As you know, undirected network is a symmetric network.

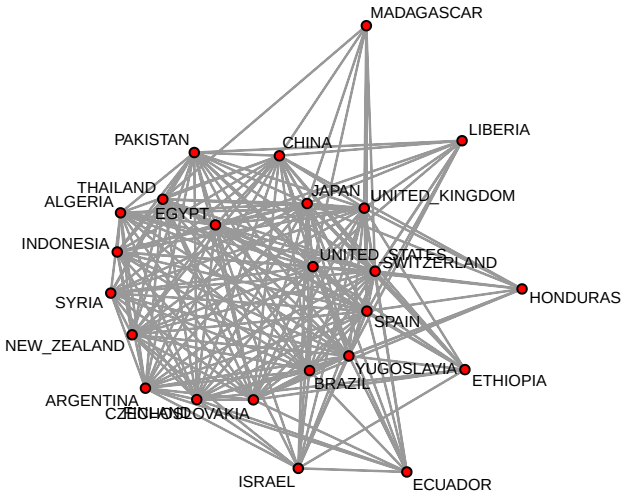
Basic network characteristics

Let's work with the network dichotomized at 2 or more ties present.

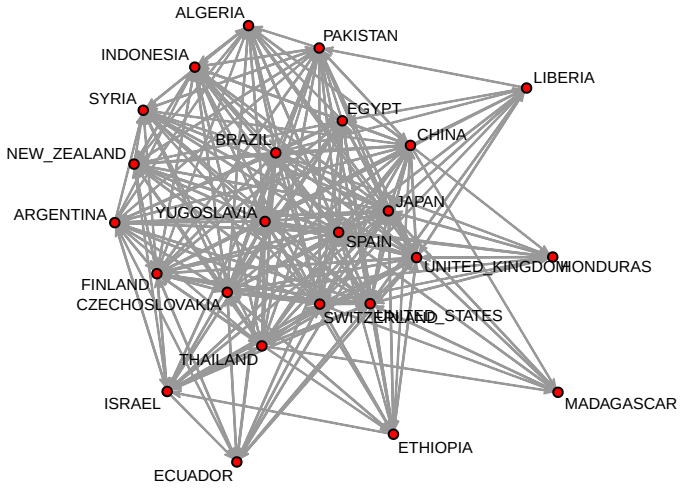
```
library(network) #put the package back

# Create an undirected (symmetrical) network:
```

```
tradenet.sym.2<- network(trade.2, directed=FALSE)
plot(tradenet.sym.2,
displaylabels=TRUE,
label.cex =.5,
edge.col = 'gray60')
```




```
#Now create an undirected (symmetrical) network:
tradenet2 <- network(trade.2)
plot(tradenet2,
displaylabels=TRUE,
label.cex =.5,
edge.col = 'gray60')
```




```
# Let's calculate some additional network stats:
# Density:
tradenet.any<-as.network(trade.any) # we did not turn this matrix into a network before
network.density(tradenet.any)
```

```
## [1] 0.8097826
```

```
network.density(tradenet.sym.2)
```

```
## [1] 0.7536232
```

```
network.density(tradenet2)
```

```
## [1] 0.6666667
```

```
graph.density(tradegraph.5)
```

```
## [1] 0.1811594
```

```
# Diameter:
diameter(tradegraph.any)
```

```
## [1] 2
```

```
diameter(tradegraph.2)
```

```
## [1] 2
```

```
diameter(tradegraph.5)
```

```
## [1] 3
```

Assignment question. Of course, there are differences between directed and undirected networks on the graph and with stats. Please answer the following questions:

1. What are the differences in graphs and how would you interpret them?
2. What are the differences in centrality?
3. what is the diameter and how do you expect it to wary?

Components

What are the components of the trade networks? Let's examine them by running the respective commands:

```
detach(package:igraph) # remove the igraph package
library(sna) # put sna back for analysis
tradenet5 <- network(trade.max) #turn it to a network also
components(tradenet.any)
```

```
## Node 1, Reach 24, Total 24
## Node 2, Reach 24, Total 48
## Node 3, Reach 24, Total 72
## Node 4, Reach 24, Total 96
## Node 5, Reach 24, Total 120
## Node 6, Reach 24, Total 144
## Node 7, Reach 24, Total 168
## Node 8, Reach 24, Total 192
## Node 9, Reach 24, Total 216
## Node 10, Reach 24, Total 240
## Node 11, Reach 24, Total 264
## Node 12, Reach 24, Total 288
## Node 13, Reach 24, Total 312
## Node 14, Reach 24, Total 336
## Node 15, Reach 24, Total 360
## Node 16, Reach 24, Total 384
## Node 17, Reach 24, Total 408
## Node 18, Reach 24, Total 432
## Node 19, Reach 24, Total 456
## Node 20, Reach 24, Total 480
## Node 21, Reach 24, Total 504
## Node 22, Reach 24, Total 528
## Node 23, Reach 24, Total 552
## Node 24, Reach 24, Total 576
```

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

```
components(tradenet2)
```

```
## Node 1, Reach 24, Total 24
## Node 2, Reach 24, Total 48
## Node 3, Reach 24, Total 72
## Node 4, Reach 24, Total 96
## Node 5, Reach 24, Total 120
## Node 6, Reach 24, Total 144
## Node 7, Reach 24, Total 168
## Node 8, Reach 24, Total 192
```

```
## Node 9, Reach 24, Total 216
## Node 10, Reach 24, Total 240
## Node 11, Reach 24, Total 264
## Node 12, Reach 24, Total 288
## Node 13, Reach 24, Total 312
## Node 14, Reach 24, Total 336
## Node 15, Reach 24, Total 360
## Node 16, Reach 24, Total 384
## Node 17, Reach 24, Total 408
## Node 18, Reach 24, Total 432
## Node 19, Reach 24, Total 456
## Node 20, Reach 24, Total 480
## Node 21, Reach 24, Total 504
## Node 22, Reach 24, Total 528
## Node 23, Reach 24, Total 552
## Node 24, Reach 24, Total 576
```

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

```
components(tradenet5)
```

```
## Node 1, Reach 1, Total 1
## Node 2, Reach 24, Total 25
## Node 3, Reach 24, Total 49
## Node 4, Reach 24, Total 73
## Node 5, Reach 24, Total 97
## Node 6, Reach 24, Total 121
## Node 7, Reach 24, Total 145
## Node 8, Reach 1, Total 146
## Node 9, Reach 24, Total 170
## Node 10, Reach 1, Total 171
## Node 11, Reach 24, Total 195
## Node 12, Reach 1, Total 196
## Node 13, Reach 24, Total 220
## Node 14, Reach 1, Total 221
## Node 15, Reach 1, Total 222
## Node 16, Reach 24, Total 246
## Node 17, Reach 1, Total 247
## Node 18, Reach 24, Total 271
## Node 19, Reach 24, Total 295
## Node 20, Reach 1, Total 296
## Node 21, Reach 24, Total 320
## Node 22, Reach 24, Total 344
## Node 23, Reach 24, Total 368
## Node 24, Reach 24, Total 392
```

```
## [1] 9
```

Assignment question. What are the differences between the three networks? How would you explain them from the theoretical level?

Geodesic distances

We have seen how to calculate the measure of geodesic distance in Seminar 3, but it will be much more fun

with this data. Again, remember, geodesics assume that the network is binary.

Let's see what happens if we calculate geodesics on directed and undirected network, which we generated from the same matrices:

```
geo.dist <-geodist(tradenet.sym.2) #for symmetric network
geo.dist.dir <-geodist(tradenet2) # for directed network
summary(geo.dist.dir)
```

```
##        Length Class  Mode
## counts 576    -none- numeric
## gdist  576    -none- numeric
```

```
summary(geo.dist)
```

```
##        Length Class  Mode
## counts 576    -none- numeric
## gdist  576    -none- numeric
```

```
# See? Geodesic calculations, at least done this way, completely ignore the arc direction.
```

Centrality

There are many centrality measures, as we have talked in lecture. We will learn them in chunks, so that we are not overwhelmed by the sheer number of them. Remember, it's very important to be able to interpet the results, not just run the code.

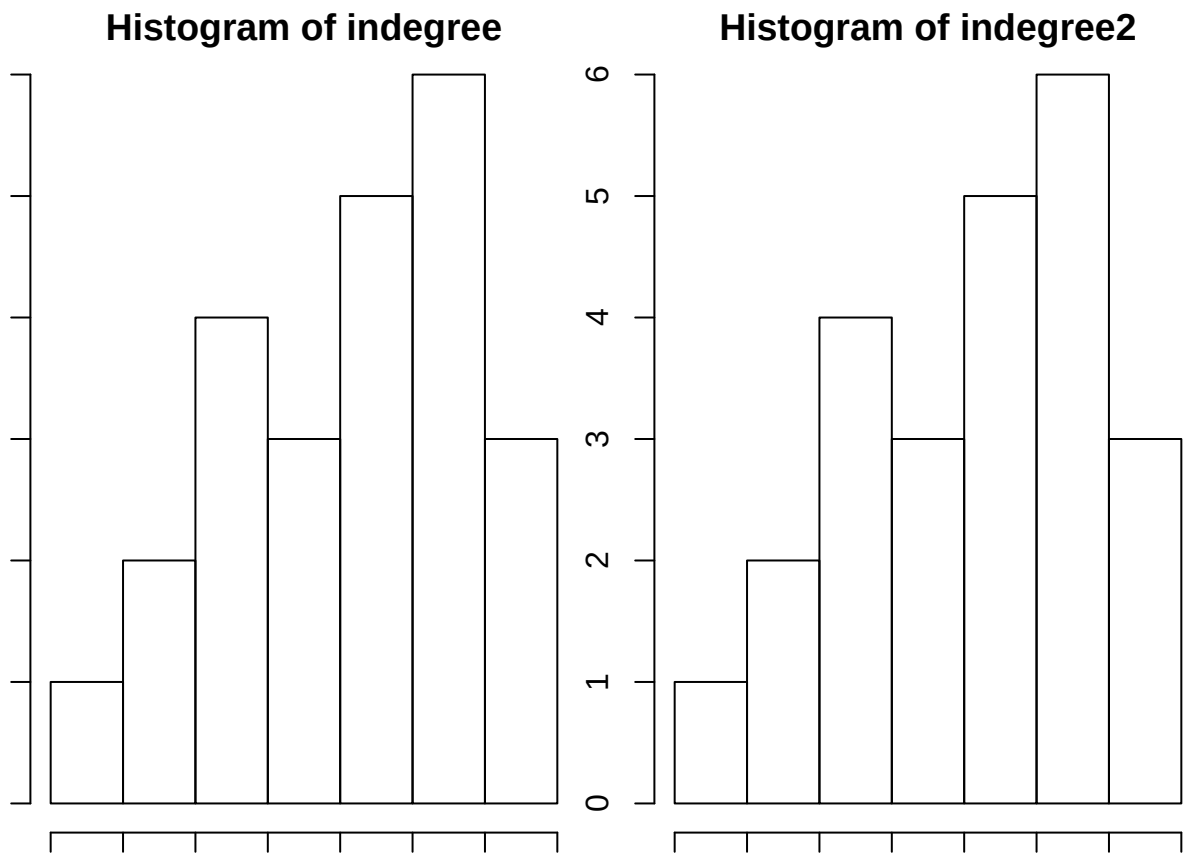
So, a little at a time.

```
# First let's work with directed and valued data
tradenet.valued <- as.network(trade.all,directed=TRUE) #create network
trade.att.valued <- trade.att #load attributes

# Degree centrality:
indegree <- degree(tradenet.valued,
  gmode = 'digraph', #uncomment the mode depending on type of data
  #gmode = 'graph', # mode can be graph (for undirected) or digraph
  diag = FALSE,
  cmode = 'indegree',
  rescale = FALSE,
  ignore.eval = FALSE)
indegree2 <- degree(tradenet.valued,
  gmode = 'digraph',
  diag = FALSE,
  cmode = 'indegree',
  rescale = FALSE,
  ignore.eval = TRUE)
```

Let's create a histogram to see how degrees are distributed:

```
par(mar=c(1,1,1,1),mfrow=c(1,2))
hist(indegree)
hist(indegree2)
```

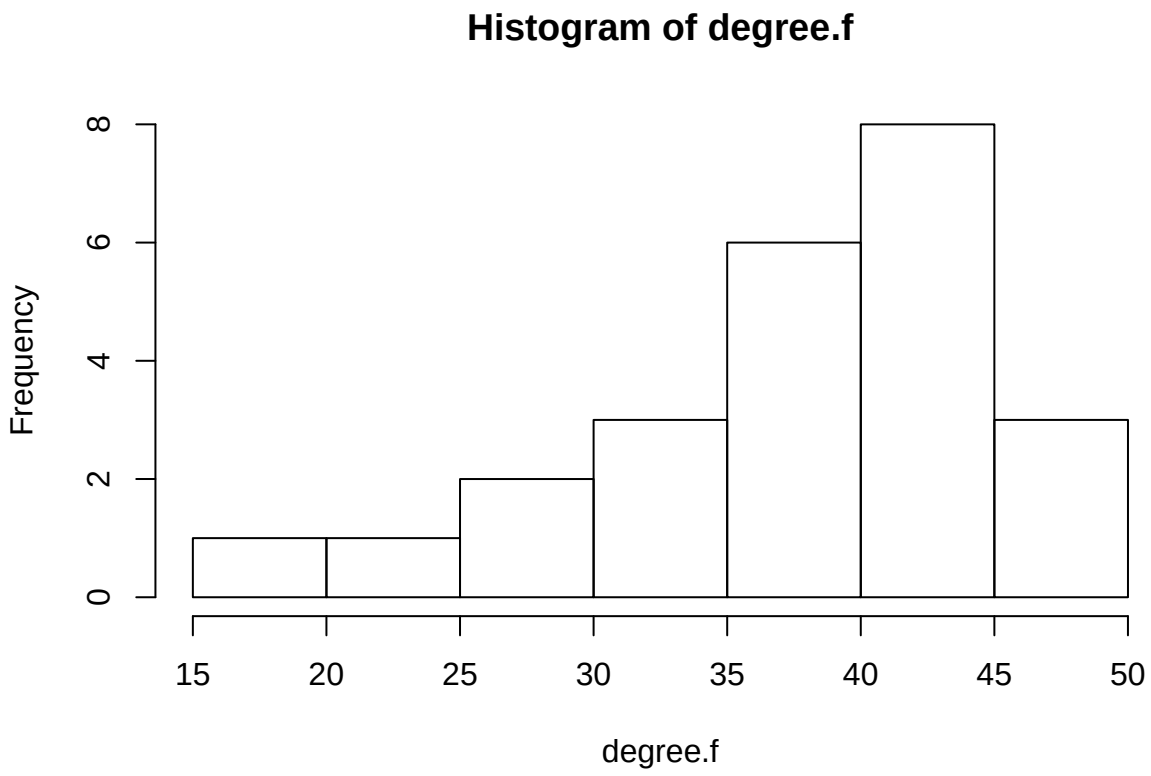


Assignment question. Is there a difference between valued data and non-valued data for degree centrality? Why?

OK, let's calculate outdegree and Freedman's degree (total degree):

```
outdegree <- degree(tradenet.valued,
  gmode = 'digraph',
  diag = FALSE,
  cmode = 'outdegree',
  rescale = FALSE)

# Freeman's degree (in + out):
degree.f <- degree(tradenet.valued,
  gmode = 'digraph',
  diag = FALSE,
  cmode = 'freeman',
  rescale = FALSE)
hist(degree.f)
```



Other types of centrality:

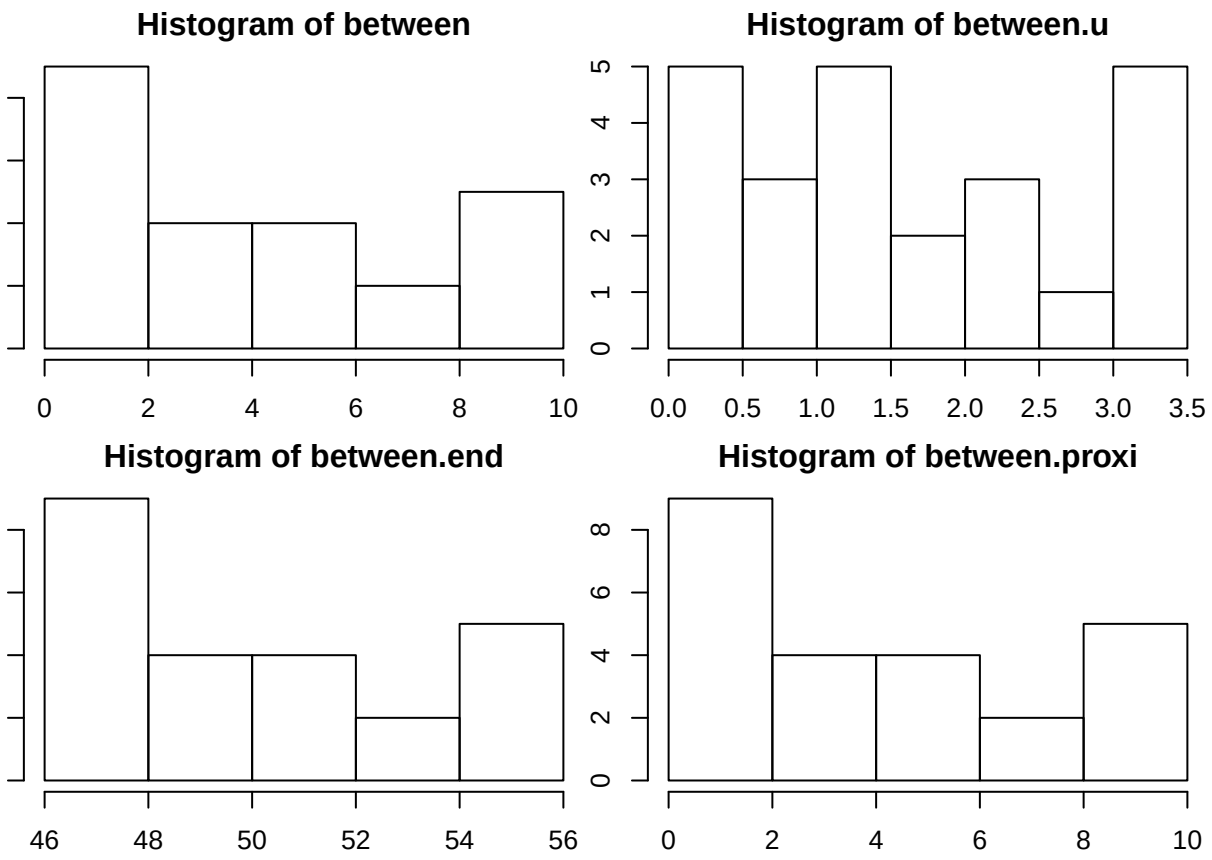
```
# Betweenness
between <- betweenness(tradenet.valued,
  gmode = 'digraph',
  diag = FALSE,
  cmode = 'directed')

# Let's experiment with cmode and plot histograms:

between.u <- betweenness(tradenet.valued,
  gmode = 'digraph',
  diag = FALSE,
  cmode = 'undirected')
between.end <- betweenness(tradenet.valued,
  gmode = 'digraph',
  diag = FALSE,
  cmode = 'endpoints')
between.proxi <- betweenness(tradenet.valued,
  gmode = 'digraph',
  diag = FALSE,
  cmode = 'proximalsrc')

par(mar=c(2,1,2,1),mfrow=c(2,2)) # Create a 2x2 matrix of plots
hist(between)
hist(between.u)
hist(between.end)
```

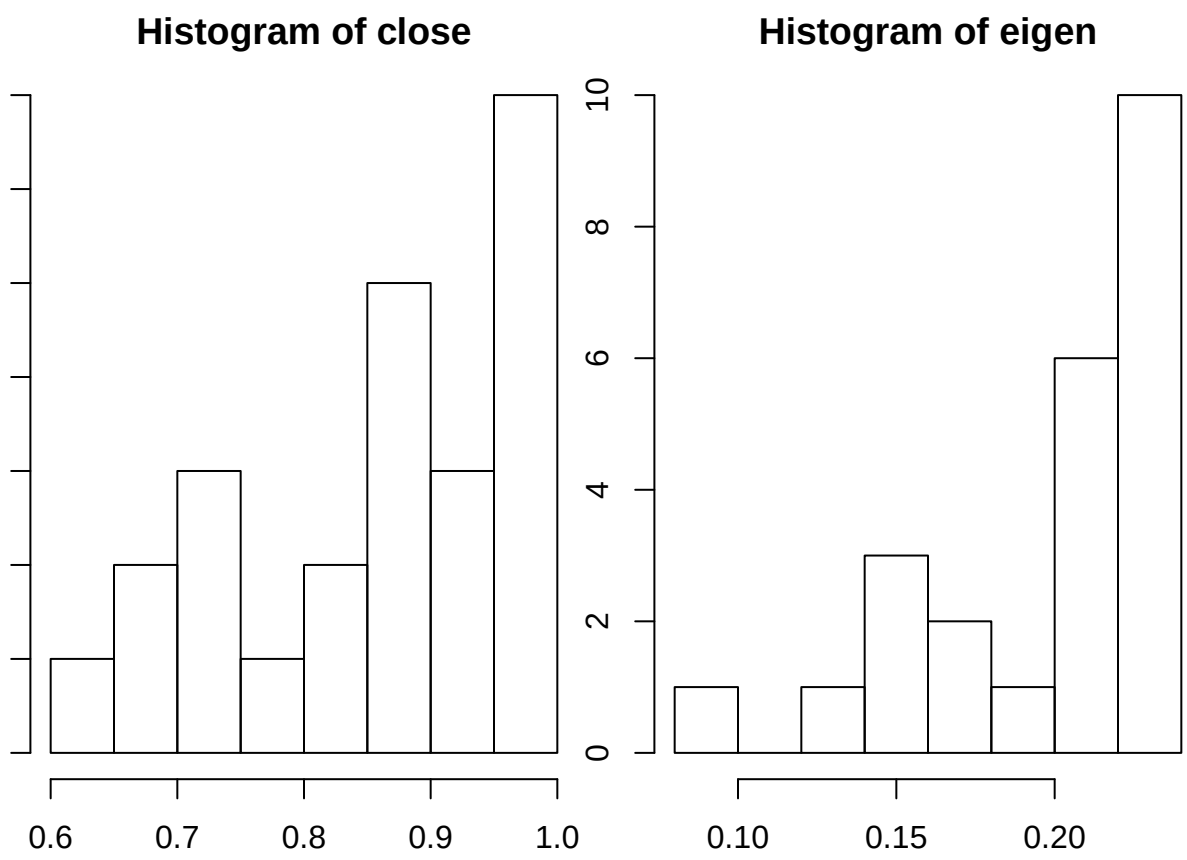
```r
hist(between.proxi)
```




```r
# Closeness
close <- closeness(tradenet.valued,
  gmode = 'digraph',
  # gmode ='graph',
  diag = FALSE,
  cmode = 'directed',
  rescale = FALSE)

#Eigenvalue
  eigen <- evcent(tradenet.valued,
  gmode = 'digraph',
  diag = FALSE,
  rescale = FALSE)

par(mar=c(2,1,2,1),mfrow=c(1,2))
hist(close)
hist(eigen)
```



Assignment question. Why do some of centrality histograms look the same while others look so different? What do they each show us?

Now, let's add node-level statistics to attributes data frame:

```
trade.att.valued <- cbind(trade.att.valued, indegree,  outdegree, degree.f, between, close, eigen)

class(trade.att.valued)
```

```
## [1] "data.frame"
```

```
names(trade.att.valued)
```

```
##  [1] "POP_GROWTH" "GNP"        "SCHOOLS"    "ENERGY"     "indegree"
##  [6] "outdegree"  "degree.f"   "between"    "close"      "eigen"
```

Centrality at the graph level

Let's look at one more command, "centralization."

```
centralization(tradenet.valued, FUN ='degree',
  normalize = TRUE)
```

```
## [1] 0.2075099
```

```
# Some other options:
centralization(tradenet.valued, FUN ='betweenness',
  normalize = TRUE)
```

```
## [1] 0.0113248
```

```
centralization(tradenet.valued, FUN ='closeness',
  normalize = TRUE)
```

```
## [1] 0.1560584
```

```
centralization(tradenet.valued, FUN ='evcent',
  normalize = TRUE)
```

```
## [1] 0.03954187
```

Assignment question. What do indexes above mean? What do they tell us about our network?

Final assignment task. There are several networks in the "trade.Rdata" file, described above. We have fully explored the "trade.all" network. Now, select *one* of the individual trade networks (manufacture, food, crude, etc.) and show me everything you've learned in this class so far. At the very minimum, please do the following:

1. Create an appropriate graph with all possible options.
2. Generate all possible network measures.
3. Tell me what inferences you can make about your selected network based on the information you've obtained. Supplement your arguments with logic and theory.
