Applied Network Science with R

George G. Vega Yon

2022-06-23

Contents

Chapter 1

About

Placeholder

1.0.1 The Book

1.0.2 The author

Part I

Applications

Chapter 2

Introduction

Social Network Analysis and Network Science, have a long scholarly tradition. From social diffusion models to protein-interaction networks, these complex-systems disciplines cover a wide range of problems across scientific fields. Yet, although these could be seen as wildly different, the object under the microscope is the same, networks.

With a long history (and insufficient levels of inter-discipline collaboration, if you allow me to say) of scientific advances happing in a somewhat isolated fashion, the potential of cross-pollination between disciplines within network science is immense.

This book is an attempt to compile the many methods available in the realm of complexity sciences, provide an in-depth mathematical examination–when possible–, and provide a few examples illustrating their usage.

Chapter 3

R Basics

Placeholder

3.1 What is R

3.2 How to install packages

3.3 Prerequisits

3.4 A ~~gentle~~ Quick n' Dirty Introduction to R

Chapter 4

Network Nomination Data

Placeholder

4.1 Data preprocessing

4.1.1 Reading the data into R

4.1.2 Creating a unique id for each participant

4.2 Creating a network

4.2.1 From survey to edgelist

4.2.2 igraph network

4.3 Network descriptive stats

4.4 Plotting the network in igraph

4.4.1 Single plot

4.4.2 Multiple plots

4.5 Statistical tests

4.5.1 Is nomination number correlated with indegree?

Chapter 5

Exponential Random Graph Models

Placeholder

5.1 A naïve example

5.2 Estimation of ERGMs

5.3 The `ergm` package

5.4 Running ERGMs

5.5 Model Goodness-of-Fit

5.6 More on MCMC convergence

5.7 Mathematical Interpretation

5.8 Markov independence

Chapter 6

Using constraints in ERGMs

Placeholder

6.1 Example 1: Interlocking egos and disconnected alters

6.2 Example 2: Bi-partite networks

Chapter 7

(Separable) Temporal Exponential Family Random Graph Models

This tutorial is great! `https://statnet.org/trac/raw-attachment/wiki/Sunbelt2016/tergm_tutorial.pdf`

Chapter 8

Simulating and visualizing networks

In this chapter, we will build and visualize artificial networks using Exponential Random Graph Models [ERGMs.] Together with chapter 3, this will be an extended example of how to read network data and visualize it using some of the available R packages out there.

For this chapter we will be using the following R packages: `ergm`, `sna`, `igraph`, `intergraph`, `netplot`, `netdiffuseR`, and `rgexf`.

8.1 Random Graph Models

While there are tons of social network data, we will use an artificial one for this chapter. We do this as it is always helpful to have more examples simulating Random networks. For this chapter, we will classify random graph models for sampling and generating networks into three categories:

1. **Exogenous**: Graphs where the structure is determined by a macro rule, e.g., expected density, degree distribution, or degree-sequence. In these cases, ties are assigned to comply with a macro-property.
2. **Endogenous**: This category includes all Random Graphs generated based on endogenous information, e.g., small-world, scale-free, etc. Here, a tie creation rule gives origin to a macro property, for example, preferential attachment in scale-free networks.
3. **Exponential Random Graph Models**: Overall, since ERGMs compose a family of statistical models, we can always (or almost always) find a model specification that matches the previous categories. Whereas we are thinking about degree sequence, preferential attachment, or a mix of both, ERGMs can be the baseline for any of those models.

The later, ERGMs, are a generalization that covers all classes. Because of that, we will use ERGMs to generate our artificial network.

8.2 Social Networks in Schools

A common type of network we analyze is friendship networks. In this case, we will use ERGMs to simulate friendship networks within a school. In our simulated world, these networks will be dominated by the following phenomena

- Low density,
- Race homophily,
- Structural balance,
- And age homophily.

If you have been paying attention to the previous chapters, you will notice that, out of these five properties, only one constitutes Markov graphs. Within a tie, homophily and density only depend on ego and alter. In race homophily, only ego and alter's race matter for the tie formation, but, in the case of Structural balance, ego is more likely to befriend alter if a fried of ego is friends with alter, i.e., "the friend of my friend is my friend."

The simulation steps are as follows:

1. Draw a population of *n* students and randomly distribute race and age across them.
2. Create a `network` object.
3. Simulate the ties in the empty network.

Here is the code

```r
set.seed(712)
n <- 200

# Step 1: Students
race   <- sample(c("white", "non-white"), n, replace = TRUE)
age    <- sample(c(10, 14, 17), n, replace = TRUE)

# Step 2: Create an empty network
library(ergm)
```

```
## Loading required package: network

##
## 'network' 1.17.2 (2022-05-20), part of the Statnet Project
## * 'news(package="network")' for changes since last version
## * 'citation("network")' for citation information
## * 'https://statnet.org' for help, support, and other information

##
## 'ergm' 4.2.1 (2022-05-10), part of the Statnet Project
## * 'news(package="ergm")' for changes since last version
## * 'citation("ergm")' for citation information
## * 'https://statnet.org' for help, support, and other information
## 'ergm' 4 is a major update that introduces some backwards-incompatible
## changes. Please type 'news(package="ergm")' for a list of major
## changes.
```

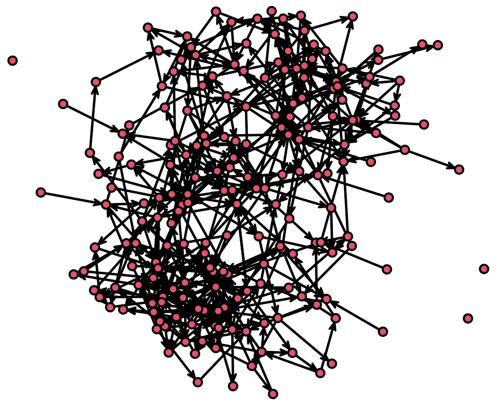
```r
library(network)
net <- network.initialize(n)

net %v% "race"   <- race
net %v% "age"    <- age

# Step 3: Simulate a graph
net_sim <- simulate(
    net ~ edges +
    nodematch("race") +
    triangle +
    absdiff("age"),
    coef = c(-4, .5, .5, -.5)
    )
```

What just happened? Here is a line-by-line breakout:

1. `set.seed(712)` Since this is a random simulation, we need to fix a seed so it is reproducible. Otherwise, results would change with every iteration.
2. `n <- 200` We are assigning the value 200 to the object n. This will make things easier as, if needed, changing the size of the networks can be done at the top of the code.
3. `race <- sample(c("white", "non-white"), n, replace = TRUE)` We are sampling 200, or actually, n values from the vector `c("white", "non-white")` with replacement.
4. `age <- sample(c(10, 14, 17), n, replace = TRUE)` Same as before, but with ages!
5. `library(ergm)` Loading the ergm R package, which we need to simulate the networks!
6. `library(network)` Loading the `network` R package, which we need to create the empty graph.
7. `net <- network.initialize(n)` Creating an empty graph of size n.
8. `net %v% "race" <- race` Using the `%v%` operator, we can access vertices features in the network object. Since race does not exist in the network yet, the operator just creates it. Notice that the number of vertices matches the length of the race vector.
9. `net %v% "age" <- age` Same as with race!
10. `net_sim <- simulate(` Simulating an ERGM!

Let's take a quick look at the resulting graph

```r
library(sna)
gplot(net_sim)
```



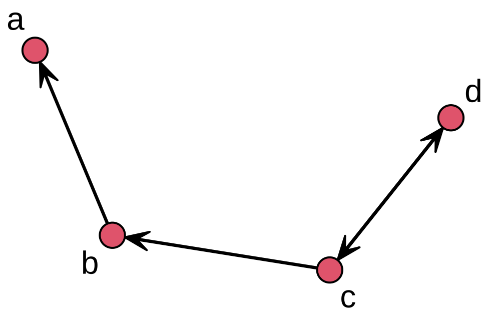
We can now start to see whether we got what we wanted! Before that, let's save the network as a plain-text file so we can practice reading networks back in R!

```r
write.csv(as.edgelist(net_sim), file = "06-edgelist.csv")
write.csv(as.data.frame(net_sim, unit = "vertices"), file = "06-nodes.csv")
```

8.3 Reading a network

The first step to analyzing network data is to read it in. Many times you'll find data in the form of an adjacency matrix. Other times, data will come in the form of an edgelist. Another common format is the adjacency list, which is a compressed version of an edgelist. Let's see how the formats look like for the following network:

```r
example_graph <- matrix(0L, 4, 4, dimnames = list(letters[1:4], letters[1:4]))
example_graph[c(2, 7)] <- 1L
example_graph["c", "d"] <- 1L
example_graph["d", "c"] <- 1L
example_graph <- as.network(example_graph)
set.seed(1231)
gplot(example_graph, label = letters[1:4])
```



- **Adjacency matrix** a matrix of size *n* by *n* where the *ij*-th entry represents the tie between *i* and *j*. In a directed network, we say *i* connects to *j*, so the *i*-th row shows the ties *i* sends to the rest of the network. Likewise, in a directed graph, the *j*-th column shows the ties sent to *j*. For undirected graphs, the adjacency matrix is usually upper or lower diagonal.

The adjacency matrix of an undirected graph is symmetric, so we don't need to report the same information twice. For example:

```
as.matrix(example_graph)
```

```
##   a b c d
## a 0 0 0 0
## b 1 0 0 0
## c 0 1 0 1
## d 0 0 1 0
```

- **Edge list** a matrix of size $|E|$ by 2, where $|E|$ is the number of edges. Each entry represents a tie in the graph.

```
as.edgelist(example_graph)
```

```
##      [,1] [,2]
## [1,]    2    1
## [2,]    3    2
## [3,]    3    4
## [4,]    4    3
## attr(,"n")
## [1] 4
## attr(,"vnames")
## [1] "a" "b" "c" "d"
## attr(,"directed")
## [1] TRUE
## attr(,"bipartite")
## [1] FALSE
## attr(,"loops")
## [1] FALSE
## attr(,"class")
## [1] "matrix_edgelist" "edgelist"        "matrix"          "array"
```

- **Adjacency list** a list of vectors of at most size n.

```
igraph::as_adj_list(intergraph::asIgraph(example_graph)) |>
  lapply(\(x) as.vector(x))
```

```
## [[1]]
## [1] 2
##
## [[2]]
## [1] 1 3
##
## [[3]]
## [1] 2 4 4
```

```
##
## [[4]]
## [1] 3 3
```

Here we will deal with an edgelist that includes node information. In my opinion, this is one of the best ways to share network data.

```r
edges <- read.csv("06-edgelist.csv")
nodes <- read.csv("06-nodes.csv")
```

Using igraph

Using `network`

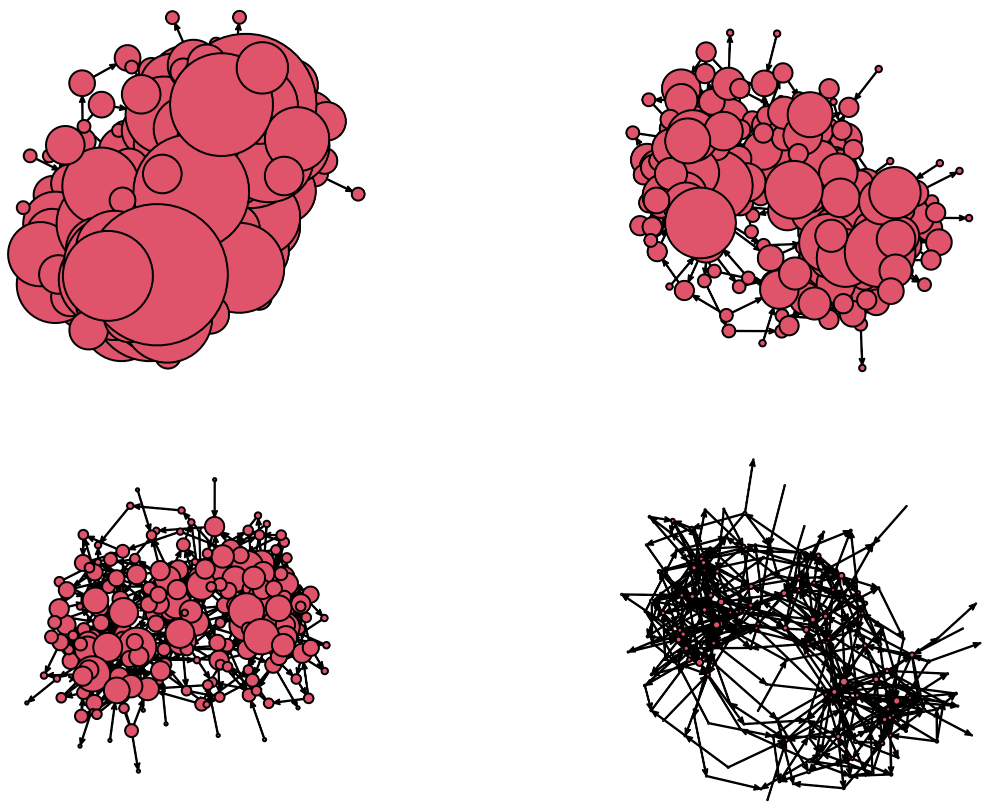
8.4 Visualizing the network

We will focus on three different attributes that we can use for this visualization: Node size, node shape, and node color. While there are no particular rules, some ideas you can follow are:

- **Node size** Use it to describe a continuous measurement. This feature is often used to highlight important nodes, e.g., using one of the many available degree measurements.
- **Node shape** Shapes can be used to represent categorical values. A good figure will not feature too many of them; less than four would make sense.
- **Node color** Like shapes, colors can be used to represent categorical values, so the same idea applies. Furthermore, it is not crazy to use both shape and color to represent the same feature.

Notice that we have not talked about layout algorithms. The R packages to build graphs usually have internal rules to decide what algorithm to use. We will discuss that later on. Let's start by size. Finding the right scale can be somewhat difficult. We will draw the graph four times to see what size would be the best:

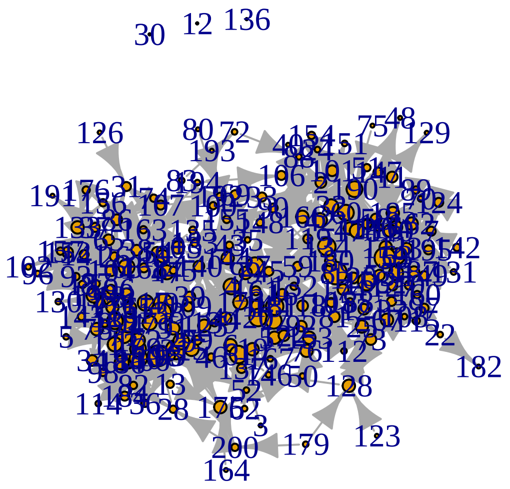
```r
# Sized by indegree
net_sim %v% "indeg" <- degree(net_sim, gmode = "digraph")

# Preparing the graphical device to hold four nets.
# This line sets a 2 x 2 viz device and stores the
# original value. We will use the `op` object to reset
# the configugarion
op <- par(mfrow = c(2, 2), mai = c(.1, .1, .1, .1))
gplot(net_sim, vertex.cex = (net_sim %v% "indeg") * 2)
gplot(net_sim, vertex.cex = net_sim %v% "indeg")
gplot(net_sim, vertex.cex = (net_sim %v% "indeg")/2)
gplot(net_sim, vertex.cex = (net_sim %v% "indeg")/10)
```



```r
par(op)
```

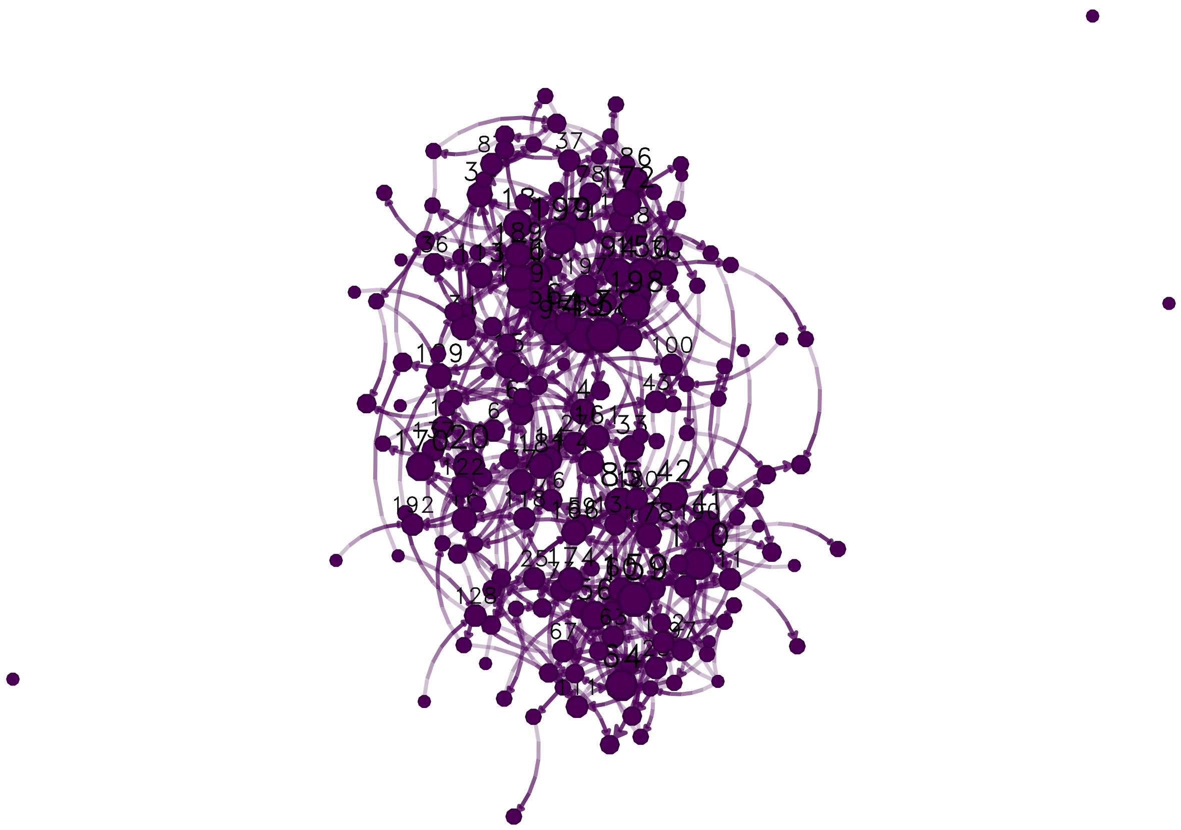
If we were using igraph, setting the size can be easier thanks to the netdiffuseR R package. Let’s start by converting our network to an igraph object with the R package `intergraph`

```r
library(intergraph)
library(igraph)

# Converting the network object to an igraph object
net_sim_i <- asIgraph(net_sim)

# Plotting with igraph
plot(
  net_sim_i,
  vertex.size = netdiffuseR::rescale_vertex_igraph(
    vertex.size = V(net_sim_i)$indeg,
    minmax.relative.size = c(.01, .1)
  )
)
```



We could also have tried netplot, which should make things easier (and prettier):

```r
library(netplot)
nplot(net_sim)
```



Chapter 9

Hypothesis testing in networks

Placeholder

9.1 Comparing networks

9.1.1 Network bootstrap

9.1.2 When the statistic is normal

9.1.3 When the statistic is NOT normal

9.2 Examples

9.2.1 Average of node-level stats

Chapter 10

Network diffussion with `netdiffuseR`

Placeholder

10.1 Network diffusion of innovation

10.1.1 Diffusion networks

10.1.2 Thresholds

10.2 The netdiffuseR R package

10.2.1 Overview

10.2.2 Datasets

10.2.3 Visualization methods

10.2.4 Problems

10.3 Simulation of diffusion processes

10.3.1 Simulating diffusion networks

10.3.2 Rumor spreading

10.3.3 Difussion

10.3.4 Mentor Matching

10.3.5 Example by changing threshold

10.3.6 Problems

10.4 Statistical inference

10.4.1 Moran's I

10.4.2 Using geodesics

10.4.3 Structural dependence and permutation tests

10.4.4 Idea

10.4.4.1 Example Not random TOA

10.4.5 Regression analysis

10.4.5.1 Lagged exposure models

10.4.5.2 Contemporaneous exposure models

10.4.6 Problems

Chapter 11

Stochastic Actor Oriented Models

Stochastic Actor Oriented Models (SOAM), also known as Siena models were introduced by CITATION NEEDED.

As a difference from ERGMs, Siena models look at the data generating process from the individuals' point of view. Based on McFadden's ideas of probabilistic choice, the model is founded in the following equation

$$U_i(x) - U_i(x') \sim \text{Extreame Value Distribution}$$

In other words, individuals choose between states x and x' in a probabilistic way (with some noise),

$$\frac{\exp f_i^Z(\beta^z, x, z)}{\sum_{Z' \in \mathcal{C}} \exp f_i^Z(\beta, x, z')}$$

snijders_(sociological methodology 2001)

(Snijders, Bunt, and Steglich 2010, @lazega2015, @Ripley2011)

Part II

Statistical Foundations

11.1 Bayes' Rule

11.2 Markov Chain

11.2.1 Metropolis Algorithm

11.2.2 Metropolis-Hastings

11.2.3 Likelihood-free MCMC

Appendix A

Datasets

Placeholder

A.1 SNS data

A.1.1 About the data

A.1.2 Variables

References

Lazega, Emmanuel, and Tom AB Snijders. 2015. *Multilevel Network Analysis for the Social Sciences: Theory, Methods and Applications*. Vol. 12. Springer.

Ripley, Ruth M., Tom AB Snijders, Paulina Preciado, and Others. 2011. “Manual for RSIENA.” *University of Oxford: Department of Statistics, Nuffield College*, no. 2007. https://www.uni-due.de/hummell/sna/R/RSiena{_}Manual.pdf.

Snijders, Tom A B, Gerhard G. van de Bunt, and Christian E G Steglich. 2010. “Introduction to stochastic actor-based models for network dynamics.” *Social Networks* 32 (1): 44–60. https://doi.org/10.1016/j.socnet.2009.02.004.