Supplemental Material

CBE—Life Sciences Education

Grunspan *et al.*

*Note**:** In the interest of protecting student privacy rights, and due to IRB agreements, we cannot release the exact data used in this tutorial. However, we can provide similarly formatted data which can be used along with all of the following code. This means that results will vary regarding significance, but it will allow readers to follow along step by step. If you would like data to accompany the code, please e-mail the corresponding author (grunny@u.washington.edu).*

R Script

In this tutorial, we provide the syntax for the social network analyses, without explaining the full details of all of the syntax for the sake of brevity. If unfamiliar with R, some aspects of the syntax should be deducible from these examples; beyond that, we recommend familiarizing yourself with any number of useful tutorials or other resources. Popular print works include *The Art of R Programming* (Matloff and Matloff 2011) and *Discovering Statistics Using R (Unwin 2013)*, while numerous user-contributed online resources can be found on the R webpage at http://cran.r-project.org/other-docs.html and across the web.

To get started, first download and install R:

1) Go to http://cran.r-project.org/.
2) Go to “Mirrors” on the left side panel.
3) Select a location close to you.
4) Select the proper version for your operating system.
5) Follow the instructions given from there.

We will be using the *statnet* suite of *R* packages throughout this tutorial, so we must obtain it as well:

1) Open R.
2) At the R cursor >, type:

```
> install.packages("statnet")
> library(statnet)
```

The first command installs the *statnet* packages onto your machine, while the second loads those packages into your current session so that you can use them. *Note: packages only need to be installed once; however, make sure to load them with the* library *command at the beginning of every R session in which you plan to use them. Also note the presence of quotes in the first command, but their absence in the second.*

The two main packages from the statnet suite that we will use in this tutorial are *network* and *sna*. *Network* handles all of the data manipulation and exploration, while *sna* provides tools for the classical analysis of networks. You can look at the overall documentation for any of them with the “**?**” command. For example:

```
> ?sna
```

Note: The **?***command is useful whenever you'd like more information about functions or packages (e.g. ?library, ?sum, ?sna, ?plot, etc.)*

Now that we have our workspace ready, we can load data into it. First, we'll set our working directory to the location where our data are stored:

```
> setwd("C:/StudentLearning/data")
```

There are different functions to load data depending on the format. We have our attribute data in a comma-separated value (.csv) file, so we will use the "**read.csv**" function.

```
> data <- read.csv("studentattributes.csv", as.is=TRUE,
col.names=c("ID", "major", "ethnicity", "gender", "lab",
"grade", "exam1", "exam2"), header=FALSE)
```

This creates a data frame named "**data**" with eleven different variables arranged in columns, each accessible using a "**$**" sign (such as "**data$major**").

Note: the symbol <-, comprising a "less-than" sign and a dash, represents assignment from right to left, much like the equals sign in many other programming languages; it is pronounced "gets". Also, every object in R has a class, which determines how it will be handled by the various functions available to the user. The first object that we have created, data, is of class "data.frame"

We consider the co-studying networks of students in our class at two time points. Our two networks are stored in sociomatrix form as data (.dat) files, which we can read in using the "**read.table**" function. This will create two matrix objects (ie. with class "matrix") named "**study1**"and "**study2**".

```
> study1.matrix <- read.table("studynet1.dat", header=TRUE)
> study2.matrix <- read.table("studynet2.dat", header=TRUE)
```

We can look at the dimensions to make sure we have square matrices:

```
> dim(study1.matrix)
[1] 187 187
> dim(study2.matrix)
[1] 187 187
```

Both networks are the same size of 187x187. We should also make sure our data frame contains data for 187 students, too:

```
> dim(data)
[1] 187  8
```

Exploratory data analysis:

In this section, we will visualize our data in a variety of ways and produce descriptive statistics. This will help us generate hypotheses about the structure that we can test more formally in the subsequent section.

We can transform the matrices containing our relational data into an object of class “network” using the “**as.network**” function from the network package. This will allow us to employ all of the functionality for network analysis found through the *statnet* suite. We want to treat our study networks as undirected, meaning a tie between two students is always considered reciprocal, so we set the “directed” argument to false. This symmetrizes the network.

```
> study1 <- as.network(study1.matrix, directed=FALSE)
> study2 <- as.network(study2.matrix, directed=FALSE)
```

Note: We would not symmetrize the data for a directed relationship. For example, if students were asked to nominate a ‘best explainer’ in their lab section, directionality would carry useful information and we would treat the network as directed.

We can visualize our networks with the “**plot**” command. First, we will create two panels for displaying graphs, with margins set for optimal viewing in *R* with the “**par**” function, and then graph our two networks

```
> par(mfrow=c(1,2), oma=c(0,0,0,0), mar=c(1,1,1,1))
> gplot(study1)
> gplot(study2)
```

The output can be seen in Supplemental Figure 1.

*****Supplemental FIGURE 1*****

Caption: Sociographs of our two study networks. Circles represent individual students and lines between students represent a study partnership between two students.

We can make the sociographs more informative if we represent nodal attributes through colors, shapes and sizes. Figure 1 in our article can be produced with the following code, which uses various arguments such as “vertex.col” and “vertex.cex” to change the color and shape of nodes according to the attributes we assign.

```
> par(mfrow=c(1,2), oma=c(0,0,0,0), mar=c(1,1,1,1))
> gplot(study1, vertex.col=as.factor(data$lab),
     vertex.cex=data$grade/3, vertex.sides = data$gender+2,
gmode=”graph”)
> gplot(study2, vertex.col=as.factor(data$lab),
     vertex.cex=data$grade/3, vertex.sides = data$gender+2,
gmode=”graph”)
```

To obtain the densities, transitivities and triad census of each network, we use the “gden”, “gtrans”, and “triad.census” functions.

```
> densities <- c(gden(study1), gden(study2))
> densities
[1] 0.008682652 0.010637686
```

*Our network of 187 nodes has 17,391 possible edges (187 choose 2, or 187*186/2); these densities then correspond to 151 and 185 edges, respectively. We could have gotten these latter numbers directly with the network.edgecount command.*

```
> triad.census(study1, mode="graph")
           0     1   2  3
[1,] 1044790 27407 216 32

> triad.census(study2, mode="graph")
           0     1   2  3
[1,] 1038672 33384 326 63

> transitivities <- c(gtrans(study1), gtrans(study2))
> transitivities
[1] 0.3076923 0.3669903
```

The outputs are network densities, triad censuses, and transitivities for exams one and two, respectively.

To display degree distributions we start by assigning the indegree of each node to a vector called study1.degree and display the results as a table. (We could also use outdegree because symmetrized networks have equal indegree and outdegree values).

```
> study1.degree <- degree(study1, cmode="indegree")
> table(study1.degree)
> study2.degree <- degree(study2, cmode="indegree")
> table(study2.degree)
```

To run permutation correlation tests we first create a function that will take in two variables of interest, permute random distributions, and provide a distribution of correlations.

```
> perm.cor.test<-function(x,y,niter=100000){
>  c.obs<-cor(x,y,use="complete.obs")
>  c.rep<-vector()
>  for(i in 1:niter)
>    c.rep[i]<-cor(x,sample(y),use="complete.obs")
>  cat("Vector Permutation Test:\n\tObserved correlation: 
",c.obs,"\tReplicate quantiles (niter=",niter,")\n",sep="")
>  cat("\t\tPr(rho>=obs):",mean(c.rep>=c.obs),"\n")
>  cat("\t\tPr(rho<=obs):",mean(c.rep<=c.obs),"\n")
>  cat("\t\tPr(|rho|>=|obs|):",
mean(abs(c.rep)>=abs(c.obs)),"\n")
>  invisible(list(obs=c.obs,rep=c.rep))
>  hist(c.rep,main="Permuted Correlations")
>  abline(v=c.obs, col="red")
>}
```

We can now create our degree and betweenness variables to correlate alongside exam scores, and then call perm.cor.test to test the correlations. This function will output both text and a histogram with results.

```
study1.degree<-degree(study1, cmode="indegree")
study2.degree<-degree(study2, cmode="indegree")

study1.bet <- betweenness(study1, gmode="graph")
study2.bet <- betweenness(study2, gmode="graph")

perm.cor.test(study1.degree, data$exam1)
perm.cor.test(study2.degree, data$exam2)

perm.cor.test(study1.bet, data$exam1)
perm.cor.test(study2.bet, data$exam2)
```