```r
#####
#A short tutorial on analyzing ego network data in R
#for the Social Networks and Health
#workshop at Duke University on May 17, 2016

#In this tutorial we learn some basic functionality
#for R with dealing with ego network
#

#We will assume that the data were collected in a traditional
#manner, with independently sampled respondents
#reporting on their network alters
#
#Much of the difficulty in analyzing ego network data in R
#is to transform the traditional survey into something that
#network packages in R, like igraph and sna, can work with.
#The problem is that the form of a survey is not the
#form of network data (matrices, edgelist, etc.)
#and each survey is different, different variables, structure, etc.

#We will get a bit of practice here using some example ego network data
#based on a school, with information on grade and race


#I. Getting the data ready

#first need to read in the data
setwd("~/classes/workshops/social_networks_health_duke/data/")
#here will use a setwd to set the working directory to where the
#ego network data, called example_egonetdata.Rdata is stored.
#this can be downloaded at:
#https://www.dropbox.com/s/1k21oyx5ip5cm87/example_egonetdata.Rdata?dl=0
#on the shared drop box page for this workshop

load(file="example_egonetdata.Rdata")
ls()
egonet.data[1:4,] #let's take a look at the first four rows
#here, each row is a respondent, reporting on their network ties
#actual.degree is the number of total people named
#listed.degree is the number of people they reported on, here capped at 5
#race, grade are characteristics of the respondent
#race1...race5 and grade1...grade5 are alter characteristics
#e12, e13...is the presence/absence of tie between alter 1-2, 1-3, etc.

#now we need to load a package, egonetR that will be useful in turning
#that data into something easier to work with, in terms of calculating
#our network measures of interest

library(egonetR)
library(igraph) #also load igraph, this will be useful

#function we want here is called read.egonet.one.file
#inputs:
#egos=dataset with ego network information
#netsize=a vector, designating the degree (or network size) of that alter
```

```
#egoID=variable names on dataset that identifies ego
#attr.start.col=column where attributes for alters begin
#attr.end.col=column where attributes for alters end
#dy.max.alter=maximum number of alters that we have information on
#often useful if they could name (for example) 10 alters
#but only gave information on a subset
#here we will keep things simple and only work with the listed
#alters, those they reported on for each characteristic

#dy.first.var=column where alter-alter tie information begins
#should be in the right order, 1-2, 1-3, 1-4, etc.
#ego.vars=columns, if desired, of ego variables to be included in output
#var.wise=T or F, T if goes race1 race2..grade1 grade2..
#instead of race1 grade1 race2 grade2...

egonetlist=read.egonet.one.file(egos=egonet.data,
netsize=egonet.data$listed.degree, egoID = "id"
, attr.start.col=6,attr.end.col=15,
  dy.max.alteri=5, dy.first.var=16, ego.vars = c(1:3), var.wise = T)

#in this case the inputs are:
#egonet.data as this is the data of interest
#netsize we set to the vector listed.degree, note it is the actual values,
rather
#than the name of the variable in the dataset
#egoID is set equal to "id" as this is the variable corresponding
#to the respondent id
#attr.start.col=6,attr.end.col=15 as these are the columns where the alter
#attributes start and stop
#dy.max.alteri=5 as they were allowed to describe up to 5 alters
#although could have named more
#dy.first.var=16 as this is where the e12... variables start
#the variables denoting ties between alters
#var.wise equals T in this case

#clearly these inputs would be different with a different dataset

#what do we get from this function?
egonetlist
names(egonetlist)


#let's first look at the list of alter characteristics
alteri.list=egonetlist$alteri.list
alteri.list

#note the ego information is repeated for each alter
#also have information on each alter
#this will be useful in calculating measures of interest

#now let's look at the edges between alters
egonetlist$edges

#check to make sure it is right!
```

```
egonetlist$edges[6]
egonet.data[6,]

#now, let's look at the network objects, based in igraph
#that were created

egonetlist$graphs

#let's plot one of the graphs
egonets=egonetlist$graphs

egonets[[6]]


#let's plot that network, note that ego is not included
plot(egonets[[6]])

#four together
par(mfrow=c(2,2))
plot(egonets[[6]])
plot(egonets[[7]])
plot(egonets[[9]])
plot(egonets[[11]])


#note that networks generated do not include ego
#thus we do not see ego in the plot

#note also that isolates are automatically excluded here
#in the output, egonetlist
#can find them under excluded


##########################################################

#II. Analyzing the ego networks

#a. Degree

#Degree is easy here with ego network data as it is assumed
#to be part of the initial dataset

deg=egonet.data$actual.degree #if we wanted to use the actual, full degree

#deg=egonet.data$listed.degree #if we wanted to use degree, capped at five


#let's get a histogram for that distribution

hist(deg,col="light blue",freq=F
,main="",xlab="Degree",xlim=c(0,15),breaks=15)
lines(density(deg),col = 2, lty = 2, lwd = 2)


#b. Composition
```

```
#with ego networks we are often interested in the
#the composition of the network, in terms
#of demographic characteristics, kin/non-kin,
#drug use, etc.


#we will start here by looking at how similar ego
#is to their alters, or homophily

#here we will consider grade

#let's calculate two measures

#proportion same/different
#one way of doing this is to write a little function
#to calculate this for each ego and then do this over all egos

#we can use the alteri.list part of the
#list as a useful starting point

#let's take a look at the first ego
ego1=alteri.list[[1]]
ego1

#example
#here we will write a little function
#to calculate the proportion of alters with the same
#characteristic as ego

homophily.function=function(dat,ego.var,alter.var){
 #here we define
#the generic inputs into our function
#dat=the ego network, represented as in alteri.list
#ego.var is the variable of interest for ego
#alter.var is the variable of interest for the alters

same=dat[,ego.var]==dat[,alter.var] #here ask, for each ego-alter pair,
#if same or different
sum(same)/length(same) #here, sume up number the same and divide but number
#of alters, this is what is returned when you run the function
}

#here, let's run our function for the first ego
#and look at grade, so ego.var="grade"
#and alter.var is "grade1"

homophily.function(ego1,ego.var="grade",alter.var="grade1")

#now let's do this over all egos
lapply(alteri.list,homophily.function,ego.var="grade",alter.var="grade1")

#here is the same basic logic, but we want to do this over a
#all egos in our list, alteri.list
#we will use an lapply statement, telling R what function, here
```

```
#homophily.function we want to use over the list
#we also need to input the other inputs, as before

prop.same=lapply(alteri.list,homophily.function,ego.var="grade",alter.var="grade
1")
prop.same=unlist(prop.same)  #getting our input in a vector, easier to work with
prop.same

#E-I index (from point of view of ego)
#we will use the same basic logic as above, but
#here the function is different
#want: E-I/(E+I)
#where E=in-group edges and I=out-group edges

EI_function=function(dat,ego.var,alter.var){
#same inputs as above

same=dat[,ego.var]==dat[,alter.var] #same as before,
#again, as how many sae as ego

E=sum(same) #here we sum the number of alter that
#are the same as ego
I=length(same)-E #here we calculate the number different from ego
(E-I)/(E+I) #here calculate index
}


#again, we can use an lapply statement to calculate
#EI index over all egos at once

EI=lapply(alteri.list,EI_function,ego.var="grade",alter.var="grade1")
EI=unlist(EI)
EI

#homogeneity/heterogeneity
#We can also calculate measures of homogeneity or heterogeneity
#when we ignore ego, just looking at the characteristics
#of the alters, how diverse is one's social network?

#here we will calculate the IQV for each ego network

#same setup, here we write a little function
#to calculate the IQV
#showing how heterogeneous the ego network is
#compared maximum possible heterogeneity
#1 is where evenly distributed
#0 is where all in one category

IQV_function=function(dat,alter.var,k=NULL){
#output is a little different here,
#still need alter data but now no longer
#ego variable
#also add a k input showing how many categories
#there were for the characteristic of interest
tab=table(dat[,alter.var]) #grab alter characteristics
#and produce a frequency table
pi=tab/sum(tab) #now proportion in each category
k/(k-1)*(1-sum(pi^2)) #here calculate IQV
}

IQV=lapply(alteri.list,IQV_function,alter.var="grade1",k=3)
IQV=unlist(IQV)
IQV

#group mixing
#here we will look at the rate of mixing across grades
#aggregated across all networks

#here will use the alter.df part of the list generated originally
#this is a dataset of alters, all put together

alteri.df=egonetlist$alteri.df
alteri.df[1:3,]

mixing.table=table(alteri.df[,"grade"],alteri.df[,"grade1"])
#here create a simple table of ego grade by alter grade
#each ego will be included in the table multiple times
#once for each alter

#we could then use that information to simulate networks, run log-linear models,
etc.
#as a way of analyzing social boundaries
#note we can also run ergms on ego network data, and terms for mixing between
groups
#will be, generally, one of the main parts of the model


#c. structure of ego networks

#for simplicity we will just focus on density of
#the ego networks

#here we can use the canned functions in igraph
#and the network part of the list

#egonetlist$graphs
edge_density(egonets[[1]])

#over all of the networks using an lapply statement
egodensity=lapply(egonets,edge_density)
egodensity=unlist(egodensity)
egodensity
```