Part IV - Stochastic Actor Based (Siena) Models

0. Getting Started

0.1. Things you'll need

- Download and install the latest release (3.2.0) of R's base distribution from http://cran.r-project.org/
- All course materials will be available for download from http://bit.ly/EPIC-SNA.
- NOTE: This tutorial assumes you save everything in the same working directory. If you choose to move files around, you will need to specify file paths when called.

0.2. Tutorial conventions

- All text in Century Schoolbook font provides general information.
- Text in Courier font represents R-specific information:
  - R Code will all appear with a highlighted background:
    - **Bold type is code (which you can copy into R).**
    - Regular type is comments (begin with #).
  - R Output will appear in Courier without highlighting.

0.3. Setting up your R environment

```
#install.packages("RSiena")                          # The first time you install a package
library(RSiena)
#install.packages("snow")                             # The first time you install a package
library(snow)                                        # some RSiena dependencies
#install.packages("rlecuyer")                         # The first time you install a package
library(rlecuyer)                                    # some RSiena dependencies
library(network)
library(sna)
```

Let's grab the data file constructed for today and a take a look at what format it's in.

```
load("Tutorial.Rdata")
ls()                                                 # what's in the data file?
```

9. Stochastic Actor-Based Models

We are going to use a script adapted by Tom Snijders and several of his colleagues to estimate a simple model of the coevolution of networks and one behavior, allowing for one other covarying behavior. The data are a synthetic dataset extracted from a larger study of substance use among teens in Scotland. The tutorial we have is an extract from their code, which is available on Tom's website. For further details on what we're doing here, you should see their extremely well documented files - http://www.stats.ox.ac.uk/~snijders/siena/.

```
class(friend.data.w1)
class(drink)                                         # What format are these data in?
```

Ok, so basically what we have are a few matrices – one each for the friendship adjacency matrices across the time steps, and one each for the behaviors, with the columns corresponding to the waves. The values are simply coded frequencies of use. Let's start by getting a sense of what the behavior & network data look like.

```
drink
apply(drink, 2, table)                  # frequencies for each time point "2" applies the table
                                          function within columns [use "1" for rows])
apply(smoke,2,table)
net1 <- as.network(friend.data.w1)
net2 <- as.network(friend.data.w2)
net3 <- as.network( friend.data.w3)
par(mfrow=c(1,3))                       # Creating a 3 panel display of the networks
plot( net1, xlab = 'friendship t1')
plot( net2, xlab = 'friendship t2')
plot( net3, xlab = 'friendship t3')
par(mfrow=c(1,1))
```

Before moving on, let's add the attributes to the network data and regenerate these plots.

```
set.vertex.attribute(net1, "drink", drink[,1])    # one way to attach attribute info
set.vertex.attribute(net2, "drink", drink[,2])
set.vertex.attribute(net3, "drink", drink[,3])
net1 %v% "smoke" <- smoke[,1]                     # an equivalent way to do the same
net2 %v% "smoke" <- smoke[,2]
net3 %v% "smoke" <- smoke[,3]
par(mfrow=c(1,3))                                 # Creating a 3 panel display of the networks
plot(net1, vertex.cex=0.5*get.vertex.attribute(net1, "drink"), vertex.col= "smoke", xlab =
  'friendship t1')
plot(net2, vertex.cex=0.5*get.vertex.attribute(net2, "drink"), vertex.col= "smoke", xlab =
  'friendship t2')
plot(net3, vertex.cex=0.5*get.vertex.attribute(net3, "drink"), vertex.col= "smoke", xlab =
  'friendship t3')
par(mfrow=c(1,1))
```

RSiena more or less was developed as a "wrapper" set of scripts for handling an old version of this program that ran as standalone software (known as StOCNet). That is useful for knowing why some of the specification/code norms are a bit "non-standard." In essence you're creating objects that recreate the file formats that StOCNet relied on. To start, we specify each of the variables we're going to use, noting what type of variable they will be used as in the model you're constructing. Here we're going to predict the coevolution of friendship and drinking behavior, with smoking as a time-varying covariate. To run other sorts of models, you would build your data in a different format corresponding to that model. This is a good point to note that behavioral data in RSiena must be coded as integer values, no higher than 10, and estimation has suggested it's best and handling those with some, but not too much variation in potential values (e.g., 3-5).

```
friendship <- sienaNet(array(c(friend.data.w1,        # the networks – "sienaNet" identifies DVs
  friend.data.w2, friend.data.w3 ), dim = c(50, 50,
  3)), type="oneMode")
drinkingbeh <- sienaNet(drink, type = "behavior")     # also as a DV
smoke1 <- varCovar(smoke)                             # a time-varying covariate
#gend <- coCovar(gender)                              # if instead we wanted a constant covariate
```

There are other variable options as well, and you should see the documentation for the expected formats of others. Now we need to combine these into a single data object, and initialize the effects object.

```
mydat <- sienaDataCreate (friendship, smoke1, drinkingbeh)
myeff <- getEffects(mydat)                                  # initial model specification
```

This specifies an initial model, using the defaults that Siena identifies given the variable definitions you provided (remember we've already specified we have 2 DVs and one time-varying covariate; this isn't *just* a data.frame. Let's take a look at what that did.

```
print01Report(mydat, myeff, modelname = 's50_3_init')        # this writes a file to your wd
```

You should find and take a look at the file we just created. It has some details about an initial model specification with the variables we've defined. Next, RSiena has some inherent conventions that may be less than obvious. One of these is "shortnames." Here we will make use of them, but explicitly specifying which of them we want via "interactions" (which unfortunately aren't interactions in the way you're thinking). Take a look at the available terms and their "shortnames." You'll note that some of them share "shortnames."

```
cbind(myeff$effectName,myeff$shortName)
```

You should take a look at the effects already to be included in the model with the myeff object. Then, we're going to add some effects to the model.

```
myeff <- includeEffects(myeff, transTrip, cycle3)     # adding transitive and cyclic triples
```

Now for some node-level behavioral effects. Here we'll use the ego, alter and similarity versions for each. If we didn't add the "interaction1" part of these specifications, it wouldn't know which of the "shortname" egoX/altX/simX variables to grab.

```
myeff <- includeEffects(myeff, egoX, altX, simX,        # also as a DV
  interaction1 = "smoke1")
myeff <- includeEffects(myeff, egoX, altX, simX,        # a time-varying covariate
  interaction1 = "drinkingbeh")
myeff <- includeEffects(myeff, name = "drinkingbeh",    # avAlt is the alter-based average (ties)
  avAlt, indeg, outdeg, interaction1 = "friendship")
#myeff <- includeEffects(myeff, cycle3, include=F)      # how to remove an effect from the model
myeff                                                   # what have we done to this thing
```

Now let's initialize the model evaluation and tell it where to write the files. You shouldn't use the same one as the file above, as you'd like to keep those initialized values and information from being overwritten. Once this file is initialized, each subsequent model you run will have its output appended to the specified file. Once initialized, we'll run a model.

```
myCoEvModel <- sienaModelCreate( useStdInits = TRUE,    # the provide projname is the filename
  projname = 'model_1' )
ans <- siena07(myCoEvModel, data = mydat, effects =     # this "siena07" is the actual model call
  myeff)
```

This will bring up a visual console that will summarize the progress of the model being fit. Once it's run, you should open up the file produced and take a look at the information provided there. Some of it can be summarized by simply calling the model object.

A few things we might want to do. If you would prefer the model estimation updates to be spooled to the R console instead of the visual popup, you can suppress that. Also, if models fail to converge (as indicated by t-statistics that fail to drop below 0.1), you might want to continue running the model, picking up from the end of the prior estimation, rather than starting over.

```
#ans <- siena07( myCoEvModel, data = mydat, effects =  # suppressing the visual console
  myeff, batch=T, verbose=F)
ans2 <- siena07(myCoEvModel, data=mydat,               # rerun, starting from where we left off
  effects=myeff, prevAns=ans)
```

Here's an alternate model dropping the in-/out- degree effects on behavior, which improves the fit modestly. Note that since we use the same initialized model, it will be appended to the same file.

```
myeff2 <- includeEffects(myeff, name = "drinkingbeh", indeg, outdeg, interaction1 = "friendship",
  include=F)
ans3 <- siena07( myCoEvModel, data = mydat, effects = myeff2)
```