Advanced Network Analysis

Stochastic Actor-Oriented Models

Olga Chyzh [www.olgachyzh.com]

SAOM

Stochastic actor oriented model developed primarily by Snijders is implemented in the RSiena package on CRAN:

- https://www.stats.ox.ac.uk/~snijders/siena/
- Recent overview piece by Snijders

The Siena webpage

SIENA is a program for the statistical analysis of network data, with the focus on social networks. Networks here are understood as entire (complete) networks, not as personal (egocentered) networks: it is assumed that a set of nodes (social actors) is given, and all ties (links) between these nodes are known - except perhaps for a moderate amount of missing data.
SIENA is designed for analyzing various types of data as dependent variables:

Longitudinal network data:
This refers to repeated measures of networks on a given node set (although it is allowed that there are some changes in the node set). Models can be specified with actor-oriented as well as tie-oriented dynamics; but mainly the former.

Longitudinal data of networks and behavior:
This is like longitudinal network data, but in addition there are one or more changing nodal variables that are also treated as dependent variables, and referred to as *behavior*. The network will influence the dynamics of the behavior, and the behavior will influence the dynamics of the network. In other words, this is about the co-evolution of networks and behavior.

Cross-sectional network data

SAOM Assumptions

- Actors control their outgoing ties and have full knowledge of broader network
- Evolution of network process occurs in microsteps
- Only one tie can change at a microstep
- Tie change only depends on the present network

SAOM Broadstrokes

- The simulation starts out at the network observed at the first time point t_0
- An actor is chosen randomly using a rate function
- The identified actor gets the opportunity to set a micro step. The actor's choice is determined by their objective function
- Model time is updated and simulation proceeds at step 2
- The simulation terminates once modified network resembles network at t_1

Rate Function

- Waiting time until change can be made by any actor follows an exponential distribution with parameter $\lambda_t g$ (g refers to number of actors in the network)
    - Values of λ_t are estimated by calculating the number of edge differences between networks:
    - The higher λ_t is the greater the number of changes between observation moments
- Probability that an actor i has the opportunity to make a change is equal to $1/g$

Actor's Objective Function

$f_i(\beta, x(i \rightsquigarrow j)) = \sum_{l=1}^{k} \beta_l s_{il}(x(i \rightsquigarrow j)) + U_i(t, x, j)$, where

- $s_{il}(x(i \rightsquigarrow j))$ represents k structural and exogenous effects
- β_l are statistical parameters
- and $U_i(t, x, j)$ is a random utility term

Multinomial Choice Model

$$p_{ij}(\beta, x(i \rightsquigarrow j)) = \frac{exp(f_i(\beta, x(i \rightsquigarrow j)))}{\sum_{h=1}^{g} exp(f_i(\beta, x(i \rightsquigarrow h)))}$$

- Represents the probability with which actor i changes his outgoing ties
- When $i = j$ this probability refers to the probability of not changing anything

Parameter Estimation

Solving the model requires the estimation of $\boldsymbol{\theta} = (\boldsymbol{\lambda}, \boldsymbol{\beta})$ using a Method of Moments approach (MoM)

- Suitable statistic for $\boldsymbol{\lambda}$:

$$S_{\lambda_{tm}} = \sum_{i,j=1,\ i \neq j}^{g} |Y_{ij,t_{m+1}} - Y_{ij,t_m}|$$

- Suitable statistic for $\boldsymbol{\beta}$:

$$S_{\beta_{l,tm}} = \sum_{i=1}^{g} s_{il}(Y_{t_{m+1}})$$

Stochastic Approximation Process

Combining the suitable statistics, we next determine the value $\hat{\boldsymbol{\theta}}$ for $\boldsymbol{\theta}$ as the solution of the system of equations:

$$g_n(\boldsymbol{\theta}|z_n) = \sum_{t_a \in T} (E_{\boldsymbol{\theta}}\{u(Y^{(a+1)}|Y^{(a)} = y^{(a)})\} - u(y^{(a+1)})),$$

- z_n simply means all available data
- $u(x)$ corresponds to the statistic being estimated
- The estimation for the MoM relies on MCMC simulations of the network change process (Robbins & Monro, 1951)

Example: Friendship Networks

```r
library(RSiena)
    friend.data.w1 <- s501
    friend.data.w2 <- s502
    friend.data.w3 <- s503
    drink <- s50a
    smoke <- s50s
```

Specify the Network DV:

```r
friendship <- sienaDependent(
                    array( c( friend.data.w1, friend.data.w2, frienc
                    dim = c( 50, 50, 3 ) ) )
```

```r
friendship
```

```
## Type         oneMode
## Observations 3
## Nodeset      Actors (50 elements)
```

```r
class(friendship)
```

```
## [1] "sienaDependent"
```

```r
dim( friendship)
```

```
## [1] 50 50  3
```

```r
attributes(friendship)
```

```
## $dim
## [1] 50 50  3
##
## $class
## [1] "sienaDependent"
##
## $type
## [1] "oneMode"
##
## $sparse
## [1] FALSE
##
## $nodeSet
## [1] "Actors"
##
```

Specify the Behavior DV:

```
drinkingbeh <- sienaDependent( drink, type = "behavior" )
drinkingbeh
```

```
## Type         behavior
## Observations 3
## Nodeset      Actors (50 elements)
```

Specify IVs:

```
smoke1 <- coCovar( smoke[ , 1 ] )

# Put the variables together in the data set for analysis
NBdata <- sienaDataCreate( friendship, smoke1, drinkingbeh)
NBdata
```

```
## Dependent variables:  friendship, drinkingbeh
## Number of observations: 3
##
## Nodeset                          Actors
## Number of nodes                      50
##
## Dependent variable friendship
## Type               oneMode
## Observations       3
## Nodeset            Actors
## Densities          0.046 0.047 0.05
##
## Dependent variable drinkingbeh
## Type               behavior
## Observations       3
## Nodeset            Actors
## Range              1 - 5
```

Possible Types of IVs

- `coCovar`--constant node-level covariate (does not change between time periods)
- `varCovar`--time-variable node-level covariate
- `coDyadCovar`--constant edge-level covariate
- `varDyadCovar`--time-varying edge-level covariate
- `sienaCompositionChange`--over time changes in node set (e.g., some actors leave the network)

```
?coCovar
```

Specify Endogenous Effects

```
NBeff <- getEffects( NBdata )
NBeff
```

```
##   name        effectName                               include fix   test
## 1 friendship  constant friendship rate (period 1) TRUE    FALSE FALSE
## 2 friendship  constant friendship rate (period 2) TRUE    FALSE FALSE
## 3 friendship  outdegree (density)                 TRUE    FALSE FALSE
## 4 friendship  reciprocity                         TRUE    FALSE FALSE
## 5 drinkingbeh rate drinkingbeh (period 1)         TRUE    FALSE FALSE
## 6 drinkingbeh rate drinkingbeh (period 2)         TRUE    FALSE FALSE
## 7 drinkingbeh drinkingbeh linear shape            TRUE    FALSE FALSE
## 8 drinkingbeh drinkingbeh quadratic shape         TRUE    FALSE FALSE
##   initialValue parm
## 1    4.69604   0
## 2    4.32885   0
## 3   -1.46770   0
## 4    0.00000   0
## 5    0.70571   0
## 6    0.84939   0
## 7    0.32237   0
## 8    0.00000   0
```

Effects Description

```
effectsDocumentation(NBeff)
```

row	name	effectName	shortName	type	inter1	inter2	parm	interactionType
1	friendship	constant friendship rate (period 1)	Rate	rate			0	
2	friendship	constant friendship rate (period 2)	Rate	rate			0	
3	friendship	outdegree effect on rate friendship	outRate	rate			0	
4	friendship	indegree effect on rate friendship	inRate	rate			0	
5	friendship	reciprocity effect on rate friendship	recipRate	rate			0	
6	friendship	effect 1/outdegree on rate friendship	outRateInv	rate			0	
7	friendship	effect ln(outdegree+1) on rate friendship	outRateLog	rate			1	
8	friendship	effect smoke1 on rate	RateX	rate	smoke1		0	
9	friendship	effect drinkingbeh on rate	RateX	rate	drinkingbeh		0	

Specify Effects

```
NBeff <- includeEffects( NBeff, transTrip, transRecTrip )
```

```
##   effectName                 include fix   test  initialValue parm
## 1 transitive triplets        TRUE    FALSE FALSE          0   0
## 2 transitive recipr. triplets TRUE    FALSE FALSE          0   0
```

```
NBeff <- includeEffects( NBeff, egoX, egoSqX, altX, altSqX, diffSqX,
                         interaction1 = "smoke1"   )
```

```
##   effectName           include fix   test  initialValue parm
## 1 smoke1 alter         TRUE    FALSE FALSE          0   0
## 2 smoke1 squared alter TRUE    FALSE FALSE          0   0
## 3 smoke1 ego           TRUE    FALSE FALSE          0   0
## 4 smoke1 squared ego   TRUE    FALSE FALSE          0   0
## 5 smoke1 diff. squared TRUE    FALSE FALSE          0   0
```

```
NBeff
```

```
##    name        effectName                           include fix   test
## 1  friendship  constant friendship rate (period 1) TRUE    FALSE FALSE
## 2  friendship  constant friendship rate (period 2) TRUE    FALSE FALSE
```

Define the Model:

```
myalgorithm1 <- sienaAlgorithmCreate( projname = 's50_NB' )

# Estimate using the second algorithm right from the start.
NBans <- siena07(myalgorithm1, data = NBdata, effects = NBeff)
NBans <- siena07(myalgorithm1, data = NBdata, effects = NBeff, batch=
```

Look at results

```
NBans
```

```
## Estimates, standard errors and convergence t-ratios
##
##                                           Estimate   Standard   Conve
##                                                        Error      t-r
## Network Dynamics
##    1. rate constant friendship rate (period 1)  6.2738  ( 1.2186   )    0.
##    2. rate constant friendship rate (period 2)  5.0845  ( 0.8572   )   -0.
##    3. eval outdegree (density)                 -2.6246  ( 0.2222   )    0.
##    4. eval reciprocity                          2.7737  ( 0.2664   )    0.
##    5. eval transitive triplets                  0.8915  ( 0.1360   )    0.
##    6. eval transitive recipr. triplets         -0.5130  ( 0.2160   )    0.
##    7. eval smoke1 alter                         0.2548  ( 0.2914   )   -0.
##    8. eval smoke1 squared alter                -0.2197  ( 0.2489   )    0.
##    9. eval smoke1 ego                           0.0873  ( 0.3034   )    0.
##   10. eval smoke1 squared ego                   0.0104  ( 0.2501   )    0.
##   11. eval smoke1 diff. squared                -0.0981  ( 0.0679   )   -0.
##
## Behavior Dynamics
##   12. rate rate drinkingbeh (period 1)          1.1712  ( 0.3042   )   -0.
##   13. rate rate drinkingbeh (period 2)          1.6518  ( 0.4074   )    0.
```

Actors Entering and Exiting the Network

```
library(devtools)
#install_github("ochyzh/networkdata")
data("duqueData")
dim(dipl_ties[[1]])
```

```
## [1] 134 134
```

```
dim(dipl_ties[[2]])
```

```
## [1] 148 148
```

- Remember that in these data, time periods have varying numbers of observations, as states enter and leave the system.
- In order to use RSiena, we must have the same number of actors in each time period. If an actor is missing, their tie values are coded as either `NA` or some other code we will passed to `sienaCompositionChange` option (see the manual for the second option).
- Note: if you plan to use network analysis, you have to become very comfortable with these types of data issues.

Setting Up the DV

```
library(tidyverse)
#get the full list of actors:
myactors<-unique(do.call("c",lapply(dipl_ties,names)))
dyads<-expand.grid(myactors,myactors)

dipl<-array(NA, dim = c( 194, 194, 8 ),
    dimnames=list(myactors,myactors,seq(from=1970,to=2005,by=5)))

for(t in 1:8){
    d<-dipl_ties[[t]]
    for(i in 1:nrow(d)){
      for (j in 1:ncol(d)){
        a1 = names(d)[i]
        a2 = colnames(d)[j]
        val = as.numeric(as.character(d[i,j]))

        dipl[i,j,t] <- val
        dipl[j,i,t] <- val
      }}
    }

dipl <- sienaDependent(dipl)
```

Your Turn

1. Set up `allies` and `contig` as edge-level covariates.
2. Set up `polity` as a time-varying node-level covariate.
3. Estimate a model that includes the following covariates: outdegree, reciprocity, transitive triplets, polity alter, polity ego, polity diff., contiguity, and allies.

```r
ans<-readRDS("data/ans.rds")
summary(ans)
```

```
## Estimates, standard errors and convergence t-ratios
##
##                                    Estimate   Standard   Convergence
##                                                 Error      t-ratio
##
## Rate parameters:
##   0.1      Rate parameter period 1 87.4577  ( 6.8145   )
##   0.2      Rate parameter period 2 76.4185  ( 4.2766   )
##   0.3      Rate parameter period 3 63.6222  ( 2.6922   )
##   0.4      Rate parameter period 4  5.1526  ( 0.2161   )
##   0.5      Rate parameter period 5 38.1545  ( 1.1522   )
##   0.6      Rate parameter period 6  5.6488  ( 0.2027   )
##   0.7      Rate parameter period 7  7.0067  ( 0.2222   )
##
## Other parameters:
##   1.  eval degree (density)        -1.1331  (     NA   )    1.8070
##   2.  eval GWESP (69)               0.2528  (     NA   )   -1.0283
##   3.  eval cont                     0.3578  (     NA   )   -1.8619
##   4.  eval ally                     0.5063  (     NA   )   -3.3637
##   5.  eval dem alter                0.1103  (     NA   )   -1.4067
##   6.  eval dem ego                  0.1170  (     NA   )   -1.4067
##
## Overall maximum convergence ratio:       NA
##
```

TERGM vs. SAOM

- Block et al. 2017
- Block et al. 2018
- Leifeld & Cranmer 2018