Advanced Network Analysis

ERGM Specification and Implementation

Olga Chyzh [www.olgachyzh.com]

Readings

David R. Hunter, Mark S. Handcock, Carter T. Butts, Steven M. Goodreau, and Martina Morris. Ergm: A package to fit, simulate and diagnose exponential-family models for networks. *Journal of Statistical Software*, 24(3):1--29, 2008.

Emily Kalah Gade, Michael Gabbay, Mohammed M. Hafez, and Zane Kelly. Networks of cooperation: Rebel alliances in fragmented civil wars. *Journal of Conflict Resolution*, 63(9):2071--2097, 2019.

ERGM Specification: Ground Rules

We will be working with a network that consists of the ten 10 most ideological senators from the 109th Congress. An edge ij in this network is defined as equal to 1 if i cosponsored j at least two times. Suppose you think that senators are more likely to co-sponsor legisation if they: a. Come from nearby states, b. Have similar ideology.

ERGM Specification: Ground Rules

1. Your dependent variable must be a network object with binary edges. The `network` command creates network objects from edgelists or adjacency matrices. Pay attention to whether you are working with a directed or an undirected network.

```
library(statnet)
library(devtools)
#install_github("ochyzh/networkdata")
library(networkdata)
```

```
data(legnet)
mynet<-network(el, matrix.type="edgelist",
                directed=TRUE, loops=FALSE)
```

ERGM Specification: Ground Rules

1. All edge-level/dyadic covariates must be stored as matrix objects and defined as *network attributes* using `set.network.attribute` command. Make sure your vertices are named/sorted the same way in all datasets.

```
# Convert the object "edist" which contains euclidean distance (units
edist <- as.matrix(edist)
# Define network attribute
set.network.attribute(mynet,"dist",edist)
```

ERGM Specification: Ground Rules

1. All node-level covariates must be defined as *vertex attributes* using `set.vertex.attribute` command. Again, pay attention to names/order in which your nodes are sorted in the data.

```
# Define object "dwnom" (ideology) as a vertex attribute
#detach("package:igraph", unload=TRUE) the below command seems to cla
set.vertex.attribute(mynet,"ideol",dwnom$dwnom)
```

Your Turn

1. Plot the Cosponsorship Network. What network features (e.g., triangles, 2-stars) seem prevalent?
2. Estimate an `ergm` with edges, ideological, and geographic homophily;
3. Check goodness of fit. Plot observed values against boxplots of simulated networks.
4. Now add a control for popularity (`istar(2)`).
5. Check MCMC performance. Does the trace plot look good?
6. Add a control for transitivity (`triangle`). If that does not work, try using `gwesp` instead.
7. Increase MCMC sample size to 10000. Make a trace plot. Does it look good?

Gade et al. 2019

- Why Do Rebel Groups Cooperate?
    - power
    - ideology
    - state sponsorship
- Ideological similarigty is the primary driver for rebel cooperation.

Gade et al. 2019 Hypothesis

Ideology:

- Ideological proximity in rebel networks should yield greater militant cooperation than ideological distance.

Power:

- An overriding concern for capability aggregation in rebel movements will tend to produce *symmetric* alliances.
- The desire of strong groups to form alliances that maximize decision-making autonomy vis-$\grave{a}$-vis rivals will generate *asymmetric* alliances.

State sponsorship:

- Rebel groups that share the same state sponsor will cooperate more frequently.

Data

```r
# load data
data(gadeData)


gadeData #to look at the data
```

```
##       Var1  Var2 coopActions          id ideol_diff.dyad powerdiff.dyad lo
## 2     13th 101st    1.000000  13th_101st     0.000000000           0.20
## 3     AARB 101st    0.000000  AARB_101st     0.333333333           7.00
## 4       AF 101st    0.000000    AF_101st     0.000000000           8.00
## 5      ANF 101st    0.000000   ANF_101st     3.666666667           5.00
## 6     ASIM 101st    1.000000  ASIM_101st     1.833333333          13.00
## 7     ISIL 101st    0.000000  ISIL_101st     4.000000000          23.00
## 8     AASB 101st    0.000000  AASB_101st     1.000000000           0.50
## 9      ADF 101st    0.000000   ADF_101st     3.666666667           0.45
## 10    AASG 101st    0.000000  AASG_101st     1.556666667           1.00
## 11     ARC 101st    0.000000   ARC_101st     0.666666667           4.00
## 12      LF 101st    0.000000    LF_101st     1.333333333           6.00
## 13     ATB 101st    0.000000   ATB_101st     1.333333333           6.00
## 14     JAI 101st    0.000000   JAI_101st     2.000000000          15.00
## 15     AFB 101st    0.000000   AFB_101st     0.666666667           0.00
## 16     1st 101st    0.000000   1st_101st     0.333333333           1.10
## 17     AIG 101st    0.000000   AIG_101st     2.000000000           6.50
```

Results

Table 3. Square Root Transformed Dependent Variable.

Variable	Model 1	Model 2	Model 3	Model 4	Model 5
Intercept	.07 (.11)	.75 (.00)	−.00 (.00)	−.48 (.28)	−.41 (.32)
State sponsorship (node)				.07 (.11)	.09 (.12)
Ave. ideology (node)				.07 (.04)	.07 (.04)
Power (node)				.01 (.01)	.00 (.01)
ASIM (node)					.32 (.28)
Ideol. diff. (dyad)	−.04*** (.01)			−.05*** (.00)	−.05*** (.01)
Power diff. (dyad)		−.06* (.04)		−.01*** (.00)	−.01*** (.00)
Shared St. sponsor (dyad)			.06 (.05)	−.00 (.05)	−.10 (.05)
Shared location (dyad)				.17*** (.04)	.17*** (.04)

Note: Results of additive and multiplicative effects regression analysis. Dependent variable is square root of the count of collaborative ties. Standard errors are given in parenthesis.

*$p < .05$.

**$p < .01$.

***$p < .001$.

Replicate Gade et al's Analysis Using an ERGM

- Note that the command `ergm` requires the data to be saved as a `network` object.
- A `network` object may be constructed from a matrix or edgelist.
- Gade et al's dependent variable is `coopActions`.

```
table(gadeData$coopActions) #the dv takes the following values.
hist(gadeData$coopActions)
```

The Dependent Variable

- `coopActions` is coded as the square root of the total number of cooperative acts between rebel groups.
- For our purposes, we will recode this variable as binary, so that it equals to 1 if two groups cooperated at least once and 0 otherwise.

```
gadeData$coopBin<-as.numeric(gadeData$coopActions>0)
table(gadeData$coopBin)
```

```
##
##   0   1
## 758 172
```

Dyadic Covariates

- Note that Gade et al' hypothesis are tested using four dyadic covariates: ideol_diff.dyad, powerdiff.dyad. These covariates are constructed as a function of nodal covariates.
- Also note that there are two edge-level covariates: loc.dyad (location), and spons.dyad (same sponsor). These need to be specified as separate networks.

Prepare the Data

```
# data characs
actors = sort(unique(c(gadeData$Var1, gadeData$Var2)))
gadeData<-sort(gadeData)
#These are the dyadic variables. They
#must be in matrix form.
dyadVars = names(gadeData)[c(12,5:8)]
n = length(actors) ; p = length(dyadVars)

# create empty arr object for all dyad vars
dyadArray = array(0,
    dim=c(n,n,p),
    dimnames=list(actors,actors,dyadVars)
    )
```

```r
# loop through and fill in
for(param in dyadVars){
    for(i in 1:nrow(gadeData)){
        a1 = gadeData$Var1[i]
        a2 = gadeData$Var2[i]
        val = gadeData[i,param]
        dyadArray[a1,a2,param] = val
    }
}
```

```r
# These are node-level variables.
nodeVars = names(gadeData)[9:11]
nodeData = unique(gadeData[,c('Var1',nodeVars)])
rownames(nodeData) = nodeData$Var1
nodeData = nodeData[actors,c(-1)]
```

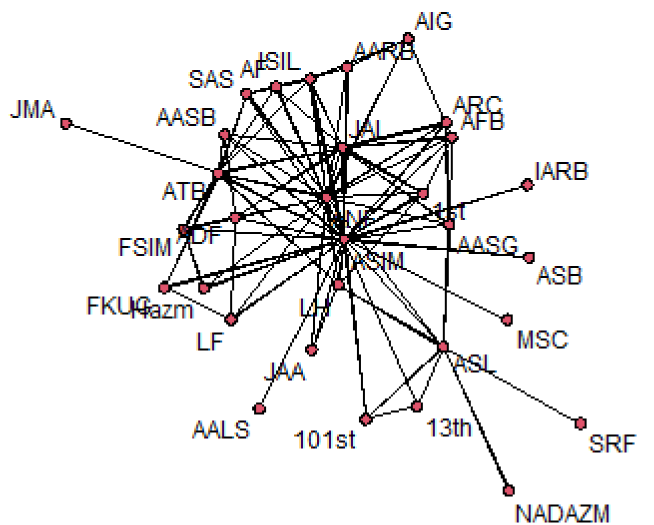
```r
# The DV must be a network object
net = as.network(
    dyadArray[,,'coopBin'],
    directed=FALSE, loops=FALSE,
    matrix.type='adjacency'
    )
```

```r
# Set node attributes
for(param in nodeVars){
    set.vertex.attribute(net, param, nodeData[,param])
}

# Set network attributes:
set.network.attribute(net,'loc.dyad',dyadArray[,,'loc.dyad'])
set.network.attribute(net,'spons.dyad',dyadArray[,,'spons.dyad'])
```

Make a Network Graph:

```
plot(net, label = network.vertex.names(net))
```



Estimate a Logit

```
m0 = ergm(
    net ~
    edges +
    nodecov('averageId.node') +
    nodecov('size.node') +
    nodecov('spons_actor.node') +
    absdiff('averageId.node') +
    absdiff('size.node') +
    edgecov('loc.dyad') +
    edgecov('spons.dyad')
    )
```

Estimate a Logit

```
summary(m0)
```

```
## Call:
## ergm(formula = net ~ edges + nodecov("averageId.node") + nodecov("size.noc
##     nodecov("spons_actor.node") + absdiff("averageId.node") +
##     absdiff("size.node") + edgecov("loc.dyad") + edgecov("spons.dyad"))
##
## Iterations:  7 out of 20
##
## Monte Carlo MLE Results:
##                           Estimate Std. Error MCMC % z value Pr(>|z|)
## edges                     -7.30989    1.24208      0  -5.885  < 1e-04 ***
## nodecov.averageId.node     0.42619    0.11065      0   3.852 0.000117 ***
## nodecov.size.node          0.11015    0.02443      0   4.509  < 1e-04 ***
## nodecov.spons_actor.node   0.53841    0.29636      0   1.817 0.069256 .
## absdiff.averageId.node    -0.21684    0.12422      0  -1.746 0.080877 .
## absdiff.size.node         -0.10494    0.03149      0  -3.332 0.000862 ***
## edgecov.loc.dyad           3.01050    1.02633      0   2.933 0.003354 **
## edgecov.spons.dyad         0.04250    0.41806      0   0.102 0.919018
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

Add Triangles

```
m1 = ergm(
    net ~
    edges +
    nodecov('averageId.node') +
    nodecov('size.node') +
    nodecov('spons_actor.node') +
    absdiff('averageId.node') +
    absdiff('size.node') +
    edgecov('loc.dyad') +
    edgecov('spons.dyad')+
        gwesp
        )
```

Add Triangles

```
summary(m1)
```

```
## Call:
## ergm(formula = net ~ edges + nodecov("averageId.node") + nodecov("size.noc
##     nodecov("spons_actor.node") + absdiff("averageId.node") +
##     absdiff("size.node") + edgecov("loc.dyad") + edgecov("spons.dyad") +
##     gwesp)
##
## Iterations:  4 out of 20
##
## Monte Carlo MLE Results:
##                          Estimate Std. Error MCMC % z value Pr(>|z|)
## edges                    -6.96793    1.19522      0  -5.830  < 1e-04 ***
## nodecov.averageId.node    0.21949    0.08262      0   2.657  0.00789 **
## nodecov.size.node         0.05958    0.01977      0   3.014  0.00258 **
## nodecov.spons_actor.node  0.14499    0.21489      0   0.675  0.49986
## absdiff.averageId.node   -0.21550    0.11265      0  -1.913  0.05576 .
## absdiff.size.node        -0.06690    0.02711      0  -2.468  0.01358 *
## edgecov.loc.dyad          2.56820    1.01579      0   2.528  0.01146 *
## edgecov.spons.dyad        0.05460    0.36702      0   0.149  0.88174
## gwesp                     0.84757    0.32276      0   2.626  0.00864 **
## gwesp.decay               0.83144    0.25561      0   3.253  0.00114 **
```

Assess Model Fit

```
AIC(m1)
```

```
## [1] 342.8487
```

```
BIC(m1)
```

```
## [1] 384.2691
```

```
set.seed(6886)
gofM1 = gof( m1,
    GOF=~degree+espartners+distance-model )
```

Assess Model Fit

```r
# we'll compare against four plots, so set up plotting window
par(mfrow = c(2, 2))
plot(gofM1)
```