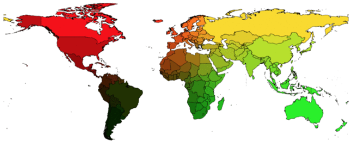
Advanced Network Analysis

TERGM Introduction

Olga Chyzh [www.olgachyzh.com]

Longitudinal networks



-1965

Longitudinal networks

- Networks that change over time.
- Examples: networks of friends, conflict networks, trade networks.
- Want to model network dynamics within and across time periods.

Outline

- Setting up longitudinal network data
- Visualizing longitudinal data
- Descriptive statistics
- Inferential analysis

Getting the data ready

We are used to data in this format:

```r
#install_github("ochyzh/networkdata")
library(networkdata)
data(allyData)
head(dyadData)[,1:8]
```

```
##   cname1 cname2 year ally war contiguity  id1_cinc  id2_cinc
## 2    CAN    USA 1991    1   0          1 0.0119571 0.1364806
## 3    MEX    USA 1991    1   0          1 0.0125758 0.1364806
## 4    COL    USA 1991    1   0          0 0.0046681 0.1364806
## 5    VEN    USA 1991    1   0          0 0.0052502 0.1364806
## 6    PER    USA 1991    1   0          0 0.0033841 0.1364806
## 7    BRA    USA 1991    1   0          0 0.0240151 0.1364806
```

Getting the data ready

We need to convert this information such that:

- the dependent variable must be a `list` of `network` objects
- nodal covariates are vertex attributes in the `list` of `network` objects
- dyadic covariates are included separately in a `list` of `matrices`

Start with setting up war

Output should look like this:

```
class(war)
```

```
## [1] "list"
```

```
length(war)
```

```
## [1] 10
```

```
class(war[[1]])
```

```
## [1] "matrix" "array"
```

```
dim(war[[1]])
```

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

```
war[[1]][1:3,1:3]
```

```
##     USA CHN IND
## USA   0   0   0
## CHN   0   0   1
## IND   0   0   0
```

contiguity should be easier

Output should look like this:

```
class(contiguity)
```

```
## [1] "matrix" "array"
```

```
dim(contiguity)
```

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

```
contiguity[1:3,1:3]
```

```
##     USA CHN IND
## USA   0   0   0
## CHN   0   0   1
## IND   0   1   0
```

Now set up DV with vertex attributes

Output should look like this:

```
class(ally)
```

```
## [1] "list"
```

```
length(ally)
```

```
## [1] 10
```

```
class(ally[[1]])
```

```
## [1] "network"
```

```
list.vertex.attributes(ally[[1]])
```

```
## [1] "cinc"         "cname"        "polity"       "vertex.names" "year"
```

Exploring temporal network data

The `statnet` package includes a range of "sub-packages" that enable you to understand the characteristics of dynamic networks:

- `networkDynamic`: storage and management of temporal network data
- `tsna`: descriptive statistics and graphics for exploratory network analysis
- `ndtv`: utilities for plotting temporal networks (including network movies)

Prepping data

First step is going to be formatting our list of `network` objects into a format that these packages can recognize:

```
allyDyn = networkDynamic(network.list=ally)
```

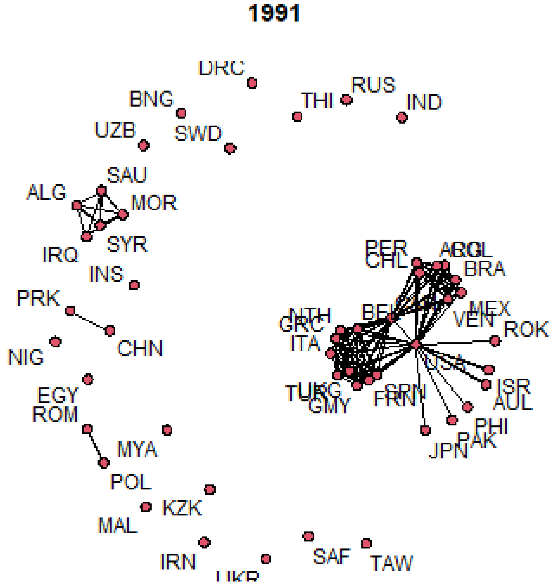
```
## Neither start or onsets specified, assuming start=0
## Onsets and termini not specified, assuming each network in network.list sh
## Argument base.net not specified, using first element of network.list inste
## Created net.obs.period to describe network
##  Network observation period info:
##   Number of observation spells: 1
##   Maximal time range observed: 0 until 10
##   Temporal mode: discrete
##   Time unit: step
##   Suggested time increment: 1
```

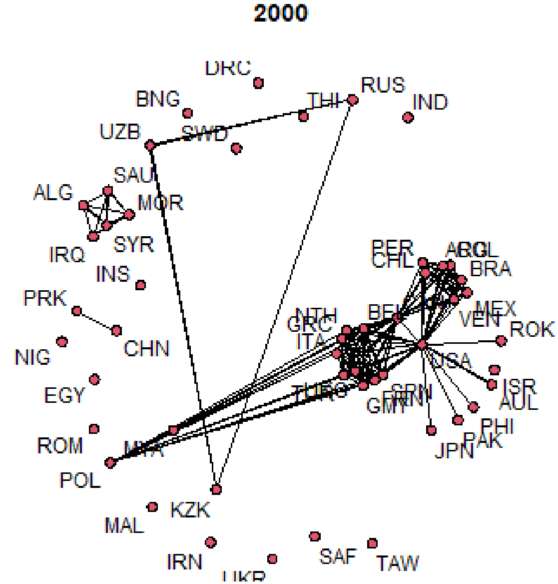
Quick snapshots

`networkDynamic` makes it easy to generate some snapshots of a longitudinal network:

```
par(mfrow = c(1,2))
p<-plot(
    network.extract(allyDyn, at = 0),
    main = "1991", displaylabels = T)
plot(
    network.extract(allyDyn, at = 9),
    main = "2000", displaylabels = T,coord=p)
```

Quick snapshots





Quick movie

`ndtv` makes it pretty easy to render a simple D3 movie for a longitudinal network:

```
library(ndtv)
render.d3movie(allyDyn,
    plot.par=list(displaylabels=T),filename="AlliesNetwork.html", lau
```

Descriptive analyses

`tsna` enables us to quickly calculate some basic descriptive statistics such as the density of a graph:

```r
library(tsna)
tSnaStats(allyDyn, "gden") # Changes in graph density
```

```
## Time Series:
## Start = 0
## End = 10
## Frequency = 1
##         Series 1
##  [1,] 0.08816327
##  [2,] 0.09061224
##  [3,] 0.09061224
##  [4,] 0.08979592
##  [5,] 0.08979592
##  [6,] 0.08979592
##  [7,] 0.09795918
##  [8,] 0.09795918
##  [9,] 0.09795918
## [10,] 0.09795918
## [11,]         NA
```

Descriptive analyses

Can also examine changes in transitivity over time:

```r
tSnaStats(allyDyn, "gtrans") # Changes in graph transitivity
```

```
## Time Series:
## Start = 0
## End = 10
## Frequency = 1
##         Series 1
##  [1,] 0.7617647
##  [2,] 0.7768924
##  [3,] 0.7768924
##  [4,] 0.7768924
##  [5,] 0.7768924
##  [6,] 0.7768924
##  [7,] 0.7974684
##  [8,] 0.7974684
##  [9,] 0.7974684
## [10,] 0.7974684
## [11,]        NA
```

Descriptive analyses

The `tErgmStats` enables us to calculate changes in `ergm` terms over time:

```
tErgmStats(allyDyn, "~ edges+triangle")
```

```
## Time Series:
## Start = 0
## End = 10
## Frequency = 1
##    edges triangle
##  0   108      259
##  1   111      260
##  2   111      260
##  3   110      260
##  4   110      260
##  5   110      260
##  6   120      315
##  7   120      315
##  8   120      315
##  9   120      315
## 10     0        0
```

Your Turn

1. Format the war data from above as a `networkDynamic` object.
2. Summarize changes in the number of war initiation (edges) and triangles in the war network.
3. Plot the war network from 1991 and 2000 side-by-side using the `network.extract` function.
4. Make a quick movie of the war network over time using the `render.d3movie` function.

TERGM: Discrete time model

- Developed by Robins & Pattison (2001) and further developed by Hanneke et al. (2010)
- Scholars in political science most notably Cranmer and Desmarais (2011) have eased the use and highlighted the utility of these types of models for political science
- Extension of ERGM to the temporal setting is based on the idea of panel regression
- In a sequence of observations, lagged earlier observations or derived information thereof can be used as predictors for later observations.
    - In other words, some of the statistics are direct functions of an earlier realization of the network
    - In its most basic form, the TERGM is a conditional ERGM with an earlier observation of the network occurring among the predictors.

TERGM: Discrete time model

- To extend ERGM to a longitudinal context, Hanneke et al. (2010) make a Markov assumption on the network from one time step to the next
- Specifically, given an observed network Y^t, make the assumption that Y^t is independent of $Y^1, \ldots, Y^{t-2}$
- Thus a sequence of network observations has the property that:

$$Pr(Y^2, Y^3, \ldots, Y^t | Y^1) = Pr(Y^t | Y^{t-1}) Pr(Y^{t-1} | Y^{t-2}) \ldots Pr(Y^2 | Y^1)$$

TERGM: Discrete time model

- With this assumption in mind we just need to choose a form for the conditional PDF of $P(Y^t|Y^{t-1})$
- $Y^t|Y^{t-1}$ can be expressed through an ERGM distribution, which then gives us what is referred to as a TERGM:

$$\Pr(Y^t|\theta, Y^{t-1}) = \frac{\exp(\theta^T g(Y^t, Y^{t-1},))}{k}$$

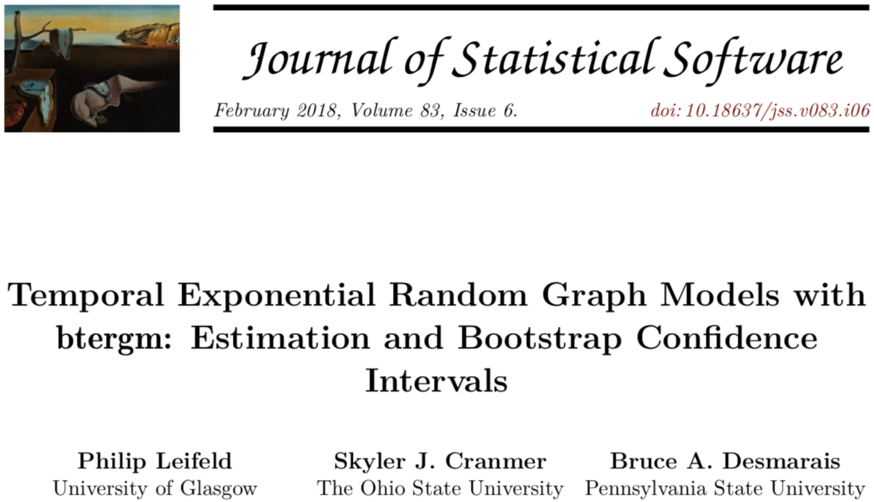
TERGM: Block-diag visualization

- TERGM is essentially estimated through an ERGM with the dependent variable modeled as a block-diagonal matrix (such as below)
- Constraints are put on the model such that cross-network edges in the off-diagonal blocks are prohibited

	a_1	b_1	c_1	d_1	a_2	b_2	c_2	d_2	a_3	b_3	c_3	d_3
a_1	0	1	0	0	×	×	×	×	×	×	×	×
b_1	0	0	1	0	×	×	×	×	×	×	×	×
c_1	0	1	0	0	×	×	×	×	×	×	×	×
d_1	0	0	0	0	×	×	×	×	×	×	×	×
a_2	×	×	×	×	0	1	0	0	×	×	×	×
b_2	×	×	×	×	1	0	0	0	×	×	×	×
c_2	×	×	×	×	1	1	0	1	×	×	×	×
d_2	×	×	×	×	1	0	0	0	×	×	×	×
a_3	×	×	×	×	×	×	×	×	0	0	1	0
b_3	×	×	×	×	×	×	×	×	1	0	0	0
c_3	×	×	×	×	×	×	×	×	0	1	0	1
d_3	×	×	×	×	×	×	×	×	1	1	0	0

btergm Package

- The `btergm` package has been developed by Leifeld, Cranmer, & Desmarais (2018) to estimate longitudinal networks using TERGM



Journal of Statistical Software

February 2018, Volume 83, Issue 6. doi: 10.18637/jss.v083.i06

Temporal Exponential Random Graph Models with btergm: Estimation and Bootstrap Confidence Intervals

Philip Leifeld
University of Glasgow

Skyler J. Cranmer
The Ohio State University

Bruce A. Desmarais
Pennsylvania State University

- Package provides two functions to estimate a TERGM, one using a pseudolikelihood (`btergm`) and the other using MCMC-MLE (`mtergm`)

Running a TERGM

- We are going to run a TERGM on the longitudinal alliance network, and will employ the following specification:
    - `edges`: density term
    - `edgecov(war)`: list of matrices where cross-sections denote war
    - `edgecov(contiguity)`: matrix of distances between countries
    - `absdiff(polity)`: Absolute difference between polity of i and j
    - `absdiff(cinc)`: Absolute difference between cinc of i and j
    - `gwesp(.5, fixed = TRUE)`: Geometric weighted triangle term

Running a TERGM

```r
library(btergm)

tergmFit <- btergm(
    ally ~ edges +
    edgecov(war) + edgecov(contiguity) +
    nodecov('polity') + absdiff("polity") +
    nodecov('cinc') + absdiff("cinc") +
    gwesp(.5, fixed = TRUE)
    )
```

Peaking at the results

```
summary(tergmFit)
```

```
##                               Estimate    2.5%   97.5%
## edges                       -5.3675652 -5.5108 -5.2266
## edgecov.war[[i]]             1.0547194  0.7522  1.3258
## edgecov.contiguity[[i]] -0.2176738 -0.3391 -0.0962
## nodecov.polity               0.0774665  0.0644  0.0883
## absdiff.polity              -0.0099237 -0.0163 -0.0030
## nodecov.cinc                -5.5272184 -7.6859 -3.1217
## absdiff.cinc                21.0799677 18.9045 23.1091
## gwesp.fixed.0.5              2.2031934  2.1109  2.2925
```

Duque (2018)

The DV is diplomatic ties, `dipl_ties`:

```
#Clear your memory and unload `btergm` as it clashes with `network`:

detach("package:btergm", unload=TRUE)
data("duqueData")

class(dipl_ties)
```

```
## [1] "list"
```

```
length(dipl_ties)
```

```
## [1] 8
```

```
class(dipl_ties[[1]])
```

```
## [1] "data.frame"
```

Get an error that need vertex.pid (persistent identifies), as our networks are not of equal size.

The Dependent Variable

we can see that `dipl_ties` is currently a list of `data.frames`. Let's convert it into a list of networks.

```
library(statnet)
for (i in 1:8) {
dipl_ties[[i]]<-as.network(as.matrix(dipl_ties[[i]]))
}
class(dipl_ties[[1]])
```

```
## [1] "network"
```

Use `networkDynamic` for Visualizing the Netwokrk

```
diplDyn = networkDynamic(network.list=dipl_ties, vertex.pid='vertex.n
```

Get an error that need vertex.pid (persistent identifiers), as our networks are not of equal size.

Try Again

```r
#Define network pids:
for (i in 1:8) {
set.network.attribute(dipl_ties[[i]], 'vertex.pid','vertex.names')
}
#Takes 5 min to run:
diplDyn = networkDynamic(network.list=dipl_ties, vertex.pid='vertex.r
diplDyn
```

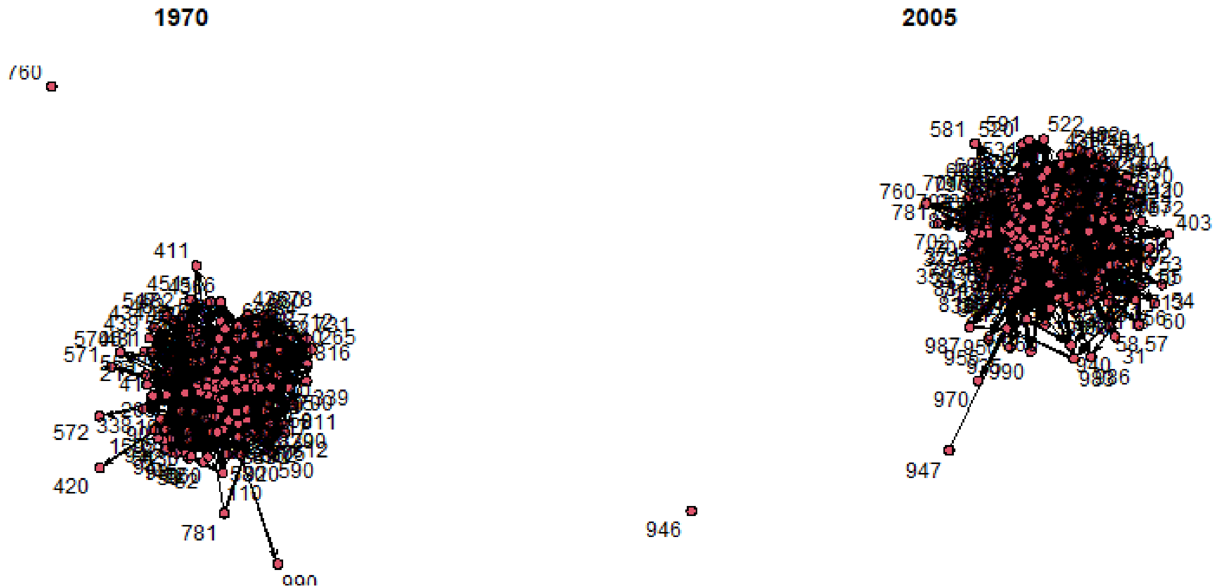
```
## NetworkDynamic properties:
##   distinct change times: 9
##   maximal time range: 0 until  8
##
## Includes optional net.obs.period attribute:
##  Network observation period info:
##   Number of observation spells: 1
##   Maximal time range observed: 0 until 8
##   Temporal mode: discrete
##   Time unit: step
##   Suggested time increment: 1
##
##  Network attributes:
##   vertices = 194
```

Quick snapshots

```
par(mfrow = c(1,2))
plot(
    network.extract(diplDyn, at = 1),
    main = "1970", displaylabels = T)
plot(
    network.extract(diplDyn, at = 6),
    main = "2005", displaylabels = T)
```

Note: cannot use coordinates, because networks are not of equal size, include different actors.

Quick snapshots



What Have We Learned?

- Networks of embassies are dense.
- Some states host very few embassies.
- Note: network graphs of dense networks are not very esthetically pleasing or informative.

Descriptive analyses

`tsna` enables us to quickly calculate some basic descriptive statistics such as the density of a graph:

```
library(tsna)
tSnaStats(diplDyn, "gden") # Changes in graph density
```

```
## Time Series:
## Start = 0
## End = 8
## Frequency = 1
##         Series 1
##  [1,] 0.2050836
##  [2,] 0.2368542
##  [3,] 0.2169688
##  [4,] 0.2145924
##  [5,] 0.2195870
##  [6,] 0.1851018
##  [7,] 0.1902708
##  [8,] 0.2099215
##  [9,]        NA
```

Descriptive analyses

Can also examine transitivity over time:

```r
tSnaStats(diplDyn, "gtrans") # Changes in graph transitivity
```

```
## Time Series:
## Start = 0
## End = 8
## Frequency = 1
##        Series 1
##  [1,] 0.4830388
##  [2,] 0.4953440
##  [3,] 0.5264074
##  [4,] 0.5159558
##  [5,] 0.5269708
##  [6,] 0.4841114
##  [7,] 0.4875437
##  [8,] 0.5132955
##  [9,]        NA
```

Descriptive analyses

The `tErgmStats` enables us to calculate changes in `ergm` terms over time:

```
tErgmStats(diplDyn, "~ edges+triangle")
```

```
## Time Series:
## Start = 0
## End = 8
## Frequency = 1
##   edges triangle
## 0  3655   100235
## 1  5153   190717
## 2  5314   203523
## 3  5597   215072
## 4  5870   238736
## 5  6165   236593
## 6  6618   270378
## 7  7380   336926
## 8     0        0
```

Duque (2018)

Popularity hypothesis: High-status states should receive more recognition simply because of their position in the social structure, rather than because of the possession of status attributes (`2-instars`).

Reciprocity and transitivity: A state's existing relations should influence the state's ability to achieve status (`mutual` and `triangle`).

Homophily: States should recognize states that have similar values and resources as them (`absdiff`).

Dyadic Covariates

Contiguity (`contig`) and alliances (`allies`) are time-varying edge-level covariates. We must make sure that they are stored as lists of matrices.

```
#Contiguity:
class(contig)
```

```
## [1] "list"
```

```
length(contig)
```

```
## [1] 8
```

```
class(contig[[1]])
```

```
## [1] "data.frame"
```

```
dim(contig[[1]])
```

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

```
contig[[1]][1:3,1:3]
```

```
##     2 20 40
## 2   0  1  0
## 20  1  0  0
## 40  0  0  0
```

Allies

```r
#Allies:
class(allies)
```

```
## [1] "list"
```

```r
length(allies)
```

```
## [1] 8
```

```r
class(allies[[1]])
```

```
## [1] "data.frame"
```

```r
dim(allies[[1]])
```

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

```r
allies[[1]][1:3,1:3]
```

```
##    2 20 40
## 2  0  1  0
## 20 1  0  0
## 40 0  0  0
```

Dyadic Covariates

It looks like `allies` and `contig` are currently stored as lists of `data.frames`. We must convert them to lists of matrices.

```
for (i in 1:8) {
contig[[i]]<-as.matrix(contig[[i]])
allies[[i]]<-as.matrix(allies[[i]])
}
```

Now set up DV with vertex and dyadic attributes

Our node-level covariate, `polity$dem_dum` must be defined as a vertex attribute in each of the `dipl_ties` networks.

```
#Define Dem as a vertex attribute for each year of dipl_ties (didn't
  set.vertex.attribute(dipl_ties[[1]],"dem",polity$dem_dum[polity$yea
  set.vertex.attribute(dipl_ties[[2]],"dem",polity$dem_dum[polity$yea
  set.vertex.attribute(dipl_ties[[3]],"dem",polity$dem_dum[polity$yea
  set.vertex.attribute(dipl_ties[[4]],"dem",polity$dem_dum[polity$yea
  set.vertex.attribute(dipl_ties[[5]],"dem",polity$dem_dum[polity$yea
  set.vertex.attribute(dipl_ties[[6]],"dem",polity$dem_dum[polity$yea
  set.vertex.attribute(dipl_ties[[7]],"dem",polity$dem_dum[polity$yea
  set.vertex.attribute(dipl_ties[[8]],"dem",polity$dem_dum[polity$yea

dipl_ties[[1]] %v% "dem"
```

```
##   [1] 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0
##  [38] 0 1 0 0 1 0 0 0 0 1 0 0 0 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0
## [112] 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0
```

```
dipl_ties[[2]] %v% "dem"
```

Specify the Model:

```
library(btergm)
#This runs for 5 min:
tergm_Duque<-btergm(dipl_ties ~ edges + istar(2) + ostar(2) + mutual
                  absdiff("dem")+
                  +nodeicov("dem")+
                  +edgecov(allies)
                  +edgecov(contig),
                  R=1000)

summary(tergm_Duque)
```

Run the Model

```
##                           Estimate    2.5%   97.5%
## edges                    -5.377882 -5.6133 -5.1253
## istar2                    0.028279  0.0267  0.0313
## ostar2                    0.029487  0.0266  0.0340
## mutual                    2.485567  2.3500  2.5956
## triangle                  0.011750  0.0103  0.0127
## absdiff.dem              -0.287938 -0.3656 -0.2207
## nodeicov.dem             -0.204423 -0.2515 -0.1246
## edgecov.allies[[i]]       1.205762  1.1238  1.3205
## edgecov.contig[[i]]       1.765076  1.5732  2.0623
```

Results

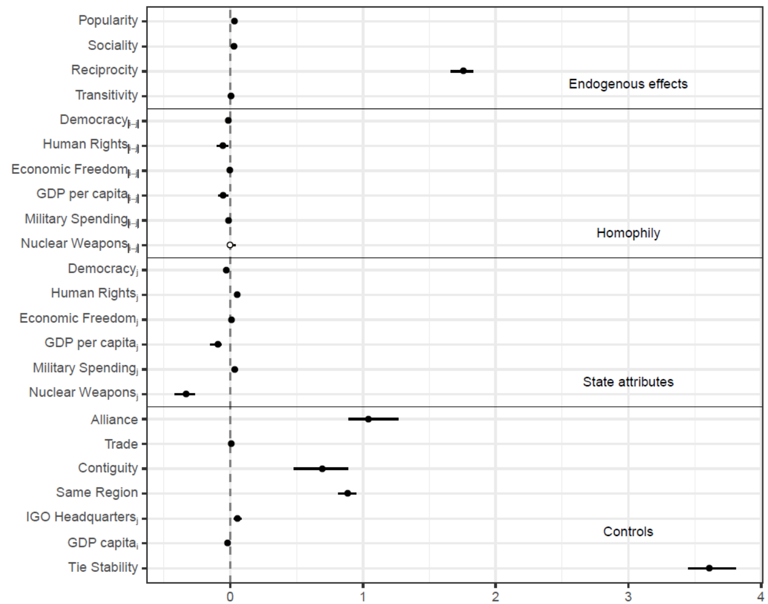


Figure 3. Temporal exponential random graph model of diplomatic ties for 1995-2005.

Bayesing ERGM (BERGM)

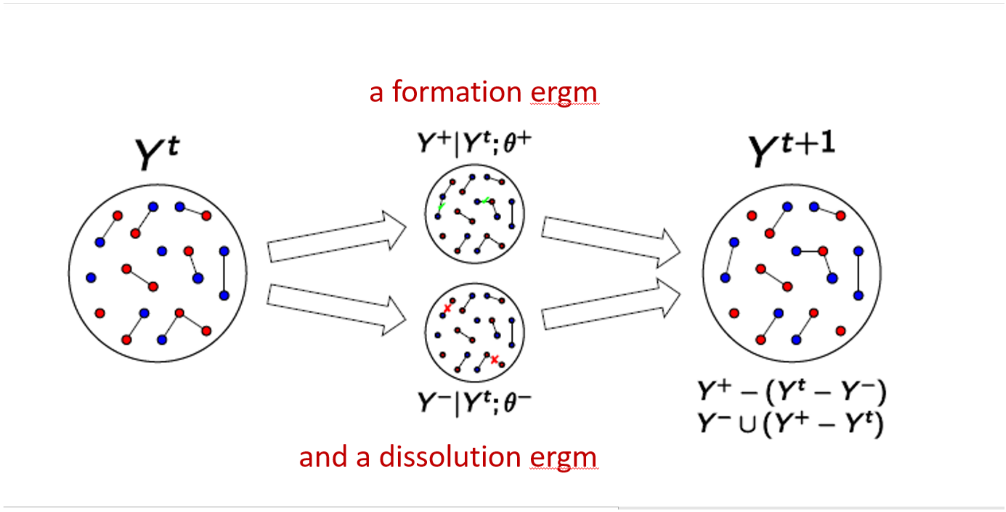
- Paper: Caimo & Friel (2011)
- `Bergm` on CRAN
- Vignette

ego-ERGM

- Paper: Salter-Townshend & Murphy (2016)

Separable temporal ERGM (STERGM)

- Paper: Krivitsky & Handcock (2012)
- `tergm` on CRAN
- Vignette



Hierarchical Exponential-Family Graph Model (HERGM)

- Paper: Schweinberger & Handcock (2015)
- `hergm` on CRAN
- Vignette

Multilevel ERGM

- Paper: Wang et al. (2016)

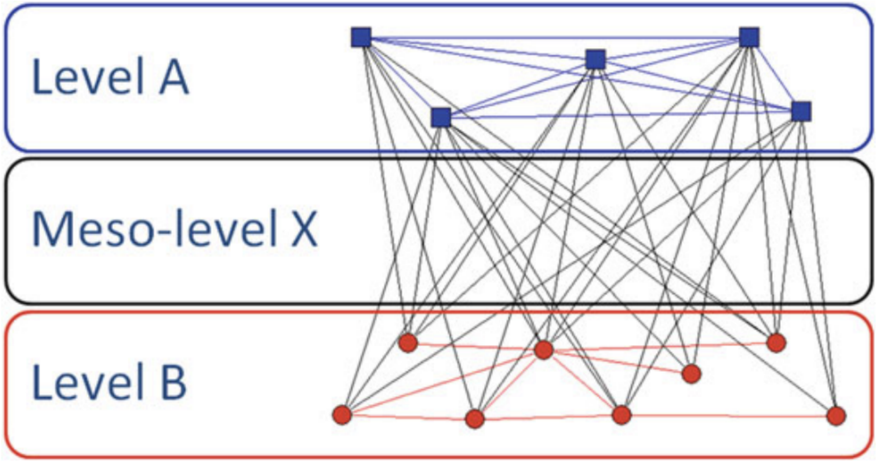


Fig. 6.1 A two-level network representation