SUPPLEMENTARY MATERIALS

Appendix One: Pseudocode for ERGM Estimation Algorithms

The following two figures describe the two algorithms we use to estimate ERGMs and TERGMs. The algorithms are described in pseudocode to make their interpretation parsimonious and independent of the `R` language in which they are implemented.

```
$t$ = number of simulated networks used to approximate likelihood
$\alpha$ = threshold for stopping iterative optimization
$\boldsymbol{\theta}$ = parameter vector
$\Gamma_{hm}$ = the $m^{th}$ (out of $k$) statistics computed on the $h^{th}$ network.
$\Delta_{ll}$ = change in log-likelihood
$o$ = indicator for the observed network
$LL$ = log-likelihood

Initialize $\Delta_{ll}$ to $\infty$
Initialize $LL$ to $-\infty$
Initialize $\boldsymbol{\theta}$ to starting values

while($\Delta_{ll} > \alpha$){
1. Draw $t$ networks from the distribution parametrized with $\boldsymbol{\theta}$
2. Using a hill-climbing algorithm, find $\boldsymbol{\theta}^*$ to maximize $LL^* = \log\left(\frac{\exp(-\sum_{j=1}^{k}\Gamma_{oj}\theta_j^*)}{\sum_{i=1}^{t}\exp(-\sum_{j=1}^{k}\Gamma_{ij}\theta_j^*)}\right)$
3. Store $\boldsymbol{\theta} = \boldsymbol{\theta}^*$
4. Store $\Delta_{ll} = LL^* - LL$
5. Store $LL = LL^*$
}

$\boldsymbol{\theta}$ is now the MCMC-MLE
```

Figure 1: This figure gives the MCMC-MLE algorithm used to estimate ERGMs.

```
$m$ = number of iterations of the bootstrapped resampling
$\boldsymbol{\theta}$ = $p$-length parameter vector
$\boldsymbol{\Theta}$ = $m \times p$ Sample of bootstrap estimates
$T$ = number of networks in the sample
$\boldsymbol{I}$ = $T$-length vector of bootstrap sample indices


for($r \in 1, 2, \ldots, m$){
1. Draw $\boldsymbol{I}$ from $\{1, 2, \ldots, T\}$ with replacement.
2. Using a hill-climbing algorithm, find $\boldsymbol{\theta}^*$ to maximize
       $\sum_{t \in \boldsymbol{I}} \sum_{i=2}^{n} \sum_{j=1}^{i} \log \left[ \prod_{i=2}^{n} \prod_{j=1}^{i} (p_{ij}^t)^{Y_{ij}^t} (1 - p_{ij}^t)^{1 - Y_{ij}^t} \right]$
3. Store$\boldsymbol{\Theta}[r, \ ] = \boldsymbol{\theta}^*$
}
$\boldsymbol{\Theta}$ is now the sample of $m$ bootstrap estimates
```

Figure 2: This figure gives the Bootstrapped Pseudolikelihood algorithm we use to estimate the TERGM.

Appendix Two: R Code for Conducting ERGM Analysis

Below is R code for a synthetic application of the ERGM using the package `ergm`. If the package "ergm" is installed, this can simply be pasted into R.

```
### Use install.packages("ergm") if you have not already ###
library(ergm)

# set the seed for replication purposes
set.seed(5)

## The simulate() function can be used to draw from an erg distribution

# First create an empty network, at which to start the simulation
net0 <- network(matrix(0,50,50), directed=T)
# net0 is directed with 50 nodes

# Now create edge and node-wise covariates
Xedge <- matrix(rnorm(50^2),50,50)
Xnode <- rnorm(50)

# Must set the node covariate as a network attribute
```

```
set.vertex.attribute(net0, attrname = "Xnode",value=Xnode)

# Create a network that depends on the covariates, and has reciprocity
# Make parameters reflect this
theta <- c(-1, 1.25,-.5,-.75)

net1 <- simulate(net0 ~ edges + edgecov(Xedge)+ absdiff("Xnode")+asymmetric,
                 theta0=theta, nsim=1, burnin=1000000)

# Take a simple look at the network just created
plot(net1)

## Now use ergm to recover the effects
# first with MCMC-MLE
ergm.mle <- ergm(net1 ~ edges + edgecov(Xedge)+ absdiff("Xnode")+asymmetric,
                 MCMCsamplesize=10000, maxit=5)
# MCMCsamplesize is the size of the network sample used for approximation
# maxit is the number of EM style updates to the parameters
summary(ergm.mle)




# Now with maximum pseudolikelihood
ergm.mple <- ergm(net1 ~ edges + edgecov(Xedge)+ absdiff("Xnode")+asymmetric,
                  MPLEonly=T)
summary(ergm.mple)

# Now without reciprocity (i.e. dyadic logit for comparison)
ergm.logit <- ergm(net1 ~ edges + edgecov(Xedge)+ absdiff("Xnode"))

## Now Compare models based on fit to out-degree and geodesic distance
# *** Warning: This May Take 5-25 Minutes, Depending on Hardware***
gf.mle <- gof(ergm.mle, GOF = ~odegree+distance)
gf.mple <- gof(ergm.mple, GOF = ~odegree+distance)
gf.logit <- gof(ergm.logit, GOF = ~odegree+distance)

## Produce a simple plot of the goodness of fit, see ?plot.gofobject for explanation
# First column is out-degree, second is geodesic distance
par(mfrow=c(3,2))
plot(gf.mle)
plot(gf.mple)
plot(gf.logit)

#### End of Example #####
```