Statistical inference for ERGMs

(Chapter 4 & 5 of the manual)

Alberto Caimo

Dublin Institute of Technology
Ireland

SSNAR 2017
JUNE 21-23, 2017
Birkbeck, University of London

Classical inference for ERGMs

Model specification

- ERGM specification = choice of network statistics $s(y)$ to include in the model;
- The first statistics $s_1(y)$ = number of edges, it's a sort of baseline density effect;
- The other statistics are selected according to the type of effects we are interested in.

For example:

```
library(statnet)
load(url("https://acaimo.github.io/lazega.RData"))
y <- network(Y, directed = FALSE)
model.1.0 <- y ~ edges + kstar(2) + triangle
```

Degeneracy

- To estimate an ERGM we can use the `ergm()` function;
- ATTENTION! Some ERGM specifications lead to a **degenerate** model which may not be estimated

For example:

```
model.1.0 <- y ~ edges + kstar(2) + triangle
ergm(model.1.0)
```

Higher-order network statistics

- In order to try to overcome degeneracy issues, a new specification of network statistics based on geometrically weighted functions of extra-triadic network statistics distributions have been proposed by Snijders et al. (2006);
- This statistics include, for example:
  - geometrically-weighted degree (`gwdegree`) statistic (replacing the stars)
  - geometrically-weighted edgewise shared partner (`gwesp`) statistic (replacing triangles)

ERGMs in action

For example:

```
model.1.1 <- y ~ edges +
                 gwdegree(1, fixed = TRUE) +
                 gwesp(1, fixed = TRUE)
MLE.1.1 <- ergm(model.1.1)
summary(MLE.1.1)

 ## Monte Carlo MLE Results:
 ##               Estimate Std. Error MCMC % p-value
 ## edges          -4.1437     0.5667      0  <1e-04 ***
 ## gwdegree        0.7340     0.4580      0   0.109
 ## gwesp.fixed.1   0.9327     0.1716      0  <1e-04 ***
 ## ---
 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'
 ##
 ## AIC: 524.4    BIC: 537.7    (Smaller is better.)
```

ERGMs in action

Another example with covariate information:

```
# Add attributes to the network object y:
set.vertex.attribute(y, "Office", X$Office)
set.vertex.attribute(y, "Years", X$Years)

model.2 <- y ~ edges +
               nodematch("Office") +
               nodecov("Years") +
               gwesp(0.6, fixed = TRUE) +
               gwdegree(0.6, fixed = TRUE)

MLE.2 <- ergm(model.2)
summary(MLE.2)
```

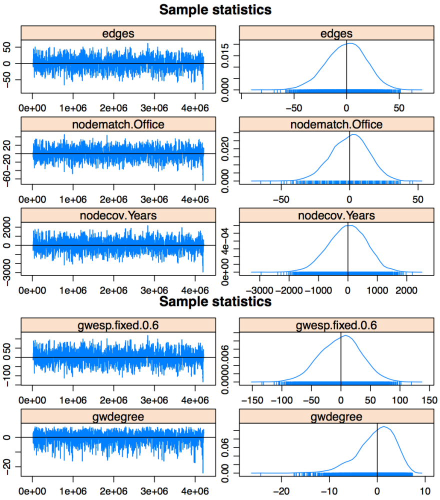
ERGMs in action

Another example with covariate information (cont'd):

```
## Monte Carlo MLE Results:
##                    Estimate Std. Error MCMC % p-value
## edges             -4.309415   0.617740      0  <1e-04 ***
## nodematch.Office   0.878821   0.174202      0  <1e-04 ***
## nodecov.Years     -0.013317   0.005855      0  0.0233 *
## gwesp.fixed.0.6    1.385871   0.276452      0  <1e-04 ***
## gwdegree           0.360304   0.619493      0  0.5610
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'
##
## AIC: 500.6    BIC: 522.9    (Smaller is better.)
```

Output diagnostics

```
mcmc.diagnostics(MLE.2)
```



Goodness of fit diagnostics

- If the estimated ERGM is a good fit to the observed data, then networks simulated $y_1, \ldots, y_m$ should resemble the connectivity structure of the observed data y.
- To do this, M graphs are simulated from the MLE of the parameter $\hat{\theta}$ and compared to the observed graph in terms of high-level network statistics $g(y)$ which are not modelled explicitly.
- We expect that:

$$E(g(\tilde{y})) = \frac{1}{M} \sum_{i=1}^{M} g(\tilde{y_i}) \approx g(y);$$

Goodness of fit diagnostics

```
par(mfrow = c(2, 2))
plot(model.2.gof, main = '')
```