Bergm

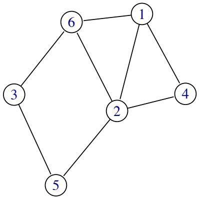
Brief Intro - Basic functions

Alberto Caimo
acaimo.github.io

Definitions and notation

- Networks are generally represented by graphs of nodes (actors) and edges (relations)
- N number of nodes (**fixed**)
- Y **random** $N \times N$ adjacency matrix where:
  - $Y_{ij} = 1$, if i and j are connected
  - $Y_{ij} = 0$, if i and j are not connected
- y realisation of Y

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \quad \leftsquigarrow$$



Exponential random graph models (ERGMs)

Basic assumptions

- The observed network y is generated by a stochastic process in which edges are created because of the presence or absence of other edges.
- Local effects are represented by **network statistics** $s(y)$ that generate dyadic relations and may depend on the surrounding environment.

For example:

- Similar attributes are generally more likely to form friendship edges (**homophily**)
- Two nodes connected to a third node are likely to form an edge between them (**clustering**)

Exponential random graph models (ERGMs)

Definition

$$\Pr(Y = y) = \frac{\exp\{\theta^T s(y)\}}{c(\theta)}$$

- Describe the probability of y as a function of the parameter θ;
- $s(y)$ is a vector of network statistics (e.g. number of edges, number of triangles, etc.) associated to effects of interest;
- θ is the vector of parameters associated to the network statistics $s(y)$;
- $c(\theta)$ is a normalising constant which is **intractable** for not trivially small networks.

Bayesian inference for ERGMs with Bergm

Bayesian inference for ERGMs is based on the posterior distribution of θ given the data y:

$$p(\theta|y) = \frac{\exp\{\theta^t s(y)\}}{c(\theta)} \; \frac{p(\theta)}{p(y)},$$

where:

- $p(\theta)$ is the prior distribution of the parameter
- $p(y)$ is the marginal likelihood of y which is typically **intractable**.

Bayesian inference for ERGMs with Bergm

- Bergm is an R package which provides a comprehensive framework for Bayesian estimation and model adequacy for ERGMs using advanced Monte Carlo algorithms (see, e.g., Caimo and Friel, 2011).
- Bergm is based on the statnet suite of packages (Handcock et al., 2007) and makes use of the same model set-up and network simulation procedures.

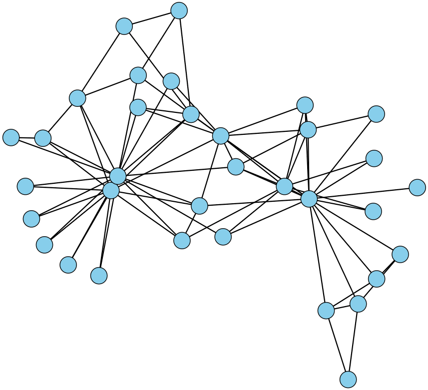
The Bergm package

Main functions

- `bergm()`: function to fit Bayesian exponential random graphs models using the approximate exchange algorithm
- `bergm.output()`: function to return the posterior parameter density estimate and creates simple diagnostic plots for the MCMC produced from a fit.
- `bgof()`: function to calculate summaries for degree, minimum geodesic distances, and edgewise shared partner distributions to diagnose Bayesian goodness-of-fit.

Example - Bergm in action!

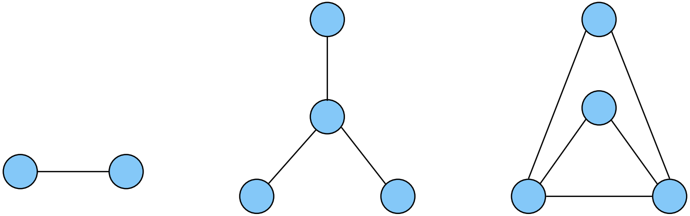
```
library(statnet)
data(zach) # 34 interacting members of a karate club
y <- zach
plot(y, vertex.col = "skyblue", vertex.cex = 2)
```



Example - Bergm in action!

ERGM specification:

$$\Pr(Y = y) \propto \exp\{\theta_1 \text{ edges}(y) + \theta_2 \text{ gwdegree}(y) + \theta_3 \text{ gwesp}(y)\}$$



edges   degrees   edgewise shared partners

```
library(Bergm)

model <- y ~ edges +
             gwdegree(0.2, fixed = TRUE) +
             gwesp(0.2, fixed = TRUE)
```

Example - Bergm in action!

Prior specification:

- vague normal prior distribution with no correlation between the parameters;
- we assume low density of the network: $\theta_1 < 0$;

So, for example:

$$\theta \sim N\left(\begin{bmatrix}-2\\0\\0\end{bmatrix}, \begin{bmatrix}5 & 0 & 0\\0 & 5 & 0\\0 & 0 & 5\end{bmatrix}\right)$$

```
mean.prior <- c(-2, 0, 0)
sigma.prior <- diag(5, 3)
```

Example - Bergm in action!

The **approximate exchange algorithm** (`bergm()` function):

---

for `main.iters` $\times$ `n.chains` iterations:

1. simulate θ' from $\epsilon(\cdot|\theta, \texttt{gamma})$
2. simulate y' from $f(\cdot|\theta')$ using `aux.iters` iterations
3. update $\theta \to \theta'$ with the log of the probability:

$$\min\left(0,\ [\theta - \theta']^t \, [s(y') - s(y)] + \log \frac{p(\theta')}{p(\theta)}\right)$$

end for

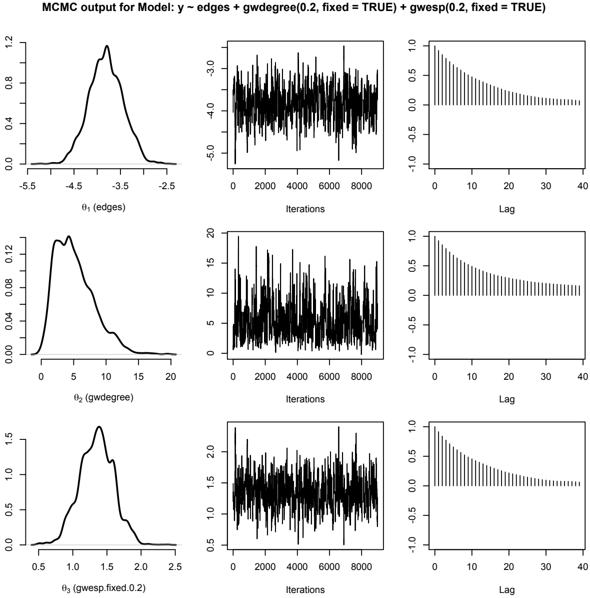
Example - Bergm in action!

```
posterior <- bergm(
    model,
    main.iters = 1500,
    n.chains = 6, # 6×1.5k = 9k iterations
    aux.iters = 3000, # for network simulation
    gamma = 0.9, # tuned to get ~20% acceptance rate
    mean.prior = mean.prior,
    sigma.prior = sigma.prior)

bergm.output(posterior)
```

```
## Posterior Density Estimate for Model:
## y ~ edges + gwdegree(0.2, fixed = TRUE) + gwesp(0.2, fixed = TRUE)
##
##                         Mean        SD    Naive SE Time-series SE
## theta1 (edges)     -3.810812 0.3650827 0.003848309     0.01942382
## theta2 (gwdegree)   5.169665 3.0124907 0.031754440     0.16939821
## theta3 (gwesp)      1.347093 0.2483486 0.002617824     0.01265695
##
##                         2.5%       25%       50%       75%     97.5%
## theta1 (edges)     -4.501821 -4.053643 -3.811498 -3.565977 -3.107281
## theta2 (gwdegree)   0.913240  2.875273  4.633890  7.005652 12.135357
## theta3 (gwesp)      0.874649  1.178549  1.353660  1.514104  1.830226
##
## Acceptance rate: 0.201
```

Example - Bergm in action!



Example - Bergm in action!

Bayesian GoF diagnostics (`bgof()` function):

1. Draws `sample.size` parameters $\{\tilde{\theta}_1, \tilde{\theta}_2, \cdots\}$ from the estimated posterior
2. Simulate networks $\{\tilde{y}_1, \tilde{y}_2, \cdots\}$ from each $\{\tilde{\theta}_1, \tilde{\theta}_2, \cdots\}$ using `aux.iters` iterations
3. Compare some network statistics distributions of the $\{\tilde{y}_1, \tilde{y}_2, \cdots\}$ to the observed network y

Example - Bergm in action!

```
bgof(posterior, sample.size = 100, aux.iters = 3000,
     n.deg = 18, n.dist = 10, n.esp = 8)
```