Bref aperçu des modèles ERGM (avec R)

16 février 2021

François Briatte

1/ Pourquoi

Pourquoi mesurer/modéliser des réseaux

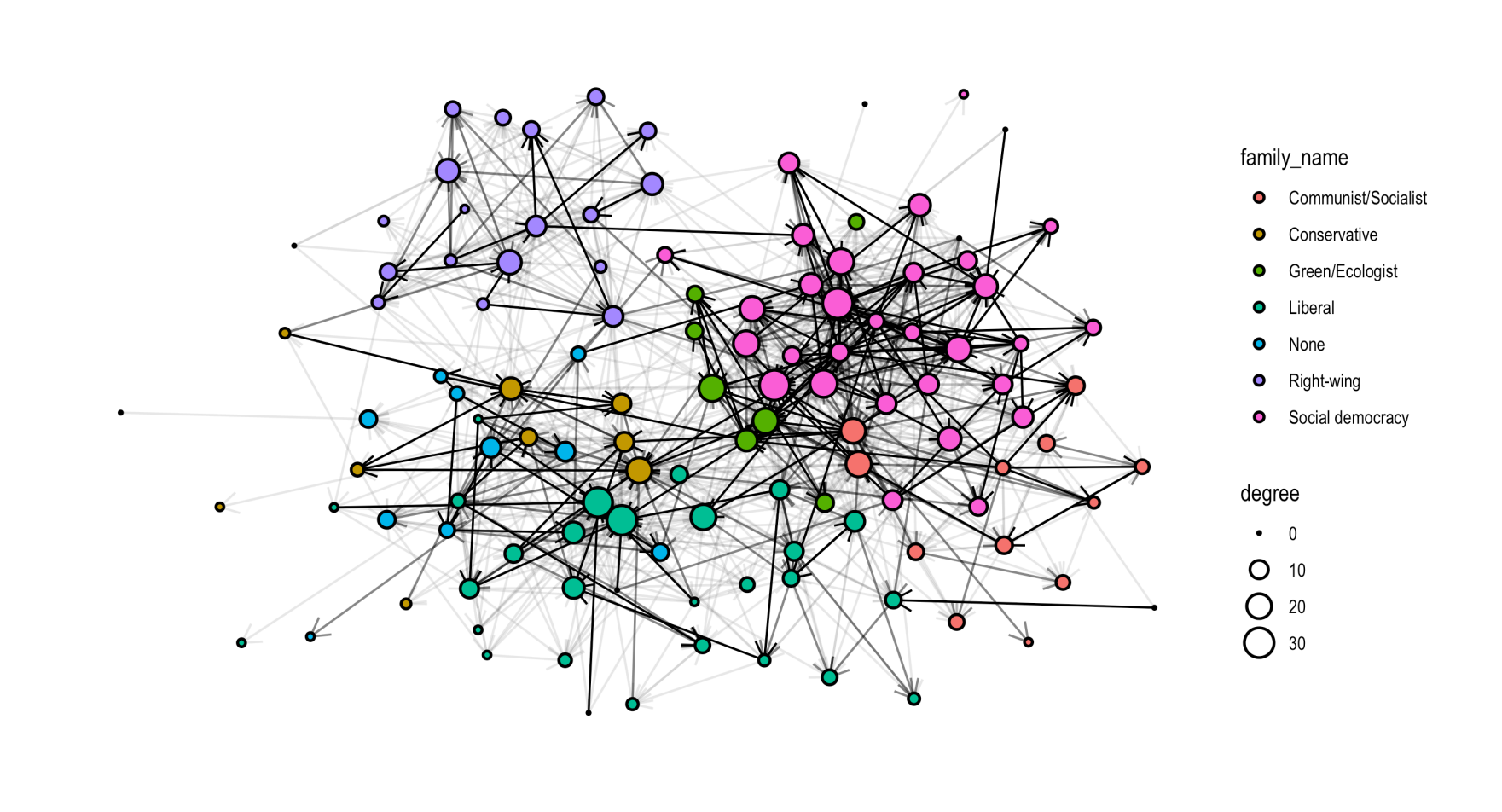
Objectifs

- S'échapper de "Flatland"
- Produire des mesures conçues en conséquence
- Moins de faux positifs

Prix à payer

- Requiert des modèles précis (ou au moins parcimonieux)
- Les termes structurels sont compliqués à théoriser
- Aspects computationnels
- Moins de faux positifs

Exemple


family_name
Communist/Socialist
Conservative
Green/Ecologist
Liberal
None
Right-wing
Social democracy
degree
0
10
20
30

Hypothèses de dépendance

Bernoulli

- liens indépendants les uns des autres
- la densité du réseau reflète la probabilité générale de réalisation

Dépendance dyadique

- $i \to j$ influence $j \to i$

Dépendance de Markov

- ij influence ik (i en commun)
- jk dépend aussi de ij, jk

Circuit social

- ij et kl sont dépendants s'ils partagent suffisamment de tiers

Comment s'y prendre

ERGMs avec R : `ergm`, suite `statnet`

- *Pro* – nouveautés, documentation et tutoriels
- *Con* – changements fréquents et non rétro-compatibles

À garder à l'esprit

- Difficultés de paramétrage et d'estimation
- Nombre limité d'applications à des données réelles
  - Difficultés croissantes avec la taille (en nœuds) des réseaux
  - Comparaison difficile d'un réseau à l'autre

Alternatives

- Régression dyadique
- Autres modèles de réseaux (notamment SAOM)

2/ Comment

Principes de base

Variable aléatoire

- La variable dépendante est le réseau observé $G = (V, E)$
- Possible prise en compte de la pondération (et du signe) des liens

Les prédicteurs sont des attentes sur les relations E_{ij}

- homophilie : i et j partagent des attributs
- triades : i, j et k forment des “triangles”

Trois types de prédicteurs

- *Node/vertex-level* – attributs nodaux de $i, j, k, \ldots$
- *Tie/edge-level* – attributs des liens E (hors poids)
- *Network-level* – propriétés structurelles : nb. de liens E, de “triangles”

Quelques remarques

Deux manières de voir les choses

On peut choisir de s'intéresser...

- aux "effets-réseau", en contrôlant les attributs nodaux
- aux attributs nodaux, en contrôlant les propriétés structurelles

Pour aller *moins* loin

- Modèles p^* (ancêtres des ERGMs)
- Endogénéisation limitée des propriétés structurelles

Pour aller *plus* loin

- Modèles par espace latent (LSM)
- Capture d'interdépendances encore plus compliquées

Forme générale du modèle

$$P(Y = y) = \frac{\exp(\theta' g(y))}{k(\theta)}$$

- Y : liens du réseau $A_{ij} = N \times N$
  - réalisation y peut être binaire, signée, valuée
- θ : paramètres du modèle
  - coefficients
- $g(y)$: caractéristiques du réseau
  - prédicteurs (termes) : liens, homophilies, triangles etc.
- $k(\theta)$: constante de normalisation
  - somme du numérateur dans tous les réseaux $N \times N$ réalisables

Détails

Termes p du modèle

$$\log(\exp(\theta' g(y))) = \theta_1 g_1(y) + \theta_2 g_2(y) + \ldots + \theta_p g_p(y)$$

Le numérateur est la partie “facile” — à interpréter : les coefficients sont linéaires dans cette forme, ce qui fait qu’on peut lire les coefficients comme dans une régression logistique : $P(Y = y) \sim P(A_{ij} = 1)$.

Constante de normalisation

$$k(\theta, y) = \sum_{z \in \mathcal{Y}} \theta' g(z)$$

Le dénominateur constitue la partie difficile — à estimer : il faut simuler des réseaux de taille N (...) en espérant qu’ils convergent (...) sans trop de problèmes (...) vers des résultats qui ressemblent à $Y = y$.

Estimation par MCMLE

L'objectif est de maximiser la vraisemblance de θ :

$$\theta^* = \arg\max_{\theta} \mathrm{P}_{\theta,y}(Y = y)$$

Markov Chain Maximum Likelihood Estimation (MCMLE)

1. Commencer avec une valeur de départ θ_0 et un graphe de départ, qui est généralement vide.
2. Obtenir un espace de graphes $\mathrm{P}_{\theta_0,y}(Y)$ via MCMC.
3. Après convergence, échantillonner $\tilde{Y}$ de cet espace ; voir si $\tilde{Y}$ s'approche de Y, et sélectionner θ^+ pour maximiser cette vraisemblance.
4. Recommencer en remplaçant θ_0 par θ^+.

Échantillonnage par MCMC

Algorithme de Metropolis-Hastings

1. Partir d'un graphe (généralement vide)
2. Choisir aléatoirement un lien y_{ij} et l'inverser (0/1)
3. Comparer le nouveau graphe à l'ancien, et garder le plus probable
4. Recommencer

Paramétrage

- *Burn-in* — éliminer les n premières itérations de la chaîne
- *Thinning* — éliminer k itérations entre deux graphes aléatoires

Problème

Aucune garantie de convergence, et même en cas de convergence, le résultat peut être très différent du graphe observé.

Pourquoi pas de convergence garantie ?

Problème

- La solution (l'espace de graphes de taille N) est finie
- La *taille* de la solution est au-delà de ce qui est calculable

Approximation

- Maximum pseudolikelihood (MPLE)
  - sous-estimation des erreurs standard
  - corrigible dans des réseaux longitudinaux
- MCMC maximum likelihood (MCMC-MLE)
  - *cf. slides précédentes*
  - Problème : que signifie $N \to \infty$ dans ce contexte ?

3/ Comment (dans **R**)

Implémentation : ergm

```r
library(ergm)

class(observed_network)
# "network"

ergm(
  # formula
  observed_network ~ network_term_1 + ... + network_term_k,
  # estimation parameters
  control = control.ergm(seed = 1234, ...)
)
```

```r
# help
?ergm::`ergm-terms`
?ergm::control.ergm
```

Réseaux binaires

```
library(ergm)

ergm(
  observed_network ~
    # pseudo-intercept
    edges +
    # node-level covariates
    # edge-level covariates
    # network-level covariates
    ...,
  control = control.ergm(seed = 1234, ...)
)
```

Réseaux valués : ergm.count

```
library(ergm)
library(ergm.count)

ergm(
  observed_network ~
    # pseudo-intercept
    sum +
    # covariates
    ...,
  response = "count",
  # prob. distribution
  reference = ~ Bernoulli(3),
  control = control.ergm(seed = 1234, ...)
)
```

Contrôle de l'estimateur

```
# example for a weighted network from an old paper
control.ergm(
  seed = 1234,
  MCMC.prop.weights = "0inflated",
  MCMLE.trustregion = 1000, # default: 20
  MCMC.samplesize = 50000,  # default: 1024
  MCMC.burnin = 50000,      # default: 16384
  MCMC.interval = 3000,     # default: 1024
  MCMLE.maxit = 50,         # default: 20
  # kill your laptop
  parallel = parallel::detectCores()
)
```

Exemple de terme : réciprocité

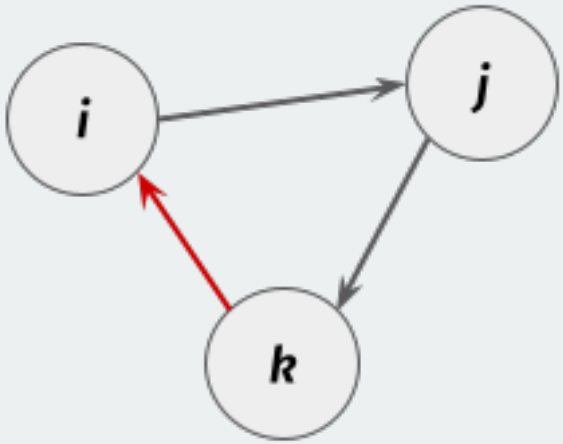
Expression (réseau binaire)

$g_R(N) = \sum_{i<j} N_{ij} N_{ji}$

Implémentation

```
# for binary responses
mutual()
# for valued responses
mutual(form = "min", threshold = 0)
```

Exemple de terme : cyclicalité



Expression (réseau binaire)

$$g_C(N) = \sum_{i,j \neq i,k} N_{ij} N_{jk} N_{ki}$$

Implémentation

```
# for binary responses
cyclicalties()
# for valued responses
cyclicalties(threshold = 0)
cyclicalweights()
```

Termes courants

Effet “principal” d’un attribut nodal x

$$\sum_{i \neq j} y_{ij}(x_i + x_j)$$

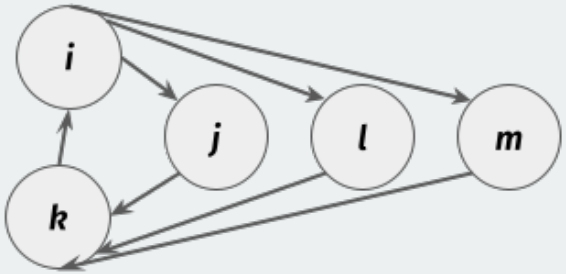
```
# for binary responses
nodefactor()
```

Homophilie liée à un attribut modal x

$$\sum_{i \neq j} y_{ij}(x_i = x_j)$$

```
nodematch()
```

Termes plus compliqués



Geometrically weighted ...

```
# ... edgewise shared partner distribution (fig. above)
gwesp(decay = 0, fixed = FALSE, cutoff = 30)
# ... dyadwise shared partner distribution
gwdsp()
# ... in-degree distribution
gwidegree()
# ... out-degree distribution
gwodegree()
```

Là où ça se complique *vraiment*

Comprendre les termes

- Quelques termes interprétables comme des coefficients de régression
- Beaucoup de termes n'ont strictement aucun équivalent

Paramétrer (et recalibrer) les termes

- Les options de certains termes peuvent faire "exploser" l'estimateur
- Beaucoup de termes n'ont strictement aucun équivalent

Complications liées aux données

- De nombreux termes ne permettent pas d'avoir des valeurs manquantes
- Problèmes habituels de validité externe des réseaux observés

Exemple d'échec à l'estimation

Starting maximum pseudolikelihood estimation (MPLE):
Evaluating the predictor and response matrix.
Maximizing the pseudolikelihood.
Finished MPLE.
Starting Monte Carlo maximum likelihood estimation (MCMLE):
Iteration 1 of at most 20:
Optimizing with step length 0.471170896026408.
The log-likelihood improved by 13.62.
Iteration 2 of at most 20:
Error in ergm.MCMLE(init, nw, model, initialfit = (initialfit <- NULL), :
Unconstrained MCMC sampling did not mix at all. Optimization cannot continue.

Autre exemple d'échec (collinéarité)

Iteration 14 of at most 20:
Warning: Model statistics 'transitiveties' are linear combinations of some set of preceding statistics at the current stage of the estimation. This may indicate that the model is nonidentifiable.
Optimizing with step length 0.
The log-likelihood improved by < 0.0001.
Error in ergm.MCMLE(init, nw, model, initialfit = (initialfit <- NULL), :
MCMLE estimation stuck. There may be excessive correlation between model terms, suggesting a poor model for the observed data. If target.stats are specified, try increasing SAN parameters.

Le message qui rend triste

> *Error: Number of edges in a simulated network exceeds that in the observed by a factor of more than 20.* ***This is a strong indicator of model degeneracy.*** *If you are reasonably certain that this is not the case, increase the MCMLE.density.guard control.ergm() parameter.*

Causes possibles

- Réseau observé très faiblement dense (*sparsity*)
- Estimateur réellement trop conservateur par défaut

Solutions possibles

- Éviter les termes les plus susceptibles d'en arriver là...
  - *looking at you, stars and triangles*
- Recalibrer
  - Fixer les paramètres des termes sur des valeurs plus stables
  - Reparamétrer l'estimateur (*slide suivante*)

Plus généralement, concernant l'estimation

Si les diagnostics montrent des problèmes, ...

- Convergence des chaînes de Markov : *traceplots*
- GOF (*goodness-of-fit*) : morphologie des réseaux simulés

... laisser du temps au MCMC, ...

- Augmenter les itérations
- Augmenter le *burn-in* (itérations initiales)
- Augmenter le *sample size* (temps de convergence)

N.B. S'applique aussi aux simulations (GOF) !

... et si rien ne change, ...

- *Rajouter* des termes (de contrôle)
- *Remplacer* des termes (par des variantes)
- Tolérer certaines différences

4/ Conclusion

Quelques problèmes finaux

Stabilité

- Adéquation concepts/termes
- Implémentations (*packages*)
- Réseaux plus voire très larges
- Estimateurs

Diversification

- Familles de modèles
- Données (ex. textuelles)

Réductions structurelles

- *Backbone extraction*

Quelques extensions

Réseaux valués

- `ergm.count` (réduction vers Bernoulli ou Poisson)
- `GERGM` (réduction vers un modèle linéaire)

Autres types de réseaux

- Bipartites (implémentés dans `ergm`)
- Multi-niveaux (`mlergm`, `hergm`) ; généralisation : `multilayer.ergm`
- Temporels (TERGM) : `tergm`, modèles proches des SAOM
- Relational Event Models (REM) : `rem`, `relevent`

Autres estimateurs

- `Bergm` : estimateur bayésien ($P(\theta|y) = \frac{P(y|\theta)P(\theta)}{P(y)}$)
  - converge mieux
  - peut gérer les données manquantes — *cf.* `Bergm::bergmM`
- `btergm` : bootstrapped TERGM
- `fergm` : Frailty ERGM, avec effets aléatoires

Lectures conseillées

statnet.org

- Mailing-list `[statnet_help]`
- Sunbelt workshops

Ouvrages

- Cranmer, Desmarais & Morgan, *Inferential Network Analysis*, 2021
- Lusher, Koskinen, Robins, *ERGMs for Social Networks*, 2012

Articles

- Brailly *et al.*, *Ann. Soc.*, 2017
- Cranmer *et al.*, *Am. J. Pol. Sci*, 2016
- Goodreau *et al.*, *Demography*, 2009
- Salter-Townshend *et al.*, *Stat. An. and Data Mining*, 2012

Merci pour votre attention

Slides et exemple de code

- `f.briatte.org/temp/ap3s`

Inspiration

- Zack Almquist, 2017
- Romain Ferrali, 2017
- Thomas Grund, 2016
- Matthew Jackson, 2017
- Lisa Lechner, 2020