

ERPM — Avancement Wrapper Et Nouveaux Effets

Jérémie Chichignoud (CUB'iTECH)

4 novembre 2025

Table des matières

Objectifs (portée du document)	3
Travail réalisé	3
Travail restant et perspectives	4
Tableau de correspondance des effets	5
Explication effet par effet	7
1 Comptage de groupes par taille : <code>groups(from, to)</code>	7
2 Somme des puissances de tailles : <code>squared_sizes(from, to, pow)</code>	7
3 k-cliques sur la projection 1-mode : <code>cliques(k, normalized)</code>	8
4 Cliques géométriquement pondérées : <code>cliques_GW(λ)</code>	8
Annexe — ERGM, Metropolis-Hastings, vraisemblance et critères d'ajustement	11
A.1 Rappel : qu'est-ce qu'un ERGM ?	11
A.2 Log-vraisemblance, déviance, AIC et BIC	12
A.3 Sorties usuelles de <code>summary(ergm)</code>	13
A.4 Contraintes et mécanisme de proposition (MH) en biparti	13
A.5 Exemple minimal (R)	13

A.6	Schéma du processus d'estimation	14
A.7	Points d'attention	14

État d'avancement : wrapper `erpm` et statistiques/effets

Objectifs (portée du document)

L'objectif est de concevoir une fonction `erpm()` inspirée de la logique des modèles linéaires généralisés, capable d'estimer des effets statistiques définis sur des **partitions** plutôt que sur des graphes. La syntaxe cible est :

```
erpm(partition ~ effets + options)
```

La partie gauche représente la partition des acteurs en groupes, et la partie droite décrit les effets statistiques à calculer.

La fonction agit comme un **wrapper** autour du moteur `ergm()`. Elle transforme la partition en un **graphe biparti** (acteurs-groupes), sur lequel `ergm()` peut être appliqué directement. Cette traduction permet d'utiliser les outils et algorithmes d'ERGM tout en conservant une lecture centrée sur les groupes.

Un autre rôle clé de `erpm()` est d'assurer la **correspondance entre effets ERPM et ERGM**, selon trois situations :

1. **Effet direct** : déjà présent dans ERGM et utilisable tel quel ;
2. **Effet partiel** : partiellement couvert, nécessitant un traitement complémentaire (agrégation, normalisation, etc.) ;
3. **Effet spécifique** : absent d'ERGM, à créer avec `InitErgmTerm()` et une fonction C de changement (`change_stat`).

L'objectif final est que `erpm()` pilote l'ensemble du processus : de la définition de la partition à l'appel de `ergm()`, en passant par la préparation des effets, des contraintes et des propositions.

Travail réalisé

La structure principale de `erpm()` est **fonctionnelle et stable**. Elle gère correctement la conversion des entrées, le renommage des effets et l'appel au moteur `ergm()`.

- **Structure du wrapper** : prise en charge d'une partition et d'une formule d'effets ; construction de la matrice d'affiliation (acteurs \times groupes) ; création du **network biparti** ; ajout automatique des contraintes et propositions nécessaires (`InitErgmConstraint.b1part`, `InitErgmProposal.B1Part`) ; puis exécution de `ergm()`.
- **Traduction des effets** : les effets connus sont traduits vers leurs équivalents ERGM (par ex. `groups` \rightarrow `b2degrange`). Les effets dyadiques utilisent les opérateurs `Proj1` ou `B` selon le mode d'analyse.
- **Effets implémentés** : un effet spécifique, `squared_sizes`, est pleinement fonctionnel. Il dispose de sa fonction `InitErgmTerm()` et de son code C associé, basé sur les macros d'ERGM (`ergm_changestat_common.do_not_include_directly.h`, `ergm_changestat.h`, etc.).
- **Tests et validation** : les scripts `squared_sizes.R` et `test_groups.R` vérifient la cohérence du pipeline complet : création de partitions simples, conversion bipartie, appel à `erpm()`, et calcul des statistiques. Ces tests sont conçus comme des *selftests* autonomes.

Les résultats montrent que l'intégration avec `ergm()` fonctionne correctement et que la structure du wrapper est fiable.

Travail restant et perspectives

La prochaine étape est la **caractérisation complète des effets ERPM**. Chaque effet doit être identifié comme direct, partiel ou spécifique, et implémenté en conséquence, avec un test unitaire dédié.

- **Finalisation du wrapper** : améliorer la gestion des options `control.ergm`, définir des valeurs par défaut cohérentes, traiter les cas limites (groupes vides, partitions incomplètes), et renforcer la traduction automatique des effets directs `erpm` → `ergm`.
- **Validation des effets et tests d'intégration** : ajouter un *selftest* autonome pour chaque nouvel effet, et construire des cas minimaux pour vérifier

la stabilité statistique (moyennes, variances, convergence).

- **Effets liés aux attributs dyadiques** : une partition ne comporte pas d'attributs dyadiques, mais il peut être utile de modéliser la probabilité d'appartenance à un groupe à partir d'informations dyadiques externes. Dans ce cas, une matrice d'attributs dyadiques peut être transmise en complément de la partition.
- **Documentation** : compléter les blocs `roxygen2` et rédiger une vignette “`erpm` en dix lignes” illustrant le flux complet : partition → graphe biparti → estimation → résumé.

En résumé, `erpm()` est stable, bien intégré à `ergm()`, et prêt à accueillir progressivement de nouveaux effets ERPM.

Tableau de correspondance des effets

Effet ERPM	Correspondance ERGM	Alias/Termes ERGM	Calcul partiel (si partielle)	Si aucune : InitErgmTerm + C	Commentaire	Status
<code>groups(from,to)</code>	Directe	<code>b2degrange(from,to)</code>	—	—	Taille des groupes = degré (mode 2).	✓
<code>squared_sizes(from,to,pow)</code>	ErgmTerm	—	—	<code>InitErgmTerm.squared_sizes c_squared_sizes</code>	Somme des tailles des groupes (mode 2) élevées à la puissance <code>pow</code> .	✓
<code>cov_match(cov)</code>	Directe via projection	<code>Proj1(~nodematch("cov"))</code>	—	—	Homophilie intra-groupe (projection mode 1).	✗
<code>cov_diff(cov)</code>	Directe via projection	<code>Proj1(~absdiff("cov"))</code>	—	—	Définition des différences intra-groupe (paires).	✗
<code>dyadcov(X)</code>	Directe via projection	<code>Proj1(~edgecov(X))</code>	—	—	Effets dyadiques sur la projection. <i>Optionnel</i> : <code>B(form="nonzero")</code> si nécessaire.	✗
<code>cov_fullmatch(cov)</code>	Partielle	<code>Proj1</code> + comptage	Post-traitement : groupes 100% homogènes.	<i>(optionnel)</i> <code>InitErgmTerm.cov_fullmatch c_cov_fullmatch</code>	Post-hoc OK ; term C si nécessaire en MCMC.	✗
<code>cliques(cliique_size,normalized)</code>	Partielle	<code>b2degrange(0,1)</code> ou <code>edges</code>	Normalisation optionnelle par la proportion de paires d'acteurs co-membres : $\frac{\sum_g \binom{n_g}{2}}{\binom{N_1}{2}}$	—	Compte, pour chaque groupe du mode 2, le nombre de k -cliques d'acteurs qu'il induit dans la projection 1-mode ; chaque groupe de taille n_g contribue $\binom{n_g}{k}$.	✓
<code>cliques_GW(\$\lambda\$)</code>	Pas encore évaluée	—	—	<code>InitErgmTerm.cliques_GW c_cliques_GW</code>	Pondération géométrique (probable custom).	✗

Effet ERPM	Correspondance ERGM	Alias/Termes ERGM	Calcul partiel (si partielle)	Si aucune : InitErgmTerm + C	Commentaire	Status
<code>log_factorial_sizes</code>	Directe (custom)	<code>b2degm1lfactorial</code> ↔ <code>sum_lfactorial_idegreem1</code>	—	<code>InitErgmTerm.b2degm1lfactorial</code> <code>c_sum_lfactorial_idegreem1</code>	C OK (<code>lgammafn</code>).	X
<code>inertia_longitudinal</code>	Pas encore évaluée	—	—	<code>InitErgmTerm.inertia</code> <code>c_inertia</code>	Effet temporel avec auxiliaires ($t-1$).	X
<code>dyadcov_intragroup_sum</code>	Partielle	<code>Proj1(~edgecov)</code> + agrégations	Sommes/normalisations par groupe (R).	(<i>optionnel</i>) term dédié si requis	Pour métriques par groupe dans la vraisemblance.	X
<code>range_attribute(attr)</code>	Partielle	<code>Proj1 + absdiff</code> /résumés	Max-min par groupe (post-traitement).	(<i>optionnel</i>) term dédié	Contraintes d'hétérogénéité intra-groupe.	X

Explication effet par effet

1 Comptage de groupes par taille : `groups(from, to)`

Définition.

Pour le biparti $A-G$ et $n_g = d(g)$ la taille du groupe g ,

$$\text{groups}(\text{from}, \text{to}) = \sum_{g \in G} \mathbf{1}\{n_g \in [\text{from}, \text{to}]\},$$

Cela compte le nombre de groupes dont la taille appartient à l'intervalle $[\text{from}, \text{to}]$ (borne inférieure incluse, supérieure exclue).

Correspondance ERGM.

`groups(from,to)` —> `b2degrange(from=from, to=to)` sur le mode 2.

Appel dans `erpml()`.

```
1 # Tous les groupes de taille ≥1
2 erpm(nw ~ groups)
3
4 # Groupes de taille exactement 3  -> [3,4)
5 erpm(nw ~ groups(3))
6
7 # Intervalle explicite [2,5)
8 erpm(nw ~ groups(from = 2, to = 5))
```

2 Somme des puissances de tailles : `squared_sizes(from, to, pow)`

Définition.

Sur les noeuds groupes $g \in G$ de taille n_g ,

$$\text{squared_sizes}(\text{from}, \text{to}, \text{pow}) = \sum_{g \in G} \mathbf{1}\{n_g \in [\text{from}, \text{to}]\} n_g^{\text{pow}},$$

Par défaut $\text{pow} = 2$. Le terme est vectorisable sur plusieurs intervalles et puissances.

Implémentation ERGM dédiée.

Terme spécifique `squared_sizes` côté `{ergm}`, calcul en « un-toggle » sur le mode 2 uniquement, avec entrées $[\text{from}, \text{to}, \text{pow}]$ et gestion de $[\text{from}, \text{to}]$ ($\text{to} = \infty \Rightarrow$ remplacé par $n + 1$).

Appel dans `erpml()`.

```
1 # Somme des carrés pour tous les groupes (from=1, to=Inf, pow
2 # =2)
3 erpm(nw ~ squared_sizes)
4
5 # Somme des cubes pour tailles [2,5)
6 erpm(nw ~ squared_sizes(from = 2, to = 5, pow = 3))
7
8 # Vectorisation sur deux intervalles
9 erpm(nw ~ squared_sizes(from = c(1,4), to = c(4, Inf), pow = c
10      (2,2)))
```

3 k-cliques sur la projection 1-mode : `cliques(k, normalized)`

Cliques d'acteurs via la projection 1-mode.

Pour une partition P , on construit le graphe biparti $A-G$ où A est l'ensemble des acteurs et G l'ensemble des groupes. Pour tout $g \in G$, on note $n_g = d(g)$ le degré (i.e. la taille) du groupe g . On note $N = \{n_g : g \in G\}$ l'ensemble des tailles de groupes.

La projection 1-mode sur A relie deux acteurs s'ils sont co-membres d'au moins un groupe. Le nombre total de k -cliques d'acteurs dans cette projection vaut :

$$\#\text{k-cliques} = \sum_{g \in G} \binom{n_g}{k},$$

où l'on adopte la **convention combinatoire étendue par zéro** :

$$\binom{n}{k} = \begin{cases} \frac{n!}{k!(n-k)!}, & \text{si } n \geq k \geq 0, \\ 0, & \text{sinon.} \end{cases}$$

Ainsi, les groupes de taille $n_g < k$ ne contribuent pas au comptage. S'il n'existe aucun groupe de taille au moins k , alors $\#\text{k-cliques} = 0$.

Exemple. Pour la partition $P = \{1, 1, 1, 1, 2, 2, 3\}$, les tailles de groupes sont $N = \{4, 2, 1\}$. On obtient :

$$\#\text{2-cliques} = \binom{4}{2} + \binom{2}{2} + \binom{1}{2} = 6 + 1 + 0 = 7,$$

$$\#\text{4-cliques} = \binom{4}{4} + \binom{2}{4} + \binom{1}{4} = 1 + 0 + 0 = 1,$$

Ainsi, l'ensemble des tailles de cliques présentes est $K = \{2, 4\}$.

Appel dans `erpml()`.

```

1 # k par défaut = 2, non normalisé
2 erpm(nw ~ cliques())
3
4 # k explicite
5 erpm(nw ~ cliques(k = 3))
6
7 # Forme abrégée
8 erpm(nw ~ cliques(4))
9
10 # Normalisation par C(N1, k)
11 erpm(nw ~ cliques(k = 2, normalized = TRUE))
```

4 Cliques géométriquement pondérées : `cliques_GW(λ)`

Définition combinatoire.

Pour une partition P et le graphe biparti $A-G$ associé, chaque groupe $g \in G$ de taille $n_g = d(g)$ génère, dans la projection 1-mode sur A , un ensemble de cliques d'acteurs :

$$c_k(P) = \sum_{g \in G} \binom{n_g}{k}, \quad k \geq 1,$$

Ces $c_k(P)$ comptent le nombre total de k -cliques d'acteurs issus de la coappartenance à un même groupe (voir 3).

L'effet `cliques_GW(lambda)` combine ces contributions pour toutes les tailles

$k \geq 1$ à l'aide d'une série géométrique à décroissance contrôlée par λ :

$$\begin{aligned} \text{cliques_GW}(\lambda) &= \sum_{k \geq 1} \left(-\frac{1}{\lambda} \right)^{k-1} c_k(P) \\ \text{avec } c_k(P) &= \sum_{g \in G} \binom{n_g}{k} \quad \Rightarrow \quad \sum_{k \geq 1} \left(-\frac{1}{\lambda} \right)^{k-1} \sum_{g \in G} \binom{n_g}{k} \\ (\text{sommes finies : } \#G < \infty, n_g < \infty) &\Rightarrow \sum_{g \in G} \sum_{k \geq 1} \left(-\frac{1}{\lambda} \right)^{k-1} \binom{n_g}{k} \\ (\text{convention } \binom{n}{k} = 0 \text{ pour } k > n) &\Rightarrow \sum_{g \in G} \sum_{k=1}^{n_g} \left(-\frac{1}{\lambda} \right)^{k-1} \binom{n_g}{k}, \end{aligned}$$

On introduit alors la contribution individuelle d'un groupe g :

$$S(n_g, \lambda) = \sum_{k=1}^{n_g} \left(-\frac{1}{\lambda} \right)^{k-1} \binom{n_g}{k},$$

La statistique globale s'écrit alors simplement :

$$\boxed{\text{cliques_GW}(\lambda) = \sum_{g \in G} S(n_g, \lambda),}$$

Expression de $S(n_g, \lambda)$ sous forme fermée.

On part de la définition combinatoire

$$S(n_g, \lambda) = \sum_{k=1}^{n_g} \left(-\frac{1}{\lambda} \right)^{k-1} \binom{n_g}{k},$$

et l'on cherche une expression explicite sans symbole de somme, dite *forme fermée*. En posant $r = -1/\lambda$, on obtient :

$$S(n_g, \lambda) = \sum_{k=1}^{n_g} r^{k-1} \binom{n_g}{k},$$

On multiplie par r pour aligner les puissances :

$$r S(n_g, \lambda) = \sum_{k=1}^{n_g} r^k \binom{n_g}{k},$$

Puis on complète la somme par le terme $k=0$ que l'on retranche aussitôt :

$$r S(n_g, \lambda) = \sum_{k=0}^{n_g} \binom{n_g}{k} r^k - \binom{n_g}{0} r^0 = \sum_{k=0}^{n_g} \binom{n_g}{k} r^k - 1,$$

On applique ensuite la **formule du binôme de Newton**, $(x+y)^n = \sum_{k=0}^n \binom{n}{k} x^k y^{n-k}$, avec $x = r$ et $y = 1$, d'où $(1+r)^{n_g} = \sum_{k=0}^{n_g} \binom{n_g}{k} r^k$. On en déduit :

$$r S(n_g, \lambda) = (1+r)^{n_g} - 1 \quad \Rightarrow \quad S(n_g, \lambda) = \frac{(1+r)^{n_g} - 1}{r},$$

Enfin, la substitution $r = -1/\lambda$ conduit à la forme fermée recherchée :

$$\boxed{S(n_g, \lambda) = \lambda \left[1 - \left(\frac{\lambda-1}{\lambda} \right)^{n_g} \right],}$$

Par sommation sur tous les groupes $g \in G$, on obtient la statistique globale :

$$\text{cliques_GW}(\lambda) = \sum_{g \in G} S(n_g, \lambda) = \sum_{g \in G} \lambda \left[1 - \left(\frac{\lambda-1}{\lambda} \right)^{n_g} \right],$$

Interprétation.

Chaque groupe g de taille n_g contribue une valeur $S(n_g, \lambda)$ qui agrège l'ensemble de ses cliques potentielles, pondérées par la série géométrique $(-1/\lambda)^{k-1}$. Le terme est donc une *mesure continue de densité de coappartenance*, où λ agit comme un facteur de décroissance. L'effet **cliques_GW**(λ) quantifie, de manière lissée et graduelle, à quel point les acteurs appartiennent ensemble à des groupes communs.

Cas particuliers et limites.

- **Cas $\lambda = 1$** : Les poids deviennent $(-1)^{k-1}$. On obtient $S(n_g, 1) = \sum_{k=1}^{n_g} (-1)^{k-1} \binom{n_g}{k} = 1$. Chaque groupe non vide contribue exactement 1. L'effet `cliques_GW(1)` compte donc simplement le nombre total de groupes, indépendamment de leur taille.
- **Cas $\lambda \rightarrow \infty$** : $(1 - 1/\lambda)^{n_g} \approx 1 - n_g/\lambda$, d'où $S(n_g, \lambda) \approx n_g/\lambda$. Les grandes cliques deviennent négligeables : seule la composante linéaire ($k = 1$) subsiste. L'effet tend alors vers une mesure proportionnelle au nombre total d'acteurs.
- **Cas λ petit (proche de 1)** : Les poids $(-1/\lambda)^{k-1}$ décroissent lentement, les cliques de grande taille sont fortement prises en compte. Le terme devient sensible à la concentration d'acteurs dans des groupes volumineux.

Résumé interprétatif.

Régime de λ	Formule dominante	Interprétation intuitive
$\lambda = 1$	$S(n_g, 1) = 1$	Chaque groupe vaut 1 (compte des groupes)
$\lambda \rightarrow \infty$	$S(n_g, \lambda) \approx n_g/\lambda$	Poids linéaire des acteurs (1-cliques)
λ petit	Série complète sur k	Forte sensibilité aux cliques denses

Appel dans `erpml()`.

```

1 # Forme par défaut : lambda = 2
2 erpml(nw ~ cliques_GW)
3
4 # Lambda explicite
5 erpml(nw ~ cliques_GW(lambda = 3))
6
7 # Vectorisation (plusieurs décroissances)
8 erpml(nw ~ cliques_GW(lambda = c(1.5, 2, 4)))

```

Remarque.

L'effet `cliques_GW` est à la fois un *lissage analytique* du comptage discret des `cliques(k)` et une *pondération géométrique* contrôlant la profondeur des interactions denses. Il fournit un indicateur continu de la cohésion interne des groupes à travers leurs cliques d'acteurs.

Annexe — ERGM et Metropolis–Hastings

A.1 Rappel : qu'est-ce qu'un ERGM ?

Un modèle de graphe aléatoire exponentiel (ERGM, *Exponential Random Graph Model*) définit une loi de probabilité sur l'ensemble des graphes possibles y :

$$\Pr_{\theta}(Y = y) \propto \exp(\theta^T g(y)),$$

où :

- $g(y)$ est le **vecteur de statistiques du graphe** : il regroupe des caractéristiques mesurées sur le réseau, par exemple le *nombre d'arêtes* (statistique dénombrable ou « brute ») et le *degré moyen* (statistique agrégée ou « moyenne »).
- θ est le **vecteur de paramètres du modèle** : chaque composante pondère la propension du réseau à présenter la structure correspondante (par ex. triangles, réciprocité).
- \Pr_{θ} désigne la **distribution de probabilité induite par θ** : c'est la loi selon laquelle un graphe est susceptible d'être généré.

Motivation.

On dispose d'un **graphe observé** y_{obs} (données empiriques). Ce graphe est fixe : on calcule ses statistiques $g(y_{\text{obs}})$ une fois pour toutes. L'objectif est de trouver θ tel que les graphes simulés depuis \Pr_{θ} aient, *en moyenne*, les mêmes statistiques que y_{obs} :

$$\mathbb{E}_{\theta}[g(Y)] \approx g(y_{\text{obs}}),$$

afin de reproduire les régularités structurelles du réseau (densité, distribution des degrés, triangles, homophilie, etc.).

1. Pour deux graphes y et y' sous un même θ :

$$\frac{\Pr_{\theta}(Y = y')}{\Pr_{\theta}(Y = y)} = \exp\left\{ \theta^T [g(y') - g(y)] \right\} = \exp(\Delta), \quad \Delta = \theta^T [g(y') - g(y)],$$

Si $\Delta > 0$, y' est plus probable que y (toutes choses égales par ailleurs).

Principe général.

L'estimation de θ se fait par deux boucles :

- **Boucle interne (MCMC–MH)** : pour un θ fixé, on échantillonne des graphes selon \Pr_{θ} à l'aide de **Metropolis–Hastings (MH)**. Concrètement :

1. partir d'un graphe initial $y^{(0)}$;
2. proposer une modification locale donnant un candidat y' (ajout/suppression d'une arête) ;
3. calculer la variation de log-probabilité ¹ :

$$\Delta = \theta^T [g(y') - g(y)],$$

4. accepter y' avec probabilité $\alpha = \min(1, \exp(\Delta) \times \frac{q(y' \rightarrow y)}{q(y \rightarrow y')})$, où q est la loi de proposition.

En répétant ces étapes, on obtient une chaîne de Markov dont la loi stationnaire est précisément \Pr_{θ} . L'échantillon $\{y^{(s)}\}$, constitué de plusieurs graphes simulés indépendamment (après burn-in et thinning), sert à approximer les moyennes des statistiques $g(y)$ sous la distribution \Pr_{θ} .

- **Boucle externe (mise à jour de θ)** : après simulation, on compare la moyenne simulée $\mathbb{E}_{\theta^{(t)}}[g(Y)]$ à $g(y_{\text{obs}})$ et on ajuste

$$\theta^{(t+1)} = \theta^{(t)} + \alpha \left(g(y_{\text{obs}}) - \frac{1}{S} \sum_{s=1}^S g(y^{(s)}) \right),$$

où α est un **pas d'apprentissage**. Un pas trop grand peut faire diverger l'algorithme ; trop petit, la convergence devient lente.

Lien avec les algorithmes de gradient.

La mise à jour de θ est analogue à une **montée de gradient** sur la log-vraisemblance

$$\log L(\theta) = \theta^\top g(y_{\text{obs}}) - \log Z(\theta),$$

car $g(y_{\text{obs}}) - \mathbb{E}_\theta[g(Y)]$ joue le rôle d'un **gradient stochastique**. Le pas α influence vitesse et stabilité de convergence. La fonction de vraisemblance peut être **non convexe** dans l'espace des graphes : la convergence dépend du point de départ et de la structure du réseau.

A.2 Log-vraisemblance, déviance, AIC et BIC

Constante de normalisation $Z(\theta)$.

Dans un ERGM, l'écriture exacte de la probabilité de tirage est :

$$\Pr_\theta(Y = y) = \frac{\exp\{\theta^\top g(y)\}}{Z(\theta)}, \quad Z(\theta) = \sum_{y' \in \mathcal{Y}} \exp\{\theta^\top g(y')\},$$

où \mathcal{Y} est l'ensemble de *tous* les graphes possibles sur le même jeu de noeuds. $Z(\theta)$ assure la **normalisation** (les probabilités somment à 1). Comme \mathcal{Y} est gigantesque, $Z(\theta)$ n'est pas calculable exactement ; on l'approxime par MCMC.

Log-vraisemblance.

Idée générale.

La **vraisemblance** $L(\theta)$ mesure, pour un θ donné, à quel point le modèle juge plausible le graphe observé y_{obs} :

$$L(\theta) = \Pr_\theta(Y = y_{\text{obs}}) = \frac{\exp\{\theta^\top g(y_{\text{obs}})\}}{Z(\theta)},$$

Maximiser $L(\theta)$ (ou son logarithme) revient à choisir les paramètres qui rendent y_{obs} le plus probable dans la famille des modèles.

Pourquoi travailler au log.

On utilise la **log-vraisemblance**

$$\ell(\theta) = \log L(\theta) = \theta^\top g(y_{\text{obs}}) - \log Z(\theta),$$

car (i) les produits de probabilités deviennent des *sommes* (plus stables numériquement), et (ii) le calcul des dérivées/gradiants est direct.

Déviance et critères d'information.

La **déviance** d'un modèle est définie par

$$D = -2\ell(\hat{\theta}),$$

où $\hat{\theta}$ est l'estimateur (par exemple le maximum de vraisemblance). Sous conditions régulières, les **rapports de vraisemblance** ont une loi asymptotique de type χ^2 , d'où le facteur -2 qui permet d'interpréter les *différences* de déviance dans des tests de comparaison de modèles.

Déviance et AIC/BIC (version unifiée).

Dans un ERGM, $D = -2\ell(\hat{\theta})$ sert d'indicateur global d'ajustement : plus $\ell(\hat{\theta})$ est grand, plus D est petit. La déviance mesure toutefois l'ajustement *pur* : un modèle plus complexe a presque toujours une déviance plus faible. Pour **équilibrer ajustement et complexité**, on utilise les critères d'information AIC/BIC :

$$\text{AIC} = 2k - 2\ell(\hat{\theta}), \quad \text{BIC} = k \log n - 2\ell(\hat{\theta}),$$

où k est le nombre de paramètres et n une taille effective (en réseau, souvent proche du nombre de dyades potentielles). On compare des modèles ajustés sur les mêmes données et on retient en pratique celui dont l'AIC/BIC est **minimal** (meilleur compromis « fidélité/parcimonie »). AIC/BIC permettent en outre de comparer des modèles *non nécessairement emboîtés*, ce que ne permet pas un test basé uniquement sur la différence de déviance.

A.3 Sorties usuelles de `summary(ergm)`

- **Call** : l'appel exact (traçabilité, reproductibilité).
- **Coefficients** : estimés $\hat{\theta}_i$ (un par statistique), **erreur-type** (SE), **statistique** z ($z = \hat{\theta}_i/\text{SE}(\hat{\theta}_i)$) et **p-valeur bilatérale** pour $H_0 : \theta_i = 0$. Un grand $|z|$ suggère un effet éloigné de 0 au regard de l'incertitude.
- **MCMC %** : fraction (en %) de l'erreur-type due au *bruit Monte-Carlo*. *Interprétation et actions* : un MCMC% élevé indique que l'échantillon simulé est peu informatif (forte autocorrélation). **Burn-in** = nombre d'itérations initiales jetées avant d'enregistrer des échantillons (laisser la chaîne se stabiliser); **Thinning** = ne garder qu'une itération sur m pour **décorrérer les échantillons** entre eux (réduire l'autocorrélation serielle); **Taille d'échantillon** = nombre total d'états conservés pour estimer les moyennes. Si la chaîne *mélange mal*, il peut être nécessaire d'ajuster les contraintes ou le mécanisme de proposition.
- **Log-vraisemblance, déviance D, AIC, BIC** : indicateurs globaux (définitions ci-dessus). AIC/BIC plus petits \Rightarrow meilleur compromis.

A.4 Contraintes et mécanisme de proposition (MH) en biparti

Contraintes (restriction de l'espace d'état).

Les **contraintes** (par exemple `b1part`) restreignent l'espace des graphes explorés par MH aux configurations *valides* : en biparti, interdire les arêtes intra-mode, imposer certaines bornes de degrés, etc. Elles ne constituent pas un « assouplissement » : elles **excluent** simplement les états invalides et garantissent que chaque graphe visité respecte la structure voulue.

Mécanisme de proposition.

Le **mécanisme de proposition** (souvent appelé « proposal ») génère un candidat y' à partir de y (par exemple tirer une dyade admissible et toggler l'arête), en *respectant* les contraintes. Il définit la distribution $q(y \rightarrow y')$ qui intervient dans

le **ratio de Hastings**. Si q est asymétrique (par exemple plus de façons de détruire que de créer une structure), le terme $\frac{q(y' \rightarrow y)}{q(y \rightarrow y')}$ corrige cette asymétrie dans la probabilité d'acceptation.

Pourquoi ces corrections sont indispensables.

Le couple « Δ du modèle » + « ratio de Hastings » assure la **réversibilité** (détail de balance) de la chaîne et garantit que la **loi stationnaire** visée est bien Pr_θ . Sans cette correction, la chaîne convergerait vers une distribution biaisée (dépendante du mécanisme de proposition) au lieu de la loi ERGM recherchée.

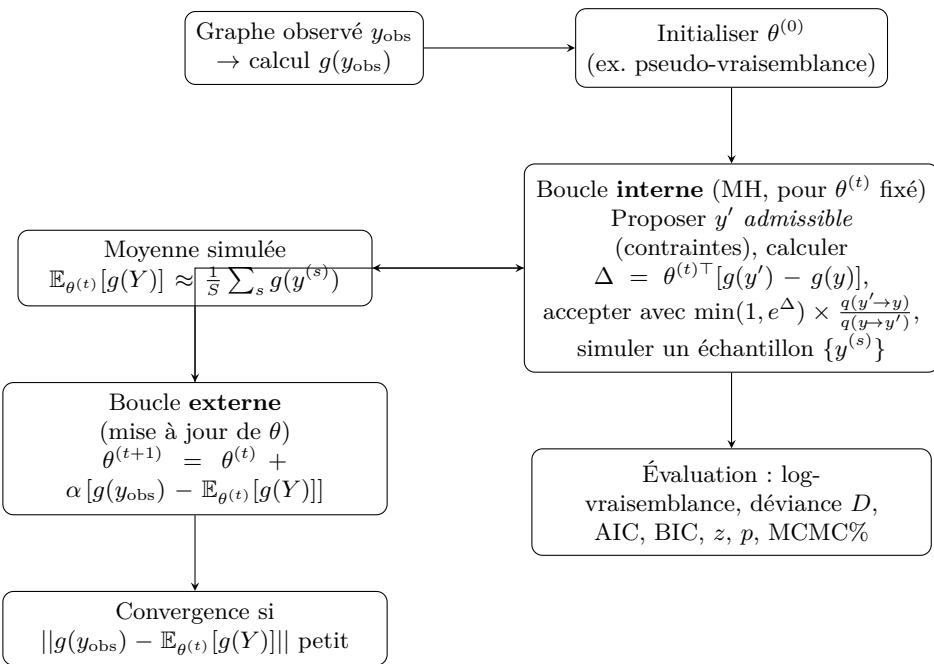
A.5 Exemple minimal (R)

```

1 library(ergm)
2 # Réseau biparti jouet : 4 acteurs -> 2 groupes
3 m <- matrix(c(1,0,
4                 1,0,
5                 0,1,
6                 0,1), nrow=4, byrow=TRUE)
7 nw <- network::network(m, bipartite=2, directed=FALSE)
8
9 # Modèle 1 : densité (edges)
10 fit1 <- ergm(nw ~ edges)
11
12 # Modèle 2 : densité + distribution de degrés côté groupes (
13 #               mode 2)
14 fit2 <- ergm(nw ~ edges + b2degrange(from=2, to=3))
15
16 summary(fit1)
17 summary(fit2) # comparer log-vraisemblance, AIC, BIC, z, p-
values, MCMC%

```

A.6 Schéma du processus d'estimation



A.7 Points d'attention

Choix de $g(y)$.

Les statistiques doivent refléter des mécanismes plausibles (densité, degrés, triangles, homophilie, effets d'attributs, etc.) sans surcharger le modèle : empiler trop de termes peut entraîner instabilités, colinéarités ou non-identifiabilité. Une approche progressive (noyau simple, diagnostics, puis complexification) est souvent utile.

Diagnostics MCMC.

Surveiller l'autocorrélation et la stabilité des moyennes simulées. Un **MCMC%** élevé signale une forte dépendance entre échantillons. Augmenter le **burn-in** (laisser la chaîne se stabiliser avant de collecter), appliquer du **thinning** (garder une itération sur m pour **décorreler les échantillons entre eux**), et/ou accroître la **taille d'échantillon**. Si le mélange reste faible, ajuster contraintes et mécanisme de proposition peut aider.

Comparaison de modèles.

AIC/BIC comparent des modèles ajustés sur les mêmes données en pénalisant la complexité. L'information est dans les **écart**s d'AIC/BIC (pas dans leur niveau absolu). Les conclusions gagnent à être croisées avec une **évaluation de type goodness-of-fit** : comparer, entre réseaux simulés et observé, des distributions de degrés, distances géodésiques, motifs triadiques, etc.