Inférence de réseaux génétiques à partir de données hétérogènes : Données d'expression de gènes et matrices d'interaction issues d'autres types de données

Audrey Noeser

Inférence de réseaux génétiques à partir de données hétérogènes :

Données d'expression de gènes et matrices d'interaction issues d'autres types de données

Responsable du diplôme :

Mme Armelle GUILLOU

Maître de stage :

Mme Sophie LEBRE

Audrey NOESER

Master 2 de Mathématiques

Mention Statistique et Applications

Université de Strasbourg

Remerciements

Je tiens à remercier dans un premier temps Michel de Mathelin, Directeur du laboratoire ICube, laboratoire des sciences de l'Ingénieur, de l'Informatique et de l'Imagerie, pour m'avoir permis d'effectuer mon stage dans son laboratoire.

Un grand merci à Sophie LEBRE, pour m'avoir accompagnée, guidée et conseillée tout au long de mon stage. Je lui suis reconnaissante de son soutien, de sa grande patience et de la confiance qu'elle m'a accordée.

Je remercie également tout le service «Bioinformatique théorique, Fouille de données et Optimisation stochastique» de m'avoir accueillie au sein de leur équipe.

Merci aussi à Nicolas Poulin, ingénieur de recherche à l'IRMA, pour ses conseils concernant la rédaction du présent rapport.

Pour finir, merci à tous les enseignants qui m'ont permis d'acquérir un grand nombre de connaissances durant toutes ces années.

Notations

$\propto$: symbole de proportionnalité

$\approx$: signifie « est approximativement égal à »

|. | : déterminant d'une matrice

^{t} : transposée d'une matrice

$\delta(x, y)$: symbole de Kronecker

Table des matières

1 Introduction

1.1 Présentation du laboratoire

1.1.1 ICube

ICube est une unité mixte de recherche (UMR7357) sous la cotutelle de l'Université de Strasbourg, du CNRS, de l'ENGEES et de l'INSA de Strasbourg. Les domaines d'application privilégiés du laboratoire ICube sont la santé et l'environnement.

Le laboratoire est composé de quatre départements :

- Informatique Recherche
- Imagerie, Robotique, Télédétection & Santé
- Électronique du Solide, Systèmes & Photonique
- Mécanique

rendant possibles des projets ambitieux à l'interface de plusieurs disciplines : informatique, robotique, biophysique, microélectronique, photonique, mécanique, traitement d'images.

1.1.2 Equipe "Bioinformatique théorique, Fouille de données et Optimisation stochastique" (BFO)

Au sein du Département Informatique Recherche, l'équipe BFO couvre un large spectre de recherches en informatique, allant de la bioinformatique à l'intelligence artificielle. Ses thèmes de recherche sont :

- la bioinformatique théorique
- la fouille de données
- l'ingénierie des connaissances
- l'optimisation stochastique

1.2 Présentation du stage

Une question essentielle en génomique est celle de la compréhension du rôle de chaque gène. L'ensemble des gènes d'un grand nombre d'organismes ont aujourd'hui été identifié mais il s'agit maintenant de comprendre le rôle de chacun d'eux. La plupart du temps, un gène ne fonctionne pas seul mais en étroite collaboration avec d'autres gènes. En effet, un gène peut activer un autre gène ou au contraire le bloquer. Ces interactions, qui peuvent se produire entre des milliers de gènes (28 000 et 34 000 gènes chez l'homme [1]), sont représentées par des réseaux de régulation comme celui représenté en Figure 1 où chaque nœud représente un gène et chaque arête correspond à l'activation du gène fils par le gène parent. Ainsi, sur la figure 1, le gène G^1 active le gène G^2.

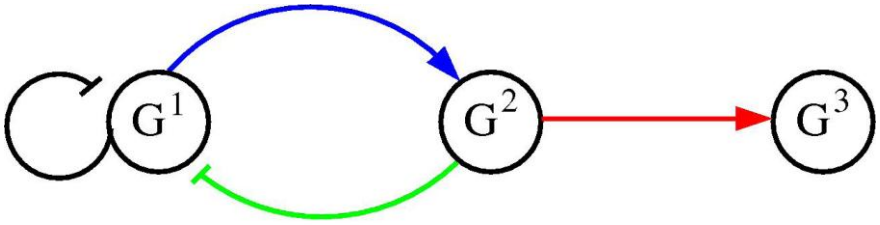


Figure 1 : Réseau génétique

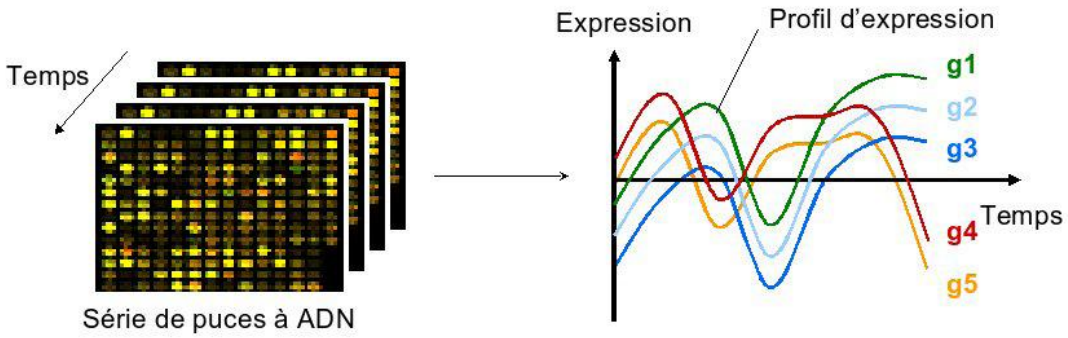


Figure 2 : Gauche : Biopuces ou Puces à ADN. Droite : Courbes d'activités ou d'expression obtenues grâce aux biopuces pour chaque gène.

Afin d'une part, de mieux comprendre le vivant et d'autre part, de développer des traitements contre certaines maladies, un des défis essentiels de la génétique consiste à mettre en évidence ces relations de régulation entre les gènes, afin de reconstruire le réseau de régulation. Grâce au développement technologique, nous disposons de données qui permettent d'observer le comportement des gènes :

- Les biopuces ou puces à ADN permettent aujourd'hui d'observer simultanément l'ensemble des gènes au sein d'un organisme. Chaque point lumineux de la biopuce représentée Figure 2 correspond à une mesure de fluorescence traduisant l'activité d'un gène. En reproduisant successivement la mesure par biopuces, on peut suivre un processus biologique (réponse au stress du à un produit chimique, développement d'un embryon...) et représenter une courbe d'activité des gènes au cours du temps. On obtient alors des données sous formes de séries temporelles (Figure 2).
- Les données ChIP-chip (« ChIP » pour Chromatin Immuno Precipitation et « chip » pour puce en anglais), permettent d'observer la capacité des protéines à se fixer sur une région particulière de l'ADN. Ces données se présentent sous la forme d'une matrice dont chaque élément nous apporte une information sur la capacité d'interaction de 2 gènes, soit sur la présence d'arête entre ces 2 gènes sur le réseau.
- Les données d'interactions de protéines : expérience qui permet d'avoir une indication sur la capacité de deux protéines de se lier en un complexe protéique. L'expression de certains gènes est activée non pas par une seule protéine mais par un complexe protéique.

Il s'agit maintenant d'extraire et de regrouper les informations issues d'un grand nombre de données.

L'équipe BFO a développé plusieurs approches de reconstruction de réseaux génétiques. En particulier, une méthode de reconstruction de réseaux génétiques appelée Auto Regressive TIme VArying network model (ARTIVA) permet d'estimer un réseau génétique dont la structure varie au cours du temps à partir de l'observation des gènes au sein d'une ou plusieurs cellules [2]. Cet algorithme repose sur une méthode itérative MCMC (Markov Chain Monte Carlo) qui permet d'explorer l'ensemble des modèles possibles et d'estimer la probabilité a posteriori de chaque configuration de réseau d'après les données biologiques. Cet algorithme se montre efficace sur des données synthétiques ainsi que sur des données réelles (réponse au stress chez la levure, développement de la drosophile, ... [2]).

L'objectif de ce stage consiste à étendre cette méthode d'inférence de réseau Bayésien pour permettre l'intégration de différents types de données (données d'interactions de protéines, données de sites promoteurs...), en proposant un modèle permettant de prendre en compte différentes sources de données a priori.

2 Modélisation et méthodes statistiques utilisées

2.1 Réseaux Bayésiens

Un réseau Bayésien est un graphe dans lequel les nœuds représentent des variables aléatoires, et les arcs représentent des relations de cause à effet entre variables. Un réseau Bayésien est un graphe orienté acyclique $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ avec V l'ensemble des nœuds du réseaux et E l'ensemble des arcs. À chaque nœud x appartenant à V du graphe est associé la distribution de probabilité conditionnelle suivante :

$$p\big(x|pa(x)\big)$$

où $pa(x)$ représente les parents immédiats de x dans V.

L'ensemble V est donc un ensemble de variables aléatoires discrètes. Chaque nœud de V est conditionnellement indépendant de ses non-descendants, étant donné ses parents immédiats. Il est ainsi possible de factoriser les distributions de probabilité conditionnelles sur l'ensemble des variables en faisant le produit :

$$p(\mathrm{V}) = \prod_{x \in \mathrm{V}} p\big(x|pa(x)\big)$$

C'est la modélisation que nous utilisons pour représenter la dépendance entre un facteur de transcription et un gène cible.

2.2 Le principe de l'inférence bayésienne

Tout d'abord, on exprime un a priori sur la valeur d'une grandeur à estimer. Puis, le principe de l'analyse bayésienne est d'actualiser cet a priori à partir de l'information provenant des données observées (issues d'enquêtes ou d'expérimentations).

Supposons que l'on cherche à estimer le vecteur de paramètre θ. L'analyse bayésienne considère le paramètre θ comme un vecteur aléatoire auquel on associe des lois de probabilités a priori $\pi(\theta)$.

L'idée est ensuite d'estimer la distribution a posteriori de θ conditionnellement au vecteur des observations $\underline{x}$. Cette distribution conditionnelle résume l'information disponible sur θ une fois $\underline{x}$ observé. L'estimation de la distribution a posteriori est réalisée grâce à la règle de Bayes, que l'on présente sous sa forme proportionnelle dans le cas continu :

$$\pi(\theta|\underline{x}) \propto f(\underline{x}|\theta).\pi(\theta)$$

avec :

$\pi(\theta|\underline{x})$: la loi a posteriori

$f(\underline{x}|\theta)$: la vraisemblance du modèle

$\pi(\theta)$: la loi a priori sur le vecteur de paramètres θ

Il s'agit donc d'estimer la loi a posteriori, puis d'en prendre l'espérance $\mathbb{E}(\theta|\underline{x})$, qui constitue l'estimateur bayésien du paramètre θ.

Cependant, il est rare de pouvoir obtenir une expression explicite de la loi a posteriori, notamment lorsque la dimension de θ est élevée.

Ce problème peut être évité grâce aux méthodes de Monte Carlo par chaîne de Markov (méthodes MCMC).

2.3 Méthode MCMC

Les méthodes MCMC ont donc été développées pour obtenir une estimation numérique de la loi a posteriori, et donc de son espérance. Le principe de base des méthodes MCMC, dans le contexte de l'inférence bayésienne, est de générer une chaîne de Markov $(\theta^{\mathrm{t}})_{\mathrm{t}=1,\ldots,\mathrm{T}}$ de loi stationnaire la distribution a posteriori $\pi(\theta|\underline{x})$, puis d'approcher $\mathbb{E}(\theta|\underline{x})$ par la loi forte des grands nombres, de la manière suivante :

$$\mathbb{E}(\theta|\underline{x}) \approx \frac{1}{\mathrm{T}}\sum_{\mathrm{t}=1}^{\mathrm{T}} \theta^{\mathrm{t}}$$

Il existe plusieurs algorithmes permettant de générer une telle chaîne; les deux plus connus sont l'algorithme de Gibbs et l'algorithme de Metropolis-Hastings. Nous verrons ici uniquement l'algorithme de Metropolis-Hastings :

Supposons que l'on veuille générer une chaîne de Markov $\{\theta_t\}_{t\geq 0}$ homogène, de loi stationnaire $\pi(\theta|\underline{x})$, lorsque $\pi(\theta|\underline{x})$ est connue à une constante de normalisation près $\left(\pi(\theta|\underline{x}) \propto f(\underline{x}|\theta)\pi(\theta)\right)$.

Supposons de plus que l'on dispose d'une loi instrumentale $q(.,\theta)$ sur Θ *(loi conditionnelle à θ)*. On simule la chaîne $\{\theta^{\mathrm{t}}\}_{t\geq 0}$ comme suit :

Itération 0 : Initialiser avec une valeur arbitraire θ^0

Itération t : Mettre à jour θ^{t} par $\theta^{\mathrm{t+1}}$ de la façon suivante :

Générer $\theta^* \sim q(. \mid \theta^{\mathrm{t}})$

Poser $\rho(\theta, \theta^*) = min\left(1, \frac{\pi(\theta^*|\underline{x})q(\theta^t|\theta^*)}{\pi(\theta^t|\underline{x})q(\theta^*|\theta^t)}\right)$

Prendre $\theta^{\mathrm{t+1}} = \begin{cases} \theta^* \; avec \; probabilité \; \rho(\theta, \theta^*) \\ \theta^{\mathrm{t}} \text{ avec probabilité } 1 - \rho(\theta, \theta^*) \end{cases}$

2.4 Méthode MCMC à sauts réversibles (RJ- MCMC)

La situation est plus compliquée quand la dimension du vecteur des paramètres inconnus θ lui-même est inconnue. En bayésien, on peut utiliser un algorithme MCMC à sauts réversibles, introduit par Green (1995) [5], en utilisant une loi de proposition qui permet de se déplacer dans des espaces de différentes dimensions.

On considère un vecteur $\theta^{(\mathrm{k})} \in \mathbb{R}^{\mathrm{n_k}}, \mathrm{n_k} \in \mathbb{N}$. On définit la loi jointe :

$$p\left(k, \theta^{(\mathrm{k})}, x\right) = p\left(x \middle| k, \theta^{(\mathrm{k})}\right) p\left(\theta^{(\mathrm{k})} \middle| k\right) p(k)$$

où

$p(k)$ est la loi a priori sur le dimension du vecteur des paramètres $\theta^{(\mathrm{k})}$

$p\left(\theta^{(\mathrm{k})} \middle| k\right)$ est la loi a priori sur les paramètres sachant k

$p\left(x \middle| k, \theta^{(\mathrm{k})}\right)$ est la densité des observations x (vraisemblance).

On s'intéresse à la loi a postériori de $\left(k, \theta^{(\mathrm{k})}\right) \in \bigcup_{k \in \mathcal{K}} \mathcal{C}_k, \mathcal{C}_k = \{k\} \times \mathbb{R}^{\mathrm{n}_k}$:

$$p\left(k, \theta^{(\mathrm{k})} \middle| x\right) = \frac{p\left(k, \theta^{(\mathrm{k})}, x\right)}{\iint p(k, \theta^{(\mathrm{k})}, x) dk d\theta^{(k)}} \propto p\left(k, \theta^{(\mathrm{k})}, x\right)$$

Le but de la méthode est donc de simuler la loi a postériori en utilisant une chaîne de Markov sur $\mathcal{C}_k$ se déplaçant de $\left(k, \theta^{(k)}\right)$ vers $\left(k', \theta^{(k')}\right)$. Pour se déplacer de $\mathbb{R}^{\mathrm{n}_k}$ vers $\mathbb{R}^{\mathrm{n}_{k'}}$, avec $k \neq k'$, on doit compléter ces espaces pour se placer dans un espace de même dimension. Il existe alors deux entiers positifs tels que :

$$\mathrm{n}_k + \mathrm{n}_{kk'} = \mathrm{n}_{k'} + \mathrm{n}_{k'k}$$

On définit alors deux applications inverses l'une de l'autre correspondant aux sauts de la chaîne de Markov permettant de sauter d'un espace à un autre.

$$g_{kk\prime} = \begin{cases} g_{1kk\prime} \begin{cases} \mathbb{R}^{n_k + n_{kk\prime}} \longrightarrow \mathbb{R}^{n_{k\prime}} \\ \left(\theta^{(k)}, u\right) \longmapsto \theta^{(k\prime)} \end{cases} \\ g_{2kk\prime} \begin{cases} \mathbb{R}^{n_k + n_{kk\prime}} \longrightarrow \mathbb{R}^{n_{k\prime k}} \\ \left(\theta^{(k)}, u\right) \longmapsto u \end{cases} \end{cases}$$

et

$$g_{k\prime k} = \begin{cases} g_{1k\prime k} \begin{cases} \mathbb{R}^{n_{k\prime} + n_{k\prime k}} \longrightarrow \mathbb{R}^{n_k} \\ \left(\theta^{(k\prime)}, u'\right) \longmapsto \theta^{(k)} \end{cases} \\ g_{2k\prime k} \begin{cases} \mathbb{R}^{n_{k\prime} + n_{k\prime k}} \longrightarrow \mathbb{R}^{n_{kk\prime}} \\ \left(\theta^{(k\prime)}, u'\right) \longmapsto u \end{cases} \end{cases}$$

Le nouvel état $\theta^{(k\prime)} = g_{1kk\prime}\left(\theta^{(k)}, u\right)$ est accepté avec la probabilité :

$$\rho_{kk\prime} = min\left\{\frac{p_{k\prime} f_{k\prime}\left(\theta^{(k\prime)}\right)}{p_k f_k(\theta^{(k)})} \frac{p_{k\prime k} q_{k\prime k} f_{k\prime}\left(\theta^{(k\prime)}, u'\right)}{p_{kk\prime} q_{kk\prime}(\theta^{(k)}, u)} \left|\frac{\partial g_{kk\prime}}{\partial \theta^{(k)} \partial u}\right|, 1\right\}$$

où

$p_{k\prime k}$ est la probabilité de tenter un déplacement de $\mathbb{R}^{n_{k\prime}}$ vers $\mathbb{R}^{n_k}$

$p_{kk\prime}$ est la probabilité de tenter un déplacement de $\mathbb{R}^{n_k}$ vers $\mathbb{R}^{n_{k\prime}}$

$\left|\frac{\partial g_{kk\prime}}{\partial \theta^{(k)} \partial u}\right|$ est le jacobien de la transformation

$\frac{p_{k\prime} f_{k\prime}\left(\theta^{(k\prime)}\right)}{p_k f_k\left(\theta^{(k)}\right)} = \frac{p_{k\prime}}{p_k} \frac{p\left(\theta^{(k\prime)} | k\prime\right)}{p\left(\theta^{(k)} | k\right)} \frac{p\left(x | k\prime, \theta^{(k\prime)}\right)}{p\left(x | k, \theta^{(k)}\right)}$

2.5 Burn-in ou période de chauffe

A chaque itération de l'algorithme MCMC, une nouvelle configuration du réseau de régulation est proposée, de façon aléatoire, à partir de distributions de probabilité basée sur la construction d'une chaîne de Markov qui a comme distribution d'équilibre la distribution du modèle du réseau. La distribution d'équilibre est obtenue lorsque les chaînes de Markov convergent. Il n'est pas possible de prouver la convergence mais on peut observer une stabilisation de la chaine après un grand nombre d'itérations. En règle générale, les itérations initiales sont très instables. Le burn-in ou période de chauffe désigne ces itérations initiales qui ne seront pas utilisées pour l'estimation du réseau de régulation.

3 Objectif de ce stage : Ajouter le couplage de deux a priori exponentiels

On introduira ici la modélisation utilisée et les extensions pour prendre en compte deux sources de données a priori.

Soient p le nombre de gènes observés et $x = (x_i)_{1 \leq i \leq p}$ les valeurs d'expression des gènes mesurées. On note G le réseau défini par un ensemble d'arêtes dirigées entre les p gènes, un graphe orienté, représenté par une matrice dont le coefficient G_{ij} est égal à zéro s'il y a absence d'arête orientée du gène i vers le gène j et égal à 1 s'il existe une arête orientée du gène i vers le gène j.

G_i est le sous-réseau associé au gène cible i, déterminé par l'ensemble de ses parents, à savoir les nœuds avec une arête orientée vers le gène i : ce sont les régulateurs potentiels du gène cible.

3.1 Modèle de régression

Pour chaque gène i, la variable x_i représente l'expression du gène i. L'expression du gène i dépend d'un ensemble de s_i gènes parents, noté G_i, indicés par $S_i = \{j_1, \dots, j_{s_i}\} \subset \{1, \dots, p\}$, pour que les parents du gène cible soient différents du gène cible et ainsi rendre le réseau acyclique. On définit alors le modèle de régression suivant :

$$x_i = a_{i0} + \sum_{j \in S_i} a_{ij} x_j + \varepsilon_i$$

avec $1 \leq i \leq p$ et où $a_{ij} \in \mathbb{R}$, $\varepsilon_i \sim \mathcal{N}(0, (\sigma_i)^2)$ avec $\sigma_i > 0$.

3.2 Lois a priori

Nous présentons ici les lois a priori utilisées dans l'article Lèbre et al. (2010) [2] pour l'inférence de réseaux bayésiens sans information a priori (uniquement des données d'expression).

Pour chaque gène i, le nombre s_i de parents suit une loi de Poisson tronquée de moyenne Λ et de maximum $\bar{s} = 5$:

$$P(s_i|\Lambda) \propto \frac{\Lambda^{s_i}}{s_i!} \mathbb{I}_{\{s_i \leq \bar{s}\}}$$

Conditionnellement à s_i, la loi a priori pour l'ensemble de parents G_i est une loi uniforme sur tout l'ensemble des parents de cardinal s_i :

$$P(G_i|s_i) = 1 \Big/ \binom{p}{s_i} = \frac{s_i!\,(p-s_i)!}{p!}$$

La loi a priori globale de la structure de réseau est donnée par la marginalisation :

$$P(G_i|\Lambda) = \sum_{s_i=0}^{\bar{s}} P(G_i|s_i)P(s_i|\Lambda)$$

Le terme Λ peut être interprété comme le nombre de parents. Cet hyper-paramètre suit une loi gamma :

$$P(\Lambda) = \mathcal{G}a(\alpha, \beta)$$

où le paramètre d'échelle α et le paramètre d'intensité β sont choisis tels que la distribution a priori diminue lorsque le nombre de parents augmente. D'après l'article [2], on fixe $\alpha = 1$ et $\beta = 0.5$.

Conditionnellement à l'ensemble des parents G_i de taille s_i, les $s_i + 1$ coefficients du modèle de régression forme un sous-ensemble de a_i noté $a_{G_i} = \left(a_{i0}, a_{ij}\right)_{j \in S_i}$, qui suivent une loi Gaussienne multivariée de moyenne 0 et de matrice de covariance ${\sigma_i}^2 \Sigma_{G_i}$:

$$P(a_i|G_i, \sigma_i) = \left|2\pi({\sigma_i}^2)\Sigma_{S_i}\right|^{-\frac{1}{2}} exp\left(-\frac{{}^t a_{G_i} {\Sigma_{G_i}}^{-1} a_{G_i}}{2{\sigma_i}^2}\right)$$

où $\Sigma_{G_i} = \delta^{-2}\ {}^t D_{G_i} D_{G_i}$, avec D_{G_i} un vecteur de dimension $s_i + 1$ contenant la constante du modèle de régression a_{i0} et les observations des j gènes de G_i, $\left(x_j\right)$.

La loi a priori de la variance ${\sigma_i}^2$ est une loi Inverse Gamma :

$$P({\sigma_i}^2) = \mathcal{IG}(\upsilon_0, \gamma_0)$$

Comme dans l'article [2], on fixe les hyper-paramètres $\upsilon_0 = 1$ et $\gamma_0 = 0.1$.

Le rapport signal-sur-bruit attendu δ^2 suit une loi Inverse Gamma :

$$P(\delta^2) = \mathcal{IG}(\alpha_{\delta^2}, \beta_{\delta^2})$$

D'après l'article [2], on fixe $\alpha_{\delta^2} = 2$ et $\beta_{\delta^2} = 0.2$.

3.3 Deux sources d'informations a priori

Afin d'estimer le réseau génétique, on souhaite intégrer deux sources d'informations a priori, notées B^1 et B^2, en utilisant l'approche proposée par Werhli et Husmeier (2007) [3] qui se sert d'une loi exponentielle.

B^1 et B^2, les matrices des connaissances a priori contiennent les coefficients $\left(B_{ij}^1\right)_{1 \le i \le p, 1 \le j \le s_i}$ et $\left(B_{ij}^2\right)_{1 \le i \le p, 1 \le j \le s_i}$ respectivement, tels que $B_{ij} \in [0,1]$:

- $0 \le B_{ij} < 0.5$ si on a la preuve a priori qu'il n'y ait pas d'arête dirigée entre le gène i et le gène j
- $B_{ij} = 0.5$ si on n'a pas de connaissance a priori sur la présence ou l'absence d'arête dirigée entre le gène i et le gène j
- $0.5 < B_{ij} \le 1$ si on a la preuve a priori qu'il y ait une arête dirigée entre le gène i et le gène j

La probabilité a priori d'un réseau G_i sachant les hyper-paramètres β_1 et β_2 des connaissances a priori B^1 et B^2 est donnée par :

$$P(G_i | \beta_1, \beta_2) = \frac{e^{-\{\beta_1 E_1(G_i) + \beta_2 E_2(G_i)\}}}{Z(\beta_1, \beta_2)}$$

où $E_1(G_i)$ et $E_2(G_i)$, mesurant la concordance entre le réseau G_i et les connaissances a priori B^1 et B^2, sont définis ainsi :

$$E_1(G_i) = \sum_{j=1}^{N} \left| B_{ij}^1 - G_{ij} \right|$$

$$E_2(G_i) = \sum_{j=1}^{N} \left| B_{ij}^2 - G_{ij} \right|$$

avec N le nombre de gènes dans le réseau G_i, et où :

$$Z(\beta_1, \beta_2) = \sum_{G_i \in \mathcal{G}} e^{-\{\beta_1 E_1(G_i) + \beta_2 E_2(G_i)\}}$$

3.4 Loi a posteriori

D'après l'article [4], la loi a postériori est donnée par :

$$P(x_i|G_i, \delta^2) = (\delta^2 + 1)^{-\frac{s_{i+1}}{2}} \frac{\left(\frac{\gamma_0}{2}\right)^{\upsilon_0/2}}{\Gamma\left(\frac{\upsilon_0}{2}\right)} \Gamma\left(\frac{\upsilon_0 + length(x_i)}{2}\right) \left(\frac{\gamma_0 + {}^t x_i P_i x_i}{2}\right)^{-\left(\frac{\upsilon_0 + length(x_i)}{2}\right)}$$

où

$P_i = I - D_{G_i} M_i \, {}^t D_{G_i}$

$M_i = \frac{\delta^2}{\delta^2 + 1} \left({}^t D_{G_i} D_{G_i} \right)^{-1}$

avec I la matrice identité de taille $length(x_i)$.

3.5 Procédure MCMC à sauts réversibles (RJ-MCMC)

Une fois les lois a priori définies, l'objectif est d'échantillonner la structure du réseau ainsi que les hyper-paramètres. Pour cela, on utilise un schéma standard de Métropolis-Hastings. La nouvelle structure de réseau $G_{i,new}$ est proposée à partir de la distribution de proposition $Q\left(G_{i,new}|G_{i,old}\right)$. Le processus est identique pour les hyper-paramètres β_1 et β_2. D'après l'article [2], les probabilités d'acceptation de la nouvelle structure de réseau $G_{i,new}$, et des nouveaux hyper-paramètres $\beta_{1,\text{new}}$ et $\beta_{2,\text{new}}$ sont données par :

$$A\left(G_{i,new}|G_{i,old}\right) = min\left\{1, \frac{P\left(x|G_{i,new}\right)}{P\left(x|G_{i,old}\right)} \frac{P\left(G_{i,new}|\beta_1, \beta_2\right)}{P\left(G_{i,old}|\beta_1, \beta_2\right)} \frac{Q\left(G_{i,old}|G_{i,new}\right)}{Q\left(G_{i,new}|G_{i,old}\right)}\right\}$$

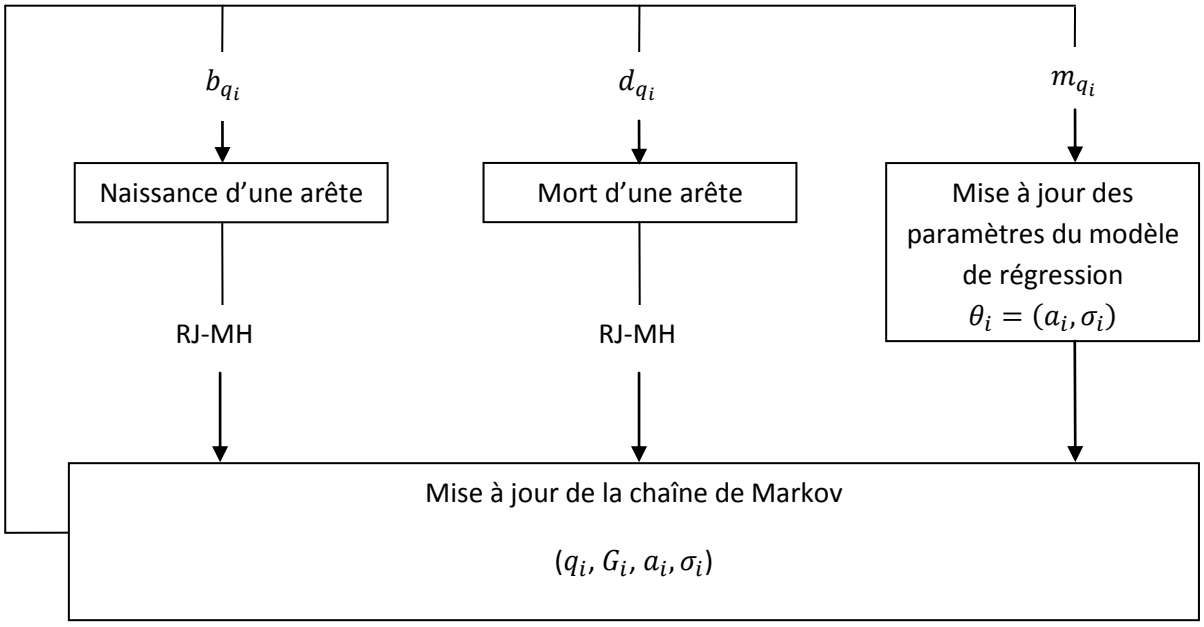
$$A(\beta_{1,\text{new}}|\beta_{1,\text{old}}) = min\left\{1, \frac{P(G|\beta_{1,\text{new}}, \beta_2)}{P(G|\beta_{1,\text{old}}, \beta_2)}\right\}$$

$$A(\beta_{2,\text{new}}|\beta_{2,\text{old}}) = min\left\{1, \frac{P(G|\beta_1, \beta_{2,\text{new}})}{P(G|\beta_1, \beta_{2,\text{old}})}\right\}$$

On choisi comme loi a priori pour les hyper-paramètres β_1 et β_2 une loi uniforme d'intervalle $[0, b]$.

Note : il faut aussi échantillonner les coefficients et la variance du modèle de régression et les hyper paramètres.

Pour proposer la nouvelle structure de réseau, on va considérer la naissance d'une arête, la mort d'une arête ou la mise à jour des coefficients du modèle de régression a_{G_i}. Le nombre de parents pour la structure de réseau actuelle est notée q_i. La nouvelle structure de réseau proposée peut être de dimension variable car le nombre de parents du gène i peut varier de q_i à $q_i + 1$ ou de q_i à $q_i - 1$. Etant donné que la dimension des paramètres est variable, on va utiliser une procédure RJ-MCMC :



Les probabilités de choisir la naissance ou la mort d'une arête, b_{q_i} et d_{q_i} respectivement, sont données par :

$$b_{q_i} = C_{q_i} min\left\{1, \frac{P(q_i + 1)}{P(q_i)}\right\}$$

$$d_{q_i} = C_{q_i} min\left\{1, \frac{P(q_i - 1)}{P(q_i)}\right\}$$

avec C_{q_i} une constante de normalisation dépendant de s_i est défini tel que $b_{q_i} + d_{q_i} = 1$.

L'algorithme pour échantillonner la structure du réseau est le suivant :

Initialisation : $\left({q_i}^0, {G_i}^0, {\theta_i}^0\right)$

Itération t :

Echantillonner $u \sim \mathcal{U}[0,1]$

Si $u < b_{q_i}$ **alors** une **naissance d'une arête** est proposée :

choisir un nouveau parent $j^* \sim \mathcal{U}([1,p] \not\subset S_i)$

$G_{i,new} = G_{i,old} \cup \{j^*\}$

Calculer $A\left(G_{i,new} \middle| G_{i,old}\right)$

Echantillonner $u \sim \mathcal{U}[0,1]$

Si $u \leq A\left(G_{i,new} \middle| G_{i,old}\right)$ **alors** le modèle devient $\left({q_i}^t + 1, {G_{i,new}}^t\right)$: mise à jour des paramètres θ_i

Sinon le modèle reste inchangé $\left({q_i}^t, {G_i}^t\right) = \left({q_i}^{t-1}, {G_i}^{t-1}\right)$: mise à jour des paramètres θ_i

Sinon Si $u < b_{q_i} + d_{q_i}$ **alors** la **mort d'une arête** est proposée :

Choisir un parent existant $j^* \sim \mathcal{U}[S_i]$

$G_{i,new} = G_{i,old} \setminus \{j^*\}$

Calculer $A\left(G_{i,new} \middle| G_{i,old}\right)$

Echantillonner $u \sim \mathcal{U}[0,1]$

Si $u \leq A\left(G_{i,new} \middle| G_{i,old}\right)$ **alors** le modèle devient $\left({q_i}^t - 1, {G_{i,new}}^t\right)$: mise à jour des paramètres θ_i

Sinon le modèle reste inchangé $\left({q_i}^t, {G_i}^t\right) = \left({q_i}^{t-1}, {G_i}^{t-1}\right)$: mise à jour des paramètres θ_i

Sinon on considère seulement une mise à jour des paramètres :

$$(\sigma_i)^2|x_i, G_i \sim \mathcal{IG}\left(\frac{\upsilon_0}{2} + \frac{s_i}{2}, \frac{\gamma_0}{2} + \frac{\gamma_0 + {}^t x_i P_i x_i}{2}\right)$$

$$a_i|x_j, G_i, \sigma_i \sim \mathcal{N}\left(M_i \; {}^t D_{G_i} x_j, M_i(\sigma_i)^2\right)$$

A chaque itération de l'algorithme, les hyper paramètres sont mis à jour de la manière suivante :

$$\Lambda|s_i \sim \mathcal{G}a\left(\frac{1}{2} + s_i, 1\right)$$

$$\delta|s_i, G_i, \theta_i \sim \mathcal{IG}\left(s_i + \alpha_{\delta^2}, \frac{\theta_i \; {}^t D_{G_i} D_{G_i} \theta_i}{2(\sigma_i)^2} + \beta_{\delta^2}\right)$$

La probabilité de proposition $Q\left(G_{i,new}\middle|G_{i,old}\right)$ est donnée par :

$$\begin{aligned} Q\left(G_{i,new}\middle|G_{i,old}\right) & \\ &= b_{q_i}\delta\left(q_{i,new}, q_{i,old} + 1\right)Q^+\left(G_{i,new}\middle|G_{i,old}\right) \\ &+ d_{q_i}\delta\left(q_{i,new}, q_{i,old} - 1\right)Q^-\left(G_{i,new}\middle|G_{i,old}\right) \end{aligned}$$

où $q_{i,old}$ représente le nombre de parents avant le mouvement proposé, $q_{i,new}$ le nombre de parents après le mouvement proposé, $Q^+\left(G_{i,new}\middle|G_{i,old}\right) = {1}\Big/{\left(s_i - q_{i,new}\right)}$ est la probabilité de proposition d'une naissance d'arête et $Q^-\left(G_{i,new}\middle|G_{i,old}\right) = {1}\Big/{q_{i,new}}$ est la probabilité de proposition de la mort d'une arête.

4 Implémentation dans R de l'algorithme pour l'inférence du réseau de régulation

Dans cette partie, nous décrirons la fonction principale, *subnet*, implémentée en R pour estimer le réseau de régulation pour un gène cible. La méthode globale consiste à utiliser cette fonction en parallèle pour chaque gène cible. Nous détaillerons également la fonction *bdu* qui permet de choisir le mouvement (naissance d'une arête, mort d'une arête ou mise à jour des coefficients du modèle de régression) entre chaque itération de la procédure MCMC.

4.1 La fonction *subnet*

4.1.1 Description de la fonction

Cette fonction génère un échantillon RJ-MCMC pour approcher la distribution a posteriori d'un sous-réseau de régulation, dans le cadre du modèle de régression. A partir de mesures d'expression génique d'un gène d'intérêt, appelé « gène cible » et un ensemble de gènes, dénommés « gènes parents », la procédure estime les interactions qui se produisent entre les gènes parents et le gène cible. Pour éviter l'autorégulation, l'ensemble des gènes de parents ne doit pas contenir le gène cible.

4.1.2 Paramètres de la fonction

Afin d'exécuter cette fonction, plusieurs paramètres doivent être renseignés :

- *targetData* Un vecteur contenant les mesures d'expression du gène cible.
- *parentData* Une matrice (ou un vecteur s'il n'y a qu'un seul gène parent) avec les mesures d'expression des gènes parents. Les gènes des parents sont indiqués en ligne et les mesure répétées d'expression des gènes en colonne.
- *targetName* Nom du gène cible (facultatif, par défaut : *targetName* = "Target").
- *parentNames* Un vecteur contenant le(s) nom(s) du (des) gène(s) parent(s) (facultatif, par défaut: *parentNames* = NULL).
- *b1, b2* Vecteurs des connaissances a priori sur les interactions entre le gène cible analysé et les parents potentiels. Les coefficients du vecteur sont compris entre 0 et 1 selon la connaissance a priori sur la présence ou l'absence d'arête entre les gènes.
- *saveEstimations* Booléen, si TRUE toutes les distributions a postériori estimées sont enregistrés en tant que fichiers texte, soit dans un nouveau sous-dossier nommé«Résultats» créé par défaut dans le dossier courant

ou dans un dossier spécifié par l'argument *outputPath* (voir ci-dessous) (facultatif, par défaut : *saveEstimations* = TRUE).

- *saveIterations* Booléen, si TRUE, la configuration pour toutes les itérations est enregistrée sous forme de fichiers texte, soit dans un nouveau sous-dossier nommé« Résultats» créé par défaut dans le dossier courant ou dans un dossier spécifié par l'argument *outputPath* (voir ci-dessous) (facultatif, par défaut : *saveIterations* = FALSE).
- *savePictures* Booléen, si TRUE toutes les distributions a postériori estimées et les réseaux sont tracés dans un fichier pdf, soit dans un nouveau sous-dossier nommé « Résultats» créé par défaut dans le dossier courant ou dans un dossier spécifié par l'argument *outputPath* (voir ci-dessous) (facultatif, par défaut : *savePictures* = TRUE).
- *outputPath* Chemin du fichier dans un dossier dans lequel les résultats de sortie doivent être enregistrés, soit un chemin complet ou le nom d'un dossier à créer dans le répertoire courant (facultatif, par défaut: *outputPath* = NULL).
- *maxPred* Nombre maximal d'arêtes estimées pour le gène cible (facultatif, par défaut : *maxPred* = NULL, si *maxPred* = NULL alors le nombre maximal d'arêtes est maxPred = min (dim (*parentData*) [1], 15)).
- *niter* Nombre d'itérations à effectuer de l'algorithme RJ-MCMC (facultatif, par défaut: *niter* = 50000).
- *burn_in* Nombre d'itérations initiales éliminées pour l'estimation de la distribution du modèle (distribution a posteriori) (facultatif, par défaut : *burn_in* = NULL, si *burn_in* = NULL alors la première tranche de 25% des itérations est laissé pour burn_in).
- *edgesThreshold* Seuil de probabilité pour la sélection des arêtes du réseau de régulation (facultatif, par défaut : edgesThreshold = 0,5).
- *layout* Nom de la fonction de détermination de la position des arêtes pour dessiner un graphe. Les valeurs possibles parmi d'autres sont : random, cercle, domaine, fruchterman.reingold, reingold.tilford, fruchterman.reingold.grid, voir le package igraph0 pour plus de détails (facultatif, par défaut : *layout* =" fruchterman.reingold ").
- *c* Constante utiliser pour le calcul de la probabilité de proposition d'un mouvement d'une arête (la naissance ou la mort d'une arête) (facultatif, par défaut: *c* = 0,4).
- *alphaEdges* Hyper paramètre pour échantillonner le nombre q de parents pour le gène cible analysé. q suit une distribution gamma : rgamma (1, shape= *alphaEdges*, rate = (voir ci-dessous *betaEdges*), (par défaut : *alphaEdges* = 1).
- *betaEdges* Hyper paramètre pour échantillonner le nombre q de parents pour le gène cible analysé. q suit une distribution gamma : rgamma (1, shape = (voir *alphaEdges* ci-dessus), rate = betaEdges (facultatif, par défaut : *betaEdges* = 0,5).
- *v0* Hyper paramètre pour échantillonner la variance du bruit (notée σ^2) dans le modèle de régression définissant le réseau de régulation.

La distribution a priori de la variance du bruit est une distribution inverse gamma avec le paramètre de forme v0/2 et le paramètre d'intensité gamma0/2 : rinvgamma (1, shape = v0 / 2, scale = gamma0/2) (facultatif, par défaut : *v0* = 1).

- *gamma0* — Hyper paramètre pour échantillonner la variance du bruit (notée σ^2) dans le modèle de régression définissant le réseau de régulation. La distribution a priori de la variance du bruit est une distribution inverse gamma avec le paramètre de forme v0/2 et le paramètre d'intensité gamma0/2 : rinvgamma (1, shape = v0 / 2, scale = gamma0) (facultatif, par défaut : *gamma0* = 0,1).
- *alphad2* — Hyper paramètre pour échantillonner un paramètre qui représente le signal-sur-bruit attendue (notée δ^2). Il est échantillonné selon une loi inverse gamma : rinvgamma (1, shape = alphad2, scale = betad2), (facultatif, par défaut : *alphad2* = 2).
- *betad2* — Hyper paramètre pour échantillonner un paramètre qui représente le signal-sur-bruit attendu (notée δ^2). Il est échantillonné selon une loi gamma inverse: rinvgamma (1, shape = alphad2, scale = betad2), (facultatif, par défaut : *betad2* = 0,2).
- *silent* — Booléen, si TRUE les messages sont imprimés le long de la procédure (facultatif, par défaut : *silent* = FALSE).

4.1.3 Sorties de la fonction

- *Samples* — Résultats obtenus à chaque itération de la procédure RJ-MCMC. *Samples* est une liste composée des éléments suivants:
  (1) *Samples$Edges* : un vecteur avec le nombre de parents pour le gène cible analysé pour chaque itération.
  (2) *Samples$coeff* : une matrice avec en ligne les différentes itérations et en colonnes les valeurs des coefficients de régression.
  (3) *Samples$variance* : un vecteur avec les différentes itérations des valeurs de variance du bruit des données.
- *Counters* — Les résultats obtenus à chaque itération de la procédure RJ-MCMC. *Counters* est une liste composée des éléments suivants:
  (1) *Counters$EdgesMoveCount* : Nombre de propositions de modifications au cours des itérations (la naissance d'une nouvelle arête entre les gènes parents et le gène cible, la mort d'une arête existante ou la mise à jour des coefficients de régression pour les arêtes existantes).
  (2) *Counters$EdgesMovesAcceptationPrct*: Pourcentage des modifications acceptées au cours des itérations (la naissance d'une nouvelle arête entre les gènes parents et le gène cible,

la mort d'une arête existante ou la mise à jour des coefficients de régression pour les arêtes existantes).

(3) *Counters$iterations* : Nombre total d'itérations générées par la procédure.

- *PostDist* — *PostDist* est une liste composée des éléments suivants :
  (1) *PostDist$edgesPostDist* : Un vecteur contenant la distribution à postériori approximative pour les arêtes entrantes.
  (2) *PostDist$edgesCoeff* : Un vecteur contenant les coefficients estimés pour les arêtes entrantes.
- *Network* — Un tableau contenant les informations pour tracer (voir la fonction *traceNetworks*) le réseau estimé avec la procédure *subnet*.
- *GLOBvar* — Une liste des paramètres utilisés dans la procédure *subnet*.
- *HYPERvar* — Une liste des hyper paramètres utilisés dans la procédure *subnet*.
- *targetData* — Le vecteur *targetData* donné en entrée de la procédure *subnet*.
- *parentData* — La matrice *parentData* donnée en entrée de la procédure *subnet*.

4.1.4 Description de l'implémentation

La fonction *subnet* vérifie dans un premier temps que les données d'expression des gènes parents et du gène cible ne contiennent pas de données manquantes et ne continue que quand ce n'est pas le cas. Elle vérifie ensuite que les paramètres *targetData* et *parentData* soient des vecteurs ou des matrices et si ce n'est pas le cas, elle est stoppée. Dans le cas où *parentData* est un vecteur, ce qui signifie qu'il n'y a qu'un seul gène parent pour le gène cible analysé, il est convertit en une matrice.

La fonction vérifie également que le nombre de mesures répétées est identique pour chaque gène.

Suivant la valeur du paramètre *silent*, des messages sont imprimés au cours de la procédure.

La fonction définit ensuite la valeur q du nombre de parents en calculant le nombre ligne de la matrice *parentData,* le nombre maximal de parents possibles pour le gène cible analysé en limitant ce nombre à 15, la valeur de *burn_in* à 25% des itérations si celle-ci n'est pas renseignée.

Elle créé les liste *GLOBvar* et *HYPERvar* contenant les variables et hyper paramètres utilisés dans toutes les autres fonctions.

A l'aide d'une fonction *buildXY*, qui prend en paramètre *targetData*, *parentData* et le nombre de répétition, un vecteur y, contenant les mesures d'expression du gène cible, et une matrice x, contenant les mesures d'expression des gènes parents et une colonne composée de « 1 » pour la constante du modèle de régression sont construits.

Afin de calculer les valeurs $E_1(G_i)$ et $E_2(G_i)$, la fonction *makeConfigParents* est utilisée. Cette dernière prend en paramètre le nombre de parents possibles q et crée une liste de vecteurs contenant toutes les combinaisons de réseau de régulation possibles de 0 à 3 parents parmi les q parents possibles.

La procédure RJ-MCMC débute avec l'initialisation des paramètres et valeurs nécessaires à la simulation. Pour cela, une fonction *init* a été implémentée

Les *niter* itérations sont ensuite effectuées à l'aide de la fonction *main*.

Suivant les valeurs affectées à *saveEstimations*, *saveIterations* et *savePictures*, la fonction *subnet* enregistre ou non les résultats de l'algorithme dans un fichier.

Finalement, la fonction *subnetAnalysis* est appelée pour estimer, à partir des itérations de l'algorithme RJ-MCMC, le réseau de régulation.

4.2 La fonction *bdu*

4.2.1 Description de la fonction

Cette fonction permet d'effectuer un mouvement de naissance d'arête, de mort d'arête ou de mise à jour des paramètres du modèle de régression à chaque itération du processus RJ-MCMC engendrée par la fonction *main*.

4.2.2 Paramètres de la fonction

- *x* — Matrice contenant en ligne les données d'expression des gènes parents et en colonne les mesures répétées, en ajoutant une dernière colonne de 1 pour la constante du modèle de régression.
- *y* — Vecteur contenant les données d'expression du gène cible.
- *S* — Vecteur indiquant les interactions entre le gène cible analysé et les parents proposés (0 si absence d'interaction et 1 sinon), avec pour dernier coefficient, 1 pour la constante du modèle de régression.
- *smax* — Le nombre maximal d'arêtes possibles.
- *c* — Constante utilisée dans la fonction *computeRho3*.
- *q* — Nombre de parents possibles pour le gène cible analysé.
- *b1, b2* — Vecteur des connaissances a priori sur les interactions entre le gène cible analysé et les parents potentiels. Les coefficients du vecteur sont compris entre 0 et 1 selon la connaissance a priori sur la présence ou l'absence d'arête entre les gènes.
- *beta* — Vecteur contenant les valeurs beta1 et beta2, hyper paramètres des connaissances a priori b1 et b2, qui suivent une loi uniforme d'intervalle [0,b].
- *delta2* — Paramètre qui représente le signal-sur-bruit attendue (notée δ^2). Il est échantillonné selon une loi inverse gamma : rinvgamma (1, shape = alphad2, scale = betad2).

- *lambda* : Hyper paramètre pour échantillonner le nombre de parents. La distribution a priori du nombre de parents est une distribution de Poisson tronquée de moyenne lambda et de maximum 5. La distribution a priori de lambda est une distribution gamma : rgamma(1, shape= alphalbd, rate=betalbd).
- *Sig2* : Variance du bruit dans le modèle de régression définissant le réseau de régulation. La distribution a priori de la variance du bruit est une distribution inverse gamma avec le paramètre de forme v0/2 et le paramètre d'intensité gamma0/2 : rinvgamma (1, shape = v0 / 2, scale = gamma0).
- *v0* : Hyper paramètre pour échantillonner la variance du bruit (notée δ^2) dans le modèle de régression définissant le réseau de régulation. La distribution a priori de la variance du bruit est une distribution inverse gamma avec le paramètre de forme *v0*/2 et le paramètre d'intensité *gamma0*/2 : rinvgamma (1, shape = v0 / 2, scale = gamma0/2) (facultatif, par défaut : *v0* = 1).
- *gamma0* : Hyper paramètre pour échantillonner la variance du bruit (notée δ^2) dans le modèle de régression définissant le réseau de régulation. La distribution a priori de la variance du bruit est une distribution inverse gamma avec le paramètre de forme *v0*/2 et le paramètre d'intensité *gamma0*/2 : rinvgamma (1, shape = v0 / 2, scale = gamma0/2) (facultatif, par défaut : *gamma0* = 0,1).

4.2.3 Sorties de la fonction

- *newS* : Nouveau vecteur indiquant les interactions entre gène cible analysé et les parents après le mouvement proposé (naissance d'une arête, mort d'une arête ou mise à jour des paramètres du modèle de régression).
- *newB* : Vecteur contenant les nouveaux coefficients du modèle de régression.
- *move* : Variable décrivant le type de déplacement proposé (1 = naissance d'arête, 2 = mort d'arête, 3 = mise à jour des coefficients).
- *accept* : Booléen indiquant si le mouvement est accepté ou non (1 si il est accepté, 0 sinon).

4.2.4 Description de l'implémentation

La fonction *bdu* commence par calculer $E_1(G_{i,old})$ et $E_2(G_{i,old})$, les concordances entre le gène étudié et la connaissance a priori sur celui-ci.

Elle calcule ensuite la matrice de projection avec l'arête courante P_i puis les probabilités des différents mouvements (naissance d'arête, mort d'arête ou mise à jour des paramètres du modèle de régression).

L'algorithme permettant de choisir le mouvement commence avec l'échantillonnage d'une variable u qui suit une loi uniforme d'intervalle [0,1].

Si u est inférieur à la probabilité d'une naissance, alors la valeur 1 est affectée au paramètre *move* et la position d'une nouvelle arête est échantillonnée parmi les parents possibles. Un nouveau vecteur temporaire $G_{i,tmp}$ est créé prenant en compte l'arête supplémentaire.

La probabilité d'acceptation d'une naissance d'arête est ensuite calculée afin de déterminer si le mouvement est accepté ou non. Si le mouvement est accepté le paramètre *accept* prend la valeur 1 et $G_{i,new} = G_{i,tmp}$ sinon $G_{i,new} = G_{i,old}$.

On procède de même pour la naissance d'une arête et la mise à jour des coefficients de la régression.

Dans tous les cas, les coefficients sont mis à jour à la fin de la fonction grâce à la fonction *sampleBxy*.

Conclusion

L'objectif de ce stage était d'étendre une méthode d'inférence de réseau Bayésien établi par Lèbre et al. (2010) [2] pour proposer un modèle permettant de prendre en compte différentes sources de données a priori. Pour cela, nous avons implémenté une fonction générant un échantillon RJ-MCMC afin d'estimer le réseau de régulation des gènes. L'originalité de l'approche a consisté à modifier les lois a priori sur les arêtes du réseau pour tenir compte d'a priori issus d'autres données biologiques. Dans la version initiale du modèle de réseau proposé par Lèbre et al. (2010) [2], les réseaux considérés sont pénalisés quand le nombre d'arêtes augmente, équiprobables à nombre d'arêtes égal. Pour cela nous avons adapté la méthode proposée par Werlhi et Husmeier (2007) [3] pour le modèle de réseau initial en utilisant une loi a priori exponentielle définie par un paramètre β_i pour chaque jeu de données a priori i. La partie essentielle a consisté à calculer les probabilités d'acceptation de chaque mouvement proposée de la méthode RJ-MCMC : ajout d'une arête, suppression d'une arête, modification des hyper paramètres β_i.

Afin de valider ce modèle, il reste à effectuer des analyses sur des données simulées dont on connait la structure d'interaction pour observer la convergence de la méthode et tester la bonne estimation des coefficients qui évaluent l'intensité du couplage.

Bibliographie

[1] *Nature Genetics* (2000), vol. 25, no 2, pp. 235-238

[2] Lebre, S., Becq, J., Devaux, F., Stumpf, M.P.H., Lelandais , G. (2010). Statistical inference of the time-varying structure of gene-regulation networks, *BMC Systems Biology*, 4:130.

[3] Werhli, A.V. and Husmeier, D. (2007). Reconstructing gene regulatory networks with Bayesian networks by combining expression data with multiple sources of prior knowledge, Statistical Applications in Genetics and Molecular Biology, Vol. 6 : Iss. 1, Article 15.

[4] Lebre, S., Dondelinger, F., and Husmeier, D., Nonhomogeneous dynamic Bayesian networks in systems biology, 2012

[5] Green, PJ (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*

Glossaire

Pour faciliter la lecture et la compréhension de ce manuscrit, certains termes techniques sont rappelés ici :

BFO : Bioinformatique théorique, Fouille de données et Optimisation stochastique

ChIP-chip : « ChIP » pour Chromatin Immuno Precipitation et « chip » pour puce en anglais

ARTIVA : Auto Regressive TIme VArying

MCMC : Markov Chain Monte Carlo

RJ- MCMC : Méthode MCMC à sauts réversibles