

STA305 : Partie II

Calcul numérique pour l'analyse bayésienne

Boris Hejblum

ISPED M2 Biostatistique, Université de Bordeaux
Inserm BPH U1219 / Inria BSO, équipe SISTM

boris.hejblum@u-bordeaux.fr
<https://borishejblum.science>

Objectifs du cours

- 1 Comprendre les algorithmes d'échantillonnage et leur utilité
- 2 Comprendre le fonctionnement des algorithmes MCMC
- 3 Savoir utiliser le logiciel JAGS et en interpréter les sorties

Introduction

Une difficile estimation de la loi *a posteriori*

Intégration numérique – I

Applications réelles : $\boldsymbol{\theta} = (\theta_1, \dots, \theta_d)$

⇒ la loi *a posteriori* conjointe des d paramètres

⚠ difficile à calculer :

- vraisemblance complexe
- constante d'intégration $f(\mathbf{y}) = \int_{\Theta^d} f(\mathbf{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}) d\boldsymbol{\theta}$
- ...

Solution analytique rarement disponible

⇒ calcul numérique : intégrale de multiplicité d
 – difficile lorsque d est grand (les problèmes numériques apparaissent dès que $d > 4$)

Intégration numérique – II

Des problèmes peuvent se poser même en dimension 1 !

Exemple :

Soit un échantillon x_1, \dots, x_n *iid* selon une loi de Cauchy $\mathcal{C}(\theta, 1)$ avec l'*a priori* $\pi(\theta) = \mathcal{N}(\mu, \sigma^2)$ (μ et σ connus)

$$\begin{aligned} p(\theta|x_1, \dots, x_n) &\propto f(x_1, \dots, x_n|\theta)\pi(\theta) \\ &\propto e^{-\frac{(\theta-\mu)^2}{2\sigma^2}} \prod_{i=1}^n (1 + (x_i - \theta)^2)^{-1} \end{aligned}$$

⚠ La constante de normalisation ne peut se calculer analytiquement \Rightarrow impossible de donner une expression analytique pour cette loi *a posteriori*

Distributions marginales *a posteriori*

Objectif : tirer des conclusions à partir de cette distribution *a posteriori* conjointe

⇒ probabilité de toutes les valeurs possibles pour chaque paramètre (i.e. leurs distributions marginales, unidimensionnelles)

⚠ Reconstituer toute la densité *a posteriori* **numériquement** nécessite le calcul d'intégrales multidimensionnelles **pour chaque valeur possible du paramètre**

⇒ un calcul suffisamment précis de ces intégrales paraît impossible

Distributions marginales *a posteriori*

Objectif : tirer des conclusions à partir de cette distribution *a posteriori* conjointe

⇒ probabilité de toutes les valeurs possibles pour chaque paramètre (i.e. leurs distributions marginales, unidimensionnelles)

⚠ Reconstituer toute la densité *a posteriori* **numériquement** nécessite le calcul d'intégrales multidimensionnelles **pour chaque valeur possible du paramètre**

⇒ un calcul suffisamment précis de ces intégrales paraît impossible

Algorithmes basés sur des **simulations d'échantillonnage**
en particulier les méthodes de **Monte-Carlo par chaînes de Markov**
(*Markov chain Monte Carlo* – MCMC)

Solutions computationnelles

Théorème de Bayes \Rightarrow loi *a posteriori*

Solutions computationnelles

Théorème de Bayes \Rightarrow loi *a posteriori*

⚠ en pratique :

- une expression analytique rarement possible (cas bien particuliers)
- calcul de l'intégrale au dénominateur est souvent très difficile

Solutions computationnelles

Théorème de Bayes \Rightarrow loi *a posteriori*

⚠ en pratique :

- une expression analytique rarement possible (cas bien particuliers)
- calcul de l'intégrale au dénominateur est souvent très difficile

Comment estimer la distribution *a posteriori* ?

\Rightarrow générer un échantillon distribué selon la loi *a posteriori*

- **méthodes d'échantillonnage** directes
- méthodes de **Monte-Carlo par chaînes de Markov**

Méthode de Monte-Carlo

Monte-Carlo : von Neumann & Ulam

(*Los Alamos Scientific Laboratory* – 1955)

⇒ utiliser des nombres aléatoires pour estimer des quantités difficiles (ou impossible) à calculer analytiquement

Méthode de Monte-Carlo

Monte-Carlo : von Neumann & Ulam

(Los Alamos Scientific Laboratory – 1955)

⇒ utiliser des nombres aléatoires pour estimer des quantités difficiles (ou impossible) à calculer analytiquement

- **Loi des Grands Nombres**
- **échantillon dit « de Monte-Carlo »**

⇒ calculer divers fonctionnelles à partir de la distribution de l'échantillon

Exemple : On veut calculer $\mathbb{E}[f(X)] = \int f(x)p_X(x)dx$

Si $x_i \stackrel{iid}{\sim} p_X$, $\mathbb{E}[f(X)] = \frac{1}{N} \sum_{i=1}^N f(x_i)$ (LGN)

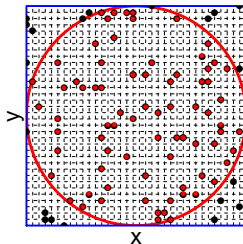
⇒ si on sait échantillonner selon $p(x)$, on peut ainsi estimer $\mathbb{E}[f(X)] \dots$

Méthode de Monte-Carlo : illustration

Estimation de π :

Méthode de Monte-Carlo : illustration

Estimation de π :



Une roulette de casino (à Monte-Carlo?)

Un cible quadrillée en 36×36

- 1 La probabilité d'être dans le cercle plutôt que dans le carré : $p_C = \frac{\pi R^2}{(2R)^2} = \frac{\pi}{4}$
- 2 n points $\{(x_{11}, x_{21}), \dots, (x_{1n}, x_{2n})\} = \{P_1, \dots, P_n\}$ dans le repère 36×36 (à l'aide de la roulette qui génère les coordonnées une à une)
- 3 Compter le nombre de points dans le cercle

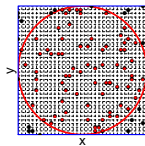
⇒ Calculer le ratio (probabilité estimée d'être dans le cercle) : $\hat{p}_C = \frac{\sum P_i \in \text{cercle}}{n}$

Méthode de Monte-Carlo : illustration

Estimation de π :



Une roulette de casino (à Monte-Carlo?)



Un cible quadrillée en 36×36

Si $n = 1000$ et que l'on trouve 765 points dans le cercle : $\hat{\pi} = 4 \times \frac{765}{1000} = 3,06$

On peut améliorer l'estimation en augmentant :

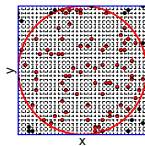
- la résolution de la grille et aussi
- le nombre de points n : $\lim_{n \rightarrow +\infty} \hat{p}_C = p_C = \pi$ (LGV)

Méthode de Monte-Carlo : illustration

Estimation de π :



Une roulette de casino (à Monte-Carlo?)



Un cible quadrillée en 36×36

Si $n = 1000$ et que l'on trouve 765 points dans le cercle : $\hat{\pi} = 4 \times \frac{765}{1000} = 3,06$

On peut améliorer l'estimation en augmentant :

- la résolution de la grille et aussi
- le nombre de points n : $\lim_{n \rightarrow +\infty} \hat{p}_C = p_C = \pi$ (LGV)

échantillon de Monte-Carlo \Rightarrow calculer de nombreuses fonctionnelles
e.g. $\pi = 4$ fois la probabilité d'être dans le cercle

Méthodes d'échantillonnage directes

Nombres aléatoires & pseudo-aléatoires

Il existe plusieurs manières de générer des nombres dits « aléatoires » selon des lois connues

NB : Les programmes informatiques ne génèrent pas des nombres totalement aléatoire

On parle plutôt de nombres **pseudo-aléatoires**, qui semblent aléatoires mais sont en réalité générés selon un processus déterministe (qui dépend notamment d'un paramètre appelé « **graine** » – *seed*).

Génération d'un échantillon uniforme

Algorithme congruentiel linéaire : génération d'un échantillon pseudo-aléatoire selon la loi uniforme sur $[0;1]$ (Lehmer, 1948)


- 1 Générer une suite d'entiers y_n tel que :

$$y_{n+1} = (ay_n + b) \bmod m$$

- 2 $x_{n+1} = \frac{y_{n+1}}{m-1}$

Choisir a , b et m de manière à ce que y_n ait une période très longue et que (x_1, \dots, x_n) puisse être considéré comme *iid*

avec y_0 la graine

Remarque : $0 \leq y_n \leq m-1 \Rightarrow$ en pratique m très grand (ex. 2^{19937} , le défaut dans  qui utilise l'algorithme Mersenne-Twister)

Dans la suite la génération de nombre pseudo-aléatoires selon la loi uniforme sur $[0;1]$ sera considérée comme fiable et utilisé par les différents algorithmes d'échantillonnage

Autres distributions usuelles

On s'appuie sur les **relations entre les différentes lois usuelles** en partant de $U_i \sim \mathcal{U}_{[0;1]}$

Autres distributions usuelles

On s'appuie sur les **relations entre les différentes lois usuelles** en partant de $U_i \sim \mathcal{U}_{[0;1]}$

Loi binomiale $Bin(n, p)$:

$$Y_i = \mathbb{1}_{U_i \leq p} \sim \text{Bernoulli}(p)$$

$$X = \sum_{i=1}^n Y_i \sim Bin(n, p)$$

Autres distributions usuelles

On s'appuie sur les **relations entre les différentes lois usuelles** en partant de $U_i \sim \mathcal{U}_{[0;1]}$

Loi binomiale $Bin(n, p)$:

$$Y_i = \mathbb{1}_{U_i \leq p} \sim \text{Bernoulli}(p)$$

$$X = \sum_{i=1}^n Y_i \sim Bin(n, p)$$

Loi normale $\mathcal{N}(0, 1)$ (algorithme de Box-Müller) :

U_1 et U_2 sont 2 variables uniformes $[0; 1]$ indépendantes

$$Y_1 = \sqrt{-2 \log U_1} \cos(2\pi U_2)$$

$$Y_2 = \sqrt{-2 \log U_1} \sin(2\pi U_2)$$

⇒ Y_1 & Y_2 sont indépendantes et suivent chacune la loi $\mathcal{N}(0, 1)$

Méthode par inversion

Définition : Pour une fonction F définie sur \mathbb{R} , on définit son **inverse généralisée** par : $F^{-1}(u) = \inf\{x \text{ tq } F(x) > u\}$

Méthode par inversion

Définition : Pour une fonction F définie sur \mathbb{R} , on définit son **inverse généralisée** par : $F^{-1}(u) = \inf\{x \text{ tq } F(x) > u\}$

Propriété : Soit

- F la fonction de répartition d'une distribution de probabilité
- U une variable aléatoire suivant une loi uniforme sur $[0; 1]$

Alors $F^{-1}(U)$ définit une variable aléatoire ayant pour fonction de répartition F .

- Si
- ① on connaît la fonction de répartition F de la loi selon laquelle simuler
 - ② on est capable d'inverser F
- ⇒ on peut alors générer un échantillon suivant cette loi à partir d'un échantillon uniforme sur $[0; 1]$

Méthode par inversion : illustration

Exemple : On veut générer un échantillon suivant la loi exponentielle de paramètre λ

Méthode par inversion : illustration

Exemple : On veut générer un échantillon suivant la loi exponentielle de paramètre λ

- la densité de la loi exponentielle est $f(x) = \lambda \exp(-\lambda x)$
- la fonction de répartition (son intégrale) est donc $F(x) = 1 - \exp(-\lambda x)$

Posons $F(x) = u$

On obtient alors $x = \dots$

Méthode par inversion : illustration

Exemple : On veut générer un échantillon suivant la loi exponentielle de paramètre λ

- la densité de la loi exponentielle est $f(x) = \lambda \exp(-\lambda x)$
- la fonction de répartition (son intégrale) est donc $F(x) = 1 - \exp(-\lambda x)$

Posons $F(x) = u$

On obtient alors $x = -\frac{1}{\lambda} \log(1 - u)$

⇒ et si $U \sim \mathcal{U}_{[0;1]}$, alors $X = F^{-1}(U) = -\frac{1}{\lambda} \log(1 - U) \sim E(\lambda)$

Méthode d'acceptation-rejet : algorithme

Utiliser une **loi instrumentale** g (dont on sait échantillonner selon la loi)
⇒ afin d'échantillonner selon la loi cible f

Le principe générale est de **choisir g proche de f** et de proposer des échantillons selon g , d'en accepter certains et d'en rejeter d'autres afin d'obtenir un échantillon suivant la loi de f .

Méthode d'acceptation-rejet : algorithme

Utiliser une **loi instrumentale** g (dont on sait échantillonner selon la loi)
⇒ afin d'échantillonner selon la loi cible f

Le principe générale est de **choisir** g **proche de** f et de proposer des échantillons selon g , d'en accepter certains et d'en rejeter d'autres afin d'obtenir un échantillon suivant la loi de f .

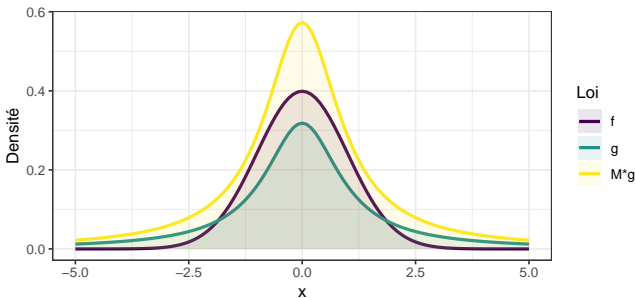
Soit une loi d'intérêt de densité f .

Soit une loi de proposition de densité g (à partir de laquelle on sait échantillonner) telle que, pour tout x : $f(x) \leq Mg(x)$

- 1 Générer $x_i \sim g$ et $u_i \sim \mathcal{U}_{[0;1]}$
- 2 Si $u_i \leq \frac{f(x_i)}{Mg(x_i)}$ on **accepte** le tirage :
 $y_i := x_i$
sinon on le **rejette** et on retourne en 1.

⇒ $(y_1, \dots, y_n) \stackrel{iid}{\sim} f$

Acceptation-rejet : importance de la loi de proposition



Exemple de loi de proposition et de loi cible pour l'algorithme d'acceptation-rejet

Remarque : Plus M est petit, plus le taux de rejet est faible

⇒ plus l'algorithme est efficace (moins d'itération pour un échantillon de taille n)

Donc on souhaite g le plus proche possible de f !

⚠ g aura nécessairement des queues plus lourdes que la loi cible

⇒ lorsque le nombre de paramètres augmente, le taux d'acceptation décroît très rapidement (*fléau de la dimension*)

Exercices

Exercice 1 : Construire un pseudo-échantillon de taille n selon la loi discrète suivante (multinomiale à m éléments $\{x_1, \dots, x_m\}$) :

$$P(X = x) = p_1 \delta_{x_1}(x) + p_2 \delta_{x_2}(x) + \dots + p_m \delta_{x_m}(x) \quad \text{avec} \quad \sum_{i=1}^m p_i = 1 \quad \text{et} \quad \delta_a(x) = \mathbb{1}_{\{x=a\}}$$

Exercice 2 : Grâce à la méthode par inversion, générer un pseudo-échantillon de taille suivant une loi de Cauchy (dont la densité est $f(x) = \frac{1}{\pi(1+x^2)}$), sachant que $\arctan'(x) = \frac{1}{(1+x^2)}$ et $\lim_{x \rightarrow -\infty} \arctan(x) = -\frac{\pi}{2}$.

Exercice 3 : Écrire un algorithme d'acceptation-rejet pour simuler la réalisation d'un pseudo-échantillon de taille n d'une loi normale $N(0, 1)$ en utilisant une loi de Cauchy comme proposition. Trouvez la valeur de M optimale.

Algorithmes MCMC

Définition d'une chaîne de Markov

Chaîne de Markov : processus stochastique à temps discret

Définition : une chaîne de Markov est une suite de variable aléatoire $X_0, X_1, X_2, X_3, \dots$ (toutes définies sur le même espace) possédant la **propriété de Markov** (« sans mémoire ») :

$$p(X_i = x | X_0 = x_0, X_1 = x_1, \dots, X_{i-1} = x_{i-1}) = p(X_i = x | X_{i-1} = x_{i-1})$$

L'ensemble E des valeurs possible pour X_i est appelé **espace d'état**

Déterminée par 2 paramètres :

- 1 sa distribution initiale $p(X_0)$
- 2 son noyau de transition $T(x, A) = p(X_i \in A | X_{i-1} = x)$

NB : on ne va considérer ici que des chaînes de Markov **homogènes** :

$$p(X_{i+1} = x | X_i = y) = p(X_i = x | X_{i-1} = y)$$

Propriétés des chaînes de Markov

Propriété : Une chaîne de Markov est dite **irréductible** : si tous les ensembles de probabilité non nulle peuvent être atteints à partir de tout point de départ (i.e. tout état est accessible à partir de n'importe quel autre).

Propriétés des chaînes de Markov

Propriété : Une chaîne de Markov est dite **irréductible** : si tous les ensembles de probabilité non nulle peuvent être atteints à partir de tout point de départ (i.e. tout état est accessible à partir de n'importe quel autre).

Propriété : Une chaîne de Markov est dite **récurrente** si les trajectoires (X_j) passent une infinité de fois dans tout ensemble de probabilité non nulle de l'espace d'état.

Propriétés des chaînes de Markov

Propriété : Une chaîne de Markov est dite **irréductible** : si tous les ensembles de probabilité non nulle peuvent être atteints à partir de tout point de départ (i.e. tout état est accessible à partir de n'importe quel autre).

Propriété : Une chaîne de Markov est dite **récurrente** si les trajectoires (X_j) passent une infinité de fois dans tout ensemble de probabilité non nulle de l'espace d'état.

Propriété : Une chaîne de Markov est dite **apériodique** si rien n'induit un comportement périodique des trajectoires.

Loi stationnaire & théorème ergodique

Définition : Une distribution de probabilité \tilde{p} est appelée **loi invariante** (ou **loi stationnaire**) pour une chaîne de Markov si elle vérifie la propriété suivante :

$$\text{si } X_i \sim \tilde{p}, \text{ alors } X_{i+j} \sim \tilde{p} \quad \forall j \geq 1$$

Remarque : Une chaîne de Markov peut admettre plusieurs lois stationnaires.

Loi stationnaire & théorème ergodique

Définition : Une distribution de probabilité \tilde{p} est appelée **loi invariante** (ou **loi stationnaire**) pour une chaîne de Markov si elle vérifie la propriété suivante :

$$\text{si } X_i \sim \tilde{p}, \text{ alors } X_{i+j} \sim \tilde{p} \quad \forall j \geq 1$$

Remarque : Une chaîne de Markov peut admettre plusieurs lois stationnaires.

Théorème ergodique (espace infini) : Une chaîne de Markov irréductible et récurrente positive (i.e. le temps de retour moyen est fini) admet une unique loi de probabilité invariante \tilde{p} . Si cette chaîne de Markov est de plus apériodique, alors elle converge en loi vers \tilde{p} .

Exemple de chaîne de Markov à espace d'état discret – I

Doudou (le hamster) suit à chaque minute une chaîne de Markov discrète à 3 états :

D dormir

M manger

E faire de l'exercice

⇒ son état dans une minute ne dépend que de son état actuel (et pas des minutes précédentes)

La matrice des probabilité de transition soit alors la suivante :

$$P = \begin{pmatrix} X_i/X_{i+1} & D & M & E \\ D & 0.9 & 0.05 & 0.05 \\ M & 0.7 & 0 & 0.3 \\ E & 0.8 & 0 & 0.2 \end{pmatrix}$$

Exemple de chaîne de Markov à espace d'état discret – I

Doudou (le hamster) suit à chaque minute une chaîne de Markov discrète à 3 états :

D dormir

M manger

E faire de l'exercice

⇒ son état dans une minute ne dépend que de son état actuel (et pas des minutes précédentes)

La matrice des probabilité de transition soit alors la suivante :

$$P = \begin{pmatrix} X_i / X_{i+1} & D & M & E \\ D & 0.9 & 0.05 & 0.05 \\ M & 0.7 & 0 & 0.3 \\ E & 0.8 & 0 & 0.2 \end{pmatrix}$$

- 1) Selon vous, la chaîne est-elle irréductible ? Récurrente ? Apériodique ?
- 2) Supposons que Doudou dorme. Que fait-il 2 min après ? et 10 min après ?
- 3) Supposons maintenant qu'il fasse de l'exercice. Que fait-il 10 min après ?

Exemple de chaîne de Markov à espace d'état discret – II

1) Selon vous, la chaîne est-elle irréductible? Récurrente? Apériodique?

...

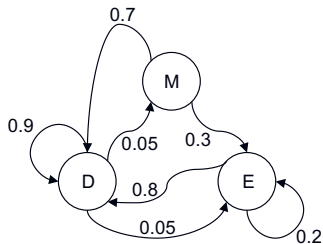
2) Supposons que Doudou dorme. Que fait-il 2 min après? et 10 min après?

...

3) Supposons maintenant qu'il fasse de l'exercice. Que fait-il 10 min après?

Exemple de chaîne de Markov à espace d'état discret – II

1) Selon vous, la chaîne est-elle irréductible? Récurrente? Apériodique?



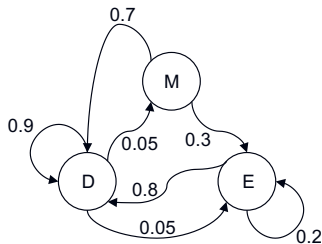
2) Supposons que Doudou dorme. Que fait-il 2 min après? et 10 min après?

...

3) Supposons maintenant qu'il fasse de l'exercice. Que fait-il 10 min après?

Exemple de chaîne de Markov à espace d'état discret – II

1) Selon vous, la chaîne est-elle irréductible? Récurrente? Apériodique?



2) Supposons que Doudou dorme. Que fait-il 2 min après? et 10 min après?

$$x_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}^T \quad x_2 = x_0 P^2 = \begin{pmatrix} 0.885 \\ 0.045 \\ 0.070 \end{pmatrix}^T \quad x_{10} = x_2 P^8 = x_0 P^{10} = \begin{pmatrix} 0.884 \\ 0.044 \\ 0.072 \end{pmatrix}^T$$

Exemple de chaîne de Markov à espace d'état discret – II

3) Supposons maintenant qu'il fasse de l'exercice. Que fait-il 10 min après ?

...

Exemple de chaîne de Markov à espace d'état discret – II

3) Supposons maintenant qu'il fasse de l'exercice. Que fait-il 10 min après ?

$$x_0 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^T \quad x_{10} = x_0 P^{10} = \begin{pmatrix} 0.884 \\ 0.044 \\ 0.072 \end{pmatrix}^T$$

Ici la loi est apériodique, récurrente et irréductible, il y a donc une loi stationnaire : $\tilde{p} = \tilde{p}P$.

Algorithmes MCMC : principe général

Approximer une intégrale (ou d'une autre fonctionnelle)
d'une distribution d'intérêt

Algorithmes MCMC : principe général

Approximer une intégrale (ou d'une autre fonctionnelle)
d'une distribution d'intérêt

⇒ générer une chaîne de Markov dont la loi stationnaire est la loi *a posteriori*, puis d'y appliquer la méthode de Monte-Carlo

Algorithmes MCMC : principe général

Approximer une intégrale (ou d'une autre fonctionnelle)
d'une distribution d'intérêt

⇒ générer une chaîne de Markov dont la loi stationnaire est la loi *a posteriori*, puis d'y appliquer la méthode de Monte-Carlo

Nécessite une **double convergence** :

- 1 la convergence de la chaîne de Markov vers sa distribution stationnaire : $\forall X_0, X_n \xrightarrow[n \rightarrow +\infty]{\mathcal{L}} \tilde{p}$

Algorithmes MCMC : principe général

Approximer une intégrale (ou d'une autre fonctionnelle)
d'une distribution d'intérêt

⇒ générer une chaîne de Markov dont la loi stationnaire est la loi *a posteriori*, puis d'y appliquer la méthode de Monte-Carlo

Nécessite une **double convergence** :

- 1 la convergence de la chaîne de Markov vers sa distribution

$$\text{stationnaire : } \forall X_0, X_n \xrightarrow[n \rightarrow +\infty]{\mathcal{L}} \tilde{p}$$

- 2 la convergence de Monte-Carlo, une fois la distribution stationnaire atteinte :

$$\frac{1}{N} \sum_{i=1}^N f(X_{n+i}) \xrightarrow[N \rightarrow +\infty]{} \mathbb{E}[f(X)]$$

Algorithmes MCMC : principe général

Approximer une intégrale (ou d'une autre fonctionnelle)
d'une distribution d'intérêt

⇒ générer une chaîne de Markov dont la loi stationnaire est la loi *a posteriori*, puis d'y appliquer la méthode de Monte-Carlo

Nécessite une **double convergence** :

- 1 la convergence de la chaîne de Markov vers sa distribution stationnaire : $\forall X_0, X_n \xrightarrow[n \rightarrow +\infty]{\mathcal{L}} \tilde{p}$

- 2 la convergence de Monte-Carlo, une fois la distribution stationnaire atteinte :

$$\frac{1}{N} \sum_{i=1}^N f(X_{n+i}) \xrightarrow[N \rightarrow +\infty]{} \mathbb{E}[f(X)]$$

convergence de la chaîne de Markov

$$\overbrace{X_0 \rightarrow X_1 \rightarrow X_2 \rightarrow \dots \rightarrow X_n}$$

échantillon de Monte-Carlo

$$\rightarrow \overbrace{X_{n+1} \rightarrow X_{n+2} \rightarrow \dots \rightarrow X_{n+N}}$$

Schéma général des algorithmes MCMC

Les algorithmes MCMC s'appuient sur une approche d'acceptation-rejet

- 1 Initialiser $x^{(0)}$
- 2 Pour $t = 1, \dots, n + N$:
 - a Proposer un nouveau candidat $y^{(t)} \sim q(y^{(t)} | x^{(t-1)})$
 - b Accepter $y^{(t)}$ avec la probabilité $\alpha(x^{(t-1)}, y^{(t)})$:
⇒ si $t > n$, « sauver » $x^{(t)}$ (pour ensuite calculer la fonctionnelle d'intérêt)

avec q la loi instrumentale de proposition
et α la probabilité d'acceptation

Schéma général des algorithmes MCMC

Les algorithmes MCMC s'appuient sur une approche d'acceptation-rejet

- 1 Initialiser $x^{(0)}$
- 2 Pour $t = 1, \dots, n + N$:
 - a Proposer un nouveau candidat $y^{(t)} \sim q(y^{(t)} | x^{(t-1)})$
 - b Accepter $y^{(t)}$ avec la probabilité $\alpha(x^{(t-1)}, y^{(t)})$:
⇒ si $t > n$, « sauver » $x^{(t)}$ (pour ensuite calculer la fonctionnelle d'intérêt)

avec q la loi instrumentale de proposition
et α la probabilité d'acceptation

Choix de la loi instrumentale

Pas de choix absolument optimal pour la loi instrumentale de proposition q

⇒ infinité de lois possibles : certaines meilleures que d'autres

Choix de la loi instrumentale

Pas de choix absolument optimal pour la loi instrumentale de proposition q

⇒ infinité de lois possibles : certaines meilleures que d'autres

Afin de garantir la convergence vers la loi cible \tilde{p} :

- le support de q doit contenir le support \tilde{p}
- q ne doit pas générer de valeurs périodiques

Choix de la loi instrumentale

Pas de choix absolument optimal pour la loi instrumentale de proposition q

⇒ infinité de lois possibles : certaines meilleures que d'autres

Afin de garantir la convergence vers la loi cible \tilde{p} :

- le support de q doit contenir le support \tilde{p}
- q ne doit pas générer de valeurs périodiques

NB : *Idéalement* on choisit q de manière à ce que son **calcul** soit **simple** (et **rapide**)

Algorithme de Metropolis-Hastings

- 1 Initialiser $x^{(0)}$
- 2 Pour $t = 1, \dots, n + N$:
 - a Proposer $y^{(t)} \sim q(y^{(t)} | x^{(t-1)})$
 - b Calculer la probabilité d'acceptation
$$\alpha^{(t)} = \min \left\{ 1, \frac{\tilde{p}(y^{(t)})}{q(y^{(t)} | x^{(t-1)})} / \frac{\tilde{p}(x^{(t-1)})}{q(x^{(t-1)} | y^{(t)})} \right\}$$
 - c Étape d'acceptation-rejet : générer $u^{(t)} \sim \mathcal{U}_{[0;1]}$
$$x^{(t)} = \begin{cases} y & \text{si } u^{(t)} \leq \alpha^{(t)} \\ x^{(t-1)} & \text{sinon} \end{cases}$$

$$\alpha^{(t)} = \min \left\{ 1, \frac{\tilde{p}(y^{(t)})}{\tilde{p}(x^{(t-1)})} \frac{q(x^{(t-1)} | y^{(t)})}{q(y^{(t)} | x^{(t-1)})} \right\}$$

⇒ calculable en ne connaissant \tilde{p} qu'à une constante près !

Metropolis-Hastings : cas particuliers

On obtient parfois un calcul simplifié pour $\alpha^{(t)}$:

- **Metropolis-Hastings indépendant** : $q(y^{(t)}|x^{(t-1)}) = q(y^{(t)})$
- **Metropolis-Hastings à marche aléatoire** :
 $q(y^{(t)}|x^{(t-1)}) = g(y^{(t)} - x^{(t-1)})$
Si g est symétrique ($g(-x) = g(x)$), alors :

$$\frac{\tilde{p}(y^{(t)})}{\tilde{p}(x^{(t-1)})} \frac{q(y^{(t)}|x^{(t-1)})}{q(x^{(t-1)}|y^{(t)})} = \frac{\tilde{p}(y^{(t)})}{\tilde{p}(x^{(t-1)})} \frac{g(y^{(t)} - x^{(t-1)})}{g(x^{(t-1)} - y^{(t)})} = \frac{\tilde{p}(y^{(t)})}{\tilde{p}(x^{(t-1)})}$$

Pour et contre de l'algorithme de Metropolis-Hastings

- 😊 très simple & très général
- 😊 permet d'échantillonner selon des lois uni- ou multi-dimensionnelles
- 😞 choix de la loi de proposition crucial, mais difficile
 - ⇒ impact considérable sur les performances de l'algorithme
- 😞 devient inefficace dans les problèmes de trop grande dimension

NB : un fort taux de rejet implique souvent des temps de calculs très importants

Algorithme du recuit-simulé

Faire varier le calcul de $\alpha^{(t)}$ au cours de l'algorithme :

- 1 $\alpha^{(t)}$ doit d'abord être grande afin d'explorer l'ensemble de l'espace
- 2 puis $\alpha^{(t)}$ doit diminuer au fur et à mesure que l'algorithme converge

Algorithme du recuit-simulé

Faire varier le calcul de $\alpha^{(t)}$ au cours de l'algorithme :

- 1 $\alpha^{(t)}$ doit d'abord être grande afin d'explorer l'ensemble de l'espace
- 2 puis $\alpha^{(t)}$ doit diminuer au fur et à mesure que l'algorithme converge

1 Initialiser $x^{(0)}$

2 Pour $t = 1, \dots, n + N$:

a Proposer $y^{(t)} \sim q(y^{(t)} | x^{(t-1)})$

b Calculer la probabilité d'acceptation

$$\alpha^{(t)} = \min \left\{ 1, \left(\frac{\tilde{p}(y^{(t)})}{\tilde{p}(x^{(t-1)})} \frac{q(x^{(t-1)} | y^{(t)})}{q(y^{(t)} | x^{(t-1)})} \right)^{\frac{1}{T(t)}} \right\}$$

c Étape d'acceptation-rejet : générer une valeur $u^{(t)} \sim \mathcal{U}_{[0;1]}$

$$x^{(t)} := \begin{cases} y^{(t)} & \text{si } u^{(t)} \leq \alpha^{(t)} \\ x^{(t-1)} & \text{sinon} \end{cases}$$

Algorithme du recuit-simulé

Faire varier le calcul de $\alpha^{(t)}$ au cours de l'algorithme :

- 1 $\alpha^{(t)}$ doit d'abord être grande afin d'explorer l'ensemble de l'espace
- 2 puis $\alpha^{(t)}$ doit diminuer au fur et à mesure que l'algorithme converge

1 Initialiser $x^{(0)}$

2 Pour $t = 1, \dots, n + N$:

a Proposer $y^{(t)} \sim q(y^{(t)} | x^{(t-1)})$

b Calculer la probabilité d'acceptation

$$\alpha^{(t)} = \min \left\{ 1, \left(\frac{\tilde{p}(y^{(t)})}{\tilde{p}(x^{(t-1)})} \frac{q(x^{(t-1)} | y^{(t)})}{q(y^{(t)} | x^{(t-1)})} \right)^{\frac{1}{T(t)}} \right\}$$

c Étape d'acceptation-rejet : générer une valeur $u^{(t)} \sim \mathcal{U}_{[0;1]}$

$$x^{(t)} := \begin{cases} y^{(t)} & \text{si } u^{(t)} \leq \alpha^{(t)} \\ x^{(t-1)} & \text{sinon} \end{cases}$$

Ex : $T(t) = T_0 \left(\frac{T_f}{T_0} \right)^{\frac{t}{n}} \Rightarrow$ particulièrement utile en présence d'optimums locaux

Échantillonneur de Gibbs

Dimension $\nearrow \Rightarrow$ très difficile de proposer des valeurs probables

Échantillonneurs de Gibbs : réactualisation coordonnée par coordonnée, en conditionnant sur les dernières valeurs obtenues (pas d'acceptation-rejet)

- 1 Initialiser $x^{(0)} = (x_1^{(0)}, \dots, x_d^{(0)})$
- 2 Pour $t = 1, \dots, n + N$:
 - a Simuler $x_1^{(t)} \sim p(x_1 | x_2^{(t-1)}, \dots, x_d^{(t-1)})$
 - b Simuler $x_2^{(t)} \sim p(x_2 | x_1^{(t)}, x_3^{(t-1)}, \dots, x_d^{(t-1)})$
 - c ...
 - d Simuler $x_i^{(t)} \sim p(x_i | x_1^{(t)}, \dots, x_{i-1}^{(t)}, x_{i+1}^{(t-1)}, \dots, x_d^{(t-1)})$
 - e ...
 - f Simuler $x_d^{(t)} \sim p(x_d | x_1^{(t)}, \dots, x_{d-1}^{(t)})$

NB : si la loi conditionnelle est inconnue pour certaines coordonnées, on peut introduire une étape d'acceptation-rejet pour cette coordonnée uniquement (*Metropolis within gibbs*)

Méthodes numériques alternatives

- **Bayésien variationnel**
- **Calcul Bayésien Approché (ABC)**


MCMC en pratique

Logiciels MCMC

- **BUGS** : *Bayesian inference Using Gibbs Sampling*

1989 MRC BSU Université de Cambridge (UK)

⇒ logiciel flexible pour l'analyse bayésienne de modèles statistiques complexes à l'aide d'algorithmes MCMC



- WinBUGS : ⚠ clic-bouton + *Windows only* + développement arrêté
<https://www.mrc-bsu.cam.ac.uk/software/bugs/the-bugs-project-winbugs/>
- OpenBUGS : ⚠ clic-bouton + *Windows only* + *Linux partiel*
<https://www.mrc-bsu.cam.ac.uk/software/bugs/openbugs/>
- JAGS : 😊 ligne de commande interfacé avec 
<http://mcmc-jags.sourceforge.net/>

- **STAN**

<http://mc-stan.org/>

rjags

Le logiciel JAGS est moderne et performant :

- s'appuie sur le langage BUGS pour spécifier un modèle bayésien
- interfacé avec  grâce au package rjags
- analyse des résultats dans  grâce aux packages
 - coda
 - HDInterval

Convergence de Markov

Dans l'analyse bayésienne, on utilise les algorithmes MCMC pour générer un **échantillon de Monte-Carlo** de la loi *a posteriori*.

⇒ nécessite d'avoir atteint la **convergence** de la **chaîne de Markov** vers sa loi stationnaire (la loi *a posteriori*).

Convergence de Markov

Dans l'analyse bayésienne, on utilise les algorithmes MCMC pour générer un **échantillon de Monte-Carlo** de la loi *a posteriori*.

⇒ nécessite d'avoir atteint la **convergence** de la **chaîne de Markov** vers sa loi stationnaire (la loi *a posteriori*).

⚠ *Aucune garantie sur la convergence en temps fini*

⇒ **étudier la convergence effective à chaque analyse**

Convergence de Markov

Dans l'analyse bayésienne, on utilise les algorithmes MCMC pour générer un **échantillon de Monte-Carlo** de la loi *a posteriori*.

⇒ nécessite d'avoir atteint la **convergence** de la **chaîne de Markov** vers sa loi stationnaire (la loi *a posteriori*).

⚠ *Aucune garantie sur la convergence en temps fini*

⇒ **étudier la convergence effective à chaque analyse**

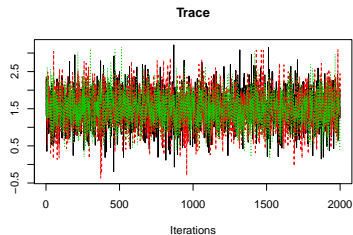
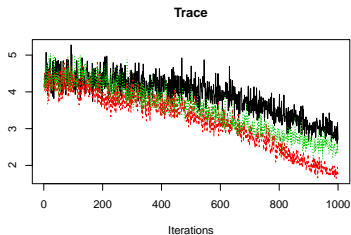
- 💡 Initialisation de **plusieurs chaînes de Markov** à partir de valeurs différentes
- ⇒ Si la **convergence est atteinte**, alors ces **chaînes doivent se confondre**

Diagnostiques graphiques

- Trace
- Densité *a posteriori*
- Quantiles courants (*running quantiles*)
- Auto-corrélation
- Diagramme de Gelman-Rubin

Trace

```
coda::traceplot()
```

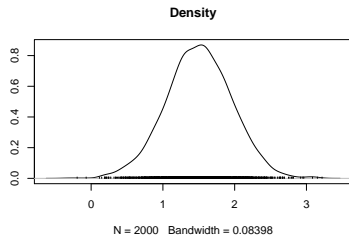
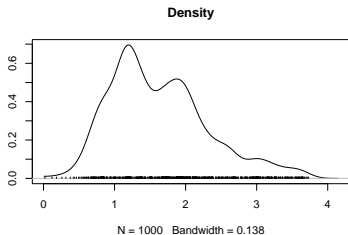


😊 Les chaînes doivent se superposer et se confondre

😞 ↗ n.iter et/ou ↗ phase de chauffe (*burn-in*)

Densité

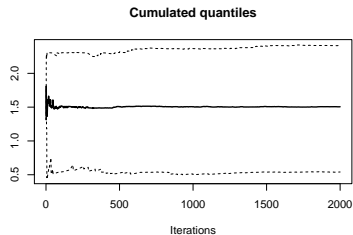
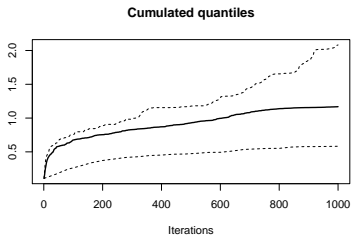
```
coda::densplot()
```



- 😊 Les densités doivent être bien lisses et uni-modales
- 😞 ↗ n.iter et/ou ↗ phase de chauffe (*burn-in*)

Quantiles courants

```
coda::cumuplot()
```



- 😊 Les quantiles cumulés doivent être stables au cours des itérations
- 😞 ↗ n.iter et/ou ↗ phase de chauffe (*burn-in*)

Statistique de Gelman-Rubin

- variation entre les différentes chaînes
- variation à l'intérieur d'une même chaîne

Si l'algorithme a bien convergé, la variation inter-chaîne doit être proche de zéro

$\theta_{[c]} = (\theta_{[c]}^{(1)}, \dots, \theta_{[c]}^{(N)})$ le N -échantillon de la chaîne $c = 1, \dots, C$

La **statistique de Gelman-Rubin** : $R = \frac{\frac{N-1}{N} W \frac{1}{N} B}{W}$

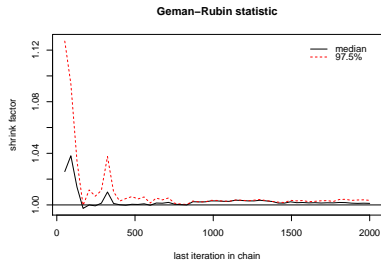
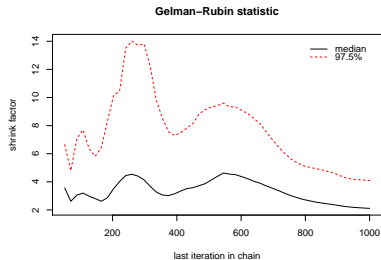
- variance inter-chaînes : $B = \frac{N}{C-1} \sum_{c=1}^C (\bar{\theta}_{[c]} - \bar{\theta})^2$
- moyenne par chaîne : $\bar{\theta}_{[c]} = \frac{1}{N} \sum_{t=1}^N \theta_{[c]}^{(t)}$
- moyenne globale : $\bar{\theta} = \frac{1}{C} \sum_{c=1}^C \bar{\theta}_{[c]}$
- variance intra-chaîne : $s_{[c]}^2 = \frac{1}{N-1} \sum_{t=1}^N (\theta_{[c]}^{(t)} - \bar{\theta}_{[c]})^2$

$$N \rightarrow +\infty \ \& \ B \rightarrow 0 \Rightarrow R \rightarrow 1$$

D'autres statistiques existent. . .

Diagramme de Gelman-Rubin

```
coda::gelman.plot()
```



😄 La médiane de la statistique de Gelman-Rubin doit se maintenir sous le seuil de 1,01 (voir 1,05)

😞 ↗ n.iter et/ou ↗ phase de chauffe (*burn-in*)

Taille d'échantillon effective

Propriété de Markov \Rightarrow **auto-corrélation** entre les valeurs générées à la suite les unes des autres (échantillonnage dépendant) :

- diminue la quantité d'information disponible
- ralentit la convergence de la loi des grands nombres

Taille d'échantillon effective (*effective sample size*) quantifie cela :

$$ESS = \frac{N}{1 + 2 \sum_{k=1}^{+\infty} \rho(k)}$$

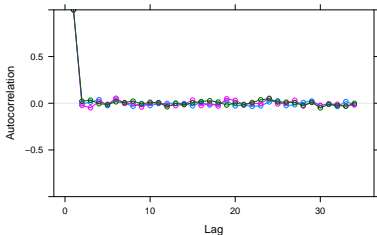
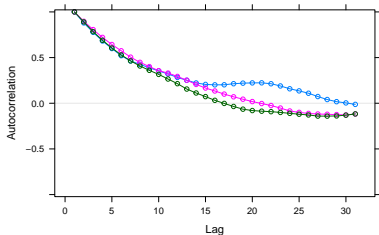
où $\rho(k)$ désigne l'auto-corrélation avec *lag* de rang k .

Espacer les échantillonnages conservés (e.g. toutes les 2, 5, ou 10 itérations)

\Rightarrow diminue la dépendance au sein de l'échantillon de Monte-Carlo généré

Auto-corrélation

```
coda::acfplot()
```



😓 les auto-corrélations doivent décroître rapidement pour osciller autour de zéro

😞 ↗ thin et/ou ↗ n.iter et/ou ↗ phase de chauffe (*burn-in*)

Erreur de Monte-Carlo

Pour un paramètre donnée, quantifie l'erreur introduite par la méthode de Monte-Carlo


- Cette erreur doit être la même d'une chaîne à l'autre
- Plus le nombre d'itérations N est grand, plus cette erreur de Monte-Carlo sera faible

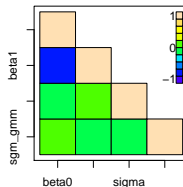
⚠ cette **erreur de Monte-Carlo** doit être faible au regard de la variance estimée de la loi *a posteriori*.

Estimation

Grâce à un algorithme MCMC, on est capable d'obtenir un **échantillon de Monte-Carlo de la loi *a posteriori*** pour un **modèle bayésien**

On peut donc utiliser la **méthode de Monte-Carlo** pour obtenir des **estimations *a posteriori*** :

- Estimation ponctuelle (moyenne *a posteriori*, médiane *a posteriori*, ...)
- Intervalle de crédibilité (le plus étroit : *Highest Density Interval* – *HDI* via le package  `HDInterval`)
- Correlations croisées entre les paramètres
- ...



Deviance Information Criterion (*DIC*)

La déviance s'écrit comme : $D(\theta) = -2\log(p(\theta|\mathbf{y})) + C$ où C est une constante.

Le **Deviance Information Criterion** est alors :

$$DIC = \overline{D(\theta)} + p_D$$

où $p_D = \left(D(\bar{\theta}) - \overline{D(\theta)} \right)$ représente une pénalité pour le nombre effectif de paramètres

⇒ Le *DIC* permet de comparer différents modèles sur les mêmes données
plus le DIC est bas, meilleur est le modèle !

[M Plummer, *Penalized loss functions for Bayesian model comparison*, *Biostatistics*, 2008]

Questions ?

