Simulation et estimation de quantiles extrêmes et de probabilités d'événements rares

d'après l'article d'Arnaud GUYADER, Nicolas HENGARTNER et Eric MATZNER-LØBER

WOLINSKI Pierre,
sous la direction de Josselin GARNIER

17 septembre 2012

Introduction

Il est parfois nécessaire, notamment dans des domaines proches de l'informatique, de savoir estimer des probabilités d'événements rares. Par exemple, l'industrie du disque utilise des tatouages numériques pour empêcher la copie de leurs disques munis de copyright. Si un disque possède ce type de tatouage, la copie du disque est refusée.

Le principe de fonctionnement du tatouage est le suivant : lorsqu'une copie du disque est demandée, une fonction est appliquée sur les données contenues dans le disque, et renvoie un réel. Si ce réel dépasse une certaine valeur fixée q, le disque est considéré comme tatoué, et la copie est refusée. Le risque de refuser la copie d'un disque non tatoué est alors non nul. La probabilité p qu'un tel événement survienne doit être la plus faible possible : seule la copie de disques tatoués doit être refusée. Du point de vue de l'industrie, p est fixée à une valeur très faible, et la valeur recherchée est q (réglage du système de tatouage numérique). Du point de vue de l'utilisateur, q est connu, et la valeur recherchée est p (vérification de la fiabilité du système de tatouage numérique). Cette probabilité est usuellement de l'ordre de 10^{-9} - 10^{-12}.

L'objectif principal est d'une part estimer cette probabilité au mieux, d'autre part l'estimer à l'aide d'un algorithme d'une complexité raisonnable.

Table des matières

1 Définitions

Formellement, on considère :

- un vecteur aléatoire $X \in \mathbb{R}^d$, représentant les données contenues dans le disque ;
- une fonction "boîte noire" $\Phi : \mathbb{R}^d \to \mathbb{R}$;
- un quantile q, tel que : X est considéré comme tatoué $\Leftrightarrow \Phi(X) > q$;
- une loi de probabilité μ sur $\mathbb{R}^d$;
- une probabilité p telle que, si $X \overset{\mathcal{L}}{\sim} \mu$, alors $\mathbb{P}(\Phi(X) > q) = p$.

On cherche à estimer p connaissant q, et q connaissant p.

2 Défauts de la méthode de Monte-Carlo classique

La méthode de Monte-Carlo classique consiste à tirer N variables aléatoires indépendantes $(X_i)_{i\in[\![1,N]\!]}$ selon la loi μ. On estime alors p par $\hat{p}_{mc} = \#\{i|\Phi(X_i) > q\}/N$.

La variance relative de $\hat{p}_{mc}$ est : $\mathrm{Var}(\hat{p}_{mc})/p^2 \sim_{p\to 0} 1/(Np)$.

Par conséquent, pour obtenir une précision appréciable à l'aide de cette méthode, il est nécessaire de choisir $N \approx 1/p \approx 10^{12}$, ce qui est considérable. Il faut donc développer une autre méthode.

3 L'algorithme

3.1 Principe général

On considère N variables aléatoires $(X_i^1)_{i\in[\![1,N]\!]}$ indépendantes identiquement distribuées selon la loi μ.

Si, pour $m \in \mathbb{N}^*$ donné, on connait les $(X_i^m)_{i\in[\![1,N]\!]}$, alors on peut poser :

- $L_m = \min_i(\Phi(X_i^m))$;
- $X_i^{m+1} = \begin{cases} X^\star \sim \mathcal{L}(X|\Phi(X) > L_m) & \text{si } \Phi(X_i^m) = L_m \\ X_i^m & \text{si } \Phi(X_i^m) > L_m \end{cases}$.

On définit ainsi les $(X_i^m)_{i\in[\![1,N]\!],m\in\mathbb{N}^*}$.

Idéalement, $X^\star$ est indépendant des $(X_i^m)_{i\in[\![1,N]\!]}$.

L'algorithme consiste donc à remplacer à chaque itération les X_i^m pour lesquels $\Phi(X_i^m)$ est le plus petit par des X_i^{m+1} pour lesquels $\Phi(X_i^{m+1})$ est strictement plus grand, et à conserver les autres X_i^m. En arrêtant les itérations à un instant bien choisi, on peut estimer p ou q.

On notera que l'efficacité de l'algorithme dépend directement de la méthode choisie pour générer $X^\star$. On proposera une méthode dans le paragraphe 4 permettant de générer une variable aléatoire selon la loi $\mathcal{L}(X|\Phi(X) > L_m)$, *presque* indépendamment des $(X_i^m)_{i\in[\![1,N]\!]}$.

On considère dans le reste de cette partie l'algorithme *idéal*, c'est-à-dire pour lequel $X^\star$ est générée selon la loi $\mathcal{L}(X|\Phi(X) > L_m)$ et indépendamment des $(X_i^m)_{i\in[\![1,N]\!]}$.

3.2 Estimation de p

On suppose que q est connu. Soit $M = \max\{m|L_m \leq q\}$.

3.2.1 Calcul d'un estimateur $\hat{p}$ de p

On cherche à obtenir la loi de M en fonction de p pour parvenir à estimer p en fonction de M. Pour faciliter le calcul de la loi de M, on introduit les fonctions suivantes :

- la fonction de répartition de $\Phi(X)$, *supposée continue* : $F : y \mapsto \mathbb{P}(\Phi(X) \leq y)$;
- $S : y \mapsto 1 - F(y) = \mathbb{P}(\Phi(X) > y)$;
- $\Lambda : y \mapsto -\ln(S(y))$.

Théorème 1. *Soient $(E_i)_{i\in\mathbb{N}^*}$ des variables aléatoires exponentielles de paramètre 1 indépendantes. On a :*

$$\forall m \in \mathbb{N}^*, \Lambda(L_m) \overset{\mathcal{L}}{\sim} \frac{1}{N}\sum_{i=1}^{m} E_i$$

Démonstration

- *On commence par montrer que :*

$$\big(\Lambda(\Phi(X_i^1))\big)_{i\in[\![1,N]\!]} \overset{\mathcal{L}}{\sim} (E_i)_{i\in[\![1,N]\!]}$$

On a :

$$\begin{aligned} &X \sim \mu, \forall a \in \mathbb{R}^+, \\ &\mathbb{P}(\Lambda(\Phi(X)) \leq a) = \mathbb{P}(1 - F(\Phi(X)) \geq e^{-a}) \\ &\qquad\qquad = \mathbb{E}(\mathbb{1}_{1-F(\Phi(X))\geq e^{-a}}) \end{aligned}$$

Comme F est continue et croissante, alors il existe b tel que : $1-F(y) \geq e^{-a} \Leftrightarrow y \leq b$, et $F(b) = 1-e^{-a}$ Par conséquent :

$$\begin{aligned} \mathbb{P}(\Lambda(\Phi(X)) \leq a) &= \mathbb{E}(\mathbb{1}_{1-F(\Phi(X))\geq e^{-a}}\mathbb{1}_{\Phi(X)\geq b} + \mathbb{1}_{1-F(\Phi(X))\geq e^{-a}}\mathbb{1}_{\Phi(X)<b}) \\ &= 0 + \mathbb{E}(1 \cdot \mathbb{1}_{\Phi(X)<b}) \\ &= \mathbb{P}(\Phi(X) < b) \\ &= F(b) \\ &= 1 - e^{-a} \end{aligned}$$

Comme $\Lambda(\Phi) \geq 0$, on a bien :

$$\big(\Lambda(\Phi(X_i^1))\big)_{i\in[\![1,N]\!]} \overset{\mathcal{L}}{\sim} (E_i)_{i\in[\![1,N]\!]}$$

- *On montre par récurrence sur m que les variables aléatoires $\Lambda(L_m)$ peuvent être considérées comme des temps d'arrivée d'un processus de Poisson.*
  - *Pour $m = 1$:*
    Comme Λ est croissante, on a :

$$\Lambda(L_1) = \Lambda\big(\min_{i\in[\![1,N]\!]}(\Phi(X_i^1))\big) = \min_{i\in[\![1,N]\!]}\big(\Lambda(\Phi(X_i^1))\big)$$

Par conséquent :

$$\Lambda(L_1) \overset{\mathcal{L}}{\sim} \frac{E_1}{N}$$

$*$ *On suppose que, pour* $m \in \mathbb{N}^*$ *:*

$$\Lambda(L_m) \overset{\mathcal{L}}{\sim} \frac{1}{N}\sum_{j=1}^{m} E_j$$

On fixe L_m*. Soit* $y > L_m$*. On pose :*

$$\Lambda_{m+1}(y) = \Lambda(y) - \Lambda(L_m)$$

Par définition des X_j^{m+1}*, on a :*

$$\left(\Lambda_{m+1}(\Phi(X_j^{m+1}))\right)_{j\in[\![1,N]\!]} \overset{\mathcal{L}}{\sim} (E_j)_{j\in[\![1,N]\!]}$$

Par la propriété de perte de mémoire des variables aléatoires exponentielles, cette famille est indépendante de $(L_i)_{i\in[\![1,m]\!]}$*. On a donc :*

$$\Lambda_{m+1}(L_{m+1}) = \Lambda(L_{m+1}) - \Lambda(L_m) \overset{\mathcal{L}}{\sim} \frac{E_{m+1}}{N}$$

Par conséquent :

$$\Lambda(L_{m+1}) \overset{\mathcal{L}}{\sim} \Lambda(L_m) + \frac{E_{m+1}}{N}$$

Or $\Lambda_{m+1}(L_{m+1})$ *est indépendante de* $\Lambda_m(L_m)$*, donc, par l'hypothèse de récurrence :*

$$\Lambda(L_{m+1}) \overset{\mathcal{L}}{\sim} \frac{1}{N}\sum_{j=1}^{m+1} E_j$$

où $(E_j)_{j\in[\![1,m+1]\!]}$ *est une famille de variables aléatoires indépendantes.*
Cela achève la démonstration.

Corollaire 1. *La variable aléatoire* M *suit une loi de Poisson* $\mathcal{P}$ *de paramètre* $-N\ln(p)$*.*

Démonstration
On a :

$$\begin{aligned} M &= \max\{m | L_m \leq q\} \\ &= \max\{m | S(L_m) \geq p\} \\ &= \max\{m | \Lambda(L_m) \leq -\ln(p)\} \end{aligned}$$

Donc, par le théorème 1, $M \overset{\mathcal{L}}{\sim} \mathcal{P}(-N\ln(p))$*.*

Comme on suppose que $p \to 0$ et que $N \geq 100$, on peut approximer $\mathcal{P}(-N\ln(p))$ par la gaussienne $\mathcal{N}(-N\ln(p), -N\ln(p))$.

Un estimateur convenable de p serait donc :

$$\hat{p} = \left(1 - \frac{1}{N}\right)^M$$

Proposition 1. *L'estimateur* $\hat{p}$ *de* p *est à valeurs dans :*

$$\mathcal{S} = \left\{ \left(1 - \frac{1}{N}\right)^m \middle| m \in \mathbb{N} \right\}$$

avec :

$$\forall m \in \mathbb{N}, \mathbb{P}\left(\hat{p} = \left(1 - \frac{1}{N}\right)^m\right) = \frac{p^N(-N\ln(p))^m}{m!}$$

Donc $\hat{p}$ est un estimateur non biaisé de p de variance :

$$\mathrm{Var}(\hat{p}) = p^2\left(p^{-\frac{1}{N}} - 1\right)$$

Démonstration

- *On démontre que la loi de M est bien celle donnée par l'énoncé.*
  On a, par le corollaire 1 :

$$\begin{aligned}\forall m \in \mathbb{N}, \mathbb{P}\left(\hat{p} = \left(1 - \frac{1}{N}\right)^m\right) &= \mathbb{P}\left(\left(1 - \frac{1}{N}\right)^M = \left(1 - \frac{1}{N}\right)^m\right) \\ &= \mathbb{P}(M = m) \\ &= \frac{p^N(-N\ln(p))^m}{m!}\end{aligned}$$

- *On montre que $\hat{p}$ est un estimateur non biaisé de p.*
  On a :

$$\begin{aligned}\mathbb{E}(\hat{p} - p) &= \mathbb{E}\left[\left(1 - \frac{1}{N}\right)^M\right] - p \\ &= \sum_{m=0}^{+\infty}\left[\left(1 - \frac{1}{N}\right)^m \frac{p^N(-N\ln(p))^m}{m!}\right] - p \\ &= p^N \sum_{m=0}^{+\infty}\left[\left(-N\ln(p) + \frac{N\ln(p)}{N}\right)^m \frac{1}{m!}\right] - p \\ &= p^N \sum_{m=0}^{+\infty}\left[\left(-N\ln(p) + \ln(p)\right)^m \frac{1}{m!}\right] - p \\ &= p^N e^{(1-N)\ln(p)} - p \\ &= 0\end{aligned}$$

- *On calcule la variance de $\hat{p}$:*

$$\begin{aligned}\mathrm{Var}(\hat{p}) &= \mathbb{E}\left[\left(\left(1 - \frac{1}{N}\right)^M - p\right)^2\right] \\ &= \mathbb{E}\left[\left(1 - \frac{1}{N}\right)^{2M}\right] - p^2 \\ &= \sum_{m=0}^{+\infty}\left(1 - \frac{1}{N}\right)^{2m} \frac{p^N(-N\ln(p))^m}{m!} - p^2 \\ &= p^N \sum_{m=0}^{+\infty}\left[\left(1 - \frac{1}{N}\right)^2(-N\ln(p))\right]^m \frac{1}{m!} - p^2 \\ &= p^N e^{(-N+2-\frac{1}{N})\ln(p)} - p^2 \\ &= p^2\left(p^{-\frac{1}{N}} - 1\right)\end{aligned}$$

  On remarque que, lorsque $p \to 0$:

$$\frac{\mathrm{Var}(\hat{p})}{p^2} \sim \frac{-\ln(p)}{N} \ll \frac{1}{Np} \sim \frac{\mathrm{Var}(\hat{p}_{mc})}{p^2}$$

Cela montre que la précision obtenue à l'aide de la version *idéalisée* de l'algorithme est, à N égal, meilleure que celle obtenue avec la méthode de Monte-Carlo classique. Il ne faut en outre pas perdre de vue que la génération pratique de $X^\star$ peut être coûteuse en complexité. On ne peut donc pas encore conclure sur la pertinence d'un tel résultat.

3.2.2 Contrôle de $\hat{p}$

On cherche à obtenir un intervalle de confiance pour p.

Proposition 2. *Soit $Z_{1-\alpha/2}$ le quantile d'ordre $1-\alpha/2$ de $\mathcal{N}(0,1)$, défini pour $\alpha \in [0,1]$.*
On pose :

$$\hat{p}_{\pm} = \hat{p}\exp\left(\pm \frac{Z_{1-\alpha/2}}{\sqrt{N}}\sqrt{-\ln(\hat{p}) + \frac{Z_{1-\alpha/2}^2}{4N}} - \frac{Z_{1-\alpha/2}^2}{2N}\right)$$

Donc, avec probabilité $1-\alpha$, on a : $p \in I_{1-\alpha}(p) = [\hat{p}_-, \hat{p}_+]$.

Démonstration
On pose : $l = \ln(p)$, $\hat{l} = \ln(\hat{p})$. On a donc : $l, \hat{l} < 0$
Comme $M \overset{\mathcal{L}}{\sim} \mathcal{P}(-N\ln(p)) \approx \mathcal{N}(-N\ln(p), -N\ln(p))$,
alors on peut approximer la loi de $\hat{l}$ par une loi normale :

$$\hat{l} = M\ln\left(1-\frac{1}{N}\right) \approx -\frac{M}{N} \overset{\mathcal{L}}{\sim} \mathcal{N}\left(\ln(p), -\frac{\ln(p)}{N}\right) = \mathcal{N}\left(l, -\frac{l}{N}\right)$$

Par définition de $Z_{1-\alpha/2}$, on a, avec probabilité $1-\alpha$:

$$|l - \hat{l}| \leq Z_{1-\alpha/2}\sqrt{-\frac{l}{N}}$$

Ce qui est équivalent à :

$$l^2 - 2\left(\hat{l} - \frac{Z_{1-\alpha/2}^2}{2N}\right)l + \hat{l}^2 \leq 0$$

Ce trinôme en l a pour discriminant réduit :

$$\Delta' = \left(\hat{l} - \frac{Z_{1-\alpha/2}^2}{2N}\right)^2 - \hat{l}^2 = \frac{Z_{1-\alpha/2}^2}{N}\left(-\hat{l} + \frac{Z_{1-\alpha/2}^2}{4N}\right)$$

Comme $\hat{l} < 0$, alors $\Delta' > 0$.
Par conséquent :

$$\hat{l} - \frac{Z_{1-\alpha/2}}{2N} - \sqrt{\Delta'} \leq L \leq \hat{l} - \frac{Z_{1-\alpha/2}}{2N} + \sqrt{\Delta'}$$

En passant à l'exponentielle, on arrive au résultat attendu.

On peut de plus simplifier l'expression des bornes de l'intervalle $I_{1-\alpha}(p)$ si N est suffisamment grand. Dans ce cas, $\frac{1}{N} \ll \frac{1}{\sqrt{N}}$, donc :

$$\hat{p}\exp\left(-\frac{Z_{1-\alpha/2}}{\sqrt{N}}\sqrt{-\ln(\hat{p})}\right) \leq p \leq \hat{p}\exp\left(\frac{Z_{1-\alpha/2}}{\sqrt{N}}\sqrt{-\ln(\hat{p})}\right)$$

On peut appliquer ce résultat avec $\alpha = 0.05$, ce qui correspond à $Z_{1-\alpha/2} \approx 2$. Dans ce cas, on a, avec probabilité 0.95 :

$$\hat{p}\exp\left(-2\sqrt{-\frac{\ln(\hat{p})}{N}}\right) \leq p \leq \hat{p}\exp\left(2\sqrt{-\frac{\ln(\hat{p})}{N}}\right)$$

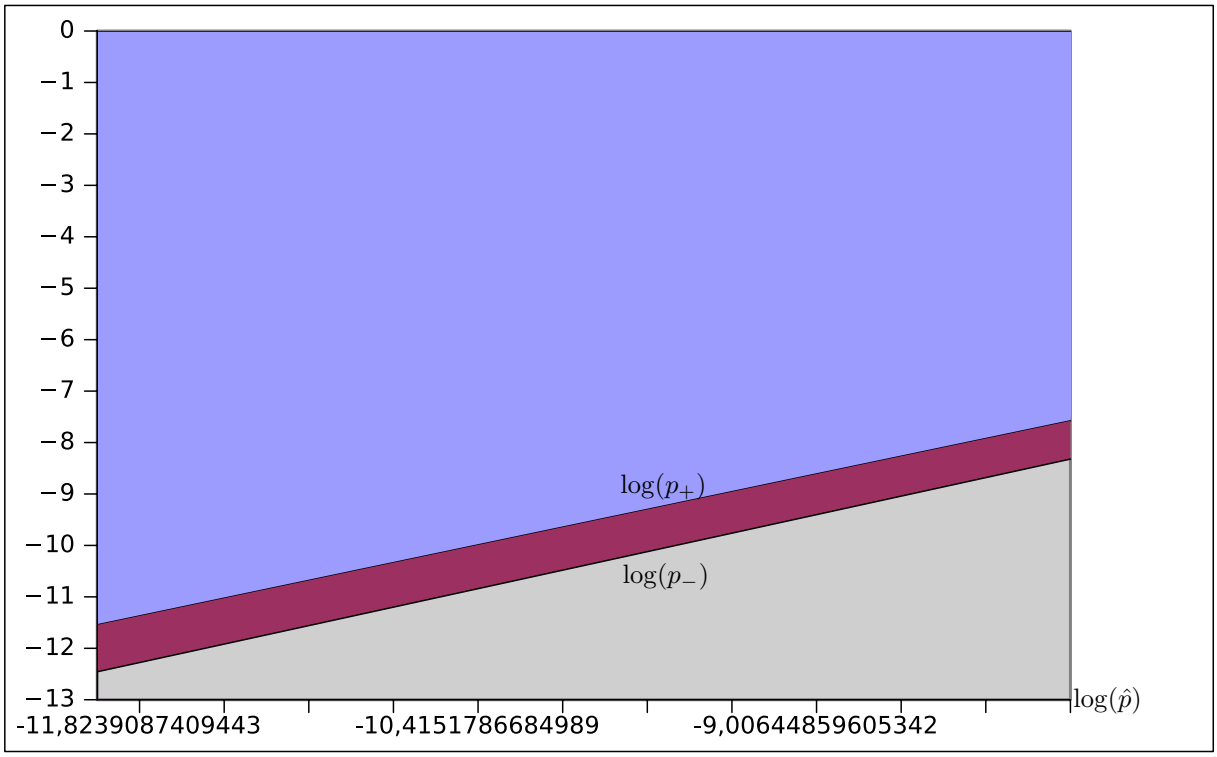

FIGURE 1 – Intervalle de confiance à 95% pour p, avec $N = 100$ et $\hat{p} \in [10^{-11}, 10^{-8}]$: avec probabilité 0.95, p est situé dans la zone centrale.

On remarque sur le graphe que, même pour N faible, l'intervalle de confiance est assez étroit : on parvient à estimer un ordre de grandeur de p. Cela illustre l'efficacité de l'algorithme.

3.3 Estimation de q

On suppose ici que p est connue.

3.3.1 Calcul d'un estimateur $\hat{q}$ de q

On a montré que :

$$\hat{p} = \left(1 - \frac{1}{N}\right)^M$$

était un estimateur non biaisé de p.

Comme $M = \max\{m | L_m \leq q\}$, un estimateur naturel de q serait :

$$\hat{q} = L_m, \text{ avec : } m = \left\lceil \frac{\ln(p)}{\ln(1 - N^{-1})} \right\rceil$$

Proposition 3. *Si F est dérivable en q de dérivée continue en q, avec $F'(q) = f(q) \neq 0$, alors :*

$$\sqrt{N}(\hat{q} - q) \xrightarrow[N \to +\infty]{\mathcal{L}} \mathcal{N}\left(0, -\frac{p^2 \ln(p)}{f(q)^2}\right)$$

Démonstration

On sait que :

$$\Lambda(L_m) \overset{\mathcal{L}}{\approx} \frac{1}{N} \sum_{j=1}^{m} E_j \overset{\mathcal{L}}{\approx} \frac{G_m}{N}, \text{ où } G_m \overset{\mathcal{L}}{\approx} \Gamma(m, 1)$$

Donc :

$$S(L_m) = \exp\left(-\frac{G_m}{N}\right)$$

Par conséquent :

$$\hat{q} = S^{-1}\left(\exp\left(-\frac{G_m}{N}\right)\right)$$

Comme $\mathbb{E}(E_i) = 1$, *et* $\mathrm{Var}(E_i) = 1$, *alors le théorème central limite permet d'écrire :*

$$\frac{G_m - m}{\sqrt{m}} \xrightarrow[m\to+\infty]{\mathcal{L}} \mathcal{N}(0,1)$$

Par ailleurs :

$$\frac{G_m - m}{\sqrt{m}} = \sqrt{\frac{N}{m}}\left(\sqrt{N}\left(\frac{G_m}{N} - (-\ln(p))\right)\right) - \frac{N}{\sqrt{m}}\left(\ln(p) + \frac{m}{N}\right)$$

Or :

$$\frac{\ln(p)}{\ln(1-N^{-1})} \leq m = \left\lceil \frac{\ln(p)}{\ln(1-N^{-1})} \right\rceil < 1 + \frac{\ln(p)}{\ln(1-N^{-1})}$$

Donc :

$$\frac{N}{\sqrt{m}}\left(\ln(p) + \frac{m}{N}\right) \xrightarrow[N\to+\infty]{} 0, \text{ et : } \sqrt{\frac{N}{m}} \xrightarrow[N\to+\infty]{} \sqrt{-\frac{1}{\ln(p)}}$$

Par conséquent :

$$\sqrt{-\frac{1}{\ln(p)}}\left(\sqrt{N}\left(\frac{G_m}{N} - (-\ln(p))\right)\right) \xrightarrow[N\to+\infty]{\mathcal{L}} \mathcal{N}(0,1)$$

C'est-à-dire :

$$\sqrt{N}\left(\frac{G_m}{N} - (-\ln(p))\right) \xrightarrow[N\to+\infty]{\mathcal{L}} \mathcal{N}(0, -\ln(p))$$

On peut appliquer la méthode delta (appendice A) à l'expression précédente en posant $X_N = G_m/N$, $\theta = -\ln(p)$, $\sigma^2 = -\ln(p)$, *et* $g : x \mapsto S^{-1}(\exp(-x))$. *Ce qui donne :*

$$\sqrt{N}(\hat{q} - q) \xrightarrow[N\to+\infty]{\mathcal{L}} \mathcal{N}(0, -\frac{p^2\ln(p)}{f(q)^2})$$

De la même manière, on peut construire à l'aide de la méthode de Monte-Carlo classique un estimateur $\hat{q}_{mc}$ de q. On trie la famille $(Y_i = \Phi(X_i))_{i\in[\![1,N]\!]}$: $Y_{\sigma(1)} \leq Y_{\sigma(1)} \leq \cdots \leq Y_{\sigma(N)}$. On pose alors : $\hat{q}_{mc} = Y_{\lfloor(1-p)N\rfloor}$. En utilisant là aussi la méthode delta, on a :

$$\sqrt{N}(\hat{q}_{mc} - q) \xrightarrow[N\to+\infty]{\mathcal{L}} \mathcal{N}(0, -\frac{p(1-p)}{f(q)^2})$$

Là aussi, la complexité de la méthode de Monte-Carlo classique est, à variance égale, plus élevée que celle de l'algorithme : il faut générer $(-p\ln(p))^{-1}$ plus de variables aléatoires.

3.3.2 Contrôle de $\hat{q}$

On cherche à obtenir un intervalle de confiance pour p.

Proposition 4. *On pose :*

$$\begin{cases} m_- = \lfloor -N\ln(p) - Z_{1-\alpha/2}\sqrt{-N\ln(p)} \rfloor \\ m_+ = \lceil -N\ln(p) + Z_{1-\alpha/2}\sqrt{-N\ln(p)} \rceil \end{cases}$$

Avec probabilité $1-\alpha$, *on a :* $q \in I_{1-\alpha}(q) = [L_{m_-}, L_{m_+}]$.

Démonstration

On a :

$$\textit{Pour } k \in \mathbb{N}^* \textit{ fixé, on a : } \mathbb{P}(L_k \leq q \leq L_{k+1}) = \mathbb{P}(M = k)$$

On recherche l'intervalle $[L_k, L_K]$ le plus petit possible tel que :

$$\sum_{j=k}^{K} \mathbb{P}(M = j) \geq 1 - \alpha$$

On peut résoudre ce problème en remarquant que $M \overset{\mathcal{L}}{\sim} \mathcal{N}(-N\ln(p), -N\ln(p))$. On a donc, avec probabilité $1-\alpha$:

$$-N\ln(p) - Z_{1-\alpha/2}\sqrt{-N\ln(p)} \leq M \leq -N\ln(p) + Z_{1-\alpha/2}\sqrt{-N\ln(p)}$$

Ce qui conclut la preuve.

Cette proposition permet de contrôler $\hat{q}$, mais cela a un coût : il est nécessaire pour calculer m_+ de continuer d'exécuter l'algorithme alors que $\hat{q}$ a déjà été obtenu.

4 Génération pratique de $X^\star$

4.1 Présentation de la méthode

Pour $m \in \mathbb{N}^*$, on pose :

- $A_m = \{x \in \mathbb{R}^d | \Phi(x) > L_m\}$;
- $\mu_m = \frac{\mu|_{A_m}}{\mu(A_m)}$, la mesure de probabilité induite par μ sur A_m.

La méthode consiste à construire un noyau de transition $K_m(x, dx')$ irréductible et μ_m-invariant, et à appliquer celui-ci un grand nombre de fois à X_i^m, choisi aléatoirement dans A_m. Par le théorème de stabilisation, le vecteur $X^\star$ ainsi obtenu appartient à A_m, et est presque indépendant des $(X_i^m)_{i \in [\![1,N]\!]}$.

4.2 Construction de K_m

Soit $\sigma > 0$, on pose :

$$K(x, dx') = \sqrt{\frac{1+\sigma^2}{2\pi\sigma^2}} \exp\left(-\frac{1+\sigma^2}{2\sigma^2}\left(x' - \frac{x}{\sqrt{1+\sigma^2}}\right)^2\right)\lambda(dx')$$

La transition $X \rightsquigarrow X'$ proposée par K est : $X' = (X + \sigma W)/\sqrt{1+\sigma^2}$, où W est un vecteur aléatoire gaussien standard.

Soit $m \in \mathbb{N}^*$, on pose :

$$K_m(x, dx') = \mathbb{1}_{A_m^c}(x)\delta_x(dx') + \mathbb{1}_{A_m}(x)(K(x, dx')\mathbb{1}_{A_m}(x') + K(x, A_m^c)\delta_x(dx'))$$

Partant de x, K propose une transition $x \rightsquigarrow x'$. Si $x' \in A_m$, la transition est acceptée. Sinon, la transition est refusée et x reste inchangé. Par ailleurs, K_m est irréductible et μ_m-invariant.

Par le théorème de stabilisation, on a :

$$\forall x \in \mathbb{R}^d, \lim_{n \to +\infty} \left(\int_{\mathbb{R}^d} |K_m^n(x, dx') - \mu_m(dx')| \right) = 0$$

Donc on obtient le résultat voulu pour n suffisamment grand.

Le seul paramètre que l'on contrôle est σ, et l'efficacité de l'algorithme dépend grandement de σ. Si σ est trop faible, la transition $x \rightsquigarrow x'$ est presque toujours acceptée, mais, le pas $x \rightsquigarrow x'$ étant petit, il faut prendre n très grand pour avoir quasi-indépendance entre $X^\star$ et X_i^m. Si au contraire σ est trop élevée, le pas $x \rightsquigarrow x'$ est grand, donc il y a rapidement quasi-indépendance entre $X^\star$ et X_i, mais la transition $x \rightsquigarrow x'$ est presque toujours refusée. Dans les deux cas, la complexité de l'algorithme est élevée, c'est pourquoi le choix judicieux d'un σ intermédiaire est nécessaire.

En pratique, on effectuera T transitions, avec T fixé.

5 Performance de l'algorithme

5.1 Complexité

On calcule ici la complexité de l'algorithme *non idéal*, pour la comparer, à variance égale, à celle de la méthode de Monte-Carlo classique. De manière générale, on peut estimer la complexité de l'algorithme en distinguant deux opérations dans le code utilisé ici (appendice C) :

- le tri de $(X_i^1)_{i\in[\![1,N]\!]}$, à l'aide de l'algorithme de tri fusion, de complexité en $\mathcal{O}(N\ln(N))$;
- l'insertion d'un élément dans une liste triée par dichotomie, de complexité en $\mathcal{O}(\ln(N))$, opération réalisée $\mathcal{O}(-N\ln(p))$ fois.

Par conséquent, l'algorithme est de complexité en $\mathcal{O}(-N\ln(p)\ln(N))$, ce qui est bien supérieur à la complexité de l'algorithme de Monte-Carlo classique, en $\mathcal{O}(N)$. Mais les deux algorithmes n'ayant pas la même variance, cette remarque n'est pas pertinente.

On peut définir l'efficacité d'un algorithme de Monte-Carlo comme étant inversement proportionnel au produit de la variance obtenue et de la complexité C de l'algorithme. Cette définition permet de comparer la complexité de deux algorithmes à variance égale. Soit e l'efficacité de l'algorithme :

- pour la méthode de Monte-Carlo classique, on a :

$$\mathrm{Var}(\hat{p}_{mc}) \sim \frac{p}{N} \text{ , et : } C_{mc} \sim N$$

  Donc : $e_{mc} \propto p^{-1}$;
- pour l'algorithme, on a :

$$\mathrm{Var}(\hat{p}) \sim -\frac{p^2\ln(p)}{N} \text{ , et : } C \sim -\lambda N\ln(p)\ln(N)$$

  Donc : $e \propto (\lambda p^2\ln(p)^2\ln(N))^{-1}$

Par conséquent :

$$\frac{e}{e_{mc}} \sim \frac{1}{\lambda p\ln(p)^2\ln(N)}$$

Pour $N = 5000$ et $\lambda = 20$, on a :

$$\frac{e}{e_{mc}} \sim \frac{10^{-3}}{p\ln(p)^2}$$

Donc l'algorithme serait effectivement plus efficace que la méthode de Monte-Carlo classique pour des probabilités faibles, c'est-à-dire $p \leq 10^{-4}$.

5.2 Estimation de p

Pour vérifier que l'algorithme présenté précédemment est applicable, il suffit de comparer, pour une fonction Φ donnée, $\hat{p}$ à p. Il faut donc construire une fonction Φ telle que l'on puisse calculer p numériquement.

5.2.1 Calcul de p pour une fonction Φ donnée

On considère :

- $q \in \mathbb{R}^+$;
- $u \in \mathbb{R}^d$;
- $\Phi : x \mapsto \frac{|{}^t xu|}{\|x\|}$;
- $\mu \overset{\mathcal{L}}{\sim} \mathcal{N}(0,1)$.

Lorsque q est fixé, on cherche à estimer : $p = \mathbb{P}(\Phi(X) \geq q)$. En posant $\theta = \mathrm{Arccos}(q)$, on peut interpréter géométriquement cet événement : il survient lorsque X est dans l'hypercône de révolution d'axe dirigé par u et d'angle θ.

Lemme 1. *Soit G la fonction de répartition de la loi de Fisher-Snedecor (appendice B) de degrés de liberté 1 et $d-1$. On a :*

$$p = 1 - G\left((d-1)\frac{q^2}{1-q^2}\right)$$

Démonstration
Soit P_u la projection orthogonale sur u. On a :

$$p = \mathbb{P}(\Phi(X) \geq q) = \mathbb{P}\left(\frac{\|P_u X\|}{\|X\| \geq q}\right) = \mathbb{P}\big(\|P_u X\|^2 \geq q^2 \|X\|^2\big)$$

Or : $X = P_u X + (X - P_u X)$, où $P_u X \perp (X - P_u X)$. Donc, par le théorème de Pythagore :

$$p = \mathbb{P}\big(\|P_u X\|^2 \geq q^2(\|P_u X\|^2 + \|X - P_u X\|^2)\big)$$

Donc :

$$p = \mathbb{P}\left(\frac{\|P_u X\|^2}{\|X - P_u X\|^2/(d-1)} \geq (d-1)\frac{q^2}{1-q^2}\right)$$

Par conséquent :

$$p = \mathbb{P}\left(Z \geq (d-1)\frac{q^2}{1-q^2}\right)$$

où Z est une variable aléatoire réelle suivant la loi de Fisher-Snedecor avec 1 et $d-1$ degrés de liberté.

5.2.2 Comparaison avec $\hat{p}$

On calcule $\hat{p}$ à l'aide de l'algorithme, avec les paramètres suivants :

- dimension de l'espace : $d = 20$;
- quantile : $q = 0.95$;
- nombre de variables aléatoires : $N = 100, 200, 500, 1000$;
- nombre de transitions : $T = 20$;
- paramètre de K : $\sigma = 0.3$;
- nombre d'expériences : $P = 100$.

P correspond au nombre de fois que l'on calcule $\hat{p}$ avec les mêmes paramètres, dans le but de comparer l'efficacité de l'algorithme pour plusieurs valeurs de N.

On a choisi $\sigma = 0.3$, car σ est de l'ordre de grandeur du pas que l'on effectue lors d'une transition, et une longueur caractéristique de la base de l'hypercône est : $\theta = \text{Arccos}(q) \approx 0.3$. Donc σ n'est ni "trop petit", ni "trop grand".

Avec $q = 0.95$, un calcul numérique indique que $p \approx 4.704 \cdot 10^{-11}$.

Les estimations de p avec les paramètres précédents ont été réalisées à l'aide du code situé à l'appendice C. Le diagramme suivant illustre les résultats obtenus.

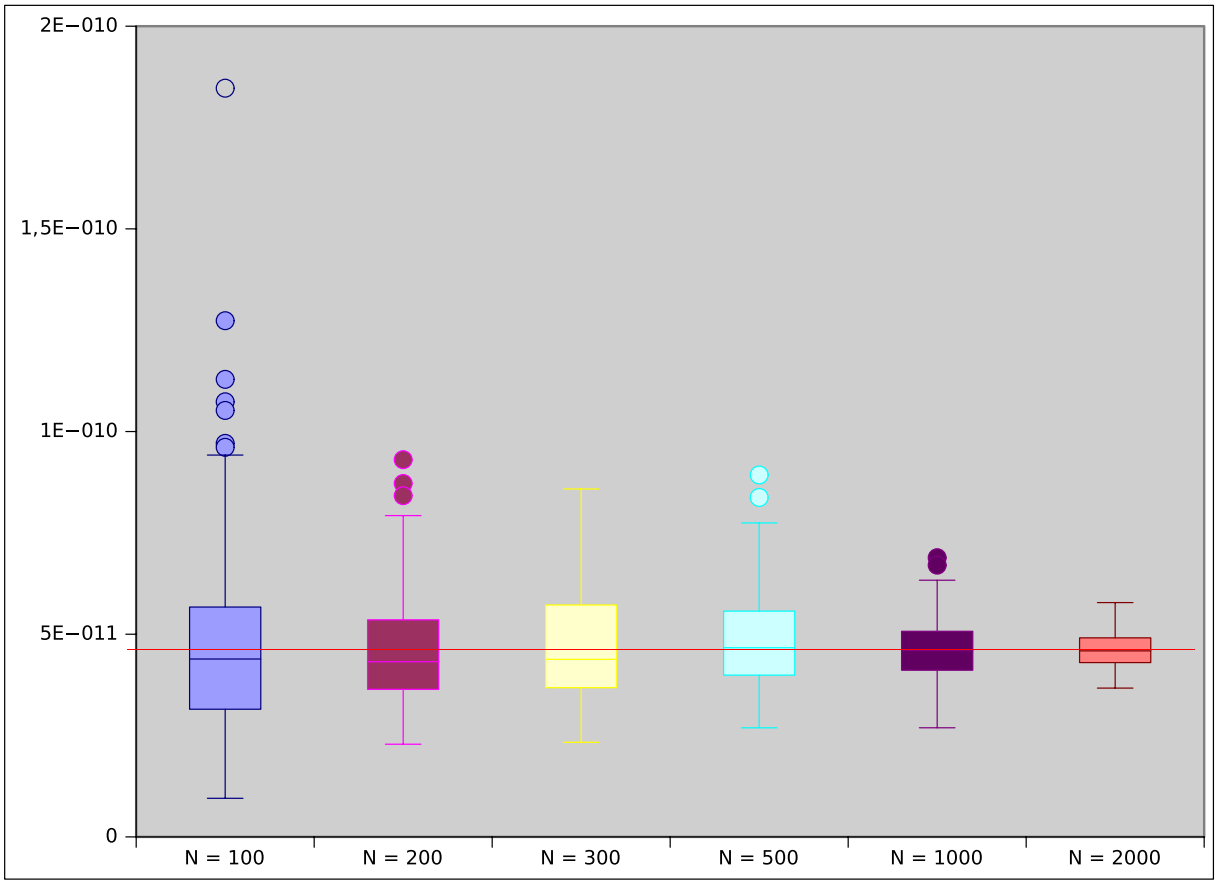

FIGURE 2 – Diagramme à moustaches illustrant la qualité de l'estimation de p en fonction de N

On peut aussi vérifier que p est bien dans l'intervalle de confiance dans 95% des cas.

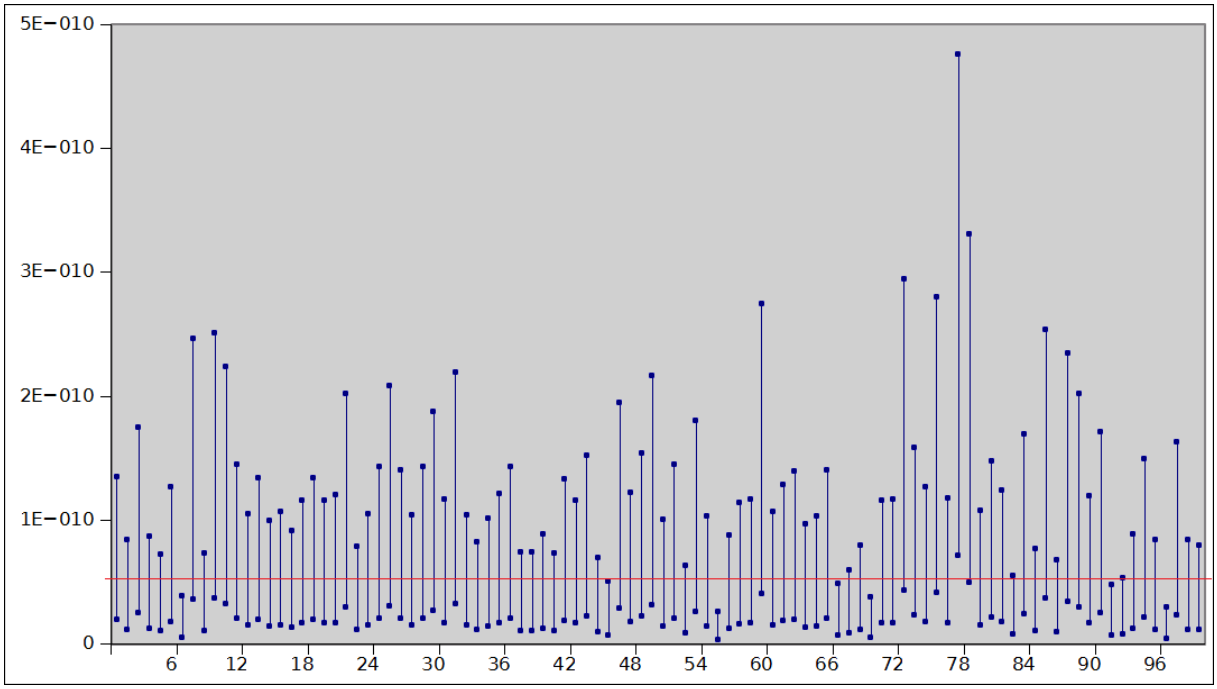

FIGURE 3 – Intervalles de confiance de p pour $N = 100$ sur $P = 100$ estimations.

Sur 100 estimations, 94 ont un intervalle de confiance contenant p, ce qui est cohérent avec les résultats théoriques.

5.3 Estimation de q

On cherche maintenant à estimer q. Les paramètres sont les mêmes que précédemment. On prend $p = 4.704 \cdot 10^{-11}$. On sait donc que $q \approx 0.95$.

Les résultats des estimations de q sont les suivants :

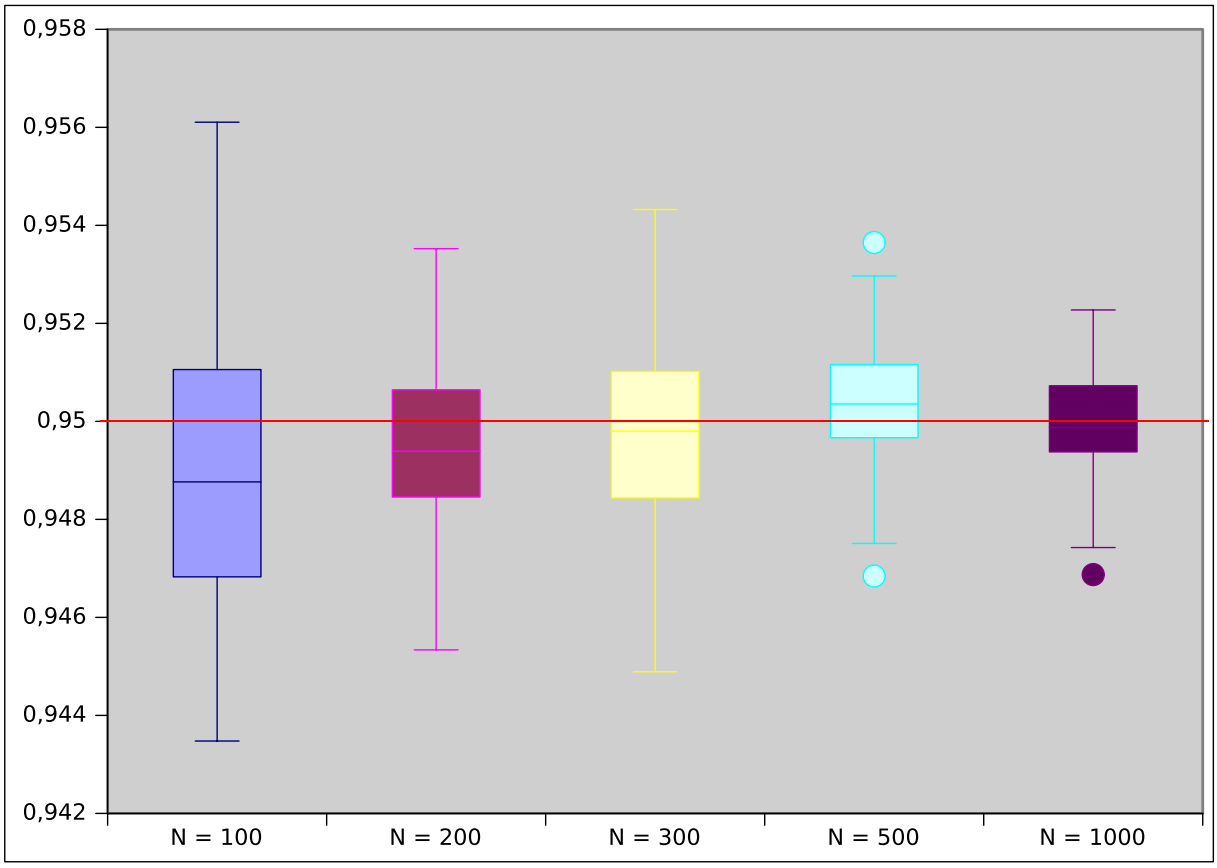

FIGURE 4 – Diagramme à moustaches illustrant la qualité de l'estimation de q en fonction de N

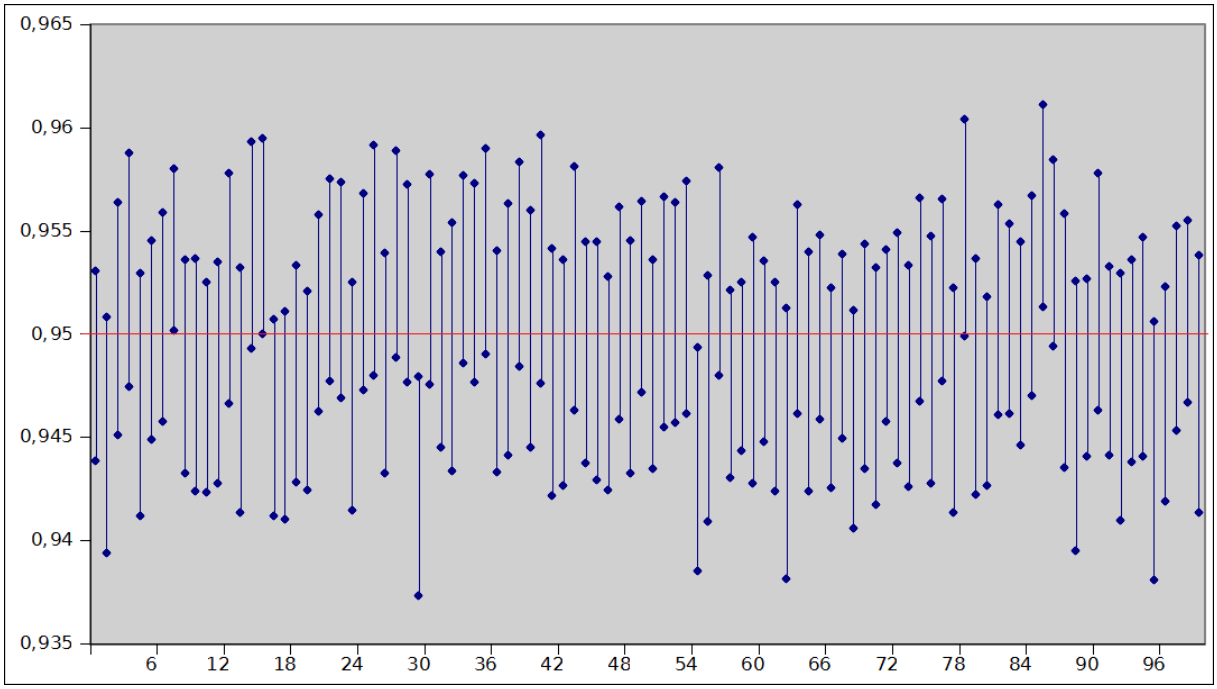

FIGURE 5 – Intervalles de confiance de q pour $N = 100$ sur $P = 100$ estimations.

On remarque ici que, sur 100 estimations, 95 ont un intervalle de confiance contenant q. Cela montre

que les approximations réalisées pour aboutir à ces intervalles sont valides.

Appendices

A La méthode delta

A.1 Énoncé

La méthode delta permet de déduire de la convergence en loi d'une suite de variables aléatoires vers une gaussienne la convergence en loi de l'image de cette suite par une fonction vers une autre gaussienne.

Soit $(X_n)_n$ une suite de variables aléatoires de même variance σ^2. Soit θ une constante. Soit g une fonction dérivable en θ de dérivée continue en θ, avec $g'(\theta) \neq 0$.
On a :

$$\sqrt{n}(X_n - \theta) \xrightarrow[n\to+\infty]{\mathcal{L}} \mathcal{N}(0, \sigma^2) \Rightarrow \sqrt{n}(g(X_n) - g(\theta)) \xrightarrow[n\to+\infty]{\mathcal{L}} \mathcal{N}(0, \sigma^2(g'(\theta))^2)$$

A.2 Démonstration

Le théorème des accroissements finis permet d'écrire :

$$\exists \tilde{\theta} \text{ entre } X_n \text{ et } \theta \big| g'(\tilde{\theta}) = \frac{g(X_n) - g(\theta)}{X_n - \theta}$$

Donc :

$$\sqrt{n}(g(X_n) - g(\theta)) = g'(\tilde{\theta})\sqrt{n}(X_n - \theta)$$

Par ailleurs, si $\tilde{\theta}$ est comprise entre X_n et θ, alors $X_n \xrightarrow[n\to+\infty]{\mathcal{P}} \theta \Rightarrow \tilde{\theta} \xrightarrow[n\to+\infty]{\mathcal{P}} \theta$.
Or g' est continue en θ, donc :

$$g'(\tilde{\theta}) \xrightarrow[n\to+\infty]{\mathcal{P}} g'(\theta)$$

Comme

$$\sqrt{n}(X_n - \theta) \xrightarrow[\to N]{\mathcal{L}} (0, \sigma^2)$$

Alors :

$$\sqrt{n}(g(X_n) - g(\theta)) \xrightarrow[n\to+\infty]{\mathcal{L}} \mathcal{N}(0, \sigma^2(g'(\theta))^2)$$

Ce qui est le résultat attendu.

B La loi de Fisher-Snedecor

On rappelle que la loi χ^2 de degré de liberté k est définie de la manière suivante :

$$X \overset{\mathcal{L}}{\sim} \chi^2(k) \Leftrightarrow X \overset{\mathcal{L}}{\sim} \sum_{i=1}^{k} X_i^2 \text{ , où } \forall i \in [\![1, k]\!], X_i \overset{\mathcal{L}}{\sim} \mathcal{N}(0, 1)$$

Soient $U_1 \overset{\mathcal{L}}{\sim} \chi^2(d_1)$ et $U_2 \overset{\mathcal{L}}{\sim} \chi^2(d_2)$. On note $\mathcal{F}(d_1, d_2)$ la loi de Fisher-Snedecor de degrés de liberté d_1 et d_2. On a :

$$\mathcal{F}(d_1, d_2) \overset{\mathcal{L}}{\sim} \frac{U_1/d_1}{U_2/d_2}$$

C Code de l'algorithme (Octave)

L'algorithme a été codé pour le logiciel de calcul matriciel Octave. Le code source est le suivant :

```
% Fonction renvoyant la liste des normes des colonnes de la matrice x
% (utile pour appliquer Phi à une famille de vecteurs)
function Ret = NormVect(x)
    Ret = transpose(diag(sqrt(transpose(x) * x)));
endfunction

% La fonction Phi
function Ret = Phi(x)
    d = length(x(:,1));
    u = 0 * (1:1:d) .+ 1;
    u = u / norm(u);
    Ret = abs(u * x) ./ NormVect(x);
endfunction

% Algorithme de tri fusion
% (utilisé pour trier pour la première fois la famille)
function [RetX, RetPhiX] = MergeSort(X, PhiX)
    N = length(PhiX);
    k = floor(N / 2);
    if N <= 1
        RetX = X;
        RetPhiX = PhiX;
    else
        RetPhiX = [];
        RetX = [];
        X1 = X(:,1:k);
        X2 = X(:,(k + 1):N);
        PhiX1 = PhiX(:,1:k);
        PhiX2 = PhiX(:,(k + 1):N);
        [X1, PhiX1] = MergeSort(X1, PhiX1);
        [X2, PhiX2] = MergeSort(X2, PhiX2);
        i = 1;
        j = 1;
        while i <= k && j <= (N - k)
            if PhiX1(i) > PhiX2(j)
                RetPhiX = [RetPhiX, PhiX2(j)];
                RetX = [RetX, X2(:,j)];
                j++;
            else
                RetPhiX = [RetPhiX, PhiX1(i)];
                RetX = [RetX, X1(:,i)];
                i++;
            end
        end
        if i <= k
            RetPhiX = [RetPhiX, PhiX1(i:k)];
            RetX = [RetX, X1(:,i:k)];
        end
        if j <= (N - k)
            RetPhiX = [RetPhiX, PhiX2(j:(N - k))];
            RetX = [RetX, X2(:,j:(N - k))];
        end
```

```
    end
endfunction

% Fonction insérant un élément dans une liste déjà triée par dichotomie
% (l'élément à insérer est en fait le premier élément de la famille)
function [RetX, RetPhiX] = Insert(X, PhiX)
    RetX = X;
    RetPhiX = PhiX;
    N = length(PhiX);
    PhiY = PhiX(1);
    iMin = 1;
    iMax = N;
    i = 1;
    while PhiY < PhiX(i) || PhiY > PhiX(i + 1)
        i = ceil((iMin + iMax) / 2);
        if i >= N
            break;
        end
        if PhiX(i) > PhiY
            iMax = i;
        else
            iMin = i;
        end
    end
    if i == N
        RetPhiX = [PhiX(2:N),PhiY];
        RetX = [X(:,2:N),X(:,1)];
    elseif i > 1
        RetPhiX = [PhiX(2:i), PhiY, PhiX((i + 1):N)];
        RetX = [X(:,2:i), X(:,1), X(:,(i + 1):N)];
    end
endfunction

% Tire une variable aléatoire X' en partant de X via le noyau K
function YStar = DrawRandom(Y, sigma)
    d = length(Y);
    YStar = (Y + sigma * randn(d, 1)) / sqrt(1 + sigma^2);
endfunction

% Algorithme utilisé pour estimer p
function p = AlgorithmP(q, d, N, T, sigma)
    m = 1;
    disp("Generating vector...");
    X = randn(d, N);
    disp("Applying Phi...");
    PhiX = Phi(X);
    disp("Sorting vector...");
    [X, PhiX] = MergeSort(X, PhiX);
    Lm = PhiX(1);
    disp("Computing M...");
    while Lm < q
        R = floor(2 + (N - 1) * rand);
        X(:,1) = X(:,R);
        PhiX(1) = PhiX(R);
        for t = 1:T
            YStar = DrawRandom(X(:,1), sigma);
```

```
            PhiYStar = Phi(YStar);
            if (PhiYStar > Lm)
                X(:,1) = YStar;
                PhiX(1) = PhiYStar;

            end
        end
        [X, PhiX] = Insert(X, PhiX);
        Lm = PhiX(1);
        m++;
    end
    p = (1 - 1 / N)^(m - 1);
endfunction

% Algorithme utilisé pour estimer q
function q = AlgorithmQ(p, d, N, T, sigma)
    disp("Generating vector...");
    X = randn(d, N);
    disp("Applying Phi...");
    PhiX = Phi(X);
    disp("Sorting vector...");
    [X, PhiX] = MergeSort(X, PhiX);
    Lm = PhiX(1);
    M = ceil(log(p) / log(1 - 1 / N));
    disp("Computing M...");
    for m = 2:M
        R = floor(2 + (N - 1) * rand);
        X(:,1) = X(:,R);
        PhiX(1) = PhiX(R);
        for t = 1:T
            YStar = DrawRandom(X(:,1), sigma);
            PhiYStar = Phi(YStar);
            if (PhiYStar > Lm)
                X(:,1) = YStar;
                PhiX(1) = PhiYStar;

            end
        end
        [X, PhiX] = Insert(X, PhiX);
        Lm = PhiX(1);
    end
    q = Lm;
endfunction

% Algorithme pour estimer q, modifié pour calculer l'intervalle de confiance
function [q,qm,qp] = AlgorithmQError(p, d, N, T, sigma)
    X = randn(d, N);
    PhiX = Phi(X);
    [X, PhiX] = MergeSort(X, PhiX);
    Lm = PhiX(1);
    M = ceil(log(p) / log(1 - 1 / N));
    Mm = floor(- N * log(p) - 2 * sqrt(- N * log(p)));
    Mp = ceil(- N * log(p) + 2 * sqrt(- N * log(p)));
    for m = 2:Mp
        R = floor(2 + (N - 1) * rand);
        X(:,1) = X(:,R);
```

```
        PhiX(1) = PhiX(R);
        for t = 1:T
            YStar = DrawRandom(X(:,1), sigma);
            PhiYStar = Phi(YStar);
            if (PhiYStar > Lm)
                X(:,1) = YStar;
                PhiX(1) = PhiYStar;

            end
        end
        [X, PhiX] = Insert(X, PhiX);
        Lm = PhiX(1);
        if m == Mm
            qm = Lm;
        end
        if m == M
            q = Lm;
        end
    end
    qp = Lm;
endfunction
```