Méthodes de Simulation spécifiques pour des lois de Probabilités discrètes

Nous présentons quelques méthodes qui peuvent être efficientes pour des lois de probabilités particulières :

1) Loi binomiale :

Une variable aléatoire X de loi binomiale $B(n,p)$ peut être interprétée comme la somme de "n" expériences indépendantes de Bernoulli.

Ainsi, il existe une façon d'échantillonnage en générant "n" nombres aléatoires $U_1, U_2, \ldots, U_n \sim U(0,1)$ et en faisant égaliser X a celles d'entre elles qui sont inférieures à p.

Si nous considérons la $i^{\text{ème}}$ expérience succès lorsque $U_i < p$ et que la probabilité de cet événement est égale à p, il est claire que l'on obtient une variable binomiale de paramètres n et p.

L'algorithme est :

Faire $X=0$
Répéter n fois
  Générer $U \sim U(0,1)$
  Si $U < p$, faire $X = X+1$
Sortir X

Cette méthode requiere "n" nombres aléatoires et "n" vérifications (comparaisons). Pour cela, on peut opter pour la méthode suivante basée sur la formule recurrente d'inversion qui permet d'obtenir la probabilité $P(X=i+1)=p_{i+1}$

$$P(X=i+1)=p_{i+1}=C_n^{i+1}\, p^{i+1}\,(1-p)^{n-(i+1)}$$

$$=\frac{(n-i)p}{(i+1)(1-p)}\,p_i=\frac{(n-i)p}{(i+1)(1-p)}\,P(X=i)$$

Dans l'algorithme suivant, on note $P=P(X=i)$ et $F=P(X\leq i)$.

Générer $U \rightsquigarrow U(0,1)$

Faire $i=0$, $P=F=(1-p)^n$

Jusqu'à ce que $U\leq F$

Faire $P=\frac{(n-i)p}{(i+1)(1-p)}P$, $F=F+P$

$i=i+1$

Sortir $X=i$

le nombre de comparaisons (vérifications) est un plus que la valeur de X, pour cela, on a besoin, en moyenne, de $(1+np)$ comparaisons pour générer la variable X. Evidemment, si le procédé est répété plusieurs fois pour les mêmes n et p, les F_i peuvent s'accumuler, comme dans la méthode générale d'inversion. Lorsque n est grand, la méthode est peu efficiente et on peut opter pour une méthode générale. Autre possibilité, à partir du théorème centrale de la limite, en se basant sur que

$$(*)\quad Z=\frac{X-np+0,5}{\sqrt{np(1-p)}}$$

tend vers une loi normale $N(0,1)$ lorsque n croît indéfiniment.

Dans ce cas, on génère $Z \sim N(0,1)$,

on résoud (*) précédente en X et on arrondit à une valeur entière non négative, en faisant

$$X = \max\left\{0,\ \mathrm{ent}\left(-0{,}5 + np + Z\sqrt{np(1-p)}\right)\right\}.$$

2) Loi de Poisson :

Dans le cas de la loi de Poisson $\mathcal{P}(\lambda)$, pour λ petit, on peut utiliser la méthode d'inversion, selon la formule de recurrence $P(X=i+1) = \frac{\lambda}{i+1} P(X=i)$

Avec la même notation F, P d'avant, on obtient :

Générer $U \sim \mathcal{U}(0,1)$
Faire $i=0$, $F = P = e^{-\lambda}$
Jusqu'à que $U < F$
  Faire $P = \frac{\lambda}{(i+1)} P$, $F = F + P$
  $i = i+1$
Sortir $X = i$

le nombre de comparaisons sera "une" plus que la valeur de Poisson générée. En moyenne, il sera nécessaire $1+\lambda$ comparaisons

Un autre algorithme intéressant se base sur la relation qui existe entre la loi de Poisson et la loi exponentielle.

Si les durrées (temps) T_i entre les évènements d'un processus ponctuel sont i.i.d (indépendantes identiquement distribués) selon une loi exponentielle $\mathcal{E}xp(\lambda)$, le nombre X des évènements dans un intervalle de durrée 1 suit la loi $\mathcal{P}(\lambda)$.

Alors $$\sum_{i=1}^{X} T_i \leq 1 \leq \sum_{i=1}^{X+1} T_i$$

comme $T_i = \frac{-1}{\lambda} \ln(U_i)$, on peut re-écrire la formule antérieure

$$-\frac{1}{\lambda} \sum_{i=1}^{X} \ln(U_i) \leq 1 \leq \frac{-1}{\lambda} \sum_{i=1}^{X+1} \ln(U_i)$$

ou bien $$\prod_{i=1}^{X} U_i \geqslant e^{-\lambda} \geqslant \prod_{i=1}^{X+1} U_i$$

d'où, on obtient l'algorithme :

Faire $a = 1$, $k = 0$
Jusqu'à que $a < e^{-\lambda}$
Générer $U_k \rightsquigarrow \mathcal{U}(0,1)$
Faire $a = aU_k$
$k = k+1$
Sortir $X = k$

Pour λ grand, on peut approximer (approcher) la loi de Poisson par une loi normale. Lorsque $\lambda > 10$, la loi de

$$(**) \qquad Z = \frac{X - \lambda + 0{,}5}{\sqrt{\lambda}}$$ tend vers une normale $N(0,1)$.

On génère $Z \rightsquigarrow N(0,1)$, on résoud $(**)$ précédente, par rapport à X et on arrondit à une valeur entière non négative, on obtient $$X = \max\{0,\ \text{ent}(-0{,}5 + \lambda + Z\sqrt{\lambda})\}.$$

3) Loi géométrique :

La loi géométrique $Ge(p)$ est connue parfois comme la loi exponentielle discrète, puisqu'elle peut être générer en échantillant par discrétisation d'une loi exponentielle. En effet, supposons que $Y \rightsquigarrow \mathcal{E}xp(\lambda)$.

Soit $X = \text{ent}(Y)$. Alors,

$$P(X=r) = P(r \leq Y < r+1) = \int_r^{r+1} \lambda e^{-\lambda s} ds =$$

$$= e^{-\lambda r} - e^{-\lambda(r+1)} \qquad (***)$$

$$= \left(e^{-\lambda}\right)^r \left(1 - e^{-\lambda}\right)$$

pour $r = 0, 1, \ldots,$ est la fonction de loi de probabilité $Ge(p = 1 - e^{-\lambda})$. En prenant $\lambda = -\text{Ln}(1-p)$, on remarque que (***) précédente est identique à la fonction de probabilité d'une loi géométrique de paramètre p. Donc

$$X = \text{ent}\left(\frac{\text{Ln}\, U}{\text{Ln}(1-p)}\right) = \text{ent}\left(\frac{-V}{\text{Ln}(1-p)}\right)$$

où $V = -\text{Ln}\, U \rightsquigarrow \mathcal{E}xp(1)$.

On conclut que pour générer une variable géométrique $Ge(p)$, premièrement, on génère une variable exponentielle $\mathcal{E}xp(-\text{Ln}(1-p))$, et on arrondit à la valeur entière.

4) Loi binomiale négative :

Il y a plusieurs algorithmes qui ont été proposés pour la génération d'une loi binomiale négative $BN(k,p)$.

Ici, nous citons seulement un seul algorithme qui utilise la relation donnée par l'égalité $P(X \leq r) = P(Y \geq k)$ où $X \rightsquigarrow BN(k,p)$ et $Y \rightsquigarrow B(p, k+r)$.

Si nous disposons d'un procédé pour générer la variable binomiale, il est immédiat de construire un algorithme basé sur l'égalité précédente.

5) Loi hypergéométrique :

On veut échantillonner à partir d'une loi hypergéométrique $\mathcal{H}(m,n,p)$, c'est-à-dire, on fait n tirages sans remise d'un ensemble de m éléments de deux classes, avec p la fraction d'éléments d'une classe et $q = 1-p$ la fraction de l'autre.

Un procédé simple pour générer la variable X, en notant c_i $i = 1, 2$ le nombre restant d'éléments de chaque classe dans la population après X tirages, est le suivant :

Faire $X = 0$, $c_1 = mp$, $c_2 = m - c_1$
Répéter n fois
  Générer $U \rightsquigarrow \mathcal{U}(0,1)$
  Si $U \leq \frac{c_1}{m}$, faire $X = X+1$ et $c_1 = c_1 - 1$
    Sinon, $c_2 = c_2 - 1$
  Faire $m = m - 1$
Sortir X.