Calcul des probabilités
Méthodes de simulation de variables aléatoires

Jean-Baptiste Henniart

October 10, 2017

Contents

1 Généralités

On rappelle le théorème fondamental de la simulation.

Pour toute variable aléatoire d-dimensionnelle X, il existe une fonction borélienne L de $\mathbb{R}^n$ dans $\mathbb{R}^d$ telle que X ait la même loi que $L(U_1, ..., U_n)$, où $U_1, ..., U_n$ sont n variables aléatoires indépendantes et de loi uniforme sur $[0, 1]$.

Ainsi on a par exemple :

- pour la loi binômiale $B(p, n)$,

$$L(U_1, ..., U_n) = \sum_{i=1}^{i=n} 1_{\{U<p\}}$$

- pour la loi géométrique $G(p)$ sur $1, 2, 3...$,

$$L(U_1, ..., U_n) = \inf\{n, U_n < p\}$$

- pour la loi normale $N(0, 1)$,

$$L(U_1, U_2) = \sqrt{-2 \log U_1} * \cos(2 * \pi * U_2)$$

2 La méthode de la fonction de répartition inverse ou fonction quantile

On peut trouver L à l'aide de la fonction de répartition de la variable aléatoire réelle à simuler lorsque celle-ci possède une inverse simple à calculer.

On note $F_X(x) = P(X \leq x)$ la fonction de répartion de la variable aléatoire X.

La méthode des quantiles nous dit alors que si on note $G_X(y) = \inf(x, F_X(x) \geq y)$, alors $G_X(U)$ a la même loi que X, si U suit une loi uniforme sur $[0, 1]$. On prend donc $L(U) = G_X(U)$.

Ainsi pour une loi exponentielle $E(\lambda)$, on :

$$F_X(x) = 1 - \exp(-\lambda * x), \quad G_X(y) = -1/\lambda * \log(1 - y)$$

Question 1 *Simulation de variables aléatoires de loi binômiale et géométrique, de loi exponentielle et normale.*

En utilisant les rappels ci-dessus, réaliser pour chacune des ces quatre lois :

- *l'écriture d'une fonction Scilab de simulation d'un tirage aléatoire ,*
- *un algorithme de simulation de s tirages (ici encore une fonction Scilab dépendant des paramètres de la loi et du nombre s de tirages),*
- *un histogramme pour vérifier l'adéquation à la loi originale.*

Question 2 *Simulation de loi normale.*

- *On donne une procédure Scilab de simulation de variables aléatoires de loi normale $N(0, 1)$ qu'on écrit dans un fichier* `s_normal.sci` *:*

```
function [x]=s_normal(s,a,b)
x=[];
for i=1:s,
z=rand(1,2);
y=a+b*sqrt(-2*log(y(1)))*cos(2*%pi*y(2))
x=[x y],
```

- *On appelle alors la fonction :*

```
-->getf(``s_normal.sci'',''c'')
-->x=s_normal(1000,0,1)
-->histplot(1000,x)
```

- *Comparer avec la loi originale.*
- *Écrire une autre procédure* `s_normal.sci` *ne faisant appel qu'une seule fois à la fonction* ***rand****.*

3 La méthode du rejet

Si X est une variable aléatoire réelle bornée admettant la densité p sur $\mathbf{R}$, si $(U_k(1), U_k(2))_{k>0}$ est une suite de variables aléatoires indépendantes de loi uniforme sur un rectangle de $\mathbf{R}^2$ contenant le graphe de p, alors

$$\nu = \inf\left(k, U_k(2) \leq p(U_k(1))\right), \quad Y = U_\nu(1)$$

définit une variable aléatoire de même loi que X.

Question 3 *Simulation de la loi beta.*

En utilisant le rappel ci-dessus, écrire un algorithme de simulation d'une loi beta de paramètres $r, s > 0$, par exemple $r = 2, s = 2$

4 Simulation de loi discrètes

La fonction `grand` permet la simulation des lois discrètes usuelles. Voici pour le compléter deux fonctions. La première engendre un échantillon d'une loi discrète, par la méthode d'inversion, la seconde calcule les fréquences empiriques d'un tel échantillon. Ces deux fonctions peuvent être placées à la suite l'une de l'autre dans un fichier `frequences.sci`.

La fonction `ech_dist(x,d,m,n)` retourne une matrice de taille `m*n` dont les coefficients sont des réalisations indépendantes de la loi sur `x` spécifiée par le vecteur `d`, normalise a `1`.

```
function e = ech_dist(x,d,m,n)

taille=min(max(size(x)),length(d));// ajuster les longueurs
x=x(1:taille);
d=d(1:taille);

loi = d/sum(d);                  // normaliser par la somme
loi=cumsum(loi);                 // calculer la fonction de repartition

for i=1:m,
     for j=1:n,
          k=1;
          r=rand(1,1);           // appel de random
          while r>loi(k),        // simulation par inversion
               k=k+1,
          end;
          e(i,j)=x(k);
     end;
end
```

La fonction `freq_emp(ech)` calcule les fréquences empiriques des valeurs différentes de `ech` et retourne un vecteur des valeurs différentes de `ech` (`v`) et un vecteur des fréquences correspondantes (`f`)

```
function [v,f] = freq_emp(ech)

taille=max(size(ech));           // taille de l'echantillon

v=[ech(1)];                      // valeurs differentes
for k = 2:taille,                // parcourir l'echantillon
     if ech(k)~=v then,          // la valeur v(k) est nouvelle
          v = [v,ech(k)];        // la rajouter
     end;
end;
v = mtlb_fliplr(sort(v));        // trier les valeurs trouvees
nbval=max(size(v));              // nombre de valeurs differentes

effectifs = [];                  // effectifs des valeurs
for k = 1:nbval,                 // parcourir les valeurs
     e = length(find(ech==v(k)));// calculer l'effectif de la valeur k
     effectifs = [effectifs,e];  // le rajouter
end;

f=effectifs/sum(effectifs);      // calculer les frequences
```

Question 4 *Charger ces deux fonctions par* ***`getf`*** *et réaliser les deux exemples suivants.*

```
-->getf("frequences.sci")
-->x = ech_dist(["a","b","c"],[0.5,0.3,0.2],1,100)
-->x = ech_dist(["a","b","c"],[0.5,0.3,0.2],1,1000);
-->[v,f] = freq_emp(x)
-->plot2d3("gnn",[1:3]',f',4,"111","frequences",[0,0,4,1])
-->x = ech_dist([1:10],ones(1,10),1,1000);
-->histplot(0.5:10.5,x)
-->[v,f] = freq_emp(x)
-->plot2d3("gnn",v',f',4,"111","frequences",[0,0,11,0.5])
```