Calcul des probabilités
Primitives Scilab pour le calcul des probabilités

Jean-Baptiste Henniart

October 10, 2017

Contents

Lorsqu'on doit traiter un nombre important de variables aléatoires de différentes lois, il est souvent difficile de calculer analytiquement un certain nombre de grandeurs caractéristiques (valeurs moyennes, variances, paramètres de lois, quantiles, etc...). On doit alors avoir recours au calcul numérique et à la simulation.

1 Le générateur aléatoire *rand*

La librairie standard d'un langage d'implémentation sur ordinateur contient généralement un générateur de nombres aléatoires. L'appel à une fonction de type random fournit une suite de nombres $x_1, ..., x_n \in [0, 1]$ sensés être n réalisations $X_1(\omega), ..., X_n(\omega)$ de n variables aléatoires indépendantes de loi uniforme sur $[0, 1]$.

Sous Scilab, la fonction `rand` permet de réaliser une telle simulation, à ceci près qu'elle permet de générer des matrices aléatoires au lieu de simples listes.

Ainsi, l'appel à la fonction :

- `rand`(m,n) fournit une matrice aléatoire de dimension m*n, dont chaque terme est la réalisation d'une suite de m*n variables aléatoires indépendantes de loi uniforme sur $[0, 1]$

- `rand`(a) fournit une matrice aléatoire de même dimension que la matrice a
- `rand`() sans argument retourne un scalaire.

Question 1 *Générer une suite de n réalisations de variables aléatoires de loi uniforme sur* $[0,1]$ *(i.e. une matrice de dimension (1,n)) puis tracer l'histogramme correspondant avec la fonction* ***histplot***

```
-->n=1000
-->x=rand(1,n)
-->histplot(n,x)
```

Que constatez vous ? Ceci vous semble-t-il cohérent ? Pourquoi ?

Question 2 *Augmenter le nombre n. L'histogramme change-t-il significativement ?*

La fonction `rand` permet également de simuler directement une matrice de variables aléatoires indépendantes suivant une autre loi que la loi uniforme. Voici un exemple avec la loi normale.

```
-->n=1000;
-->x=mu*ones(1,n)+sigma*rand(1,n,'normal')
-->histplot(n,x)
-->rand('uniform')
```

La deuxième ligne permet d'obtenir une suite de n variables aléatoires de loi normale $N(\mu, \sigma^2)$.

La quatrième ligne permet de revenir à la simulation ¡¡ uniforme ¿¿. On peut également avoir la simulation ¡¡ normale ¿¿ par défaut, en tapant

```
-->rand('normal')
```

2 La fonction *grand* et les différentes lois simulées en standard

2.1 La fonction *grand*

La fonction `grand` est un générateur aléatoire qui permet de simuler des variables aléatoires d'un certain nombre de lois. Elle s'appelle et fonctionne comme la fonction `rand` à ceci près qu'il faut en plus entrer les arguments des différentes lois. Nous donnons ici quelques exemples utiles.

- loi beta $B(a,b)$

```
-->n=1000
-->a=2
-->b=2
-->x=grand(1,n,'bet',a,b)
-->histplot(n,x)
```

- loi binômiale de paramètres (n, p)

```
-->r=1000
-->n=10
-->p=0.25
-->x=grand(1,n,'bin',n,p)
-->histplot(r,x)
```

- loi du chi-deux de degré de liberté d

```
-->n=1000
-->d=2
-->x=grand(1,n,'chi',d)
-->histplot(n,x)
```

- loi exponentielle

```
-->n=1000
-->lambda=3
-->x=grand(1,n,'exp',lambda)
-->histplot(n,x)
```

- loi gamma de paramètres (a, b)

```
-->n=1000
-->a=2
-->b=2
-->x=grand(1,n,'gam',a,b)
-->histplot(n,x)
```

2.2 Fonctions de répartition

Scilab propose également des fonctions cdf*, à partir desquelles on retrouve la fonction de répartition, la densité et la fonction quantile des lois les plus courantes. On pourra se référer à l'aide de Scilab, pour voir comment les appeler.

- cdfbet - cumulative distribution function Beta distribution,

- cdfbin - cumulative distribution function Binomial distribution,
- cdfchi - cumulative distribution function chi-square distribution,
- cdfchn - cumulative distribution function non-central chi-square distribution,
- cdff - cumulative distribution function F distribution,
- cdffnc - cumulative distribution function non-central f-distribution,
- cdfgam - cumulative distribution function gamma distribution,
- cdfnbn - cumulative distribution function negative binomial distribution,
- cdfnor - cumulative distribution function normal distribution,
- cdfpoi - cumulative distribution function poisson distribution,
- cdft - cumulative distribution function Student's T distribution.

3 Représentations graphiques

3.1 Diagrammes en bâtons avec *plot2d3*

La fonction `plot2d3` permet de représenter des diagrammes en bâtons.

3.2 Histogrammes

La fonction *histplot*

La fonction `histplot` représente des histogrammes. Son premier paramètre peut être un entier (nombre de classes), auquel cas l'histogramme est régulier, ou un vecteur donnant les bornes des classes.

```
-->x=rand(1,1000);
-->xbasc();histplot(10,x)
-->xbasc();histplot([0,0.2,0.5,0.6,0.9,1],x)

-->y=grand(x,"exp",1);
-->xbasc()
-->histplot(10,y)
-->xbasc()
-->histplot(-log([1:-0.1:0.01]),y)
```

La fonction *hist3d*

La fonction `hist3d` représente des histogrammes dans $\mathbf{R}^3$ mais n'effectue pas le calcul des fréquences de classes.

Pour cela, on pourra utiliser la fonction `freq2d` définie ci-dessous. Elle prend en entrée deux vecteurs de bornes, bornex et borney, et une matrice echant, à 2 lignes et ncolonnes.

```
function f = freq2d(echant,bornex,borney)
//      freq2d retourne une matrice dont les
//      coefficients sont les frequences de echant
//      relatives a bornex et borney

kx = length(bornex)-1;
ky = length(borney)-1;

for i=1:kx,
    for j=1:ky,
        f(i,j) = length(find(..
            echant(1,:) >  bornex(i)   &..
            echant(1,:) <= bornex(i+1) &..
            echant(2,:) >  borney(j)   &..
            echant(2,:) <= borney(j+1)));
   end;
end;
f=f/sum(f);
```

La fonction `hist3d` prendra alors en entrée une liste formée de la matrice des fréquences et des deux vecteurs de bornes.

Voici quelques exemples d'utilisation.

```
-->getf("freq2.sci")
-->u=rand(2,2000);
-->plot2d(u(1,:),u(2,:),-4)
-->bx=[0:0.1:1]; by=bx;
-->f=freq2d(u,bx,by);
-->xbasc()
-->hist3d(list(f,bx,by))

-->t1=max(rand(2,2000),"r");
-->t2=min(rand(2,2000),"r");
-->xbasc()
-->plot2d(t1,t2,-4)
-->bx=[0:0.1:1]; by=bx;
-->f=freq2d([t1;t2],bx,by);
```

```
-->xbasc()
-->hist3d(list(f,bx,by))

-->M=[0;0]; S=[1,0.8;0.8,1];
-->v=grand(2000,"mn",M,S);
-->xbasc()
-->plot2d(v(1,:),v(2,:),-4)
-->bx=[-3:0.6:3]; by=bx;
-->f=freq2d(v,bx,by);
-->xbasc()
-->hist3d(list(f,bx,by))
```

3.3 Nuages de points

Pour représenter un nuage de points dans le plan, on peut utiliser `plot2d` avec un style de représentation négatif.

On peut visualiser des nuages de points à trois dimensions à l'aide de `param3d1`, et utiliser la rotation à l'aide de la souris.

```
-->M=[0;0;0];
-->S=[1,0.5,0.9 ; 0.5,1,0.3 ; 0.9,0.3,1];
-->v=grand(2000,"mn",M,S);
-->xbasc()
-->param3d1(v(1,:)',v(2,:)',list(v(3,:)',-4))
```