Option Calcul

L2 Mass **2015-2016**

Formulaire 2 : Génération de variables aléatoires

version du 7 mars 2016

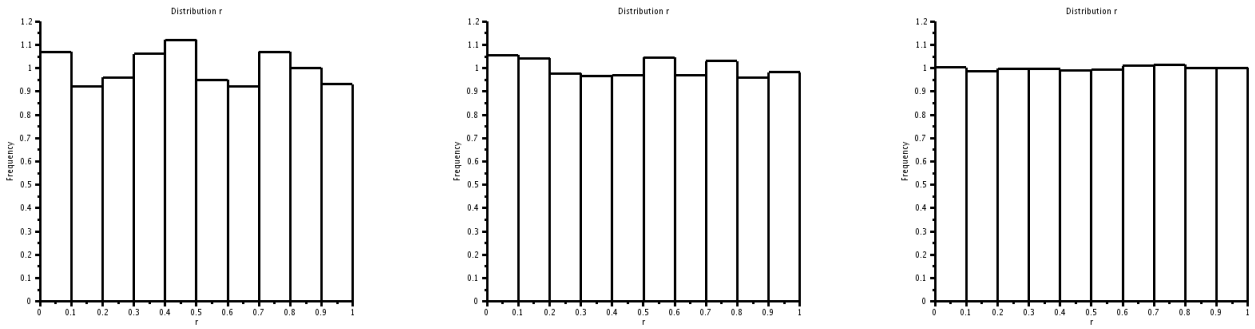
1) Les tirages aléatoires

"Un générateur de nombres pseudo-aléatoires est un algorithme qui génère une séquence de nombres présentant certaines propriétés du hasard. Par exemple, les nombres sont supposés être suffisamment indépendants les uns des autres, et il est potentiellement difficile de repérer des groupes de nombres qui suivent une certaine règle (comportements de groupe). $\cdots$

Les méthodes pseudo-aléatoires sont souvent employées sur des ordinateurs, dans diverses tâches comme la méthode de Monte-Carlo, la simulation ou les applications cryptographiques. $\cdots$

La plupart des algorithmes pseudo-aléatoires essaient de produire des sorties qui sont uniformément distribuées. Une classe très répandue de générateurs utilise une congruence linéaire." (Wikipedia).

1.1) La fonction `rand`. L'instruction `rand(N,M,"uniform")` permet de générer un tableau de `N` lignes et `M` colonnes de nombres aléatoires selon une loi uniforme entre 0 et 1. Les instructions `r=rand(N,1,"uniform")` et `histplot(10,r)` permettent de générer la suite aléatoire puis de visualiser l'histogramme avec 10 classes. La figure 1 représente les distributions obtenues pour `N=1000`, `N=10000` et `N=100000`.

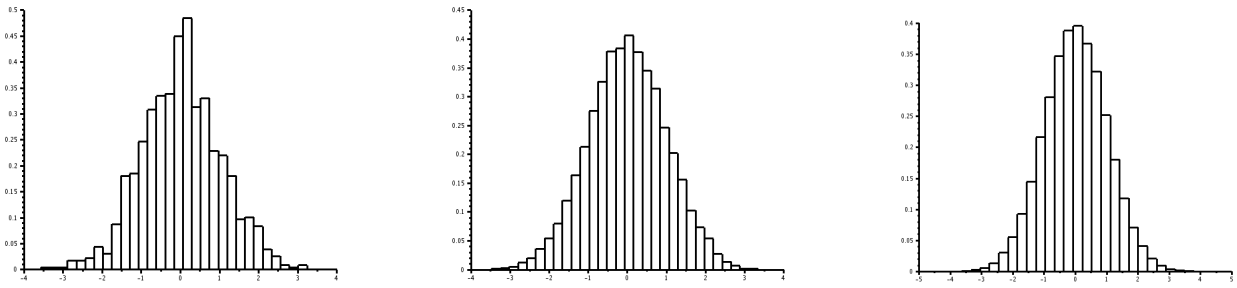


`N=1000` `N=10000` `N=100000`

FIGURE 1 – Distribution pour $N = 1000$, 10000 et 100000 (loi uniforme dans $[0, 1]$)

L'instruction `rand(N,M,"normal")` permet de générer un tableau de `N` lignes et `M` colonnes de nombres aléatoires selon une loi normale de moyenne 0 et de variance 1. La figure 2 représente les distributions obtenues pour `N=1000`, `N=10000` et `N=100000`.

1.2) Exemple 1 : calcul de la moyenne et de la variance. La moyenne et la variance peuvent être calculées explicitement ou par les instructions `m=mean(r)` et `s=variance(r)` (le résulat n'est pas tout à fait le même pour la variance !).



N=1000 N=10000 N=100000

FIGURE 2 – Distribution pour $N = 1000$, 10000 et 100000 (loi normale $m = 0$ et $\sigma = 1$)

Le programme suivant génère un tableau à `N` lignes et 1 colonne de nombres aléatoires, trace l'histogramme puis calcule la moyenne et la variance :

```
clear
N=1000;
// Tirages aléatoires
r=rand(N,1,"uniform");
i_fig=1;scf(i_fig);i_fig==gcf();clf(i_fig);xset("thickness",3);
histplot(10,r);
xtitle("Distribution r","r","Frequency");
m=ones(1,N)*r/N;r_c=r-m*ones(N,1);Var=r_c'*r_c/N;
printf(" moyenne=%f  variance=%f \n ",m,Var);
```

1.3) Exemple 2 : le calcul de π par la méthode de Monte-Carlo. On effectue un tirage aléatoire de N couples de nombre (x, y) dans le carré $[0, 1]^2$. On compte le nombre M de couple (x, y) vérifiant la condition $x^2+y^2 < 1$. La quantité $p = 4\,\dfrac{M}{N}$ est une approximation du nombre π :

```
clear
N=1000;
// Tirages aléatoires
X=rand(N,2,"uniform");

// Estimation du nombre Pi
M=0;
for i=1:N
R=X(i,:)*X(i,:)';
if (R<1) M=M+1; end
end
Pi=4*M/N;erreur=abs(Pi-%pi);
printf("N= %d ; Pi (?)=%f ; erreur=%e \n",N,Pi,erreur);
```

On obtient les résultats suivants :

N	1000	10000	100000	1000000
p	3.1440	3.11760	3.14360	3.141016
erreur	3.7e-02	2.40e-03	5.67e-04	9.88e-04

2) Autres variable aléatoires

2.1) Loi de Bernoulli. Voir par exemple le site :
`https://fr.wikipedia.org/wiki/Loi_de_Bernoulli`.
Le programme suivant simule une variable aléatoire suivant une loi de Bernoulli :

```
clear

function X=Bernoulli(N,p)
x=rand(N,1,"uniform");X=min(1,floor(x/(1-p)));
endfunction

// Tirages aléatoires
p=0.3;N=10000;X=Bernoulli(N,p);
m=ones(1,N)*X/N;Var=0;
for i=1:N Var=Var+X(i)**2/N; end
Var=Var-m**2;
printf("p=%f ; N=%d ; Moyenne=%f ; Variance=%f \n",p,N,m,Var);
```

Remarquez que la notion de `function` est utilisée pour définir une fonction. On obtient les résultats suivants :

```
p=0.300000 ; N=10000 ; Moyenne=0.304400 ; Variance=0.211741
```

2.2) Loi Binomiale. Voir par exemple le site :
`https://fr.wikipedia.org/wiki/Loi_binomiale`
Le programme suivant simule une variable aléatoire suivant une loi de binomiale :

```
clear

function X=Bernoulli(N,M,p)
x=rand(N,M,"uniform");
X=min(1,floor(x/(1-p)));
endfunction

// Tirages aléatoires
p=0.3;N=100000;M=10;Y=Bernoulli(N,M,p);X=Y*ones(M,1);
m=ones(1,N)*X/N;Var=0;
for i=1:N Var=Var+X(i)**2/N; end
Var=Var-m**2;
printf("p=%f ; N=%d ; M=%d ; Moyenne=%f ; Variance=%f \n",p,N,M,m,Var);
```

On obtient les résultats suivants :

```
p=0.500000 ; N=100000 ; M=10 ; Moyenne=5.008500 ; Variance=2.495968
```

et le diagramme en baton de la figure 3.

Exercices

• Exercice 1 : Génération aléatoire “uniforme” et “normale”.

1.1) Utiliser l'instruction `X=rand(N,1,"uniform")` pour générer aléatoirement N nombres dans l'intervalle $[0, 1]$. Calculer la moyenne et la variance et comparer ces résultats au valeurs théoriques. Mêmes questions pour l'instruction `X=rand(N,1,"normal")`.

1.2) Utiliser l'instruction `X=rand(N,2,"uniform")` pour générer aléatoirement N points (x, y) dans le carré $[0, 1]^2$. Calculer la moyenne et la variance de x et y puis la covariance de x et y. Mêmes questions pour l'instruction `X=rand(N,1,"normal")`.

• Exercice 2 : Estimation du nombre π par la méthode de Monte-Carlo.

2.1) Tracer dans une fenêtre le cercle $x^2 + y^2 = 1$. On utilisera la représentation paramétrique :

$$t \in [0, 2\pi] \quad ; \quad x = \cos(t) \quad ; \quad y = \sin(t)$$

2.2) Représenter dans cette fenêtre les points (x, y) générés aléatoirement dans le carré $[-1, 1]^2$ en utilisant un symbole différent selon que le point (x, y) est à l'intérieur ou à l'extérieur du cercle. On utilisera l'instruction `if then else` :

```
if (test) then instructions_1; else instructions_2; end
```

• Exercice 3 : Loi binomiale.

3.1) Loi de Bernoulli. On utilise le programme du paragraphe 2.1). Proposer d'autres façons d'écrire la `function X=Bernoulli(N,p)`. Calculer la moyenne et la variance et comparer aux résultats théoriques.

3.2) Loi binomiale. On utilise le programme du paragraphe 2.2). Calculer la moyenne et la variance de `X` et comparer aux résultats théoriques. Compter le nombre d'occurence de l'évènement `X=k` pour un entier `k` entre 0 et `M`.

3.3) Représenter la statistique de ces résultats par un diagramme en bâtons et comparer à la probabilité d'une loi binomiale. On utilisera les instructions `plot2d3` et `binomial(p,M)`.

3.4) Renormaliser la loi binomiale pour vérifier la convergence vers une loi normale. On vérifiera ce résultat d'abord sur les probabilités puis avec les tirages aléatoires.

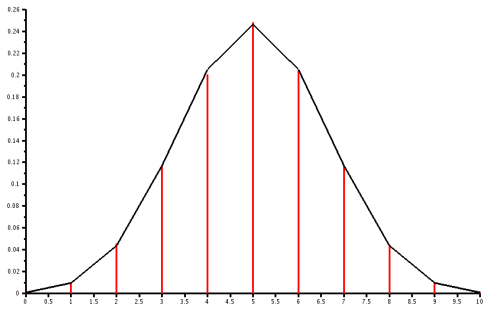


FIGURE 3 – Distribution pour $N = 100000$ et $M = 10$

Solutions

• Exercice 1 :

```
clear
// Exercice 1.1
N=10000;
// Tirages aléatoires
X=rand(N,1,"uniform");
m=ones(1,N)*X/N;X_c=X-m*ones(N,1);Var=X_c'*X_c/N;
printf("Nombre de tirage aléatoire= %d \n",N);
printf("Moyenne=%f ; Variance=%f \n",m,Var);
```

```
clear
// Exercice 1.2
N=1000;
// Tirages aléatoires
X=rand(N,2,"uniform");
m=ones(1,N)*X/N;X_c=X-ones(N,1)*m;Var=X_c'*X_c/N;
printf("Nombre de tirage aléatoire= %d \n",N);
printf("Colonne 1: moyenne = %f  ; variance = %f \n",m(1),Var(1,1));
printf("Colonne 2: moyenne = %f  ; variance = %f \n",m(2),Var(2,2));
printf("Covariance colonnes 1 et 2 = %f \n",Var(1,2));
```

• Exercice 2 :

```
clear
// Exercice 2
N=100;
// Tirages aléatoires
X=rand(N,2,"uniform");
m=ones(1,N)*X/N;
X_c=X-ones(N,1)*m;Y=2*X_c;
printf("m1=%f m2=%f \n",m(1),m(2));

// Tracé
i_fig=1;scf(i_fig);i_fig==gcf();clf(i_fig);xset("thickness",1);
t=0:2*%pi/100:2*%pi;x=cos(t);y=sin(t); plot2d(x,y,5);
// Calcul de la surface par Monte-Carlo
M=0;
for i=1:N
R2=Y(i,:)*Y(i,:)';
if (R2<1) then M=M+1; plot2d(Y(i,1),Y(i,2),-4);
else plot2d(Y(i,1),Y(i,2),-1); end
end
p=4*M/N;erreur=abs(p-%pi);
printf("Nombre de tirage aléatoire= %d \n",N);
printf("Pi (?)=%f ; erreur=%e \n",p,erreur);
```

• Exercice 3 :

```
clear
// Exercice 3.1
function X=Bernoulli(N,p)
x=rand(N,1,"uniform");X=zeros(N,1);
for i=1:N if(x(i)>1-p) X(i)=1; end end
endfunction

// Tirages aléatoires
p=0.3;N=10000;X=Bernoulli(N,p);
m=ones(1,N)*X/N;Var=0;
for i=1:N Var=Var+X(i)**2/N; end
Var=Var-m**2;
printf("Nombre de tirage aléatoire de Bernoulli= %d \n",N);
printf("p=%f ; N=%d ; Moyenne=%f ; Variance=%f \n",p,N,m,Var);

clear
// Exercices 3.2 et 3.3
function X=Bernoulli(N,M,p)
x=rand(N,M,"uniform");X=min(1,floor(x/(1-p)));
endfunction

// Tirages aléatoires
p=0.7;N=10000;M=50;Y=Bernoulli(N,M,p);X=Y*ones(M,1);
m=ones(1,N)*X/N;Var=0;Z=zeros(1,M+1);
for i=1:N Var=Var+X(i)**2/N;Z(X(i)+1)=Z(X(i)+1)+1/N; end
Var=Var-m**2;
printf("Loi binomiale p=%f ; M=%d \n",p,M);
printf("Nombre de tirage aléatoire d une loi binomiale= %d \n",N);
printf("Moyenne=%f ; Variance=%f \n",m,Var);
xset("thickness",3);plot2d3(0:1:M,Z,5);plot2d(0:1:M,binomial(p,M),1);
```

```
clear

function y=Normale(x)
y=exp(-x*x/2)/sqrt(2*%pi);
endfunction

// Convergence vers une loi normale
p=0.7;q=1-p;M=100;
m=M*p;Var=M*p*q;s=sqrt(Var);
printf("Loi binomiale p=%f ; M=%d \n",p,M);
printf("Moyenne=%f ; Variance=%f \n",m,Var);

xset("window",1);xset("thickness",3);
Val=0:1:M;Prob=binomial(p,M);plot2d3(Val,Prob,5);

xset("window",2);xset("thickness",3);
x_n=-3*s:s/100:3*s;
for k=1:length(x_n) y_n(k)=Normale(x_n(k)); end
plot2d(x_n,y_n,1);

Val_n=(Val-m*ones(1,M+1))/s;Prob_n=Prob*s;
plot2d(Val_n,Prob_n,5);
```