Chapitre 8

Simulation de VAR

Ce cours traite de la simulation de variables aléatoires réelles sur un espace probabilisé fini.

1 Fonctions aléatoires du module `random`

1.1 `randint` et `random`

On utilise les fonctions `randint()` et `random()` du module `random` après les avoir importées :

```
from random import random, randint
```

`randint(a,b)`	renvoie un entier aléatoire dans $[\![a,b]\!]$; `a` et `b` doivent être de type entier.
`random()`	renvoie un nombre flottant aléatoire dans l'intervalle $[0;1[$.

Exemple :

```
>>> randint(1,3)
3
>>> randint(1,3)
2
>>> random()
0.027388299549970685
>>> random()
0.9440210743215997
```

1.2 Lois de `randint` et `random`

• Loi de `randint`

L'appel à `randint(a,b)` est programmé pour suivre une loi uniforme sur $[\![a,b]\!]$, ou autrement dit, tous les résultats dans $[\![a,b]\!]$ sont équiprobables.
Pour le vérifier : on appelle un très grand nombre N de fois `randint(1,3)` (par exemple) et on calcule la fréquence d'apparitions des 3 issues possibles ; lorsque N augmente, elles se rapprochent de $1/3$:

```
def test_randint13(N):
    L = [0,0,0]
    for k in range(N):
        resultat = randint(1,3)
        L[resultat-1] += 1
    return [x/N for x in L]
```

```
>>> test_randint13(100)
[0.26, 0.34, 0.4]

>>> test_randint13(1000)
[0.332, 0.355, 0.313]

>>> test_randint13(10000)
[0.33, 0.3413, 0.3287]

>>> test_randint13(100000)
[0.33386, 0.33268, 0.33346]
```

• Loi de `random`.

La fonction `random` ne prend pas de paramètre ; l'appel à `random()` est programmé pour suivre une loi uniforme sur $[0,1[$, c'est à dire que toutes les issues de flottants dans cet intervalle sont équiprobables. Mathématiquement, plutôt, la probabilité que le résultat renvoyé soit dans $[a,b]$ vaut $b-a$ pour tout $(a,b) \in [0,1[^2$. Vérifions-le par exemple pour $a=0$ et $b=1/2$:

```
def test_random(N):
    L = [0,0]
```

```
    for k in range(N):
        resultat = random()
        if resultat <= 0.5:
            L[0] += 1
        else:
            L[1] += 1
    return [x/N for x in L]
```

```
>>> test_random(100)
[0.55, 0.45]

>>> test_random(1000)
[0.505, 0.495]

>>> test_random(10000)
[0.5081, 0.4919]

>>> test_random(100000)
[0.50073, 0.49927]
```

2 Estimation empirique d'une probabilité, d'une espérance

Remarquons, comme nous venons de l'utiliser, l'interprétation empirique d'une probabilité :

Lorsqu'on renouvelle un grand nombre N de fois une expérience aléatoire, la fréquence d'apparition de chaque issue "tend" lorsque $N \longrightarrow +\infty$ vers sa probabilté théorique.

Ce fait permet d'estimer empiriquement une probabilité, en renouvelant un grand nombre N de fois l'expérience et en calculant la fréquence d'apparition de chaque issue (c'est à dire : nombre de réussite/N).

Exemple : on lance un très grand nombre de fois un dé 6 non pipé ; la fréquence d'apparition de chaque face devrait selon toute probabilité être proche de $1/6$.

L'espérance d'une variable aléatoire réelle peut aussi s'interpréter, et s'estimer, empiriquement :

Lors d'une expérience aléatoire, soit X une VAR sur un espace probabilisé fini associé à l'expérience ; on renouvelle un grand nombre N de fois l'expérience ; la moyenne des valeurs prises par X "tend" lorsque $N \longrightarrow +\infty$ vers l'espérance mathématique $E(X)$ de X.

Exemple : on lance un très grand nombre de fois un dé 6 et on effectue la moyenne des tous les résultats obtenus ; elle devrait selon toute probabilité être proche de : $\frac{1+2+3+4+5+6}{6} = 3,5$.

Ces résultats constituent ce que l'on appelle la **loi des grands nombres**. Elle n'est pas si simple à formuler mathématiquement (la convergence est une "convergence en probabilité") et sera établie en 2ème année (loi faible des grands nombres).

3 Simulation d'une loi uniforme

La loi uniforme sur $[\![a;b]\!]$ peut se simuler à l'aide de `randint(a,b)` ; chaque appel à `randint(a,b)` renverra la valeur prise par une variable $X \hookrightarrow \mathscr{U}([\![a;b]\!])$ sur une issue aléatoire de l'expérience.

Par exemple pour simuler le résultat du lancer d'un dé 6 équilibré, on peut appeler `randint(1,6)`.

Estimons empiriquement l'espérance de $\mathscr{U}([\![1;6]\!])$:

```
def estim_esp_uniforme(a,b,N):
    """Renvoie une estimation empirique de l'espérance
    de U(a,b) en renouvelant N fois l'expérience"""
    S = 0
    for k in range(N):
        S += randint(a,b)
    return S/N
```

```
>>> estim_esp_uniforme(1,6,100000)
3.50149
```

Le résultat théorique vaut $\frac{a+b}{2}$, soit ici $3,5$; c'est cohérent.
Soit $X \hookrightarrow \mathscr{U}([\![1;6]\!])$; estimons la probabilité de l'évènement $(X \in \{2,4,6\})$:

```
def estim_proba_uniforme(a,b,L,N):
    Reussite = 0
    for k in range(N):
```

```
        if randint(a,b) in L:
            Reussite += 1
    return Reussite/N
```

```
>>> estim_proba_uniforme(1,6,[2,4,6],100000)
0.50137
```

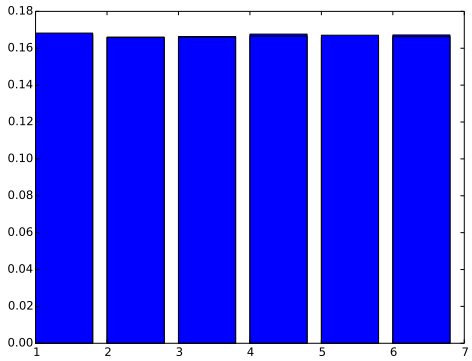
La probabilité théorique vaut $1/2$; c'est cohérent.
Traçons l'histogramme de la loi $\mathscr{U}(\llbracket 1;6 \rrbracket)$; on utilise pour cela la fonction `bar` du module `matplotlib.pyplot`.

```
from matplotlib.pyplot import figure, bar, show

def histogramme_uniforme(a,b,N):
    Abs = [ k for k in range(a,b+1) ]
    Ord = [ 0 for k in range(a,b+1) ]
    for k in range(N):
        X = randint(a,b)
        Ord[X-a] += 1
    Ord = [x/N for x in Ord]
    return Abs, Ord

X, Y = histogramme_uniforme(1,6,100000)
figure(1)
bar(X,Y)
show()
```

On obtient l'histogramme :



4 Simulation d'une loi de Bernoulli

La loi de Bernoulli de paramètre $p \in]0,1[$ peut se simuler à l'aide de la fonction `random` ; puisque la probabilité que l'appel à `random()` renvoie un résultat $\leq p$ vaut $p - 0 = p$, chaque appel à `bernoulli(p)` renvoie 1 avec probabilité p et 0 avec probabilité $1-p$; c'est à dire qu'elle se comporte comme une variable aléatoire qui suit $\mathscr{B}(p)$:

```
def bernoulli(p):
    if random() <= p:
        return 1
    else :
        return 0
```

Estimons alors empiriquement l'espérance de $\mathscr{B}(p)$ pour $p = 0,6$:

```
def estim_esp_bernoulli(p,N):
    """Renvoie une estimation empirique de l'espérance
    de B(p) en renouvelant N fois l'expérience"""
    S = 0
    for k in range(N):
        S += bernoulli(p)
    return S/N
```

```
>>> estim_esp_bernoulli(0.6,100000)
0.59919
```

Le résultat est proche de sa valeur théorique de p, ce qui est cohérent.

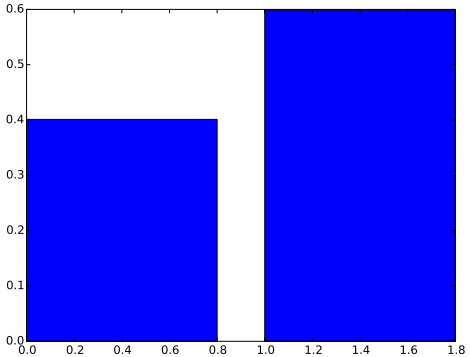
Traçons l'histogramme de $\mathscr{B}(0,6)$:

```
def histogramme_bernoulli(p,N):
    Abs = [ 0, 1]
    Ord = [ 0, 0]
    for k in range(N):
        X = bernoulli(p)
        Ord[X] += 1
    Ord = [x/N for x in Ord]
    return Abs, Ord

X, Y = histogramme_bernoulli(0.6,100000)
figure(2)
```

```
bar(X,Y)
show()
```

On obtient l'histogramme :



5 Simulation d'une loi binomiale

Pour simuler une loi binomiale, on peut appliquer le résultat de cours :

Soient $X_1, X_2, \dots, X_n$ n variables aléatoires indépendantes qui suivent une même loi de Bernoulli de paramètre p. Alors $X = X_1 + X_2 + \cdots + X_n$ suit la loi binomiale $\mathscr{B}(n,p)$.

```
def binomiale(n,p):
    X = 0
    for k in range(n):
        X = X + bernoulli(p)
    return X
```

Estimons alors empiriquement l'espérance de $\mathscr{B}(n,p)$ pour $n = 10$ et $p = 0,6$:

```
def estim_esp_binomiale(n,p,N):
    S = 0
    for k in range(N):
        S += binomiale(n,p)
    return S/N
```

```
>>> estim_esp_binomiale(10,0.6,100000)
5.9992
```

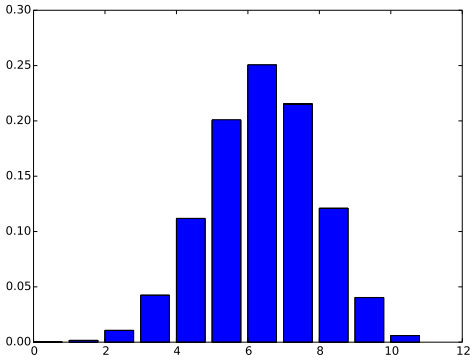
ce qui est cohérent puisque l'espérance théorique est de $n \times p = 10 \times 0,6 = 6$.

Traçons l'histogramme de $\mathscr{B}(10,0,6)$:

```
def histogramme_binomiale(n,p,N):
    Abs = [k for k in range(n+1)]
    Ord = [0 for k in range(n+1)]
    for k in range(N):
        X = binomiale(n,p)
        Ord[X] += 1
    Ord = [x/N for x in Ord]
    return Abs, Ord

X, Y = histogramme_binomiale(10,0.6,1000000)
figure(3)
bar(X,Y)
show()
```

On obtient l'histogramme :



Le tracé de la fonction de répartition s'obtient alors à l'aide des tableaux déjà construits, en utilisant : si $X \hookrightarrow \mathscr{B}(n,p)$:

$$F_X(0) = \mathbb{P}(X = 0) \text{ et } \forall k \in [\![1,n]\!],\ F_X(k) = F_X(k-1) + \mathbb{P}(X = k)$$

```
n = 10
Fx = [0] * (n+1)
Fx[0] = Y[0]
for k in range(1,n+1):
    Fx[k] = Fx[k-1] + Y[k]
figure(4)
bar(X,Fx)
show()
```

On obtient le tracé de la fonction de répartition (sous forme d'histogramme...) :