Coder une variable aléatoire

Marc Lorenzi - 9 juin 2018

In [1]:
```
from math import sqrt
from fractions import Fraction
```

Dans ce notebook nous allons mettre en place quelques fonctions permettant de coder simplement et élégamment la loi d'une variable aléatoire. Nous verrons que notre codage s'adapte immédiatement à la représentation d'un couple de variables aléatoires et permet de calculer facilement espérance, variance, covariance et tout un tas d'autres choses.

Un autre titre pour ce notebook pourrait être : **dictionnaires**.

1. Coder la loi d'une variable aléatoire

Pour coder la loi d'une variable aléatoire X (prenant un nombre fini de valeurs, cela va de soi), nous allons utiliser un **dictionnaire** que nous noterons comme la variable aléatoire, X. Bref, nous oublierons l'ensemble de départ de X pour ne garder que sa loi. Les clés du dictionnaire sont les valeurs prises par X et pour toute clé x, $X[x] = P[X = x]$.

Nous allons coder les lois usuelles dans la suite de cette section.

1.1 Loi de Bernoulli

Commençons par la loi de Bernoulli de paramètre $p \in [0, 1]$.

In [2]:
```
def Bernoulli(p):
    return {0:1-p, 1:p}
```

In [3]:
```
Bernoulli(Fraction(1,3))
```

Out[3]:
```
{0: Fraction(2, 3), 1: Fraction(1, 3)}
```

1.2 Les dictionnaires en Python

La structure de dictionnaire permet d'associer des **valeurs** à des objets, les **clés** du dictionnaire. Les clés peuvent être à peu près n'importe quoi (ce n'est pas tout à fait vrai mais n'entrons pas dans les détails), les valeurs aussi.

Quelques fonctions permettent de travailler avec les dictionnaires. Nous venons de voir comment en initialiser un dans la fonction `Bernoulli`. Si d est un dictionnaire, voici quelques méthodes associées :

- Obtenir la valeur associée à une clé k : `d[k]`
- Associer à la clé k la valeur v : `d[k]=v`
- Le dictionnaire d a-t-il la clé k ? `k in d`
- Supprimer la clé k : `del d[k]`
- Itérer sur l'ensemble des clés : `for k in d ...`

Les trois méthodes ci-dessous renvoient des objets de type `view`. Je ne précise pas plus avant.

- Obtenir toutes les clés : `d.keys()`. Une instruction du type `list(d.keys())` renvoie la liste des clés.
- Obtenir toutes les valeurs : `d.values()`
- Obtenir tous les couples (clé, valeur) : `d.items()`

et encore bien d'autres méthodes. Consultez la documentation de Python.

En exemple, voici en une ligne l'ensemble des valeurs prises par une variable aléatoire codée avec notre modèle :

In [4]:
```
def valeurs_prises(X):
    return list(X.keys())
```

In [5]:
```
valeurs_prises(Bernoulli(Fraction(1, 3)))
```

Out[5]:
```
[0, 1]
```

Et voici une fonction qui vérifie si une variable aléatoire en est bien une.

In [6]:
```
def est_va(X):
    s = 0
    for x in X:
        if X[x] < 0 or X[x] > 1: return False
        s = s + X[x]
    return s == 1
```

In [7]:
```
est_va(Bernoulli(Fraction(1,3)))
```

Out[7]:
```
True
```

1.3 Fonctions utiles

Avant de passer aux lois binomiale et hypergéométrique, définissons quelques fonctions bien utiles.

La fonction `power_dn` prend en paramètres deux entiers n et k et renvoie $(n)_k = n(n-1)\dots(n-k+1)$.

In [8]:
```
def power_dn(n, k):
    p = 1
    for i in range(k):
        p = p * (n - i)
    return p
```

In [9]:
```
power_dn(6, 3)
```

Out[9]:
```
120
```

La fonction `factorial` calcule ... la factorielle de l'entier n.

In [10]:
```
def factorial(n):
    return power_dn(n, n)
```

In [11]:
```
factorial(10)
```

Out[11]:
```
3628800
```

Et enfin, la fonction `binom` prend deux entiers n et k en paramètres et renvoie le coefficient binomial $\binom{n}{k}$.

In [12]:
```
def binom(n, k):
    return power_dn(n, k) // factorial(k)
```

In [13]:
```
binom(6, 3)
```

Out[13]:
```
20
```

Remarquons que cette fonction marché très bien même si $k > n$:

In [14]:
```
binom(5, 12)
```

Out[14]:
```
0
```

1.4 La loi binomiale

La fonction `Binomial` prend en paramètres un entier n et un réel $p \in [0, 1]$ et renvoie (la loi d') une variable aléatoire suivant la loi binomiale $\mathcal{B}(n, p)$. Dans la suite j'abuserai et j'identifierai la variable aléatoire et sa loi.

In [15]:
```
def Binomial(n, p):
    X = {}
    for k in range(n + 1):
        X[k] = binom(n, k) * p ** k * (1 - p) ** (n - k)
    return X
```

In [16]:
```
Binomial(5, Fraction(1, 4))
```

Out[16]:
```
{0: Fraction(243, 1024),
 1: Fraction(405, 1024),
 2: Fraction(135, 512),
 3: Fraction(45, 512),
 4: Fraction(15, 1024),
 5: Fraction(1, 1024)}
```

In [17]:
```
est_va(_)
```

Out[17]:
```
True
```

1.5 La loi hypergéométrique

Soient r, n, N trois entiers tels que $r, n \leq N$. La loi hypergéométrique de paramètres r, n, N est définie ainsi : pour $0 \leq k \leq n$,

$$P(X = k) = \frac{\binom{r}{k}\binom{N-r}{n-k}}{\binom{N}{n}}$$

In [18]:
```
def Hypergeom(r, n, N):
    X = {}
    for k in range(n + 1):
        X[k] = Fraction(binom(r, k) * binom(N - r, n - k), binom(N, n)
    return X
```

Exemple : une urne contient $N = 20$ boules. Parmi celles-ci, $r = 8$ boules sont rouges. On tire $n = 10$ boules dans l'urne **sans remise**. Soit X le nombre de boules rouges tirées. Alors X suit une loi hypergéométrique $\mathcal{H}(r, n, N)$.

In [19]:
```
Hypergeom(8, 10, 20)
```

Out[19]:
```
{0: Fraction(3, 8398),
 1: Fraction(40, 4199),
 2: Fraction(315, 4199),
 3: Fraction(1008, 4199),
 4: Fraction(1470, 4199),
 5: Fraction(1008, 4199),
 6: Fraction(315, 4199),
 7: Fraction(40, 4199),
 8: Fraction(3, 8398),
 9: Fraction(0, 1),
 10: Fraction(0, 1)}
```

Remarquez que les événements $\{X = 9\}$ et $\{X = 10\}$ sont de probabilité nulle, ce qui est heureux puisqu'ils sont vides.

In [20]:
```
est_va(_)
```

Out[20]:
```
True
```

2. Espérance, variance

2.1 Formule de transfert

Au lieu de réécrire plusieurs fois la même chose, codons une fonction très générale `gen_esp`. Cette fonction prend en paramètres une fonction $f : \mathbb{R} \to \mathbb{R}$ et une variable aléatoire réelle X. Elle renvoie $E(f(X))$, l'espérance de la variable aléatoire $f(X)$.

Comment ? Eh bien mettons que $X : \Omega \to \mathbb{R}$. Le théorème de transfert nous dit que

$$E(f(X)) = \sum_{x \in X(\Omega)} f(x) P(X = x)$$

La fonction `gen_esp` est donc facile à écrire.

In [21]:
```
def gen_esp(f, X):
    s = 0
    for x in X:
        s = s + f(x) * X[x]
    return s
```

In [22]:
```
gen_esp(lambda x:  x ** 2, Binomial(5, Fraction(1, 3)))
```

Out[22]:
```
Fraction(35, 9)
```

2.2 Moments d'une variable aléatoire réelle

Définition : Pour tout entier $n \in \mathbb{N}$, le moment d'ordre n de la variable aléatoire X est l'espérance de X^n :

In [23]:
```
def moment(X, n):
    return gen_esp(lambda x: x ** n, X)
```

In [24]:
```
moment(Binomial(5, Fraction(1, 3)), 2)
```

Out[24]:
```
Fraction(35, 9)
```

2.3 Espérance

L'espérance est donc le moment d'ordre 1 !

In [25]:
```
def esperance(X):
    return moment(X, 1)
```

In [26]:
```
esperance(Binomial(5, Fraction(1, 3)))
```

Out[26]:
```
Fraction(5, 3)
```

On retrouve le résultat bien connu. Si X suit une loi binomiale $\mathcal{B}(n, p)$, alors $E(X) = np$.

In [27]:
```
esperance(Hypergeom(8, 10, 20))
```

Out[27]:
```
Fraction(4, 1)
```

On retrouve le résultat moins bien connu. Si X suit une loi hypergéométrique $\mathcal{H}(r, n, N)$, alors $E(X) = \frac{nr}{N}$.

2.4 Variance

Et voici en une ligne la variance.

In [28]:
```
def variance(X):
    return moment(X, 2) - esperance(X) ** 2
```

In [29]:
```
variance(Binomial(5, Fraction(1, 3)))
```

Out[29]:
```
Fraction(10, 9)
```

Le résultat général pour la loi binomiale $\mathcal{B}(n,p)$ est $V(X) = np(1-p)$.

In [30]:
```
variance(Hypergeom(8, 10, 20))
```

Out[30]:
```
Fraction(24, 19)
```

Le résultat général pour la loi hypergéométrique $\mathcal{H}(r,n,H)$ est $V(X) = np(1-p)\frac{N-n}{N-1}$ où $p = \frac{r}{N}$.

In [31]:
```
p = Fraction(8, 20)
10 * p * (1 - p) * Fraction(20 - 10, 20 - 1)
```

Out[31]:
```
Fraction(24, 19)
```

On a bien sûr pour pas cher l'écart-type d'une variable aléatoire.

In [32]:
```
def ecart_type(X):
    return sqrt(variance(X))
```

Exercice : Le **moment centré** d'ordre n d'une variable aléatoire réelle est $M_n(X) = E(X - E(X))^n$. Écrire une fonction `moment_centré(X, n)` .

3. Couples de variables aléatoires

Comment coder un couple de variables aléatoires ? Mais dans un dictionnaire bien entendu ! Sauf que cette fois-ci, les clés du dictionnaire sont des couples.

3.1 L'exemple

Nous allons travailler sur l'exemple de la loi du couple (X, Y) où X est l'ordre d'une permutation de $\mathfrak{S}_6$ et Y est le nombre d'orbites non triviales de la permutation. Il nous faut d'abord trouver la loi conjointe. Pour cela, listons toutes les permutations de $\mathfrak{S}_6$ (il y en a $6! = 720$) par la nature de leur décomposition en produit de cycles de supports disjoints. Dans l'ordre ci-dessous : le type de permutation, entre parenthèses le nombre de telles permutations, puis leur ordre et leur nombre d'orbites.

- id : (1) $1, 0$
- (ij) : (15) $2, 1$
- (ijk) : (40) $3, 1$
- $(ijkl)$: (90) $4, 1$
- $(ijklm)$: (144) $5, 1$
- $(ijklmn)$: (120) $6, 1$
- $(ij)(kl)$: (45) $2, 2$
- $(ij)(kl)(mn)$: (15) $2, 3$
- $(ijk)(lm)$: (120) $6, 2$
- $(ijk)(lmn)$: (40) $3, 2$
- $(ijkl)(mn)$: (90) $4, 2$

Au lecteur le soin de vérifier tous ces chiffres. C'est un excellent exercice.

In [33]:
```
XY = {
    (1,0): Fraction(1, 720),
    (2,1): Fraction(15, 720),
    (3,1): Fraction(40, 720),
    (4,1): Fraction(90, 720),
    (5,1): Fraction(144, 720),
    (6,1): Fraction(120, 720),
    (2,2): Fraction(45, 720),
    (2,3): Fraction(15, 720),
    (6,2): Fraction(120, 720),
    (3,2): Fraction(40, 720),
    (4,2): Fraction(90, 720)

}
```

3.2 Lois marginales

Soit $Z = (X, Y)$ un couple de variables aléatoires réelles. Connaissant la loi de Z, commet calculer la loi de X ?

Il suffit de sommer sur les valeurs prises par Y :

$$P(X = x) = \sum_y P(X = x, Y = y)$$

où y varie dans l'ensemble des valeurs prises par Y. On obtient évidemment, en sommant sur x, la loi de Y. La fonction `loi_marginale` fait le travail. Elle prend un paramètre Z (représentant un couple (X, Y) de variables aléatoires) et un entier j qui vaut 0 ou 1. Si $j = 0$ la fonction renvoie la loi de X. Si $j = 1$ elle renvoie la loi de Y.

In [34]:
```
def loi_marginale(Z, j):
    X = {}
    for t in Z:
        x = t[j]
        if x in X: X[x] = X[x] + Z[t]
        else: X[x] = Z[t]
    return X
```

Reprenons notre exemple fétiche.

In [35]:
```
X = loi_marginale(XY, 0)
print(X)
```

```
{1: Fraction(1, 720), 2: Fraction(5, 48), 3: Fraction(1, 9), 4: Frac
tion(1, 4), 5: Fraction(1, 5), 6: Fraction(1, 3)}
```

In [36]:
```
Y = loi_marginale(XY, 1)
print(Y)
```

```
{0: Fraction(1, 720), 1: Fraction(409, 720), 2: Fraction(59, 144), 3
: Fraction(1, 48)}
```

Petite vérification ?

In [37]:
```
est_va(X), est_va(Y)
```

Out[37]:
```
(True, True)
```

Tout va bien. Au passage :

In [38]:
```
print(esperance(X), variance(X))
print(esperance(Y), variance(Y))
```

```
3271/720 922079/518400
29/20 1051/3600
```

3.2 Covariance

Étant données deux variables aléatoires X et Y, leur covariance est $Cov(X, Y) = E(XY) - E(X)E(Y)$. Avec un peu de réflexion on se rend compte que pour calculer $E(XY)$ il suffit d'utiliser la fonction `gen_esp` que nous avons déjà vue, sauf que maintenant la fonction f est une "fonction de deux variables" ! Bref, tout est déjà écrit, il suffit de recoller les morceaux.

In [39]:
```
def covariance(Z):
    EX = esperance(loi_marginale(Z, 0))
    EY = esperance(loi_marginale(Z, 1))
    EXY = gen_esp(lambda z: z[0] * z[1], Z)
    return EXY - EX * EY
```

In [40]:
```
covariance(XY)
```

Out[40]:
```
Fraction(-2459, 14400)
```

In [41]:
```
float(_)
```

Out[41]:
```
-0.17076388888888888
```

In [42]:
```
print(ecart_type(X) * ecart_type(Y))
```

```
0.7206123100840118
```

Pourquoi ce dernier calcul ? Rappelons-nous : étant données deux variables aléatoires X et Y on a $|Cov(X,Y)| \leq \sigma(X)\sigma(Y)$, le produit de leurs écarts-types. Si X et Y ont des écarts-types non nuls, on pose

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sigma(X)\sigma(Y)}$$

Proposition :

- $-1 \leq \rho(X,Y) \leq 1$
- $\rho(X,Y) = \pm 1$ si et seulement si il existe trois réels a, b, c, $(a,b) \neq (0,0)$ tels que $aX + bY + c = 0$ presque sûrement.
- Si X et Y sont indépendantes, $\rho(X,Y) = 0$.

Démonstration : faite en cours.

Définition $\rho(X,Y)$ est le coefficient de corrélation des variables X et Y.

In [43]:
```
def correlation(Z):
    X = loi_marginale(Z, 0)
    Y = loi_marginale(Z, 1)
    return covariance(Z) / (ecart_type(X) * ecart_type(Y))
```

Et pour finir, le coefficient de corrélation de la signature et de l'ordre dans $\mathfrak{S}_5$.

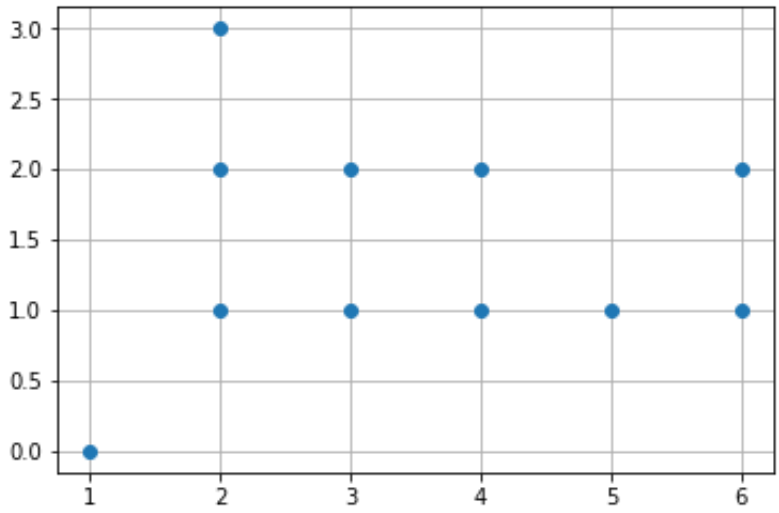
In [44]:
```
correlation(XY)
```

Out[44]:
```
-0.2369705408848491
```

Que signifie cette valeur ? Traçons les couples (x,y) des valeurs prises par le couple (X,Y).

In [45]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [46]:
```
sx = [x for (x,y) in XY]
sy = [y for (x,y) in XY]
plt.plot(sx, sy, 'o')
plt.grid()
plt.show()
```



si notre coefficient de corrélation était ± 1, les points seraient alignés. Un coefficient de 0 indiquerait des points "un peu n'importe comment". Ici, je n'ose rien dire :-).

Mini projet : automatiser tout cela dans $\mathfrak{S}_n$ (avec n quelconque mais raisonnable, cela va de soi). Voici un plan possible :

- Écrire une fonction qui calcule toutes les permutations de $\mathfrak{S}_n$.
- Utiliser pour le reste des étapes les fonctions qui ont été écrites dans le notebook dédié aux permutations.

In [ ]:
```
```