

## SIMULATIONS cas discret

### 1/loi binomiale

Pour une épreuve de Bernoulli de probabilité de succès  $p$  on choisit un nombre  $X$  au hasard entre 0 et 1 grâce à **`X=random()`** ; si  $X < p$  c'est un succès sinon un échec.

Pour une loi binomiale  $B(n,p)$  on recommence  $n$  fois une lois de Bernoulli , en comptant le nombre  $c$  de succès.

On fait cela  $N$  fois ,pour  $i$  in range( $N$ ) on fait  $i$  expériences :chaque expérience consistant à faire  $n$  fois un tirage de  $x$  uniforme sur  $[0 ;1]$  et si  $x < p$  c'est un succès .  $N$  sera grand et on compte les fréquences pour chaque valeur de  $c$  qui va de 0 à  $n$  par  $s[c]=s[c]+1$

On va faire un programme qui compare la simulation à la loi exacte ;pour cela on va importer de `scipy.special` la fonction `binom` : `binom(6,3)` donne 2

On fera attention à l'indentation

```
from random import random
from scipy.special import binom
def binomial(N,n,p):
    s=(n+1)*[0];e=(n+1)*[0]
    for i in range(N):
        c=0
        for j in range(n):
            x=random()
            if x<p:
                c=c+1
        s[c]=s[c]+1
    for i in range(n+1):
        s[i]=s[i]/N
    for i in range(n+1):
        e[i]=binom(n,i)*p**i*(1-p)**(n-i)
    return(s,e)
    version avec graphe :
```

```
from random import random
from scipy.special import binom
import matplotlib.pyplot as plt
def binomial(N,n,p):
    s=(n+1)*[0];e=(n+1)*[0]
    for i in range(N):
        c=0
        for j in range(n):
            x=random()
            if x<p:
                c=c+1
        s[c]=s[c]+1
    for i in range(n+1):
        s[i]=s[i]/N
    for i in range(n+1):
        e[i]=binom(n,i)*p**i*(1-p)**(n-i)
    r=range(n+1)
    plt.plot(r,s,'o',color='black')
    plt.plot(r,e,'*',color='red')
    plt.axis([0,n+1,0,1])
    plt.show()
```

**exercice** : simuler une loi de Poisson  $P(\lambda)$  par  $B(N, \lambda/N)$  on comparera les valeurs  $p(X=k)$  avec  $\frac{e^{-\lambda} \lambda^k}{k!}$

pour  $k$  de 0 à 5 (plus  $N$  grand plus l'approximation sera correcte). Voir aussi à la fin.

## 2/loi hypergéométrique :

On pourra utiliser le module random de python et la fonction sample(population,k) qui choisit  $k$  éléments sans remise parmi la population :

Exemple : prenons 5 cartes d'un jeu de 32 et cherchons la probabilité d'avoir au moins un as : montrer que c'est environ 0.5119

La population sera définie par

```
cartes=[x,y] for x in range(1,9) for y in ['coeur','trefle','pique','carreau']
```

```
from random import sample
```

```
def poker(N):
    c=0
    for i in range(N):
        x=sample(cartes,4)
        if x[0][0]==1 or x[1][0]==1 or x[2][0]==1 or x[3][0]==1 or x[4][0]==1:
            c=c+1
    return(c/N)
```

## 3/a)loi géométrique de paramètre p :

On répète tirer un nombre  $X$  au hasard entre 0 et 1 jusqu'à avoir un succès soit  $X \leq p$

On compte le nombre  $c$  de tirage

Dans cette simulation on va juste vérifier l'espérance  $1/p$  soit la moyenne des résultats  $c$  lorsqu'on fait  $N$  fois l'expérience.

```
from random import random
def geo(N,p):
    s=0
    for i in range(N):
        a=random()
        c=1
        while a>p:
            a=random()
            c=c+1
        s=s+c
    return(s/N)
```

b) On va simuler l'attente de deux piles consécutifs d'une pièce normale soit  $p=0.5$  et vérifier que l'espérance est 6 : on choisira un entier 0 ou 1 par **randint(a,b)** qui choisit un entier de  $a$  compris à  $b$  compris, on fera **randint(0,1)** et pile sera 1

```
def pp(N) :
    s=0
    for i in range(N):
        a=randint(0,1);b=randint(0,1) ;c=2
        while a!=1 or b!=1:
            a=b ;b=randint(0,1) ;c=c+1
        s=s+c
    return(s/N)
```

Exercice :a) simuler l'espérance du maximum ou minimum de deux lois géométriques de paramètre  $p_1, p_2$

b)simuler l'espérance de deux piles consécutifs lorsque la probabilité de pile est  $p$

on donne que l'espérance est  $(1+p)/p^2$  :pour un expérience on fait :

```
a1=random() a2=random() c=2 while not(a1<p and a2<p) : a1=a2 ;a2=random() c=c+1
```

#### 4/Nombre de séries dans une suite d'épreuves de Bernoulli

Si on lance  $n$  fois une pièce de monnaie on a un certains nombres de séries (pile =1 et face=0)

Par exemple si on a :110111001 ici  $n=9$  et il y a 5 séries

a)On voudrait connaitre l'espérance : faire un programme  $s(u)$  qui compte le nombre de séries d'une liste  $u$  formée de 0 et de 1 :il suffira de voir si  $u[i] \neq u[i+1]$

ensuite on va faire  $N$  fois le choix d'une liste  $u$  au hasard de 0 et de 1 pour une probabilité de  $\frac{1}{2}$  on pourra utiliser `randint(0,1)` **mais que faire si on veut simuler une probabilité  $p$  par exemple  $\frac{1}{3}$  ,  $\frac{1}{4}$  ?** puis pour chacune des  $N$  listes  $u$  on compte le nombre de séries puis on donnera la moyenne qui sera une approximation de  $E(X)$

calcul exact : suite de  $n$  lancés  $p_1 p_2 \dots p_n$

on pose  $X=1$  , et pour  $i > 1$   $X_i=1$  si  $p_i \neq p_{i-1}$  et 0 sinon alors  $X=1+X_2+\dots+X_n$

trouver  $E(X_i)=2p(1-p)$  d'où  $E(X)=1+2(n-1)p(1-p)$

b)On voudrait connaitre la loi du nombre de séries : il faut alors faire  $N$  expériences d'où une liste  $u$  de résultats et alors il faut trouver les fréquences

c)Simuler pour trouver l'espérance de la longueur de la plus grande série pour 01110011011110010 c'est 4 .On peut montrer que l'espérance est pour  $n$  grand  $-\ln(n(1-p))/\ln p$

faire un programme qui entre u suite de 0 et de 1 et qui donne la longueur de la plus grande série

```
def maxserie(u):
    m1=0
    n=len(u)
    l1=1
    for i in range(1,n):
        if u[i]== u[i-1]:
            l1=l1+1
        else:
            m1=max(m1,l1)
```

```

l1=1
return(m1)

def sms(N,n):

    e=0
    for i in range(N):
        u=[]
        for j in range(n):
            x=randint(0,1)
            u.append(x)
        e=e+maxserie(u)
    return(e/N)

```

5/complément : simulation de X de loi discrète (finie)  $p(X=1)=p_1, \dots, p(X=n)=p_n$  avec  $p_1+\dots+p_n=1$

On tire un nombre U au hasard entre 0 et 1 :

Si  $U < p_1$  alors  $X=1$

Si  $p_1 \leq U < p_1+p_2$  alors  $X=2$

Si  $p_1+p_2 \leq U < p_1+p_2+p_3$  alors  $X=3$

Etc...

**Car la probabilité que U uniforme soit entre  $a= p_1+\dots+p_{i-1}$  et  $b=p_1+\dots+p_i$  est  $b-a=p_i$**

Par exemple : prenons  $p_1=0.4$  ,  $p_2=0.1$  et  $p_3=0.3$  ,  $p_4=0.2$  d'espérance 2.3

Les probabilités cumulées (fonction de répartition) sont 0.4, 0.5 , 0.8, 1

Def h() : # noter la fonction sans variable

```
U=random()
```

```
If U<0.4 : return(1)
```

```
If U<0.5 : return(2) # si on arrive là c'est qu'il n'y a pas eut return et donc  $U \geq 0.4$ 
```

```
If U<0.8 : return(3)
```

```
return(4) #si on arrive là c'est que  $U \geq 0.8$ 
```

Puis après on peut vérifier par exemple l'espérance avec :  $N=1000$  ou  $10000$

```
s= 0
```

```
for i in range(N) : x=h() ;s=s+x
```

```
print(s/N)
```

on peut vérifier  $p_2$  en comptant le nombre de fois où 2 sort

```
s=0
```

```
for i in range(N):
```

```
    x=h();
```

```
    if x==2:
```

```
        s=s+1
```

```
return(s/N)
```

**exercices** :a) simuler la loi  $p_i = \frac{2^i}{n(n+1)}$  pour i allant de 1 à n puis l'espérance qu'on cherchera

on pourra créer la liste des  $p_i$  et la liste de  $s_i=p_1+p_2+\dots+p_i$  et avec un while chercher le i tel que  $s_i \leq U < s_{i+1}$

b)On peut aussi avec cette méthode simuler des lois discrètes infinies comme une loi de Poisson

on choisit U uniforme dans  $[0 ;1]$  et la valeur sera k lorsque  $P[k-1] \leq u < P[k]$  où  $P[k] = \sum_{i=0}^k \frac{e^{-\lambda} \lambda^i}{i!}$

