Chaîne de Lettres

Sommaire

Chaîne de Lettres

Modèle Informatique

Variables essentielles

Acheteurs potentiels : P_{tot}

Génération : g

Acheteurs à la génération g: $P(g)$

Nombre d'exemplaires de la lettre à la génération g Z_g

$$\mathbb{P}_{\geq 1}(g) = \mathbb{P}_{\geq 1}(0)\frac{P(g)}{P_{tot}}$$

$$\mathbb{P}_{\geq 2}(g) = \mathbb{P}_{\geq 1}(g)^2$$

$$\mathbb{P}_0(g) = 1 - \mathbb{P}_{\geq 1}(g)$$

$$\mathbb{P}_2(g) = \mathbb{P}_{\geq 2}(g)$$

$$\mathbb{P}_1(g) = 1 - \mathbb{P}_2(g) - \mathbb{P}_0(g)$$

Chaîne de Lettres

Explication de Code

```
import random


def sold(p1, p2):
    p0 = 1-p1-p2
    return(random.choices((0,1,2),(p0,p1,p2))[0])
```

On note **p1** et **p2** les probabilités de vendre 1 ou 2 lettres, ce programme renvoie soit 0, soit 1 soit 2 avec les probabilités correspondantes

```
def pop_gen(g_max, p1, p2):
    g = 0
    pop = [[g, -50]]
    indice = 0
    while g < g_max and len(pop)-indice > 0:
        g += 1
        l = len(pop)
        for i in range(indice, l):
            s = sold(p1, p2)
            pop[i][1] += 50*s
            pop += [[g, -50]]*s
        indice = l
    return(pop)
```

La liste **pop** est la liste des participants, on connait leur génération **g** et l'argent qu'ils ont gagné.
La variable **indice** correspond à l'indice dans **pop** du début de la génération en cours de traitement.
On force la simulation à s'arrêter à la génération **g_max**.

Impossible

```
>>> pop_gen(12, .5, .4)
[[0, 50], [1, 100], [1, 100], [2, 100], [2, 100], [2, 50], [3, 50]
```

Il y avait une erreur dans le programme **pop_gen**, car personne ne peut en théorie gagner plus de 50 dollars.

```
>>> pop_gen(12, .5, .4)
[[0, 50], [1, 100], [1, 100], [2, 100], [2, 100], [2, 50], [3, 50]
```

Il y avait une erreur dans le programme **pop_gen**, car personne ne peut en théorie gagner plus de 50 dollars.

```
for i in range(indice, l):
    s = sold(p1, p2)
    pop[i][1] += 50*s
    pop += [[g, -50]]*s
```

Cette notation complète la liste pop avec des sous-listes qui sont dépendantes :

```
>>> t = [[0]]*2

>>> t
[[0], [0]]

>>> t[0][0] = 1

>>> t
[[1], [1]]
```

Lorsqu'une personne vend sa lettre, on veut que elle seule touche les 50 dollars, ce qui n'était pas le cas. Voici la version corrigée :

```
def pop_gen_repaired(g_max, p1, p2):
    g = 0
    pop = [[g, -50]]
    indice = 0
    while g < g_max and len(pop)-indice > 0:
        g += 1
        l = len(pop)
        for i in range(indice, l):
            s = sold(p1, p2)
            pop[i][1] += 50*s
            for i in range(s):
                pop += [[g, -50]]
        indice = l
    return(pop)
```

```
def total_sum(t):
    sum = 0
    for elem in t:
        sum += elem[1]
    return(sum)
```

t contient des couples d'éléments, **total_sum** calcule la somme des deuxièmes arguments.

```
def total_sum(t):
    sum = 0
    for elem in t:
        sum += elem[1]
    return(sum)
```

t contient des couples d'éléments, **total_sum** calcule la somme des deuxièmes arguments.

```
>>> t = pop_gen_repaired(24, .5, .4)

>>> total_sum(t)
-50
```

```
def update_letter(pop, i):
    letter = pop[i][2]
    first = letter[0]
    if first is not None:
        pop[first][1] += 50
    return(letter[1::]+[i])
```

La liste **letter** de 12 indices correspondant aux précédents acheteurs de la lettre

Les 12 noms qui étaient à l'origine sur la liste sont remplacés par **None**

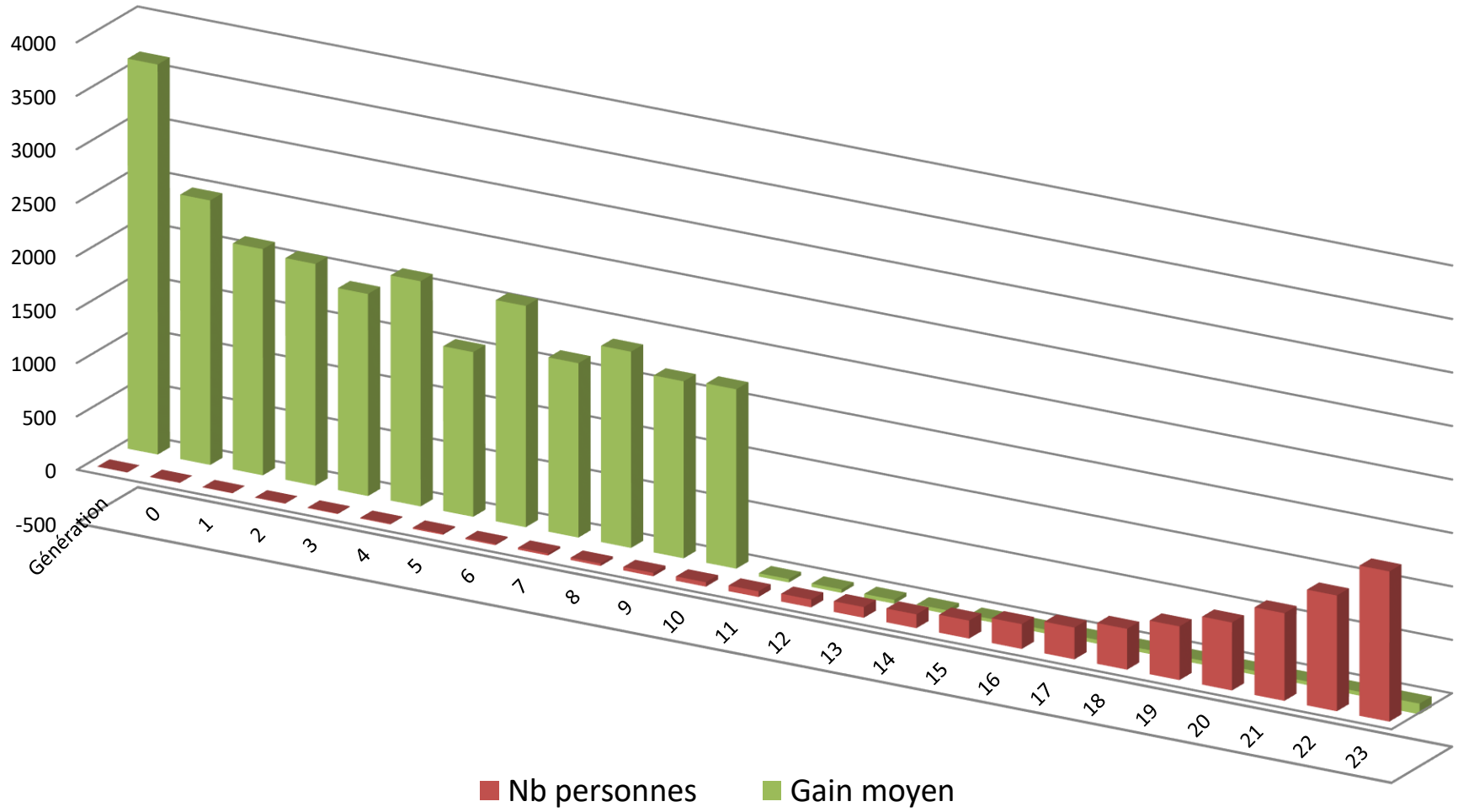
Population totale

```
def pop_gen2(p1, p2, population):
    g = 0
    pop = [[g, -100, [None]*12]]
    indice = 0
    while len(pop)-indice > 0:
        g += 1
        l = len(pop)
        pop_dispo = population-l
        p1_g = p1*(pop_dispo)/population
        p2_g = p2*(pop_dispo)/population
        for i in range(indice,l):
            s = sold(p1_g, p2_g)
            pop[i][1] += 50*s
            for j in range(s):
                pop.extend([[g, -100, update_letter(pop, i)]])
        indice = l
    return(pop)
```

On note **p1_g** et **p2_g** les probabilités à la génération **g** de vendre respectivement 1 et 2 lettres.

Chaîne de Lettres

Exploitation de la Simulation


4000
3500
3000
2500
2000
1500
1000
500
0
-500
Génération
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
Nb personnes
Gain moyen

Compilation

```
def data(t):
    return([elem[:2] for elem in t])
```

```
def data_display(t):
    for elem in t:
        print(elem[0], elem[1])
```

```
def avg(t):
    if t == []:
        return([])
    gen_max = t[-1][0]
    stats = []
    i = 0
    for gen in range(gen_max+1):
        sum = 0
        n = 0
        while i < len(t) and t[i][0] == gen
            n +=1
            sum += t[i][1]
            i += 1
        stats += [[gen, n, sum//n]]
    return(stats)
```

```
def avg_display(t):
    if t == []:
        return([])
    gen_max = t[-1][0]
    i = 0
    for gen in range(gen_max+1):
        sum = 0
        n = 0
        while i < len(t) and t[i][0] == gen:
            n +=1
            sum += t[i][1]
            i += 1
        print(gen, n, sum//n)
```

Les versions **_display** permettent un affichage en colonnes plus facile à lire que les versions standards, qui renvoient un tableau de valeurs, exploitable par d'autres programmes.

Affichage

```
>>> t = pop_gen2(.5, .4, 3000)

>>> avg_display(t)
0 1 -50
1 1 -100
```

```
>>> t = pop_gen2(.5, .4, 3000)

>>> avg_display(t)
0 1 1750
1 1 2350
2 1 3350
3 2 2050
4 3 1683
5 3 2016
6 3 2166
7 5 1150
8 7 678
9 11 318
10 13 250
11 20 102
12 25 48
13 36 1
14 48 -15
```

Gen	Pop	$
15	67	-27
16	83	-32
17	104	-38
18	124	-46
19	131	-54
20	118	-59
21	98	-60
22	79	-56
23	71	-61
24	56	-67
25	38	-68
26	25	-70
27	15	-50
28	15	-70
29	9	-73
30	5	-70
31	3	-17
32	5	-90
33	1	-100

Librairie/Afficher

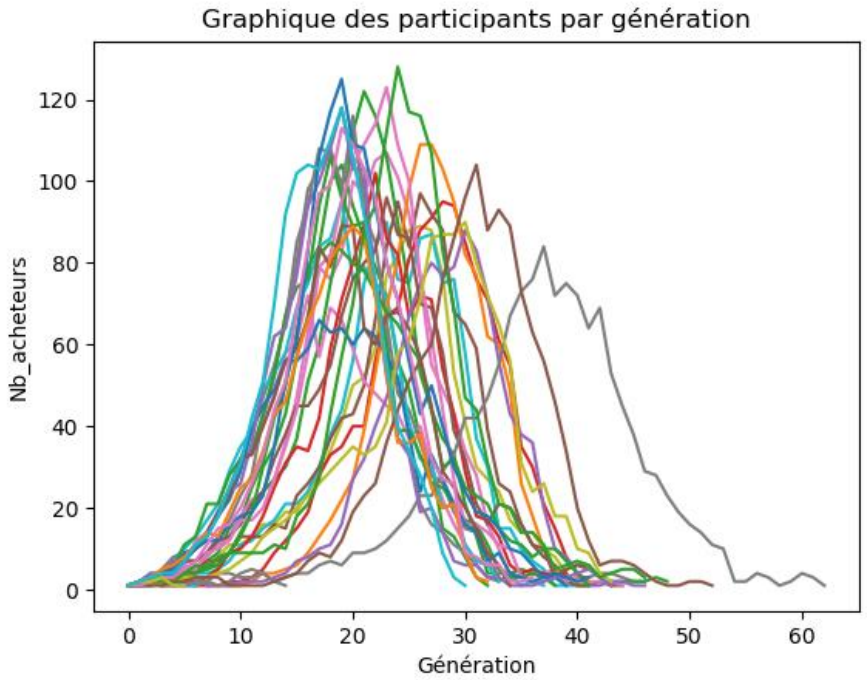
```python
import matplotlib.pyplot as plt

def coord_gen(n, g_max, p1, p2):
    plt = []
    for i in range(n):
        t = pop_gen(g_max, p1, p2)
        x = [elem[0] for elem in avg(t)]
        y1 = [elem[1] for elem in avg(t)]
        y2 = [elem[2] for elem in avg(t)]
        plt += [[x,y1,y2]]
    return(plt)

def coord_gen2(n, p1, p2, population):
    plt = []
    for i in range(n):
        t = pop_gen2(p1, p2, population)
        x = [elem[0] for elem in avg(t)]
        y1 = [elem[1] for elem in avg(t)]
        y2 = [elem[2] for elem in avg(t)]
        plt += [[x,y1,y2]]
    return(plt)
```

```python
def coord_plot(coord, type):
    for elem in coord:
        x = elem[0]
        y1 = elem[1]
        y2 = elem[2]
        if type == 1:
            plt.plot(x,y1)
        else :
            plt.plot(x,y2)

def plot_display(type):
    plt.xlabel("Génération")
    if type == 1:
        plt.ylabel("Nb_acheteurs")
    else:
        plt.ylabel("Gain moyen")
    plt.title("Graphique des gains par génération")
    plt.show()
```

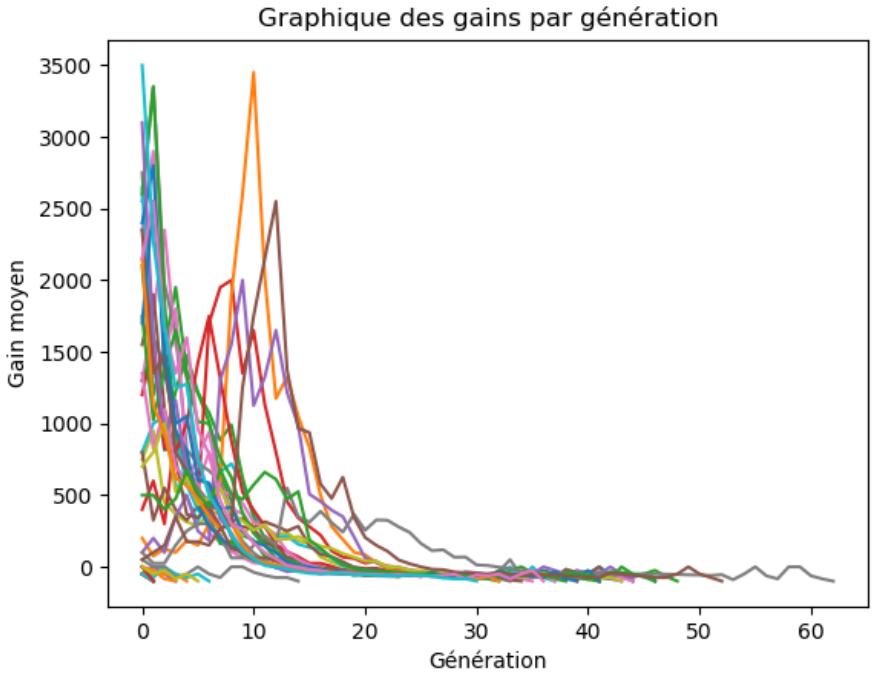
On note **n** le nombre de fois où on lance la simulation

Premières générations

```
>>> coord = coord_gen2(50, .5, .4, 3000)
>>> coord_plot(coord, 1)
>>> plot_display(1)
```




```
>>> coord_plot(coord, 2)
>>> plot_display(2)
```



Une génération

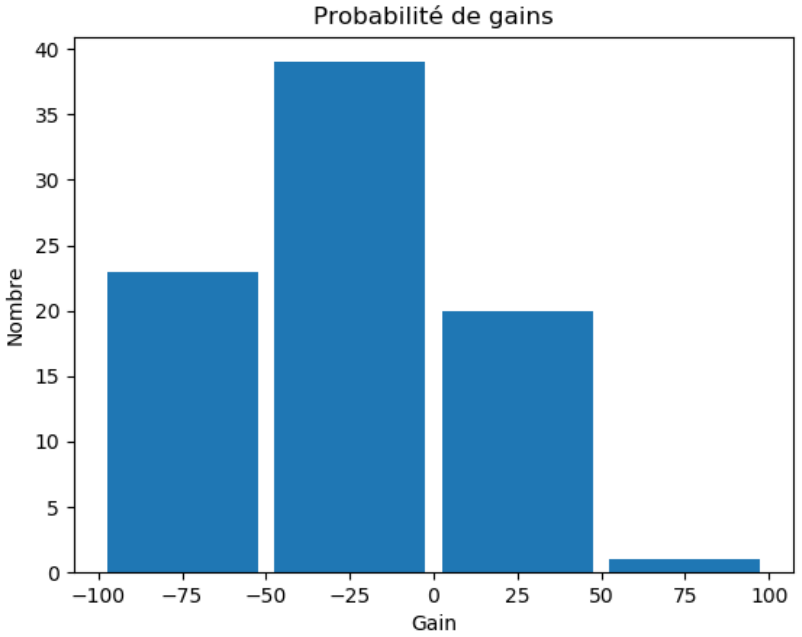
```
def histo_data(p1, p2, population):
    t = pop_gen2(p1, p2, population)
    g_max = t[-1][0]
    h = [[elem[1] for elem in data(t) if
elem[0] == g] for g in range(g_max+1)]
    return(h)

def histo_bar(h, g):
    if len(h) <= g:
        dat = [-2]
    else:
        dat = h[g]
    categories = [50*i for i in range(-2,
max(dat)//50)]
    plt.hist(dat, categories, rwidth = 0.9)

def histo_display():
    plt.xlabel("Gain")
    plt.ylabel("Nombre")
    plt.title("Probabilité de gains")
    plt.show()
```

```
>>> h = histo_data(.5, .4, 3000)

>>> histo_bar(h, 17)

>>> histo_display()
```

Une simulation

```
def histo_data(p1, p2, population):
    t = pop_gen2(p1, p2, population)
    g_max = t[-1][0]
    h = [[elem[1] for elem in data(t) if
elem[0] == g] for g in range(g_max+1)]
    return(h)

def histo_bar(h, g):
    if len(h) <= g:
        dat = [-2]
    else:
        dat = h[g]
    categories = [50*i for i in range(-2,
max(dat)//50)]
    plt.hist(dat, categories, rwidth = 0.9)

def histo_display():
    plt.xlabel("Gain")
    plt.ylabel("Nombre")
    plt.title("Probabilité de gains")
    plt.show()
```

```
>>> h = histo_data(.5, .4, 3000)

>>> histo_bar(h, 17)

>>> histo_display()
```



Les données de la génération **17**
pour une simulation de paramètres
p1 = .5, **p2 = .4**, **pop = 3000**

Loi faible

```
def prob_bar(n, p1, p2, population, g):
    dat = []
    for i in range(n):
        h = histo_data(p1, p2, population)
        if len(h) > g:
            dat += h[g]
    categories = [50*i for i in range (-2,
max(dat)//50)]
    plt.hist(dat, categories, rwidth = 0.9)
```

```
>>> prob_bar(500, .5, .4, 3000, 17)

>>> histo_display()
```

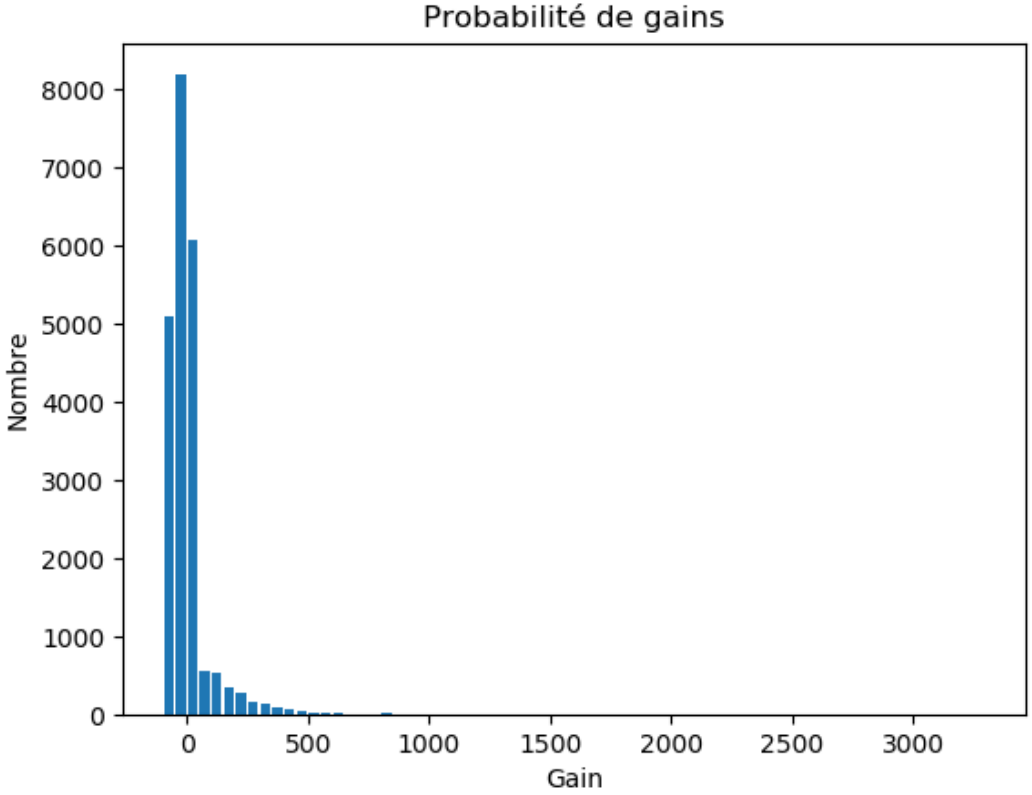
On moyenne les données de la génération **17** sur **500** simulations pour les paramètres
p1 = .5, **p2 = .4**, **pop = 3000**

Probabilités théoriques

```
def prob_bar(n, p1, p2, population, g):
    dat = []
    for i in range(n):
        h = histo_data(p1, p2, population)
        if len(h) > g:
            dat += h[g]
    categories = [50*i for i in range (-2,
max(dat)//50)]
    plt.hist(dat, categories, rwidth = 0.9)
```

```
>>> prob_bar(500, .5, .4, 3000, 17)

>>> histo_display()
```

On moyenne les données de la génération **17** sur **500** simulations pour les paramètres
p1 = .5, **p2 = .4**, **pop = 3000**

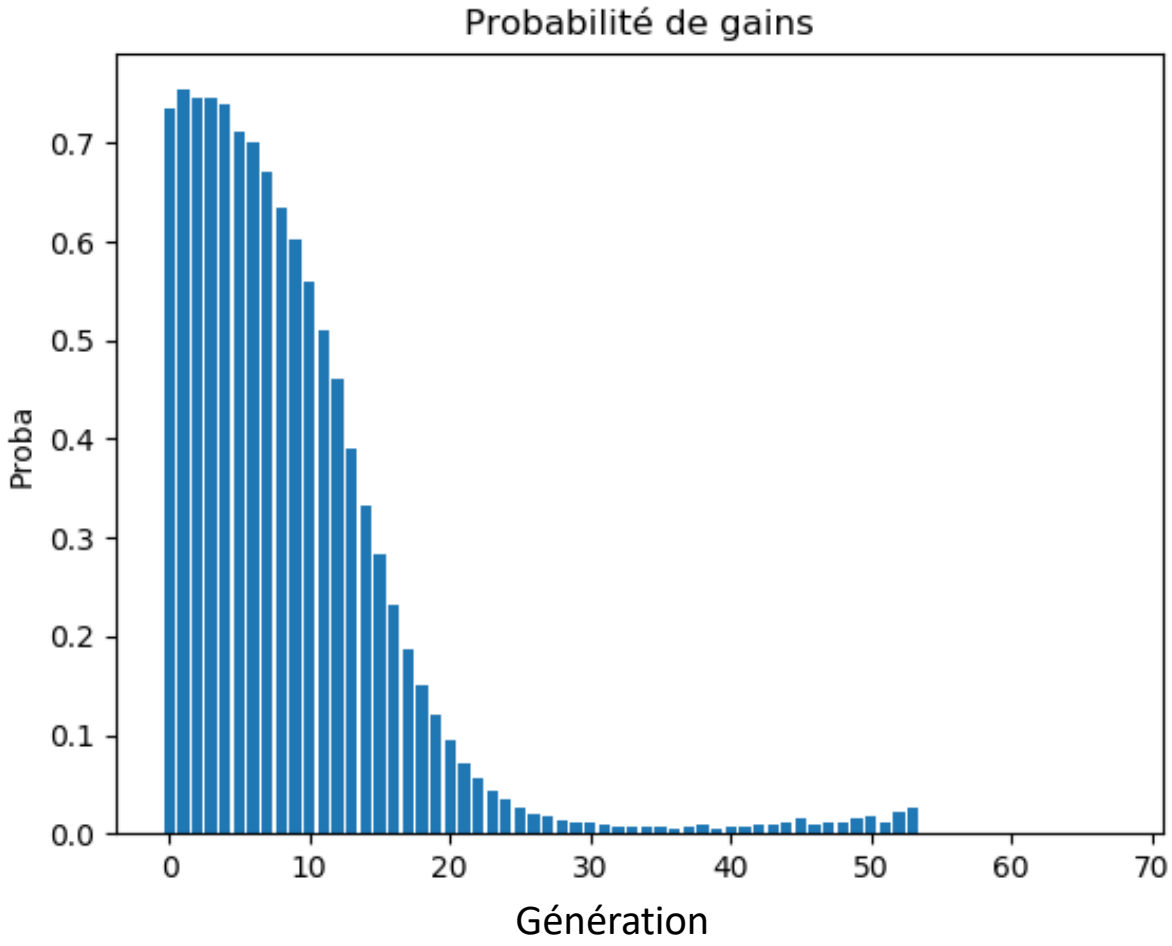


Gain par génération

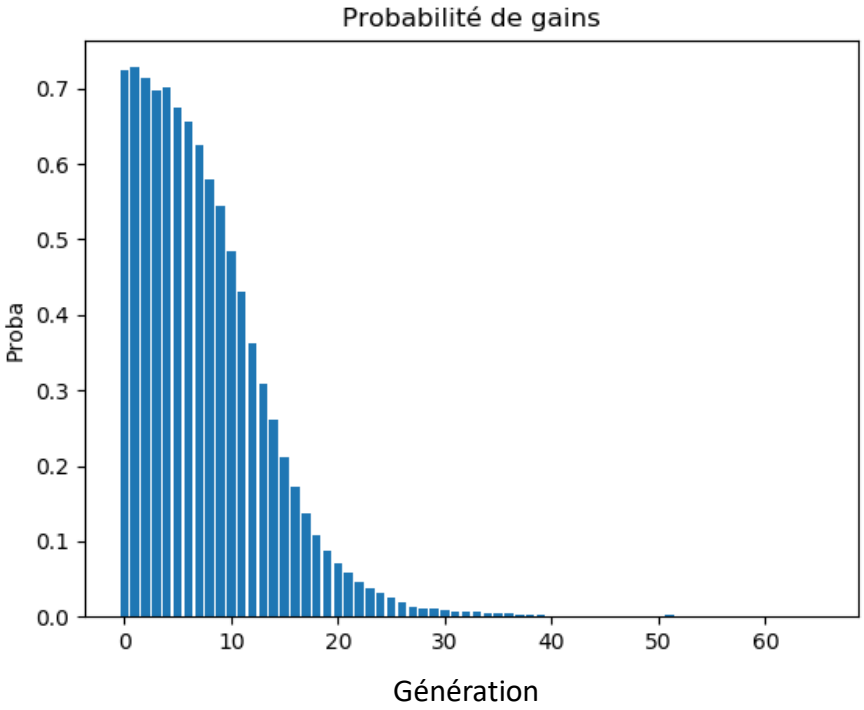
```
def prob_gain(n, cap, p1, p2, population):
    dat = [[]]
    for i in range(n):
        h = histo_data(p1, p2, population)
        diff = len(h)-len(dat)
        if diff > 0:
            for j in range(diff):
                dat += [[]]
        for g in range(len(h)):
            p = len([elem for elem in h[g] if elem >
cap])/len(h[g])
            dat[g] += [p]
    for i in range(len(dat)):
        dat[i] = sum(dat[i])/len(dat[i])
    return(dat)

def bar_display(dat):
    plt.bar([i for i in range(len(dat))], dat)
    plt.xlabel("Gain")
    plt.ylabel("Proba")
    plt.title("Probabilité de gains")
    plt.show()
```

On note **cap** le seuil de gain au dessus duquel on considère un succès

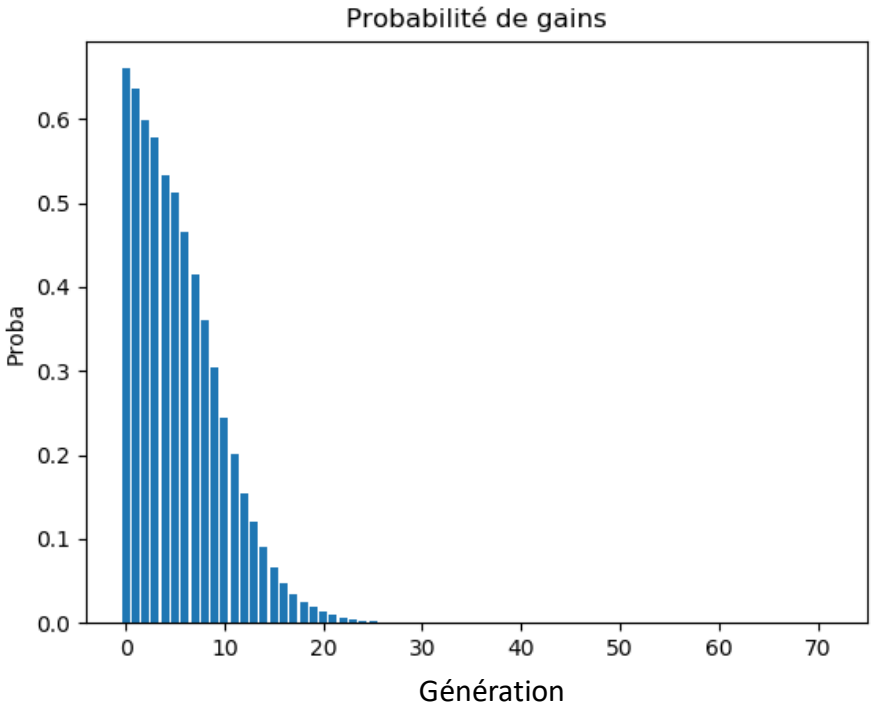
```
>>> dat = prob_gain(500, 0, .5, .4, 3000)

>>> bar_display(dat)
```



La probabilité de gagner plus que **0$** en fonction de la génération, moyennée sur **500** simulations



Pour un gain strictement supérieur à **100$**



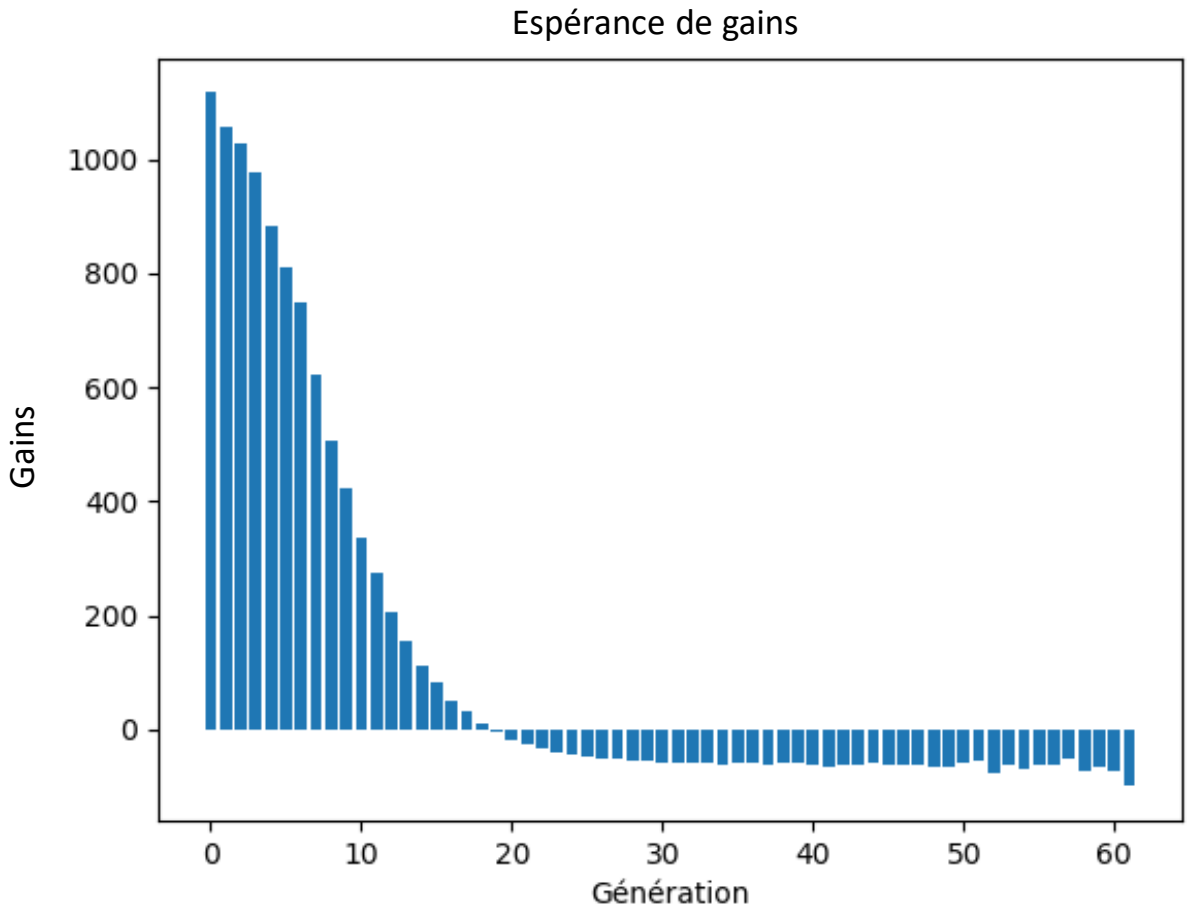
Pour un gain strictement supérieur à **500$**

Espérance

```
def expect_val(n, p1, p2, population):
    dat = [[]]
    for i in range(n):
        h = histo_data(p1, p2, population)
        diff = len(h)-len(dat)
        if diff > 0:
            for j in range(diff):
                dat += [[]]
        for g in range(len(h)):
            e = sum(h[g])/len(h[g])
            dat[g] += [e]
    for i in range(len(dat)):
        dat[i] = sum(dat[i])/len(dat[i])
    return(dat)

def val_display(dat):
    plt.bar([i for i in range(len(dat))], dat)
    plt.xlabel("Génération")
    plt.ylabel("Gain")
    plt.title("Espérance de gain")
    plt.show()
```

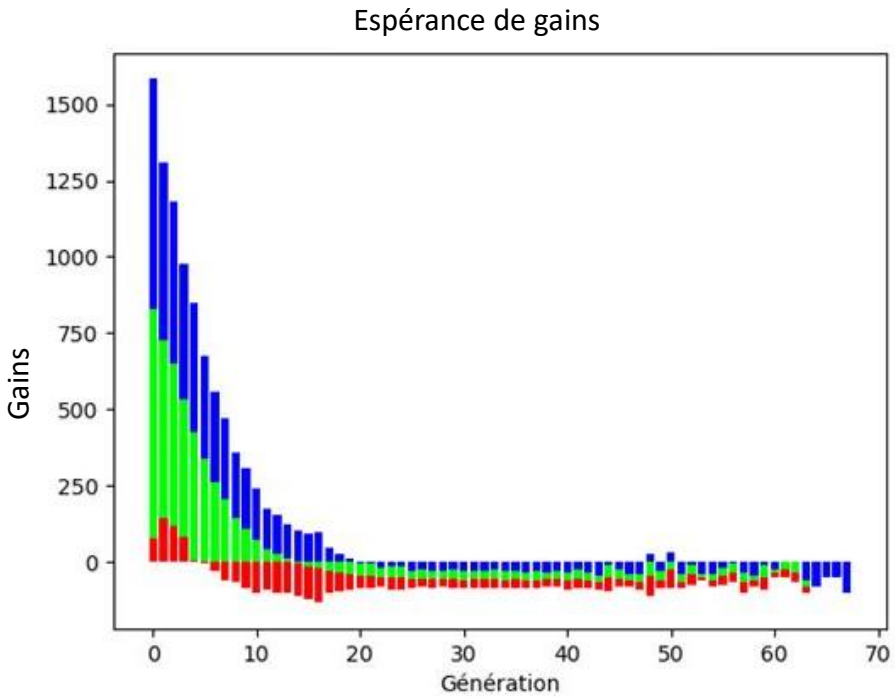
Moyenne

```
>>> dat = expect_val(500, .5, .4, 3000)
>>> val_display(dat)
```

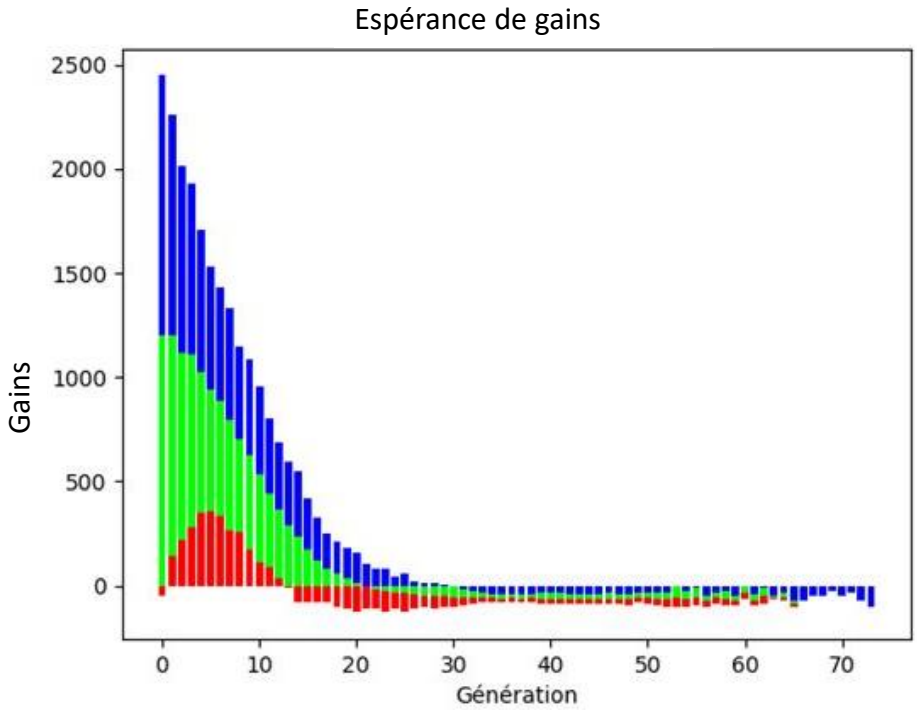


L'espérance de gains moyennée sur **500** simulations

Écart type



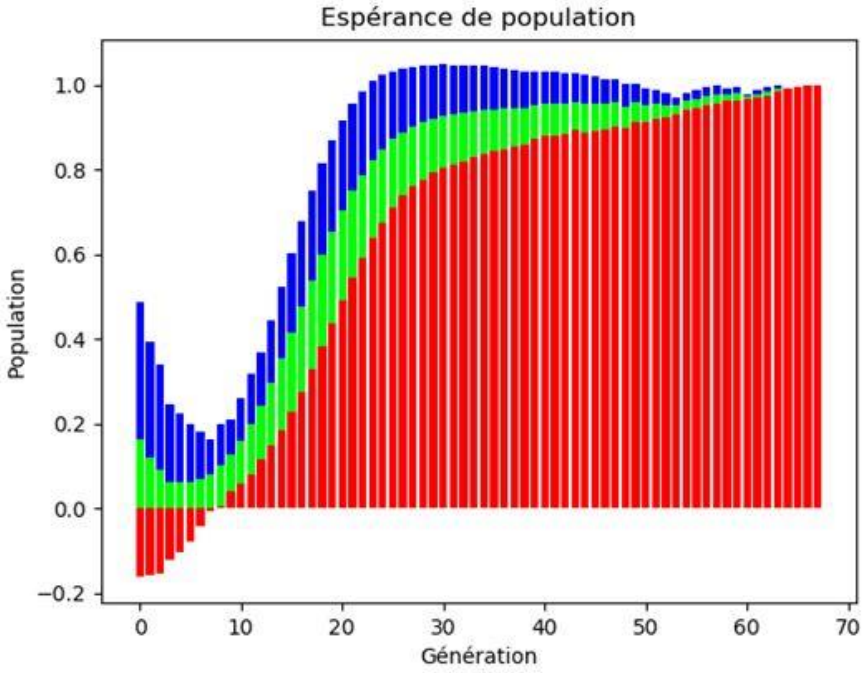
1000 personnes



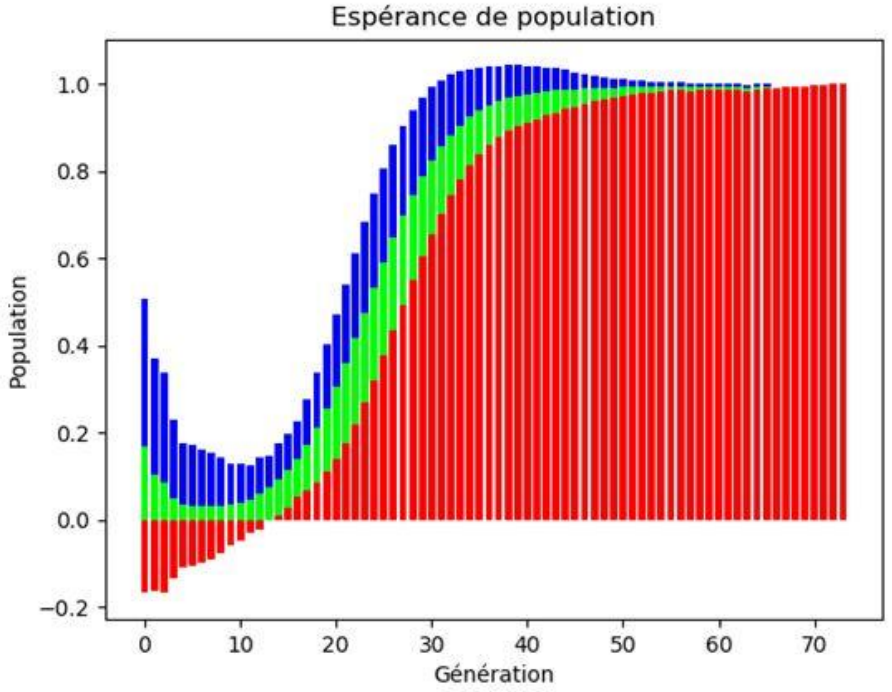
5000 personnes

Proportion

On fait apparaitre l'écart type sur les courbes de population



1000 personnes



5000 personnes

On obtient un seuil $\frac{k}{N}$ de 0.36

1. L'évolution de la simulation pourrait se faire en fonction du temps plutôt que de la génération.
2. Il y a un « effet de mode » initial et on peut simuler un taux de confiance de la population en fonction des gains et des pertes des premiers participants.
3. On peut permettre aux participants d'acheter des lettres plusieurs fois, où d'acheter plusieurs lettres à la fois.
4. Les probabilités de ventes sont inhomogènes car certains habitants connaissent plus de monde.
5. Tenir compte de l'augmentation du nombre de participants potentiels en modélisant la façon dont la lettre peut atteindre de nouvelles villes.

Chaîne de Lettres

Processus de Branchement

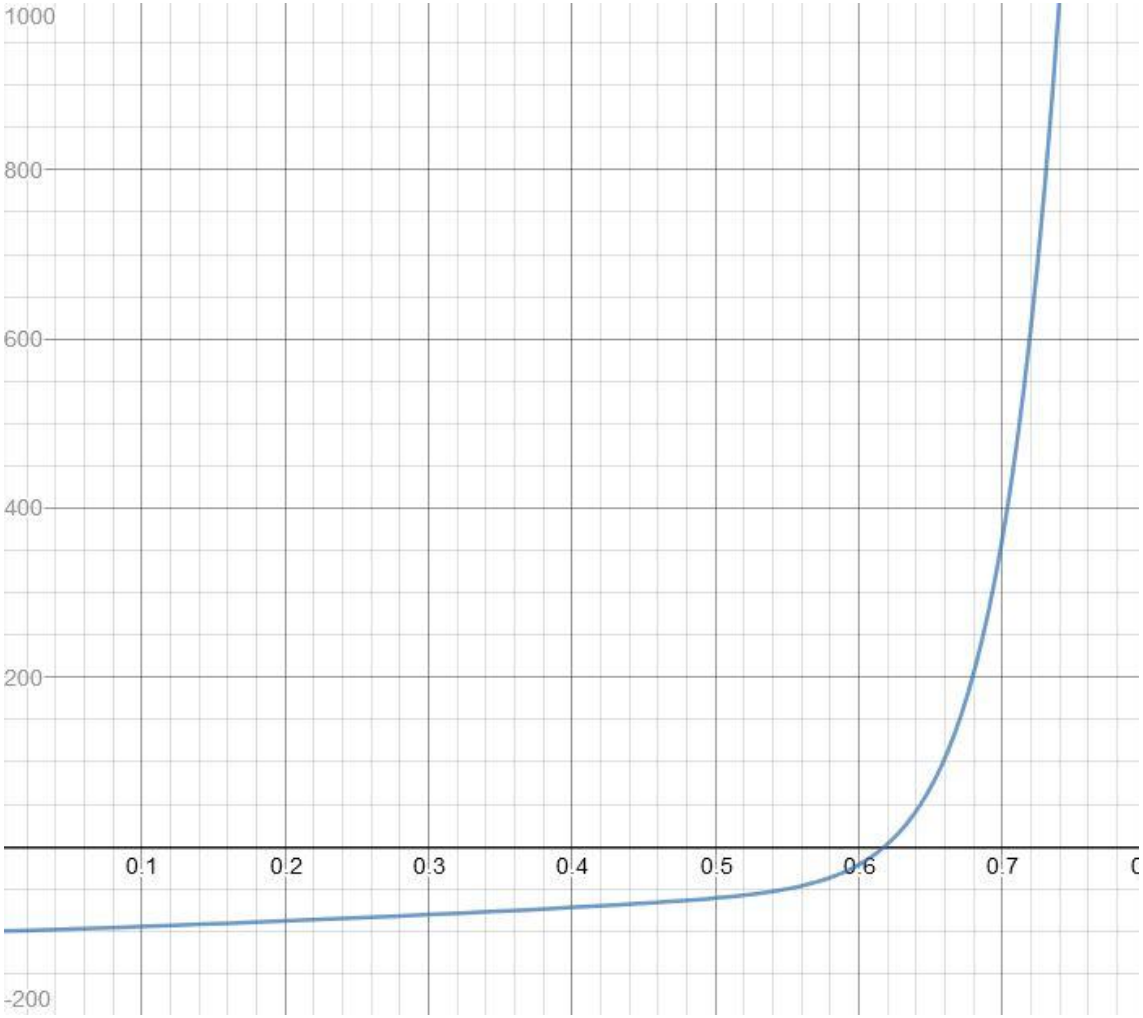
Approximation à l'origine :

$$P(g) \sim P_{tot}$$
$$\mathbb{P}_1(g) = \mathbb{P}_1(0) \equiv p_1$$
$$\mathbb{E} = \mathbb{E}(Z_1) + \mathbb{E}(Z_{12}) - 100 = 50(m + m^{12} - 2)$$
$$m = \mathbb{P}_1(g) + 2 * \mathbb{P}_2(g) = \mathbb{P}_{\geq 1}(g) + \mathbb{P}_{\geq 2}(g) = p_1(1 + p_1)$$

Condition de gain :

$$\mathbb{E} > 0 \iff m > 1 \iff p_1 > \frac{\sqrt{5} - 1}{2} \sim 0.61$$

$$\mathbb{E} = f(p_1)$$



Espérance à l'origine en fonction de la probabilité de vendre une lettre

Probabilités extrémales

Perte maximale : -100
Probabilité : $1 - p_1$

Gain maximal : $204\,800$
Probabilité : $p_1^{2*(2^{11}-1)}$

p_1	Probabilité de perte maximale	Probabilité de gain maximal	Espérance
0.25	0.75	10^{-2465}	-84.4
0.5	0.5	10^{-1233}	-60.9
0.62	0.38	10^{-850}	2.9
0.75	0.25	10^{-512}	1272.3
0.9	0.1	10^{-188}	31 240
0.99	0.01	10^{-18}	170 933

Fonction génératrice des probabilités :

$$F_X(t) = \mathbb{E}(t^X) = \sum_{k=0}^{+\infty} \mathbb{P}(X = k)t^k$$

Propriétés :

$$F_X(0) = \mathbb{P}(X = 0)$$

$$F_X^{(k)}(1) = \mu_k = \mathbb{E}\big(X(X-1)\dots(X-k+1)\big)$$

$(X_k)_{k\leq n}$ mutuellement indépendantes identiquement distribuées (VAIID)

$$S_n = \sum_{k=1}^{n} X_k$$

$$F_{S_n}(t) = \left(F_{X_1}(t)\right)^n$$

Lemme :

Si (Z_n) est un processus de branchement alors on a pour $n \in \mathbb{N}$

$$F_{Z_n}(t) = F_{Z_1}^{\circ(n)}(t)$$

Démonstration

Lemme :

Si (Z_n) est un processus de branchement alors on a pour $n \in \mathbb{N}$

$$F_{Z_n}(t) = F_{Z_1}^{\circ(n)}(t)$$

Démonstration :

$$F_{Z_n}(t) = \sum_{k=0}^{+\infty} \mathbb{P}(Z_n = k)t^k = \sum_{k=0}^{+\infty} t^k \sum_{l=0}^{k} \mathbb{P}(Z_n = k|Z_{n-1} = l)\mathbb{P}(Z_{n-1} = l)$$

$$= \sum_{l=0}^{+\infty} \mathbb{P}(Z_{n-1} = l) \sum_{k=l}^{+\infty} \mathbb{P}(Z_n = k|Z_{n-1} = l)t^k = \sum_{l=0}^{+\infty} \mathbb{P}(Z_{n-1} = l)F_{Z_n|Z_{n-1}=l}(t)$$

$$F_{Z_n|Z_{n-1}=l}(t) = \left(F_{Z_1}(t)\right)^l \Rightarrow F_{Z_n}(t) = \sum_{l=0}^{+\infty} \mathbb{P}(Z_{n-1} = l)\left(F_{Z_1}(t)\right)^l = F_{Z_{n-1}}\left(F_{Z_1}(t)\right)$$

Calcul de l'itéré de $F_{Z_1} = p_2 t^2 + p_1 t + (1 - p_1 - p_2)$

Calcul explicite :

```
def Fn(p1,p2,n,x):
    p0 = 1-p1-p2
    a = x
    for i in range(n):
        a = p0+p1*a+p2*a**2
    return(a)
```

Calcul de l'itéré de $F_{Z_1} = p_2 t^2 + p_1 t + (1 - p_1 - p_2)$

Calcul explicite de $F_{Z_1}^{\circ 3}(0)$ pour **p1=0.4, p2=0.5**

```
>>> Fn(.4,.5,3,0)
0.16851249999999995
```

Calcul de l'itéré de $F_{Z_1} = p_2 t^2 + p_1 t + (1 - p_1 - p_2)$

Calcul formel :

```
from sympy import symbols, simplify

p1 = symbols('p1', float = True)
p2 = symbols('p2', float = True)

F = [1-p1-p2,p1,p2]

def Somme(P,Q):
    S = []
    m = min(len(P), len(Q))
    for  i in range(m):
        S.append(P[i]+Q[i])
    S.extend(P[m:])
    S.extend(Q[m:])
    return(S)

def Produit(P,Q):
    S = [0]*(len(P)+len(Q)-1)
    for i, a in enumerate(P) :
        for j, b in enumerate(Q):
            S[i+j] = S[i+j] + a*b
    return(S)
```

```
def Puissances(P,n):
    Pn = [[1]]
    for i in range(n):
        Pn.append(Produit(Pn[-1],P))
    return(Pn)

def Compose(P,Q):
    PQ = []
    for a, Qi in zip(P, Puissances(Q, len(P)-1)):
        PQ = Somme(PQ, Produit([a], Qi))
    return(PQ)

def Iteres(P,n):
    L = [[0,1]]
    for i in range(n):
        L.append(Compose(L[-1],P))
    return(L)
```

Calcul de l'itéré de $F_{Z_1} = p_2 t^2 + p_1 t + (1 - p_1 - p_2)$

Calcul des coefficients de $F_{Z_1}^{\circ 3}$ pour **p1=0.4, p2=0.5**

```
>>> P = [0.1,0.4,0.5]

>>> [round(e,10) for e in Iteres(P,3)[-1]]
[0.1685125, 0.109, 0.19985, 0.175, 0.162575, 0.091, 0.06125, 0.025,
0.0078125]
```

Calcul de l'itéré de $F_{Z_1} = p_2 t^2 + p_1 t + (1 - p_1 - p_2)$

Calcul formel des coefficients de $F_{Z_1}^{\circ 2}$

```
>>> F = [1-p1-p2,p1,p2]

>>> [simplify(e) for e in Iteres(F,2)[-1]]
[-p1**2 - p1*p2 + p2*(p1 + p2 - 1)**2 - p2 + 1, p1*(p1 - 2*p2*(p1 +
p2 - 1)), p2*(p1**2 + p1 - 2*p2*(p1 + p2 - 1)), 2*p1*p2**2, p2**3]
```

Calcul des probabilités :

$$F_{Z_n}(X) = \sum_{k=0}^{+\infty} \mathbb{P}(Z_n = k) X^k$$

$\mathbb{P}(Z_n = k)$ vaut

```
>>> simplify(Iteres(F,"n")[-1]["k"])
```

Probabilité d'extinction :

$F_{Z_n}(0) = \mathbb{P}(Z_n = 0)$ qui vaut pour $n = 3$

```
>>> simplify(Iteres(F,3)[-1][0])
-2*p1*p2**2*(p1 + p2 - 1)**3 - p1*(p1 - 2*p2*(p1 + p2 - 1))*(p1 + p2
- 1) - p1*(p1 + p2 - 1) - p1 + p2**3*(p1 + p2 - 1)**4 + p2*(p1 + p2
- 1)**2*(p1**2 + p1 - 2*p2*(p1 + p2 - 1)) + p2*(p1 + p2 - 1)**2 - p2
+ 1
```

Probabilité d'extinction :

$\left(F_{Z_n}(0)\right)_n$ croissante et majorée donc $F_{Z_n}(0) \xrightarrow[n\to+\infty]{} l$

$F_{Z_1}(l) = l, \qquad l' = \min_{[0,1]}\{x, \qquad F_{Z_1}(x) = x\}$

F_{Z_1} croissante convexe

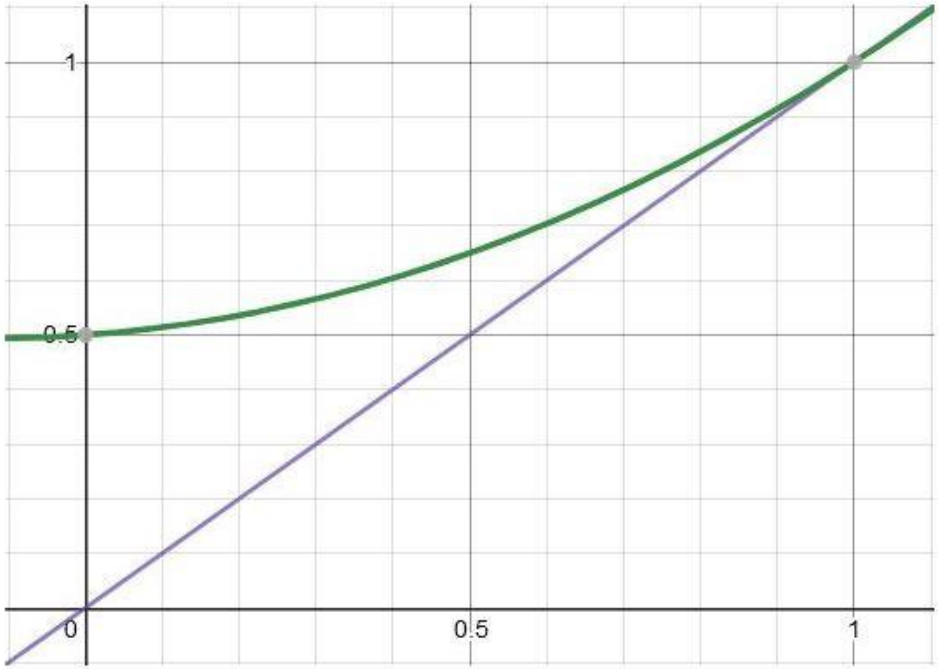
$0 \leq l' \Rightarrow F_{Z_n}(0) \leq l' \Rightarrow l \leq l' \Rightarrow l = l'$

Point fixe

$$\mathbb{E}(Z_1) = m, \qquad \mathbb{V}(Z_1) = \sigma^2$$

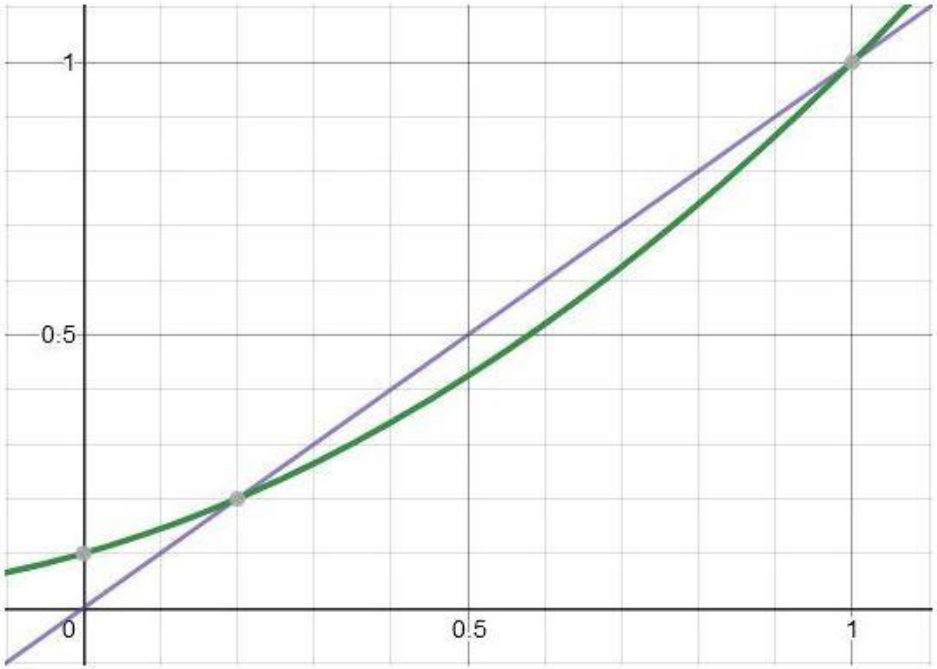
$F'_{Z_1}(1) = m \leq 1$, F'_{Z_1} croissante

$\Rightarrow l = 1$



$F'_{Z_1}(1) = m > 1, \qquad F_{Z_1}(0) = p_0 > 0$

$\Rightarrow l < 1$



Espérance :

$$\mathbb{E}(Z_n) = F'_{Z_n}(1) = \left(F_{Z_1}^{\circ n}\right)'(1) = \prod_{k=0}^{n-1}\left(F'_{Z_1} \circ F_{Z_1}^{\circ k}\right)(1) \underbrace{=}_{F_{Z_1}(1)=1} \left(F'_{Z_1}(1)\right)^n = m^n$$

$$\mathbb{E}(Z_n) = m^n$$

Variance :

$$\mathbb{V}(Z_n) \underset{\substack{m\neq 1\\ m=1}}{=} \begin{cases} m^{n-1}\sigma^2\left(\dfrac{1-m^n}{1-m}\right) \\ \sigma^2 n \end{cases}$$

Démonstration

Variance :

$$\mathbb{V}(Z_n) \underset{\substack{m\neq 1\\ m=1}}{=} \begin{cases} m^{n-1}\sigma^2\left(\dfrac{1-m^n}{1-m}\right) \\ \sigma^2 n \end{cases}$$

Démonstration :

$$\mathbb{V}(Z_n) = F''_{Z_n}(1) + \mathbb{E}(Z_n) - \mathbb{E}(Z_n)^2$$

$$F''_{Z_n}(1) = \sum_{i=0}^{n-1} \left(F'_{Z_1} \circ F^{oi}_{Z_1}\right)'(1) \prod_{\substack{k=0\\ k\neq i}}^{n-1} F'_{Z_1} \circ F^{ok}_{Z_1}(1)$$

$$= m^{n-1} \sum_{i=0}^{n-1} \left(\prod_{l=0}^{i-1} \left(F'_{Z_1} \circ F^{ol}_{Z_1}\right)(1) \right) \left(F''_{Z_1} \circ F^{oi}_{Z_1}(1)\right) = m^{n-1}(\sigma^2 + m - m^2) \sum_{i=0}^{n-1} m^i$$

$$\underset{\substack{m\neq 1\\ m=1}}{=} \begin{cases} m^{n-1}(\sigma^2 + m - m^2)\left(\dfrac{1-m^n}{1-m}\right) = \begin{cases} m^{n-1}\sigma^2\left(\frac{1-m^n}{1-m}\right) + m^n - m^{2n} \\ \sigma^2 n \end{cases} \\ m^{n-1}(\sigma^2 + m - m^2)n \end{cases}$$

Lemme :

1) Si $m > 1$ alors $\mathbb{P}(Z_n > 0) \underset{+\infty}{\longrightarrow} 1 - l$

2) Si $m < 1$ alors $\mathbb{P}(Z_n > 0) \sim Cm^n$

3) Si $m = 1$ alors $\mathbb{P}(Z_n > 0) \sim \frac{2}{\sigma^2 n}$

Lemme :

1) Si $m > 1$ alors $\mathbb{P}(Z_n > 0) \underset{+\infty}{\longrightarrow} 1 - l$

2) Si $m < 1$ alors $\mathbb{P}(Z_n > 0) \sim Cm^n$

3) Si $m = 1$ alors $\mathbb{P}(Z_n > 0) \sim \frac{2}{\sigma^2 n}$

Démonstration :

2)

$$\mathbb{P}(Z_n > 0) \leq \sum_{k=0}^{+\infty} \mathbb{P}(Z_n > 0) = \mathbb{E}(Z_n) = m^n$$

$$\mathbb{P}(Z_{n+1} > 0) = \frac{1 - F_{Z_1}\left(F_{Z_n}(0)\right)}{1 - F_{Z_n}(0)} \mathbb{P}(Z_n > 0)$$

$$\frac{1 - F_{Z_1}\left(F_{Z_n}(0)\right)}{1 - F_{Z_n}(0)} \underbrace{=}_{TAF} F'_{Z_1}(c) \underbrace{\leq}_{convexité} F'_{Z_1}(1) = m$$

$$\left(\frac{\mathbb{P}(Z_n > 0)}{m^n}\right)_n \underset{n \to +\infty}{\longrightarrow} C > 0$$

3)

$$F''_{Z_1}(1) = \sigma^2 \text{ car } m = 1$$

$$F_{Z_1}(1 - x) \underset{0}{=} F_{Z_1}(1) - F'_{Z_1}(1)x + F''_{Z_1}(1)\frac{x^2}{2} + o(x^2)$$

$$\underset{0}{=} 1 - x + \sigma^2 \frac{x^2}{2} + o(x^2)$$

$$u_n = \mathbb{P}(Z_n > 0), \qquad u_{n+1} = 1 - F_{Z_1}(1 - u_n)$$

$$\frac{1}{u_{n+1}} - \frac{1}{u_n} = \frac{1}{u_n}\left(\frac{1}{1 - \frac{\sigma^2}{2}u_n + o(u_n)} - 1\right) = \frac{1}{u_n}\left(\frac{\sigma^2}{2}u_n + o(u_n)\right) \underset{n \to +\infty}{\longrightarrow} \frac{\sigma^2}{2}$$

$$\frac{1}{u_n} = \frac{1}{u_0} + \sum_{k=0}^{n-1}\left(\frac{1}{u_{k+1}} - \frac{1}{u_k}\right) \sim \frac{\sigma^2 n}{2}$$

$$u_n \sim \frac{2}{\sigma^2 n}$$

Temps d'arrêt :

$$K = \min\{k, \quad Z_k = 0\}$$

Espérance dans le cas sous-critique :

$$\mathbb{E}(K) = \sum_{k=1}^{+\infty} \mathbb{P}(Z_k > 0) = \frac{C}{1 - \mathbb{E}(Z_1)}$$

Chaîne de Lettres

Vente pyramidale Avec quota

Variables essentielles

Coût d'entrée : $\boldsymbol{c}$

Quota : $\boldsymbol{N}$

Nombre de personnes recrutées : $\boldsymbol{R}$

Gain par recrutement : $\boldsymbol{d}$

Rang d'entrée : $\boldsymbol{k}$

Condition :

$$\mathbb{E}(R)d > c$$

$$\mathbb{E}(R) > \frac{c}{d}$$

$$X_i \sim \mathcal{B}\left(\frac{1}{i}\right)$$

$$R = \sum_{i=k}^{N-1} X_i$$

$$\mathbb{E}(R) \underset{N\to+\infty}{\sim} \ln\left(\frac{N}{k}\right)$$

Ne dépend que du quotient $\frac{N}{k}$

Condition :

$$\frac{N}{k} > \exp\left(\frac{c}{d}\right)$$

Seuil :

$$\frac{k}{N} < exp\left(-\frac{c}{d}\right) \underbrace{=}_{c=d} e^{-1} \cong 0.368$$

Espérance à l'origine :

$$\mathbb{E} = d\mathbb{E}(R) - c$$

$$\mathbb{E} \underset{N\to+\infty}{\sim} d\ln(N)$$

Probabilité de perte totale :

$$\mathbb{P}_k(0) = \prod_{i=k}^{N-1}(1 - 1/i) = \frac{(k-1)}{(N-1)}$$

Nombre de participants qui perdent tout :

$$\sum_{k=1}^{N} \mathbb{P}_k(0) = \frac{1}{N-1}(1 + \cdots + N - 1) = \frac{N}{2}$$

Lemme :

La proportion de participants qui recrutent exactement r personnes tend vers $2^{-(r+1)}$ quand $N \to +\infty$

Lemme :

La proportion de participants qui recrutent exactement r personnes tend vers $2^{-(r+1)}$ quand $N \to +\infty$

Démonstration :

$$X_i \sim \mathcal{B}(p_i)$$

$$\forall x, \quad \mathbb{P}(Y_i \geq x) \geq \mathbb{P}(X_i \geq x)$$

Contrainte :

$$\mathbb{P}(Y_i = 0) \leq \mathbb{P}(X_i = 0) = 1 - p_i$$

Définition :

$$Y_i \sim \mathcal{P}(\lambda_i)$$

$$e^{-\lambda_i} = 1 - p_i \Rightarrow \lambda_i = -ln(1 - p_i)$$

Modèle :

$$p_i = \frac{1}{i}$$

$$S_k = \sum_{i=k}^{N-1} X_i \preccurlyeq T_k = \sum_{i=k}^{N-1} Y_i$$

Propriété :

$$T_k \sim \mathcal{P}(\gamma_k)$$

$$\gamma_k = \sum_{i=k}^{N-1} \lambda_i = \sum_{i=k}^{N-1} (ln(i) - ln(i-1)) = ln\left(\frac{N-1}{k-1}\right)$$

Montrons que :

$$\frac{1}{N-2} \sum_{k=2}^{N-1} \mathbb{P}(T_k = r) \xrightarrow[N \to +\infty]{} 2^{-(r+1)}$$

$$\frac{1}{N-2} \sum_{k=2}^{N-1} \mathbb{P}(T_k = r) = \frac{1}{N-2} \sum_{k=2}^{N-1} e^{-\gamma_k} \frac{\gamma_k^r}{r!}$$

$$= \frac{1}{r!\,(N-2)} \sum_{k=2}^{N-1} \frac{k-1}{N-1} \left(ln\left(\frac{N-1}{k-1}\right)\right)^r$$

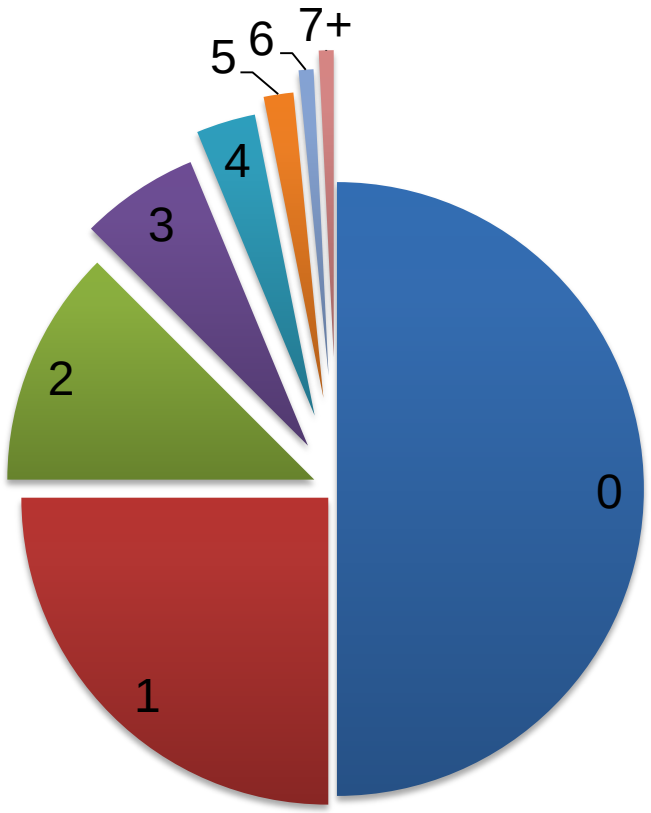
La somme de Riemann tend vers

$$\frac{1}{r!} \int_0^1 v \left(ln\left(\frac{1}{v}\right)\right)^r dv \underset{z = ln\left(\frac{1}{v}\right)}{=} \frac{1}{r!} \int_0^{+\infty} e^{-2z} z^r dz = 2^{-(r+1)}$$

Lemme :

La proportion de participants qui recrutent exactement r personnes tend vers $2^{-(r+1)}$ quand $N \to +\infty$

Proportion de participants en fonction du nombre de recrues



Chaîne de Lettres

Chaîne de lettres Avec plusieurs Participations

Variables essentielles

Quota : N

Nombre de participants : k

Participants de chaque type : (U_k, V_k, W_k)

Temps d'arrêt de la chaîne : K

$$(U_0, V_0, W_0) = (u_0, 0, 0)$$

Vente d'une lettre :

$$U_{k+1} - U_k = W_{k+1} - W_k = \begin{cases} 1 \\ 0 \end{cases}$$

$$\Rightarrow \sum_{k=0}^{n-1} (U_{k+1} - U_k) = \sum_{k=0}^{n-1} (W_{k+1} - W_k)$$

$$\Rightarrow W_n = U_n - U_0 + W_0 = U_n - u_0$$

Achat d'une lettre :

$$U_{k+1} + V_{k+1} + W_{k+1} = U_k + V_k + W_k + 1$$

$$\Rightarrow U_n + V_n + W_n = U_0 + V_0 + W_0 + n$$

$$\Rightarrow V_n = n + u_0 - U_n - W_n = n + 2u_0 - 2U_n$$

Répartition selon le nombre de lettres vendues :

$$(U_k, V_k, W_k) = (U_k, k + 2u_0 - 2U_k, U_k - u_0)$$

Règle de fonctionnement de la chaîne de Markov :

$$\mathbb{P}(U_{k+1} = u | U_k = u) = 1 - \mathbb{P}(U_{k+1} = u + 1 | U_k = u)$$

$$= 1 - \frac{V_k}{U_k + V_k} = \frac{u}{k + 2u_0 - u}$$

Quota :

$$U_k + V_k < N \Rightarrow k + 2u_0 - U_k < N$$

$$\Rightarrow U_k > 2u_0 + k - N$$

$$V_k \geq 0 \Rightarrow k + 2u_0 - 2U_k \geq 0$$

$$\Rightarrow U_k \leq u_0 + \frac{k}{2}$$

Encadrement :

$$2u_0 + k - N < U_k \leq u_0 + \frac{k}{2}$$

$$2u_0 + k - N < u_0 + \frac{k}{2} \Rightarrow k < 2N - 2u_0$$

Effondrement :

$$k < 2N - 2u_0$$

Temps d'arrêt :

$$K = \min\{k, \qquad U_k + V_k = N\}$$

Équilibre :

$$\mathbb{P}(U_{k+1} = U_k + 1) = \frac{V_k}{V_k + U_k} \xrightarrow[k \to +\infty]{} \alpha$$

$$\frac{U_k}{k} \xrightarrow[k \to +\infty]{} \alpha$$

$$\frac{V_k}{U_k + V_k} \sim \frac{\left(1 - \frac{2U_k}{k}\right)}{1 - \frac{U_k}{k}}$$

$$\alpha = \frac{(1 - 2\alpha)}{1 - \alpha}$$

$$\alpha = \frac{\left(3 - \sqrt{5}\right)}{2} \cong 0.382$$

Contraintes :

$$\begin{cases} U_K = K\alpha \\ U_K + V_K + W_K = K \\ 2W_K + V_K = K - 1 \approx K \end{cases}$$

Proportions :

$$(0.382, 0.236, 0.382)$$

$$U_K + V_K = N \Rightarrow K = \frac{N}{(0.382 + 0.236)} = 1.62N$$

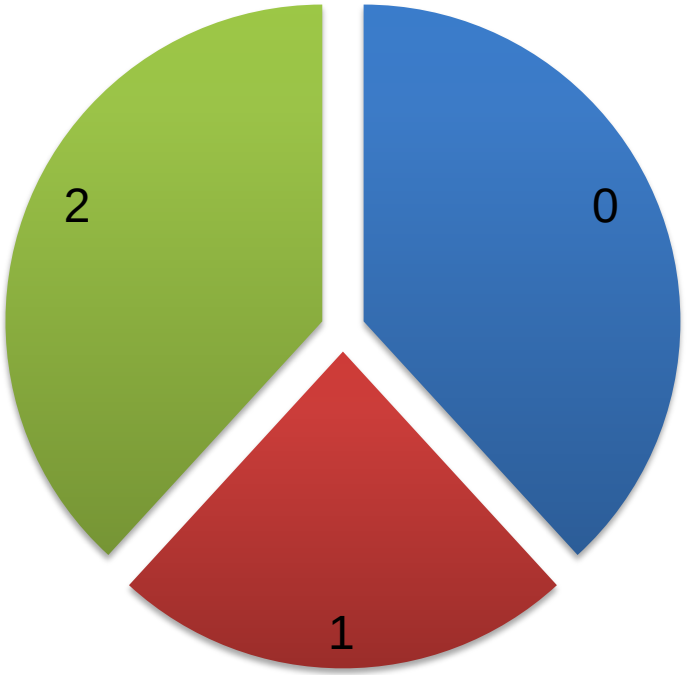
Vérification :

$$W_K = 0.62N = 0.62(U_K + V_K) \text{ car } 0.62 * (0.382 + 0.236) = 0.382$$

Proportions en fonction du nombre de lettres vendues

Proportions :

$(0.382, 0.236, 0.382)$



Chaîne de Lettres

Systèmes De Ponzi

Variables essentielles

Dépôt initial : S_0

Flux entrant : $s(t)$

Taux nominal promis : r_p

Taux nominal d'intérêt : r_n

Taux nominal de retrait : r_w

Taux réels et nominaux :

$\boldsymbol{R_n}$ et $\boldsymbol{R_w}$ réels

$$S(t+1) = S(t)(1+R_n)(1-R_w)$$

$\boldsymbol{r_n}$ et $\boldsymbol{r_w}$ nominaux

$$S(t+dt) = S(t)(1+r_n dt)(1-r_w dt)$$

$$\Rightarrow S(t+1) = S(t)(1+r_n dt)^{\frac{1}{dt}}(1-r_w dt)^{\frac{1}{dt}} \xrightarrow[dt\to 0]{} S(t)e^{r_n - r_w}$$

$$S(t) = S_0 e^{t(r_n - r_w)}$$

Argent promis :

$$\frac{dS_f}{dt}(t) = (r_p - r_w)S_f(t) + s(t)$$

$$S_f(t) = e^{(r_p - r_w)t}\left(S_0 + \int_0^t e^{-(r_p - r_w)u} s(u)du\right)$$

Argent réel :

$$\frac{dS}{dt}(t) = r_n S(t) + s(t) - r_w S_f(t)$$

$$S(t) = e^{r_n t}\left(S_0 + \int_0^t e^{-r_n u}\big(s(u) - r_w S_f(u)\big)du\right)$$

Investissement :

$$s(t) = S_0(e^{r_i t} - 1)$$

Argent promis :

$$S_f(t) = S_0\left(-\frac{1}{r_w - r_p}\right) + S_0\left(\frac{1}{r_i + r_w - r_p}\right)e^{r_i t} + S_0\left(1 + \frac{1}{r_w - r_p} - \frac{1}{r_i + r_w - r_p}\right)e^{(r_p - r_w)t}$$

Argent réel :

$$S(t) = a + be^{r_n t} + ce^{r_i t} + de^{(r_p - r_w)t}$$

Calcul des constantes :

$$a = S_0\left(\frac{1}{r_n} - \frac{r_w}{r_n(r_w - r_p)}\right)$$

$$b = S_0\left(1 - \frac{1}{r_i - r_n} - \frac{1}{r_n} + \frac{r_w}{r_n(r_w - r_p)} + \frac{r_w}{(r_i + r_w - r_p)(r_i - r_n)} + \frac{r_w}{r_p - r_w - r_n}\left(1 + \frac{1}{r_w - r_p} - \frac{1}{r_i + r_w - r_p}\right)\right)$$

$$c = S_0\left(\frac{1}{r_i - r_n} - \frac{r_w}{(r_i - r_n)(r_i + r_w - r_p)}\right)$$

$$d = S_0\left(-\frac{r_w}{r_p - r_w - r_n}\right)\left(1 + \frac{1}{r_w - r_p} - \frac{1}{r_i + r_w - r_p}\right)$$

Madoff :

$$r_p = 10\%, \qquad r_n = 2.5\%, \qquad S_0 = 1$$

$$r_w = ?, \qquad r_i = ?,$$

Date de l'effondrement (années) :

r_w/r_i	0%	2%	4%	6%	8%
1%	31,3	55,6	63,6	73,9	95,8
3%	20,3	40,6	47,4	54,6	67,6
5%	16,2	34,6	41,1	47,3	57,8
7%	13,8	31,0	37,4	43,1	52,6
9%	12,2	28,6	35,0	40,4	49,2

Son escroquerie a duré 40 ans

Gains :

$$W(t) = -S_0 + \int_0^t \left(r_w S_f(u) - s(u)\right) du$$

$$W(t) = \alpha + \beta e^{r_i t} + \gamma e^{(r_p - r_w)}$$

$$\alpha = S_0 \left(-1 + \frac{1}{r_i} - \frac{r_w}{r_i(r_i + r_w - r_p)} - \frac{r_w}{r_p - r_w} \left(1 + \frac{1}{r_w - r_p} - \frac{1}{r_i + r_w - r_p} \right) \right)$$

$$\beta = S_0 \left(-\frac{1}{r_i} + \frac{r_w}{r_i(r_i + r_w - r_p)} \right)$$

$$\gamma = S_0 \left(\frac{r_w}{r_p - r_w} \right) \left(1 + \frac{1}{r_w - r_p} - \frac{1}{r_i + r_w - r_p} \right)$$

Valeurs de taux

Gain total à l'effondrement :

r_w/r_i	0%	2%	4%	6%	8%
1%	−34,0	−48,9	−17,7	142,0	2213,7
3%	−28,7	−54,7	−55,4	−38,0	90,5
5%	−32,2	−67,5	−76,0	−76,5	−49,5
7%	−45,8	−102,3	−120,9	−133,3	−139,4
9%	−121,8	−185,3	−347,3	−396,9	−469,7

Chaîne de Lettres

Tableau des Constantes

Modèle	Seuil	Temps d'arrêt	Proportions à l'arrivée	Espérance à l'origine
Vente pyramidale	$1/e$ $\cong 0.368$	N	50% 0 25% 1 25% 2 +	$d \ln N$
Markov	$\frac{(3-\sqrt{5})}{2}$ $\cong 0.382$	$1.62N$	38.2% 0 23.6% 1 38.2% 2	
Branchement		$\frac{C}{1-\mathbb{E}(Z_1)}$		$50(m+m^{12}-2)$
Ponzi	$r_i > 6\%$ $r_w < 3\%$	~40 ans	$12\% > 0$ $88\% < 0$	$-120S_0$