Etre ruiné à un jeu de PILE ou FACE, un jeu de probabilité très complexe.

Année 2022-2023

Elèves : Jean-Batiste Devaux (2nde), Radwane Aissaoui, Pauline Favre, Morgan Laurent, Mamou Traoré (Terminale générale spécialité mathématiques)

Enseignant : Nicolas Broussan

Établissement : Lycée Jacques Amyot (Melun 77)

Chercheur : Batiste Goujaud CMAP, Ecole Polytechnique, Institut Polytechnique de Paris

I – Problème étudié

On réalise un jeu de hasard avec une pièce équilibrée, en sachant que lorsqu'on obtient PILE, on gagne 2 € et lorsqu'on obtient FACE, on perds 1€.

La question à laquelle nous devions répondre était la suivante :

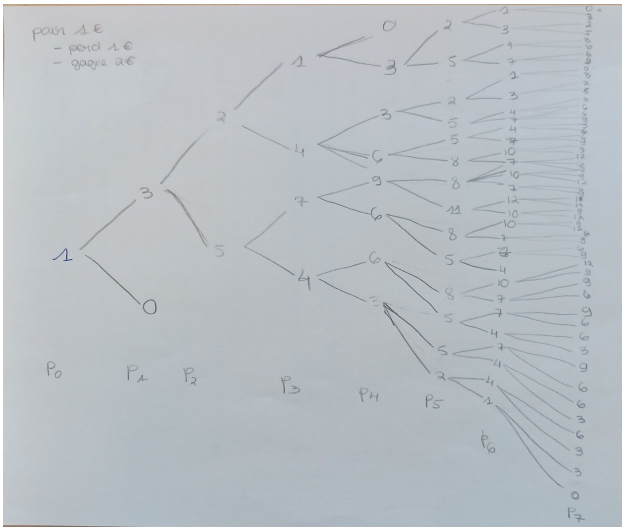
Quelle est la probabilité d'être ruiné sachant qu'on a au départ *n* euros ?

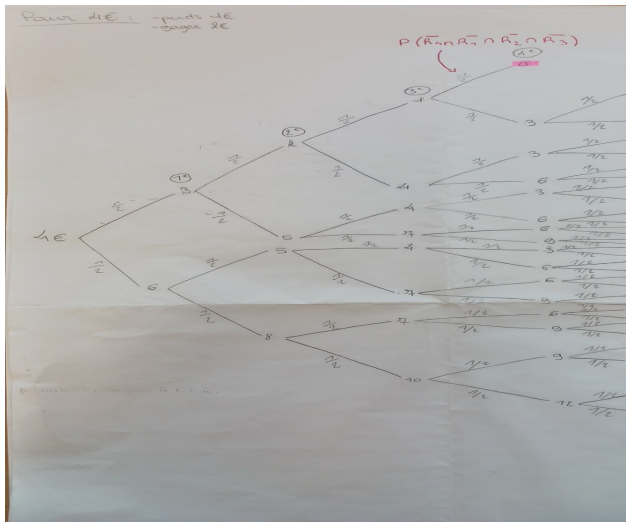
II – Premières recherches : arbres pondérés

Nous avons utilisé nos connaissances mathématiques en probabilités pour modéliser ce problème à l'aide d'arbres pondérés, d'abord « à la main », puis à l'aide d'un programme Python.

a) Nos arbres de probabilité à la main

Dans un premier temps, nous avons réalisé plusieurs arbres de probabilités avec pour somme de départ : 1€, 2€, 3€ et 4€.





Ensuite nous avons calculé différentes probabilités en notant les évènements, avec n entier naturel non nul :

- R_n : « être ruiné à la partie n » ;
- T_n : « être ruiné au plus tard à la partie n » ;
- U_n : « être ruiné uniquement à la partie n ».

Ainsi, pour la probabilité d'être ruiné au plus tard à la partie 2 avec 4 euros comme somme de départ, on obtient $P(T_2)=P(R_1)+P(\overline{R_1}\cap R_2)=\frac{1}{2}+(\frac{1}{2})^2=\frac{3}{4}$

De même, pour la probabilité d'être ruiné uniquement à la partie 4, en partant avec 4 euros en somme de départ : $P(U_4)=P(\overline{R_1}\cap\overline{R_2}\cap\overline{R_3}\cap R_4)=(\frac{1}{2})^4=\frac{1}{16}$

Toutefois, au fil de la réalisation d'arbres de probabilités nous nous sommes vite rendus compte que nous perdions énormément de temps à les réaliser, nous avons donc décidé de les générer à l'aide d'un programme Python.

b). <u>Nos arbres de probabilité sur python :</u>

```python
# Module Python
import asyncio, time, math

# Importation des fonctions nécéssaire à l'affichage
from excel import makeExcel
from graphic import makeGraphic

# Valeur du gain, perte ainsi que la tableau de valeur
GAIN, LOSS, TAB = 2, 1, {}

# Fonction recursive pour enregister tout les branches
async def check(columns:int, row:int, euro:int, step:int = 0) -> None:
    if euro == row:
        # Mise à jour du nombre de branche dans le tableau de valeur
        if row not in TAB:
            TAB[row] = {i: 0 for i in range(columns + 1)}
        TAB[row][step] += 1

    # Vérifie si il na va pas trop long ou si l'on est pas ruiné
    if euro > 0 and step < columns:
        # Vérifie les scénarios gagnant et perdant
        await check(columns, row, euro + GAIN, step + 1)
        await check(columns, row, euro - LOSS, step + 1)
```

Dans cette première partie, nous avons importé tous les modules et fonctions dont nous avions besoin ; nous avons aussi établi les constantes afin d'avoir un code plus clair. Ensuite, on a crée la fonction à la base de tout notre programme qui nous permettra une fois lancée de générer l'arbre.

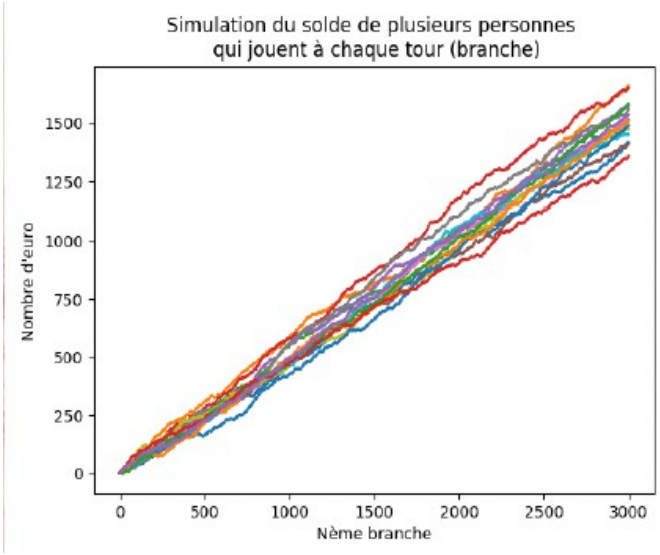
```python
# Demander à l'utilisateur le nombre de colonnes et de lignes
columns = min(int(input('Nombre de colonne (Branche - Maximun 20) : ')), 20)
rows = min(int(input('Nombre de ligne (Somme Euro - Maximun 20) : ')), 20)

# Demander à l'utilisateur la somme initiale
sum = int(input('Somme initial : '))

# Afficher un message pour indiquer le chargement des données
print('Chargement des données...')

# Enregistrer l'heure de début
start_time = time.time()

# Lancement des calculs de branche
for row in range(rows + 1):
    asyncio.run(check(columns, row, sum))

# Transforme le tableau de valeur en fichier excel
makeExcel(TAB, columns)
# Transforme les ruines du tableau de valeur en un graphique
makeGraphic(TAB[0])

# Afficher le temps écoulé
elapsed_time = time.time() - start_time
print(f"Données chargées en {(elapsed_time):.2f} secondes.")
```

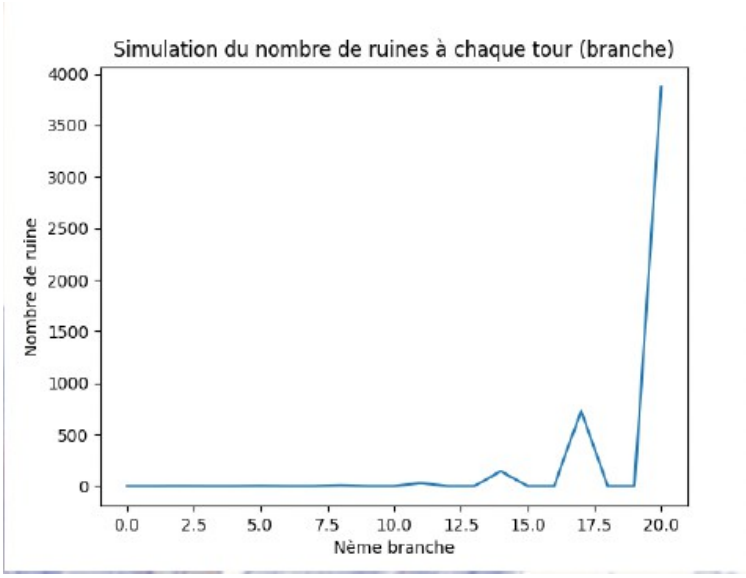
Dans cette seconde partie, le programme demande le nombre de colonnes (une colonne représente un lancer) et de lignes (on assigne à chaque ligne une somme) afin de mettre en place notre tableau ; ensuite on demande la somme initiale « n euros » à laquelle notre programme commencera, sachant qu'il ne peut y avoir plus de 20 colonnes (la variable qui donne le temps d'exécution du programme nous a obligé à limiter le nombre de lancers à 20).

Une fois l'arbre crée,le programme générera une feuille de calcul automatisé sur Excel ainsi que 2 graphiques sous forme d'images :

```python
import xlsxwriter

def makeExcel(tab:dict, columns:int) -> None:
    # Crée un nouveau fichier Excel et ajoutez une feuille de calcul.
    workbook = xlsxwriter.Workbook('./result/proba.xlsx')
    worksheet = workbook.add_worksheet()

    # Ajoute un format gras à utiliser pour mettre en surbrillance les cellules.
    bold = workbook.add_format({'bold': True})

    # Élargisse la première colonne pour rendre le texte plus clair.
    worksheet.set_column('A:A', 15)

    # Texte avec mise en forme.
    worksheet.write('A1', 'Somme \ Etape', bold)

    # Écrivez quelques nombres, avec une notation de ligne/colonne.
    for c in range(columns + 1):
        worksheet.write(f"{chr(ord('A') + c + 1)}1", c, bold)

    for row in tab:
        worksheet.write(f"A{row + 2}", str(row) + 'E', bold)

        for column, value in tab[row].items():
            worksheet.write(row + 1, column + 1, value)

    # Fermer et enregistrer le fichier Excel
    workbook.close()
```

Somme \ Etape	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0E	0	0	1	0	0	2	0	0	7	0	0	30	0	0	143	0
1E	0	1	0	0	2	0	0	7	0	0	30	0	0	143	0	0
2E	1	0	0	2	0	0	7	0	0	30	0	0	143	0	0	728
3E	0	0	2	0	0	7	0	0	30	0	0	143	0	0	728	0
4E	0	1	0	0	5	0	0	23	0	0	113	0	0	585	0	0
5E	0	0	0	3	0	0	16	0	0	83	0	0	442	0	0	2420
6E	0	0	1	0	0	9	0	0	53	0	0	299	0	0	1692	0
7E	0	0	0	0	4	0	0	30	0	0	186	0	0	1107	0	0
8E	0	0	0	1	0	0	14	0	0	103	0	0	665	0	0	4122
9E	0	0	0	0	0	5	0	0	50	0	0	366	0	0	2430	0
10E	0	0	0	0	1	0	0	20	0	0	180	0	0	1323	0	0





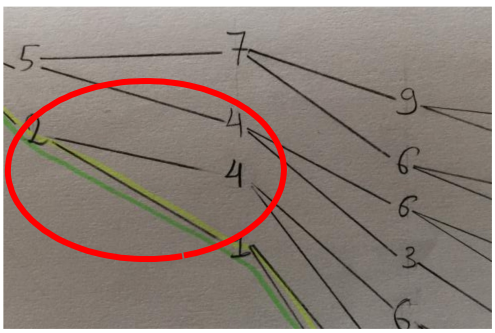
Grâce à ses simulations, nous nous sommes aperçus que plus on a d'argent au départ, moins on est ruiné (graphique de gauche) ; et le nombre de parties où le joueur peut-être ruiné ne dépend pas du numéro de lancer. De plus, grâce à la feuille de calcul automatisée nous nous sommes aperçus que le nombre de ruines à un numéro de lancer ne peut être donné à l'aide du nombre de ruines aux lancers précédents.

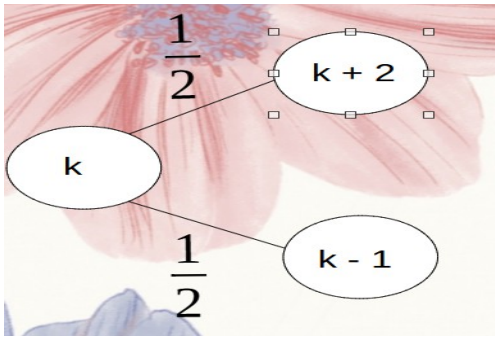
III – Un nouveau chemin vers la résolution du problème : la propriété de Markov Faible

La suite des valeurs de la somme d'argent constitue une chaîne de Markov : chaque étape ne dépend que de la précédente.

Nous avons appliqué la propriété de Markov que nous avons admise :

Cette propriété dit que la probabilité d'être ruiné à l'étape n sachant qu'on a k euros dépend de la probabilité d'être ruiné à l'étape $n-1$:





k représente la somme en euros

On note, pour tout $n \in \mathbb{N}$, u_n la probabilité d'être ruinée sachant qu'on a n euros au départ.

Grâce à cette propriété, on en déduit la forme récurrente de la suite (u_n) : pour tout k nombre entier naturel non nul :

$u_k = \frac{1}{2} u_{k+2} + \frac{1}{2} u_{k-1}$ © $2u_k = u_{k+2} + u_{k-1}$

© $u_{j+1+2} = 2u_{j+1} - u_j$ en posant $j = k-1$

© $u_{j+3} = 2u_{j+1} - u_j$

© $u_{k+3} = 2u_{k+1} - u_k$ en posant $k = j$

© $u_{k+3} - 2u_{k+1} + u_k = 0$

(u_n) est donc une suite linéaire récurrente d'ordre 3.

Nous avons par la suite cherché la forme explicite de cette suite, associé à l'équation caractéristique $X^3 - 2X + 1 = 0$.

IV – Résolution de l'équation caractéristique

On rappelle que l'on cherche à résoudre $X^3 - 2X + 1 = 0$.

$1^3 - 2 \times 1 + 1 = 0$ donc, $X = 1$ est une solution évidente.

On peut donc factoriser le polynôme $X^3 - 2X + 1$ par $X - 1$:

$X^3 - 2X + 1 = (X-1)(aX^2 + bX + c)$ avec a, b, c des constantes réelles à déterminer.

Or :

$$\begin{aligned}(X-1)(aX^2+bX+c) &= aX^3 + bX^2 + cX - aX^2 - bX - c \\ &= aX^3 - {}^2aX + {}^2bX - bX + cX - c \\ &= aX^3 + (b-a)^2 X + (c-b)X - c \ .\end{aligned}$$

Donc :

$1X^3 - 2X + 1 = aX^3 + (b-a)^2 X + (c-b)X - c$.

Ces deux polynômes sont égaux, on a donc nécessairement :

$$\begin{cases} & & a=1 \\ b & - & a=0 \\ c & - & b=-2 \\ & & -c=1 \end{cases} \quad \copyright \quad \begin{cases} a=1 \\ b=1 \\ c=-1 \end{cases}$$

Ainsi :

$$X^3-X+1=(X-1)(1X^2+1X-1)$$
$$= (X-1)(X^2+X-1)$$

Revenons à la résolution de $X^3-2X+1=0$:

Cette équation équivaut à :

$$(X-1)(X^2+X-1)=0 \ \copyright \ X-1=0 \text{ ou } X^2+X-1=0$$

$X^2+X-1=0$ est une équation du second degré, on commence par calculer le discriminant $\Delta=b^2-4ac$.

$\Delta=b^2-4ac=1^2-4x1x(-1)=5$.

$\Delta>0$, l'équation possède donc deux solutions :

$$\frac{-b-\sqrt{\Delta}}{2a}=\frac{-1-\sqrt{5}}{2} \quad \text{et} \ \frac{-b+\sqrt{\Delta}}{2a}=\frac{-1+\sqrt{5}}{2}$$

Finalement :

$$(X-1)(X^2+X-1)=0 \ \copyright \ X=1 \text{ ou } X=\frac{-1-\sqrt{5}}{2} \text{ ou } X=\frac{-1+\sqrt{5}}{2}$$

V – La forme explicite de (u_n)

Donc la forme explicite de (u_n) est :

$$\forall n \in \mathbb{N},\ u_n= \alpha\times 1^n+ \beta\left(\frac{-1-\sqrt{5}}{2}\right)^n+ \gamma\left(\frac{-1+\sqrt{5}}{2}\right)^n \text{ avec } \alpha,\ \beta,\ \gamma \in \mathbb{R} .$$

soit :

$$\forall n \in \mathbb{N},\ u_n= \alpha+ \beta\left(\frac{-1-\sqrt{5}}{2}\right)^n+ \gamma\left(\frac{-1+\sqrt{5}}{2}\right)^n$$

On souhaite déterminer les constantes α, β et γ.

Pour cela calculons β, en utilisant la limite de (u_n) en $+\infty$:

$\left(\left(\frac{-1+\sqrt{5}}{2}\right)^n\right)_{n \in \mathbb{N}}$ est une suite géométrique de raison $q=\frac{-1+\sqrt{5}}{2}\approx 0{,}618$, donc $-1 < q < 1$.

Donc $\lim\limits_{n\to\infty}\left(\frac{-1+\sqrt{5}}{2}\right)^n=0$

Donc par produit $\lim\limits_{n\to\infty}\gamma\left(\frac{-1+\sqrt{5}}{2}\right)^n=0$

Donc par somme $\lim\limits_{n\to\infty}\alpha+\gamma\left(\frac{-1+\sqrt{5}}{2}\right)^n=\alpha$

D'autre part $\left(\left(\frac{-1-\sqrt{5}}{2}\right)^n\right)_{n\in\mathbb{N}}$ est une suite géométrique de raison $k=\frac{-1-\sqrt{5}}{2}\approx -1{,}618$, donc $k<-1$.

Donc la suite $\left(\left(\frac{-1-\sqrt{5}}{2}\right)^n\right)$ n'a pas de limite en +∞.

On en conclut que en $+\infty$, (u_n) se comporte comme $\left(\alpha+\beta k^n\right)_{na\,\mathbb{N}}$

Supposons que $\beta \neq 0$:

On a $|k|>1$ donc $\lim\limits_{n\to\infty}|k|^n=+\infty$,

donc par produit $\lim\limits_{n\to\infty}\left|\beta k^n\right|=+\infty$

et par somme $\lim\limits_{n\to\infty}\left|\alpha+\beta k^n\right|=+\infty$, donc $\lim\limits_{n\to+\infty}|u_n|=+\infty$ (*)

Or (u_n) est une probabilité donc $\lim\limits_{n\to\infty}u_n\in[0;1]$, donc $\lim\limits_{n\to+\infty}|u_n|\in[0;1]$, ce qui est contradictoire avec (*).

On en conclut donc que $\boldsymbol{\beta=0}$.

Donc $\forall n \in \mathbb{N}$, $u_n= \alpha+ \gamma \left(\frac{-1+\sqrt{5}}{2}\right)^n$.

Calculons à présent α :

On a $\lim\limits_{n\to\infty} u_n = \alpha$.

D'autre part, l'espérance mathématique lié à chaque lancer de la pièce est :

$$\frac{1}{2}\times 2 + \frac{1}{2}\times(-1) = 0{,}5\,€$$

Donc le jeu est favorable pour le joueur et si on lance k fois la pièce, de manière indépendante, l'espérance mathématique du gain est alors de $k\times 0{,}5\,€$ (linéarité de l'espérance mathématique).

On admet donc que plus on lance la pièce, plus la probabilité d'être ruinée est proche de 0 (voir graphique page 4 pour une approche par simulation) donc on admet que $\lim\limits_{n\to\infty} u_n = 0$.

Donc par unicité de la limite $\alpha = 0$.

Donc, $\forall n \in \mathbb{N},\ u_n = \gamma\left(\frac{-1+\sqrt{5}}{2}\right)^n$.

Calculons γ :

On a en particulier pour $n=0$: $u_0 = \gamma\left(\frac{-1+\sqrt{5}}{2}\right)^0 = \gamma\times 1 = \gamma$.

or l'évènement « Etre ruiné avec 0 € au départ » est certain donc $u_0 = 1$ donc $\gamma = 1$.

Finalement la probabilité d'être ruinée avec une somme de départ de n € est :

$$\boxed{\forall \boldsymbol{n} \in \mathbb{N},\ \boldsymbol{u_n} = \left(\frac{\boldsymbol{-1+\sqrt{5}}}{\boldsymbol{2}}\right)^{\boldsymbol{n}}}$$