

Stratégies d'arbitrage dans les paris sportifs

Léo Pouilly (15004)

CPGE Scientifique

TIPE : Jeux et Sports

Introduction



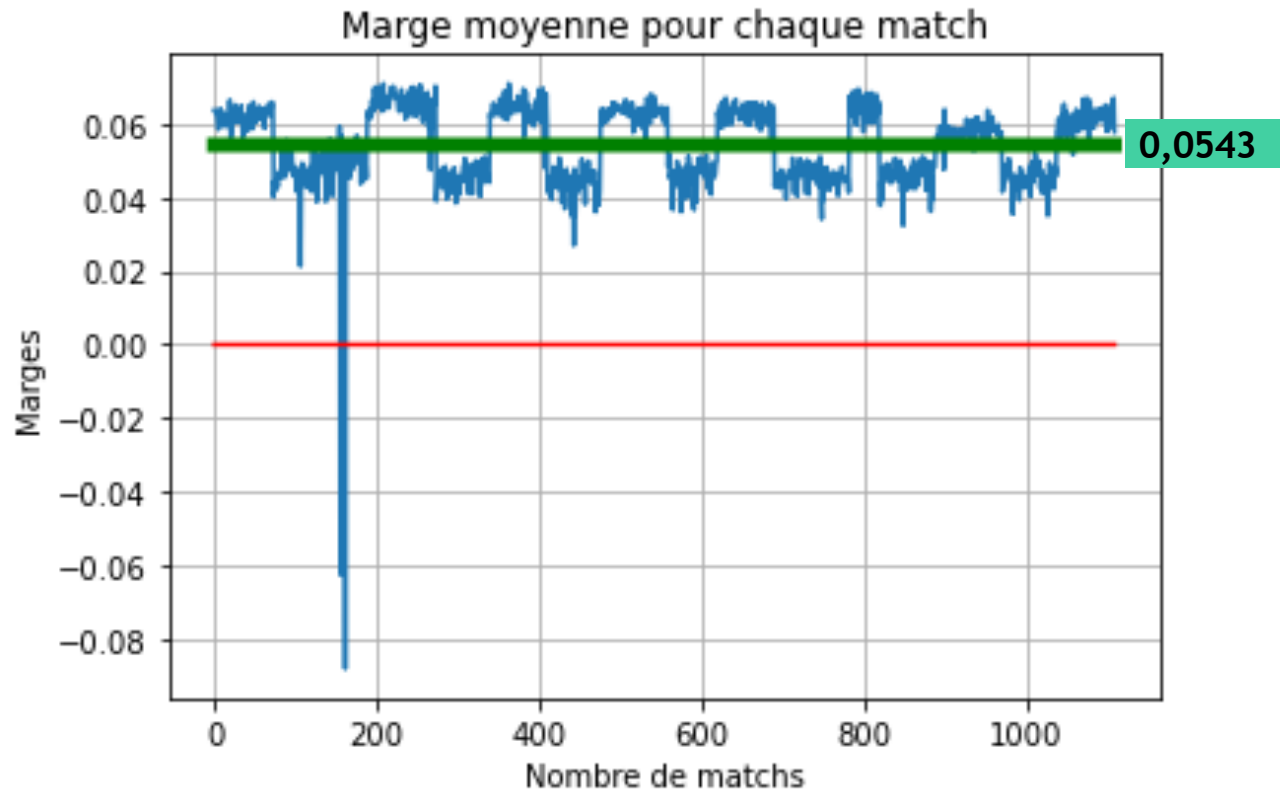
Cote du
joueur n°1



Cote du
joueur n°2

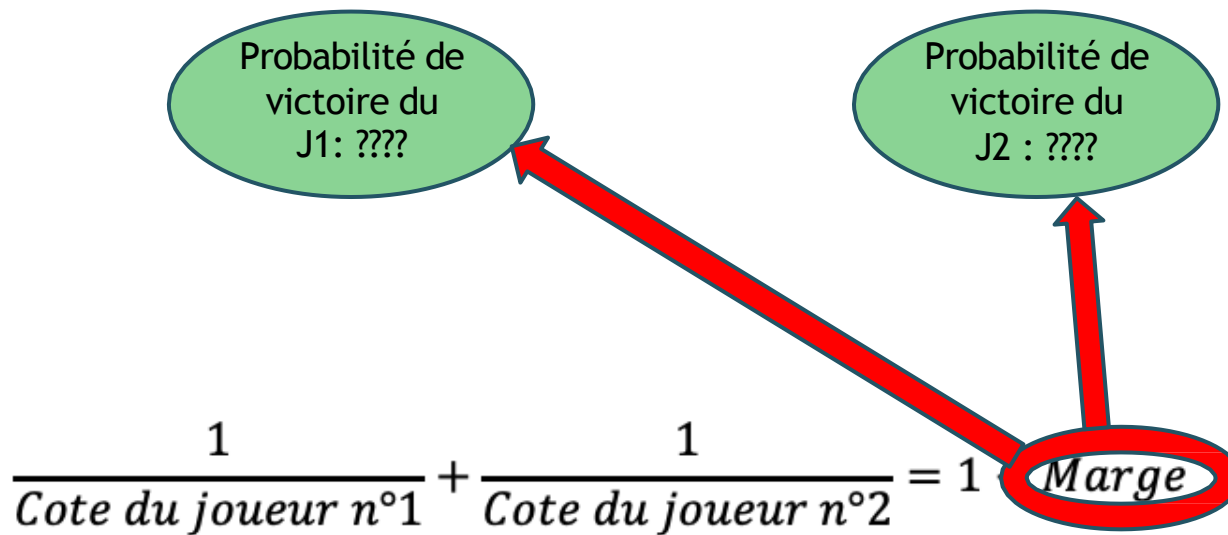
$$\text{On a } \frac{1}{\text{Cote du joueur n°1}} + \frac{1}{\text{Cote du joueur n°2}} = 1 + \text{Marge}$$

Introduction



Marges établies par les bookmakers pour 1108 matchs de tennis

Existe t-il une stratégie pour corriger cette marge ?



DANS QUELLE MESURE EST-IL POSSIBLE
D'ÉTABLIR UNE MODÉLISATION ADÉQUATE
POUR UNE TELLE STRATÉGIE ?

► **Stratégie d'arbitrage**

- A) Définition
- B) Comment la mettre en place ?
- C) Limites et inconvénients de ce modèle

► **"Web scraping": récolte des données**

- A) Comment les scraper (ie « récolter ») sur une page web ?
- B) «Parser» les données (« Data Parsing »)
- C) Mise en place de dictionnaires exploitables (filtration des données)
- D) Sauvegarde des données dans les fichiers Json

► **Martingale derrière les paris**

- A) Définition mathématique
- B) Lien entre martingale et arbitrage
- C) Modélisation de la martingale
- D) Analyse asymptotique




A) Définition

Définition

Une **stratégie d'arbitrage** est une technique utilisée en finance, qui consiste à tirer profit **des différences de prix pour un même actif**, sur *deux marchés différents*.

Pour les **paris sportifs** l'arbitrage se résume à parier sur **deux sites** de paris sportifs **distincts (ie deux bookmakers différents)**

A) Définition

	Joueur n° 1 	Joueur n° 2 
Sites de paris sportif n° 1	1,78	2,35
Sites de paris sportif n° 2	1,86	2,26

A) Définition

- ▶ On choisit les 2 plus grosses cotes de telle sorte que le pari **ne soit pas totalement recouvert**:

$$\frac{1}{c_1} + \frac{1}{c'_2} = 1 - \alpha < 1, \text{ avec } \alpha > 0$$

- ▶ c_1 : cote du joueur n°1 sur un site de paris sportifs B
- ▶ c'_2 : cote du joueur n°2 sur un **site de paris sportifs différent de B'**

B) Comment la mettre en place ?

- ▶ Hypothèse: $\frac{1}{c_1} + \frac{1}{c'_2} = 1 - \alpha < 1$, avec $\alpha > 0$

- ▶ Formule du gain:

Mise

Fortune

Marge du bookmaker

dans le cas d'un arbitrage : $M_{n,i} = \frac{S_n}{(1 - \alpha) \times C_i}$ avec $\alpha > 0$

$M_{n,i}$: mise lors du n – ième sur le joueur n° i pour $i \in \{1,2\}$

De cette façon, si on gagne alors:

$$S_{n+1} = M_{n,i} \times C_i = \frac{S_n}{1 - \alpha} > S_n$$

Cote joueur n° i

↑

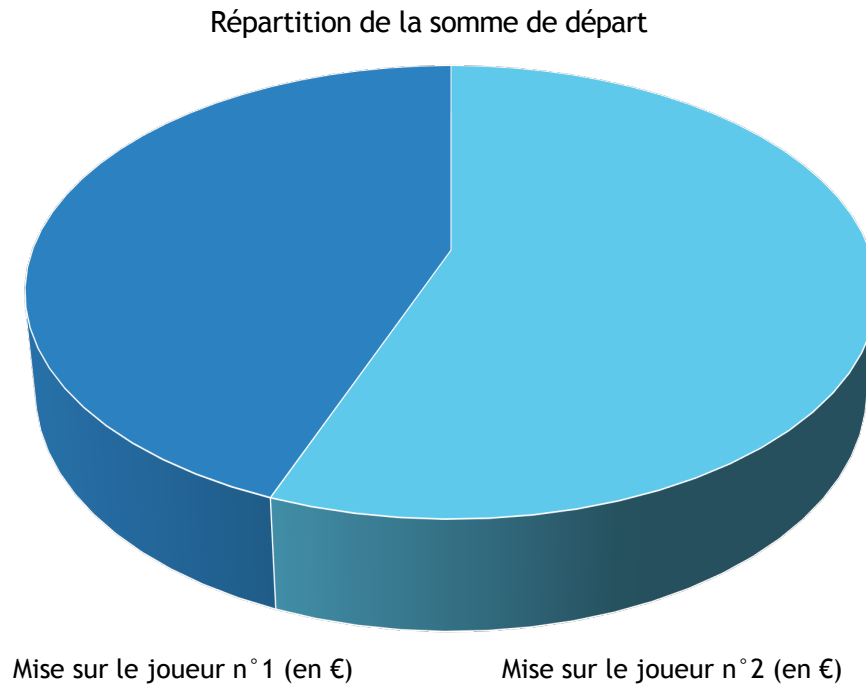
B) Comment la mettre en place ?

Illustration avec un exemple:

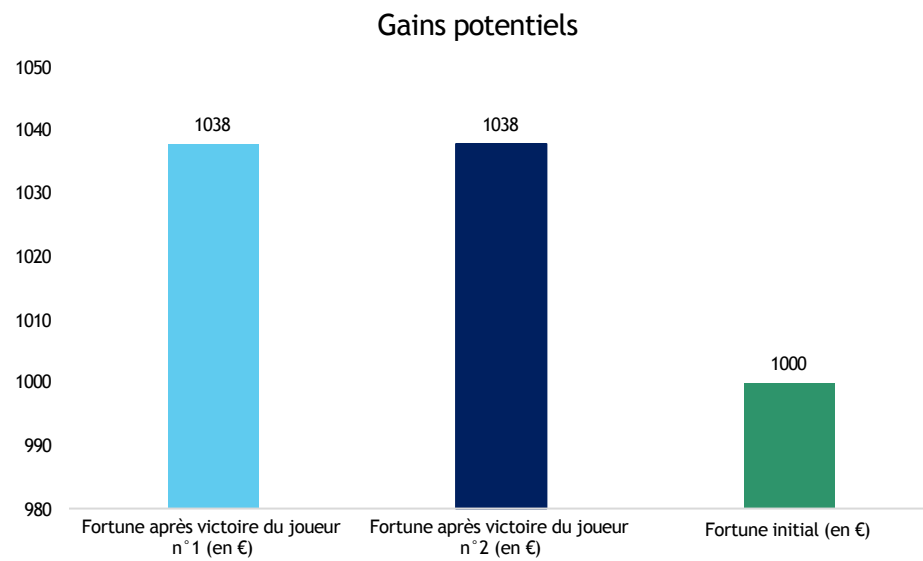
	Cote joueur n° 1	Cote joueur n° 2
Bookmaker n° 1	1,78	2,35
Bookmaker n° 2	1,86	2,26

- ▶ Marge du bookmaker : $1 - \left(\frac{1}{1,86} + \frac{1}{2,35} \right) \approx 0,037$
- ▶ Fortune de départ : 1000 €
- ▶ Mise sur le joueur n° 1 : $\frac{1000}{(1 - 0,037) \times 1,86} \approx 558 \text{ €}$
- ▶ Mise sur le joueur n° 2 : $\frac{1000}{(1 - 0,037) \times 2,35} \approx 442 \text{ €}$

B) Comment la mettre en place ?



B) Comment la mettre en place ?



B) Comment la mettre en place ?

Avantages:

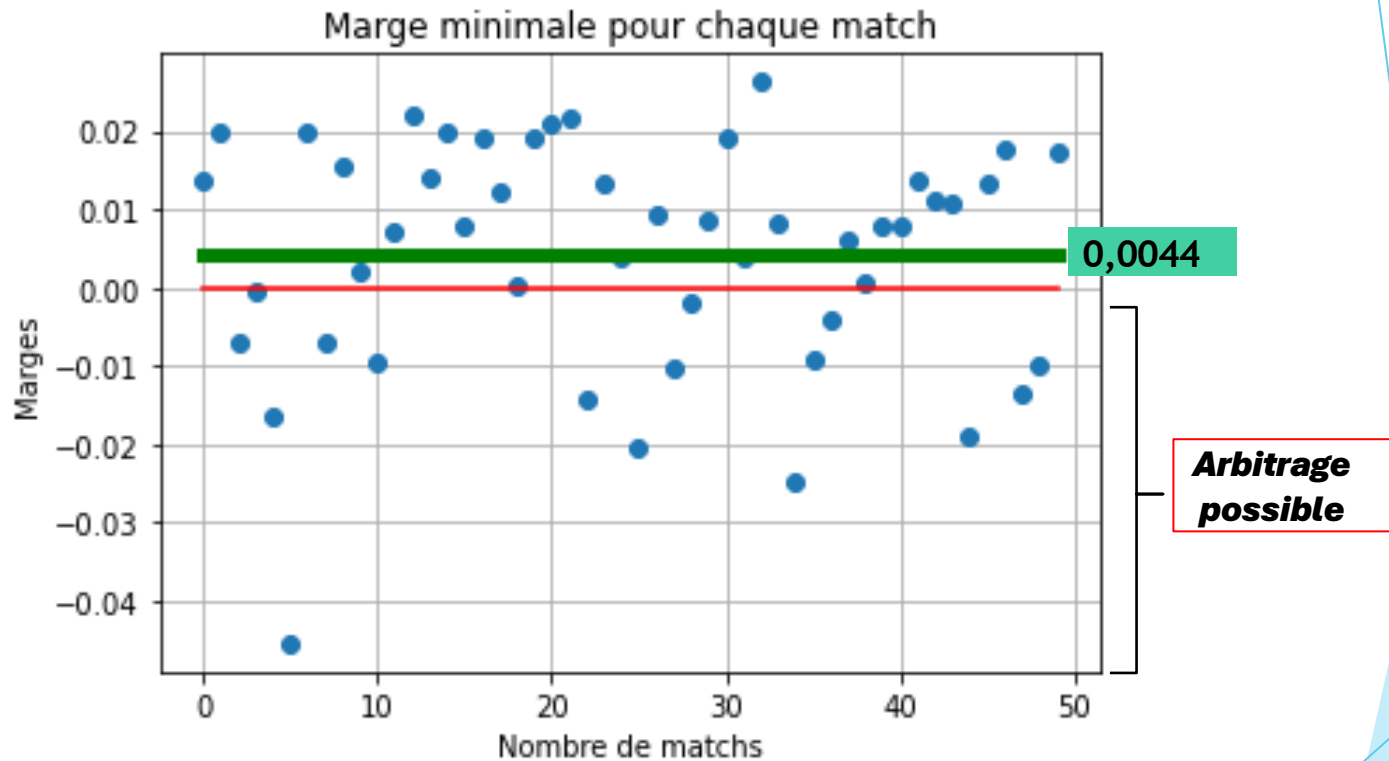
■ **Profit assuré:** obtention d'un gain quel que soit l'issue du match

■ **Risque minime:** stratégie à faible risque car indépendante de l'issue de l'événement

Inconvénients:

- **Exploitation difficile:** fluctuations sans cesse des cotes
- **Très surveillée par les bookmakers:** qui peuvent réduire l'utilisation de cette stratégie en gelant les comptes des parieurs et donc l'utilisation de leur gain
- **Rarement utilisable:** sur 50 matchs, seulement quelques uns sont des opportunités d'arbitrages

C) Limites et inconvénient de ce modèle



A) Comment les scraper (ie récolter) sur une page web ?

- **Scraping des URLs** : Récupération du code source des différentes pages de statistiques qui nous intéressent

Problème : présence de fonctions javascript, permettant aux gérants du site de faire le lien entre leurs données et celles présentes sur le site

Novak Djokovic | Overview | x +

atptour.com/en/players/novak-djokovic/d643/overview

Maps YouTube Gmail

AI & DIGITAL INNOVATION PARTNER

EN

ATP TOUR THIS IS TENNIS

Scores Latest H2H Stats Rankings Players Tournaments Challenger More

Q Emirates

NOVAK DJOKOVIC


🔊

Singles Doubles

YTD	1 Rank	- Move	14 - 6 W-L	0 Titles	\$1,400,552 Prize Money
Career	1 Career High Rank (2011.07.04)	1101 - 219 W-L	98 Titles	\$182,043,906 Prize Money Singles & Doubles Combined	

Overview Bio Activity Stats Ranking

OFFICIAL NEWS IN YOUR INBOX SUBSCRIBE NOW

ATP TOUR

f X in

Personal details

Age	37 (1987.05.22)	Country	Serbia 🇷🇸
Weight	170 lbs (77kg)	Birthplace	Belgrade, Serbia
Height	6'2" (188cm)	Plays	Right-Handed, Two-Handed Backhand
Turned pro	2003	Coach	
Follow player	f i X v		


Latest news

View all →


NEWS

View All →

Related Most Recent



Djokovic edges Musetti in marathon late-night thriller



Djokovic on Nadal: 'I've experienced his evolution'

DevTools is now available in French!

Always match Chrome's language Switch DevTools to French Don't show again

Elements Console Sources Network >>

1 8 1

!DOCTYPE html>
<html class="no-js" lang="en" dir="ltr">
><head> </head>
><body class style>
> <!-- GTM Body Tag -->
> <!-- Google Tag Manager (noscript) -->
><noscript> </noscript>
><style nonce> .gtm-hide { display: none; visibility: hidden; } </style>
><!-- End Google Tag Manager (noscript) -->
><!--start header section -->
><div class="atp_header" id="atpTourHeaderApp" data-v-app>
>><div class="atp_header-secondary"> </div> (flex)
>><div class="atp_header-primary"> </div> (flex)
><!-->
>><input type="hidden" class="atp_header-endpoint" value="/en/-/tour/navigationtop"> ==
>><input type="hidden" class="atp_header-currentPage" value="{F68BD0BF-192D-4090-84D3-E
>><input type="hidden" class="atp_header-currentHead" value="{0FA961C8-E757-4840-8447-6
>><input type="hidden" class="atp_header-isMoreActive" value="false">
></div>
>><script nonce id="atpTourHeaderSponsor" type="text/x-template"> </script>
>><script id="atpTourNavigation" type="text/x-template"> </script>
html.no-js body div#atpTourHeaderApp.atp_header input.atp_header-endpoint
Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility
Filter :hov .cls + ⌨ □
element.style {
}
*, ::after, ::before {
-webkit-box-sizing: border-box;
box-sizing: border-box;
}
* {
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
outline: 0;
}
18/

A) Comment les «scraper»(ie récolter) sur une page web ?

Solution:
javascript



pour supprimer la partie

Lancement de plusieurs instances de navigateurs pour récupérer les données **présentes dans les codes sources** sur le site: <https://www.atptour.com/en> :

Celles de **Overview** (Main) , **Stats** (Tout), et **Activity** (carrière):

A) Comment les «scraper»(ie récolter) sur une page web ?

The screenshot shows a web interface for a tennis player's profile. At the top, there is a navigation bar with tabs: Overview, Bio, Activity, Stats, and Ranking. Below this, there is a section for Google Ads with a button to 'Envoyer un commentaire' and a link 'Pourquoi cette annonce ?'. The main content area is titled 'Personal details' and contains two columns of information. The 'Plays' field, which indicates the player's handedness and backhand style, is highlighted with a red rectangular box.

Personal details	
Age	37 (1987.05.22)
Weight	170 lbs (77kg)
Height	6'2" (188cm)
Turned pro	2003
Country	Serbia 🇷🇸
Birthplace	Belgrade, Serbia
Plays	Right-Handed, Two-Handed Backhand
Coach	

A) Comment les «scraper»(ie récolter) sur une page web ?

Overview Bio Activity **Stats** Ranking

←

Annonces Google

Envoyer un commentaire Pourquoi cette annonce ? ▶


Infos **Stats**

Career (All) ▾ Surfaces (All) ▾

↻

Serve		Return	
Aces	7163	1st Serve Return Points Won	34%
Double Faults	2918	2nd Serve Return Points Won	55%
1st Serve	65%	Break Points Opportunities	11183
1st Serve Points Won	74%	Break Points Converted	44%
2nd Serve Points Won	56%	Return Games Played	15506
Break Points Faced	6481	Return Games Won	32%
Break Points Saved	65%	Return Points Won	42%
Service Games Played	15952	Total Points Won	54%
Service Games Won	86%		
Total Service Points Won	68%		

A) Comment les «scraper»(ie récolter) sur une page web ?

 **NOVAK DJOKOVIC** | Players ▾


[Activity](#) [Win/Loss](#) [Titles and Finals](#)

Player activity

Singles ▾

Career ▾

Tourn(All) ▾



Singles	1101-219 W-L	98 Titles	\$162,103,739 Prize Money
----------------	------------------------	---------------------	-------------------------------------

Points: 0, ATP Ranking: 1, Prize Money: €0

B) «Parser» les données («Data Parsing»)

Définition

L'analyse syntaxique de données (« Data parsing ») est le processus de transformation d'un format à un autre. Il sert à **structurer les données** récoltées: ie convertir des données non structurées en données structurées.

Dans notre cas:



Module : **BeautifulSoup** (Bs4) pour trouver les informations que l'on souhaite extraire dans les bons conteneurs

Facilite l'extraction car le **code HTML est instancié dans les objets Bs4** : les fonctions Bs4 permettent donc d'extraire directement les contenus des conteneurs

C) Mise en place de dictionnaire exploitable (Filtration des données)

Après cela, les dictionnaires sont de la forme suivante:

Problème: les valeurs associées à chaque clé sont en string et donc non exploitables

```
{'Activity Stats': {'Clay': '0.500',  
  'Grass': '0.000',  
  'Hard': '0.688',  
  'vs. Left Handers': '0.667',  
  'vs. Right Handers': '0.647'},  
'Career Stats': {'1st Serve': '58%',  
  '1st Serve Points Won': '72%',  
  '1st Serve Return Points Won': '24%',  
  '2nd Serve Points Won': '51%',  
  '2nd Serve Return Points Won': '50%',  
  'Aces': '166',  
  'Break Points Converted': '42%',  
  'Break Points Faced': '135',  
  'Break Points Opportunities': '93',  
  'Break Points Saved': '67%',  
  'Double Faults': '56',  
  'Return Games Played': '238',  
  'Return Games Won': '16%',  
  'Return Points Won': '34%',  
  'Service Games Played': '237',  
  'Service Games Won': '81%',  
  'Total Points Won': '49%',  
  'Total Service Points Won': '63%'},  
'Personnal Details': {'Age': '18 (2005.09.01)',  
  'Birthplace': 'Prostejov',  
  'Coach': 'Tomas Josefus',  
  'Country': 'Czechia',  
  'Follow player': '',  
  'Height': '6'4" (193cm)',  
  'Plays': 'Right-Handed, Two-Handed Backhand',  
  'Turned pro': '2022',  
  'Weight': '184 lbs (83kg)'},  
'YTD&Career Player Stats': {'Career': {'Loses': 7,  
  'Rank': 65,  
  'Titles': 0,  
  'Wins': 13},  
  'YTD': {'Loses': 6,  
  'Rank': 76,  
  'Titles': 0,  
  'Wins': 10}}}
```


C) Mise en place de dictionnaire exploitable (Filtration des données)

```
import re

def filter_player(infos: dict) -> dict:
    ...hands_infos = infos["Personal Details"]["Plays"].split(", ")
    ...hand = "Right-Handed" if "Right-Handed" in hands_infos else "Left-Handed"
    ...surface_dict = infos["Activity Stats"]["Surface"]

    ...infos_f = {"Rank (YTD)": int(infos["YTD&Career Player Stats"]["YTD"]["Rank"]),
    ...            "Titles (Career)": int(infos["YTD&Career Player Stats"]["Career"]["Titles"]),
    ...            "Wins (YTD)": int(infos["YTD&Career Player Stats"]["YTD"]["Wins"]),
    ...            "Loses (YTD)": int(infos["YTD&Career Player Stats"]["YTD"]["Loses"]),
    ...            "Wins (Career)": int(infos["YTD&Career Player Stats"]["Career"]["Wins"]),
    ...            "Loses (Career)": int(infos["YTD&Career Player Stats"]["Career"]["Loses"]),
    ...            "Preferred Surface": max(surface_dict, key=surface_dict.get),
    ...            "Dominant Hand": hand,
    ...            "1st Serve Points Won": int(infos["Career Stats"]["1st Serve Points Won"][:-1]),
    ...            "1st Serve Return Points Won": int(infos["Career Stats"]["1st Serve Return Points Won"][:-1]),
    ...            "2nd Serve Points Won": int(infos["Career Stats"]["2nd Serve Points Won"][:-1]),
    ...            "2nd Serve Return Points Won": int(infos["Career Stats"]["2nd Serve Return Points Won"][:-1]),
    ...            "Break Points Converted": int(infos["Career Stats"]["Break Points Converted"][:-1]),
    ...            "Break Points Saved": int(infos["Career Stats"]["Break Points Saved"][:-1]),
    ...            "Clay Index (Career)": float(infos["Activity Stats"]["Surface"]["Clay"]),
    ...            "Grass Index (Career)": float(infos["Activity Stats"]["Surface"]["Grass"]),
    ...            "Hard Index (Career)": float(infos["Activity Stats"]["Surface"]["Hard"]),
    ...            "VS Right-Handers Index (Career)": float(infos["Activity Stats"]["vs Hands"]["vs. Right Handers*"]),
    ...            "VS Left-Handers Index (Career)": float(infos["Activity Stats"]["vs Hands"]["vs. Left Handers*"]),
    ...            }
    ...return infos_f

def player_name(url: str) -> str:
    ...extract_name = re.search(r"/players/([a-zA-Z-]+[a-zA-Z-]+)/", url)
    ...names = extract_name.group(1).split("-")
    ...return f"{names[-1][0].upper()}{names[-1][1:].lower()} {names[0][0].upper()}."
```

On transforme tous les strings en entier, flottants pour rendre les données présentes dans les dictionnaires exploitables

On met en forme les noms des joueurs présents dans les URLs

```
"/en/players/tomas-martin-etcheverry/ea24/",
```

Ex: on a Etcheverry M.

C) Mise en place de dictionnaire exploitable (Filtration des données)

► Dans le premier dictionnaire:

```
Info_joueurs = { [nom du joueur] : { [rang] : ...,  
                                     [nombre de titres] : ...,  
                                     [victoires] : ...,  
                                     [défaites]: ...,  
                                     [surface favorite]: ...,  
                                     etc...  
               },  
               [nom du joueur] : etc...
```

► Dans le deuxième dictionnaire:

```
Matches = { [référence du match] : { [ date] : ...,  
                                     [ surface] : ...,  
                                     [joueur n°1 ]: ...,  
                                     [joueur n°2]: ..., [  
                                     cote n°1] : ...,  
               },  
               [référence du match] : etc...
```

► Dans le troisième dictionnaire:

```
Info_cotes = { [référence du match] : { [1er site de pari] : {  
                                     [cote du joueur n°1] : ...,  
                                     [cote du joueur n°2] : ...  
                                     },  
                                     [2e site de pari] : {  
                                     [cote du joueur n°1] : ...,  
                                     [cote du joueur n°2] : ...  
                                     },  
                                     [dernier site de pari] : {  
                                     [cote du joueur n°1] : ...,  
                                     [cote du joueur n°2] : ...  
                                     }  
               },  
               Etc..  
               [référence du match] : etc...
```

D) Sauvegarde des données dans des fichiers Json: sous forme de dictionnaires

```
{
  "Djokovic N.": {
    "Rank (YTD)": 1,
    "Titles (Career)": 98,
    "Wins (YTD)": 12,
    "Loses (YTD)": 5,
    "Wins (Career)": 1099,
    "Loses (Career)": 218,
    "Preferred Surface": "Grass",
    "Dominant Hand": "Right-Handed",
    "1st Serve Points Won": 74,
    "1st Serve Return Points Won": 34,
    "2nd Serve Points Won": 56,
    "2nd Serve Return Points Won": 55,
    "Break Points Converted": 44,
    "Break Points Saved": 65,
    "Clay Index (Career)": 0.801,
    "Grass Index (Career)": 0.858,
    "Hard Index (Career)": 0.847,
    "VS Right-Handers Index (Career)": 0.845,
    "VS Left-Handers Index (Career)": 0.768
  },
  "Sinner J.": {
```

info joueur

```
{
  "2024C-MiHa0": {
    "Date": 2024,
    "Surface": "Clay",
    "Player 1": "Misolic F.",
    "Player 2": "Halys Q.",
    "Odds Player 1": 2.24,
    "Odds Player 2": 1.62,
    "Score Player 1": 2,
    "Score Player 2": 1
  },
  "2024C-BaKu1": {
    "Date": 2024,
    "Surface": "Clay",
    "Player 1": "Barrere G.",
    "Player 2": "Kudla D.",
    "Odds Player 1": 1.28,
    "Odds Player 2": 3.54,
    "Score Player 1": 2,
    "Score Player 2": 1
  },
```

matches

```
{
  "xYXGult3": {
    "10x10bet": {
      "Odds Player 1": 1.14,
      "Odds Player 2": 5.25
    },
    "1xBet": {
      "Odds Player 1": 1.15,
      "Odds Player 2": 5.55
    },
    "Alphabet": {
      "Odds Player 1": 1.15,
      "Odds Player 2": 5.5
    },
    "bet-at-home": {
      "Odds Player 1": 1.16,
      "Odds Player 2": 5.0
    },
    "bet365": {
      "Odds Player 1": 1.14,
      "Odds Player 2": 5.5
    },
    "BetInAsia": {
      "Odds Player 1": 1.16,
      "Odds Player 2": 6.01
    },
    "GGBET": {
      "Odds Player 1": 1.18,
      "Odds Player 2": 5.45
    },
    "Lasbet": {
      "Odds Player 1": 1.15,
      "Odds Player 2": 5.5
    },
    "Pinnacle": {
      "Odds Player 1": 1.17,
      "Odds Player 2": 6.01
    },
    "Unibet": {
      "Odds Player 1": 1.16,
      "Odds Player 2": 5.4
    },
    "VOBET": {
      "Odds Player 1": 1.15,
      "Odds Player 2": 5.5
    },
    "Vulkan Bet": {
      "Odds Player 1": 1.18,
      "Odds Player 2": 5.45
    },
    "William Hill": {
      "Odds Player 1": 1.14,
      "Odds Player 2": 5.5
    }
  },
```

Info cotes

Total: 312 joueurs , 1108 matches

A) Définition mathématique

Définition

Une **filtration** \mathcal{A} est une suite croissante $F = \{\mathcal{F}_n\}_{n \geq 0}$ de sous-tribus de \mathcal{A}

$$\mathcal{F}_0 \subset \mathcal{F}_1 \subset \mathcal{F}_2 \subset \dots \subset \mathcal{A}$$

On dit que $(\Omega, \mathcal{A}, F, P)$ est un **espace probabilisé filtré**

En particulier, si $\{X_n\}_{n \geq 0}$ est un processus aléatoire, alors la suite

$$\mathcal{F}_n = \sigma(X_i, i \leq n) \text{ avec } n \geq 0$$

est appelée **filtration naturelle du processus**

Définition

Soit $X = \{X_n\}_{n \geq 0}$ un processus adapté à l'espace probabilisé filtré $(\Omega, \mathcal{A}, \mathcal{F}, P)$

Si X_n est intégrable (ie $\mathbb{E}(|X_n|) < +\infty$)

- une ***martingale*** si

$$\mathbb{E}(X_n | \mathcal{F}_{n-1}) = X_{n-1} \text{ avec } n \geq 1$$

- une ***sous-martingale*** si

$$\mathbb{E}(X_n | \mathcal{F}_{n-1}) \geq X_{n-1} \text{ avec } n \geq 1$$

- Une martingale est un processus aléatoire dont l'espérance conditionnelle par rapport au passé reste constante

Définition

Soit X, Y, Z trois variables aléatoires à valeurs dans E **dénombrable**:

$$X: \Omega \rightarrow E$$

$$Y: \Omega \rightarrow E$$

$$Z: \Omega \rightarrow E$$

On appelle **espérance conditionnelle de X sachant Y** :

$$\mathbb{E}(X|Y) = \sum_{x \in E} x \mathbb{P}(X = x | Y = y) = \frac{\mathbb{E}(X \times \mathbb{I}_{\{Y=y\}})}{\mathbb{P}(Y = y)}$$

Propriété

L'espérance conditionnelle est un **opérateur linéaire**.

- ▶ **Objectif de gain identique**
- ▶ **Gestion du risque** assez différente selon la martingale employée
- ▶ **Exploitations des opportunités** (dans les marchés de paris):
 - ➡ **Martingale**: trouver des événements à parier de manière répétitive
 - ➡ **Arbitrage**: trouver des différences significatives de cotes

Modélisation: Marche aléatoire dans \mathbb{R}

Soit $\{\xi_n\}_{n \geq 1}$ une suite de variables aléatoires dans \mathbb{R} modélisant **le gain ou la perte au n-ième pari**

Soit $S = \{S_n\}_{n \geq 0}$ une suite de variables aléatoire modélisant la **fortune du joueur après le n-ième pari**

L'issue d'un match est incertaine donc on suppose que $\mathbb{E}(\xi_n) = 0$

La **marche aléatoire** $S = \{S_n\}_{n \geq 0}$ telle que:

$$S_n = \sum_{i=1}^n \xi_i \text{ avec } S_0 = 0 \text{ est une martingale}$$

pour la filtration $\sigma(\xi_i, i \leq n)$

$$\mathbb{E}(S_{n+1}|\mathcal{F}_n) = \mathbb{E}(S_n + \xi_{n+1} | \mathcal{F}_n) = \mathbb{E}(S_n|\mathcal{F}_n) + \mathbb{E}(\xi_{n+1}|\mathcal{F}_n) = S_n + \mathbb{E}(\xi_{n+1})$$

Or, **dans le cas d'un arbitrage**: $\mathbb{E}(\xi_{n+1}) \geq 0$ (on récupère **au moins** sa somme mise)

$$\text{d'où: } \mathbb{E}(S_{n+1}|\mathcal{F}_n) \geq S_n$$

Donc la suite S est une **sous-martingale**

Le joueur peut-il espérer gagner de l'argent s'il sait s'arrêter au bon moment ?

- ▶ Le nombre de partie qu'il va jouer sera un nombre aléatoire T

D) Analyse Asymptotique

Dans le cas de la stratégie: le joueur doit gagner une fois pour atteindre la somme voulue

Somme de départ: 1000€

La mise n'est pas constante : elle **dépend de la cote et du bénéfice souhaité**

On peut utiliser la notion de **temps d'arrêt**.

Définition

Soit (Ω, \mathcal{A}, P) un espace probabilisé muni d'une filtration $\{\mathcal{F}_n\}_{n \geq 0}$

Une variable aléatoire est appelée **temps d'arrêt** si

$$T: \Omega \rightarrow \mathbb{N} \cup \{+\infty\}$$

$$\{T \leq n\} \in \mathcal{F}_n$$

Définition

La martingale **S arrêtée au temps T** est définie par:

$$S_n^T = \begin{cases} S_n & \text{si } n \leq T \\ S_T & \text{si } n \geq T \end{cases}$$

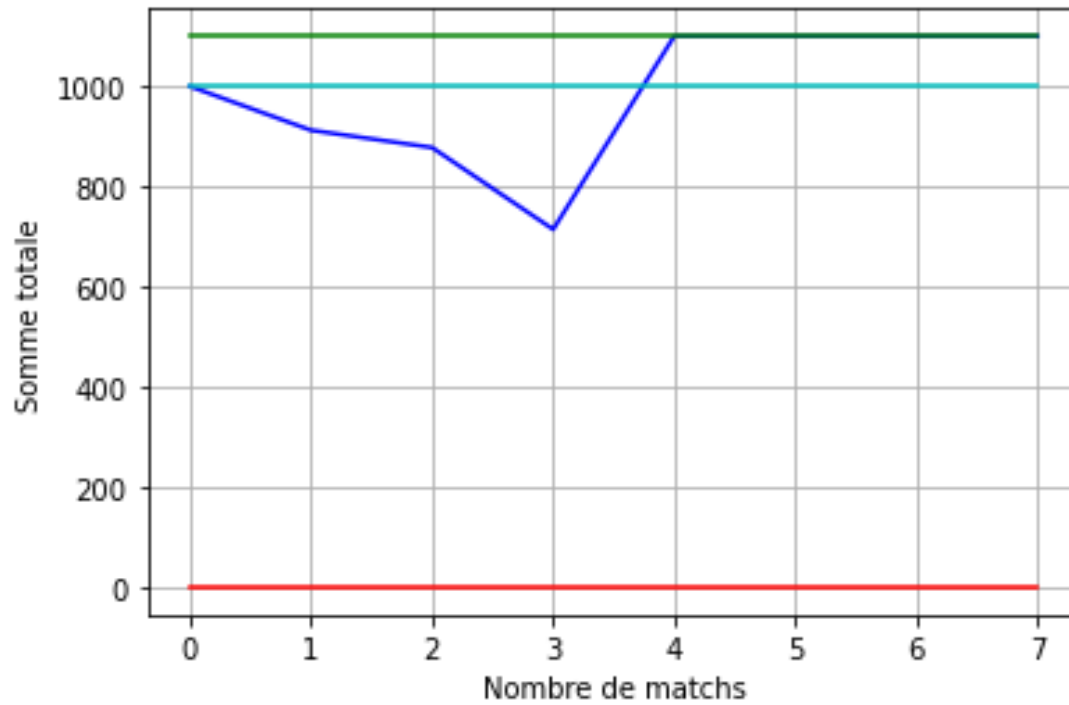
Théorème d'arrêt de Doob

Soit $S = \{S_n\}_{n \geq 0}$ une sous-martingale et T un temps d'arrêt bornée sur $(\Omega, \mathcal{A}, \mathcal{F}, P)$ alors:

$$\mathbb{E}(S_T) = \mathbb{E}(S_0) = 0$$

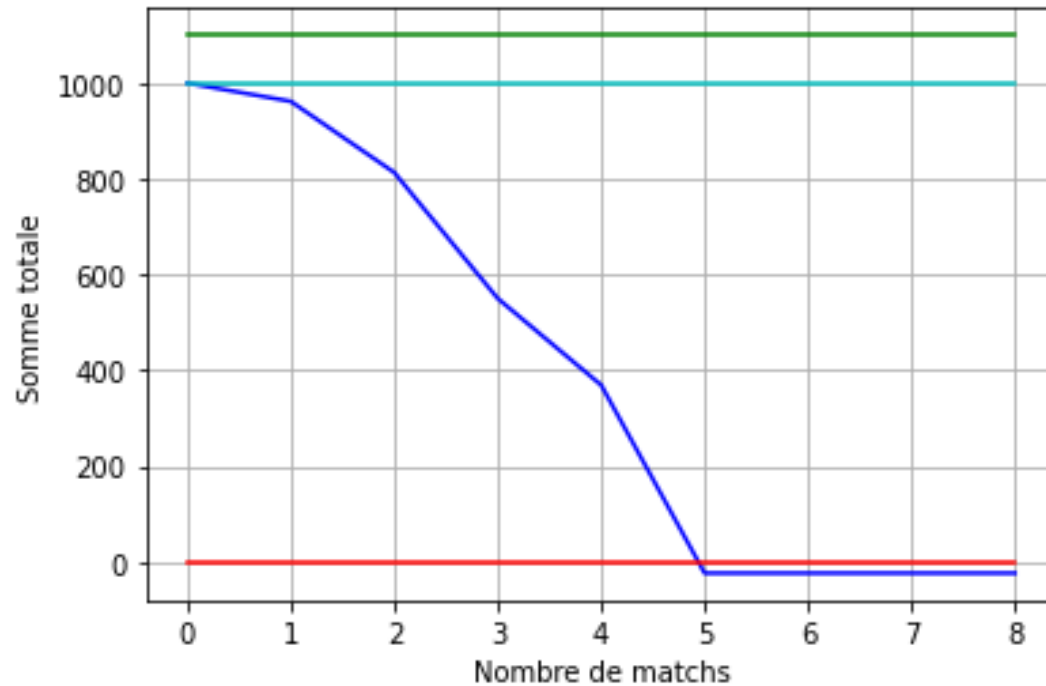
Ce théorème est bien cohérent avec une stratégie d'arbitrage

D) Analyse Asymptotique



Cela nécessite d'utiliser un temps d'arrêt: $T = \inf \{n \geq 0, S_n = 1100\}$

D) Analyse Asymptotique



Parfois, le temps d'arrêt **n'est jamais atteint donc non bornée**: le joueur est ruinée avant

Existe-t-il un autre temps d'arrêt pouvant être atteint ?

► Temps d'atteintes de barrière :

$$T_{0,bénéfice} = \inf\{n \geq 0, S_n \in \{0, fortune\ initial + bénéfice\}\}$$

On évalue le temps d'arrêt selon la valeur du bénéfice dans le cadre de la même stratégie:

Bénéfice n° 1: 1€

Bénéfice n° 2: 10 €

Bénéfice n° 3: 50€

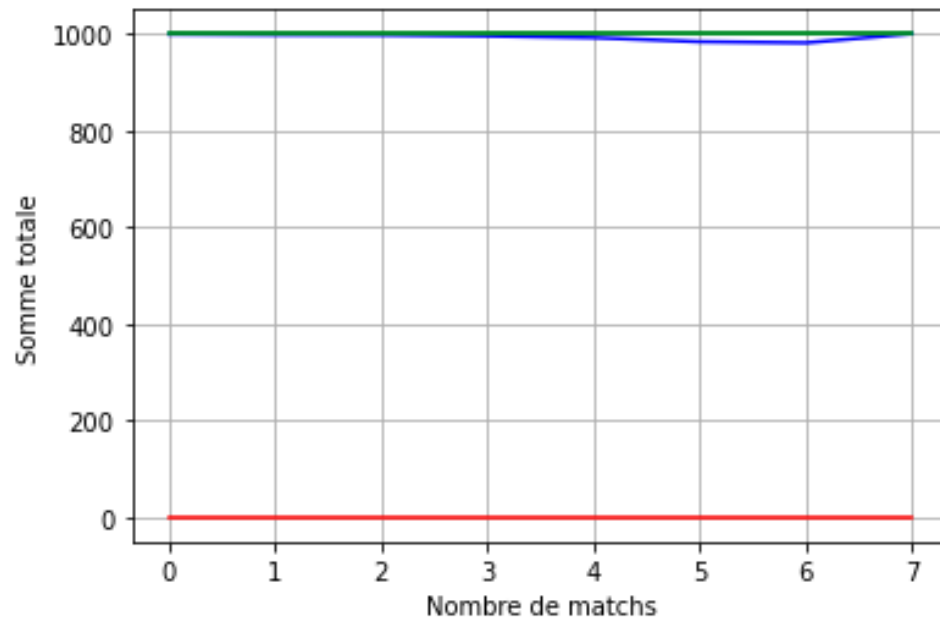
Bénéfice n° 4: 100€

D) Analyse Asymptotique

► Bénéfice n°1

Sur 30 tests, le temps d'arrêt est toujours borné.

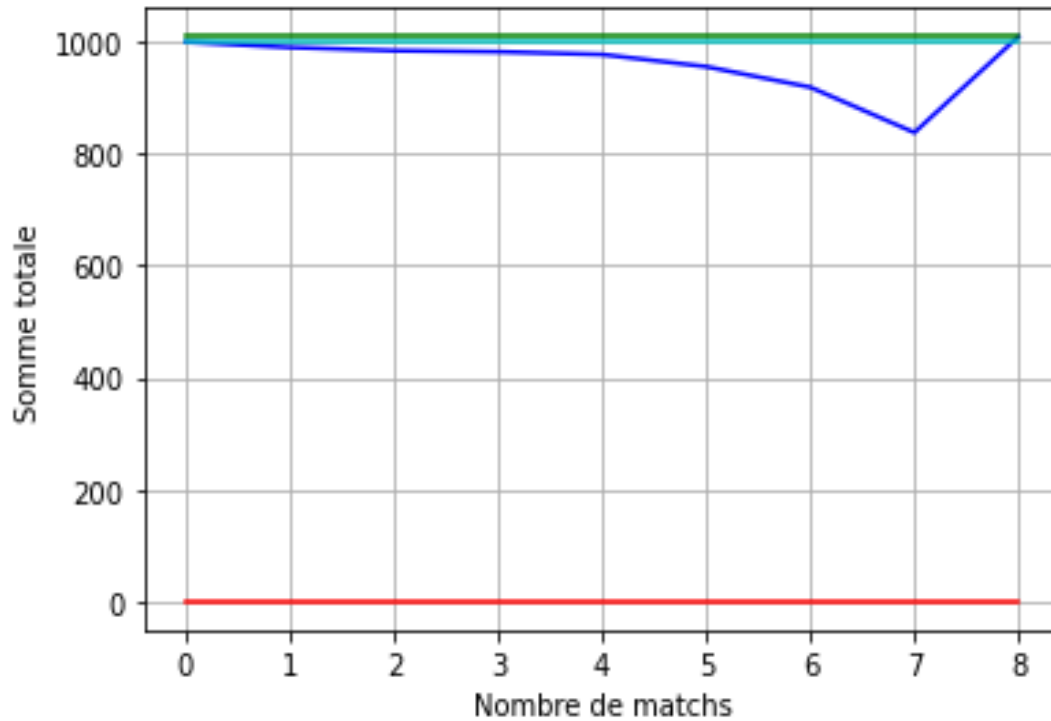
Pire cas ci-dessous:



D) Analyse Asymptotique

► Bénéfice n°2

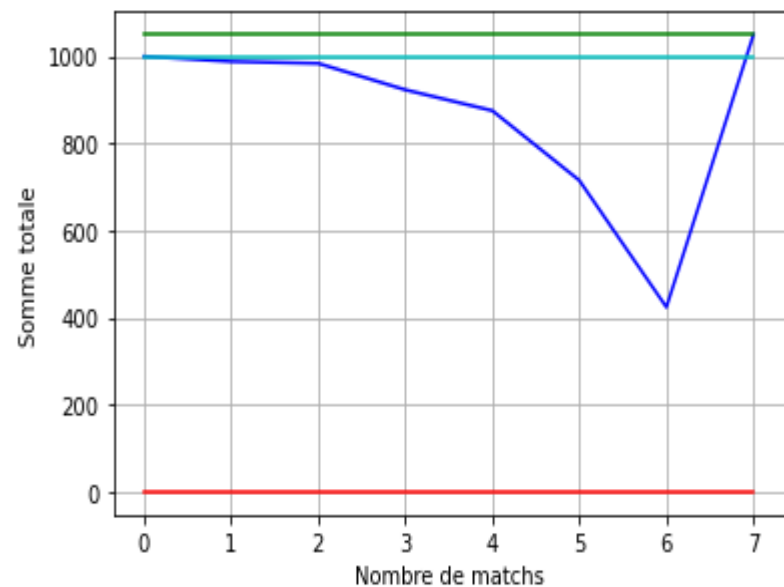
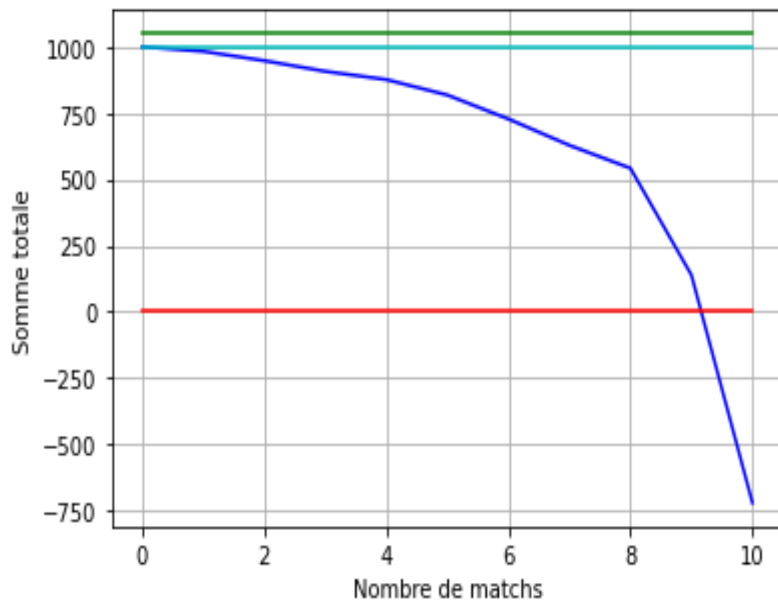
Le temps est **toujours bornée mais plus important** que précédemment.



D) Analyse Asymptotique

► Bénéfice n°3:

Sur 30 tests, **26 aboutissent** à un temps d'arrêt borné et 4 à une ruine du joueur

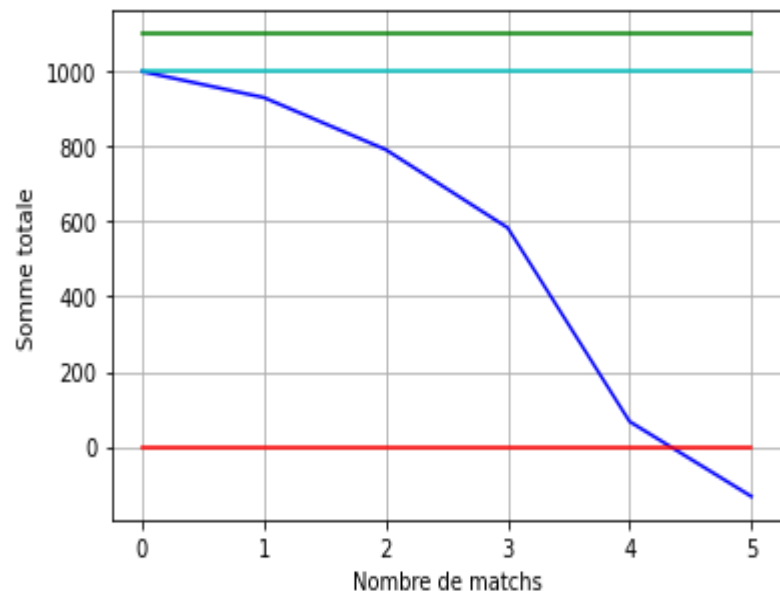
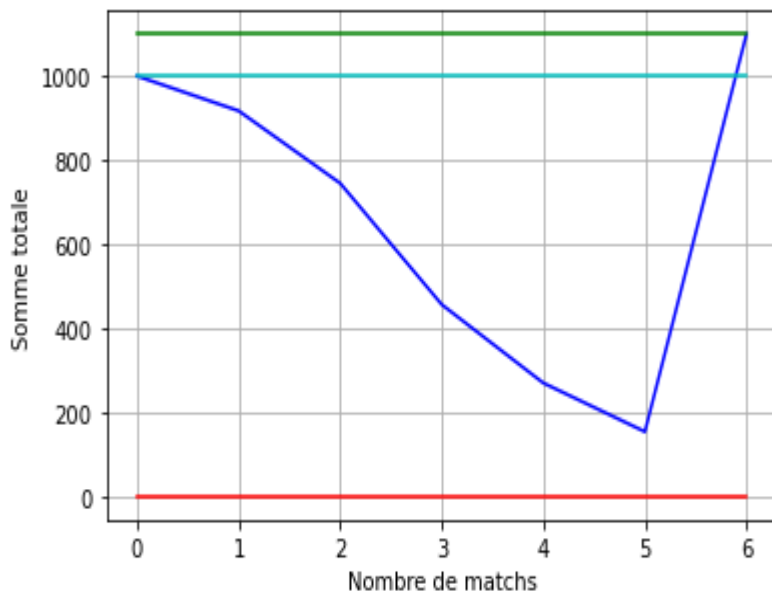


Observation: Temps d'arrêt inférieur ou égal à 7
Ruine assez lente

D) Analyse Asymptotique

► Bénéfice n°4

Sur 30 tests, **20 aboutissent** à un temps d'arrêt borné et 10 à une ruine du joueur



Observation: Temps d'arrêt inférieur ou égal à 6
Ruine plus rapide

Théorème: Inégalité maximale de Doob

Soit une sous-martingale positive. Pour tout $a > 0$ et $n \in \mathbb{N}$

$$\mathbb{P}(\max_{0 \leq k \leq n} S_k \geq a) \leq \frac{\mathbb{E}(S_n)}{a}$$

Dans notre cas, « a » est notre somme initiale additionnée au bénéfice

Loi des grands nombres due à Kolmogorov

$$\lim_{n \rightarrow +\infty} \frac{S_n}{n} = \mathbb{E}(\xi)$$

Dans le cas d'un arbitrage, l'espérance est positive

CONCLUSION

- ▶ Stratégie peu utilisable donc plus de chance de perdre que de gagner car la sous-martingale est peu existante dans le cas d'une situation de pari sportif
- ▶ Le gain ne sera jamais conséquent, d'après l'inégalité maximale de Doob car dans le cas d'un bénéfice élevé:

$$\mathbb{P}(\max_{0 \leq k \leq n} S_k \geq a) \leq \frac{\mathbb{E}(S_n)}{a} \ll 1$$

- ▶ D'après la loi des grands nombres due à Kolmogorov, cela signifie qu'en cas de jeu défavorable, le joueur est absolument assuré de perdre sur le long terme.

FIN
Merci de votre attention

Annexe: tracés

```
31 def curve_risky(benefice):
32     values = risky(benefice)
33     Y = np.array(values)
34     X = np.array([i for i in range(len(values))])
35     Y0 = np.array([0 for i in range(len(values))])
36     Yinit = np.array([1000 for i in range(len(values))])
37     Ygain = np.array([(1000 + benefice) for i in range(len(values))])
38     plt.plot(X, Y, 'b')
39     plt.plot(X, Y0, 'r')
40     plt.plot(X, Yinit, 'c')
41     plt.plot(X, Ygain, 'g')
42     plt.xlabel("Nombre de matchs")
43     plt.ylabel("Somme totale")
44     plt.grid()
45     plt.show()
```

Annexe: tracés

```
81
82 def curve_marge():
83     values = liste_marge()
84     average = 0
85     for i in values:
86         average += i
87     average /= len(values)
88     Y = np.array(values)
89     X = np.array([i for i in range(len(values))])
90     Y0 = np.array([0 for i in range(len(values))])
91     Ymoy = np.array([average for i in range(len(values))])
92     plt.plot(X, Y, 'r')
93     plt.plot(X, Y0, 'r')
94     plt.plot(X, Ymoy, 'g', linewidth=5)
95     plt.xlabel("Nombre de matchs")
96     plt.ylabel("Marges")
97     plt.title("Marge moyenne pour chaque match")
98     plt.grid()
99     plt.show()
100
```


Annexe: tracés

```
103 def curve_arbitrage():
104     values = liste_arbitrage()
105     average = 0
106     for i in values:
107         average += i
108     average /= len(values)
109     Y = np.array(values)
110     Y0 = np.array([0 for i in range(len(values))])
111     X = np.array([i for i in range(len(values))])
112     Ymoy = np.array([average for i in range(len(values))])
113     plt.plot(X, Y, 'o')
114     plt.plot(X, Y0, 'r')
115     plt.plot(X, Ymoy, 'g', linewidth=5)
116     plt.xlabel("Nombre de matchs")
117     plt.ylabel("Marges")
118     plt.title("Marge minimale pour chaque match")
119     plt.grid()
120     plt.show()
121
```

Annexe: scraper atp

```
73 def scrape_atp(url: str) -> dict:
74     driver = webdriver.Chrome()
75     driver.get(f"https://www.atptour.com{url}overview")
76     html_overview = driver.page_source
77     driver.quit()
78
79     driver = webdriver.Chrome()
80     driver.get(f"https://www.atptour.com{url}player-stats?year=all&surface=all")
81     html_stats = driver.page_source
82     driver.quit()
83
84     driver = webdriver.Chrome()
85     driver.get(f"https://www.atptour.com{url}atp-win-loss?tourType=Tour")
86     html_activity = driver.page_source
87     driver.quit()
88
89     soup1 = BeautifulSoup(html_overview, 'html.parser')
90     soup2 = BeautifulSoup(html_stats, 'html.parser')
91     soup3 = BeautifulSoup(html_activity, 'html.parser')
92
93     atp_player_stats = soup1.find('div', class_='atp_player-stats')
94     career_stats = soup2.find('div', class_='statistics_content')
95     activity_stats = soup3.find('div', class_='atp_player-win_loss-index')
96     infos = {}
97
98     infos["Personnal Details"] = scrape_personal_details(soup1)
99     infos["YTD&Career Player Stats"] = scrape_atp_player_stats(atp_player_stats) if atp_player_stats else 0
100     infos["Career Stats"] = scrape_career_stats(career_stats) if career_stats else 0
101     infos["Activity Stats"] = scrape_activity_stats(activity_stats) if activity_stats else 0
102
103     return infos
```

Annexe: scraper atp

```
11 def scrape_atp_player_stats(datas: BeautifulSoup) -> dict:
12     player_stats = {
13         'YTD': {},
14         'Career': {}
15     }
16
17     all_stats_details = datas.find_all('div', class_='player-stats-details')
18
19     for stat_detail in all_stats_details:
20         type_stat = stat_detail.find('div', class_='type').text.strip()
21
22         rank = stat_detail.find('div', class_='stat').text
23         w_l = stat_detail.find('div', class_='wins').text
24         titles = stat_detail.find('div', class_='titles').text
25
26         stats = {
27             'Wins': int(w_l.split()[0]),
28             'Loses': int(w_l.split()[2]),
29             'Rank': int(rank.split()[0]),
30             'Titles': int(titles.split()[0])
31         }
32
33         if type_stat == 'YTD':
34             player_stats['YTD'].update(stats)
35         elif type_stat == 'Career':
36             player_stats['Career'].update(stats)
37
38     return player_stats
```

Annexe: scraper atp

```
1 from bs4 import BeautifulSoup
2 from selenium import webdriver
3 import time
4
5
6 def scrape_personal_details(datas: BeautifulSoup) -> dict:
7     hands = {"Plays": datas.find('span', text='Plays').find_next_sibling('span').text}
8     return hands
9
```

```
41 def scrape_career_stats(datas: BeautifulSoup) -> dict:
42     datas_extraites = {}
43
44     stats = datas.find_all('li', class_='stats_items')
45
46     for stat in stats:
47         categorie = stat.find('span', class_='stats_record').text
48         valeur = stat.find('span', class_='stats_percentage').text
49         datas_extraites[categorie] = valeur
50
51     return datas_extraites
```

Annexe: scraper atp

```
54 def scrape_activity_stats(datas: BeautifulSoup) -> dict:
55     ... hands = {}
56     ... surface = {}
57
58     ... table = datas.find('table')
59     ... rows = table.find_all('tr')
60
61     ... for row in rows:
62         ... cells = row.find_all('th') + row.find_all('td')
63         ... if cells[0].get_text(strip=True) in ['Clay', 'Grass', 'Hard']:
64             ... career_index = cells[4].get_text(strip=True)
65             ... surface[cells[0].get_text(strip=True)] = float(career_index)
66         ... elif cells[0].get_text(strip=True) in ['vs. Right Handers*', 'vs. Left Handers*']:
67             ... career_index = cells[4].get_text(strip=True)
68             ... hands[cells[0].get_text(strip=True)] = float(career_index)
69
70     ... return {"vs. Hands": hands, "Surface": surface}
```

Annexe: scraper oddsportal

```
106 def scrape_oddsportal(url: str, year: int, surface: str) -> dict:
107     driver = webdriver.Chrome()
108     driver.get(url)
109     time.sleep(0.5)
110     driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
111     time.sleep(1)
112     html_oddsportal = driver.page_source
113     driver.quit()
114
115     soup = BeautifulSoup(html_oddsportal, 'html.parser')
116
117     match_div = soup.find_all('div', class_='group flex')
118     infos = {}
119
120     for match in match_div:
121         players = match.find_all('p', class_='participant-name')
122         player1 = players[0].text.strip().split()
123         player2 = players[1].text.strip().split()
124
125         odds = match.find_all('p', class_='height-content')
126
127         odds1 = float(odds[0].text.strip())
128         odds2 = float(odds[1].text.strip())
129
130         score_parts = match.find('div', class_='flex gap-1 font-bold font-bold')
131         if score_parts:
132             scores = score_parts.text.strip().split('-')
133             score1 = scores[0].strip()
134             score2 = scores[1].strip()
135             infos[f"{year}{surface[0]}-{player1[0][:2]}{player2[0][:2]}{len(infos)}"] = {
136                 "Date": year,
137                 "Surface": surface,
138                 "Player 1": f"{player1[0]}-{player1[-1]}",
139                 "Player 2": f"{player2[0]}-{player2[-1]}",
140                 "Odds Player 1": float(odds1),
141                 "Odds Player 2": float(odds2),
142                 "Score Player 1": int(score1) if score1 != '' else '',
143                 "Score Player 2": int(score2) if score2 != '' else ''
144             }
145         else:
146             continue
147
148     return infos
```


Annexe: scraper oddsportal

```
150 def scrape_allodds(url: str) -> dict:
151     ... driver = webdriver.Chrome()
152     ... driver.get(url)
153     ... html_allodds = driver.page_source
154     ... driver.quit()
155
156     ... soup = BeautifulSoup(html_allodds, 'html.parser')
157
158     ... matches = soup.find_all('div', class_='border-black-borders flex h-9 border-b border-l border-r text-xs')
159     ... infos = {}
160
161     ... for match in matches:
162         ... bookmaker_tag = match.find('p', class_='height-content')
163         ... bookmaker = bookmaker_tag.text.strip()
164
165         ... odds = match.find_all('p', class_='height-content')
166         ... odds1 = float(odds[1].text.strip())
167         ... odds2 = float(odds[2].text.strip())
168
169         ... infos[bookmaker] = {'Odds Player 1': odds1, 'Odds Player 2': odds2}
170
171     ... return infos
```

Annexe: parser

```
1 from tpe_save import read_json
2
3
4 def get_match(player: str, surface=("Grass", "Clay", "Hard"), only_last=False) -> dict:
5     matches = read_json('save_matches.json')
6     played_matches = {}
7
8     for id, match in matches.items(): #lit le id associé à la valeur du match en question
9         if player in (match["Player 1"], match["Player 2"]) and match["Surface"] in surface:
10             played_matches[id] = match
11
12     if played_matches:
13         if only_last:
14             last_id = sorted(played_matches.keys())[-1]
15             # sortir dans l'ordre croissant, dernier année (prendra un match forcément en terre battu car 'Clay')
16             return {last_id: played_matches[last_id]}
17         return played_matches
18
19     return None
```


Annexe: save Json

```
1 import json
2 import logging
3
4
5 def save_json(infos: dict, force=False, file='save.json') -> None:
6     ...if force:
7         ...with open(file, 'w') as f: # ouvre le fichier + écriture write
8             ...json.dump(infos, f, indent=4)
9             ...# on importe la librairie, et dump ça réécrit dans le fichier f, indentation pour gérer l'espace
10            ...return None
11
12    ...try:
13        ...with open(file, 'r') as f: # lecture
14            ...data = json.load(f) # pour le charger
15        ...except (FileNotFoundError, json.JSONDecodeError): # s'il y a une erreur
16            ...data = {} # d'ou le dico vide
17
18    ...data.update(infos)
19
20    ...with open(file, 'w') as f:
21        ...json.dump(data, f, indent=4) # on réécrit comme en haut
22
23
24 def read_json(file='save.json') -> dict:
25     ...with open(file, 'r') as f:
26         ...data = json.load(f)
27     ...return data
28
29
30 def is_player_in_json(player_name: str, file='save.json') -> bool:
31     ...try:
32         ...with open(file, 'r') as f:
33             ...return player_name in json.load(f)
34         ...# vérifie dans si ton joueur est dans ton fichier, toujours sous forme de dico !
35     ...except:
36         ...return False # cas où il y a un problème de lecture
37
38
39 logging.basicConfig(filename='log_player.txt', level=logging.INFO, format='%(asctime)s - %(message)s')
40 #pour avoir un journal de tout ce qui s'est passé
41
42
43 def log_player_status(name: str, status: str = 'waiting'):
44     ...logging.info(f'{name} - Status: {status}') # vérification dans le fichier log
45
```

Annexe: configuration(logging)

```
1 2024-06-01 18:26:21,968 - Djokovic N. - Status: waiting
2 2024-06-01 18:26:21,970 - Djokovic N. - Status: ok (already scraped)
3 2024-06-01 18:26:21,970 - Sinner J. - Status: waiting
4 2024-06-01 18:26:21,972 - Sinner J. - Status: ok (already scraped)
5 2024-06-01 18:26:21,972 - Alcaraz C. - Status: waiting
6 2024-06-01 18:26:21,974 - Alcaraz C. - Status: ok (already scraped)
7 2024-06-01 18:26:21,974 - Zverev A. - Status: waiting
8 2024-06-01 18:26:21,976 - Zverev A. - Status: ok (already scraped)
9 2024-06-01 18:26:21,976 - Medvedev D. - Status: waiting
10 2024-06-01 18:26:21,978 - Medvedev D. - Status: ok (already scraped)
11 2024-06-01 18:26:21,978 - Rublev A. - Status: waiting
12 2024-06-01 18:26:21,980 - Rublev A. - Status: ok (already scraped)
13 2024-06-01 18:26:21,980 - Ruud C. - Status: waiting
14 2024-06-01 18:26:21,981 - Ruud C. - Status: ok (already scraped)
15 2024-06-01 18:26:21,981 - Hurkacz H. - Status: waiting
16 2024-06-01 18:26:21,983 - Hurkacz H. - Status: ok (already scraped)
17 2024-06-01 18:26:21,983 - Tsitsipas S. - Status: waiting
18 2024-06-01 18:26:21,985 - Tsitsipas S. - Status: ok (already scraped)
19 2024-06-01 18:26:21,985 - Dimitrov G. - Status: waiting
20 2024-06-01 18:26:21,987 - Dimitrov G. - Status: ok (already scraped)
21 2024-06-01 18:26:21,987 - Minaur A. - Status: waiting
22 2024-06-01 18:26:21,989 - Minaur A. - Status: ok (already scraped)
23 2024-06-01 18:26:21,989 - Fritz T. - Status: waiting
24 2024-06-01 18:26:21,990 - Fritz T. - Status: ok (already scraped)
25 2024-06-01 18:26:21,990 - Rune H. - Status: waiting
26 2024-06-01 18:26:21,993 - Rune H. - Status: ok (already scraped)
27 2024-06-01 18:26:21,993 - Paul T. - Status: waiting
28 2024-06-01 18:26:21,995 - Paul T. - Status: ok (already scraped)
29 2024-06-01 18:26:21,995 - Shelton B. - Status: waiting
30 2024-06-01 18:26:21,997 - Shelton B. - Status: ok (already scraped)
31 2024-06-01 18:26:21,997 - Jarry N. - Status: waiting
32 2024-06-01 18:26:21,998 - Jarry N. - Status: ok (already scraped)
33 2024-06-01 18:26:21,998 - Humbert U. - Status: waiting
34 2024-06-01 18:26:22,000 - Humbert U. - Status: ok (already scraped)
35 2024-06-01 18:26:22,000 - Khachanov K. - Status: waiting
36 2024-06-01 18:26:22,003 - Khachanov K. - Status: ok (already scraped)
37 2024-06-01 18:26:22,003 - Bublik A. - Status: waiting
38 2024-06-01 18:26:22,004 - Bublik A. - Status: ok (already scraped)
39 2024-06-01 18:26:22,004 - Baez S. - Status: waiting
```

Annexe: configuration(URL)

```
1 oddsportal_url_clay = {2024: ["https://www.oddsportal.com/tennis/france/atp-french-open/results/#",
2                               "https://www.oddsportal.com/tennis/france/atp-french-open/results/#/page/2/",
3                               "https://www.oddsportal.com/tennis/france/atp-french-open/results/#/page/3/"],
4                               2023: ["https://www.oddsportal.com/tennis/france/atp-french-open-2023/results/",
5                                     "https://www.oddsportal.com/tennis/france/atp-french-open-2023/results/#/page/2/",
6                                     "https://www.oddsportal.com/tennis/france/atp-french-open-2023/results/#/page/3/",
7                                     "https://www.oddsportal.com/tennis/france/atp-french-open-2023/results/#/page/4/",
8                                     "https://www.oddsportal.com/tennis/france/atp-french-open-2023/results/#/page/5/"],
9                               2022: ["https://www.oddsportal.com/tennis/france/atp-french-open-2022/results/",
10                                      "https://www.oddsportal.com/tennis/france/atp-french-open-2022/results/#/page/2/",
11                                      "https://www.oddsportal.com/tennis/france/atp-french-open-2022/results/#/page/3/",
12                                      "https://www.oddsportal.com/tennis/france/atp-french-open-2022/results/#/page/4/",
13                                      "https://www.oddsportal.com/tennis/france/atp-french-open-2022/results/#/page/5/"]}]
14
15 oddsportal_url_grass = {2023: ["https://www.oddsportal.com/tennis/united-kingdom/atp-wimbledon/results/",
16                                "https://www.oddsportal.com/tennis/united-kingdom/atp-wimbledon/results/#/page/2/",
17                                "https://www.oddsportal.com/tennis/united-kingdom/atp-wimbledon/results/#/page/3/",
18                                "https://www.oddsportal.com/tennis/united-kingdom/atp-wimbledon/results/#/page/4/",
19                                "https://www.oddsportal.com/tennis/united-kingdom/atp-wimbledon/results/#/page/5/"],
20                                2022: ["https://www.oddsportal.com/tennis/united-kingdom/atp-wimbledon-2022/results/",
21                                       "https://www.oddsportal.com/tennis/united-kingdom/atp-wimbledon-2022/results/#/page/2/",
22                                       "https://www.oddsportal.com/tennis/united-kingdom/atp-wimbledon-2022/results/#/page/3/",
23                                       "https://www.oddsportal.com/tennis/united-kingdom/atp-wimbledon-2022/results/#/page/4/",
24                                       "https://www.oddsportal.com/tennis/united-kingdom/atp-wimbledon-2022/results/#/page/5/"]}]
25
26 oddsportal_url_hard = {2024: ["https://www.oddsportal.com/tennis/australia/atp-australian-open/results/#/page/1/",
27                                "https://www.oddsportal.com/tennis/australia/atp-australian-open/results/#/page/2/",
28                                "https://www.oddsportal.com/tennis/australia/atp-australian-open/results/#/page/3/",
29                                "https://www.oddsportal.com/tennis/australia/atp-australian-open/results/#/page/4/",
30                                "https://www.oddsportal.com/tennis/australia/atp-australian-open/results/#/page/5/"],
31                                2023: ["https://www.oddsportal.com/tennis/australia/atp-australian-open-2023/results/",
32                                       "https://www.oddsportal.com/tennis/australia/atp-australian-open-2023/results/#/page/2/",
33                                       "https://www.oddsportal.com/tennis/australia/atp-australian-open-2023/results/#/page/3/",
34                                       "https://www.oddsportal.com/tennis/australia/atp-australian-open-2023/results/#/page/4/",
35                                       "https://www.oddsportal.com/tennis/australia/atp-australian-open-2023/results/#/page/5/"],
36                                2022: ["https://www.oddsportal.com/tennis/australia/atp-australian-open-2022/results/",
37                                       "https://www.oddsportal.com/tennis/australia/atp-australian-open-2022/results/#/page/2/",
38                                       "https://www.oddsportal.com/tennis/australia/atp-australian-open-2022/results/#/page/3/",
39                                       "https://www.oddsportal.com/tennis/australia/atp-australian-open-2022/results/#/page/4/",
40                                       "https://www.oddsportal.com/tennis/australia/atp-australian-open-2022/results/#/page/5/"]}]
41
```

Annexe: configuration(URL)

```
42 atp_url = [  
43     "/en/players/novak-djokovic/d643/",  
44     "/en/players/jannik-sinner/s0ag/",  
45     "/en/players/carlos-alcaraz/a0e2/",  
46     "/en/players/alexander-zverev/z355/",  
47     "/en/players/daniil-medvedev/mm58/",  
48     "/en/players/andrey-rublev/re44/",  
49     "/en/players/casper-ruud/rh16/",  
50     "/en/players/hubert-hurkacz/hb71/",  
51     "/en/players/stefanos-tsitsipas/te51/",  
52     "/en/players/grigor-dimitrov/d875/",  
53     "/en/players/alex-de-minaur/dh58/",  
54     "/en/players/taylor-fritz/fb98/",  
55     "/en/players/holger-rune/r0dg/",  
56     "/en/players/tommy-paul/pl56/",  
57     "/en/players/ben-shelton/s0s1/",  
58     "/en/players/nicolas-jarry/j551/",  
59     "/en/players/ugo-humbert/hh26/",  
60     "/en/players/karen-khachanov/ke29/",  
61     "/en/players/alexander-bublik/bk92/",  
62     "/en/players/sebastian-baez/b0bi/",  
63     "/en/players/felix-auger-aliassime/ag37/",  
64     "/en/players/adrian-mannarino/me82/",  
65     "/en/players/francisco-cerundolo/c0au/",  
66     "/en/players/jiri-lehecka/l0bv/",  
67     "/en/players/alejandro-tabilo/te30/",  
68     "/en/players/frances-tiafoe/td51/",  
69     "/en/players/tallon-griekspoor/gj37/",  
70     "/en/players/sebastian-korda/k0ah/",  
71     "/en/players/tomas-martin-etcheverry/ea24/",  
72     "/en/players/arthur-fils/f0f1/",  
73     "/en/players/lorenzo-musetti/m0ej/",  
74     "/en/players/mariano-navone/n0bs/",  
75     "/en/players/cameron-norrie/n771/",  
76     "/en/players/alejandro-davidovich-fokina/dh50/",  
77     "/en/players/jack-draper/d0co/",
```

Annexe: configuration

```
12
13     from pprint import pprint
14     from tipe_scraper import *
15     from tipe_filter import *
16     from tipe_save import *
17     from config import *
18     from tipe_parser import *
19
20     # COMPTER JOUEURS DIFFERENTS DANS DONNEES
21
22     matches_dict = read_json('save_matches.json')
23     lst = []
24     for match in matches_dict:
25         if matches_dict[match]["Player 1"] not in lst:
26             lst.append(matches_dict[match]["Player 1"])
27         if matches_dict[match]["Player 2"] not in lst:
28             lst.append(matches_dict[match]["Player 2"])
29     print(len(lst))
30
```


Annexe: configuration

```
32 # SCRAPE URLS MEILLEURS JOUEURS ATP
33
34 driver = webdriver.Chrome()
35 driver.get("https://www.atptour.com/en/rankings/singles?RankRange=201-300&Region=all&DateWeek=2022-03-21")
36 time.sleep(0.5)
37 driver.execute_script("window.scrollTo(0, document.body.scrollHeight);") # va tout en bas du doc
38 time.sleep(1)
39 html_save = driver.page_source # prend code source
40 driver.quit()
41
42 soup = BeautifulSoup(html_save, 'html.parser')
43 player_urls = {}
44
45 lower_rows = soup.find_all('tr', class_='lower-row')
46
47 for row in lower_rows:
48     name_item = row.find('li', class_='name')
49     if name_item:
50         a_tag = name_item.find('a')
51         if a_tag and 'href' in a_tag.attrs:
52             player_name = a_tag.text.strip()
53             player_url = a_tag['href']
54             player_urls[player_name] = player_url
55
56 print(player_urls)
57 for p in player_urls.values(): print(p)
58
```

Annexe: configuration

```
60 # QUANTIFIER LA SAUSAGE PARTY
61
62 dict = {}
63 lst = []
64 lst2 = []
65 players_dict = read_json()
66 matches_dict = read_json('save_matches.json')
67 for match in matches_dict:
68     if matches_dict[match]["Player 1"] not in players_dict:
69         nom_j1 = matches_dict[match]["Player 1"]
70         if nom_j1 not in dict:
71             dict[nom_j1] = 1
72         else:
73             dict[nom_j1] += 1
74         if match not in lst: lst.append(match)
75     if matches_dict[match]["Player 2"] not in players_dict:
76         nom_j2 = matches_dict[match]["Player 2"]
77         if nom_j2 not in dict:
78             dict[nom_j2] = 1
79         else:
80             dict[nom_j2] += 1
81     if match not in lst: lst.append(match)
82
83 print(sum(dict.values()))
84 print(len(dict))
85 pprint(dict)
86 print(lst)
87 print(len(lst))
88
```

Annexe: configuration

```
90 # SUPPRIMER MATCHS UNITILISABLES (stats joueurs non scrappées)
91
92 print(len(matches_dict))
93 for key in lst:
94     matches_dict.pop(key, None)
95 print(len(matches_dict))
96 save_json(matches_dict, True, 'save_matches.json')
97
```


Annexe: configuration

```
119
120 # ODDSPORTAL SCRAPE MATCHES
121
122 for year in oddsportal_url_hard:
123     for URL in oddsportal_url_hard[year]:
124         try:
125             infos = scrape_oddsportal(URL, year, "Hard")
126             # pprint(infos) # debug
127             save_json(infos, False, 'save_matches.json')
128         except Exception as e:
129             continue
130
131
132 # ATPTOUR SCRAPE PLAYERS
133
134 for URL in atp_url:
135     name_filtered = player_name(URL)
136
137     log_player_status(name_filtered)
138
139     if is_player_in_json(name_filtered):
140         log_player_status(name_filtered, "ok (already scraped)")
141
142     else:
143         try:
144             infos_raw = scrape_atp(URL)
145             infos_filtered = {name_filtered: filter_player(infos_raw)}
146             # pprint(infos_filtered) # debug
147             save_json(infos_filtered)
148
149             log_player_status(name_filtered, "ok (written)")
150
151         except Exception as e:
152             log_player_status(name_filtered, f'error: {e}')
153             continue
```

Annexe: configuration

```
156 # SCRAPE ALL ODDS
157
158 driver = webdriver.Chrome()
159 driver.get("https://www.oddsportal.com/tennis/france/atp-french-open/results/#/page/2/")
160 time.sleep(0.5)
161 driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
162 time.sleep(1)
163 html_odds = driver.page_source
164 driver.quit()
165
166 url_matches = []
167 soup = BeautifulSoup(html_odds, 'html.parser')
168
169 group_flex_divs = soup.find_all('div', class_='group flex')
170
171 for index, div in enumerate(group_flex_divs):
172     a_tag = div.find('a', class_='next-m: flex next-m: !mt-0 ml-2 mt-2 min-h-[32px] w-full hover:cursor-pointer')
173     if a_tag:
174         match_url = a_tag.get('href')
175         url_matches.append(match_url)
176
177 print(url_matches)
178
179 odds = read_json('save_odds.json')
180
181 for URL in url_matches:
182     id = URL[-9:-1]
183     if id not in odds.keys():
184         odds[id] = scrape_allodds(f"https://www.oddsportal.com/{URL}")
185         save_json(odds, False, 'save_odds.json')
186 # pprint(odds)
```