# /HOW TO SCRAPE THE FCA WEBSITE

BONNIE AND CLOUD

LDA HEC PARIS 2023

INDEX.HTML

# /TABLE OF CONTENTS

## /01
## <AIM OF THE ALGORITHM>

VBER 2022

INDEX.HTML

## /AIM OF THE ALGORITHM

### /CONTEXT

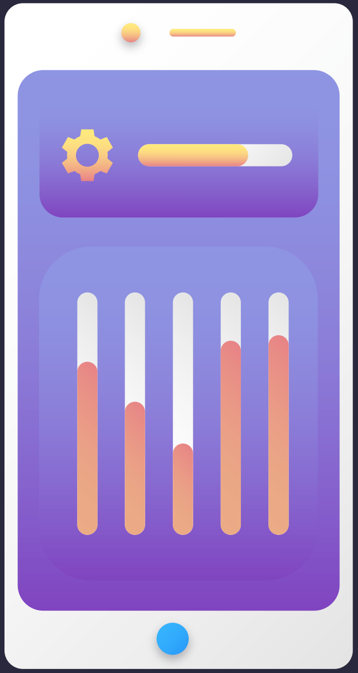
Vertical Block Exemption Regulation "the VBER" (came into force on June 1, 2022)

### /FOCUS ON DISTRIBUTION

Agreements between distributors and suppliers

### /ALGORITHM

Have decisions by the FCA tended to decrease or increase over time ?



/02

## <CODING METHOD>

INDEX.HTML

# /KEY FEATURES

## /WEB REQUESTS

Reaching the FCA's website

## /ITERATIONS

Scraping different layers of data

## /PATTERNS

Identifying relevant decisions

## /DATA VISUALIZATION

Building an easy-to-manipulate data frame

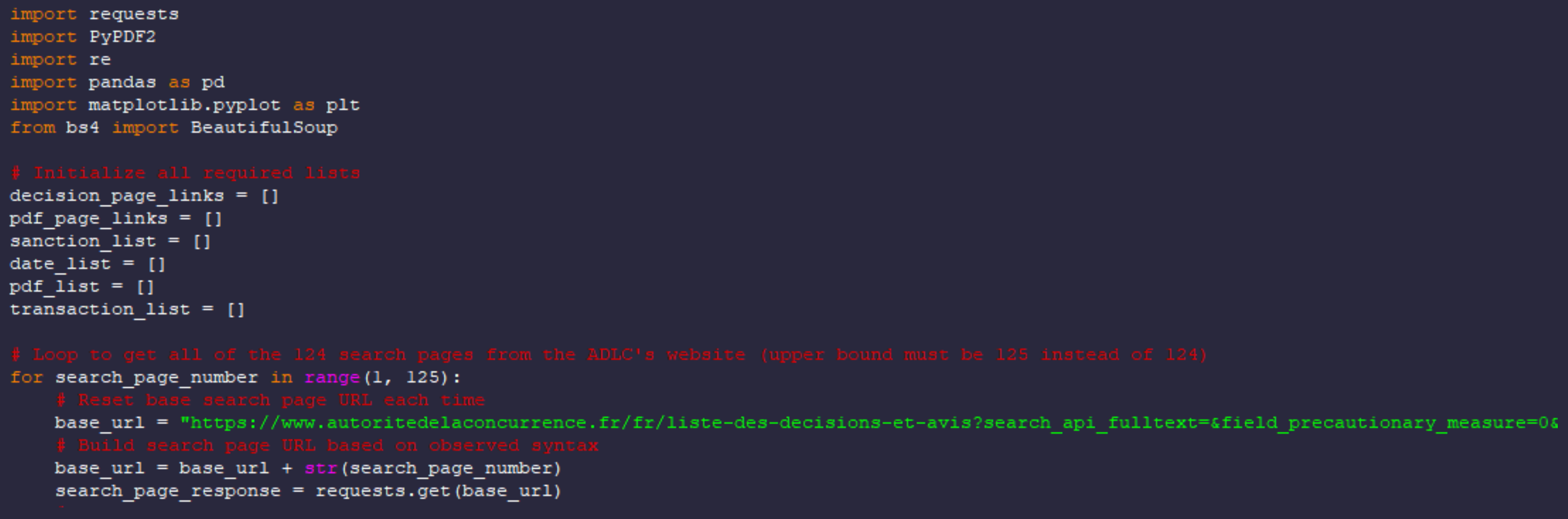
## /CHARTS

Understanding and interpreting trends

INDEX.HTML

# /WEB REQUESTS

After analysing the syntax of URL addresses on the FCA's website, we reached out to the search pages with our chosen criteria using the *Requests* module (imported with all others at the start of the code)



```python
import requests
import PyPDF2
import re
import pandas as pd
import matplotlib.pyplot as plt
from bs4 import BeautifulSoup

# Initialize all required lists
decision_page_links = []
pdf_page_links = []
sanction_list = []
date_list = []
pdf_list = []
transaction_list = []

# Loop to get all of the 124 search pages from the ADLC's website (upper bound must be 125 instead of 124)
for search_page_number in range(1, 125):
    # Reset base search page URL each time
    base_url = "https://www.autoritedelaconcurrence.fr/fr/liste-des-decisions-et-avis?search_api_fulltext=&field_precautionary_measure=0&
    # Build search page URL based on observed syntax
    base_url = base_url + str(search_page_number)
    search_page_response = requests.get(base_url)
```

INDEX.HTML

## /ITERATIONS

We needed to scrap information on 3 different levels:

- Obtaining individual decision page links from the search page
- Obtaining the PDF link from each individual decision page
- Obtaining the fine amount from each PDF file

```
if decision_page_response.status_code == 200:
    print("Second level request (individual decision page " + decision_page_link + ") successful")
    decision_page_html_content = decision_page_response.text
    decision_page_soup = BeautifulSoup(decision_page_html_content, "html.parser")
    # Find all the links in the HTML using the 'a' tag
    unfiltered_decision_page_links = decision_page_soup.find_all("a")
    pdf_page_links = []
    # Note: add filter for transaction
    for link in unfiltered_decision_page_links:
        # Exclude from the links all those whose URL do not follow the syntax used for decision PDF files
        if str(link)[:32] == '<a aria-label="le texte intégral':
        # Extract the 'href' attribute from each link
            pdf_page_links = pdf_page_links + [link.get("href")]
            print("PDF link found on this page is " + str(link.get("href")))
    if pdf_page_links == []:
        print("Could not find any PDF link on decision page " + str(decision_page_link))
    else:
        print("Finished listing PDF links on " + str(decision_page_link))
```

This was achieved with the *BeautifulSoup* and *PyPDF2* modules

INDEX.HTML

## /PATTERNS

```python
# Use regex to search for "distribution" in the decision's PDF file's text
legal_basis = re.search(r"distribution", text)
if legal_basis:
    # Use regex to search for sanction patterns in the decision's PDF file's text
    sanction_pattern_1 = re.search(r"une sanction de (\d+(?:\s+\d+)*)", text)
    sanction_pattern_2 = re.search(r"une sanction pécuniaire de (\d+(?:\s+\d+)*)", text)
    # For each possible sanction pattern, retrieve the fine, date and PDF link
    if sanction_pattern_1:
        sanction_text = sanction_pattern_1.group().split("de")[1].strip()
        sanction = str(sanction_text)
        sanction_list = sanction_list + [sanction]
        print("Sanction found with value: ", sanction)
        date_text = str(pdf_link)[-9:][:2]
        date_list = date_list + [date_text]
        pdf_list = pdf_list + [str(pdf_link)]
        # Identify bargains
        transaction_boolean = re.search(r"Transaction", decision_page_html_content)
        if transaction_boolean:
            transaction_list = transaction_list + ["Bargain"]
        else:
            transaction_list = transaction_list + ["Other"]
    elif sanction_pattern_2:
        sanction_text = sanction_pattern_2.group().split("de")[1].strip()
        sanction = str(sanction_text)
        sanction_list = sanction_list + [sanction]
        print("Sanction found with value: ", sanction)
        date_text = str(pdf_link)[-9:][:2]
        date_list = date_list + [date_text]
        pdf_list = pdf_list + [str(pdf_link)]
        # Identify bargains
        transaction_boolean = re.search(r"Transaction", decision_page_html_content)
        if transaction_boolean:
            transaction_list = transaction_list + ["Bargain"]
        else:
            transaction_list = transaction_list + ["Other"]
```

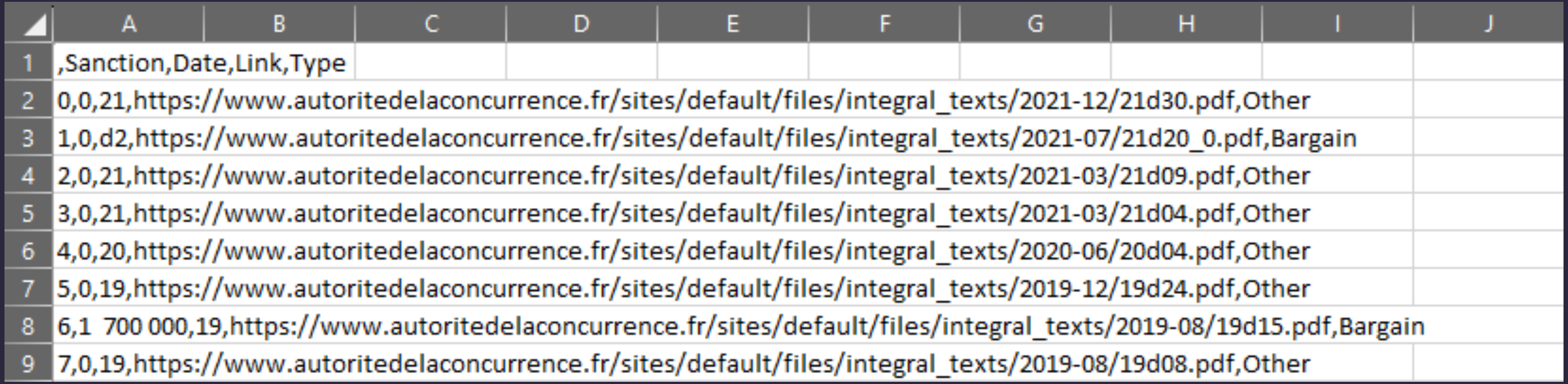
We identified relevant decisions and their types using different patterns:

- Distribution was used as a key word to identify decisions related to it
- 2 different patterns for introducing sanctions were identified and used
- Bargains were identified using "transaction" as a key word

## /DATAFRAME CREATION

Having extracted all relevant pieces of information in lists, we put them into the columns of a CSV file

```python
# Put the data in a dataframe
sanction_dataframe = pd.DataFrame({'Sanction': sanction_list, 'Date': date_list, 'Link': pdf_list, 'Type': transaction_list})
# Export the final dataframe to CSV format
sanction_dataframe.to_csv('ADLC_Sanctions.csv')
```



| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ,Sanction,Date,Link,Type | | | | | | | | | |
| 2 | 0,0,21,https://www.autoritedelaconcurrence.fr/sites/default/files/integral_texts/2021-12/21d30.pdf,Other | | | | | | | | | |
| 3 | 1,0,d2,https://www.autoritedelaconcurrence.fr/sites/default/files/integral_texts/2021-07/21d20_0.pdf,Bargain | | | | | | | | | |
| 4 | 2,0,21,https://www.autoritedelaconcurrence.fr/sites/default/files/integral_texts/2021-03/21d09.pdf,Other | | | | | | | | | |
| 5 | 3,0,21,https://www.autoritedelaconcurrence.fr/sites/default/files/integral_texts/2021-03/21d04.pdf,Other | | | | | | | | | |
| 6 | 4,0,20,https://www.autoritedelaconcurrence.fr/sites/default/files/integral_texts/2020-06/20d04.pdf,Other | | | | | | | | | |
| 7 | 5,0,19,https://www.autoritedelaconcurrence.fr/sites/default/files/integral_texts/2019-12/19d24.pdf,Other | | | | | | | | | |
| 8 | 6,1 700 000,19,https://www.autoritedelaconcurrence.fr/sites/default/files/integral_texts/2019-08/19d15.pdf,Bargain | | | | | | | | | |
| 9 | 7,0,19,https://www.autoritedelaconcurrence.fr/sites/default/files/integral_texts/2019-08/19d08.pdf,Other | | | | | | | | | |

This vastly facilitates data visualization and cleaning

# /CHARTS

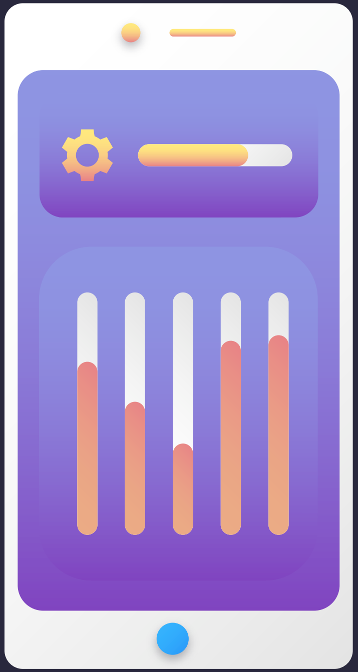
```python
# Read the final dataframe
sanction_reading = pd.read_csv('ADLC_Sanctions.csv')
# Prepare the decisions by year bar chart
# Use a pattern to drop all abnormal years
pattern = r"([0-2][0-9])"
sanction_reading = sanction_reading[sanction_reading['date'].str.contains(pattern, na=False)]
# Group the rows by year and count the number of occurrences
year_counts = sanction_reading.groupby('date').size()
# Plot the counts as a bar chart
year_counts.plot(kind='bar')
# Add axis labels and a title
plt.xlabel('Year')
plt.ylabel('Number of decisions')
plt.title('Decisions in each year')
plt.show()
# Prepare the decision type pie chart
type_counts = sanction_reading['type'].value_counts()
# Create a pie chart showing the proportion of "Bargain" and "Other" values
type_counts.plot(kind='pie', labels=type_counts.index, autopct='%1.1f%%')
# Add a title to the chart
plt.title('Bargains as a proportion of total decisions')
# Show the chart
plt.show()
# Prepare the fine status pie chart
# Count the number of occurrences of each value in the column of interest
sanction_counts = sanction_reading['sanction'].value_counts()
# Create a new column with a binary indicator for each category
sanction_reading['category'] = sanction_reading['sanction'].apply(lambda x: 'No fine' if x == "0" else 'Fine')
# Count the number of occurrences of each category
category_counts = sanction_reading['category'].value_counts()
# Create a pie chart with two categories
category_counts.plot(kind='pie', labels=category_counts.index, autopct='%1.1f%%')
# Add a title to the chart
plt.title('Proportion of fined and not fined parties')
# Show the chart
plt.show()
```

Using the final data frame, we had Python create several charts to facilitate interpretation:

- A bar chart on the number of decisions in each year
- A pie chart on the proportion of fined companies
- A pie chart on the proportion of bargains

## /03

## METHODOLOGICAL BIASES AND DIFFICULTIES



INDEX.HTML

## /METHODOLOGICAL BIASES

Our code is perfectible and to make it run we had to make some simplifications

### /SEARCH FOR FINES IN THE LAST TWO PAGES

- Fines not indicated in the last two pages are not extracted

### /SEARCH FOR DISTRIBUTION IN THE LAST TWO PAGES

- Decisions in which distribution appears only in the statement of the decision are not extracted

### /SYNTAXES THAT WE PROBABLY MISSED

- Syntax deduced from an empirical study of some decision

## /ISSUES ENCOUNTERED

/01 /FCA WEBSITE: NOT A DATA-FRIENDLY MEDIA

/02 /INADEQUACY OF THE DATA SET FOR SOME KEY WORDS

/03 /SPEED OF EXECUTION

/04 /REQUEST MODULE: PYTHON COPYRIGHT ISSUES

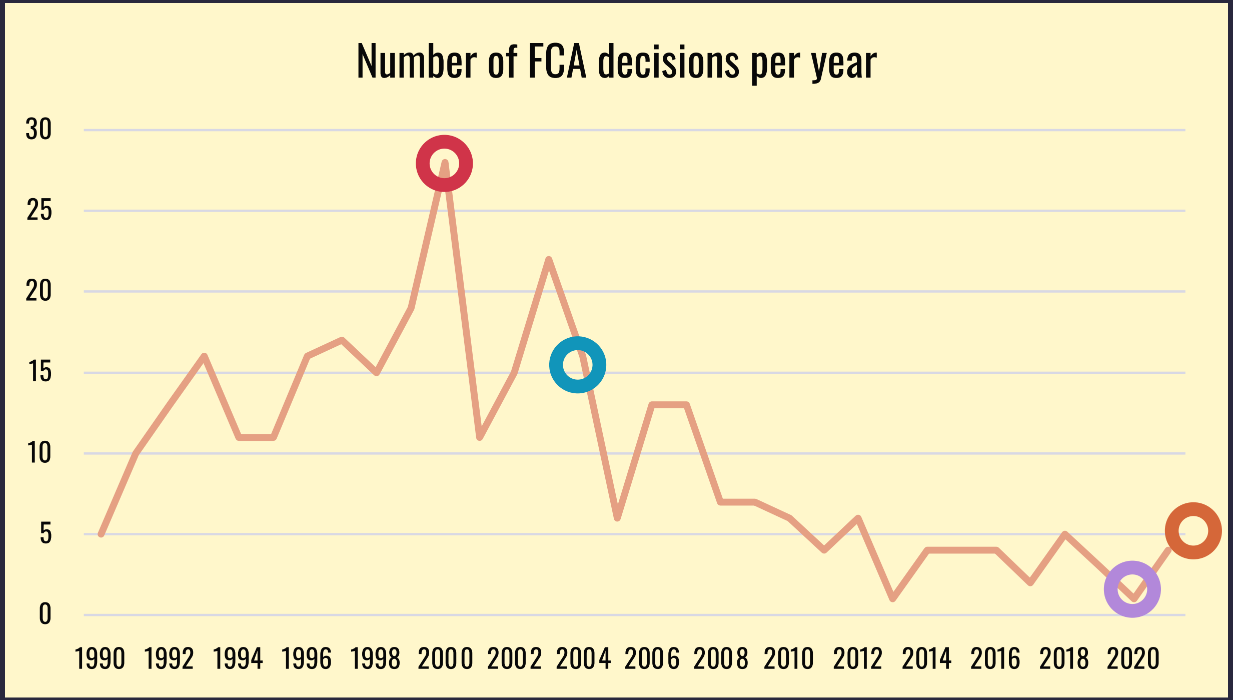
/05 /DECISIONS BEFORE 2001 IN FRENCH FRANCS

€

## /04 <ANALYSIS OF THE SCRAPING>

INDEX.HTML

# /GRAPH ANALYSIS



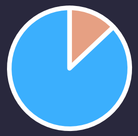
**2000** > Highest point

**2004** > Introduction of self-assessment

**2020** > New priority for the FCA : digital

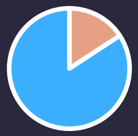
**2022** > Introduction of the VBER

Since 1990



🟧 Sentences

Since 2016



🟦 Others

# /SENTENCES

> The FCA still tends to condemn as much

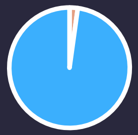
> The average financial penalty is 474,757 €

> Highest financial penalty : 50 M€

INDEX.HTML

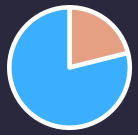
## /SETTLEMENT

Since 1990



🟧 Settlement

Since 2016



🟦 Others

> It allows companies that do not challenge the allegations brought against them to obtain a financial penalty within a range proposed by the General Rapporteur and agreed by the parties

> The FCA more open to settlement procedures

# /THANKS!

/DO YOU HAVE ANY QUESTIONS?

BONNIE AND CLOUD

</>

LDA HEC PARIS 2023

INDEX.HTML