DATA ANALYSIS

Inspython des impôts

The supervision of the tax administration by the Conseil d'Etat : the evolution of the control of the accounting audit

Purpose of the work carried out

Analyze through a python code the number of decisions related to tax audits that go to the Conseil d'Etat

To achieve this, an 7-step approach had to be undertaken :

- Step 1 : Preparation
- Step 2 : Setting of the search parameters
- Step 3 : Creation of the dataframe
- Step 4 : Using a function to extract the line of Ariane webpage table's
- Step 5 : Creating a list to put every columns element's : row
- Step 6 : Extraction of relevant data
- Step 7 : Exportation of the data in a CSV file

TABLE OF CONTENTS

01.

The process

With a Python code

Step 1: Preparation

1.1. Python's prep

```
import pandas as pd
import regex as re
import time
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import WebDriverWait
from webdriver_manager.chrome import ChromeDriverManager
```

1.2. Driver's setup

```
# CHANGER CETTE VALEUR À True POUR REGENERER LE CSV:
CREATE_CSV = False

if __name__ == '__main__':
    if CREATE_CSV:
        # Setup du driver et initialisation
        chrome_options = Options()
        chrome_options.add_experimental_option("detach", True)
        driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=chrome_options)
        driver.get("https://www.conseil-etat.fr/arianeweb/#/recherche")
```

As we are analyzing Conseil d'Etat's decision, we must use Ariane as a base.

Step 2 : Setting of the search parameters :

2.1. Selection of the settings

```
# Application des parametres de recherche
el = driver.find_element(By.XPATH, r'.//*[contains(text(), "Décisions du Conseil")]')
el.click()

el2 = driver.find_element(By.XPATH, r'.//*[@ng-model="search.currentSearch.query"]')
el2.click()
el2.send_keys("vérification de comptabilité")
```

Our settings were the following ones :

- **« Décisions du Conseil » ;**
- **« Vérification de comptabilité ».**

2.2. Launch of the search

```
button = driver.find_element(By.XPATH, ".//button[@class='btn btn-primary']")
button.click()
```

Step 3 : Creation of the dataframe

Then, we created a dataframe in which we planned to put results :

```
declarations_df = pd.DataFrame()
```

```
next_page = "//table/tbody/tr[2]/td[1]/div[2]/ul/li/a[@ng-click='selectPage(page + 1)']"
next_page_disabled = "//table/tbody/tr[2]/td[1]/div[2]/ul/li[@class='ng-scope disabled']"
```

Step 4 : Using a function to extract the line of Ariane webpage table's

Our function extract every line for each page of the table of the Ariane website, open the affiliate document and try to get every administrative court of appeal concerned and its decision. As far as the Ariane webpage display 50 results per page, this function add 50 lines to our data frame of results, which is declarations_df as we exposed before.

```
def compute_and_append_page_results_to_df(df):

WebDriverWait(driver, 3).until(EC.visibility_of_element_located((By.XPATH, next_page)))
declarations = driver.find_element(By.XPATH, '(//table)[2]').get_attribute('outerHTML')
soup = BeautifulSoup(declarations, 'html.parser')
```

We create columns in our table :

```
 # Liste des colonnes: ['id', 'Fonds', 'Juridiction', 'Formation de jugement', 'Date de lecture', "Numéro d'affaire", 'Code de publication']
 columns = [th.getText().rstrip(' ') for th in soup.find_all('th')]  # Colonnes du tableau de résultat
 columns[0] = 'id'
 columns.extend(['Cour', 'Décision'])

# page de départ du driver
main_window = driver.current_window_handle
```

Step 5 : Creating a list to put every columns element's : row

Row is a list in which every relevant element will be put. To create our list, we create a loop:

```
rows = []
for tr in soup.find_all('tr', {"ng-repeat-start": 'result in results'}):
```

The following line is going to read the content of the table's line :

```
row_soup = BeautifulSoup(''.join(str(el) for el in tr.contents), features='lxml')
```

Then, for each line, we create a list which is going to contain columns elements' :

```
row = []
for td in row_soup.find_all('td', {'ng-repeat': 'field in result'}):
    row.append(td.getText())
```

A window which contains every document associated with the lines pops up.

```
if row[0] == '1':
    id_element = driver.find_element(By.XPATH, "(//table)[2]/tbody/tr/td[text()=" + row[0] + "]")
    id_element.click()
    time.sleep(5)
```

A window which contains every document associated with the lines pops up.

```
if row[0] == '1':
    id_element = driver.find_element(By.XPATH, "(//table)[2]/tbody/tr/td[text()=" + row[0] + "]")
    id_element.click()
    time.sleep(5)


driver.switch_to.window(driver.window_handles[1])
driver.switch_to.frame(driver.find_element(By.XPATH, '//iframe'))
```

If the pop up is open, we move on to the next document with an “else”.

```
else:
    driver.switch_to.window(driver.window_handles[1])
    driver.find_element(
        By.XPATH,
        "//button[@title='document suivant']"
    ).click()
    time.sleep(0.1)
    driver.switch_to.frame(driver.find_element(By.XPATH, '//iframe'))
```

Step 6 : Extraction of relevant data

Then we need to identify and extract the interesting data for the study. First, we focus on the extraction of the Administrative Courts of Appeal :

```
doc_source = driver.page_source.replace('\n', '')
caa_re = re.compile(r".*cour administrative d'appel (de |d')(?P<caa>\w+).*")
cours_des_comptes_re = re.compile(r'.*(?P<cdc>(C|c)our des (C|c)omptes).*')
caa = caa_re.match(doc_source)
cours_des_comptes = cours_des_comptes_re.match(doc_source)
cour = ''
if caa:
    cour = caa.group('caa')
elif cours_des_comptes:
    cour = cours_des_comptes.group('cdc')
else:
    cour = None
```

Then we do the exact same thing for all the decisions, by making a list of lines :

```
doc_source_rows = driver.page_source.split('\n')
decision_re = re.compile(r"(?i)^.*article 1er\s*[:\-;].*\s(?P<decision>(annul(é|e)|rejet(é|e))e?s?).*$")
decision = ''

for doc_source_row in doc_source_rows:
    decision = decision_re.match(doc_source_row)
    if decision:
        decision = decision.group('decision')
        if decision.startswith(('A', 'a')):
            decision = 'annulé'
        else:
            decision = 'rejeté'
        break
    else:
        decision = None
```

Then we add the data found to the row, and we add it to the dataframe :

```
row.append(cour)
row.append(decision)
rows.append(row)
```

We return to the Ariane web page and add the new lines to the dataframe.

```
driver.switch_to.window(main_window)
df = pd.concat((df, pd.DataFrame(data=rows, columns=columns)), ignore_index=True)
return df
```

Then we go to the next page :

```
element = driver.find_element(By.XPATH, next_page)
driver.execute_script("arguments[0].click();", element)
time.sleep(0.5)
```

And then we create a loop on all the other web pages that stops when the button to go to the next page is no longer available :

```
while not driver.find_elements(By.XPATH, next_page_disabled):
    declarations_df = compute_and_append_page_results_to_df(declarations_df)
    element = driver.find_element(By.XPATH, next_page)
    driver.execute_script("arguments[0].click();", element)
    time.sleep(0.5)
```

And finally we add last page results :

```
declarations_df = compute_and_append_page_results_to_df(declarations_df)
```

Step 7 : Exportation of the data in a CSV file

We then export the results in a csv file :

```
declarations_df.to_csv('results.csv', index=False)
```

Final step : we load the dataframe in order to analyze the data :

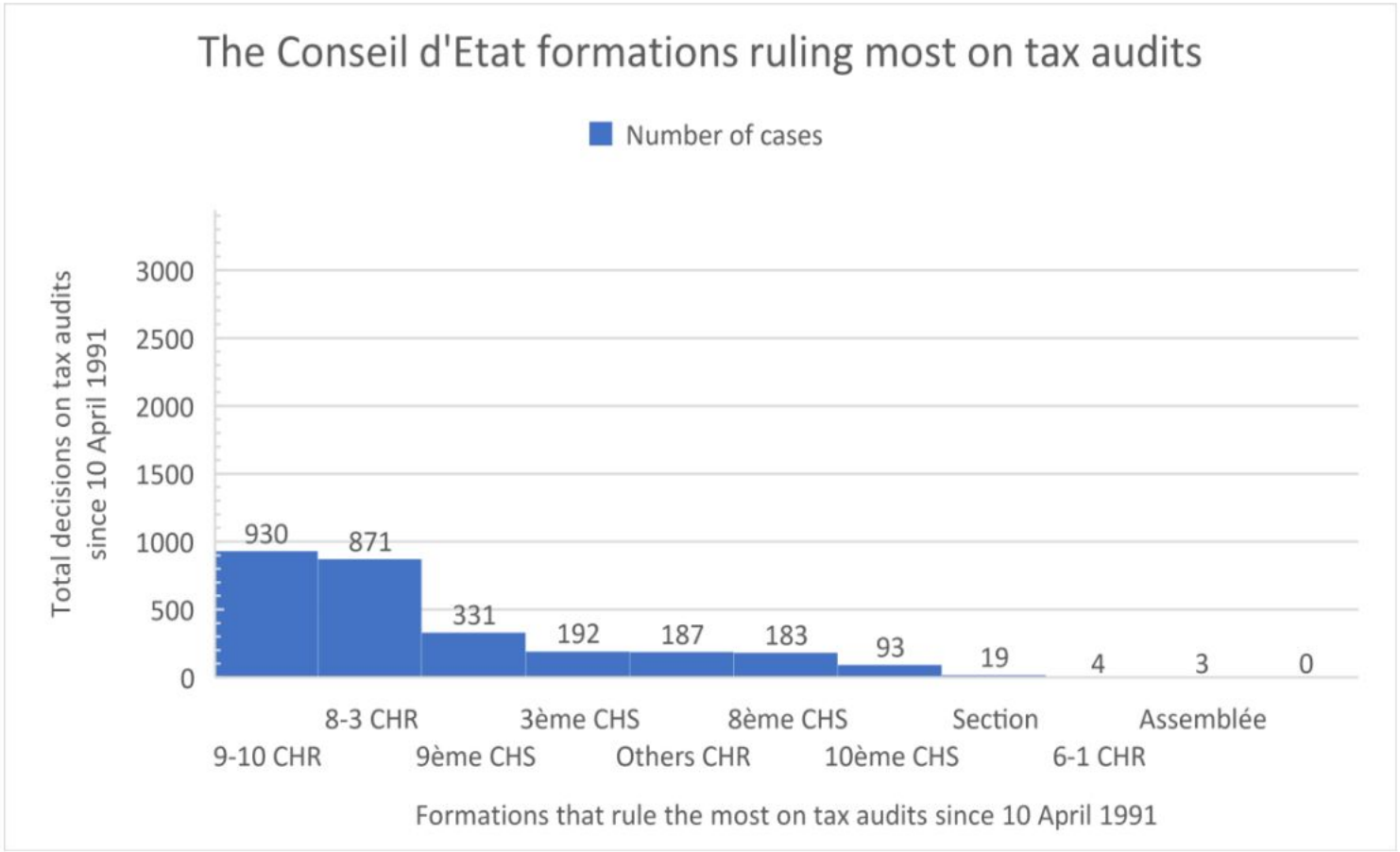
```
declarations_df = pd.read_csv('results.csv', index_col=0)

df_of_interest = declarations_df[
    ['Formation de jugement', 'Date de lecture', "Numéro d'affaire", 'Code de publication', 'Cour', 'Décision']
]
df_of_interest['Date de lecture'] = pd.to_datetime(df_of_interest['Date de lecture'], format="%d/%m/%Y")
```

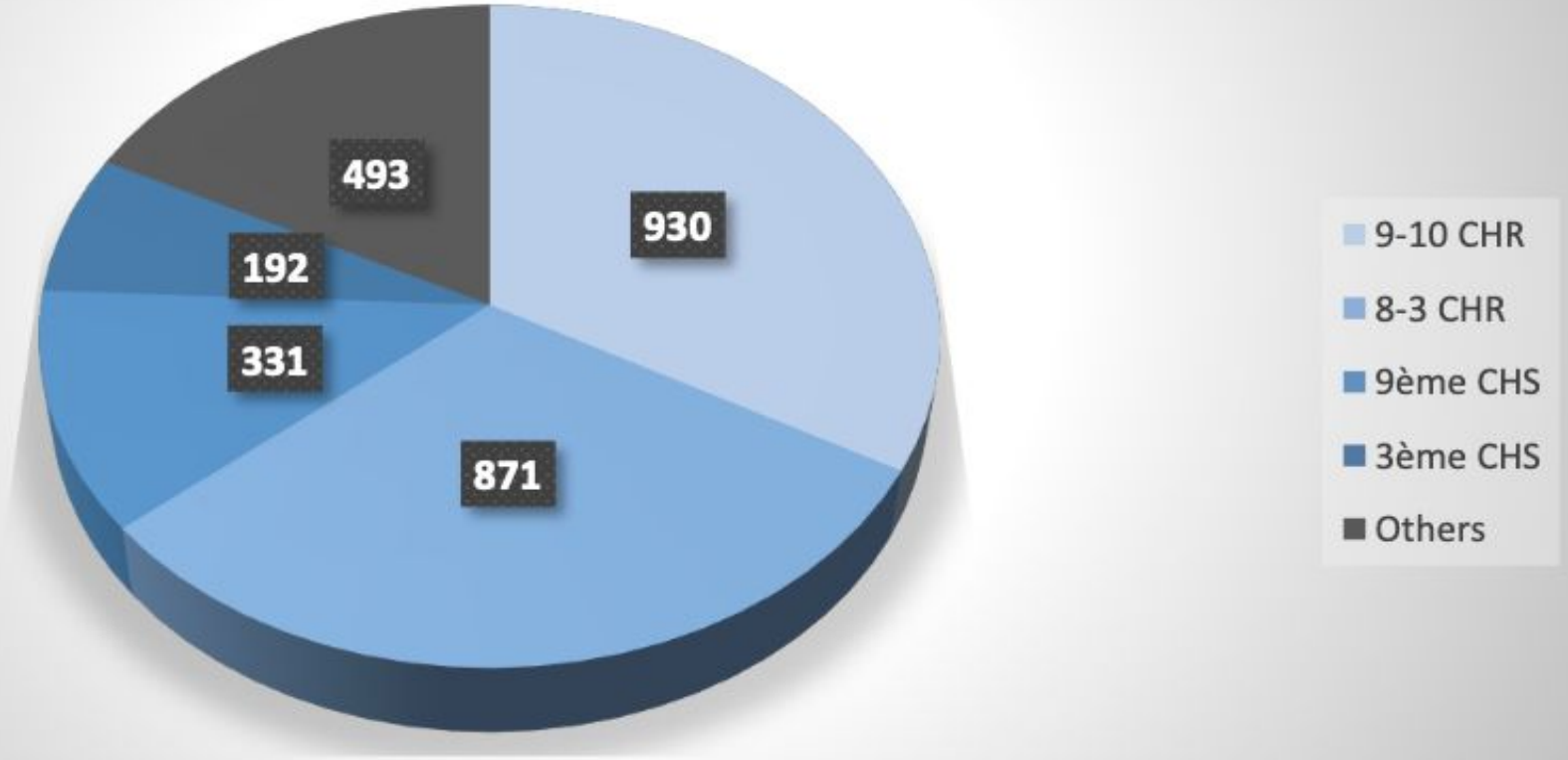
02.

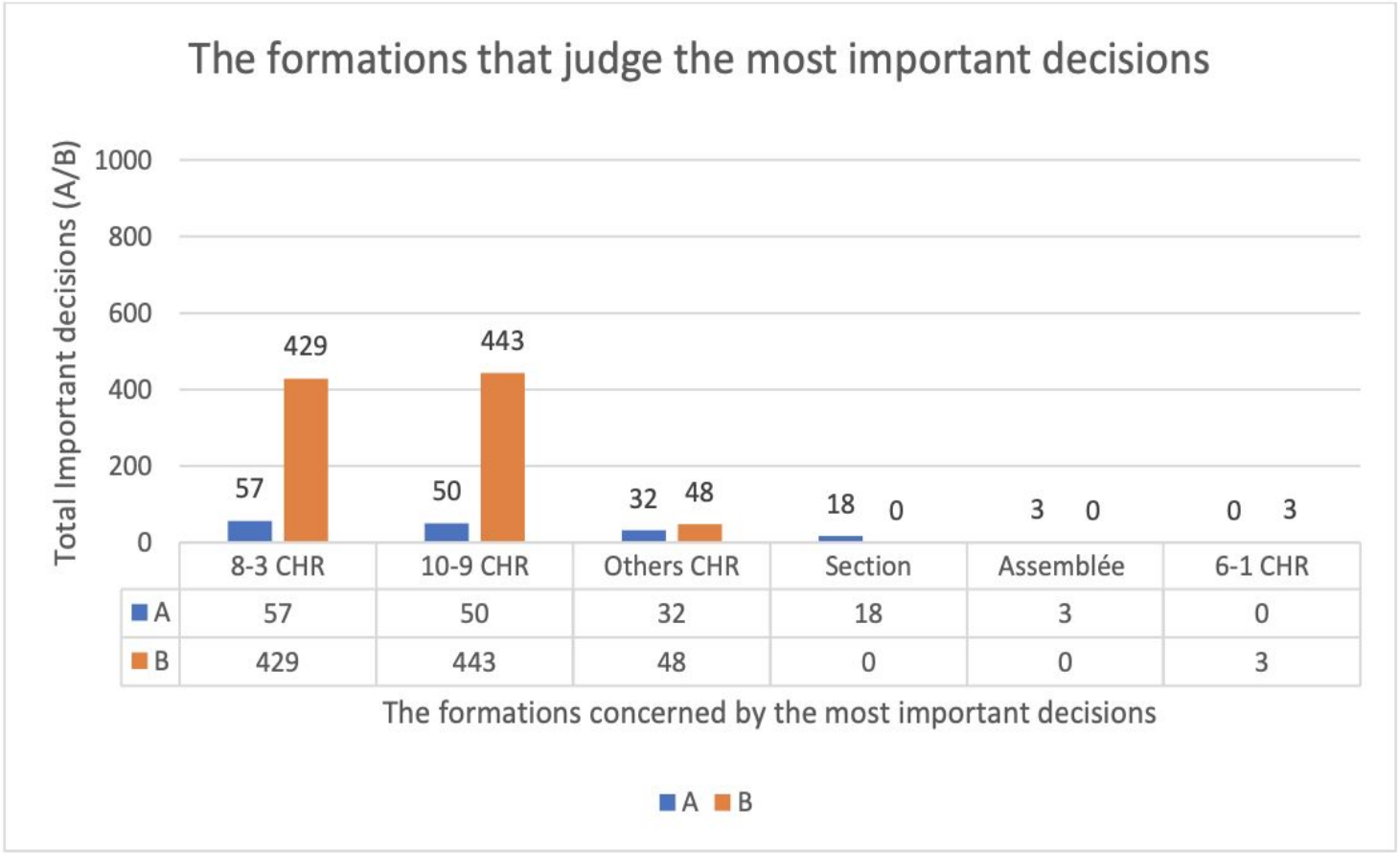
Data Analysis

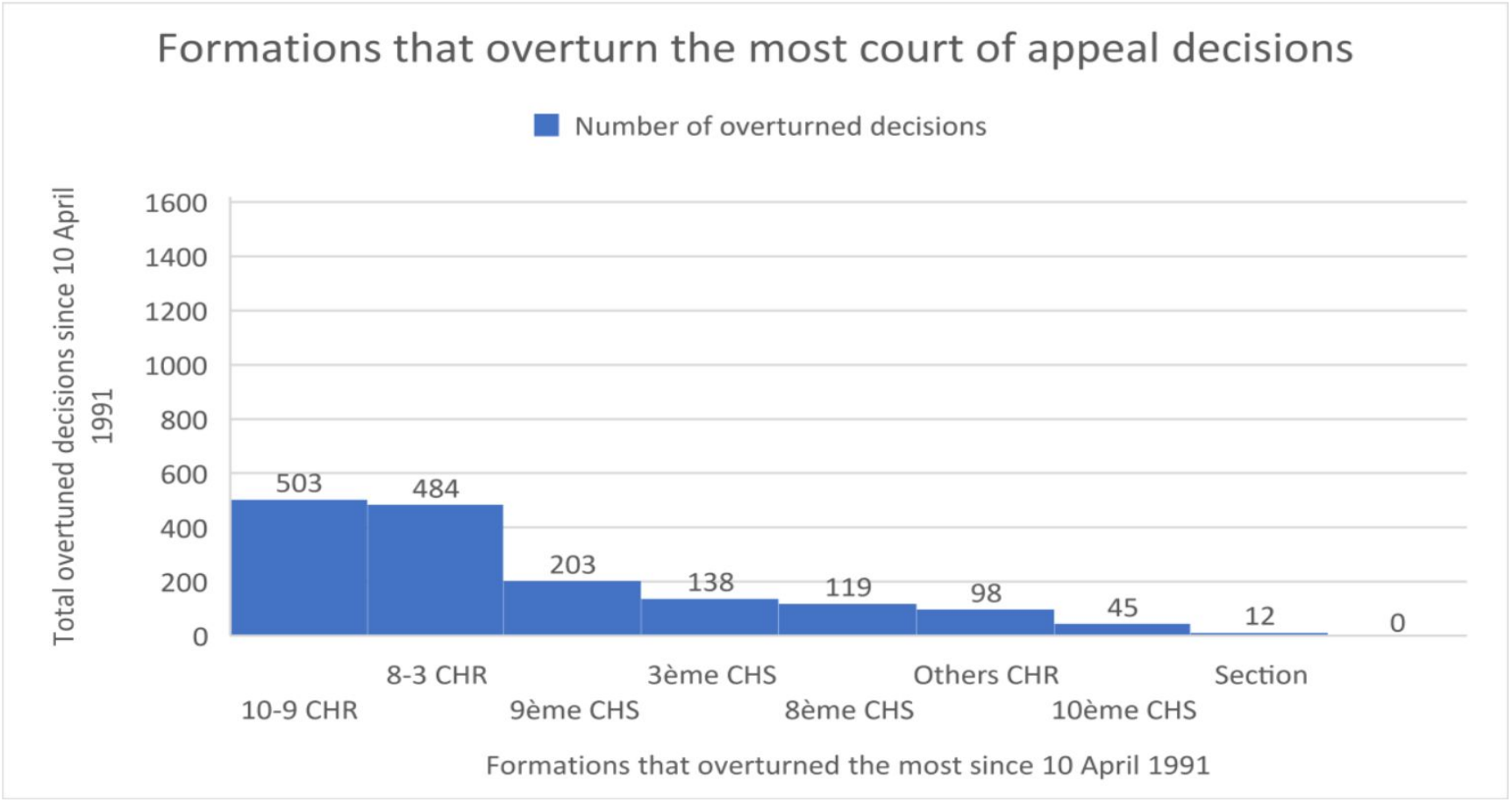
With graphs

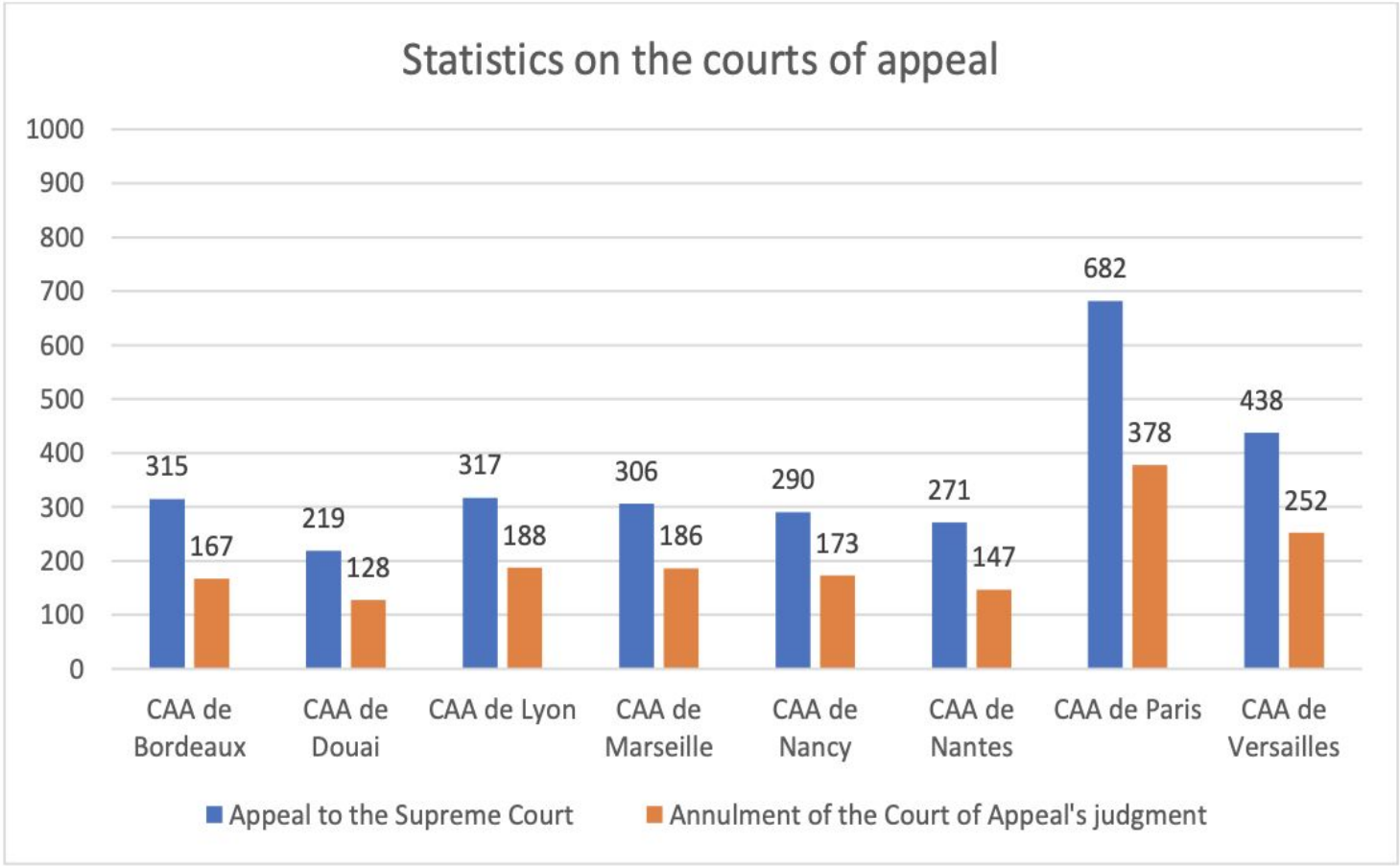

The Conseil d'Etat formations ruling most on tax audits
Number of cases
Total decisions on tax audits since 10 April 1991
3000
2500
2000
1500
1000
500
0
930
871
331
192
187
183
93
19
4
3
0
9-10 CHR
8-3 CHR
9ème CHS
3ème CHS
Others CHR
8ème CHS
10ème CHS
Section
6-1 CHR
Assemblée
Formations that rule the most on tax audits since 10 April 1991

The Conseil d'Etat chambers ruling most on tax audits




The formations that judge the most important decisions
Total Important decisions (A/B)
1000
800
600
400
200
0
429
443
57
50
32
48
18
0
3
0
0
3
8-3 CHR
10-9 CHR
Others CHR
Section
Assemblée
6-1 CHR
A
57
50
32
18
3
0
B
429
443
48
0
0
3
The formations concerned by the most important decisions
A
B


Formations that overturn the most court of appeal decisions
Number of overturned decisions
Total overtuned decisions since 10 April 1991
1600
1400
1200
1000
800
600
400
200
0
503
484
203
138
119
98
45
12
0
10-9 CHR
8-3 CHR
9ème CHS
3ème CHS
8ème CHS
Others CHR
10ème CHS
Section
Formations that overturned the most since 10 April 1991


Statistics on the courts of appeal
1000
900
800
700
600
500
400
300
200
100
0
315
167
219
128
317
188
306
186
290
173
271
147
682
378
438
252
CAA de Bordeaux
CAA de Douai
CAA de Lyon
CAA de Marseille
CAA de Nancy
CAA de Nantes
CAA de Paris
CAA de Versailles
Appeal to the Supreme Court
Annulment of the Court of Appeal's judgment

To Conclude

52%

Formations ruling the most on tax audits

9-10 CHR and 8-3 CHR

90%

Formations judging the most important decisions

9-10 CHR and 8-3 CHR

61%

Formations that overturn the most

9-10 CHR and 8-3 CHR

OUR TEAM

Chloé Rochard

Mathilde Paganini

Margot Bolaers

Océane Sylla

THANKS !

Do you have any questions?

Inspython des impôts