Data analytics

Standardization or lack of legal certainty in decision awarding the recovery of legal fees under Article 700 French Code of Civil Procedure ?

Plan

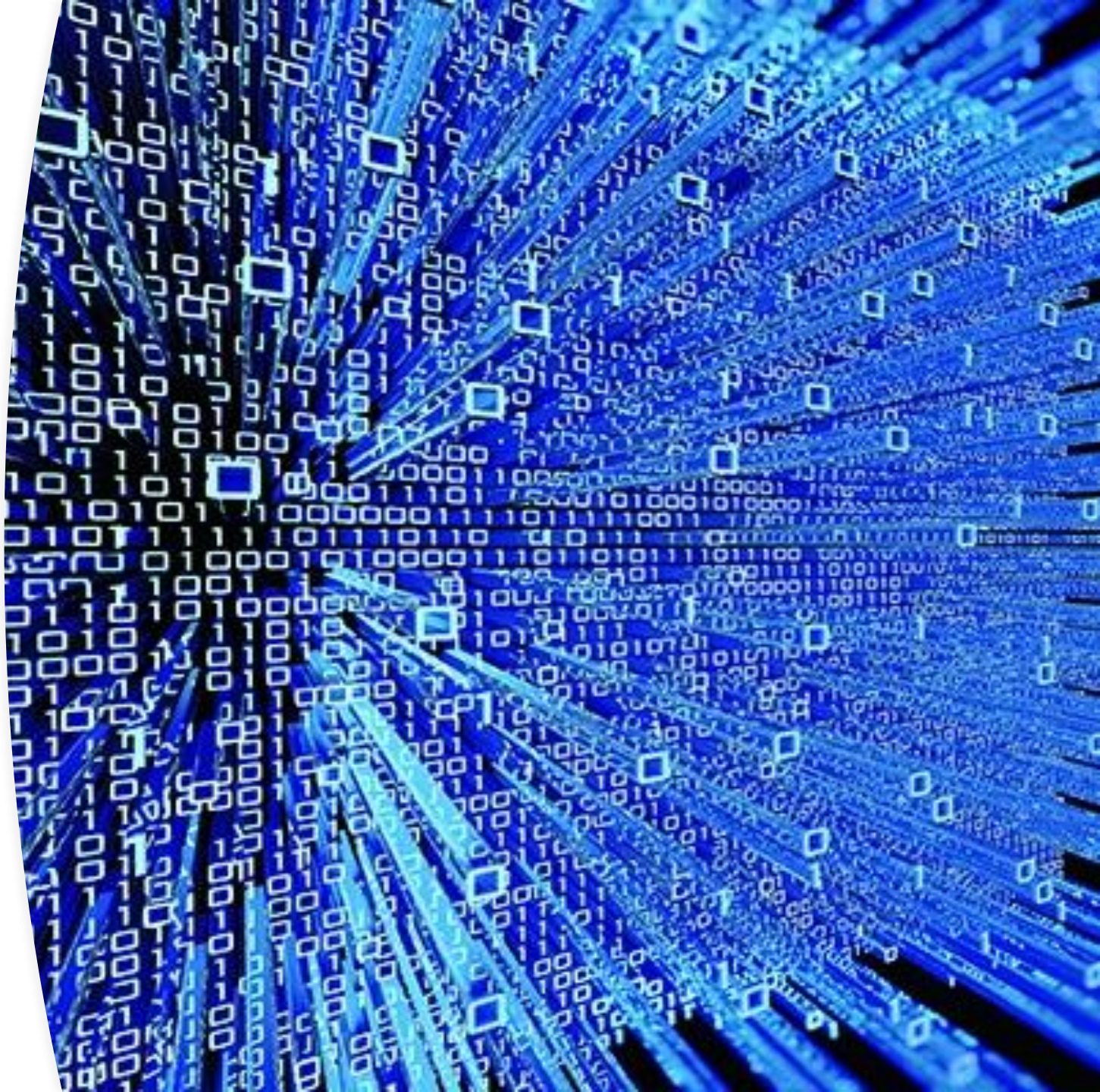
I-The aim of this algorithm

II-The presentation of the algorithm

III- The data frame

III- Results & Interpretation

V-The key areas for improvement



I-The aim of this algorithm

A need

- Legal certainty on the quantification of legal fees

A goal

- better knowing about the possible amounts granted under article 700 of the french Code de procédure civile

A method

- The statistical analysis
  - Cour d'appel
  - Insurance law related cases

II-The presentation of the algorithm

The scraping process

1

```
base_url = "https://www.legifrance.gouv.fr"
webpage = "https://www.legifrance.gouv.fr/search/juri?tab_selection=juri&query=%7B(
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
driver.get(webpage)
url1 = "https://www.legifrance.gouv.fr/search/juri?tab_selection=juri&query=%7B(%40
url2 = "&tab_selection=juri#juri"
url_full =""
```

2

3

```
for page_index in range(2,18): # CHANGE TO (2, 18) FOR FINAL PRESENTATION
    soup = BeautifulSoup(driver.page_source)
    decisions = soup.find_all("article", class_="result-item")
    for decision in decisions:
        my_element = decision.find("a")
        decision_link = my_element.get("href")
        full_link = base_url + decision_link
        decisions_links.append(full_link)
    x = [url1,str(page_index),url2]
    url_full = "".join(x)
    driver.get(url_full)
```

```
for url in decisions_links:
    result_list = []
    result_dflist = []
    sub_list = []
    digit_list = []
    amount_list = []
    driver.get(url)
    soup = BeautifulSoup(driver.page_source)
    meta_jurisdiction = soup.find("h2", class_="title horsAbstract print-black") # jurisdiction + date + Case number # IN THE SAME LOOP AS PREVIOUSLY
    meta_jurisdiction_text = meta_jurisdiction.get_text()
    search_meta_jurisdiction = re.search(r"Cour d'appel d'Agen|Cour d'appel d'Aix-en-Provence|Cour d'appel d'Amiens|Cour d'appel d'Angers|Cour d'appel de Basse-T
    if search_meta_jurisdiction:
        search_meta_jurisdiction.start()
        search_meta_jurisdiction_text_clear = meta_jurisdiction_text[search_meta_jurisdiction.start():search_meta_jurisdiction.end()]
    #meta_jurisdiction_digit_clear = re.sub("\d+", "", meta_jurisdiction_text)
    #meta_jurisdiction_text_clear = re.sub("-\s\w+|-\s", "", meta_jurisdiction_digit_clear)
    sub_list.append(search_meta_jurisdiction_text_clear)
    print(search_meta_jurisdiction_text_clear)
    meta_date = soup.find("div", class_="h2 title horsAbstract print-black")
    meta_date_text = meta_date.get_text()
    search_date = re.search(r"\d{1,2} \w{3,9} \d{4}", meta_date_text)
    meta_date_clear = search_date.group()
    sub_list.append(meta_date_clear) #<div class="h2 title horsAbstract print-black">Audience publique du mercredi 17 novembre 2021</div>
    main_div = soup.find("div", class_="content-page")
    text_var = main_div.get_text()
    #text_list.append(text_var)
    result_dispositif = re.search(r"PAR CES MOTIFS|PAR CES MOTIFS :|PAR CES MOTIFS, LA COUR|PARCESMOTIFS|P A R C E S M O T I F|P A R C E S M O T I F S :|P A R CE
    if result_dispositif:
        result_dispositif.start()
        par_ces_motifs = text_var[result_dispositif.start():]
        #line_break = re.search(r"\\|\n|\r", par_ces_motifs)
        hyphen = re.search(r" - ", par_ces_motifs)
        semi_colon = re.search(r";", par_ces_motifs)
        comma = re.search(r",", par_ces_motifs)
        dot = re.search(r"\.", par_ces_motifs)
        #if line_break:
        #    par_ces_motifs_split = re.split(r"\\|\n|\r", par_ces_motifs) How to split by line break although the text is a string?
        if hyphen:
            par_ces_motifs_split = re.split(r" - ", par_ces_motifs)
        elif semi_colon:
            par_ces_motifs_split = re.split(r";", par_ces_motifs)
        elif dot:
            par_ces_motifs_split = re.split(r"([A-L][N-Z][a-z])\.", par_ces_motifs)
        elif comma:
            par_ces_motifs_split = re.split(r",", par_ces_motifs)
    else:
```

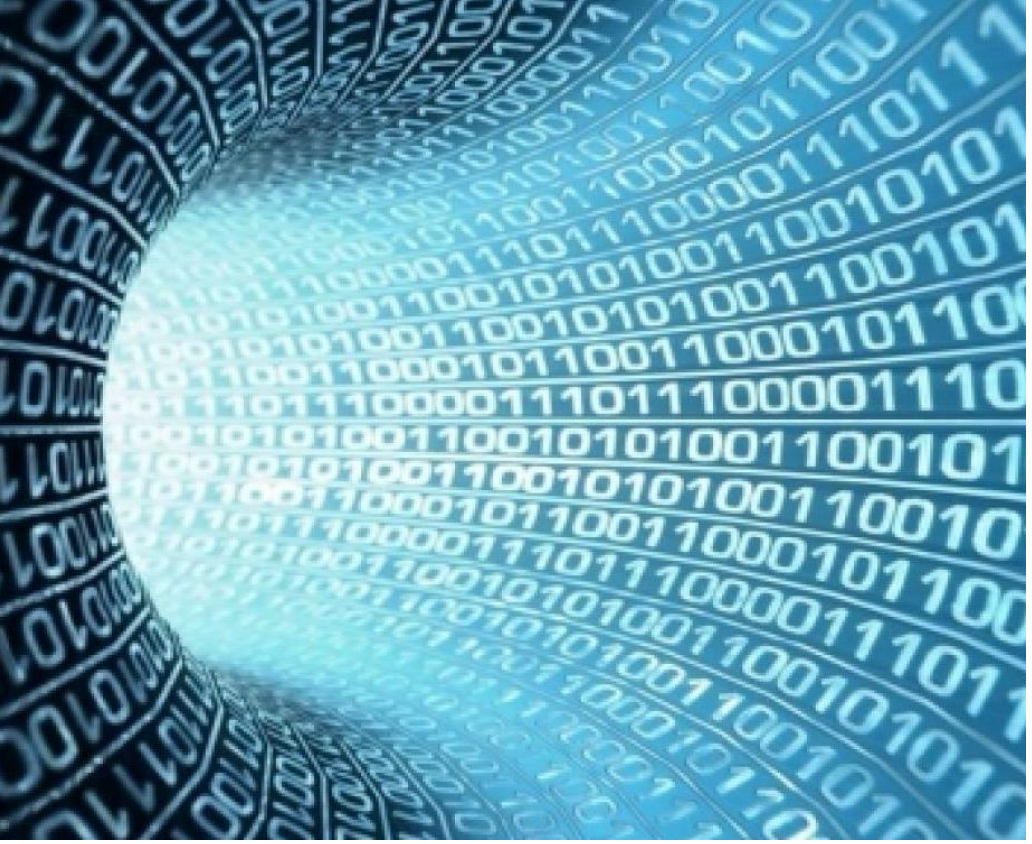
```
for line in par_ces_motifs_split:
    result = re.search("en application de l'article 700|au titre de l'article 700|au titre des frais irrépétibles|au titre des frais exclus des dépens|su
    result_bis = re.search("Vu l'article 700|En application de l'article 700|En vertu de l'article 700|Au titre de l'article 700", line)
    result_ter = re.search("article 700|irrépétibles|non répétibles|exclus des dépens|frais exposés", line)
    if result:
        result.start()
        extract = line[result.start()-100:result.end()+150]
        result_reject = re.search("avoir lieu|pas lieu|rejette|déboute|réserve|déboutées|rejeter|écarte|ne saurait|Pitre,Ecarte les demandes|ajoutant,Rej
        if result_reject:
            result_list.append("0 € - NOT GRANTED") #0 euros granted
        else:
            result_list.append(extract)
    elif result_bis:
        result_bis.start()
        extract = line[result_ter.start():result_ter.end()+250]
        result_reject = re.search("avoir lieu|pas lieu|rejette|déboute|réserve|déboutées|rejeter|écarte|ne saurait|Pitre,Ecarte les demandes|ajoutant,Rej
        if result_reject:
            result_list.append("0 € - NOT GRANTED") #0 euros granted
        else:
            result_list.append(extract)
    elif result_ter:
        result_ter.start()
        extract = line[result_ter.start()-150:result_ter.end()+150]
        result_damages = re.search("préjudice|intérêts|dommages-intérêts|dommages et intérêts", extract, re.IGNORECASE)
        result_reject = re.search("avoir lieu|pas lieu|rejette|déboute|réserve|déboutées|rejeter|écarte|ne saurait|Pitre,Ecarte les demandes|ajoutant,Rej
        if result_reject:
            result_list.append("0 € - NOT GRANTED") #0 euros granted
        elif result_damages:
            extract_1 = line[result_ter.start()-15:result_ter.end()+200]
            result_list.append(extract_1)
        else:
            extract_2 = line[result_ter.start()-100:result_ter.end()+100]
            result_list.append(extract_2) #or line ?
    else:
        #extract = "0 € - NO REF"
        result_list.append("0 € - NO REF") #0 reference to the preceding expressions
    print(result_list)
result_list_str = "".join(result_list)
reg=r"\b((?:\d+|\d{1,3}(?:[,.\s]\d{3}))(?:[,.\s]*\d+)?)\s?(?:euros?|€|€|¿|\x80|ä)" #r"\b((?:\d+|\d{1,3}(?:[,.\s]\d{3}))(?:[,.\s]*\d+)?)\d*00\s(?:euros?|€
amount_article_700_real = re.findall(reg, result_list_str) #if amount_article_700_real: ##number = amount_article_700_real.group()
```

```
    for number in amount_article_700_real:
        no_eur_number = re.sub(r"[euros|€|€|¿|\x80]", "", number, re.IGNORECASE) #la somme de 800, 00 euros
        no_space_character = re.sub(r"\xa0", " ", no_eur_number)
        no_space = re.sub(r" |\n", "", no_space_character)
        replaced_number = re.sub(r"[\.|,]\d{2,2}$", "", no_space)
        dot_digit = replaced_number.replace(".", " ")
        comma_digit = dot_digit.replace(",", " ")
        joined_digit = comma_digit.replace(" ", "")
        if re.search(r"\d{3,5}", joined_digit) and re.search(r"0{2,4}", joined_digit):#end_with_0_digit = re.sub(r"^([1-9]+)$", "0", joined_digi
            digit = int(joined_digit)
            digit_list.append(digit)
        else:
            digit_list.append(0)
    if len(digit_list)>0:
        max_amount = max(digit_list)
        print(max_amount)
        sub_list.append(max_amount)
    else:
        sub_list.append(0)
    #amount_article_700_list.append(amount_article_700_real)
    #sub_list.append(amount_article_700_real)
    #sub_list.append(digit)
    for result_item in result_list:
        if re.search(r"0 € - NOT GRANTED", result_item):
            comment_notgranted = re.sub(r"0 € - NOT GRANTED","Article 700 not granted", result_item)
            result_dflist.append(comment_notgranted)
        elif re.search(r"0 € - NO REF", result_item):
            delete_noref = re.sub(r"0 € - NO REF", "", result_item)
            result_dflist.append(delete_noref)
        else:
            result_dflist.append(result_item)
    sub_list.append(result_dflist)
    sub_list.append(url)
    df_list.append(sub_list)

#print(amount_article_700)
#print(amount_article_700_list)
#print(sub_list)
print(df_list)

df = pd.DataFrame(df_list, columns = ['jurisdiction', 'date', 'amount','comment', 'url'])
locale.setlocale(locale.LC_ALL, 'fr_FR.UTF-8')
df['date'] = pd.to_datetime(df['date'], format = "%d %B %Y")

print(df)
file_name = "Article_700_project_final_Code_des_assurances_FINALVERSION_23032022.xlsx"
df.to_excel(file_name)
```



```python
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import locale
import spacy
from collections import Counter
import seaborn as sb
df = pd.read_excel("Article_700_project_final_Code_
print(df)
```

III. The data frame

- 1.**Importation of panda and numpy** to manipulate the data and build the dataframe.

2.Observation of the **amount of the compensation** allocated for each case no matter of the cour d'appel

```
#ALL AMOUNTS IDENTIFIED FOR ALL COURS D'APPEL OVER THE WHOLE PERIOD

new_df = df[df["amount"]> 0]
new_df.index = pd.to_datetime(new_df.date, format='%Y-%m-%d') #.dt.year
new_df.index
new_df # 537 decisions
new_df.amount.mean()
new_df.resample("1Y").amount.mean()
new_df.resample("1Y").amount.mean().plot.area() #the average amount granted has increased overtime

# ALL ARTICLE 700 NOT GRANTED
df_0 = df.loc[df['comment'] == "['Article 700 not granted']"]
df_0
df_0.value_counts() # = 242 decisions that refused to grant an amount pursuant to Article 700
df_0.groupby("jurisdiction").size()

# ALL DECISIONS NOT REFERRING TO ARTICLE 700 or SIMILAR
df_noref = df.loc[df['comment'] == "['']"]
df_noref
df_noref.value_counts() # = 150 decisions that do not mention article 700
df_noref.groupby("jurisdiction").size()

#929 decisions in total out of 1600??? => IMPROVE
```

- 537 decisions granted more than 0 euros
- 242 decisions refused to grant an amount pursuant to article 700
- 150 decisions do not mention article 700

3.Analysis of the cours d'appel which allocated more than 100 euros per decision

```
df["jurisdiction"].value_counts()
df1 = df["jurisdiction"].value_counts() > 100
print(df1)

# 5 cours d'appel with most decisions'

#Cour d'appel d'Aix-en-Provence                178
#Cour d'appel de Bastia                         153
#Cour d'appel de Paris                          138
#Cour d'appel de Versailles                     114
#Cour d'appel de Lyon                           112
```

4. Focus on the cour d'appel de Paris

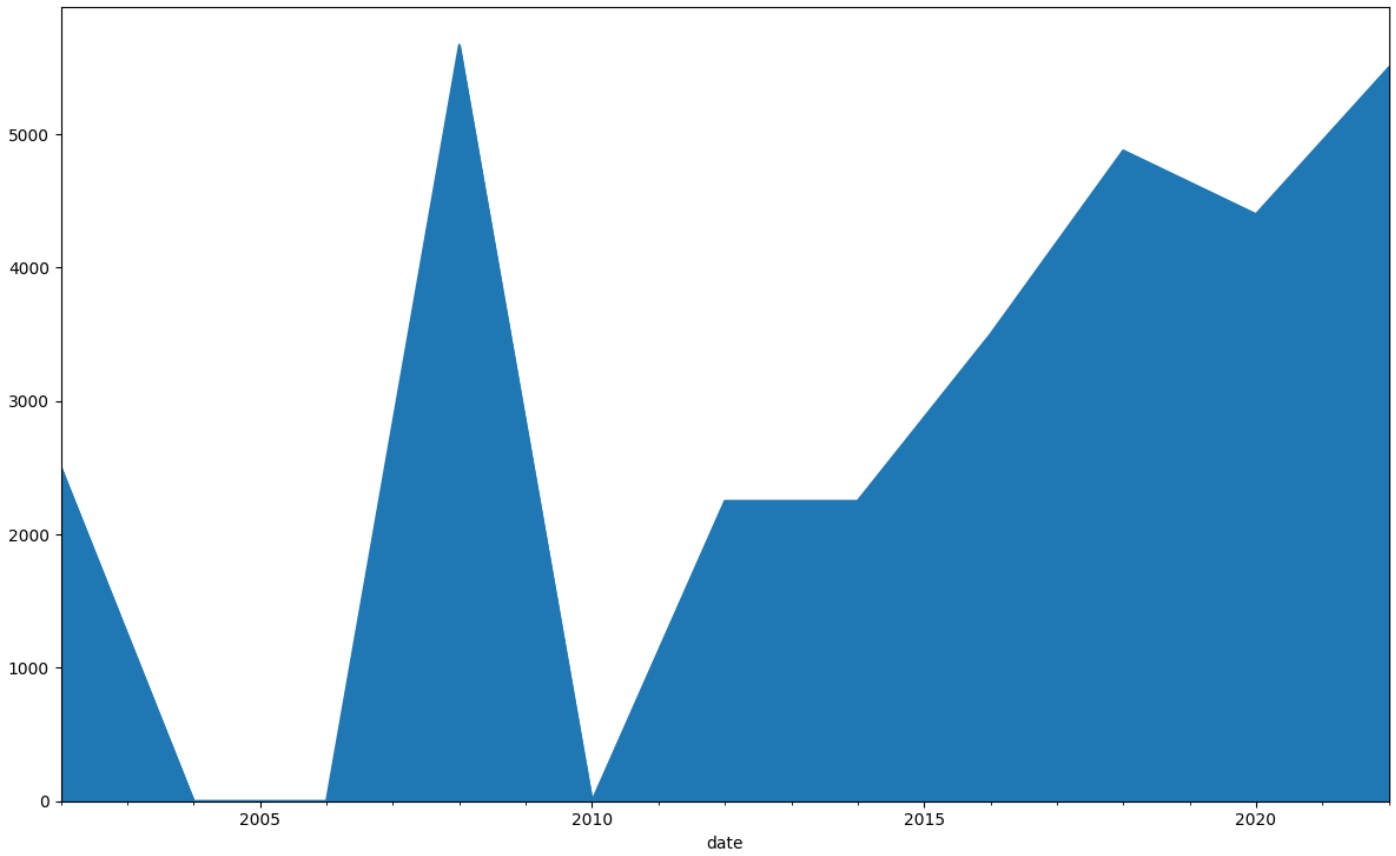
```
graph_list_Paris = []
df_Paris = df[df.jurisdiction.str.contains("Paris")]
df_Paris.index = pd.to_datetime(df_Paris.date, format='%Y-%m-%d').dt.year
df_Paris
df_Paris_positive = df_Paris[df_Paris["amount"]> 0] # 46 decisions for CA Paris  with a positive amount
df_Paris_positive
df_Paris_positive.amount.mean()
df_Paris_positive.amount.std()
df_Paris_positive.resample("2Y").amount.mean()
df_Paris_positive.resample("2Y").amount.mean().plot.area()# augmentation du montant moyen octroyé par la CA de Paris
df_Paris_positive.resample("2Y").amount.std()
df_Paris_positive.resample("2Y").amount.std().plot.area()

#For BàM:
Paris_list = df_Paris_positive.resample("2Y").amount.mean()
for item in Paris_list:
    graph_list_Paris.append(item)
graph_list_Paris
```

- Indexing by date of the decision
- Selection of the amounts superior to 0 euros
- Calculation of the average « *positive.amount.mean* » and standard deviation

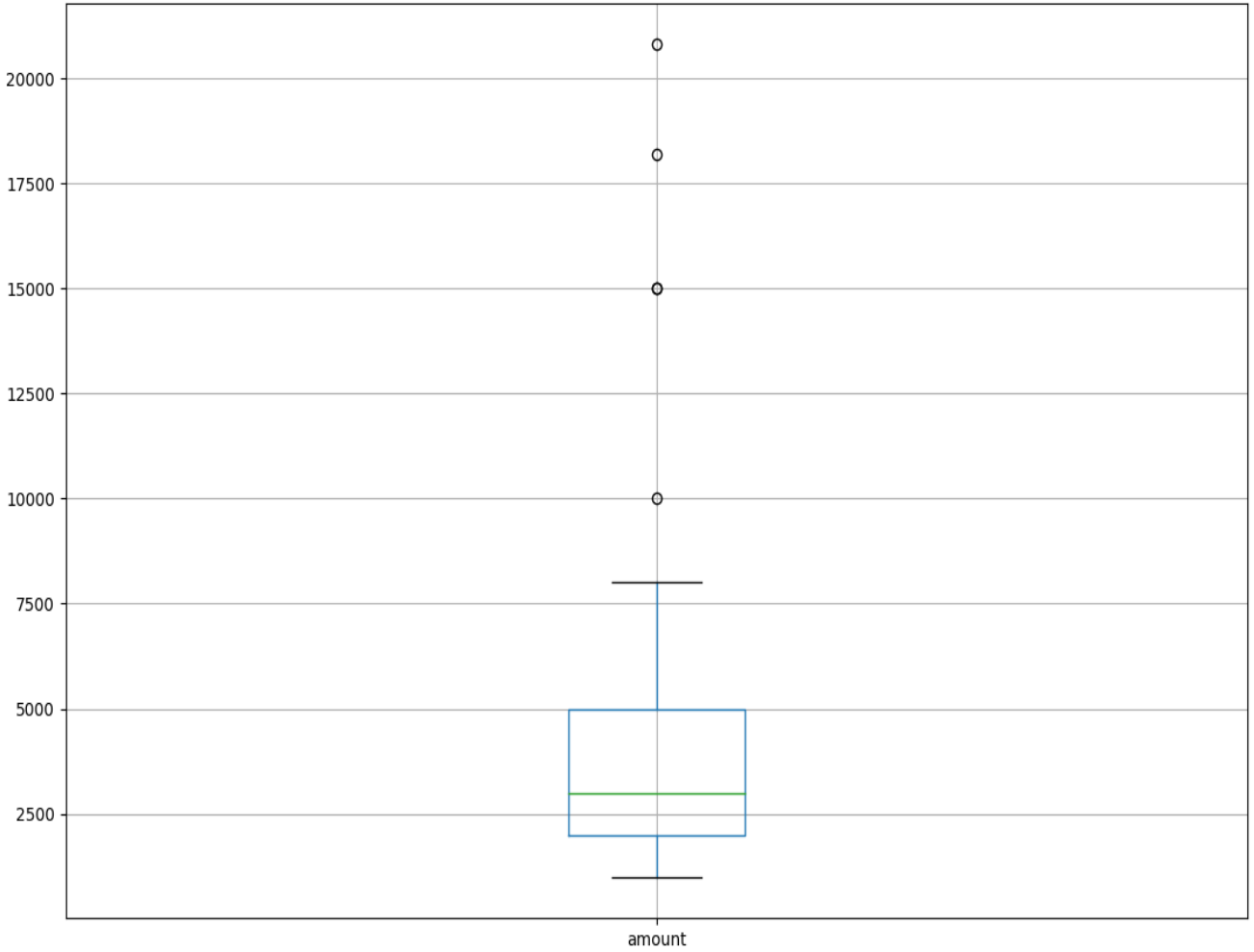
5. The standard deviation

- It allows to understand the distribution of the data of one serie.
- It measures how the data is distributed.
- The standard deviation has been calculated every two years.



6.The Boxplots

- The aim of a boxplot is to understand the distribution of the different data all around the median.
- For exemple the blue bloxplot:
  - The dash at the bottom : it is the minimum value
  - In the midddle : it is the median
  - The last dash : it means that all of the data are lower than 80



IV. Results & Interpretation

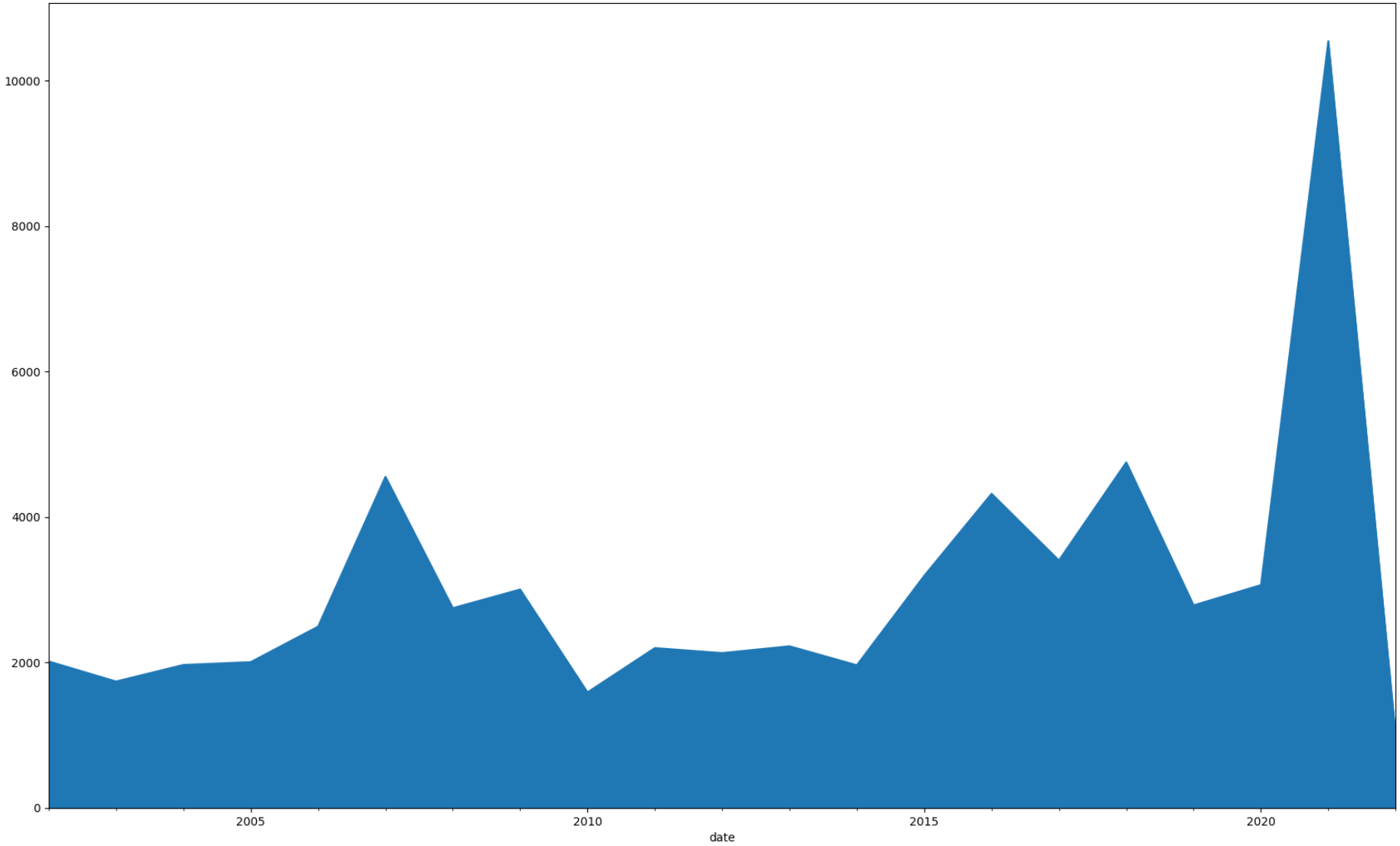
Global average of article 700 over 20 years:

3233.33 €

```
In [3]: new_df = df[df["amount"]> 0]
   ...: new_df.index = pd.to_datetime(new_df.date, format='%Y-%m-%d') #.dt.year
   ...: new_df.index
   ...: new_df # 537 decisions
   ...: new_df.amount.mean()
   ...: new_df.resample("1Y").amount.mean()
   ...: new_df.resample("1Y").amount.mean().plot.area() #the average amount granted has increased overtime
   ...:
   ...:
Out[3]: <AxesSubplot:xlabel='date'>
In [4]: new_df.amount.mean()
Out[4]: 3233.3333333333335
```

Average of article 700 granted per year

```
In [5]: new_df.resample("1Y").amount.mean()
Out[5]:
date
2002-12-31     2016.666667
2003-12-31     1740.740741
2004-12-31     1968.421053
2005-12-31     2007.142857
2006-12-31     2495.454545
2007-12-31     4555.339806
2008-12-31     2749.166667
2009-12-31     3005.263158
2010-12-31     1590.909091
2011-12-31     2200.000000
2012-12-31     2131.578947
2013-12-31     2225.000000
2014-12-31     1964.285714
2015-12-31     3193.333333
2016-12-31     4321.428571
2017-12-31     3405.000000
2018-12-31     4753.846154
2019-12-31     2787.500000
2020-12-31     3066.666667
2021-12-31    10545.454545
2022-12-31     1000.000000
Freq: A-DEC, Name: amount, dtype: float64
```



V- Difficulties & Improvement

- No standardization of courts decision
- Taking into account more decisions
- Linking amounts to requests
- Apply analysis to other law fields
- We focused our analysis only on the decisions held by the main *cours d'appel* but we could have taken all the *cours d'appel*
- Loi normale to approximate the 95% distribution range of the amount granted (+/- 2 standard deviations from the mean)

Thank you !