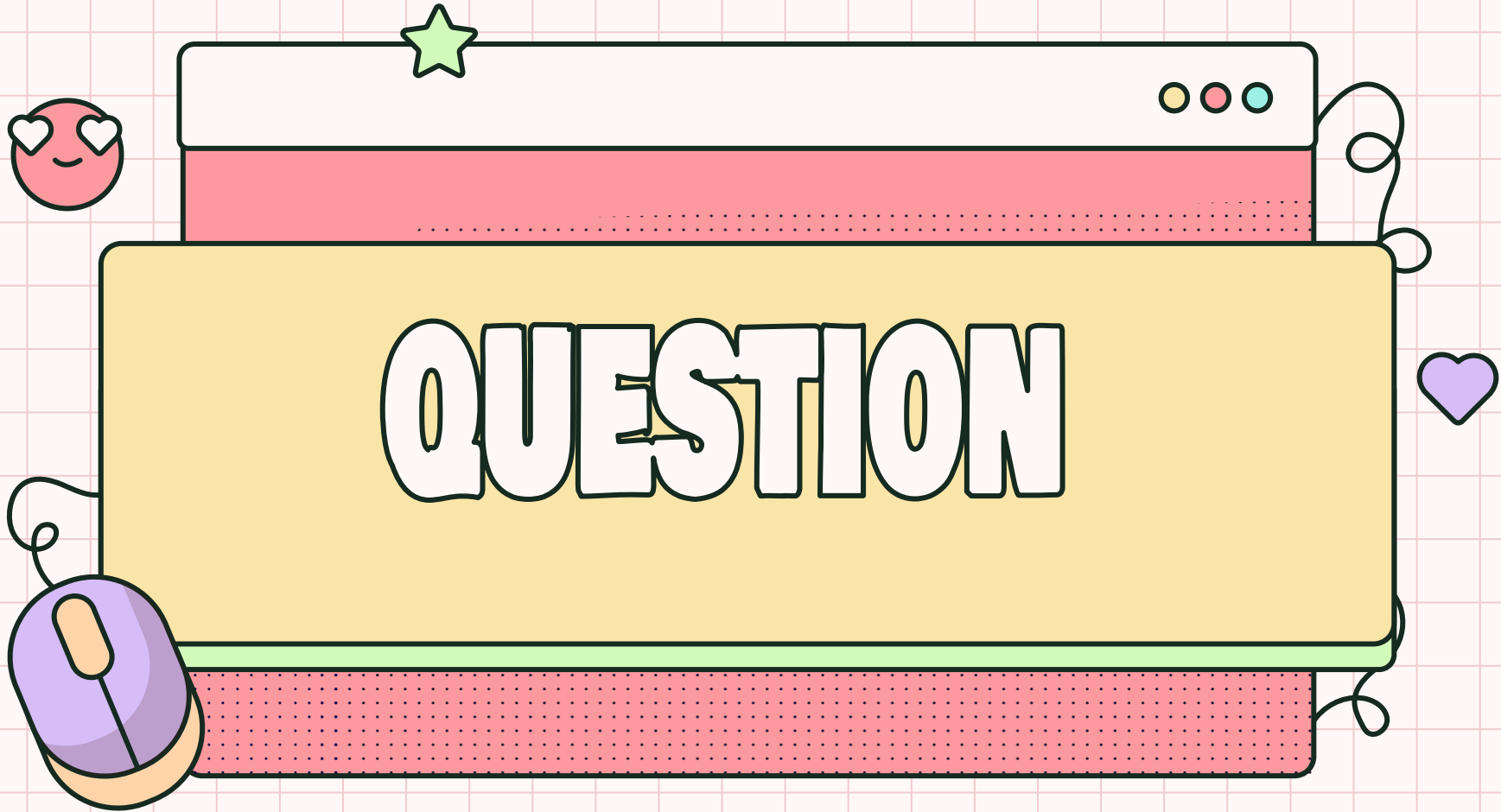




**EVOLUTION OF THE**

# **PLACE OF WOMEN IN COURTS**

Henri Carlier, Léa Colomb, Alice  
Leclercq & Mélanie Tonani





# STEPS

**01**

## RESEARCH

Goal of the work  
conducted

**02**

## SCRAPING

Collect data from  
LegiFrance since 1953

**03**

## DATA TREATMENT

Organising the data in  
order to exploit them

**04**

## ANALYSIS

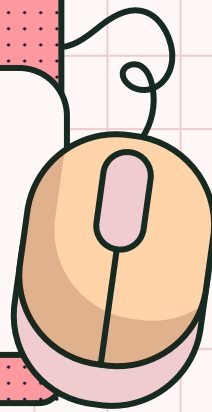
Conclude on women's  
place in jurisdiction



01

# RESEARCH

To analyze, using a code written in Python, the evolution of the presence of female presidents in the decisions of the Cour cassation



# 02

## SCRAPPING

### METHODE



### SCRAPPING

Using the website legifrance  
Decision from 1946



### EXTRACTION

Extraction date and  
jurisdiction as well as the  
information of the judge



### DATA FRAME

Creation of a dataframe in  
csv format meeting certain  
analysis criteria: date,  
jurisdiction, judge

# IMPORTATION OF USEFUL LIBRARIES



```
] : from bs4 import BeautifulSoup
    from time import time, sleep
    from selenium import webdriver
    from selenium.webdriver.common.by import By
    import concurrent.futures as cf
    import numpy as np
    import pandas as pd
    import json
    import matplotlib.pyplot as plt
    import threading
    import requests
    import re
```



# INITIALIZE THE CHROME DRIVER

```
def getDriver():  
    options = webdriver.ChromeOptions()  
    options.binary_location = "C:\\Program Files\\BraveSoftware\\Brave-Browser\\Application\\brave.exe"  
    options.add_argument("--headless")  
    return webdriver.Chrome(chrome_options=options, executable_path="C:\\Windows\\chromedriver.exe")
```

```
base_url = "https://www.legifrance.gouv.fr/search/juri?tab_selection=juri&searchField=ALL&query=*&searchType=ALL&dateDecision=01%2F01%2F1946+%3E+01%2F01%2F2023&cassPubliBulletin=T&cassPubliBulletin=F&cassDecision=ARRET&cassFormation=ASS  
EMBLEE_PLENIER&cassFormation=CHAMBRE_MIXTE&cassFormation=CHAMBRES_REUNIES&cassFormation=CHAMBRE_CIVILE_1&cassFormation=CHAMBRE_CIVILE_2&cassFormation=CHAMBRE_CIVILE_3&cassFormation=CHAMBRE_COMMERCIALE&cassFormation=CHAMBRE_SOCIALE&cass  
Formation=CHAMBRE_CRIMINELLE&cassFormation=COMMISSION_REEXAMEN&cassFormation=COMMISSION_REPARATION_DETENTION&cassFormation=COMMISSION_REVISION&cassFormation=COUR_REVISION&cassFormation=JURIDICTION_NATIONALE_LIBERTE_CONDITIONNELLE&cassFo  
rmation=ORDONNANCE_PREMIER_PREIDENT&cassDecisionAttaquee=COMMISSION_INDEMNISATION_VICTIMES_INFRACTIONS&cassDecisionAt  
taquee=CONSEIL_PRUDHOMME&cassDecisionAttaquee=COUR_APPEL&cassDecisionAttaquee=COUR_ASSISES&cassDecisionAttaquee=COUR_C  
ASSATION&cassDecisionAttaquee=COUR_JUSTICE_REPUBLIQUE&cassDecisionAttaquee=COUR_NATIONAL_INCAPACITE_TARIFICATION&cassD  
ecisionAttaquee=TRIBUNAL_CORRECTIONNEL&cassDecisionAttaquee=TRIBUNAL_COMMERCE&cassDecisionAttaquee=TRIBUNAL_Grande_INS  
TANCE&cassDecisionAttaquee=TRIBUNAL_POLICE&cassDecisionAttaquee=TRIBUNAL_Premiere_Instance&cassDecisionAttaquee=TRIBUN  
AL_AFFAIRES_SECURITE_SOCIALE&cassDecisionAttaquee=TRIBUNAL_FORCES_ARMEEES&cassDecisionAttaquee=TRIBUNAL_Instance&cassDe  
cisionAttaquee=TRIBUNAL_CONTENTIEUX_INCAPACITE&cassDecisionAttaquee=TRIBUNAL_MARITIME_COMMERCIAL&cassDecisionAttaquee=  
TRIBUNAL_PARITAIRE_BAUX_RURAUX&cassDecisionAttaquee=TRIBUNAL_SUPERIEURS_APPEL&juridictionJudiciaire=Cour+de+cassation&  
typePagination=DEFAULT&sortValue=DATE_ASC&pageSize=100&page=1&tab_selection=juri#juri"  
driver = getDriver()  
driver.get(base_url)  
#soup = BeautifulSoup(driver.page_source, 'html.parser')
```

**definition of a getDriver() function and use of Selenium to launch a web browser and access a specific URL.**

## NUMBER OF PAGES TO BROWSE:



```
nb_pages = int(
    driver.find_element(By.CLASS_NAME, "container-pager")\
        .find_elements(By.CLASS_NAME, "pager-item")[3]\
        .find_element(By.TAG_NAME, "span").get_attribute("innerHTML")
)
```

1 2 3 ... 4691



## DEFINITION OF A FUNCTION EXTRACTYEAR WHICH TAKES A STRING AS INPUT AND RETURNS THE YEAR FOUND IN THE STRING



```
def extractYear(sentence):  
    ys = [str(y) for y in range(1946, 2024)]  
    for y in ys:  
        if (sentence.find(y) > 0):  
            return int(y)
```

Date ou période de décision

Format JJ/MM/AAAA, MM/AAAA ou AAAA ⓘ

01/01/1946



01/01/2023



↺ RÉINITIALISER

Valider



## LIST OF PAGES TO BROWSE



```
pages = [f"https://www.legifrance.gouv.fr/search/juri?tab_selection=juri&searchField=ALL&query=*&searchType=ALL&dateDe  
cision=01%2F01%2F1946+%3E+01%2F01%2F2023&cassPubliBulletin=T&cassPubliBulletin=F&cassDecision=ARRET&cassFormation=ASSE  
MBLEE_PLENIERE&cassFormation=CHAMBRE_MIXTE&cassFormation=CHAMBRES_REUNIES&cassFormation=CHAMBRE_CIVILE_1&cassFormation  
=CHAMBRE_CIVILE_2&cassFormation=CHAMBRE_CIVILE_3&cassFormation=CHAMBRE_COMMERCIALE&cassFormation=CHAMBRE_SOCIALE&cassF  
ormation=CHAMBRE_CRIMINELLE&cassFormation=COMMISSION_REEXAMEN&cassFormation=COMMISSION_REPARATION_DETENTION&cassFormat  
ion=COMMISSION_REVISION&cassFormation=COUR_REVISION&cassFormation=JURIDICTION_NATIONALE_LIBERTE_CONDITIONNELLE&cassFor  
mation=ORDONNANCE_PREMIER_PREIDENT&cassDecisionAttaquee=COMMISSION_INDEMNISATION_VICTIMES_INFRACTIONS&cassDecisionAtt  
aquee=CONSEIL_PRUDHOMME&cassDecisionAttaquee=COUR_APPEL&cassDecisionAttaquee=COUR_ASSISES&cassDecisionAttaquee=COUR_CA  
SSATION&cassDecisionAttaquee=COUR_JUSTICE_REPUBLIQUE&cassDecisionAttaquee=COUR_NATIONAL_INCAPACITE_TARIFICATION&cassDe  
cisionAttaquee=TRIBUNAL_CORRECTIONNEL&cassDecisionAttaquee=TRIBUNAL_COMMERCE&cassDecisionAttaquee=TRIBUNAL_Grande_INST  
ANCE&cassDecisionAttaquee=TRIBUNAL_POLICE&cassDecisionAttaquee=TRIBUNAL_Premiere_INSTANCE&cassDecisionAttaquee=TRIBUNAL  
L_AFFAIRES_SECURITE_SOCIALE&cassDecisionAttaquee=TRIBUNAL_FORCES_ARMEEES&cassDecisionAttaquee=TRIBUNAL_INSTANCE&cassDec  
isionAttaquee=TRIBUNAL_CONTENTIEUX_INCAPACITE&cassDecisionAttaquee=TRIBUNAL_MARITIME_COMMERCIAL&cassDecisionAttaquee=T  
RIBUNAL_PARITAIRE_BAUX_RURAUX&cassDecisionAttaquee=TRIBUNAL_SUPERIEURS_APPEL&jurisdictionJudiciaire=Cour+de+cassation&t  
ypePagination=DEFAULT&sortValue=DATE_ASC&pageSize=100&page={i}&tab_selection=juri#juri" for i in range(1, nb_pages+1)]  
num_threads = 10  
sub_pages = [pages[i:i+num_threads] for i in range(0, len(pages), num_threads)]
```

## FUNCTION THAT SCRAPES USEFUL INFORMATION (LINK AND YEAR), PAGE BY PAGE



```
def retrieve(url):
    to_add = []
    dr = getDriver()
    dr.get(url)
    articles = dr.find_elements(By.CLASS_NAME, "result-item")
    for article in articles:
        a = article.find_element(By.TAG_NAME, 'a')
        link = a.get_attribute("href")
        title = a.get_attribute("innerText")
        to_add.append((link, extractYear(title)))
    dr.quit()
    return to_add
```



## CODE THAT RETRIEVES LINKS AND YEARS.



```
with cf.ThreadPoolExecutor(max_workers=10) as executor:
    futures = [executor.submit(retrieve, url) for url in pages]
    for future in cf.as_completed(futures):
        try:
            results = [future.result() for future in futures]
            with open("data.json", "w") as json_file:
                json.dump(results, json_file)
        except requests.ConnectTimeout:
            pass
```

0 texte(s) trouvé(s)

Date de décision chronologique ▼

Afficher 10 résultats par page ▼

La recherche est limitée aux 10 000 premiers éléments, merci d'affiner votre recherche.

[Retourner à la recherche](#)

**Unfortunately, the government site complicates our task once again by allowing us access to the first 10,000 results only.**

## DATA FORMATTING IN A PANDAS DATAFRAME



```
with open("data.json", "r") as json_file:
    data = json.load(json_file)
data = [sub for sub in data if len(sub) > 0]
flat = []
for sub in data:
    for item in sub:
        flat.append(item)
df = pd.DataFrame(flat, columns=['url', 'year'])
df.head()


to_concat = []
for y in set(df['year']):
    subdf = df[df['year'] == y]
    to_concat.append(subdf.sample(n = min(100, len(subdf)), random_state = 42))
normalized = pd.concat(to_concat, axis=0).reset_index(drop=True)
```

**Limited to a maximum of 100 data per year**

## SCRAPE THE LINKS OF THE JUDGMENTS WE KEPT TO REMOVE THE DATE, THE JURISDICTION AND THE GENDER OF THE PRESIDENT



```
results = {'date': [], 'gender': [], 'jurisdiction': []}
```



```
driver = getDriver()
patternF = r"Mme "
patternM = r"M\.|M "
patternDate = r"\b\d{1,2}\s+(?:janvier|février|mars|avril|mai|juin|juillet|août|septembre|octobre|novembre|décembre)\s+\d{4}\b"
for url in normalized['url']:
    driver.get(url)
    try:
        jurisdiction = driver.find_elements(By.CLASS_NAME, "horsAbstract")[0]\
            .get_attribute("innerText")
        jurisdiction = jurisdiction[jurisdiction.find("-")+2:]
    except:
        jurisdiction = ''
    try:
        pdt = driver.find_elements(By.CLASS_NAME, 'frame-block')[1]\
            .find_elements(By.TAG_NAME, 'div')[0]\
            .find_elements(By.TAG_NAME, 'dd')[0]\
            .get_attribute("innerText")
        if re.findall(patternF, pdt, re.IGNORECASE):
            gender = 'F'
        elif re.findall(patternM, pdt, re.IGNORECASE):
            gender = 'M'
        else:
            gender = ''
    except:
        gender = ''
    try:
        date = driver.find_elements(By.CLASS_NAME, "horsAbstract")[1]\
            .get_attribute("innerText")
        date = re.findall(patternDate, date, re.IGNORECASE)[0]
    except:
        date = ''
    results['date'].append(date)
    results['gender'].append(gender)
    results['jurisdiction'].append(jurisdiction)
```

```

for annee in range(1947, 2024):
    base_url = f"https://www.legifrance.gouv.fr/search/juri?tab_selection=juri&sear
archField=ALL&query={searchType}&ALLdateDecision=01%2F01%2F(annee)+%3E+01%2F01%
2F(annee+1)}&scsPubliBulletin=F&scsDecision=ARRET&scsForma
tion=ASSEMBLEE_PLENIERE&scsFormation=CHAMBRE_MIXTE&scsFormation=CHAMBRES_REUNI
ES&scsFormation=CHAMBRE_CIVILE_1&scsFormation=CHAMBRE_CIVILE_2&scsFormation=C
HAMBRE_CIVILE_3&scsFormation=CHAMBRE_COMMERCIALE&scsFormation=CHAMBRE_SOCIALE&
scsFormation=CHAMBRE_CRIMINELLE&scsFormation=COMMISSION_REEXAMEN&scsFormation
=COMMISSION_REPARATION_DETENTION&scsFormation=COMMISSION_REVISION&scsFormation
=COUR_REVISION&scsFormation=JURIDICTION_NATIONALE_LIBERTE_CONDITIONNELLE&scsFo
rmation=ORDONNANCE_PREMIER_PRESIDENT&scsDecisionAttache=COMMISSION_INDEMNISATI
ON_VICTIMES_INFRACTIONS&scsDecisionAttache=CONSEIL_PRUDHOMME&scsDecisionAttac
ue=COUR_APPEL&scsDecisionAttache=COUR_ASSISES&scsDecisionAttache=COUR_CASSA
TION&scsDecisionAttache=COUR_JUSTICE_REPUBLIQUE&scsDecisionAttache=COUR_NATI
ONAL_INCAPACITE_TARIFICATION&scsDecisionAttache=TRIBUNAL_CORRECTIONNEL&scsDec
isionAttache=TRIBUNAL_COMMERCE&scsDecisionAttache=TRIBUNAL_Grande_Instance&ca
ssDecisionAttache=TRIBUNAL_POLICE&scsDecisionAttache=TRIBUNAL_Premiere_Instance
CE&scsDecisionAttache=TRIBUNAL_AFFAIRES_SECURITE_SOCIALE&scsDecisionAttache=TR
IBUNAL_FORCES_ARMEEES&scsDecisionAttache=TRIBUNAL_INSTANCE&scsDecisionAttac
ue=TRIBUNAL_CONTENTIEUX_INCAPACITE&scsDecisionAttache=TRIBUNAL_MARITIME_COMMER
CIAL&scsDecisionAttache=TRIBUNAL_PARITAIRE_BAUX_RURAUX&scsDecisionAttache=TR
IBUNAL_SUPERIEURS_APPELS_jurisdictionJudiciaire=Cour+de+cassation+typePagination=DE
FAULT&sortValue=DATE_ASC&pageSize=100&page={i}&tab_selection=juri#juri"
    driver = getDriver()
    driver.get(base_url)
    number_pattern = r'\d+'
    nb_pages_string = driver.find_element(By.CLASS_NAME, "nb-result").get_attribu
te("innerHTML")
    nb_pages = ceil(int(re.findall(number_pattern, nb_pages_string)[0])/100)
    driver.quit()
    pages = [f"https://www.legifrance.gouv.fr/search/juri?tab_selection=juri&sear
chField=ALL&query={searchType}&ALLdateDecision=01%2F01%2F(annee)+%3E+01%2F01%2F
(annee+1)}&scsPubliBulletin=F&scsDecision=ARRET&scsFormati
on=ASSEMBLEE_PLENIERE&scsFormation=CHAMBRE_MIXTE&scsFormation=CHAMBRES_REUNIES
&scsFormation=CHAMBRE_CIVILE_1&scsFormation=CHAMBRE_CIVILE_2&scsFormation=CHA
MBRE_CIVILE_3&scsFormation=CHAMBRE_COMMERCIALE&scsFormation=CHAMBRE_SOCIALE&ca
ssFormation=CHAMBRE_CRIMINELLE&scsFormation=COMMISSION_REEXAMEN&scsFormation=C
OMMISSION_REPARATION_DETENTION&scsFormation=COMMISSION_REVISION&scsFormation=C
OUR_REVISION&scsFormation=JURIDICTION_NATIONALE_LIBERTE_CONDITIONNELLE&scsForma
tion=ORDONNANCE_PREMIER_PRESIDENT&scsDecisionAttache=COMMISSION_INDEMNISATION
_VICTIMES_INFRACTIONS&scsDecisionAttache=CONSEIL_PRUDHOMME&scsDecisionAttache
=COUR_APPEL&scsDecisionAttache=COUR_ASSISES&scsDecisionAttache=COUR_CASSATI
ON&scsDecisionAttache=COUR_JUSTICE_REPUBLIQUE&scsDecisionAttache=COUR_NATION
AL_INCAPACITE_TARIFICATION&scsDecisionAttache=TRIBUNAL_CORRECTIONNEL&scsDecis
ionAttache=TRIBUNAL_COMMERCE&scsDecisionAttache=TRIBUNAL_Grande_Instance&scs
DecisionAttache=TRIBUNAL_POLICE&scsDecisionAttache=TRIBUNAL_Premiere_Instance
&scsDecisionAttache=TRIBUNAL_AFFAIRES_SECURITE_SOCIALE&scsDecisionAttache=TR
IBUNAL_FORCES_ARMEEES&scsDecisionAttache=TRIBUNAL_INSTANCE&scsDecisionAttache
=TRIBUNAL_CONTENTIEUX_INCAPACITE&scsDecisionAttache=TRIBUNAL_MARITIME_COMMERCI
AL&scsDecisionAttache=TRIBUNAL_PARITAIRE_BAUX_RURAUX&scsDecisionAttache=TR
IBUNAL_SUPERIEURS_APPELS_jurisdictionJudiciaire=Cour+de+cassation+typePagination=DE
FAULT&sortValue=DATE_ASC&pageSize=100&page={i}&tab_selection=juri#juri" for i in
range(1, min(100, nb_pages+1))]
    with cf.ThreadPoolExecutor(max_workers=10) as executor:
        futures = [executor.submit(retrieve, url) for url in pages]
        for future in cf.as_completed(futures):
            try:
                results = [future.result() for future in futures]
                with open(f"data_{annee}.json", "w") as json_file:
                    json.dump(results, json_file)

```

```

except requests.ConnectTimeout:
    pass

```

Formatage des datas dans un DataFrame pandas

In [7]:

```

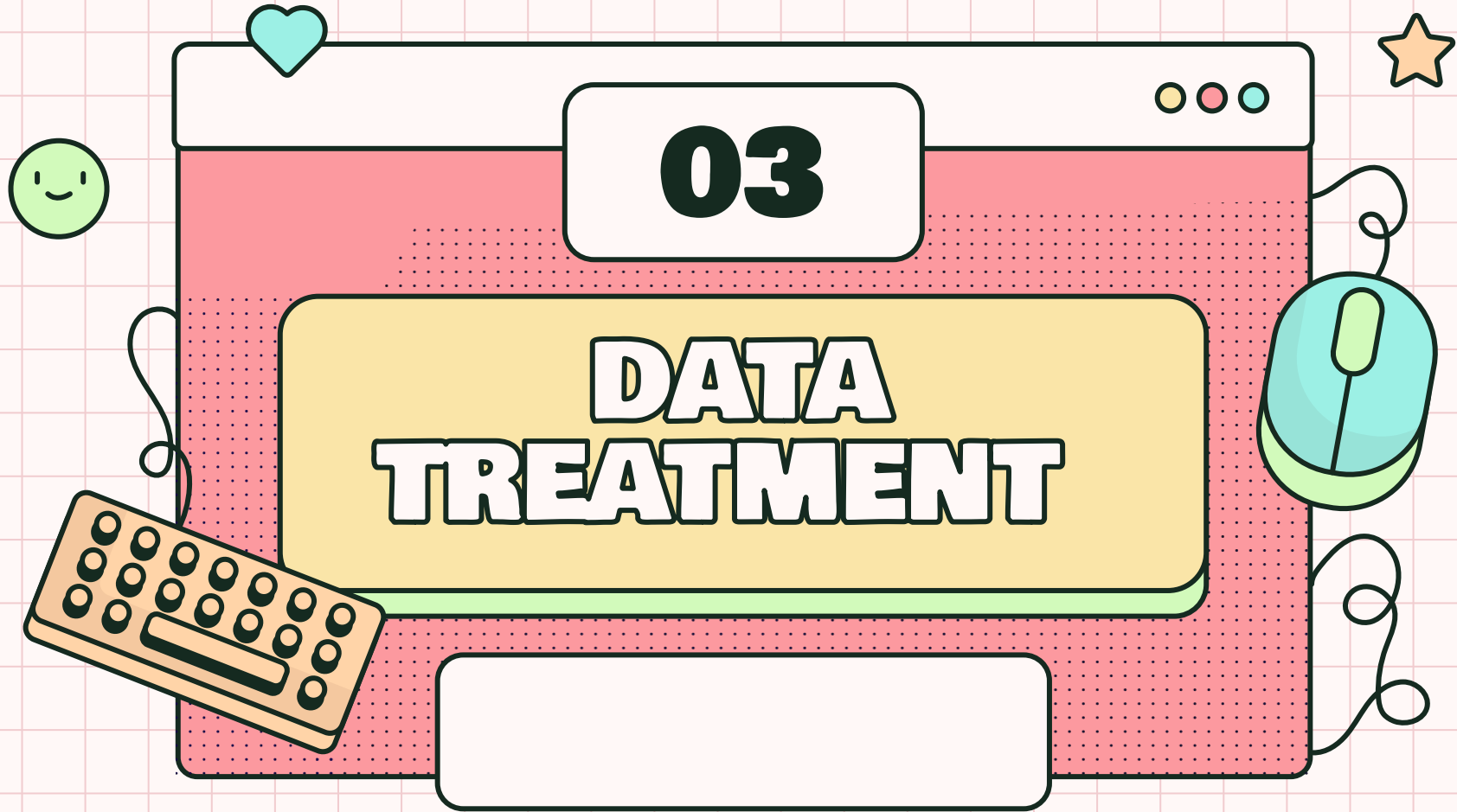
to_concat = []
for year in range(1947, 2024):
    if not year == 1952:
        with open(f"data_{year}.json", "r") as json_file:
            data = json.load(json_file)
            data = [sub for sub in data if len(sub) > 0]
            flat = []
            for sub in data:
                for item in sub:
                    flat.append(item)
            df = pd.DataFrame(flat, columns=['url', 'year'])
            to_concat.append(df.sample(n = min(100, len(df)), random_state = 42))

```

On se limite à maximum 100 données par an

In [8]:

```
normalized = pd.concat(to_concat, axis=0).reset_index(drop=True)
```





## THE RESULTS ARE FORMATTED IN A DATAFRAME AND SAVED IN A CSV FILE

In [49]:

```
print(f"Nombre de jugements étudiés: {len(results)}\nNombre de jugements où le  
sexe du président est indiqué: {len(results[results['gender'] != ''])}")
```

Nombre de jugements étudiés: 689

Nombre de jugements où le sexe du président est indiqué: 571

```
results = pd.DataFrame(results)  
results.tail()  
results.to_csv("dataframe.csv", index=False)
```

## RESULTS

```
results[results['gender'] == 'F']
```

	date	gender	jurisdiction
58	28 octobre 1957	F	Chambre sociale



## CLEANING

```
In [ ]:
```

```
clean = results[results['gender'] != '']
```

```
In [ ]:
```

```
serie = clean.groupby(clean.index.year)['gender'].sum()  
serie = serie.apply(lambda val: val.count('F'))
```

```
In [ ]:
```

```
mois = {" janvier ": "/01/", " février ": "/02/", " mars ": "/03/", " avril ":  
        "/04/", " mai ": "/05/", " juin ": "/06/", " juillet ": "/07/", " août ": "/08/  
        /", " septembre ": "/09/", " octobre ": "/10/", " novembre ": "/11/", " décemb  
        re ": "/12/"}
```

```
def convert(dt):  
    for m in mois:  
        if m in dt:  
            return dt.replace(m, mois[m])
```

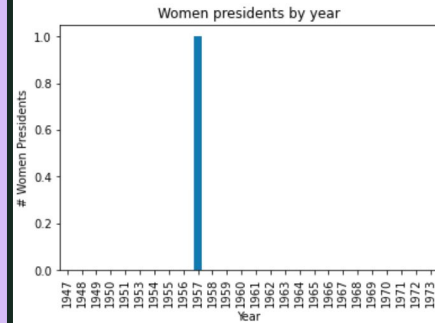
```
In [ ]:
```

```
clean.loc[:, 'date'] = clean['date'].apply(convert)  
clean.loc[:, 'date'] = pd.to_datetime(clean['date'], format='%d/%m/%Y')  
clean.set_index('date', inplace=True)
```

# RESULTS

In [50]:

```
serie.plot.bar()  
plt.xlabel('Year')  
plt.ylabel('# Women Presidents')  
plt.title('Women presidents by year')  
plt.show()
```



## EXTENDING OUR SCOPE AFTER 1973

```
In [33]:
def retrieve2(url):
    res = {'date': '', 'gender': '', 'jurisdiction': ''}
    driver = getDriver()
    patternF = r"Mme "
    patternM = r"M\.[M "
    patternDate = r"\b\d{1,2}\s+(?:janvier|février|mars|avril|mai|juin|juillet|a
    out|septembre|octobre|novembre|décembre)\s+\d{4}\b"
    driver.get(url)
    try:
        jurisdiction = driver.find_elements(By.CLASS_NAME, "horsAbstract")[0]\
            .get_attribute("innerText")
        jurisdiction = jurisdiction[jurisdiction.find("-")+2:]
    except Exception as e:
        jurisdiction = ''
        print("An error occurred:", str(e))
    try:
        pdt = driver.find_elements(By.CLASS_NAME, 'frame-block')[1]\
            .find_elements(By.TAG_NAME, 'div')[0]\
            .find_elements(By.TAG_NAME, 'dd')[0]\
            .get_attribute("innerText")
        if re.findall(patternF, pdt, re.IGNORECASE):
            gender = 'F'
        elif re.findall(patternM, pdt, re.IGNORECASE):
            gender = 'M'
        else:
            gender = ''
    except Exception as e:
        gender = ''
        print("An error occurred:", str(e))
    try:
        date = driver.find_elements(By.CLASS_NAME, "horsAbstract")[1]\
            .get_attribute("innerText")
        date = re.findall(patternDate, date, re.IGNORECASE)[0]
    except Exception as e:
        date = ''
        print("An error occurred:", str(e))
    driver.quit()
    res['date'] = date
    res['gender'] = gender
    res['jurisdiction'] = jurisdiction
    return res
```

In [ ]:

```
with cf.ThreadPoolExecutor(max_workers=10) as executor:
    futures = [executor.submit(retrieve2, url) for url in normalized[normalized
    ['year'] >= 1973]['url']]
    for future in cf.as_completed(futures):
        try:
            results = [future.result() for future in futures]
            with open(f"final_data.json", "w") as json_file:
                json.dump(results, json_file)
        except requests.ConnectTimeout:
            pass
```

## EXTENDING OUR SCOPE AFTER 1973

```
In [33]:
def retrieve2(url):
    res = {'date': '', 'gender': '', 'jurisdiction': ''}
    driver = getDriver()
    patternF = r"Mme "
    patternM = r"M\.[M "
    patternDate = r"\b\d{1,2}\s+(?:janvier|février|mars|avril|mai|juin|juillet|a
out|septembre|octobre|novembre|décembre)\s+\d{4}\b"
    driver.get(url)
    try:
        jurisdiction = driver.find_elements(By.CLASS_NAME, "horsAbstract")[0]\
            .get_attribute("innerText")
        jurisdiction = jurisdiction[jurisdiction.find("-")+2:]
    except Exception as e:
        jurisdiction = ''
        print("An error occurred:", str(e))
    try:
        pdt = driver.find_elements(By.CLASS_NAME, 'frame-block')[1]\
            .find_elements(By.TAG_NAME, 'div')[0]\
            .find_elements(By.TAG_NAME, 'dd')[0]\
            .get_attribute("innerText")
        if re.findall(patternF, pdt, re.IGNORECASE):
            gender = 'F'
        elif re.findall(patternM, pdt, re.IGNORECASE):
            gender = 'M'
        else:
            gender = ''
    except Exception as e:
        gender = ''
        print("An error occurred:", str(e))
    try:
        date = driver.find_elements(By.CLASS_NAME, "horsAbstract")[1]\
            .get_attribute("innerText")
        date = re.findall(patternDate, date, re.IGNORECASE)[0]
    except Exception as e:
        date = ''
        print("An error occurred:", str(e))
    driver.quit()
    res['date'] = date
    res['gender'] = gender
    res['jurisdiction'] = jurisdiction
    return res
```

In [ ]:

```
with cf.ThreadPoolExecutor(max_workers=10) as executor:
    futures = [executor.submit(retrieve2, url) for url in normalized[normalized
['year'] >= 1973]['url']]
    for future in cf.as_completed(futures):
        try:
            results = [future.result() for future in futures]
            with open(f"final_data.json", "w") as json_file:
                json.dump(results, json_file)
        except requests.ConnectTimeout:
            pass
```

# RESULTS

```
results = pd.DataFrame(results)
results.to_csv("dataframe.csv", index=False)
```

**565 women president over 4619 decisions  
between 1973 and 2023**

Out[40]:

	date	gender	jurisdiction
0	04 avril 1973	M	Chambre civile 2
1	13 novembre 1973	M	Chambre criminelle
2	14 février 1973	M	Chambre commerciale
3	03 décembre 1973	M	Chambre commerciale
4	14 février 1973	M	Chambre civile 3
...	...	...	...
5095	01 mars 2023	M	Chambre commerciale
5096	15 février 2023	M	Chambre commerciale
5097	18 janvier 2023	F	Chambre sociale
5098	25 janvier 2023	F	Chambre civile 3
5099	26 janvier 2023	M	Chambre civile 2

5100 rows x 3 columns

Les résultats sont mis en forme dans un DataFrame puis sont sauves dans un fichier csv

# RESULTS

**Of the 5100 decisions studied, only 4619 specified the sex of the president, i.e. a loss of data of 9.43%.**

**Then we use the same functions as we did to clean and format the results**

	date	gender	jurisdiction
1213	25 octobre 1985	F	Assemblée plénière
1364	28 février 1986	F	Chambre mixte
2759	18 janvier 2000	F	Chambre civile 1
2975	15 mai 2002	F	Chambre civile 3
3127	12 octobre 2004	F	Chambre sociale
...	...	...	...
5088	18 janvier 2023	F	Chambre civile 3
5092	11 janvier 2023	F	Chambre civile 3
5094	16 mars 2023	F	Chambre civile 3
5097	18 janvier 2023	F	Chambre sociale
5098	25 janvier 2023	F	Chambre civile 3

565 rows x 3 columns

On se rend compte qu'il y a beaucoup de données manquantes

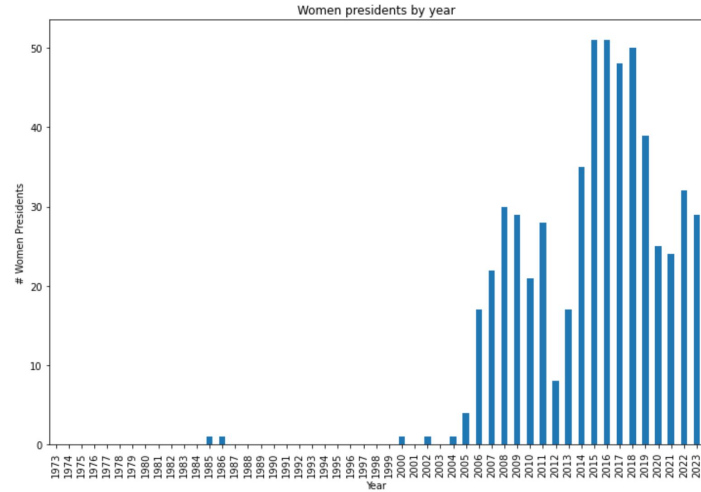
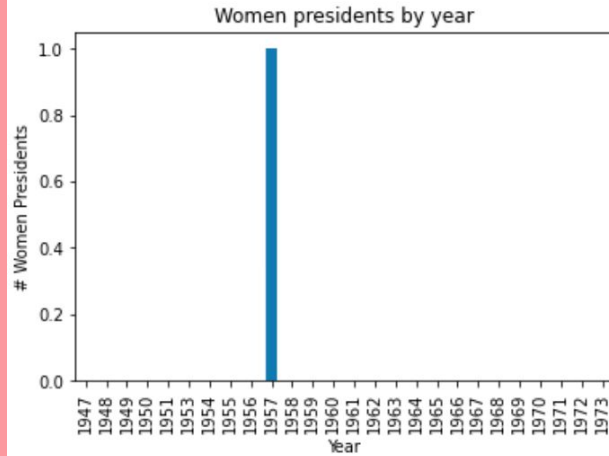
```
In [42]:
```

```
print(f"Nombre de jugements étudiés: {len(results)}\nNombre de jugements où le s  
exe du président est indiqué: {len(results[results['gender'] != ''])}")
```

Nombre de jugements étudiés: 5100

Nombre de jugements où le sexe du président est indiqué: 4619

# RESULTS





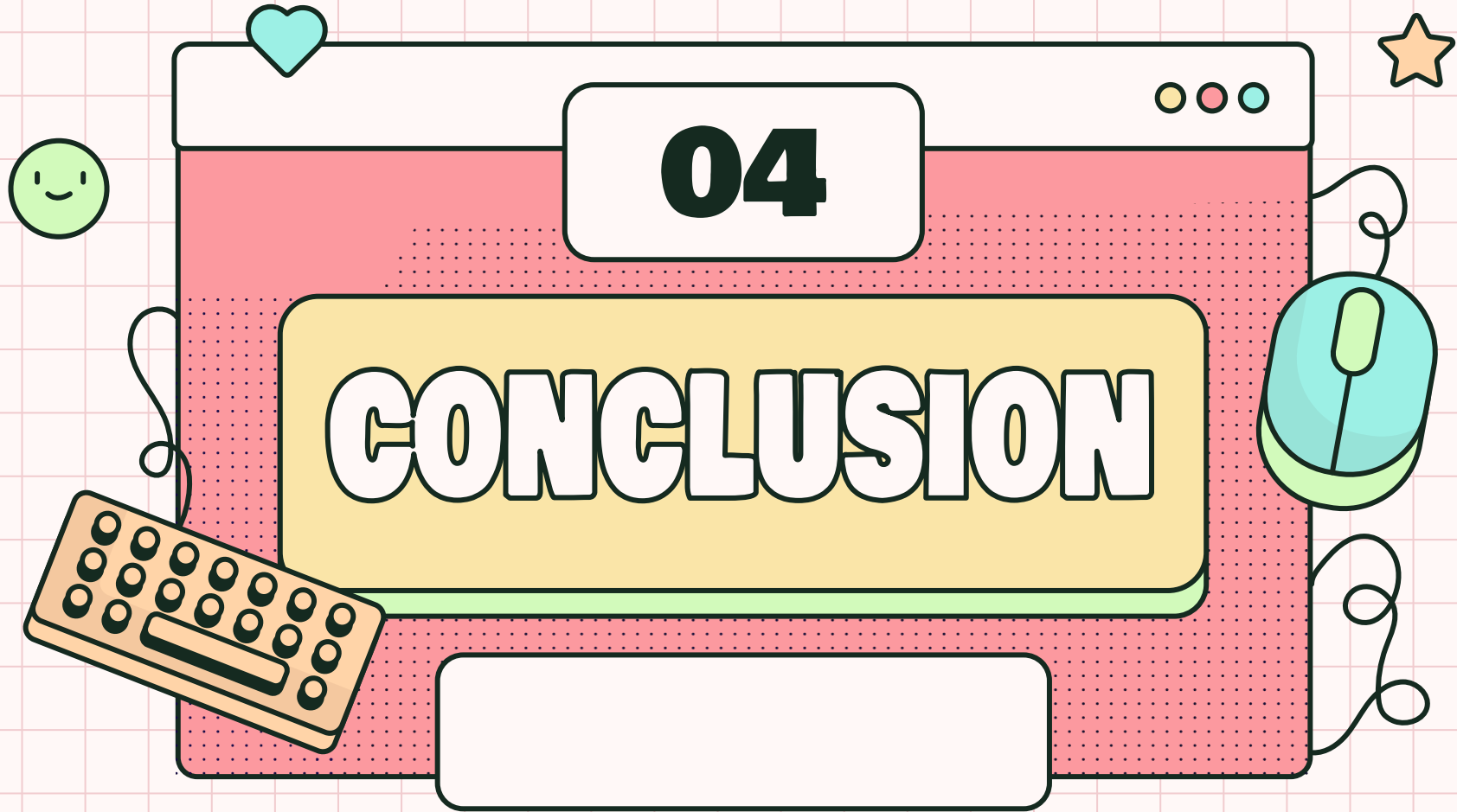
# RESULTS

**Number of  
decisions studied :  
5789**

**Number of  
decisions with  
usable data: 5190**



**566 with woman  
as a president**



# OUR RESEARCH ON THE EVOLUTION SUBJECT

**1900**

enrolment in the  
Faculty of Law,

**1920**

practice as a lawyer

**1946**

first woman advisor to  
the Court of Cassation  
in 1946

**1970-1990**

slow progression of women in  
the judiciary  
but 1984-1988 : Simone Rozès =>  
first female president of the  
Court of Cassation

**2011**

first woman president  
of a chamber in 2011

**2019**

Chantal Arens:  
seconde female  
president of the Court  
of Cassation

