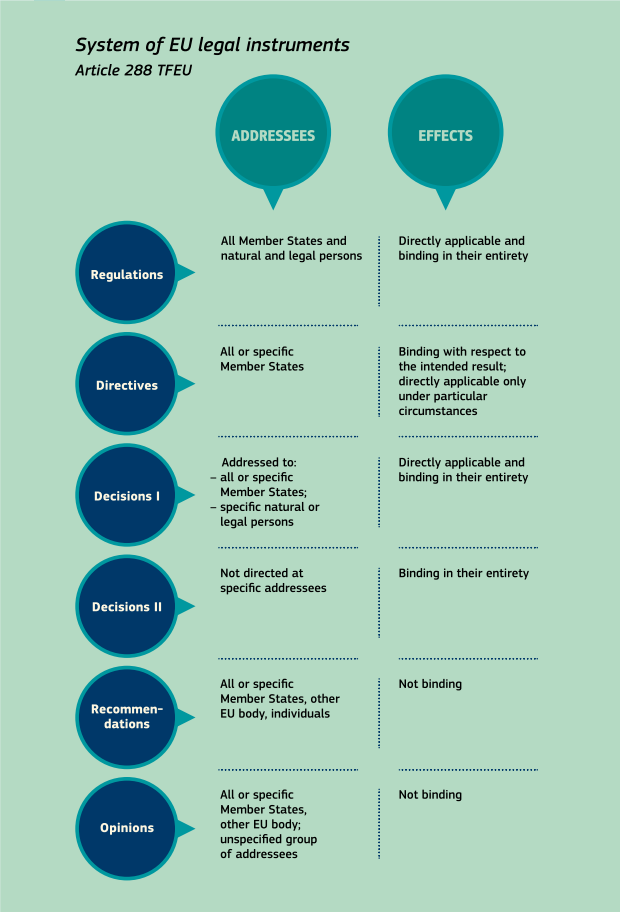
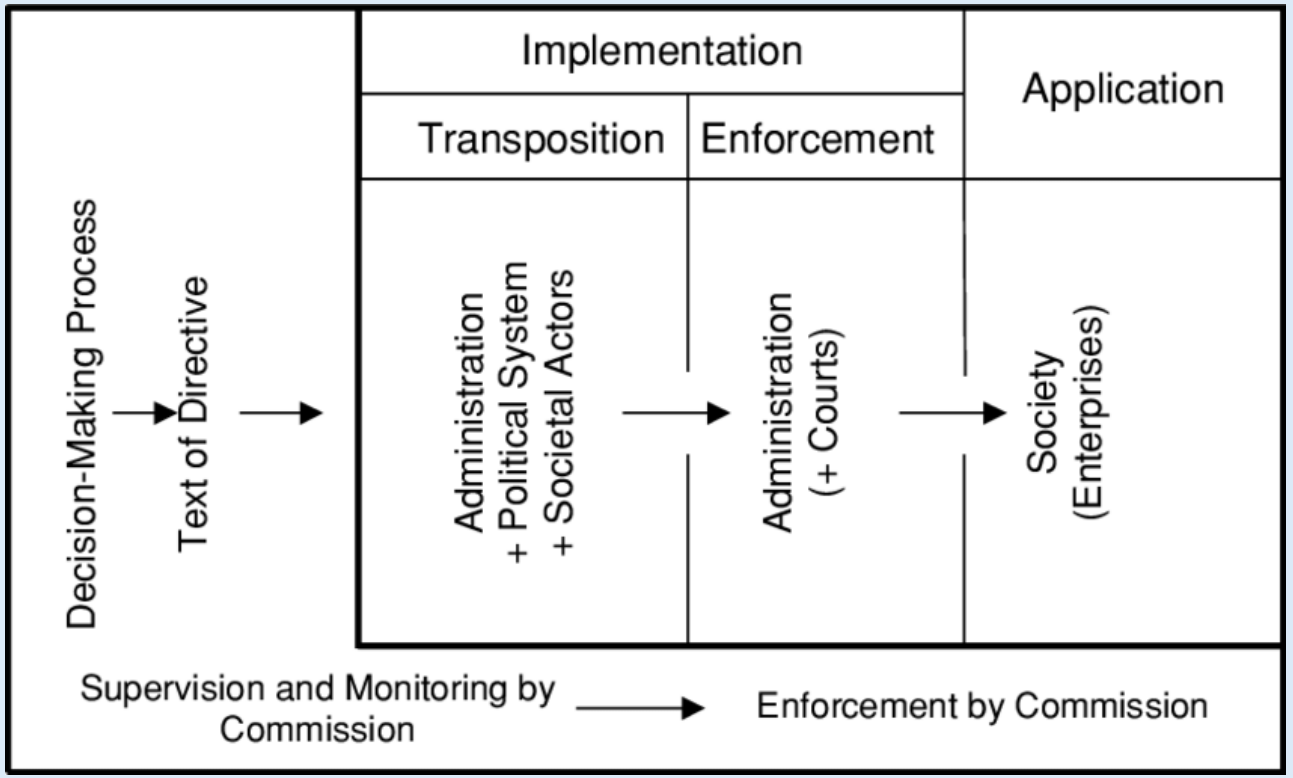
Transposing EU directives

Arthur Lacour, Louis-Barthélemy Legoux, Léonilde Lopes Tavares, Arthur Poinsard

Table of Contents

EU directives: a quick reminder





Lingering on EU Directives:

Why such a choice of topic?

- A telltale legal and political witness of the coordination and conflict between Member States and the Union

→A stumbling block & an indicator of frictions around the EU project in general

- Temporal boundaries of our study: the post-Brexit era
  - A quantitative and technical motive
  - An urgency and relevancy motive
  - An intensity and criticity motive

Overview of the general path we wended

1

Creating two different databases, gathering:

- Directives &
- Transposed legal acts

2

Matching both databases

- Use of the CELEX number to check the presence of corresponding acts in national law at all

3

Numbering the delay through the difference between the due transposition date and the effective one

4

Comparing country-per-country performances over the defined period

5

Spotting possible patterns and interpreting results

EUR-Lex

Access to European Union law

Step 1: Crafting two databases through scraping

Gathering relevant directives & transposed legal acts

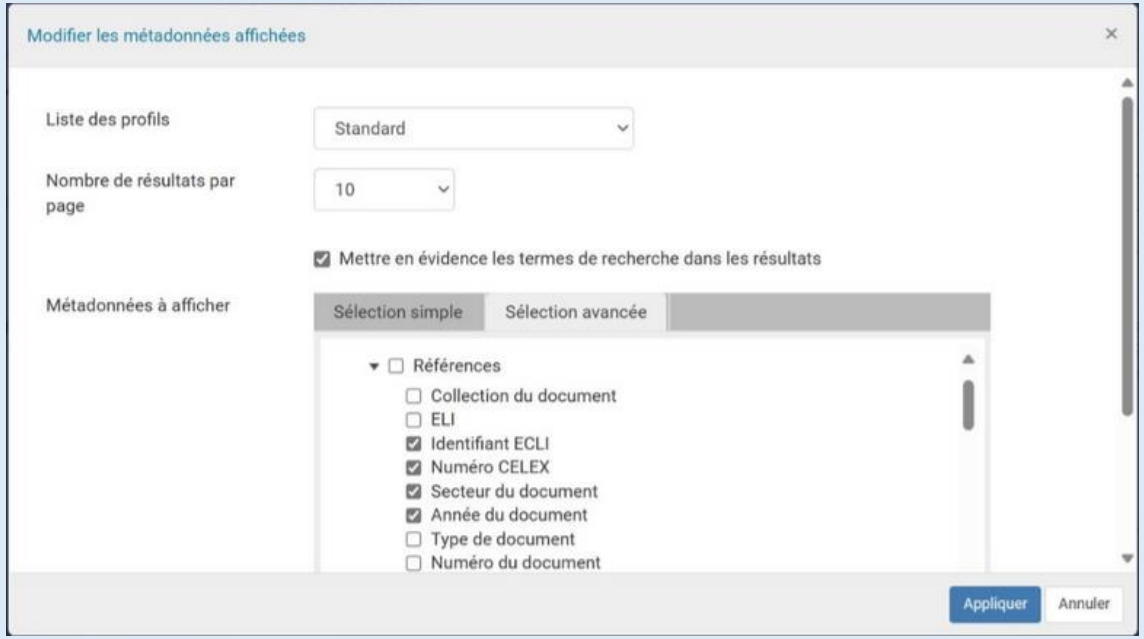
Database 1 – Directives

- CELEX number
- Issuing date
- Topic / subject

Database 2 – Transposed legal acts

- CELEX number
- Country
- Date of the act
- Nature of the date

Scraping the Eurlex database website : a pseudo static site only



Beginning the scraping process

- As a first step, **scrape the CELEX number**, i.e. the common denominator to navigate all legal acts
- Repeat the process for the date and the subject

```
In [10]: import csv
    ...: from selenium import webdriver
    ...: from selenium.webdriver.common.by import By
    ...: from selenium.webdriver.firefox.service import Service
    ...: from webdriver_manager.firefox import GeckoDriverManager
    ...: from selenium.webdriver.support.ui import WebDriverWait
    ...: from selenium.webdriver.support import expected_conditions as EC
    ...: import re
    ...: from time import sleep
    ...: from selenium.common.exceptions import NoSuchElementException

In [11]: def extract_directive_info(parent_element): #define a function to scrap for celex number, looking at two positions
    ...:     try:
    ...:         celex_number_candidate1 = parent_element.find_element(By.XPATH, ".//div[1]/dl/dd[1]").text
    ...:         if re.match(r'3\d{4}L', celex_number_candidate1):
    ...:             celex_number = celex_number_candidate1
    ...:         else:
    ...:             celex_number_candidate2 = parent_element.find_element(By.XPATH, ".//div[1]/dl/dd[2]").text
    ...:             if re.match(r'3\d{4}L', celex_number_candidate2):
    ...:                 celex_number = celex_number_candidate2
    ...:             else:
    ...:                 celex_number = "N/A"
    ...:     except NoSuchElementException: #in case there is no celex, the code does not brake
    ...:         pass
    ...:     #we do the same for the publication date and subject
    ...:     try:
    ...:         publication_date = parent_element.find_element(By.XPATH, ".//div[2]/dl/dd[2]").text.split(";")[0]
    ...:     except NoSuchElementException:
    ...:         publication_date = "Unknown"
    ...:
    ...:     subjects = ''
    ...:     try:
    ...:         subjects_elements = parent_element.find_elements(By.XPATH, ".//dl/dd[4]/ul/li/a") + parent_element.find_elements(By.XPATH, ".//dl/dd[5]/ul
    ...: /li/a")
    ...:         subjects = ', '.join([elem.text for elem in subjects_elements]) #there are sometimes more than oen subject, separated by comma
    ...:     except NoSuchElementException:
    ...:         pass
    ...:
    ...:     return celex_number, publication_date, subjects
    ...:
```

CELEX number emplacement patterns

The CELEX number being not always at the same place in the webpage, we need to check the form of the gathered element

Implementing the scraping process

```
In [12]: service = Service(executable_path=GeckoDriverManager().install())
    ...: driver = webdriver.Firefox(service=service)
    ...:
    ...: # URL for directives, the code will append the page number
    ...: directive_base_url = "https://eur-lex.europa.eu/search.html?SUBDOM_INIT=ALL_ALL&DB_TYPE_OF_ACT=directive&DTS_SUBDOM=ALL_ALL&typeOfActStatus=DIRECT
    ...: IVE%2COTHER&or0=DN%3D3*%2CDN-old%3D3*&type=advanced&date0=ALL%3A01022020%7C.&qid=1711414697566&DTC=false&orFM_CODEDGroup=FM_CODED%3DDIR&DTS_DOM=AL
    ...: L&FM_CODED=&lang=fr&excConsLeg=true&page="
    ...:
    ...: #URL for transposed acts, the code will append the page number
    ...: transposed_act_base_url = "https://eur-lex.europa.eu/search.html?SUBDOM_INIT=MNE&DTS_SUBDOM=MNE&sortOneOrder=desc&sortOne=IDENTIFIER_SORT&lang=en&
    ...: type=advanced&qid=1711241216437&page="
    ...:
    ...: # Scraping
    ...: with open('directives.csv', 'w', newline='', encoding='utf-8') as directives_file, open('transposed_act.csv', 'w', newline='', encoding='utf-8') a
    ...: s transposed_file:
    ...:     directives_writer = csv.writer(directives_file)
    ...:     transposed_writer = csv.writer(transposed_file)
    ...:     directives_writer.writerow(['CELEX', 'Date', 'Subjects'])
    ...:     transposed_writer.writerow(['Transposed act', 'Country', 'Date', 'Date nature'])
    ...:
```

- Inputting the two relevant URLs
- Preparing to loop over all relevant databases pages

Around 1424 directives and more than 180 000 transposed acts to scrape… which took much time!!

Scraping directives

```
In [14]: # Scraping directives
    ...:     page_number = 1
    ...:     while True:
    ...:         current_page_url = f"{directive_base_url}{page_number}"
    ...:         driver.get(current_page_url)
    ...:         sleep(5)
    ...:         if page_number == 1:
    ...:             WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.ID, "link-change-metadata_top"))).click() #click to personalize results
    ...:
    ...:             sleep(2) #giving some time to load information
    ...:             WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.ID, "simple_c31classificationsCT"))).click() #click on the checkbox for
    ...:  subject
    ...:             sleep(2)
    ...:             WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, "//*[@id='nbResultPerPage']"))).click()  #©change the number of
    ...: displayed result for 20 by page
    ...:             sleep(2)
    ...:             WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, "/html/body/div[3]/div/div/div[2]/div/div/form[2]/fieldset/div[1
    ...: ]/div[1]/div/div[2]/select/option[3]"))).click()
    ...:             sleep(1)
    ...:             WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.ID, "button.apply"))).click() #click to apply changes
    ...:             sleep(5)
    ...:             driver.get(current_page_url)
    ...:             sleep(5)
    ...:
    ...:         parent_elements = WebDriverWait(driver, 10).until(EC.presence_of_all_elements_located((By.XPATH, "//*[starts-with(@id, 'MoreSR_')]")))
    ...:         if not parent_elements:
    ...:             print(f"End of pages or no data found at page {page_number} for directives.")
    ...:             break
    ...:         for parent in parent_elements:
    ...:             celex_number, publication_date, subjects = extract_directive_info(parent)
    ...:             directives_writer.writerow([celex_number, publication_date, subjects]) #allows to write in a new row of the csv, each element is in a
    ...: column
    ...:             print(f"Directive CELEX: {celex_number}, Date: {publication_date}, Subjects: {subjects}")
    ...:         page_number += 1
```

The loop holds as long as the URL is relevant

We allow the algorithm to go without bugging by giving it time before implementing the next line of code: Eurlex database is heavy to handle…

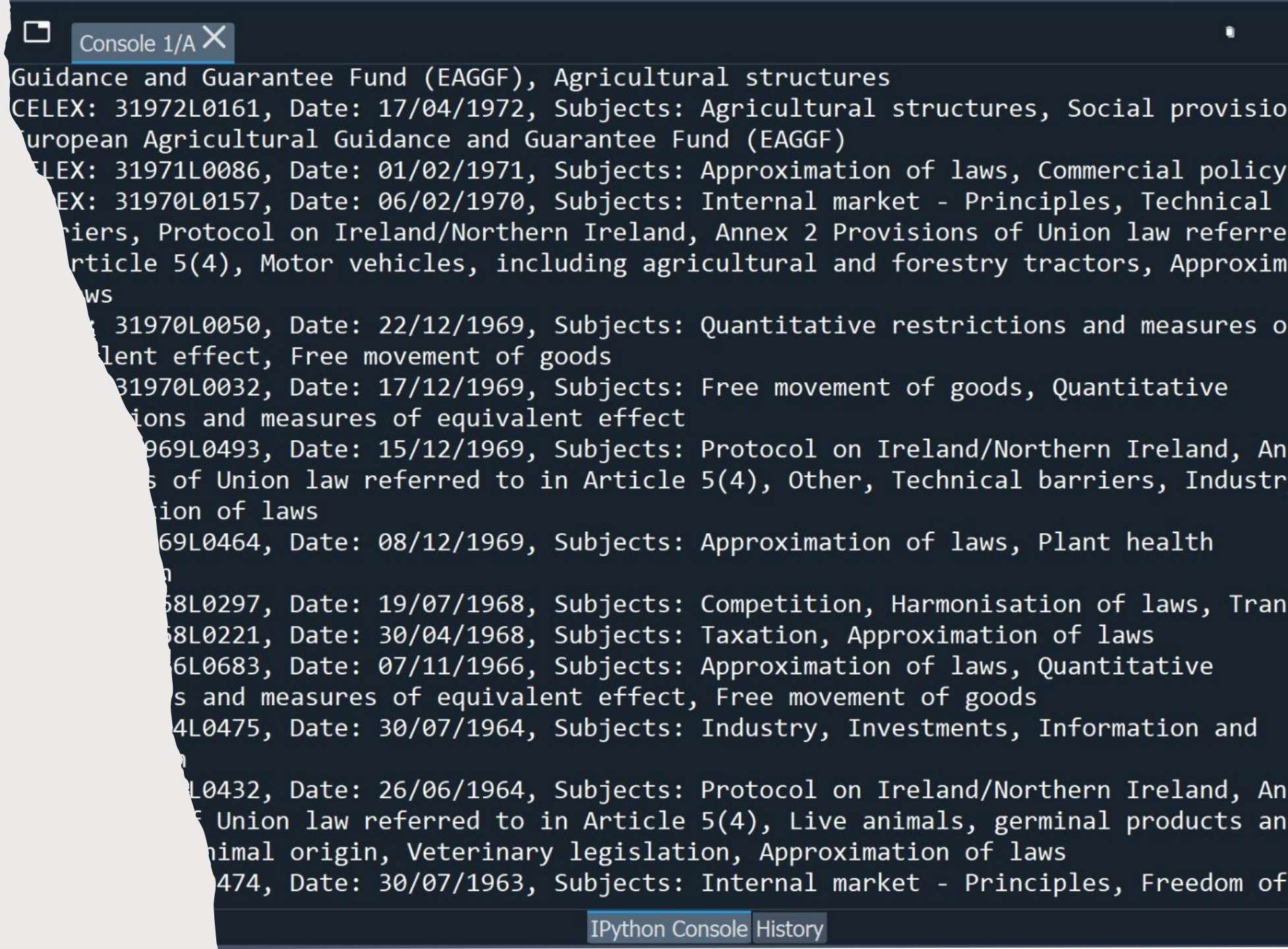
We automatise the successive clicking operations to unfold the data we seek

Extract the trio of elements we need on each page scraped, raked through by the algorithm

Repeat the operation on all pages

Console results:

Database 1 (Directives)



Scraping transposed legal acts to create database 2

```
In [17]:  # Reset page number for transposed acts
    ...:     page_number = 1
    ...:     while True:
    ...:         current_page_url = f"{transposed_act_base_url}{page_number}"
    ...:         driver.get(current_page_url)
    ...:         sleep(2)
    ...:         if page_number == 1:
    ...:                 WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.ID, "link-change-metadata_top"))).click() #click to personalize res
    ...: ults
    ...:                 sleep(2) #giving some time to load information
    ...:                 WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, "//*[@id='nbResultPerPage']"))).click()  #change the number
    ...:  of displayed result for 20 by page
    ...:                 sleep(2)
    ...:                 WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, "/html/body/div[3]/div/div/div[2]/div/div/form[2]/fieldset/d
    ...: iv[1]/div[1]/div/div[2]/select/option[3]"))).click()
    ...:                 sleep(1)
    ...:                 WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.ID, "button.apply"))).click() #click to apply changes
    ...:                 sleep(5)
    ...:                 driver.get(current_page_url)
    ...:                 sleep(5)
    ...:         parent_elements = driver.find_elements(By.XPATH, "//*[starts-with(@id, 'MoreSR')]")
    ...:         if not parent_elements:
    ...:             print(f"End of pages or no data found at page {page_number} for transposed acts.")
    ...:             break
    ...:
    ...:         for parent in parent_elements:
    ...:             transposed_acts_elements = parent.find_elements(By.XPATH, ".//dl/dd[3]//a")
    ...:             country = parent.find_element(By.XPATH, ".//div[2]/dl/dd[1]").text.strip()
    ...:             date_text = parent.find_element(By.XPATH, ".//div[2]/dl/dd[2]").text.strip()
    ...:             date, date_nature = (date_text.split("; ") + [""])[:2] #scraping output gives date;nature of the date, so we split by the ; and get a
    ...: list of two elements, we put additional "" in case the split gives only one element and [:2] to get only two elements in case there are more than
    ...: one ; (or zero)
    ...:
    ...:             if transposed_acts_elements:
    ...:                 for act_element in transposed_acts_elements:
    ...:                     transposed_act = act_element.text.strip()
    ...:                     transposed_writer.writerow([transposed_act, country, date, date_nature])
    ...:                     print(f"Transposed act: {transposed_act}, Country: {country}, Date: {date}, Date nature: {date_nature}")
    ...:             else:
    ...:                 transposed_writer.writerow(['N/A', country, date, date_nature])
    ...:                 print(f"Transposed act: N/A, Country: {country}, Date: {date}, Date nature: {date_nature}")
    ...:
    ...:         page_number += 1
    ...:
    ...: driver.quit()
```

We make sure we start at page one

In an analogous manner to the scraping operation for directives, we automatise the different clicking operations to extract all 4 elements we search

We separate the date and its nature

The variable here gives the CELEX number of each transposed act

We build the second database

Console results:

Database 2 (transposed legal acts)

```
Transposed act: 31964L0054, Country: Belgium, Date: 31/12/1968, Date nature: Entry into force
Transposed act: 31964L0054, Country: Belgium, Date: 15/02/1969, Date nature: Date of publication
Transposed act: 31964L0054, Country: Belgium, Date: 08/08/1968, Date nature: Entry into force
Transposed act: 31964L0054, Country: Belgium, Date: 28/05/1968, Date nature: Entry into force
Transposed act: 31964L0054, Country: Belgium, Date: 28/08/1968, Date nature: Entry into force
ransposed act: 31964L0054, Country: Belgium, Date: 16/02/1968, Date nature: Date of publication
nsposed act: 31964L0054, Country: Belgium, Date: 29/02/1968, Date nature: Date of publication
sposed act: 31964L0054, Country: Belgium, Date: 26/04/2004, Date nature: Date of notification
sposed act: 31964L0054, Country: Austria, Date: 30/06/1994, Date nature: Date of publication
posed act: 31964L0054, Country: Austria, Date: 30/06/1994, Date nature: Date of publication
posed act: 31963L0607, Country: Portugal, Date: 07/10/1993, Date nature: Date of publication
osed act: 31963L0607, Country: United Kingdom, Date: 05/01/1998, Date nature: Date of publication
sed act: 31963L0607, Country: Finland, Date: 01/01/1001, Date nature:
ed act: 31963L0607, Country: Spain, Date: 27/06/1986, Date nature: Date of publication
ed act: 31963L0607, Country: Austria, Date: 01/01/1001, Date nature:
d act: 31963L0474, Country: Portugal, Date: 20/10/1986, Date nature: Entry into force
act: 31963L0474, Country: Portugal, Date: 03/10/1985, Date nature: Entry into force
ct: 31963L0474, Country: Poland, Date: 23/07/2004, Date nature: Date of notification
ct: 31963L0474, Country: Greece, Date: 19/06/1987, Date nature: Date of publication
ct: 31963L0474, Country: France, Date: 01/09/1963, Date nature: Date of publication
t: 31963L0474, Country: Spain, Date: 01/01/1001, Date nature:
t: 31963L0474, Country: Spain, Date: 03/07/1991, Date nature: Date of publication
t: 31963L0474, Country: Spain, Date: 11/07/1990, Date nature: Date of publication
: 31963L0474, Country: Spain, Date: 03/07/1990, Date nature: Date of publication
: 31963L0474, Country: Spain, Date: 13/06/1989, Date nature: Date of publication
: 31963L0474, Country: Spain, Date: 01/01/1001, Date nature:
: 31963L0474, Country: Spain, Date: 03/04/1987, Date nature: Date of publication
: 31963L0474, Country: Spain, Date: 03/04/1987, Date nature: Date of publication
: 31963L0474, Country: Spain, Date: 01/01/1001, Date nature:
: 31963L0474, Country: Spain, Date: 27/09/1979, Date nature: Date of publication
31963L0474, Country: Czechia, Date: 10/02/2003, Date nature: Date of publication
31963L0474, Country: Czechia, Date: 13/12/2002, Date nature: Date of publication
31963L0474, Country: Czechia, Date: 09/05/2002, Date nature: Date of publication
32005L0060, Country: Czechia, Date: 31/12/2001, Date nature: Date of publication
1963L0474, Country: Czechia, Date: 31/12/2001, Date nature: Date of publication
2005L0060, Country: Czechia, Date: 29/09/1995, Date nature: Date of publication
963L0474, Country: Czechia, Date: 29/09/1995, Date nature: Date of publication
```

EUR-Lex

Access to European Union law

Step 2: Matching both databases in search of transpositions

```
In [20]: import pandas as pd
    ...: from datetime import datetime
    ...: import matplotlib.pyplot as plt
    ...: import matplotlib.colors as mcolors
    ...: import numpy as np
    ...:
    ...: # Format dates
    ...: def clean_and_parse_date(date_str):
    ...:     date_str = date_str.split(';')[0]
    ...:     return datetime.strptime(date_str, "%d/%m/%Y")
    ...:
    ...: # Load the data
    ...: directives_df = pd.read_csv('directives.csv')
    ...: transposed_act_df = pd.read_csv('transposed_act.csv')
    ...:
    ...: # Applying the function of format the date
    ...: directives_df['Date'] = directives_df['Date'].apply(clean_and_parse_date)
    ...: transposed_act_df['Date'] = transposed_act_df['Date'].apply(clean_and_parse_date)
    ...:
    ...: # Group by celex and select the oldest occurence for calculation
    ...: transposed_min_dates = transposed_act_df.groupby(['Transposed act', 'Country'])['Date'].min().reset_index()
    ...:
    ...: # Scoring
    ...: results = []
    ...: for _, row in transposed_min_dates.iterrows():
    ...:     transposed_celex = row['Transposed act']
    ...:     country = row['Country']
    ...:     transposed_date = row['Date']
    ...:
    ...:     directive_row = directives_df[directives_df['CELEX'] == transposed_celex]
    ...:
    ...:     if not directive_row.empty:
    ...:         directive_date = directive_row.iloc[0]['Date']
    ...:         subject = directive_row.iloc[0]['Subjects']  # Capture the subject from the directive row
    ...:         difference = (transposed_date - directive_date).days
    ...:         corrected_score = max(difference, 0)  # Corrected score: 0 if negativen otherwise keep the result
    ...:
    ...:         results.append({
    ...:             'Transposed CELEX': transposed_celex,
    ...:             'Country': country,
    ...:             'Subject': subject,  # Include the subject in the results
    ...:             'Date Difference (Days)': difference,
    ...:             'Corrected Score': corrected_score
    ...:         })
    ...:
    ...: # Create the dataframe for further analysis
    ...: results_df = pd.DataFrame(results)
    ...:
    ...: print(results_df)
```

Approximation

We compute the difference, in days, between the date of the first transposed act and the theoretical transposition date

Conducting the analysis

- We have recourse to the *pandas* module
- We first have to select one transposed legal act to compare its date to that of the theoretical transposition date of the directive
- Then only can we compare the transposition performance through a numbering tool

EUR-Lex

Access to European Union law

Step 3: Numbering the transposition performance

```
...: # Save in csv
...: results_df.to_csv('date_differences_with_subjects.csv', index=False)
...:
...: # Calculate score for each country
...: country_scores = results_df.groupby('Country')['Corrected Score'].sum().reset_index()
...:
...: # Rank countries
...: country_rankings = country_scores.sort_values(by='Corrected Score', ascending=True).reset_index(drop=True)
...: print("Delay of transposition by countries (days)")
...: print(country_rankings)
...:
...: # Normalize score for color purpose in the graph
...: scores = country_rankings['Corrected Score']
...: normalized_scores = (scores - scores.min()) / (scores.max() - scores.min())
...:
...: #color gradient by score
...: colors = [mcolors.to_rgba(c) for c in plt.cm.RdYlGn_r(np.linspace(0, 1, len(scores)))]
...: bar_colors = [colors[int(np.round(score * (len(colors) - 1)))] for score in normalized_scores]
...:
...: plt.figure(figsize=(10, 6))
...: bars = plt.bar(country_rankings['Country'], country_rankings['Corrected Score'], color=bar_colors)
...:
...: plt.xlabel('Country')
...: plt.ylabel('Corrected Score')
...: plt.title('Transposition delay (day)')
...:
...: plt.xticks(rotation=45)
...: plt.tight_layout()
...: plt.show()
...: # Group by countries and count transposed act
...: unique_transposed_counts = transposed_act_df.groupby('Country')['Transposed act'].nunique().reset_index()
...:
...: unique_transposed_counts['Number of Transposed Acts'] = unique_transposed_counts['Transposed act']
...:
...: unique_transposed_counts.drop(columns=['Transposed act'], inplace=True)
...:
...:
...: # Sort countries by acts transposed
...: unique_transposed_counts_sorted = unique_transposed_counts.sort_values(by='Number of Transposed Acts', ascending=False).reset_index(drop=True)
...: print("\nNumber of transposed acts")
...: print(unique_transposed_counts_sorted)
...:
...: # Show subject by celex
...: celex_subjects = directives_df[['CELEX', 'Subjects']]
...: # Separate subject to have on a single row
...: subjects_expanded = directives_df.set_index('CELEX')['Subjects'].str.split(', ', expand=True).stack().reset_index(level=1, drop=True).reset_index(name='Subject')
...:
```

In order to compare the performance of member states, we create an aggregate indicator in summing the individual score obtained for each directive

In order to display visually rank the countries, we create a gradient of colours

```
...: # Merge dataframe to get subject with country
...: merged_df = pd.merge(subjects_expanded, transposed_act_df, left_on='CELEX', right_on='Transposed act', how='inner')
...:
...: # Get the number of celex by subject
...: subject_country_counts = merged_df.groupby(['Subject', 'Country']).size().reset_index(name='Count')
...:
...: print(subject_country_counts.head())
...:
...:
...: # Today
...: today = pd.to_datetime('today')
...:
...: # Get all celex
...: unique_celex = directives_df['CELEX'].unique()
...:
...: # Get all countries
...: unique_countries = transposed_act_df['Country'].unique()
...:
...: # List to collect results
...: results = []
...:
...: # Non transposed celex
...: for country in unique_countries:
...:     transposed_celex_for_country = transposed_act_df[transposed_act_df['Country'] == country]['Transposed act'].unique()
...:     non_transposed_celex = [celex for celex in unique_celex if celex not in transposed_celex_for_country]
...:
...:     # Calculate delay compared to today's date
...:     for celex in non_transposed_celex:
...:         directive_info = directives_df[directives_df['CELEX'] == celex]
...:         if not directive_info.empty:
...:             publication_date = pd.to_datetime(directive_info.iloc[0]['Date'])
...:             days_difference = (today - publication_date).days
...:             results.append({
...:                 'Country': country,
...:                 'CELEX': celex,
...:                 'Days Since Publication': days_difference
...:             })
...:
...: results_df = pd.DataFrame(results)
...: print(results_df)
...:
...: results_df.to_csv('non_transposed_celex_by_country.csv', index=False)
...:
...: # Split all subject into a dataframe
...: expanded_subjects_df = directives_df.drop('Subjects', axis=1).join(directives_df['Subjects'].str.split(', ', expand=True).stack().reset_index(level=1, drop=True).rename('Subject'))
...:
```

We need to address the directives which have not transposed at all yet

That is why we compute the delay in comparison with today

```
...: # count by country number of celex by subject
...: subject_country_transposed_counts = transposed_act_df.merge(expanded_subjects_df, left_on='Transposed act', right_on='CELEX').groupby(['Country', 'Subject']).size().reset_index(name='Unique Transpos
...: ed Acts Count')
...:
...: print("Number of acts transposed:")
...: print(subject_country_transposed_counts)
...:
...: # Visualisation for a given country
...: selected_country = 'France'
...: selected_data = subject_country_transposed_counts[subject_country_transposed_counts['Country'] == selected_country].sort_values(by='Unique Transposed Acts Count', ascending=False)
...:
...: # counting
...: total_transposed_by_subject = subject_country_transposed_counts.groupby('Subject')['Unique Transposed Acts Count'].sum().reset_index()
...:
...: # sorting
...: top_subjects = total_transposed_by_subject.sort_values(by='Unique Transposed Acts Count', ascending=False).head(10)
...:
...: # first 10 subjects
...: plt.figure(figsize=(12, 8))
...: plt.bar(top_subjects['Subject'], top_subjects['Unique Transposed Acts Count'], color='skyblue')
...: plt.title('Top 10 Subjects by Number of Unique Transposed Acts for France')
...: plt.xlabel('Subject')
...: plt.ylabel('Number of Unique Transposed Acts')
...: plt.xticks(rotation=45, ha='right')
...: plt.tight_layout()
...: plt.show()
...:
...: # counting
...: subject_country_transposed_counts = transposed_act_df.merge(
...:     expanded_subjects_df, left_on='Transposed act', right_on='CELEX').groupby(['Subject']).size().reset_index(name='Unique Transposed Acts Count')
...:
...: # sorting
...: sorted_subjects = subject_country_transposed_counts.sort_values(by='Unique Transposed Acts Count', ascending=False)
...:
...: # plotting
...: plt.figure(figsize=(12, 8))
...: plt.bar(unique_transposed_counts_sorted['Country'], unique_transposed_counts_sorted['Number of Transposed Acts'], color='teal')
...: plt.title('Number of transposed act by country')
...: plt.xlabel('Country')
...: plt.ylabel('Number of acts')
...: plt.xticks(rotation=90)  # Rotation des étiquettes de l'axe des x pour une meilleure lisibilité
...: plt.tight_layout()
...: plt.show()
...:
...: # selection of the top ten subjects
...: top_10_subjects = sorted_subjects.head(10)
...:
...: print("Acts transposed by subject :")
...: print(sorted_subjects)
...:
...: total_transposed_by_country = transposed_act_df.groupby('Country').size().reset_index(name='Total Transposed Acts')
...:
...: sorted_countries_by_transposed_acts = total_transposed_by_country.sort_values(by='Total Transposed Acts', ascending=False)
...:
```

We wanted to understand the breakdown of directives sectors, and display the most recurrent ones, chosing France as a witness State

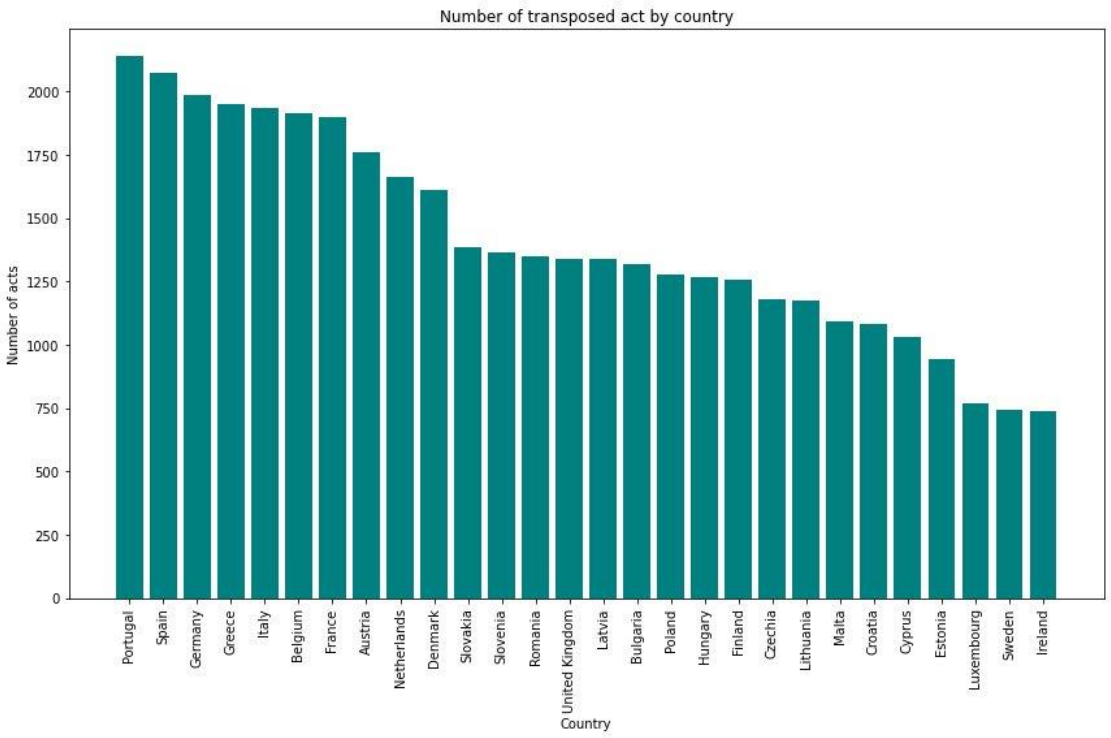
Finally, we plot the different graphs

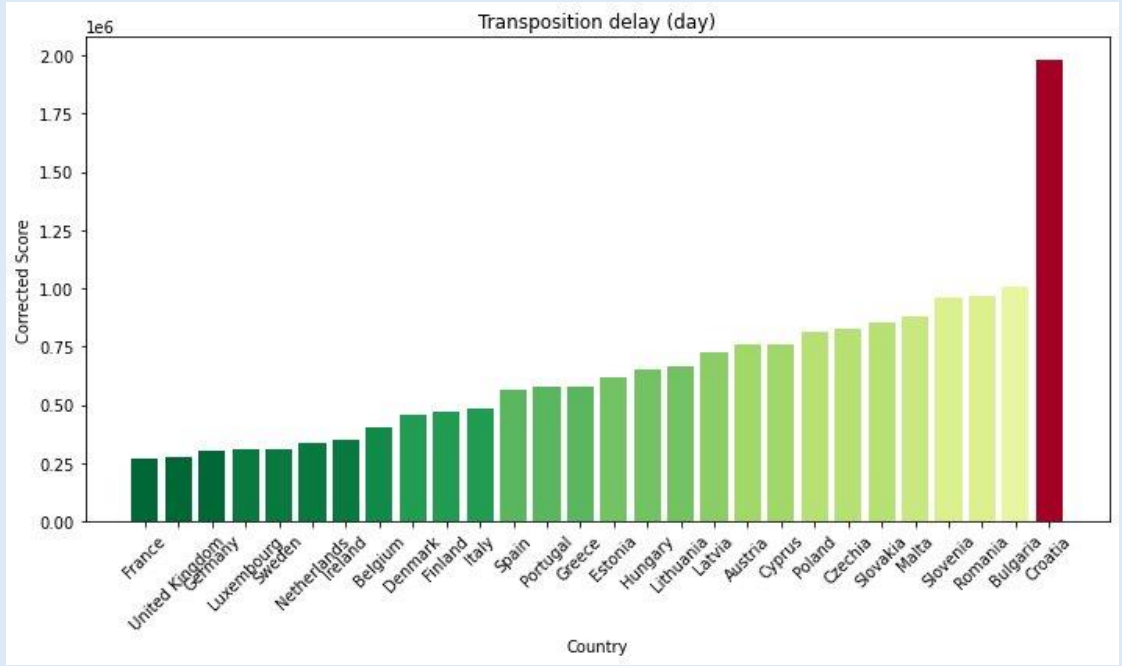
EUR-Lex

Access to European Union law

Step 4: Comparing country-per-country performance

Country comparison



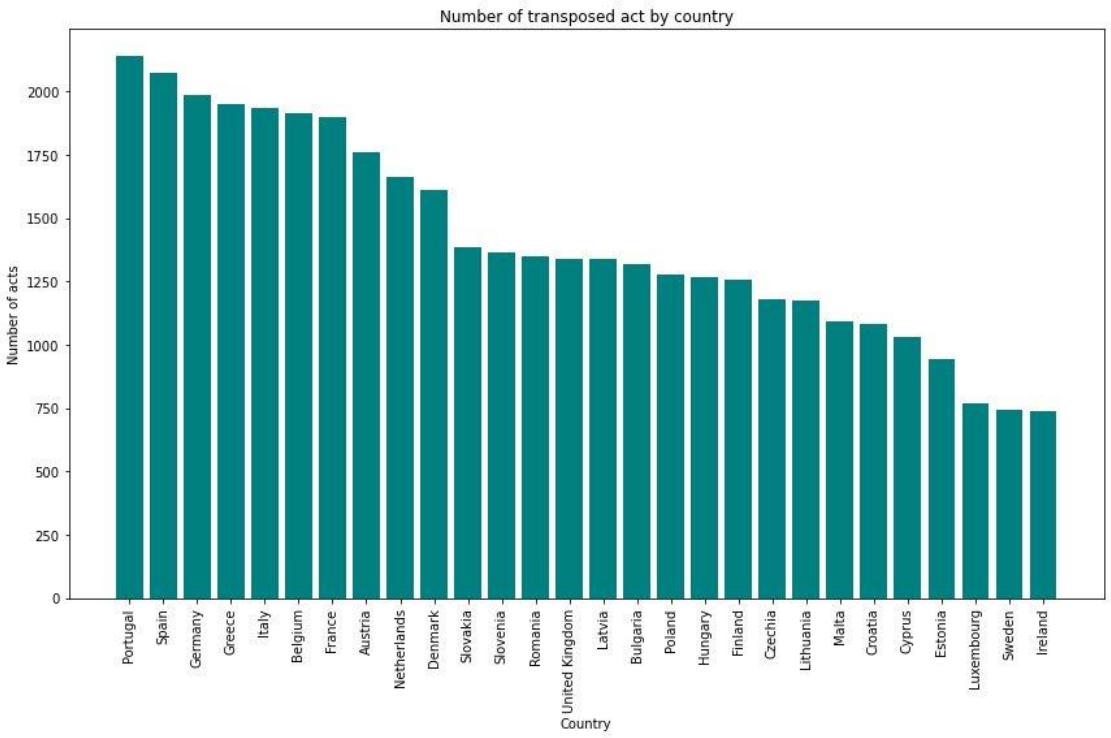


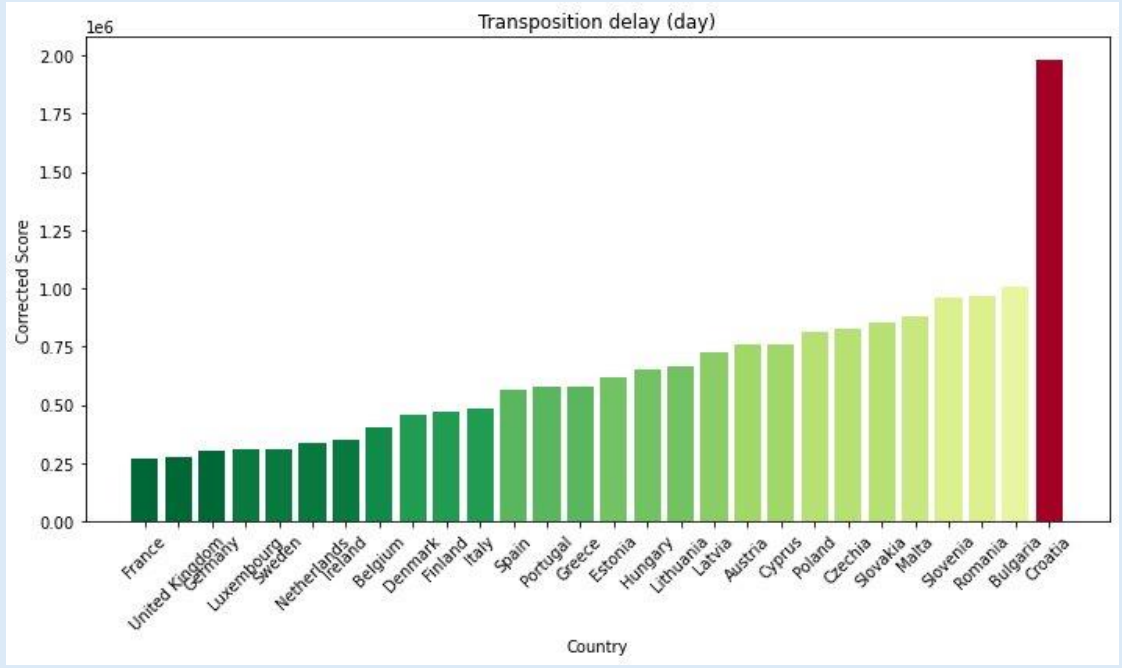
EUR-Lex

Access to European Union law

Step 5: Interpreting the results through pattern spotting

Comparison interpretation





Difficulties encountered

- **Technical difficulties**

⇒ The type of website (semi-static only, dynamic mainly)

⇒ Directives entanglement with rectifications and addenda

⇒ Data entanglement (such as the date and its nature)

⇒ Heaviness of the data at hand causing dramatic implementation delays

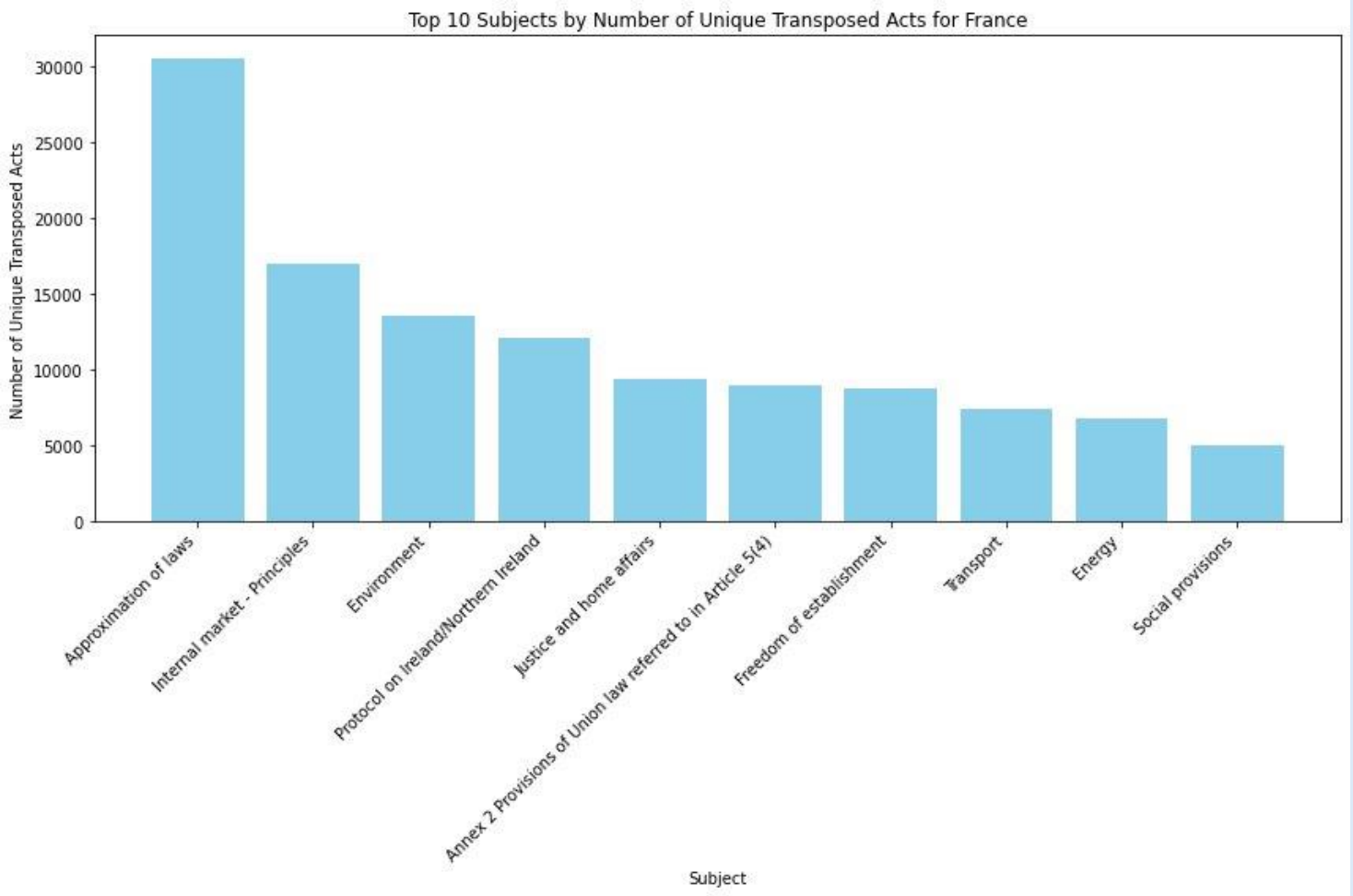
- **A lack of data uniformity**

⇒ Transposition idiosyncrasies

⇒ Unequal time of transposition linked to each directive

⇒ An unequal classification of subjects of directives

Directive sectors



Results limitations & further refinements

Ad maximum

- Data **aggregation** and entanglement in the main indicator we used

Ad minimum

- Data **approximation** which hid possible delays in transposing as well as the quality of transposition

Possible improvements and extensions of the project

- Bundling the sectors of directives in intermediary subsectors for clarity
- Considering all forms of transposed acts
- Pondering over the type of transposition to delve into a finer qualitative analysis
- Extending the temporal scope and ramifying the comparison: