

Main: Analysis of Executive Orders

Aniket Kamat Yuyang Wu Brycen Manners
Soohyun Kim

Wednesday 17th December, 2025



```
title: "Main: Analysis of Executive Orders"
author: "Group 05"
bibliography: references.bib
  hide_input: true
exports:
  - format: pdf
```

0.1 Intro

The executive order data used in this analysis were obtained from the U.S. Government Open Data Portal [[U.S. Government, 2025](#)]. For the language model experiment, we rely on the Hugging Face Transformers library to fine-tune a pretrained causal language model [[Wolf and others, 2020](#)].

Start with necessary imports

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

EO = pd.read_csv("executive_orders.csv")
print('Shape of dataframe is:', EO.shape)
print('Columns :')
list(EO.columns)
```

```
Shape of dataframe is: (1000, 13)
Columns :
```

```
['citation',
 'document_number',
 'end_page',
 'html_url',
 'pdf_url',
```

```

'type',
'subtype',
'publication_date',
'signing_date',
'start_page',
'title',
'disposition_notes',
'executive_order_number']

```

We will now perform necessary preprocessing of the data. The code for this can be found in `processing.py`, a script imported above. We will start by verifying all the rows by checking their type and subtype. We will then parse a few of the columns to make plotting and analysis easier later on.

```

import EO_processing.verify as verify
import EO_processing.additions as add

count_type_incorrect = verify.verify_type(EO, 'Presidential Document')
if count_type_incorrect > 0:
    EO_type_verified = verify.fix_type(EO, 'Presidential Document')
    print('Dataframe modified to ensure type consistency.')
else:
    EO_type_verified = EO
    print('Types are consistent.')

count_subtype_incorrect = verify.verify_subtype(EO_type_verified, 'Executive Order')
if count_subtype_incorrect > 0:
    EO_subtype_verified = verify.fix_subtype(EO_type_verified, 'Executive Order')
    print('Dataframe modified to ensure subtype consistency.')
else:
    EO_subtype_verified = EO_type_verified
    print('Subtypes are consistent.')

EO_with_years = add.add_years(EO_subtype_verified, 'signing_date')
EO_with_months = add.add_months(EO_with_years, 'signing_date')

Types are consistent.
Subtypes are consistent.

```

1 Frequency of Executive Orders

```

year_data = []
month_data = []
for _, row in EO_with_months.iterrows():
    year_data.append(int(row['year']))
    month_data.append(int(row['month']))
month_map = {1:"January", 2:"February", 3:"March", 4:"April", 5:"May", 6:"June", 7:"July", 8:"August", 9:"September", 10:"October", 11:"November", 12:"December"}

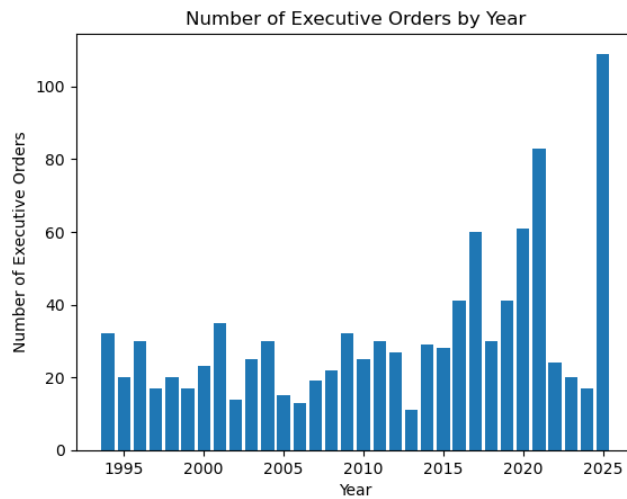
```

```

month_data_name = [month_map[i] for i in month_data]
years = np.array(year_data)
months = np.array(month_data)
months_names = np.array(month_data_name)

vals, years_plot1 = np.unique(years, return_counts=True)
plt.bar(vals, years_plot1)
plt.xlabel("Year")
plt.ylabel("Number of Executive Orders")
plt.title("Number of Executive Orders by Year")
plt.show();

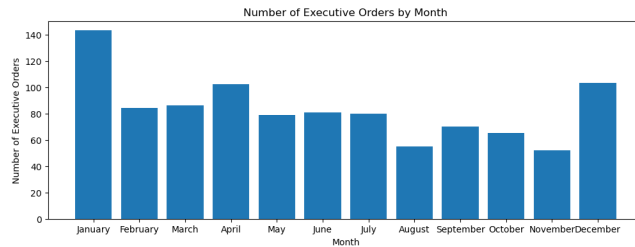
```



```

month_order = {"January": 1, "February": 2, "March": 3, "April": 4, "May": 5, "June": 6, "July": 7, "August": 8, "September": 9, "October": 10, "November": 11, "December": 12}
vals2, months_plot2 = np.unique(months_names, return_counts=True)
indices = np.argsort([month_order[m] for m in vals2])
vals2 = vals2[indices]
months_plot2 = months_plot2[indices]
plt.figure(figsize=(12, 4))
plt.bar(vals2, months_plot2)
plt.xlabel("Month")
plt.ylabel("Number of Executive Orders")
plt.title("Number of Executive Orders by Month")
plt.show();

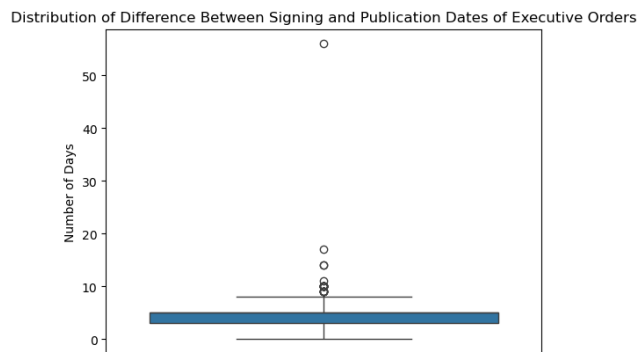
```



2 Changes in Publication Delay

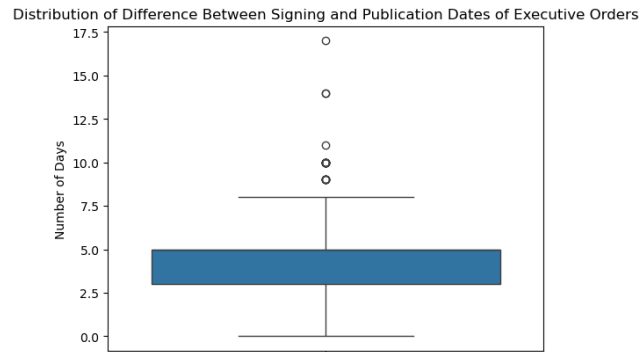
```
publication_EO = EO_with_months
publication_EO['pub_date_cleaned'] = pd.to_datetime(publication_EO["publication_date"], format='%m/%d/%Y')
publication_EO['signed_date_cleaned'] = pd.to_datetime(publication_EO["signing_date"], format='%m/%d/%Y')
publication_EO["days_diff"] = (publication_EO["pub_date_cleaned"] - publication_EO["signed_date_cleaned"]).dt.days

sns.boxplot(y=publication_EO['days_diff'])
plt.ylabel("Number of Days")
plt.title("Distribution of Difference Between Signing and Publication Dates of Executive Orders")
plt.show()
```



There is obviously a major outlier, so we will first replot this and exclude that EO. Then, we will examine that entry.

```
sns.boxplot(y=publication_EO.loc[publication_EO['days_diff'] < 50, 'days_diff'])
plt.ylabel("Number of Days")
plt.title("Distribution of Difference Between Signing and Publication Dates of Executive Orders (Excluding Outliers)")
plt.show()
```



```
max_delay = publication_EO['days_diff'].idxmax()
max_delay_row = publication_EO.loc[max_delay]
max_delay_row
```

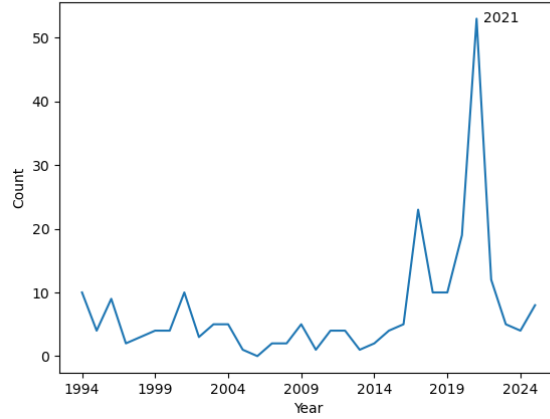
```
citation                85 FR 59171
document_number         2020 -20887
end_page                59172
html_url                https://www.federalregister.gov/documents/2020...
pdf_url                 https://www.govinfo.gov/content/pkg/FR -2020 -09...
type                    Presidential Document
subtype                 Executive Order
publication_date         09/18/2020
signing_date             2020 -07 -24
start_page              59171
title                   Lowering Drug Prices by Putting America First
disposition_notes       NaN
executive_order_number  13947
year                    2020
month                   07
pub_date_cleaned        2020 -09 -18 00:00:00
signed_date_cleaned     2020 -07 -24 00:00:00
days_diff              56
Name: 935, dtype: object
```

Looking at this entry gives us some interesting information about the dataset. First, both the links appear to be deprecated, so we will have to use the web to learn more about this EO. Going to the new website link for this EO ([click on this for link](#)) and going to the ‘Executive Order Details’ section shows us that the signing date is actually 09/13/2020, which is 5 days from the given publication date. Looking at our boxplot above, we can see that 5 is not an outlier while the one given above (07/24/2020 corresponding to 50+) is. This suggests that there may be some data integrity issues in our csv file.

3 Word Analysis

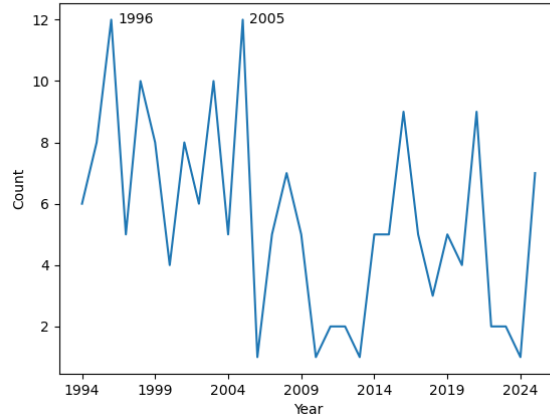
3.1 Frequency of EOs containing “Revoked” in Disposition Notes

Number of Executive Orders Containing the Word 'revoked' Over Time



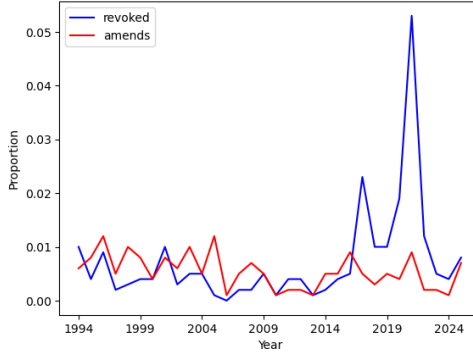
3.2 Frequency of EOs containing “Amends” in Disposition Notes

Number of Executive Orders Containing the Word 'amends' Over Time



3.3 Proportion of Revokals and Amendments Over Time

Proportions of Executive Orders Containing the Words 'revoked' and 'amends' Over Time



4 Language Model Experiment on Executive Order Titles

To explore whether a small pretrained language model can adapt to the stylistic conventions of U.S. executive order titles, we fine-tuned a causal language model on historical EO titles. The objective of this experiment is exploratory and qualitative, focusing on stylistic adaptation rather than predictive performance. All analysis were done in the LLM.ipynb file and results were saved in “outputs/llm_title_outputs.csv”.

```
import pandas as pd
llm_title_result = pd.read_csv("outputs/llm_title_outputs.csv")
for i, row in llm_title_result.iterrows():
    print(f"===== {i+1}th prompt: \"{row['prompt']}\n" =====\n")
    print(f"before -tuning: {row['before']}\n")
    print(f"after -tuning: {row['after']}\n")

===== 1th prompt: "Executive Order on " =====

before -tuning: Executive Order on ix -8 -9.

after -tuning: Executive Order on ixenable Employment, Economic Performance and Support for

===== 2th prompt: "Executive Order on Protecting " =====

before -tuning: Executive Order on Protecting - - - - -

after -tuning: Executive Order on Protecting illls and Jobs From Terrorist, Terrorist, and Te

===== 3th prompt: "Establishing the " =====
```

before -tuning: Establishing the vernacular is a process that allows us to incorporate the s

after -tuning: Establishing the étente Agreement on Civil Rights and Equal Opportunity and B

===== 4th prompt: "Amending Executive Order " =====

before -tuning: Amending Executive Order _____

after -tuning: Amending Executive Order _____ of 2018 to Prohibit Executive Order No. 13981.

===== 5th prompt: "Executive Order on National Security and " =====

before -tuning: Executive Order on National Security and Æ Trade in Information."

after -tuning: Executive Order on National Security and ills of the White House Council on t

Comparing the model outputs before and after fine-tuning reveals a noticeable shift toward more formal, EO-like phrasing after training. Post-fine-tuning outputs more frequently adopt administrative constructions such as “Establishing...”, “Amending Executive Order...”, and references to executive order numbers or effective dates. While some artifacts remain, the overall structure and tone of the generated titles better reflect the conventions of real executive order titles.

5 Author Contributions

Aniket Kamat: I performed the data analysis in main.ipynb, wrote most of the functions in EO_processing, and worked on a few of the reproducibility features. This includes the myst site, the Binder deployment, and the licensing.

Brycen Manners: I performed the data analysis in word-analysis.ipynb, created the environment.yml, and created the Makefile.

Soohyun Kim: I performed data analysis in LLM.ipynb, updated environment.yml, myst.yml, Makefile, pdf_builds, main.ipynb, rendered Myst Website and helped polish overall structure of project.

Yuyang Wu: I wrote functions and tests in for EO_processing, and create binder. I also do the ORCID thing and the final checking for the repo.

References

U.S. Government. Executive Orders Dataset. <https://catalog.data.gov>, 2025. URL <https://catalog.data.gov>. Accessed via the U.S. Government Open Data Portal.

T. Wolf and others. Transformers: State-of-the-Art Natural Language Processing. <https://huggingface.co/docs/transformers>, 2020. URL <https://huggingface.co/docs/transformers>. Hugging Face Transformers library.