Legal Data Analysis
Final Project

Research question

How does the subject matter of the complaint influence the length of the proceedings and the final conclusions of the ECHR?

"How long do proceedings before the Court usually last?

It is impossible to indicate the length of proceedings before the Court. The Court endeavours to deal with cases within three years after they are brought, but the examination of some cases can take longer and some can be processed more rapidly.

The length of the proceedings before the Court obviously varies depending on the case, the formation to which it is assigned, the diligence of the parties in providing the Court with information and many other factors, such as the holding of a hearing or referral to the Grand Chamber.

Some applications may be classified as urgent and handled on a priority basis, especially in cases where the applicant is alleged to be facing an imminent threat of physical harm. "

European Court of Human Rights: The ECHR in 50 questions, 2021

Approach
Collecting Data

- Pull judgments from **open data ECHR** dataset
- Scrape judgment for the date the complaint was lodged
- Create a **dataset** with the following factors:
  - Articles
  - Subarticle(s)
  - Start date (date the complaint was lodged)
  - End date (date of the judgment)
  - Importance
  - Conclusions
  - Number of associated articles

Approach

Analyzing Data

- **Calculate** the length in months of proceedings, based on:
  - Start date
  - End date

- **Compare** length of proceedings in months, based on:
  - Case importance - ranked 1 through 4
  - Conclusions - violation or no violation
  - Start date - before or after 2015
  - Category of article
    - different human rights violations
    - articles 2 to 18

The Process

```python
import requests
from bs4 import BeautifulSoup
import os
import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)
import pandas as pd
import regex as re
import json
from datetime import datetime, timedelta
import seaborn as sns
import matplotlib.pyplot as plt

# Make sure the ECHR file is in your directory
# Opening JSON file
f = open('echr_2_0_0_unstructured_cases.json')

# returns JSON object as a dictionary

data = json.load(f)
data_scraped = []

# This is the pattern to find the date that the complaint is lodged
pattern = r"(?:lodged)\s(?:\w+\W+){0,20}?(?:Article)\s(?:\w+\W+){0,50}?(\d{1,2}\s[A-Z][a-z]{2,8}\s\d{4})"
```

```python
# This function excludes cases with judgments prior to 2002, do not have a clear date that it was lodged in the main text,  or that other key elements are missing
def admissibility(case):
    e_date = pd.to_datetime(case['judgementdate'])
    ar = case['article']
    ar_again = case['__articles']
    c_list = []
    for k in case['conclusion']:
        co = k['element']
        c_list.append(co)
    c_list = list(set(c_list))
    for i, info in case.items():
        sub_error = []
        if i == 'content':
            for k in info:
                txt = str(case['content'][k][0:1000])
                txt = txt.replace("\\xa0", " ")
                pat = r"(?:lodged)\s(?:\w+\W+){0,20}?(?:Article)\s(?:\w+\W+){0,50}?(\d{1,2}\s[A-Z][a-z]{2,8}\s\d{4})"
                st_date = re.search(pat, txt)
                if st_date == None:
                    star_date = pd.to_datetime(0)
                else:
                    try:
                        star_date = pd.to_datetime(st_date.group(1))
                    except:
                        return False
                if star_date >= datetime(2002, 1,
                                         1) and e_date != "" and ar != [] and ar_again != "" and c_list != [] and star_date < e_date:
                    return True
                else:
                    return False
```

```python
accepted = 0
too_early = 0
rejected = 0

# Now we start getting the data, add the range here depending on how many decisions you want to pull
for decision in data:
    # We pass each decision through the admissibility function defined above, to see if we will include it in our database
    if admissibility(decision) == True:
        accepted += 1
        sublist = []
        iid = decision['itemid']
        sublist.append(iid)
        for p_id, p_info in decision.items():
            if p_id == 'content':
                for key in p_info:
                    # This will loop through the dictionary keys until it gets to content, go into the subdictionary, then pull the date it finds at the beginning of the doc
                    text = str(decision['content'][key][0:1000])
                    text = text.replace("\\xa0", " ")
                    s_date = re.search(pattern, text)
                    start_date = pd.to_datetime(s_date.group(1))
                    sublist.append(start_date)
```

```
        # Now we pull the judgment date from the decision and add it to the sublist
        end_date = pd.to_datetime(decision['judgementdate'])
        sublist.append(end_date)
        importance = decision['importance']
        sublist.append(importance)
        arts = decision['article']
        sublist.append(arts)
        arts_again = decision['__articles']
        sublist.append(arts_again)
        con_list = []
        # There are multiple conclusions, so we go through them in a loop to put them all in the same list, and thus column
        for key in decision['conclusion']:
            con = key['element']
            con_list.append(con)  # not sure if we should use this or ['__conclusion']
        # To eliminate any doubles, convert to a set and back
        con_list = list(set(con_list))
        sublist.append(con_list)
        con_again = decision['__conclusion']
        sublist.append(con_again)
        data_scraped.append(sublist)
    else:
        rejected += 1
        for p_id, p_info in decision.items():
            if p_id == 'content':
                for key in p_info:
                    text = str(decision['content'][key][0:1000])
                    text = text.replace("\\xa0", " ")
                    s_date = re.search(pattern, text)
                    try:
                        start_date = pd.to_datetime(s_date.group(1))
                    except:
                        continue
                    else:
                        if start_date < datetime(2002, 1, 1):
                            too_early += 1
```

```python
print("data scraped")
print("***")
print("accepted:")
print(accepted)
print(str((accepted / (accepted + rejected)) * 100) + "%")
print("***")
print("too early:")
print(too_early)
print(str((too_early / (accepted + rejected)) * 100) + "%")
print("***")
print("rejected for other reasons:")
print(rejected - too_early)
print(str((((rejected - too_early) / (accepted + rejected))) * 100) + "%")
print("***")
print("total rejected:")
print(rejected)
print(str(((rejected / (accepted + rejected))) * 100) + "%")
print("***")
```

```
data scraped
***
accepted:
9127
61.181123474996646%
***
too early:
3018
20.230593913393218%
***
rejected for other reasons:
2773
18.588282611610136%
***
total rejected:
5791
38.818876525003354%
```

```python
cf = pd.DataFrame(data_scraped,
                  columns=["Item ID", "Start_Date", "End_Date", "Importance", "Articles", "Articles and Subarticles",
                           "Conclusions", "Conclusions 2"])
```

```python
# Now we create a new data frame with a row for each article
aa = []
for row, columns in cf.iterrows():
    for article in cf['Articles'][row]:
        bb = []
        bb.append(article[0:2])

        # Now we separate the articles and subarticles in a different row, splitting them so each number is separate
        sub_arts = []
        subarts_again = cf['Articles and Subarticles'][row]
        arts_split = subarts_again.replace("+", ";").split(";")
        sub_arts.append(arts_split)

        # Now we put them all into a single list
        concat_arts = [j for i in sub_arts for j in i]

        # And remove duplicates
        artsies = list(set(concat_arts))
        real_subs = []

        # Now we find the subarticles by identifying which ones have a '-'
        for sub in artsies:
            if "-" in sub:

                # And we add the subarticle to the row only if it starts with the same number as the main article of the row
                if article in sub[0:2]:
                    real_subs.append(sub)
```

```
        bb.append(real_subs)
        bb.append(cf['Item ID'][row])
        bb.append(cf['Start_Date'][row])
        bb.append(cf['End_Date'][row])
        bb.append(cf['Importance'][row])
        bb.append(cf['Conclusions'][row])
        bb.append(cf['Conclusions 2'][row])

        # We add the list of associated articles to the row
        asso_arts = []
        for asso in cf['Articles'][row]:
            if asso != article:
                asso_arts.append(asso)
        bb.append(asso_arts)

        # We measure and then add the number of associated articles to the row
        artlen = len(asso_arts)
        bb.append(artlen)
        aa.append(bb)
df = pd.DataFrame(aa,
                  columns=['Article', 'Subarticle(s)', 'Item ID', 'Start_Date', 'End_Date', 'Importance', 'Conclusions',
                           'Conclusions 2', 'Associated Articles', 'Number_of_Associated_Articles'])
print("The Data Frame has been created")

# Turn the empty lists into empty strings
df['Subarticle(s)'] = df['Subarticle(s)'].apply(lambda y: "" if len(y) == 0 else y)
df['Associated Articles'] = df['Associated Articles'].apply(lambda y: "" if len(y) == 0 else y)

print(
    "------------------------------------------------------------------------------------------------------------------------------------")
```

```python
# Analysis of the dataframe "cf"

# We create a new column with time delta
delta_months = []
for i in cf.index:
    months = ((cf["End_Date"][i] - cf["Start_Date"][i]).days) / 30.437
    delta_months.append(months)
cf = cf.assign(Delta_months=delta_months)

# We create a new column with conclusions (violation or no violation)
violations = []
for i in cf.index:
    if "no violation" in cf["Conclusions 2"][i].lower():
        ccl = "No"
    else:
        ccl = "Yes"
    violations.append(ccl)
cf = cf.assign(Violation=violations)
# We calculate and print some data on the distribution of judgments (based on cf)

Mean_cf_months = cf["Delta_months"].mean().round()
Median_cf_months = cf["Delta_months"].median().round()
Min_cf_months = cf["Delta_months"].min().round()
Max_cf_months = cf["Delta_months"].max().round()

print("\n")
print("Length of proceedings")
print("Average = " + str(Mean_cf_months) + " months (" + str((Mean_cf_months / 12).round(1)) + " years)")
print("Median = " + str(Median_cf_months) + " months (" + str((Median_cf_months / 12).round(1)) + " years)")
print("Minimum = " + str(Min_cf_months) + " months")
print("Maximum = " + str(Max_cf_months) + " months (" + str((Max_cf_months / 12).round(1)) + " years)")
print("\n")
```

```python
# We create a boxplot with the length of proceeding depending on conclusions

data_1 = cf[cf.Violation == "Yes"]["Delta_months"]
data_2 = cf[cf.Violation == "No"]["Delta_months"]
data = [data_1, data_2]
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111)
bp = ax.boxplot(data, patch_artist=True,
                notch='True', vert=0)
colors = ['#0000FF', '#00FF00']
for patch, color in zip(bp['boxes'], colors):
    patch.set_facecolor(color)
for whisker in bp['whiskers']:
    whisker.set(color='#8B008B',
                linewidth=1.5,
                linestyle=":")
for cap in bp['caps']:
    cap.set(color='#8B008B',
            linewidth=2)
for median in bp['medians']:
    median.set(color='red',
               linewidth=3)
for flier in bp['fliers']:
    flier.set(marker='D',
              color='#e7298a',
              alpha=0.5)
ax.set_yticklabels(['Violation', 'No violation'])
plt.title("Length of proceeding depending on conclusions")
ax.get_xaxis().tick_bottom()
ax.get_yaxis().tick_left()
plt.xlabel('Months')
plt.ylabel('Conclusions')
plt.show()
```

```python
# We create a boxplot with the length of proceeding depending on importance

data_1 = cf[cf.Importance == "1"]["Delta_months"]
data_2 = cf[cf.Importance == "2"]["Delta_months"]
data_3 = cf[cf.Importance == "3"]["Delta_months"]
data_4 = cf[cf.Importance == "4"]["Delta_months"]
data = [data_1, data_2, data_3, data_4]
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111)
bp = ax.boxplot(data, patch_artist=True,
                notch='True', vert=0)
colors = ['#0000FF', '#00FF00',
          '#FFFF00', '#FF00FF']
for patch, color in zip(bp['boxes'], colors):
    patch.set_facecolor(color)
for whisker in bp['whiskers']:
    whisker.set(color='#8B008B',
                linewidth=1.5,
                linestyle=":")
for cap in bp['caps']:
    cap.set(color='#8B008B',
            linewidth=2)
for median in bp['medians']:
    median.set(color='red',
               linewidth=3)
for flier in bp['fliers']:
    flier.set(marker='D',
              color='#e7298a',
              alpha=0.5)
ax.set_yticklabels(['Importance 1', 'Importance 2',
                    'Importance 3', 'Importance 4'])
plt.title("Length of proceeding depending on importance")
ax.get_xaxis().tick_bottom()
ax.get_yaxis().tick_left()
plt.xlabel('Months')
plt.ylabel('Importance')
plt.show()
```

```python
# We create a boxplot with the length of proceeding depending on importance Start Date (before/after 2015)

data_1 = cf[cf.Start_Date < '2015-01-01']["Delta_months"]
data_2 = cf[cf.Start_Date >= '2015-01-01']["Delta_months"]
data = [data_1, data_2]
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111)
bp = ax.boxplot(data, patch_artist=True,
                notch='True', vert=0)
colors = ['#FFFF00', '#FF00FF']
for patch, color in zip(bp['boxes'], colors):
    patch.set_facecolor(color)
for whisker in bp['whiskers']:
    whisker.set(color='#8B008B',
                linewidth=1.5,
                linestyle=":")
for cap in bp['caps']:
    cap.set(color='#8B008B',
            linewidth=2)
for median in bp['medians']:
    median.set(color='red',
               linewidth=3)
for flier in bp['fliers']:
    flier.set(marker='D',
              color='#e7298a',
              alpha=0.5)
```

```python
ax.set_yticklabels(['Before 2015', 'After 2015'])
plt.title("Length of proceeding depending on the start date")
ax.get_xaxis().tick_bottom()
ax.get_yaxis().tick_left()
plt.xlabel('Months')
plt.ylabel('Start date')
plt.show()
```

```python
# Analysis of the dataframe "df"

# We create a new column with time delta
delta_months = []
for i in df.index:
    months = ((df["End_Date"][i] - df["Start_Date"][i]).days) / 30.437
    delta_months.append(months)
df = df.assign(Delta_months=delta_months)
```

```python
# We create a dictionary of articles with number and name ("subject")

dictionary = {}
for i in df.Article:
    # We exclude articles from protocols
    if "p" in i:
        pass
    else:
        a = int(i)
```

```python
    if a == 2:
        dictionary[a] = "Right to life"
    elif a == 3:
        dictionary[a] = "Prohibition of torture"
    elif a == 4:
        dictionary[a] = "Prohibition of slavery and forced labour"
    elif a == 5:
        dictionary[a] = "Right to liberty and security"
    elif a == 6:
        dictionary[a] = "Right to a fair trial"
    elif a == 7:
        dictionary[a] = "No punishment without law"
    elif a == 8:
        dictionary[a] = "Right to respect for private and family life"
    elif a == 9:
        dictionary[a] = "Freedom of thought, conscience and religion"
```

```python
    elif a == 10:
        dictionary[a] = "Freedom of expression"
    elif a == 11:
        dictionary[a] = "Freedom of assembly and association"
    elif a == 12:
        dictionary[a] = "Right to marry"
    elif a == 13:
        dictionary[a] = "Right to an effective remedy"
    elif a == 14:
        dictionary[a] = "Prohibition of discrimination"
    elif a == 15:
        dictionary[a] = "Derogation in time of emergency"
    elif a == 16:
        dictionary[a] = "Restrictions on political activity of aliens"
    elif a == 17:
        dictionary[a] = "Prohibition of abuse of rights"
    elif a == 18:
        dictionary[a] = "Limitation on use of restrictions on rights"
```

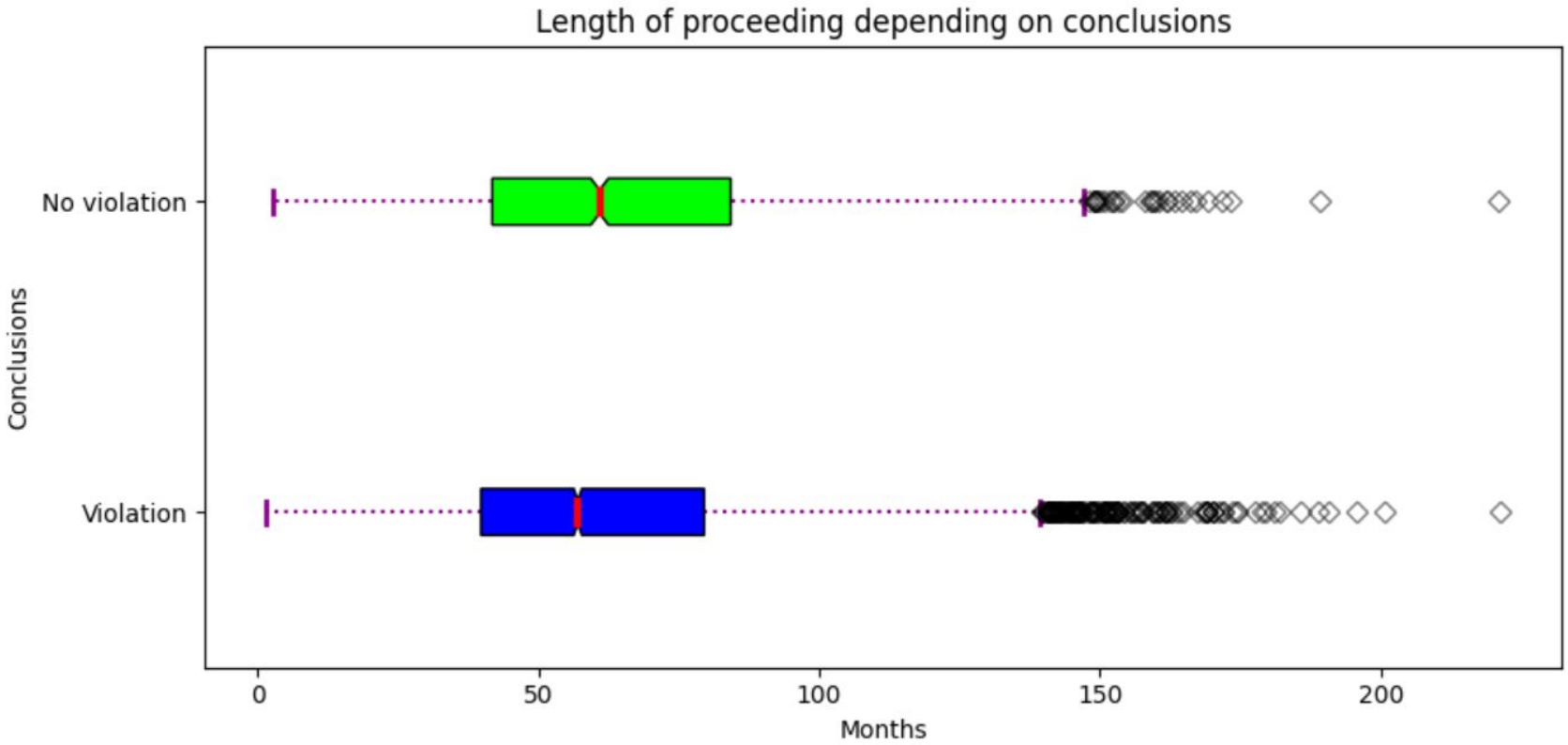
```python
from collections import OrderedDict

dictionary = OrderedDict(sorted(dictionary.items()))

# We add a column with the name of each article in df
new_column = []
for i in df.index:
    if "p" in df["Article"][i]:
        new_column.append("Not relevant")
    else:
        a = int(df["Article"][i])
        if a in dictionary.keys():
            new_column.append(dictionary[a])
        else:
            new_column.append("Not relevant")
df = df.assign(Subject=new_column)
```

```python
# We add a column with the name of each article in df
new_column = []
for i in df.index:
    if "p" in df["Article"][i]:
        new_column.append("Not relevant")
    else:
        a = int(df["Article"][i])
        if a in dictionary.keys():
            new_column.append(dictionary[a])
        else:
            new_column.append("Not relevant")
df = df.assign(Subject=new_column)

# We calculate the mean and median of Delta_months, the number of judgments and the number of associated articles for each relevant article
data_articles = {}
for i in dictionary.keys():
    new_df = df[(df.Article == str(i))]
    new_len = len(new_df)
    new_average = new_df["Delta_months"].mean().round()
    new_median = new_df["Delta_months"].median().round()
    new_average_nb = new_df["Number_of_Associated_Articles"].mean().round()
    new_list = [i, dictionary[i], new_len, new_average, new_median, new_average_nb]
    data_articles[i] = new_list
df_articles = pd.DataFrame.transpose(pd.DataFrame(data_articles))
df_articles.columns = ["Article", "Subject", "Number_Judgments", "Average_Months", "Median_Months",
                       "Average_Number_Associated Articles"]

# We add the number of judgments in a new column in df
new_column = []
df_articles2 = df_articles.set_index("Subject")
for i in df.Subject:
    if i == "Not relevant":
        new_column.append(0)
    else:
        new = df_articles2["Number_Judgments"][i]
        new_column.append(int(new))
df = df.assign(number_judgments=new_column)
```

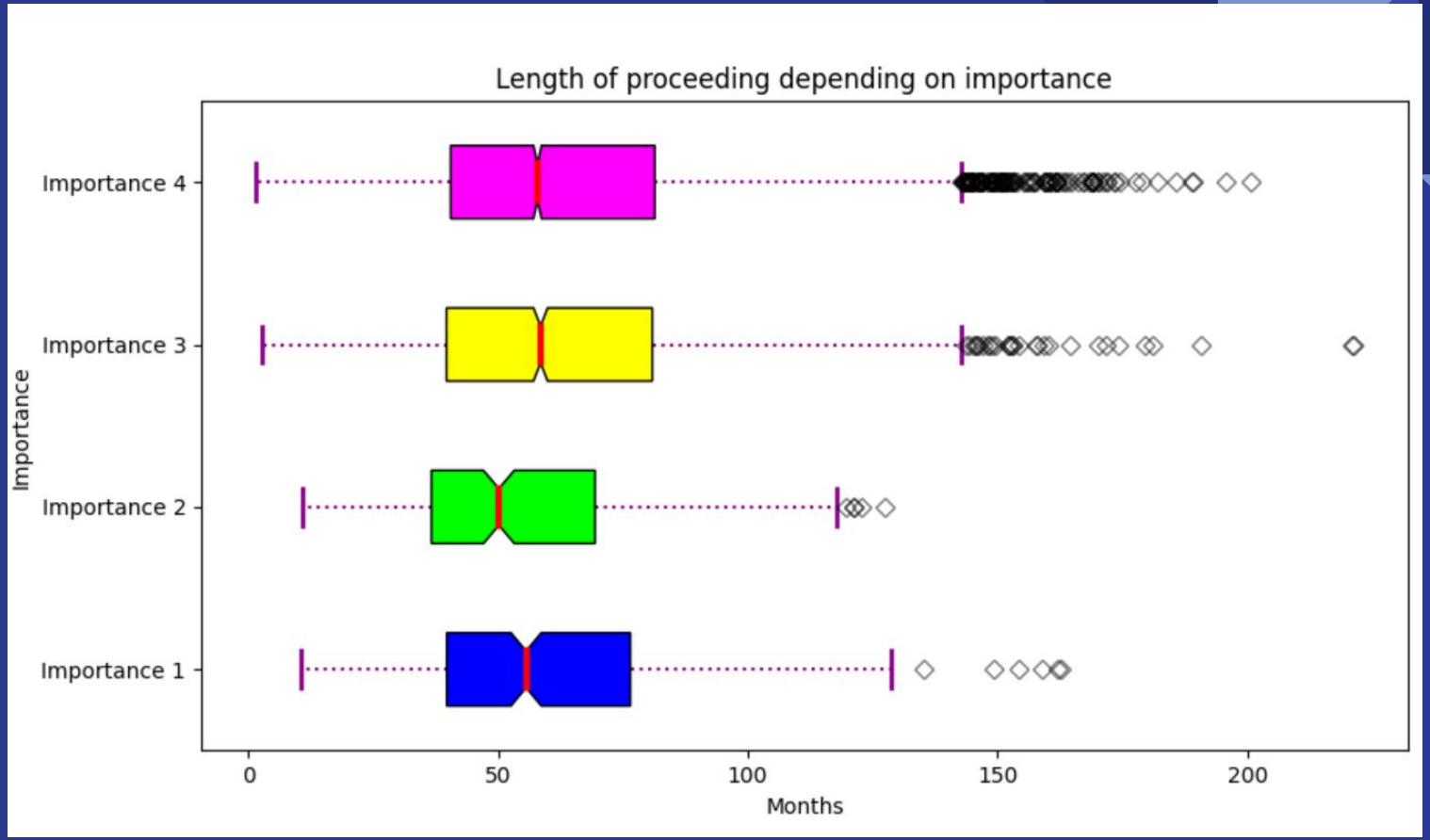
```
# We create scatterplots (based on medians in df) and we save the files

sns.scatterplot(data=df_articles, x="Median_Months", y="Number_Judgments", hue="Subject")

plt.savefig('Scatterplot_all.png')
plt.close()

sns.scatterplot(data=df_articles[df_articles.Number_Judgments > 100], x="Median_Months", y="Number_Judgments",
                hue="Subject")

plt.savefig('Scatterplot_morethan100.png')
plt.close()

sns.scatterplot(data=df_articles[df_articles.Number_Judgments < 100], x="Median_Months", y="Number_Judgments",
                hue="Subject")

plt.savefig('Scatterplot_lessthan100.png')
plt.close()

# We create distribution plots (based on df)

sns.displot(data=df[df.Subject != "Not relevant"], x="Delta_months", hue="Subject", multiple="stack")

sns.displot(data=df[df.Subject != "Not relevant"][df.number_judgments < 100], x="Delta_months", hue="Subject",
            multiple="stack")

sns.displot(data=df[df.Subject != "Not relevant"][df.Delta_months > 150], x="Delta_months", hue="Subject",
            multiple="stack")

print("------------------------------------------------------------------------------------------")
```

The Results

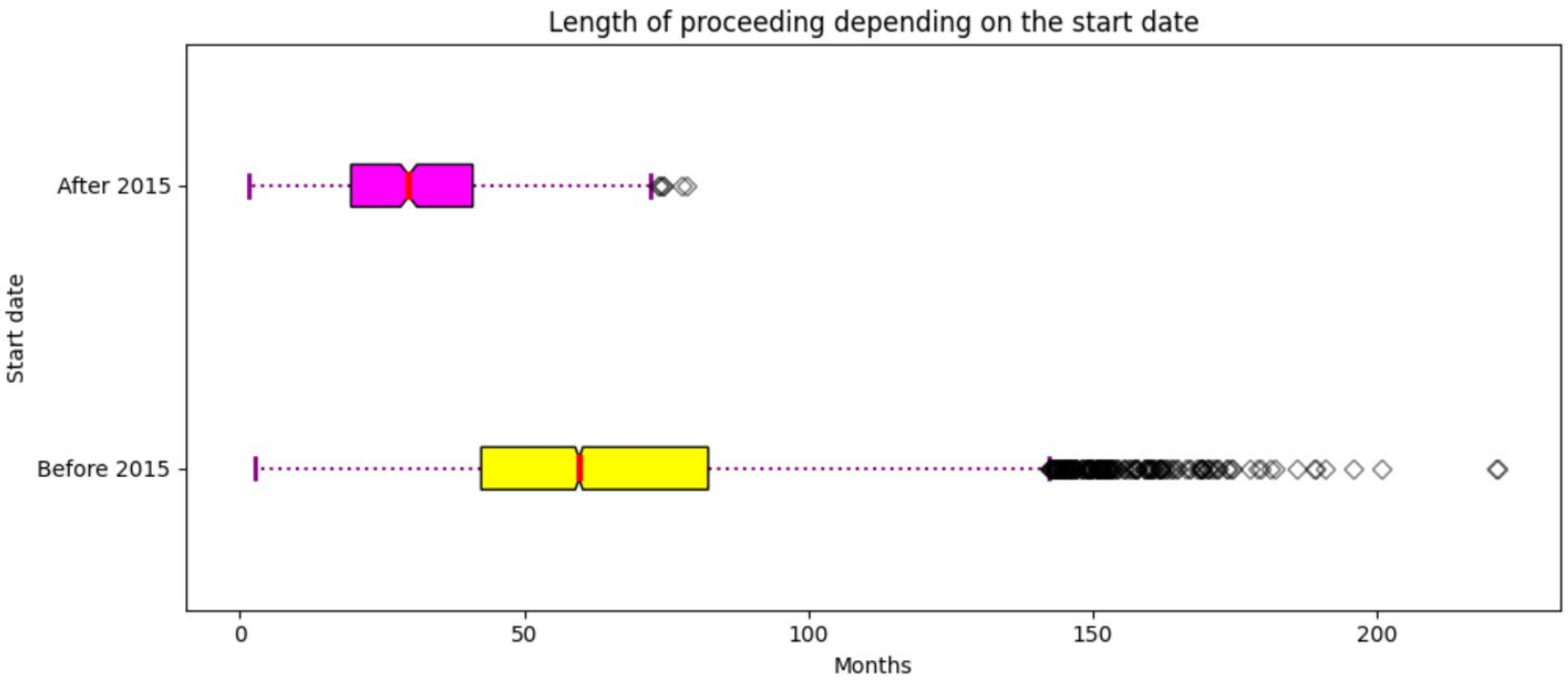
```
Length of proceedings
Average = 63.0 months (5.2 years)
Median = 58.0 months (4.8 years)
Minimum = 2.0 months
Maximum = 221.0 months (18.4 years)
```



The ECHR responds slightly more quickly when there is a violation.

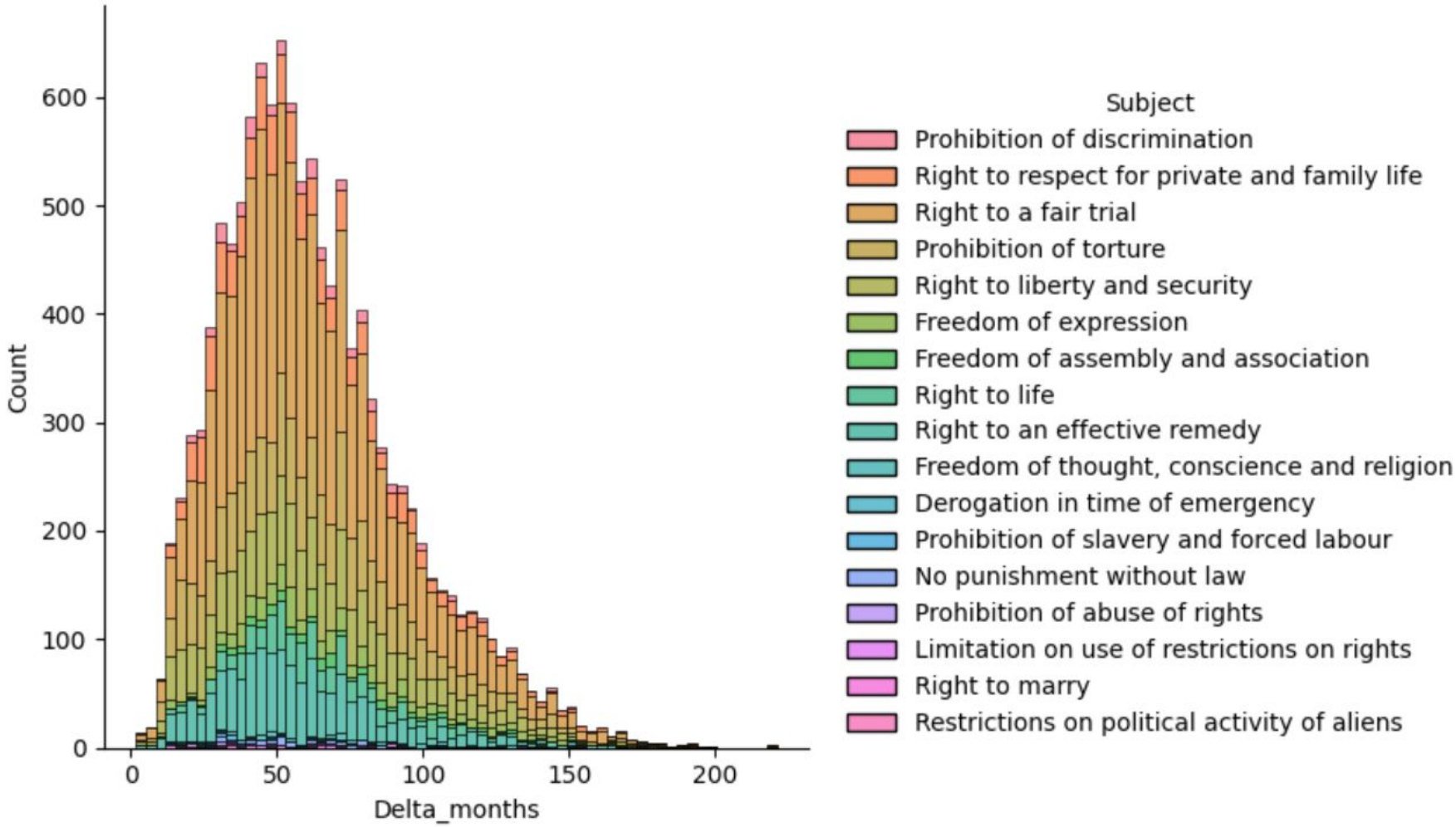


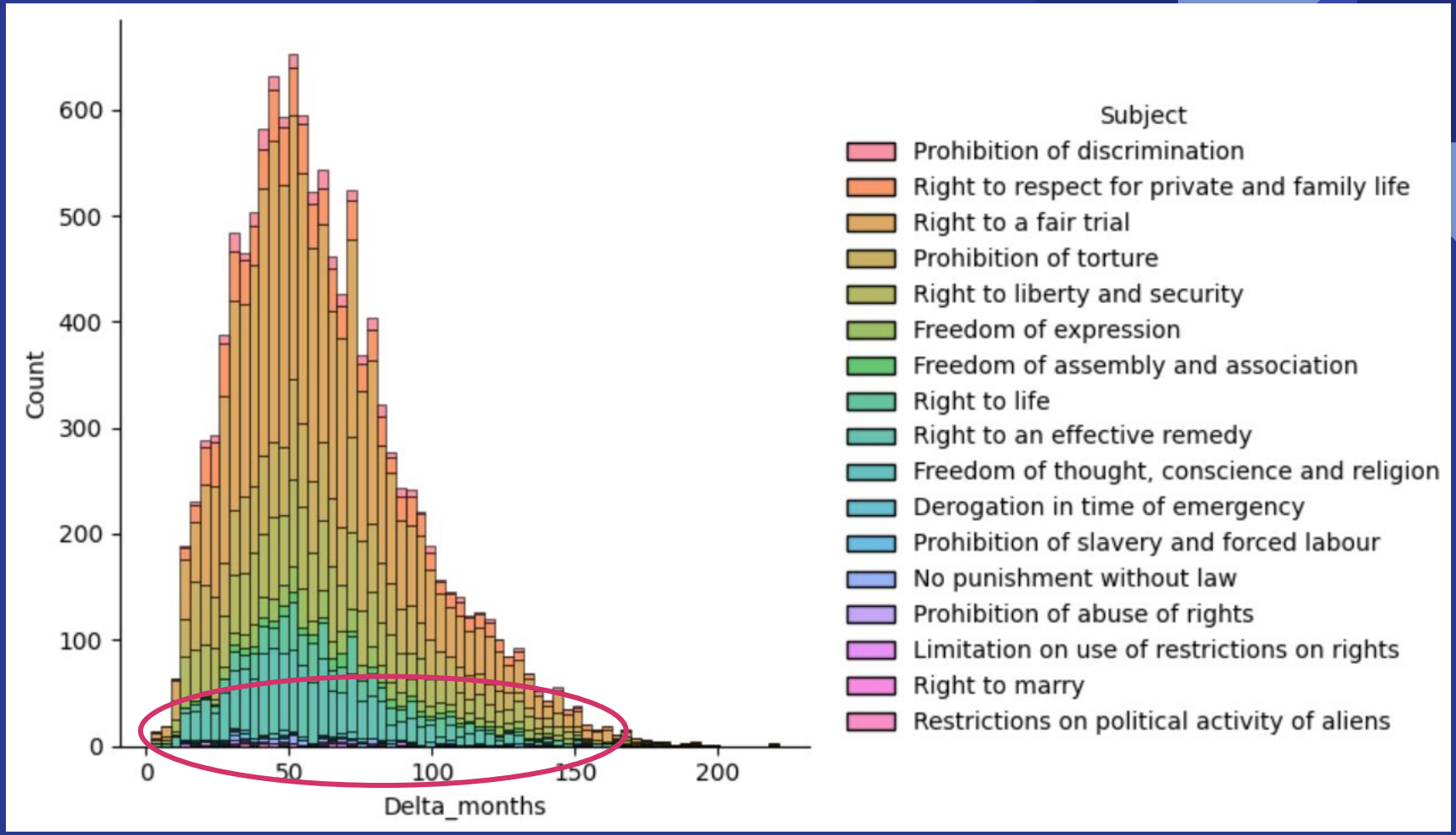
The ECHR responds slightly more quickly for cases of higher importance.

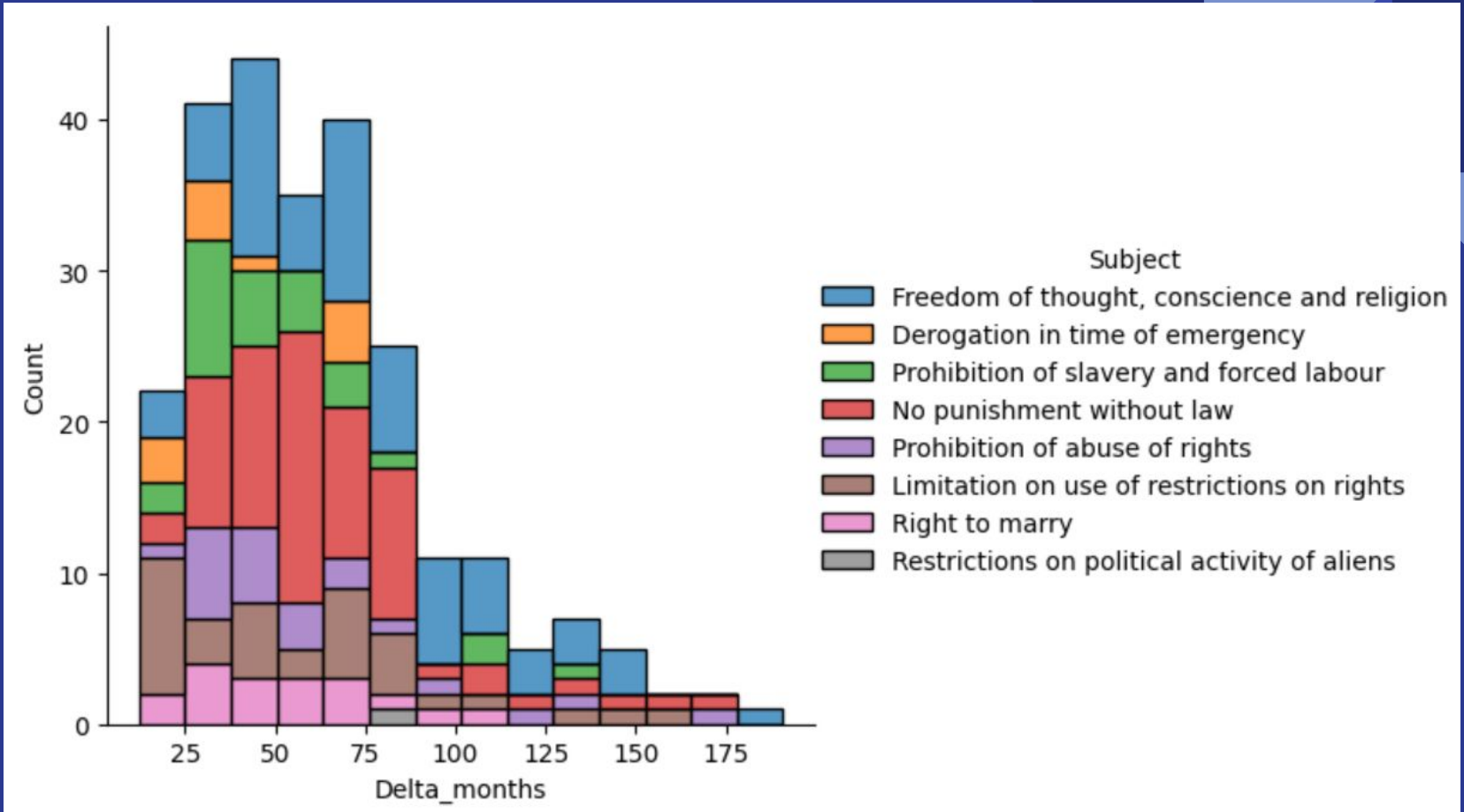


A massive increase in the number of requests in 2011 led the ECHR to change its method : proceedings are faster after 2015. (https://www.rue89strasbourg.com/parcours-requetes-cour-europeenne-des-droits-homme-125332)

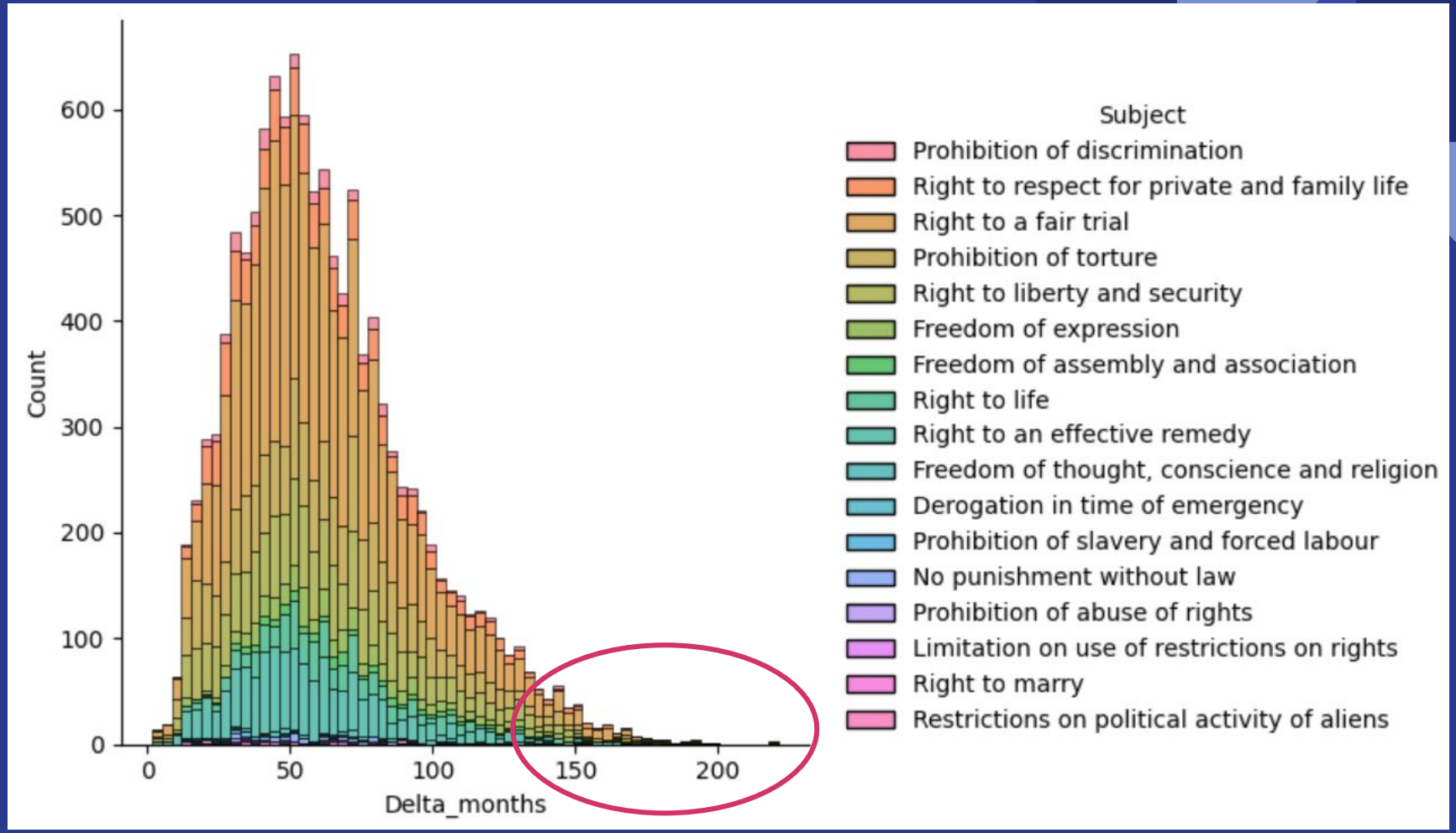
(N.B: The procedure has a maximum duration of 84 months if the complaint was filed in 2015)

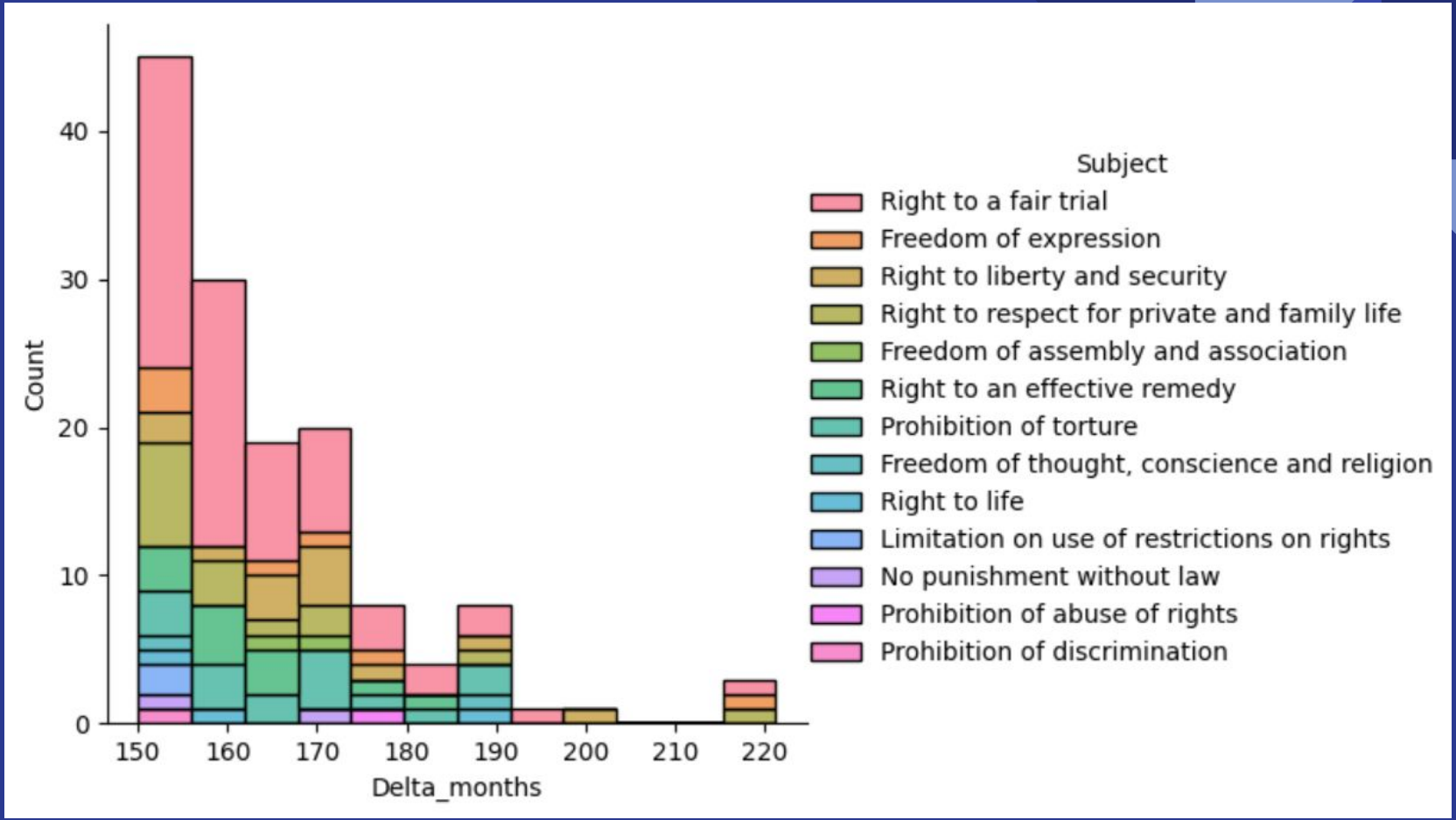

Count
600
500
400
300
200
100
0
0
50
100
150
200
Delta_months
Subject
Prohibition of discrimination
Right to respect for private and family life
Right to a fair trial
Prohibition of torture
Right to liberty and security
Freedom of expression
Freedom of assembly and association
Right to life
Right to an effective remedy
Freedom of thought, conscience and religion
Derogation in time of emergency
Prohibition of slavery and forced labour
No punishment without law
Prohibition of abuse of rights
Limitation on use of restrictions on rights
Right to marry
Restrictions on political activity of aliens


Count
600
500
400
300
200
100
0
0
50
100
150
200
Delta_months
Subject
Prohibition of discrimination
Right to respect for private and family life
Right to a fair trial
Prohibition of torture
Right to liberty and security
Freedom of expression
Freedom of assembly and association
Right to life
Right to an effective remedy
Freedom of thought, conscience and religion
Derogation in time of emergency
Prohibition of slavery and forced labour
No punishment without law
Prohibition of abuse of rights
Limitation on use of restrictions on rights
Right to marry
Restrictions on political activity of aliens

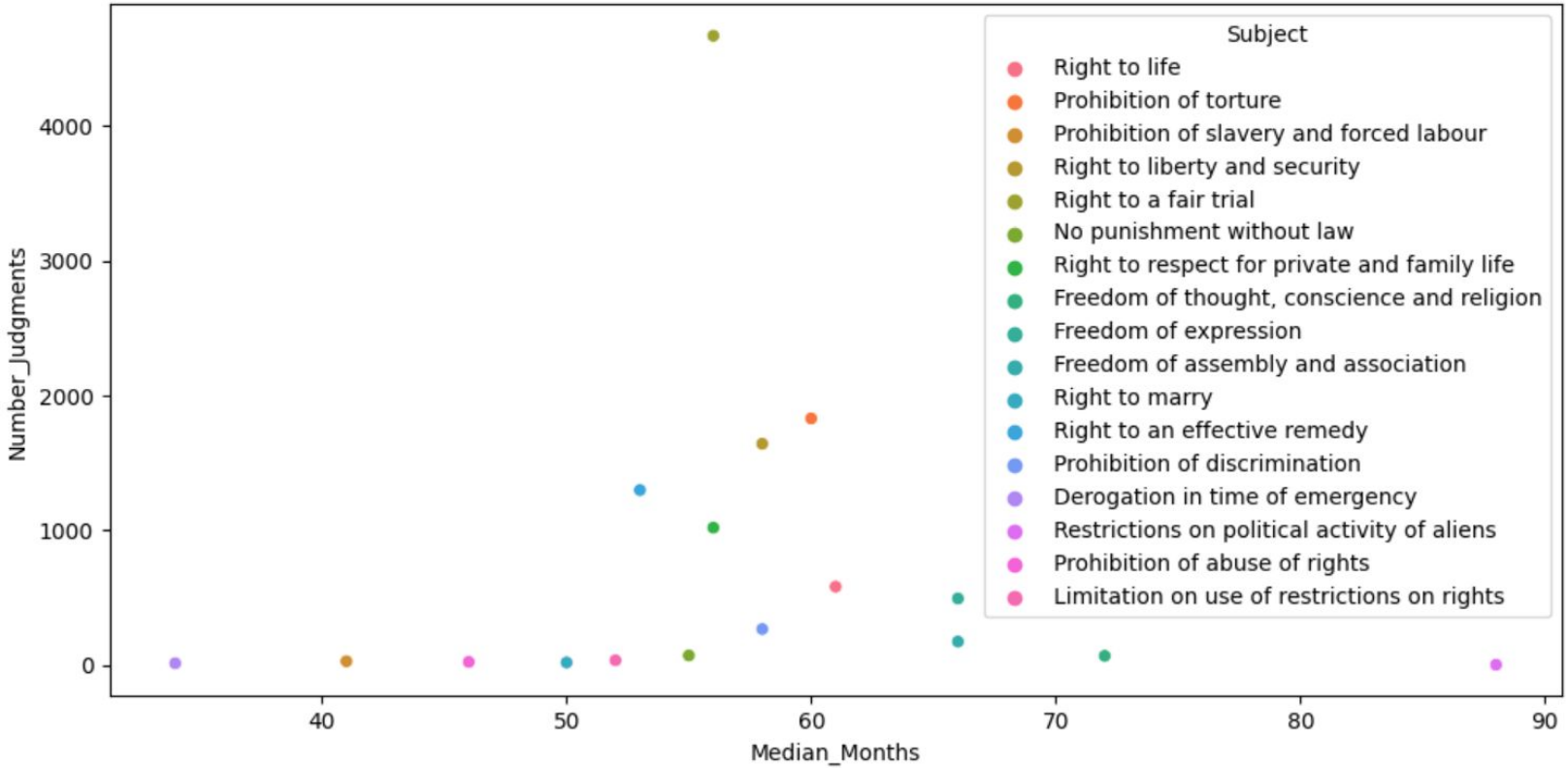


Faster proceedings: derogation in time of emergency, prohibition of slavery, right to marry


Count
600
500
400
300
200
100
0
0
50
100
150
200
Delta_months
Subject
Prohibition of discrimination
Right to respect for private and family life
Right to a fair trial
Prohibition of torture
Right to liberty and security
Freedom of expression
Freedom of assembly and association
Right to life
Right to an effective remedy
Freedom of thought, conscience and religion
Derogation in time of emergency
Prohibition of slavery and forced labour
No punishment without law
Prohibition of abuse of rights
Limitation on use of restrictions on rights
Right to marry
Restrictions on political activity of aliens

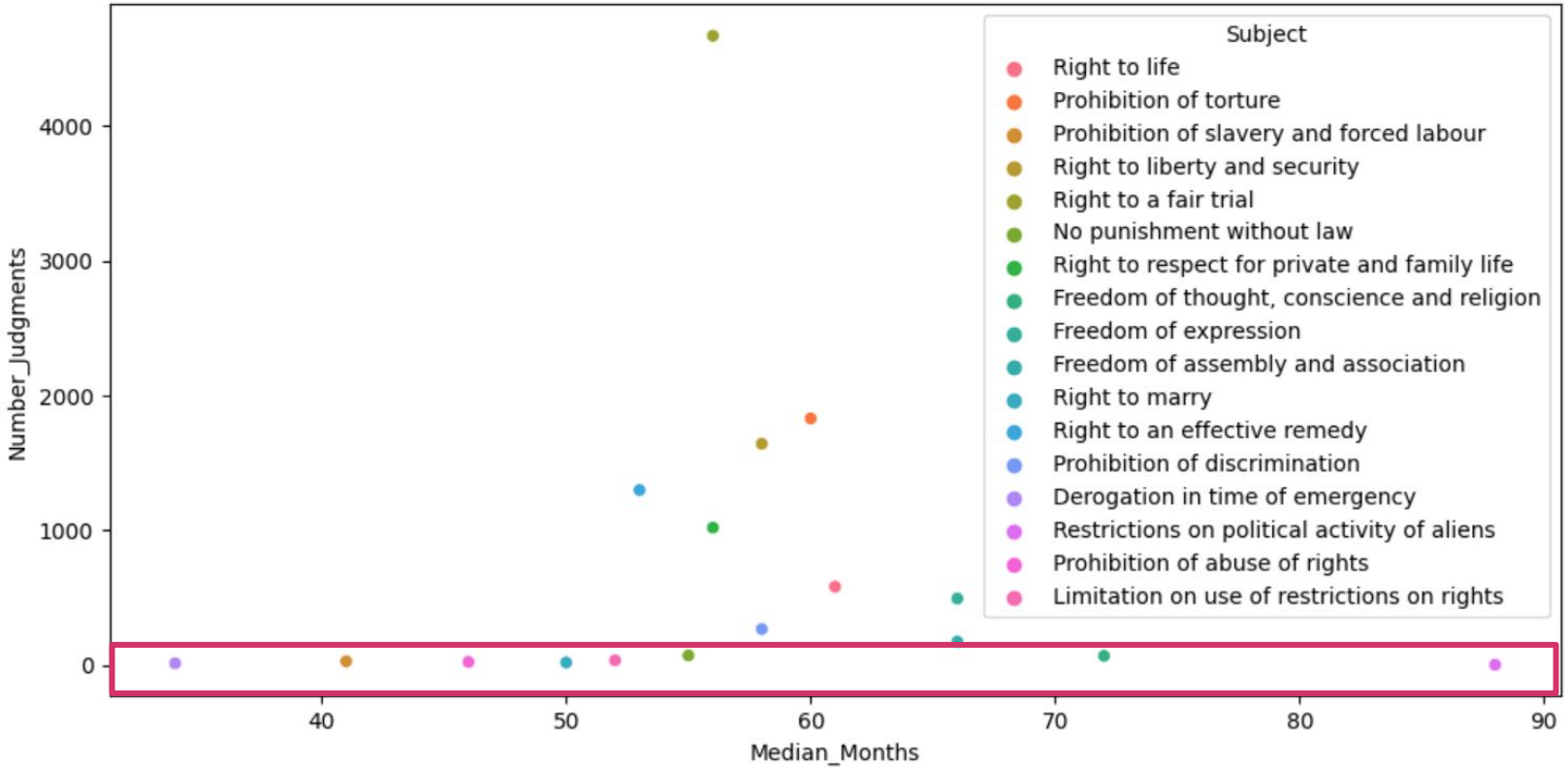


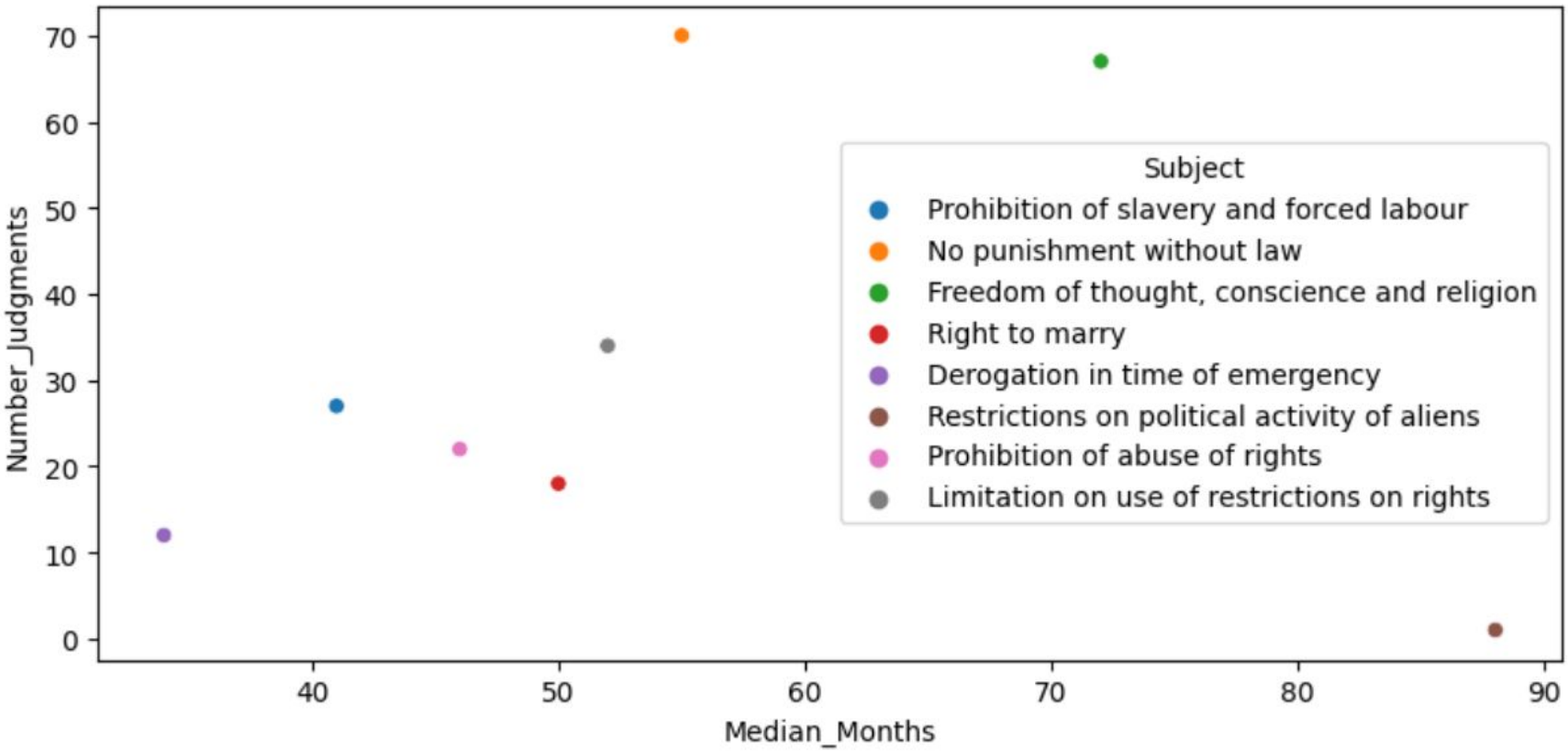
Two 18 years proceedings : article 6 (right to a fair trial) ; articles 8 (right to respect for private life) and 10 (freedom of expression)

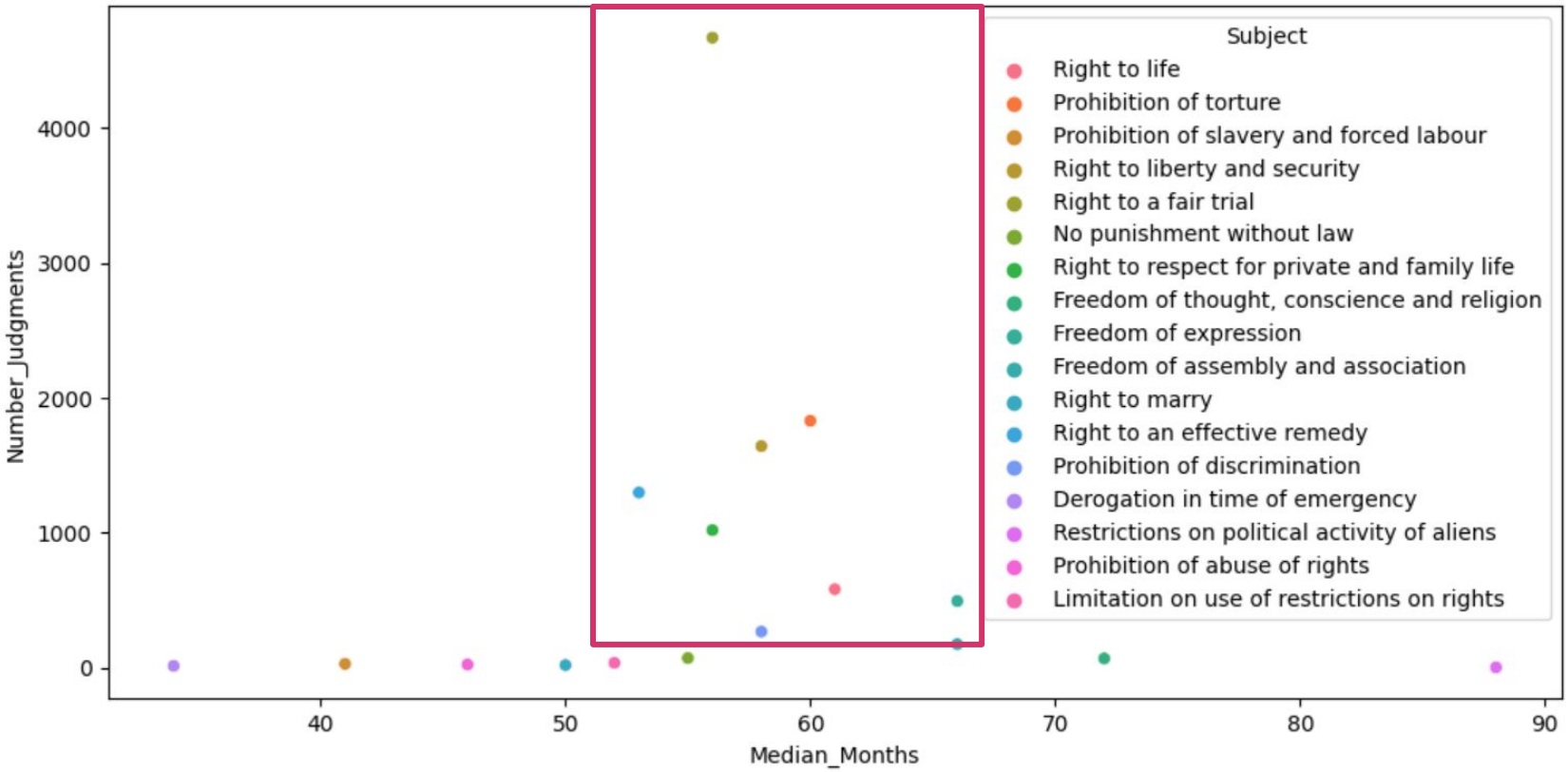


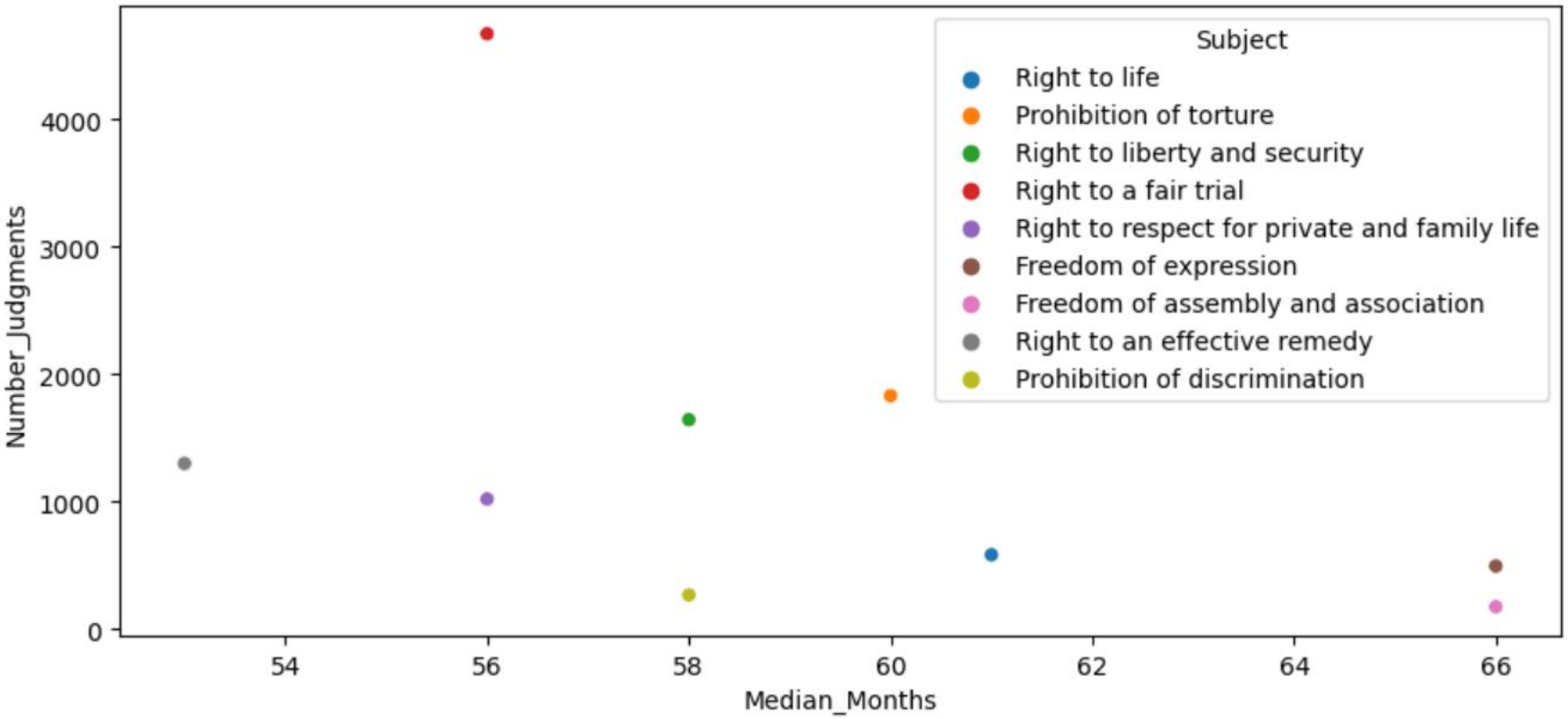
Faster proceedings : **derogation in time of emergency** (34), **prohibition of slavery** (41), **prohibition of abuse of rights** (46)

Longer proceedings : **restrictions on political activity of aliens** (88, 1 judgment), **freedom of thought** (72), **freedom of expression** (66), **freedom of assembly** (66)


Subject
Right to life
Prohibition of torture
Prohibition of slavery and forced labour
Right to liberty and security
Right to a fair trial
No punishment without law
Right to respect for private and family life
Freedom of thought, conscience and religion
Freedom of expression
Freedom of assembly and association
Right to marry
Right to an effective remedy
Prohibition of discrimination
Derogation in time of emergency
Restrictions on political activity of aliens
Prohibition of abuse of rights
Limitation on use of restrictions on rights
Number_Judgments
0
1000
2000
3000
4000
40
50
60
70
80
90
Median_Months


Number_Judgments
Median_Months
0
10
20
30
40
50
60
70
40
50
60
70
80
90
Subject
Prohibition of slavery and forced labour
No punishment without law
Freedom of thought, conscience and religion
Right to marry
Derogation in time of emergency
Restrictions on political activity of aliens
Prohibition of abuse of rights
Limitation on use of restrictions on rights


Subject
Right to life
Prohibition of torture
Prohibition of slavery and forced labour
Right to liberty and security
Right to a fair trial
No punishment without law
Right to respect for private and family life
Freedom of thought, conscience and religion
Freedom of expression
Freedom of assembly and association
Right to marry
Right to an effective remedy
Prohibition of discrimination
Derogation in time of emergency
Restrictions on political activity of aliens
Prohibition of abuse of rights
Limitation on use of restrictions on rights
Number_Judgments
0
1000
2000
3000
4000
Median_Months
40
50
60
70
80
90


Subject
Right to life
Prohibition of torture
Right to liberty and security
Right to a fair trial
Right to respect for private and family life
Freedom of expression
Freedom of assembly and association
Right to an effective remedy
Prohibition of discrimination
Number_Judgments
0
1000
2000
3000
4000
54
56
58
60
62
64
66
Median_Months

(Analysis)

Thank you