



Legal Data Analysis **Final Project**

Kiana Saint-Macary & Elena Susini

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. "

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

```
1 import requests
2 from bs4 import BeautifulSoup
3 import os
4 import warnings
5
6 warnings.simplefilter(action='ignore', category=FutureWarning)
7 import pandas as pd
8 import regex as re
9 import json
10 from datetime import datetime, timedelta
11 import seaborn as sns
12 import matplotlib.pyplot as plt
13
14 # Make sure the ECHR file is in your directory
15 # Opening JSON file
16 f = open('echr_2_0_0_unstructured_cases.json')
17
18 # returns JSON object as a dictionary
19
20 data = json.load(f)
21 data_scraped = []
22
23 # This is the pattern to find the date that the complaint is lodged
24 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})"
```

```

27 # 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
28 def admissibility(case):
29     e_date = pd.to_datetime(case['judgementdate'])
30     ar = case['article']
31     ar_again = case['__articles']
32     c_list = []
33     for k in case['conclusion']:
34         co = k['element']
35         c_list.append(co)
36     c_list = list(set(c_list))
37     for i, info in case.items():
38         sub_error = []
39         if i == 'content':
40             for k in info:
41                 txt = str(case['content'][k][0:1000])
42                 txt = txt.replace("\xa0", " ")
43                 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})"
44                 st_date = re.search(pat, txt)
45                 if st_date == None:
46                     star_date = pd.to_datetime(0)
47                 else:
48                     try:
49                         star_date = pd.to_datetime(st_date.group(1))
50                     except:
51                         return False
52                 if star_date >= datetime(2002, 1,
53                                     1) and e_date != "" and ar != [] and ar_again != "" and c_list != [] and star_date < e_date:
54                     return True
55             else:
56                 return False

```

```
59 accepted = 0
60 too_early = 0
61 rejected = 0
62
63 # Now we start getting the data, add the range here depending on how many decisions you want to pull
64 for decision in data:
65     # We pass each decision through the admissibility function defined above, to see if we will include it in our database
66     if admissibility(decision) == True:
67         accepted += 1
68         sublist = []
69         iid = decision['itemid']
70         sublist.append(iid)
71         for p_id, p_info in decision.items():
72             if p_id == 'content':
73                 for key in p_info:
74                     # 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
75                     text = str(decision['content'][key][0:1000])
76                     text = text.replace("\xa0", " ")
77                     s_date = re.search(pattern, text)
78                     start_date = pd.to_datetime(s_date.group(1))
79                     sublist.append(start_date)
```



```

81     # Now we pull the judgment date from the decision and add it to the sublist
82     end_date = pd.to_datetime(decision['judgementdate'])
83     sublist.append(end_date)
84     importance = decision['importance']
85     sublist.append(importance)
86     arts = decision['article']
87     sublist.append(arts)
88     arts_again = decision['__articles']
89     sublist.append(arts_again)
90     con_list = []
91     # There are multiple conclusions, so we go through them in a loop to put them all in the same list, and thus column
92     for key in decision['conclusion']:
93         con = key['element']
94         con_list.append(con) # not sure if we should use this or ['__conclusion']
95     # To eliminate any doubles, convert to a set and back
96     con_list = list(set(con_list))
97     sublist.append(con_list)
98     con_again = decision['__conclusion']
99     sublist.append(con_again)
100    data_scraped.append(sublist)
101 else:
102     rejected += 1
103     for p_id, p_info in decision.items():
104         if p_id == 'content':
105             for key in p_info:
106                 text = str(decision['content'][key][0:1000])
107                 text = text.replace("\xa0", " ")
108                 s_date = re.search(pattern, text)
109                 try:
110                     start_date = pd.to_datetime(s_date.group(1))
111                 except:
112                     continue
113                 else:
114                     if start_date < datetime(2002, 1, 1):
115                         too_early += 1

```

[illegible]

```
140 # Now we create a new data frame with a row for each article
141 aa = []
142 for row, columns in cf.iterrows():
143     for article in cf['Articles'][row]:
144         bb = []
145         bb.append(article[0:2])
146
147         # Now we separate the articles and subarticles in a different row, splitting them so each number is separate
148         sub_arts = []
149         subarts_again = cf['Articles and Subarticles'][row]
150         arts_split = subarts_again.replace("+", ";").split(";")
151         sub_arts.append(arts_split)
152
153         # Now we put them all into a single list
154         concat_arts = [j for i in sub_arts for j in i]
155
156         # And remove duplicates
157         artsies = list(set(concat_arts))
158         real_subs = []
159
160         # Now we find the subarticles by identifying which ones have a '-'
161         for sub in artsies:
162             if "-" in sub:
163
164                 # And we add the subarticle to the row only if it starts with the same number as the main article of the row
165                 if article in sub[0:2]:
166                     real_subs.append(sub)
```

```

167     bb.append(real_subs)
168     bb.append(cf['Item ID'][row])
169     bb.append(cf['Start_Date'][row])
170     bb.append(cf['End_Date'][row])
171     bb.append(cf['Importance'][row])
172     bb.append(cf['Conclusions'][row])
173     bb.append(cf['Conclusions 2'][row])
174
175     # We add the list of associated articles to the row
176     asso_arts = []
177     for asso in cf['Articles'][row]:
178         if asso != article:
179             asso_arts.append(asso)
180     bb.append(asso_arts)
181
182     # We measure and then add the number of associated articles to the row
183     artlen = len(asso_arts)
184     bb.append(artlen)
185     aa.append(bb)
186 df = pd.DataFrame(aa,
187                   columns=['Article', 'Subarticle(s)', 'Item ID', 'Start_Date', 'End_Date', 'Importance', 'Conclusions',
188                           'Conclusions 2', 'Associated Articles', 'Number_of_Associated_Articles'])
189 print("The Data Frame has been created")
190
191 # Turn the empty lists into empty strings
192 df['Subarticle(s)'] = df['Subarticle(s)'].apply(lambda y: "" if len(y) == 0 else y)
193 df['Associated Articles'] = df['Associated Articles'].apply(lambda y: "" if len(y) == 0 else y)
194
195 print(
196     "-----")

```

```
198 # Analysis of the dataframe "cf"
199
200 # We create a new column with time delta
201 delta_months = []
202 for i in cf.index:
203     months = ((cf["End_Date"][i] - cf["Start_Date"][i]).days) / 30.437
204     delta_months.append(months)
205 cf = cf.assign(Delta_months=delta_months)
206
207 # We create a new column with conclusions (violation or no violation)
208 violations = []
209 for i in cf.index:
210     if "no violation" in cf["Conclusions 2"][i].lower():
211         ccl = "No"
212     else:
213         ccl = "Yes"
214     violations.append(ccl)
215 cf = cf.assign(Violation=violations)
216
217 # We calculate and print some data on the distribution of judgments (based on cf)
218
219 Mean_cf_months = cf["Delta_months"].mean().round()
220 Median_cf_months = cf["Delta_months"].median().round()
221 Min_cf_months = cf["Delta_months"].min().round()
222 Max_cf_months = cf["Delta_months"].max().round()
223
224 print("\n")
225 print("Length of proceedings")
226 print("Average = " + str(Mean_cf_months) + " months (" + str((Mean_cf_months / 12).round(1)) + " years)")
227 print("Median = " + str(Median_cf_months) + " months (" + str((Median_cf_months / 12).round(1)) + " years)")
228 print("Minimum = " + str(Min_cf_months) + " months")
229 print("Maximum = " + str(Max_cf_months) + " months (" + str((Max_cf_months / 12).round(1)) + " years)")
230 print("\n")
```

```

232 # We create a boxplot with the length of proceeding depending on conclusions
233
234 data_1 = cf[cf.Violation == "Yes"]["Delta_months"]
235 data_2 = cf[cf.Violation == "No"]["Delta_months"]
236 data = [data_1, data_2]
237 fig = plt.figure(figsize=(10, 7))
238 ax = fig.add_subplot(111)
239 bp = ax.boxplot(data, patch_artist=True,
240                 notch='True', vert=0)
241 colors = ['#0000FF', '#00FF00']
242 for patch, color in zip(bp['boxes'], colors):
243     patch.set_facecolor(color)
244 for whisker in bp['whiskers']:
245     whisker.set(color='#8B008B',
246                 linewidth=1.5,
247                 linestyle=":")
248 for cap in bp['caps']:
249     cap.set(color='#8B008B',
250            linewidth=2)
251 for median in bp['medians']:
252     median.set(color='red',
253               linewidth=3)
254 for flier in bp['fliers']:
255     flier.set(marker='D',
256              color='#e7298a',
257              alpha=0.5)
258 ax.set_yticklabels(['Violation', 'No violation'])
259 plt.title("Length of proceeding depending on conclusions")
260 ax.get_xaxis().tick_bottom()
261 ax.get_yaxis().tick_left()
262 plt.xlabel('Months')
263 plt.ylabel('Conclusions')
264 plt.show()

```

```

266 # We create a boxplot with the length of proceeding depending on importance
267
268 data_1 = cf[cf.Importance == "1"]["Delta_months"]
269 data_2 = cf[cf.Importance == "2"]["Delta_months"]
270 data_3 = cf[cf.Importance == "3"]["Delta_months"]
271 data_4 = cf[cf.Importance == "4"]["Delta_months"]
272 data = [data_1, data_2, data_3, data_4]
273 fig = plt.figure(figsize=(10, 7))
274 ax = fig.add_subplot(111)
275 bp = ax.boxplot(data, patch_artist=True,
276                 notch='True', vert=0)
277 colors = ['#0000FF', '#00FF00',
278           '#FFFF00', '#FF00FF']
279 for patch, color in zip(bp['boxes'], colors):
280     patch.set_facecolor(color)
281 for whisker in bp['whiskers']:
282     whisker.set(color='#8B008B',
283                 linewidth=1.5,
284                 linestyle=":")
285 for cap in bp['caps']:
286     cap.set(color='#8B008B',
287            linewidth=2)
288 for median in bp['medians']:
289     median.set(color='red',
290               linewidth=3)
291 for flier in bp['fliers']:
292     flier.set(marker='D',
293              color='#e7298a',
294              alpha=0.5)
295 ax.set_yticklabels(['Importance 1', 'Importance 2',
296                    'Importance 3', 'Importance 4'])
297 plt.title("Length of proceeding depending on importance")
298 ax.get_xaxis().tick_bottom()
299 ax.get_yaxis().tick_left()
300 plt.xlabel('Months')
301 plt.ylabel('Importance')
302 plt.show()

```

```

304 # We create a boxplot with the length of proceeding depending on importance Start Date (before/after 2015)
305
306 data_1 = cf[cf.Start_Date < '2015-01-01']["Delta_months"]
307 data_2 = cf[cf.Start_Date >= '2015-01-01']["Delta_months"]
308 data = [data_1, data_2]
309 fig = plt.figure(figsize=(10, 7))
310 ax = fig.add_subplot(111)
311 bp = ax.boxplot(data, patch_artist=True,
312                 notch='True', vert=0)
313 colors = ['#FFFF00', '#FF00FF']
314 for patch, color in zip(bp['boxes'], colors):
315     patch.set_facecolor(color)
316 for whisker in bp['whiskers']:
317     whisker.set(color='#8B008B',
318                 linewidth=1.5,
319                 linestyle=":")
320 for cap in bp['caps']:
321     cap.set(color='#8B008B',
322             linewidth=2)
323 for median in bp['medians']:
324     median.set(color='red',
325                linewidth=3)
326 for flier in bp['fliers']:
327     flier.set(marker='D',
328               color='#e7298a',
329               alpha=0.5)

```

```

330 ax.set_yticklabels(['Before 2015', 'After 2015'])
331 plt.title("Length of proceeding depending on the start date")
332 ax.get_xaxis().tick_bottom()
333 ax.get_yaxis().tick_left()
334 plt.xlabel('Months')
335 plt.ylabel('Start date')
336 plt.show()

```



```

338 # Analysis of the dataframe "df"
339
340 # We create a new column with time delta
341 delta_months = []
342 for i in df.index:
343     months = ((df["End_Date"][i] - df["Start_Date"][i]).days) / 30.437
344     delta_months.append(months)
345 df = df.assign(Delta_months=delta_months)

```

```

347 # We create a dictionary of articles with number and name ("subject")
348
349 dictionary = {}
350 for i in df.Article:
351     # We exclude articles from protocols
352     if "p" in i:
353         pass
354     else:
355         a = int(i)

```

```

356 if a == 2:
357     dictionary[a] = "Right to life"
358 elif a == 3:
359     dictionary[a] = "Prohibition of torture"
360 elif a == 4:
361     dictionary[a] = "Prohibition of slavery and forced labour"
362 elif a == 5:
363     dictionary[a] = "Right to liberty and security"
364 elif a == 6:
365     dictionary[a] = "Right to a fair trial"
366 elif a == 7:
367     dictionary[a] = "No punishment without law"
368 elif a == 8:
369     dictionary[a] = "Right to respect for private and family life"
370 elif a == 9:
371     dictionary[a] = "Freedom of thought, conscience and religion"

```

```

372 elif a == 10:
373     dictionary[a] = "Freedom of expression"
374 elif a == 11:
375     dictionary[a] = "Freedom of assembly and association"
376 elif a == 12:
377     dictionary[a] = "Right to marry"
378 elif a == 13:
379     dictionary[a] = "Right to an effective remedy"
380 elif a == 14:
381     dictionary[a] = "Prohibition of discrimination"
382 elif a == 15:
383     dictionary[a] = "Derogation in time of emergency"
384 elif a == 16:
385     dictionary[a] = "Restrictions on political activity of aliens"
386 elif a == 17:
387     dictionary[a] = "Prohibition of abuse of rights"
388 elif a == 18:
389     dictionary[a] = "Limitation on use of restrictions on rights"

```

```

390 from collections import OrderedDict
391
392 dictionary = OrderedDict(sorted(dictionary.items()))
393
394 # We add a column with the name of each article in df
395 new_column = []
396 for i in df.index:
397     if "p" in df["Article"][i]:
398         new_column.append("Not relevant")
399     else:
400         a = int(df["Article"][i])
401         if a in dictionary.keys():
402             new_column.append(dictionary[a])
403         else:
404             new_column.append("Not relevant")
405 df = df.assign(Subject=new_column)

```



```

394 # We add a column with the name of each article in df
395 new_column = []
396 for i in df.index:
397     if "p" in df["Article"][i]:
398         new_column.append("Not relevant")
399     else:
400         a = int(df["Article"][i])
401         if a in dictionary.keys():
402             new_column.append(dictionary[a])
403         else:
404             new_column.append("Not relevant")
405 df = df.assign(Subject=new_column)
406
407 # We calculate the mean and median of Delta_months, the number of judgments and the number of associated articles for each relevant article
408 data_articles = {}
409 for i in dictionary.keys():
410     new_df = df[(df.Article == str(i))]
411     new_len = len(new_df)
412     new_average = new_df["Delta_months"].mean().round()
413     new_median = new_df["Delta_months"].median().round()
414     new_average_nb = new_df["Number_of_Associated_Articles"].mean().round()
415     new_list = [i, dictionary[i], new_len, new_average, new_median, new_average_nb]
416     data_articles[i] = new_list
417 df_articles = pd.DataFrame.transpose(pd.DataFrame(data_articles))
418 df_articles.columns = ["Article", "Subject", "Number_Judgments", "Average_Months", "Median_Months",
419                       "Average_Number_Associated_Articles"]
420
421 # We add the number of judgments in a new column in df
422 new_column = []
423 df_articles2 = df_articles.set_index("Subject")
424 for i in df.Subject:
425     if i == "Not relevant":
426         new_column.append(0)
427     else:
428         new = df_articles2["Number_Judgments"][i]
429         new_column.append(int(new))
430 df = df.assign(number_judgments=new_column)

```

```
432 # We create scatterplots (based on medians in df) and we save the files
433
434 sns.scatterplot(data=df_articles, x="Median_Months", y="Number_Judgments", hue="Subject")
435
436 plt.savefig('Scatterplot_all.png')
437 plt.close()
438
439 sns.scatterplot(data=df_articles[df_articles.Number_Judgments > 100], x="Median_Months", y="Number_Judgments",
440                 hue="Subject")
441
442 plt.savefig('Scatterplot_morethan100.png')
443 plt.close()
444
445 sns.scatterplot(data=df_articles[df_articles.Number_Judgments < 100], x="Median_Months", y="Number_Judgments",
446                 hue="Subject")
447
448 plt.savefig('Scatterplot_lessthan100.png')
449 plt.close()
450
451 # We create distribution plots (based on df)
452
453 sns.displot(data=df[df.Subject != "Not relevant"], x="Delta_months", hue="Subject", multiple="stack")
454
455 sns.displot(data=df[df.Subject != "Not relevant"][df.number_judgments < 100], x="Delta_months", hue="Subject",
456             multiple="stack")
457
458 sns.displot(data=df[df.Subject != "Not relevant"][df.Delta_months > 150], x="Delta_months", hue="Subject",
459             multiple="stack")
460
461 print("-----")
```

The Results

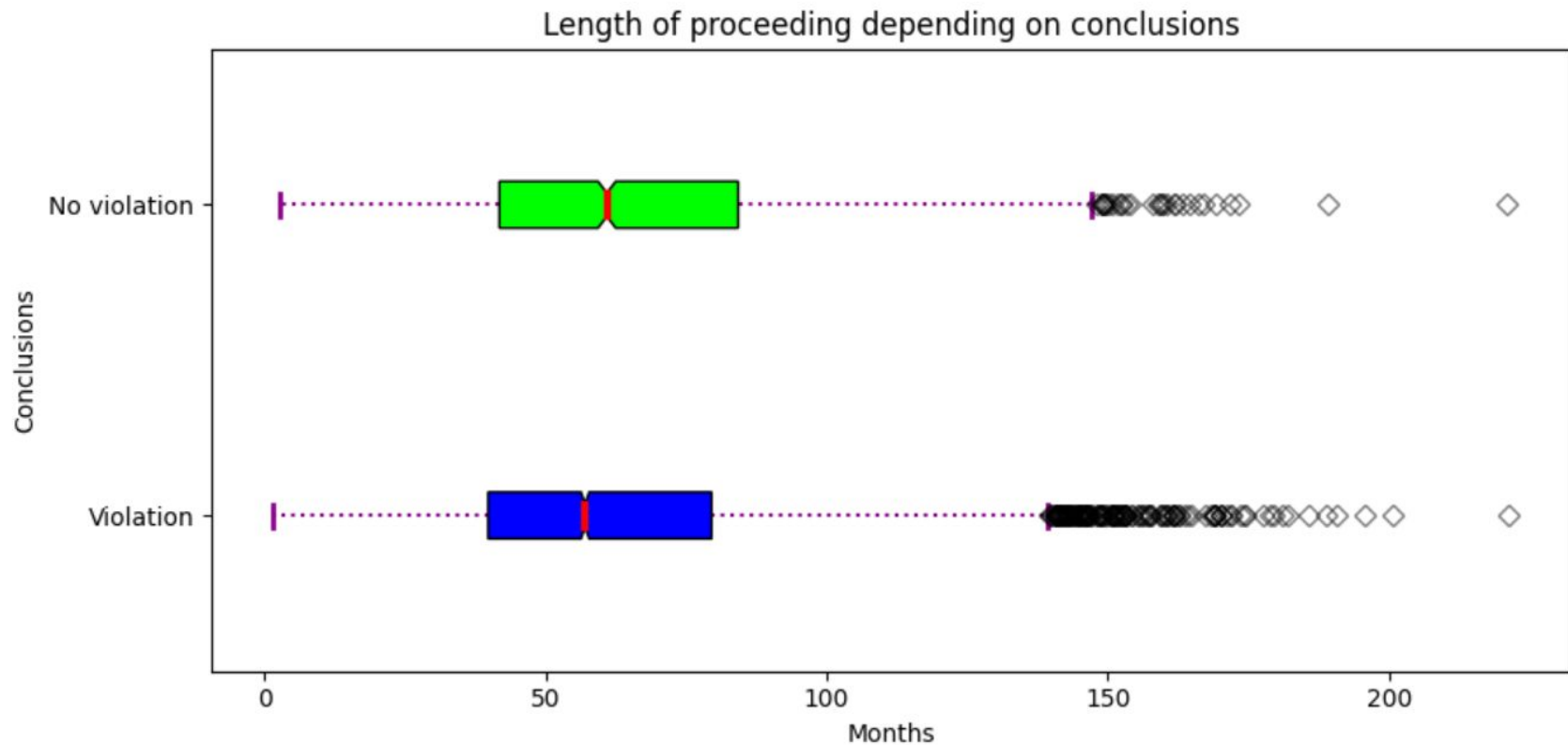
Length of proceedings

Average = 63.0 months (5.2 years)

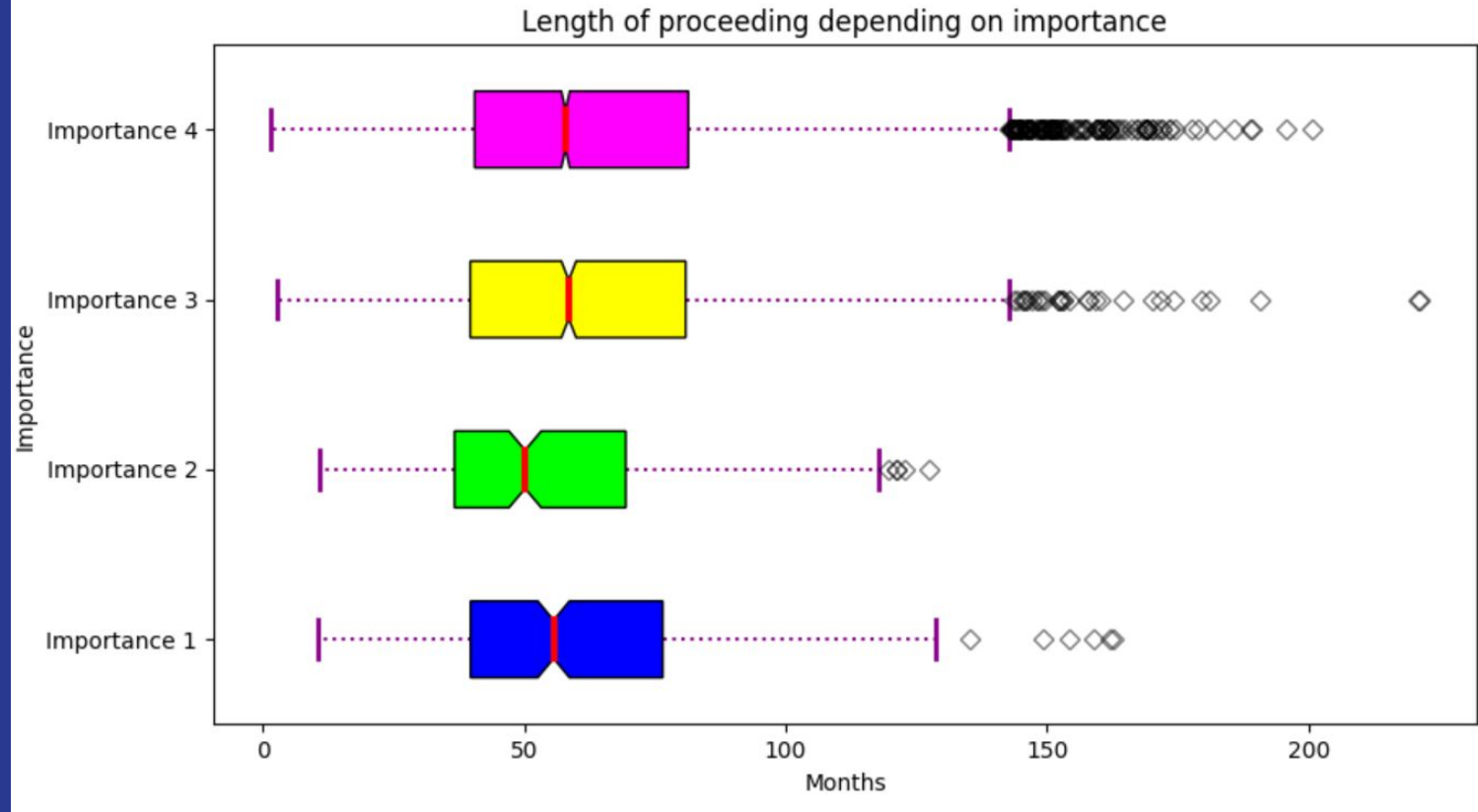
Median = 58.0 months (4.8 years)

Minimum = 2.0 months

Maximum = 221.0 months (18.4 years)

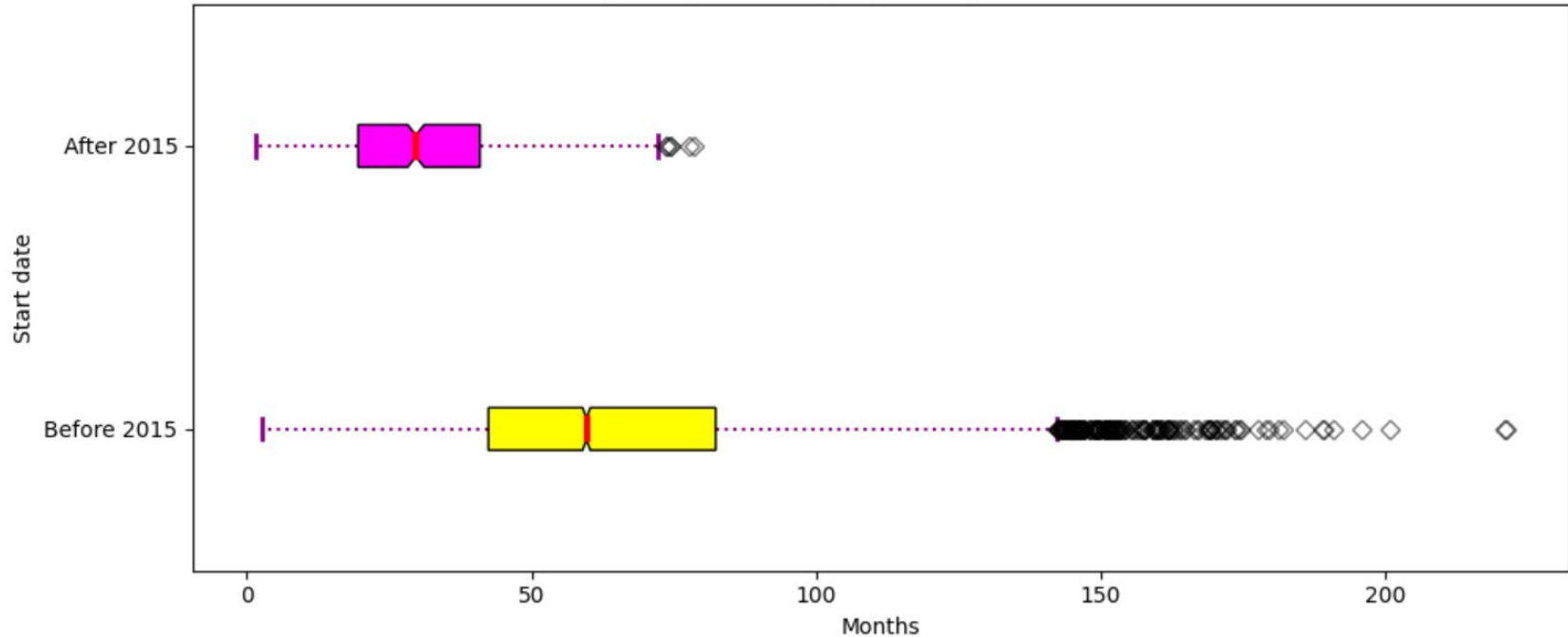


The ECHR responds slightly more quickly when there is a violation.



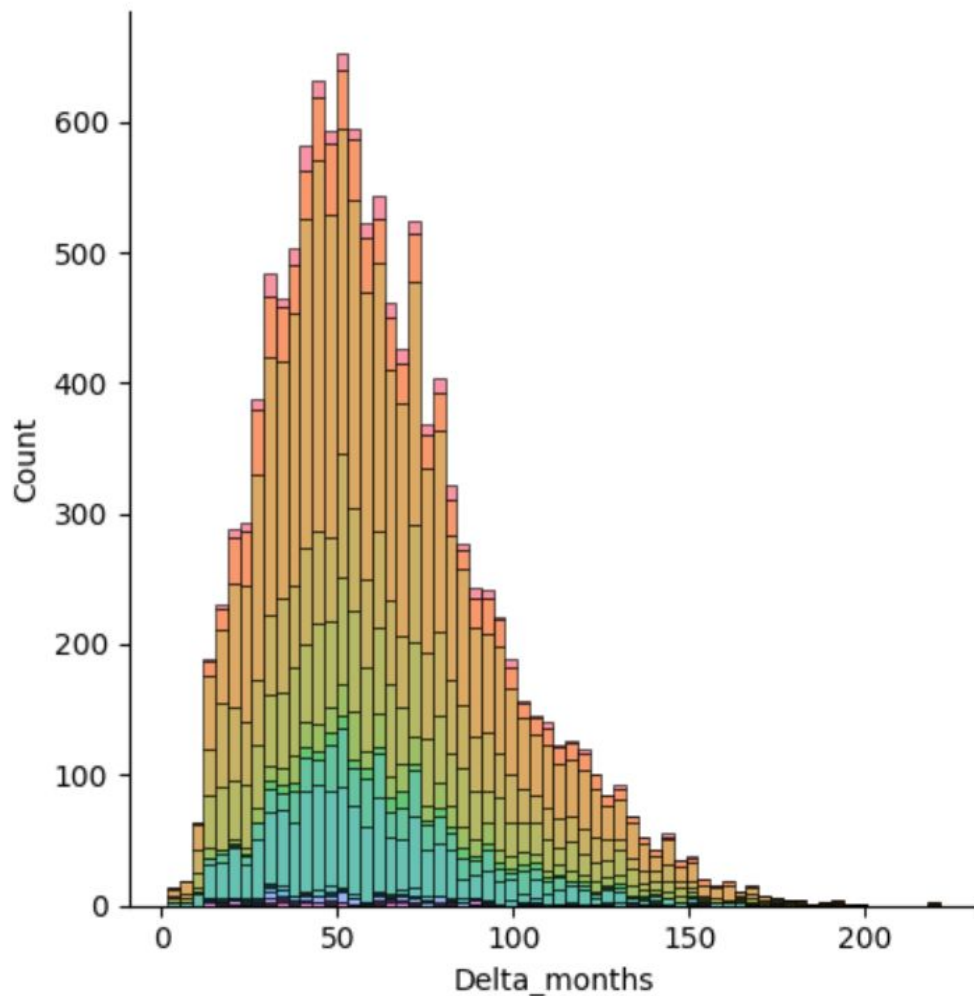
The ECHR responds slightly more quickly for cases of higher importance.

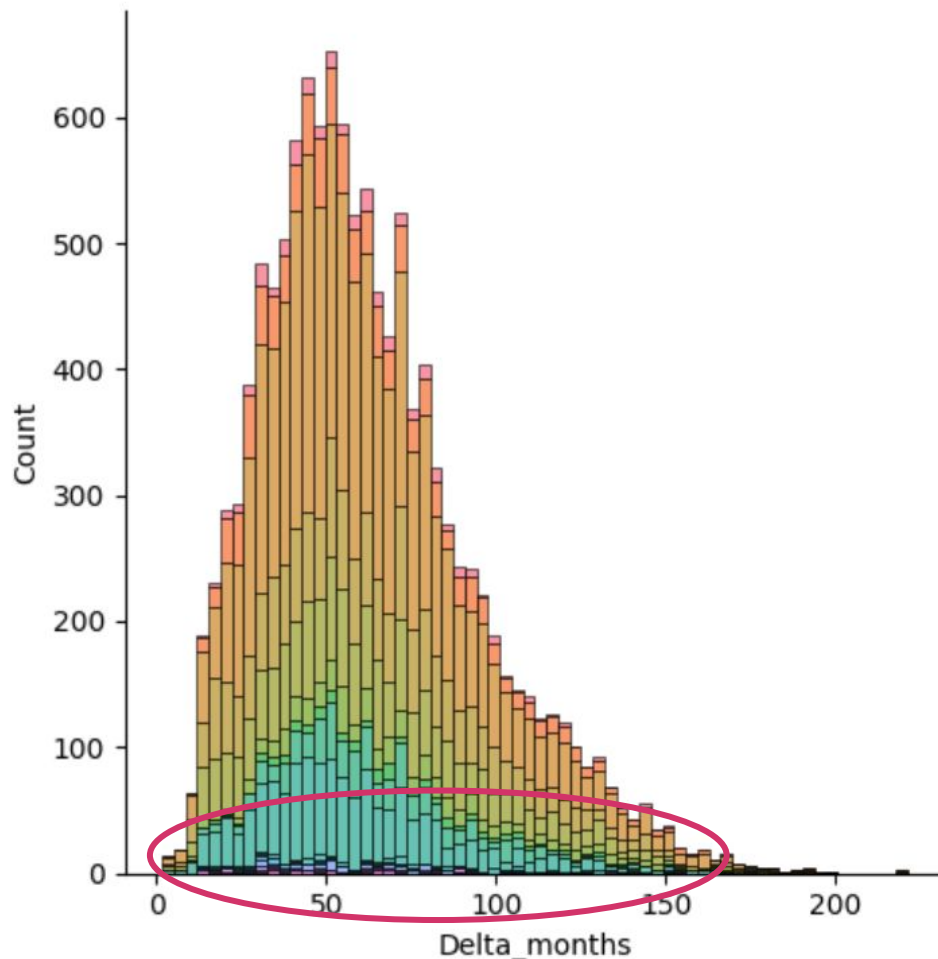
Length of proceeding depending on the start date

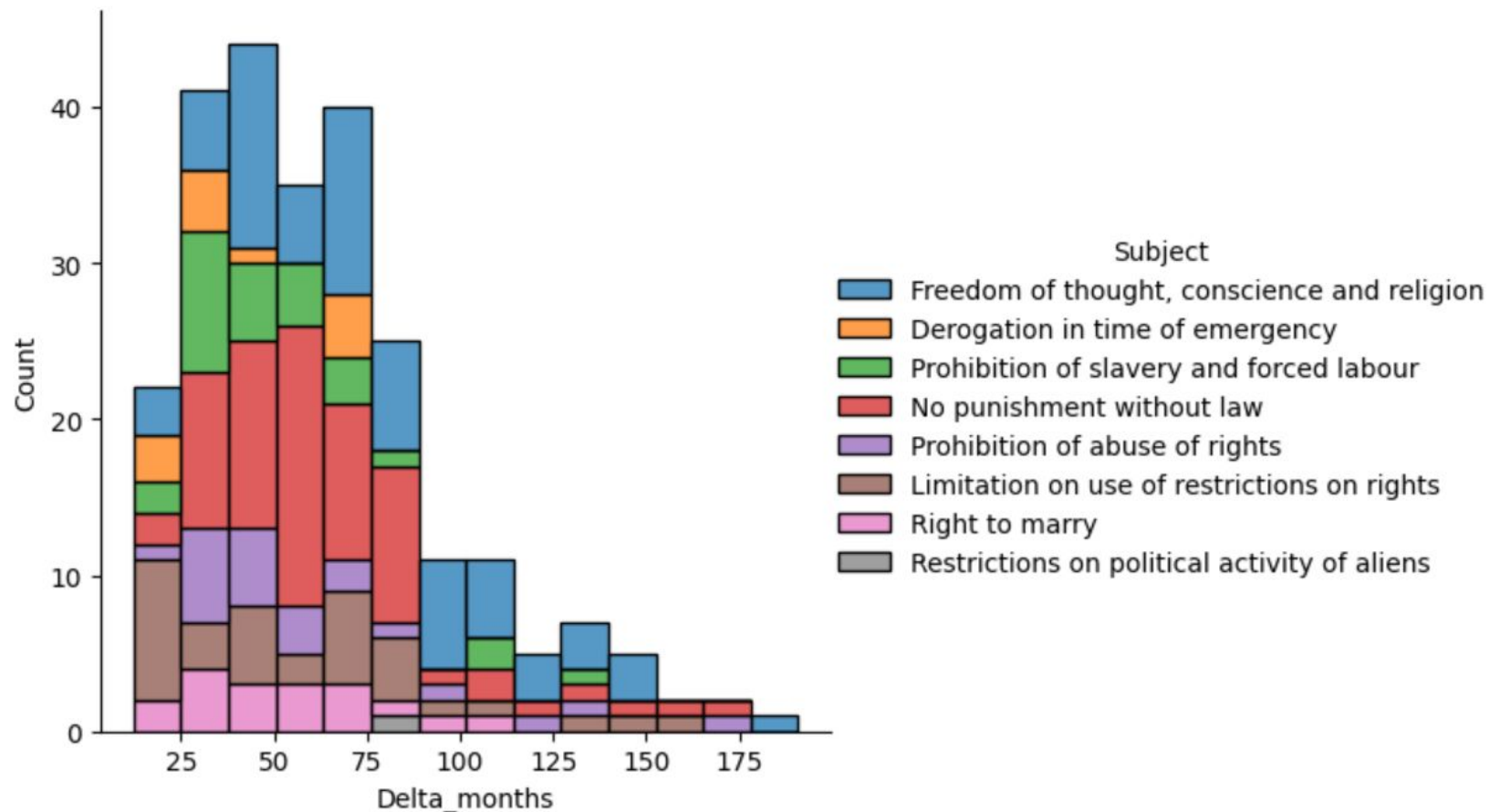


A massive increase in the number of requests in 2011 led the ECHR to change its method : proceedings are faster after 2015. (<https://www.rue89strasbourg.com/parcours-requetes-cour-europeenne-des-droits-homme-125332>)

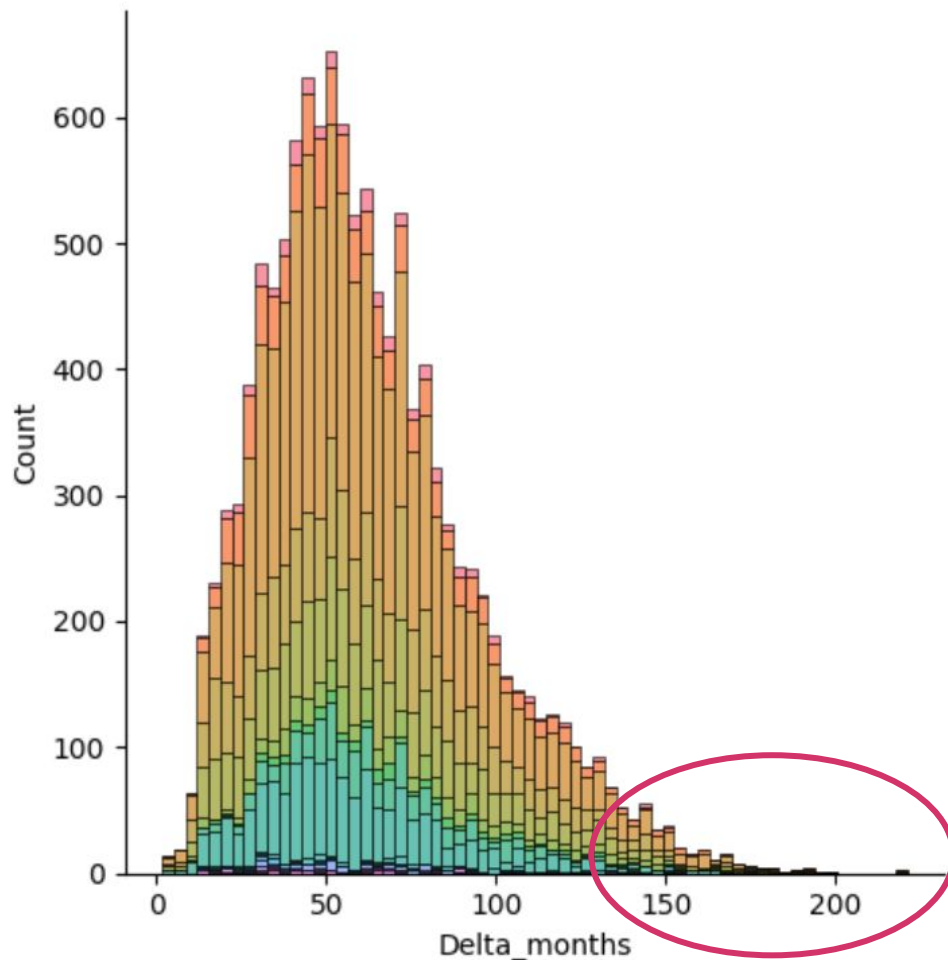
(N.B: The procedure has a maximum duration of 84 months if the complaint was filed in 2015)



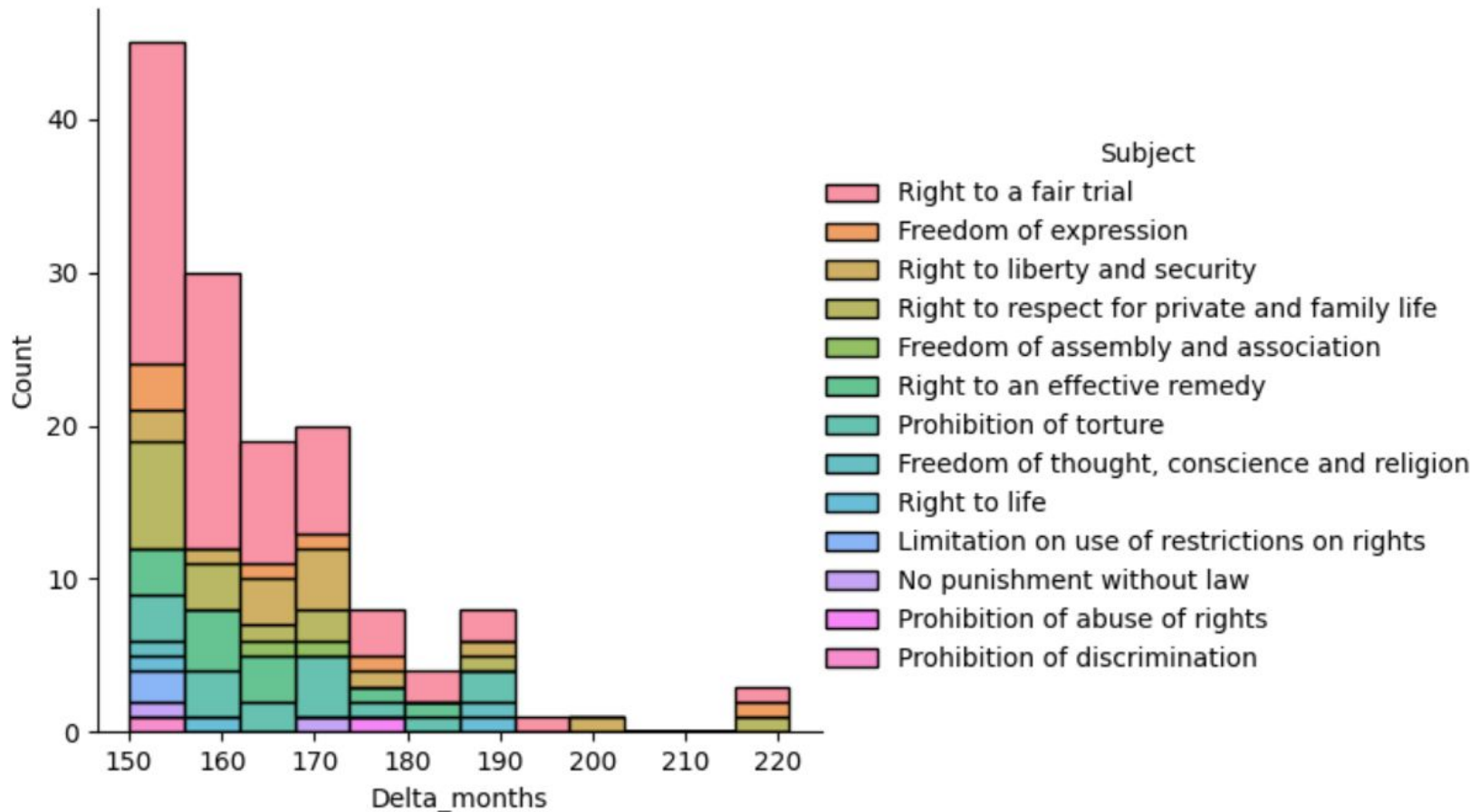




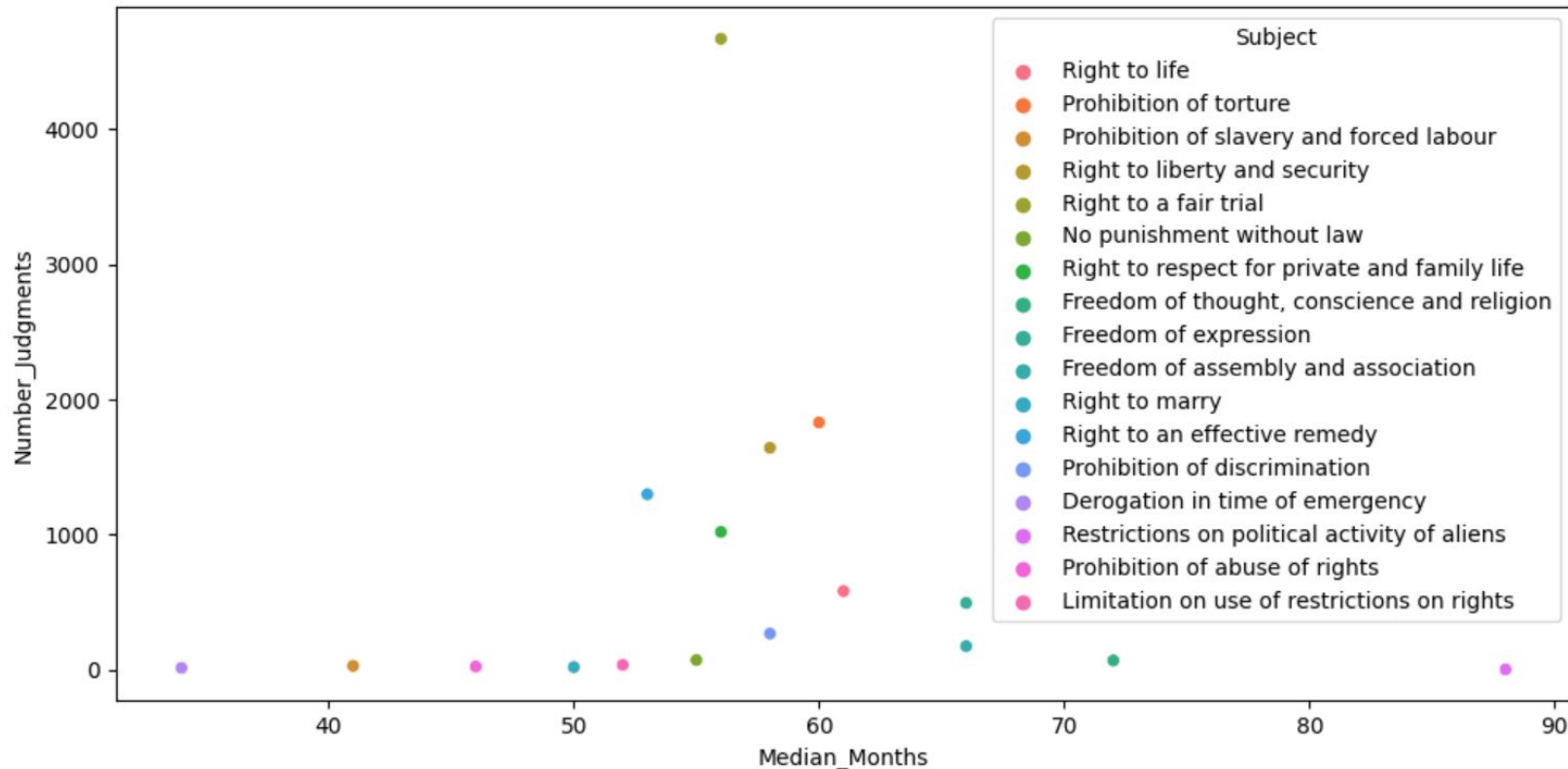
Faster proceedings: derogation in time of emergency, prohibition of slavery, right to marry



- 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

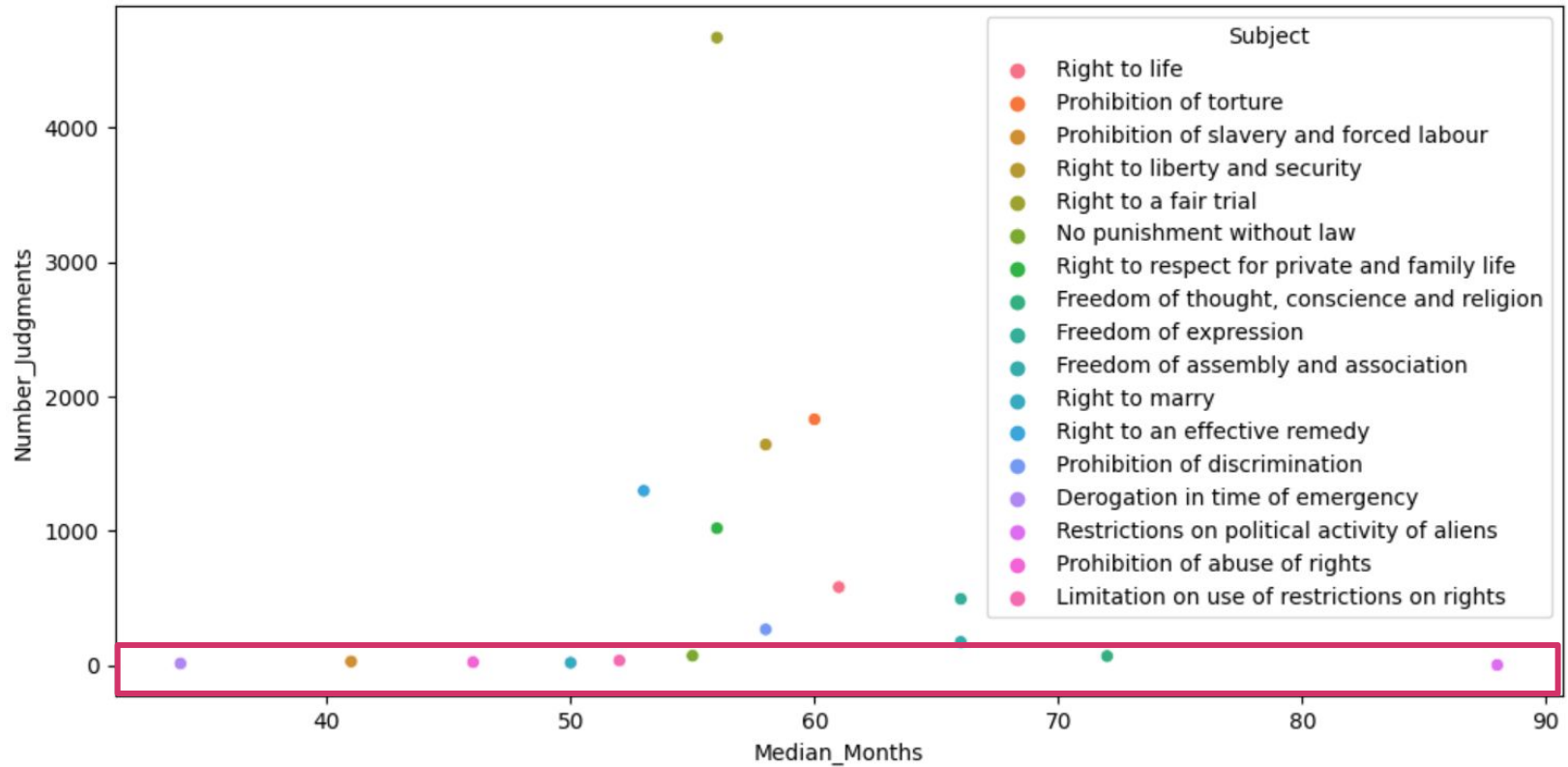


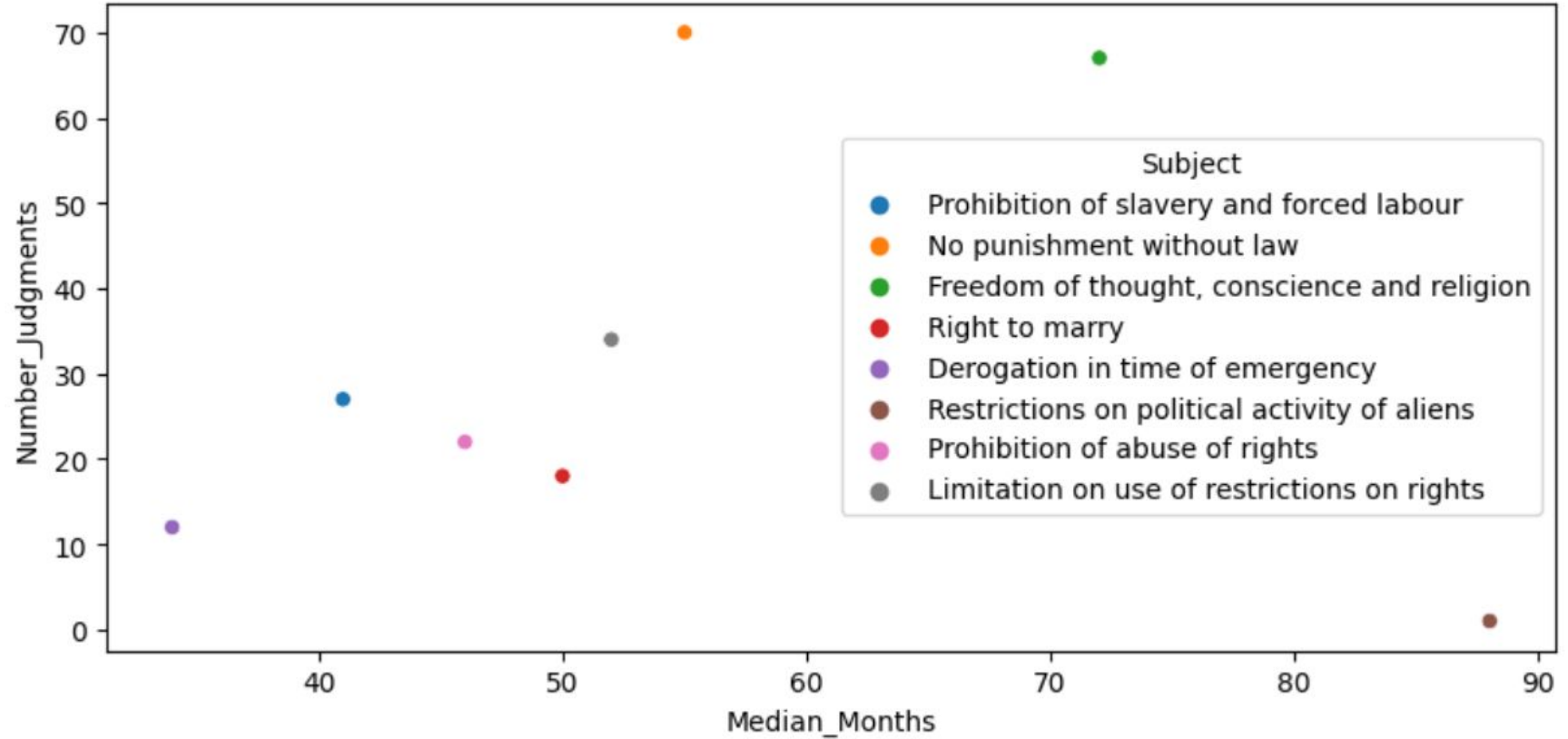
Two 18 years proceedings : article 6 (right to a fair trial) ; articles 8 (right to respect for private life) and 10 (freedom of expression)

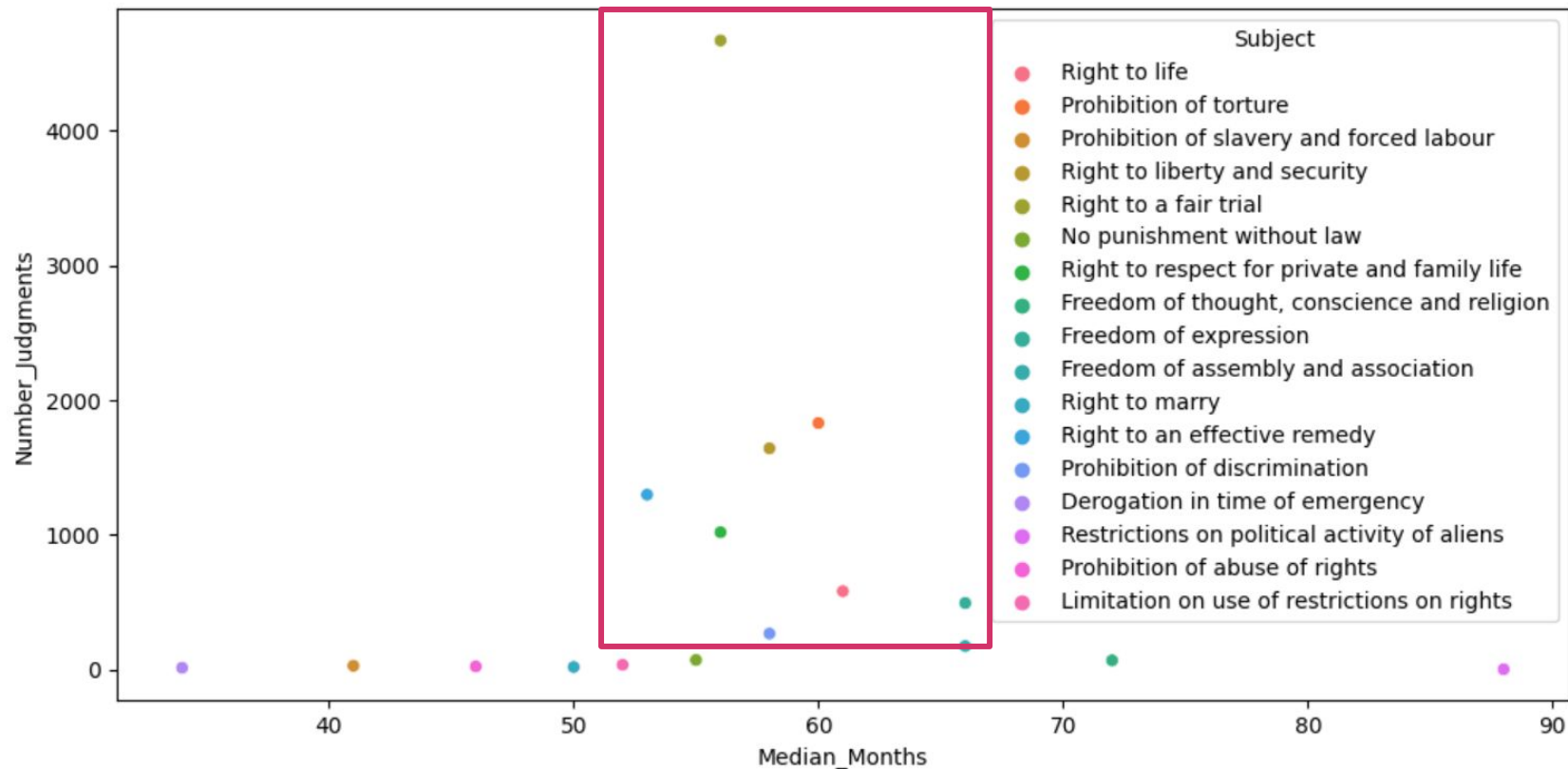


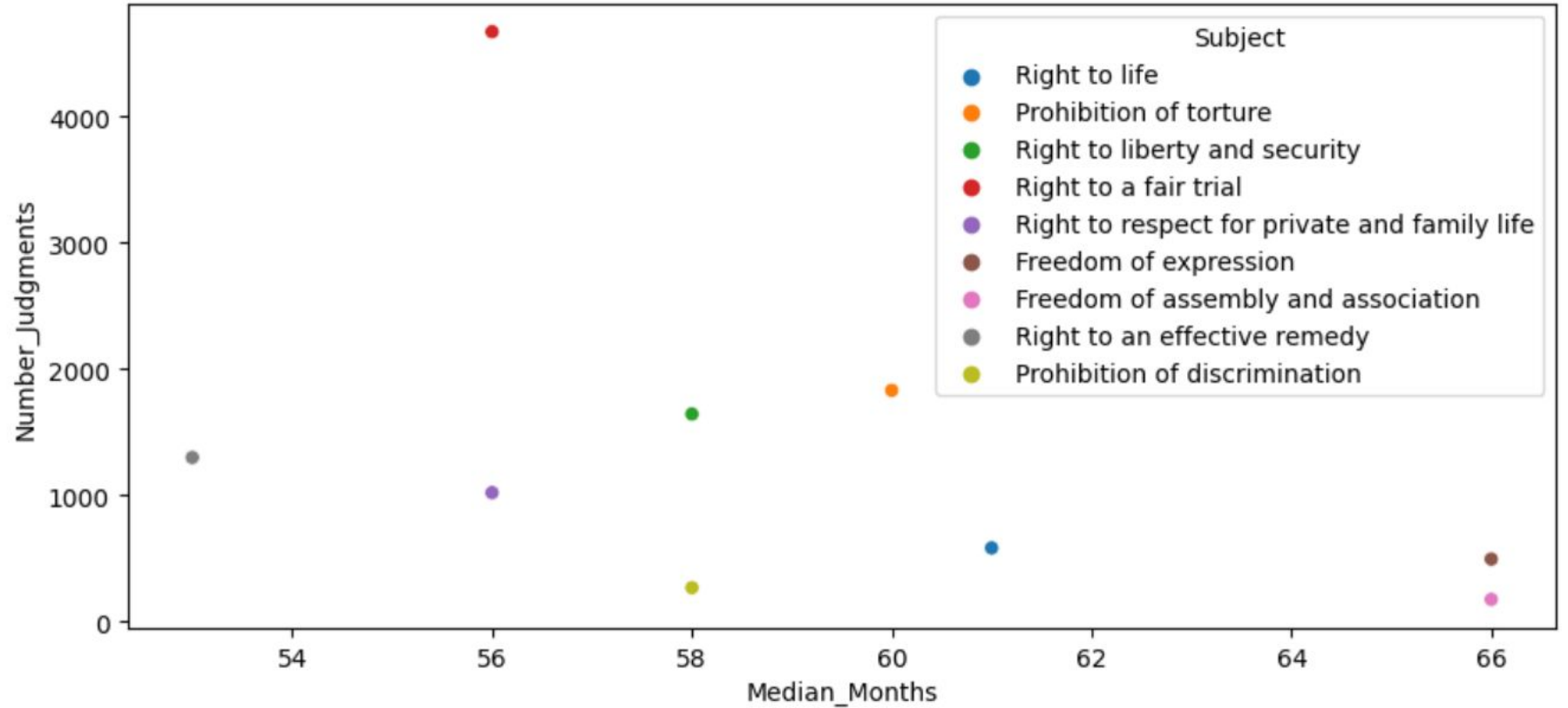
Faster proceedings : **derogation in time of emergency** (34), **prohibition of slavery** (41), **prohibition of abuse of rights** (46)

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









(Analysis)

The background is a solid pink color. In the top right corner, there is a decorative pattern of overlapping squares and triangles in various shades of pink and magenta, creating a geometric, stepped effect.

Thank you