Tutorial #2: Web Scraping membership data from the IEA Database (Number of members)

Alice Soldà

March 1, 2021

General Introduction

The IEA Database[1] currently contains over 1,300 MEAs, over 2,200 BEAs, 250 other environmental agreements, and over 90,000 individual country "membership actions" that are constantly being revised and updated.

Retrieving membership data manually can be time consuming. Luckily for us, the database is designed in a way that allows us to automate this task using a simple Python code.

Prerequisite

For this tutorial, you need some basic knowledge on how to use a Python IDLE to write Python scripts and how to run these scripts in the Python console.

You will need to download and install Python (this tutorial was made using Python 3.7.3) as well as the pandas library. If you need help installing packages with python, have a look here.

This tutorial can be undertaken separately from the other tutorials in the series.

[1](c) Ronald B. Mitchell and the IEA Database Project, 2002-2020

1 Retrieving the relevant IDs

The first thing we need to do is to identify the IDs of our treaties of interest. Each treaty is given an ID (called Mitch ID) that links it to various sets of information (metadata, membership data, etc.).

Identifying treaties IDs can be easily done by downloading the list of agreements in CSV format from the IEADB Agreement List and using filters on excel. You can already use some filters directly on the IEADB website. The treaties IDs are stored in the first column of the CSV file.

A1 | IEA# (click for add'l info)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	IEA# (click fo	Inclusion	Signature Yea	Signature Dat	Date IEA ente	Treaty Name	Treaty Text	Members	Agreement T	Lineage	Sequence in	Data	Articles, etc.	Secretariat	UNEP Info		
2	4964	MEA	2010	01/01/2010	01/01/2010	Cooperation	Treaty Text**	Members	Agreement	Te Vaka	100.000	Data	Lit	Te Vaka Moana			
3	4972	MEA	2010	02/02/2010	14/02/2017	Agreement o	Treaty Text**	Members	Agreement	Prespa Park	100.000	Data	Lit	Prespa Park Secretariat			
4	4614	MEA	2010	25/03/2010	23/06/2010	Amendment:	Treaty Text**	Members	Amendment	CITES	100.180	Data	Lit	CITES Secretariat			
5	4615	MEA	2010	26/03/2010	01/08/2011	Amendment:	Treaty Text**	Members	Amendment	MARPOL	204.040	Data	Lit	International	UNEP Info		
6	4695	MEA	2010	26/03/2010	01/08/2011	Amendment:	Treaty Text**	Members	Amendment	MARPOL	203.540	Data	Lit	International Maritime Organization (IMO)			
7	4612	MEA	2010	31/03/2010	01/01/2599	Protocol for t	Treaty Text**	Members	Protocol	Eastern Afric	103.000	Data	Lit	UNEP Region	UNEP Info		
8	4613	MEA	2010	31/03/2010	01/01/2599	Amended Co	Treaty Text**	Members	Agreement	Eastern Afric	200.000	Data	Lit	UNEP Regional Coordinating Unit for the Eastern Afric			
9	4611	MEA	2010	30/04/2010	01/01/2599	Protocol to th	Treaty Text**	Members	Protocol	Liability Com	101.000	Data	Lit	International	UNEP Info		
10	4957	MEA	2010	14/05/2010	01/01/2599	Agreement o	Treaty Text**	Members	Agreement	Nile River Ba	1000.000	Data	Lit	Nile Basin Initiative			
11	8025	MEA	2010	21/05/2010	19/05/2011	Protocol on amendments t		Members	Amendment	Belarus-Kaza	201.000	Data	Lit	Secretariat not yet identified			
12	8026	MEA	2010	21/05/2010	19/05/2011	Protocol on amendments t		Members	Amendment	Belarus-Kaza	101.000	Data	Lit	Secretariat not yet identified			
13	4939	MEA	2010	01/06/2010	27/11/2015	Protocol On f	Treaty Text**	Members	Protocol	Sava River Ba	103.000	Data	Lit	International Sava River Basin Commission			
14	4847	MEA	2010	04/06/2010	04/06/2010	Agreement b	Treaty Text**	Members	Agreement	Lithuania-Est	200.000	Data	Lit	No secretariat established			
15	4899	MEA	2010	08/06/2010	01/01/2011	Amendment:	Treaty Text**	Members	Amendment	Rhine Polluti	1100.020	Data	Lit	Secretariat for the Convention relative Â la collecte, a			
16	4691	MEA	2010	25/06/2010	11/01/2011	Amendment:	Treaty Text**	Members	Amendment	International	300.650	Data	Lit	International Whaling Commission			
17	4924	MEA	2010	11/09/2010	01/01/2899	Amendment:	Treaty Text**	Members	Amendment	Micronesia Fi	200.020	Data	Lit	Parties to the Nauru Agreement			
18	5033	MEA	2010	11/09/2010	11/09/2010	2010 Amendr	Treaty Text**	Members	Amendment	Micronesia Fi	103.010	Data	Lit	Parties to the Nauru Agreement			
19	4637	MEA	2010	01/10/2010	01/01/2014	Amendment:	Treaty Text**	Members	Amendment	MARPOL	203.550	Data	Lit	International Maritime Organization (IMO)			
20	4705	MEA	2010	01/10/2010	01/02/2012	Amendment:	Treaty Text**	Members	Amendment	MARPOL	204.050	Data	Lit	International	UNEP Info		
21	4636	MEA	2010	15/10/2010	05/03/2018	Nagoya-Kuala	Treaty Text**	Members	Protocol	Biological Div	102.000	Data	Lit	Secretariat fo	UNEP Info		
22	8024	MEA	2010	21/10/2010	23/05/2011	Agreement b	Treaty Text**	Members	Agreement	Not Yet Assigned				Secretariat not yet identified			
23	4638	MEA	2010	29/10/2010	12/10/2014	Nagoya Proto	Treaty Text**	Members	Protocol	Biological Div	103.000	Data	Lit	Secretariat fo	UNEP Info		
24	5081	MEA	2010	12/11/2010	01/01/2599	Amendment	Treaty Text**	Members	Amendment	Conservation	600.020	Data	Lit	ACCOBAMS Secretariat			
25	8511	MEA	2010	12/11/2010	12/04/2011	Amendment	Treaty Text**	Members	Amendment	Conservation	600.030	Data	Lit	ACCOBAMS Secretariat			
26	4898	MEA	2011	28/01/2011	01/01/2599	Amendment	Treaty Text**	Members	Amendment	Association c	100.010	Data	Lit	Association of Caribbean States			
27	4966	MEA	2011	29/04/2011	27/10/2012	Amendment:	Treaty Text**	Members	Amendment	Persistent Or	100.030	Data	Lit	UNEP Chemi	UNEP Info		
28	8023	MEA	2011	19/05/2011	26/12/2011	Protocol on amendments t		Members	Amendment	CIS Environmental Agreem		Data	Lit	Secretariat not yet identified			
29	4872	MEA	2011	27/05/2011	21/10/2013	Protocol on S	Treaty Text**	Members	Protocol	Carpathian Sk	102.000	Data	Lit	Interim Secretariat of the Framework Convention on t			

2 Retrieving the number of members

2.1 Inspecting the website

The second thing we need to do is to figure out where membership data is stored on the IEADB website. If you go on the IEADB Agreement List, you can see a column called "Members" with a link to the membership data for this particular treaty. Let's open this link for the Paris Agreement (ID 5046). The membership data page contains a table in wide format with the dates as column heads and the countries as rows as in the screenshot below.

Wide 5046

Wide | Long | Action

Membership Information for: Paris Agreement under the United Nations Framework Convention on Climate Change

Signed: 2015-12-12; Entered into force: 2016-11-04; Terminated:

Source for membership data: https://treaties.un.org/pages/ViewDetails.aspx

Last downloaded: 2016-10-15; Last known update: 2016-11-04

The following is data currently in the IEA database for agreement actions. Agreement actions (signature, ratification, entry into force) can change frequently. The IEA Database Project updates this data approximately once a year and tries to ensure it is current and accurate. *You are encouraged to visit the Secretariat of the UN Convention of Climate Change website to look for the most accurate and up-to-date membership information.* If you have questions about this data or notice inaccuracies, please contact the Project Director.

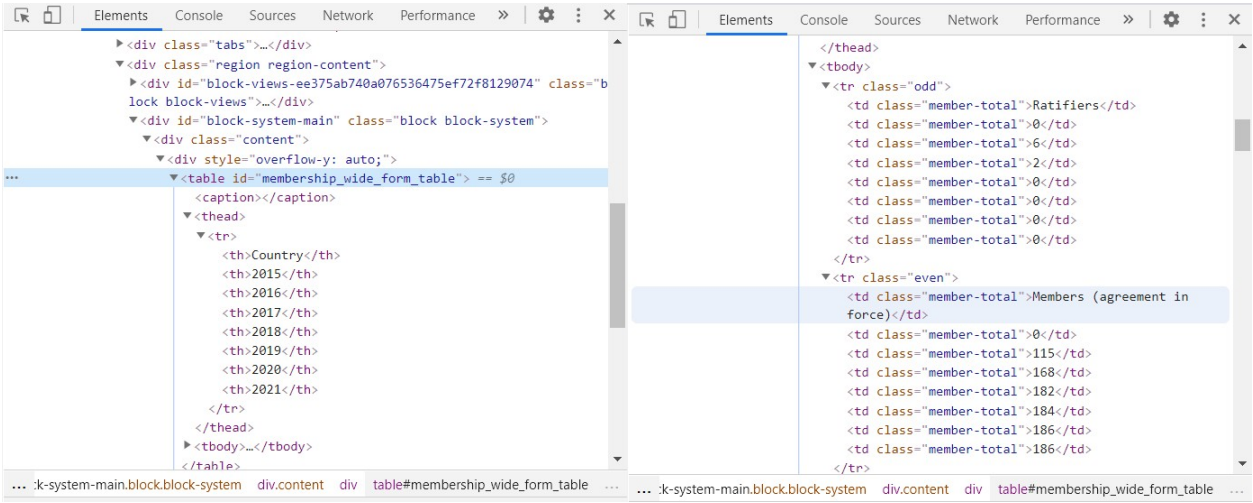
Country	2015	2016	2017	2018	2019	2020	2021
Ratifiers	0	6	2	0	0	0	0
Members (agreement in force)	0	115	168	182	184	186	186
Afghanistan	0	0	2	2	2	2	2
Albania	0	2	2	2	2	2	2
Algeria	0	2	2	2	2	2	2
Andorra	0	0	2	2	2	2	2
Angola	0	0	0	0	0	0	0
Antigua and Barbuda	0	2	2	2	2	2	2
Argentina	0	2	2	2	2	2	2
Armenia	0	0	2	2	2	2	2
Australia	0	2	2	2	2	2	2
Austria	0	2	2	2	2	2	2
Azerbaijan	0	0	2	2	2	2	2

The data of interest to us is the number of members for this specific treaty for each year the treaty has been open and can be found in the red boxes in the screenshot below:

Country	2015	2016	2017	2018	2019	2020	2021
Ratifiers	0	6	2	0	0	0	0
Members (agreement in force)	0	115	168	182	184	186	186
Afghanistan	0	0	2	2	2	2	2
Albania	0	2	2	2	2	2	2

Now we need to figure out the HTML tags associated with these two rows so we can retrieve the data that they contain with our python program. Simply put, there is a lot of code on a website page and we want to find the relevant pieces of code that contains our data. You can learn more about HTML tags here.

There are several tags that are relevant for our purpose. If you want to know more about how to identify relevant HTML tags, you can check out Julia Kho's tutorial on web scraping with Python here. First, we can see that the table has a specific id `membership_wide_form_table`. Second, the dates are within the tags `<thead></thead>`. In addition, the number of members are associated with the class `"member_total"`. However, the row containing the ratifiers data is also associated with this class. Luckily for us, our data of interest is also associated with the class `"even"`, while the ratifiers are associated with the class `"odd"`.



Now that we identified our set of tags, look at the URL of the membership data page. you will notice that it is of the the form https://iea.uoregon.edu/members/[treaty ID]. This will be helpful to code our Python program.

2.2 Python Code

Open your Python IDLE and start a new Python file. let's start by importing the relevant libraries.

```
import requests
from bs4 import BeautifulSoup
import pandas
```

We use the requests library to tell our python program where to look for our data, the Beautiful Soup library for pulling information from web pages, and the pandas libray to transform our data to an exploitable dataset.

Next, we create a list that contains our treaties IDs. However, we cannot just copy-paste the first column from our CSV file into our python code because each element of a Python list needs to be separated by a comma. Luckily, we can easily do this in excel with the function CONCAT. If you need help with this step, check here. We can now copy-paste our concatenated IDs into our list of treaties. For instance:

```
list_treaty = [
3059,
8501,
3100,
5006]
```

Let's say we want to scrap the membership data for the first treaty in the list. We need to set the URL to membership data page, and access the page with our requests library.

```
url_string='https://iea.uoregon.edu/members/'+str(list_treaty[0])
website_url=requests.get(url_string).text
```

As mentioned earlier, the URL of the pages containing the membership data are all of the form: https://iea.uoregon.edu/members/[treaty ID]. Because we want the first treaty in the list, we tell python to replace [treaty ID] by `list_treaty[0]` (i.e. the first element of our list).

To check that the URL is correct, we can print the URL in the Python console.

```
print(url_string)
```

If the request is successful, you should see the following output.

```
>>> https://iea.uoregon.edu/treaty-text/3059
```

Next, we parse the HTML with BeautifulSoup so that we can work with a nicer, nested BeautifulSoup data structure.

```
soup = BeautifulSoup(website_url,'lxml')
```

We use the method `.find` to locate our text based on the HTML tag and class we identified earlier. We first locate the table.

```
my_table = soup.find(id='membership_wide_form_table')
```

To check that the code worked correctly, we can print our table:

```
print(my_table)
```

We next locate the row associated with the HTML tag `<thead>` that contains the dates, and save each cell within the HTML tag `<th>` (each corresponding to a date) into a list (`list_dates`).

```
dates = my_table.findAll('thead')

list_dates = []

for date in dates:
    cells = date.findAll('th',{'class':""})
    print(cells)
```

We can print our variable `cells` to check that it contains our dates. The output should look like this:

```
[<th>Country</th>, <th>2015</th>, <th>2016</th>, <th>2017</th>, <th>2018</th>, <
th>2019</th>, <th>2020</th>, <th>2021</th>]
```

Because the first element contains the header for the "Country" column, we will not append it to `list_dates`. We use `cells[1:]` to tell python to start appending the elements in `cells` from the second element. We use the beautifulsoup method `.text` (or `.get_text()`) to append the text without the HTML tags into our final list of dates. Finally, we use `int()` to append our dates (so far in text element) as integers. By storing the dates as numbers, excel will later recognize them as such, which will be helpful to manipulate the data. We also want to make sure that our table contains at least one date (so two columns) before adding them into our list.

```
for date in dates:
    cells = date.findAll('th',{'class':""})
        if len(cells) > 1:
            list_dates = [int(c.text) for c in cells[1:]]
            print(list_dates)
```

We can check that the code worked by printing our list of dates. The final output should look like this:

```
['2015', '2016', '2017', '2018', '2019', '2020', '2021']
```

Now, let's do the same for the numbers of members. We first collect all the rows associated with `class="even"`. Because we are only interested in the first row of this class, we look for the cells in the first element of `members` only. To do so, we first need to check that our variable `members` is not empty. If this condition is satisfied, we collect the cells associated with `class="member-total"`, each corresponding to the number of members at a specific date.

```
members = my_table.findAll('tr',{'class':"even"})

if len(members) > 0:
```

```
        cells = members[0].findAll('td',{'class':"member-total"})
        print(cells)
```

Let's see what's in `cells`:

```
[<td class="member-total">Members (agreement in force)</td>, <td class="member-t
otal">0</td>, <td class="member-total">115</td>, <td class="member-total">168</t
d>, <td class="member-total">182</td>, <td class="member-total">184</td>, <td cl
ass="member-total">186</td>, <td class="member-total">186</td>]
```

As before, we don't want to keep the first element as it is not a date. If there are at least two elements in `cells` (which is the case in our example), we use `.text` to retrieve the number of members free of the HTML tags. Because some treaties are associated with half-members (e.g. 83.5), we have to use `float()` instead of `int()` to transform the text elements into float elements (i.e. numbers with decimal points).

```
if len(members) > 0:
        cells = members[0].findAll('td',{'class':"member-total"})
        if len(cells) > 1:
            list_members = [float(c.text) for c in cells[1:]]
            print(list_members)
```

We can check that the code worked by printing the list of numbers of members. The output should look like this:

```
[0.0, 115.0, 168.0, 182.0, 184.0, 186.0, 186.0]
```

2.3 Exporting the data in .csv format using pandas

Now that we have collected and stored the data of interest to us, we need to export it in an exploitable format. To do so, we need to create two dictionaries (one for the dates and one for the number of members) using the treaty ID as the key. This will allow us to create tables that contains the treaty ID in the first column and our data in the subsequent columns.

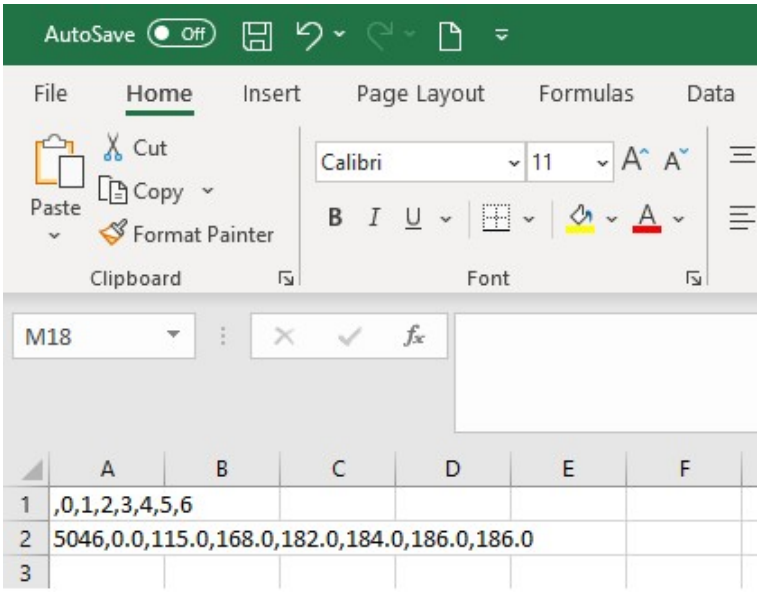
```
treaty_dates = {}
treaty_members = {}

treaty_dates[list_treaty[0]] = list_dates
treaty_members[list_treaty[0]] = list_members
```

We next transform these dictionaries into dataframes and export them in csv format somewhere on our computer.

```
pandas.DataFrame.from_dict(treaty_dates, orient='index').to_csv(r'C:\...
\dates.csv')
pandas.DataFrame.from_dict(treaty_members, orient='index').to_csv(r'C:\...
\members.csv')
```

Here is an exemple of the output for the number of members for the treaty with the ID 5046:

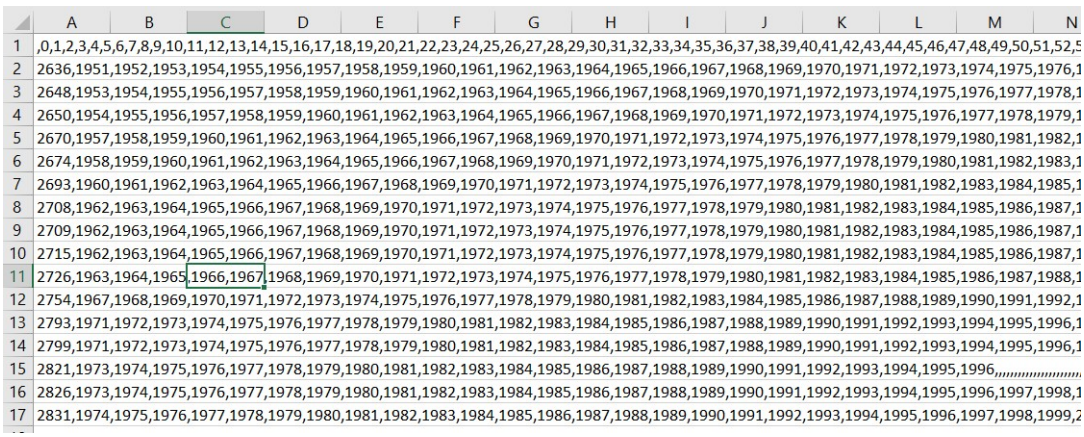


```
,0,1,2,3,4,5,6
5046,0.0,115.0,168.0,182.0,184.0,186.0,186.0
```

Now that we understand how to web scrap the membership data for one treaty, let's try web scraping the membership data for our entire list of treaties. One of the main difference compared to what we have seen so far is that because the treaties don't necessary have the same number of values (because some were opened for ratification longer than others), we need some additional steps to normalize the size of our table, because pandas does not like incomplete datasets (thank you Camshaka for your help with this!) The full code can be downloaded directly from here.

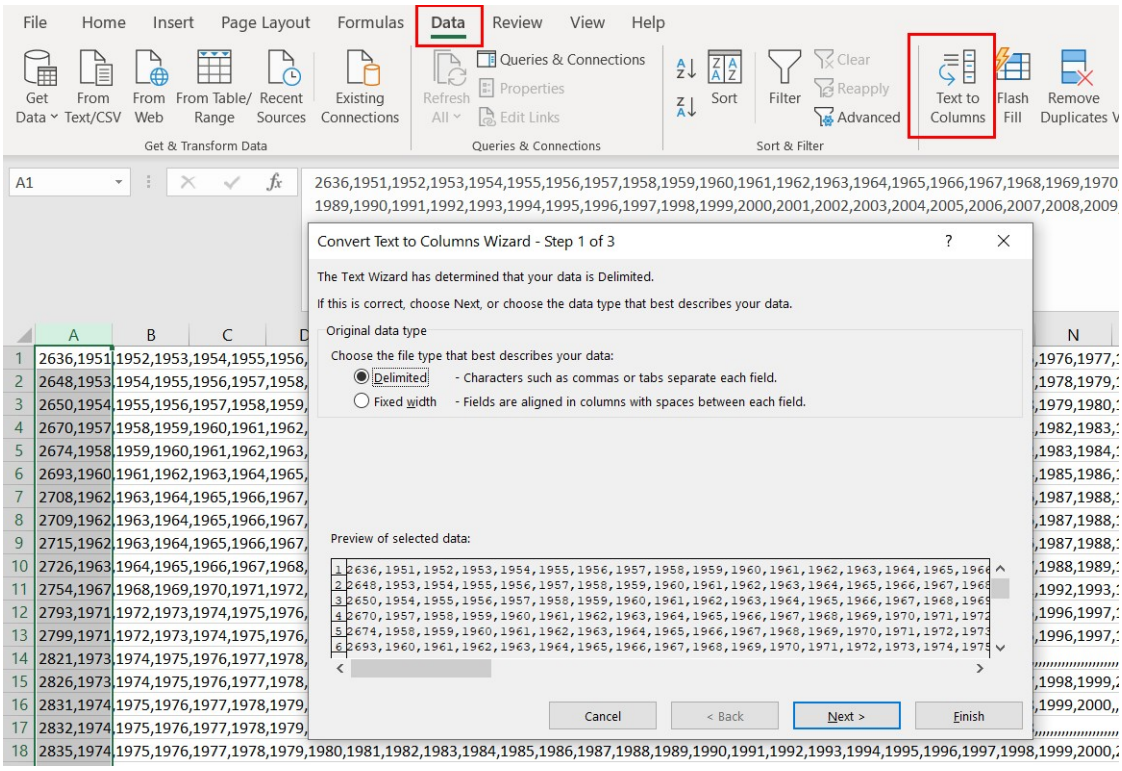
3 Formatting the data using Excel

After web scraping a bunch of treaties membership data, you should end up with two excel files: dates.csv and members.csv. Let's open the dates file first. It should look like this:



```
,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,
2636,1951,1952,1953,1954,1955,1956,1957,1958,1959,1960,1961,1962,1963,1964,1965,1966,1967,1968,1969,1970,1971,1972,1973,1974,1975,1976,1
2648,1953,1954,1955,1956,1957,1958,1959,1960,1961,1962,1963,1964,1965,1966,1967,1968,1969,1970,1971,1972,1973,1974,1975,1976,1977,1978,1
2650,1954,1955,1956,1957,1958,1959,1960,1961,1962,1963,1964,1965,1966,1967,1968,1969,1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1
2670,1957,1958,1959,1960,1961,1962,1963,1964,1965,1966,1967,1968,1969,1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1
2674,1958,1959,1960,1961,1962,1963,1964,1965,1966,1967,1968,1969,1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1
2693,1960,1961,1962,1963,1964,1965,1966,1967,1968,1969,1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1
2708,1962,1963,1964,1965,1966,1967,1968,1969,1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1
2709,1962,1963,1964,1965,1966,1967,1968,1969,1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1
2715,1962,1963,1964,1965,1966,1967,1968,1969,1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1
2726,1963,1964,1965,1966,1967,1968,1969,1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1988,1
2754,1967,1968,1969,1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1988,1989,1990,1991,1992,1
2793,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1988,1989,1990,1991,1992,1993,1994,1995,1996,1
2799,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1988,1989,1990,1991,1992,1993,1994,1995,1996,1
2821,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1988,1989,1990,1991,1992,1993,1994,1995,1996,,,,,,,,,,,,,,,,,,,,,,,
2826,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1988,1989,1990,1991,1992,1993,1994,1995,1996,1997,1998,1
2831,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1988,1989,1990,1991,1992,1993,1994,1995,1996,1997,1998,1999,2
```

What do we do with this now? First we can delete the first row, which indicates the number of columns in our table. We don't need it. Next, we are gonna use the "text to columns" feature of excel (in the "data" tab) to format our file. In the prompt, choose "Delimited" → Next → "Comma" → Next → "General" → "Finish".

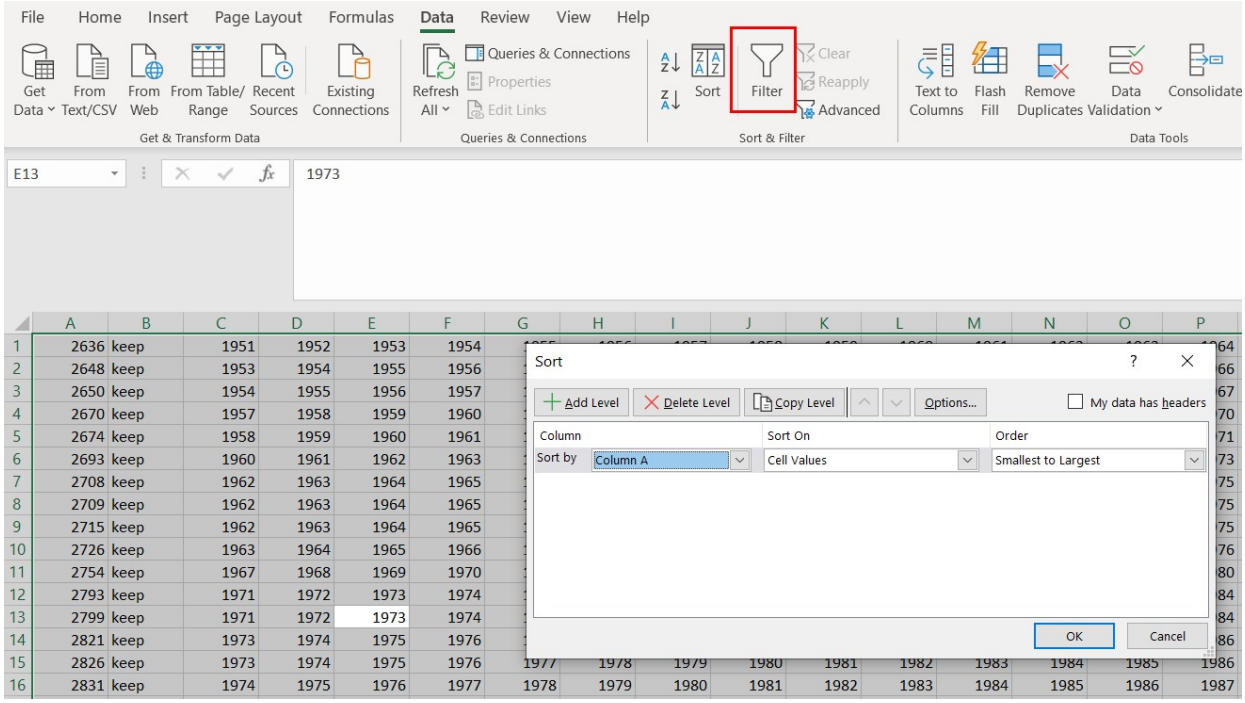


Now your file should look like this:

	A	B	C	D	E	F	G	H	I	J
1	2636	1951	1952	1953	1954	1955	1956	1957	1958	1959
2	2648	1953	1954	1955	1956	1957	1958	1959	1960	1961
3	2650	1954	1955	1956	1957	1958	1959	1960	1961	1962
4	2670	1957	1958	1959	1960	1961	1962	1963	1964	1965
5	2674	1958	1959	1960	1961	1962	1963	1964	1965	1966
6	2693	1960	1961	1962	1963	1964	1965	1966	1967	1968
7	2708	1962	1963	1964	1965	1966	1967	1968	1969	1970
8	2709	1962	1963	1964	1965	1966	1967	1968	1969	1970
9	2715	1962	1963	1964	1965	1966	1967	1968	1969	1970
10	2726	1963	1964	1965	1966	1967	1968	1969	1970	1971
11	2754	1967	1968	1969	1970	1971	1972	1973	1974	1975
12	2793	1971	1972	1973	1974	1975	1976	1977	1978	1979

Note that the first column corresponds to the treaties IDs. Next to the ID column, insert a new column. Populate each cell from this column with the word "keep".

Now, let's do the same for the members file. But instead of populating the inserted column with the word "keep", leave it empty. Finally, copy-paste the table from the members file into the dates file, below the last row of dates. Use the "sort" feature in the "Data" tab to sort your first column from the smallest ID to the largest.



You should now have for each treaty, the row containing the dates, followed by the row containing the number of members.

	A	B	C	D	E	F	G	H	I	J	K
1	2636	keep	1951	1952	1953	1954	1955	1956	1957	1958	1959
2	2636		0	12	19	24	31	37	39	39	39
3	2648	keep	1953	1954	1955	1956	1957	1958	1959	1960	1961
4	2648		0	6	11	11	11	12	15	15	16
5	2650	keep	1954	1955	1956	1957	1958	1959	1960	1961	1962
6	2650		0	0	0	0	11	12	12	14	19
7	2670	keep	1957	1958	1959	1960	1961	1962	1963	1964	1965
8	2670		0	0	13	14	15	15	16	16	16
9	2674	keep	1958	1959	1960	1961	1962	1963	1964	1965	1966
10	2674		0	0,5	0,5	0,5	23	27	29	35	40
11	2693	keep	1960	1961	1962	1963	1964	1965	1966	1967	1968
12	2693		0	0	0	0	0	0	0	0	6
13	2708	keep	1962	1963	1964	1965	1966	1967	1968	1969	1970
14	2708		0	0	0	0	0	0	0	0	0
15	2709	keep	1962	1963	1964	1965	1966	1967	1968	1969	1970
16	2709		0	9	9	9	9	9	9	9	9

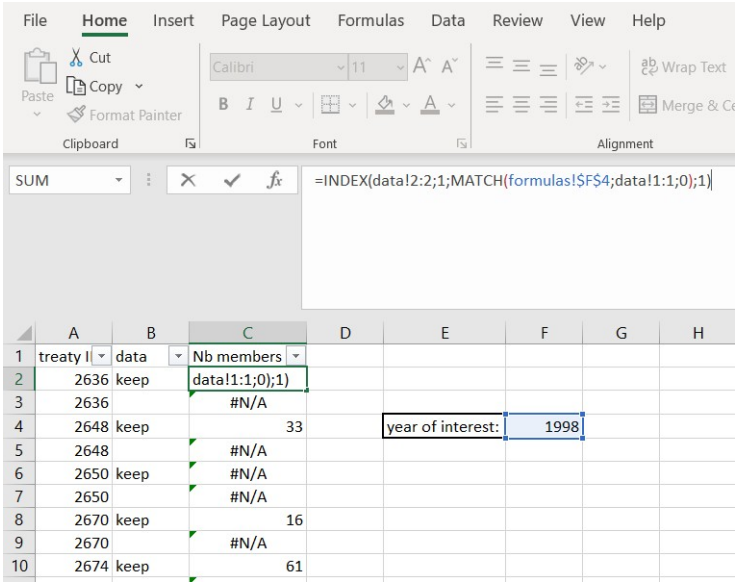
Finally, we are going to create a formula in another sheet that will return for each treaty, the number of members for a specific date. To do so we are going to use the INDEX and MATCH functions. **Don't forget to save your excel file into a .xls or .xlsx file, as the csv format cannot handle different sheets.** To make things easier, rename sheet1 "data" and sheet2 "formulas".

In the formula sheet, copy paste the two first columns from the data sheet (keeping the duplicates). Keep the third column empty (this is where the number of members will be displayed). In one of the next column, choose a cell in which you indicate the year of interest. In this example, I use cell F4. Your formulas sheet should look like this:

	A	B	C	D	E	F	G
1	treaty ID	data	Nb members				
2	2636	keep					
3	2636						
4	2648	keep			year of interest:	1998	
5	2648						
6	2650	keep					
7	2650						
8	2670	keep					
9	2670						
10	2674	keep					
11	2674						
12	2693	keep					
13	2693						

Finally, enter the following formula into the first cell of the "Nb members" column:

```
=INDEX(data!2:2;1;MATCH(formulas!$F$4;data!1:1;0);1)
```



For more information about how to use INDEX and MATCH to return particular values, have a look here. Extend the formulas to the last treaty of the list. Finally, use the "filter" feature of excel to only show the cells populated with "keep" in the second column. *Voilà!*

A	B	C	D	E	F	G
treaty I	data	Nb members				
2636	keep	96				
2648	keep	33		year of interest:	1998	
2650	keep	#N/A				
2670	keep	16				
2674	keep	61				
2693	keep	14				
2708	keep	0				
2709	keep	9				
2715	keep	4				
2726	keep	121				
2754	keep	27				
2793	keep	114				
2799	keep	14				
2821	keep	#N/A				
2826	keep	28				
2831	keep	10				
2832	keep	13				
2835	keep	23				
2866	keep	64				
2874	keep	104				

A	B	C	D	E	F	G
treaty I	data	Nb members				
2636	keep	96				
2648	keep	36		year of interest:	2010	
2650	keep	#N/A				
2670	keep	16				
2674	keep	62				
2693	keep	16				
2708	keep	0				
2709	keep	9				
2715	keep	4				
2726	keep	124				
2754	keep	28				
2793	keep	160				
2799	keep	17				
2821	keep	#N/A				
2826	keep	32				
2831	keep	#N/A				
2832	keep	#N/A				
2835	keep	27				
2866	keep	74				
2874	keep	150				

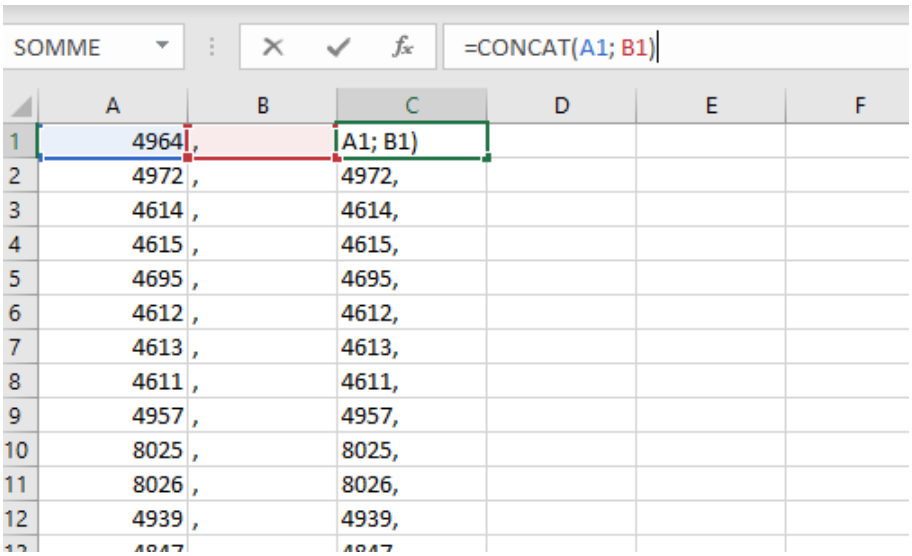
The number of members will update when you change the date of interest. When there is no data for a particular year, the formula will return #N/A.

If you have troubles with the formulas, you can also download the excel file from here. and insert your data into the "data" sheet directly.

4 Extra Help

4.1 CONCAT in Excel

What we need is a column with the treaties IDs and one column filled with commas. Then we use the CONCAT function to add the comma to each ID. (Note that if you are using the ENGLISH version of excel, you need to use "," instead of ";" in the CONCAT function).



We can now copy-paste the content from the last column into our list of treaties in our python code!