BAlRifai_Assignment1

June 16, 2021

[1]:
```
import requests
from bs4 import BeautifulSoup
from urllib.parse import urlparse
import pandas as pd
import numpy as np
from datetime import datetime
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
import scipy, pylab
```

1 Step 1: Scraping data for the top 50 solar flares shown on SpaceWeatherLive

1.1 Step 1.2 Use requests to get (as in, HTTP GET) the URL

[2]:
```
SpaceWeatherLivepage = requests.get('https://cmsc320.github.io/files/
↪top-50-solar-flares.html') #Extracting Space Weather Live HTML via URL
```

1.2 Step 1.3 Extract the text from the page

[3]:
```
SpaceWeatherLivepageText = SpaceWeatherLivepage.content #Getting the content of␣
↪the page
# print(SpaceWeatherLivepageText) # Commented out to save space for reader
```

1.3 Step 1.4 Use BeautifulSoup to read and parse the data, either as html or lxml and 1.5 Use prettify() to view the content and find the appropriate table

[4]:
```
soup = BeautifulSoup(SpaceWeatherLivepageText, 'html.parser') #Parsing the page␣
↪text as HTML for beautiful soup
# print(soup.prettify()) # Commented out to save space for reader
```

1.4 Step 1.6 Use find to save the table as a variable

[5]:
```
table = soup.find("table") #Finding the table (there is only 1) on the page
# print(table) # Commented out to save space for reader
```

1.5 Step 1.7 Use pandas to read in the HTML file. HINT make-sure the above data is properly typecast, if necessary and Step 1.8 Set reasonable names for the table columns, e.g., rank, x_classification, date, region, start_time, maximum_time, end_time, movie.

[6]:
```
pdTables = pd.read_html(SpaceWeatherLivepageText, flavor = 'bs4') #Read the␣
 ↪table from the html
spaceWeatherdf = pdTables[0] #Store the table
```

[7]:
```
spaceWeatherdf = spaceWeatherdf.rename(columns={"Unnamed: 0": "Rank", "Unnamed:␣
 ↪1": "X_classification", "Unnamed: 2": "Date", "Unnamed: 7": "Movie"},␣
 ↪errors="raise") #Properly name the columns
#and set proper datatypes (all str)
spaceWeatherdf['Region'] = spaceWeatherdf['Region'].apply(str)
spaceWeatherdf['Rank'] = spaceWeatherdf['Rank'].apply(str)
spaceWeatherdf['X_classification'] = spaceWeatherdf['X_classification'].
 ↪apply(str)
spaceWeatherdf['Start'] = spaceWeatherdf['Start'].apply(str)
spaceWeatherdf['Date'] = spaceWeatherdf['Date'].apply(str)
spaceWeatherdf['Maximum'] = spaceWeatherdf['Maximum'].apply(str)
spaceWeatherdf['End'] = spaceWeatherdf['End'].apply(str)
spaceWeatherdf
```

[7]:

	Rank	X_classification	Date	Region	Start	Maximum	End
0	1	X28+	2003/11/04	486	19:29	19:53	20:06
1	2	X20+	2001/04/02	9393	21:32	21:51	22:03
2	3	X17.2+	2003/10/28	486	09:51	11:10	11:24
3	4	X17+	2005/09/07	808	17:17	17:40	18:03
4	5	X14.4	2001/04/15	9415	13:19	13:50	13:55
5	6	X10	2003/10/29	486	20:37	20:49	21:01
6	7	X9.4	1997/11/06	8100	11:49	11:55	12:01
7	8	X9.3	2017/09/06	2673	11:53	12:02	12:10
8	9	X9	2006/12/05	930	10:18	10:35	10:45
9	10	X8.3	2003/11/02	486	17:03	17:25	17:39
10	11	X8.2	2017/09/10	2673	15:35	16:06	16:31
11	12	X7.1	2005/01/20	720	06:36	07:01	07:26
12	13	X6.9	2011/08/09	1263	07:48	08:05	08:08
13	14	X6.5	2006/12/06	930	18:29	18:47	19:00
14	15	X6.2	2005/09/09	808	19:13	20:04	20:36
15	16	X6.2	2001/12/13	9733	14:20	14:30	14:35
16	17	X5.7	2000/07/14	9077	10:03	10:24	10:43
17	18	X5.6	2001/04/06	9415	19:10	19:21	19:31

18	19	X5.4	2012/03/07	1429	00:02	00:24	00:40
19	20	X5.4	2005/09/08	808	20:52	21:06	21:17
20	21	X5.4	2003/10/23	486	08:19	08:35	08:49
21	22	X5.3	2001/08/25	9591	16:23	16:45	17:04
22	23	X4.9	2014/02/25	1990	00:39	00:49	01:03
23	24	X4.9	1998/08/18	8307	22:10	22:19	22:28
24	25	X4.8	2002/07/23	39	00:18	00:35	00:47
25	26	X4	2000/11/26	9236	16:34	16:48	16:56
26	27	X3.9	2003/11/03	488	09:43	09:55	10:19
27	28	X3.9	1998/08/19	8307	21:35	21:45	21:50
28	29	X3.8	2005/01/17	720	06:59	09:52	10:07
29	30	X3.7	1998/11/22	8384	06:30	06:42	06:49
30	31	X3.6	2005/09/09	808	09:42	09:59	10:08
31	32	X3.6	2004/07/16	649	13:49	13:55	14:01
32	33	X3.6	2003/05/28	365	00:17	00:27	00:39
33	34	X3.4	2006/12/13	930	02:14	02:40	02:57
34	35	X3.4	2001/12/28	9767	20:02	20:45	21:32
35	36	X3.3	2013/11/05	1890	22:07	22:12	22:15
36	37	X3.3	2002/07/20	39	21:04	21:30	21:54
37	38	X3.3	1998/11/28	8395	04:54	05:52	06:13
38	39	X3.2	2013/05/14	1748	00:00	01:11	01:20
39	40	X3.1	2014/10/24	2192	21:07	21:41	22:13
40	41	X3.1	2002/08/24	69	00:49	01:12	01:31
41	42	X3	2002/07/15	30	19:59	20:08	20:14
42	43	X2.8	2013/05/13	1748	15:48	16:05	16:16
43	44	X2.8	2001/12/11	9733	07:58	08:08	08:14
44	45	X2.8	1998/08/18	8307	08:14	08:24	08:32
45	46	X2.7	2015/05/05	2339	22:05	22:11	22:15
46	47	X2.7	2003/11/03	488	01:09	01:30	01:45
47	48	X2.7	1998/05/06	8210	07:58	08:09	08:20
48	49	X2.6	2005/01/15	720	22:25	23:02	23:31
49	50	X2.6	2001/09/24	9632	09:32	10:38	11:09

	Movie
0	MovieView archive
1	MovieView archive
2	MovieView archive
3	MovieView archive
4	MovieView archive
5	MovieView archive
6	MovieView archive
7	MovieView archive
8	MovieView archive
9	MovieView archive
10	MovieView archive
11	MovieView archive
12	MovieView archive

```
13  MovieView archive
14  MovieView archive
15  MovieView archive
16  MovieView archive
17  MovieView archive
18  MovieView archive
19  MovieView archive
20  MovieView archive
21  MovieView archive
22  MovieView archive
23       View archive
24  MovieView archive
25  MovieView archive
26  MovieView archive
27       View archive
28  MovieView archive
29  MovieView archive
30  MovieView archive
31  MovieView archive
32  MovieView archive
33  MovieView archive
34  MovieView archive
35  MovieView archive
36  MovieView archive
37  MovieView archive
38  MovieView archive
39  MovieView archive
40  MovieView archive
41  MovieView archive
42  MovieView archive
43  MovieView archive
44       View archive
45  MovieView archive
46  MovieView archive
47  MovieView archive
48  MovieView archive
49  MovieView archive
```

1.6 Step 1 description

Including the comments left within the code, the overall goal of step 1 was to extract the table from SpaceWeatherLive.com and ensure that all column names were properly labeled and all data was extracted. This was done using pd.read_html and df.rename respectively.

2 Step 2: Tidy the top 50 solar flare data using pandas

2.1 Step 2.1 Drop the last column of the table, since we are not going to use it moving forward.

[8]:
```
spaceWeatherdf = spaceWeatherdf.drop(['Movie'], axis = 1)
spaceWeatherdf
```

[8]:

	Rank	X_classification	Date	Region	Start	Maximum	End
0	1	X28+	2003/11/04	486	19:29	19:53	20:06
1	2	X20+	2001/04/02	9393	21:32	21:51	22:03
2	3	X17.2+	2003/10/28	486	09:51	11:10	11:24
3	4	X17+	2005/09/07	808	17:17	17:40	18:03
4	5	X14.4	2001/04/15	9415	13:19	13:50	13:55
5	6	X10	2003/10/29	486	20:37	20:49	21:01
6	7	X9.4	1997/11/06	8100	11:49	11:55	12:01
7	8	X9.3	2017/09/06	2673	11:53	12:02	12:10
8	9	X9	2006/12/05	930	10:18	10:35	10:45
9	10	X8.3	2003/11/02	486	17:03	17:25	17:39
10	11	X8.2	2017/09/10	2673	15:35	16:06	16:31
11	12	X7.1	2005/01/20	720	06:36	07:01	07:26
12	13	X6.9	2011/08/09	1263	07:48	08:05	08:08
13	14	X6.5	2006/12/06	930	18:29	18:47	19:00
14	15	X6.2	2005/09/09	808	19:13	20:04	20:36
15	16	X6.2	2001/12/13	9733	14:20	14:30	14:35
16	17	X5.7	2000/07/14	9077	10:03	10:24	10:43
17	18	X5.6	2001/04/06	9415	19:10	19:21	19:31
18	19	X5.4	2012/03/07	1429	00:02	00:24	00:40
19	20	X5.4	2005/09/08	808	20:52	21:06	21:17
20	21	X5.4	2003/10/23	486	08:19	08:35	08:49
21	22	X5.3	2001/08/25	9591	16:23	16:45	17:04
22	23	X4.9	2014/02/25	1990	00:39	00:49	01:03
23	24	X4.9	1998/08/18	8307	22:10	22:19	22:28
24	25	X4.8	2002/07/23	39	00:18	00:35	00:47
25	26	X4	2000/11/26	9236	16:34	16:48	16:56
26	27	X3.9	2003/11/03	488	09:43	09:55	10:19
27	28	X3.9	1998/08/19	8307	21:35	21:45	21:50
28	29	X3.8	2005/01/17	720	06:59	09:52	10:07
29	30	X3.7	1998/11/22	8384	06:30	06:42	06:49
30	31	X3.6	2005/09/09	808	09:42	09:59	10:08
31	32	X3.6	2004/07/16	649	13:49	13:55	14:01
32	33	X3.6	2003/05/28	365	00:17	00:27	00:39
33	34	X3.4	2006/12/13	930	02:14	02:40	02:57
34	35	X3.4	2001/12/28	9767	20:02	20:45	21:32
35	36	X3.3	2013/11/05	1890	22:07	22:12	22:15
36	37	X3.3	2002/07/20	39	21:04	21:30	21:54
37	38	X3.3	1998/11/28	8395	04:54	05:52	06:13

38	39	X3.2	2013/05/14	1748	00:00	01:11	01:20
39	40	X3.1	2014/10/24	2192	21:07	21:41	22:13
40	41	X3.1	2002/08/24	69	00:49	01:12	01:31
41	42	X3	2002/07/15	30	19:59	20:08	20:14
42	43	X2.8	2013/05/13	1748	15:48	16:05	16:16
43	44	X2.8	2001/12/11	9733	07:58	08:08	08:14
44	45	X2.8	1998/08/18	8307	08:14	08:24	08:32
45	46	X2.7	2015/05/05	2339	22:05	22:11	22:15
46	47	X2.7	2003/11/03	488	01:09	01:30	01:45
47	48	X2.7	1998/05/06	8210	07:58	08:09	08:20
48	49	X2.6	2005/01/15	720	22:25	23:02	23:31
49	50	X2.6	2001/09/24	9632	09:32	10:38	11:09

2.2 Step 2.2 Use datetime import to combine the date and each of the three time columns into three datetime columns, Step 2.3 Update the values in the dataframe, and Step 2.4 Mark regions coded as - as missing

[9]:
```
for index, row in spaceWeatherdf.iterrows():
    #Update the values of the start, maximum, and end columns to be in the␣
↪datetime format. I used datetime.strptime function for this
    #Also, the set_value suggested in the documentation is depricated
    spaceWeatherdf.at[index, 'Start'] = datetime.strptime(row['Date'] + " " +␣
↪row['Start'], '%Y/%m/%d %H:%M')
    spaceWeatherdf.at[index, 'Maximum'] = datetime.strptime(row['Date'] + " " +␣
↪row['Maximum'], '%Y/%m/%d %H:%M')
    spaceWeatherdf.at[index, 'End'] = datetime.strptime(row['Date'] + " " +␣
↪row['End'], '%Y/%m/%d %H:%M')

    #Check for any missing values in the region column and update them to be␣
↪'missing data'
    spaceWeatherdf = spaceWeatherdf.replace('-', np.nan)

    #Fix decimals so that Part 2.2 works properly when matching by␣
↪classification
    if (len(row['X_classification']) == 2):
        spaceWeatherdf.at[index, 'X_classification'] = row['X_classification']␣
↪+ ".0"
    elif ('+' in row['X_classification']):
        spaceWeatherdf.at[index, 'X_classification'] = row['X_classification'][:
↪-1] + "."

    #Fix region inaccuracy, 0486 is actually 10486 and so forth with any␣
↪regions that start with 0
    if (len(row['Region']) == 3):
        spaceWeatherdf.at[index, 'Region'] = "10" + row['Region']
    elif (len(row['Region']) == 2):
```

```
        spaceWeatherdf.at[index, 'Region'] = "100" + row['Region']

#Dropping the date column and renaming start, maximum, and end to be Datetime␣
 ↪columns. Also moved region to end to match documentation
spaceWeatherdf = spaceWeatherdf.drop(['Date'], axis = 1)
spaceWeatherdf = spaceWeatherdf.rename(columns={"Start": "Start_Datetime",␣
 ↪"Maximum": "Maximum_Datetime", "End": "End_Datetime"}, errors="raise")
spaceWeatherdf = spaceWeatherdf[['Rank', 'X_classification', 'Start_Datetime',␣
 ↪'Maximum_Datetime', 'End_Datetime', 'Region']]
spaceWeatherdf['Start_Datetime'] = pd.
 ↪to_datetime(spaceWeatherdf['Start_Datetime'])
spaceWeatherdf['Maximum_Datetime'] = pd.
 ↪to_datetime(spaceWeatherdf['Maximum_Datetime'])
spaceWeatherdf['End_Datetime'] = pd.to_datetime(spaceWeatherdf['End_Datetime'])
spaceWeatherdf
```

[9]:

	Rank	X_classification	Start_Datetime	Maximum_Datetime	\
0	1	X28.	2003-11-04 19:29:00	2003-11-04 19:53:00	
1	2	X20.	2001-04-02 21:32:00	2001-04-02 21:51:00	
2	3	X17.2.	2003-10-28 09:51:00	2003-10-28 11:10:00	
3	4	X17.	2005-09-07 17:17:00	2005-09-07 17:40:00	
4	5	X14.4	2001-04-15 13:19:00	2001-04-15 13:50:00	
5	6	X10	2003-10-29 20:37:00	2003-10-29 20:49:00	
6	7	X9.4	1997-11-06 11:49:00	1997-11-06 11:55:00	
7	8	X9.3	2017-09-06 11:53:00	2017-09-06 12:02:00	
8	9	X9.0	2006-12-05 10:18:00	2006-12-05 10:35:00	
9	10	X8.3	2003-11-02 17:03:00	2003-11-02 17:25:00	
10	11	X8.2	2017-09-10 15:35:00	2017-09-10 16:06:00	
11	12	X7.1	2005-01-20 06:36:00	2005-01-20 07:01:00	
12	13	X6.9	2011-08-09 07:48:00	2011-08-09 08:05:00	
13	14	X6.5	2006-12-06 18:29:00	2006-12-06 18:47:00	
14	15	X6.2	2005-09-09 19:13:00	2005-09-09 20:04:00	
15	16	X6.2	2001-12-13 14:20:00	2001-12-13 14:30:00	
16	17	X5.7	2000-07-14 10:03:00	2000-07-14 10:24:00	
17	18	X5.6	2001-04-06 19:10:00	2001-04-06 19:21:00	
18	19	X5.4	2012-03-07 00:02:00	2012-03-07 00:24:00	
19	20	X5.4	2005-09-08 20:52:00	2005-09-08 21:06:00	
20	21	X5.4	2003-10-23 08:19:00	2003-10-23 08:35:00	
21	22	X5.3	2001-08-25 16:23:00	2001-08-25 16:45:00	
22	23	X4.9	2014-02-25 00:39:00	2014-02-25 00:49:00	
23	24	X4.9	1998-08-18 22:10:00	1998-08-18 22:19:00	
24	25	X4.8	2002-07-23 00:18:00	2002-07-23 00:35:00	
25	26	X4.0	2000-11-26 16:34:00	2000-11-26 16:48:00	
26	27	X3.9	2003-11-03 09:43:00	2003-11-03 09:55:00	
27	28	X3.9	1998-08-19 21:35:00	1998-08-19 21:45:00	
28	29	X3.8	2005-01-17 06:59:00	2005-01-17 09:52:00	
29	30	X3.7	1998-11-22 06:30:00	1998-11-22 06:42:00	

```
30   31             X3.6 2005-09-09 09:42:00 2005-09-09 09:59:00
31   32             X3.6 2004-07-16 13:49:00 2004-07-16 13:55:00
32   33             X3.6 2003-05-28 00:17:00 2003-05-28 00:27:00
33   34             X3.4 2006-12-13 02:14:00 2006-12-13 02:40:00
34   35             X3.4 2001-12-28 20:02:00 2001-12-28 20:45:00
35   36             X3.3 2013-11-05 22:07:00 2013-11-05 22:12:00
36   37             X3.3 2002-07-20 21:04:00 2002-07-20 21:30:00
37   38             X3.3 1998-11-28 04:54:00 1998-11-28 05:52:00
38   39             X3.2 2013-05-14 00:00:00 2013-05-14 01:11:00
39   40             X3.1 2014-10-24 21:07:00 2014-10-24 21:41:00
40   41             X3.1 2002-08-24 00:49:00 2002-08-24 01:12:00
41   42             X3.0 2002-07-15 19:59:00 2002-07-15 20:08:00
42   43             X2.8 2013-05-13 15:48:00 2013-05-13 16:05:00
43   44             X2.8 2001-12-11 07:58:00 2001-12-11 08:08:00
44   45             X2.8 1998-08-18 08:14:00 1998-08-18 08:24:00
45   46             X2.7 2015-05-05 22:05:00 2015-05-05 22:11:00
46   47             X2.7 2003-11-03 01:09:00 2003-11-03 01:30:00
47   48             X2.7 1998-05-06 07:58:00 1998-05-06 08:09:00
48   49             X2.6 2005-01-15 22:25:00 2005-01-15 23:02:00
49   50             X2.6 2001-09-24 09:32:00 2001-09-24 10:38:00
```

```
          End_Datetime Region
0  2003-11-04 20:06:00  10486
1  2001-04-02 22:03:00   9393
2  2003-10-28 11:24:00  10486
3  2005-09-07 18:03:00  10808
4  2001-04-15 13:55:00   9415
5  2003-10-29 21:01:00  10486
6  1997-11-06 12:01:00   8100
7  2017-09-06 12:10:00   2673
8  2006-12-05 10:45:00  10930
9  2003-11-02 17:39:00  10486
10 2017-09-10 16:31:00   2673
11 2005-01-20 07:26:00  10720
12 2011-08-09 08:08:00   1263
13 2006-12-06 19:00:00  10930
14 2005-09-09 20:36:00  10808
15 2001-12-13 14:35:00   9733
16 2000-07-14 10:43:00   9077
17 2001-04-06 19:31:00   9415
18 2012-03-07 00:40:00   1429
19 2005-09-08 21:17:00  10808
20 2003-10-23 08:49:00  10486
21 2001-08-25 17:04:00   9591
22 2014-02-25 01:03:00   1990
23 1998-08-18 22:28:00   8307
24 2002-07-23 00:47:00  10039
```

```
25 2000-11-26 16:56:00   9236
26 2003-11-03 10:19:00  10488
27 1998-08-19 21:50:00   8307
28 2005-01-17 10:07:00  10720
29 1998-11-22 06:49:00   8384
30 2005-09-09 10:08:00  10808
31 2004-07-16 14:01:00  10649
32 2003-05-28 00:39:00  10365
33 2006-12-13 02:57:00  10930
34 2001-12-28 21:32:00   9767
35 2013-11-05 22:15:00   1890
36 2002-07-20 21:54:00  10039
37 1998-11-28 06:13:00   8395
38 2013-05-14 01:20:00   1748
39 2014-10-24 22:13:00   2192
40 2002-08-24 01:31:00  10069
41 2002-07-15 20:14:00  10030
42 2013-05-13 16:16:00   1748
43 2001-12-11 08:14:00   9733
44 1998-08-18 08:32:00   8307
45 2015-05-05 22:15:00   2339
46 2003-11-03 01:45:00  10488
47 1998-05-06 08:20:00   8210
48 2005-01-15 23:31:00  10720
49 2001-09-24 11:09:00   9632
```

2.3 Step 2 description

Including the comments left within the code, the overall goal of step 2 was to tidy up the table from SpaceWeatherLive.com that was extracted in step 1 by dropping the movies column, fixing any missing values (NaN) using df.replace, converting all times to Datetimes to drop the date column, and finally cleaning up the decimal values with + signs to match the NASA table used in Step 3. rename is used once again to properly name the new datetime columns from their original "x_time" format.

A big part of this step was also discovering that the data had a region inaccuracy with regions with more than 4 characters (such as 10486) which was originally written as 0486. This was fixed by adding 10 or 100 behind the numbers depending on their length.

3 Step 3: Scrape the NASA data

3.1 Step 3.1 Use BeautifulSoup functions (e.g., find, findAll) and string functions (e.g., split and built-in slicing capabilities) to obtain each row of data as a long string and Step 3.2 Use string::split and list comprehensions or similar to separate each line of text into a data row. Choose appropriate names for columns.

[10]:
```
NASApage = requests.get('https://www.hcbravo.org/IntroDataSci/misc/waves_type2.
 ↪html') #Extracting NASA page HTML
NASApageText = NASApage.content #Getting the content from the page
# print(NASApageText) # Commented out to save space for reader
```

[11]:
```
#Prettify the page to determine the structure
NASASoup = BeautifulSoup(NASApageText, 'html.parser')
# print(NASASoup.prettify()) # Commented out to save space for reader
```

3.1.1 It can be seen here that the tables of NASAs website are not formatted as a table but rather a bunch of text and links within a <pre> tag and the width is fixed. The pattern is that there are 14 elements in each row seperated by space ' ' excluding the plots

[12]:
```
preText = NASASoup.find('pre').text
results = []

for line in preText.split('\n')[1:-1]:
    results.append(line)

NASAdf = pd.DataFrame(results)
NASAdf = NASAdf.drop([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 494, 493]) #Drop first␣
 ↪10 rows and last 2; unnecessary rows before and after data
NASAdf.columns = ['rows'] #Assigning a column name (I could not figure out the␣
 ↪default column name)

#Split by whitespace and add appropriate number of columns w/ names
NASAdf = NASAdf['rows'].str.split(" +",expand = True) #Split by space
cols = [14,15,16,17,18,19,20,21,22,23] #marking the columns for deletion
NASAdf = NASAdf.drop(NASAdf.columns[cols], axis=1) #Remove last 7 columns␣
 ↪because they come from the explanations column

#Reset the indicies to start from 0 since some rows were deleted
NASAdf = NASAdf.reset_index(drop=True)
NASAdf.columns = ['Start_Date', 'Start_Time', 'End_Date', 'End_Time',␣
 ↪'Start_Frequency', 'End_Frequency', 'Flare_Location', 'Flare_Region',␣
 ↪'Flare_Classification', 'Cme_Date', 'Cme_Time', 'Cme_Angle', 'Cme_Width',␣
 ↪'Cme_Speed']
NASAdf
```

[12]:

	Start_Date	Start_Time	End_Date	End_Time	Start_Frequency	End_Frequency
0	1997/04/01	14:00	04/01	14:15	8000	4000
1	1997/04/07	14:30	04/07	17:30	11000	1000
2	1997/05/12	05:15	05/14	16:00	12000	80
3	1997/05/21	20:20	05/21	22:00	5000	500
4	1997/09/23	21:53	09/23	22:16	6000	2000
..	…	…	…	…	…	…
477	2014/12/13	14:27	12/13	14:51	14000	3900
478	2014/12/17	04:09	12/17	04:19	2900	2100
479	2014/12/17	05:00	12/17	05:09	14000	11500
480	2014/12/18	22:31	12/18	22:54	5100	1300
481	2014/12/21	12:05	12/21	12:28	14000	7400

	Flare_Location	Flare_Region	Flare_Classification	Cme_Date	Cme_Time
0	S25E16	8026	M1.3	04/01	15:18
1	S28E19	8027	C6.8	04/07	14:27
2	N21W08	8038	C1.3	05/12	05:30
3	N05W12	8040	M1.3	05/21	21:00
4	S29E25	8088	C1.4	09/23	22:02
..	…	…	…	…	…
477	W90b	-----	----	12/13	14:24
478	S11E33	12241	M1.1	12/17	02:00
479	S20E09	12242	M8.7	12/17	05:00
480	S11E15	12241	M6.9	12/19	01:04
481	S14W25	12241	M1.0	12/21	12:12

	Cme_Angle	Cme_Width	Cme_Speed
0	74	79	312
1	Halo	360	878
2	Halo	360	464
3	263	165	296
4	133	155	712
..	…	…	…
477	Halo	360	2222
478	107	108	869
479	Halo	360	587
480	Halo	360	1195
481	Halo	360	669

[482 rows x 14 columns]

3.2 Step 3 description

Including the comments left within the code, the overall goal of step 3 was to extract the table from NASAs solar flare data site. This proved especially challenging because the table was formatted as a pre tag, which is a tag with a set width in html. To extract the rows of the table, I used .split() to split on new line characters, which gave me a big chunk of the data but left some stray rows and

columns that had unnecessary text for the table (such as the description and dividers '==='). I dropped these extra rows and columns in this step and then removed trailing whitespace between values (some values had more than one space ahead of them). Finally, I reset the index of the table to account for the dropped rows and properly labeled the table.

4 Step 4: Tidy the NASA the table

[13]:
```
NASAdf = NASAdf.assign(Is_Halo = 'False')
NASAdf = NASAdf.assign(Width_Lower_Bound = 'False')

for index, row in NASAdf.iterrows():
    #Step 4.1 Recode any missing entries as NaN.
    #The indicators of missing values are:
    # ----
    # -----
    # BACK
    # --/--
    # --:--
    # FILA
    # DSF
    # ????
    # altr
    # EP?
    # EP
    NASAdf = NASAdf.replace('----', np.nan) #For flare region and sometimes CME width and speed
    NASAdf = NASAdf.replace('-----', np.nan) #For flare classification
    NASAdf = NASAdf.replace('---', np.nan) #For flare width
    NASAdf = NASAdf.replace('BACK', np.nan) #For flare location
    NASAdf = NASAdf.replace('--/--', np.nan) #For Date
    NASAdf = NASAdf.replace('--:--', np.nan) #For Time
    NASAdf = NASAdf.replace('FILA', np.nan) #For Filament
    NASAdf = NASAdf.replace('DSF', np.nan) #For Disappearing Solar Filament
    NASAdf = NASAdf.replace('EP', np.nan) #Not explained
    NASAdf = NASAdf.replace('EP?', np.nan) #Not explained
    NASAdf = NASAdf.replace('DIM', np.nan) #Dimming
    NASAdf = NASAdf.replace('altr', np.nan) #Not explained
    NASAdf = NASAdf.replace('????', np.nan) #For Frequencies
    NASAdf = NASAdf.replace('360h', '360') #For width variation

    #Update the values of the any time columns to be in the datetime format.
    NASAdf.at[index, 'Start_Date'] = datetime.strptime(row['Start_Date'] + " " + row['Start_Time'], '%Y/%m/%d %H:%M')

    year = str(NASAdf.at[index, 'Start_Date'].year)
```

```
    #Fixing 24 hour time... I have no idea why NASA did this
    if (row['End_Time'] == "24:00"):
        row['Start_Date'] = "00:00"
        NASAdf.at[index, 'End_Time'] = "00:00"
        NASAdf.at[index, 'End_Date'] = datetime.strptime(year + "/" +␣
↪row['End_Date'] + " 00:00", '%Y/%m/%d %H:%M')
    else:
        NASAdf.at[index, 'End_Date'] = datetime.strptime(year + "/" +␣
↪row['End_Date'] + " " + row['End_Time'], '%Y/%m/%d %H:%M')

    if (row['Cme_Date'] != '--/--'):
        NASAdf.at[index, 'Cme_Date'] = datetime.strptime(year + "/" +␣
↪row['Cme_Date'] + " " + row['Cme_Time'], '%Y/%m/%d %H:%M')

    if (row['Cme_Angle'] == 'Halo'):
        NASAdf.at[index, 'Is_Halo'] = 'True'
        NASAdf.at[index, 'Cme_Angle'] = np.nan

    if ('>' in row['Cme_Width']):
        NASAdf.at[index, 'Width_Lower_Bound'] = 'True'
        NASAdf.at[index, 'Cme_Width'] = row['Cme_Width'][1:]

#Dropping the times columns to make them Datetime columns.
NASAdf = NASAdf.drop(['Start_Time', 'End_Time', 'Cme_Time'], axis = 1)
NASAdf = NASAdf.rename(columns={"Start_Date": "Start_Datetime", "End_Date":␣
 ↪"End_Datetime", "Cme_Date": "Cme_Datetime"}, errors="raise")
NASAdf['Start_Datetime']= pd.to_datetime(NASAdf['Start_Datetime'])
NASAdf['End_Datetime']= pd.to_datetime(NASAdf['End_Datetime'])
NASAdf['Cme_Datetime']= pd.to_datetime(NASAdf['Cme_Datetime'])
NASAdf['Flare_Region']= NASAdf['Flare_Region'].apply(str)
NASAdf = NASAdf.replace('nan', np.nan) #For regions
NASAdf
```

[13]:

	Start_Datetime	End_Datetime	Start_Frequency	End_Frequency	\
0	1997-04-01 14:00:00	1997-04-01 14:15:00	8000	4000	
1	1997-04-07 14:30:00	1997-04-07 17:30:00	11000	1000	
2	1997-05-12 05:15:00	1997-05-14 16:00:00	12000	80	
3	1997-05-21 20:20:00	1997-05-21 22:00:00	5000	500	
4	1997-09-23 21:53:00	1997-09-23 22:16:00	6000	2000	
..	…	…	…	…	
477	2014-12-13 14:27:00	2014-12-13 14:51:00	14000	3900	
478	2014-12-17 04:09:00	2014-12-17 04:19:00	2900	2100	
479	2014-12-17 05:00:00	2014-12-17 05:09:00	14000	11500	
480	2014-12-18 22:31:00	2014-12-18 22:54:00	5100	1300	
481	2014-12-21 12:05:00	2014-12-21 12:28:00	14000	7400	

	Flare_Location	Flare_Region	Flare_Classification	Cme_Datetime	\

```
0           S25E16         8026                M1.3 1997-04-01 15:18:00
1           S28E19         8027                C6.8 1997-04-07 14:27:00
2           N21W08         8038                C1.3 1997-05-12 05:30:00
3           N05W12         8040                M1.3 1997-05-21 21:00:00
4           S29E25         8088                C1.4 1997-09-23 22:02:00
..             …           …                   …                   …
477           W90b          NaN                 NaN 2014-12-13 14:24:00
478         S11E33        12241                M1.1 2014-12-17 02:00:00
479         S20E09        12242                M8.7 2014-12-17 05:00:00
480         S11E15        12241                M6.9 2014-12-19 01:04:00
481         S14W25        12241                M1.0 2014-12-21 12:12:00

    Cme_Angle Cme_Width Cme_Speed Is_Halo Width_Lower_Bound
0          74        79       312   False             False
1         NaN       360       878    True             False
2         NaN       360       464    True             False
3         263       165       296   False             False
4         133       155       712   False             False
..        …       …       …     …               …
477       NaN       360      2222    True             False
478       107       108       869   False             False
479       NaN       360       587    True             False
480       NaN       360      1195    True             False
481       NaN       360       669    True             False

[482 rows x 13 columns]
```

4.1 Step 4 description

Including the comments left within the code, the overall goal of step 4 was to tidy up the NASA data gathered in step 3 by filling in missing values, of which there were a lot, half which were not explained in the description. I had to manually look through the data to find missing value codes and use .replace to convert them to NaN. I then converted all the times to be datetimes using the same methods used in step 2. The only time tidying up I had to do was converting 24:00 to 00:00 to match proper military time supported by Datetime. Lastly, I added Is_Halo and Width_Lower_Bound columns to indicate when a solar flare is a halo and if its width is indicated to be lower bound. As usual, I made sure the column names matched up and that columns had proper formats before displaying.

5 Part 2: Analysis

5.1 Question 1: Replication

Can you replicate the top 50 solar flare table in SpaceWeatherLive.com exactly using the data obtained from NASA? That is, if you get the top 50 solar flares from the NASA table based on their classification (e.g., X28 is the highest), do you get data for the same solar flare events?

Include code used to get the top 50 solar flares from the NASA table (be careful when ordering by

classification). Write a sentence or two discussing how well you can replicate the SpaceWeatherLive data from the NASA data.

5.1.1 Answer

Since the top 50 solar flares are all classified as X and then a decimal number, it is trivial to sort the NASA table by classification in decending order and then taking the top 50 results to attempt to match the SpaceWeatherLive data. So the answer is yes.

5.1.2 Solution

[14]:
```
NASAdfTop50 = NASAdf.sort_values(by='Flare_Classification', ascending=False)
NASAdfTop50 = NASAdfTop50.iloc[:50]
NASAdfTop50 = NASAdfTop50.sort_values(by='Flare_Classification',␣
 ↪ascending=False, key=lambda col: pd.to_numeric(col.str[1:]))
NASAdfTop50
```

[14]:

	Start_Datetime	End_Datetime	Start_Frequency	End_Frequency	\
242	2003-11-04 20:00:00	2003-11-04 00:00:00	10000	200	
119	2001-04-02 22:05:00	2001-04-03 02:30:00	14000	250	
234	2003-10-28 11:10:00	2003-10-29 00:00:00	14000	40	
128	2001-04-15 14:05:00	2001-04-16 13:00:00	14000	40	
235	2003-10-29 20:55:00	2003-10-29 00:00:00	11000	500	
8	1997-11-06 12:20:00	1997-11-07 08:30:00	14000	100	
330	2006-12-05 10:50:00	2006-12-05 20:00:00	14000	250	
238	2003-11-02 17:30:00	2003-11-03 01:00:00	12000	250	
290	2005-01-20 07:15:00	2005-01-20 16:30:00	14000	25	
360	2011-08-09 08:20:00	2011-08-09 08:35:00	16000	4000	
333	2006-12-06 19:00:00	2006-12-08 00:00:00	16000	30	
319	2005-09-09 19:45:00	2005-09-09 22:00:00	10000	50	
83	2000-07-14 10:30:00	2000-07-15 14:30:00	14000	80	
123	2001-04-06 19:35:00	2001-04-07 01:50:00	14000	230	
376	2012-03-07 01:00:00	2012-03-08 19:00:00	16000	30	
137	2001-08-25 16:50:00	2001-08-25 23:00:00	8000	170	
444	2014-02-25 00:56:00	2014-02-25 11:28:00	14000	100	
195	2002-07-23 00:50:00	2002-07-23 04:00:00	11000	400	
106	2000-11-26 17:00:00	2000-11-26 17:15:00	14000	7000	
240	2003-11-03 10:00:00	2003-11-03 12:30:00	6000	400	
289	2005-01-17 10:00:00	2005-01-17 10:35:00	6100	1500	
223	2003-05-28 01:00:00	2003-05-29 00:30:00	1000	200	
334	2006-12-13 02:45:00	2006-12-13 10:40:00	12000	150	
162	2001-12-28 20:35:00	2001-12-29 03:00:00	14000	350	
194	2002-07-20 21:30:00	2002-07-20 22:20:00	10000	2000	
405	2013-05-14 01:16:00	2013-05-14 02:35:00	16000	700	
202	2002-08-24 01:45:00	2002-08-24 03:25:00	5000	400	
404	2013-05-13 16:15:00	2013-05-13 19:10:00	16000	300	
239	2003-11-03 01:15:00	2003-11-03 01:25:00	3000	1500	
19	1998-05-06 08:25:00	1998-05-06 08:35:00	14000	5000	

```
144 2001-09-24 10:45:00 2001-09-25 20:00:00            7000            30
286 2005-01-15 23:00:00 2005-01-15 00:00:00            3000            40
9   1997-11-27 13:30:00 1997-11-27 14:00:00           14000          7000
278 2004-11-10 02:25:00 2004-11-10 03:40:00           14000          1000
73  2000-06-06 15:20:00 2000-06-08 09:00:00           14000            40
101 2000-11-24 15:25:00 2000-11-24 22:00:00           14000           200
125 2001-04-10 05:24:00 2001-04-10 00:00:00           14000           100
347 2011-02-15 02:10:00 2011-02-15 07:00:00           16000           400
362 2011-09-06 22:30:00 2011-09-07 15:40:00           16000           150
421 2013-10-25 15:08:00 2013-10-25 22:32:00           16000           200
320 2005-09-10 21:45:00 2005-09-10 01:00:00           14000           300
7   1997-11-04 06:00:00 1997-11-05 04:30:00           14000           100
276 2004-11-07 16:25:00 2004-11-08 20:00:00           14000            60
287 2005-01-17 09:25:00 2005-01-17 16:00:00           14000            30
127 2001-04-12 10:20:00 2001-04-12 10:40:00           14000          7000
100 2000-11-24 05:10:00 2000-11-24 15:00:00           14000           100
104 2000-11-25 19:00:00 2000-11-25 19:35:00            6000          2000
191 2002-07-18 07:55:00 2002-07-18 08:45:00           14000          1500
102 2000-11-24 22:24:00 2000-11-24 22:36:00            4000          3000
49  1999-10-14 09:10:00 1999-10-14 10:00:00           14000          4000

    Flare_Location Flare_Region Flare_Classification        Cme_Datetime  \
242         S19W83        10486                  X28. 2003-11-04 19:54:00
119         N19W72         9393                  X20. 2001-04-02 22:06:00
234         S16E08        10486                  X17. 2003-10-28 11:30:00
128         S20W85         9415                  X14. 2001-04-15 14:06:00
235         S15W02        10486                  X10. 2003-10-29 20:54:00
8           S18W63         8100                  X9.4 1997-11-06 12:10:00
330         S07E68        10930                  X9.0                  NaT
238         S14W56        10486                  X8.3 2003-11-02 17:30:00
290         N14W61        10720                  X7.1 2005-01-20 06:54:00
360         N17W69        11263                  X6.9 2011-08-09 08:12:00
333         S05E64        10930                  X6.5                  NaT
319         S12E67        10808                  X6.2 2005-09-09 19:48:00
83          N22W07         9077                  X5.7 2000-07-14 10:54:00
123         S21E31         9415                  X5.6 2001-04-06 19:30:00
376         N17E27        11429                  X5.4 2012-03-07 00:24:00
137         S17E34         9591                  X5.3 2001-08-25 16:50:00
444         S13E82        11990                  X4.9 2014-02-25 01:25:00
195         S13E72        10039                  X4.8 2002-07-23 00:42:00
106         N18W38         9236                  X4.0 2000-11-26 17:06:00
240         N08W77        10488                  X3.9 2003-11-03 10:06:00
289         N15W25        10720                  X3.8 2005-01-17 09:54:00
223         S06W21        10365                  X3.6 2003-05-28 00:50:00
334         S06W23        10930                  X3.4 2006-12-13 02:54:00
162         S26E90         9756                  X3.4 2001-12-28 20:30:00
194          SE90b        10039                  X3.3 2002-07-20 22:06:00
```

405	N08E77	11748	X3.2	2013-05-14 01:25:00
202	S02W81	10069	X3.1	2002-08-24 01:27:00
404	N11E85	11748	X2.8	2013-05-13 16:07:00
239	N10W83	10488	X2.7	2003-11-03 01:59:00
19	S11W65	8210	X2.7	1998-05-06 08:29:00
144	S16E23	9632	X2.6	2001-09-24 10:30:00
286	N15W05	10720	X2.6	2005-01-15 23:06:00
9	N17E63	8113	X2.6	1997-11-27 13:56:00
278	N09W49	10696	X2.5	2004-11-10 02:26:00
73	N20E18	9026	X2.3	2000-06-06 15:54:00
101	N22W07	9236	X2.3	2000-11-24 15:30:00
125	S23W09	9415	X2.3	2001-04-10 05:30:00
347	S20W12	11158	X2.2	2011-02-15 02:24:00
362	N14W18	11283	X2.1	2011-09-06 23:05:00
421	S06E69	11882	X2.1	2013-10-25 15:12:00
320	S13E47	10808	X2.1	2005-09-10 21:52:00
7	S14W33	8100	X2.1	1997-11-04 06:10:00
276	N09W17	10696	X2.0	2004-11-07 16:54:00
287	N15W25	10720	X2.0	2005-01-17 09:30:00
127	S19W43	9415	X2.0	2001-04-12 10:31:00
100	N20W05	9236	X2.0	2000-11-24 05:30:00
104	N20W23	9236	X1.9	2000-11-25 19:31:00
191	N19W30	10030	X1.8	2002-07-18 08:06:00
102	N21W14	9236	X1.8	2000-11-24 22:06:00
49	N11E32	8731	X1.8	1999-10-14 09:26:00

	Cme_Angle	Cme_Width	Cme_Speed	Is_Halo	Width_Lower_Bound
242	NaN	360	2657	True	False
119	261	244	2505	False	False
234	NaN	360	2459	True	False
128	245	167	1199	False	False
235	NaN	360	2029	True	False
8	NaN	360	1556	True	False
330	NaN	NaN	NaN	False	False
238	NaN	360	2598	True	False
290	NaN	360	882	True	False
360	NaN	360	1610	True	False
333	NaN	NaN	NaN	False	False
319	NaN	360	2257	True	False
83	NaN	360	1674	True	False
123	NaN	360	1270	True	False
376	NaN	360	2684	True	False
137	NaN	360	1433	True	False
444	NaN	360	2147	True	False
195	NaN	360	2285	True	False
106	NaN	360	980	True	False
240	293	103	1420	False	False

289	NaN	360	2547	True	False
223	NaN	360	1366	True	False
334	NaN	360	1774	True	False
162	NaN	360	2216	True	False
194	NaN	360	1941	True	False
405	NaN	360	2625	True	False
202	NaN	360	1913	True	False
404	NaN	360	1850	True	False
239	304	65	827	False	False
19	309	190	1099	False	False
144	NaN	360	2402	True	False
286	NaN	360	2861	True	False
9	98	91	441	False	False
278	NaN	360	3387	True	False
73	NaN	360	1119	True	False
101	NaN	360	1245	True	False
125	NaN	360	2411	True	False
347	NaN	360	669	True	False
362	NaN	360	575	True	False
421	NaN	360	1081	True	False
320	NaN	360	1893	True	False
7	NaN	360	785	True	False
276	NaN	360	1759	True	False
287	NaN	360	2094	True	False
127	NaN	360	1184	True	False
100	NaN	360	1289	True	False
104	NaN	360	671	True	False
191	NaN	360	1099	True	False
102	NaN	360	1005	True	False
49	NaN	360	1250	True	False

5.1.3 Explanation

As shown by the data, the top 50 solar flares from the NASA data share SOME resembelance to the SpaceWeatherLive.com data, however, they are not exactly the same. Some of the solar flares listed by SpaceWeatherLive.com are not in the NASA data and vice versa. The replication is very close though, save for a few missing classifications. |

5.2 Question 2: Integration

For each of the top 50 solar flares in the SpaceWeatherLive data, find the best matching row from the NASA data. Here, you have to decide for yourself how you determine what "best matching" means in this context (you will have to justify your approach!) Multiple flares may match to the same row from the NASA data, depending on your chosen method, you will be expected to notice this if it occurs.

In your submission, include an explanation of how you are defining best matching rows across the two datasets in addition to the code used to find the best matches. Finally, use your function to

add a new column to the NASA dataset indicating its rank according to SpaceWeatherLive, if it appears in that dataset. If more than one SpaceWeatherLive entry "best matches", choose one and explain how you chose.

5.2.1 Solution

[15]:
```
result = pd.merge(NASAdf, spaceWeatherdf, left_on=['Flare_Classification',␣
↪'Flare_Region'], right_on=['X_classification', 'Region'], how='inner')
#Append the rank to the NASA dataframe
NASAdfRanks = result
for i in range(5):
        del NASAdfRanks[NASAdfRanks.columns.values[-1]]
#Adjust column names for the merge
NASAdfRanks = NASAdfRanks.rename(columns={"Start_Datetime_x": "Start_Datetime",␣
↪"End_Datetime_x": "End_Datetime"})
NASAdfRanks['Rank'] = NASAdfRanks['Rank'].apply(int)
NASAdfRanks.sort_values(by=['Rank']) #Prints the matching rows
```

[15]:

	Start_Datetime	End_Datetime	Start_Frequency	End_Frequency	\
15	2003-11-04 20:00:00	2003-11-04 00:00:00	10000	200	
4	2001-04-02 22:05:00	2001-04-03 02:30:00	14000	250	
0	1997-11-06 12:20:00	1997-11-07 08:30:00	14000	100	
20	2006-12-05 10:50:00	2006-12-05 20:00:00	14000	250	
12	2003-11-02 17:30:00	2003-11-03 01:00:00	12000	250	
18	2005-01-20 07:15:00	2005-01-20 16:30:00	14000	25	
21	2006-12-06 19:00:00	2006-12-08 00:00:00	16000	30	
19	2005-09-09 19:45:00	2005-09-09 22:00:00	10000	50	
2	2000-07-14 10:30:00	2000-07-15 14:30:00	14000	80	
5	2001-04-06 19:35:00	2001-04-07 01:50:00	14000	230	
6	2001-08-25 16:50:00	2001-08-25 23:00:00	8000	170	
9	2002-07-23 00:50:00	2002-07-23 04:00:00	11000	400	
3	2000-11-26 17:00:00	2000-11-26 17:15:00	14000	7000	
14	2003-11-03 10:00:00	2003-11-03 12:30:00	6000	400	
17	2005-01-17 10:00:00	2005-01-17 10:35:00	6100	1500	
11	2003-05-28 01:00:00	2003-05-29 00:30:00	1000	200	
22	2006-12-13 02:45:00	2006-12-13 10:40:00	12000	150	
8	2002-07-20 21:30:00	2002-07-20 22:20:00	10000	2000	
10	2002-08-24 01:45:00	2002-08-24 03:25:00	5000	400	
13	2003-11-03 01:15:00	2003-11-03 01:25:00	3000	1500	
1	1998-05-06 08:25:00	1998-05-06 08:35:00	14000	5000	
16	2005-01-15 23:00:00	2005-01-15 00:00:00	3000	40	
7	2001-09-24 10:45:00	2001-09-25 20:00:00	7000	30	

	Flare_Location	Flare_Region	Flare_Classification	Cme_Datetime	\
15	S19W83	10486	X28.	2003-11-04 19:54:00	
4	N19W72	9393	X20.	2001-04-02 22:06:00	
0	S18W63	8100	X9.4	1997-11-06 12:10:00	

20	S07E68	10930	X9.0	NaT
12	S14W56	10486	X8.3	2003-11-02 17:30:00
18	N14W61	10720	X7.1	2005-01-20 06:54:00
21	S05E64	10930	X6.5	NaT
19	S12E67	10808	X6.2	2005-09-09 19:48:00
2	N22W07	9077	X5.7	2000-07-14 10:54:00
5	S21E31	9415	X5.6	2001-04-06 19:30:00
6	S17E34	9591	X5.3	2001-08-25 16:50:00
9	S13E72	10039	X4.8	2002-07-23 00:42:00
3	N18W38	9236	X4.0	2000-11-26 17:06:00
14	N08W77	10488	X3.9	2003-11-03 10:06:00
17	N15W25	10720	X3.8	2005-01-17 09:54:00
11	S06W21	10365	X3.6	2003-05-28 00:50:00
22	S06W23	10930	X3.4	2006-12-13 02:54:00
8	SE90b	10039	X3.3	2002-07-20 22:06:00
10	S02W81	10069	X3.1	2002-08-24 01:27:00
13	N10W83	10488	X2.7	2003-11-03 01:59:00
1	S11W65	8210	X2.7	1998-05-06 08:29:00
16	N15W05	10720	X2.6	2005-01-15 23:06:00
7	S16E23	9632	X2.6	2001-09-24 10:30:00

	Cme_Angle	Cme_Width	Cme_Speed	Is_Halo	Width_Lower_Bound	Rank
15	NaN	360	2657	True	False	1
4	261	244	2505	False	False	2
0	NaN	360	1556	True	False	7
20	NaN	NaN	NaN	False	False	9
12	NaN	360	2598	True	False	10
18	NaN	360	882	True	False	12
21	NaN	NaN	NaN	False	False	14
19	NaN	360	2257	True	False	15
2	NaN	360	1674	True	False	17
5	NaN	360	1270	True	False	18
6	NaN	360	1433	True	False	22
9	NaN	360	2285	True	False	25
3	NaN	360	980	True	False	26
14	293	103	1420	False	False	27
17	NaN	360	2547	True	False	29
11	NaN	360	1366	True	False	33
22	NaN	360	1774	True	False	34
8	NaN	360	1941	True	False	37
10	NaN	360	1913	True	False	41
13	304	65	827	False	False	47
1	309	190	1099	False	False	48
16	NaN	360	2861	True	False	49
7	NaN	360	2402	True	False	50

The results above show the matching rows between the two dataframes based on classification AND

region, sorted by rank in the SpaceWeatherLive data

[16]:
```
merged_NASAdf = NASAdf.merge(NASAdfRanks, how = 'left', on = ['Start_Datetime',␣
↪'End_Datetime', 'Start_Frequency', 'End_Frequency', 'Cme_Datetime',␣
↪'Cme_Angle', 'Cme_Speed', 'Cme_Width', 'Flare_Location', 'Flare_Region',␣
↪'Flare_Classification', 'Is_Halo', 'Width_Lower_Bound'])

merged_NASAdf
```

[16]:

	Start_Datetime	End_Datetime	Start_Frequency	End_Frequency
0	1997-04-01 14:00:00	1997-04-01 14:15:00	8000	4000
1	1997-04-07 14:30:00	1997-04-07 17:30:00	11000	1000
2	1997-05-12 05:15:00	1997-05-14 16:00:00	12000	80
3	1997-05-21 20:20:00	1997-05-21 22:00:00	5000	500
4	1997-09-23 21:53:00	1997-09-23 22:16:00	6000	2000
..	…	…	…	…
477	2014-12-13 14:27:00	2014-12-13 14:51:00	14000	3900
478	2014-12-17 04:09:00	2014-12-17 04:19:00	2900	2100
479	2014-12-17 05:00:00	2014-12-17 05:09:00	14000	11500
480	2014-12-18 22:31:00	2014-12-18 22:54:00	5100	1300
481	2014-12-21 12:05:00	2014-12-21 12:28:00	14000	7400

	Flare_Location	Flare_Region	Flare_Classification	Cme_Datetime
0	S25E16	8026	M1.3	1997-04-01 15:18:00
1	S28E19	8027	C6.8	1997-04-07 14:27:00
2	N21W08	8038	C1.3	1997-05-12 05:30:00
3	N05W12	8040	M1.3	1997-05-21 21:00:00
4	S29E25	8088	C1.4	1997-09-23 22:02:00
..	…	…	…	…
477	W90b	NaN	NaN	2014-12-13 14:24:00
478	S11E33	12241	M1.1	2014-12-17 02:00:00
479	S20E09	12242	M8.7	2014-12-17 05:00:00
480	S11E15	12241	M6.9	2014-12-19 01:04:00
481	S14W25	12241	M1.0	2014-12-21 12:12:00

	Cme_Angle	Cme_Width	Cme_Speed	Is_Halo	Width_Lower_Bound	Rank
0	74	79	312	False	False	NaN
1	NaN	360	878	True	False	NaN
2	NaN	360	464	True	False	NaN
3	263	165	296	False	False	NaN
4	133	155	712	False	False	NaN
..	…	…	…	…	…	…
477	NaN	360	2222	True	False	NaN
478	107	108	869	False	False	NaN
479	NaN	360	587	True	False	NaN
480	NaN	360	1195	True	False	NaN
481	NaN	360	669	True	False	NaN

```
[482 rows x 14 columns]
```

and above is shown the merged NASA dataframe with the ranks.

5.2.2 Explanation

I obtained my results by matching based on BOTH region and classification. This first required me to clean up the data from SpaceWeatherLive.com because they had an inaccuracy in labeling regions that started with a 1. I believe this is because the column seems to be limited to 4 characters. I decided these were the best columns to match on because they have no duplicates I have to deal with. While they are not individually unique, they are unique together. I considered using datetimes, but NONE of the datetimes matched up well between the two dataframes.

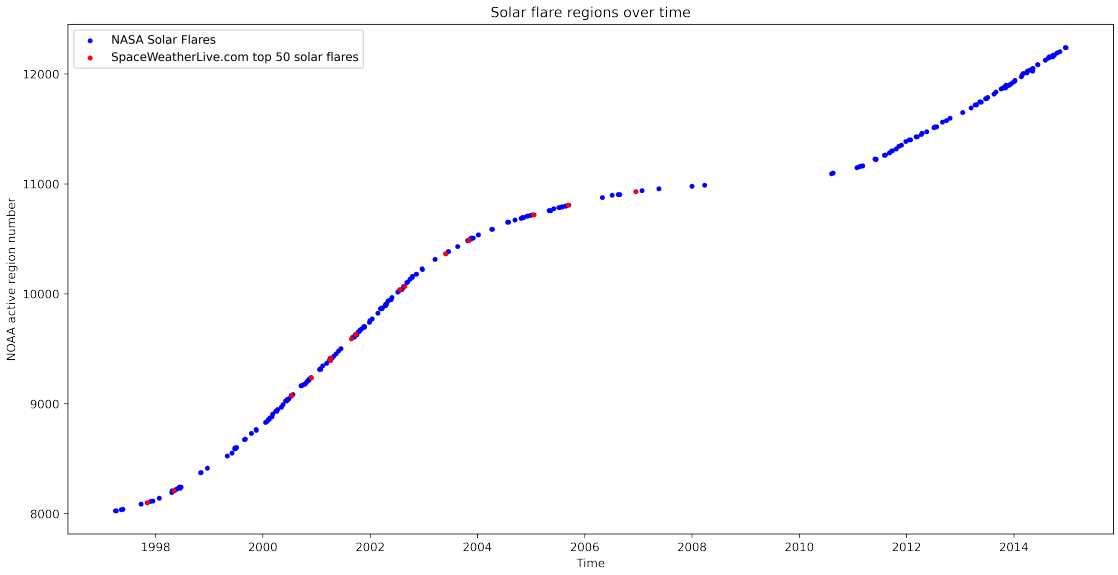
5.3 Question 3: Analysis

Prepare one plot that shows the top 50 solar flares in context with all data available in the NASA dataset. Here are some possibilities (you can do something else)

1. Plot attributes in the NASA dataset (e.g., starting or ending frequenciues, flare height or width) over time. Use graphical elements (e.g., text or points) to indicate flares in the top 50 classification.

6 Solution, plotting CME Speeds over time using PyPlot

[17]:
```
merged_NASAdf = merged_NASAdf.sort_values('Cme_Datetime', ascending=True)
NASAdfRanks = NASAdfRanks.sort_values('Cme_Datetime', ascending=True)
merged_NASAdf['Flare_Region'] = merged_NASAdf['Flare_Region'].apply(float)
NASAdfRanks['Flare_Region'] = NASAdfRanks['Flare_Region'].apply(float)
#Higher resolution so the graph is more readable
plt.rcParams['figure.figsize'] = [16, 8]
plt.rcParams['figure.dpi'] = 500
ax = pylab.subplot(111)
ax.scatter(merged_NASAdf['Cme_Datetime'], merged_NASAdf['Flare_Region'], s=10,␣
 ↪c='b', label='NASA Solar Flares')
ax.scatter(NASAdfRanks['Cme_Datetime'], NASAdfRanks['Flare_Region'],  s=10,␣
 ↪c='r', label='SpaceWeatherLive.com top 50 solar flares')
plt.title("Solar flare regions over time")
plt.xlabel("Time")
plt.ylabel("NOAA active region number")
plt.legend(loc='upper left');
ax.figure.show()
```



(a) a short description (2 sentences) of what the intent of your plot is

The intention of this plot is to show that there is a correlation between solar flare regions and time; solar flare regions have been moving consistently for the past ~20 years. The plot has a very low variance and covariance, strengthening the correlation between the two variables. The plot also shows that powerful solar flares (from the top 50) occurred around the same regions (close together).

(b) code to produce your plot

See above

(c) a short text description of your plot

The plot above shows the NOAA active region number of solar flares over time. Points in blue indicate solar flares listed on http://cdaw.gsfc.nasa.gov/CME_list/radio/waves_type2.html while points in orange/red incidcate top 50 solar flares listed on SpaceWeatherLive.com that matched data on the NASA page.

(d) a sentence or two of interpretation of your plot (again think of variation, co-variation, etc.).

Once again, the graph has a lot of variance since active regions are very rarely the same for any two solar flares. The covariance, on the other hand, is very low between regions and time because the two variables are dependent (time has a large effect on CME regions), which is what the graph intended to show. The sun is an active reaction and therefore, activity is always moving around it, so it makes sense that regions are moving over time instead of being still.