ER190C: Data, Envinronment and Society.

Lecture 2: September 3, 2019

In this notebook, we'll do a brief tour of the data set we'll be working with from the California Independent System Operator.

In [1]:

```
import requests # this is a really useful library for pulling data from the web
import csv # this helps us work with csv files
import numpy as np # numpy is something like a matlab replacement for python.
 Numeric and scientific computing.
import pandas as pd # we'll learn more about this soon
```

California ISO is the system operator for the California grid. They tell generators when and how much to produce.

They record renewable production data here (http://content.caiso.com/green/renewrpt/)

That page links to files giving production for the *day* in question.

Let's look at Aug 21, 2017, the day you'll explore in the HW

In [2]:

```
# figure out what the url should be and enter it here:
url = 'http://content.caiso.com/green/renewrpt/20170821_DailyRenewablesWatch.tx
t' # do this in lecture
```

Let's "tab into" `requests` to see how we can get data from the url.

Some cool 'help' features of Jupyter

1. pushing tab at the right time shows you what methods are available to apply to an object.
2. pushing shift-tab repeatedly gives you help files
3. typing a question mark before a command pulls up the full help file.

In [3]:

```
caiso_data = requests.get(url) # do this in lecture
```

In [4]:

```
?requests.get # do this in lecture
```

In [5]:

```
# let's see what we got
caiso_data
```

Out[5]:

```
<Response [200]>
```

'Response' is the object returned by requests. In this case we've opened a connection to the url but we haven't actually grabbed the text.

Let's look at the requests documentation to figure out what to do. (Search for python requests in your favorite search engine and see what you find.)

Looks like we can tack .text on the end of the object to actually pull the data.

In [6]:

```
caiso_data.text # do this in lecture
```

Out[6]:

'08/21/17\t\t\tHourly Breakdown of Renewable Resources (MW)\t\t\t\t
\t\t\t\t\t\t\t\t\t\t\r\n\tHour\t\tGEOTHERMAL\tBIOMASS\t\tBIOGAS\t\tS
MALL HYDRO\tWIND TOTAL\tSOLAR PV\tSOLAR THERMAL\t\t\t\t\t\t\r\n\t1\t
\t971\t\t245\t\t164\t\t406\t\t2032\t\t0\t\t0\t\t\r\n\t2\t\t971\t\t24
6\t\t174\t\t408\t\t2056\t\t0\t\t0\t\t\r\n\t3\t\t971\t\t248\t\t175\t
\t406\t\t1912\t\t0\t\t0\t\t\r\n\t4\t\t972\t\t250\t\t175\t\t415\t\t18
08\t\t0\t\t0\t\t\r\n\t5\t\t972\t\t251\t\t175\t\t405\t\t1771\t\t0\t\t
0\t\t\r\n\t6\t\t973\t\t250\t\t175\t\t412\t\t1681\t\t0\t\t0\t\t\r\n\t
7\t\t972\t\t254\t\t175\t\t441\t\t1560\t\t174\t\t0\t\t\r\n\t8\t\t971
\t\t256\t\t174\t\t442\t\t1559\t\t2376\t\t0\t\t\r\n\t9\t\t969\t\t259
\t\t174\t\t495\t\t1582\t\t5484\t\t86\t\t\r\n\t10\t\t965\t\t258\t\t16
9\t\t463\t\t1534\t\t5312\t\t141\t\t\r\n\t11\t\t945\t\t260\t\t172\t\t
451\t\t1372\t\t3503\t\t110\t\t\r\n\t12\t\t939\t\t259\t\t176\t\t457\t
\t1297\t\t6329\t\t201\t\t\r\n\t13\t\t936\t\t256\t\t176\t\t460\t\t130
8\t\t8700\t\t472\t\t\r\n\t14\t\t936\t\t257\t\t176\t\t468\t\t1496\t\t
8857\t\t527\t\t\r\n\t15\t\t936\t\t260\t\t177\t\t471\t\t1584\t\t8732
\t\t598\t\t\r\n\t16\t\t936\t\t262\t\t175\t\t501\t\t1620\t\t7983\t\t3
95\t\t\r\n\t17\t\t936\t\t260\t\t176\t\t486\t\t1649\t\t7020\t\t262\t
\t\r\n\t18\t\t935\t\t258\t\t174\t\t528\t\t1580\t\t5329\t\t228\t\t\r
\n\t19\t\t955\t\t258\t\t172\t\t610\t\t1644\t\t2193\t\t119\t\t\r\n\t2
0\t\t964\t\t255\t\t172\t\t604\t\t1845\t\t114\t\t0\t\t\r\n\t21\t\t967
\t\t247\t\t171\t\t583\t\t2358\t\t0\t\t0\t\t\r\n\t22\t\t970\t\t235\t
\t171\t\t484\t\t2282\t\t0\t\t0\t\t\r\n\t23\t\t970\t\t230\t\t171\t\t4
36\t\t1840\t\t0\t\t0\t\t\r\n\t24\t\t971\t\t228\t\t172\t\t423\t\t1620
\t\t0\t\t0\t\t\r\n\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\r\n\t\t\t\t\t\t
\t\t\t\t\t\t\t\t\t\t\t\r\n\t\t\tHourly Breakdown of Total Production
by Resource Type (MW)\t\t\t\t\t\t\t\t\t\t\t\t\t\t\r\n\tHour\t\tRENEW
ABLES\tNUCLEAR\t\tTHERMAL\t\tIMPORTS\t\tHYDRO\t\t\t\t\t\t\r\n\t1\t
\t3582\t\t2259\t\t8401\t\t6163\t\t3916\t\t\t\t\t\t\r\n\t2\t\t3855\t
\t2260\t\t7597\t\t6016\t\t3465\t\t\t\t\t\t\r\n\t3\t\t3712\t\t2262\t
\t7843\t\t5518\t\t3166\t\t\t\t\t\t\r\n\t4\t\t3620\t\t2262\t\t8039\t
\t5221\t\t3132\t\t\t\t\t\t\r\n\t5\t\t3574\t\t2260\t\t8178\t\t5363\t
\t3454\t\t\t\t\t\t\r\n\t6\t\t3491\t\t2259\t\t8668\t\t6062\t\t3754\t
\t\t\t\t\t\r\n\t7\t\t3585\t\t2263\t\t9654\t\t6903\t\t3872\t\t\t\t\t
\t\r\n\t8\t\t5781\t\t2265\t\t9935\t\t6070\t\t3628\t\t\t\t\t\t\r\n\t9
\t\t9050\t\t2265\t\t8834\t\t5107\t\t3306\t\t\t\t\t\t\r\n\t10\t\t8842
\t\t2264\t\t9391\t\t5684\t\t3448\t\t\t\t\t\t\r\n\t11\t\t6813\t\t2260
\t\t10239\t\t7475\t\t3926\t\t\t\t\t\t\r\n\t12\t\t9659\t\t2259\t\t956
3\t\t4812\t\t3270\t\t\t\t\t\t\r\n\t13\t\t12309\t\t2259\t\t8558\t\t33
76\t\t3676\t\t\t\t\t\t\r\n\t14\t\t12717\t\t2258\t\t8686\t\t3795\t\t4
014\t\t\t\t\t\t\r\n\t15\t\t12758\t\t2258\t\t9694\t\t3944\t\t4204\t\t
\t\t\t\t\r\n\t16\t\t11872\t\t2255\t\t10764\t\t4889\t\t4475\t\t\t\t\t
\t\r\n\t17\t\t10789\t\t2252\t\t11271\t\t6032\t\t5018\t\t\t\t\t\t\r\n
\t18\t\t9032\t\t2251\t\t12269\t\t6944\t\t5360\t\t\t\t\t\t\r\n\t19\t
\t5951\t\t2256\t\t13060\t\t8713\t\t5447\t\t\t\t\t\t\r\n\t20\t\t3955
\t\t2260\t\t13525\t\t9581\t\t5494\t\t\t\t\t\t\r\n\t21\t\t4326\t\t226
3\t\t12865\t\t9845\t\t5366\t\t\t\t\t\t\r\n\t22\t\t4141\t\t2263\t\t11
939\t\t9009\t\t5280\t\t\t\t\t\t\r\n\t23\t\t3647\t\t2258\t\t10938\t\t
8271\t\t4554\t\t\t\t\t\t\r\n\t24\t\t3414\t\t2256\t\t10125\t\t7268\t
\t4015\t\t\t\t\t\t\r\n'

Ack! That's pretty ugly! What are we looking at?

(a tab delimited file)

I wrote a function that will pull a date range and massage it into the form we want:

In [13]:

```
import datetime # helps us to work with dates and times in different formats
import os # helps us talk to the operating system command line
def CAISOrenewables(year, month, start_day, end_day, production = False, matrix
= False):
    """Scrape CAISO's daily renewable watch .txt files and
    convert to a DataFrame or Numpy record array. Will only scrape
    a range of days in a given month.

    Keyword arguments:
    Year -- year of the date to scrape
    Month -- Month of date to scrape
    start_day -- starting day of month to scrape
    end_day -- ending day to scrape
    production -- If False, will collect hourly breakdown of renewable resource
s.
                  If True, will scrape hourly breakdown of total production by
 resource type.
    matrix -- If False, function will return a Pandas DataFrame
              If True, will return numpy recarray
    """
    base_url = 'http://content.caiso.com/green/renewrpt/'
    tail = '_DailyRenewablesWatch.txt'

    rv = pd.DataFrame()

    for day in range(start_day, end_day + 1):
        #format date and URL to pull
        if month < 10:
            str_month = '0' + str(month)
        else:
            str_month = str(month)
        if day < 10:
            str_day = '0'+ str(day)
        else:
            str_day = str(day)

        str_m_day = str_month + str_day
        url = base_url + str(year) + str_m_day + tail

        #Write scraped file to drive
        caiso_data = requests.get(url).text
        txt_filename = str(year) +str_m_day + '.txt'
        csv_filename = str(year) + str_m_day + '.csv'

        with open(txt_filename, 'w') as f:
            f.write(str(caiso_data))

        #Convert the .txt file to a csv.
        with open(txt_filename) as txtfile, open(csv_filename,'w') as new_csv:
            for line in txtfile:
                new_csv.write(line.replace('\t',','))

        #Get day of year for dataframe index
```

```
        date = datetime.date(year, month, day)

        #Load data to dataframe.
        data = pd.read_csv(csv_filename, delimiter='\t')

        if not production:
            data = data.iloc[range(0, 25)]
        else:
            data = data.iloc[range(28, 53)].reset_index(drop=True)

        #Get column names
        columns = [i for i in np.array2string(data.iloc[0].values).split(',') i
f len(i)>3]

        #Grab first row of data to put in a dictionary then append the rest.
        first_row = [[int(i)] for i in np.array2string(data.iloc[1].values).spl
it(',') if i.isdigit()]
        df_data = dict(zip(columns, first_row))

        #Do the same for the rest of the rows
        for row in range(2, data.shape[0]):
            vals = [int(i) for i in np.array2string(data.iloc[row].values).spli
t(',') if i.isdigit()]
            for item in range(len(columns)):
                df_data[columns[item]].append(vals[item])

        #create DataFrame with collected data
        d_df = pd.DataFrame(df_data, [date]*24)[columns]
        rv = rv.append(d_df)

        os.remove(txt_filename)
        os.remove(csv_filename)

    if matrix:
        return rv.to_records(index=True)

    return rv
```

Ok, now we can pull whatever data we want for renewables production from the CAISO website.

Here we'll pull CAISO renewables data for August 20 through 22, 2017.

In [14]:

```
caiso_data = CAISOrenewables(2017, 8, 20, 22) # do this in lecture
```

In [15]:

```
caiso_data # this shows the data frame
```

Out[15]:

	Hour	GEOTHERMAL	BIOMASS	BIOGAS	SMALL HYDRO	WIND TOTAL	SOLAR PV	SOLAR THERMAL
2017-08-20	1	970	248	175	427	2792	0	0
2017-08-20	2	971	247	175	441	2717	0	0
2017-08-20	3	970	248	176	407	2487	0	0
2017-08-20	4	969	248	177	364	2166	0	0
2017-08-20	5	969	247	176	374	2079	0	0
2017-08-20	6	970	248	176	396	1817	0	0
2017-08-20	7	971	251	176	398	1625	200	0
2017-08-20	8	969	251	176	390	1484	2753	168
2017-08-20	9	965	249	177	390	1309	6160	414
2017-08-20	10	962	248	176	391	1015	7914	535
2017-08-20	11	963	245	177	394	960	8723	565
2017-08-20	12	963	250	176	391	935	9053	565
2017-08-20	13	961	249	177	436	990	9037	565
2017-08-20	14	961	245	177	452	1130	8995	505
2017-08-20	15	960	244	177	483	1186	8775	549
2017-08-20	16	959	253	177	494	1128	8341	339
2017-08-20	17	959	254	176	482	1625	7519	213

	Hour	GEOTHERMAL	BIOMASS	BIOGAS	SMALL HYDRO	WIND TOTAL	SOLAR PV	SOLAR THERMAL
2017-08-20	18	960	254	176	486	1899	5616	185
2017-08-20	19	961	254	177	616	2212	2201	88
2017-08-20	20	963	257	175	631	2350	140	0
2017-08-20	21	964	254	160	612	2263	0	0
2017-08-20	22	966	253	157	511	2388	0	0
2017-08-20	23	969	250	156	478	2187	0	0
2017-08-20	24	970	249	156	463	2087	0	0
2017-08-21	1	971	245	164	406	2032	0	0
2017-08-21	2	971	246	174	408	2056	0	0
2017-08-21	3	971	248	175	406	1912	0	0
2017-08-21	4	972	250	175	415	1808	0	0
2017-08-21	5	972	251	175	405	1771	0	0
2017-08-21	6	973	250	175	412	1681	0	0
...	...	...	...	...	...	...	...	...
2017-08-21	19	955	258	172	610	1644	2193	119
2017-08-21	20	964	255	172	604	1845	114	0
2017-08-21	21	967	247	171	583	2358	0	0
2017-08-21	22	970	235	171	484	2282	0	0

	Hour	GEOTHERMAL	BIOMASS	BIOGAS	SMALL HYDRO	WIND TOTAL	SOLAR PV	SOLAR THERMAL
2017-08-21	23	970	230	171	436	1840	0	0
2017-08-21	24	971	228	172	423	1620	0	0
2017-08-22	1	972	228	172	368	1445	0	0
2017-08-22	2	972	225	172	382	1338	0	0
2017-08-22	3	972	228	172	393	1219	0	0
2017-08-22	4	972	228	172	389	1188	0	0
2017-08-22	5	972	226	172	384	1065	0	0
2017-08-22	6	972	227	173	373	944	0	0
2017-08-22	7	971	230	171	379	853	197	0
2017-08-22	8	968	233	165	423	659	2792	101
2017-08-22	9	965	235	166	428	579	6032	430
2017-08-22	10	961	235	166	400	382	7673	555
2017-08-22	11	939	237	165	411	343	8599	580
2017-08-22	12	935	236	167	434	336	8822	627
2017-08-22	13	931	237	168	451	382	8965	634
2017-08-22	14	933	238	165	463	608	8963	479
2017-08-22	15	934	238	166	465	487	8746	481
2017-08-22	16	934	239	169	489	598	8182	513

	Hour	GEOTHERMAL	BIOMASS	BIOGAS	SMALL HYDRO	WIND TOTAL	SOLAR PV	SOLAR THERMAL
2017-08-22	17	936	238	173	539	755	7451	467
2017-08-22	18	935	235	175	552	826	5630	361
2017-08-22	19	914	237	176	604	1357	2182	157
2017-08-22	20	957	235	176	610	1666	167	0
2017-08-22	21	965	233	177	622	2017	19	0
2017-08-22	22	967	236	176	502	2096	21	0
2017-08-22	23	970	227	176	451	1903	0	0
2017-08-22	24	1087	220	177	448	1820	0	0

72 rows × 8 columns

Now let's use the `.loc` method in pandas to look at an individual data column (more on pandas next time)

In [ ]:

```
caiso_data.loc[:,'SOLAR PV'] #do this in lecture
```

In [ ]:

```
import matplotlib.pyplot as plt # this gives us libraries to plot nice figures.
```

Let's plot the solar generation data using `plt.plot` and the `.loc` method

In [ ]:

```
plt.plot(caiso_data.loc[:,'SOLAR PV']) # do this in lecture
```

The problem is that the "index" of the data frame is clustered at the same value for each day -- so the data get plotted just at one location for each day.

Let's fix the index with a list comprehension. Replace the current indexes with [1, 2, 3, ...]

In [ ]:

```
caiso_data.index = [i for i in range(0,len(caiso_data.index))] # do this in lec
ture
```

In [ ]:

```
# Now we can plot according to unique indexes:
plt.plot(caiso_data.loc[:,'SOLAR PV']) # do this in lecture
```

In [ ]:

```
# alternatively we can plot by hour of day to see things overlap
plt.plot(caiso_data.loc[:,'Hour'],caiso_data.loc[:,'SOLAR PV']) # do this in le
cture
```