Lecture 05.2 EDA

September 7, 2023

1 ER131: Data Cleaning and Exploratory Data Analysis

Duncan Callaway

In this notebook we'll work with PurpleAir data to explore the concepts of Structure, Granularity, Scope, Temporality and Faithfulness. Along the way we'll talk about data cleaning as well.

Here's PurpleAir's website – They have really cool maps!

The way I developed this lecture was by pulling the data down and exploring it. You'll see my (edited) process of examining the data.

This began by me visiting this website to look for data. I used the Chrome browser to pull data (other browsers didn't work).

The folks are PurpleAir also sent me a pdf describing their data, which is available from the instructors.

[1]:
```
import numpy as np
import pandas as pd
import os
```

1.1 Structure: how are the data stored?

First let's look at what's in the data directory using `os.listdir` (remember this is a set of command line-style commands that work across platforms, i.e. mac, linux, windows)

[2]:
```
os.listdir()
```

[2]:
```
['~$05 Pivot.pptx',
 '.DS_Store',
 'CAISO_2017to2018_stack.csv',
 'Lecture 05.1 Pivot.pdf',
 'Lecture 05.2 EDA.ipynb',
 '.ipynb_checkpoints',
 'US-EPA-PM2.5-AQI-Monitoring.png',
 'data',
 '05 Pivot.pptx',
 '05-06 Data Cleaning, EDA.pptx',
 '~$05-06 Data Cleaning, EDA.pptx',
 'Lecture 05.1 Pivot.ipynb']
```

[3]:
```
!ls # this does the same thing
```

```
05 Pivot.pptx                       Lecture 05.2 EDA.ipynb
05-06 Data Cleaning, EDA.pptx       US-EPA-PM2.5-AQI-Monitoring.png
CAISO_2017to2018_stack.csv          data
Lecture 05.1 Pivot.ipynb            ~$05 Pivot.pptx
Lecture 05.1 Pivot.pdf              ~$05-06 Data Cleaning, EDA.pptx
```

Let's look in the data directory:

[4]:
```
os.listdir('data')
```

[4]:
```
['Alameda Gold Coast (outside) (37.767347 -122.267255) Primary Real Time
09_08_2021 09_07_2022.csv',
 'Bower House (outside) (37.803884 -122.297151) Primary Real Time 09_08_2021
09_07_2022.csv',
 'Backyard (outside) (37.826875 -122.245254) Primary Real Time 09_08_2021
09_07_2022.csv',
 'Moraga Ave (outside) (37.83023 -122.239963) Primary Real Time 09_08_2021
09_07_2022.csv',
 'manzanita at villanova (outside) (37.84099 -122.196456) Primary Real Time
09_08_2021 09_07_2022.csv',
 'B59-Mech (outside) (37.875921 -122.253082) Primary Real Time 09_08_2021
09_07_2022.csv']
```

1.1.1 Q: What can we learn from these file names?

- the sensor location appears to be provided in lat / lon coordinates in parens
- the date range is listed
- they are probably csv files.

If you type the lat-lon values into google maps, you'll find they correspond to the locations of purple air sensors with the same name. Here is a route through these sites.

Before proceeding let's find the size of some of these files:

[6]:
```
os.path.getsize('data/Bower House (outside) (37.803884 -122.297151) Primary␣
↪Real Time 09_08_2021 09_07_2022.csv')
```

[6]:
```
28354380
```

What are the units? Let's shift tab in to `getsize` to find out.

[9]:
```
os.path.getsize
```

[9]:
```
<function genericpath.getsize(filename)>
```

Not much information. Google search reveals this information page, which says the units are bytes.

[10]:
```
1e-6*os.path.getsize('data/Bower House (outside) (37.803884 -122.297151)␣
↪Primary Real Time 09_08_2021 09_07_2022.csv')
```

[10]: 28.35438

SO 28 Mb.

Let's read in one of the .csv files:

[11]:
```
Bower = pd.read_csv('data/Bower House (outside) (37.803884 -122.297151) Primary␣
↪Real Time 09_08_2021 09_07_2022.csv')
```

[12]:
```
Bower.head()
```

[12]:
```
                 created_at  entry_id  PM1.0_CF1_ug/m3  PM2.5_CF1_ug/m3  \
0  2021-09-08 00:00:30 UTC    271065            14.34            29.12
1  2021-09-08 00:02:30 UTC    271066            15.74            29.59
2  2021-09-08 00:04:30 UTC    271067            13.86            28.89
3  2021-09-08 00:06:30 UTC    271068            14.43            28.59
4  2021-09-08 00:08:30 UTC    271069            13.00            27.02

   PM10.0_CF1_ug/m3  UptimeMinutes  RSSI_dbm  Temperature_F  Humidity_%  \
0             38.14        40112.0     -71.0           84.0        38.0
1             37.28        40114.0     -70.0           84.0        38.0
2             41.70        40116.0     -75.0           84.0        37.0
3             45.93        40118.0     -76.0           84.0        37.0
4             41.47        40120.0     -70.0           84.0        36.0

   PM2.5_ATM_ug/m3  Unnamed: 10
0            28.73          NaN
1            29.15          NaN
2            28.64          NaN
3            28.41          NaN
4            27.02          NaN
```

1.1.2 Q: What do you notice about the file contents?

Several things to ask from this: 1. Dates are UTC. 2. Each entry has a unique ID – could be used to check for time stamp errors or gaps in data 3. Headers have 'CF1' or 'ATM' at the top – what does that mean? 1. From the PurpleAir documentation, in this directory, *"ATM is"atmospheric", meant to be used for outdoor applications. CF=1 is meant to be used for indoor or controlled environment applications. However, PurpleAir uses CF=1 values on the map. This value is lower than the ATM value in higher measured concentrations."*
2. The explanation is a little vague and suggests further exploration required! 3. This cool paper suggests that the ATM data are 'raw' measurements and that CF_1 data have a 3/2 multiplication at concentrations over 25 μ g / m^3 4. The columns "UptimeMinutes" and "RSSI_dbm" are not immediately obvious 1. again from documentation: "uptimeminutes" is time since last restart, and "RSSI_dbm" is wifi signal strength for the device.
5. The "unnamed: 10" column seems useless, why is it there? 1. Looking at the data we see a comma before the `\n` (newline character) at the end of the first (header) line, it appears this is generating the extra row.

[14]:
```
N = 2
airdat = "data/Bower House (outside) (37.803884 -122.297151) Primary Real Time␣
↪09_08_2021 09_07_2022.csv"
with open(airdat) as myfile:
    head = [next(myfile) for x in range(N)]
print(head)
```

```
['created_at,entry_id,PM1.0_CF1_ug/m3,PM2.5_CF1_ug/m3,PM10.0_CF1_ug/m3,UptimeMin
utes,RSSI_dbm,Temperature_F,Humidity_%,PM2.5_ATM_ug/m3,\n', '2021-09-08 00:00:30
UTC,271065,14.34,29.12,38.14,40112.00,-71.00,84.00,38.00,28.73\n']
```

[15]:
```
Bower.describe()
```

[15]:
```
            entry_id  PM1.0_CF1_ug/m3  PM2.5_CF1_ug/m3  PM10.0_CF1_ug/m3  \
count  351942.000000    351942.000000    351942.000000     351942.000000
mean   390592.636500         7.703851        15.989965         24.196788
std     68985.096555         9.894243        19.466634         24.491579
min    271065.000000         0.000000         0.000000          0.000000
25%    336116.000000         1.210000         3.590000          6.750000
50%    380108.500000         3.760000         8.750000         16.410000
75%    444398.750000        10.220000        20.280000         33.270000
max    532384.000000       336.360000      1297.910000       1582.450000

        UptimeMinutes       RSSI_dbm  Temperature_F     Humidity_%  \
count  351942.000000  351942.000000  351942.000000  351942.000000
mean    26747.629834     -70.180916      65.679018      50.249541
std     18879.817383       5.026611      11.876395      14.772064
min         1.000000     -93.000000      36.000000       5.000000
25%     10005.000000     -71.000000      58.000000      41.000000
50%     23995.500000     -69.000000      63.000000      55.000000
75%     41299.000000     -67.000000      71.000000      62.000000
max     71582.000000     -53.000000     126.000000      77.000000

       PM2.5_ATM_ug/m3  Unnamed: 10
count    351942.000000          0.0
mean         14.203361          NaN
std          14.610303          NaN
min           0.000000          NaN
25%           3.590000          NaN
50%           8.750000          NaN
75%          20.270000          NaN
max         865.120000          NaN
```

As you can see, at this location the average CF1 value is more than the EPA standards. As an aside, before we do more EDA, let's check the other location.

[17]:
```
backyard = pd.read_csv('data/Backyard (outside) (37.826875 -122.245254) Primary␣
↪Real Time 09_08_2021 09_07_2022.csv')
```

```
np.mean(backyard['PM2.5_CF1_ug/m3'])
```

[17]: 11.776786947061435

```
[18]: moraga = pd.read_csv('data/Moraga Ave (outside) (37.83023 -122.239963) Primary␣
    ↪Real Time 09_08_2021 09_07_2022.csv')
np.mean(moraga['PM2.5_CF1_ug/m3'])
```

[18]: 10.18030246986375

```
[19]: manzanita = pd.read_csv('data/manzanita at villanova (outside) (37.84099 -122.
    ↪196456) Primary Real Time 09_08_2021 09_07_2022.csv')
np.mean(manzanita['PM2.5_CF1_ug/m3'])
```

[19]: 8.031333700168869

```
[20]: alameda = pd.read_csv('data/Alameda Gold Coast (outside) (37.767347 -122.
    ↪267255) Primary Real Time 09_08_2021 09_07_2022.csv')
np.mean(alameda['PM2.5_CF1_ug/m3'])
```

[20]: 13.041692402454741

Now you can see that the mean PM2.5 numbers vary significantly by location.

If you inspect the data, you'll see a general trend: the further away from the Bay the sensor is, the lower its mean.

Let's dig in to one sensor a little more

1.2 Granularity: how are the data aggregated?

We'll talk a little more about Temporality in a moment, but time also matters for thinking about granularity.

First we need to pay attention to the fact that this is UTC. Let's put it in datetime format to prevent mistakes.

```
[21]: Bowertime = pd.to_datetime(Bower['created_at'], utc=True)
```

```
[22]: Bower['created_at']=Bowertime
```

```
[23]: Bower['created_at'].dtype
```

[23]: datetime64[ns, UTC]

Yes, that response really means the time are recorded down to the nanosecond.

Note: The data are instantaneous measurements, not averaged over time.
* So these data have granularity of nanoseconds! * In practice, this just means there is *no* aggregation in the primary data.

[24]:
```
Bower.head()
```

[24]:
```
                   created_at  entry_id  PM1.0_CF1_ug/m3  PM2.5_CF1_ug/m3  \
0 2021-09-08 00:00:30+00:00    271065            14.34            29.12
1 2021-09-08 00:02:30+00:00    271066            15.74            29.59
2 2021-09-08 00:04:30+00:00    271067            13.86            28.89
3 2021-09-08 00:06:30+00:00    271068            14.43            28.59
4 2021-09-08 00:08:30+00:00    271069            13.00            27.02

   PM10.0_CF1_ug/m3  UptimeMinutes  RSSI_dbm  Temperature_F  Humidity_%  \
0             38.14        40112.0     -71.0           84.0        38.0
1             37.28        40114.0     -70.0           84.0        38.0
2             41.70        40116.0     -75.0           84.0        37.0
3             45.93        40118.0     -76.0           84.0        37.0
4             41.47        40120.0     -70.0           84.0        36.0

   PM2.5_ATM_ug/m3  Unnamed: 10
0            28.73          NaN
1            29.15          NaN
2            28.64          NaN
3            28.41          NaN
4            27.02          NaN
```

Nice thing about the datetime formate is that you can easily get time information out of it. For example let's look at the 1,000th entry:

[38]:
```
Bower.iloc[1000,0].hour
```

[38]: 9

Note, we could rename the cols to make things easier if we wished. I'm not going to because we're not going to be workign with this data set for long, but in other cases you might decide to.

1.3 Scope: how much time, how many people, what spatial area?

So far we have focused on data from one location – A sensor in West Oakland.

From the file name it looks like the time is from the last 12 months, let's confirm:

[39]:
```
Bower['created_at'].min()
```

[39]: Timestamp('2021-09-08 00:00:30+0000', tz='UTC')

[40]:
```
Bower['created_at'].max()
```

[40]: Timestamp('2022-09-07 23:58:17+0000', tz='UTC')

So it's about one year of data.

Does the data cover the topic of interest?

In this case, we need to answer the question: For the PurpleAir data, what topic of interest might the data cover?

–> **class discussion on this.** Possible answers why the data might be of interest * near highways and port of oakland * near communities that are historically underserved

Possible reasons *not* of interest: * more important to look at many recent wildfire seasons * it might be valuable to compare across sites rather than evaluate just one.

1.4 Temporality: How is time represented in the data?

We've already figured out that we're working with UTC dates. UTC is "universal time coordinated" and is essentially greenwich mean time, the time on the prime meridian.

Can we figure out how frequent measurements are?

Unfortunately I found it difficult to take differences with datetime objects, so I had to write a for loop:

[77]:
```
diffs = np.zeros(len(Bower['created_at']))

for i in range(0, len(diffs)-1):
    diffs[i] = ((Bower['created_at'][i+1]
                  - Bower['created_at'][i]).total_seconds())  # we apply␣
 ↪total_seconds in order to store the data as a float in the list

diffs = np.sort((diffs))

print('max diffs:', diffs[:-30:-1])
print('median:', np.median(diffs))
```

```
max diffs: [79585. 43629.  3842.  1782.  1782.  1561.  1443.  1441.  1437.
976.
   976.   727.   724.   720.   718.   718.   495.   495.   480.   480.
   388.   363.   360.   359.   358.   358.   346.   292.   276.]
median: 120.0
```

Looks like for the most part we're sampling every 2 minutes, with a few gaps in the data.

1.5 Faithfulness: are the data trustworthy?

This one's much harder to assess. Let's have a look at some basic things we might care about

[78]:
```
sum(Bower['PM2.5_ATM_ug/m3'].isna())
```

[78]:
```
0
```

That tells us there are no NaN values in the PM2.5 data. Impressive!

[79]:
```
Bower.describe()
```

[79]:

```
            entry_id  PM1.0_CF1_ug/m3  PM2.5_CF1_ug/m3  PM10.0_CF1_ug/m3  \
count  351942.000000    351942.000000    351942.000000     351942.000000
mean   390592.636500         7.703851        15.989965         24.196788
std     68985.096555         9.894243        19.466634         24.491579
min    271065.000000         0.000000         0.000000          0.000000
25%    336116.000000         1.210000         3.590000          6.750000
50%    380108.500000         3.760000         8.750000         16.410000
75%    444398.750000        10.220000        20.280000         33.270000
max    532384.000000       336.360000      1297.910000       1582.450000

       UptimeMinutes       RSSI_dbm  Temperature_F     Humidity_%  \
count  351942.000000  351942.000000  351942.000000  351942.000000
mean    26747.629834     -70.180916      65.679018      50.249541
std     18879.817383       5.026611      11.876395      14.772064
min         1.000000     -93.000000      36.000000       5.000000
25%     10005.000000     -71.000000      58.000000      41.000000
50%     23995.500000     -69.000000      63.000000      55.000000
75%     41299.000000     -67.000000      71.000000      62.000000
max     71582.000000     -53.000000     126.000000      77.000000

       PM2.5_ATM_ug/m3  Unnamed: 10
count    351942.000000          0.0
mean         14.203361          NaN
std          14.610303          NaN
min           0.000000          NaN
25%           3.590000          NaN
50%           8.750000          NaN
75%          20.270000          NaN
max         865.120000          NaN
```

That's a pretty high PM2.5 average. And the max is very suspiciously high. What's going on?

Options: 1. Wildfire smoke really pumped up the 2.5 values 2. We have a lot of missing data and only values during the wild fires 3. There are some erroneously high values.

Let's start by looking at how many values are big.

[81]:

```
log_ind = Bower.loc[:,'PM2.5_CF1_ug/m3'] > 500 # gives a list for logical␣
 ↪indexing
Bower.loc[log_ind,'PM2.5_CF1_ug/m3']
```

[81]:

```
150102     608.21
150822     608.21
305431    1297.91
305432     878.52
Name: PM2.5_CF1_ug/m3, dtype: float64
```

Let's look in the vicinity of the high values to see if we believe the trend:

[82]:
```
Bower.loc[305420:305435,:]
```

[82]:
```
                        created_at  entry_id  PM1.0_CF1_ug/m3  PM2.5_CF1_ug/m3  \
305420 2022-07-05 05:46:09+00:00    485863             5.17            10.33
305421 2022-07-05 05:48:09+00:00    485864             6.66            10.88
305422 2022-07-05 05:50:09+00:00    485865             6.81            12.11
305423 2022-07-05 05:52:09+00:00    485866             6.98            11.89
305424 2022-07-05 05:54:09+00:00    485867             5.82            11.07
305425 2022-07-05 05:56:09+00:00    485868             5.65            10.33
305426 2022-07-05 05:58:09+00:00    485869             7.17            13.38
305427 2022-07-05 06:00:09+00:00    485870             5.09             8.67
305428 2022-07-05 06:02:09+00:00    485871             4.52             8.33
305429 2022-07-05 06:04:09+00:00    485872             5.62            10.02
305430 2022-07-05 06:06:09+00:00    485873            11.17            17.88
305431 2022-07-05 06:08:09+00:00    485874           336.36          1297.91
305432 2022-07-05 06:10:09+00:00    485875           332.55           878.52
305433 2022-07-05 06:12:09+00:00    485876           173.87           342.52
305434 2022-07-05 06:14:09+00:00    485877            68.84           129.05
305435 2022-07-05 06:16:09+00:00    485878            26.69            48.02

        PM10.0_CF1_ug/m3  UptimeMinutes  RSSI_dbm  Temperature_F  Humidity_%  \
305420             12.72        70909.0     -67.0           73.0        59.0
305421             12.31        70911.0     -67.0           72.0        59.0
305422             14.12        70913.0     -72.0           72.0        60.0
305423             14.58        70915.0     -69.0           72.0        60.0
305424             14.45        70917.0     -68.0           73.0        60.0
305425             13.14        70919.0     -67.0           73.0        60.0
305426             16.48        70921.0     -72.0           72.0        60.0
305427             10.78        70923.0     -65.0           72.0        60.0
305428             11.19        70925.0     -69.0           72.0        59.0
305429             11.29        70927.0     -67.0           72.0        60.0
305430             23.90        70929.0     -71.0           72.0        60.0
305431           1582.45        70931.0     -68.0           72.0        60.0
305432           1027.02        70933.0     -71.0           72.0        60.0
305433            392.19        70935.0     -68.0           68.0        59.0
305434            155.05        70937.0     -67.0           73.0        60.0
305435             60.86        70939.0     -69.0           72.0        60.0

        PM2.5_ATM_ug/m3  Unnamed: 10
305420            10.33          NaN
305421            10.88          NaN
305422            12.11          NaN
305423            11.89          NaN
305424            11.07          NaN
305425            10.33          NaN
305426            13.38          NaN
305427             8.67          NaN
```

```
305428              8.33           NaN
305429             10.02           NaN
305430             17.88           NaN
305431            865.12           NaN
305432            585.00           NaN
305433            227.58           NaN
305434             85.29           NaN
305435             39.97           NaN
```

Looks like there was a stretch of time with really high values, somewhat suspciously clustered around 5000. If I were doing more work here I would look into the sensor more carefully to see if there is any significance to that number.

But for now – let's just go ahead and drop them and see what happens:

[83]:
```python
Bower.loc[log_ind,'PM2.5_CF1_ug/m3'] = np.nan
Bower.describe()
```

[83]:
```
            entry_id  PM1.0_CF1_ug/m3  PM2.5_CF1_ug/m3  PM10.0_CF1_ug/m3  \
count  351942.000000    351942.000000    351938.000000     351942.000000
mean   390592.636500         7.703851        15.980506         24.196788
std     68985.096555         9.894243        19.240002         24.491579
min    271065.000000         0.000000         0.000000          0.000000
25%    336116.000000         1.210000         3.590000          6.750000
50%    380108.500000         3.760000         8.750000         16.410000
75%    444398.750000        10.220000        20.280000         33.270000
max    532384.000000       336.360000       346.050000       1582.450000

       UptimeMinutes       RSSI_dbm  Temperature_F     Humidity_%  \
count  351942.000000  351942.000000  351942.000000  351942.000000
mean    26747.629834     -70.180916      65.679018      50.249541
std     18879.817383       5.026611      11.876395      14.772064
min         1.000000     -93.000000      36.000000       5.000000
25%     10005.000000     -71.000000      58.000000      41.000000
50%     23995.500000     -69.000000      63.000000      55.000000
75%     41299.000000     -67.000000      71.000000      62.000000
max     71582.000000     -53.000000     126.000000      77.000000

       PM2.5_ATM_ug/m3  Unnamed: 10
count    351942.000000          0.0
mean         14.203361          NaN
std          14.610303          NaN
min           0.000000          NaN
25%           3.590000          NaN
50%           8.750000          NaN
75%          20.270000          NaN
max         865.120000          NaN
```

You can see the average came down a little, and the standard deviation came *really* far down. And as we'd hope the max is now below 500.

```
[ ]:
```