Lecture 04.1 Pandas loc iloc

September 5, 2023

1 Pandas .loc and .iloc

Duncan Callaway

My objective in this notebook is to teach people how to access the information in a Pandas dataframe.

```
[1]: import pandas as pd
```

Let's load in the California ISO data we used last time

```
[2]: caiso_data = pd.read_csv('CAISO_2017to2018.csv')
```

```
[3]: caiso_data.head()
```

```
[3]:           Unnamed: 0  GEOTHERMAL  BIOMASS  BIOGAS  SMALL HYDRO  WIND TOTAL  \
     0  2017-08-29 00:00:00        1181      340     156          324        1551
     1  2017-08-29 01:00:00        1182      338     156          326        1556
     2  2017-08-29 02:00:00        1183      337     156          337        1325
     3  2017-08-29 03:00:00        1185      339     156          313        1158
     4  2017-08-29 04:00:00        1190      344     156          320        1209

        SOLAR PV  SOLAR THERMAL
     0         0              0
     1         0              0
     2         0              0
     3         0              0
     4         0              0
```

1.0.1 Q: What do you get if you call the dict of lists with a key?

```
[4]: caiso_data['BIOGAS']
```

```
[4]: 0        156
     1        156
     2        156
     3        156
     4        156
            …
```

```
8755    236
8756    234
8757    233
8758    234
8759    235
Name: BIOGAS, Length: 8760, dtype: int64
```

[5]:
```
type(caiso_data['BIOGAS'])
```

[5]:
```
pandas.core.series.Series
```

Ans: the list associated with the key. This is called a pandas 'series'

1.0.2 Q: Figure out how to get solar production at 2pm on August 29 2017

First let's check that we've got the right index for the time we want:

[6]:
```
caiso_data['Unnamed: 0'][14]
```

[6]:
```
'2017-08-29 14:00:00'
```

Now call the `SOLAR PV` column

[7]:
```
caiso_data['SOLAR PV'][14]
```

[7]:
```
6820
```

1.0.3 Anatomy of the data frame.

Let's talk a little about the anatomy of the data frame.

We have the following important attributes: 1. Rows 2. Columns 2. Index 3. Column names

The "index" can be numeric, but as we'll see we can also make the indices strings.

[8]:
```
caiso_data = pd.read_csv('CAISO_2017to2018.csv')
caiso_data.columns
```

[8]:
```
Index(['Unnamed: 0', 'GEOTHERMAL', 'BIOMASS', 'BIOGAS', 'SMALL HYDRO',
       'WIND TOTAL', 'SOLAR PV', 'SOLAR THERMAL'],
      dtype='object')
```

Note that we can't reassign easily because column and index names lists are immutable. Here is the workaround:

[9]:
```
cols = caiso_data.columns.tolist()
cols[0] = 'Date and time'
caiso_data.columns = cols
caiso_data
```

```
[9]:              Date and time  GEOTHERMAL  BIOMASS  BIOGAS  SMALL HYDRO  \
     0     2017-08-29 00:00:00        1181      340     156          324
     1     2017-08-29 01:00:00        1182      338     156          326
     2     2017-08-29 02:00:00        1183      337     156          337
     3     2017-08-29 03:00:00        1185      339     156          313
     4     2017-08-29 04:00:00        1190      344     156          320
     …                       …         …       …       …           …
     8755  2018-08-28 19:00:00         962      332     236          581
     8756  2018-08-28 20:00:00         967      336     234          547
     8757  2018-08-28 21:00:00         972      336     233          502
     8758  2018-08-28 22:00:00         975      333     234          361
     8759  2018-08-28 23:00:00         977      333     235          262

           WIND TOTAL  SOLAR PV  SOLAR THERMAL
     0           1551         0              0
     1           1556         0              0
     2           1325         0              0
     3           1158         0              0
     4           1209         0              0
     …            …        …              …
     8755        3300        70             24
     8756        3468         0             17
     8757        3310         0             17
     8758        3068         0              0
     8759        2921         0              0

      [8760 rows x 8 columns]
```

Ok, that looks a little better for now.

As you can see, all the data are the same type of numeric value – MWh.

In these cases, sometimes it's natural to "stack" the data.

We could do the stacking with a pandas command, `.stack`

1.1 Indexing and slicing in Pandas

First let's figure out how to slice these data frames.

`.iloc` allows us to index and slice on **i**nteger row and column positions, like numpy:

```
[10]: caiso_data.iloc[1,1]
```

```
[10]: 1182
```

But what's nice about `.iloc` is that you can also slice. It works just like numpy.

1.1.1 Q: Take a slice of the caiso_data dataframe that grabs the first four columns of data and the first 10 rows

```
[11]: caiso_data.iloc[0:10, :4]
```

```
[11]:           Date and time  GEOTHERMAL  BIOMASS  BIOGAS
      0  2017-08-29 00:00:00        1181      340     156
      1  2017-08-29 01:00:00        1182      338     156
      2  2017-08-29 02:00:00        1183      337     156
      3  2017-08-29 03:00:00        1185      339     156
      4  2017-08-29 04:00:00        1190      344     156
      5  2017-08-29 05:00:00        1194      351     157
      6  2017-08-29 06:00:00        1196      359     155
      7  2017-08-29 07:00:00        1194      363     153
      8  2017-08-29 08:00:00        1187      364     153
      9  2017-08-29 09:00:00        1189      367     157
```

1.1.2 Q: What would you do if you wanted to get the *last* 10 rows?

```
[12]: caiso_data.iloc[-10:, :4]
```

```
[12]:              Date and time  GEOTHERMAL  BIOMASS  BIOGAS
      8750  2018-08-28 14:00:00         933      338     240
      8751  2018-08-28 15:00:00         933      338     238
      8752  2018-08-28 16:00:00         934      337     239
      8753  2018-08-28 17:00:00         934      336     235
      8754  2018-08-28 18:00:00         955      337     237
      8755  2018-08-28 19:00:00         962      332     236
      8756  2018-08-28 20:00:00         967      336     234
      8757  2018-08-28 21:00:00         972      336     233
      8758  2018-08-28 22:00:00         975      333     234
      8759  2018-08-28 23:00:00         977      333     235
```

1.1.3 Q: Can you print out the last ten rows in reverse order?

```
[13]: caiso_data.iloc[:-10:-1,:4]
```

```
[13]:              Date and time  GEOTHERMAL  BIOMASS  BIOGAS
      8759  2018-08-28 23:00:00         977      333     235
      8758  2018-08-28 22:00:00         975      333     234
      8757  2018-08-28 21:00:00         972      336     233
      8756  2018-08-28 20:00:00         967      336     234
      8755  2018-08-28 19:00:00         962      332     236
      8754  2018-08-28 18:00:00         955      337     237
      8753  2018-08-28 17:00:00         934      336     235
      8752  2018-08-28 16:00:00         934      337     239
      8751  2018-08-28 15:00:00         933      338     238
```

`.loc` is similar to `.iloc`, but it allows you to call the index and column names:

```
[14]: caiso_data.loc[0:5,'Date and time']
```

```
[14]: 0    2017-08-29 00:00:00
      1    2017-08-29 01:00:00
      2    2017-08-29 02:00:00
      3    2017-08-29 03:00:00
      4    2017-08-29 04:00:00
      5    2017-08-29 05:00:00
      Name: Date and time, dtype: object
```

You can even slice with column names:

```
[15]: caiso_data.loc[0:5,'Date and time':'BIOGAS']
```

```
[15]:          Date and time  GEOTHERMAL  BIOMASS  BIOGAS
      0  2017-08-29 00:00:00        1181      340     156
      1  2017-08-29 01:00:00        1182      338     156
      2  2017-08-29 02:00:00        1183      337     156
      3  2017-08-29 03:00:00        1185      339     156
      4  2017-08-29 04:00:00        1190      344     156
      5  2017-08-29 05:00:00        1194      351     157
```

1.1.4 Q: Is .loc end-inclusive or exclusive when you slice?

Ans: *inclusive*. This is because it requires less knowledge about other rows in the DataFrame.

Note that this is true for both the index and the column names.

1.2 Recap

- Pandas dataframes are sophisticated dicts of lists.
  - They have attributes like columns and index that have special meaning in the pandas context.
  - You can store any combination of data types in the dataframe
- You can access information in them as though they are dicts of lists
- But you can also use the .loc and .iloc methods to access information in a way similar to numpy, including clean slicing.