Lecture 04.1 Pandas loc iloc

September 6, 2022

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.

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

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

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

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

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

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

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

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:

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

Now call the `SOLAR PV` column

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

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.

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

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

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

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:

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

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

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

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

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

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

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

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

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

You can even slice with column names:

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

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.