Exemplar_ Dataframes with pandas

December 16, 2023

Exemplar: Dataframes with pandas

0.1 Introduction

Your work as a data professional for the U.S. Environmental Protection Agency (EPA) requires you to analyze air quality index data collected from the United States and Mexico.

The air quality index (AQI) is a number that runs from 0 to 500. The higher the AQI value, the greater the level of air pollution and the greater the health concern. For example, an AQI value of 50 or below represents good air quality, while an AQI value over 300 represents hazardous air quality. Refer to this guide from AirNow.gov for more information.

In this lab, you will practice working in pandas. You will load a dataframe, examine its metadata and summary statistics, and explore it using iloc indexing and sorting. You will also practice Boolean masking, grouping, and concatenating data.

0.2 Tips for completing this lab

As you navigate this lab, keep the following tips in mind:

- `### YOUR CODE HERE ###` indicates where you should write code. Be sure to replace this with your own code before running the code cell.
- Feel free to open the hints for additional guidance as you work on each task.
- To enter your answer to a question, double-click the markdown cell to edit. Be sure to replace the "[Double-click to enter your responses here.]" with your own answer.
- You can save your work manually by clicking File and then Save in the menu bar at the top of the notebook.
- You can download your work locally by clicking File and then Download and then specifying your preferred file format in the menu bar at the top of the notebook.

0.3 Task 1: Read data from csv file into a pandas dataframe

You are given two files of data. Begin with the first file, which contains the three states with the most observations (rows): California, Texas, and Pennsylvania.

0.3.1 1a: Import statements

Import numpy and pandas. Use their standard aliases.

```
[1]: ### YOUR CODE HERE ###
     import numpy as np
     import pandas as pd
```

Hint 1

Begin with the `import` keyword for each statement.

Hint 2

Use the `as` keyword to assign an alias.

Hint 3

The conventional aliases are `np` for numpy and `pd` for pandas.

0.3.2 1b: Read in the first file

1. Use the `pd.read_csv()` function to read in the data from the three states with the most observations. The file is called `'epa_ca_tx_pa.csv'` and is already in your working directory. Assign the resulting dataframe to a variable named `top3`.
2. Use the `head()` method on the `top3` dataframe to inspect the first five rows.

```
[2]: # 1. ### YOUR CODE HERE ###
     top3 = pd.read_csv('epa_ca_tx_pa.csv')

     # 2. ### YOUR CODE HERE ###
     top3.head()
```

```
[2]:    state_code  state_name  county_code county_name   aqi
     0           6  California            1     Alameda  11.0
     1           6  California            7       Butte   6.0
     2           6  California           19      Fresno  11.0
     3           6  California           29        Kern   7.0
     4           6  California           29        Kern   3.0
```

Hint

Because the file is already in your working directory, you can simply pass the file name to the `pd.read_csv()` function as a string.

0.4 Task 2: Summary information

Now that you have a dataframe with the AQI data for California, Texas, and Pennsylvania, get some high-level summary information about it.

0.4.1 2a: Metadata

Use a DataFrame method to examine the number of rows and columns, the column names, the data type contained in each column, the number of non-null values in each column, and the amount of memory the dataframe uses.

[3]:
```
### YOUR CODE HERE ###
top3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 546 entries, 0 to 545
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   state_code   546 non-null    int64
 1   state_name   546 non-null    object
 2   county_code  546 non-null    int64
 3   county_name  546 non-null    object
 4   aqi          546 non-null    float64
dtypes: float64(1), int64(2), object(2)
memory usage: 21.5+ KB
```

Hint

The `info()` method returns a dataframe's metadata.

0.4.2 2b: Summary statistics

Examine the summary statistics of the dataframe's numeric columns. The output should be a table that includes row count, mean, standard deviation, min, max, and quartile values.

[4]:
```
### YOUR CODE HERE ###
top3.describe()
```

[4]:

	state_code	county_code	aqi
count	546.000000	546.000000	546.000000
mean	20.593407	83.179487	8.906593
std	19.001484	92.240873	9.078479
min	6.000000	1.000000	0.000000
25%	6.000000	29.000000	3.000000
50%	6.000000	66.000000	6.000000
75%	42.000000	98.500000	11.000000
max	48.000000	479.000000	93.000000

Hint

The `describe()` method returns a table of summary statistics for a dataframe's numeric columns.

0.5 Task 3: Explore your data

Practice exploring your data by completing the following exercises.

0.5.1 3a: Rows per state

Select the `state_name` column and use the `value_counts()` method on it to check how many rows there are for each state in the dataframe.

[5]:
```
### YOUR CODE HERE ###
top3['state_name'].value_counts()
```

[5]:
```
California      342
Texas           104
Pennsylvania    100
Name: state_name, dtype: int64
```

Hint 1

This code should all be on a single line. Begin by selecting the `top3` dataframe at the `state_name` column. You can use either selector brackets or dot notation.

Hint 2

When using brackets to select a column, the column's name should be entered as a string.

Hint 3

The `value_counts()` method is added to the end of the selection statement using dot notation. Its parentheses are empty.

0.5.2 3b: Sort by AQI

1. Create a new dataframe called `top3_sorted` by using the `sort_values()` method on the `top3` dataframe. Refer to the sort_values pandas documentation for more information about how to use this method.
   - The new dataframe should contain the data sorted by AQI, beginning with the rows with the highest AQI values.
2. Print the top 10 rows of `top3_sorted`.

[6]:
```
# 1. ### YOUR CODE HERE ###
top3_sorted = top3.sort_values(by='aqi', ascending=False)

# 2. ### YOUR CODE HERE ###
top3_sorted.head(10)
```

[6]:
```
     state_code  state_name  county_code  county_name   aqi
76            6  California           37  Los Angeles  93.0
146           6  California           37  Los Angeles  59.0
```

```
41             6  California           83  Santa Barbara  47.0
122            6  California           59         Orange  47.0
184            6  California           59         Orange  47.0
51            48       Texas          141        El Paso  47.0
80             6  California           65      Riverside  43.0
136           48       Texas          141        El Paso  40.0
58             6  California           65      Riverside  40.0
91            48       Texas          141        El Paso  40.0
```

Hint 1

Attach the `sort_values()` method to the `top3` dataframe using dot notation.

Hint 2

- The `by` argument of the `sort_values()` method should be a string of the column you want to sort by.
- The default behavior of the `sort_values()` method is to sort in ascending order. You want to sort in *descending* order. Which keyword argument modifies this behavior? Refer to the sort_values() pandas documentation.

Hint 3

- Use the `head()` method on the `top3_sorted` dataframe.
- The default behavior of the `head()` method is to print the first five rows of a dataframe. You want to print the first 10 rows. Refer to the head() pandas documentation for more information on how to modify this behavior.

0.5.3 3c: Use `iloc` to select rows

Use `iloc` to select the two rows at indices 10 and 11 of the `top3_sorted` dataframe.

[7]:
```
### YOUR CODE HERE ###
top3_sorted.iloc[10:12]
```

[7]:
```
     state_code  state_name  county_code  county_name   aqi
186           6  California           73    San Diego  39.0
74            6  California           37  Los Angeles  38.0
```

Hint 1

To use `iloc` on the `top3_sorted` dataframe, use dot notation.

Hint 2

`iloc` uses brackets to index data.

Hint 3

`iloc` index ranges are separated by a colon. Remember, the end index is not included in the range of returned indices.

0.6 Task 4: Examine California data

You notice that the rows with the highest AQI represent data from California, so you want to examine the data for just the state of California.

0.6.1 4a: Basic Boolean masking

1. Create a Boolean mask that selects only the observations of the `top3_sorted` dataframe that are from California.
2. Apply the Boolean mask to the `top3_sorted` dataframe and assign the result to a variable called `ca_df`.
3. Print the first five rows of `ca_df`.

[8]:
```
# 1. ### YOUR CODE HERE ###
mask = top3_sorted['state_name'] == 'California'

# 2. ### YOUR CODE HERE ###
ca_df = top3_sorted[mask]

# 3. ### YOUR CODE HERE ###
ca_df.head()
```

[8]:
```
     state_code  state_name  county_code    county_name   aqi
76            6  California           37    Los Angeles  93.0
146           6  California           37    Los Angeles  59.0
41            6  California           83  Santa Barbara  47.0
122           6  California           59         Orange  47.0
184           6  California           59         Orange  47.0
```

Hint 1

Refer to what you've learned about Boolean masking.

Hint 2

Define a `mask` variable that is the `top3_sorted` dataframe selected where the `state_name` column is `California`.

Hint 3

- Apply the `mask` variable to the `top3_sorted` dataframe and assign the result to a new dataframe called `ca_df`.
- Use the `head()` method on the new `ca_df` dataframe.

0.6.2 4b: Validate CA data

Inspect the shape of your new `ca_df` dataframe. Does its row count match the number of California rows determined in Task 3a?

```
[9]: ### YOUR CODE HERE ###
     ca_df.shape
```

```
[9]: (342, 5)
```

Hint

- Use the `shape` attribute on the `ca_df` dataframe.
- Attributes don't use parentheses.

0.6.3 4c: Rows per CA county

Examine a list of the number of times each county is represented in the California data.

```
[10]: ### YOUR CODE HERE ###
      ca_df['county_name'].value_counts()
```

```
[10]: Los Angeles        55
      Santa Barbara      26
      San Bernardino     21
      Orange             19
      San Diego          19
      Sacramento         17
      Alameda            17
      Fresno             16
      Riverside          14
      Contra Costa       13
      Imperial           13
      San Francisco       8
      Monterey            8
      Humboldt            8
      Santa Clara         7
      El Dorado           7
      Placer              6
      Butte               6
      Kern                6
      Mendocino           6
      Solano              5
      San Joaquin         5
      Tulare              5
      Ventura             5
      Sutter              4
      San Mateo           4
      Marin               3
      Sonoma              3
      Stanislaus          3
      San Luis Obispo     2
```

```
Napa                 2
Santa Cruz           2
Calaveras            2
Shasta               1
Tuolumne             1
Inyo                 1
Yolo                 1
Mono                 1
Name: county_name, dtype: int64
```

Hint

Select the `county_name` column of `ca_df` and apply the `value_counts()` method to it using dot notation.

0.6.4 4d: Calculate mean AQI for Los Angeles county

You notice that Los Angeles county has more than twice the number of rows of the next-most-represented county in California, and you want to learn more about it.

- Calculate the mean AQI for LA county.

[11]:
```
### YOUR CODE HERE ###
mask = ca_df['county_name'] == 'Los Angeles'
ca_df[mask]['aqi'].mean()
```

[11]:
```
13.4
```

Hint 1

Use Boolean masking to create a mask. Then apply the mask to the `ca_df` dataframe, select the `aqi` column of the result, and calculate its mean.

Hint 2

The Boolean mask is `ca_df` selected where `county_name` is `Los Angeles`.

Hint 3

- Apply the Boolean mask to `ca_df`.
- Then use selector brackets to select the `aqi` column.
- Then use dot notation to attach the `mean()` method to the end of the expression.

0.7 Task 5: Groupby

Group the original dataframe (`top3`) by state and calculate the mean AQI for each state.

[12]:
```
### YOUR CODE HERE ###
top3.groupby('state_name').mean()[['aqi']]
```

[12]:
```
                    aqi
state_name
California     9.412281
Pennsylvania   6.690000
Texas          9.375000
```

Hint 1

Use the `groupby()` method on the `top3` dataframe.

Hint 2

Group by `state_name` and use dot notation to chain the `mean()` method to the expression.

Hint 3

`top3.groupby('state_name').mean()` will produce a table of the mean values of every numeric column for each state. To filter the table on just the `aqi` column, add `['aqi']` to the end of the expression.

0.8 Task 6: Add more data

Now that you have performed a short examination of the file with AQI data for California, Texas, and Pennsylvania, you want to add more data from your second file.

0.8.1 6a: Read in the second file

1. Read in the data for the remaining territories. The file is called `'epa_others.csv'` and is already in your working directory. Assign the resulting dataframe to a variable named `other_states`.
2. Use the `head()` method on the `other_states` dataframe to inspect the first five rows.

[13]:
```python
# 1. ### YOUR CODE HERE ###
other_states = pd.read_csv('epa_others.csv')

# 2. ### YOUR CODE HERE ###
other_states.head()
```

[13]:
```
   state_code state_name  county_code county_name   aqi
0           4    Arizona           13    Maricopa  18.0
1           4    Arizona           13    Maricopa   9.0
2           4    Arizona           19        Pima  20.0
3           8   Colorado           41     El Paso   9.0
4          12    Florida           31       Duval  15.0
```

0.8.2 6b: Concatenate the data

The data from `other_states` is in the same format as the data from `top3`. It has the same columns in the same order.

1. Add the data from `other_states` as new rows beneath the data from `top3`. Assign the result to a new dataframe called `combined_df`.
2. Verify that the length of `combined_df` is equal to the sum of the lengths of `top3` and `other_states`.

```
[14]: # 1. ### YOUR CODE HERE ###
      combined_df = pd.concat([top3, other_states], axis=0)

      # 2. ### YOUR CODE HERE ###
      len(combined_df) == len(top3) + len(other_states)
```

```
[14]: True
```

Hint 1

Use the `concat()` function. For more information on this function, refer to the concat() pandas documentation.

Hint 2

- Enter the two dataframes being joined as a list in the argument field of the `concat()` function.
- To add rows, concatenate along axis 0. To add columns, concatenate along axis 1.

Hint 3

Use the `len()` function in a comparison statement to determine if the length of `combined_df` equals the length of `top3` plus the length of `other_states`.

0.9 Task 7: Complex Boolean masking

According to the EPA, AQI values of 51-100 are considered of "Moderate" concern. You've been tasked with examining some data for the state of Washington.

- Use Boolean masking to return the rows that represent data from the state of Washington with AQI values of 51+.

```
[15]: ### YOUR CODE HERE ###
      mask = (combined_df['state_name'] == 'Washington') & (combined_df['aqi'] >= 51)
      combined_df[mask]
```

```
[15]:      state_code  state_name  county_code county_name   aqi
      40           53  Washington           33        King  55.0
      82           53  Washington           61   Snohomish  76.0
      121          53  Washington           77      Yakima  58.0
      122          53  Washington           77      Yakima  57.0
```

Hint 1

Create a Boolean mask for `combined_df` with two conditions:

- The state is Washington
- The AQI is greater than or equal to 51

Hint 2

Remember to enclose each condition in its own set of parentheses.

Hint 3

Separate the two conditions with the `&` operator, because both conditions need to evaluate as true for the row to be included in the filtered dataframe.

0.10 Conclusion

What are your key takeaways from this lab?

pandas is a powerful tool for working with data in Python because: * It it comes with many built-in functions and tools specifically designed for use with tabular data to simplify common tasks such as: * Reading and writing data to/from files * Quickly computing summary statistics about your data * Manipulating, selecting, and filtering data * Grouping and aggregating data * Adding new data to existing data

- It's powered by NumPy, which uses the power of array operations to enhance performance.
- Its interface makes working with tabular data easy because it allows you to visualize your data in rows and columns.

Congratulations! You've completed this lab. However, you may not notice a green check mark next to this item on Coursera's platform. Please continue your progress regardless of the check mark. Just click on the "save" icon at the top of this notebook to ensure your work has been logged.