lab04

March 11, 2020

0.1 Lab 4: Pandas Continued

This assignment should be completed by Tuesday February 25, 2020 at 11:59 AM.

Pandas is one of the most widely used Python libraries in data science. In this lab, you will learn commonly used data wrangling operations/tools in Pandas. We plan to continue the discussion from Lab 3 learning about

- Grouping dataframes
- Merging dataframes

We will use the baby names dataset from the Social Security Administration stratified by state.

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

import os
```

0.2 Grouping Tables

Load the data from `NY_babynames.csv`

[2]:
```
ny = pd.read_csv(os.environ["HOME"] + "/shared/NY_babynames.csv")
```

0.2.1 Question 1a: value_counts

To count the number of instances of each unique value in a `Series`, we can use the `value_counts()` method as `df["col_name"].value_counts()`.

Count the number of different names for each Year in `NY` (New York). (You may use the `ny` DataFrame created above.)

Note: *We are not computing the number of babies but instead the number of names (rows in the table) for each year.*

[3]:
```
num_of_names_per_year = ny["Year"].value_counts()
# YOUR CODE HERE
```

```
#raise NotImplementedError()
```

[4]:
```
# TEST
assert num_of_names_per_year[2007] == 4680
```

[5]:
```
# TEST
assert num_of_names_per_year[:5].sum() == 23521
```

0.2.2 Question 1b

Count the number of different names for each gender in `NY`.

[6]:
```
num_of_names_per_gender = ny["Sex"].value_counts()
# YOUR CODE HERE
#raise NotImplementedError()
```

[7]:
```
# TEST
assert num_of_names_per_gender["F"] == 169145
```

0.2.3 Question 2: groupby

Before we jump into using the groupby function in Pandas, let's recap how grouping works in general for tabular data through a guided set of questions based on a small dataset of movies and genres. Please see Week 4 lecture slides for a review of grouping.

Problem Setting: Below is a dataframe with 5 columns: name of the movie as a `string`, the genre of the movie as a `string`, the first name of the director of the movie as a `string`, the average rating out of 10 on Rotten Tomatoes as an `integer`, and the total gross revenue made by the movie as an `integer`. The point of these guided questions (parts a and b) below is to understand how grouping of data works in general, **not** how grouping works in code. We will worry about how grouping works in Pandas in 7c, which will follow.

Below is the `movies` dataframe we are using, imported from the `movies.csv` file.

[8]:
```
movies = pd.read_csv(os.environ["HOME"] + "/shared/movies.csv")
```

0.2.4 Question 2a

If we grouped the `movies` dataframe above by `genre`, how many groups would be in the output and what would be the groups? Assign `num_groups` to the number of groups created (hard-code) and fill in `genre_list` as a list containing the names of genres as strings that represent the groups.

[14]:
```
#print(movies)
num_groups = movies.groupby(["genre"]).size().count()
genre_list =["Action &␣
 ↪Adventure","Animation","Comedy","Drama","Horror","Science Fiction & Fantasy"]
```

```
# YOUR CODE HERE
#raise NotImplementedError()
```

[15]:
```
# TEST
assert num_groups == 6
```

0.2.5 Question 2b

Whenever we group tabular data, it is usually the case that we need to aggregate values from the ungrouped column(s). If we were to group the `movies` dataframe above by `genre`, which column(s) in the `movies` dataframe would it make sense to aggregate if we were interested in finding how well each genre did in the eyes of people? Fill in `agg_cols` with the column name(s).

[18]:
```
agg_cols = ['rating', 'revenue']
# YOUR CODE HERE
#raise NotImplementedError()
```

[19]:
```
# TEST
assert sorted(agg_cols) == ['rating', 'revenue']
```

Now, let's see `groupby` in action, instead of keeping everything abstract. To aggregate data in Pandas, we use the `.groupby()` function. The code below will group the `movies` dataframe by `genre` and find the average revenue and rating for each genre. You can verify you had the same number of groups as what you answered in 2a.

[20]:
```
movies.loc[:, ['genre', 'rating', 'revenue']].groupby('genre').mean()
```

[20]:

genre	rating	revenue
Action & Adventure	6.333333	153569934.5
Animation	5.000000	374408165.0
Comedy	6.000000	56719237.4
Drama	6.000000	17146165.5
Horror	7.000000	68765655.0
Science Fiction & Fantasy	6.000000	312674899.0

0.2.6 Question 2c

Let's move back to baby names and specifically, the `ny` dataframe. Find the sum of `Count` for each `Name` in the `ny` table. You should use `df.groupby("col_name").sum()`. Your result should be a Pandas Series.

Note: *In this question we are now computing the number of registered babies with a given name.*

[188]:
```
#print(movies)
s = ny.groupby("Name")
count_for_names = s["Count"].sum()
# YOUR CODE HERE
#raise NotImplementedError()


count_for_names.sort_values(ascending=False)[:5]
```

```
Name
Aaban      12
Aaden     253
Aadhya     28
Aadi       31
Aadil       5
          …
Zyire       5
Zyla       16
Zylah       6
Zymir      18
Zyon      105
Name: Count, Length: 14592, dtype: int64
```

[123]:
```
# TEST
assert count_for_names["Michael"] == 438074
```

[124]:
```
# TEST
assert count_for_names[:100].sum() == 72157
```

0.2.7 Question 2d

Find the sum of `Count` for each female name after year 1999 (`>1999`) in New York.

[186]:
```
s = ny[ny["Sex"]=="F"]
#print(s)
t = s[s["Year"]>1999]
#print(t)
r = t.groupby("Name").sum()
#print(r)
female_name_count= r["Count"]

#female_name_count = t["Count"].sum()
#print(female_name_count)
# YOUR CODE HERE
#raise NotImplementedErcror()
```

```
female_name_count.sort_values(ascending=False)[:5]
```

[186]:
```
Name
Emily       21485
Isabella    21446
Olivia      21420
Emma        20587
Sophia      19883
Name: Count, dtype: int64
```

[137]:
```
# TEST
assert female_name_count["Emily"] == 21485
```

[138]:
```
# TEST
assert female_name_count["Isabella"] == 21446
```

0.2.8 Question 3: Grouping Multiple Columns

Let's move back to the `movies` dataframe. Which of the following lines of code will output the following dataframe? Write your answer (hard-coded) as either 1, 2, 3, or 4. Recall that the arguments to `pd.pivot_table` are as follows: `data` is the input dataframe, `index` includes the values we use as rows, `columns` are the columns of the pivot table, `values` are the values in the pivot table, and `aggfunc` is the aggregation function that we use to aggregate `values`.

rating

5

6

7

8

genre

Action & Adventure

208681866.0

129228350.0

318344544.0

6708147.0

Animation

374408165.0

NaN

NaN

NaN

Comedy

55383976.0

30561590.0

NaN

111705055.0

Drama

NaN

17146165.5

NaN

NaN

Horror

NaN

NaN

68765655.0

NaN

Science Fiction & Fantasy

NaN

312674899.0

NaN

NaN

1) `pd.pivot_table(data=movies, index='genre', columns='rating', values='revenue', aggfunc=np.mean)`
2) `movies.groupby(['genre', 'rating'])['revenue'].mean()`
3) `pd.pivot_table(data=movies, index='rating', columns='genre', values='revenue', aggfunc=np.mean)`
4) `movies.groupby('revenue')[['genre', 'rating']].mean()`

[143]:
```
q3_answer = 1
#raise NotImplementedError()
```

[ ]:
```

```

0.2.9 Question 4: Merging

Question 4a Time to put everything together! Merge `movies` and `count_for_names` to find the number of registered baby names for each director using `pd.merge`. Only include names that appear in both `movies` and `count_for_names`.

Hint: You might need to convert the `count_for_names` series to a dataframe. Take a look at the `to_frame` method of a series to do this.

Your first row should look something like this:

Note: It is ok if you have 2 separate columns with names instead of just one column.

director

genre

movie

rating

revenue

Count

0

David

Action & Adventure

Deadpool 2

7

318344544

371646

[216]:
```
#print(movies)
#print(count_for_names)
s = movies.merge(count_for_names.to_frame(),how = 'left',left_on='director',␣
 ↪right_on='Name')
merged_df = s.dropna()
# YOUR CODE HERE
#raise NotImplementedError()
print(s)


merged_df.head()
```

```
      director                       genre  \
0        David         Action & Adventure
1         Bill                      Comedy
2          Ron  Science Fiction & Fantasy
3      Baltasar                      Drama
```

```
4          Bart                      Drama
5          Gary         Action & Adventure
6          Drew         Action & Adventure
7          Brad                  Animation
8          Jeff                     Comedy
9          J.A.  Science Fiction & Fantasy
10      Charles                     Comedy
11       Gerard                     Horror
12       Peyton         Action & Adventure
13       Genndy                  Animation
14       Rawson         Action & Adventure
15           Ol                     Comedy
16  Christopher         Action & Adventure
17         Marc                     Comedy
```

```
                                  movie  rating    revenue     Count
0                           Deadpool 2       7  318344544  280681.0
1                            Book Club       5   68566296    3949.0
2                Solo: A Star Wars Story       6  213476293    1367.0
3                                Adrift       6   31445012       NaN
4                       American Animals       6    2847319    1145.0
5                               Oceans 8       6  138803463   62840.0
6                           Hotel Artemis       8    6708147    4192.0
7                           Incredibles 2       5  594398019    3407.0
8                                     Tag       6   54336863    4707.0
9          Jurassic World: Fallen Kingdom       6  411873505       NaN
10                           Uncle Drew       5   42201656  161375.0
11                      The First Purge       7   68765655   16924.0
12                  Ant-Man and the Wasp       5  208681866    4036.0
13  Hotel Transylvania 3: Summer Vacation       5  154418311       NaN
14                           Skyscraper       6   66801215       NaN
15           Mamma Mia! Here We Go Again       8  111705055       NaN
16           Mission: Impossible-Fallout       6  182080372  171144.0
17                     Christopher Robbin       6    6786317   21322.0
```

[216]:
```
  director                     genre                  movie  rating  \
0    David         Action & Adventure          Deadpool 2       7
1     Bill                     Comedy           Book Club       5
2      Ron  Science Fiction & Fantasy  Solo: A Star Wars Story       6
4     Bart                     Drama         American Animals       6
5     Gary         Action & Adventure                Oceans 8       6

     revenue     Count
0  318344544  280681.0
1   68566296    3949.0
2  213476293    1367.0
4    2847319    1145.0
```

```
5  138803463   62840.0
```

[213]:
```
# TEST
assert merged_df.loc[0, 'Count'] == 280681
```

[214]:
```
# TEST
assert merged_df.loc[7, 'Count'] == 3407
```

[215]:
```
# TEST
assert len(merged_df) == 13
```

Question 4b How many directors in the original `movies` table did not get included in the `merged_df` dataframe? Please hard-code your answer as a number in `q4b`, then explain your answer in 1-2 sentences as a comment below.

[206]:
```
q_4b = 5

# YOUR CODE HERE
raise NotImplementedError()

# Explain your solution: becase their names are rare and maybe some of the␣
 ↪kids' names do not take into count
```

```
␣
↪---------------------------------------------------------------------------

NotImplementedError                       Traceback (most recent call␣
↪last)

<ipython-input-206-82c578e68c31> in <module>
  2
  3 # YOUR CODE HERE
----> 4 raise NotImplementedError()
  5
  6 # Explain your solution: …


NotImplementedError:
```

[ ]:
```

```