Part 2 — Solutions for Lecture 3

TECH2: Introduction to Programming, Data, and Information Technology

Richard Foltyn
Norwegian School of Economics (NHH)

October 16, 2024

See GitHub repository for notebooks and data:

https://github.com/richardfoltyn/TECH2-H24

Contents

1 Grouping and aggregation with pandas

This notebook contains the solutions for the short exercises from lecture 3.

[1]:
```
# Uncomment this to use files in the local data/ directory
DATA_PATH = '../data'

# Uncomment this to load data directly from GitHub
# DATA_PATH = 'https://raw.githubusercontent.com/richardfoltyn/TECH2-H24/main/data'
```

1.1 Exercise 1

Your turn. Use the Titanic data set to perform the following aggregations:

1. Compute the average survival rate by sex (stored in the Sex column).
2. Count the number of passengers aged 50+. Compute the average survival rate by sex for this group.
3. Count the number of passengers below the age of 20 by class and sex. Compute the average survival rate for this group (by class and sex).

Solution.

Part (1)

[2]:
```
import pandas as pd

# File name of Titanic data set
fn = f'{DATA_PATH}/titanic.csv'

# Load Titanic data set
df = pd.read_csv(fn, index_col='PassengerId')
df.head(5)
```

[2]:
```
             Survived  Pclass  \
PassengerId
1                   0       3
2                   1       1
3                   1       3
4                   1       1
5                   0       3

                                                 Name     Sex   Age  \
PassengerId
1                                   Braund, Mr. Owen Harris    male  22.0
2           Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
3                                    Heikkinen, Miss Laina  female  26.0
4                 Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
5                                  Allen, Mr. William Henry    male  35.0

                 Ticket     Fare Cabin Embarked
PassengerId
1             A/5 21171   7.2500   NaN        S
2              PC 17599  71.2833   C85        C
3             STON/O2. 3101282   7.9250   NaN        S
4                113803  53.1000  C123        S
5                373450   8.0500   NaN        S
```

[3]:
```
# Group by Sex
groups = df.groupby('Sex')

# Select Survived column (containing 0/1 values) and compute mean
groups['Survived'].mean()
```

[3]:
```
Sex
female    0.742038
male      0.188908
Name: Survived, dtype: float64
```

[4]:
```
# You can also chain these operations into a single line
df.groupby('Sex')['Survived'].mean()
```

[4]:
```
Sex
female    0.742038
male      0.188908
Name: Survived, dtype: float64
```

Part (2)

[5]:
```
# Load Titanic data set
df = pd.read_csv(fn, index_col='PassengerId')

# Select sub-sample aged 50+
```

```
df_50 = df.query('Age >= 50')

# Report number of passengers in this age group
N = len(df_50)
print(f'Number of passengers aged 50+: {N}')

# Create group object based on DataFrame you just created
groups = df_50.groupby('Sex')

# Select Survived column (containing 0/1 values) and compute mean
groups['Survived'].mean()
```

```
Number of passengers aged 50+: 74
```

[5]:
```
Sex
female    0.909091
male      0.134615
Name: Survived, dtype: float64
```

[6]:
```
# As before, you can perform all this in a single step
df.query('Age >= 50').groupby('Sex')['Survived'].mean()
```

[6]:
```
Sex
female    0.909091
male      0.134615
Name: Survived, dtype: float64
```

Part (3)

[7]:
```
# Load Titanic data set
df = pd.read_csv(fn, index_col='PassengerId')

# Select sub-sample aged below 20
df_20 = df.query('Age < 20')

# Report number of passengers in this age group
N = len(df_20)
print(f'Number of passengers aged below 20: {N}')

# Create group object based on DataFrame you just created
groups = df_20.groupby(['Pclass', 'Sex'])

# Select Survived column (containing 0/1 values) and compute mean
groups['Survived'].mean()
```

```
Number of passengers aged below 20: 164
```

[7]:
```
Pclass  Sex
1       female    0.928571
        male      0.571429
2       female    1.000000
        male      0.526316
3       female    0.533333
        male      0.190476
Name: Survived, dtype: float64
```

[8]:
```
# You can perform all this in a single step
df.query('Age < 20').groupby(['Pclass', 'Sex'])['Survived'].mean()
```

[8]:
```
Pclass  Sex
1       female    0.928571
        male      0.571429
2       female    1.000000
        male      0.526316
3       female    0.533333
        male      0.190476
Name: Survived, dtype: float64
```

1.2 Exercise 2

Your turn. Use the Titanic data set to perform the following aggregations:

1. Compute the minimum, maximum and average age by embarkation port (stored in the column Embarked) in a single `agg()` operation. Note that there are several ways to solve this problem.
2. Compute the number of passengers, the average age and the fraction of women by embarkation port in a single `agg()` operation. This one is more challenging and probably requires use of lambda expressions.

Solution.

Part (1)

[9]:
```
import pandas as pd

# File name of Titanic data set
fn = f'{DATA_PATH}/titanic.csv'

# Load Titanic data set
df = pd.read_csv(fn, index_col='PassengerId')

# Create groups by port
groups = df.groupby('Embarked')

# Compute min, max and mean age by port
groups['Age'].agg(['min', 'max', 'mean'])
```

[9]:

Embarked	min	max	mean
C	0.42	71.0	30.814769
Q	2.00	70.5	28.089286
S	0.67	80.0	29.445397

[10]:
```
# Perform task in a single line
df.groupby('Embarked')['Age'].agg(['min', 'max', 'mean'])
```

[10]:

Embarked	min	max	mean
C	0.42	71.0	30.814769
Q	2.00	70.5	28.089286
S	0.67	80.0	29.445397

Part (2)

There are various ways to compute the share of women. The first approach is the create a `Female` indicator variable and compute its mean.

[11]:
```
# Load Titanic data set
df = pd.read_csv(fn, index_col='PassengerId')

# Create Female column which is 1 when female, 2 if male
df['Female'] = df['Sex'] == "female"

# Tabulate number of men & women
df['Female'].value_counts()
```

[11]:
```
Female
False    577
True     314
Name: count, dtype: int64
```

[12]:
```
# Create groups by port
groups = df.groupby('Embarked')

# Compute desired statistics, assign them to new columns
groups.agg(
    num_passengers=('Age', 'size'),
    avg_age=('Age', 'mean'),
    frac_women=('Female', 'mean')
)
```

[12]:

Embarked	num_passengers	avg_age	frac_women
C	168	30.814769	0.434524
Q	77	28.089286	0.467532
S	644	29.445397	0.315217

Note that for the number of passengers we could have used an arbitrary column since the function `size` returns the number of observations in each group which is the same for each column.

Alternatively, we need not create the `Female` indicator but can use a lambda expression to compute the fraction of women. The lambda expression defines a function in-place which creates the female indicator on the spot and computes its average within each group.

[13]:
```
import numpy as np

# Use lambda function for computing share of women
groups.agg(
    num_passengers=('Age', 'size'),
    avg_age=('Age', 'mean'),
    frac_women=('Sex', lambda x: np.mean(x == 'female'))
)
```

[13]:

Embarked	num_passengers	avg_age	frac_women
C	168	30.814769	0.434524
Q	77	28.089286	0.467532
S	644	29.445397	0.315217

1.3 Exercise 3

Your turn. Use the Titanic data set to perform the following aggregations:

1. Compute the excess fare paid by each passenger relative to the minimum fare by embarkation port and class, i.e., compute $Fare - min(Fare)$ by port and class.

Solution.

[14]:
```
import numpy as np

# File name of Titanic data set
fn = f'{DATA_PATH}/titanic.csv'

# Load Titanic data set
df = pd.read_csv(fn, index_col='PassengerId')

# Define a function to compute excess fare
def excess_fare(x):
    # Compute difference between each observation and the min. value
    # within each group
    return x - np.min(x)

# Group by port and class
groups = df.groupby(['Embarked', 'Pclass'])

# Compute excess fare for each observation
result = groups['Fare'].transform(excess_fare)

# Store result as new column in DataFrame
df['Excess_Fare'] = result

# Print first 5 observations
df[['Embarked', 'Pclass', 'Fare', 'Excess_Fare']].head(5)
```

[14]:

PassengerId	Embarked	Pclass	Fare	Excess_Fare
1	S	3	7.2500	7.2500
2	C	1	71.2833	44.7333
3	S	3	7.9250	7.9250
4	S	1	53.1000	53.1000
5	S	3	8.0500	8.0500

Alternatively, we could combine many of these operations into a single line using a lambda expression:

[15]:
```
# Compute excess fare in single line using lambda expression
df['Excess_Fare'] = df.groupby(['Embarked', 'Pclass'])['Fare'].transform(lambda x: x - np.
 ↪min(x))

# Print first 5 observations
df[['Embarked', 'Pclass', 'Fare', 'Excess_Fare']].head(5)
```

[15]:

PassengerId	Embarked	Pclass	Fare	Excess_Fare
1	S	3	7.2500	7.2500
2	C	1	71.2833	44.7333
3	S	3	7.9250	7.9250
4	S	1	53.1000	53.1000
5	S	3	8.0500	8.0500