Department of Data Science - Data and Visual Analytics Lab

Lab4. Pandas Grouping and Aggregation

Objectives

In this lab, you will learn how to

- apply functions to Series and Dataframe
- group data in Pandas
- aggregate values in groups
- plot the results of aggregation
- aggregate multiple columns and multiple functions

You will explore what Americans typically eat for Thanksgiving dinner. The dataset contains 1058 online survey responses collected by FiveThirtyEight.

Each survey respondent was asked questions about what they typically eat for Thanksgiving, along with some demographic questions, like their gender, income, and location.

This dataset will allow us to discover regional and income-based patterns in what Americans eat for Thanksgiving dinner.
Now, we will compute group summary statistics, discover patterns, and slice up the data in various ways.

Import necessary modules

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

In [ ]:
```
data = pd.read_csv("thanksgiving-2015-poll-data.csv", encoding="Latin-1")
```

In [2]: # Print top 5 rows from data `data.head()`

Out[2]:

	RespondentID	Do you celebrate Thanksgiving?	What is typically the main dish at your Thanksgiving dinner?	What is typically the main dish at your Thanksgiving dinner? - Other (please specify)	How is the main dish typically cooked?	How is the main dish typically cooked? - Other (please specify)	What kind stuffing/dressi do you typica hav
0	4337954960	Yes	Turkey	NaN	Baked	NaN	Bread-bas
1	4337951949	Yes	Turkey	NaN	Baked	NaN	Bread-bas
2	4337935621	Yes	Turkey	NaN	Roasted	NaN	Rice-bas
3	4337933040	Yes	Turkey	NaN	Baked	NaN	Bread-bas
4	4337931983	Yes	Tofurkey	NaN	Baked	NaN	Bread-bas

5 rows × 65 columns

In [3]: # what is the size? `data.shape`

Out[3]: (1058, 65)

As you can see above, the data has 65 columns of mostly categorical data. For example, the first column appears to allow for Yes and No responses only. Let's verify by using the pandas.Series.unique method to see what unique values are in the Do you celebrate Thanksgiving? column of data.

What are unique values of "Do you celebrate Thanksgiving?" column?

In [4]: `data["Do you Celebrate Thanksgiving?"].unique()`

Out[4]: array(['Yes', 'No'], dtype=object)

View all column names (top 5)

```
In [5]: data.columns[:5]

Out[5]: Index(['RespondentID', 'Do you celebrate Thanksgiving?',
               'What is typically the main dish at your Thanksgiving dinner?',
               'What is typically the main dish at your Thanksgiving dinner? - Other
        (please specify)',
               'How is the main dish typically cooked?'],
              dtype='object')
```

Apply function to Series

DATA CLEANING - Now, let us transform gender to numeric value.

We'll assign 0 to Male, and 1 to Female. Before we dive into transforming the values, let's confirm that the values in the column are either Male or Female. We can use the pandas.Series.value_counts method to help us with this. We'll pass the dropna=False keyword argument to also count missing values.

How many male, female and NaN in "What is your gender?" column

```
In [6]: data["What is your gender?"].unique()

Out[6]: Female    544
        Male      481
        NaN        33
        Name: What is your gender?, dtype: int64
```

data["What is your gender?"].value_counts (dropna=False)

Yes, they are female, male or nan

Let apply a user defined function to each value in the What is your gender? column to transform Male to 0 and female to 1

```
In [7]: import math

        def gender_code(gender_string):
            if isinstance(gender_string, float) and math.isnan(gender_string):
                return gender_string
            return int(gender_string == "Female")
```

Apply gender_code() to What is your gender? column

Let us apply this function to every row of What is your gender? column. It is something like automatic looping. Create a new column 'gender' and put it there

```
In [8]: gender_codes = data["What is your gender?"].apply(gender_code)
        print(type(gender_codes), gender_codes.head())
```

Now, count male and females as 0s and 1s. How many in "gender" column?

```
In [9]: data["gender"] = gender_codes
Out[9]: 1.0    544
        0.0    481
        NaN     33
        Name: gender, dtype: int64
```

data["gender"].value_counts(dropna=False)

Applying functions to DataFrames

The apply method will work across each column in the DataFrame. If we pass the axis=1 keyword argument, it will work across each row.

Check the data type of each column in data using a lambda function. Just visualize data types of first 5 columns

```
In [10]: def get_type(row):
             return row.dtype
Out[10]: RespondentID
         object
         Do you celebrate Thanksgiving?
         object
         What is typically the main dish at your Thanksgiving dinner?
         object
         What is typically the main dish at your Thanksgiving dinner? - Other (please
         specify)    object
         How is the main dish typically cooked?
         object
         dtype: object
```

data.apply(get_type).head()

DATA CLEANING - Let us clean up Income column

We need to convert string values representing income in **"How much total combined money did all members of your HOUSEHOLD earn last year"** column into numeric values. Check the unique values first

In [11]: column_name = "How much total combined money did all members of your HOUSEHOLD

```
data[column_name].value_counts(dropna=False)
```

Out[11]:

$25,000 to $49,999	180
Prefer not to answer	136
$50,000 to $74,999	135
$75,000 to $99,999	133
$100,000 to $124,999	111
$200,000 and up	80
$10,000 to $24,999	68
$0 to $9,999	66
$125,000 to $149,999	49
$150,000 to $174,999	40
NaN	33
$175,000 to $199,999	27

Name: How much total combined money did all members of your HOUSEHOLD earn last year?, dtype: int64

Looking at this, there are 4 different patterns for the values in the column: X to Y — an example is 25,000to49,999. We can convert this to a numeric value by extracting the numbers and averaging them. NaN We'll preserve NaN values, and not convert them at all. X and up — an example is $200,000 and up. We can convert this to a numeric value by extracting the number. Prefer not to answer We'll turn this into an NaN value.

In [12]:
```
import numpy as np

def clean_income(value):

    if value == "$200,000 and up":
        return 200000
    elif value == "Prefer not to answer":
        return np.nan
    elif isinstance(value, float) and math.isnan(value):
        return np.nan

    value = value.replace("$", "").replace(",", "")
    income_high, income_low = value.split(" to ")

    return (int(income_high) + int(income_low)) / 2
```

Now apply this function to the "How much total combined money did all members of your HOUSEHOLD earn last year?" column and put it in new column "income"

In [ ]:
```
column_name = 'How much total combined money did all members of your HOUSEHOLD
data["income"] = data[column_name].apply(clean_income)
```

In [13]:
```
data["income"].head()
```

Out[13]:

0	87499.5
1	62499.5
2	4999.5
3	200000.0
4	112499.5

Name: income, dtype: float64

Grouping Data with Pandas

Who earn more income?

Suppose, we want to find who earn more income?. Is it People eating homemade sauce or people eating canned sauce during the Thanksgiving Day?

Check unique values in column, "What type of cranberry saucedo you typically have?" first.

```
In [14]: data["What type of cranberry saucedo you typically have?"].value_counts()
Out[14]: Canned                  502
         Homemade                301
         None                    146
         Other (please specify)   25
         Name: What type of cranberry saucedo you typically have?, dtype: int64
```

We can now filter data to get two DataFrames, namely, homemade_df & canned_df, that only contain rows where the What type of cranberry saucedo you typically have? is Canned or Homemade, respectively

Create a datafrme by filtering values "Homemade"

```
In [15]: homemade_df = data[homemade_mask]
```

Create another datafrme by filtering values "Canned"

```
In [16]: canned_df = data[data[column_name] == 'Canned']
```

Now print mean income of homemade_df and canned_df for these two groups of people

```
In [17]: print(homemade["income"].mean())
         94878.1072874494
         print(Canned["income"].mean())
         83823.40340909091
```

Conclusion: Wow, great. We can understand from these values that **people who eat home made cranberry sauce earn more income** that the other group.

In [ ]:

Use groupby() and aggregate() to find out "Who earn more income?"

Split dataset based on "What type of cranberry saucedo you typically have?" column automatically into groups based on unique values

In [18]:
```
column_name = "What type of cranberry saucedo you typically have?"
grouped = data.groupby(column_name)
grouped
```

Out[18]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x00000197B706D7F0>
```

List out all groups that are created by groupby()

In [19]:
```
grouped.groups
```

Out[19]:
```
{'Canned': Int64Index([   4,    6,    8,   11,   12,   15,   18,   19,   26,
27,
                ...
                1040, 1041, 1042, 1044, 1045, 1046, 1047, 1051, 1054, 1057],
               dtype='int64', length=502),
 'Homemade': Int64Index([   2,    3,    5,    7,   13,   14,   16,   20,   2
1,   23,
                ...
                1016, 1017, 1025, 1027, 1030, 1034, 1048, 1049, 1053, 1056],
               dtype='int64', length=301),
 'None': Int64Index([   0,   17,   24,   29,   34,   36,   40,   47,   49,
51,
                ...
                 980,  981,  997, 1015, 1018, 1031, 1037, 1043, 1050, 1055],
               dtype='int64', length=146),
 'Other (please specify)': Int64Index([   1,    9,  154,  216,  221,  233,  2
49,  265,  301,  336,  380,
                435,  444,  447,  513,  550,  749,  750,  784,  807,  860,  87
2,
                905, 1000, 1007],
               dtype='int64')}
```

In [20]:
```
grouped.size()
```

Out[20]:
```
What type of cranberry saucedo you typically have?
Canned                    502
Homemade                  301
None                      146
Other (please specify)     25
dtype: int64
```

```
In [21]: for name, group in grouped:
             print(name)
             print(group.shape)
             print(type(group))
```

```
Canned
(502, 67)
<class 'pandas.core.frame.DataFrame'>
Homemade
(301, 67)
<class 'pandas.core.frame.DataFrame'>
None
(146, 67)
<class 'pandas.core.frame.DataFrame'>
Other (please specify)
(25, 67)
<class 'pandas.core.frame.DataFrame'>
```

Here each group is a DataFrame, and you can use any normal DataFrame methods on it. We can also extract a single column from a group. This will allow us to perform further computations just on that specific column:

```
In [22]: grouped["income"]
```

```
Out[22]: <pandas.core.groupby.generic.SeriesGroupBy object at 0x00000197B707D4E0>
```

```
In [23]: grouped["income"].size()
```

```
Out[23]: What type of cranberry saucedo you typically have?
         Canned                    502
         Homemade                  301
         None                      146
         Other (please specify)     25
         Name: income, dtype: int64
```

Aggregating values in groups

Spliting data into groups will not be sufficient. Real power comes when we can apply computation on each group.

Now, find out average income

We could find the average income for people who served each type of cranberry sauce. Extract income column from grouped DF and fine mean value for each group

In [24]: grouped["Income"].agg(np.mean)

Out[24]:
```
What type of cranberry saucedo you typically have?
Canned                     83823.403409
Homemade                   94878.107287
None                       78886.084034
Other (please specify)     86629.978261
Name: income, dtype: float64
```

If you want to consider all numberic attributes and find the mean for each group for every column in data, you can do as below.

In [25]: grouped.agg(np.mean)

Out[25]:

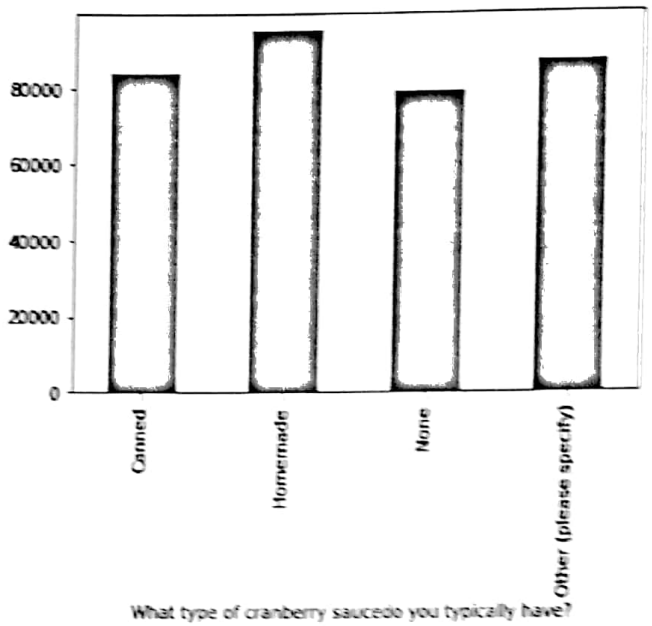
What type of cranberry saucedo you typically have?	RespondentID	gender	income
Canned	4.336699e+09	0.552846	83823.403409
Homemade	4.336792e+09	0.533101	94878.107287
None	4.336765e+09	0.517483	78886.084034
Other (please specify)	4.336763e+09	0.640000	86629.978261

Plotting the results of aggregation

What is the average income of each category?

In [26]: # matplotlib inline

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x197b70886d8>



Aggregating with multiple columns

Find the average income of people who eat Homemade cranberry sauce and Tofurkey

We need to apply groupby on two columns "What type of cranberry saucedo you typically have?" and "What is typically the main dish at your Thanksgiving dinner?"

In [27]: grouped = data.groupby(["What type of cranberry saucedo you typically have?", "What is typically the main dish at your Thanksgiving dinner?"])

Out[27]: grouped.agg(np.mean)

What type of cranberry saucedo you typically have?	What is typically the main dish at your Thanksgiving dinner?	RespondentID	gender	income
Canned	Chicken	4.336354e+09	0.333333	80999.600000
	Ham/Pork	4.336757e+09	0.642857	77499.535714
	I don't know	4.335987e+09	0.000000	4999.500000
	Other (please specify)	4.336682e+09	1.000000	53213.785714
	Roast beef	4.336254e+09	0.571429	25499.500000
	Tofurkey	4.337157e+09	0.714286	100713.857143
	Turkey	4.336705e+09	0.544444	85242.682045
Homemade	Chicken	4.336540e+09	0.750000	19999.500000
	Ham/Pork	4.337253e+09	0.250000	96874.625000
	I don't know	4.336084e+09	1.000000	NaN
	Other (please specify)	4.336863e+09	0.600000	55356.642857
	Roast beef	4.336174e+09	0.000000	33749.500000
	Tofurkey	4.336790e+09	0.666667	57916.166667
	Turducken	4.337475e+09	0.500000	200000.000000
	Turkey	4.336791e+09	0.531008	97690.147982
None	Chicken	4.336151e+09	0.500000	11249.500000
	Ham/Pork	4.336680e+09	0.444444	61249.500000
	I don't know	4.336412e+09	0.500000	33749.500000
	Other (please specify)	4.336688e+09	0.600000	119106.678571
	Roast beef	4.337424e+09	0.000000	162499.500000
	Tofurkey	4.336950e+09	0.500000	112499.500000
	Turducken	4.336739e+09	0.000000	NaN
	Turkey	4.336784e+09	0.523364	74606.275281
Other (please specify)	Ham/Pork	4.336465e+09	1.000000	87499.500000
	Other (please specify)	4.337335e+09	0.000000	124999.666667
	Tofurkey	4.336122e+09	1.000000	37499.500000
	Turkey	4.336724e+09	0.700000	82916.194444

As you can see above, we get a nice table that shows us the mean of each column for each group. This enables us to find some interesting patterns, such as:

- People who have Turducken and Homemade cranberry sauce seem to have high household incomes.
- People who eat Canned cranberry sauce tend to have lower incomes, but those who also have Roast Beef have the lowest incomes.
- It looks like there's one person who has Canned cranberry sauce and doesn't know what type of main dish he's having.

Aggregating with multiple functions

Find sum, mean and standard deviation of each group in the income column of grouped dataframe

In [28]: grouped["Income"].agg([np.mean, np.sum, np.std])

Out[28]:

		mean	sum	std
What type of cranberry saucedo you typically have?	**What is typically the main dish at your Thanksgiving dinner?**			
Canned	Chicken	80999.600000	404998.0	75779.481362
	Ham/Pork	77499.535714	1084993.5	56645.063944
	I don't know	4999.500000	4999.5	NaN
	Other (please specify)	53213.785714	372496.5	29780.946290
	Roast beef	25499.500000	127497.5	24584.039638
	Tofurkey	100713.857143	704997.0	61361.484439
	Turkey	85242.682045	34182315.5	56687.436122
Homemade	Chicken	19999.500000	59998.5	16393.696311
	Ham/Pork	96874.625000	387498.5	77308.452806
	I don't know	NaN	0.0	NaN

One of the limitations of aggregation is that each function has to return a single number. While we can perform computations like finding the mean, we can't for example, call value_counts to get the exact count of a category. We can do this using the pandas.GroupBy.apply method. This method will apply a function to each group, then combine the results.

Find the number of people who live in each area type (Rural, Suburban, etc) who eat different kinds of main dishes for Thanksgiving

In [29]:

Out[29]:

How would you describe where you live?	What is typically the main dish at your Thanksgiving dinner?	
Rural	Turkey	189
	Other (please specify)	9
	Ham/Pork	7
	Tofurkey	3
	I don't know	3
	Turducken	2
	Chicken	2
	Roast beef	1
Suburban	Turkey	449
	Ham/Pork	17
	Other (please specify)	13
	Tofurkey	9
	Roast beef	3
	Chicken	3
	I don't know	1
	Turducken	1
Urban	Turkey	198
	Other (please specify)	13
	Tofurkey	8
	Chicken	7
	Roast beef	6
	Ham/Pork	4

Name: What is typically the main dish at your Thanksgiving dinner?, dtype: int64

In [ ]: