Week 2 Lab Introduction to Statistics in Python

Import all relevant libraries and functions

[ ]
```
import numpy as np
from scipy import stats
import math as mt
import random as rnd
```

Calculating mean in Python

[ ]
```
# Number of pets each person owns
sample = [1, 3, 2, 5, 7, 0, 2, 3]
mean = sum(sample) / len(sample)
print(mean) # prints 2.875
```

Solve the following question using the sample code above

1. A group of students were given a short mental arithmetic test. Their scores were 7, 6, 7, 5, 6, 8, 5, 7, 8, 9.

   Calculate the mean.

Calculating weighted mean

[ ]
```
# Three exams of .20 weight each and final exam of .4
0 weight
sample = [90, 80, 63, 87]
weights = [.20, .20, .20, .40]
weighted_mean = sum(s * w for s,w in zip(sample, weigh
ts)) / sum(weights)
print(weighted_mean) # prints 81.4
```

Solve the following question using the sample code above

1. The information below presents the weights of different decision features of an automobile. With the help of this information, we need to calculate the weighted average. Quantity Weight Safety - 8/10 weight = 40% Comfort - 6/10 weight = 20% Fuel mileage - 5/10 weight = 30% Exterior looks - 8/10 weight = 10%

Calculating median

[ ]
```
# Number of pets each person owns
sample = [0, 1, 5, 7, 9, 10, 14]

def median(values):
    ordered = sorted(values)
    print(ordered)
    n = len(ordered)
    mid = int(n / 2) - 1 if n % 2 == 0 else int(n/2)

    if n % 2 == 0:
        return (ordered[mid] + ordered[mid+1]) / 2.0
    else:
        return ordered[mid]

print(median(sample)) # prints 7
```

Question: Find the median for the following data set: 102, 56, 34, 99, 89, 101, 10.

Calculating Mode

[ ]
```
# Number of pets each person owns
from collections import defaultdict
```

```
sample = [1, 3, 2, 5, 7, 0, 2, 3]

def mode(values):
    counts = defaultdict(lambda: 0)

    for s in values:
        counts[s] += 1

    max_count = max(counts.values())
    modes = [v for v in set(values) if counts[v] == max_count]
    return modes

print(mode(sample)) # [2, 3]
```

Question: Here are the weights, in kg, of 8 people. 63 65 65 70 72 86 90 97. Calculate the mode using the sample code above.

Calculating Standard deviation and variance

[ ]

```
# Number of pets each person owns
data = [0, 1, 5, 7, 9, 10, 14]

def variance(values):
    mean = sum(values) / len(values)
    _variance = sum((v - mean) ** 2 for v in values) / len(values)
    return _variance

print(variance(data))  # prints 21.387755102040813

from math import sqrt

# Number of pets each person owns
data = [0, 1, 5, 7, 9, 10, 14]

def variance(values):
    mean = sum(values) / len(values)
    _variance = sum((v - mean) ** 2 for v in values) / len(values)
    return _variance
```

```
def std_dev(values):
   return sqrt(variance(values))

print(std_dev(data))  # prints 4.624689730353898
```

Question: Find the variance and standard deviation of the following scores on an exam: 92, 95, 85, 80, 75, 50 using the sample code given above.

Create a student data set

[4]
```
import pandas as pd
student_df = pd.DataFrame({'name': ['Alice', 'Bob', 'Carol', \
                                    'Dan', 'Eli', 'Fran'],\
                          'gender': ['female', 'male', \
                                     'female', 'male', \
                                     'male', 'female'],\
                          'class': ['FY', 'SO', 'SR', \
                                    'SO',' JR', 'SR'],\
                          'gpa': [90, 93, 97, 89, 95, 92],\
                          'num_classes': [4, 3, 4, 4, 3, 2]})
student_df
```

In a new code cell, create a new attribute named 'female_flag' whose individual cells should hold the Boolean value True if the corresponding student is female, and False otherwise.

[6]
```
student_df['female_flag'] = student_df['gender'] == 'female'
student_df = student_df.drop('gender', axis=1)
student_df
```

Using group-by

Showing mean GPA according to gender

[8]
```
gender_group = student_df.groupby('female_flag')
gender_group['gpa'].mean()
```

[ ]
Showing number of students of each gender

[9]
```
gender_group['num_classes'].sum()
```

For Data visualisation, import the following library

[ ]
```
>>> import matplotlib.pyplot as plt
```

Using scatter plot function

[10]
```
x = [1, 2, 3, 1.5, 2]
y = [-1, 5, 2, 3, 0]
import matplotlib.pyplot as plt
plt.scatter(x, y)
plt.show()
```

Say we have an attribute in our dataset that contains the sample data stored in x. We can call plt.hist() on x to plot the distribution of the values in the attribute like so:

[12]
```
x = np.random.randn(100)
plt.hist(x)
plt.show()
```

[13]

```
student_df['gpa'].plot.hist()
plt.show()
```

Do the following exercise (source link: https://learning.oreilly.com/library/view/the-statistics-and/9781800209763/)

Analyzing the Communities and Crime Dataset

In this activity, we will practice some basic data processing and analysis techniques on a dataset available online called *Communities and Crime*, with the hope of consolidating our knowledge and techniques. Specifically, we will process missing values in the dataset, iterate through the attributes, and visualize the distribution of their values.

First, we need to download this dataset to our local environment, which can be accessed on this page: https://packt.live/31C5yrZ

The dataset should have the name **CommViolPredUnnormalizedData.txt**. From the same directory as this dataset text file, create a new Jupyter notebook. Now, perform the following steps:

1. As a first step, import the libraries that we will be using: pandas, NumPy, and Matplotlib.
2. Read in the dataset from the text file using pandas and print out the first five rows by calling the **head()** method on the **DataFrame** object.
3. Loop through all the columns in the dataset and print them out line by line. At the end of the loop, also print out the total number of columns.
4. Notice that missing values are indicated as **'?'** in different cells of the dataset. Call the **replace()**method on the **DataFrame** object to replace that character with **np.nan** to faithfully represent missing values in Python.
5. Print out the list of columns in the dataset and their respective numbers of missing values using **df.isnull().sum()**, where **df** is the variable name of the **DataFrame** object.
6. Using the **df.isnull().sum()[column_name]** syntax (where **column_name** is the name of the column we are interested in), print out the number of missing values in the **NumStreet** and **PolicPerPop**columns.

7. Compute a **DataFrame** object that contains a list of values in the **state** attribute and their respective counts. Then, use the **DataFrame.plot.bar()**method to visualize that information in a bar graph.
8. Observe that, with the default scale of the plot, the labels on the x-axis are overlapping. Address this problem by making the plot bigger with the **f, ax = plt.subplots(figsize=(15, 10))** command. This should be placed at the beginning of any plotting commands.
9. Using the same value count **DataFrame** object that we used previously, call the **DataFrame.plot.pie()** method to create a corresponding pie chart. Adjust the figure size to ensure that the labels for your graph are displayed correctly.
10. Create a histogram representing the distribution of the population sizes in areas in the dataset (included in the **population** attribute). Adjust the figure size to ensure that the labels for your graph are displayed correctly.

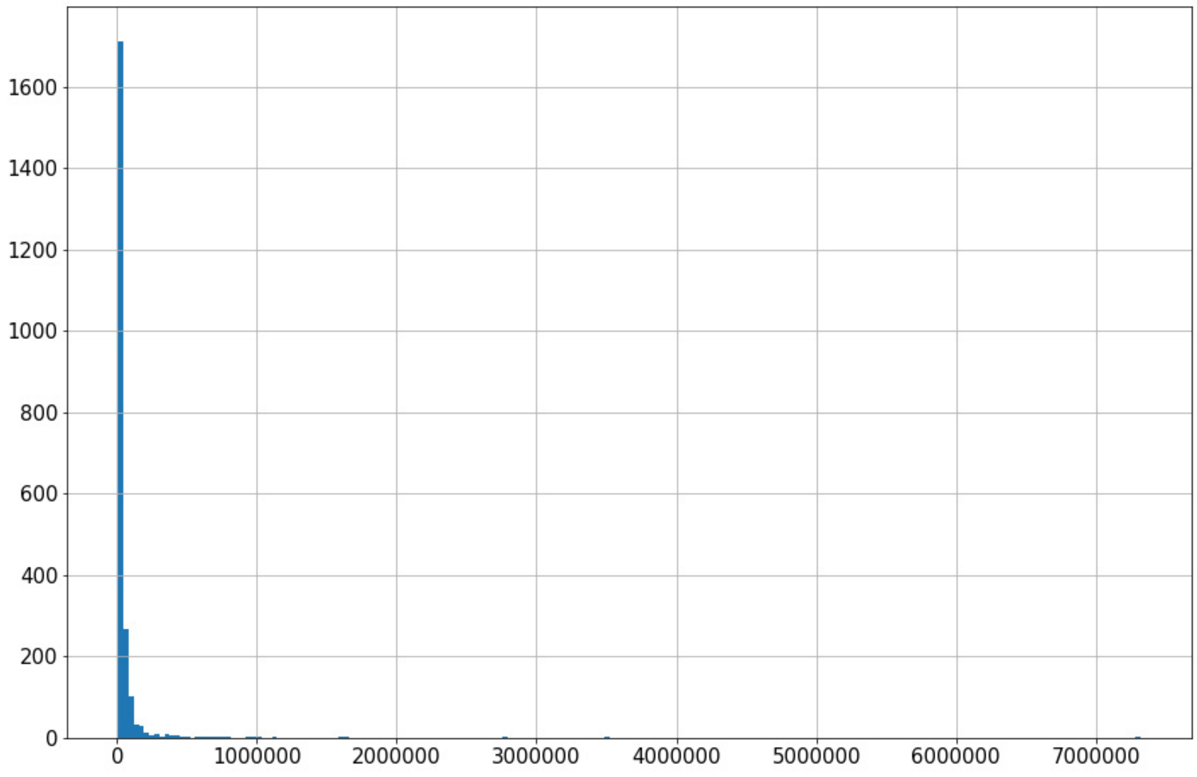

Figure 2.21: Histogram for population distribution

11. Create an equivalent histogram to visualize the distribution of household sizes in the dataset (included in the **householdsize** attribute).

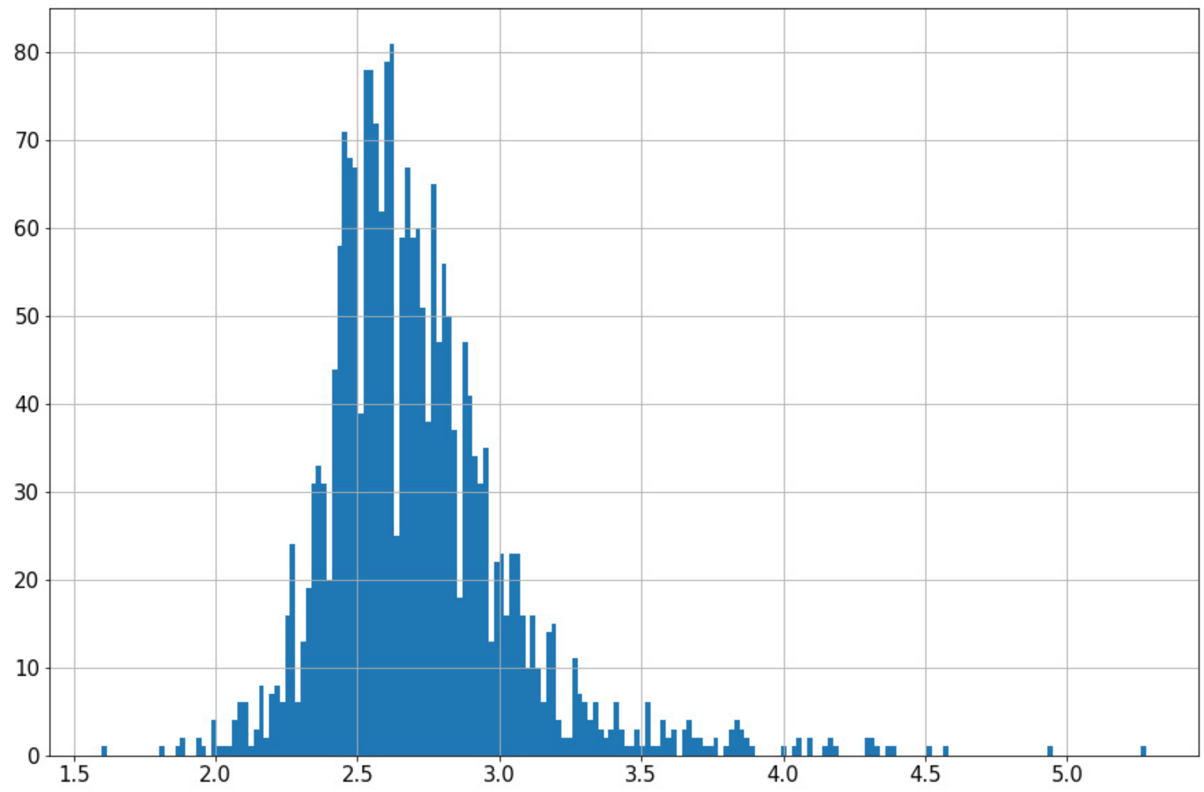

Figure 2.22: Histogram for household size distribution