Annotated follow-along guide_Compute descriptive statistics with Python

October 8, 2024

1 Annotated follow-along guide: Compute descriptive statistics with Python

This notebook contains the code used in the following instructional video: Compute descriptive statistics with Python.

1.1 Introduction

Throughout this notebook, we will practice computing descriptive statistics to explore and summarize a dataset. Before getting started, watch the associated instructional video and complete the in-video question. All of the code we will be implementing and related instructions are contained in this notebook.

1.2 Overview

Earlier in the program, you learned about the process of exploratory data analysis, or EDA, from discovering to presenting your data. Whenever a data professional works with a new dataset, the first step is to understand the context of the data during the discovering stage. Often, this involves discussing the data with project stakeholders and reading documentation about the dataset and the data collection process. After that, the data professional moves on to data cleaning and addresses issues like missing data, incorrect values, and irrelevant data. Computing descriptive stats is a common step to take after data cleaning.

In this notebook, we will use descriptive stats to get a basic understanding of the literacy rate data for each district in your education dataset.

1.3 Import packages and libraries

Before getting started, we will need to import all the required libraries and extensions. Throughout the course, we will be using pandas and numpy for operations and matplotlib for plotting.

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

[2]:
```
education_districtwise = pd.read_csv('education_districtwise.csv')
```

1.4 Explore the data

Let's start with the `head()`function to get a quick overview of the dataset. Recall that `head()` will return as many rows of data as you input into the variable field.

[3]:
```
education_districtwise.head(10)
```

[3]:

	DISTNAME	STATNAME	BLOCKS	VILLAGES	CLUSTERS	TOTPOPULAT	OVERALL_LI
0	DISTRICT32	STATE1	13	391	104	875564.0	66.92
1	DISTRICT649	STATE1	18	678	144	1015503.0	66.93
2	DISTRICT229	STATE1	8	94	65	1269751.0	71.21
3	DISTRICT259	STATE1	13	523	104	735753.0	57.98
4	DISTRICT486	STATE1	8	359	64	570060.0	65.00
5	DISTRICT323	STATE1	12	523	96	1070144.0	64.32
6	DISTRICT114	STATE1	6	110	49	147104.0	80.48
7	DISTRICT438	STATE1	7	134	54	143388.0	74.49
8	DISTRICT610	STATE1	10	388	80	409576.0	65.97
9	DISTRICT476	STATE1	11	361	86	555357.0	69.90

Note: To interpret this data correctly, it's important to understand that each row, or observation, refers to a different *district* (and not, for example, to a state or a village). So, the `VILLAGES` column indicates how many villages are in each district, the `TOTPOPULAT` column indicates the population for each district, and the `OVERALL_LI` column indicates the literacy rate for each district.

1.4.1 Use describe() to compute descriptive stats

Now that we have a better understanding of the dataset, let's use Python to compute descriptive stats.

When computing descriptive stats in Python, the most useful function to know is `describe()`. Data professionals use the `describe()` function as a convenient way to calculate many key stats all at once. For a numeric column, `describe()` gives you the following output:

- `count`: Number of non-NA/null observations
- `mean`: The arithmetic average
- `std`: The standard deviation
- `min`: The smallest (minimum) value
- `25%`: The first quartile (25th percentile)
- `50%`: The median (50th percentile)
- `75%`: The third quartile (75th percentile)
- `max`: The largest (maximum) value

Reference: pandas.DataFrame.describe

Our main interest is the literacy rate. This data is contained in the `OVERALL_LI` column, which shows the literacy rate for each district in the nation. Use the `describe()` function to reveal key

stats about literacy rate.

[4]:
```
education_districtwise['OVERALL_LI'].describe()
```

[4]:
```
count    634.000000
mean      73.395189
std       10.098460
min       37.220000
25%       66.437500
50%       73.490000
75%       80.815000
max       98.760000
Name: OVERALL_LI, dtype: float64
```

The summary of stats gives us valuable information about the overall literacy rate. For example, the mean helps to clarify the center of your dataset; we now know the average literacy rate is about 73% for all districts. This information is useful in itself and also as a basis for comparison. Knowing the mean literacy rate for *all* districts helps us understand which individual districts are significantly above or below the mean.

Note: `describe()` excludes missing values (`NaN`) in the dataset from consideration. You may notice that the count, or the number of observations for `OVERALL_LI` (634), is fewer than the number of rows in the dataset (680). Dealing with missing values is a complex issue outside the scope of this course.

You can also use the `describe()` function for a column with categorical data, like the `STATNAME` column.

For a categorical column, `describe()` gives you the following output:

- `count`: Number of non-NA/null observations
- `unique`: Number of unique values
- `top`: The most common value (the mode)
- `freq`: The frequency of the most common value

[5]:
```
education_districtwise['STATNAME'].describe()
```

[5]:
```
count          680
unique          36
top       STATE21
freq            75
Name: STATNAME, dtype: object
```

The `unique` category indicates that there are 36 states in our dataset. The `top` category indicates that `STATE21` is the most commonly occurring value, or mode. The `frequency` category tells you that `STATE21` appears in 75 rows, which means it includes 75 different districts.

This information may be helpful in determining which states will need more educational resources based on their number of districts.

1.4.2 Functions for stats

The `describe()` function is also useful because it reveals a variety of key stats all at once. Python also has separate functions for the mean, median, standard deviation, minimum, and maximum. Earlier in the program, you used `mean()` and `median()` to detect outliers. These individual functions are also useful if you want to do further computations based on descriptive stats. For example, you can use the `min()` and `max()` functions together to compute the range of your data.

1.4.3 Use max() and min() to compute range

Recall that the **range** is the difference between the largest and smallest values in a dataset. In other words, range = max - min. You can use `max()` and `min()` to compute the range for the literacy rate of all districts in your dataset.

[6]:
```
range_overall_li = education_districtwise['OVERALL_LI'].max() -␣
↪education_districtwise['OVERALL_LI'].min()
range_overall_li
```

[6]:
```
61.540000000000006
```

The range in literacy rates for all districts is about 61.5 percentage points.

This large difference tells you that some districts have much higher literacy rates than others. Later on, you will continue to analyze this data, and you can discover which districts have the lowest literacy rates. This will help the government better understand literacy rates nationally and build on their successful educational programs.

1.5 Conclusion

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.

You now understand how to compute descriptive statistics with Python. Going forward, you can start using descriptive statistics to explore and summarize your own datasets.