A Quick Guide on Descriptive Statistics using Pandas and Seaborn

Bee Guan Teo
Nov 2, 2019 · 13 min read ★

Image by rawpixel from Pixabay

Descriptive statistics is a study of data analysis to describe, show or summarize data in a meaningful way. It involves the calculation of various measures such as the **measure of center**, **the measure of variability**, **percentiles** and also the **construction of tables & graphs.**

handle those calculations of descriptive measures and to construct tables & graphs will be demonstrated using **Pandas and Seaborn.** Pandas and Seaborn are Python libraries which are commonly used for statistical analysis and visualization.

. . .

Prerequisites Python libraries

- **Pandas**
- **Matplolib**
- **Seaborn**

Note:

This is highly recommended to use **Jupyter Notebook** to follow all the coding tasks in this article. All the Python scripts presented here are written and tested in a **Jupyter Notebook**. You can refer to Jupyter official site for further instructions to set up Jupyter Notebook in your machine.

. . .

Datasets

We will use a public dataset relevant to house prices in Melbourne, "*MELBOURNE_HOUSE_PRICES_LESS.csv*" as our sample data. The dataset is available at Kaggle.

. . .

Topic 1: Types of Data

1.1 Numerical and categorical data

Prior to starting any technical calculation and plotting works, this is very important to understand the type of data which are commonly seen in a statistical study. There are

- **Categorical data:** non-numerical information such as gender, race, religion, marital status etc.
- **Numerical data:** measurement or count such as height, weight, age, salary, number of children, etc

1.2 Python code in practice

In a statistical analysis or a data science project, the data (either categorical or numerical or both)are often stored in a tabular format (like a spreadsheet) in a CSV file.

To get a better idea, let's get started by visualizing the data in our CSV file (*MELBOURNE_HOUSE_PRICES_LESS.csv*) using the Pandas library.

```
1  import pandas as pd
2
3  df = pd.read_csv("MELBOURNE_HOUSE_PRICES_LESS.csv")
4  df.head()
```

read_data.py hosted with ♡ by GitHub view raw

Read and show the first five rows of data

- **Line 1:** Import *Pandas* library
- **Line 3:** Use *read_csv* method to read the raw data in the CSV file into a *data frame, df* . The *data frame* is a two-dimensional array-like data structure for statistical and machine learning models.
- **Line 4:** Use *head()* method of the data frame to show the first five rows of the data.

When we run the codes in *Jupyter Notebook*, you shall see the data is presented in a table which consists of 13 variables (columns).

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Postcode	Regionname	Propertycount	Distance	CouncilArea
0	Abbotsford	49 Lithgow St	3	h	1490000.0	S	Jellis	1/04/2017	3067	Northern Metropolitan	4019	3.0	Yarra City Council
1	Abbotsford	59A Turner St	3	h	1220000.0	S	Marshall	1/04/2017	3067	Northern Metropolitan	4019	3.0	Yarra City Council
2	Abbotsford	119B Yarra St	3	h	1420000.0	S	Nelson	1/04/2017	3067	Northern Metropolitan	4019	3.0	Yarra City Council
													Moonee

The first five rows of data in a tabular format

Pandas also offer another useful method, *info(),* to get further details of the data type for each variable in our dataset. In the same Jupyter Notebook, just create a new cell below the previous codes and add the following line of code and run it:

```
1  df.info()
```

info.py hosted with ♡ by GitHub view raw

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63023 entries, 0 to 63022
Data columns (total 13 columns):
Suburb          63023 non-null object
Address         63023 non-null object
Rooms           63023 non-null int64
Type            63023 non-null object
Price           48433 non-null float64
Method          63023 non-null object
SellerG         63023 non-null object
Date            63023 non-null object
Postcode        63023 non-null int64
Regionname      63023 non-null object
Propertycount   63023 non-null int64
Distance        63023 non-null float64
CouncilArea     63023 non-null object
dtypes: float64(2), int64(3), object(8)
memory usage: 6.3+ MB
```

Details of data type for each column

The result reveals a total of 63023 entries in the dataset. Generally, the columns tied with data type "**int64**" and "**float64**" denotes numerical data while data type "**object**" denotes categorical data.

*The only exception is "Postcode". Although the postcode is represented as a number (int64), it doesn't necessarily make it quantitative. Postcode is just numbers applied to categorical data.

categorical data.

Categorical data	Numerical data
Suburb	Rooms
Address	Price
Type	Propertycount
Method	Distance
SellerG	
Postcode	
Regionname	
CouncilArea	

Categorical Data vs Numerical Data

. . .

Topic 2: Measure of Center

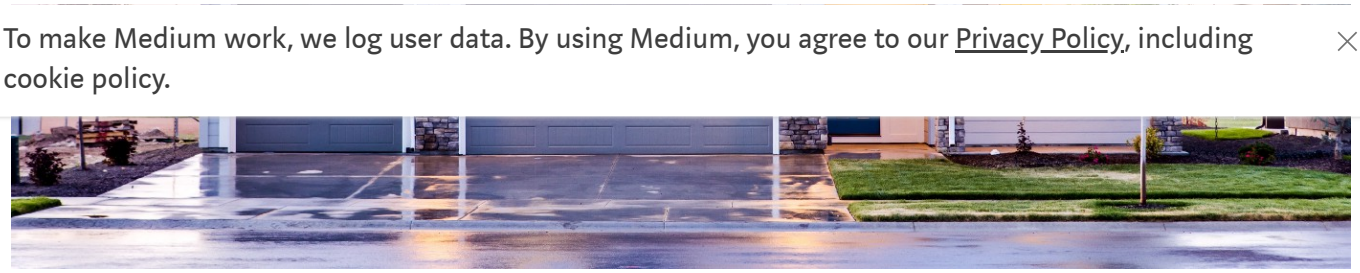


Photo by Binyamin Mellish from Pexels

One common way to summarize our numerical data is to find out the central tendency of our data. For example, we might question "What is the most typical value of the house price in our dataset?". To address this question, we can resort to the two most common measures of center: **mean and median.**

2.1 Mean

Mean is an average of all the numbers. The steps required to calculate a mean are:

1. sum up all the values of a target variable in the dataset
2. divide the sum by the number of values

For example, if we have a set of five values, [70, 60, 85, 80, 92],

$$\text{Mean} = (70 + 60 + 85 + 80 + 92) / 5 = 77.4$$

However, sometimes a mean can be misleading and may not effectively show a typical value in our dataset. This is because a mean might be influenced by the **outliers**.

Outliers are the numbers which are either extremely high or extremely low compared to the rest of the numbers in a dataset.

Let's us look into another two datasets, [70, 60, 1, 80, 92] and [70, 60, 300, 80, 92].

$$\text{Mean\_1} = (70 + 60 + 1 + 80 + 92)/ 5 = 60.6$$

$$\text{Mean\_2} = (70 + 60 + 300 + 80 + 92) / 5 = 120.4$$

2.2 Median

Median is the middle value of a sorted list of numbers. The steps required to get a median from a list of numbers are:

1. sort the numbers from smallest to highest
2. if the list has an odd number of values, the value in the middle position is the median
3. if the list has an even number of values, the average of the two values in the middle will be the median

The followings are two examples to show how we can get the median from an odd number of values and an even number of values.

```
odd_set = [30, 45, 67, 87, 94, 102, 124]

median = 87
```

If we have a set of eight values, [30, 45, 67, 87, 94, 102, 124],

```
even_set = [30, 45, 67, 87, 94, 102, 124]

median = (87 + 94) / 2 = 90.5
```

Note: A median is not influenced by the outliers.

The choice we make either to use mean or median as a measure of center is dependent on the question we address. As a general rule, we should report both mean and median in our statistical study and let readers interpret the results themselves.

median(). Let's us use Pandas to get the mean and median of our house price from the dataset.

```
1  average = df['Price'].mean()
2  print(average)
3
4  med = df['Price'].median()
5  print(med)
```

mean_median.py hosted with ♡ by GitHub view raw

Get mean and median using Pandas

Line 1 & 4: *df['Price']* will select the column where the price values are populated. It is followed with **a dot syntax** to call the method *mean() and median(), respectively.*

Line 2 & 5: Print the mean and median.

Note that the Pandas *mean* and *median* methods have already encapsulated the complicated formula and calculation for us. All what we need is just to ensure that we select the right column from our dataset and call the methods to get mean and median. The output is shown below:

```
997898.2414882415
830000.0
```

The mean and median of house prices

. . .

Topic 3: Measure of Variation

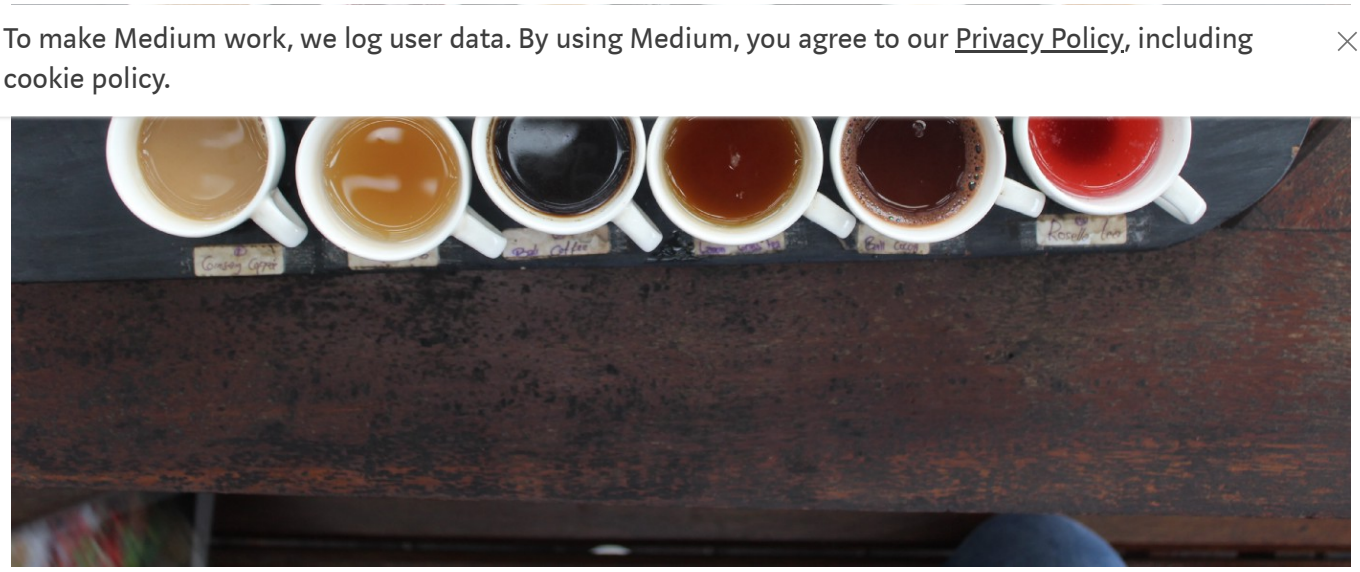


Photo by Pritesh Sudra on Unsplash

Variation is always observed in a dataset. This is very unusual to see an entire set of numbers share the exact same values as follows:

```
Sample Dataset 1: 5 5 5 5 5 5 5 5 5

Sample Dataset 2: 1 1 1 1 1 1 1 1 1
```

When comparing the difference/variability between two datasets, mean and median are not a good option to serve for that purpose. To explain this further, let us look at the two examples below

```
Sample Dataset 1:  100   200  300   400   500

Sample Dataset 2:    1      2   300   499   698
```

Both datasets above share the same mean and median which is **300**. However, they have a different level of variation. The numbers in the first dataset have a lower variation than the second one despite both of them share the same mean and median. Hence, we need another kind of measurement to examine the variability of our dataset.

standard deviation (SD). The SD is just a measurement to tell how a set of values spread out from their mean. A low SD shows that the values are close to the mean and a high SD shows a high diversion from the mean.

The steps to calculate SD are as follows:

1. Calculate the mean of a dataset
2. For each number in the dataset, subtract it with the mean
3. Square the difference obtained from Step 2
4. Sum up the results from Step 3
5. Divide the summation from Step 4 by the number of values in the dataset minus one
6. Square root the result from Step 5

Sample Dataset 1 3 5 7 9

Step 1: Mean = (1 + 3 + 5 + 7 + 9) / 5 = 5

Step 2: (1 – 5) | (3 - 5) | (5 - 5) | (7 - 5) | (9 - 5)

Step 3: $(1-5)^2$ | $(3-5)^2$ | $(5-5)^2$ | $(7-5)^2$ | $(9-5)^2$

Step 4: $(1-5)^2 + (3-5)^2 + (5-5)^2 + (7-5)^2 + (9-5)^2$

Step 5: $((1-5)^2 + (3-5)^2 + (5-5)^2 + (7-5)^2 + (9-5)^2) / (5-1)$

Step 6: $\sqrt{((1-5)^2 + (3-5)^2 + (5-5)^2 + (7-5)^2 + (9-5)^2) / (5-1)} = 3.16$

Note:

- SD must be a positive number
- SD is affected by outliers as its calculation is based on the mean
- The smallest possible value of SD is zero. If SD is zero, all the numbers in a dataset share the same value.

Pandas method to calculate the SD for our housing price.

```
1  standard_deviation = df['Price'].std()
2  print(standard_deviation)
```

sd_method.py hosted with ♡ by GitHub view raw

Line 1: call the method std() to calculate SD of house price

Again, we can see the Pandas *std()* method has already encapsulated the complex SD calculation for us and we can obtain the SD value on the fly.

```
593498.9190372769
```

SD of house price

. . .

Topic 4: Graphical way to show the numerical data distribution

The sections above cover two examples of descriptive measures (measure of center & measure of variation) **using a single value**. In this section, we are going to see how we can also explore the data distribution using **a graphical way**.

4.1 Boxplot

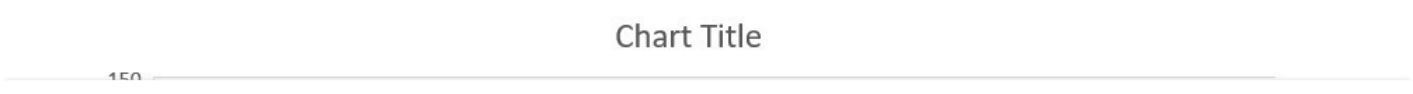
One approach to reveal the data distribution is to find a **Five-Number Summary** from our dataset**.** The Five-Number Summary includes:

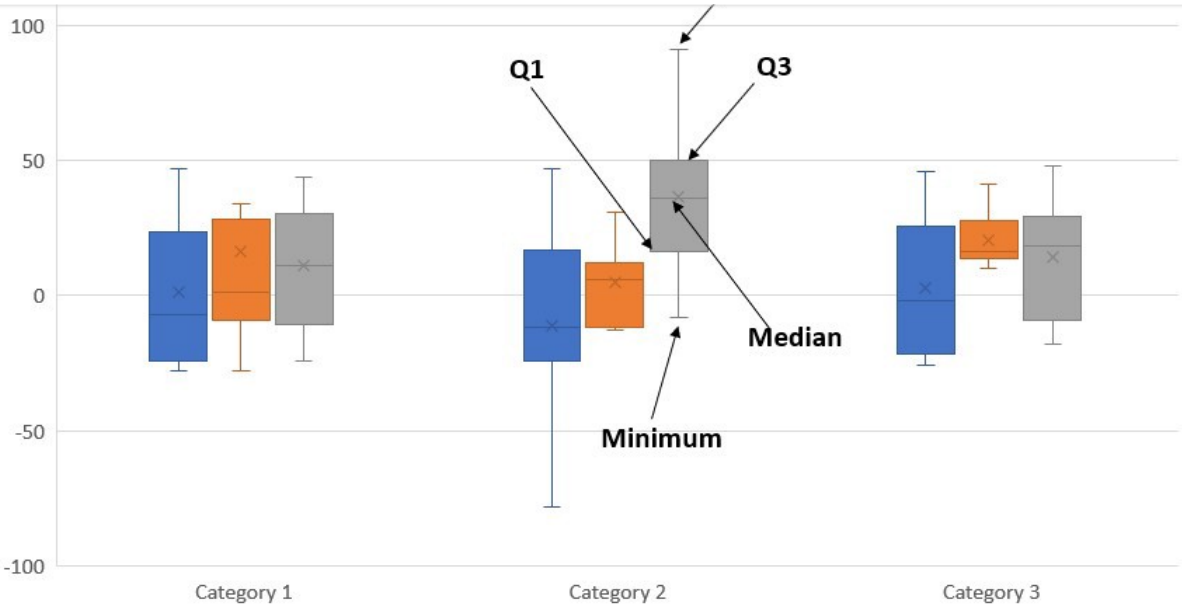
1. The minimum
2. The 25th percentile or the first quartile (Q1)
3. The median
4. The 75th percentile or the third quartile (Q3)
5. The maximum

This is necessary here to explain the meaning of "**percentile**".

> "A percentile is a measure used in statistics indicating the value below which a given percentage of observations in a group of observations falls. For example, the 20th percentile is the value below which 20% of the observations may be found." (Source: Wikipedia)

The Five-Number Summary provides a quick way to enable us to roughly locate the minimum, 25th percentile, median, 75th percentile, and the maximum number from our dataset. One graphical way to present this Five-Number Summary is by creating a **boxplot**.





Samples of boxplot

4.2 Histogram

A histogram is a graphical display that uses rectangular bars to show the frequency distribution of a set of numerical data. The histogram enables us to visualize the underlying distribution pattern of our data.

The steps required to build a histogram are as follows:

1. Split our data into intervals which are called **bins**.
2. Record number of occurrences (frequency) for each number in your dataset and tabulate them into a **Frequency Table**.
3. Plot a histogram based on the Frequency Table obtained from Step 2.

Sample dataset: 1 3 3 3 5 6 6 6 7 8 8 9 9 9 9 9 10 10 10 10 10 10 11 11 11 11 11 11 12 12 12 12 12 12 13 13 13 13 13 14 14 14 14 14 14 15 15 15 15 15 15 15 15 16 16 16 16 17 17 17 17 17 17 18 18 18 18 19 19 19 20 21 22 22 24 24

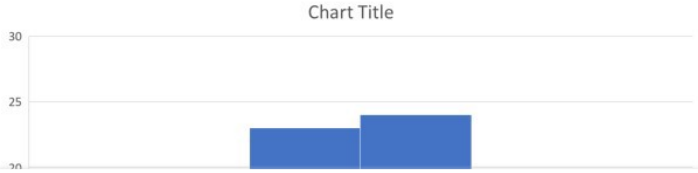
Step 1: [1, 5] (5, 9] (9 ,13] (13, 17] (17, 21] (21, 25]

Bins

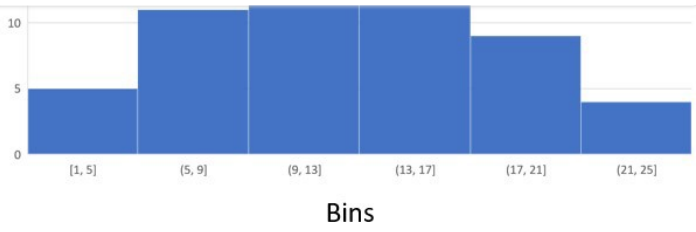
Step 2:

Bin	Frequency
[1, 5]	5
(5, 9]	11

Step 3:



(17, 21]	9
(21, 25]	4



Steps to create a histogram

4.3 Python code in practice

In this section, we will use the Seaborn library to create a boxplot and a histogram for our housing price.

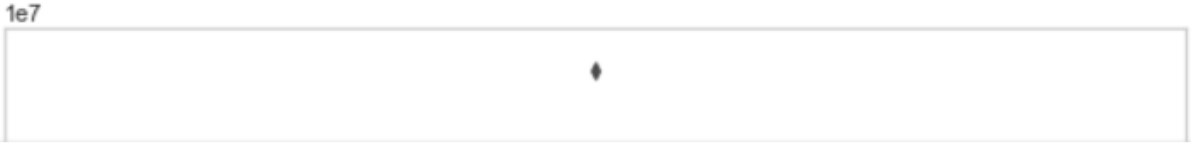
Firstly, let us start with the boxplot.

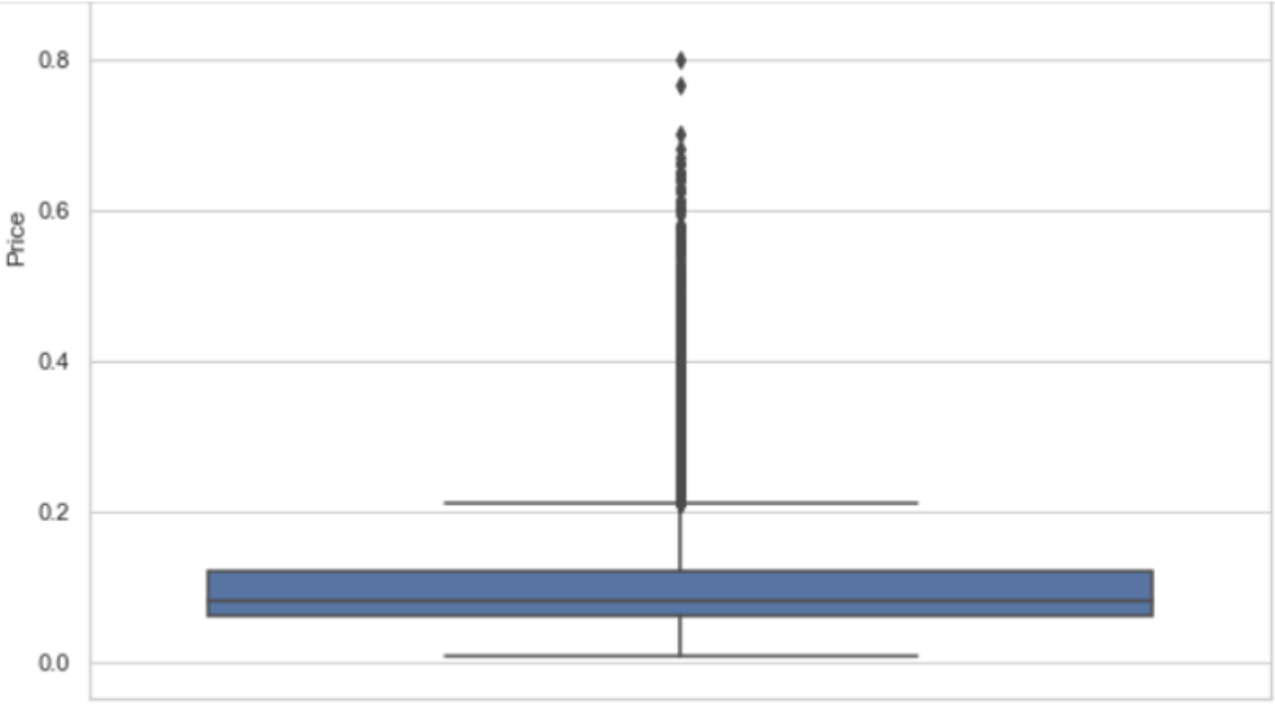
```
1  import matplotlib.pyplot as plt
2  import seaborn as sns
3
4  %matplotlib inline
5
6  sns.set(style="whitegrid")
7  plt.figure(figsize=(10,8))
8  ax = sns.boxplot(x='Price', data=df, orient="v")
```

create_boxplot.py hosted with ♡ by GitHub view raw

Create a boxplot using Seaborn

- **Line 1 & 2**: import Matplolib and Seaborn library
- **Line 4:** A magic function that enables our graph rendered in the Jupyter Notebook
- **Line 6:** Set a Seaborn theme
- **Line 7:** Set the figure size of the plot
- **Line 8:** Use *Seaborn boxplot()* method to generate a boxplot. We set the "*Price*" as the inputs for plotting data. The "*v*" value in parameter "*orient*" is to render our boxplot in a vertical style.





The Seaborn Boxplot of housing price

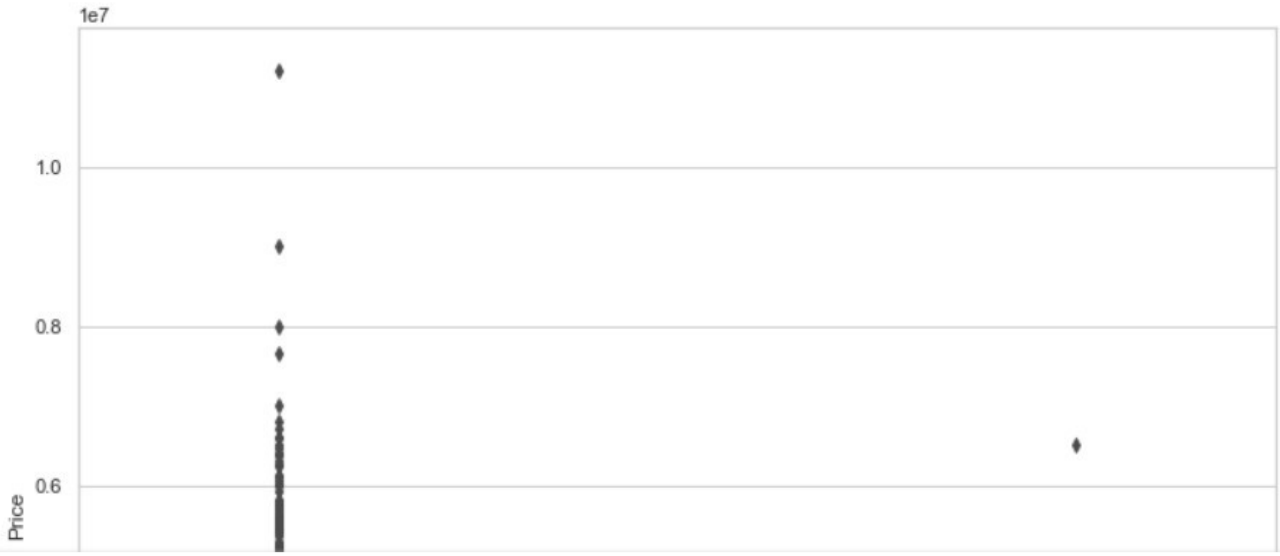
In fact, there are three different types of houses in our dataset which are "H" — House, "U" — Unit and "T" — Townhouse. We can create a box plot for each type of house. To do so, we just need to add another parameter, "y" in our *Seaborn boxplot()* method.

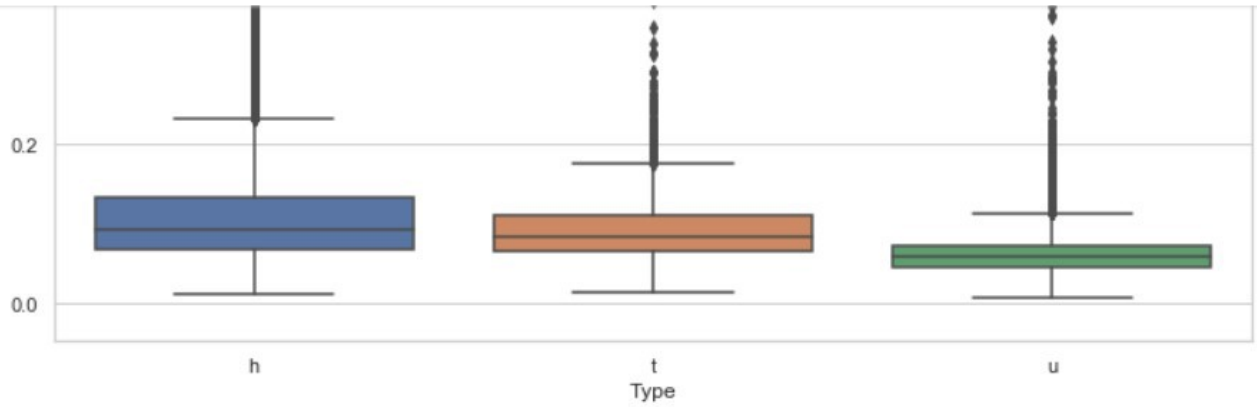
```
1  ax = sns.boxplot(x='Type', y='Price', data=df, orient="v")
```

several_boxplots.py hosted with ♡ by GitHub view raw

Create Seaborn boxplot for each type of house

- **Line 1:** Set "*Type*" as x-axis data and "*Price*" as y-axis data.





Seaborn boxplot for each type of house

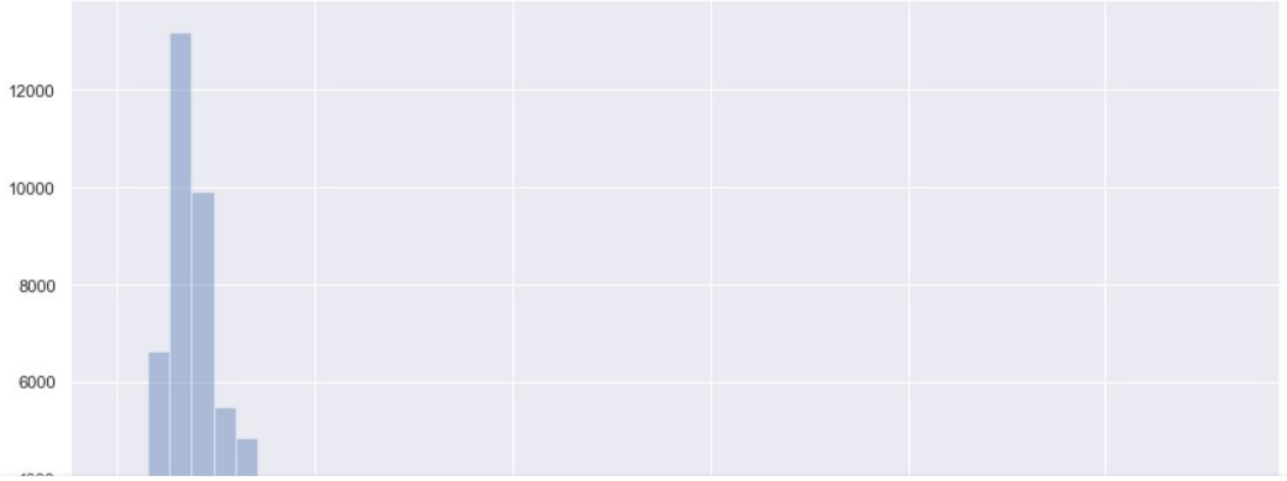
Now, let's try to build a histogram to observe the data distribution for our housing price data. We will use *Seaborn distplot()* to create our histogram.

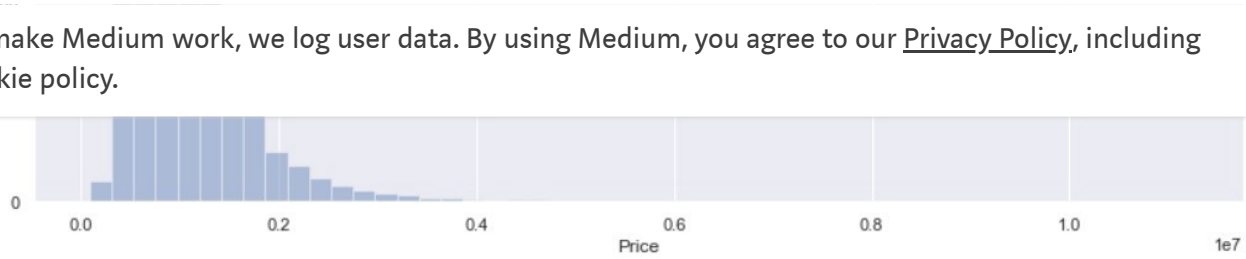
```
1  filter_data = df.dropna(subset=['Price'])
2  plt.figure(figsize=(14,8))
3  sns.distplot(filter_data['Price'], kde=False)
```

create_histogram.py hosted with ♡ by GitHub view raw

Create Seaborn histogram

- **Line 1:** This is a necessary step to use "dropna()" method to remove all the null values from the column "Price" in our housing dataset. Seaborn won't be able to generate a histogram if there is any null value exist in a column.
- **Line 2:** Set the figure size of the histogram plot.
- **Line 3:** Use *distplot()* method to generate the histogram. There is only one required input which is the filtered housing price data (without any null values).





Seaborn Histogram for House Price

. . .

Topic 5: Exploring Categorical Data

Photo by Providence Doucet on Unsplash

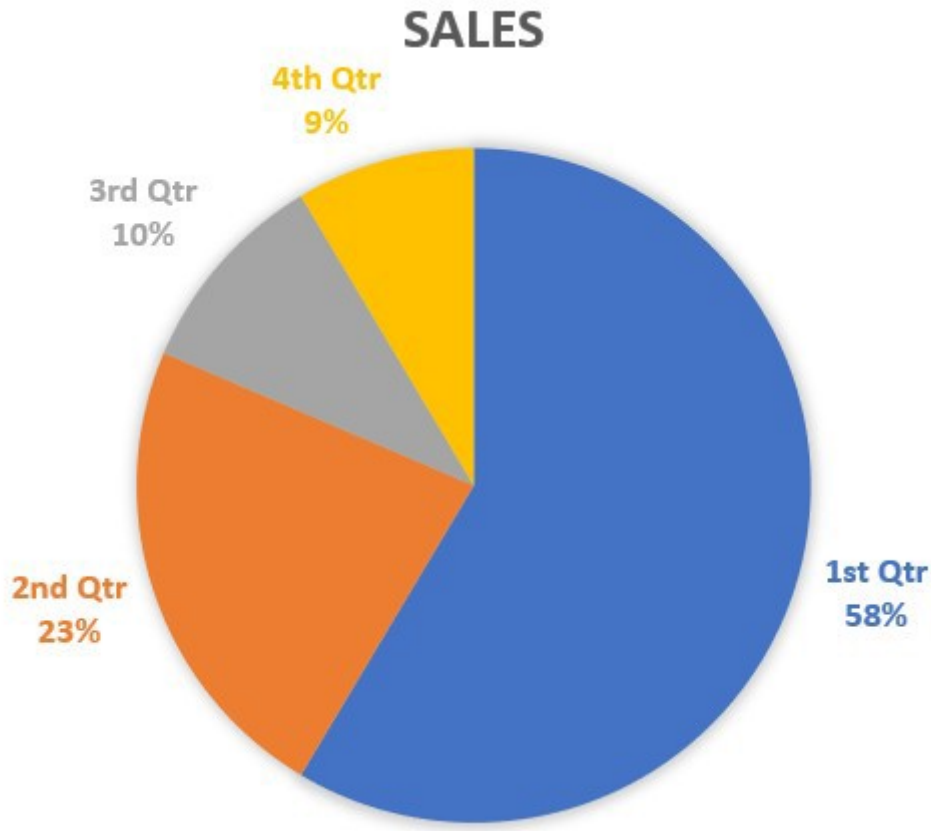
In the sections above, we have only covered the descriptive statistical analysis of the numerical data (housing price). How about the categorical data? In our mind, we might have some questions relevant to the categorical data in our housing dataset:

- What is the proportion of each type of house (h - house, t — townhouse and u — unit) in our dataset?
- Which region has the highest number of property sales?

5.1 Pie Chart

Pie Chart is a simple graphical way to show the numerical proportion of categorical data in a dataset. A Pie Chart is also known as Circular Chart (Source: Wikipedia) which is divided into wedge-shaped pieces. The arc length of each piece is proportional to the relative frequency of the categorical data.

Let us look at one example of a Pie Chart.

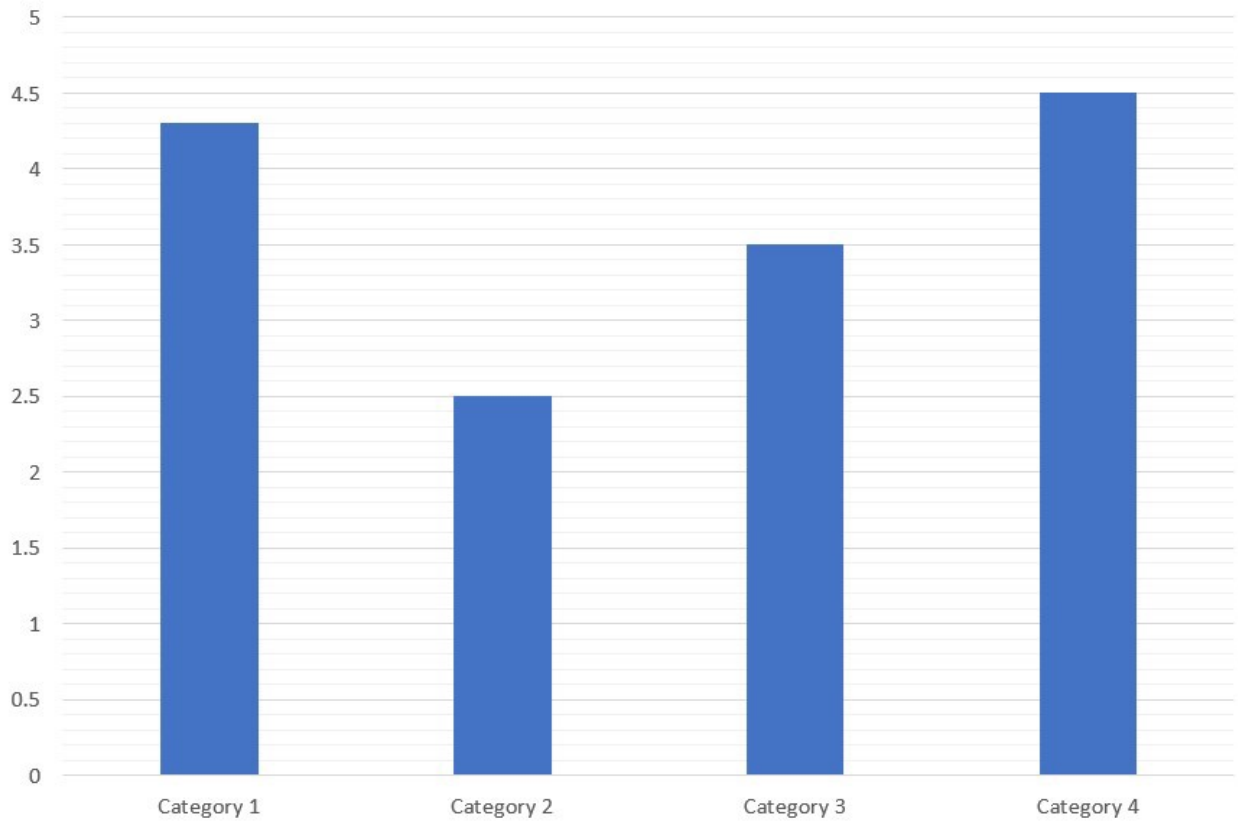


A sample Pie Chart

A glimpse of the sample Pie Chart above should instantly give us an idea of the sales performance in a year. Obviously, more than half of the sales are achieved in the 1st quarter while the 4th quarter hits the lowest sales.

5.2 Bar Chart

Bar Chart is another effective graphical display for categorical data. **A bar chart is a graphical display to show the number of occurrences or frequency for each categorical data using bars.**



Sample bar chart

5.3 Python code in Practice

Pandas offers a plot feature to generate a Pie Chart to show the proportion of each type of house in our dataset.

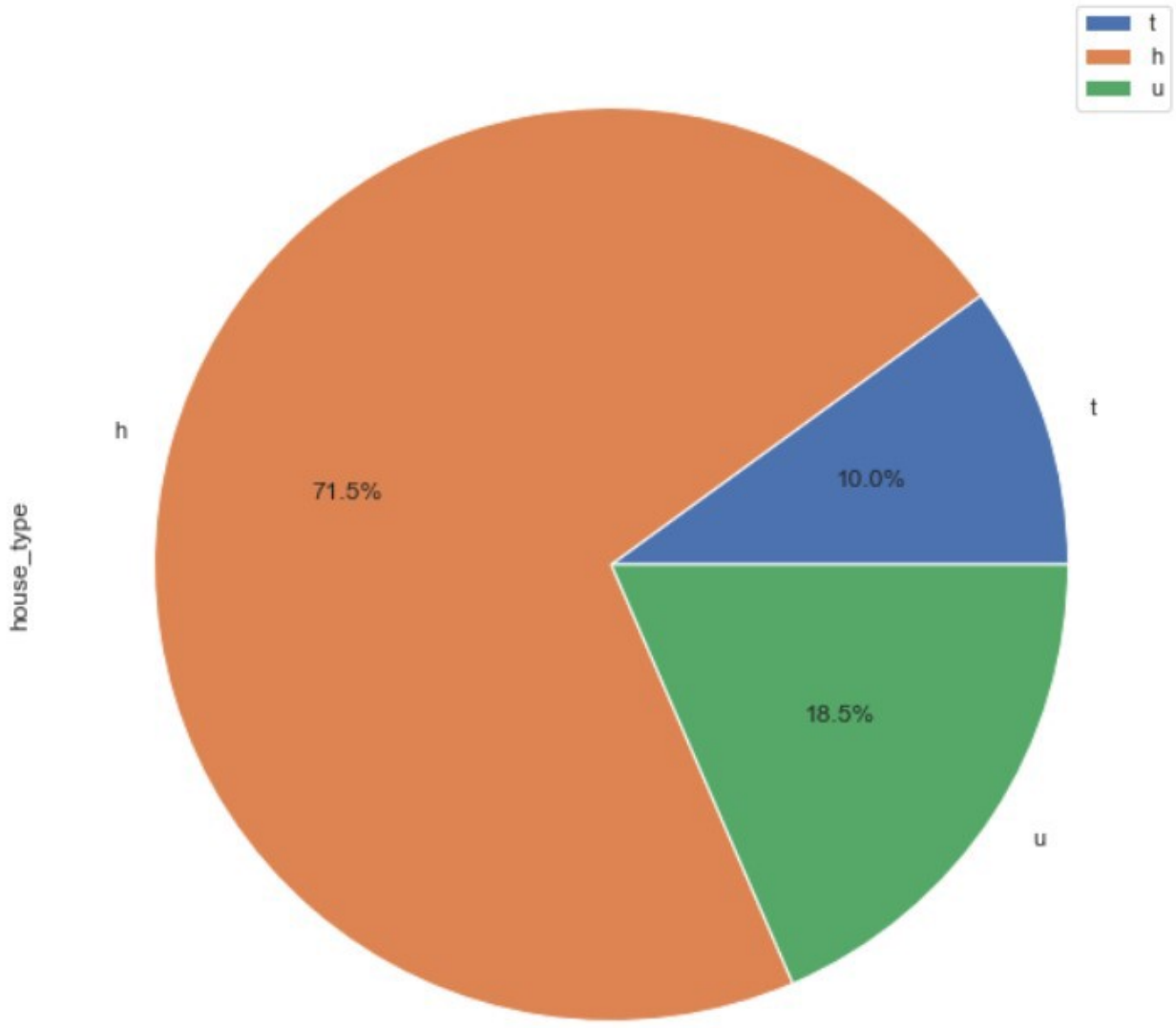
```
type_counts = df['Type'].value_counts()
df2 = pd.DataFrame({'house_type': type_counts},
                   index = ['t', 'h', 'u']
                  )
df2.plot.pie(y='house_type', figsize=(10,10), autopct='%1.1f%%')
```

create_pie.py hosted with ♡ by GitHub view raw

Create a Pie Chart to show the proportion of house

- **Line 1:** Use Pandas method "*value_counts()*" to get the frequency for each house type.
- **Line 2–4:** Create a new dataframe, *df2*. This new dataframe has only a single column, "*house_type*" to hold the count number of each house type, "*type_counts*"

number of each house type. The parameter "*autopct*" is enabled to show the percentage of each house type in the Pie Chart.



Pie Chart to show the proportion of each house type

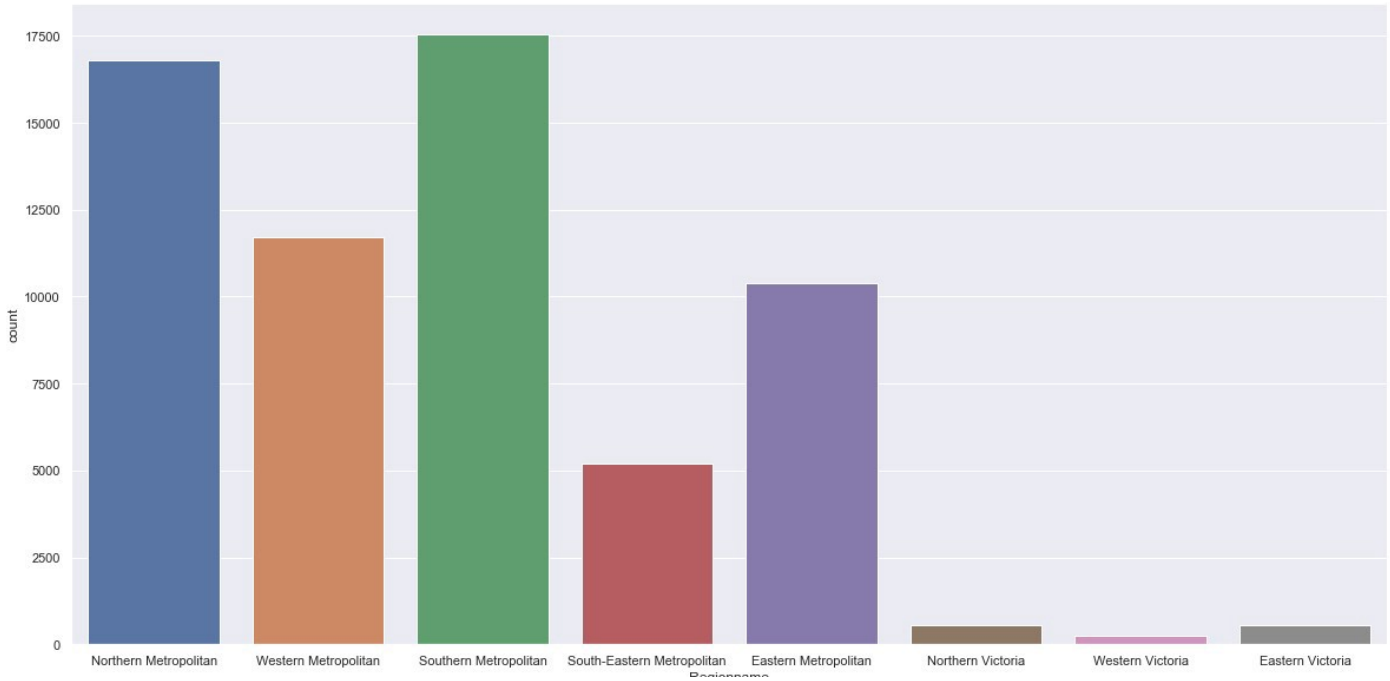
From the Pie Chart, we can easily identify the percentage breakdown for each type of house.

Next, we can use Seaborn to create a bar chart to show the number of property sales based on regions.

```
sns.set(style='darkgrid')
plt.figure(figsize=(20,10))
ax = sns.countplot(x='Regionname', data=df)
```

create_bar_chart.py hosted with ♡ by GitHub view raw

- **Line 1–2:** Set the Seaborn theme to "darkgrid" and set the figure size of plot
- **Line 3:** Assign the categorical variable "*Regionname*" to the parameter "*x*". Assign the plotting dataset, "*df*", to the parameter "data". The Seaborn "*countplot()*" method will automatically capture the number of occurrences of every region and render a bar chart.



Bar Chart to show the number of property sales in each region

From the Bar Chart above, we can conclude that the Southern Metropolitan has the highest number of property sales.

. . .

Conclusions

This article provides a quick guide on the descriptive statistics by introducing a few main descriptive measures and graphical displays which are widely adopted by the community. The main aim is just to show some possible ways to summarize our dataset in a meaningful way by using Pandas and Seaborn. Pandas and Seaborn should be two of the essentials toolkits for those who wish to involve in statistical analysis or data science projects. One major advantage of Pandas and Seaborn is that they have already encapsulated lots of complicated calculations and plotting steps into few lines of

However, the topics covered here are not exhaustive and there are many more topics which are not discussed here. Besides, the existing topics presented in this article can also be broken down and discussed with further details (which may require a few more articles dedicated to each specific topic).

I wish this article can offer a kickstart and a general recap to the descriptive statistics. I hope you enjoy reading.

. . .

Github Resources:

All the Python scripts presented here are written in a Jupyter Notebook and shared through a Github Repo. Feel free to download the notebook from https://github.com/teobeeguan2016/Descriptive_Statistic_Basic.git

References

1. Definition of Percentile (Retrieved from https://en.wikipedia.org/wiki/Percentile)
2. Definition of Pie Chart (Retrieved from https://en.wikipedia.org/wiki/Pie_chart)

Python Statistics Programming Visual Analytics