

Study Session 7

Introduction to Data Visualization Outline

- Data visualization
- Before you visualize your data
- Data visualization techniques
- Data visualization with seaborn
- Plotting continuous data with seaborn
- Plotting categorical data with seaborn

Study Session Duration

This Study Session requires a minimum of 3 hours' formal study time. You may spend an additional 2-3 hours on revision.

INTRODUCTION TO DATA VISUALIZATION

matplotlib

 seaborn

Preamble

Data visualization refers to the graphical representation of data by visual elements such as charts, infographics, and maps to understand the data.

There is a story behind every data, and data visualization brings them to life. It enables decision-makers, stakeholders or your readers to see data analytics presented visually, so they can grasp difficult concepts or identify new patterns.

Learning Outcomes of Study Unit 7

Upon completion of this study unit, you should be able to:

- 7.1 Work with the concept of storytelling through data visu...
- 7.2 Visualize data with and Seaborn and Matplotlib package



Terminologies, Acronyms and their Meaning

AI	Artificial Intelligence
ML	Machine Learning
RL	Reinforcement Learning
DL	Deep learning
EDA	Exploratory Data Analysis
Np	Numpy
Plt	Matplotlib

NaN	Not a Number
NULL	Missing value
Viz	Visualization
TF	TensorFlow
os	Operating system
pd	Pandas
sns	Seaborn

7.0 Introduction

According to English Language adage, a picture is worth a thousand words. This means that complex and sometimes multiple ideas can be conveyed by a single still image, which conveys its meaning rather than a mere data analysis inform of ordinary tables or verbal descriptions.

In the other hand, data visualization provides an accessible way to see and understand trends (upward or downward direction), outliers (extreme values), and patterns in data. Our eyes are easily drawn to colours and pattern. Data visualization helps grab the readers' interest and keeps their eyes on the message of the data.



The importance of Data Visualization is as follows:

- ✚ It is a powerful way to explore data with presentable results
- ✚ It displays the summary of our data at a glance
- ✚ Charts and graphs make it easier to identify patterns and trends
- ✚ It helps to identify areas that need attention or improvement.

It can be hard for the audience to grasp the true meaning of the findings without data visualization.

7.1 Before you visualize your data

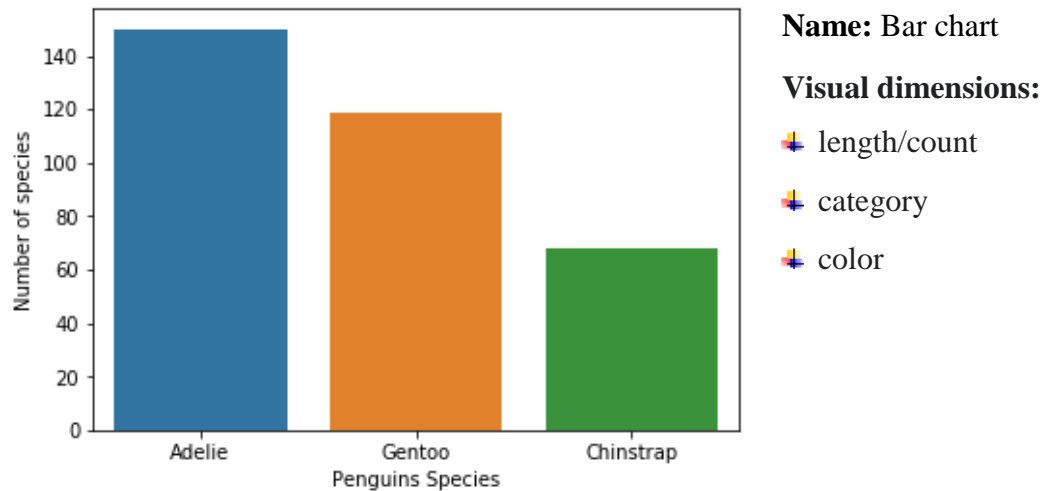
Presentation of data and information is not just about picking any data visualization design. Matching data to the right visualization begins by answering the following five (5) key questions:

1. What relationship am I trying to understand between my data sets?
2. Do I want to understand the distribution of data and look for outliers?
3. Am I looking to compare multiple values or looking to analyze a single value over time?
4. Am I interested in analyzing trends in my data sets?
5. Is this visualization an important part of my overarching data story?

With those questions (and your answers) in mind, we'll dive into different visualization techniques at which we can represent our data and data story to life.

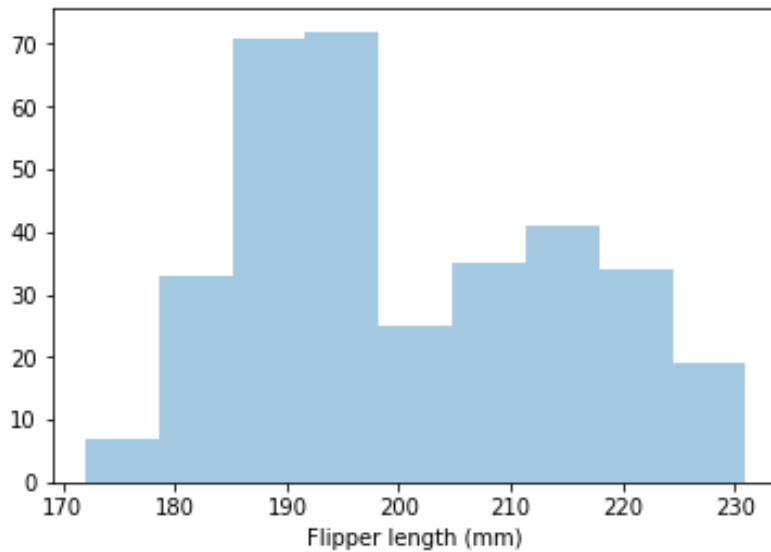
7.1.2 Data Visualization Techniques

In this section, you will learn about various data visualization techniques.



Description / Example usages

- ✚ Presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally.
- ✚ A bar graph shows comparisons among discrete categories. One axis of the chart shows the specific categories being compared, and the other axis represents a measured value.
- ✚ Some bar graphs present bars clustered in groups of more than one, showing the values of more than one measured variable. These clustered groups can be differentiated using colour.



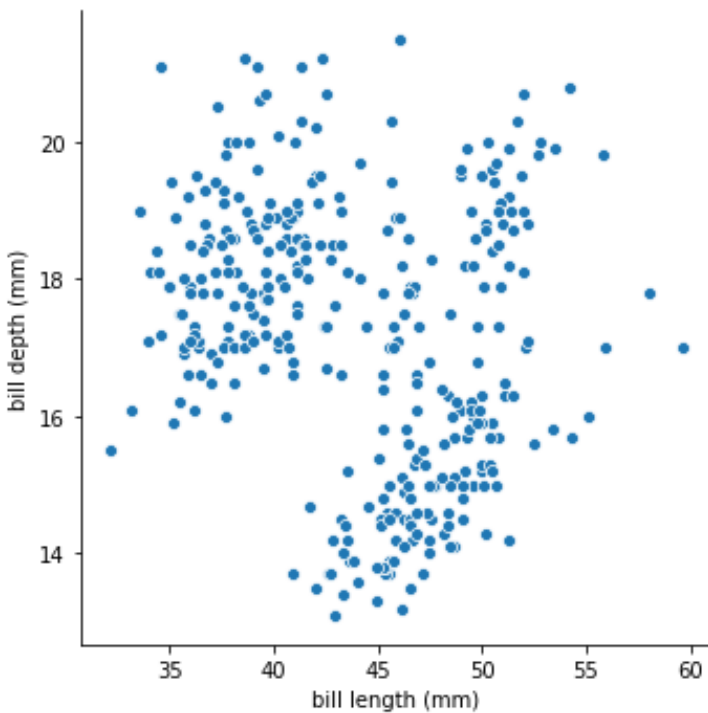
Name: Histogram

Visual dimensions:

- bin limits
- count/length
- color






Description / Example usages

- An approximate representation of the distribution of numerical data. Divide the entire range of values into a series of intervals and then count how many values fall into each interval this is called binning. The bins are usually specified as consecutive, non-overlapping intervals of a variable. The bins (intervals) must be adjacent, and are often (but not required to be) of equal size.
- The height of the bar represents the number of penguins that lies within flipper length (mm) respective bin (range).







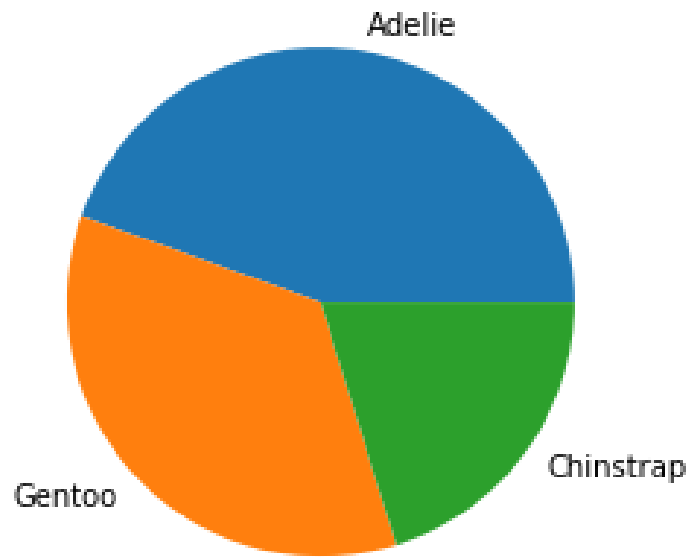
Name: Scatter plot

Visual dimensions:

-  x position
-  y position
-  shape
-  color
-  size

Description / Example usages

-  Uses Cartesian coordinates to display values for typically two variables for a set of data.
-  Points can be coded via color, shape and/or size to display additional variables.
-  Each point on the plot has an associated x and y term that determine its location on the cartesian plane.
-  Scatter plots are often used to highlight correlation between variables (x and y).



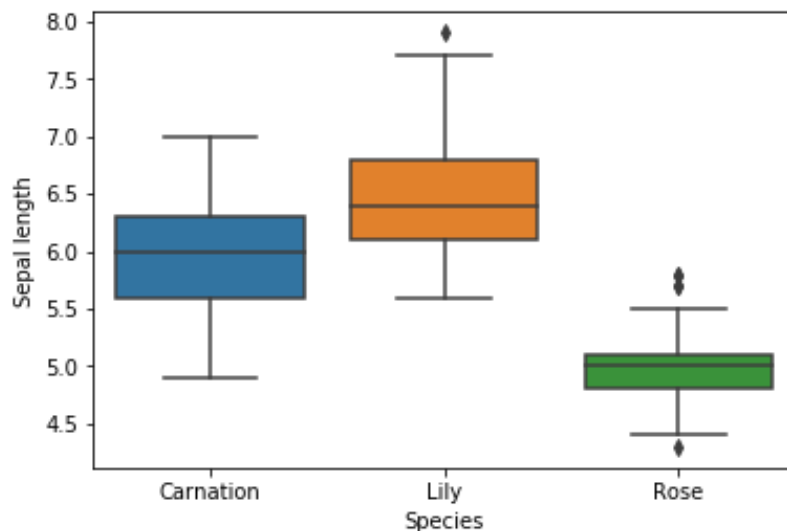
Name: Pie chart

Visual dimensions:

color

Description / Example usages

- Represents one categorial variable which is divided into slices to illustrate numerical proportion. In a pie chart, the arc length of each slice is proportional to the quantity it represents.
- For example, as shown in the graph above, the proportion of Penguins species.



Name: Box and Whisker Plot

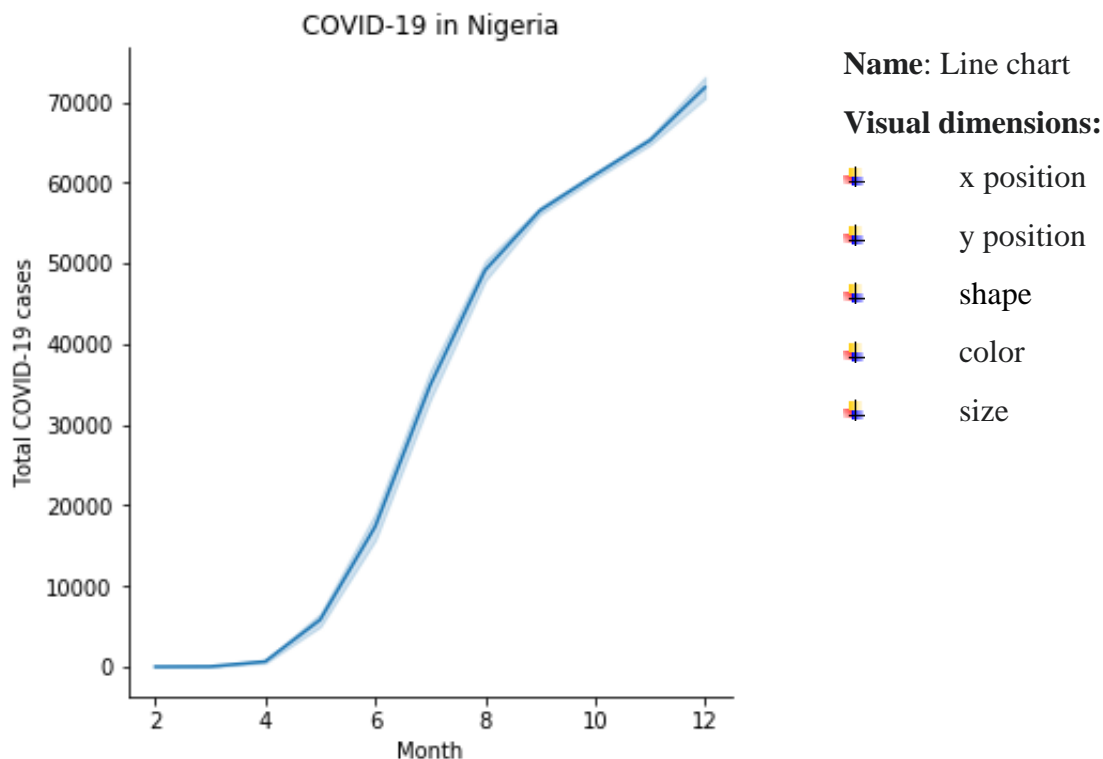
Visual dimensions:

x axis

y axis

Description / Example usages

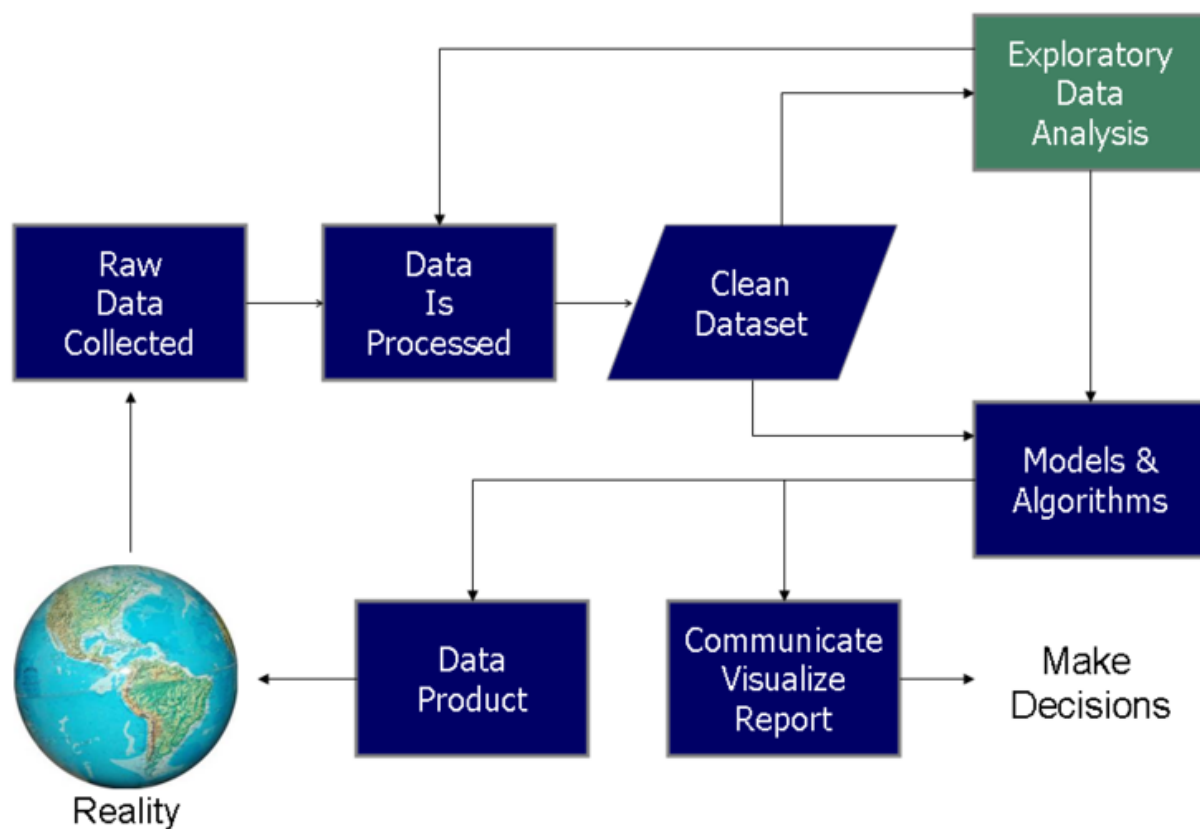
- ✚ A method for graphically depicting groups of numerical data through their quartiles.
- ✚ Box plots may also have lines extending from the boxes (*whiskers*) indicating variability outside the upper and lower quartiles.
- ✚ Outliers may be plotted as individual points.
- ✚ The two boxes graphed on top of each other represent the middle 50% of the data, with the line separating the two boxes identifying the median data value and the top and bottom edges of the boxes represent the 75th and 25th percentile data points respectively.
- ✚ Shows the distribution of quantitative data in a way that facilitates comparisons across levels of a categorical variable. For example, comparing the distribution of sepal length among flower species (e.g. Carnation, Lily, and Rose).



Description / Example usages

- ✚ Represents information as a series of data points called 'markers' connected by straight line segments.
- ✚ Similar to a scatter plot except that the measurement points are ordered (typically by their x-axis value) and joined with straight line segments.
- ✚ Often used to visualize a trend in data over intervals of time – a time series – thus the line is often drawn chronologically.

Summary



Source: Data Visualization Process, <https://en.wikipedia.org>





Additional resources

For more additional resources on data visualization, check the following resources:

- ✚ <https://www.klipfolio.com/resources/articles/what-is-data-visualization>
- ✚ <https://www.import.io/post/what-is-data-visualization/>

7.1.3 Data Visualization

There are many tools/packages for data visualization in Python programming. Some of them are as follows:

-  **Matplotlib** produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms; you can generate plots, histograms, bar charts, line chart, scatterplots, etc., with just a few lines of code
-  **Seaborn** is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
-  **Plotly** generates the most interactive graphs; allows saving them offline and create very rich web-based visualizations
-  **Pandas** also possesses its own data visualization functionalities based on Matplotlib.

In this course, we will consider using Seaborn and Pandas for data visualization. We will also consider using some functions in matplotlib package, since Seaborn and Pandas based their visualization on matplotlib.

7.1.4 Installation

The Anaconda distribution of Python comes with Matplotlib and Pandas pre-installed and no further installation steps are necessary. Seaborn is not directly included but can easily be installed with **conda install seaborn** or **pip install seaborn**. Open the Anaconda Prompt or your terminal and run:

conda install seaborn

or

pip install seaborn

Once seaborn library is installed, we are ready to explore the capabilities.

7.1.5 Import Packages

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Using the code above, we have imported both Seaborn and Pandas. We assign both of these aliases to make calling their methods easier. `numpy` is assigned the alias `np`, `pandas` is assigned the alias `pd`, `matplotlib.pyplot` is assigned the alias `plt`, and `seaborn` is assigned the alias `sns`.

For our data visualization in this section, we shall use Food servers' tips dataset.

7.1.6 Description of food servers' tips dataset

This dataset is from a restaurant in Nairobi, Kenya where many come to eat food and after eating based on a total bill they have paid some tips. The following are the variables in the tips dataset:

total_bill: Cost of the meal in Kenya Shilling

tip: Gratuity in Kenya Shilling

gender: Sex of person paying for the meal

smoker: Whether they smoke in the party or not

day: Day of the week for the party

time: Time of the day whether for lunch or dinner

size: Size of the party

7.1.7 Load our data in a Pandas DataFrame

```
tips_data = pd.read_csv("datasets/tips.csv")

tips_data.head()
```

	total_bill	tip	gender	smoker	day	time	size
0	2125.50	360.79	Male	No	Thur	Lunch	1
1	2727.18	259.42	Female	No	Sun	Dinner	5
2	1066.02	274.68	Female	Yes	Thur	Dinner	4
3	3493.45	337.90	Female	No	Sun	Dinner	1
4	3470.56	567.89	Male	Yes	Sun	Lunch	6

```
tips_data.tail()
```

	total_bill	tip	gender	smoker	day	time	size
739	3164.27	645.28	Male	No	Sat	Dinner	3
740	2962.62	218.00	Female	Yes	Sat	Dinner	2
741	2471.03	218.00	Male	Yes	Sat	Dinner	2
742	1942.38	190.75	Male	No	Sat	Dinner	2
743	2047.02	327.00	Female	No	Thur	Dinner	2

```
tips_data.columns
```

```
Index(['total_bill', 'tip', 'gender', 'smoker', 'day', 'time', 'size'], dtype='object')
```

```
tips_data.shape
```

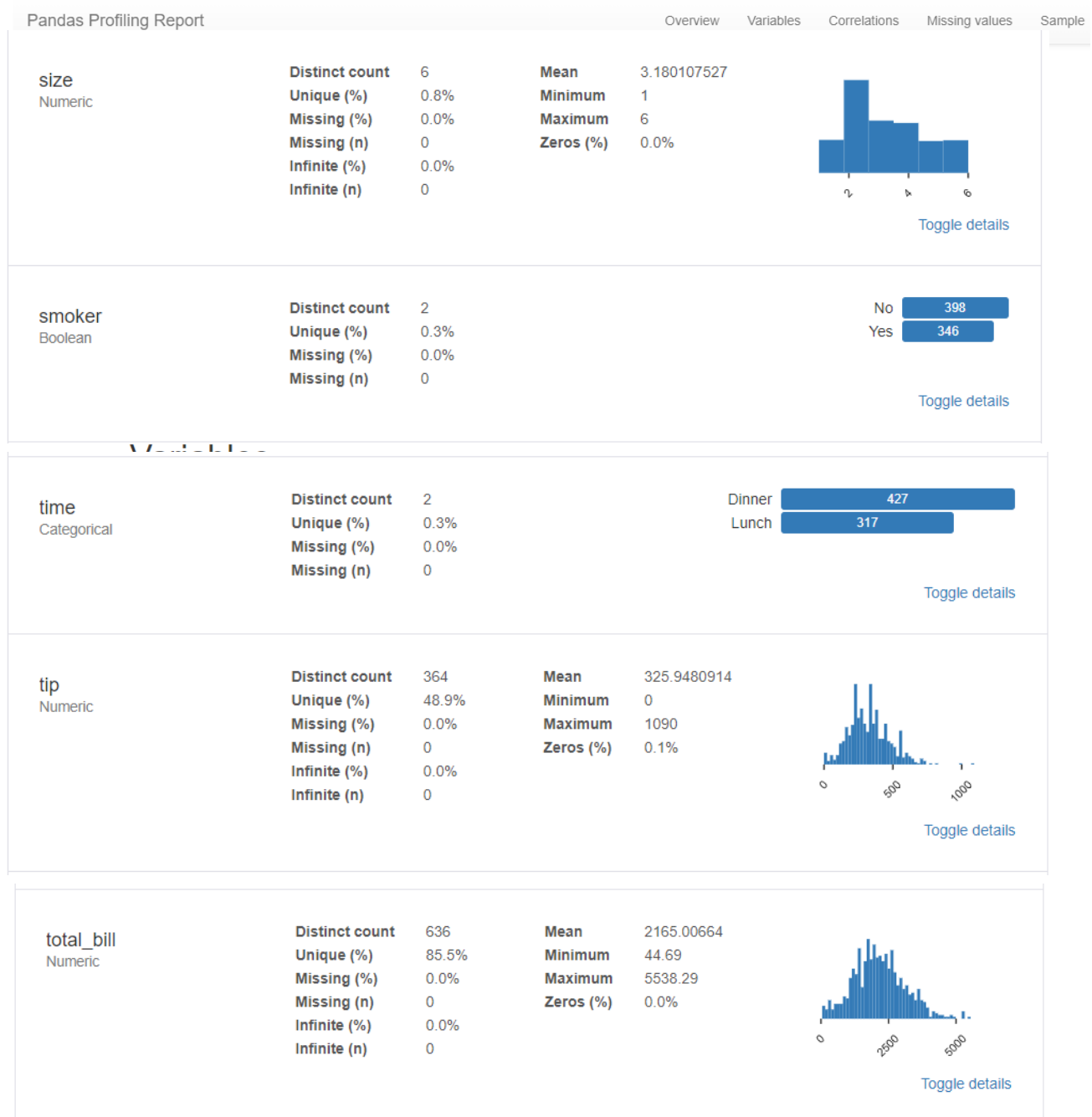
```
(744, 7)
```

7.1.8 Fast data exploratory using Pandas Profiling

In the last section, we talk about Pandas Profiling. Let's import and use pandas profiling before we do some data visualization in this section. Remember we shall use `.profile_report()` attribute on the DataFrame.

```
from pandas_profiling import ProfileReport
```

```
tips_data.profile_report()
```



7.1.9 Styles for Seaborn in Python

One of the benefits of Seaborn is that controlling aesthetics is much simpler than Matplotlib.

Seaborn has five built-in themes:

- Darkgrid
- Whitegrid
- Dark
- White
- Ticks

By default, Seaborn uses the darkgrid style.

Let's apply the whitegrid style:

```
sns.set_style("whitegrid")
```

This piece of code tells Seaborn to use the whitegrid style.

7.2 Plotting continuous data with Seaborn

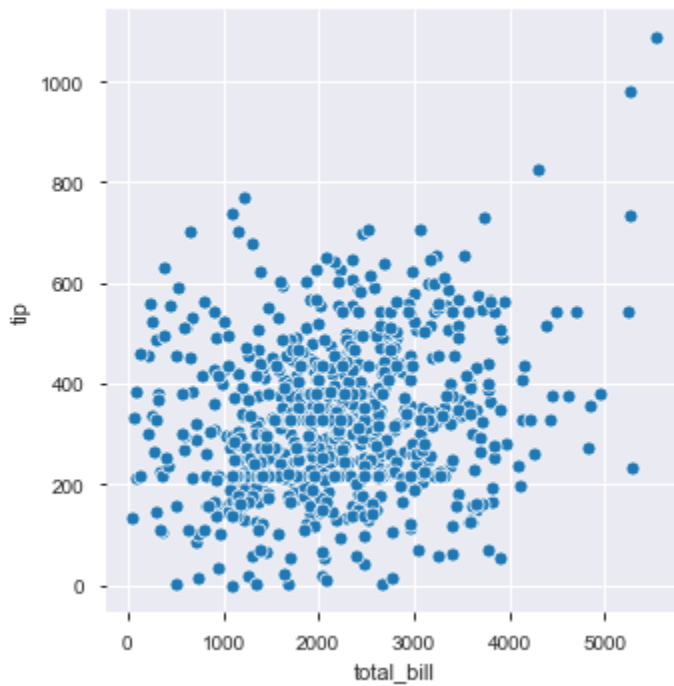
Scatterplot

Scatter plots are used to plot data points on the horizontal and vertical axis. It shows how much one variable is affected by another. It shows the extent of correlation. It is also used to find the relationship between two variables that are continuous.

To plot a scatter plot we use `sns.relplot()` function of seaborn library. It can be done by using:

```
sns.relplot(x='total_bill', y='tip', data = tips_data)
```

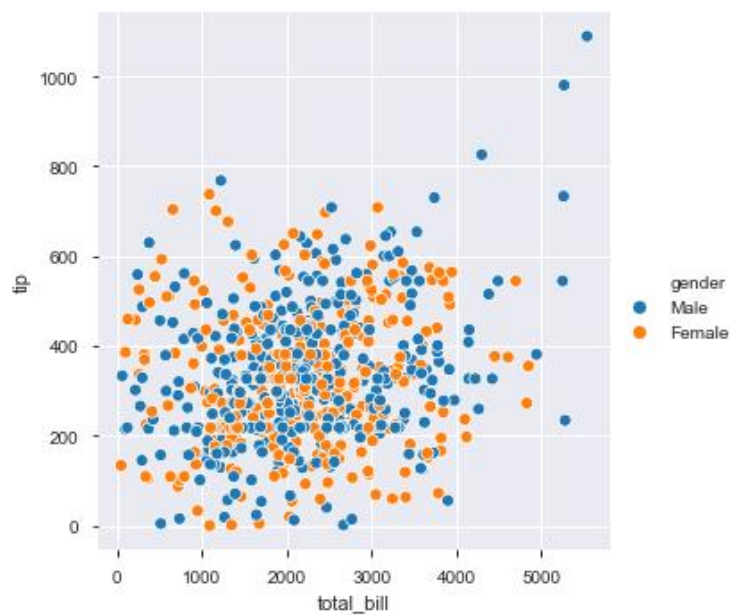
```
<seaborn.axisgrid.FacetGrid at 0x1bb976b21d0>
```



You can use any variable to classify scatter plot. For this, there is a parameter called **hue**. You can use hue as follows:

```
sns.relplot(x='total_bill', y='tip', hue='gender', data=tips_data)
```

```
<seaborn.axisgrid.FacetGrid at 0x1bb976fa278>
```



As you can see the scatter plot is classified based on sex by giving color to each point.

Important note

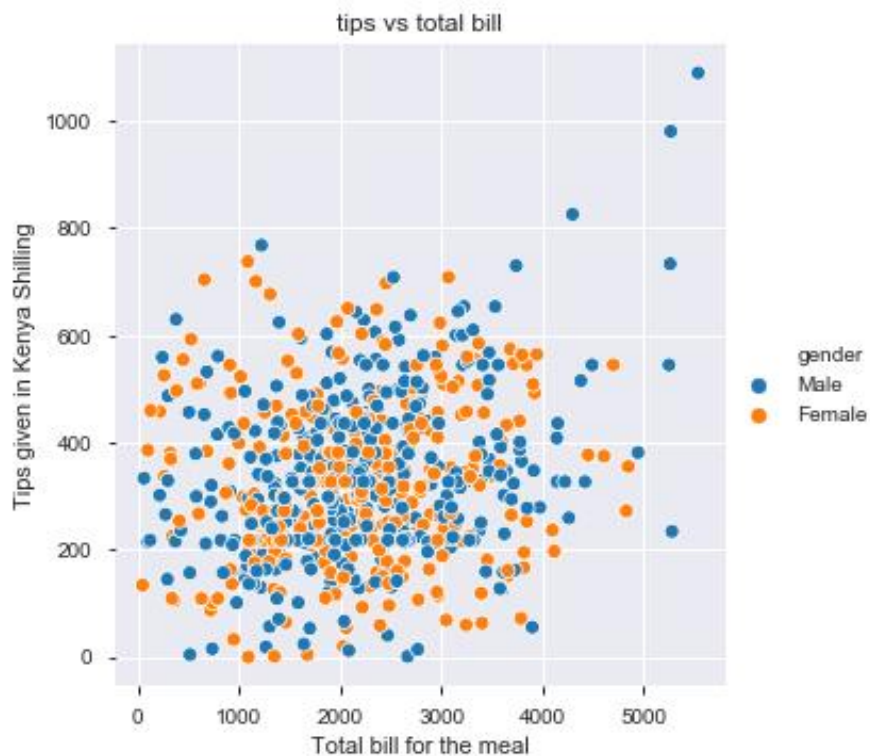
We can relabel our coordinates (x and y axis) and also give the plot a title by using matplotlib functions as follows:

- ✚ For x-axis use `plt.xlabel("label to change to")`
- ✚ For y-axis use `plt.ylabel("label to change to")`
- ✚ For the title use `plt.title("Title of the chart")`

For example:

```
sns.relplot(x='total_bill', y='tip', hue='gender', data=tips_data)
plt.xlabel("Total bill for the meal")
plt.ylabel("Tips given in Kenya Shilling")
plt.title("tips vs total bill")
```

```
Text(0.5, 1.0, 'tips vs total bill')
```

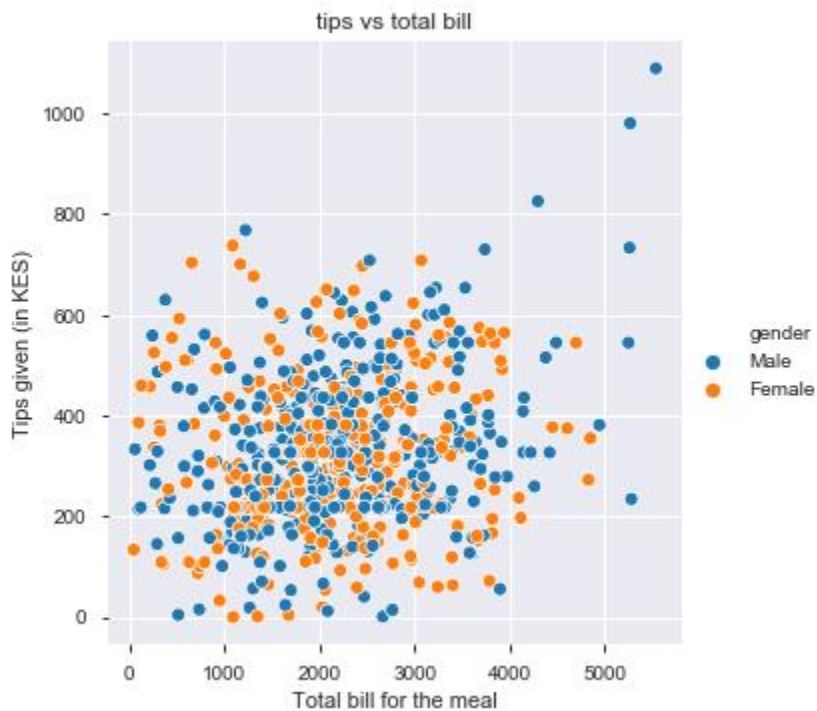


Important note

We always use semi colon (;) at the last line of code to stop the texts written before the visual is shown. For example:

```
sns.relplot(x='total_bill', y='tip', hue='gender', data=tips_data)

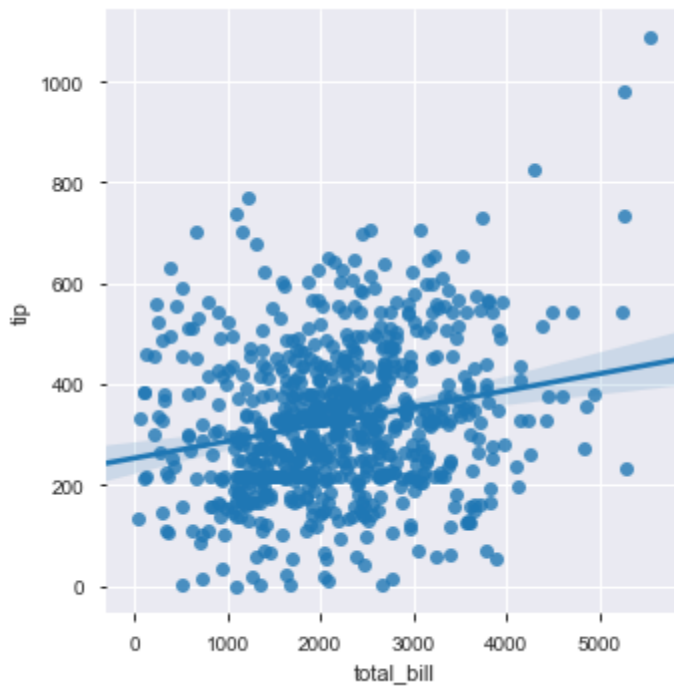
plt.xlabel("Total bill for the meal")
plt.ylabel("Tips given (in KES)")
plt.title("tips vs total bill");
```



Scatter and regression line

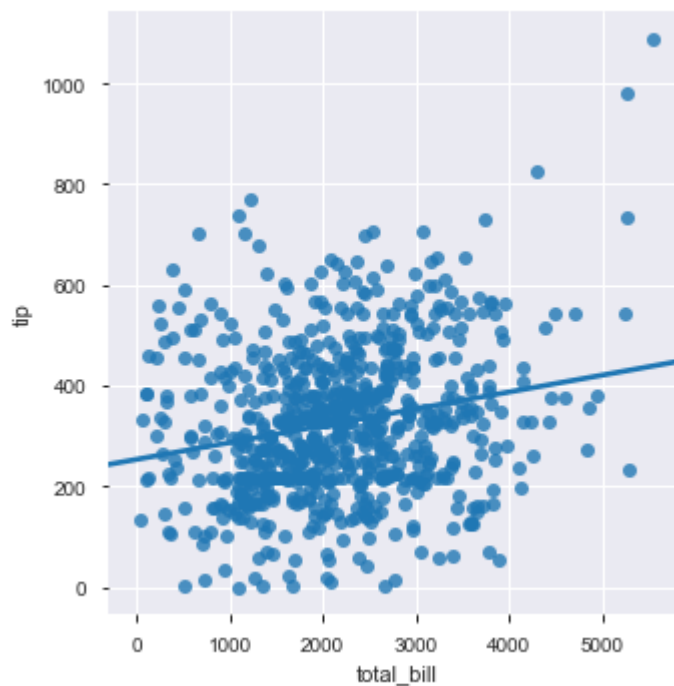
Seaborn makes this easy by using the `lmplot()` function. For example,

```
sns.lmplot(x='total_bill', y='tip', data=tips_data);
```



You can remove the confidence interval in the regression line by setting `ci = None` in `sns.lmplot()` function

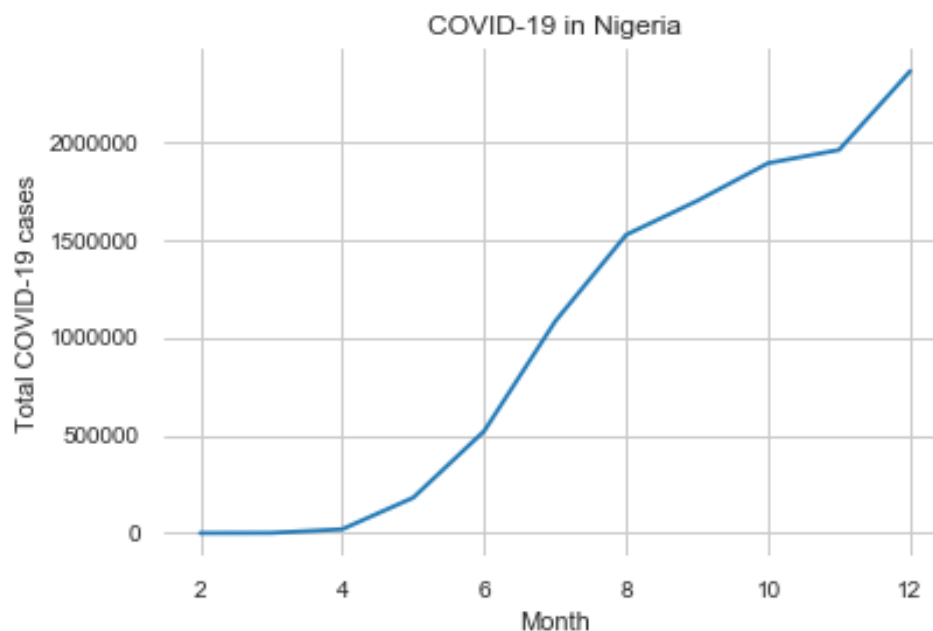
```
sns.lmplot(x= 'total_bill', y='tip', data= tips_data, ci = None);
```



Line chart

With some datasets, you may want to understand changes in one variable as a function of time, in this situation, a good choice is to draw a line plot. In seaborn, this can be accomplished by using `sns.lineplot()` function. For example, let's import Nigeria monthly COVID-19 cases dataset.

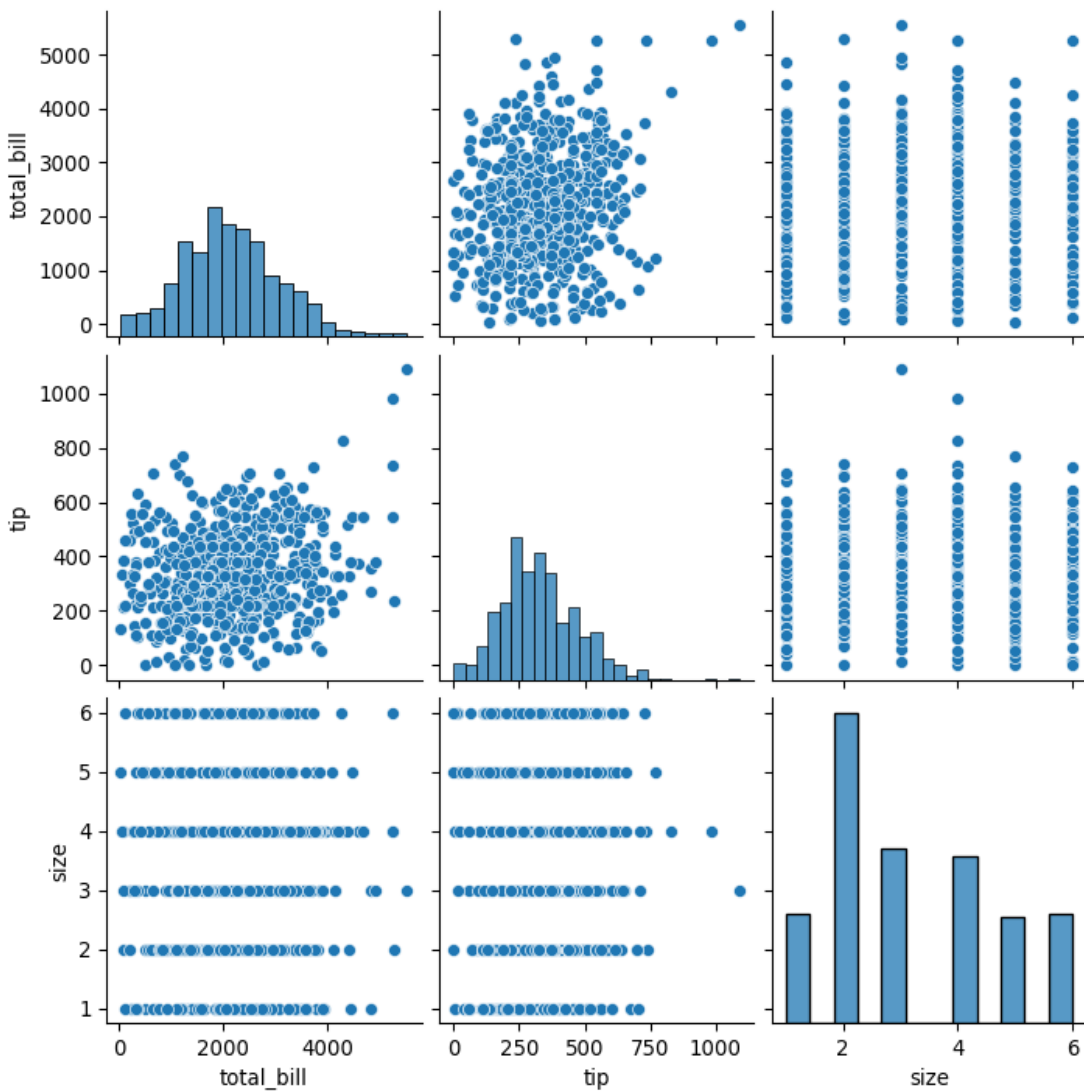
```
Nigeria_COVID19 = pd.read_csv("datasets/Nigeria monthly COVID-19 cases.csv")
sns.lineplot(data= Nigeria_COVID19, x = "month", y = "Total cases")
plt.xlabel("Month")
plt.ylabel("Total COVID-19 cases")
plt.title("COVID-19 in Nigeria");
```



Pair Plot

In the pair plot, one variable in the same data row is matched with the value of another variable. That is, it combines or do permutation and combination of all the variables in the dataset. This plot can only be a plot on numerical data. It can plot by using `sns.pairplot()` on the DataFrame:

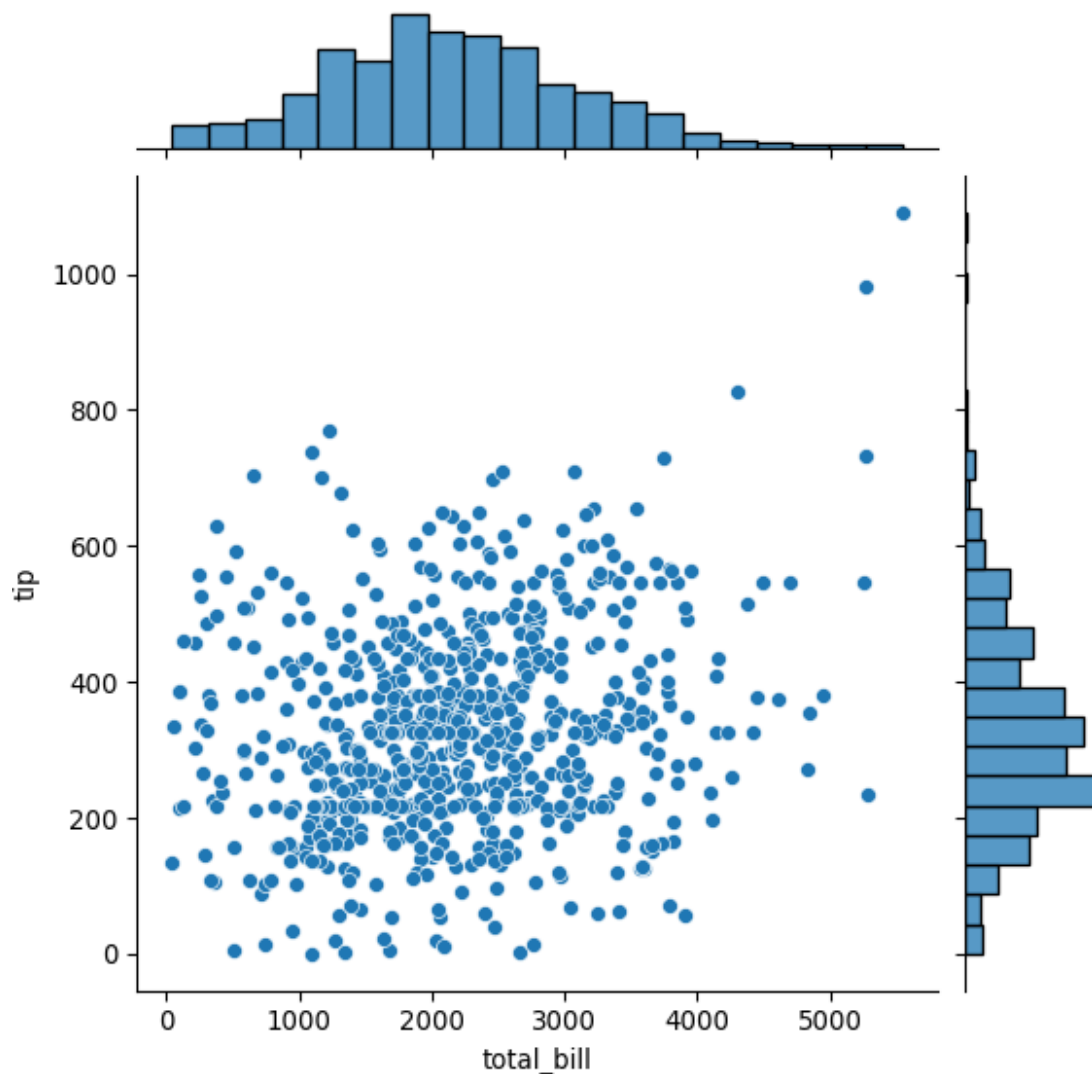
```
sns.pairplot(tips_data);
```



Join Plot

A join plot helps to learn about the relationship between 2 numeric variables. It is used to do univariate analysis. It displays a correlation between two variables. You can plot a join plot as

```
sns.jointplot(x='total_bill', y='tip', data=tips_data)
```



Correlation Matrices

We can plot correlation matrices by using a feature called **heatmap**. Heatmap helps us to find a correlation (interrelation) between every continuous variable in our dataset.

The basic requirement for finding correlation is that the variable should be numeric (continuous) i.e. data type must be **int** or **float**.

Correlation matrices cannot be found for categorical features because they are object (or string) data type. Whenever you will find correlation matrices the value will be ranging from -1 to +1 which is Pearson correlation. Therefore, to find the correlation you can use:

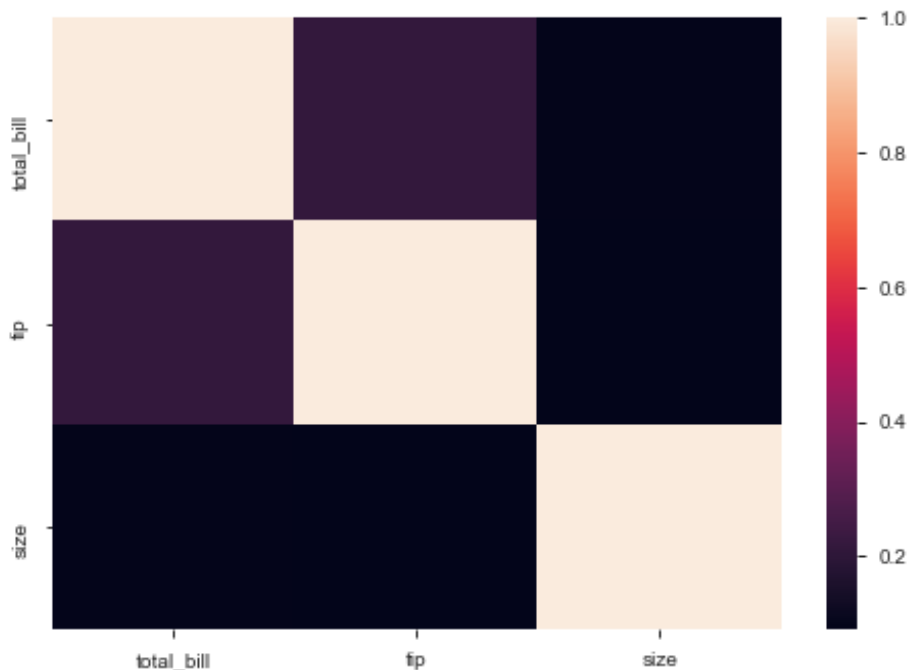
```
tips_data.corr()
```

	total_bill	tip	size
total_bill	1.000000	0.214756	0.096942
tip	0.214756	1.000000	0.090766
size	0.096942	0.090766	1.000000

As you can see, we are getting only 3 features because only these here are numerical and the rest are categorical.

To visualize it for getting better understanding you can use

```
sns.heatmap(tips_data.corr());
```

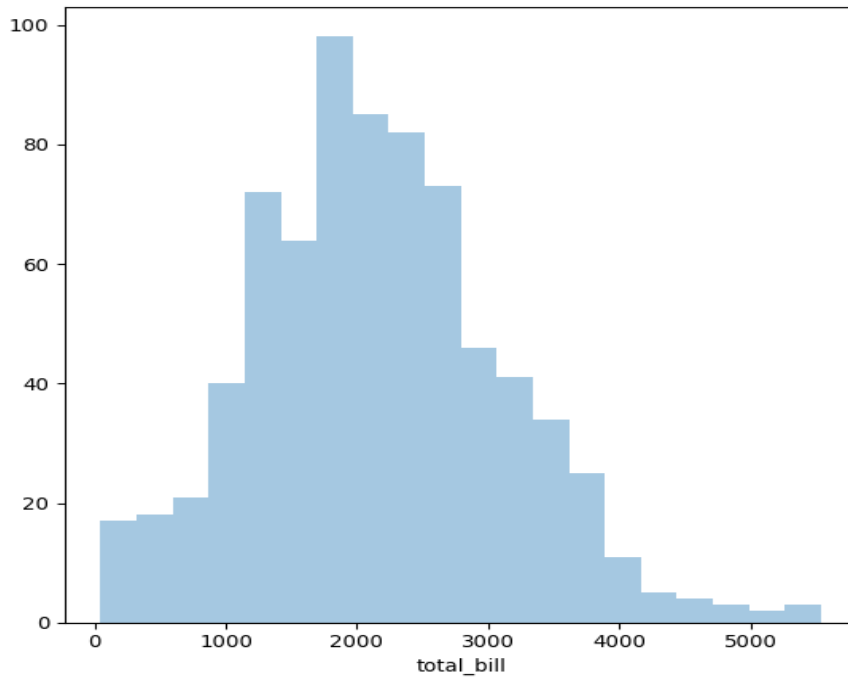


Histogram Plot

Hist Plot helps to create histograms. To do that, we will use a function called as `histplot()`. It creates a frequency distribution of continuous variables. It can be created by using `sns.histplot()` on the variable of choice:

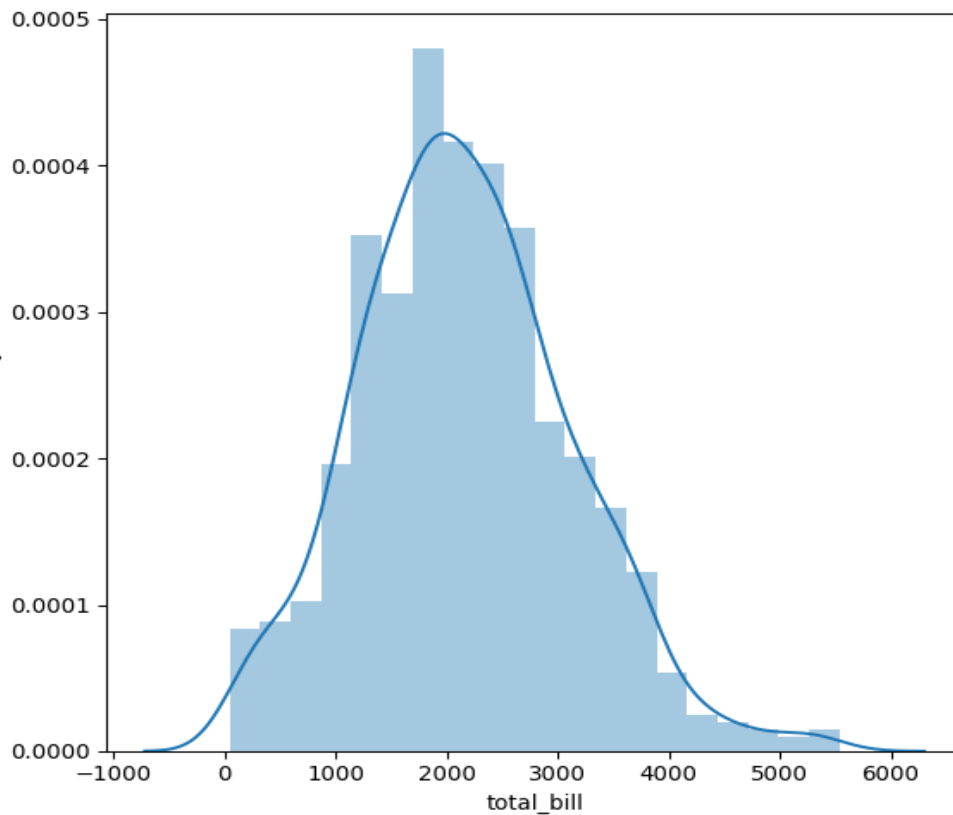
For histogram only use:

```
sns.histplot(x = tips_data['total_bill'], kde = False);
```



For both histogram and density plot use:

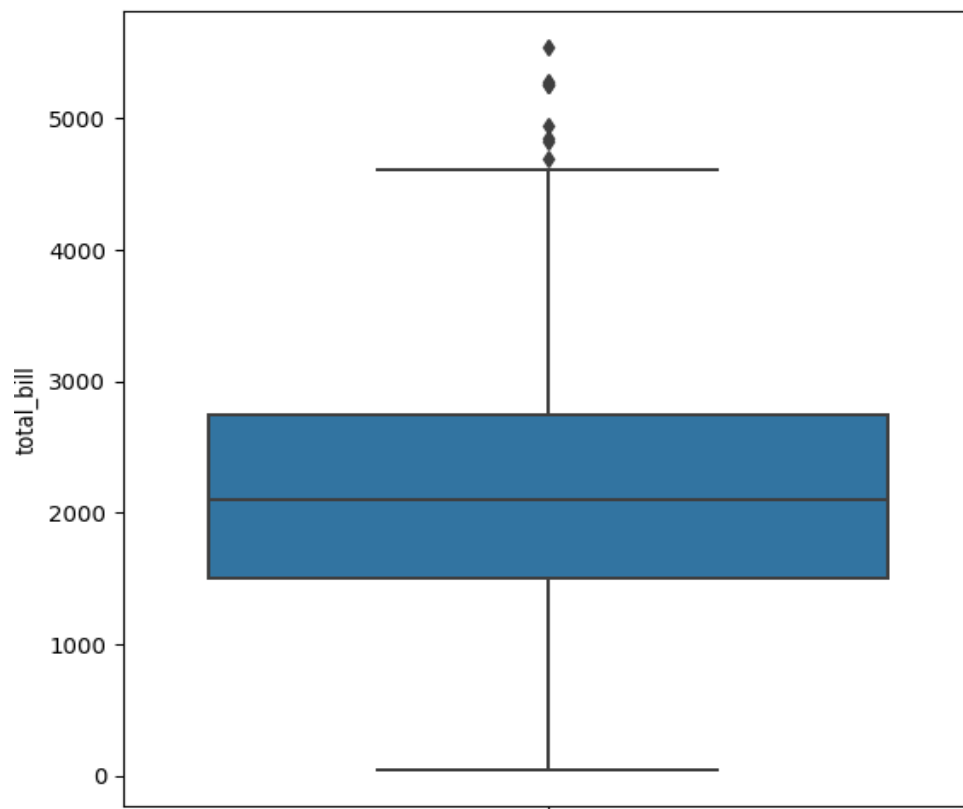
```
sns.histplot(x = tips_data['total_bill'], kde = True);
```



Box plot (box-and-whisker plot)

We can also show the distribution of a continuous variable using a box plot. The box plot shows the quartile values of the distribution. Each value in the box plot corresponds to actual observation in the data. It also shows outliers. You can plot boxplot as:

```
sns.boxplot(y = 'total_bill', data = tips_data);
```

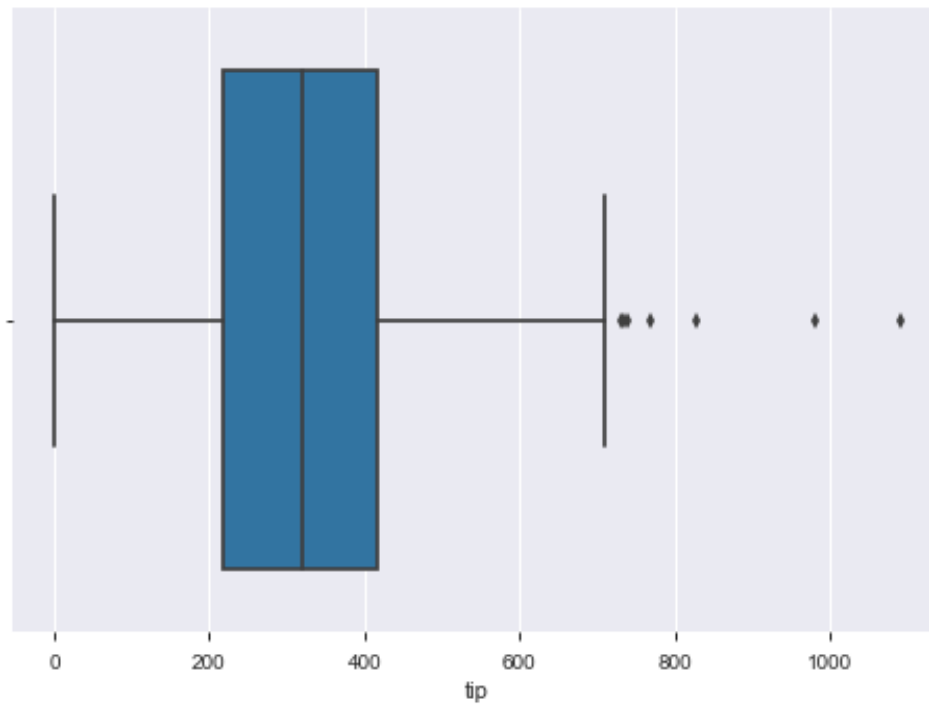



The distribution of total bill is symmetric i.e. it is not skewed. It looks like it is normally distributed. The point above the upper whisker shows some extreme values (total bill that is abnormal) known as outlier.

Important note

We can change orientation of any plot in seaborn either vertically or horizontally by changing the position of x and y axis. For example:

```
sns.boxplot(x = 'tip', data = tips_data);
```



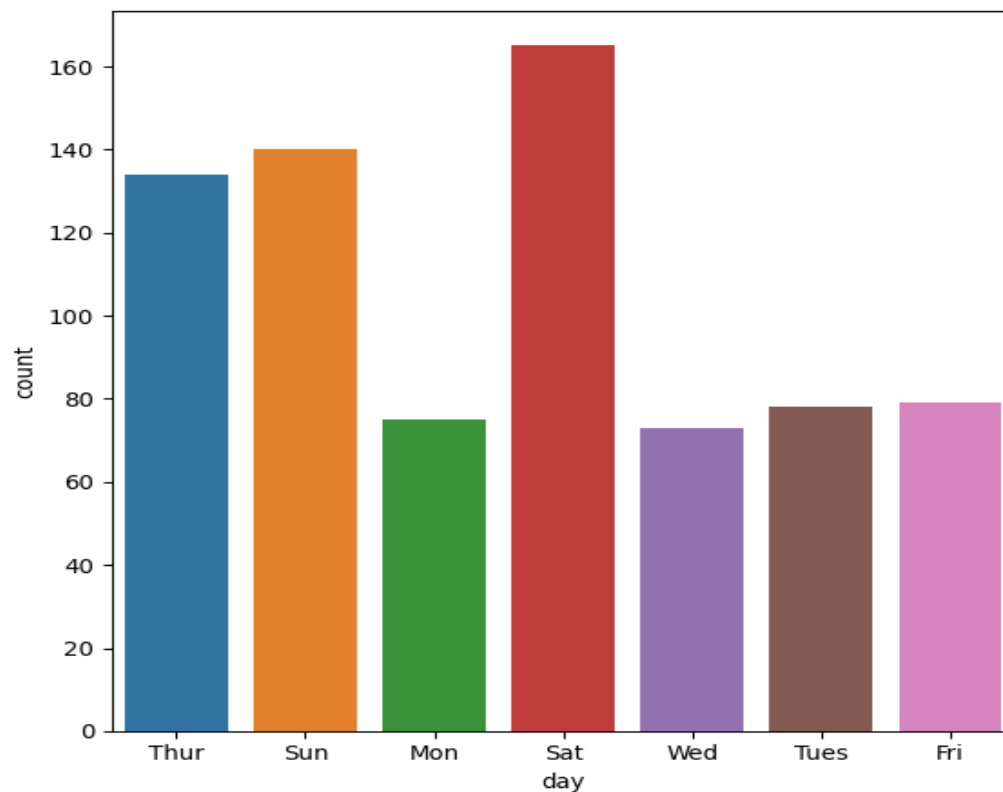
7.2.1 Plotting categorical data with Seaborn

So, far we have visualized data which is specifically numerical i.e. int or float datatype. Now we will try to visualize categorical data type. In our dataset, there are 4 categorical features and they include day, gender, time, smoker.

Count Plot

It shows the counts of observations in each category using bars. This is different from histogram because it has a gap (space) after every bar and the number of count (length of the bar) is proportional to the number of categories in that variable.

```
sns.countplot(x = "day", data = tips_data);
```



As you can see it plots the number of bars as there are categories in the variable. We can use `.value_counts()` to confirm that.

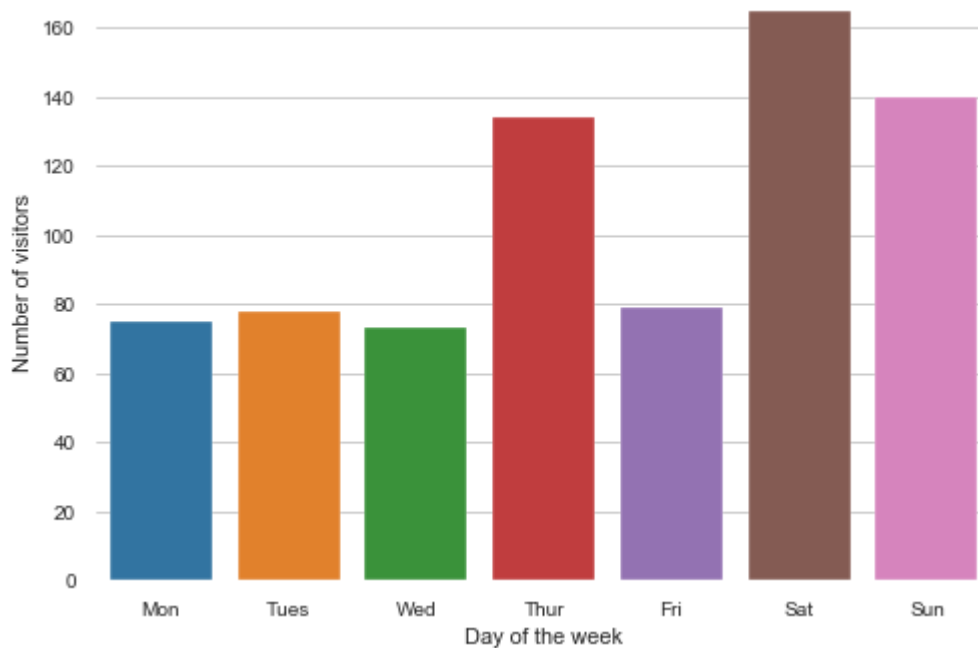
```
tips_data["day"].value_counts()
```

```
Sat    165
Sun    140
Thur    134
Fri     79
Tues    78
Mon     75
Wed     73
Name: day, dtype: int64
```

Important note

We can order the bar in the right order of the week i.e. from Monday to Sunday using `order` parameter

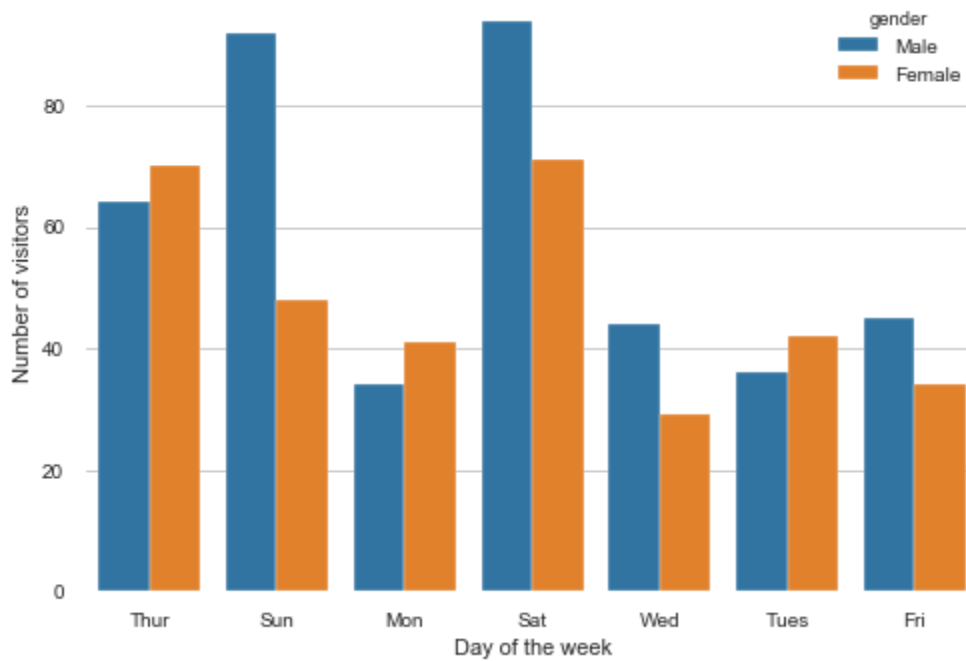
```
sns.countplot(x = "day", order = ["Mon", "Tues", "Wed", "Thur", "Fri", "Sat", "Sun"], data = tips_data)
plt.xlabel("Day of the week")
plt.ylabel("Number of visitors");
```



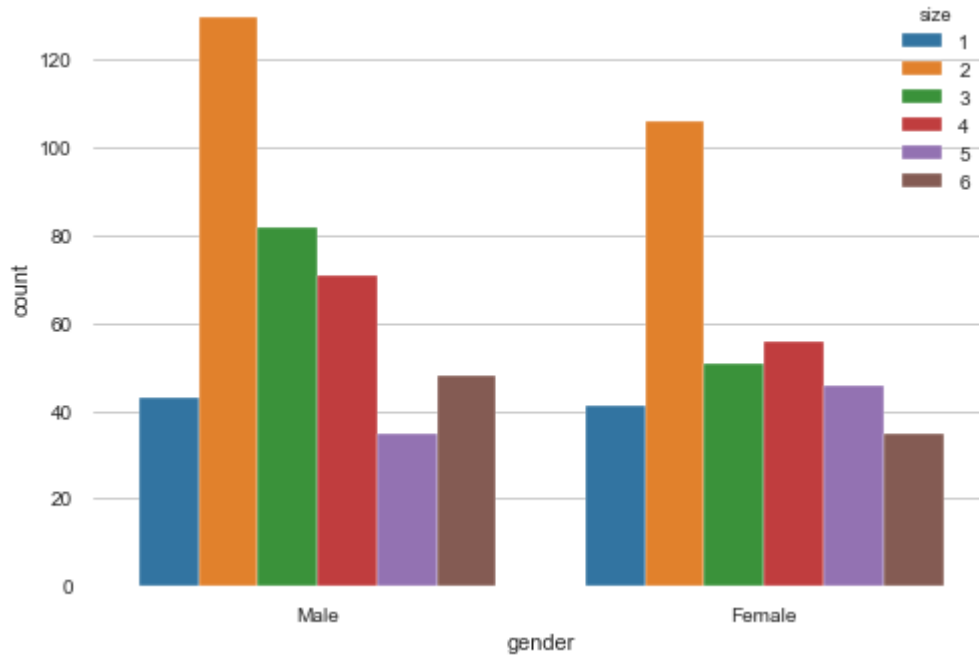
As you can see, a lot of visitors came to the restaurant on Saturday when compare to other days of the week

We can also compare day of the week and gender using the **hue** parameter as follows:

```
sns.countplot(x = "day", hue="gender", data = tips_data)
plt.xlabel("Day of the week")
plt.ylabel("Number of visitors");
```



```
sns.countplot(x='gender', data=tips_data, hue='size');
```

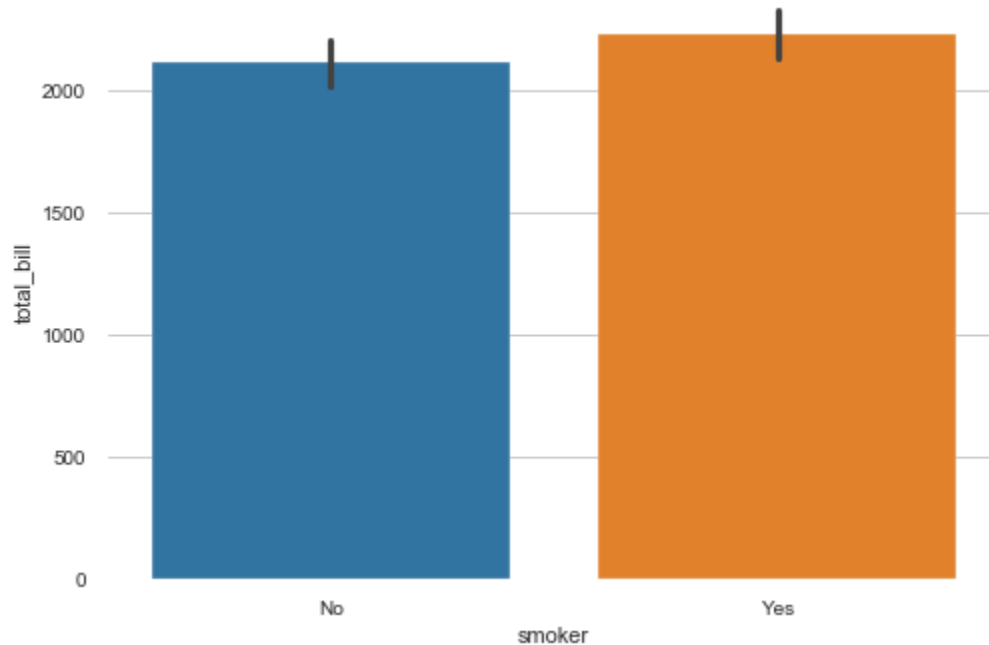


For both male and female, the most common size of the party in the restaurant is 2.

Bar Plot

Bar plot does the same work as count plot. But in this, we have to specify both x and y. Based on one feature it will display other value.

```
sns.barplot(x = 'smoker', y = 'total_bill', data = tips_data);
```

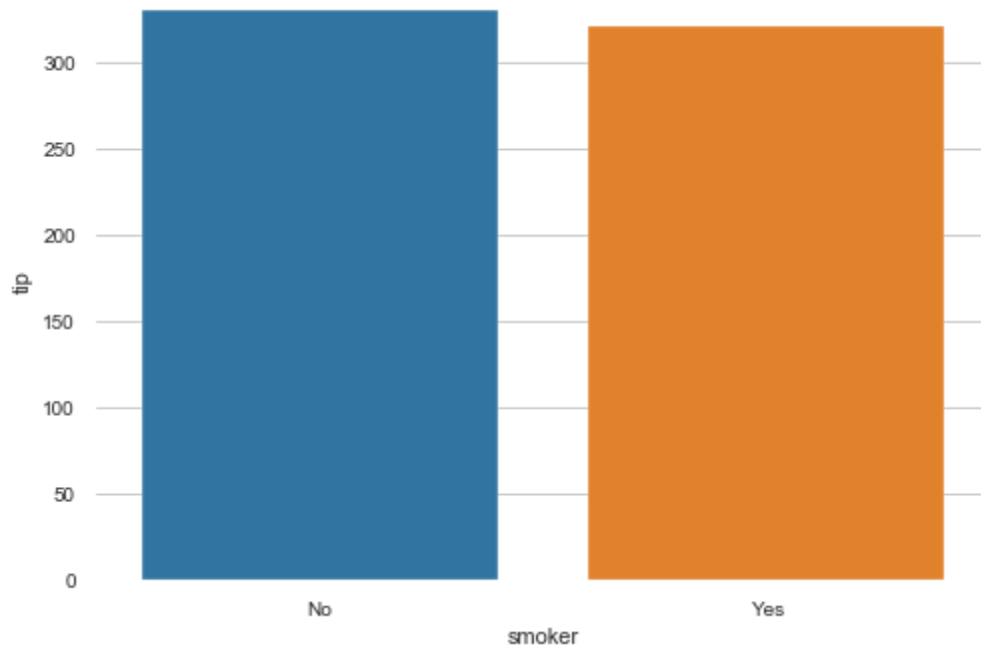


Those that smoke paid a slightly higher bill than those that did not smoke during their stay in the restaurant.

Important note

You can remove the error bar by setting `ci = None` in the `sns.barplot()`.

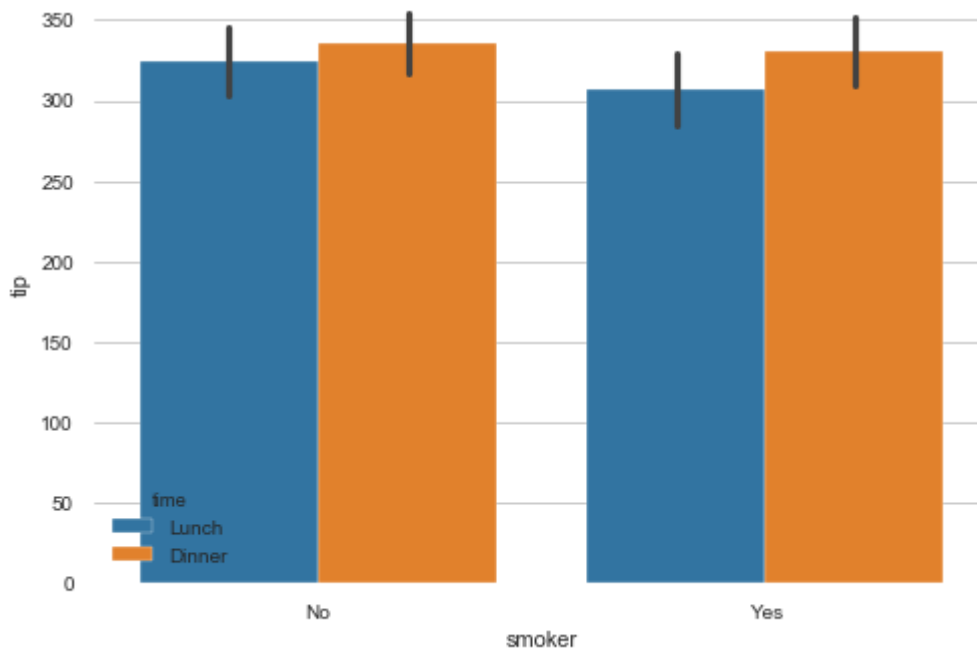
```
sns.barplot(x='smoker', y='tip', ci = None, data=tips_data);
```



Those that did not smoke give more tips than those that smoke

We can also compare smoker and time of the day using the `hue` parameter as follows:

```
sns.barplot(x = "smoker", y = "tip", hue = "time", data=tips_data);
```

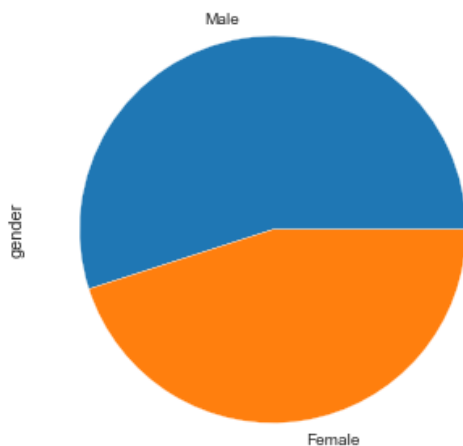


It seems that people give more tips at the dinner than at the lunch time irrespective of whether they smoke or not at the restaurant.

Pie Chart

A pie chart is a circular statistical graphic, which is divided into slices to illustrate numerical proportion. Seaborn is not supporting pie chart currently. We will use `.plot()` attribute of Pandas to do this.

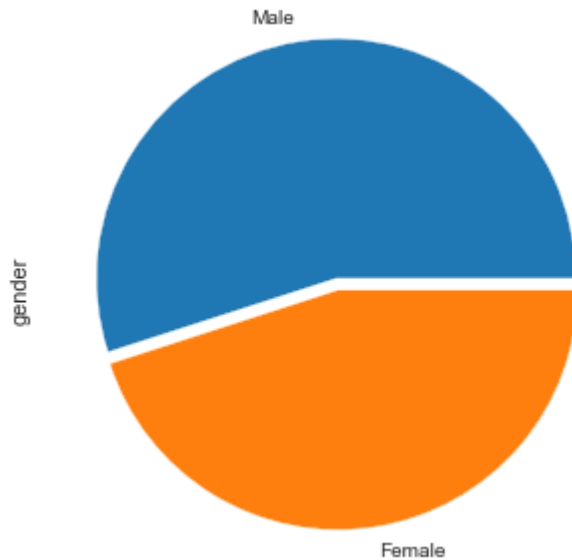
```
tips_data["gender"].value_counts().plot(kind = "pie");
```



To highlight a particular value in the plot, use `explode` parameter

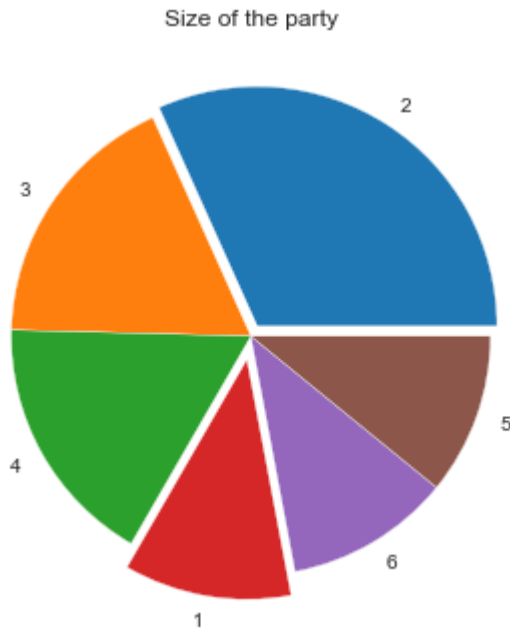
Explode 1st slice

```
tips_data["gender"].value_counts().plot(kind = "pie", explode = (0.05, 0));
```



To highlight the first and fourth value in the size of the party, use `explode` parameter and then put a non-zero value to those positions

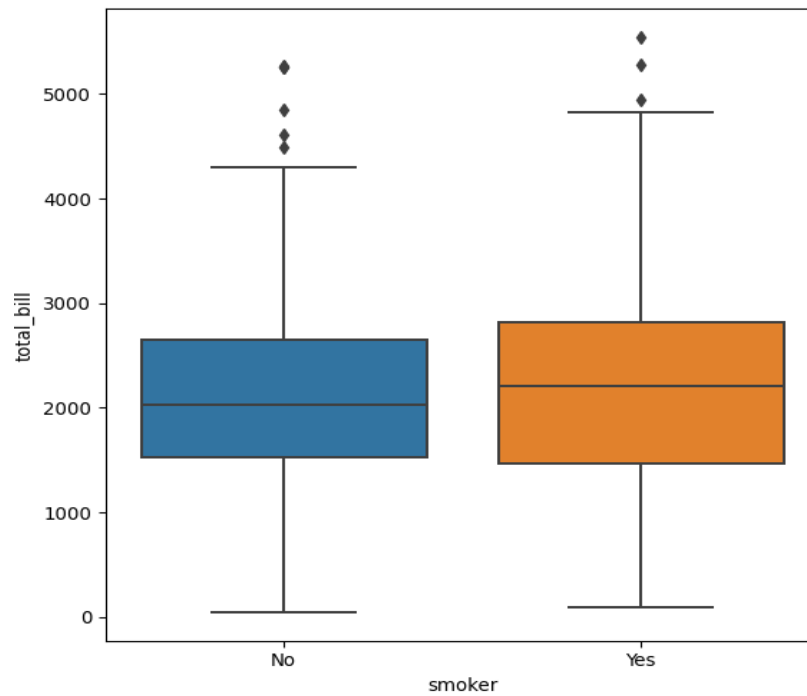
```
tips_data["size"].value_counts().plot(kind = "pie", explode = (0.05, 0, 0, 0.1, 0, 0))
plt.title("Size of the party")
plt.ylabel("");
```



Box Plot

A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable.

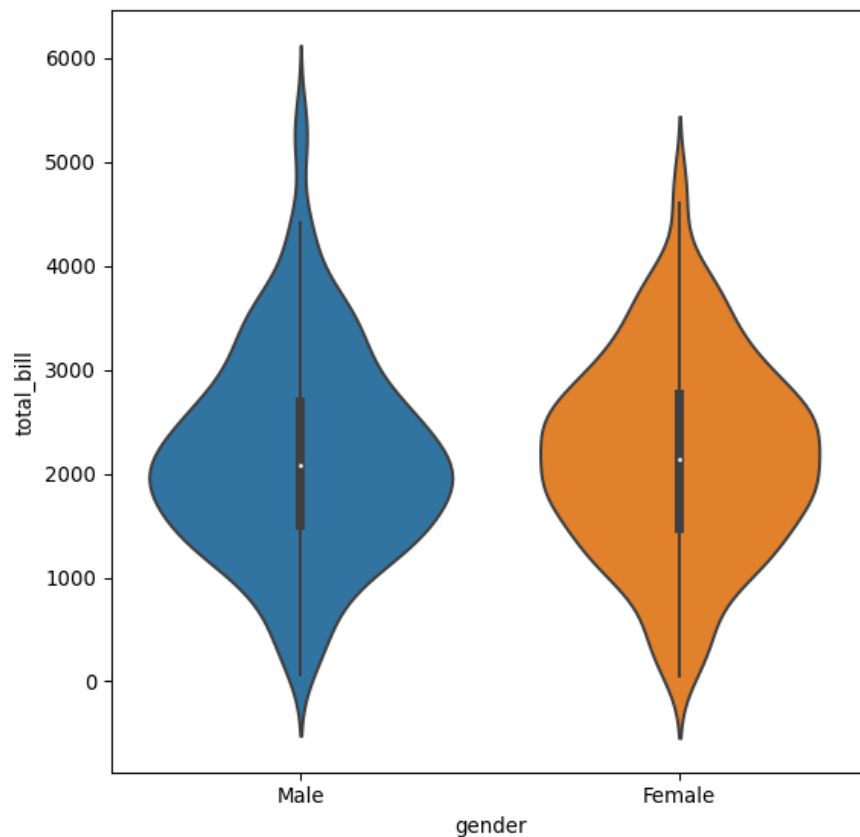
```
sns.boxplot(x='smoker', y='total_bill', data=tips_data);
```



Violin Plot

Violin plots take boxplots one step further by showing the kernel density distribution within each category. You can plot violin plot as:

```
sns.violinplot(x='gender', y='total_bill', data=tips_data);
```



7.2.2 Saving Plots in File

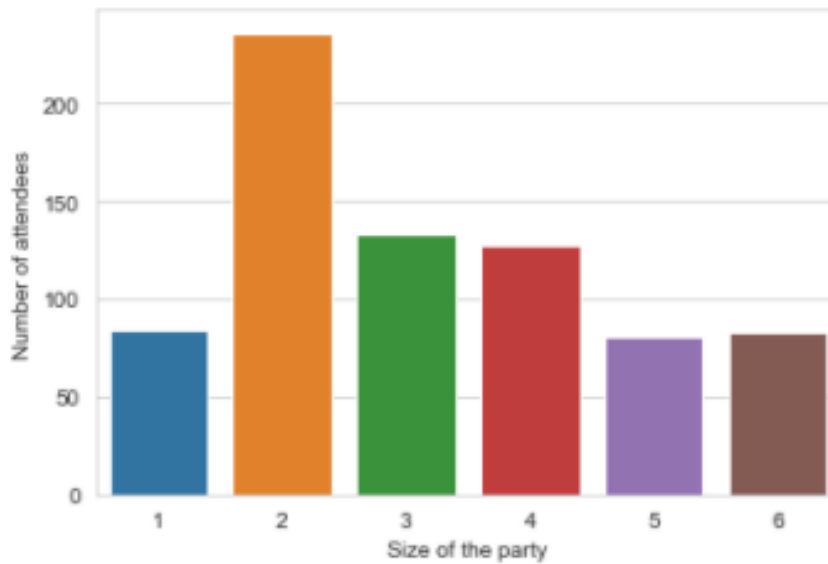
We can save or export the plot by using `plt.savefig()` function. First, we need to create our plot. For example, visualize size of the party and then use `plt.savefig()` to export a plot as a PNG file i.e. save it as `size.png`

```
sns.countplot(x = "size", data = tips_data)

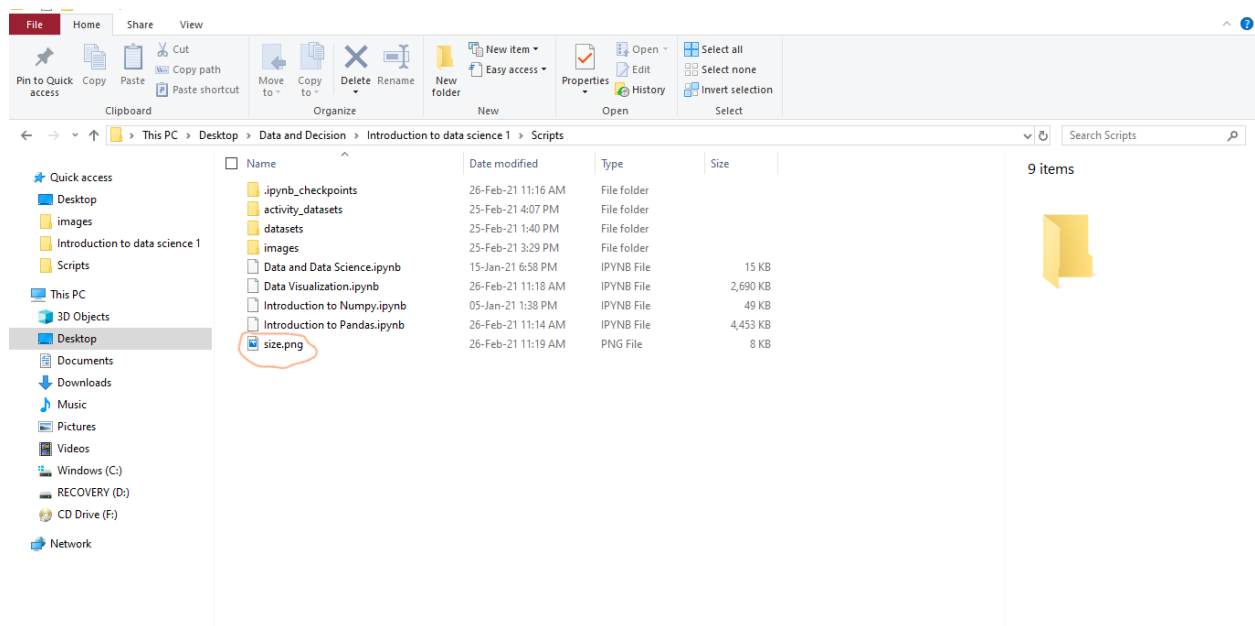
plt.xlabel("Size of the party")

plt.ylabel("Number of attendees")

plt.savefig("size.png")
```



Check your current working directory for [size.png](#) image



Class activity 14 (Pilot question 2)



Visualization of the Africa COVID-19 dataset

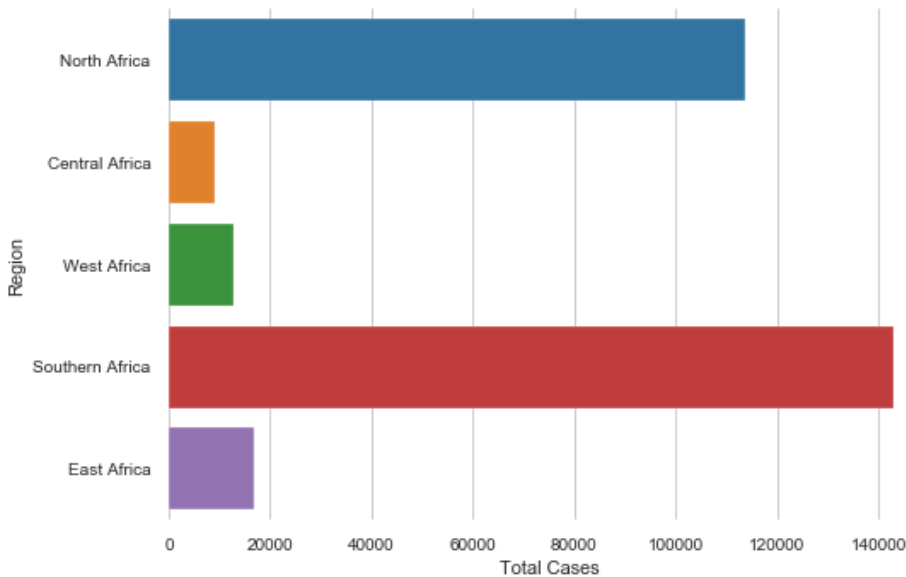
1. Import the necessary libraries
2. Import the **Africa COVID-19** dataset into your notebook. Note that this dataset is in the **activity_datasets** directory. If you encounter **UnicodeDecodeError** set **encoding = "ISO-8859-1"** in **pd.read_csv()**.
3. Draw a bar plot of region on the x axis and total cases on the y axis?
4. Remove the error bar by setting **ci = None**
5. Which region had a greater number of COVID-19 cases?

Pilot answer 2

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

covid = pd.read_csv("activity_datasets/Africa COVID-19 Dec 6, 2020.csv", encoding = "ISO-8859-1")

sns.barplot(y = "Region", x = "Total Cases", data = covid, ci = None);
```



South Africa region had a greater number of COVID-19 cases

Class activity 15 (Pilot question 3)



Visualization of the Penguins dataset (Part 1)

1. Import the necessary libraries
2. Import penguins dataset. Note that this dataset is in the **activity_datasets** directory.
3. Use the **.describe()** function to check the summary statistics on penguins DataFrame
4. Plot a boxplot for the **bill_length_mm** column by using **sns.boxplot()** function
5. Plot a histogram for the **body_mass_g** column

Pilot answer 3

Solution 1

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Solution 2

```
penguins = pd.read_csv("activity_datasets/penguins.csv")
```

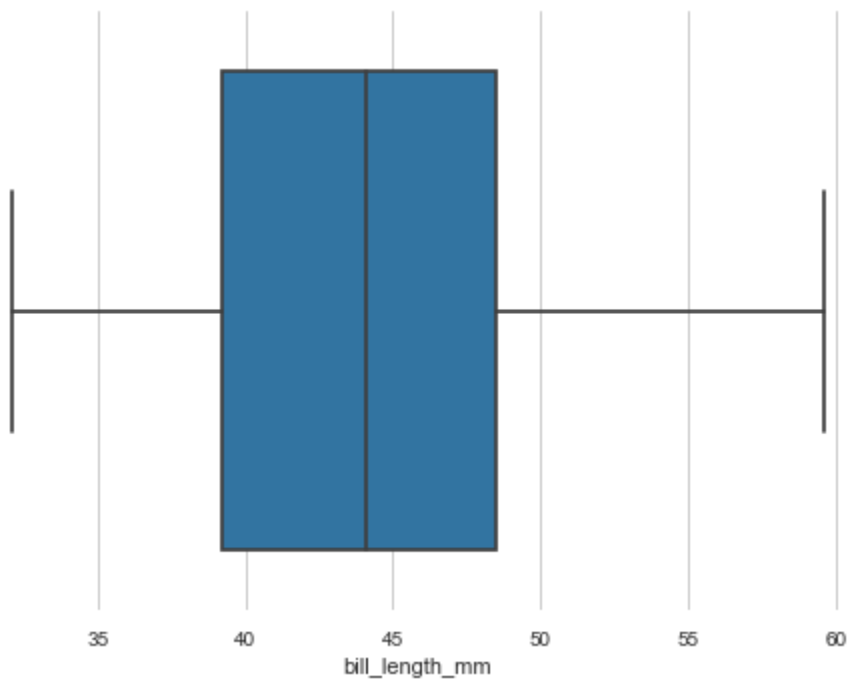
Solution 3

```
penguins.describe()
```

	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
count	337.000000	337.000000	337.000000	337.000000
mean	43.920772	17.176855	200.804154	4200.816024
std	5.484202	1.965112	14.021356	803.385972
min	32.100000	13.100000	172.000000	2700.000000
25%	39.200000	15.600000	190.000000	3550.000000
50%	44.100000	17.300000	197.000000	4000.000000
75%	48.500000	18.700000	213.000000	4750.000000
max	59.600000	21.500000	231.000000	6300.000000

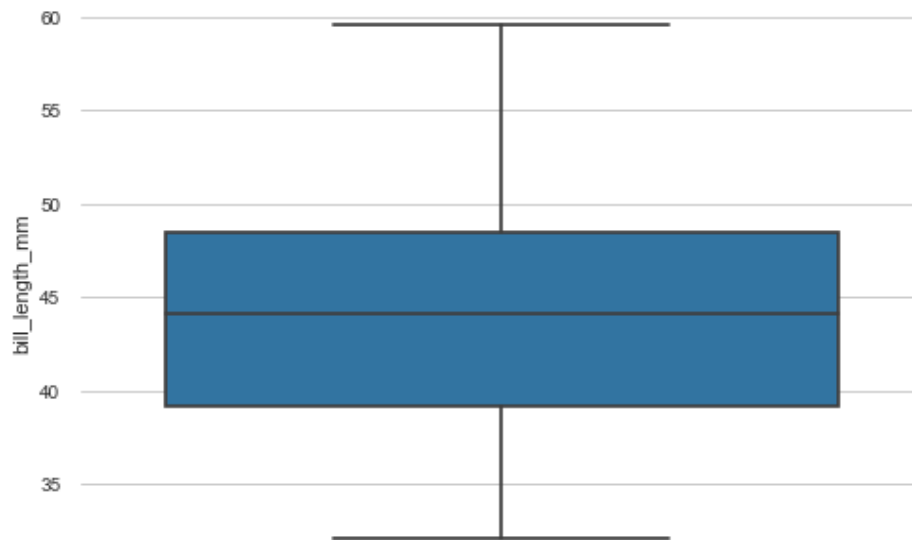
Solution 4

```
sns.boxplot(x = "bill_length_mm", data = penguins);
```



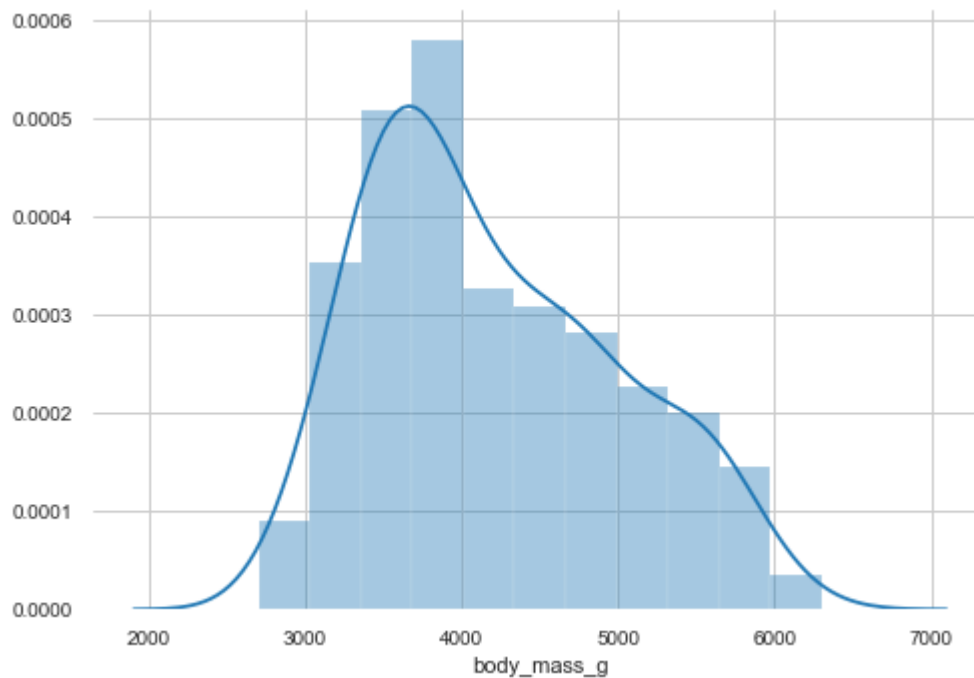
or


```
sns.boxplot(y = "bill_length_mm", data = penguins);
```



Solution 5

```
sns.distplot(penguins["body_mass_g"]);
```



Class activity 16 (Pilot question 4)



Visualization of the Penguins dataset (Part 2)

1. Import the necessary libraries
2. Import penguins dataset. Note that this dataset is in the **activity_datasets** directory.
3. Plot a boxplot with the x axis as **species** and the y axis as **flipper_length_mm** by using `sns.boxplot()` function. Change the y-axis label to **flipper length (mm)**
4. Use the plot from 4. and change the order of the species to “Adelie”, “Chinstrap”, and “Gentoo”
5. In the above plot set **hue** as **sex**
6. Plot a scatter diagram with the x axis as **bill_length_mm** and y axis as **flipper_length_mm** by using `sns.scatter()` function
7. Create a pairplot using `sns.pairplot()` on Penguins DataFrame

Pilot answer 4

Solution 1

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

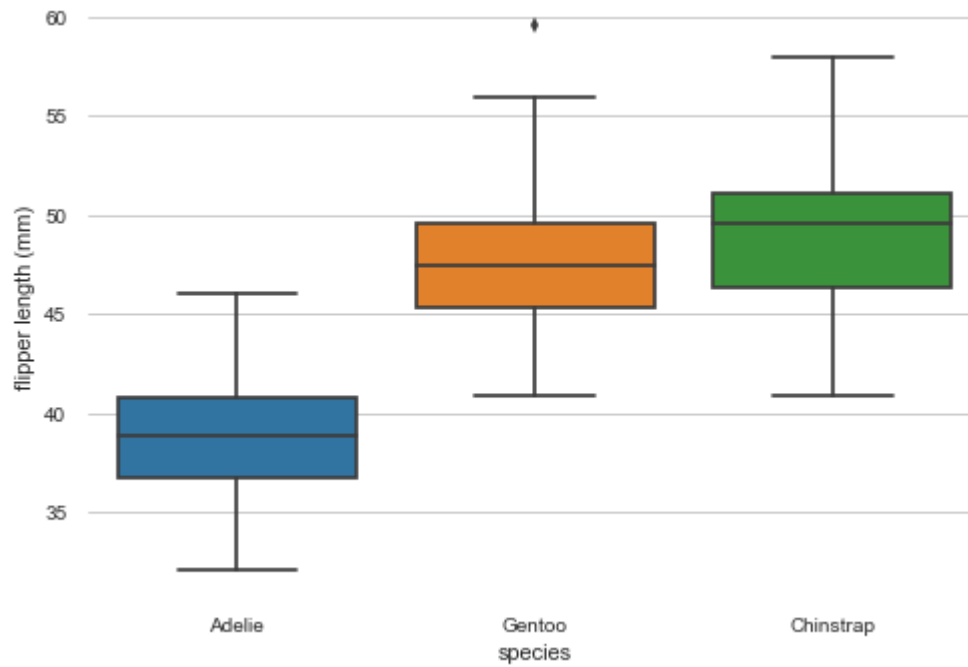
Solution 2

```
penguins = pd.read_csv("activity_datasets/penguins.csv")
```

Solution 3

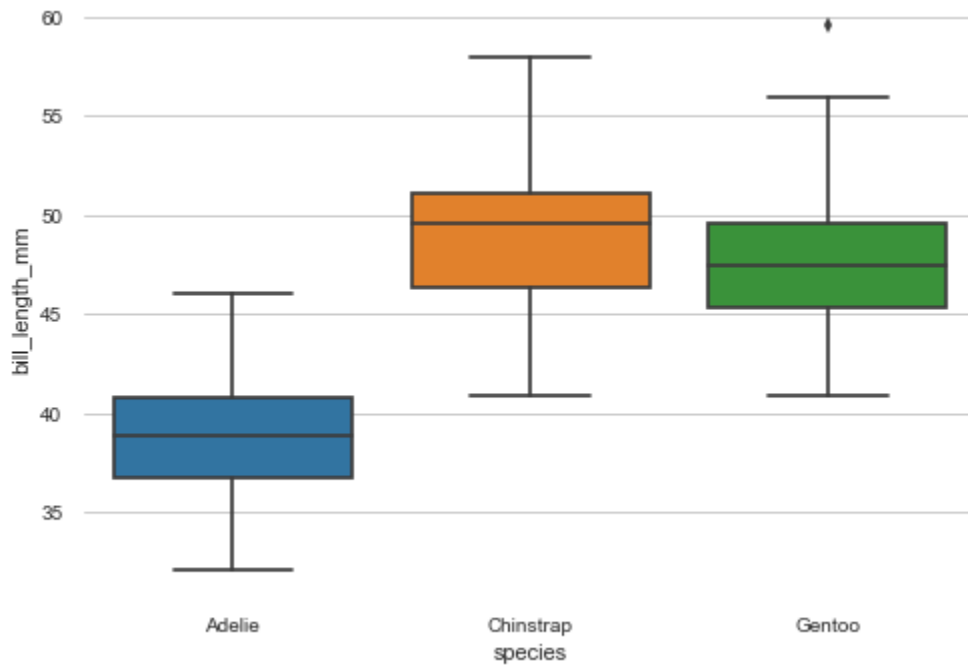
```
sns.boxplot(x = "species", y = "bill_length_mm", data = penguins)

plt.ylabel("flipper length (mm)");
```



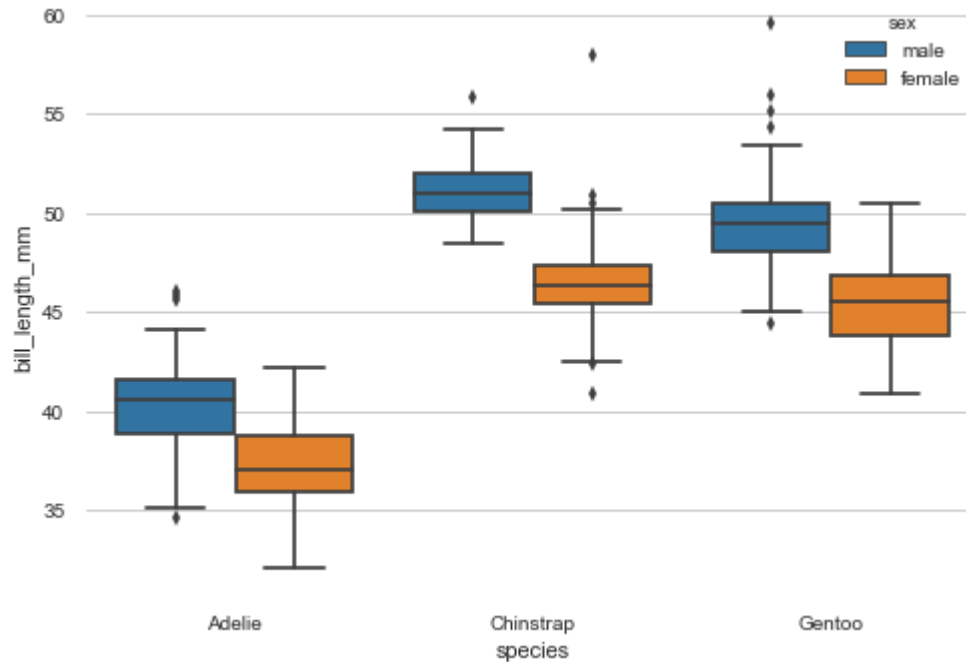
Solution 4

```
sns.boxplot(x = "species", y = "bill_length_mm", data = penguins, order = ["Adelie", "Chinstrap", "Gentoo"])
```



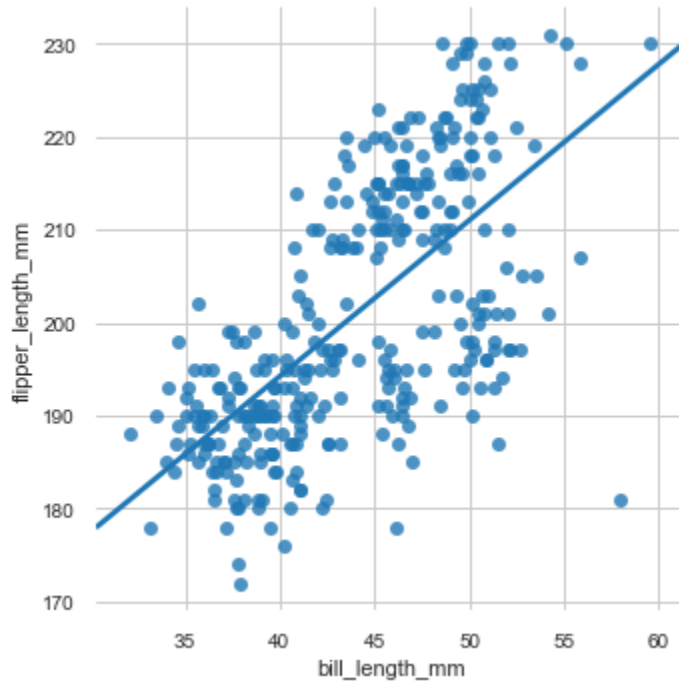
Solution 5

```
sns.boxplot(x = "species", y = "bill_length_mm", data = penguins, order = ["Adelie", "Chinstrap", "Gentoo"], hue = "sex");
```



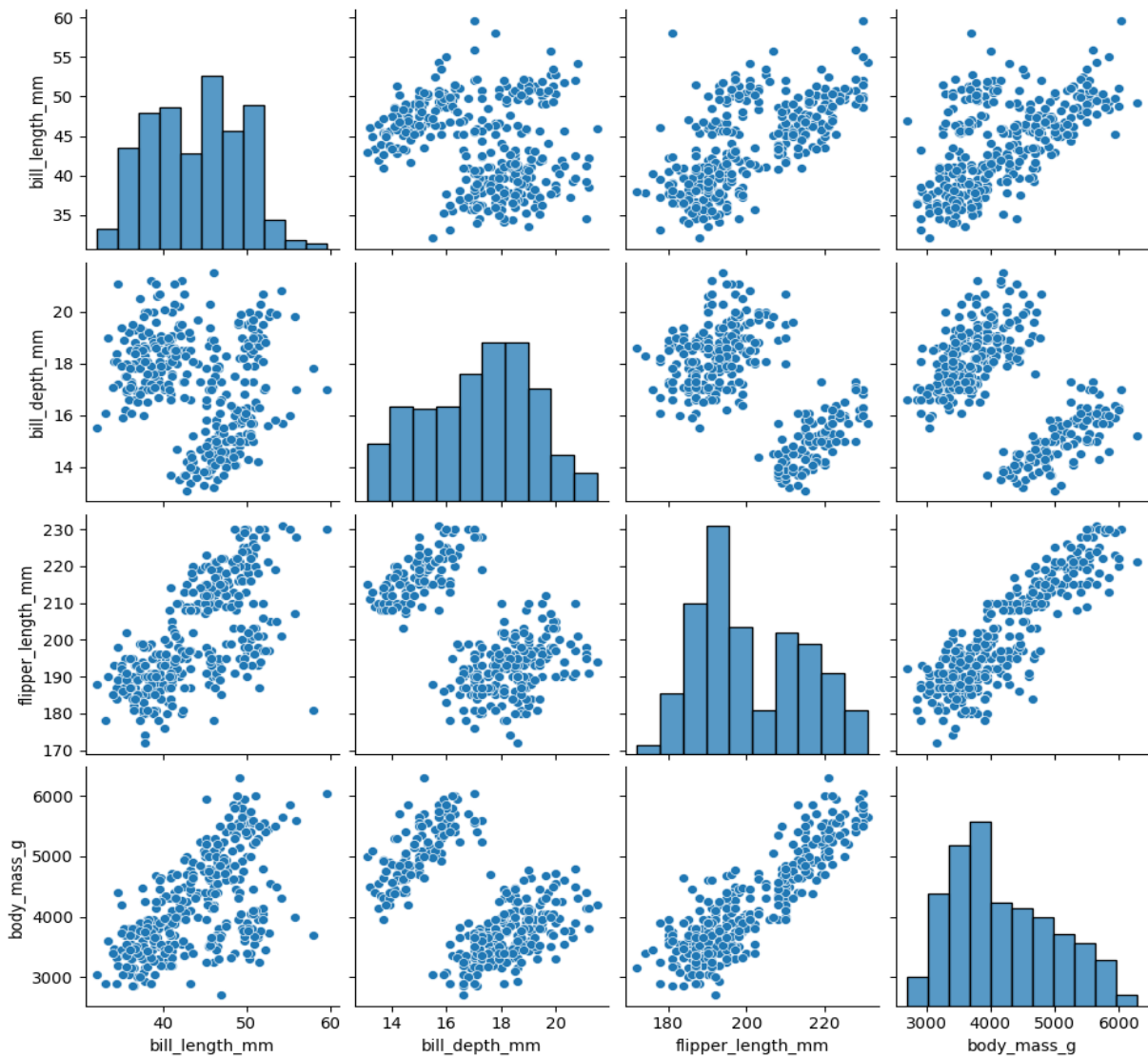
Solution 6

```
sns.lmplot(x = "bill_length_mm", y = "flipper_length_mm", data = penguins, ci = None);
```



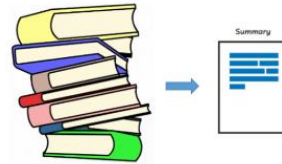
Solution 7

```
sns.pairplot(penguins);
```



Summary of Study Unit 7



In this study unit, you have learnt that:



1. Data visualization is the graphical representation of data by visual elements such as charts, Infographics, and maps to understand the data
2. Before you visualize your dataset, you need to answer five (5) questions in section [4.1](#)
3. We have different visualization designs or techniques such as bar chart, pie chart, histogram, scatter diagram, etc.
4. Some visuals are good to represent categorical data e.g. bar chart while some are good for continuous data e.g. histogram, scatter diagram, etc.
5. Scatter diagram is used to visualize the relationship between two continuous variables.

Additional resources

For more additional resources on data visualization, check the following resources:

-  <https://datagy.io/python-seaborn/>
-  <http://bit.ly/Introduction-to-seaborn-YouTube-video>
-  https://seaborn.pydata.org/tutorial/function_overview.html
-  <https://seaborn.pydata.org/tutorial/relational.html>
-  <https://seaborn.pydata.org/tutorial/categorical.html>