5.1.10 Plot a Pandas DataFrame and Series

Now that you're up to speed on all four types of charts (line, bar, scatter, and pie), you're eager to dig into the data. One of the questions V. Isualize is famous for asking during these presentations is about the exploratory part of the analytical process. In other words, when you have a new dataset, what is your process in determining relationships between variables in the data? Or what type of data do you have to work with? And are there any outliers or patterns in the data?

This critical, exploratory data analytic step can save roughly 15–50% of your time on a project because it provides a targeted plan for how to clean, sort, and create smaller datasets. It's not something a savvy analyst will skip. And of course you want V. Isualize to immediately recognize you as a savvy analyst!

Up to this point, you've been working with a small dataset, but soon you'll be analyzing large datasets that will be read into a DataFrame. Since you're getting to be an expert at creating visuals, Omar is going to show you how to plot data from a DataFrame just by referencing the data columns.

During data analysis, one best practice is to visualize the data once it's in a DataFrame to determine if it needs to be cleaned, sorted, or modified before it's analyzed. This can also prevent the scenario of being knee deep in the analysis and having to go back to the beginning to change a data type or create a feature. One way to do that is to plot data directly from a DataFrame or Data Series.

We can quickly visualize our data from a DataFrame or Series by using the `plot()` function by adding the Series (`ds`) or DataFrame (`df`) in the following format: `ds.plot()` or `df.plot()`.

To plot a DataFrame, we add the x and y values inside the parentheses as the DataFrame columns we want.

Let's briefly walk through how to plot data from a DataFrame, but first let's download the practice dataset into our `Resources` folder.

- In your PyBer_Analysis folder, create a folder named Resources.
- Click the following link to download the PyBer_ride_data.csv file into your Resources folder.

  Download PyBer_ride_data.csv (https://2u-data-curriculum-team.s3.amazonaws.com/dataviz-online/module_5/PyBer_ride_data.csv)

  You should now have a CSV file named `PyBer_ride_data.csv` in your Resources folder.
- Create a new Jupyter Notebook file named `PyBer_ride_data`.
- In the first cell, do the following.
  - Add the magic command `%matplotlib inline` on the first line.
  - Import the following dependencies.
    - `matplotlib.pyplot as plt`
    - `numpy as np`
    - `pandas as pd`
  - Read the `PyBer_ride_data.csv` into a Pandas DataFrame.

Your Jupyter Notebook file should look like this:

```
%matplotlib inline
# Dependencies
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
# Load in csv
pyber_ride_df = pd.read_csv("Resources/PyBer_ride_data.csv")
pyber_ride_df
```

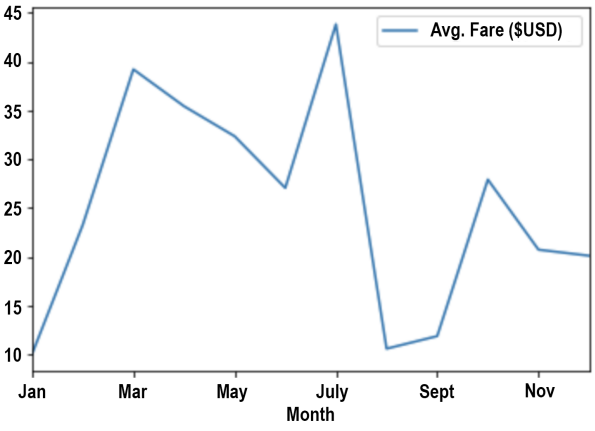
When we run this cell, the output is the ride-sharing data we've been working with in a DataFrame:

	Month	Avg. Fare ($USD)
0	Jan	10.02
1	Feb	23.24
2	Mar	39.20
3	April	35.42
4	May	32.34
5	June	27.04
6	July	43.82
7	Aug	10.56
8	Sept	11.85
9	Oct	27.90
10	Nov	20.71
11	Dec	20.09

We can plot the months along the x-axis and the fare on the y-axis using the `plot()` function.

```
pyber_ride_df.plot(x="Month", y="Avg. Fare ($USD)")
plt.show()
```

When we run this cell, it gives us a similar line chart to the one we saw before:



We can adjust the x-ticks to show all the months by editing our code to look like this:

```
# Set x-axis and tick locations.
x_axis = np.arange(len(pyber_ride_df))
tick_locations = [value for value in x_axis]
# Plot the data.
pyber_ride_df.plot(x="Month", y="Avg. Fare ($USD)")
plt.xticks(tick_locations, pyber_ride_df["Month"])
plt.show()
```

Let's examine the code to see what is going on:

1. First, we need an array that is the length of the DataFrame. This will allow us to plot each month for each value in the array. With the NumPy module, we create an array of numbers using the `arange` function for the length of the `pyber_ride_df`:

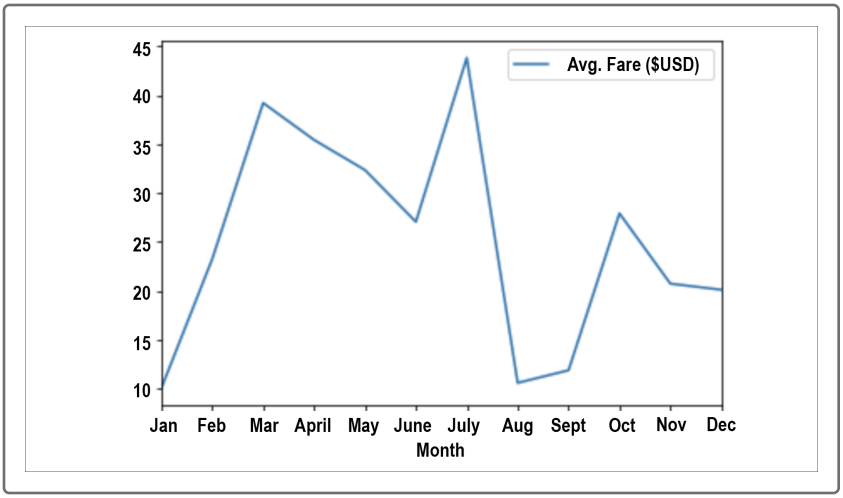
```
x_axis = np.arange(len(uber_ride_df))
x_axis
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

2. Then we set the tick locations for every value, or month, in the x-axis using list comprehension.

3. We add the `xticks()` function and pass in the `tick_locations` and the labels of the x-ticks, `pyber_ride_df["Month"]`.

When we run this cell, we get all the months on the x-axis:



There are two ways to create a bar chart with the same data. In the first approach, we "chain" the `bar()` function to the `plot()` function. Let's give that a try.

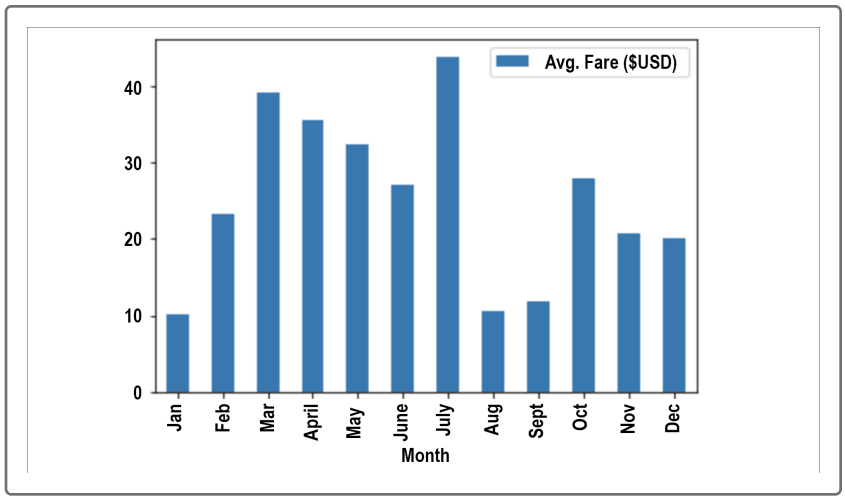
Add the following code to a new cell and run the cell to see a bar chart of the same data:

```
pyber_ride_df.plot.bar(x="Month", y="Avg. Fare ($USD)")
plt.show()
```

The other approach is to add the `kind` parameter to the `plot()` function. Add the following code to a new cell:

```
pyber_ride_df.plot(x="Month", y="Avg. Fare ($USD)", kind='bar')
plt.show()
```

Both of these approaches produce a bar chart with all the months on the x-axis, as you see here:



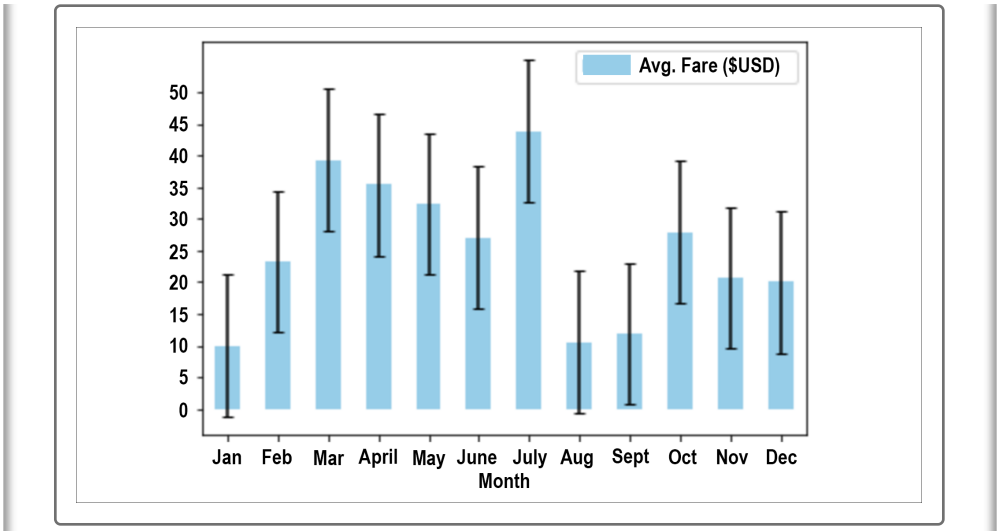
Now test your skills with the following Skill Drill.

SKILL DRILL

Using the data from the PyBer ride DataFrame, create a bar chart with the following annotations and characteristics:

1. Add error bars.
2. Add caps to the error bars.
3. Change the color of the bars to sky blue.
4. Rotate the labels on the x-axis to horizontal.
5. Set the y-axis increment to every $5.

Your chart should look similar to this:



Now that you're skilled in creating visualizations, it's time to move on to working with the "real" ride-sharing data. But before you do, here is a table that will help you decide which type of graph to create depending on what you want to show. We haven't yet covered all of these, like the stacked bar chart and the box-and-whisker plot, because we only had one dataset and we weren't looking for outliers. However, we may need to use them later in this module.

What Do You Want to Do?	Use These Charts
Compare values of datasets	Line, bar, scatter, and pie
Show how individual parts make up a whole	Pie and stacked bar
Show distribution of the data and outliers	Line, bar, scatter, and box-and-whisker
Show trends over time	Line or bar
Establish relationships between variables	Line, scatter, and bubble

NOTE

> For more information on plotting a Series or DataFrame, please see the **Pandas documentation on visualization (https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html)** .