5.1.7 Create Scatter Plots and Bubble Charts

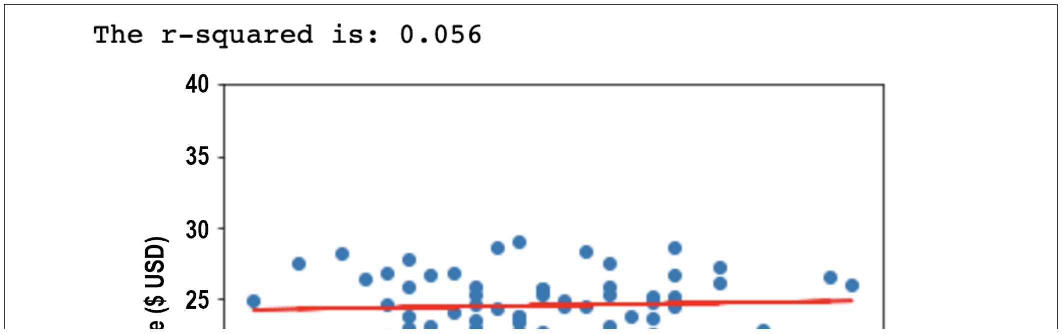
You've finished your bar charts, so you take a break and grab some lunch with a few other analysts on your team. While you're waiting for the food to arrive, you start talking with one of them, Sasha, and discover that she gave a presentation to V. Isualize and the executive team just last week!

When you tell Sasha that you're prepping some visualizations for V. Isualize on the rideshare data from 2019, she offers you a word of advice: be sure to include scatter plots! Sasha confides that she didn't think to include these in her presentation and V. Isualize seemed a bit disappointed, especially because scatter plots are particularly good at depicting multiple datasets on the same chart. Because Sasha just finished putting together some scatter plots as a follow-up to her own presentation, she offers to work with you after lunch to make sure you have some ready to go when you walk into the conference room to give your presentation.

You get your lunch to go, move a few meetings around, and head back to the office with Sasha ready to get to work on scatter plots. Thanks to her help, nothing in this presentation is going to catch you by surprise!

Scatter plots have many benefits over line and bar charts. A scatter plot visually depicts a positive, negative, or no-correlation relationship between two factors.

In the following scatter plot, we determine that there is no correlation between the number of rides and the average fare, because the *r* value is positive and close to zero. This is called a **regression analysis**. We'll walk through how to perform regression analysis in the next module.



Let's dive in to creating scatter plots. As we've done with other charts, we'll use the MATLAB method first and then the object-oriented interface.

Create a Scatter Plot Using the MATLAB Method

There are two ways to create scatter plots using the MATLAB approach:

- `plt.plot()`
- `plt.scatter()`

We'll learn how to use both of these, starting with with `plt.plot()`. We'll use the same ride-sharing data, shown here:

```
# Set the x-axis to a list of strings for each month.
x_axis = ["Jan", "Feb", "Mar", "April", "May", "June", "July", "Aug", "Sept"

# Set the y-axis to a list of floats as the total fare in US dollars accumul
y_axis = [10.02, 23.24, 39.20, 35.42, 32.34, 27.04, 43.82, 10.56, 11.85, 27.
```

If we were only to use the `plt.plot()` function with our x-axis and y-axis, it would create a line plot and that's not what we're looking for. So how do we make it a scatter plot?

When we use the `plt.plot()` function to create a scatter plot, we need to add a lowercase "o" as a parameter inside the parentheses. This switches the plot from a line chart to a scatter plot.

Add the following code to your `matplotlib_practice.ipynb` Jupyter Notebook file:

```
plt.plot(x_axis, y_axis, 'o')
```

When you run this code, you should see the following scatter plot:

To create a scatter plot using the `plt.scatter()` function, add the x-axis and y-axis parameters inside the parentheses, as shown here:

```
plt.scatter(x_axis, y_axis)
```

When we execute the code, the result should be identical to the graph when we used `plt.plot(x_axis, y_axis, 'o')`.

SKILL DRILL

Using the Matplotlib MATLAB plotting approach, make the following changes to your scatter plot:

1. Change the color of the markers to red.
2. Add a legend for the city of Chicago.

3. Add a title and axes labels.
4. Switch the axis so that the Fare($) data is on the x-axis.
5. Add a limit for the x-axis data.
6. Invert the y-axis so that January is at the top.

Create a Bubble Chart Using the MATLAB Method

Bubble charts are useful when presenting financial, population, and weather data because you can add a third and fourth factor to convey more information.

The first two factors are the x- and y-axes data, which we have been using quite frequently. By changing the "dot" into a "bubble," we are adding a third factor: size. If there is more than one dataset that uses the same axes, we can change the color of each marker for each dataset, which will add a fourth factor: color.

Let's see how to create a bubble chart by making a simple modification to the existing scatter plot code.

In a new cell, add the following code:

```
plt.scatter(x_axis, y_axis)
```

Next, to change the size of our marker on the graph, we have to use the `s`, or size parameter, for the marker. The `s` parameter must be equal to a feature of our data. We will make the size of each marker equal to the data floating-point decimal values of the y-axis by using `s=y_axis`.

Go ahead and edit the `plt.scatter()` function so that it looks like this:

```
plt.scatter(x_axis, y_axis, s=y_axis)
```

After running this cell, the marker size has changed based on the value of the fare in USD, as you can see here:

Some of the markers look very small. Let's make them bigger so that the chart is more legible and impactful.

We can adjust the size by multiplying the data in the y-axis by any value. Let's multiply each data point in the y-axis by 3 and see what happens. To do this, we can iterate through the y-axis data and multiply each data point by 3 and add it to a new list, like this:

```
y_axis_larger = []
for data in y_axis:
  y_axis_larger.append(data*3)
```

Then we can use the new y-axis list for the s parameter:

```
plt.scatter(x_axis, y_axis, s=y_axis_larger)
```

After running this cell, the marker size has increased in size, as you can see here:

That's a much better chart to present to the CEO, isn't it?

We can refactor the code we used to create a scatter plot above, but instead of using a for loop to create the list `y_axis_larger` for the size parameter of the marker, we can use Python's list comprehension technique inside the `plt.scatter()` function. You can use **list comprehension** to replace many `for` and `while` loops. List comprehension is faster because it is optimized for Python to spot a predictable pattern during looping.

The format for list comprehension is as follows.

```
new_list = [expression for item in list if conditional]
```

Using the list comprehension technique to multiply each data point in the `y-axis` list, we would do the following in the `plt.scatter()` function:

```
plt.scatter(x_axis, y_axis, s = [i * 3 for i in y_axis])
```

When we run this line of code, the output will be the same as before.

NOTE

> For more information about creating scatter plots and bubble charts using the MATLAB method, as well as using list comprehension, refer to the following documentation:
>
> [Matplotlib documentation on matplotlib.pyplot.scatter (https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.scatter.html#matplotlib.pyplot.scatter)](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.scatter.html#matplotlib.pyplot.scatter)

> **Matplotlib documentation on a simple scatter plot (https://matplotlib.org/stable/gallery/shapes_and_collections/scatter.html#sphx-glr-gallery-shapes-and-collections-scatter-py)**
>
> **Python documentation on list comprehension (https://docs.python.org/3.7/tutorial/datastructures.html#list-comprehensions)**

Create a Scatter Plot Using the Object-Oriented Interface

Now that we've created a scatter plot using the MATLAB approach, let's make one using the object-oriented approach.

In a new cell, add the following code:

```
fig, ax = plt.subplots()
ax.scatter(x_axis, y_axis)
```

When you run that cell, here's what your scatter plot will look like:

Great job! Now let's create a bubble chart.

Create a Bubble Chart Using the Object-Oriented Interface

To create a bubble chart using the object-oriented interface approach, we'll add the `s` parameter just like we did when we used the MATLAB approach.

In a new cell, add the following code:

```
fig, ax = plt.subplots()
ax.scatter(x_axis, y_axis, s=y_axis)
```

When you run the cell, you should see the same graph that you created using the MATLAB approach:

Now test your skills in the following Skill Drill.

SKILL DRILL

Using the object-oriented approach, make the following changes to your bubble chart:

1. Change the color of the markers to sky blue.
2. Change the size of the markers to 5 times each data point.
3. Make the color 20% transparent.
4. Add a black edge color to the circles.
5. Make the linewidth of the circles 2 points.
6. Add a legend for the city of Boston.
7. Add a title.
8. Switch the axis so that the Fare($) data is on the x-axis.
9. Add a limit for the x-axis data.
10. Invert the y-axis so that January is at the top.

NOTE

> For more information on creating scatter plots and bubble charts using the object-oriented method, please refer to the following documentation:
>
> **Matplotlib documentation on matplotlib.axes.Axes.scatter (https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.scatter.html)**
>
> **Matplotlib documentation on scatter charts with a legend (https://matplotlib.org/stable/gallery/lines_bars_and_markers/scatter_with_legend.html#sphx-glr-gallery-lines-bars-and-markers-scatter-with-legend-py)**
>
> **Matplotlib documentation on plotting a list of the named colors supported by Matplotlib (https://matplotlib.org/stable/gallery/color/named_colors.html)**