Section 3: Data Visualization

The human brain excels at finding patterns in visual representations of the data; so in this section, we will learn how to visualize data using `pandas` along with the `matplotlib` and `seaborn` libraries for additional features. We will create a variety of visualizations that will help us better understand our data.

Plotting with pandas

We can create a variety of visualizations using the `plot()` method. In this section, we will take a brief tour of some of this functionality, which under the hood uses `matplotlib`.

Once again, we will be working with the TSA traveler throughput data that we cleaned up in the previous section:

In [1]:

Out[1]:

	year	travelers	holiday
date			
2019-01-01	2019	2126398.0	New Year's Day
2019-01-02	2019	2345103.0	New Year's Day
2019-01-03	2019	2202111.0	NaN
2019-01-04	2019	2150571.0	NaN
2019-01-05	2019	1975947.0	NaN

To embed our plots in the notebook, we will also call the `%matplotlib inline` magic:

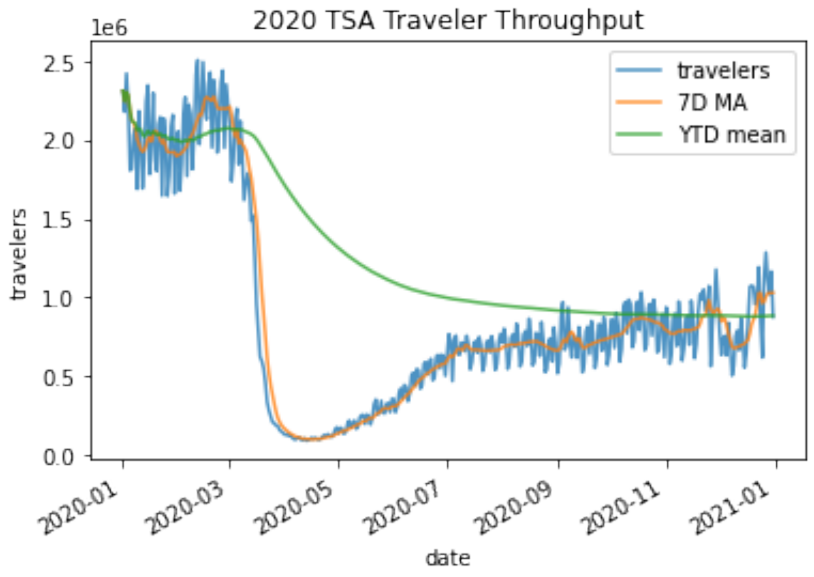
In [2]:

Line plots

The `plot()` method will generate line plots for all numeric columns by default:

In [3]:

Out[3]:
```
<AxesSubplot:title={'center':'2020 TSA Traveler Throughput'}, xlabel='date', y
label='travelers'>
```



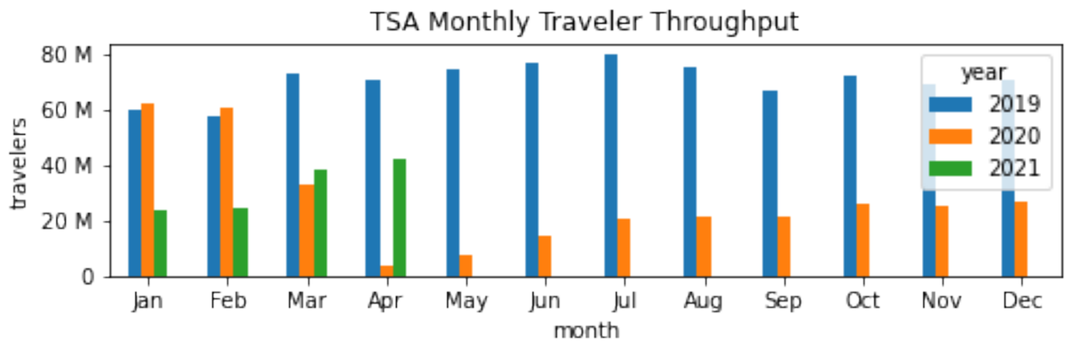
Tip: This method returned an `Axes` *object that can be modified further (e.g. to add reference lines, annotations, labels, etc.).*

Bar plots

Pandas offers other plot types via the `kind` parameter. Here, we plot vertical bars to compare monthly TSA traveler throughput across years. Then, we further format the visualization using the `Axes` object returned by the `plot()` method:

In [4]:
```

```



Some additional things to keep in mind:

- Matplotlib's `ticker` module provides functionality for customizing both the tick labels and locations – check out the documentation for more information.
- Pandas supports horizontal and stacked bars as well; this blog post shows how to make stacked horizontal bars using a pivot table.
- The `plot()` method takes a lot of parameters, many of which get passed down to `matplotlib`; however, sometimes we need to use `matplotlib` calls directly.

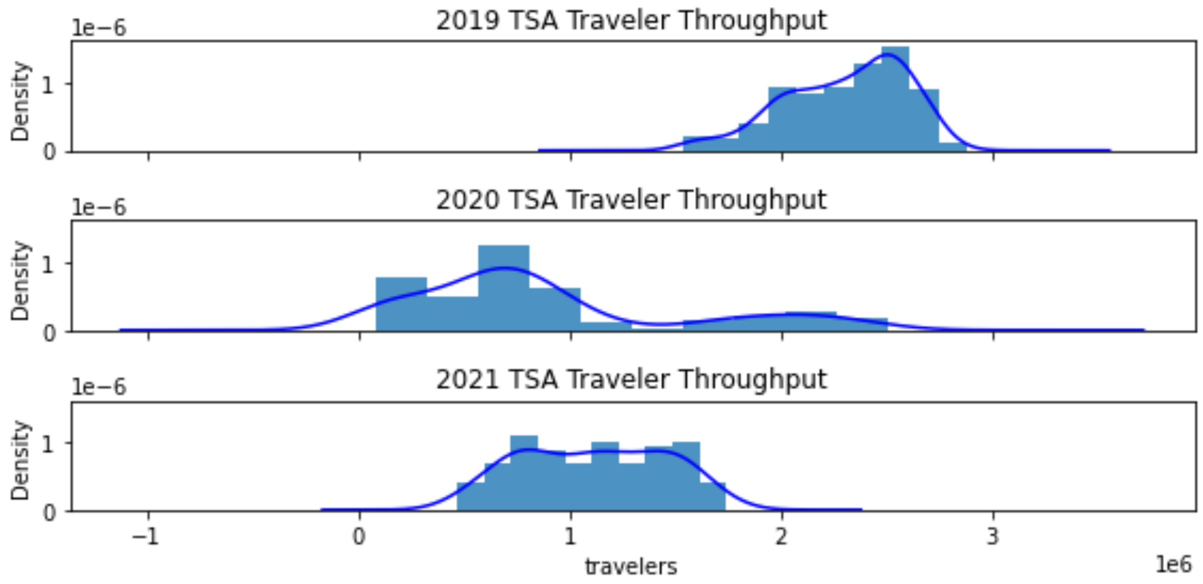
Plotting distributions

Pandas has generated the `Figure` and `Axes` objects for both examples so far, but we can build custom layouts by creating them ourselves with `matplotlib` using the `plt.subplots()` function. First, we will need to import the `pyplot` module:

```
In [5]:
```

Let's compare the distribution of daily TSA traveler throughput across years:

```
In [6]:
```



Tip: While `pandas` lets us specify that we want subplots and their layout (with the `subplots` and `layout` parameters, respectively), this gives us additional flexibility.

Plotting with seaborn

The `seaborn` library provides the means to easily visualize long-format data without first pivoting it. In addition, it also offers some additional plot types – once again building on top of `matplotlib`. Here, we will look at a few examples of visualizations we can create with `seaborn`.

Visualizing long-format data

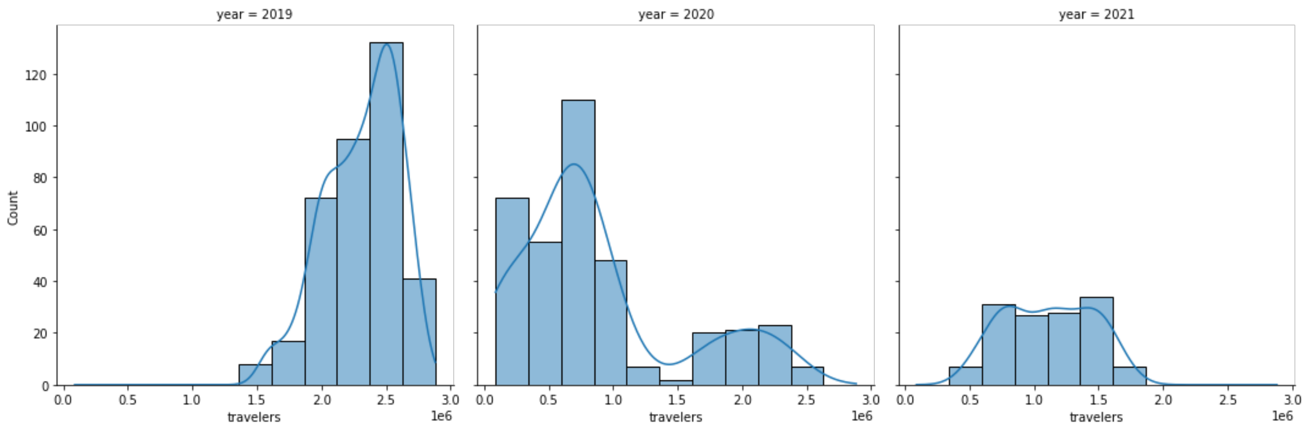
With `seaborn`, we can specify plot colors according to values of a column with the `hue` parameter. When working with functions that generate subplots, we can also specify how to split the subplots by values of a long-format column with the `col` and `row` parameters. Here, we revisit the comparison of the distribution of TSA traveler throughput across years:

In [7]:

```
```

Out[7]:

```
<seaborn.axisgrid.FacetGrid at 0x783ab75e4358>
```



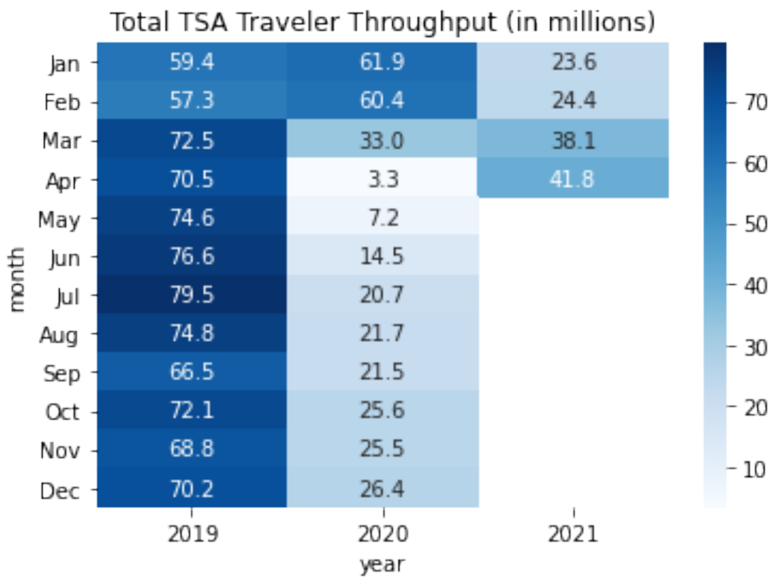
Heatmaps

We can use `seaborn` to visualize pivot tables as heatmaps:

In [8]:

```
```

Out[8]:
```
Text(0.5, 1.0, 'Total TSA Traveler Throughput (in millions)')
```



We're moving on from `seaborn` now, but there is a lot more available in the API. Be sure to check out the following at a minimum:

- pairwise plots with `pairplot()`
- categorical scatter plots with `swarmplot()`
- joint distribution plots with `jointplot()`
- FacetGrids for custom layouts with any plot type

Customizing plots with matplotlib

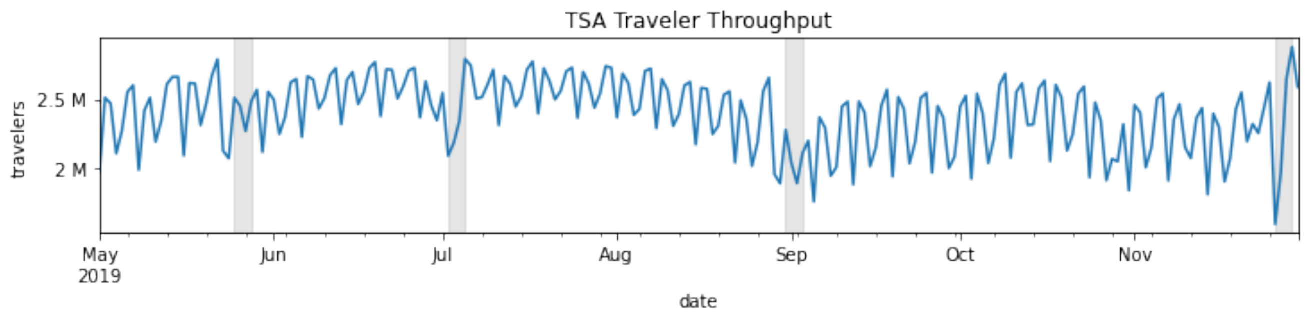
In this final section, we will discuss how to use `matplotlib` to customize plots. Since there is a lot of functionality available, we will only be covering how to add shaded regions and annotations here, but be sure to check out the documentation for more.

Adding shaded regions

When looking at a plot of TSA traveler throughput over time, it's helpful to indicate periods during which there was holiday travel. We can do so with the `plt.axvspan()` function:

In [9]:
```

```

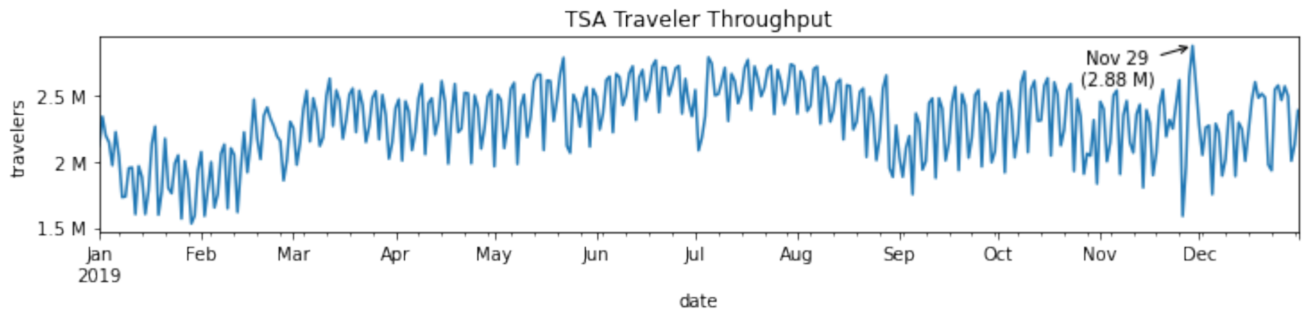


Tip: Use `plt.axhspan()` for horizontally shaded regions and `plt.axvline()` / `plt.axhline()` for vertical/horizontal reference lines.

Adding annotations

We can use the `plt.annotate()` function to add annotations to the plot. Here, we point out the day in 2019 with the highest TSA traveler throughput, which was the day after Thanksgiving:

```
In [10]:
```



Some things to keep in mind:

- We used `plt` functions to customize our plots, but `Axes` objects have equivalent methods for adding shaded regions, reference lines, annotations, etc. – although the method names might be slightly different than their `plt` function counterparts (e.g. `Axes.set_xlabel()` vs. `plt.xlabel()`).
- Likewise, `plt` functions can be used to add plot titles and axis labels, with the important caveat that `Axes` methods should always be used instead when dealing with subplots – otherwise, the action will only affect the last subplot.

Up Next: Hands-On Data Analysis Lab

Let's take a 15-minute break for some exercises to check your understanding:

1. Create box plots of TSA traveler throughput for each year in the data. Hint: Pass `kind='box'` into the `plot()` method to generate box plots.
2. Using `seaborn`, create a heatmap that shows the 2019 TSA median traveler throughput by day of week and month. Hint: Make a pivot table first.
3. Annotate the medians in the box plot from *#1*. Hint: The `x` coordinates will be 1, 2, and 3 for 2019, 2020, and 2021, respectively. Alternatively, to avoid hardcoding values, you can use the `Axes.get_xticklabels()` method, in which case you should look at the documentation for the `Text` class.

Exercises

1. Create box plots of TSA traveler throughput for each year in the data. Hint: Pass `kind='box'` into the `plot()` method to generate box plots.

In [ ]:

2. Using `seaborn`, create a heatmap that shows the 2019 TSA median traveler throughput by day of week and month.

In [ ]:

3. Annotate the medians in the box plot from *#1*. Hint: The `x` coordinates will be 1, 2, and 3 for 2019, 2020, and 2021, respectively. Alternatively, to avoid hardcoding values, you can use the `Axes.get_xticklabels()` method, in which case you should look at the documentation for the `Text` class.

In [ ]: