Lesson 6

August 7, 2020

1 Lesson 6: Data Visualization II

1.1 Statistic & Python Refresher

In the previous chapter we covered a number of statistics concepts and others that are worth remembering. We'll briefly review them here, but if these concepts are new to you, please review Lesson 5.

Average (mean): The sum of all of the values divided by the number of values. The mean can be a useful tool for quickly summarize your data. However, the mean is an incomplete picture of what your distribution actually looks like. Specifically, it gives a sense of where the center of the data is, but not the spread of the data (how widely the values range).

Range (spread): The distance between the largest number in a set and the smallest number.

Standard Deviation: The difference in spread can be quantified by something called the "standard deviation." For now, the details of how this is calculated don't matter. The intuition is that for each value, we calculate how far it is from the mean and then combine all those differences together into a single number the standard deviation). Therefore, if many of the values are far from the mean, we get a big standard deviation, meaning that the spread of the values is high. Alternatively, if most of the values are close to the mean, we will get a small standard deviation, and that means that the spread is narrow.

1.1.1 Python Packages & Importing Files

Instead of writing our own code to deal with large data sets, we will use what we refer to as a Python package. Packages are collections of code someone else has already written to perform complicated tasks that you can then use so you don't have to reinvent the wheel! There are many packages available to serve all sorts of applications and needs.

Often times we need to do more than use packages, we need to import data from other computers. The syntax to achieve this for a sample file called HumanHeightWeightData.csv has been shown below.

[2]:
```
import pandas # work with big data
import numpy # statistics functions
human_data = pandas.read_csv('HumanHeightWeightData.csv')
```

1.2 Outliers

So far, we have seen some pretty symmetrical distributions, where all the data points were centered around the mean of 0. What if we had a distribution where most of the data points were still around a mean of 0, but we had one additional data point far away from this mean? We can visualize this case using the following lines of code:

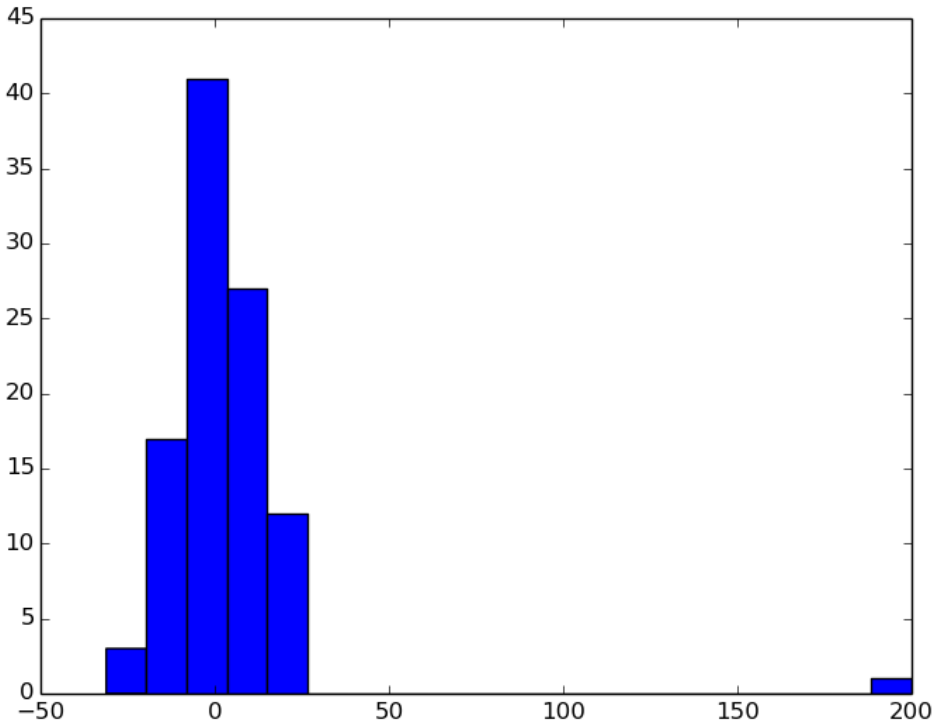
[3]:
```
# Import necessary packages for data visualization
import numpy # this package allows us to use some statistical functions
import matplotlib.pyplot as plt # this package allows us to graph data

# Create a histogram with a mean of 0 and a standard deviation of 10, with an␣
 ↪additional outlier data point
fig4 = plt.figure()
d = list(numpy.random.normal(loc = 0, scale = 10, size = 100)) + [200] # Create␣
 ↪a set of data points, 100 (size) of which have a mean (loc) of 0 and a␣
 ↪standard deviation (scale) of 10, bu with an additional outlier data point␣
 ↪equal to 200

# Note that it is not important if you perfectly understand this line of code␣
 ↪right now
plt.hist(d, bins = 20) # Plot these data

fig4.savefig('Histogram[0,10]+Outlier.png')
```

This produces the following histogram, located in the tab titled “Histogram[0,10]+Outlier.png”:



Notice that there is one data point that looks far away from the rest of the data (at 200 on the x-axis). We call these data points that do not fit the same pattern of the expected distribution "outliers." Verify yourself using the terminal below

1.2.1 Medians

In our outlier example, the mean of the expected distribution is 0 but if we calculate the mean of the data including the outlier, we get a non-zero answer. One statistic that is not as influenced by the outlier is the median. The median is calculated by ranking all of the data from the smallest value to the largest value and then selecting the middle value, referred to as the median. If we calculate the median for our data with the outlier, it will be much closer to the intuitive value of 0.

[1]:
```
# Import necessary packages for data visualization
import numpy # this package allows us to use some statistical functions
import matplotlib.pyplot as plt # this package allows us to graph data

d = list(numpy.random.normal(loc = 0, scale = 10, size = 100)) + [200] # Create␣
 ↪a set of data points, 100 (size) of which have a mean (loc) of 0 and a␣
 ↪standard deviation (scale) of 10, bu with an additional outlier data point␣
 ↪equal to 200

# Calculate the mean of the distribution, which is expected to be equal to 0 if␣
 ↪not for the outlier
```

```
print(numpy.mean(d))

# Calculate the median of the distribution, which is much closer to 0 than the␣
 ↪mean (less affected by the presence of an outlier)
print(numpy.median(d))
```

```
2.4987621322095044
0.955781824314872
```

1.3 Checkpoint 1

Use the median() command to calculate the median of the human height data.

[5]:
```
import numpy
import pandas
human_data = pandas.read_csv('HumanHeightWeightData.csv')
print(numpy.median(human_data.height_inches))
```

```
67.9957
```

Do not be concerned if the terminal displayed some additional information. This is just related to the importing and updating of packages

1.4 Boxplots

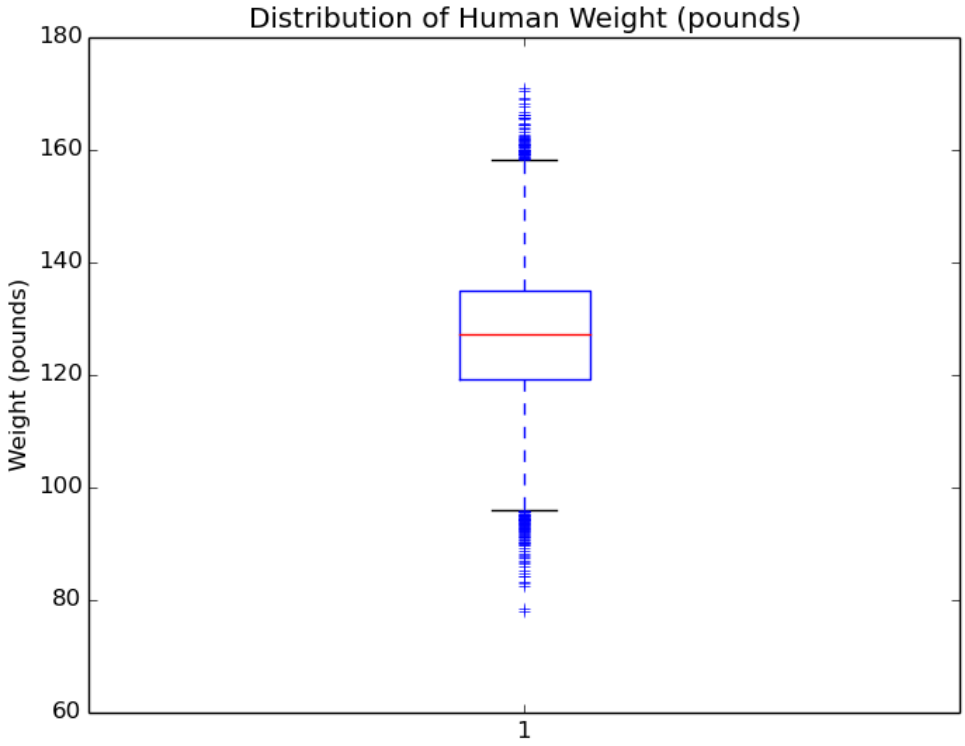
There are multiple ways to visualize data other than a histogram, with which it can be difficult to see the median of the distribution. We can also visualize medians (and distributions) with another type of graph called a box plot. Let's look at our human weight data in box plot form.

[6]:
```
# Import necessary packages for data import, analysis, and visualization
import pandas # this package allows us to work with two-dimensional data rather␣
 ↪than one-dimensional data
import matplotlib.pyplot as plt # this package allows us to graph data

# Load the data stored in a file called HumanHeightWeightData.csv into a␣
 ↪variable called human_data using pandas function read_csv()
human_data = pandas.read_csv('HumanHeightWeightData.csv')
# We need to re-import the packages and data in every new coding window

# Create a boxplot of the human weight data
fig1 = plt.figure()
plt.boxplot(human_data.weight_pounds)
plt.title('Distribution of Human Weight (pounds)')
plt.ylabel('Weight (pounds)')
fig1.savefig('WeightBoxPlot.png')
```

This code should yield a box plot titled WeightBoxPlot.png that looks like this:



Let's break this down. The red line in the middle, as you have likely guessed, is the median of the data. The upper blue line represents the 75th percentile of the data. The 75th percentile is the value at which 75% of the data points lie below it. So in this data, the 75th percentile is approximately 135, meaning that 75% of the data points are less than 135. The bottom blue line is the 25th percentile, which is the value at which 25% of the data points lie below it. So in this data, the 25th percentile is 120, meaning that 25% of the data points are less than 120.

The lines extending from either side of the box are called "whiskers" (box plots are also called box-and-whisker plots for this reason). They denote the range of the data; points outside this range are considered outliers and are plotted individually (the crosses on this graph).

So far, all the data we've looked at, even the data with a huge outlier, have been mostly symmetrical (except for the outlier itself) – the shape of the histogram is relatively similar on both sides of the mean.

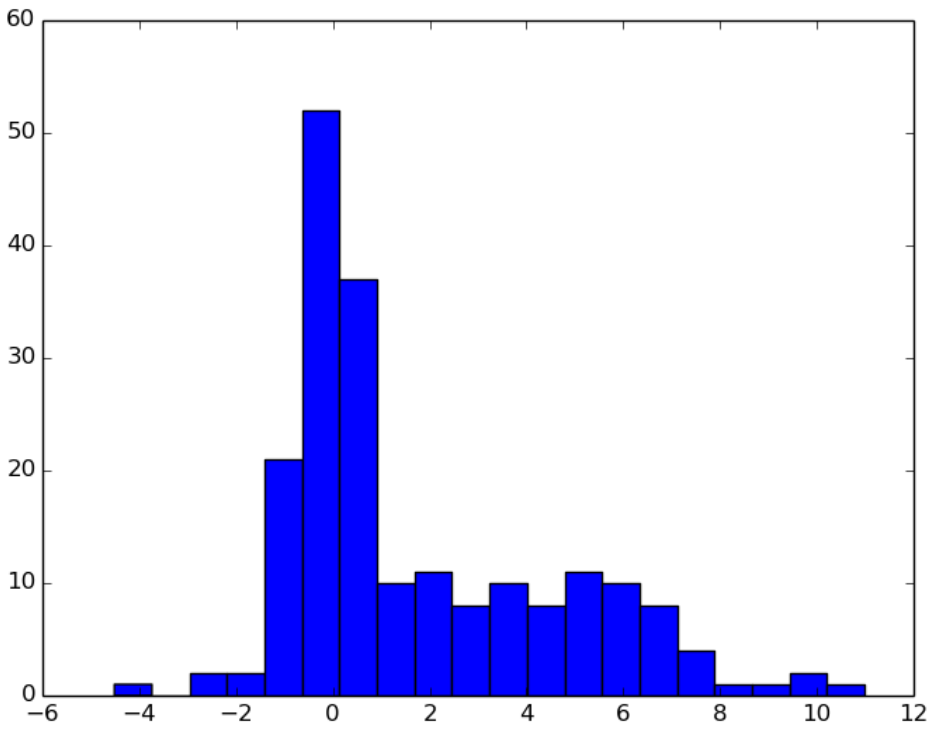
Sometimes data is very asymmetric, though. For example, let's look at the histogram generated with the code below.

[1]:
```
# Import necessary packages for data import, analysis, and visualization
import numpy # this package allows us to use some statistical functions
import matplotlib.pyplot as plt # this package allows us to graph data
```

```
# Create data, 100 points of which are sampled from a distribution with a mean␣
 ↪of 3 and a standard deviation of 3 and another 100 points of which are␣
 ↪sampled from a distribution with a mean of 0 and a standard deviation of 0.5
my_data = list(numpy.random.normal(loc = 3, scale = 3, size = 100)) +␣
 ↪list(numpy.random.normal(loc = 0, scale = .5, size = 100))
# Note that it is not important if you perfectly understand this line of code␣
 ↪right now

# Plot data as a histogram
fig1 = plt.figure()
plt.hist(my_data, bins = 20) # Plot these data
fig1.savefig('AsymmetricHistogram.png')
```

This generates a histogram titled AsymmetricHistogram.png, which should look something like this:



Mean and standard deviation quiz: - For the above data do you think… the mean is greater than the median; because there is a “skew” toward larger numbers, the mean of this distribution is greater than its median

Lets calculate the exact mean and median of the above distribution to check our intuition.

[1]:
```
# Import necessary packages for data import, analysis, and visualization
import numpy # this package allows us to use some statistical functions


# Create data, 100 points of which are sampled from a distribution with a mean␣
 ↪of 3 and a standard deviation of 3 and another 100 points of which are␣
 ↪sampled from a distribution with a mean of 0 and a standard deviation of 0.5
my_data = list(numpy.random.normal(loc = 3, scale = 3, size = 100)) +␣
 ↪list(numpy.random.normal(loc = 0, scale = .5, size = 100))
# Note that it is not important if you perfectly understand this line of code␣
 ↪right now

print("Mean: ", numpy.mean(my_data)) # Calculate mean
print("Median: ", numpy.median(my_data)) # Calculate median
```

```
('Mean: ', 1.4738596749513155)
('Median: ', 0.48856947604115525)
```

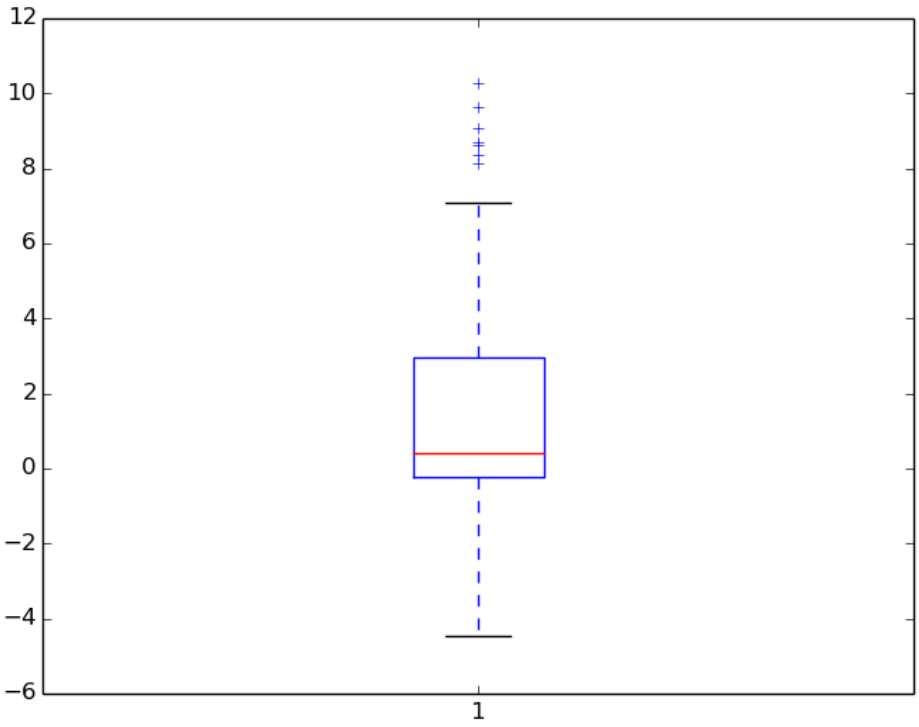
1.5 Checkpoint 2

Create a box plot using the skewed data. Notice that the median is no longer centered between the top and and bottom of the blue box – now it is lower because the range of values in our data above it is much larger than the range of values below it. That is, the values above the median range from about 0 to 12, while the values below range from only about 0 to -6.

[3]:
```
import numpy
import matplotlib.pyplot as plt

my_data = list(numpy.random.normal(loc = 3, scale = 3, size = 100)) +␣
 ↪list(numpy.random.normal(loc = 0, scale = 0.5, size = 100))

fig1 = plt.figure()
plt.boxplot(my_data)
fig1.savefig('Checkpoint2.png')
```



Do not be concerned if the terminal displayed some additional information. This is just related to the importing and updating of packages

1.6 Correlation

So far, we've been exploring height and weight separately, only looking at one variable at a time. But what if we wanted to learn about the relationship between those two variables? Intuitively, we would expect that taller people weigh more. Can we show this in our data?

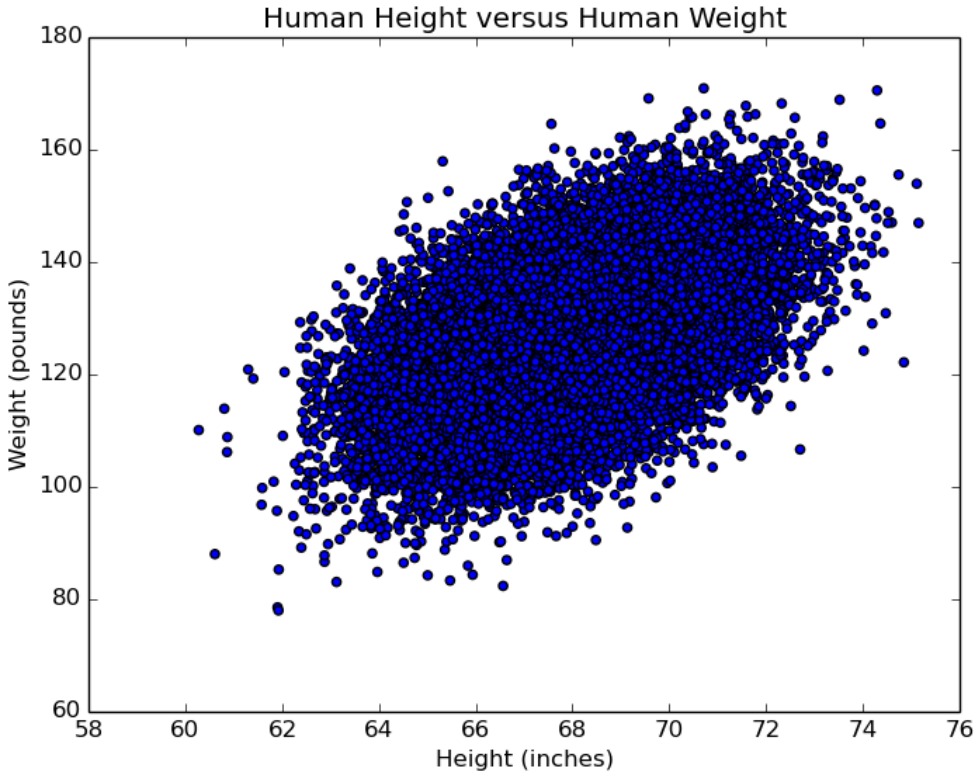
Programming and statistics offer a variety of tools to examine these kinds of relationships. We've already talked about histograms, but those won't be very informative for showing relationships between two different variables; they're mainly used for information about the distribution of a single variable at a time. To look at two variables, we commonly use a type of graph called a scatter plot.

On a scatter plot, the value of one of the variables is plotted along the x-axis, and the value of the other is plotted on the y-axis. If we consider our height and weight data, each point would represent one person, with their height plotted on one axis and their weight on the other. Like histograms and box plots, scatter plots are easily created using Python!

[ ]:
```
# Import necessary packages for data import, analysis, and visualization
import pandas # this package allows us to work with two-dimensional data rather␣
 ↪than one-dimensional data
import matplotlib.pyplot as plt # this package allows us to graph data
```

```
# Load the data stored in a file called HumanHeightWeightData.csv into a␣
 ↪variable called human_data using pandas function read_csv()
human_data = pandas.read_csv('HumanHeightWeightData.csv')
# We need to re-import the packages and data in every new coding window

# Create a boxplot of the human weight data
fig1 = plt.figure()
plt.scatter(human_data.height_inches, human_data.weight_pounds)
plt.title('Human Height versus Human Weight')
plt.xlabel('Height (inches)')
plt.ylabel('Weight (pounds)')
fig1.savefig('HeightWeightScatterPlot.png')
```

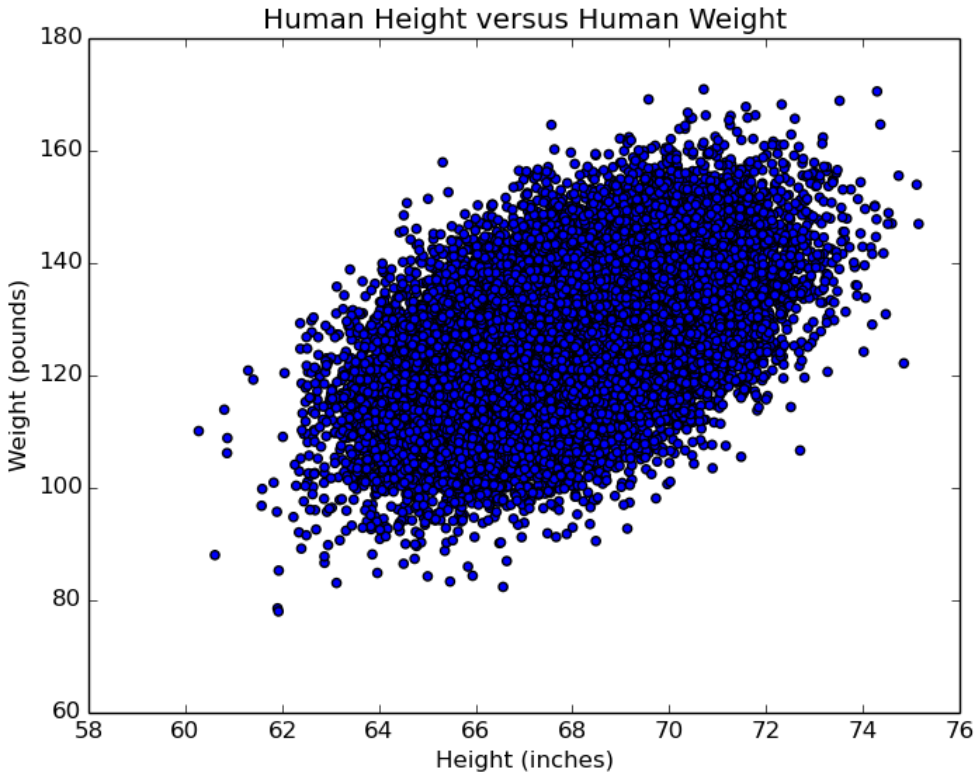


You might notice that we can't really tell how many points there are in the middle of the graph – there are so many points that there's tons of overlap. We can use a trick to help us improve our visualization: a parameter of graphs called alpha. In the graph setting, alpha controls the transparency of each point. Alpha can have any value from 0 to 1 (the default is 1, which is fully opaque). Let's try an example where we set alpha to 0.1, so we can see the point distribution a little more clearly (no pun intended).

```
[ ]: # Import necessary packages for data import, analysis, and visualization
import pandas # this package allows us to work with two-dimensional data rather␣
 ↪than one-dimensional data
import matplotlib.pyplot as plt # this package allows us to graph data

# Load the data stored in a file called HumanHeightWeightData.csv into a␣
 ↪variable called human_data using pandas function read_csv()
human_data = pandas.read_csv('HumanHeightWeightData.csv')
# We need to re-import the packages and data in every new coding window

# Create a boxplot of the human weight data
fig1 = plt.figure()
plt.scatter(human_data.height_inches, human_data.weight_pounds)
plt.title('Human Height versus Human Weight')
plt.xlabel('Height (inches)')
plt.ylabel('Weight (pounds)')
fig1.savefig('HeightWeightScatterPlot_withTransparency.png')
```



In general, we see that points with lower heights also have lower weights, and similarly, points with higher heights have higher weights. This causes a diagonal shape in the scatter plot, which helps us see that there is a correlation in these data. Specifically, these data are called positively correlated, whereas if higher heights had lower weights, and vice versa, we would say the data

are negatively correlated (in which case the diagonal would be flipped). Other variables that are positively correlated are temperature and ice cream consumed. Alternatively, there is a negative correlation between temperature and hot chocolate consumption. Life is certainly sweet year-round! Keep in mind, not all variables are correlated; there is no correlation between temperature and potato chip consumption (we can eat potato chips year-round too!).

We want to leave you with a final thought about correlation:

"Correlation does not equal causation"

— Every data scientist, ever

1.7 Summary

You should now understand the following:

Define mean, median, and standard deviation

Compare the utility of the mean compared to a full distribution to summarize data

Create interpretable histograms, box plots, and scatter plots using real biological data