Learning Objectives

- **Create a histogram**
- **Calculate and visualize histogram frequency densities**
- **Create a box plot**
- **Calculate and visualize the box plot median and quartiles**

definition

Assumptions

- Learners are comfortable reading and importing CSV data sets, extracting relevant data into data frames, and printing that data to the console.

Limitations

- This section will cover distribution charts in brief details only and will offer practical visualization functions for learners to start creating charts right away.

Histograms vs Bar and Column Charts

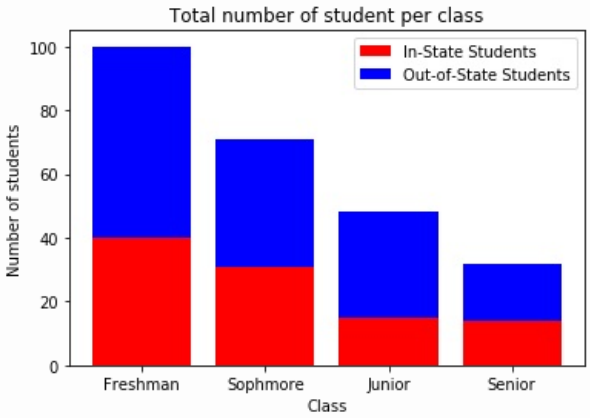
Bar charts and **column charts** are the same aside from orientation (vertical vs horizontal).

Histograms also have bars, making them look similar to column charts. However, these visualizations, despite their similarities visually, are different statistically.

Bar and Column Charts

Bar charts and column charts are a visualization of a categorical independent variable with a quantitative dependent variable.

In the example below, the categorical independent variable is the class or grade of students. The categories are listed along the x-axis: Freshman Sophomore, Junior, and Senior. The count of students is the quantitative dependent variable represented on the y-axis.

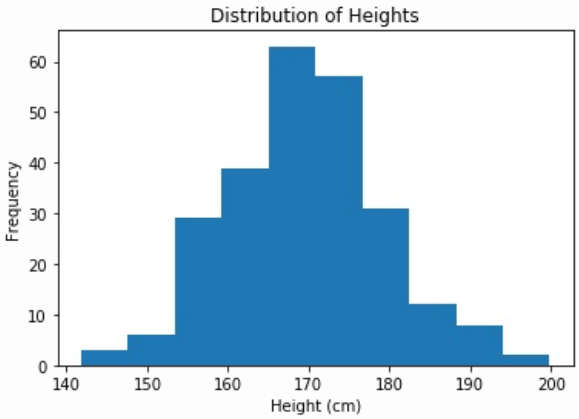


A bar or column chart allows one to visually compare categories.

Histograms

In contrast, a histogram has a quantitative variable that is charted on the x-axis. The quantitative range is broken up into **bins**. The number of data points, or **frequency**, in each bin is plotted on the y-axis.

In the example below, height is the quantitative variable plotted along the x-axis. You can see that there are 10 **bins** along the x-axis and the y-axis is frequency.



A histogram allows one to visually inspect the **distribution** or spread of the variable on the x-axis.

info

Spacing between bars

Histograms are typically plotted without space between the bars, while column/bar charts have space between the different categories bars.

While spacing between bars is the convention, it is not reliable to rely on bar spacing to distinguish between histograms and bar/column charts.

Creating Histograms

The basic syntax to create a histogram is:

```
plt.hist(data)
```

As noted on the previous page, you only need to provide one list of data since the `hist` method calculates the **bins** and **frequency**.

Creating a Histogram

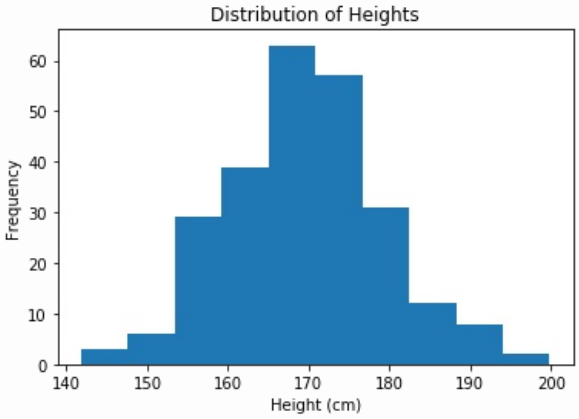
Let's generate some random data using `numpy`'s `random` method and plot it on a histogram:

```
import numpy as np
import matplotlib.pyplot as plt

height = np.random.normal(170, 10, 250)

plt.hist(height)
```

Be sure to include a title and axis labels:

```
plt.title('Distribution of Heights')
plt.xlabel('Height (cm)')
plt.ylabel('Frequency')
plt.show()
```

You should see a plot similar to the one below – but due to the randomized nature of the dataset, they will not be exactly the same.



important

Remember to clarify units

Remember to include units (like `cm`) when setting axis labels to make sure your plot is clear to the reader.

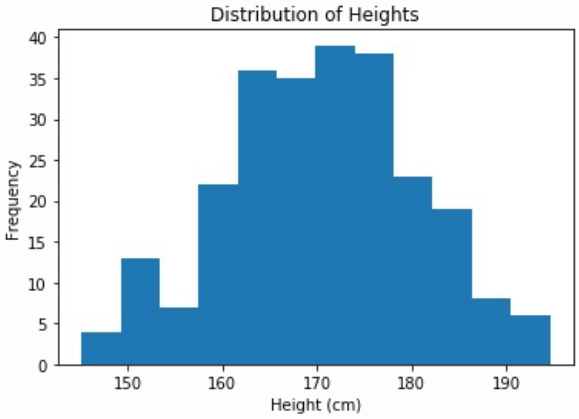
Specifying `bins` parameter

There are two different ways to specify bins:
1. the **number** of bins
2. the **edges** of the bins

You can simply give an integer to specify the number of bins. This is a great way to get more or less bars quickly:

```
plt.hist(height, bins=12)
```

We now have a couple more bins then before, which can reveal details in the distribution that were hidden:



info

Default value: bins=10

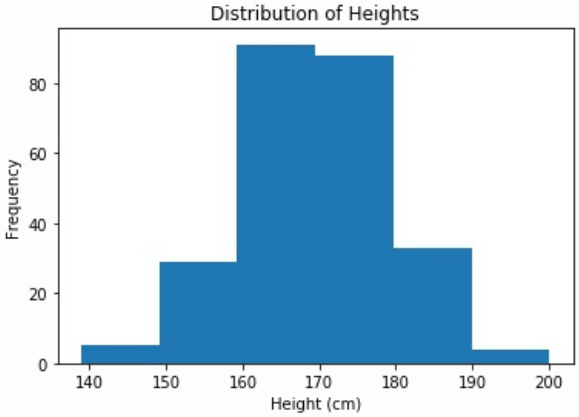
If you do not specify a `bins` parameter (like the very first example on this page), `hist` will default to 10 bins.

Sometimes, specifying the number of bins is not enough to get the plot to look how you want. Instead, you can specify where the bins are using the same `bins` parameter:

```
plt.hist(height, bins=[140,150,160,170,180,190,200])
```

The first bin is [140, 150) (including 140, but excluding 150) and the second [150, 160), and so on. The last bin, however, is [190, 200], which includes 200.

The resulting graph should look something like this:



Descriptive Statistics and Quartiles

Before moving on to **box plots** (sometimes called **box and whisker plots** due to their appearance), we need to understand some basic descriptive statistics.

We have used `numpy` previously to generate some randomized data for our histogram plot – but it also has a useful `quantile` method. Try out the following code:

```
import numpy as np
import matplotlib.pyplot as plt

data=[2,12,23,42,56,75,79,84]

print("Maximum: " + str(np.quantile(data, 1.0)))
print("Q3: " + str(np.quantile(data, 0.75)))
print("Median: " + str(np.quantile(data, 0.5)))
print("Q1: " + str(np.quantile(data, 0.25)))
print("Minimum: " + str(np.quantile(data, 0.0)))
```

You should see the following output:

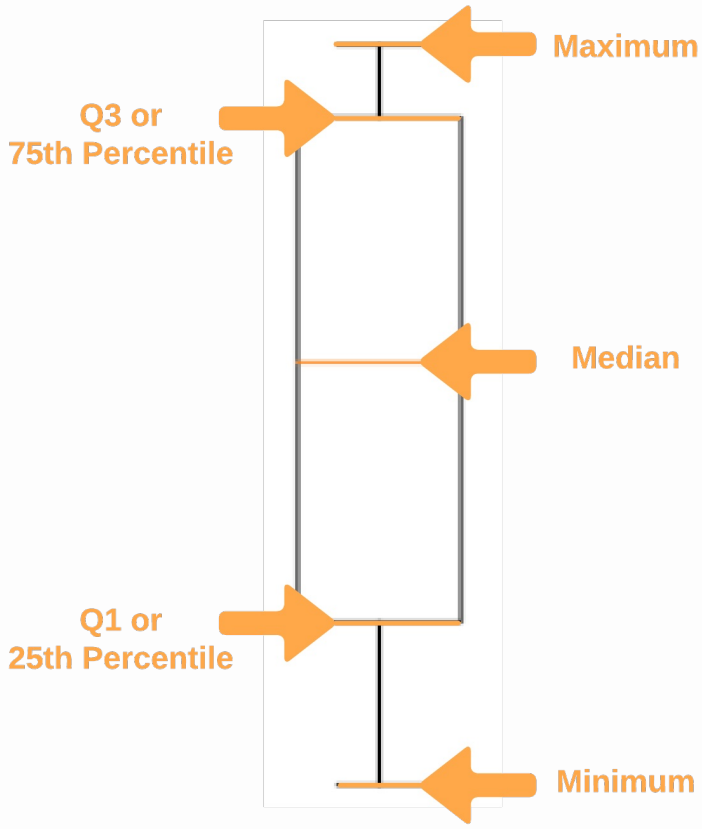
```
Maximum: 84.0
Q3: 76.0
Median: 49.0
Q1: 20.25
Minimum: 2.0
```

The **minimum** and **maximum** are the easiest to understand – they are the smallest and largest values in the dataset.

The **median** is the middle value. Because the above dataset is even, the two middle numbers (i.e. 42 and 56) are averaged to get 49. The median splits the data set in half – 4 data points are smaller than the median and 4 data points are larger than the median.

The **quartiles**, Q1 and Q3, split the halves created by the median in half again. Put another way, the quartiles mark where one quarter or 1/4th of the data is.

These 5 points of interest are the 5 horizontal lines that make up a box plot:



The box portion of the box plot, between Q1 and Q3 is often referred to as the **Interquartile Range** or **IQR**. This region represents the middle half of the data set.

The lines hanging off of the box are often referred to as **whiskers** – which is why this plot is sometimes referred to as a **box and whiskers plot**.

info

Fliers

On occasion, you might see dots that extend past the whiskers. These dots, sometimes called **fliers**, are **outliers**.

Matplotlib calculates what points are outliers as follows:
> The whiskers extend from the box by 1.5x the inter-quartile range (IQR). Flier points are those past the end of the whiskers.

Box Plots

Box plot are another way to visualize the distribution of quantitative values. As discussed on the previous page, box plots divide the data set into four quartiles, each containing $\frac{1}{4}$ of the data set.

Similar to histograms, the `boxplot` method in matplotlib only takes a single dataset parameter. The basic syntax is:

```
plt.boxplot(data)
```

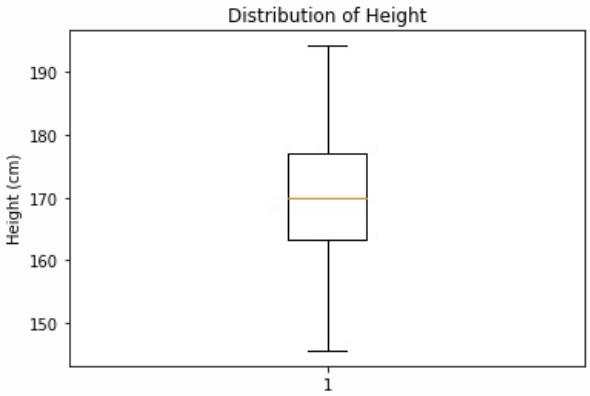
Creating Box Plots

Create a data set similar to the one we used for the histogram:

```
import matplotlib.pyplot as plt
import numpy as np

height = np.random.normal(170, 10, 250)

plt.boxplot(height)
```

Add a title and label (with units!):

```
plt.title('Distribution of Height')
plt.ylabel('Height (cm)')
plt.show()
```



Similar to the histogram, the box plot shows the high-density of heights around 170cm as well as the range of heights from about 140cm to 200cm.

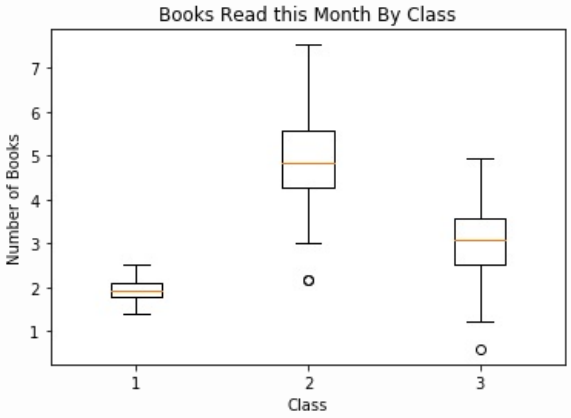
Multiple Box Plots

One reason to use Box Plots over histograms is the ability to easily compare different box plots.

Try plotting three boxes using the code below:

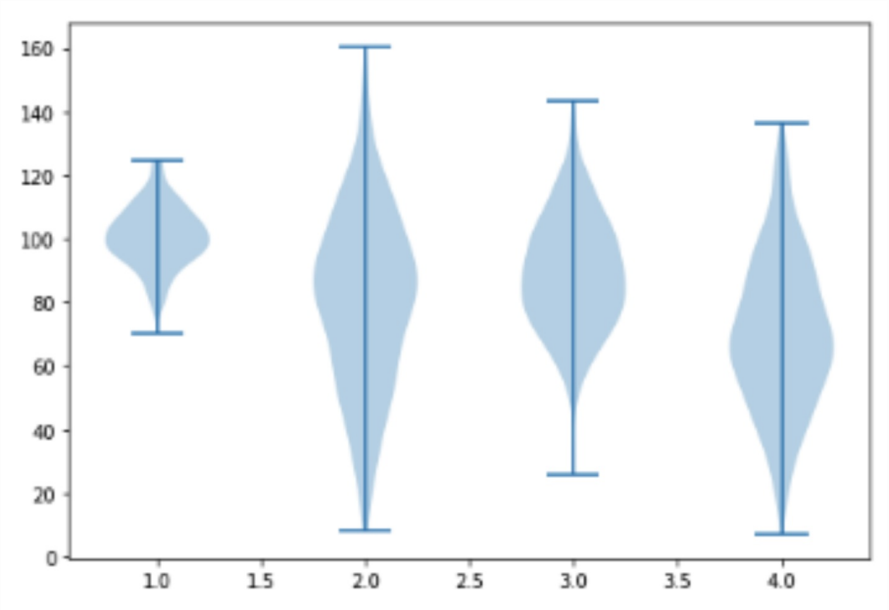
```
import matplotlib.pyplot as plt
import numpy as np

books = np.random.normal((2, 5, 3), (.25, 1.00, .75), (100, 3))

plt.boxplot(books)
plt.title('Books Read this Month By Class')
plt.xlabel('Class')
plt.ylabel('Number of Books')
plt.show()
```

From the output you can quickly compare the differences between the number of books read by each class of students this month:



Violin Plots

Violin plots are similar to box plot, they are use to visualize the distribution of numerical data. A violin plot is more informative than a plain box plot although less popular. Unlike a box plot that can only show summary statistics, violin plots depict the full distribution of data.



Creating Violin Plots

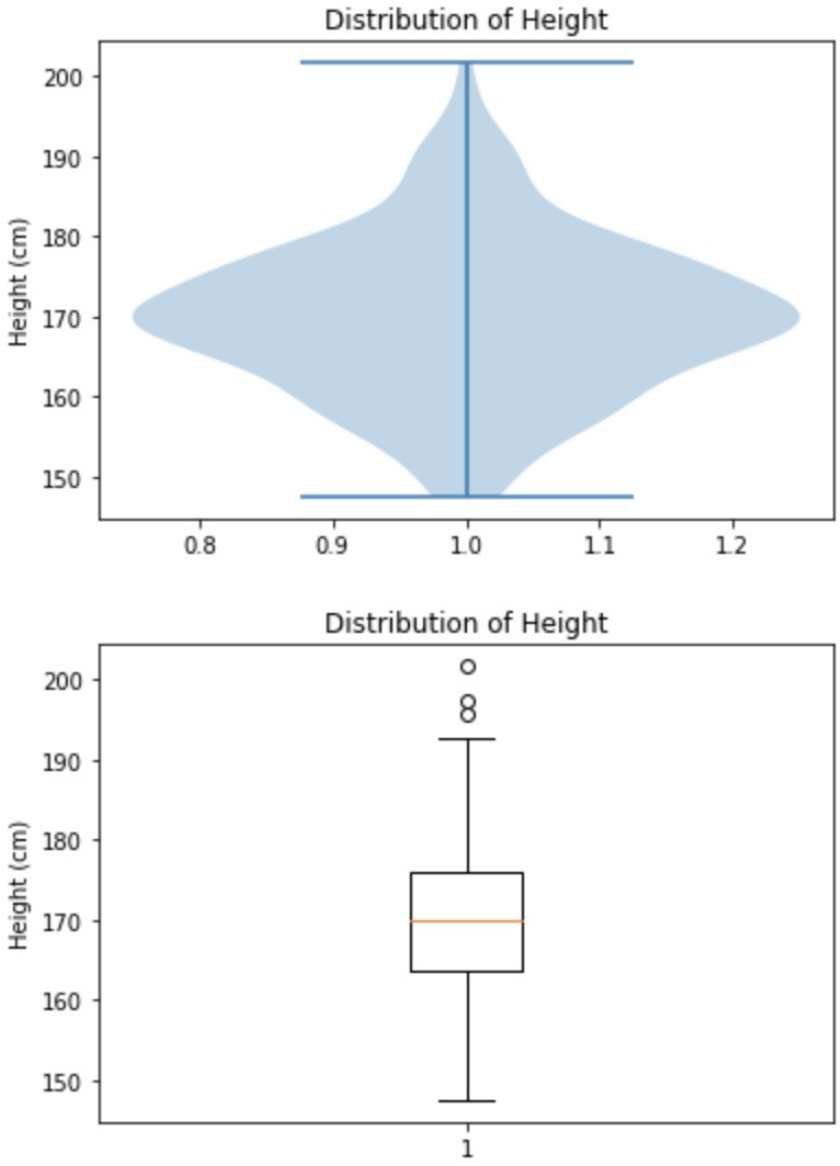
The basic syntax to create a violin plot is:

```
plt.violinplot(data)
```

We can actually create a boxplot and a violin plot on the same data set to compare the visualizations. We need to run `plt.figure()` in between the two commands:

```
import matplotlib.pyplot as plt
import numpy as np

height = np.random.normal(170, 10, 250)

plt.violinplot(height)
plt.title('Distribution of Height')
plt.ylabel('Height (cm)')
plt.show()

plt.figure()

plt.boxplot(height)
plt.title('Distribution of Height')
plt.ylabel('Height (cm)')
plt.show()
```

You should see two plots, like this:



Adding Quartiles

To get all the same richness of information that a box plot has about summary statistics on a violin plot, we can use a few parameters.

Without doing any work, you can see from the violin plots above, we already have the minimum and maximum indicated by horizontal lines.

We can add the median line by setting the parameter `showmedians=True`.

Finally, we need the lines for Q1 and Q3. We can set those using the `quantiles` parameter (i.e. `quantiles=[0.25, 0.75]`).

Your violin plot command with all 5 horizontal lines that the box plot has looks like this:

```
plt.violinplot(height, showmedians=True, quantiles=[0.25, 0.75])
```

If you look at these side-by-side now, you can see how the horizontal lines are aligned: