. Quantitative comparisons: histograms

Bar-charts show us the value of a variable in different conditions. Now, we're going to look at histograms. This visualization is useful because it shows us the entire distribution of values within a variable.

2. Histograms

Let's look at another example. In this case, we are looking at data about the athletes who participated in the 2016 Olympic Games. We've extracted two DataFrames from this data: all of the medal winners in men's gymnastics and all of the medal winners in men's rowing. Here are the five first rows in the men's rowing DataFrame. You can see that the data contains different kinds of information: what kinds of medals each competitor won, and also the competitor's height and weight.

3. A bar chart again

Let's start by seeing what a comparison of heights would look like with a bar chart. After creating the Figure and Axes objects, we add to them a bar with the mean of the rowing "Height" column. Then, we add a bar with the mean of the gymnastics "Height" column. We set the y-axis label and show the figure, which gives us a sense for the difference between the groups.

4. Introducing histograms

But a histogram would instead show the full distribution of values within each variable. Let's see that. We start again by initializing a Figure and Axes. We then call the Axes hist method with the entire "Height" column of the men's rowing DataFrame. We repeat this with the men's gymnastics DataFrame. In the histogram shown, the x-axis is the values within the variable and the height of the bars represents the number of observations within a particular bin of values. For example, there are 12 gymnasts with heights between 164 and 167 centimeters, so the highest bar in the orange histogram is 12 units high. Similarly, there are 20 rowers with heights between 188 and 192 centimeters, and the highest bar in the blue histogram is 20 units high.

5. Labels are needed

Because the x-axis label no longer provides information about which color represents which variable, labels are really needed in histograms. As before, we can label a variable by calling the hist method with the label key-word argument and then calling the legend method before we call plt-dot-show, so that a legend appears in the figure.

6. Customizing histograms: setting the number of bins

You might be wondering how Matplotlib decides how to divide the data up into the different bars. Per default, the number of bars or bins in a histogram is 10, but we can customize that. If we provide an integer number to the bins key-word argument, the histogram will have that number of bins.

7. Customizing histograms: setting bin boundaries

If we instead provide a sequence of values, these numbers will be set to be the boundaries between the bins, as shown here. There is one last thing to customize. Looking at this figure, you might wonder whether there are any rowing medalists with a height of less than 180 centimeters. This is hard to tell because the bars for the gymnastics histogram are occluding this information.

8. Customizing histograms: transparency

The occlusion can be eliminated by changing the type of histogram that is used. Instead of the "bar" type that is used per default, you can specify a histtype of "step", which displays the histogram as thin lines, instead of solid bars,

9. Histogram with a histtype of step

exposing that yes: there are rowers with a height of less than 180 centimeters.