1. Quantitative comparisons: scatter plots

Bar charts show us the values of one variable across different conditions, such as different countries. But what if you want to compare the values of different variables across observations? This is sometimes called a bi-variate comparison, because it involves the values of two different variables.

2. Introducing scatter plots

A standard visualization for bi-variate comparisons is a scatter plot. Let's look at an example. We'll use the climate change data that we have used previously. Recall that this dataset has a column with measurements of carbon dioxide and a column with concurrent measurements of the relative temperature. Because these measurements are paired up in this way, we can represent each measurement as a point, with the distance along the x-axis representing the measurement in one column and the height on the y-axis representing the measurement in the other column. To create this plot, we initialize a Figure and Axes objects and call the Axes scatter method. The first argument to this method will correspond to the distance along the x-axis and the second argument will correspond to the height along the y-axis. We also set the x-axis and y-axis labels, so that we can tell how to interpret the plot and call plt-dot-show to display the figure.

3. Customizing scatter plots

We can customize scatter plots in a manner that is similar to the customization that we introduced in other plots. For example, if we want to show two bivariate comparisons side-by-side, we want to make sure that they are visually distinct. Here, we are going to plot two scatter plots on the same axes. In one, we'll show the data from the nineteen-eighties and in the other, we'll show the data from the nineteen-nineties. We can select these parts of the data using the time-series indexing that you've seen before to create two DataFrames called eighties and nineties. Then, we add each one of these DataFrames into the Axes object. First, we add the data from the eighties. We add customization: we set the color of the points to be red and we label these data with the string "eighties". Then, we add the data from the nineties. These points will be blue and we label them with the string "nineties". We call the legend method to add a legend that will tell us which DataFrame is identified with which color, we add the axis labels and call plt-dot-show.

4. Encoding a comparison by color

This is what this figure looks like. You can see that the relationship between temperatures and carbon dioxide didn't change much during these years, but both levels of carbon dioxide and temperatures continued to rise in the nineties. Color can be used for a comparison, as we did here.

5. Encoding a third variable by color

But we can also use the color of the points to encode a third variable, providing additional information about the comparison. In the climate change data, we have a continuous variable denoting time stored in the DataFrame index. If we enter the index as input to the c key-word argument, this variable will get encoded as color. Note that this is not the color key-word argument that we used before, but is instead just the letter c. As before, we set the axis labels and call plt-dot-show.

6. Encoding time in color

Now, time of the measurements is encoded in the brightness of the color applied to the points, with dark blue points early on and later points in bright yellow.