1. Plotting time-series data

Many kinds of data are organized as time-series, and visualizations of time-series are an excellent tool to detect patterns in the data.

2. Time-series data

For example, the weather dataset that we used in the previous chapter is a relatively simple example of time-series data. Continuous variables, such as precipitation or temperatures are organized in our data table according to a time-variable, the months of the year. In this chapter, we'll dive deeper into using Matplotlib to visualize time-series data.

3. Climate change time-series

Let's look at a more complex dataset, that contains records of the change in climate in the last half a century or so. The data is in a CSV file with three columns. The "date" column indicates when the recording was made and is stored in the year-month-date format. A measurement was taken on the 6th day of every month from 1958 until 2016. The column "co2" contains measurements of the carbon dioxide in the atmosphere. The number shown in each row is parts-per-million of carbon dioxide. The column "relative-underscore-temp" denotes the temperature measured at this date, relative to a baseline which is the average temperature in the first ten years of measurements. If we want pandas to recognize that this is a time-series, we'll need to tell it to parse the "date" column as a date. To use the full power of pandas indexing facilities, we'll also designate the date column as our index by using the index-underscore-col key-word argument.

4. DateTimeIndex

This is the index of our DataFrame. It's a DateTimeIndex object with 706 entries, one for each measurement. It has a DateTime datatype and Matplotlib will recognize that this is a variable that represents time. This will be important in a little bit.

5. Time-series data

The other two columns in the data are stored as regular columns of the DataFrame with a floating point data-type, which will allow us to calculate on them as continuous variables. There are a few points in the CO2 data that are stored as NaNs or Not-a-Number. These are missing values where measurements were not taken.

6. Plotting time-series data

To start plotting the data, we import Matplotlib and create a Figure and Axes. Next, we add the data to the plot. We add the index of our DataFrame for the x-axis and the "co2" column for the y-axis. We also label the x- and y-axes. Matplotlib automatically chooses to show the time on the x-axis as years, with intervals of 10 years. The data visualization tells a clear story: there are some small seasonal fluctuations in the amount of CO2 measured, and an overall increase in the amount of CO2 in the atmosphere from about 320 parts per million to about 400 parts per million.

7. Zooming in on a decade

We can select a decade of the data by slicing into the DataFrame with two strings that delimit the start date and end date of the period that we are interested in. When we do that, we get the plot of a part of the time-series encompassing only ten years worth of data. Matplotlib also now knows to label the x-axis ticks with years, with an interval of one year between ticks. Looking at this data, you'll also notice that the missing values in this time series are represented as breaks in the line plotted by Matplotlib.

8. Zooming in on one year

Zooming in even more, we can select the data from one year. Now the x-axis automatically denotes the months within that year.