**Area Plots**

After watching this video, you'll be able to: describe what is an area plot, explain how to create an area plot using Matplotlib. What is an area plot? An area plot, also known as an area chart or graph, displays the magnitude and proportion of multiple variables over a continuous axis, typically representing time or another ordered dimension. It's similar to a line plot, but with the area below the line filled with color to emphasize the cumulative magnitude of the variables. This kind of graph is commonly used when trying to compare two or more quantities. Let's learn how to generate an area plot with Matplotlib. Before we go over the code on how to generate an area plot, let's do a quick recap of our dataset. Recall that each row represents a country and contains metadata about the country, such as it's geographic location and its development status. Each row also contains numerical data of annual immigration from that country to Canada from 1980 - 2013. Now let's process the dataframe so that the country name becomes the index of each row. This should make retrieving rows pertaining to specific countries a lot easier. Also, let's add an extra column that represents the cumulative sum of annual immigration from each country from 1980 - 2013. For Afghanistan, it's 58,639 total. For Albania, it's 15,699, and so on. Let's name our dataframe df\_canada. Now that we know how we have stored our data in the dataframe df\_canada. Let's try to generate area plots for the countries with the highest immigration to Canada. We can try to find these countries by sorting our dataframe in descending order of cumulative total immigration 1980-2013. We use the sort\_values function to sort our dataframe in descending order. Here is the result. It turns out that India followed by China, then the United Kingdom, the Philippines, and Pakistan, are the top five countries with the highest immigration to Canada. Can we now go ahead and generate the area plots using the first five rows of this dataframe? Not quite yet. First, we need to create a new dataframe of these five countries only and exclude the total column. More importantly, to generate the area plots for these countries, we should plot the years on the horizontal axis and the annual immigration on the vertical axis. Note that Matplotlib plots the indices of a dataframe on the horizontal axis and with the dataframe as shown, Matplotlib plots the countries on the horizontal axis. To fix this, we need to take the transpose of the dataframe. Let's see how we can do this. After we sort our dataframe in descending order of cumulative annual immigration, we create a new dataframe of the top five countries, and we call it ‘df\_top5’. We then select only the columns representing the years 1980 - 2013 in order to exclude the total column before applying the transpose method. The resulting dataframe is exactly what we want with five columns, where each column represents one of the top five countries and the years being the indices. Now we can go ahead and use the plot function on dataframe, df\_top five to generate the area plots. First, we import Matplotlib as MPL and it's scripting interface as PLT. Then we call the plot function on the dataframe df\_top5, and we set kind=area to generate an area plot. Then to complete the figure, we give it a title and label both the axes appropriately. Finally, we then use the show function to display the figure. Note that here we are generating the area plot using the inline backend, and there you have it. An area plot that depicts the immigration trend of the five countries with the highest immigration to Canada, 1980-2013. Area plots are particularly effective and depicting data with a cumulative nature, such as tracking stock market performance, visualizing population demographics, or displaying the distribution of resources across various sectors. In this video, you learned that: an area plot depicts cumulated totals using numbers or percentages over time. The process of creating an area plot involves importing Matplotlib and calling the plot function on the dataframe with kind parameter assigned as area. Area plots provide a visually appealing and intuitive way to showcase the relationship and proportion of multiple variables in a single chart.

**Histograms**

After watching this video, you'll be able to: define a histogram with the help of an illustration, explore the process of creating a histogram using Matplotlib, let's start by defining what a histogram is. A histogram is a way of representing the frequency distribution of a numeric data set. The way it works is that it partitions the spread of the numeric data into bins, assigns each data point in the data set to a bin, and then counts the number of data points assigned to each bin. So, the vertical axis is essentially the frequency or the number of data points in each bin. For example, let's say the range of the numeric values in the data set is 34,129. Now, the first step in creating a histogram is partitioning the horizontal axis in, say, 10 bins of equal width. Then we construct the histogram by counting how many data points have a value that is between the limits of the first bin, the second bin, the third bin, and so on. Say, the number of data points that have a value between 0 and 3,413 is 175, then we draw a bar of that height for this bin, we repeat the same step for all the other bins. And if no data points fall into a bin, then that bin would have a bar of height zero. So how do we create a histogram using Matplotlib? Let's process the data frame so that the country name becomes the index of each row, this should make retrieving rows pertaining to specific countries a lot easier. Also, let's add an extra column that represents the cumulative sum of annual immigration from each country from 1980 to 2013, so for Afghanistan it's 58,639 total, and for Albania it's 15,699, and so on. And let's name our data frame df\_canada. Considering the Canada immigration data set, having countries as the index, and having another column as total represents the cumulative sum of annual immigration from each country from 1980 to 2013. Say we want to visualize the distribution of immigrants to Canada in the year 2013, the simplest way to do that is to generate a histogram of the data in column 2013. Let's see how we can do that with Matplotlib, first, we import Matplotlib as mpl and its scripting interface as plt. Then we call the plot function on the data in column 2013 and we specify kind=hist to generate a histogram, then, to complete the figure, we give it a title and label both its axes. Finally, we use the show function to display the figure, and there you have it: a histogram that depicts the distribution of immigration to Canada in 2013. But notice how the bins are not aligned with the tick marks on the horizontal axis, this can make the histogram hard to read. So, let's try to fix this in order to make our histogram more effective. One way to solve this issue is to borrow the histogram function from the NumPy library, so as usual, we start by importing Matplotlib and its scripting interface, but this time we also import the NumPy library, then we call the NumPy histogram function on the data in column 2013. This function is going to partition the spread of the data in column 2013 in ten bins of equal width, where ten is the default number of bins. It also computes the number of data points that fall in each bin and then return this frequency of each bin which we are calling ‘count’ here, and the bin edges which we will call ‘bin\_ edges’. We then pass these bin edges as an additional parameter in our plot function to generate the histogram, and there you go. A precisely generated histogram with the bin edges and tick marks clearly aligned on the horizontal axis. In this video you learned that: a histogram is a way of representing the frequency distribution of a numeric data set. To generate a histogram on Matplotlib, you import Matplotlib as mpl and its scripting interface is plt. You can call the plot function on the data frame with kind parameter assigned as hist. You can use the NumPy library to create bins for the histogram representation.

**Bar Charts**

After watching this video, you'll be able to: describe a bar chart with the help of an illustration, explore the process of creating a bar chart using Matplotlib. A bar chart is a popular visualization tool. Unlike a histogram, a bar chart, also known as a bar graph, is a type of plot where the length of each bar is proportional to the value of the item that it represents. It's commonly used to compare the values of a variable at a given point in time. For example, say we want to visualize in a discrete fashion, how immigration from Iceland to Canada looked 1980-2013. One to do that is by building a bar chart where the height of the bar represents the total immigration from Iceland to Canada in a particular year. How do we do that with Matplotlib? From our dataset on immigration to Canada, we created a dataframe called df\_canada. Having country names as the index of each row and a column total represents the cumulative sum of annual immigration from each country 1980-2013. Let's see how we can use Matplotlib to generate a bar chart to visualize what immigration from Iceland to Canada looked like 1980-2013. As usual, we start by importing Matplotlib and it's scripting interface. Then we use the years variable to create a new dataframe. Let's name it df\_iceland, which includes the data pertaining to annual immigration from Iceland to Canada, and excluding the total column. Then we use the plot function on df\_iceland, and we set kind=bar to generate a bar chart. To complete the figure, we give it a title and label both of its axes. Finally, we use the show function to display the figure. There you have it. A bar chart depicts immigration from Iceland to Canada 1980-2013. By examining the bar chart, we noticed that immigration to Canada from Iceland has seen an increasing trend since 2010. I'm sure the curious among you are already wondering who the culprit behind this increasing trend is. You can also create a bar chart with horizontal bars by assigning bar to the kind parameter of the plot function. Note the use of the color parameter as you can change the color of the bar with this. Let's suppose you want to highlight the years with highest and lowest number of Icelandic immigrants to Canada between the year 1980 to 1990. You can pass a list of colors to the color parameter accordingly. Here we have highlighted the bars for the years 1981 and 1990 with the color red. By assigning the color of your choice to the edge color parameter, you can change the borderline color of each bar. In this video, you learned that: a bar chart is a type of plot where the length of each bar is proportional to the value of the item that it represents. You can create a bar chart using Matplotlib representing the total immigration from Iceland to Canada.

**Pie Charts**

After watching this video, you'll be able to describe a pie chart with the help of an example. Explore the process of creating a pie chart using Matplotlib. So what is a pie chart? A pie chart is a circular statistical graphic divided into segments to illustrate numerical proportion. For example, here is a pie chart of the Canadian federal election. It represents the partywise percentage of seats won in the House of Commons. Next, let's learn how to create a pie chart with Matplotlib. Now, let's try to visualize the continentwise breakdown of immigration to Canada from our data set df\_canada. The first step is to group the data by continent using the continent column, and we use Pandas for this. We call the Pandas group by function on df\_canada, and we sum the number of immigrants from the countries that belong to the same continent. Here is the resulting data frame, and let's name it df\_continents. The resulting data frame has six rows, each representing a continent, and 35 columns representing the years from 1980 to 2013, plus the cumulative sum of immigration for each continent. Now, we're ready to start creating our pie chart. We start as usual by importing Matplotlib as mpl and its scripting layer, the pyplot interface, as plt. Then we call the plot function on total column of the data frame df\_continents, and we set kind = pie to generate a pie chart. Then to complete the figure, we give it a title. Finally, we use the show function to display the figure. And there you have it, a pie chart that depicts each continent's proportion of immigration to Canada from 1980 to 2013. The explode property of a pie chart in Matplotlib allows you to offset one or more slices from the center, highlighting specific sections of the chart. By assigning values to the explode parameter, you can control the degree of separation and emphasize particular segments of the pie. As shown in this pie, the continents where the total is less than 10% are exploded out to be highlighted. A final point about pie charts, there are some strong critics who oppose using pie charts in any condition. They argue that pie charts do not display accurate data consistently. When it comes to depicting data consistently and communicating the point, bar charts perform significantly better. If you are interested in learning about the arguments against pie charts, here's a link to a very interesting article that discusses very clearly the flaws of pie charts. You can also find the link under the video. In this video, you learned that a pie chart is a circular statistical graphic divided into segments to illustrate numerical proportion. The process of creating a pie chart involves involves importing Matplotlib to represent a large set of data over a period of time.

**Box Plots**

After watching this video, you'll be able to describe a box plot with the help of an illustration. Explain how to create box plots using Matplotlib, so what is a box plot? A box plot is a way of statistically representing the distribution of given data through five primary dimensions, minimum is the smallest number in the sorted data. First quartile is the point 25% of the way through the sorted data, in other words, a quarter of the data points are less than this value. Median is the median of the sorted data, third quartile is the point 75% of the way through the sorted data. In other words, three quarters of the data points are less than this value, and maximum is the highest number in the sorted data. Let's see how we can create a box plot with Matplotlib. We first process the data frame df\_canada to set the country name as the index and add a column representing the cumulative sum of annual immigration from each country from 1980 to 2013. We want to create a box plot to visualize immigration from Japan to Canada. As with the other tools we've learned, we start by importing matplotlib as mpl and the pyplot interface as plt. Then we create a new data frame on the data about Japan and we exclude the total column using the years variable. Then we transpose the resulting data frame in the correct format to create the box plot. Let's name this new data frame df\_japan. Following that, we call the plot function on df\_japan and set a kind=box to generate a box plot, then we complete the figure. We give it a title and label the vertical axis appropriately. Finally, we use the show function to display the figure, and there you have it. A box plot that provides a good distribution of Japanese immigration to Canada from 1980 to 2013. From this plot, we can verify that there are no outliers in this data. Also, we can see that the median is closer to the top, indicating more data concentration in the upper half. In this video, you learned that a box plot is a way of statistically representing given data distribution through five main dimensions. The five main dimensions are minimum, first quartile, median, third quartile and maximum. You can create a box plot using Matplotlib.

**Scatter Plots**

After watching this video, you'll be able to: describe what is a scatter plot with the help of an example, explore the scatter plot creation process using Matplotlib. What is a scatter plot? A scatter plot is a type of plot that displays values pertaining to typically two variables against each other. Usually, it's a dependent variable that is plotted against an independent variable to determine if any correlation between the two variables exist. For example, here's a scatter plot of income versus education. And by looking at the plotted data, one can conclude that an individual with more years of education is likely to earn a higher income than an individual with fewer years of education. How can we create a scatter plot with matplotlib? We are considering dataframe df\_canada, which has country names set as an index and one column is total, which represents the cumulative sum of annual immigration from each country from 1980-2013. Let's say we want to create a scatter plot of the total annual immigration to Canada from 1980-2013. To be able to do that, we first need to create a new dataframe that shows each year and the corresponding total number of immigrants from all countries worldwide as shown here. Let's name this new dataframe as df\_total. Then we proceed as usual, we import matplotlib as mpl and its scripting layer, the pyplot interface as plt. We use the plot function on the dataframe df\_total, and we set kind=scatter to generate a scatter plot. Unlike the other data visualization tools, we're only passing in the kind parameter was enough to generate the plot. With scatter plots, we also need to pass the variable, which is on the horizontal axis as the x parameter and the variable that is on the vertical axis as the y parameter. In this case, we're passing column year as the x parameter and column total as the y parameter. Then to complete the figure, we give it a title and label its axes appropriately. Finally, we use the show function to display the figure. There you have it. A scatter plot that shows total immigration to Canada from countries all over the world from 1980-2013. The scatter plot clearly depicts an overall rising trend of immigration with time. Consider the use of the color parameter, which we have assigned a value of dark blue. You may like to pick a color of your choice from the available color palettes. For instance, see how the plot is in the color red. You can also use the s parameter to represent any third variable if you want to. Here we have included the total from the African continent to represent the size of each marker over the years. And it is evident that the number of immigrants has increased over these years as the size of the markers in the scatter plot is increasing. In this video, you learned That: a scatter plot displays values pertaining to typically two variables against each other. The process of creating a scatter plot involves importing matplotlib to visualize a large set of data.

**Plotting Directly with Matplotlib**

After watching this video, you'll be able to explore various functions offered by Matplotlib for data visualization and plotting. Differentiate between data storytelling and data visualization. Matplotlib is a general purpose, comprehensive plotting library that provides a flexible interface for creating a wide range of plots. It's pyplot module offers a convenient way to create and customize plots quickly. Let's start plotting directly with matplotlib. First, we need to import the library. Import matplotlib.pyplot as PLT. Here we're importing pyplot as PLT. We have also imported NumPy as np as numpy arrays are usually used as the data source for plotting and also support mathematical functions. Next, for tabular data, import pandas. Then call the subplot function and create a figure, the Canvas window and the axes. It's the area where the plot appears. The figure axes pair provides greater control over the figure or Canvas. Now name the plot that you want to create on these axes. Let's now call plot function on the axes to display the line plot, we'll create some synthetic data to generate the plot using NumPy years is equal to np.arange and passing 1980. 2014 to a range function will generate a 1D array of numbers 1980-2013 excluding 2014 immigrants equals np.random.randint will generate a 1D array of random integers, between 2,000-10,000 range. Size equals 34, specifies the number of elements in the array. X.plot function will create a line plot. It takes the x and y values as arguments corresponding to the data to be considered for the x-axis and the y-axis. X.plot years immigrants. Finally display the plot using plt.show function and if you want to display this data as a scatter plot called the scatter function on x instead as shown here. X.scatter and past years and immigrants to it. Now let's put a title to the plots. To do so, simply pass the title string to the plt.title function. plt.title immigrants between 1980 to 2013. On both the plots, label the axis with plt.xlabel as years and plt.y label as total immigrants on both axes. You can apply the label entitled directly to the axes as shown here. Axes.set\_title and pass the title string. Axes.set\_ x-label or y-label. See now the plots have the title and the labels. Now using x-lim and y-lim functions, you can set the limits on the x and y-axis. To improve the readability, you can enable grid lines with grid function and pass it with true, like x.grid, true. Legend function will include the legend to your plot. Notice that the x-axis now starts with 1975 and ends with 2015, and now it shows the grid at the background of the plot. There are various options available to customize the plot. You can customize the plot with line styles, marker styles, and color customization options to achieve the desired visual effects. With marker, select a style to represent data points in your plot, like S for square and 0 for dots. For line plot, the marker is marker size, while in scatter it's just S. Similarly, you select a color and a size for it. With the line style parameter, you can select different line styles, such as solid, dashed, or dotted lines. Notice the use of LOC in the legend parameter to specify the location where the legend is placed on the figure. Let's now use a bar plot to represent the number of immigrants for each year. Ax.bar function creates a bar plot, passing years corresponding to the x-axis and the immigrants to the y-axis. The height of each bar represents the number of immigrants for that year. Ax.hist generates a histogram with bins set to 20. Notice the use of edge color and color parameters to specify the border and bar column of the histogram. Ax.pie will generate a pie on the axis. Here we're plotting a pie for only five years, 1980 to 1984. Colors you are assigned to each pie slice has a list of colors and labels are set is years. Auto PCT displays the percentage of immigrants with one decimal point. Now, let's explore how to display more than one plot on the same figure and specify the number of rows and columns to be created to the subplots function. For instance, let's create a line and scatter plot in one row, plt.subplots and pass one to it. Both the subplot will be sharing the same y-axis as the data in the y-axis is the same, so assign the Sheri parameter as true. Now you have two axes, axs with index zero and axs with index one, and you can plot your plots as you did earlier. Axs0.plot function for the line axs1.scatter for a scatter plot. And see you have created two plots together. Alternatively, we can use the add underscore subplot function. It takes three arguments, first, the number of rows, second, columns and the last is an index of the subplot. You can create different axes on the figure and add subplots as shown here. Ax 1 equals figure.add\_subplot 2, 2, 1. This one means the first axes on the two-by-two divided figure. Then on these axes, create the plot like an ax1.plot. Similarly, you can plot all the plots you want on the four axes. Like this. You have all your plots on this figure. Great work. Lastly, let's learn about data storytelling and data visualization. These are two different terms and serve different purposes. Data storytelling is the art of storytelling that involves creating a narrative around the data. It presents a compelling and engaging story. On the other hand, data visualization is an important aspect of data storytelling and evolves, creating informative charts to understand and explore patterns, trends, and relationships within the data. It brings data to life. We recommend reading this article titled data storytelling, the essential data science skill everyone needs on www.forbes.com for more on data storytelling. In this video, you learned that matplotlib is a versatile plotting library that offers a flexible interface for creating various types of plots. Matplotlib pyplot module offers a convenient way to create and customize plots quickly. Data storytelling is the art of storytelling that involves creating a narrative around the data. Data visualization is an important aspect of data storytelling and involves creating, engaging visuals.