Data Analysis – Analyzing and Visualizing

Learning Goals

- Perform basic analyses on data, including calculating statistics and probabilities, to answer simple questions

- Choose an appropriate visualization to create based on the number of dimensions and data types

- Create simple matplotlib visualizations that show the state of a dataset using APIs and examples

Analysis

Basic Data Analyses – Statistics Library

There are many basic analyses we can run on features in data to get a sense of what the data means. You've learned about some of them already in math or statistics classes, such as mean, median, and mode.

You can implement these in Python yourself, but you don't have to! There's already a statistics library that does this for you.

```
import statistics

data = [41, 65, 64, 50, 45, 13, 29, 14, 7, 14]

statistics.mean(data) # 34.2
statistics.median(data) # 35.0
statistics.mode(data) # 14
```

More Analysis Methods

There's plenty of other data analysis methods we could cover – bucketing, detecting outliers, dealing with missing data – but what kind of method you need will depend entirely on the context of the problem you're solving.

Sometimes we may want to investigate a dataset more broadly. For example, how many times does each individual flavor occur in any of a person's preferences?

Visualization

Exploration vs. Presentation

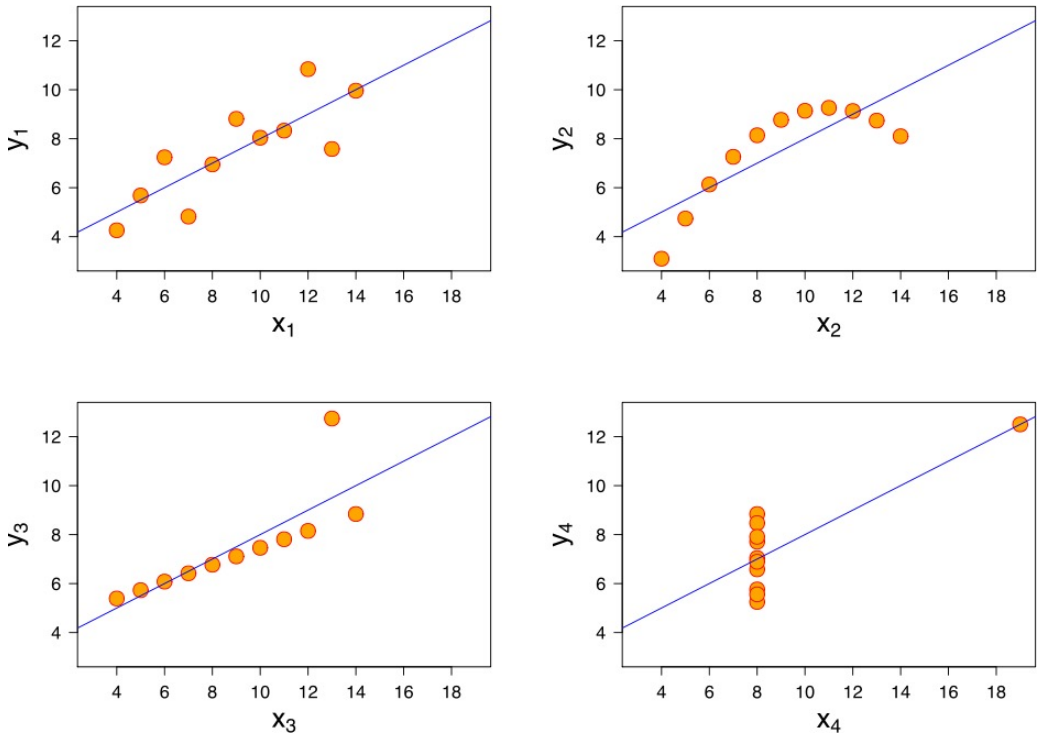
Data Visualization is the process of taking a set of data and representing it in a visual format. Whenever you've made charts or graphs in past math or science classes, you've visualized data!

Visualization is used for two primary purposes: exploration and presentation.

Data Exploration

In data exploration, charts created from data can provide information about that data beyond what is found in simple analyses alone.

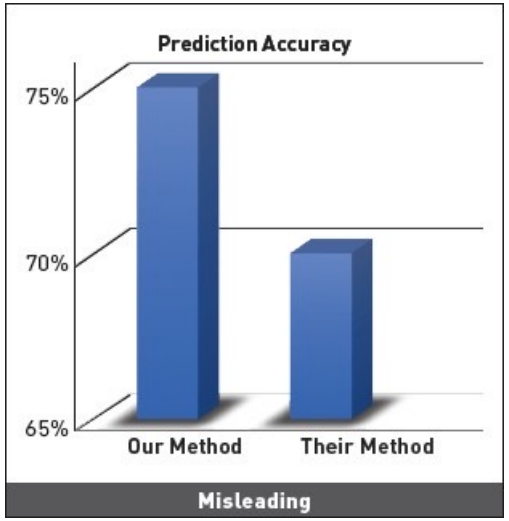
For example, the four graphs to the right all have the same mean and the same best-fit linear regression. But they tell very different stories.

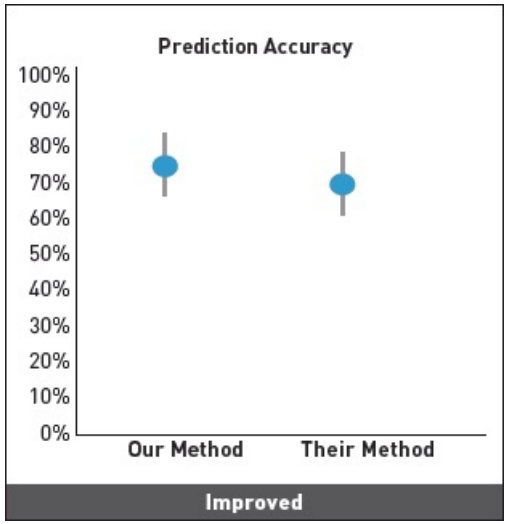


Data Presentation

In data presentation, you've already found an interesting pattern in the data and you need to make that pattern easily visible to other people.

In order to choose the best visualization for the job, consider the type of the data you're presenting (categorical, ordinal, or numerical), and how many dimensions of data you need to visualize.





https://interactions.acm.org/archive/view/july-august-2018/the-good-the-bad-and-the-biased

One-Dimensional Data

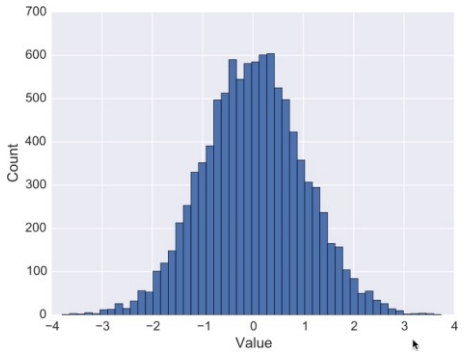
A one-dimensional visualization only visualizes a single feature of the dataset. For example:

"I want to know how many of each product type are in my data"

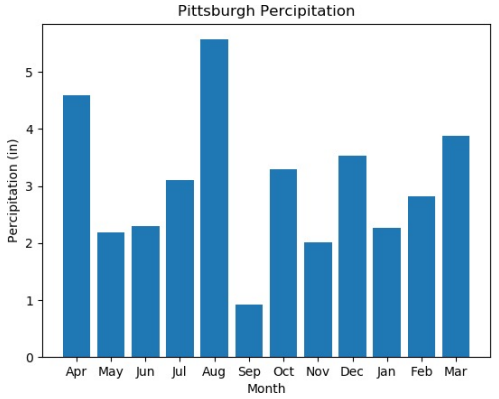
"I want to know the proportion of people who have cats in my data"

Charts for One-Dimensional Data

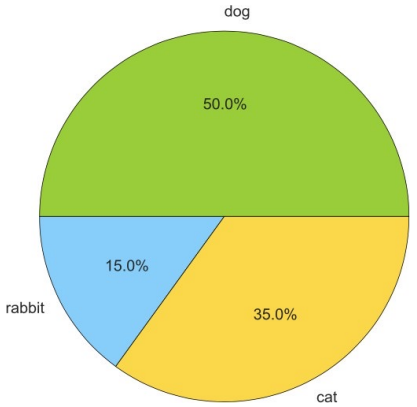
To visualize numerical data, use a histogram.



To visualize ordinal data, use a bar chart.



To visualize categorical data, use a pie chart.



Two-Dimensional Data

A two-dimensional visualization shows how two features in the dataset relate to each other. For example:

"I want to know the cost of each product category that we have"

"I want to know the weight of the animals that people own, by pet species"

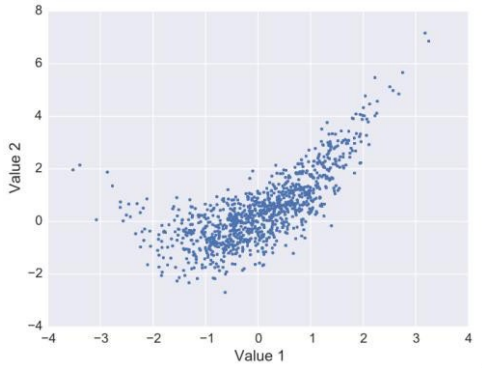
"I want to know how the size of the product affects the cost of shipping"

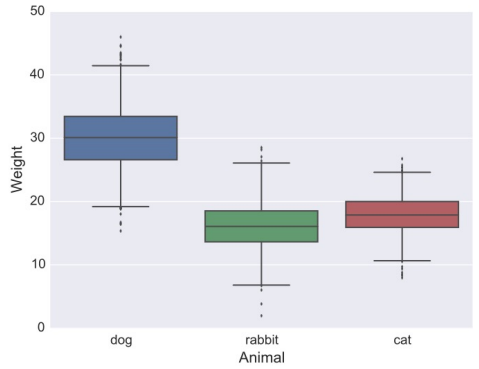
Charts for Two-Dimensional Data

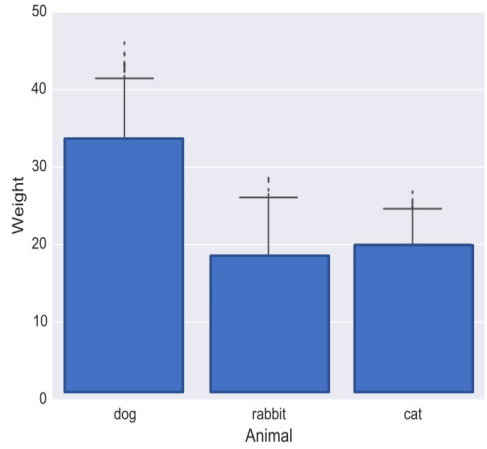
To analyze numerical x numerical data, use a scatter plot.

To analyze numerical x ordinal/categorical data, use a bar chart for averages or a box-and-whiskers plot for ranges.

It is difficult to analyze ordinal/categorical x ordinal/categorical data visually; use a table instead.







Three-Dimensional Data

A three-dimensional visualization tries to show the relationship between three different features at the same time. For example:

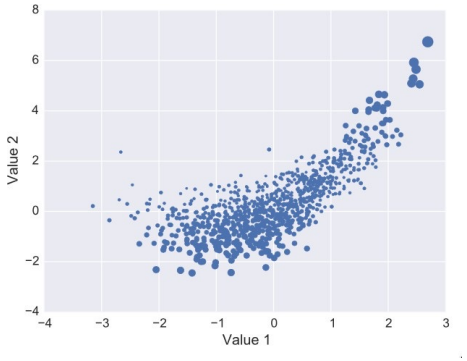
"I want to know the cost and the development time by product category"

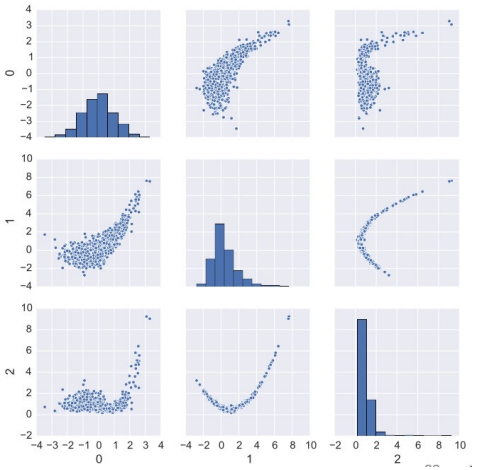
"I want to know the weight of the animals that people own and how much they cost, by pet species"

"I want to know how the size of the product and the manufacturing location affects the cost of shipping"

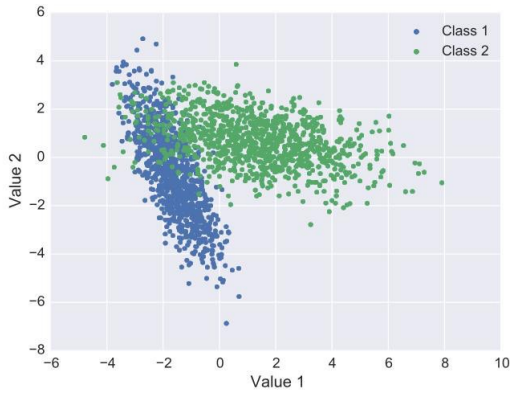
Charts for Three-Dimensional Data

To analyze numerical x numerical x numerical data, use a bubble plot to compare all three or a scatter plot matrix to compare all the pairs.





To analyze numerical x numerical x ordinal/categorical data, use a colored scatter plot.



Activity: Pick a Visualization

You do: for each of the problem prompts, determine the number of dimensions, then pick the best visualization to use based on the data types.

- graph the % of people who have gotten COVID vs. the % of people who have been vaccinated, separated by state
- graph the distribution of grades in a class
- graph the ages of pets at a shelter compared to the species of pets

Coding Visualizations with Matplotlib

Matplotlib Makes Visualizations

The matplotlib library can be used to generate interesting visualizations in Python.

Unlike the previous libraries we've discussed, matplotlib is external – you need to install it on your machine to run it. Use the `pip` command to do this.

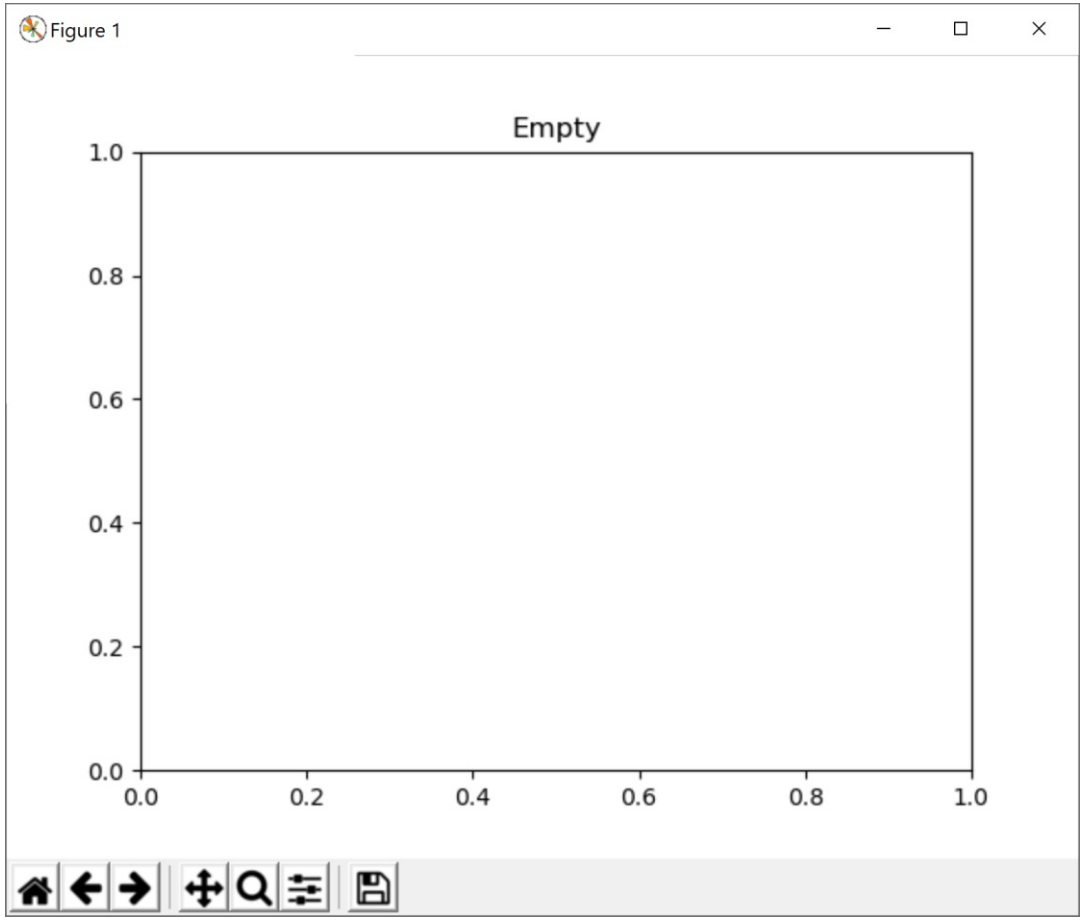
```
pip install matplotlib
```

Draw Visualizations on the Plot

Matplotlib visualizations can be broken down into several components. We'll mainly care about one: the plot (called `plt`). This is like Tkinter's canvas, except that we'll draw visualizations on it instead of shapes.

We can construct an (almost) empty plot with the following code. Note that matplotlib comes with built-in buttons that let you zoom, move data around, and save images.

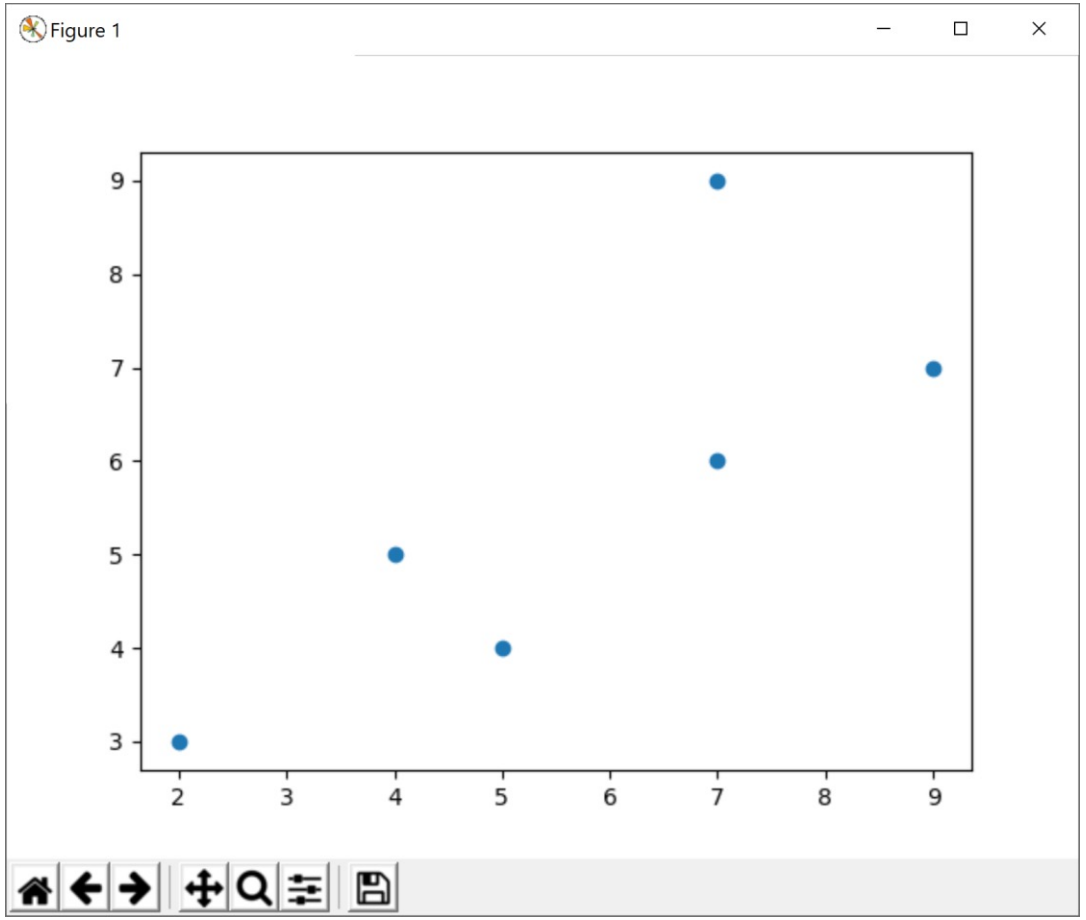
```
import matplotlib.pyplot as plt

plt.title("Empty")
plt.show()
```



Add Visualizations with Methods

There are lots of built-in methods that let you construct different types of visualizations. For example, to make a scatterplot use `plt.scatter(xValues, yValues)`.

```
x = [2, 4, 5, 7, 7, 9]
y = [3, 5, 4, 6, 9, 7]
plt.scatter(x, y)
plt.show()
```

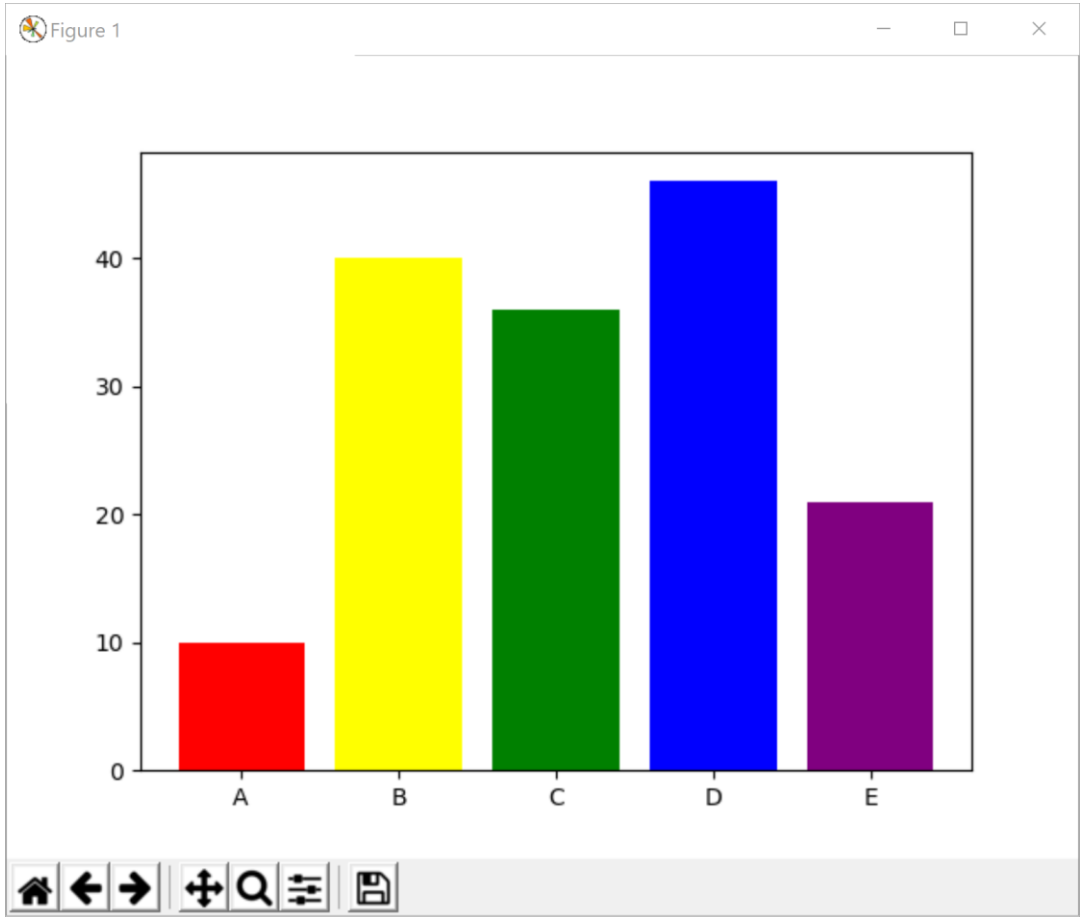


Visualization Methods have Keyword Args

You can customize how a visualization looks by adding keyword arguments. We used these in Tkinter to optionally change a shape's color or outline; in Matplotlib we can use them to add labels, error bars, and more.

For example, we might want to create a bar chart (with `plt.bar`) with a unique color for each bar. Use the keyword argument `color` to set the colors.

```
labels = [ "A", "B", "C", "D", "E" ]
yValues = [ 10, 40, 36, 46, 21 ]
colors = [ "red", "yellow", "green",
           "blue", "purple" ]
plt.bar(labels, yValues, color=colors)
plt.show()
```



Don't Memorize – Use the Website!

There are a ton of visualizations you can draw in Matplotlib, and hundreds of ways to customize them. It isn't productive to try to memorize all of them.

Instead, use the documentation! Matplotlib's website is very well organized and has tons of great examples: https://matplotlib.org/

When you want to create a visualization, start by searching the API and the pre-built examples to find which methods might do what you need.

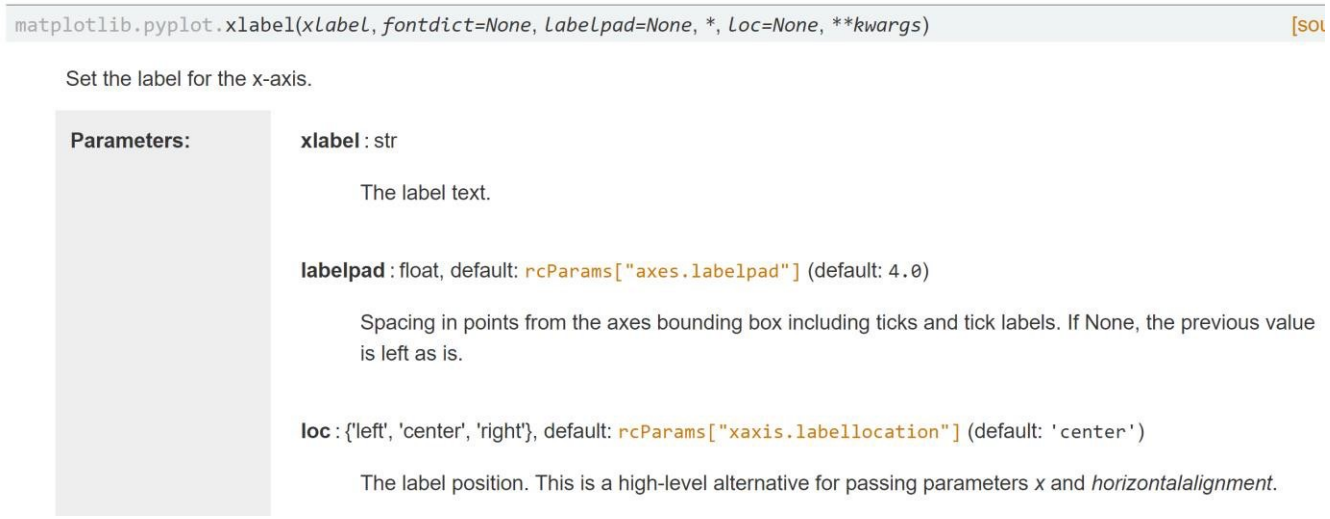
API Example

For example – how can we add x-axis and y-axis labels to the bar chart?

Go to the plot API: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html

Search 'label' and you'll soon find the functions `xlabel` and `ylabel`. You can click on the function to find more information. The page describes what the function does, what the required arguments are, and what it returns.

Note that the keyword arguments will be listed with default values. That's how we know they're optional.



`matplotlib.pyplot.xlabel(xlabel, fontdict=None, labelpad=None, *, loc=None, **kwargs)` [sou

Set the label for the x-axis.

Parameters:

xlabel : str

The label text.

labelpad : float, default: `rcParams["axes.labelpad"]` (default: `4.0`)

Spacing in points from the axes bounding box including ticks and tick labels. If None, the previous value is left as is.

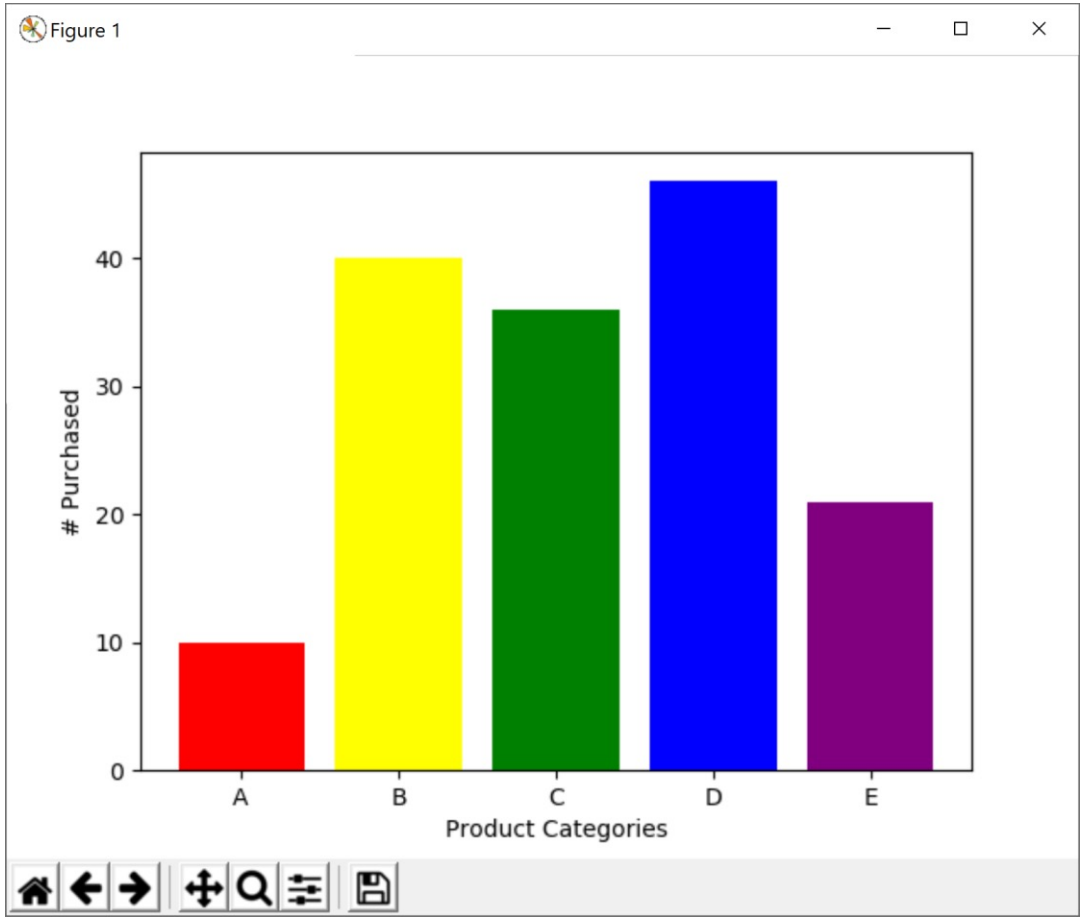
loc : {'left', 'center', 'right'}, default: `rcParams["xaxis.labellocation"]` (default: `'center'`)

The label position. This is a high-level alternative for passing parameters *x* and *horizontalalignment*.

If nothing obvious shows up, you can do a broader internet search of 'matplotlib x-axis label', which will often point you to the right place.

Adding xlabel and ylabel

```
labels = [ "A", "B", "C", "D", "E" ]
yValues = [ 10, 40, 36, 46, 21 ]
colors = [ "red", "yellow", "green",
           "blue", "purple" ]
plt.bar(labels, yValues, color=colors)

plt.xlabel("Product Categories")
plt.ylabel("# Purchased")

plt.show()
```



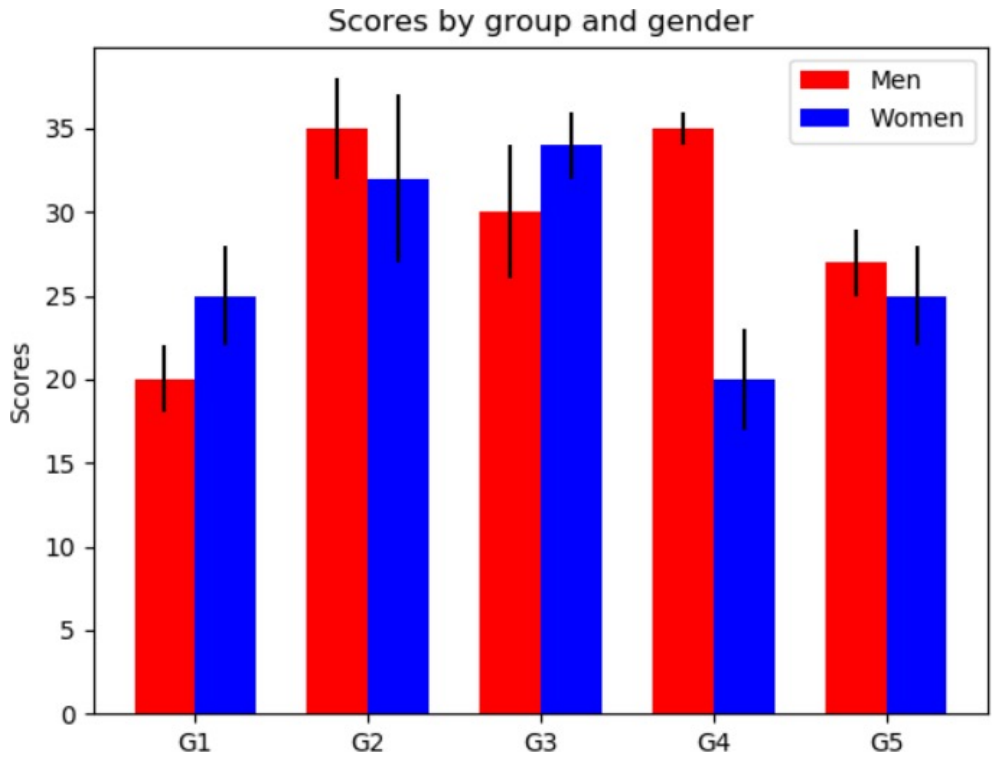
Example

Alternatively, you can browse the Matplotlib examples page to find visualizations and features that might prove useful:

https://matplotlib.org/stable/gallery/index.html

Perhaps we're interested in using grouped bar charts to show the breakdown between products purchased in different countries. The example code provides a starting place for which functions to use.

Try copying the example code into your editor and running it. Then try changing some things to see how the results are affected.



Plot vs Axis

You might have noticed that the grouped bar chart example looks slightly different than the code we've written so far. It sets

```
fig, ax = plt.subplots()
```

and calls methods on `ax` for the rest of the code.

This is an alternate way to write code in Matplotlib. Instead of drawing on the plot, break the plot into one of more axes with `plt.subplots`, then draw directly on the axis.

This is mainly useful if you want to draw more than one visualization in a single window. For the visualizations we'll do in this class, `plt` will work fine.