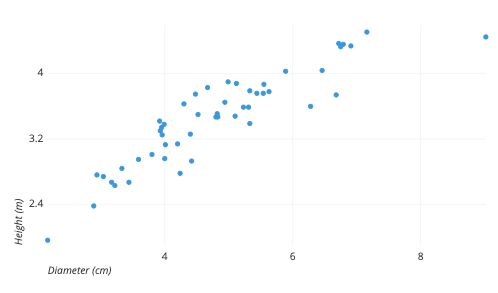
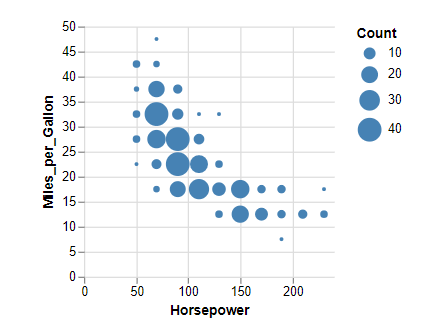
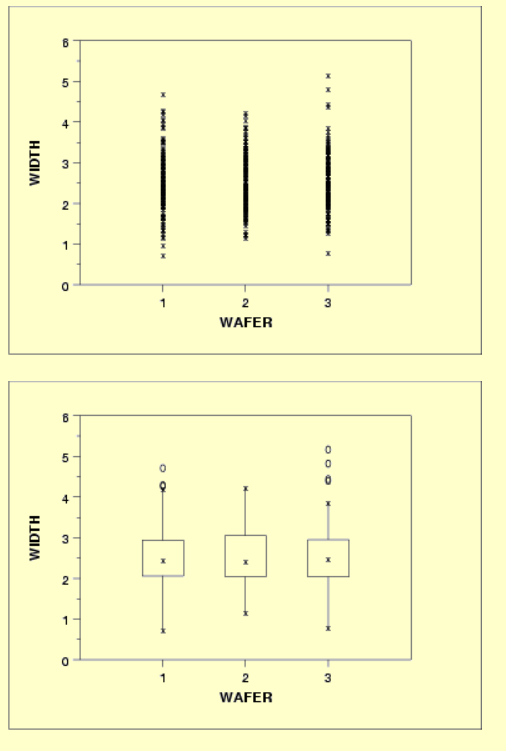
We now know how to visualize the relationship between two variables and to determine if there is one in the first place. There are many tools currently used for visualizing data such as bar/column charts, line charts, pie charts, etc. Each of these has their own specific use case; however, not all of them can be used to visualize *relationships* between data. For example, a histogram gives us information about the frequency distribution of one variable, but not its relationship to other variables. For this, we will need visualizations such as scatter plots, box plots, and correlation matrices, to name a few. I will be describing each of these plots, as well as what binning is and some Python packages that can be used to visualize data relationships.

By far the simplest and the most common method used to visualize data is the scatter plot. This plot will assign each data point from an array of variables to a point on one axis of an x-y coordinate plane based on the user’s axis assignment. When a variable is placed on each axis of a Cartesian plane, we will see the points begin to form a line if they are linearly correlated or another shape if they have a non-linear correlation. Below is a scatterplot showing a strong linear correlation. If our scatter plot has very many data points that a correlation becomes difficult to determine, we can use what is calling *binning,* where data points are grouped into bins with each bin containing a different density of points, similar to histogram binning. Below our scatter plot example is a binned scatter plot showing point density of vehicle mileage versus power. At first glance, the relationship would have appeared linear, but the usage of bins helps us to deduce that it is, in fact, inverse exponential.

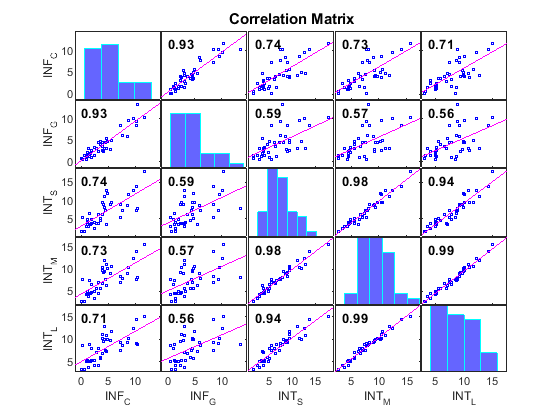




If the points on a scatter plot are arranged into “columns” such that one x value could have several y values, then box plots are more appropriate for visualizing a relationship. True to its name, the box plot will display a box corresponding to the median and quartiles of the distribution, as well as “whiskers” showing the minimum and maximum values of the distribution and any outliers. With box plots, one can easily several variables at once compared to scatter plots which can only visualize two variables at once. While statistical comparison might be easier with a box plot, it is often unclear whether a relationship is linear of nonlinear. Below is an example of a box plot showing how a silicon wafer’s width varies between 3 construction methods. In this case, construction method has a negligible effect on wafer width.



A solution to scatter plots’ graphing of only two variables is to use correlation matrices. Their output is a matrix containing scatter plots for every possible combination of two variables as well as either the Pearson or Spearman correlation coefficient of each plot, depending on the user specification. Since the correlation coefficient of a variable with itself will always be 1, the matrix replaces those scatter plots with histograms of said variable. Correlation matrices allow us to find the correlation coefficients (and thus R-squared) of a very large number of relationships at once. Below is an example of a correlation matrix for inflation rates versus interest rates over different time periods.



There are many Python packages with which we can plot data relationships, but I will focus on some of the most commonly used libraries.

**Matplotlib:** The most widely used Python package for plotting data in general, despite being over a decade old. This package was based on MATLAB, a programming language from the 1980s also built for visualization. Some libraries, such as pandas, integrate with it, allowing a user to access some of matplotlib with less code. While it may be versatile, matplotlib has been criticized for its default style graphic presentation and for lacking the ability to build publication quality charts. Most likely this is due to the age of the package.

**Seaborn:** Seaborn focuses on aesthetics and plot customization, making visually pleasing charts in few lines of code. It is one of the packages built upon matplotlib. Seaborn is dataset-oriented, containing functions that operate on data frames and arrays. As such, it includes support for the numpy and pandas libraries.

**ggplot:** Based on the R package ggplot2, ggplot works differently than matplotlib in that it constructs plots in layers. For example, ggplot functions can be used to build a scatter plot by first constructing axes, then layering on the points, then a trendline. Because it is designed to integrate with pandas, ggplot best reads data as a data frame. It is quite the opposite of seaborn, sacrificing aesthetics for simplicity of plot construction.

**Plotly:** Although Plotly is a web-based toolkit, it can be accessed as a package from Python notebooks. It has many of the same functions as matplotlib.

All the aforementioned packages can construct the same plots. Although each package is designed for a specific purpose in plotting data relationships, the Python package one uses is entirely dependent on user preference.

References:

https://chartio.com/learn/charts/what-is-a-scatter-plot/.

https://vega.github.io/vega/examples/binned-scatter-plot/.

https://www.itl.nist.gov/div898/handbook/pmc/section6/pmc612.htm#:~:text=The%20run%20sequence%20plot%20is,size%20to%20show%20greater%20detail.&text=For%20comparison%2C%20we%20generate%20both,points%20and%20groups%20become%20larger.

https://www.mathworks.com/help/econ/corrplot.html

https://mode.com/blog/python-data-visualization-libraries/

https://analyticsindiamag.com/top-5-python-libraries-for-data-visualization/