|  |
| --- |
| **National University of Computer and Emerging Sciences** |
| Lab Manual 8  “Data Visualization using Python” |
|  |
| Data Warehousing and Data Mining |
|  |
|  |
|  |

|  |  |
| --- | --- |
| Course Instructor | Ishaq Raza |
| Lab Instructor(s) | Farhan Azhar  Nouman Ali |
| Section | CS |
| Semester | Fall 2019 |

|  |
| --- |
|  |

Data visualization is the discipline of trying to understand data by placing it in a visual context so that patterns, trends and correlations that might not otherwise be detected can be exposed.

Python offers multiple great graphing libraries that come packed with lots of different features. No matter if you want to create interactive, live or highly customized plots python has an excellent library for you.

To get a little overview here are a few popular plotting libraries:

* [**Matplotlib:**](https://matplotlib.org/)low level, provides lots of freedom
* [**Pandas Visualization:**](https://pandas.pydata.org/pandas-docs/stable/visualization.html)easy to use interface, built on Matplotlib
* [**Seaborn:**](https://seaborn.pydata.org/)high-level interface, great default styles
* [**ggplot:**](http://ggplot.yhathq.com/)based on R’s ggplot2, uses [Grammar of Graphics](https://www.amazon.com/Grammar-Graphics-Statistics-Computing/dp/0387245448)
* [**Plotly:**](https://plot.ly/python/)can create interactive plots

In this article, we will learn how to create basic plots using Matplotlib, Pandas visualization and Seaborn as well as how to use some specific features of each library. This article will focus on the syntax and not on interpreting the graphs, which I will cover in another blog post.

In further articles, I will go over interactive plotting tools like Plotly, which is built on D3 and can also be used with JavaScript.

**Importing Datasets**

In this article, we will use two datasets which are freely available. The [Iris](https://archive.ics.uci.edu/ml/datasets/iris) and [Wine Reviews](https://www.kaggle.com/zynicide/wine-reviews) dataset, which we can both load in using pandas read\_csv method.

import pandas as pd

iris = pd.read\_csv('iris.csv', names=['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width', 'class'])

print(iris.head())

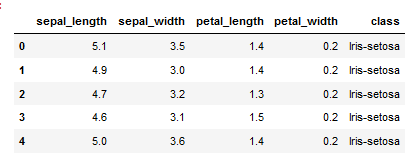


Figure 2: Iris dataset head

wine\_reviews = pd.read\_csv('winemag-data-130k-v2.csv', index\_col=0)

wine\_reviews.head()



Figure 3: Wine Review dataset head

**Matplotlib**

Matplotlib is the most popular python plotting library. It is a low-level library with a Matlab like interface which offers lots of freedom at the cost of having to write more code.

To install Matplotlib pip and conda can be used.

pip install matplotlib  
or  
conda install matplotlib

Matplotlib is specifically good for creating basic graphs like line charts, bar charts, histograms and many more. It can be imported by typing:

import matplotlib.pyplot as plt

**Scatter Plot**

To create a scatter plot in Matplotlib we can use the scatter method. We will also create a figure and an axis using plt.subplots so we can give our plot a title and labels.

# create a figure and axis

fig, ax = plt.subplots()

# scatter the sepal\_length against the sepal\_width

ax.scatter(iris['sepal\_length'], iris['sepal\_width'])

# set a title and labels

ax.set\_title('Iris Dataset')

ax.set\_xlabel('sepal\_length')

ax.set\_ylabel('sepal\_width')

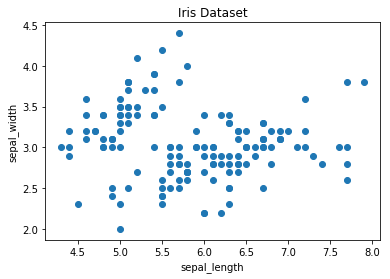


Figure 4: Matplotlib Scatter plot

We can give the graph more meaning by coloring in each data-point by its class. This can be done by creating a dictionary which maps from class to color and then scattering each point on its own using a for-loop and passing the respective color.

# create color dictionary

colors = {'Iris-setosa':'r', 'Iris-versicolor':'g', 'Iris-virginica':'b'}

# create a figure and axis

fig, ax = plt.subplots()

# plot each data-point

for i in range(len(iris['sepal\_length'])):

ax.scatter(iris['sepal\_length'][i], iris['sepal\_width'][i],color=colors[iris['class'][i]])

# set a title and labels

ax.set\_title('Iris Dataset')

ax.set\_xlabel('sepal\_length')

ax.set\_ylabel('sepal\_width')

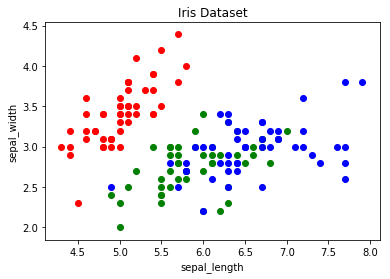


Figure 5: Scatter Plot colored by class

**Line Chart**

In Matplotlib we can create a line chart by calling the plot method. We can also plot multiple columns in one graph, by looping through the columns we want and plotting each column on the same axis.

# get columns to plot

columns = iris.columns.drop(['class'])

# create x data

x\_data = range(0, iris.shape[0])

# create figure and axis

fig, ax = plt.subplots()

# plot each column

for column in columns:

ax.plot(x\_data, iris[column])

# set title and legend

ax.set\_title('Iris Dataset')

ax.legend()

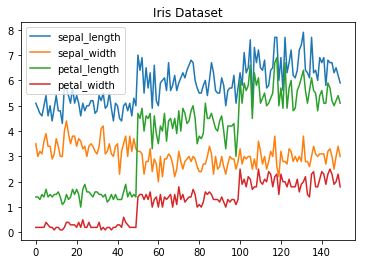


Figure 6: Line Chart

**Histogram**

In Matplotlib we can create a Histogram using the hist method. If we pass it categorical data like the points column from the wine-review dataset it will automatically calculate how often each class occurs.

# create figure and axis

fig, ax = plt.subplots()

# plot histogram

ax.hist(wine\_reviews['points'])

# set title and labels

ax.set\_title('Wine Review Scores')

ax.set\_xlabel('Points')

ax.set\_ylabel('Frequency')

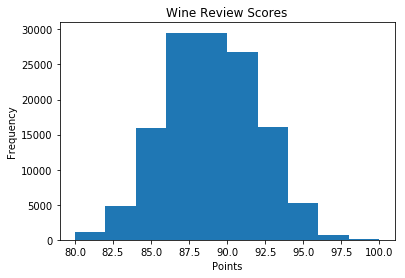


Figure 7: Histogram

**Bar Chart**

A bar chart can be created using the bar method. The bar-chart isn’t automatically calculating the frequency of a category so we are going to use pandas value\_counts function to do this. The bar-chart is useful for categorical data that doesn’t have a lot of different categories (less than 30) because else it can get quite messy.

# create a figure and axis

fig, ax = plt.subplots()

# count the occurrence of each class

data = wine\_reviews['points'].value\_counts()

# get x and y data

points = data.index

frequency = data.values

# create bar chart

ax.bar(points, frequency)

# set title and labels

ax.set\_title('Wine Review Scores')

ax.set\_xlabel('Points')

ax.set\_ylabel('Frequency')

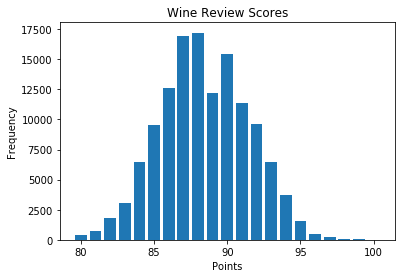


Figure 8: Bar-Chart

**Pandas Visualization**

Pandas is an open source high-performance, easy-to-use library providing data structures, such as dataframes, and data analysis tools like the visualization tools we will use in this article.

Pandas Visualization makes it really easy to create plots out of a pandas dataframe and series. It also has a higher level API than Matplotlib and therefore we need less code for the same results.

Pandas can be installed using either pip or conda.

pip install pandas  
or  
conda install pandas

**Scatter Plot**

To create a scatter plot in Pandas we can call <dataset>.plot.scatter() and pass it two arguments, the name of the x-column as well as the name of the y-column. Optionally we can also pass it a title.

iris.plot.scatter(x='sepal\_length', y='sepal\_width', title='Iris Dataset')

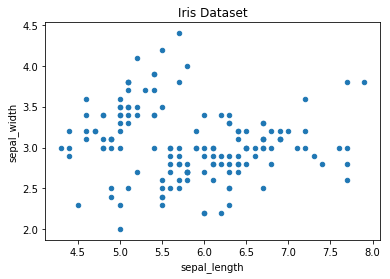


Figure 9: Scatter Plot

As you can see in the image it is automatically setting the x and y label to the column names.

**Line Chart**

To create a line-chart in Pandas we can call <dataframe>.plot.line(). Whilst in Matplotlib we needed to loop-through each column we wanted to plot, in Pandas we don’t need to do this because it automatically plots all available numeric columns (at least if we don’t specify a specific column/s).

iris.drop(['class'], axis=1).plot.line(title='Iris Dataset')

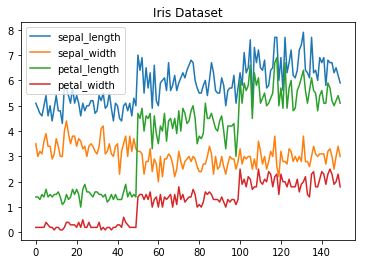


Figure 10: Line Chart

If we have more than one feature Pandas automatically creates a legend for us, as can be seen in the image above.

**Histogram**

In Pandas, we can create a Histogram with the plot.hist method. There aren’t any required arguments but we can optionally pass some like the bin size.

wine\_reviews['points'].plot.hist()



Figure 11: Histogram

It’s also really easy to create multiple histograms.

iris.plot.hist(subplots=True, layout=(2,2), figsize=(10, 10), bins=20)

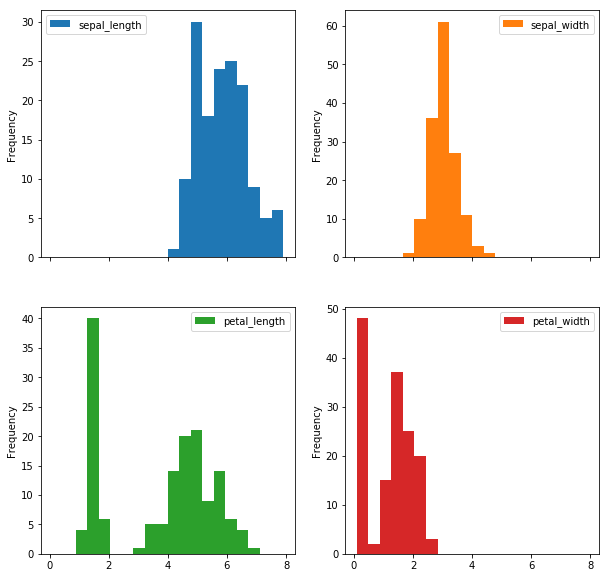


Figure 12: Multiple Histograms

The subplots argument specifies that we want a separate plot for each feature and the layout specifies the number of plots per row and column.

**Bar Chart**

To plot a bar-chart we can use the plot.bar() method, but before we can call this we need to get our data. For this we will first count the occurrences using the value\_count() method and then sort the occurrences from smallest to largest using the sort\_index() method.

wine\_reviews['points'].value\_counts().sort\_index().plot.bar()

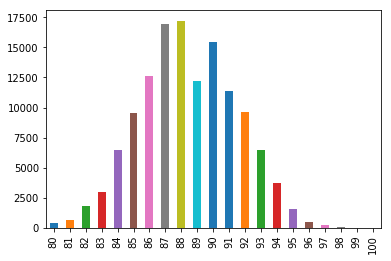


Figure 13: Vertical Bar-Chart

It’s also really simple to make a horizontal bar-chart using the plot.barh() method.

wine\_reviews['points'].value\_counts().sort\_index().plot.barh()

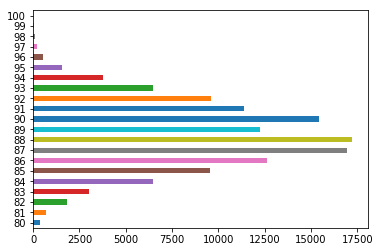


Figure 14: Horizontal Bar-Chart

We can also plot other data then the number of occurrences.

wine\_reviews.groupby("country").price.mean().sort\_values(ascending=False)[:5].plot.bar()

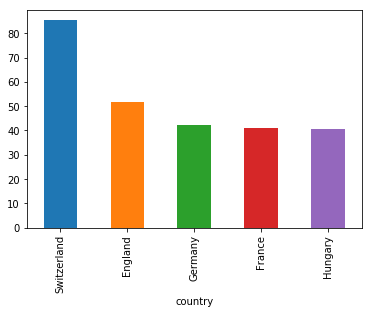


Figure 15: Countries with the most expensive wine(on average)

In the example above we grouped the data by country and then took the mean of the wine prices, ordered it, and plotted the 5 countries with the highest average wine price.

**Seaborn**

Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for creating attractive graphs.

Seaborn has a lot to offer. You can create graphs in one line that would take you multiple tens of lines in Matplotlib. Its standard designs are awesome and it also has a nice interface for working with pandas dataframes.

It can be imported by typing:

import seaborn as sns

**Scatter plot**

We can use the .scatterplot method for creating a scatterplot, and just as in Pandas we need to pass it the column names of the x and y data, but now we also need to pass the data as an additional argument because we aren’t calling the function on the data directly as we did in Pandas.

sns.scatterplot(x='sepal\_length', y='sepal\_width', data=iris)

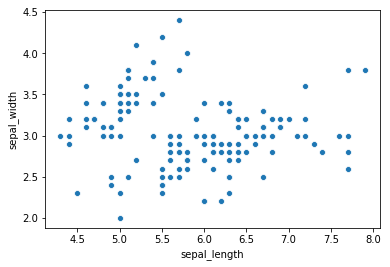


Figure 16: Scatterplot

We can also highlight the points by class using the hue argument, which is a lot easier than in Matplotlib.

sns.scatterplot(x='sepal\_length', y='sepal\_width', hue='class', data=iris)

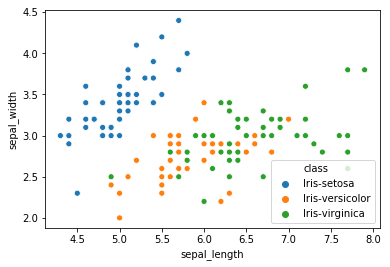


Figure 17: Scatterplot colored by class

**Line chart**

To create a line-chart the sns.lineplot method can be used. The only required argument is the data, which in our case are the four numeric columns from the Iris dataset. We could also use the sns.kdeplot method which rounds of the edges of the curves and therefore is cleaner if you have a lot of outliers in your dataset.

sns.lineplot(data=iris.drop(['class'], axis=1))

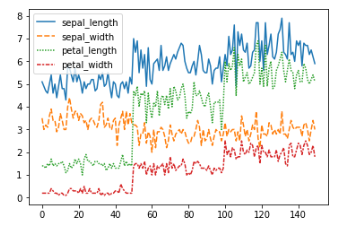


Figure 18: Line Chart

**Histogram**

To create a histogram in Seaborn we use the sns.distplot method. We need to pass it the column we want to plot and it will calculate the occurrences itself. We can also pass it the number of bins, and if we want to plot a gaussian kernel density estimate inside the graph.

sns.distplot(wine\_reviews['points'], bins=10, kde=False)

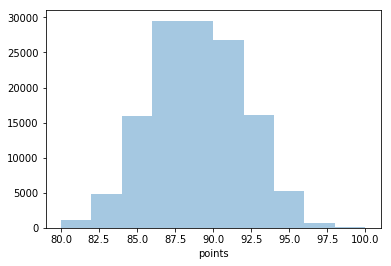


Figure 19: Histogram

sns.distplot(wine\_reviews['points'], bins=10, kde=True)

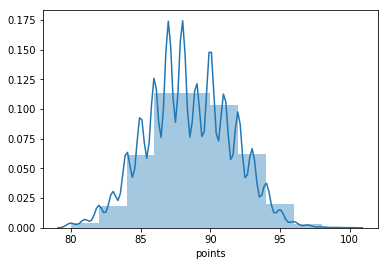


Figure 20: Histogram with gaussian kernel density estimate

**Bar chart**

In Seaborn a bar-chart can be created using the sns.countplot method and passing it the data.

sns.countplot(wine\_reviews['points'])

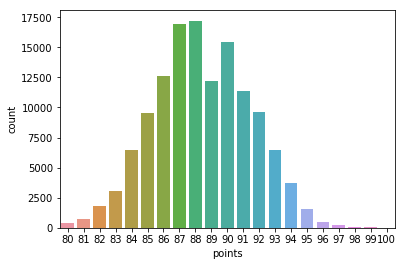


Figure 21: Bar-Chart

**Other graphs**

Now that you have a basic understanding of the Matplotlib, Pandas Visualization and Seaborn syntax I want to show you a few other graph types that are useful for extracting insides.

For most of them, Seaborn is the go-to library because of its high-level interface that allows for the creation of beautiful graphs in just a few lines of code.

**Box plots**

A Box Plot is a graphical method of displaying the [five-number summary](https://en.wikipedia.org/wiki/Five-number_summary). We can create box plots using seaborns sns.boxplot method and passing it the data as well as the x and y column name.

df = wine\_reviews[(wine\_reviews['points']>=95) & (wine\_reviews['price']<1000)]

sns.boxplot('points', 'price', data=df)

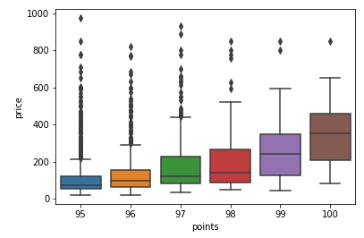


Figure 22: Boxplot

Box Plots, just like bar-charts are great for data with only a few categories but can get messy really quickly.

**Heatmap**

A Heatmap is a graphical representation of data where the individual values contained in a [matrix](https://en.wikipedia.org/wiki/Matrix_(mathematics)) are represented as colors. Heatmaps are perfect for exploring the correlation of features in a dataset.

To get the correlation of the features inside a dataset we can call <dataset>.corr(), which is a Pandas dataframe method. This will give us the [correlation matrix](https://www.displayr.com/what-is-a-correlation-matrix/).

We can now use either Matplotlib or Seaborn to create the heatmap.

Matplotlib:

import numpy as np

# get correlation matrix

corr = iris.corr()

fig, ax = plt.subplots()

# create heatmap

im = ax.imshow(corr.values)

# set labels

ax.set\_xticks(np.arange(len(corr.columns)))

ax.set\_yticks(np.arange(len(corr.columns)))

ax.set\_xticklabels(corr.columns)

ax.set\_yticklabels(corr.columns)

# Rotate the tick labels and set their alignment.

plt.setp(ax.get\_xticklabels(), rotation=45, ha="right",

rotation\_mode="anchor")

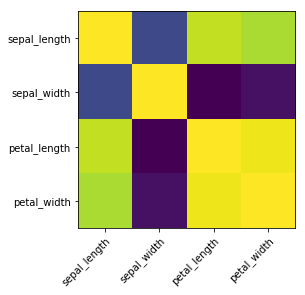


Figure 23: Heatmap without annotations

To add annotations to the heatmap we need to add two for loops:

# get correlation matrix

corr = iris.corr()

fig, ax = plt.subplots()

# create heatmap

im = ax.imshow(corr.values)

# set labels

ax.set\_xticks(np.arange(len(corr.columns)))

ax.set\_yticks(np.arange(len(corr.columns)))

ax.set\_xticklabels(corr.columns)

ax.set\_yticklabels(corr.columns)

# Rotate the tick labels and set their alignment.

plt.setp(ax.get\_xticklabels(), rotation=45, ha="right",

rotation\_mode="anchor")

# Loop over data dimensions and create text annotations.

for i in range(len(corr.columns)):

for j in range(len(corr.columns)):

text = ax.text(j, i, np.around(corr.iloc[i, j], decimals=2),

ha="center", va="center", color="black")

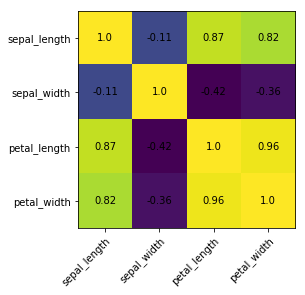


Figure 24: Heatmap with annotations

Seaborn makes it way easier to create a heatmap and add annotations:

sns.heatmap(iris.corr(), annot=True)

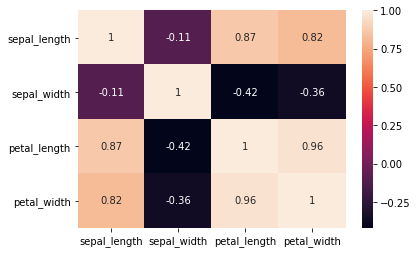


Figure 25: Heatmap with annotations

**Faceting**

Faceting is the act of breaking data variables up across multiple subplots and combining those subplots into a single figure.

Faceting is really helpful if you want to quickly explore your dataset.

To use one kind of faceting in Seaborn we can use the FacetGrid. First of all, we need to define the FacetGrid and pass it our data as well as a row or column, which will be used to split the data. Then we need to call the map function on our FacetGrid object and define the plot type we want to use, as well as the column we want to graph.

g = sns.FacetGrid(iris, col='class')

g = g.map(sns.kdeplot, 'sepal\_length')

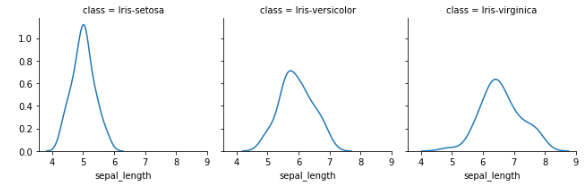


Figure 26: Facet-plot

You can make plots a lot bigger and more complicated than the example above. You can find a few examples [here](https://seaborn.pydata.org/generated/seaborn.FacetGrid.html).

**Pairplot**

Lastly, I will show you Seaborns pairplot and Pandas scatter\_matrix , which enable you to plot a grid of pairwise relationships in a dataset.

sns.pairplot(iris)

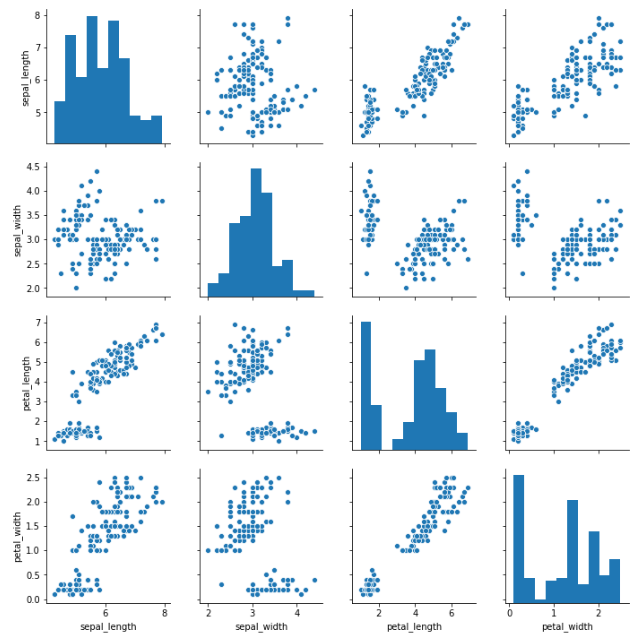


Figure 27: Pairplot

from pandas.plotting import scatter\_matrix

fig, ax = plt.subplots(figsize=(12,12))

scatter\_matrix(iris, alpha=1, ax=ax)

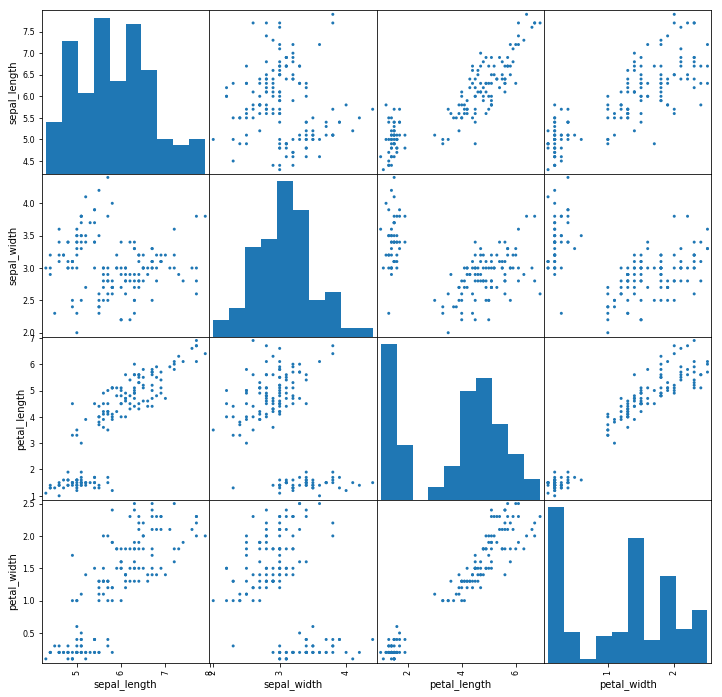
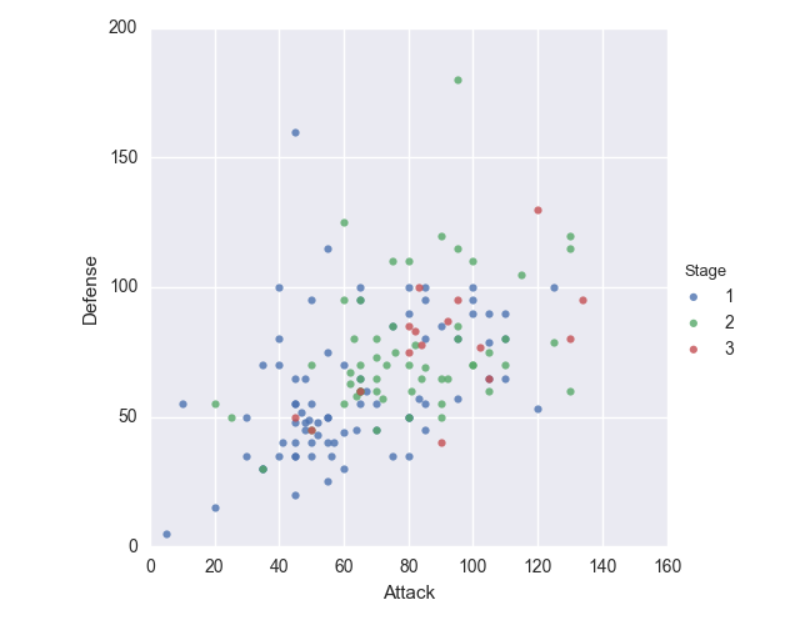


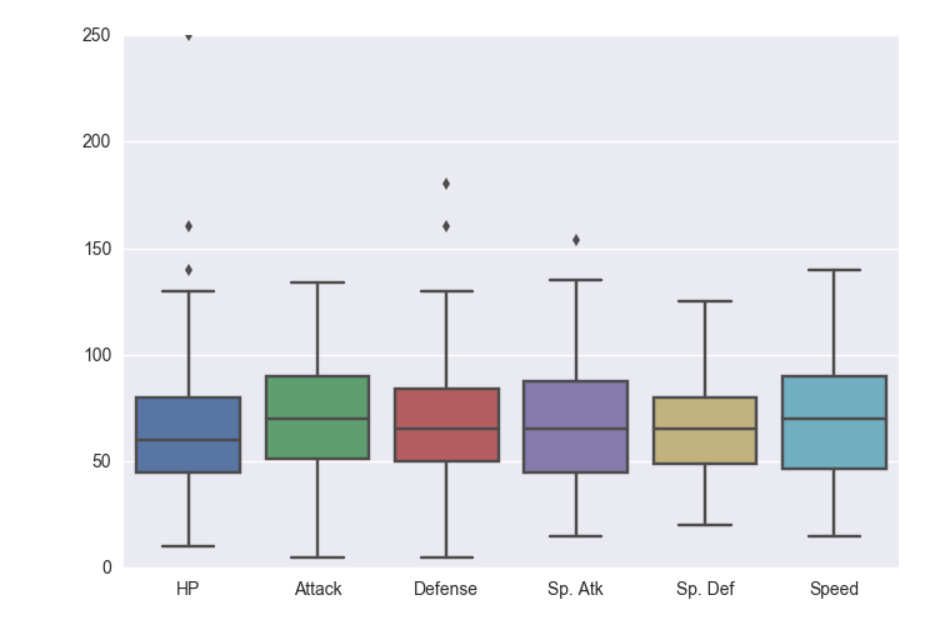
Figure 28: Scatter matrix

As you can see in the images above these techniques are always plotting two features with each other. The diagonal of the graph is filled with histograms and the other plots are scatter plots.

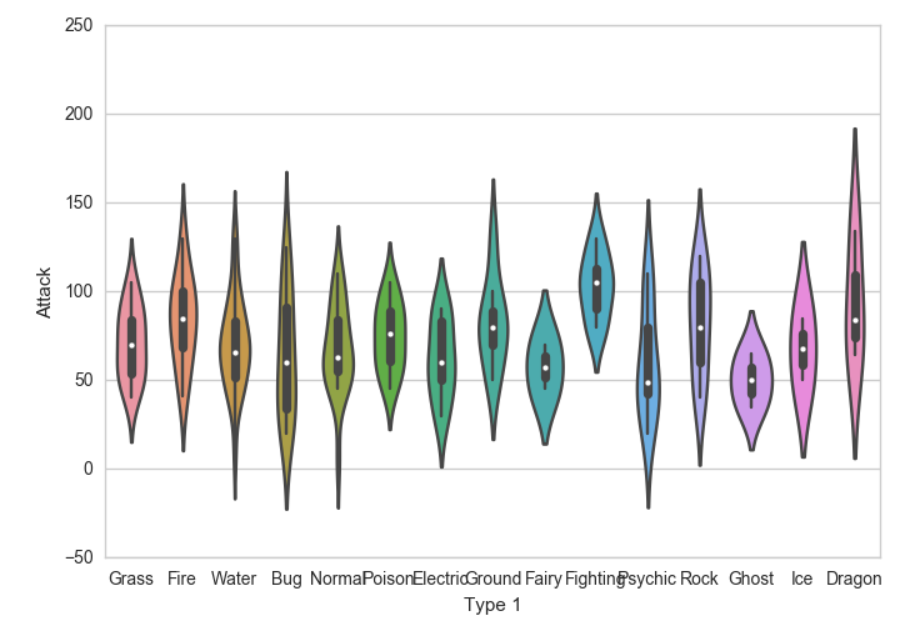
**Task 1:** Plot distribution of defense and attack of every Pokemon, with using Stage as third dimension of information using color.



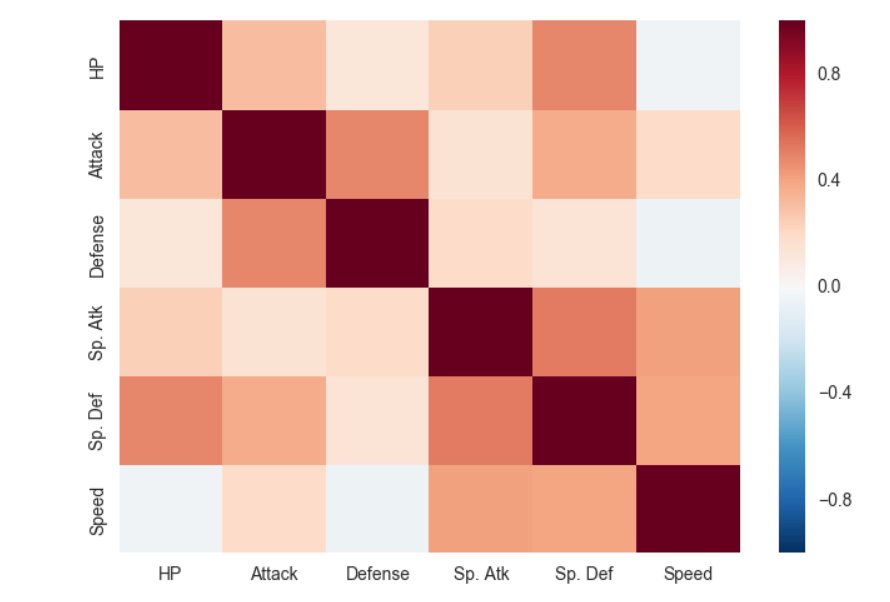
**Task 2:** Make a box plot for Pokémon's combat stats



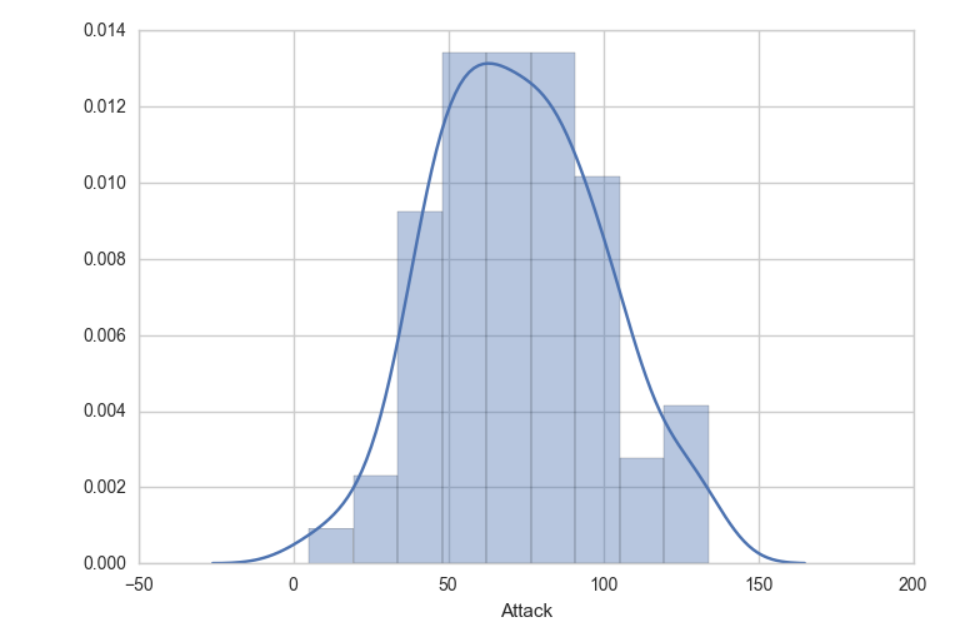
**Task 3:** Visualize the distribution of Attack by Pokémon's primary type using Violin plot



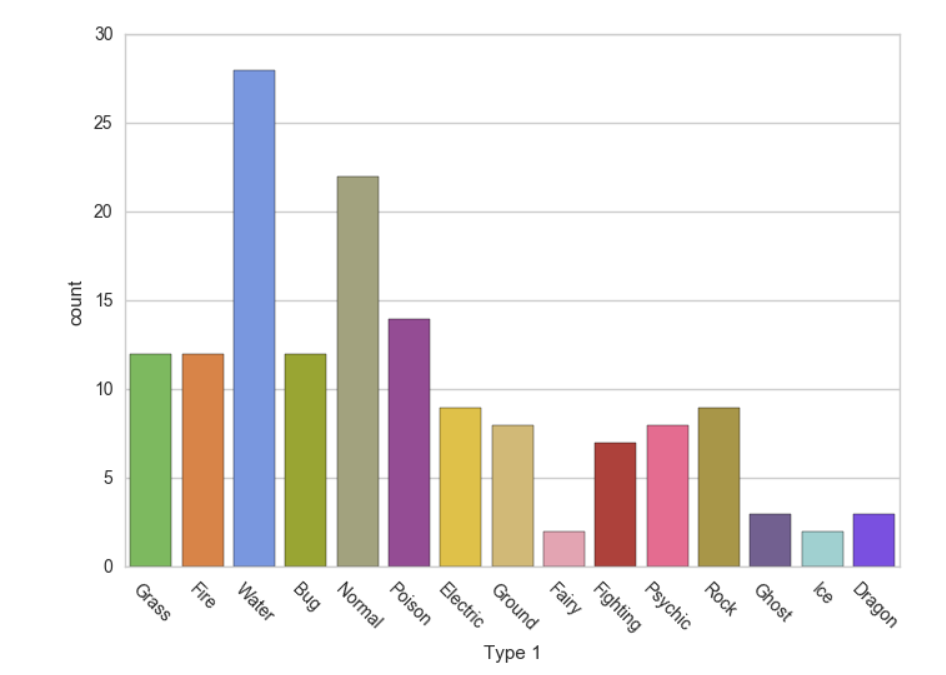
**Task 4:** Create correlation analysis for different Pokemon stats



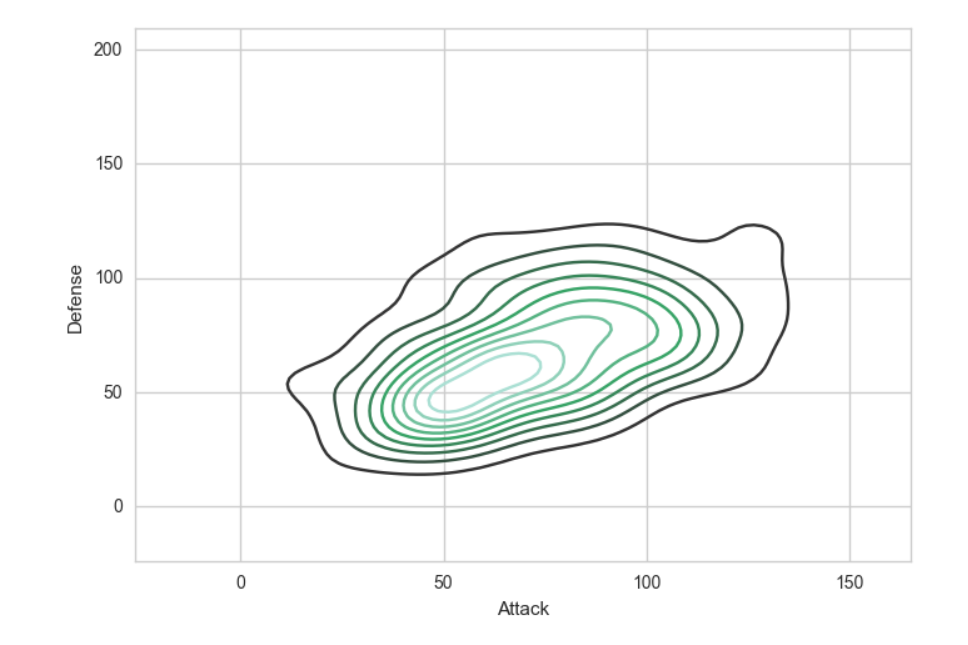
**Task 5:** Create Histogram for Attack data



**Task 6:** Create Bar Chart using ‘Type 1’ attribute.



**Task 7:** Create density plot using Attack and Defense attributes



**Task 8:** Extract the top 5 Stage 3 Pokemon which have the highest Speed and Sp. Defense. Make a 3D Plot of their HP, Sp. Attack and Type attributes.