Introduction to Data Visualization in Python

How to make graphs using Matplotlib, Pandas and Seaborn

Gilbert Tanner
Jan 23, 2019 · 9 min read

Figure 1: Photo by Lukas Blazek on Unsplash

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:** low level, provides lots of freedom
- **Pandas Visualization:** easy to use interface, built on Matplotlib
- **Seaborn:** high-level interface, great default styles
- **ggplot:** based on R's ggplot2, uses Grammar of Graphics
- **Plotly:** 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 and 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_widt
print(iris.head())
```

load_iris.py hosted with ♡ by GitHub view raw

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Figure 2: Iris dataset head

```
wine_reviews = pd.read_csv('winemag-data-130k-v2.csv', index_col=0)
wine_reviews.head()
```

load_wine_reviews.py hosted with ♡ by GitHub view raw

	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	title	variety	winery
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	Kerin O'Keefe	@kerinokeefe	Nicosia 2013 Vulkà Bianco (Etna)	White Blend	Nicosia
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	Roger Voss	@vossroger	Quinta dos Avidagos 2011 Avidagos Red (Douro)	Portuguese Red	Quinta dos Avidagos
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Rainstorm 2013 Pinot Gris (Willamette Valley)	Pinot Gris	Rainstorm
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	Alexander Peartree	NaN	St. Julian 2013 Reserve Late Harvest Riesling ...	Riesling	St. Julian
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Sweet Cheeks 2012 Vintner's Reserve Wild Child...	Pinot Noir	Sweet Cheeks

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')
```

matplotlib_simple_scatterplot.py hosted with ♡ by GitHub view raw

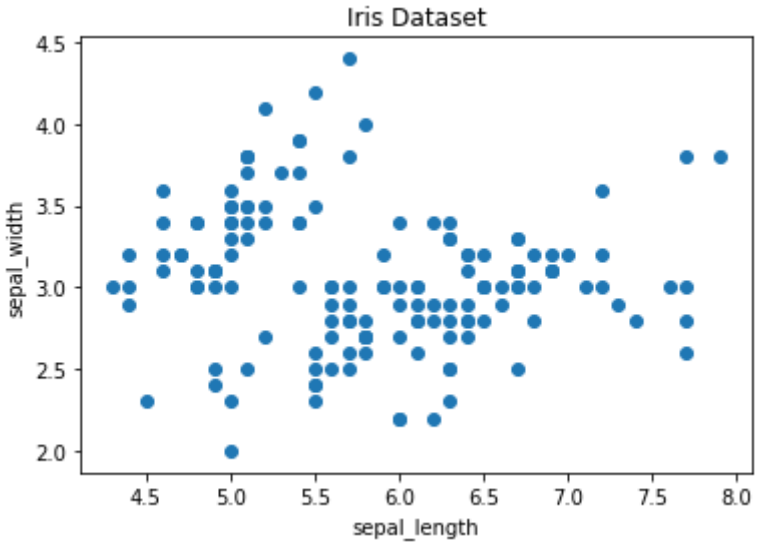


Figure 4: Matplotlib Scatter plot

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')
```

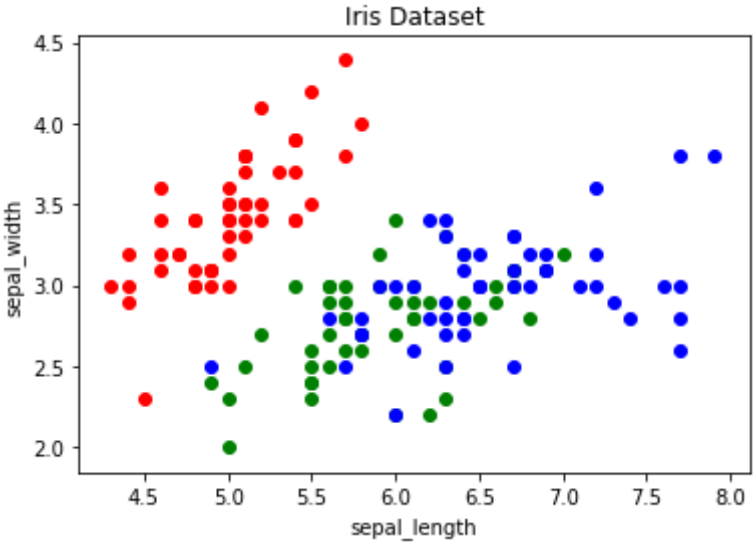


Figure 5: Scatter Plot colored by class

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], label=column)
# set title and legend
ax.set_title('Iris Dataset')
ax.legend()
```

matplotlib_simple_linechart.py hosted with ♡ by GitHub view raw

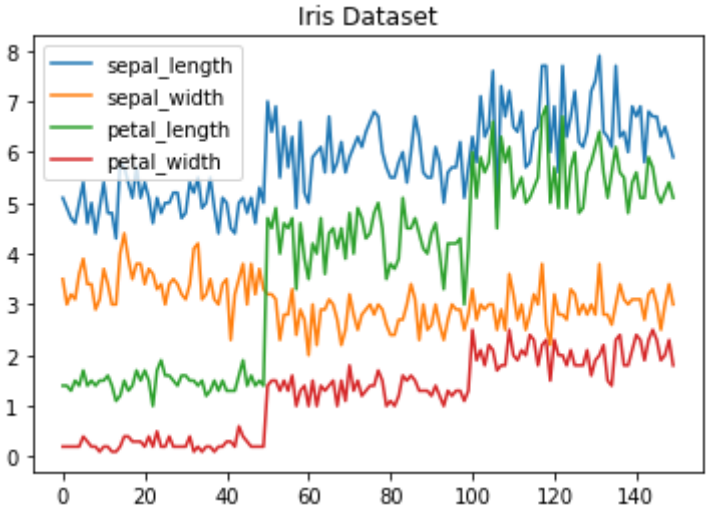


Figure 6: Line Chart

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')
```

matplotlib_simple_histogram.py hosted with ♡ by GitHub view raw

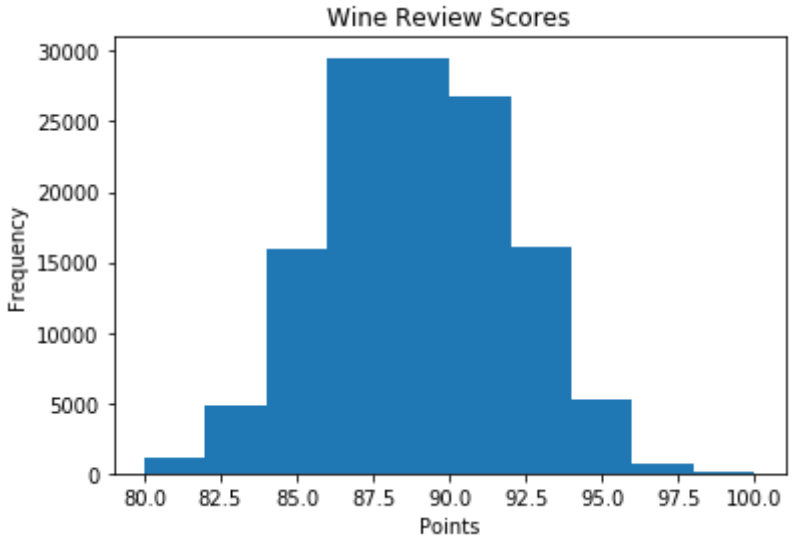


Figure 7: Histogram

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')
```

matplotlib simple barchart.py hosted with ♡ by GitHub view raw

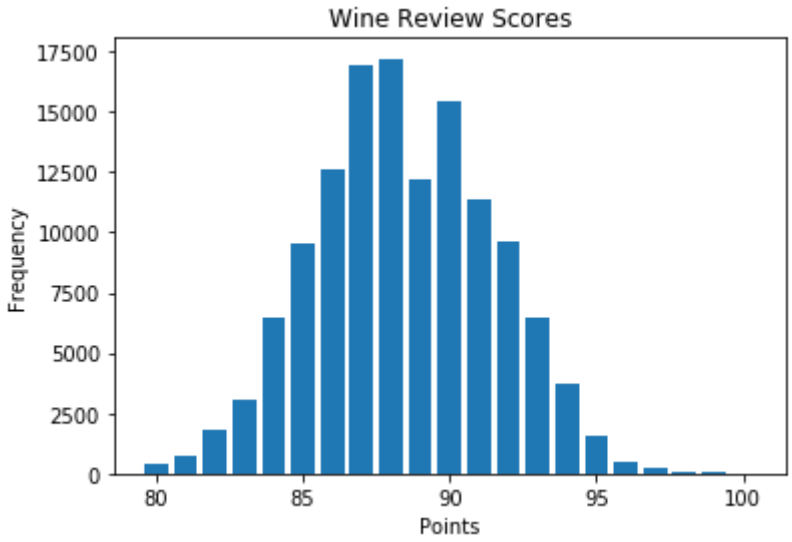


Figure 8: Bar-Chart

• • •

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.

```
1  iris.plot.scatter(x='sepal_length', y='sepal_width', title='Iris Dataset')
```

pandas_simple_scatterplot.py hosted with ♡ by GitHub view raw

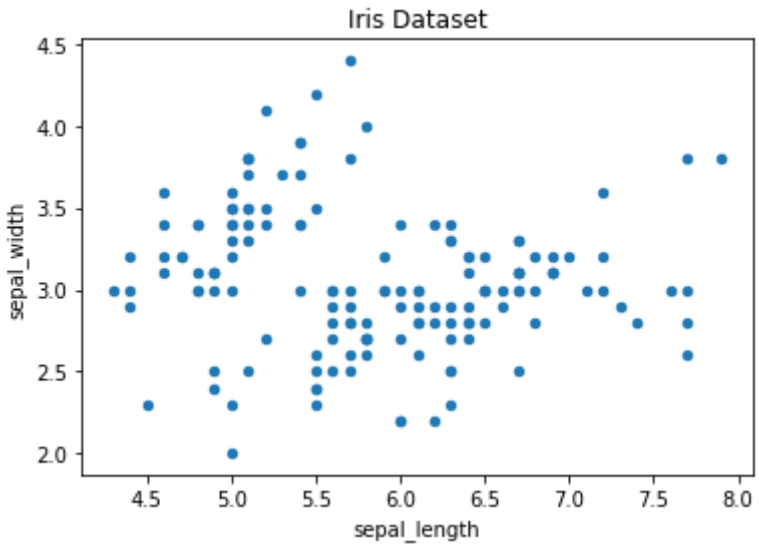


Figure 9: Scatter Plot

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')
```

pandas_simple_linechart.py hosted with ♡ by GitHub view raw

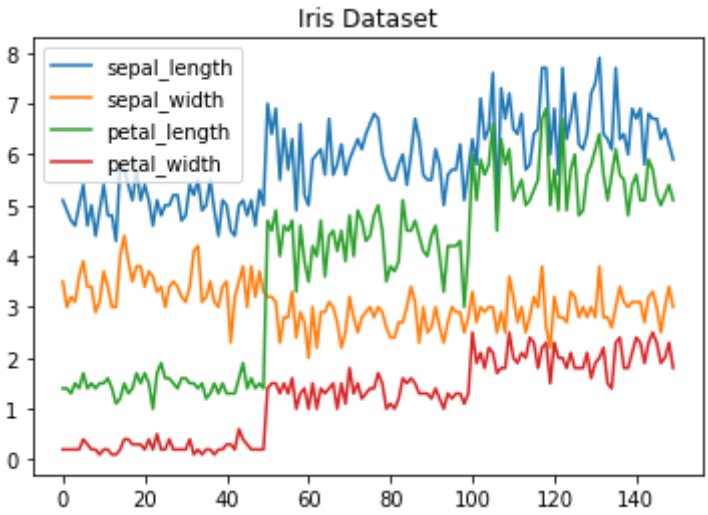


Figure 10: Line Chart

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()
```

pandas_simple_histogram.py hosted with ♡ by GitHub view raw

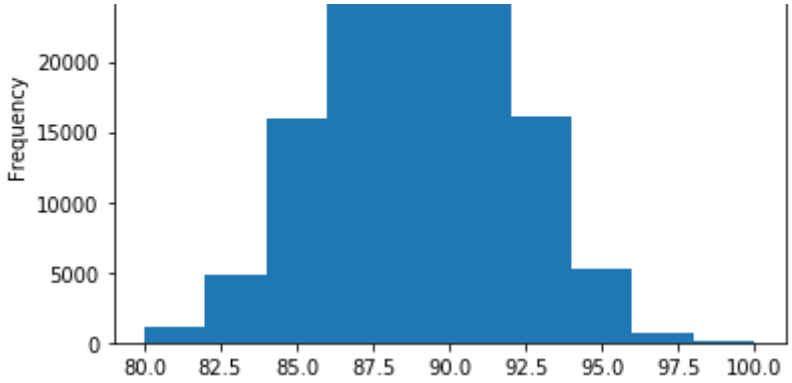


Figure 11: Histogram

It's also really easy to create multiple histograms.

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

pandas_multiple_histograms.py hosted with ♡ by GitHub view raw

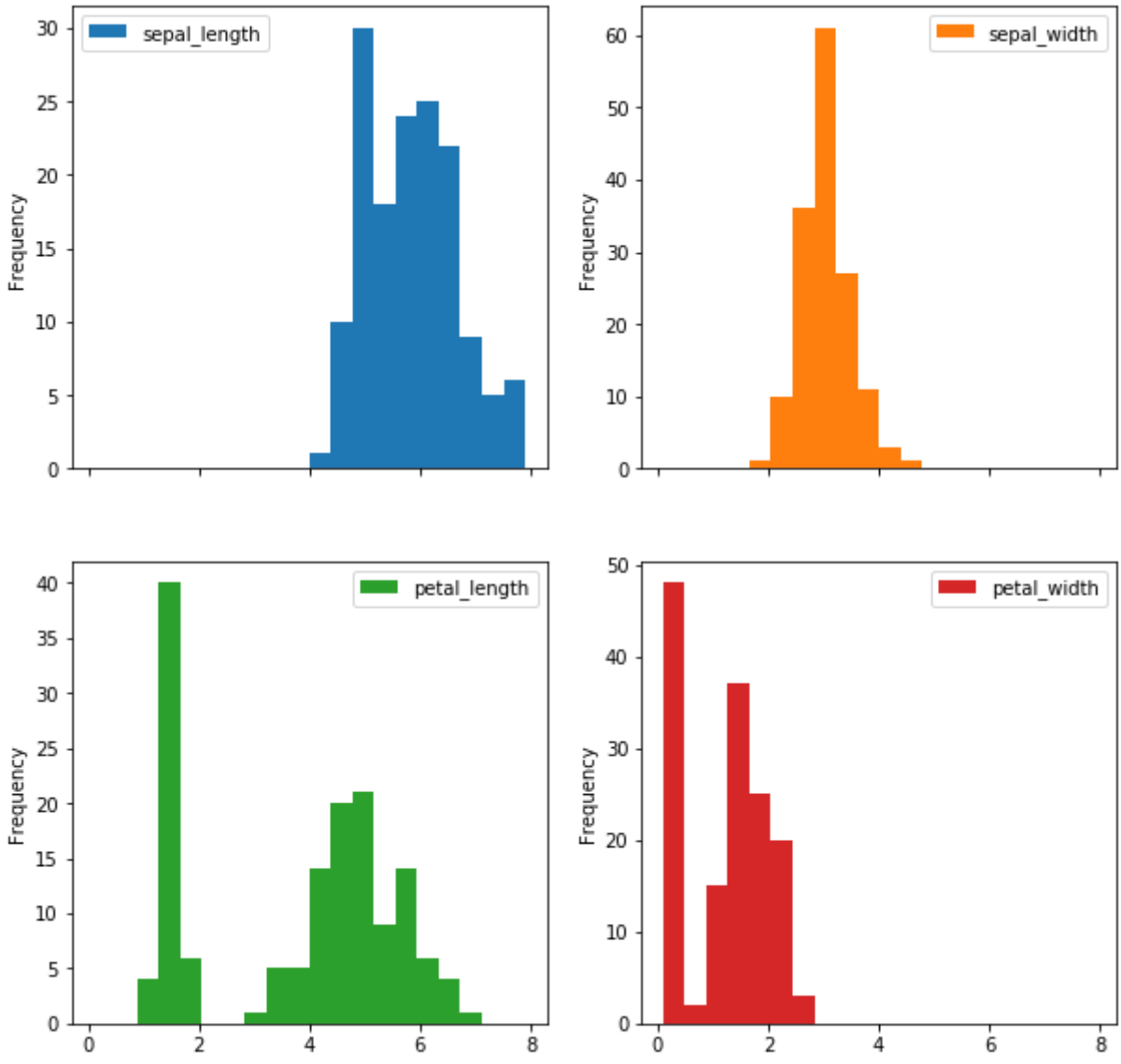


Figure 12: Multiple Histograms

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.

```
1  wine_reviews['points'].value_counts().sort_index().plot.bar()
```

pandas_simple_barchart_vertical.py hosted with ♡ by GitHub view raw

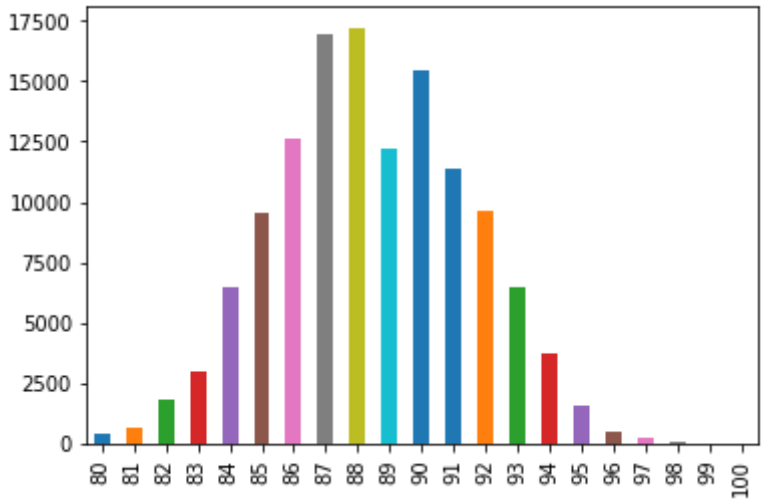
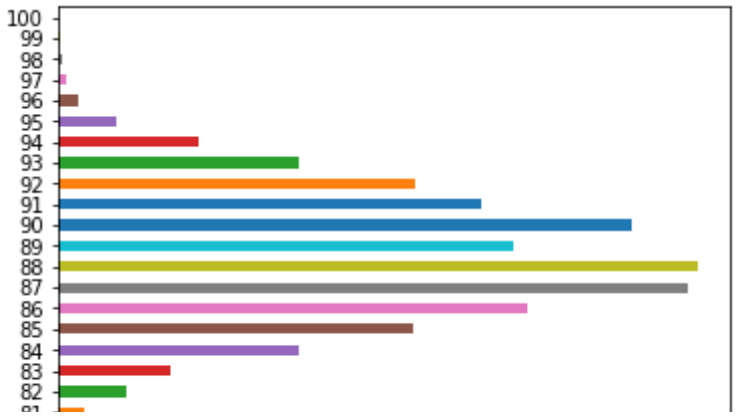


Figure 13: Vertical Bar-Chart

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

```
1  wine_reviews['points'].value_counts().sort_index().plot.barh()
```

pandas_simple_barchart_horizontal.py hosted with ♡ by GitHub view raw



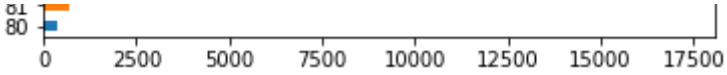


Figure 14: Horizontal Bar-Chart

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

```
1  wine_reviews.groupby("country").price.mean().sort_values(ascending=False)[:5].plot.bar()
```

pandas_simple_barchart_vertical_2.py hosted with ♡ by GitHub view raw

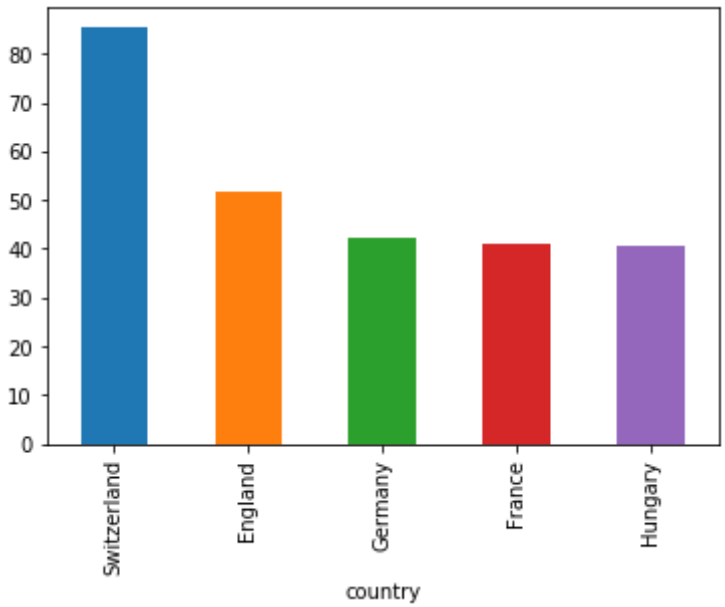


Figure 15: Countries with the most expensive wine(on average)

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.

```
1  sns.scatterplot(x='sepal_length', y='sepal_width', data=iris)
```

seaborn_simple_scatterplot.py hosted with ♡ by GitHub view raw

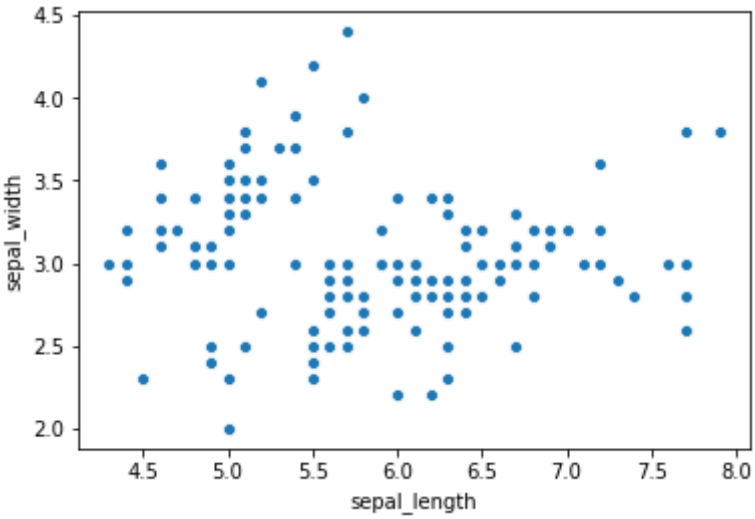
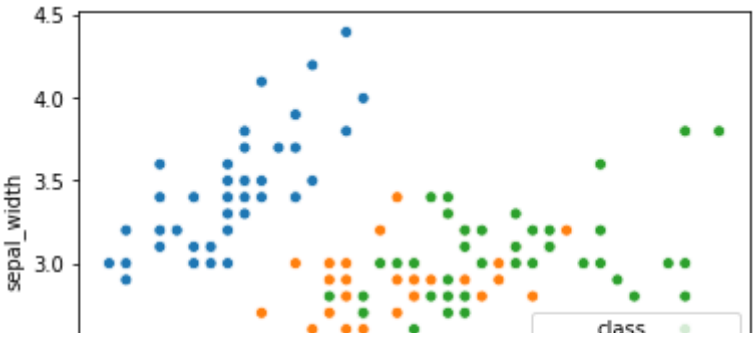


Figure 16: Scatterplot

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

```
1  sns.scatterplot(x='sepal_length', y='sepal_width', hue='class', data=iris)
```

seaborn_simple_scatterplot_colored.py hosted with ♡ by GitHub view raw



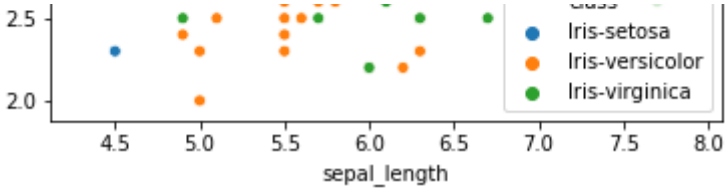


Figure 17: Scatterplot colored by class

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.

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

seaborn_simple_linechart.py hosted with ♡ by GitHub view raw

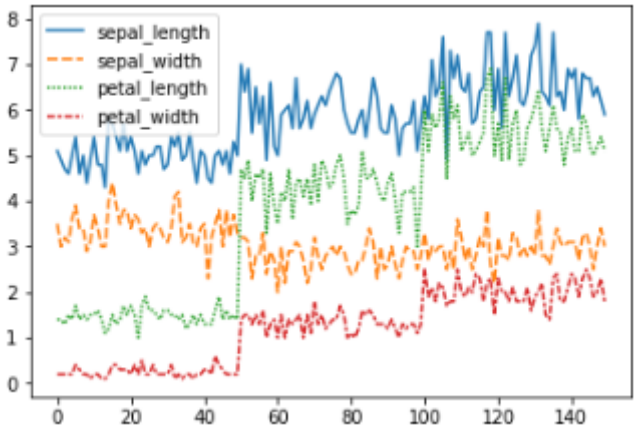


Figure 18: Line Chart

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.

```
1  sns.distplot(wine_reviews['points'], bins=10, kde=False)
```

seaborn_simple_histogram.py hosted with ♡ by GitHub view raw

Figure 19: Histogram

```
sns.distplot(wine_reviews['points'], bins=10, kde=True)
```

seaborn_simple_histogram_with_gaussian.py hosted with ♡ by GitHub view raw

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'])
```

seaborn_simple_bar_chart.py hosted with ♡ by GitHub view raw

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. We can create box plots using seaborns `sns.boxplot` method and passing it the data as well as the x and y column name.

```
1  df = wine_reviews[(wine_reviews['points']>=95) & (wine_reviews['price']<1000)]
2  sns.boxplot('points', 'price', data=df)
```

seaborn_boxplot.py hosted with ♡ by GitHub view raw

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 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.

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")
```

matplotlib_heatmap.py hosted with ♡ by GitHub view raw

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)
```

seaborn_heatmap.py hosted with ♡ by GitHub view raw

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')
```

seaborn_faceting.py hosted with ♡ by GitHub view raw

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.

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.

```
1  sns.pairplot(iris)
```

seaborn_pairplot.py hosted with ♡ by GitHub view raw

Figure 27: Pairplot

```
from pandas.plotting import scatter_matrix

fig, ax = plt.subplots(figsize=(12,12))
scatter_matrix(iris, alpha=1, ax=ax)
```

pandas_scatter_matrix_exampel.py hosted with ♡ by GitHub view raw

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.

. . .

Recommended Reading

Scraping Reddit data

How to scrape data from Reddit using the Python Reddit API Wrapper(PRAW)

towardsdatascience.com

. . .

Conclusion

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. In this article, we looked at Matplotlib, Pandas visualization and Seaborn.

If you liked this article consider subscribing on my Youtube Channel and following me on social media.

The code covered in this article is available as a Github Repository.

If you have any questions, recommendations or critiques, I can be reached via Twitter or the comment section.

Thanks to TDS Team and David Errington.

Data Science Data Visualization Matplotlib Pandas Seaborn