TOOL

Plotting Vectors, Matrices, and Pandas DataFrames

Plotting can be extremely useful for visually interpreting complex and/or large amounts of numerical information. As you may have already observed, there are various approaches and several plotting libraries for visualizing data in Python. Although this tool will introduce some conventions along with a few libraries, there is no single correct method; how you choose to display and format your plots is a matter of preference.

Conventions

1. Label your axes and specify metrics.
2. Use distinguishing colors.
3. Use space meaningfully.

Matplotlib

Matplotlib is an all-purpose visualization library that has many functions for creating different types of static and interactive plots. Pyplot is a popular interface of Matplotlib that has similar plotting functionalities to that of another programming language, MATLAB. Pyplot can be imported with:

```python
import matplotlib.pyplot as plt
```

A plot consists of a *figure*, which is the graphical area that includes all plot elements, and its *axes* object(s), which specify the coordinate system. These can be created together or separately:

```python
fig, ax = plt.subplots()
fig = plt.figure()
ax = plt.axes()
```

It is good practice to include axes labels, and you can do so by calling the following axes object's functions:

```
ax.set_title('axis title')
ax.set_xlabel('x')
ax.set_ylabel('y')
```

You can also use the axes object to plot shapes and objects on the figure. Below is a function that plots an arrow with a text label, given a vector x:

```
def Vec(x, text='', col='black', width=0.001):
ax.arrow(x=0, y=0, dx=x[0], dy=x[1], width=width, head_width=0.2,
   head_length=0.1, fc=col, ec=col)
ax.text(x[0], x[1], s=text, size=15, color='black')
```

Please refer to Matplotlib's axis documentation for details of function arguments as well as other functions. You can edit the *plt* object itself. Below, the code specifies the grid space of the plot and the surrounding white space:

```
plt.grid()
plt.xlim(-2,5)
plt.ylim(-2,5)
plt.tight_layout()
```

Refer to pyplot documentation for additional information. To display the plot, you can call:

```
plt.show()
```

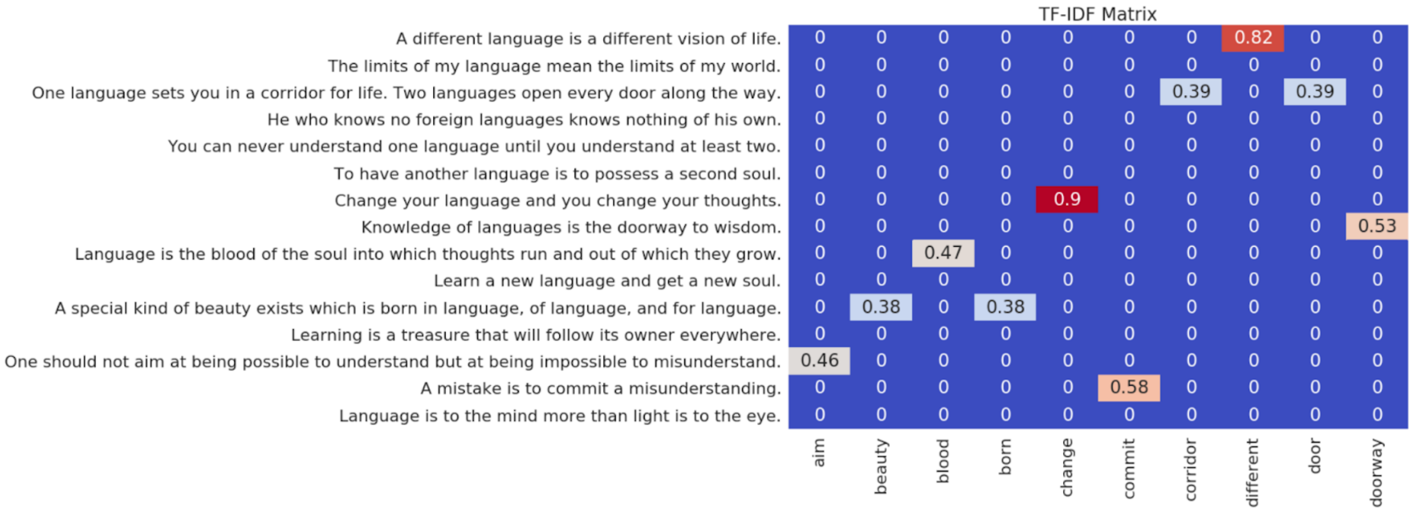
Seaborn

Seaborn is a higher-level API based on Matplotlib that allows you to create beautiful plots. To use this library, you can import it by calling:

```
import seaborn as sns
```

You can use Seaborn to plot a heatmap of a Pandas DataFrame matrix stored in the dfTFIDF1 variable:

```
ax = sns.heatmap(dfTFIDF1, annot=True, cmap='coolwarm', cbar=False)
_ = ax.set_title('TF-IDF Matrix')
```



Notice that the Seaborn heatmap function includes color schemes, like 'coolwarm', that you can specify in the function arguments. Please refer to Seaborn documentation for additional functionalities.

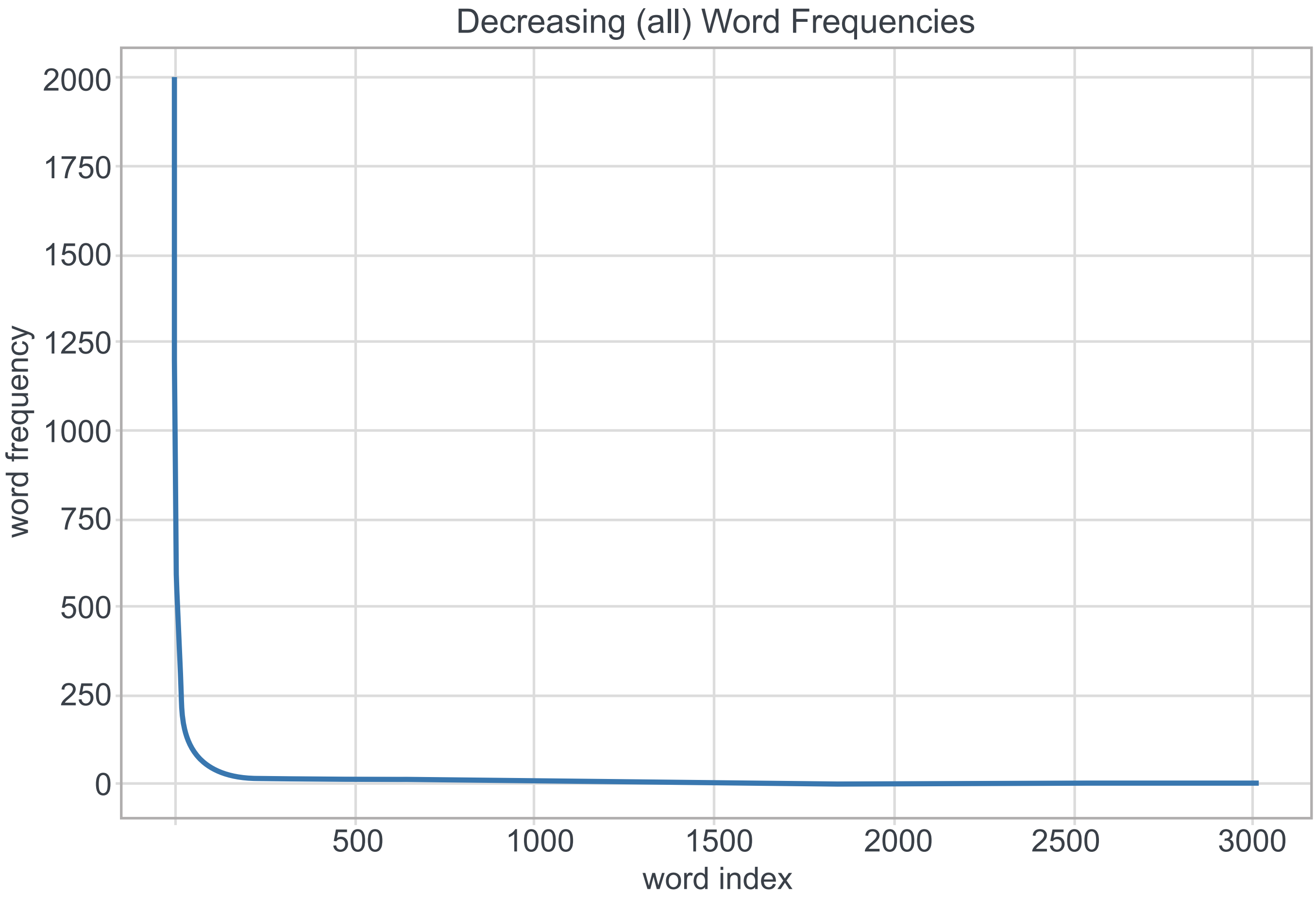
Pandas

Although more limited in formatting options and plotting styles compared to Matplotlib and Seaborn, Pandas also has several built in plotting functions for its DataFrames. You can import Pandas by calling:

```
import pandas as pd
```

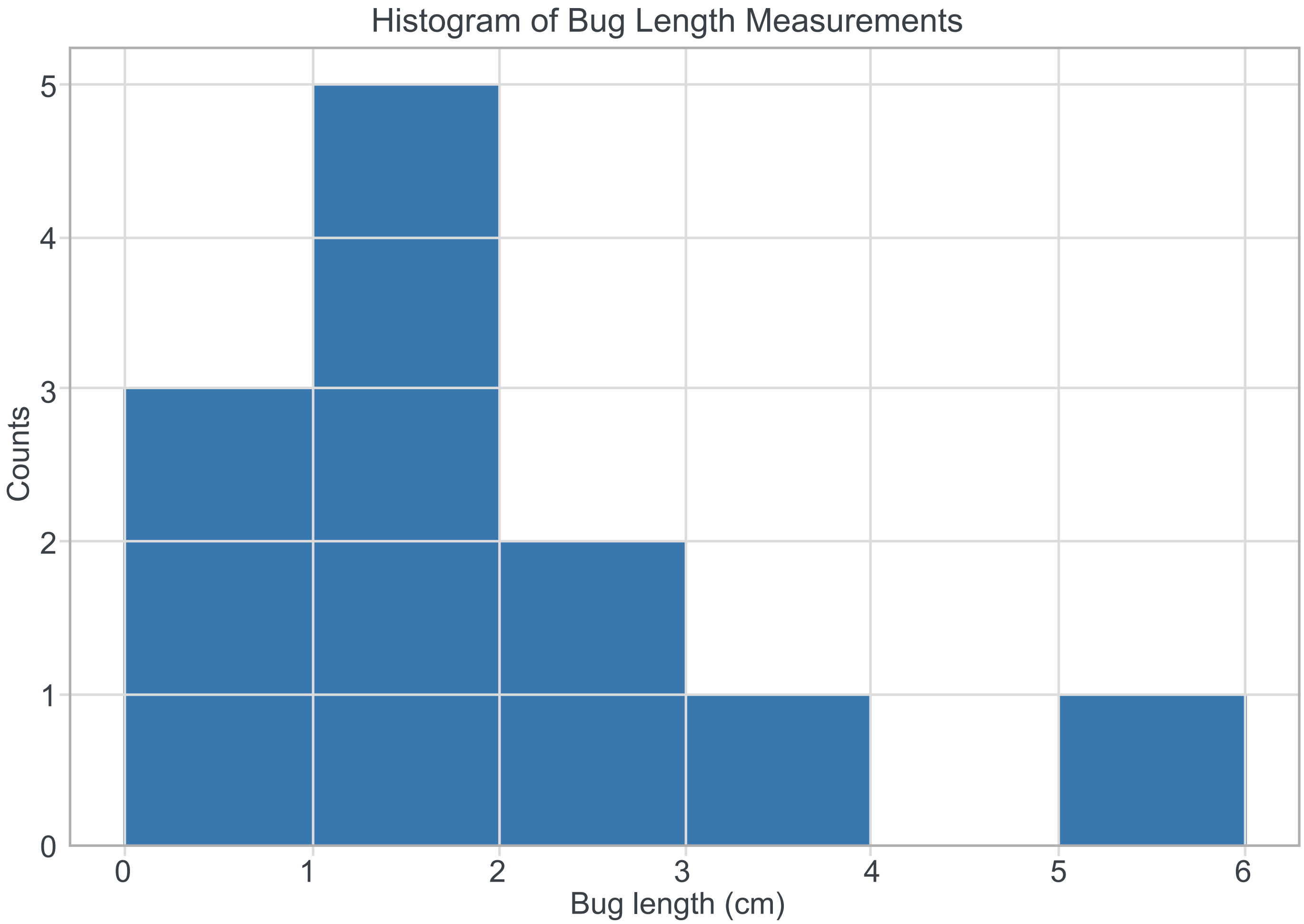
You can use Pandas to plot data from a specific column in a DataFrame by using the `plot()` method:

```
ax = df.freq.plot(figsize=(30,4), title='Decreasing (all) word frequencies',
grid=True, lw=4);
ax.set_xlabel("word index")
ax.set_ylabel("word frequency")
```



Another type of plot that you can make using a built-in Pandas DataFrame method is a histogram. You can plot the counts of classification labels using the `hist()` method:

```
df = pd.DataFrame({'labels': [0, 0, 1, 2, 0, 3, 2, 1, 1, 1, 1, 5]})
plot = df.hist(figsize=[10,5], bins=[0, 1, 2, 3, 4, 5, 6])
for ax in plot.flatten():
    ax.set_xlabel('Bug length (cm)')
    ax.set_ylabel('Counts')
    ax.set_title('Histogram of Bug Length Measurements')
```



Please refer to Pandas documentation for additional functionalities.

Long Descripton for Histogram of Bug Length Measurements:

The data for Histogram of Bug Length Measurements is detailed in the table below:

X: Bug Length (cm)	Y: Counts of Classification Labels
0-1	3
1-2	5
2-3	2
3-4	1
4-5	0
5-6	1