Data Viusalisation with Python Programming

- Presentation By Uplatz
- Contact us: https://training.uplatz.com
- Email: info@uplatz.com
- Phone: +44 7836 212635

Advanced data Visualisation tools

Uplatz

Learning outcomes:

- Waffle Chart
- Word Cloud
- HEAT MAP

Waffle Chart

A Waffle Chart is a gripping visualization technique that is normally created to display progress towards goals. Where each cell in the Waffle Chart constitutes of 10 X 10 cell grid in which each cell represents one percentage point summing up to total 100%. It is commonly an effective option when you are trying to add interesting visualization features to a visual. Waffle Charts are widely used as an Excel dashboard.

Waffle Chart

Waffle chart has a similar squares structure and is an interesting plot mainly used to display progress towards the goal.
Waffle charts can be used to display proportions as well. A waffle chart shows progress towards a target or a completion percentage. There is a grid of small cells, of which coloured cells represent the data. A chart can consist of one category or several categories. Multiple waffle charts can be put together to show a comparison between different charts.

Waffle Chart

For generating Waffle Chart in Python, modules needed are – **matplotlib**, **pandas** and **pyWaffle**. To install **pyWaffle**, open command prompt and change the directory to current working directory of python and run any of the following command

pip3 install pyWaffle OR
python -m pip install pyWaffle OR
pip install --user pyWaffle

Waffle Chart

Creating Waffle Chart: First example

First of all we need to import the necessary libraries for creating waffle chart.

```
import pandas as pd
import matplotlib.pyplot as plt
from pywaffle import Waffle
```

Waffle Chart

Creating Waffle Chart:

Create the data.

```
data ={'phone': ['Xiaomi', 'Samsung',
                 'Apple', 'Nokia', 'Realme'],
       'stock': [44, 12, 8, 5, 3]
      }
```

Waffle Chart

Creating Waffle Chart:

Convert the data into data frame and use the plt.figure() as shown below.

```
df = pd.DataFrame(data) #Creating data frame

 fig = plt.figure(
   FigureClass = Waffle,
   rows = 5,
   values = df.stock,
   labels = list(df.phone) )
```

Waffle Chart

Creating Waffle Chart:

Example_2: Using pandas calculations for a waffle chart.

First we’ll read in/create our dataset.

```
df = pd.DataFrame({ 'District': ['District 12', 'District 23', 'District 32', 'District 45', 'District 65', 'District 67', 'District 33'], 'party': ['Republican', 'Republican', 'Republican', 'Republican', 'Democrat', 'Democrat', 'Democrat'] })
print(df)
```

Waffle Chart

Creating Waffle Chart:

Example_2: Using pandas calculations for a waffle chart.

After the execution of the code we will get the data frame look like this.

	District	party
0	District 12	Republican
1	District 23	Republican
2	District 32	Republican
3	District 45	Republican
4	District 65	Democrat
5	District 67	Democrat
6	District 33	Democrat

Waffle Chart

Creating Waffle Chart:

Example_2: Using pandas calculations for a waffle chart.

We can’t use this raw data for our waffle chart, because we need a list of numbers for it! In order to chart the number of Democrats and Republicans, we need to first get a value counts.

```
calculated = df.party.value_counts()
print(calculated)
```

Let’s execute this code.

Waffle Chart

Creating Waffle Chart:

Example_2: Using pandas calculations for a waffle chart.

After execution, you will get the following output.

```
Republican    4
Democrat      3
Name: party, dtype: int64
```

Waffle Chart

Creating Waffle Chart:

Example_2: Using pandas calculations for a waffle chart.

Now we’ll feed the index and values into the waffle chart separately. The index (on the left) will be our labels, the values (on the right) will be the number of each square to display. If we don’t use list() with the labels part it won’t work.

Waffle Chart

Creating Waffle Chart:

Example_2: Using pandas calculations for a waffle chart.

```
fig = plt.figure( FigureClass=Waffle, rows=2,
values=list(calculated.values),
labels=list(calculated.index) )
fig.show()
```

Now let’s run this example.

Word Cloud

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. Word clouds are widely used for analysing data from social network websites.

Word Cloud

3 reasons you should use word clouds to present your text data:

Word clouds add simplicity and clarity. The most used keywords stand out better in a word cloud

Word clouds are a potent communication tool. They are easy to understand, to be shared and are impactful

Word clouds are visually engaging than a table data

Uplatz

Word Cloud

Who is using word clouds ?

Researchers : for reporting qualitative data
Marketers : for highlighting the needs and pain points of customers
Educators : to support essential issues
Politicians and journalists
social media sites : to collect, analyse and share user sentiments

Word Cloud

For generating word cloud in Python, modules needed are – matplotlib, pandas and wordcloud. We need pandas if we want to read the data from CSV file.
To install **wordcloud**, open command prompt and change the directory to current working directory of python and run any of the following command

pip3 install wordcloud OR
python -m pip install wordcloud OR
pip install --user wordcloud

Word Cloud

Creating Word Cloud: First example.

First of all we need to import the necessary libraries for creating word cloud.

```
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS
```

Stopwords are the English words which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence. For example, the words like the, he, have, has, to, of etc.

Word Cloud

Creating Word Cloud: First example.

Word clouds are commonly used to perform high-level analysis and visualization of text data. Let's try to analyse a short text data which is present in ‘word_cloud.txt’ file. This file is present in current working directory of python.
So read the data/text from this file.

```
text = open('word_cloud.txt', 'r').read()
print(text)
```

Word Cloud

Creating Word Cloud: First example.

Next, let's use the stopwords that we imported from wordcloud. We use the function *set* to remove any redundant stopwords.
And then instantiate the word cloud object.

```
stopwords = set(STOPWORDS)

# instantiate a word cloud object
WCO = WordCloud(
    background_color='white',
    stopwords=stopwords )
```

Word Cloud

Creating Word Cloud: First example.

Generate the word cloud and then display the word cloud.

```
# generate the word cloud
WCO.generate(text)

# display the word cloud
plt.imshow(WCO, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Word Cloud

Creating Word Cloud: Second example.

```
# Libraries
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
#Create the collection of words
text=("Python Python Python Matplotlib Matplotlib
Seaborn Network Plot Violin Chart Pandas
Datascience Wordcloud Spider Radar Parrallel
Alpha Color Brewer Density Scatter Barplot Barplot
Boxplot Violinplot Treemap Stacked Area Chart
Chart Visualization Dataviz Donut Pie Time-Series
Wordcloud Wordcloud Sankey Bubble“)
```

Word Cloud

Creating Word Cloud: Second example.

```
# Create the wordcloud object and generate it
wordcloud = WordCloud(width=480,
height=480).generate(text)

# Display the generated image
plt.imshow(wordcloud, interpolation=‘bicubic')
plt.axis("off")
plt.show()
```

Word Cloud

Creating Word Cloud:

Example_3: Customize word cloud

Maximum and minimum font size: You can control minimum and maximum font size of your wordcloud.

Number of words: It is possible to set a maximum number of words to display on the tagcloud.

Remove some words: You can remove some words that don’t interest you. Just list them

Change background and colour of words:

You can Change the colour of the background and colour of the words.

Let’s see the example.

Heat Map

A heat map (or **heatmap**) is a graphical representation of data where the individual values contained in a matrix are represented as colours. It is really useful to display a general view of numerical data, not to extract specific data point. A **heatmap** contains values representing various shades of the same colour for each value to be plotted. Usually the darker shades of the chart represent higher values than the lighter shade. For a very different value a completely different colour can also be used.

Heat Map

Creating Heat Map: The below example is a two-dimensional plot of values which are mapped to the indices and columns of the chart.

```
import pandas as pd
import matplotlib.pyplot as plt
data = [[2,3,4,1],[6,3,5,2],[6,3,5,4],[3,7,5,4],[2,8,1,5]]
Index= ['1', '2','3','4','5']
Cols = ['C1', 'C2', 'C3','C4']
df = pd.DataFrame(data, index=Index, columns=Cols)
plt.pcolor(df)
plt.show()
```

Here, **pcolor(df)** creates a pseudocolor plot using the values in data frame ‘df’

Heat Map

Creating Heat Map:

Example_2:

```
import matplotlib.pyplot as plt
import numpy as np
#Create an array of the given shape and populate it with random samples from a uniform distribution over (0, 1)
a = np.random.rand(5,8)
plt.imshow(a, cmap='hot', interpolation='nearest')
plt.show()
```

Thank you

Uplatz