DATA SCIENCE

How to do visualization usi from scratch

A step-by-step guide using Matplotlib and Seaborn libr

Sharan Kumar Ravindran

Nov 8, 2020 10 min read

Hands-on Tutorials

Photo by Luke Chesser on Unsplash

Visualization is an important skill set for a data scientist. A good visualization can help in clearly communicating insights identified in the analysis also it is a good technique to better understand the dataset. Our brain is wired in a way that makes it easy for us to extract patterns or trends from visual data as compared to extracting details based on reading or other mear

In this article, I will be covering the visualization c basics using python. Below are the steps to learn basic,

- **Step 1: Importing data**
- **Step 2: Basic visualization using Matplotlib**
- **Step 3: More advanced visualizations, still us**
- **Step 4: Building quick visualizations for data Seaborn**
- **Step 5: Building interactive charts**

By the end of this journey, you would be equippec that is required to build a visualization. Though w covering every single visualization that can be bui learning the concepts behind building a chart and easy for you to build any new charts that are not article.

The scripts and the data used in this article can a git repository here. All data used in this article ca

"Data" folder within the mentioned git repository and the scripts are available in the folders ‘Day23, Day 24, and Day25’.

Importing Data

The first step is to read the required datasets. We can use pandas to read the data. Below is a simple command that c
data from a CSV file

On reading the dataset it is important to transfor
suitable for the visualization we would apply. For
we have sales details at the customer level and if
build a chart that shows the day-wise sales trend
to group the data and aggregate them at the day
a trend chart.

Basic Visualization using Matplotlib

Let us start with some basic visualization. It is be
code ‘fig,ax=plt.subplots()’, where ‘plt.subplots()’ i
will return a tuple with figure and axes objects ar
variables ‘fig’ and ‘ax’ respectively. Without using
can print a chart but by using this you would be a
changes to figure like you would be able to re-siz
depending on how it looks and to save the chart
the ‘ax’ variable here can be used to provide labe
Below is a simple example where I have passed t
and have print it as a chart directly

In the above code at first, the required libraries a
then the ‘plt.subplots()’ function is used to gener

the axes objects and then the data is directly passed as an array to the axes object to print the chart. In the second chart, the axes variable ‘ax’ has taken inputs for labels specific to the x-axis, y-axis, and the title.

Trend Charts

Now, let’s start using some real data and learn ab
interesting charts and about customizing them to
intuitive. As explained, in most real-life use-cases
require some transformation to make it usable fo
is an example where I have used the Netflix data
transformed the data to consolidate the number
shows by year wise. And then I have used the ‘plt
function but I have also added few additional det
chart more intuitive and self-explanatory.

Loading

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

In [2]:
```
netflix_data = pd.read_csv('../Data/netflix_titles.csv')
```

In [3]:
```
n_data = netflix_data.groupby(['release_year', 'type'
n_data.columns = ['release_year', 'type', 'count']
n_data_pivot = n_data.pivot(index='release_year', col
n_data_pivot.fillna(0, inplace = True)
n_data_pivot.head()
```

Out[3]:

type	release_year	Movie	TV Show
0	1925	0.0	1.0
1	1942	2.0	0.0
2	1943	3.0	0.0
3	1944	3.0	0.0
4	1945	3.0	0.0

VisualizationArticle_02.ipynb hosted with ❤ by GitHub

There are a few more customization that can be c
chart like creating a dual-axis. In the above case,
difference between the number of movies and TV
data appears OK, if there has been a huge differe
then the chart will not be very clear in those case
of dual-axis so that the attribute with smaller val
scaled in line with the other one.

Scatter Plots

We can also make use of the Scatter Plot, to bring
relationship between the variables that we are pl
helps in bringing the correlation between variable

happens to one attribute when the other attribute is increasing/decreasing.

More advanced visualizations, still using Matplotlib

Once you are comfortable with the simple trend- covered so far you are ready to move to slightly m charts and functionalities to better customizatio

Bar Charts

The Bar Charts help us to compare multiple value by plotting them side-to-side. There are different Charts,

- Vertical Bar Chart
- Horizontal Bar Chart
- Stacked Bar Chart

Below is an example of a Bar Chart, there are a nu customization added to this plot. They are,

- Axis labels and title are added
- Font size has been provided
- Figure size is provided as well (default chart v smaller and cluttered)
- A function is used to generate and add values of the bars to help the viewers get the actual

Loading

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [4]:
```
# Data from: https://www.kaggle.com/shivamb/netflix-shows
# With transformations
netflix_data = pd.read_csv('../Data/netflix_titles.csv')
netflix_data['year_added'] = netflix_data['date
n_data_added = netflix_data.groupby(['year_adde
n_data_added.columns = ['year_added', 'type', '
n_data_added = n_data_added.pivot(index='year_a
n_data_added.fillna(0, inplace = True)
n_data_added.head()
```

Out[4]:

type	year_added	Movie	TV Show
0	2008	1.0	1.0
1	2009	2.0	0.0
2	2010	1.0	0.0
3	2011	13.0	0.0
4	2012	4.0	3.0

VisualizationArticle_03.ipynb hosted with ❤ by GitHub

Horizontal and Stacked Bar Chart

Vertical Bar charts are most common but we can
the horizontal bar charts especially when the dat
name and it is very difficult to print them below a
case of the stacked bar chart, the bars will be sta
another within a category. below is the example c
horizontal and stacked bar charts. The below cod
customization to the chart color.

Loading

In [6]:

```
data = pd.DataFrame(dict(Year = n_data_added['year_added'],
                         Movie = n_data_added['Movie'], TVshow=n_data_added[

ind = np.arange(len(data))
width = 0.4

fig1, ax1 = plt.subplots()
ax1.barh(ind, data.Movie, width, color='red', label='Movies')
ax1.barh(ind + width, data.TVshow, width, color='green', label='TV Shows')

ax1.set(yticks=ind + width, yticklabels=data.Year, y

ax1.set_ylabel('Year added on Netflex')
ax1.set_xlabel('Count of Number of Movies/TV Shows')
ax1.set_title('Number of movies and TV Shows by year

ax1.legend()
fig1.set_size_inches(10, 10)
plt.show()
```

VisualizationArticle_04.ipynb hosted with ❤ by GitHub

Pie and Donut Chart

Pie charts are useful to show the proportion of di
in the data and these pie charts can easily be mo
chart by covering the center part of the pie chart
re-aligning the text/values to suit the donut chart
example where I have implemented the pie chart
it into a donut chart

Loading

In [4]:

```
# Small modification to data to suit for the Pie Chart
n_data_added['Total'] = n_data_added['Movie'] + n_data_added['TV Show']
# To select last few rows only
n_data_added_flt = n_data_added[-4:]
n_data_added_flt = n_data_added_flt.reset_index(drop=True) # For resetting
n_data_added_flt.head()
```

Out[4]:

	type	year_added	Movie	TV Show	Total
0		2017	913.0	387.0	1300.0
1		2018	1290.0	492.0	1782.0
2		2019	1546.0	803.0	2349.0
3		2020	147.0	37.0	184.0

In [5]:

```
# Pie chart
labels = n_data_added_flt['year_added']
sizes = n_data_added_flt['Total']
explode = (0, 0.1, 0, 0)  # only "explode" the 2nd sl

fig2, ax2 = plt.subplots()
ax2.pie(sizes, explode=explode, labels=labels, autopc
        shadow=True, startangle=90)
```

VisualizationArticle_05.ipynb hosted with ❤ by GitHub

Why is it important to learn Matplotlib?

Matplotlib is a very important visualization library many other visualization libraries in python are de matplotlib. Some of the advantages/benefits of le are,

- It is easy to learn
- It is efficient
- It allows a lot of customizations hence possib any kind of visuals
- Libraries like Seaborn are built on top of Matp

I have covered only the most essential visualization in Matplotlib but the important factor is by practicing these charts you would have acquired the knowledge for building much more visualization. Matplotlib supports a number of visualization here is the link to the gallery of all supported charts.

Building quick visualizations for data using Seaborn

We have covered a variety of visualization using th library. I am not sure if you have noticed, though high customization it involves a lot of coding and time-consuming especially when you are working analysis and would want to make a few quick plo the data better and make the decisions faster. Th offered by Seaborn library, here are some benefit seaborn library,

- Default themes are still attractive
- Simple and quick to build visualizations espe analysis
- Its declarative API allows us to just focus on the charts

There are few downsides too like it doesn't offer customization and it could lead to memory issue we work on large datasets. But still, the benefit o disadvantages.

Visualizations with just one line code

Below are some simple visualizations that are implemented with just a single code using the seaborn library.

Loading

Preparing Dataset

In [2]:
```
# Data from: https://www.kaggle.com/andrewmvd/heart-failure-clinical-data
heart_failure_data = pd.read_csv('../Data/heart_failu
heart_failure_data.head()
```

Out[2]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejec
0	75	0	582	0	
1	55	0	7861	0	
2	65	0	146	0	
3	50	1	111	0	
4	65	1	160	1	

In [3]:
```
# transforming data for visualization
agg_data = heart_failure_data.groupby(['age', 'DEATH_
agg_data.columns = ['Age', 'Death_Event', 'Count']
agg_data.head()
```

VisualizationArticle_06.ipynb hosted with ❤ by GitHub

As shown in the above snapshot the visualization just a single line of code and they look quite pres The Seaborn library is widely used in the data ana can build charts quickly with ease and with minin make the charts presentable. Visualization is key as they help in bringing out patterns in the data a library fits apt for the purpose.

Heatmaps

Heatmaps are another interesting visualization that is widely used on time-series data to bring out the seasonalities and other patterns in the dataset. However to build a heatmap we need to transform the data into a specific format to support heatmap plotting. Below is a sample code to transform the data to suit the heatmap plot and seaborn library used to build the heatmap

Loading

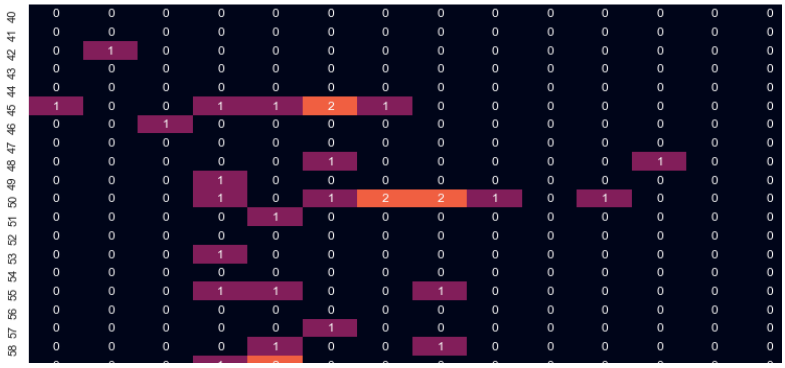
Heatmap

In [ ]:

```
# Data for heatmap
agg_data2 = heart_failure_data.groupby(['age', 'eject
agg_data2.columns = ['Age', 'ejection_fraction', 'Dea
agg_data2_p = agg_data2.pivot(index='Age', columns='e
agg_data2_p.fillna(0, inplace=True)
agg_data2_p.head(20)
```

In [9]:

```
plt.subplots(figsize=(20,15))
sns.heatmap(agg_data2_p, annot=True)
plt.show()
```



VisualizationArticle_07.ipynb hosted with ❤ by GitHub

Pair Plot – my favorite functionality of Seaborn

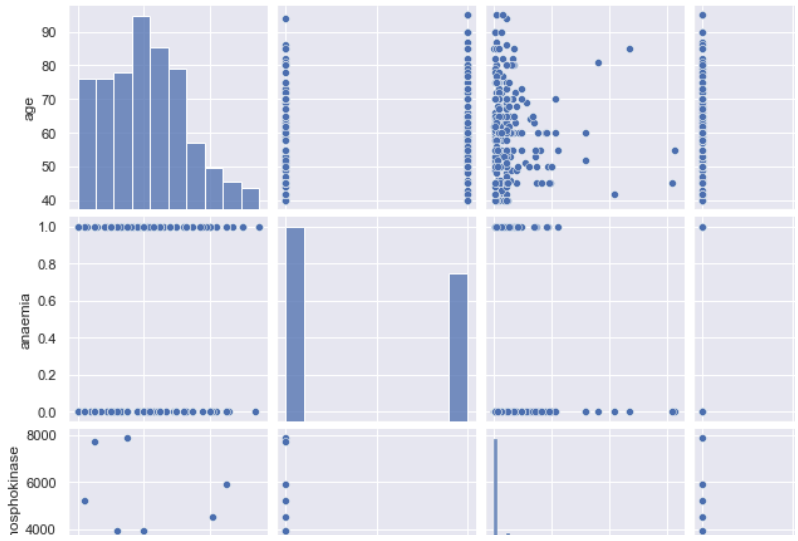
I consider the pair plot as one of the best feature library. It helps in comparison of each attribute in every other attribute through visuals and again in code. Below is a sample code to build pair plots.

plot might not be feasible when the dataset we are working on has a large number of columns. In those cases, the pair-plots can be used to analyze the relationship between a specific set of attributes alone.

Loading

Pair Plot

```
In [ ]: subset = heart_failure_data.iloc[:,[0,1,2,3,4]]
        subset.head()

In [11]: sns.pairplot(subset)

Out[11]: <seaborn.axisgrid.PairGrid at 0x249c7b16c10>
```



VisualizationArticle_08.ipynb hosted with ❤ by GitHub

Building interactive charts

While working on Data Science projects sometime requirement to share some visualization with the Dashboarding tools are widely used for this purpo there is an interesting pattern that you have notic performing data analysis and would like to share

user. If they are shared as an image there might not be much the business user can do but if they are shared as an interactive chart then it gives the business user power to look into the granular details by zooming in or out or use other functionality to interact with the chart. Below is an example where we are creating an HTML file as an output which includes the visualization that can be shared with any other user and they can be simply opene
browser.

Loading

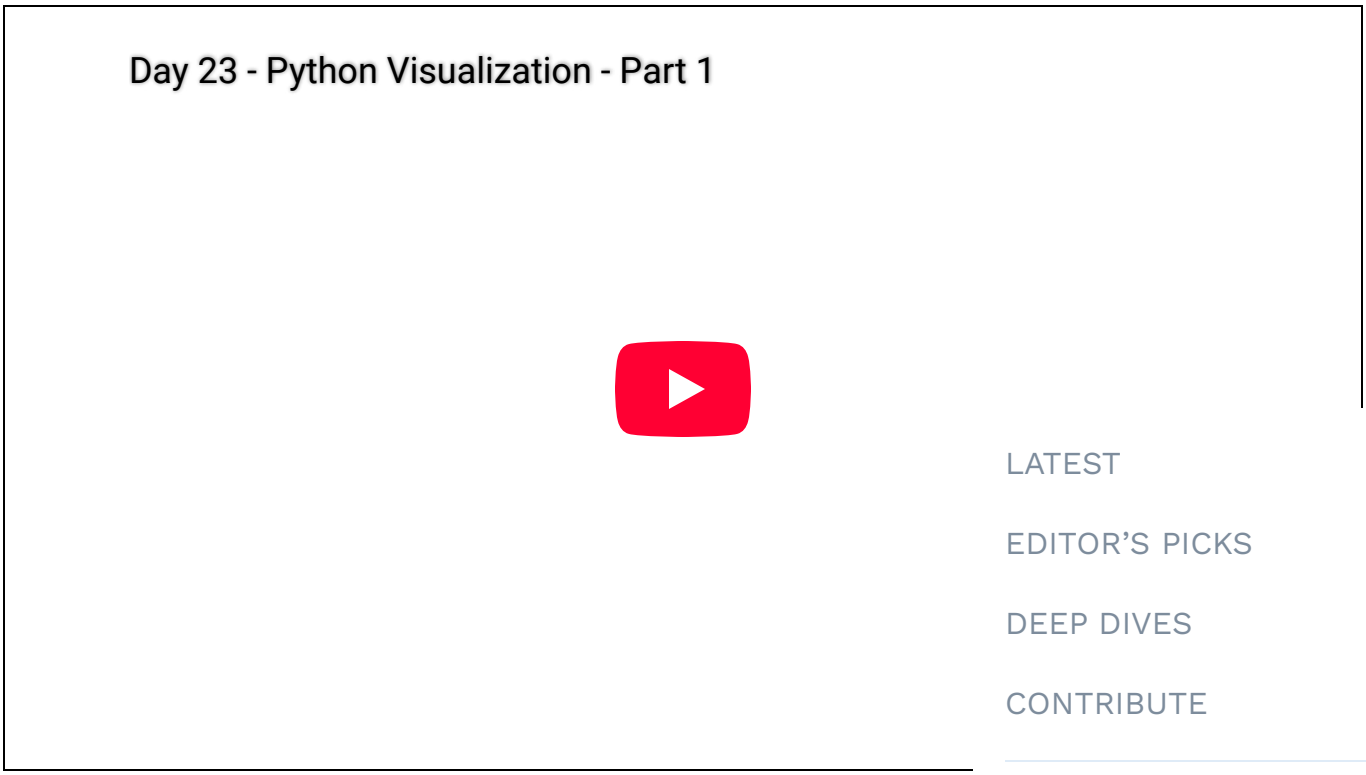
Interactive Plots

In [12]:

```
from bokeh.plotting import figure, output_file, show

p = figure(title = "Age Vs Ejection Fraction")
p.xaxis.axis_label = 'Age'
p.yaxis.axis_label = 'Ejection Fraction'

p.circle(agg_data2["Age"], agg_data2["ejection_fracti
         fill_alpha=0.2, size=10)

output_file("test.html", title="Example")

show(p)
```

VisualizationArticle_09.ipynb hosted with ❤ by GitHub

If you are keen to learn about visualizations using
check out my playlist below. It includes three vide
tutorial length of just over one hour.



About Me

I am a Data Science professional with over 10 yea and I have authored 2 books in data science, they sale here. I have a YouTube channel where I teach various data science concepts. If interested, subs channel below.

Data Science with Sharan

WRITTEN BY

Sharan Kumar Ravindran

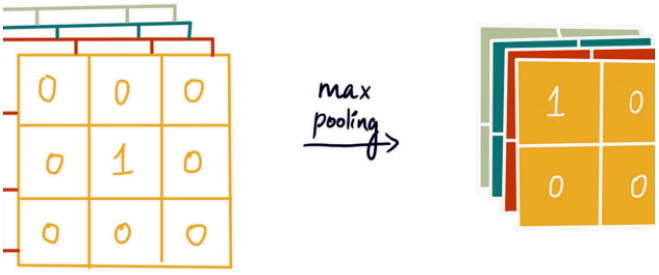
See all from Sharan Kumar Ravindran

Topics:

Data Science | Hands On Tutorials | Python | Visualization

Share this article:

Related Articles



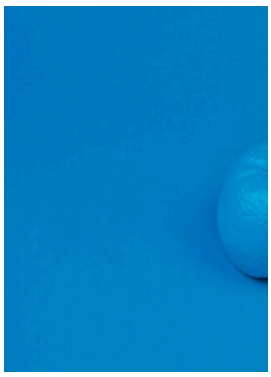
ARTIFICIAL INTELLIGENCE

Implementing Convolutional Neural Networks in TensorFlow

Step-by-step code guide to building a Convolutional Neural Network

Shreya Rao

August 20, 2024 6 min read



DATA SCIENCE

Hands-on Tim Detection usi with Python

Here's how to us detect signals wi lines of...

Piero Paialunga

August 21, 2024 1

DATA SCIENCE

Solving a Constrained Project Scheduling Problem with Quantum Annealing

Solving the resource constrained project scheduling problem (RCPSP) with D-Wave's hybrid constrained quadratic model (CQM)

Luis Fernando PÉREZ ARMAS, Ph.D.

August 20, 2024 29 min read

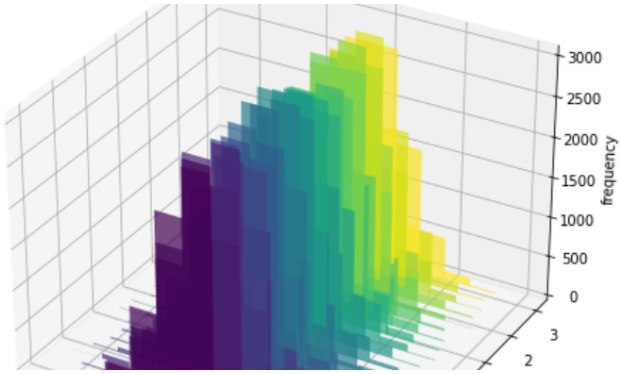
DATA SCIENCE

Back To Basic Regression ar

An illustrated gui learning concept

Shreya Rao

February 3, 2023



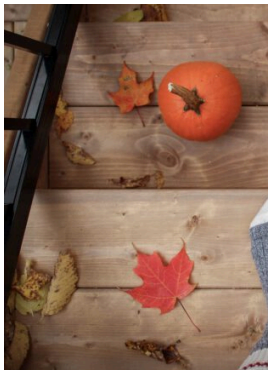
DATA SCIENCE

Must-Know in Statistics: The Bivariate Normal Projection Explained

Derivation and practical examples of this powerful concept

Luigi Battistoni

August 14, 2024 7 min read



DATA SCIENCE

Our Columns

Columns on TDS collections of po or category...

TDS Editors

November 14, 2020

DATA SCIENCE

Optimizing Marketing Campaigns with Budgeted Multi-Armed Bandits

With demos, our new solution, and a video

Vadim Arzamasov

August 16, 2024 10 min read