

[Open in app](#)

Following ▾

571K Followers



Data Visualization: Say it with Charts in Python

A complete hands-on guide to the best practices and concepts of visualization in python using matplotlib, pyplot, and seaborn



Angel Das · Apr 3, 2020 · 10 min read



Photo by [William Iven](#) on [Unsplash](#)

[Open in app](#)

detect significant events like gaining insights into customer behavior, monitoring key performance indicators over time and gaining market and sales intelligence to adapt quickly to their changing and dynamic business. Visualization forms the backbone of any BI, hence an analyst or data scientist needs to understand the nuances of visualization and how it plays an important role in communicating insights and findings effectively.

We will lay down some business questions using real-world data and get an understanding of how to create presentations in Python that are **powerful**, **effective** and **insightful**, the **PEI framework** of visualization.

Here are the five things we will cover in this article:

- Importance of visualization in the analytics industry
- How to decide on which chart to use
- Introduction and background to matplotlib and seaborn in python
- A series of graphs and visualization using python to answer relevant questions from a real-world data
- Best practices of storytelling using visualization

Why data visualization is important?

Data and analytics are changing the basis of competition. Leading companies are using their capabilities not only to improve their core operations but to launch entirely new business models. Topline KPIs (Key Performing Indicators) are becoming the need of the hour with CXOs keeping a constant track of the ever dynamic market thus enabling them to make informed decisions at scale. Dashboards are becoming popular among the stakeholders and with the large influx of data collected over numerous data sources, visualization forms the key to analyzing this data. In short four main reasons to build visuals involve:

1. For exploratory data analysis popularly referred to as the data EDA
2. Communicate topline findings clearly to different stakeholders

[Open in app](#)

Using visualization to support findings, insights, and recommendations

Before we start let us take a look at the fundamentals of charts in visualization. Charts generally fall under two broad categories:

- **Data charts**, also called quantitative charts, depict numbers graphically to make a point. They include pie charts, bar graphs, stacked bar graphs, scatter plots, line charts, etc.
- **Concept charts**, also called non-quantitative charts, use words and images. Concept charts describe a situation, such as interaction, interrelationship, leverage or forces at work. Some common examples are process flow charts, gantt charts, matrix, etc.

Data charts are the most widely used chart type in the analytics industry. From dashboards, regular reports to presentations they play an important role in communicating our analysis in a way that stakeholders can understand. Concept chart plays a major role in consulting and strategy outlining, especially in scenarios where stepwise outlining of strategy is important or scenarios that require a SWOT analysis (Strength, Weakness, Opportunities, Threat).

Want to read more about the current BI trends in the market?

The dawn of modern analytics — A look into some of the recent trends of Business Intelligence...

A survey based approach to identify the BI tools every Data Scientist should know







towardsdatascience.com

The chart selection framework

While working with platforms like Excel, Python, R, Tableau, R Shiny, Power BI or QlikSense, users are exposed to multiple chart layouts that are attractive and eye-catching. However, in the world of analytics more than creating an attractive visualization, it is important to create a visualization that is **effective and impactful**. Below is the selection framework that forms the basis of any chart selection.

Open in app



Plot		Continuous	Continuous	- Trend analysis, change in KPI over time	- How sales of a company varied over a period of time?
Bar Graph		Categorical	Continuous	- How Y (can be any performance indicator) varies across different categories?	- How sales in 2019 varied for different mobile phone brands? i.e. mobile phone brand is the category and sales is the KPI
Stack Bar Graph		Categorical	Continuous	- Relative comparison of multiple categories within a category	- Comparison of revenue generated by Apple, Samsung & Xiaomi across different products like mobile phone, laptops, television, and headsets
Box Plot		Continuous		- Outlier detection - Analysing data distribution across Median and Inter Quartile Range	- How different sales figures across a year is distributed?
Pie Chart		Categorical & Continuous		- Relative comparison of different categories for one single entity in terms of proportion/percentages	- What percentage of Sales in 2019 is constituted by different products under Apple?
Histogram Plot		Continuous	-	- How distribution of values of x varies across different range buckets?	- Distribution of income across income buckets for developing countries

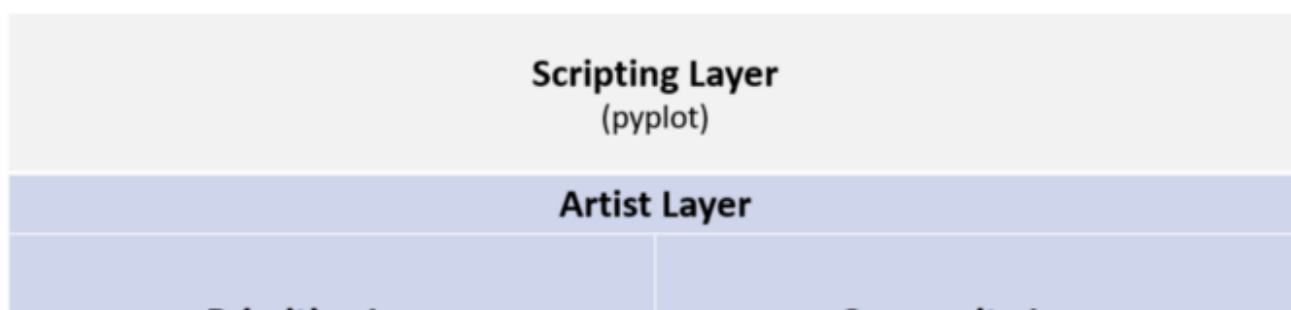
The Chart Selection Framework

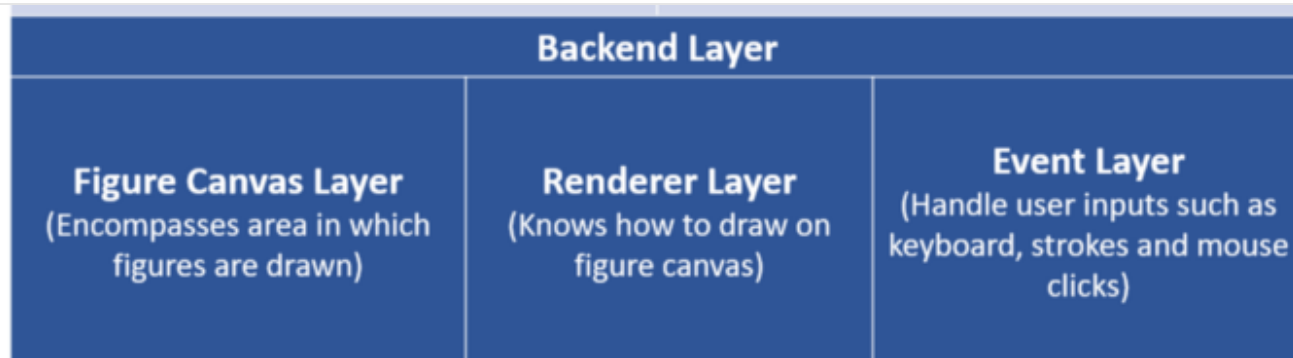
Introduction to Data Visualization in Python

Matplotlib History and Architecture

Matplotlib is one of the most widely used data visualization libraries in Python. It was created by John Hunter, who was a neurobiologist and was working on analyzing Electrocorticography signals. His team had access to a licensed version of proprietary software for the analysis and was able to use it in turns only. To avoid this limitation, John developed a MATLAB based version which in the later stages was equipped with a scripting interface for quick and easy generation of graphics, currently known as matplotlib.

Matplotlib operates on a three-layer architecture system, The Backend Layer, Artist Layer & Scripting Layer.



[Open in app](#)

Matplotlib Architecture

Let's look at the code below:

```
# - - - - - Backend Layer, Figure Canvas Layer:
# Encompasses area in which figures are drawn - - - - -

from matplotlib.backends.backend_agg import FigureCanvasAgg as
FigureCanvas

#- - - - - -From Artist layer, importing Artist Figure:
# Artist layer knows how to use a renderer to draw an object-----
#-----
from matplotlib.figure import Figure
fig=Figure()
canvas=FigureCanvas(fig) #-----passing object of artist layer to
# backend layer (Figure Canvas)

#- - - - - -Using numpy to create random variables-----
#-----
import numpy as np
x=np.random.rand(100)

#- - - - - -Plotting a histogram-----
ax=fig.add_subplot(111)
ax.hist(x,100)
ax.set_title('Normal Distribution')
fig.savefig('matplotlib_histogram.png')
```

We can see that to plot a histogram of random numbers using a combination of backend and artist layer, we need to work out multiple lines of the code snippet. To reduce this effort, Matplotlib introduced the scripting layer called **Pyplot**.

Open in app



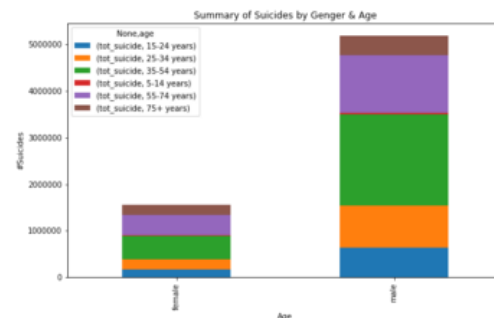
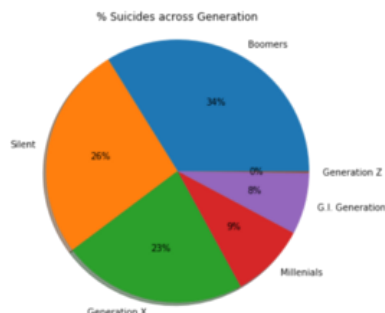
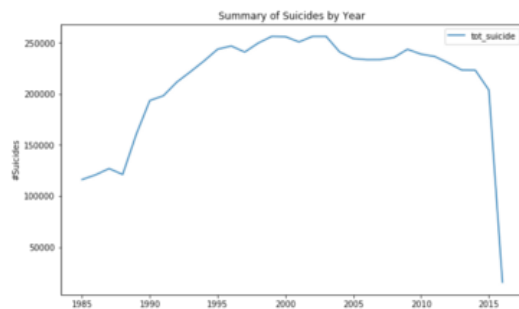
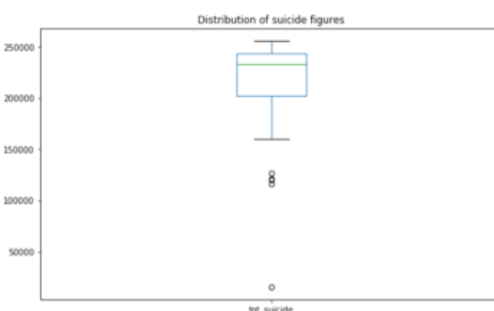
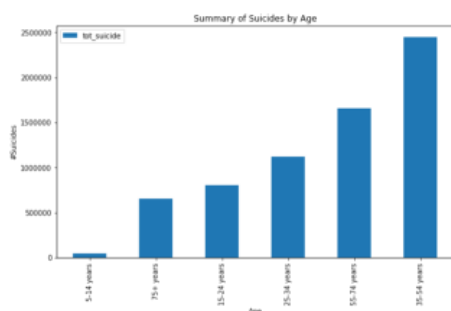
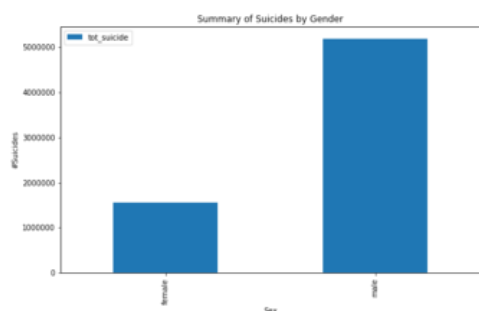
```
plt.ylabel( '#suicides' );

#- - - - -Proportion of Suicide by Generation- - - - -
generation_summary=suicide_data.groupby('generation').agg(tot_suicide=
('suicides_no','sum')).sort_values(by='tot_suicide',ascending=False).
reset_index()
generation_summary.head(10)

#- - - - -Plotting pie chart- - - - -
fig1, ax1 = plt.subplots();
fig1.set_size_inches(8,6)

ax1.pie(generation_summary['tot_suicide'],labels=generation_summary['
generation'],autopct='%1.0f%%',shadow=True);
ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as
a circle.
plt.title('% Suicides across Generation')
plt.show();

#- - - - -Histogram Plot- - - - -
year_summary.plot(kind='box', figsize=(10,6)); #-----To plot
histogram change kind='box' to kind='hist'
plt.title('Distribution of suicide figures');
```



Visualization of Suicide data using Matplotlib and Pyplot


[Open in app](#)

```
ax=sns.barplot(x=gender_age_summary['sex'],y=gender_age_summary['tot_
suicide'],hue=gender_age_summary['age']);
ax.set(xlabel='Gender', ylabel='#Suicides');
plt.show()
```

```
gender_age_summary1=suicide_data.groupby(['sex','age']).agg(tot_suici
de=('suicides_no','sum')).unstack()
gender_age_summary1.head()
```

```
#- - - - -Stack bar-----
```

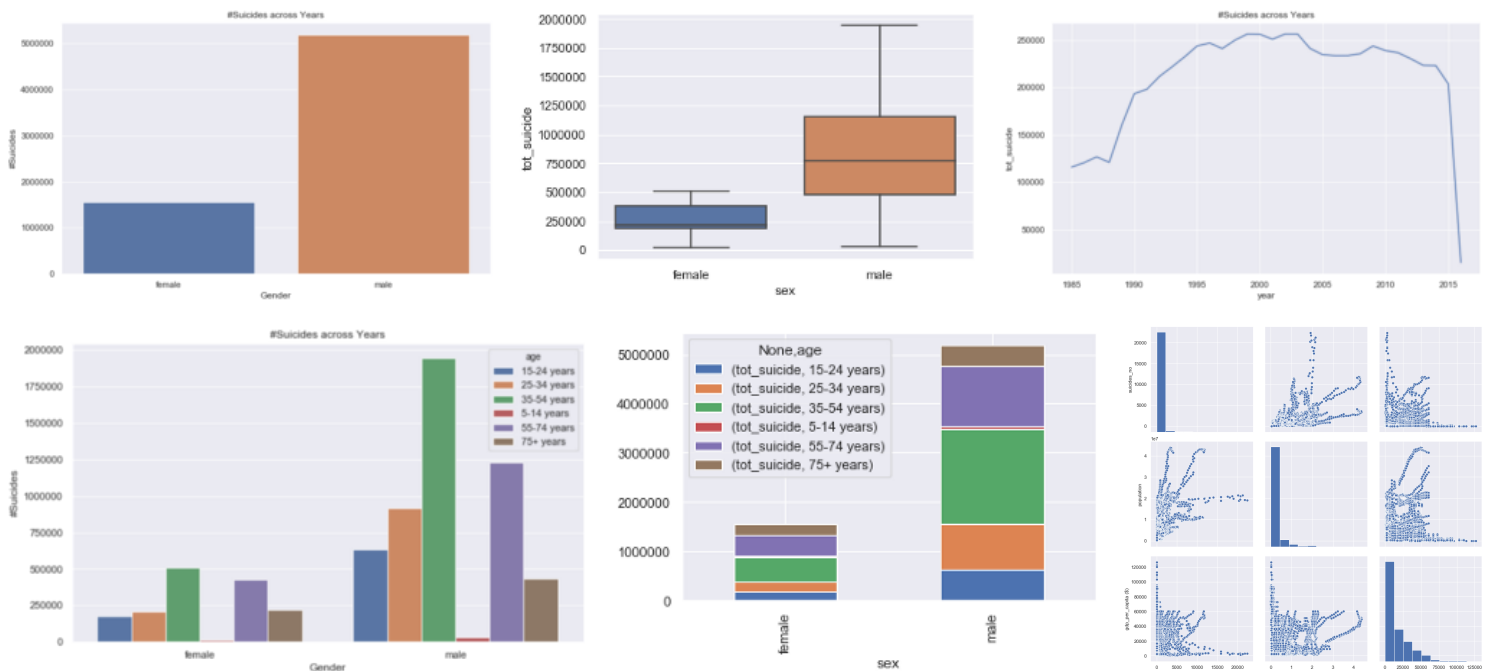
```
sns.set()
gender_age_summary1.plot(kind='bar', stacked=True)
```

```
#- - - - -Checking correlation between suicide,
population and gdp per capita
```

```
sns.pairplot(suicide_data[['suicides_no', 'population',
'gdp_per_capita ($)']], size=4, aspect=1);
```

```
#- - - - -Boxplot-----
```

```
sns.boxplot(gender_age_summary['sex'],gender_age_summary['tot_suicide
']);
```



Visualization of Suicide data using Seaborn

Best practices of storytelling using visualization

[Open in app](#)

1. Define the problem statement clearly and use a top-down approach to break it into multiple hypotheses. In the above case study, the problem statement involves understanding the suicide rates across different countries and factors influencing them. So before jumping into a visualization list down all possible factors that might affect suicide rates, e.g. age, gender, GDP, population, growth, generation, etc.
2. Identify the key factors that might add value to your story. Think of how the factors can be interconnected to get a bigger picture of what is happening. e.g. instead of looking at suicides by age and gender separately try comparing the suicide rate across both the groups. You can derive quality insights like, Male individuals with age between 18 and 25 have higher suicide rates
3. Once you decide on the factors ensure you use the chart selection framework to decide which graph suits the best
4. Ensure consistency in formatting across all your graphs which includes color, text size, title size, axis spacing, legend position and alignment of chart objects
5. Below every visualization call out a few findings; This will help you stitch together a story for your stakeholder

New to the Data Science Industry? Here are a few tips and tricks for you to start!

Are your coding skills good enough for a Data Science job?

5 coding sniffs you must know if you are working in the Data Science industry

towardsdatascience.com

I can't teach you Data Science in 10 days

A Case Study approach towards understanding Entities & Requirements in Data Science Space

towardsdatascience.com

[Open in app](#)

About the Author: Advanced analytics professional and management consultant helping companies find solutions for diverse problems through a mix of business, technology, and math on organizational data. A Data Science enthusiast, here to share, learn and contribute; You can connect with me on [Linked](#) and [Twitter](#);

Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. [Take a look.](#)

Get this newsletter

Emails will be sent to shubhangi.waghmare7@gmail.com.
[Not you?](#)

[Data Science](#)[Machine Learning](#)[Programming](#)[Python](#)[Coding](#)[About](#) [Help](#) [Legal](#)

Get the Medium app

