Kaushik Katari

Aug 11, 2020 · 8 min read · Member-only

Seaborn: Python

Seaborn is a library in Python predominantly used for making statistical graphics.

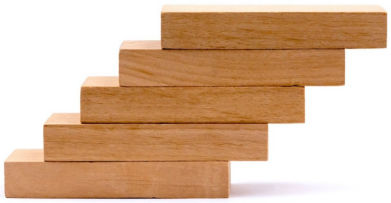


Photo by Volodymyr Hryshchenko on Unsplash

Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

Kaushik Katari

353 Followers

Software Engineer | Python | Machine Learning | Writer

Follow

One has to be familiar with **Numpy** and **Matplotlib** and **Pandas** to learn about Seaborn.

Seaborn offers the following functionalities:

1. Dataset oriented API to determine the relationship between variables.
2. Automatic estimation and plotting of linear regression plots.
3. It supports high-level abstractions for multi-plot grids.
4. Visualizing univariate and bivariate distribution. (U, B)

These are only some of the functionalities offered by Seaborn, there are many more of them, and we can explore all of them here.

To initialize the Seaborn library, the command used is:

```
import seaborn as sns
```

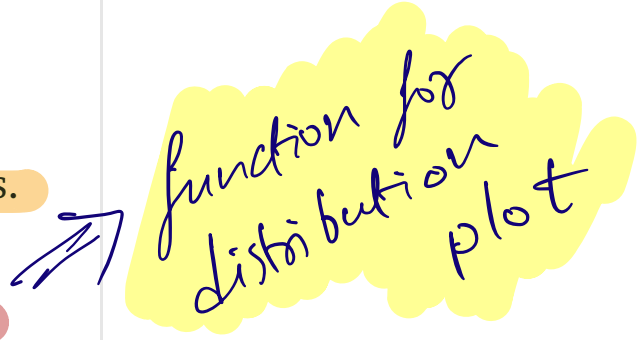
Using Seaborn we can plot wide varieties of plots like:

1. Distribution Plots
2. Pie Chart & Bar Chart
3. Scatter Plots
4. Pair Plots
5. Heat maps

> For this entirety of the article, we are using the dataset of Google Playstore downloaded from Kaggle.

1. Distribution Plots

We can compare the distribution plot in Seaborn to histograms in Matplotlib. They both offer pretty similar functionalities. Instead of frequency plots in the histogram, here we'll plot an approximate probability density across the y-axis.



We will be using **sns.distplot()** in

the code to plot distribution graphs.

Before going further, first, let's access our dataset,

getting the dataset from online and importing libraries, pandas

Accessing Dataset from our system

import matplotlib.pyplot and seaborn

The dataset looks like this,

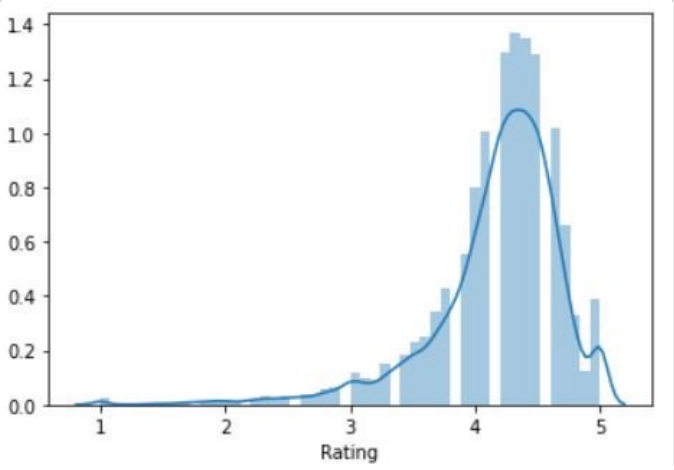
Google Play Store Dataset from Kaggle

Now, let's see how distribution plot looks like if we plot for 'Rating'

column from the above dataset,

Code for Rating column distribution plot

The Distribution Plot looks like this for Rating's column,



Distribution Plot — Rating

Here, the curve(**KDE**) that appears drawn over the distribution graph is

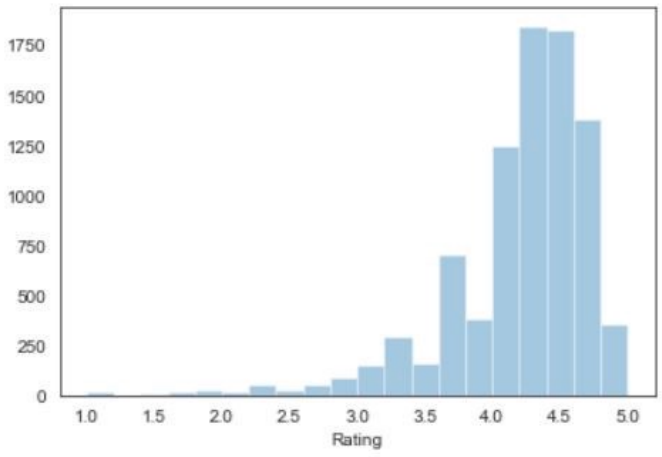
the approximate probability density curve.

Similar to the histograms in the matplotlib, in distribution too, we can change the number of bins and make the graph more understandable.

We just have to add the number of bins in the code,

```
#Change the number of bins
sns.distplot(inp1.Rating,
bins=20, kde = False)
plt.show()
```

Now, the graph looks like this,

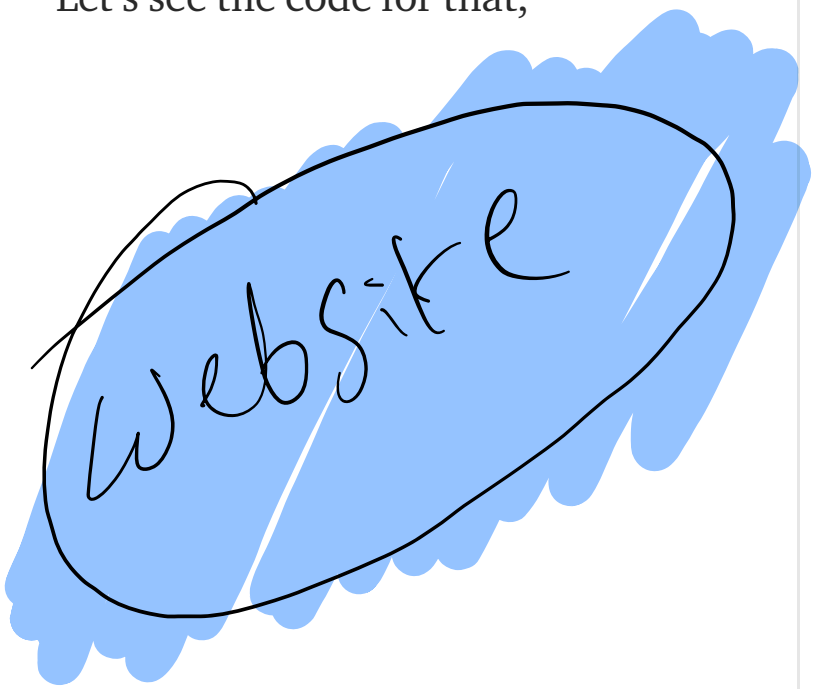


Distribution Plot with specific number of bins

In the above graph, there is no

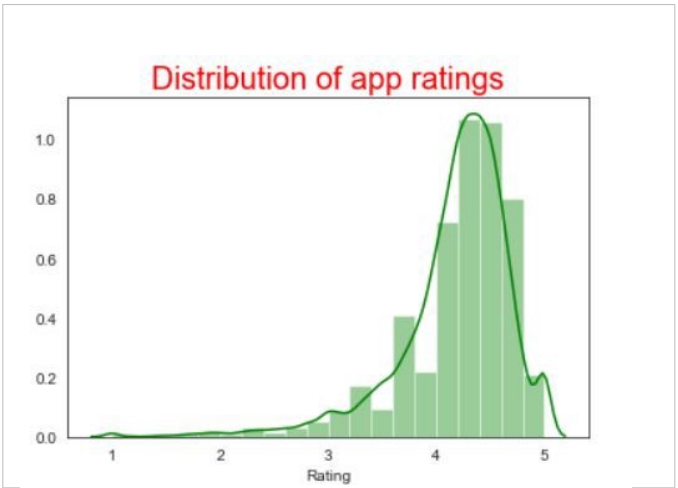
probability density curve. To remove the curve, we just have to write **'kde = False'** in the code.

We can also provide the title and color of the bins similar to matplotlib to the distribution plots. Let's see the code for that,



. . .

The distribution graph, for the same column rating, looks like this:



Distributon plot with Title

Styling the Seaborn graphs

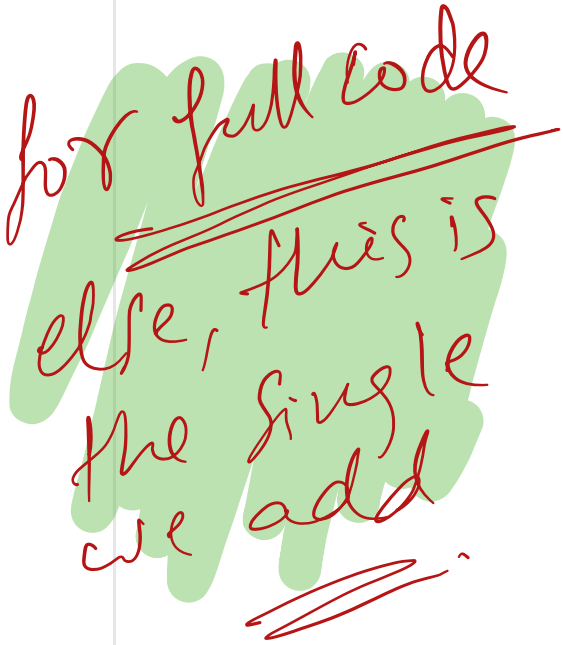
One of the biggest advantages of using Seaborn is, it offers a wide range of default styling options to our graphs.

These are the default styles offered by Seaborn.

```
'Solarize_Light2',
 '_classic_test_patch',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn',
 'seaborn-bright',
 'seaborn-colorblind',
 'seaborn-dark',
 'seaborn-dark-palette',
 'seaborn-darkgrid',
 'seaborn-deep',
```

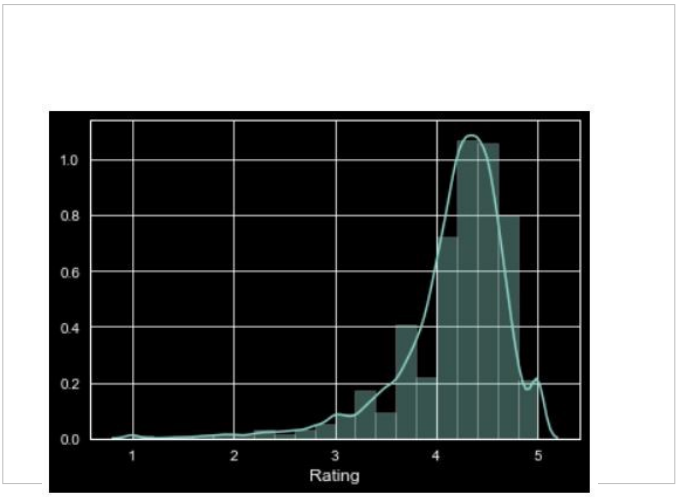
```
    'seaborn-muted',
    'seaborn-notebook',
    'seaborn-paper',
    'seaborn-pastel',
    'seaborn-poster',
    'seaborn-talk',
    'seaborn-ticks',
    'seaborn-white',
    'seaborn-whitegrid',
    'tableau-colorblind10'
```

We just have to write one line of code to incorporate these styles into our graph.




```
plt.style.use("dark_background")
```

After applying the dark background to our graph, the distribution plot looks like this,



Distribution plot with dark background

2. Pie Chart & Bar Chart

Pie Chart is generally used to analyze the data on how a numeric variable changes across different categories.

In the dataset we are using, we'll analyze how the top 4 categories in the Content Rating column is performing.

First, we'll do some data cleaning/mining to the Content rating column and check what are the categories in there.

Now, the categories list will be,

Content rating count

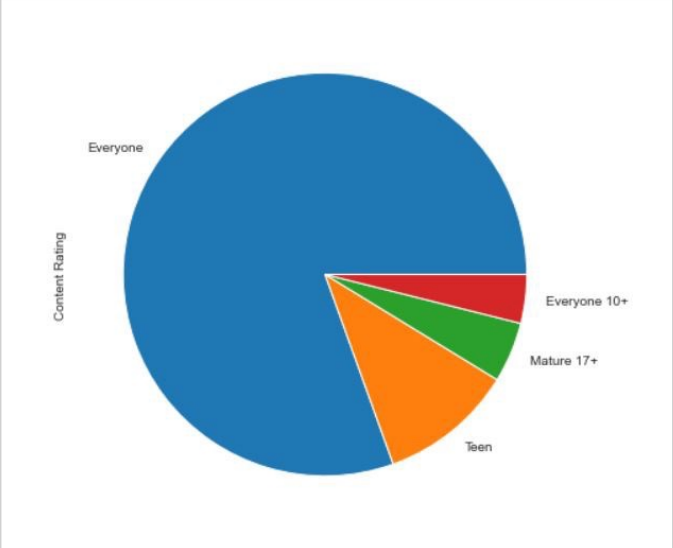
As per the above output, since the count of “Adults only 18+” and “Unrated” are significantly less compared to the others, we’ll drop those categories from the Content Rating and update the dataset.

The categories present in the “Content Rating” column after updating the sheet are,

Content Rating count after updating the dataset

Now, let’s plot Pie Chart for the categories present in the Content Rating column.

The Pie Chart for the above code looks like the following,



Pie Chart for Content Rating

From the above Pie diagram, we cannot correctly infer whether “Everyone 10+” and “Mature 17+”. It is very difficult to assess the difference between those two categories when their values are somewhat similar to each other.

We can overcome this situation by plotting the above data in **Bar chart.**

Now, the bar Chart looks like the following,

Bar Chart for Content rating column

Similar to Pie Chart, we can customize our Bar Graph too, with different Colors of Bars, the title of the chart, etc.

3. Scatter Plots

Up until now, we have been dealing with only a single numeric column from the dataset, like Rating,

Reviews or Size, etc. But, what if we have to infer a relationship between two numeric columns, say “Rating and Size” or “Rating and Reviews”.

Scatter Plot is used when we want to plot the relationship between any two numeric columns from a dataset. These plots are the most powerful visualization tools that are being used in the field of machine learning.

Let’s see how the scatter plot looks like for two numeric columns in the dataset “Rating” & “Size”. First, we’ll plot the graph using matplotlib after that we’ll see how it looks like in seaborn.

Scatter Plot using matplotlib

```
#import all the necessary
libraries


#Plotting the scatter plot
plt.scatter(pstore.Size,
pstore.Rating)
plt.show()
```

Now, the plot looks like this

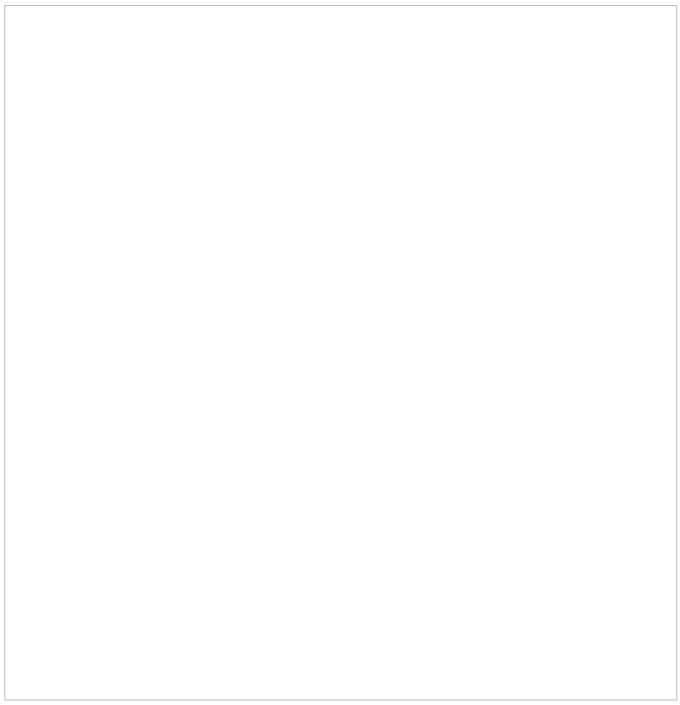
Scatter Plot using Matplotlib

Scatter Plot using Seaborn

We will be using **sns.joinplot()** in the code for scatter plot along with the histogram.

sns.scatterplot() in the code for only scatter plots.

The Scatter plot for the above code looks like,



Scatter Plot using Seaborn

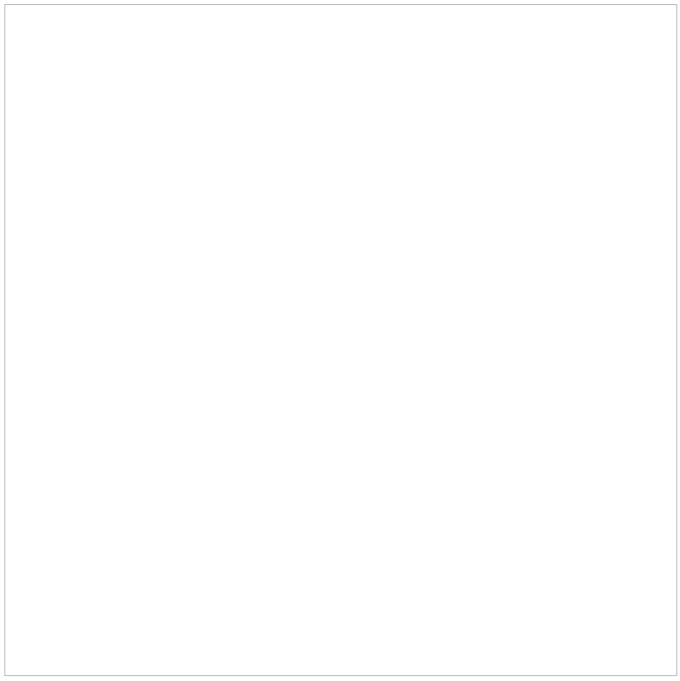
The main advantage of using a scatter plot in seaborn is, we'll get both the scatter plot and the histograms in the graph.

If we want to see only the scatter plot instead of "**jointplot**" in the code, just change it with "**scatterplot**"

Regression Plot

Regression plots create a regression line between 2 numerical parameters in the jointplot(scatterplot) and help to visualize their linear relationships.

The graph looks like the following,



Regression Plot using jointplot in Seaborn

From the above graph, we can infer that there is a steady increase in the Rating if the Price of the apps increases.

4. Pair Plots

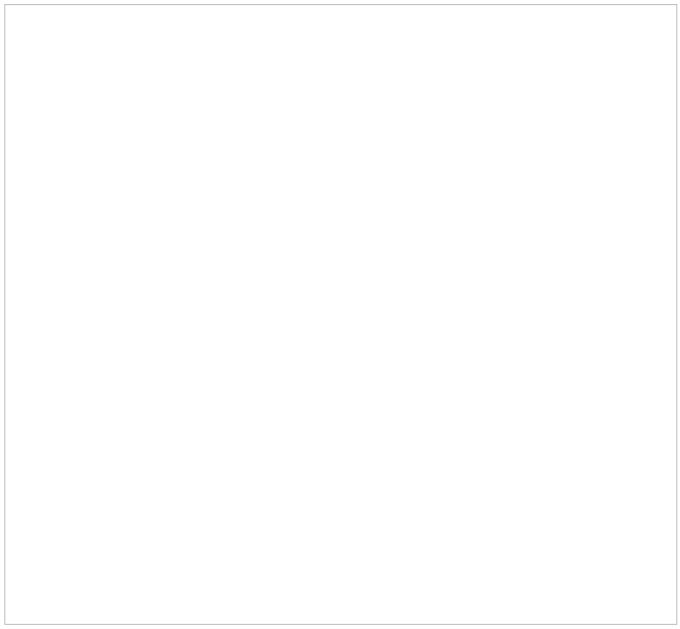
Pair Plots are used when we want to see the relationship pattern among more than 3 different numeric variables. For example, let's say we want to see how a company's sales are affected by three different factors, in that case, pair plots will be very helpful.

Let's create a pair plot for Reviews, Size, Price, and Rating columns

from of dataset.

We will be using **sns.pairplot()** in the code to plot multiple scatter plots at a time.

The output graph for the above graphs looks like this,



Pair Plot using Seaborn

- For the non-diagonal views, the graph will be a **scatter plot** between 2 numeric variables
- For the diagonal views, it plots a **histogram** since both the axis(x,y) is the same.

5. Heatmaps

The heatmap represents the data in a 2-dimensional form. The ultimate goal of the heatmap is to show the summary of information in a colored graph. It utilizes the concept of using colors and color intensities to visualize a range of values.

Most of us would have seen the following type of graphics in a

football match,

Heatmap of a player in football

Heatmaps in Seaborn create exactly these types of graphs.

We'll be using **sns.heatmap()** to plot the visualization.

When you have data as the following we can create a heatmap.

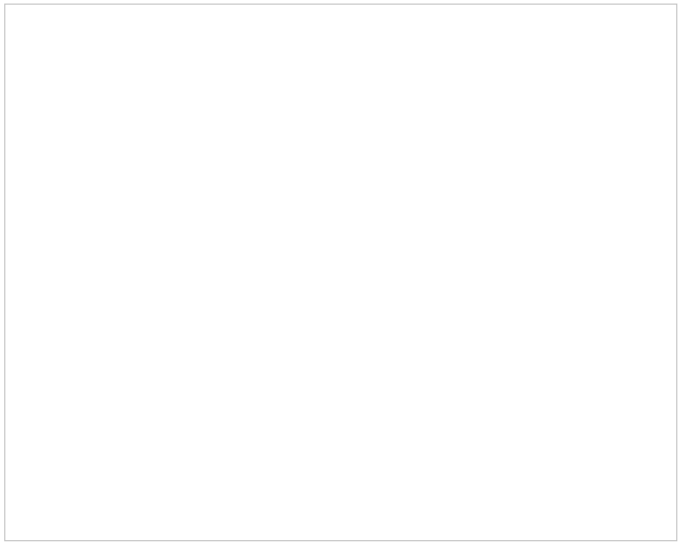
> The above table is created using the Pivot table from Pandas. You can see how Pivot tables are created in my previous article Pandas.

Now, let's see how we can create a heatmap for the above table.

In the above code, we have saved the data in the new variable "heat."

The heatmap looks like the

following,



Default heatmap created on Seaborn

We can apply some customization to the above graph, and also can change the color gradient so that the highest value will be darker in color and the lowest value will be lighter.

The updated code will be something like this,

The heatmap for the above-updated code looks like this,

Heatmap with some customizations to the code

If we observe, in the code we have given "**annot = True**", what this means is, when **annot is true**, each cell in the graph displays its value. If we haven't mention **annot** in our code, then the default value it takes is **False.**

Seaborn also supports some of the other types of graphs like **Line Plots, Bar Graphs, Stacked bar charts,** etc. But, they don't offer anything

different from the ones created through matplotlib.

Conclusion

So, this is how Seaborn works in Python and the different types of graphs we can create using seaborn. As I have already mentioned, Seaborn is built on top of the matplotlib library. So, if we are already familiar with the Matplotlib and its functions, we can easily build Seaborn graphs and can explore more depth concepts.

Thank you for reading and **Happy Coding!!!**

Check out my previous articles about Python here

- Pandas: Python
- Matplotlib: Python
- NumPy: Python
- Time Complexity and Its Importance in Python
- Python Recursion or Recursive Function in Python

- **Python Programs to check for Armstrong Number (n digit) and Fenced Matrix**
- **Python: Problems for Basics Reference — Swapping, Factorial, Reverse Digits, Pattern Print**

References

- **Seaborn:** https://www.w3schools.com/python/numpy_random_seaborn.asp
- **Official seaborn tutorial:** https://seaborn.pydata.org/tutorial.html
- **Seaborn | Regression Plots:** https://www.geeksforgeeks.org/seaborn-regression-plots/
- **Seaborn heatmap:** https://likegeeks.com/seaborn-heatmap-tutorial/
- **Google Play Store Apps — Dataset:** https://www.kaggle.com/lava18/google-play-store-apps

Thanks to Michael Armanious