Ref link: <https://seaborn.pydata.org/tutorial/introduction.html>

Seaborn is a library for making statistical graphics in Python. It builds on top of [matplotlib](https://matplotlib.org/) and integrates closely with [pandas](https://pandas.pydata.org/) data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

Seaborn notebook code: [seaborn](http://localhost:8888/notebooks/seaborn.ipynb)

Seaborn:

Histograms, KDE, Box Plots, Categorical Scatter Plots, Pair Plots, Heatmaps

Histograms:

<https://seaborn.pydata.org/examples/faceted_histogram.html>

[**displot**](https://seaborn.pydata.org/generated/seaborn.displot.html#seaborn.displot)

Figure-level interface to distribution plot functions.

[**kdeplot**](https://seaborn.pydata.org/generated/seaborn.kdeplot.html#seaborn.kdeplot)

Plot univariate or bivariate distributions using kernel density estimation.

[**rugplot**](https://seaborn.pydata.org/generated/seaborn.rugplot.html#seaborn.rugplot)

Plot a tick at each observation value along the x and/or y axes.

[**ecdfplot**](https://seaborn.pydata.org/generated/seaborn.ecdfplot.html#seaborn.ecdfplot)

Plot empirical cumulative distribution functions.

[**jointplot**](https://seaborn.pydata.org/generated/seaborn.jointplot.html#seaborn.jointplot)

Draw a bivariate plot with univariate marginal distributions.

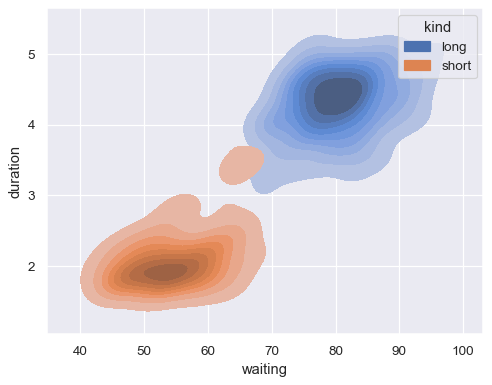
<https://seaborn.pydata.org/generated/seaborn.histplot.html>

**Histogram Visualization with Seaborn**: <https://www.kdnuggets.com/2023/01/creating-beautiful-histograms-seaborn.html>

**Seaborn KDS**: <https://seaborn.pydata.org/generated/seaborn.kdeplot.html>

A kernel density estimate (KDE) plot is a method for visualizing the distribution of observations in a dataset, analogous to a histogram. KDE represents the data using a continuous probability density curve in one or more dimensions.

Relative to a histogram, KDE can produce a plot that is less cluttered and more interpretable, especially when drawing multiple distributions. But it has the potential to introduce distortions if the underlying distribution is bounded or not smooth. Like a histogram, the quality of the representation also depends on the selection of good smoothing parameters.



Code: <https://www.geeksforgeeks.org/seaborn-kdeplot-a-comprehensive-guide/>

**Box Plots:**

<https://seaborn.pydata.org/generated/seaborn.boxplot.html>

A box plot helps to maintain the distribution of quantitative data in such a way that it facilitates the comparisons between variables or across levels of a categorical variable. The main body of the box plot showing the quartiles and the median’s confidence intervals if enabled. The medians have horizontal lines at the median of each box and while whiskers have the vertical lines extending to the most extreme, non-outlier data points and caps are the horizontal lines at the ends of the whiskers.

Ref link:

<https://www.geeksforgeeks.org/boxplot-using-seaborn-in-python/>

**Categorical Scatter Plots:** [**https://seaborn.pydata.org/tutorial/categorical.html**](https://seaborn.pydata.org/tutorial/categorical.html)

[**https://seaborn.pydata.org/tutorial/categorical.html**](https://seaborn.pydata.org/tutorial/categorical.html)

Categorical scatterplots:

* [**stripplot()**](https://seaborn.pydata.org/generated/seaborn.stripplot.html#seaborn.stripplot) (with kind="strip"; the default)
* [**swarmplot()**](https://seaborn.pydata.org/generated/seaborn.swarmplot.html#seaborn.swarmplot) (with kind="swarm")

Categorical distribution plots:

* [**boxplot()**](https://seaborn.pydata.org/generated/seaborn.boxplot.html#seaborn.boxplot) (with kind="box")
* [**violinplot()**](https://seaborn.pydata.org/generated/seaborn.violinplot.html#seaborn.violinplot) (with kind="violin")
* [**boxenplot()**](https://seaborn.pydata.org/generated/seaborn.boxenplot.html#seaborn.boxenplot) (with kind="boxen")

Categorical estimate plots:

* [**pointplot()**](https://seaborn.pydata.org/generated/seaborn.pointplot.html#seaborn.pointplot) (with kind="point")
* [**barplot()**](https://seaborn.pydata.org/generated/seaborn.barplot.html#seaborn.barplot) (with kind="bar")
* [**countplot()**](https://seaborn.pydata.org/generated/seaborn.countplot.html#seaborn.countplot) (with kind="count")

These families represent the data using different levels of granularity. When deciding which to use, you’ll have to think about the question that you want to answer. The unified API makes it easy to switch between different kinds and see your data from several perspectives.

In this tutorial, we’ll mostly focus on the figure-level interface, **[catplot()](https://seaborn.pydata.org/generated/seaborn.catplot.html" \l "seaborn.catplot" \o "seaborn.catplot)**. Remember that this function is a higher-level interface each of the functions above, so we’ll reference them when we show each kind of plot, keeping the more verbose kind-specific API documentation at hand.

## **Categorical scatterplots**[**#**](https://seaborn.pydata.org/tutorial/categorical.html#categorical-scatterplots)

The default representation of the data in **[catplot()](https://seaborn.pydata.org/generated/seaborn.catplot.html" \l "seaborn.catplot" \o "seaborn.catplot)** uses a scatterplot. There are actually two different categorical scatter plots in seaborn. They take different approaches to resolving the main challenge in representing categorical data with a scatter plot, which is that all of the points belonging to one category would fall on the same position along the axis corresponding to the categorical variable. The approach used by **[stripplot()](https://seaborn.pydata.org/generated/seaborn.stripplot.html" \l "seaborn.stripplot" \o "seaborn.stripplot)**, which is the default “kind” in **[catplot()](https://seaborn.pydata.org/generated/seaborn.catplot.html" \l "seaborn.catplot" \o "seaborn.catplot)** is to adjust the positions of points on the categorical axis with a small amount of random “jitter”:

*Try a simple exercise for Categorical Scatter Plots on your own*

Pair Plots: <https://seaborn.pydata.org/generated/seaborn.pairplot.html>

Heatmaps:

<https://seaborn.pydata.org/generated/seaborn.heatmap.html>

The **Seaborn.heatmap()** method is used to plot rectangular data in the form of a color-coded matrix.

Let us define the heat map first −

A **heat map** is one of the data visualization tools that shows the magnitude of a certain phenomenon in the form of colors. The hue and intensity attributes are used to depict the variation in the data plotted on the heat map. These usually show the nature of the phenomenon plotted; whether it is clustered or dispersed over a space.

### **Syntax**

Following is the syntax of the seaborn.heatmap() method −

seaborn.heatmap(data, \*, vmin=None, vmax=None, cmap=None, center=None, robust=False, annot=None, fmt='.2g', annot\_kws=None, linewidths=0, linecolor='white', cbar=True, cbar\_kws=None, cbar\_ax=None, square=False, xticklabels='auto', yticklabels='auto', mask=None, ax=None, \*\*kwargs)

### **Parameters**

Some of the parameters of the heatmap() method are discussed below.

|  |  |
| --- | --- |
| **S.No** | **Parameter and Description** |
| 1 | **Data**  It takes a rectangular dataset as input. An ndarray that can be forced into a 2D dataset. If a Pandas DataFrame is provided, the columns and rows will be labelled using the index and column information. |
| 2 | **Vmin, vmax**  This optional arguments takes floating point values as input and these values serve as colormap anchors; if not, the data and other keyword arguments will be used to infer them. |
| 3 | **Cmap**  This optional parameter takes matplotlib colors or a list of colors as input. It performs the transformation of data values into a colour space. The default, if not specified, will depend on whether centre is set. |
| 4 | **Center**  Takes floating point values as input and determines The value at which to center the colormap when plotting divergant data. |
| 5 | **Mask**  Takes bool array or DataFrame and is an optional parameter. If passed, data will not be shown in cells where mask is True. |
| 6 | **Robust**  This optional parameter takes Boolean values and the colormap range is produced using robust quantiles rather than the extreme values if True and vmin or vmax are not present. |
| 7 | **Linewidth**  Takes floating value and determines the width of lines that divide each cell. |
| 8 | **linecolor**  Takes color as input and determines the color of the lines that divide each cell. |
| 9 | **cbar**  Takes Boolean values and determines if a colorbar should be drawn or not. |

## **Return Value**

This method returns the axes object with the heatmap plotted. Let us load the seaborn library and the dataset before moving on to developing the plots.

***Thank you***