

Home > About Python > Learn Python

Python Seaborn Tutorial For Beginners: Start Visualizing Data

This Seaborn tutorial introduces you to the basics of statistical data visualization

Updated Mar 2023 · 20 min read

CONTENTS

- What is Seaborn?
- Seaborn vs. Matplotlib
- Installing Seaborn
- Sample Datasets
- Seaborn Plot types
- Seaborn Examples
- Customizing Seaborn plots
- Best practices for Seaborn visualization
- Comparing Seaborn to Other Plotting Libraries
- Conclusion

SHARE



Visualization is a crucial aspect of data analysis and interpretation, as it allows for easy comprehension of complex data sets. It helps in identifying patterns, relationships, and trends that might not be apparent through raw data alone. In recent years, Python has become one of the most popular programming languages for data analysis, owing to its vast array of libraries and frameworks.

Visualization libraries in Python enable users to create intuitive and interactive data visualizations that can effectively communicate insights to a broad audience. Some of the popular visualization libraries and frameworks in Python include Matplotlib, Plotly, Bokeh, and Seaborn. Each of these libraries has its own unique features and capabilities that cater to specific needs.

In this tutorial, we will focus on Seaborn, a popular data visualization library in Python that offers an easy-to-use interface for creating informative statistical graphics.

What is Seaborn?

Built on top of Matplotlib, Seaborn is a well-known Python library for data visualization that offers a user-friendly interface for producing visually appealing and informative statistical graphics. It is designed to work with Pandas dataframes, making it easy to visualize and explore data quickly and effectively.

Seaborn offers a variety of powerful tools for visualizing data, including scatter plots, line plots, bar plots, heat maps, and many more. It also provides support for advanced statistical analysis, such as regression analysis, distribution plots, and categorical plots.

You can learn all about Seaborn and its uses for with our [Introduction to Data Visualization with Seaborn](#) course.

Seaborn's key benefit lies in its capability to generate attractive plots with minimal coding efforts. It provides a range of default themes and color palettes, which you can easily customize to suit your preferences. Additionally, Seaborn offers a range of built-in statistical functions, allowing users to easily perform complex statistical analysis with their visualizations.

Another notable feature of Seaborn is its ability to create complex multi-plot visualizations. With Seaborn, users can create grids of plots that allow for easy comparison between multiple variables or subsets of data. This makes it an ideal tool for exploratory data analysis and presentation.

Seaborn is a powerful and flexible data visualization library in Python that offers an easy-to-use interface for creating informative and aesthetically pleasing statistical graphics. It provides a range of tools for visualizing data, including advanced statistical analysis, and makes it easy to create complex multi-plot visualizations.

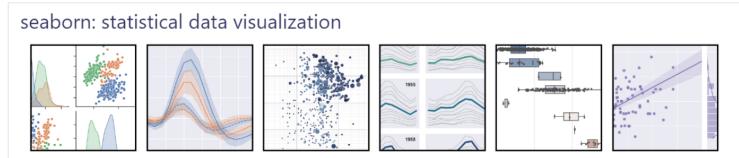


Image Source

Seaborn vs. Matplotlib

Python's two most widely used data visualization libraries are Matplotlib and Seaborn. While both libraries are designed to create high-quality graphics and visualizations, they have several key differences that make them better suited for different use cases.

One of the main differences between Matplotlib and Seaborn is their focus. Matplotlib is a



Moez Ali
Data Scientist, Founder & Creator of PyCaret

TOPICS

- Python
- Data Visualization

RELATED



10 Essential Python Skills All Data Scientists Should Master



The 7 Best Python Certifications For All Levels



Textacy: An Introduction to Text Data Cleaning and Normalization in Python



Types of Data Plots and How to Create Them in Python

[low-level plotting library](#) that provides a wide range of tools for creating highly customizable visualizations. It is a highly flexible library, allowing users to create almost any type of plot they can imagine. This flexibility comes at the cost of a steeper learning curve and more verbose code.

[Seaborn, on the other hand, is a high-level interface for creating statistical graphics](#). It is built on top of Matplotlib and provides a simpler, more intuitive interface for creating common statistical plots. Seaborn is designed to work with Pandas dataframes, making it easy to create visualizations with minimal code. It also offers a range of built-in statistical functions, allowing users to easily perform complex statistical analyses with their visualizations.

Another key difference between Matplotlib and Seaborn is their default styles and color palettes. Matplotlib provides a limited set of default styles and color palettes, requiring users to customize their plots manually to achieve a desired look. Seaborn, on the other hand, offers a range of default styles and color palettes that are optimized for different types of data and visualizations. This makes it easy for users to create visually appealing plots with minimal customization.

While both libraries have their strengths and weaknesses, [Seaborn is generally better suited for creating statistical graphics and exploratory data analysis](#), while Matplotlib is better suited for creating highly customizable plots for presentations and publications. However, it is worth noting that Seaborn is built on top of Matplotlib, and the two libraries can be used together to create complex, highly customizable visualizations that leverage the strengths of both libraries.

You can explore Matplotlib in more detail with our [Introduction to Plotting with Matplotlib in Python](#) tutorial.

Matplotlib and Seaborn are both powerful data visualization libraries in Python, with different strengths and weaknesses. Understanding the differences between the two libraries can help users choose the right tool for their specific data visualization needs.

Installing Seaborn

Seaborn is supported on Python 3.7+ and has very minimal core dependencies. Installing Seaborn is pretty straightforward. You can either install it with Python's pip manager or conda package manager.

```
# install seaborn with pip  
pip install seaborn
```

[Explain code](#)

Powered by OpenAI

When you use pip, Seaborn and its required dependencies will be installed. If you want to access additional and optional features, you can also include optional dependencies in pip install. For example:

```
pip install seaborn[stats]
```

[Explain code](#)

Powered by OpenAI

Or with conda:

```
# install seaborn with conda  
conda install seaborn
```

[Explain code](#)

Powered by OpenAI

Sample Datasets

Seaborn provides several built-in datasets that we can use for data visualization and statistical analysis. These datasets are stored in pandas dataframes, making them easy to use with Seaborn's plotting functions.

One of the most common datasets that's also used in all the official examples of Seaborn is called 'tips dataset'; it contains information about tips given in restaurants. Here's an example of loading and visualizing the Tips dataset in Seaborn:

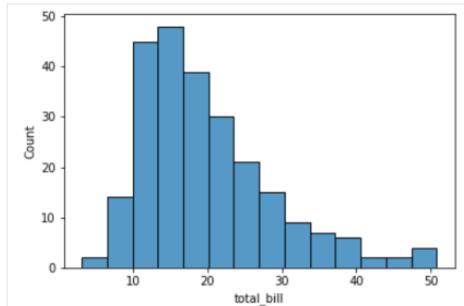
```
import seaborn as sns  
  
# Load the Tips dataset  
tips = sns.load_dataset("tips")
```

```
# Create a histogram of the total bill amounts  
sns.histplot(data=tips, x="total_bill")
```

[Explain code](#)

Powered by OpenAI

Output:



If you don't understand this plot yet - no worries. This is called a histogram. We will explain more in detail about histograms later in this tutorial. For now, the takeaway is that Seaborn comes with a lot of sample datasets as pandas DataFrames that are easy to use and practice your visualization skills. Here is another example of loading the 'exercise' dataset.

```
import seaborn as sns
```



```
# Load the exercise dataset  
exercise = sns.load_dataset("exercise")
```



```
# check the head  
exercise.head()
```

[Explain code](#)

Powered by OpenAI

Output:

	Unnamed: 0	id	diet	pulse	time	kind
0	0	1	low fat	85	1 min	rest
1	1	1	low fat	85	15 min	rest
2	2	1	low fat	88	30 min	rest
3	3	2	low fat	90	1 min	rest
4	4	2	low fat	92	15 min	rest

Seaborn Plot types

Seaborn provides a wide range of plot types that can be used for data visualization and exploratory data analysis. Broadly speaking, any visualization can fall into one of the three categories.

- **Univariate** – x only (contains only one axis of information)
- **Bivariate** – x and y (contains two axis of information)
- **Trivariate** – x, y, z (contains three axis of information)

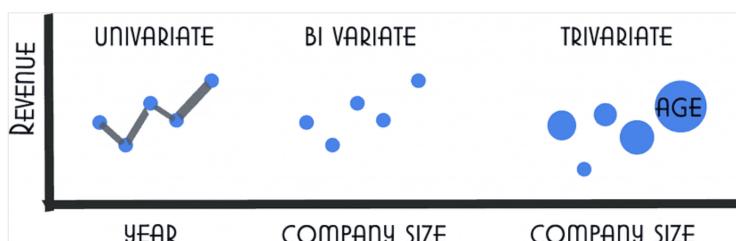


Image Source

Here are some of the most commonly used plot types in Seaborn:

- **Scatter Plot.** A scatter plot is used to **visualize the relationship between two variables**. Seaborn's `scatterplot()` function provides a simple way to create scatter plots.
- **Line Plot.** A line plot is used to **visualize the trend of a variable over time**. Seaborn's `lineplot()` function provides a simple way to create line plots.
- **Histogram.** A histogram is used to **visualize the distribution of a variable**. Seaborn's `histplot()` function provides a simple way to create histograms.
- **Box Plot.** A box plot is used to **visualize the distribution of a variable**. Seaborn's `boxplot()` function provides a simple way to create box plots.
- **Violin Plot.** A violin plot is similar to a box plot, but provides a more detailed view of the distribution of the data. Seaborn's `violinplot()` function provides a simple way to create violin plots.
- **Heatmap.** A heatmap is used to **visualize the correlation between different variables**. Seaborn's `heatmap()` function provides a simple way to create heatmaps.
- **Pairplot.** A pairplot is used to **visualize the relationship between multiple variables**. Seaborn's `pairplot()` function provides a simple way to create pairplots.

We will now see examples and detailed explanations for each of these in the next section of this tutorial.

Seaborn Examples

Seaborn scatter plots

Scatter plots are used to visualize the relationship between two continuous variables. Each point on the plot represents a single data point, and the position of the point on the x and y-axis represents the values of the two variables.

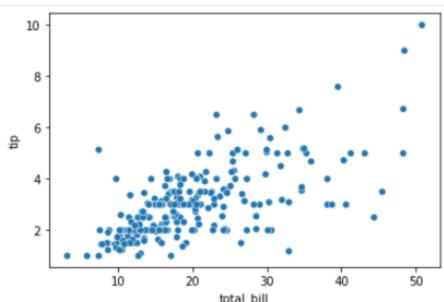
The plot can be customized with different colors and markers to help distinguish different groups of data points. In Seaborn, scatter plots can be created using the `scatterplot()` function.

```
import seaborn as sns  
  
tips = sns.load_dataset("tips")  
  
sns.scatterplot(x="total_bill", y="tip", data=tips)
```

[Explain code](#)

Powered by OpenAI

Output:



This simple plot can be improved by customizing the 'hue' and 'size' parameters of the plot. Here's how:

```
import seaborn as sns  
  
import matplotlib.pyplot as plt  
  
  
tips = sns.load_dataset("tips")  
  
  
# customize the scatter plot  
  
sns.scatterplot(x="total_bill", y="tip", hue="sex", size="size", sizes=(50, 200),
```

```

# add labels and title
plt.xlabel("Total Bill")
plt.ylabel("Tip")
plt.title("Relationship between Total Bill and Tip")

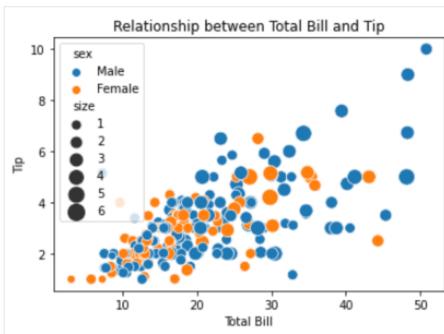
# display the plot
plt.show()

```

[Explain code](#)

Powered by OpenAI

Output:



In this example, we have used the `seaborn` library for simple scatter plot and have used `matplotlib` for further customizing the scatter plot.

Seaborn line plots

Line plots are used to [visualize trends in data over time or other continuous variables](#). In a line plot, each data point is connected by a line, creating a smooth curve. In Seaborn, line plots can be created using the `lineplot()` function.

```

import seaborn as sns

fmri = sns.load_dataset("fmri")

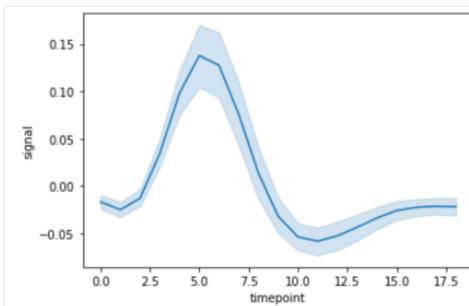
sns.lineplot(x="timepoint", y="signal", data=fmri)

```

[Explain code](#)

Powered by OpenAI

Output:



We can very easily customize this by using `event` and `region` columns from the dataset.

```

import seaborn as sns

import matplotlib.pyplot as plt

fmri = sns.load_dataset("fmri")

```

[Explain code](#)

```

# customize the line plot

sns.lineplot(x="timepoint", y="signal", hue="event", style="region", markers=True

# add labels and title
plt.xlabel("Timepoint")
plt.ylabel("Signal Intensity")
plt.title("Changes in Signal Intensity over Time")

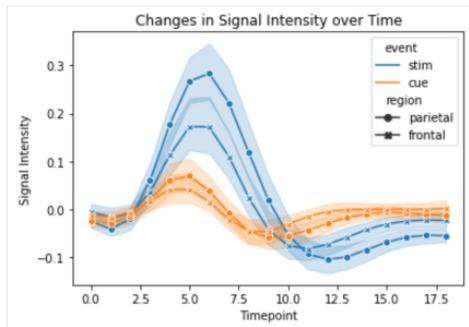
# display the plot
plt.show()

```

[Explain code](#)

Powered by OpenAI

Output:



Again, we have used the `seaborn` library to do a simple line plot and have used the `matplotlib` library to customize and improve the simple line plot. You can take a more in-depth look at [Seaborn line plots](#) in our separate tutorial.

Seaborn bar plots

Bar plots are used to visualize the relationship between a categorical variable and a continuous variable. In a bar plot, each bar represents the mean or median (or any aggregation) of the continuous variable for each category. In Seaborn, bar plots can be created using the `barplot()` function.

```

import seaborn as sns

titanic = sns.load_dataset("titanic")

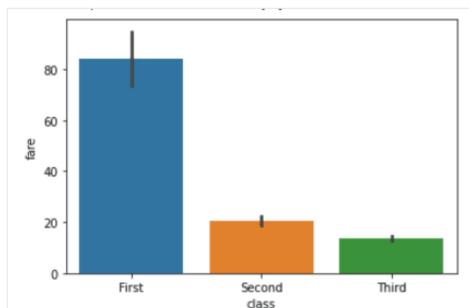
sns.barplot(x="class", y="fare", data=titanic)

```

[Explain code](#)

Powered by OpenAI

Output:



Let's customize this plot by including 'sex' column from the dataset.

```

import seaborn as sns

```

[Explain code](#)

```

import matplotlib.pyplot as plt

titanic = sns.load_dataset("titanic")

# customize the bar plot
sns.barplot(x="class", y="fare", hue="sex", ci=None, palette="muted", data=titanic)

# add labels and title
plt.xlabel("Class")
plt.ylabel("Fare")
plt.title("Average Fare by Class and Gender on the Titanic")

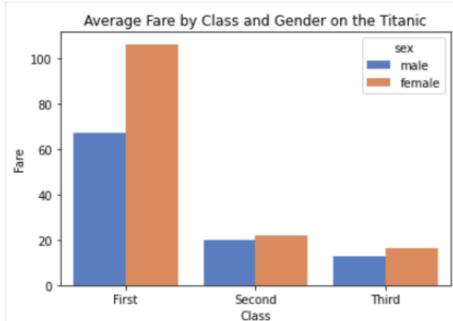
# display the plot
plt.show()

```

[Explain code](#)

Powered by OpenAI

Output:



Seaborn histograms

Histograms visualize the distribution of a continuous variable. In a histogram, the data is divided into bins and the height of each bin represents the frequency or count of data points within that bin. In Seaborn, histograms can be created using the `histplot()` function.

```

import seaborn as sns

iris = sns.load_dataset("iris")

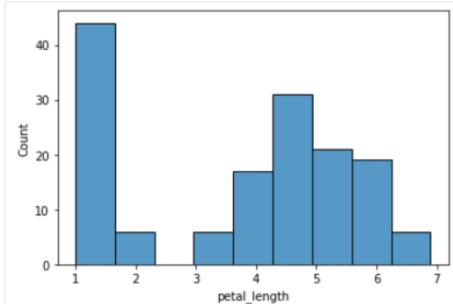
sns.histplot(x="petal_length", data=iris)

```

[Explain code](#)

Powered by OpenAI

Output:



Customizing a histogram

```

import seaborn as sns
import matplotlib.pyplot as plt

iris = sns.load_dataset("iris")

# customize the histogram
sns.histplot(data=iris, x="petal_length", bins=20, kde=True, color="green")

# add labels and title
plt.xlabel("Petal Length (cm)")
plt.ylabel("Frequency")
plt.title("Distribution of Petal Lengths in Iris Flowers")

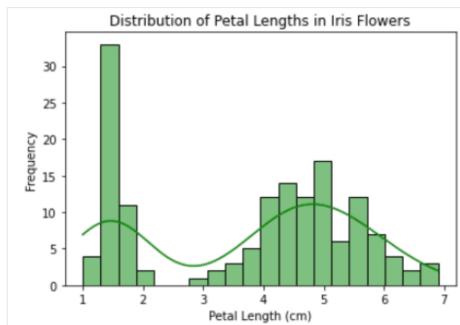
# display the plot
plt.show()

```

[Explain code](#)

Powered by OpenAI

Output:



Seaborn density plots

Density plots, also known as kernel density plots, are a type of data visualization that display the distribution of a continuous variable. They are similar to histograms, but instead of representing the data as bars, density plots use a smooth curve to estimate the density of the data. In Seaborn, density plots can be created using the `kdeplot()` function.

```

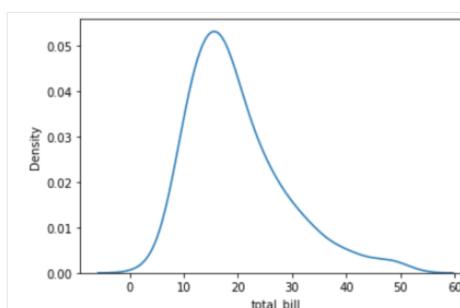
import seaborn as sns
tips = sns.load_dataset("tips")
sns.kdeplot(data=tips, x="total_bill")

```

[Explain code](#)

Powered by OpenAI

Output:



Let's improve the plot by customizing it.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Load the "tips" dataset from Seaborn
tips = sns.load_dataset("tips")

# Create a density plot of the "total_bill" column from the "tips" dataset
# We use the "hue" parameter to differentiate between "lunch" and "dinner" meal times
# We use the "fill" parameter to fill the area under the curve
# We adjust the "alpha" and "linewidth" parameters to make the plot more visually appealing
sns.kdeplot(data=tips, x="total_bill", hue="time", fill=True, alpha=0.6, linewidth=1)

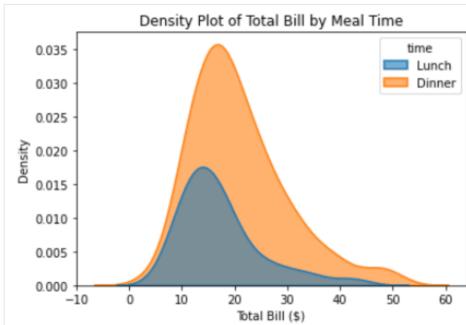
# Add a title and labels to the plot using Matplotlib
plt.title("Density Plot of Total Bill by Meal Time")
plt.xlabel("Total Bill ($)")
plt.ylabel("Density")

# Show the plot
plt.show()
```

[Explain code](#)

Powered by OpenAI

Output:



Seaborn box plots

Box plots are a type of visualization that shows the distribution of a dataset. They are commonly used to compare the distribution of one or more variables across different categories.

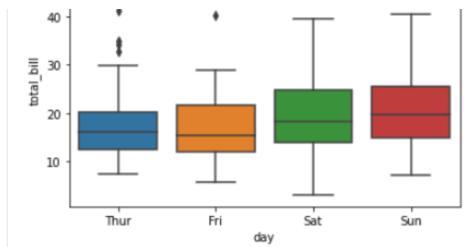
```
import seaborn as sns
tips = sns.load_dataset("tips")
sns.boxplot(x="day", y="total_bill", data=tips)
```

[Explain code](#)

Powered by OpenAI

Output:





Customize the box plot by including 'time' column from the dataset.

```
import seaborn as sns
import matplotlib.pyplot as plt

# load the tips dataset from Seaborn
tips = sns.load_dataset("tips")

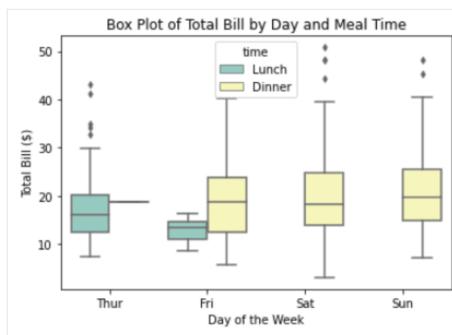
# create a box plot of total bill by day and meal time, using the "hue" parameter
# customize the color scheme using the "palette" parameter
# adjust the linewidth and fliersize parameters to make the plot more visually appealing
sns.boxplot(x="day", y="total_bill", hue="time", data=tips, palette="Set3", linewidth=2, fliersize=10)

# add a title, xlabel, and ylabel to the plot using Matplotlib functions
plt.title("Box Plot of Total Bill by Day and Meal Time")
plt.xlabel("Day of the Week")
plt.ylabel("Total Bill ($)")

# display the plot
plt.show()
```

[Explain code](#)

Powered by OpenAI



Seaborn violin plots

A violin plot is a type of data visualization that combines aspects of both box plots and density plots. It displays a density estimate of the data, usually smoothed by a kernel density estimator, along with the interquartile range (IQR) and median in a box plot-like form.

The width of the violin represents the density estimate, with wider parts indicating higher density, and the IQR and median are shown as a white dot and line within the violin.

```
import seaborn as sns
```

```

# load the iris dataset from Seaborn

iris = sns.load_dataset("iris")

# create a violin plot of petal length by species

sns.violinplot(x="species", y="petal_length", data=iris)

# display the plot

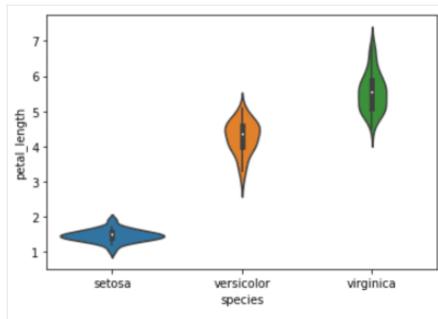
plt.show()

```

[Explain code](#)

Powered by  OpenAI

Output:



Seaborn heatmaps

A heatmap is a graphical representation of data that uses colors to depict the value of a variable in a two-dimensional space. Heatmaps are commonly used to visualize the correlation between different variables in a dataset.

```

import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
tips = sns.load_dataset('tips')

# Create a heatmap of the correlation between variables
corr = tips.corr()

sns.heatmap(corr)

# Show the plot
plt.show()

```

[Explain code](#)

Powered by  OpenAI

Output:





Another example of a heatmap using the `flights` dataset.

```
import seaborn as sns

import matplotlib.pyplot as plt

# Load the dataset

flights = sns.load_dataset('flights')

# Pivot the data

flights = flights.pivot('month', 'year', 'passengers')

# Create a heatmap

sns.heatmap(flights, cmap='Blues', annot=True, fmt='d')

# Set the title and axis labels

plt.title('Passengers per month')

plt.xlabel('Year')

plt.ylabel('Month')

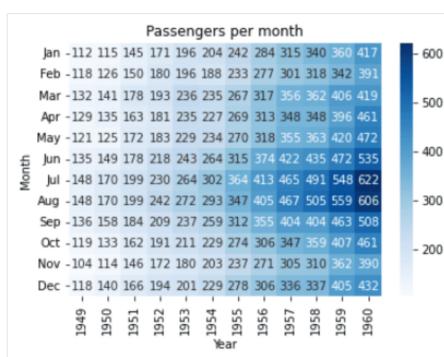
# Show the plot

plt.show()
```

[Explain code](#)

Powered by OpenAI

Output:



In this example, we are using the `flights` dataset from the `seaborn` library. We pivot the data to make it suitable for heatmap representation using the `.pivot()` method. Then, we create a heatmap using the `sns.heatmap()` function and pass the pivoted flights variable as the argument.

Seaborn pair plots

Pair plots are a type of visualization in which multiple pairwise scatter plots are displayed in a matrix format. Each scatter plot shows the relationship between two variables, while the diagonal plots show the distribution of the individual variables.

```
import seaborn as sns
```

```

# Load iris dataset

iris = sns.load_dataset("iris")

# Create pair plot

sns.pairplot(data=iris)

# Show plot

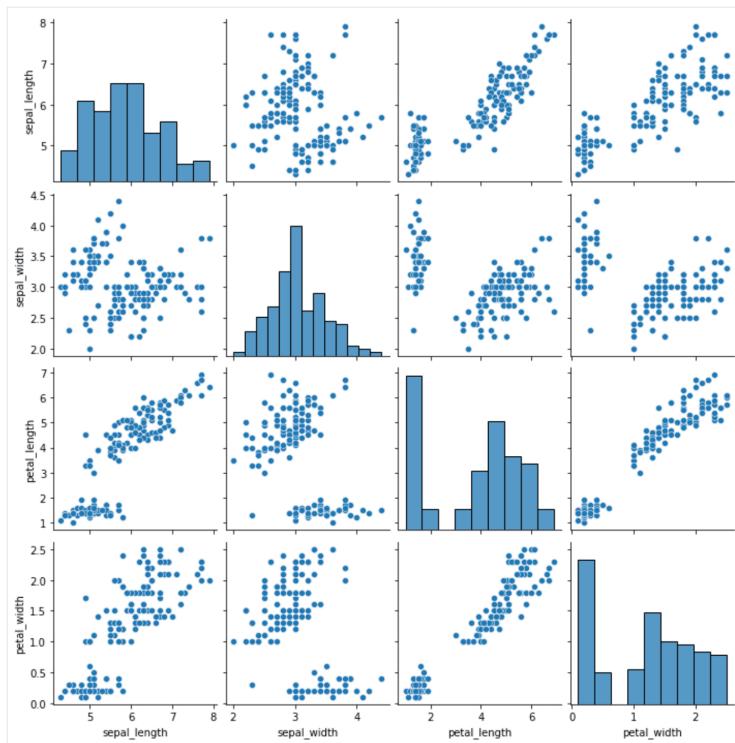
plt.show()

```

[Explain code](#)

Powered by OpenAI

Output:



We can customize this plot by using `hue` and `diag_kind` parameter.

```

import seaborn as sns
import matplotlib.pyplot as plt

# Load iris dataset

iris = sns.load_dataset("iris")

# Create pair plot with custom settings

sns.pairplot(data=iris, hue="species", diag_kind="kde", palette="husl")

# Set title

plt.title("Iris Dataset Pair Plot")

# Show plot

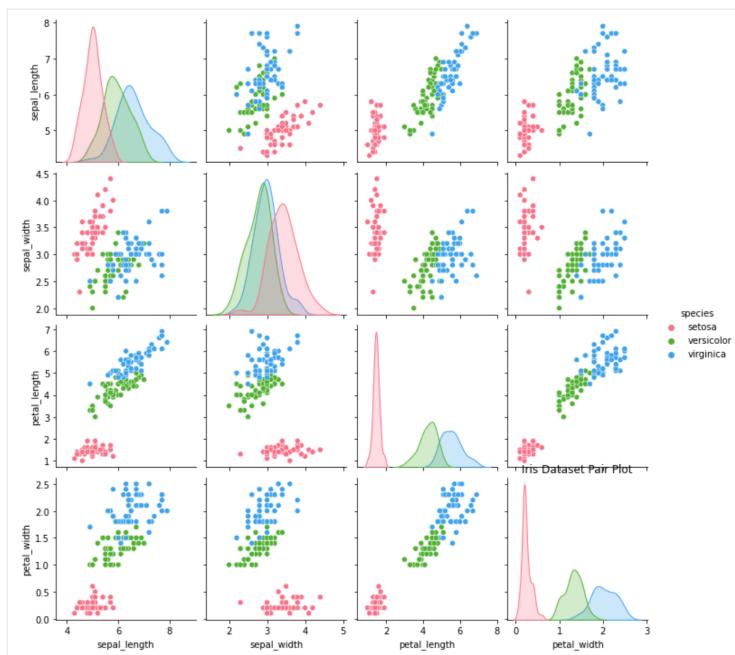
```

```
plt.show()
```

[Explain code](#)

Powered by OpenAI

Output:



Seaborn joint plots

Joint plot is a powerful visualization technique in seaborn that combines two different plots in one visualization: a scatter plot and a histogram. The scatter plot shows the relationship between two variables, while the histogram shows the distribution of each individual variable. This allows for a more comprehensive analysis of the data, as it shows the correlation between the two variables and their individual distributions.

Here is a simple example of building a seaborn joint plot using the iris dataset:

```
import seaborn as sns
import matplotlib.pyplot as plt

# load iris dataset
iris = sns.load_dataset("iris")

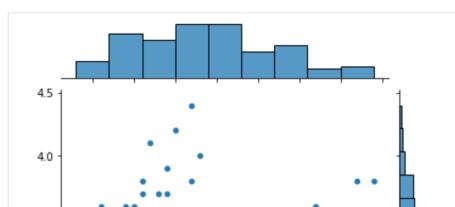
# plot a joint plot of sepal length and sepal width
sns.jointplot(x="sepal_length", y="sepal_width", data=iris)

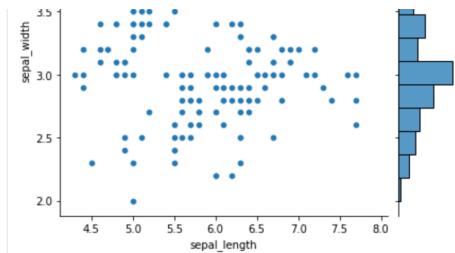
# display the plot
plt.show()
```

[Explain code](#)

Powered by OpenAI

Output:





Seaborn facet grids

FacetGrid is a powerful seaborn tool that allows you to visualize the distribution of one variable as well as the relationship between two variables, across levels of additional categorical variables.

FacetGrid creates a grid of subplots based on the unique values in the categorical variable specified.

```
import seaborn as sns
```

```
# load the tips dataset
```

```
tips = sns.load_dataset('tips')
```

```
# create a FacetGrid for day vs total_bill
```

```
g = sns.FacetGrid(tips, col="day")
```

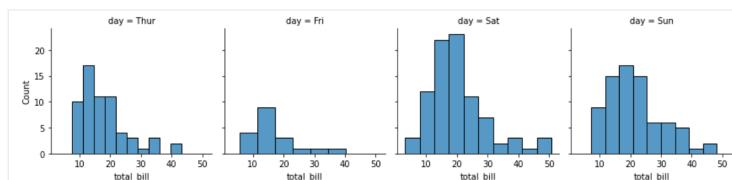
```
# plot histogram for total_bill in each day
```

```
g.map(sns.histplot, "total_bill")
```

[Explain code](#)

Powered by OpenAI

Output:



Python Seaborn Cheat Sheet

This Python Seaborn cheat sheet with code samples guides you through the data visualization library that is based on Matplotlib.

Customizing Seaborn plots

Seaborn is a powerful data visualization library that provides numerous ways to customize the appearance of plots. Customizing Seaborn plots is an essential part of creating meaningful and visually appealing visualizations.

Here are some examples of customizing seaborn plots:

Changing Color Palettes

Here is an example of how you can change the color palettes of your seaborn plots:

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
# Load sample dataset
```

```

tips = sns.load_dataset("tips")

# Create a scatter plot with color palette
sns.scatterplot(x="total_bill", y="tip", hue="day", data=tips, palette="Set2")

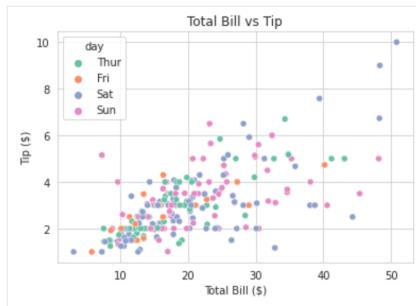
# Customize plot
plt.title("Total Bill vs Tip")
plt.xlabel("Total Bill ($)")
plt.ylabel("Tip ($)")
plt.show()

```

[Explain code](#)

Powered by  OpenAI

Output:



Adjusting Figure Size

To adjust the figure size on your seaborn plots, you can use the example below as a guide:

```

import seaborn as sns
import matplotlib.pyplot as plt

# Load sample dataset
iris = sns.load_dataset("iris")

# Create a violin plot with adjusted figure size
plt.figure(figsize=(8,6))
sns.violinplot(x="species", y="petal_length", data=iris)

# Customize plot
plt.title("Petal Length Distribution by Species")
plt.xlabel("Species")
plt.ylabel("Petal Length (cm)")
plt.show()

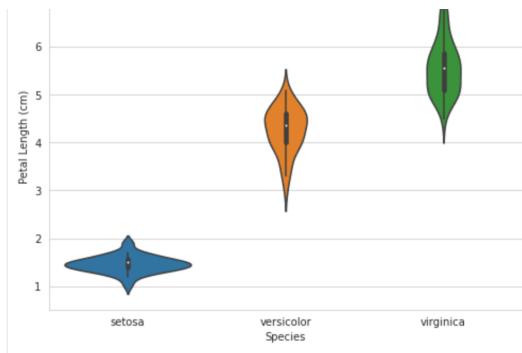
```

[Explain code](#)

Powered by  OpenAI

Output:





Adding Annotations

Annotations can help to make your visualizations easier to read. We've shown an example of how to add them below:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Load sample dataset
diamonds = sns.load_dataset("diamonds")

# Create a scatter plot with annotations
sns.scatterplot(x="carat", y="price", data=diamonds)

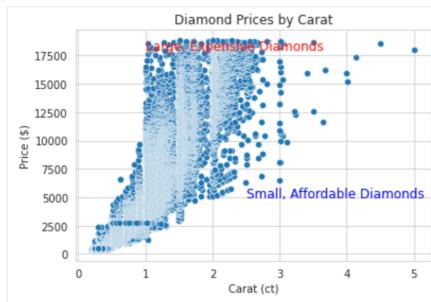
# Add annotations
plt.text(1, 18000, "Large, Expensive Diamonds", fontsize=12, color="red")
plt.text(2.5, 5000, "Small, Affordable Diamonds", fontsize=12, color="blue")

# Customize plot
plt.title("Diamond Prices by Carat")
plt.xlabel("Carat (ct)")
plt.ylabel("Price ($)")
plt.show()
```

[Explain code](#)

Powered by OpenAI

Output:



Best practices for Seaborn visualization

Choose the right plot type for your data

Seaborn provides a wide range of plot types, each designed for different types of data and purposes.

analysis. It's important to choose the right plot type for your data to effectively communicate your findings. For example, a scatter plot may be more appropriate for visualizing the relationship between two continuous variables, while a bar plot may be more appropriate for visualizing categorical data.

Use color effectively

Color can be a powerful tool for data visualization, but it's important to use it effectively. Avoid using too many colors or overly bright colors, as this can make the visualization difficult to read. Instead, use color to highlight important information or to group similar data points.

Label your axes and use clear titles

Labels and titles are essential for effective data visualization. Make sure to label your axes clearly and provide a descriptive title for your visualization. This will help your audience understand the message you are trying to convey.

Consider the audience

When creating visualizations, it's important to consider the audience and the message you are trying to communicate. If your audience is non-technical, use clear and concise language, avoid technical jargon, and provide clear explanations of any statistical concepts.

Use appropriate statistical analysis

Seaborn provides a range of statistical functions that you can use to analyze your data. When choosing a statistical function, make sure to choose the one that is most appropriate for your data and research question.

Customize your visualizations

You'll find a wide range of customization options in Seaborn that you can use to enhance your visualizations. Experiment with different fonts, styles, and colors to find the one that best communicates your message.

Comparing Seaborn to Other Plotting Libraries

Seaborn vs. Matplotlib

Seaborn is built on top of Matplotlib and provides a higher-level interface for creating statistical graphics. While Matplotlib is a general-purpose plotting library, Seaborn is specifically designed for statistical data visualization.

Seaborn offers several advantages over Matplotlib, including simpler syntax for creating complex plots, built-in support for statistical visualizations, and aesthetically pleasing default settings that can be easily customized.

Additionally, Seaborn offers several specialized plot types that are not available in Matplotlib, such as violin plots and swarm plots.

Seaborn vs. Pandas

Pandas is a powerful data manipulation library in Python that offers a range of functionality for working with structured data. While Pandas offers basic plotting capabilities through its `DataFrame.plot()` method, Seaborn provides more advanced visualization functionality that is specifically designed for statistical data.

Seaborn's functions are optimized to work with Pandas data structures, making it easy to create a wide range of informative visualizations directly from Pandas data frames.

Seaborn also offers specialized plot types, such as facet grids and pair plots, that are not available in Pandas.

Seaborn vs. Plotly

Plotly is a web-based data visualization library that offers interactive and collaborative data visualizations.

While Seaborn is primarily focused on creating static visualizations, Plotly offers more interactive and dynamic visualizations that can be used in web applications or shared online. Plotly also offers several specialized plot types that are not available in Seaborn, such as contour plots and 3D surface plots.

However, Seaborn offers simpler syntax and easier customization for creating static visualizations, making it a better choice for certain types of projects.

Conclusion

Seaborn is a powerful data visualization library in Python that provides an intuitive and easy-to-use interface for creating informative statistical graphics. With its vast array of

visualization tools, Seaborn makes it possible to quickly and efficiently explore and communicate insights from complex data sets.

From scatter plots and line plots to heatmaps and facet grids, Seaborn offers a wide range of visualizations to suit different needs. Moreover, Seaborn's ability to integrate with Pandas and Numpy makes it an indispensable tool for data analysts and scientists.

With this beginner's guide to Python Seaborn, you can start exploring the world of data visualization and communicate your insights effectively to a broader audience.

If you would like to further broaden your knowledge in this topic, check out our [Introduction to Data Visualization with Seaborn](#) or [Intermediate Data Visualization with Seaborn](#) courses.

Throughout these courses, you will learn how to utilize Seaborn's advanced visualization tools to analyze various real-world datasets, such as American Housing Survey, college tuition data, and The Daily Show guests.

You can also check out this free Seaborn cheat sheet:

Python Seaborn: Statistical Data Visualization

A Seaborn cheat sheet with the five basic steps to create beautiful statistical graphs in Python.

TOPICS

Python Data Visualization

Learn more about Python and Seaborn

Introduction to Data Visualization with Seaborn

• Beginner ⏸ 4 hr 📺 106.7K

Learn how to create informative and attractive visualizations in Python using the Seaborn library.

[See Details →](#)

[Start Course](#)

Intermediate Data Visualization with Seaborn

• Beginner ⏸ 4 hr 📺 64.2K

Use Seaborn's sophisticated visualization tools to make beautiful, informative visualizations with ease.

[See Details →](#)

[Start Course](#)

Improving Your Data Visualizations in Python

• Beginner ⏸ 4 hr 📺 14K

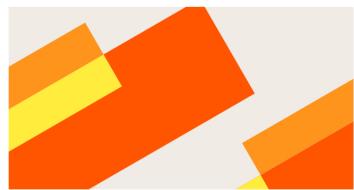
Learn to construct compelling and attractive visualizations that help communicate results efficiently and effectively.

[See Details →](#)

[Start Course](#)

[See More →](#)

Related



10 Essential Python Skills All Data Scientists Should Master

All data scientists need expertise in Python, but which skills are the most important for them to master? Find out th...

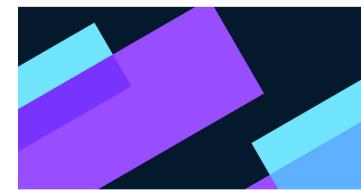
Thaylise Nakamoto • 9 min



The 7 Best Python Certifications For All Levels

Find out whether a Python certification is right for you, what the best options are, and the alternatives on offer in this...

Matt Crabtree • 18 min



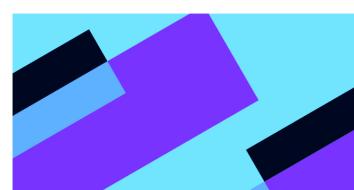
Textacy: An Introduction to Text Data Cleaning and...

Discover how Textacy, a Python library, simplifies text data preprocessing for machine learning. Learn about its unique...

Mustafa El-Dalil • 5 min



Types of Data Plots and How to Create Them in Python



Coding Best Practices and Guidelines for Better Code



Pandas Profiling (ydata-profiling) in Python: A Guide fo...

Explore various types of data plots—from the most common to advanced and unconventional ones—what they show,...

 Elena Kosourova • 21 min

Learn coding best practices to improve your programming skills. Explore coding guidelines for collaboration, code structur...

 Amberle McKee • 26 min

Learn how to use the ydata-profiling library in Python to generate detailed reports for datasets with many features.

 Satyam Tripathi • 9 min

[See More →](#)

Grow your data skills with DataCamp for Mobile

Make progress on the go with our mobile courses and daily 5-minute coding challenges.



LEARN	DATA COURSES	WORKSPACE	RESOURCES	PLANS	SUPPORT	ABOUT
Learn Python	Upcoming Courses	Get Started	Resource Center	Pricing	Help Center	About Us
Learn R	Python Courses	Templates	Upcoming Events	For Business	Become an Affiliate	Learner Stories
Learn AI	R Courses	Integrations	Blog	For Universities		Careers
Learn SQL	SQL Courses	Documentation	Code Alongs	Discounts, Promos & Sales		Become an Instructor
Learn Power BI	Power BI Courses	CERTIFICATION	Tutorials	DataCamp Donates		Press
Learn Tableau	Tableau Courses		Open Source			Leadership
Learn Data Engineering	Spreadsheets Courses	Certifications	RDocumentation			Contact Us
Assessments	Data Analysis Courses	Data Scientist	Course Editor			
Career Tracks	Data Visualization Courses	Data Analyst	Book a Demo with DataCamp for Business			
Skill Tracks	Machine Learning Courses	Data Engineer	Hire Data Professionals			
Courses			Data Portfolio			
Data Science Roadmap			Hire Data Professionals	Portfolio Leaderboard		
	Data Engineering Courses					



[Privacy Policy](#) [Cookie Notice](#) [Do Not Sell My Personal Information](#) [Accessibility](#) [Security](#) [Terms of Use](#)



© 2023 DataCamp, Inc. All Rights Reserved.