

How to Easily Create a Seaborn Scatterplot with Correlation Coefficient

Authored by
stats writer

November 20, 2025

RECOMMENDED CITATION

stats writer (2025). *How to Easily Create a Seaborn Scatterplot with Correlation Coefficient*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=98505>

Visualizing the relationship between two continuous variables is a fundamental task in data analysis. While a [Seaborn](#) scatterplot effectively displays the distribution and potential trend, incorporating the numerical [correlation coefficient](#) directly onto the visualization enhances clarity and provides immediate quantitative insight. This guide details the expert method for generating a Seaborn scatterplot and annotating it precisely with the calculated correlation value.

To accomplish this integration, we leverage the statistical capabilities of the [SciPy](#) library to compute the correlation, and the annotation functionality of [Matplotlib](#), which underlies Seaborn, to display the result on the plot canvas. This combination ensures a robust statistical calculation paired with professional-grade data visualization.

While Seaborn offers functions like `regplot()` that can automatically draw regression lines, manually calculating and annotating the correlation coefficient using `plt.text()` provides greater control over the placement, formatting, and precision of the statistical output, allowing data analysts to customize the visual presentation according to specific requirements.

Establishing the Core Methodology for Annotation

Integrating a correlation coefficient requires a sequence of steps: first, importing the necessary libraries; second, calculating the coefficient using a statistical function; and finally, plotting the data and adding the result as a text annotation. This methodology separates the statistical computation from the graphical rendering, ensuring accuracy and flexibility.

For standard linear relationships, the [Pearson correlation coefficient](#) (r) is the most common measure used. The `scipy.stats.pearsonr()` function is highly reliable for this purpose, returning both the correlation coefficient and the p-value. We focus specifically on extracting the correlation value (the first element of the returned tuple).

The standard syntax below outlines the necessary imports and the basic flow required to calculate the correlation and place it onto a general scatterplot before diving into a detailed practical example. Note that `x` and `y` must be defined based on an existing [pandas DataFrame](#) named `df`.

The following syntax demonstrates how to calculate the correlation and apply it as a text label using the `plt.text()` function, which allows precise coordinate placement:

```
import scipy
```

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

```
import seaborn as sns
```

```
#calculate correlation coefficient between x and y
```

```
r = scipy.stats.pearsonr(x=df.x, y=df.y)
```

```
#create scatterplot
sns.scatterplot(data=df, x=df.x, y=df.y)

#add correlation coefficient to plot
plt.text(5, 30, 'r = ' + str(round(r, 2)))
```

The critical element here is `plt.text()`. The first two arguments (5, 30 in this example) define the coordinates where the text will appear on the plot, based on the current axis limits. The third argument is the string content, which dynamically includes the calculated, rounded correlation coefficient `r`.

Prerequisites: Preparing the Data Structure

Before any statistical analysis or visualization can occur, the data must be loaded into a suitable structure, typically a [pandas DataFrame](#). DataFrames provide robust indexing and easy access to column data, which is essential when feeding variables into statistical functions and plotting routines. We will use a hypothetical dataset representing basketball player statistics to illustrate the process.

Suppose we are interested in the relationship between the number of points scored and the number of assists recorded by various players. This analysis requires two numerical series (columns) extracted from the DataFrame. Ensuring the data is clean and correctly structured prevents errors during the correlation calculation phase.

The example below demonstrates the creation of the sample DataFrame, containing variables for 'team', 'points', and 'assists'.

import pandas as pd

```
#create DataFrame
df = pd.DataFrame({'team': ,
'points': ,
'assists': })
```

```
#view DataFrame
print(df)
```

```
team points assists
0 A 12 4
1 A 11 7
2 A 18 7
```

3 A 15 8
4 B 14 9
5 C 20 10
6 C 25 10
7 C 24 12
8 D 32 10
9 D 30 15

This DataFrame serves as the foundation for our visualization. The columns 'assists' and 'points' will be treated as the independent (X) and dependent (Y) variables, respectively, in the subsequent statistical and graphical operations. The data clearly shows variability, suggesting a relationship worth investigating through the [correlation coefficient](#).

Calculating the Pearson Correlation Coefficient

Once the data is prepared, the next critical step is calculating the correlation. We utilize the `pearsonr()` function from the `scipy.stats` module. This function takes two array-like objects (our DataFrame columns) and returns a tuple where the first element is the coefficient r and the second is the two-tailed p-value.

We only require the correlation value, so we access it using index `[0]` after the function call. It is standard practice to store this value in a variable, often named `r`, which will then be converted to a string for plotting annotation. To improve readability on the plot, it is highly recommended to round the coefficient to two or three decimal places using Python's built-in `round()` function.

The code snippet below integrates the setup, calculation, and visualization steps, leading to the final annotated scatterplot.

Generating the Seaborn Scatterplot and Annotation

The `Seaborn` `scatterplot()` function is used to create the visualization. It is a highly versatile function built on top of Matplotlib, allowing for easy mapping of data columns to the visual axes. After the plot is generated, we use the `Matplotlib` `plt.text()` function to overlay the statistical result.

The coordinates (5, 30) passed to `plt.text()` are crucial; they determine where the text string is anchored on the plot. These values should be chosen carefully to ensure the annotation does not overlap with data points or other critical plot elements. Typically, a position in the top-right corner of the plot is preferred for displaying summary statistics.

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

```
import seaborn as sns
```

```
#calculate correlation coefficient between assists and points
```

```
r = scipy.stats.pearsonr(x=df.assists, y=df.points)
```

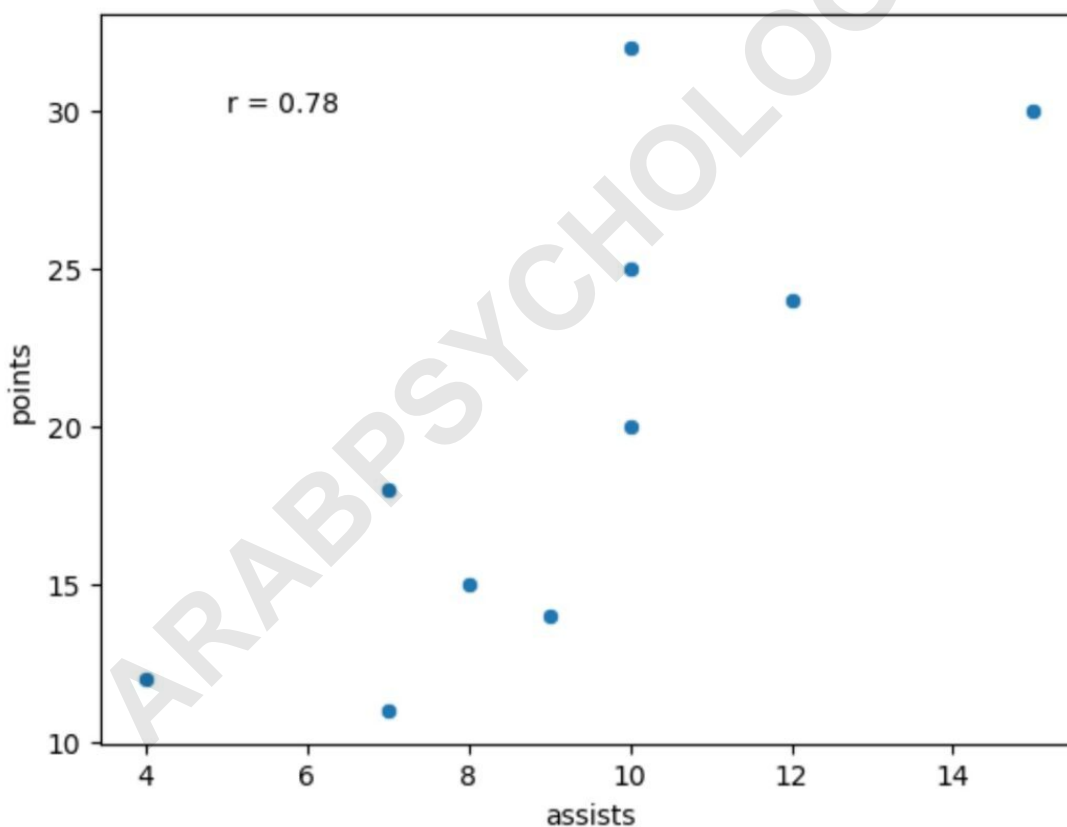
```
#create scatterplot
```

```
sns.scatterplot(data=df, x=df.assists, y=df.points)
```

```
#add correlation coefficient to plot
```

```
plt.text(5, 30, 'r = ' + str(round(r, 2)))
```

Executing this code yields the scatterplot visualization, complete with the calculated correlation coefficient displayed prominently. The resulting plot visually confirms the relationship between assists and points.



Upon reviewing the output, we observe that the Pearson correlation coefficient, denoted as r , is approximately **0.78**. This value indicates a strong, positive linear relationship: as the number of assists increases, the number of points scored generally also increases. The annotation confirms this quantitative relationship directly on the graphic.

Customizing Annotation Precision and Style

Data visualization best practices often require flexibility in presentation. When annotating a plot with statistical results, the analyst may need to adjust both the numerical precision and the visual styling of the text. For instance, sometimes four decimal places are needed for highly precise scientific reporting, or a larger font size is necessary for audience visibility.

We control the numerical precision by modifying the argument passed to the `round()` function. By changing `round(r, 2)` to `round(r, 4)`, we increase the decimal places shown. To adjust the appearance, the `plt.text()` function accepts optional keyword arguments such as `fontsize` or `color` to control the visual aesthetics of the label.

The following example demonstrates how to round the correlation coefficient to four decimal places and significantly increase the font size of the annotation text using the `fontsize` parameter.

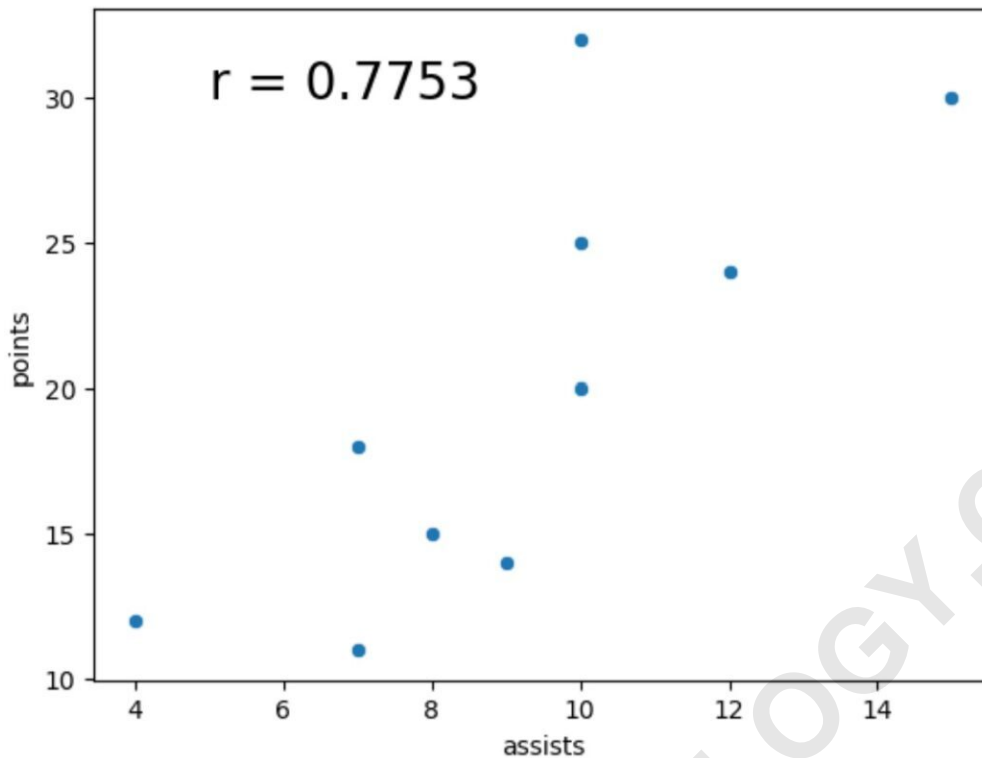
```
import scipy
import matplotlib.pyplot as plt
import seaborn as sns

#calculate correlation coefficient between assists and points
r = scipy.stats.pearsonr(x=df.assists, y=df.points)

#create scatterplot
sns.scatterplot(data=df, x=df.assists, y=df.points)

#add correlation coefficient to plot with custom formatting
plt.text(5, 30, 'r = ' + str(round(r, 4)), fontsize=20)
```

This customization demonstrates the power of utilizing [Matplotlib](#) functions directly on a Seaborn plot, allowing for highly tailored visual outputs that convey both the data distribution and precise statistical findings effectively.



As evident in the updated visualization, the correlation coefficient is now displayed with enhanced precision (e.g., 0.7761, depending on the exact float value) and significantly larger font, ensuring that the quantitative measure of relationship strength is immediately noticeable to the viewer. This level of customization is invaluable when preparing plots for publications or detailed reports.

Summary of Key Functions and Considerations

Successfully annotating a Seaborn scatterplot with a correlation coefficient relies on the seamless interaction of three primary libraries: pandas for data handling, SciPy for statistical calculation, and Matplotlib/Seaborn for visualization and annotation. Maintaining clean code by separating the calculation step from the visualization step is key to generating reproducible and readable plots.

When selecting the annotation coordinates for `plt.text()`, always verify that the chosen location (e.g., top-left, top-right) is clear of any data points that might obscure the text. Furthermore, remember that while the Pearson coefficient measures linear correlation, non-linear relationships require different measures or transformations before correlation analysis is appropriate.

For advanced visualization needs, explore the official documentation for additional arguments that can be passed to both `sns.scatterplot()` and `plt.text()`, such as alpha transparency, text bounding boxes, and different marker styles.

Further Resources for Advanced Visualization

To deepen your expertise in combining statistical computation with advanced data visualization techniques, consult the following authoritative documentation:

Official [Seaborn scatterplot\(\)](#) Documentation: Details all available parameters for customizing the appearance of the scatterplot.

Official [SciPy Statistics Module](#) Reference: Comprehensive guide to statistical tests and correlation functions available in SciPy.

[Matplotlib Text Tutorial](#): Provides extensive examples on text placement, rotation, and formatting within plots.

ARABPSYCHOLOGY.COM