

How can a Bell Curve be created in Python?

Authored by
stats writer

April 16, 2024

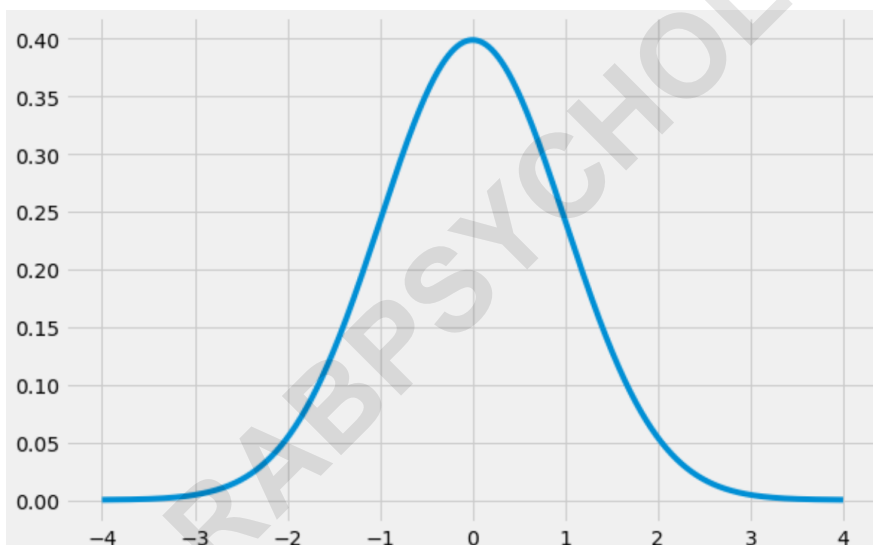
RECOMMENDED CITATION

stats writer (2024). *How can a Bell Curve be created in Python?*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=136115>

A Bell Curve, also known as a normal distribution, can be created in Python using the NumPy library. This can be achieved by first importing the necessary libraries, such as NumPy and Matplotlib, and then using the NumPy's random module to generate a set of normally distributed random numbers. These numbers can then be plotted on a graph using Matplotlib, with the x-axis representing the values and the y-axis representing the frequency or occurrence of those values. By adjusting the mean and standard deviation of the randomly generated numbers, the shape and position of the Bell Curve can be altered. This allows for the creation of a customized Bell Curve in Python, which can be useful in visualizing and analyzing data that follows a normal distribution.

Make a Bell Curve in Python

A "bell curve" is the nickname given to the shape of a normal distribution, which has a distinct "bell" shape:



This tutorial explains how to make a bell curve in Python.

How to Create a Bell Curve in Python

The following code shows how to create a bell curve

using the numpy, scipy, and matplotlib libraries:

```
import numpy as np
```

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

```
from scipy.stats import norm
```

```
#create range of x-values from -4 to 4 in increments of  
.001
```

```
x = np.arange(-4, 4, 0.001)
```

```
#create range of y-values that correspond to normal pdf  
with mean=0 and sd=1
```

```
y = norm.pdf(x,0,1)
```

```
#define plot
```

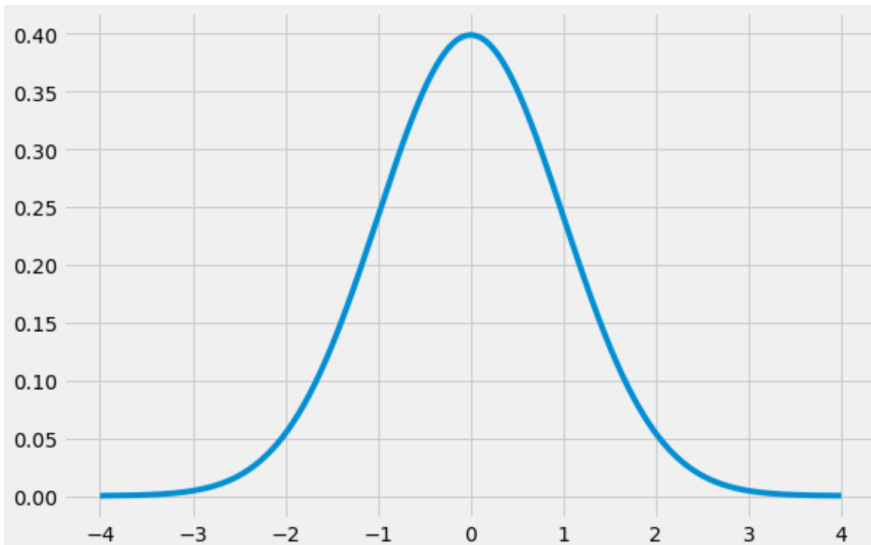
```
fig, ax = plt.subplots(figsize=(9,6))
```

```
ax.plot(x,y)
```

```
#choose plot style and display the bell curve
```

```
plt.style.use('fivethirtyeight')
```

```
plt.show()
```



How to Fill in a Bell Curve in Python

The following code illustrates how to fill in the area under the bell curve ranging from -1 to 1:

```
x = np.arange(-4, 4, 0.001)
```

```
y = norm.pdf(x,0,1)
```

```
fig, ax = plt.subplots(figsize=(9,6))
```

```
ax.plot(x,y)
```

```
#specify the region of the bell curve to fill in
```

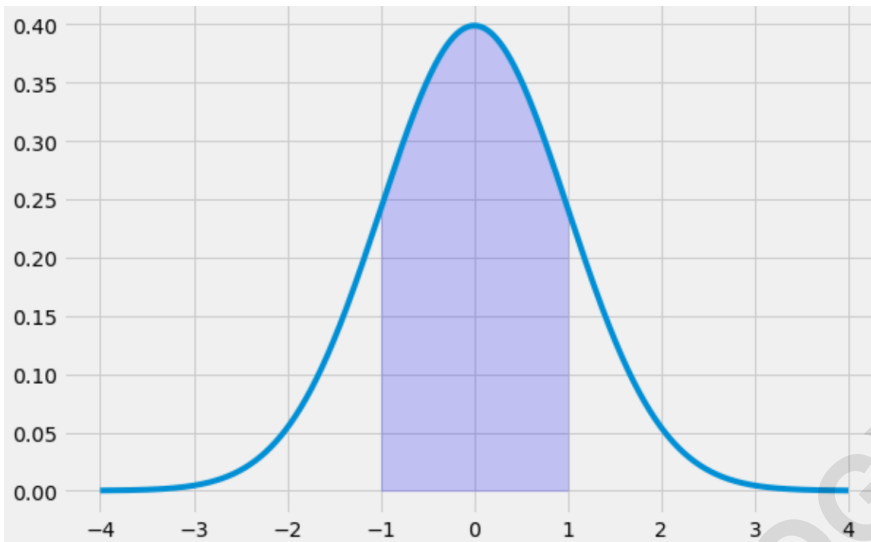
```
x_fill = np.arange(-1, 1, 0.001)
```

```
y_fill = norm.pdf(x_fill,0,1)
```

```
ax.fill_between(x_fill,y_fill,0, alpha=0.2, color='blue')
```

```
plt.style.use('fivethirtyeight')
```

plt.show()



Note that you can also style the graph in any way you'd like using the many matplotlib styling options. For example, you could use a "solar light" theme with a green line and green shading:

```
x = np.arange(-4, 4, 0.001)
```

```
y = norm.pdf(x,0,1)
```

```
fig, ax = plt.subplots(figsize=(9,6))
```

```
ax.plot(x,y, color='green')
```

#specify the region of the bell curve to fill in

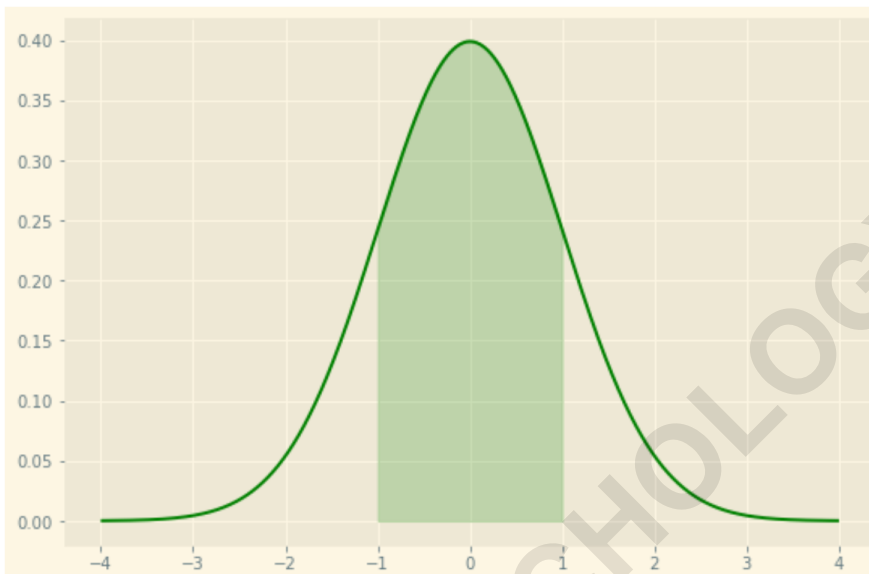
```
x_fill = np.arange(-1, 1, 0.001)
```

```
y_fill = norm.pdf(x_fill,0,1)
```

```
ax.fill_between(x_fill,y_fill,0, alpha=0.2, color='green')
```

```
plt.style.use('Solarize_Light2')
```

```
plt.show()
```



You can find the complete style sheet reference guide for matplotlib [here](#).

How to Make a Bell Curve in Excel