Histograms and Probability Density Function

In this lesson, we will learn about representing data using histograms and probability density functions.

WE'LL COVER THE FOLLOWING

Representing data

One of the most common ways to represent a data set is to draw a histogram. For a histogram, you count how many data points fall within a certain interval. For example, how many data points are between 5 and 6. These intervals are called bins. The bar graph of the number of data points in each bin is called a histogram. The function to compute and plot a histogram is called `hist()` and is part of the `matplotlib` package. The simplest way of plotting a histogram is to let `hist()` decide what bins to use; the default number of bins is `nbin=10`.

`hist()` even figures out where to put the limits of the bins. The `hist()` function creates a histogram graph and *returns* a tuple of three items:

1. The first item is an array of length `nbin` with the number of data points in each bin.
2. The second item is an array of length `nbin+1` with the limits of the bins.
3. The third item is a list of objects that represent the bars of the histogram; this is the least used item.

Let's draw a histogram:

```
import numpy as np
import matplotlib.pyplot as plt
```

```
import matplotlib.pyplot as plt
import numpy.random as rnd

data = rnd.normal(loc=6, scale=2, size=100)
hist_data = plt.hist(data)
plt.xlabel('bins')
plt.ylabel('number of data points')
print('number of data points in each bin:', hist_data[0])
print('limits of the bins:', hist_data[1])
```

In the code above, we make a histogram of 100 points drawn from a Normal distribution with mean 6 and a standard deviation of 2. Note that with a dataset of 100 points, the histogram doesn't look too much like the typical bell-shaped curve of a Normal distribution, even though the data points are actually drawn from a real Normal distribution.

> 💡 Run the code several times to see how it changes with a new random set of 100 data points.

Improving the histogram

As you can see from the example above, the limits of the bins are not chosen as *convenient* numbers, but how are the bins selected?

`hist()` takes the minimum and maximum value of the data and divides it into `nbin` equal intervals. You normally want to specify the number of bins with the `bins` keyword, and the range (minimum and maximum limits of the bins) with the `range` keyword. If data values are outside this range, they are ignored.

In the code below, 12 bins are chosen equally spaced from 0 to 12. Note that we use the same date as for the graph above, but by simply choosing different bins the histogram looks quite different.

```
import numpy as np
import matplotlib.pyplot as plt
import numpy.random as rnd

data = rnd.normal(loc=6, scale=2, size=100)
hist_data = plt.hist(data, bins=12, range=(0, 12))
print('number of data points in each bin:', hist_data[0])
```

```
print('limits of the bins:', hist_data[1])
plt.xlabel('bins')
plt.ylabel('number of data points');
```

Probability density function

A line representing the probability density function of the underlying normal distribution may be added as well.

First, we import the `norm` class from the `scipy.stats` package. Then we call the `norm.pdf` function (pdf stands for probability density function) to compute the values of the normal distribution given three arguments:

1. The x values for normal distribution
2. The mean
3. The standard deviation

Let's add the underlying Normal distribution to the histogram we just created. The one thing we have to change in the histogram is the vertical axis. In the graph above, the vertical axis shows the number of data points. We need to normalize this so that the vertical axis gives the probability that a data point lies in a bin.

> 💡 The histogram may be normalized by specifying the `density=True` keyword.

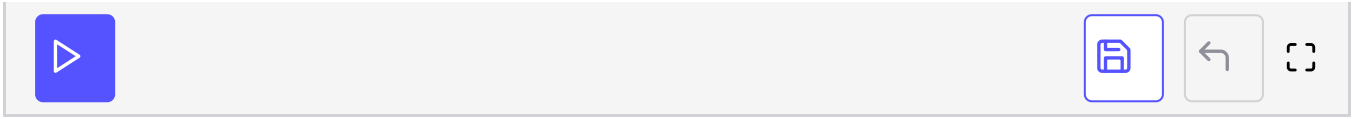
```
import numpy as np
import matplotlib.pyplot as plt
import numpy.random as rnd
from scipy.stats import norm

mu = 6
sig = 2
n = 100

data = rnd.normal(loc=mu, scale=sig, size=n)
hist_data = plt.hist(data, bins=12, range=(0, 12), density=True)
print('number of data points in each bin:', hist_data[0])
print('limits of the bins:', hist_data[1])
plt.xlabel('bins')
plt.ylabel('number of data points');

# plotting pdf
```

```
x = np.linspace(0, 12, 100)
y = norm.pdf(x, loc=mu, scale=sig) # mu=6, sig=2
plt.plot(x, y, 'r')
plt.xlabel('value')
plt.ylabel('probability');
```



> **Do it yourself:** Increase the value of `n` in the code below and see how the histogram fits inside the probability distribution function.

In the next lesson, we will learn about percentiles.