Histograms in Python

Hugo Touchette

Last updated: 13 Nov 2019

Python 3

Reference:

- https://www.datacamp.com/community/tutorials/histograms-matplotlib (https://www.datacamp.com/community/tutorials/histograms-matplotlib)
- https://docs.scipy.org/doc/numpy/reference/generated/numpy.histogram.html (https://docs.scipy.org/doc/numpy/reference/generated/numpy.histogram.html)

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
data=np.array([np.random.randn() for i in range(1000)])
```

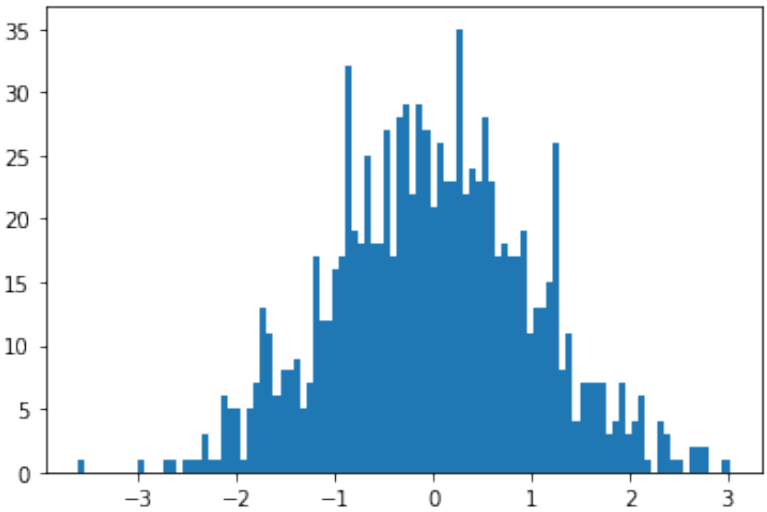
Simple way: hist function

The function hist is part of the Matplotlib library. By default, hist gives the histogram values in addition to the plot.

In [3]:
```
plt.hist(data,100)
```

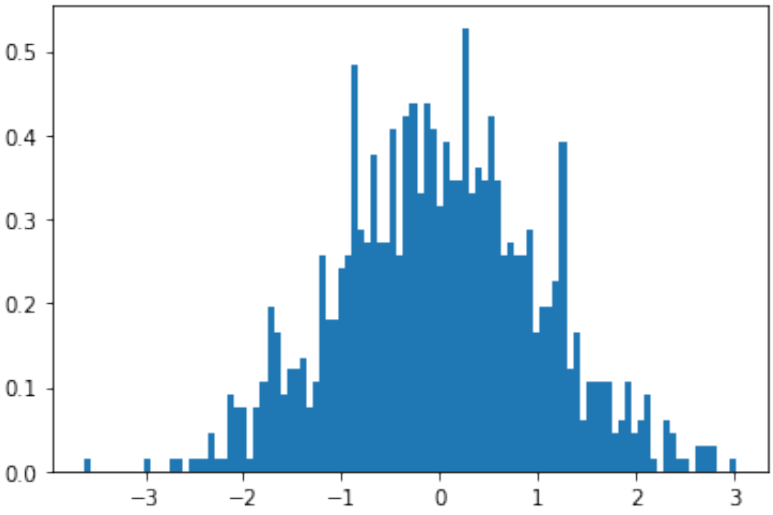
Out[3]:
```
(array([ 1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,
         1.,  1.,  0.,  1.,  1.,  1.,  3.,  1.,  1.,  6.,  5.,  5.,
         5.,  7., 13., 11.,  6.,  8.,  8.,  9.,  5.,  7., 17., 12.,
        16., 17., 32., 19., 18., 25., 18., 18., 27., 17., 28., 29.,
        29., 27., 21., 26., 23., 23., 35., 22., 24., 23., 28., 23.,
        18., 17., 17., 19., 11., 13., 13., 15., 26.,  8., 11.,  4.,
         7.,  7.,  7.,  3.,  4.,  7.,  3.,  4.,  6.,  1.,  0.,  4.,
         1.,  1.,  0.,  2.,  2.,  2.,  0.,  0.,  1.]),
 array([-3.61579361, -3.54948806, -3.48318252, -3.41687697, -3.35057142,
        -3.28426588, -3.21796033, -3.15165479, -3.08534924, -3.01904369,
        -2.95273815, -2.8864326 , -2.82012705, -2.75382151, -2.68751596,
```

```
        -2.62121041, -2.55490487, -2.48859932, -2.42229377, -2.355
98823,
        -2.28968268, -2.22337713, -2.15707159, -2.09076604, -2.024
46049,
        -1.95815495, -1.8918494 , -1.82554385, -1.75923831, -1.692
93276,
        -1.62662722, -1.56032167, -1.49401612, -1.42771058, -1.361
40503,
        -1.29509948, -1.22879394, -1.16248839, -1.09618284, -1.029
8773 ,
        -0.96357175, -0.8972662 , -0.83096066, -0.76465511, -0.698
34956,
        -0.63204402, -0.56573847, -0.49943292, -0.43312738, -0.366
82183,
        -0.30051628, -0.23421074, -0.16790519, -0.10159965, -0.035
2941 ,
         0.03101145,  0.09731699,  0.16362254,  0.22992809,  0.296
23363,
         0.36253918,  0.42884473,  0.49515027,  0.56145582,  0.627
76137,
         0.69406691,  0.76037246,  0.82667801,  0.89298355,  0.959
2891 ,
         1.02559465,  1.09190019,  1.15820574,  1.22451129,  1.290
81683,
         1.35712238,  1.42342792,  1.48973347,  1.55603902,  1.622
34456,
         1.68865011,  1.75495566,  1.8212612 ,  1.88756675,  1.953
8723 ,
         2.02017784,  2.08648339,  2.15278894,  2.21909448,  2.285
40003,
         2.35170558,  2.41801112,  2.48431667,  2.55062222,  2.616
92776,
         2.68323331,  2.74953886,  2.8158444 ,  2.88214995,  2.948
45549,
         3.01476104]),
 <a list of 100 Patch objects>)
```



To only get the plot, use the show command. Also use the 'density' option to make sure the histogram is normalised as a probability density.

In [4]:
```
1 plt.hist(data, 100, density=True)
2 plt.show()
```



Better way: histogram function

Numpy has a histogram command which is quite comprehensive:

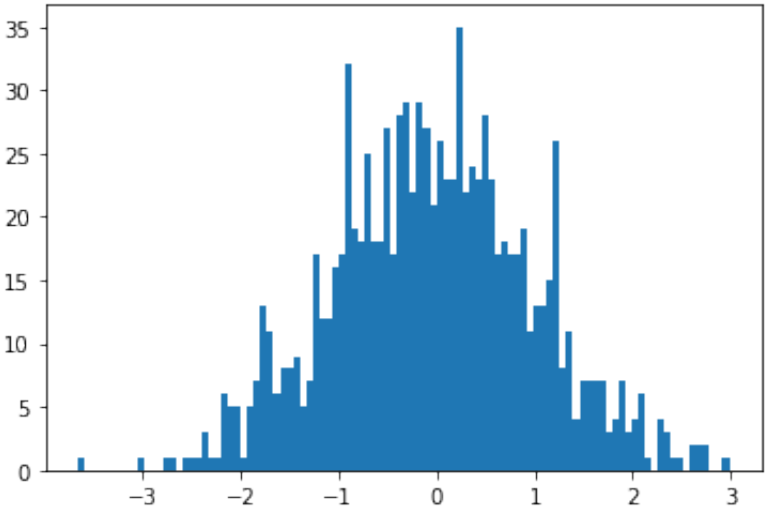
In [5]:
```
1 np.histogram(data, 10)
```

Out[5]:
```
(array([  2,   8,  55, 100, 219, 257, 197, 111,  39,  12]),
 array([-3.61579361, -2.95273815, -2.28968268, -1.62662722, -0.963
57175,
        -0.30051628,  0.36253918,  1.02559465,  1.68865011,  2.351
70558,
         3.01476104]))
```

The first array is the bin counts, not normalised. The second array contains the $n + 1$ points that specify the bases of the n bins we want, so that array has one more point than the bin count array. With this in mind, here's how we can plot the result as a bar chart:

In [7]:

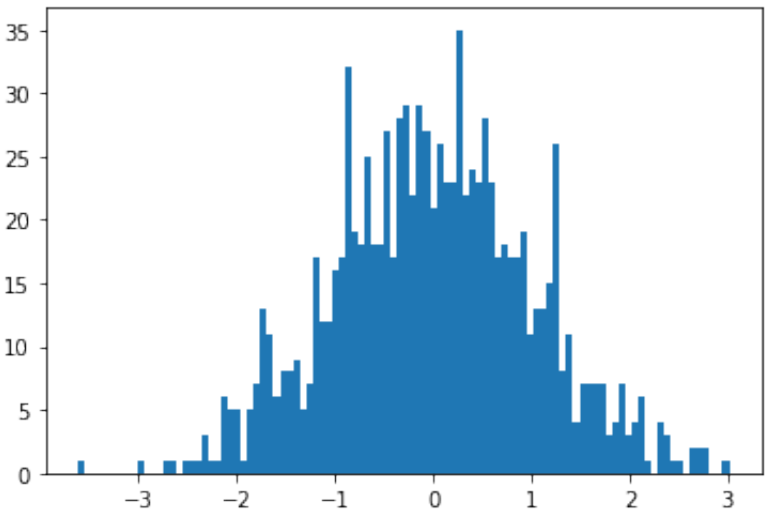
```
nbins = 100
hist, bin_spec = np.histogram(data, nbins)
a, b = min(bin_spec), max(bin_spec)
dx = (b-a)/nbins
plt.bar(bin_spec[:-1], hist, width = dx)
plt.show()
```



The directive bin_spec[:-1] takes all elements except the last, so we are positioning the bins with the left coordinates. To position the bins with center points, push the position by $dx/2$:

In [8]:

```
nbins = 100
hist, bin_spec = np.histogram(data, nbins)
a, b = min(bin_spec), max(bin_spec)
dx = (b-a)/nbins
plt.bar(bin_spec[:-1]+dx/2.0, hist, width = dx)
plt.show()
```



As it is, the histogram is not normalised as a probability density. To force that normalisation, use again the 'density' option:

In [9]:

```
nbins = 100
hist, bin_spec = np.histogram(data, nbins, density=True)
a, b = min(bin_spec), max(bin_spec)
dx = (b-a)/nbins
plt.bar(bin_spec[:-1]+dx/2.0, hist, width = dx)
plt.show()
```