python Intermediate

Statistics Tools

Mauricio Sevilla

email= j.sevillam@uniandes.edu.co

29.04.19

Introduction

Data can be result from many sources depending on the problem. We may say that, there are many (almost every) experiments such that the result is not completely defined.

What we mean is that, if we repeat the exact same experiment, under the exact same preparation, the result is slightly different (That is why errorbars are SO important!).

Some definitions

We may now define some concepts, in order to talk in the same way during the class

- A ***random variable*** is a variable whose value is determined by the outcome of a random experiment.
- A ***discrete random variable*** is one whose set of discrete values (Number of ...).
- A ***continuous random variable*** is one whose set of values is that can be continuous (Temperature).

Eventhought the results are not the same, after many repetitions, the results get concentrated on a specific value and with a characteristic spreading.

These two characteristics can describe the complete set of repetitions.

That is what statistics do, allows us to make descriptions of large sets of data, from generalities representated on a small set of characteristics.

One of the most widely used characteristics is the average!. When we want to describe several data with only one number, for instance, the temperature during the week, more or less one can say only one value. (The average).

A different but more ilustrative example is the grading. Because not all of the grades have the same weight.

The weights mean that, some values for any reason, are *more important* than others.

Distributions

Is the same idea when we are talking about random varables. There are different ways these weights can affect the result, usually when we are grading, the result is divided by 100, because the weigths are percentages, but what we are going to have in general is that the weights have to satisfy some conditions so that, they can be named a *probability distribution*

By definition, this *probability distribution* is a real function which provides the probability of an specific outcome on a experiment.

They must satisfy that,

- $p(x)$ is a real function.
- $p(x)$ is a positive function.
- $p(x)$ sums 1. There are two cases,
    - Discrete Case
    $$\sum_i p_i = 1$$
    - Continuous Case
    $$\int_D dx p(x) = 1$$

Expected Value

One can derive most of the things we are going to use from the expected value definition,

- Discrete case,
  $$E[X] = \sum_i p_i x_i$$
- Continuous case
  $$E[X] = \int_D dx p(x) x$$

where $p_i\left(p(x)\right)$ is the *probability* of the event $X = x_i(x)$

Let us explore some cases.

Dice probability

what is the probability of getting the number 1 when throwing a dice?

```
A:\\ There are 6 numbers on a dice, so that the probability is 1/6
```

But, all of them have the same probability.

This is called ***Uniform*** distribution.

But, what if i have two dices? (Like on some popular board games)

What is the probability of having a certain number?

Let us construct the possible combinations

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

In [2]:
```
combinations=np.array([])
for i in range(1,7):
    for j in range(1,7):
        combinations=np.append(combinations,i+j)
```

In [3]:
```
for i in range(len(combinations)):
    print((i%6)+1,(i//6)+1,combinations[i])
```

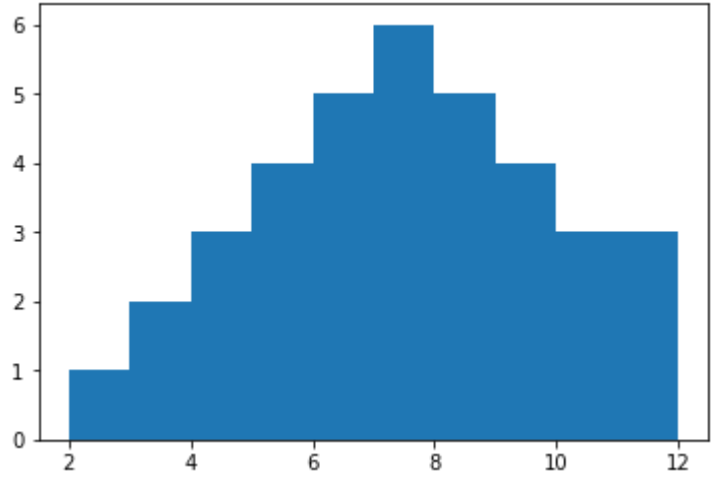
```
1 1 2.0
2 1 3.0
3 1 4.0
4 1 5.0
5 1 6.0
6 1 7.0
1 2 3.0
2 2 4.0
3 2 5.0
4 2 6.0
5 2 7.0
6 2 8.0
1 3 4.0
2 3 5.0
3 3 6.0
4 3 7.0
5 3 8.0
6 3 9.0
1 4 5.0
2 4 6.0
3 4 7.0
4 4 8.0
5 4 9.0
6 4 10.0
1 5 6.0
2 5 7.0
3 5 8.0
4 5 9.0
5 5 10.0
6 5 11.0
1 6 7.0
2 6 8.0
3 6 9.0
```

```
4 6 10.0
5 6 11.0
6 6 12.0
```

let us construct the histogram, a histogram is having a set of *boxes* with a certain size, looking how many *points* or *results* from a experiment goes in each box.

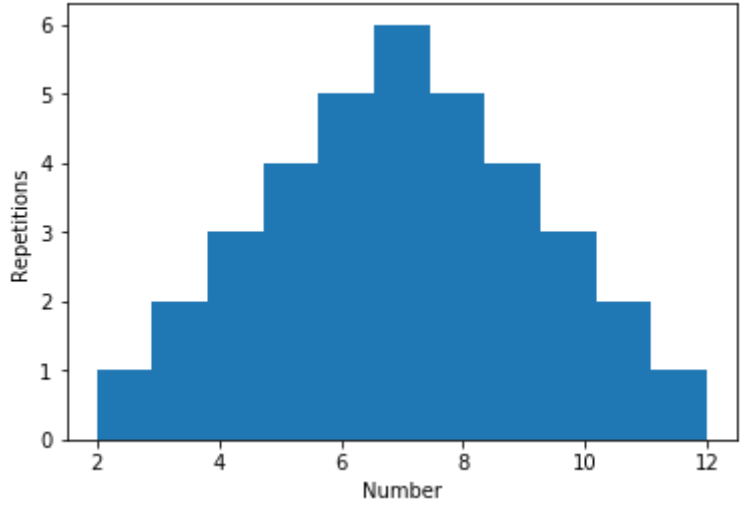
One can have a *bad* way to choose the boxes

In [4]:
```
plt.hist(combinations)
plt.show()
```



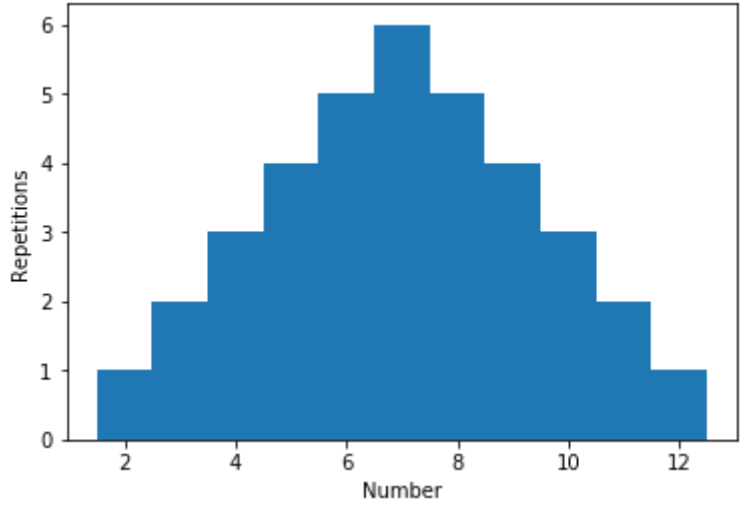
And once corrected,

In [5]:
```
plt.xlabel('Number')
plt.ylabel('Repetitions')
plt.hist(combinations,bins=11)
plt.show()
```



Or even better,

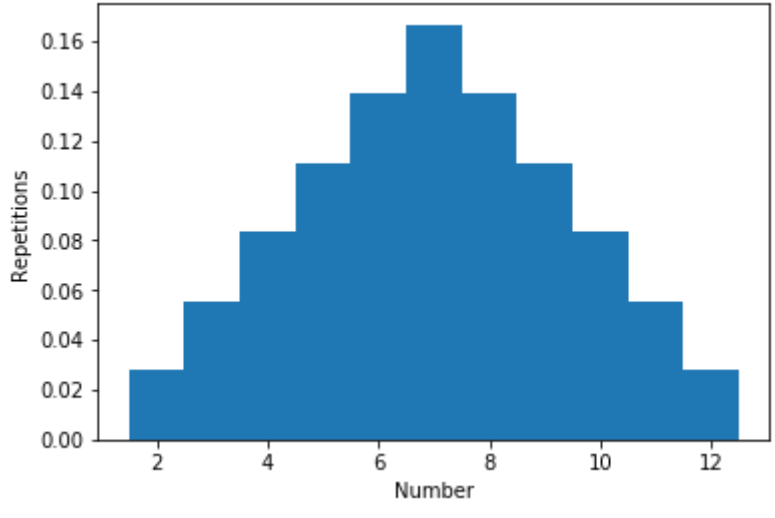
In [6]:
```
plt.xlabel('Number')
plt.ylabel('Repetitions')
plt.hist(combinations,bins=np.arange(1.5,13.5,1))
plt.show()
print(np.arange(1.5,13.5,1))
```




```
[ 1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5]
```

As the number 7 is repeated more than the others, we may say that is the *most probable* result.

In [7]:
```
plt.xlabel('Number')
plt.ylabel('Repetitions')
histogram=plt.hist(combinations,density=True,bins=np.arange(1.5,13.5,1))
plt.show()
```



In [8]:
```
print('#',' ','Probability [%]')
for i in range(11):
    print((histogram[1][:-1]+histogram[1][1:])[i]/2,100*histogram[0][i])
```

```
#   Probability [%]
2.0 2.7777777777777777
3.0 5.555555555555555
4.0 8.333333333333332
5.0 11.11111111111111
6.0 13.88888888888889
7.0 16.666666666666664
8.0 13.88888888888889
9.0 11.11111111111111
10.0 8.333333333333332
11.0 5.555555555555555
12.0 2.7777777777777777
```

In [9]:
```
print(histogram[0].sum())
```

```
1.0
```

Take a look at the `plt.hist` function!.

All the distribution are well defined by their so called *moments*.

We have already worked with them!, the first moment is the mean,

$$\mu = E[x]$$

And the rest of them are calculated as,

$$\mu^n = E[(x - \mu)^n]$$

For the first moments, `NumPy` + `SciPy` has implemented some functions. Look for them!.

Distribution Examples

Depending on the nature of the experiment, we may have different distributions.

There are some typical cases, which are those we'll explore today.

Uniform Distribution

The simplest case, all the options have the same probability, for instance, a fair dice.

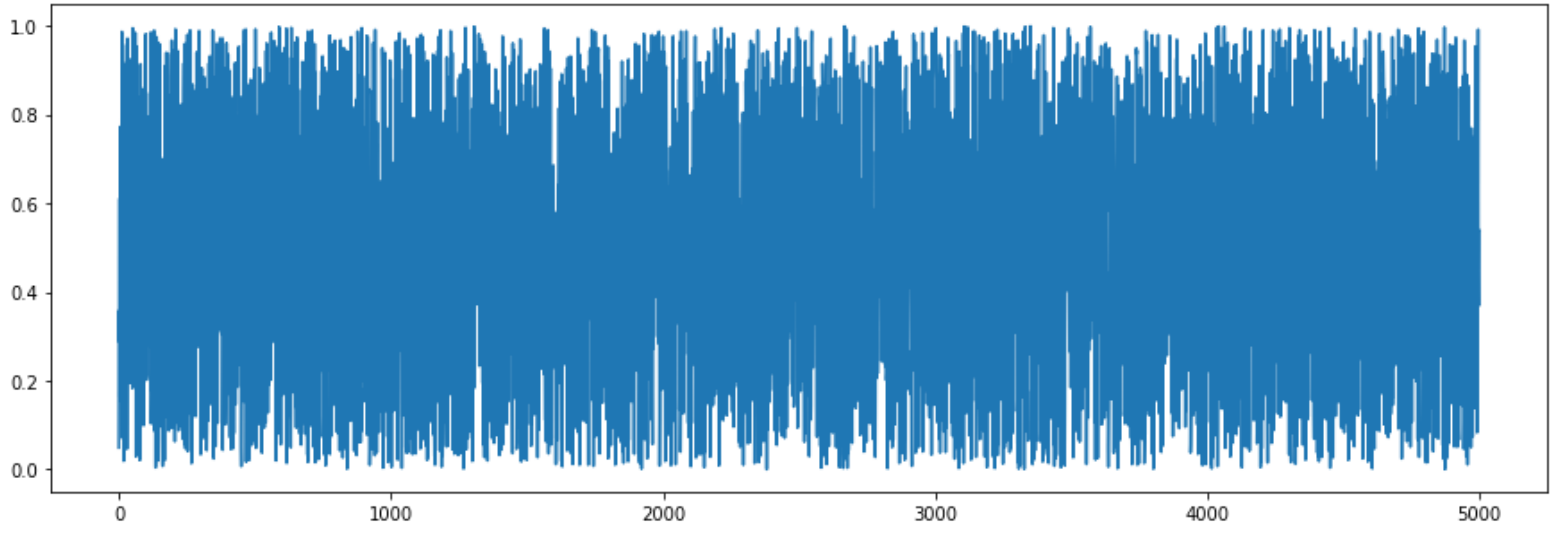
While computing, almost all random number generator (Which becomes a complet topic for a course!!), understand the Uniform distribution as a number r such that $r \in [0, 1)$.

Python is the case.

In [35]:
```
x=np.random.random(5000)
```

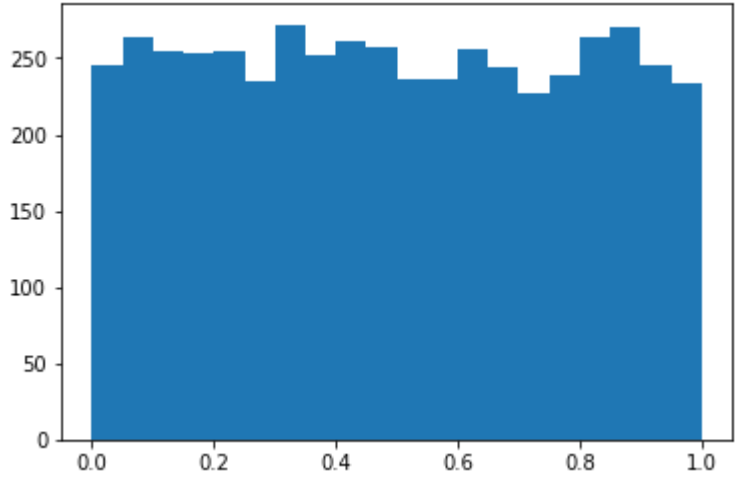
In [36]:
```
fig=plt.figure(figsize=(15,5))
ax=fig.add_subplot(111)
ax.plot(x)
```

Out[36]:
```
[<matplotlib.lines.Line2D at 0x26d4b07c908>]
```



In [37]:
```
plt.hist(x,bins=20)
```

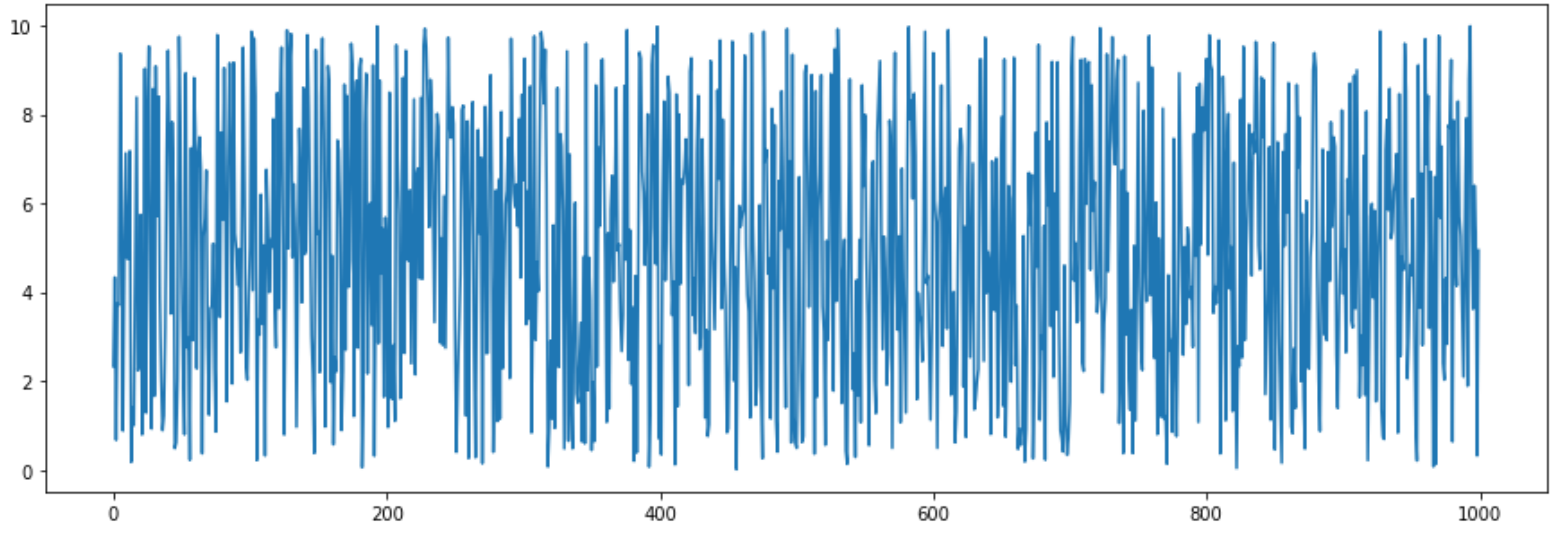
Out[37]:
```
(array([246., 264., 254., 253., 254., 235., 272., 252., 261., 257., 236.,
        236., 256., 244., 227., 239., 264., 270., 246., 234.]),
 array([8.16530396e-05, 5.00739704e-02, 1.00066288e-01, 1.50058605e-01,
        2.00050922e-01, 2.50043240e-01, 3.00035557e-01, 3.50027874e-01,
        4.00020192e-01, 4.50012509e-01, 5.00004826e-01, 5.49997144e-01,
        5.99989461e-01, 6.49981778e-01, 6.99974096e-01, 7.49966413e-01,
        7.99958730e-01, 8.49951048e-01, 8.99943365e-01, 9.49935682e-01,
        9.99928000e-01]),
 <a list of 20 Patch objects>)
```



In [38]:
```
x=np.random.uniform(0,10,1000)
```

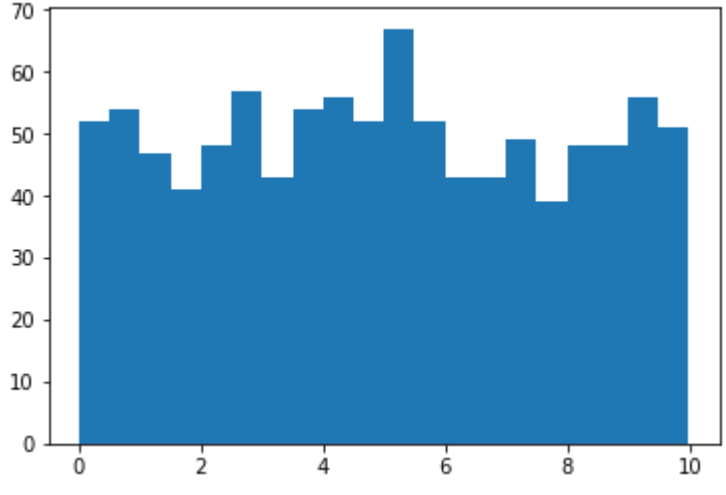
In [39]:
```
fig=plt.figure(figsize=(15,5))
ax=fig.add_subplot(111)
ax.plot(x)
```

Out[39]:
```
[<matplotlib.lines.Line2D at 0x26d4b315240>]
```



In [40]:
```
plt.hist(x,bins=20)
```

Out[40]:
```
(array([52., 54., 47., 41., 48., 57., 43., 54., 56., 52., 67., 52., 43.,
        43., 49., 39., 48., 48., 56., 51.]),
 array([0.01597785, 0.51463437, 1.01329089, 1.51194741, 2.01060392,
        2.50926044, 3.00791696, 3.50657348, 4.00522999, 4.50388651,
        5.00254303, 5.50119954, 5.99985606, 6.49851258, 6.9971691 ,
        7.49582561, 7.99448213, 8.49313865, 8.99179516, 9.49045168,
        9.9891082 ]),
 <a list of 20 Patch objects>)
```



Bernoulli Distribution

Takes value 1 with probability p and value 0 with probability $q = 1 - p$.

Answer to a yes or no question with a certain probability each value.

$$f(k;p) = \begin{cases} p & \text{if } k = 1 \\ q = 1 - p & \text{if } k = 0 \end{cases}$$

Binomial Distribution

Describes the number of successes in a series of independent Yes/No experiments all with the same probability of success.

Many Bernoulli experiments (All with the same probability!!)

$$f(k,n,p) = \Pr(k;n,p) = \Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

for $k = 0, 1, 2, \ldots, n$,where

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Poisson distribution

Describes the number of successes in a series of independent Yes/No experiments with different success probabilities.

$$P(k \text{ events in interval }) = e^{-\lambda} \frac{\lambda^k}{k!}$$

Maxwell Boltzmann distribution

It was first defined and used for describing particle speeds in idealized gases! (Kinetic theory)

$$f_{\mathrm{p}}\left(p_x, p_y, p_z\right) = \left(2\pi m k_B T\right)^{-3/2} \exp\left[-\frac{p_x^2 + p_y^2 + p_z^2}{2 m k_B T}\right]$$

Geometric distribution,

Describes the number of attempts needed to get the first success in a series of independent Bernoulli trials(the number of losses before the first success).

$$\Pr(X = k) = (1-p)^{k-1}p$$

Hypergeometric distribution

Describes the probability of k successes in n draws, without replacement, from a finite population of size N that contains exactly K objects with that feature. In contrast, the binomial distribution describes the probability of k successes in n draws with replacement.

$$p_X(k) = \Pr(X = k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$$

Dirac delta function

Although not strictly a function, is a limiting form of many continuous probability functions. It represents a discrete probability distribution concentrated at 0.

Widely used when defining charge densities.

Note: This is not a strict definition!!

$$\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases}$$

And

$$\int_{-\infty}^{\infty} \delta(x) dx = 1$$

Wigner semicircle distribution

Is important in the theory of random matrices. (And in our Homework this week!).

$$f(x) = \frac{2}{\pi R^2}\sqrt{R^2 - x^2}$$

for $-R \leq x \leq R$, and $f(x) = 0$ if $|x| > R$

Normal distribution

The most used distribution by far!.

There are several reasons for that, and the name suggest it.

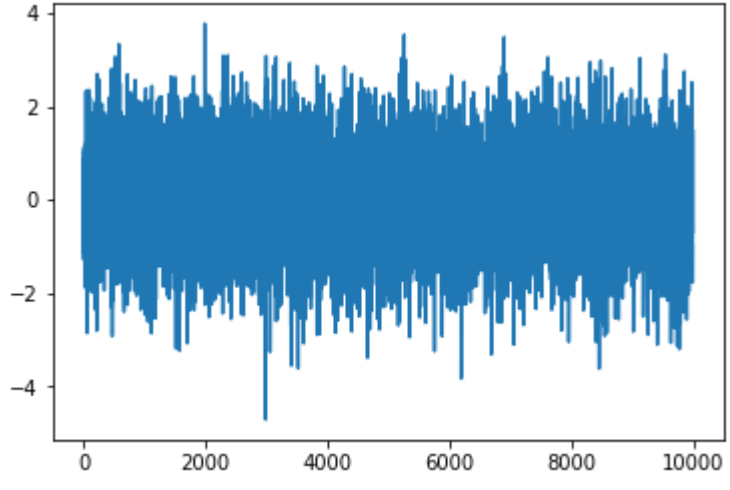
It does not look *normal*

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

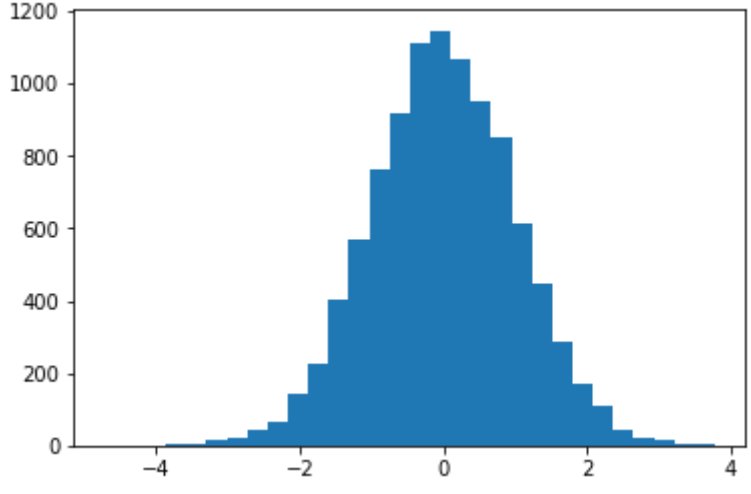
In [51]:
```
x=np.random.normal(0,1,10000)
```

In [52]:
```
plt.plot(x)
```

Out[52]:
```
[<matplotlib.lines.Line2D at 0x26d4b659358>]
```

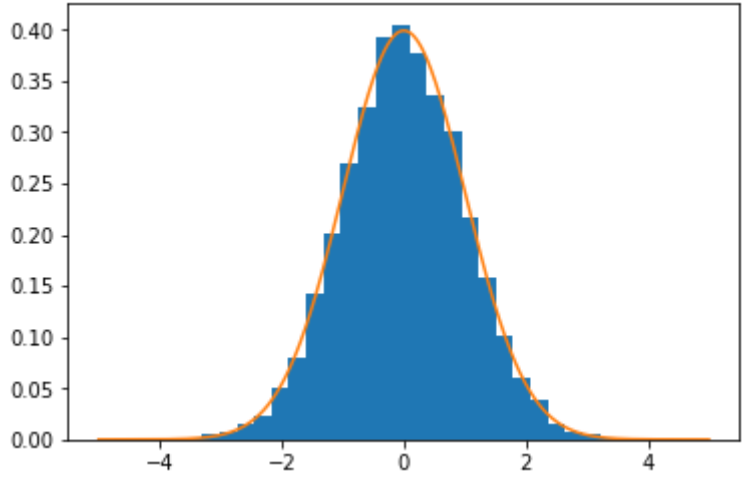


In [56]:
```
plt.hist(x,bins=30);
```



In [77]:
```
x_plot=np.linspace(-5,5,1001)
def gauss(mu,sigma,x):
    return np.exp(-0.5*(x-mu)**2/sigma**2)/(sigma*np.sqrt(2*np.pi))
plt.hist(x,density=True,bins=30);
plt.plot(x_plot,gauss(0,1,x_plot))
```

Out[77]:
```
[<matplotlib.lines.Line2D at 0x26d379fcc18>]
```



The cumulative function is

$$\frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]$$

This will allow us to get the probability of x in an interval instead than a value.

Let us define it

In [87]:
```
from scipy import special as sp
def int_gauss(mu,sigma,x):
    return (1+sp.erf((x-mu)/(sigma*np.sqrt(2))))/2.
```

In [88]:

```
print(int_gauss(0,1,1)-int_gauss(0,1,-1))
print(int_gauss(0,1,2)-int_gauss(0,1,-2))
print(int_gauss(0,1,3)-int_gauss(0,1,-3))
```

```
0.6826894921370859
0.9544997361036416
0.9973002039367398
```