Herramientas Computacionales para Ciencias
Homework 7

Mauricio Sevilla*

18/03/2019

Rules

Note Read carefully the complete homework before starting, so you will know what is the results you have to get!.
This week we are going to concentrate on `NumPy` functions and `array`s. On this assignment you will have to construct some functions and use the `NumPy` structure to evaluate it's performance.There is an additional point that must be developed on class before sending the assignment. This part must be saved on a jupyter Notebook named as your UniAndes username.

`numpy.array`s become particularly useful when working with data, this task will be continued next week when we learn how to use `matplotlib` and use real experimental data.

[1.0/4.0] Getting the Data Ready

We are going to *simulate* to have a signal, to do so we are going to do the following

- **[0.2/1.0]** Create a `numpy.array` with 1000 numbers from 0 to 10 equally spaced including the limits. (I will call this array `x` for the further instructions.)
- **[0.2/1.0]** Use this recently created array to create a different one with the sine. (**Hint:** Use the function `np.sin`). (I will call this array `y` for the further instructions.).
- **[0.6/1.0]** the function `np.random.random(N)` creates an array of N random numbers on the interval $(0, 1]$. Use this function to add some noise to the array `y` by adding at every cell a random number on the interval $(-0.2, 0.2]$. (**Hint:** Use that the sum of two arrays is done component by component.).

To plot this, you can use the following lines,

```
import matplotlib.pylab as plt
plt.plot(x,y)
plt.show()
```

The result, should be something like

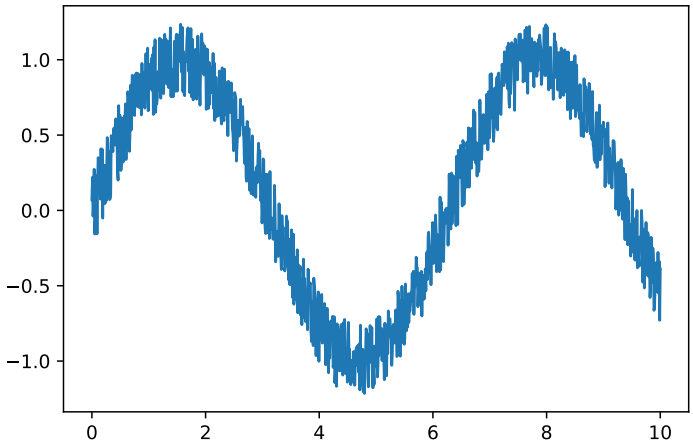

Figure 1: Sinusoidal signal with some noise.

Note: We will learn how to customize these kind of plots next week!.

*email=j.sevillam@uniandes.edu.co

[2.0/4.0]Characterizing the Amplitudes

Once we have the data we are going to work with on arrays, we can characterize them, one of the most popular ways to do it, is by calculate their moments, given a set of numbers $\mathbf{x}$ its firsts moments are defined as

Mean

$$\mu(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1}$$

Variance

$$\sigma(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} (x_i - \mu)^2 \tag{2}$$

Skewness

$$\tau(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} (x_i - \mu)^3 \tag{3}$$

Kurtosis

$$\kappa(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} (x_i - \mu)^4 \tag{4}$$

This give us the information of the highs, so it is a good idea to plot a histogram, which we also will discuss later. Once matplotlib is imported, you can use it to build a histogram as simply as,

```
plt.hist(y,bins=25)
plt.show()
```

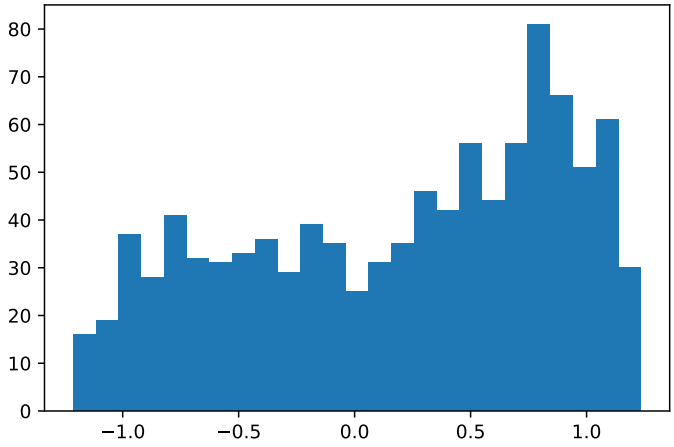

Figure 2: Histogram of highs.

- **[1.0/2.0]** Write 4 functions, one for every of the four firsts moments. **Hint:** use the function `np.sum`.
- **[0.5/2.0]** Read the documentation of the function `np.percentile` and evaluate the percentile 5 and 95.
- **[0.5/2.0]** Write a small explanation of the previous commands.

[1.0/4.0] Transforming the Data

- **[0.5/1.0]** Explore the function `np.argmax` and use the result as a parameter to print the maximum value saved on `y` and its correspondent value on `x`.
- **[0.25/1.0]** Transform the data saving on a different array the result of using `np.sort(y)` (I will call this array `y2`) when plotting both arrays with the same x, we get something like

```
plt.plot(x,y)
plt.plot(x,y2)
plt.show()
```

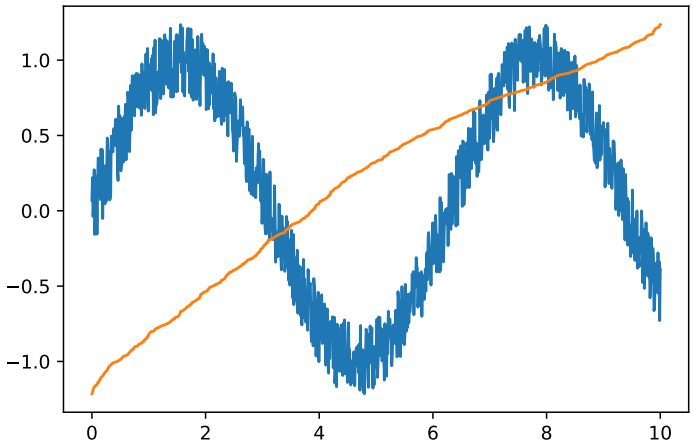

Figure 3: Transformed (Orange) and Original (Blue) data.

- **[0.25/1.0]** Create a new array with the product of `y` times `y2` and plot them (Just adding the line `plt.plot(x,y3)` to the previous plot commands.). you may get something like

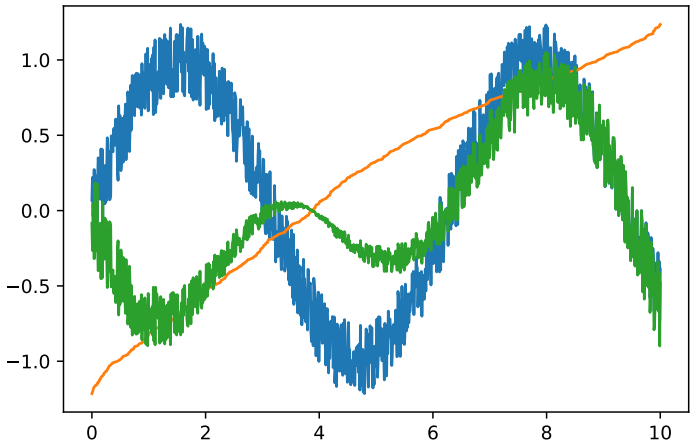

Figure 4: Final plot.