TP1- ROUSSEAU Data Analysis

October 4, 2018

1 Data Analysis - TP 1

ISEP Paris – October 2nd, 2018 Matthieu Rousseau (solo)

This document has been done using python on Jupyter Notebook with the librairies:

- Numpy to manipulate arrays
- matplotlib to plot graphics
- pandas to import csv
- scipy for mathematicals usage
- maths for sqrt, pi, exp

2 Exercice A : Discrete series

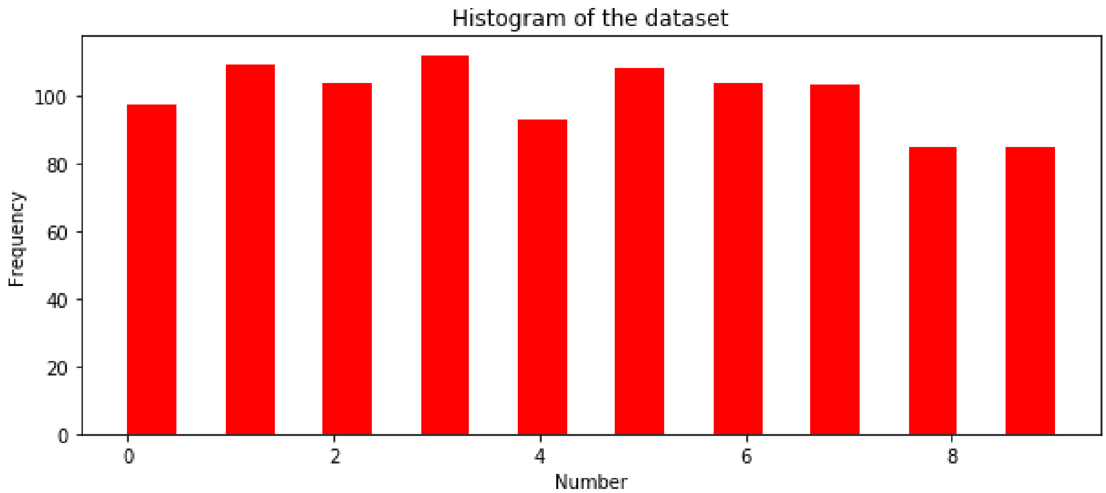
In [34]:
```
#Import of libraries

import matplotlib as plt
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy import stats
from math import sqrt,pi,exp
```

In [35]:
```
#We Generate a random series of 1000 element and we plot them with matplotlib

A = np.random.randint(0,10,1000)
figure,axe = plt.subplots(figsize=(10,4))
plt.hist(A,bins=19,color='red')
axe.set_title("Histogram of the dataset")
axe.set_xlabel("Number")
axe.set_ylabel("Frequency")
```

Out[35]:
```
Text(0,0.5,'Frequency')
```



In [36]:
```
#Compute the mean/max/mode

#Compute the mean : the mean is the sum of all the element
#divided by the len of the array

mean = sum(A)/len(A)

#Compute the median : we select the mean between the central element
#and central element +1 and we convert it into an integer

A_sorted = np.sort(A)
index1 = len(A_sorted)/2
index2=len(A_sorted)/2+1
index_median = (index1+index2)/2
median=A_sorted[int(index_median)]

#Compute the mode : we use numpy.unique who return a 2D array with the number of
#occurence of each element

unique, counts = np.unique(A, return_counts=True)
mode = counts.argmax()

#Print the result
print("The mean is {0}".format(mean))
print("The median is {0}".format(median))
print("The mode is {0}".format(mode))
```

```
The mean is 4.355
The median is 4
The mode is 3
```

In [37]:
```
#We compute the mean, median and the mode using numpy function
mean_withlib = np.mean(A)
median_withlib = np.median(A)
mode_withlib = stats.mode(A)[0][0]

#We print the result
print("The difference between the mean with lib and without is {0}"
      .format(mean_withlib-mean))
print("The difference between the median with lib and without is {0}"
      .format(median_withlib-median))
print("The difference between the mode with lib and without is {0}".
      format(mode_withlib-mode))
```

```
The difference between the mean with lib and without is 0.0
The difference between the median with lib and without is 0.0
The difference between the mode with lib and without is 0
```

The results obtained with the numpy functions are exactly the same as the results of the manually calculated functions. It is normal that the median and the mean are different because the mean is impacted by large and small values while the median only takes into account the number of elements.

In [38]:
```
#The we compute the range, the variance and the standard deviation
#of the same discrete series, firsyt by ourself and then with
#the numpy functions

#We compute the range
range = np.max(A)-np.min(A)
range_withlib =np.ptp(A,axis=0)

#We compute the Variance
num=0
for x in A:
    num += (x-mean)*(x+mean)
variance = num/len(A)
variance_withlib = np.var(A)

#We compute de standard deviation

std_withlib = np.std(A)
std = sqrt(variance)

#We compare the result
print("The difference between the range with lib and without is {0}"
      .format(range_withlib-range))
print("The difference between the variance with lib and without is {0}"
      .format(variance_withlib-variance))
```

```
print("The difference between the standard deviation with lib and without is {0}"
      .format(std_withlib-std))
```

```
The difference between the range with lib and without is 0
The difference between the variance with lib and without is -1.163513729807164e-13
The difference between the standard deviation with lib and without is -2.042810365310288e-14
```

It can be seen that the difference between the two methods creates very small deviations for the variance and the standard variation. The standard deviation is less than 10 et close to the means which means that de series is homogeneous.

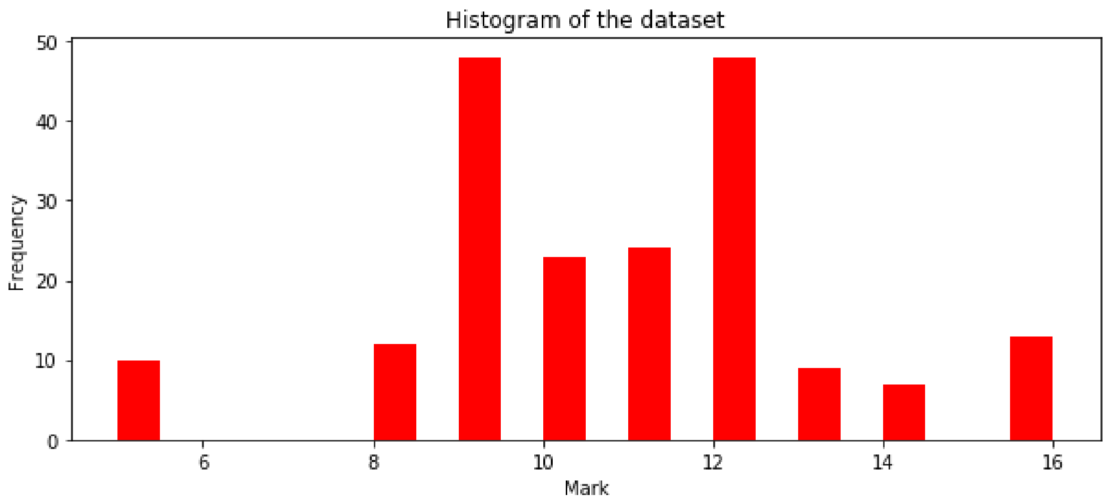
3 Exercice B : Grouped discrete series

```
In [39]: #We input the following array

         Mark = [5,8,9,10,11,12,13,14,16]
         Number = [10,12,48,23,24,48,9,7,13]

         #Then we plot these arrays, as an histogram
         figure,axe = plt.subplots(figsize=(10,4))
         plt.hist(Mark,bins=22,color='red',weights=Number)
         axe.set_title("Histogram of the dataset")
         axe.set_xlabel("Mark")
         axe.set_ylabel("Frequency")


         totalmark=[a*b for a,b in zip(Mark,Number)]
```



In [40]:
```
#We compute the mean, the median, the variance,
#the standard deviation, the min and the max

mean = np.sum(totalmark)/np.sum(Number)


dispersion_mark_median = stats.binned_statistic(Number,Mark,statistic="median",bins=1)
median = dispersion_mark_median[0][0]

numvar = []
i=0
while i<len(Mark):
    numvar.append(Number[i]*(Mark[i]-mean)**2)
    i+=1

variance = sum(numvar)/sum(Number)

dispersion_mark_min = stats.binned_statistic(Number,Mark,statistic="min",bins=1)
min = dispersion_mark_min[0][0]


dispersion_mark_max = stats.binned_statistic(Number,Mark,statistic="max",bins=1)
max = dispersion_mark_max[0][0]


mode = Mark[np.argmax(Number)]


sumarray = np.dot(Mark,Number)



#We display everything

print("Min: {0} ".format(min))
print("Mean: {0} ".format(mean))
print("Variance: {0} ".format(variance))
print("Standard deviation: {0}  ".format(std))
print("Median: {0} ".format(median))
print("Max: {0} ".format(max))
print("Mode: {0} ".format(mode))
```

```
Min: 5.0
Mean: 10.675257731958762
Variance: 5.848150706770113
Standard deviation: 2.8055257974219585
Median: 11.0
Max: 16.0
```

```
Mode: 9
```

```
D:\Programme\Anaconda\envs\ialab\lib\site-packages\scipy\stats\_binned_statistic.py:607: FutureW
  result = result[core]
```

We notice that this distribution is bimodal because it has 2 peaks at 9 and 12. This may be related to the fact that there are two different types of populations in this data set. Those who passed the exam and those who did not.

4 Exercice C : Normal distributions

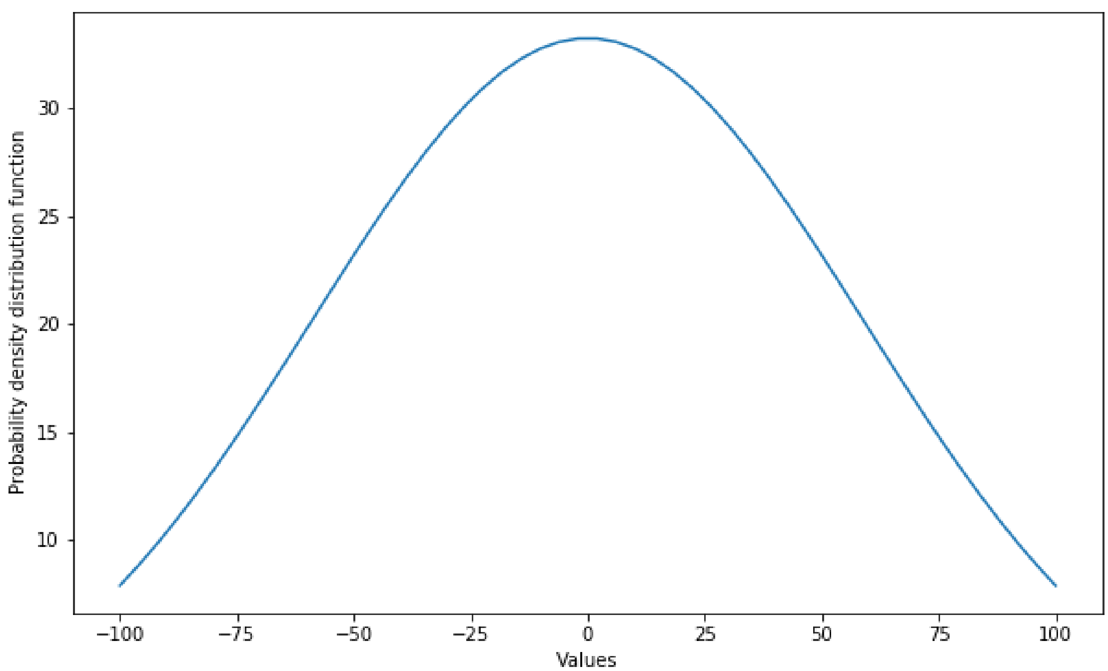
In [41]:
```python
#We generate a sample of n random variables that follow a normal
#distribution of mean m and standard deviation sd

x = np.linspace(-100,100)
std = x.std()
m= x.mean()

probaility_density_fct =np.exp((-0.5*(x-m)**2)/std**2)*(1/(np.sqrt(np.pi))*std)

#We plot it
fig,axe2 = plt.subplots(figsize=(10,6))
axe2.plot(x,probaility_density_fct)
axe2.set_xlabel('Values')
axe2.set_ylabel('Probability density distribution function')
```

Out[41]:
```
Text(0,0.5,'Probability density distribution function')
```

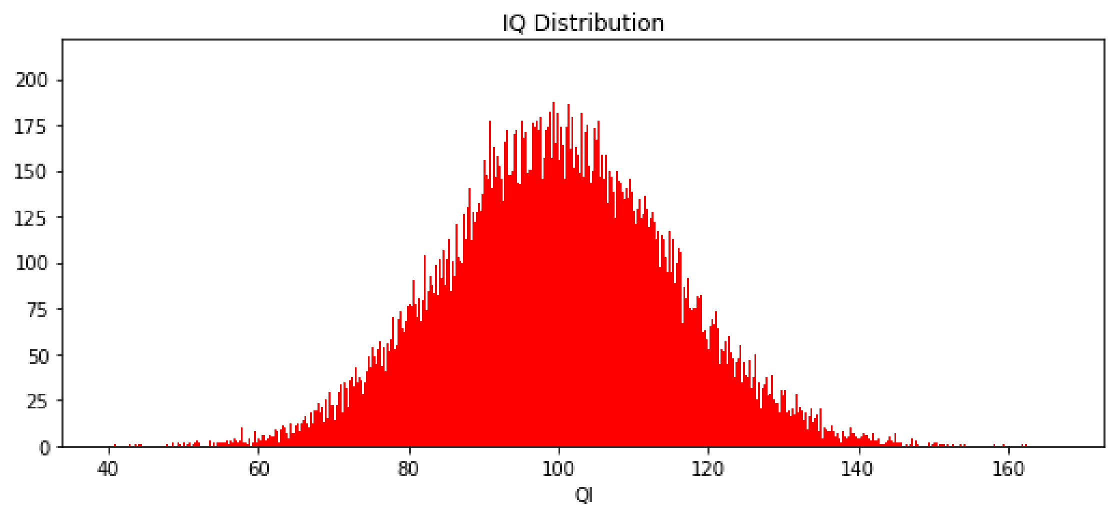


In [42]:
```
#We gnerate a sample of size 100000 with np.random.normal with
#parameters: mu =100, sigma = 255. Then we display its histogram

mu = 100
sigma = sqrt(225)
QI = np.random.normal(mu, sigma, 100000)

figure,axe = plt.subplots(figsize=(10,4))
plt.hist(QI,bins=2000,color='red')
axe.set_title("IQ Distribution")
axe.set_xlabel("QI")
```

Out[42]:
```
Text(0.5,0,'QI')
```



In [43]:
```
#Compute the mean and the standard deviation of our distribution
mean_norm = np.mean(QI)
std_norm = np.std(QI)

print("Mean: ",mean_norm)
print("Standard deviation: ",std_norm)
print("Variance : ", std_norm**2)
```

```
Mean:  99.97417399964813
Standard deviation:  14.966496430905504
Variance :  223.9960154163072
```

The standard deviation is very far from the mean, which means that there are many disparities among the different IQs present in the data set. The average is 100, slightly higher than the French average (98) but still low compared to Hong Kong: 108.

In [44]:
```
#We create a function who return the probability for a QI to be above or
#below a certain value

def proba_isSup(val,isSup):
    count = 0
    for element in QI:
        if element < val and isSup == True:
            count+=1
        if element > val and isSup == False:
            count+=1
    return (count/len(QI))*100

range_95 = stats.norm.interval(0.95,loc=mean_norm, scale=std_norm)

print("Percentage of the sample that has an IQ bellow 60", proba_isSup(60,True))
print("Percentage of the sample that has an IQ above 130",proba_isSup(130,False))
print("Range of values that contains 95 percent of the sample",range_95)
```

```
Percentage of the sample that has an IQ bellow 60 0.387
Percentage of the sample that has an IQ above 130 2.3369999999999997
Range of values that contains 95 percent of the sample (70.64038002032608, 129.30796797897017)
```

5 Exercice D : IQ analysis

In [45]:
```
#We generate 3 different samples of size 10, 1000 and 100000
#(set10, set100, set100000) with a mean value
#of 100 and a standard deviation of 15

std = 15
mean = 100

set10= np.random.normal(mean, std, 10)
set100= np.random.normal(mean, std, 100)
set100000= np.random.normal(mean, std, 100000)

mean_value10 = np.mean(set10)
mean_value100 = np.mean(set100)
mean_value100000 = np.mean(set100000)

std_value10 = np.std(set10)
std_value100 = np.std(set100)
std_value100000 = np.std(set100000)
```

```python
print("Comparaison mean 10 :", mean_value10-mean)
print("Comparaison mean 100 :", mean_value100-mean)
print("Comparaison mean 100000 :", mean_value100000-mean,'\n')

print("Comparaison std 10 :", std_value10-std)
print("Comparaison std 100 :", std_value100-std)
print("Comparaison std 100000 :", std_value100000-std,'\n')

range_10 = stats.norm.interval(0.95,loc=mean_value10, scale=std_value10)
range_100 = stats.norm.interval(0.95,loc=mean_value100, scale=std_value100)
range_100000 = stats.norm.interval(0.95,loc=mean_value100000, scale=std_value100000)

typerror10 = std/sqrt(10)
typerror100 = std/sqrt(100)
typerror100000 = std/sqrt(100000)

print("Interval 10 :", range_10)
print("Interval 100 :", range_100)
print("Interval 100000 :", range_100000,'\n')

print("typerror10 : " , typerror10)
print("typerror100 : " , typerror100)
print("typerror100000 : " , typerror100000)


a= mean_value100 + 1.96*(std_value100/100)
print ("val = ", a)
```

```
Comparaison mean 10 : -0.8596166502354379
Comparaison mean 100 : 0.0976825460552817
Comparaison mean 100000 : -0.09479871876243351

Comparaison std 10 : 1.2904007951243806
Comparaison std 100 : 0.2940737884636775
Comparaison std 100000 : -0.01794482301835565

Interval 10 : (67.21178449759812, 131.06898220193102)
Interval 100 : (70.12184874376841, 130.07351634834214)
Interval 100000 : (70.54091271996168, 129.26948984251345)

typerror10 :  4.743416490252569
typerror100 :  1.5
typerror100000 :  0.04743416490252569
val =  100.39744639230916
```

The more data there are, the more the confidence interval increases, the closer the mean is to the theoretical mean and the closer the standard deviation is to the theoretical mean. On the other hand, with a small dataset, the errors are huge (see typerror) and the confidence interval is reduced.

```
In [27]: #We import the csv file with pandas
         import pandas as pd

         dataframe = pd.read_csv("malnutrition.csv")

         mean_malnutrition = np.mean(dataframe.values)
         std_malnutrition = np.std(dataframe.values)

         malnutrition_interval = stats.norm.interval(0.95,loc=mean_malnutrition,
                                                      scale=std_malnutrition)
```

```
In [49]: #Comparaison with set1000
         diff_mean = mean_malnutrition-mean_value100
         diff_std = std_malnutrition-std_value100
         diff_intervalMin = diff_mean - 1.96*sqrt((std_malnutrition**2/len(dataframe))
                                                  +((std_value100**2/len(set100))))
         diff_intervalMax = diff_mean + 1.96*sqrt((std_malnutrition**2/len(dataframe))
                                                  +((std_value100**2/len(set100))))


         print("Mean difference : " ,diff_mean)
         print("Standard deviation difference : " ,diff_std)
         print("Confidence interval difference: " ,diff_intervalMin, ";"
               ,diff_intervalMax)
```

```
Mean difference :  -12.208793657166396
Standard deviation difference :  -5.659446750566907
Confidence interval difference:  -15.75673132394913 ; -8.660855990383663
```

Conclusion

According to the results, people suffering from malnutrition have an IQ that is 12.5 points lower on average, which is a lot. The standard deviation is higher, which means that malnutrition has a more or less significant impact on individuals. We also note with the confidence interval that it is those with high IQ that decline more than those with low IQ already. It would be interesting to know to what extent each individual is affected by malnutrition with indicators such as their weight or their cholesterol level for example. However, it would be interesting to ask whether malnutrition leads to a decrease in IQ or whether a lower IQ increases the chances of malnutrition.