Lab #1

[II.2313] Data Analysis

Due date: Septbember 19th, 2019

MERCIER Lucas 10067

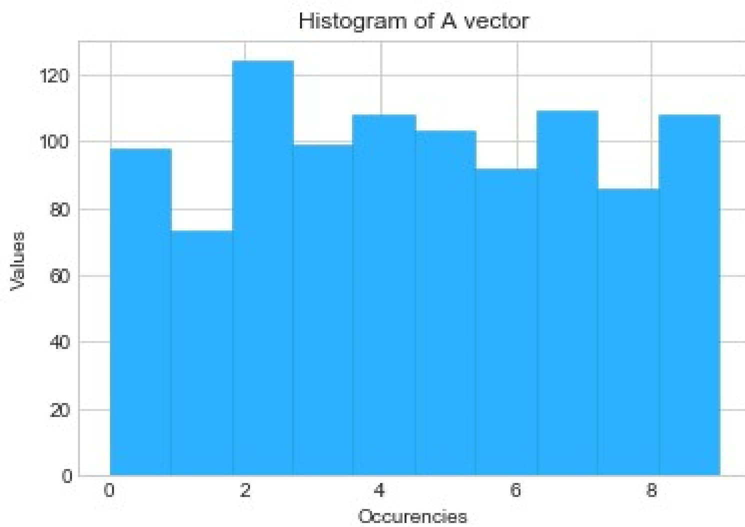
In [196]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statistics
import scipy.stats
from math import sqrt,pi
```

In [197]:
```
def clean_hist(values,title,xLabel='Values',yLabel="Occurrencies",step
s=1,color_mean=False,):
    plt.title(title)
    plt.xlabel(xLabel)
    plt.ylabel(yLabel)
    n,bins,patches=plt.hist(values,bins=np.arange(round(min(values)+
1),round(max(values)),steps), facecolor = '#2ab0ff', edgecolor='#169ac
f', linewidth=0.5)
    if color_mean:
        patches[len(bins)//2-1].set_fc('red') # Set color
    plt.show()
```

A - Discrete series

In [249]:

```
A=np.random.randint(0,10,1000)
plt.title("Histogram of A vector")
plt.xlabel("Occurencies")
plt.ylabel("Values")
plt.hist(A,bins=10, facecolor = '#2ab0ff', edgecolor='#169acf', linewi
dth=0.5)
plt.show()
```



In [155]:

```
def mean(x):
    return sum(x)/len(x)

def median(x):
    sort=np.sort(x)
    N=len(x)
    if(N%2!=0):
        return x[(N)//2]
    else:
        return (sort[(N-1)//2]+sort[(N+1)//2])/2

def mode(x):
    dic=dict()
    for i in x:
        if(i not in dic):
            dic[i]=np.count_nonzero(x==i)
    return max(dic, key=dic.get)
```

In [200]:

```
print("Central tendencie measures (without numpy functions):")
print("    Computed mean is {:.2f}".format(mean(A)))
print("    Computed median is {:.2f}".format(median(A)))
print("    Computed Mode is {:.2f}".format(mode(A)))
print("")
print("Central tendencie measures (with numpy and statistics function
s):")
print("    Real mean (with numpy) is {:.2f}".format(np.mean(A)))
print("    Real median (with numpy) is {:.2f}".format(np.median(A)))
print("    Real median (with statistics) is {:.2f}".format(statistics.
mode(A)))
```

```
Central tendencie measures (without numpy functions):
    Computed mean is 4.61
    Computed median is 5.00
    Computed Mode is 6.00

Central tendencie measures (with numpy and statistics functions):
    Real mean (with numpy) is 4.61
    Real median (with numpy) is 5.00
    Real median (with statistics) is 6.00
```

4. We can see that results are very close to the theorical values, we can conclude that our functions are accurate enough.

5. Median and Mean value can differs a lot if values are spreaded or not.

If there is a value too far from the reste of the data, it will impact a lot the mean value (based on all elements), whereas outer values doesn't impact the median.

In [203]:

```
def sample_range(x):
    return max(x)-min(x)

def variance(x):
    s=0
    mu=np.mean(x)
    for i in x:
        s+=(i-mu)**2
    return s/len(x)

def sd(x):
    return sqrt(variance(x))
```

In [204]:
```
print("Dispersion criteria :")
print("Computed range (without numpy) is {:.5f}".format(sample_range
(A)))
print("Real range (with numpy) is {:.5f}".format(np.ptp(A)))
print("Computed var (without numpy) is {:.5f}".format(variance(A)))
print("Real var (with numpy) is {:.5f}".format(np.var(A)))
print("Computed standard deviation (without numpy) is {:.5f}".format(s
d(A)))
print("Real standard deviation (with numpy) is {:.5f}".format(sqrt(np.
var(A))))
```

```
Dispersion criteria :
Computed range (without numpy) is 9.00000
Real range (with numpy) is 9.00000
Computed var (without numpy) is 8.22300
Real var (with numpy) is 8.22300
Computed standard deviation (without numpy) is 2.86758
Real standard deviation (with numpy) is 2.86758
```

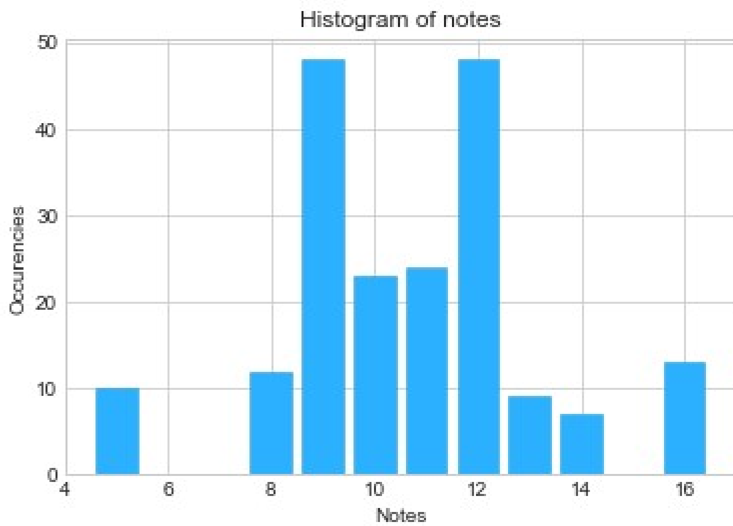
6. Once again values we computed with our own functions are very close to the expected ones (obtained by numpy built-in functions).

If we take a look at those results we can see that values are more or less equally spreaded.

B - Discrete series with frequencies

In [166]:

```
note=np.array([5,8,9,10,11,12,13,14,16])
numbers=np.array([10,12,48,23,24,48,9,7,13])
notes=np.array([note, numbers])
plt.title("Histogram of notes")
plt.xlabel("Notes")
plt.ylabel("Occurencies")
plt.bar(note,numbers, facecolor = '#2ab0ff', edgecolor='#169acf', line
width=0.5)
plt.show()
```



In [167]:

```python
def ponderated_var(val,freq,mean):
    mean=ponderated_mean(val,freq)
    s=0
    return ponderated_mean(val**2,freq)-ponderated_mean(val,freq)**2


def ponderated_mean(val,freq):
    s=0
    for i in range(len(val)):
        s+=val[i]*freq[i]
    return s/sum(freq)

def ponderated_median(val,freq):
    ttl=[]
    for i in range(len(val)):
        for j in range(freq[i]):
            ttl.append(val[i])
    return median(ttl)

def ponderated_mode(val,freq):
    dic=dict()
    for i in freq:
        if(i not in dic):
            dic[i]=np.count_nonzero(freq==i)

    newmax=max(dic, key=dic.get)
    maxs=[]
    for i in range(len(val)):
        if freq[i]==newmax: maxs.append(val[i])
    return maxs
```

In [168]:

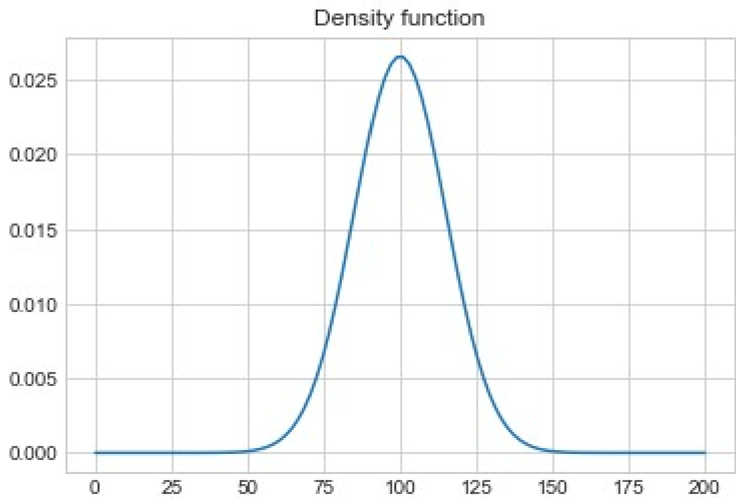
```python
print("Central measure tendencies")
print("Mean: {:.2f}".format(ponderated_mean(note,numbers)))
print("Median: {:.2f}".format(ponderated_median(note,numbers)))
print("Mode(s):")
print(ponderated_mode(note,numbers))
print("")
print("Dispersion criteria")
print("Range: {:.2f}".format(max(note)-min(note)))
var=ponderated_var(note,numbers,ponderated_mean(note,numbers))
print("Ponderated variance: {:.2f}".format(var))
print("Standard deviation: {:.2f}".format(sqrt(var)))
```

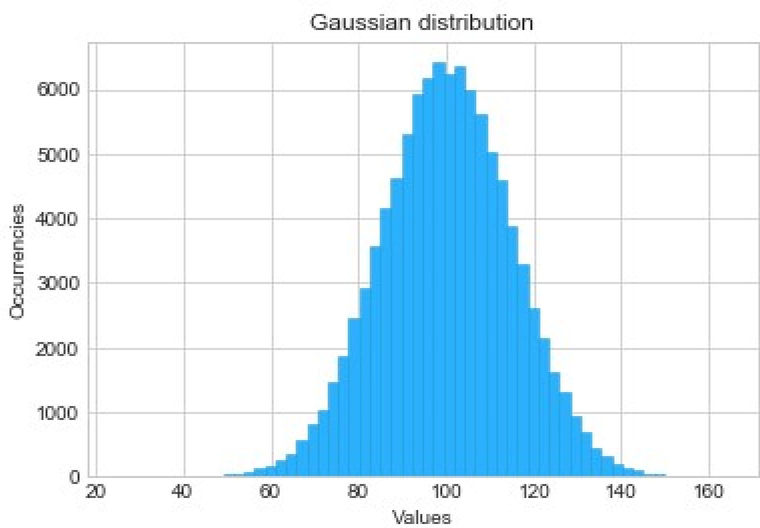
```
Central measure tendencies
Mean: 10.68
Median: 11.00
Mode(s):
[9, 12]

Dispersion criteria
Range: 11.00
Ponderated variance: 5.85
Standard deviation: 2.42
```

3. We can see that there is two modes (9 and 12) ,the distribution is then bimodal.

C - Gaussian distribution

In [176]:
```
m=100
sd=sqrt(225)
n=100000
norm=np.random.normal(m,sd,n)

plt.style.use('seaborn-whitegrid')
plt.title("Density function")
plt.plot(np.linspace(0, 200, 100), scipy.stats.norm.pdf(np.linspace(m-100, m+100, 100),m,sd))
plt.show()

clean_hist(norm,"Gaussian distribution",steps=2.4)
```





In [177]:
```
print("Computing mean and median to check if values are accurate")
print("Mean of the gaussian distribution {0:.2f} , should have found {1:.2f}".format(np.mean(norm),m))
print("Variance of the gaussian distribution {0:.2f} , should have found {1:.2f}".format(np.var(norm),sd**2))
```

```
Computing mean and median to check if values are accurate
Mean of the gaussian distribution 100.03 , should have found 100.00
Variance of the gaussian distribution 223.69 , should have found 225.
00
```

3. Values are close to desired one. The more samples we take, the closest we get to real values, which are μ=100 and σ=15

In [12]:
```
print("4. In our sample, {:.2f}% of people have an IQ bellow 60.".format(scipy.stats.norm.cdf(60,m,sd)*100))
print("5. In our sample, {:.2f}% of people have an IQ above 130.".format((1-scipy.stats.norm.cdf(130,m,sd))*100))
```

```
4. In our sample, 0.38% of people have an IQ bellow 60.
5. In our sample, 2.28% of people have an IQ above 130.
```

In [13]:
```
print("6. 95% of the data are in the interval [{0:.2f};{1:.2f}].".format(m-1.96*sd,m+1.96*sd))
```

```
6. 95% of the data are in the interval [70.60;129.40].
```

D - IQ analysis

In [209]:
```
m=100
sd=15
n10=np.random.normal(m,sd,10)
n1000=np.random.normal(m,sd,1000)
n100000=np.random.normal(m,sd,100000)
```

In [210]:
```
print("1st sample (n=10):")
print("Mean value: {0:.2f}".format(np.mean(n10)))
print("SD value: {0:.2f}".format(sqrt(np.var(n10))))
print("")
print("2nd sample (n=1000):")
print("Mean value: {0:.2f}".format(np.mean(n1000)))
print("SD value: {0:.2f}".format(sqrt(np.var(n1000))))
print("")
print("3rd sample (n=100000):")
print("Mean value: {0:.2f}".format(np.mean(n100000)))
print("SD value: {0:.2f}".format(sqrt(np.var(n100000))))
```

```
1st sample (n=10):
Mean value: 99.98
SD value: 10.85

2nd sample (n=1000):
Mean value: 98.94
SD value: 15.03

3rd sample (n=100000):
Mean value: 99.94
SD value: 15.09
```

1. The highest number of samples we take, the closest we are from the mean and variance that defines our law.

In [211]:
```
def estimated_sd(x,real_mean):
    N=len(x)-1
    s=0
    for i in x:
        s+=(i-real_mean)**2
    return s/N

def standard_error(x,real_mean):
    return estimated_sd(x,real_mean)/sqrt(len(x))

def C95(x,real_mean):
    return [sum(x)/len(x)-standard_error(x,real_mean)*1.96,sum(x)/len
(x)+standard_error(x,real_mean)*1.96]
```

In [212]:
```
print("1st sample (n=10):")
print("Standard error of the mean value of the 1st sample: {:.2f}".format(standard_error(n10,m)))
print("Confidence interval (95%) of the estimated mean:[{0:.2f},{1:.2f}]".format(C95(n10,m)[0],C95(n10,m)[1]))
print("")
print("1st sample (n=1000):")
print("Standard error of the mean value of the 1st sample: {:.2f}".format(standard_error(n1000,m)))
print("Confidence interval (95%) of the estimated mean:[{0:.2f},{1:.2f}]".format(C95(n1000,m)[0],C95(n1000,m)[1]))
print("")
print("1st sample (n=100000):")
print("Standard error of the mean value of the 1st sample: {:.2f}".format(standard_error(n100000,m)))
print("Confidence interval (95%) of the estimated mean:[{0:.2f},{1:.2f}]".format(C95(n100000,m)[0],C95(n100000,m)[1]))
```

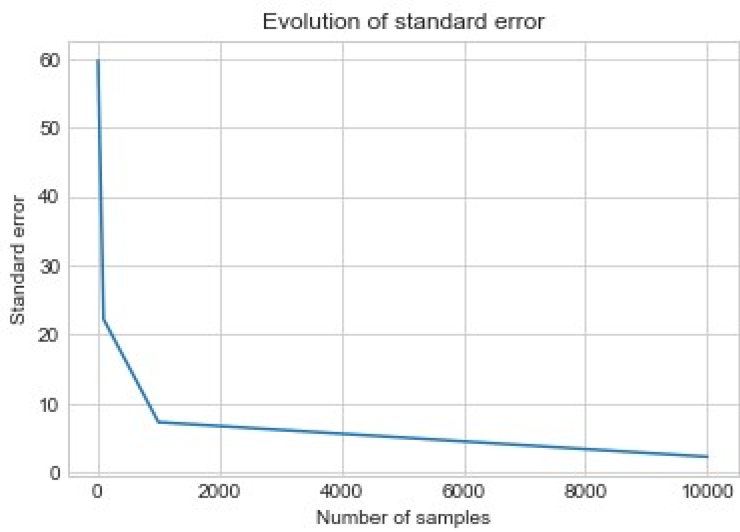
```
1st sample (n=10):
Standard error of the mean value of the 1st sample: 41.36
Confidence interval (95%) of the estimated mean:[18.91,181.05]

1st sample (n=1000):
Standard error of the mean value of the 1st sample: 7.19
Confidence interval (95%) of the estimated mean:[84.85,113.02]

1st sample (n=100000):
Standard error of the mean value of the 1st sample: 0.72
Confidence interval (95%) of the estimated mean:[98.53,101.36]
```

As we could have expected for very small samples (>100) results are inaccurate and confidence interval is way too broad.

This range tends to reduce as more samples we take. To highligh this phenomenom, we can plot the evolution of the standard error based on the number of samples (fig. bellow)

In [220]:
```
vals=[]
x=[]
for i in range(5):
    x.append(10**i)
    n=np.random.normal(m,sd,10**i)
    vals.append(standard_error(n,m))
plt.title("Evolution of standard error")
plt.xlabel("Number of samples")
plt.ylabel("Standard error")
plt.plot(x,vals)
plt.show()
```

```
C:\Users\Lucas\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: R
untimeWarning: divide by zero encountered in double_scalars
```



We can see that standard error start decaying really quickly when we increase number of samples. Based on previous results with confidence interval we can conclude that if we want to make conclusion based on this data we have to take a lot of samples such as n=100000.

In [48]:
```
malnutrition=pd.read_csv('malnutrition.csv',header=None)
values=malnutrition.values
```

In [19]:
```
print("3. Mean and variance of the data:")
print("   Mean of the data: {:.2f}".format(np.mean(values)))
print("   Variance of the data: {:.2f}".format(sqrt(np.var(values))))
```

```
3. Mean and variance of the data:
   Mean of the data: 87.98
   Variance of the data: 9.63
```

In [53]:
```
def compare(x,y):
    return abs(x-y)

def DCI95(A,B,real_mean,choice):
    if choice == "mean": D=np.mean(A)-np.mean(B)
    elif choice == "sd": D=np.std(A)-np.std(B)
    else: return "Wrong parameter, please select either 'mean' or 'sd'"
    #Sa=estimated_sd(A,real_mean)
    Sa=np.std(A)
    #Sb=estimated_sd(B,real_mean)
    Sb=np.std(B)
    Na=len(A)
    Nb=len(B)
    sqr=sqrt((Sa**2/Na)+(Sb**2/Nb))
    return [D-1.96*sqr,D+1.96*sqr]
```

In [222]:
```
print("Let's compare mean and sd of data from csv and sample with
(n)")
print("The difference for the mean is {0:.2f}".format(compare(np.mean
(values),np.mean(n100000))))
print("The difference for the sd is {0:.2f}".format(compare(np.std(val
ues),np.std(n100000))))
print("")
print("To conclude let's compute confidence interval for those compari
son of those two values")
print("Confidence interval (95%) for the mean [{0:.2f};{1:.2f}]".forma
t(DCI95(n100000,values,m,"mean")[0],DCI95(n100000,values,m,"mea
n")[1]))
print("Confidence interval (95%) for the sd [{0:.2f};{1:.2f}]".format
(DCI95(n100000,values,m,"sd")[0],DCI95(n100000,values,m,"sd")[1]))
```

```
Let's compare mean and sd of data from csv and sample with (n)
The difference for the mean is 11.96
The difference for the sd is 5.46

To conclude let's compute confidence interval for those comparison of
those two values
Confidence interval (95%) for the mean [10.07;13.85]
Confidence interval (95%) for the sd [3.58;7.35]
```

4. Based on this results we can say that there is a mean difference of 10 to 14 IQ point between peoples who suffers from malnutrtion and those who don't.

However, we have to keep multiple factors in mind. The 'malnutrition' dataset only contains 100 samples wich is few compared to the test sample (n=100000). Morever we don't know precisly every factors in our case. Malnutrtion might be caused by another phenomenom way more important or might lead to consequences that we don't know here. Our data only has one attribute so we have to be conscient about those facts before concluding.

Finaly we can conclude that there is a **correlation** between malnutrition and IQ but malnutrition **is not necessarily the cause** of IQ loses.