TP1 SERIES TEMPS

Réaliser par MEKA MOISE CHRISTIAN JUNIOR 21T2561

Il s'agit de l'implémentation du TP1 sur le dataset des données des ventes d'une entreprse

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

In [2]:
```
FILEPATH="Month_Value_1.csv"
extension=FILEPATH.split(".")[-1]
SEP=","
if extension=="csv":
    df=pd.read_csv(FILEPATH, sep=SEP)
else:
    df=pd.read_excel(FILEPATH, index_col=0)
```

In [3]:
```
extension
```

Out[3]:
```
'csv'
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96 entries, 0 to 95
Data columns (total 5 columns):
 #   Column                                    Non-Null Count  Dtype  
---  ------                                    --------------  -----  
 0   Period                                    96 non-null     object 
 1   Revenue                                   64 non-null     float64
 2   Sales_quantity                            64 non-null     float64
 3   Average_cost                              64 non-null     float64
 4   The_average_annual_payroll_of_the_region  64 non-null     float64
dtypes: float64(4), object(1)
memory usage: 3.9+ KB
```

In [5]:
```
df.head()
```

Out[5]:

	Period	Revenue	Sales_quantity	Average_cost	The_average_annual_payroll_of
0	01.01.2015	1.601007e+07	12729.0	1257.763541	
1	01.02.2015	1.580759e+07	11636.0	1358.507000	
2	01.03.2015	2.204715e+07	15922.0	1384.697024	
3	01.04.2015	1.881458e+07	15227.0	1235.606705	
4	01.05.2015	1.402148e+07	8620.0	1626.621765	

In [6]:
```
metrique="Average_cost"
```

```
period="Period"
```

In [7]:

```
df[metrique].describe()
```

Out[7]:

```
count      64.000000
mean     1695.061159
std       296.844793
min      1110.576805
25%      1499.142841
50%      1654.399798
75%      1916.401096
max      2559.328184
Name: Average_cost, dtype: float64
```

In [8]:

```
df=df.dropna()
```

Question 1

In [9]:

```
#moyenne
np.mean(df[metrique])
```

Out[9]:

```
1695.0611591371598
```

In [10]:

```
#varaince
np.std(df[metrique])**2
```

Out[10]:

```
86740.00550102274
```

In [11]:

```
#ecart-type
np.std(df[metrique])
```

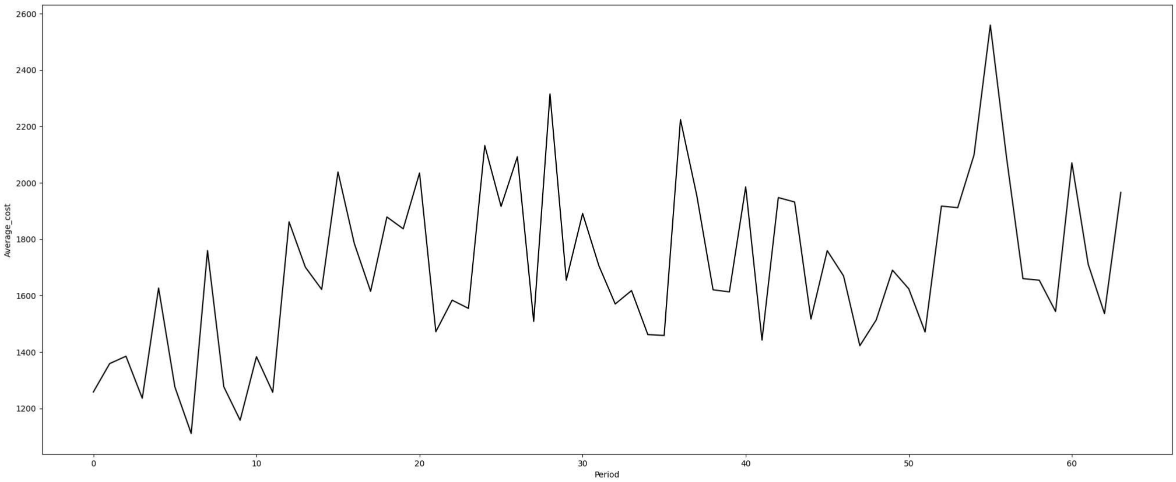
Out[11]:

```
294.5165623543483
```

Question 2

In [12]:

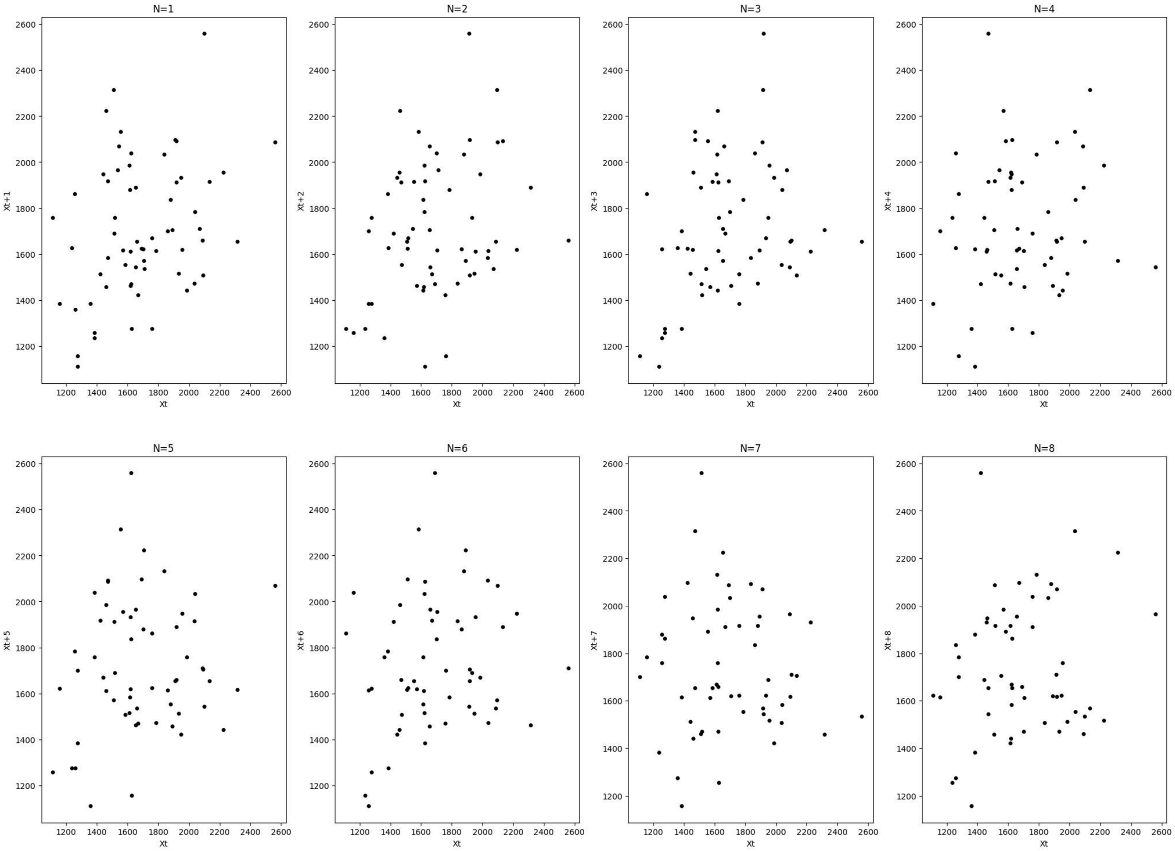
```
plt.figure(figsize=[25,10])
plt.plot(df[metrique], color='black',)
plt.xlabel(period)
plt.ylabel(metrique)
plt.show()
```



Question 3

In [13]:

```
###NX
plt.figure(figsize=[25,18])
for N in range(1,9):
    plt.subplot(2,4,N)
    debut=0
    fin=df.shape[0]
    plt.scatter(df[metrique][debut:fin-N], df[metrique][debut+N:fin], color='bla
    plt.xlabel("Xt")
    plt.ylabel(f"Xt+{N}")
    plt.title(f"N={N}")
plt.show()
```



Question 4

In [14]:

```
#Calcul l'auto-Covariance empirique
def auto_cov(data, K, moy):
    debut=0
    fin=len(data)
    Xt=data[debut:fin-K]
    Xt_k=data[debut+K:fin]

    cov=0
    for i in range(fin-K):
        cov+=(Xt[i]-moy)*(Xt_k[i]-moy)
    return cov/(fin-K)
```

In [15]:

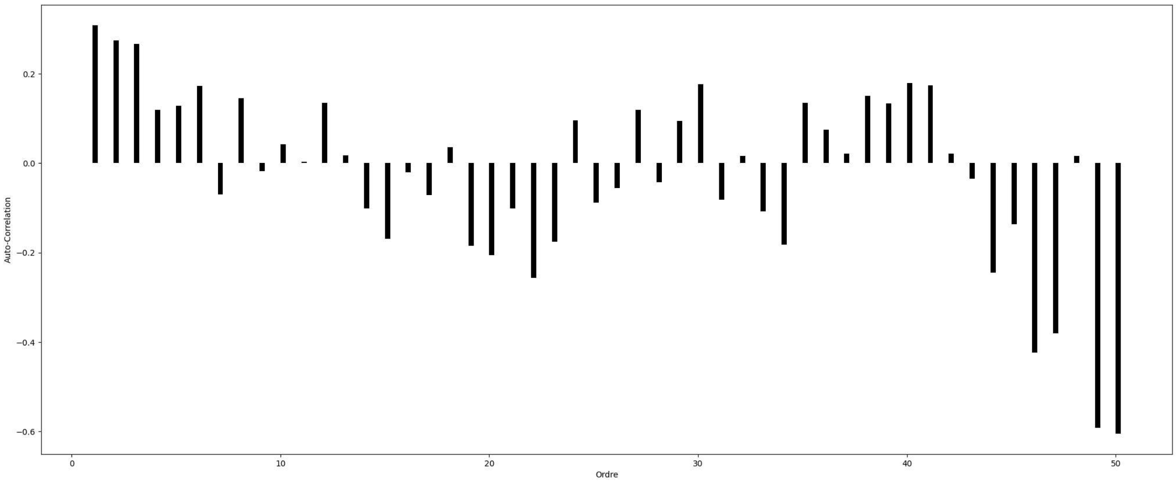
```
def auto_cor(data, K):
    moy=np.mean(data)
    cov_0=auto_cov(data, 0, moy)
    cov_K=auto_cov(data, K, moy)
    return cov_K/cov_0
```

In [16]:
```
from tqdm import tqdm
auto_cor_all=list()
data=list(df[metrique])
for i in tqdm(range(1, 51)):
    auto_cor_all.append(auto_cor(data, i))
```

```
100%|██████████████████████████████████████████████████████████████████████████████
███| 50/50 [00:00<00:00, 16711.71it/s]
```

In [17]:
```
indexes=[i for i in range(1,51)]
plt.figure(figsize=[25,10])
plt.bar(indexes,auto_cor_all, align='edge', width= 0.25, color='black')
plt.xlabel("Ordre")
plt.ylabel("Auto-Correlation")
```

Out[17]:
```
Text(0, 0.5, 'Auto-Correlation')
```


Auto-Correlation
Ordre