TP1 SERIES TEMPS

Réaliser par MEKA MOISE CHRISTIAN JUNIOR 21T2561

Il s'agit de l'implémentation du TP1 sur le dataset des données météorologique

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

In [2]:
```
FILEPATH="annual_rainfall_dallas.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: 121 entries, 0 to 120
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Year    121 non-null    int64  
 1   Total   121 non-null    float64
dtypes: float64(1), int64(1)
memory usage: 2.0 KB
```

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

Out[5]:

	Year	Total
0	2019	34.52
1	2018	55.97
2	2017	36.62
3	2016	35.48
4	2015	62.61

In [6]:
```
metrique="Total"
period="Year"
```

In [7]:
```
df[metrique].describe()
```

Out[7]:
```
count    121.000000
mean      33.327769
std        9.186430
min       17.910000
25%       26.440000
50%       33.140000
75%       39.290000
max       62.610000
Name: Total, dtype: float64
```

In [8]:
```
df=df.dropna()
```

Question 1

In [9]:
```
#moyenne
np.mean(df[metrique])
```

Out[9]:
```
33.32776859504133
```

In [10]:
```
#varaince
np.std(df[metrique])**2
```

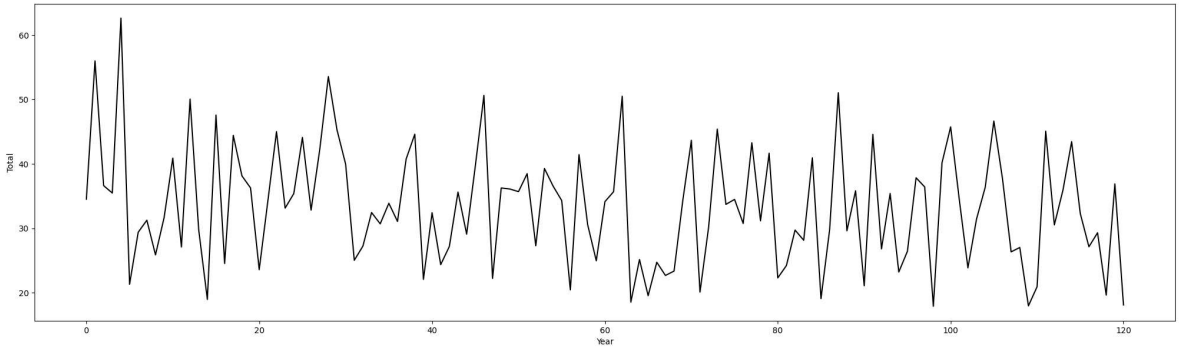
Out[10]:
```
83.69304873984018
```

In [11]:
```
#ecart-type
np.std(df[metrique])
```

Out[11]:
```
9.14839049996447
```

Question 2

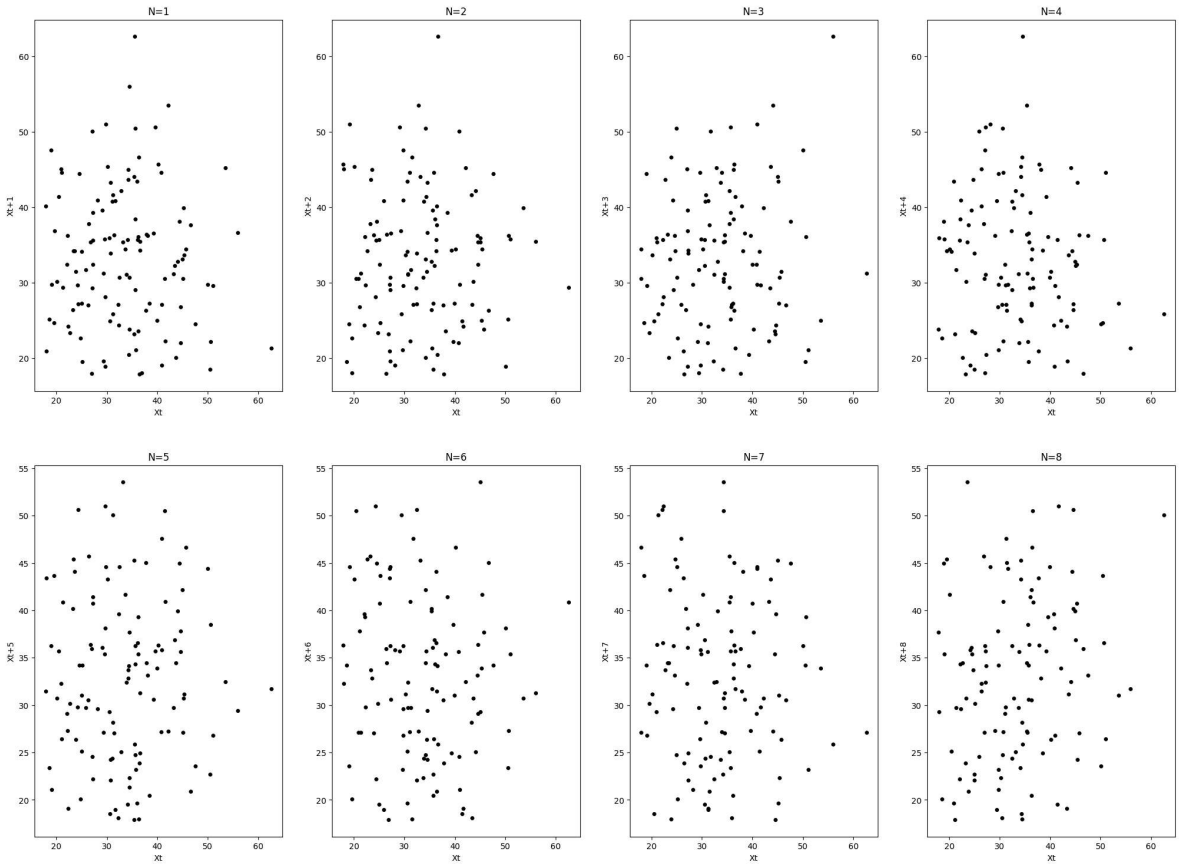
In [12]:
```
plt.figure(figsize=[25,7])
plt.plot(df[metrique], color='black',)
plt.xlabel(period)
plt.ylabel(metrique)
plt.show()
```



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, 5570.72it/s]
```

In [17]:
```
indexes=[i for i in range(1,51)]
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')