L’analyse des séries chronologiques avec Python

Présenté par :

EL MOUTAQUI Hicham

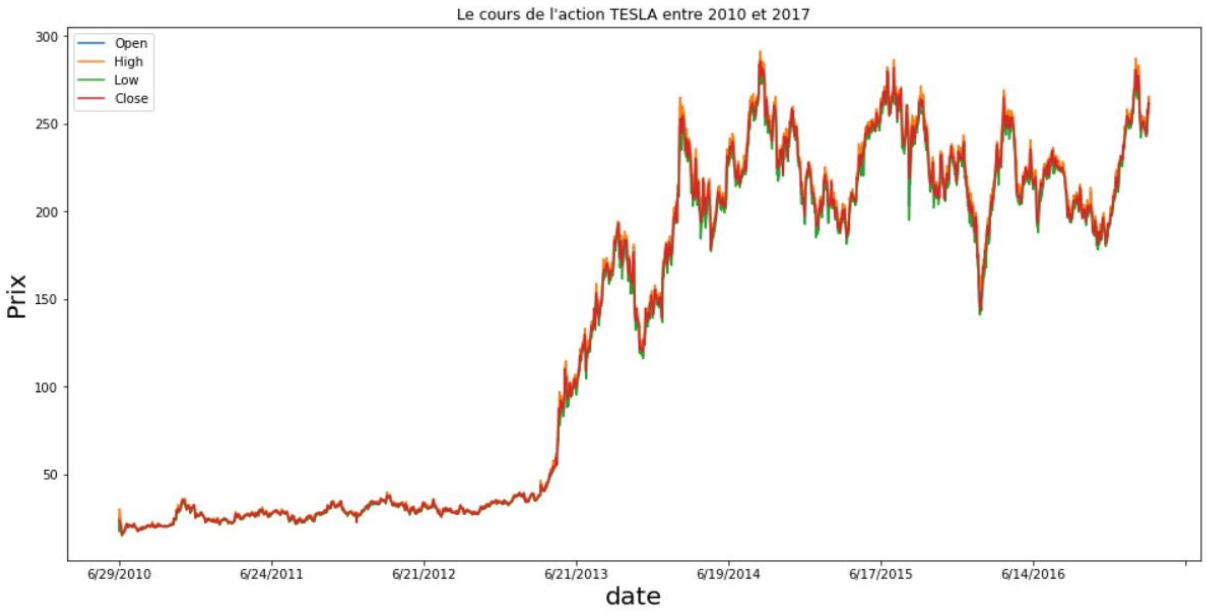
Octobre 03, 2021

Table des matières :

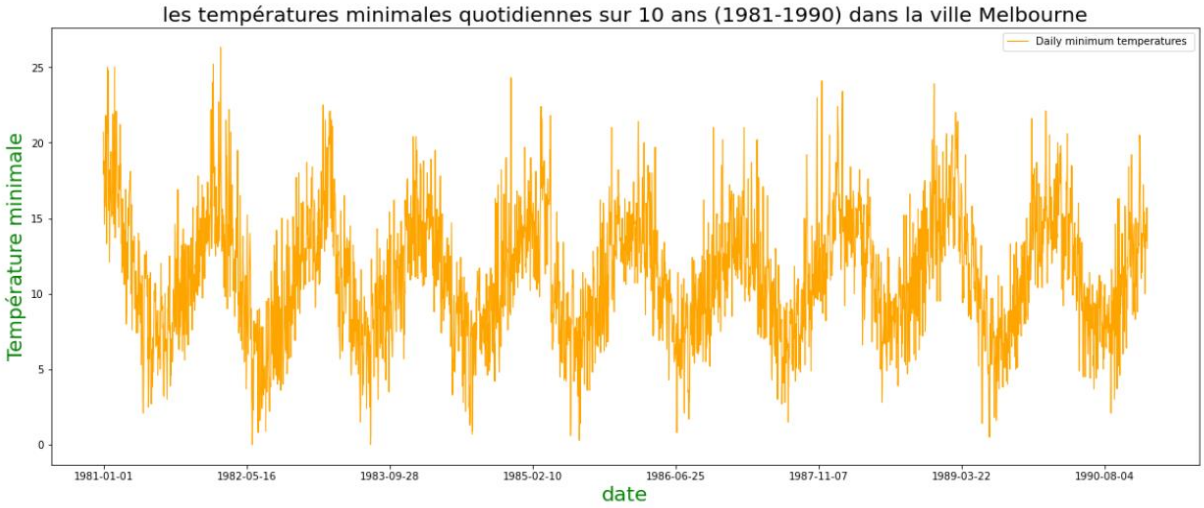
Série chronologique : Une série chronologique est une séquence d'observations enregistrées à intervalles de temps réguliers. Une série chronologique peut généralement être quotidienne, hebdomadaire, mensuelle, trimestrielle ou annuelle…

Exemples :

1. **Le cours de l'action TESLA**



2. La température minimale journalière dans la ville Melbourne à l'Australie.



La source des données est le bureau australien de météorologie.

Les éléments composant une série chronologique :

⇨ On peut décomposer une série chronologique en 3 éléments **:**

+ La tendance (Trend) : décrit le comportement de la série à long terme, elle traduit le comportement moyen de la série.

+ La composante saisonnière (Saisonnalité s(t)) : décrit la répétition d'un phénomène à intervalle de temps régulier, de manière générale, il s'agit d'un phénomène saisonnier. Il suffit de trouver la période p qui vérifie s(t) = s(t+p).

+ La composante résiduelle ou bruit : Ce sont des fluctuations irrégulières aléatoires, pour obtenir cette composante il suffit de retirer la tendance et la saisonnalité de la série chronologique.

1. Les bibliothèques nécessaires pour accomplire la première tâche.

Entrée [19]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import mpld3
mpld3.enable_notebook()
pd.set_option('display.max_rows', 7)
import statsmodels.api as sm
```

2. Génerer les 3 élements composant la série chronologique.

Entrée [20]:
```
compos_lineare = np.linspace(50,250, 120)
compos_saisonnière = np.cos(compos_lineare)*20
compos_residuelle = np.random.randn(120)
```

3. Elaborer la "DataFrame" à l'aide de la biblio Pandas.

Entrée [21]:
```
df = pd.DataFrame(compos_lineare+compos_saisonnière+compos_residuelle,
                  index= sm.tsa.datetools.dates_from_range('2021m1', length = 120),
                  columns = ['Valeures de la serie chrono'])
```

Entrée [22]:
```
df
```

Out[22]:

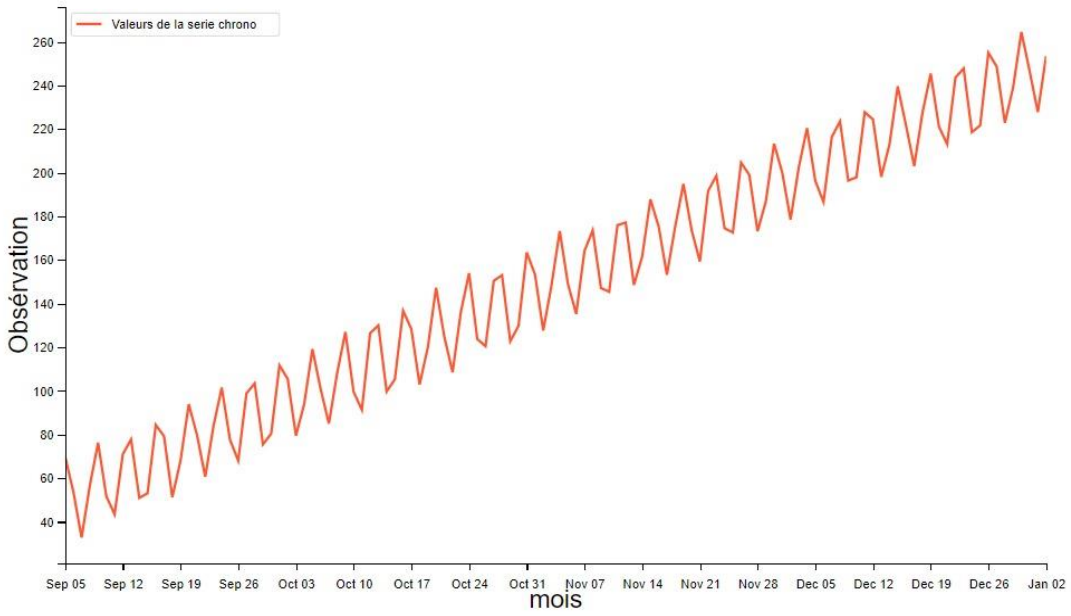
	Valeures de la serie chrono
2021-01-31	69.696666
2021-02-28	53.594893
2021-03-31	32.155608
...	...
2030-10-31	247.349774
2030-11-30	230.345006
2030-12-31	255.185182

120 rows × 1 columns

4. Illustrer la série chronologique à l'aide de matplotlib.

Entrée [5]:
```
Fig = df.plot(kind='line', lw = 2, linestyle = '-', color= '#FF5733', figsize = (14, 8))
plt.xlabel('mois', fontsize = 20)
plt.ylabel('Obsérvation', fontsize = 20)
Fig
```

Out[5]: <AxesSubplot:xlabel='mois', ylabel='Obsérvation'>



1. Filtrage de la série temporelle avec Hodrick Prescott méthode.

Cette méthode permet de décomposer la série en 2 éléments à savoir la tendence, et la Saisonnalité.

Entrée [24]:
```
elmnt_saisonnier, elmnt_tendance = sm.tsa.filters.hpfilter(df['Valeures de la serie chrono'], lamb = 129600)
df_decomp = df[['Valeures de la serie chrono']]
df_decomp['Tendance'] = elmnt_tendance
df_decomp['saisonnalité'] = elmnt_saisonnier
df_decomp
```

Out[24]:

	Valeures de la serie chrono	Tendance	saisonnalité
2021-01-31	69.696666	50.884602	18.812064
2021-02-28	53.594893	52.534468	1.060424
2021-03-31	32.155608	54.184480	-22.028872
...	...	...	...
2030-10-31	247.349774	246.521789	0.827985
2030-11-30	230.345006	248.196036	-17.851030
2030-12-31	255.185182	249.870323	5.314859

120 rows × 3 columns

Pour vérifier que la somme de ces deux nouvelles colonnes vaut les valeurs de la série ajoutons une colonne intitulée vérification.

2. Vérification

Entrée [25]:
```
df_decomp['Vérification'] = df_decomp['Tendance'] + df_decomp['saisonnalité']
df_decomp
```

Out[25]:

	Valeures de la serie chrono	Tendance	saisonnalité	Vérification
2021-01-31	69.696666	50.884602	18.812064	69.696666
2021-02-28	53.594893	52.534468	1.060424	53.594893
2021-03-31	32.155608	54.184480	-22.028872	32.155608
...	...	...	...	...
2030-10-31	247.349774	246.521789	0.827985	247.349774
2030-11-30	230.345006	248.196036	-17.851030	230.345006
2030-12-31	255.185182	249.870323	5.314859	255.185182

120 rows × 4 columns

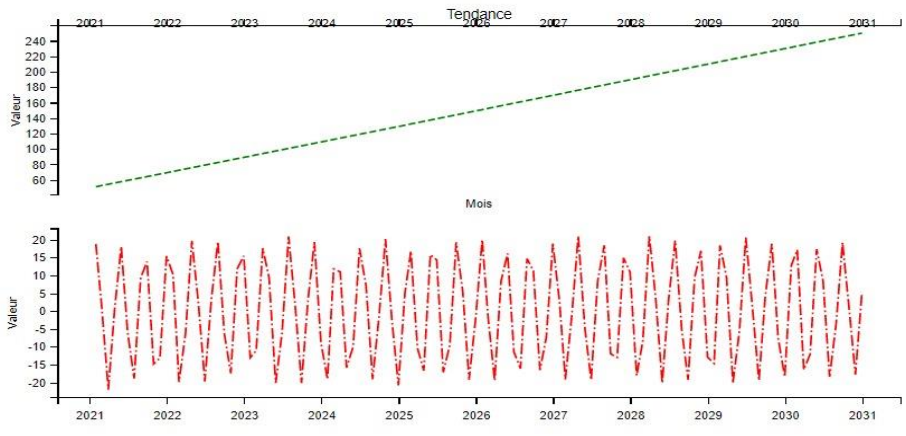
On a bien l'égalité comme prévu !

3. Visualiser les deux composantes.

Entrée [26]:
```
fig1, (ax1, ax2) = plt.subplots(2, figsize=(12, 6),
sharex=True)
ax1.plot(df_decomp.index.values,df_decomp['Tendance'] , color='green', linestyle='--',label='Tandence')

ax2.plot(df_decomp.index.values,df_decomp['saisonnalité'] , color='red', linestyle='-.',label='Saisonnalité')
ax1.set_title('Tendance')
ax1.set_xlabel('Mois')
ax2.set_ylabel('Valeur')
ax1.set_ylabel('Valeur')
```

Out[26]:
```
Text(0, 0.5, 'Valeur')
```



1.Décomposer la série avec la méthode UnobservedComponents (UC) de la biblio statsmodels.

Entrée [30]:
```
objet = sm.tsa.UnobservedComponents(df['Valeures de la serie chrono'],
                                    level = 'lltrend',
                                    cycle = True,
                                    stochastic_cycle = True)
resultat = objet.fit(method = 'powell', disp = True)
```

```
Optimization terminated successfully.
         Current function value: 4.101180
         Iterations: 6
         Function evaluations: 369
```

Entrée [31]:
```
decomp_UC = df[['Valeures de la serie chrono']]
decomp_UC['UC_Tendance'] = resultat.level.smoothed
decomp_UC['Uc_Saisonnalité'] = resultat.cycle.smoothed
decomp_UC['Uc_residuelle'] = resultat.resid
decomp_UC
```

Out[31]:

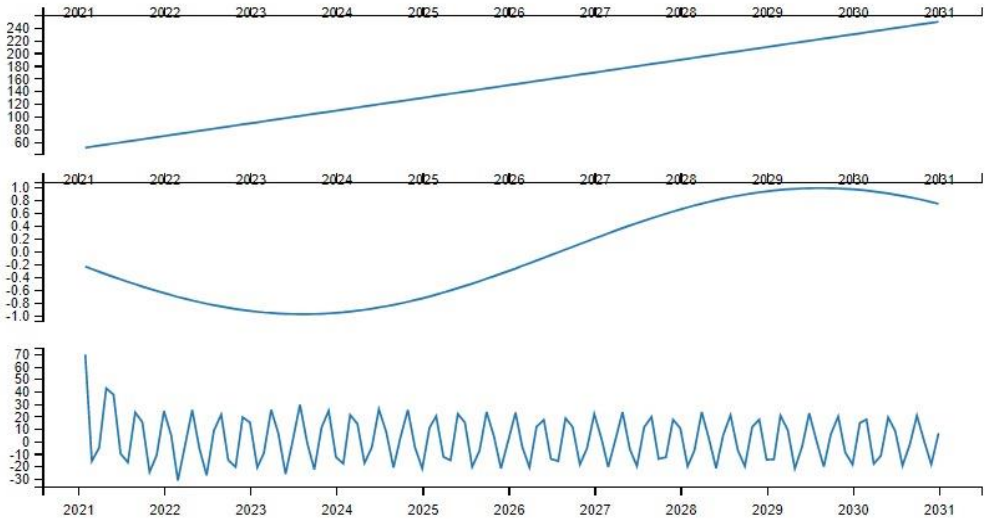
	Valeures de la serie chrono	UC_Tendance	Uc_Saisonnalité	Uc_residuelle
2021-01-31	69.696666	51.622510	-0.238845	69.696666
2021-02-28	53.594893	53.278127	-0.280860	-16.061062
2021-03-31	32.155608	54.933735	-0.322355	-5.289250
...	...	...	...	...
2030-10-31	247.349774	245.364604	0.792981	-0.020272
2030-11-30	230.345006	247.020329	0.766385	-18.709857
2030-12-31	255.185182	248.676338	0.738819	6.402572

120 rows × 4 columns

Illustration des résultats.

Entrée [34]:
```
fig1, (ax1, ax2, ax3) = plt.subplots(3, figsize=(12, 6),
sharex=True)
ax1.plot(decomp_UC.index.values,decomp_UC['UC_Tendance'] )
ax2.plot(decomp_UC.index.values,decomp_UC['Uc_Saisonnalité'] )
ax3.plot(decomp_UC.index.values,decomp_UC['Uc_residuelle'] )
```

Out[34]: `[<matplotlib.lines.Line2D at 0x296ebbe6820>]`



Série stationnaire : la particularité d'une série stationnaire est le fait que ces propriétés statistiques restent constantes, à savoir la moyenne, la variance …

Pourquoi une série stationnaire ?

L'estimation des séries non stationnaires conduit à des régressions fallacieuses ou illusoires. D'où la nécessité de la stationnarisation des séries chronologiques.

Vérification de la stationnarité d'une série chronologique

Pour vérifier si la série est stationnaire pu non, on calcule la p-valeur, c'est la probabilité quelle ne soit pas stationnaire, si cette probabilité est faible, on accepte l'hypothèse de stationnarité, sinon la série est non stationnaire.

Le test AUgmented Dickey-Fuller pour la stationnarité d'une série chronologique.

Entrée [35]:
```
from statsmodels.tsa.stattools import adfuller
```

Entrée [41]:
```
resultat = adfuller(df['Valeures de la serie chrono'])
print(f"Test statistique: {resultat[0]}\np-valeur : {resultat[1]}\ndélai : {resultat[2]}")
```

```
Test statistique: -0.03666862013177312
p-valeur : 0.9553925054369202
délai : 10
```

On peut transformer une série chronologique non stationnaire en stationnaire par des opérations mathématiques comme la soustraction.

Transformation de la série.

Entrée [15]:
```
df_diff = (df.shift(-1) - df).fillna(0)
```

Entrée [16]:
```
df_diff
```

Out[16]:

	Valeures de la serie chrono
2021-01-31	-15.086228
2021-02-28	-21.165948
2021-03-31	21.197791
...	...
2030-10-31	-16.540388
2030-11-30	25.432676
2030-12-31	0.000000

120 rows × 1 columns

Le test AUgmented Dickey-Fuller pour la stationnarité appliqué sur la série transformée.

Entrée [17]:
```
resultat = adfuller(df_diff)
print(f"Test statistique: {resultat[0]}\np-valeur : {resultat[1]}\ndélai : {resultat[2]}")
```

```
Test statistique: -5.13097944176684
p-valeur : 1.2141555842523469e-05
délai : 6
```

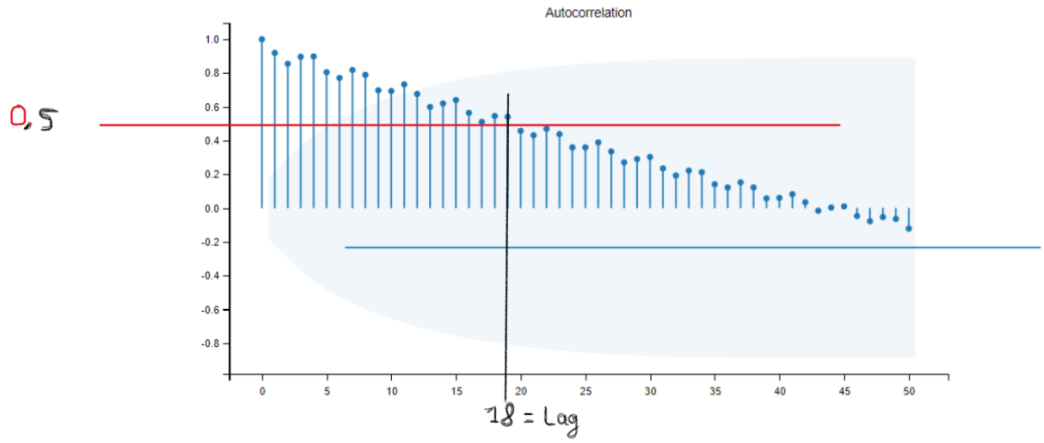
la p-valeur est très petite (1.2141555842523469e-05) donc la série est stationnaire.

L’autocorrélation :

L’autocorrélation d’une série chronologique est une mesure de similarité entre la série et elle-même précédemment, d’où le préfixe auto, l’autocorrélation dépend du délai quand on fixe entre les 2 séries (Décalage).

Autocorrelation

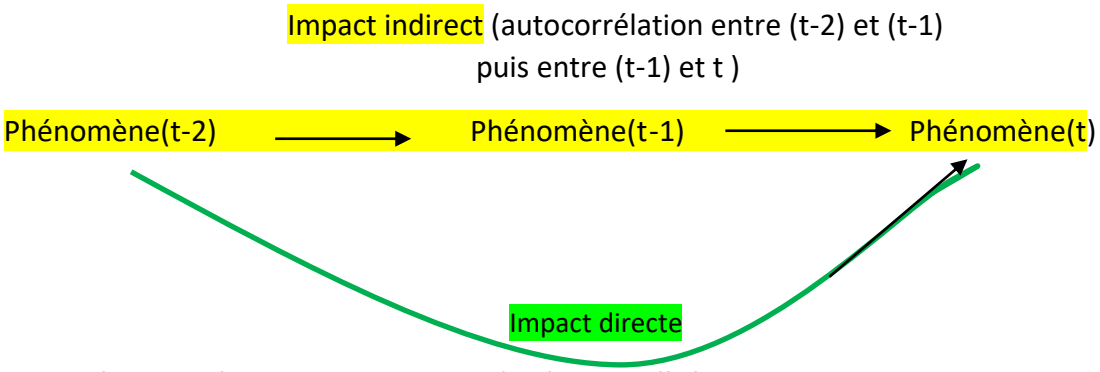
```
Entrée [20]: from statsmodels.graphics.tsaplots import plot_acf
             fig = plot_acf(df['Valeures de la serie chrono'], lags = 50)
             fig.set_size_inches(12,6)
```



Toute les séries qui s’écartent de la série originelle d’un délai qui ne dépassent pas 18 ont une forte corrélation avec la série (au delà de 0.5).

L’autocorrélation partielle :

Pour calculer cette autocorrélation on exclue l’impact indirect provenant des délais intermédiaires.

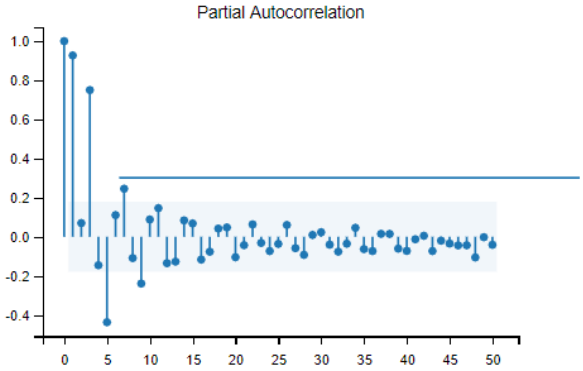


+ **Dans l’autocorrélation partielle on ne s’intéresse qu’à l’impact directe.**

Autocorrelation partielle.

Entrée [21]:
```
from statsmodels.graphics.tsaplots import plot_pacf
```

Entrée [22]:
```
plot_pacf(df['Valeures de la serie chrono'], lags = 50)
fig.set_size_inches(12,6)
```



elmoutaqui.hisham@gmail.com