Econométrie des séries temporelles univariées

Théorie et application sous Python

Duvérier DJIFACK ZEBAZE

Econométrie des séries temporelles univariées

Théorie et application sous Python

Duvérier DJIFACK ZEBAZE

Table des matières

Première partie

Analyse descriptive des séries temporelles

1 Introduction au séries temporelles

Sommaire

1.1 Définitions, objectifs et notations

On appelle série temporelle (ou série chronologique ou chronique), une suite d'observations numériques d'une grandeur effectuée à intervalle de temps régulier au cours du temps. Une série temporelle est donc une série d'observations $x_1, x_2, ..., x_T$ indexée par le temps. On suppose qu'il s'agit d'une réalisation d'un processus X c'est-à-dire d'une suite X_t de variable aléatoire. Pour chaque temps t, X_t est une variable aléatoire et x_t en est une réalisation. Par exemple, la série du chiffre d'affaire mensuel de la société CONGELCAM de Juin 2021 à Décembre 2022 donne lieu à une suite de 19 observations mensuelles $x_1, x_2, ..., x_{19}$ concernant respectueusement les différents mois de la période considérée.

La théorie des séries temporelles est appliquée dans les domaines aussi variés que la finance, les assurances, la médecine, la météorologie, l'économie, etc.... La variable étudiée peut par exemple être macroéconomique (le PIB d'un pays, l'inflation, etc...), microéconomique (vente d'une entreprise, son nombre d'employé, le revenu d'un individu, etc...), politique (nombre de votants, nombre de voix reçues par un candidat, etc...) ou démographique (la taille moyenne des habitants, l'âge, etc...).

1.1.1 Objectifs et notations

Les objectifs de l'étude des séries chronologiques sont en général de décrire, d'expliquer les variations du phénomène observé et de prédire les valeurs futures. Les prévisions sont faites à partir des valeurs observées de la série dans le passé. On définit la période d'une série chronologique (infra-annuelle) par le nombre d'observations dans l'année. On la note généralement p. La période peut être mensuelle ($p = 12$), trimestrielle ($p = 4$), etc...

Généralement, on distingue deux types de notations des séries temporelles. Dans la première notation, les observations sont repérées à l'aide d'un seul indice t. Cet indice représente l'intervalle de temps séparant deux observations consécutives. Dans ce cas la série est notée par X_t. Par contre, dans le seconde, les observations sont repérées à travers deux indices i et j où

i représente l'année et j représente la période d'observation (le mois, le trimestre, le semestre, etc…). La série est notée dans ce cas par $X_{i,j}$.

Les deux types de notations sont étroitement liés. Voyons cela à travers les deux exemples suivants :

Exemple 1.1

Soit une série trimestrielle avec $X_{4,2} = 40$. $X_{4,2}$ est l'observation du 2^e trimestre de la 4^e année. En terme de trimestre, on a : $[(4-1) \times 4] + 2 = 14$. Ainsi donc $X_{4,2} = X_{14}$.

Exemple 1.2

Soit une série mensuelle avec $X_{2,6} = 40$. $X_{2,6}$ est l'observation du 6^e mois de la 2^e année. En termes de mois, on a : $[(2-1) \times 12] + 6 = 18$. Ainsi donc $X_{2,6} = X_{18}$.

De façon générale, si on note par p la période la série, on a : $X_{i,j} = X_t$ avec :

$$t = [(i-1) \times p] + j \tag{1.1}$$

Le tableau ci-après permet de récapituler la liaison en $X_{i,j}$ et X_t pour une série trimestrielle observée durant 3 années.

année \ trimestre	1	2	3	4
année 1	$t=1$	$t=2$	$t=3$	$t=4$
année 2	$t=5$	$t=6$	$t=7$	$t=8$
année 3	$t=9$	$t=10$	$t=11$	$t=12$

1.1.2 Représentation d'une chronique

1.1.2.1 Représentation sous forme de tableau

La série peut être représentée sous la forme d'un tableau à deux colonnes (ou deux lignes) et $n \times p$ lignes (ou $n \times p$ colonnes) avec p la période et n le nombre d'année.

t	1	2	···	···	$n \times p$
X_t	X_1	X_2	···	···	$X_{n \times p}$

La série peut être aussi représentée dans un tableau contenant p colonnes et n lignes.

i \ j	1	···	j	···	p
1	$X_{1,1}$	···	$X_{1,j}$	···	$X_{1,p}$
···					
i	$X_{1,1}$	···	$X_{i,j}$	···	$X_{i,p}$
···					
n	$X_{n,1}$	···	$X_{n,j}$	···	$X_{n,p}$

Exemple 1.3 *Entreprise EYENGA*

Considérons la série trimestrielle de chiffre d'affaire (en millions de FCFA) de l'entreprise EYENGA de 2020 à 2022.

ou encore

A ces deux types de notation, on peut associer deux types de graphiques

Table 1.1 – Chiffre d'affaire de l'entreprise EYENGA

t	1	2	3	4	5	6	7	8	9	10	11	12
X_t	26	12	30	15	17	10	11	9	8	7	11	12

année \ trimestre	1	2	3	4
2020	26	12	30	15
2021	17	10	11	9
2022	8	7	11	12

1.1.2.2 Représentation graphique

On peut faire deux types de graphiques selon le type de tableau. La première forme représente l'évolution de la série de façon globale (*cf.* Figure 1.1). On trace le nuage de points (t, x_t) et on relie les points.

```
# Chargement des données
import seaborn as sns
flights = sns.load_dataset("flights")
```

Nous allons transformer notre jeu données de façon à avoir des mois pour chaque observation sous forme de date. Nous supprimons les colonnes `month` et `year`.

```
# Suppression de year et month
donnee = flights.drop(columns=["year", "month"])
# Transformation
import pandas as pd
donnee.index = pd.date_range(start="1949-01-31",end = "1960-12-31",freq="M")
```

La figure 1.1 représente le total mondial des passagers aériens par mois entre 1949 et 1960.

```
# Représentation graphique
import matplotlib.pyplot as plt
fig, axe = plt.subplots(figsize=(16,6));
donnee.passengers.plot(ax=axe,color="black");
axe.set_xlabel('année');
axe.set_ylabel('valeur');
plt.show()
```

La deuxième forme représente la série de manière infra-annuelle (*cf.* Figure **??**). On fait un graphique pour chaque année (trimestre, etc.). On parle dans ce cas, de graphique des courbes superposées.

```
# Représentation graphique
sns.relplot(data=flights,x="month",y="passengers",hue="year",kind="line",
            height=4, aspect=2);
```

Remarque 1.1

Cette deuxième forme de représentation permet de comparer les mêmes périodes pour différentes années. Cependant, elle ne permet pas de voir l'évolution globale de la série.

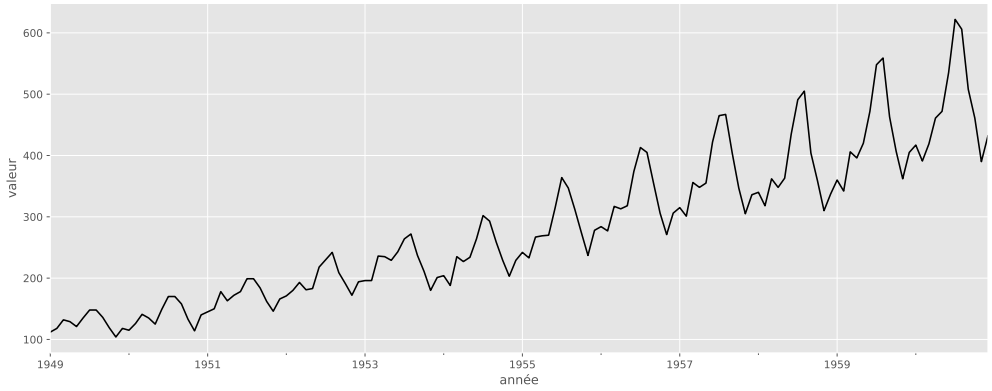


Figure 1.1 – Données Air passengers (évolution globale)

1.1.2.3 Lag plot

Un lag plot ou diagramme retardé est le diagramme de dispersion des points ayant pour abscisse la série retardée de h instants et pour ordonnée la série non retardée. Si le diagramme retardé suggère une corrélation entre les deux séries, on dit que la série présente une autocorrélation d'ordre h. Ce diagramme permet de comprendre la dépendance de la série par rapport à son passé. Il donne une vision locale de la série : y a-t-il une corrélation entre la série à un instant et la série $1, 2, \ldots$ instants avant ? Évidemment, si une telle dépendance apparaît, on doit en comprendre le mécanisme : s'agit-il d'une simple corrélation empirique sans pendant inférentiel ou bien d'une autocorrélation empirique qui traduit une corrélation entre variables aléatoires ?

Exemple 1.4 *Lag plot de la série Air passengers*

Sous Python, la fonction pandas.plotting.lag_plot permet de visualiser le lag plot (*cf.* Figure 1.2).

```
# lag plot
fig, axe = plt.subplots(1,2,figsize=(16,6));
pd.plotting.lag_plot(donnee.passengers,lag = 1,ax = axe[0],c = 'black');
pd.plotting.lag_plot(donnee.passengers,lag = 2,ax = axe[1],c = 'black');
plt.tight_layout();
plt.show()
```

1.1.2.4 Month plot

Un month plot est une représentation simultanée des chronogrammes des séries associées à chaque saison, appelée conventionnellement mois. Il dessine un chronogramme de la sous-série correspondant à chaque niveau de la saison (mois de l'année, jour de la semaine,...). Les points du chronogramme du mois j correspondent à la série du mois j, pour toutes les années. C'est une représentation pertinente pour étudier la saisonnalité d'une série, qu'elle soit mensuelle ou autre.

Exemple 1.5 *Month plot de la série Air passengers*

Sous Python, la fonction month_plot permet de visualiser un month plot (*cf.* Figure 1.3).

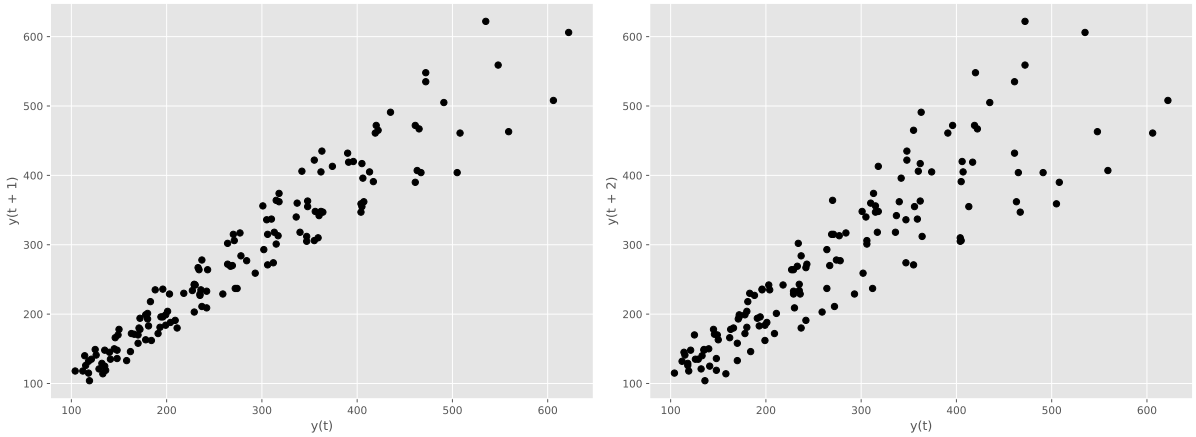


Figure 1.2 – Lag plot de la série Air passengers

```
# Month plot
import statsmodels.api as sm
fig, axe = plt.subplots(figsize=(16,6));
sm.graphics.tsa.month_plot(donnee.passengers,ax=axe);
axe.set_xlabel('mois');
axe.set_ylabel('valeur');
plt.show()
```

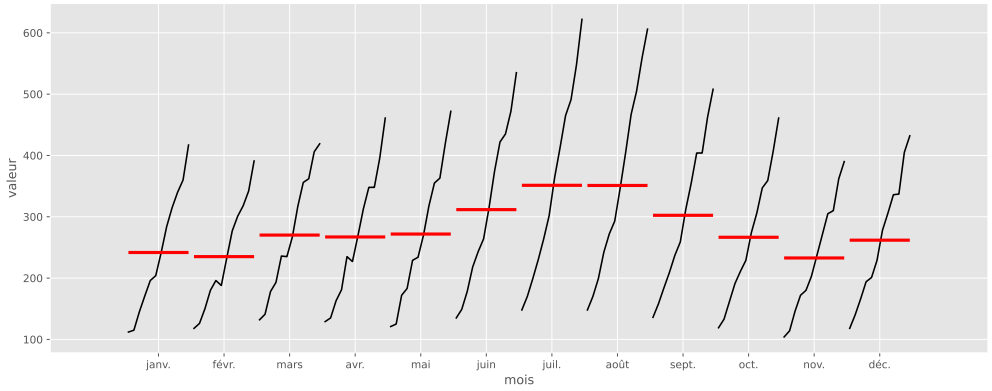


Figure 1.3 – Month plot de la série Air passengers

Le trait horizontal dans chaque chronogramme de saison indique la valeur moyenne de la saison.

1.1.3 Les composantes d'une chronique

Pour modéliser une série chronologique (X_t), on la considère en générale comme la résultante de différentes composantes parmi lesquelles la tendance, la saisonnalité et la composante résiduelle.

1.1.3.1 La tendance ou trend

Notée généralement Z_t, la tendance représente l'évolution à long terme de la série étudiée. Elle traduit le comportement moyen de la série. C'est aussi l'évolution fondamentale de la série. On distingue plusieurs types de tendances :

— Tendance affine : $Z_t = \alpha_0 + \alpha_1 t$
— Tendance polynomiale : $Z_t = \sum_{k=0}^{k=p} \alpha_k t^k$ où p est le degré le plus élevé du polynôme.
— Tendance logarithmique : $Z_t = \alpha_0 + \alpha_1 \ln(t)$
— etc...

On considère l'équation de tendance suivante :

$$Z_t = -1.0875 + 2.75t, \quad t = 1, \dots, 50 \tag{1.2}$$

```
# Tendance linéaire
t = np.arange(1, 51)
Trend = -1.0875 + t*2.75
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6));
axe.scatter(t, Trend, color = "black");
axe.set_xlabel("Minutes");
axe.set_ylabel("Demande de production");
plt.show()
```

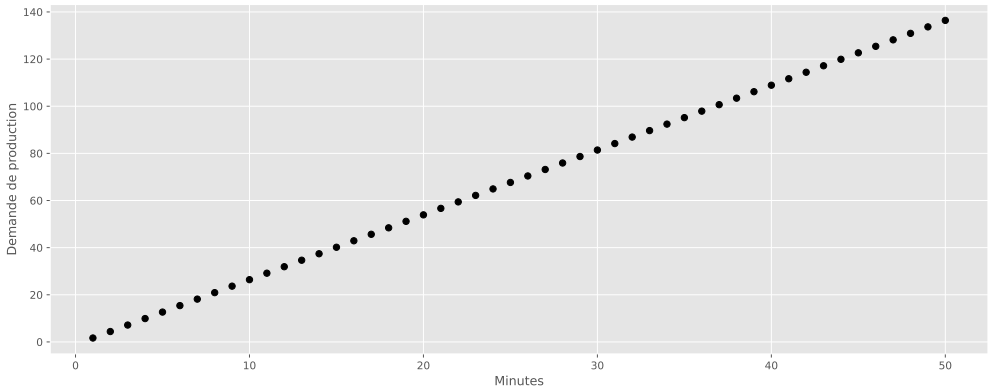


Figure 1.4 – Evolution de la tendance linéaire : $Z_t = -1.0875 + 2.75 \times t$

1.1.3.2 La composante saisonnière

Encore appelée variation saisonnière, elle correspond à des fluctuations périodiques à l'intérieur d'une année et qui se reproduisent de manière identique d'une année à l'autre. On la note généralement S_t. Il s'agit d'un phénomène qui se répète à l'identique à intervalle de temps régulier ou périodique. C'est une fonction périodique de période égale à la période d'observation (p).

$$S_{t+p} = S_t \quad \forall t \tag{1.3}$$

On considère la composante saisonnière $(S_t)_t$ donnée par l'équation suivante :

$$S_t = 10 + 10sin(t), \quad t = 1, \dots, 50 \tag{1.4}$$

```
# Composante saisonnière
seasonality = 10 + np.sin(t) * 10
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6));
axe.plot(t,seasonality,color = "black");
axe.set_xlabel("Minutes");
axe.set_ylabel("Saisonnalité");
plt.show()
```

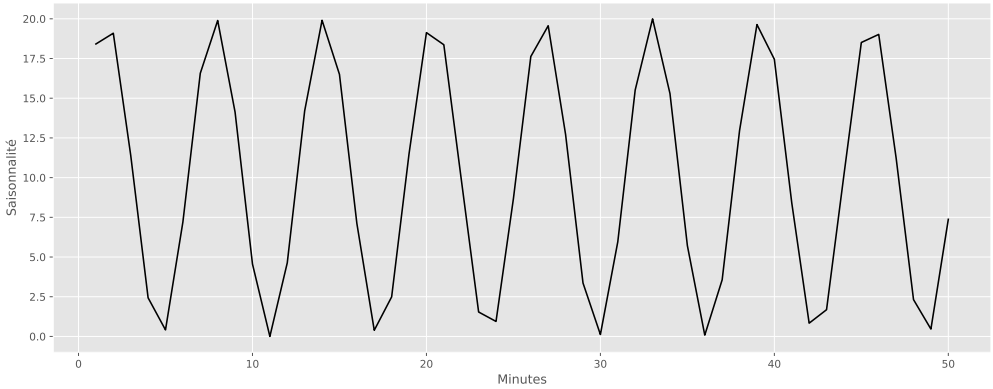


Figure 1.5 – Evolution de la saisonnalité $S_t = 10 + 10\sin(t)$

1.1.3.3 La composante résiduelle ou bruit

Encore appelée variation résiduelle ou accidentelle, elle correspond à des fluctuations irrégulières qui sont en général de faible intensité mais de nature imprévisible. Elles proviennent des circonstances imprévisibles : catastrophes naturelles, les crises boursières, les grèves, etc….

Supposons que la composante résiduelle $(\varepsilon_t)_t$ suit une distribution normale centrée et réduite, c'est - à - dire :

$$\varepsilon_t \sim \mathcal{N}(0,1) \tag{1.5}$$

```
# Bruit gaussien
np.random.seed(10)
residual = np.random.normal(loc=0.0, scale=1, size=len(t))
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6));
axe.plot(t,residual,color = "black");
axe.set_xlabel("Minutes");
axe.set_ylabel("résidu");
plt.show()
```

1.1.3.4 La composante cyclique

Elle exprime les mouvements irréguliers de déviation de la variable autour de la tendance de la série sur les périodes de temps relativement longue (par exemple les phases économiques

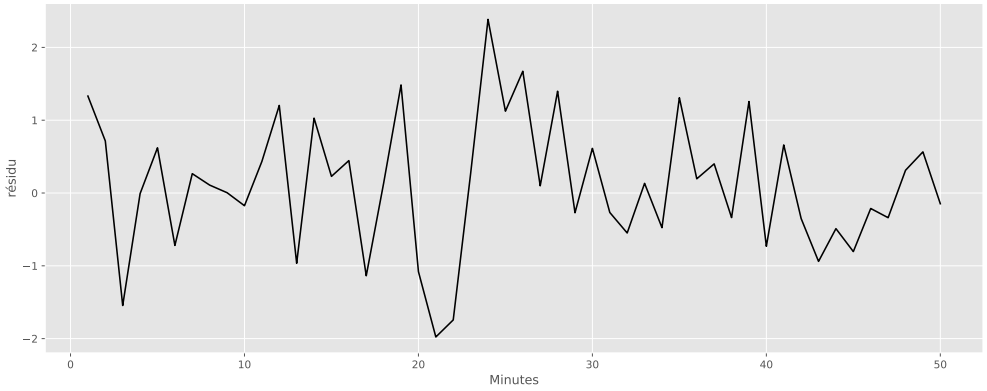


Figure 1.6 – Evolution de la composante résiduelle : $\varepsilon_t \sim \mathcal{N}(0,1)$

d'expansion et de récession qui ne sont pas de durée fixe). On la note C_t. Sous information spécifique, il est très difficile de dissocier la tendance du cycle.

1.1.4 Les principaux modèles de composition d'une chronique

Un modèle est une représentation simplifiée de la réalité qui vise à traduire le fonctionnement d'un phénomène étudié et permet ainsi de mieux le comprendre. On distingue en général les principaux modèles ci-après.

1.1.4.1 Le modèle additif

Avec ce modèle, les trois composantes globales d'une série chronologique sont reliées par une relation du type :

$$X_t = Z_t + S_t + \varepsilon_t \quad \forall t = 1, ..., T \tag{1.6}$$

Dans cette relation : X_t représente la série considérée ; Z_t est la tendance (qui peut se décomposer en une tendance lourde souvent représentée par une fonction polynôme du temps plus d'un cycle de périodicité supérieure à un an) ; S_t est la composante saisonnière de périodicité 4, 12, 52 ou 365 selon qu'il s'agit des données trimestrielles, mensuelles, hebdomadaires ou journalières. La période p est l'étendue des intervalles de temps successifs sur lesquels le phénomène se reproduit de manière analogue. On a ainsi : $S_{t+p} = S_t \quad \forall t$.

ε_t représente la variation aléatoire due à de nombreuses causes pas forcément bien identifiées de répercussion limitée. Dans le vocabulaire conventionnel des séries chronologiques, on emploie également les termes innovations, irrégularités, erreurs, résidus pour désigner la variation aléatoire. Il s'agit des variables aléatoires (généralement centrées, on considère le plus souvent un bruit blanc).

```
# Modèle additif
additive_model = Trend + seasonality + residual
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6));
axe.plot(t,additive_model,color="black");
axe.set_xlabel("temps");
```

```
axe.set_ylabel("valeur");
plt.show()
```

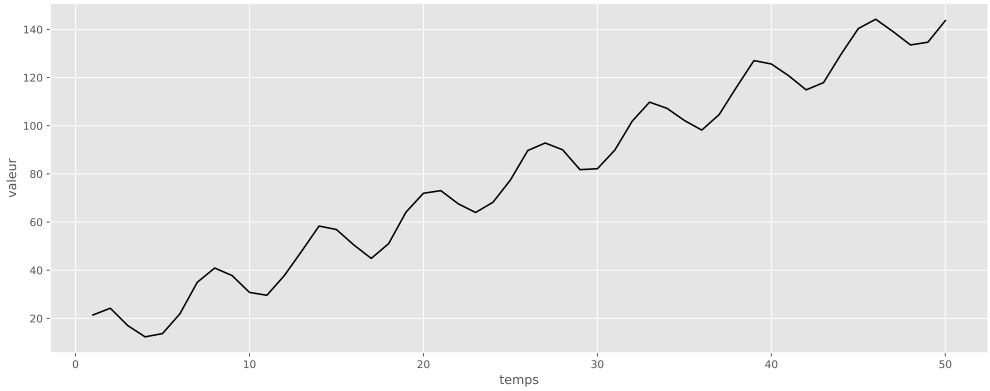


Figure 1.7 – Exemple d'un modèle additif : $X_t = Z_t + S_t + \varepsilon_t$

Précisons que le modèle additif est utilisé lorsque le mouvement a une amplitude constante au cours du temps.

1.1.4.2 Le modèle multiplicatif

Ce modèle est généralement utilisé lorsque le mouvement saisonnier est d'une modulation qui varie au cours du temps. Ici, on suppose que les variations saisonnières ainsi que les variations résiduelles dépendent de la tendance. La série s'écrit alors [1] :

$$X_t = Z_t \times S_t \times \varepsilon_t, \quad \forall t = 1, \dots, T \tag{1.7}$$

```
# Modèle multiplicatif
ignored_residual = np.ones_like(residual)
multiplicative_model = Trend * seasonality * ignored_residual
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6));
axe.plot(t,multiplicative_model,color="black");
axe.set_xlabel("temps");
axe.set_ylabel("valeur");
plt.show()
```

Remarque 1.2

Si X_t est à valeurs positives, le modèle ci-dessous peut être ramené à un modèle additif à l'aide de la transformation de type logarithme :

$$\ln X_t = \ln Z_t + \ln S_t + \ln \varepsilon_t, \quad \forall t = 1, \dots, T \tag{1.8}$$

1. Dans certains ouvrages, le modèle multiplicatif a la forme suivante : $X_t = Z_t(1 + S_t)(1 + \varepsilon_t)$.

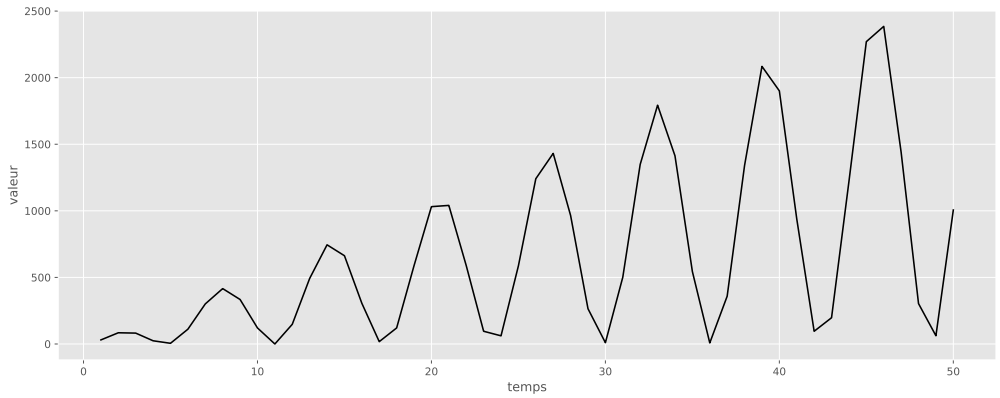


Figure 1.8 – Exemple d'un modèle multiplicatif : $X_t = Z_t \times S_t \times \varepsilon_t$

En effectuant un changement de variable, on pose : $X_t^{'} = \ln X_t$, $Z_t^{'} = \ln Z_t$, $S_t^{'} = \ln S_t$ et $\varepsilon_t^{'} = \ln \varepsilon_t$, on obtient :

$$X_t^{'} = Z_t^{'} + S_t^{'} + \varepsilon_t^{'}, \quad \forall t = 1, \dots, T \tag{1.9}$$

1.1.4.3 Le modèle pseudo - additif ou modèle mixte

Pour ce type de modèle, on suppose que les variations saisonnières dépendent de la tendance et sont indépendantes des variations résiduelles. La série s'écrit alors :

$$X_t = Z_t \times S_t + \varepsilon_t, \quad \forall t = 1, \dots, T \tag{1.10}$$

Graphiquement, l'amplitude des variations saisonnières varie le long de la tendance. Le modèle pseudo - additif est utilisé lorsque la chronique comporte de faibles valeurs ou des zéros.

```
# Modèle pseudo - additif
pseudo_additif_model = Trend * seasonality + residual
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6));
axe.plot(t,pseudo_additif_model,color="black");
axe.set_xlabel("temps");
axe.set_ylabel("valeur");
plt.show()
```

1.2 Choix du modèle

Avant toute modélisation et étude approfondie, on tente d'abord de déterminer si on est en présence d'une série dans laquelle, pour une observation X donnée, la variation saisonnière S s'ajoute simplement à la tendance Z (modèle additif) ou est proportionnelle à la tendance (modèle multiplicatif). Deux types de méthodes permettent de faire cette distinction : les méthodes graphiques et La méthode analytique.

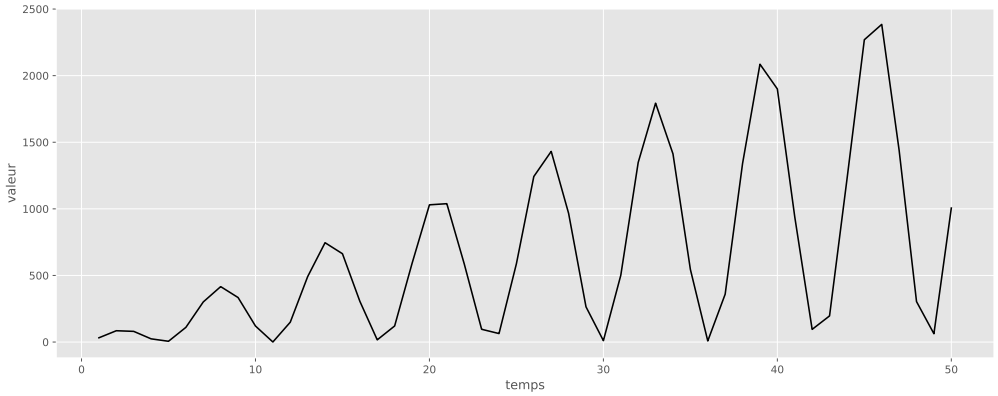


Figure 1.9 – Exemple d'un modèle pseudo - additif : $X_t = Z_t \times S_t + \varepsilon_t$

1.2.1 Les méthodes graphiques

1.2.1.1 Méthode de la bande

La méthode de la bande consiste à joindre les maxima entre eux et les minima entre eux du mouvement brut. On projette verticalement les maxima sur la ligne des minima ou sur la courbe enveloppe basse, on projette également les minima sur la ligne des maxima. La tendance relit les différents points médians. Si les deux courbes sont parallèles, le modèle est de type additif. Sinon, le modèle est de type multiplicatif.

1.2.1.2 Méthode du profil

Pour cette méthode, on utilise le graphique des courbes superposées. Si les courbes obtenues sont à peu près parallèles, alors le modèle est de type additif. Dans le cas contraire, les pics et les creux s'accentuent, on choisit le modèle est de type multiplicatif.

1.2.2 La méthode analytique : Méthode de Buys-Ballot

La méthode analytique de Buys - Ballot (*cf.* Iwueze et al. (2011)) est basée sur la moyenne et l'écart-type. Son principe est le suivant :

1. On calcule les moyennes $\overline{X}_i$ et les écarts types σ_i pour chacune des périodes considérées ;
2. On trace le nuage des points d'abscisse la moyenne $\bar{x}$ et d'ordonnée l'écart-type σ ;
3. On établit la droite des moindres carrés : $\sigma = a\overline{X} + b$.

Si l'écart-type est indépendante de la moyenne, c'est-à-dire que la pente de la droite a est significativement nulle, alors le modèle est de type additif. Sinon, le modèle est de type multiplicatif.

Généralement, on se base sur le test de Student en testant l'hypothèse nulle $H_0 : a = 0$ contre l'hypothèse alternative $H_1 : a \neq 0$. Si la pvalue associée au coefficient a est inférieure au seuil de 0.05, on rejette l'hypothèse nulle H_0 au seuil de 5%. Le coefficient de la pente est significativement non nulle. Par conséquent le modèle est multiplicatif. Sinon, le modèle est additif.

Exemple 1.6 *Application de la méthode de Buys - Ballot sur la série Air passengers*

On cherche à déterminer le schéma de décomposition de la série Air passengers en utilisant la méthode de Buys - Ballot. Pour cela, on se sert des données du tableau 1.2.

```
# Transformation en série infra-annuelle
data = flights.pivot(index="year",columns="month",values="passengers")
```

Table 1.2 – Données Air passengers (infra - annuelle)

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1949	112	118	132	129	121	135	148	148	136	119	104	118
1950	115	126	141	135	125	149	170	170	158	133	114	140
1951	145	150	178	163	172	178	199	199	184	162	146	166
1952	171	180	193	181	183	218	230	242	209	191	172	194
1953	196	196	236	235	229	243	264	272	237	211	180	201
1954	204	188	235	227	234	264	302	293	259	229	203	229
1955	242	233	267	269	270	315	364	347	312	274	237	278
1956	284	277	317	313	318	374	413	405	355	306	271	306
1957	315	301	356	348	355	422	465	467	404	347	305	336
1958	340	318	362	348	363	435	491	505	404	359	310	337
1959	360	342	406	396	420	472	548	559	463	407	362	405
1960	417	391	419	461	472	535	622	606	508	461	390	432

```
# Test de Buys - Ballot
meansd = pd.concat([data.mean(axis = 1),data.std(axis=1)],axis = 1)
meansd.columns = ["Mean", "SD"]
```

Table 1.3 – Moyennes et écart types annuelles

	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
Mean	126.667	139.667	170.167	197.000	225.000	238.917	284.00	328.250	368.417	381.00	428.333	476.167
SD	13.720	19.071	18.438	22.966	28.467	34.924	42.14	47.862	57.891	64.53	69.830	77.737

On trace un nuage de points avec en abscisse la moyenne et en ordonnée l'écart-type.

```
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6));
axe.scatter(meansd.Mean, meansd.SD,color="black");
axe.set_xlabel('moyenne');
axe.set_ylabel('ecart-type');
plt.show()
```

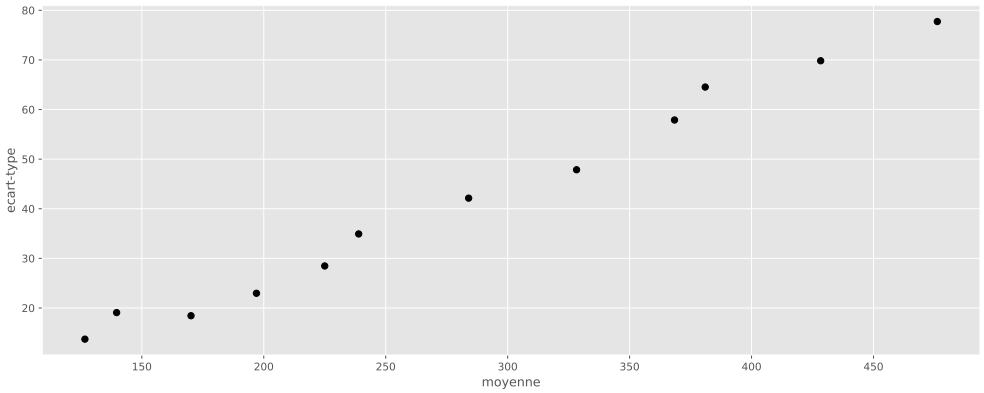


Figure 1.10 – Nuage des points de l'écart type en fonction de la moyenne

```
# Estimation de la droite des Buys - Ballot : y = ax + b
import statsmodels.formula.api as smf
model = smf.ols('SD~Mean', data = meansd).fit()
```

On construit une fonction d'extraction des paramètres.

```
# Extraction des paramètres
from mizani.formatters import scientific_format
def extractParams(res, model_type = "lm", seuil = 0.05,approx = 4):
  stderr = list()
  if model_type == "lm"or model_type == "arima":
    stderr = np.sqrt(np.diag(res.cov_params()))
  elif model_type == "arch":
    stderr = np.sqrt(np.diag(res.param_cov))
  level_inf = seuil/2; level_sup = 1 - level_inf
  params = pd.DataFrame({
    "coef" : res.params, "std err" : stderr, "t" : res.tvalues,
    "P>|t|" : scientific_format()(res.pvalues),
    "["+str(level_inf) :list(res.conf_int(alpha = seuil).values[:,0]),
    str(level_sup)+"]" :list(res.conf_int(alpha = seuil).values[:,1]),
    },index = list(res.params.index)).round(approx)
  return params

# Application
model_params = extractParams(model,model_type = "lm", seuil = 0.05,approx = 4)
```

Table 1.4 – Coefficients du modèle linéaire : $\sigma = a\overline{X} + b$

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-11.4033	1.9828	-5.7511	1.849e-04	-15.8212	-6.9853
Mean	0.1886	0.0066	28.6763	6.192e-11	0.1740	0.2033

On a une pvalue nulle, par conséquent on rejette l'hypothèse nulle ($H_0 : a = 0$) au seuil de 5%. Le coefficient de la pente est significativement non nul. Le modèle est multiplicatif.

1.2.3 Les étapes de la modélisation d'une chronique

Les principales étapes de modélisation d'une série chronologique peuvent se résumer comme suit :

1.2.3.1 Correction des données

L'étude d'une série chronologique peut en effet nécessiter une modification préalable des données brutes. Il peut s'agit par exemple : d'évaluer les données manquantes ; de remplacer les données accidentelles ; d'un découpage en sous série ; d'une standardisation des données afin de se ramener à des intervalles de longueur fixe. On peut également décider de travailler sur les données transformées plutôt que sur les données brutes. Par exemple, on peut utiliser la transformatée de Box - Cox (*cf.* Davidson, MacKinnon, et al. (1993)).

Définition 1.1 *Transformée de Box - Cox*

Pour toute valeur de x positive, on définit la transformée de Box - Cox de la manière suivante :

$$Y_t = \begin{cases} \dfrac{X_t^\lambda - 1}{\lambda} & \text{si} \quad \lambda \neq 0 \\ \ln(X_t) & \text{si} \quad \lambda = 0 \end{cases}, \quad \forall t = 1, \dots, T \tag{1.11}$$

Si λ est supérieur à 1, la transformée de Box - Cox amplifie les grandes valeurs X_t. Dans le cas contraire ($\lambda < 1$), elle réduit ces grandeurs valeurs.

Exemple 1.7 *Transformée de Box - Cox de la série Air passengers*

Sous Python, les fonctions scipy.stats.boxcox et scipy.special.boxcox permettent d'effectuer la transformée de Box - Cox pour une série.

```
# Transformation de box-cox
import scipy.stats as st
#perform Box-Cox transformation on original data
boxcox, lambd = st.boxcox(donnee.passengers)
# Représentation graphique
transformed_data = pd.DataFrame(boxcox,index = donnee.index,columns=['boxcox'])
fig, axe = plt.subplots(figsize=(16,6));
transformed_data.boxcox.plot(ax=axe,color="black",label="$\lambda$="+str(lambd));
axe.set_xlabel('année');
axe.set_ylabel('valeur');
axe.legend();
plt.show()
```

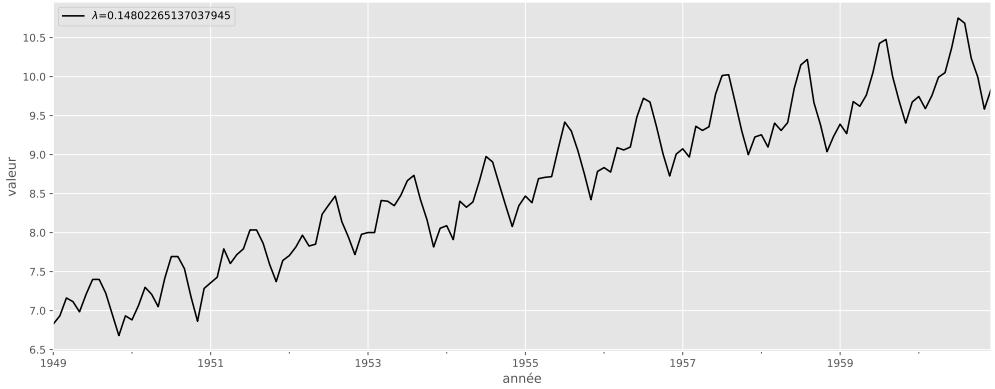


Figure 1.11 – Données Air passengers (Box - Cox transformation avec $\lambda = 0.148$)

1.2.3.2 Observation graphique de la série

Une fois la série corrigée, on peut la représenter graphiquement en vue de visualiser son évolution en fonction du temps. L'observation de ce graphique est souvent une aide à la modélisation de la série chronologique car elle permet non seulement de se faire une idée des différentes composantes de la série mentionnées plus haut, mais aussi du type de modèle correspondant (additif ou multiplicatif).

1.2.3.3 Modélisation

Un modèle est une image simplifiée de la réalité qui vise à traduire les mécanismes de fonctionnement du phénomène étudié et permet de mieux les comprendre. Un modèle peut être meilleur qu'un autre pour décrire la réalité et bien sûr, plusieurs questions se posent alors : comment mesurer cette qualité ? Comment diagnostiquer un modèle ?

Dans cette partie, on cherche à choisir le modèle le mieux adapté de notre série parmi les modèles possibles. On distingue généralement deux types de modèle : les modèles déterministes et les modèles stochastiques.

a) Les modèles déterministes

Les modèles déterministes relèvent de la statistique descriptive. Ils ne font intervenir que de manière sous-jacente le calcul des probabilités et consiste à supposer que l'évolution de la série à la date t est une fonction du temps t et d'une variable aléatoire ε_t centrée faisant office d'erreur au modèle, représentant la différence entre la réalité et le modèle proposé :

$$X_t = f(t, \varepsilon_t) \tag{1.12}$$

On suppose de plus que les ε_t sont décorrélées.

On modélise un processus par la somme d'une partie déterministe et d'une partie aléatoire (modèle additif) ou par le produit d'une partie déterministe et d'une partie aléatoire (modèle multiplicatif).

b) Les modèles stochastiques

Les modèles stochastiques sont du même type que les modèles déterministes à ceci près que les variables de bruit ε_t ne sont pas indépendantes et identiquement distribuées (i.i.d.) mais possèdent une structure de corrélation non nulle : ε_t est une fonction des valeurs passées ($\pm$ lointaines suivant le modèle) et d'un terme d'erreur η_t

$$\varepsilon_t = g(\varepsilon_{t-1}, \varepsilon_{t-2}, \dots, \eta_t) \tag{1.13}$$

La classe des modèles de ce type la plus fréquemment utilisée est la classe des modèles SARIMA (et de ses sous - modèles ARIMA, ARMA,…)

Les deux types de modèles ci - dessus induisent des techniques de prévision bien particulières. Schématiquement,on s'interesse tout d'abord à la tendance et à la saisonnalité éventuelle(s) que l'on isole tout d'abord. Ensuite on cherche à les modéliser, les estimer. Enfin on les élimine de la série : ces deux opérations s'appellent la détendancialisation et la désaisonnalisation de la série. Une fois ces composantes éliminées, on obtient la série aléatoire ε_t :

- Pour les modèles déterministes, cette série sera considérée comme décorrélée et il n'y a plus rien à faire.
- Pour les modèles stochastiques, on obtient (du moins en l'espérant) une série stationnaire (ce qui signifie que les observations successives de la série sont identiquement distribuées mais pas nécessairement indépendantes) qu'il s'agit de modéliser.

1.2.4 Analyse de la série à partir de ses composantes et prévision

1.2.4.1 Analyse de la série

Une fois l'étape de modélisation effectuée, on étudie les composantes du modèle les unes après les autres.

a) La série corrigée de la tendance

La tendance agit comme une forte corrélation entre les variables X_t mais cette corrélation n'exprime aucune liaison à caractère explicatif. Il s'agit donc d'isoler cette tendance puis de l'étudier à partir et enfin de l'éliminer de la série pour voir si des liaisons à caractère explicatif existent et étudier seulement ces corrélations sans tendance. On déduit la série corrigée de la tendance $(X_{CST,t})_t$ en supprimant la tendance. La série détendancialisée est :

— Pour le modèle additif : $X_{CT,t} = S_t + \varepsilon_t$.
— Pour le modèle multiplicatif : $X_{CT,t} = S_t \times \varepsilon_t$.

b) La série corrigée des variations saisonnières

Dans le même ordre d'idée, nous corrigerons les éventuelles variations saisonnières qui résultent d'un comportement périodique dans la série observée. Par exemple, considérons la figure représentant le nombre de passagers mensuels (*cf.* Figure 1.1) : on observe une relation directe entre la période de l'année et le nombre de passagers.

Pour pouvoir réellement comparer l'effectif des passagers d'un mois à l'autre, on doit supprimer l'effet de la saisonnalité et on définit la série corrigée des variations saisonnières $(X_{CVS,t})_t$ en supprimant la composante saisonnière $(S_t)_t$ du modèle. La série désaisonnalisée est :

— Pour le modèle additif : $X_{CVS,t} = Z_t + \varepsilon_t$
— Pour le modèle muliplicatif : $X_{CVS,t} = Z_t\varepsilon_t$.

c) La série lissée des prédictions

On définit la série lissée des prédictions $(X_{LP,t})_t$ en supprimant les fluctuations irrégulières $(\varepsilon_t)_t$ du modèle.C'est à partir de cette série que nous ferons les prédictions et en utilisant les modélisations et estimations de la tendance et de la saisonnalité. Par exemple, après avoir supprimé les fluctuations irrégulières, on déduit :

— Pour le modèle additif : $X_{LP,t} = Z_t + \varepsilon_t$.
— Pour le modèle multiplicatif : $X_{LP,t} = Z_t S_t$

1.2.4.2 Diagnostic du modèle

Une fois le modèle construit et ses paramètres estimés, on vérifie que le modèle proposé est bon. Ici, on teste sa validité en étudiant les résidus et en effectuant les différents tests.

1.2.4.3 Prévision

Enfin, une fois ces différentes étapes terminées, il est possible de faire des prévisions.

2

Test de détection de la tendance et de la saisonnalité

Sommaire

Les tests statistiques représentent une procédure usuelle pour évaluer l'acceptabilité d'une hypothèse en utilisant les données d'échantillons aléatoires données. Chaque test statistique est associé à un niveau de significativité qui représente une règle permettant de décider d'accepter ou de rejeter l'hypothèse initialement prononcée.

2.1 Test de détection de la saisonnalité

L'étude de la saisonnalité est un préalable au traitement d'une série temporelle. En effet, lorsque cette composante existe, il convient de l'isoler afin de pouvoir analyser les autres composantes. Il est nécessaire de corriger les séries de type infra - annuelle des variations saisonnières avant de les étudier. Deux tests sont utilisés afin d'étudier la variabilité entre périodes : un test paramétrique, l'analyse de la variance (ANOVA), si les données sont normalement distribuées et un test non paramétrique, le test de Kruskal - Wallis, si les données ne sont pas normalement distribuées.

2.1.1 Analyse de la variance et test de Fisher

Le test de Fisher à partir de l'analyse de la variance suppose la chronique sans tendance ou encore sans extra-saisonnalité (*cf.* Laloire (1972)). Dans le cas contraire cette composante est éliminée par une régression sur le temps (extra-saisonnalité déterministe), ou par une procédure de filtrage (extra-saisonnalité aléatoire).

Soit :

- N le nombre d'années.
- p lenombre d'observations (la périodicité) dans l'année (trimestre $p = 4$, mois $p = 12$, etc.).
- X_{ij} la valeur de chronique pour la i^{ime} année ($i = 1, \cdots, N$) et la j^{ime} période ($j = 1, \cdots, p$) supposée telle que $X_{ij} = Y_{ij} + e_{ij}$; les e_{ij} sont les résidus considérées comme aléatoires formés d'éléments indépendants :$e_{ij} \sim \mathcal{N}(0; \sigma^2)$.

Les Y_{ij} sont les éléments d'une composante de la chronique qui s'écrivent : $Y_{ij} = \alpha_i + \beta_j$ où α_i mesure l'effet année en ligne du tableau et β_j mesure l'effet période en colonne du tableau.

Deux effets absents sont testés contre deux effets significativement présent :

- Si l'effet période est significatif, la série est saisonnière ;
- Si l'effet année est significatif, ceci suggère deux interprétations :
  - La chronique de départ n'a pas été transformée, elle possède alors des paliers horizontaux.
  - La chronique a été transformée, des changements de tendance existent dans la chronique

Le test se déroule comme suit :

2.1.1.1 Calcul de la variance totale du tableau

Soit S_T la somme totale des carrés : $S_T = \sum_{i=1}^{N}\sum_{j=1}^{p}\left(X_{ij} - \overline{X}\right)^2$

avec $\overline{X} = \dfrac{1}{N \times p}\sum_{i=1}^{i=N}\sum_{j=1}^{j=p} X_{ij}$: moyenne générale de la chronique sur les $N \times p$ observations.

On note : $\overline{X}_{i\bullet} = \dfrac{1}{p}\sum_{j=1}^{p} X_{ij}$: moyenne de l'année i et $\overline{X}_{\bullet j} = \dfrac{1}{N}\sum_{i=1}^{N} X_{ij}$: moyenne de la période j.

Comme $X_{ij} = Y_{ij} + e_{ij}$ avec $e_{ij} \sim \mathcal{N}\left(0; \sigma^2\right)$ et $Y_{ij} =$ effet période + effet année, nous obtenons :

$$
\begin{aligned}
S_T &= \sum_{i=1}^{i=N}\sum_{j=1}^{j=p}\left(X_{ij} - \overline{X}\right)^2 = \sum_{i=1}^{i=N}\sum_{j=1}^{j=p}\left[\left(X_{ij} - \overline{X}_{i\bullet} + \overline{X}_{i\bullet} - \overline{X}_{\bullet j} + \overline{X}_{j} - \overline{X} - \overline{X} + \overline{X}\right)\right]^2 \\
&= \sum_{i=1}^{i=N}\sum_{j=1}^{j=p}\left[\left(X_{ij} - \overline{X}_{i\bullet} - \overline{X}_{\bullet j} + \overline{X}\right) + \left(\overline{X}_{i\bullet} - \overline{X}\right) + \left(\overline{X}_{\bullet j} - \overline{X}\right)\right]^2 \\
&= \sum_{i=1}^{i=N}\sum_{j=1}^{j=p}\left(X_{ij} - \overline{X}_{i\bullet} - \overline{X}_{\bullet j} + \overline{X}\right)^2 + \sum_{i=1}^{i=N}\sum_{j=1}^{j=p}\left(\overline{X}_{i\bullet} - \overline{X}\right)^2 + \sum_{i=1}^{i=N}\sum_{j=1}^{j=p}\left(\overline{X}_{\bullet j} - \overline{X}\right)^2 \\
&\quad + (\text{terme rectangle nul}) \\
&= \sum_{i=1}^{i=N}\sum_{j=1}^{j=p}\left(X_{ij} - \overline{X}_{i\bullet} - \overline{X}_{\bullet j} + \overline{X}\right)^2 + p\sum_{i=1}^{i=N}\left(\overline{X}_{i\bullet} - \overline{X}\right)^2 + N\sum_{j=1}^{j=p}\left(\overline{X}_{\bullet j} - \overline{X}\right)^2 \\
&= \underset{(\text{résidus})}{S_R} + \underset{(\text{année})}{S_A} + \underset{(\text{période})}{S_P}
\end{aligned}
$$

Le tableau (2.1) présente les calculs intermédiaires avec les notations précédentes :

Table 2.1 – Calculs des moyennes par année et période

Périodes / Années	1	...	j	...	p	Moyennes années
1	X_{11}		X_{1j}		X_{1p}	
$\vdots$	$\vdots$		$\vdots$		$\vdots$	
i	X_{i1}		X_{ij}		X_{ip}	$\overline{X}_{i\bullet} = \dfrac{1}{p}\sum_{j=1}^{j=p} X_{ij}$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	
N	X_{N1}		X_{Nj}		X_{Np}	
Moyennes périodes			$\overline{X}_{\bullet j} = \dfrac{1}{N}\sum_{i=1}^{i=N} X_{ij}$			$\overline{X} = \dfrac{1}{N \times p}\sum_{j=1}^{j=p}\sum_{i=1}^{i=N} X_{ij}$

Nous utilisons ces résultats pour effectuer l'analyse de la variance de la série (tableau (2.2))

Table 2.2 – Analyse de la variance pour détecter une saisonnalité

Désignation	Somme des carrés	Degré de liberté	Variance
Variance Période	$S_P = N\sum_{j=1}^{j=p}\left(\overline{X}_{\bullet j} - \overline{X}\right)^2$	$p-1$	$V_P = \frac{S_P}{p-1}$
Variance Année	$S_A = p\sum_{i=1}^{i=N}\left(\overline{X}_{i\bullet} - \overline{X}\right)^2$	$N-1$	$V_A = \frac{S_A}{N-1}$
Variance Résidu	$\sum_{i=1}^{i=N}\sum_{j=1}^{j=p}\left(X_{ij} - \overline{X}_{i\bullet} - \overline{X}_{\bullet j} + \overline{X}\right)^2$	$(p-1)\times(N-1)$	$V_R = \frac{S_R}{(p-1)\times(N-1)}$
Variance Totale	$S_T = S_P + S_A + S_R$	$N\times p - 1$	$V_T = \frac{S_T}{N\times p} - 1$

2.1.1.2 Test d'influence du facteur colonne - période

L'hypothèse nulle est celle d'absence d'influence. On calcule la statistique de Fisher empirique $F_c = \frac{V_p}{V_R}$ que l'on compare au Fisher lu dans la table $F^{\alpha}_{d_1;d_2}$ à $d_1 = p-1$ et $d_2 = (N-1)(p-1)$ degrés de liberté. Si le Fisher empirique est supérieur au Fisher lu dans la table, on rejette l'hypothèse H_0, la série est donc saisonnière.

Exemple 2.1 *Application du test anova et test de Fisher sur la série Air passengers*

On demande d'effectuer le test de détection de saisonnalité à partir des données du tableau (2.3).

```
# Chargement des données
import seaborn as sns
flights = sns.load_dataset("flights")
# Transformation en série infra-annuelle
data = flights.pivot(index="year", columns="month", values="passengers")
```

Table 2.3 – Données Air passengers (infra - annuelle)

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1949	112	118	132	129	121	135	148	148	136	119	104	118
1950	115	126	141	135	125	149	170	170	158	133	114	140
1951	145	150	178	163	172	178	199	199	184	162	146	166
1952	171	180	193	181	183	218	230	242	209	191	172	194
1953	196	196	236	235	229	243	264	272	237	211	180	201
1954	204	188	235	227	234	264	302	293	259	229	203	229
1955	242	233	267	269	270	315	364	347	312	274	237	278
1956	284	277	317	313	318	374	413	405	355	306	271	306
1957	315	301	356	348	355	422	465	467	404	347	305	336
1958	340	318	362	348	363	435	491	505	404	359	310	337
1959	360	342	406	396	420	472	548	559	463	407	362	405
1960	417	391	419	461	472	535	622	606	508	461	390	432

```
# Moyennes par périodes
x_bar_j = data.mean(axis = 0)
```

Table 2.4 – Moyennes par période

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
moyenne par périodes	241.75	235	270.167	267.083	271.833	311.667	351.333	351.083	302.417	266.583	232.833	261.833

```
# Moyennes par année
x_bar_i = data.mean(axis = 1)
```

Table 2.5 – Moyennes par année

	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
moyenne par année	126.67	139.67	170.17	197	225	238.92	284	328.25	368.42	381	428.33	476.17

```
# Moyenne générale
x_bar = data.mean().mean()
print(f"Moyenne générale : {x_bar}")


## Moyenne générale : 280.29861111111114

## Calcul des variances
N, p = data.shape
# Variance Année
SA = p*np.sum([(x - x_bar)**2 for x in np.array(x_bar_i)])
# Variance Période
SP = N*np.sum([(x - x_bar)**2 for x in np.array(x_bar_j)])
# Variance Résidu
SR = (
  data.sub(list(x_bar_j), axis ="columns")  # soustraction par les colonnes
  .sub(list(x_bar_i), axis = "index")       # Soustraction par les lignes
  .add(x_bar).pow(2).sum().sum()            # Application de la double somme
)
# Variance totale
ST = SP+SA+SR
# Affichage
anova_df = pd.DataFrame({
  "Désignation" : ["Variance Période","Variance Année",
                   "Variance Résidu", "Variance Totale"],
  "Somme des carrés" : [SP, SA, SR, ST],
  "Degré de liberté" : [(p-1),(N-1),(p-1)*(N-1),(N*p - 1)],
  "Variance": [SP/(p-1), SA/(N-1),SR/((p-1)*(N-1)),ST/(N*p - 1)]
})
```

Table 2.6 – Tableau Anova

Désignation	Somme des carrés	Degré de liberté	Variance
Variance Période	218382.24	11	19852.9312
Variance Année	1772584.74	11	161144.0676
Variance Résidu	67077.17	121	554.3568
Variance Totale	2058044.16	143	14391.9172

On vérifie que la variance totale est identique en utilisant la formule générale :

```
# Variance totale
ST = ((data - x_bar)**2).sum().sum()
print(ST)
```

```
## 2058044.1597222222
```

On retombe sur la même valeur. On effectue le test de Fisher

```
ddl1, ddl2 = p-1, (p-1)*(N-1)              # ddl numérateur, ddl dénominateur
fstat = (SP/ddl1)/(SR/ddl2)                # Statistique de Fisher
import scipy.stats as st
flue =  st.f.ppf(0.95,ddl1,ddl2)           # Fisher lue
print("""Fisher empirique : %.2f
Fisher lu à %d et %d degrés de liberté : %.2f""" %(fstat,ddl1,ddl2,flue))
```

```
## Fisher empirique : 35.81
## Fisher lu à 11 et 121 degrés de liberté : 1.87
```

La Fisher empirique est supérieure au Fisher lu dans la table. La série est donc saisonnière.

2.1.2 Test de Kruskal - Wallis

De même que le test d'analyse de la variance, le test de Kruskal - Wallis permet également de déterminer si les données d'une période sont significativement différentes d'une autre. Il est appliqué pour les chroniques dont les données ont une distribution non normale.

Définition 2.1 *Test de Kruskal - Wallis*

L'hypothèse nulle du test de Kruskal - Wallis est que les périodes ne sont pas différentes les unes des autres. La statistique calculée est définie comme suit :

$$K = (n-1)\frac{\sum_{i=1}^{g} n_i \left(\overline{r}_i - \overline{r}\right)^2}{\sum_{i=1}^{g}\sum_{j=1}^{n_i} \left(r_{ij} - \overline{r}\right)^2} \tag{2.1}$$

où g est le nombre de période, n_i est le nombre d'observation dans la période i, r_{ij} est le rang de l'observation j dans le groupe i, n est le nombre total de données, $\overline{r}_i = \dfrac{\sum_{j=1}^{n_i} r_{ij}}{n_i}$ et $\overline{r} = \dfrac{1}{2\left(n+1\right)}$. La statistique est ensuite comparée aux quantiles d'une loi de chi 2 à $(g-1)$ degrés de liberté.

Exemple 2.2 *Application du test de Kruskal - Wallis sur la série Air passengers*

On demande d'effectuer le test de détection de saisonnalité de Kruskal - Wallis à partir des données Air Passengers.

```
# Test de saisonnalité de Kruskal - Wallis
from scipy.stats import kruskal
def kw_seasonalityTest(series, period):
  idx = np.arange(len(series.index)) % period
  kw_stat, kw_pvalue = kruskal(series, idx)
  df = pd.DataFrame({"statistic" : kw_stat, "p value" : kw_pvalue},
                    index = ["KW test"])
  return df
# Application
kw = kw_seasonalityTest(flights.passengers, 12)
```

Table 2.7 – Test de saisonnalité de Kruskal - Wallis

	statistic	p value
KW test	215.4398	0

On a une pvalue qui est nulle, par conséquent on rejette l'hypothèse nulle d'absence de saisonnalité.

2.2 Test de détection de la tendance

Si une série chronologique ne possède pas de saisonnalité ou a été désaisonnalisée, il est indispensable de tester la présence ou non d'une tendance. Dans la littérature économétrique, plusieurs tests permettent de détecter l'existence éventuelle d'une composante tendancielle : ANOVA, test de Mann - Kendall et celui de Mann - Kendall modifié.

2.2.1 Test d'analyse de la variance

L'analyse de la variance présentée dans le cadre du test de saisonnalité peut être adaptée à celui de la tendance. En effet, ici on teste l'influence du facteur ligne avec l'hypothèse nulle celle d'absence d'influence du facteur année.

La statistique de Fisher empirique est $F_c = \dfrac{V_A}{V_R}$ que l'on compare au Fisher lu dans la table $F^{\alpha}_{d_3,d_2}$ à $d_3 = N-1$ et $d_2 = (N-1)(p-1)$ degrés de liberté. Si la Fisher empirique est supérieure au Fisher lu, on rejette l'hypothèse H_0, la série est donc affectée d'une tendance.

Exemple 2.3 *Application du test de l'influence du facteur année sur la série Air passengers*

On souhaite tester la présence d'une tendance sur la série Air passengers à partir du test de l'influence du facteur ligne.

```
ddl3, ddl2 = N-1, (p-1)*(N-1)              # ddl numérateur, ddl dénominateur
fstat2 = (SA/ddl3)/(SR/ddl2)               # Statistique de Fisher
flue2 = st.f.ppf(0.95,ddl3,ddl2)           # Fisher Lue
print("""Fisher empirique : %.2f
Fisher lu à %d et %d degrés de liberté : %.2f""" %(fstat2,ddl3,ddl2,flue2))
```

```
## Fisher empirique : 290.69
## Fisher lu à 11 et 121 degrés de liberté : 1.87
```

L'hypothèse H_0 est rejetée, la chronique est affectée d'une tendance.

2.2.2 Test de Mann - Kendall

Le test de Mann - Kendall (*cf.* Mann (1945), Kendall (1938)) est un test non paramétrique qui permet de détecter des tendances non nécessairement linéaires.

Définition 2.2 *Test de tendance de Mann - Kendall*

L'hypothèse nulle testée est l'absence de la tendance et la statistique calculée est définit comme suit :

$$S = \sum_{i=1}^{i=n-1} \sum_{j=i+1}^{j=n} \text{sgn}\left[(y_j - y_i)(x_j - x_i)\right] \tag{2.2}$$

où la fonction `sign` est définie par : $\text{sign}(X) = 1$ pour $X > 0$, $\text{sign}(X) = 0$ pour $X = 0$ et $\text{sign}(X) = -1$ pour $X < 0$. Mann et Kendall ont démontré que $\mathbb{E}(X) = 0$ et $V(X) = n(n-1)(2n+5)/18$.

Dès que l'échantillon contient une dizaine de données, la loi de la statistique de test Z ci - dessous peut être approchée par une loi normale centrée - réduite :

$$Z = \begin{cases} \dfrac{S-1}{\sqrt{V(S)}} & \text{si} \quad S > 0 \\ 0 & \text{si} \quad S = 0 \\ \dfrac{S+1}{\sqrt{V(S)}} & \text{si} \quad S < 0 \end{cases} \tag{2.3}$$

Remarque 2.1

S'il y a des ex-aequo dans la série, la variance de S est corrigée de la façon suivante :

$$V(S) = \left[n(n-1)(2n+5) - \sum_{p=1}^{g} t_p(p-1)(2p+5)\right] /18 \tag{2.4}$$

où t_p est le nombre d'égalités impliquant g valeurs.

La tendance est dite significative d'un point de vue statistique lorsque la p-value du test est inférieure à 5%.

Exemple 2.4 *Application du test de tendance de Mann - Kendall sur la série Air passengers*

On souhaite vérifier si la série Air passengers possède une tendance en effectuant le test non paramétrique de Mann - Kendall. Nous utilisons la fonction original_test du package pymann-kendall.

```
# Test de tendance de Mann - Kendall
import pymannkendall as mk
_,_,p,z,_,_,_,_,_ = mk.original_test(flights.passengers,0.05)
print("Mann - Kendall statistic : %.2f et p-value : %.2f"%(z,p))
```

```
## Mann - Kendall statistic : 14.38 et p-value : 0.00
```

La p-value est nulle, on rejette l'hypothèse nulle d'absence de tendance.

2.2.3 Test de Mann - Kendall modifié

Ce test ne peut être appliqué que si la chronique dispose d'au moins 40 observations. Il permet de prendre en compte l'autocorrélation des données dans la série chronologique. Le principe repose sur une modification du test de Mann - Kendall plutôt que de modifier les données elles - mêmes.

La modification du test correspond au fait qu'un échantillon autocorrélé positivement de taille n se comporte comme un échantillon indépendant de taille $n^* < n$ (et inversement pour un échantillon autocorrélé négativement). Plusieurs méthodes de calcul de γ sont relevées dans la littérature. La formule proposée par Hamed and Rao (1998) se base sur une formule empirique spécifiquement calculée pour corriger la statistique de Mann - Kendall.

Définition 2.3 *Test de Mann - Kendall modifié*

Cette formule prend en compte les autocorrélations des résidus de régression calculées aux différents rangs si celles - ci sont significatives :

$$\gamma = 1 + \frac{2}{n(n-1)(n-2)} \sum_{k=1}^{k=n-1} (n-k)(n-k-1)(n-k-2)\rho_k \tag{2.5}$$

où n est le nombre de données et ρ_k est l'autocorrélation à l'ordre k, si elle est significative, $\rho_k = 0$ sinon.

Le seuil de significativité choisi pour l'autocorrélation est 1%. On a :

$$V_\rho(S) = \gamma V_{\rho=0}(S) \tag{2.6}$$

où γ est un facteur correctif appliqué à la variance. Ainsi, seule la p-value du test de Mann - Kendall est modifiée.

Exemple 2.5 *Application du test de Mann - Kendall modifié sur la série Air passengers*

On souhaiter vérifier la présence d'une tendance sur la série Air passengers en effectuant le test de Mann - Kendall modifié.

```
# Test de Mann - Kendall modifié
_,_,p,z,_,_,_,_,_ = mk.hamed_rao_modification_test(flights.passengers,0.05,36)
print("Modified Mann - Kendall statistic : %.2f et p-value : %.2f"%(z,p))
```

```
## Modified Mann - Kendall statistic : 10.24 et p-value : 0.00
```

La p-value est nulle, on rejette l'hypothèse nulle d'absence de tendance.

2.2.4 Test de Mann - Kendall saisonnier

Ce test est proposé par Hirsch, Slack, and Smith (1982) et permet d'estimer des tendances de séries cycliques saisonnières.

Le principe est identique à celui du test de Mann - Kendall mais le caractère saisonnier de la série est pris en compte. Autrement dit, pour des données mensuelles ayant une saisonnalité de 12 mois, on ne va pas chercher à savoir s'il y a une croissance au global sur la série, mais simplement d'un mois de janvier à l'autre, d'un mois de février à l'autre et ainsi de suite, il y a une tendance.

Définition 2.4 *Test de Mann - Kendall saisonnier*

La statistique S_k de Kendall se calcule à partir de la somme des statistiques pour chaque saison :

$$S_k = \sum_{i=1}^{s} S_i \tag{2.7}$$

où s est le nombre de saison et S_i la statistique S de Mann - Kendall :

$$\begin{cases} S_i & = \displaystyle\sum_{k=1}^{k=n-1} \sum_{j=k+1}^{j=n} \operatorname{sgn}\left[(y_{ji} - y_{ki})(x_{ji} - x_{ki})\right] \\ \sigma^2_{S_k} & = \dfrac{1}{18} \displaystyle\sum_{i=1}^{i=n} n_i(n_i - 1)(2n_i + 5) \end{cases} \tag{2.8}$$

où n_i est le nombre de donnée pour la saison i. La statistique calculée est

$$Z_{S_k} = \frac{S_k}{\sigma_{S_k}} \tag{2.9}$$

Si le produit du nombre de saison par le nombre d'année est supérieur à 25, la distribution des S_k peut être approximéepar une distribution normale.

L'hypothèse nulle est à rejeter à un niveau de significativité α si $|Z_{S_k}| > Z_{\alpha/2}$ où $Z_{\alpha/2}$ est la valeur de ma distribution normale avec une probabilité de dépassement de $\alpha/2$.

Exemple 2.6 *Application du test de Mann - Kendall saisonnier sur la série Air passengers*

On souhaite effectuer le test de Mann - Kendall saisonnier sur les données Air passengers.

```
# Test de Kendall saisonnier
_,_,p,z,_,_,_,_,_  = mk.seasonal_test(flights.passengers,12,0.05)
print("Saisonality Mann - Kendall statistic : %.2f et p-value : %.2f"%(z,p))
```

```
## Saisonality Mann - Kendall statistic : 15.51 et p-value : 0.00
```

La p-value est nulle, on rejette l'hypothèse nulle d'absence de tendance.

3 Analyse de la tendance et de la saisonnalité

Sommaire

Le problème posé est le suivant : Peut - on trouver une fonction simple du temps qui modélise au mieux la tendance de la série $(X_i)_{i=1,\dots,T}$? On distingue deux types de méthodes :

1. **Les méthodes dites non - paramétriques**

   Ces méthodes supposent que la tendance $(Z_i)_{i=1,\dots,T}$ est de la forme $(Z(t_i))_{i=1,\dots,T}$ où Z est un paramètre fonctionnel, donc de dimension infinie. Parmi ces méthodes, on trouve le lissage par moyennes mobiles.
2. **Les méthodes dites paramétriques**

   Ici, on suppose que la tendance $(Z_i)_{i=1,\dots,T}$ est de la forme $(Z_\theta(t_i))_{i=1,\dots,T}$ où θ est un paramètre inconnu de dimension finie, qu'on estimera à l'aide des observations. Il nous faut alors :

   — choisir une famille de fonctions $\{Z_\theta\}$ dans une collection donnée de fonctions paramétriques, et en général, c'est l'**analyse graphique** de la chronique qui détermine la famille de fonctions paramétriques à considérer.
   — une fois la famille choisie, déterminer la valeur de θ qui conduit au **meilleur ajustement** de la série $(X_i)_{i=1,\dots,T}$ et en général, on choisit θ à l'aide du critère des moindres carrés, ie. on cherche la valeur de θ qui rend minimale

$$\sum_{i=1}^{i=T} (X_i - Z_\theta(t_i))^2 \tag{3.1}$$

3.1 Analyse de la tendance

3.1.1 Ajustement linéaire

Soient X et Y deux variables quantitatives définies sur une même population Ω. On considère le couple de variables (X, Y) dont les modalités sont les couples (X_i, y_i) où $x_i = X(\omega_i)$ et $y_i = Y(\omega_i)$ pour l'individu i.

Si les observations de deux variables statistiques X et Y sont connues individuellement, on commence par les visualiser en les représentant sous la forme d'un nuage de points :

Dans ce repère cartésien, chaque observation (x_i, y_i) est figurée par le point $M_i(x_i, y_i)$ et la forme du nuage donne une information sur le type d'une éventuelle liaison.

Supposons que l'examen du nuage de points conduise à rechercher une droite d'ajustement. Le calcul des coefficients de cette droite va être exposé dans le cas où les observations sont connues individuellement.

Définition 3.1 *Moyenne et écart - type*

Soit X une variable statistique, on a les formules suivantes :

— Cas de données individuelles :

$$\overline{X} = \frac{1}{N}\sum_{i=1}^{i=N} X_i \quad \text{et} \quad \sigma_X = \sqrt{\frac{1}{N}\sum_{i=1}^{i=N}\left(X_i - \overline{X}\right)^2} \tag{3.2}$$

— Cas de données groupées dans un tableau de contingence (covariance pondérée) :

$$\overline{X} = \sum_{i=1}^{i=N} f_i X_i \quad \text{et} \quad \sigma_X = \sqrt{\sum_{i=1}^{i=N} f_i\left(X_i - \overline{X}\right)^2} \tag{3.3}$$

où f_i est la fréquence d'apparition de X_i avec $\sum_{i=1}^{i=N} f_i = 1$.

Sous Python, les fonctions np.mean() et np.std() permettent le calcul de la moyenne et de l'écart type respectivement.

Exemple 3.1 *Moyenne et écart - type des notes de statistique descriptive*

On a relevé les notes de 16 élèves d'une classe après un examen de statistique descriptive. Elles sont contenues dans le tableau 3.1. On demande de calculer la note moyenne et l'écart-type associé.

Table 3.1 – Note des élèves en statistique descriptive

N°	Nom	note	N°	Nom	note	N°	Nom	note	N°	Nom	note
n°1	Arnaud	10	n°5	Bernard	9	n°9	Wilfried	12	n°13	Serge	11
n°2	Rodrigue	10	n°6	Lucie	8	n°10	Claire	14	n°14	Natasha	11
n°3	Olivier	9	n°7	Françoise	16	n°11	Sarah	5	n°15	Sandrine	12
n°4	Aicha	10	n°8	Fanta	11	n°12	Romain	10	n°16	Erwan	13

```
# Note de statistique descriptive.
import numpy as np
note = np.array([10, 9, 12, 11, 10, 8, 14 ,11 ,9 ,16 ,5 ,12 ,10 ,11 ,10 ,13])
print(f"La note moyenne est {np.mean(note)} et l'écart-type est {np.std(note)}.")
```

```
## La note moyenne est 10.6875 et l'écart-type est 2.44230296032249.
```

Définition 3.2 *Covariance entre X et Y*

Soient X et Y deux variables statistiques, la covariance entre X et Y est déinie par :

— Cas de données individuelles :

$$\sigma_{XY} = \frac{1}{N}\sum_{i=1}^{i=N}\left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right) = \frac{1}{N}\sum_{i=1}^{i=N} X_i Y_i - \overline{XY} \tag{3.4}$$

— Cas de données groupées dans un tableau de contingence (covariance pondérée) :

$$\sigma_{XY} = \sum_{i,j=1}^{i,j=k,l} f_{ij}\left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right) = \sum_{i,j=1}^{i,j=k,l} f_{ij} X_i Y_j - \overline{X} \times \overline{Y} \tag{3.5}$$

Sous Python, la fonction np.cov() retourne la matrice de variance covariance entre 2 ou plusieurs variables statistiques : les éléments sur la diagonale principale sont les variances tandis que les éléments hors diagonaux les covariances.

$$\Sigma_{XY} = \begin{pmatrix} \sigma_X^2 & \sigma_{XY} \\ \sigma_{YX} & \sigma_Y^2 \end{pmatrix} \tag{3.6}$$

Exemple 3.2 *Covariance entre les notes de statistique descriptive et les notes de théorie d'estimation*

Pour les mêmes élèves du tableau 3.1, on a également relevé les notes obtenues à l'examen de théorie d'estimation. Ces notes sont contenues dans le tableau 3.2. On demande de calculer la covariance entre les notes obtenues à l'examen de statistique descriptive et celles obtenues à l'examen de théorie de l'examen.

Table 3.2 – Note des élèves en théorie d'estimation

N°	Nom	note	N°	Nom	note	N°	Nom	note	N°	Nom	note
n°1	Arnaud	13	n°5	Bernard	17	n°9	Wilfried	0	n°13	Serge	0
n°2	Rodrigue	2	n°6	Lucie	19	n°10	Claire	17	n°14	Natasha	14
n°3	Olivier	2	n°7	Françoise	10	n°11	Sarah	15	n°15	Sandrine	0
n°4	Aicha	6	n°8	Fanta	1	n°12	Romain	9	n°16	Erwan	15

```
# Note de théorie d'estimation
note2 = np.array([13, 2, 2, 6, 17, 19, 10, 1, 0, 17, 15, 9, 0, 14, 0, 15])
# Covariance entre note et note2
cov = np.cov(note,note2)
print(f"La covariance est {cov[0,1]}.")
```

```
## La covariance est 1.25.
```

Propriété 3.1 *Propriétés de la covariance*

1. $\mathrm{Cov}(X,Y) = \mathrm{Cov}(Y,X)$
2. $\mathrm{Cov}(X,X) = \mathrm{Var}(X)$;
3. $\forall \alpha, \beta \in \mathbb{R}$, on $\mathrm{Cov}(\alpha X + \beta Y) = \alpha^2 \mathrm{Var}(X) + \beta^2 \mathrm{Var}(Y) + 2\alpha\beta \mathrm{Cov}(X,Y)$;
4. $\forall \alpha, \beta, b \in \mathbb{R}$, on a $\mathrm{Cov}(\alpha X + a, \beta Y + b) = \alpha\beta \mathrm{Cov}(X,Y)$;
5. $|\mathrm{Cov}(X,Y)| \leq \sqrt{\mathrm{Var}(X)\mathrm{Var}(Y)}$.

Remarque 3.1

En probabilités et en statistiques, la corrélation entre plusieurs variables aléatoires ou statistiques est une notion de liaison qui contredit leur indépendance.

3.1.2 Rappels sur la régression linéaire

Lorsqu'une liaison linéaire forte entre deux variables X et Y semble raisonnable au vue du nuage de points, on a alors une relation du type :

$$Y = aX + b + \varepsilon \tag{3.7}$$

où les coefficients a et b sont des inconnus. Le coefficient a est appelé la pente de la droite ou encore coefficient directeur de la droite car il détermine la direction (la pente) de la droite. Lorsque a est positif la droite est croissante, lorsque a est négatif la droite est décroissante. Le nombre b s'appelle l'ordonnée à l'origine car c'est l'ordonnée du point de la droite d'abscisse 0 (intersection de la droite avec l'axe des ordonnées). ε est le terme d'erreur.

Le problème est donc d'estimer ces coefficients grâce aux valeurs observées sur l'échantillon. Si les points du nuage sont parfaitement alignés (sur une même droite), il serait facile de donner des valeurs à a et b : il suffirait en effet de prendre pour a la tente de la droite sur laquelle se trouvent les points du nuage et pour b la valeur en $x = 0$ (la solution se trouve en résolvant un système de deux équations à deux inconnues à partir de deux points du nuage). Le problème est que les points du nuage sont rarement (parfaitement) alignés : ils sont « proches » d'une droite.

Nous cherchons donc une droite qui passe au plus près des points du nuage. Pour cela, il faut donc mesurer l'éloignement des points du nuage par rapportà une droite Δ d'équation $y = ax + b + \varepsilon$ puis minimiser au critère d'erreur donné. On peut envisager de minimiser :

— La somme des erreurs en valeur absolue : $\min\limits_{a,b} \sum\limits_{t=1}^{t=T} |\varepsilon_t| = \min\limits_{a,b} \sum\limits_{t=1}^{t=T} |y_t - ax_t - b|$

— La somme des erreurs au carré : $\min\limits_{a,b} \sum\limits_{t=1}^{t=T} \varepsilon_t^2 = \min\limits_{a,b} \sum\limits_{t=1}^{t=T} (y_t - ax_t - b)^2$

3.1.2.1 La méthode des moindres carrés ordinaires

La méthode des moindres carrés ordinaires consiste à minimiser la somme des carrés des erreurs. Soit le programme :

$$\min_{a,b} \sum_{t=1}^{t=T} \varepsilon_t^2 = \min_{a,b} \sum_{t=1}^{t=T} (y_t - ax_t - b)^2 = \mathcal{Q}(a,b) \tag{3.8}$$

Ce programme admet son minimum lorsque les dérivées partielles premières par rapport à a et b sont nulles, c'est - à - dire :

$$\begin{cases} \dfrac{\partial}{\partial \hat{a}} \mathcal{Q}(\hat{a},\hat{b}) &= 0 \\ \dfrac{\partial}{\partial \hat{b}} \mathcal{Q}(\hat{a},\hat{b}) &= 0 \end{cases} \Longrightarrow \begin{cases} -2\sum\limits_{t=1}^{t=T} x_t \left(y_t - \hat{a}x_t - \hat{b}\right) &= 0 \\ -2\sum\limits_{t=1}^{t=T} \left(y_t - \hat{a}x_t - \hat{b}\right) &= 0 \end{cases}$$

En sommant par rapport à t, il vient :

$$\begin{cases} \sum\limits_{t=1}^{t=T} x_t y_t - \hat{a}\sum\limits_{t=1}^{t=T} x_t^2 - \hat{b}\sum\limits_{t=1}^{t=T} x_t &= 0 \\ \sum\limits_{t=1}^{t=T} y_t - \hat{a}\sum\limits_{t=1}^{t=T} x_t - \sum\limits_{t=1}^{t=T} \hat{b} &= 0 \end{cases}$$

On obtient les estimateurs $\hat{a}$ et $\hat{b}$ suivants :

$$\begin{cases} \hat{a} & = \dfrac{\dfrac{1}{T}\sum_{t=1}^{t=t} x_t y_t - \overline{x} \times \overline{y}}{\dfrac{1}{T}\sum_{t=1}^{t=T} x_t^2 - \overline{x}^2} = \dfrac{\mathrm{Cov}(x,y)}{\mathrm{Var}(x)} \\ \hat{b} & = \overline{y} - \hat{a}\overline{x} \end{cases} \tag{3.9}$$

Exemple 3.3

On souhaite regresser y sur x où y représente les notes obtenues à l'examen de théorie de l'estimation et x les notes obtenues à l'exament de statistique descriptive. On considère le modèle suivant :

$$y_i = \alpha_0 + \alpha_1 x_i + \varepsilon_i, \quad \forall i = 1, \dots, 16 \tag{3.10}$$

Sous fonction seaborn.regplot() retourne un nuage des points avec la droite ajustée.

```
# Ajustement entre note et note2
import seaborn as sns
note_df = pd.DataFrame({"y":note2,"x":note},index = [x for x in range(16)])
fig, axe = plt.subplots(figsize=(16,6));
sns.regplot(x = "x", y = "y",data = note_df,fit_reg=True,ax = axe);
plt.show()
```

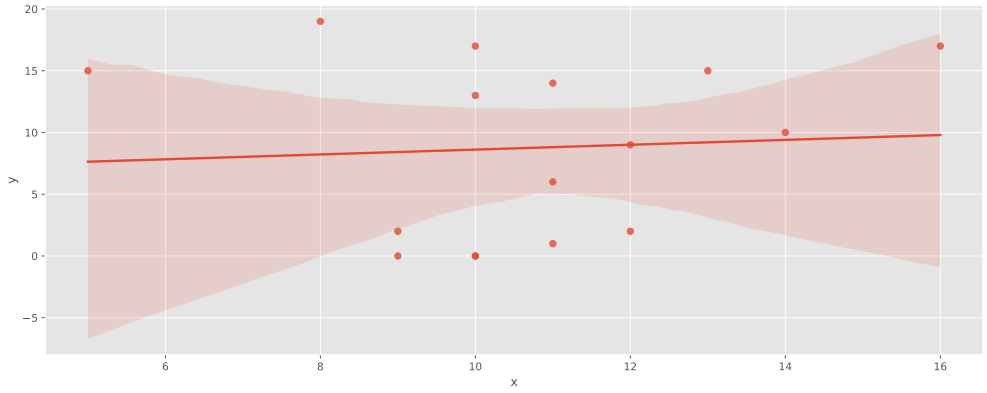


Figure 3.1 – Ajustement linéaire entre les notes de théorie de l'estimation et ceux de statistique descriptive.

La figure 3.1 montre qu'un ajustement linéaire n'est pas approprié pour nos données car la disposition des points dans le nuage montre qu'il n'y a aucune liaison linéaire entre les données.

Propriété 3.2

1. La droite d'équation $y = \hat{a}x + \hat{b}$ est appelée droite de régression de Y sur X et est notée : $\Delta_{Y/X}$.
2. Cette droite passe par le point moyen $M(\overline{x}, \overline{y})$.
3. Le coefficient directeur a de $\Delta_{Y/X}$, $\mathrm{Cov}(x,y)$ et $\rho_{x,y}$ sont du même signe :

— Lorsqu'ils sont positifs, on parle de corrélation positive (y augmente quand x augmente).
— Lorsqu'ils sont négatifs, on parle de corrélation négative (y diminue quand x augmente).

3.1.2.2 La notion de coefficient de corrélation linéaire

Afin de confirmer qu'il est raisonnable d'approximer le nuage de points par une droite, on calcule le coefficient de corrélation linéaire ou coefficient de Bravais - Pearson. C'est un indice statistique qui exprime l'intensité et le sens (positif ou négatif) de la relation linéaire entre deux variables quantitatives. Il assume des valeurs se situant dans l'intervalle qui va de -1 et $+1$. Une valeur égale à -1 ou à $+1$ indique l'existence d'une relation linaire parfaite (fonctionnelle) entre les deux variables. En revanche, ce coefficient est nul ($\rho = 0$) lorsqu'il n'y a pas de relation linéaire entre les variables (ce qui n'exclut pas l'existence d'une relation autre que linéaire : par exemple de forme « quadratique »).

L'intensité de la relation linéaire sera donc d'autant plus forte que la valeur du coefficient est proche de $+1$ ou de -1, et d'autant plus faible qu'elle est proche de 0. Par ailleurs, le coefficient est de signe positif si la relation est positive (directe, croissante) et de signe négatif si la relation est négative (inverse, décroissante).

Définition 3.3 *Coefficient de corrélation linéaire de Pearson*

Le coefficient ρ de Bravais - Pearson entre deux variables X et Y se calcule en appliquant la formule suivante (voir Saporta (2006)) :

$$\rho_{X,Y} = \frac{\text{Cov}(X,Y)}{\sigma_X \sigma_Y} \tag{3.11}$$

Sous Python, la fonction np.corrcoef() retourne la matrice des corrélations entre 2 ou plusieurs variables statistiques : les éléments sur la diagonale principale sont égaux à 1 tandis que les éléments hors diagonaux les corrélations linéaires de Pearson entre variables prises deux à deux.

$$\rho = \begin{pmatrix} 1 & \rho_{XY} \\ \rho_{YX} & 1 \end{pmatrix} \tag{3.12}$$

Exemple 3.4 *Corrélation linéaire entre les notes de statistique descriptive et les notes de théorie d'estimation*

On souhaite calculer le coefficient de corrélation linéaire entre les notes obtenues à l'examen de statistique descriptive et celles obtenues à l'examen de théorie de l'estimation.

```
# Corrélation linéaire entre note et note2
corr = np.corrcoef(note,note2)
print(f"Le coefficient de corrélation linéaire est {corr[0,1]}.")
```

```
## Le coefficient de corrélation linéaire est 0.06985029892941777.
```

Propriété 3.3 *Propriétés du coefficient de corrélation linéaire*

1. Le coefficient de corrélation linéaire est symétrique :

$$\rho_{X,Y} = \rho_{Y,X} \tag{3.13}$$

2. L'inégalité de Cauchy - Schwarz donne :

$$-1 \leq \rho_{XY} \leq 1 \tag{3.14}$$

3. **Transformation des données**

   Soient α, β, λ et μ quatre nombres réels quelconques avec $\alpha \neq 0$ et $\lambda \neq 0$. Posons $Z = \alpha X + \beta$ et $T = \lambda Y + \mu$. On a alors :

$$\begin{aligned} \rho_{Z,T} &= \frac{\mathrm{Cov}(\alpha X + \beta, \lambda Y + \mu)}{\sigma_{\alpha X+\beta}\sigma_{\lambda X+\mu}} = \frac{\alpha\lambda \mathrm{Cov}(X,Y)}{\alpha\lambda\sigma_X\sigma_Y} \\ &= \begin{cases} +\rho_{X,Y} & \text{si } \alpha \text{ et } \lambda \text{ sont de même signe} \\ -\rho_{X,Y} & \text{si } \alpha \text{ et } \lambda \text{ sont de signe opposé} \end{cases} \end{aligned}$$

   En particulier, pour $\alpha = \lambda = 0$, on voit que le coefficient de corrélation linéaire est invariant par translations et pour $\beta = \mu = 0$, il est invariant au signe près par homothéties.

Remarque 3.2

On peut simplifier les calculs du coefficient de corrélation linéaire :

— Si $\rho_{X,Y} = 0$

  Dans ce cas l'éloignement des points du nuage avec la droite de régression de Y en X est maximal. On dira alors que X et Y sont linéairement indépendants.

— Si $\rho_{X,Y} > 0$

  Dans ce cas la droite de régression de Y en X est croissante ; on parle alors de corrélation linéaire croissante entre X et Y. Lorsque $\rho_{X,Y}$ est proche de 1, les points du nuage sont donc presque alignés, on a donc une forte corrélation linéaire croissante (ou positive) entre X et Y.

  Dans le cas extrême $\rho_{X,Y} = 1$, les points du nuage sont alors parfaitement alignés, on peut donc parler de corrélation linéaire croissante totale : pour un individu, sa donnée suivant X détermine entièrement sa donnée suivant Y.

  Arbitrairement, on considèrera la corrélation linéaire croissante faible lorsque $0 < \rho_{X,Y} < 0.3$, moyenne lorsque $0.3 \leq \rho_{X,Y} \leq 0.7$ et forte lorsque $\rho_{X,Y} > 0.7$.

— Si $\rho_{X,Y} < 0$.

  Dans ce cas la droite de régression de Y en X est décroissante ; on parle alors de corrélation linéaire décroissante entre X et Y. Lorsque $\rho_{X,Y}$ est proche de -1, les points du nuage sont donc presque alignés, on a donc une forte corrélation linéaire décroissante (ou négative) entre X et Y.

  Dans le cas extrême $\rho_{X,Y} = -1$, les points du nuage sont alors parfaitement alignés, on peut donc parler de corrélation linéaire décroissante totale : pour un individu, sa donnée suivant X détermine entièrement sa donnée suivant Y.

  Arbitrairement, on considèrera la corrélation linéaire décroissante faible lorsque $-0.3 < \rho_{X,Y} < 0$, moyenne lorsque $-0.7 \leq \rho_{X,Y} \leq -0.3$ et forte lorsque $\rho_{X,Y} < -0.7$.

3.1.3 Ajustement tendanciel linéaire

Ces méthodes supposent que la série chronologique va s'accroître du même montant à chaque période :

$$\Delta Z_t = Z_t - Z_{t-1} = constante \quad \forall t \tag{3.15}$$

On veut juste ajuster les données par une droite et donc modéliser la tendance par une fonction de la forme :

$$Z_\theta(t) = at + b \quad \text{avec} \quad \theta = (a, b)^{'} \tag{3.16}$$

Deux méthodes sont utilisées : (i) la méthode des moindres carrés ; (ii) La méthode des 2 points.

3.1.3.1 Ajustement tendanciel linéaire par les moindres carrés

Soit X_t la chronique. La méthode des MCO consiste à retenir la tendance de la série X_t par une fonction linéaire : $\widehat{Z}_t = Z_{\hat{\theta}}(t) = \hat{a}t + \hat{b}$. Elle correspond à la droite qui passe au plus près des points du nuage. On mesure donc l'éloignement des points du nuage par rapport à la droite d'équation $Z_\theta(t) = at + b$.

En gros, cette méthode consiste à choisir les coefficients a et b de sorte que la quantité

$$\sum_{t=1}^{t=T} \left(X_t - Z_\theta(t)\right)^2 = \sum_{t=1}^{t=T} \left(X_t - (at + b)\right)^2 = m(a, b) \tag{3.17}$$

soit minimale.

En opérant par dérivation par rapport à a et b afin de trouver le minimum [1] de cette fonction, on obtient les résultats suivants :

$$\begin{cases} \dfrac{\partial}{\partial \hat{a}} m(\hat{a}, \hat{b}) & = -2 \displaystyle\sum_{t=1}^{t=T} t \left(X_t - \hat{a}t - \hat{b}\right) = 0 \\ \dfrac{\partial}{\partial \hat{b}} m(\hat{a}, \hat{b}) & = -2 \displaystyle\sum_{t=1}^{t=T} \left(X_t - \hat{a}t - \hat{b}\right) = 0 \end{cases}$$

En sommant par rapport à t, il vient :

$$\begin{cases} \displaystyle\sum_{t=1}^{t=T} tX_t - \hat{a} \sum_{t=1}^{t=T} t^2 - \hat{b} \sum_{t=1}^{t=T} t = 0 \\ \displaystyle\sum_{t=1}^{t=T} X_t - \hat{a} \sum_{t=1}^{t=T} t - n\hat{b} = 0 \end{cases}$$

Les estimateurs $\hat{a}$ et $\hat{b}$ de a et b sont :

1. Nous considérons les conditions du deuxième ordre comme vérifiées car la fonction est convexe.

$$\begin{cases} \hat{a} & = \dfrac{\sum_{t=1}^{t=T}(t-\overline{t})\left(X_t-\overline{X}\right)}{\sum_{t=1}^{t=T}(t-\overline{t})^2} = \dfrac{\sum_{t=1}^{t=T} tX_t - n\overline{t}\,\overline{X}}{\sum_{t=1}^{t=T} t^2 - n\overline{t}^2} = \dfrac{\mathrm{Cov}(t,X_t)}{V(t)} \\ \hat{b} & = \overline{X} - \hat{a}\overline{t} \end{cases} \tag{3.18}$$

où $\overline{t} = \dfrac{1}{T}\sum_{t=1}^{t=T} t$ et $\overline{X} = \dfrac{1}{T}\sum_{t=1}^{t=T} X_t$.

Propriété 3.4

- La droite d'équation $Z_{\hat{\theta}}(t) = \hat{a}t + \hat{b}$ s'appelle **la droite des moindres carrés** (DMC).
- Le point moyen $(\overline{t}, \overline{X})$ appartient à la droite des moindres carrés.
- On peut apprécier la qualité de l'ajustement linéaire à l'aide du **coefficient de corrélation linéaire** noté $\rho_{t,X}$ et défini par :

$$\rho_{t,X} = \frac{\sum_{t=1}^{t=T}(t-\overline{t})\left(X_t-\overline{X}\right)}{\sqrt{\sum_{t=1}^{t=T}(t-\overline{t})^2}\sqrt{\sum_{t=1}^{t=T}\left(X_t-\overline{X}\right)^2}} \tag{3.19}$$

- La variance de la série $(X_t)_{t=1,\dots,T}$ se décompose en :

$$\underbrace{\frac{1}{T}\sum_{t=1}^{t=T}\left(X_t-\overline{X}\right)^2}_{\text{Variance totale de x}} = \underbrace{\frac{1}{T}\sum_{t=1}^{t=T}\left(\hat{a}t+\hat{b}-\overline{X}\right)^2}_{\text{Variance expliquée par la DMC}} + \underbrace{\frac{1}{T}\sum_{t=1}^{t=T}\left(X_t-\left(\hat{a}t+\hat{b}\right)\right)^2}_{\text{Variance résiduelle}}$$

La proportion de variance expliquée par le modèle linéaire est donnée par :

$$\frac{\sum_{t=1}^{t=T}\left(\hat{a}t+\hat{b}-\overline{X}\right)^2}{\sum_{t=1}^{t=T}\left(X_t-\overline{X}\right)^2} = \rho_{t,X}^2$$

Exemple 3.5 *Ajustement de la série Air passengers par moindre carré ordinaire*

On souhaite ajuster la série Air passengers par la méthode des moindres carrés ordinaires.

```
# Chargement des données
import seaborn as sns
import pandas as pd
import statsmodels.formula.api as smf
donnee = sns.load_dataset("flights").drop(columns=["year", "month"])
donnee.index = pd.date_range(start="1949-01-31",end = "1960-12-31",freq="M")
# Estimation de la droite de tendance par MCO
dataframe = pd.DataFrame({
```

```
    "passengers" : donnee.passengers,
    "temps" : np.arange(1,len(donnee.passengers)+1)
})
model = smf.ols("passengers ~ temps", data = dataframe).fit()
print(model.summary2())
```

```
##                  Results: Ordinary least squares
## ================================================================
## Model:              OLS              Adj. R-squared:     0.853
## Dependent Variable: passengers       AIC:                1513.6466
## Date:               2023-08-06 22:39 BIC:                1519.5862
## No. Observations:   144              Log-Likelihood:     -754.82
## Df Model:           1                F-statistic:        828.2
## Df Residuals:       142              Prob (F-statistic): 4.02e-61
## R-squared:          0.854            Scale:              2121.3
## ----------------------------------------------------------------
##                Coef.   Std.Err.     t     P>|t|    [0.025   0.975]
## ----------------------------------------------------------------
## Intercept     87.6528    7.7163  11.3594  0.0000  72.3990  102.9065
## temps          2.6572    0.0923  28.7784  0.0000   2.4747    2.8397
## ----------------------------------------------------------------
## Omnibus:               24.637       Durbin-Watson:          0.537
## Prob(Omnibus):         0.000        Jarque-Bera (JB):       33.905
## Skew:                  0.940        Prob(JB):               0.000
## Kurtosis:              4.454        Condition No.:          168
## ================================================================
## Notes:
## [1] Standard Errors assume that the covariance matrix of the
## errors is correctly specified.
```

Nos coefficients estimés sont significativement différents de 0. De plus, le pouvoir explicatif du modèle, R^2, est très bon (85.4%). La significativité globale du modèle est testée à l'aide de la statistique de Fisher, $F = 828.2$, à 1 et 142 degrés de liberté avec une pvalue qui est nulle. On conclut que le modèle est globalement significatif.

3.1.3.2 Ajustement tendanciel par la méthode de Mayer

La méthode de Mayer, aussi appelée méthode des fractionnements ou des valeurs moyennes, est la simplification poussée à l'extrême de la méthode des moyennes échelonnées :

- On partage l'ensemble des observations en deux groupes ayant le même nombre d'éléments, à une parité près ;
- On substitue aux points figuratifs de chacun de ces groupes un point moyen ayant pour abscisse la moyenne de leurs abscisses et pour ordonnée la moyenne de leurs ordonnées ;
- La droite d'ajustement est celle qui passe par ces deux points moyens.

Soient $G_1 = (\overline{t}_1, \overline{X}_1)$ et $G_2 = (\overline{t}_2, \overline{X}_2)$ les points moyens des groupes 1 et 2. La droite (Δ) : $Z_\theta(t) = at + b$ passe par les points G_1 et G_2 si et seulement si a et b sont solutions du système ci-après :

$$(S)\begin{cases} a\bar{t}_1 + b &= \overline{X}_1 \\ a\bar{t}_2 + b &= \overline{X}_2 \end{cases} \tag{3.20}$$

En résolvant le système (S), on trouve les coefficients de la droite (Δ) donnés par :

$$\begin{cases} a &= \dfrac{\overline{X}_2 - \overline{X}_1}{\bar{t}_2 - \bar{t}_1} \\ b &= \overline{X}_2 - a\bar{t}_2 = \overline{X}_1 - a\bar{t}_1 \end{cases} \tag{3.21}$$

Remarque 3.3

La méthode des points moyens est empirique. Elle n'est basée sur aucun critère à minimiser. Elle peut cependant s'avérer efficace en présence de valeurs aberrantes, ce qui n'est pas le cas de la méthode des moindres carrés.

Exemple 3.6 *Ajustement de la série Air passengers par la méthode des deux points*

On souhaite ajuster la série Air passengers par la méthode des points moyens.

```
# Subdivision du data set en deux groupes 1 et 2
T = len(donnee.passengers)
donnee1 = dataframe[:int(T/2)] # groupe 1
donnee2 = dataframe[int(T/2):] # groupe 2
```

On calcule les moyennes pour chacune des variables sur les 2 groupes.

```
# Calcul des moyennes par groupe
mean1 = donnee1.mean().values.reshape(1,2)
mean2 = donnee2.mean().values.reshape(1,2)
matrix = np.append(mean1, mean2, axis = 0)
datamean = pd.DataFrame(matrix,columns = ["moyenne sur X", "moyenne sur t"],
                        index = ["groupe 1", "groupe 2"])
```

Table 3.3 – Moyenne par groupe

	moyenne sur X	moyenne sur t
groupe 1	182.9028	36.5
groupe 2	377.6944	108.5

Sous Python, la fonction np.linalg.solve() de Numpy permet de résoudre le système (S).

```
# Extraction des vecteurs x et Y
Y = matrix[:,:1]
x = matrix[:,1:2]
# Création de la matrice X
X = np.hstack((x, np.ones(x.shape)))
# Resolution : Xa = Y
coef = pd.DataFrame(np.linalg.solve(X,Y),index = ["slope", "intercept"],
                    columns = ["coefficient"])
```

Table 3.4 – Estimation de la tendance par la méthode de Mayer

	slope	intercept
coefficient	2.70544	84.15422

On peut à présent calculer les valeurs tendancielles à partir des coefficients a et b obtenus et la représenter graphiquement.

```
# Valeurs tendancielles par points moyens
mayertrend =list([(coef.values[0,0]*k+coef.values[1,0]) for k in range(1,T+1)])
fitmayer = pd.DataFrame({"mayer":np.array(mayertrend)},index = dataframe.index)


# Représentation graphique
import matplotlib.pyplot as plt
fig, axe = plt.subplots(figsize=(16,6));
axe.plot(dataframe.passengers,color='black');
axe.plot(model.fittedvalues,color='red',label = "DMC");
axe.plot(fitmayer.mayer,color='blue',linestyle = "--",label = "Mayer");
axe.set_xlabel('Année');
axe.set_ylabel('value');
axe.legend();
plt.show()
```

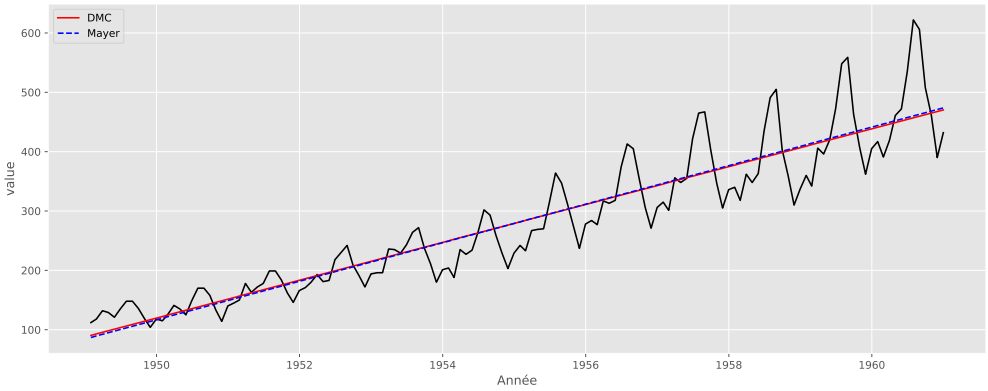


Figure 3.2 – Ajustement par moindre carré ordinaire et par mayer

La droite en rouge représente la tendance estimée par la méthode des moindres carrés ordinaires. Les valeurs des coefficients obtenus par la méthode de Mayer (droite bleue) sont presque similaires à celles obtenues par les MCO, l'ajustement est identique à des constantes près.

3.1.4 Ajustement non linéaire de la tendance

On peut reprocher aux méthodes d'ajustement linéaire de la tendance d'être parfois trop simplificatrice lorsque la tendance n'est pas linéaire, on peut toutefois la linéariser par une technique de changement de variable appropriée lorsque cela est possible.

Exemple 3.7

— Si $Z_\theta(t) = at^2 + b$, en posant $Y(t) = t^2$ on se ramène à : $Z_\theta(t) = aY(t) + b$ et on peut ainsi faire un ajustement linéaire entre $Z_\theta(t)$ et $Y(t)$
— Si $Z_\theta(t) = be^{at}$, en posant $Y_\theta(t) = \ln Z_\theta(t)$ on se ramène à : $Y_\theta(t) = at + \ln b$ et on peut ainsi faire un ajustement linéaire entre $Y_\theta(t)$ et t. Cette tendance exponentielle peut sembler plus réaliste. En effet, au lieu de penser que la série va s'accroître du même montant à chaque période ($\Delta Z_\theta(t) = Z_\theta(t) - Z_\theta(t-1) = constante$), il peut paraître plus juste qu'elle s'accroisse du même pourcentage à chaque période : le taux d'accroissement est alors : $\dfrac{\partial}{\partial t} \ln Z_\theta(t) = constante$

3.1.4.1 Tendance auto-régressive

L'on suppose que la valeur à t dépend de la précédente de sorte que l'on peut écrire :

$$X_t = a + bX_{t-1}, \quad \forall t = 1, \dots, T \tag{3.22}$$

Le comportement de la série diffère selon les valeurs de a et b. Si $a = 0$ et $|b| \neq 1$ alors b est le taux de croissance de la série. L'observation d'une trajectoire du processus sur $t \in \{1, ..., n\}$ nous conduit naturellement à l'estimateur des moindres carrés donné par :

$$\begin{cases} \hat{b} & = \dfrac{\sum_{t=1}^{t=T}(X_t - \overline{X})(X_{t-1} - \overline{X})}{\sum_{t=1}^{t=T}(X_t - \overline{X})^2} \\ \hat{a} & = (1 - \hat{b})\overline{X} \end{cases} \tag{3.23}$$

avec $\overline{X} = \dfrac{1}{T}\sum_{t=1}^{t=T} X_t$ et on considère arbitrairement que $X_0 = 0$ (presque sûrement).

```
# Définition de la fonction params
def params(x,y):
  u = ((x - x.mean())*(y-x.mean())).sum()
  v = ((x - x.mean())**2).sum()
  slope = u/v
  intercept = (1 - slope)*x.mean()
  return intercept,slope
```

Dans cette fonction x représente le processus X_t et y le processus retardé d'une période X_{t-1}. La fonction renvoie les paramètres `intercept` et `slope` qui correspondent respectivement aux estimations des paramètres a et b.

Exemple 3.8 *Ajustement de la série Air passengers par tendance autorégressive d'ordre 1*

Pour illustrer, on crée notre processus retardé d'une période en remplissant la première observation par 0. Ensuite on transforme sous forme de array.

```
# Création du vecteur retardé d'une période
datashift = donnee.shift(periods = 1).fillna(0).values
```

On peut à présent estimer les paramètres de notre tendance.

```
# Estimation des coefficients
coeff = pd.DataFrame(np.array(params(donnee.values, datashift)),
                     columns = ["coefficients"],index = ["intercept", "slope"])
```

Table 3.5 – Tendance autorégressive

	intercept	slope
coefficients	8.137338	0.970969

3.1.4.2 Ajustement polynômial

L'objectif est d'ajuster les données par un **polynôme de degré** p et donc modéliser la tendance par une fonction de la forme :

$$Z_\theta(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_{p-1} t^{p-1} + a_p t^p, \quad \text{avec} \quad \theta = \left(a_0, a_1, \dots, a_p\right)^{'} \tag{3.24}$$

Le cas $p = 0$ correspond à la tendance constante et le cas $p = 1$ à la tendance linéaire.

Trouver la valeur de θ qui minimise :

$$\sum_{t=1}^{t=T} \left(X_t - \left(a_0 + a_1 t + a_2 t^2 + \cdots + a_p t^p\right)\right)^2 \tag{3.25}$$

Autrement dit, calculer l'estimation des moindres carrés θ^{MC} définie par :

$$\theta^{MC} = \arg \min_{\theta \in \mathbb{R}^{p+1}} \sum_{t=1}^{t=T} \left(X_t - \left(a_0 + a_1 t + a_2 t^2 + \cdots + a_p t^p\right)\right)^2 \tag{3.26}$$

Posons :

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_T \end{pmatrix} \quad \text{et} \quad Z = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 2^2 & \cdots & 2^p \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & T & T^2 & \cdots & T^p \end{pmatrix}$$

On montre aisément qu'on a encore :

$$\hat{\theta}^{MC} = \arg \min_{\theta \in \mathbb{R}^{p+1}} ||X - Z\theta||^2 = \arg \min_{\theta \in \mathbb{R}^{p+1}} S(\theta) \tag{3.27}$$

Pour minimiser cette fonction par rapport à θ, nous différencions S par rapport à θ :

$$\frac{\partial}{\partial \theta} S(\theta) = -2Z^{'}X + 2Z^{'}Z\hat{\theta}^{MC} = 0 \rightarrow \hat{\theta}^{MC} = \left(Z^{'}Z\right)^{-1} Z^{'}X \tag{3.28}$$

où $Z^{'}$ désigne la matrice transposée de Z et

$$Z'Z = \begin{pmatrix} T & \sum_{t=1}^{t=T} t & \sum_{t=1}^{t=T} t^2 & \cdots & \sum_{t=1}^{t=T} t^p \\ \sum_{t=1}^{t=T} t & \sum_{t=1}^{t=T} t^2 & \sum_{t=1}^{t=T} t^3 & \cdots & \sum_{t=1}^{t=T} t^{p+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \sum_{t=1}^{t=T} t^p & \sum_{t=1}^{t=T} t^{p+1} & \sum_{t=1}^{t=T} t^{p+2} & \cdots & \sum_{t=1}^{t=T} t^{2p} \end{pmatrix} \quad \text{et} \quad Z'X = \begin{pmatrix} \sum_{t=1}^{t=T} X_t \\ \sum_{t=1}^{t=T} tX_t \\ \vdots \\ \sum_{t=1}^{t=T} t^p X_t \end{pmatrix}$$

En particulier, on peut vérifier que lorsque $p = 0$, alors $Z'Z = T$ et $Z'X = \sum_{t=1}^{t=T} X_t$, et donc $\hat{a}_0 = \overline{X}$.

Dans le cas où $p = 1$, alors

$$Z'Z = \begin{pmatrix} T & \sum_{t=1}^{t=T} t \\ \sum_{t=1}^{t=T} t & \sum_{t=1}^{t=T} t^2 \end{pmatrix} \quad \text{et} \quad Z'X = \begin{pmatrix} \sum_{t=1}^{t=T} X_t \\ \sum_{t=1}^{t=T} tX_t \end{pmatrix}$$

Propriété 3.5 *Choix du degré du polynôme p*

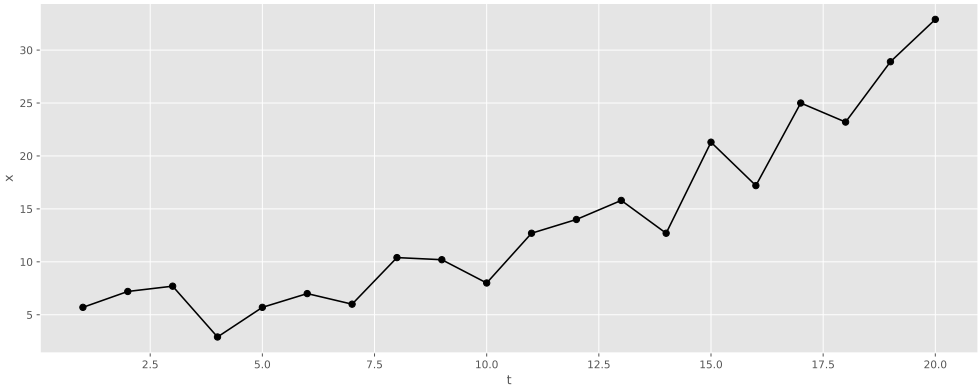
En général, le choix du degré du polynôme p est basé sur l'examen :

- du graphe de la série $(X_t)_{t=1,\ldots,T}$ qui permet de se faire une première idée du degré ;
- des résidus obtenus après l'ajustement. En particulier, on voudra :
  - obtenir des résidus qui fluctuent autour de 0 et de faible amplitude.
  - utiliser un polynôme de degré le plus faible possible.

Exemple 3.9 *Ajustement polynomial*

Considérons la chronique suivante :

```
# Création de la chronique
df = pd.DataFrame({
  "t" : np.arange(1,21),
  "x" : [5.7,7.2,7.7,2.9,5.7,7,6,10.4,10.2,8,
         12.7,14,15.8,12.7,21.3, 17.2, 25, 23.2,28.9, 32.9]})
# Visualisation
fig, axe = plt.subplots(figsize=(16,6));
axe.scatter(df.t, df.x, color = "black");
axe.plot(df.t, df.x, color = "black");
axe.set_xlabel("t");
axe.set_ylabel("x");
plt.show()
```



On veut ajuster cette chronique à l'aide d'un polynôme de degré p avec $p = 0, 1, 2$ et 3. La fonction poly retourne un polynôme de degré p.

```
# fonction poly
def poly(x,p):
  return x**p
```

On ajuste nos 4 modèles :

— Ajustement par un polynôme de degré 0.

```
# Ajustement polynomial degré 0
res0 = smf.ols("x ~ 1", data = df).fit()
print(res0.summary2())
```

```
##                   Results: Ordinary least squares
## =================================================================
## Model:              OLS              Adj. R-squared:     0.000
## Dependent Variable: x                AIC:                143.3935
## Date:               2023-08-06 22:39 BIC:                144.3892
## No. Observations:   20               Log-Likelihood:     -70.697
## Df Model:           0                F-statistic:        nan
## Df Residuals:       19               Prob (F-statistic): nan
## R-squared:          0.000            Scale:              72.464
## -----------------------------------------------------------------
##               Coef.    Std.Err.     t      P>|t|    [0.025    0.975]
## -----------------------------------------------------------------
## Intercept   13.7250     1.9035   7.2105   0.0000   9.7410   17.7090
## -----------------------------------------------------------------
## Omnibus:              3.090        Durbin-Watson:           0.218
## Prob(Omnibus):        0.213        Jarque-Bera (JB):        2.418
## Skew:                 0.830        Prob(JB):                0.298
## Kurtosis:             2.617        Condition No.:           1
## =================================================================
## Notes:
## [1] Standard Errors assume that the covariance matrix of the
## errors is correctly specified.
```

On a :

$$Z_0(t) = \overline{X} = 13.725 \tag{3.29}$$

— Ajustement par un polynôme de degré 1.

```
# Ajustement polynomial degré 1
res1 = smf.ols("x~t", data = df).fit()
print(res1.summary2())
```

```
##                   Results: Ordinary least squares
## ================================================================
## Model:              OLS              Adj. R-squared:     0.828
## Dependent Variable: x                AIC:                109.1276
## Date:               2023-08-06 22:39 BIC:                111.1190
## No. Observations:   20               Log-Likelihood:     -52.564
## Df Model:           1                F-statistic:        92.35
## Df Residuals:       18               Prob (F-statistic): 1.64e-08
## R-squared:          0.837            Scale:              12.477
## ----------------------------------------------------------------
##                 Coef.   Std.Err.     t     P>|t|    [0.025  0.975]
## ----------------------------------------------------------------
## Intercept      -0.0963    1.6408  -0.0587  0.9538  -3.5436  3.3510
## t               1.3163    0.1370   9.6100  0.0000   1.0285  1.6041
## ----------------------------------------------------------------
## Omnibus:              1.010        Durbin-Watson:           1.165
## Prob(Omnibus):        0.603        Jarque-Bera (JB):        0.880
## Skew:                 0.283        Prob(JB):                0.644
## Kurtosis:             2.142        Condition No.:           25
## ================================================================
## Notes:
## [1] Standard Errors assume that the covariance matrix of the
## errors is correctly specified.
```

La droite de tendance linéaire s'écrit :

$$Z_1(t) = -0.0963 + 1.3163t \tag{3.30}$$

— Ajustement par un polynôme de degré 2.

```
# Ajustement polynomial degré 2
res2 = smf.ols("x~t+poly(t,2)",data = df).fit()
print(res2.summary2())
```

```
##                   Results: Ordinary least squares
## ================================================================
## Model:              OLS              Adj. R-squared:     0.936
## Dependent Variable: x                AIC:                90.1760
## Date:               2023-08-06 22:40 BIC:                93.1632
## No. Observations:   20               Log-Likelihood:     -42.088
## Df Model:           2                F-statistic:        140.1
```

```
## Df Residuals:       17               Prob (F-statistic): 2.75e-11
## R-squared:          0.943            Scale:              4.6341
## ------------------------------------------------------------------
##                Coef.   Std.Err.     t     P>|t|    [0.025   0.975]
## ------------------------------------------------------------------
## Intercept      6.9208    1.6016   4.3213  0.0005   3.5418  10.2998
## t             -0.5974    0.3512  -1.7009  0.1072  -1.3385   0.1436
## poly(t, 2)     0.0911    0.0162   5.6092  0.0000   0.0569   0.1254
## ------------------------------------------------------------------
## Omnibus:               2.641        Durbin-Watson:          3.045
## Prob(Omnibus):         0.267        Jarque-Bera (JB):       1.855
## Skew:                  -0.564       Prob(JB):               0.395
## Kurtosis:              2.023        Condition No.:          646
## ==================================================================
## Notes:
## [1] Standard Errors assume that the covariance matrix of the
## errors is correctly specified.
```

La droite de tendance polynomiale d'order 2 s'écrit :

$$Z_2(t) = 6.9208 - 0.5974t + 0.0911t^2 \tag{3.31}$$

— Ajustement par un polynôme de degré 3.

```
# Ajustement polynomial degré 3
res3 = smf.ols("x~t+poly(t,2)+poly(t,3)",data = df).fit()
print(res3.summary2())
```

```
##                  Results: Ordinary least squares
## ==================================================================
## Model:              OLS              Adj. R-squared:     0.933
## Dependent Variable: x                AIC:                91.9373
## Date:               2023-08-06 22:40 BIC:                95.9202
## No. Observations:   20               Log-Likelihood:     -41.969
## Df Model:           3                F-statistic:        89.00
## Df Residuals:       16               Prob (F-statistic): 3.40e-10
## R-squared:          0.943            Scale:              4.8653
## ------------------------------------------------------------------
##                Coef.   Std.Err.     t     P>|t|    [0.025   0.975]
## ------------------------------------------------------------------
## Intercept      6.1476    2.4094   2.5515  0.0213   1.0399  11.2553
## t             -0.2029    0.9695  -0.2093  0.8369  -2.2581   1.8523
## poly(t, 2)     0.0453    0.1059   0.4276  0.6746  -0.1792   0.2698
## poly(t, 3)     0.0015    0.0033   0.4383  0.6670  -0.0056   0.0085
## ------------------------------------------------------------------
## Omnibus:               2.634        Durbin-Watson:          3.066
## Prob(Omnibus):         0.268        Jarque-Bera (JB):       1.650
## Skew:                  -0.467       Prob(JB):               0.438
## Kurtosis:              1.948        Condition No.:          17167
## ==================================================================
## Notes:
```

```
## [1] Standard Errors assume that the covariance matrix of the
## errors is correctly specified.
## [2] The condition number is large, 1.72e+04. This might indicate
## that there are strong multicollinearity or other numerical
## problems.
```

La droite de tendance polynomiale d'order 3 s'écrit :

$$Z_3(t) = 6.1476 - 0.2029t + 0.0453t^2 + 0.0015t^3 \tag{3.32}$$

```
# Valeurs ajustées
fit = pd.DataFrame({"fit0" : res0.fittedvalues, "fit1": res1.fittedvalues,
  "fit2" : res2.fittedvalues, "fit3": res3.fittedvalues
},index= df.index)

# Représentation graphique
fig = plt.figure(figsize=(16,8))
for idx,name in enumerate(fit.columns):
  ax = fig.add_subplot(2,2,idx+1);
  ax.scatter(df.t,df.x,color='black');
  ax.plot(df.t,df.x,color='black');
  ax.plot(df.t,fit[name],color='red');
  ax.set_title("p = "+str(idx));
  ax.set_xlabel("t");
  ax.set_ylabel("x");
plt.tight_layout();
plt.show()
```

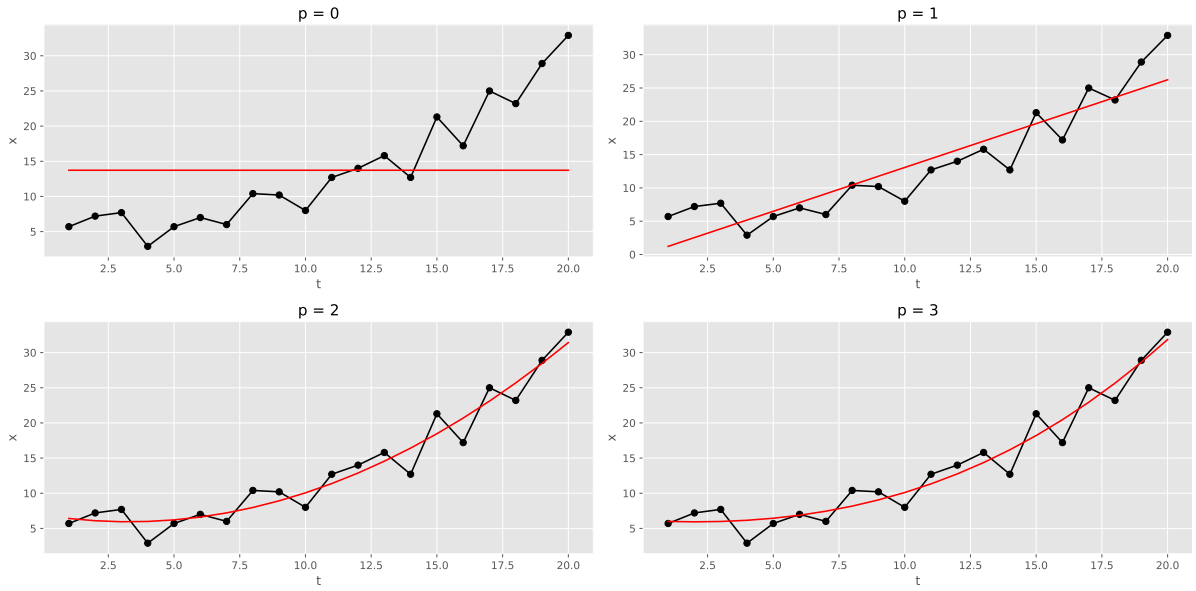


Figure 3.3 – Ajustement polynomial

La somme des carrés des résidus s'obtient :

```
# Somme des carrés des résidus
def SCR(x_true, x_pred):
  scr = np.sum([(x - y)**2 for x,y in zip(x_true,x_pred)])
  return scr
scr = np.array([SCR(df.x, fit[i]) for i in fit.columns])
```

La part de la variance expliquée est déterminée à l'aide de la fonction r2_score.

```
# Part de variance expliquée
from sklearn.metrics import r2_score
from mizani.formatters import percent_format
r2score = np.array([r2_score(df.x, fit[i]) for i in fit.columns])
r2score = percent_format()(r2score)
res = pd.DataFrame({"SCR" : np.around(scr,2),"R2 score" : r2score})
res = res.reset_index().rename(columns = {"index":"p"})
```

Table 3.6 – Critères de choix numériques

p	SCR	R2 score
0	1376.82	0.0%
1	224.58	83.7%
2	78.78	94.3%
3	77.84	94.3%

3.1.5 Autres méthodes d'estimation

3.1.5.1 Estimation par la méthode du noyau

L'estimation par noyau est une méthode non - paramétrique d'estimation de la densité d'une variable aléatoire. Elle a été introduite par Rosenblatt (1956) et développée par Parzen (1962). Cette méthode permet d'obtenir une densité continue et constitue en ce sens une généralisation de la méthode de l'histogramme. En effet, la fonction indicatrice utilisée pour l'histogramme est ici remplacée par une fonction continue (le noyau) et une somme de fonctions continues reste continue.

Soit X une variable aléatoire qui admet une densité $f_X(x)$. Nous nous proposons d'estimer cette dernière à partir d'un $T-$échantillon de X.

Définition 3.4 *noyau d'une fonction*

On appelle noyau de Parzen (1962), toute fonction $K(x)$ positive, paire, bornée, satisfaisant aux conditions :

$$\lim_{x\to+\infty} xK(x) = 0 \quad \text{et} \quad \int_{-\infty}^{+\infty} K(x)\mathrm{d}x < +\infty \tag{3.33}$$

Remarque 3.4

Nous avons les conditions supplémentaires suivantes :

1. Le noyau est une densité : $\int_{-\infty}^{+\infty} K(x)\mathrm{d}x = 1$
2. Le noyau est de carré intégrable : $d^2(K) = \int_{-\infty}^{+\infty} K^2(x)\mathrm{d}x < +\infty$
3. Le noyau admet un moment d'ordre 2 : $\sigma^2(K) = \int_{-\infty}^{+\infty} x^2 K(x)\mathrm{d}x < +\infty$

Ces conditions supplémentaires sont très utiles pour obtenir les propriétés usuelles pour les estimateurs de densités.

Le tableau (3.7) donne les noyaux les plus utilisés :

Table 3.7 – Exemples de noyau

Noyaux	Densités	Noyaux	Densités
Uniforme $\mathcal{U}(]-1,1[)$	$\frac{1}{2}1_{]-1,1[}(x)$	Gaussien $\mathcal{N}(0,1)$	$\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{x^2}{2}\right\}$
Triangulaire $\mathcal{TR}(0)$	$(1-\lvert x\rvert)\,1_{]-1,1[}(x)$	Circulaire	$\frac{\pi}{4}\cos\left(\frac{\pi}{2}x\right)1_{]-1,1[}(x)$
Epanechnikov	$\frac{3}{4}\left(1-x^2\right)1_{]-1,1[}(x)$	Quartique	$\frac{15}{16}\left(1-x^2\right)^2 1_{]-1,1[}(x)$
Tricube	$\frac{70}{81}\left(1-\lvert x\rvert^3\right)^3 1_{]-1,1[}$	Logistique	$\frac{1}{1+\mathrm{e}^{-x}}$
Triweight	$\frac{35}{32}\left(1-x^2\right)^3 1_{]-1,1[}(x)$		

Exemple 3.10

Nous représentons graphiquement les différents noyaux.

```
# Fonction indicatrice
def indicatrice(x):
  if np.abs(x)<1:return 1
  else : return 0
# Noyau gaussien
import scipy.stats as st
def gaussian_kd(x):return st.norm.pdf(x)
# Noyau uniforme
def uniform_kd(x):return 0.5*indicatrice(x)
# Noyau triangulaire
def triangular_kd(x):return (1-np.abs(x))*indicatrice(x)
# Noyau circulaire
def circular_kd(x):return (np.pi/4)*np.cos(0.5*np.pi*x)*indicatrice(x)
# Noyau d'epanechnikov
def epanechnikov_kd(x):return (3/4)*(1-x**2)*indicatrice(x)
# Noyau quartique
def quartic_kd(x):return (15/16)*((1-x**2)**2)*indicatrice(x)
# Noyau tricube
def tricube_kd(x):return (70/81)*((1-np.abs(x)**3)**3)*indicatrice(x)
# Noyau logistique
def logistic_kd(x):return 1/(1+np.exp(-x))
# Noyau triweight
def triweight_kd(x):return (35/32)*((1-x**2)**3)*indicatrice(x)
# Application
pas =np.linspace(-3,3,100)
kernel = pd.DataFrame({
```

```
  "gaussian" : np.array([gaussian_kd(x) for x in pas]),
  "uniform" : np.array([uniform_kd(x) for x in pas]),
  "triangular" : np.array([triangular_kd(x) for x in pas]),
  "circular" : np.array([circular_kd(x) for x in pas]),
  "epanechnikov" : np.array([epanechnikov_kd(x) for x in pas]),
  "quartic" : np.array([quartic_kd(x) for x in pas]),
  "tricube" : np.array([tricube_kd(x) for x in pas]),
  "logistic" : np.array([logistic_kd(x) for x in pas]),
  "triweight" : np.array([triweight_kd(x) for x in pas])
})

# Représentation graphique
fig = plt.figure(figsize=(16,8));
for idx, name in enumerate(kernel.columns):
  ax = fig.add_subplot(3,3,idx+1);
  ax.plot(pas,kernel[name],color="red");
  ax.set_title(f"{name} kernel");
  ax.set_xlabel("x");
  ax.set_ylabel("K(x)");
plt.tight_layout();
plt.show()
```

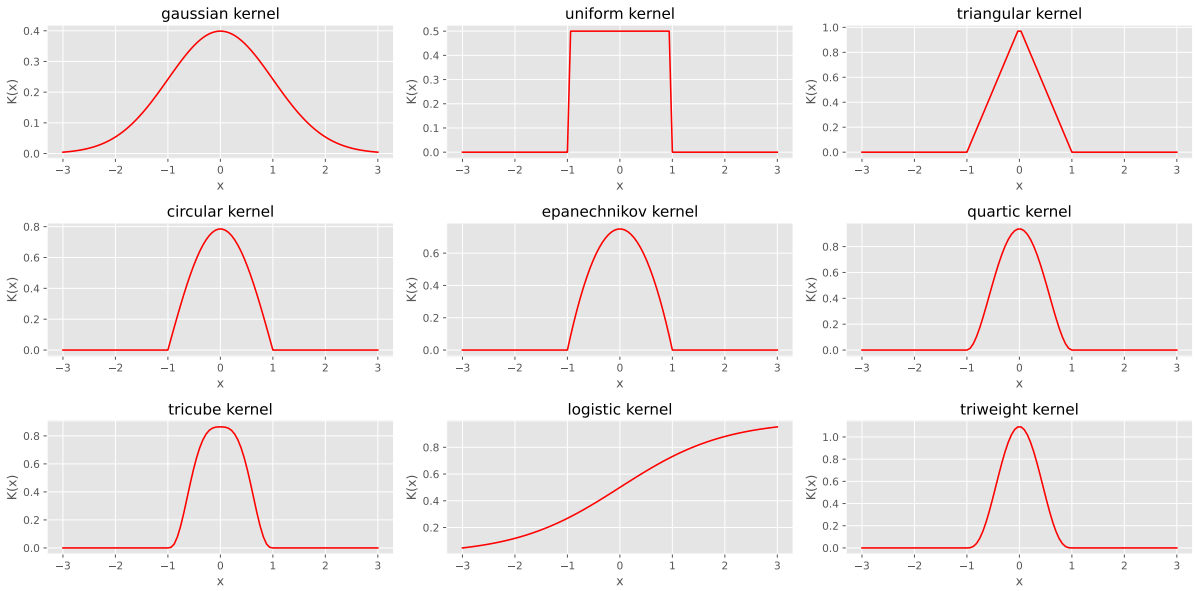


Figure 3.4 – Exemple de noyau de fonction

Une façon rapide et efficace d'avoir ces différentes figures est d'utiliser le module Statsmodels.

```
# Noyau avec Statsmodels
from statsmodels.nonparametric.kde import kernel_switch
fig = plt.figure(figsize=(16,8))
for i, (ker_name, ker_class) in enumerate(kernel_switch.items()):
    kernel = ker_class()
    domain = kernel.domain or [-3, 3]
    x_vals = np.linspace(*domain, num=2 ** 10)
```

```
    y_vals = kernel(x_vals)
    ax = fig.add_subplot(3, 3, i + 1);
    ax.set_title('Kernel function "{}"'.format(ker_name));
    ax.plot(x_vals, y_vals, lw=3, label="{}".format(ker_name));
    ax.set_xlim([-3,3]);
plt.tight_layout();
plt.show()
```

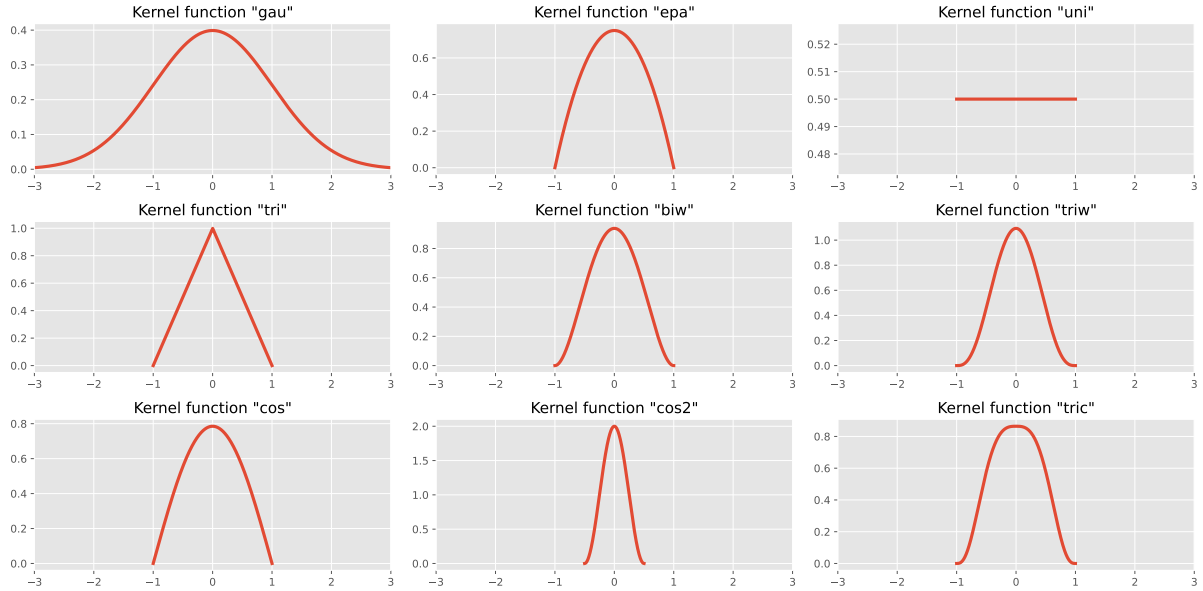


Figure 3.5 – Exemple de noyau de fonction avec Statsmodels

Toutes ces fonctions sont bien des noyaux et satisfont à l'ensemble des conditions de la remarque (3.4). En particulier nous avons :

Table 3.8 – Propriétés des noyaux usuels

Noyaux	$d^2(K)$	$\sigma^2(K)$	Noyaux	$d^2(K)$	$\sigma^2(K)$
Uniforme	$1/2$	$1/3$	Triangulaire	$2/3$	$1/6$
Epanechnikov	$3/5$	$1/5$	Quartique	$5/7$	$1/7$
Circulaire	$\pi^2/16$	$1-(8/\pi^2)$	Gaussien	$1/2$	1

Définition 3.5 *Estimateur de Parzen - Rosenblatt*

Soit $X_1, \ldots, X_T$ un échantillon indépendant et identiquement distribué d'une variable aléatoire de densité f continue. On appelle estimateur de Parzen - Rosenblatt ou estimateur à noyau de la densité f, la variable aléatoire définie par

$$\hat{f}_h(x) = \frac{1}{Th} \sum_{t=1}^{t=T} K\left(\frac{x - X_t}{h}\right) = \frac{1}{h} \mathbb{E}\left[K\left(\frac{x - X}{h}\right)\right] \tag{3.34}$$

où K est le noyau de Parzen et le nombre $h > 0$ est un paramètre de lissage appelé fenêtre.

Définition 3.6 *Estimateur à noyau de f*

Soit un réel $h > 0$, soit un noyau K. On appelle estimateur à noyau de f_X associé à la fenêtre h et au noyau K, la fonction $\hat{f}_h$ définie par

$$\hat{f}_h(x) = \frac{\sum_{t=1}^{t=T} X_t K\left(\frac{x-t}{h}\right)}{\sum_{t=1}^{t=T} K\left(\frac{x-t}{h}\right)} \tag{3.35}$$

C'est une estimation non - paramétrique de la tendance de la série. La régularité de cet estimateur dépend de la taille de fenêtre du noyau.

Une fonction permettant d'effectuer une régression à noyau en Python est KernelReg du package Statsmodels.

Exemple 3.11 *Estimation de la tendance par noyau sur la série Air passengers*

Appliquons le noyau gaussien sur la série Airpassengers avec différentes valeurs de la fenêtre h.

```
# Estimation à noyau
from statsmodels.nonparametric.kernel_regression import KernelReg
xval = dataframe.temps
yval = dataframe.passengers
kernel_reg=KernelReg(endog=yval,exog=xval,var_type='c',bw=np.array([10])).fit()
# Représentation graphique
fig, axe =  plt.subplots(figsize=(16,6))
axe.plot(xval,yval,color="black",label="AirPassengers");
axe.plot(xval,kernel_reg[0],color="red",label="Kernel régression");
axe.set_xlabel("Date");
axe.set_ylabel("Nombre de passagers");
axe.legend();
plt.show()
```

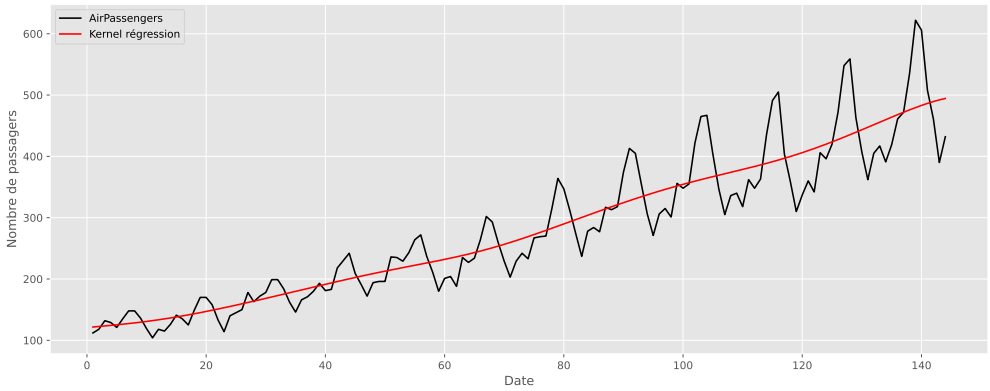


Figure 3.6 – Estimation à noyau

Proposition 3.1 *Choix de h*

Le choix de h est une étape importante lorsque de l'estimation par noyau dans le sens où $\hat{h}$ nécessite de connaître la densité que l'on cherche à estimer, ce qui n'est en pratique pas le cas. Par ailleurs, si h est trop petit, le biais de l'estimateur devient petit devant sa variance et l'estimateur trop fluctuant : on obtient un phénomène de sous lissage. Dans le cas contraire, lorsque h est trop grand, le biais prend l'ascendant sur la variance et l'estimateur varie peu : on obtient un phénomène de sur - lissage.

```
# Représentation graphique
kernel_reg1=KernelReg(endog=yval,exog=xval,var_type='c',bw=np.array([5])).fit()
kernel_reg2=KernelReg(endog=yval,exog=xval,var_type='c',bw=np.array([50])).fit()
# Représentation graphique
fig, axe =  plt.subplots(figsize=(16,6))
axe.plot(xval,yval,color="black",label="AirPassengers");
axe.plot(xval,kernel_reg1[0],color="red",label="Kernel régression (h=5)");
axe.plot(xval,kernel_reg2[0],color="blue",label="Kernel régression (h=50)");
axe.set_xlabel("Date");
axe.set_ylabel("Nombre de passagers");
axe.legend();
plt.show()
```

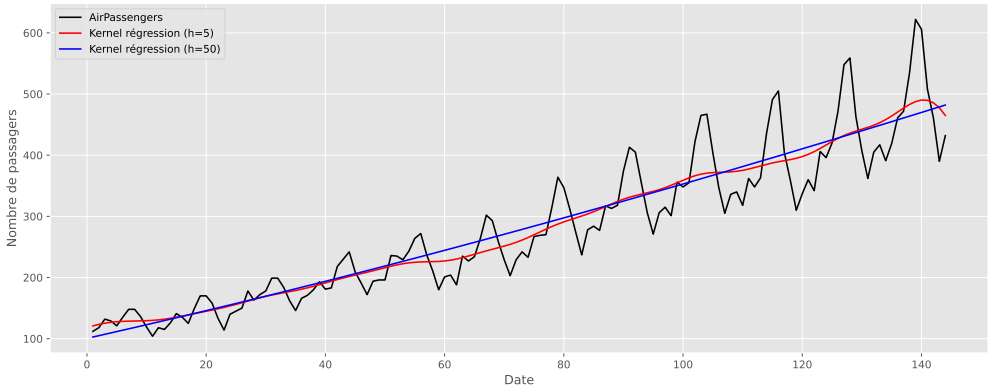


Figure 3.7 – Estimation à noyau avec h = 5 (en rouge) et h = 50 (en bleu)

En pratique, on utilise souvent la méthode de validation croisée pour choisir automatiquement h. Cette méthode demande dans un premier temps de trouver un estimateur sans biais de $J(h) = MSE(\hat{f}_h) - ||f||_2^2$ et de minimiser ensuite cet estimateur.

On trouve que $\widehat{J}(h) = ||f||_2^2 - \dfrac{2}{T(T-1)h} \displaystyle\sum_{i=1}^{i=T} \sum_{j\neq i=1}^{j=T} K\left(\dfrac{X_i - X_j}{h}\right)$ convient. Il suffit alors de minimiser cette fonction et l'on obtient :

$$\hat{h} = \underset{h>0}{\arg\min} \widehat{J}(h) \tag{3.36}$$

```
# Estimation à noyau
kernel_reg_ls = KernelReg(endog=yval,exog=xval,var_type='c',bw= "cv_ls").fit()
# Représentation graphique
fig, axe =  plt.subplots(figsize=(16,6))
axe.plot(xval,yval,color="black",label="AirPassengers");
```

```
axe.plot(xval,kernel_reg_ls[0],color="red",label="Kernel régression");
axe.set_xlabel("Date");
axe.set_ylabel("Nombre de passagers");
axe.legend();
plt.show()
```

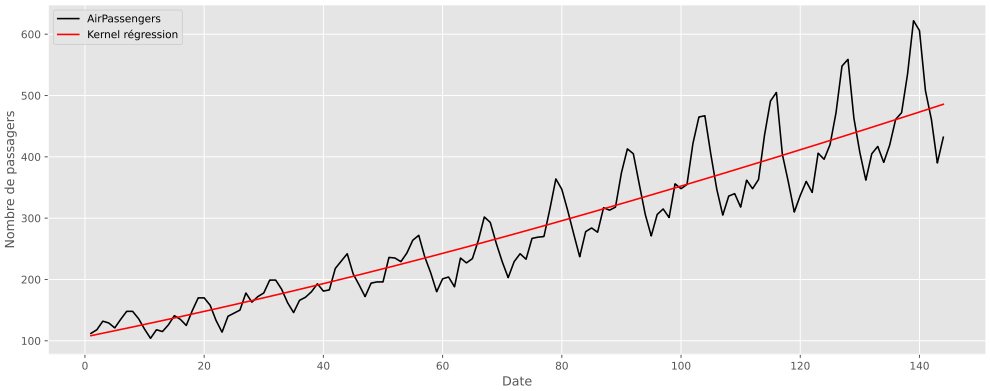


Figure 3.8 – Estimation à noyau avec fenêtre optimale

3.1.5.2 Estimation semi - paramétrique de la tendance

Une autre alternative pour estimer la tendance par la fonction f est de procéder par projection sur des bases de fonctions adaptés, par exemple des fonctions splines polynomiales par morceau.

Définition 3.7

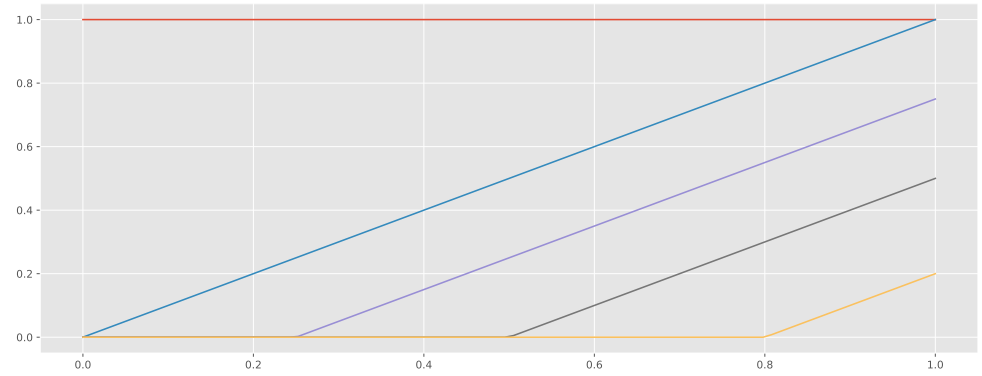
Soit $1 \leq n_1 \leq \cdots \leq n_k \leq n$ un vecteur de coefficients appelés noeuds définissant une partitition du temps, q un entier strictemet positif, alors : $(1, x, x^2, \ldots, x^q, (x - n_1)_+^q, \ldots, (x - n_k)_+^q)$ est appelée base de truncated power functions de degré q et est de classe $\mathcal{C}^{q-1}$.

Par exemple, pour $q = 1$ on obtient une base de fonction spline linéaire dont l'aspect est le suivant :

```
# Exemple d'application
n = 100
def f1(x): return x
def f2(x): return np.maximum(x - 0.25, 0)
def f3(x): return np.maximum(x - 0.5, 0)
def f4(x): return np.maximum(x - 0.8, 0)

pas = np.linspace(0,1,n)
design = pd.DataFrame({
  "const": np.repeat(1,n),
  "f1" : list([f1(x) for x in pas]),
  "f2" : list([f2(x) for x in pas]),
  "f3" : list([f3(x) for x in pas]),
  "f4" : list([f4(x) for x in pas])
},index = pas)
```

```
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(design);
plt.show()
```



ce qui permet de modéliser des tendances linéaires par morceau du type :

```
 # Modélisation des tendances linéaires
np.random.seed(123)
coef = np.random.uniform(low=-1.0, high=1.0, size=5)
f = design.values.dot(coef)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(pas, f, color="purple");
axe.vlines([0.25, 0.5, 0.8],np.min(f)-0.1,np.max(f)+0.1,colors='black',
           linestyles="--");
plt.show()
```

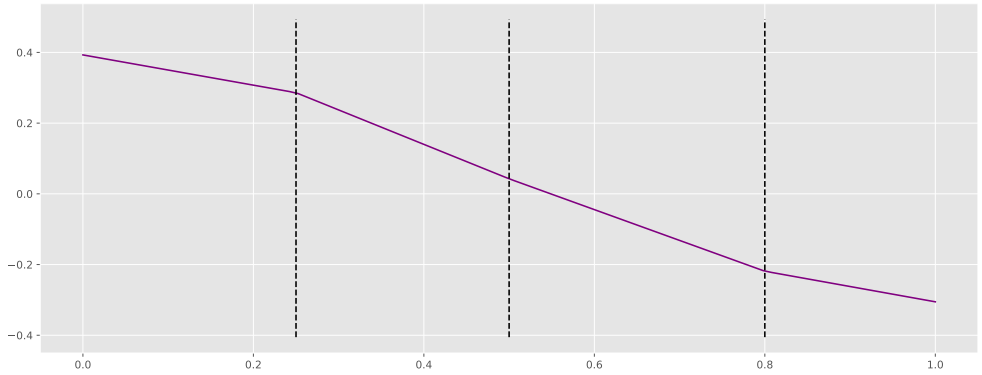


Figure 3.9 – truncated power functions

En pratique, les coefficients de projection sur la base sont estimés par régression linéaire sur la matrice de design : $X = (1, x, x^2, \dots, x^q, (x - n_1)_+^q, \dots, (x - n_k)_+^q)$.

Le choix des « cassures » donc des noeuds est déterminant. $q = 3$ est un choix courant car il permet de contourner ce problème du choix des noeuds en le substituant à un problème de régularisation en résolvant le problème de régression pénalisée suivant :

$$\min_{f \in S_3} (X_t - f(t))^2 + \lambda \int f''(x)^2 \mathrm{d}x \tag{3.37}$$

où S_3 est l'espace engendré par $X = (1, x, x^2, x^3, (x-n_1)_+^3, \dots, (x-n_k)_+^3)$ en prenant k suffisamment grand, $\lambda > 0$ est un paramètre à calibrer, par exemple par validation croisée.

En pratique, on pourra utiliser la fonction LinearGAM du package pygam.

```
# Generalized Additive Models
from pygam import LinearGAM, s
gam = LinearGAM(s(0,n_splines=10)).fit(xval,yval)
fit_gam = gam.predict(xval)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(xval,yval, color= "black",label="air passengers");
axe.plot(xval,fit_gam, color = "red",label = "gam model");
axe.set_xlabel("Date");
axe.set_ylabel("Nombre de passagers");
axe.legend();
plt.show()
```

3.2 Estimation de la saisonnalité et prévision

L'étude de la saisonnalité est une étape fondamentale dans l'étude des séries chronologiques. On cherche tout d'abord à savoir si la série qu'on étudie présente des variations saisonnières. Dans le cas échéant, on détermine la saisonnalité à travers les coefficients saisonniers. Cependant, ces coefficients doivent respecter le principe de conservation des aires.

3.2.1 Principe de conservation des aires et coefficients saisonniers

L'un des objectifs de l'analyse de la saisonnalité est de faire une répartition de la distribution infra-annuelle de la série sans pour autant modifier le niveau global. C'est comme si on diminuait les grandes valeurs pour augmenter légèrement les petites valeurs ou encore faire une compensation entre les pics et les creux. Cet exercice conduit à une nouvelle série appelée série corrigée des variations saisonnières X_t^{cvs} ou série désaisonnalisée. C'est la série obtenue après élimination de la composante saisonnière.

Propriété 3.6 *Principe de conservation des aires*

Le principe de conservation des aires voudrait que la série initiale X_t ne soit pas en moyenne différente de la série corrigée des variations saisonnières $X_{CVS,t}$, c'est-à-dire : $\overline{X} = \overline{X}_{CVS}$.

Définition 3.8 *Coefficients saisonniers*

Un coefficient saisonnier est un coefficient affecté à une période de l'année (trimestre ou mois) et qui permet d'illustrer le caractère propice ou non de cette période pour la série.

Pour calculer les coefficients saisonniers, on limite l'analyse à la composante tendancielle et la composante saisonnière.

3.2.1.1 Modèle additif

Le modèle devient : $X_t = Z_t + S_t$. On calcule ensuite les valeurs des variations saisonnières : $S_t = X_t - Z_t$.

Le coefficient saisonnier d'une période quelconque est la moyenne arithmétique des valeurs saisonnières S_t relatives. Toutefois, dans un modèle additif, la somme des coefficients saisonniers doit être égale à 0. Il faut donc ajuster les moyennes calculées pour rendre leur somme égale à 0.

- On calcule la moyenne $\overline{C} = \frac{1}{p}\sum_{j=1}^{j=p} C_j$
- Puis, on calcule les coefficients saisonniers définitifs C_j^* à l'aide la formule $C_j^* = C_j - \overline{C} \quad \forall j$.
- On vérifie que $\sum_{j=1}^{j=p} C_j^* = 0$

3.2.1.2 Modèle multiplicatif

Le modèle multiplicatif : $X_t = Z_t \times S_t$. On calcule ensuite les valeurs des variations saisonnières : $S_t = \frac{X_t}{Z_t}$ ou $S_t = \frac{X_t}{Z_t} \times 100$.

Le coefficient saisonnier d'une période quelconque est égal à la moyenne arithmétique des S_t correspondantes. Toutefois avec un modèle multiplicatif, la somme des coefficients saisonniers doit être égale à 4 (ou 400) s'il s'agit des observations trimestrielles (car 4 trimestres dans l'année) ou 12 (ou 1200) s'il s'agit des données mensuelles (car 12 mois dans l'année). Il faut donc ajuster les moyennes calculées pour rendre leur somme compatible.

Exemple 3.12 *Composante saisonnière de la série Air passengers par MCO*

Le graphique suivant nous montre l'évolution de la composante saisonnière pour le jeu de données Air passengers.

```
# Calcul de la composante saisonnière
season = pd.DataFrame({
  "seasonal" : list(donnee.passengers.values/model.fittedvalues.values)
},index =  donnee.index)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(season.seasonal,color='black');
axe.set_xlabel('mois');
axe.set_ylabel('value');
plt.show()
```

On peut à présent calculer les coefficients saisonniers.

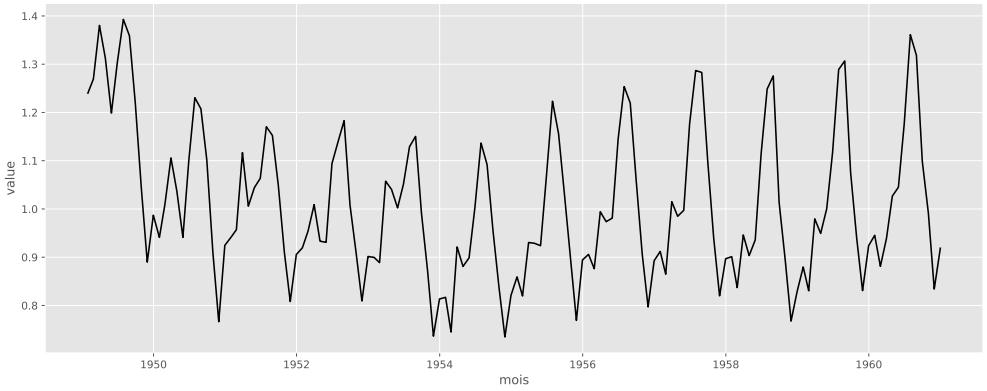


Figure 3.10 – Évolution de la composante saisonnière

```
# Coefficients saisonniers
p = 12
m = int(T/p)
meancoef = np.mean(season.seasonal.values.reshape(m,p), axis = 0)
```

On vérifie si la somme des coefficients saisonniers est égale à $p = 12$.

```
# Somme Coefficients saisonniers
meansum = meancoef.sum()
print(meansum.round(4))
```

```
## 12.1142
```

On constate que la somme est supérieure à 12 (12.1142). On doit appliquer le principe de conservation des aires afin d'ajuster la moyenne à 12.

- On calcule la moyenne $\overline{C} = \frac{1}{p}\sum_{j=1}^{j=p} C_j$
- Puis, on calcule les coefficients saisonniers définitifs C_j^* à l'aide la formule $C_j^* = \frac{C_j}{\overline{C}} \quad \forall j.$
- On vérifie que $\sum_{j=1}^{j=p} C_j^* = p$

```
# Principe de conservation des aires
c_hat = np.divide(meancoef,meancoef.mean())
```

On vérifie à présent que la somme de nos nouveaux coefficients est égale à 12.

```
# Somme
print(np.sum(c_hat).round(4))
```

```
## 12.0
```

On voit effectivement que le résultat est égal à 12.

```
import calendar
saison_coef = pd.DataFrame(c_hat,columns = ["saisonal coef."])
saison_coef.index = [calendar.month_abbr[x] for x in range(1,13)]
```

Table 3.9 – Coefficients saisonniers

	janv.	févr.	mars	avr.	mai	juin	juil.	août	sept.	oct.	nov.	déc.
saisonal coef.	0.92	0.9	1.02	0.99	0.98	1.11	1.23	1.21	1.05	0.91	0.79	0.88

3.2.2 Désaisonalisation de la série

La désaisonalisation a pour but d'éliminer l'influence saisonnière pour ne garder que la tendance réelle.

- Le modèle additif : $X_{CVS,t} = X_t - C$
- Le modèle multiplicatif : $X_{CVS,t} = \dfrac{X_t}{C}$ (Il faut multiplier par 100 si le coefficient est en pourcentage)

Dans les deux équations, C représentent le coefficient saisonnier de la période considérée. Si on note C_j le $j^{\text{ième}}$ coefficient saisonnier, alors C_j représente une moyenne des écarts observées à la période j de chaque année. Si le principe de conservation des aires est vérifiée ($\sum_{j=1}^{j=p} C_j = 0$ pour le modèle additif et $\sum_{j=1}^{j=p} C_j = p$ pour le modèle multiplicatif), alors les C_j correspondent aux coefficients saisonniers définitifs.

Exemple 3.13 *Série corrigée des variations saisonnières de la série Air passengers*

On souhaite calculer la série corrigée des variations saisonnières pour la série Air passengers à partir des coefficients saisonniers obtenus par MCO.

```
# Séries CVS (série desaisonnalisée)
K = np.tile(c_hat,m)
xcvs = pd.DataFrame(np.array(donnee.passengers)/K,columns=['cvs'],
                    index = donnee.index)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(donnee.passengers,color='black', label = "Air passengers");
axe.plot(xcvs.cvs,color='red', label = "Séries CVS");
axe.set_xlabel('mois');
axe.set_ylabel('value');
axe.legend();
plt.show()
```

3.2.3 Prévision des valeurs futures

Lorsque la tendance est ajustée et que l'on a son expression en fonction du temps t, on peut facilement faire des prévisions à la date $T+h$ où $h \geq 1$, c'est - à dire l'horizon h. Pour cela, on calcule d'abord la tendance estimée à la date H : $\widehat{Z}_{T+h} = \hat{a} + \hat{b} \times (T+h)$. Puis

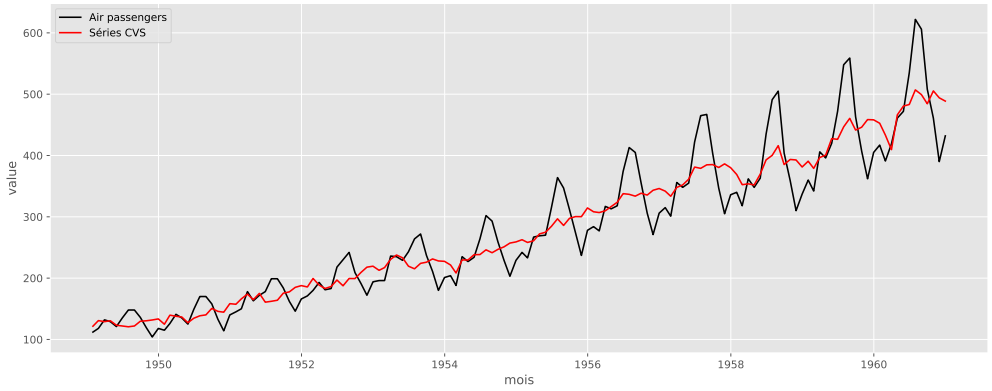


Figure 3.11 – Évolution de la série corrigée des variations saisonnières

— Pour le modèle additif, on additionne à cette valeur le coefficient saisonnier correspondant : $\hat{X}_{T+h} = \widehat{Z}_{T+H} + \hat{c}_j$
— Pour le modèle multiplicatif, on multiplie cette valeur par le coefficient saisonnier correspondant : $\widehat{X}_{T+h} = \widehat{Z}_{T+h} \times \hat{c}_j$ (Il faut divise par 100 si le coefficient saisonnier est en pourcentage).

Exemple 3.14 *Prévision de la série Air passengers par MCO*

Pour la série Air passengers, on cherche à effectuer une prévision mensuelle pour les deux prochaines années, à savoir 1961 et 1962.

```
## Prévision
# Création de la période
indexpred = pd.date_range(start="1961-01-31",end = "1962-12-31",freq="M")
tpred = pd.DataFrame(np.arange(T+1,T+25,1),index = indexpred,columns = ["temps"])
# Calcul des valeurs ajustée pour la prédiction
ypreddf = model.predict(tpred); ypred = np.array(ypreddf).reshape(len(tpred),1)
# Lissage avec les coefficients saisonniers
xpred = ypred*np.array(K[:len(tpred)]).reshape(len(tpred),1)
xpreddf = pd.DataFrame(xpred,index = indexpred, columns = ["forecast"])
```

Table 3.10 – Valeurs prévues par moindre carré ordinaire

	forecast
1961-01-31	435.768
1961-02-28	429.208
1961-03-31	489.341
1961-04-30	475.459
1961-05-31	474.987
1961-06-30	538.364

```
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(donnee.passengers,color='black', label = "Air Passengers");
axe.plot(xpreddf.forecast,color='red', label = "Prévision par MCO");
axe.set_xlabel('mois');
```

```
axe.set_ylabel('value');
axe.legend();
plt.show()
```

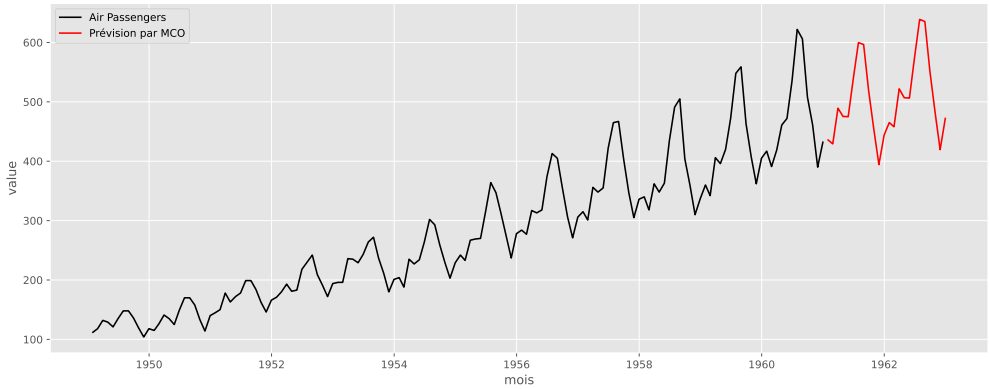


Figure 3.12 – Prévision par moindre carré ordinaire

3.2.4 Séries ajustées et variations résiduelles

3.2.4.1 Séries ajustées

La série ajustée, notée $\widehat{X}_t$ est la série obtenue en recomposant les deux composantes estimées, à savoir, la tendance et les variations saisonnières (coefficients saisonniers) selon le modèle choisit :

- Pour un modèle additif : $\widehat{X}_t = \widehat{Z}_t + C$
- Pour un modèle multiplicatif : $\widehat{X}_t = \widehat{Z}_t \times C$

Exemple 3.15 *Série ajustée de la série Air passengers*

On calcule la série ajustée de notre série Air passengers.

```
# Séries ajustées
xadjusted = pd.DataFrame(model.fittedvalues*K, index = donnee.index,
                         columns = ["adjusted"])
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(donnee.passengers,color='black',label = "Air Passengers");
axe.plot(xadjusted.adjusted,color='red',label = "Série ajustée");
axe.set_xlabel('mois');
axe.set_ylabel('value');
axe.legend();
plt.show()
```

La série ajustée $\widehat{X}_t$ représente l'évolution qu'aurait subit la série étudiée X_t si les variations saisonnières étaient périodiques et s'il n'y avait pas de variations résiduelles.

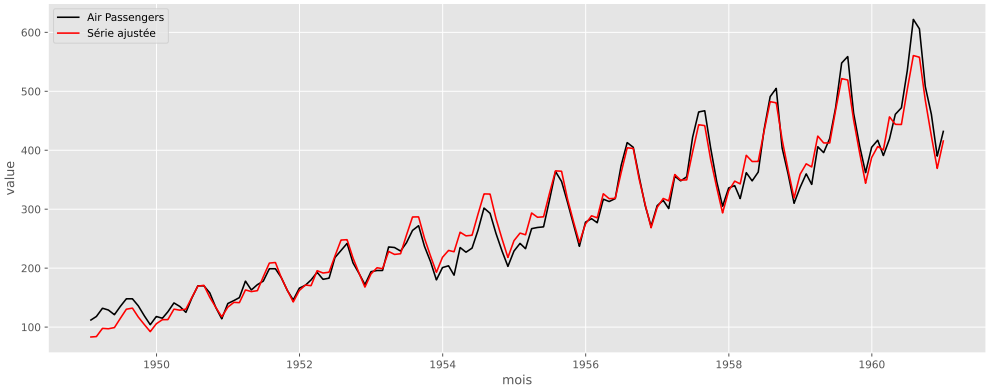


Figure 3.13 – Evolution de la série Air passengers ajustée

3.2.4.2 Variations résiduelles

Les variations résiduelles ou accidentelles se calculent de la façon suivante.

- Pour le modèle additif : $\hat{\varepsilon}_t = X_t - \widehat{X}_t$
- Pour le modèle multiplicatif complet : $\hat{\varepsilon}_t = \dfrac{X_t}{\widehat{X}_t}$

Une fois estimées les différentes composantes du modèle, on peut vérifier la pertinence du modèle par une analyse des résidus. En effet, si le modèle est bon, il ne doit rester dans les résidus aucune trace du saisonnier. Pour le vérifier, on trace le corrélogramme des résidus c'est - dire le graphe d'un estimateur de la fonction d'autocorrélation. Ce dernier n'est tracé en théorie que dans le cas où la série est stationnaire, ce qui implique en particulier qu'il n'y a dans cette série ni tendance ni saisonnalité. En pratique, on s'en sert (dans le cas de l'analyse des résidus) pour vérifier justement l'absence de saisonnalité dans les résidus.

Exemple 3.16 *Résidus ajustés de la série Air passengers*

On calcule la série des résidus ajustés de la série Air passengers.

```
# résidus ajustés
xadjresid = pd.DataFrame(donnee.passengers.values/xadjusted.adjusted.values,
                         columns = ['resid'],index= donnee.index)
import statsmodels.api as sm
import scipy.stats as st
fig, axe = plt.subplots(2,2,figsize=(16,6))
sm.graphics.tsa.plot_acf(xadjresid,lags=36,ax=axe[0,1]);
axe[0,0].plot(xadjresid.resid,color='black');
axe[0,0].set_xlabel('mois');
axe[0,0].set_ylabel('value');
sm.qqplot(xadjresid,st.t,line ='45', fit=True, ax = axe[1,0]);
sns.histplot(data = xadjresid,kde=True,ax = axe[1,1]);
plt.tight_layout();
plt.show()
```

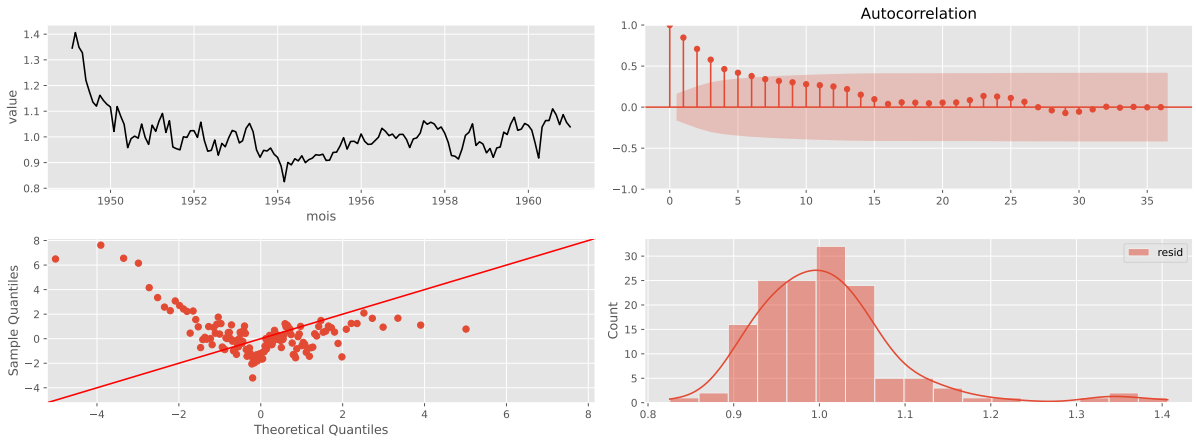


Figure 3.14 – Diagnostics des résidus

4

Désaisonnalisation par régression linéaire

Sommaire

La désaisonnalisation vise à éliminer les effets saisonniers et de calendrier combinés. Avant d'appliquer cette méthode, il faut donc s'assurer que de tels effets sont présents et qu'il est possible de les estimer correctement. Lorsqu'il est impossible de repérer des effets saisonniers et/ou de calendrier dans une série chronologique, cette série est réputée désaisonnalisée de facto. Les séries désaisonnalisées ne doivent pas présenter de saisonnalité résiduelle et sont généralement plus lisses que les séries brutes correspondantes.

4.1 Le modèle linéaire

Au chapitre introductif, nous avons affirmé que la série chronologique X_t était la somme de 2 composantes déterministes (une tendance Z_t et une saisonnalité S_t) et d'une composante aléatoire ε_t :

$$X_t = Z_t + S_t + \varepsilon_t \tag{4.1}$$

On suppose que la composante tendancielle Z_t et la composante saisonnière S_t sont des combinaisons linéaires de fonctions continues dans le temps , Z_t^i et S_t^j, i.e

$$\begin{cases} Z_t = \phi_1 Z_t^1 + \phi_2 Z_t^2 + \cdots + \phi_p Z_t^p \\ S_t = \theta_1 S_t^1 + \theta_2 S_t^2 + \cdots + \theta_q S_t^q \end{cases} \tag{4.2}$$

Le but est d'estimer les paramètres $\phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q$ à partir des T observations, c'est-à-dire :

$$X_t = \sum_{i=1}^{p} \phi_i Z_t^i + \sum_{j=1}^{q} \theta_j S_t^j + \varepsilon_t, \quad \text{pour} \quad t = 1, \ldots, T \tag{4.3}$$

Proposition 4.1 *Hypothèses sur les erreurs*

On supposera que les erreurs vérifient les propriétés suivantes :

— Centrées : $\mathbb{E}(\varepsilon_t) = 0$, $\forall t$
— Homoscédasticité : $\mathrm{V}(\varepsilon_t) = \sigma_\varepsilon^2$, $\forall t$
— Non autocorrélation : $\mathrm{Cov}(\varepsilon_t, \varepsilon_{t-h}) = 0$ $\forall t$,$\forall h > 0$

4.1.1 Composante tendancielle et saisonnière du modèle

4.1.1.1 Composante saisonnière

La forme de S_t dépend du type de données, et de la forme de la saisonnalité. On considérera ici les fonction S_t^j indicatrices,

$$S_t^j = \begin{cases} 1 & \text{si } t = \text{période } j \\ 0 & \text{sinon} \end{cases} \tag{4.4}$$

Par exemple, pour des données trimestrielles, on a : $S_t = \theta_1 S_t^1 + \theta_2 S_t^2 + \theta_3 S_t^3 + \theta_4 S_t^4$ où S_t^j est la fonction indicatrice du trimestre j.

4.1.1.2 Composante tendancielle

Cette composante a généralement une forme simple, reflétant la croissance moyenne. Par exemple, pour une tendance linéaire, $Z_t = \phi_1 + \phi_2 t$, on pose $Z_t^1 = 1$ et $Z_t^2 = t$.

Plusieurs types de composantes tendancielles existent. On peut citer :

1. Linéaire : $Z_t = \phi_1 + \phi_2 t$
2. Exponentielle : $Z_t = \alpha\beta^t$, ou $Z_t = \alpha(1+r)^t$ ou encore $Z_t = \alpha e^{rt}$
3. Quadratique : $Z_t = \phi_1 + \phi_2 t + \phi_3 t^2$

Le cas (1) se traite par régression linéaire, le cas (2) se ramène au cas (1) par transformation logarithmique et le cas (3) se traite par régression multiple. Il est également possible d'utiliser des modèles avec des ruptures :

$$Z_t = \begin{cases} \alpha_0 + \alpha_1 t & \text{pour } t \leq t_0 \\ \beta_0 + \beta_1 t & \text{pour } t > t_0 \end{cases} \tag{4.5}$$

Cette tendance est une des composantes les plus compliquées à modéliser car il n'existe pas vraiment de méthode.

4.1.2 Modèle mensuel de Buys-Ballot

La désaisonnalisation par régression linéaire, dans le cas où la tendance est supposée linéaire, et les données sont mensuelles, équivaut à tester le modèle linéaire suivant :

$$X_t = \underbrace{\phi_1 + \phi_2 t}_{Z_t} + \underbrace{\theta_1 S_t^1 + \theta_2 S_t^2 + \cdots + \theta_{12} S_t^{12}}_{S_t} + \varepsilon_t \tag{4.6}$$

où Z_t est la tendance (linéaire) et où S_t est la composante saisonnière.

Sous forme matricielle, le modèle s'écrit :

$$X = Y\Theta + \varepsilon \tag{4.7}$$

et l'écriture de l'estimateur des moindres carrés ordinaires s'écrit :

$$\widehat{\Theta} = \left(Y^{'}Y\right)^{-1} Y^{'}X \tag{4.8}$$

Toutefois, cette écriture n'est possible que si $Y^{'}Y$ est inversible, ce qui n'est pas le cas ici car la première colonne (correspondant à la constante) est égale à la somme des 12 dernières colonnes (les composantes mensuelles). Deux méthodes sont alors possibles pour faire malgré tout l'identification du modèle :

1. Ne pas tenir compte de la constante, et identifier le modèle :

$$X_t = \phi t + \theta_1 S_t^1 + \theta_2 S_t^2 + \cdots + \theta_{12} S_t^{12} + \varepsilon_t \tag{4.9}$$

2. Rajouter une contrainte, et identifier le modèle :

$$\begin{cases} X_t = \phi_1 + \phi_2 t + \theta_1 S_t^1 + \theta_2 S_t^2 + \cdots + \theta_{12} S_t^{12} + \varepsilon_t \\ \text{sous contrainte } \theta_1 + \theta_2 + \cdots + \theta_{12} = 0 \end{cases} \tag{4.10}$$

4.2 Estimateur des moindres carrés ordinaires

4.2.1 Solutions générales

On considère un modèle de la forme :

$$X_t = \sum_{i=1}^{p} \phi_i Z_t^i + \sum_{j=1}^{q} \theta_j S_t^j + \varepsilon_t, \quad \text{pour } t = 1, \dots, T \tag{4.11}$$

La méthode des moindres carrés ordinaires consiste à choisir les ϕ_i et θ_j de façon à minimiser le carré des erreurs, soit :

$$\begin{aligned} \left(\hat{\phi}_i, \hat{\theta}_j\right) &= \arg\min \left\{ \sum_{t=1}^{T} \varepsilon_t^2 \right\} \\ &= \arg\min \left\{ \sum_{t=1}^{T} \left[X_t - \sum_{i=1}^{p} \phi_i Z_t^i - \sum_{j=1}^{q} \theta_j S_t^j \right]^2 \right\} \end{aligned}$$

Notons par $\phi = \left(\phi_1, \dots, \phi_p\right)^{'}$ et $\theta = \left(\theta_1, \dots, \theta_q\right)^{'}$. La tendance Z et la composante saisonnière S se décomposent comme suit :

$$Z = \begin{bmatrix} | & & | \\ Z^1 & \cdots & Z^p \\ | & & | \end{bmatrix} = \left[Z_t^i\right]_{\substack{i=1,\dots,p \\ t=1,\dots,T}} \quad \text{et} \quad S = \begin{bmatrix} | & & | \\ S^1 & \cdots & S^q \\ | & & | \end{bmatrix} = \left[S_t^j\right]_{\substack{j=1,\dots,q \\ t=1,\dots,T}}$$

Le modèle s'écrit :

$$X = Z\phi + S\theta + \varepsilon = \begin{bmatrix} Z & S \end{bmatrix} \begin{bmatrix} \phi \\ \theta \end{bmatrix} + \varepsilon = Y\Theta + \varepsilon \tag{4.12}$$

et $\widehat{\Theta} = \left(\hat{\phi}, \hat{\theta}\right)^{'}$ vérifie alors l'équation

$$Y^{'}Y\widehat{\Theta} = Y^{'}X \quad \text{soit} \quad \begin{bmatrix} Z & S \end{bmatrix} \begin{bmatrix} Z^{'} \\ S^{'} \end{bmatrix} \widehat{\Theta} = \begin{bmatrix} Z^{'} \\ S^{'} \end{bmatrix} X$$

et donc

$$\begin{bmatrix} \hat{\phi} \\ \hat{\theta} \end{bmatrix} = \begin{bmatrix} Z^{'}Z & Z^{'}S \\ S^{'}Z & S^{'}S \end{bmatrix}^{-1} \begin{bmatrix} Z^{'}X \\ S^{'}X \end{bmatrix}$$

ce qui donne les coefficients :

$$\begin{cases} \hat{\phi} = \left[Z^{'}Z - Z^{'}S\left(S^{'}S\right)^{-1}S^{'}Z\right]^{-1}\left[Z^{'}X - Z^{'}S\left(S^{'}S\right)^{-1}S^{'}X\right] \\ \hat{\theta} = \left[S^{'}S - S^{'}Z\left(Z^{'}Z\right)^{-1}Z^{'}S\right]^{-1}\left[S^{'}X - S^{'}Z\left(Z^{'}Z\right)^{-1}Z^{'}X\right] \end{cases} \tag{4.13}$$

Remarque 4.1

S'il n'y a pas d'effet saisonnier, alors $X = Z\phi + \varepsilon$, et on retrouve le modèle linéaire usuel, avec pour estimateur des moindres carrés ordinaires $\hat{\phi} = [Z^{'}Z]^{-1}Z^{'}X$.

4.2.2 Cas particulier : le modèle trimestriel de Buys - Ballot

Considérons le modèle suivant :

$$X_t = \phi_1 + \phi_2 t + \theta_1 S_t^1 + \theta_2 S_t^2 + \theta_3 S_t^3 + \theta_4 S_t^4 + \varepsilon_t \tag{4.14}$$

Il est possible d'expliciter les différents coefficients. On résout le programme suivant :

$$\begin{cases} \min\limits_{\phi,\theta} \sum\limits_{t=1}^{T} \left[X_t - \phi_1 - \phi_2 t - \theta_1 S_t^1 - \theta_2 S_t^2 - \theta_3 S_t^3 - \theta_4 S_t^4\right]^2 \\ \text{sous contrainte } \theta_1 + \theta_2 + \theta_3 + \theta_4 = 0 \end{cases}$$

Ce programme peut être réécrit, en supprimant la constante :

$$\begin{cases} \min\limits_{\phi,\theta} \sum\limits_{t=1}^{T} \left[X_t - \phi_2 t - \sum\limits_{j=1}^{4} \varphi_j S_t^j\right]^2 \end{cases} \quad \text{où} \quad \begin{cases} \phi_1 = \dfrac{1}{4}\sum\limits_{j=1}^{4} \varphi_j \\ \theta_j = \varphi_j - \phi_1 \end{cases}$$

En notant N le nombre d'années entières ($N = T/4$), on pose :

- $\tilde{X}_n$: moyenne des X_t relatives à l'année n
- $\overline{X}_j$: Moyenne des X_t au trimestre j

— $\overline{X}$: moyenne de toutes les observations X_t

On a alors les estimateurs suivants :

$$\hat{\phi}_2 = \frac{3}{N\left(N^2-1\right)}\left[\sum_{n=1}^{n=N} n\widetilde{X}_n - \frac{N(N+1)}{2}\overline{X}\right] \tag{4.15}$$

$$\widehat{\varphi}_j = \overline{X}_j - [j+2(N-1)]\,\hat{\phi}_2 \quad \text{pour} \quad j=1,\ldots,4 \tag{4.16}$$

D'où finalement :

$$\begin{cases} \hat{\phi}_1 & = \dfrac{1}{4}\displaystyle\sum_{j=1}^{j=4}\widehat{\varphi}_j \\ \hat{\theta}_j & = \widehat{\varphi}_j - \hat{\phi}_1 \end{cases} \tag{4.17}$$

4.2.3 Généralisation des formules de Buys - Ballot

Les relations obtenues dans le cas précédant peuvent en fait être généralisées dans le cas d'une périodicité q et en notant N le nombre d'années entières. En supposant que la tendance est linéaire, le modèle s'écrit :

$$X_t = \phi_1 + \phi_2 t + \theta_1 S_t^1 + \cdots + \theta_q S_t^q + \varepsilon_t \tag{4.18}$$

On résout le programme

$$\begin{cases} \min\limits_{\phi,\theta} \displaystyle\sum_{t=1}^{T}\left[X_t - \phi_1 - \phi_2 t - \theta_1 S_t^1 - \cdots - \theta_q S_t^q\right]^2 \\ \text{sous contrainte } \theta_1 + \cdots + \theta_q = 0 \end{cases}$$

Ce programme admet pour solution, en notant :

$$\begin{cases} \hat{\phi}_2 & = \dfrac{12}{q} \times \dfrac{\displaystyle\sum_{n=1}^{N} n\widetilde{X}_n - \frac{N(N+1)}{2}\overline{X}}{N\left(N^2-1\right)} \\ \hat{\phi}_1 & = \overline{X} - \hat{\phi}_2 \dfrac{Nq+1}{2} \\ \hat{\theta}_j & = \overline{X}_j - \overline{X} - \hat{\phi}_2\left[j - \dfrac{q+1}{2}\right] \end{cases}$$

4.3 Application à la série Air passengers

Considérons le logarithme de la série Air passengers, infra-annuelle, représentée ci-dessous :

```
# Logarithme de la série infra-annuelle
import numpy as np
import seaborn as sns
flights = sns.load_dataset("flights")
```

```
data = flights.pivot(index="year",columns="month",values="passengers")
X = data.apply(lambda x : np.log(x),axis=0)
```

On calcule les moyennes $\tilde{X}_n$.

```
# Moyennes annuelles
Xtilde = X.mean(axis=1)
```

Table 4.1 – Moyennes annuelles

1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
4.84	4.93	5.13	5.28	5.41	5.47	5.64	5.78	5.9	5.93	6.05	6.15

et $\overline{X}_j$, $\forall i = 1, \dots, 12$.

```
# Moyennes mensuelles/ Moyennes par période
Xbar = X.mean(axis=0)
```

Table 4.2 – Moyennes mensuelles

Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
5.4	5.39	5.53	5.51	5.52	5.65	5.76	5.76	5.63	5.5	5.37	5.49

La moyenne générale $\overline{X}$ est :

```
# Moyenne générale
Xmean = X.mean().mean()
print('Moyenne générale : %.2f'%(Xmean))
```

```
## Moyenne générale : 5.54
```

Considérons alors un modèle de la forme suivante, avec une saisonnalité en 12 composantes (les données étant mensuelles : chaque composante correspond à un mois), et une tendance supposée linéaire ($Z_t = \phi_1 + \phi_2 t$) :

$$X_t = \phi_1 + \phi_2 t + \theta_1 S_t^1 + \theta_2 S_t^2 + \cdots + \theta_{12} S_t^{12} + \varepsilon_t \tag{4.19}$$

Compte tenu de la sur-identification de ce modèle, on rajoute la contrainte que la somme des θ_j soit nulle (c'est-à-dire que la composante saisonnière soit centrée : $\mathbb{E}(S_t) = 0$). On peut alors faire l'estimation de la fonction suivante :

1. On estime le modèle (4.9), c'est-dire sans contrainte, et sans constante ϕ_1
2. et on se ramène au modèle (4.10) en utilisant les relations.

Pour l'étape (1) deux méthodes analogues sont possibles : soit en utilisant les expressions des estimateurs, soit en effectuant la régression sous Python.

4.3.1 Calcul direct des estimateurs

Nous avons $N = 12$ (12 années d'observations) et la pente de la droite de la tendance est donnée par :

$$\hat{\phi}_2 = \frac{1}{N\left(N^2-1\right)}\left[\sum_{n=1}^{N} n\widetilde{X}_n - \frac{N(N+1)}{2}\overline{X}\right]$$

```
# Calcul de phi_{2}
def phi2_hat(N,q,xtilde,xmean):
    den = q*N*(N**2 -1)
    num = 12*(np.dot(np.arange(1,N+1),xtilde) - N*(N+1)*xmean/2)
    return num/den

# Application
phi2 = phi2_hat(12,12,Xtilde,Xmean)
print("phi2 : %.6f"%(phi2))
```

```
## phi2 : 0.010069
```

On estime la constante $\hat{\phi}_1 = \overline{X} - \hat{\phi}_2 \dfrac{Nq+1}{2}$:

```
# Calcul de phi_{1}
def phi1_hat(xmean,N,q,phi2hat):
    return xmean - phi2hat*(N*q+1)/2

# Application
phi1 = phi1_hat(Xmean,12,12,phi2)
print('phi1 : %.6f'%(phi1))
```

```
## phi1 : 4.812188
```

En utilisant les moyennes par mois, et la moyenne générale, on a les estimations de $\hat{\theta}_j$:

$$\hat{\theta}_j = \overline{X}_j - \overline{X} - \hat{\phi}_2\left[j - \frac{q+1}{2}\right], \quad \forall j = 1, \dots, 12 \tag{4.20}$$

```
# Calcul de theta
def theta_hat(xbar,xmean,phi2hat,p,q):
    return xbar - xmean - phi2hat*(p -(q+1)/2)

# Application
theta = np.array([theta_hat(Xbar[j-1],Xmean,phi2,j,12) for j in range(1,13)])
theta =  pd.DataFrame(theta, columns = ["coefficient"])
```

Ainsi le modèle s'écrit :

$$\widehat{X}_t = 4.8 + 0.01t - 0.085S_1^1 - 0.107S_t^2 + \cdots - 0.107S_t^{12} \tag{4.21}$$

On calcule les valeurs estimées de $\widehat{\varphi}_j$: $\widehat{\varphi}_j = \hat{\theta}_j + \hat{\phi}_1$, $\forall j = 1, \dots, 12$.

Table 4.3 – Coefficients $\hat{\theta}$

	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	$\hat{\theta}_4$	$\hat{\theta}_5$	$\hat{\theta}_6$	$\hat{\theta}_7$	$\hat{\theta}_8$	$\hat{\theta}_9$	$\hat{\theta}_{10}$	$\hat{\theta}_{11}$	$\hat{\theta}_{12}$
coefficient	-0.085	-0.107	0.023	-0.009	-0.011	0.111	0.215	0.206	0.061	-0.077	-0.221	-0.107

```
# Calcul de varphi
varphi = theta.add(phi1)
```

Table 4.4 – Coefficients $\hat{\varphi}$

	$\hat{\varphi}_1$	$\hat{\varphi}_2$	$\hat{\varphi}_3$	$\hat{\varphi}_4$	$\hat{\varphi}_5$	$\hat{\varphi}_6$	$\hat{\varphi}_7$	$\hat{\varphi}_8$	$\hat{\varphi}_9$	$\hat{\varphi}_{10}$	$\hat{\varphi}_{11}$	$\hat{\varphi}_{12}$
coefficient	4.727	4.705	4.835	4.804	4.801	4.923	5.027	5.018	4.873	4.735	4.592	4.705

4.3.2 Application sous Python

Ici, nous effectuons la régression du modèle contraint, en ne prenant pas en compte la constante.

$$X_t = \phi_2 t + \theta_1 S_t^1 + \theta_2 S_t^2 + \cdots + \theta_{12} S_t^{12} + \varepsilon_t \tag{4.22}$$

Tout d'abord, on crée les vecteurs de la composante saisonnière :

```
# creation du dataframe
df = pd.DataFrame(index = pd.date_range(start="1949-01-31",
                  end = "1960-12-31", freq="M"))
df['X'] = X.values.reshape((-1,1))
df['t'] = np.arange(1,145)

# Création de la composante saisonnière
A = np.identity(12)
for i in range(11):
    B = np.identity(12)
    A = np.append(A,B,axis=0)

for i in range(1,13):
    df['s'+str(i)] = A[:,i-1]
```

Ensuite, on effectue la régression linéaire sans constante :

```
# Estimation des paramètres
import statsmodels.formula.api as smf

formul = 'X~t+s1+s2+s3+s4+s5+s6+s7+s8+s9+s10+s11+s12-1'
model = smf.ols(formula = formul, data = df).fit()
print(model.summary2())
```

```
##                  Results: Ordinary least squares
## ================================================================
## Model:              OLS              Adj. R-squared:     0.982
```

```
## Dependent Variable: X                AIC:                -392.5952
## Date:               2023-08-06 22:40 BIC:                -353.9876
## No. Observations:   144              Log-Likelihood:     209.30
## Df Model:           12               F-statistic:        649.4
## Df Residuals:       131              Prob (F-statistic): 2.31e-110
## R-squared:          0.983            Scale:              0.0035169
## ---------------------------------------------------------------------
##          Coef.     Std.Err.       t        P>|t|     [0.025    0.975]
## ---------------------------------------------------------------------
## t        0.0101      0.0001     84.3990    0.0000    0.0098    0.0103
## s1       4.7268      0.0189    250.1797    0.0000    4.6894    4.7642
## s2       4.7047      0.0189    248.3449    0.0000    4.6672    4.7422
## s3       4.8350      0.0190    254.5287    0.0000    4.7974    4.8725
## s4       4.8037      0.0190    252.1922    0.0000    4.7660    4.8414
## s5       4.8013      0.0191    251.3734    0.0000    4.7635    4.8391
## s6       4.9235      0.0192    257.0523    0.0000    4.8856    4.9613
## s7       5.0274      0.0192    261.7440    0.0000    4.9894    5.0654
## s8       5.0181      0.0193    260.5224    0.0000    4.9800    5.0562
## s9       4.8735      0.0193    252.2934    0.0000    4.8353    4.9117
## s10      4.7353      0.0194    244.4382    0.0000    4.6970    4.7736
## s11      4.5916      0.0194    236.3348    0.0000    4.5532    4.6300
## s12      4.7055      0.0195    241.4909    0.0000    4.6669    4.7440
## ---------------------------------------------------------------------
## Omnibus:               2.692         Durbin-Watson:           0.425
## Prob(Omnibus):         0.260         Jarque-Bera (JB):        2.412
## Skew:                  -0.224        Prob(JB):                0.299
## Kurtosis:              2.551         Condition No.:           584
## =====================================================================
## Notes:
## [1] Standard Errors assume that the covariance matrix of the
## errors is correctly specified.
```

On calcule la constante $\hat{\phi}_1$:

```
# Constante
phi1_hat = model.params[1:].mean()
print('phi1 hat : %.6f'%(phi1_hat))
```

```
## phi1 hat : 4.812188
```

La lecture de la sortie donne effectivement les mêmes résultats numériques que les calculs présentés ci-dessus. Comme le montre la sortie ci-dessus, tous les paramètres sont significatifs, le R^2 est bon (98%), la statistique de Fisher F est suffisamment grande pour valider le modèle.

```
# Composante tendancielle
T = phi1_hat+model.params[0]*df['t']
# Représentation graphique
import matplotlib.pyplot as plt
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(df.X,color='black');
axe.plot(T,color='red');
```

```
axe.set_xlabel('mois');
axe.set_ylabel('valeur');
plt.show()
```

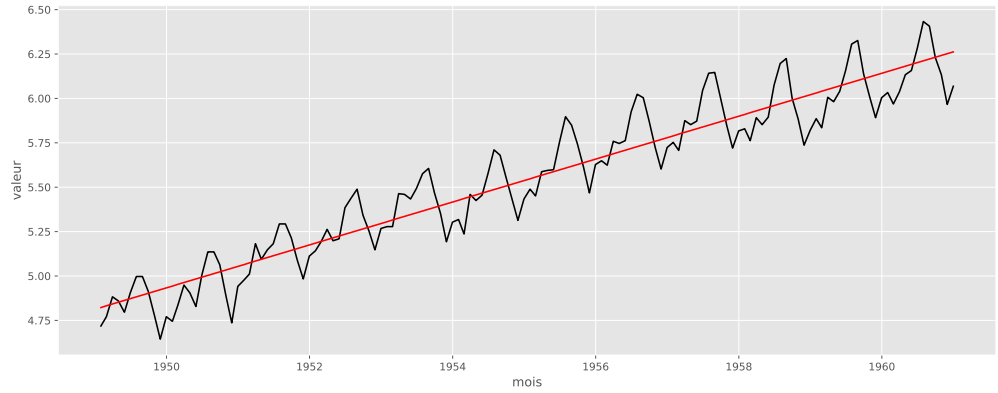


Figure 4.1 – Séries Air passengers

Visualisons les résidus d'estimation

```
# Représentation graphique des residus
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(model.resid,color='black');
axe.set_xlabel('mois');
axe.set_ylabel('résidu');
plt.show()
```

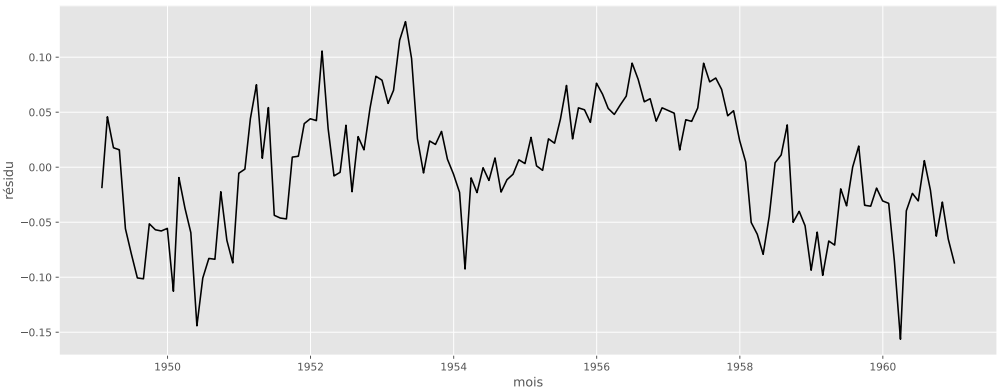


Figure 4.2 – Residual plot

```
# Durbin watson
from statsmodels.stats.stattools import durbin_watson
dw = durbin_watson(model.resid)
print('Durbin-Watson statistic :%.3f'%(dw))
```

```
## Durbin-Watson statistic :0.425
```

La série corrigée des valeurs saisonnières correspond à la série :

$$X_t^{\text{cvs}} = X_t - \widehat{S}_t = \sum_{i=1}^{p} \phi_i Z_t^i + \varepsilon_t$$

```
# Série corrigée des variations saisonnières
c = model.params[1:] - model.params[1:].mean()
Xcvs = pd.DataFrame(df.X.values -np.dot(A,c),index=df.index)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(np.exp(df.X),color='black',label = "ln(air passengers)");
axe.plot(np.exp(Xcvs),color='red', label = "ln(air passengers) cvs");
axe.set_xlabel('mois');
axe.set_ylabel('valeur');
axe.legend();
plt.show()
```

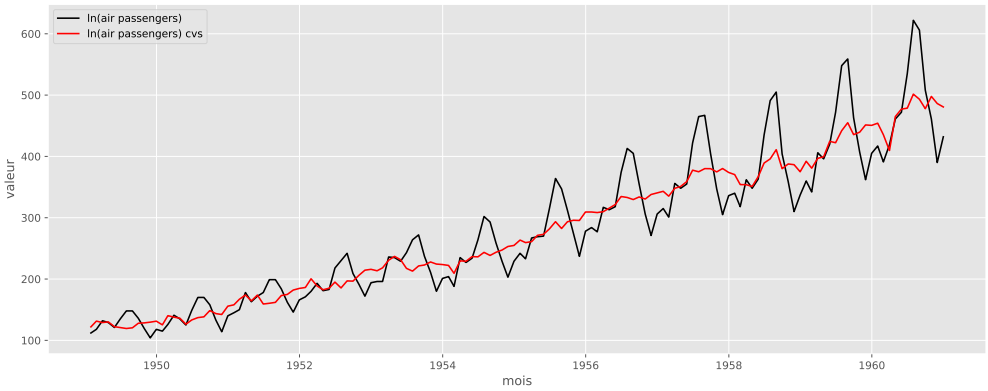


Figure 4.3 – Air passengers (CVS)

4.3.3 Propriétés des estimateurs et prévision

4.3.3.1 Propriétés statistiques des estimateurs

Sous l'hypothèse $\mathbb{E}(\varepsilon_t) = 0$, les estimateurs des moindres carrés ordinaires sont sans biais :

$$\begin{cases} \mathbb{E}\left(\hat{\phi}_i\right) & = \phi_i \quad \forall i \\ \mathbb{E}\left(\hat{\theta}_j\right) & = \theta_j \quad \forall j \end{cases}$$

La variance des estimateurs peut être estimée par :

$$\widehat{\mathrm{V}}\begin{pmatrix}\hat{\phi}\\ \hat{\theta}\end{pmatrix} = \hat{\sigma}_\varepsilon^2 \begin{bmatrix} Z'Z & Z'S \\ S'Z & S'S \end{bmatrix}^{-1}, \quad \text{où} \quad \hat{\sigma}_\varepsilon^2 = \frac{1}{T-p-q}\sum_{t=1}^{T}\hat{\varepsilon}_t^2$$

ce qui permet d'obtenir des intervalles de confiance sur les estimateurs.

4.3.3.2 Prévision à un horizon h

Soit $h \geq 1$. On suppose que le modèle reste valide en $T+h$, c'est-à-dire que

$$X_{T+h} = \sum_{i=1}^{p} \phi_i Z_{T+h}^i + \sum_{j=1}^{q} \theta_j S_{T+h}^j + \varepsilon_{T+h} \tag{4.23}$$

avec $\mathbb{E}(\varepsilon_{T+h}) = 0$, $\mathrm{V}(\varepsilon_{T+h}) = \sigma_\varepsilon^2$ et $\mathrm{Cov}(\varepsilon_t, \varepsilon_{T+h}) = 0$ pour $t = 1, \ldots, T$. La variable X_{T+h} peut être approchée par :

$$\widehat{X}_T(h) = \sum_{i=1}^{p} \hat{\phi}_i Z_{T+h}^i + \sum_{j=1}^{q} \hat{\theta}_j S_{T+h}^j \tag{4.24}$$

Cette prévision est la meilleure (au sens de l'erreur quadratique moyenne) prévision linéaire en $X_1, \ldots, X_T$ et sans biais. Un intervalle de confiance de cette prévision est de la forme :

$$\left[\widehat{X}_T(h) - t_{1-\alpha/2}\sqrt{\hat{e}_h}; \widehat{X}_T(h) + t_{1-\alpha/2}\sqrt{\hat{e}_h}\right] \tag{4.25}$$

où $t_{1-\alpha/2}$ est le quantile d'ordre α de la loi de Student à $T-p-q$ degrés de liberté, et où :

$$\begin{aligned}
\hat{e}_h &= \hat{\mathbb{E}}\left(\left[\widehat{X}_T(h) - X_{T+h}\right]^2\right) = \widehat{\mathrm{V}}\left(\sum_{i=1}^{p} \hat{\phi}_i Z_{T+h}^i + \sum_{j=1}^{q} \hat{\theta}_j S_{T+h}^j - \varepsilon_{T+h}\right) \\
&= \begin{bmatrix} \hat{\phi}' & \hat{\theta}' \end{bmatrix} \left[\widehat{V}\begin{pmatrix} \hat{\phi} \\ \hat{\theta} \end{pmatrix}\right] \begin{bmatrix} \hat{\phi} \\ \hat{\theta} \end{bmatrix} + \hat{\sigma}_\varepsilon^2
\end{aligned}$$

5 Désaisonnalisation par moyennes mobiles

Sommaire

La méthode de lissage par la moyenne mobile est la méthode non paramétrique qui permet de reproduire le plus fidèlement possible la tendance tout en éliminant le résidu. L'avantage de cette méthode est qu'elle ne fait pas d'hypothèse sur la tendance qui peut avoir une forme quelconque. Elle est simple dans les calculs, permet une mise à jour facile et surtout réagit aux changements de régime de la série.

5.1 Les moyennes mobiles

5.1.1 Notion de filtre

5.1.1.1 Filtre linéaire

Définition 5.1 *Opérateur de convolution*

Soit $\mathcal{S}$ l'espace des séries stationnaires à moyenne nulle, ou l'espace des séries déterministes avec norme $\mathbb{L}^2$ fini. Soit ψ une suite telle que $\sum_{i\geq 0}\psi_i^2 < +\infty$. Un operateur de la forme $F_\psi : \mathcal{S} \longrightarrow \mathcal{S}$ défini par

$$Y_t = F_\psi(X_t) = \sum_{i=0}^{+\infty} \psi_i X_{t-i} \tag{5.1}$$

est appelé opérateur de convolution ou filtre [1], et denoté par $\psi(B)$.

Remarque 5.1

Il est facile de voir que les opérations de filtrages commutent, i.e. que

1. Voir http ://www.songho.ca/dsp/convolution/convolution.html

$$F_{\psi_1} F_{\psi_2} = F_{\psi_2} F_{\psi_1} = F_{\psi_1 * \psi_2} \tag{5.2}$$

En effet, l'ensemble des filtres est isomorphe au ensemble des fonctions complexes $\psi(z)$, l'isomorphisme étant « la transformé z » des suites. Cet isomorphisme explique quelques manipulations formelles avec les filtres, mais pas celles liés à l'inversion.

Une question de porté pratique pour un filtre c'est la déterminantion de son noyau et son espace invariant.

Theorème 5.1 *1. Un filtre conserve les polynômes de degré $\leq p$ si et seulement si 1 est une racine d'ordre au moins p de l'équation $\psi(z) = 1$;*

2. Un filtre enlève les composantes périodiques d'ordre p si et seulement si $\psi(z)$ est divisible par $1 + z + \cdots + z^{p-1}$ (donc si $\psi(z) = 0$, pour toutes les racines d'ordre p de l'unité, sauf $z = 1$).

5.1.1.2 Investion des filtres

Une autre question importante est celle liée à l'inversion des filtres.

Theorème 5.2 *1. Pour un polynôme $\psi(z) = \prod_{i=1}^{i=p} (1 - z/z_i)$ qui a toutes ses racines z_i à l'extérieur du cercle unitaire $|z| \leq 1$, $\frac{1}{\psi(z)}$ a un développement en série de Taylor*

$$\frac{1}{\psi(z)} = \sum_{n=0}^{+\infty} \psi_n z^n \tag{5.3}$$

qui est convergente à l'intérieur du cercle unitaire $|z| = 1$. Dans le cas le plus simple des racines z_i distinctes, on a

$$\psi_n = \sum_{i=1}^{p} \frac{K_i}{z_i^{n+1}} \tag{5.4}$$

où $K_i = -\frac{1}{\psi'(z_i)}$ où ψ' est la dérivée première de ψ par rapport à z_i. Dans le cas des racines confondues, on a des formules similaires qui utilisent dérivées de degré supérieur.

2. Pour un polynôme $\psi(z) = \prod_{i=1}^{i=p} (1 - z/z_i)$ qui a toutes ses racines z_i à l'intérieur du cercle unitaire $|z| \leq 1$, $\frac{1}{\psi(z)}$ a un développement en série de Taylor

$$\frac{1}{\psi(z)} = \sum_{n=-1}^{-\infty} \psi_n z^n \tag{5.5}$$

qui est convergente sur le cercle unitaire $|z| = 1$. Dans le cas le plus simple des racines z_i distinctes, on a

$$\psi_n = -\sum_{i=1}^{p} K_i z_i^{n+1} \tag{5.6}$$

où $K_i = -\dfrac{1}{\psi'(z_i)}$ *où* ψ' *est la dérivée première de* ψ *par rapport à* z_i.

3. *Dans le cas mixte avec racines à l'extérieur et aussi à l'extérieur du cercle unitaire, on a un mélange des formules ci - dessus.*

Et sous python, une des nombreuses alternatives est la fonction np.convolve. L'opération de convolution discret y est définie comme suit :

$$(\psi * X)_t = \sum_{i=-\infty}^{+\infty} \psi_i X_{t-i} \tag{5.7}$$

Le plus difficile dans cette opération de convolution est de trouver la bonne suite ψ.

Exemple 5.1 *Application à la série elecequip*

On considère la série elecequip contenue dans le package fpp2 de R. Cette série représente la fabrication mensuelle de matériel électrique : produits informatiques, électroniques et optiques de janvier 1996 à mars 2012. Les données ont été corrigées des jours ouvrables. Nous supprimons l'année 2012

```
# Chargement des données
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
elec = sm.datasets.get_rdataset("elecequip", "fpp2").data.drop("time",axis=1)
elecequip = elec.rename(columns = {"value" : "elecequip"})
elecequip.index = pd.date_range("1996-01-31",periods=len(elecequip),freq = "M")
elecequip = elecequip[elecequip.index<="2011-12-31"]
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(elecequip,color = "black");
axe.set_xlabel("année");
axe.set_ylabel("Indice des nouvelles commandes");
plt.show()
```

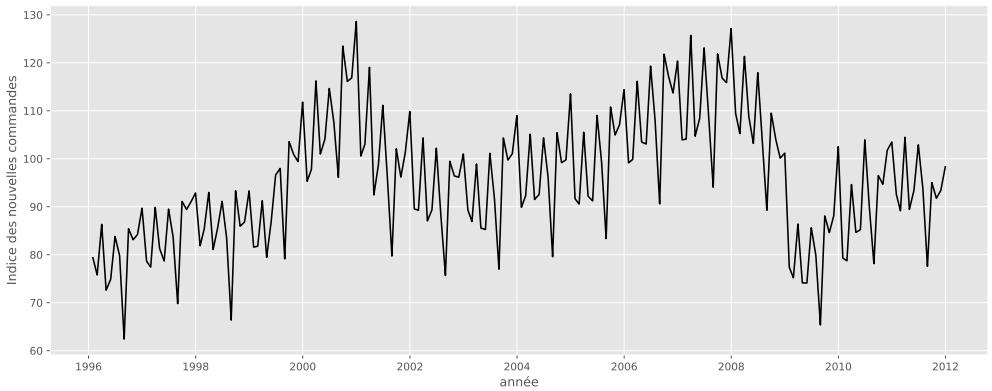


Figure 5.1 – Fabrication d'équipements électriques (zone euro)

La saisonnalité est de période $p = 12$, i.e. $m = 6$. Nous avons la formule de convolution suivante

$$Y_t = \sum_{i=-6}^{i=6} \psi_i X_{t-i} \tag{5.8}$$

avec $\psi_{-6} = \psi_6 = \frac{1}{24}$ et $\psi_i = \frac{1}{12}, i = -5, \ldots, 5.$

```
# Création de la fonction ConvolveFilter
def ConvolveFilter(x,psi,two_sided=True):
  if len(psi)%2 == 0:
    p = len(psi)
  else:
    p = len(psi) - 1
  if two_sided:
    m = p/2
    y = np.convolve(a = psi,v = x,mode = "valid")
    y = np.hstack((np.hstack((np.zeros(int(m))+np.nan,y)),
                   np.zeros(int(m))+np.nan))
  else:
    y = np.convolve(a = psi,v = x,mode = "full")
    y = np.hstack((np.zeros(p)+np.nan,y[p:-p:1]))
  y = pd.Series(data=y,index=x.index,dtype=np.float64,name="trend")
  return y


# Filtre de convolution
theta = list([1/24,1/12,1/12,1/12,1/12,1/12,1/12,1/12,1/12,1/12,1/12,1/12,1/24])
convolvefilter = ConvolveFilter(elecequip.elecequip,theta,two_sided=True)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(elecequip.elecequip,color="black",label="$X_{t}$");
axe.plot(convolvefilter,color ="red",linestyle="--",label="$\widehat{X}_{t}$");
axe.set_xlabel("mois");
axe.set_ylabel("Indice des nouvelles commandes");
axe.legend();
plt.show()
```

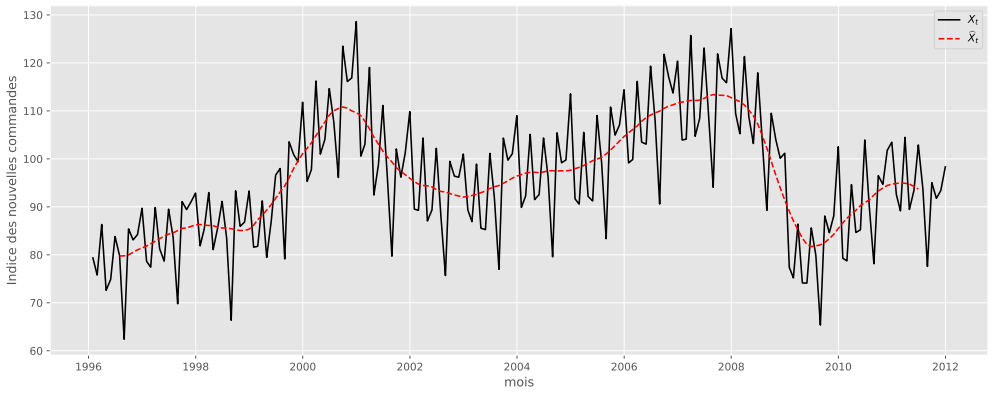


Figure 5.2 – Estimation de l'indice des commandes de matériel électrique par filtre de convolution (filtre linéaire)

Remarque 5.2

On peut également se servir de la fonction convolution_filter de la librairie Statsmodels. Elle est plus simple et plus réaliste par rapport la fonction np.convolve de numpy. On peut ainsi réadapter notre fonction `ConvolveFilter` de façon à tenir compte de la période de la chronique.

```
# Filter linéaire par convolution
from statsmodels.tsa.filters.filtertools import convolution_filter
def ConvolveFilter2(x,weighted=None,period=None,two_sided=True):
  if weighted is None:
    if period % 2 == 0:
      weighted = np.array([0.5] + [1] * (period - 1) + [0.5]) / period
    else:
      weighted = np.repeat(1.0 / period, period)
  nsides = int(two_sided) + 1
  result = convolution_filter(x, weighted, nsides)
  return result
```

5.1.1.3 Filtrage récursif

Les filtres récursifs sont définis par une équation de récurrence. Le filtre est spécifié par :

$$Y_t = X_t + \sum_{i=1}^{p} \psi_i Y_{t-i} \tag{5.9}$$

où $\psi_i, i = 1, \dots, p$ sont des constantes réelles.

Le paramètre p est déterminé grâce à la fonction d'autocorrélation de X.

Sous Python, la fonction recursive_filter de Statsmodels permet d'ajuster une série temporelle en utilisant le filtrage récursif. Dans cette fonction, la formule du filtrage récursif est définie comme suit :

$$Y_t = \psi_1 Y_{t-1} + \psi_2 Y_{t-2} + \cdots + \psi_p Y_{t-p} + X_t \tag{5.10}$$

Exemple 5.2 *Application du filtre récursif sur la série elecequip*

On considère la série elecequip. On souhaite appliquer une filtrage récursif en prenant $p = 12$ avec $\psi_i = (1/10)^i, i = 1, \dots, 12$.

```
# Filtrage récursif
from statsmodels.tsa.filters.filtertools import recursive_filter
psi = np.array([(1/10)**i for i in range(1,13)])
recursivefilter = recursive_filter(x = elecequip.elecequip, ar_coeff=psi)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(elecequip.elecequip,color="black",label="$X_{t}$");
axe.plot(recursivefilter,color ="red",linestyle="--",label="$\widehat{X}_{t}$");
axe.set_xlabel("mois");
axe.set_ylabel("Indice des nouvelles commandes");
axe.legend();
plt.show()
```

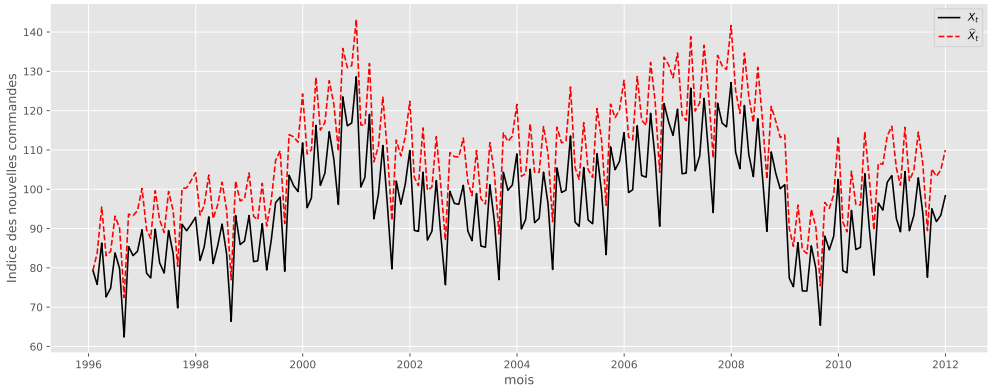


Figure 5.3 – Estimation de l'indice des commandes de matériel électrique par filtre récursif

5.1.1.4 Filtres de Hodrick-Prescott et de King - Baxter

Les filtres de Hodrick and Prescott (1997) et de R. G. King and Baxter (1995) sont des outils classiques de la décomposition cycle - tendance. Cependant, deux principales questions sont posées par rapport à ces filtres (voir Ahamada and Jolivaldt (2010)). La première question est celle de la performance comparative entre les deux filtres. Guay and St.-Amant (2005) conclut que les deux filtres donnent des résultats sérieusement biaisés dans l'hypothèse selon laquelle lacomposante cyclique théorique serait concentrée sur les basses fréquences. Or selon Granger (1966), les composantes cycliques dans la plupart des séries macroéconomiques sont de très basses fréquences. Il apparaît donc que les deux filtres donneraient plus souvent des résultats altérés lorsqu'ils sont confrontés aux observations macroéconomiques.

La deuxième question, moins évidente que la précédente, est la suivante : quels sont les comportements de ces filtres dans un environnement instable, plus précisément dans une situation où la composante cyclique est instable ? La question est justifiée au regard du phénomène de la grande modération (Stock and Watson (2005)), qui implique de manière sous - jacente une modification du cyle dans som amplitude et ou dans sa périodicité.

Définition 5.2 *Filtre HP*

Le filtre HP est conçu pour décomposer de manière additive une série temporelle X_t en deux composantes [2] :

$$X_t = C_t + Z_t \tag{5.11}$$

où C_t (respectivement Z_t) représente la composante cyclique (respectivement la composante tendancielle).

Le principe du filtre HP est un compromis entre la régularité de la composante tendancielle et la minimisation de la variance de la composante cyclique. Plus précisement, la composante Z_t est obtenue en minimisant la variance de C_t sous contrainte de pénalité de la dérivée seconde de Z_t :

$$\{Z_t\}_{t=1}^{t=T} = \arg\min \sum_{t=1}^{t=T} \left[(C_t)^2 + \lambda \left[(Z_{t+1} - Z_t) - (Z_t - Z_{t-1}) \right]^2 \right] \tag{5.12}$$

2. Voir https ://www0.gsb.columbia.edu/faculty/rhodrick/prescott-hodrick1997.pdf

La paramètre λ est un facteur de pénalité permettant de contrôler le lissage de Z_t : une valeur plus élevée de λ donnera une tendance plus linéaire et une composante cyclique plus fluctuante, et inversement. Hodrick and Prescott (1997) suggèrent une valeur de $\lambda = 1600$ pour des observations trimestrielles.

R. G. King and Rebelo (1993) montrent que le filtre HP peut stationnariser les processus non stationnaires intégrés jusqu'à l'ordre quatre. Singleton (1988) montre que le filtre HP est une bonne approximation d'un filtre passe - haut lorsqu'il est appliqué à une série stationnaire. Pour mieux comprendre cela, rappelons que toute série temporelle stationnaire est une combinaison linéaire de composantes cycliques de périodes comprises dans l'intervalle $[-\pi, \pi]$. Les conclusions de Singleton (1988) assurent que le filtre HP, lorsqu'il est appliqué à un processus stationnaire, permet d'obtenir la composante du cycle économique en supprimant les basses fréquences qui composent la série étudiée (les basses fréquences caractérisent la tendance Z_t).

Sous Python, la fonction hpfilter permet de lisser la tendance par la méthode du filtre de Hodrick and Prescott (1997). Soit $(X_t)_t$ une série temporelle, le programme d'optimisation est le suivant :

$$\min \sum_{t=1}^{t=T} \left[(X_t - Z_t)^2 + \lambda \left[(Z_{t+1} - Z_t) - (Z_t - Z_{t-1}) \right]^2 \right] \tag{5.13}$$

Sous Statsmodels, l'implémentation du filtre HP s'est faite en tant que règle de régression ridge à l'aide de scipy.spare. Dans ce sens, la solution peut s'écrire (voir Ravn and Uhlig (2002)) :

$$Z = \left(I_T + \lambda A^{'} A \right)^{-1} \tag{5.14}$$

où I_T est la matrice identité d'ordre T et A une matrice de format $((T-2) \times T)$ telle que

$$a_{i,j} = \begin{cases} 1 & \text{si} \quad i = j \quad \text{ou} \quad i = j+2 \\ -2 & \text{si} \quad i = j+1 \\ 0 & \text{sinon} \end{cases} \tag{5.15}$$

Exemple 5.3 *Application du filtre HP*

On considère les données macroéconomiques macrodata contenues dans la librairie Statsmodels. Ces données sont trimestrielles, ce qui justifie de fixer le paramètre λ à 1600.

```
# Chargement des données
dta = sm.datasets.macrodata.load_pandas().data
dta.index = pd.date_range(start='1959-01-31',periods=len(dta),freq='3M')
# Filtre HP
hpcycle, hptrend = sm.tsa.filters.hpfilter(x = dta.realgdp, lamb = 1600)
gdp_decomp = dta[['realgdp']]
gdp_decomp["cycle"] = hpcycle
gdp_decomp["trend"] = hptrend
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(gdp_decomp["realgdp"]["2000-03-31":],color="black",label = "$X_{t}$");
axe.plot(gdp_decomp["trend"]["2000-03-31":],color = "red",
         label="$\widehat{X}_{t}$",linestyle = "--");
axe.set_xlabel("année");
axe.set_ylabel("produit intérieur brut réel");
```

```
axe.legend();
plt.show()
```

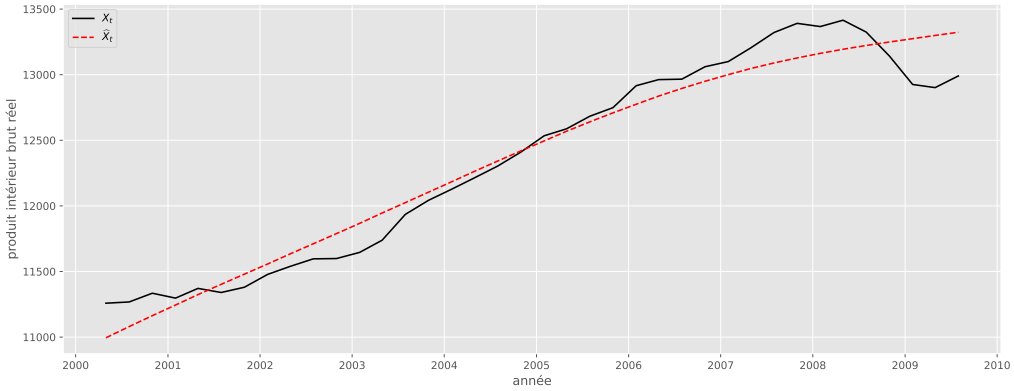


Figure 5.4 – Estimation du produit intérieur bruit réel par le filtre HP.

Définition 5.3 *Filtre BK*

Le filtre HP est une approximation d'un filtre passe - bande, c'est - à - dire ne laissant qu'une bande de fréquences entre une haute et une basse fréquences. Si on se réfère à Burns and Mitchell (1946a), les composantes du cycle économique se localisent dans une bande de fréquences qui correspond à une bande de périodicités comprise entre 6 trimestres et 32 trimestres. Cette définition suggère de supprimer les fréquences les plus hautes et les plus basses qui composent la série étudiée. Il faut donc extraire la composante du cycle économique sur une bande de fréquences bien définie. Cette approche a été adaptée par R. G. King and Baxter (1995). Appliqué sur données trimestrielles, le filtre BK extrait la composante cyclique C_t de la manière suivante [3] :

$$C_t = \sum_{j=-12}^{j=12} a_j X_{t-j} = a(B)X_t \tag{5.16}$$

où B est l'opérateur retard. La composante cyclique prend alors la forme d'une moyenne mobile. Les coefficients $\{a_j\}$ résultent du problème de minimisation suivant :

$$\min_{a_j} \mathcal{Q} = \int_{-\pi}^{\pi} |\beta(\omega) - \alpha(\omega)|^2 \mathrm{d}\omega \quad \text{et} \quad \alpha(0) = 0 \tag{5.17}$$

où $\alpha(\omega)$ est la transformée de Fourrier des $\{a_j\}$, c'est - à - dire $\alpha(\omega) = \sum_{j=-12}^{j=12} a_j \mathrm{e}^{-i\omega j}$.

$|\beta(\omega)|$ est le gain du filtre idéal, c'est - à - dire que $|\beta(\omega)| = I(\omega \in [\omega_1, \omega_2])$ avec $I(.)$ la fonction indicatrice valant 1 si l'argument est vrai et 0 sinon. La bande de fréquences $[\omega_1, \omega_2]$ délimite les composantes dont la périodicité est comprise entre 6 et 32 trimestres, $\omega_1 = \pi/12$ et $\omega_2 = \pi/3$ (voir Burns and Mitchell (1946b)). La contrainte $\alpha(0) = 0$ sert à isoler toute tendance dans C_t.

Sous Python, la fonction bkfilter de Statsmodels filtre la série en utilisant la bande passante de R. G. King and Baxter (1995). Cette fonction renvoie une moyenne mobile pondérée centrée de la série d'origine. Où les poids a_j sont calculés comme suit :

3. Voir http ://www.nber.org/papers/w5022.pdf

$$b_j = \begin{cases} \dfrac{(\sin(\omega_2 j) - \sin(\omega_1 j))}{\pi j} & \text{pour} \quad |j| \geq 1 \\ \dfrac{\omega_2 - \omega_1}{\pi} & \text{pour} \quad j = 0 \end{cases} \tag{5.18}$$

$$a_j = b_j + \theta, \quad \text{pour} \quad \forall j = -K, \ldots, K \tag{5.19}$$

avec K la longueur d'avance-retard du filtre : $K = 12$ (respectivement $K = 3$) pour les données trimestrielles (respectivement les données annuelles). θ est une constante normalisée :

$$\theta = -\frac{1}{2K+1} \sum_{j=-K}^{j=K} b_j \tag{5.20}$$

avec $\omega_1 = \dfrac{2\pi}{P_H}$ et $\omega_2 = \dfrac{2\pi}{P_L}$ où P_H (respectivement P_L) représente la périodicité des fréquences de coupure haute (respectivement basse).

Exemple 5.4 *Application du filtre BK*

On considère les données sur l'investissement réel des Etats - Unis contenues dans la base `macrodata` disponible sous Statsmodels. On applique le filtre BK en fixant la bande inférieureà 6, la bande supérieure à 24. Puisque les données sont trimestrielles, alors $K = 12$.

```
# Filtre BK
cycles = sm.tsa.filters.bkfilter(x=dta[['realinv']],low=6,high=24,K=12)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(cycles, color = "black", label = "$C_{t}$");
axe.set_xlabel("année");
axe.set_ylabel("investissement réel (cycle)");
axe.legend();
plt.show()
```

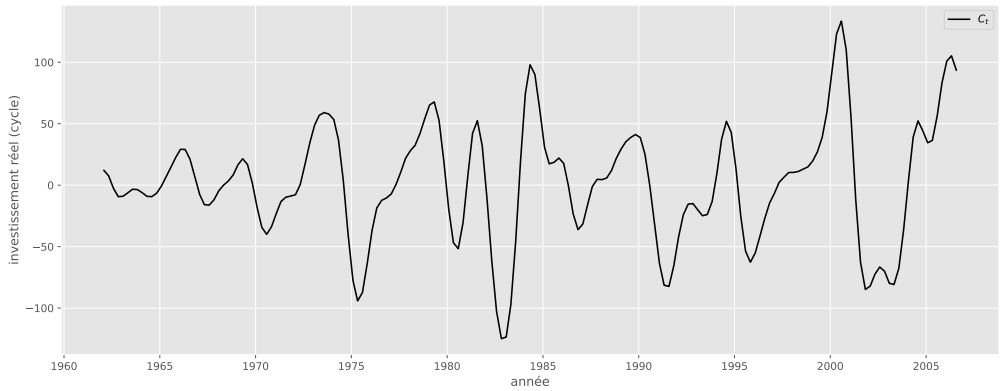


Figure 5.5 – Evolution du cycle de la série investissement réel (Filtre BK)

5.1.1.5 Filtre de passe-bande de Christiano et Fitzgerald

Christiano and Fitzgerald (2003) ont proposé une approximation finie et optimale à passe - bande dans le but d'extraire les mouvements cycliques qui, d'après eux sont des périodes de récurrences dans un intervalle $[\omega_1, \omega_2]$.

Le critère d'optimalité retenu par Christiano and Fitzgerald (2003) pour approximer le filtre infini par un filtre fini consiste à minimiser l'espérance de l'erreur quadratique $\mathbb{E}\left[(Y_t - Y_t^*)^2 | X_1, \dots, X_T\right]$. Cette erreur est mesurée entre Y_t issue du filtre idéal et Y_t^* issue du filtre approximé, pour chaque t. On détermine ainsi un filtre optimal pour chaque observation de la série considérée.

Par construction, puisqu'on cherche un filtre linéaire, Y_t^* appartient au sous - espace engendré par les X_t. La détermination de ce filtre dans le cas général est complexe. Le filtre obtenu dépend de l'ensemble de la série considérée et il varie d'une observation à l'autre. Ainsi, un tel filtre n'est pas linéaire par rapport aux séries, et les liens entre les filtres déterminés pour divers intervalles de fréquences semblent, eux aussi, complexes.

Définition 5.4 *Filtre de passe-bande de Christiano et Fitzgerald*

Soit $(X_t)_t$ une chronique, le filtre de passe - bande de Christiano et Fitzgerald appliqué à X_t est :

$$Y_t^* = B_0 X_t + B_1 X_{t+1} + \cdots + B_{T-1} X_{T-1} + \widetilde{B}_{T-t} X_T + B_1 X_{t-1} + \cdots + B_{t-2} X_2 + \widetilde{B}_{t-1} X_1 \quad (5.21)$$

pour $t = 3, 4, \dots, T-2$, avec

$$B_j = \begin{cases} \dfrac{\sin(j\omega_2) - \sin(j\omega_1)}{\pi j} & \text{pour} \quad j \geq 1 \\ \dfrac{\omega_2 - \omega_1}{\pi} & \text{pour} \quad j = 0 \end{cases}$$

$$\omega_1 = \frac{2\pi}{P_U}, \quad \omega_2 = \frac{2\pi}{P_L}$$

$\widetilde{B}_{T-t}$ et $\widetilde{B}_{t-1}$ sont des fonctions linéaires de B_j. Les valeurs à $t = 1, 2, T-1$ et T sont également calculés de la même manière. P_U et P_L ont les mêmes interprétations que P_H et P_L.

Remarque 5.3

Le filtre de passe-bande de Christiano and Fitzgerald (2003) est approprié pour les séries qui peuvent suivre une marche aléatoire.

Exemple 5.5 *Application du filtre CF*

On considère les séries de taux de chômage et d'inflation des Etats - Unis contenues dans la base `macrodata` disponible sous Statsmodels. Sous Python, la fonction cffilter applique le filtre de Christiano and Fitzgerald (2003) à la chronique X_t. Par défaut $P_L = 6$ et $P_U = 36$.

```
# Filtre CF
from statsmodels.tsa.filters.cf_filter import cffilter
cf_cycles, cf_trend = cffilter(dta[["infl", "unemp"]],low=6,high=32,drift=True)
# Représentation graphique
fig, axe = plt.subplots(2,1,figsize=(16,8))
label = ["infl","unemp"]
for idx, name in enumerate(label):
  axe[idx].plot(dta[name],color = "black",label = f"{name}");
  axe[idx].plot(cf_trend[f"{name}_trend"],color="red",label = f"{name}_trend");
  axe[idx].set_xlabel("Date");
  axe[idx].set_ylabel(f"{name}");
  axe[idx].legend();
plt.tight_layout()
plt.show()
```

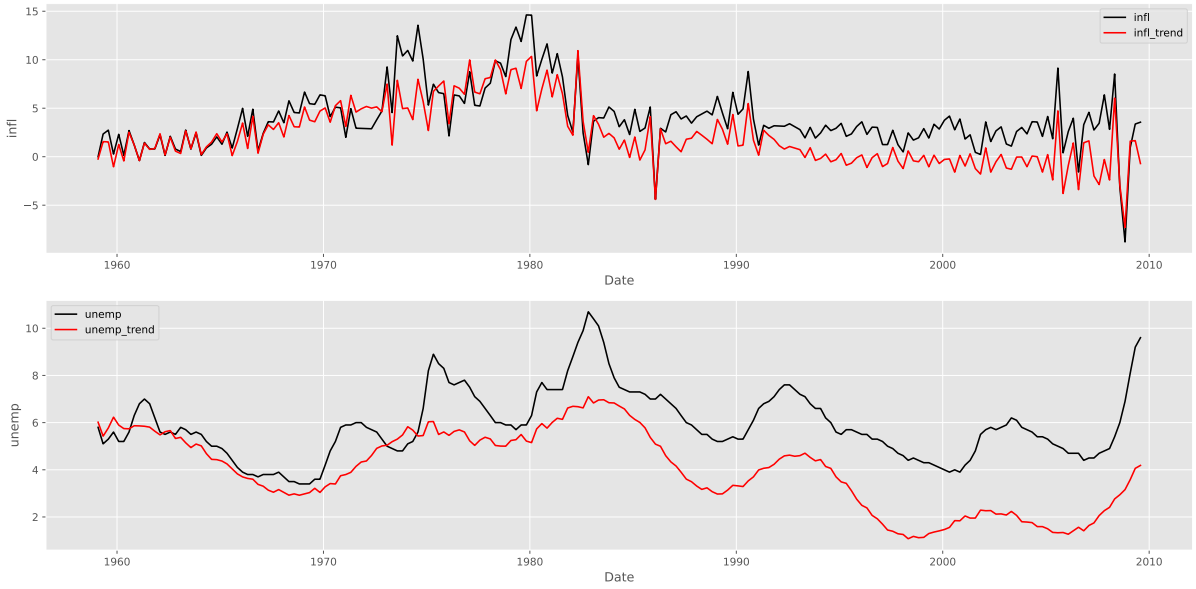


Figure 5.6 – Application du filtre CF sur les séries de taux de chômage et d'inflation.

```
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
cf_cycles.plot(ax=axe, style=['r--', 'b-']);
plt.show()
```

5.1.2 Définitions des moyennes mobiles

Elles font partie des premières méthodes pour l'analyse des séries temporelles. D'ailleurs, Poynting (1884) est le premier à l'avoir utilisé pour éliminer les variations accidentelles ou périodiques d'une série.

Définition 5.5 *Moyenne mobile*

Soit X_t une série chronologique, $m_1, m_2 \in \mathbb{N}$. On appelle moyenne mobile de longueur (d'ordre) $m_1 + m_2 + 1$ de X_t, une transformation de X_t, notée Y_t ou $M_{m_1+m_1+1}(X_t)$, s'écrivant comme

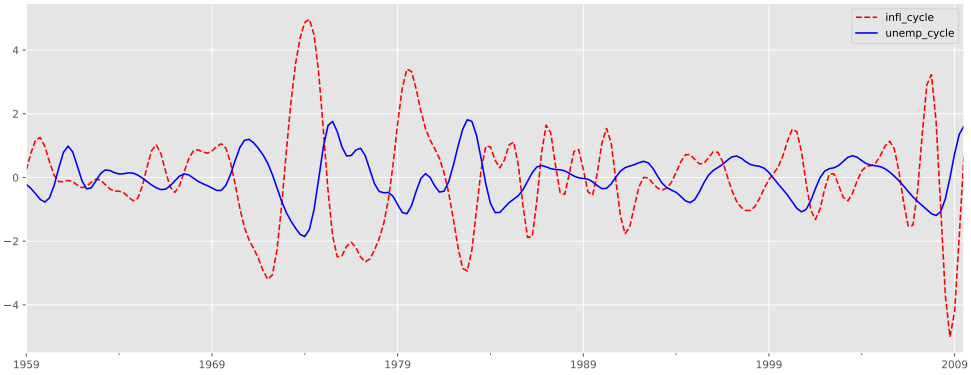


Figure 5.7 – Evolution du cycle des séries de taux de chômage et d'inflation calculés (filtre CF)

combinaison linéaire finie des valeurs de la série correspondant à des dates entourant t. La série transformée s'écrit alors :

$$Y_t = M_{m_1+m_1+1}(X_t) = \sum_{i=-m_1}^{i=m_2} \theta_i X_{t+i}, \quad t \in \{1, 2, ..., T\} \tag{5.22}$$

avec $\theta_i, i = -m_1, \dots, -1, 0, 1, \dots, m_2$ des constantes réelles et $m_1, m_2 \in \mathbb{N}$.

Définition 5.6

On appelle ordre de la moyenne mobile, la valeur $m_1 + m_2 + 1$.

Remarque 5.4

La moyenne mobile Y_t est définie $\forall t$:

$$m_1 + 1 \leq t \leq T - m_2 \tag{5.23}$$

car à $i = -m_1$, on a X_{t-m1}. Or la première valeur de la série est X_1, donc $t - m_1 \geq 1$, ce qui implique que $m_1 + 1 \leq t$. De même, si $i = m_2$, on a X_{t+m_2}, or $\max\{t\} = T$, donc $t + m_2 \leq \max\{t\} = T$, ce qui entraîne que $t \leq T - m_2$. D'où : $m_1 + 1 \leq t \leq T - m_2$

5.1.2.1 Opérateur retard et opérateur avance

Définition 5.7 *Opérateur retard*

On appelle opérateur retard, l'application notée B (ou L), qui à tout processus $(X_t)_{t\in\mathbb{Z}}$ associe le processus défini comme suit :

$$B : X_t \longrightarrow B(X_t) = X_{t-1} \tag{5.24}$$

Si on compose B avec lui - même, on obtient $B^2 = B \circ B$ tel que :

$$B \circ BX_t = B\left[B(X_t)\right] = B(X_{t-1}) = X_{t-2} \tag{5.25}$$

On montre ainsi par récurrence que pour tour $n \in \mathbb{N}^*$, on a :

$$B^n = X_{t-n} \tag{5.26}$$

En utilisant l'opérateur retard, l'équation (5.22) se réécrit comme suit :

$$M_{m_1+m_2+1}(X_t) = \sum_{i=-m_1}^{i=m_2} \theta_i X_{t+i} = \sum_{i=-m_1}^{i=m_2} \theta_i B^{-i}(X_t) \tag{5.27}$$

Propriété 5.1

L'opérateur B est un opérateur linéaire et inversible.

Définition 5.8 *Opérateur avance*

On appelle opérater avance, notée F, l'application qui à tout processus $(X_t)_{t\in\mathbb{Z}}$ associe le processus défini comme suit :

$$F : X_t \longrightarrow F(X_t) = X_{t+1} \tag{5.28}$$

Par définition, on a :

$$F \circ FX_t = F\left[F(X_t)\right] = F(X_{t+1}) = X_{t+2} \tag{5.29}$$

On montre par récurrence que pour tout $n \in \mathbb{N}$

$$F^n = X_{t+n} \tag{5.30}$$

En utilisant l'opérateur retard, l'équation (5.22) se réécrit comme suit :

$$M_{m_1+m_2+1}(X_t) = \sum_{i=-m_1}^{i=m_2} \theta_i X_{t+i} = \sum_{i=-m_1}^{i=m_2} \theta_i F^i(X_t) \tag{5.31}$$

Propriété 5.2

— $B^{-i} = F^i$: l'inverse de l'operateur retard est l'operateur avance
— $B^i = F^{-i}$: l'inverse de l'operateur avance est l'operateur retard.

On peut réécrire la moyenne mobile en terme d'opérateurs B et F :

$$M_{m_1+m_2+1}(X_t) = \sum_{i=-m_1}^{i=m_2} \theta_i X_{t+i} = \sum_{i=-m_1}^{i=m_2} \theta_i B^{-i}(X_t) = \sum_{i=-m_1}^{i=m_2} \theta_i F^i(X_t) \tag{5.32}$$

Proposition 5.1 *Forme canonique d'une moyenne mobile*

En factorisant par B^{m_1} et en effectuant le changement de variable $j = i + m_1$, on obtient la forme canonique suivante :

$$M_{m_1+m_2+1} = B^{m_1} \sum_{j=0}^{j=m_1+m_2} \theta_{j-m_1} F^j = B^{m_1} \sum_{i=-m_1}^{i=m_2} \theta_i F^{m_1+i} =: B^{m_1}\Theta(F) \tag{5.33}$$

Remarque 5.5

— On notera que $-m_1$ est donc le plus petit exposant de F et m_2 le plus grand dans la définition de la moyenne mobile.
— De manière équivalente, m_1 est donc le plus grand exposant de B et $-m_2$ le plus petit.

Proposition 5.2

Le polynôme Θ intervenant dans l'expression (5.33) et défini par

$$\Theta(z) = \sum_{i=-m_1}^{i=m_2} \theta_i z^{m_1+i} = \theta_{-m_1} + \theta_{-m_1+1} z + \cdots + \theta_{m_2} z^{m_1+m_2} \tag{5.34}$$

est appelé polynôme caractéristique de la moyenne mobile $M_{m_1+m_2+1}$. On remarque que Θ peut aussi s'écrire sous la forme suivante :

$$\Theta(z) = \sum_{j=0}^{j=m_1+m_2} \theta_{j-m_1} z^j \tag{5.35}$$

5.1.2.2 Moyennes mobiles centrées

Définition 5.9 *Moyenne mobile centrée*

Une moyenne mobile est dite *centrée* si et seulement $m_1 = m_2 = m$. La moyenne mobile s'écrit alors :

$$M_{2m+1}(X_t) = \sum_{i=-m}^{i=m} \theta_i X_{t+i} = \sum_{i=-m}^{i=m} \theta_i B^{-i}(X_t) \tag{5.36}$$

On pose : $M_{2m+1} = \sum_{i=-m}^{i=m} \theta_i B^{-i} = B^m \left[\sum_{i=-m}^{i=m} \theta_i B^{-(m+i)} \right]$. Sous forme extensive M s'écrit comme suit :

$$M_{2m+1} = B^m \left[\theta_{-m} + \theta_{-m+1} B^{-1} + \cdots + \theta_m B^{-2m}\right] = B^m \Theta(B^{-1}) \tag{5.37}$$

avec $\Theta(z) = \theta_{-m} + \theta_{-m+1} z + \cdots + \theta_m z^{2m}$. Θ est appelé polynôme caractéristique associé à la moyenne mobile M.

Exemple 5.6

Soient M^1 et M^2 deux moyennes mobiles définies de la manière suivante :

$$\begin{cases} M^1(X_t) &= (1-B)X_t = X_t - X_{t-1} \\ M^2(X_t) &= (F-1)X_t = X_{t+1} - X_t \end{cases} \tag{5.38}$$

— Calculer $M^1 \circ M^2(X_t)$.

$$\begin{aligned} M^1 \circ M^2(X_t) &= M^1\left[M^2(X_t\right] = M^1\left[X_{t+1} - X_t\right] \\ &= M^1(X_{t+1}) - M^1(X_t) = (X_{t+1} - X_t) - (X_t - X_{t-1}) \\ &= X_{t+1} - 2X_t + X_{t-1} \end{aligned}$$

On remarque que $M^1 \circ M^2$ est centrée sans que M^1 et M^2 ne le soient.

— Polynôme caractéristique de $M^1 \circ M^2$:

$$\begin{aligned} M^1 \circ M^2(X_t) &= X_{t+1} - 2X_t + X_{t+1} = B^{-1}(X_t) - 2B^0(X_t) + B^1(X_t) \\ &= B\left[B^{-2}(X_t) - 2B^{-1}(X_t) + Id\right] = B\left[\Theta(B^{-1})\right](X_t) \end{aligned}$$

avec $\Theta(z) = z^2 - 2z + 1$.

Exemple 5.7

Donner le polynôme caractéristique des moyennes mobiles suivantes : $M^1(X_t) = \dfrac{1}{3}\left(X_{t-1} + X_t + X_{t+1}\right)$, $M^2(X_t) = (1-B)X_t$ et $M^3(X_t) = (F-1)X_t$

5.1.2.3 Moyennes mobiles symétriques

Définition 5.10 *Moyenne mobile symétrique*

Une moyenne mobile centrée est dite symétrique si elle vérifie la propriété :

$$\theta_i = \theta_{-i}, \quad \forall i = -m, \ldots, m \tag{5.39}$$

Propriété 5.3 *Propriétés des moyennes mobiles*

1. La composée ou le produit de deux moyennes mobiles est une moyenne mobile.
2. Si M^1 est d'ordre m_1 et M^2 d'ordre m_2, alors $M^1 \circ M^2$ est d'ordre $m_1 + m_2 - 1$.
3. $M^1 \circ M^2 = M^2 \circ M^1$
4. Si M^1 et M^2 sont centrées, alors $M^1 \circ M^2$ est aussi centrée.
5. Une moyenne mobile M est symétrique si et seulement si le polynôme associé est symétrique.
6. Si M^1 et M^2 sont symétriqques, alors $M^1 \circ M^2$ est aussi symétrique.

preuve

Attention

$M^1 \circ M^2$ peut êre centrée et symétrique sans que M^1 et M^2 ne le soient.

5.1.2.4 Moyennes mobiles arithmétiques

Définition 5.11 *Moyenne mobile arithmétique*

Une moyenne mobile est dite arithmétique si elle est symétrique ($\theta_i = \theta_{-i}$) et vérifie $\sum_i \theta_i = 1$. C'est la moyenne mobile qui préserve les constantes. Pour une moyenne mobile arithmétique $m_1 = m_2 = m$ et $\theta_i = \dfrac{1}{2m+1} \quad \forall i$.

Soit le programme :

$$\min \sum_{i=-m}^{i=m} \theta_i \quad \text{sous la contrainte} \quad \sum_{i=-m}^{i=m} \theta_i = 1 \tag{5.40}$$

La résolution de ce programme par la méthode de Lagrange donne le résultat : $\theta_i = \dfrac{1}{2m+1}$, $\forall i$.

Donc la moyenne mobile arithmétique s'écrit :

$$M_{2m+1}(X_t) = \frac{1}{2m+1} \sum_{i=-m}^{i=m} X_{t+i} = B^m \left[\Theta(F)\right] X_t \tag{5.41}$$

avec

$$\begin{aligned} \Theta(z) &= \theta_{-m} + \theta_{-m+1} z + \theta_{-m+2} z^2 + \cdots + \theta_m z^{2m} \\ &= \frac{1}{2m+1} \left(1 + z + z^2 + \cdots + z^{2m}\right) \end{aligned}$$

Propriété 5.4 *Propriété d'une moyenne mobile arithmétique*

- $\sum_{i=-m}^{i=m} \theta_i = 1$, donc la moyenne mobile arithmétique conserve les constantes
- La moyenne mobile arithmétique étant symétrique et $\sum_{i=-m}^{i=m} \theta_i = 1$, elle conserve les polynômes de degré 1.

Proposition 5.3 *Moyenne mobile arithmétique et filtre de convolution*

La moyenne mobile arithmétique est un cas particulier du filtre de convolution. En effet, on a vu que le calcul de la moyenne mobile arithmétrique dépend d'un paramètre m appelé largeur de la fenêtre. Ce paramètre correspond au nombre d'observations inclues dans le calcul de la moyenne glissante effectuée. La moyenne mobile est donc :

$$Y_t = \sum_{i=-m}^{i=m} \psi_i X_{t-i} \tag{5.42}$$

La moyenne mobile arithmétique étant centrée et symétrique, on a : $\psi_{-i} = \psi_i = \dfrac{1}{2m+1}$.

Proposition 5.4 *Moyennes mobiles paires et impaires*

On applique à X_t une moyenne mobile d'ordre égale à la période de la série. La série Y_t obtenue correspond à une estimation de la tendance.

— Si p est paire ($p = 2m$), alors l'expression de la moyenne mobile arithmétique est :

$$Y_t = \frac{1}{p}\left(\frac{1}{2}X_{t-m} + X_{t-m+1} + \cdots + X_t + \cdots + X_{t+m-1} + \frac{1}{2}X_{t+m}\right) \tag{5.43}$$

— Si p est impaire ($p = 2m+1$), alors l'expression de la moyenne mobile arithmétique est :

$$Y_t = \frac{1}{p}\left(X_{t-m} + X_{t-m+1} + \cdots + X_t + \cdots + X_{t+m-1} + X_{t+m}\right) \tag{5.44}$$

Exemple 5.8 *Application sur la série elecsales*

On considère la série elecsales contenue dans le package fpp2 de R. Cette série représente les ventes annuelles d'électricité pour l'Australie - Méridionale en GWh de 1989 à 2008. L'électricité utilisée pour l'eau chaude a été exclue. On demande d'appliquer une moyenne mobile centrée d'ordre 5.

```
# Chargement des données
elecsales = pd.DataFrame(r.elecsales,columns =["elecsales"])
elecsales.index = pd.date_range("1989-01-31",periods=len(elecsales),freq = "Y")
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(elecsales,color = "black");
axe.set_xlabel("année");
axe.set_ylabel("valeur");
plt.show()
```

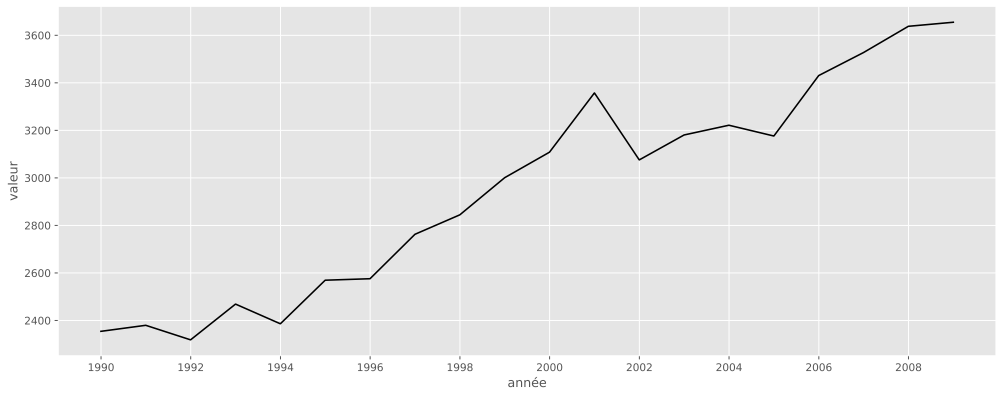


Figure 5.8 – Ventes résidentielles d'électricité (hors eau chaude) pour l'Australie-Méridionale : 1989-2008.

On a $p = 5$, ce qui implique que $m = 2$ et on applique la formule suivante :

$$M_5(X_t) = \frac{X_{t-2} + X_{t-1} + X_t + X_{t+1} + X_{t+2}}{5} \tag{5.45}$$

Nous utilisons notre fonction `ConvolveFilter2` en fixant la période à 5.

```
# Moyenne mobile d'ordre 5 centrée et symétrique
elecsales["5-MA"] = ConvolveFilter2(elecsales.elecsales,period=5)
```

Table 5.1 – Ventes annuelles d'électricité aux clients résidentiels en Australie-Méridionale. 1989–2008

	elecsales	5-MA
1989-12-31	2354.34	NaN
1990-12-31	2379.71	NaN
1991-12-31	2318.52	2381.530
1992-12-31	2468.99	2424.556
1993-12-31	2386.09	2463.758
1994-12-31	2569.47	2552.598
1995-12-31	2575.72	2627.700
1996-12-31	2762.72	2750.622
1997-12-31	2844.50	2858.348
1998-12-31	3000.70	3014.704
1999-12-31	3108.10	3077.300
2000-12-31	3357.50	3144.520
2001-12-31	3075.70	3188.700
2002-12-31	3180.60	3202.320
2003-12-31	3221.60	3216.940
2004-12-31	3176.20	3307.296
2005-12-31	3430.60	3398.754
2006-12-31	3527.48	3485.434
2007-12-31	3637.89	NaN
2008-12-31	3655.00	NaN

Dans la dernière colonne du tableau (5.1), une moyenne mobile d'ordre 5 est indiquée, fournissant une estimation de la tendance $\widehat{Z}_t$. La première valeur de cette colonne est la moyenne des cinq premières observations (1989-1993) ; la deuxième valeur dans la colonne 5-MA est la moyenne des valeurs pour 1990-1994 ; etc. Chaque valeur dans la colonne 5-MA est la moyenne des observations dans la fenêtre de cinq ans centrée sur l'année correspondante.

Il n'y a pas de valeurs ni pour les deux premières années ni pour les deux dernières années, car nous n'avons pas deux observations de part et d'autre.

Pour voir à quoi ressemble l'estimation de la tendance, nous la traçons avec les données d'origine dans la figure (5.9).

```
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(elecsales.elecsales,color = "black", label = "elecsales");
axe.plot(elecsales["5-MA"],color = "red",linestyle = "--",label="5-MA");
axe.set_xlabel("année");
axe.set_ylabel("value");
axe.legend();
plt.show()
```

Notons que la tendance (en rouge) est plus lisse que les données d'origine et capture le mouvement principal de la série chronologique sans toutes les fluctuations mineures. L'ordre de la moyenne mobile détermine la régularité de l'estimation de la tendance. En général, un ordre plus grand signifie une courbe plus lisse. La figure (5.10) montre l'effet de la modification de l'ordre de la moyenne mobile des données sur les ventes résidentielles d'électricité.

```
# Moyennes mobiles d'ordre 3, 7 et 9
elecsales["3-MA"] = ConvolveFilter2(elecsales.elecsales,period = 3)
```

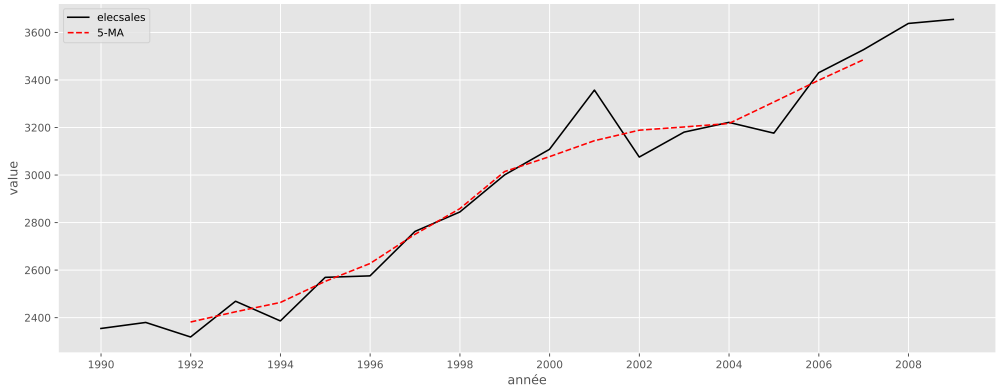


Figure 5.9 – Ventes d'électricité résidentielles (noir) avec l'estimation par moyenne mobile d'ordre 5 centrée et symétrique de la tendance (rouge)

```
elecsales["7-MA"] = ConvolveFilter2(elecsales.elecsales,period = 7)
elecsales["9-MA"] = ConvolveFilter2(elecsales.elecsales,period = 9)

# Représentation graphique
fig = plt.figure(figsize=(16,8))
for idx, name in enumerate(["3-MA","5-MA","7-MA","9-MA"]):
  ax = fig.add_subplot(2,2,idx+1)
  ax.plot(elecsales.elecsales,color = "black");
  ax.plot(elecsales[name],color = "red",linestyle = "--");
  ax.set_title(f"{name}");
  ax.set_xlabel("année");
  ax.set_ylabel("ventes (GWh)");
plt.tight_layout()
plt.show()
```

Figure 5.10 – Différentes moyennes mobiles appliquées aux données sur les ventes d'électricité résidentielles

5.1.2.5 Moyennes mobiles des moyennes mobiles

Il est possible d'appliquer une moyenne mobile à une moyenne mobile. L'une des raisons de le faire est de rendre symétrique une moyenne mobile d'ordre pair. Voyons cela avec l'exemple suivant.

Exemple 5.9 *Moyennes mobiles des moyennes mobiles sur la série beer2*

On considère la série ausbeer contenue dans le package fpp2 de R. Cette série représente la production trimestrielle totale de bière en Australie (en mégalitres) du premier trimestre 1956 au deuxième trimestre de 2010. Nous travaillons avec les données à partir de 1992.

```
# Chargement des données
beer = sm.datasets.get_rdataset("ausbeer", "fpp2").data.drop("time",axis=1)
beer = beer.rename(columns = {"value" : "ausbeer"})
beer.index = pd.date_range("1956-01-31",periods=len(beer),freq = "3M")
ausbeer = beer[beer.index >="1992-01-31"]
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(ausbeer.ausbeer,color = "black");
axe.set_xlabel("Date");
axe.set_ylabel("Production de bière (en mégalitres)");
plt.show()
```

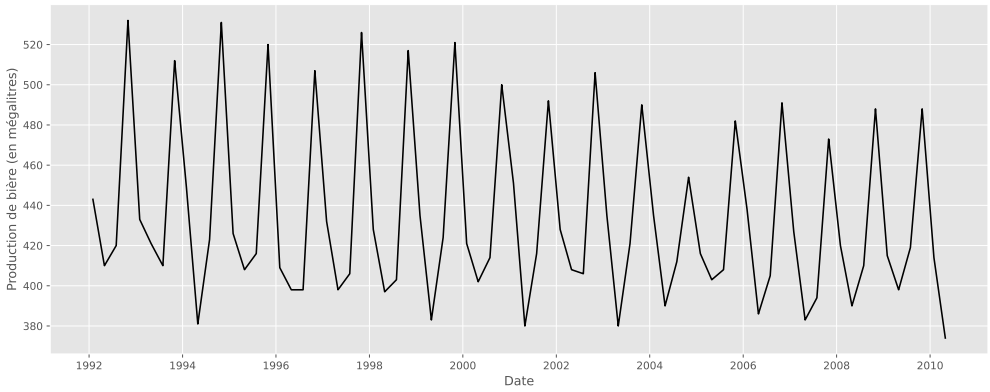


Figure 5.11 – Production trimestrielle totale de bière en Australie (en mégalitres)

Nous appliquons une moyenne mobile d'ordre 4 non centrée :

$$Y_t = \frac{X_{t-1} + X_t + X_{t+1} + X_{t+2}}{4} \tag{5.46}$$

```
theta_beer_4 = list([1/4,1/4,1/4,1/4])
x = np.convolve(a = theta_beer_4, v = ausbeer.ausbeer, mode='valid')
ausbeer["4-MA"] = np.hstack((np.nan,np.hstack((x,np.zeros(2)+np.nan))))
```

Nous appliquons aussi une moyenne mobile d'ordre 4 centrée :

$$Y_t = \frac{1}{8}X_{t-2} + \frac{1}{4}\sum_{i=-1}^{i=1} X_{t+i} + \frac{1}{8}X_{t+2} \tag{5.47}$$

```
# Moyenne mobile d'ordre 4
ausbeer["2x4-MA"] = ConvolveFilter2(ausbeer.ausbeer,period = 4)
```

Table 5.2 – Une moyenne mobile de rang 4 appliquée aux données trimestrielles sur la bière, suivie d'une moyenne mobile de rang 2. (10 premières observations)

	ausbeer	4-MA	2x4-MA
1992-01-31	443	NaN	NaN
1992-04-30	410	451.25	NaN
1992-07-31	420	448.75	450.000
1992-10-31	532	451.50	450.125
1993-01-31	433	449.00	450.250
1993-04-30	421	444.00	446.500
1993-07-31	410	448.00	446.000
1993-10-31	512	438.00	443.000
1994-01-31	449	441.25	439.625
1994-04-30	381	446.00	443.625

La notation « $2 \times 4 - \text{MA}$ » dans la dernière colonne signifie une moyenne mobile d'ordre 4 suivie d'une moyenne mobile d'ordre 2. Les valeurs de la dernière colonne sont obtenues en prenant une moyenne mobile d'ordre 2 des valeurs de la colonne précédente. Par exemple, les deux premières valeurs de la colonne 4-MA sont $451.25 = (443 + 410 + 420 + 532)/4$ et $448.75 = (410 + 420 + 532 + 433)/4$. La première valeur de la colonne $2 \times 4 - \text{MA}$ est la moyenne de ces deux : $450.00 = (451.25 + 448.75)/2$.

Lorsqu'une moyenne mobile d'ordre 2 suit une moyenne mobile d'ordre pair (comme 4), on l'appelle une « moyenne mobile centrée d'ordre 4 ». C'est parce que les résultats sont maintenant symétriques. Pour voie que c'est le cas, on peut écrire la moyenne mobile $2 \times 4 - \text{MA}$ comme suit :

$$\begin{aligned} Y_t &= \frac{1}{4}\left[\frac{1}{2}\left(X_{t-2} + X_{t-1} + X_t + X_{t+1}\right) + \frac{1}{2}\left(X_{t-1} + X_t + X_{t+1} + X_{t+2}\right)\right] \\ &= \frac{1}{8}X_{t-2} + \frac{1}{4}X_{t-1} + \frac{1}{4}X_t + \frac{1}{4}X_{t+1} + \frac{1}{8}X_{t+2} = \frac{1}{4}\left(\frac{1}{2}X_{t-2} + X_{t-1} + X_t + X_{t+1} + \frac{1}{2}X_{t+2}\right) \end{aligned}$$

C'est maintenant une moyenne pondérée des observations qui est symétrique.

5.1.3 Autres types de moyennes mobiles

5.1.3.1 Moyennes mobiles cumulatives

Les moyennes mobiles cumulatives incluent toutes les données jusqu'à un certain point spécifié par l'économètre. Par exemple, il peut vouloir visualiser la moyenne mobile à partir d'une certaine date, comme le premier du mois ou de l'année civile, pour obtenir la moyenne mobile cumulative basée sur toutes les données à partir de ce point, jusqu'à la période actuelle. Avec les moyennes mobiles cumulatives, les données les plus anciennes ne disparaissent pas et sont au contraire ajoutées à la moyenne cumulative.

$$CMA_t(X_t) = \frac{X_1 + X_2 + \cdots + X_t}{t} \tag{5.48}$$

Sous Python, nous utilisons la fonction pd.expanding() pour effectuer une moyenne mobile cumulative

Exemple 5.10

```
# Moyennes mobiles cumulatives
elecsales["5-CMA"] = elecsales.elecsales.expanding(5,center=False).mean()
```

Table 5.3 – Ventes annuelles d'électricité aux clients résidentiels en Australie-Méridionale. 1989–2008 (10 premières observations)

	elecsales	5-MA	3-MA	7-MA	9-MA	5-CMA
1989-12-31	2354.34	NaN	NaN	NaN	NaN	NaN
1990-12-31	2379.71	NaN	2350.857	NaN	NaN	NaN
1991-12-31	2318.52	2381.530	2389.073	NaN	NaN	NaN
1992-12-31	2468.99	2424.556	2391.200	2436.120	NaN	NaN
1993-12-31	2386.09	2463.758	2474.850	2494.460	2517.784	2381.530
1994-12-31	2569.47	2552.598	2510.427	2560.859	2589.602	2412.853
1995-12-31	2575.72	2627.700	2635.970	2658.313	2670.534	2436.120
1996-12-31	2762.72	2750.622	2727.647	2749.614	2785.977	2476.945
1997-12-31	2844.50	2858.348	2869.307	2888.387	2853.389	2517.784
1998-12-31	3000.70	3014.704	2984.433	2960.706	2941.668	2566.076

```
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(elecsales.elecsales,color = "black", label = "elecsales");
axe.plot(elecsales["5-CMA"],color = "red",linestyle = "--",label="5-CMA");
axe.set_xlabel("année");
axe.set_ylabel("value");
axe.legend();
plt.show()
```

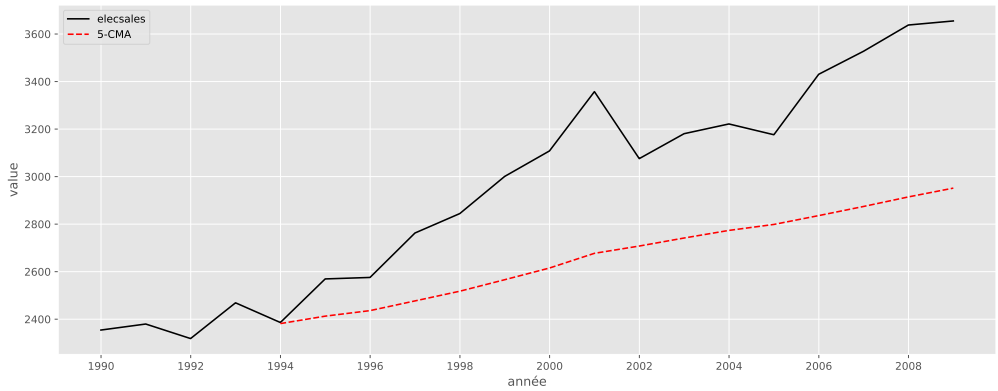


Figure 5.12 – Ventes d'électricité résidentielles (noir) avec l'estimation par moyenne mobile cumulative d'ordre 5 de la tendance (rouge)

5.1.3.2 Moyennes mobiles pondérées

Les moyennes mobiles pondérées agissent de la même manière que la moyenne mobile simple, mais elles impliquent des facteurs supplémentaires qui multiplient les données pour leur appliquer

des pondérations différentes. La moyenne pondérée est une convolution des différents points de données avec une pondération fixe appliquée. La moyenne mobile pondérée génère des signaux de transaction en attribuant une pondération plus importante aux points de données récents et une pondération moindre aux points de données passés.

Définition 5.12 *Moyenne mobile pondérée*

Soit $(X_t)_t$ une série chronologique, la moyenne mobile pondérée d'ordre m associée à X_t s'écrit :

$$WMA_{m,t}(X_t) = \theta_1 X_{t-1} + \theta_2 X_{t-2} + \cdots + \theta_m X_{t-m} = \sum_{i=1}^{m} \theta_i X_{t-i} \tag{5.49}$$

où θ_i est appelé facteur ou coefficient de pondération.

Propriété 5.5 *Les coefficients de pondération vérifient les propriétés suivantes :*

$$\theta_1 > \theta_2 > \cdots > \theta_m \quad et \quad \sum_{i=1}^{i=m} \theta_i = 1 \tag{5.50}$$

Exemple 5.11 *Ventes hebdomadaires d'un produit cosmétique*

On considère les données du tableau (5.4) qui portent sur les ventes hebdomadaires d'un produit cosmétique (en millions de FCFA). On souhaite calculer la moyenne mobile pondérée d'ordre 4. Ainsi, on a : $\theta_1 = 0.4$, $\theta_2 = 0.3$, $\theta_3 = 0.2$ et $\theta_4 = 0.1$.

Table 5.4 – Moyenne mobile pondérée sur les ventes hebdomadaires d'un produit cosmétique

Semaine	Ventes	$4-WMA$
1	39	
2	44	
3	40	
4	45	
5	38	42.7
6	43	41.1
7	39	41.6
h		$\widehat{X_t}$
8		40.6

Exemple 5.12 *Moyenne mobile pondérée sur la série elecequip*

Nous allons créer une fonction `WeightedMovingAverage` qui calcule la moyenne mobile pondérée.

```
# Moyenne mobile pondérée
def WeightedMovingAverage(x,period=4):
  x = np.array(x)
  wma = np.zeros(len(x))
  wma[:period] = np.nan
  weighted = np.arange(1,period+1)[::-1]/np.arange(1,period+1).sum()
  wma[period:] = np.convolve(weighted,x,mode='valid')[:-1]
  return wma
```

```
# Application
elecequip["4-WMA"] = WeightedMovingAverage(elecequip.elecequip,period=4)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(elecequip.elecequip,color = "black",label = "$X_{}t$");
axe.plot(elecequip["4-WMA"],color="red",linestyle = "--",
         label= "$\widehat{X}_{t}$");
axe.set_xlabel("Date");
axe.set_ylabel("Indice des nouvelles commandes");
axe.legend();
plt.show()
```

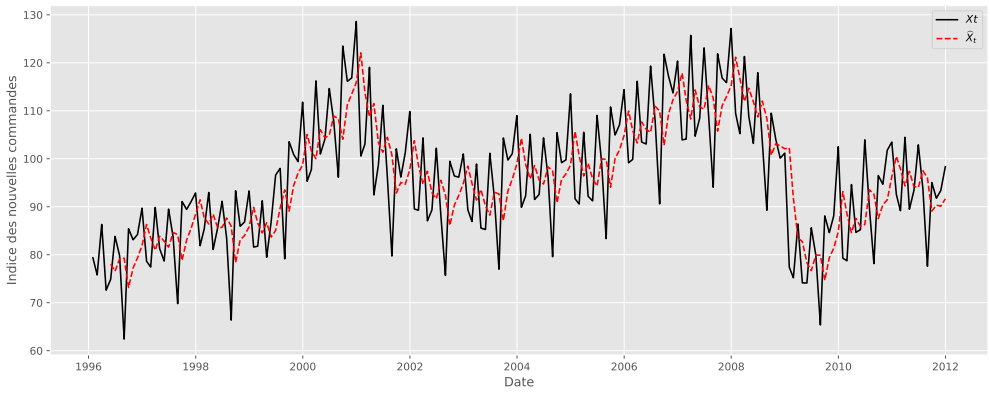


Figure 5.13 – Moyenne mobile pondérée d'ordre 4 sur la série elecequip

5.1.3.3 Moyennes mobiles exponentielles

Une moyenne mobile exponentielle (EMA, *Exponantial Moving Average*) est une moyenne mobile pondérée dans le temps. Par cette pondération, elle accorde plus de poids aux prix de clôture les plus récents, et moins aux prix de clôture les plus anciens.

Sous Python, nous utilisons la fonction pd.ewm() pour effectuer une moyenne mobile exponentielle. Dans cette fonction, deux approches sont définies dépendant du paramètre `adjust` fixé par défaut à `True`. Ainsi,

— Si `adjust = True`

La formule de calcul est la suivante :

$$EMA_t = \frac{\sum_{i=0}^{i=t} \omega_i X_{t-i}}{\sum_{j=0}^{j=t} \omega_j} \tag{5.51}$$

$\omega_i = (1-\alpha)^i$ est appelé poids de pondération.

— Si `adjust = False`

La formule de calcul est la suivante :

$$EMA_t = \begin{cases} X_0 & t = 0 \\ \alpha X_t + (1-\alpha) Y_{t-1} & t > 0 \end{cases} \tag{5.52}$$

Exemple 5.13 *Application de la moyenne mobile exponentielle à la série elecsales*

```
# Moyenne mobile exponentielle
elecsales["5-EWMA"] = elecsales.elecsales.ewm(span=5,adjust=False).mean()
elecsales["5-EWMA-Adj."] = elecsales.elecsales.ewm(span=5,adjust=True).mean()

# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(elecsales.elecsales,color = "black", label = "elecsales");
axe.plot(elecsales["5-EWMA"],color = "red",linestyle = "--",label="5-EWMA");
axe.plot(elecsales["5-EWMA-Adj."],color = "blue",linestyle = "--",
         label="5-EWMA Adjusted");
axe.set_xlabel("année");
axe.set_ylabel("value");
axe.legend();
plt.show()
```

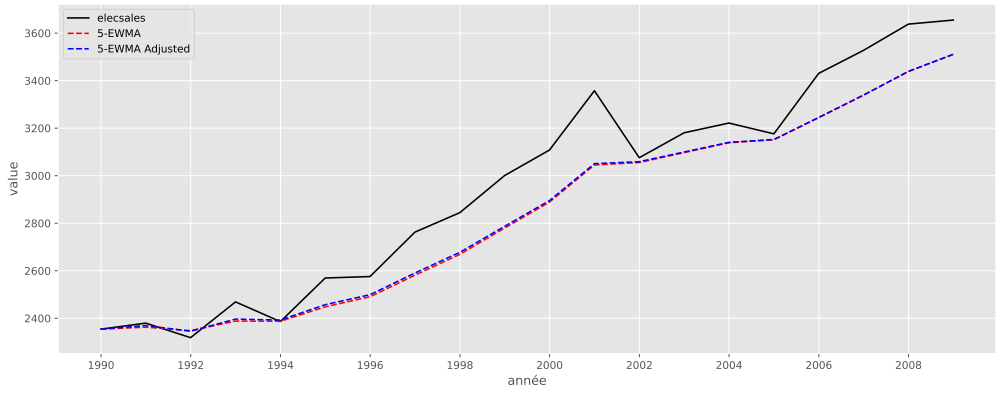


Figure 5.14 – Ventes d'électricité résidentielles (noir) avec l'estimation par moyenne mobile exponentielle d'ordre 5 de la tendance (rouge et bleue)

5.1.4 Propriété d'un lissage par moyenne mobile

5.1.4.1 Effet d'une moyenne mobile sur une tendance

L'application d'une moyenne mobile arithmétique (paire ou impaire) ne modifie pas une tendance constante. L'application d'une moyenne mobile arithmétique (paire ou impaire) conserve une tendance linéaire.

Plus précisément

Propriété 5.6 *1. Une moyenne mobile conserve les constantes si et seulement si*

$$\theta_{-m_1} + \cdots + \theta_{-m_2} = 1 \tag{5.53}$$

2. *Une moyenne mobile symétrique conservant les constantes conserve les polynômes de degré 1.*

Exercice 5.1 *Démontrer les deux propriétés précédentes*

On vient de voir qu'une moyenne mobile arithmétique d'ordre $2m+1$ conserve les polynômes de degré 1. Quelles sont les autres séries inavriantes par cette moyenne.

Exercice 5.2 *Par exemple, vérifier qu'une moyenne mobile arithmétique ne conserve pas les polynômes de degré 2.*

Exercice 5.3 *On peut trouver des séries invariantes par $M_1 \circ M_2$ à partir de celles invariantes par M_1 et M_2 :*

1. *Si X_t est une série invariante par M_1 et M_2, alors X_t est invariante par $M_1 \circ M_2$.*
2. *Qu'en est - il de la réciproque ?*

Remarque 5.6

Comme une moyenne mobile est un opérateur linéaire sur l'espace vectoriel des suites indexées par $\mathbb{Z}$, chercher les séries invariantes par une moyenne mobile, c'est - à - dire les séries qui vérifient

$$M_{m_1+m_2+1}(X_t) = X_t \tag{5.54}$$

revient donc à chercher les vecteurs propres associés à la valeur propre 1 de l'opérateur moyenne mobile considéré.

Pour trouver les séries invariantes par une moyenne mobile, on considère le polynôme

$$\Theta(z) - z^{m_1} = \theta_{-m_1} + \theta_{-m_1+1} z + \cdots + \theta_{m_2} z^{m_1+m_2} - z^{m_1} \tag{5.55}$$

dont on cherche les racines. Dans le cas centré, on considère simplement

$$\Theta(z) - z^{m} = \theta_{-m} + \theta_{-m+1} z + \cdots + \theta_{m} z^{2m} - z^{m} \tag{5.56}$$

Propriété 5.7 *La moyenne mobile d'ordre m_1+m_2+1 conserve les polynômes de degré inférieur ou égal à p si et seulement si 1 est racine d'ordre $p+1$ du polynôme $\Theta(z) - z^{m_1}$.*

Preuve

Etudions le polynôme $\Phi(z) := \Theta(z) - z^{m_1}$. Tout d'abord, nous avons pour tout n

$$\begin{cases} \Theta(z) & = \sum_{i=-m_1}^{i=m_2} \theta_i z^{m_1+i} \\ \Theta^{'}(z) & = \sum_{i=-m_1}^{i=m_2} \theta_i (m_1+i) z^{m_1+i-1} \\ \Theta^{''}(z) & = \sum_{i=-m_1}^{i=m_2} \theta_i (m_1+i)(m_1+i-1) z^{m_1+i-2} \\ & \vdots \\ \Theta^{(n)}(z) & = \sum_{i=-m_1}^{i=m_2} \theta_i (m_1+i)(m_1+i-1)\cdots(m_1+i-n+1) z^{m_1+i-n} \end{cases}$$

et immédiatement

$$\begin{cases} \Phi(1) & = \sum_{i=-m_1}^{i=m_2} \theta_i - 1 \\ \Phi^{'}(1) & = \sum_{i=-m_1}^{i=m_2} \theta_i (m_1+i) - m_1 = \sum_{i=-m_1}^{i=m_2} i\theta_i + m_1 \left(\sum_{i=-m_1}^{i=m_2} \theta_i - 1 \right) \\ \Phi^{''}(1) & = \sum_{i=-m_1}^{i=m_2} \theta_i (m_1+i)(m_1+i-1) - m_1(m_1-1) \\ & = \sum_{i=-m_1}^{i=m_2} i^2\theta_i + (2m_1-1) \sum_{i=-m_1}^{i=m_2} i\theta_i + m_1(m_1-1) \left(\sum_{i=-m_1}^{i=m_2} \theta_i - 1 \right) \\ & \vdots \\ \Phi^{(n)}(1) & = \sum_{i=-m_1}^{i=m_2} \theta_i (m_1+i)\cdots(m_1+i-n+1) - m_1(m_1-1)\cdots(m_1-n+1) \\ & = \sum_{l=1}^{l=n} \sum_{i=-m_1}^{i=m_2} i^l\theta_i + \cdots + m_1\cdots(m_1-n+1) \left(\sum_{i=-m_1}^{i=m_2} \theta_i - 1 \right) \end{cases}$$

Soit maintenant la série $X_t = t^n$, pour $n \leq p$. Sa transformée par la moyenne mobile centrée d'ordre $2m+1$ est donnée par :

$$\begin{aligned} M_{2m+1}(X_t) &= \sum_{i=-m_1}^{i=m_2} \theta_i (t+i)^n = \sum_{i=-m_1}^{i=m_2} \theta_i \sum_{k=0}^{k=n} \mathrm{C}_n^k t^{n-k} i^k \quad \text{(en appliquant la formule du binôme)} \\ &= \sum_{k=0}^{k=n} \mathrm{C}_n^k t^{n-k} \sum_{i=-m_1}^{i=m_2} i^k \theta_i = \mathrm{C}_n^0 t^n = t^n \end{aligned}$$

si on suppose que 1 est racine d'ordre $p+1$ de $\Theta(z) - z^{m_1}$. Et on en déduit que la moyenne mobile d'ordre m_1+m_2+1 conserve bien les polynômes de degré inférieur ou égal à p. On vérifie aisément que la réciproque est vraie.

Propriété 5.8 *La moyenne mobile d'ordre m_1+m_2+1 conserve les fonctions de la forme a^t si et seulement si a est racine du polynôme $\Theta(z) - z^{m_1}$.*

Exercice 5.4 *Démontrer la propriété précédente*

Exemple 5.14 *Application à la moyenne mobile d'ordre* 5

— Ecrire son polynôme associé
— Vérifier que la moyenne arithmétique conserve les polynômes de degré 1.
— Déterminer les séries invariantes par la moyenne mobile arithmétique d'ordre 5.

5.1.4.2 Effet d'une moyenne mobile sur une composante saisonnière

Si la série X_t possède une composante saisonnière de période p alors l'application d'une moyenne mobile d'ordre p supprime cette saisonnalité. La série $(M_p(X_t))_i$ ne possède plus de composante saisonnière de période p.

Plus précisement, cherchons les chroniques qui sont arrêtées par le filtre moyenne mobile centrée d'ordre $2m+1$. Ce sont les séres S_t telles que leur transformée $\widehat{S_t}$ par la moyenne mobile vérifie $\widehat{S_t} = 0$. On cherche donc à déterminer le noyau d'une moyenne mobile arithmétique d'ordre $2m+1$ ou encore les vecteurs propres associés à la valeur propre nulle.

Pour trouver les séries arrêtées par une moyenne mobile, on considère le polynôme :

$$\Theta(z) = \theta_{-m_1} + \theta_{-m_1+1} z + \cdots + \theta_{-m_2} z^{m_1+m_2} \tag{5.57}$$

dont on cherche les racines. Dans le cas centré, on considère simplement :

$$\Theta(z) = \theta_{-m} + \theta_{-m+1} z + \cdots + \theta_m z^{2m} \tag{5.58}$$

Propriété 5.9 *La moyenne mobile d'ordre $m_1 + m_1 + 1$ arrête les fonctions de la forma a^t si et seulement si a est racine du polynôme $\Theta(z)$.*

Preuve

Exemple 5.15

On peut trouver des séries arrêtées par $M_1 \circ M_2$ à partir de celles arrêtées par M_1 et M_2 /

1. Si X_t est une série arrêtée par M_1 ou M_2 alors X_t est arrêtée par $M_1 \circ M_2$.
2. Qu'en est - il de la réciproque ?

La proposition ci - dessous montre uqe l'ensemble des séries arrêtées par M : $\mathcal{K}er(M)$ est un sous - espace vectoriel.

Proposition 5.5 *Soit une moyenne mobile M définie par*

$$M = \sum_{i=-m_1}^{i=m_2} \theta_i B^{-i} \tag{5.59}$$

alors

$$\mathcal{K}er(M) = Vect\left\{t^k r_j^t, \quad t \in \mathbb{Z}; \quad \forall k = 0, \ldots, l_j - 1, \quad \forall j = 1, \ldots, m\right\} \tag{5.60}$$

où $r_1, \dots, r_m$ sont les racines d'ordres de multiplicité respectives $l_1, \dots, l_m$ de l'équation suivante :

$$\forall t \in \mathbb{Z}, \quad \sum_{i=-m_1}^{i=m_2} \theta_i X_{t+i} = 0 \tag{5.61}$$

Preuve

En utilisant le polynôme caractéristique, il est immédiat de voir que toute suite (X_t) absorbée par la moyenne mobile M est solution de l'équation de récurrence ci - dessous :

$$\forall t \in \mathbb{Z}, \quad \sum_{i=-m_1}^{i=m_2} \theta_i X_{t+i} = 0 \tag{5.62}$$

Il s'agit d'une équation de récurrence linéaire à coefficients constants d'ordre m_1+m_2. L'ensemble des solutions de cette équation forme un espace vectoriel de dimension $m_1 + m_2$ dont une base est $(t^k r_j^t, \quad t \in \mathbb{Z})$ où $r_1, \dots, r_m$ sont les racines d'ordres de multiplicité respectives $l_1, \dots, l_m$ de l'équation ci - dessus.

Exemple 5.16

On recherche les séries arrêtées par le filtre moyenne mobile arithmétique.

Propriété 5.10 *Une moyenne mobile absorbe les composantes saisonnières de période p si et seulement si son polynôme caractéristique Θ est divisible par $1 + z + \cdots + z^{p-1}$.*

Preuve

5.1.4.3 Effet d'une moyenne mobile sur les fluctuations irrégulières

Jusqu'à présent, nous ne nous sommes intéressés qu'à l'effet d'une moyenne mobile sur la partie déterministe de la série (tendance et saisonnalité). Nous allons étudier maintenant l'effet d'une moyenne mobile sur le résidu lorsque celui - ci est un bruit blanc. Par construction, une moyenne mobile consiste à faire des moyennes partielles de proche en proche.On obtient donc un lissage de la série. L'effet de la composante irrégulière est d'autant plus atténué que l'ordre de la moyenne mobile est grand.

Plus précisément,

Propriété 5.11 *Les moyennes mobiles arithmétiques d'ordre $2m+1$ sont les moyennes mobiles minimisant la variance d'un bruit blanc parmi les moyennes mobiles centrées telles que $\sum_{i=-m}^{i=m} \theta_i = 1$.*

Exemple 5.17

Nous nous proposons de démontrer la propriété précédente.

— Calculer la transformée d'un bruit blanc par une moyenne mobile centrée d'ordre $2m+1$ telle que $\sum_{i=-m}^{i=m} \theta_i = 1$ et sa variance.

— Vérifier que le problème revient à résoudre le problème de minimisation

$$\min_{\theta_i, i=-m,\dots,m} \sum_{i=-m}^{i=m} \theta_i^2 \quad \text{sous la contrainte} \quad \sum_{i=-m}^{i=m} \theta_i = 1 \tag{5.63}$$

— En appliquant la méthode des multiplicatuers de Lagrange, déterminer le $(2m+1)$-uplet $(\theta_{-m}, \dots, \theta_m)$ solution de ce problème de minimisation.
— Conclure

Ainsi, les moyennes arithmétiques d'ordre $2m+1$ transforment un bruit blanc en un processus centré (inchangé) et de variance $\dfrac{\sigma_\varepsilon^2}{2m+1}$ réduite. Notons cependant que les variables $\widehat{\varepsilon}_t$ sont à présent corrélées alors que les ε_t ne l'étaient pas. On a en effet :

$$\begin{aligned}
\gamma(h) = \mathrm{Cov}\left(\widehat{\varepsilon}_t, \widehat{\varepsilon}_{t+h}\right) &= \mathbb{E}\left(\widehat{\varepsilon}_t \widehat{\varepsilon}_{t+h}\right) = \mathbb{E}\left(\sum_{i=-m}^{i=m} \theta_i \varepsilon_{t+i} \sum_{j=-m}^{j=m} \theta_j \varepsilon_{t+h+j}\right) \\
&= \sum_{i=-m}^{i=m} \sum_{j=-m}^{j=m} \theta_i \theta_j \mathbb{E}\left(\varepsilon_{t+i} \varepsilon_{t+h+j}\right) = \begin{cases} \sigma_\varepsilon^2 \displaystyle\sum_{i=h-m}^{i=m} \theta_i \theta_{i-h} & \text{pour} \quad h \leq 2m \\ 0 & \text{pour} \quad h > 2m \end{cases} \\
&= \begin{cases} \sigma_\varepsilon^2 \dfrac{2m+1-h}{(2m+1)^2} & \text{pour} \quad h \leq 2m \\ 0 & \text{pour} \quad h > 2m \end{cases}
\end{aligned}$$

On voit ainsi que la covariance entre $\widehat{\varepsilon}_t$ et $\widehat{\varepsilon}_{t+h}$ ne dépend que de σ_ε^2 (variance du bruit blanc et de h et non de t et qu'elle s'annule dès que $h > 2m$). En divisant $\gamma(h)$ par $\mathrm{Var}(\widehat{\varepsilon}_t) = \gamma(0)$, on définit la corrélation linéaire entre $\widehat{\varepsilon}_t$ et $\widehat{\varepsilon}_{t+h}$:

$$\rho(h) = \frac{\gamma(h)}{\gamma(0)} = \begin{cases} \dfrac{2m+1-h}{2m+1} & \text{pour} \quad h \leq 2m \\ 0 & \text{pour} \quad h > 2m \end{cases} \tag{5.64}$$

La corrélation est donc indépendante de la variance du bruit blanc.

5.1.4.4 Choix pratique de l'ordre d'une moyenne mobile

Rappelons que le but d'un lissage par moyenne mobile est de faire apparaître l'allure de la tendance. Il s'agit donc de faire disparaître la saisonnalité et de réduire au maximum le bruit blanc. Nous avons vu précédemment que

— on fait disparaître une composante saisonnière de période p avec une moyenne mobile d'ordre p.
— on gomme d'autant plus le bruit que l'ordre de la moyenne mobile est grand.
— en revanche, on perd les caractéristiques de la tendance avec une moyenne mobile d'ordre trop élevé (jusqu'à obtenir une tendance constante).

En pratique, on doit donc trouver le meilleur compromis pour le choix de l'ordre de lissage optimal.

Exemple 5.18 *Cas particuliers des moyennes mobiles arithmétiques*

On a vu qu'une moyenne mobile arithmétique d'ordre $2m+1$:

— laisse invariants les polynômes de degré 1
— arrête les fonctions périodiques de période $2m+1$
— réduit la variance d'un bruit blanc de manière optimale (c'est - à - dire d'une facteur $\frac{1}{2m+1}$)
— laissent invariantes certaines suites géométriques de la forme a^t (si a est racine de $\Theta(z) - z^m$),
— absorbent certains suites géométriques de la forme a^t (si a est racine de Θ).

C'est pourquoi ce sont celles que nous utiliserons le plus fréquemment en pratique.

Remarque 5.7

Dans la pratique, il est fréquent que la saisonnalité s'exprime par une fonction périodique de période paire $p = 2m$. C'est le cas par exemple, des données mensuelles ($p = 12$), trimestrielles ($p = 4$). Dans ce cas, pour éliminer la saisonnalité, nous utiliserons la moyenne mobile

$$\frac{1}{2m}\left(\frac{1}{2}X_{t-m} + X_{t-m+1} + \cdots + X_{t+m-1} + \frac{1}{2}X_{t+m}\right) \tag{5.65}$$

En reprenant les calculs analogues à ceux effectués précédemment dans cette section, on peut montrer que les moyennes mobiles ainsi définies :

— laissent invariants les polynômes de degré 1
— arrêtent les fonctions périodiques de période $2m$ dont la somme des coefficients est nulle
— réduisent la variance d'un bruit blanc par un facteur $\frac{2m - 1/2}{4m^2}$.

5.2 Décomposition d'une série chronologique par moyennes mobiles

Nous disposons maintenant des outils de base permettant la décomposition d'une série chronologique. Il est clair qu'afin de pouvoir estimer la tendance, c'est - à - dire le mouvement du phénomène observé sur un grand intervalle de temps, il faut disposer d'une série statistique sur une longue période. Disposant de ces données, comme nous l'avons dit précédemment, le premier travail consiste à effectuer une représentation graphique adéquate permettant d'avoir une vue globale du phénomène en question. Afin d'éliminer ou d'amortir les mouvements cycliques, saisonniers et accidentels, on utilise donc la technique des moyennes mobiles et on procède ainsi en quelque sorte au lissage de la courbe.

D'après ce que nous venons de voir, la méthode des moyennes mobiles arithmétiques peut être utilisée pour tout type de modèle. Cependant, d'après ses propriétés, elle est particulièrement adaptée pour le modèle déterministe additif lorsque :

— la tendance est sensiblement linéaire
— la composante saisonnière est périodique
— le bruit est de variance faible.

Dans le cas de polynôme de degré supérieur à 1, on peut aussi utiliser des moyennes mobiles autres qu'arithmétiques laissant invariants les polynômes de degré supérieur à 1.

Certains auteurs préconisent également d'utiliser la méthode des moyennes mobiles comme technique de lissage de la série quelle que soit la forme de la tendance. On peut cependant utiliser d'autres techniques plus adaptées pour lisser la série (telle que la méthode de loess).

5.2.1 Etapes de décomposition par moyennes mobiles

La décomposition et l'étude de la série statistique (X_t) en vue de la prédiction se font selon les étapes suivantes :

1. application d'une moyenne mobile d'ordre judicieusement choisi
2. estimation de la saisonnalité
3. estimation de la tendance
4. itération éventuelle de la procédure
5. prévision des valeurs futures
6. analyse des résidus

5.2.1.1 Série lissée par moyenne mobile

Tout d'abord, on applique une moyenne mobile arithmétique d'ordre $2m+1$ dans le cas d'une saisonnalité d'ordre impair $2m+1$ ou une moyenne mobile arithmétique modifiée d'ordre $2m+1$ dans le cas d'une saisonnalité de période paire $2m$. Dans chaque cas, on a vu que la saisonnalité est ainsi annulée et que la variance du bruit est diminuée. Si le modèle est bon, la série transformée ne contient plus aucun mouvement saisonnier. Notons que la série ainsi obtenue est de longueur $T-2m$ et notée $\widehat{X}_t$.

Les données étant mensuelles, nous appliquons donc une moyenne mobile arithmétique d'ordre 12.

```
# Moyenne mobile arithmétique d'ordre 12
elecequip["12-MA"] = convolvefilter
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(elecequip.elecequip,color = "black", label = "elecequip");
axe.plot(elecequip["12-MA"],color = "red",linestyle = "--",label = "12-MA");
axe.set_xlabel("année");
axe.set_ylabel("Indice des nouvelles commandes");
axe.legend();
plt.show()
```

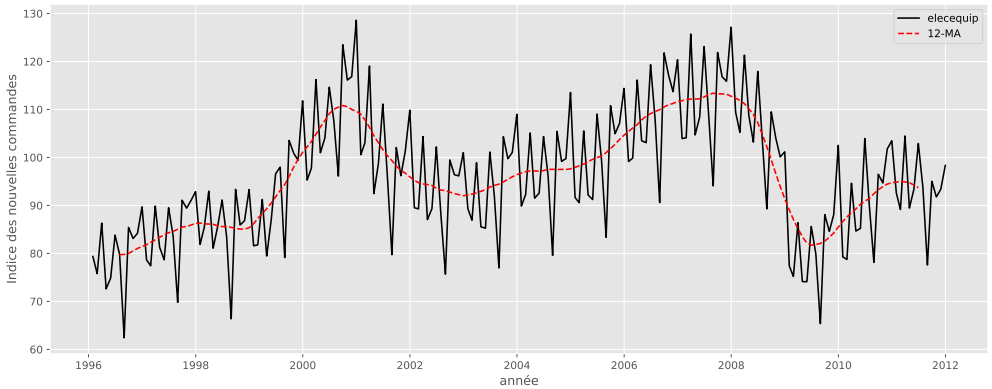


Figure 5.15 – Un 2x12-MA appliqué à l'indice des commandes de matériel électrique

La figure 5.15 montre un 2×12-MA appliqué à l'indice des commandes de matériel électrique.

5.2.1.2 Estimation de la saisonnalité

Le calcul des coefficients saisonniers doit respecter le principe de conservation des aires ($\sum_{j=1}^{j=p} c_j = 0$ dans le cas d'un schéma additif ou $\sum_{j=1}^{j=p} c_j = p$ dans le cas d'un schéma multiplicatif). Ainsi, le calcul de la saisonnalité s'effectue de deux manières différents :

— **Schéma additif**

On calcule la série diminuée de sa tendance $\widetilde{S}_t$ en retranchant à la série de départ la série transformée $\widehat{X}_t$. Plus précisement, supposons que les observations sont périodiques de périodes $p = 2m$ paire et réalisées sur N périodes. A l'issue du premier filtrage, on dispose de la série :

$$\widetilde{S}_t = X_t - \widehat{X}_t, \quad t = m+1, \dots, pN-m \tag{5.66}$$

— **Schéma multiplicatif**

Ici, on divise à la série de départ la série transformée $\widehat{X}_t$:

$$\widetilde{S}_t = X_t / \widehat{X}_t, \quad t = m+1, \dots, pN-m \tag{5.67}$$

Cette série, $\widetilde{S}_t$, est appelée série corrigée de la tendance. Disposons ces valeurs dans un tableau

Table 5.5 – Tableau de la composante saisonnière

		périodes								
		1	···	m	···	j	···	$p-m+1$	···	p
Année	1	$***$	$***$	$***$						
	⋮									
	i					$\widetilde{S}_{j+p(i-1)}$				
	⋮									
	N							$***$	$***$	$***$

On estime ensuite les p coefficients saisonnier par la moyenne des valeurs de $\widetilde{S}_t$ correspondant à chaque temps de la période. L'estimation du coefficient saisonnier c_j est donc :

$$\tilde{c}_j = \begin{cases} \dfrac{1}{N-1} \displaystyle\sum_{i=2}^{i=N} \widetilde{S}_{j+p(i-1)} & 1 \le j \le m \\ \dfrac{1}{N-1} \displaystyle\sum_{i=1}^{i=N-1} \widetilde{S}_{j+p(i-1)} & 1 \le j \le p \end{cases} \tag{5.68}$$

Cette opération consiste à appliquer à la série $(\widetilde{S}_t)_{t=m+1,\dots,pN-m}$ à laquelle on ajoute p observations nulles pour $t = 1, \dots, m$ et $t = pN-m+1, \dots, pN$, une moyenne mobile d'ordre $p(N-1)+1$ et de coefficients

$$\frac{1}{N-1}\left(\overbrace{1, \underbrace{0, \dots, 0}_{p-1}, 1, 0, \dots, 0, 1, \underbrace{0, \dots, 0}_{p-1}, 1}^{N-1}\right) \tag{5.69}$$

On peut affiner la méthode en retranchant (ou en divisant) ensuite à chaque coefficient estimé la moyenne des coefficients afin que le principe de conservation des aires soit respecté. Les estimateurs des p coefficients saisonniers sont ensuite :

$$\hat{c}_j = \begin{cases} \tilde{c}_j - \dfrac{1}{p}\displaystyle\sum_{j=1}^{j=p} \tilde{c}_j & \text{schéma additif} \\ p\dfrac{\tilde{c}_j}{\displaystyle\sum_{j=1}^{j=p} \tilde{c}_j} & \text{schéma multiplicatif} \end{cases} \tag{5.70}$$

Exemple 5.19 *Estimation de la saisonnalité et des coefficients saisonniers*

On supposera que le schéma de décomposition de la série `elecequip` est multiplicatif.

```
# Estimation de la saisonnalité
elecequip["St"] = elecequip["elecequip"]/elecequip["12-MA"]
```

Afin de pouvoir calculer les coefficients, on transforme la série $(\widetilde{S}_t)_{t=m+1,\ldots,pN-m}$ en donnée infra - annuelle de format $(N \times p)$.

```
# Transformation en série infra-annuelle
p = 12
N = int(elecequip.shape[0]/p)
import calendar
data = pd.DataFrame(elecequip["St"].values.reshape(N,p))
data.columns = list([calendar.month_abbr[x] for x in range(1,13)])
data.index = list(np.unique(pd.DatetimeIndex(elecequip.index).year))
```

Table 5.6 – Composante saisonnière en infra - annuelle

	janv.	févr.	mars	avr.	mai	juin	juil.	août	sept.	oct.	nov.	déc.
1996	NaN	NaN	NaN	NaN	NaN	NaN	1.001	0.782	1.068	1.032	1.039	1.102
1997	0.961	0.941	1.085	0.975	0.938	1.062	0.989	0.821	1.065	1.044	1.060	1.077
1998	0.948	0.991	1.080	0.942	0.999	1.065	0.975	0.777	1.095	1.010	1.021	1.093
1999	0.947	0.937	1.034	0.889	0.962	1.053	1.052	0.839	1.078	1.030	0.998	1.106
2000	0.932	0.946	1.108	0.950	0.966	1.051	0.978	0.870	1.114	1.050	1.062	1.173
2001	0.922	0.956	1.120	0.884	0.958	1.094	0.958	0.802	1.040	0.988	1.047	1.145
2002	0.940	0.942	1.105	0.922	0.949	1.092	0.946	0.813	1.072	1.042	1.042	1.097
2003	0.970	0.941	1.068	0.920	0.914	1.078	0.975	0.815	1.099	1.045	1.053	1.131
2004	0.929	0.951	1.082	0.941	0.953	1.073	0.987	0.816	1.082	1.017	1.023	1.163
2005	0.936	0.922	1.070	0.930	0.916	1.091	0.989	0.825	1.088	1.021	1.032	1.093
2006	0.940	0.941	1.087	0.960	0.949	1.093	0.985	0.824	1.102	1.055	1.022	1.078
2007	0.929	0.929	1.121	0.934	0.966	1.093	0.963	0.830	1.076	1.031	1.024	1.127
2008	0.974	0.940	1.091	0.988	0.947	1.100	0.990	0.873	1.100	1.077	1.066	1.107
2009	0.869	0.864	1.014	0.888	0.902	1.048	0.976	0.796	1.067	1.015	1.045	1.199
2010	0.915	0.899	1.069	0.948	0.944	1.144	0.983	0.845	1.034	1.008	1.078	1.092
2011	0.976	0.939	1.101	0.944	0.991	1.098	NaN	NaN	NaN	NaN	NaN	NaN

Nous supprimons les années qui contiennent les valeurs manquantes et ensuite calculons les moyennes par mois :

```
# Suppression des valeurs manquantes
data = data.dropna()
# Calcul des moyennes
c_tilde = data.apply(lambda x : np.mean(x),0)
# Principe de conservation des aires
c_hat = c_tilde.div(c_tilde.mean())
```

Table 5.7 – Composante saisonnière en infra - annuelle

janv.	févr.	mars	avr.	mai	juin	juil.	août	sept.	oct.	nov.	déc.
0.937	0.936	1.082	0.934	0.948	1.082	0.982	0.825	1.08	1.032	1.041	1.121

```
# Somme des coefficients
print(c_hat.sum())
```

```
## 12.000000000000002
```

Le principe des aires est respecté : $\sum_{j=1}^{j=p} \hat{c}_j = 12$.

```
# Représentation graphique
fig, axe =plt.subplots(figsize=(16,6))
axe.plot(elecequip["St"],color= "black");
axe.set_xlabel("année");
axe.set_ylabel("Indice des nouvelles commandes");
axe.set_title("composante saisonnière");
plt.show()
```

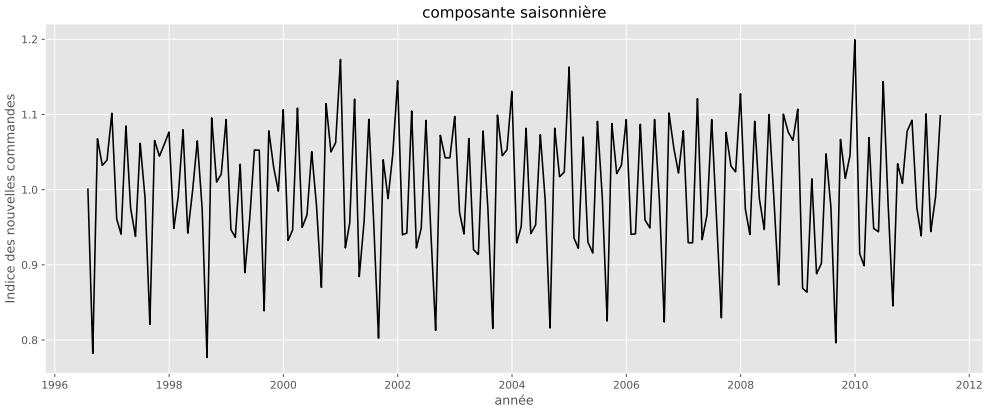


Figure 5.16 – Composante saisonnière

5.2.1.3 Estimation de la tendance

Une fois les coefficients saisonniers estimés à l'étape précédente, on retranche (ou divise) l'estimation du saisonnier à la série X_t. La série ainsi obtenue est appelée série corrigée des valeurs saisonnières ou séries déssaisonnalisées :

$$X_{CVS,t} = \begin{cases} X_t - \widehat{S}_t & \text{schéma additif} \\ X_t / \widehat{S}_t & \text{schéma multiplicatif} \end{cases} \quad t = 1, \dots, T \tag{5.71}$$

avec $\widehat{S}_t = \hat{c}_j, t \equiv [p]$.

Remarque 5.8

Notons que si l'on avait utilisé $\widetilde{S}_t$ au lieu de $\widehat{S}_t$, nous aurions simplement obtenu $\widehat{X}_t$ pour $t = m+1, \dots, T-m$ au lieu de $X_{CVS,t}$. Mais ce résultat aurait été moins précis puisque nous aurions eu des valeurs t seulement de $m+1$ à T.

On procède ensuite à l'estimation du terme représentant la tendance par une méthode de régression : on modélise le plus souvent la tendance par un polynôme $Z_\theta(t)$ (en général un polynôme de degré 1 pour être cohérent avec le choix fait lors de la première étape) où θ est le vecteur des paramètresà estimer. Puis on ajuste au sens des moindres carrés ordinaires un polynôme $\widehat{Z}_t = Z_{\hat{\theta}}(t)$ à la série corrigée des valeurs saisonnières $X_{CVS,t}$. On peut également utiliser la méthode des points médians.

Par la méthode des moindres carrés, on cherche à estimer la droite :

$$Z_\theta(t) = a + bt, \quad t = 1, \dots, T \tag{5.72}$$

Les estimateurs $\hat{a}$ et $\hat{b}$ sont donnés par le système suivante :

$$\begin{cases} \hat{a} &= \dfrac{\mathrm{Cov}(t, X_{CVS})}{\mathrm{Var}(t)} \\ \hat{b} &= \overline{X}_{CVS} - \hat{a}\overline{t} \end{cases} \tag{5.73}$$

De plus, la qualité de la régression est mesurée par le coefficient de détermination $\rho^2_{t,X_{CVS}}$ défini par

$$\rho^2_{t,X_{CVS}} = \frac{\mathrm{Cov}^2(t, X_{CVS})}{\mathrm{Var}(t)\mathrm{Var}(X_{CVS})} \tag{5.74}$$

Exemple 5.20 *Série corrigée des valeurs saisonnières pour la série elecequip*

On applique la formule : $X_{CVS,t} = X_t/\widehat{S}_t$, $t = 1, \dots, T$.

```
# Série corrigée des valeurs saisonnières
elecequip["X_CVS"] = elecequip["elecequip"].div(np.tile(c_hat, N))
# Représentation graphique
fig, axe =plt.subplots(figsize=(16,6))
axe.plot(elecequip.elecequip,color = "black",label="$X_{t}$");
axe.plot(elecequip["12-MA"],color = "blue",label="$\widehat{X}_{t}$");
axe.plot(elecequip["X_CVS"],color="red",linestyle="--",label="$X_{CVS,t}$");
axe.set_xlabel("année");
axe.set_ylabel("Indice des nouvelles commandes");
axe.legend();
plt.show()
```

```
# Estimation des valeurs de a et b
elecequip["t"] =  list([x for x in range(1,elecequip.shape[0]+1)])
import statsmodels.formula.api as smf
model = smf.ols('X_CVS~t', data = elecequip).fit()
print(model.summary2())
```

```
##                  Results: Ordinary least squares
```

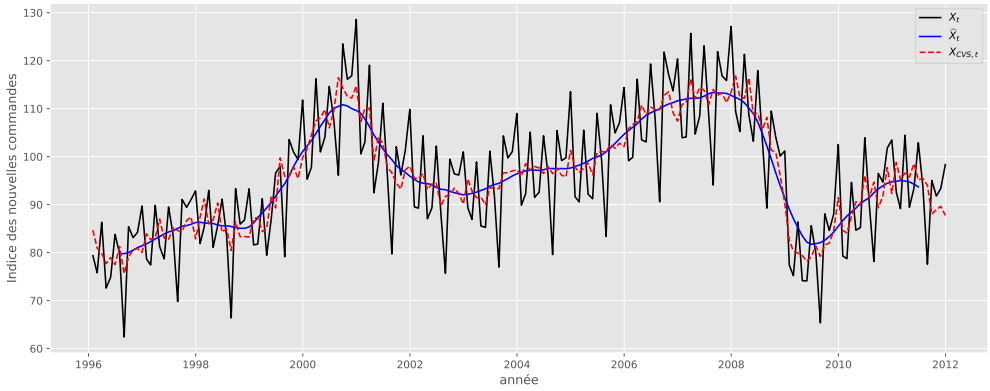


Figure 5.17 – Série corrigée des valeurs saisonnières de la série elecequip

```
## ==============================================================================
## Model:                OLS              Adj. R-squared:     0.082
## Dependent Variable:   X_CVS            AIC:                1437.4395
## Date:                 2023-08-06 22:40 BIC:                1443.9545
## No. Observations:     192              Log-Likelihood:     -716.72
## Df Model:             1                F-statistic:        18.08
## Df Residuals:         190              Prob (F-statistic): 3.33e-05
## R-squared:            0.087            Scale:              103.38
## ------------------------------------------------------------------------------
##                 Coef.   Std.Err.     t     P>|t|    [0.025   0.975]
## ------------------------------------------------------------------------------
## Intercept      90.3479    1.4733  61.3223  0.0000  87.4417  93.2540
## t               0.0563    0.0132   4.2516  0.0000   0.0302   0.0824
## ------------------------------------------------------------------------------
## Omnibus:               8.922        Durbin-Watson:           0.104
## Prob(Omnibus):         0.012        Jarque-Bera (JB):        4.641
## Skew:                  0.154        Prob(JB):                0.098
## Kurtosis:              2.303        Condition No.:           223
## ==============================================================================
## Notes:
## [1] Standard Errors assume that the covariance matrix of the
## errors is correctly specified.
```

```
# Coefficient de détermination
print("Coefficient de détermination : %.4f"%(100*model.rsquared)+"%")
```

```
## Coefficient de détermination : 8.6873%
```

Tous les coefficients sont significatifs. Cependant, notre modèle a un faible pouvoir explicatif ($\rho^2 = 9\%$).

5.2.1.4 Itération de la procédure

On procède parfois à une itération de la procédure : on estime à nouveau la tendance à partir de la série $X_{CVS,t}$ en utilisant une moyenne mobile d'ordre différent de celui utilisé à l'étape 1.

Soit $\widehat{X}_{CVS}$ cette estimation. On retranche (ou divise) à la série X_t cette tendance et on revient à l'étape 2 pour une seconde estimation du saisonnier.

5.2.1.5 Prévision des valeurs futures

Afin de prévoir les valeurs futures de la série, on utilise l'estimation de la tendance et celle de la composante saisonnière. Plus précisement, si on souhaite prévoir une valeur de la série à l'instant $T+h$, où $h \geq 1$, c'est - à - dire à l'horizon h, on utilise les estimations de la tendance et de la saisonnalité et on pose :

$$\widehat{X}_T(h) = \begin{cases} \widehat{Z}_{T+h} + \hat{c}_j & \text{(schéma additif)} \\ \widehat{Z}_{T+h} \times \hat{c}_j & \text{(schéma multiplicatif)} \end{cases}, T+h \equiv j[p] \tag{5.75}$$

Exemple 5.21 *Prévision de la série elecequip*

On souhaite prévoir les valeurs de la série pour les 3 premiers mois de l'année 2012.

```
## Prévision
# Création de la période
indexpred = pd.date_range(start="2012-01-31",periods = 3,freq="M")
tpred = pd.DataFrame(np.arange(1,4),index = indexpred,columns = ["t"])
# Calcul des valeurs ajustée pour la prédiction
ypreddf = model.predict(tpred)
ypred = np.array(ypreddf).reshape(len(tpred),1)
# Lissage avec les coefficients saisonniers
xpred = ypred*np.array(c_hat.values[:len(tpred)]).reshape(len(tpred),1)
xpreddf = pd.DataFrame(xpred,index = indexpred, columns = ["forecast"])
```

Table 5.8 – Valeurs prévues par moyennes mobiles

	forecast
2012-01-31	84.723
2012-02-29	84.683
2012-03-31	97.895

```
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(elecequip.elecequip,color='black',label = "$X_{t}$");
axe.plot(elecequip["12-MA"],color='blue',linestyle ="dashdot",
         label = "$\widehat{X}_{t}$");
axe.plot(elecequip["X_CVS"],color='red',linestyle = "--",label = "$X_{CVS}$");
axe.plot(xpreddf.forecast,color='yellow',label = "$\widehat{X}_{T}(h)$");
axe.set_xlabel('mois');
axe.set_ylabel('Indice des nouvelles commandes');
axe.legend();
plt.show()
```

Remarque 5.9

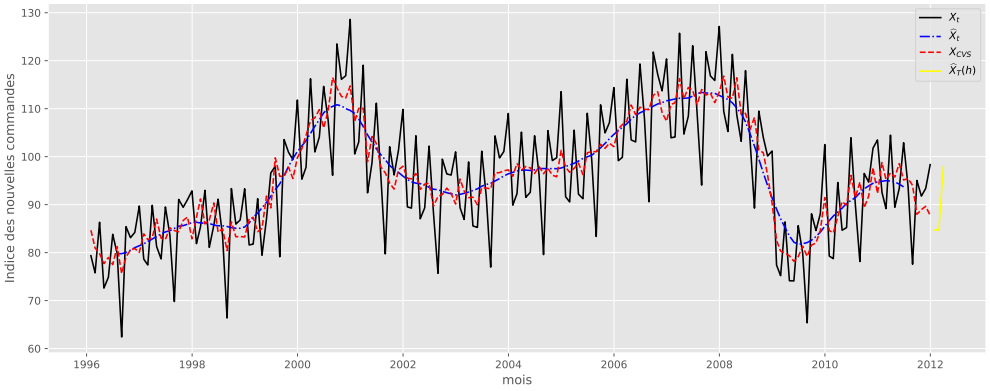


Figure 5.18 – Prévision par moyenne mobile

Cette méthode de prévision a un inconvenient majeur qui est celui d'une prévision ne tenant pas compte des valeurs les plus récentes de la série : celles - ci ont été « éliminées » par application de moyennes mobiles.Pour cette raison, on utilise plus souvent la méthode comme méthode de désaisonnalisation que comme méthode de prévision.

5.2.1.6 Analyse des résidus

Une fois estimées les composantes du modèle, on peut contrôler la pertinence du modèle par une analyse des résidus. Ceux - ci sont définis par :

$$\hat{\varepsilon}_t = X_t - \widehat{S}_t - \widehat{Z}_t = \widehat{X}_{CVS,t} - \widehat{Z}_t, \quad t = 1, \dots, T \tag{5.76}$$

Si le modèle est bon, il ne doit rester dans les résidus aucune trace du saisonnier. Pour le valider, on trace le corrélogramme des résidus c'est - à - dire legraphe d'un estimateur de la fonction d'autocorrélation.

En théorie, le corrélogramme n'est tracé que dans le cas où la série est stationnaire, ce qui implique en particulier qu'il n'y a dans cette série ni tendance ni saisonnalité. En pratique, on s'en sert (dans le cas de l'analyse des résidus) pour vérifier justement l'absence de saisonnalité dans les résidus.

Si c'est le cas et si le modèle est bon, lecorrélogramme ne doit présenter que de svaleurs faibles, indiquant une faible corrélation entre les erreurs. Si au contraire,le corrélogramme présente des pics régulièrement espacés, cela indique que le saisonnier n'a pas été complètement éliminé et c'est donc le signe que le modèle proposé a échoué. On peut alors réitérer la procédure ci - dessus ou proposer un autre modèle.

Dans le cas oùle corrélogramme des résidus n'indique pas la présence d'un mouvement saisonnier, on trace le graphe des résidus $(t, \hat{\varepsilon}_t)$ qui sert à repérer d'éventuelles observations exceptionnelles, un mouvement tendanciel…

Remarque 5.10

Dans le cas de l'hypothèse d'erreurs gaussiennes, on vérifie celle - ci en traçant l'histogramme des résidus, un QQ - Plot ou encore en effectuant un test de normalité.

Dans le cas où l'erreur du modèle est un bruit blanc,l'utilisation d'une moyenne mobile introduit des corrélations dans le processus transformé.

Proposition 5.6 *Périodogramme*

Pour mettre en évidence une éventuelle périodicité dans les résidus, on peut également tracer le périodogramme, qui est le graphe de la projection du vecteur $X = (X_1, \dots, X_T)'$ sur le sous - espace vectoriel de $\mathbb{R}^T$ engendré par les deux vecteurs

$$C = \begin{pmatrix} \cos(\omega) \\ \cos(2\omega) \\ \vdots \\ \cos(T\omega) \end{pmatrix} \quad \text{et} \quad S = \begin{pmatrix} \sin(\omega) \\ \sin(2\omega) \\ \vdots \\ \sin(T\omega) \end{pmatrix} \tag{5.77}$$

où ω est la fréquence. On cherche ainsi à ajuster la série X_t le $t^{\text{ième}}$ d'une série trigonométrique, dite Fourier, appelée une harmonique, et s'écrivent :

$$s(t) = a\cos(\omega t) + b \sim (\omega t) \tag{5.78}$$

Propriété 5.12

Dans le cas où la série X_t est de période p, on a $p = 2\pi/\omega$.

Une approximation de la norme de projection est

$$\mathcal{Q}(\omega) = \frac{T}{2}\left(\hat{a}^2 + \hat{b}^2\right) \tag{5.79}$$

où

$$\begin{cases} \hat{a}^2 & \dfrac{2}{T}\displaystyle\sum_{t=1}^{t=T} X\cos(\omega t) \\ \hat{b}^2 & \dfrac{2}{T}\displaystyle\sum_{t=1}^{t=T} X\sin(\omega t) \end{cases} \tag{5.80}$$

Le périodogramme est donc legraphe de la fonction $(\omega, \mathcal{Q}(\omega))$. On cherche alors la valeur ω^* qui maximise $\mathcal{Q}(\omega)$ (c'est - à - dire qui minimise la norme de l'erreur de projection). Si le périodogramme présente un pic en ω^*, on lit cette vaaleur sur un graphe et on retient une période $T = 2\pi/\omega^*$ ou plus exactement la valeur entière la plus proche de cette expression.

5.2.2 Fonction Seasonal decompose de Statsmodels

Sous Python l'une des manières de décomposer une série chronologique X_t par la méthode des moyennes mobiles arithmétiques est d'utiliser la fonction .seasonal_decompose() de la librairie Statsmodels en spécifiant le type de schéma de décomposition de la série, la période de la saisonnalité ainsi que le caractère symétrique ou non de la moyenne mobile. Elle est plus efficace, rapide et plus optimale. Dans cette fonction, l'estimation de le tendance se fait en appliquant un filtre de convolution.

```
# Estimation de la tendance par moyenne mobile
import statsmodels.tsa.api as smt
decompose = smt.seasonal_decompose(elecequip.elecequip,model = "multiplicative",
                                   period = 12,two_sided = False)
```

```
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(elecequip.elecequip,color='black', label = "$X_{t}$");
axe.plot(decompose.trend,color='red',linestyle ="--",label="$\widehat{X}_{t}$");
axe.set_xlabel('mois');
axe.set_ylabel('Indice des nouvelles commandes');
axe.legend();
plt.show()
```

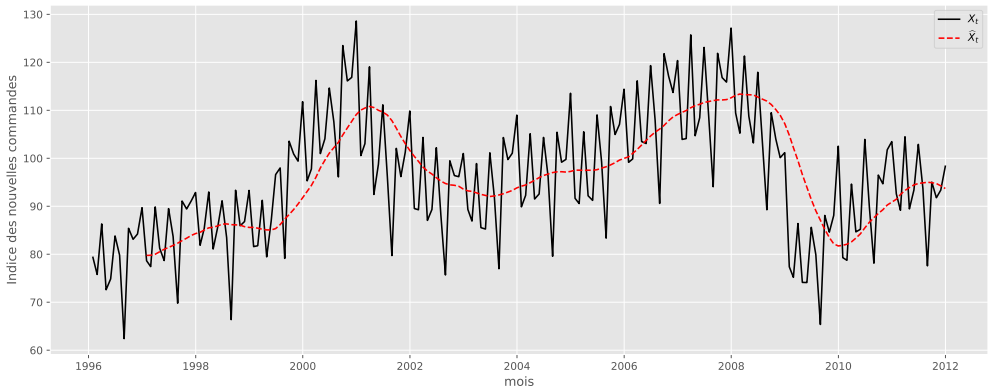
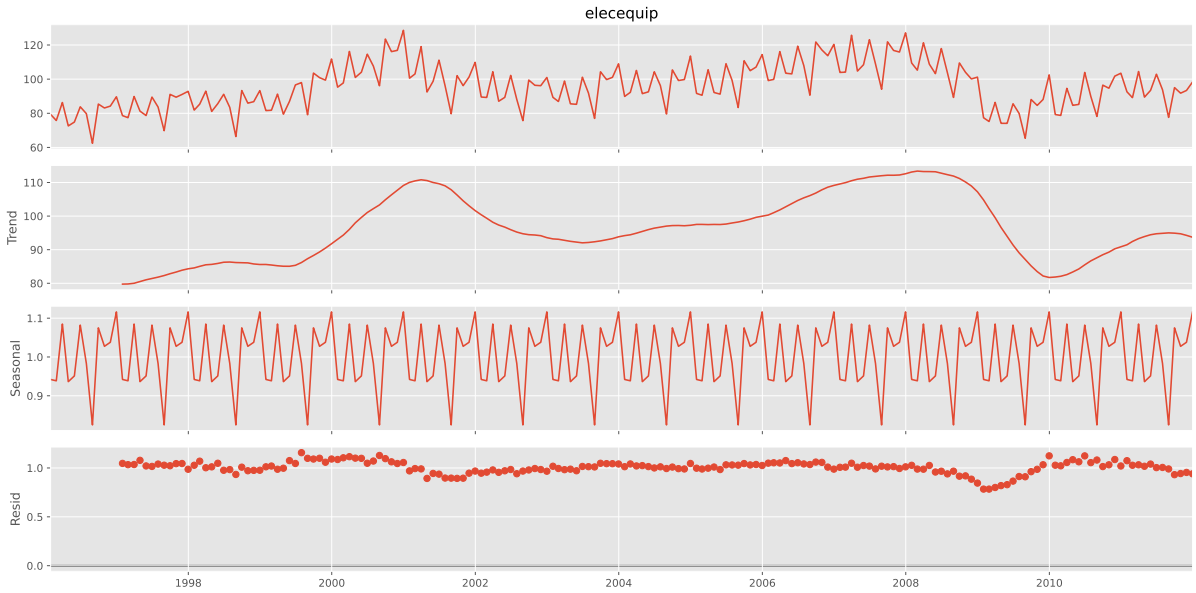


Figure 5.19 – Estimation de la tendance par Moyenne mobile arithmétique d'ordre 12

L'intérêt particulier de cette fonction est qu'elle calcule et retourne également les coefficients saisonniers decompose.seasonal ainsi que les résidus estimés decompose.resid.

```
# Ajustement par moyenne mobile
fig = decompose.plot()
fig.set_size_inches((16,8));
fig.tight_layout();
plt.show()
```



Remarque 5.11

Les moyennes mobiles présentent quelques inconvénients qui sont :

1. **Influence des valeurs aberrantes**

   Comme toute moyenne, les moyennes mobiles peuvent être influencées par la présence des valeurs aberrantes dans la série.

2. **Perte de données**

   Si on a les données d'une série chronologique sur n années avec $n \times p$ observations, on ne peut calculer une estimation de la tendance que pour $np - (p + 1)$ ou $np - p$ observations selon la parité de p. On omet donc environ une année (si la série est observée annuellement).

Malgré ces inconvénients, la moyenne mobile donne une meilleure estimation de la tendance que la méthode des moindres carrés ordinaires.

Les moyennes mobiles permettent de lisser directement la série sans hypothèse a priori sur le modèle sous - jacent. La méthode est donc valable quel que soit le modèle de décomposition. Pour cette raison, on peut classer ce type de lissage dans les méthodes non - paramétriques (par opposition aux méthodes paramétriques). C'est un outil simple à mettre en oeuvre qui met en évidence l'allure de la tendance en supprimant la composante saisonnière et en atténuant le bruit.

Dans tout ce qui précède, on peut remplacer la moyenne par la médiane et on obtient alors un lissage par médiane mobile. Ce procédé a alors l'avantage d'être moins sensible aux valeurs aberrantes.

6 Prévision par lissage exponentiel

Sommaire

Introduites par Holt (1957) ainsi que par Winters (1960) et popularisées par Brown (1959), les méthodes de lissage exponentiel constituent l'ensemble des techniques empiriques de prévision qui accordent plus ou moins d'importance aux valeurs du passé d'une série temporelle. Elles consistent à extrapoler une série en vue de faire des prévisions. Soit $(X_t)_{t=1,\dots,T}$ une série chronologique, on se trouve à la période T et on désire faire des prévisions à l'horizon h pour les dates $T+h$ avec $h>0$. On notera par $\widehat{X}_T(h)$ la valeur prédicte de X_t à l'horizon $h \in \mathbb{N}^*$.

Les méthodes de lissage exponentiel non saisonnier supposent que la série chronologique de départ est structurée de la façon suivante :

$$X_t = f_t^{(p)} + \varepsilon_t, \quad \forall t \tag{6.1}$$

avec $f_t^{(p)}$ une fonction polynômiale de degré p dont les paramètres dépendent du temps et ε_t un bruit blanc. Ces méthodes se différencient les unes des autres selon de degré de $f_t^{(p)}$ et les degrés les plus utilisés sont $p=0$ et $p=1$.

6.1 Les lissages exponentiels

6.1.1 Lissage exponentiel simple

6.1.1.1 Formulation générale

Le lissage exponentiel simple permet d'effectuer des prévisions pour des séries chronologiques dont la tendance est constante et sans saisonnalité. La série s'écrit :

$$X_t = a_t + \varepsilon_t, \quad \forall t \tag{6.2}$$

avec $a_t = a$ constante si $t \in \{T_1;T_2\} \subset \{1;T\}$. L'ensemble $\{T_1;T_2\}$ porte le nom d'ensemble (ou intervalle) local. Cet intervalle peut correspondre au pas Δt de la chronique ou encore à la période de prévision.

Soit (X_t) une telle série dont on a observé les T premiers instants $X_1, ..., X_T$. Pour $T \in \mathbb{N}^*$, on cherche à prévoir la valeur X_{T+h}, c'est - à - dire faire une prévision de la série à l'horizon h.

Etant donné un réel α tel que $0 \leq \alpha \leq 1$, comme la tendance est constante, on cherche une prévision $\widehat{X}_T(h)$ sous la forme de la constante qui s'ajuste le mieux au sens des moindres carrés pondérés au voisinage de T, c'est - à - dire la solution du problème de minimisation :

$$\min_a \sum_{j=0}^{j=T-1} (1-\alpha)^j \left(X_{T-j} - a\right)^2 \tag{6.3}$$

Remarque 6.1

Notons que dans l'expression à minimiser l'influence des observations décroit lorsqu'on s'éloigne de la date T. Les dernières observations sont prises en compte ce qui constitue un avantage majeur par rapport à la méthode de prévision des moindres carrés ordinaires.

Définition 6.1 *Prévision par lissage exponentiel simple*

La prévision de la série à l'horizon h, notée $\widehat{X}_T(h)$, fournie par la méthode de lissage exponentiel simple est donnée par :

$$\widehat{X}_T(h) = \alpha \sum_{j=0}^{j=T-1} (1-\alpha)^j X_{T-j} \tag{6.4}$$

α est appelé **constante de lissage** comprise entre 0 et 1 et dépend de l'horizon h.

Remarque 6.2

1. Dans le cas où α est indépendant de l'horizon h, on notera simplement $\widehat{X}_T$ au lieu de $\widehat{X}_T(h)$. Les prévisions sont dans ce cas identiques pour tout h.
2. La méthode du lissage exponentiel simple prend en compte tout le passé d'une série temporelle, mais en accordant de moins en moins d'importance aux observations les plus éloignées de l'instant T. On donne un poids d'autant moins important que les observations sont éloignées (plus j est grand, plus $(1-\alpha)^j$ est faible).
3. La valeur de la constante de lissage α permet de nuancer la remarque précédente :

   Si α est proche de 1, la prévision est **souple**, c'est - à - dire fortement influencée par les observations les plus récentes ($(1-\alpha)^j$ devenant négligeable pour les grandes de valeurs de j). Dans le cas extrême où $\alpha = 1$, la prévision est alors égale à la dernière valeur observée.

   En revanche, si α est proche de 0, l'influence des observations passées est d'autant plus importante et remonte loin dans le passé. On dit dans ce cas que la prévision est **rigide** en ce sens qu'elle est peu sensible aux fluctuations exceptionnelles (aussi appelées fluctuations conjoncturelles). Dans l'autre cas extrême où $\alpha = 0$, alors toutes les prévisions sont identiques (et donc à la valeur choisie pour l'initialisation).

   En pratique, on prend $\alpha \in]0, 1[$ afin d'exclure ces deux cas extrêmes dégénérés.
4. En général, pour une prévision de long terme, on prend α proche de 0 et pour une prévision de court terme, on prend α proche de 1.

La figure (6.1) illustre le poids décroissant accordé au passé en fonction de différentes valeurs de α.

```
import pandas as pd
import numpy as np
# Poids
def weight(alpha,j):
  return alpha*((1-alpha)**j)
weight_df = pd.DataFrame({
  r"$\alpha$=0.30" : np.array(list(map(lambda i : weight(0.30,i),range(11)))),
  r"$\alpha$=0.60" : np.array(list(map(lambda i : weight(0.60,i),range(11)))),
  r"$\alpha$=0.80" : np.array(list(map(lambda i : weight(0.80,i),range(11)))),
  r"$\overline{\alpha}$" : np.repeat(a=0.10,repeats=11)
},index = range(11))
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(weight_df.iloc[:,0],color="black",label=r"$\alpha$=0.30",marker = "s");
axe.plot(weight_df.iloc[:,1],color="blue",label=r"$\alpha$=0.60",marker = "|");
axe.plot(weight_df.iloc[:,2],color="red",label=r"$\alpha$=0.80",marker = "^");
axe.plot(weight_df.iloc[:,3],color='gray', label = "Moyenne",marker="x");
axe.set_xlim(0,10);
axe.legend();
axe.set_xlabel("j");
axe.set_ylabel(r"$\alpha(1-\alpha)^{j}$");
plt.show()
```

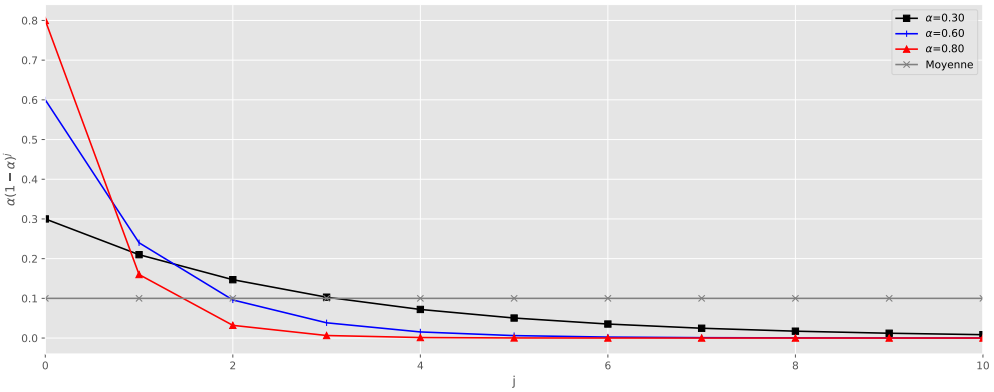


Figure 6.1 – Pondération décroissante de l'information avec son ancienneté : comparaison entre une moyenne classique (poids identique = 0.10) et trois valeurs de α (poids géométriquement décroissant).

Exemple 6.1 *Lissage exponentiel simple de la série Nile (1871 - 1970)*

Nous considérons le jeu de données Nile contenu dans la librairie pydataset de Python. Ces données contiennent 100 mesures du débit annuel du Nil à Aswan entre 1871 et 1970 (en $10^8 m^3$).

```
# Chargement des données
from pydataset import data
nile = data("Nile").set_index("time")
```

```
nile.index = pd.date_range(start='31/12/1871',periods=len(nile),freq="Y")
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(nile, color = "black");
axe.set_xlabel("année");
axe.set_ylabel("débit du Nil (1e9 $m^{3}$)");
plt.show()
```

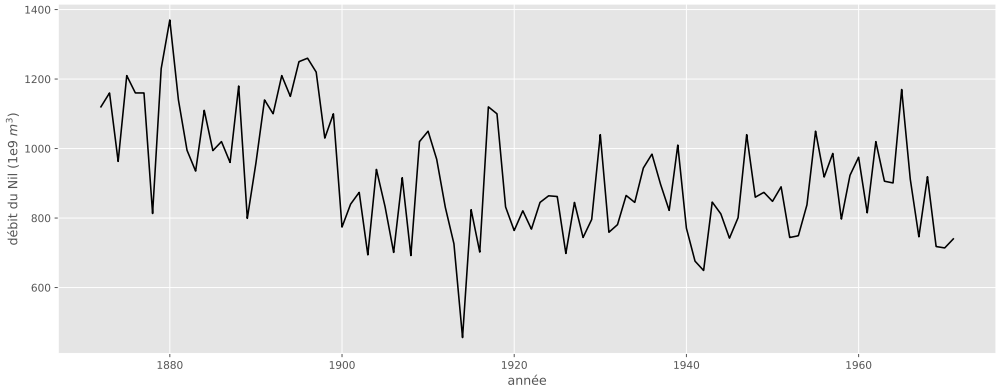


Figure 6.2 – Evolution de la série Nile (1871 - 1970)

On ajuste cette série par la méthode du lissage exponentiel simple en prenant $\alpha = 0.2$ et $\alpha = 0.8$.

```
# Lissage exponentiel simple
def LES(x,alpha):
    T = len(x)
    X = np.array(x)
    hat = np.zeros(T)
    hat[0] = float(x.mean())
    for t in range(1,T):
        hat[t] = alpha*X[t]+(1-alpha)*hat[t-1]
    hat = pd.DataFrame(hat,columns=["fittedvalues"],index=x.index)
    return hat

# Application
les1 = LES(x = nile, alpha = 0.2)
les2 = LES(x = nile, alpha = 0.8)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(nile,color="black",label = "nile");
axe.plot(les1,color="blue",linestyle = "dashed",label=r"$\alpha=0.2$");
axe.plot(les2,color="red",linestyle = "dashdot",label=r"$\alpha=0.8$");
axe.set_xlabel("année");
axe.set_ylabel("débit du Nil (1e9 $m^{3}$)");
axe.legend();
plt.show()
```

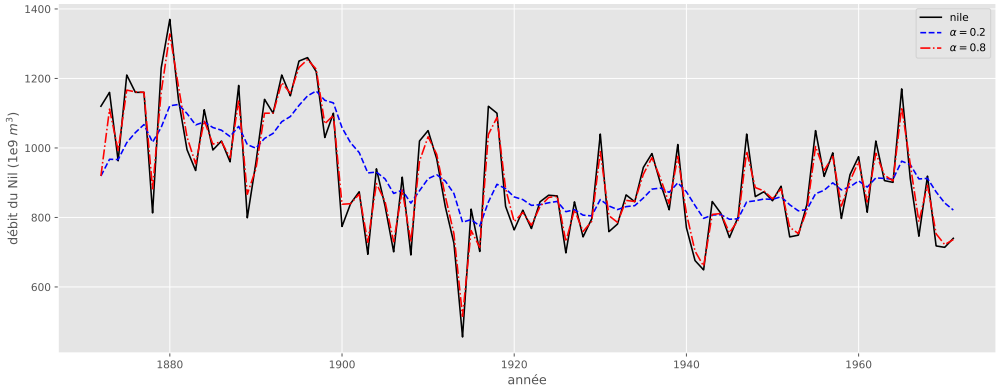


Figure 6.3 – Prévision par lissage exponentiel simple de la série Nile (1871 - 1970)

6.1.1.2 Choix de la constante de lissage

Un problème important en pratique est celui du choix de la constante de lissage α qui est en général fondé sur des critères subjectifs et varie selon le contexte de l'étude et/ou le type de prévision souhaité. En pratique, si on souhaite faire une prévision souple, on choisira $\alpha \in [0.7; 0.99]$ et si au contraire on souhaite une prévision rigide, on souhaite $\alpha \in [0.01; 0.3]$. Une autre solution, dictée par les données, consiste à choisir α comme la solution du problème des moindres carrés ordinaires suivant :

$$\sum_{t=1}^{t=T-h} \left(X_{t+h} - \widehat{X}_t(h)\right)^2 \tag{6.5}$$

c'est - à - dire de minimiser la somme des carrés des erreurs de prévision aux dates $1, \dots, T-h$.

On peut aussi ne considérer que les écarts obtenus sur la deuxième moitié de la série, afin de ne pa tenir compte de l'initialisation. On cherche alors α qui minimise :

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmin}} \left\{ \sum_{t=1}^{t=T-1} \left[X_{t+1} - \alpha \sum_{j=0}^{j=t-1} (1-\alpha)^j X_{t-j} \right]^2 \right\} \tag{6.6}$$

Le α optimal tient compte de la qualité de la prévision à l'ordre. Si $h \geq 2$, on pourrait aussi obtenir un coefficient de lissage plus adapté en minimisant :

$$\alpha \mapsto \sum_{t=1}^{t=T-h} \left(X_{t+h} - \alpha \sum_{j=0}^{j=t-1} (1-\alpha)^j X_{t-j} \right)^2 \tag{6.7}$$

Exemple 6.2 *Choix de α pour la série Nile (1871 - 1970)*

On définit la somme des carrés des résidus :

$$SCR = \sum_{t=1}^{t=T-1} \left[X_{t+1} - \widehat{X}_{t+1} \right]^2$$

```
# Somme des carrés des résidus
def scr_les(alpha,x):
    T = len(x)
    X = np.array(x)
    hat = np.zeros(T)
    hat[0] = float(x.mean())
    err = np.zeros(T)
    err[0] = 0
    for t in range(1,T):
        hat[t] = alpha*X[t]+(1-alpha)*hat[t-1]
        err[t] = X[t] - hat[t-1]
    return np.square(err).sum()

# Application
pas = np.linspace(0,1,100)
scr = np.array(list(map(lambda i : scr_les(alpha=i,x=nile),pas)))
fig, axe = plt.subplots(figsize=(16,6));
axe.plot(pas,scr,color= "black");
axe.set_xlabel(r"$\alpha$");
axe.set_ylabel("Somme des carrés des résidus");
plt.show()
```

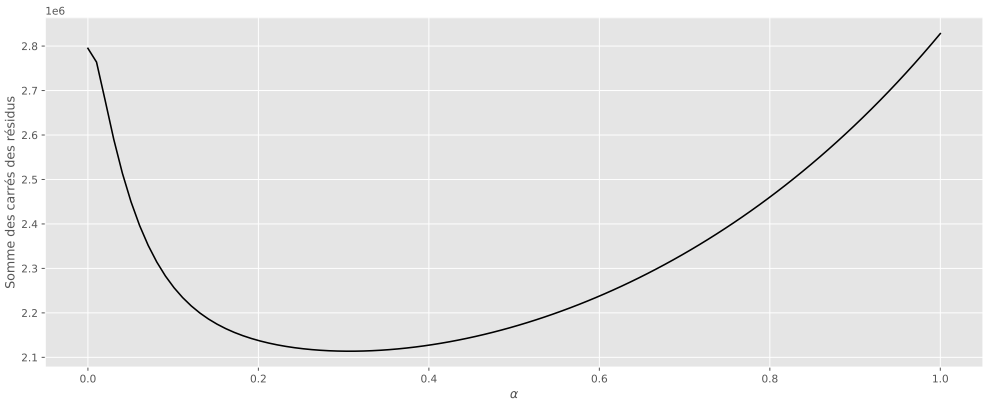


Figure 6.4 – Evolution de la somme des carrés des résidus en fonction de α

La courbe SCR en fonction de α est une parabole dont le sommet se trouve dans l'intervalle $[0.2; 0.4]$. On détermine la valeur de ce sommet, c'est - à - dire $\hat{\alpha}$.

```
# alpha optimal pour la série nile sous lissage exponentiel simple
from scipy.optimize import minimize, LinearConstraint
const_les = LinearConstraint(1,lb=0,ub=1)
res = minimize(scr_les,x0=0.01,args=(nile),method="L-BFGS-B",
               constraints=const_les).x
# LES avec alpha optimal
les3  = LES(x=nile,alpha= float(res))
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(nile,color="black",label = "nile");
axe.plot(les1,color="blue",linestyle = "dashed",label=r"$\alpha=0.2$");
axe.plot(les2,color="red",linestyle = "dashdot",label=r"$\alpha=0.8$");
```

```
axe.plot(les3,color="green",linestyle = "dashdot",
         label=r"$\widehat{\alpha}$=%s"%float(res));
axe.set_xlabel("année");
axe.set_ylabel("débit du Nil (1e9 $m^{3}$)");
axe.legend();
plt.show()
```

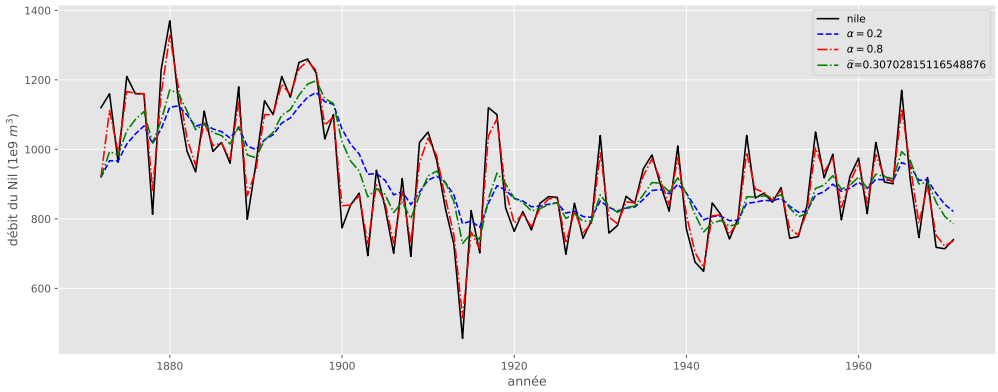


Figure 6.5 – Lissage exponentiel simple de la série Nile avec valeur optimale de α

6.1.1.3 Formule de mise à jour

A partir de la définition donnée par l'equation (6.4), on déduit les équations suivantes :

$$\widehat{X}_T(h) = (1-\alpha)\widehat{X}_{T-1}(h) + \alpha X_T \tag{6.8}$$

$$= \widehat{X}_{T-1}(h) + \alpha\left(X_T - \widehat{X}_{T-1}(h)\right) \tag{6.9}$$

La première égalité est appelée **formule de mise à jour** et permet de calculer directement (à partir de la prévision $\widehat{X}_{T-1}$ à la date $T-1$) une nouvelle prévision $\widehat{X}_T$ lorsqu'une nouvelle observation X_T est effectuée. Sous cette forme le lissage exponentiel simple apparaît comme une moyenne pondérée de la dernière réalisation et de la dernière valeur lissée

Remarque 6.3

1. Cette équation permet de mettre à jour les prévisions à l'horizon h à partir de la dernière prévision de manière extrêmement simple. L'initialisation de la récurrence est en général faite en prenant $\widehat{X}_1(h) = X_1$. Un autre choix possible consiste à prendre la moyenne arithmétique. Si T est assez grand, ce choix a en fait peu d'importance. Le choix de l'initialisation est rapidement « oublié ». Cet oubli est d'autant plus rapide que la constante de lissage est proche de 0.
2. La première équation fait apparaître $\widehat{X}_T(h)$ comme le barycentre entre $\widehat{X}_{T-1}(h)$, la valeur prédite à l'horizon h à partir des $T-1$ premières observations et X_T la dernière observation.
3. La seconde équation fait intervenir le terme $\left(X_T - \widehat{X}_{T-1}(h)\right)$ la dernière erreur de prévision. Dans cette équation, le lissage apparaît comme le résultat de la dernière valeur lissée corrigé par une pondération de l'écart entre la réalisation et la prévision.

A un instant donné de l'historique, le calcul de $\widehat{X}_T$ est réalisable puisque tous les éléments de la formule sont connus. Pour la prévision de la chronique $(T+1, \dots, T+h$,avec $T+h$ l'horizon de prévision du modèle) ce calcul ne peut être effectué que pour le premier pas du temps. En effet, pour $h = 1$, on a :

$$\widehat{X}_T(1) = (1-\alpha)\widehat{X}_{T-1}(1) + \alpha X_T$$

En revanche, pour $h = 2$, on a :

$$\widehat{X}_T(2) = (1-\alpha)\widehat{X}_T(1) + \alpha X_{T+1}$$

La valeur de X_{T+1} est inconnue. La seule information dont nous disposons est sa prévision $\widehat{X}_T(1)$. En substituant cette valeur dans l'équation, on obtient $\widehat{X}_T(2) = \widehat{X}_T(1)$ et plus généralement $\widehat{X}_T(h) = \widehat{X}_T(1)$, $\forall h \geq 1$. Les valeurs prévues à partir d'un lissage exponentiel simple sont donc identiques entre elles et égales à celle prévue en $h = 1$.

Il est possible de calculer le **délai moyen de réaction** ou **âge moyen de l'information** qui est la moyenne pondérée des coefficients de lissage :

$$\overline{\alpha} = \frac{\sum_{j=0}^{\infty} j\alpha(1-\alpha)^j}{\sum_{j=0}^{\infty} \alpha(1-\alpha)^j} \tag{6.10}$$

or par construction $\sum_{j=0}^{\infty} \alpha(1-\alpha)^j = 1$. D'où : $\overline{\alpha} = \sum_{j=0}^{\infty} j\alpha(1-\alpha)^j$

Calculons cette somme infinie :

$$\begin{aligned}
\overline{\alpha} &= \alpha(1-\alpha) + 2\alpha(1-\alpha)^2 + 3\alpha(1-\alpha)^3 + \cdots \\
&= \alpha\left[(1-\alpha) + 2(1-\alpha)^2 + 3(1-\alpha)^3 + \cdots\right] \\
&= \alpha S
\end{aligned}$$

Calculons l'expression : $S - (1-\alpha)S = \alpha S$

$$\begin{aligned}
S &= (1-\alpha) + 2(1-\alpha)^2 + 3(1-\alpha)^3 + \cdots \\
-(1-\alpha)S &= -(1-\alpha)^2 - 2(1-\alpha)^3 - 3(1-\alpha)^4 - \cdots \\
\alpha S &= (1-\alpha) + (1-\alpha)^2 + (1-\alpha)^3 + \cdots \\
&= (1-\alpha)\left[1 + (1-\alpha) + (1-\alpha)^2 + \cdots\right] \\
&= (1-\alpha) \times \frac{1}{1-(1-\alpha)}
\end{aligned}$$

D'où :

$$\overline{\alpha} = \frac{1-\alpha}{\alpha} \tag{6.11}$$

Si $\alpha = 1$, l'âge moyen est nul puisque seule la dernière valeur prise en compte ; si $\alpha = 0$ l'âge moyen est infini puisque seule la valeur initiale est prise en compte.

6.1.1.4 Intervalle de confiance de la prévision

On rappelle que : $\widehat{X}_T(h) = \alpha \sum_{j=0}^{\infty} (1-\alpha)^j X_{T-j}$

On calcule Var $\left(\widehat{X}_T(h)\right)$:

$$\text{Var}\left(\widehat{X}_T(h)\right) = \alpha^2 \text{Var}\left(\sum_{j=0}^{\infty} (1-\alpha)^j X_{T-j}\right)$$

Or $X_T = a_T + \varepsilon_T$, d'où sur un intervalle local $\text{Var}(X_T) = \sigma_\varepsilon^2$ et $\forall T$ et $T^{'}(T \neq T^{'})$. Les variables X_T et $\widehat{X}_T$ sont indépendantes. D'où :

$$\begin{aligned}
\text{Var}\left(\widehat{X}_T(h)\right) &= \alpha^2 \text{Var}\left(X_T\right) \sum_{j=0}^{\infty} (1-\alpha)^{2j} \\
&= \alpha^2 \text{Var}\left(X_T\right) \left[1 + (1-\alpha)^2 + (1-\alpha)^4 + \cdots\right] \\
&= \alpha^2 \text{Var}\left(X_T\right) \frac{1}{1-(1-\alpha)^2} \\
&= \frac{\alpha}{2-\alpha} \text{Var}\left(X_T\right)
\end{aligned}$$

Nous avons donc :

$$\frac{\text{Var}\left(\widehat{X}_T(h)\right)}{\text{Var}\left(X_T\right)} = \frac{\alpha}{2-\alpha} \tag{6.12}$$

Ainsi, dans le cas où $\alpha = 1$, $\left(\widehat{X}_T(h)\right)$ a la même variance que (X_T). Ainsi, pour assurer un rôle de filtrage efficace, il convient de choisir un α faible.

La figure 6.6 présente, en ordonnée les valeurs du rapport de l'écart type de la série lissée à l'écart type de la série brute $\left(\sqrt{\frac{\alpha}{2-\alpha}}\right)$ et, en abscisse, l'âge du lissage pour différentes valeurs de α.

```
def mean_alpha(x) :
  return (1-x)/x
def ponderation(x) :
  return np.sqrt(x/(2-x))
age_mean = pd.DataFrame({
"moyenne" : np.array(list(map(lambda i : mean_alpha(i),np.linspace(0,1,20)))),
"coef": np.array(list(map(lambda i : ponderation(i),np.linspace(0,1,20))))
},index = [i for i in range(20)])
# Représentation graphique
fig,axe = plt.subplots(figsize=(16,6))
axe.plot(age_mean.moyenne,age_mean.coef,marker="s",color="black");
```

```
axe.set_xlabel("âge moyen de l'information");
axe.set_ylabel(r"$\sqrt{\dfrac{\alpha}{2-\alpha}}$");
axe.text(2,0.9, r"$\alpha=0.84$",fontsize=12);
axe.text(4,0.6, r"$\alpha=0.63$",fontsize=12);
axe.text(7.5,0.4, r"$\alpha=0.21$",fontsize=12);
axe.arrow(2,0.9,-1.5,-0.09,head_width=0.05,head_length=0.3,fc='k',ec='k',
          color='black');
axe.arrow(4,0.6,-1.5,-0.09,head_width=0.05,head_length=0.3,fc='k',ec='k',
          color='black');
axe.arrow(7.5,0.4,-1.5,-0.09,head_width=0.05,head_length=0.3,fc='k',ec='k',
          color='black');
plt.show()
```

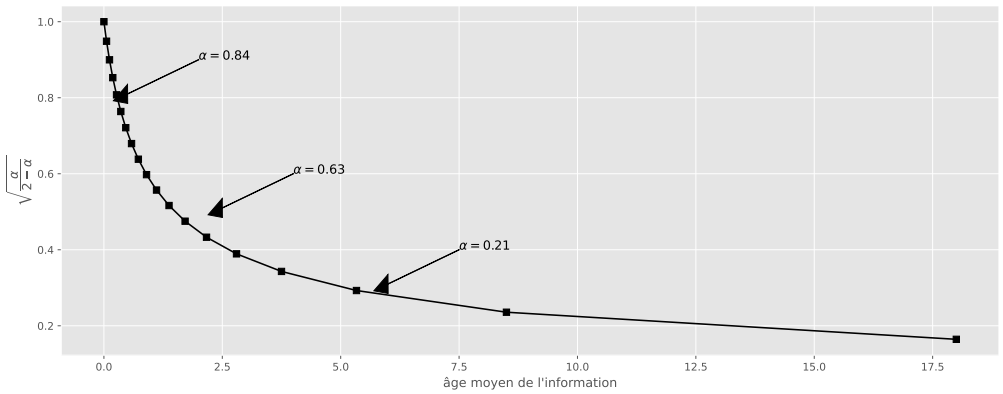


Figure 6.6 – Valeur du coefficient α, âge moyen de l'information et filtrage.

A la lecture de la figure 6.6, lorsque le coefficient α est inférieur à 0.20, l'âge du lissage s'accroît très vite pour un gain faible de filtrage ; au - delà de 0.4, la qualité du filtrage se décroît très vite pour une faible réduction élevée de l'âge moyen.

L'erreur de prévision effectuée en $T+h$ pour T est :

$$e_{T+h} = X_{T+h} - \widehat{X}_T(h) \tag{6.13}$$

Ainsi, la variance de l'erreur de prévision est égale à :

$$\mathrm{Var}(e_{T+h}) = \mathrm{Var}(X_{T+h}) + \mathrm{Var}\left(\widehat{X}_T(h)\right)$$

car X_{T+h} et $\widehat{X}_T(h)$ sont indépendantes. D'où :

$$\begin{aligned}
\mathrm{Var}(e_{T+h}) &= \mathrm{Var}(X_{T+h})\left(1 + \frac{\alpha}{2-\alpha}\right) \\
&= \mathrm{Var}(X_{T+h})\left(\frac{2}{2-\alpha}\right) \\
&= \frac{2}{2-\alpha}\sigma_\varepsilon^2
\end{aligned}$$

L'intervalle de confiance à $100(1-\beta)\%$ de la valeur prévue de X_{T+h} s'écrit donc :

$$X_{T+h} \in \left[\widehat{X}_T(h) \pm z_{\beta/2}\sigma_\varepsilon\sqrt{\frac{2}{2-\alpha}}\right] \tag{6.14}$$

avec $z_{\beta/2}$ la valeur de la loi normale centrée réduite. Par exemple, pour un intervalle de confiance à 95%, nous avons :

$$X_{T+h} \in \left[\widehat{X}_T(h) \pm 1.96\sigma_\varepsilon\sqrt{\frac{2}{2-\alpha}}\right] \tag{6.15}$$

Sous Python, on réalise un lissage exponentiel simple grâce à la fonction SimpleExpSmoothing de la librairie Statsmodels. Nous avons les équation suivantes :

— Equation de prévision : $\widehat{X}_{t+h|t} = l_t$
— Equation de lissage : $l_t = \alpha X_t + (1-\alpha)l_{t-1}$

où l_t est le niveau de la série à l'instant t et α représente la constante de lissage comprise entre 0 et 1. [1].

Exemple 6.3 *Calcul d'une prévision par un lissage exponentiel simple*

Soit les données observées de la série X_t consignées sur le tableau 6.1. On demande de calculer une prévision par un lissage exponentiel simple - assortie d'un intervalle de confiance à 95% - à un horizon de trois périodes. La constante de lissage α est égale 0.3.

Table 6.1 – Prévision à partir du modèle de lissage exponentiel simple

t	X_t	$\widehat{X}_t$	$\hat{\varepsilon}_t = X_t - \widehat{X}_t$
1	30	30	0
2	40	30.00	10.00
3	40	33.00	7.00
4	30	35.10	– 5.10
5	20	33.57	– 13.57
6	20	29.50	– 9.50
7	30	26.65	3.35
8	30	27.65	2.35
h		$\widehat{X}_T(h)$	
1		28.36	
2		28.36	
3		28.36	

On initialise : $\widehat{X}_1 = X_1 = 30$

Pour $t = 2$: $\widehat{X}_2 = 0.3X_1 + 0.7\widehat{X}_1 = 30$

Pour $t = 3$: $\widehat{X}_3 = 0.3X_2 + 0.7\widehat{X}_2 = 0.3 \times 40 + 0.7 \times 30 = 33$

...

Pour $t = 8$: $\widehat{X}_8 = 0.3X_7 + 0.7 \times \widehat{X}_7 = 0.3 \times 30 + 0.7 \times 26.65 = 27.65$

Pour $t = 9$: $\widehat{X}_9 = 0.3 \times X_8 + 0.7\widehat{X}_8 = 28.36$

Pour $t = 10, 11$: $\widehat{X}_{10} = \widehat{X}_{11} = \widehat{X}_9 = 28.36$.

Nous avons $\sigma_X = \sqrt{\frac{1}{T-1}\sum_{t=1}^{t=T}(X_t - \overline{X})^2}$ (formule des petits échantillons) et $\sqrt{\frac{2}{2-\alpha}} = 1.085$.

1. Pour plus d'informations, voir https ://otexts.com/fpp2/ses.html

D'où :

$$\begin{aligned} IC &= \widehat{X}_t \pm 1.96\sigma_X\sqrt{\frac{2}{2-\alpha}} \\ &= 28.36 \pm 1.96 \times 7.56 \times 1.085 \\ &= 28.36 \pm 16.08 \\ &= [12.28; 44.44] \end{aligned}$$

Exemple 6.4 *Prévision de la série Air passengers par lissage exponentiel simple*

Nous utilisons à présent la série Airpassengers qui mesure le nombre mensuel de passangers aériens, en milliers, de janvier 1949 à décembre 1960. Dans la suite, on notera X_t, la série Air passengers et $Y_t = \ln(X_t)$.

```
# Chargement de la base de données
import seaborn as sns
x = sns.load_dataset("flights").drop(columns = ["year","month"])
x.index = pd.date_range(start="1949-01-31",periods= len(x),freq="M")
y = x.apply(lambda i : np.log(i),axis=0)
fig, axe = plt.subplots(figsize=(16,6))
ln1 = axe.plot(x,color='red',label="nb passagers");
axe2 = axe.twinx();
ln2 = axe2.plot(y,color='blue',label="log(nb passagers)");
lns = ln1 + ln2
labels=[l.get_label() for l in lns];
axe.legend(lns, labels);
plt.tight_layout();
plt.show()
```

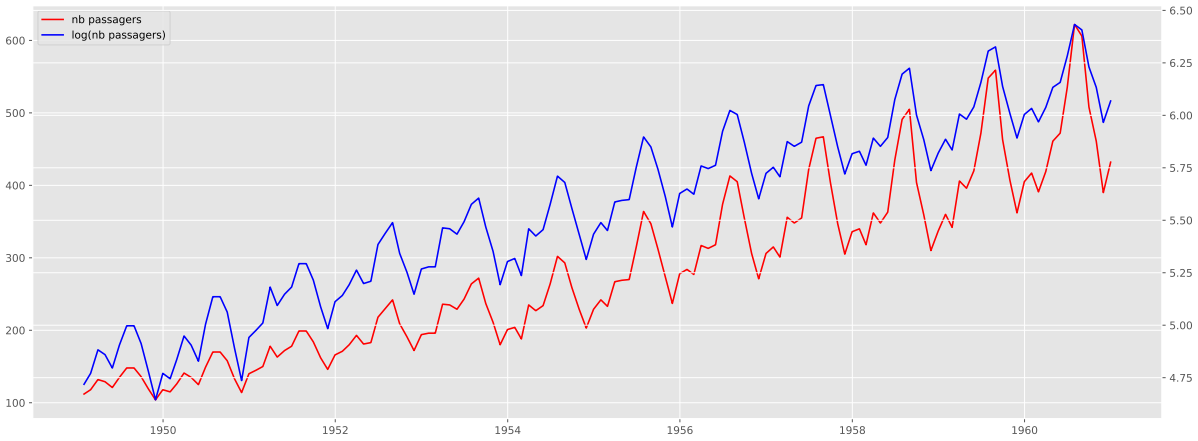


Figure 6.7 – Série Air passengers (série initiale et série en logarithme)

On voit sur la série initiale (en rouge) que l'amplitude des variations augmente chaque année de manière exponentielle (Figure 6.7). Or dans un cadre théorique parfait, on voudrait une variance constante au cours du temps. Pour s'en approcher, on peut passer par la fonction logarithme. En effet, comme $\log(ab) = \log(a) + \log(b)$, le log transforme un modèle **multiplicatif** en un modèle **additif**. Ainsi, l'accroissement de l'amplitude des pics (donc de la variance) pour la série X disparaît avec la transformation logarithmique. C'est visible sur la courbe bleue.

```
# Lissage exponentiel avec alpha = 0.8
from statsmodels.tsa.api import SimpleExpSmoothing
fit1 =  SimpleExpSmoothing(y,initialization_method="heuristic").fit(
                            smoothing_level=0.8, optimized=False)
# Lissage exponentiele avec alpha = 0.2
fit2 = SimpleExpSmoothing(y,initialization_method="heuristic").fit(
                           smoothing_level=0.2, optimized=False)
fig, (axe1, axe2) = plt.subplots(1,2,figsize=(16,6))
axe1.plot(y,color="black");
axe1.plot(fit1.fittedvalues,color="blue",label = r"$\alpha=0.8$");
axe1.set_xlabel("année");
axe1.set_ylabel("passagers");
axe1.legend();
axe2.plot(y,color="black");
axe2.plot(fit2.fittedvalues,color="red",label = r"$\alpha=0.2$");
axe2.set_xlabel("année");
axe2.set_ylabel("passagers");
axe2.legend();
plt.tight_layout()
plt.show()
```

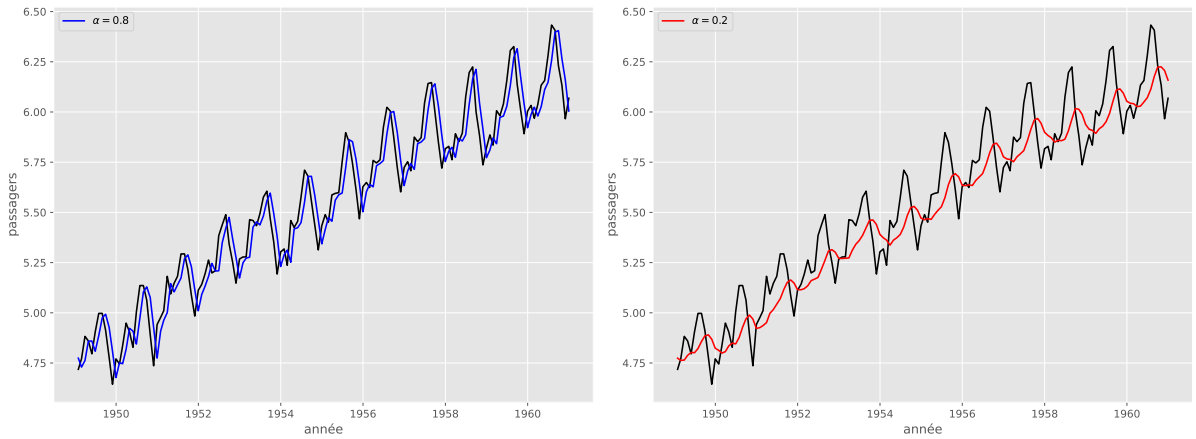


Figure 6.8 – Influence de la constante de lissage pour le lissage exponentiel simple : série temporelle initiale (trait noir), séries lisées avec $\alpha = 0.8$ (trait bleu) et avec $\alpha = 0.3$ (trait rouge).

La figure 6.8 illustre le comportement du lissage exponentiel simple en prenant la constante de lissage égale à 0.8 (lissage souple) ou 0.2 (lissage rigide). Sur les deux graphiques, la courbe noire représente la série initiale, la courbe bleue la série lissée avec $\alpha = 0.8$ (à gauche) et la courbe rouge la série lissée avec $\alpha = 0.2$ (à droite).

```
# Constante de lissage alpha = 0.8
fit11 =  SimpleExpSmoothing(y,initialization_method = "known",
           initial_level = y.values[0]).fit(smoothing_level=0.8,optimized=False)
fit12 =  SimpleExpSmoothing(y,initialization_method = "known",
            initial_level = np.mean(y)).fit(smoothing_level=0.8,optimized=False)
# Constante de lissage alpha = 0.2
fit21 = SimpleExpSmoothing(y,initialization_method = "known",
          initial_level = y.values[0]).fit(smoothing_level=0.2, optimized=False)
fit22 = SimpleExpSmoothing(y,initialization_method = "known",
```

```
            initial_level = np.mean(y)).fit(smoothing_level=0.2, optimized=False)
# Représentation graphique
fig, (axe1,axe2) = plt.subplots(1,2,figsize=(16,6))
axe1.plot(y,color="black");
axe1.plot(fit11.fittedvalues,color="blue",
            label = r"$\alpha=0.8$ et $\widehat{X}_{1}(h) = X_{1}$");
axe1.plot(fit12.fittedvalues,color="red",
            label = r"$\alpha=0.8$ et $\widehat{X}_{1}(h) = \overline{X}$ ");
axe1.set_xlabel("année");
axe1.set_ylabel("passagers");
axe1.legend();
axe2.plot(y,color="black");
axe2.plot(fit21.fittedvalues,color="blue",
            label = r"$\alpha=0.2$ et $\widehat{X}_{1}(h) = X_{1}$");
axe2.plot(fit22.fittedvalues,color="red",
            label = r"$\alpha=0.2$ et $\widehat{X}_{1}(h) = \overline{X}$ ");
axe2.set_xlabel("année");
axe2.set_ylabel("passagers");
axe2.legend();
plt.tight_layout()
plt.show()
```

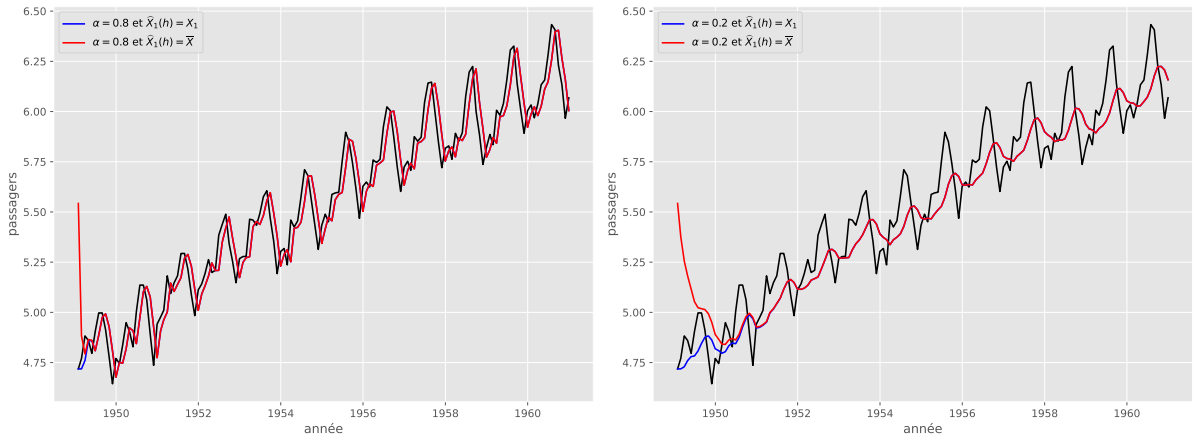


Figure 6.9 – Influence de l'initialisation pour le lissage exponentiel simple : série temporelle initiale (courbe noire), séries lissées avec $\hat{X}_1(h) = X_1$ ou $\hat{X}_1(h) = \overline{X}$ avec $\alpha = 0.8$ (a gauche) et $\alpha = 0.2$ (à droite)

La figure 6.9 illustre l'influence de l'initialisation de l'équation de mise à jour ; cette influence dépend aussi de la constante de lissage (à gauche avec $\alpha = 0.8$ et à droite avec $\alpha = 0.2$). Sur les deux groupes, la série initiale apparaît en noir. La courbe bleue correspond au lissage en prenant $\widehat{X}_1(h) = X_1$ et la courbe rouge correspond au lissage en prenant $\widehat{X}_1(h) = \overline{X}$. On s'aperçoit que quelle que soit la valeur initiale, la prévision est la même au bout d'un certain temps. En revanche, ce temps au bout duquel les prévisions coïncident est d'autant plus petit que α est proche de 0. Ces remarques permettent de nuancer certains choix automatiques. Par exemple, Python effectue par défaut un lissage exponentiel simple avec pour valeur initiale la première valeur de la série.

```
# Constante de lissage optimale
les = SimpleExpSmoothing(y,initialization_method="estimated").fit()
alpha = les.params["smoothing_level"]
```

```
# Prédiction par lissage exponentiel simple
y_pred = les.forecast(24).rename(r"$\alpha=%s$"%alpha)
y_pred.index = pd.date_range(y.index[len(y)-1],periods=24,freq='M')
x_pred = y_pred.apply(lambda i : np.exp(i))
# Représentation graphique
fig, (axe1, axe2) = plt.subplots(1,2,figsize=(16,6))
axe1.plot(y,color = "black",label='log(Air passengers)');
axe1.plot(les.fittedvalues,color = "blue",linestyle = "--",
          label = r"$\widehat{\alpha}=%s$"%alpha);
axe1.plot(y_pred,color = "red",label='Pred');
axe1.set_xlabel("année");
axe1.set_ylabel("passagers");
axe1.legend();
axe2.plot(x, color="black",label='Air passengers');
axe2.plot(np.exp(les.fittedvalues),color = "blue",linestyle = "--",
          label = r"$\widehat{\alpha}=%s$"%alpha);
axe2.plot(x_pred,color = "red",label='Pred');
axe2.set_xlabel("année");
axe2.set_ylabel("passagers");
axe2.legend();
plt.tight_layout();
plt.show()
```

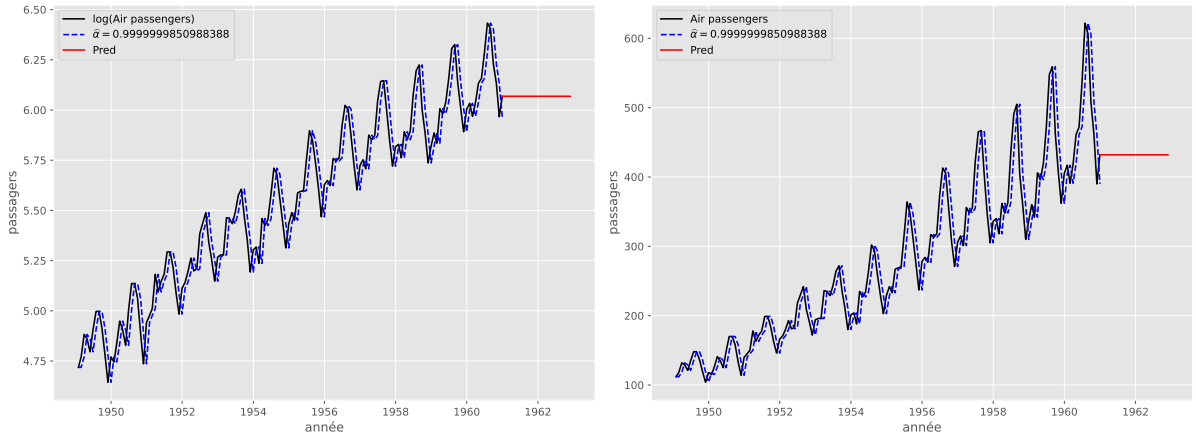


Figure 6.10 – Prévision par lissage exponentiel simple

On constate que la value prévue est la même quelle que soit la période h ($h \geq 1$).

Exemple 6.5 *Prévision à partir du lissage exponentiel simple de la série oil*

Soit les données observées de la série oil contenue dans le package fpp2 de R. Nous pouvons l'importer grâce à la méthode get_rdataset() de la librairie Statsmodels. On demande de calculer une prévision par une méthode de lissage exponentiel simple à un horizon de cinq périodes.

```
# Chargement des données
import statsmodels.api as sm
oil = sm.datasets.get_rdataset("oil", "fpp2").data.set_index("time")
oildata = oil[oil.index>=1996].rename(columns={"value": "oil"})
```

```
oildata.index = pd.date_range(start='31-12-1996',periods=len(oildata),freq="Y")
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(oildata,color="black");
axe.set_ylabel("Essence (millions de tonnes)");
axe.set_xlabel("année");
plt.show()
```

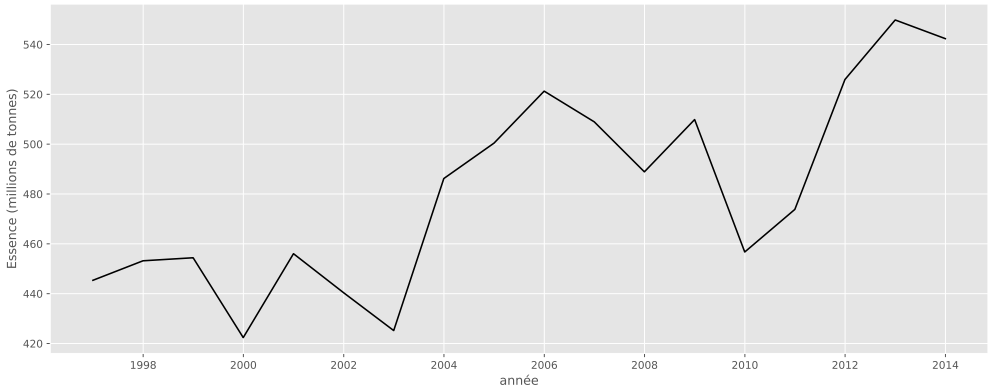


Figure 6.11 – Production de pétrole en Arabie Saoudite de 1996 à 2013.

les données de la figure 6.11 n'affichent aucun comportement de tendance claire ni aucune saisonnalité.

```
# Prévision par lissage exponentiel simple
oil_les = SimpleExpSmoothing(oildata,initialization_method="estimated").fit()
oil_alpha = oil_les.model.params["smoothing_level"]
# Prévision
oil_fcast = pd.DataFrame(oil_les.forecast(5),columns = ["forecast"])
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(oildata,color="black",label = "oil");
axe.plot(oil_les.fittedvalues,c="green",label=r"$\widehat{\alpha}=%s$"%oil_alpha);
axe.plot(oil_fcast.forecast,color="green");
axe.set_ylabel("Essence (millions de tonnes)");
axe.set_xlabel("année");
axe.legend();
plt.show()
```

6.1.2 Lissage exponentiel double

6.1.2.1 Présentation de la méthode

Le lissage exponentiel simple est adapté à des séries pouvant être ajustées par une constante au voisinage de T. Le lissage exponentiel double généralise l'idée du lissage exponentiel simple au cas où la série peut être ajustée par une droite au voisinage de T. On cherche dans ce cas une prévision à l'horizon h, $\widehat{X}_T(h)$ de la forme :

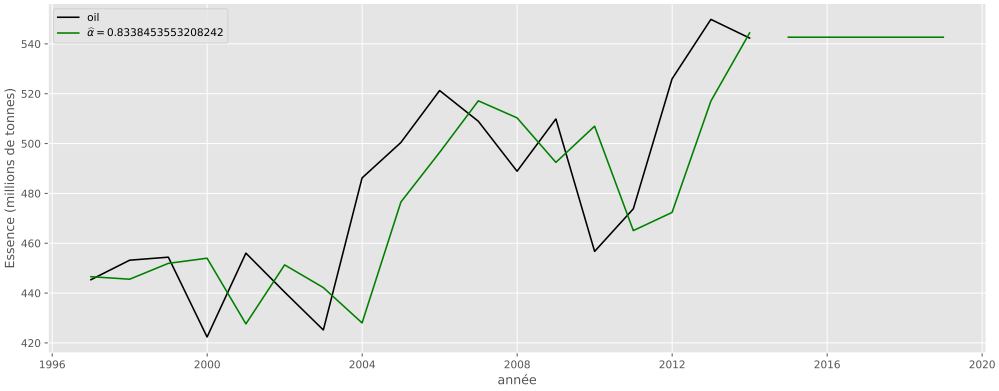


Figure 6.12 – Lissage exponentiel simple appliqué à la production de pétrole en Arabie Saoudite (1996–2013).

$$\widehat{X}_T(h) = l_T + b_T h \tag{6.16}$$

avec

$$\begin{cases} l_T &= l_{T-1} + b_{T-1} + (1-(1-\alpha)^2)\left(X_T - \widehat{X}_{T-1}(T)\right) \\ b_T &= b_{T-1} + (1-\alpha)^2(X_T - \widehat{X}_{T-1}(T)) \end{cases} \tag{6.17}$$

Remarque 6.4

l_t et b_t minimisent la somme des carrés des erreurs de prévision à chaque instant :

$$\min_{l,b} \mathcal{Q} = \sum_{j=0}^{\infty} (1-\alpha)^j \left(X_{t-j} - (l - bj)\right)^2 \tag{6.18}$$

La solution de ce problème s'obtient en annulant les dérivées partielles de la fonction $\mathcal{Q}$ par rapport à l et b. On obtient :

$$\begin{cases} \dfrac{\partial}{\partial l}\mathcal{Q}(l,b) &= -2\displaystyle\sum_{j=0}^{\infty}(1-\alpha)^j\left(X_{t-j} - l + bj\right) \\ \dfrac{\partial}{\partial b}\mathcal{Q}(l,b) &= 2\displaystyle\sum_{j=0}^{\infty} j(1-\alpha)^j\left(X_{t-j} - l + bj\right) \end{cases}$$

le minimum se réalisant où les dérivées premières s'annulent (fonction convexe). Pour simplifier, on fait les remarques suivantes :

$$\sum_{j=0}^{\infty}(1-\alpha)^j = \frac{1}{\alpha}, \quad \sum_{j=0}^{\infty} j(1-\alpha)^j = \frac{1-\alpha}{\alpha^2}, \quad \sum_{j=0}^{\infty} j^2(1-\alpha)^j = \frac{(1-\alpha)(2-\alpha)}{\alpha^3}$$

On a :

$$\begin{cases} \displaystyle\sum_{j=0}^{\infty}(1-\alpha)^j X_{t-j} - \frac{l}{\alpha} + b\frac{(1-\alpha)}{\alpha^2} &= 0 \\ \displaystyle\sum_{j=0}^{\infty} j(1-\alpha)^j X_{t-j} - l\frac{1-\alpha}{\alpha^2} + b\frac{(1-\alpha)(2-\alpha)}{\alpha^3} &= 0 \end{cases}$$

soit

$$\begin{cases} \alpha \sum_{j=0}^{\infty} (1-\alpha)^j X_{t-j} - l + b\dfrac{(1-\alpha)}{\alpha} & = 0 \\ \alpha^2 \sum_{j=0}^{\infty} j(1-\alpha)^j X_{t-j} - (1-\alpha)l + b\dfrac{(1-\alpha)(2-\alpha)}{\alpha} & = 0 \end{cases}$$

De plus, pour obtenir les formules de mise à jour, on note :

$$\begin{cases} S_1(t) & = \alpha \sum_{j=0}^{\infty} (1-\alpha)^j X_{t-j} \\ S_2(t) & = \alpha \sum_{j=0}^{\infty} (1-\alpha)^j S_1(t-j) \end{cases}$$

On remarque que $S_2(t) - \alpha S_1(t) = \alpha^2 \sum_{j=0}^{\infty} j(1-\alpha)^j X_{t-j}$ car :

$$\begin{aligned} S_2(t) - \alpha S_1(t) &= \alpha \sum_{j=1}^{\infty} (1-\alpha)^j S_1(t-j) \\ &= \alpha^2 \sum_{j=1}^{\infty} \sum_{h=0}^{\infty} (1-\alpha)^{j+h} X_{t-j-h} \\ &= \alpha^2 \sum_{j=1}^{\infty} \sum_{k=j}^{\infty} (1-\alpha)^k X_{t-k} \\ &= \alpha^2 \sum_{j=1}^{\infty} j(1-\alpha)^j X_{t-j} \end{aligned}$$

Alors

$$\begin{cases} S_1(t) - l + b\dfrac{(1-\alpha)}{\alpha} & = 0 \\ S_2(t) - \alpha S_1(t) - (1-\alpha)l + b\dfrac{(1-\alpha)(2-\alpha)}{\alpha} & = 0 \end{cases} \implies \begin{cases} S_1(t) & = l - b\dfrac{(1-\alpha)}{\alpha} \\ S_2(t) & = l - 2b\dfrac{(1-\alpha)}{\alpha} \end{cases}$$

Ainsi :

$$\begin{cases} l & = 2S_1(t) - S_1(t) \\ b & = \dfrac{\alpha}{1-\alpha}\left(S_1(t) - S_2(t)\right) \end{cases} \tag{6.19}$$

Définition 6.2 *Prévision par lissage exponentielle double*

La prévision de la série à l'horizon h, notée $\widehat{X}_T(h)$, fournie par la méthode de lissage exponentielle double est donnée par :

$$\widehat{X}_T(h) = \hat{l}_T + \hat{b}_T(h) \tag{6.20}$$

α est la **constante de lissage** et le couple $(\hat{l}_T, \hat{b}_T)$ est donné par :

$$\begin{cases} \hat{b}_T &= \dfrac{\alpha}{1-\alpha}\left(S_1(T) - S_2(T)\right) \\ \hat{l}_T &= 2S_1(T) - S_2(T) \end{cases} \tag{6.21}$$

Remarque 6.5

Les expressions de $\hat{l}$ et $\hat{b}$ dépendent de $S_1(T)$ et $S_2(T)$ qui sont respectivement le lissage exponentiel simple de la série initiale et le lissage exponentielle simple de la série lissée. On a donc effectué deux lissages consécutifs d'où le nom de lissage exponentiel double.

```
# Lissage exponentiel double (Brown)
def LED(x,alpha):
    T = len(x)
    X = np.array(x)
    S1=np.zeros(T)
    S2=np.zeros(T)
    S1[0] = float(x.mean())
    S2[0] = S1[0]
    l = np.zeros(T-1)
    b = np.zeros(T-1)
    Xhat = np.zeros(T-1)
    for t in range(1,T):
        S1[t] = alpha*X[t-1]+(1-alpha)*S1[t-1]
        S2[t] = alpha*S1[t] + (1-alpha)*S2[t-1]
        l[t-1] = 2*S1[t] - S2[t]
        b[t-1] = (alpha/(1-alpha))*(S1[t]-S2[t])
        Xhat[t-1] = l[t-1]+b[t-1]
    Xhat = pd.DataFrame(Xhat,columns=["fittedvalues"],index=x.index[1:])
    return Xhat

# Représentation graphique
led1 = LED(x = nile, alpha=0.2)
led2 = LED(x = nile, alpha=0.8)
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(nile,color="black",label = "nile");
axe.plot(led1,color="blue",linestyle = "dashed",label=r"$\alpha=0.2$");
axe.plot(led2,color="red",linestyle = "dashdot",label=r"$\alpha=0.8$");
axe.set_xlabel("année");
axe.set_ylabel("débit du Nil (1e9 $m^{3}$)");
axe.legend();
plt.show()
```

La figure 6.13 illustre le comportement du lissage double en prenant la constante de lissage égale à 0.8 ou 0.2. La courbe noire représente la série initiale, la courbe en pointillé bleu la série lissée avec $\alpha = 0.2$ et la courbe en pointillé rouge la série lissée avec $\alpha = 0.8$.

6.1.2.2 Formule de mise à jour

On déduit les formules de récurrence suivantes :

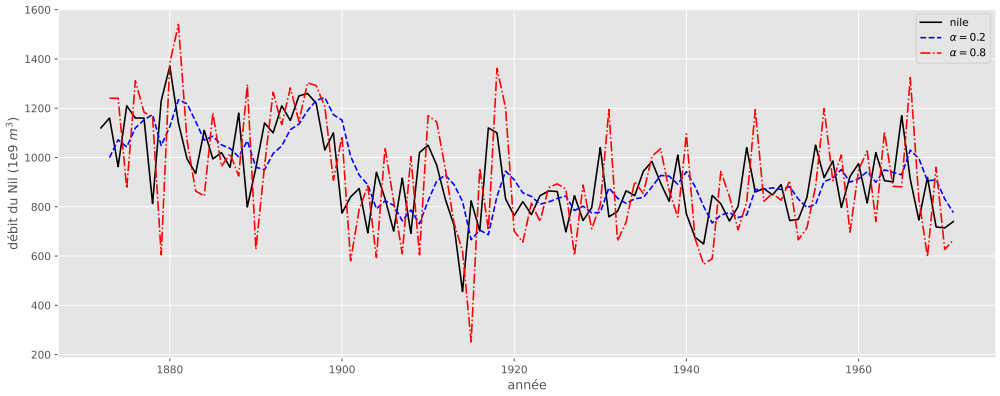


Figure 6.13 – Prévision par lissage exponentiel double de la série Nile

$$\begin{cases} S_1(t) &= \alpha X_t + (1-\alpha) S_1(t-1) \\ S_2(t) &= \alpha S_1(t) + (1-\alpha) S_2(t-1) \\ &= \alpha^2 + \alpha(1-\alpha) S_1(t-1) + (1-\alpha) S_2(t-1) \end{cases}$$

Ainsi

$$l_t = \left(1-(1-\alpha)^2\right) X_t + (1-\alpha)(2-\alpha) S_1(t-1) - (1-\alpha) S_2(t-1)$$

En incorporant $S_1(t) = l_t - \dfrac{1-\alpha}{\alpha} b_t$ et $S_2(t) = l_t - b_t \dfrac{2(1-\alpha)}{\alpha}$, on en déduit après simplification :

$$\begin{cases} l_t &= \left(1-(1-\alpha)^2\right) X_t + (1-\alpha)^2 (l_{t-1} + b_{t-1}) \\ b_t &= \alpha^2 X_t + \alpha(1-\alpha) S_1(t-1) - \alpha S_2(t-1) \\ &= \alpha^2 X_t - \alpha^2 l_{t-1} + \left(1-\alpha^2\right) b_{t-1} \end{cases}$$

en rappelant que $\widehat{X}_T(h) = l_T + b_T h$, on retrouve les formules de mise à jour attendues.

Pour utiliser ces formules de mise à jour, il faut avoir des valeurs initiales pour les suites $\hat{l}_t$ et $\hat{b}_t$. On prend en général :

$$\begin{cases} \hat{b}_2 &= X_2 - X_1 \\ \hat{l}_2 &= X_2 \end{cases}$$

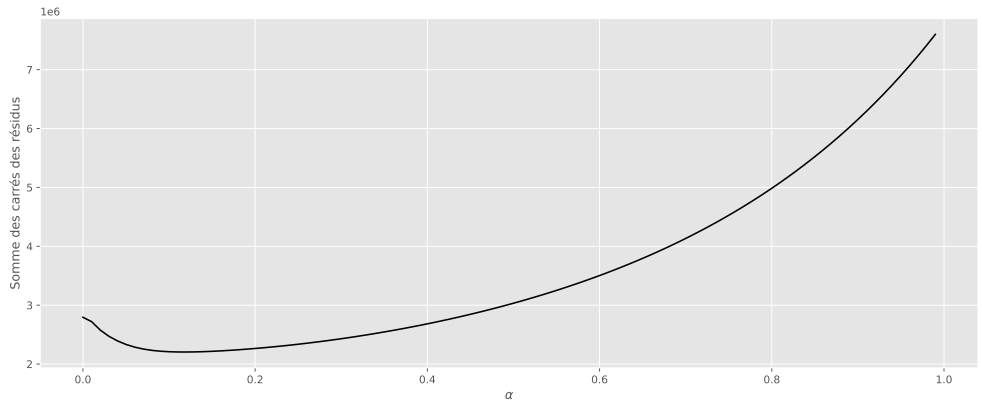
Notons enfin que pour sélectionner la constante de lissage, on peut utiliser un critère similaire à celui du lissage exponentiel simple.

Exemple 6.6 *Choix de α pour la prévision par lissage exponentiel double de la série Nile (1871 - 1970)*

```
# Somme des carrés des résidus
def scr_led(alpha,x):
    T = len(x)
    X = np.array(x)
    S1=np.zeros(T)
```

```
    S2=np.zeros(T)
    S1[0] = float(x.mean())
    S2[0] = S1[0]
    l = np.zeros(T-1)
    b = np.zeros(T-1)
    Xhat = np.zeros(T-1)
    err = np.zeros(T-1);
    for t in range(1,T):
        S1[t] = alpha*X[t-1]+(1-alpha)*S1[t-1]
        S2[t] = alpha*S1[t] + (1-alpha)*S2[t-1]
        l[t-1] = 2*S1[t] - S2[t]
        b[t-1] = (alpha/(1-alpha))*(S1[t]-S2[t])
        Xhat[t-1] = l[t-1]+b[t-1]
        err[t-1] = X[t] - Xhat[t-1]
    return np.square(err).sum()

# Affichage
scrled = np.array([scr_led(i,nile) for i in pas])
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(pas,scrled,color= "black");
axe.set_xlabel(r"$\alpha$");
axe.set_ylabel("Somme des carrés des résidus");
plt.show()
```



On détermine $\hat{\alpha}$:

```
# alpha optimal
const_led = LinearConstraint(1,lb=0,ub=1)
alpha_led = minimize(scr_led,x0=0.01,args=(nile),method="L-BFGS-B",
                     constraints=const_led).x
# LED avec alpha optimal
led3  = LED(x=nile,alpha= float(alpha_led))
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(nile,color="black",label = "nile");
axe.plot(led1,color="blue",linestyle = "dashed",label=r"$\alpha=0.2$");
axe.plot(led2,color="red",linestyle = "dashdot",label=r"$\alpha=0.8$");
axe.plot(led3,color="green",linestyle = "dashdot",
```

```
        label=r"$\widehat{\alpha}$=%s"%float(alpha_led));
axe.set_xlabel("année");
axe.set_ylabel("débit du Nil (1e9 $m^{3}$");
axe.legend();
plt.show()
```

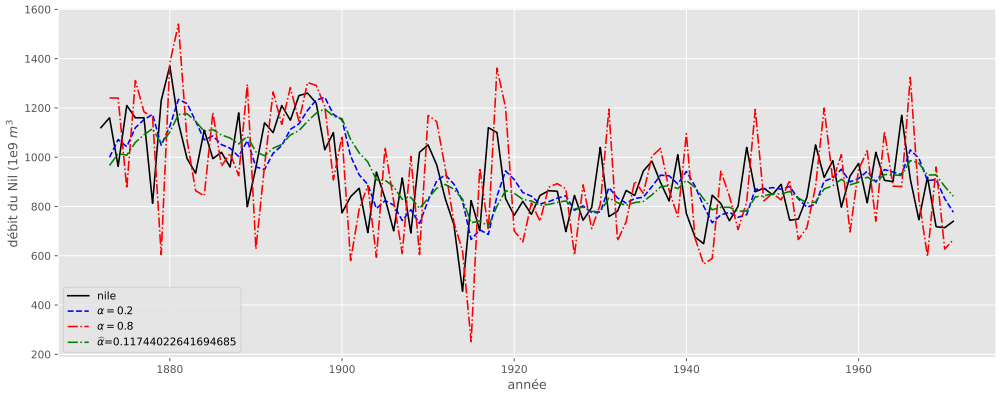


Figure 6.14 – Lissage exponentiel double de la série Nile avec valeur optimale de α.

6.1.2.3 Intervalle de confiance de la prévision

L'intervalle de confiance[2] de la prévision X_{T+h} est calculée à partir de la variance de l'erreur de prévision donnée par :

$$\text{Var}(e_{T+h}) = \sigma_X^2 C_h^2 \tag{6.22}$$

avec

$$C_h^2 = 1 + \frac{1-\alpha^{'}}{(1+\alpha^{'})^3}\left[(1+4\alpha^{'}+5\alpha^{'2}) + 2h(1-\alpha^{'})(1+3\alpha^{'}) + 2h^2(1-\alpha^{'})^2\right] \tag{6.23}$$

avec $\alpha^{'} = 1-\alpha$.

L'intervalle de confiance à $100(1-\beta)\%$ de la valeur prévue X_{T+h} s'écrit donc :

$$X_{T+h} \in \left[\widehat{X}_T(h) \pm z_{\beta/2}\sigma_X C_h\right] \tag{6.24}$$

avec $z_{\beta/2}$ la valeur de la loi normale centrée réduite. Pour un intervalle de confiance à 95%, nous avons :

$$X_{T+h} \in \left[\widehat{X}_T(h) \pm 1.96\sigma_X C_h\right] \tag{6.25}$$

Cet intervalle de confiance dépend de l'horizon de prévision h contrairement au lissage exponentiel simple.

Exemple 6.7 *Prévision d'une chronique à partir d'un lissage exponentiel double*

2. Abraham et Ledolter, 1983, p. 128 - 129.

Soit les données observées de la série X_t consignées sur le tableau 6.2. On demande de calculer une prévision par une méthode de lissage exponentiel double à un horizon de trois périodes. La constante de lissage α est égale 0.5.

On appliquera les formules suivantes :

$$\begin{cases} S_1(t) &= \alpha X_{t-1} + (1-\alpha) S_1(t-1) \\ S_2(t) &= \alpha S_1(t) + (1-\alpha) S_2(t-1) \end{cases}$$

avec $S_1(t)$ est la série lissée une fois et $S_2(t)$ la série lissée deux fois à l'instant t.

On initialise : $S_1(1) = X_1 = 10$.

$$\begin{aligned} S_1(2) &= 0.5X_1 + 0.5S_1(1) = 0.5 \times 10 + 0.5 \times 10 = 10 \\ S_1(3) &= 0.5X_2 + 0.5S_1(2) = 0.5 \times 20 + 0.5 \times 10 = 5 \\ &\dots \\ S_1(9) &= 0.5X_8 + 0.5S_1(8) = 0.5 \times 50 + 0.5 \times 42.97 = 47.49 \end{aligned}$$

Pour le second lissage, on a : $S_2(1) = S_1(1) = X_1 = 10$

Il s'en suit que :

$$\begin{aligned} S_2(2) &= 0.5S_1(2) + 0.5S_2(1) = 0.5 \times 10 + 0.5 \times 10 = 10 \\ S_2(3) &= 0.5S_1(3) + 0.5S_2(2) = 0.5 \times 15 + 0.5 \times 10 = 12.5 \\ &\dots \\ S_2(9) &= 0.5S_1(9) + 0.5S_2(8) = 0.5 \times 46.49 + 0.5 \times 36.88 = 41.69 \end{aligned}$$

Table 6.2 – Prévision à partir du modèle de lissage exponentiel exponentiel double

t	X_t	$S_1(t)$	$S_2(t)$	l_t	b_t	$\widehat{X}_t$	$\hat{\varepsilon}_t$
1	10	10	10				
2	20	10.00	10.00	10.00	0.00	10.00	10.00
3	20	15.00	12.50	17.50	2.50	20.00	0.00
4	30	17.50	15.00	20.00	2.50	22.50	7.50
5	40	23.75	19.38	28.12	4.37	32.49	7.51
6	40	31.88	25.63	38.13	6.25	44.38	- 4.38
7	50	35.94	30.79	41.09	5.15	46.24	3.76
8	50	42.97	36.88	49.06	6.09	55.15	- 5.15
h						$\widehat{X}_T(h)$	
1		46.49	41.69	51.29	4.80	56.09	
2						60.89	
3						65.69	
4						70.49	

La prévision à l'horizon h est définie comme suit :

$$\widehat{X}_T(h) = \hat{l}_T + h\hat{b}_T$$

avec

$$\begin{cases} \hat{l}_T &= 2S_1(T) - S_2(T) = 51.29 \\ \hat{b}_T &= \dfrac{\alpha}{1-\alpha}\left(S_1(T) - S_2(T)\right) = 4.80 \end{cases}$$

La variance de X vaut : $\sigma_X^2 = 193.75$. Ce implique que $\sigma_X = 13.92$.

la variance de l'erreur de prévision vaut :

$$\text{Var}(e_{T+h}) = \sigma_X^2 C_h^2$$

avec

$$C_h^2 = 1 + \frac{1-\alpha^{'}}{(1+\alpha^{'})^3}\left[(1+4\alpha^{'}+5\alpha^{'2}) + 2h(1-\alpha^{'})(1+3\alpha^{'}) + 2h^2(1-\alpha^{'})^2\right]$$

avec $\alpha^{'} = 1-\alpha = 0.5$.

Pour les quatre périodes de prévision, on a :

$C_1^2 = 2.07$, $C_2^2 = 2.67$, $C_3^2 = 3.41$, $C_4^2 = 4.30$

$C_1 = 1.44$, $C_2 = 1.63$, $C_3 = 1.85$, $C_4 = 2.07$

D'où les intervalles de confiance :

- $IC_1 = 56.09 \pm 1.96 \times 1.44 \times 13.92 = [16.81; 95.38]$;
- $IC_2 = 60.89 \pm 1.96 \times 1.63 \times 13.92 = [16.41; 10.5.37]$;
- $IC_3 = 65.69 \pm 1.96 \times 1.85 \times 13.92 = [15.23; 116.18]$;
- $IC_4 = 70.49 \pm 1.96 \times 2.07 \times 13.92 = [14.03; 126.98]$.

Remarque 6.6

Le lissage exponentiel simple et le lissage exponentiel double ne sont pas adaptés lorsque la série présente des variations saisonnières. Pour effectuer des prévisions sur une série affectée des variations saisonnières, on utilise la méthode de Holt-Winters.

6.2 Méthode de Holts - Winters

La méthode de lissage exponentiel double permet de traiter des séries présentatnt une tendance linéaire mais sans saisonnalité (*cf.* Brown (1959)). On peut également définir des lissages exponentiels généralisés sur le même principe que les techniques décrites dans les sections précédentes permettant de traiter des séries avec saisonnalité.

Une méthode un peu différente a été introduite par Holt (1957) et Winters (1960). Il existe une version non saisonnière de cette méthode, c'est - à - dire adaptée aux séries sans saisonnalité pouvant être ajustées par une droite au voisinage de T (comme pour le lissage exponentiel double). La différence entre la méthode de Holt - Winters et le lissage exponentiel double porte sur les formules de mise à jour.

6.2.1 Méthode non saisonnière : le modèle avec tendance de Holt

De la même façon que le lissage exponentiel double, l'ajustement avec la méthode non saisonnière de Holt (1957) se fait de façon linéaire au voisinage de T [3].

Définition 6.3 *Lissage exponentiel double de Holt*

Soit une série temporelle X_t. On appelle lissage exponentiel double de Holt de paramètres $\alpha \in [0;1]$ et $\beta \in [0;1]$ de cette série le processus $\widehat{X}_T$ définie comme suit :

3. Pour plus d'informations, voir https ://otexts.com/fpp2/holt.html

$$\widehat{X}_T(h) = l_T + hb_T \tag{6.26}$$

avec

$$\begin{cases} l_t &= \alpha X_t + (1-\alpha)\left(l_{t-1} + b_{t-1}\right) \\ b_t &= \beta\left(l_t - l_{t-1}\right) + (1-\beta) b_{t-1} \end{cases} \tag{6.27}$$

où $0 \leq \alpha \leq 1$ et $0 \leq \beta \leq 1$. l_t est une estimation du niveau de la série, b_t une estimation de sa pente (localement en temps).

Remarque 6.7

l_t est le barycentre affecté des poids α et $1-\alpha$ de la dernière valeur de X observé et sa prévision à l'horizon 1. L'algorithme « corrige » donc la prévision de la constante en prenant en compte le dernier écart observé. De même, b_t est au barycentre de la dernière pente prévue et l'écart entre les 2 dernières ordonnées à l'origine prévues.

Deux lissages distincts sont effectués :

- Le lissage de la moyenne l avec un coefficient de lissage α, $\alpha \in [0;1]$;
- Le lissage de la tendance b avec un coefficient de lissage β, $\beta \in [0;1]$

L'avantage de cette approche est d'avoir une plus grande flexibilité mais la contrepartie est de devoir régler deux paramètres. Si α et β sont proches de 1 tous les deux, la prévision est lisse (fort poids du passé).

```
# Lissage exponentiel double de Holt (non saisonnier)
def Holt_led(x,alpha,beta):
    T = len(x)
    X = np.array(x)
    l =np.zeros(T)
    b=np.zeros(T)
    l[0] =  X[0]
    b[0] = 0
    for t in range(1,T):
        l[t] = alpha*X[t]+(1-alpha)*(l[t-1]-b[t-1])
        b[t] = beta*(l[t-1]-b[t-1]) + (1-beta)*b[t-1]
    l = pd.DataFrame(l,columns=["fittedvalues"],index=x.index)
    return l
```

Sur la figure suivante (*cf.* Figure 6.15), on visualise l'évolution du lissage en fonction de β (α étant ici fixé).

```
# Application
holt_led1 = Holt_led(x=nile,alpha=0.2,beta=0.2)
holt_led2 = Holt_led(x=nile,alpha=0.2,beta=0.8)
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(nile,color="black",label = "nile");
axe.plot(holt_led1,color="blue",linestyle = "dashed",
        label=r"$\alpha=0.2$ et $\beta=0.2$");
axe.plot(holt_led2,color="red",linestyle = "dashed",
```

```
        label=r"$\alpha=0.2$ et $\beta=0.8$");
axe.set_xlabel("année");
axe.set_ylabel("débit du Nil (1e9 $m^{3}$)");
axe.legend();
plt.show()
```

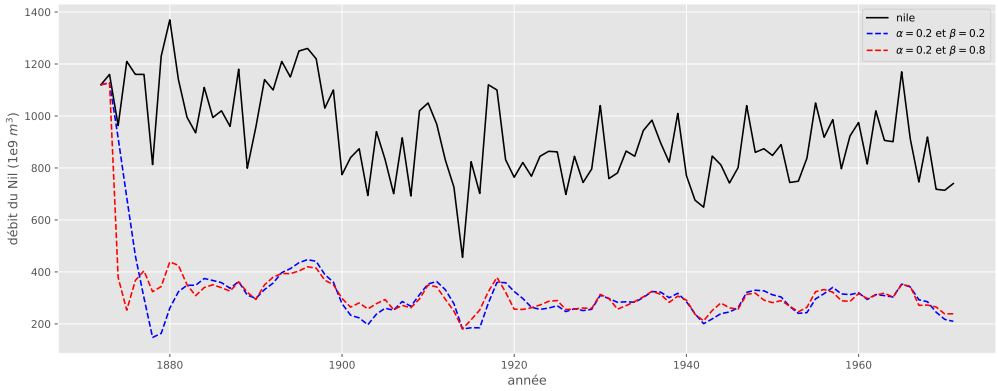


Figure 6.15 – Lissage exponentiel double de Holt sur la série Nile

La fonction Holt de Statsmodels permet de réaliser un lissage exponentiel double au sens de Holt.

Remarque 6.8

Le formule de mise à jour du lissage exponentiel double au sens de Brown (1959) est un cas particulier du lissage exponentiel double au sens de Holt (1957). En effet, on peut écrire :

$$\begin{cases} l_t &= (1-(1-\alpha)^2)\, X_t + (1-\alpha)^2 (l_{t-1} + b_{t-1}) \\ b_t &= \dfrac{\alpha^2}{1-(1-\alpha)^2}(l_t - l_{t-1}) + \left(1 - \dfrac{\alpha^2}{1-(1-\alpha)^2}\right) b_{t-1} \end{cases} \tag{6.28}$$

Exemple 6.8 *Prévision d'une chronique à partir du lissage exponentiel double de Holt*

Soit les données observées de la série ausair contenue dans le package fpp2 de R. On demande de calculer une prévision par une méthode de lissage exponentiel double de Holt à un horizon de cinq périodes.

```
# Chargement des données
air = sm.datasets.get_rdataset("ausair", "fpp2").data.set_index("time")
airdata = air[air.index>=1990]
airdata.index = pd.date_range(start='31/12/1990',periods=len(airdata),freq="Y")
airdata = airdata.rename(columns={"value":"ausair"})
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(airdata,color="black");
axe.set_ylabel("Passagers (millions)");
axe.set_xlabel("année");
plt.show()
```

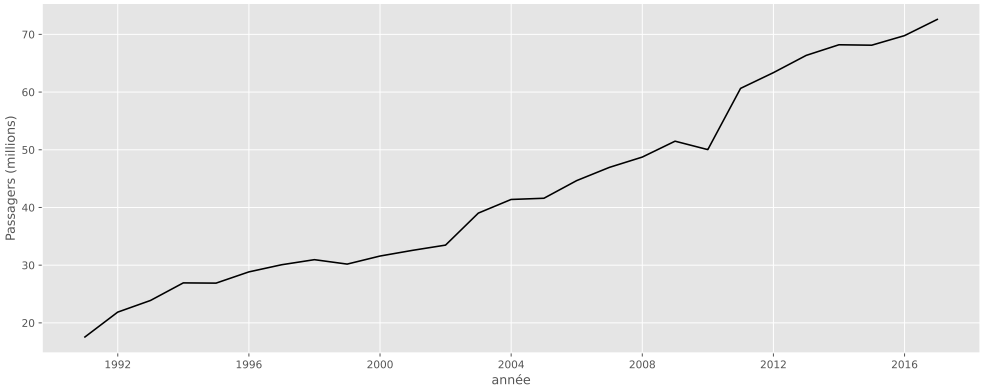


Figure 6.16 – Nombre de passagers d'Autralie (en millions) de 1990 à 2016.

les données de la figure 6.16 affichent un comportement de tendance à la hause.

```
# Prévision par lissage exponentiel double (Holt)
from statsmodels.tsa.api import Holt
air_led = Holt(airdata,initialization_method="estimated").fit()
# Paramère de lissage alpha et beta
result_led=pd.DataFrame(index=["alpha","beta","l.0","b.0"])
params_led=["smoothing_level","smoothing_trend","initial_level","initial_trend"]
result_led["coefficients"] = [air_led.params[k] for k in params_led]
```

Table 6.3 – Coefficients du lissage exponentiel double de Holt

	alpha	beta	l.0	b.0
coefficients	0.8210297	0	15.85114	2.097916

On effectue la prévision par lissage exponentiel double pour les 5 prochaines années.

```
# Prévision par LED
air_fcast = air_led.forecast(5)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(airdata,color='black');
axe.plot(air_led.fittedvalues,color='green',label = "Holt's method");
axe.plot(air_fcast,color="green");
axe.set_xlabel('année');
axe.set_ylabel('Passagers (en millions)');
axe.legend();
plt.show()
```

6.2.2 Méthode saisonnière additive

En présence d'une saisonnalité de type additive, on fait l'hypothèse que X_t peut être approchée au voisinage de T par :

$$\widehat{X}_T(h) = l_t + hb_t + s_t \tag{6.29}$$

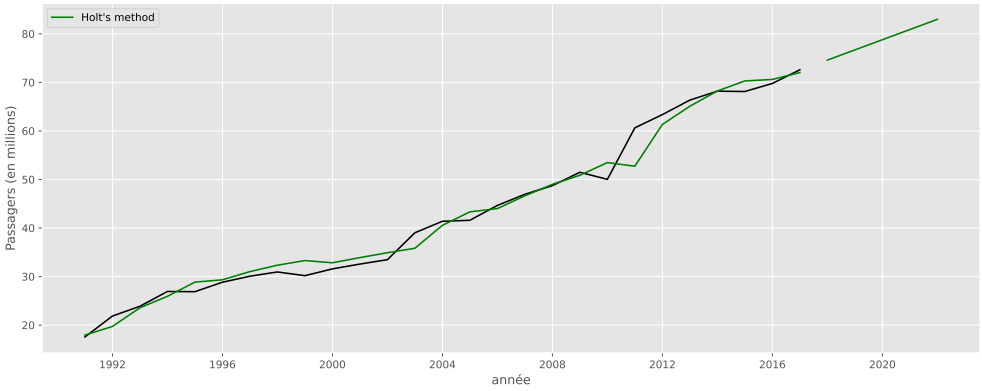


Figure 6.17 – Prévision par lissage exponentiel double (Holt)

où s_T représente la coefficient saisonnier en T.

Définition 6.4 *Lissage exponentiel double de Holt - Winters saisonnier additif*

Soit une série temporelle X_t. On appelle lissage exponentiel double de Holt - Winters saisonnier additif de cette série, le processus $\widehat{X}_T$ définie ainsi :

$$\widehat{X}_T(h) = l_T + hb_T + s_T \tag{6.30}$$

Trois lissages distincts sont effectués :

— La lissage de la moyenne l avec un coefficient de lissage α, avec $\alpha \in [0,1]$;
— La lissage de la tedndance b avec un coefficient de lissage β, avec $\beta \in [0,1]$;
— La lissage de la saisonnalité s avec un coefficient de lissage γ, avec $\gamma \in [0,1]$;

La méthode de Holt - Winters saisonnier additif propose pour l'estimation de l et b et s les formules de mise à jour suivantes :

$$\begin{cases} b_T &= \beta(l_T - l_{T-1}) + (1-\beta)b_{T-1} \\ l_T &= \alpha(X_T - s_{T-p}) + (1-\alpha)(l_{T-1} + b_{T-1}) \\ s_T &= \gamma(X_T - l_T) + (1-\gamma)s_{T-p} \end{cases} \tag{6.31}$$

où α, β et γ sont des constantes de lissage appartenant à $[0,1]$ et p la périodicité des données ($p = 12$ en mensuel, $p = 4$ en trimestriel).

— La première formule de mise à jour s'interprète comme une moyenne pondérée de la différence des niveaux estimés aux instants T et $T-1$ et la pente estimée à l'instant $T-1$;
— La deuxième comme une moyenne pondérée de l'observation X_T (à laquelle on a retranché la composante saisonnière estimée à l'étape précédente) et l'estimation de la tendance faite à l'instant $T-1$;
— Enfin, la troisième comme une moyenne pondérée de l'observation X_T (à laquelle on a retranché l eniveau calculé à l'instant T) et de la composante saisonnière calculée à l'instant $T-p$.

La prévision à l'horizon h est donnée par :

$$\widehat{X}_T(h) = \begin{cases} \hat{l}_T + h\hat{b}_T + \hat{s}_{T+h-p} & \text{si} \quad 1 \leq h \leq p \\ \hat{l}_T + h\hat{b}_T + \hat{s}_{T+h-2p} & \text{si} \quad p+1 \leq h \leq 2p \\ \vdots \end{cases} \tag{6.32}$$

6.2.3 Méthode saisonnière multiplicative

On considère une chronique dont on a observé les T premiers instants $X_1, \ldots, X_T$ et on suppose que cette série peut être approchée, au voisinage de T, par le modèle :

$$(l_t + hb_t)s_t \tag{6.33}$$

avec s_t représente le coefficient saisonnier en t.

Définition 6.5 *Lissage expoentiel double de Holt - Winters saisonnier multiplicatif*

Soit une série temporelle X_t. On appelle lissag exponentiel double de Holt - Winters saisonnier multiplicatif de cette série, le processus $\widehat{X}_T$ défini, pour tout horizon h, par :

$$\widehat{X}_T(h) = (l_T + hb_T)s_T \tag{6.34}$$

La méthode de Holt - Winters saisonnier multiplicatif propose pour l'estimation de l, b et s les formules de mise à jour suivantes :

$$\begin{cases} l_T &= \alpha X_T/s_{T-p} + (1-\alpha)(l_{T-1} + b_{T-1}) \\ b_T &= \beta(l_T - \hat{l}_{T-1}) + (1-\beta)b_{T-1} \\ s_T &= \gamma X_T/l_T + (1-\gamma)s_{T-p} \end{cases} \tag{6.35}$$

où α, β et γ sont des constantes de lissage appartenant à $[0;1]$. Ces formules de mises à jour s'interprètent comme dans le cas de la méthode saisonnière additive.

La prévision à un horizon de h périodes est définie par :

$$\widehat{X}_T(h) = \begin{cases} \left(\hat{l}_T + h\hat{b}_T\right)\hat{s}_{T+h-p} & \text{si} \quad 1 \leq h \leq p \\ \left(\hat{l}_T + h\hat{b}_T\right)\hat{s}_{T+h-2p} & \text{si} \quad p+1 \leq h \leq 2p \\ \vdots \end{cases} \tag{6.36}$$

Proposition 6.1 *Initialisation du modèle de Holt-Winters*

Comme pour les autres méthodes de lissage, dans la pratique, il y a un problème de démarrage de la technique : les valeurs de départ peuvent être estimées par la méthode des moindres carrés ordinaires ou plus simplement initialisées pour la première année ($t = 1, \ldots, T$) de la manière suivante :

— Initialisation de la saisonnalité

Les coefficients saisonniers pour la première année sont estimés par la valeur observée en $t(X_t)$ divisée par la moyenne $\overline{X}$ des T premières observations de la première année.

$$S_t = X_t/\overline{X} \quad \text{pour} \quad t = 1, \ldots, T$$

— Initialisation de la moyenne lissée : $l_p = \overline{X}$
— Initialisation de la tendance : $b_p = 0$

Remarque 6.9

— Dans certaines écritures du modèle, les coefficientss saisonniers vérifient la propriété : $\sum_{j=1}^{j=p} s_j = 0$ (dans le cas additif) ou $\sum_{j=1}^{j=p} s_j = p$ (dans le cas multiplicatif) selon le principe de la conservation des aires.
— Les paramètres α, β et γ sont optimisés comme pour les méthodes non saisonnières en minimisant la somme des carrés des erreurs prévisionnelles entre la valeur observée de la chronique et les valeurs prévues.

La fonction ExponentialSmoothing de Statsmodels permet d'estimer les données par la méthode de Holt -Winters (saisonnalité additive et multiplicative).

Exemple 6.9 *Prévision par la méthode de Holt - Winters des visiteurs internationaux de nuit en Australie*

Soit les données observées de la série austourists contenue dans le package fpp2 de R. On demande de calculer une prévision par une méthode de lissage exponentiel double de Holt - Winters (saisonnier additif et multiplicatif) à un horizon de 8 périodes.

```
# Chargement des données
aust = sm.datasets.get_rdataset("austourists", "fpp2").data.set_index("time")
austdata = aust[aust.index>=2005].rename(columns={"value":"austourists"})
austdata.index = pd.date_range(start='2005-03-31',periods=len(austdata),freq='Q')
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(austdata,color="black");
axe.set_ylabel("visiteurs de nuit (millions)");
axe.set_xlabel("année");
plt.show()
```

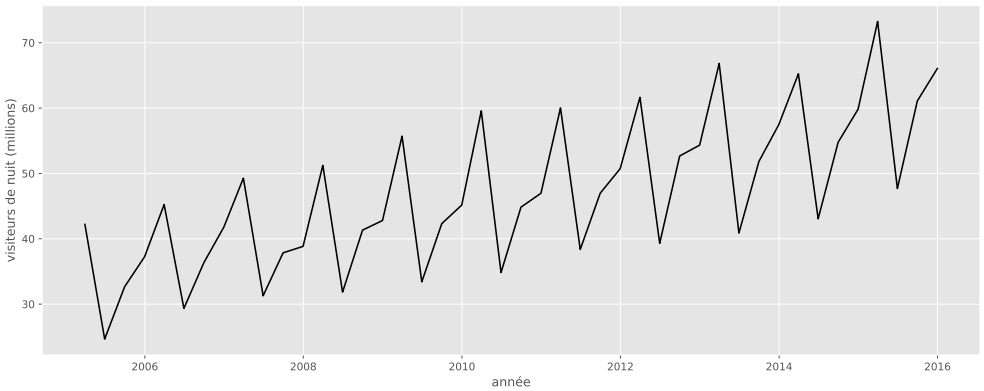


Figure 6.18 – Visiteurs internationnaux de nuit en Autralie (2005 - 2015).

```
# Estimation par Holt-Winters
from statsmodels.tsa.api import ExponentialSmoothing
aust_hw_add = ExponentialSmoothing(austdata,seasonal_periods=4,trend="add",
                          seasonal="add",initialization_method="estimated").fit()
aust_hw_mul = ExponentialSmoothing(austdata,seasonal_periods=4,trend="add",
                          seasonal="mul",initialization_method="estimated").fit()
# Représentation graphique
axe = austdata.plot(figsize=(16,6),color="black");
axe.set_ylabel("Visiteurs internationaux de nuit (en millions)");
axe.set_xlabel("année");
aust_hw_add.fittedvalues.plot(ax=axe,linestyle="dashed",color="red");
aust_hw_mul.fittedvalues.plot(ax=axe,linestyle="dashdot",color="green");
aust_hw_add.forecast(8).rename("HW (tendance et saisonnalité additive)").plot(
    ax=axe,linestyle="dashed",color="red",legend=True);
aust_hw_mul.forecast(8).rename("HW (tendance additive et saisonnalité mul)").plot(
    ax=axe,linestyle="dashdot",color="green",legend=True);
plt.show()
```

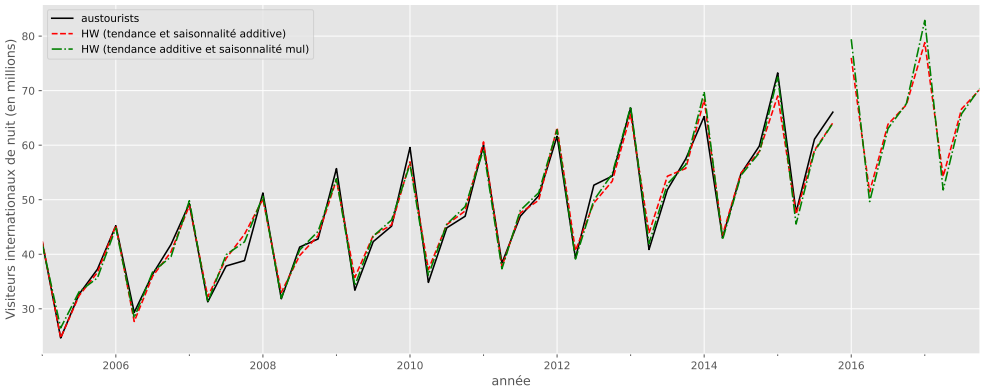


Figure 6.19 – Prévision des visiteurs internationaux de nuit en Australie par la méthode de Holt - Winters (additive et multiplicative

Nous comparons les deux approches utilisées à l'aide de la racine carrée de l'erreur quadratique moyenne (RMSE, *Root Mean Squared Error*). Sa formule est donnée par :

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{t=T}\left(X_t - \hat{X}_t\right)^2} = \sqrt{\frac{SCR}{T}} \tag{6.37}$$

où SCR = Somme des carrés résiduels (SSE, *Sum of Squared Errors*).

```
# Coefficients de lissage
results_hw = pd.DataFrame(index=["alpha","beta","gamma","l.0","b.0","SSE"])
params_hw = ["smoothing_level","smoothing_trend","smoothing_seasonal",
             "initial_level","initial_trend"]
results_hw["Additive"] = [aust_hw_add.params[p] for p in params_hw] + \
                         [aust_hw_add.sse]
results_hw["Multiplicative"] = [aust_hw_mul.params[p] for p in params_hw] + \
                           [aust_hw_mul.sse]
results_hw.loc["RMSE",:] = np.sqrt(results_hw.loc["SSE",:]/len(austdata))
```

Table 6.4 – Coefficients du lissage Holt - Winters

	alpha	beta	gamma	l.0	b.0	SSE	RMSE
Additive	0.262	0	0.455	32.809	0.701	135.921	1.758
Multiplicative	0.364	0	0.000	31.572	0.666	105.175	1.546

La faible valeur de $\hat{\gamma}$ pour le modèle multiplicatif signifie que la composante saisonnière ne change pratiquement pas dans le temps. La faible valeur de $\hat{\beta}$ pour les deux modèles signifie que la composante de pente ne change pratiquement pas dans le temps. La taille croissante de la composante saisonnière du modèle additif suggère que le modèle est moins approprié que le modèle multiplicatif (ce qui est confirmé par la valeur du RMSE).

Nous visualisons les différentes composantes des deux modèles : l_t, b_t et s_t

```
# l = slope, b = trend et s = seasonal
states1 = pd.DataFrame(
          np.c_[aust_hw_add.level, aust_hw_add.trend,aust_hw_add.season],
          columns=["level", "slope", "seasonal"],index=austdata.index)
states2 = pd.DataFrame(
          np.c_[aust_hw_mul.level,aust_hw_mul.trend,aust_hw_mul.season],
          columns=["level", "slope", "seasonal"],index=austdata.index)
fig, [[ax1, ax4], [ax2, ax5], [ax3, ax6]] = plt.subplots(3,2,figsize=(14, 8))
states1[["level"]].plot(ax=ax1,c="black");
states1[["slope"]].plot(ax=ax2,c="black");
states1[["seasonal"]].plot(ax=ax3,c="black");
states2[["level"]].plot(ax=ax4,c="black");
states2[["slope"]].plot(ax=ax5,c="black");
states2[["seasonal"]].plot(ax=ax6,c="black");
plt.tight_layout();
plt.show()
```

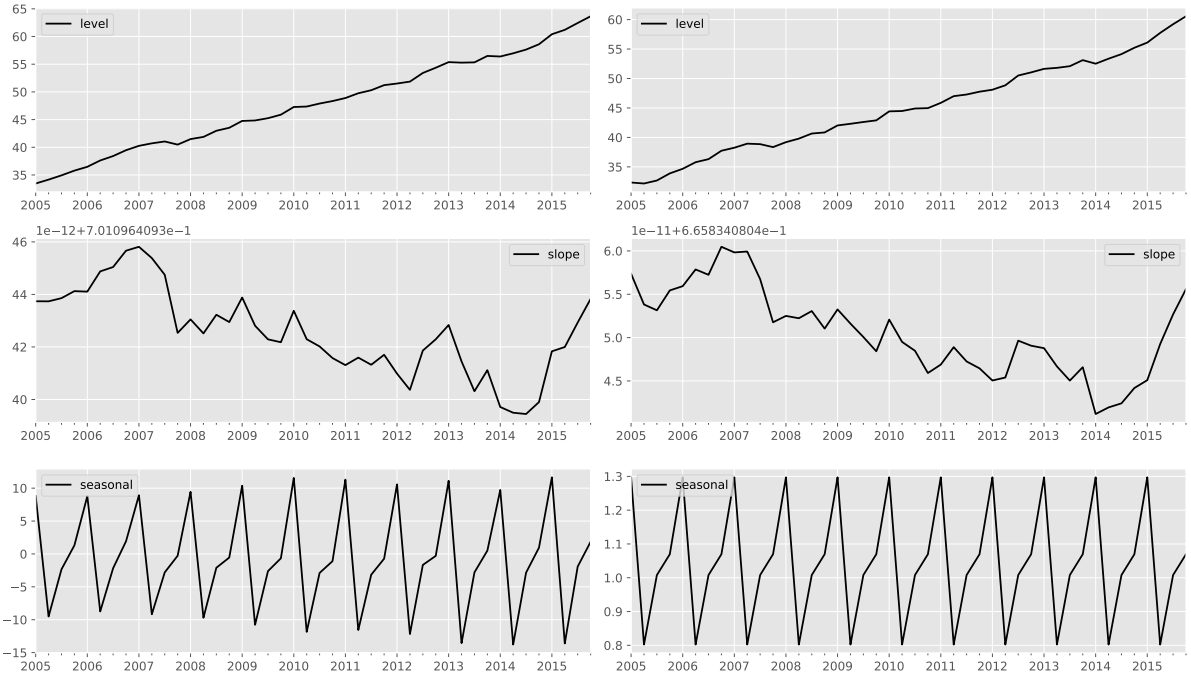


Figure 6.20 – Composantes estimées par la méthode Holt - Winters (additive et multiplicative)

A gauche, nous avons l'évolution des composantes en supposant que la saisonnalité est additive et à droite, l'évolution des composantes en supposant que la saisonnalité est multiplicative.

6.2.4 Intervalle de prédiction

Un grand avantage des modèles de lissage exponentiel est que des intervalles de prédiction peuvent également être générés. Ces intervalles de prédiction diffèreront entre les modèles selon que les méthodes sont additives ou multiplicatives. Pour la plupart de ces modèles, un intervalle de confiance peut être écrit comme suit :

$$\widehat{X}_T(h) \pm c\sigma_h \tag{6.38}$$

où c représente le quantile d'une loi de probabilité et σ_h^2 représente la variance de l'erreur de prédiction. Si on suppose que la loi probabilité est la loi normale, alors, c prendra la valeur $z_{\alpha/2}$ à $100(1-\alpha)\%$ (*cf.* tableau 6.5).

Table 6.5 – Valeurs de c pour la loi normale

	50%	55%	60%	65%	70%	75%	80%	85%	90%	95%	96%	97%	98%	99%
quantile	0.67	0.76	0.84	0.93	1.04	1.15	1.28	1.44	1.64	1.96	2.05	2.17	2.33	2.58

Pour les modèles de lissage exponentiel, la formule de σ_h^2 est compliquée (*cf.* R. Hyndman et al. (2008)). Le tableau 6.6 donne les formules pour quelques modèles :

Table 6.6 – Variance de l'erreur de prédiction des modèles de lissage exponentiel

Lissage	Variance de l'erreur de prédiction
Simple	$\sigma_h^2 = \sigma_\varepsilon^2 \left[1+\alpha^2(h-1)\right]$
Holt	$\sigma_h^2 = \sigma_\varepsilon^2 \left[1+(h-1)\left\{\alpha^2+\alpha\beta+\frac{1}{6}\beta^2 h(2h-1)\right\}\right]$
H-W (Additive)	$\sigma_h^2 = \sigma_\varepsilon^2 \left[1+(h-1)\left\{\alpha^2+\alpha\beta+\frac{1}{6}\beta^2 h(2h-1)\right\}+\gamma k\left\{2\alpha+\gamma+\beta p(k+1)\right\}\right]$

avec p la périodicité des données et k la partie entière de $(h-1)/p$.

7 Modèles avancés de décomposition

Sommaire

En matière de désaisonnalisation des séries temporelles, les filtres de lissage linéaire ou moyennes mobiles sont déjà connus au début des années 20, mais sont rarement utilisés. En effet, le meilleur instrument connu pour éliminer la composante saisonnière, la moyenne mobile centrée d'ordre douze, s'avère un piètre estimateur de la tendance - cycle (*cf.* Darné (2004)). Par ailleurs, les moyennes mobiles sont très sensibles aux cas érratiques (instables) et nécessitent de ce fait un traitement a priori des valeurs extrêmes. C'est ce genre de limite qui conduit W. W. I. King (1924) à proposer de nouvelles méthodes pour estimer les facteurs saisonniers. Vers la fin des années 20, l'élaboration de nouveaux filtres de lissage et de techniques d'application différentes permet de diffuser cette approche non paramétrique pour la désaisonnalisation des séries chronologiques.

Le développement de l'informatique, après la séconde guerre mondiale, contribue à la propagation et à l'amélioration des méthodes non paramétriques de désaisonnalisation. La méthode la plus connue est sûrement la méthode X11 développée par Shiskin (1967). Par la suite, de nombreuses méthodes dont le principe de construction est identique à celui de X11 ont été développées (*cf.* Sutcliffe (1999)).

7.1 La méthode X11 et les modèles hybrides

7.1.1 La méthode X11

La méthode X11 est basée sur l'utilisation de différentes sortes de moyennes mobiles (symétriques et asymétriques) pour décomposer une série temporelle en ses éléments de tendance, saisonnalité et résidus, sans modèle explicite sous - jacent. Chaque composante s'obtient après plusieurs itérations.

Les estimations des composantes saisonnières et de tendance résultent d'un filtrage en cascade, c'est - à - dire de l'application successive de divers filtres linéaires individuels, appliqués de manière séquentielle. Pour les données mensuelles, ce filtrage est composé de : (i) une moyenne mobile centrée à 12 termes, (ii) deux filtres mobiles saisonniers d'ordre $3 \times (2m + 1)$, et (iii) et une moyenne mobile de Henderson.

La moyenne mobile centrée à 12 termes est définie par :

$$\Theta(B) = \frac{1}{24} B^{-6} (1 + B) (1 + B + B^2 + \cdots + B^{11}) \tag{7.1}$$

où B est l'opérateur retard, et les moyennes mobiles saisonnières par

$$S_t^{3\times(2m+1)} = \frac{1}{3} \left(S_{t-12}^{(2m+1)} + S_t^{(2m+1)} + S_{t+12}^{(2m+1)} \right) \tag{7.2}$$

avec $S_t^{(2m+1)} = \dfrac{1}{2m+1} \displaystyle\sum_{j=-m}^{j=m} SI_{t+12j}$

où $S_t^{3\times(2m+1)}$ correspond à la moyenne mobile saisonnière d'ordre $3 \times (2m + 1)$ avec $m = 1, 2, 4$, et SI est la série corrigée de la tendance.

L'estimation de la tendance s'obtient par application d'un des trois filtres linéaires de Henderson, c'est - à - dire les filtres à 9, 13 et 23 termes. Ces filtres, développés par Henderson (1916), sont la solution d'une équation de récurrence linéaire. Par exemple, le filtre de tendance de Henderson à 13 termes est défini par :

$$\begin{aligned} H_{13}(B) = {} & -0.019B^{-6} - 0.028B^{-5} + 0.00B^{-4} + 0.065B^{-3} + 0.147B^{-2} + 0.214B^{-1} + 0.24B^{0} \\ & + 0.214B^{1} + 0.147B^{2} + 0.065B^{3} + 0.00B^{4} - 0.028B^{5} - 0.019B^{6} \end{aligned}$$

Pour les observations en début et en fin de série où la moyenne mobile symétrique de Henderson ne peut pas être utilisée, La méthode X11 emploie des moyennes mobiles asymétriques dérivés de la méthode de Musgrave (1964) afin de prolonger la série. Elles déterminent la longueur du filtre de tendance de Henderson à appliquer. La méthode X11 corrige les effets des variations de jours ouvrables, mais pas ceux de la fête de Pâques.

L'inconvénient majeur de la méthode X11 est une estimation peu fiable des valeurs désaisonnalisées en fin de série. Ces dernières sont importantes pour évaluer la direction de la tendance et identifier un point de retournement cyclique dans l'économie.

7.1.2 Les modèles hybrides

La vulgarisation des modèles ARIMA à partir de l'ouvrage de G. Box and Jenkings (1970) a permis de faire progresser les outils de désaisonnalisation. Elle donne lieu à l'évolution de la méthode X-11 vers X11-ARIMA, en améliorant l'estimation des observations récentes. En effet, la série initiale est modélisée par un processus ARIMA puis prolongée en début et en fin de série, limitant ainsi les révisions des estimations lorsque l'on dispose d'une observation supplémentaire. Ces modèles utilisent des filtres linéaires de la méthode X-11 combinés avec des filtres provenant de modèles ARIMA qui sont ajustés globalement aux données. Neanmoins, il faut noter que les modèles ARIMA ne peuvent pas être appliqués directement sur les séries chronologiques sans une analyse préalable de celles - ci, notamment à cause du problème de stationnarité et des effets déterministes.

7.1.2.1 La méthode X11-ARIMA

La méthode X11-ARIMA est une amélioration du X11 proposé par Dagum (*cf.*v@ dagum1975seasonal, Dagum (1978) et DAGUM (1979)) pour laquelle une première version est nommée X11-ARIMA/80. Néanmoins, celui-ci présente l'inconvénient de ne pas corriger

les effets de calendrier présents dans les séries de flux avant de procéder à leur modélisation ARIMA. Les modèles X11-ARIMA/88 et X11-ARIMA/2000 développés par Dagum (1988) résolvent ce problème à partir d'un traitement séquentiel et automatique de ces effets.

La principale amélioration de la méthode X11-ARIMA est la possibilité d'ajuster un modèle ARIMA à la série, qui permet à la méthode de prévoir jusqu'à trois années de données supplémentaires (*cf.* Dagum (1988)). Ceci mène à une meilleure désaisonnalisation pour les observations récentes. Le modèle X11-ARIMA/88 a permis de corriger les limites des estimations peu fiables en fin de série produites par la procédure X11 en utilisant les modèles ARIMA.

7.1.2.2 La méthode X12-ARIMA

La méthode X12-ARIMA est principalement fondée sur celle de X11-ARIMA (versions 1980 et 1988) et a été dévelopée par Findley et al. (1998) au US Bureau of the Census. Elle utilise la même méthode que X11, et intègre également la plupart des améliorations introduites par X11-ARIMA et d'autres nouvelles. La différence majeure est un pré - traitement supplémentaire des données, appelé *RegARIMA* (*Regression ARIMA*). Cette procédure permet une estimation simultanée des points aberrants, des variations de jours ouvrables et des effets de Pâques, avec un modèle ARIMA saisonnier. Ce dernier est sélectionné à partir d'une procédure automatique, basée sur cinq modèles, similaire à celle employée par X11-ARIMA, et est utilisé pour extrapoler la série initiale afin de la prolonger.

Actuellement, ces méthodes ne sont pas encore implémentées sous Python. Cependant, pour les plus curieux, ils peuvent consulter la fonction x13_arima_analysis de Statsmodels. Sous R, la fonction seas de la librairie seasonal dispose de cette méthode X11.

Exemple 7.1 *Ajustement par la méthode X11 de la série elecequip*

```
# Décomposition par la méthode X-11
library(fpp2)
library(seasonal)
library(ggplot2)
elecequip %>% seas(x11="") -> fit
autoplot(fit)
```

Pour plus d'informations sur la méthode X11 et les méthodes hybrides, veuillez consulter Dagum and Bianconcini (2016).

7.2 Modèle STL

La méthode STL (*Seasonal - Trend decomposition using Loess*) est une méthode de décomposition d'une série chronologique qui est souvent employée dans les analyses économiques et environnementales. Elle a été proposée par R. B. Cleveland et al. (1990). STL utilise des modèles de régression ajustés localement (localement linéaire ou localement quadratique) pour décomposer une série chronologique en composants saisonniers, de tendance et restants. L'algorithme STL effectue un lissage sur la série chronologique en utilisant LOESS en deux boucles : la boucle interne réalise une itération entre le lissage saisonnier et de tendance et la boucle externe réduit l'effet des points aberrants. Au cours de la boucle interne, le composant saisonnier est calculé d'abord, puis supprimé pour calculer le composant de tendance. Le reste est calculé en soustrayant les composants saisonniers et de tendance de la série chronologique.

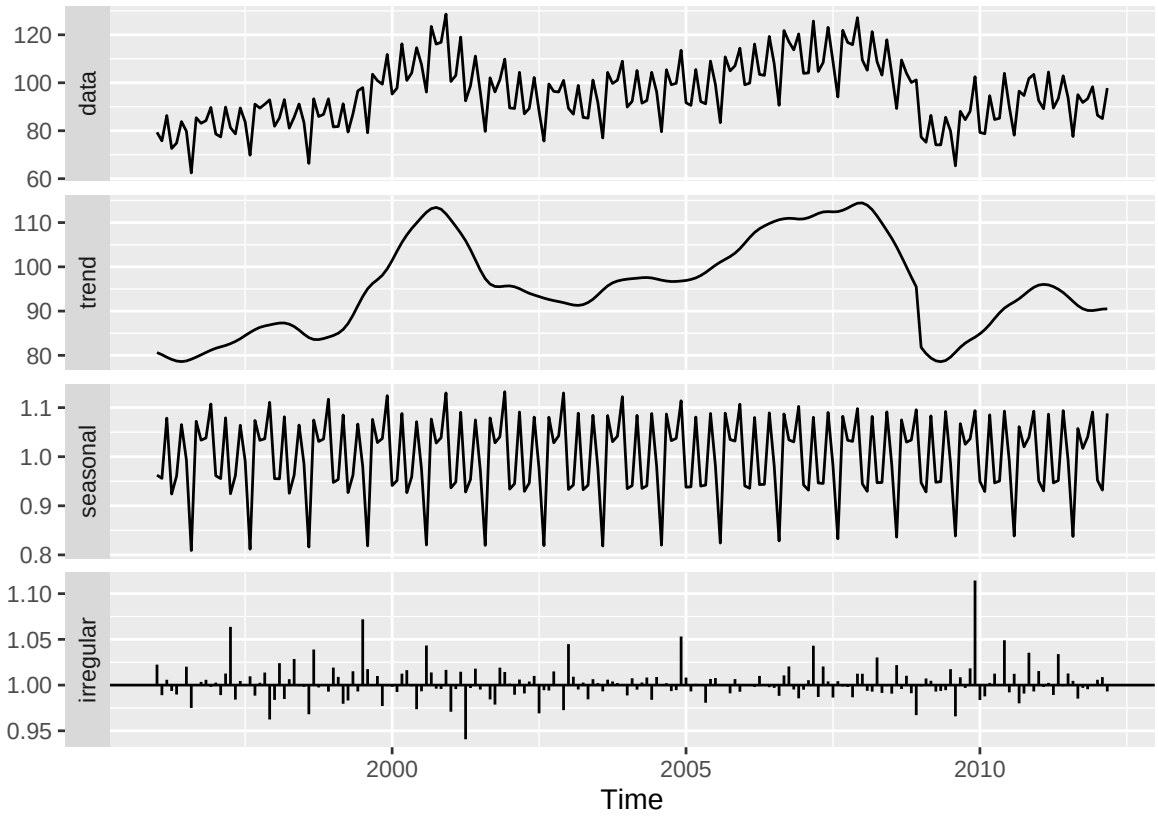


Figure 7.1 – Décomposition de la série elecequip par la méthode X11

7.2.1 Régression loess

La régression Loess (*locally estimated scatterplot smoothing*) ou lowess (*Locally Weighted Linear Regression*) est une méthode de régression non paramétrique fortement connexe qui combine plusieurs modèles de régression multiple au sein d'un méta - modèle qui repose sur la méthode des k plus proches voisins. Elle est une alternative possible aux méthodes habituelles de régression et combine la simplicité de la régression linéaire par les moindres carrés avec la flexibilité de la régression non linéaire, en effectuant une régression simple sur des sous - ensembles locaux de données. Elle permet de produire des courbes lissées, ajustées à un nuage de point. Le principe de la régression loess a été initialement décrit par W. S. Cleveland (1979), puis développé et enrichi par W. S. Cleveland (1981) et W. S. Cleveland and Devlin (1988).

7.2.1.1 Principe de la méthode

Dans une régression loess, l'ajustement de la courbe se fait localement. Pour déterminer la valeur X_t que prend la courbe au point d'abscisse t, on ajuste un polynôme de degré 1 (localement linéaire) ou 2 (localement quadratique) aux points au voisinage de t, les degrés supérieurs ayant tendance à sur - ajuster les données de chaque sous-ensemble, et le degré 0 revenant à calculer des moyennes mobiles pondérées. Concrètement, il s'agit d'ajuster localement le modèle

$$X_t = f_p(t) + \varepsilon_t, \quad t = 1, \dots, T \tag{7.3}$$

où f_p est un polynôme de degré p, $f_p(t) = \sum_{j=0}^{j=p} a_j t^j$ avec $p = 1$ ou $p = 2$. Cet ajustement se fait

avec pondération : les points les plus proches de t ont davantage de poids dans l'ajustement. Si on pose $\theta = (a_0, a_1, ..., a_p)$, alors on cherche θ qui minimise la quantité :

$$\sum_{t=1}^{t=T} w_t \left(X_t - f_p^{\theta}(t)\right)^2 \tag{7.4}$$

où w_t est appelé coefficient ou facteur de pondération. On montre aisément qu'on a encore

$$\hat{\theta} = \underset{\theta \in \mathbb{R}^{p+1}}{\arg\min} ||W(X - Z\theta)||^2 \tag{7.5}$$

avec Z une matrice de format $(T \times (p+1))$; W est une matrice diagonale de dimension $(T \times T)$.

$$Z = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & T & \cdots & T^p \end{pmatrix}, \quad W = \begin{pmatrix} w_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & w_T \end{pmatrix}, \quad X = \begin{pmatrix} X_1 \\ \vdots \\ X_T \end{pmatrix} \tag{7.6}$$

L'estimateur $\hat{\theta}$ de θ est alors donné par :

$$\hat{\theta} = (Z^{'}WZ)^{-1}Z^{'}WX \tag{7.7}$$

Preuve

Afin d'estimer le vecteur θ composé des coefficients $a_0, a_1, ..., a_p$, nous appliquons la méthode des moindres carrés ordinaires (MCO) qui consiste à minimiser la quantité suivante :

$$\begin{aligned} \mathcal{S}(\theta) &= (WX - WZ\theta)^{'}(WX - WZ\theta) \\ &= (WX)^{'}(WX) - (WX)^{'}(WZ\theta) - (WZ\theta)^{'}(WX) + (WZ\theta)^{'}(WZ\theta) \\ &= X^{'}W^{'}WX - X^{'}W^{'}WZ\theta - \theta^{'}Z^{'}W^{'}WX + \theta^{'}Z^{'}W^{'}WZ\theta \end{aligned}$$

W est une matrice diagonale idempotent, c'est - à - dire $W^{'}W = W^2 = W$. De plus, $X^{'}WZ\theta = \theta^{'}Z^{'}WX$. Ce qui permet de réécrire l'équation précédente comme suit :

$$\mathcal{S}(\theta) = X^{'}WX - 2\theta^{'}Z^{'}WX - \theta^{'}Z^{'}WZ\theta \tag{7.8}$$

Pour minimiser cette fonction par rapport à θ, nous différencions $\mathcal{S}$ par rapport à θ :

$$\frac{\partial}{\partial\theta}\mathcal{S}(\theta) = -2Z^{'}WX + 2Z^{'}WZ\hat{\theta} = 0 \quad \Longrightarrow \quad \hat{\theta} = (Z^{'}WZ)^{-1}Z^{'}WX$$

Cette solution est réalisable si la matrice $(Z^{'}WZ)$ est inversible. On appelle equations normales les équations issues de la relation :

$$(Z^{'}WZ)\hat{\theta} = Z^{'}WX \tag{7.9}$$

Soit, sous forme matricielle :

$$\begin{pmatrix} \sum_{t=1}^{t=T} w_t & \sum_{t=1}^{t=T} t w_t & \cdots & \sum_{t=1}^{t=T} t^p w_t \\ \sum_{t=1}^{t=T} t w_t & \sum_{t=1}^{t=T} t^2 w_t & \cdots & \sum_{t=1}^{t=T} t^{p+1} w_t \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{t=1}^{t=T} t^p w_t & \sum_{t=1}^{t=T} t^{p+1} w_t & \cdots & \sum_{t=1}^{t=T} t^{2p} w_t \end{pmatrix} \begin{pmatrix} \hat{a}_0 \\ \hat{a}_1 \\ \vdots \\ \hat{a}_p \end{pmatrix} = \begin{pmatrix} \sum_{t=1}^{t=T} w_t X_t \\ \sum_{t=1}^{t=T} t w_t X_t \\ \vdots \\ \sum_{t=1}^{t=T} t^p w_t X_t \end{pmatrix}$$

7.2.1.2 Voisinage

La taille du voisinage est déterminée par un paramètre α : quand $\alpha < 1$, le voisinage inclut une proportion α de points du jeu de données. Quand $\alpha \geq 1$, tous les points sont utilisés. L'ajustement ne dépend alors plus de α à travers la détermination du voisinage. En revanche α joue toujours sur le calcul des poids.

7.2.1.3 Pondération

La fonction de pondération généralement utilisée pour effectuer une régression locale est une fonction cubique pondérée. Le poids w d'un point d'abscisse $t^{'}$ au voisinage de t est d'autant plus important que sa distance $d(t, t^{'})$ à t est petite :

$$w_t = \left(1 - \left| \frac{d(t, t^{'})}{\max\limits_i d(i, t^{'})} \right|^3 \right)^3 1_{|t| \leq 1} \tag{7.10}$$

Cependant, il est possible d'utiliser toute autre fonction de pondération qui satisfait les propriétés énumérées par W. S. Cleveland (1979). Quand $\alpha < 1$, $\max\limits_i d(i, t^{'})$ est la distance maximale (en t) séparant deux points du voisinage considéré. Quand $\alpha \geq 1$, $\max\limits_i d(i, t^{'})$ est égale à α fois la distance maximale observée (en t) entre deux points du jeu de données.

7.2.1.4 Ajustement

Une fois le voisinage et les poids déterminés, une courbe correspondant à un polynôme de degré 1 ou 2 est ajustée pour une série de valeurs t. La valeur prédite en ce point t par la régression loess correspond à la valeur prédite par le polynôme ajusté localement.

Sous Python, pour ajuster une régression loess, il suffit de faire appel à la fonction localreg du package localreg.

Exemple 7.2 *Ajustement de la série Air passengers par régression locale*

On souhaite ajuster la série Air passengers par la méthode de régression locale.

```
# Chargement des données
import seaborn as sns
donnee = sns.load_dataset("flights").drop(columns=["year", "month"])
```

```
# Application
xval = np.arange(1,len(donnee.passengers)+1)
yval = donnee.passengers.values
# Estimation par régression locale
from localreg import *
lo = localreg(xval,yval,kernel=rbf.tricube,frac=0.2)
# Représentation  graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(xval, yval,color = 'black',label = "Air passengers");
axe.plot(xval, lo,color='red',label = "loess");
axe.set_xlabel("Date");
axe.set_ylabel("nombre de passagers");
axe.legend();
plt.show()
```

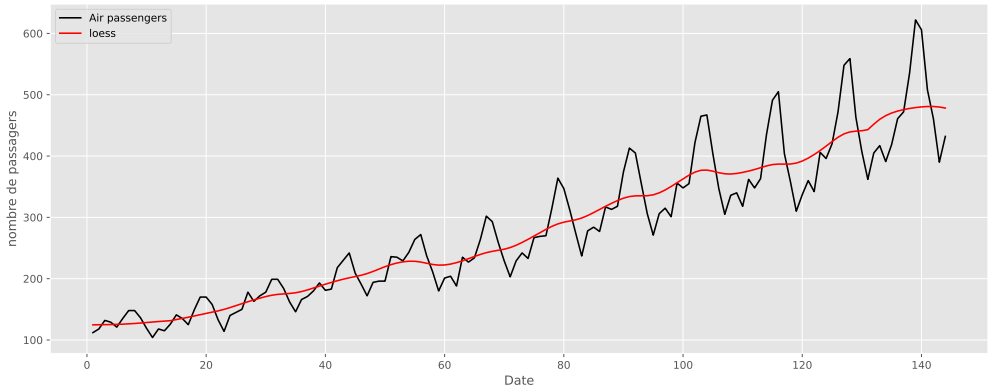


Figure 7.2 – Régression locale

En modifiant le paramètre `frac` de la fonction loessreg, on peut produire des courbes plus ou moins lissées :

```
# Régression loess avec différents points
loess1 = localreg(xval,yval,frac=0.3,kernel=rbf.tricube)
loess2 = localreg(xval,yval,frac=0.75,kernel=rbf.tricube)
loess3 = localreg(xval,yval,frac=2,kernel=rbf.tricube)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(xval, yval,color = "black",label="Air passengers");
axe.plot(xval, loess1,color='red',label = "loess with frac = 0.3");
axe.plot(xval, loess2,color='blue',label = "loess with frac = 0.75");
axe.plot(xval, loess3,color='yellow',label = "loess with frac = 2");
axe.set_xlabel("Date");
axe.set_ylabel("nombre de passagers");
axe.legend();
plt.show()
```

On peut egalement spécifier le degré du polynôme à ajuster localement grâce au paramètre `degree`.

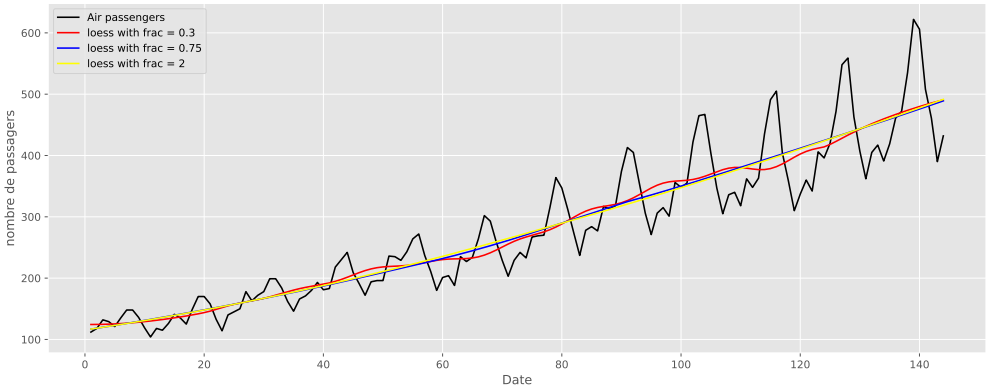


Figure 7.3 – Régression locale avec différentes valeurs du voisinage

```
# Ajustement avec polynôme de degré 1 et de degré 2
loess_degree1 = localreg(xval,yval,degree=1,frac=0.3,kernel=rbf.tricube)
loess_degree2 = localreg(xval,yval,degree=2,frac=0.3,kernel=rbf.tricube)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(xval,yval,color = 'black',label = "original data");
axe.plot(xval,loess_degree1,color='red',label = "loess with degree 1");
axe.plot(xval,loess_degree2,color='blue',label = "loess with degree 2");
axe.set_xlabel("Date");
axe.set_ylabel("nombre de passagers");
axe.legend();
plt.show()
```

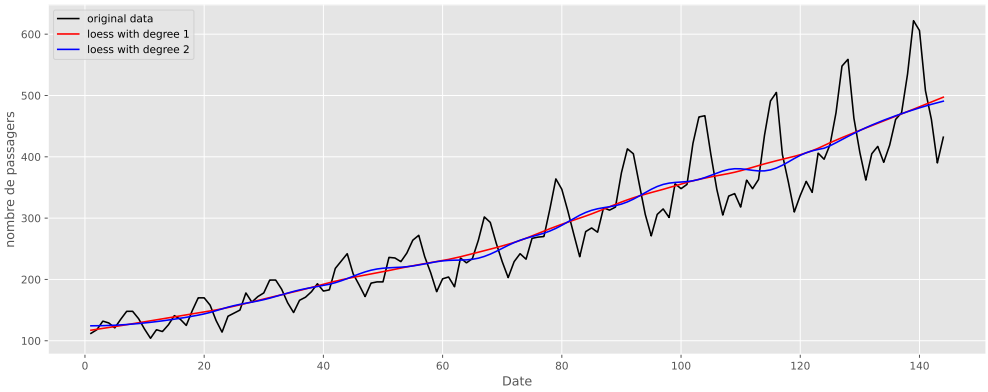


Figure 7.4 – Régression locale avec polynôme de degré 1 (en rouge) et de degré 2 (en bleu)

Statsmodels possède également une fonction lowess pour la régression locale. L'inconvénient de cette fonction est qu'elle ne permet pas de spécifier le degré du polynôme à ajuster localement.

```
# Régression locale avec Statsmodels
import statsmodels.api as sm
smoothed = sm.nonparametric.lowess(exog=xval, endog=yval, frac=0.3)
# Représentation graphique
```

```
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(xval, yval, color="black",label="Air passengers");
axe.plot(smoothed[:,0],smoothed[:, 1],color="red",label="lowess");
axe.set_xlabel("Date");
axe.set_ylabel("nombre de passagers");
axe.legend();
plt.show()
```

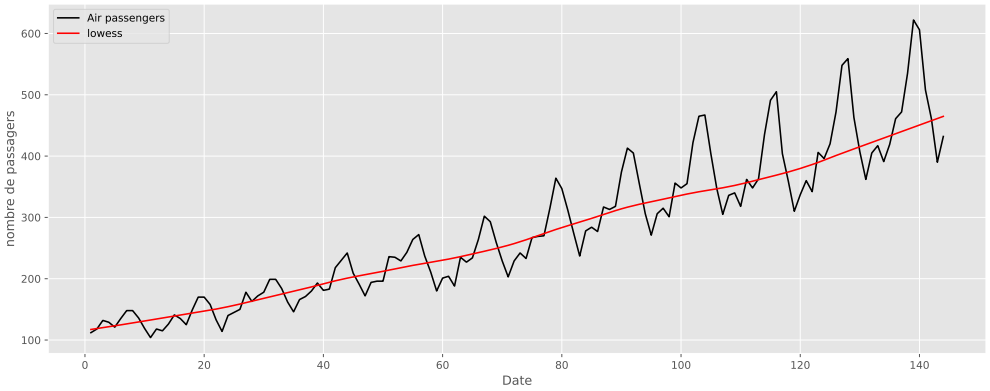


Figure 7.5 – Régression locale avec Statsmodels

Rappelons que dans le cadre la régression loess, si l'ordre p du polynôme vaut 0, le calcul se ramène à une moyenne mobile classique.

7.2.2 Décomposition par loess

La décomposition par loess se base sur le principe de régression par loess afin de pouvoir calculer les différents composants de la série temporelle : tendance, saisonnalité et résidus. Ainsi, avec un choix judicieux de k (paramètre des plus proches voisins), et éventuellement en réitérant le processus, on va pouvoir extraire la tendance générale.

Comme avec la méthode des moyennes mobiles, on va aussi pouvoir désaisonnalier une série en prenant une fenêtre égale à un an, donc, en considérant 12 termes pour les données mensuelles (13 en fait pour une fenêtre centrée sur les observations de la série). Une combinaison de ces deux traitements va permettre de décomposer une série en tendance, saisonnalité et résidus selon un modèle additif.

Comme les résidus ont rarement une distribution normale et ne sont souvent pas indépendants entre eux dans les séries, les hypothèses de base indispensables pour la régression à l'aide de la méthode des moindres carrés sont violées la plupart du temps. Ainsi, pour obtenir une estimation plus exacte des composantes, il est possible d'utiliser une méthode de régression plus robuste (i.e., moins sensible à la violation de ces hypothèses de base). On peut par exemple décider de pondérer les valeurs de manière inversement proportionnelle à leur influence respective sur la régression. Ainsi, une valeur qui, à elle seule, aurait tendance à tirer la courbe vers elle (valeur isolée, très éloignée du nuage formé par toutes les autres) recevra une pondération très faible afin de minimiser son effet.

Exemple 7.3

Le jeu de données qui se prête merveilleusement bien à la décomposition loess est celui du co2 disponible dans R. B. Cleveland et al. (1990). Ces données mensuelles (janvier 1959 à décembre 1987) présentent une tendance et une saisonnalité claires dans l'ensemble de l'échantillon.

```
# Chargement des données
co2 = pd.Series(co2,index=pd.date_range("1-1-1959",periods=len(co2),freq="M"),
                name="CO2")
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(co2,color="black");
axe.set_xlabel("Date");
axe.set_ylabel("co2");
plt.show()
```

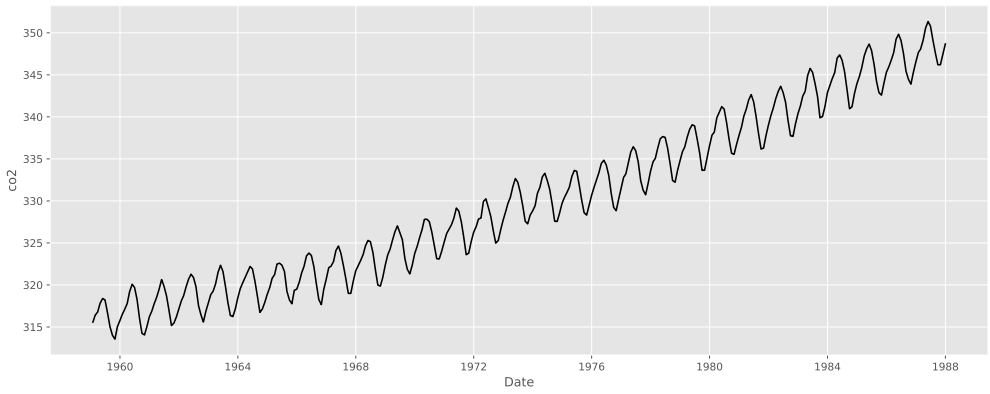


Figure 7.6 – Evolution du co2

```
# Statistiques descriptives
stats = co2.describe(include="all")
```

Table 7.1 – Statistiques descriptives sur co2

count	mean	std	min	25%	50%	75%	max
348	330.1239	10.05975	313.55	321.3025	328.82	338.0025	351.34

Sous Python, la fonction STL de Statsmodels permet d'effectuer une décomposition loess d'une série temporelle.

```
# Décomposition par loess
from statsmodels.tsa.seasonal import STL
stl = STL(co2, seasonal=13)
res = stl.fit()
plt.rc("figure", figsize=(16,8));
fig = res.plot();
plt.show()
```

L'argument `seasonal` prend par défaut la valeur 7. Il s'agit de l'étendue de la fenêtre de décomposition saisonnière (attention : l'etendue est bien $2m + 1$, où m est l'argument d'ordre

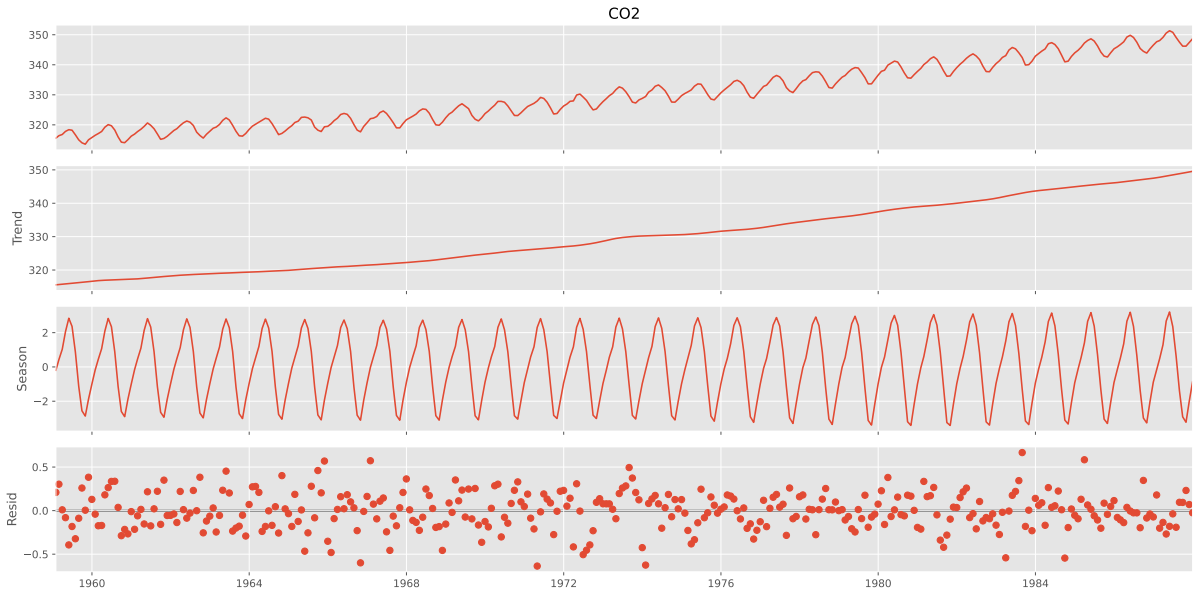


Figure 7.7 – Décomposition par loess sur la série co2

pour les moyennes mobiles). Nous choisissons la valeur impaire immédiatelemnt supérieure à la fréquence des observations pour une unité d'une an (ou d'un cycle). Donc, pour les données mensuelles, `seasonal=13`.

7.2.2.1 Ajustement robuste

En présence des valeurs extrêmes suspectes, nous pouvons utiliser la régression robuste à la place d'une régression par les moindres carrés classiques. Il suffit pour cela de préciser `robust = True`. Le réglage robuste utilise une fonction de pondération dépendante des données qui repondère les données lors de l'estimation du LOESS (et utilise donc LOWESS). L'utilisation d'une estimation robuste permet au modèle de tolérer des erreurs plus importantes qui sont visibles sur le tracé inférieur.

Exemple 7.4 *Ajustement robuste avec la série elec_equip*

Nous utilisons une série de mesures de la production d'équipements électriques dans l'UE.

```
# Chargement des données
from statsmodels.datasets import elec_equip as ds
elec_equip = ds.load().data.iloc[:, 0]
```

Ensuite, nous estimons le modèle avec et sans pondération robuste.

```
# Décomposition par loess robuste
def add_stl_plot(fig, res, legend):
    """Add 3 plots from a second STL fit"""
    axs = fig.get_axes()
    comps = ["trend", "seasonal", "resid"]
    for ax, comp in zip(axs[1:], comps):
        series = getattr(res, comp)
        if comp == "resid":
```

```
            ax.plot(series, marker="o", linestyle="none")
        else:
            ax.plot(series)
            if comp == "trend":
                ax.legend(legend, frameon=False)

stl = STL(elec_equip, period=12, robust=True)
res_robust = stl.fit()
fig = res_robust.plot()
res_non_robust = STL(elec_equip, period=12, robust=False).fit()
add_stl_plot(fig, res_non_robust, ["Robust", "Non-robust"])
plt.show()
```

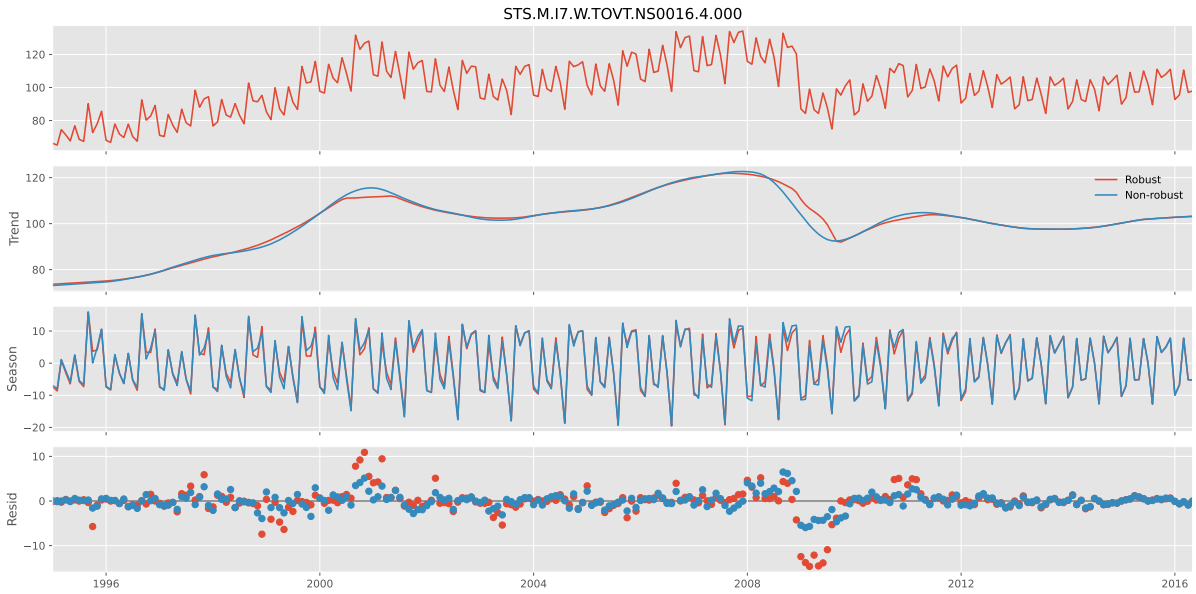


Figure 7.8 – Décomposition par loess robuste

La différence est mineure et est plus prononcée pendant la crise financière de 2008. L'estimation non robuste attribue des pondérations égales à toutes les observations et produit donc des erreurs plus faibles, en moyenne. Les poids varient entre 0 et 1.

```
# Coefficient de pondération
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(res_robust.weights, color="black", marker="o", linestyle="none");
axe.set_xlim(elec_equip.index[0], elec_equip.index[-1]);
plt.show()
```

7.2.2.2 Degré de la régression locale

La configuration par défaut estime le modèle LOESS avec une constante et une tendance. Cela peut être modifié pour n'inclure qu'une constante en définissant `COMPONENT_deg` sur 0.

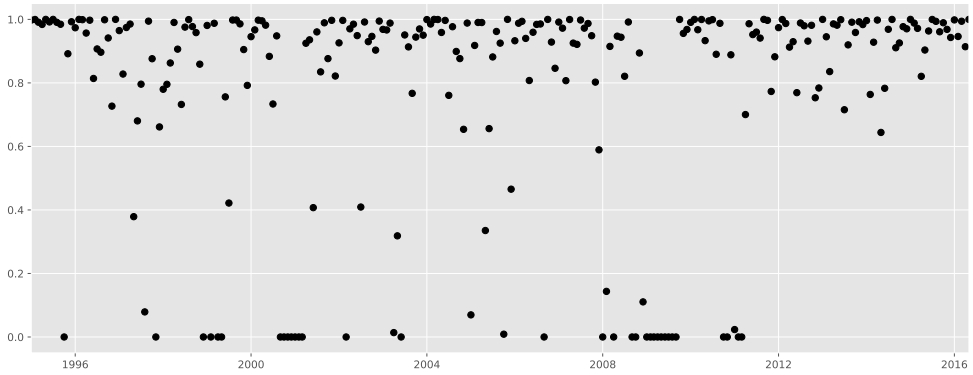


Figure 7.9 – Coefficient de pondération dans la décomposition robuste

```
# Degré de la régression locale
stl = STL(elec_equip, period=12, seasonal_deg=0, trend_deg=0, low_pass_deg=0,
          robust=True)
res_deg_0 = stl.fit()
fig = res_robust.plot()
add_stl_plot(fig, res_deg_0, ["Degree 1", "Degree 0"]);
fig.set_size_inches(16,7, forward=True);
fig.tight_layout()
plt.show()
```

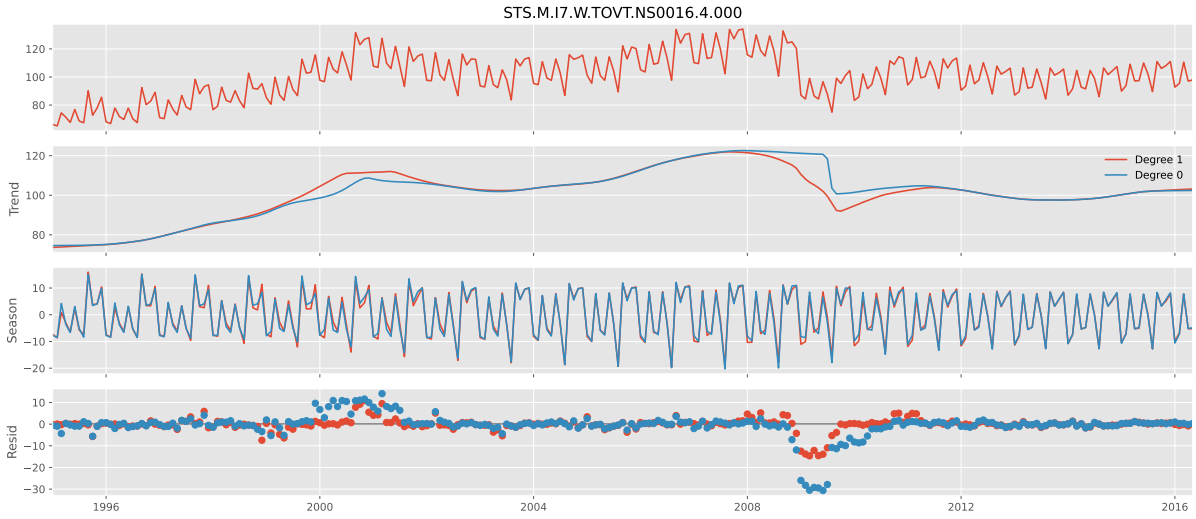


Figure 7.10 – Décomposition par loess avec différents degrés du polynôme

Ici, le degré fait peu de différence, sauf dans la tendance autour de la crise financière de 2008.

7.2.2.3 Performance

Trois options peuvent être utilisées pour réduire le coût de calcul de la décomposition STL : `seasonal_jumb`, `trend_jumb` et `low_pass_jump`. Lorsque ceux-ci sont différents de zéro, le LOESS pour le composant `COMPONENT` est estimé uniquement pour les observations `COMPONENT_jump`, et une interpolation linéaire est utilisée entre les points. Ces valeurs ne

doivent normalement pas dépasser 10 à 20% de la taille de `season`, `trend` ou `low_pass`, respectivement.

Exemple 7.5

L'exemple ci - dessous montre comment ceux-ci peuvent réduire le coût de calcul par un facteur de 15 en utilisant des données simulées avec à la fois une tendance cosinusoïdale à basse fréquence et un modèle saisonnier sinusoïdal.

```
# Simulation des données
rs = np.random.RandomState(0xA4FD94BC)
tau = 2000
t = np.arange(tau)
period = int(0.05 * tau)
seasonal = period + ((period % 2) == 0)
e = 0.25 * rs.standard_normal(tau)
y = np.cos(t / tau * 2 * np.pi) + 0.25 * np.sin(t / period * 2 * np.pi) + e
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(y,color="black");
axe.set_xlim(0,tau);
axe.set_xlabel("time");
axe.set_ylabel("valeur");
plt.show()
```

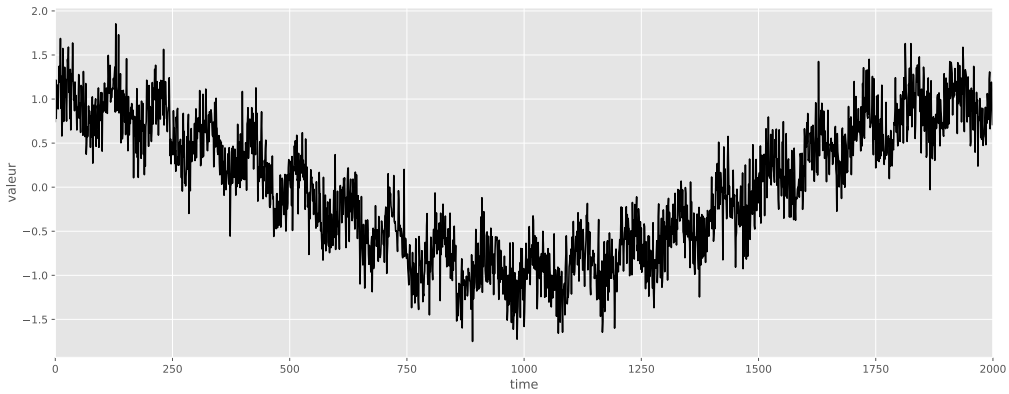


Figure 7.11 – Données simulées avec une tendance cosinusoïdale et une saisonnalité sinusoïdale

Tout d'abord, le modèle de base est estimé avec tous les sauts égaux à 1.

```
# Modèle à sauts fixés
mod = STL(y, period=period, seasonal=seasonal)
res = mod.fit()
fig = res.plot(observed=False,resid=False)
fig.set_size_inches((16,6))
fig.tight_layout()
plt.show()
```

Les sauts sont tous réglés à 15% de leur longueur de fenêtre. Une interpolation linéaire limitée fait peu de différence dans l'ajustement du modèle.

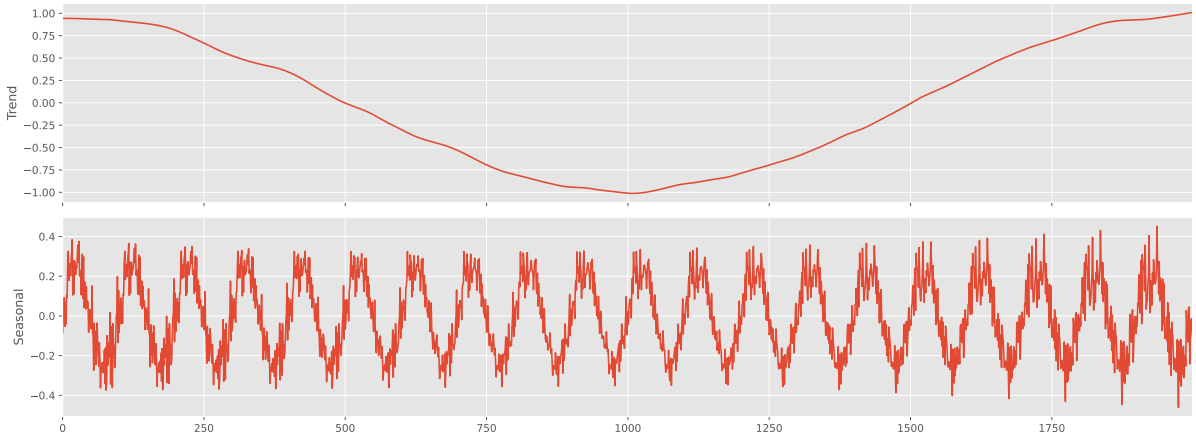


Figure 7.12 – Modèle de base avec sauts égaux à 1

```
# Modèle avec sauts réglés
low_pass_jump = seasonal_jump = int(0.15 * (period + 1))
trend_jump = int(0.15 * 1.5 * (period + 1))
mod = STL(y,period=period,seasonal=seasonal,seasonal_jump=seasonal_jump,
          trend_jump=trend_jump,low_pass_jump=low_pass_jump)
res = mod.fit()
fig = res.plot(observed=False,resid=False)
fig.set_size_inches((16,6))
fig.tight_layout()
plt.show()
```

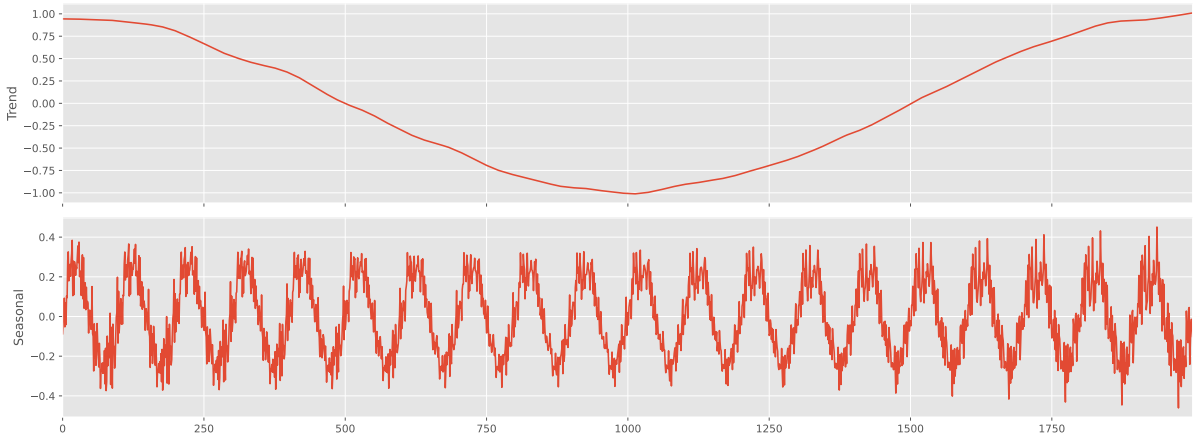


Figure 7.13 – Modèle avec sauts réglés à 15% de leur longueur de fenêtre

7.2.2.4 Prévision avec STL

Si $\widehat{S}_t$ est la composante saisonnière, alors la série désaisonnaliése est construite comme

$$X_t - \widehat{S}_t \tag{7.11}$$

La composante tendancielle n'est pas supprimée, et le modèle de série chronologique doit donc être capable d'ajuster et de prévoir de manière adéquate la tendance si elle est présente. Les

prévisions hors échantillon de la composante saisonnière sont produites sous la forme

$$\widehat{S}_{T+h} = \widehat{S}_{T-k} \tag{7.12}$$

où $k = m - h + M\lfloor (h-1)/m \rfloor$ suit le décalage de période dans le cycle complet de $1, 2, \ldots, m$ où m est la durée de la période. $\lfloor x \rfloor$ est la partie entière inférieure de x.

STLForecast simplifie le processus d'utilisation de STL pour supprimer les saisonnalités, puis d'utilisation d'un modèle de série chronologique standard pour prévoir la tendance et les composantes cycliques.

Exemple 7.6 *Prévision par STL sur la série elec_equip*

Nous utilisons STL pour gérer la saisonnalité, puis un $ARIMA(1,1,0)$ pour modéliser les données désaisonnalisées. La composante saisonnière est prévue à partir du cycle complet de recherche où

$$\mathbb{E}\left(S_{T+h}|\mathcal{F}_T\right) = \widehat{S}_{T-k} \tag{7.13}$$

où $k = m-h+m\lfloor (h-1)/m \rfloor$. La prévision ajoute automatiquement la prévision de la composante saisonnière à la prévision ARIMA.

```
# Prévision par STL
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.forecasting.stl import STLForecast

elec_equip.index.freq = elec_equip.index.inferred_freq
stlf = STLForecast(elec_equip,ARIMA, model_kwargs=dict(order=(1, 1, 0), trend="t"))
stlf_res = stlf.fit()
# Prévision
forecast = stlf_res.forecast(24)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(elec_equip,color="black",label="elecequip");
axe.plot(forecast,color="blue",label="forecast");
axe.set_xlabel("Date");
axe.set_ylabel("equipement");
axe.legend();
plt.show()
```

Signalons que la méthode STL de Statsmodels est prévue pour traiter uniquement des modèles additifs.

7.2.3 Modèle MSTL

Le modèle MSTL (*Multiple Seasonal - Trend decomposition using Loess*) est une extension du modèle STL dans le cas d'une série temporelle avec saisonnalité multiple. Par exemple, considérons la série de demande d'électricité. Elle possède une saisonnalité quotidienne ou journalière (plus de demande pendant la journée que tard le soir), une saisonnalité hebdomadaire (week-end vs jours de semaine), et saisonnalité annuelle (la demande en été et en hiver diffère en raison des besoins de chauffage et de refroidissement). La question que l'on peut se poser est la suivante :

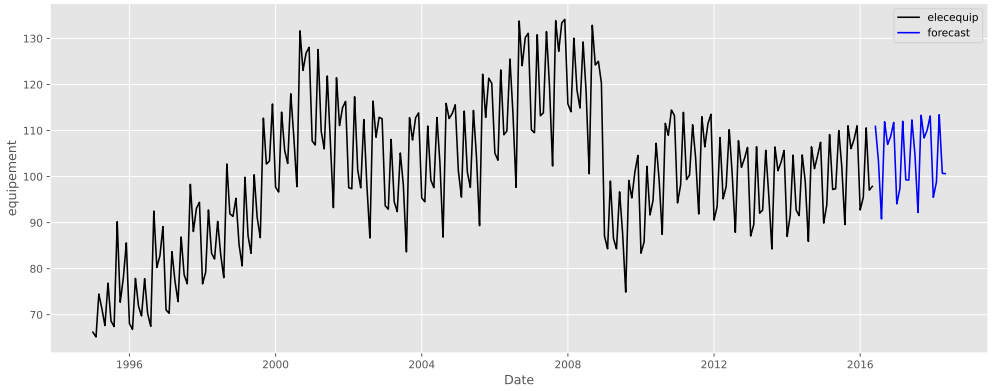


Figure 7.14 – Prévision par STL

Quelle est la fréquence des séries chronologiques multi-saisonnières ? Les séries chronologiques allant de la pollution de l'air à la demande des restaurants peuvent avoir plusieurs composantes saisonnières.

7.2.3.1 Principe de la méthode

Le modèle MSTL, proposé par Bandara, Hyndman, and Bergmeir (2021), est une méthode permettant de décomposer une série chronologique en une composante de tendance, plusieurs composantes saisonnières et une composante résiduelle. MSTL suppose que la série chronologique peut être exprimée sous la forme d'une décomposition additive

$$X_t = Z_t + S_t^{(1)} + S_t^{(2)} + \cdots + S_t^{(N)} + \varepsilon_t \tag{7.14}$$

où chaque composante saisonnière représente une saisonnalité différente (par exemple, quotidienne, hebdomadaire, annuelle, etc.).

7.2.3.2 Fonctionnement

Le modèle MSTL est construit sur la base du modèle STL. L'algorithme MSTL peut être divisé en : 1) étapes de prétraitement et 2) étapes de décomposition.

1) Prétraitement

Dans cette étape, deux prétraitements doivent être effectués. Tout d'abord, la gestion des valeurs manquantes. Contrairement au modèle implémenté sous R où celle ci est effectuée grâce au paramètre `na.interp`, sous Python, l'utilisateur devra au préalable traiter ces valeurs manquantes. Ensuite, la transformation Box - Cox de la série est spécifiée par l'utilisateur. Celle - ci est utilisée si nous pensons que la série temporelle n'est pas décrite par une décomposition additive. La transformée de Box - Cox peut convertir la série temporelle originale en une nouvelle qui peut être décrite par une décomposition additive.

Ces deux étapes peuvent être ignorées si la série chronologique d'entrée ne contient pas de données manquantes et qu'une transformation de Box Cox n'est pas requise.

2) Décomposition

Nous décomposons la partie décomposition de l'algorithme en plusieurs étapes

— **Etape 1 : Extraction de chaque composante saisonnière à l'aide de STL**

Nous passons la période de chaque composante saisonnière que nous voulons extraire à MSTL (par exemple, pour des donnés horaires avec une saisonnalité quotidienne et hebdomadaire `periods = (24,24*7)` - une période de 24 heures pour une saisonnaluté quotidenne, une période de $24 \times 7 = 168$ heures pour une saisonnalité hebdomadaire, etc.).

Nous parcourons chaque composante saisonnière en partant de la période la plus courte (par exemple, quotidienne) jusqu'à la période la plus longue (par exemple, annuelle). A chaque itération, nous extrayons la composante saisonnière via STL puis la soustrayons de la série temporelle. Nous passons ensuite la série chronologique désaisonnalisée résultante àl'itération suivante où nous extrayons la composante saisonnière suivante et ainsi de suite jusqu'à ce que toutes les composantes saisonnières aient été extraites.

Pourquoi commençons - nous à itérer avec la période la plus courte ? Parce que si la saisonnalité la plus longue était extraite en premier, l'algorithme inclurait par erreur la saisonnalité la plus courte dans le cadre de la composante saisonnière la plus longue.

— **Etape 2 : Affiner chacune des composantes saisonnières extraites**

Jusqu'à présent, nous avons une estimation pour chaque composante saisonnière et une série chronologique entièrement désaisonnalisée. Nous parcourons maintenant à nouveau chaque composant saisonnier. A chaque itération, nous ajoutons cette composante saisonnière unique à la chronique entièrement désaisonnalisée à partir de la fin de l'étape 1. En suite, extrayons la même composante saisonnière de cette série chronologique à l'aide de STL. Soustrayons cette nouvelle estimation de la composante saisonnière de la chronique afin qu'elle soit, une fois de plus, entièrement désaisonnalisée.

Cette étape est utilse car la série chronologique que nous transmettons maintenant à STL ne contient que la seule composante saisonnière d'intérêt, la tendance et le bruit. CEla perme à STL de capturer plus facilement toute partie de la composante saisonnière qu'il a manquée à l'étape 1. Répétez cette étape N fois.

— **Etape 3 : Extraction de la tendance**

Extraire la composante tendancielle du dernier ajustement STL qui se produit à l'étape 2 (c'est - à - dire pour la composante saisonnière la plus longue).

— **Etape 4 : Extraction du résidu**

La composante résiduelle est calculée en soustrayant la composante tendancielle de la série chronologique entièrement désaisonnalisée à la fin de l'étape 2.

Après ces quatre étapes, il nous reste une composante tendancielle, un ensemble de composantes saisonnières raffinées et une composante résiduelle.

7.2.3.3 Paramètres MSTL

En pratique, il n'y a que quelques paramètres que nous devons connaître pour utiliser MSTL en Python. Nous résumons ici les paramètres les plus importants :

— `periods` : la période de chaque composante saisonnière, `period`, transmise à STL (par exemple, pour les données horaires avec une saisonnalité quotidienne et hebdomadaire, nous définissons `periods = (24, 24×7)`) ;
— `windows` : les tailles de fenêtres saisonnières, `seasonal`, transmises à STL pour chaque

composant saisonnier respectif (par exemple, `windows=(11,15)`). MSTL utilise des valeurs par défaut basées sur des expéreiences qui ont donné les meilleurs résultats dans @bandara2021mstl ;
— `seasonal_deg` : le degré du polynôme de la composante saisonnière utilisé dans @bandara2021mstl est `seasonal_deg=0` ;
— `lmbda` : si une transformée de Box - Cox est requise, nous devons choisir une valeur pour le paramètre λ d'une transformée de Box - Cox. λ peut être réglé manuellement, `lmbda=0.7`, ou automatiquement, `lmbda="auto"`.

Exemple 7.7 *MSTL appliqué aux données de demande d'électricité à Victoria*

Nous allons examoner un jeu de données du mond eréel où nous pouvons appliquer MSTL. Examinons la demande d'électricité à Victorial, en Australie, jeu de données utilisé dans Bandara, Hyndman, and Bergmeir (2021). Nous pouvons trouver les données originales où la demande est enregistrée à une granularité demi - horaire de 2002 à 2015 https ://github.com/tidyverts/tsibbledata/tree/master/data-raw/vic_elec.

```
# Chargement des données
url = ("https://raw.githubusercontent.com/tidyverts/tsibbledata/master/"
       "data-raw/vic_elec/VIC2015/demand.csv")
df = pd.read_csv(url)
```

Table 7.2 – Demande d'électricité

Date	Period	OperationalLessIndustrial	Industrial
37257	1	3535.867	1086.133
37257	2	3383.499	1088.501
37257	3	3655.528	1084.472
37257	4	3510.447	1085.553
37257	5	3294.697	1081.303
37257	6	3111.846	1086.154

Les dates sont des nombres entiers représentant le nombre de jours à partir d'une date d'origine. La date d'origine de ce jeu de données est `"1899-12-30"`. Les nombres entiers de période font référence à des intervalles de 30 minutes dans une journée de 24 heures, il y en a donc 48 pour chaque jour.

```
# Transformation des données
df["Date"] = df["Date"].apply(
  lambda x: pd.Timestamp("1899-12-30") + pd.Timedelta(x, unit="days"))
df["ds"] = df["Date"] + pd.to_timedelta((df["Period"]-1)*30, unit="m")
```

On s'intéressera à `OperationalLessIndustrial` qui est la demande d'électricité excluant la demande de certains utilisateurs industriels à haute énergie. Nous rééchantillonnons les données à l'heure et restreignons notre vue au premier semestre 2012 pour nous concentrer sur la saisonnalité quotidienne et hebdomadaire.

```
# Extraction de la colonne voulue
timeseries = df[["ds", "OperationalLessIndustrial"]]
timeseries.columns = ["ds", "X"]
# Filtre pour les premiers 149 jours de 2012.
start_date = pd.to_datetime("2012-01-01")
```

```
end_date = start_date + pd.Timedelta("149D")
mask = (timeseries["ds"] >= start_date) & (timeseries["ds"] < end_date)
timeseries = timeseries[mask]

# Rééchantillonnage
timeseries = timeseries.set_index("ds").resample("H").sum()
```

Table 7.3 – Operational Less Industrial

	X
2012-01-01 00 :00 :00	7926.529
2012-01-01 01 :00 :00	7901.827
2012-01-01 02 :00 :00	7255.721
2012-01-01 03 :00 :00	6792.503
2012-01-01 04 :00 :00	6635.984
2012-01-01 05 :00 :00	6548.104

Traçons notre série chronologique (*cf.* figure 7.15) et construisons une intuition sur toute saisonnalité dans les données

```
# Des variables à utiliser plus tard
timeseries["week"] = timeseries.index.isocalendar().week
timeseries["day_of_month"] = timeseries.index.day
timeseries["month"] = timeseries.index.month
# Représentation graphique
axe = timeseries.plot(y="X",figsize=(16,6),color="black",legend=False);
axe.set_ylabel("Demande (MW)");
axe.set_xlabel("temps");
plt.show()
```

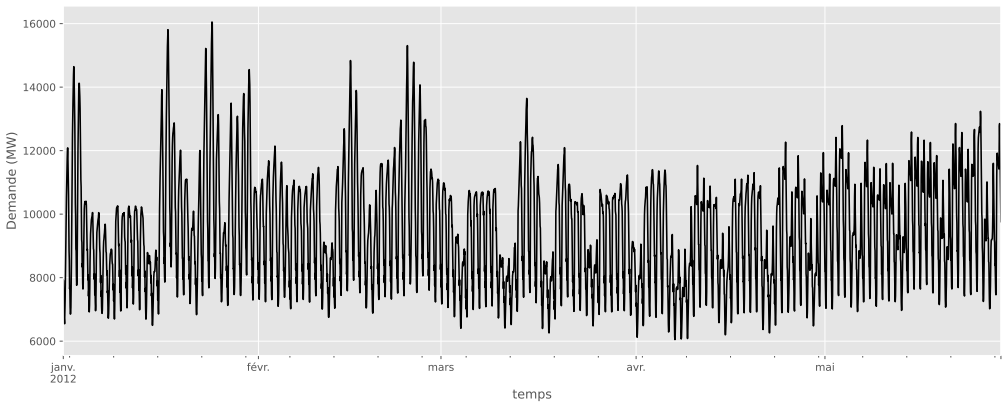


Figure 7.15 – Demande horaire d'électricité (mégawatts) à Victoria, en Australie, entre janvier et mai 2012

Nous attendons à ce qu'il y ait une saisonnalité quotidienne associée à la demande d'électricité. Confirmons cela en traçant la demande horaire pour chaque jour et en la séparant par mois.

```
# Représentation graphique
fig, axe = plt.subplots(nrows=2, ncols=3, figsize=(16,7), sharey=True)
axe = axe.flatten()
```

```
sns_blue = sns.color_palette(as_cmap=True)[0]
MONTHS = ["Jan", "Feb", "Mar", "Apr", "May"]
for ix, month in enumerate(MONTHS):
    # Graphique pour chaque série
    daily_ts = []
    for _, ts in (
        timeseries[["X", "day_of_month", "month"]]
        .query(f"month == {ix+1}")
        .groupby("day_of_month")
    ):
        daily_ts.append(ts.reset_index()["X"])
        ts.reset_index()["X"].plot(
            alpha=0.1, ax=axe[ix], color=sns_blue, label="_no_legend_");
        axe[ix].set_xticks(np.arange(0, len(ts) + 1, 8));
        axe[ix].set_title(month);
    # Représentation graphique de la moyenne de la série
    pd.concat(daily_ts, axis=1).mean(axis=1).plot(
        ax=axe[ix], color="blue", label="mean", legend=True);
    axe[ix].legend(loc="upper left", frameon=False);
    if month in ("Jan", "Feb"):
        axe[ix].tick_params(
            axis="x", which="both", bottom=False, top=False, labelbottom=False);
fig.text(0.5, -0.02, "Heure de lajournée", ha="center");
fig.text(-0.02, 0.5, "Demand (MW)", va="center", rotation="vertical");
fig.suptitle("Demande quotidienne d'électricité à Victoria, Autralie par mois");
fig.delaxes(axe[-1]);
fig.tight_layout()
plt.show()
```

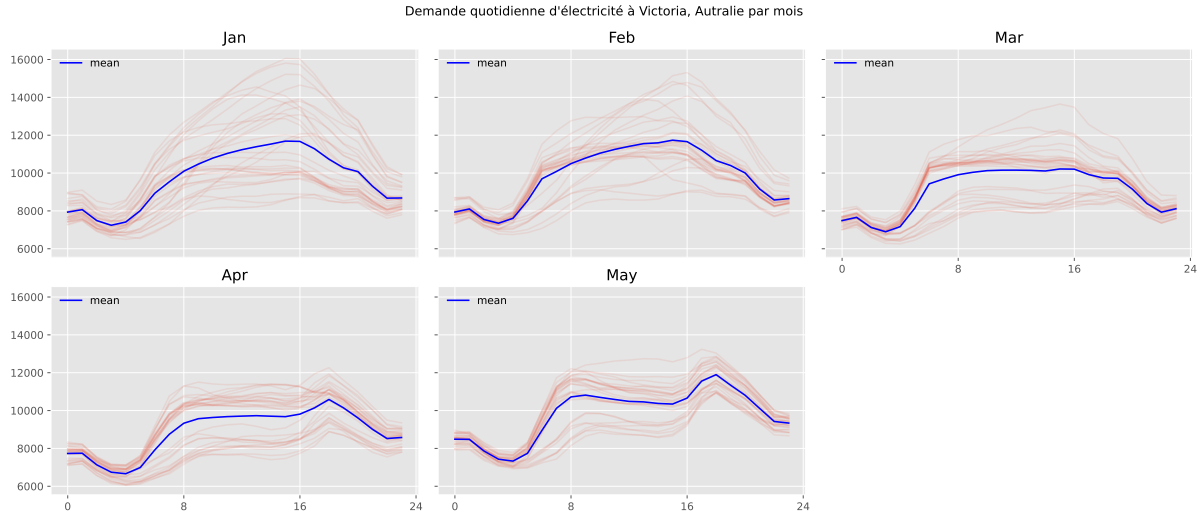


Figure 7.16 – Saisonnalité journalière. La demande pour chaque jour du mois respectif est indiquée par les lignes bleu clair. La demande moyenne est indiquée par la ligne bleu foncé. Il existe un modèle quotidien où il y a plus de demande dans la journée que tard dans la nuit

Sur la figure 7.16, on voit qu'il y a une saisonnalité quotidienne. Nous pouvons également voir la saisonnalité quotidienne changer dans le temps. Pendant les mois d'été (par exemple, janvier), il y a un pic quotidien vers 16 heures, tandis que pendant les mois d'hiver (par exemple, mai),

il y a maintenant deux pics, l'un vers 8 heures et l'autre vers 18 heures. L'une des causes du changement de saisonnalité quotidienne peut être le passage de l'utilisation de la climatisation en été à l'utilisation d'unités de chauffage en hiver.

Maintenant, traçons les données pour examiner la saisonnalité hebdomadaire.

```
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
weekly_ts = []
sns_blue = sns.color_palette(as_cmap=True)
DAYS = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
for week, ts in timeseries.groupby("week"):
    weekly_ts.append(ts.reset_index()["X"])
    ts.reset_index()["X"].plot(alpha=0.1, ax=axe, label="_no_legend_",
                               color=sns_blue);
    plt.xticks(ticks=np.arange(0, 167, 24), labels=DAYS);
pd.concat(weekly_ts, axis=1).mean(axis=1).plot(
    ax=axe, color="blue", label="mean", legend=True);
axe.set_ylabel("Demande (MW)");
axe.set_xlabel("Jour de la semaine");
axe.legend(loc="upper right",frameon=False);
plt.tight_layout()
plt.show()
```

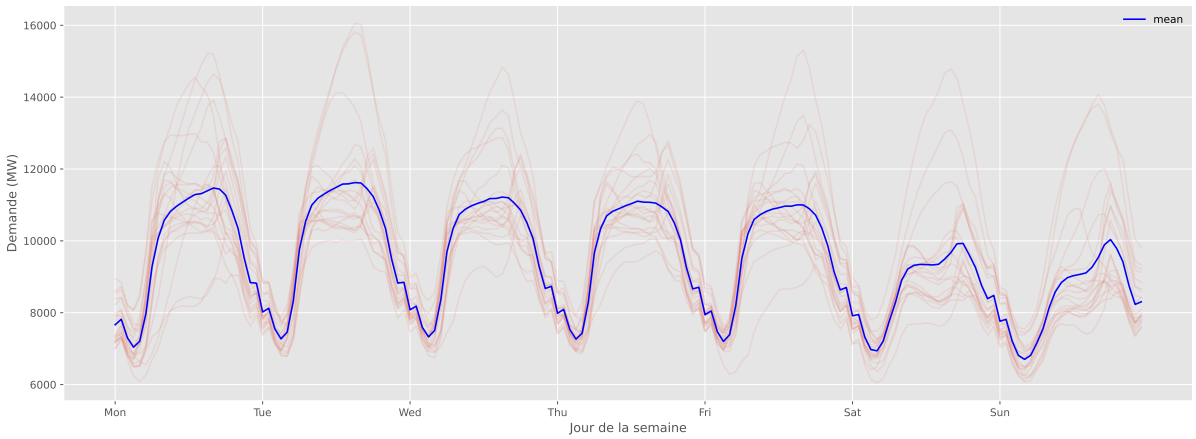


Figure 7.17 – Saisonnalité hebdomadaire. La demande pour chaque semaine de l'année est indiquée par la ligne bleu clair. La demande moyenne est indiquée par la ligne bleu foncé. Les week-ends semblent avoir une demande plus faible en moyenne que les jours de semaine.

Sur la figure 7.17, nous voyons qu'il existe une saisonnalité hebdomadaire, c'est-à-dire qu'il y a moins de demande le week-end que les jours de semaine.

Ces parcelles ont confirmé la présence d'une saisonnalité quotidienne et hebdomadaire et nous ont donné des informations sur la façon dont nous nous attendons à ce qu'elles varient. Passons à autre chose et utilisons MSTL pour extraire ces saisonnalités. Nous pouvons importer MSTL comme ceci :

```
# Décomposition par MSTL
from statsmodels.tsa.seasonal import MSTL, DecomposeResult
```

Le paramètre principal que nous devons spécifier est `periods` la période de chaque composante saisonnière de la série chronologique. Nous nous attendons à ce qu'il y ait une saisonnalité

quotidienne et hebdomadaire, par conséquent, nous avons défini `periods = (24, 24*7)`. Nous pouvons également définir les paramètres qui sont transmis au modèle STL sous-jacent en passant un dictionnaire à `stl_kwargs`.

```
# Application
model = MSTL(timeseries["X"],periods=(24, 24 * 7),stl_kwargs={"seasonal_deg":0})
res = model.fit()
```

Les composantes tendance, saisonnière et résiduelle sont toutes accessibles depuis l'objet résultats `res` :

```
# Concaténation des composantes
res_df = pd.concat([res.trend,res.seasonal,res.resid],axis=1)
```

Table 7.4 – Composantes estimées

	trend	seasonal_24	seasonal_168	resid
2012-01-01 00 :00 :00	10375.73	-1694.800	-165.2829	-589.1197
2012-01-01 01 :00 :00	10365.26	-1602.267	-231.7710	-629.3945
2012-01-01 02 :00 :00	10354.79	-2205.330	-260.7932	-632.9463
2012-01-01 03 :00 :00	10344.33	-2455.881	-387.5940	-708.3480
2012-01-01 04 :00 :00	10333.86	-2372.201	-656.5227	-669.1568
2012-01-01 05 :00 :00	10323.41	-1984.145	-1277.7520	-513.4058

Traçons la décomposition en utilisant `res.plot()`

```
# Représentation graphique des résultats de décomposition par MSTL
plt.rc("figure", figsize=(16,8))
fig = res.plot()
axs = fig.get_axes()
ax_last = axs[-1]
ax_last.xaxis.set_ticks(pd.date_range(start="2012-01-01",freq="MS",periods=5));
plt.setp(ax_last.get_xticklabels(),rotation=0, horizontalalignment="center");
for ax in axs[:-1]:
    ax.get_shared_x_axes().join(ax, ax_last);
    ax.xaxis.set_ticks(pd.date_range(start="2012-01-01",freq="MS",periods=5));
    ax.set_xticklabels([]);
axs[0].set_ylabel("X");
ax_last.set_xlabel("Time");
plt.tight_layout()
plt.show()
```

Inspectons la composante saisonnière quotidienne de MSTL et vérifions si elle capture l'intuition que nous avons obtenue des tracés quotidiens de la figure 7.16 (c'est-à-dire qu'en été, il y a un pic quotidien à 16 heures et en hiver, il y a deux pics).

```
# Représentation graphique de la saisonnalité
fig, axe = plt.subplots(nrows=2, figsize=(16,6))
axe = axe.flatten()
# Saisonnalité quotidienne en Janvier
res.seasonal["seasonal_24"].iloc[: 24 * 3].plot(
    ax=axe[0], label="Saisonnalité quotidienne (Jan)",legend=True);
```

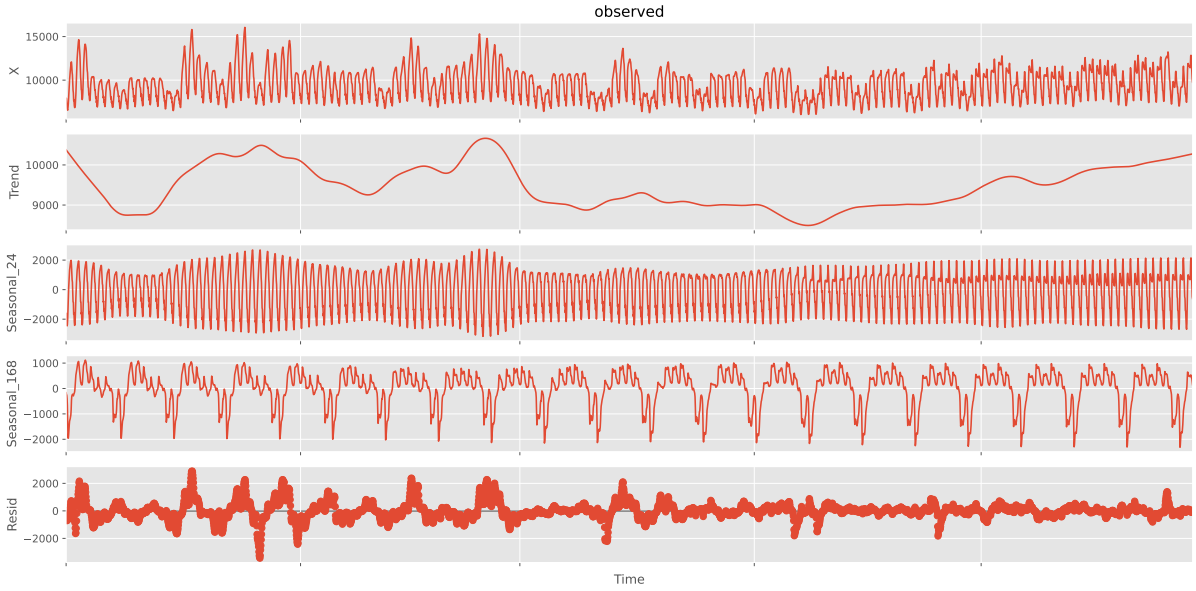


Figure 7.18 – Décomposition MSTL de la série chronologique de la demande d'électricité en tendance, saisonnalité quotidienne (Seasonal_24), saisonnalité hebdomadaire (Seasonal_168) et une composante résiduelle

```
axe[0].set_ylabel(None);
axe[0].set_xlabel(None);
axe[0].legend(loc="upper right", framealpha=0.9);
mask = res.seasonal.index.month == 5 # Saisonnalité quotidienne en Juin
res.seasonal[mask]["seasonal_24"].iloc[: 24 * 3].plot(
    ax=axe[1], label="Saisonnalité quotidienne (May)", legend=True);
axe[1].set_ylabel(None);
axe[1].set_xlabel(None);
axe[1].legend(loc="upper right", framealpha=0.9);
fig.text(0.5, 0, "Time", ha="center", fontsize=25);
fig.text(-0.02, 0.5, "Seasonal_24",va="center",rotation="vertical",fontsize=25);
plt.tight_layout()
plt.show()
```

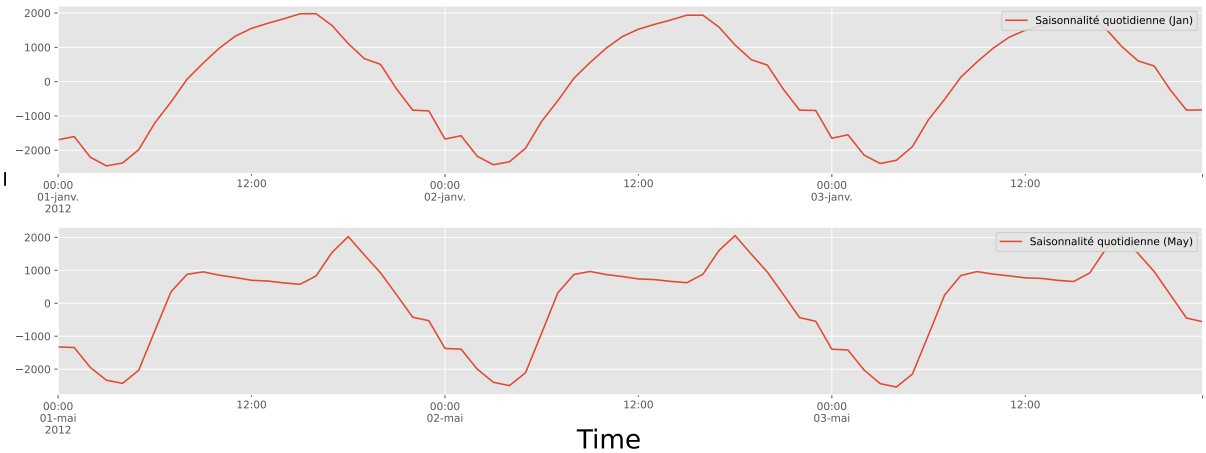


Figure 7.19 – La composante saisonnière journalière extraite par MSTL pendant un échantillon de jours en janvier et mai

Nous voyons que MSTL a capturé correctement la saisonnalité quotidienne et montre même

le pic quotidien supplémentaire en mai (*cf.* Figure 7.19). Cela montre l'utilité de MSTL pour pouvoir modéliser des composantes saisonnières qui changent dans le temps.

Inspectons la composante saisonnière hebdomadaire de MSTL et vérifions si elle capture l'intuition que nous avons obtenue des parcelles hebdomadaires de la figure 7.17 (c'est-à-dire qu'il y a moins de demande pendant les week-ends que les jours de semaine).

```
# Représentation graphique
fig, axe = plt.subplots(nrows=2, figsize=(16,8))
axe = axe.flatten()

# Saisonnalité hebdomadaire en Janvier
start = pd.Timestamp("2012-01-09")  # Lundi
end = start + pd.Timedelta("3W")
res.seasonal["seasonal_168"].loc[start:end].plot(
    ax=axe[0], label="Saisonnalité hebdomadaire (Jan)", legend=True);
axe[0].set_ylabel("seasonal_168");
axe[0].set_xlabel(None);
axe[0].set_ylabel(None);
axe[0].legend(loc="lower left", framealpha=0.9);
props = dict(boxstyle="round", facecolor="white", alpha=0.5)
axe[0].text(0.253,0.95,"week-end",transform=axe[0].transAxes,
            verticalalignment="top",bbox=props);
props = dict(boxstyle="round", facecolor="white", alpha=0.5)
axe[0].text(0.585,0.95,"week-end",transform=axe[0].transAxes,
            verticalalignment="top",bbox=props);
props = dict(boxstyle="round", facecolor="white", alpha=0.5)
axe[0].text(0.92,0.95,"week-end",transform=axe[0].transAxes,
            verticalalignment="top",bbox=props);
weekends = [("2012-01-14", "2012-01-16"),("2012-01-21", "2012-01-23"),
            ("2012-01-28", "2012-01-30")]
for start_, end_ in weekends:
    axe[0].axvspan(start_, end_, alpha=0.1, color="red");
mask = res.seasonal.index.month == 5 # Saisonnalité hebdomadaire en Mai
start = pd.Timestamp("2012-05-7")
end = start + pd.Timedelta("3W")
res.seasonal[mask]["seasonal_168"].loc[start:end].plot(
    ax=axe[1], label="weekly seasonality (May)", legend=True);
axe[1].set_ylabel(None);
axe[1].set_xlabel(None);
axe[1].legend(loc="lower left", framealpha=0.9);
weekends = [("2012-05-12", "2012-05-14"),("2012-05-19", "2012-05-21"),
            ("2012-05-26", "2012-05-28")]
for start_, end_ in weekends:
    axe[1].axvspan(start_, end_, alpha=0.1, color="red");
fig.text(0.5,0, "Time", ha="center");
fig.text(-0.02,0.5,"Seasonal_168",va="center", rotation="vertical",fontsize=25);
plt.tight_layout()
plt.show()
```

Sur la figure 7.20, nous voyons que la composante saisonnière hebdomadaire de MSTL a en effet été en mesure de capter la baisse de la demande le week-end.

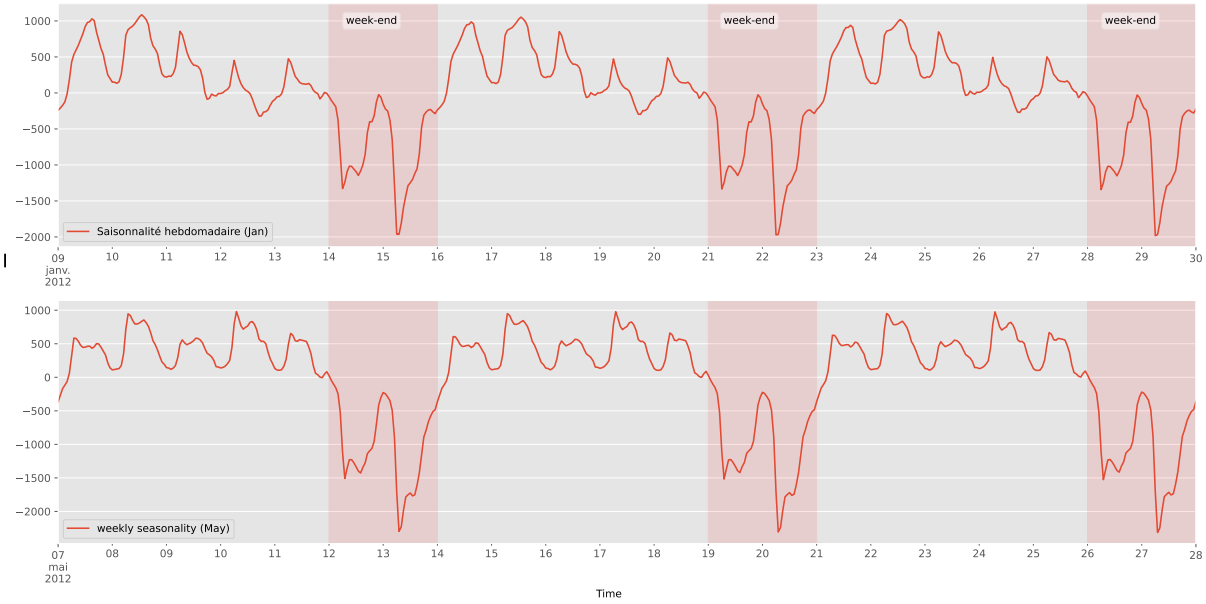


Figure 7.20 – La composante saisonnière hebdomadaire extraite par MSTL sur une période de 3 semaines en janvier et mai. La composante saisonnière hebdomadaire a capté les baisses le week-end et est stable sur les deux mois. Nous pouvons encore voir une certaine saisonnalité quotidienne qui s'est infiltrée dans la composante hebdomadaire

Deuxième partie

Économétrie des processus stationnaires et non stationnaires

8 Introduction à la théorie des processus stationnaires

Sommaire

Lorsque les composantes tendancielles et saisonnières auront été rétirées, ou bien lorsque une opération de différenciation aura permis de les absorber, la série temporelle obtenue doit être modélisée mathématiquement. Il serait trop simpliste de modéliser ces séries à l'aide de suites de variables aléatoires indépendantes et identiquement distribuées. Par exemple, au niveau empirique, on observe le plus souvent de la corrélation entre les vecteurs $(x_1, \dots, x_{T-1})$ et $(x_2, \dots, x_{T-1})$. Tenir compte de cette corrélation est important pour prévoir plus précisément les valeurs futures. L'objectif de ce chapitre est d'introduire un formalisme mathématique qui permettra de modéliser ces composantes aléatoires.

8.1 Quelques généralités sur les processus stochastiques

8.1.1 Processus stochastique et stationnarité

8.1.1.1 Processus stochastique ou aléatoire

Une série temporelle est une réalisation d'un processus stochastique. On utilise le terme de processus aléatoire (ou stochastique) pour décrire une variable dont le comportement ne peut pas être entièrement déterminé par une relation déterministe.

Définition 8.1 *Processus stochastique*

Un processus aléatoire (ou stochastique) est une suite de variables aléatoires réelles indexée dans le temps et définie sur un espace des états de la nature par :

$$X_t, \quad t \in \mathbb{Z} \tag{8.1}$$

Ainsi, pour chaque instant du temps, la valeur de la quantité étudiée X_t est appelée variable aléatoire et l'ensemble des valeurs X_t quand t varie est appelé processus aléatoire ou stochastique.

Theorème 8.1 *Soit* $\{\mu_n : \mathcal{B}(\mathbb{R}^{n+1}) \longrightarrow [0,1]; n \in \mathbb{N}\}$ *une famille de mesures de probabilités qui vérifie la condition suivante : pour tout* $n \in \mathbb{N}$ *et* $A_0, \dots, A_n \in \mathcal{B}(\mathbb{R})$,

$$\mu_n(A_0 \times \cdots A_n) = \mu_{n+1}(A_0 \times \cdots A_n \times \mathbb{R}) \tag{8.2}$$

alors, il existe un espace de probabilité $(\Omega, \mathcal{A}, \mathbb{P})$ *et un processus stochastique* $\{X_t : t \in \mathbb{N}\}$ *tel que pour tout entier* n, *la loi de* $(X_0, \dots, X_n)$ *sous* $\mathbb{P}$ *coïncide avec* μ_n.

Le théorème (8.1) s'applique en particulier pour construire une suite $(X_t)_{t\in\mathbb{N}}$ de variables aléatoires indépendantes et identiquement distribuées et toutes de loi μ. Il suffit de poser $\mu = \mu^{\otimes n}$. On peut également, grâce à ce théorème, définir une suite de variables aléatoires indépendantes mais non identiquement distribuées.

Hormis le cas des variables aléatoires indépendantes, une classe importante de processus stochastiques est celle des processus gaussiens.

Définition 8.2 *Processus gaussien*

Le processus $(X_t)_{t\in\mathbb{Z}}$ est appelé processus gaussien si la distribution de $(X_{t_1+h}, X_{t_2+h}, \dots, X_{t_n+h})$ notée $f\left(X_{t_1+h}, X_{t_2+h}, \dots, X_{t_n+h}\right)$ suit une loi normale multivariée pour tout h. En d'autres termes, le processus $(X_t)_{t\in\mathbb{Z}}$ est dit gaussien si, pour tout entier n, le vecteur $(X_0, \dots, X_n)$ suit une loi gaussienne sur $\mathbb{R}^{n+1}$.

La loi d'un processus gaussien ne dépend donc que des moyennes $m(t) = \mathbb{E}(X_t)$ et des covariances $\gamma(t,s) = \mathrm{Cov}(X_t, X_s)$ pour $t, s \in \mathbb{N}$. La fonction γ vérifie les propriétés suivantes.

Propriété 8.1 *Toute fonction de covariance* γ *vérifie les trois propriétés suivantes :*

1. $\gamma(t,s) = \gamma(s,t)$ *pour tout* $s, t \in \mathbb{N}$.
2. $|\gamma(s,t)| \leq \sqrt{\gamma(s,s)} \times \sqrt{\gamma(t,t)}$ *pour tout* $s, t \in \mathbb{N}$.
3. γ *est sémi - définie positive, i.e pour tous* $n \in \mathbb{N}$ *et* $\alpha_0, \dots, \alpha_n \in \mathbb{R}$, *on a* $\sum_{i,j=0}^{i,j=n} \alpha_i \alpha_j \gamma(i,j) \geq 0$.

Inversement, toute fonction $\gamma : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{R}$ *qui satisfait les propriétés* 1 *à* 3 *est la fonction de covariance d'un processus gaussien.*

Preuve

Si γ est une fonction de covariance, les deux premières propriétés sont évidentes (la deuxième résulte de l'inégalité de Cauchy - Schwarz). Pour la dernière propriété, il suffit de remarquer que :

$$\begin{aligned}\sum_{i,j=0}^{i,j=n} \alpha_i \alpha_j \gamma(i,j) &= \sum_{i=0}^{i=n} \alpha_i^2 \mathrm{Var}(X_i) + 2 \sum_{i\neq j=0}^{i\neq j=n} \alpha_i \alpha_j \mathrm{Cov}(X_i, X_j) \\ &= \mathrm{Var}\left(\sum_{i=0}^{i=n} \alpha_i X_i\right)\end{aligned}$$

Inversement, si γ vérifie les propriétés 1 à 3 alors on peut appliquer le théorème (8.1) en définissant pour tout $n \in \mathbb{N}$, μ_n comme étant la loi gaussienne de moyenne 0 et de matrice de variance - covariance $(\gamma(i,j))_{0\leq i,j\leq n}$.

Définition 8.3 *Distribution normale multivariée*

Une variable aléatoire X de dimension n est dite suivre une distribution normale multivariée d'espérance μ et de matrice de variance-covariance Σ (symétrique et définie positive) si :

$$f_{X,\mu,\Sigma}(x) = \frac{1}{(2\pi)^{n/2}\det(\Sigma)^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^t\Sigma^{-1}(x-\mu)\right] \tag{8.3}$$

8.1.1.2 Processus stationnaire ou stationnarité

La notion de stationnarité est un concept clé pour la validité externe d'une régression sur séries temporelles. En effet, la stationnarité dit que, d'un point de vue statistique, le passé est comparable au présent et au futur. Elle assure l'utilisation du modèle en dehors de la période sur laquelle il a été estimé.

On considère une suite de variables aléatoires $(X_t)_{t=0,1,\dots}$. On dit que cette suite est stationnaire en moyenne lorsque la moyenne de chacune des variables de la suite est identique :

$$\mathbb{E}(X_t) = \mathbb{E}(X_{t-1}) = \cdots = \mathbb{E}(X_0) \tag{8.4}$$

et on dira qu'elle est stationnaire en variance lorsque la variance de chacune des variables de la suite est identique :

$$\mathrm{V}(X_t) = \mathrm{V}(X_{t-1}) = \cdots = \mathrm{V}(X_0) \tag{8.5}$$

Définition 8.4 *Stationnarité stricte ou forte*

Un processus $(X_t)_t$ est stationnaire au sens strict, si pour tout entier $n \in \mathbb{N}$, pour tout n-uplet $(t_1, t_2, \dots, t_n) \in \mathbb{T}^n$ avec $t_1 < t_2 < \cdots < t_n$ et pour tout décalage temporel $h \in \mathbb{T}$, la loi de $\left(X_{t_1+h}, X_{t_2+h}, \dots, X_{t_n+h}\right)$ ne dépend pas de h.

Les vecteurs $\left(X_{t_1}, X_{t_2}, \dots, X_{t_n}\right)$ et $\left(X_{t_1+h}, X_{t_2+h}, \dots, X_{t_n+h}\right)$ ont la même loi de probabilité « jointe »

$$\left(X_{t_1}, X_{t_2}, \dots, X_{t_n}\right) \overset{d}{=} \left(X_{t_1+h}, X_{t_2+h}, \dots, X_{t_n+h}\right) \tag{8.6}$$

En d'autres termes, cela signifie que $\forall(x_1, \dots, x_n)$ $\forall(t_1, t_2, \dots, t_n)$ et $\forall h$:

$$\mathbb{P}\left(X_{t_1} < x_1, X_{t_2} < x_2, \dots, X_{t_n} < x_n\right) = \mathbb{P}\left(X_{t_1+h} < x_1, X_{t_2+h} < x_2, \dots, X_{t_n+h} < x_n\right) \tag{8.7}$$

Ceci implique aussi que tous les moments finis d'ordre n sont invariants au cours du temps. Cependant, cette définition de la stationnarité est trop restrictive, difficile à atteindre, c'est pour cela que l'on a défini la stationnarité au second ordre.

Définition 8.5 *Stationnarité faible ou du second ordre*

Quand la stricte stationnarité n'est pas obtenue, on peut penser à une stationnarité faible. Un processus $(X_t)_t$ est dit être stationnaire au premier ordre si sa première fonction de moment est invariante au cours du temps. Il est dit stationnaire au second ordre si sa première et sa seconde

fonction des moments sont invariantes au cours du temps. Un processus stationnaire de second ordre aura une moyenne constante, une variance constante finie et une covariance qui est une fonction de corrélation qui ne dépend que la différence des deux instants, c'est - à - dire :

1. $\mathbb{E}\left(X_t^2\right) < \infty \ \forall t \in \mathbb{Z}$
2. $\mathbb{E}\left(X_t\right) = \mu \ \forall t \in \mathbb{Z}$
3. $\text{Cov}(X_t, X_{t+h}) = \gamma(h) \ \forall t, h \in \mathbb{Z}$

On remarquera que la variance $\sigma_X^2 = \text{Cov}(X_t, X_t) = \gamma(0)$ est également indépendante du temps. Le fait que la variance soit constante au cours du temps traduit la propriété d'homoscédasticité.

Propriété 8.2

Si $(X_t)_{t\in\mathbb{Z}}$ est un processus stationnaire au second ordre et $(\alpha_i)_{i\in\mathbb{Z}}$ une suite de nombres réels absolument sommables $\left(\sum_{i=-\infty}^{\infty} |\alpha_i| < \infty\right)$ alors le processus $(Y_t)_{t\in\mathbb{Z}}$ défini par

$$Y_t = \sum_{i=-\infty}^{\infty} \alpha_i X_{t-i}, \quad t \in \mathbb{Z} \tag{8.8}$$

est un processus stationnaire.

Remarque 8.1

La notion de stationnarité stricte et de stationnarité au second ordre se confondent pour les processus gaussiens car ceux-ci sont entièrement résumés par leurs deux premiers moments (car pour un processus gaussien, les moments d'ordre supérieurs à 2 sont automatiquement nuls et pour tout $n \in \mathbb{N}^*$ et pour tout $h, t_1, ..., t_n \in \mathbb{Z}$, les vecteurs aléatoires $\left(X_{t_1}, X_{t_2}, ..., X_{t_n}\right)'$ et $\left(X_{t_1+h}, X_{t_2+h}, ..., X_{t_n+h}\right)'$ ont la même moyenne et matrice de covariance et donc la même distribution). Par contre, dès que l'on sort du cadre gaussien comme dans les modèles GARCH, on n'a plus coïncidence entre les deux termes.

Exemple 8.1 *Exemple de processus stationnaires*

1. Une suite de variables aléatoires i.i.d $(X_t)_{t\in\mathbb{Z}}$ est strictement stationnaaires.
2. Un exemple de processus stationnaire très utilisé est le bruit blanc. On dit que $(\varepsilon_t)_{t\in\mathbb{Z}}$ est un bruit blanc (faible) si $\mathbb{E}(\varepsilon_t) = 0$, $\text{Var}(\varepsilon_t) = \sigma_\varepsilon^2$ et $\text{Cov}(\varepsilon_t, \varepsilon_s) = 0$ si $t \neq s$. Lorsque les variables sont en plus i.i.d, on parle de bruit blanc fort. Un bruit blanc fort est un bruit blanc faible mais le contraire n'est pas nécessairement vrai. Pour construire un contre - exemple,on peut par exemple considérer une suite $(\varepsilon_t)_{t\in\mathbb{Z}}$ de variables aléatoires indépendantes et telle que :

$$\mathbb{P}\left(\varepsilon_t = \sqrt{|t|+1}\right) = \mathbb{P}\left(\varepsilon_t = -\sqrt{|t|+1}\right) = \frac{1}{2(|t|+1)}, \quad \mathbb{P}\left(\varepsilon_t = 0\right) = 1 - \frac{1}{|t|+1}$$

3. Lorsque $(\varepsilon_t)_{t\in\mathbb{Z}}$ est un bruit blanc, on peut construire des processus stationnaires sous la forme de moyennes mobiles en posant :

$$X_t = \sum_{i=-m_1}^{i=m_2} \theta_i \varepsilon_{t+i}$$

En effet, X_t est de carré intégrale car somme de variables aléatoires de carré intégrale. De plus, $\mathbb{E}(X_t) = 0$ et si $h \geq 0$, $s \in \mathbb{Z}$:

$$\mathrm{Cov}(X_t, X_{t+h}) = \sigma_\varepsilon^2 \sum_{i,j=-m_1}^{i,j=m_2} \theta_i \theta_j 1_{i=h+j}$$

On voir alors que $(X_t)_{t\in\mathbb{Z}}$ est stationnaire en posant :

$$\gamma(h) = \begin{cases} \sigma_\varepsilon^2 \displaystyle\sum_{i=-m_1}^{i=m_2-h} \theta_i \theta_{i+h} & \text{si} \quad 0 \leq h \leq m_2 + m_1 \\ 0 & \text{si} \quad h > m_2 + m_1 \end{cases}$$

Il n'est pas utile de refaire le calcul pour $h < 0$. En effet, on a pour $h < 0$,

$$\mathrm{Cov}(X_t, X_{t+h}) = \mathrm{Cov}(X_{t+h}, X_{t+h-h}) = \gamma(-h),$$

d'après les calculs précédents. Sans tenir compte du signe de h, on posera :

$$\gamma(h) = \begin{cases} \sigma_\varepsilon^2 \displaystyle\sum_{i=-m_1}^{i=m_2-|h|} \theta_i \theta_{i+|h|} & \text{si} \quad 0 \leq |h| \leq m_2 + m_1 \\ 0 & \text{sinon} \end{cases}$$

4. On peut aussi construire des moyennes mobiles d'ordre infini à partir d'un bruit blanc $(\varepsilon_t)_{t\in\mathbb{Z}}$ et d'une suite $(\theta_i)_{i\mathbb{N}}$ de nombres réels tels que $\displaystyle\sum_{i\in\mathbb{N}} \theta_i^2 < +\infty$. Pour $t \in \mathbb{Z}$, posons $X_t = \displaystyle\sum_{i=0}^{+\infty} \theta_i \varepsilon_{t-i}$. La somme infinie considérée a bien un sens. On la définit comme la limite dans $\mathbb{L}^2$ de la variable $X_t^{(N)} = \displaystyle\sum_{i=0}^{i=N} \theta_i \varepsilon_{t-i}$ (formellement, on peut vérifier que la suite de $(X_t^{(N)})_{t\in\mathbb{Z}}$ est une suite de Cauchy dans $\mathbb{L}^2$, espace des variables aléatoires de carré intégrale). Il est alors clair que :

$$\mathbb{E}(X_t) = \lim_{N\longrightarrow+\infty} \sum_{i=0}^{i=N} \theta_i \mathbb{E}(\varepsilon_{t+i}) = 0$$

De plus, pour $h \in \mathbb{N}$ donné, on a :

$$\begin{aligned}\mathrm{Cov}\left(X_t, X_{t+h}\right) &= \lim_{N\longrightarrow+\infty} \mathrm{Cov}\left(X_t^{(N)}, X_{t+h}^{(N)}\right)\\ &= \lim_{N\longrightarrow+\infty} \sigma_\varepsilon^2 \sum_{i=0}^{i=N-h} \theta_i\theta_{i+h}\\ &= \sigma_\varepsilon^2 \sum_{i\in\mathbb{N}} \theta_i\theta_{i+h}\end{aligned}$$

Comme nous le verrons plus loin, on obtient presque tous les processus faiblement stationnaires à partir de ces moyennes mobiles d'ordre infini. On peut aussi construire des moyennes mobiles bilatérales d'ordre infini en posant $X_t = \sum_{i=-\infty}^{+\infty} \theta_i\varepsilon_{t-i}$ en convenant que :

$$\sum_{i=-\infty}^{+\infty} \theta_i^2 = \sum_{i=0}^{+\infty} \theta_i^2 + \sum_{i=1}^{+\infty} \theta_{-i}^2 < +\infty$$

La fonction d'autocovariance est alors donnée $\gamma(h) = \sum_{i=-\infty}^{+\infty} \theta_i\theta_{i+|h|}$ (on peut même supprimer la valeur absolue car $\sum_{i\in\mathbb{Z}} \theta_i\theta_{i+h} = \sum_{i\in\mathbb{Z}} \theta_i\theta_{i-h}$).

8.1.2 Ergodicité et théorème de Wold

8.1.2.1 Ergodicité

Pour estimer la loi d'un processus, on cherche à accumuler de l'information en faisant tendre le nombre d'observations vers l'infini. Pour que ce mécanisme d'accumulation fonctionne, il faut que le processus ait une mémoire finie, c'est-à-dire qu'à partir d'un certain nombre d'observations il n'y a plus d'informations nouvelles, mais simplement confirmation des informations passées. Par exemple, dans le problème de l'estimation de la moyenne, on veut que la moyenne empirique soit un estimateur consistant et que la variance de cet estimateur tende vers zéro. On se rend compte que si un processus est cyclique, ce qui revient à dire que des observations très éloignées peuvent être corrélées entre elles, on n'arrive pas à accumuler de l'information. Cette propriété de limitation de la mémoire d'un processus s'appelle ergodicité.

Définition 8.6 *processus ergodique*

Un processus stationnaire est dit ergodique si l'on peut calculer l'ensemble de ses caractéristiques (moyenne, variance, fonction d'autocorrélation) à partir d'une seule trajectoire du processus, c'est-à-dire à partir d'une observation du processus et, par conséquent, de façon pratique, à partir de la série temporelle observée suffisamment longtemps. Ainsi, pour un processus stationnaire, l'hypothèse d'ergodicité pose l'équivalence entre moyenne d'ensemble (espérance mathématique) et moyenne temporelle (ou spatiale) sur un domaine « infini », soit :

$$\mu_X = \mathbb{E}(X_t) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} X_t \tag{8.9}$$

L'espérance du processus est la limite de la moyenne des valeurs des observations de la série quand la durée d'observation tend vers l'infini. Ainsi, on dit qu'une suite stationnaire est ergodique si elle satisfait la loi forte des grands nombres.

Définition 8.7

Un processus stationnaire au second ordre est dit ergodique si :

$$\lim_{T\to\infty}\frac{1}{T}\sum_{h=1}^{T}\gamma(h)=0 \tag{8.10}$$

où $\gamma(h)$ exprime l'autocovariance d'ordre h.

Propriété 8.3 *Condition d'ergodicité*

Une condition nécessaire, mais non suffisante pour qu'un processus stationnaire au second ordre soit ergodique est qu'il satisfasse la propriété suivante :

$$\lim_{h\to\infty}\gamma(h)=0 \tag{8.11}$$

Propriété 8.4

Soit $(X_t)_{t=1,\dots,T}$ un processus stationnaire second ordre tel que $\mathbb{E}(X_t)=\mu$ et $\mathrm{V}(X_t)=\sigma_X^2$. Soit $\overline{X}_T=\dfrac{1}{T}\sum_{t=1}^{T}X_t$. Si $\overline{X}_T$ converge en probabilité vers μ quand $T\to\infty$ alors le processus est ergodique pour la moyenne.

Remarque 8.2

1. Le théorème d'ergodicité statistique concerne l'information qui peut être obtenue à partir d'une moyenne sur le temps et concernant la moyenne commune à tout instant.
2. Il faut noter que le concept d'ergodicité repose sur une indépendance asymptotique alors que la stationnarité concerne l'indépendance par rapport au temps du processus.

Exemple 8.2

Monter que le processus stochastique définie par

$$X_t=\begin{cases}\mu_0 & \text{si} \quad t=0\\ X_{t-1} & \text{si} \quad t>0\end{cases} \tag{8.12}$$

avec $\mu_0\sim\mathcal{N}(0,\sigma_\mu^2)$ est stationnaire mais non ergodique.

Solution

Il est évident que quel que soit t, $X_t=\mu_0$, par conséquent :

1. $\mathbb{E}(X_t)=\mathbb{E}(\mu_0)=0$
2. $\mathrm{V}(X_t)=\mathrm{V}(\mu_0)=\sigma_\mu^2$
3. $\mathrm{Cov}(X_t,X_{t+h})=\mathbb{E}(X_tX_{t+h})=\sigma_\mu^2$

Ainsi, $(X_t)_t$ est stationnaire. Par contre, l'ergodicité pour la moyenne voudrait que l'on ait $\overline{X}=\dfrac{1}{T}\sum_{t=1}^{T}X_t\to 0$ alors qu'il est évident que $\overline{X}=\dfrac{1}{T}\sum_{t=1}^{T}X_t=\mu_0$. Donc il n'y a pas ergodicité.

8.1.2.2 Théorème de représentation de Wold

Un processus stochastique paramétrique se définit à partir de la distribution jointe des observations ou de ses premiers moments. Un processus stochastique non paramétrique se définit au contraire à partir d'un mécanisme de génération qui est indexé par des paramètres. Il est possible de caractériser ce mécanisme de manière très général au moyen du théorème de Wold (1954) dans le cas linéaire.

Ce théorème montre que tout processus stationnaire linéaire peut être représenté de manière unique par la somme de deux composantes indépendantes, une composante régulière d_t parfaitement prévisible parfois appelée déterministe et une composante stochastique u_t.

Définition 8.8 *Théorème de représentation de Wold*

Si $(X_t)_{t\in\mathbb{Z}}$ est un processus stationnaire au sens faible de moyenne nulle alors X_t peut se décomposer de la manière suivante :

$$X_t = d_t + u_t \tag{8.13}$$

avec

$$u_t = \sum_{i=0}^{\infty} \alpha_i \varepsilon_{t-i} \tag{8.14}$$

où ε_t est un bruit blanc, c'est-à-dire un processus de moyenne nulle, de variance constante et non autocorrélé. Il représente l'erreur faite en prévoyant X_t à partir d'une fonction linéaire et son historique. La variable d_t est non corrélée aux ε_{t-i} $\forall i \in \mathbb{Z}$.

Remarque 8.3

Si la partie déterministe d_t est dominante par rapport à la partie stochastique u_t, on dira que la série est fortement prévisible. Dans le cas contraire, on dira que la série est fortement aléatoire.

Ce théorème est à la base de la modélisation des séries temporelles stationnaires. La composante stochastique est exprimée sous la forme de ce que l'on appelle un processus moyenne mobile infinie. Un des buts de la modélisation consiste à approximer cette moyenne mobile infinie par un processus ayant un nombre fini de paramètres. C'est par exemple l'objet des processus AR, MA et ARMA.

8.2 Caractéristique d'une série temporelle

8.2.1 Moyenne et variance

Sous l'hypothèse de stationnarité, la moyenne $m_X = \mathbb{E}(X_t)$ définie le niveau à travers lequel la série fluctue (change) et la variance $\sigma_X^2 = \mathrm{V}(X_t) = \mathbb{E}\left[(X_t - \mu_X)^2\right]$ mesure la dispersion ou la variation de la série autour de la moyenne.

Si on considère une chronique $X_t, t = 1, \ldots, T$ issue d'un processus $(X_t)_{t\in\mathbb{Z}}$, alors on estime la moyenne et la variance du processus (m_X, σ_X^2) à l'aide de la moyenne empirique et de la variance empirique de la série, définies respectivement par :

$$\overline{X} = \frac{1}{T}\sum_{t=1}^{t=T} X_t \quad \text{et} \quad \hat{\sigma}_X^2 = \frac{1}{T}\sum_{t=1}^{t=T}\left(X_t - \overline{X}\right)^2 \tag{8.15}$$

Sous Python, les fonctions np.mean et np.var de la librairie numpy calculent la moyenne et la variance d'une série respectivement.

```
# Moyenne et variance de la chronique
moyenne = np.mean(default_data)
variance = np.var(default_data,ddof=0)
```

Le paramètre `ddof` est un paramètre de correction de la variance car il permet de spécifier si le dénominateur de la variance doit être égal à T ou à $T -$ ddof.Par défaut, il est fixé à 0.

8.2.2 Fonction d'autocovariance et d'autocorrélation

8.2.2.1 Fonction d'autocovariance

La covariance d'une série chronologique stationnaire est fonction d'une seule variable : $h = t - s$ le décalage temporel entre les deux observations.

Définition 8.9 *Fonction d'autocovariance*

Soit $(X_t)_{t\in\mathbb{Z}}$ une série chronologique stationnaire. Sa fonction d'autocovariance, notée $\gamma(.)$, est définie, pour tout $h \in \mathbb{Z}$ par

$$\gamma(h) = \mathrm{Cov}(X_h, X_0) \tag{8.16}$$

La fonction d'autocovariance mesure la covariance entre deux valeurs de X_t séparées par un certain délai. Elle fournit des informations sur la variabilité de la série et sur les liaisons temporelles qui existent entre les différentes composantes de la série X_t. Sa formule est donnée par :

$$\gamma(h) = \frac{1}{T-h}\sum_{t=h+1}^{t=T}(X_t - \overline{X}_1)(X_{t-h} - \overline{X}_2) \tag{8.17}$$

avec $\overline{X}_1 = \dfrac{1}{T-h}\displaystyle\sum_{t=h+1}^{t=T} X_t$, $\overline{X}_2 = \dfrac{1}{T-h}\displaystyle\sum_{t=h+1}^{t=T} X_{t-h}$ et T le nombre d'observations.

Si (X_t) est stationnaire, alors il est clair que pour tout $t \in T$, on a :

$$\mathrm{Cov}(X_{t+h}, X_t) = \cdots = \mathrm{Cov}(X_{h+1}, X_1) = \mathrm{Cov}(X_h, X_0) = \gamma(h) \tag{8.18}$$

Un estimateur de cette fonction d'autocovariance est la fonction d'autocovariance empirique, noté $\hat{\gamma}(h)$, définie par

$$\hat{\gamma}(h) = \frac{1}{T-h}\sum_{t=h+1}^{t=T}(X_t - \overline{X})(X_{t-h} - \overline{X}) \tag{8.19}$$

où $\overline{X} = \frac{1}{T}\sum_{t=1}^{t=T} X_t$ est la moyenne de la série calculée sur l'ensemble des T périodes ou observations. La fonction arma_acovf() permet de trouver les fonctions d'autocovariance.

```
# Autocovariance d'ordre h
from statsmodels.tsa.arima_process import arma_acovf
acovf = arma_acovf(ar,ma,nobs=h)
print(acovf)
```

où `ar` et `ma` sont respectivement les coefficients autorégressifs et moyennes mobiles (y compris la constante). `nobs` correspond à la valeur $h \in \mathbb{N}$.

Si on dispose d'une série temporelle $X_t, t = 1, \dots, T$, alors on estime la fonction d'autocovariance d'ordre h en utilisant la fonction acovf() de la fonction suivante :

```
# Autocovariance d'ordre h
from statsmodels.tsa.stattools import acovf
acovf = acovf(default_data,nlag=h)
print(acovf)
```

Remarque 8.4

La fonction d'autocovariance d'un processus $(X_t)_t$ stationnaire vérifie les propriétés suivantes :

1. $\gamma(0) = \text{Cov}(X_t, X_t) = \mathbb{E}\left([X_t - \mathbb{E}(X_t)]^2\right) = \text{V}(X_t) = \sigma_X^2 \geq 0$
2. $|\gamma(h)| \leq \gamma(0)$
3. $\gamma(h) = \gamma(-h)$: fonction paire

Exemple 8.3

On considère le processus X_t défini comme suit :

$$X_t = \varepsilon_t - \varepsilon_{t-1} \tag{8.20}$$

où (ε_t) est un processus stationnaire de moyenne nulle et de variance σ_ε^2 et telle que $\text{Cov}(\varepsilon_t, \varepsilon_s) = 0$ si $t \neq s$.

1. Calculer l'espérance et la variance de X_t
2. Calculer la fonction d'autocovariance $\gamma(h)$ $\forall h \geq 0$.

Solution

L'espérance mathématique de X_t est égale à :

$$\mathbb{E}(X_t) = \mathbb{E}(\varepsilon_t - \varepsilon_{t-1}) = \mathbb{E}(\varepsilon_t) - \mathbb{E}(\varepsilon_{t-1}) = 0 \tag{8.21}$$

et sa variance vaut :

$$\text{V}(X_t) = \text{V}(\varepsilon_t) + \text{V}(\varepsilon_{t-1}) - 2\text{Cov}(\varepsilon_t, \varepsilon_{t-1}) = 2\sigma_\varepsilon^2 \tag{8.22}$$

On recherche la fonction d'autocovariance $\gamma(h)\ \forall h > 1$:

$$\begin{aligned}
\gamma(1) &= \mathrm{Cov}(X_t, X_{t-1}) = \mathrm{Cov}(\varepsilon_t - \varepsilon_{t-1}, \varepsilon_{t-1} - \varepsilon_{t-2}) \\
&= \underbrace{\mathrm{Cov}(\varepsilon_t, \varepsilon_{t-1})}_{=0} - \underbrace{\mathrm{Cov}(\varepsilon_t, \varepsilon_{t-2})}_{=0} - \underbrace{\mathrm{Cov}(\varepsilon_{t-1}, \varepsilon_{t-1})}_{=\mathrm{V}(\varepsilon_{t-1})=\mathbb{E}(\varepsilon_{t-1}^2)} + \underbrace{\mathrm{Cov}(\varepsilon_{t-1}, \varepsilon_{t-2})}_{=0} = -\sigma_\varepsilon^2 \\
\gamma(2) &= \mathrm{Cov}(X_t, X_{t-2}) = \mathrm{Cov}(\varepsilon_t - \varepsilon_{t-1}, \varepsilon_{t-2} - \varepsilon_{t-3}) \\
&= \underbrace{\mathrm{Cov}(\varepsilon_t, \varepsilon_{t-2})}_{=0} - \underbrace{\mathrm{Cov}(\varepsilon_t, \varepsilon_{t-3})}_{=0} - \underbrace{\mathrm{Cov}(\varepsilon_{t-1}, \varepsilon_{t-2})}_{=0} + \underbrace{\mathrm{Cov}(\varepsilon_{t-1}, \varepsilon_{t-3})}_{=0} = 0
\end{aligned}$$

On a donc : $\gamma(h) = 0$, $\forall h \geq 2$. D'où :

$$\gamma(h) = \begin{cases} 2\sigma_\varepsilon^2 & \text{si} \quad h = 0 \\ -\sigma_\varepsilon^2 & \text{si} \quad h = 1 \\ 0 & \text{si} h \geq 2 \end{cases} \tag{8.23}$$

Exemple 8.4

Considérons le processus moyenne mobile infinie définit comme suit :

$$X_t = \sum_{i \in \mathbb{Z}} \theta_i \varepsilon_{t-i} \tag{8.24}$$

où $(\varepsilon_t)_{t\in\mathbb{Z}}$ est un bruit blanc de variance σ_ε^2. Alors sa fonction d'autocovariance d'ordre $h \in \mathbb{Z}$ est donnée par :

$$\gamma(h) = \sigma_\varepsilon^2 \sum_{i \in \mathbb{Z}} \theta_i \theta_{i+|h|} \tag{8.25}$$

et $\mathrm{Var}(X_0) = \sigma_\varepsilon^2 \sum_{i\in\mathbb{Z}} \theta_i^2$. Ainsi, si $\theta_i = 0$ lorsque $|i| > p$ (c'est le cas pour une moyenne mobile d'ordre $p+1$), on a : $\gamma(h) = 0$ lorsque $|h| > 2p$.

Exemple 8.5

Soit $(X_t)_{t\in\mathbb{Z}}$ un processus stationnaire tel que :

$$X_t = 0.4X_{t-1} + \varepsilon_t \tag{8.26}$$

où $\varepsilon_t \sim BB(0, \sigma_\varepsilon^2)$ avec $\sigma_\varepsilon^2 = 1$ et $\mathrm{Cov}(\varepsilon_s, X_{t-s}) = 0$, $s > 0$.

1. Calculer l'espérance et la variance de X_t
2. Calculer la fonction d'autocovariance $\gamma(h)\ \forall h$

Solution

On vérifie que le processus X_t est stationnaire. L'équation caractéristique associée $1-0.4z$ admet une racine $z = 2.5$ qui, en module, est strictement supérieure à 1. Donc X_t est stationnaire. Ainsi, l'espérance mathématique de X_t vaut :

$$\mathbb{E}(X_t) = 0.4\mathbb{E}(X_{t-1}) + \mathbb{E}(\varepsilon_t) \quad \Longrightarrow \quad (1-0.4)\mathbb{E}(X_t) = 0 \quad \Longrightarrow \quad \mathbb{E}(X_t) = 0 \tag{8.27}$$

et sa variance vaut :

$$\mathrm{V}(X_t) = (0.4)^2\mathrm{V}(X_{t-1}) + \mathrm{V}(\varepsilon_t) \quad \Longrightarrow \quad \mathrm{V}(X_t) = \frac{1}{1-(0.4)^2} = 1.19 \tag{8.28}$$

Par la suite, on suppose que $X_t = \phi_1 X_{t-1} + \varepsilon_t$ avec $\phi_1 = 0.4$. On recherche la fonction d'autocovariance $\gamma(h)$ pour $h > 1$.

$$\begin{aligned}
\gamma(1) &= \mathrm{Cov}(X_t, X_{t-1}) = \mathrm{Cov}(\phi_1 X_{t-1} + \varepsilon_t, X_{t-1}) \\
&= \phi_1 \underbrace{\mathrm{Cov}(X_{t-1}, X_{t-1})}_{=\gamma(0)} + \underbrace{\mathrm{Cov}(\varepsilon_t, X_{t-1})}_{=0} = \phi_1\gamma(0) \\
\gamma(2) &= \mathrm{Cov}(X_t, X_{t-2}) = \mathrm{Cov}(\phi_1 X_{t-1} + \varepsilon_t, X_{t-2}) \\
&= \phi_1 \underbrace{\mathrm{Cov}(X_{t-1}, X_{t-2})}_{=\gamma(1)} + \underbrace{\mathrm{Cov}(\varepsilon_t, X_{t-2})}_{=0} = \phi_1\gamma(1) = \phi_1^2\gamma(0)
\end{aligned}$$

On a donc : $\gamma(h) = \phi_1^h\gamma(0)$, $\forall h \geq 0$ avec $\gamma(0) = \mathrm{V}(X_t)$.

Exemple 8.6 *Fonction d'autocovariance*

On cherche à calculer l'autocovariance du processus linéaire suivant :

$$X_t = 0.75X_{t-1} - 0.25X_{t-2} + \varepsilon_t + 0.65\varepsilon_{t-1} + 0.35\varepsilon_{t-2} \tag{8.29}$$

```
# Autocovariance d'ordre h
import numpy as np
from statsmodels.tsa.arima_process import arma_acovf
arparams = np.array([.75, -.25])
maparams = np.array([.65, .35])
ar = np.r_[1, -arparams] # add zero-lag and negate
ma = np.r_[1, maparams] # add zero-lag
acovf = arma_acovf(ar,ma)
print(acovf.round(4))
```

```
## [ 4.5633e+00  3.6500e+00  1.9467e+00  5.4750e-01 -7.6000e-02 -1.9390e-01
##  -1.2640e-01 -4.6300e-02 -3.1000e-03  9.2000e-03]
```

8.2.2.2 Fonction d'autocorrélation

On étudie la mémoire d'un processus en calculant son autocorrélation de retard h (ou d'ordre h). C'est un concept lié à celui de corrélation.

Définition 8.10 *Fonction d'autocorrélation*

Soit $(X_t)_{t\in\mathbb{Z}}$ une série chronologique stationnaire. Sa fonction d'autocorrélation, notée $\rho(.)$, est définie, pour tout $h \in \mathbb{Z}$ par

$$\rho(h) = \text{Corr}(X_h, X_0) = \frac{\gamma(h)}{\gamma(0)} \tag{8.30}$$

Si on fait l'hypothèse que (X_t) est stationnaire, alors il est clair que pour tout $t \in T$, on a

$$\text{Corr}(X_{t+h}, X_t) = \cdots = \text{Corr}(X_{h+1}, X_1) = \text{Corr}(X_h, X_0) = \rho(h) \tag{8.31}$$

$\rho(h)$ mesure la corrélation de la série avec elle-même décalée de h périodes. Sa formule est donnée par :

$$\rho(h) = \frac{\sum_{t=h+1}^{t=T}(X_t - \overline{X}_1)(X_{t-h} - \overline{X}_2)}{\sqrt{\sum_{t=h+1}^{t=T}(X_t - \overline{X}_1)^2}\sqrt{\sum_{t=h+1}^{t=T}(X_{t-h} - \overline{X}_2)^2}} \tag{8.32}$$

avec $\overline{X}_1 = \dfrac{1}{T-h}\sum_{t=h+1}^{t=T} X_t$, $\overline{X}_2 = \dfrac{1}{T-h}\sum_{t=h+1}^{t=T} X_{t-h}$ et T le nombre d'observations.

$\rho(h)$ est le coefficient d'autocorrélation d'ordre h. Les coefficients d'autocorrélation sont calculés pour des ordre allant de 0 à H, H étant le décalage maximum admissible. En général, on a :

$$\frac{T}{6} < H < \frac{T}{3} \tag{8.33}$$

avec T le nombre d'observations. Si $T \geq 150$, on prendra $H = \dfrac{T}{5}$ pour que le coefficient d'autocorrélation ait un sens.

Si T est suffisamment grand par rapport à h, la formule donnée par l'équation (8.32) est difficile à manier (puisqu'elle exige de recalculer pour chaque terme $\rho(h)$ les moyennes et les variances) et peut être approximer par la fonction d'autocorrélation empirique d'ordre h. Un estimateur de cette fonction d'autocorrélation est la fonction d'autocorrélation empirique, noté $\widehat{\rho}(h)$, définie par

$$\widehat{\rho}(h) = \frac{\sum_{t=h+1}^{t=T}(X_t - \overline{X})(X_{t-h} - \overline{X})}{\sum_{t=1}^{t=T}(X_t - \overline{X})^2} \tag{8.34}$$

où $\overline{X}$ est la moyenne de la série calculée sur l'ensemble des T périodes ou observations.

Remarque 8.5

La fonction d'autocorrélation d'un processus $(X_t)_t$ stationnaire vérifie les propriétés suivantes :

1. $\rho(0) = 1$
2. $|\rho(h)| \leq \rho(0)$

3. $\rho(h) = \rho(-h)$: fonction paire
4. Elle décroît exponentiellement vers zéro.

Le test de signification sur le coefficient $\rho(h)$ permet de sélectionner les coefficients d'autocorrélation significativement différents de 0 ; il peut, si $T-h$ est suffisamment grand, s'effectuer comme pour un coefficient de corrélation linéaire simple. Soit $\rho(h)$ la valeur vraie de $\hat{\rho}(h)$ et l'hypothèse nulle $H_0 : \rho(h) = 0$ contre l'hypothèse alternative $H_1 : \rho(h) \neq 0$. Sous l'hypothèse nulle H_0, la statistique

$$t_c = \sqrt{T-h-2}\frac{|\hat{\rho}(h)|}{\sqrt{1-\hat{\rho}^2(h)}} \tag{8.35}$$

obéit à une loi de Student à $T-h-2$ degrés de liberté (ou à une loi normale centrée réduite si $T-h > 30$) dans lequel $T-h$ est le nombre d'observations servant à calculer le coefficient d'autocorrélation.

Si $t_c > t_{T-h-2}^{1-\alpha/2}$, l'hypothèse nulle H_0 est rejetée ($t_{T-h-2}^{1-\alpha/2}$ est la valeur de la loi de Student au seuil α à $T-h-2$ degrés de liberté).

Exemple 8.7 *Fonction d'autocorrélation du processus* $X_t = \varepsilon_t - \varepsilon_{t-1}$

On en déduit la fonction d'autocorrélation de X_t :

$$\rho(h) = \begin{cases} 1 & \text{si} \quad h = 0 \\ -\dfrac{1}{2} & \text{si} \quad h = 1 \\ 0 & \text{si} \quad h \geq 2 \end{cases} \tag{8.36}$$

Exemple 8.8 *Fonction d'autocorrélation du processus* $X_t = \phi_1 X_{t-1} + \varepsilon_t$ *avec* $\phi_1 = 0.4$ *et* $\varepsilon_t \sim \mathcal{N}(0,1)$

On en déduit la fonction d'autocorrélation de X_t :

$$\rho(h) = \phi_1^h \quad \forall h \geq 0 \tag{8.37}$$

Exemple 8.9 *Calcul d'une fonction d'autocorrélation*

A partir des données relatives aux ventes du tableau (8.1). On demande de calculer la fonction d'autocorrélation et de tester la significativité des coefficients d'autocorrélation par rapport à 0.

Table 8.1 – Exemple de calcul pour un coefficient d'autocorrélation d'ordre 2

Année	Saison	t	X_t	X_{t-2}	$(X_t-\overline{X}_1)(X_{t-2}-\overline{X}_2)$	$(X_t-\overline{X}_1)^2$	$(X_{t-2}-\overline{X}_2)^2$
Année 1	T1	1	1 248				
	T2	2	1 392				
	T3	3	1 057	1 248	195 360.56	431 162.99	88 518.15
	T4	4	3 159	1 392	–222 259.17	209 0540.47	23 629.84
Année 2	T1	5	891	1 057	402 349.24	676 391.15	239 336.23
	T2	6	1 065	3 159	–1 045 292.60	419 813.28	2 602 672.82
	T3	7	1 118	891	390 149.56	354 775.05	429 051.24
	T4	8	2 934	1 065	–586 700.47	1 490 767.59	230 899.47
Année 3	T1	9	1 138	1 118	246 239.35	330 659.57	183 372.34
	T2	10	1 456	2 934	–357 132.95	66 167.26	1 927 598.84
	T3	11	1 224	1 138	199 297.67	239 052.53	166 154.12
	T4	12	3 090	1 456	–123 679.42	1 896 046.27	8 067.63

Nous avons les sommes suivantes : $\sum_{t=3}^{t=12} X_t = 17132$, $\sum_{t=3}^{t=12} X_{t-2} = 15458$, $\sum_{t=3}^{t=12}(X_t-\overline{X}_1)(X_{t-2}-\overline{X}_2) = -901668.23$, $\sum_{t=3}^{t=12}(X_t-\overline{X}_1)^2 = 7995376.17$ et $\sum_{t=3}^{t=12}(X_{t-2}-\overline{X}_2)^2 = 5899300.68$.

Ainsi, $\overline{X}_1 = \frac{1}{12-2}\sum_{t=3}^{t=12} X_t = \frac{17132}{10} = 1713.2$ et $\overline{X}_1 = \frac{1}{12-2}\sum_{t=3}^{t=12} X_{t-2} = \frac{15458}{10} = 1545.8$.

D'où : $\hat{\rho}(2) = \dfrac{\sum_{t=3}^{t=12}(X_t-\overline{X}_1)(X_{t-2}-\overline{X}_2)}{\sqrt{\sum_{t=3}^{t=12}(X_t-\overline{X}_1)^2}\sqrt{\sum_{t=3}^{t=12}(X_{t-2}-\overline{X}_2)^2}} = \dfrac{-901668.23}{\sqrt{7995376.17}\sqrt{5899300.68}} = -0.13.$

Le t empirique est égal pour $T-2=10$ et $\hat{\rho}(2) = -0.13$: $t_c = \sqrt{12-2-2}\dfrac{|\hat{\rho}(2)|}{\sqrt{1-\hat{\rho}^2(2)}} = 0.38$.

Le coefficient n'est pas significativement différent de 0.

Le tableau (8.2) indique l'ensemble des valeurs de la fonction d'autocorrélation que l'on compare à la valeur lue dans la table de Student pour un seuil de 5% et à $T-h-2$ degrés de liberté. Seul le coefficient d'autocorrélation d'ordre 4 est significativement différent de 0 ; la périodicité des données étant trimestrielle, ce « pic » est donc attribué à la saisonnalité des données.

Table 8.2 – Calcul d'une fonction d'autocorrélation

h	$\hat{\rho}(h)$	$T-h$	$\lvert t_c\rvert$	$ddl = T-h-2$	t lu à 0.95
0	1	12			
1	–0.395	11	1.29	9	2.262
2	–0.132	10	0.38	8	2.306
3	–0.33	9	0.95	7	2.365
4	0.952	8	7.62	6	2.447

La fonction armar_acf() permet de calculer les autocorrélations d'ordre h pour un processus linéaire.

```
# Autocorrélation
from statsmodels.tsa.arima_process import arma_acf
acf = arma_acf(ar,ma,lags=h)
print(acf)
```

où `ar` et `ma` sont respectivement les coefficients autorégressifs et moyennes mobiles (y compris la constante).

Si on dispose d'une série chronologique $X_t, t = 1, \ldots, T$, alors on estime la fonction d'autocorrélation à l'aide de la fonction acf() de Statsmodels :

```
# Autocorrélation
import pandas as pd
from statsmodels.tsa.stattools import acf
vente = np.array([1248,1392,1057,3159,891,1065,1118,2934,1138,1456,1224,3090])
acf = acf(vente,nlags=4,adjusted=True,alpha=0.05)
acf_df = pd.DataFrame(np.append(np.c_[acf[0]],acf[1],axis=1),index = range(5),
            columns = ["rho","lower","upper"]).rename_axis('h').reset_index()
```

Table 8.3 – Calcul d'une fonction d'autocorrélation avec intervalle de confiance

h	$\hat{\rho}(h)$	0.025	0.975
0	1.000	1.000	1.000
1	-0.370	-0.936	0.196
2	-0.141	-0.779	0.498
3	-0.439	-1.088	0.209
4	0.960	0.222	1.697

Nos résultats sont légèrement différents de ceux obtenus par calcul manuel. Cependant, l'ordre de variation entre les fonctions d'autocorrélation ne change pas.

L'intérêt pratique de la fonction d'autocorrélation se trouve notamment dans l'étude des processus ARMA. Une autre fonction fondamentale dans l'étude des séries temporelles est la fonction d'autocorrélation partielle.

8.2.2.3 Fonction d'autocorrélation partielle

La fonction d'autocorrélation partielle mesure la corrélation entre X_t et X_{t-h} l'influence des variables X_{t-h+i} (pour $i < h$) ayant été retirée. Notons par $\rho(h)$ et ϕ_{hh} les fonctions respectivement d'autocorrélation et d'autocorrélation partielle de X_t. Soit $P(h)$ la matrice symétrique formée des $(h-1)$ premières autocorrélations de X_t :

$$P(h) = \begin{pmatrix} 1 & \rho(1) & \cdots & \rho(h-1) \\ & 1 & & \vdots \\ & & \ddots & \\ \rho(h-1) & & & 1 \end{pmatrix} \tag{8.38}$$

La fonction d'autocorrélation partielle est donnée par :

$$\phi_{hh} = \frac{|P^*(h)|}{|P(h)|} \tag{8.39}$$

où $|P(h)|$ est le déterminant de la matrice $P(h)$ et $|P^*(h)|$ est donnée par

$$P^*(h) = \begin{pmatrix} 1 & \rho(1) & \cdots & \rho(h-2) & \rho(1) \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ \rho(h-1) & & & & \rho(h) \end{pmatrix} \tag{8.40}$$

En d'autres termes, $P^*(h)$ est la matrice $P(h)$ dans laquelle on a remplacé la dernière colonne par le vecteur $[\rho(1), \dots, \rho(h)]'$.

Par exemple, calculons ϕ_{11}, ϕ_{22} et ϕ_{33}.

$$\begin{aligned} \phi_{11} &= \frac{|P^*(1)|}{|P(1)|}, \quad \text{avec} \quad P(1) = [1] \quad \text{et} \quad P^*(1) = [\rho(1)] \\ &= \frac{\rho(1)}{1} = \rho(1) \end{aligned}$$

On constate que la première valeur de l'autocorrélation partielle est égale à la première valeur de l'autocorrélation simple.

$$\begin{aligned} \phi_{22} &= \frac{|P^*(2)|}{|P(2)|}, \quad \text{avec} \quad P(2) = \begin{pmatrix} 1 & \rho(1) \\ \rho(1) & 1 \end{pmatrix} \quad \text{et} \quad P^*(2) = \begin{pmatrix} 1 & \rho(1) \\ \rho(1) & \rho(2) \end{pmatrix} \\ &= \frac{\rho(2) - \rho(1)^2}{1 - \rho(1)^2} \end{aligned}$$

et

$$\begin{aligned} \phi_{33} &= \frac{|P^*(3)|}{|P(3)|}, \quad \text{avec} \quad P(3) = \begin{pmatrix} 1 & \rho(1) & \rho(2) \\ \rho(1) & 1 & \rho(1) \\ \rho(2) & \rho(1) & 1 \end{pmatrix} \quad \text{et} \quad P^*(3) = \begin{pmatrix} 1 & \rho(1) & \rho(1) \\ \rho(1) & 1 & \rho(2) \\ \rho(2) & \rho(1) & \rho(3) \end{pmatrix} \\ &= \frac{\rho(1)^3 - \rho(1)\rho(2)\left(2 - \rho(2)\right) + \rho(3)\left(1 - \rho(1)^2\right)}{1 - \rho(2)^2 - 2\rho(1)^2\left(1 - \rho(2)\right)} \end{aligned}$$

On peut alors montrer que la fonction d'autocorrélation partielle s'écrit :

$$\phi_{ii} = \begin{cases} \rho(1) & \text{si } i = 1 \\ \dfrac{\rho(i) - \displaystyle\sum_{j=1}^{i-1} \phi_{i-1,j}\rho(i-j)}{1 - \displaystyle\sum_{j=1}^{i-1} \phi_{i-1,j}\rho(j)} & \text{pour } i = 2, \dots, k \end{cases} \tag{8.41}$$

et $\phi_{ij} = \phi_{i-1,j} - \phi_{ii}\phi_{i-1,i-j}$ pour $i = 2, \dots, k$ et $j = 1, \dots, i-1$.

Cet algorithme est connu sous le nom d'algorithme de Durbin (1960). Il est basé sur les équations de Yule - Walker. On constate que les coefficients d'autocorrélation partielle sont donnés par les coefficients d'autocorrélation simple et par un ensemble d'équations récursives.

Sous Python, la fonction arma_pacf de Statsmodels permet de calculer les autocorrélations partielles d'un processus linéaire.

```
# Autocorrélation partielle
from statsmodels.tsa.arima_process import arma_pacf
pacf = arma_pacf(ar, ma)
print(pacf)
```

où `ar` et `ma` sont respectivement les coefficients autorégressifs et moyennes mobiles (y compris la constante).

Exemple 8.10 *Autocorrélation partielle de la série des ventes*

Si on dispose d'une série chronologique $X_t, t = 1, \dots, T$, alors on estime la fonction d'autocorrélation partielle grâce à la fonction pacf() de Statsmodels :

```
# Autocorrélation partielle
from statsmodels.tsa.stattools import pacf
pacf = pacf(vente,nlags=4, method='ywadjusted', alpha=0.05)
pacf_df = pd.DataFrame(np.append(np.c_[pacf[0]],pacf[1],axis=1),index = range(5),
            columns = ["phi","lower","upper"]).rename_axis('h').reset_index()
```

Table 8.4 – Calcul d'une fonction d'autocorrélation partielle avec intervalle de confiance

h	$\widehat{\phi}_{hh}$	0.025	0.975
0	1.000	1.000	1.000
1	-0.370	-0.936	0.196
2	-0.322	-0.888	0.244
3	-0.811	-1.377	-0.245
4	0.863	0.298	1.429

Exemple 8.11

On calcule les autocorrélations partielles du processus définie par l'équation (9.59).

```
# Autocorrélation partielle d'ordre h
from statsmodels.tsa.arima_process import arma_pacf
pacf = arma_pacf(ar, ma)
print(pacf.round(3))
```

```
## [ 1.     0.8   -0.592  0.214  0.057 -0.11   0.052  0.005 -0.021  0.012]
```

Remarque 8.6

Dans certains ouvrages, le calcul de la fonction d'autocorrélation partielle fait appel à la notion de projection sur un sous - espace vectoriel. En effet, l'idée de la corrélation partielle est de mesurer le degré de liaison entre deux variables en neutralisant (ou en contrôlant) les effets d'une ou d'autres variables.

Par exemple, si on considère le processus $(X_t)_{t\in\mathbb{Z}}$ aux instants $t-h, \dots, t$, alors la fonction d'autocorrélation partielle mesure la corrélation entre X_t et X_{t-h}, l'influence des variables supplémentaires étant supposée retirée, c'est - à - dire lorsqu'on a éliminé les parties de X_t et X_{t-h} expliquée par les variables intermédiaires (c'est - à - dire les variables X_{t-h+i} telle que $(i < h)$).

Définition 8.11 *Fonction d'autocorrélation partielle*

La fonction d'autocorrélation partielle au retard h, notée $r(h) = \phi_{hh}$, est donnée, pour tout $t \in \mathbb{Z}$ et $h \in \mathbb{Z}$, par :

$$\begin{aligned} r(h) &= \text{Corr}\left(X_t - \widetilde{X}_t, X_{t-h} - \widetilde{X}_{t-h}\right) \\ &= \frac{\text{Cov}\left(X_t - \tilde{X}_t, X_{t-h} - \widetilde{X}_{t-h}\right)}{\sqrt{\text{Var}\left(X_t - \widetilde{X}_t\right)}\sqrt{\text{Var}\left(X_{t-h} - \widetilde{X}_{t-h}\right)}} \end{aligned}$$

avec $r(1) = \rho(1)$, et où, pour tout t, $\widetilde{X}_t$ est la régression affine de X_t sur $X_{t-1}, X_{t-2}, \dots, X_{t-h+1}$ ou projection orthogonale de X_t sur le sous - espace vectoriel de $\mathbb{L}^2$ engendré par $X_{t-1}, X_{t-2}, \dots, X_{t-h+1}$ et $\widetilde{X}_{t-h}$ est la régression affine de X_{t-h} sur $X_{t-h+1}, X_{t-h+2}, \dots, X_{t-1}$:

$$\widetilde{X}_t = \beta_1 X_{t-1} + \beta_2 X_{t-2} + \cdots + \beta_{h-1} X_{t-h+1} \tag{8.42}$$

$$\widetilde{X}_{t-h} = \beta_{h-1} X_{t-h+1} + \beta_{h-2} X_{t-h+2} + \cdots + \beta_1 X_{t-1} \tag{8.43}$$

Dans les deux équations, les coefficients $\beta_1, \dots, \beta_{h-1}$ sont identiques. On a :

$$X_t = \widetilde{X}_t + Y_{t,h} = \phi_{0,h} + \sum_{j=1}^{j=h} \phi_{j,h} X_{t-j} + Y_{t,h} \tag{8.44}$$

où Y_t est une variable aléatoire non corrélée avec $X_{t-1}, \dots, X_{t-h}$.

Définition 8.12 *Fonction d'autocorrélation partielle*

On appelle fonction d'autocorrélation partielle d'un processus stationnaire $(X_t)_{t\in\mathbb{Z}}$, la fonction $r(.)$ définie, pour tout $h \in \mathbb{Z}^*$, par :

$$r(h) = \phi_{hh} \quad \text{et} \quad r(0) = 1 \tag{8.45}$$

Pour un retard h fixé, le nombre $r(h)$ est alors le coefficient de corrélation linéaire entre la variable $X_t - \mathbb{E}\left(X_t | X_{t-1}, \dots, X_{t-h+1}\right)$ et la variable $X_{t-h} - \mathbb{E}\left(X_{t-h} | X_{t-h+1}, \dots, X_{t-1}\right)$. Par construction, on a : $r(1) = \rho(1)$ et est également à valeurs dans $[-1, 1]$.

Proposition 8.1

$r(h)$ est égale au coefficient ϕ_{hh} que l'on applique à X_{t-h} dans l'équation de régression de X_t sur $X_{t-1}, \dots, X_{t-h}$

$$\widehat{X}_t = \sum_{j=1}^{j=h} \phi_{h,j} X_{t-j} \tag{8.46}$$

équivalente à la prédiction linéaire à un pas à erreur quadratique minimale.

ϕ_{hh} représente l'apport d'explication de X_{t-h} à X_t étant donné qu'on a déja regressé sur $X_{t-1}, \dots, X_{t-h+1}$.

Theorème 8.2 *Soit $(X_t)_{t\in\mathbb{Z}}$ un processus stationnaire. On considère la régression linéaire de X_t sur $X_{t-1},\dots,X_{t-h}$:*

$$\mathbb{EL}(X_t|X_{t-1},\dots,X_{t-h}) = \sum_{j=1}^{j=h} \phi_{h,j} X_{t-j} \tag{8.47}$$

et

$$\varepsilon_t = X_t - \mathbb{EL}(X_t|X_{t-1},\dots,X_{t-h}) \tag{8.48}$$

Alors, on a :

- $\mathbb{E}(\varepsilon_t) = 0$ *et il existe* $\sigma_\varepsilon > 0$ *tel que pour tout* $t \in \mathbb{Z}$, $\mathbb{E}(\varepsilon_t) = \sigma_\varepsilon^2$
- $\forall j \in \{1,\dots,h\}$, $\mathbb{E}(\varepsilon_t X_{t-j}) = 0$

Et on a $\phi_{hh} = r(h)$ *ainsi que*

$$\begin{pmatrix} \rho(1) \\ \rho(2) \\ \vdots \\ \rho(h) \end{pmatrix} = \mathcal{R}(h) \begin{pmatrix} \phi_{h1} \\ \phi_{h2} \\ \vdots \\ \phi_{hh} \end{pmatrix} \tag{8.49}$$

où $\mathcal{R}(h)$ est la matrice d'autocorrélation simple de $(X_t,\dots,X_{t-h+1})$, $h \in \mathbb{N}^*$ et définie comme suit :

$$\mathcal{R}(h) = \begin{pmatrix} 1 & \rho(1) & \cdots & \rho(h-1) \\ \rho(1) & 1 & \cdots & \rho(h-2) \\ \vdots & \vdots & \ddots & \vdots \\ \rho(h-1) & \rho(h-2) & \cdots & 1 \end{pmatrix} \tag{8.50}$$

En inversant la matrice $\mathcal{R}(h)$, on estime alors les $\phi_{h,i}, i = 1,\dots,h$.

Ces coefficients peuvent également être estimés en se servant de la méthode des moindres carrés ordinaires.

Remarque 8.7

Le coefficient d'autocorrélation partielle d'ordre h traduit une corrélation conditionelle entre la série initiale (X_t) et celle retardée de h relevés, soit (X_{t-h}). Le conditionnement correspond à la corrélation non expliquée par les valeurs intermédiaires $(X_{t-1},\dots,X_{t-h-1})$.

8.2.3 Autres concepts généraux

8.2.3.1 Bruit blanc

Le processus le plus simple mais véritablement atome de base de tous nos modèles est le processus dit « bruit blanc » (ou white noise).

Définition 8.13 *Bruit blanc faible*

Un processus $(\varepsilon_t)_{t\in\mathbb{Z}}$ est dit de bruit blanc (faible) noté BB si :

1. $\mathbb{E}(\varepsilon_t) = 0 \ \forall t$
2. $\mathrm{V}(\varepsilon_t) < \infty, \ \forall t$ (ou $\mathrm{V}(\varepsilon_t) = \sigma_\varepsilon^2 \ \forall t$)
3. $\mathrm{Cov}(\varepsilon_t, \varepsilon_{t+h}) = 0 \ \forall h \geq 1$

Un bruit blanc est ainsi un processus de moyenne nulle, de variance constante et non autocorrélé. On note :

$$\varepsilon_t \sim BB(0, \sigma_\varepsilon^2) \tag{8.51}$$

Définition 8.14 *Bruit blanc fort*

Un bruit blanc est dit fort lorsque de plus les variables ε_t sont indépendantes et que les moments d'ordre 2 existent.

Définition 8.15 *Bruit blanc gaussien*

On appelle bruit blanc gaussien tout bruit blanc fort $(\varepsilon_t)_{t\in\mathbb{Z}}$ tel que

$$\forall t, \quad \varepsilon_t \sim \mathcal{N}(0, \sigma_\varepsilon^2) \tag{8.52}$$

Remarque 8.8

— Un bruit blanc n'est pas nécessairement gaussien.
— Si $(\varepsilon_t)_{t\in\mathbb{Z}}$ est un bruit blanc gaussien, alors les variables ε_t sont indépendantes.
— Le bruit blanc $(\varepsilon_t)_{t\in\mathbb{Z}}$ est un processus sans mémoire car la variable aléatoire ε_t est non corrélée aux variables aléatoires précédentes $\varepsilon_{t-1}, \varepsilon_{t-2}, \cdots$

Sous Python, la trajectoire du bruit blanc $BBG(0,1)$ et celle du bruit blanc uniforme $BBU_{[-1,1]}$, de longueur T sont simulées par les commandes suivantes :

```
# Bruit blanc gaussien et bruit blanc uniforme
np.random.seed(123)
bbnorm = np.random.normal(loc=0,scale=1,size=1000)
bbunif = np.random.uniform(low=-1,high=1,size=1000)
# Représentation graphique
fig, (axe1, axe2) = plt.subplots(1,2,figsize=(16,6))
axe1.plot(bbnorm, color = "black");
axe1.set_title("a) Bruit blanc gaussien");
axe2.plot(bbunif, color = "black");
axe2.set_title("b) Bruit blanc uniforme");
plt.tight_layout();
plt.show()
```

8.2.3.2 Opérateur retard

L'opérateur retard décale le processus d'une unité de temps vers le passé. On le note souvent L (Lag) ou B (Backward) suivant les ouvrages.

Définition 8.16 *Opérateur retard*

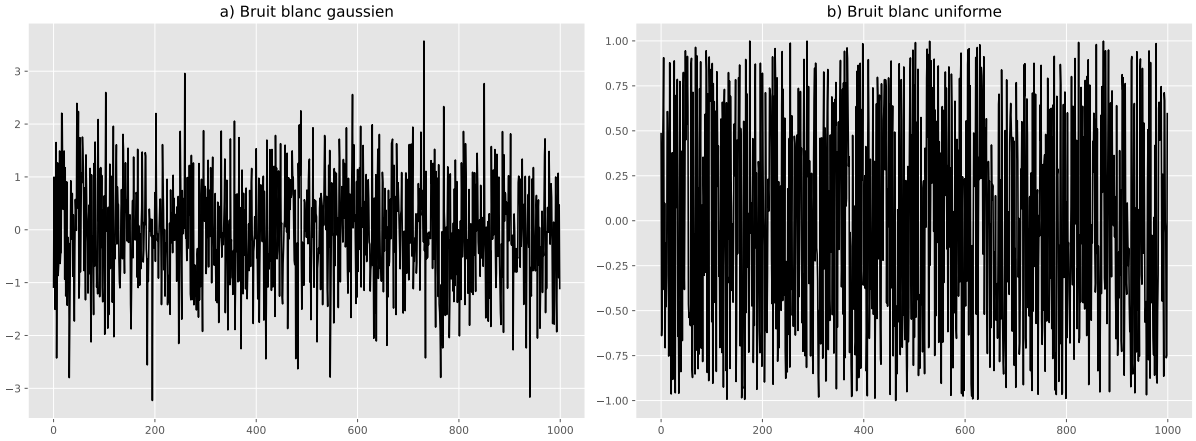


Figure 8.1 – Simulation d'un bruit blanc gaussien et d'un bruit blanc uniforme

On considère un processus stochastique $(X_t)_{t\in\mathbb{Z}}$, l'opérateur retard noté B est défini par la relation :

$$BX_t = X_{t-1} \tag{8.53}$$

Récursivement, en composant les opérateurs 2 fois, $B^2 = B \circ B$, on obtient l'avant dernière observation avant t, soit :

$$B^2X_t = B \circ BX_t = BX_{t-1} = X_{t-2} \tag{8.54}$$

Plus généralement, on a :

$$B^nX_t = X_{t-n}, \quad \forall n \in \mathbb{N} \tag{8.55}$$

On constate ainsi que l'opérateur retard transforme une variable X_t en sa valeur passée. L'opérateur retard est linéaire et inversible. Son inverse $B^{-1} = F$ est défini par :

$$\begin{cases} B^{-1}X_t = FX_t = X_{t+1} \\ F^nX_t = X_{t+n}, \quad \forall n \in \mathbb{N} \end{cases} \tag{8.56}$$

Définition 8.17 *Opérateur avance*

L'opérateur F (Forward) est appelé opérateur avance. On a :

$$\forall t \in \mathbb{Z}, \quad \begin{cases} FBX_t = B\left(FX_t\right) = BX_{t+1} = X_t \\ BFX_t = F\left(BX_t\right) = FX_{t-1} = X_t \end{cases} \tag{8.57}$$

Par conséquent, F et B sont des opérateurs inverses l'un de l'autre pour la composition :

$$\begin{cases} F = B^{-1} \Longleftrightarrow B = F^{-1} \\ B \circ F = F \circ B = \mathbb{I} : \text{Opérateur identité} \end{cases} \tag{8.58}$$

Si l'on applique le polynôme retard $\Phi(B)$ défini comme suit :

$$\Phi(B) = I - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p \tag{8.59}$$

à une série X_t, on a :

$$\Phi(B)X_t = X_t - \phi_1 X_{t-1} - \phi_2 X_{t-2} - \cdots - \phi_p X_{t-p} \tag{8.60}$$

où $\phi_1, \ldots, \phi_p$ sont des coefficients.

Propriété 8.5

L'opérateur retard vérifie les propriétés suivantes :

1. $B^n X_t = X_{t-n} \ \forall n \in \mathbb{Z}$
2. $B^n X_t = B^{n-1} B X_t = B^{n-1} X_{t-1}$
3. $B^{-n} = X_{t+n}$
4. $\left(B^n + B^m\right) X_t = B^n X_t + B^m X_t = X_{t-n} + X_{t-m} \ \forall (n,m) \in \mathbb{Z}^2$
5. Si $X_t = c \ \forall t \in \mathbb{Z}$ avec $c \in \mathbb{R}$ alors $B^n X_t = B^n c = c \ \forall n \in \mathbb{Z}$
6. $B^n \left(B^m\right) X_t = B^{n+m} X_t = X_{t-n-m} \ \forall (n,m) \in \mathbb{Z}^2$
7. $\left(\sum_{i=0}^{\infty} \alpha_i B^i\right) X_t = \sum_{i=0}^{\infty} \alpha_i X_{t-i}$
8. $\sum_{i=-\infty}^{\infty} \alpha_i B^i X_t + \sum_{i=-\infty}^{\infty} \beta_i B^i X_t = \sum_{i=-\infty}^{\infty} \left(\alpha_i + \beta_i\right) B^i X_t = \sum_{i=-\infty}^{\infty} \left(\alpha_i + \beta_i\right) X_{t-i}$
9. $a \sum_{i=-\infty}^{\infty} \alpha_i B^i = \sum_{i=-\infty}^{\infty} a\alpha_i B^i$

Toutes les opérations usuelles sur les séries entières, telles que l'addition, la multiplication, la division et l'inversion, sont également applicables sur les polynômes retard.

Exemple 8.12 *Illustration de l'opération d'inversion*

Considérons le polynôme de degré 1 suivant :

$$\Phi(B) = I - \phi_1 B, \quad \phi_1 \in \mathbb{R} \tag{8.61}$$

Ce polynôme est inversible si et seulement si $|\phi_1| \neq 1$. Plus précisément [1] :

1. Si $|\phi_1| < 1$, on a :

$$\left(I - \phi_1 B\right)^{-1} = \sum_{i=0}^{\infty} \phi_1^i B^i \tag{8.62}$$

2. Si $|\phi_1| > 1$, on a :

$$\left(I - \phi_1 B\right)^{-1} = -\sum_{i=1}^{\infty} \frac{1}{\phi_1^i} F^i = -\sum_{i=-1}^{-\infty} \phi_1^i B^i \tag{8.63}$$

1. Pour plus de détails, voir Gouriéroux et Monfort (1995).

En effet, pour le cas $|\phi_1| > 1$, on réécrit que $(I - \phi_1 B) = (-\phi_1 B)\left(I - \frac{1}{\phi_1}B^{-1}\right)$.

Exemple 8.13

Soit $\Phi(B) = I - 2B$. Le polynôme Φ admet $1/2$ comme unique racine, qui est de module strictement plus petit que 1. On a :

$$\Phi(B) = -2B\left(I - \frac{1}{2}B^{-1}\right) \tag{8.64}$$

Comme $1/2$ est bien de module plus petit que 1, on en déduit que :

$$(I - 2B)^{-1} = (-2B)^{-1}\sum_{j=0}^{+\infty}\left(\frac{1}{2}\right)^j B^{-j} = -\sum_{j=1}^{+\infty}\left(\frac{1}{2}\right)^j B^{-j} \tag{8.65}$$

Plus généralement, considérons un polynôme retard de degré p :

$$\Phi(B) = I - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p \tag{8.66}$$

et l'équation caractéristique associée :

$$1 - \phi_1 z - \phi_2 z^2 - \cdots - \phi_p z^p = 0 \tag{8.67}$$

Supposons que les racines z_j de (8.67) soient de module supérieur à 1. Sachant qu'il existe une série entière :

$$\Theta(z) = \sum_{i=0}^{\infty}\theta_i z^i \tag{8.68}$$

telle que

$$\sum_{i=0}^{\infty}|\theta_i| < \infty \tag{8.69}$$

et

$$\Phi(z)\Theta(z) = 1 \tag{8.70}$$

on a donc :

$$\Phi(B)\Theta(B) = 1 \tag{8.71}$$

On en déduit que le polynôme $\Phi(B)$ est inversible et son inverse est égal à $\Theta(B)$. Plus précisément, en factorisant le polynôme $\Phi(z)$, on peut écrire :

$$\Phi(z) = \prod_{j=1}^{p}\left(z - z_j\right)\phi_p \tag{8.72}$$

soit encore[2] :

$$\Phi(z) = \prod_{j=1}^{p} \left(1 - \frac{z}{z_j}\right) \tag{8.73}$$

On peut alors écrire :

$$\Phi^{-1}(z) = \prod_{j=1}^{p} \left(1 - \frac{z}{z_j}\right)^{-1} \tag{8.74}$$

Sachant que :

$$\left(1 - \frac{z}{z_j}\right)^{-1} = \sum_{i=0}^{\infty} \left(\frac{1}{z_j}\right)^{i} z^i \tag{8.75}$$

On en déduit l'inverse du polynôme $\Phi(B)$:

$$\Phi^{-1}(B) = \prod_{j=1}^{p} \sum_{i=0}^{\infty} \left(\frac{1}{z_j}\right)^{i} B^i \tag{8.76}$$

Exemple 8.14

Soit $\Phi(B) = I - \frac{1}{6}B - \frac{1}{6}B^2$. On souhaite inverser Φ. L'équation caractéristique associée $\Phi(z) = 1 - \frac{1}{6}z - \frac{1}{6}z^2$ admet deux racines réelles distinctes $z_1 = -3$ et $z_2 = 2$ de module strictement plus grand que 1. La décomposition en éléments simples permet d'avoir l'inverse $\Phi^{-1}(B)$ suivant :

$$\begin{aligned}\Phi^{-1}(B) &= \frac{1}{\left(I - \frac{1}{2}B\right)\left(I + \frac{1}{3}B\right)} = \frac{3/5}{1 - \frac{1}{2}B} + \frac{2/5}{1 + \frac{1}{3}B} \\ &= \frac{3}{5}\sum_{j=0}^{+\infty}\left(\frac{1}{2}\right)^{j} B^j + \frac{2}{5}\sum_{j=0}^{+\infty}\left(-\frac{1}{3}\right)^{j} B^j\end{aligned}$$

Proposition 8.2

L'opérateur $\Phi(B)$ est inversible si et seulement si les racines de Φ sont de module différent de 1. On a alors :

$$\Phi^{-1}(B) = \sum_{j \in \mathbb{Z}} \psi_j B^j \tag{8.77}$$

Proposition 8.3

$\psi_j = 0$ pour tout $j < 0$ si et seulement si toutes les racines de Φ sont de module strictement plus grand que 1.

2. En remarquant que le produit des racines est égal à $1/\phi_p$ et en notant que $(z - z_j) = -z_j\left(1 - \frac{z}{z_j}\right)$.

8.2.3.3 Processus d'innovation

Le processus d'innovation $(\varepsilon_t)_{t\in\mathbb{Z}}$ d'un processus $(X_t)_{t\in\mathbb{Z}}$ est défini comme étant l'écart entre la variable X_t au temps t et sa projection sur l'espace vectoriel engendré par les variables jusqu'au temps $t-1$, i.e :

$$\varepsilon_t = X_t - \mathbb{E}\left(X_t|\mathcal{F}_{t-1}\right) \tag{8.78}$$

c'est-à-dire

$$\varepsilon_t = X_t - X_t^* \tag{8.79}$$

où X_t^* est la régression affine de X_t sur $(X_s, s \leq t)$. C'est la meilleure approximation affine de X_t fonction de son passé. L'innovation est alors la partie de X_t non corrélée au passé de la série :

$$X_t = d_t + \varepsilon_t \tag{8.80}$$

La partie aléatoire ε_t est la partie de X_t non corrélée avec $(X_{t-s}, s > 0)$. Cette partie représente l'innovation de X_t.

On montre que le processus d'innovation d'un processus stationnaire est un bruit blanc et qu'un processus bruit blanc est son propre processus d'innovation.

8.2.4 Outils d'analyse spectrale

8.2.4.1 Densité spectrale

Il y a au moins deux façons de voir un processus stationnaire au second ordre. La première est de travailler dans le domaine temporel en considérant les corrélations entre X_t et X_{t+h}. La deuxième approche est celle des fréquences : on considère les composantes périodiques (aléatoires) qui composent le processus (traitement d'un "signal"). Ces deux approches permettent d'obtenir les mêmes résultats mais il est parfois plus commode d'utiliser l'une ou l'autre.

La densité spectrale ou spectre est la répartition de la dispersion, plus précisément de la variance, en fonction des fréquences. Elle est l'analogue, dans le domaine des fréquences, de la fonction d'autocovariance dans le domaine temporel. L'intérêt de la densité spectrale est de mettre en évidence surtout les phénomènes cycliques dans les données. Pour les données saisonnières avec une saison s, la densité spectrale a un pic dominant à la fréquence de Fourier $\lambda_j = js/T$ où T est la taille de l'échantillon, le cycle correspondant est donné par cycle $= T/j$.

Exemple 8.15 *Cours de clôture des actions de GOOG*

Considérons la série goog disponible dans la librairie fpp2 de R. Ces données représentent les cours de clôture des actions de GOOG à la bourse NASDAQ, pendant 1000 jours de bourse consécutifs entre le 25 février 2013 et le 13 février 2017. Nous travaillons avec la version réduite `goog20` qui contient les 200 premières observations de `goog`.

```
# Chargement des données
import statsmodels.api as sm
goog200 = sm.datasets.get_rdataset("goog200","fpp2").data.drop(columns=["time"])
goog200.index = pd.date_range(start ="2013/02/25",periods=len(goog200),freq="B")
```

```
goog200 = goog200.rename(columns ={'value':"goog-close"})
# Représentation graphique
fig,(axe1, axe2) = plt.subplots(1,2,figsize=(16,6))
axe1.plot(goog200,color="black");
axe2.plot(goog200.diff().dropna(),color="black");
axe1.set_xlabel("jour");
axe1.set_ylabel("cours");
axe1.set_title("a) cours de clôture");
axe2.set_xlabel("jour");
axe2.set_ylabel("cours");
axe2.set_title("b) cours de côture (différence première)");
plt.tight_layout();
plt.show()
```

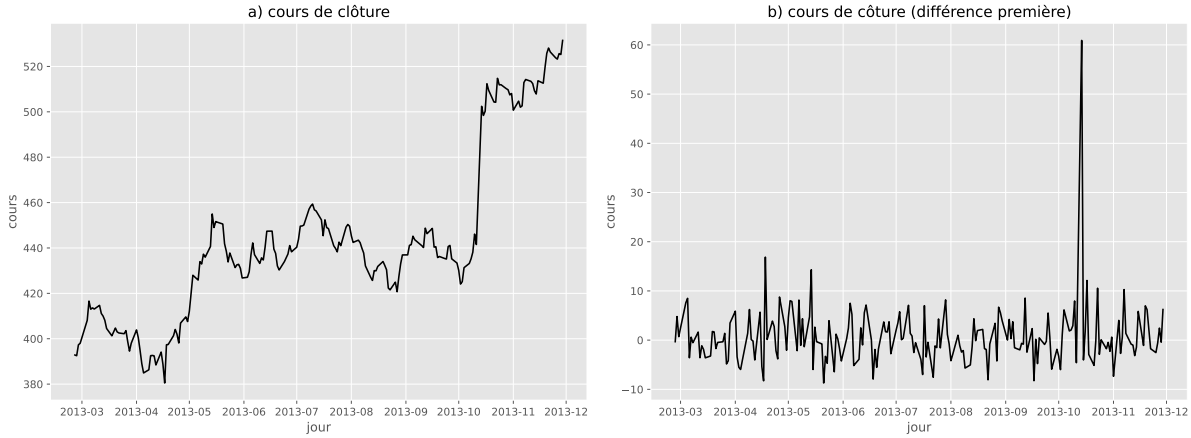
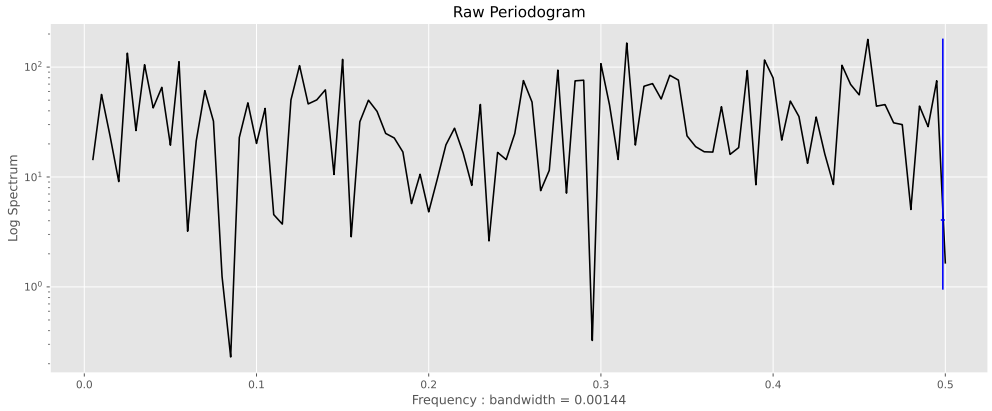


Figure 8.2 – Evolution des cours de clôtures des actions de GOOG à le bourse de NASDAQ.

La figure de gauche représente la série initiale et celle de droite la série en différence première. Au regard de leurs évolutions, la première semble non stationnaire, par conséquent, nous travaillerons avec la série en différence première.

Sous R, la fonction spec.pgram calcule le périodogramme à l'aide d'une transformée de Fourier rapide. Sous Python à ce jour, aucune librairie ne contient l'équivalent de cette fonction. Cependant, dans nos recherches, nous sommes tombés sur ce repository github https ://github.com/telmo-correa/time-series-analysis qui contient une version implémentée de la fonction.

```
from spectrum import *
results = spec_pgram(goog200.diff().dropna(),demean = False,detrend = True)
fig, axe = plt.subplots(figsize=(16,6))
plot_spec(results,coverage=0.95,ax=axe);
plt.show()
```



La densité spectrale permet aussi de mettre en évidence la nature de la mémoire des séries. En effet, la présence d'un pic quand les fréquences tendent vers 0 sera interprétée comme présence de longue mémoire.

Définition 8.18 *Densité spectrale*

Soit $(X_t)_{t\in\mathbb{Z}}$, un processus stationnaire, de fonction d'autocovariance $\gamma(.)$, la densité spectrale de $(X_t)_{t\in\mathbb{Z}}$ est la fonction $f(.)$ définie comme étant la transformée de Fourier de l fonction d'autocovariance du processus :

$$f(\lambda) = \frac{1}{2\pi} \sum_{h=-\infty}^{+\infty} \gamma(h) e^{-i\lambda h}, \quad \forall \lambda \in [-\pi, \pi] \quad h \in \mathbb{Z} \tag{8.81}$$

En utilisant la parité de la fonction d'autocovariance, on voit que la densité est une fonction réelle, et on a :

$$f(\lambda) = \frac{1}{2\pi} \sum_{h=-\infty}^{+\infty} \gamma(h) \cos(\lambda h), \quad \forall \lambda \in [-\pi, \pi] \quad h \in \mathbb{Z} \tag{8.82}$$

En pratique, on travaille plutôt sur la densité spectrale normée, qui est la transformée de Fourier de la fonction d'autocorrélation (théorique) :

$$f(\lambda) = \frac{1}{2\pi} \sum_{h=-\infty}^{+\infty} \rho(h) e^{-i\lambda h}, \quad \forall \lambda \in [-\pi, \pi] \quad h \in \mathbb{Z} \tag{8.83}$$

Exemple 8.16 *Densité spectrale d'un processus moyenne mobile infinie*

Considérons le processus moyenne mobile infinie défini par :

$$X_t = \sum_{i\in\mathbb{Z}} \theta_i \varepsilon_{t-i} \tag{8.84}$$

où $(\varepsilon_t)_{t\in\mathbb{Z}}$ est un bruit blanc de variance σ_ε^2. On a déja vu que :

$$\gamma(h) = \sigma_\varepsilon^2 \sum_{i\in\mathbb{Z}} \theta_i \theta_{i+h}. \tag{8.85}$$

Nous obtenons alors :

$$f(\lambda) = \frac{\sigma_\varepsilon^2}{2\pi}|\Theta(e^{i\lambda})|^2 \tag{8.86}$$

où $\Theta(z) = \sum_{i\in\mathbb{Z}} \theta_i z^i$, $z \in \mathbb{C}$.

Exemple 8.17

1. Pour un processus bruit blanc, on a : $f(\lambda) = \frac{\sigma_\varepsilon^2}{2\pi}$. Reciproquement, on pourra montrer que si $f(\lambda) = \frac{\sigma_\varepsilon^2}{2\pi}$ alors le processus est un bruit blanc de variance σ_ε^2 (en utilisant l'expression de la covariance en terme de transformée de Fourier).
2. Supposons que $X_t = \varepsilon_t + \theta\varepsilon_{t-1}$ où $|\theta| < 1$. D'après l'expression générale de la densité spectrale, il suffit de calculer :

   $$|1 + \theta \mathrm{e}^{i\lambda}| = 1 + \theta^2 + 2\theta\cos(\lambda)$$

   On a donc

   $$f(\lambda) = \frac{\sigma_\varepsilon^2}{2\pi}\left(1 + \theta^2 + 2\theta\cos(\lambda)\right) \tag{8.87}$$

3. Considérons la moyenne mobile infinie :

   $$X_t = \sum_{i\in\mathbb{Z}} \phi^i \varepsilon_{t-i}$$

   où $|\phi| < 1$. On obtient :

   $$f(\lambda) = \frac{\sigma_\varepsilon^2}{2\pi}\frac{1}{|1 - \phi\mathrm{e}^{i\lambda}|^2} = \frac{\sigma_\varepsilon^2}{2\pi}\frac{1}{1 + \phi^2 - 2\phi\cos(\lambda)} \tag{8.88}$$

   Remarquons l'égalité $X_t = \phi X_{t-1} + \varepsilon_t$.

Remarque 8.9

La densité spectrale est une fonction paire, positive,continue et périodique de période 2π. On peut alors raisonner sur l'intervalle continu $[-\pi, \pi], [0, 2\pi]$ ou plus généralement $\lambda = \pm 2\pi n$ pour $n \in \mathbb{R}$, par exemple pour des observations mensuelles, la période est l'année : $T = 12$ mois et la fréquence angulaire est : $\lambda = 2\pi/T = \pi/6$.

A partir de la densité spectrale, la transformation inverse de Fourier permet de retrouver la fonction d'autocovariance $\gamma(h)$, ainsi on a :

$$\gamma(h) = \int_{-\pi}^{\pi} f(\lambda)\cos(\lambda h)\mathrm{d}\lambda = \int_{-\pi}^{\pi} f(\lambda)\mathrm{e}^{i\lambda h}\mathrm{d}\lambda \tag{8.89}$$

Posons $\lambda_j = 2\pi j/T$, $j = 0, \ldots, T$. On a, dans le cas disrcétisée, la formule suivante :

$$\gamma(h) = \frac{1}{2\pi}\sum_{j=0}^{j=T} f(\lambda_j)\mathrm{e}^{i\lambda_j h} \tag{8.90}$$

Si on considère la trajectoire finie $X_1, \dots, X_T$, issue du processus stationnaire $(X_t)_{t\in\mathbb{Z}}$, alors la densité théorique du processus est estimée par la fonction, notée $I_T(.)$ calculée sur la série $X_1, \dots, X_T$ définie, pour tout $\lambda \in [-\pi, \pi]$, par

$$I_T(\lambda) = \frac{1}{2\pi T}\left|\sum_{t=1}^{t=T}(X_t - \overline{X}_T)\mathrm{e}^{-i\lambda t}\right|^2 \tag{8.91}$$

On l'appelle périodogramme.

Proposition 8.4

Si les frequences de Fourier λ_j sont non nulles pour tout j alors :

$$I_T(\lambda_j) = \frac{1}{2\pi}\sum_{h=-T}^{h=T}\hat{\gamma}(h)\mathrm{e}^{-i\lambda_j h} \tag{8.92}$$

On a aussi :

$$\begin{aligned} I_T(\lambda) &= \frac{1}{2\pi}\sum_{h=-T}^{h=T}\hat{\gamma}(h)\cos(\lambda h) \\ &= \frac{1}{2\pi}\left(\hat{\gamma}(0) + 2\sum_{h=1}^{h=T}\hat{\gamma}(h)\cos(\lambda h)\right) \end{aligned}$$

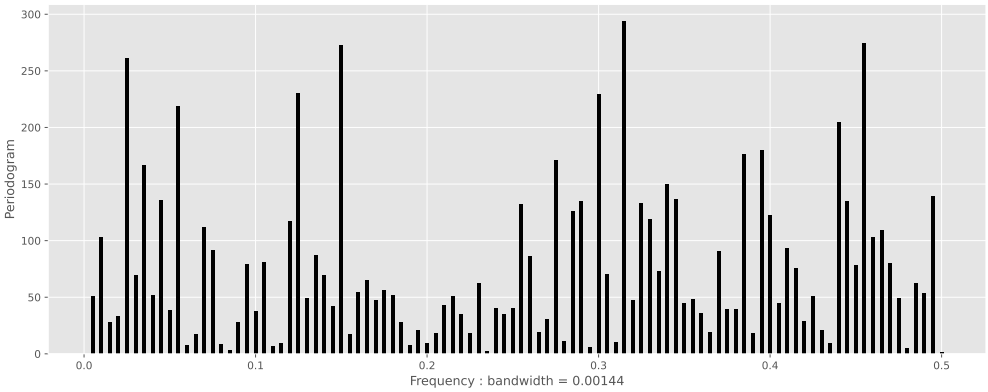
Cet estimateur de la densité spectrale du processus $(X_t)_{t\in\mathbb{Z}}$ est sans biais, mais non consistant.

Sous R, son estimation se fait grâce à la fonction periodogram du package TSA. Rappelons que la fonction `periodogram` se base sur la fonction spec du package `TSA` pour l'estimation de la densité spectrale au lieu de la fonction spec.pgram du package stats, ce qui conduit à des valeurs différentes de cette densité. Afin de remedier à ce problème, il faut fixer le paramètre `taper` à 0 dans la fonction `spec.pgram` (par défaut `taper=0.1`). Ce paramètre spécifie la proportion des données à réduire.

Nous implémentons l'équivalent de la fonction `periodogram` sous Python en utilisant la fonction `spec_pgram` et fixant `taper=0.0`.

```
# Périodogramme
def periodogram(x,plot=True):
  sp = spec_pgram(x,taper=0.0,demean=True,detrend=False,plot=False)
  sp["spec"] = 2*sp["spec"]
  temp = sp["spec"][sp["freq"]== 0.5]
  sp["spec"][sp["freq"]==0.5] = temp/2
  if plot:
    fig, axe = plt.subplots(figsize=(16,6))
    axe.bar(sp["freq"],sp["spec"],width = 0.002,color = 'black');
    axe.set_xlabel(f"Frequency : bandwidth = {round(sp['bandwidth'],5)}");
```

```
    axe.set_ylabel("Periodogram");
  return {"freq":sp["freq"], "spec":sp["spec"]}
# Application
pic = periodogram(goog200.diff().dropna())
plt.show()
```



On peut améliorer cet estimateur en utilisant une fonction de pondération. En effet, une expression du périodogramme lissé est donnée par :

$$I_T^l(\lambda_j) = \frac{1}{2\pi} \sum_{h=-m}^{h=m} \omega_T(h) I_T(\lambda_j + h) \tag{8.93}$$

où pour $j = 0, \dots, T-1$, $I_T(\lambda_j)$ est le périodogramme pour la fréquence de Fourier λ_j, m est un entier positif ou nul qui contrôle la longueur de la moyenne mobile et $(\omega_T(h))_h$ une suite de poids dont les valeurs sont telles que :

$$\omega_T = \begin{cases} \dfrac{1}{2m} & \text{pour} \quad h = -m+1, \dots, -1, 0, 1, \dots, m-1 \\ \dfrac{1}{4m} & \text{pour} \quad h = -m, m \end{cases} \tag{8.94}$$

Sous R, pour l'estimation du périodogramme par la méthode lissage, on utilise d'abord la fonction kernel de la librairie stats afin d'avoir le valeur du noyau qui sera par la suite insérer dans la fonction `spec.pgram`. Cette fonction retourne la valeur d'un noyau qui dépend de la méthode utilisée ("daniell" "dirichlet","fejer" ou "modified.daniell").

Exemple 8.18 *Densité spectrale d'un bruit blanc*

La densité spectrale d'un bruit blanc de variance σ^2 est donnée par :

$$f(\lambda) = \frac{1}{2\pi} \sum_{h=-\infty}^{+\infty} \gamma(h) \mathrm{e}^{i\lambda h} = \frac{\gamma(0)}{2\pi} = \frac{\sigma_\varepsilon^{2\pi}}{} = \text{cste} \tag{8.95}$$

car

$$\gamma(h) = \begin{cases} \sigma^2 & \text{pour} \quad h = 0 \\ 0 & \text{pour} \quad h \neq 0 \end{cases} \tag{8.96}$$

est la fonction d'autocovariance du bruit.

Inversement, si ladensité spectrale d'une série stationnaire (X_t), est constante : $f(\lambda) = c$, alors (X_t) un bruit blanc. En effet,

$$\gamma(h) = \int_{-\pi}^{\pi} f_\lambda \cos(\lambda h) \mathrm{d}\lambda = c \int_{-\pi}^{\pi} \cos(\lambda h) \mathrm{d}\lambda = 0 \quad \text{si} \quad \neq 0 \tag{8.97}$$

Cette nullité de la fonction d'autocorrélation, pour tout $h \neq 0$, est donc une caractéristique du bruit blanc.

Proposition 8.5

Soit $(X_t)_{t\in\mathbb{Z}}$ stationnaire au second ordre pouvant s'écrire sous la forme d'une moyenne mobile infinie et de densité spectrale f_X. Si $Y_t = \sum_{j\in\mathbb{Z}} \alpha_j X_{t-j}$ avec $\sum_{j\in\mathbb{Z}} |\alpha_j| < +\infty$. La densité spectrale f_Y du processus $(Y_t)_{t\in\mathbb{Z}}$ est donnée par :

$$f_Y(\lambda) = f_X(\lambda) \left| \sum_{j\in\mathbb{Z}} \alpha_j \mathrm{e}^{ij\lambda} \right|^2 \tag{8.98}$$

8.2.4.2 Corrélogramme

C'est la représentation graphique de la fonction d'autocorrélation. Du fait de la parité de cette dernière, la représentation est restreinte aux h positifs. Il est très utile pour repérer letype de modèle auquel correspond la série. Il permet également de voir si la série est stationnaire.

Pour visualiser l'autocorrélation simple d'un processus linéaire sous Python, on utilise la fonction plot_acf de la librairie Statsmodels. Nous créeons une fonction plot_colors afin de modifier la structure de coloration de la fonction plot_acf.

```
# Fonction de controle des paramètres
import matplotlib.pyplot as plt
def plot_colors(ax,markercolor="black",linecolor="black",facecolor="skyblue",
                barcolor="black", linewidth=1):
    from matplotlib.collections import PolyCollection, LineCollection
    for item in ax.collections:
        # change the color of the confidence interval
        if type(item) == PolyCollection:
            item.set_facecolor(facecolor)
        # change the color of the vertical lines
        if type(item) == LineCollection:
            item.set_color(barcolor)
    # change the color of the markers
    [line.get_label() for line in ax.lines]
    for item in ax.lines:
        item.set_color(markercolor);
    # change the color of the horizontal lines
    ax.lines[0].set_color(linecolor);
    ax.lines[0].set_linewidth(linewidth);
    ax.set_ylim(-1.05,1.05);
    return ax
```

```
# Représentation graphique
from statsmodels.graphics.tsaplots import plot_acf
fig, (axe1,axe2) = plt.subplots(1,2,figsize=(16,6))
plot_acf(goog200,ax=axe1,lags=24,alpha=0.05,adjusted=False);
plot_acf(goog200.diff().dropna(),ax=axe2,lags=24,alpha=0.05,adjusted=False);
axe1 = plot_colors(axe1);
axe1.set_title("cours de clôture");
axe1.set_xlabel("lag");
axe1.set_ylabel("ACF");
axe2 = plot_colors(axe2);
axe2.set_title("cours de clôture (différence première)");
axe2.set_xlabel("lag");
axe2.set_ylabel("ACF");
plt.tight_layout();
plt.show()
```

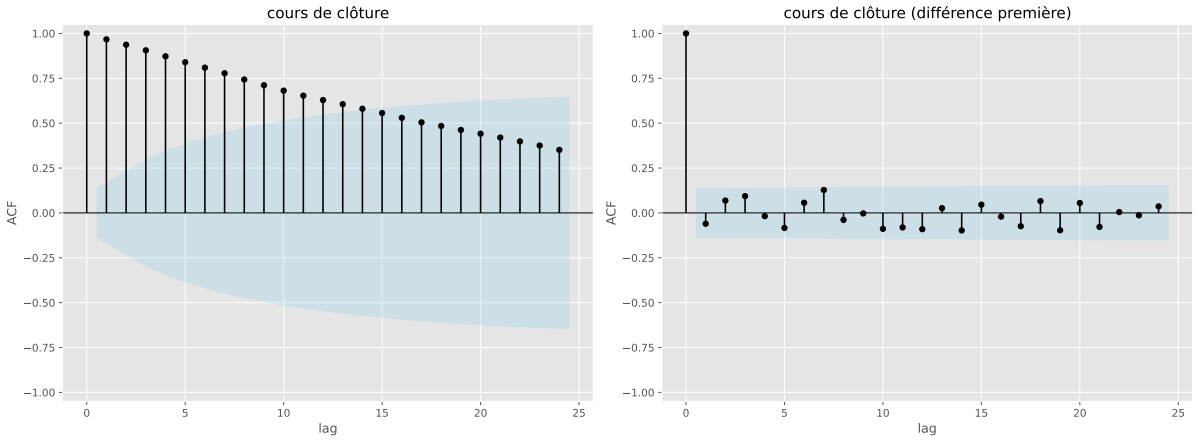


Figure 8.3 – L'ACF du cours de l'action Google (à gauche) et des variations quotidiennes du cours de l'action Google (à droite).

Pour visualiser l'autocorrélation partielle sous Python, on utilise la fonction plot_pacf de la librairie Statsmodels.

```
# Représentation graphique
from statsmodels.graphics.tsaplots import plot_pacf
fig, (axe1, axe2) = plt.subplots(1,2,figsize=(16,6))
plot_pacf(goog200,ax=axe1,lags=24,alpha=0.05);
plot_pacf(goog200.diff().dropna(),ax=axe2,lags=24,alpha=0.05);
axe1 = plot_colors(axe1);
axe1.set_title("cours de clôture");
axe1.set_xlabel("lag");
axe1.set_ylabel("PACF");
axe2 = plot_colors(axe2);
axe2.set_title("cours de clôture (différence première)");
axe2.set_xlabel("lag");
axe2.set_ylabel("PACF");
plt.tight_layout();
plt.show()
```

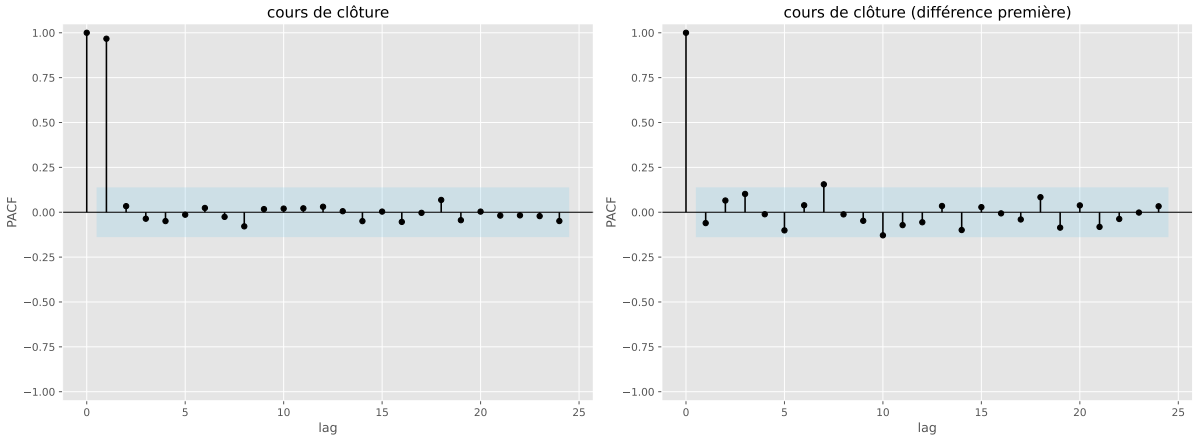


Figure 8.4 – Le PACF du cours de l'action Google (à gauche) et des variations quotidiennes du cours de l'action Google (à droite).

9 Processus linéaires de type ARMA

Sommaire

Dans ce chapitre, nous intéressons à l'étude d'une famille de processus linéaires stationnaires. Ces processus comme nous le verrons, sont très généreux car présentent une modélisation estimable, opérationnelle pour les séries temporelles. Nous allons nous intéresser au cas où la série de données à modéliser (une fois la tendance déterministe retranchée) présente une certaine homogénéité au cours du temps. On fait l'hypothèse que la variance des variables aléatoires reste constante et que les relations de dépendance entre variables successives sont les mêmes au cours du temps.

9.1 Quelques processus linéaires

Définition 9.1 *Processus linéaire*

Un processus $(X_t)_{t\in\mathbb{Z}}$ est dit processus linéaire s'il admet une décomposition de la forme suivante :

$$X_t = \sum_{i=-\infty}^{+\infty} \theta_i \varepsilon_{t-i}, \quad t \in \mathbb{Z} \tag{9.1}$$

avec $\sum_{i=-\infty}^{+\infty} \theta_i^2 < +\infty$ et $(\varepsilon_t)_t$ un bruit blanc de variance σ_ε^2 et $(\theta_i)_i$ est une suite de nombres réels.
Cette écriture est anticipative car elle s'exprime en fonction du passé mais aussi à travers le futur du processus. On peut en effet décomposer, à l'instant présent t,

$$\underbrace{X_t}_{\text{présent}} = \underbrace{\sum_{i=1}^{+\infty} \theta_i \varepsilon_{t-i}}_{\text{passé}} + \underbrace{\theta_0 \varepsilon_t}_{\text{présent}} + \underbrace{\sum_{i=-\infty}^{-1} \theta_i \varepsilon_{t-i}}_{\text{futur}} \tag{9.2}$$

Pour que (9.1) s'interprète correctement, il faut que $(X_t)_{t\in\mathbb{Z}}$ soit inversible, c'est - à - dire que l'on ait aussi :

$$\varepsilon_t = \sum_{j=-\infty}^{+\infty} \omega_j X_{t-j} \quad \text{avec} \quad \omega_0 = 1, \quad \text{et} \quad \sum_{j=0}^{+\infty} |\omega_j| < +\infty \tag{9.3}$$

Proposition 9.1

Si $(X_t)_t$ est un processus linéaire définit par (9.1) avec $\varepsilon_t \sim \mathcal{N}(0, \sigma_\varepsilon^2)$, alors $(X_t)_t$ est stationnaire et on a :

$$\gamma(h) = \sigma_\varepsilon^2 \sum_{j=-\infty}^{+\infty} \theta_j \theta_{j+h}, \quad h \in \mathbb{Z} \tag{9.4}$$

Dans la pratique, on se ramène à un modèle paramétrique en considérant les décompositions suivantes :

$$\begin{cases} X_t &= \sum_{j=1}^{j=q} \alpha_j \varepsilon_{t-j}, \quad t \in \mathbb{Z} \\ \varepsilon_t &= \sum_{j=1}^{j=q} \beta_j X_{t-j}, \quad t \in \mathbb{Z} \end{cases} \tag{9.5}$$

Les processus associés s'appellent respectivement processus moyenne mobile d'ordre q, noté $(MA(q))$ et processus autorégressif d'ordre p, noté $AR(p)$.

9.1.1 Processus moyenne mobile (Processus MA, *Moving Average)*

Un cas particulier de processus linéaire que l'on rencontre très souvent en pratique est la moyenne mobile. Cette dernière est communément utilisée pour lisser des séries chronologiques (faire disparaître les perturbations locales), ou pour modéliser les séries à très courte mémoire. Ils ont été introduits par Slutzky (1937). En effet, Slutsky dans son article a utilisé les nombres générés par la loterie Russe pour générer une série décrivant le cycle économique en Angleterre, de 1855 à 1877.

Définition 9.2 *Processus moyenne mobile*

On dit que la série chronologique $(X_t)_{t\in\mathbb{Z}}$ est un processus moyenne mobile d'ordre q si elle est défini, pour tout $t \in \mathbb{Z}$ par

$$X_t = \varepsilon_t - \theta_1 \varepsilon_{t-1} - \cdots - \theta_q \varepsilon_{t-q} \tag{9.6}$$

où les $\theta_i (i = 1, \ldots, q)$ sont des réels et (ε_t) est un bruit blanc centré de variance σ^2 et $\theta_q \neq 0$.

Le processus se réécrit : $X_t = \Theta(B)\varepsilon_t$ avec $\Theta(B) = I - \sum_{i=1}^{i=q} \theta_i B^i$. Pour signifier que le processus (X_t) est engendré par une moyenne mobile d'ordre q, on note : $(X_t)_{t\in\mathbb{Z}} \sim MA(q)$

Proposition 9.2

Par définition, un processus $MA(q)$ est toujours stationnaire. Il est inversible si le polynôme Θ a toutes ses racines de module strictement plus grand que 1.

On peut étendre cette définition aux $MA(+\infty)$ en faisant crôitre q. On pourra alors vérifier que $(X_t)_{t\in\mathbb{Z}}$ est stationnaire si et seulement si $\sum_{j\in\mathbb{Z}} \theta_j^2 < +\infty$.

Remarque 9.1

Si X est inversible alors

$$\varepsilon_t = \Theta^{-1}(B)X_t = \left(\sum_{j=0}^{+\infty} \psi_j B^j\right) X_t \tag{9.7}$$

Proposition 9.3

$(\varepsilon_t)_t$ est le processus des innovations de $(X_t)_t$ si et seulement si X est inversible.

Remarque 9.2

On peut montrer que, quitte à changer de bruit blanc et de coefficients θ_i, une moyenne mobile dont le polynôme associé a des racines à l'extérieur stricte du disque unité peut se ramener à une moyenne mobile inversible.

Exemple 9.1

Soit $(X_t)_{t\in\mathbb{Z}}$ un processus ayant la représentation $MA(1)$ suivante :

$$X_t = \varepsilon_t - 2\varepsilon_{t-1} \tag{9.8}$$

où $(\varepsilon_t)_t$ est un bruit blanc fort de variance 1 avec $\mathbb{E}\left(|\varepsilon_t|^3\right) < \infty$ et $\mathbb{E}\left(\varepsilon_0^3\right) = \mu$. Soit $(Y_t)_{t\in\mathbb{Z}}$ le processus défini par :

$$X_t = Y_t - \frac{1}{2}Y_{t-1} \tag{9.9}$$

On peut montrer facilement que

$$Y_t = \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i X_{t-i} \tag{9.10}$$

En effet, l'opérateur $\Theta(B) = \left(I - \frac{1}{2}B\right)$ est inversible et l'unique racine $z = 2$ est de module différent de 1. on a :

$$Y_t = \Theta^{-1}(B)X_t = \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i B^i X_t = \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i X_{t-i} \tag{9.11}$$

On montre également que $(Y_t)_{t\in\mathbb{Z}}$ est un bruit blanc. En effet, on a :

1. $\mathbb{E}(Y_t) = 0,\ \forall t$
2. $\mathrm{V}(Y_t) = 4 < \infty$
3. $\mathrm{Cov}(Y_t, Y_{t+h}) = 0,\ \forall h$

Exemple 9.2 *Simulation d'un processus MA(2)*

Pour tout $t \in \mathbb{Z}$, le processus $MA(2)$ est engendré par : $X_t = \epsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2}$, où (ε_t) est un bruit blanc de variance σ_ε^2. La figure (9.1) est simulée avec $\theta_1 = 0.65$ et $\theta_2 = 0.35$, $(\varepsilon_t) \sim \mathcal{N}(0,1)$ et $T = 1000$.

```
# Simulation d'un processus MA(2)
import numpy as np
from statsmodels.tsa.arima_process import arma_generate_sample
maparams = np.array([.65, .35])
ar = np.r_[1, 0, 0]
ma = np.r_[1, maparams]
sim_ma = arma_generate_sample(ar, ma, nsample = 1000, scale = 1,
                              distrvs =np.random.seed(123))
# Représentation graphique
import matplotlib.pyplot as plt
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(sim_ma,color='black');
axe.set_ylabel('$X_t$');
axe.set_xlabel('t');
plt.show()
```

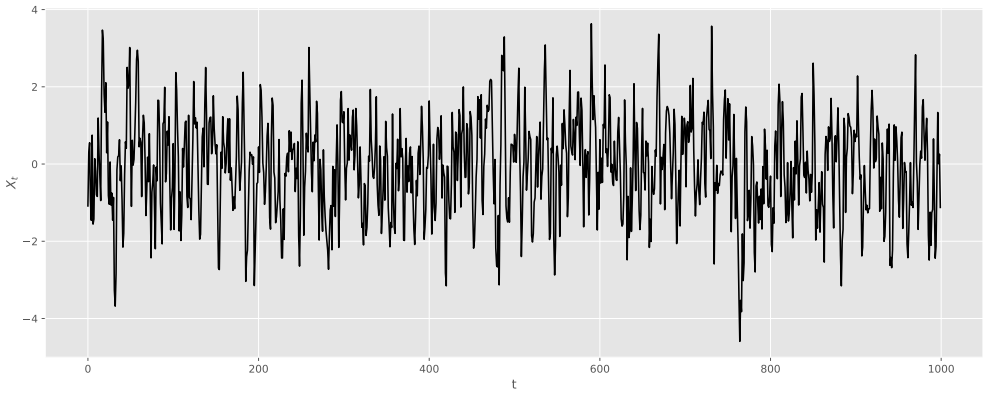


Figure 9.1 – Simulation d'un processus $X_t = (1 + 0.65B + 0.35B^2)\varepsilon_t$ pour $T = 1000$.

9.1.1.1 Autocovariance et autocorrélation simple d'un processus MA(q)

Soit $(X_t)_t$ un processus MA(q), alors la fonction d'autocovariance $\gamma(h)$ est donnée par :

$$\gamma(h) = \mathbb{E}\left(X_t X_{t-h}\right) = \begin{cases} \sigma^2\left(1 + \theta_1^2 + \ldots, +\theta_q^2\right) & \text{si } h = 0 \\ \sigma^2\left(\theta_0\theta_h + \theta_1\theta_{h+1} + \ldots, +\theta_{q-h}\theta_q\right) & \text{si } 1 \leq h \leq q \\ 0 & \text{si } h > q \end{cases} \tag{9.12}$$

avec $\theta_0 = -1$. On en déduit la fonction d'autocorrélation suivante :

$$\rho(h) = \begin{cases} 1 & \text{si } h = 0 \\ \dfrac{\theta_0\theta_h + \theta_1\theta_{h+1} + \ldots, +\theta_{q-h}\theta_q}{1 + \theta_1^2 + \ldots, +\theta_q^2} & \text{si } 1 \leq h \leq q \\ 0 & \text{si } h > q \end{cases} \tag{9.13}$$

Propriété 9.1

Pour un processus MA(q), $\rho(h) = 0$ pour $|h| > q$. En d'autres termes, les autocorrélations s'annulent à partir du rang $q+1$, lorsque le vrai processus générateur des données est un MA(q). Cette propriété fondamentale permet d'identifier l'ordre q des processus MA.

Exemple 9.3 *Fonction d'autocorrélation simple du processus $MA(2)$*

On calcule les autocorrélations simples du processus $X_t = (1 + 0.65B + 0.35B^2)\varepsilon_t$.

```
# Représentation graphique
from statsmodels.graphics.tsaplots import plot_acf
fig, axe = plt.subplots(figsize=(16,6))
plot_acf(sim_ma,ax=axe);
axe = plot_colors(axe);
axe.set_xlabel("Lag");
axe.set_ylabel("ACF");
plt.show()
```

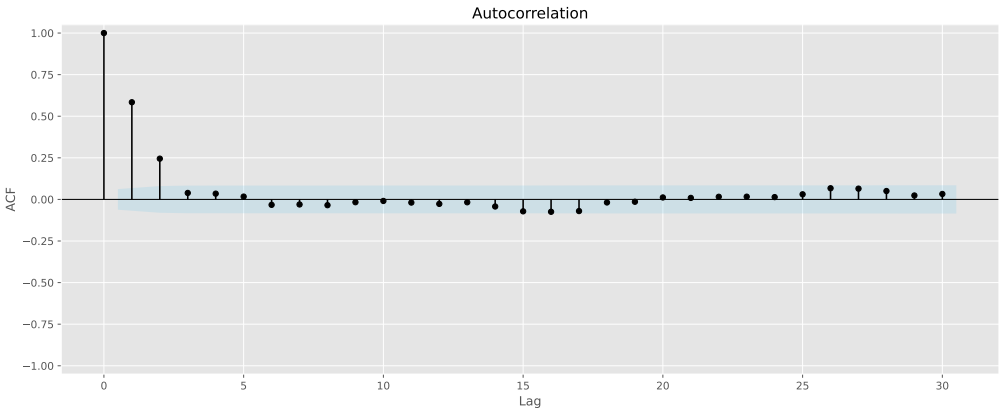


Figure 9.2 – Autocorrélation simple du processus $X_t = (1 + 0.65B + 0.35B^2)\varepsilon_t$

Exemple 9.4

On considère le processus MA(1) suivant :

$$X_t = \varepsilon_t - \theta_1\varepsilon_{t-1}, \quad |\theta_1| < 1, \quad \varepsilon_t \sim BB(0, \sigma_\varepsilon^2) \tag{9.14}$$

La fonction d'autocovariance de ce processus est :

$$\gamma(h) = \begin{cases} \sigma_\varepsilon^2 \left(1 + \theta_1^2\right) & \text{si } h = 0 \\ -\theta_1 \sigma_\varepsilon^2 & \text{si } h = 1 \\ 0 & \text{si } h > 1 \end{cases} \tag{9.15}$$

On en déduit la fonction d'autocorrélation suivante :

$$\rho(h) = \begin{cases} 1 & \text{si } h = 0 \\ -\dfrac{\theta_1}{1 + \theta_1^2} & \text{si } h = 1 \\ 0 & \text{si } h > 1 \end{cases} \tag{9.16}$$

9.1.1.2 Autocorrélations partielles d'un processus MA(q)

L'expression de la fonction d'autocorrélation partielle d'un MA(q) est relativement compliquée. On utilise l'algorithme de Durbin pour les calculer. Reprenons le cas du processus MA(1) définie précédemment, on a :

$$\begin{cases} r(1) = \rho(1) = -\dfrac{\theta_1}{1 + \theta_1^2} \\ r(h) = \dfrac{\rho(2) - \phi_{11}\rho(1)}{1 - \phi_{11}\rho(1)} = -\dfrac{\rho(1)^2}{1 - \rho(1)^2} \\ r(3) = \dfrac{\rho(3) - \phi_{21}\rho(2) - \phi_{22}\rho(1)}{1 - \phi_{21}\rho(1) - \phi_{22}\rho(2)} = -\dfrac{\phi_{22}\rho(1)}{1 - \phi_{21}\rho(1)} = \dfrac{\rho(1)^3}{\left(1 - \rho(1)^2\right)\left(1 - \phi_{21}\rho(1)\right)} \end{cases} \tag{9.17}$$

avec $\phi_{21} = \phi_{11} - \phi_{22}\phi_{11} = \phi_{11}\left(1 - \phi_{22}\right) = \dfrac{\rho(1)}{1 - \rho(1)^2}$.

La formule de récurrence pour les autocorrélations partielles d'un processus $MA(1)$ est donnée par :

$$r(h) = -\frac{\theta_1^h(1 - \theta_1^2)}{1 - \theta_1^{2(h+1)}}, \quad \text{pour} \quad h \geq 2 \tag{9.18}$$

Deux cas de figures sont envisageables :

— Si $\theta_1 > 0$, alors toutes les valeurs des autocorrélations partielles $r(h)$ sont négatives
— Si $\theta_1 < 0$, les autocorrélations partielles $r(h)$ alternent de signe.

Propriété 9.2

Pour un processus $MA(q)$, la fonction d'autocorrélation partielle n'a pas de propriété particulière. En effet, les valeurs de cette fonction ne s'annulent pas lorsque $h > q$. De façon générale, pour un $MA(q)$, la fonction d'autocorrélation partielle $r(h)$ tend vers 0 à vitesse exponentielle.

Remarque 9.3

Pour un $MA(1)$ de la forme $X_t = \varepsilon_t + \theta\varepsilon_{t-1}$ avec $||\theta| < 1$, on trouve

$$r(h) = (-1)^{h+1}\frac{\theta^h(1 - \theta^2)}{1 - \theta^{2(h+1)}}, \quad \text{pour} \quad h \geq 2 \tag{9.19}$$

Exemple 9.5 *Fonction d'autocorrélation partielle du processus $MA(2)$*

On calcule les autocorrélations partielles du processus $X_t = (1 + 0.65B + 0.35B^2)\varepsilon_t$.

```
# Représentation graphique
from statsmodels.graphics.tsaplots import plot_pacf
fig, axe = plt.subplots(figsize=(16,6))
plot_pacf(sim_ma,ax=axe);
axe = plot_colors(axe);
axe.set_xlabel("Lag");
axe.set_ylabel("ACF");
plt.show()
```

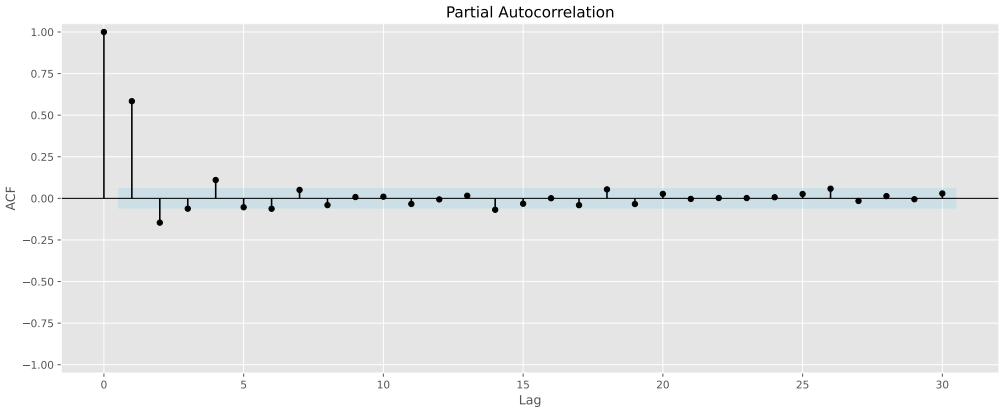


Figure 9.3 – Autocorrélation partielle du processus $X_t = (1 + 0.65B + 0.35B^2)\varepsilon_t$

Exemple 9.6

L'objectif de cet exemple est de calculer les autocorrélations partielles d'un processus $MA(1)$. Soit $(\varepsilon_t)_{t\in\mathbb{Z}}$ un bruit blanc de variance σ_ε^2 :

$$X_t = \varepsilon_t - \theta\varepsilon_{t-1} \quad t \in \mathbb{Z}, \quad 0 < |\theta| < 1. \tag{9.20}$$

On souhaite calculer les coefficients $\phi_1, \dots, \phi_n$ tels que $\sum_{j=1}^{n} \phi_j X_{n+1-j}$ soit la projection de X_{n+1} sur le sous-espace vectoriel de $\mathbb{L}^2$ engendré par $X_1, \dots, X_n$.

1. Montrer que les coefficients cherchés vérifient les équations :

$$\begin{cases} -\theta\phi_{j-1} + (1+\theta^2)\phi_j - \theta\phi_{j+1} = 0 \quad 2 \le j \le n-1 \\ (1+\theta^2)\phi_n - \theta\phi_{n-1} = 0 \\ (1+\theta^2)\phi_1 - \theta\phi_2 = -\theta \end{cases} \tag{9.21}$$

2. Résoudre ce système et montrer que l'autocorrélation partielle à l'ordre $n \ge 1$ est donnée par :

$$\phi_{nn} = -\frac{\theta^n(1-\theta^2)}{1-\theta^{2n+2}} \tag{9.22}$$

9.1.1.3 Densité spectrale d'un MA(q)

Si $(X_t)_{t\in\mathbb{Z}}$ est un processus MA(q) défini comme ci-dessus, alors sa densité spectrale est donnée par :

$$f(\lambda) = \frac{\sigma^2}{2\pi}|\Theta(e^{-i\lambda})|^2 \quad -\pi \leq \lambda \leq \pi \tag{9.23}$$

Pour le processus MA(1) définit précédemment, sa densité spectrale est donnée par :

$$f(\lambda) = \frac{\sigma_\varepsilon^2}{2\pi}|1-\theta_1 e^{-i\lambda}|^{-2} = \frac{\sigma_\varepsilon^2}{2\pi}(1-2\theta_1\cos\lambda+\theta_1^2) \tag{9.24}$$

Pour un MA(2) définit comme suit

$$X_t = \varepsilon_t - \theta_1\varepsilon_{t-1} - \theta_2\varepsilon_{t-2} \tag{9.25}$$

la densité spectrale est donnée par :

$$f(\lambda) = \frac{\sigma_\varepsilon^2}{2\pi}\left(1-2\theta_1(1-\theta_2)\cos\lambda - 2\theta_2\cos\lambda+\theta_1^2+\theta_2^2\right) \tag{9.26}$$

Exemple 9.7 *Densité spectrale du processus $MA(2)$*

On souhaite visualiser le périodogramme du processus $X_t = (1+0.65B+0.35B^2)\varepsilon_t$.

```
# Représentation graphique
from spectrum import *
res_ma = spec_pgram(sim_ma,demean = False,detrend = True)
fig, axe = plt.subplots(figsize=(16,6))
plot_spec(res_ma,coverage=0.95,ax=axe);
plt.show()
```

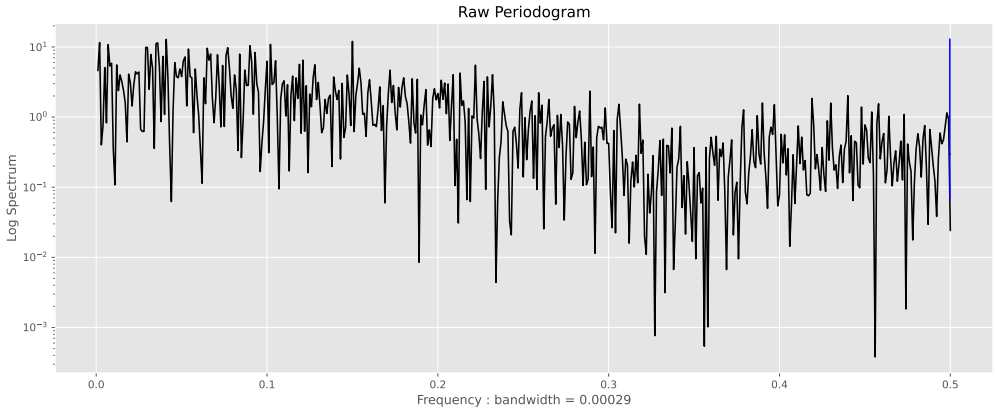


Figure 9.4 – Densité spectrale du processus $X_t = (1+0.65B+0.35B^2)\varepsilon_t$.

9.1.2 Processus Autorégressifs (Processus AR)

Les processus autorégressifs sont peut-être les plus utilisés en pratique pour modéliser les données chronologiques. Ils ont été introduits par Udny Yule (1927). L'idée est de régresser le phénomène aléatoire directement sur son passé, de manière linéaire.

Définition 9.3 *Processus autorégressif*

On appelle processus autorégressif d'ordre p, noté $AR(p)$, un processus stationnaire X_t vérifiant une relation du type :

$$X_t = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \varepsilon_t \tag{9.27}$$

où les ϕ_i(i = 1, ..., p) sont des paramètres réels à estimer (positifs ou négatifs) et (ε_t) est un bruit blanc centré de variance σ^2 et $\phi_p \neq 0$.

Il faut relever qu'ajouter à ce processus une constante ne modifie en rien les propriétés stochastiques et le corrélogramme simple d'un processus AR(p). Le processus (9.27) se réécrit : $\Phi(B)X_t = \varepsilon_t$ avec $\Phi(B) = I - \sum_{i=1}^{i=p} \phi_i B^i$. Pour signifier que le processus (X_t) est engendré par processus autorégressif d'ordre p, on note : $(X_t)_{t\in\mathbb{Z}} \sim AR(p)$.

Proposition 9.4

Si Φ a toutes ses racines de module différent de 1 alors il existe une solution stationnaire aux équations précédentes. Si, de plus, Φ a toutes ses racines de module strictement plus grand que 1 alors cette solution est unique et s'écrit sous la forme $MA(\infty)$ suivante :

$$X_t = \Phi^{-1}(B)\varepsilon_t = \left(\sum_{j=0}^{+\infty} \omega_j B^i\right) \varepsilon_t \tag{9.28}$$

Propriété 9.3

Si Φ a toutes ses racines de module strictement supérieur à 1, on dira que $(X_t)_t$ est causal.

Exemple 9.8

On considère le processus AR(2) suivant :

$$X_t = \frac{5}{6}X_{t-1} + \frac{1}{6}X_{t-2} + \varepsilon_t \tag{9.29}$$

On a $\Phi(B)X_t = \varepsilon_t$ avec $\Phi(B) = I - \frac{5}{6}B - \frac{1}{6}B^2 = \left(I - \frac{1}{2}B\right)\left(I - \frac{1}{3}B\right)$. Ses racines 2 et 3 sont de module strictement plus grand que 1, donc il existe bien une solution stationnaire et celle-ci est causale. De plus, elle s'écrit sous la forme $MA(\infty)$ suivante :

$$X_t = \Phi^{-1}(B)\varepsilon_t = \sum_{j=0}^{+\infty} \left(\frac{2}{3^j} - \frac{3}{2^j}\right) B^j \varepsilon_t = \sum_{j=0}^{+\infty} \left(\frac{2}{3^j} - \frac{3}{2^j}\right) \varepsilon_{t-j} \tag{9.30}$$

Proposition 9.5

Si X est causal alors $(\varepsilon_t)_t$ est son processus des innovations et donc $\sum_{i=1}^{p} \phi_j X_{t-j}$ est la projection orthogonale de X_t sur $\overline{\text{vect}}(X_{t-1}, X_{t-2}, \dots)$.

Exemple 9.9 *Simulation d'un processus AR(2)*

Pour tout $t \in \mathbb{Z}$, le processus AR(2) est engendré par : $X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \varepsilon_t$, où (ε_t) est un bruit blanc de variance σ_ε^2. La figure (9.5) est simulée avec $\phi_1 = 0.75$ et $\phi_2 = -0.25$, $(\varepsilon_t) \sim \mathcal{N}(0,1)$ et $T = 1000$.

```
# Simulation s"un processus AR(2)
arparams = np.array([.75, -.25])
ar = np.r_[1, -arparams]
ma = np.r_[1, 0, 0]
sim_ar = arma_generate_sample(ar, ma, nsample = 1000, scale = 1,
                              distrvs =np.random.seed(12345))
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(sim_ar,color='black');
axe.set_xlabel('t');
axe.set_ylabel('$X_t$');
plt.show()
```

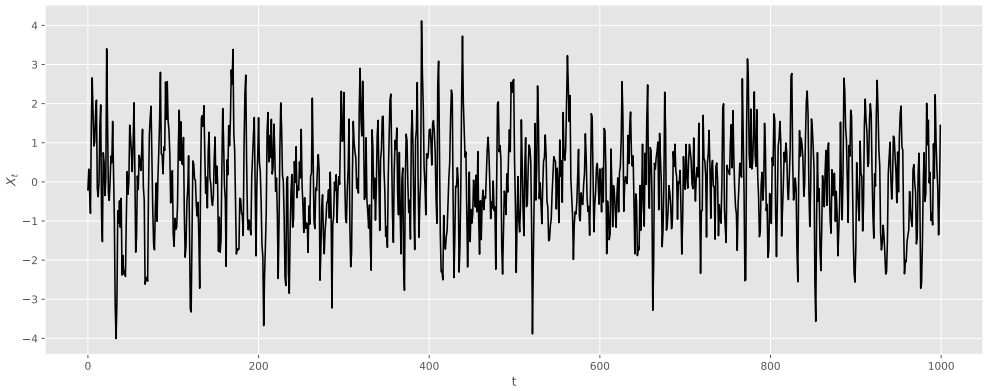


Figure 9.5 – Simulation d'un processus $(1 - 0.75B + 0.25B^2)X_t = \varepsilon_t$ pour $T = 1000$.

9.1.2.1 Autocorrélation d'un AR(p) et équation de Yule-Walker

a) Caractéristique d'un processus AR(1)

On considère le processus AR(1) suivant :

$$X_t = \phi_1 X_{t-1} + \varepsilon_t, \quad |\phi_1| < 1 \tag{9.31}$$

où ε_t est un bruit blanc de moyenne nulle et de variance σ_ε^2. On peut aisément calculer les caractéristiques suivantes : espérance mathématique, variance et autocovariance.

Le processus X est stationnaire ($|\phi_1| < 1$), alors $\mathbb{E}(X_t) = 0$. On peut généraliser l'écriture du modèle (9.31) pour considérer une espérance non nulle m avec :

$$X_t - m = \phi_1(X_{t-1} - m) + \varepsilon_t \tag{9.32}$$

On aura alors : $\mathbb{E}(X_t) = m$.

La variance du processus X est égale à :

$$\mathrm{V}(X_t) = \frac{\sigma_\varepsilon^2}{1 - \phi_1^2} = \gamma(0) \tag{9.33}$$

L'autocovariance d'ordre 1 est définie par :

$$\gamma(1) = \mathrm{Cov}(X_t, X_{t-1}) = \mathbb{E}(X_t X_{t-1}) = \phi_1 \mathbb{E}(X_{t-1}^2) + \mathbb{E}(\varepsilon_t X_{t-1}) = \phi_1 \gamma(0) \tag{9.34}$$

Plus généralement, pour un processus $AR(1)$, l'autocovariance d'ordre h est donnée par :

$$\gamma(h) = \phi_1^h \gamma(0) \tag{9.35}$$

Ce qui montre que l'autocovariance d'un processus $AR(1)$ décroît très rapidement. On déduit la fonction d'autocorrélation suivante :

$$\rho(h) = \begin{cases} 1 & \text{si } h = 0 \\ \phi_1^h & \text{si } h \geq 1 \end{cases} \tag{9.36}$$

On peut alors distinguer deux cas :

— Si $\phi_1 > 0$, $\rho(h) > 0$ et décroît de façon exponentielle quand h augmente.
— Si $\phi_1 < 0$, $\rho(h)$ alterne de signe et diminue de façon sinusoïdale (enveloppe exponentielle).

b) Caractéristique d'un processus AR(p)

Les autocovariances d'un processus $AR(p)$ peuvent être calculées en multipliant chaque membre de l'équation (9.27) par $X_{t-h} (h \geq 0)$.

$$X_t X_{t-h} = \phi_1 X_{t-1} X_{t-h} + \cdots + \phi_p X_{t-p} X_{t-h} + \varepsilon_t X_{t-h} \tag{9.37}$$

En prenant ensuite l'espérance des variables, on obtient :

$$\gamma(h) = \mathbb{E}\left(X_t X_{t-h}\right) = \phi_1 \mathbb{E}\left(X_{t-1} X_{t-h}\right) + \cdots + \phi_p \mathbb{E}\left(X_{t-p} X_{t-h}\right) + \mathbb{E}\left(\varepsilon_t X_{t-h}\right) \tag{9.38}$$

Puisque ε_t est un bruit blanc, on a :

$$\mathbb{E}\left(\varepsilon_t X_{t-h}\right) = \begin{cases} \sigma_\varepsilon^2 & \text{si} \quad h = 0 \\ 0 & \text{si} \quad h > 0 \end{cases}$$

On en déduit que :

$$\gamma(h) = \begin{cases} \sum_{i=1}^{p} \phi_i \gamma(h-i) + \sigma_\varepsilon^2 & \text{si } h = 0 \\ \sum_{i=1}^{p} \phi_i \gamma(h-i) & \text{si } h > 0 \end{cases} \tag{9.39}$$

La fonction d'autocorrélation d'un processus $AR(p)$ est donnée par :

$$\rho(h) = \sum_{i=1}^{p} \phi_i \rho(h-i), \quad h > 0 \tag{9.40}$$

Les autocorrélations d'un processus $AR(p)$ sont ainsi décrites par une équation de récurrence linéaire d'ordre p. En écrivant cette relation pour différentes valeurs de $h (h = 1, 2, \dots, p)$; on obtient les équations de Yule - Walker :

$$\begin{pmatrix} \rho(1) \\ \rho(2) \\ \vdots \\ \rho(p) \end{pmatrix} = \begin{pmatrix} 1 & \rho(1) & \rho(2) & \cdots & \rho(p-1) \\ \rho(1) & 1 & & & \vdots \\ \vdots & & & & \rho(1) \\ \rho(p-1) & & & & 1 \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_p \end{pmatrix} \tag{9.41}$$

Si les autocorrélations sont connues, la résolution de ce système permet de trouver les valeurs des paramètres du processus autorégressifs d'ordre p qui les ont engendré. En exprimant différemment le système, on peut trouver, connaissant la suite des p paramètres ϕ_i, les p premières autocorrélations du processus. Les suivantes sont alors données par la relation de récurrence précédente, en se servant des valeurs trouvées pour les p premières. Le système d'équation n'est pas difficile à trouver pour chaque cas particulier.

Exemple 9.10

Soit le processus $AR(2)$ suivant :

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \varepsilon_t \tag{9.42}$$

Les équations de Yule - Walker permettent d'écrire :

$$\begin{cases} \rho(1) = \phi_1 + \phi_2 \rho(1) \\ \rho(2) = \phi_1 \rho(1) + \phi_2 \end{cases} \tag{9.43}$$

D'où l'on tire l'expression des deux premières autocorrélations :

$$\begin{cases} \rho(1) = \dfrac{\phi_1}{1 - \phi_2} \\ \rho(2) = \phi_2 + \dfrac{\phi_1^2}{1 - \phi_2} \end{cases} \tag{9.44}$$

Il suffit ensuite d'appliquer la relation de récurrence pour trouver les autres autocorrélations.

Propriété 9.4

Pour un processus $AR(p)$, la fonction d'autocorrélation décroît de façon exponentielle ou diminue de façon sinusoïdale (enveloppe exponentielle), selon les signes des coefficients.

Exemple 9.11 *Autocorrélations simples d'un processus AR(2)*

On souhaiter représenter graphiquement les autocorrélations simples du processus $(1-0.75B+0.25B^2X_t) = \varepsilon_t$

```
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
plot_acf(sim_ar, ax=axe);
axe = plot_colors(axe);
axe.set_xlabel("Lag");
axe.set_ylabel("ACF");
plt.show()
```

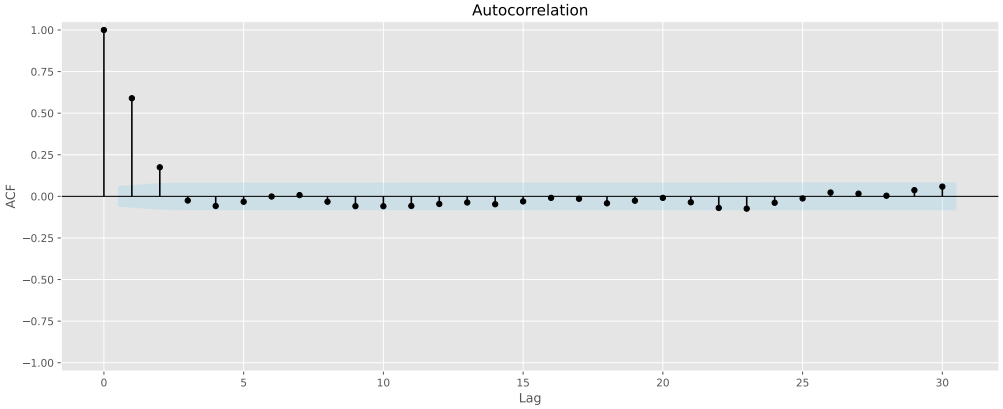


Figure 9.6 – Autocorrélation simple du processus $(1-0.75B+0.25B^2)X_t = \varepsilon_t$

Exemple 9.12

Soit $(\varepsilon_t)_{t\in\mathbb{Z}}$ un bruit blanc de variance $\sigma_\varepsilon^2 > 0$. On définit le processus $(X_t)_{t\in\mathbb{Z}}$ par :

$$X_t = \frac{1}{2}X_{t-1} - \frac{1}{4}X_{t-2} + \varepsilon_t, \quad t \in \mathbb{Z} \tag{9.45}$$

1. Montrer que $(\varepsilon_t)_{t\in\mathbb{Z}}$ est le processus innovation de $(X_t)_{t\in\mathbb{Z}}$
2. Montrer que la fonction d'autocovariance $\gamma(.)$ vérifie la récurrence

$$\gamma(h) = \frac{1}{2}\gamma(h-1) - \frac{1}{4}\gamma(h-2), \quad h > 0$$

3. Exprimer $\gamma(1)$ et $\gamma(2)$ en fonction $\gamma(0)$.
4. Résoudre l'équation de récurrence et exprimer la solution en fonction de $\gamma(0)$.
5. Calculer $\gamma(0)$ en fonction de σ_ε^2.
6. Donner également l'expression de $(X_t)_{t\in\mathbb{Z}}$ sous la forme d'une moyenne mobile infinie.

Solution

On définit le processus d'innovations de $(X_t)_{t\in\mathbb{Z}}$, le processus $(\varepsilon_t)_{t\in\mathbb{Z}}$ avec

$$\varepsilon_t = X_t - P_{F_t}(X_t) \tag{9.46}$$

où $P_{F_t}(X_t)$ est la projection orthogonale de X_t sur $F_t = \overline{\text{vect}}(X_{t-1}, X_{t-2}, \dots)$. Autrement dit, $P_{F_t}(X_t)$ est la meilleure prévision linéaire de X_t à partir du passé. Pour un processus $AR(p)$ avec $\Phi(B)X_t = \varepsilon_t$, si les racines de ϕ sont toutes de module plus grand que 1 (c'est-à-dire $(X_t)_{t\in\mathbb{Z}}$ est causal), alors $(\varepsilon_t)_{t\in\mathbb{Z}}$ est le processus d'innovation de $(X_t)_{t\in\mathbb{Z}}$.

Le polynôme $\Phi(z) = 1 - \frac{1}{2}z + \frac{1}{4}z^2$ admet deux racines complexes $z_1 = 1 - i\sqrt{3}$ et $z_2 = 1 + i\sqrt{3}$ de module supérieur à 1, donc $(\varepsilon_t)_{t\in\mathbb{Z}}$ est le processus d'innovation de $(X_t)_{t\in\mathbb{Z}}$.

pour tout $h > 0$, on a :

$$\begin{aligned}
\gamma(h) &= \text{Cov}(X_t, X_{t-h}) = \text{Cov}\left(\frac{1}{2}X_{t-1} - \frac{1}{4}X_{t-2} + \varepsilon_t, X_{t-h}\right) \\
&= \frac{1}{2}\text{Cov}(X_{t-1}, X_{t-h}) - \frac{1}{4}\text{Cov}(X_{t-2}, X_{t-h}) + \text{Cov}(\varepsilon_t, X_{t-h}) \\
&= \frac{1}{2}\gamma(h-1) - \frac{1}{4}\gamma(h-2) + \text{Cov}(\varepsilon_t, X_{t-h})
\end{aligned}$$

$\text{Cov}(\varepsilon_t, X_{t-h}) = \mathbb{E}(\varepsilon_t X_{t-h}) - \mathbb{E}(\varepsilon_t)\mathbb{E}(X_{t-h}) = \mathbb{E}(\varepsilon_t X_{t-h}) = 0, h > 0$ car $\varepsilon_t \in \overline{\text{vect}}(X_{t-1}, X_{t-2}, \dots)^T$, d'où :

$$\gamma(h) = \frac{1}{2}\gamma(h-1) - \frac{1}{4}\gamma(h-2), \quad h > 0 \tag{9.47}$$

Pour $h = 1$ et $h = 2$, on obtient le système suivant :

$$\begin{cases} \gamma(1) = \dfrac{2}{5}\gamma(0) \\ \gamma(2) = -\dfrac{1}{20}\gamma(0) \end{cases} \tag{9.48}$$

L'équation caractéristique $r^2 - \frac{1}{2}r - \frac{1}{4} = 0$ associée à l'équation de récurrence (9.47) admet deux racines complexes $r_1 = \frac{1}{4} - i\frac{\sqrt{3}}{4}$ et $r_2 = \frac{1}{4} + i\frac{\sqrt{3}}{4}$. La solution générale est :

$$\forall a, b \in \mathbb{R}, \quad \gamma(h) = \left(\frac{1}{2}\right)^h \left[a\cos\left(\frac{\pi}{3}h\right) + b\sin\left(\frac{\pi}{3}h\right)\right], \quad h > 0 \tag{9.49}$$

Les paramètres a et b sont déterminées grâce aux expressions de $\gamma(1)$ et $\gamma(2)$. On obtient :

$$\begin{cases} a = \gamma(0) \\ b = \dfrac{\sqrt{3}}{5}\gamma(0) \end{cases} \tag{9.50}$$

D'où, l'unique solution générale est :

$$\gamma(h) = \gamma(0)\left(\frac{1}{2}\right)^h \left[\cos\left(\frac{\pi}{3}h\right) + \frac{\sqrt{3}}{5}\sin\left(\frac{\pi}{3}h\right)\right], \quad \forall h > 0 \tag{9.51}$$

Pour $h = 0$, on a :

$$\gamma(0) = \text{Cov}(X_t, X_t) = \text{Cov}\left(\frac{1}{2}X_{t-1} - \frac{1}{4}X_{t-2} + \varepsilon_t, \frac{1}{2}X_{t-1} - \frac{1}{4}X_{t-2} + \varepsilon_t\right) = \frac{80}{63}\sigma_\varepsilon^2 \tag{9.52}$$

Les racines de Φ sont de module différent de 1, par conséquent $\Phi(B)$ est inversible. D'où :

$$X_t = \Phi^{-1}(B)\varepsilon_t = \frac{2}{\sqrt{3}}\sum_{j=0}^{+\infty}\frac{1}{2^j}\sin\left[(1+j)\frac{\pi}{3}\right]\varepsilon_{t-j} \tag{9.53}$$

9.1.2.2 Autocorrélation partielle d'un AR(p)

Il est possible de calculer les autocorrélations partielles du processus AR à partir des équations de Yule - Walker et des autocorrélations. On utilise pour cela l'algorithme de Durbin (1960) :

$$\begin{cases} r(1) = \rho(1) & \text{initialisation de l'algorithme} \\ r(h) = \dfrac{\rho(h) - \sum_{j=1}^{h-1}\phi_{h-1,j}\rho(h-j)}{1 - \sum_{j=1}^{h-1}\phi_{h-1,j}\rho(j)} & \text{pour} \quad h = 2, 3, ... \end{cases} \tag{9.54}$$

avec $\phi_{hj} = \phi_{h-1,j} - \phi_{hh}\phi_{h-1,h-j}$ pour $h = 2, 3, ...$ et $j = 1, ..., h-1$

a) Caractéristique d'un AR(1)

On utilise l'algorithme de Durbin, on a : $\begin{cases} r(1) & = \rho(1) = \phi_1 \\ r(2) & = \dfrac{\rho(2) - r(1)\rho(1)}{1 - \phi_{11}\rho(1)} = \dfrac{\rho(2) - \rho(1)^2}{1 - \rho(1)^2} \end{cases}$

or $\rho(2) = \phi_1\rho(1) = \rho(1)^2$, d'où : $r(2) = 0$

On a donc, pour un processus AR(1) :

$$r(h) = 0, \quad \forall h > 1 \tag{9.55}$$

Propriété 9.5

Pour un processus AR(p), $r(h) = 0 \quad \forall |h| > p$. En d'autres termes, pour un processus AR(p), les autocorrélations partielles s'annulent à partir du rang $p + 1$. Cette propriété fondamentale permet d'identifier l'ordre p des processus AR.

Exemple 9.13 *Autocorrélations partielles du processus $AR(2)$*

On souhaite visualiser les autocorrélations partielles du processus $(1 - 0.75B + 0.25B^2X_t) = \varepsilon_t$

```
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
plot_pacf(sim_ar,ax=axe);
```

```
axe = plot_colors(axe);
axe.set_xlabel("Lag");
axe.set_ylabel("PACF");
plt.show()
```

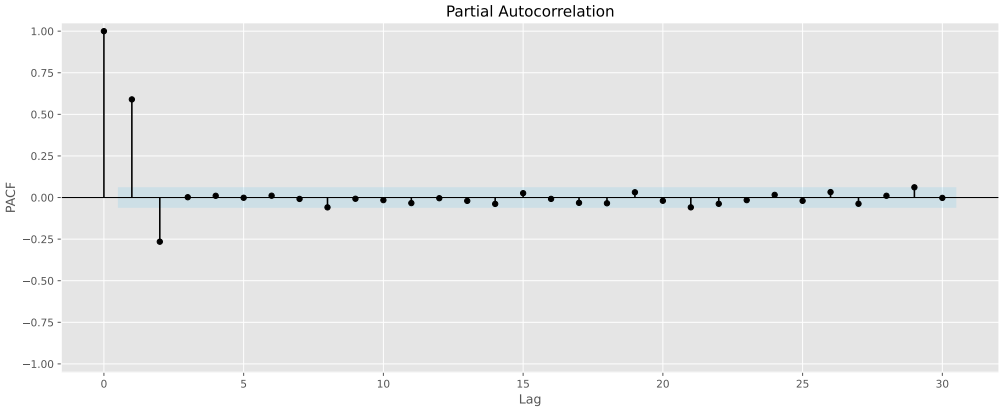


Figure 9.7 – Autocorrélation partielle du processus $(1 - 0.75B + 0.25B^2 X_t) = \varepsilon_t$

9.1.2.3 Densité spectrale d'un AR(p)

Si $(X_t)_{t\in\mathbb{Z}}$ est un processus AR(p) défini comme ci-dessus, alors sa densité spectrale est donnée par :

$$f(\lambda) = \frac{\sigma_\varepsilon^2}{2\pi} \frac{1}{|\Phi(e^{-i\lambda}|^2} \quad -\pi \leq \lambda \leq \pi \tag{9.56}$$

Soit X_t un processus AR(1) défini comme suit : $X_t - \phi X_{t-1} = \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, \sigma_\varepsilon^2)$, la densité spectrale de (X_t) est donnée par :

$$f(\lambda) = \frac{\sigma_\varepsilon^2}{2\pi} |1 - \phi e^{-i\lambda}|^{-2} = \frac{\sigma_\varepsilon^2}{2\pi(1 - 2\phi\cos\lambda + \phi^2)} \tag{9.57}$$

Pour un AR(2) : $X_t - \phi_1 X_{t-1} - \phi_2 X_{t-2} = \varepsilon_t$, la densité spectrale est donnée par :

$$f(\lambda) = \frac{\sigma_\varepsilon^2}{2\pi\left(1 - 2\phi_1(1 - \phi_2)\cos\lambda - 2\phi_2\cos\lambda + \phi_1^2 + \phi_2^2\right)} \tag{9.58}$$

Exemple 9.14 *Densité spectrale d'un processus AR(2)*

On souhaite visualiser le périodogramme du processus $(1 - 0.75B + 0.25B^2 X_t) = \varepsilon_t$

```
# Représentation graphique
res_ar = spec_pgram(sim_ar,demean = False,detrend = True)
fig, axe = plt.subplots(figsize=(16,6));
plot_spec(res_ar,coverage=0.95,ax=axe);
plt.show()
```

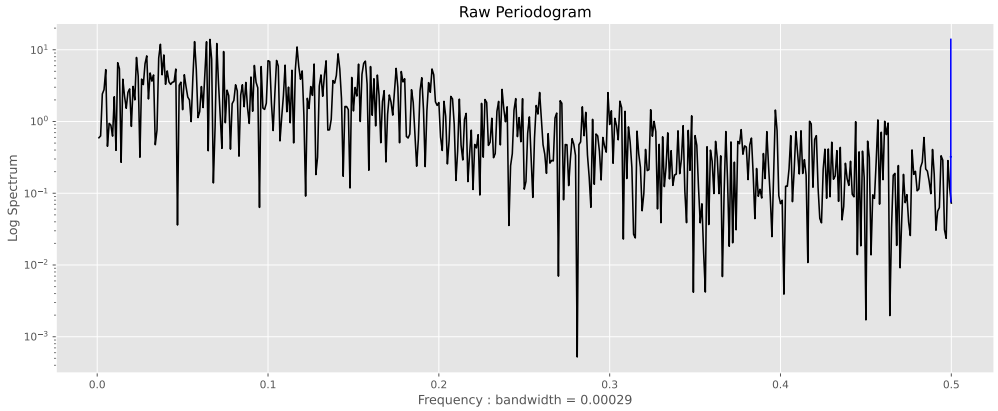


Figure 9.8 – Densité spectrale du processus $(1 - 0.75B + 0.25B^2 X_t) = \varepsilon_t$.

9.1.3 Processus ARMA constants

Cette partie est consacré aux processus ARMA à coefficients constants, principalement aux processus univariés. Les résultats sont bine connus (par exmple Quenouille (1957), E. Hannan (1973), Fuller (2009), Priestley (1981)).

Plusieurs processus aléatoires stationnaires ne peuvent être modélisés uniquement comme des processus MA purs ou AR purs car leurs caractéristiques sont souvent des combinaisons de deux types de processus. Ainsi, une classe de processus combinant les processus AR et MA a été développée par G. Box and Jenkings (1970) : les modèles ARMA.

Le processus ARMA permet de modéliser un processus autorégressif engendré par une moyenne mobile. Ainsi, sur $\mathbb{Z}$, si $(Y_t)_t$ est un MA(q) et si $(X_t)_t$ est un AR(p) engendré par $(Y_t)_t$, alors $(X_t)_t$ est un $ARMA(p, q)$.

Définition 9.4 *Processus autorégressif moyenne mobile*

On dit qu'un processus stationnaire $(X_t)_t$ est un processus autorégressif moyenne mobile d'ordre (p,q), noté $ARMA(p, q)$, s'il est défini, pour tout $t \in \mathbb{Z}$, par

$$X_t = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \cdots - \theta_q \varepsilon_{t-q} \tag{9.59}$$

où les $\phi_i (i = 1, \ldots, p)$ et $\theta_i (i = 1, \ldots, q)$ sont des réels et (ε_t) est un bruit blanc centré de variance σ_ε^2, $\phi_p \neq 0$ et $\theta_q \neq 0$.

Pour signifier que le processus (X_t)est un autorégressif moyenne mobile d'ordre(p, q), on note : $(X_t)_{t\in\mathbb{Z}} \sim ARMA(p, q)$.

En introduisant l'opérateur retard, l'équation (9.59) s'écrit : $\Phi(B)X_t = \Theta(B)\varepsilon_t$ avec $\Phi(B) = I - \phi_1 B - \cdots - \phi_p B^p$ et $\Theta(B) = I - \theta_1 B - \cdots - \theta_q B^q$.

Propriété 9.6

Un processus ARMA est stationnaire si le polynôme Φ a toutes ses racines à l'extérieur du disque unité et inversible si toutes les racines du polynôme Θ sont à l'extérieur du disque unité. En conséquence, si Φ et Θ ont leurs racines à l'extérieur du disque unité, on peut écrire un processus ARMA(p,q) :

1. Soit sous la forme $MA(\infty)$:

$$X_t = \frac{\Theta(B)}{\Phi(B)}\varepsilon_t = \sum_{i=1}^{+\infty} \alpha_i \varepsilon_{t-i}, \quad \text{avec} \quad \sum_{i=1}^{+\infty} |\alpha_i| < \infty \quad \text{et} \quad \alpha_0 = 1 \tag{9.60}$$

2. Soit sous la forme $AR(\infty)$:

$$\varepsilon_t = \frac{\Phi(B)}{\Theta(B)}X_t = \sum_{i=1}^{+\infty} \beta_i X_{t-i}, \quad \text{avec} \quad \sum_{i=1}^{+\infty} |\beta_i| < \infty \quad \text{et} \quad \beta_0 = 1 \tag{9.61}$$

Exemple 9.15 *Simulation d'un processus ARMA(2,2)*

Pour tout $t \in \mathbb{Z}$, le processus ARMA(2,2) est engendré par $X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2}$, où (ε_t) est un bruit blanc de variance σ_ε^2. La figure (9.9) est simulée avec $\phi_1 = 0.75$, $\phi_2 = -0.25$, $\theta_1 = -0.65$ et $\theta_2 = -0.35$, $(\varepsilon_t) \sim \mathcal{N}(0,1)$ et $T = 1000$.

```
# Représentation graphique
arparams = np.array([.75, -.25])
maparams = np.array([.65, .35])
ar = np.r_[1, -arparams]
ma = np.r_[1, maparams]
sim_arma = arma_generate_sample(ar, ma, nsample = 1000, scale = 1,
                                distrvs =np.random.seed(123))
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(sim_arma, color='black');
axe.set_xlabel('t');
axe.set_ylabel('$X_t$');
plt.show()
```

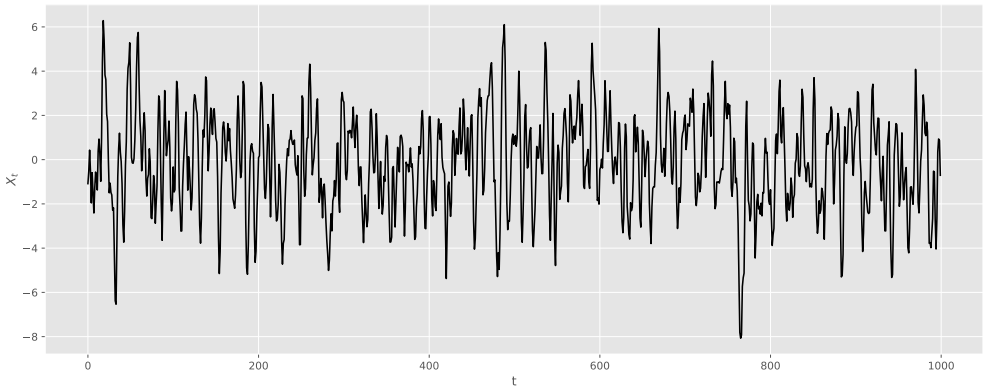


Figure 9.9 – Simulation d'un processus $(1 - 0.75B + 0.25B^2)X_t = (1 + 0.65B + 0.35B^2)\varepsilon_t$ pour $T = 1000$.

9.1.3.1 Autocorrélation d'un ARMA(p,q)

Pour calculer les autocorrélations d'un processus ARMA, on procède comme dans le cas des processus AR. Cette fonction répond à la même équation aux différences que ces des processus AR.

a) Caractéristique d'un ARMA(1,1)

On considère le processus ARMA(1,1) suivant :

$$X_t = \phi_1 X_{t-1} + \varepsilon_t - \theta_1 \varepsilon_{t-1}, \quad \forall t \in \mathbb{Z}, |\phi_1| < 1 \text{ et } |\theta_1| < 1 \tag{9.62}$$

On voit facilement que $\mathbb{E}(X_t) = 0$. Pour calculer la variance, on va prendre le carré des deux membres de (9.62) et en prendre l'espérance :

$$\gamma(0) = \frac{1 - 2\phi_1\theta_1 + \theta_1^2}{1 - \phi_1^2}\sigma_\varepsilon^2 \tag{9.63}$$

On remarque que si $\phi_1 = 0$, on retrouve la variance d'un MA(1) et pour $\theta_1 = 0$, on retrouve celle d'un AR(1).

L'autocovariance à l'ordre 1 s'obtient d'une manière similaire en multipliant l'équation (9.62) par X_{t-1} et en prenant l'espérance :

$$\gamma(1) = \frac{(1 + \phi_1\theta_1)(\phi_1 + \theta_1)}{1 - \phi_1^2}\sigma_\varepsilon^2 \tag{9.64}$$

Pour $h \geq 2$, on a la relation suivante :

$$\gamma(h) = \phi_1 \gamma(h-1), \quad h \geq 2 \tag{9.65}$$

Mais à l'inverse des processus AR purs, cette relation n'est pas vraie pour $h = 1$. On a la superposition de deux types d'autocorrélation, celle du MA qui est nuelle pour $h \geq 2$ et celle de l'AR qui décroît rapidement avec h. La fonction d'autocorrélation s'écrit donc :

$$\rho(h) = \phi_1 \rho(h-1), \quad h \geq 2 \tag{9.66}$$

Ainsi, à partir de l'ordre $h = 2$, on retrouve les autocorrélations du processus AR(1). On en déduit le comportement suivant pour la fonction d'autocorrélation :

- Si $\phi_1 > 0$, les autocorrélations diminuent de façon exponentielle,
- Si $\phi_1 < 0$, les autocorrélations diminuent de façon sinusoïdale (enveloppe exponentielle).

Remarque 9.4

Une autre façon de calculer l'autocovariance d'un processus $ARMA(1,1)$ est de passer par la forme moyenne mobile infinie du processus. En effet, puisque $|\theta_1| < 1$ et $|\phi_1| < 1$, on peut écrire :

$$X_t = \sum_{i=0}^{+\infty} \phi_1^i \left(\varepsilon_{t-i} - \theta_1 \varepsilon_{t-1-i}\right) = \varepsilon_t + \sum_{i=1}^{+\infty} \left(\phi_1^i - \theta_1 \phi_1^{i-1}\right) \varepsilon_{t-i} = \varepsilon_t + (\phi_1 - \theta_1) \sum_{i=1}^{+\infty} \phi_1^{i-1} \varepsilon_{t-i} \tag{9.67}$$

On peut directement calculer la fonction d'autocovariance à partir de cette expression. La variance $\gamma(0)$ vaut

$$\gamma(0) = \sigma_\varepsilon^2 \left(1 + (\phi_1 - \theta_1)^2 \sum_{i=1}^{2} \phi_1^{2(i-1)}\right) = \sigma_\varepsilon^2 \left(1 + \frac{(\phi_1 - \theta_1)^2}{1 - \phi_1^2}\right)$$

De plus, pour $h > 0$, on a :

$$\mathrm{Cov}(X_t, X_{t+h}) = \sigma_\varepsilon^2 \left((\phi_1 - \theta_1)\, \phi_1^{h-1} + (\phi_1 - \theta_1)^2\, \phi_1^h \sum_{i=1}^{+\infty} \phi_1^{2(j-1)} \right)$$

On voit alors que

$$\gamma(h) = \sigma_\varepsilon^2 \left((\phi_1 - \theta_1)\, \phi_1^{h-1} + \frac{(\phi_1 - \theta_1)^2\, \phi_1^h}{1 - \phi_1^2} \right) \tag{9.68}$$

b) Caractéristique d'un ARMA(p,q)

Plus généralement, pour un processus ARMA(p,q), la fonction d'autocovariance est définie par :

$$\gamma(h) = \begin{cases} \displaystyle\sum_{i=1}^{p} \phi_i \gamma(h-i) & \text{pour } h > q \text{ (ou } h \geq \max(p, q+1)) \\ \displaystyle\sum_{i=1}^{p} \phi_i \gamma(h-i) + \sigma_\varepsilon^2 \sum_{j=h}^{q} \theta_j \psi_{j-h} & \text{pour } 0 < h < \max(p, q+1) \\ \displaystyle\sum_{i=1}^{p} \phi_i \gamma(i) + \sigma_\varepsilon^2 \left(1 - \sum_{j=1}^{q} \theta_j \psi_j \right) & \text{pour } h = 0 \end{cases} \tag{9.69}$$

avec

$$\psi_j = \begin{cases} 1 & \text{si } j = 1 \\ \theta_j - \displaystyle\sum_{i=1}^{j} \phi_i \psi_{j-i} & \text{si } 0 < j \leq \max(p, q+1) \end{cases} \tag{9.70}$$

La fonction d'autocorrélation est donnée par :

$$\rho(h) = \frac{\gamma(h)}{\gamma(0)} \tag{9.71}$$

Propriété 9.7

Pour un processus ARMA(p,q), la fonction d'autocorrélation décroît de façon exponentielle ou diminue de façon sinusoïdale (enveloppe exponentielle), selon les signes des coefficients.

Exemple 9.16 *Autocorrélation simple d'un processus $ATMA(2,2$*

On souhaite visualiser l'autocorrélation simple du processus $(1 - 0.75B + 0.25B^2)X_t = (1 + 0.65B + 0.35B^2)\varepsilon_t$

```
# Représentation
fig, axe = plt.subplots(figsize=(16,6))
plot_acf(sim_arma, ax=axe);
axe = plot_colors(axe);
```

```
axe.set_xlabel("Lag");
axe.set_ylabel("ACF");
plt.show()
```

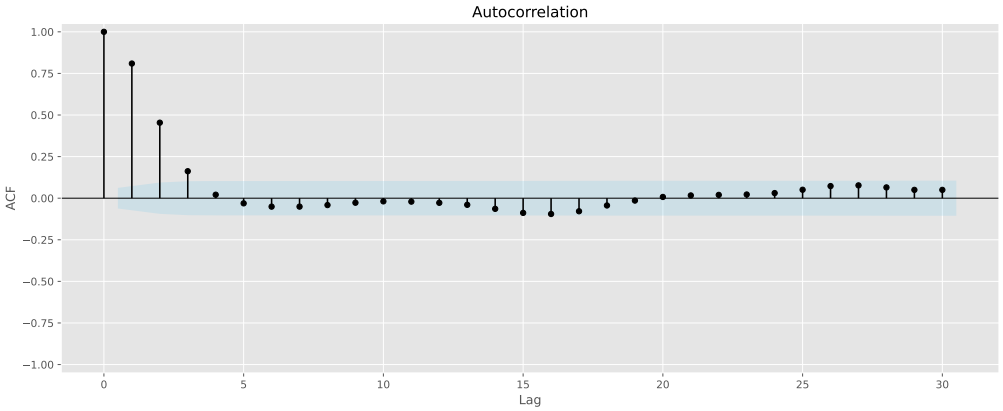


Figure 9.10 – Autocorrélation simple du processus $(1-0.75B+0.25B^2)X_t = (1+0.65B+0.35B^2)\varepsilon_t$

9.1.3.2 Autocorrélation partielle d'un ARMA(p,q)

La fonction d'autocorrélation partielle des processus ARMA n'a pas d'expression simple. Elle dépend de l'ordre de chaque partie (p et q) et de la valeur des paramètres. Elle se caractérise le plus fréquemment soit par une forme exponentielle décroissante, soit par une oscillatoire amortie.

Propriété 9.8

Pour un processus ARMA(p,q), la fonction d'autocorrélation décroît de façon exponentielle ou diminue de façon sinusoïdale (enveloppe exponentielle), selon les signes des coefficients.

Exemple 9.17 *Autocorrélations partielles d'un processus* $ARMA(2,2)$

On souhaite visualiser les autocorrélations partielles du processus $(1-0.75B+0.25B^2)X_t = (1+0.65B+0.35B^2)\varepsilon_t$

```
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
plot_pacf(sim_arma, ax=axe);
axe = plot_colors(axe);
axe.set_xlabel("Lag");
axe.set_ylabel("PACF");
plt.show()
```

9.1.3.3 Densité spectrale d'un ARMA(p,q)

Si $(X_t)_t$ est un processus ARMA(p,q), alors sa densité spectrale f_X est donnée par :

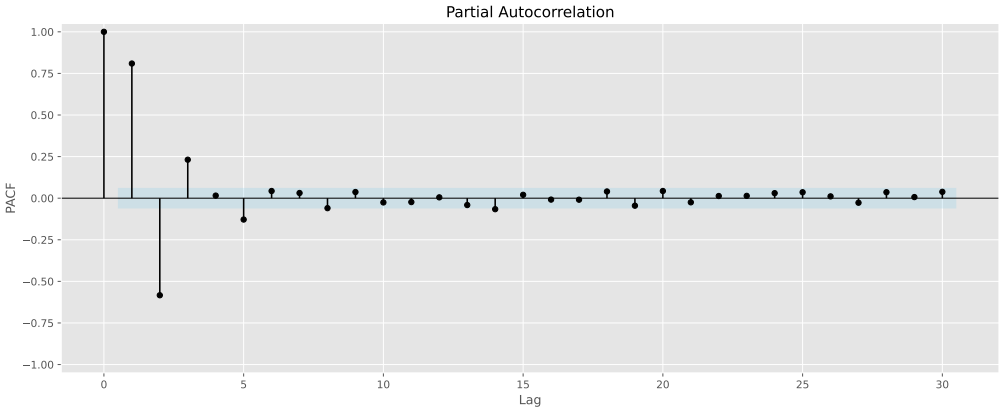


Figure 9.11 – Autocorrélation partielle du processus $(1-0.75B+0.25B^2)X_t = (1+0.65B+0.35B^2)\varepsilon_t$

$$f_X(\lambda) = \frac{\sigma_\varepsilon^2}{2\pi}\frac{|\Theta(e^{-i\lambda})|^2}{|\Phi(e^{-i\lambda})|^2}, \quad -\pi \leq \lambda \leq \pi \tag{9.72}$$

Pour un processus ARMA$(1,1)$: $X_t - \phi_1 X_{t-1} = \varepsilon_t - \theta_1 \varepsilon_{t-1}$, la densité spectrale est donnée par

$$f_X(\lambda) = \frac{\sigma_\varepsilon^2}{2\pi}\frac{1-2\theta_1\cos\lambda+\theta_1^2}{1-2\phi_1\cos\lambda+\phi_1^2} \tag{9.73}$$

Exemple 9.18 *Densité spectrale du processus $ARMA(2,2)$*

On souhaite visualiser le périodogramme du processus $(1-0.75B+0.25B^2)X_t = (1+0.65B+0.35B^2)\varepsilon_t$.

```
# Représentation graphique
res_arma = spec_pgram(sim_arma,demean = False,detrend = True)
fig, axe = plt.subplots(figsize=(16,6))
plot_spec(res_arma,coverage=0.95,ax=axe);
plt.show()
```

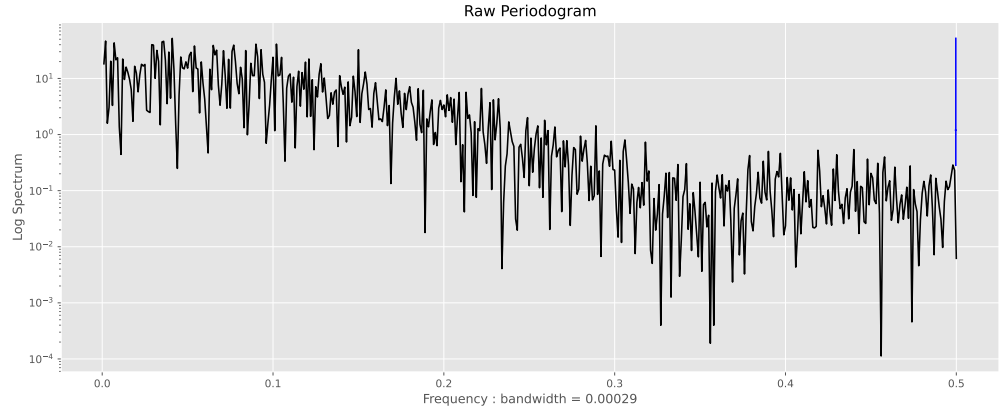


Figure 9.12 – Densité spectrale du processus $(1-0.75B+0.25B^2)X_t = (1+0.65B+0.35B^2)\varepsilon_t$.

Exemple 9.19

On considère le processus ARMA(1,1) définit comme suit :

$$X_t = \frac{1}{2}X_{t-1} + \varepsilon_t - \frac{1}{4}\varepsilon_{t-1} \tag{9.74}$$

où $(\varepsilon_t)_{t\in\mathbb{Z}}$ est un bruit blanc de variance σ_ε^2.

1. La représentation ARMA est - elle minimale ?
2. Montrer que
$$X_t = \varepsilon_t + \sum_{j\geq 1} \frac{1}{2^{j+1}}\varepsilon_{t-j} \tag{9.75}$$
3. Calculer la fonction d'autocorrélation $\gamma(.)$ du processus.
4. Lorsque $(\varepsilon_t)_{t\in\mathbb{Z}}$ est un bruit blanc gaussien, déterminer la loi du couple (X_t, X_{t+1})
5. Donner une expression de la projection orthogonale de X_t sur le sous-espace vectoriel fermé de $\mathbb{L}^2$ engendré par $X_{t-1}, X_{t-2}, \ldots$.

9.2 Identification et estimation des paramètres

9.2.1 Identification des paramètres p et q

L'étape d'identification ou de la spécification d'un processus ARMA(p,q) consiste à observer la nature des fonctions empiriques d'autocorrélation et d'autocorrélation partielle, de la série-elle, pour choisir les ordres des parties AR (choix de l'entier p) et MA (choix de l'entier q). Le choix des entiers p et q se fait à l'aide de l'ACF empirique et la PACF empirique.

Theorème 9.1 *Ordre p : Théorème de Quenouille*

Soit $(X_t)_{t\in\mathbb{Z}}$ un processus AR(p) stationnaire gaussien. Sous l'hypothèse nulle $H_0 : \phi_{hh} = 0$ pour tout $h \geq p+1$, on a quand $T \longrightarrow +\infty$

$$\hat{\phi}_{hh}\sqrt{T} \longrightarrow \mathcal{N}(0,1) \tag{9.76}$$

Ainsi, pour tester la nullité des autocorrélations partielles, on compare la statistique

$$t_{\widehat{\phi}_{hh}} = \frac{\hat{\phi}_{hh}}{\sqrt{1/T}} \tag{9.77}$$

et la valeur critique d'une loi de Student à $T-k$ degré de liberté où k est le nombre de paramètre à estimer. On applique la règle de décision suivante :

— $|t_{\widehat{\phi}_{hh}}| < t_{1-\alpha/2}$, on accepte l'hypothèse H_0
— $|t_{\widehat{\phi}_{hh}}| \geq t_{1-\alpha/2}$, on rejette l'hypothèse H_0.

Sous l'hypothèse $H_0 : \phi_{hh} = 0$, l'intervalle de confiance de $\hat{\phi}_{hh}$, au risque de premier espèce $\alpha = 5\%$ est donnée par :

$$\hat{\phi}_{hh} \in \left[0 \pm 1.96\frac{1}{\sqrt{T}}\right] \tag{9.78}$$

Donc ce théorème nous permet d'identifier l'ordre p tout en sachant $\phi_{hh} = 0$ au-delà de p i.e. pour $h \geq p+1$.

Theorème 9.2 *Ordre q : Test de Bartlett*

Soit $(X_t)_{t\in\mathbb{Z}}$ un processus $MA(q)$ stationnaire gaussien. Sous l'hypothèse nulle $H_0 : \rho(h) = 0$ pour $h > q$ (i.e. que les autocorrélations ne sont pas significativement différent de zéro), on a quand $T \longrightarrow +\infty$

$$\sqrt{T}\hat{\rho}(h) \longrightarrow \mathcal{N}\left(0, 1 + 2\sum_{i=1}^{q}\hat{\rho}^2(i)\right) \tag{9.79}$$

Pour tester l'hypothèse H_0, on utilise le test de Student en comparant la statistique

$$t_{\hat{\rho}(h)} = \frac{\hat{\rho}(h)}{\hat{\sigma}_{\hat{\rho}(h)}} \tag{9.80}$$

avec $\hat{\sigma}^2_{\hat{\rho}(h)} = \frac{1}{T}\left(1 + 2\sum_{i=1}^{h-1}\hat{\rho}^2_X(i)\right)$ et la valeur critique d'une loi de Student à $T-q$ degrés de liberté.

Le test est bilatéral symétrique, la région de rejet de H_0 est $\left]-\infty, -t_{1-\alpha/2}\right] \cup \left[t_{1-\alpha/2}, +\infty\right[$. On a la règle de décision suivante :

- $|t_{\hat{\rho}(h)}| < t_{1-\alpha/2}$, on accepte l'hypothèse $H_0 : \rho(h)$ n'est pas significatif.
- $|t_{\hat{\rho}(h)}| \geq t_{1-\alpha/2}$, on rejette l'hypothèse $H_0 : \rho(h)$ est significatif.

Sous l'hypothèse $H_0 : \rho_X(h) = 0$, l'intervalle de confiance de $\hat{\rho}(h)$, au risque de premier espèce $\alpha = 5\%$ est donnée par :

$$\hat{\rho}_X(h) \in \left[0 \pm 1.96\sqrt{\frac{1}{T}\left(1 + 2\sum_{i=1}^{h-1}\hat{\rho}^2_X(i)\right)}\right] \tag{9.81}$$

Donc ce théorème nous permet d'identifier l'ordre q tout en sachant $\rho(h) = 0$ au-delà de q i.e. pour tout $h \geq q+1$.

Ces intervalles permettent de décider empiriquement quel est l'ordre p pour un processus AR(p) et quel est l'ordre q pour un processus MA(q). Ces ordres correspondent aux valeurs de h pour lesquelles l'ACF empirique et le PACF empirique se trouvent à l'extérieur des intervalles de confiance.

En pratique, lorsque l'on doit ajuster un modèle AR, MA ou ARMA à des données réelles le première question qui se pose est celle du choix des ordres p et q du modèle ARMA. Nous pouvons exploiter les résultats suivants :

Table 9.1 – Récapitulatif

Type de processus	Autocorrélations	Autocorrélations partielles
AR(p)	$\rho(h) \searrow 0$	$\phi_{hh} = 0$ pour $h \geq p+1$
MA(q)	$\rho(h) = 0$ pour $h \geq q+1$	ϕ_{hh} rien en particulier
ARMA(p,q)	$\rho(h) \searrow 0, h \geq q+1$	$\phi_{hh} \searrow 0, h \geq \max(q+1, p)$

Ses résultats sont facilement exploitables dans le cas d'un AR ou MA pur. Dans le cas d'un ARMA il existe un moyen de déterminer les ordres p et q à l'aide de la méthode du coin basée sur certains déterminant de matrices de corrélations que nous ne développons pas ici. Une autre

approche est de considérer un ensemble crédible de modèles ARMA(p,q) puis par de sélectionner un candidat par une méthode de sélection de modèle basée sur des critères d'information de type BIC ou AIC.

9.2.2 Estimation des paramètres d'un AR(p)

On dispose d'une observation $(x_0, \dots, x_T)$ d'un processus $(X_t)_t$ supposé suivre un modèle AR(p), c'est-à-dire vérifiant :

$$X_t = \mu + \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \varepsilon_t, \quad t \in \mathbb{Z} \tag{9.82}$$

avec $X_0 = x_0$ et $\mu, \phi_1, \dots, \phi_p$ des paramètres inconnus et $\varepsilon_t \sim BB(0, \sigma_\varepsilon^2)$. On cherche alors à estimer ces paramètres à l'aide des observations disponibles. Pour cela, différentes méthodes sont possibles : moindres carrés conditionnels, maximum de vraisemblance et méthodes des moments.

9.2.2.1 Méthode des moments

La méthode des moments est basée sur les équations de Yule - Walker. Elle consiste à reprendre les équations de Yule - Walker en inversant les relations : on exprime les coefficients en fonction des autocovariances. On applique alors le raisonnement de la méthode des moments : on trouve les paramètres estimés d'après les auto-covariances estimées.

On a, pour tout $h = 1, \dots, p$, l'équation de Yule - Walker

$$\begin{cases} \displaystyle\sum_{i=1}^{p} \phi_i \rho(h-i) = \rho(h) \\ \sigma_X^2 = \displaystyle\sum_{i=1}^{p} \rho_i \gamma(i) + \sigma_\varepsilon^2 \end{cases} \tag{9.83}$$

où σ_X^2 est la variance du processus $(X_t)_{t\in\mathbb{Z}}$, égale à $\gamma(0)$ et σ_ε^2 est la variance du processus bruit blanc $(\varepsilon_t)_{t\in\mathbb{Z}}$.

En écriture matricielle, l'équation de Yule - Walker peut se mettre sous la forme suivante :

$$\begin{pmatrix} 1 & \rho(1) & \rho(2) & \cdots & \rho(p-1) \\ \rho(1) & 1 & \rho(2) & \cdots & \rho(p-2) \\ \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ \rho(p-1) & \rho(p-2) & \cdots & \cdots & 1 \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \vdots \\ \phi_p \end{pmatrix} = \begin{pmatrix} \rho(1) \\ \rho(2) \\ \vdots \\ \vdots \\ \rho(p) \end{pmatrix} \tag{9.84}$$

qui peut se réécrire :

$$\Gamma \phi^t = \rho^t \tag{9.85}$$

où Γ est la matrice de variance - covariance normée du processus $(X_t)_{t\in\mathbb{Z}}$. Γ est symétrique et est à valeurs propres $\lambda_i > 0$, donc Γ est définie positive. Γ étant une matrice définie positive, Γ est donc régulière et admet donc une inverse Γ^{-1},

$$\Gamma \phi^t = \rho^t \quad \Longrightarrow \quad \phi^t = \Gamma^{-1} \rho^t \tag{9.86}$$

Donc un estimateur de $\phi = \big(\phi_1, \phi_2, \dots, \phi_p\big)$, noté $\widehat{\phi}$, est donné par :

$$\hat{\phi}^t = \hat{\Gamma}^{-1}\hat{\rho}^t \tag{9.87}$$

La variance résiduelle $\hat{\sigma}_\varepsilon^2$ et la constante $\hat{\mu}$ sont alors données par les équations :

$$\begin{cases} \hat{\sigma}_\varepsilon^2 = \hat{\sigma}_X^2 - \sum_{i=1}^{j=p} \hat{\phi}\hat{\gamma}(i) \\ \hat{\mu} = \left(1 - \sum_{j=1}^{j=p} \hat{\phi}_j\right) \overline{X} \end{cases} \tag{9.88}$$

Proposition 9.6

La distribution des estimateurs de Yule - Walker $\hat{\phi}$ du modèle paramétrique d'un processus AR(p) est asymptotiquement normale :

$$\sqrt{T}\left(\hat{\phi} - \phi\right) \xrightarrow{d} \mathcal{N}\left(0, \sigma^2 \Gamma_p^{-1}\right) \tag{9.89}$$

et

$$\hat{\sigma}_\varepsilon^2 \xrightarrow{d} \sigma_\varepsilon^2 \tag{9.90}$$

Sous Python, la fonction yule_walker() permet d'effectuer une estimation des paramètres d'un processus AR(p) en se basant sur les équations de Yule- Walker.

```
# Estimation des paramètres d'un AR(p) par Yule - Walker
import statsmodels.api as sm
phi, sigma = sm.regression.yule_walker(default_data,order=p,method="mle")
```

Exemple 9.20

On observe une trajectoire $(x_1, \dots, x_T)$ d'un processus $(X_t)_t$. On souhaite le modéliser par un AR(2). On cherche donc à estimer les paramètres ϕ_1 et ϕ_2.

1. Écrire les équations de Yule - Walker correspondantes et les estimateurs qui en découlent.
2. A partir des observations, on calcule les autocovariances empiriques suivantes : $\hat{\gamma}(0) = 1.774$, $\hat{\gamma}(1) = 1.186$ et $\hat{\gamma}(2) = 0.692$. En déduire une estimation de ϕ_1 et ϕ_2.

9.2.2.2 Moindres carrés conditionnels

Une autre méthode d'estimation des processus AR(p) est celle des moindres carrés conditionnels. Si ε_t est gaussien, le vecteur des observations $(X_1, \dots, X_T)$ est gaussien et on peut alors calculer sa vraisemblance. Une approche simplifiée est de calculer la densité de $(X_{p+1}, \dots, X_T)$ conditionnellement à $(X_1, \dots, X_p)$ qui est gaussienne. Sachant que $X_t / X_{t-1}, \dots, X_{t-p}$ est gaussienne d'espérance $\sum_{j=1}^{j=p} \phi_j X_{t-j}$ est de variance σ^2, on a, en utilisant la formule des conditionnements successifs :

$$f\left(X_{p+1},\dots,X_T|X_1,\dots,X_p\right) = \prod_{t=p+1}^{t=T} f(X_t|X_{t-1},\dots,X_{t-p})$$
$$= \frac{1}{(2\pi\sigma^2)^{\frac{T-p}{2}}} \exp\left\{-\frac{1}{2\sigma^2}\sum_{t=p+1}^{t=T}\left(X_t-\mu-\sum_{j=1}^{p}\phi_j X_{t-j}\right)^2\right\}$$

Maximiser cette vraisemblance conditionnelle en ϕ revient à, en passant au log, minimiser :

$$\left(\phi_1,\dots,\phi_p,\sigma^2\right) \longrightarrow \frac{1}{\sigma^2}\sum_{t=p+1}^{t=T}\left(X_t-\mu-\sum_{j=1}^{p}\phi_j X_{t-j}\right)^2 + (T-p)\ln\left(\sigma^2\right) \tag{9.91}$$

à σ^2 fixé, cela correspond au critère des moindres carrés.

9.2.3 Estimation des paramètres d'un ARMA(p,q)

L'estimation des paramètres d'un processus ARMA repose sur la méthode du maximum de vraisemblance en supposant que les résidus suivent une loi normale de moyenne nulle et de variance σ^2. L'intérêt de cette méthode est qu'elle utilise toute l'information à court comme à terme concernant les comportements des séries puisque sont estimés simultanément les paramètres autorégressif et moyenne mobile. La méthode s'inscrit dans le domaine temporel.

Notons $\beta = (\mu, \phi_1, \dots, \phi_p, \theta_1, \dots, \theta_q, \sigma_\varepsilon^2)'$ le paramètre à estimer. La méthode préconisée par Box et Jenkins (1976, p.211) conditionne la vraisemblance du processus sur les p premières valeurs observées, $X_1,\dots,X_p$, du processus $(X_t)_{t\in\mathbb{Z}}$; et sur les q valeurs du processus gaussien $(\varepsilon_t)_{t\in\mathbb{Z}}$, telles que :

$$\varepsilon_p = \varepsilon_{p-1} = \cdots = \varepsilon_{p-q+1} = 0 \tag{9.92}$$

Ainsi à partir de la suite $X_1,\dots,X_T$, on peut alors calculer par itération la suite $\varepsilon_{p+1},\varepsilon_{p+2},\dots,\varepsilon_T$, de la manière suivante, pour tout $t = p+1,\dots,T$:

$$\varepsilon_t = X_t - \phi_1 X_{t-1} - \cdots - \phi_p X_{t-p} + \theta_1\varepsilon_{t-1} + \cdots + \theta_q\varepsilon_{t-q} \tag{9.93}$$

On suppose que les perturbations sont gaussiennes conditionnellement aux $X_t, t \le p$, alors X_t, pour tout $t = p+1,\dots,T$ est normalement distribué avec une moyenne égale à $X_t - \phi_1 X_{t-1} - \cdots - \phi_p X_{t-p} + \theta_1\varepsilon_{t-1} + \cdots + \theta_q\varepsilon_{t-q}$ et une variance égale à σ_ε^2.

La log-vraisemblance conditionnelle est alors donnée par l'équation suivante :

$$\mathcal{L}(\beta) = \log f\left(X_T,\dots,X_{p+1}|X_p,\dots,X_1,\varepsilon_p=\cdots=\varepsilon_{p-q+1}=0\right)$$
$$= \log\prod_{t=p+1}^{t=T} f\left(X_t|\mathcal{F}_{t-1}\right) \quad \text{où } \mathcal{F}_{t-1} = \sigma\left(X_{t-1},\dots,X_{p+1},X_p,\dots,X_1,\varepsilon_p=\cdots=\varepsilon_{p-q+1}=0\right)$$
$$= -\frac{T-p}{2}\log(2\pi) - \frac{T-p}{2}\log\left(\sigma_\varepsilon^2\right) - \sum_{t=p+1}^{t=T}\frac{\varepsilon_t^2}{2\sigma_\varepsilon^2}$$

L'estimateur du maximum de vraisemblance (EMV), noté $\hat{\beta}_{\text{EMV}}$, est le paramètre qui maximise la log - vraisemblance, i.e. :

$$\hat{\beta}_{\text{EMV}} = \arg\max_{\beta} \mathcal{L}(\beta) \tag{9.94}$$

En pratique, les estimateurs du maximum de vraisemblance sont trouvés numériquement par des algorithmes itératifs : on cherche le maximum de la fonction de vraisemblance numériquement (par exemple, par la méthode de Newton-Raphson), démarrant d'une valeur initiale pour le vecteur du paramètre recherché. Cette valeur initiale peut être fournie par les estimateurs de Yule - Walker ou par les estimateurs des moindres carrés. Sous Python, la fonction ARIMA permet l'estimation des coefficients d'un modèle ARMA.

```
# Estimation des paramètres d'un ARMA(p,q)
from statsmodels.tsa.arima.model import ARIMA
model = ARIMA(default_data,order=(p,0,q)).fit()
print(model.summary())
```

Remarque 9.5 *Estimation des processus ARMA creux*

Les ARMA creux ou percés sont estimés de la manière suivante. Supposons que l'on dispose du processus $ARMA(3,4)$ suivant :

$$(I - \phi_1 B - \phi_3 B^3) X_t = (I - \theta_1 B - \theta_4 B^4) \varepsilon_t \tag{9.95}$$

Pour la partie AR, on crée un vecteur a dont la longueur correspond au retard p maximum. Ce vecteur contient des éléments $a_1, \dots, a_p$ qui prennent la valeur 1 si le coefficient ϕ_i doit être estimé et 0 sinon. Le raisonnement est le même pour la partie MA.

```
# Estimation d'un ARMA creux
model = ARIMA(default_data,order=((1,0,1),0,(1,0,0,1))).fit()
print(model.summary())
```

Remarque 9.6

- La méthode des MCO est inapplicable pour les modèles MA(p) et ARMA(p,q), car ε_{t-q}, variable aléatoire explicative du modèle, est inobservable.
- La méthode des MCO est applicable pour les modèles AR(p) car les termes d'erreurs sont non corrélés et la variance non conditionnelle constante mais toutefois elle reste moins performante que l'estimateur de la vraisemblance maximale.

Exemple 9.21 *Application sur données réelles : Indice des prix à la consommation*

On considère la série des indices des prix à la connsomation. Cette série est disponible dans la librairie Statsmodels et va du premier trimestre 1951 au troisième trimestre 2009. Dans le cadre de notre exemple, nous travaillerons sur la série en différence première.

```
# Chargement des données
import statsmodels.api as sm
import scipy.stats as st
import pandas as pd
macrodta = sm.datasets.macrodata.load_pandas().data
macrodta.index = pd.Index(sm.tsa.datetools.dates_from_range("1959Q1", "2009Q3"))
cpi = macrodta["cpi"]
cpi.index = pd.date_range(start="1959-03-31",periods=len(cpi),freq="Q")
# Série en différence première
cpi2 = cpi.diff().dropna()
# Représentation graphique
fig, (axe1, axe2) = plt.subplots(1,2,figsize=(16,6))
axe1.plot(cpi,color = "black");
axe1.set_xlabel("année");
axe1.set_ylabel("cpi");
axe2.plot(cpi2,color = "black");
axe2.set_xlabel("année");
axe2.set_ylabel("cpi2");
plt.tight_layout();
plt.show()
```

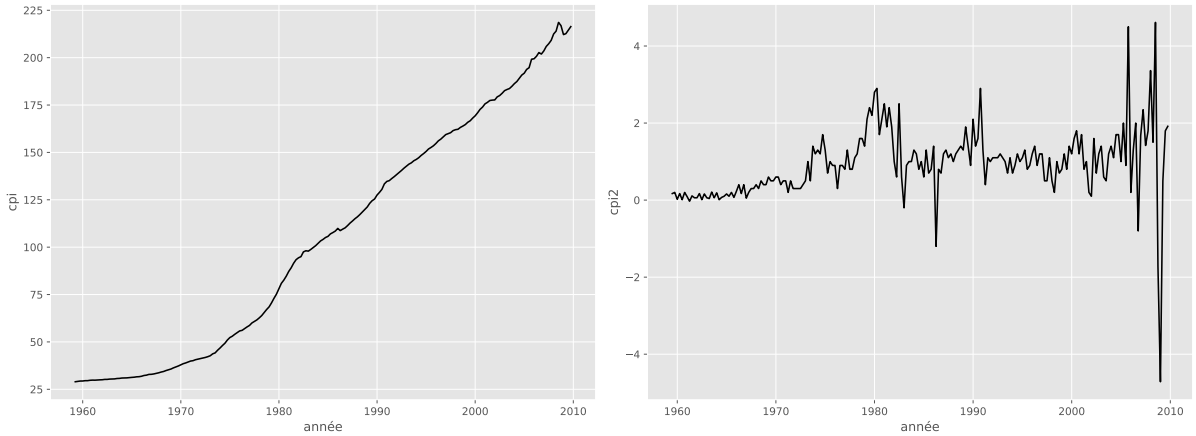


Figure 9.13 – Evolution de la série index de prix à la consommation

Nous calculons les autocorrélations simples.

```
# Calcul des autocorrélations simples
from mizani.formatters import scientific_format
acf, qstat, pvalue = sm.tsa.acf(cpi2,adjusted=False,nlags=12,qstat=True)
pacf = sm.tsa.pacf(cpi2,nlags=12)
data = np.c_[acf[1:], pacf[1:], qstat, pvalue].T
table = pd.DataFrame(data,index=["ACF","PACF","Q - Stat", "Prob(>Q)"],
                     columns = list(range(1,13)))
```

Les `Q - Stat` sont celles de Ljung - Box et `Prob(>Q)` sont les probabilités associées. Les p-values associées sont largement inférieures à la valeur critique de 5%. Ce qui signifie que les valeurs de la série `cpi` ne sont pas indépendantes.

Nous représentons graphiquement les autocorrélogrammes simples et partiels.

Table 9.2 – Autocorrélations simples et partielles de la série cpi

	1	2	3	4	5	6	7	8	9	10	11	12
ACF	0.38	0.22	0.29	0.16	0.27	0.27	0.17	0.21	0.27	0.14	0.23	0.10
PACF	0.39	0.09	0.21	-0.03	0.22	0.08	0.03	0.06	0.16	-0.09	0.15	-0.17
Q - Stat	30.25	40.65	57.94	63.15	78.43	93.22	99.62	109.26	125.31	129.35	140.90	142.93
Prob(>Q)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

```
# Autocorrélogramme simple et partielle
fig, (axe1, axe2) = plt.subplots(1,2,figsize=(16,6))
plot_acf(cpi2, ax = axe1);
axe1 = plot_colors(axe1);
axe1.set_ylabel("ACF");
axe1.set_xlabel("Lag");
plot_pacf(cpi2, ax = axe2);
axe2 = plot_colors(axe2);
axe2.set_ylabel("PACF");
axe2.set_xlabel("Lag");
plt.tight_layout();
plt.show()
```

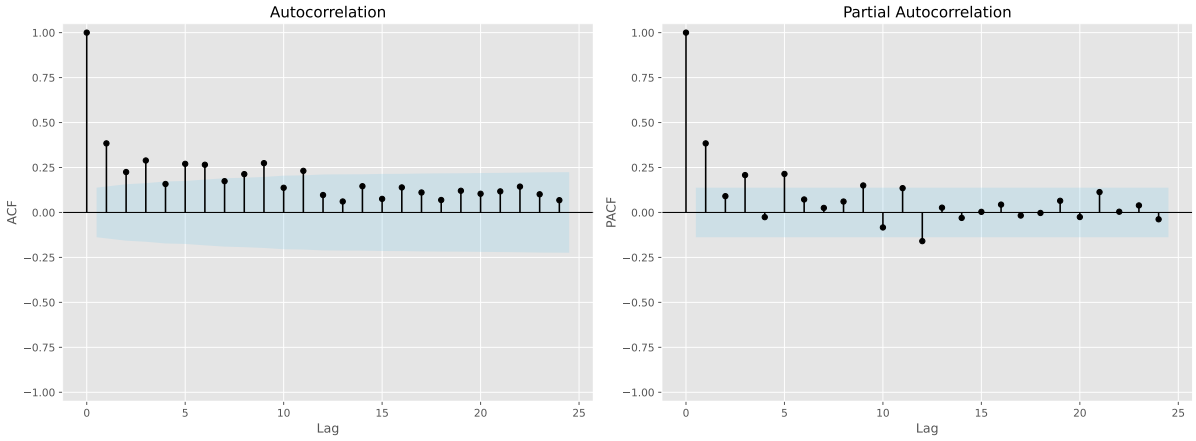


Figure 9.14 – Autocorrélogrammes simples et partiels de la série cpi

Le graphique et les autocorrélogrammes simples et partielles montrent que la série cpi2 semble non stationnaire. Nous supposons par la suite que cette série est stationnaire c'est - à - dire qu'elle est peut être modélisée par une représentation $ARMA(p, q)$.

Nous estimerons les paramètres des modèles suivants : $AR(1)$, $AR(2)$, $MA(2)$, $MA(4)$, $ARMA(1, 1)$ et $ARMA(1, 2)$.

— Estimation des coefficients du modèle $AR(1)$: $cpi_t = \mu + \phi_1 cpi_{t-1} + \varepsilon_t$

```
# Estimation d'un AR(1)
from statsmodels.tsa.arima.model import ARIMA
model1 = ARIMA(cpi2,order=(1,0,0),trend="c").fit()
model1_params = extractParams(model1, model_type = "arima")
```

— Estimation des coefficients du modèle $AR(2)$: $cpi_t = \mu + \phi_1 cpi_{t-1} + \phi_2 cpi_{t-2} + \varepsilon_t$

Table 9.3 – Coefficients du modèle AR(1) : $cpi_t = \mu + \phi_1 cpi_{t-1} + \varepsilon_t$

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.9284	0.0959	9.6803	3.656e-22	0.7405	1.1164
ar.L1	0.3859	0.0281	13.7108	8.754e-43	0.3307	0.4411
sigma2	0.6759	0.0281	24.0707	5.067e-128	0.6208	0.7309

```
# Estimation d'un AR(2)
model2 = ARIMA(cpi2,order=(2,0,0),trend="c").fit()
model2_params = extractParams(model2, model_type = "arima")
```

Table 9.4 – Coefficients du modèle AR(2) : $cpi_t = \mu + \phi_1 cpi_{t-1} + \phi_2 cpi_{t-2} + \varepsilon_t$

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.9288	0.1191	7.8012	6.133e-15	0.6954	1.1621
ar.L1	0.3504	0.0286	12.2676	1.351e-34	0.2944	0.4064
ar.L2	0.0933	0.0367	2.5382	1.114e-02	0.0212	0.1653
sigma2	0.6700	0.0295	22.7255	2.510e-114	0.6122	0.7277

— Estimation des coefficients du modèle $MA(2) : cpi_t = \mu + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2}$

```
# Estimation d'un MA(2)
model3 = ARIMA(cpi2,order=(0,0,2),trend="c").fit()
model3_params = extractParams(model3, model_type = "arima")
```

Table 9.5 – Coefficients du modèle MA(2) : $cpi_t = \mu + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2}$

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.9281	0.0913	10.1657	2.821e-24	0.7491	1.1070
ma.L1	0.3593	0.0348	10.3273	5.303e-25	0.2911	0.4275
ma.L2	0.0515	0.0520	0.9914	3.215e-01	-0.0503	0.1533
sigma2	0.6890	0.0303	22.7104	3.533e-114	0.6295	0.7484

— Estimation des coefficients du modèle $MA(4) : cpi_t = \mu + \varepsilon_t - \sum_{j=1}^{j=4} \theta_j \varepsilon_{t-j}$

```
# Estimation d'un MA(4)
model4 = ARIMA(cpi2,order=(0,0,4),trend="c").fit()
model4_params = extractParams(model4, model_type = "arima")
```

Table 9.6 – Coefficients du modèle MA(2) : $cpi_t = \mu + \varepsilon_t - \sum_{j=1}^{j=4} \theta_j \varepsilon_{t-j}$

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.9330	0.1177	7.9278	2.231e-15	0.7023	1.1636
ma.L1	0.4095	0.0355	11.5315	9.158e-31	0.3399	0.4791
ma.L2	0.1211	0.0672	1.8023	7.150e-02	-0.0106	0.2528
ma.L3	0.2544	0.0589	4.3194	1.564e-05	0.1390	0.3699
ma.L4	-0.0756	0.0871	-0.8683	3.852e-01	-0.2463	0.0951
sigma2	0.6388	0.0329	19.4316	4.170e-84	0.5744	0.7033

— Estimation des coefficients du modèle $ARMA(1,1) : cpi_t = \mu + cpi_{t-1} + \varepsilon_t - \theta_1 \varepsilon_{t-1}$

```
# Estimation d'un ARMA(1,1)
model5 = ARIMA(cpi2,order=(1,0,1),trend="c").fit()
model5_params = extractParams(model5, model_type = "arima")
```

Table 9.7 – Coefficients du modèle ARMA(1,1) : $cpi_t = \mu + \theta 1 cpi_{t-1} + \varepsilon_t - \theta_1 \varepsilon_{t-1}$

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.8809	0.3875	2.2730	2.303e-02	0.1213	1.6404
ar.L1	0.9399	0.0566	16.5972	7.297e-62	0.8289	1.0509
ma.L1	-0.7710	0.0693	-11.1277	9.201e-29	-0.9068	-0.6352
sigma2	0.6398	0.0274	23.3556	1.209e-120	0.5861	0.6935

— Estimation des coefficients du modèle $ARMA(1,2)$: $cpi_t = \mu + \phi_1 cpi_{t-1} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2}$

```
# Estimation d'un ARMA(1,2)
model6 = ARIMA(cpi2,order=(1,0,2),trend="c").fit()
model6_params = extractParams(model6, model_type = "arima")
```

Table 9.8 – Coefficients du modèle ARMA(1,2) : $cpi_t = \mu + \phi_1 cpi_{t-1} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2}$

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.8600	0.4765	1.8050	7.107e-02	-0.0738	1.7939
ar.L1	0.9686	0.0536	18.0622	6.326e-73	0.8635	1.0737
ma.L1	-0.6679	0.0650	-10.2798	8.692e-25	-0.7952	-0.5405
ma.L2	-0.1797	0.0431	-4.1670	3.087e-05	-0.2642	-0.0952
sigma2	0.6251	0.0273	22.9209	2.875e-116	0.5716	0.6785

Remarque 9.7

Sous R, la fonction Arima de la librairie forecast estime le modèle autorégressif moyenne mobile $ARMA(p,q)$ suivant :

$$X_t = \mu + \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \varepsilon_{t-q} \tag{9.96}$$

Estimons les coefficients du modèle $ARMA(1,2)$

```
library(forecast)
library(caschrono)
model <- forecast::Arima(as.vector(py$cpi2),order = c(1,0,2))
res = t(rbind("coef" = coef(model),t_stat(model),t(confint(model))))
print(res)
```

```
##                 coef    t.stat    p.val      2.5 %      97.5 %
## ar1        0.9686186 31.147778 0.000000  0.9076686  1.02956862
## ma1       -0.6678673 -7.994486 0.000000 -0.8316046 -0.50412995
## ma2       -0.1797765 -2.128091 0.033330 -0.3453499 -0.01420304
## intercept  0.8604084  3.454386 0.000552  0.3722263  1.34859045
```

On remarque que, pour le modèle $ARMA(1,2)$, les coefficients sont identiques à ceux obtenus avec la fonction ARIMA de Statsmodels. Cependant, les pvalues sont différentes. On remarque qu'au seuil de 5%, la constante est significative sous R, mais non significative sous Python.

Cercle unité

Nous implémentons une fonction graphique qui retourne le cercle unité basé sur les racines du polynôme du processus AR ou MAR.

```
# Cercle unité
def plotroot(root,title= "AR roots",ax=None):
    from matplotlib.patches import Ellipse
    real = np.real([1/x for x in root])
    im = np.imag([1/x for x in root])
    # Représentation graphique
    if ax is None:
        ax = plt.gca()
    ax.scatter(real,im,color = "blue");
    ellipse=Ellipse((0,0),width=4,height=4,facecolor="none",edgecolor = "black");
    ax.add_patch(ellipse);
    ax.set_title(title);
    ax.set_xlabel("Re(1/root)");
    ax.set_ylabel("Im(1/root)");
    ax.axvline(x=0,linewidth=2, color='black');
    ax.axhline(y=0,linewidth=2, color='black');
# Racine du polynôme AR
print(f"""Racines du polynôme : - AR : {model6.arroots} - MA : {model6.maroots}""")
```

```
## Racines du polynôme : - AR : [1.03243983] - MA : [ 1.14470968 -4.86110587]
```

```
# Application au modèle ARMA(1,2)
fig, (axe1,axe2) = plt.subplots(1,2,figsize=(16,7))
plotroot(model6.arroots,ax=axe1);
plotroot(model6.maroots,ax=axe2,title= "MA roots");
plt.tight_layout();
plt.show()
```

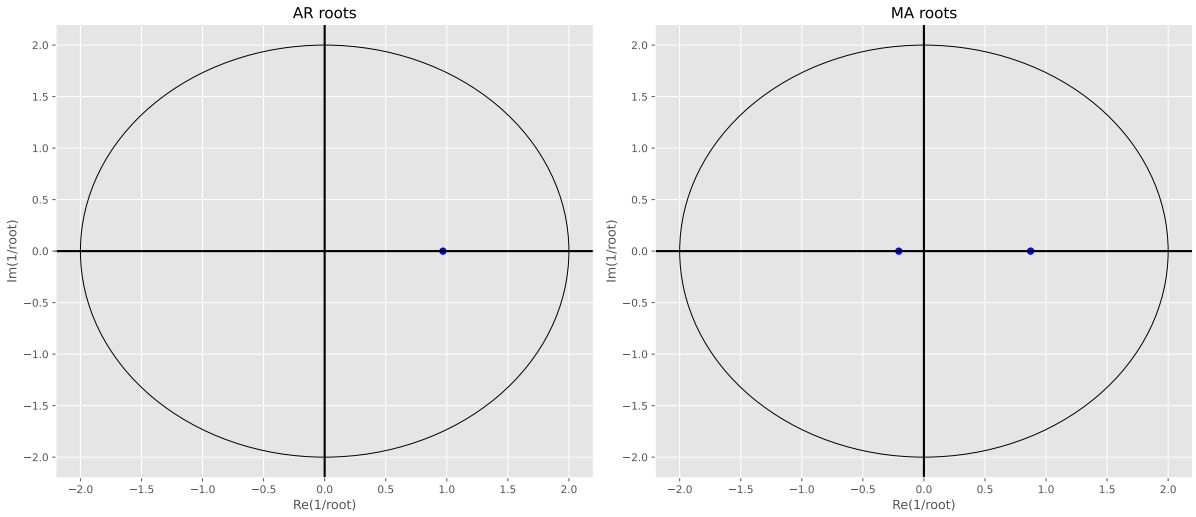


Figure 9.15 – Racines caractéristiques inverses du modèle ARMA(1,2) ajustées à l'indice des prix à la consommation

10

Validation et prévision des processus ARMA

Sommaire

Une fois l'étape d'estimation achevée, l'étape suivante consiste à valider le(s) modèle(s) estimé(s). Lorsque plusieurs modèles passent à l'étape d'estimation, on peut suivant l'objectif les départager avec d'autres critères. On peut par exemple comparer la qualité prédictive ou la significativité des paramètres estimés.

10.1 Validation des processus

Au début de cette étape, on dispose de plusieurs processus ARMA dont on a estimé les paramètres. Il faut maintenant valider ces modèles afin de les départager. Pour cela, on applique des tests sur les paramètres (ils doivent être significativement différent de 0) et d'une analyse sur les résidus estimés permettant de tester les hypothèses effectuées sur le processus d'innovation $(\varepsilon_t)_{t\in\mathbb{Z}}$ (les résidus estimés doivent suivre un processus de bruit blanc : $e_t \sim BB\left(0, \sigma_\varepsilon^2\right)$ où e_t est l'estimateur de l'erreur ε_t puisque l'on a supposé que $\varepsilon_t \sim BB\left(0, \sigma_\varepsilon^2\right)$ lors de la définition du processus $ARMA(p,q)$). Si plusieurs modèles sont valide, l'étape de validation doit se poursuivre par une comparaison des qualités de ces derniers

10.1.1 Tests sur les paramètres

Le premier test que l'on peut mener consiste à tester l'hypothèse nulle $p^{'} = p-1$ et $q^{'} = q$. On regarde si l'on peut diminuer d'une unité le nombre de retards intervenant dans la partie AR. En d'autres termes, on teste l'hypothèse nulle de processus $ARMA(p-1,q)(\phi_p = 0)$ contre l'hypothèse alternative de processus $ARMA(p,q)(\phi_p \neq 0)$.

Ce test est très simple à mettre en oeuvre puisqu'il s'agit d'un test de significativité usuel sur le coefficient ϕ_p. On calcule donc la statistique de Student du coefficient

$$t_{\widehat{\phi}_p} = \frac{\hat{\phi}_p}{\hat{\sigma}_{\widehat{\phi}_p}} \tag{10.1}$$

que l'on compare à la valeur critique lue dans la table de la loi de Student. On applique la règle de décision suivante :

— Si $|t_{\widehat{\phi}_p}| < t_{1-\alpha/2}$, on accepte l'hypothèse nulle de processus ARMA(p-1, q).
— Si $|t_{\widehat{\phi}_p}| \leq t_{1-\alpha/2}$, l'hypothèse nulle est rejetée et in retient un processus ARMA(p, q)

où $t_{1-\alpha/2}$ est le quantile d'ordre $(1-\alpha/2)$ de la loi de Student à $(T-h)$ degrés de liberté et $h = p+q+1$ étant le nombre de paramètres estimés, et

$$\hat{\sigma}^2_{\widehat{\phi}_p} = \frac{\hat{\sigma}^2_\varepsilon}{\sum_{t=1}^{t=T}(X_t - \overline{X})^2} \quad \text{avec} \quad \hat{\sigma}_\varepsilon = \frac{1}{T-h}\sum_{t=1}^{t=T}\hat{\varepsilon}_t^2 \tag{10.2}$$

Sous l'hypothèse nulle $H_0 : \phi_p = 0$, pour le modèle, l'intervalle de confiance de $\hat{\phi}_p$, au risque de premier espèce $\alpha = 5\%$ est donnée par :

$$\widehat{\phi_p} \in \left[0 \pm 1.96\hat{\sigma}_{\widehat{\phi}_p}\right] \tag{10.3}$$

On peut appliquer un raisonnement similaire au test de l'hypothèse nulle $p^{'} = p$ et $q^{'} = q-1$. De façon symétrique, il est possible de mener un deuxième test de l'hypothèse nulle $p^{'} = p+1$ et $q^{'} = q$. On regarde cette fois-ci s'il faut introduire un retard supplémentaire dans partie autorégressive.

On compare la valeur absolue de chacun des paramètres estimés avec sa variance. Ainsi, si la valeur absolue du paramètre est plus grande que $1.96 \times$ ecart-type du paramètre, alors on rejette, au risque de $\alpha = 0.05$, l'hypothèse de nullité du paramètre.

Attention, on ne peut pas tester simultanément $p^{'} = p+1$ et $q^{'} = q+1$ (ou $p^{'} = p-1$ et $q^{'} = q-1$) car tout processus admettant une représentation $ARMA(p,q)$ admet également une représentation $ARMA(p+1, q+1)$.

Exemple 10.1 *Test sur les paramètres du modèle ARMA(1,2)*

Table 10.1 – Coefficients du modèle ARMA(1,2) : $cpi_t = \mu + \phi_1 cpi_{t-1} + \varepsilon_t - \theta_1\varepsilon_{t-1} - \theta_2\varepsilon_{t-2}$

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.8600	0.4765	1.8050	7.107e-02	-0.0738	1.7939
ar.L1	0.9686	0.0536	18.0622	6.326e-73	0.8635	1.0737
ma.L1	-0.6679	0.0650	-10.2798	8.692e-25	-0.7952	-0.5405
ma.L2	-0.1797	0.0431	-4.1670	3.087e-05	-0.2642	-0.0952
sigma2	0.6251	0.0273	22.9209	2.875e-116	0.5716	0.6785

Au seuil de 5%, tous les coefficients sont significatifs, sauf la constante.

Nous pouvons par la suite effectuer une test de Wald de nullité de la constante. Ceci s'effectue grâce à la fonction res.wald_test() en y incorporant la matrice R de format $(N \times (p+q+1))$ où N est le nombre de restrictions ou de contraintes à imposer.

Comme on le verra dans les prochains chapitres, le test de Wald consiste à tester l'hypothèse nulle $R\beta = c$ contre l'hypothèse alternative $R\beta \neq c$ avec R une matrice de format $(N \times (p+q+1))$ et $c = 0$. La statistique du test s'écrit :

$$\lambda_W = T\beta' R' \left[R\widehat{\Sigma}_\beta R'\right] R\beta \tag{10.4}$$

où $\widehat{\Sigma}_\beta$ est l'estimateur du maximum de vraisemblance de Σ_β. Sous H_0, cette statistique suit une loi de χ^2 à N degrés de liberté.

Exemple 10.2 *Test de Wald de nullité de la constante du modèle $ARMA(1,2)$*

Dans notre exemple, $\beta = (\mu, \phi_1, \theta_1, \theta_2, \sigma_\varepsilon^2)'$ et la matrice R s'écrit :

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{10.5}$$

```
# Test de Wald
from mizani.formatters import scientific_format
R =  np.array([1,0,0,0,0])
wald = model6.wald_test(R,use_f = False)
wald_res = pd.DataFrame(np.c_[wald.statistic,wald.df_denom,wald.pvalue],
                  columns = ["statistic","ddl","p-value"],index = ["Wald test"])
wald_res.iloc[:,2] = scientific_format(digits=3)(wald_res.iloc[:,2])
```

Table 10.2 – Test de Wald de nullité de la constante du modèle $ARMA(1,2)$

	statistic	ddl	p-value
Wald test	3.258096	1	7.107e-02

La p - value du test est supérieure seuil critique de 5%, on accepte l'hypothèse de nullité de la constante.

On peut également tester la nullité du coefficient ou d'une ensemble de coefficients à partir du test F de Fisher. La statistique de test est :

$$\lambda_F = \lambda_W / N \tag{10.6}$$

Sous H_0, cette statistique suit une loi de Fisher à N et $T - p - q - 1$ degrés de libertés. Ceci s'effectue soit en fixant le paramètre `use_f` à `True`, soit en utilisant la fonction res.f_test().

```
# Test de Fisher
ftest1 = model6.wald_test(R,use_f = True)
ftest1_res = np.c_[ftest1.statistic,ftest1.df_denom,ftest1.pvalue]
ftest2 = model6.f_test(R)
ftest2_res = np.c_[ftest2.statistic,ftest2.df_denom,ftest2.pvalue]
ftest1_res
```

```
## array([[3.25809611,        inf,        nan]])
```

```
ftest2_res
```

```
## array([[3.25809611,        inf,        nan]])
```

On peut également vérifier cela en utilisant la fonction res.t_test() pour le test de Student.

```
# Test de Student
ttest = model6.t_test(R)
ttest_res = pd.DataFrame(np.c_[ttest.statistic,ttest.pvalue],
                         columns = ["z","P>|z|"],index = ["t test"])
ttest_res.iloc[:,1] = scientific_format(digits=3)(ttest_res.iloc[:,1])
```

Table 10.3 – Test de Student de nullité de la constante du modèle $ARMA(1,2)$

	z	P>|z|
t test	1.80502	7.107e-02

La p - value du test est supérieure seuil critique de 5%, on accepte l'hypothèse de nullité de la constante.

10.1.2 Le coefficient de détermination

Les coefficients de détermination (R^2 classique ou $\overline{R}^2$ corrigé) des modèles estimés sont :

$$R^2 = \frac{\sum_{t=1}^{t=T}\left(\widehat{X}_t - \overline{X}\right)^2}{\sum_{t=1}^{t=T}\left(X_t - \overline{X}\right)^2} = 1 - \frac{\sum_{t=1}^{t=T}\hat{\varepsilon}_t^2}{\sum_{t=1}^{t=T}\left(X_t - \overline{X}\right)^2} \quad \text{et} \quad \overline{R}^2 = 1 - \frac{T-1}{T-p-q-1}\left(1-R^2\right) \tag{10.7}$$

où $\hat{\varepsilon}_t$ est le résidu d'estimation. On utilise $\overline{R}^2$ dans de nombreux cas, puisqu'il permet de prendre en compte le nombre de variables explicatives, c'est-à-dire ici les retards p de l'AR et les retards q du MA.

Il est nécessaire de faire attention lorsque l'on calcule le coefficient de détermination parce qu'il peut conduire à des conclusions erronées. En effet, certains points d'influence peuvent particulièrement augmenter la valeur du coefficient de détermination, ce qui peut parfois laisser penser que les prédictions sont assez précises.

Exemple 10.3 *Coefficient de détermination classique et ajusté*

On calcule les coefficients de détermination (classique et ajusté) des modèles $ARMA(p,q)$ estimés.

```
# Coefficients de détermination classique et ajusté
from mizani.formatters import percent_format
def ExtractR2score(res,x):
  p,q = len(res.arparams),len(res.maparams)
  sct = np.sum([np.square(i - x.mean()) for i in np.array(x)])
  rsquared = 1 -(res.sse/sct)
  adjrsquared = 1 - ((res.nobs - 1)/(res.nobs - p - q - 1))*(1-rsquared)
  return list([rsquared,adjrsquared])
# Application
order = [(1,0,0),(2,0,0),(0,0,2),(0,0,4),(1,0,1),(1,0,2)]
label=["AR(1)","AR(2)","MA(2)","MA(4)","ARMA(1,1)","ARMA(1,2)"]
```

```
rsquared_res = pd.DataFrame(index = ["R2","Adj. R2"],columns = label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  rsquared_res[lab] = ExtractR2score(model,cpi2)
```

Table 10.4 – Coefficients de détermination (classique et ajusté)

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
R2	0.1484301	0.1558466	0.1320692	0.1944684	0.1934907	0.2120121
Adj. R2	0.1441722	0.1473626	0.1233463	0.1781125	0.1853851	0.2000729

Nos modèles $ARMA(p,q)$ manquent d'un pouvoir explicatif car les coefficients de détermination (classique et ajusté) sont très faibles ($< 50\%$).

La significativité de ce coefficient de détermination est testée à l'aide d'une statistique de Fisher classique (ce coefficient est proche de 1 si $\sum_{t=1}^{t=T} \hat{\varepsilon}_t$ tend vers 0). Les hypothèses du test sont :

$$\begin{cases} H_0 : & SCE = 0 \\ H_1 : & SCE \neq 0 \end{cases} \tag{10.8}$$

Pour effectuer ce test, la statistique de test calculée est la suivante :

$$F_{cal} = \frac{T-p-q-1}{p+q} \times \frac{SCE}{SCR} = \frac{T-p-q-1}{p+q} \times \frac{R^2}{1-R^2} \tag{10.9}$$

Sous H_0, cette statistique suit une loi de Fisher à $(p+q)$ et $(T-p-q-1)$ degrés de liberté. On rejette l'hypothèse nulle au risque α si $F_{cal} > F^{\alpha}_{p+q,T-p-q-1}$.

Exemple 10.4 *Test de significativité globale du modèle*

On effectue le test de significativité globale pour les différents modèles $ARMA(p,q)$ estimés.

```
# Test de significativité globale du modèle
def global_test(res,x):
  p,q = len(res.arparams),len(res.maparams)
  sct = np.sum([np.square(i - x.mean()) for i in np.array(x)])
  ddl1,ddl2 = p+q, res.nobs-p-q-1
  val = (ddl2/ddl1)*((sct/res.sse)-1)
  pvalue = st.f.sf(val,ddl1,ddl2)
  return list([val,int(ddl1),int(ddl2),pvalue])
# Application
global_res = pd.DataFrame(columns =["F-stat","df num","df denom","p-value"],
                          index = label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  global_res.loc[lab,:] = global_test(model,cpi2)
global_res.iloc[:,3] = scientific_format(digits=3)(global_res.iloc[:,3])
```

Toutes les p - values sont nulles, donc nos modèles $ARMA(p,q)$ sont globalement significatifs.

Table 10.5 – Test de significativité globale du modèle

	F-stat	df num	df denom	p-value
AR(1)	34.86034	1	200	1.498e-08
AR(2)	18.36957	2	199	4.775e-08
MA(2)	15.14048	2	199	7.573e-07
MA(4)	11.88975	4	197	1.131e-08
ARMA(1,1)	23.87118	2	199	5.101e-10
ARMA(1,2)	17.75763	3	198	3.009e-10

Remarque 10.1

Ce test présente un intérêt limité car il permet seulement de savoir si un ordre de la AR ou MA est significatif.

10.2 Tests sur les résidus du bruit blanc normal

Ces tests ont pour objet de vérifier que les résidus estimés $\hat{\varepsilon}_t = \dfrac{\widehat{\Phi}(L)}{\widehat{\Theta}(L)} X_t$ suivent bien un processus de bruit blanc. A cette fin, on peut appliquer des tests d'absence d'autocorrélation et des tests d'homoscédasticité.

10.2.1 Test de nullité de la moyenne des résidus

Un bruit blanc est d'espérance mathématique nulle. On effectue donc sur les résidus $\hat{\varepsilon}_t$, prévisionnels du modèle un test de nullité de leur moyenne. Pour T suffisant grand ($T > 30$),

$$\overline{\hat{\varepsilon}} \sim \mathcal{N}\left(\mu_{\hat{\varepsilon}}; \frac{\sigma_{\hat{\varepsilon}}}{\sqrt{T}}\right) \quad \text{et} \quad \frac{\overline{\hat{\varepsilon}} - \mu_{\varepsilon}}{\sigma_{\hat{\varepsilon}}}\sqrt{T} \sim \mathcal{N}(0;1) \tag{10.10}$$

Sous l'hypothèse nulle H_0 de nullité de la moyenne et un seul d'acceptation à 95%, l'intervalle de confiance de $\overline{\hat{\varepsilon}}$:

$$\overline{\hat{\varepsilon}} \in \left[-1.96\frac{\sigma_{\hat{\varepsilon}}}{\sqrt{T}}; 1.96\frac{\sigma_{\hat{\varepsilon}}}{\sqrt{T}}\right] \quad \text{avec} \quad (z_{\alpha/2} = 1.96) \tag{10.11}$$

En général, cette hypothèse de nullité des résidus est vérifiée car elle découle de la méthode d'estimation des paramètres des processus ARMA.

Exemple 10.5 *Test de nullité des résidus des différents modèle $ARMA(p,q)$*

On teste la nullité des résidus des différents modèles $ARMA(p,q)$ estimés.

```
# Test de nullité de la moyenne des résidus
import scipy.stats as st
def t_test(x,mu=0):
  statistic, pvalue = st.ttest_1samp(a=x, popmean=mu)
  return list([statistic,pvalue])
# Application
ttest_res = pd.DataFrame(index = ["t-stat","p-value"],columns = label)
```

```
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  ttest_res[lab] = t_test(model.resid)
ttest_res.iloc[1,:] = scientific_format(digits=3)(ttest_res.iloc[1,:])
```

Toutes les p - values sont strictement supérieures au seuil critique de 5%, on accepte l'hypothèse de nullité de la moyenne des résidus pour tous les modèles $ARMA(p,q)$ estimés.

Table 10.6 – Test de nullité de la moyenne des résidus

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
t-stat	0.02516617	0.03717111	0.01378762	0.02063018	0.2858793	0.4512036
p-value	9.799e-01	9.704e-01	9.890e-01	9.836e-01	7.753e-01	6.523e-01

10.2.2 Test d'absence d'autocorrélation

Il existe un grand nombre de tests d'absence d'autocorrélation. Les plus connus étant ceux de G. E. Box and Pierce (1970), Ljung and Box (1978) et de Durbin and Watson (1950).

10.2.2.1 Tests de Box-Pierce et de Ljung - Box

Les tests de Box - Pierce et de Ljung - Box sont appelés des tests de Portmanteau [1]. Ces tests ont pour objet de tester le caractère non autocorrélé des résidus. L'hypothèse nulle est celle d'absence d'autocorrélation, c'est - à - dire :

$$H_0 : \hat{\rho}_{\hat{\varepsilon}}(1) = \cdots = \hat{\rho}_{\hat{\varepsilon}}(H) = 0 \tag{10.12}$$

La statistique de test de G. E. Box and Pierce (1970) s'écrit :

$$Q_{BP}(H) = T \sum_{j=1}^{j=H} \hat{\rho}^2_{\hat{\varepsilon}}(j) \tag{10.13}$$

où $\hat{\rho}_{\hat{\varepsilon}}(j)$ est le coefficient d'autocorrélation d'ordre j des résidus estimés et H est le nombre maximal de retards. Sous H_0, la statistique $Q_{BP}(H)$ suit une loi de χ^2 à $(H-p-q)$ degrés de liberté.

Exemple 10.6 *Statistique de Box - Pierce*

Sous Python, la fonction res.test_serial_correlation() permet de tester l'autocorrélation sur les résidus. Pour le test de Box- Pierce, on fixe le paramètre `method = "boxpierce"`.

```
# Test de Box-Pierce
bp = model6.test_serial_correlation(method="boxpierce",df_adjust=False,lags=12)
bp_res = pd.DataFrame(np.c_[bp[0][0],bp[0][1]].T,index=["Q-Stat","Prob"],
                      columns = [f"Q({x+1})" for x in range(12)])
```

1. Portmanteau Test Statistics : http ://cran.nexr.com/web/packages/portes/vignettes/portesFunctions.pdf

Table 10.7 – Statistique de Box - Pierce sur la série des résidus du modèle ARMA(2,1)

	Q(1)	Q(2)	Q(3)	Q(4)	Q(5)	Q(6)	Q(7)	Q(8)	Q(9)	Q(10)	Q(11)	Q(12)
Q-Stat	0.056	0.542	2.672	6.913	7.857	8.766	9.522	9.534	13.202	15.562	18.606	20.077
Prob	0.813	0.762	0.445	0.141	0.164	0.187	0.217	0.299	0.154	0.113	0.069	0.066

Toutes les p-values étant supérieures au seuil critique de 5%, on conclut que les résidus estimés ne sont pas autocorrélés au sens du test de Box - Pierce.

Le test de Ljung and Box (1978) est à appliquer, de préférence au test de Box-Pierce, lorsque l'échantillon est de petite taille. La distribution de la statistique du test Ljung and Box (1978) est en effet plus proche de celle du Khi - deux en petit échantillon que ne l'est celle du test de Box - Pierce. La statistique de test s'écrit :

$$Q_{JB}(H) = T(T+2)\sum_{j=1}^{j=H}\frac{\hat{\rho}_{\hat{\varepsilon}}^2(j)}{T-j} \tag{10.14}$$

Sous l'hypothèse nulle d'absence d'autocorrélation, la statistique $Q_{LB}(H)$ suit une loi de χ^2 à $(H-p-q)$ degrés de liberté.

Exemple 10.7 *Statistique de Ljung - Box*

Pour le test de Ljung - Box, on fixe le paramètre `method = "ljungbox"`.

```
# Test de Ljung - Box
lb = model6.test_serial_correlation(method="ljungbox",df_adjust=False,lags=12)
lb_res = pd.DataFrame(np.c_[lb[0][0],lb[0][1]].T,index=["Q-Stat","Prob"],
                      columns = [f"Q({x+1})" for x in range(12)])
```

Table 10.8 – Statistique de Ljung - Box sur la série des résidus du modèle ARMA(2,1)

	Q(1)	Q(2)	Q(3)	Q(4)	Q(5)	Q(6)	Q(7)	Q(8)	Q(9)	Q(10)	Q(11)	Q(12)
Q-Stat	0.057	0.553	2.737	7.106	8.084	9.030	9.820	9.833	13.709	16.218	19.468	21.048
Prob	0.812	0.758	0.434	0.130	0.152	0.172	0.199	0.277	0.133	0.094	0.053	0.050

Toutes les p-values étant supérieures au seuil critique de 5%, on conclut que les résidus estimés ne sont pas autocorrélés au sens du test de Ljung - Box.

Remarque 10.2

Dans le cas d'un résidu hétéroscédastique, il convient d'utiliser la statistique de Box - Pierce corrigée (*cf.* Mignon and Abraham-Frois (1998)). On remplace dans la statistique initiale la variance d'autocorrélation de $\hat{\rho}_{\hat{\varepsilon}}(i)$ par la variance asymptotique $\hat{\sigma}^2_{\hat{\rho}_{\hat{\varepsilon}}(i)}$ calculée dans le cas de l'hétéroscédasticité par Lo and MacKinlay (1989) selon la formule :

$$\hat{\sigma}^2_{\hat{\rho}_{\hat{\varepsilon}}(i)} = \frac{\sum_{t=i+1}^{t=T}\hat{\varepsilon}_t^2\hat{\varepsilon}_{t-i}^2}{\sum_{t=i+1}^{t=T}\hat{\varepsilon}_t^2} \tag{10.15}$$

Sous H_0, cette statistique suit une loi du χ^2 à h degrés de liberté où h est le nombre de retards retenus.

Remarque 10.3

Plusieurs versions du test de portmanteau ont été proposées par la suite par certains auteurs (*cf.* Danioko et al. (2022)). Chen (2002) a affiné le test de Box - Pierce en proposant la statistique Q_{LM} suivante :

$$Q_{LM}(H) = \frac{H(H+1)}{2T} + Q_{BP}(H) = \frac{H(H+1)}{2T} + T\sum_{j=1}^{j=H}\hat{\rho}^2_{\hat{\varepsilon}}(j) \tag{10.16}$$

Cette approche ne corrige que la moyenne de la statistique de Box - Pierce et par conséquent ne parvient pas à ajuster correctement les taux d'erreur de type I.

Monti (1994) a proposé un test de portmanteau basé sur l'autocorrélation partielle des résidus. Ce test est défini comme suit :

$$Q_M(H) = T(T+2)\sum_{j=1}^{j=H}\frac{\hat{\pi}^2_{\hat{\varepsilon}}(j)}{T-j} \tag{10.17}$$

avec $\hat{\pi}(j) = \hat{\phi}_{j,j}$. Monti (1994) a montré via des simulations que les performances de Q_M sont comparables à celles de Q_{LB}. De plus, il conclut que dans certains scénarios, Q_{LB} surpasse Q_M.

Peña and Rodríguez (2002) ont proposé un test basé sur une mesure différente de la dépendance de l'autocorrélation simple des résidus. Ils définissent la statistique D suivante :

$$D = T\left(1 - \left|\widehat{R}_H\right|^{1/H}\right) \tag{10.18}$$

où

$$\widehat{R}_H = \begin{pmatrix} 1 & \hat{\rho}_{\hat{\varepsilon}_t}(1) & \cdots & \hat{\rho}_{\hat{\varepsilon}_t}(H) \\ \hat{\rho}_{\hat{\varepsilon}_t}(1) & 1 & \cdots & \hat{\rho}_{\hat{\varepsilon}_t}(H-1) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\rho}_{\hat{\varepsilon}_t}(H) & \hat{\rho}_{\hat{\varepsilon}_t}(H-1) & \cdots & 1 \end{pmatrix} \tag{10.19}$$

Dans leurs travaux, les auteurs ont montré que dans des conditions particulières, leur test surpassait largement le test de Ljung - Box Q_{LB}. De plus, ils ont démontré que le test avait un avantage sur le test de Chen (2002) quelle que soit la taille de l'échantillon. Cependant, la convergence de la distribution asymptotique du test développé par Peña and Rodríguez (2002) est très lente (*cf.* Lin and McLeod (2006)).

Fisher (2011) a proposé de nouvelles versions pondérées du test de G. E. Box and Pierce (1970) et Monti (1994), les statistiques de test sont les suivantes :

$$Q_{WL}(H) = T(T+2)\sum_{j=1}^{j=H}\frac{H-j+1}{H(T-j)}\hat{\rho}^2_{\hat{\varepsilon}}(j) \tag{10.20}$$

et

$$Q_{WM}(H) = T(T+2)\sum_{j=1}^{j=H}\frac{H-j+1}{H(T-j)}\hat{\pi}^2_{\hat{\varepsilon}}(j) \tag{10.21}$$

Une étude comparative par simulation menée par Safi and Al-Reqep (2014) a montré que pour les échantillons de petite taille et des valeurs H, Q_{WL} fonctionne mieux que Q_{LB}. Pour les échantillons de taille modérée, ils ont également constaté que Q_{WL} fait mieux que Q_{LB} et que Q_{WM} surpasse Q_M.

10.2.2.2 Test de Durbin-Watson

Le test de Durbin and Watson (1950) (Durbin and Watson (1971)) permet de tester la présence d'autocorrélation à l'ordre 1 des résidus (c'est-à-dire, le résidu en t dépend du résidu en $t-1$, mais pas du retard en $t-2, t-3, ...$ On considère le processus suivant décrivant une autocorrélation à l'ordre 1 des résidus :

$$\hat{\varepsilon}_t = \rho\hat{\varepsilon}_{t-1} + \nu_t \tag{10.22}$$

ν_t est un bruit blanc et $\hat{\varepsilon}_t$ désigne les résidus estimés. Les hypothèses du test sont :

$$\begin{cases} H_0 : \rho = 0 & \text{(absence d'autocorrélation à l'ordre 1 des résidus)} \\ H_1 : \rho \neq 0 & \text{(présence d'autocorrélation à l'ordre 1 des résidus)} \end{cases} \tag{10.23}$$

La statistique de Durbin-Watson, noté DW, est donnée par :

$$DW = \frac{\sum_{t=2}^{t=T}(\hat{\varepsilon}_t - \hat{\varepsilon}_{t-1})^2}{\sum_{t=1}^{t=T}\hat{\varepsilon}_t^2} \tag{10.24}$$

Cette statistique varie entre 0 et 4 et vaut 2 en l'absence d'autocorrélation à l'ordre 1 des résidus. Durbin et Watson ont tabulé les valeurs critiques de la statistique DW en fonction de la taille de l'échantillon T et du nombre de variables explicatives. Cette approximation de DW, quand T est grand, tend vers $2(1-\hat{\rho})$; ce qui simplifie la lecture du test. Ainsi,

- Une valeur de DW proche de 0 signifie une autocorrélation positive.
- Une valeur proche de 4 signifie une autocorrélation négative.
- Une valeur proche de 2 signifie la non autocorrélation.

Durbin and Watson (1950) montrent que la distribution de la statistique DW oscille entre deux distributions $d_1(T)$ et $d_2(T)$. Les valeurs de $d_1(T)$ et $d_2(T)$ sont données par la table de Durbin (*cf.* Table **??**). La règle de décision est la suivante :

- Si $DW < d_1(T)$: on rejette H_0
- Si $DW > d_2(T)$: on accepte H_0
- Si $d_1(T) < DW < d_2(T)$: on ne peut rien conclure.

Il est important de noter que ce test n'est plus valable dès lors que le modèle estimé comprend une variable endogène retardée parmi les variables explicatives, ce qui est le cas lorsqu'on estime un processus avec une composante autorégressive. Dans ce cas, on calcule la statistique h de Durbin :

$$h = \hat{\rho}\sqrt{\frac{T}{1 - T \times \hat{\sigma}^2_{\phi_1}}} \tag{10.25}$$

où $\hat{\rho}$ est l'estimateur MCO de ρ dans l'équation précédente et $\hat{\sigma}^2_{\phi_1}$ désigne la variance estimée du coefficient de X_{t-1}. Sous l'hypothèse nulle, $\rho = 0$, la statistique h suit une loi normale centrée réduite. Le test h de Durbin est relativement peu puissant et ne peut être appliquée que si $T \times \hat{\sigma}^2_{\phi_1} < 1$.

Exemple 10.8 *Test de Durbin - Watson sur les résidus*

On considère les modèles $ARMA(p, q)$ estimés pour la série cpi. On demande de calculer la statistique de Durbin - Watson.

```
# Test de Durbin - Watson
from statsmodels.stats.stattools import durbin_watson
dw_res = pd.DataFrame(index = ["DW","rho"],columns = label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  dw = durbin_watson(model.resid)
  dw_res[lab] = list([dw, 1 - (dw/2)])
```

Table 10.9 – Statistique de Durbin - Watson sur la série des résidus des différents modèles estimés

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
DW	2.0626209	2.0294002	1.9619598	2.009874	1.7837048	2.0203013
rho	-0.0313104	-0.0147001	0.0190201	-0.004937	0.1081476	-0.0101507

10.2.2.3 Runs Test

Ce test (*cf.* Bradley (1968)) examine la fréquence de caractères répétitifs dans une chronique. Le nombre R de « *runs* » est le nombre de passage du signe « + » au signe « - » et inversement. Si les observations sont indépendantes (donc non corrélées), il est improbable d'avoir un nombre faible de « *runs* ». Dans le cas d'une indépendance, le nombre attendu de *runs* est égal à :

$$m = \frac{1}{T}\left(T(T+1) - \sum_{i=1}^{i=3} T_i^2\right) \tag{10.26}$$

où T_i est le nombre de runs respectivement positifs, négatifs ou nuls ($i = 1, 2, 3$).

La variance de m est :

$$\sigma^2_m = \frac{1}{T^2(T-1)}\left\{\left(\sum_{i=1}^{i=3} T_i^2\right)\left(\sum_{i=1}^{i=3} T_i^2 + T(T+1)\right) - 2T\sum_{i=1}^{i=3} T_i^3 - T^3\right\} \tag{10.27}$$

Pour T grand, la distribution est considérée comme normale et on démontre que :

$$Z = \frac{R + 0.5 - m}{\sigma_m} \longrightarrow \mathcal{N}(0; 1) \tag{10.28}$$

avec R, le nombre actuel de runs. L'hypothèse nulle d'indépendance des observations est réfusée au seuil de $\alpha = 5\%$ si $|z| > 1.96$.

Exemple 10.9 *Application du runs test sur les résidus des modèles $ARMA(p,q)$ estimés*

On demande d'appliquer le runs test sur les résidus des modèles $ARMA(p,q)$ estimés. Sous Python, la fonction runstest_1samp() de la librairie Statsmodels permet d'effectuer le runs test.

```
# Runs test
from statsmodels.sandbox.stats.runs import runstest_1samp
def runs_test(x,cor=False):
  statistic,pvalue = runstest_1samp(x,correction=cor)
  return list([statistic,pvalue])

# Application
rtest_res = pd.DataFrame(index = ["z-stat","p-value"],columns = label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  rtest_res[lab] = runs_test(model.resid)
rtest_res.iloc[1,:] = scientific_format(digits=3)(rtest_res.iloc[1,:])
```

Table 10.10 – Runs test sur les séries des résidus des modèles ARMA(p,q) estimés

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
z-stat	-3.641894	-3.630436	-3.933837	-3.317225	-4.133172	-1.513187
p-value	2.706e-04	2.829e-04	8.360e-05	9.092e-04	3.578e-05	1.302e-01

Pour le modèle $ARMA(1,2)$, nous avons une p-value qui est supérieure au seuil critique de 5%. Pour ce modèle, on accepte l'hypothèse nulle d'indépendance des valeurs résiduelles.

10.2.2.4 Test de Von Neumann

Une statistique populaire pour tester l'indépendance est le test de Von Neumann (1941). L'hypothèse nulle H_0 est celle d'indépendance des résidus. Dans le cas sans biais, la statistique du test est :

$$VN = \frac{T}{T-1} \times \frac{\sum_{t=1}^{t=T-1} \left(\hat{\varepsilon}_t - \hat{\varepsilon}_{t+1}\right)^2}{\sum_{t=1}^{t=T} \left(\hat{\varepsilon}_t - \bar{\hat{\varepsilon}}\right)^2} \tag{10.29}$$

Sous H_0, la statistique $z_1 = (VN - \mu)/\sigma$ suit asymptotiquement une loi normale standard avec :

$$\mu = \frac{2T}{T-1} \quad \text{et} \quad \sigma^2 = 4\frac{T^2(T-2)}{(T^2-1)(T-1)^2} \tag{10.30}$$

Dans le cas biaisé (original), la statistique du test est :

$$VN = \frac{\sum_{t=1}^{t=T-1} \left(\hat{\varepsilon}_t - \hat{\varepsilon}_{t+1}\right)^2}{\sum_{t=1}^{t=T} \left(\hat{\varepsilon}_t - \bar{\hat{\varepsilon}}\right)^2} \tag{10.31}$$

Sous H_0, la statistique $z_2 = (VN - 2)/\sigma$ suit asymptotiquement une loi normale standard avec :

$$\sigma^2 = 4\frac{(T-2)}{T^2-1} \tag{10.32}$$

On rejette l'hypothèse nulle d'indépendance si la pvalue est inférieure à la valeur critique α.

Exemple 10.10 *Application du test de Von - Neumann sur la série des résidus des différents modèles estimés*

On applique le test d'indépendence de Von - Neumann sur les séries des résidus des différents models estimés. Nous créons ici le code tel que présenté dans la fonction VonNeumannTest de la librairie DescTools. Cette fonction doit être utilisée avec précaution car elle comporte une erreur au niveau du cas sans biais. En effet, les auteurs de la fonction ont divisé le numerateur $(VN - \mu)$ par la variance au lieu de l'écart - type telle que définie par la formule.

```
# Test de Von Neumann
def VonNeumannTest(x,alternative = "two.sided",unbiased=True):
  x = np.array(x.dropna())
  T =  len(x)
  num = np.sum([(x[i] - x[i+1])**2 for i in range(T-1)])
  den = np.square((x - x.mean())).sum()
  if unbiased:
    VN = T*num/(den*(T-1))
    mu = 2*T/(T-1)
    sigma = np.sqrt(4*(T**2)*(T-2)/((T**2 - 1)*(T-1)**2))
    z = (VN - mu)/sigma
  else:
    VN = num/den
    sigma = np.sqrt(4*(T-2)/(T**2 - 1))
    z = (VN - 2)/sigma
  if alternative == "less":
    pval = st.norm.cdf(z)
  elif alternative == "greater":
    pval = st.norm.sf(z)
  else :
    pval = 2*st.norm.cdf(-np.abs(z))
  return list([VN,z,pval])

# Application
vntest_res = pd.DataFrame(index=["VN","z","pvalue"],columns=label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  vntest_res[lab] = VonNeumannTest(model.resid)
vntest_res.iloc[2,:] = scientific_format(digits=3)(vntest_res.iloc[2,:])
```

Au seuil de 5%, toutes les pvalues sont supérieures à cette valeur critique, par conséquent, on accepte l'hypothèse nulle d'indépendance des valeurs des résidus.

Table 10.11 – Test de Von - Neumann sur la série des résidus des différents modèles estimés

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
VN	2.072889	2.039511	1.971723	2.019878	1.793308	2.032409
z	0.4472655	0.2100671	-0.2716583	0.0705475	-1.539534	0.1595997
pvalue	6.547e-01	8.336e-01	7.859e-01	9.438e-01	1.237e-01	8.732e-01

10.2.2.5 Test de rang de Von Neumann

Une alternative au test de Von Neumman a été proposée par Bartels (1982). Ce test est basé sur les rangs de la série $(\widehat{\varepsilon}_t)_{t=1,\dots,T}$. Soit $(R_i)_{i=1,\dots,T}$ la nouvelle chronique obtenue. Le coefficient du ratio de test de Von Neumann (1941) (*cf.* Cromwell and Terraza (1994)) est :

$$RVN = \frac{\sum_{i=1}^{i=T-1} (R_i - R_{i+1})^2}{\sum_{j=1}^{j=T} (R_j - \mu_R)^2} \tag{10.33}$$

où $\mu_R = \dfrac{T+1}{2}$ est la moyenne des rangs. Les valeurs critiques pour ce test sont fournies dans Bartels (1982). Par exemple, pour le seuil de signification $\alpha = 5\%$, la valeur critique est obtenue est $\tau = 1.67$.

Table 10.12 – Valeurs critiques τ tabulées par Bartels(1982)

T	20	30	40	50	70	100
1%	1.04	1.20	1.29	1.36	1.45	1.54
5%	1.30	1.42	1.49	1.54	1.61	1.67

L'hypothèse nulle d'indépendance des valeurs des résidus est rejetée si $RVN > \tau$.

Sous H_0, la statistique $z_1 = (RVN - 2)/\sigma$ suit asymptotiquement une loi normale standard $\mathcal{N}(0,1)$ avec

$$\sigma^2 = \frac{4}{5} \times \frac{(T-2)(5T^2 - 2T - 9)}{T(T^2-1)(T-1)} \tag{10.34}$$

On peut également construire une statistique basée sur la loi beta. Ainsi, sous H_0, la statistique $z_2 = RVN/4$ suit asymtotiquement une loi $\mathcal{B}(p,p)$ avec

$$p = \frac{5}{2} \times \frac{T(T^2-1)(T-1)}{(T-2)(5T^2-2T-9)} - \frac{1}{2} \tag{10.35}$$

Ainsi, on rejette l'hypothèse nulle d'indépendance des valeurs des résidus si la pvalue associé à cette statistique est inférieure au seuil critique α.

Exemple 10.11 *Application du test de rang de Von - Neumann sur la série des résidus des différents modèles estimés*

Nous implémentons ici une fonction `VonNeumannRatioTest` identique à la fonction BartelsRank-Test de la librairie DescTools de R.

```
# Test de rang de Von de Neumann
def VonNeumannRatioTest(x,alternative ="two.sided",methode="normal"):
  pvalue = methode
  x = np.array(x.dropna()) # drop missing values
  R = np.array(st.rankdata(x,method='ordinal'))
  T = len(x)
  num = np.sum([(R[i] - R[i+1])**2 for i in range(T-1)])
  den = np.square((R - R.mean())).sum()
  RVN = num/den
  sigma = np.sqrt((4*(T-2)*(5*T**2-2*T-9))/(5*T*(T**2-1)*(T-1)))
  z = (RVN - 2)/sigma
  if methode == "auto":
    if T <= 100:
      pvalue = "beta"
    else:
      pvalue = "normal"
  if pvalue == "beta":
    btp = (5*T*(T**2-1)*(T-1))/(2*(T-2)*(5*T**2-2*T-9))-1/2
    pv0 = st.beta.cdf(x=RVN/4,a = btp,b = btp)
  if pvalue == "normal":
    pv0 = st.norm.cdf(z)
  if alternative == "two.sided":
    pval = 2*np.min([pv0, 1 - pv0])
  if alternative == "trend":
    pval = pv0
  if alternative == "oscillation" :
    pval = 1 - pv0
  return list([RVN,z,pval])

# Application
vnrtest_res = pd.DataFrame(index=["RVN","z","pvalue"],columns=label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  vnrtest_res[lab] = VonNeumannRatioTest(model.resid,methode="normal")
vnrtest_res.iloc[2,:] = scientific_format(digits=3)(vnrtest_res.iloc[2,:])
```

Table 10.13 – Test de rang de Von - Neumann sur la série des résidus des différents modèles estimés

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
RVN	1.500123	1.551684	1.358381	1.451006	1.540029	1.80807
z	-3.564732	-3.197036	-4.575518	-3.914995	-3.280148	-1.368691
pvalue	3.642e-04	1.388e-03	4.750e-06	9.041e-05	1.038e-03	1.711e-01

Au seuil de 5%, la pvalue du modèle $ARMA(1,2)$ est supérieure à cette valeur critique, par conséquent, on accepte l'hypothèse nulle d'indépendance des valeurs des résidus pour ce modèle.

10.2.3 Test d'homoscédasticité

Nous rappelons ci - après brièvement le principe des tests d'homoscédasticité les plus couramment employés en les adaptant au cas des processus ARMA.

10.2.3.1 Test de Goldfeld et Quandt

Ce test n'est valable que si l'une des variables X_j est la cause de l'hétéroscédasticité. On suppose que l'écart type de l'erreur augmente proportionnellement avec X_j. Dans notre cas, X_j désigne la variable endogène retardée, pour un certain retard j. Le principe du test est le suivant :

1. On classe les observations des variables endogènes (retardées et non retardées) en fonction des valeurs croissantes de X_j.
2. On néglige ensuite les m valeurs centrales de l'échantillon. On obtient alors deux sous échantillons : Un échantillon correspondant aux valeurs faibles de X_j et un échantillon correspondant aux valeurs élevées de X_j.
3. On estime par la méthode des MCO le modèle sur ces deux sous échantillons extrêmes, c'est-à-dire sur les $\frac{(T-2)}{2}$ plus petites valeurs et sur les $\frac{(T-2)}{2}$ plus grandes valeurs de X_j. Pour chacune des deux régressions, on calcule les sommes des carrés des résidus, SCR_1 et SCR_2, ainsi que le rapport $\frac{SCR_2}{SCR_1}$.

Sous l'hypothèse nulle d'homoscédasticité, la statistique $\frac{SCR_2}{SCR_1}$ suit une loi de Fisher à $\frac{T-m-2h}{2}$ et $\frac{T-m-2h}{2}$ degrés de liberté, où $h = p + q + 1$ est le nombre de paramètres estimés.

```
# Test de Goldfeld & Quandt
from statsmodels.stats.diagnostic import het_goldfeldquandt
```

Exemple 10.12 *Test d'hétéroscédasticité sur les résidus des modèles ARMA estimés*

Sous Python, la fonction res.test_heteroskedasticity() permet de tester l'hétéroscédasticité sur les résidus (*cf.* Harvey (1990)). L'hypothèse nulle est celle d'homoscédasticité. Pour $h = [T/3]$, la statistique de test utilisée s'écrit :

$$H(h) = \frac{\sum_{t=T-h+1}^{t=T} \hat{\varepsilon}_t^2}{\sum_{t=d+1}^{t=d+1+h} \hat{\varepsilon}_t^2} \tag{10.36}$$

Sous H_0, cette statistique suit une loi de Fisher à h et h degrés de liberté. De façon alternative, sous H_0, la statistique $h \times H(h)$ suit une loi de χ^2 à h degrés de liberté.

```
# Test d'hétéroscédasticité
def h_test(res):
  het = res.test_heteroskedasticity(method=None,alternative="two-sided",
                                    use_f=False)
  return list([het[0][0],het[0][1]])

# Application
htest_res = pd.DataFrame(index = ["chi2-stat","p-value"],columns = label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
```

```
  htest_res[lab] = h_test(model)
htest_res.iloc[1,:] = scientific_format(digits=3)(htest_res.iloc[1,:])
```

Table 10.14 – Test d'hétéroscédasticité sur la série des résidus des différents modèles estimés

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
chi2-stat	7.771011	9.327015	5.930069	7.854773	19.45655	19.46925
p-value	4.226e-71	3.580e-91	4.102e-48	3.614e-72	3.415e-228	2.279e-228

Toutes les pvalues sont nulles, par conséquent on rejette l'hypothèse nulle d'homoscédasticité au seuil de 5%. Les résidus sont hétéroscédastiques.

10.2.3.2 Test de White

Ce test général d'homoscédasticité est fondé sur l'existence d'une relation entre le carré du résidu et une ou plusieurs variables explicatives (endogènes retardées, dans notre cas) en niveau et au carré :

$$\hat{\varepsilon}_t^2 = \alpha_0 + \sum_{i=1}^{p} (\alpha_i X_{t-1} + \beta_i X_{t-1}^2) + \nu_t \tag{10.37}$$

Si au moins un des coefficients de régression est significatif, on rejette l'hypothèse nulle d'homoscédasticité en faveur de l'hypothèse alternative d'hétéroscédasticité. Pour effectuer ce test, on utilise la statistique du multiplicateur de Lagrange $T \times R^2$ où T est le nombre d'observation et R^2 et le coefficient de détermination associée à la régression (10.37). Sous l'hypothèse nulle, cette statistique suit une loi de Khi-deux à $2p$ degrés de liberté. En conséquence :

— Si $T \times R^2 < \chi^2(2p)$, on accepte l'hypothèse nulle d'homoscédasticité
— Si $T \times R^2 > \chi^2(2p)$, on conclut en faveur de l'hypothèse alternative d'hétéroscédasticité.

```
# Test de White
from statsmodels.stats.diagnostic import het_white
```

10.2.3.3 Test de Breusch et Pagan

Il s'agit d'un test très général dans la mesure où il couvre un grand nombre de cas d'hétéroscédasticité. On suppose que les résidus du processus suivi par l'endogène X_t (processus de type ARMA dans notre cas) sont indépendants et suivent une loi normale de variance $\sigma_{\varepsilon_t}^2 = h(Z_t^{'}\alpha)$ où α est un vecteur de coefficients de dimension $(p \times 1)$ et Z_t est un vecteur $(p \times 1)$ de variables dont on pense qu'elles peuvent être responsables de l'hétéroscédasticité. Dans notre cas, Z_t est un vecteur de variables endogènes retardées.

La fonction $h(.)$ est supposée continue et au moins deux fois différentiables. Afin d'éviter que la matrice d'information soit non singulière sous l'hypothèse nulle d'homoscédasticité, on suppose également que $h(0) = 1$ et que $h^{'}(0) \neq 0$. L'hypothèse nulle d'homoscédasticité s'écrit :

$$\alpha_1 = \alpha_2 = \cdots = \alpha_p = 0 \tag{10.38}$$

puisque, dans ce cas, la variance des erreurs $\sigma_{\varepsilon_t}^2$ est donnée par $\sigma_{\varepsilon_t}^2 = h(\alpha_0)$ et est donc bien constante au cours du temps. L'hypothèse alternative d'hétéroscédasticité correspond alors au cas où α contient des éléments non nuls. Afin de mettre en oeuvre le test, on estime $\sigma_{\varepsilon_t}^2$ par :

$$\hat{\sigma}^2_{\varepsilon_t} = \frac{1}{T}\sum_{t=1}^{T}\hat{\varepsilon}^2_t \text{ et on calcule } \nu_t = \frac{\hat{\varepsilon}^2_t}{\hat{\sigma}^2_{\varepsilon_t}} \tag{10.39}$$

On effectue alors la régression de ν_t sur Z_t et on calcule, sur la base de cette régression, la somme des carrés expliquées (SCE). Sous l'hypothèse nulle d'homoscédasticité, la statistique $Q = \frac{SCE}{2}$ suit asymptotiquement une loi de Khi-deux à $(p-1)$ degrés de liberté.

```
# Test de Breusch - Pagan
from statsmodels.stats.diagnostic import het_breuschpagan
```

10.2.3.4 Test d'hétéroscédasticité de ARCH

Ce test, très fréquemment employé en économétrie des séries temporelles financières, a pour objet de tester l'hypothèse nulle d'homoscédasticité contre l'hypothèse alternative d'hétéroscédasticité conditionnelle (*cf.* Engle (1982), McLeod and Li (1983)). On effectue la régression suivante :

$$\hat{\varepsilon}^2_t = \alpha_0 + \sum_{i=1}^{l}\alpha_i\hat{\varepsilon}^2_{t-i} \tag{10.40}$$

où $\hat{\varepsilon}_t$ sont les résidus issus de l'estimation du processus de type $ARMA(p,q)$. On calcule la statistique $T \times R^2$ où T est le nombre d'observations de la série $\hat{\varepsilon}_t$ et R^2 est le coefficient de détermination associé à l'équation précédente.

Sous l'hypothèse nulle d'homoscédasticité ($\alpha_i = 0$, $\forall i = 1, \dots, l$), la statistique $T \times R^2$ suit une loi de Khi-deux à l degrés de liberté. La règle de décision est alors :

- Si $T \times R^2 < \chi^2(l)$, on accepte l'hypothèse nulle d'homoscédasticité
- Si $T \times R^2 \geq \chi^2(l)$, on rejette l'hypothèse nulle en faveur de l'hypothèse alternative d'hétéroscédasticité conditionnelle.

Exemple 10.13 *ARCH - LM test sur les résidus des modèles $ARMA(p,q)$ estimés*

On souhaite effectuer un test du multiplicateur de Lagrange sur les résidus des modèles $ARMA(p,q)$ estimés.

```
# Test ARCH
from statsmodels.stats.diagnostic import het_arch
def ArchTest(res):
  p, q = len(res.arparams),len(res.maparams)
  archtest = het_arch(res.resid,ddof=(p+q))
  return list([archtest[0],archtest[1],archtest[2],archtest[3]])

# Application
archtest_res = pd.DataFrame(index = ["LM stat","LM pvalue","F stat","F pvalue"],
                            columns = label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  archtest_res[lab] = ArchTest(model)
archtest_res.iloc[1,:] = scientific_format(digits=3)(archtest_res.iloc[1,:])
archtest_res.iloc[3,:] = scientific_format(digits=3)(archtest_res.iloc[3,:])
```

Table 10.15 – Test d'hétéroscédasticité (ARCH - LM test) sur la série des résidus des différents modèles estimés

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
LM stat	80.91308	79.19248	73.27383	70.47185	68.18096	75.76894
LM pvalue	3.324e-13	7.227e-13	1.030e-11	3.594e-11	9.941e-11	3.370e-12
F stat	13.30337	12.9358	11.36212	10.85306	10.1304	12.11168
F pvalue	2.351e-17	6.406e-17	5.256e-15	2.279e-14	1.896e-13	6.290e-16

Toutes les pvalues sont nulles, par conséquent on rejette l'hypothèse nulle d'homoscédasticité au seuil de 5%. Les résidus sont hétéroscédastiques.

10.2.3.5 Test de stabilité structurelle de Chow

Le test de Chow (1960) (voir aussi Toyoda (1974)) sert à comparer des variances des résidus sur des sous - périodes de la chronique. Considérons le modèle suivant :

$$\hat{\varepsilon}_t = \begin{cases} \alpha_0 + \alpha_1 \hat{\varepsilon}_{t-1} + \alpha_2 \hat{\varepsilon}_{t-2} & \text{si} \quad t \leq \tau \\ \beta_0 + \beta_1 \hat{\varepsilon}_{t-1} + \beta_2 \hat{\varepsilon}_{t-2} & \text{si} \quad t > \tau \end{cases} \tag{10.41}$$

Le test de Chow (1960) permet de savoir si les paramètres sont identiques au cours de deux ou plusieurs sous périodes. L'hypothèse nulle est qu'il n'y a pas de changement structurel, i.e. les coefficients sont égaux pour les deux groupes de données. La statistique du test est donné par :

$$F_{\text{chow}} = \frac{(SCR - SCR_1 - SCR_2)/k}{(SCR_1 + SCR_2)/(T - 2k)} \tag{10.42}$$

où SCR_1 et SCR_2 sont respectivement les sommes des carrés des résidus à partir du premier et du deuxième régime ; SCR est la somme des carrés des résidus issus de la régression linéaire utilisant toutes les données ; T est le nombre total des observations et k est le nombre de paramètres à estimer.

Sous l'hypothèse nulle d'absence de changement structurel, la statistique F_{chow} suit une loi de Fisher à k et $T - 2k$ degrés de liberté.

Exemple 10.14 *Application du test de Chow dans le cadre d'un modèle linéaire*

Nous illustrons le test de stabilité de Chow dans le cadre d'un modèle linéaire multiple. Considérons le modèle suivant :

$$y_t = \begin{cases} \alpha_0 + \sum_{j=1}^{j=p} \alpha_j x_{j,t} & \text{si} \quad t \leq \tau_1 \\ \beta_0 + \sum_{j=1}^{j=p} \beta_j x_{j,t} & \text{si} \quad t \geq \tau_2 \end{cases} \tag{10.43}$$

avec $\tau_2 > \tau_1$. La statistique de Chow est donnée par :

$$F_{\text{chow}} = \frac{(SCR - SCR_1 - SCR_2)/k}{(SCR_1 + SCR_2)/(T_1 + T_2 - 2k)} \tag{10.44}$$

avec $k = p + 1$. Sous l'hypothèse nulle d'absence de changement structurelle, la statistique F_{chow} suit une loi de Fisher à k et $T_1 + T_2 - 2k$ degrés de liberté.

```
# Test de Chow
def ChowTest(X,y,tau1=int,tau2=int,alpha=0.05):
  import statsmodels.api as sm
  def calculate_RSS(X, y):
    x = sm.add_constant(X)
    model = sm.OLS(y, x).fit()
    return np.square(model.resid).sum()
  # SCR total
  scr = calculate_RSS(X, y)
  # SCR Modèle 1
  X1 = X.loc[:tau1]
  y1 = y.loc[:tau1]
  scr1 = calculate_RSS(X1, y1)
  # SCR modèle 2
  X2 = X.loc[tau2:]
  y2 = y.loc[tau2:]
  scr2 = calculate_RSS(X2, y2)
  # Longueur des séries
  k = X.shape[1] + 1
  T1 = X1.shape[0]
  T2 = X2.shape[0]
  # Numérateur
  numerator = (scr - scr1  - scr2)/k
  # Dénominateur
  denominator = (scr1 + scr2)/(T1 + T2 - 2*k)
  # Statistique de Chow
  f_chow = numerator / denominator
  # P value associée à fchow
  import scipy.stats as st
  p_value = st.f.sf(f_chow,dfn = k,dfd = (T1 + T2 - 2*k))
  if p_value < alpha :
    msg = "rejet"
  else:
    msg = "accept"
  return list([f_chow,p_value,msg])
```

Considérons les données représentées par la figure (10.1) qui met en relation la variable x et la variable y.

```
# Données
df = pd.DataFrame({
  'x': [1, 1, 2, 3, 4, 4, 5, 5, 6, 7, 7, 8, 8, 9, 10, 10,
       11, 12, 12, 13, 14, 15, 15, 16, 17, 18, 18, 19, 20, 20],
  'y': [3, 5, 6, 10, 13, 15, 17, 14, 20, 23, 25, 27, 30, 30, 31,
      33, 32, 32, 30, 32, 34, 34, 37, 35, 34, 36, 34, 37, 38, 36]})
# Représentation graphique
fig, axe =plt.subplots(figsize=(16,6))
axe.scatter(df.x, df.y, color = "black");
axe.set_xlabel("x");
axe.set_ylabel("y");
plt.show()
```

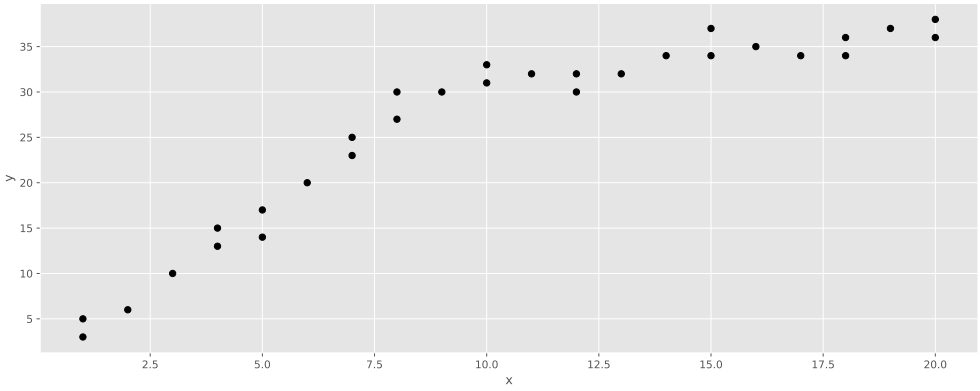


Figure 10.1 – Nuage de points entre x et y

La visualisation du nuage de points montre qu'il y a un changement au niveau du quinzième et du seizième point, ce qui signifie que $\tau_1 = 15$ et $\tau_2 = 16$.

```
# Application du test de Chow
chowtest = ChowTest(y=df[['y']],X=df[['x']],tau1=15,tau2=16,alpha=0.05)
chowtest_res = pd.DataFrame(
  {"statistic": chowtest[0],"p-value":chowtest[1],"decision" : chowtest[2]},
  index=["Chow test"])
chowtest_res.iloc[:,1] = scientific_format(digits=3)(chowtest_res.iloc[:,1])
```

Table 10.16 – Test de stabilité structurelle de Chow

	statistic	p-value	decision
Chow test	118.141	8.926e-14	rejet

Comme la pvalue est inférieure au seuil de significativité de 5%, on rejette l'hypothèse nulle d'absence de changement structurel entre les donnée.

10.3 Tests de normalité des résidus

Il existe deux grandes familles de tests de normalité : la famille des tests paramétriques et celle des tests non paramétriques.

10.3.1 Tests paramétriques

Les tests paramétriques sont asymptotiquement puissants. Ce qui signifie que l'on ne peut les appliquer à des séries d'observation que si l'on dispose d'un nombre d'observations suffisant, au minimum trente observations.

10.3.1.1 QQ - Plot et droite de Henry

L'hypothèse de normalité est examinée d'abord graphiquement en affichant le graphique quantile - quantile aussi appelé QQ - Plot. Le QQ - Plot est une technique graphique qui permet de comparer les distributions de deux ensembles de données. Les échantillons ne sont pas forcément

de même taille. Il se peut également qu'un des ensembles de données soient générées à partir d'une loi de probabilité qui sert de référentiel. Concrètement, il s'agit :

1. de trier les données de manière croissante pour former la série $\hat{\varepsilon}_{(i)}$.
2. à chaque valeur $\hat{\varepsilon}_{(i)}$, nous associons la fonction de répartition empirique $F_i = \dfrac{i - 0.375}{T + 0.25}$.
3. de calculer les quantiles successifs $z^*_{(i)}$ d'ordre F_i en utilisant l'inverse de la loi normale centrée et réduite.
4. dénormaliser les données initiales en appliquant la transformation $\hat{\varepsilon}^*_i = z^*_i \times s_{\hat{\varepsilon}} + \bar{\hat{\varepsilon}}$

Si les données sont compatibles avec la loi normale, les points $(\hat{\varepsilon}_{(i)}, \hat{\varepsilon}^*_{(i)})$ forment uen droite, dite droite de Henry, alignés sur la diagonale principale.

Exemple 10.15 *QQ - Plot des résidus du modèle* $ARMA(1,2)$

Sous Python, la fonction qqplot() de Statsmodels permet de visualiser le QQ - Plot associé à une série

```
## QQPlot
from statsmodels.graphics.gofplots import qqplot
fig, axe = plt.subplots(figsize=(16,6))
qqplot(model6.resid,dist = st.t, fit =True, line="45",ax = axe);
axe.set_title("QQ plot");
axe.set_xlabel("Theoretical Quantiles");
axe.set_ylabel("Sample Quantiles");
plt.show()
```

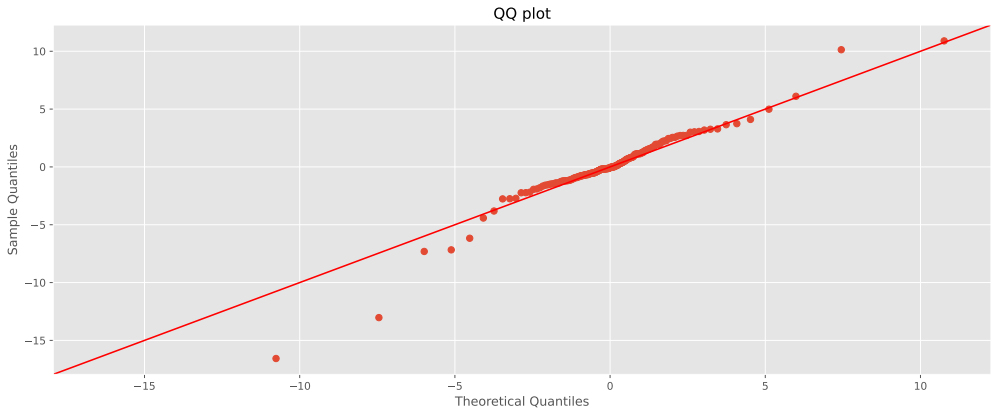


Figure 10.2 – QQ- plot sur les résidus du modèle $ARMA(1,2)$

10.3.1.2 Test de Skewness et Kurtosis

On peut regarder les moments d'ordre supérieur et la distribution empirique des résidus estimés. On regarde particulièrement les moments d'ordre 3 et 4 qui correspondent respectivement au Skewness et au Kurtosis et qui sont estimés par leurs valeurs empiriques, $\widehat{S}$ et $\widehat{K}$, définies par :

$$\widehat{S} = \frac{\hat{\mu}_3}{\widehat{\sigma_{\hat{\varepsilon}}}^3} = \frac{\frac{1}{T}\sum_{t=1}^{t=T}(\hat{\varepsilon}_t - \overline{\hat{\varepsilon}})^3}{\left(\frac{1}{T}\sum_{t=1}^{t=T}(\hat{\varepsilon}_t - \overline{\hat{\varepsilon}})^2\right)^{3/2}} \quad \text{et} \quad \widehat{K} = \frac{\hat{\mu}_4}{\widehat{\sigma_{\hat{\varepsilon}}}^4} = \frac{\frac{1}{T}\sum_{t=1}^{t=T}(\hat{\varepsilon}_t - \overline{\hat{\varepsilon}})^4}{\left(\frac{1}{T}\sum_{t=1}^{t=T}(\hat{\varepsilon}_t - \overline{\hat{\varepsilon}})^2\right)^{2}} \tag{10.45}$$

où $\overline{\hat{\varepsilon}}$ est la moyenne empirique calculée sur l'ensemble de l'échantillon $\hat{\varepsilon}_1, \ldots, \hat{\varepsilon}_T$.

On sait que pour une distribution symétrique, en particulier pour le loi normale, sa moyenne est son point symétrique et ses moments centrés d'ordre impaire sont nuls, donc son Skewness est égal à 0 et son Kurtosis vaut 3. Donc, si $\widehat{S} = 0$, la distribution des résidus est symétrique. De plus, si $\widehat{K} = 3$, la distribution des résidus semble être normale. Si ces deux conditions sont vérifiées, on conclut que $\hat{\varepsilon}_t \sim \mathcal{N}(0, \sigma^2_{\hat{\varepsilon}})$.

Le test de Skewness et de Kurtosis consiste à tester l'hypothèse nulle de normalité des résidus contre l'hypothèse alternative selon laquelle les résidus ne suivent pas une loi normale.

Définition 10.1 *Test de Skewness*

Les hypothèses du test de Skewness sont :

$$\begin{cases} H_{01} : & \mu_3 = 0 \quad \Longrightarrow \quad S = 0 \\ H_{11} : & \mu_3 \neq 0 \quad \Longrightarrow \quad S \neq 0 \end{cases} \tag{10.46}$$

Sous l'hypothèse de normalité des résidus, l'estimateur $\widehat{S}$ de S suit asymptotiquement une loi $\mathcal{N}(0; 3!/T)$ où $(3! = 6)$. La statistique centrée et réduite

$$z_{\widehat{S}} = \sqrt{T/6}\widehat{S} \tag{10.47}$$

suit asymptotiquement une loi $\mathcal{N}(0,1)$. On rejette H_{01} si $z_{\widehat{S}} \notin \left]-z_{\alpha/2}; z_{\alpha/2}\right[$ où $z_{\alpha/2}$ est le quantile de la loi normale standard.

Si on rejette H_{01}, le test est terminé car la loi n'étant pas symétrique, elle ne peut pas être normale. Si on accepte H_{01}, on passe au test suivant.

Définition 10.2 *Test de Kurtosis*

Ce test est à effectuer si le test de Skewness accepte l'hypothèse d'une distribution symétrique des résidus. Les hypothèses du test de Kurtosis sont :

$$\begin{cases} H_{02} : & \mu_4 = 0 \quad \Longrightarrow \quad K = 3 \\ H_{12} : & \mu_4 \neq 0 \quad \Longrightarrow \quad K \neq 3 \end{cases} \tag{10.48}$$

Sous l'hypothèse de normalité des résidus, l'estimateur $\widehat{K}$ de K suit asymptotiquement une loi $\mathcal{N}(3, 4!/T)$ où $(4! = 24)$. La statistique centrée et réduite

$$z_{\widehat{K}} = \sqrt{T/24}\left(\widehat{K} - 3\right) \tag{10.49}$$

suit asymptotiquement une loi $\mathcal{N}(0,1)$. On rejette H_{02} si $z_{\widehat{K}} \notin \left]-z_{\alpha/2}; z_{\alpha/2}\right[$ où $z_{\alpha/2}$ est le quantile de la loi normale standard.

Si on accepte H_{02}, alors on décide d'une normalité des résidus car les deux propriétés sont vérifiées. Si on rejette H_{02}, le test est terminé car la loi n'ayant pas un coefficient d'aplatissement égal à 3, nous ne sommes pas dans le cadre de la loi normale.

Exemple 10.16 *Application du test de Skewness et de Kurtosis sur les résidus*

Sous Python, les fonctions skewtest et kurtosistest de la libriarie Scipy permettent d'effectuer le test de skewness et de kurtosis respectivement

```
# Test du Skewness et du Kurtosis
def SkewnessKurtosisTest(res):
  skewstat, skewpvalue = st.skewtest(res.resid)
  kurtosisstat, kurtosispvalue =  st.kurtosistest(res.resid)
  return list([skewstat,skewpvalue,kurtosisstat,kurtosispvalue])

# Application
sktest_res = pd.DataFrame(columns = label,
  index=["S stat","S pvalue","K stat","K pvalue"])
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  sktest_res[lab] = SkewnessKurtosisTest(model)
sktest_res.iloc[1,:] = scientific_format(digits=3)(sktest_res.iloc[1,:])
```

Table 10.17 – Test de Skewness et de Kurtosis sur la série des résidus des différents modèles estimés

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
S stat	-3.499719	-4.953214	-1.989563	-2.691516	-8.518682	-7.13614
S pvalue	4.657e-04	7.300e-07	4.664e-02	7.113e-03	1.614e-17	9.599e-13
K stat	6.88186	7.177595	6.645217	7.072932	8.180422	7.719337
K pvalue	5.907591e-12	7.094859e-13	3.027702e-11	1.516941e-12	2.82851e-16	1.169364e-14

Nous avons des valeurs négatives du skewness, ce qui traduit une asymétrie avec une queue de de distribution plus étendue à gauche. Les valeurs du kurtosis centré $(K-3)$ sont largement supérieures à 0. ce qui traduit une distribution « pointue » (leptokurtique). Pour le test de Skewness, les pvalues sont nulles, donc la distribution n'est pas symétrique. On s'arrête là.

10.3.1.3 Test de Jarque-Bera

Le test de normalité de Jarque and Bera (1987) est fondé sur les coefficients d'asymétrie et d'aplatissement. Il évalue les écarts simultanés de ces coefficients avec les valeurs de référence de la loi normale. La formulation est très simple par rapport au test de D'Agostino, le prix est une puissance mindre. Il ne devient réellement intéressant que lorsque les effectifs sont très élevés.

Prenons les coefficients estimés d'asymétrie et d'aplatissement de Pearson $\widehat{S}$ et $\widehat{K}$, la loi conjointe de ces estimateurs est normale bivariée, on écrit :

$$\sqrt{T}\begin{pmatrix}\widehat{S}\\ \widehat{K}\end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix}0\\3\end{pmatrix}, \begin{pmatrix}6 & 0\\ 0 & 24\end{pmatrix}\right] \tag{10.50}$$

La matrice de variance - covariance présentée ici est une expression simplifiée valable pour les grandes valeurs de T. Nous notons que la covariance de S et K est nulle. La forme quadratique associée permet de produire la statistique de Jarque - Bera JB qui s'écrit :

$$JB = T\left(\frac{\widehat{S}^2}{6} + \frac{(\widehat{K}-3)^2}{24}\right) \tag{10.51}$$

Sous H_0, cette statistique est distribuée asymptotiquement selon une loi du χ^2 à 2 degrés de libertés. La statistiue JB prend des valeurs d'autant plus élevées que l'écart entre la distribution empirique et la loi normale est manifeste. Pour un risque α, la région critique du test est définie par : $JB > \chi^2_{1-\alpha}(2)$. Pour un risque $\alpha = 5\%$, le seuil critique est $\chi^2_{0.95}(2) = 5.99$.

Exemple 10.17 *Application du test de Jarque Bera sur les résidus*

On applique le test de Jarque Bera sur les résidus des modèles $ARMA(p,q)$ estimés. Sous Python, la fonction jarque_bera teste la normalité des résidus à partir du test de Jarque - Bera.

```
## Test de normalité de Jarque-Bera
from statsmodels.stats.stattools import jarque_bera
def JarqueBeraTest(res):
  jbstatistic, jbpvalue,_,_ = jarque_bera(model.resid)
  return list([jbstatistic,jbpvalue])

# Application
jbtest_res = pd.DataFrame(index=["JB stat","JB pvalue"],columns = label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  jbtest_res[lab] = JarqueBeraTest(model)
jbtest_res.iloc[1,:] = scientific_format(digits=3)(jbtest_res.iloc[1,:])
```

Table 10.18 – Test de Jarque - Bera sur la série des résidus des différents modèles estimés

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
JB stat	830.4922	1135.834	649.1038	999.5804	3459.556	2051.579
JB pvalue	4.581e-181	2.274e-247	1.119e-141	8.788e-218	0.000e+00	0.000e+00

Toutes les pvalues sont nulles, par conséquent on rejette l'hypothèse nulle de normalité des résidus.

On peut également se servir de la fonction res.test_normality() du modèle ARMA estimé. Cette fonction utilise le test de Jarque - Bera pour tester la normalité des résidus.

```
# Test de Jarque - Bera
def JarqueBeraTest2(res):
  jbtest = res.test_normality(method = "jarquebera")
  return list([jbtest[0][0],jbtest[0][1]])

# Application
jbtest2_res = pd.DataFrame(index=["JB stat","pvalue"],columns = label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  jbtest2_res[lab] = JarqueBeraTest2(model)
jbtest2_res.iloc[1,:] = scientific_format(digits=3)(jbtest2_res.iloc[1,:])
```

Table 10.19 – Test de Jarque - Bera sur la série des résidus des différents modèles estimés

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
JB stat	833.1939	1139.796	650.759	1006.718	3479.69	2064.623
pvalue	1.186e-181	3.137e-248	4.892e-142	2.477e-219	0.000e+00	0.000e+00

Les statistiques renvoyées sont légèrement différentes comparativement à celles obtenues avec la fonction jarque_bera. Cependant, les conclusions sont identiques. On rejette l'hypothèse nulle de normalité des résidus.

10.3.2 Tests non paramétriques

Pour ces tests, on ne spécifie pas sous H_0 l'égalité d'un paramètre à une norme, mais de façon plus générale l'égalité entre les distributions de probabilité empirique et théorique.

10.3.2.1 Test de Shapiro - Wilk

Très populaire, le test de Samuel Sanford Shapiro and Wilk (1965) est basé sur la statistique W. En comparaison des autres tests, il est particulièrement puissant pour les petits effectifs ($T \leq 50$). La statistique du test s'écrit :

$$W = \frac{1}{\sum_{i=1}^{i=T} \left(\hat{\varepsilon}_i - \bar{\hat{\varepsilon}}\right)^2} \left[\sum_{i=1}^{i=[T/2]} a_i \left(\hat{\varepsilon}_{(T-i+1)} - \hat{\varepsilon}_{(i)} \right) \right]^2 \tag{10.52}$$

où $\hat{\varepsilon}_{(i)}$ correspond à la série des données triées ; $[T/2]$ est la partie entière du rapport $T/2$. Les a_i sont des constantes générées à partir de la moyenne et de la matrice de variance - covariance des quantiles d'un échantillon de taille T suivant la loi normale. Ces constantes sont fournies dans des tables spécifiques.

La statistique W peut donc être interprétée comme le coefficient de détermination (le carré du coefficient de corrélation) entre la série des quantiles générées à partir de la loi normale et les quantiles empiriques obtenues à partir des données. Plus W est élevé, plus la compatibilité avec la loi normale est crédible. La région critique, rejet de la normalité, s'écrit : $W < W_{crit}$.

Les valeurs seuils W_{crit} pour différents risques α et effectifs T sont lues dans la table de Shapiro - Wilk.

Exemple 10.18 *Application du test de Shapiro - Wilk sur les résidus*

On effectue le test de Shapiro - Wilk sur les résidus des modèles $ARMA(p,q)$ estimés. Sous Python, la fonction shapiro de la librairie Scipy effectue un test de normalité basé sur la statistique W de Shapiro - Wilk.

```
# Test de normalité de Shapiro - Wilk
def ShapiroWilkTest(res):
  statistic,pvalue = st.shapiro(res.resid)[0],st.shapiro(res.resid)[1]
  return list([statistic, pvalue])

# Application
```

```
sw_res = pd.DataFrame(index=["W","pvalue"],columns = label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  sw_res[lab] = ShapiroWilkTest(model)
sw_res.iloc[1,:] = scientific_format(digits=3)(sw_res.iloc[1,:])
```

Table 10.20 – Test de Shapiro - Wilk sur la série des résidus des différents modèles estimés

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
W	0.824774	0.8090302	0.8447028	0.8058081	0.7426497	0.7617677
pvalue	2.487e-14	5.350e-15	2.014e-13	3.950e-15	1.925e-17	8.633e-17

Toutes les pvalues sont nulles, par conséquent on rejette l'hypothèse nulle de normalité des résidus.

10.3.2.2 Test de Shapiro - Francia

Le test de Samuel S. Shapiro and Francia (1972) est une version simplifiée du test de Samuel Sanford Shapiro and Wilk (1965) en remplaçant les coefficients $(a_1, \dots, a_{[T/2]})$ par les quantiles normaux. Son principal apport est de permettre d'appliquer l'idée du test de Shapiro - Wilk sur des échantillons de taille plus grande ainsi qu'un gain de puissance en présence de certaines confifuations des données. La formule de la statistique de test de Shapiro - Francia est :

$$z = \frac{1}{\sigma}\left[\log\left(1-\rho^2_{\hat{\varepsilon},Q}\right)-\mu\right] \tag{10.53}$$

avec :

— T est la taille d'éhantillon de $\hat{\varepsilon}$;
— $Q = (m_1, \dots, m_T)$, vecteurs des moments d'ordre 1 à T dans le cadre d'une loi normale et de formule de fonction quantile :

$$\forall i \in [1,T], \quad m_i = F^{-1}_{\hat{\varepsilon}}\left(\frac{i-0.375}{T+0.25}\right) \tag{10.54}$$

— $\rho_{\hat{\varepsilon},Q}$, coefficient de corrélation de Pearson entre la variable $\hat{\varepsilon}$ et Q ;

$$\rho_{\hat{\varepsilon},Q} = \frac{\text{Cov}(\hat{\varepsilon},Q)}{\sigma_{\hat{\varepsilon}}\sigma_Q} = \frac{\sum_{i=1}^{i=T}\left(\hat{\varepsilon}_{(i)}-\overline{\hat{\varepsilon}}\right)(m_i-\overline{m})}{\sqrt{\sum_{i=1}^{i=T}\left(\hat{\varepsilon}_{(i)}-\overline{\hat{\varepsilon}}\right)^2}\sqrt{\sum_{i=1}^{i=T}(m_i-\overline{m})^2}} \tag{10.55}$$

— μ et σ sont définies par les formules suivantes :

$$\begin{cases}\mu &= -1.2725 + 1.0521 \times [\log(\log(T)) - \log(T)] \\ \sigma &= 1.0308 - 0.26758 \times \left[\log(\log(T)) - \dfrac{2}{log(T)}\right]\end{cases} \tag{10.56}$$

La statistique de test de Samuel S. Shapiro and Francia (1972) suit une loi normale centrée - réduite et l'hypothèse nulle H_0 est :

$$\hat{\varepsilon} \quad \text{suit une loi normale} \quad F_{\hat{\varepsilon}} = F_{L(\mu,\sigma)} \tag{10.57}$$

Afin d'être comparable au test de Shapiro - Wilk, celui de Shapiro - Francia se base également sur une approche unilatérale à gauche. Soit $z_{1-\alpha}$ la valeur seuil de la distribution de la statistique de test z pour un niveau de confiance α, l'hypothèse alternative est alors :

$$H_1 : F_{\hat{\varepsilon}} \neq F_{L(\mu,\sigma)}, \quad \text{soit} \quad z > z_{1-\alpha} \tag{10.58}$$

La loi à laquelle reporter la statistique de test z de Samuel S. Shapiro and Francia (1972) est celle de la loi normale centrée et réduite. Etant donné que sa fonction de répartition est basée sur la fonction erreur de Gauss, notée $erf(.)$, définie par :

$$\text{erf}(x) = \frac{2}{\pi}\int_0^x \text{e}^{-t^2}\text{d}t = 2\Phi(x\sqrt{2}) - 1 \tag{10.59}$$

il conviendra dpasser par la méthode analytique proposée par Abramowitz et Stegun et permettant une estimation fiable à 10^{-7} près. Soit le changement de variable suivant :

$$t = \frac{1}{1 + 0.2316419z} \tag{10.60}$$

La formule d'usage et faisant intervenir aussi bien z que sa transformée t est :

$$p = 1 - F_{L(0,1)}(Z) = 1 - \frac{1}{\sqrt{2\pi}}\text{e}^{-z^2/2}\sum_{k=1}^{k=5} t^k C_k \tag{10.61}$$

avec

Table 10.21 – Valeurs des coefficients de pondération

Pondération	C_1	C_2	C_3	C_4	C_5
valeur	0.319381530	−0.356563782	1.781477937	−1.821255978	1.330274429

Par construction analogue au test de Shapiro - Wilk, le test de Samuel S. Shapiro and Francia (1972) est unialtéral. La statistique de test z dépend du coefficient de corrélation $\rho^2_{\hat{\varepsilon},Q}$ entre $\hat{\varepsilon}$ et les quantiles théoriques. Ainsi :

- Si $\rho^2_{\hat{\varepsilon},Q} \to 0$, ce qui implique que $z \to \frac{1}{\sigma}\left[\log(1-0) - \mu\right] = -\frac{\mu}{\sigma}$, correspondant au cas où $\hat{\varepsilon}$ ne suit pas une loi normale.
- Si $\rho^2_{\hat{\varepsilon},Q} \to 1$, ce qui implique que $z \to \frac{1}{\sigma}\left[\log(1-1) - \mu\right] = -\infty$, correspondant au cas où $\hat{\varepsilon}$ suit une loi normale.

La distribution normale peut alors être vue comme une pente de coefficient 1 sur un diagramme $Q-Q$ plot et la statistique de test de Shapiro - Francia permet de mesurer de combien les données s'éloigent de cette droite.

Exemple 10.19 *Application du test de Shapiro - Francia*

Nous créeons une fonction ppoints(n,a) qui génère la séquence de points de probabilité à partir de la formule suivante :

$$(1 : m - a)/(m + (1 - a) - a) \tag{10.62}$$

avec `m=n` si `len(n)==1` ou `len(n)`. `1:m` représente la séquence des nombres compris entre 1 et m.

```
# Génération d'une séquence de point de probabilité
def ppoints(n,a=3/8):
  if n > 0 :
    return (np.arange(1,n+1,dtype=int)-a)/(n + 1-2*a)
  else:
    return float()

# Test de Shapiro - Francia
def ShapiroFranciaTest(res):
  x = res.resid
  x = np.sort(x.dropna())
  n = len(x)
  if n < 5 or n > 5000:
    raise Exception("sample size must be between 5 and 5000")
  y = st.norm.ppf(ppoints(n,a=3/8))
  W  = np.square(np.corrcoef(x, y)[0,1])
  u = np.log(n)
  v = np.log(u)
  mu = -1.2725 + 1.0521*(v - u)
  sig = 1.0308 - 0.26758*(v + 2/u)
  z =  (np.log(1 - W) - mu)/sig
  pval = st.norm.sf(z)
  return list([W,z,pval])

# Application
sf_res = pd.DataFrame(index=["W","z","pvalue"],columns = label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  sf_res[lab] = ShapiroFranciaTest(model)
sf_res.iloc[2,:] = scientific_format(digits=3)(sf_res.iloc[2,:])
```

Table 10.22 – Test de Shapiro - Francia sur la série des résidus des différents modèles estimés

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
W	0.8138314	0.797273	0.8342681	0.7942206	0.7285907	0.7487594
z	7.076017	7.252312	6.835429	7.283233	7.855984	7.696221
pvalue	7.418e-13	2.049e-13	4.088e-12	1.630e-13	1.983e-15	7.007e-15

Toutes les pvalues sont nulles, par conséquent on rejette l'hypothèse nulle de normalité des résidus.

10.3.2.3 Test de Kolmogorov - Smirnov

Le test de Kolmogorov-Smirnov est une approche non paramétrique permettant de tester l'adéquation de la distribution d'une variable continue $\hat{\varepsilon}$ à une loi de distribution fixée. Soit F_T et F respectivement les fonctions de répartition observées et théorique de $\hat{\varepsilon}$. La statistique D du test de Kolmogorov - Smirnov est basée sur la distance maximale entre F et F_T et est définie par :

$$D = \max_{i=1,\dots,T} |F(i) - F_T(i)| = \max_{i=1,\dots,T} \left(\left| F(i) - \frac{i-1}{T} \right|, \left| \frac{i}{T} - F(i) \right| \right) \tag{10.63}$$

Cette statistique de test suit une loi de « Kolmogorov - Smirnov » de paramètre T et l'hypothèse nulle H_0 est que « $\hat{\varepsilon}$ suit la loi que l'on a fixée $F_{\hat{\varepsilon}} = F$ ».

Soit $D_{T,1-\alpha}$ la valeur seuil de la distribution de la statistique de test pour un niveau de confiance α et pour le paramète T, les hypothèses alternatives sont alors :

— $H_1 : F < F_T$, soit $D^+ = \max\limits_{i=1\ldots,T} \left(\dfrac{i}{T} - F(i) \right) > D^+_{T,1-\alpha}$, pour un test unilatéral à doite.

— $H_1 : F > F_T$, soit $D^- = \max\limits_{i=1,\ldots,T} \left(F(i) - \dfrac{i-1}{T} \right) > D^-_{T,1-\alpha}$, pour un test unilatéral à gauche.

— $H_1 : F \neq F_T$, soit $D = \max\limits_{i=1,\ldots,T} \left(\left| F(i) - \dfrac{i-1}{T} \right|, \left| \dfrac{i}{T} - F(i) \right| \right) > D_{T,1-\alpha}$, pour un test bilatéral.

La pvalue p associée à la statistique de test D_{obs} de Kolmogorov - Smirnov de détermine au travers de la formule suivante :

$$p = \mathbb{P}\left(D_{\text{obs}} \geq D_{T,1-\alpha}\right) = 1 - 2\sum_{k=1}^{+\infty}(-1)^{k+1}\mathrm{e}^{-2k^2TD^2_{\text{obs}}} \tag{10.64}$$

Ce calcul itératif converge rapidement, notamment à partir de $k \geq 3$ itérations. La table des valeurs critiques $D_{T,1-\alpha}$ pour les petites valeurs de T et différentes valeurs de α doivent être utilisées. Lorsque les effectifs sont élevés, typiquement $T > 100$, il est possible d'approcher la valeur critique à l'aide de formules simples (Tableau 10.23) :

Table 10.23 – Valeurs critiques de Kolmogorov - Smirnov pour $T > 100$

α	1%	2%	5%	10%	20%
$D_{T>100,1-\alpha}$	$1.629/\sqrt{T}$	$1.518/\sqrt{T}$	$1.358/\sqrt{T}$	$1.223/\sqrt{T}$	$1.073/\sqrt{T}$

Exemple 10.20 *Application du test de Kolmogorov - Smirnov*

Sous Python, la fonction kstest de Scipy teste la normalité des résidus à partir du test de Kolmogorov - Smirnov.

```
# Test de Kolmogorov - Smirnov
def KolmogorovSmirnovTest(res):
  mean, std = res.resid.mean(), res.resid.std(ddof=1)
  kstest,kspvalue = st.kstest(rvs=res.resid,cdf="norm",args=(mean,std))
  return list([kstest, kspvalue])

# Application
kstest_res = pd.DataFrame(index=["KS stat","pvalue"],columns = label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  kstest_res[lab] = KolmogorovSmirnovTest(model)
kstest_res.iloc[1,:] = scientific_format(digits=3)(kstest_res.iloc[1,:])
```

Toutes les pvalues sont nulles, on rejette l'hypothèse nulle de normalité des résidus.

10.3.2.4 Test de Lilliefors

Dans le cas de l'adéquation à la loi normale, le test de Kolmogorov - Smirnov ne peut être utiliser que si les paramètres m et σ sont connus. Or, lorsque ce n'est pas le cas, la seule solution

Table 10.24 – Test de Kolmogorov - Smirnov sur la série des résidus des différents modèles estimés

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
KS stat	0.1656434	0.1737443	0.1459043	0.1534226	0.194347	0.1723371
pvalue	2.595e-05	8.387e-06	3.234e-04	1.286e-04	3.696e-07	1.025e-05

restante (outre celle d'avoir recours à un autre test) est d'estimer directement dépuis $\hat{\varepsilon}$: $m_{\hat{\varepsilon}}$ et $\sigma_{\hat{\varepsilon}}$. Ce qui enntraînera une lourde perte de puissance. Pour y pallier, on utilisera alors une table de distribution mieux adaptée, celle de Lilliefors. Cette procédure porte alors le nom de test de Lilliefors (1967). L'hypothèse nulle reste la même que pour la version de Kolmogorov - Smirnov.

Le calcul de la p-value p va se baser sur la même statistique de test D à laquelle on va appliquer quelques modifications. L'algorithme de Lilliefors (1967) peut être présenté comme suit :

- **Etape 1 :** Estimation des paramètres $m_{\hat{\varepsilon}}$ et $\sigma_{\hat{\varepsilon}}$ directement depuis $\hat{\varepsilon}$.
- **Etape 2 :** Calcul de la statistique de test D de Kolmogorov.
- **Etape 3 :** Si $T \leq 100$, alors on conserve $D_{\text{obs}} = D$ en l'état et on fixe $\delta = T$, sinon on lui applique l'ajustement suivant :

$$D_{\text{obs}} = D \times \left(\frac{T}{100}\right)^{0.49} \quad \text{et on fixe} \quad \delta = 100 \tag{10.65}$$

- **Etape 4 :** On peut désormais calculer la p-value exacte :

$$\ln p = -7.01256 \times D_{\text{obs}}^2(\delta + 2.79019) + 2.99587 \times D_{\text{obs}}\sqrt{\delta + 2.78019} - 0.122119 + \frac{0.974598}{\sqrt{\delta}} + \frac{1.67997}{\delta} \tag{10.66}$$

- **Etape 5 :** Si $0 \leq p \leq 0.1$, alors on conserve p, sinon on l'ajuste à son tour en posant $\widetilde{D}_{\text{obs}} = D\left(\sqrt{T} - 0.01 + \frac{0.85}{\sqrt{T}}\right)$ et :
  - Si $\widetilde{D}_{\text{obs}} \leq 0.302$ alors $p = 1$.
  - Si $\widetilde{D}_{\text{obs}} > 1.31$ alors $p = 0$
  - Pour les autres cas, p prend la forme suivante :

$$p = \sum_{j=0}^{j=4} \theta_j \widetilde{D}_{\text{obs}}^j, \quad \text{avec} \quad \theta_0, \ldots, \theta_4 \in \mathbb{R} \tag{10.67}$$

Les valeurs du vecteur $\theta = (\theta_0, \theta_1, \ldots, \theta_4)^{'}$ sont disponibles dans le tableau (10.25) :

Table 10.25 – Valeurs du vecteur θ pour le calcul de la pvalue (test de Lilliefors)

$D_{\text{obs}}^{'}$	θ_0	θ_1	θ_2	θ_3	θ_4
$]0.302; 0.5]$	2.76773	−19.828315	80.709644	−138.55152	81.218052
$]0.5; 0.9]$	−4.901232	40.662806	−97.490286	94.029866	32355711
$]0.9; 1.31]$	6.198765	−19.558097	23.186922	−12.234627	2.423045

La distribution de Lilliefors (1967) est stochastiquement plus petite que la distribution de Kolmogorov-Smirnov et a été calculée uniquement par la méthode de Monte-Carlo. En effet le test de Lilliefors (1967) prend en compte le fait que la fonction de répartition est plus proche des données empiriques que ce qu'elle devrait être, étant donné qu'elle repose sur une estimation réalisée sur les données empiriques. Le maximum de variance est donc plus faible que ce qu'il devrait être si l'hypothèse nulle avait été testée sur une distribution normale de paramètres connus. La région critique du test est la même que celle de Kolmogorv - Smirnov.

Comme pour le test de Kolmogorov - Smirnov, la table des valeurs critiques $D_{T,1-\alpha}$ pour les petites valeurs de T et différents valeurs de α doivent être utilisées. Lorsque les effectifs sont élevés, typiquement $T > 30$, il est possible d'approcher la valeur critique à l'aide de formules simples (Tableau 10.26) :

Table 10.26 – Valeurs critiques du test de Lilliefors pour $T > 30$

α	1%	5%	10%	15%	20%
$D_{T>30,1-\alpha}$	$1.031/\sqrt{T}$	$0.886/\sqrt{T}$	$0.805/\sqrt{T}$	$0.768/\sqrt{T}$	$0.736/\sqrt{T}$

Remarque 10.4 *Calcul de la pvalue*

Abdi and Molin (2007) fournissent des approximations plus précises pour le calcul de la pvalue. Surtout, ils fournissent une formule assez complexe pour calculer la probabilité critique du test de Lilliefors (1967). Ce qui simplifie beaucoup la procédure car il suffit de comparer cette probabilité critique avec le risque α que l'on a choisi. Lorsque les effectifs sont élevés, typiquement $T > 50$, il est possible d'approcher la valeur critique à l'aide de formules simples (Tableau 10.27) :

Table 10.27 – Valeurs critiques du test de Lilliefors pour $T > 50$ founies par Abdi and Molin (2007)

α	1%	5%	10%	15%	20%
$D_{T>50,1-\alpha}$	$1.035/f(T)$	$0.895/f(T)$	$0.819/f(T)$	$0.775/f(T)$	$0.741/f(T)$

avec $f(T) = \dfrac{0.83+T}{\sqrt{T}} - 0.01$.

Exemple 10.21 *Application du test de Lilliefors*

Sous Python, la fonction kstest_normal teste la normalité des résidus à partir du test de Kolmogorov - Smirnov au sens de Lilliefors. Nous devons fixer le paramètre `pvalmethod = "approx"` afin que notre fonction utilise la formule approximative de Dallal and Wilkinson (1986) et que nos résultats soient conformes à ceux fournis par la fonction lillie.test de la librairie nortest de R.

```
# Test de normalité de Kolmogorov-Smirnov au sens de Lilliefors
from statsmodels.stats.diagnostic import kstest_normal
def ModifiedKolmogorovSmirnovTest(res):
  kstest,kspvalue = kstest_normal(res.resid,dist="norm",pvalmethod="approx")
  return list([kstest, kspvalue])

# Application
mkstest_res = pd.DataFrame(index=["Modified KS stat","pvalue"],columns = label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  mkstest_res[lab] = ModifiedKolmogorovSmirnovTest(model)
mkstest_res.iloc[1,:] = scientific_format(digits=3)(mkstest_res.iloc[1,:])
```

Table 10.28 – Test de Kolmogorov - Smirnov au sens de Lilliefors sur la série des résidus des différents modèles estimés

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
Modified KS stat	0.1656434	0.1737443	0.1459043	0.1534226	0.194347	0.1723371
pvalue	9.411e-15	2.572e-16	2.756e-11	1.504e-12	1.163e-20	4.872e-16

On peut également utiliser la fonction lilliefors afin de tester la normalité des résidus à partir du test de Lilliefors.

```
# Test de normalité de Kolmogorov-Smirnov au sens de Lilliefors
from statsmodels.stats.diagnostic import lilliefors
def LillieforsTest(res):
  lilltest,lillpvalue = lilliefors(res.resid,dist="norm",pvalmethod="approx")
  return list([lilltest, lillpvalue])

# Application
lilltest_res = pd.DataFrame(index=["Modified KS stat","pvalue"],columns = label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  lilltest_res[lab] = LillieforsTest(model)
lilltest_res.iloc[1,:] = scientific_format(digits=3)(lilltest_res.iloc[1,:])
```

Table 10.29 – Test de Kolmogorov - Smirnov au sens de Lilliefors sur la série des résidus des différents modèles estimés

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
Modified KS stat	0.1656434	0.1737443	0.1459043	0.1534226	0.194347	0.1723371
pvalue	9.411e-15	2.572e-16	2.756e-11	1.504e-12	1.163e-20	4.872e-16

Nous avons des p-values nulles, on rejette l'hypothèse de normalité des résidus des modèles $ARMA(p,q)$ estimés.

10.3.2.5 Test de Anderson Darling

Le test de Anderson - Darling (*cf.* Stephens (1970), Stephens (1974)) est une variante du test de Kolmogorov - Smirnov, à la différence qu'elle donne plus d'importance aux queues de distribution. De ce point de vue, elle est plus indiquée dans la phase d'évaluation des données précédant la mise en oeuvre d'un test paramètrique (comparaison de moyenne, de varainces, etc.).

Une autre particularité de ce test est que ses valeurs critiques sont tabulées différemment selon la loi théorique de référence, un coefficient multiplicatif correctif dépendant de la taille d'échantillon n peut être aussi introduit.

Concernant l'adéquation à la loi normale, la statistique du test s'écrit :

$$A = -T - \frac{1}{T}\sum_{i=1}^{i=T}(2i-1)\left[\log(F_i) + \log\left(1 - F_{n-i+1}\right)\right] \tag{10.68}$$

où F_i est la fréquence théorique de la loi de répartition normale centrée et réduite associée à la valeur standardisée $\hat{\varepsilon}^*_{(i)} = \dfrac{\hat{\varepsilon}_{(i)} - \mu_{\hat{\varepsilon}}}{\sigma_{\hat{\varepsilon}}}$.

Afin de déterminer la pvalue associée à la statistique de test A, il faut appliquer une transformation sélon :

$$\widetilde{A} = A\left(1 + \frac{0.75}{T} + \frac{2.25}{T^2}\right) \tag{10.69}$$

Trois écoles existent :

— la première école par du principe que cet ajustement doit se faire uniquement si les paramètres $m_{\hat{\varepsilon}}$ et $\sigma_{\hat{\varepsilon}}$ sont inconnus, sinon il faut se baser sur A directement ;

— la deuxième école qui part du principe que cet ajustement doit se faire uniquement si $T \leq 40$;
— Enfin, la dernière qui part du principe qu'il faut appliquer cet ajustement quelque soit la situation.

La pvalue est calculée à partir de la statistique $\tilde{A}$ par interpolation à partir d'une table décrite dans Leslie, Stephens, and Fotopoulos (1986), R. D'Agostino (2017), Stephens (2017). Elle prend la forme suivante :

$$p = \beta_0 + \beta_1 \mathrm{e}^{\sum_{j=0}^{j=2} \theta_j \widetilde{A}^j} \quad \text{avec} \quad \beta_0, \beta_1, \theta_0, \theta_1, \theta_2 \in \mathbb{R} \tag{10.70}$$

Le tableau 10.30 donne les valeurs du vecteur $\theta = (\beta_0, \beta_1, \theta_0, \theta_1, \theta_2)$ pour le calcul de la pvalue :

Table 10.30 – Valeurs du vecteur θ pour le calcul de la pvalue (test de Anderson Darling)

$\widetilde{A}$	β_0	β_1	θ_0	θ_1	θ_2
$]; 0.2[$	1	−1	−13.436	101.14	−223.73
$[0.2; 0.34[$	1	−1	−8.318	42.796	−59.938
$[0.34; 0.6[$	0	1	0.9177	−4.279	−1.38
$[0.6; 10[$	0	1	1.2937	−5.709	0.0186

Si $\tilde{A} \geq 10$,alors $p < 0.0001$.

Exemple 10.22 *Application du test de Anderson Darling*

Sous Python, la fonction anderson de la librairie Scipy teste la normalité des résidus à partir du test de Anderson Darling

```
# Test de Anderson Darling
def AndersonDarlingTest(res,alpha = 0.05):
  T = len(res.resid)
  A,_,_ = st.anderson(res.resid, dist='norm')
  AA = A*(1+0.75/T+2.25/T**2)
  if AA < 0.2 :
    pvalue = 1 - np.exp(-13.436 + 101.14*AA - 223.73*AA**2)
  elif AA < 0.34:
    pvalue = 1 - np.exp(-8.318 + 42.796*AA - 59.938*AA**2)
  elif AA < 0.6 :
    pvalue = np.exp(0.9177 - 4.279*AA - 1.38*AA**2)
  elif AA < 10 :
    pvalue = np.exp(1.2937 - 5.709*AA + 0.0186*AA**2)
  else:
    pvalue = 3.7e-24
  return list([round(A,4),round(AA,4),pvalue])

# Application
anderson_res = pd.DataFrame(index=["A stat","AA stat","pvalue"],columns = label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  anderson_res[lab] = AndersonDarlingTest(model)
anderson_res.iloc[2,:] = scientific_format(digits=3)(anderson_res.iloc[2,:])
```

Les pvalues étant nulles, on rejette l'hypothèse nulle de normalité des résidus.

Table 10.31 – Test de Anderson Darling sur la série des résidus des différents modèles estimés

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
A stat	7.4545	8.3469	6.2336	8.3136	11.6287	10.8461
AA stat	7.4826	8.3783	6.2571	8.345	11.6726	10.887
pvalue	2.897e-18	2.269e-20	2.313e-15	2.716e-20	3.700e-24	3.700e-24

10.3.2.6 Test de D'Agostino - Pearson

Le test de D'Agostino - Pearson, connu également sous l'appellation test K^2 de D'Agostino - Pearson, ou test de R. B. D'Agostino (1970) est une approche non paramétrique permettant de tester si une variable continue X suit une loi normale. Il est basé sur les coefficients d'asymétrie et d'aplatissement. Lorsque ces deux indicateurs diffèrent simultanément de la valeur de référence 0, on conclut que la distribution empirique n'est pas compatible avec la loi normale. L'enjeu est de construire une combinaison efficace de ces indicateurs.

L'idée est très simple à comprendre, sa puissance est considérée comme très bonne au point que son auteur préconise de le substituer aux tests basés sur la statistique de Kolmogorov - Smirnov. Le test de D'Agostino présenterait une puissance similaire à celle de Shapiro - Wilk à mesure que les effectifs augmentent. Il devient particulièrement efficace à partir de $T \geq 20$, on le préfère alors aux tests basés sur la statistique de Kolmogorov - Smirnov. Par rapport au test de Shapiro - Wilk, il serait de surcroît peu sensible à l'existence des ex-aequo dans l'échantillon.

Si l'idée est simple, les formules sont relativement complexes. Il faut procéder par étapes. Le fil directeur est de centrer et réduire les deux coefficients (asymétrie et aplatissement) de manière à obtenir des valeurs γ_1 et γ_2 distribuées asymptotiquement selon une loi normale $\mathcal{N}(0,1)$. La transformation intègre des corrections supplémentaires de manière à rendre l'approximation normale plus efficace.

Le test K^2 de D'Agostino - Pearson se base sur le coefficient d'asymétrie centré - réduit :

$$\gamma_1 = D \times \ln\left(C + \sqrt{C^2+1}\right) \tag{10.71}$$

où :

— $A = \dfrac{\dfrac{1}{T}\displaystyle\sum_{t=1}^{t=T}(\widehat{\varepsilon}_t - \overline{\widehat{\varepsilon}})^3}{\left(\dfrac{1}{T}\displaystyle\sum_{t=1}^{t=T}(\widehat{\varepsilon}_t - \overline{\widehat{\varepsilon}})^2\right)^{3/2}}$

— $B = \dfrac{3(T^2+27T-70)(T+1)(T+3)}{(T-2)(T+5)(T+7)(T+9)}$

— $C = \dfrac{A\sqrt{\dfrac{(T+1)(T+3)}{6(T-2)}}}{\sqrt{\dfrac{2}{\sqrt{2(B-1)}-2}}}$

— $D = \dfrac{1}{\sqrt{\ln\left(\sqrt{\sqrt{2(B-1)}-1}\right)}}$

Et le coefficient d'aplatissemnt centré - réduit :

$$\gamma_2 = \frac{\left(1 - \frac{2}{9K}\right) - L^{1/3}}{\sqrt{\frac{2}{9K}}} \tag{10.72}$$

où :

- $g_2 = \frac{T(T+1)}{(T-1)(T-2)(T-3)} \sum_{t=1}^{t=T} \left(\frac{\hat{\varepsilon}_t - \bar{\hat{\varepsilon}}}{s} \right)^4 - \frac{3(T-1)^2}{(T-2)(T-3)}$
- $G = \frac{24T(T-2)(T-3)}{(T+1)^2(T+3)(T+5)}$
- $H = \frac{(T-2)(T-3)g_2}{(T+1)(T-1)\sqrt{G}}$
- $J = \frac{6(T^2-5T+2)}{(T+7)(T+9)} \sqrt{\frac{6(T+3)(T+5)}{T(T-2)(T-3)}}$
- $K = 6 + \frac{8}{J}\left[\frac{2}{J} + \sqrt{1 + \frac{4}{J^2}}\right]$
- $L = \left(\frac{1 - \frac{2}{K}}{1 + H\sqrt{\frac{2}{K-4}}} \right)$

γ_1 et γ_2 suivent tous deux asymptotiquement une loi normale $\mathcal{N}(0,1)$. La statistique du test K^2 de D'Agostino - Pearson, connue également sous l'appelation de statistique omnibus K^2, est :

$$K^2 = \gamma_1^2 + \gamma_2^2 \tag{10.73}$$

Sous l'hypothèse nulle de normalité des résidus, cette statistique suit asymptotiquement une loi de χ^2 à 2 degrés de libertés. L'incompatibilité de la distribution évalués avec la loi normale est d'autant plus marquée que la statistique K^2 prend une valeur élevée. Pour un risque α, la région critique du test s'écrit : $K2 > \chi^2_{1-\alpha}(2)$. Pour $\alpha = 0.05$, le seuil critique est $\chi^2_{0.95}(2) = 5.99$.

Exemple 10.23 *Application du test de D'Agostino - Pearson*

On souhaite tester la normalité des résidus des modèles $ARMA(p,q)$ estimésà l'aide du test de D'Agostino - Pearson. Sous Python, la fonction normaltest effectue un test de normalité basé sur la statistique de D'Agostino - Pearson.

```
# Test de normalité d'Agostino - Pearson
def AgostinoPearsonTest(res):
  statistic, pvalue = st.normaltest(res.resid)[0],st.normaltest(res.resid)[1]
  return list([round(statistic,4), pvalue])

# Application
dp_res = pd.DataFrame(index=["K2","pvalue"],columns = label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  dp_res[lab] = AgostinoPearsonTest(model)
dp_res.iloc[1,:] = scientific_format(digits=3)(dp_res.iloc[1,:])
```

Tous les pvalues sont nulles, par conséquent on rejette l'hypothèse nulle de normalité des résidus.

Table 10.32 – Test de D'Agostino - Pearson sur la série des résidus des différents modèles estimés

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
K2	59.608	76.0522	48.1173	57.2706	139.4873	110.5127
pvalue	1.138e-13	3.058e-17	3.560e-11	3.663e-13	5.137e-31	1.006e-24

Remarque 10.5

Le reproche usuellement adressé au test de D'Agostino est qu'il ne permet pas directement de comprendre la nature de la déviation de la loi normale en cas de rejet de l'hypothèse nulle. Il faut compléter l'analyse avec l'étude individuelle des coefficients, ou en mettant en oeuvre les techniques descriptives décrites précédemment.

10.3.2.7 Test de Cramer - Von Mises

Le test de Cramer - von Mises, également nommé critère de Cramer - von Mises, est une approche non paramétrique permettant de tester si une variable continue X suit une loi de distribution fixée.On parle alors de test d'ajustemen de Cramer - von Mises.

Le test de Cramer - von Mises peut être vu comme une version plus puissante du test de Kolmogorov - Smirnov. Les deux tests se basent sur l'écart entre les deux fonctions de répartition que l'on souhaite comparer, la seule différence étant que le second utilise la valeur maximale de cette différence tandis que le premier utilisera la somme des différences. Ainsi, si le test de Kolmogorov - Smirnov est sensible aux outliers, le second l'est beaucoup moins voire pas du tout.

Soit F_n et F^* les fonctions de répartition que l'on cherche à comparer et, respectivement, associéesà X (distribution empirique observée) et à la loi de distribution de référence fixée (distribution théorique). La statistique du test d'ajustement de Cramer - von Mises est :

$$W_T = T \int_{\mathbb{R}} \left[F_T(\hat{\varepsilon}) - F^*(\hat{\varepsilon})\right]^2 \mathrm{d}F^*(\hat{\varepsilon}) \tag{10.74}$$

que l'on peut alors simplifier en

$$W_T = \frac{1}{12T} + \sum_{i=1}^{i=T} \left[\frac{F(\hat{\varepsilon}^*_{(i)}) - (2i-1)}{2T}\right]^2 \tag{10.75}$$

avec $\hat{\varepsilon}^*_{(i)} = \dfrac{\hat{\varepsilon}_{(i)} - \overline{\hat{\varepsilon}}}{s_{\hat{\varepsilon}}}$ valeur standardisée de $\hat{\varepsilon}_{(i)}$.

Cette statistique de test suit une loi de « Cramer - von Mises » à T degrés de liberté et l'hypothèse nulle H_0 est :

$$\hat{\varepsilon} \quad \text{suit la loi que l'on a fixée} \quad /F_{\hat{\varepsilon}} = F \tag{10.76}$$

Avec $W_{T,1-\alpha}$ la valeur seuil de la distribution de la statistique de test pour un niveau de confiance α et pour le paramètre T, les hypothèses alternatives sont alors :

— $H_1 : F < F_T$, soit $W_T > W_{T,1-\alpha}$, pour un test unilatéral à gauche ;
— Le test n'admet pas de version bilatéral ou unilatéral à droite.

La loi à laquelle reporter la statistique de test du W est celle de Cramer - von Mises à n degrés de liberté. La formule (théorique) de calcul de la p - value est :

$$p = \lim_{T \longrightarrow +\infty} \mathbb{P}\left(W_T \leq x\right) = 1 - \frac{1}{\pi} \sum_j (-1)^{j-1} \int_{(2j-1)^2\pi^2}^{4j^2\pi^2} \sqrt{\frac{-\sqrt{y}}{\sin(\sqrt{y})}} \frac{\mathrm{e}^{-xy/2}}{y} \mathrm{d}y \tag{10.77}$$

que l'on peut aisément approcher par la méthode de MCMC via l'algorithme suivant pour B rééchantillonnages :

$$\forall b \in \{1, B\}, \quad V_b = \frac{1}{12n} + \sum_{k=1}^{k=T} \left[\frac{2k-1}{2T} - Z^b_{k,L(\cdots)}\right]^2 \tag{10.78}$$

avec $Z^b_{L(\cdots)}$ la variable aléatoire tirée à nouveau pour chaque rééchantillonnage b selon la loi fixée L de paramètre(s) $(\cdots)$. La p - value est alors obtenue par la formule :

$$p = \frac{\mathrm{Card}(V > W_T)}{B} \tag{10.79}$$

Remarque 10.6

Les auteurs définissent une nouvelle statistique $\tilde{W}_T$ basée sur W_T comme suit :

$$\widetilde{W}_T = W_T \left(1 + \frac{0.5}{T}\right) \tag{10.80}$$

La pvalue est calculée à partir de la statistique $\tilde{A}$ par interpolation et prend la forme suivante :

$$p = \beta_0 + \beta_1 \mathrm{e}^{\sum_{j=0}^{j=2} \theta_j \widetilde{W}_T^j} \qquad \text{avec} \quad \beta_0, \beta_1, \theta_0, \theta_1, \theta_2 \in \mathbb{R} \tag{10.81}$$

Le tableau (10.33) donne les valeurs du vecteur $\theta = (\beta_0, \beta_1, \theta_0, \theta_1, \theta_2)$ pour le calcul de la pvalue :

Table 10.33 – Valeurs du vecteur θ pour le calcul de la pvalue (test de Anderson Darling)

$\widetilde{W}_T$	β_0	β_1	θ_0	θ_1	θ_2
$]; 0.0275[$	1	−1	−13.953	775.5	−12542.61
$[0.0275; 0.051[$	1	−1	−5.903	179.546	−1515.29
$[0.051; 0.092[$	0	1	0.886	−31.62	10.897
$[0.092; [$	0	1	1.111	−34.242	12.832

Si la pvalue est supérieure à 1, alors la statistique de test W_T est inexistante.

Exemple 10.24 *Application du test de Cramer - Von Mises*

Sous Python, la fonction cramervonmises teste la normalité des résidus à partir du test de Cramer - Von Mises. Cependant, la pvalue ne corrobore pas celle définie par le tableau (10.33).

```
# Test de Cramer - Von Mises
def CramerVonMisesTest(res):
  mean, std = res.resid.mean(), res.resid.std(ddof=1)
  cramer = st.cramervonmises(res.resid,cdf=st.norm.cdf,args=(mean,std))
```

```
  W = cramer.statistic
  T = len(res.resid)
  WW = (1 + 0.5/T) * W
  if WW < 0.0275 :
    pvalue = 1 - np.exp(-13.953 + 775.5*WW - 12542.61*WW**2)
  elif WW < 0.051 :
    pvalue = 1 - np.exp(-5.903 + 179.546*WW - 1515.29*WW**2)
  elif WW < 0.092 :
    pvalue = np.exp(0.886 - 31.62*WW + 10.897*WW**2)
  else:
    pvalue = np.exp(1.111 - 34.242*WW + 12.832*WW**2)
  if pvalue > 1 :
    pvalue = 1
    W = np.nan
  return list([round(W,4),round(WW,4),pvalue])

# Application
cvm_res = pd.DataFrame(index=["omega2","WW","pvalue"],columns = label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  cvm_res[lab] = CramerVonMisesTest(model)
cvm_res.iloc[2,:] = scientific_format(digits=3)(cvm_res.iloc[2,:])
```

Table 10.34 – Test de Cramer - von Mises sur la série des résidus des différents modèles estimés

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
omega2	1.1578	1.3422	0.9295	1.3114	2.0828	1.8869
WW	1.1607	1.3455	0.9318	1.3146	2.0879	1.8916
pvalue	5.364e-10	3.651e-10	2.914e-09	3.663e-10	5.338e-07	1.963e-08

Puisque les pvalues sont nulles, on rejette l'hypothèse de normalité des résidus.

Remarque 10.7

Pour voir de près la distribution empirique des résidus estimés $\widehat{\varepsilon}_t$, on peut tracer l'histogramme et la densité de distribution des résidus standardisés. La remise en cause de l'hypothèse de normalité de la série des résidus doit conduire à s'interroger sur l'existence de points extrêmes dans la série (considérés comme des points aberrants) et sur leur traitement. Si l'hypothèse de bruit blanc est rejeté sur la série résiduelle, autrement dit si on du bruit corrélé, la régression linéaire peut être remplacée par une méthode stochastique d'interpolation spatiale appelé `krigeage` en géostatistique.

La fonction res.plot_diagnostics nous donne certaines informations concernant la série des résidus standardisés : histogramme et densité, QQ - Plot normal, corrélogramme simple (Brockwell and Davis (2002), Brockwell and Davis (2009)).

```
# Diagnostics sur les résidus
model6.plot_diagnostics(lags=12,figsize=(16,6));
plt.show()
```

Lorsqu'on regarde la courbe de densité estimée des résidus, elle est plus pointue que la normale traduisant une distribution leptokurtique.

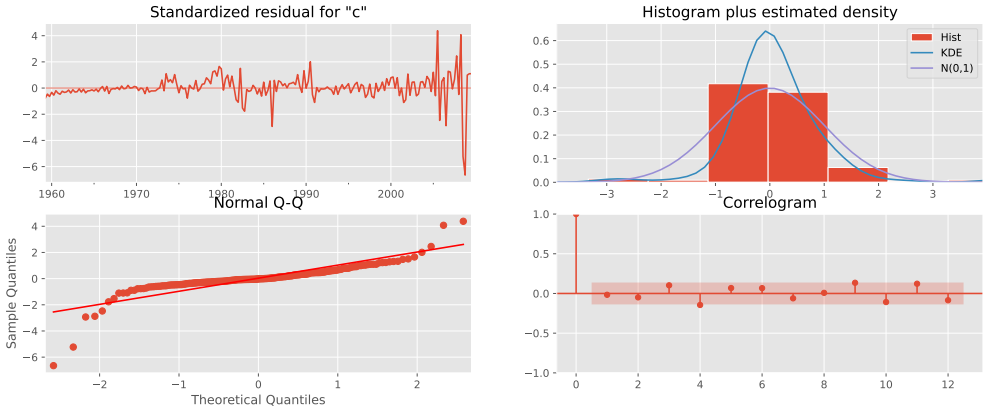


Figure 10.3 – Diagnostics des résidus

10.4 Critères de choix des modèles

Après examen des coefficients et analyse des résidus, certains modèles sont écartés. pour choisir le meilleur de ces modèles restants, pour la prévision de l'évolution future de la série à partir des données passées, on fait appel aux critères standards et aux critères d'information.

10.4.1 Critères standards

Ils sont fondés sur le calcul de l'erreur de prévision que l'on cherche à minimiser. On rappelle ici l'expression des trois critères les plus fréquemment utilisés.

10.4.1.1 Erreur absolue moyenne (MAE)

L'erreur absolue moyenne (MAE, *Mean Absolute Error*) fait référence à la valeur moyenne des valeurs d'erreur absolues calculées pour chaque point de l'ensemble de données. Elle est calculée à l'aide de l'équation suivante :

$$MAE = \frac{1}{T}\sum_{t=1}^{T}|\widehat{\varepsilon_t}| \tag{10.82}$$

L'erreur absolue est calculée pour chaque paire de valeurs prédites et réelles en prenant la valeur absolue de la différence entre les deux valeurs. Les termes d'erreur absolue calculés pour chaque paire de valeurs sont ensuite additionnés et la quantité résultante est divisée par le nombre total d'observations. La valeur résultante représente l'erreur absolue moyenne ou la distance verticale moyenne entre chaque paire de valeurs prévues et réelles lorsqu'elle est représentée graphiquement.

L'interprétation de MAE est simple car elle est représentée dans les mêmes unités que nos données d'origine. Un modèle parfait produit un MAE de zéro, et plus le MAE observé est proche de zéro, mieux le modèle correspond aux données. Le calcul de MAE traite toutes les pénalités de la même manière, que la valeur prévue soit inférieure ou supérieure à la valeur réelle.

```
# Mean absolute error
from sklearn.metrics import mean_absolute_error
```

```
mae = mean_absolute_error(default_data,res.fittedvalues)
print("Mean absolute error : %.4f" % (mae))
```

10.4.1.2 Racine de l'erreur quadratique moyenne (Root Mean Squared Error)

La racine de l'erreur quadratique moyenne (RMSE, *Root Mean Squared Error*) correspond à la racine de la moyenne de la somme des carrés des erreurs. Son expression est donnée par :

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\hat{\varepsilon}_t^2} \tag{10.83}$$

Cet indice fournit une indication par rapport à la dispersion ou la variabilité de la qualité de la prédiction. Le RMSE peut être relié à la variance du modèle. Souvent, les valeurs de RMSE sont difficiles à interpréter parce que l'on est pas en mesure de dire si une valeur de variance est faible ou forte. Pour pallier à cet effet, il est plus intéressant de normaliser le RMSE pour que cet indicateur soit exprimé comme un pourcentage de la valeur moyenne des observations. Cela peut être utilisé pour donner plus de sens à l'indicateur.

Par exemple, un RMSE de 10 est relativement faible si la moyenne des observations est de 500. Pourtant, un modèle a une variance forte s'il conduit à un RMSE de 10 alors que la moyenne des observations est de 15. En effet, dans le premier cas, la variance du modèle correspond à seulement 5% de la moyenne des observations alors que dans le second cas, la variance atteint plus de 65% de la moyenne des observations.

```
# Root mean squared error
from sklearn.metrics import mean_squared_error
rmse = mean_squared_error(default_data,res.fittedvalues,squared=False)
print("Root mean squared error : %.4f" % (rmse))
```

10.4.1.3 Écart absolu moyen en pourcentage (Mean Absolute Percent Error)

Son expression est :

$$MAPE = 100 \times \frac{1}{T}\sum_{t=1}^{T}\left|\frac{\hat{\varepsilon}_t}{X_t}\right| \tag{10.84}$$

où T est le nombre d'observations de la série X_t étudiée et $\hat{\varepsilon}_t$ désigne les résidus estimés.

```
# mean absolute percentage error
from sklearn.metrics import mean_absolute_percentage_error
mape = mean_absolute_percentage_error(default_data, model.fittedvalues))
print("Mean absolute percentage error : %.4f" % (mape))
```

Exemple 10.25 *Evaluation des modèles à l'aide des critères standards*

On affiche nos modèles suivant les critères standards.

```
# Critères standards
def StandardCriterion(res):
  return list([res.mae,res.mse,res.sse])

# Application
standard_res = pd.DataFrame(index=["MAE","MSE","SSE"],columns = label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  standard_res[lab] = StandardCriterion(model)
```

Table 10.35 – Critères standards

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
MAE	0.5270126	0.5134277	0.5465649	0.4986793	0.4458729	0.4476322
MSE	0.6763108	0.6704207	0.6893045	0.6397475	0.6405239	0.6258144
SSE	136.6147824	135.4249747	139.2395029	129.2289888	129.3858370	126.4145120

Remarque 10.8

Plus la valeur des ces critères est faible, plus le modèle estimé est proche des observations. La racine de l'erreur quadratique moyenne est très utile pour comparer plusieurs estimateurs, notamment lorsque l'un d'eux est biaisé. Si les deux estimateurs à comparer sont sans biais, l'estimateur le plus efficace est uniquement celui qui a la variance la plus faible.

10.4.2 Critères d'information

L'idée sous-jacente consiste à choisir un modèle sur la base d'une mesure de l'écart entre la vraie loi inconnue et le modèle estimé. Cette mesure peut être fournie par la qualité d'information de Kullback. Les différents critères ont alors pour objet d'estimer cette quantité d'information. Nous présentons ici les trois critères les plus fréquemment employés.

10.4.2.1 Critère d'information de Akaike

Présenté par Akaike (*cf.* Akaike (1973),Akaike (1974)) pour un modèle $ARMA(p,q)$, le critère AIC (*Akaike Informations Criterion*) pénalise les modèles pour lesquels l'ajout de nouvelles variables explicatives n'apporte pas suffisamment d'information au modèle (l'information est mesurée au travers de la somme des carrés des erreurs). Il démontre que le meilleur des modèles ARMA non filtré est celui qui minimise la statistique :

$$AIC(p,q) = \log \tilde{\sigma}_\varepsilon^2 + \frac{2(p+q+k+1)}{T} \tag{10.85}$$

avec $\tilde{\sigma}_\varepsilon^2 = T^{-1} \sum_{t=1}^{t=T} \hat{\varepsilon}_t^2$ l'estimateur du maximum de vraisemblance de σ_ε^2 et k un paramètre qui prend la valeur 1 si la constante est incluse dans le modèle et 0 sinon.

Dans certains ouvrages, la fonction AIC est définie comme suit :

$$AIC(p,q) = T \log \tilde{\sigma}_\varepsilon^2 + 2(p+q+k+1) \tag{10.86}$$

```
# Critère AIC
def AIC(res):
  return np.log(res.sse/res.nobs)+2*len(res.params)/res.nobs
```

10.4.2.2 Critère d'information d'Akaike corrigé

De nombreuses modifications au critère AIC sont proposées dans la littérature, en particulier celui du AIC corrigé (AICC) (*cf.* Sugiura (1978), Hurvich and Tsai (1989)). Le AICC de Hurvich and Tsai (1989) peut être écrit comme suit :

$$AICC(p,q) = T \ln \tilde{\sigma}_\varepsilon^2 + T\frac{1+(p+q)/T}{1-(p+q+2)/T} \tag{10.87}$$

La correction se rapporte au biais d'estimation de l'information de Kullback and Leibler (1951) effectuée dans le AIC. Il est à noter que ce critère corrigé est fréquemment écrit sous la forme d'une somme du AIC et d'une pénalité non stochastique supplémentaire mais que Hurvich and Tsai (1989) écrivent le AIC sous une forme qui est une transformation de celle donnée ici, mesurée par T :

$$AICC(p,q) = AIC(p,q) + 2\frac{(p+q+k+1)(p+q+k+2)}{T-p-q-k-2} \tag{10.88}$$

Ce critère est recommandé quand le nombre de paramètres $p+q$ est grand par rapport au nombre d'observations T, i.e., si $T/(p+q) < 40$ (*cf.* Hurvich and Tsai (1995)).

```
# Critère AICC
def AICC(res):
  num = 2*len(res.params)*(len(res.params)+1)
  return AIC(res)+num/(res.nobs-len(res.params)-1)
```

10.4.2.3 Critère d'information bayésien

Ce critère est la version bayésienne du critère AIC. Le critère d'information bayésien (BIC, *Bayesian Informations Criterion*) est proposé par Akaike (1979) et pour un $ARMA(p,q)$, il s'écrit :

$$\begin{aligned} BIC(p,q) = T\log\tilde{\sigma}_\varepsilon^2 - (T-p-q-k-1)\log\left[1-\frac{p+q+k+1}{T}\right] \\ + (p+q+k+1)\log(T) + \log\left[\frac{\hat{\sigma}_X^2}{(p+q+k+1)(\tilde{\sigma}_\varepsilon^2-1)}\right] \end{aligned} \tag{10.89}$$

De manière générale, le critère BIC a des caractéristiques plus intéressantes que celles du critère AIC. Il est convergent et pénalise plus fortement les paramètres en surnombre que le critère AIC.

10.4.2.4 Critère d'estimation bayésien

Le critère d'estimation bayésien (BEC, *Bayesian Estimation Criterion*) a été proposé par Geweke and Meese (1981) et s'écrit :

$$BEC(p,q) = \tilde{\sigma}_\varepsilon^2 + (p+q+k+1)\tilde{\sigma}_L^2 \left(\frac{\log(T)}{T-L}\right) \tag{10.90}$$

où $\hat{\sigma}_L^2$ est l'estimateur du maximum de vraisemblance de la variance résiduelle de l'ordre du modèle le plus grand.

10.4.2.5 Critère d'information de Schwarz

Le critère de Schwarz (1978) (SIC, *Schwarz Informations Criterion*) est proche du critère AIC et, comme ce dernier, on cherche à le minimiser. Sa formule est :

$$SIC(p,q) = \log\tilde{\sigma}_\varepsilon^2 + (p+q+k+1)\frac{\log T}{T} \tag{10.91}$$

Dans certains ouvrages, on retrouve la formule suivante pour le critère de Schwarz :

$$SIC(p,q) = T\log\tilde{\sigma}_\varepsilon^2 + (p+q+k+1)\log T \tag{10.92}$$

```
# Critère SIC
def SIC(res):
  return np.log(res.sse/res.nobs)+len(res.params)*np.log(res.nobs)/res.nobs
```

Remarque 10.9

Dans certains ouvrages, la formule du critère d'information de Schwarz (1978) est celle que l'on utilise pour le critère BIC. C'est cette définition que l'on utilisera par la suite lorsque nous parlerons de critère BIC.

10.4.2.6 Critère d'information de Hannan - Quinn (HQIC)

E. J. Hannan and Quinn (1979) proposent un critère (HQIC) pour les processus ARMA qui peut être considér comme un intermédiaire entre le AIC et le BIC. En effet, le HQIC adoucit quelque peu la sévérité de la fonction de pénalité du BIC relativement à la croissance de la taille de l'échantillon tout en maintenant une forte convergence dans l'identification de l'ordre réel du modèle. Le coût de ces améliorations est l'obligation de choisir une valeur pour le paramètre α (supérieure à 1). Sa formule de calcul est :

$$HQ = \log\tilde{\sigma}_\varepsilon^2 + 2\alpha\frac{(p+q+k+1)\log(\log(T))}{T} \tag{10.93}$$

où p est l'ordre de la partie AR et q est l'ordre de la partie MA.

Malheureusement, il existe très peu d'information susceptible d'aider au choix de la valeur de α. Plusieurs études, parmi celle de E. J. Hannan and Quinn (1979), utilisent la valeur limite de $\alpha = 1$ pour conduire des expériences de simulation, en dépit du fait que la preuve de la convergence de $HQIC$ requière que α soit strictement supérieur à 1.

```
# Critère HQIC
def HQIC(res,alpha=1):
  num = 2*alpha*len(res.params)*np.log(np.log(res.nobs))
  return np.log(res.sse/res.nobs)+num/res.nobs
```

10.4.2.7 Erreur de prédiction finale (FPE)

Un autre principe de sélection est celui de la minimisation de l'erreur de prédiction finale (FPE, *Final Prediction Error*) d'un modèle de séries temporelles potentiellement incorrectement spécifié. La pénalité en résultant est similaire à celle du AIC quoique typiquement plus parcimonieuse :

$$FPE(p,q) = \tilde{\sigma}_\varepsilon^2 \left(1 + 2\frac{p+q+k+2}{T}\right) \tag{10.94}$$

```
# Erreur de prédiction finale
def FPE(res):
  return (res.sse/res.nobs)*(1+2*(len(res.params)+1)/res.nobs)
```

Remarque 10.10

Un critère similaire à l'erreur de prédiction finale, celui de Mallows (2000), dénoté par C_p, est applicable au contexte des régressions.

Parmi tous ces critères, les plus employés pour comparer les modèles entre eux sont les critères AIC, BIC, SIC et HQIC. Les autres critères sont essentiellement utilisés pour vérifier des performances prévisionnelles.

Exemple 10.26 *Evaluation des modèles à partir des critères d'information*

```
# Critère d'information
def InformationCriterion(res):
  return list([AIC(res),AICC(res),SIC(res),HQIC(res),FPE(res)])

# Application
infos_res = pd.DataFrame(index=["AIC","AICC","SC","HQIC","FPE"],columns=label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  infos_res[lab] = InformationCriterion(model)
```

Table 10.36 – Critères d'information

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
AIC	-0.3613996	-0.3602459	-0.3324682	-0.3872758	-0.4058648	-0.4191965
AICC	-0.2401874	-0.1572003	-0.1294226	0.0434934	-0.2028191	-0.1130740
SC	-0.3122669	-0.2947357	-0.2669580	-0.2890104	-0.3403546	-0.3373086
HQIC	-0.3415204	-0.3337404	-0.3059627	-0.3475175	-0.3793592	-0.3860645
FPE	0.7030954	0.7036098	0.7234285	0.6840864	0.6722331	0.6629915

Remarque 10.11 *Critère d'information sous ARIMA forecast de R*

Dans certains ouvrages de séries temporelles, notamment celle de R. J. Hyndman and Athanasopoulos (2014), les formules suivantes sont celles définies pour les critères d'information AIC, AICC et BIC :

— **Critère d'information de Akaike**

$$AIC(p,q) = -2\log(L) + 2(p+q+k+1) \tag{10.95}$$

où L est la vraisemblance du modèle.

```
# Critère AIC
def AIC2(res):
  return -2*res.llf+2*len(res.params)
```

— **Critère d'information de Akaike corrigé**

$$AICC(p,q) = AIC(p,q) + 2\frac{(p+q+k+1)(p+q+k+2)}{T-p-q-k-2} \tag{10.96}$$

```
# Critère AICC
def AICC2(res):
  num = 2*len(res.params)*(len(res.params)+1)
  return AIC2(res)+num/(res.nobs-len(res.params)-1)
```

— **Critère d'information bayésien**

$$BIC(p,q) = AIC(p,q) + [\log(T) - 2]\,(p+q+k+1) \tag{10.97}$$

```
# Critère BIC
def BIC2(res):
  return AIC2(res)+(np.log(res.nobs)-2)*len(res.params)
```

— **Critère d'information de Hannan - Quinn**

$$HQIC(p,q) = AIC(p,q) + 2\,[\log(\log(T)) - 1]\,(p+q+k+1) \tag{10.98}$$

```
# Critère HQIC
def HQIC2(res):
  return AIC2(res)+2*(np.log(np.log(res.nobs))-1)*len(res.params)

# Critère d'information
def InformationCriterion2(res):
  return list([AIC2(res),AICC2(res),BIC2(res),HQIC2(res)])

# Application
infos2_res = pd.DataFrame(index=["AIC","AICC","BIC","HQIC"],columns=label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  infos2_res[lab] = InformationCriterion2(model)
```

Remarque 10.12 *Critère d'information sous ARIMA Statsmodels*

Les formules des critères d'information disponibles dans l'ouvrage de R. J. Hyndman and Athanasopoulos (2014) sont identiques à celles qui ont été implémentées dans la fonction ARIMA de Statsmodels (les résultats obtenus confirment nos propos).

Table 10.37 – Critères d'information sous ARIMA forecast (R package)

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
AIC	500.2830	500.5236	506.1231	495.2194	491.5112	488.8925
AICC	500.4042	500.7266	506.3261	495.6502	491.7142	489.1986
BIC	510.2078	513.7567	519.3561	515.0690	504.7442	505.4338
HQIC	504.2986	505.8777	511.4772	503.2506	496.8653	495.5852

```
# Critère d'information
def InformationCriterion3(res):
  return list([res.aic,res.aicc,res.bic,res.hqic])

# Application
infos3_res = pd.DataFrame(index=["AIC","AICC","BIC","HQIC"],columns=label)
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  infos3_res[lab] = InformationCriterion3(model)
```

Table 10.38 – Critères d'information sous ARIMA Statsmodels

	AR(1)	AR(2)	MA(2)	MA(4)	ARMA(1,1)	ARMA(1,2)
AIC	500.2830	500.5236	506.1231	495.2194	491.5112	488.8925
AICC	500.4042	500.7266	506.3261	495.6502	491.7142	489.1986
BIC	510.2078	513.7567	519.3561	515.0690	504.7442	505.4338
HQIC	504.2986	505.8777	511.4772	503.2506	496.8653	495.5852

Le fonction res.info_criteria(criteria, method) retourne les valeurs des critères d'information basées soit sur la méthode standard, soit sur la méthode de Lütkepohl (2005). Par défaut, on a `method = "standard"`. Le tableau (10.39) donne les formules appliquées par les différentes méthodes pour le calcul des critères d'information :

Table 10.39 – Formule d'estimation des critères d'information

Critère	Standard	Lütkepohl
AIC(p,q)	$-2\log L\left(X_T/\widetilde{\psi}\right)+2d$	$\log(\mathcal{Q})+2\frac{d}{T}$
BIC(p,q)	$-2\log L\left(X_T/\widetilde{\psi}\right)+d\log(T)$	$\log(\mathcal{Q})+\frac{d\log(T)}{T}$
HQIC(p,q)	$-2\log L\left(X_T/\widetilde{\psi}\right)+2d\log(\log(T))$	$\log(\mathcal{Q})+2\frac{d\log(\log(T))}{T}$

Dans la formule standard, $\widetilde{\psi}$ est l'estimateur du maximum de vraisemblance des paramètres, T est le nombre d'observations, $d=p+q+k+1$ le nombre de paramètes du modèle avec k un paramètre binaire qui prend la valeur 1 si la constante est incluse dans le modèle et 0 sinon. Dans la formule de Lütkepohl (2005), $\mathcal{Q}$ représente la matrice de covariance état (dans le modèle à espace - état). Soulignons également que la formule de Lütkepohl (2005) ne s'applique pas à tous les modèles à espace - état et doit être utilisé avec précaution en déhors des modèles SARIMAX et VARMAX.

Exemple 10.27 *Critères d'information standards et de lütkepohl*

```
# Critères d'information
def InfoCriteria(res):
  aic1 = res.info_criteria(criteria = "aic", method="standard")
  bic1 = res.info_criteria(criteria = "bic", method="standard")
```

```
  hqic1 = res.info_criteria(criteria = "hqic", method="standard")
  aic2 = res.info_criteria(criteria = "aic", method="lutkepohl")
  bic2 = res.info_criteria(criteria = "bic", method="lutkepohl")
  hqic2 = res.info_criteria(criteria = "hqic", method="lutkepohl")
  return list([aic1,bic1,hqic1,aic2,bic2,hqic2])

# Application
infocriteria_res = pd.DataFrame(index=label,columns=range(6))
for lab,orde in zip(label,order):
  model = ARIMA(cpi2,order=orde,trend="c").fit()
  infocriteria_res.loc[lab,:] = InfoCriteria(model)
```

Table 10.40 – Critères d'information

	Standard			Lütkepohl		
	AIC	BIC	HQIC	AIC	BIC	HQIC
AR(1)	500.283	510.2078	504.2986	-0.362039	-0.3129063	-0.3421599
AR(2)	500.5236	513.7567	505.8777	-0.3609366	-0.2954263	-0.334431
MA(2)	506.1231	519.3561	511.4772	-0.3329687	-0.2674584	-0.3064631
MA(4)	495.2194	515.069	503.2506	-0.3886984	-0.290433	-0.34894
ARMA(1,1)	491.5112	504.7442	496.8653	-0.407004	-0.3414937	-0.3804984
ARMA(1,2)	488.8925	505.4338	495.5852	-0.4203887	-0.3385009	-0.3872567

On remarque, les résultats des tableaux 10.37 et 10.40 sont identiques pour les formules standards. Cependant, ceux du tableaux 10.36 diffèrent légèrement de ceux obtenus au tableau 10.40 à partir des formules de Lütkepohl (2005). Cette différence de résultat peut être imputée au calcul de la matrice de covariance état $\mathcal{Q}$ dans le modèle à espace - état, mais ne modifie en rien le choix du modèle.

Remarque 10.13

Le critère le plus utilisé est le critère AIC. En effet, Akaike a démontré que le meilleur des modèles ARMA non filtré est celui qui minimise la statistique du critère de AIC. Cependant, Hannan (1980) a montré que seules les estimations des ordres p et q déduites des critères BIC et $HQIC$ sont convergentes et conduisent à une sélection asymptotiquement correcte du modèle. On cherche à minimiser ces différents critères. Leur application permet de retenir un modèle parmi les divers processus $ARMA$ validés. Le modèle sélectionné sera alors utilisé pour la prévision.

10.5 Prévision

Supposons que l'on dispose d'un grand nombre d'observations consécutives de X jusquà la date T. Une fois le modèle choisi et ses paramètres estimés, il va être possible de faire de la prévision.

10.5.1 Prévision d'un AR(p)

Considérons un processus AR(p) défini par l'équation :

$$\Phi(B)X_t = \varepsilon_t \tag{10.99}$$

On veut prédire les valeurs futures $X_{t+1}, \dots, X_{t+h}$ de la série $(X_t)_t$ à partir des valeurs observées $\{X_1, X_2, \dots, X_t\}$.

Soit $\widehat{X}_t(h)$ le prédicteur de $(X_t)_{t\in\mathbb{Z}}$ à l'instant t et à l'horizon $h > 0$ (ou bien la valeur prévue au temps $t+h$ de (X_t) c'est-à-dire la valeur estimée de la valeur prévue pour l'instant futur $t+h$, calculée au moment présent t). Par définition, on a l'expression suivante :

$$\widehat{X}_t(h) = \mathbb{E}\left(X_{t+h}|I_t\right) \tag{10.100}$$

où $I_t = \sigma\left(X_1, X_2, \dots, X_t\right)$ est l'ensemble d'information disponible à la date t. Ce prédicteur représente la meilleure prévision de la série X_t conditionnellement à l'ensemble d'information disponible.

A $t+1$, le processus X_t sous forme extensive s'écrit :

$$X_{t+1} = \phi_1 X_t + \phi_2 X_{t-1} + \cdots + \phi_p X_{t-p+1} + \varepsilon_{t+1} \tag{10.101}$$

Ainsi, pour $h = 1$, on trouve :

$$\begin{aligned}
\widehat{X}_t(1) = \mathbb{E}(X_{t+1}|I_t) &= \mathbb{E}\left(\phi_1 X_t + \phi_2 X_{t-1} + \cdots + \phi_p X_{t-p+1} + \varepsilon_{t+1}|I_t\right) \\
&= \phi_1 X_t + \phi_2 X_{t-1} + \cdots + \phi_p X_{t-p+1} + \mathbb{E}\left(\varepsilon_{t+1}|I_t\right) \\
&= \phi_1 X_t + \phi_2 X_{t-1} + \cdots + \phi_p X_{t-p+1}
\end{aligned}$$

Pour $h = 2$, on a :

$$\begin{aligned}
\widehat{X}_t(2) = \mathbb{E}(X_{t+2}|I_t) &= \mathbb{E}\left(\phi_1 X_{t+1} + \phi_2 X_t + \cdots + \phi_p X_{t-p+2} + \varepsilon_{t+2}|I_t\right) \\
&= \phi_1 \mathbb{E}\left(X_{t+1}|I_t\right) + \phi_2 X_t + \cdots + \phi_p X_{t-p+2} \\
&= \phi_1 \hat{X}_t(1) + \phi_2 X_t + \cdots + \phi_p X_{t-p+2}
\end{aligned}$$

Ainsi, à la période h, on a :

$$\widehat{X}_t(h) = \sum_{j=1}^{p} \hat{X}_t(h-j) \tag{10.102}$$

or $\hat{X}_t(h-j) = X_{t-j+h}$ si $h \leq j$.

D'où l'écriture suivante :

$$\widehat{X}_t(h) = \sum_{j=1}^{h-1} \phi_j \hat{X}_t(h-j) + \sum_{j=h}^{p} \phi_j X_{t-j+h} \tag{10.103}$$

Exemple 10.28

On considère AR(1) stationnaire défini par :

$$X_t = \phi_1 X_{t-1} + \varepsilon_t \tag{10.104}$$

où $\varepsilon_t \sim BB(0, \sigma_\varepsilon^2)$.
La prévision à l'ordre $h = 1$ est donnée par :

$$\widehat{X}_t(1) = \mathbb{E}(X_{t+1}|I_t) = \mathbb{E}\left(\phi_1 X_t + \varepsilon_{t+1}|I_t\right) = \phi_1 X_t \tag{10.105}$$

La prévision à l'ordre $h = 2$ est donnée par :

$$\widehat{X}_t(2) = \mathbb{E}(X_{t+2}|I_t) = \mathbb{E}\left(\phi_1 X_{t+1} + \varepsilon_{t+2}\right) = \phi_1 \mathbb{E}\left(X_{t+1}|I_t\right) = \phi_1 \widehat{X}_t(1) = \phi_1^2 X_t \tag{10.106}$$

Par identification, pour un processus AR(1), la prévision à l'horizon h $(h > 0)$ est donnée par :

$$\widehat{X}_t(h) = \phi_1^h X_t \tag{10.107}$$

Exemple 10.29

Considérons le processus AR(2) stationnaire défini par :

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \varepsilon_t \tag{10.108}$$

où $\varepsilon_t \sim BB(0, \sigma_\varepsilon^2)$.
La prévision à l'ordre $h = 1$ est donnée par :

$$\widehat{X}_t(1) = \mathbb{E}(X_{t+1}|I_t) = \mathbb{E}\left(\phi_1 X_t + \phi_t X_{t-1} + \varepsilon_{t+1}|I_t\right) = \phi_1 X_t + \phi_2 X_{t-1} \tag{10.109}$$

La prévision à l'ordre $h = 2$ est donnée par :

$$\widehat{X}_t(2) = \mathbb{E}(X_{t+2}|I_t) = \mathbb{E}\left(\phi_1 X_{t+1} + \phi_2 X_t + \varepsilon_{t+2}|I_t\right) = \phi_1 \widehat{X}_t(1) + \phi_2 X_t \tag{10.110}$$

La prévision à l'ordre $h = 3$ est donnée par :

$$\widehat{X}_t(3) = \mathbb{E}(X_{t+3}|I_t) = \mathbb{E}\left(\phi_1 X_{t+2} + \phi_2 X_{t+1} + \varepsilon_{t+3}|I_t\right) = \phi_1 \widehat{X}_t(2) + \phi_2 \widehat{X}_t(1) \tag{10.111}$$

et ainsi de suite.

10.5.2 Prévision d'un MA(q)

Considérons un processus MA(q) défini par l'équation :

$$X_t = \Theta(B)\varepsilon_t = \varepsilon_t - \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} \tag{10.112}$$

La prévision optimale à l'ordre h $(h > 0)$ est donnée par :

$$\widehat{X}_t(h) = \mathbb{E}(X_{t+h}|I_t) \tag{10.113}$$

on a :

$$X_{t+h} = \varepsilon_{t+h} - \sum_{j=1}^{q} \theta_j \varepsilon_{t-j+h} \tag{10.114}$$

Si $h > q$, alors $\hat{X}_t(h) = 0$, sinon on a la prédiction suivante :

$$\hat{X}_t(h) = -\sum_{j=h}^{q} \theta_j \varepsilon_{t-j+h} \tag{10.115}$$

Cette forme est exacte mais n'est pas utilisable en pratique car les ε_{t-k} ne sont pas observés pour $k \geq 0$. Mais on peut les calculer à partir des observations en utilisant la forme $AR(\infty)$:

$$\varepsilon_t = \Theta^{-1}(B) X_t \tag{10.116}$$

on a donc :

$$X_t = \varepsilon_t - \sum_{k=1}^{\infty} \alpha_k X_{t-k} \tag{10.117}$$

Ainsi, à $h > 0$, on a :

$$X_{t+h} = \varepsilon_{t+h} - \sum_{k=1}^{\infty} \alpha_k X_{t-k+h} \tag{10.118}$$

Pour les prévisions optimales, on a :

$$\begin{aligned}
\widehat{X}_t(1) &= -\sum_{k=1}^{\infty} \alpha_k X_{t-k+1} \\
&\vdots \\
\widehat{X}_t(h) &= -\sum_{k=1}^{k=h-1} \alpha_k \widehat{X}_t(h-k) - \sum_{k=h}^{\infty} \alpha_k X_{t-k+h}
\end{aligned}$$

En pratique, on n'observe pas les X_t pour $t < 0$. On n'a qu'une prévision approchée en tronquant :

$$\widehat{X}_t(h) = -\sum_{k=1}^{h-1} \alpha_k \widehat{X}_t(h-k) - \sum_{k=h}^{k=t+h} \alpha_k X_{t-k+h} \tag{10.119}$$

En fait on néglige le terme

$$\left\| \sum_{k=t+h+1}^{\infty} \alpha_k X_{t-k+h} \right\|_2 \leq \underbrace{\left(\sum_{k=t+h+1}^{\infty} |\alpha_k| \right)}_{\begin{array}{c} h \longrightarrow \infty \\ \Downarrow \\ \longrightarrow 0 \end{array}} ||X_i||_2 \tag{10.120}$$

10.5.3 Prévision d'un ARMA(p,q)

Pour estimer les prévisions optimales $\widehat{X}_t(h)$ d'un modèle $ARMA(p,q)$, il faut disposer :

- des coefficients ψ_j qui lient X à son bruit blanc d'innovation ε : $X_t = \sum_{j\geq 0} \psi_j \varepsilon_{t-j}$
- des coefficients ω_j qui lient ε à X : $\varepsilon_t = \sum_{j\geq 0} \omega_j X_{t-j}$
- des valeurs numériques observées des $X_t, t \leq T$
- des valeurs numériques obervées observées des $\varepsilon_t, t \leq T$

Considérons un processus ARMA(p,q) défini par l'équation :

$$\Phi(B)X_t = \Theta(B)\varepsilon_t \tag{10.121}$$

On se trouve à l'instant T et on veut prédire les valeurs futures $X_{T+1}, \dots, X_{T+h}$ de la série $(X_t)_{t\leq T}$ à partir des valeurs observées $\{X_1, X_2, \dots, X_T, \varepsilon_1, \varepsilon_2, \dots, \varepsilon_T\}$.

Soit $\widehat{X}_T(h)$ le prédicteur de $(X_t)_{t\in\mathbb{Z}}$ à l'instant T et à l'horizon $h > 0$ (ou bien la valeur prévue au temps $T+h$ de (X_T) c'est-à-dire la valeur estimée de la valeur prévue pour l'instant futur $T+h$, calculée au moment présent T). Par définition, on a l'expression suivante :

$$\widehat{X}_T(h) = \mathbb{E}\left(X_{T+h}|I_T\right) \tag{10.122}$$

où $I_T = \sigma\left(X_1, X_2, \dots, X_T, \varepsilon_1, \varepsilon_2, \dots, \varepsilon_T\right)$ est l'ensemble d'information disponible à la date T. Ce prédicteur représente la meilleure prévision de la série X_t conditionnellement à l'ensemble d'information disponible.

10.5.3.1 Prévison déduite de la forme autorégressive

Le théorème suivant donne les prévisions optimales obtenues grâce à la représentation $AR(\infty)$ du processus $ARMA(p,q)$ lorsque celle - ci existe :

Theorème 10.1 *1. Les prévisions optimales s'expriment comme des combinaisons linéaires*

$$\widehat{X}_T(h) = \sum_{j\geq 0} \omega_j(h) X_{T-j} \tag{10.123}$$

avec pour chaque h

$$\sum_{j\geq 0} \omega_j(h) z^j = \left[\frac{1}{z^h}\frac{\Phi(z)}{\Theta(z)}\right]_{|} \frac{\Theta(z)}{\Phi(z)} \tag{10.124}$$

où $[.]_{|}$ signifie que l'on ne retient que les termes en z^k pour $k \geq 0$.

2. Les $\widehat{X}_T(h)$ s'obtiennent par récurrence selon :

$$\widehat{X}_T(h) = -\sum_{j\geq 0} \omega_j \widehat{X}_T(h-j) \tag{10.125}$$

avec les conditions initiales pour $t \leq T$: $\widehat{X}_T(t-T) = X_t$.

Considérons la forme $AR(\infty)$ du processus ARMA, on a :

$$\varepsilon_t = \underbrace{\Theta^{-1}(B)\Phi(B)}_{\Omega(B)} X_t \tag{10.126}$$

Par conséquent, on peut écrire :

$$X_t = -\sum_{j=1}^{+\infty} \omega_j X_{t-j} + \varepsilon_t \tag{10.127}$$

On a la prévision optimale suivante :

$$\widehat{X}_T(h) = -\sum_{j=1}^{j=h-1} \omega_j \widehat{X}_T(h-j) - \sum_{j=h}^{T+h} \omega_j X_{T-j+h} \tag{10.128}$$

Le deuxième point du théorème permet de calculer récursivement les prévisions à partir des prévisions précédentes. Remarquons d'abord que, pour $t \leq T$, $\widehat{X}_T(t-T) = X_t$. Ainsi, pour la prévisionà l'horizon 1, on a la formule suivante :

$$\widehat{X}_T(1) = -\sum_{j\geq 1}^{+\infty} \omega_j X_{T+1-j} \tag{10.129}$$

qui ne fait intervenir que des valeurs observées de la série temporelle.

Pour la prévision à l'horizon 2, on a une formule basée sur les observations et sur la prévision donnée ci - dessus :

$$\widehat{X}_T(2) = -\omega_1 \widehat{X}_T(1) - \sum_{j\geq 2}^{+\infty} \omega_j X_{T+1-j} \tag{10.130}$$

et ainsi de suite pour tout $h \in \mathbb{N}^*$. Cependant, ces prévisions font intervenir des valeurs non observées, à savoir X_t pour $t \leq 0$. Il faut alors effectuer une approximation en tronquant la série.On obtient alors la prévision suivante :

$$\widehat{X}_T(h) = -\sum_{j=1}^{j=T+h-1} \omega_j X_{T+h-h} \tag{10.131}$$

avec $\widehat{X}_T(t-T) = X_t$ pour $t \leq T$.

10.5.3.2 Prévision déduite de la forme moyenne mobile

Le théorème suivant donne les prévisions optimales obtenues grâce à la représentation $MA(\infty)$ du processus $ARMA(p,q)$ lorsque celle - ci existe :

Theorème 10.2 *1. Les $\widehat{X}_T(h)$ s'expriment comme des combinaisons linéaires des valeurs passées du bruit blanc d'innovation*

$$\widehat{X}_T(h) = \sum_{j\geq h} \phi_j \varepsilon_{T+h-j} \tag{10.132}$$

2. Enfin, la variance de l'erreur de prédiction est donnée par :

$$\mathbb{E}\left[(X_{T+h} - \widehat{X}_T(h))^2\right] = \sigma_\varepsilon^2 \sum_{j=0}^{j=h-1} \psi_j^2 \tag{10.133}$$

Les erreurs de prédiction $X_{T+h} - \widehat{X}_T(h)$ ne sont donc pas non - corrélées. En effet, la covariance entre l'erreur de prévision à l'horizon h et l'erreur de prévision à l'horizon k avec $h \geq k$ vaut :

$$\mathbb{E}\left[\left(X_{T+h} - \widehat{X}_T(h)\right)\left(X_{T+k} - \widehat{X}_T(k)\right)\right] = \sigma_\varepsilon^2 \sum_{j=0}^{k-1} \psi_j \psi_{j+h-k} \tag{10.134}$$

Exemple 10.30 *Prévision d'un processus ARMA(1,1)*

Considérons un processus ARMA(1,1) défini par l'équation suivante :

$$X_t = \phi_1 X_{t-1} + \varepsilon_t - \theta_1 \varepsilon_{t-1} \tag{10.135}$$

avec la condition de stationnarité et d'inversibilité suivante : $0 < \phi_1, \theta_1 < 1$.

Pour calculer les prévisions pour l'horizon $h > 1$, on considère la forme MA(∞) du processus ARMA, on a :

$$X_t = \underbrace{\Phi(B)^{-1}\Theta(B)}_{\Psi(B)} \varepsilon_t \tag{10.136}$$

Soit,

$$X_t = \varepsilon_t + \psi_1 \varepsilon_{t-1} + \psi_2 \varepsilon_{t-2} + \psi_3 \varepsilon_{t-3} + \cdots \tag{10.137}$$

Donc la prévision théorique peut s'écrire de la forme suivante :

$$X_{T+h} = \varepsilon_{T+h} + \psi_1 \varepsilon_{T+h-1} + \psi_2 \varepsilon_{T+h-2} + \psi_3 \varepsilon_{T+h-3} + \cdots \tag{10.138}$$

On a alors

$$\widehat{X}_T(h) = \mathbb{E}\left(X_{T+h} | I_T\right) = \mathbb{E}\left(\varepsilon_{T+h} + \psi_1 \varepsilon_{T+h-1} + \psi_2 \varepsilon_{T+h-2} + \psi_3 \varepsilon_{T+h-3} + \cdots | I_T\right) \tag{10.139}$$

Pour $h = 1$ et $h = 2$, on obtient respectivement :

$$\begin{cases} \widehat{X}_T(1) = \psi_1 \varepsilon_t + \psi_2 \varepsilon_{T-1} + \cdots \\ \widehat{X}_T(2) = \psi_2 \varepsilon_t + \psi_3 \varepsilon_{T-1} + \cdots \end{cases} \tag{10.140}$$

Ainsi, on en déduit

$$\widehat{X}_T(h) = \psi_h \varepsilon_T + \psi_{h+1} \varepsilon_{T-1} + \psi_{h+2} \varepsilon_{T-2} + \cdots = \sum_{i \geq 0} \psi_{h+i} \varepsilon_{T-i} \tag{10.141}$$

On définit ainsi, l'erreur de prévision ε_{T+h} par,

$$\varepsilon_{T+h} = X_{T+h} - \widehat{X}_T(h) = \sum_{i=0}^{i=h-1} \psi_i \varepsilon_{T+h-i} \quad \text{avec} \quad \psi_0 = 1 \tag{10.142}$$

En supposant que ε_t est un bruit blanc gaussien de variance σ_ε^2, ε_{T+h} obéit donc à une loi $\mathcal{N}(0, \sigma_h^2)$ où σ_h^2 la variance de l'erreur de prévision est donnée par :

$$\sigma_h^2 = \mathrm{V}\left(\varepsilon_{T+h}\right) = \mathrm{V}\left(\sum_{i=0}^{i=h-1} \psi_i \varepsilon_{T+h-i}\right) = \sum_{i=0}^{i=h-1} \psi_i^2 \mathbb{E}\left(\varepsilon_{T+h-i}\right)^2 = \sigma_\varepsilon^2 \sum_{i=0}^{i=h-1} \psi_i^2 \tag{10.143}$$

La précision des estimations est donc d'autant plus forte que la variance du bruit blanc est faible. Et cette précision diminue quand on veut prévoir plus loin dans le futur car la variance de l'erreur de prévision, dépendant aussi de l'horizon h, devient élevée.

Proposition 10.1

Nous avons les deux propositions suivantes :

1. L'erreur de prévision à l'horizon 1 pour un processus ARMA est lebruit d'innovation ε_{T+1}.
2. La variance de l'erreur de prévision à l'horizon h pour un processus ARMA croît depuis la variance du bruit d'innovation (valeur prise pour $h-1$) jusqu'à la variance du processus lui - même.

10.5.3.3 Intervalle de confiance pour la prévision

On peut enfin construire un intervalle de confiance autour des prévisions réalisées. En effet, par hypothèse que ε_t est un bruit blanc gaussien, la distribution conditionnelle de la valeur à prévoir est donc une variable aléatoire d'espérance $\widehat{X}_T(h)$ et d'écart-type $\sigma_{\hat{\varepsilon}}$. Ainsi, la valeur observée :

$$\frac{X_{T+h} - \widehat{X}_T(h)}{\sigma_\varepsilon \sqrt{\sum_{i=0}^{i=h-1} \psi_i^2}} \sim \mathcal{N}(0,1) \tag{10.144}$$

L'intervalle de confiance de la prévision, au seuil $\alpha\%$, est donc donné par :

$$X_{T+h} \in \left[\widehat{X}_T(h) \pm z_{1-\alpha/2} \times \hat{\sigma}_\varepsilon \sqrt{\sum_{i=0}^{i=h-1} \psi_i^2}\right] \tag{10.145}$$

où $\hat{\sigma}_\varepsilon$ est l'estimation de l'écart-type σ_ε du bruit blanc et $q_{1-\alpha/2}$ le quantile d'ordre $1-\alpha/2$ de la loi normale centrée réduite ($\alpha = 5\% \Rightarrow z_{1-\alpha/2} = 1.96$).

10.5.3.4 Mise à jour

Le problème est le suivant : on a,en se basant sur les observations antérieures à T, prévu $X_{T+1}, X_{T+2}, \ldots, X_{T+h}$. On observe maintenant X_{T+1}. Non seulement on n'a plus à le prévoir, mais encore on connaît maintenant la valeur de l'erreur de prévision $X_{T+1} - \widehat{X}_T(1)$. De plus, on va pouvoir modifier les prévisions de $X_{T+2}, X_{T+3}, \ldots, X_{T+h}$. Pour cela, on écrit le développement en moyenne mobile

$$X_{T+h} = \varepsilon_{T+h} + \psi_1 \varepsilon_{T+h-1} + \cdots + \psi_{h-1}\varepsilon_{T+1} + \psi_h \varepsilon_T + \cdots \tag{10.146}$$

Le prédicteur est donné par :

$$\widehat{X}_{T+1}(h-1) = \psi_{h-1}\varepsilon_{T+1} + \psi_h \varepsilon_T + \cdots + \cdots = \psi_{h-1}\varepsilon_{T+1} + \widehat{X}_T(h) \tag{10.147}$$

Donc la prévision de X_{T+h} s'obtient en ajoutant $\psi_{h-1}\varepsilon_{T+1}$ à la prévision précédemment obtenue.Or on connaît la valeur de ce terme complémentaire. En effet, ε_{T+1} est, l'erreur de prévision à l'horizon $h = 1$, c'est - à - dire $X_{T+1} - \widehat{X}_T(1)$. On a donc le resultat suivant :

Proposition 10.2

Lorsque l'observation X_{T+1} vient s'jouter aux précédentes, la mise à jour des prévisions se fait de la façon suivante :

$$\widehat{X}_{T+1}(h-1) = \psi_{h-1}\left(X_{T+1} - \widehat{X}_T(1)\right) + \widehat{X}_T(1) \tag{10.148}$$

Autrement dit, on ajoute au prédicteur précédend de X_{T+h} une correction proportionnelle à l'erreur que l'on avait faite en prédisant, avant de l'avoir observée, la donnée que l'on vient de recueillir.

10.5.4 L'approche de Box et Jenkins

Les différents paramètres d'un processus ARMA sont :

- les coefficients du polynôme Φ ;
- les coefficients du polynôme Θ ;
- la variance σ_ε^2 du bruit blanc

En fait, implicitement, il y a avant tout l'ordre (p, q) du processus ARMA à déterminer.

Nous abordons à présent la méthode de Box - Jenkins qui est une des méthodes de traitement des processus ARMA la plus couramment utilisée en raison de sa simplicité, de l'économie de temps qu'elle permet et de la fiabilité de ses résultats. Il s'agit de décrire ici une démarche générale pour l'ajustement de données à l'aide d'un modèle ARMA.

On suppose que l'on dispose de T observations $X_1, \ldots, X_T$ d'une série chronologique stationnaire. Les étapes de cette démarche sont les suivantes :

1. **Estimation de la tendance** :

   on commence par estimer la moyenne

   $$\overline{X} = \frac{1}{T}\sum_{t=1}^{t=T} X_t \tag{10.149}$$

   puis on centre la série d'observations le cas échéant.
2. **Estimation des fonctions d'autocorrélation simple et partielle**

   On estime la fonction d'autocorrélation simple $\rho(h)$ par $\hat{\rho}(h)$ et d'autocorrélation partielle $r(h)$ par $\hat{r}(h)$. Il est également intéressant d'etudier les autocorrélations inverses. En effet, si $(X_t)_t$ est un processus $ARMA(p, q)$, solution de l'équation $\Phi(B)X_t = \Theta(B)\varepsilon_t$ et si Θ est inversible, alors l'équation $\Phi(B)X_t = \Theta(B)\varepsilon_t$ défini un processus dual $ARMA(q, p)$

dont on peut étudier la fonction d'autocorrélation, appelée **fonction d'autocorrélation inverse**.

3. **Vérification des hypothèses**

   Si $\hat{\rho}(h)$ et $\hat{r}(h)$ ne tendent pas vers 0 quand h croît, alors on rejette l'hypothèse d'une modélisation par un processus ARMA. Dans ce cas, on peut essayer de transformer les données à l'aide d'outils classiques : différenciation, différenciation saisonnière, transformation linéaire, exponentielle, transformation de Box - Cox...en effectuant un minimum de transformation.

4. **Détermination de p_{max} et q_{max}**

   Si l'étape précédente s'est bien passée, on cherche alors à modéliser les données par un $MA(q_{max})$ puis un $AR(p_{max})$ avec p_{max} et q_{max} maximum en s'aidant des autocorrélations simples et des autocorrélations partielles.

5. **Estimation des paramètres**

   Pour chaque processus ARMA d'ordre (p, q) choisi, on estime les différents paramètres du modèle : les coefficients $\phi_i, i = 1, \dots, p$ (partie AR), les coefficients $\theta_j, j = 1, \dots, q$ (partie MA) et la variance σ_ε^2 du processus d'innovation. On peut également, si elle existe, estimer la constante du modèle.

6. **Amélioration de l'estimation des paramètres**

   Si on peut émettre l'hypothèse de bruit blanc gaussien (pour cela, on peut utiliser un test adéquat), on est alors dans le cas d'un processus gaussien et les estimateurs peuvent être affinés par la méthode du maximum de vraisemblance (algorithme de Box et Jenkins).

7. **Sélection de modèles**

   On compare les différents modèles. Pour cela, on dispose de plusieurs critères tels que AIC, BIC, SIC, HQIC et FPE.

11

Processus autorégressifs avec variables exogènes

Sommaire

On souhaite expliquer X_t par ses valeurs passées, mais également par l'influence d'une (ou plusieurs) variable (Z_t) exogène observée en parallèle. Par exemple, si l'on est en train d'établir un modèle pour la consommation électrique quotidienne des ménages, il peut être intéressant d'intégrer au modèle la donnée de la température extérieure.

11.1 Présentation du modèle ARX

11.1.1 Formulation générale

Dans ce type de modèles de séries temporelles, la variable endogène X_t dépend de d variables exogènes $Z_{1,t}, \cdots, Z_{d,t}$ à la période t et des valeurs qu'elle prend pendant les périodes précédentes $X_{t-1}, \cdots, X_{t-p}$ avec p le décalage maximal possible. Soit la formulation suivante :

$$X_t = c + \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \beta_1 Z_{1,t} + \cdots + \beta_d Z_{d,t} + \varepsilon_t \tag{11.1}$$

avec $\varepsilon_t \sim i.i.d(0, \sigma_\varepsilon^2)$.

Cette formulation peut se réécrire comme suit :

$$\Phi(B) X_t = c + Z\beta + \varepsilon_t \tag{11.2}$$

avec $\Phi(B) = (1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p)$ et $\beta = (\beta_1, \ldots, \beta_d)'$. On dit que X_t suit un processus autorégressif d'ordre p avec variables exogènes que l'on note ARX(p).

Dans ce modèle, l'hypothèse d'indépendance entre les variables explicatives et l'erreur n'est plus satisfaite car les variables $X_{t-1}, \ldots, X_{t-p}$ qui dépendent de $\varepsilon_{t-1}, \ldots, \varepsilon_{t-p}$ sont aléatoires puisque X_{t+1} est fonction de X_t qui dépend de ε_t, soit : $\mathbb{E}\left(X_{t+1}\varepsilon_t\right) \neq 0$.

Remarquons que si les variables exogènes $Z_{k,t}(k = 1, \ldots, d)$ et les erreurs ε_t sont fixées, alors les variables endogènes sont solutions de l'équation de récurrence :

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + Y_t \tag{11.3}$$

avec $Y_t = c + Z\beta + \varepsilon_t$. La solution générale de cette équation de récurrence est explosive si une des racines de son équation caractéristique [1] à un module supérieur à 1. C'est pourquoi il convient d'ajouter l'hypothèse de stabilité du processus.

Proposition 11.1 *Condition de stabilité d'un processus ARX*

Un processus $ARX(p)$ est stable si les racines de son polynôme caractéristique $\Phi(z) = 1 - \phi_1 z - \phi_2 z^2 - \cdots - \phi_p z^p$ sont de module supérieur à 1.

Un cas particulier de processus autorégressif avec variables exogènes le plus couramment utilisé est le processus $ARX(1)$ de la forme :

$$X_t = c + \phi_1 X_{t-1} + \beta_1 Z_{1,t} + \cdots + \beta_d Z_{d,t} + \varepsilon_t \tag{11.4}$$

Ce processus $ARX(1)$ est stable si et seulement si $|\phi_1| < 1$.

11.1.2 Autocorrélation croisée

Pour connaître l'influence d'une série sur une autre, on utilise la fonction de corrélation croisée (CCF, pour cross-correlation function).

On appelle covariance croisée de $(X_t)_t$ avec $(Z_t)_t$ la fonction

$$\Gamma_{X,Z}(t,s) \quad \to \quad \mathrm{Cov}(X_t, Z_s) = \mathbb{E}\left[(X_t - \mathbb{E}[X_t])(Z_s - \mathbb{E}[Z_s])\right]$$

Proposition 11.2 *Mutuellement stationnaires*

On dit que $(X_t)_t$ et $(Z_t)_t$ sont mutuellement stationnaires si :

- $(X_t)_t$ et $(Z_t)_t$ sont chacune stationnaires.
- $\Gamma_{X,Z}(t+h,t)$ est une fonction de h uniquement (donc ne dépend pas de t).

  Dans ce cas, on note $\gamma_{X,Z}(h) = \Gamma_{X,Z}(t+h,t) = \Gamma_{X,Z}(h,0)$.

```
# Fonction d'autocovariance croisée
from statsmodels.tsa.stattools import ccovf
print(ccovf(x,y))
```

Remarque 11.1

La fonction $\gamma_{X,Z}$ n'est pas paire. En effet, on n'a pas nécessairement $\gamma_{X,Z}(h) \neq \gamma_{X,Z}(h)$, par contre, $\gamma_{X,Z}(h) = \gamma_{Z,X}(-h)$.

1. L'équation caractéristique s'écrit : $r^p - \phi_1 r^{p-1} - \phi_2 r^{p-2} - \cdots - \phi_p = 0$.

11.1.2.1 Définition

Soient $(X_t)_t$ et $(Z_t)_t$ mutuellement stationnaires, de fonctions d'autocovariance respectives γ_X et γ_Z. Leur *fonction d'autocorrélation croisée (CCF)* $\rho_{X,Z}$ est définie pour tout $h \in \mathbb{Z}$ par :

$$\rho_{X,Z}(h) = \frac{\gamma_{X,Z}(h)}{\sqrt{\gamma_X(0)\gamma_Z(0)}} = \frac{\mathrm{Cov}(X_h, Z_0)}{\sqrt{\mathrm{Var}(X_0)\mathrm{Var}(Z_0)}} \tag{11.5}$$

```
# Fonction d'autocorrélation croisée
from statsmodels.tsa.stattools import ccf
print(ccf(x,y))
```

Exemple 11.1

Soit $(\varepsilon_t)_t$ un bruit blanc de variance σ^2_ε, soient $(X_t)_t$ et $(Z_t)_t$ deux processus définis respectivement pour tout $t \in \mathbb{Z}$ par

$$\begin{cases} X_t &= \varepsilon_t + \varepsilon_{t-1} \\ Z_t &= \varepsilon_t - \varepsilon_{t-1} \end{cases} \tag{11.6}$$

Montrons que $(X_t)_t$ et $(Z_t)_t$ sont mutuellement stationnaires.

Solution

Tout d'abord, ce sont des processus MA(1) dont ils sont stationnaires et on montre facilement que $\gamma_X(0) = \gamma_Z(0) = 2\sigma^2$. Il reste donc à regarder leur covariance croisée. On a :

$$\Gamma_{X,Z}(t+h,t) = \mathrm{Cov}(X_{t+h}, Z_t) = \mathrm{Cov}(\varepsilon_{t+h} + \varepsilon_{t+h-1}, \varepsilon_t - \varepsilon_{t-1}) \tag{11.7}$$

Ainsi, on déduit de la corrélation du bruit blanc que

- Pour $h = 0$, $\Gamma_{X,Z}(t,t) = 0$
- Pour $h = -1$, $\Gamma_{X,Z}(t-1,t) = -\sigma^2$
- Pour $h = 1$, $\Gamma_{X,Z}(t+1,t) = \sigma^2$
- et pour $h \geq 2$, $\Gamma_{X,Z}(t+h,t) = 0$

Donc, celle-ci ne dépend jamais de la valeur de t. Les processus sont dont mutuellement stationnaires et

$$\gamma_{X,Z} = \begin{cases} 0 & \text{si} \quad h = 0 \quad \text{ou} \quad |h| \geq 2 \\ -\sigma^2 & \text{si} \quad h = -1 \\ \sigma^2 & \text{si} \quad h = 1 \end{cases} \quad \text{et} \quad \rho_{X,Z} = \begin{cases} 0 & \text{si } h = 0 \quad \text{ou} \quad |h| \geq 2 \\ -1/2 & \text{si} \quad h = -1 \\ 1/2 & \text{si} \quad h = 1 \end{cases} \tag{11.8}$$

11.2 Test d'autocorrélation et méthodes d'estimation

La méthode d'estimation adéquate dépend d'une éventuelle dépendance des erreurs ; or, dans le cas d'un processus ARX, le test de Durbin et Watson a une puissance réduite et est biaisé. C'est pourquoi il convient d'utiliser une autre statistique, celle du « h » de Durbin.

11.2.1 Test d'autocorrélation des erreurs

Dans le cas d'un modèle autorégressif d'ordre 1, le test classique de Durbin et Watson sous-estime le risque d'autocorrélation, un nouveau test d'autocorrélation des erreurs doit alors être utilisé. La statistique utilisée est la suivante :

$$h = \hat{\rho}\sqrt{\frac{T}{1 - T\hat{\sigma}^2_{\hat{\phi}_1}}} \tag{11.9}$$

avec $\hat{\rho} = 1 - dw/2$ (dw est la statistique de Durbin et Watson calculée sur le modèle (11.4)) ; T est le nombre d'observations et $\hat{\sigma}^2_{\hat{\phi}_1}$ la variance estimée du coefficient ϕ_1 à partir du modèle (11.4).

Cette statistique « h » est distribuée de la manière asymptotique comme une variable normale centrée réduite. Ainsi, il y a équivalence entre les deux tests d'hypothèses suivants :

$$\begin{cases} H_0 : & \rho = 0 \\ H_1 : & \rho \neq 0 \end{cases} \quad \text{equivaut à} \quad \begin{cases} H_0 : & h = 0 \\ H_1 : & h \neq 0 \end{cases} \tag{11.10}$$

Nous acceptons l'hypothèse H_0 d'indépendance des erreurs si $|h| < z_{\alpha/2}$ où $z_{\alpha/2}$ est la valeur issue de la loi normale pour un test bilatéral au seuil de α.

Remarque 11.2

Si $T\hat{\sigma}^2_{\hat{\phi}_1} \geq 1$, la statistique « h » ne peut pas être calculée ; dans ce cas, nous pouvons utiliser la statistique de Durbin et Watson en incluant la zone de doute dans la zone d'autocorrélation des erreurs.

11.2.2 Estimation en cas d'absence d'autocorrélation des erreurs

Dans le cas d'absence d'autocorrélation des erreurs, les estimateurs des moindres carrés ordinaires appliqués au modèle (11.4) convergent asymptotiquement vers les valeurs vraies des paramètres et ont une variance minimale parmi les estimateurs convergents.

Cependant, pour les petits échantillons, lors de l'estimation d'un modèle autorégressif d'ordre p, les résultats asymptotiques sont alors très approximatifs, car le nombre de périodes d'estimation est de $T - p$. De plus, les problèmes de colinéarité entre les variables explicatives décalées interdisent pratiquement d'utiliser les moindres carrés ordinaires.

En résumé, pour l'estimation des modèles autorégressifs à erreurs indépendants, l'application de la méthode classique des moindres carrés ordinaires est licite si le nombre d'observations est suffisant (souvent dans la pratique $T > 15$).

11.2.3 Estimation en cas d'autocorrélation des erreurs

La modèle (11.1) s'écrit alors :

$$X_t = c + \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \beta_1 Z_{1,t} + \cdots + \beta_d Z_{d,t} + \varepsilon_t \tag{11.11}$$

avec $\varepsilon_t = \rho\varepsilon_{t-1} + \mu_t$ et $\mu_t \sim i.i.d.\left(0, \sigma^2_\mu\right)$. Il s'agit d'un modèle à autocorrélation des erreurs. Nous pouvons citer différentes méthodes d'estimation.

11.2.3.1 Régression sur les différences premières

On procède à deux régressions par la méthode des moindres carrés ordinaires :

1. La première sur le modèle spécifié avec des variables non transformées ;
2. La seconde sur le modèle spécifié avec des variables transformées en différences premières.

Si les deux résultats d'estimation sont proches, nous pouvons conclure que les paramètres du modèle sont suffisamment bien déterminés sans rechercher la liaison éventuelle des erreurs.

11.2.3.2 Correction de l'autocorrélation des erreurs par régression sur les quasi-différences premières

Nous pouvons procéder à la correction de l'autocorrélation des erreurs en utilisant les procédures de correction de l'autocorrélation des erreurs telles que : estimation directe de ρ ; la méthode de Hildreth-Lu et la méthode de Cochrane-Orcutt.

Remarque 11.3

D'autres méthodes peuvent également être utiliser telles que la méthode du maximum de vraisemblance et la méthode des variables instrumentales. La deuxième méthode résout le problème de convergence des estimateurs lorsque nous sommes en présence d'erreurs à moyenne mobile de type :

$$\varepsilon_t = \mu_t - \theta_1 \mu_{t-1} - \cdots - \theta_q \mu_{t-q} \tag{11.12}$$

11.2.4 Prévision d'un ARX(p)

La prévision de la variable endogène X, à l'horizon h, est donnée par :

$$\widehat{X}_{T+h} = \hat{c} + \hat{\phi}_1 \widehat{X}_{T+h-1} + \cdots + \hat{\phi}_p \widehat{X}_{T+h-p} + \hat{\beta} Z_{1,T+h} + \cdots + \hat{\beta}_d Z_{d,T+h} \tag{11.13}$$

Les erreurs qui affectent les valeurs successives de $\widehat{X}_{T+h-1}, \ldots, \widehat{X}_{T+h-p}$ se cumulent d'une période sur l'autre, et ainsi la variance de l'erreur de prévision peut devenir très importante. Ainsi, l'utilisation à des fins de prévision du modèle autorégressif doit donc être limitée à quelques périodes.

Exemple 11.2 *Mesure de relation entre prix officiels du café et prix appliqués*

Un économètre désire tester la relation entre les prix officiels (PO) de la tonne de café et les prix réellement appliqués à l'exportation (PE) par les pays producteurs. Il se propose d'estimer la relation :

$$PO_t = c + \phi_1 PO_{t-1} + \beta_1 PE_t + \varepsilon_t \tag{11.14}$$

dans laquelle le prix officiel est fonction de manière instantanée du prix observé et s'ajuste avec un retard d'un an au prix officiel. Il dispose des données, sur 16 années, présentées au tableau (11.1) :

```
# Chargement des données
import pandas as pd
import numpy as np
df =  pd.read_stata("./donnee/C7EX1.dta",columns=["po","pe"])
df.index = pd.date_range(start="2007-01-31",periods=len(df),freq="Y")
```

Table 11.1 – Prix officiel et prix à l'exportation

	po	pe
2007-12-31	455.0	615
2008-12-31	500.0	665
2009-12-31	555.0	725
2010-12-31	611.0	795
2011-12-31	672.0	870
2012-12-31	748.5	970
2013-12-31	846.0	1095
2014-12-31	954.5	1235
2015-12-31	1090.0	1415
2016-12-31	1243.5	1615
2017-12-31	1390.0	1795
2018-12-31	1559.0	2015
2019-12-31	1781.0	2315
2020-12-31	2046.5	2660
2021-12-31	2311.0	2990
2022-12-31	2551.0	3280

```
# Représentation graphique
import matplotlib.pyplot as plt
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(df.po,label = "PO",color="black");
axe.plot(df.pe,label = "PE",color="red");
axe.set_xlabel('Année');
axe.set_ylabel('prix');
axe.legend();
plt.show()
```

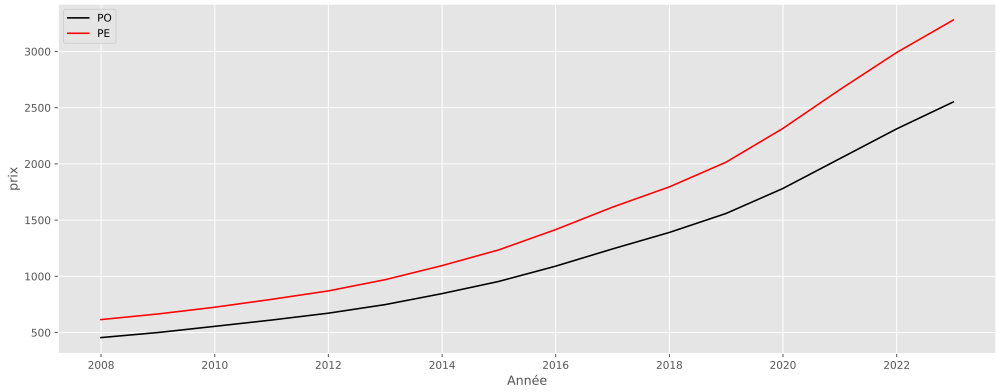


Figure 11.1 – Prix officiel (en noir) et prix à l'exportation (en rouge)

Les résultats de l'estimation du modèle (11.14) sont les suivants :

```
# Estimation du modèle ARX(1)
from statsmodels.tsa.api import AutoReg
model = AutoReg(endog=df.po,lags=1,exog=df.pe,trend='c').fit()
arx_params = extractParams(model, model_type = "arima")
```

Table 11.2 – Coefficients du modèle ARX(1) sur les séries en niveau

	coef	std err	t	P>|t|	[0.025	0.975]
const	-7.0787	1.6557	-4.2755	1.907e-05	-10.3237	-3.8337
po.L1	0.2240	0.0332	6.7526	1.452e-11	0.1590	0.2890
pe	0.6223	0.0228	27.2648	1.110e-163	0.5776	0.6671

L'équation est estimée sur 15 périodes $(T-1)$ car dans le modèle figure une variable retardée d'une période.

```
# Statistique de Durbin et Watson
from statsmodels.stats.stattools import durbin_watson
dw = durbin_watson(model.resid)
print('Durbin-Watson statistic : %.3f'%(dw))
```

```
## Durbin-Watson statistic : 0.626
```

La statistique de Durbin et Watson laisse augurer d'une autocorrélation des erreurs, ce qui est confirmé par le « h » de Durbin :

```
# << h >> de Durbin
import numpy as np; import scipy.stats as st
def h_durbin_watson(dw,sigma,T,seuil = 0.05):
    rho = 1 - (dw/2)
    critical = np.abs(st.norm.ppf(seuil/2))
    h = rho*np.sqrt(T/(1-T*sigma))
    seuil = int(100*seuil)
    print('===='*10)
    print('Test of Hypothesis : h = 0')
    print('===='*10)
    if h >= critical :
        print("On rejete l'hypothèse nulle")
        print('===='*10)
    else:
        print("On accepte l'hypothèse nulle")
        print('===='*10)
    hstat = pd.DataFrame({'Value': h, f'{seuil}% Critical value':critical},
                         index = ['h-statistic'])
    return  print(hstat.round(4))
# Application
sigmaphi1 = model.cov_params().iloc[1,1]
h_durbin_watson(dw,sigmaphi1,15,0.05)
```

```
## ========================================
## Test of Hypothesis : h = 0
```

```
## ======================================
## On rejete l'hypothèse nulle
## ======================================
##                Value  5% Critical value
## h-statistic  2.6832               1.96
```

Nous rejetons l'hypothèse nulle, soit $h \neq 0 \rightarrow \rho \neq 0$.

Nous allons à présent comparer les estimations obtenues à partir du modèle brut à celles obtenues à partir du modèle en différences premières. Soit le modèle :

$$\Delta PO_t = c^{'} + \phi_1^{'} \Delta PO_{t-1} + \beta_1^{'} \Delta PE_t + \varepsilon_t^{'} \tag{11.15}$$

```
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(df.po.diff(periods=1).dropna(),label = "$\Delta$PO",color="black");
axe.plot(df.pe.diff(periods=1).dropna(),label = "$\Delta$PE",color = "red");
axe.set_xlabel('Année');
axe.set_ylabel('prix');
axe.legend();
plt.show()
```

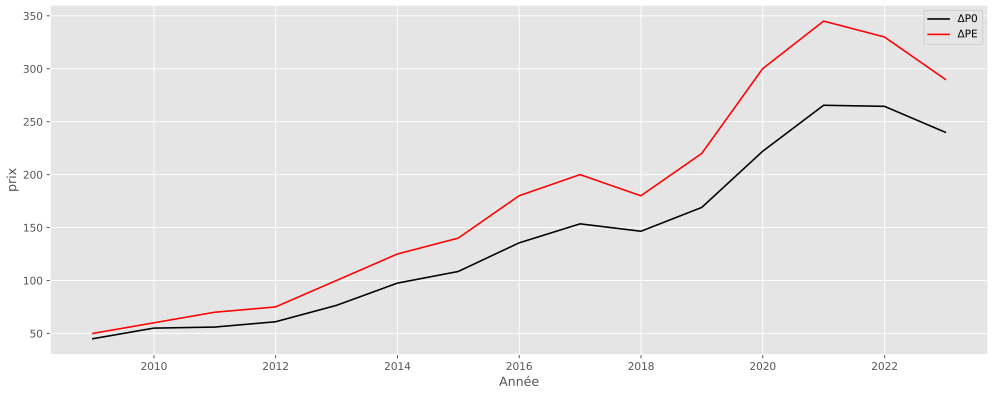


Figure 11.2 – Prix officiel et prix à l'exportation (séries en différence première)

Nous procédons à l'estimation du modèle en différences premières sur 14 ans car nous perdons de nouveau une observation lors du calcul des différences premières de PO_{-1}. Les résultats sont les suivants :

```
# Estimation du modèle 2
model2 = AutoReg(endog = df.po.diff(periods=1).dropna(), lags=1,
                 exog = df.pe.diff(periods=1).dropna(), trend="c").fit()
arx2_params = extractParams(model2, model_type = "arima")
```

Table 11.3 – Coefficients du modèle ARX(1) sur les séries en différence première

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.8916	1.1508	2.5127	1.198e-02	0.6361	5.1472
po.L1	0.2407	0.0214	11.2601	2.065e-29	0.1988	0.2826
pe	0.5980	0.0164	36.3635	1.609e-289	0.5658	0.6302

Nous observons que les différences entre les coefficients de régression sont assez faibles pour ϕ_1 (0.224 et 0.241) comme pour β_1 (0.622 et 0.598) ; nous pouvons considérer les résultats obtenus à la première régression comme valides.

12

Introduction aux modèles à retards échelonnés

Sommaire

La théorie économique postule couramment l'existence d'effets étalés dans le temps entre les différentes grandeurs économiques, l'ignorance de ce cas de figure et le fait de se contenter uniquement de variables prises de façon instantanée pourrait induire en erreur lors de la prise de décision. D'où tout l'intérêt d'étudier des modèles qui prennent en considération la notion de temps dans l'établissement des relations entre les variables étudiées.

L'usage des modèles à retards échelonnés en économie se justifie en général pour deux raisons principales : l'introduction de retards échelonnés dans les relations économiques peut tout d'abord rendre compte de la formation des anticipations des agents, compte tenu des observations passées des variables qui président à leur prise de décision ; ou cette introduction peut se proposer de rendre compte des délais de réalisation de décisions ou d'actions économiques ; les délais de réalisation et la structure globale des retards n'étant pas connus, l'économétrie des modèles à retards échelonnés se propose alors de les rechercher et de les estimer au mieux.

Sur le plan strictement théorique, la justification de l'introduction des modèles à retards échelonnés en économie ne présente guère de difficulté, l'abondance de la littérature consacrés à ce sujet pouvant le cas échéant le confirmer.

12.1 Présentation du modèle

12.1.1 Formulation générale

Dans certaines spécifications de modèles économétriques temporelles, nous pouvons postuler que la variable à expliquer (variable endogène) dépend des valeurs prises par une variable exogène à des époques antérieures, tel que :

$$X_t = c + \theta_0 Z_t + \theta_1 Z_{t-1} + \theta_2 Z_{t-2} + \cdots + \theta_p Z_{t-p} + \varepsilon_t = c + \sum_{j=0}^{p} \theta_j Z_{t-j} + \varepsilon_t \tag{12.1}$$

avec $\varepsilon_t \sim i.i.d.\,(0, \sigma_\varepsilon^2)$.

De manière générale, l'effet de la variable exogène est supposé de plus en plus faible avec le temps :

$$\theta_0 > \theta_1 > \theta_2 > \cdots > \theta_p \tag{12.2}$$

Ce modèle (12.1) peut se simplifier dans son écriture en utilisant l'opérateur retard B, soit :

$$X_t = c + \sum_{j=0}^{p} \theta_j B^j Z_t + \varepsilon_t = c + \left[\sum_{j=0}^{p} \theta_j B^j\right] Z_t + \varepsilon_t = c + \Theta(B) Z_t + \varepsilon_t \tag{12.3}$$

où $\Theta(B)$ est un polynôme de degré p, tel que :

$$\Theta(B) = \theta_0 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_p B^p \tag{12.4}$$

Le nombre de retards, p, peut être fini ou infini. Cependant, la somme des coefficients θ_j tend vers une limite finie, sinon X_t serait un processus explosif.

Exemple 12.1

Pour $B = 1$, le polynôme $\Theta(1) = \theta_0 + \theta_1 + \theta_2 + \cdots + \theta_p$ permet de mesurer, à long terme, l'impact d'une variation la variable explicative Z_t d'une quantité ΔZ sur la valeur de X_t. En effet, les coefficients θ_j représentent les multiplicateurs instantanés et leur somme le multiplicateur cumulé.

12.1.2 Retard moyen

La notion de retard moyen permet d'appréhender le laps de temps qui correspond à la valeur moyenne des coefficients θ_j. Il est égal à la moyenne pondérée des coefficients, soit :

$$\overline{B} = \frac{\sum_{j=0}^{p} j\theta_j}{\sum_{j=0}^{p} \theta_j} = \frac{\Theta'(1)}{\Theta(1)} \tag{12.5}$$

12.2 Identification de p et estimation des paramètres

12.2.1 Identification de p

Lorsque la valeur p du nombre de retards du modèle (12.1) est inconnue, il existe des critères statistiques permettant de la déterminer. Il s'agit de déterminer quelle est la période maximum d'influence de la série explicative. Notons qu'il peut très bien arriver, c'est même souvent le cas, que des coefficients de rang inférieur au décalage p ne soit pas significativement différents de 0.

12.2.1.1 Test de Fisher

Le test le plus naturel est celui de Fisher dans lequel nous testons l'hypothèse de la nullité des coefficients de régression pour les retards supérieurs à p. La formalisation des hypothèses est la

suivante, lorsque l'on teste, d'une manière descendante, une valeur de p comprise entre 0 et M : $0 < p < M$.

$$\begin{cases} H_0^1 : p = M-1 \rightarrow \theta_M = 0 \quad H_1^1 : p = M \rightarrow \theta_M \neq 0 \\ H_0^2 : p = M-2 \rightarrow \theta_{M-1} = 0 \quad H_1^2 : p = M-1 \rightarrow \theta_M - 1 \neq 0 \\ \vdots \\ H_0^i : p = M-i \rightarrow \theta_{M-i+1} = 0 \quad H_1^i : p = M-i+1 \rightarrow \theta_{M-i+1} \neq 0 \end{cases}$$

Chacune de ces hypothèses fait l'objet du classique test de Fisher, soit :

$$F_1^* = \frac{\left(SCR_{M-i} - SCR_{M-i+1}\right)/1}{SCR_{M-i+1}/(T-M+i-3)} \tag{12.6}$$

que l'on compare du F lu dans la table à 1 et $(T-M+i-3)$ degrés de liberté.

Dès que, pour un seuil donné, le F empirique est supérieur au F lu, nous rejetons alors l'hypothèse H_0^i et la procédure est terminée. La valeur du retard est égale à $M-i+1 : p = M-i+1$.

Afin de pouvoir procéder à ce test dans ses conditions d'application, la Somme des Carrés Totaux doit rester constante d'un estimation à l'autre. Cela oblige donc à estimer les différents modèles avec un nombre d'observations identique correspondant donc au nombre d'observations réellement disponible pour le décalage le plus important, chaque décalage entraînant la perte d'une donnée.

12.2.1.2 Critère de Akaike (AIC)

Une autre méthode consiste à retenir comme valeur de p celle qui minimise la fonction de Akaike (1973) qui est donnée par :

$$AIC(p) = \ln\left(\frac{SCR_p}{T}\right) + \frac{2p}{T} \tag{12.7}$$

avec SCR_p représente la Somme des Carrés des Résidus pour le modèle à p retards ; T le nombre d'observations disponible (chaque retard entraîne la perte d'une observation) et ln le logarithme népérien.

```
# Critère de Akaike
import numpy as np
def Akaike(resid,T):
    h,scr= T-len(resid),np.sum(resid**2)
    return (np.log(scr/len(resid))+(2*h)/len(resid))
```

12.2.1.3 Critère de Schwarz (SC)

Enfin, une méthode très proche de la précédente consiste à retenir la valeur de p qui minimise la fonction de Schwarz (1978) :

$$SC(p) = \ln\left(\frac{SCR_p}{T}\right) + \frac{p\ln(T)}{T} \tag{12.8}$$

```
# Critère de Schwarz
def Schwarz(resid,T):
    h,scr = T-len(resid),np.sum(resid**2)
    return (np.log(scr/len(resid))+(h*np.log(len(resid)))/len(resid))
```

Exemple 12.2 *Détermination du nombre de retards dans un modèle DL*

La théorie économique postule que les dépenses d'investissement (notées Y_t) peuvent être expliquées par les profits passés (notés X_t). Le modèle prend la forme d'un modèle à retards échelonnés tel le modèle (12.1). Nous disposons des données trimestrielles concernant l'industrie chimique française.

```
# Chargement des données
import pandas as pd
df =  pd.read_stata("./donnee/C7EX2.dta",columns=["y","x"])
df.index = pd.date_range(start='3/1/2012',periods=len(df), freq='3M')
```

```
# Représentation graphique
import matplotlib.pyplot as plt
fig, (axe1,axe2) = plt.subplots(1,2,figsize=(16,6))
axe1.plot(df.y, color = "black");
axe1.set_xlabel("Date");
axe1.set_ylabel("investissement");
axe2.plot(df.x, color = "black");
axe2.set_xlabel("Date");
axe2.set_ylabel("profit");
plt.tight_layout();
plt.show()
```

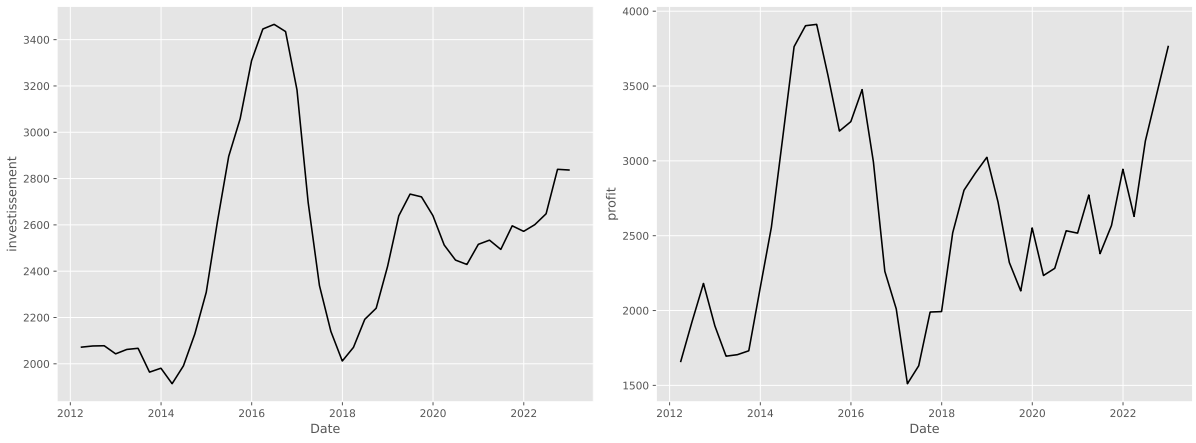


Figure 12.1 – Investissement et profit

N'étant pas certains de la spécification exacte du modèle, nous désirons tout d'abord rechercher le nombre de décalages trimestriels qui semblent avoir un effet sur les dépenses d'investissement. Puis après avoir déterminé le nombre de retards, nous calculerons le délai moyen.

Sous Python, la fonction ARDL permet d'estimer les paramètres d'un modèle à retards échelonnés. Nous calculons les deux critères : Akaike et Schwarz.

```
# Estimation
from statsmodels.tsa.api import ARDL
result = pd.DataFrame(np.zeros(shape=(11,2)),columns=['Akaike',"Schwarz"])
for i in range(result.shape[0]):
    model = ARDL.from_formula("y~x",df,lags=0, order=i, trend='c').fit()
    result.loc[i,"Akaike"] = Akaike(model.resid,model.nobs)
    result.loc[i,"Schwarz"] = Schwarz(model.resid,model.nobs)
result = result.rename_axis('Décalage').reset_index()
```

Table 12.1 – Résultats de la recherche du nombre de décalages optimal

Décalage	Akaike	Schwarz
0	11.96	11.96
1	11.50	11.54
2	11.04	11.12
3	10.55	10.68
4	10.25	10.42
5	9.99	10.21
6	9.84	10.10
7	9.88	10.19
8	9.96	10.31
9	10.03	10.43
10	10.10	10.55

On représente graphiquement les résultats.

```
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(result.Akaike,label = "Akaike", color = "black");
axe.plot(result.Schwarz,label = "Schwarz",color = "red");
axe.text(x=6-0.3,y=np.min(result.Akaike)-0.05,s=round(np.min(result.Akaike),2),
         color="black");
axe.text(x=6-0.3,y=np.min(result.Schwarz)-0.05,s=round(np.min(result.Schwarz),2),
         color="red");
axe.axvline(x=6,linestyle = "--",color="blue");
axe.set_xticks(result.index);
axe.set_xticklabels(result.index);
axe.set_xlabel("p");
axe.set_ylabel("valeur");
axe.legend();
plt.show()
```

Nous observons immédiatement que les minima des deux critères ($AIC(p)$ et $SC(p)$) sont situés sur la ligne 6, correspondant donc à 6 décalages.

12.2.2 Estimation des paramètres

L'estimation des paramètres du modèle (12.1) soulève deux types de difficultés :

- En pratique, la valeur de p qui mesure l'importance du nombre de retards est rarement connue, même si nous en connaissons l'ordre de grandeur ;
- Le second problème résulte de la colinéarité entre les variables exogènes décalées. En effet, lorsque le nombre de retards est important, la colinéarité entre les variables explicatives décalées risque d'entraîner une imprécision dans l'estimation des coefficients.

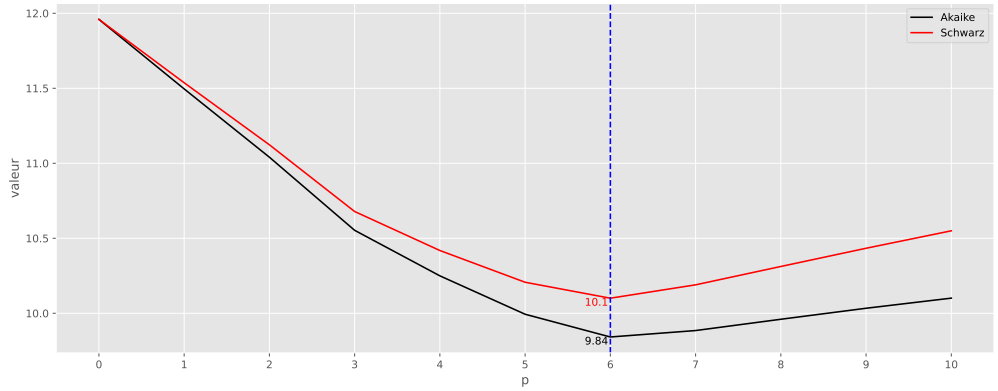


Figure 12.2 – Identification de p (Critère de Akaike et de Schwarz)

L'estimation des modèles à retards échelonnés en économie se fonde sur le recours à trois méthodes : la méthode des moindres carrés ordinaires, la méthode polynomiale d'Almon et la méthode de la régression spectrale. Nous présentons uniquement celle d'Almon.

12.2.2.1 La méthode polynomiale d'ALMON

La méthode polynomiale d'ALMON a été mise au point en 1964 par S. ALMON. Les deux objectifs principaux poursuivis par cette méthode sont les suivantes : d'une part la réduction de la colinéarité des variables explicatives entre elles et d'autre part, une spécification suffisamment souple de la distribution des retards pour permettre de restituer tout profil d'évolution temporelle des coefficients.

Pour cela, ALMON a choisi d'exprimer la distribution des retards par une fonction polynomiale. En effet, on sait que pour toute suite de n points, il est possible de trouver une fonction polynomiale de degré $(n-1)$ passant par tous les points, donc de rendre compte de profils forts complexes.

La méthode des retards d'ALMON est très utilisée. Cette technique consiste à imposer aux coefficients d'appartenir à un même polynôme de degré q, tel que :

$$\theta_j = \phi_0 + \phi_1 j + \phi_2 j^2 + \cdots + \phi_q j^q = \sum_{i=0}^{q} \phi_i j^i \tag{12.9}$$

Exemple 12.3

Pour un polynôme de degré 2 ($q = 2$), nous avons la séquence des coefficients suivant :

$$\begin{cases} \theta_0 = \phi_0 \\ \theta_1 = \phi_0 + \phi_1 + \phi_2 \\ \theta_2 = \phi_0 + 2\phi_1 + 4\phi_2 \\ \theta_4 = \phi_0 + 3\phi_1 + 9\phi_2 \end{cases} \quad \text{ou encore} \quad \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 2^2 \\ 1 & 3 & 3^2 \end{pmatrix} \begin{pmatrix} \phi_0 \\ \phi_1 \\ \phi_2 \end{pmatrix}$$

Nous pouvons généraliser la formulée précédente pour p retards et un polynôme de degré q :

$$\begin{pmatrix}\theta_0\\ \theta_1\\ \theta_2\\ \vdots\\ \theta_p\end{pmatrix} = \begin{pmatrix}1 & 0 & 0 & \cdots & 0\\ 1 & 1 & 1 & \cdots & 1\\ 1 & 2 & 2^2 & \cdots & 2^q\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ 1 & p & p^2 & \cdots & p^q\end{pmatrix}\begin{pmatrix}\phi_0\\ \phi_1\\ \phi_2\\ \vdots\\ \phi_q\end{pmatrix} \tag{12.10}$$

Le modèle initial $X = Z\theta + \varepsilon$ peut alors s'écrire $X = ZH\phi + \varepsilon = Y\phi + \varepsilon$ dans lequel $ZH = Y$ est la matrice des observations des nouvelles variables explicatives.

Les estimations des coefficients ϕ_i sont obtenues en appliquant la méthode des moindres carrés au modèle transformé. les valeurs estimées de ϕ permettent d'obtenir à partir des relations (12.9), les estimations des $p+1$ paramètres $\theta(\widehat{\theta}_0, \widehat{\theta}_1, \cdots, \widehat{\theta}_p)$. Cette méthode appelle à plusieurs remarques :

- Faisant appel aux techniques de régression habituelles, on se retrouve confronté aux problèmes de celles-ci : hypothèse de stationnarité du modèle, estimateur à variance non minimale s'il y a autocorrélation des erreurs, etc...
- On remarque également que cette nécessite non seulement la spécification a priori du nombre de retards, mais aussi la spécification du degré du polynôme. Le choix de ce dernier paramètre est particulièrement délicat et une mauvaise spécification peut introduire un biais important - qui n'est pas toujours décelé par les tests statistiques classiques - lors de l'estimation de certains coefficients.

Les logiciels d'économétrie traitant des estimateurs d'ALMON offrent la possibilité d'introduire des contraintes supplémentaires concernant les valeurs des coefficients θ_{-1} et/ou θ_{p+1} afin qu'elles soient nulles.Cela se justifie par le fait que θ_{-1} est le coefficient théorique de la variable Z_{t+1} qui est sans influence sur X_t.

Exemple 12.4 *Estimation des paramètres du modèle*

Le résultat de l'estimation pour 6 retards est le suivant :

```
# Estimation du modèle avec retard p=6
model = ARDL.from_formula("y~x",df,lags=0, order=6, trend='c').fit()
dl_params = extractParams(model, model_type = "arima")
```

Table 12.2 – Coefficients du modèle à retards échelonnés

	coef	std err	t	P>|t|	[0.025	0.975]
const	501.5414	154.8486	3.2389	2.581e-03	187.4939	815.5889
x.L0	-0.0114	0.0815	-0.1397	8.897e-01	-0.1767	0.1540
x.L1	0.0613	0.1249	0.4905	6.268e-01	-0.1921	0.3146
x.L2	0.2276	0.1196	1.9022	6.517e-02	-0.0151	0.4702
x.L3	0.1679	0.1130	1.4862	1.459e-01	-0.0612	0.3971
x.L4	0.1187	0.1275	0.9316	3.578e-01	-0.1398	0.3772
x.L5	0.0002	0.1369	0.0012	9.990e-01	-0.2775	0.2778
x.L6	0.2372	0.0841	2.8213	7.735e-03	0.0667	0.4077

Nous remarquons que la probabilité critique du coefficient de la variable `x.L6` est largement inférieure à 0.05 ; le coefficient est donc significativement différent de 0. Le modèle à retards échelonnés comporte 6 retards : l'investissement des entreprises de ce secteur est fonction des profits réalisés sur les six derniers trimestres, soit un an et dernier. Il convient de noter que seul le coefficient du sixième retard est significativement différent de 0.

Le graphique (12.3) illustre la structure de pondération des retards pour le modèle ainsi estimé.

```
lag = pd.DataFrame({"lag": range(0,7),"coef" : model.params[1:]})
# Représentation graphique
fig, axe = plt.subplots(figsize = (16,6))
axe.plot(lag.lag, lag.coef, color = "black");
axe.set_xlabel("Retards");
axe.set_ylabel("Coefficients");
plt.show()
```

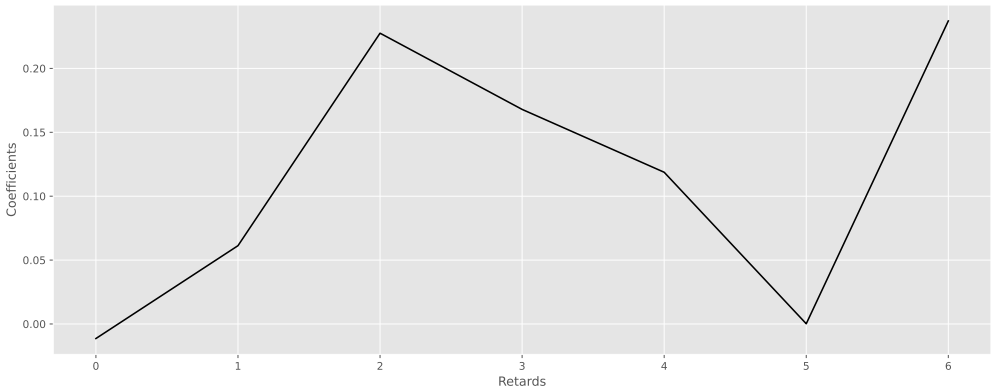


Figure 12.3 – Structure de la pondération des retards

Le retard moyen est égal à :

```
# Retard moyen
Dbar = np.dot(range(0,7),model.params[1:])/np.sum(model.params[1:])
print('Retard moyen : %.2f'%(Dbar))
```

```
## Retard moyen : 3.64
```

Le délai moyen de réaction est de 3.64 trimestres, soit presque une année.

Remarque 12.1

Sous Python, nous pouvons également estimer les coefficients d'un modèle ARDL selon deux approches différentes :

12.2.2.2 Approche 1

Nous définissons une fonction lag pour le décalage des séries.

```
# Lag function
def lag(x, n):
    if n == 0:
        return x
    if isinstance(x, pd.Series):
        return x.shift(n)
    else:
```

```
        x = pd.Series(x)
        return x.shift(n)
    x = x.copy()
    x[n:] = x[0:-n]
    x[:n] = np.nan
    return x
```

Ensuite, nous pouvons estimer les paramètres du modèle en utilisant les moindres carrés ordinaires.

```
# Estimation du modèle
import statsmodels.formula.api as smf
formula = 'y~lag(x, 0)+lag(x,1)+lag(x,2)+lag(x,3)+lag(x,4)+lag(x,5)+lag(x,6)'
model2 = smf.ols(formula = formula, data = df).fit()
dl2_params = extractParams(model2, model_type = "lm")
```

Table 12.3 – Coefficients du modèle à retards échelonnés

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	501.5414	154.8486	3.2389	2.929e-03	185.2984	817.7845
lag(x, 0)	-0.0114	0.0815	-0.1397	8.898e-01	-0.1779	0.1551
lag(x, 1)	0.0613	0.1249	0.4905	6.274e-01	-0.1938	0.3164
lag(x, 2)	0.2276	0.1196	1.9022	6.679e-02	-0.0168	0.4719
lag(x, 3)	0.1679	0.1130	1.4862	1.477e-01	-0.0628	0.3987
lag(x, 4)	0.1187	0.1275	0.9316	3.590e-01	-0.1416	0.3790
lag(x, 5)	0.0002	0.1369	0.0012	9.990e-01	-0.2794	0.2798
lag(x, 6)	0.2372	0.0841	2.8213	8.403e-03	0.0655	0.4089

12.2.2.3 Approche 2

La deuxième approche consiste à créer la série des variables retardées.

```
# 2e approche d'estimation
import statsmodels.api as sm
X = pd.concat([df.x,df.x.shift(1),df.x.shift(2),df.x.shift(3),
               df.x.shift(4),df.x.shift(5),df.x.shift(6)], axis=1)
X.columns = ['x.L0','x.L1','x.L2','x.L3','x.L4','x.L5','x.L6']
xvar = X.dropna()
xvar = sm.add_constant(xvar) # Ajout de la constante
yvar = df.y.iloc[6:]
# Estimation des paramètres
model3 = sm.OLS(yvar,xvar).fit()
dl3_params = extractParams(model3, model_type = "lm")
```

12.3 Distribution infinie des retards

Dans ce type de modèle, l'effet de la variable exogène n'est plus limité dans le temps, mais a un effet illimité, bien que, naturellement, cet effet s'estompe pour les périodes anciennes.

Table 12.4 – Coefficients du modèle à retards échelonnés

	coef	std err	t	P>|t|	[0.025	0.975]
const	501.5414	154.8486	3.2389	2.929e-03	185.2984	817.7845
x.L0	-0.0114	0.0815	-0.1397	8.898e-01	-0.1779	0.1551
x.L1	0.0613	0.1249	0.4905	6.274e-01	-0.1938	0.3164
x.L2	0.2276	0.1196	1.9022	6.679e-02	-0.0168	0.4719
x.L3	0.1679	0.1130	1.4862	1.477e-01	-0.0628	0.3987
x.L4	0.1187	0.1275	0.9316	3.590e-01	-0.1416	0.3790
x.L5	0.0002	0.1369	0.0012	9.990e-01	-0.2794	0.2798
x.L6	0.2372	0.0841	2.8213	8.403e-03	0.0655	0.4089

$$X_t = \sum_{j=0}^{+\infty} \theta_j Z_{t-j} + c + \varepsilon_t = \left[\sum_{j=0}^{+\infty} \theta_j B^j\right] Z_t + c + \varepsilon_t \tag{12.11}$$

Afin de se ramener à un nombre fini de paramètres à estimer, nous devons postuler une forme particulière que peut prendre la succession des coefficients $\theta_0, \theta_1, \theta_2, ...$ Nous pouvons admettre plusieurs types de spécifications, deux modèles particuliers - les plus utilisés - font l'objet d'une présentation : une décroissance géométrique des effets de la variable exogène avec le temps ; une croissance suivie d'une décroissance.

12.3.1 Modèle de Koyck (progression géométrique)

Le modèle de Koyck postule une décroissance géométrique de la structure des retards telle que, dans le modèle (12.11), les coefficients soient liés de la manière suivante :

$$\begin{cases} \theta_1 = \lambda\theta_0 \\ \theta_2 = \lambda^2\theta_0 \end{cases} \tag{12.12}$$

et en général $\theta_j = \lambda^j\theta_0$ avec $0 < \lambda < 1$.

Soit

$$\begin{aligned} X_t &= c + \theta_0 Z_t + \lambda\theta_0 Z_{t-1} + \cdots + \lambda^i\theta_0 Z_{t-i} + \cdots + \varepsilon_t \\ &= c + \theta_0 \left(Z_t + \lambda Z_{t-1} + \lambda^2 Z_{t-2} + \cdots + \lambda^i Z_{t-i} + \cdots\right) + \varepsilon_t \end{aligned}$$

La fonction $\Theta(B)$ associé s'écrit alors : $\Theta(B) = \theta_0 + \lambda\theta_0 B + \lambda^2\theta_0 B^2 + \cdots$. Le modèle (12.11) peut alors s'écrire :

$$\Phi(B)X_t = \Phi(B)c + \Phi(B)\Theta(B)Z_t + \Phi(B)\varepsilon_t \tag{12.13}$$

avec $\Phi(B) = \Theta(B)^{-1}$. Or $\Theta(B) = \theta_0\left(1 + \lambda B + \lambda^2 B^2 + \cdots\right) = \theta_0\left(1 - \lambda B\right)^{-1}$. Nous avons donc $\Phi(B) = \left(1 - \lambda B\right)/\theta_0$. Soit :

$$\left(1 - \lambda B\right) X_t = \theta_0 Z_t + \left(1 - \lambda\right) c + \left(1 - \lambda B\right) \varepsilon_t \tag{12.14}$$

ou encore :

$$X_t = \lambda X_{t-1} + \theta_0 Z_t + (1 - \lambda)c + \varepsilon_t - \lambda\varepsilon_{t-1} \tag{12.15}$$

qui est un modèle autorégressif à erreurs liées.

Cette transformation - qui consiste à passer d'un modèle à retards échelonnés à un modèle autorégressif - est habituellement appelée « transformation de Koyck ». Nous passons d'un modèle à retards échelonnés, difficile à estimer par l'abondance des paramètres, à un modèle autorégressif, simple dans sa spécification.

Exemple 12.5 *Spécification des retards de Koyck*

Nous reprenons les données de l'exemple précédent concernant la relation Investissement/profit. On nous demande d'estimer les paramètres du modèle si on suppose que les coefficients suivent une spécification de Koyck. D'après (12.15), l'estimation des paramètres du modèle s'effectue sous la forme d'un modèle autorégressif à autocorrélation des erreurs.

```
# Estimation de Koyck
model4 = ARDL.from_formula("y~x",df,lags=1, order=0, trend='c').fit()
dl4_params = extractParams(model4, model_type = "arima")
```

Table 12.5 – Estimation de Koyck

	coef	std err	t	P>|t|	[0.025	0.975]
const	-222.2354	105.6297	-2.1039	4.172e-02	-435.7210	-8.7498
y.L1	0.9075	0.0359	25.2452	3.436e-26	0.8348	0.9801
x.L0	0.1796	0.0236	7.6000	2.777e-09	0.1318	0.2273

Nous obtenons :

$$\begin{cases} \hat{\lambda} & = 0.907 \\ \hat{\theta}_0 & = 0.179 \\ \hat{c} & = -222.23/(1-0.907) = -2400.23 \end{cases} \tag{12.16}$$

Le modèle peut donc s'écrire :

$$y_t = -2400.23 + 0.179x_t + 0.907 \times 0.179x_{t-1} + 0.907^2 \times 0.179x_{t-2} + \cdots + e_t \tag{12.17}$$

ou encore :

$$y_t = 0.907y_{t-1} + 0.179x_t - 222.23 + \hat{\varepsilon}_t \tag{12.18}$$

12.3.2 Modèle de Solow (Distribution de Pascal)

Les coefficients sont distribués selon :

$$\theta_j = (1-\lambda)^{h+1} C_{h+j}^{j} \lambda^j \tag{12.19}$$

où C_{h+j}^{j} est le coefficient du binôme de Newton, h et λ sont deux paramètres avec $0 < \lambda < 1$ et $h \in \mathbb{N}$.

Remarque 12.2

Pour $h = 0$, nous retrouvons la distribution géométrique de Koyck.

Le modèle général (12.11) s'écrit :

$$X_t = c + \sum_{j=0}^{+\infty} (1-\lambda)^{h+1} C_{h+j}^{j} \lambda^j Z_{t-j} + \varepsilon_t \tag{12.20}$$

ou encore

$$X_t = c + \Theta(B) Z_t + \varepsilon_t \tag{12.21}$$

Une démonstration analogue à la précédente permet d'écrire le modèle (12.21) :

$$\Phi(B) Z_t = \Phi(B) c + \Phi(B)\Theta(B) Z_t + \Phi(B)\varepsilon_t \tag{12.22}$$

avec $\Phi(B) = \Theta(B)^{-1}$ et :

- pour $h = 0 \rightarrow \Phi(B) = (1-\lambda B)/\theta_0$, soit le modèle précédent.
- pour $h = 1 \rightarrow \Phi(B) = (1-\lambda B)^2/\theta_0$. Soit :

$$X_t = 2\lambda X_{t-1} - \lambda^2 X_{t-2} + \theta_0 Z_t + (1 - 2\lambda + \lambda^2) c + \mu_t \tag{12.23}$$

  avec $\mu_t = (1 - 2\lambda B + \lambda^2 B^2)\varepsilon_t = \varepsilon_t - 2\lambda\varepsilon_{t-1} + \lambda^2\varepsilon_{t-2}$.
- pour $h = 2 \rightarrow \Phi(B) = \Phi(B) = (1-\lambda B)^3/\theta_0$. Soit :

$$X_t = 3\lambda X_{t-1} - 3\lambda^2 X_{t-2} + \lambda^3 X_{t-3} + \theta_0 Z_t + (1 - 3\lambda + 3\lambda^2 - \lambda^3) c + \mu_t \tag{12.24}$$

  avec $\mu_t = (1 - 3\lambda B + 3\lambda^2 B^2 - \lambda^3 B^3)\varepsilon_t = \varepsilon_t - 3\lambda\varepsilon_{t-1} + 3\lambda^2\varepsilon_{t-2} - \lambda^3\varepsilon_{t-3}$

Afin de déterminer les valeurs du paramètres h, Maddala and Rao (1971) suggèrent l'utilisation d'une procédure de balayage dont la fonction objectif à maximiser est le coefficient de détermination ajusté ($\overline{R}^2$).

Exemple 12.6 *Estimation des coefficients selon une distribution de Pascal*

Nous reprenons les données concernant la relation Investissement/Profit. On demande de rechercher l'ordre d'une distribution de Pascal et d'estimer les paramètres du modèle pour l'ordre adéquat.

Il s'agit d'estimer des modèles autorégressifs d'ordre $1, 2, 3$, etc. Nous procédons à l'estimation du modèle d'ordre 2.

```
# Estimation selon une distribution de Pascal
model5 = ARDL.from_formula("y~x",df,lags=2, order=0, trend='c').fit()
dl5_params = extractParams(model5, model_type = "arima")
```

Le modèle estimé s'écrit :

$$y_t = 40.78 + 1.42 y_{t-1} - 0.54 y_{t-2} + 0.096 x_t + \hat{\varepsilon}_t \tag{12.25}$$

D'après (12.23), nous obtenons en procédant par identification :

Table 12.6 – Estimation des coefficients selon une distribution de Pascal

	coef	std err	t	P>|t|	[0.025	0.975]
const	40.7818	101.5004	0.4018	6.901e-01	-164.6950	246.2586
y.L1	1.4264	0.1050	13.5906	3.602e-16	1.2140	1.6389
y.L2	-0.5414	0.1062	-5.0993	9.731e-06	-0.7563	-0.3264
x.L0	0.0967	0.0252	3.8374	4.561e-04	0.0457	0.1477

$$\begin{cases} \hat{\lambda} & = \sqrt{0.541} \approx 1.426/2 \approx 0.72 \\ \hat{\theta}_0 & = 0.096 \\ \hat{c} & = 40.78/\left(1 - 2\hat{\lambda} + \hat{\lambda}^2\right) = 40.78/0.0784 = 520.15 \end{cases} \tag{12.26}$$

Le modèle estimé est alors :

$$y_t = 1.426 y_{t-1} - 0.541 y_{t-2} + 0.096 x_t + 40.78 + \hat{\varepsilon}_t \tag{12.27}$$

ou encore :

$$y_t = 520.15 + \sum_{j=0}^{+\infty} (1 - 0.72)^2 \, C_{1+j}^{j} 0.72^j x_{t-j} + e_t \tag{12.28}$$

Exemple 12.7 *Élasticité de long terme*

En considérant le modèle à distribution géométrique des retards où la variable exogène et la variable endogène sont sous forme logarithmique :

$$\log y_t = c + \theta_0 \log x_t + \lambda\theta_0 \log x_{t-1} + \lambda^2\theta_0 \log x_{t-2} + \cdots + \varepsilon_t \tag{12.29}$$

Le modèle, sous forme réduite, s'exprime par :

$$\log y_t = \lambda \log y_{t-1} + \theta_0 \log x_t + c_0 + \mu_t \tag{12.30}$$

L'élasticité de long terme est égal a : $\xi_{LT} = \theta_0/(1-\lambda)$. L'élasticité de court terme est donnée par : $\xi_{CT} = \theta_0$ et le paramètre d'ajustement est λ.

```
# Estimation du modèle sous forme logarithmique
model6 = ARDL.from_formula("y~x",df.apply(lambda x : np.log(x)),
                           lags=1, order=0, trend='c').fit()
dl6_params = extractParams(model6, model_type = "arima")
```

Table 12.7 – Estimation du modèle sur les séries logarithmiques

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.6994	0.2956	-2.3663	2.290e-02	-1.2967	-0.1020
y.L1	0.9049	0.0331	27.3305	1.703e-27	0.8380	0.9718
x.L0	0.1848	0.0218	8.4882	1.744e-10	0.1408	0.2288

L'estimation du modèle Dépenses d'Investissement/Profit sous forme logarithmique conduit aux résultats suivants :

$$\log y_t = 0.904 \log y_{t-1} + 0.184 \log x_t - 0.699 + \mu_t \tag{12.31}$$

soit $\xi_{LT} = \hat{\theta}_0 / \left(1 - \hat{\lambda}\right) = 0.184/(1 - 0.904) = 1.91$.

L'élasticité de long terme est donc égale à 1.91, nous sommes dans la zone des rendements croissants. Lorsque le profit augmente par exemple de 10%, l'investissement à long terme augmente de 19.1%.

13

Introduction à la non stationnarité et à la saisonnalité

Sommaire

Les variables économiques et financières sont rarement des réalisations de processus stationnaires. La non stationnarité des processus peut aussi bien concerner le premier moment (espérance mathématique) que celui du second ordre (variance et covariance du processus).

13.1 Définition des concepts

13.1.1 Opérateur différence

Soit $(X_t)_t$ un processus stochastique. On appelle opérateur différence, noté Δ, l'opérateur définit comme suit :

$$\Delta X_t = X_t - X_{t-1} \tag{13.1}$$

En utilisant l'opérateur retard B, l'équation (13.1) peut se réécrive de la façon suivante :

$$\Delta X_t = (I - B)X_t = X_t - X_{t-1} \tag{13.2}$$

On a les remarques suivantes :

— $\Delta^n X_t = (I - B)^n X_t$, $\forall n \in \mathbb{N}$
— $\Delta_n = (I - B^n)\, X_t = X_t - X_{t-n}$, $\forall n \in \mathbb{N}$

L'opérateur Δ possède les propriétés suivantes :

1. L'opérateur Δ élimine les tendances linéaires.

   En effet, pour un processus de la forme $X_t = \beta_0 + \beta_1 t + \varepsilon_t$ où ε_t est un processus stationnaire, on aura :

   $$\Delta X_t = \beta_1 + (\varepsilon_t - \varepsilon_{t-1}) \tag{13.3}$$

2. Plus généralement, l'opérateur $\Delta^p = (I - B)^p$ élimine les tendances polynomiales de degré p.

   Pour une tendance de degré 2 de type $X_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \varepsilon_t$, on aura :

$$\Delta^2 X_t = 2\beta_2 + (\varepsilon_t - \varepsilon_{t-1} + \varepsilon_{t-2}) \tag{13.4}$$

3. L'opérateur $\Delta_p = (I - B^p)$ élimine une saisonnalité de période p ($p \in \mathbb{N}$).

   On utilise fréquemment des opérateurs Δ_{12} pour des données mensuelles, Δ_7 pour des données journalières et Δ_{24} pour des données horaires.
4. L'opérateur Δ_p est coûteux en information puisqu'il raccourcit la série de p valeurs, les p premières valeurs servant de référence pour la saisonnalité.

13.1.2 Processus intégrés

Une classe importante de processus non stationnaire est celle des processus intégrés. On les retrouve couramment en pratique et ils ont l'avantage de présenter un type de non stationnarité modélisable. Un processus intégré est un processus qui peut être rendu stationnaire par différenciation. Si un processus stochastique doit être différencié d fois pour atteindre la stationnarité, alors il dit intégré d'ordre d et on note $I(d)$. Par définition, les processus stationnaires sont $I(0)$. On parle en principe de processus intégré lorsque l'ordre d'intégration est supérieur ou égal à 1.

Exemple 13.1 *Marche aléatoire sans dérive*

Un exemple de processus intégré est la marche aléatoire définit par :

$$X_t = X_{t-1} + \varepsilon_t \tag{13.5}$$

où (ε_t) est un bruit blanc. On constate que le processus $\Delta X_t = \varepsilon_t$ est stationnaire. Par conséquent, une marche aléatoire est un processus intégré d'ordre 1. En exprimant X_{t-1} en fonction de $X_{t-2}, ...$ et d'une valeur initiale X_0, on obtient :

$$X_t = X_0 + \varepsilon_1 + \varepsilon_2 + \cdots + \varepsilon_t \tag{13.6}$$

La série obéissant à une marche aléatoire prend, à deux dates consécutives, des valeurs proches et la variation par rapport à la date précédente est indépendante du passé. La figure (13.1) présente un bruit blanc et la marche aléatoire qui en est obtenue par intégration.

```
# Simulation d'un bruit blanc et d'une marche aléatoire
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(12345)
x = np.random.normal(size=1000)
y = np.cumsum(x)
# Représentation graphique
fig, axe = plt.subplots(2,1,figsize=(16,7));
axe[0].plot(x, color='black');
axe[0].set_title('a) Bruit blanc');
axe[0].set_ylabel('$\epsilon_t$');
axe[0].set_xlabel('t');
```

```
axe[1].plot(y,color='black');
axe[1].set_title('b) Marche aléatoire');
axe[1].set_ylabel('$X_t$');
axe[1].set_xlabel('t');
fig.tight_layout();
plt.show()
```

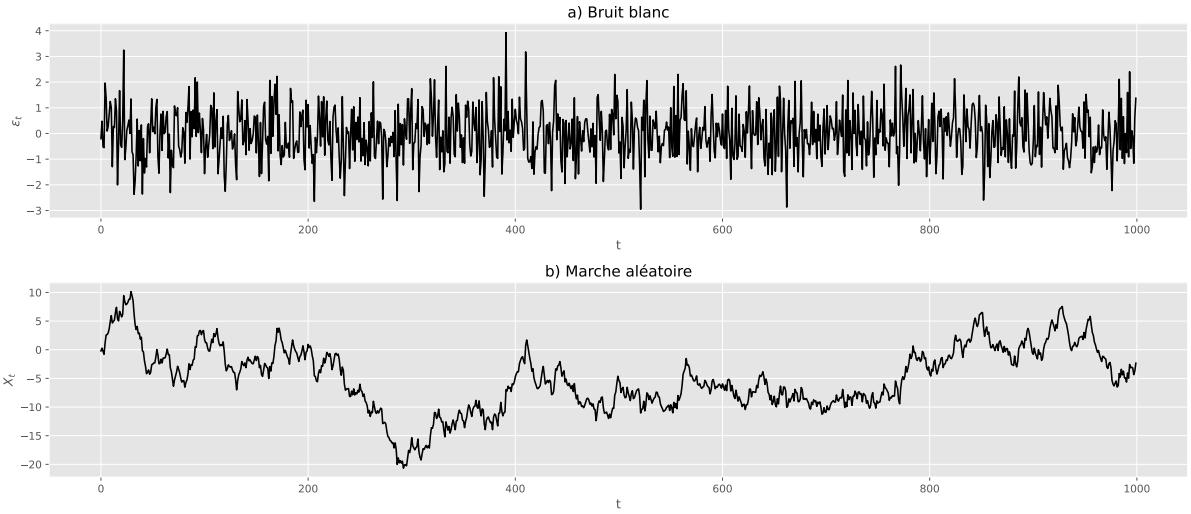


Figure 13.1 – Bruit blanc et sa série intégrée (marche aléatoire)

Exemple 13.2 *Marche aléatoire avec dérive*

On considère maintenant le processus $(X_t)_t$ définit comme suit :

$$X_t = \mu + X_{t-1} + \varepsilon_t \tag{13.7}$$

On dit que X_t est une marche aléatoire avec dérive μ. En exprimant X_{t-1} en fonction de $X_{t-2}, \ldots$ et d'une valeur initial X_0, on obtient :

$$X_t = \mu t + X_0 + \varepsilon_1 + \varepsilon_t + \cdots + \varepsilon_t \tag{13.8}$$

Le graphique de X_t en fonction du temps est donc celui d'une droite à laquelle est superposée une marche aléatoire.

Pour une marche aléatoire, on obtient les caractéristiques suivantes :

$$\begin{cases} \mathbb{E}(X_t) = X_0 \quad (X_0 + \mu t \text{ dans le cas avec dérive}) \\ \mathrm{V}(X_t) = t\sigma_\varepsilon^2 \\ \mathrm{Cov}(X_t, X_{t+s}) = t\sigma_\varepsilon^2 \quad (s > 0) \end{cases} \tag{13.9}$$

13.2 Processus ARIMA

Les processus économiques sont rarement la réalisation de processus stationnaires. C'est la raison pour laquelle - dans l'algorithme de Box et Jenkins - l'étape de la stationnarisation de la

chronique est la première. Si la chronique présente des mouvements de long terme de type non linéaire (logarithme, exponentiel ou puissance) ou des fluctuations importantes autour de ce mouvement, alors il est souvent utile d'utiliser la transformation de Box - Cox qui permet de retrouver des linéarités ou d'amoindrir les fluctuations importantes. Cette transformation a été présenté au chapitre 1. La difficulté de son utilisation réside dans le choix de λ. On peut pour cela réaliser plusieurs transformations avec des λ différents et choisir la valeur de λ en fonction de l'aspect de la chronique transformée. La meilleure solution est de faire recours à des filtres aux différences pour stationnariser ces séries. Le recours à ces filtres permet de définir les processus ARMA intégrés notés ARIMA.

13.2.1 Processus ARIMA non saisonnier

Ce modèle est une généralisation de l'écriture ARMA en ajoutant des racines unitaires dans le polynôme autorégressif.

Définition 13.1 *Processus ARIMA*

On dit qu'une série temporelle $(X_t)_t$ est un processus autorégressif moyenne mobile intégré d'ordre (p,d,q), noté ARIMA(p,d,q), si la série

$$Y_t = (I - B)^d X_t \tag{13.10}$$

admet une écriture ARMA(p,q). On note : $X \sim ARIMA(p, d, q)$.

et d un entier positif appelé ordre d'intégration ou de différentiation. Si Y_t suit un modèle ARMA, on a :

$$\Phi_p(B) Y_t = \Theta(B)\varepsilon_t \tag{13.11}$$

Ce qui peut se réécrire de la manière suivante :

$$\Phi_p(B) \left(I - B\right)^d X_t = \Theta(B)\varepsilon_t \tag{13.12}$$

Si $Y_t = \left(I - B\right)^d X_t$ est stationnaire, alors esimer un processus ARIMA(p,d,q) sur X_t est équivalent à estimer un processus ARMA(p,q) sur Y_t.

Remarque 13.1

- $(I - B)^d$ joue le rôle d'un filtre permettant de stationnariser le processus. Il élimine d racines unitaires, mais aussi les tendances polynomiales de degré d qu'il transforme en constantes. En particulier, il absorbe le décentrage.
- Dans un ARIMA(p,1,q), les incréments $(X_t - X_{t-1})_t$ sont stationnaires. En pratique, lorsqu'une trajectoire n'est pas stationnaire, on la différencie avant de refaire le test. On va très rarement au-delà de $d = 1$.

Exemple 13.3 *Simulation d'un ARIMA (p,d,q)*

Pour tout $t \in \mathbb{Z}$, le processus ARIMA(2,1,2) est engendré par

$$(I - B)(I - \phi_1 B - \phi_2 B^2)X_t = (I - \theta_1 B - \theta_2 B^2)\varepsilon_t \tag{13.13}$$

ou encore

$$X_t - (1+\phi_1)X_{t-1} - (\phi_2 - \phi_1)X_{t-2} + \phi_2 X_{t-3} = \varepsilon_t - \theta_1\varepsilon_{t-1} - \theta_2\varepsilon_{t-2} \tag{13.14}$$

où (ε_t) est un bruit blanc de variance σ^2. L'exemple ci-dessus est simulé avec $\phi_1 = 0.75$, $\phi_2 = -0.25$, $\theta_1 = -0.65$ et $\theta_2 = -0.35$, $(\varepsilon_t) \sim \mathcal{N}(0,1)$ et $n = 1000$.

Nous implémentons une fonction de simulation d'un processus ARIMA(p,d,q). Dans le cas où $d = 0$, la fonction retourne un processus ARMA(p,q) simulé.

```
# Simulation d'un processus ARIMA(p,d,q)
class ARIMA:
    def __init__(self,ar =None,ma=None,d=0,t=0,loc=0,scale=1,burn=10):
        self.ar = ar; self.ma = ma; self.d  = d; self.t  = t; self.loc = loc
        self.scale = scale; self.burn = burn; self.x = None
    def simulate(self,nsample=100):
        # set max lag length AR model & # set max lag length MA model
        p, q = self.ar.shape[0], self.ma.shape[0]
        # simulate n + q error terms
        a = np.random.normal(self.loc,self.scale,
                             (nsample + max(p, q) + self.burn, 1))
        # create array for returned values
        x = np.zeros((nsample + max(p, q) + self.burn, 1))
        # initialize first time series value
        x[0] = a[0]
        for i in range(1, x.shape[0]):
            AR = np.dot(self.ar[0:min(i,p)],np.flip(x[i - min(i, p):i],0))
            MA = np.dot(self.ma[0:min(i+1,q)],np.flip(a[i-min(i,q-1):i+1],0))
            x[i] = AR + MA + self.t
        # add unit roots
        if self.d != 0:
            ARMA = x[-nsample: ];m = ARMA.shape[0]
            # create temp array
            z = np.zeros((m + 1, 1))
            for i in range(self.d):
                for j in range(m):
                    z[j + 1] = ARMA[j] + z[j]
                ARMA = z[1: ]
            x[-nsample: ] = z[1: ]
        self.x = np.array(x[-nsample: ])
        return self.x
```

Nous testons notre fonction avec les paramètres définis précédemment.

```
# Application
np.random.seed(12345)
arparams = np.array([.75, -.25])
maparams = np.array([.65, .35])
ar = np.r_[1, -arparams]
ma = np.r_[1, maparams]
T = 1000
sim_arima = ARIMA(ar=ar,ma=ma,d=1).simulate(T)
```

```
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6));
axe.plot(sim_arima, color='black');
axe.set_xlabel('t');
axe.set_ylabel('$X_t$');
plt.show()
```

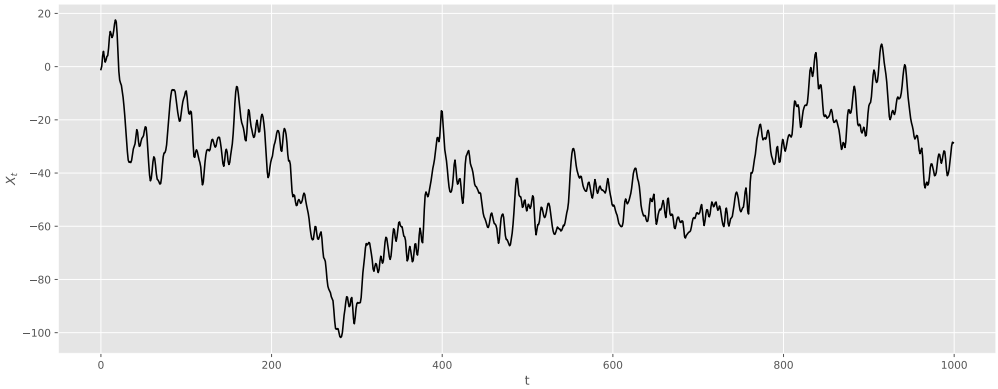


Figure 13.2 – Simulation du processus $(I-B)(I-0.75B+0.25B^2)X_t = (I-0.65B+0.35B^2)\varepsilon_t$

13.2.1.1 Stationnarité et différenciation

Il est important de connaître la nature de la non stationnarité afin de savoir quelle transformation lui appliquée. En effet :

- En présence d'une tendance déterministe, il faut régresser la série d'origine sur une constante et un trend. Puis, modéliser à l'aide d'un modèle ARMA la série des résidus de la régression.
- Si la tendance est stochastique, la meilleure méthode pour l'éliminer est de différencier la série : la différenciation est une opération simple qui consiste à calculer des variations successives des valeurs de la série. On l'utilise lorsque la moyenne d'une série varie au cours du temps. La différenciation d'ordre 1 transforme (X_t) en $(X_t - X_{t-1})$; la différenciation d'ordre d consiste à effectuer d différentiations d'ordre 1.

Exemple 13.4

Si (X_t) possède une tendance linéaire, alors $Y_t = (I-B)X_t$ est différenciée. La chronique (Y_t) est alors stationnaire et est modélisée par un processus ARMA.

$$Y_t = (I-B)X_t = X_t - X_{t-1} \tag{13.15}$$

Remarque 13.2

Pour stationnariser la série X_t à tendance linéaire, on applique un filtre de la forme $(I-B)$. Ce filtre permet d'enlever la tendance linéaire de la série, ce qui revient donc à choisir l'entier d du processus ARIMA égale à 1.

```
# différenciation
data_diff = default_data.diff(periods= lags)
```

Les autocorrélations simples des processus ARIMA décroissent très lentement vers 0 comme le montre la figure (13.3).

```
# Autocorrélation ARMA(2,1,2)
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
fig, (axe1, axe2) = plt.subplots(1,2,figsize=(16,6))
plot_acf(sim_arima,ax=axe1);
axe1 = plot_colors(axe1);
axe1.set_xlabel("Lag");
axe1.set_ylabel("ACF");
plot_pacf(sim_arima,ax=axe2);
axe2 = plot_colors(axe2);
axe2.set_xlabel("Lag");
axe2.set_ylabel("PACF");
plt.tight_layout();
plt.show()
```

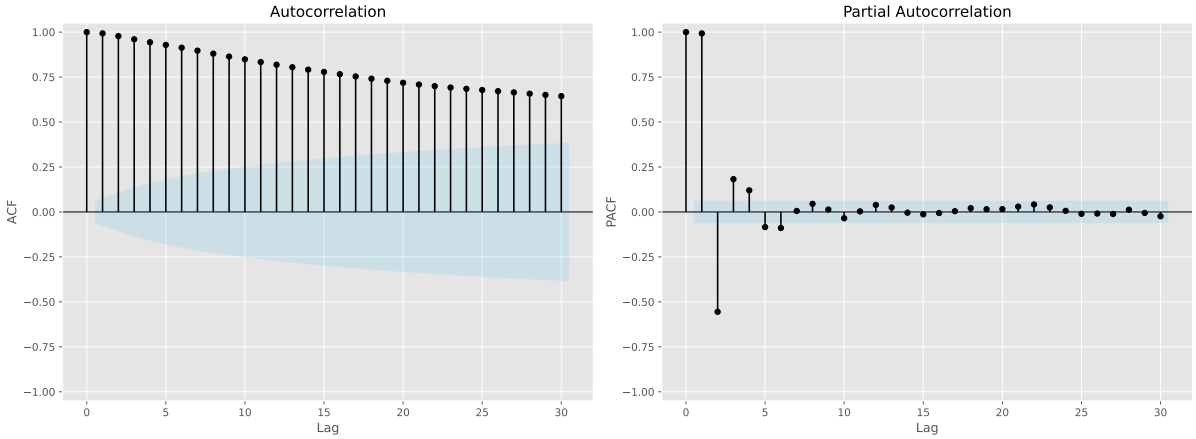


Figure 13.3 – Corrélogrammes simple et partielle du processus $(I-B)(I-0.75B+0.25B^2)X_t = (I-0.65B+0.35B^2)\varepsilon_t$

13.2.2 Le processus ARIMA saisonnier

Les processus SARIMA constituent une généralisation des processus ARIMA pour prendre en compte une saisonnalité des données. On introduit deux nouveaux opérateurs :

$$\Phi_P(B) = I - \alpha_1 B - \cdots - \alpha_P B^P \tag{13.16}$$
$$\Theta_Q(B) = I - \beta_1 B - \cdots - \beta_Q B^Q \tag{13.17}$$

où P et Q sont des entiers. Ce sont des polynômes générateurs respectifs d'un AR(P) saisonnier et d'un MA(Q) saisonnier.

Définition 13.2 *Processus* $SARIMA(p,d,q) \times (P,D,Q)_S$

On dit que $(X_t)_t$ est un processus autorégressif moyenne mobile intégré et saisonnier d'ordre $(p, d, q) \times (P, D, Q)_S$, si la série

$$Y_t = (I - B^S)^D (I - B)^d X_t \tag{13.18}$$

admet l'écriture ARMA

$$\Phi_s(B^S)\Phi(B)Y_t = \Theta_s(B^S)\Theta(B)\varepsilon_t \tag{13.19}$$

Ses ordres d'intégration d'ordre d et D et sa période est $S \geq 2$.

On dit que X_t est un processus SARMA(p,q)(P,Q) s'il vérifie l'équation :

$$\Phi_p(B)\Phi_P(B^S)X_t = \Theta_q(B)\Theta_Q(B^S)\varepsilon_t \tag{13.20}$$

Remarque 13.3

- $(I - B^S)^D (I - B)^d$ joue le rôle de filtrage qui stationnarise et élimine les tendances saisonnières de période S ainsi que certaines tendances polynomiales.
- En pratique, lorsqu'une trajectoire est saisonnière, on la différencie (périodiquement et/ou simplement) pour la stationnariser. Ensuite, on détermine p et q normalement et P et Q en regardant les retards espacés périodiquement uniquement. On cherche à ce qu'ils satisfassent des critères (ACF, PACF) compatibles avec une modélisation ARMA(P,Q).

Exemple 13.5

Si $(X_t)_t$ a une composante de période 12, alors $Y_t = (I - B^{12})\, X_t$ est désaisonnalisée. La série (Y_t) est alors « saisonnièrement » stationnaire et est modélisée par un processus $ARMA$ (saisonnier), auquel on superpose un ARMA classique.

$$Y_t = (I - B^{12})\, X_t = X_t - X_{t-12} \tag{13.21}$$

Pour stationnariser la série X_t saisonnière, on applique un filtre de la forme $(I - B^{12})$. Ce filtre permet de faire disparaître la saisonnalité de la série.

```
# Différenciation saisonnière
def difference(x, differences=12):
    diff = list()
    for i in range(differences, len(x)):
        value = x[i] - x[i - differences]
        diff.append(value)
    return np.array(diff)
```

Si (X_t) possède une tendance et un saisonnalité de 12 mois, alors $Y_t = (I - B)\,(I - B^{12})\, X_t$ est stationnarisée. La chronique (Y_t) est alors stationnaire et est modélisée par un processus ARMA classique :

$$Y_t = (I - B)\,(I - B^{12})\, X_t = X_t - X_{t-1} - X_{t-12} + X_{t-13} \tag{13.22}$$

Remarque 13.4

Pour stationnariser la série X_t possédant une tendance linéaire et une saisonnalité, on applique successivement un filtre de la forme $(I-B)$ et un filtre de la forme $(I-B^{12})$. Dans un premier temps, le filtre $(I-B)$ permet d'enlever la tendance linéaire de la série, et dans un second temps le filtre $(I-B^{12})$ permet de faire disparaître la saisonnalité.

```
# Enlever la tendance,puis la saisonnalité
data_diff_1_12 = difference(default_data.diff(periods=1),differences=12)
```

13.2.3 Identification de d

Si la trajectoire observée est issue d'un processus stationnaire, on dira que $(X_t)_{t\in\mathbb{Z}}$ est intégré d'ordre zéro, sinon on suppose qu'il existe un entier $d>0$ tel que $(I-B)^dX_t$ est stationnaire, et on dira que le processus $(X_t)_{t\in\mathbb{Z}}$ est intégré d'ordre d. Cependant, dans la majorité des cas rencontrés en pratique, l'entier d correspondant à l'ordre d'intégration est inférieur ou égal à 1. Ainsi, pour déterminer d, dans le cas de la modélisation $ARIMA$, deux hypothèses s'imposent :

$$H_0 : d = 0 \quad \text{contre} \quad H_1 : d = 1 \tag{13.23}$$

On peut se servir des tests de racine unitaire de Dickey - Fuller (Dickey and Fuller (1979),Dickey and Fuller (1981)) ou de Phillips and Perron (1988) pour le choix de d à partir des données dont on dispose. On retiendra en pratique, que la présence d'une tendance linéaire entraîne le choix $d=1$ et qu'une moyenne constante entraîne le choix $d=0$.

13.3 Tests de racine unitaire

Des méthodes traditionnelles de détection de la non stationnarité en moyenne correspondent aux techniques graphiques d'analyse de la série par rapport au temps. On notera que la courbe est sans tendance et coupe souvent l'axe du temps si la série est stationnaire. Par ailleurs, ce constat peut être renforcé par l'étude du corrélogramme (test de Barlett). Dès lors, des tests plus « rigoureux » apparaissent comme indispensables, parallèlement aux analyses graphiques.

Il existe deux classes de tests de racines unitaires : les tests qui adoptent comme H_0 la non stationnarité (DF, ADF, PP) et les tests qui adoptent comme H_0 la stationnarité (KPSS)

13.3.1 Test de Dickey-Fuller simple

Il concerne les séries autorégressives d'ordre 1.

13.3.1.1 La série ne montre aucun terme déterministe

Elle ne peut donc être ni une marche aléatoire avec dérive, ni une série avec un trend déterministe. La régression considérée, modèle [1], est

$$(1-\phi_1 B)X_t = \varepsilon_t \quad \Longleftrightarrow \quad X_t = \phi_1 X_{t-1} + \varepsilon_t \tag{13.24}$$

On teste $H_0 : \phi_1 = 1$, c'est-à-dire la série est $I(1)$ sans dérive contre $H_1 : |\phi_1| < 1$ et la série est $I(0)$ avec une moyenne éventuellement non nulle. Cette approche convient aux séries financières sans tendance, comme les taux d'intérêt, les taux de change. Sous l'hypothèse nulle, le modèle s'écrit alors

$$(I-B)X_t = \varepsilon_t \quad \Longleftrightarrow \quad X_t = X_{t-1} + \varepsilon_t \quad \Longleftrightarrow \quad X_t = X_0 + \sum_{i=1}^{t} \varepsilon_i \tag{13.25}$$

Il s'agit d'un modèle DS de « marche aléatoire ». Si H_0 est acceptée, alors le processus stationnarisé est

$$\Delta X_t = \varepsilon_t \tag{13.26}$$

Si l'hypothèse alternative $H_1 : |\phi_1| < 1$ est acceptée, alors X_t est un $AR(1)$ stationnaire. La statistique de test est $T(\widehat{\phi}_1 - 1)$ où $\widehat{\phi}_1$ est l'estimateur des moindres carrés ordinaires de ϕ_1. Sous l'hypothèse nulle, $\phi_1 = 1$, cette statistique **ne suit pas** une loi standard. On peut également utiliser la statistique de Student habituelle $(\widehat{\phi}_1 - 1)/\widehat{\sigma}_{\widehat{\phi}_1}$. Bien entendu,, elle ne suit pas une loi de Student sous l'hypothèse nulle.

13.3.1.2 La série ne montre pas de tendance

Celle - ci peut apparaître si la série est une marche aléatoire avec dérive. La régression considérée, modèle [2], est :

$$(1-\phi_1 B)(X_t - \mu) = \varepsilon_t \quad \Longleftrightarrow \quad X_t = c + \phi_1 X_{t-1} + \varepsilon_t \tag{13.27}$$

avec $c = \mu(1-\phi_1)$. Sous $H_0 : \phi_1 = 1$, le modèle s'écrit (avec $c = 0$ par conséquent) :

$$X_t = X_{t-1} + \varepsilon_t \tag{13.28}$$

C'est un modèle DS de marche aléatoire. Comme le cas précédent, on le rend stationnaire par la transformation :

$$\Delta X_t = \varepsilon_t \tag{13.29}$$

Sous l'hypothèse alternative : $H_0 : |\phi_1| < 1$, on a un $AR(1)$ stationnaire avec constante.

13.3.1.3 La série montre une tendance

Ce cas apparaît si la série est stationnaire à un trend déterministe. La régression considérée, modèle [3], est

$$(1-\phi_1 B)(X_t - \alpha - \beta t) = \varepsilon_t \quad \Longleftrightarrow \quad X_t = \phi_1 X_{t-1} + bt + c + \varepsilon_t \tag{13.30}$$

où $c = \alpha(1-\phi_1) + \phi_1\beta$ et $b = \beta(1-\phi_1)$. c et b sont destinés à capter l'éventuelle tendance déterministe si $|\phi_1| < 1$. Cette approche convient aux séries ayant une tendance, comme le cours d'un titre, le niveau d'un agrégat macroéconomique. Sous $H_0 : \phi_1 = 1$ (alors $b = 0$ et $c = \beta$), le modèle s'écrit :

$$X_t = X_{t-1} + \beta + \varepsilon_t \tag{13.31}$$

Il s'agit d'une marche aléatoire avec dérive. La solution est de la forme :

$$X_t = X_0 + \beta t + \sum_{i=1}^{i=t} \varepsilon_i \tag{13.32}$$

Il est non stationnaire, déterministe et aléatoire à la fois. On le rend stationnaire par le filtre aux différences premières :

$$\Delta X_t = \beta + \varepsilon_t \tag{13.33}$$

Avec l'hypothèse alternative : $H_1 : |\phi_1| < 1$, le modèle peut s'écrire, en posant $Y_t = X_t - \alpha - \beta t$, $(I - \phi_1 B)Y_t = \varepsilon_t$.

La statistique de test est la statistique de Fisher F, pour tester H_0, calculée comme dans la méthode des moindres carrés ordinaires, mais sous l'hypothèse nulle. Cette statistique ne suit pas une loi de Fisher.

Considérons le modèle $(1 - \phi_1 B)Y_t = \varepsilon_t$. Comme $Y_t = \dfrac{1}{1 - \phi_1 B}\varepsilon_t$ est un processus $ARMA$ $(MA(\infty))$, le processus X_t est un processus TS avec erreur $ARMA$; on peut le rendre stationnaire en calculant les écarts par rapport à la tendance estimée par les moindres carrés ordinaires.

Si la valeur calculée de la t-statistique associée à ϕ_1 est supérieure à la valeur critique, on accepte l'hypothèse nulle de non stationnarité. Il est fondamental de noter que l'on n'effectue pas le test sur les trois modèles. Il convient en effet d'appliquer le test de Dickey - Fuller sur un seul des trois modèles [1].

Ces tests révèlent l'existence d'une racine unitaire mais restent insuffisants pour discriminer entre les processus TS et DS surtout pour les modèle [2] et [3] qui font intervenir d'autres variables explicatives que X_{t-1}. Ils sont, par conséquent, complétés par les tests d'hypothèses jointes. Ces tests d'hypothèses jointes concernent les modèles [2] et [3]. Ils permettent de détecter une racine unitaire et de distinguer les processus DS des processus linéaires.

Pour le modèle [2], on effectue le test de recherche d'une racine unitaire, puis le test d'hypothèse jointe noté $H_0^1 : (c, \phi_1) = (0, 1)$ contre l'hypothèse H_1^1. Pour effectuer ce test, on calcule la statistique empirique F_1 :

$$F_1 = \frac{(SCR_c - SCR_2)/2}{SCR_2/(T-2)} \tag{13.34}$$

avec SCR_c la somme des carrés des résidus du modèle [2] contraint sous l'hypothèse H_0^1, soit $SCR_c = \sum_t (X_t - X_{t-1})^2$ et SCR_2 la somme des carrés des résidus du modèle [2] non contraint estimé par les moindres carrés ordinaires. T est le nombre d'observations utilisées pour estimer les paramètres du modèle.

Cette statistique de F_1 est analogue à une loi de Fisher. Dickey et Fuller ont tabulé les valeurs critiques de sa distribution empirique et asymptotique. Si F_1 est supérieur à la valeur Φ_1 lue dans la table à un seuil α, on rejette l'hypothèse H_0^1 à un seuil α.

Pour le modèle [3], après le test de racine unitaire, on effectue les tests des deux hypothèses jointes suivantes :

$$H_0^2 : (c, b, \phi_1) = (0, 0, 1) \quad \text{contre l'hypothèse l'un des paramètres est différent} \tag{13.35}$$

$$H_0^3 : (c, b, \phi_1) = (c; 0, 1) \quad \text{contre l'hypothèse l'un des paramètres est différent} \tag{13.36}$$

1. Voir la stratégie de test dans Vallery Mignon et Sandrine Lardic.

Ainsi, pour ces hypothèses, on calcule les statistiques F_2 et F_3 associées respectivement aux hypothèses H_0^2 et H_0^3 définies par :

$$F_2 = \frac{(SCR_c - SCR_3)/3}{SCR_3/(T-3)} \tag{13.37}$$

$$F_3 = \frac{(SCR_c^3 - SCR_3)/2}{SCR_3/(T-3)} \tag{13.38}$$

avec $SCR_c^3 = \sum_t (X_t - X_{t-1} - \hat{c})^2$ la somme des carrés des résidus du modèle [3] contraint sous l'hypothèse H_0^3 et SCR_3 la somme des carrés des résidus du modèle [3] non contraint estimé par les MCO.

Ces statistiques F_2 et F_3 sont analogues à une loi de Fisher, on se réfère donc aux tables de Dickey - Fuller. Si F_2 est supérieur à la valeur lue dans la table Φ_2, on rejette l'hypothèse H_0^2 à un seuil α. Si F_3 est supérieur à la valeur lue dans la table Φ_3, on rejette l'hypothèse H_0^2 à un seuil α.

Exemple 13.6

On demande d'appliquer la stratégie de test DF en produits pharmaceutiques en France connues mensuellement sur 14 ans. Le graphique (13.4) illustre l'évolution de ces produits.

```
# Chargement des données
import pandas as pd
df = pd.read_excel("./donnee/c5ex2.xls", header=0, index_col=0)
df.index = pd.date_range(start="2000-01-31", periods=len(df),freq = "M")
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
df.TOTPHARSA.plot(ax=axe,color='black');
axe.set_xlabel('Date');
axe.set_ylabel('value');
plt.show()
```

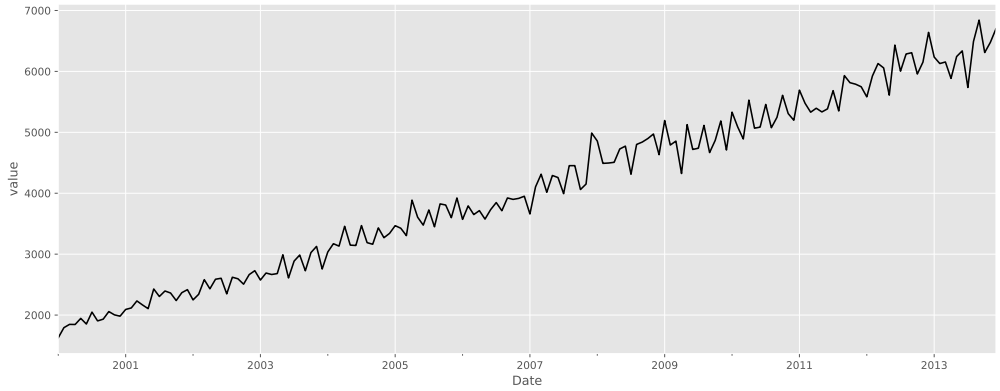


Figure 13.4 – Évolution sur 14 ans des dépenses CVS en produits pharmaceutiques en France

Nous appliquons la stratégie de test de Dickey-Fuller. L'estimation du modèle [3] : $TOTPHARSA_t = \phi_1 TOTPHARSA_{t-1} + bt + c + \varepsilon_t$ conduit aux résultats suivants :

```
# Dickey - Fuller : Modèle [3]
import statsmodels.api as sm
reg1 = sm.tsa.SARIMAX(df.TOTPHARSA,trend='ct',order=(1,0,0)).fit()
reg1_params = extractParams(reg1, model_type = "arima")
```

Table 13.1 – Coefficients du modèle [3] du test de Dickey et Fuller

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	1847.3507	132.1782	13.9762	2.178e-44	1588.2863	2106.4152
drift	30.8420	2.1960	14.0448	8.287e-45	26.5380	35.1460
ar.L1	-0.0844	0.0755	-1.1168	2.641e-01	-0.2324	0.0637
sigma2	37818.9548	3714.6375	10.1811	2.409e-24	30538.3991	45099.5105

Sous H_0 de racine unitaire, la statistique du t empirique vaut $t = \dfrac{\tilde{\phi}_1 - 1}{\hat{\sigma}_{\tilde{\phi}_1}} = \dfrac{-0.0844 - 1}{0.076} = -13.87$. Cette statistique est inférieure à la valeur lue dans la table à $1\%(= -3.99)$. On rejette donc H_0. L'hypothèse de racine unitaire est, bien entendu, rejetée.

Remarque 13.5

La fonction ADF du package `arch` effectue le test de Dickey - Fuller simple basé sur les modèles suivants (en fixant le paramètre `lags = 0`) :

— Modèle [1] : modèle sans constante ni tendance déterministe

$$\Delta X_t = \rho X_{t-1} + \varepsilon_t \tag{13.39}$$

— Modèle [2] : modèle avec constante sans tendance déterministe

$$\Delta X_t = \rho X_{t-1} + c + \varepsilon_t \tag{13.40}$$

— Modèle [3] : Modèle avec constante et tendance déterministe

$$\Delta X_t = \rho X_{t-1} + c + bt + \varepsilon_t \tag{13.41}$$

avec $\rho = \phi_1 - 1$.

Exemple 13.7

Appliquons la fonction ADF à l'exemple précédent sur les dépenses CVS en produits pharmaceutiques.

```
# Dickey - Fuller : Modèle [3]
from arch.unitroot import ADF
res_df = ADF(df.TOTPHARSA,lags=0,trend = 'ct').regression
res_df_params = extractParams(res_df, model_type = "lm")
```

On obtient la statistique empirique $t_{\hat{\rho}} = \dfrac{-1.0843}{0.078} = -13.876$, identique à celle calculée précédemment. On rejette l'hypothèse H_0.

Remarque 13.6

Table 13.2 – Coefficients du modèle [3] du test de Dickey et Fuller avec le package `arch`

	coef	std err	t	P>|t|	[0.025	0.975]
Level.L1	-1.0843	0.0781	-13.8764	1.855e-29	-1.2386	-0.9300
const	1847.3507	134.6594	13.7187	5.104e-29	1581.4612	2113.2403
trend	30.8420	2.2423	13.7547	4.050e-29	26.4145	35.2694

Table 13.3 – Valeurs critiques du modèle [3]

	statistic	1pct	5pct	10pct
tau3	-13.8764	-3.99	-3.43	-3.13
phi2	65.5211	6.22	4.75	4.07
phi3	96.2834	8.43	6.49	5.47

Pour réaliser un test de Dickey et Fuller, le résultat n'est pas identique selon l'utilisation de l'un des trois modèles comme processus générateur de la chronique de départ. Les conclusions auxquelles on parvient sont donc différentes et peuvent entraîner des transformations erronées. C'est la raison pour laquelle Dickey et Fuller et à leur suite d'autres auteurs, ont élaboré des stratégies de tests. Celles-ci sont nombreuses mais ont été néanmoins synthétisées et présentées de façon exhaustive par Ertur (1992). Nous proposons (figure (**??**)) une stratégie de test qui résume nos propos et présente l'avantage de la simplicité.

13.3.2 Test de Dickey - Fuller Augmenté

Dans les modèles précédents, utilisés pour les tests de Dickey - Fuller simples, le processus est, par hypothèse, un bruit blanc. Or il n'y a aucune raison pour que, a priori, l'erreur soit non corrélée ; on appelle tests de Dickey - Fuller Augmentés la prise en compte de cette hypothèse. Tout comme dans le cas du test de Dickey - Fuller simple, trois modèles servent de base à la construction du test de Dickey - Fuller augmenté :

— Modèle [4] : Modèle sans constante, ni tendance déterministe

$$\Delta X_t = \phi_1 X_{t-1} + \sum_{j=1}^{j=p} \gamma_j \Delta X_{t-j} + \varepsilon_t \tag{13.42}$$

— Modèle [5] : modèle avec constante sans tendance déterministe :

$$\Delta X_t = \phi_1 X_{t-1} + \sum_{j=1}^{j=p} \gamma_j \Delta X_{t-j} + c + \varepsilon_t \tag{13.43}$$

— Modèle [6] : modèle avec constante et tendance déterministe :

$$\Delta X_t = \phi_1 X_{t-1} + \sum_{j=1}^{j=p} \gamma_j \Delta X_{t-j} + c + bt + \varepsilon_t \tag{13.44}$$

avec $\varepsilon_t \sim i.i.d.(0, \sigma_\varepsilon^2)$.

La mise en oeuvre du test ADF est similaire à celle du test de DF simple : on adopte la même stratégie séquentielle descendante partant de l'estimation du modèle [6]. Les statistiques de test sont les mêmes que dans le cas du test de DF simple. Il convient cependant de remarquer que l'application du test ADF nécessite au préalable de choisir le nombre de retards p à introduire de sorte à blanchir les résidus (Voir Perron (1991) et Campbell and Perron (1991)).

Après avoir recherché l'existence d'une racine unitaire, on effectue les tests d'hypothèses jointes. Dans le modèle [5], on effectue le test de l'hypothèse jointe noté[2] $H_0^4 : (c, \phi_1) = (0, 0)$ contre l'hypothèse alternative H_1^4 au moins un des éléments est différent. Pour effectuer ce test, on calcule la statistique empirique F_4 :

$$F_4 = \frac{(SCR_c - SCR_5)/2}{SCR_5/(T-p-2)} \tag{13.45}$$

où $SCR_c = \sum_t \left(\Delta X_t - \sum_{j=1}^{p} \hat{\gamma}_j \Delta X_{t-j} \right)^2$ la somme des carrés des résidus du modèle [5] contraint par H_0^4 et SCR_5 la somme des carrés des résidus du modèle [5] estimé par MCO.

Cette statistique F_4 est analogue à une loi de Fisher, elle a été tabulée par Dickey and Fuller (1981). Si F_4 est supérieur à la valeur lue dans la table $\Phi^{\alpha}_{2,T-p-1}$, on rejette l'hypothèse H_0^4 à un seuil α.

Dans le modèle [6], on effectue les tests d'hypothèses nulles suivantes :

$$H_0^5 : (c, b, \phi_1) = (0, 0, 0) \tag{13.46}$$
$$H_0^6 : (c, b, \phi_1) = (c, 0, 0) \tag{13.47}$$

et à ces deux hypothèses, on calcule les statistiques F_5 et F_6 :

$$F_5 = \frac{(SCR_c - SCR_6)/3}{SCR_6/(T-p-3)} \tag{13.48}$$
$$F_6 = \frac{(SCR_D - SCR_6)/2}{SCR_6/(T-p-3)} \tag{13.49}$$

avec SCR_c (respectivement $SCR_D = \sum_t \left(\Delta X_t - \sum_{j=1}^{p} \hat{\gamma}_j \Delta X_{t-j} - \hat{c} \right)^2$) la somme des carrés des résidus du modèle [6] contraint par H_0^5 (respectivement par H_0^6).

Si F_5 (respectivement F_6) est supérieur à la valeur lue dans la table $\Phi^{\alpha}_{2,T-p-2}$ (respectivement $\Phi^{\alpha}_{3,T-p-2}$), on rejette l'hypothèse H_0^5 (respectivement H_0^6) à un seuil α.

Exemple 13.8

Soit l'estimation du modèle [6] en fixant le nombre de retard $p = 2$: $\Delta TOTPHARSA_t = \phi_1 TOTPHARSA_{t-1} + \sum_{j=1}^{2} \Phi_j \Delta TOTPHARSA_{t-j} + c + bt + \varepsilon_t$.

```
# Augmented Dickey - Fuller
res_adf = ADF(df.TOTPHARSA,lags=2, trend = 'ct').regression
res_adf_params = extractParams(res_adf, model_type = "lm")
```

La statistique du t empirique $t = \frac{-0.7529}{0.139} = -5.410$ est inférieure à la valeur lue dans la table à 1%($= -3.99$), on rejette donc H_0. L'hypothèse de racine unitaire est rejetée.

2. Cette hypothèse jointe correspond à celle H_0^1.

Table 13.4 – Coefficients du modèle [6] du test de ADF

	coef	std err	t	P>|t|	[0.025	0.975]
Level.L1	-0.7529	0.1392	-5.4095	2.266e-07	-1.0277	-0.4780
Diff.L1	-0.3155	0.1101	-2.8651	4.729e-03	-0.5330	-0.0980
Diff.L2	-0.3480	0.0745	-4.6731	6.258e-06	-0.4951	-0.2009
const	1349.4658	239.4711	5.6352	7.707e-08	876.5339	1822.3977
trend	21.4537	3.9645	5.4115	2.245e-07	13.6243	29.2831

Table 13.5 – Valeurs critiques du modèle [3]

	statistic	1pct	5pct	10pct
tau3	-5.4095	-3.99	-3.43	-3.13
phi2	17.1893	6.22	4.75	4.07
phi3	14.6588	8.43	6.49	5.47

Nous procédons maintenant au test d'hypothèses jointes conformément à la stratégie de test. On calcule la statistique de Fisher F_6 :

$$F_6 = \frac{(SCR_D - SCR_6)/2}{SCR_6/(T-p-3)} \tag{13.50}$$

où SCR_D est la somme des carrés des résidus du modèle estimé sous $H_0^6 : (c; b; \phi_1) = (c; 0; 0)$, c'est-à-dire $\Delta TOTPHARSA_t = \sum_{j=1}^{2} \Phi_j \Delta TOTPHARSA_{t-j} + c + \varepsilon_t$, soit :

```
# Augmented Dickey - Fuller # Estimation d'un AR(2)
from statsmodels.tsa.arima.model import ARIMA
model_ar = ARIMA(df.TOTPHARSA.diff(periods=1).dropna(),order=(2,0,0)).fit()
model_ar_params = extractParams(model_ar, model_type = "arima")
```

Table 13.6 – Coefficients du modèle [6] du test de Dickey - Fuller augmenté sous H_0^6

	coef	std err	t	P>|t|	[0.025	0.975]
const	28.9104	6.6431	4.3519	1.349e-05	15.8901	41.9308
ar.L1	-0.8154	0.0521	-15.6436	3.673e-55	-0.9176	-0.7133
ar.L2	-0.5908	0.0551	-10.7256	7.717e-27	-0.6988	-0.4829
sigma2	39491.6303	3466.7745	11.3915	4.612e-30	32696.8771	46286.3835

```
# Sum of residual squared model
SCRD = np.square(model_ar.resid).sum()
print('Sum of residual squared (Modele diff) : %.2f'%(SCRD))
```

```
## Sum of residual squared (Modele diff) : 6564871.56
```

et SCR_6 la somme des carrés des résidus du modèle [6] estimé par MCO :

```
# Sum of residual squared model [6]
SCR6 = np.square(res_adf.resid).sum()
print('Sum of residual squared (modèle [6]) : %.2f'%(SCR6))
```

```
## Sum of residual squared (modèle [6]) : 5524702.26
```

On calcule la statistique F_6 :

```
# Statistique de Fisher
ftest = (res_adf.nobs - 2 - 3)*(SCRD - SCR6)/(2*SCR6)
print('f-statistic : %.4f' %(ftest))
```

```
## f-statistic : 15.0621
```

Cette statistique est supérieure à la valeur leu dans la table 1%(= 8.43), on rejette donc $H_0^6 = (c; b; \phi_1) = (c; 0; 0)$.

Les tests DF et DFA de racine unitaire, nous font rejeter l'hypothèse H_0, les coefficients de la tendance et du terme constant sont significativement différents de 0. Le processus est de type TS (tendance déterministe). La bonne méthode de stationnarisation consiste à calculer le résidu issu de la régression : TOTPHARSA $= \hat{a}_0 + \hat{a}_1 t + \hat{\varepsilon}_t$.

Un « choc » sur les dépenses pharmaceutiques (déremboursement, promotion des génériques, ...) a donc un impact instantané et n'affecte pas la pente de la tendance de long terme.

13.3.3 Test de Phillips - Perron

Phillips and Perron (1988) ont développé une approche alternative de traitement de l'éventuelle autocorrélation du terme d'erreur. Ils proposent une statistique de test (Z-test) permettant de corriger de façon non paramétrique (sans spécification paramétrique du processus des erreurs) l'autocorrélation des résidus.

Le test de Phillips and Perron (1988) est construit sur une correction non paramétrique des statistiques de Dickey-Fuller pour prendre en compte des erreurs hétéroscédastiques et/ou autocorrélées. Il se déroule en quatre étapes :

1. Estimation par les moindres carrés ordinaires des trois modèles de base des tests de Dickey - Fuller et calcul des statistiques associées, soit $\hat{\varepsilon}_t$ le résidu estimé ;
2. Estimation de la variance dite de court terme des résidus

$$\hat{\sigma}_\varepsilon^2 = \frac{1}{T}\sum_{t=1}^{t=T}\hat{\varepsilon}_t^2 \tag{13.51}$$

3. Estimation d'un facteur correctif s_t^2 (appelé variance de long terme) établi à partir de la structure des covariances des résidus des modèles précédemment estimés. on a :

$$s_t^2 = \frac{1}{T}\sum_{t=1}^{t=T}\hat{\varepsilon}_t^2 + 2\sum_{i=1}^{i=h}\left(1 - \frac{i}{h+1}\right)\frac{1}{T}\sum_{t=i+1}^{t=T}\hat{\varepsilon}_t\hat{\varepsilon}_{t-i} \tag{13.52}$$

   Pour estimer cette variance de long terme, il est nécessaire de définir un nombre de retards h (troncature de Newey - West) estimé en fonction du nombre d'observations T,

$$h \approx 4\left(\frac{T}{100}\right)^{2/9} \tag{13.53}$$

4. Calcul de la statistique de PP :

$$t^*_{\hat{\phi}_1} = \sqrt{k} \times \frac{\left(\hat{\phi}_1 - 1\right)}{\hat{\sigma}_{\hat{\phi}_1}} + \frac{T(k-1)\hat{\sigma}_{\hat{\phi}_1}}{\sqrt{k}} \tag{13.54}$$

avec $k = \frac{\hat{\sigma}^2_\varepsilon}{s^2_t}$ (qui est égal à 1 - de manière asymptotique - si $\hat{\varepsilon}_t$ est un bruit blanc). Cette statistique est à comparer aux valeurs critiques de la table de MacKinnon.

Sous Python, le calcul du test de Phillips et Perron s'effectue grâce à la fonction PhillipsPerron eu package arch.

```
# Test de Phillips-Perron
from arch.unitroot import PhillipsPerron
pp_model = PhillipsPerron(default_data)
pp_params = extractParams(pp_model,model_type = "lm")
```

13.3.4 Test KPSS

Kwiatkowski et al. (1992) proposent un test fondé sur l'hypothèse nulle de stationnarité. Après estimation des modèles [2] ou [3], on calcule la somme partielle des résidus :

$$S_t = \sum_{i=1}^{i=t} \hat{\varepsilon}_i \tag{13.55}$$

et on estime la variance de long terme (s^2_t) comme pour le test de Phillips et Perron. La statistique est alors :

$$LM = \frac{1}{s^2_t} \frac{1}{T^2} \sum_{t=1}^{T} S^2_t \tag{13.56}$$

On rejette l'hypothèse de stationnarité si cette statistique est supérieure aux valeurs critiques suivantes :

$n = \infty$	1%	5%	10%
Modèle [2]	0.739	0.463	0.347
Modèle [3]	0.216	0.146	0.119

Sous Python, le calcul du test de KPSS s'effectue à l'aide de la fonction KPSS du package arch.

```
# Test KPSS
from arch.unitroot import KPSS
kpss_model = KPSS(default_data)
kpss_params = extractParams(kpss_model, model_type = "lm")
```

13.3.5 Test Zivot-Andrews

Zivot and Andrews (2002) proposent un test de racine unitaire avec une rupture endogène.L'hypothèse nulle suppose que la série présente une racine unitaire mais sans aucune rupture.L'hypothèse alternative suppose que la série est stationnaire avec une seule rupture à date inconnue. La date de rupture minimise la $t-$statistique de ϕ_1 des équations (13.58)

(13.59)(13.60) ci-dessous. En d'autres termes, la rupture choisie est celle qui peut être la moins favorable à l'hypothèse nulle.

Sous l'hypothèse nulle de racine unitaire, on a le modèle suivant :

$$X_t = c + X_{t-1} + \varepsilon_t \tag{13.57}$$

Ils proposent, sous l'hypothèse alternative, les trois modèles ci-dessous :

— Modèle A_{za}

$$\Delta X_t = \phi_1 X_{t-1} + \sum_{j=1}^{p} \gamma_j \Delta X_{t-j} + c + bt + \rho DU_t + \varepsilon_t \tag{13.58}$$

— Modèle B_{za}

$$\Delta X_t = \phi_1 X_{t-1} + \sum_{j=1}^{p} \gamma_j \Delta X_{t-j} + c + bt + \theta DT_t + \varepsilon_t \tag{13.59}$$

— Modèle C_{za}

$$\Delta X_t = \phi_1 X_{t-1} + \sum_{j=1}^{p} \gamma_j \Delta X_{t-j} + c + bt + \rho DU_t + \theta DT_t \varepsilon_t \tag{13.60}$$

Le modèle A_{za} autorise un changement dans la constante. Le modèle B_{za} teste la stationnarité de la série autour d'une tendance cassée. Finalement, le modèle C_{za} permet un changement dans la constante et dans la tendance. DU_t est une variable indicatrice qui capte le changement dans la constante à la date Tb. On a :

$$DU_t = \begin{cases} 1 & \text{si } t > Tb \\ 0 & \text{sinon} \end{cases} \tag{13.61}$$

DT_t est une autre variable indicatrice qui représente la rupture dans la tendance à la date Tb. On a :

$$DT_t = \begin{cases} t - Tb & \text{si } t > Tb \\ 0 & \text{sinon} \end{cases} \tag{13.62}$$

Zivot and Andrews (2002) supposent que la date de rupture $Tb \in [T_1; T_2]$, où $T_1 = \lambda \times T$ et $T_2 = (1 - \lambda) \times T$ avec $\lambda \in [0, 1]$.

Sous Python, le calcul du test de Zivot and Andrews (2002) s'effectue à l'aide de la fonction ZivotAndrews du package arch.

```
# Test Zivot-Andrew
from arch.unitroot import ZivotAndrews
za_model = ZivotAndrews(default_data)
za_params = extractParams(za_model, model_type = "lm")
```

Exemple 13.9 *Unit root test with CAC40 dataset*

On demande d'appliquer la stratégie des tests DF, ADF, PP et KPSS à l'indice CAC40 (indice représentatif de l'évolution des cours de bourse) sur 1160 observations quotidiennes. La figure (13.5) illustre l'évolution de l'indice CAC40 et nous suggère un processus DS.

```
# Chargement du dataset
cac40 =  pd.read_excel("./donnee/c5ex3.xls",index_col=0,header=0)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6));
axe.plot(cac40, color='black');
axe.set_xlabel('t');
axe.set_ylabel('value');
plt.show()
```

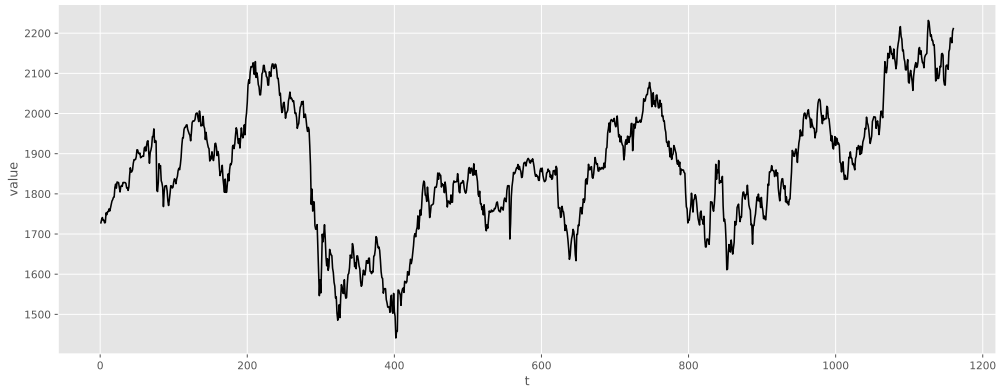


Figure 13.5 – Évolution de l'indice boursier CAC40

— **Test de Dickey - Fuller simple**

La première étape consiste à l'estimation de modèle [3] : $\Delta CAC_t = \phi_1 CAC_{t-1} + c + bt + \varepsilon_t$.

```
# Dickey - Fuller : Modèle [3]
reg_df1 = ADF(cac40,lags=0,trend = 'ct').regression
reg_df1_params = extractParams(reg_df1, model_type = "lm")
```

Table 13.7 – Coefficients du modèle [3] du test ADF sur le CAC40

	coef	std err	t	P>|t|	[0.025	0.975]
Level.L1	-0.0084	0.0041	-2.0538	4.022e-02	-0.0165	-0.0004
const	14.9391	7.4376	2.0086	4.481e-02	0.3464	29.5319
trend	0.0021	0.0019	1.1089	2.677e-01	-0.0016	0.0058

Table 13.8 – Valeurs critiques du modèle [3]

	statistic	1pct	5pct	10pct
tau3	-2.0538	-3.96	-3.41	-3.12
phi2	1.6552	6.09	4.68	4.03
phi3	2.2456	8.27	6.25	5.34

La valeur empirique $t_{\widehat{\phi_1}} = -2.054 >$ aux trois valeurs critiques, on accepte l'hypothèse H_0.

Nous effectuons maintenant le test de l'hypothèse $H_0^3 : (c; b; \phi_1) = (c; 0; 1)$. On calcule la statistique F_3

$$F_3 = \frac{(SCR_c^3 - SCR_3)/2)}{SCR_3/(T-3)} \tag{13.63}$$

avec $SCR_c^3 = \sum_t (CAC_t - CAC_{t-1} - \hat{c})^2 = 493729.2$, somme des carrés des résidus du modèle [3] contraint sous l'hypothèse H_0^3, soit $\Delta CAC_t = c + \varepsilon_t$:

```
#Dickey - Fuller : Modèle [3] contraint sous hypothèse H_{0}^{3}
import statsmodels.formula.api as smf
reg_df2 = smf.ols('CAC~1',data = cac40.diff(periods=1).dropna()).fit()
reg_df2_params = extractParams(reg_df2, model_type = "lm")
```

Table 13.9 – Coefficients du modèle [3] du test DF sous H_0^3 sur le CAC40

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.4172	0.6065	0.6879	4.917e-01	-0.7728	1.6072

```
# Somme des carrés des résidus du modèle [3] contraint
SCR_c_3 = np.square(reg_df2.resid).sum()
print('Sum squared resid : %.1f'%(SCR_c_3))
```

```
## Sum squared resid : 493729.2
```

et $SCR_3 = 491818.4$ (somme des carrés des résidus du modèle [3]) :

```
# Somme des
SCR3 = np.square(reg_df1.resid).sum()
print('Sum squared resid : %.1f'%(SCR3))
```

```
## Sum squared resid : 491818.4
```

D'où : $F_3 = \frac{(493729.2 - 491818.4)/2}{491818.4/(1159-3)} = 2.24.$

Nous sommes amenés à accepter les hypothèses H_0 et H_0^3 ($F_3 = 2.24 <$) aux trois valeurs critiques pour Φ_3. Nous allons sur la branche de droite de la figure (**??**) et effectuons le test de l'hypothèse $H_0^2 : (c; b; \phi_1) = (0; 0; 1)$. On calcule la statistique F_2 :

$$F_2 = \frac{(SCR_c - SCR_3)/3}{SCR_3/(T-3)} \tag{13.64}$$

avec $SCR_c = \sum_t (CAC_t - CAC_{t-1})^2$, somme des carrés des résidus du modèle [2] contraint sous l'hypothèse H_0^2, soit $\sum_{t=1}^{t=1159} (CAC_t - CAC_{t-1})^2 = 493930.9$:

```
# Somme des carrés des résidus du modèle [2] constraint
SCR_c_2 = np.square(cac40.diff(periods=1).dropna()).sum()
print('Sum squared resid : %.1f'%(SCR_c_2))
```

```
## Sum squared resid : 493930.9
```

D'où : $F_2 = \dfrac{(493930.9 - 491818.4)/3}{491818.4/(1159-3)} = 1.65$. Nous sommes amenés à accepter l'hypothèse $H_0^2(F_2 = 1.65 <)$ aux trois valeurs critiques pour Φ_2. Nous estimons maintenant le modèle [2] : $\Delta CAC_t = \phi_1 CAC_{t-1} + c + \varepsilon_t$, et effectuons le test DF :

```
# Dickey - Fuller : Modèle [2]
reg_df3 = ADF(cac40,lags=0,trend = 'c').regression
reg_df3_params = extractParams(reg_df3, model_type = "lm")
```

Table 13.10 – Coefficients du modèle [2] du test DF sur le CAC40

	coef	std err	t	P>|t|	[0.025	0.975]
Level.L1	-0.0071	0.0039	-1.8058	7.120e-02	-0.0148	0.0006
const	13.6356	7.3449	1.8565	6.364e-02	-0.7751	28.0464

Table 13.11 – Valeurs critiques du modèle [2]

	statistic	1pct	5pct	10pct
tau2	-1.8058	-3.43	-2.86	-2.57
phi1	1.8676	6.43	4.59	3.78

La valeur empirique $t_{\hat{\phi}_1} = -1.806 >$ aux trois valeurs critiques, on accepte l'hypothèse H_0. Nous effectuons maintenant le test de l'hypothèse $H_0^1 : (c; \phi_1) = (0; 1)$, soit le calcul de la statistique F_1 :

$$F_1 = \frac{(SCR_c - SCR_2)/2}{SCR_2/(T-2)} \tag{13.65}$$

avec $SCR_2 = 492341.5$, la somme des carrés des résidus du modèle [2] non contraint estimé par les MCO, soit $\Delta CAC_t = \phi_1 CAC_{t-1} + \varepsilon_t$, et $SCR_c = 493930.9$, la somme des carrés des résidus du modèle [2] contraint sous l'hypothèse H_0^1.

```
# Somme des carrés des résidus sur le modèle [2]
SCR_2 = np.square(reg_df3.resid).sum()
print('Sum squared resid : %.1f'%(SCR_2))
```

```
## Sum squared resid : 492341.5
```

D'où : $F_1 = \dfrac{(493930.9 - 492341.5)/2}{492341.5/(1159-2)} = 1.87$. Nous sommes amenés à accepter l'hypothèse $H_0^1(F_1 = 1.87 <)$ aux trois valeurs critiques pour Φ_1.

En dernière étape, nous testons la nullité de la moyenne du CAC40 :

```
# Test de nullité de la moyenne
import scipy.stats as st
def tstatistic(x,mu=0):
    tstat=st.ttest_1samp(x,popmean=mu)
    # Affichage
    print('===='*9)
    print(f'Test of Hypothesis : Mean = {mu}')
    print('===='*9)
    print('Sample Mean = %.4f'%(np.mean(x)))
    print('Sample Std. Dev. = %.4f'%(np.std(x,ddof=1)))
    df = pd.DataFrame({'Value': tstat[0],'Probability':tstat[1]})
    df.index = ['t-statistic']
    return print(df)
# Application
tstatistic(cac40)
```

```
## ====================================
## Test of Hypothesis : Mean = 0
## ====================================
## Sample Mean = 1863.7602
## Sample Std. Dev. = 154.5946
##                    Value  Probability
## t-statistic  410.605364          0.0
```

La moyenne est - largement - significativement différente de 0. Le processus est une marche aléatoire sans dérive.

— **Test de Dickey - Fuller augmenté**

Nous appliquons la même stratégie dans le cadre du test de Dickey - Fuller Augmenté (DFA). Soit le modèle [6] à 5 variables décalées : $\Delta CAC_t = \phi_1 CAC_{t-1} + \sum_{j=1}^{j=4} \gamma_j \Delta CAC_{t-j} + c + bt + \varepsilon_t$.

```
# Agmented Dickey - Fuller : Modèle [6]
reg_adf1 = ADF(cac40,lags=4,trend = 'ct').regression
reg_adf1_params = extractParams(reg_adf1, model_type = "lm")
```

Table 13.12 – Coefficients du modèle [3] du test ADF sur le CAC40

	coef	std err	t	P>|t|	[0.025	0.975]
Level.L1	-0.0095	0.0042	-2.2783	2.289e-02	-0.0176	-0.0013
Diff.L1	0.0698	0.0295	2.3692	1.799e-02	0.0120	0.1277
Diff.L2	-0.0055	0.0295	-0.1875	8.513e-01	-0.0635	0.0524
Diff.L3	-0.0369	0.0295	-1.2502	2.115e-01	-0.0949	0.0210
Diff.L4	0.0693	0.0295	2.3481	1.904e-02	0.0114	0.1272
const	16.7796	7.5268	2.2293	2.599e-02	2.0118	31.5474
trend	0.0022	0.0019	1.1469	2.516e-01	-0.0015	0.0059

La valeur empirique $t_{\widehat{\phi_1}} = -2.278 >$ aux trois valeurs critiques, on accepte l'hypothèse H_0. Nous effectuons maintenant le test de l'hypothèse $H_0^6 : (c; b; \phi_1) = (c; 0; 0)$, soit le calcul de la statistique F_6 :

$$F_6 = \frac{(SCR_D - SCR_6)/2}{SCR_6/(T-p-3)} \tag{13.66}$$

Table 13.13 – Valeurs critiques du modèle [6]

	statistic	1pct	5pct	10pct
tau3	-2.2783	-3.96	-3.41	-3.12
phi2	1.9456	6.09	4.68	4.03
phi3	2.7208	8.27	6.25	5.34

avec $SCR_6 = 486574.6$, la somme des carrés des résidus du modèle [6] estimé par MCO :

```
# Sum squared resid
SCR_6 = np.square(reg_adf1.resid).sum()
print('Sum squared resid : %.1f'%(SCR_6))
```

```
## Sum squared resid : 486574.6
```

et $SCR_D = 488881$, la somme des carrés des résidus du modèle [6] contraint par H_0^6, soit : $\Delta CAC_t = \sum_{j=1}^{j=4} \gamma_j \Delta CAC_{t-j} + c + \varepsilon_t$, c'est-à-dire $SCR_D = \sum_t \left(\Delta CAC_t - \sum_{j=1}^{4} \hat{\gamma}_j \Delta CAC_{t-j} - \hat{c} \right)^2$.
On a :

```
# Fonction retard
def lag(x, n):
    if n == 0:
        return x
    if isinstance(x, pd.Series):
        return x.shift(n)
    else:
        x = pd.Series(x)
        return x.shift(n)
    x = x.copy()
    x[n:] = x[0:-n]
    x[:n] = np.nan
    return x
# Agmented Dickey - Fuller : Modèle [6] avec contrainte H_{0}^{6}
formul = "CAC~lag(CAC,1)+lag(CAC,2)+lag(CAC,3)+lag(CAC,4)"
reg_adf2 = smf.ols(formul,data = cac40.diff(periods=1).dropna()).fit()
reg_adf2_params = extractParams(reg_adf2, model_type = "lm")
```

Table 13.14 – Coefficients du modèle [6] du test ADF sous H_0^6 sur le CAC40

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.3810	0.6071	0.6276	5.304e-01	-0.8101	1.5722
lag(CAC, 1)	0.0647	0.0294	2.2003	2.799e-02	0.0070	0.1225
lag(CAC, 2)	-0.0106	0.0295	-0.3593	7.194e-01	-0.0685	0.0473
lag(CAC, 3)	-0.0419	0.0295	-1.4218	1.554e-01	-0.0998	0.0159
lag(CAC, 4)	0.0641	0.0295	2.1770	2.968e-02	0.0063	0.1219

```
# Sum squared resid
SCR_D = np.square(reg_adf2.resid).sum()
print('Sum squared resid : %.1f'%(SCR_D))
```

```
## Sum squared resid : 488881.0
```

D'où : $F_6 : \dfrac{(488881 - 486574.6)/2}{486574.6/(1155-4-3)} = 2.72 <$ aux trois valeurs critiques pour Φ_3. Nous sommes amenés à accepter les hypothèses H_0 et H_0^6.

Nous effectuons le test de l'hypothèse $H_0^5 : c(c; b; \phi_1) = (0; 0; 0)$, on calcule la statistique F_5 :

$$F_5 = \frac{(SCR_c - SCR_6)/3}{SCR_6/(T-p-3)} \tag{13.67}$$

avec $SCR_c = 489048.4$, la somme des carrés des résidus du modèle contraint, soit : $\Delta CAC_t = \sum_{j=1}^{4} \gamma_j \Delta CAC_{t-j} + \varepsilon_t$, c'est-à-dire $SCR_c = \sum_t \left(\Delta CAC_t - \sum_{j=1}^{4} \hat{\gamma}_j \Delta CAC_{t-j} \right)^2$. On a :

```
# Agmented Dickey - Fuller : Modèle [6] avec contrainte H_{0}^{5}
formul = "CAC~lag(CAC,1)+lag(CAC,2)+lag(CAC,3)+lag(CAC,4)-1"
reg_adf3 = smf.ols(formul,data = cac40.diff(periods=1).dropna()).fit()
reg_adf3_params = extractParams(reg_adf3, model_type = "lm")
```

Table 13.15 – Coefficients du modèle [6] du test ADF sous H_0^5 sur le CAC40

	coef	std err	t	P>|t|	[0.025	0.975]
lag(CAC, 1)	0.0651	0.0294	2.2136	2.705e-02	0.0074	0.1228
lag(CAC, 2)	-0.0103	0.0295	-0.3494	7.269e-01	-0.0681	0.0475
lag(CAC, 3)	-0.0416	0.0295	-1.4118	1.583e-01	-0.0995	0.0162
lag(CAC, 4)	0.0645	0.0294	2.1900	2.872e-02	0.0067	0.1222

```
# Sum squared resid
SCR_C = np.square(reg_adf3.resid).sum()
print('Sum squared resid : %.1f'%(SCR_C))
```

```
## Sum squared resid : 489048.4
```

D'où : $F_5 = \dfrac{(489048.4 - 486574.6)/2}{486574.6/(1155-4-3)} = 1.94 <$ aux trois valeurs critiques pour Φ_2, l'hypothèse H_0^5 est donc acceptée.

Nous estimons donc maintenant le modèle [5] : $\Delta CAC_t = \phi_1 CAC_{t-1} + \sum_{j=1}^{4} \gamma_j \Delta CAC_{t-j} + c + \varepsilon_t$

```
# Augmented Dickey - Fuller : Modèle [5]
reg_adf4 = ADF(cac40,lags=4,trend = 'c').regression
reg_adf4_params = extractParams(reg_adf4, model_type = "lm")
```

La valeur empirique $t_{\hat{\phi}_1} = -2.031 >$ aux trois valeurs critiques, on accepte l'hypothèse H_0. Nous effectuons maintenant le test de l'hypothèse de l'hypothèse $H_0^4 : (c; \phi_1) = (0; 0)$, soit le calcul de la statistique empirique F_4 :

$$F_4 = \frac{(SCR_c - SCR_5)/2}{SCR_5/(T-p-2)} \tag{13.68}$$

Table 13.16 – Coefficients du modèle [5] du test ADF sur le CAC40

	coef	std err	t	P>|t|	[0.025	0.975]
Level.L1	-0.0081	0.0040	-2.0310	4.248e-02	-0.0159	-0.0003
Diff.L1	0.0696	0.0295	2.3590	1.849e-02	0.0117	0.1274
Diff.L2	-0.0059	0.0295	-0.2003	8.413e-01	-0.0639	0.0520
Diff.L3	-0.0373	0.0295	-1.2621	2.072e-01	-0.0952	0.0207
Diff.L4	0.0689	0.0295	2.3361	1.966e-02	0.0110	0.1268
const	15.4326	7.4356	2.0755	3.816e-02	0.8437	30.0215

Table 13.17 – Valeurs critiques du modèle [5]

	statistic	1pct	5pct	10pct
tau2	-2.031	-3.43	-2.86	-2.57
phi1	2.260	6.43	4.59	3.78

avec $SCR_5 = 487132.1$, la somme des carrés des résidus du modèle [5] estimé par MC0, soit :

```
# Sum squared resid
SCR_5 = np.square(reg_adf4.resid).sum()
print('Sum squared resid : %.1f'%(SCR_5))
```

```
## Sum squared resid : 487132.1
```

et $SCR_c = 489048.4$, la somme des carrés des résidus du modèle [5] contraint par $H_0^4 : \Delta CAC_t = \sum_{j=1}^{4} \gamma_j \Delta CAC_{t-j} + \varepsilon_t$, c'est-à-dire $SCR_c = \sum_t \left(\Delta CAC_t - \sum_{j=1}^{4} \hat{\gamma}_j \Delta CAC_{t-j} \right)^2$.

D'où : $F_4 = \dfrac{(489048.4 - 487132.1)/2}{487132.1/(1155 - 4 - 2)} = 2.26 <$ aux trois valeurs critiques pour Φ_1. Nous acceptons l'hypothèse H_0^4.

Il est inutile de re-procéder au test de la moyenne, les conclusions sont donc les mêmes : le processus CAC40 est une marche aléatoire sans dérive.

— **Test de Phillips - Perron**

Nous procédons aux tests PP avec un nombre de retards $l = 6$ dans le cadre de trois modèles.

— Modèle [3] : Modèle avec tendance et constante

```
# PP test : Modèle [3]
from arch.unitroot import PhillipsPerron
reg_pp1 = PhillipsPerron(cac40, trend="ct",lags=6,test_type="tau")
# Extraction des résultats
def extractUnitRootParams(res):
  df1 = pd.DataFrame({"stat":res.stat,"p-value":res.pvalue}, index = ["test"])
  df2 = pd.DataFrame(res.critical_values,index = ["test"])
  return pd.concat([df1,df2],axis=1)
reg_pp1_res = extractUnitRootParams(reg_pp1)
```

— Modèle [2] : Modèle avec constante seule

Table 13.18 – Modèle [3] du test PP sur le CAC40

	stat	p-value	1%	5%	10%
test	-2.197413	0.4913066	-3.966602	-3.414285	-3.129284

```
# PP test : Modèle [2]
reg_pp2 = PhillipsPerron(cac40, trend="c",lags=6,test_type="tau")
reg_pp2_res = extractUnitRootParams(reg_pp2)
```

Table 13.19 – Modèle [2] du test PP sur le CAC40

	stat	p-value	1%	5%	10%
test	-1.948684	0.3095165	-3.436005	-2.864037	-2.568099

— Modèle [1] : Modèle sans constante

```
# PP test : Modèle [1]
reg_pp3 = PhillipsPerron(cac40, trend="n",lags=6,test_type="tau")
reg_pp3_res = extractUnitRootParams(reg_pp3)
```

Table 13.20 – Modèle [1] du test PP sur le CAC40

	stat	p-value	1%	5%	10%
test	0.5004788	0.8249891	-2.567672	-1.941234	-1.616593

Les probabilités critiques sont toutes supérieures à 0.05, nous ne rejetons pasl'hypothèse ; le processus CAC40 possède une racine unitaire.

— **Test KPSS**

Nous procédons aux tests KPSS avec un nombre de retards $l = 6$ dans le cadre de trois modèles.

— Modèle [3] : Modèle avec tendance et constante

```
# Test KPSS : Modèle [3]
from arch.unitroot import KPSS
reg_kpss3 = KPSS(cac40,trend="ct",lags=6)
reg_kpss3_res = extractUnitRootParams(reg_kpss3)
```

Table 13.21 – Modèle [3] du test KPSS sur le CAC40

	stat	p-value	1%	5%	10%
test	1.308105	1e-04	0.2175	0.1479	0.1193

— Modèle [2] : Modèle avec constante seule

```
# Test KPSS : Modèle [2]
reg_kpss2 = KPSS(cac40,trend="c",lags=6)
reg_kpss2_res = extractUnitRootParams(reg_kpss2)
```

La statistique est supérieure à la valeur critique (pour un seuil de 5%) pour les deux spécifications, nous rejetons l'hypothèse H_0, le processus CAC40 possède donc une racine unitaire.

Table 13.22 – Modèle [2] du test KPSS sur le CAC40

	stat	p-value	1%	5%	10%
test	2.581505	1e-04	0.7428	0.4614	0.3475

13.4 Prévision et application

13.4.1 Prévision d'un SARIMA

Considérons le processus X_t défini par un modèle $SARIMA(p,d,q) \times (P,D,Q)_S$. Alors le processus $Y_t = (I - B^S)^D (I - B)^d X_t$ suit un $ARMA(p + PS, q + QS)$ stationnaire.

Au chapitre précédent, nous avons effectué la prédiction d'un processus ARMA(p,q). Par conséquent, on sait donc faire de la prédiction pour Y. Soit $\widehat{Y}_t(h)$ son meilleur prédicteur linéaire, on peut reconstruire pas à pas le prédicteur $\widehat{X}_t(h)$ en fonction de $\widehat{Y}_t(h)$ et des valeurs observées. Ci-dessous quelques exemples pour comprendre la démarche :

1. $(X_t)_t \sim ARIMA(p,1,q)$ Alors le processus $Y_t = (I-B)X_t$ suit un $ARMA(p,q)$ donc on sait trouver son meilleur prédicteur linéaire. On reconstruit celui de X_t en utilisant que $Y_{t+1} = X_{t+1} - X_t$. Ainsi, par exemple,
$$\widehat{X}_t(1) = \widehat{Y}_t(1) + x_t \tag{13.69}$$
2. $(X_t) \sim SARIMA(p,0,q) \times (P,1,Q)_{12}$: $Y_t = (I - B^{12}) X_t$ donc $Y_{t+1} = X_{t+1} - X_{t-11}$, ainsi
$$\widehat{X}_t(1) = \widehat{Y}_t(1) + x_{t-11} \tag{13.70}$$
3. $(X_t)_t \sim SARIMA(p,1,q) \times (P,1,Q)_{24}$: $Y_t = (I - B^{24})(I - B)X_t$ donc $Y_{t+1} = X_{t+1} - X_t - X_{t-23} + X_{t-24}$ et on en déduit que
$$\widehat{X}_t(1) = \widehat{Y}_t(1) + x_t + x_{t-23} - x_{t-24} \tag{13.71}$$

13.4.2 Application : Air passengers

La série Air Passengers est une série chronologique « classique » présentant une tendance linéaire et un mouvement saisonnier de période 12. On désigne par X_t la série Air Passengers, et on considère $Y_t = \ln(X_t)$. On travaille en effet sur le logarithme de la série afin de pallier l'accroissement de la saisonnalité.

```
# Chargement du data set
import seaborn as sns
donnee = sns.load_dataset("flights").drop(columns = ["year","month"])
donnee.index = pd.date_range(start="1949-01-31",periods= len(donnee),freq="M")
```

13.4.2.1 Prétraitement de la série

On passe ainsi d'un modèle multiplicatif à un modèle modèle additif. La figure (13.6) montre la série log-transformée.

```
# Transformation en log
donnee['ln_passengers'] = np.log(donnee['passengers'])
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(donnee["ln_passengers"], color="black");
axe.set_xlabel('année');
axe.set_ylabel('valeur');
plt.show()
```

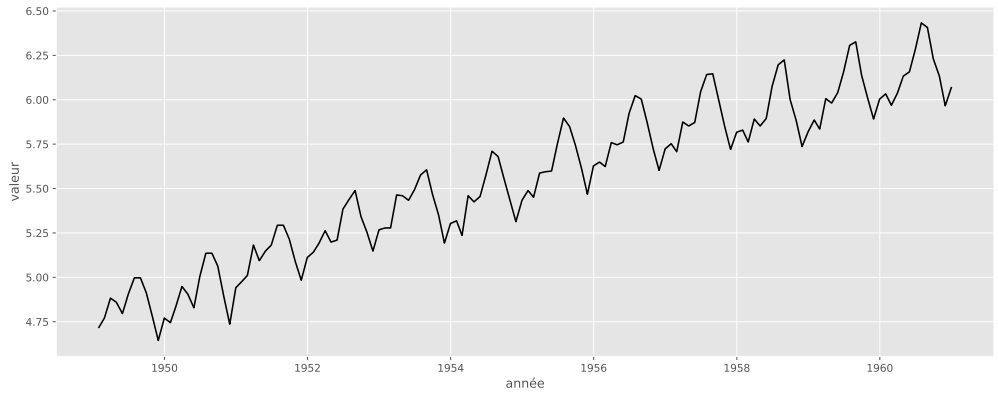


Figure 13.6 – Evolution du logarithme de la série Air passengers

On visualise l'autocorrélogramme simple.

```
# Extraction de la colonne logpassengers
y = donnee['ln_passengers']
s, nlag = 12, 36
# Autocorrélation simple
from statsmodels.graphics.tsaplots import plot_acf
fig, axe = plt.subplots(figsize=(16,6));
plot_acf(y,lags=nlag,ax=axe);
axe = plot_colors(axe);
axe.set_xlabel("Lag");
axe.set_ylabel("ACF");
plt.show()
```

La sortie ACF présente une décroissante lente vers 0, ce qui traduit un problème de non-stationnarité. On effectue donc une différentiation $(I - B)$.

```
# log Air passengers en différence première
y_diff_1 = donnee['ln_passengers'].diff(periods=1)
fig, axe = plt.subplots(figsize=(16,6));
axe.plot(y_diff_1, color="black");
axe.set_xlabel('t');
axe.set_ylabel('valeur');
plt.show()
```

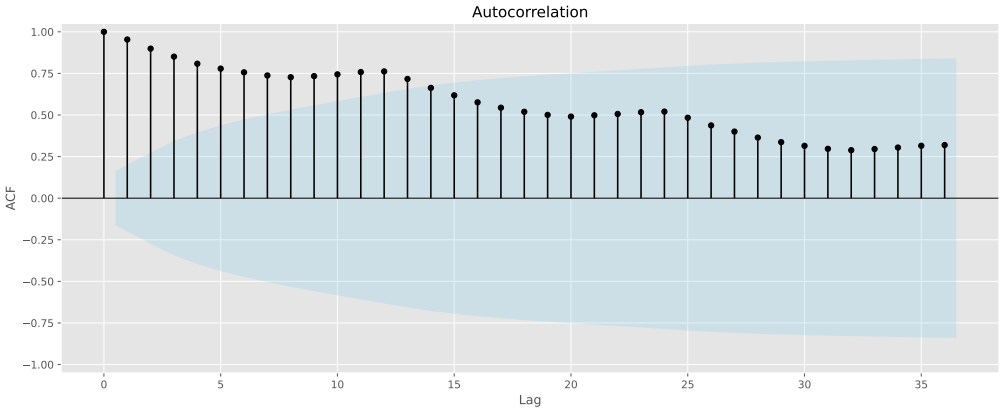


Figure 13.7 – Autocorrélation simple du logarithme de la série Air Passengers

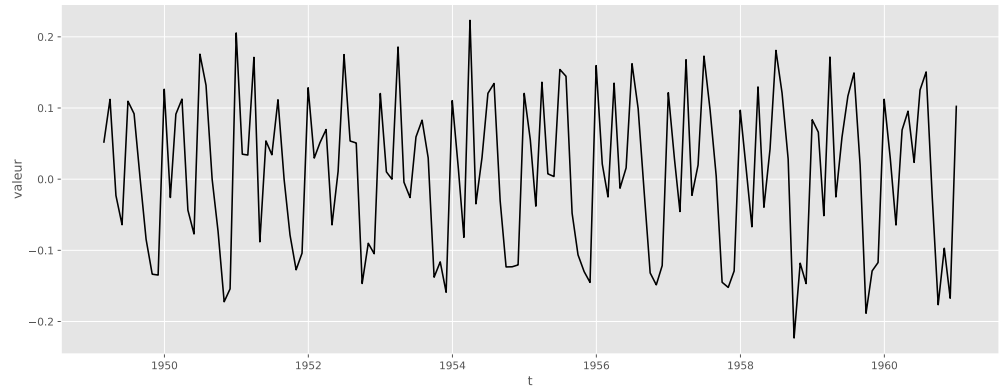


Figure 13.8 – Evolution de la série $(I-B)\ln(X_t)$

```
# ACF de (I-B)ln(Xt)
fig, axe = plt.subplots(figsize=(16,6))
plot_acf(y_diff_1.dropna(),lags=nlag,ax=axe);
axe = plot_colors(axe);
axe.set_xlabel("Lag");
axe.set_ylabel("ACF");
plt.show()
```

La sortie ACF de la série ainsi différenciée présente encore une décroissance lente vers 0 pour les multiples de 12. On effectue cette fois la différenciation $(I-B^{12})$.

```
# Séries (I-B)(I-B^12)ln(Xt)
y_diff_12_1 = y_diff_1 - y_diff_1.shift(12)
y_diff_12_1 = y_diff_12_1.dropna()
fig, axe = plt.subplots(figsize=(16,6));
axe.plot(y_diff_12_1, color= "black");
axe.set_xlabel('année');
axe.set_ylabel('valeur');
plt.show()
```

On représente le corrélogramme associé.

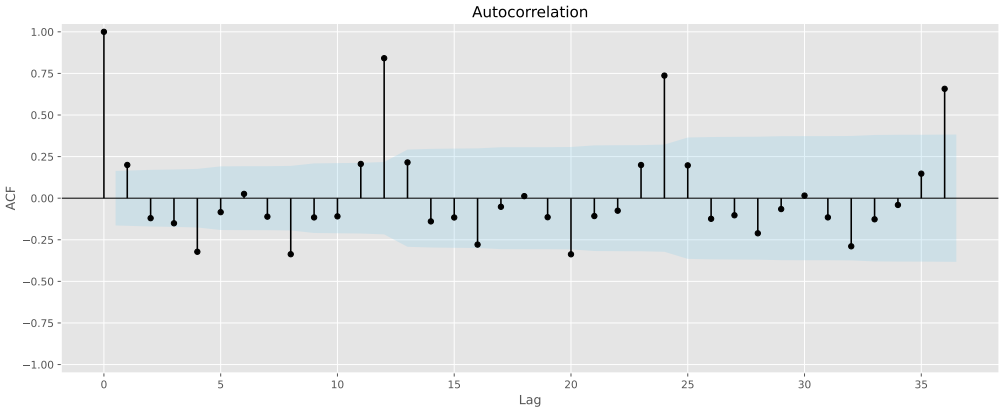


Figure 13.9 – Autocorrélation simple de la série $(I-B)\ln(X_t)$

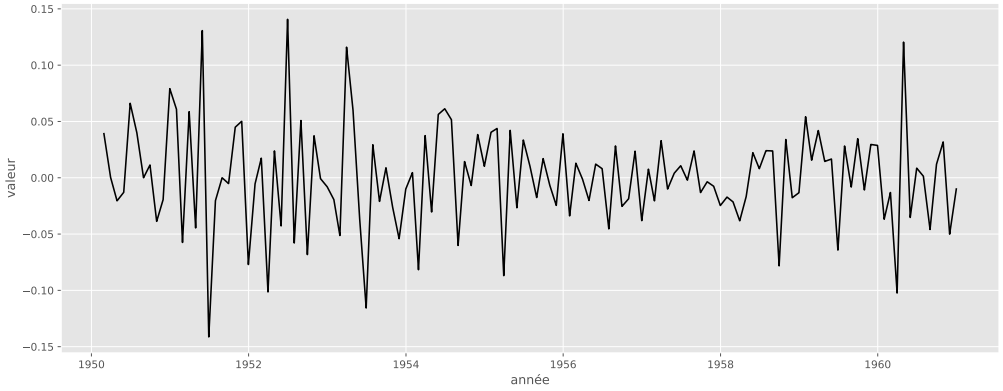


Figure 13.10 – Evolution de la série $(I-B)(I-B^{12})\ln(X_t)$

```
# ACF (I-B)(I-B^12)ln(Xt)
fig, axe = plt.subplots(figsize=(16,6));
plot_acf(y_diff_12_1,lags=nlag,ax=axe);
axe = plot_colors(axe);
axe.set_xlabel("Lag");
axe.set_ylabel("ACF");
plt.show()
```

La sortie ACF de la série doublement différenciée semble pourvoir être interprétée comme un autocorrélogramme simple empirique. On identifiera donc un modèle $ARMA$ sur la série $(I-B)(I-B^{12})\ln(X_t)$.

13.4.2.2 Identification, estimation et validation de modèles

On s'appuie sur les autocorrélogrammes simples et partiels estimés :

```
# ACF (I-B)(I-B^12)ln(Xt)
fig, axe = plt.subplots(figsize=(16,6));
plot_acf(y_diff_12_1,lags=nlag, ax=axe);
axe = plot_colors(axe);
```

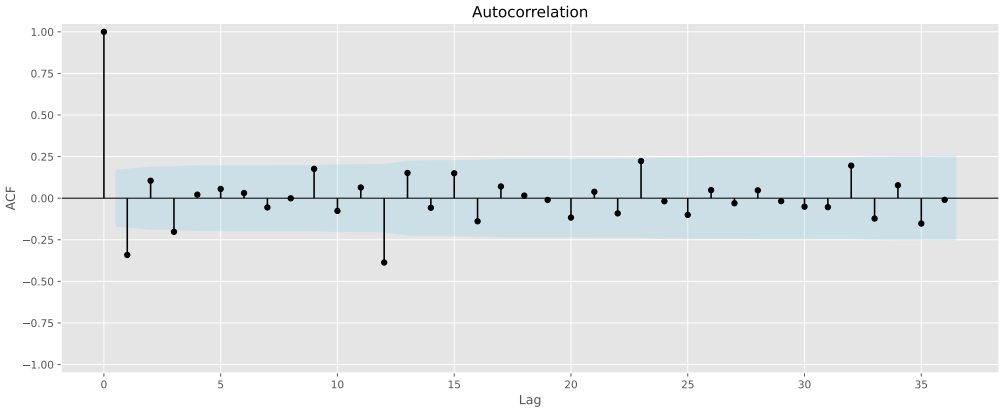


Figure 13.11 – Autocorrélation simple de la série $(I-B)\left(I-B^{12}\right)\ln\left(X_t\right)$

```
axe.set_xlabel("Lag");
axe.set_ylabel("ACF");
plt.show()
```

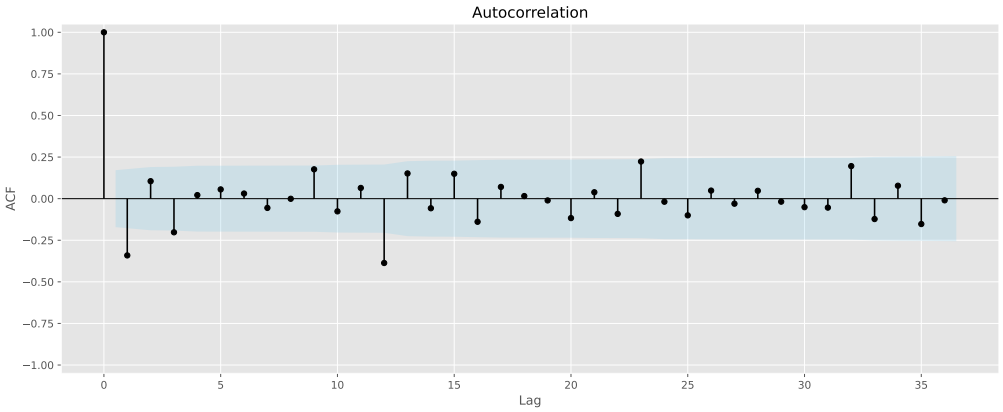


Figure 13.12 – Autocorrélation simple de la série $(I-B)\left(I-B^{12}\right)\ln\left(X_t\right)$

```
# PACF (I-B)(I-B^12)ln(Xt)
from statsmodels.graphics.tsaplots import plot_pacf
fig, axe = plt.subplots(figsize=(16,6));
plot_pacf(y_diff_12_1, lags=nlag,ax=axe);
axe = plot_colors(axe);
axe.set_xlabel("Lag");
axe.set_ylabel("PACF");
plt.show()
```

Tous les tests sont effectués au niveau de test 5%.

— **Modèle 1 :** SARIMA$(1,1,1)(1,1,1)_{12}$

On estime en premier lieu un modèle $SARIMA(1,1,1)(1,1,1)_{12}$ au vu des autocorrélogrammes empiriques simples et partiels :

$$(I-\varphi_1 B)(I-\varphi_1^{'}B^{12})(I-B)(I-B^{12})\ln(X_t)=(I+\theta_1 B)(I+\theta_1^{'}B^{12})\varepsilon_t \tag{13.72}$$

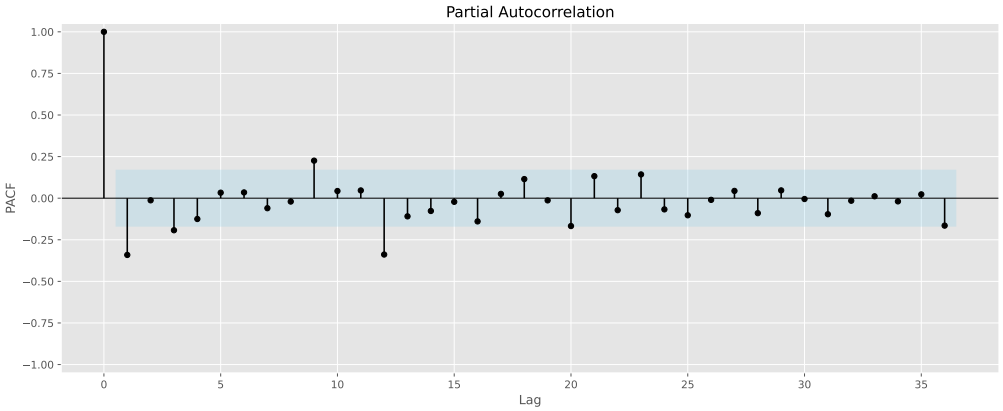


Figure 13.13 – Autocorrélation partielle de la série $(I-B)\left(I-B^{12}\right)\ln\left(X_t\right)$

```
# Estimation d'un SARIMA(1,1,1)(1,1,1)_{12}
from statsmodels.tsa.arima.model import ARIMA
model1 = ARIMA(y,order=(1,1,1),seasonal_order=(1, 1, 1, 12)).fit()
model1_params = extractParams(model1, model_type = "arima")
```

Table 13.23 – Coefficients du modèle SARIMA$(1,1,1)(1,1,1)_{12}$

	coef	std err	t	P>|t|	[0.025	0.975]
ar.L1	0.1685	0.2128	0.7916	4.286e-01	-0.2486	0.5855
ma.L1	-0.5630	0.1844	-3.0527	2.268e-03	-0.9244	-0.2015
ar.S.L12	-0.0996	0.1970	-0.5057	6.131e-01	-0.4857	0.2865
ma.S.L12	-0.4967	0.2102	-2.3633	1.811e-02	-0.9087	-0.0848
sigma2	0.0013	0.0002	8.4600	2.674e-17	0.0010	0.0016

```
# Test de ljung-box sur le modèle SARIMA(1,1,1)(1,1,1)_{12}
from statsmodels.stats.diagnostic import acorr_ljungbox
ljungbox1 = acorr_ljungbox(model1.resid,lags=[6,12,18,24,30,36],return_df=True)
ljungbox1 = ljungbox1.reset_index().rename(columns={"index": "h"})
```

Table 13.24 – Test de Ljung - Box sur les résidus du modèle SARIMA$(1,1,1)(1,1,1)_{12}$

h	lb_stat	lb_pvalue
6	0.1869	0.9999
12	26.3704	0.0095
18	26.4831	0.0892
24	26.5628	0.3252
30	26.7274	0.6375
36	26.7470	0.8688

Ce modèle ayant des paramètres non significatifs, on en test un second.

— **Modèle 2 :** SARIMA$(1,1,1)(0,1,1)_{12}$

Soit le modèle :

$$(I-\varphi_1' B^{12})(I-B)(I-B^{12})\ln(X_t) = (I+\theta_1 B)(I+\theta_1' B^{12})\varepsilon_t \tag{13.73}$$

```
# Modèle SARIM(1,1,1)(0,1,1)_{12}
model2 = ARIMA(y,order=(1,1,1),seasonal_order=(0, 1, 1, 12)).fit()
model2_params = extractParams(model2, model_type = "arima")
```

Table 13.25 – Coefficients du modèle SARIMA$(1,1,1)(0,1,1)_{12}$

	coef	std err	t	P>|t|	[0.025	0.975]
ar.L1	0.1970	0.1989	0.9904	3.220e-01	-0.1929	0.5870
ma.L1	-0.5792	0.1714	-3.3798	7.254e-04	-0.9151	-0.2433
ma.S.L12	-0.5643	0.1043	-5.4130	6.197e-08	-0.7687	-0.3600
sigma2	0.0013	0.0002	8.5917	8.571e-18	0.0010	0.0016

```
# Test de ljung-box sur le modèle SARIMA(1,1,1)(0,1,1)_{12}
ljungbox2 = acorr_ljungbox(model2.resid, lags=[6,12,18,24,30,36],return_df=True)
ljungbox2 = ljungbox2.reset_index().rename(columns={"index": "h"})
```

Table 13.26 – Test de Ljung - Box sur les résidus du modèle SARIMA$(1,1,1)(0,1,1)_{12}$

h	lb_stat	lb_pvalue
6	0.1873	0.9999
12	26.3948	0.0094
18	26.5096	0.0887
24	26.5925	0.3238
30	26.7705	0.6353
36	26.7911	0.8674

Ce modèle ayant des paramètres non significatifs, on en teste un troisième.

— **Modèle 3 :** $SARIMA(0,1,1)(0,1,1)_{12}$

Soit le modèle :

$$(I-B)(I-B^{12})\ln(X_t) = (I+\theta_1 B)(I+\theta_1' B^{12})\varepsilon_t \tag{13.74}$$

```
# Modèle SARIMA(0,1,1)(0,1,1)_{12}
model3 = ARIMA(y,order=(0,1,1),seasonal_order=(0, 1, 1, 12)).fit()
model3_params = extractParams(model3, model_type = "arima")
```

Table 13.27 – Coefficients du modèle SARIMA$(0,1,1)(0,1,1)_{12}$

	coef	std err	t	P>|t|	[0.025	0.975]
ma.L1	-0.4016	0.0730	-5.5010	3.777e-08	-0.5448	-0.2585
ma.S.L12	-0.5570	0.0963	-5.7849	7.254e-09	-0.7457	-0.3683
sigma2	0.0013	0.0001	9.1210	7.444e-20	0.0011	0.0016

```
# Test de ljung-box sur le modèle SARIMA(0,1,1)(0,1,1)_{12}
ljungbox3 = acorr_ljungbox(model3.resid, lags=[6,12,18,24,30,36],return_df=True)
ljungbox3 = ljungbox3.reset_index().rename(columns={"index": "h"})
```

Les tests de significativité des paramètres et de blancheur du résidu sont validés au niveau 5%.

Table 13.28 – Test de Ljung - Box sur les résidus du modèle SARIMA$(0,1,1)(0,1,1)_{12}$

h	lb_stat	lb_pvalue
6	0.1815	0.9999
12	26.4100	0.0094
18	26.5282	0.0883
24	26.6178	0.3226
30	26.7856	0.6345
36	26.8053	0.8670

```
# Diagnostics des résidus
fig, axe = plt.subplots(figsize=(16,6))
model3.plot_diagnostics(fig=fig);
fig.tight_layout();
plt.show()
```

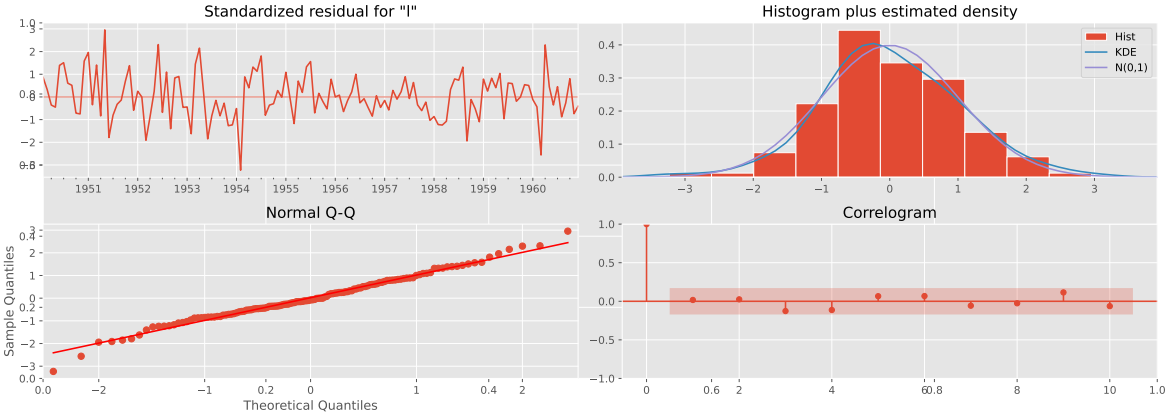


Figure 13.14 – Diagnostic des résidus

```
# Test de normalité de Shapiro - Wilk
import scipy.stats as st
def ShapiroWilkTest(res):
  sw = pd.DataFrame({
    "statistic":st.shapiro(res.resid)[0],"pvalue":st.shapiro(res.resid)[1]
  }, index = ["Shapiro - Wilk Test"])
  return sw
# Application
res_sw = ShapiroWilkTest(model3)
```

Table 13.29 – Test de normalité de Shapiro - Wilk pour le modèle SARIMA$(0,1,1)(0,1,1)_{12}$

	statistic	pvalue
Shapiro - Wilk Test	0.169	0

Le test de normalité est rejeté pour ce modèle.

13.4.2.3 Procédure de sélection automatique de modèles

Nous pouvons constater que la sélection automatique testée ici avec un critère BIC fournit le même résultat :

```
# Sélection automatique
from pmdarima.arima import auto_arima
model4 = auto_arima(y,start_p=1,start_q=1,m=s,test='adf',criterion = 'bic')
res_auto_arima = pd.DataFrame({
    "coef": list(model4.params()), "pvalue" : list(model4.pvalues()),
    "[0.025" : model4.conf_int(alpha = 0.05).values[:,0],
    "0.975]": model4.conf_int(alpha = 0.05).values[:,1]
},index = list(model4.params().index))
```

Table 13.30 – Coefficients du modèle SARIMA par sélection automatique

	coef	pvalue	[0.025	0.975]
ma.L1	-0.4016	0	-0.5448	-0.2585
ma.S.L12	-0.5570	0	-0.7457	-0.3683
sigma2	0.0013	0	0.0011	0.0016

13.4.2.4 Prévision

```
# Prévision modèle 3
pred_model3 = model3.get_forecast(steps = 12,alpha = 0.05).summary_frame()
```

Table 13.31 – Prévision à partir du modèle SARIMA$(0,1,1)(0,1,1)_{12}$

	mean	mean_se	mean_ci_lower	mean_ci_upper
1961-01-31	6.1102	0.0367	6.0382	6.1821
1961-02-28	6.0538	0.0428	5.9699	6.1376
1961-03-31	6.1717	0.0481	6.0775	6.2660
1961-04-30	6.1993	0.0529	6.0957	6.3029
1961-05-31	6.2326	0.0573	6.1203	6.3448
1961-06-30	6.3688	0.0613	6.2486	6.4890
1961-07-31	6.5073	0.0651	6.3796	6.6350
1961-08-31	6.5029	0.0687	6.3682	6.6376
1961-09-30	6.3247	0.0722	6.1832	6.4661
1961-10-31	6.2090	0.0754	6.0611	6.3569
1961-11-30	6.0635	0.0786	5.9095	6.2175
1961-12-31	6.1680	0.0816	6.0081	6.3279

```
# Forecast
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(donnee['passengers'],label = 'Air Passengers',color="black");
axe.plot(np.exp(pred_model3['mean_ci_upper']),color = "red",
         linestyle = '--', label = 'int95%_sup');
axe.plot(np.exp(pred_model3['mean']), label = 'Prevision',color='orange');
axe.plot(np.exp(pred_model3['mean_ci_lower']),color = "green",
         linestyle = '--', label = 'int95%_inf');
axe.set_xlabel('année');
axe.set_ylabel('valeur');
```

```
plt.legend();
plt.show()
```

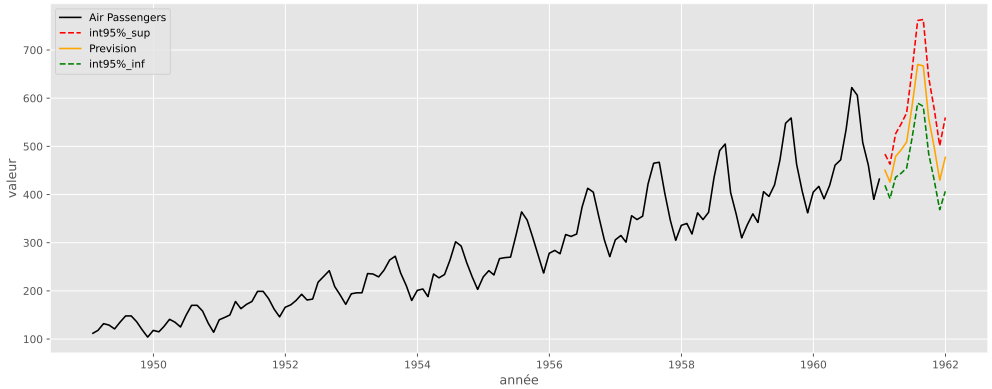


Figure 13.15 – Prévision année 1961

```
# Forecast
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(donnee.passengers[donnee['passengers'].index>"1959-12-31"],
         label = 'Air Passengers',color='black');
axe.plot(np.exp(pred_model3['mean_ci_upper']),color = "red",
         linestyle = '--', label = 'int95%_sup');
axe.plot(np.exp(pred_model3['mean']), label = 'Prevision',color="orange");
axe.plot(np.exp(pred_model3['mean_ci_lower']),color = "green",
         linestyle = '--', label = 'int95%_inf');
axe.set_xlabel('année');
axe.set_ylabel('valeur');
plt.show()
```

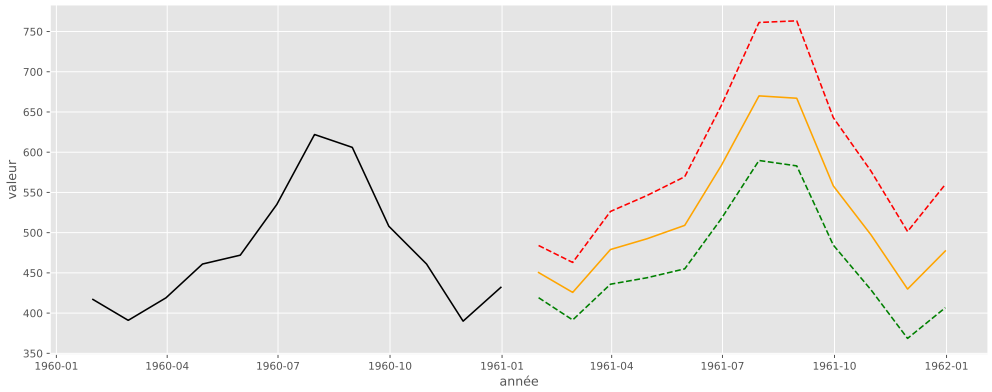


Figure 13.16 – Prévision

13.4.2.5 Analyse a posteriori

On tronque la série de l'année 1960, qu'on cherche ensuite à prévoir à partir de l'historique 1949 – 1959.

```
# Subdivision apprentissage - test
x_train = donnee.passengers[donnee['passengers'].index<="1959-12-31"]
x_train_log = np.log(x_train) # à entraîner
x_test = donnee.passengers[donnee['passengers'].index>"1959-12-31"] # à prévoir
```

On vérifie que le modèle 3 sur la série tronquée est toujours validé.

```
# Vérification
model3_tronc = ARIMA(x_train_log,order=(0,1,1),seasonal_order=(0,1,1,12)).fit()
model3_tronc_params = extractParams(model3_tronc, model_type = "arima")
```

Table 13.32 – Coefficients du modèle SARIMA$(0,1,1)(0,1,1)_{12}$ tronqué

	coef	std err	t	P>|t|	[0.025	0.975]
ma.L1	-0.3483	0.0810	-4.3003	1.705e-05	-0.5071	-0.1896
ma.S.L12	-0.5624	0.0944	-5.9576	2.560e-09	-0.7474	-0.3774
sigma2	0.0013	0.0002	8.4962	1.959e-17	0.0010	0.0016

```
# Test de shapiro - Wilk sur les résidus du mondèle tronqué
res_sw_tronc = ShapiroWilkTest(model3_tronc)
```

Table 13.33 – Test de normalité de Shapiro - Wilk pour le modèle SARIMA$(0,1,1)(0,1,1)_{12}$ tronqué

	statistic	pvalue
Shapiro - Wilk Test	0.1749114	0

On constate que la réalisation 1960 est dans l'intervalle de prévision à 95% (basé sur les données antérieures à 1959) :

```
# Prévision
pred_model3_tronc = (model3_tronc.get_forecast(steps = 12,alpha = 0.05 )
                     .summary_frame())
```

Table 13.34 – Prévision à partir du modèle SARIMA$(0,1,1)(0,1,1)_{12}$ tronqué

	mean	mean_se	mean_ci_lower	mean_ci_upper
1960-01-31	6.0386	0.0362	5.9676	6.1096
1960-02-29	5.9888	0.0432	5.9040	6.0735
1960-03-31	6.1454	0.0493	6.0489	6.2420
1960-04-30	6.1190	0.0546	6.0119	6.2261
1960-05-31	6.1596	0.0595	6.0430	6.2763
1960-06-30	6.3047	0.0640	6.1792	6.4301
1960-07-31	6.4333	0.0682	6.2995	6.5670
1960-08-31	6.4459	0.0722	6.3044	6.5875
1960-09-30	6.2667	0.0760	6.1178	6.4156
1960-10-31	6.1362	0.0795	5.9803	6.2921
1960-11-30	6.0079	0.0830	5.8453	6.1705
1960-12-31	6.1143	0.0863	5.9453	6.2834

```
# Forecast
fig, axe = plt.subplots(figsize=(16,6));
axe.plot(x_test,label = 'Air Passengers',color= "black");
```

```
axe.plot(np.exp(pred_model3_tronc['mean_ci_upper']),color = "red",
         linestyle = '--', label = 'int95%_sup');
axe.plot(np.exp(pred_model3_tronc['mean']), label = 'Prevision',color='orange');
axe.plot(np.exp(pred_model3_tronc['mean_ci_lower']),color = "green",
         linestyle = '--', label = 'int95%_inf');
axe.set_xlabel('t');
axe.set_ylabel('');
plt.legend();
plt.show()
```

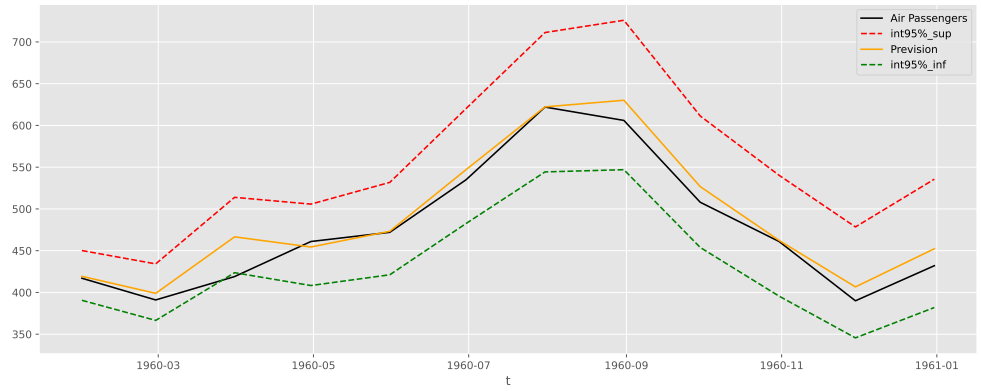


Figure 13.17 – Analyse a posteriori (Forecast)

On calcule les RMSE, MAPE et R^2.

```
# RMSE et MAPE
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import r2_score
rmse=mean_squared_error(x_test,np.exp(pred_model3_tronc['mean']),squared=False)
mape=mean_absolute_percentage_error(x_test,np.exp(pred_model3_tronc['mean']))
r2score = r2_score(x_test, np.exp(pred_model3_tronc['mean']))
res = pd.DataFrame({'RMSE' : rmse,'MAPE' : 100*mape,'R2 score' : r2score},
                   index = ['statistic'])
```

Table 13.35 – Performance du modèle SARIMA$(0,1,1)(0,1,1)_{12}$ tronqué

	RMSE	MAPE	R2 score
statistic	18.5932	2.9043	0.9376

L'interprétation des critères d'erreur dépend de la série et de la qualité de prévision exigée. Dans le cas présent, un MAPE de 2.9% semble satisfaisant a priori.

Troisième partie

Économétrie des processus non linéaires

14

Processus stochastiques non linéaires

Sommaire

L'économétrie des séries temporelles s'est appuyée pendant longtemps sur la modélisation linéaire, et de manière spécifique sur les modèles du type ARMA. Le principale avantage de ces processus est qu'ils sont faciles à appliquer. Le fait qu'ils reposent sur une représentation linéaire rend plus aisé l'estimation et la prévision. Toutefois, on peut souligner que cet avantage constitue en même temps une limite majeure dans la mesure où la formalisation linéaire a la propriété de restreindre automatiquement le type de dynamiques possibles. En effet, l'hypothèse de linéarité peut être considérée comme une hypothèse forte, notamment en ce qui concerne l'évolution des séries macroéconomiques financières. En réalité, la dynamique des marchés financiers au cours du temps s'est révélée plus complexe de tel sorte qu'on peut penser que les séries financières soient caractérisées par des dynamiques de type non linéaire.

Nous commençons par dresser un rapide panorama des insuffisances des modèles linéaires. On présente ensuite les processus à non linéarités en variance.

14.1 Les insuffisances des modèles linéaires

Il semple important de justifier tout d'abord d'un point de vue économique, l'adoption de l'hypothèse de non linéarité.

14.1.1 Justifications économiques de la non linéarité

Dans les années 30, le paradigme dominant en économie était celui de Frisch et Slutsky. Les modèles proposés par les deux auteurs avaient pour ambition de décrire les cycles économiques sur l'hypothèse de linéarité. Ils reflétaient la manière donc pouvaient se propager des chocs dans un système économique. En absence d'un choc exogène[1], la dynamique du système pouvait suivre quatre trajectoires (évolutions) possibles :

1. choc exogène : évènement brutal et imprévisible qui affecte le fonctionnement d'une économie et perturbe l'équilibre.

1. Non oscillatoire et stable (convergence monotone) ;
2. Non oscillatoire et instable ;
3. Oscillatoire et stable (fluctuations amorties) ;
4. Oscillatoire et instable (fluctuations explosives).

Ces trajectoires sont celles qui peuvent générer un modèle linéaire de manière endogène. Il n'existe que deux possibilités pour les fluctuations : soit elles explosent, soit elles disparaissent (c'est-à-dire lissées de manière progressive jusqu'au point nul). En d'autres termes, en absence de choc, les modèles linéaires ne sont pas en mesure d'expliquer le caractère auto-entretenu des fluctuations (sauf pour des variables très particulières des paramètres).

Cette représentation linéaire de l'économie est insuffisante pour plusieurs raisons :

— L'existence des variations totalement amorties n'est pas réaliste dans les économies actuelles où les chocs ont tendance à intervenir fréquemment ;
— L'existence des fluctuations explosives ne semble non plus suffisamment réaliste puisque les variations peuvent être affectées par des contraintes techniques ou institutionnelles affectant les variables ;
— Les modèles linéaires sont peu appropriés pour rendre compte de l'asymétrie des phases du cycle économique pourtant on sait que les phases de récession sont plus longues que les phases d'expansion. Par ailleurs, certains auteurs ont montré par exemple que le taux de chômage peut augmenter plus rapidement en fonction de certaines phases du cycle économique et inversement.

En s'appuyant sur les travaux de Beaudry and Koop (1993), on s'est rendu compte que les chocs défavorables ont des conséquences plus importantes et plus durables que des chocs favorables. Ces limites rendent nécessaire les recours aux modèles non linéaires en économétrie. Hicks (1950) fut le premier a développé ce type de modèle, ensuite Goodwin (1951). Mais à cette époque, ces modèles n'ont pas reçu un écho très favorable sans doute à cause d'un faible développement des instruments en termes de techniques quantitatives. Ce n'est que récemment avec le développement de ces derniers que les applications théoriques et techniques des modèles non linéaires se sont rendues possibles.

14.1.2 La linéarité : une hypothèse économétrique très restrictive

Comme nous l'avons souligné précédemment, les processus dominant en économétrie des séries macroéconomiques financières temporelles ont été pendant longtemps mais également aujourd'hui les processus $ARMA(p,q)$. Sous forme générale, le processus $ARMA(p,q)$ s'écrit :

$$\Phi(B)X_t = \Theta(B)\varepsilon_t \tag{14.1}$$

soit encore :

$$X_t = \sum_{i=1}^{i=p} \phi_i X_{t-i} + \sum_{j=0}^{j=q} \theta_j \varepsilon_{t-j}, \quad \text{avec } \theta_0 = 1 \tag{14.2}$$

Dans ce modèle, ε_t est un bruit blanc ; $\Phi(B)$ et $\Theta(B)$ sont des polynômes retard d'ordre p et q respectivement. Ces polynômes résument l'ensemble des paramètres à estimer lié au processus stochastique X_t et au choc respectivement. Ces processus sont fondés sur une combinaison linéaire des valeurs passées et présentes de la variable endogène X_t et du bruit ε_t.

Les modèles ARMA sont devenus insuffisants pour rendre compte des problématiques posées dans le domaine de la finance. Les séries macroéconomiques financières sont en général marquées

par des dynamiques non linéaires et par une volatilité (variabilité) excessive au cours du temps. D'une manière générale, on attribue aux modèles linéaires deux principaux inconvénients :

1. Les modèles $ARMA(p,q)$ ne sont pas adaptés pour rendre compte des phénomènes d'asymétrie qui sont en réalité une caractéristique des cycles économiques, des coûts d'ajustement et des rigidités institutionnelles, ni alors des ruptures de forte amplitude ;
2. Les modèles $ARMA(p,q)$ ne prennent en considération que les moments d'ordre 2 à travers la fonction d'auto-covariance, ce qui met en lumière une exploitation insuffisante de l'information contenue dans la série financière X_t. Les modèles non linéaires prennent en compte des moments supérieurs à l'ordre 2 et permettent d'affiner l'étude de la dynamique de la série.

Ces limites d'un point de vue économétrique justifient le recours à des modèles non linéaires. On définit un processus stochastique non linéaire comme tout processus qui peut être décrit de la manière suivante :

$$X_t = m\left(\varepsilon_{t-1}, \varepsilon_{t-2}, \dots\right) + h\left(\varepsilon_{t-1}, \varepsilon_{t-2}, \dots\right)\varepsilon_t \tag{14.3}$$

où $m(.)$ et $h(.)$ représentent respectivement la moyenne conditionnelle du processus X_t et le coefficient de proportionnalité entre X_t et le choc ε_t. On distingue deux grandes catégories de modèles non linéaires :

1. Les modèles à non linéarités dans la moyenne pour lesquels $m(.)$ est non linéaire.
2. Les modèles à non linéarités en variance pour lesquels $h(.)$ est non linéaire.

Exemple 14.1

— $X_t = \varepsilon_t + \alpha\varepsilon_{t-1}^2$.

Ici, la fonction $m(.) = \alpha\varepsilon_{t-1}^2$ et la fonction $h = 1$: non linéarité en moyenne.

— $X_t = \varepsilon_t\sqrt{\alpha\varepsilon_{t-1}^2}$

Ici, la fonction $m(.) = 0$ et $h = \sqrt{\alpha\varepsilon_{t-1}^2}$: non linéarité en variance.

— $X_t = \varepsilon_t + \alpha\varepsilon_{t-1}^2 + \varepsilon_t\sqrt{\alpha\varepsilon_{t-1}^2}$

Ici, la fonction $m(.) = \alpha\varepsilon_{t-1}^2$ et la fonction $h = \left(1 + \sqrt{\alpha\varepsilon_{t-1}^2}\right)$: non linéarité en moyenne et en variance.

Dans ce chapitre, on présentera uniquement les processus à non linéarités en variance.

14.2 Les processus non linéaires en variance

La variance dans les modèles précédemment étudiés était supposée toujours constante (comportement « homoscédastique »). Par conséquent, l'hypothèse de processus ARMA stationnaire (en variance) ne permet pas de prendre en compte les changements de la variance au cours du temps (que l'on qualifie de comportement « hétéroscédastique »). En pratique, la prise en compte de ce phénomène accroît le potentiel prédictif des modèles. D'où la nécessité d'aller vers des modélisations non linéaires avec l'introduction de modèles non linéaires tels que les modèles (G)ARCH (Generalized AutoRegressive Conditional Heteroscedasticity). Ces modèles rendent compte du fait stylisé connu, dit de clustering de volatilité (agrégats de volatilité), à savoir que les périodes de forte volatilité alternent avec les périodes de faible volatilité. La modélisation ARCH/GARCH et ses extensions (I-GARCH, FI-GARCH, GI-GARCH, ...) correspond à une représentation spécifique de la non linéarité qui permet une modélisation simple de l'incertitude

(modélisation de la variance de ε_t). De même que les modèles ARMA étudient les séries temporelles en modélisant leur auto-corrélation, les modèles $(G)ARCH$ les étudient en modélisant leur variance.

14.2.1 Présentation générale

Considérons le modèle $AR(p)$ suivant :

$$\Phi(B)X_t = \varepsilon_t \tag{14.4}$$

où ε_t est un bruit blanc de variance σ_ε^2.

Soit $\mathcal{F}_{t-1} = \sigma\left(X_{t-1}, X_{t-2}, \dots\right)$ représentant la $\sigma-$algèbre engendrée par toute l'information sur le passé de X_t jusqu'au temps $t-1$. On a les propriétés suivantes :

Proposition 14.1 (Moyenne et variance non conditionnelle)

La moyenne et la variance non conditionnelle de X_t sont définies par :

$$\begin{cases} \mathbb{E}(X_t) = 0 \\ \mathrm{V}(X_t) = \sigma_\varepsilon^2 \sum_{i=1}^{\infty} \phi_i^2 \end{cases} \tag{14.5}$$

Ainsi, pour un processus $AR(1) : X_t = \phi_1 X_{t-1} + \varepsilon_t$ avec $\varepsilon_t \sim BB(0, \sigma_\varepsilon^2)$ et $|\phi_1| < 1$, on a :

$$\begin{cases} \mathbb{E}(X_t) = 0 \\ \mathrm{V}(X_t) = \dfrac{\sigma_\varepsilon^2}{1-\phi_1^2} \end{cases} \tag{14.6}$$

Proposition 14.2 (Moyenne et variance conditionnelle)

La moyenne et la variance conditionnelle de X_t sont définies par :

$$\begin{cases} \mathbb{E}\left(X_t | \mathcal{F}_{t-1}\right) = \sum_{i=1}^{p} \phi_i X_{t-i} \\ \mathrm{V}\left(X_t | \mathcal{F}_{t-1}\right) = \sigma_\varepsilon^2 \end{cases} \tag{14.7}$$

On remarque que la moyenne conditionnelle dépend de l'information disponible au temps $t-1$ et n'est donc pas nécessairement constante. Par contre, la variance conditionnelle est fixe et ne dépend pas de l'information disponible au temps $t-1$. En fait, l'hypothèse de $\varepsilon_t \sim BB\mathcal{N}(0, \sigma_\varepsilon^2)$ nous amène à ce résultat, ce qui est manifestement trop restrictif. Il nous faut un modèle beaucoup plus souple et plus réaliste de la variance conditionnelle.

Considérons le modèle $AR(p)$ suivant :

$$\Phi(B)X_t = \varepsilon_t \tag{14.8}$$

où ε_t suit une loi normale de moyenne nulle et de variance σ_t^2 et :

$$\sigma_t^2 = \omega + \sum_{i=1}^{i=p} \alpha_i \varepsilon_{t-i}^2 \tag{14.9}$$

La moyenne et la variance conditionnelle de X_t sont définies par :

$$\mathbb{E}\left(X_t|\mathcal{F}_{t-1}\right) = \sum_{i=1}^{i=p} \phi_i X_{t-i}, \quad V\left(X_t|\mathcal{F}_{t-1}\right) = \omega + \sum_{i=1}^{i=p} \alpha_i \varepsilon_{t-i}^2 \tag{14.10}$$

On remarque que la variance conditionnelle n'est plus fixe, elle dépend de l'information disponible au temps $t-1$. En fait, l'hypothèse de $\varepsilon_t \sim BB(0, \sigma_t^2)$ nous amène à ce résultat.

La variance non conditionnelle de X_t est définie par

$$V\left(X_t\right) = \mathbb{E}\left(V\left(X_t|\mathcal{F}_{t-1}\right)\right) + V\left(\mathbb{E}\left(X_t|\mathcal{F}_{t-1}\right)\right) \tag{14.11}$$

equivalent à

$$V\left(X_t\right) = \frac{\omega}{\left(1 - \sum_{i=1}^{i=p} \alpha_i\right)\left(1 - \sum_{i=1}^{i=p} \phi_i^2\right)} \tag{14.12}$$

On note $\varepsilon_t = \sigma_t z_t$ où z_t est un processus dont l'espérance conditionnelle est nulle et dont la variance conditionnelle est constante et égale à l'unité. On admet également que z_t suit une loi normale. Alors, le processus ε_t est appelé processus ARCH d'ordre p et il est noté $ARCH(p)$. Et X_t est alors appelé modèle $AR(p)$ à erreur ARCH.

14.2.2 Les modèles ARCH

Engle (1982) a été le premier à développer les modèles ARCH afin de permettre à la variance d'une série chronologique (ou chronique) de dépendre de l'ensemble d'informations disponible, et notamment du temps. Cette classe de modèle a pour objectif de pallier les insuffisances des représentations (modélisations) $ARMA$ traditionnelles non adaptées pour analyser les problèmes ou des phénomènes dans le domaine de l'économie et de la finance. En effet, les séries macroéconomiques financières sont caractérisées par une volatilité variable et par des phénomènes d'asymétrie qui ne peuvent être pris en compte par les modélisations de type ARMA.

Or, d'un point de vue économique, il est particulièrement important de comprendre et de modéliser la volatilité d'une série. En effet, les décisions en termes d'investissement dépendent fortement, non seulement de l'évaluation des rentabilités futures, mais également des risques afférents aux diverses actions constituant les portefeuilles. L'estimation de la volatilité de la rentabilité d'une action donne une mesure du risque qui s'y est attaché. En outre, si le processus suivi par la volatilité est correctement spécifié, celui-ci peut fournir des informations utiles à la détermination du processus qui génère les rentabilités [2].

Les modèles de type ARCH sont basés sur une paramétrisation endogène de la variance conditionnelle. Ces processus ont connu un succès important dans les applications financières [3] en raison notamment du fait qu'ils permettent de réécrire les modèles structurels concernant les

2. La volatilité joue un rôle considérable dans la théorie d'évaluation des options. En effet, si l'on considère le modèle de Black-Scholes, la volatilité est directement introduite dans la détermination du prix des options.

3. Pour une revue de la littérature théorique et empirique, on pourra se reporter à Bollerslev, Chou, Jayaraman et Kroner (1991), Zakoian (1992), Bollerslev, Engle et Nelson (1994), Bera et Higgins (1995) et Palm (1996).

choix de portefeuille optimaux, les liens entre rentabilité d'actifs et rentabilité du portefeuille de marché, etc.

14.3 Processus ARCH

Le processus $ARCH(q)$ est le modèle de base des processus ARCH proposé par Engle (1982) lors d'une étude sur la variance de l'inflation en Grande Bretagne. Le modèle $ARCH(q)$ est basé sur une paramétrisation quadratique de la variance conditionnelle. σ_t^2 apparaît comme une fonction linéaire des q valeurs du processus du carré des innovations.

Définition 14.1

Un processus $(\varepsilon_t)_t$ est défini comme étant un processus $ARCH(q)$ s'il vérifie le système suivant :

$$\begin{cases} \varepsilon_t = \sigma_t z_t \\ \sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 = \omega + \alpha(B)\varepsilon_t^2 \end{cases} \tag{14.13}$$

avec $\omega > 0$ et $\alpha_i \geq 0 \ \forall i$ et où $(z_t)_{t\in\mathbb{Z}}$ est un bruit blanc gaussien centré de variance unitaire, indépendant du passé de $(\varepsilon_t)_t \in \mathbb{Z}$. ω représente le niveau minimum de la volatilité. Les contraintes sur les coefficients garantissent la positivité de la variance conditionnelle.

La variance conditionnelle dépend du temps : si les valeurs précédentes sont grandes (en valeur absolue), la variance sera grande, et inversement. Ainsi, si on observe un choc dans la série (valeur anormalement grande), il sera suivi d'une période de haute volatilité, dont la durée dépend de l'ordre q du modèle ARCH. Le modèle ARCH(q) permet de prendre en compte les regroupements de volatilité, c'est-à-dire le fait que les fortes (respectivement faibles) variations de prix sont suivies par d'autres fortes (respectivement faibles) variations de prix mais dont le signe n'est pas prévisible.

Soit $\mathcal{F}_{t-1} = \sigma(\varepsilon_{t-1}, \varepsilon_{t-2}, \dots)$ représentant la $\sigma-$algèbre engendrée par toute l'information sur le passé de ε_t jusqu'au temps $t-1$. On a les propriétés suivantes :

14.3.1 Propriétés d'un processus ARCH

Proposition 14.3

Si $(\varepsilon_t)_{t\in\mathbb{Z}}$ est un processus $ARCH(q)$, donné par (14.13), alors on montre que :

1. Les moyennes non conditionnelle et conditionnelle de $(\varepsilon_t)_{t\in\mathbb{Z}}$ sont respectivement égales à :

$$\begin{cases} \mathbb{E}(\varepsilon_t) = 0 \\ \mathbb{E}(\varepsilon_t|\mathcal{F}_{t-1}) = 0 \end{cases} \tag{14.14}$$

2. Les variances non conditionnelle et conditionnelle de $(\varepsilon_t)_{t\in\mathbb{Z}}$ sont respectivement égales à :

$$\begin{cases} \mathrm{V}(\varepsilon_t) = \mathbb{E}(\varepsilon_t^2) = \dfrac{\omega}{1 - \sum_{i=1}^{q} \alpha_i} & \text{si } \sum_{i=1}^{q} \alpha_i < 1 \\ \mathrm{V}(\varepsilon_t|\mathcal{F}_{t-1}) = \mathbb{E}(\varepsilon_t^2|\mathcal{F}_{t-1}) = \sigma_t^2 \end{cases} \tag{14.15}$$

Cette expression laisse entendre que la variance conditionnelle de ε_t évolue en fonction des valeurs précédentes de ε_t^2 comme un modèle AR(p) : d'où le nom de modèle autoregressive conditional heteroskedastic d'ordre p, ARCH(p). On voit que la variance conditionnelle qui est un indicateur de la volatilité d'un titre varie au cours du temps à la différence des processus $ARMA$ où elle est constante.

3. Les autocovariances non conditionnelles et conditionnelles de $(\varepsilon_t)_{t\in\mathbb{Z}}$ sont respectivement égale à :
$$\begin{cases} \text{Cov}(\varepsilon_t, \varepsilon_{t+h}) = \sigma_h = 0, \quad h > 0 \\ \text{Cov}\left(\varepsilon_t, \varepsilon_{t+h} | \mathcal{F}_{t-1}\right) = 0 \end{cases} \tag{14.16}$$
4. La loi conditionnelle de $(\varepsilon_t)_{t\in\mathbb{Z}}$ est une loi normale centrée de variance σ_t^2, c'est-à-dire :
$$\mathcal{L}\left(\varepsilon_t | \mathcal{F}_{t-1}\right) \sim \mathcal{N}(0, \sigma_t^2) \tag{14.17}$$
5. Si l'on pose $\eta_t = \varepsilon_t^2 - \sigma_t^2$, alors on a :
$$\varepsilon_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 + \eta_t \tag{14.18}$$
où $(\eta_t)_{t\in\mathbb{Z}}$ est un processus centré, non corrélé et de variance non constante. Ainsi, le processus $(\varepsilon_t^2)_{t\in\mathbb{Z}}$ admet une représentation proche d'un processus AR(p). Ce point de la proposition sera utile pour détecter la présence d'hétéroscédasticité conditionnelle (en effet, la présence de valeurs significativement non nulles pour de nombreux retard constitue un premier indicateur de la présence d'hétéroscédasticité) mais aussi pour estimer les paramètres du processus à partir de la série des carrés.

Remarque 14.1

Un processus ARCH est conditionnellement hétéroscédastique mais inconditionnellement homoscédastique !. Sa moyenne non conditionnelle est ses autocovariances $\gamma(h)$, $h > 0$ étant nulles, ce processus peut être caractérisé comme étant un processus bruit blanc non indépendant (ou bruit blanc faible).

Proposition 14.4 (Condition suffisante de stationnarité)

Le processus $(\varepsilon_t)_{t\in\mathbb{Z}}$ est stationnaire si et seulement si
$$\sum_{i=1}^{p} \alpha_i < 1 \tag{14.19}$$

Moments du processus ARCH (peuve)

1. Moyenne conditionnelle et non conditionelle
$$\begin{cases} \mathbb{E}\left(\varepsilon_t | \mathcal{F}_{t-1}\right) & = \mathbb{E}\left(z_t \sigma_t | \mathcal{F}_{t-1}\right) = \sigma_t \mathbb{E}\left(z_t | \mathcal{F}_{t-1}\right) = 0 \\ \mathbb{E}(\varepsilon_t) & = \mathbb{E}\left[\mathbb{E}(\varepsilon_t) | \mathcal{F}_{t-1}\right] = 0 \end{cases} \tag{14.20}$$
2. Variance conditionelle et non conditionnelle
$$\begin{cases} \text{V}\left(\varepsilon_t | \mathcal{F}_{t-1}\right) & = \text{V}\left(z_t \sigma_t | \mathcal{F}_{t-1}\right) = \sigma_t^2 \text{V}\left(z_t | \mathcal{F}_{t-1}\right) = \sigma_t^2 \text{V}(\varepsilon_t) = \sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 \\ \text{V}(\varepsilon_t) & = \mathbb{E}\left[\left(\varepsilon_t - \mathbb{E}(\varepsilon_t)\right)^2\right] = \mathbb{E}\left(\varepsilon_t^2\right) \end{cases} \tag{14.21}$$

De l'écriture autorégressive sur ε_t^2 du modèle ARCH sous hypothèse de stationnarité $\left(\mathbb{E}\left(\varepsilon_t^2\right) = \omega + \sum_{i=1}^{q} \alpha_i \mathbb{E}\left(\varepsilon_{t-i}^2\right)\right)$, on a :

$$\mathbb{E}\left(\varepsilon_t^2\right) = \mathrm{V}(\varepsilon_t) = \frac{\omega}{1 - \sum_{i=1}^{i=q} \alpha_i} \tag{14.22}$$

14.3.1.1 Caractéristique d'un ARCH(1)

Le processus ARCH le plus étudié et le plus utilisé dans la littérature statistique est le processus ARCH(1), défini, pour tout $t \in \mathbb{Z}$, par

$$\begin{cases} \varepsilon_t = \sigma_t z_t \\ \sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 \end{cases} \qquad z_t \sim \mathcal{N}(0,1) \tag{14.23}$$

où $\omega > 0$ et $0 \leq \alpha_1 < 1$. Ce processus peut être réécrit de la manière suivante :

$$\varepsilon_t = z_t \sqrt{\omega + \alpha_1 \varepsilon_{t-1}^2} \tag{14.24}$$

La figure 14.1 donne la représentation d'une trajectoire de longueur $T = 200$, issue d'un processus ARCH(1) en utilisant l'équation (14.13) avec $\omega = 0.1$ et $\alpha_1 = 0.3$.

```
# Simulation d'un processus ARCH(1)
import numpy as np
import matplotlib.pyplot as plt
def simulate_ARCH(n, omega, alpha):
    np.random.seed(4)
    # Initialize the parameters
    white_noise = np.random.normal(size = n)
    resid = np.zeros_like(white_noise)
    variance = np.zeros_like(white_noise)
    for t in range(1, n):
        # Simulate the variance (sigma squared)
        variance[t] = omega + alpha * resid[t-1]**2
        # Simulate the residuals
        resid[t] = np.sqrt(variance[t]) * white_noise[t]

    return resid, variance

# Simulate a ARCH(1) series
arch_resid, arch_variance = simulate_ARCH(n= 200,omega = 0.1, alpha = 0.3)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(arch_resid,color='black',label='Residuals');
axe.plot(arch_variance,color='red',label='Variance');
axe.set_xlabel('t');
axe.legend();
plt.show()
```

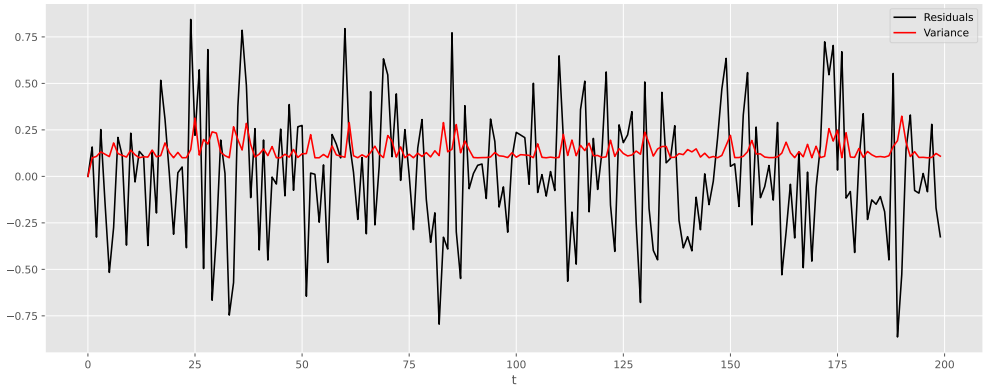


Figure 14.1 – Simulation d'un processus ARCH(1)

La figure 14.2 donne la représentation d'une trajectoire d'un processus AR(1) à erreurs ARCH(1). Le processus AR(1) simulé est : $X_t = 0.8X_{t-1} + \varepsilon_t$ avec $\varepsilon_t = z_t\sqrt{0.1 + 0.3\varepsilon_{t-1}^2}$ où $z_t \sim \mathcal{N}(0,1)$.

```
# Simulation du processus X(t) = 0.8X(t-1)+epsilon(t)
np.random.seed(12345)
x = np.zeros(200)
x[0] = np.random.normal(0,1,1)
for i in range(1,200):
    x[i] = 0.8*x[i-1]+arch_resid[i]

# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(x, color='black');
axe.set_xlabel('t');
axe.set_ylabel('$X_{t}$');
plt.show()
```

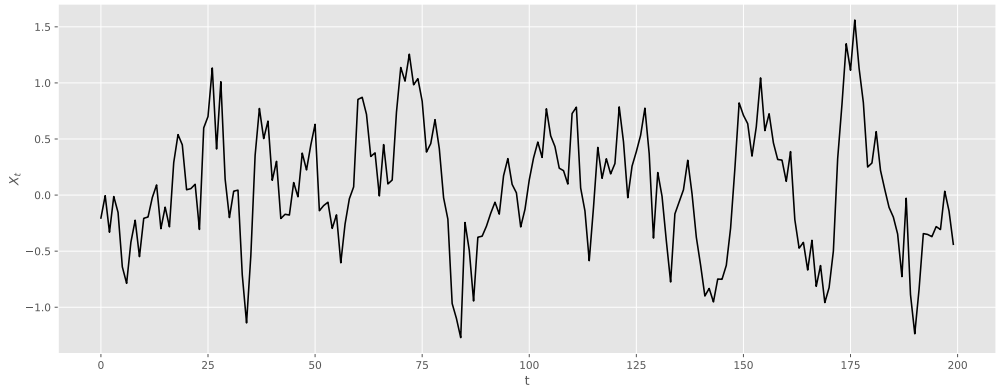


Figure 14.2 – Simulation du processus AR(1) à erreurs ARCH(1)

Nous observons sur la figure 14.1 et la figure 14.2 que la variabilité de l'erreur et du processus X_t n'est plus constante avec le temps. Elle connaît des phases de stabilité suivies de périodes de variance plus élevée.

La moyenne non conditionnelle est donnée par :

$$\mathbb{E}(\varepsilon_t) = \mathbb{E}\left[z_t \times \left(\omega + \alpha_1 \varepsilon_{t-1}^2\right)^{1/2}\right] = \mathbb{E}(z_t) \times \mathbb{E}\left[\left(\omega + \alpha_1 \varepsilon_{t-1}^2\right)^{1/2}\right] = 0$$

puisque par hypothèse nous avons $\mathbb{E}(z_t) = 0$. De même $\mathbb{E}(\varepsilon_t \varepsilon_{t-i}) = 0$ puisque $\mathbb{E}(z_t z_{t-i}) = 0$ pour $i \neq 0$.

La variance non conditionnelle de ε_t est donnée par :

$$\mathbb{E}(\varepsilon_t^2) = \mathbb{E}\left[z_t^2 \times \left(\omega + \alpha_1 \varepsilon_{t-1}^2\right)\right] = \mathbb{E}(z_t^2) \times \mathbb{E}\left[\left(\omega + \alpha_1 \varepsilon_{t-1}^2\right)\right]$$

or $\sigma_z^2 = 1$ et la variance de ε_t est égale à celle de ε_{t-i}. La variance (non conditionnelle) de ε_t est donc égale à

$$\mathrm{V}(\varepsilon_t) = \mathbb{E}(\varepsilon_t^2) = \frac{\omega}{1 - \alpha_1}$$

On constate que la moyenne et la variance non conditionnelle ne sont donc pas fonction du temps.

La moyenne conditionnelle est donnée par :

$$\mathbb{E}(\varepsilon_t | \mathcal{F}_{t-1}) = \mathbb{E}\left[z_t \times \left(\omega + \alpha_1 \varepsilon_{t-1}^2\right)^{1/2} | \mathcal{F}_{t-1}\right] = \mathbb{E}(z_t) \times \mathbb{E}\left[\left(\omega + \alpha_1 \varepsilon_{t-1}^2\right)^{1/2}\right] = 0$$

La variance conditionnelle est donnée par : $\mathrm{V}(\varepsilon_t | \mathcal{F}_{t-1}) = \mathbb{E}(\varepsilon_t^2 | \mathcal{F}_{t-1}) = \omega + \alpha_1 \varepsilon_{t-1}^2$ (puisque $\sigma_z^2 = 1$).

Ainsi dans une spécification de type $ARCH$ seule la variance conditionnelle dépend des erreurs passées.

Proposition 14.5

Si $(\varepsilon_t)_{t \in \mathbb{Z}}$ est un processus ARCH(1), les moments d'ordres impairs de ε_t sont nuls par symétrie. On démontre par ailleurs qu'avec $\omega > 0$ et $0 \leq \alpha_1 < 1$, alors les moments d'ordre $2r$ de ε_t existent si et seulement si :

$$\alpha_1^r \prod_{j=1}^{p} (2j - 1) < 1, \quad r \in \mathbb{N}^*$$

Ainsi, le moment d'ordre 2 existe si $\alpha_1 < 1$ et le moment d'ordre 4 existe si $3\alpha_1^2 < 1$, soit $\alpha_1 < 1/\sqrt{3}$, et on montre que

$$\begin{cases} \mathbb{E}(\varepsilon_t^2) = \mathrm{V}(\varepsilon_t) = \dfrac{\omega}{1 - \alpha_1} \\ \mathbb{E}(\varepsilon_t^4) = \dfrac{3\omega^2}{(1 - \alpha_1)^2} \dfrac{1 - \alpha_1^2}{1 - 3\alpha_1^2} \end{cases}$$

14.3.1.2 Kurtosis d'un processus ARCH

Le Kurtosis est le rapport du moment centré d'ordre 4 sur le carré du moment centré d'ordre 2. Pour un processus $ARCH(1)$, il vient :

$$k = \mathbb{E}\left[\left(\frac{\varepsilon_t - \mu}{\sigma}\right)^4\right] = \frac{\mathbb{E}\left(\varepsilon_t^4\right)}{\left[\mathbb{E}\left(\varepsilon_t^2\right)\right]^2} = 3\frac{1-\alpha_1^2}{1-3\alpha_1^2} > 3 \tag{14.25}$$

k est toujours supérieur à 3, un processus ARCH a donc une distribution leptokurtique qui permet de modéliser les phénomènes rares.

Ainsi, la distribution d'un processus ARCH a :

— Un skewness nul (moment centré d'ordre 3) : la distribution est donc symétrique ;
— Un kurtosis (moment d'ordre 4) supérieur à 3 : la distribution est leptokurtique (plus pointue que la gaussienne, les queues de distribution sont lourdes (plus épaisses que celles d'une gaussienne), ce qui permet de modéliser les phénomènes rares.

Cette propriété est très intéressante en pratique, dans la mesure où elle permet de détecter un *ARCH* à travers le comportement de sa distribution.

Exemple 14.2

La valeur du kurtosis empirique de la série est calculée en utilisant la commande suivante :

```
# Kurtosis
arch_kusto = np.mean((arch_resid-np.mean(arch_resid))**4/np.var(arch_resid,ddof=0)**2)
arch_kusto
```

```
## 2.71493531454782
```

14.3.2 Test de présence d'erreurs ARCH

Avant de chercher à ajuster un processus de type *ARCH* sur une série chronologique, il est important de détecter la présence d'hétéroscédasticité dans la série. Contrairement à une composante non linéaire longue mémoire, une composante hétéroscédastique dans une série n'est pas observable à l'aide de l'ACF ou la densité spectrale. Cependant plusieurs méthodes pour tester cette présence d'hétéroscédasticité existent.

14.3.2.1 Test de Portmanteau de non corrélation des résidus

Comme nous l'avons montré précédemment, un processus $(\varepsilon_t)_{t\in\mathbb{Z}}$ suivant un $ARCH(q)$ implique que $(\varepsilon_t^2)_{t\in\mathbb{Z}}$ suit un $AR(q)$.On peut alors s'intéresser à la série $(\varepsilon_t^2)_{t\in\mathbb{Z}}$ et utiliser l'approche de Box-Jenkins pour étudier la structure de corrélation de cette série afin d'identifier les propriétés d'un AR.

Considérons le processus

$$\varepsilon_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 + \eta_t \tag{14.26}$$

D'après la construction de cette équation, le processus $(\eta_t)_{t\in\mathbb{Z}}$ est centré, non corrélé de variance non constante.

Le test est pratiqué sur la série des résidus estimés $\hat{\eta}_t$ de $(\varepsilon_t^2)_{t\in\mathbb{Z}}$. La statistique du test est définie (*cf.* G. E. Box and Pierce (1970)) par :

$$BP_K = T \sum_{h=1}^{K} \hat{\rho}_{\eta_t}^2(h) \tag{14.27}$$

ou par la version améliorée (*cf.* Ljung and Box (1978))

$$LP_K = T(T+2) \sum_{h=1}^{K} \frac{\hat{\rho}_{\eta_t}^2(h)}{T-h} \tag{14.28}$$

$\hat{\rho}_{\eta_t}(h)$ est l'autocorrélation d'ordre h des résidus estimé sur ε_t^2 et K le nombre de retards inclus dans la sommation. T représente la longueur de la série.

Cette statistique BK_K, sous l'hypothèse de non corrélation des K premières valeurs de la série, suit asymptotiquement une distribution de χ^2 à $K-q$ degrés de liberté, où q est l'ordre de la partie AR. L'adéquation du modèle est rejetée au risque α si :

$$BP_K > \chi^2_{1-\alpha}(K-q)$$

Si $BP_K > \chi^2_{1-\alpha}(K-q)$, on rejette l'hypothèse nulle \$H_{0} : \$ les résidus $\hat{\eta}_t$ sont autocorrélés à l'ordre p, l'erreur ne peut alors être modélisée par un processus $ARCH(q)$.

14.3.2.2 Test du multiplicateur de Lagrange

Soit un modèle $\Phi(B)X_t = \Theta(B)\varepsilon_t$ avec une spécification de type $ARCH$ pour l'erreur ε_t telle que

$$\begin{cases} \varepsilon_t = \sigma_t z_t \\ \sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 = \omega + \alpha(B)\varepsilon_t \end{cases} \qquad \text{avec} \quad z_t \sim \mathcal{N}(0,1)$$

Les hypothèses du test sont :

$$\begin{cases} H_0 : \alpha_1 = \alpha_2 = \cdots = \alpha_q = 0 \\ H_1 : \text{il existe au moins un } i \text{ tel que } \alpha_i \neq 0 \end{cases}$$

Si l'hypothèse H_0 est acceptée, la variance de l'erreur est constante $\sigma_t^2 = \omega$. Dans le cas contraire les termes de l'erreur suivent un $ARCH$ dont l'ordre p est à déterminer.

Le test est fondé sur le test du multiplicateur de Lagrange (LM). De manière pratique, on procède de la manière suivante :

1. Calcul de $\hat{\varepsilon}_t$ le résidu du modèle $ARMA$;
2. Calcul des $\hat{\varepsilon}_t^2$
3. Régression autorégressive des résidus sur p retards (résidu décalé) ou seuls les retards significatifs sont conservés
$$\hat{\varepsilon}_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \hat{\varepsilon}_{t-i}^2$$
4. Calcul de la statistique du multiplicateur de Lagrange, $LM = T \times R^2$ où T est le nombre d'observations servant au calcul de la régression et R^2 le coefficient de détermination de l'étape 3.

Si $LM > \chi^2(q)$ à q degrés de liberté lu dans la table à un seuil α fixé (en général 0.05), on rejette H_0 ; on considère que le processus est justifiable d'un modèle $ARCH(q)$.

Sous Python, la fonction het_arch permet de réaliser un test du multiplicateur de Lagrange.

Remarque 14.2

Le test d'un modèle de type $ARCH$ peut aussi être fondé sur un test de Fisher classique. Soit $SSR_0 = \sum_{t=q+1}^{T} (\varepsilon_t^2 - \overline{\omega})$, où $\overline{\omega} = (1/T)\sum_{t=1}^{T} \varepsilon_t^2$ la moyenne empirique de ε_t^2 et $SSR_1 = \sum_{t=q+1}^{T} \hat{e}_t^2$, où $\hat{e}_t^2$ est le résidu issu de la régression linaire. Sous l'hypothèse nulle, la statistique

$$F = \frac{(SSR_0 - SSR_1)/q}{SSR_1/(T-2q-1)}$$

suit asymptotiquement une distribution de χ^2 à q degrés de liberté. On rejette H_0 si $F > \chi^2(q)$ à q degrés de liberté lu dans la table à un seuil α fixé (en général 0.05), ou encore que la p-value associée à F soit plus petite que α.

14.3.3 Estimation d'un processus ARCH

Le modèle ARCH peut être estimé par la méthode du maximum de vraisemblance, par la méthode des moments généralisés ou par la méthode du quasi-maximum de vraisemblance. Cependant, la technique du maximum de vraisemblance reste la plus utilisée. Trois fonctions de vraisemblance sont communément utilisées.

Sous l'hypothèse de normalité, la fonction de vraisemblance d'un modèle ARCH(q) s'écrit :

$$\begin{aligned} L(\varepsilon_1, \dots, \varepsilon_T|\theta) &= f(\varepsilon_T|\mathcal{F}_{T-1}) f(\varepsilon_{T-1}|\mathcal{F}_{T-2}) \cdots f(\varepsilon_{q+1}|\mathcal{F}_q) f(\varepsilon_1, \dots, \varepsilon_q|\theta) \\ &= \prod_{t=q+1}^{T} \frac{1}{\sqrt{2\pi\sigma_t^2}} \exp\left(-\frac{\varepsilon_t^2}{2\sigma_t^2}\right) f(\varepsilon_1, \dots, \varepsilon_q|\theta) \end{aligned}$$

où $\theta = (\omega, \alpha_1, \dots, \alpha_q)$ est le vecteur paramètre à estimer et $f(\varepsilon_1, \dots, \varepsilon_q|\theta)$ la densité jointe de $\varepsilon_1, \dots, \varepsilon_q$. Puisque la forme exacte de $f(\varepsilon_1, \dots, \varepsilon_q|\theta)$ est complexe, elle est souvent supprimée de la fonction de vraisemblance conditionnelle, spécialement lorsque la taille T est grande. Ainsi, la fonction de vraisemblance conditionnelle s'écrit :

$$L(\varepsilon_{q+1}, \dots, \varepsilon_T|\theta, \varepsilon_1, \dots, \varepsilon_q) = \prod_{t=q+1}^{T} \frac{1}{\sqrt{2\pi\sigma_t^2}} \exp\left(-\frac{\varepsilon_t^2}{2\sigma_t^2}\right)$$

où σ_t^2 peut être évaluée de façon récursive.

Maximiser la fonction de vraisemblance conditionnelle est équivalente à maximiser la log-vraisemblance, qui est facile à manipuler. La fonction de log-vraisemblance conditionnelle s'écrit :

$$\mathcal{L}(\varepsilon_{q+1}, \dots, \varepsilon_q|\theta, \varepsilon_1, \dots, \varepsilon_q) = \sum_{t=q+1}^{T} \left(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\sigma_t^2) - \frac{\varepsilon_t^2}{2\sigma_t^2}\right)$$

où $\sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2$ peut être évaluée de manière récursive.

La log-vraisemblance d'un processus $ARCH$ est donc une fonction des paramètres $\omega, \alpha_1, \dots, \alpha_q$. L'estimation par maximum de vraisemblance est la recherche des valeurs pour ces paramètres qui maximisent la log-vraisemblance.

$$\hat{\theta}_{MV} = \arg\max_{\theta} \mathcal{L}(\varepsilon_{q+1}, \dots, \varepsilon_T | \theta, \varepsilon_1, \dots, \varepsilon_q)$$

L'estimateur du maximum de vraisemblance θ sous l'hypothèse de normalité, satisfait le système non linéaire d'équations suivant (CPO) :

$$\frac{\partial \mathcal{L}(\varepsilon_{q+1}, \dots, \varepsilon_T | \theta, \varepsilon_1, \dots, \varepsilon_q)}{\partial \theta}|_{\theta=\hat{\theta}} = 0$$

L'estimateur θ n'a pas de formule explicite, et peut être obtenu en utilisant des méthodes d'optimisation numériques comme par exemple l'algorithme de Newton-Raphson.

L'estimateur θ vérifie

$$\sqrt{T}\left(\hat{\theta} - \theta\right) \xrightarrow{d} \mathcal{N}\left(0, I_T^{-1}(\theta)\right)$$

où I_T est la matrice d'information de Fisher.

Dans certaines applications, il est plus appropriée de considérer que z_t suit une distribution à queue lourde telle qu'une distribution t de Student. Soit X_ν une variable aléatoire suivant une distribution t de Student à ν degrés de liberté. Alors $\mathrm{V}(X_\nu) = \nu/\nu - 2$ pour $\nu > 2$, et on utilise $z_t = X_\nu / \sqrt{\nu/\nu - 2}$. La densité de probabilité de z_t est

$$f(z_t|\nu) = \frac{1}{\sqrt{(\nu-2)\pi}} \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{z_t^2}{\nu - 2}\right)^{-\frac{\nu+1}{2}}, \quad \nu > 2$$

où $\Gamma(.)$ est la fonction Gamma d'Euler (i.e. $\Gamma(t) = \displaystyle\int_0^\infty x^{t-1}e^{-x}\mathrm{d}x$).

```
# Densités loi N(0,1) et loi de Student
import scipy.stats as st
pas = np.linspace(-4,4,100)
normal = st.norm.pdf(pas)
student = st.t.pdf(pas,df=3)

# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(pas,normal,color='black',label="Normale N(0,1)");
axe.plot(pas,student, color='red',linestyle='--',label='Student(3)');
axe.set_ylabel('probabilité');
axe.set_xlabel('value');
axe.legend();
plt.show()
```

Utilisant la relation $\varepsilon_t = \sigma_t z_t$, on obtient la vraisemblance conditionnelle de ε_t suivante :

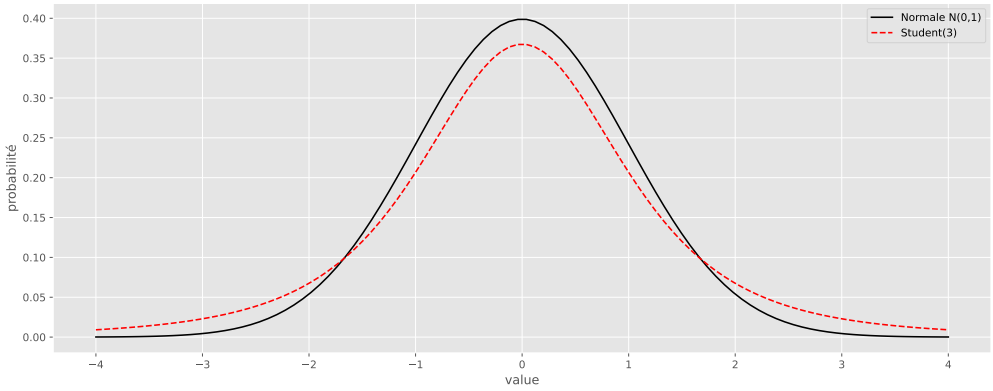


Figure 14.3 – Comparaison entre les distributions Normale et de Student

$$L(\varepsilon_{q+1},\dots,\varepsilon_T|\theta,\varepsilon_1,\dots,\varepsilon_q)=\prod_{t=q+1}^{T}\frac{1}{\sqrt{(\nu-2)\pi}}\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)}\frac{1}{\sigma_t}\left(1+\frac{\varepsilon_t^2}{(\nu-2)\sigma_t^2}\right)^{-\frac{\nu+1}{2}}$$

où $\nu>2$. Le degré de liberté ν peut être spécifié a priori ou estimé conjointement avec les autres paramètres. Une valeur entre 3 et 6 est souvent utilisée.

Si le degré de liberté ν de la distribution t de Student est déterminé au préalable, alors la log-vraisemblance conditionnelle associée s'écrit :

$$\mathcal{L}(\varepsilon_{q+1},\dots,\varepsilon_T|\theta,\varepsilon_1,\dots,\varepsilon_q)=-\sum_{t=m+1}^{T}\left[\frac{\nu+1}{2}\log\left(1+\frac{\varepsilon_t^2}{(\nu-2)\sigma_t^2}\right)+\frac{1}{2}\log(\sigma_t^2)\right] \tag{14.29}$$

Dans le cas où nous cherchons à estimer ν conjointement avec les autres paramètres, la log-vraisemblance conditionnelle devient :

$$\begin{aligned}\mathcal{L}(\varepsilon_{q+1},\dots,\varepsilon_T|\theta,\nu,\varepsilon_1,\dots,\varepsilon_q) &= (T-m)\left[\log\left(\Gamma\left(\frac{\nu+1}{2}\right)\right)-\log\left(\Gamma\left(\frac{\nu}{2}\right)\right)-\frac{1}{2}\log\left((\nu-2)\pi\right)\right]\\ &+ \mathcal{L}(\varepsilon_{q+1},\dots,\varepsilon_T|\theta,\varepsilon_1,\dots,\varepsilon_q)\end{aligned}$$

Une autre approche est de supposer que z_t suit une loi de Student dissymétrique standardisée de paramètre ξ, où ξ est le paramètre d'asymétrie permettant d'améliorer l'ajustement. Sous cette hypothèse, la log-vraisemblance conditionnelle s'écrit :

$$\begin{aligned}\mathcal{L}(\varepsilon_t|\theta) &= \log\Gamma\left(\frac{\nu+1}{\nu}\right)-\log\Gamma\left(\frac{\nu}{2}\right)+\log\left(\frac{2}{\xi+\frac{1}{\xi}}\right)+\log(s)\\ &- \frac{1}{2}\left[\log\pi(\nu-2)+\log(\sigma_t^2)+(1-\nu)\log\left(1+\frac{(sz_t+m)^2}{\nu-2}\xi^{-2I_t}\right)\right]\end{aligned}$$

avec

$$\begin{cases} m = \dfrac{\Gamma\left(\dfrac{\nu-1}{2}\right)\sqrt{\nu-2}}{\sqrt{\pi}\Gamma\left(\dfrac{\nu}{2}\right)}\left(\xi - \dfrac{1}{\xi}\right) \\ s^2 = \left(\xi^2 + \dfrac{1}{\xi^2} - 1\right) - m^2 \end{cases} \quad \text{et} \quad I_t = \begin{cases} 1 & \text{si } z_t \geq -\dfrac{m}{s} \\ 0 & \text{si } z_t \leq -\dfrac{m}{s} \end{cases}$$

Enfin, la dernière hypothèse est de supposer que z_t suit une distribution normale généralisée (*GED*, Generalized Error Distribution) de paramètre $\nu > 0$. Sa densité de probabilité est définie par :

$$f(z_t|\nu) = \frac{\nu}{\lambda\Gamma\left(\dfrac{1}{\nu}\right)} 2^{-\dfrac{\nu+1}{\nu}} \exp\left(-\frac{1}{2}\left|\frac{z_t}{\lambda}\right|^{\nu}\right), \quad x \in \mathbb{R}, \quad 0 < \nu \leq +\infty$$

où $\Gamma(.)$ désigne la fonction gamma et λ est une constante définie par

$$\lambda = \left[2^{-\frac{2}{\nu}} \frac{\Gamma\left(\dfrac{1}{\nu}\right)}{\Gamma\left(\dfrac{3}{\nu}\right)}\right]^{\frac{1}{2}} \tag{14.30}$$

Si $\nu = 2$, on retrouve une distribution gaussienne et dans le cas où $\nu < 2$, on a une distribution à queue lourde. Sous cette hypothèse, la log-vraisemblance associée à une observation z_t et à l'ensemble de paramètres θ s'écrit :

$$\mathcal{L}(\varepsilon_t|\theta) = \log\left(\frac{\nu}{\lambda}\right) - \frac{1}{2}\left|\frac{z_t}{\lambda}\right|^{\nu} - \left(1 + \frac{1}{\nu}\right)\log(2) - \log\Gamma\left(\frac{1}{\nu}\right) - \frac{1}{2}\log\left(\sigma_t^2\right) \tag{14.31}$$

14.3.4 Validation du modèle et prévision

La validation du modèle passe par l'estimation de la variance conditionnelle et l'estimation des résidus normalisés en vue d'approcher la loi conditionnelle du processus. En effet, on vérifie ici si l'hypothèse de bruit blanc gaussien des résidus normalisés est validée ou pas. Pour cela, on vérifie si les résidus normalisés (ε_t/σ_t) ne présentent pas de corrélation sérielle apparente sur l'ACF et le PACF empirique des résidus. Et on regarde aussi si leur densité de distribution est proche de celle de distribution théorique de la loi Normale à l'aide des graphes ou de manière formelle à l'aide du test d'adéquation de Kolmogorov.

14.3.4.1 Critères de choix des modèles

Avant d'examiner la possibilité de prévoir l'évolution future de la série considérée à partir des données passées, il convient dans un premier temps d'analyser et tester la puissance des modèles ARCH quand à leurs capacités à appréhender l'information historique, et prendre en compte cette dernière dans la détermination des données futures. Pour cela, on détermine les puissances de prévision des processus ARCH retenus et on les compare. La comparaison sera basée sur les critères de la racine de l'erreur quadratique moyenne (RMSE Root Squared Mean Error) et de la racine de la moyenne des erreurs de la moyenne des erreurs de prévision au carré (RMSFE) :

$$\begin{cases} RMSE = \dfrac{1}{T}\displaystyle\sum_{t=1}^{T} e_t^2 \\ RMSFE_\sigma = \dfrac{1}{N}\displaystyle\sum_{j=1}^{N} RMSFE_j \quad \text{avec} \quad RMSFE_j = \sqrt{\dfrac{1}{T}\displaystyle\sum_{t=1}^{T}\left(\hat{\sigma}_t^2 - \sigma_t^2\right)^2} \end{cases}$$

où $\hat{\sigma}_t^2$ est la variance conditionnelle estimée, σ_t^2 est la variance conditionnelle actuelle et N le nombre de réplications.

Le RMSE est très utile pour comparer plusieurs estimateurs, notamment lorsque l'un d'eux est biaisé. Si les deux estimateurs à comparer sont sans biais, l'estimateur le plus efficace est amplement celui qui a la variance la plus petite. Le RMSFE est une mesure de la volatilité de l'erreur de prévision.

14.3.4.2 Prévision de la volatilité

La prévision d'un modèle ARCH défini par l'équation (14.13) peut être obtenue de façon récursive comme dans le cas d'un processus AR. Considérons un modèle ARCH(q). A l'instant terminal T, on cherche à effectuer une prévision pour l'instant $T+h$ avec $h > 0$. Ainsi, à l'instant $h = 1$, la prévision s'écrit :

$$\sigma_T^2(1) = \omega + \alpha_1 \varepsilon_T^2 + \cdots + \alpha_q \varepsilon_{T+1-q}^2$$

A l'instant $h = 2$, la prévision est :

$$\sigma_T^2(2) = \omega + \alpha_1 \sigma_T^2(1) + \alpha_2 \varepsilon_T^2 + \cdots + \alpha_q \varepsilon_{T+2-q}^2$$

Ainsi, pour un horizon $h > 0$, la prévision s'écrit :

$$\sigma_T(h) = \omega + \sum_{i=1}^{q} \alpha_i \sigma_T^2(h-i)$$

avec $\sigma_T^2(h-i) = \varepsilon_{T+h-i}^2$ si $h - i \leq 0$.

Exemple 14.3 *Étude d'un processus ARCH*

Le jeu de données à partir duquel nous allons mettre en oeuvre l'étude d'un processus ARCH (*cf.* Bourbonnais and Terraza (2010)).

Soit un processus X_t, on demande d'en étudier les propriétés et d'estimer les paramètres du modèle par une méthode adéquate. Commençons par charger les données.

```
# Chargement des données
arch_dataset = pd.read_excel("./donnee/arch_dataset.xls",index_col=0)

# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(arch_dataset,color='black');
axe.set_xlabel('t');
axe.set_ylabel('$X_t$');
plt.show()
```

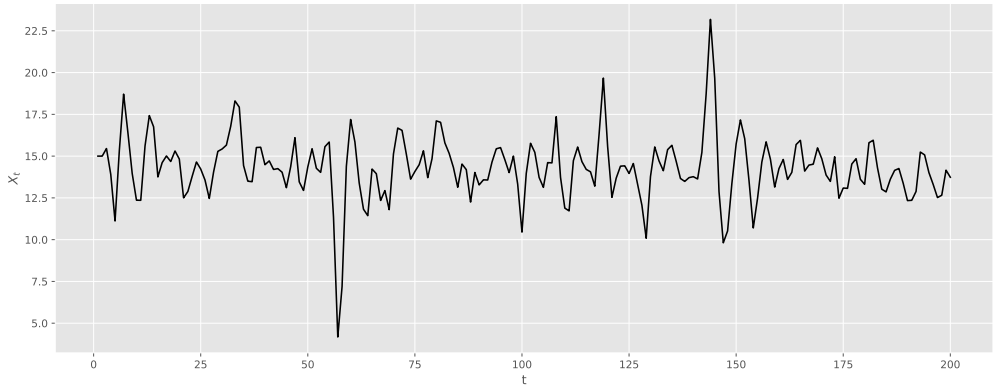


Figure 14.4 – Représentation du processus X_t

On fait l'étude du corrélogramme du processus.

```
# Autocorrélogramme simple et partiel
from statsmodels.graphics.tsaplots import plot_pacf, plot_acf
fig, axe = plt.subplots(1,2,figsize=(16,6))
plot_acf(arch_dataset, ax=axe[0]);
axe[0] = plot_colors(axe[0]);
axe[0].set_xlabel("Lag");
axe[0].set_ylabel("ACF");
axe[0].set_title("a) Autocorrélation simple");
plot_pacf(arch_dataset, ax=axe[1]);
axe[1] = plot_colors(axe[1]);
axe[1].set_xlabel("Lag");
axe[1].set_ylabel("PACF");
axe[1].set_title("b) Autocorrélation partielle");
plt.tight_layout()
plt.show()
```

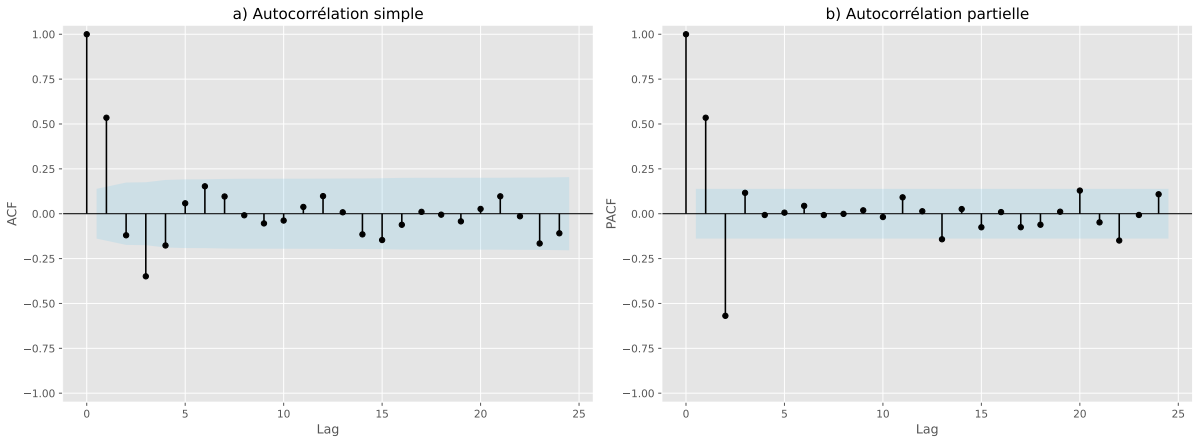


Figure 14.5 – Corrélogramme simple et partiel de X_t

L'examen du corrélogramme laisse présager d'un modèle autorégressif d'ordre 2 de type AR(2). L'estimation des paramètres conduit aux résultats suivants :

```
# Fonction retard
def lag(x, n):
    if n == 0:
        return x
    if isinstance(x, pd.Series):
        return x.shift(n)
    else:
        x = pd.Series(x)
        return x.shift(n)
    x = x.copy()
    x[n:] = x[0:-n]
    x[:n] = np.nan
    return x

# Estimation d'un AR(2)
import statsmodels.formula.api as smf
model_ar = smf.ols("Y~lag(Y,1)+lag(Y,2)",data = arch_dataset).fit()
model_ar_params = extractParams(model_ar, model_type = "arima")
```

Table 14.1 – Coefficients du modèle AR(2)

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	10.4251	0.8169	12.7614	1.648e-27	8.8139	12.0362
lag(Y, 1)	0.8395	0.0589	14.2575	4.579e-32	0.7233	0.9556
lag(Y, 2)	-0.5692	0.0589	-9.6711	2.548e-18	-0.6853	-0.4531

Toutes les p-values sont nulles, par conséquent ils sont significatifs. Le corrélogramme des résidus carrés est le suivant :

```
# Corrélogramme des résidus estimés au carré
squared_resid = np.square(model_ar.resid)
fig, axe = plt.subplots(1,2,figsize=(16,6))
plot_acf(squared_resid, ax=axe[0]);
axe[0] = plot_colors(axe[0]);
axe[0].set_xlabel("Lag");
axe[0].set_ylabel("ACF");
axe[0].set_title('a) Autocorrelation simple');
plot_pacf(squared_resid, ax=axe[1]);
axe[1] = plot_colors(axe[1]);
axe[1].set_xlabel("Lag");
axe[1].set_ylabel("PACF");
axe[1].set_title('b) Autocorrélation partielle');
plt.tight_layout()
plt.show()
```

Les premiers termes sont significativement différents de 0, on corrobore ce test par celui du multiplicateur de Lagrange. Tout d'abord, une effectuons une estimation d'un modèle AR(1) sur les résidus au carré. Puisqu'on effectuera le test du multiplicateur de Lagrange à plusieurs reprises, il est préférable pour nous de l'embrigarder dans une fonction. On crée la fonction Lmtest

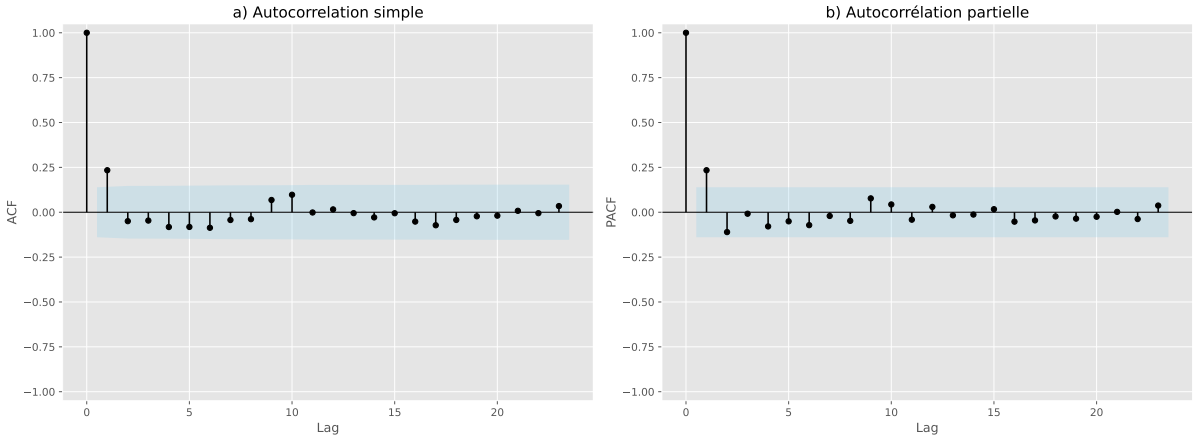


Figure 14.6 – Corrélogramme simple et partiel de e_t^2

```
# Création de la fonction LMtest
def Lmtest(x, arorder = 1,dll=1, approx = 4):
  import scipy.stats as st
  # ARIMA(p,d,q) model
  x = pd.DataFrame(x, columns = ["y"])
  formul = "y~"+"+".join(["lag(y,{})".format(i) for i in range(1,arorder+1)])
  model = smf.ols(formul,data = x).fit()
  params = extractParams(model,model_type="lm",approx = approx)
  # LM test
  stats = model.nobs*model.rsquared,
  pvalue = st.distributions.chi2.sf(stats,1)
  lmtest = pd.DataFrame({'r-squared':model.rsquared,'F-statistic': stats,
                         'p-value':pvalue},index=['ArchTest']).round(approx)
  return params,lmtest

# ARCH test : ARCH(1)
lmtest1_params,lmtest1_test = Lmtest(squared_resid,1)
```

Table 14.2 – Coefficients estimés sur le carré des résidus (modèle AR(1))

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.3749	0.3349	4.1054	5.931e-05	0.7144	2.0355
lag(y, 1)	0.2341	0.0696	3.3622	9.305e-04	0.0968	0.3714

Le test de spécification de type ARCH(1) conduit aux résultats :

Table 14.3 – LM test avec ARCH(1)

	r-squared	F-statistic	p-value
ArchTest	0.0548	10.7947	0.001

A la lecture des résultats, le F du Fisher empirique laisse supposer une spécification de type ARCH(1). La probabilité critique est inférieure à 0.05. Nous sommes donc amenés à rejeter l'hypothèse H_0 : F de nullité du α_1. Le test d'un ARCH(2) doit alors être effectuée :

```
# ARCH test : ARCH(2)
lmtest2_params, lmtest2_test = Lmtest(squared_resid,2)
```

Table 14.4 – Coefficients estimés sur le carré des résidus (modèle AR(2))

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.5311	0.3496	4.3795	1.949e-05	0.8415	2.2206
lag(y, 1)	0.2598	0.0715	3.6316	3.608e-04	0.1187	0.4009
lag(y, 2)	-0.1106	0.0716	-1.5449	1.240e-01	-0.2517	0.0306

Le test de spécification de type ARCH(2) conduit aux résultats :

Table 14.5 – LM test avec ARCH(2)

	r-squared	F-statistic	p-value
ArchTest	0.0663	12.9898	3e-04

Le coefficient de l'ordre 2 n'est pas significativement différent de 0, la spécification retenue est donc de type $ARCH(1)$. L'estimation des paramètres peut alors être effectuée à l'aide la méthode du maximum de vraisemblance.

```
# Estimation du modèle AR(2) - ARCH(1)
from arch.univariate import arch_model
ar_arch = arch_model(arch_dataset,mean='AR',lags=2,vol='arch', p=1,
                     dist="Normal").fit(disp = "off")
ar_arch_params = extractParams(ar_arch, model_type = "arch")
```

Table 14.6 – Coefficients du modèle AR(2) - ARCH(1) (Distribution normale)

	coef	std err	t	P>|t|	[0.025	0.975]
Const	10.6215	0.4690	22.6448	1.570e-113	9.7022	11.5408
Y[1]	0.7202	0.0707	10.1887	2.226e-24	0.5817	0.8587
Y[2]	-0.4659	0.0538	-8.6541	4.969e-18	-0.5715	-0.3604
omega	0.8335	0.1596	5.2220	1.770e-07	0.5207	1.1463
alpha[1]	0.5799	0.1848	3.1383	1.699e-03	0.2177	0.9421

Le modèle estimé s'écrit donc :

$$X_t = 10.62 + 0.72X_{t-1} - 0.47X_{t-2} + e_t$$

dont les erreurs théoriques suivent un ARCH(1) :

$$\begin{cases} \varepsilon_t = z_t\sigma_t \quad \text{avec} \quad z_t \sim \mathcal{N}(0,1) \\ \sigma_t^2 = \omega + \alpha_1\varepsilon_{t-1}^2 \end{cases}$$

La volatilité estimée est :

$$\hat{\sigma}_t^2 = 0.83 + 0.58e_{t-1}^2$$

Le module de la racine du polynôme opérateur retard est égal à $1.46 > 1$, le processus est stationnaire. La prévision à l'ordre 5 est donnée par :

```
# Prédiction à l'ordre 5
ar_arch_forecast = (ar_arch.forecast(horizon = 5, method = "simulation")
                    .variance[-1:]).round(4)
```

Table 14.7 – Prévision de la volatilité du modèle AR(2) - ARCH(1)

	h.1	h.2	h.3	h.4	h.5
200	1.6507	2.5594	2.7574	2.993	3.1084

La figure 14.7 donne une représentation de l'histogramme des résidus standardisés.

```
 # Histogramme des résidus standardisés
import seaborn as sns
ar_arch_std_resid = ar_arch.resid/ar_arch.conditional_volatility
fig, axe = plt.subplots(figsize=(16,6))
sns.distplot(ar_arch_std_resid,norm_hist=True,fit=st.norm, bins=50,ax=axe);
axe.legend(('Normal', 'Standardized residuals'));
plt.show()
```

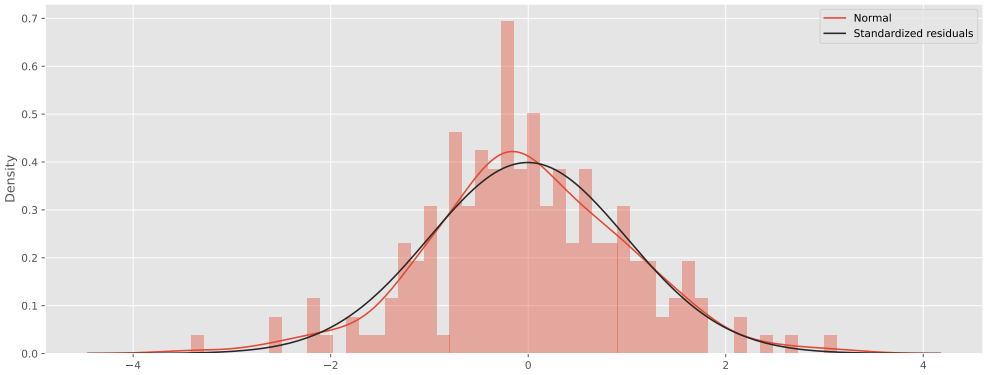


Figure 14.7 – Histogramme des résidus standardisés

On voir que la courbe des résidus standardisés est plus aplatie que celle de la loi normale, c'est-à-dire qu'elle suit une distribution platikurtique.

Nous allons à présent supposer que z_t suit une loi de Student dissymétrique standardisée. L'estimation du modèle nous donne les résultats suivants :

```
# Estimation du modèle AR(2) - ARCH(1) sous Student dissymétrique
ar_arch2 = arch_model(arch_dataset,mean='AR',lags=2,vol='arch', p=1,
                      dist="skewstudent").fit(disp = 'off')
ar_arch2_params = extractParams(ar_arch2, model_type = "arch")
```

le coefficient λ est non significatif. On refait l'estimation en supposant une loi de Student. Les résultats sont les suivants :

```
# Estimation du modèle AR(2) - ARCH(1) sous Student
ar_arch3 = arch_model(arch_dataset,mean='AR',lags=2,vol='arch', p=1,
                      dist="studentst").fit(disp = 'off')
ar_arch3_params = extractParams(ar_arch3, model_type = "arch")
```

Table 14.8 – Coefficients du modèle AR(2) - ARCH(1) (Distribution student dissymétrique)

	coef	std err	t	P>|t|	[0.025	0.975]
Const	10.6867	0.4827	22.1405	1.288e-108	9.7407	11.6327
Y[1]	0.7160	0.0718	9.9715	2.031e-23	0.5753	0.8568
Y[2]	-0.4662	0.0557	-8.3629	6.122e-17	-0.5754	-0.3569
omega	0.8471	0.1730	4.8960	9.782e-07	0.5080	1.1862
alpha[1]	0.5740	0.2112	2.7171	6.586e-03	0.1599	0.9880
eta	8.9794	5.1960	1.7281	8.396e-02	-1.2045	19.1633
lambda	0.0394	0.1031	0.3818	7.026e-01	-0.1627	0.2415

Table 14.9 – Coefficients du modèle AR(2) - ARCH(1) (Distribution student standard)

	coef	std err	t	P>|t|	[0.025	0.975]
Const	10.7069	0.4754	22.5211	2.579e-112	9.7751	11.6387
Y[1]	0.7109	0.0687	10.3408	4.605e-25	0.5762	0.8457
Y[2]	-0.4629	0.0542	-8.5406	1.335e-17	-0.5692	-0.3567
omega	0.8453	0.1713	4.9344	8.041e-07	0.5095	1.1811
alpha[1]	0.5725	0.2071	2.7651	5.691e-03	0.1667	0.9783
nu	9.2897	5.4913	1.6917	9.070e-02	-1.4730	20.0524

Tous les coefficients sont significatifs. Le degré de liberté estimé vaut 9.

14.4 Modèles ARCH généralisées : GARCH

Pour de nombreuses applications, l'introduction d'un grand nombre de retards q dans l'équation de la variance conditionnelle du modèle ARCH(q) est nécessaire pour tenir compte de la longue mémoire de la volatilité qui caractérise certaines séries monétaires et financières. Ce nombre important de paramètres peut poser des problèmes d'estimations. Dans cette perspective, une extension importante, le modèle autorégressif conditionnellement hétéroscédastique généralisé (GARCH), est suggérée par Bollerslev (1986). Cette approche exige moins de paramètres à estimer que la formulation ARCH(q) pour modéliser les phénomènes de persistance des chocs, il présente les mêmes propriétés et les mêmes propriétés et les mêmes fondements que le processus ARCH. La seule différence se situe au niveau de la définition telle que la variance conditionnelle de la variable étudiée et déterminée par le carré des q termes d'erreur passés (innovations) et des p variances conditionnelles retardées.

Définition 14.2 *Processus GARCH(p,q)*

Un processus ε_t satisfait une représentation GARCH(p,q) si

$$\begin{cases} \varepsilon_t = \sigma_t z_t \\ \sigma_t^2 = \omega + \sum_{i=1}^{i=q} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{j=p} \beta_j \sigma_{t-j}^2 = \omega + \alpha(B)\varepsilon_t^2 + \beta(B)\sigma_t^2 \end{cases} \tag{14.32}$$

avec $\omega > 0$ et $\alpha_i \geq 0, \beta_j \geq 0$ $\forall i$ $\forall j$ et où $(z_t)_{t\in\mathbb{Z}}$ est un bruit blanc gaussien centré de variance unitaire, indépendant du passé de $(\varepsilon_t)_t \in \mathbb{Z}$.

Classiquement, pour les processus GARCH rencontrés dans le domaine de la finance, la variable aléatoire z_t peut suivre, au lieu d'une loi gaussienne, une loi de Student à ν degré de liberté et de variance 1 qui présente des queues de distribution plus importantes que celles de la loi gaussienne ; la variable ε_t suivra une loi conditionnelle du même type.

Proposition 14.6 (Condition nécessaire et suffisante de stationnarité)

Un processus ε_t satisfaisant une représentation GARCH(p,q) telle que définie par (14.32) est stationnaire au second ordre si et seulement si :

$$\alpha(1) + \beta(1) = \sum_{i=1}^{q} \alpha_i + \sum_{j=1}^{p} \beta_j < 1 \tag{14.33}$$

On traduit ici simplement le fait que si les racines du polynôme :

$$1 - \sum_{i=1}^{\max(p,q)} (\alpha_i + \beta_i) B^i$$

sont à l'extérieur du disque unité, alors la série $\mathbb{E}(\varepsilon_t^2)$ converge et le processus devient asymptotiquement stationnaire à l'ordre 2.

Dans le cas où (14.33)) est saturée, i.e. $\sum_{i=1}^{q} \alpha_i + \sum_{j=1}^{p} \beta_j = 1$, on dira alors que le processus GARCH(p,q) est intégré, et on parlera de processus $IGARCH(p,q)$. Cette dénomination peut se justifier par l'existence d'une racine unité dans la composante autorégressive (14.32). Toutefois, cette analogie avec l'extension des modèles ARMA aux modèles $ARIMA$ peut être trompeuse : un processus ARIMA n'est pas stationnaire (au second ordre ou au sens fort) alors qu'il existe une solution stationnaire (au sens fort) d'un modèle IGARCH (qui admettra une variance infinie).

14.4.1 Propriétés d'un processus GARCH

Proposition 14.7

Si $(\varepsilon_t)_{t\in\mathbb{Z}}$ est un processus GARCH(p), donné par (14.32), alors on montre que :

1. Les moyennes non conditionnelle et conditionnelle de $(\varepsilon_t)_{t\in\mathbb{Z}}$ sont respectivement égales à :

$$\begin{cases} \mathbb{E}(\varepsilon_t) = 0 \\ \mathbb{E}(\varepsilon_t|\mathcal{F}_{t-1}) = 0 \end{cases}$$

2. Les variances non conditionnelle et conditionnelle de $(\varepsilon_t)_{t\in\mathbb{Z}}$ sont respectivement égales à :

$$\begin{cases} \mathrm{V}(\varepsilon_t) = \dfrac{\omega}{1 - \sum_{i=1}^{\max(p,q)} (\alpha_i + \beta_i)} & \text{si } \sum_{i=1}^{p} \alpha_i + \sum_{j=1}^{q} \beta_j < 1 \\ \mathrm{V}(\varepsilon_t|\mathcal{F}_{t-1}) = \mathbb{E}(\varepsilon_t^2|\mathcal{F}_{t-1}) = \sigma_t^2 \end{cases}$$

3. Les autocovariances non conditionnelles et conditionnelles de $(\varepsilon_t)_{t\in\mathbb{Z}}$ sont respectivement égale à :

$$\begin{cases} \mathrm{Cov}(\varepsilon_t, \varepsilon_{t+h}) = \sigma_h = 0, \quad \forall h > 0 \\ \mathrm{Cov}(\varepsilon_t, \varepsilon_{t+h}|\mathcal{F}_{t-1}) = 0 \end{cases}$$

14.4.1.1 Spécification des ordres d'un processus GARCH(p,q)

Dans la pratique, la spécification des ordres d'un processus GARCH(p,q) est un exercice difficile. Une propriété importante du processus GARCH est qu'il peut s'interpréter comme un processus ARMA sur le carré des innovations.

Considérons un processus GARCH(1,1). Soit ε_t un processus GARCH(1,1) tel que

$$\begin{cases} \varepsilon_t = z_t \sigma_t \\ \sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \end{cases}$$

En posant $\eta_t = \varepsilon_t^2 - \sigma_t^2 = \sigma_t^2 \left(z_t^2 - 1\right)$, on :

$$\varepsilon_t^2 = \omega + \left(\alpha_1 + \beta_1\right) \varepsilon_{t-1}^2 + \eta_t - \beta_1 \eta_{t-1}$$

Avec cette expression, le processus $(\varepsilon_t)_{t \in \mathbb{Z}}$ peut être vu comme un processus ARMA(1,1) entraîné par un bruit η_t.

Proposition 14.8

Si ε_t est un processus GARCH(p,q), alors ε_t^2, défini par :

$$\varepsilon_t^2 = \omega + \sum_{i=1}^{\max(p,q)} \left(\alpha_i + \beta_i\right) \varepsilon_{t-i}^2 + \eta_t - \sum_{j=1}^{p} \beta_j \eta_{t-j} \tag{14.34}$$

est un processus ARMA(m,q) de bruit $\eta_t = \sigma_t^2 \left(z_t^2 - 1\right)$, où $m = \max(p, q)$ avec $\alpha_i > 0$, $i > q$ et $\beta_j = 0$ pour $j >$.

Ainsi, pour identifier un $GARCH(p, q)$, on identifiera tout d'abord le processus $ARMA(m, q)$ qui modélise ε_t^2. Pour identifier p dans le cas où $m = q (p \leq q)$, il faut effectuer des tests de significativité des coefficients $\alpha_q, \ldots, \alpha_1$ du processus $ARMA(m, q)$.

La figure (14.8) donne la représentation d'une trajectoire de longueur $T = 1000$, issue d'un processus GARCH(1,1) en utilisant l'équation (14.32) avec $\omega = 0.1$, $\alpha_1 = 0.2$ et $\beta_1 = 0.2$.

```
# Simulation d'un processus GARCH(1,1)
def simulate_GARCH(n, omega, alpha, beta = 0):
    np.random.seed(1234)
    # Initialize the parameters
    white_noise = np.random.normal(size = n)
    resid = np.zeros_like(white_noise)
    variance = np.zeros_like(white_noise)
    for t in range(1, n):
        # Simulate the variance (sigma squared)
        variance[t] = omega + alpha * resid[t-1]**2 + beta * variance[t-1]
        # Simulate the residuals
        resid[t] = np.sqrt(variance[t]) * white_noise[t]
    return resid, variance

# Simulate a GARCH(1,1) series
garch_resid, garch_variance = simulate_GARCH(n=200,omega = 0.1, alpha = 0.3,
```

```
                                                beta = 0.2)
# Rprésentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(garch_resid,color='black',label='Residuals');
axe.plot(garch_variance, color='red',label='Variance');
axe.set_xlabel('t');
axe.legend();
plt.show()
```

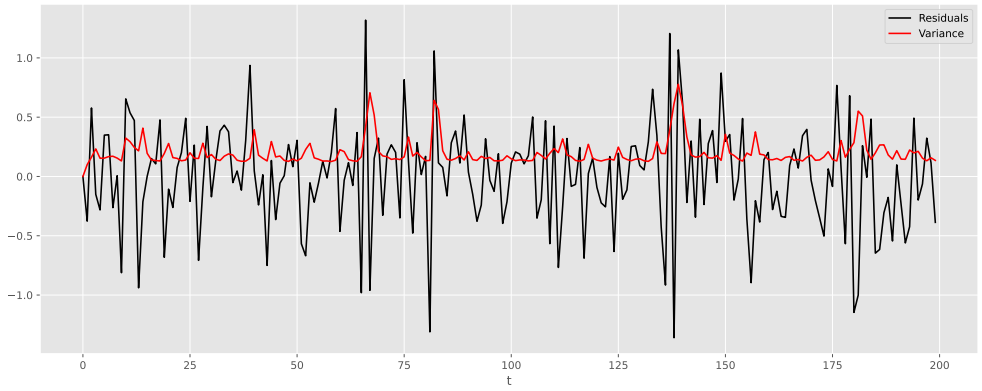


Figure 14.8 – Simulation d'un processus GARCH(1,1)

Dans le cas d'un processus GARCH(1,1), il est possible de montrer que le kurtosis de ε_t est de la forme :

$$\kappa(\varepsilon_t) = \frac{1-(\alpha_1+\beta_1)^2}{1-(\alpha_1+\beta_1)^2-\alpha_1^2(\mu_4-1)}\kappa(z_t) \quad \text{où } \mu_4 = \mathbb{E}(z_t^4)$$

Remarque 14.3

Empiriquement, l'identification des ordres p et q d'un processus GARCH s'effectue à partir de l'étude des autocorrélations et autocorrélations partielles de la série ε_t^2. Ainsi, pour un processus ARCH(q), la fonction d'autocorrélation partielle s'annule à partir du rang $q+1$. Pour un processus GARCH(p,q), cette fonction n'est en général pas nulle et décroît de façon exponentielle lorsque le nombre de retards augmente.

14.4.2 Test d'un modèle de type GARCH

Dans le cas d'une hétéroscédasticité conditionnelle supposée, on ne peut tester une spécification de type ARCH que contre une spécification de type GARCH. Le teste porte sur l'hypothèse nulle H_0 d'une erreur ARCH(q) contre l'hypothèse H_1 d'une erreur GARCH(p,q). On teste ainsi :

$$\begin{cases} H_0 : \beta_1 = \beta_2 = \cdots = \beta_p \\ H_1 : \text{il } \exists \text{ au moins un } j \in \{1, \ldots, p\} \text{ tel que } \beta_j \neq 0 \end{cases}$$

Le test du multiplicateur de Lagrange est celui qui convient le mieux : $T \times R^2 \sim \chi^2(q)$ (q = degré de liberté) où R^2 est le coefficient de détermination obtenu dans la régression par les MCO

dans l'équation $\sigma_t^2 = \hat{\omega} + \sum_{t=1}^{q} \hat{\alpha}_i \varepsilon_{t-i}^2 + \sum_{j=1}^{p} \hat{\beta}_j \sigma_{t-j}^2$.

Si $TR^2 > \chi^2(p)$ lu dans la table à un seuil de confiance (en général 0.05) et p degrés de liberté, alors on rejette l'hypothèse H_0. Les erreurs obéissent à un processus GARCH(p,q).

14.4.3 Prévision de la volatilité

La prédiction se fait de la même façon que pour les modèles de type ARMA. L'expression du prédicteur optimal n'est pas modifié par la présence d'hétéroscédasticité conditionnelle.

14.4.3.1 Cas d'un modèle AR(1)-GARCH(1,1)

Considérons un modèle AR(1) dont les erreurs suivent un modèle GARCH, i.e. un processus $(X_t)_{t\in\mathbb{Z}}$ vérifiant la relation suivante :

$$X_t = \phi_1 X_{t-1} + \varepsilon_t \quad |\phi_1| < 1$$

où

$$\begin{cases} \varepsilon_t = \sigma_t z_t \quad z_t \sim BB(0,1) \\ \sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \end{cases}$$

Le prédicteur optimal (c'est-à-dire à MSE minimum) est :

$$\hat{X}_t(h) = \mathbb{E}\left(X_{t+h}|I_t\right) = \sum_{i=0}^{\infty} \psi_{h+i} \varepsilon_{t-i} \quad \text{avec} \quad \psi_0 = 1 \tag{14.35}$$

On définit l'erreur de prévision $\hat{\varepsilon}_{t+h}$ par :

$$\hat{\varepsilon}_{t+h} = X_{t+h} - \hat{X}_t(h) = \sum_{i=0}^{h-1} \psi_h \varepsilon_{t+h-i} \quad \text{avec} \quad \psi_0 = 1 \tag{14.36}$$

Si l'on suppose que le bruit $(z_t)_{t\in\mathbb{Z}}$ est gaussien, $\hat{\varepsilon}_{t+h}$ obéit donc à une loi $\mathcal{N}\left(0, \sigma^2_{\hat{\varepsilon}_{t+h}}\right)$ où $\sigma^2_{\hat{\varepsilon}_{t+h}}$ est la variance de l'erreur de prévision donnée par :

$$\sigma^2_{\hat{\varepsilon}_{t+h}} = \mathrm{V}\left(\hat{\varepsilon}_{t+h}\right) = \mathbb{E}\left(\varepsilon^2_{t+h}|I_t\right) = \sum_{i=0}^{h-1} \psi_i^2 \mathbb{E}\left(\varepsilon^2_{t+h-i}|I_t\right) = \sum_{i=0}^{h-1} \psi_i^2 \mathbb{E}\left(\sigma^2_{t+h-i}|I_t\right) \tag{14.37}$$

avec

$$\mathbb{E}\left(\varepsilon^2_{t+h}|I_t\right) = \begin{cases} \omega + \alpha_1 \sigma_t^2 + \beta_1 \sigma_t^2 & \text{pour } h = 1 \\ \omega + (\alpha_1 + \beta_1)\, \mathbb{E}\left(\varepsilon^2_{t+h-1}|I_t\right) & \text{pour } h > 1 \end{cases}$$

On montre que si $\alpha_1 + \beta_1 < 1$, alors $\mathbb{E}\left(\varepsilon^2_{t+h}|I_t\right) = \omega + (\alpha_1 + \beta_1)\, \mathbb{E}\left(\varepsilon^2_{t+h-1}|I_t\right)$ tend vers

$$\mathbb{E}\left(\varepsilon_t^2\right) = \frac{\omega}{1 - \alpha_1 - \beta_1} \tag{14.38}$$

quand $h \longrightarrow +\infty$. En d'autres termes, le MSE conditionnel converge vers le MSE inconditionnel quand l'horizon de prévision k augmente. Cette propriété de retour vers la variance de long terme est due à la stationnarité du processus GARCH.

14.4.3.2 Intervalle de confiance de prévision

L'intervalle de confiance de la prévision, au seuil de $\alpha = 5\%$, est donné par

$$\left[\widehat{X}_t(h) \pm q_{1-\alpha/2}\sigma^2_{\widehat{\varepsilon}_{t+h}}\right] \tag{14.39}$$

où $\sigma^2_{\widehat{\varepsilon}_{t+h}}$ est la variance de l'erreur de prévision et $q_{1-\alpha/2}$ le quantile d'ordre $1-\alpha/2$ de la loi normale centrée réduite.

Exemple 14.4 *Étude d'un processus GARCH*

Le jeu de données à partir duquel nous allons mettre en oeuvre l'étude d'un processus GARCH (*cf.* Bourbonnais and Terraza (2010)).

Soit un processus X_t. On demande d'en étudier les propriétés et d'estimer les paramètres du modèle par une méthode adéquate.

```
# Chargement des données
garch_dataset = pd.read_excel("./donnee/garch_dataset.xls",index_col=0)
garch_dataset.index = pd.date_range('2000-01-01',periods=len(garch_dataset),
                                    freq = "M")
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(garch_dataset,color='black');
axe.set_xlabel('t');
axe.set_ylabel('$X_t$');
plt.show()
```

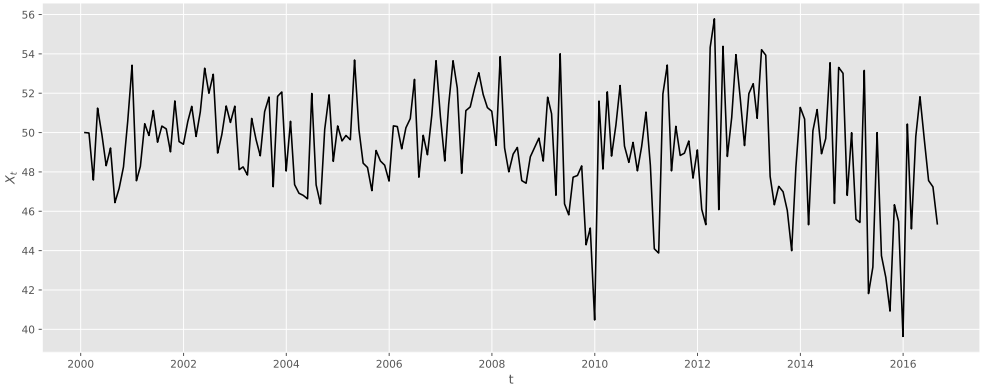


Figure 14.9 – Représentation du processus X_t

L'étude complète (tests de spécification, ...) de ce processus à partir de la méthodologie de Box et Jenkins laisse présager d'un ARMA(1,1). L'estimation des paramètres aux résultats suivants :

```
# Estimation d'un ARM(1,1)
from statsmodels.tsa.arima.model import ARIMA
model_arma = ARIMA(garch_dataset,order=(1,0,1),trend = "c").fit()
model_arma_params = extractParams(model_arma, model_type = "arima")
```

Table 14.10 – Coefficients du modèle ARMA(1,1)

	coef	std err	t	P>|t|	[0.025	0.975]
const	49.2577	0.4305	114.4262	0.000e+00	48.4140	50.1014
ar.L1	0.8412	0.0843	9.9784	1.895e-23	0.6760	1.0065
ma.L1	-0.6452	0.1180	-5.4676	4.561e-08	-0.8765	-0.4139
sigma2	6.6778	0.6744	9.9013	4.110e-23	5.3559	7.9997

Le corrélogramme des résidus est le suivant :

```
# Autocorrélogramme simple et partiel
fig, axe = plt.subplots(1,2,figsize=(16,6))
plot_acf(model_arma.resid, ax=axe[0]);
axe[0] = plot_colors(axe[0]);
axe[0].set_xlabel("Lag");
axe[0].set_ylabel("ACF");
axe[0].set_title('a) Autocorrelation');
plot_pacf(model_arma.resid, ax=axe[1]);
axe[1] = plot_colors(axe[1]);
axe[1].set_xlabel("Lag");
axe[1].set_ylabel("PACF");
axe[1].set_title('b) Partial Autocorrelation');
plt.tight_layout()
plt.show()
```

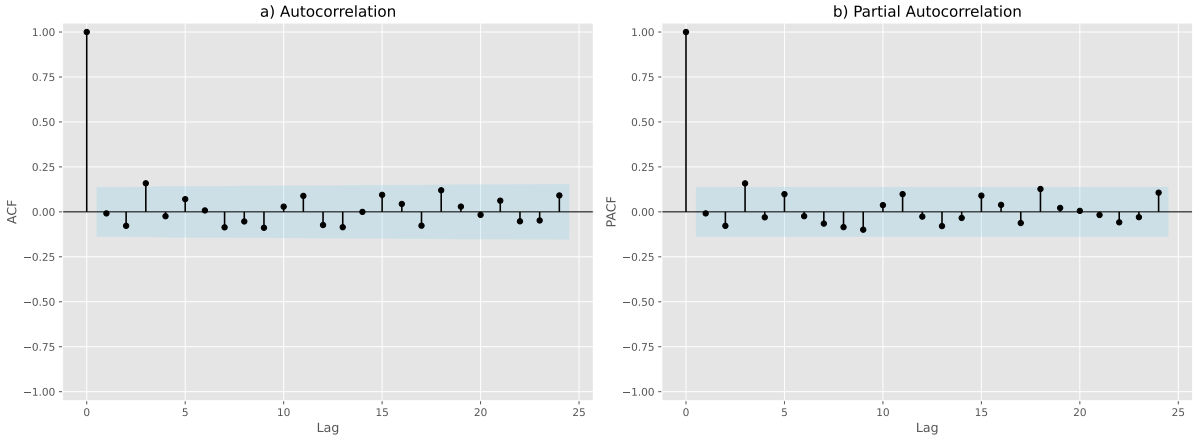


Figure 14.10 – Corrélogramme de e_t

On effectue un test de Ljung-Box sur les résidus estimés.

```
# Test de Ljung-Box
from statsmodels.stats.diagnostic import acorr_ljungbox
ljungbox1 = acorr_ljungbox(model_arma.resid, lags=range(1,11),return_df=True)
ljungbox1 = ljungbox1.reset_index().rename(columns={"index": "retards"})
```

Table 14.11 – Test de Ljung - Box sur les résidus du modèle ARMA(1,1)

retards	lb_stat	lb_pvalue
1	0.0150	0.9024
2	1.2468	0.5361
3	6.4168	0.0930
4	6.5393	0.1623
5	7.5727	0.1814
6	7.5857	0.2701
7	9.1388	0.2428
8	9.7391	0.2838
9	11.3982	0.2494
10	11.5763	0.3144

La statistique de Ljung-Box indique un corrélogramme dont les termes ne sont pas significativement différents de 0, les résidus sont dont non corrélés.

Le corrélogramme du carré des résidus est le suivant :

```
# Corrélogramme des résidus estimés au carré
squared_resid = np.square(model_arma.resid)
fig, axe = plt.subplots(1,2,figsize=(16,6))
plot_acf(squared_resid, ax=axe[0]);
axe[0] = plot_colors(axe[0]);
axe[0].set_xlabel("Lag");
axe[0].set_ylabel("ACF");
axe[0].set_title('a) Autocorrelation');
plot_pacf(squared_resid, ax=axe[1]);
axe[1] = plot_colors(axe[1]);
axe[1].set_xlabel("Lag");
axe[1].set_ylabel("PACF");
axe[1].set_title('b) Partial Autocorrelation');
plt.tight_layout();
plt.show()
```

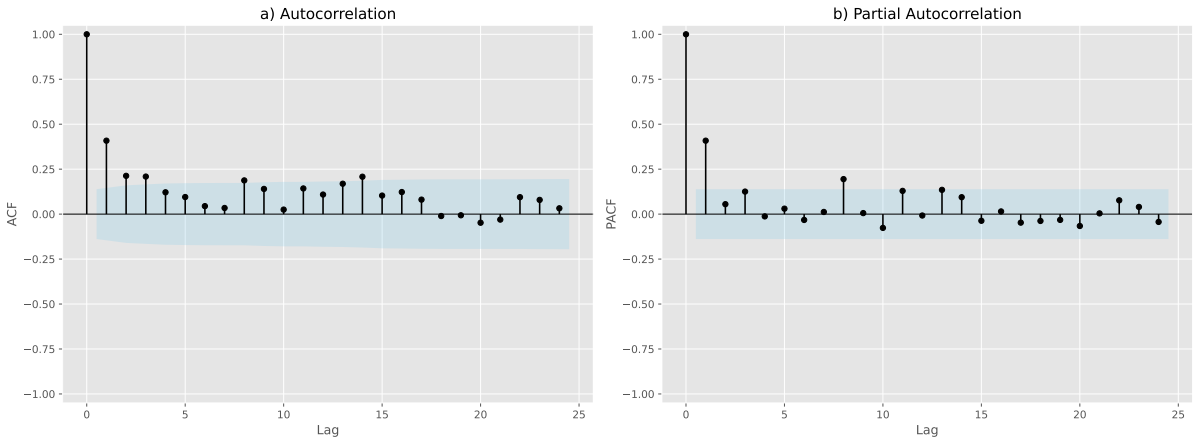


Figure 14.11 – Corrélogramme de e_t^2

On effectue un test de Ljung-Box sur les résidus estimés au carré.

```
# Test de Ljung-Box
ljungbox2 = acorr_ljungbox(squared_resid, lags=range(1,9),return_df=True)
ljungbox2 = ljungbox2.reset_index().rename(columns={"index": "retards"})
```

Table 14.12 – Test de Ljung - Box sur le carré des résidus du modèle ARMA(1,1)

retards	lb_stat	lb_pvalue
1	33.8564	0
2	43.0975	0
3	52.0360	0
4	55.0718	0
5	56.9402	0
6	57.3512	0
7	57.5942	0
8	64.9901	0

La statistique de Ljung-Box indique un corrélogramme dont les termes sont significativement différents de 0, une spécification de type ARCH est donc à retenir. Ceci est corroboré par la statistique du multiplicateur de Lagrange ($q = 1$).

```
# ARCH test : ARCH(1)
lmtest3_params, lmtest3_test = Lmtest(squared_resid,1)
```

Table 14.13 – Coefficients du modèle AR(1) sur le carré des résidus

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.9877	0.7567	5.2698	3.568e-07	2.4954	5.4800
lag(y, 1)	0.4087	0.0650	6.2902	2.008e-09	0.2806	0.5368

Table 14.14 – LM test sur ARCH(1)

	r-squared	F-statistic	p-value
ArchTest	0.1673	33.2831	0

Après différents essais de spécifications ARCH(1), ARCH(2) et GARCH(1,1), le modèle dont les coefficients sont tous significatifs s'avère être un GARCH(1,1). C'est donc cette dernière qui est retenue. On estime le modèle ARMA(1,1)-GARCH(1,1).

```
# Estimation du modèle GARCH(1,1) sur les résidus du modèle ARM(1,1)
garch_model = arch_model(model_arma.resid,mean='zero',vol='garch', p=1,o=0,q=1,
                         dist="Normal").fit(disp = "off")
garch_params = extractParams(garch_model, model_type = "arch")
```

Table 14.15 – Coefficients du modèle GARCH(1,1) sur les résidus du modèle ARMA(1,1)

	coef	std err	t	P>|t|	[0.025	0.975]
omega	1.2603	0.8174	1.5418	1.231e-01	-0.3418	2.8624
alpha[1]	0.2764	0.1021	2.7070	6.788e-03	0.0763	0.4766
beta[1]	0.5364	0.2081	2.5782	9.932e-03	0.1286	0.9442

Les coefficients des variables sont tous significatifs, le modèle estimé s'écrit donc :

$$X_t = 49.26 + 0.84X_{t-1} + e_t - 0.65e_{t-1} \tag{14.40}$$

dont les erreurs théoriques suivent un GARCH(1,1) :

$$\begin{cases} \varepsilon_t = z_t \sigma_t \quad \text{avec} \quad z_t \sim \mathcal{N}(0,1) \\ \sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \end{cases}$$

La variance estimée s'écrit :

$$\hat{\sigma}_t^2 = 1.26 + 0.28\varepsilon_{t-1}^2 + 0.54\sigma_{t-1}^2 \tag{14.41}$$

15

Les extensions du modèle GARCH

Sommaire

Les applications des modèles $ARCH(q)$ et $GARCH(p,q)$ en finance sont très nombreuses. Toutes, selon certains auteurs (*cf.* Nelson (1991), Cao and Tsay (1992)), les formulations ARCH et GARCH sont trop restrictives car elles imposent une relation quadratique entre l'erreur et la variance conditionnelle. Une telle formulation est appropriée si les variations analysées sont de même signe ou d'ampleurs équivalentes. Mais dès que l'on observe des mouvements de sens opposés, ou d'amplitudes très variables, ces modèles ne rendent plus compte de manière satisfaisante des variations constatées. Ainsi, selon Nelson (1991), les modèles GARCH peuvent se révéler insuffisants pour deux raisons principales :

1. Le choix d'une forme quadratique pour la variance conditionnelle a des conséquences importantes sur la trajectoire temporelle de la série. Typiquement, les séries sont caractérisées par des périodes de forte volatilité (correspondant aux fortes valeurs passées des erreurs, que que soit leur signe) et par des périodes où la volatilité est faible. Au travers des modèles ARCH, l'impact des valeurs passées de l'innovation sur la volatilité présente est donc prise en compte en fonction de leur ampleur. Or, dans le cas des séries macroéconomiques financières, la volatilité tend à être plus importante après une baisse qu'après une hausse (*cf.* Black (1976) et Zakoian (1990)). En d'autres termes, la volatilité tend à croître en réponse aux mauvaises nouvelles (rentabilités espérées plus faibles que prévues) et à décroître en réponse aux bonnes nouvelles (rentabilités attendues plus élevées que prévues). Le choix d'une forme (quadratique) symétrique pour la variance conditionnelle ne permet pas de modifier ce phénomène d'asymétrie.
2. Une autre restriction des modèles ARCH traditionnels provient des contraintes de non négativité imposées sur les paramètres. Comme l'ont montré Nelson et Cao (1991), ces hypothèses sont seulement suffisantes. Toutefois, les modèles GARCH restent fortement contraints à ce que la variance conditionnelle soit positive. Par conséquent, un choc, quel que soit son signe, a toujours un effet positif sur la volatilité courante : l'impact augmente avec l'ampleur du choc. Ceci rend alors le modèle inadéquat pour témoigner notamment des comportement cycliques oscillatoires de la volatilité.

15.1 Extensions des modèles GARCH

Ces critiques ont donnée lieu au développement de trois types de processus, à savoir : le modèle GARCH exponentiel noté EGARCH, le modèle GARCH à seuil noté TGARCH et le modèle GARCH quadratique noté QGARCH. Ces modèles (EGARCH, TGARCH, QGARCH, etc.) se distinguent des modèles précédents ((G)ARCH usuels) par le fait qu'ils rejettent l'hypothèse forte de symétrie liée à la représentation quadratique de la variance conditionnelle.

15.1.1 Processus EGARCH

15.1.1.1 Présentation générale

Il s'agit d'un modèle log-linéaire présenté par Nelson (1991) lors d'une étude sur les rentabilités des actifs financiers La spécification porte sur le logarithme de la variance conditionnelle et permet ainsi d'éviter les contraintes de positivités sur les coefficients α_i et β_j.

Définition 15.1 *Processus EGARCH(p,q)*

Un processus ε_t satisfait une représentation EGARCH(p,q) si et seulement si :

$$\begin{cases} \varepsilon_t = z_t \sigma_t \\ \ln \sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i g\left(z_{t-i}\right) + \sum_{j=1}^{p} \beta_j \ln \sigma_{t-j}^2 \end{cases} \tag{15.1}$$

où le résidu normalisé z_t est un bruit blanc faible et où la fonction $g(.)$ vérifie :

$$g\left(z_{t-i}\right) = \theta z_{t-i} + \gamma\left(|z_{t-i}| - \mathbb{E}|z_{t-i}|\right) \tag{15.2}$$

où θ et γ sont des constantes réelles. z_t et $\mathbb{E}(|z_t|)$ sont des variables aléatoires continues centrées indépendantes et identiquement distribuées. De plus, $\mathbb{E}\left[g\left(z_t\right)\right] = 0$.

Selon Nelson (1991), les contraintes de positivité sur les coefficients des modèles GARCH(p,q), d'une part, étaient souvent violées en pratique, et d'autre part, éliminaient toute possibilité de comportement cyclique. Cette critique n'est plus valable dans le cas des modèles GARCH exponentiels puisque les coefficients peuvent être positifs ou négatifs. En effet :

$$g(z_t) = \begin{cases} (\theta + \gamma) z_t - \gamma \mathbb{E}(|z_t|) & \text{si } z_t \geq 0 \\ (\theta - \gamma) z_t - \gamma \mathbb{E}(|z_t|) & \text{si } z_t < 0 \end{cases}$$

Ce qui implique que :

- Si $z_t \geq 0$, alors $g(z_t)$ est une fonction de pente $\theta + \gamma$.
- Si $z_t < 0$, la pente devient $\theta - \gamma$.

Ainsi, la variance conditionnelle répond de façon asymétrique au signe des innovations z_{t-i}. Les effets de signe et d'amplitude sont respectivement pris en compte par les coefficients ϕ et γ. $\mathbb{E}\left(|z_{t-i}|\right)$ dépend de la loi supposée de z_t. Le tableau (**??**) donne les valeurs de $\mathbb{E}\left(|z_t|\right)$ en fonction de la loi de z_t :

Remarque 15.1

Table 15.1 – Valeurs de $\mathbb{E}(|z_t|)$ en fonction de la loi de z_t

Loi	$\mathbb{E}(\lvert z_t\rvert)$
Gaussienne	$\sqrt{\frac{2}{\pi}}$
Student (ν)	$2\frac{\Gamma\left(\frac{\nu+1}{2}\right)\sqrt{\nu-2}}{\sqrt{\pi}(\nu-1)\Gamma\left(\frac{\nu}{2}\right)}$
Skew Student(ξ)	$\frac{4\xi^2\Gamma\left(\frac{\nu+1}{2}\right)\sqrt{\nu-2}}{\left(\xi+\frac{1}{\xi}\right)\sqrt{\pi}(\nu-1)\Gamma\left(\frac{\nu}{2}\right)}$
GED(ν)	$\frac{\Gamma\left(\frac{2}{\nu}\right)}{\sqrt{\Gamma\left(\frac{1}{\nu}\right)\Gamma\left(\frac{3}{\nu}\right)}}$

Comme l'indique Nelson (1991), le choix des variables « standardisées » z_{t-i} plutôt que ε_{t-i}, permet d'obtenir des conditions de stationnarité faible portant uniquement sur le polynôme $\beta(B)$.

Proposition 15.1 *Condition de stationnarité*

Supposons que $g(z_t)$ n'est pas presque partout nul et que les polynômes $\alpha(z) = \sum_{i=1}^{q} \alpha_i z^i$ et $\beta(z) = 1 - \sum_{i=1}^{p} \beta_i z^i$ n'ont pas des racines communes, et que $\alpha(z)$ n'est pas identiquement nul. Alors, le modèle $EGARCH(p,q)$ admet une solution strictement stationnaire si et seulement si les racines de $\beta(z)$ sont en dehors du cercle unitaire. Cette solution implique $\mathbb{E}\left[\left(\log(\varepsilon_t^2)\right)^2\right] < \infty$ lorsque $\mathbb{E}\left[\left(\log(z_t^2)\right)^2\right] < \infty$ et $\mathbb{E}\left[g^2(z_t)\right] < \infty$.

Proposition 15.2

Le modèle EGARCH(p,q) peut également être écrit de la façon suivante (*cf.* Nelson (1991)) :

$$\begin{cases} \varepsilon_t = \sigma_t z_t \\ \log(\sigma_t^2) = \omega + \dfrac{1 + \beta_1 B + \beta_2 B^2 + \cdots + \alpha_{q-1} B^{q-1}}{1 - \alpha_1 B - \alpha_2 B^2 - \cdots - \alpha_p B^p} g(z_{t-1}) \end{cases} \tag{15.3}$$

où ω est une constante réelle, B est l'opérateur retard vérifiant $Bg(z_t) = g(z_{t-1})$ et $1 + \beta_1 B + \beta_2 B^2 + \cdots + \alpha_{q-1} B^{q-1}$ et $1 - \alpha_1 B - \alpha_2 B^2 - \cdots - \alpha_p B^p$ sont des polynômes dont les racines sont à l'extérieur du cercle unité (on suppose également qu'ils n'ont pas de racine commune).

Dans le package arch de Python, la dynamique de la variance conditionnelle s'écrit sous le modèle $EGARCH(p,q)$ s'écrit :

$$\log \sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \left(|z_{t-i}| - \mathbb{E}|z_{t-i}|\right) + \sum_{j=1}^{o} \gamma_j z_{t-j} + \sum_{k=1}^{q} \beta_q \log \sigma_{t-k}^2 \tag{15.4}$$

où $z_t = \varepsilon_t / \sigma_t$ et $\mathbb{E}|z_t| = \sqrt{2/\pi}\ \forall t$.

D'une part, l'écriture porte sur le logarithme de la variance conditionnelle σ_t^2 de $\varepsilon_t \Rightarrow$ aucune restriction n'a besoin d'être imposée sur les différents paramètres de l'équation pour assurer la

positivité de σ_t^2. D'autre part, la variance conditionnelle σ_t^2 fait apparaître un effet de signe, $\gamma_j z_{t-j}$, et un effet d'amplitude mesuré par $\alpha_i \left[|z_{t-i}| - \mathbb{E}\left(|z_{t-i}|\right)\right]$.

Exemple 15.1 *Simulation d'un processus EGARCH(p,q)*

Considérons le processus EGARCH(1,1) définit par :

$$\begin{cases} r_t = 0.4 + \varepsilon_t \\ \varepsilon_t = \sigma_t z_t \\ \log \sigma_t^2 = 0.1 + 0.3\left(|z_{t-1}| - \mathbb{E}|z_{t-1}|\right) + 0.2 z_{t-1} + 0.4 \log \sigma_{t-1}^2 \end{cases} \tag{15.5}$$

avec $z_t \sim \mathcal{N}(0,1)$. Nous effectuons une simulation d'une trajectoire de 1000 observations.

```
# Simulation d'un EGARCH(1,1)
import numpy as np
from arch.univariate import arch_model

params = np.array([0.4, 0.1, 0.3, 0.2, 0.4])
model = arch_model(None,mean = "Constant",p=1, o=1, q=1, vol = "egarch",
                   dist="normal")
sim_egarch = model.simulate(params, 1000)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(sim_egarch.data,color = "black", label = "data");
axe.plot(sim_egarch.volatility, color = "red", label = "vol");
axe.set_xlabel("t");
axe.set_ylabel("valeur");
axe.legend();
plt.show()
```

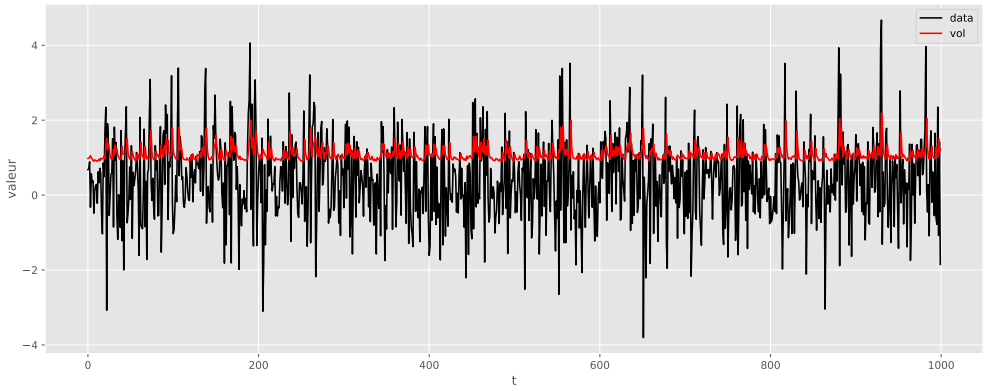


Figure 15.1 – Simulation d'un processus EGARCH(1,1)

Exemple 15.2 *Exemple illustratif*

Pour illustrer le modèle EGARCH(p,q), nous considérons la série des log rendements de l'indice IBM de Janvier 1926 à Décembre 2003 pour 936 observations. Tout d'abord, visualisons la série des données.

```
# Chargement des données
import pandas as pd
egarch_dataset_long = pd.read_csv("./donnee/ibm-stock-long.csv",sep=';',header=0,
                                  index_col=0,parse_dates=['Date'])
# Convert string to float
for name in egarch_dataset_long.columns:
    egarch_dataset_long[name] = pd.to_numeric(egarch_dataset_long[name],
                                              errors='coerce')

# Calcul du log rendement
egarch_dataset_long = egarch_dataset_long.apply(lambda x : 1+x,axis=1)
# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
egarch_dataset_long.ibm.plot(ax=axe,color='black');
axe.set_xlabel('Année');
axe.set_ylabel('IBM');
plt.show()
```

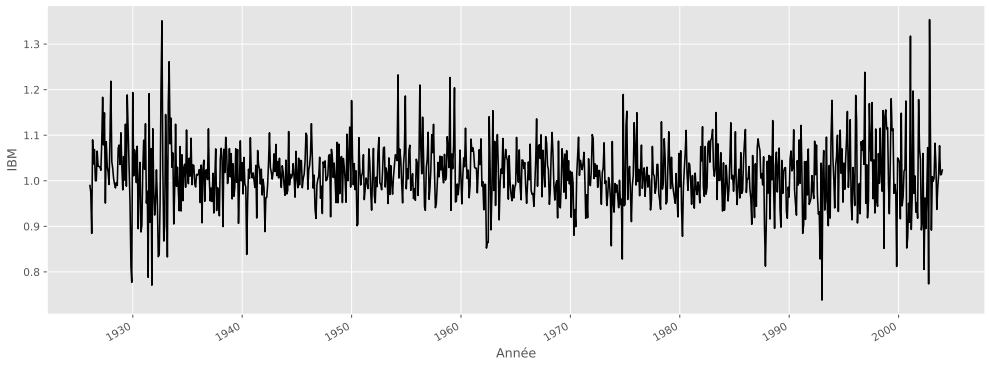


Figure 15.2 – IBM monthly log returns from 1926 to 2003

Un modèle AR(1)-EGARCH(1,1) est utilisé pour modéliser la série.

```
# Estimation d'un modèle AR(1) -EGARCH(1,1)
ar_egarch = arch_model(egarch_dataset_long.ibm,mean='AR',lags=1,vol='EGARCH',
                       p = 1 , o = 1 , q = 1,dist="NORMAL").fit(disp = 'off')
ar_egarch_params = extractParams(ar_egarch, model_type = "arch")
```

Table 15.2 – Coefficients du modèle AR(1) - EGARCH(1,1) (Distribution normale)

	coef	std err	t	P>|t|	[0.025	0.975]
Const	0.9520	0.0017	561.8990	0.000e+00	0.9486	0.9553
ibm[1]	0.0607	0.0002	244.7395	0.000e+00	0.0602	0.0612
omega	-0.2725	0.1713	-1.5914	1.115e-01	-0.6082	0.0631
alpha[1]	0.2020	0.0612	3.3013	9.625e-04	0.0821	0.3219
gamma[1]	-0.0400	0.0337	-1.1890	2.344e-01	-0.1060	0.0260
beta[1]	0.9487	0.0319	29.7180	4.501e-194	0.8862	1.0113

Dans l'estimation de l'équation de la volatilité, la constante ω et le coefficient γ sont non significatifs.

15.1.1.2 Prévision d'un modèle EGARCH

Pour illustrer la prévision d'un modèle EGARCH(p,q), nous considérons un modèle EGARCH(1,1) dont la forme donnée est donnée par l'équation (15.3) et où z_t suit une loi gaussienne. Pour notre modèle, nous avons le système suivant :

$$\begin{cases} \varepsilon_t = \sigma_t z_t \\ \log(\sigma_t^2) = (1-\alpha_1)\omega + \alpha_1 \log(\sigma_{t-1}^2) + g(z_{t-1}) \\ g(z_{t-1}) = \theta z_{t-1} + \gamma(|z_{t-1}| - \sqrt{2/\pi}) \end{cases}$$

En prenant l'exponentiel de l'équation de volatilité, on obtient le système suivant :

$$\begin{cases} \sigma_t^2 = \sigma_{t-1}^{2\alpha_1} \exp[(1-\alpha_1)\omega] \exp[g(z_{t-1})] \\ g(z_{t-1}) = \theta z_{t-1} + \gamma(|z_{t-1}| - \sqrt{2/\pi}) \end{cases}$$

A l'instant T, on souhaite effectuer la prévision à l'horizon $T+h$ avec $h > 0$. A l'instant $h = 1$, la volatilité estimée est donnée par :

$$\sigma_{T+1}^2 = \sigma_T^{2\alpha_1} \exp[(1-\alpha_1)\omega] \exp[g(z_T)]$$

où toute la quantité à droite est connue. La valeur prévue à l'instant $h = 1$ est donnée par $\hat{\sigma}_T^2(1) = \sigma_{T+1}^2$. Pour $h = 2$, la valeur estimée est :

$$\sigma_{T+2}^2 = \sigma_{T+1}^{2\alpha_1} \exp[(1-\alpha_1)\omega] \exp[g(z_{T+1})]$$

En prenant l'espérance conditionnelle, on a :

$$\hat{\sigma}_T^2(2) = \hat{\sigma}_T^{2\beta_1}(1) \exp[(1-\alpha_1)\omega] \mathbb{E}_T[\exp[g(z_{T+1})]]$$

où $\mathbb{E}_T$ désigne l'espérance mathématique pris à T (instant origine). On a :

$$\begin{aligned} \mathbb{E}_T\left[\exp(g(z))\right] &= \int_{-\infty}^{\infty} \exp\left(\theta z + \gamma(|z| - \sqrt{2/\pi})\right) f(z)\mathrm{d}z \\ &= \exp(-\gamma\sqrt{2/\pi}) \left[\int_0^{+\infty} e^{(\theta+\gamma)z} f(z)\mathrm{d}z + \int_{-\infty}^{0} e^{(\theta-\gamma)z} f(z)\mathrm{d}z\right] \\ &= \exp(-\gamma\sqrt{2/\pi}) \left[e^{(\theta+\gamma)^2/2}\Phi(\theta+\gamma) + e^{(\theta-\gamma)^2/2}\Phi(\gamma-\theta)\right] \end{aligned}$$

où $f(z)$ et $\Phi(x)$ représentent respectivement la fonction densité et la fonction de répartition d'une loi gaussienne. En conséquence, la volatilité prévue à l'instant $h = 2$ est définie par :

$$\hat{\sigma}_T^2(2) = \hat{\sigma}_T^{2\beta_1}(1) \exp((1-\alpha_1)\omega - \gamma\sqrt{2/\pi}) \left[e^{(\theta+\gamma)^2/2}\Phi(\theta+\gamma) + e^{(\theta-\gamma)^2/2}\Phi(\gamma-\theta)\right]$$

En répétant le processus précédent, on obtient de façon récursive la formule suivante, pour $h > 0$:

$$\hat{\sigma}_T^2(h) = \hat{\sigma}_T^{2\alpha_1}(h-1) \exp((1-\alpha_1)\omega - \gamma\sqrt{2/\pi}) \left[e^{(\theta+\gamma)^2/2}\Phi(\theta+\gamma) + e^{(\theta-\gamma)^2/2}\Phi(\gamma-\theta)\right] \quad (15.6)$$

Pour illustrer la précision, on considère le modèle AR(1)-EGARCH(1,1) estimé précédemment pour le log-rendement de l'action IBM. Nous effectuons la prévision.

```
# Forcatsing with AR(1)-EGARCH(1,1)
ar_egarch_forecast = ar_egarch.forecast(horizon=5, method = "simulation")
egarch_forecast_var = ar_egarch_forecast.variance[-1:]
```

Table 15.3 – Prévision de la variance conditionnelle à partir du modèle AR(1) - EGARCH(1,1)

	h.1	h.2	h.3	h.4	h.5
2003-12-31	0.0035267	0.0036253	0.0037212	0.003796	0.0038868

15.1.2 Processus TGARCH

Dans la formulation GARCH à seuil, introduite par Zakoian (1990) (*cf.* Zakoian (1994)), la forme quadratique (14.32) est remplacée par une fonction linéaire par morceaux - chacun des segments étant associé à des chocs de même « nature » -, ce qui permet d'obtenir différentes fonctions de volatilité selon le signe et les valeurs des chocs.

Définition 15.2 *Processus TGARCH(p,q)*

Un processus $(\varepsilon_t)_{t\in\mathbb{Z}}$ satisfait une représentation TGARCH(p,q) seulement si :

$$\begin{cases} \varepsilon_t &= z_t\sigma_t \\ \sigma_t &= \omega + \displaystyle\sum_{i=1}^{q} \alpha_i^+\varepsilon_{t-i}^+ - \sum_{i=1}^{q} \alpha_i^-\varepsilon_{t-i}^- + \sum_{j=1}^{p} \beta_j\sigma_{t-j} \\ &= \omega + \alpha^+(B)\varepsilon_t^+ - \alpha^-(B)\varepsilon_t^- + \beta(B)\sigma_t \end{cases} \tag{15.7}$$

où $\begin{cases} \varepsilon_t^+ = \max(\varepsilon_t, 0) \\ \varepsilon_t^- = \min(\varepsilon_t, 0) \end{cases}$ et le résidu normalisé z_t est un bruit blanc.

Dans la mesure où la spécification ne porte pas sur un carré mais sur l'écart-type conditionnel, il est possible de supprimer les contraintes de positivité des coefficients. La suppression de ces contraintes permet de prendre en compte les phénomènes d'asymétrie qui ont été précédemment décrit sur la volatilité. L'effet d'un choc ε_{t-i} sur la variance conditionnelle dépendra ainsi à la foi de l'amplitude et du signe de ce choc.

15.1.3 Processus QGARCH

Définition 15.3 *Processus QGARCH(p,q)*

Ces processus ont été introduits par Sentana (1995). Le processus $(\varepsilon_t)_{t\in\mathbb{Z}}$ satisfait une représentation $QGARCH(p,q)$ seulement si :

$$\begin{cases} \varepsilon_t = z_t\sigma_t \quad \text{avec} \quad z_t \sim \mathcal{N}(0,1) \\ \sigma_t^2 = \sigma_z^2 + \psi' Z_{t-q} + Z' A Z_{t-q} + \displaystyle\sum_{j=1}^{p} \beta_j\sigma_{t-j}^2 \end{cases} \tag{15.8}$$

avec Ψ est un vecteur de paramètres, σ_z^2 est la variance non conditionnelle de $z_t = \dfrac{\varepsilon_t}{\sigma_t}$ et $Z_{t-q} = \left(z_{t-1}, \dots, z_{t-q}\right)'$.

L'asymétrie est prise en considération par l'intermédiaire du terme linéaire. Par ailleurs, A est une matrice dont les termes en dehors de la diagonale principale tiennent compte des effets d'interaction des valeurs retardées de Z_t sur la variance conditionnelle.

Remarque 15.2

A partir de l'équation (15.8), on peut retrouver plusieurs types de processus :

- Lorsque $\phi = 0$, on obtient le processus GAARCH (Augmented GARCH) proposé par Bera et Lee (1990).
- Le modèle ARCH de Engle (1982) est tel que : $\phi = 0$, $\beta_j = 0$ et A est une matrice diagonale.
- Le modèle EGARCH est tel que A est une matrice diagonale.

L'étude de certaines variables financières fait en outre apparaître une relation entre la moyenne et la variance de la variable analysée. Les modèles présentés précédemment ne permettent pas de prendre en considération ce phénomène. Engle, Lilien et Robins (1937) ont proposé un modèle spécifique dans leur analyse sur la structure des taux d'intérêt : il s'agit du modèle ARCH en moyenne, noté ARCH-M (ARCH in Mean).

15.1.4 Processus GARCH-M

Les modèles de type ARCH-M sont certainement la catégorie des modèles ARCH la plus pertinente d'un point de vue économique. En effet, l'évaluation du risque constitue un point central de l'économie financière. Or, comme le notent Engle, Lilien, and Robins (1987), les méthodes usuelles de mesure et de prévision du risque sont souvent extrêmement simples et par conséquent inadaptées à l'analyse des séries temporelles financières. Il est raisonnable de penser que, puisque le degré d'incertitude concernant les rentabilités varie au cours du temps, la compensation que reçoivent les agents - présentant de l'aversion pour le risque - issue de la détention d'actions doit également varier au cours du temps. Ainsi, il convient non seulement de mesurer le risque, de tenir compte de sa variation au cours du temps, mais également d'inclure cette information comme un déterminant de la rentabilité du titre ou du portefeuille. La modélisation ARCH-M permet de tenir compte de ce phénomène en introduisant la variance conditionnelle dans l'équation de la moyenne.

Définition 15.4 *Processus GARCH-M*

Un modèle GARCH-M s'écrit (dans le cas où l'équation de la moyenne est un processus ARMA) :

$$\begin{cases} \Phi(B)X_t = \Theta(B)\varepsilon_t + \delta\sigma_t^2 \\ \sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i} + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2 \end{cases} \tag{15.9}$$

où X_t est un processus stationnaire, $\Phi(B)$ et $\Theta(B)$ sont les polynômes de retard autorégressif et moyenne mobile respectivement.

Une variation de la variance conditionnelle sera donc accompagnée d'une variation de la moyenne conditionnelle de X_t.

Remarque 15.3

L'estimation convergente d'un modèle ARCH-M nécessite une bonne spécification de σ_t^2, alors que pour un modèle GARCH(p,q), des estimations convergents des paramètres peuvent être obtenus même lorsque la spécification de σ_t^2 est mauvaise. Pagan et Ullah (1988) ont montré que toute erreur de spécification dans l'équation de la variance entraîne des biais et une éventuelle non convergence des estimateurs des paramètres de l'équation de la moyenne [1].

Remarquons que si l'on adopte pour l'équation de la moyenne une formulation du type :

$$R_t = a + b\sigma_t^2 + \varepsilon_t \tag{15.10}$$

où R_t désigne la rentabilité d'une action ou d'un portefeuille et b correspond au coefficient relatif d'aversion pour le risque, on trouve bien les modèles d'évaluation traditionnels dans lesquels la rentabilité est fonction linéaire du risque σ_t^2.

Remarque 15.4

Il est bien sûr possible de définir des processus ARCH-M, EGARCH-M, TGARCH-M, etc. Il suffit de reprendre l'équation de la variance de ces processus et d'introduire la variance conditionnelle (ou l'écart type conditionnel)comme variable explicative dans l'équation de la moyenne.

15.1.5 Processus IGARCH

Le modèle IGARCH (Integrated GARCH) a été introduit par Engle et Bollerslev (1986) et permet de prendre en compte l'existence d'une racine unitaire dans la variance. Ainsi, il y aura un phénomène de persistance dans la variance lorsqu'il existe une racine unitaire dans le polynôme $\alpha(B) + \beta(B)$, soit :

$$\alpha(B) + \beta(B) = 1 \quad \Longleftrightarrow \quad \alpha_1 + \cdots + \alpha_q + \beta_1 + \cdots + \beta_p = 1 \tag{15.11}$$

La présence d'une racine unitaire dans la variance renvoie au phénomène de mémoire infinie : lorsque se produit un choc sur la variance conditionnelle, celui-ci se répercute sur les prévisions de toutes ses valeurs futures.

Pour illustrer la prévision dans un modèle IGARCH(p,q), considérons le modèle IGARCH(1,1) défini comme suit :

$$\begin{cases} \varepsilon_t = \sigma_t z_t \\ \sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \end{cases} \tag{15.12}$$

où $\omega > 0$, $\alpha_1, \beta_1 \geq 0$ avec $\alpha_1 + \beta_1 = 1$. La prévision de la variance conditionnelle à différents horizons h est donnée par :

$$\hat{\sigma}_T^2(h) = \mathbb{E}\left[\sigma_{T+h}^2 | I_t\right] = (\alpha_1 + \beta_1)^h \sigma_T^2 + \omega \sum_{i=1}^{h-1} (\alpha_1 + \beta_1)\, i \tag{15.13}$$

Ainsi, comme $\alpha_1 + \beta_1 = 1$, cette variance conditionnelle prévisionnelle vaut :

$$\hat{\sigma}_T^2(h) = \sigma_T^2 + \omega h \tag{15.14}$$

Par conséquent, en présence d'un terme constant, cette variance diverge avec h.

1. Cette remarque s'applique également pour les processus GARCH exponentiels.

15.1.6 Processus GJR-GARCH

15.1.6.1 Présentation générale

Considérons la série temporelle des rendements $r_t = \mu + \varepsilon_t$, où μ est le rendement espéré et ε_t un bruit blanc centré. Le modèle Glosten - Jagannathan - Runkle GARCH (GJR-GARCH) assume une paramétrisation spécifique de l'hétéroscédasticité conditionnelle.

Définition 15.5 *Processus GJR - GARCH*

Le processus ε_t satisfait une représentation GJR-GARCH si et seulement si :

$$\begin{cases} \varepsilon_t = \sigma_t z_t \\ \sigma_t^2 = \omega + \sum_{i=1}^{q} (\alpha_i + \gamma_i I_{t-i})\, \varepsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2 \end{cases} \tag{15.15}$$

où I_{t-i} est une fonction indicatrice définie comme suit :

$$I_{t-i} = \begin{cases} 1 & \text{si } \varepsilon_{t-i} < 0 \\ 0 & \text{si } \varepsilon_{t-i} \geq 0 \end{cases}$$

et α_i, γ_i et β_j des paramètres non négatifs satisfaisant la condition comme dans un modèle GARCH.

Nous illustrons le modèle GJR-GARCH, on considère le log rendement de l'action IBM de 1926 à 2003. Nous estimons un modèle GJR-GARCH(1,1) en supposant que z_t suit une loi GED.

```
# Estimation d'un modèle GJR-GARCH(1,1)
gjr_garch = arch_model(egarch_dataset_long.ibm.values, p=1, o=1, q=1,
                       dist='GED').fit(disp='off')
gjr_garch_params = extractParams(gjr_garch, model_type = "arch")
```

Table 15.4 – Coefficients du modèle GJR - GARCH(1,1) (Distribution GED)

	coef	std err	t	P>|t|	[0.025	0.975]
mu	1.0130	0.0021	479.3173	0.000e+00	1.0088	1.0171
omega	0.0004	0.0002	2.2364	2.532e-02	0.0000	0.0007
alpha[1]	0.0630	0.0224	2.8205	4.795e-03	0.0192	0.1068
gamma[1]	0.0988	0.0577	1.7112	8.705e-02	-0.0144	0.2120
beta[1]	0.8181	0.0526	15.5570	1.425e-54	0.7150	0.9212
nu	1.4995	0.0915	16.3908	2.226e-60	1.3202	1.6788

Le modèle estimé s'écrit :

$$\begin{cases} r_t = 1.0130 + \varepsilon_t \\ \sigma_t^2 = 3.5573 \times 10^{-4} + (0.0630 + 0.0988 I_{t-1})\, \varepsilon_{t-1}^2 + 0.8181 \sigma_t^2 \end{cases}$$

où le paramètre estimé de la GED vaut $\nu = 1.4995$.

15.1.6.2 Prévision d'un processus GJR-GARCH

pour illustrer la prévision d'un processus GJR-GARCH, considérons le processus GJR-GARCH(1,1) défini comme suit :

$$\begin{cases} \varepsilon_t = \sigma_t z_t \\ \sigma_t^2 = \omega + (\alpha_1 + \gamma_1 I_{t-1})\,\varepsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \end{cases}$$

où $I_{t-1} = \begin{cases} 1 & \text{si } \varepsilon_{t-1} < 0 \\ 0 & \text{si } \varepsilon_{t-1} \geq 0 \end{cases}$

La prévision de la variance conditionnelle à l'horizon h en partant de T est donnée par :

$$\hat{\sigma}_T^2(h) = \hat{\omega} + \left(\hat{\alpha}_1 + \frac{\hat{\gamma}_1}{2} + \hat{\beta}\right)\hat{\sigma}_T^2(h-1)$$

Notons que pour de grandes valeurs de h, cette valeur prévue de la volatilité converge vers $\sqrt{h}\sqrt{\dfrac{\hat{\omega}}{1-\hat{\alpha}_1 - \dfrac{\widehat{\gamma_1}}{2} - \hat{\beta}_1}}$.

15.2 Modélisation avec Scipy

Dans cette section, nous allons implémenter des algorithmes d'estimation des paramètres de quelques processus non linéaire en variance en utilisant la librairie scipy. Dans le cadre de cette modélisation, nous travaillerons avec l'action S&P 500 journalier allant du 1^{er} Janvier 2010 au 31 Janvier 2022. Rappelons qu'il s'agit uniquement d'un exemple illustratif, par conséquent les résultats obtenus doivent être pris avec un esprit critique.

Avant de commencer, importons d'abord notre base de données.

```
# Chargement des données
import datetime; import yfinance as yf
stocks = '^GSPC'
start, end = datetime.datetime(2010, 1, 1), datetime.datetime(2022, 2, 1)
s_p500 = yf.download(stocks,start=start,end = end, interval='1d',progress=False)
```

Nous nous concentrerons sur le cours de fermeture ajusté (Adj Close). Visualisons la série.

```
# Visualisation de la série
import matplotlib.pyplot as plt
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(s_p500['Adj Close'], color='black');
axe.set_xlabel('Time');
axe.set_ylabel('Adj Close');
plt.show()
```

Dans l'étude des séries macroéconomiques financières, il est préférable de modéliser le rendement de cette série.

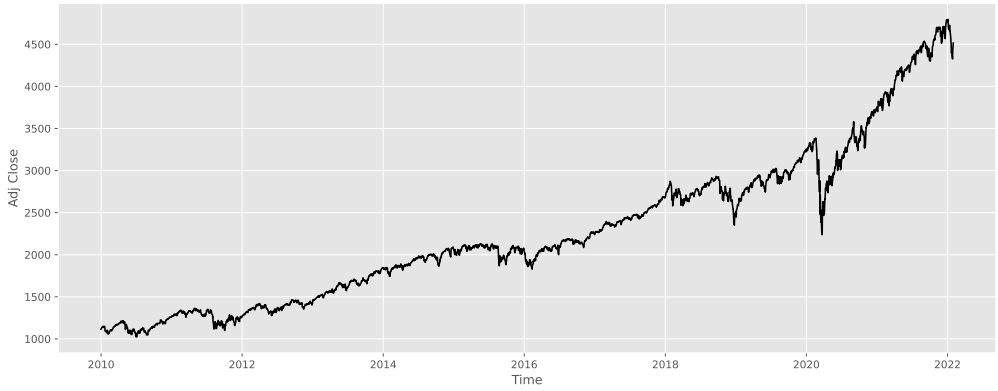


Figure 15.3 – Adj Close SP-500

```
# Rendement
ret = 100 * (s_p500.pct_change()[1:]['Adj Close'])

# Représentation graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot(ret, color='black');
axe.set_ylabel('Daily returns');
axe.set_xlabel('Date');
plt.show()
```

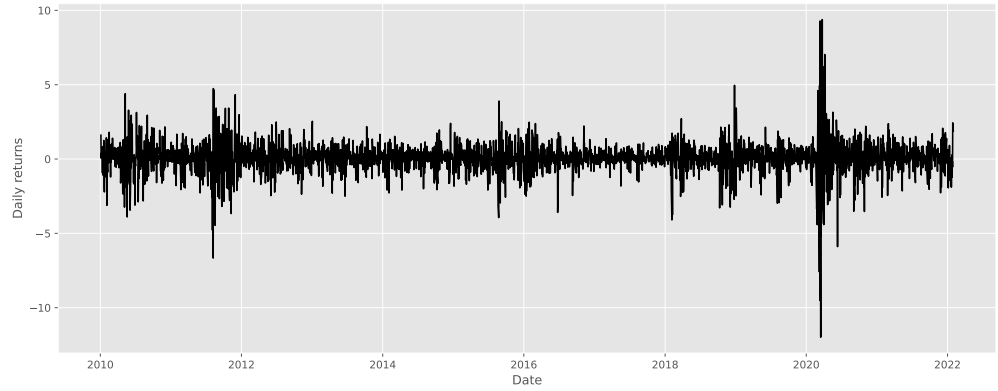


Figure 15.4 – Daily return of SP-500

15.2.1 Modélisation ARCH

On rappelle que si un processus ε_t suit un modèle ARCH(q) alors on a :

$$\begin{cases} \varepsilon_t = \sigma_t z_t \\ \sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i} \end{cases} \qquad z_t \sim BB(0, \sigma_z^2)$$

Si l'on suppose que le bruit blanc z_t est gaussien, la fonction de log-vraisemblance conditionnelle s'écrit :

$$\mathcal{L}(\varepsilon_{q+1}, \ldots, \varepsilon_q | \theta, \varepsilon_1, \ldots, \varepsilon_q) = \sum_{t=q+1}^{T} \left(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\sigma_t^2) - \frac{\varepsilon_t^2}{2\sigma_t^2} \right)$$

où $\sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2$ peut être évaluée de manière récursive.

Pour la modélisation du rendement de l'action S&P-500, on supposera qu'il suit un processus ARCH(1). Nous créons la fonction de vraisemblance associée :

```
# log-vraisemblance du modèle ARCH(1)
import numpy as np

def arch_likelihood(init_params, data):
    omega = abs(init_params[0]);alpha = abs(init_params[1])
    T = len(data); logliks = 0; sigma2 = np.zeros(T); sigma2[0] = np.var(data)
    for t in range(1, T):
        sigma2[t] = omega + alpha * (data[t - 1]) ** 2
    logliks = np.sum(0.5 * (np.log(sigma2)+data ** 2 / sigma2))
    return logliks
```

On initialise nos paramètres : $\omega_0 = 0.1$ et $\alpha_0 = 0.3$. On calcule la vraisemblance associée :

```
# Test de la fonction de vraisemblance
params_init_arch = [0.1,0.3]
logliks = arch_likelihood(params_init_arch, ret.values)
print('La log-vraisemblance est {:.4f}.'.format(logliks))
```

```
## La log-vraisemblance est 4694.8541.
```

L'objectif est de trouver les valeurs $\hat{\omega}$ et $\hat{\alpha}_1$ qui minimisent cette log-vraisemblance. On définit une fonction de minimisation :

```
# Fonction d'optimisation
from scipy.optimize import minimize

def opt_params(function, x0, data):
    opt_result = minimize(function, x0=x0, args = (data),
                          method='Nelder-Mead', options={'maxiter': 5000})
    params = opt_result.x
    def left_adjust(val, length = 6): return str(val).ljust(length)
    def right_adjust(val, length = 6): return str(val).rjust(length)
    print('=='*11)
    print(' '*16,'coef\n', '--'*10)
    for coef, value in zip(['omega','alpha[1]'], params):
        print(left_adjust(coef,8), '      ', right_adjust(round(value,4)))
    return params
```

```
# Estimation du ARCH(1,1)
params_arch = opt_params(arch_likelihood, params_init_arch, ret.values)
```

```
## ======================
##                    coef
##  --------------------
## omega           0.6985
## alpha[1]        0.3831
```

Nous avons modélisé le rendement à l'aide de notre fonction d'optimisation. Comparons nos résultats avec ceux fournis par la fonction arch_model du package arch.

```
# Comparaison avec le package arch
arch1 = arch_model(ret,mean='zero',vol='ARCH',p=1).fit(disp='off')
arch1_params = extractParams(arch1,model_type = "arch")
```

Table 15.5 – Coefficients du modèle ARCH

	coef	std err	t	P>|t|	[0.025	0.975]
omega	0.6985	0.0483	14.4522	2.426e-47	0.6038	0.7933
alpha[1]	0.3837	0.0677	5.6683	1.442e-08	0.2510	0.5164

Maintenant nous devons chercher la meilleure valeur de q du modèle ARCH suivant le critère BIC en itérant de 1 à 5 :

```
# Choix de q
bic_arch =list()
for p in range(1,5):
    arch = arch_model(ret,mean='zero',vol='ARCH', p=p).fit(disp='off')
    bic_arch.append(arch.bic)
    if arch.bic == np.min(bic_arch):
        best_p = p
# Application
arch2 = arch_model(ret, mean='zero',vol='ARCH', p=best_p).fit(disp='off')
arch2_params = extractParams(arch2,model_type = "arch")
```

Table 15.6 – Coefficients du modèle ARCH

	coef	std err	t	P>|t|	[0.025	0.975]
omega	0.2858	0.0258	11.0715	1.725e-28	0.2352	0.3364
alpha[1]	0.1463	0.0331	4.4161	1.005e-05	0.0814	0.2112
alpha[2]	0.2298	0.0348	6.6037	4.009e-11	0.1616	0.2980
alpha[3]	0.1979	0.0368	5.3769	7.577e-08	0.1258	0.2700
alpha[4]	0.1831	0.0400	4.5773	4.710e-06	0.1047	0.2615

15.2.2 Modélisation GARCH

Si un processus ε_t suit un modèle GARCH(p,q), alors sa forme générale est donnée par :

$$\begin{cases} \varepsilon_t = \sigma_t z_t \\ \sigma_t^2 = \omega + \sum_{t=1}^{q} \alpha_i \varepsilon_{t-i} + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2 \end{cases} \qquad z_t \sim BB(0, \sigma_z^2)$$

Si l'on suppose que le bruit blanc z_t est gaussien, alors la fonction de log-vraisemblance conditionnelle associée s'écrit :

$$\mathcal{L}(\varepsilon_{q+1}, ..., \varepsilon_q | \theta, \varepsilon_1, ..., \varepsilon_q) = \sum_{t=q+1}^{T} \left(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\sigma_t^2) - \frac{\varepsilon_t^2}{2\sigma_t^2} \right)$$

où $\sigma_t^2 = \omega + \sum_{i=1}^{i=q} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{j=p} \beta_j \sigma_{t-j}^2$ peut être évaluée de manière récursive.

Nous supposons à présent que notre rendement de l'action S&P-500 peut être modélisé par un processus $GARCH(1,1)$. Nous définissons notre fonction de log-vraisemblance associée.

```
# Log-vraisemblance d'un GARCH(1,1)
def garch_likelihood(init_params, data):
    omega, alpha, beta = init_params[0], init_params[1],init_params[2]
    T = len(data);logliks = 0; sigma2 = np.zeros(T); sigma2[0] = np.var(data)
    for t in range(1, T):
        sigma2[t] = omega + alpha * (data[t - 1]) ** 2 +beta * sigma2[t-1]
    logliks = np.sum(0.5 * (np.log(sigma2) + data** 2 / sigma2))
    return logliks
```

On initialise nos paramètres : $\omega_0 = 0.1$, $\alpha_0 = 0.3$ et $\beta_0 = 0.2$. On calcule la vraisemblance associée :

```
# Application
params_init_garch = [0.1,0.3,0.2]
logliks = garch_likelihood(params_init_garch, ret.values)
print('La log-vraisemblance est {:.4f}.'.format(logliks))
```

```
## La log-vraisemblance est 2325.4661.
```

L'objectif est de trouver les valeurs $\hat{\omega}$, $\hat{\alpha}_1$ et $\hat{\beta}_1$ qui minimisent cette log-vraisemblance. On définit une fonction de minimisation en prenant en compte la contrainte de stationnarité suivante :

$$\alpha_1 + \beta_1 < 1 \quad \iff \quad 1 - \alpha_1 - \beta_1 < 0$$

```
# Contrainte de stationnarité
def garch_constraint(init_params):
    omega,alpha, beta = init_params[0],init_params[1],init_params[2]
    return np.array([1 - alpha - beta])

# Fonction d'optimisation
def opt_paramsG(function,x0,data,bounds):
    opt_result = minimize(function,x0=x0,constraints=garch_constraint(x0),
```

```
                        bounds=bounds,args = (data),method='Nelder-Mead',
                        options={'maxiter': 5000})
    params = opt_result.x
    def left_adjust(val, length = 6): return str(val).ljust(length)
    def right_adjust(val, length = 6): return str(val).rjust(length)
    print('=='*11)
    print(' '*16,'coef\n', '--'*10)
    for coef, value in zip(['omega','alpha[1]','beta[1]'], params):
        print(left_adjust(coef,8), '     ', right_adjust(round(value,4)))
    return params

# Estimation du GARCH(1,1)
bounds =[(0.0,+np.inf),(0.0,1.0),(0.0,1.0)]
params_garch = opt_paramsG(garch_likelihood,params_init_garch,ret.values,bounds)
```

```
## ======================
##                   coef
##  --------------------
## omega           0.0399
## alpha[1]        0.1706
## beta[1]         0.7918
```

Nous comparons nos résultats avec ceux du package arch.

```
# Comparaison GARCH(1,1)
garch1 = arch_model(ret,mean='zero',vol='GARCH',p=1,o=0,q=1).fit(disp='off')
garch1_params = extractParams(garch1, model_type = "arch")
```

Table 15.7 – Coefficients du modèle GARCH

	coef	std err	t	P>|t|	[0.025	0.975]
omega	0.0399	0.0083	4.8028	1.564e-06	0.0236	0.0561
alpha[1]	0.1706	0.0215	7.9403	2.018e-15	0.1285	0.2127
beta[1]	0.7918	0.0218	36.3195	7.956e-289	0.7491	0.8345

Tout comme pour le modèle ARCH, nous recherchons les meilleurs paramètres p et q de notre modèle $GARCH$ en se basant sur le critère BIC.

```
# Choix de p et q
bic_garch = list()
for p in range(1, 5):
    for q in range(1, 5):
        garch = arch_model(ret,mean='zero',vol='GARCH',p=p,o=0,
                           q=q).fit(disp='off')
        bic_garch.append(garch.bic)
        if garch.bic == np.min(bic_garch):
            best_param = p, q
garch2 = arch_model(ret, mean='zero', vol='GARCH',p=best_param[0],o=0,
                    q=best_param[1]).fit(disp='off')
garch2_params = extractParams(garch2, model_type = "arch")
```

Table 15.8 – Coefficients du meilleur modèle GARCH

	coef	std err	t	P>|t|	[0.025	0.975]
omega	0.0399	0.0083	4.8028	1.564e-06	0.0236	0.0561
alpha[1]	0.1706	0.0215	7.9403	2.018e-15	0.1285	0.2127
beta[1]	0.7918	0.0218	36.3195	7.956e-289	0.7491	0.8345

15.2.3 Modélisation GJR-GARCH

On rappelle que si un processus ε_t satisfait un modèle GJR-GARCH si et seulement si :

$$\begin{cases} \varepsilon_t = \sigma_t z_t \\ \sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 + \sum_{i=1}^{q} \gamma_i I_{t-i} \varepsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2 \end{cases}$$

où I_{t-i} est une fonction indicatrice définie comme suit : $I_{t-i} = \begin{cases} 1 & \text{si } \varepsilon_{t-i} < 0 \\ 0 & \text{si } \varepsilon_{t-i} \geq 0 \end{cases}$.

La log-vraisemblance conditionnelle d'une variable aléatoire gaussienne s'écrit :

$$\ln f\left(\varepsilon_t | \sigma_t^2\right) = -\frac{1}{2}\left(\ln 2\pi + \ln \sigma_t^2 + \frac{\varepsilon_t^2}{\sigma_t^2}\right)$$

```
# Log - vraisemblance d'GJR-GARCH(1,1)
def gjr_garch_likelihood(parameters, data, sigma2, out=None):
    ''' Returns negative log-likelihood for GJR-GARCH(1,1,1) model.'''
    omega, alpha = parameters[0],parameters[1]
    gamma, beta = parameters[2], parameters[3]
    T = np.size(data,0); eps = data
    # Data and sigma2 are T by 1 vectors
    for t in range(1,T):
        sigma2[t] = (omega + alpha * eps[t-1]**2
                     + gamma * eps[t-1]**2 * (eps[t-1]<0) + beta * sigma2[t-1])
    logliks = 0.5*(np.log(2*np.pi) + np.log(sigma2) + eps**2/sigma2)
    loglik = np.sum(logliks)
    if out is None:
        return loglik
    else:
        return loglik, logliks, np.copy(sigma2)
```

La condition de stationnarité d'un modèle GJR-GARCH(1,1) est définie par : $\alpha_1 + \frac{\gamma_1}{2} + \beta_1 \leq 1$.

```
# Contrainte de stationnarité
def gjr_constraint(parameters, data, sigma2, out=None):
    """Constraint that alpha+gamma/2+beta<=1"""
    alpha, gamma, beta = parameters[1], parameters[2],parameters[3]
    return np.array([1-alpha-gamma/2-beta])
```

La variance asymptotique est estimée en utilisant l'expression suivante :

$$\mathcal{J}^{-1}\mathcal{I}\mathcal{J}^{-1}$$

où $\mathcal{J}$ est la matrice hessienne estimée et $\mathcal{I}$ est la covariance du score. Pour estimée la matrice hessienne, on utilise la définition suivante :

$$\mathcal{J}_{ij} \approx \frac{f\left(\theta + e_i h_i + e_j h_j\right) - f\left(\theta + e_i h_i\right) - f\left(\theta + e_j h_j\right) + f(\theta)}{h_i h_j}$$

où h_i est le pas et e_i un vecteur.

```
# Matrice Hessienne
def hessian_2sided(fun, theta, args):
    f = fun(theta, *args);h = 1e-5*np.abs(theta);thetah = theta + h
    h = thetah - theta; K = np.size(theta,0);h = np.diag(h)
    fp = np.zeros(K);fm = np.zeros(K)
    for i in range(K):
        fp[i] = fun(theta+h[i], *args); fm[i] = fun(theta-h[i], *args)
    fpp = np.zeros((K,K));fmm = np.zeros((K,K))
    for i in range(K):
        for j in range(i,K):
            fpp[i,j] = fun(theta + h[i] + h[j],  *args)
            fpp[j,i] = fpp[i,j]
            fmm[i,j] = fun(theta - h[i] - h[j],  *args)
            fmm[j,i] = fmm[i,j]

    hh = (np.diag(h));hh = hh.reshape((K,1));hh = hh@hh.T
    H = np.zeros((K,K))
    for i in range(K):
        for j in range(i,K):
            H[i,j] = (fpp[i,j] - fp[i] - fp[j] + f
                      + f - fm[i] - fm[j] + fmm[i,j])/hh[i,j]/2
            H[j,i] = H[i,j]
    return H
```

Nous initialisons les paramètres à estimer :

```
# Params init
startingVals = np.array([ret.var() * .01,.03, .09, .90])
```

Nous définissons également les bornes de nos paramètres.

```
# Estimate parameters
from scipy.optimize import fmin_slsqp
finfo = np.finfo(np.float64)
bounds = [(finfo.eps, 2*ret.var()),(0.0,1.0), (0.0,1.0), (0.0,1.0)]
T = ret.shape[0]; sigma2 = np.ones(T) * ret.var()
# Pass a NumPy array, not a pandas Series
args = (np.asarray(ret), sigma2)
estimates = fmin_slsqp(gjr_garch_likelihood, startingVals,iprint=-1,
                       f_ieqcons=gjr_constraint, bounds = bounds,args = args)
```

La log-vraisemblance optimale et la variance estimée sont obtenues en fixant le paramètre `out=True`.

```
# Get result
loglik,logliks,sigma2final = gjr_garch_likelihood(estimates,ret,sigma2,out=True)
```

Nous utilisons la définition de la dérivabilité

$$\frac{\partial(\theta)}{\partial\theta_i} \approx \frac{f(\theta + e_i h_i) - f(\theta)}{h_i}$$

afin de calculer la covariance du score.

```
# Covariance of scores
step = 1e-5 * estimates; scores = np.zeros((T,4))
for i in range(4):
    h = step[i]
    delta = np.zeros(4)
    delta[i] = h
    loglik,logliksplus,sigma2 = gjr_garch_likelihood(estimates + delta, \
                                np.asarray(ret), sigma2, out=True)
    loglik,logliksminus,sigma2 = gjr_garch_likelihood(estimates - delta, \
                                np.asarray(ret), sigma2, out=True)
    scores[:,i] = (logliksplus - logliksminus)/(2*h)

I = (scores.T@scores)/T
```

La prochaine étape est d'estimer la matrice hessienne.

```
# Matrice hessienne
J = hessian_2sided(gjr_garch_likelihood, estimates, args)
J = J/T
Jinv = np.mat(np.linalg.inv(J))
vcv = Jinv*np.mat(I)*Jinv/T
vcv = np.asarray(vcv)
```

Nous pouvons maintenant afficher les résultats obtenus.

```
# Affichage des résultats
output = np.vstack((estimates,np.sqrt(np.diag(vcv)),
                    estimates/np.sqrt(np.diag(vcv)))).T
df = pd.DataFrame(output,columns = ["Estimate","Std. Err.","T-stat"],
                  index = ['omega','alpha','gamma','beta'])
```

Table 15.9 – Coefficients estimés du modèle GJR - GARCH(1,1)

	Estimate	Std. Err.	T-stat
omega	0.0432	0.0077	5.6374
alpha	0.0339	0.0294	1.1545
gamma	0.2776	0.0461	6.0191
beta	0.7951	0.0261	30.4373

Nous comparons nos résultats avec ceux du package arch.

```
# Verification : estimation d'un modèle GJR-GARCH(1,1)
gjr_garch = arch_model(ret,mean="Zero",p=1,o=1,q=1,dist='normal').fit(disp='off')
gjr_garch_params = extractParams(gjr_garch, model_type = "arch")
```

Table 15.10 – Coefficients du modèle GJR - GARCH

	coef	std err	t	P>|t|	[0.025	0.975]
omega	0.0432	0.0077	5.6414	1.687e-08	0.0282	0.0582
alpha[1]	0.0339	0.0294	1.1520	2.493e-01	-0.0237	0.0915
gamma[1]	0.2776	0.0461	6.0254	1.687e-09	0.1873	0.3679
beta[1]	0.7951	0.0261	30.4757	5.471e-204	0.7440	0.8463

On constate que les résultats sont identiques.

Tables statistiques

Table 11 – Distribution cumulative empirique de t_ρ sous $H_0 : \rho = 0$

	Probabilité d'une valeur inférieure							
T	0.01	0.025	0.05	0.10	0.90	0.95	0.975	0.99
Modèle [1] : $t_{\rho \bullet}$								
25	−2.66	−2.26	−1.95	−1.60	0.92	1.33	1.70	2.16
50	−2.62	−2.25	−1.95	−1.61	0.91	1.31	1.66	2.08
100	−2.60	−2.24	−1.95	−1.61	0.90	1.29	1.64	2.03
250	−2.58	−2.23	−1.95	−1.62	0.89	1.29	1.63	2.01
500	−2.58	−2.23	−1.95	−1.62	0.89	1.28	1.62	2.00
∞	−2.58	−2.23	−1.95	−1.62	0.89	1.28	1.62	2.00
Modèle [2] : $t_{\widehat{\rho}}$								
25	−3.75	−3.33	−3.00	−2.63	−0.37	0.00	0.34	0.72
50	−3.58	−3.22	−2.93	−2.60	−0.40	−0.03	0.29	0.66
100	−3.51	−3.17	−2.89	−2.58	−0.42	−0.05	0.26	0.63
250	−3.46	−3.14	−2.88	−2.57	−0.42	−0.06	0.24	0.62
500	−3.44	−3.13	−2.87	−2.57	−0.43	−0.07	0.24	0.61
∞	−3.43	−3.12	−2.89	−2.57	−0.44	−0.07	0.23	0.60
Modèle [3] : $t_{\widetilde{\rho}}$								
25	−4.38	−3.95	−3.60	−3.24	−1.14	−0.80	−0.50	−0.15
50	−4.15	−3.80	−3.50	−3.18	−1.19	−0.87	−0.58	−0.24
100	−4.04	−3.73	−3.45	−3.15	−1.22	−0.90	−0.62	−0.28
250	−3.99	−3.69	−3.43	−3.13	−1.23	−0.92	−0.64	−0.31
500	−3.98	−3.68	−3.42	−3.13	−1.24	−0.93	−0.65	−0.32
∞	−3.96	−3.66	−3.41	−3.12	−1.25	−0.94	−0.66	−0.33

Source : Fuller 1976, p. 373, Tableau 8.5.2.

Table 12 – Distribution empirique de t pour $(c, \rho) = (0, 0)$ Modèle [2]

	Probabilité d'une valeur inférieure			
T	0.90	0.95	0.975	0.99
Modèle [2] : $t_{\widehat{c}}$				
25	2.20	2.61	2.97	3.41
50	2.18	2.56	2.89	3.28
100	2.17	2.54	2.86	3.22
250	2.16	2.53	2.84	3.19
500	2.16	2.52	2.83	3.18
∞	2.16	2.52	2.83	3.18

Source : Dickey et Fuller, 1981, p1062, Tableau I.

Table 13 – Distribution empirique de Φ_1 pour $H_0^1 : (c, \rho) = (0, 0)$ Modèle [2]

	Probabilité d'une valeur inférieure							
T	0.01	0.025	0.05	0.10	0.90	0.95	0.975	0.99
Modèle [2] Φ_1 $H_0^1 : (c, \rho) = (0, 0)$								
25	0.29	0.38	0.49	0.65	4.12	5.18	6.30	7.88
50	0.29	0.39	0.50	0.66	3.94	4.86	5.80	7.06
100	0.29	0.39	0.50	0.67	3.86	4.71	5.57	6.70
250	0.30	0.39	0.51	0.67	3.81	4.63	5.45	6.52
500	0.30	0.39	0.51	0.67	3.79	4.61	5.41	6.47
∞	0.30	0.40	0.51	0.67	3.78	4.59	5.38	6.43

Source : Dickey et Fuller, 1981, p1063, Tableau IV.

Table 14 – Distribution empirique de $t_{\widehat{c}}$ pour $(c, b, \rho) = (0, 0, 0)$ Modèle [3]

	Probabilité d'une valeur inférieure			
T	0.90	0.95	0.975	0.99
Modèle [3] : $t_{\widehat{c}}$				
25	2.77	3.20	3.59	4.05
50	2.75	3.14	3.47	3.87
100	2.73	3.11	3.42	3.78
250	2.73	3.09	3.39	3.74
500	2.72	3.08	3.38	3.72
∞	2.72	3.08	3.38	3.71

Source : Dickey et Fuller, 1981, p1062, Tableau II.

Table 15 – Distribution empirique de $t_{\hat{b}}$ pour $(c, b, \rho) = (0, 0, 0)$ Modèle [3]

	Probabilité d'une valeur inférieure			
T	0.90	0.95	0.975	0.99
Modèle [3] : $t_{\hat{b}}$				
25	2.39	2.85	3.25	3.74
50	2.38	2.81	3.18	3.60
100	2.38	2.79	3.14	3.53
250	2.38	2.79	3.12	3.49
500	2.38	2.78	3.11	3.48
∞	2.38	2.78	3.11	3.46

Source : Dickey et Fuller, 1981, p1062, Tableau III.

Table 16 – Distribution empirique de Φ_2 pour $H_0^2 : (c, b, \rho) = (0, 0, 0)$ Modèle [3]

	Probabilité d'une valeur inférieure							
T	0.01	0.025	0.05	0.10	0.90	0.95	0.975	0.99
Modèle [3] Φ_2 $H_0^2 : (c, b, \rho) = (0, 0, 0)$								
25	0.61	0.75	0.89	1.10	4.67	5.68	6.75	8.21
50	0.62	0.77	0.91	1.12	4.31	5.13	5.94	7.02
100	0.63	077	0.92	1.12	4.16	4.88	5.59	6.50
250	0.63	0.77	0.92	1.13	4.07	4.75	5.40	6.22
500	0.63	0.77	0.92	1.13	4.05	4.71	5.35	6.15
∞	0.63	077	0.92	1.13	4.03	4.68	5.31	6.09

Source : Dickey et Fuller, 1981, p1063, Tableau V.

Table 17 – Distribution empirique de Φ_3 pour $H_0^3 : (c, b, \rho) = (c, 0, 0)$ Modèle [3]

	Probabilité d'une valeur inférieure							
T	0.01	0.025	0.05	0.10	0.90	0.95	0.975	0.99
Modèle [3] Φ_3 $H_0^3 : (c, b, \rho) = (c, 0, 0)$								
25	0.74	0.90	1.08	1.33	5.91	7.24	8.65	10.61
50	0.76	0.93	1.11	1.37	5.61	6.73	7.81	9.31
100	0.76	0.94	1.12	1.38	5.47	6.49	7.44	8.73
250	0.76	0.94	1.13	1.39	5.39	6.34	7.25	8.43
500	0.76	0.94	1.13	1.39	5.36	6.30	7.20	8.34
∞	0.77	0.94	1.13	1.39	5.34	6.25	7.16	8.27

Source : Dickey et Fuller, 1981, p1063, Tableau VI.

Bibliographie

Abdi, Herve, and Paul Molin. 2007. "Lilliefors/van Soest's Test of Normality." *Encyclopedia of Measurement and Statistics*, 540–44.

Abraham, Bovas, and Johannes Ledolter. 1983. *Statistical Methods for Forecasting*. Vol. 179. Wiley Online Library.

Ahamada, Ibrahim, and Philippe Jolivaldt. 2010. "Filtres Usuels Et Filtre Fondé Sur Les Ondelettes : étude Comparative Et Application Au Cycle économique." *Economie Prevision*, no. 4 : 149–61.

Akaike, Hirotugu. 1973. "Information Theory and an Extention of the Maximum Likelihood Principle." In *2nd International Symposium on Information Theory, 1973*, 267–81. Akademiai Kiado.

———. 1974. "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control* 19 (6) : 716–23.

———. 1979. "A Bayesian Extension of the Minimum AIC Procedure of Autoregressive Model Fitting." *Biometrika* 66 (2) : 237–42.

Aragon, Yves. 2011. *Séries Temporelles*. Ed. Techniques Ingénieur.

Avram, Florin. 2017. "Séries Temporelles Avec r." Edition.

Bandara, Kasun, Rob J Hyndman, and Christoph Bergmeir. 2021. "MSTL : A Seasonal-Trend Decomposition Algorithm for Time Series with Multiple Seasonal Patterns." *arXiv Preprint arXiv :2107.13462*.

Bartels, Robert. 1982. "The Rank Version of von Neumann's Ratio Test for Randomness." *Journal of the American Statistical Association* 77 (377) : 40–46.

Beaudry, Paul, and Gary Koop. 1993. "Do Recessions Permanently Change Output ?" *Journal of Monetary Economics* 31 (2) : 149–63.

Bell, William R, and Steven C Hillmer. 1984. "Issues Involved with the Seasonal Adjustment of Economic Time Series." *Journal of Business & Economic Statistics* 2 (4) : 291–320.

Black, Fischer. 1976. "Studies of Stock Market Volatility Changes." *1976 Proceedings of the American Statistical Association Bisiness and Economic Statistics Section*.

Bollerslev, Tim. 1986. "Generalized Autoregressive Conditional Heteroskedasticity." *Journal of Econometrics* 31 (3) : 307–27.

Bourbonnais, Régis. 2018. *Économétrie-10e éd. : Cours Et Exercices Corrigés*. Dunod.

Bourbonnais, Régis, and Michel Terraza. 2010. *Analyse Des séries Temporelles-3ème édition*. Hachette.

Box, George EP, and David A Pierce. 1970. "Distribution of Residual Autocorrelations in Autoregressive-Integrated Moving Average Time Series Models." *Journal of the American Statistical Association* 65 (332) : 1509–26.

Box, G, and G Jenkings. 1970. “Time Series Analysis : Forecasting and Control, Holden Day, San Francisco, USA.”

Bradley, James V. 1968. “Distribution-Free Statistical Tests.”

Brockwell, Peter J, and Richard A Davis. 2002. *Introduction to Time Series and Forecasting*. Springer.

———. 2009. *Time Series : Theory and Methods*. Springer science & business media.

Brown, Robert Goodell. 1959. “Statistical Forecasting for Inventory Control.”

Burns, Arthur F, and Wesley C Mitchell. 1946a. *Measuring Business Cycles*. burn46-1. National bureau of economic research.

———. 1946b. “The Basic Measures of Cyclical Behavior.” In *Measuring Business Cycles*, 115–202. NBER.

Campbell, John Y, and Pierre Perron. 1991. “Pitfalls and Opportunities : What Macroeconomists Should Know about Unit Roots.” *NBER Macroeconomics Annual* 6 : 141–201.

Cao, Charles Q, and Ruey S Tsay. 1992. “Nonlinear Time-Series Analysis of Stock Volatilities.” *Journal of Applied Econometrics* 7 (S1) : S165–85.

Charpentier, Arthur. 2006. “Cours de séries Temporelles : Théorie Et Applications.” *Université Paris Dauphine*.

———. 2012. “Modèles de Prévision séries Temporelles.” *Paris, France*.

Chen, Yi. 2002. “Ting,(2002)” on the Robustness of Ljung- Box and McLeod- Li q Tests : A Simulation Study.”.” *Economics Bulletin* 3 (17) : 1–10.

Chow, Gregory C. 1960. “Tests of Equality Between Sets of Coefficients in Two Linear Regressions.” *Econometrica : Journal of the Econometric Society*, 591–605.

Christiano, Lawrence J, and Terry J Fitzgerald. 2003. “The Band Pass Filter.” *International Economic Review* 44 (2) : 435–65.

Cleveland, Robert B, William S Cleveland, Jean E McRae, and Irma Terpenning. 1990. “STL : A Seasonal-Trend Decomposition.” *J. Off. Stat* 6 (1) : 3–73.

Cleveland, William S. 1979. “Robust Locally Weighted Regression and Smoothing Scatterplots.” *Journal of the American Statistical Association* 74 (368) : 829–36.

———. 1981. “LOWESS : A Program for Smoothing Scatterplots by Robust Locally Weighted Regression.” *American Statistician* 35 (1) : 54.

Cleveland, William S, and Susan J Devlin. 1988. “Locally Weighted Regression : An Approach to Regression Analysis by Local Fitting.” *Journal of the American Statistical Association* 83 (403) : 596–610.

Cromwell, Jeff B, and Michel Terraza. 1994. *Multivariate Tests for Time Series Models*. 100. Sage.

D’Agostino, Ralph B. 1970. “Transformation to Normality of the Null Distribution of G1.” *Biometrika*, 679–81.

D’Agostino, RalphB. 2017. *Goodness-of-Fit-Techniques*. Routledge.

DAGUM, ED. 1979. “Ajustement Saisonnier Et Problèmes de séries Temporelles.” *Economie Appliquée Paris* 32 (1) : 5–129.

Dagum, Estela Bee. 1978. *A Comparison and Assessment of Seasonal Adjustment Methods for Employment and Unemployment Statistics*. 5. National Commission on Employment ; Unemployment Statistics.

———. 1988. *The X11arima/88 Seasonal Adjustment Method : Foundations and User’s Manual*. Statistics Canada, Time Series Research ; Analysis Division.

Dagum, Estela Bee, and Silvia Bianconcini. 2016. *Seasonal Adjustment Methods and Real Time Trend-Cycle Estimation*. Springer.

Dallal, Gerard E, and Leland Wilkinson. 1986. “An Analytic Approximation to the Distribution of Lilliefors’s Test Statistic for Normality.” *The American Statistician* 40 (4) : 294–96.

Danioko, Sidy, Jianwei Zheng, Kyle Anderson, Alexander Barrett, and Cyril S Rakovski. 2022. “A Novel Correction for the Adjusted Box-Pierce Test.”

Darné, Olivier. 2004. “Les méthodes Et Logiciels de désaisonnalisation Des séries économiques : Une Revue de La Littérature.” *Journal de La Société Française de Statistique* 145 (4) : 79–102.

Davidson, Russell, James G MacKinnon, et al. 1993. *Estimation and Inference in Econometrics*. Vol. 63. Oxford New York.

Davies, Nelville, CM Triggs, and Paul Newbold. 1977. “Significance Levels of the Box-Pierce Portmanteau Statistic in Finite Samples.” *Biometrika* 64 (3) : 517–22.

De Livera, Alysha M, Rob J Hyndman, and Ralph D Snyder. 2011. “Forecasting Time Series with Complex Seasonal Patterns Using Exponential Smoothing.” *Journal of the American Statistical Association* 106 (496) : 1513–27.

Dickey, David A, and Wayne A Fuller. 1979. “Distribution of the Estimators for Autoregressive Time Series with a Unit Root.” *Journal of the American Statistical Association* 74 (366a) : 427–31.

———. 1981. “Likelihood Ratio Statistics for Autoregressive Time Series with a Unit Root.” *Econometrica : Journal of the Econometric Society*, 1057–72.

Dokumentov, Alexander, and Rob J Hyndman. 2022. “STR : Seasonal-Trend Decomposition Using Regression.” *INFORMS Journal on Data Science* 1 (1) : 50–62.

Durbin, James. 1960. “The Fitting of Time-Series Models.” *Revue de l’Institut International de Statistique*, 233–44.

Durbin, James, and Geoffrey S Watson. 1950. “Testing for Serial Correlation in Least Squares Regression : i.” *Biometrika* 37 (3/4) : 409–28.

———. 1971. “Testing for Serial Correlation in Least Squares Regression. III.” *Biometrika* 58 (1) : 1–19.

Engle, Robert F. 1982. “Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation.” *Econometrica : Journal of the Econometric Society*, 987–1007.

Engle, Robert F, David M Lilien, and Russell P Robins. 1987. “Estimating Time Varying Risk Premia in the Term Structure : The ARCH-m Model.” *Econometrica : Journal of the Econometric Society*, 391–407.

Ertur, Cem. 1992. “Tests de Non Stationnarité Et Tendances Non Linéaires.” PhD thesis, Laboratoire d’analyse et de techniques économiques (LATEC).

Findley, David F, Brian C Monsell, William R Bell, Mark C Otto, and Bor-Chung Chen. 1998. “New Capabilities and Methods of the x-12-ARIMA Seasonal-Adjustment Program.” *Journal of Business & Economic Statistics* 16 (2) : 127–52.

Fisher, Thomas J. 2011. “Testing Adequacy of Arma Models Using a Weighted Portmanteau Test on the Residual Autocorrelations.” *Contributed Paper* 327 : 2011.

Fuller, Wayne A. 2009. *Introduction to Statistical Time Series*. John Wiley & Sons.

Geweke, John, and Richard Meese. 1981. “Estimating Regression Models of Finite but Unknown Order.” *International Economic Review*, 55–70.

Goodwin, Richard M. 1951. “The Nonlinear Accelerator and the Persistence of Business Cycles.” *Econometrica : Journal of the Econometric Society*, 1–17.

Granger, Clive WJ. 1966. “The Typical Spectral Shape of an Economic Variable.” *Econometrica : Journal of the Econometric Society*, 150–61.

Granger, Clive WJ, and Paul Newbold. 1974. “Spurious Regressions in Econometrics.” *Journal of Econometrics* 2 (2) : 111–20.

Griffiths, William E, George G Judge, R Carter Hill, Helmut Lütkepohl, and Tsoung-Chao Lee. 1985. *The Theory and Practice of Econometrics*. Wiley.

Guay, Alain, and Pierre St.-Amant. 2005. “Do the Hodrick-Prescott and Baxter-King Filters Provide a Good Approximation of Business Cycles ?” *Annales d’Economie Et de Statistique*, 133–55.

Hallin, Marc, and Guy Mélard. 1988. “Rank-Based Tests for Randomness Against First-Order Serial Dependence.” *Journal of the American Statistical Association* 83 (404) : 1117–28.

Hamed, Khaled H, and A Ramachandra Rao. 1998. "A Modified Mann-Kendall Trend Test for Autocorrelated Data." *Journal of Hydrology* 204 (1-4) : 182–96.

Hannan, Edward J, and Barry G Quinn. 1979. "The Determination of the Order of an Autoregression." *Journal of the Royal Statistical Society : Series B (Methodological)* 41 (2) : 190–95.

Hannan, EJ. 1973. "Multivariate Time Series Analysis." *Journal of Multivariate Analysis* 3 (4) : 395–407.

Harvey, Andrew C. 1990. "Forecasting, Structural Time Series Models and the Kalman Filter."

Haslwanter, Thomas. 2016. "An Introduction to Statistics with Python." *With Applications in the Life Sciences. Switzerland : Springer International Publishing*.

Henderson, Robert. 1916. "Note on Graduation by Adjusted Average." *Transactions of the Actuarial Society of America* 17 : 43–48.

Hicks, John Richard. 1950. "A Contribution to the Theory of the Trade Cycle."

Hirsch, Robert M, James R Slack, and Richard A Smith. 1982. "Techniques of Trend Analysis for Monthly Water Quality Data." *Water Resources Research* 18 (1) : 107–21.

Hodrick, Robert J, and Edward C Prescott. 1997. "Postwar US Business Cycles : An Empirical Investigation." *Journal of Money, Credit, and Banking*, 1–16.

Holt, Charles C. 1957. "Forecasting Trends and Seasonals by Exponentially Weighted Moving Averages." *ONR Memorandum* 52 (52) : 5–10.

Hurvich, Clifford M, and Chih-Ling Tsai. 1989. "Regression and Time Series Model Selection in Small Samples." *Biometrika* 76 (2) : 297–307.

———. 1995. "Model Selection for Extended Quasi-Likelihood Models in Small Samples." *Biometrics*, 1077–84.

Hyndman, Rob J, and George Athanasopoulos. 2014. "Forecasting : Principles and Practice. Otexts ; 2014." *Online at Http ://Otexts. Org/Fpp*.

Hyndman, Rob, Anne B Koehler, J Keith Ord, and Ralph D Snyder. 2008. *Forecasting with Exponential Smoothing : The State Space Approach*. Springer Science & Business Media.

Iwueze, Iheanyi S, Eleazar C Nwogu, Ohakwe Johnson, Jude C Ajaraogu, et al. 2011. "Uses of the Buys-Ballot Table in Time Series Analysis." *Applied Mathematics* 2 (05) : 633.

Jarque, Carlos M, and Anil K Bera. 1987. "A Test for Normality of Observations and Regression Residuals." *International Statistical Review/Revue Internationale de Statistique*, 163–72.

Kendall, Maurice G. 1938. "A New Measure of Rank Correlation." *Biometrika* 30 (1/2) : 81–93.

King, Robert G, and Marianne Baxter. 1995. *Measuring Business Cycles Approximate Band-Pass Filters for Economic Time Series*. National Bureau of Economic Research.

King, Robert G, and Sergio T Rebelo. 1993. "Low Frequency Filtering and Real Business Cycles." *Journal of Economic Dynamics and Control* 17 (1-2) : 207–31.

King, W Willford I. 1924. "An Improved Method for Measuring the Seasonal Factor." *Journal of the American Statistical Association* 19 (147) : 301–13.

Kullback, Solomon, and Richard A Leibler. 1951. "On Information and Sufficiency." *The Annals of Mathematical Statistics* 22 (1) : 79–86.

Kwiatkowski, Denis, Peter CB Phillips, Peter Schmidt, and Yongcheol Shin. 1992. "Testing the Null Hypothesis of Stationarity Against the Alternative of a Unit Root : How Sure Are We That Economic Time Series Have a Unit Root ?" *Journal of Econometrics* 54 (1-3) : 159–78.

Laloire, Jean-Claude. 1972. *Méthodes Du Traitement Des Chroniques : Statistiques Et Prévisions de Ventes*. Dunod.

Leslie, JR, Michael A Stephens, and S Fotopoulos. 1986. "Asymptotic Distribution of the Shapiro-Wilk *W* for Testing for Normality." *The Annals of Statistics* 14 (4) : 1497–1506.

Lewinson, Eryk. 2020. *Python for Finance Cookbook : Over 50 Recipes for Applying Modern Python Libraries to Financial Data Analysis*. Packt Publishing Ltd.

Lilliefors, Hubert W. 1967. "On the Kolmogorov-Smirnov Test for Normality with Mean and Variance Unknown." *Journal of the American Statistical Association* 62 (318) : 399–402.

Lin, Jen-Wen, and A Ian McLeod. 2006. “Improved Peña–Rodriguez Portmanteau Test.” *Computational Statistics & Data Analysis* 51 (3) : 1731–38.

Ljung, Greta M, and George EP Box. 1978. “On a Measure of Lack of Fit in Time Series Models.” *Biometrika* 65 (2) : 297–303.

Lo, Andrew W, and A Craig MacKinlay. 1989. “The Size and Power of the Variance Ratio Test in Finite Samples : A Monte Carlo Investigation.” *Journal of Econometrics* 40 (2) : 203–38.

Lobato, Ignacio, John C Nankervis, and N Eugene Savin. 2001. “Testing for Autocorrelation Using a Modified Box-Pierce q Test.” *International Economic Review* 42 (1) : 187–205.

Lütkepohl, Helmut. 2005. *New Introduction to Multiple Time Series Analysis*. Springer Science & Business Media.

———. 2013. *Introduction to Multiple Time Series Analysis*. Springer Science & Business Media.

Lütkepohl, Helmut, and Markus Krätzig. 2004. *Applied Time Series Econometrics*. Cambridge university press.

Maddala, GS, and AS Rao. 1971. “Maximum Likelihood Estimation of Solow’s and Jorgenson’s Distributed Lag Models.” *The Review of Economics Statistic*, 80–88.

Mallows, Colin L. 2000. “Some Comments on Cp.” *Technometrics* 42 (1) : 87–94.

Mann, Henry B. 1945. “Nonparametric Tests Against Trend.” *Econometrica : Journal of the Econometric Society*, 245–59.

McLeod, Allan I, and William K Li. 1983. “Diagnostic Checking ARMA Time Series Models Using Squared-Residual Autocorrelations.” *Journal of Time Series Analysis* 4 (4) : 269–73.

Mignon, Valérie, and Gilbert Abraham-Frois. 1998. *Marchés Financiers Et Modélisation Des Rentabilités Boursières*. Economica.

Monti, Anna Clara. 1994. “A Proposal for a Residual Autocorrelation Test in Linear Models.” *Biometrika* 81 (4) : 776–80.

Musgrave, J. 1964. “A Set of End Weights to End All End Weights, Working Paper.” *US Bureau of the Census, Washington*.

Nelson, Daniel B. 1991. “Conditional Heteroskedasticity in Asset Returns : A New Approach.” *Econometrica : Journal of the Econometric Society*, 347–70.

Nelson, Daniel B, and Charles Q Cao. 1992. “Inequality Constraints in the Univariate GARCH Model.” *Journal of Business & Economic Statistics* 10 (2) : 229–35.

O’Hara-Wild, M, RJ Hyndman, and E Wang. 2021. “Tsibbledata : Diverse Datasets For’tsibble’.” *URL : Https ://CRAN. R-Project. Org/Package= Tsibbledata. R Package Version 0.3. 0.(Creative Commons License)*.

Parzen, Emanuel. 1962. “On Estimation of a Probability Density Function and Mode.” *The Annals of Mathematical Statistics* 33 (3) : 1065–76.

Peña, Daniel, and Julio Rodríguez. 2002. “A Powerful Portmanteau Test of Lack of Fit for Time Series.” *Journal of the American Statistical Association* 97 (458) : 601–10.

Phillips, Peter CB, and Pierre Perron. 1988. “Testing for a Unit Root in Time Series Regression.” *Biometrika* 75 (2) : 335–46.

Poynting, John Henry. 1884. “XV. On the Transfer of Energy in the Electromagnetic Field.” *Philosophical Transactions of the Royal Society of London*, no. 175 : 343–61.

Priestley, Maurice Bertram. 1981. *Spectral Analysis and Time Series : Univariate Series*. Vol. 1. Academic press.

Quenouille, Maurice H. 1957. “The Analysis of Multiple Time-Series. London : Griffin.” *QuenouilleThe Analysis of Multiple Time Series1957*.

Ravn, Morten O, and Harald Uhlig. 2002. “On Adjusting the Hodrick-Prescott Filter for the Frequency of Observations.” *Review of Economics and Statistics* 84 (2) : 371–76.

Rosenblatt, Murray. 1956. “Remarks on Some Nonparametric Estimates of a Density Function.” *The Annals of Mathematical Statistics*, 832–37.

Safi, Samir K, and Alaa A Al-Reqep. 2014. “Comparative Study of Portmanteau Tests for the Residuals Autocorrelation in ARMA Models.” *Journal of Applied Mathematics and Statistics*

2 (1).
Saporta, Gilbert. 2006. *Probabilités, Analyse Des Données Et Statistique*. Editions technip.
Schwarz, Gideon. 1978. "Estimating the Dimension of a Model." *The Annals of Statistics*, 461–64.
Seabold, Skipper, and Josef Perktold. 2010. "Statsmodels : Econometric and Statistical Modeling with Python." In *9th Python in Science Conference*.
Sentana, Enrique. 1995. "Quadratic ARCH Models." *The Review of Economic Studies* 62 (4) : 639–61.
Shapiro, Samuel Sanford, and Martin B Wilk. 1965. "An Analysis of Variance Test for Normality (Complete Samples)." *Biometrika* 52 (3/4) : 591–611.
Shapiro, Samuel S, and RS Francia. 1972. "An Approximate Analysis of Variance Test for Normality." *Journal of the American Statistical Association* 67 (337) : 215–16.
Sheppard, Kevin. 2014. "Introduction to Python for Econometrics, Statistics and Data Analysis." University of Oxford.
Shiskin, Julius. 1967. *The x-11 Variant of the Census Method II Seasonal Adjustment Program*. 15. US Department of Commerce, Bureau of the Census.
Singleton, Kenneth J. 1988. "Econometric Issues in the Analysis of Equilibrium Business Cycle Models." *Journal of Monetary Economics* 21 (2-3) : 361–86.
Slutzky, Eugen. 1937. "The Summation of Random Causes as the Source of Cyclic Processes." *Econometrica : Journal of the Econometric Society*, 105–46.
Stephens, Michael A. 1970. "Use of the Kolmogorov–Smirnov, Cramer–von Mises and Related Statistics Without Extensive Tables." *Journal of the Royal Statistical Society : Series B (Methodological)* 32 (1) : 115–22.
———. 1974. "EDF Statistics for Goodness of Fit and Some Comparisons." *Journal of the American Statistical Association* 69 (347) : 730–37.
———. 2017. "Tests Based on EDF Statistics." In *Goodness-of-Fit Techniques*, 97–194. Routledge.
Stock, James H, and Mark W Watson. 2005. "Understanding Changes in International Business Cycle Dynamics." *Journal of the European Economic Association* 3 (5) : 968–1006.
Sugiura, Nariaki. 1978. "Further Analysts of the Data by Akaike's Information Criterion and the Finite Corrections : Further Analysts of the Data by Akaike's." *Communications in Statistics-Theory and Methods* 7 (1) : 13–26.
Sutcliffe, Andrew. 1999. *Seasonal Adjustment : Comparison of Philosophies*. Australian Bureau of Statistics.
Taylor, Sean J, and Benjamin Letham. 2018. "Forecasting at Scale." *The American Statistician* 72 (1) : 37–45.
Toyoda, Toshihisa. 1974. "Use of the Chow Test Under Heteroscedasticity." *Econometrica : Journal of the Econometric Society*, 601–8.
Tsay, Ruey S. 2013. *Multivariate Time Series Analysis : With r and Financial Applications*. John Wiley & Sons.
Udny Yule, George. 1927. "On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer's Sunspot Numbers." *Philosophical Transactions of the Royal Society of London Series A* 226 : 267–98.
Von Neumann, John. 1941. "Distribution of the Ratio of the Mean Square Successive Difference to the Variance." *The Annals of Mathematical Statistics* 12 (4) : 367–95.
Winters, Peter R. 1960. "Forecasting Sales by Exponentially Weighted Moving Averages." *Management Science* 6 (3) : 324–42.
Zakoian, Jean-Michel. 1994. "Threshold Heteroskedastic Models." *Journal of Economic Dynamics and Control* 18 (5) : 931–55.
Zivot, Eric, and Donald W K Andrews. 2002. "Further Evidence on the Great Crash, the Oil-Price Shock, and the Unit-Root Hypothesis." *Journal of Business & Economic Statistics* 20 (1) : 25–44.

Analyse des séries temporelles univariées :

Théorie et application sous Python

Cet ouvrage présente les bases de l'économétrie des séries temporelles. Il met l'accent sur les méthodes et les compétences indispensables à tout étudiant ou tout praticien désireux d'appliquer les méthodes basiques d'économétrie des séries temporelles L'auteur met rapidement en pratique les divers concepts présentés avec des exemples appliqués sous le langage de programmation Python.

Ce manuel s'adresse aux étudiants de Licence et Master de mathématiques appliquées, d'économie et d'économétrie, ainsi qu'aux élèves des écoles d'ingénieurs. Il s'adresse également aux professionnels, praticiens de l'économétrie des séries temporelles.

Duvérier DJIFACK ZEBAZE est ingénieur en Data Science et gestion des risques de l'école nationale de la statistique et de l'analyse de l'information (ENSAI) de Rennes et Ingénieur Statisticien Economiste (ISE) de l'école nationale de la statistique et de l'analyse économique (ENSAE) de Dakar. Sa carrière débute en tant qu'ingénieur quantitatif au sein de l'équipe de modélisation de la Business Unit GLBA (Global Banking and Advisory) du groupe Société Générale.