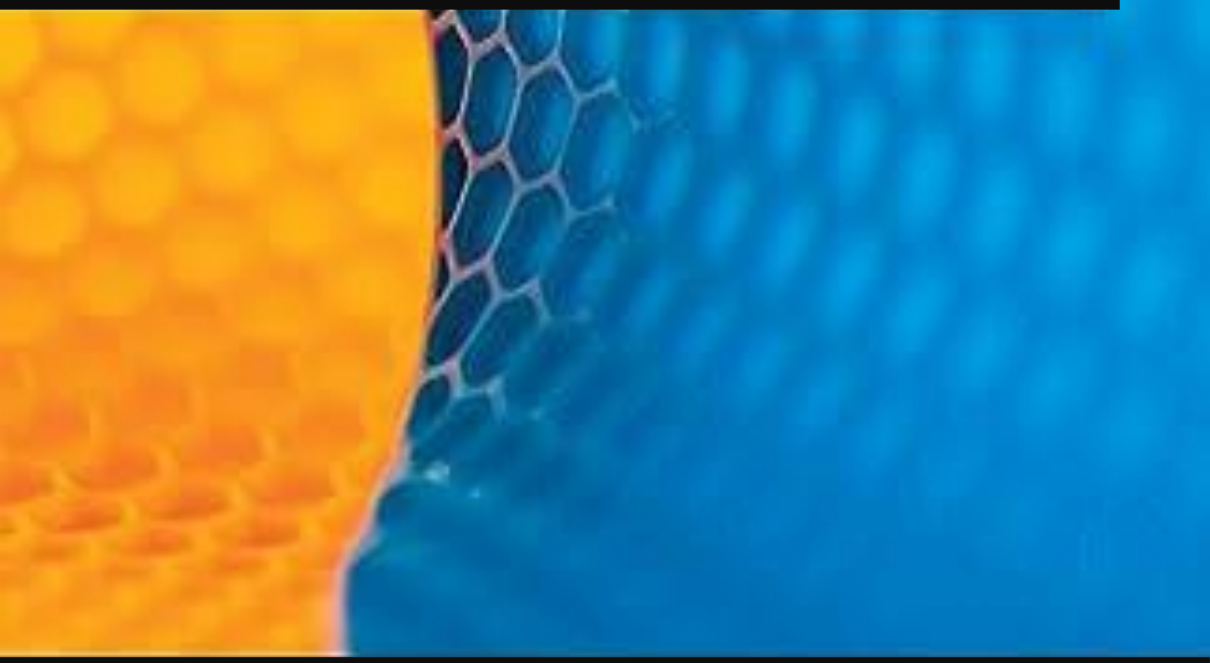
Econométrie du modèle linéaire

Théorie et application sous Python



Duvérier DJIFACK ZEBAZE

Econométrie du modèle linéaire

Théorie et application sous Python

Duvérier DJIFACK ZEBAZE

Table des matières

1 Le modèle linéaire de régression simple

Sommaire

On parle de modèle linéaire de régression simple, lorsqu'une seule variable est utilisée pour expliquer une autre variable. La variable expliquée est appelée variable endogène et la variable explicative est appelée variable exogène. Un tel modèle peut être spécifié en séries temporelles ou en coupe instantanée. En séries temporelles, les variables sont observées en intervalle de temps régulier alors qu'en coupe instantanée, les données sont observées au même instant et concernent les valeurs prises par la variable pour un groupe d'individu spécifique.

1.1 Le modèle et ses hypothèses

1.1.1 Présentation du modèle

Spécifié en séries temporelles, le modèle linéaire de régression simple s'écrit :

$$y_t = \beta_0 + \beta_1 x_t + \varepsilon_t, \quad \forall t = 1, \dots, n \tag{1.1}$$

Dans ce modèle, y_t est la variable à expliquer au temps t (variable endogène) ; x_t est la variable explicative au temps t (variable exogène) ; β_0 et β_1 sont les coefficients ou paramètres du modèle ; n est le nombre de période (nombre d'observations) et ε_t est le terme d'erreur (l'erreur de spécification), c'est-à-dire la différence entre le modèle vrai et le modèle spécifié.

En effet, le modèle ci-dessus stipule que la variable endogène y est expliquée par la variable exogène x, mais il existe peut être d'autres variables que l'on ignore qui contribue aussi à l'explication de la variable y. L'erreur de spécification (ε) qui est un terme aléatoire à pour rôle de prendre en compte toutes ses autres variables explicatives non expliquées.

L'une des étapes de la résolution du modèle consiste à estimer les paramètres β_0 et β_1. Certaines hypothèses sont émises sur le terme d'erreur aléatoire (ε) en vue d'obtenir des estimateurs sans biais et convergents des paramètres du modèle.

1.1.1.1 Les hypothèses du modèle

Ces hypothèses pèsent sur les propriétés des estimateurs (biais,convergence) et l'inférence statistique (distribution des coefficents estimés).

H1) Les valeurs de la variable explicative sont observées sans erreur

Cette variable est donc non aléatoire (alors que la variable endogène y est aléatoire en raison de la présence du terme aléatoire (ε)).

H2) L'espérance mathématique de l'erreur est nulle : $\mathbb{E}(\varepsilon_t) = 0, \quad \forall t = 1, \dots, n$

Cela signifie qu'en moyenne le modèle est bien spécifié puisque l'erreur moyenne est nulle.

H3) La variance de l'erreur est une constante : $\mathrm{V}(\varepsilon_t) = \sigma_\varepsilon^2, \quad \forall t = 1, \dots, n$

C'est l'hypothèse de l'homoscédasticité. Lorsqu'elle n'est pas vérifiée, on dit que le modèle est hétéroscédastique.

H4) Les termes d'erreur sont indépendants les uns des autres dans le temps : $\mathrm{Cov}(\varepsilon_t, \varepsilon_s) = 0, \quad \forall t \neq s$

C'est l'hypothèse de non corrélation ou d'indépendance des erreurs. Une erreur à l'instant t n'a pas d'impact sur les autres erreurs. Lorsque cette hypothèse est violée, on dit que les erreurs sont corrélées ou d'autocorrélation des erreurs.

H5) L'erreur est indépendante de la variable explicative : $\mathrm{Cov}(x_t, \varepsilon_t) = 0 \quad \forall t = 1, \dots, n$.

H6) Lorsque le nombre d'observation n devient grand, les premiers moments empiriques de x (espérance mathématique et variance) sont finis.

Remarque 1.1

Le modèle linéaire de régression simple spécifié en coupe instantanée s'écrit :

$$y_i = \beta_0 + \beta_1 x_i, \quad \forall i = 1, \dots, n \tag{1.2}$$

y_i est la variable à expliquer de l'individu i et x_i la variable explicative de l'individu i.

1.1.1.2 Estimateur des moindres carrés ordinaires (MCO)

Notre objectif est de déterminer les valeurs de β_0 et β_1 en utilisant les informations apportées par l'échantillon. Nous voulons que l'estimation soit la meilleur possible c'est - à - dire la droite de régression doit apporter *au mieux* le nuage de points.

La méthode des moindre carrés ordinaires consiste à minimiser la somme des carrés des erreurs entre les vraies valeurss de y et les valeurs prédites avec le modèle de prédiction. L'estimateur des moindres carrés ordinaires (MCO) des paramètres β_0 et β_1 doit donc répondre au programme

$$\min_{\beta_0, \beta_1} \sum_{t=1}^{t=n} \left(y_t - \beta_0 - \beta_1 x_t\right)^2 \tag{1.3}$$

Ce programme admet son minimum lorsque les dérivées partielles premières par rapport à β_0 et β_1 sont nulles. On obtient les estimateurs suivants :

$$\begin{cases} \hat{\beta}_1 = \dfrac{\mathrm{Cov}(x_t, y_t)}{\mathrm{V}(x_t)} \\ \hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x} \end{cases} \tag{1.4}$$

Preuve

La fonction $\mathcal{Q}(\beta_1, \beta_2) = \sum_{t=1}^{t=n} (y_t - \beta_0 - \beta_1 x_t)^2$ est strictement convexe. Si elle admet un point singulier, celui-ci correspond à l'unique minimum. Annulons les dérivées partielles, nous obtenons un système d'équations appelées « équations normales » :

$$\begin{cases} \dfrac{\partial}{\partial \beta_0} \mathcal{Q}(\beta_0, \beta_1) &= -2 \sum_{t=1}^{t=n} \left(y_t - \hat{\beta}_0 - \hat{\beta}_1 x_t \right) = 0 \\ \dfrac{\partial}{\partial \beta_1} \mathcal{Q}(\beta_0, \beta_1) &= -2 \sum_{t=1}^{t=n} x_t \left(y_t - \hat{\beta}_0 - \hat{\beta}_1 x_t \right) = 0 \end{cases}$$

La première équation donne

$$n\hat{\beta}_0 + \hat{\beta}_1 \sum_{t=1}^{t=n} x_t = \sum_{t=1}^{t=n} y_t$$

et nous avons un estimateur de l'ordonnée à l'origine

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x} \tag{1.5}$$

où $\overline{x} = n^{-1} \sum_{t=1}^{t=n} x_t$. La séconde équation donne

$$\hat{\beta}_0 \sum_{t=1}^{t=n} x_t + \hat{\beta}_1 \sum_{t=1}^{t=n} x_t^2 = \sum_{t=1}^{t=n} x_t y_t$$

En remplaçant $\hat{\beta}_1$ par son expression (1.5) nous avons une première écriture de

$$\hat{\beta}_1 = \frac{\sum_{t=1}^{t=n} x_t y_t - \sum_{t=1}^{t=n} x_t \overline{y}}{\sum_{t=1}^{t=n} x_t^2 - \sum_{t=1}^{t=n} x_t \overline{x}}$$

et en utilisant astucieusement la nullité de la somme $\sum_{t=1}^{t=n} (x_t - \overline{x})$, nous avons d'autres cériture pour l'estimateur de la pente de la droite

$$\hat{\beta}_1 = \frac{\sum_{t=1}^{t=n} (x_t - \overline{x})(y_t - \overline{y})}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \tag{1.6}$$

Remarque 1.2

1. La relation $\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x}$ montre que la droite des moindres carrés ordinaires passe par le centre de gravité du nuage $(\overline{x}, \overline{y})$.
2. L'estimateur $\hat{\beta}_1$ peut aussi s'écrire comme suit :

$$\hat{\beta}_1 = \beta_1 + \frac{\sum_{t=1}^{t=n} (x_t - \overline{x})\,\varepsilon_t}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \tag{1.7}$$

   Si cette décomposition n'est pas intéressante pour le calcul effectif de $\hat{\beta}_1$ puisqu'elle fait intervenir les quantités inconnues β_1 et ε_t, elle l'est par contre pour démontrer des propriéts théoriques des estimateurs (biais et variance). Son avantage est en effet de mettre en exergue la seule source d'aléa du modèle, à savoir les erreurs ε_t.

Exemple 1.1 *Concentration en ozone*

Nous allons traiter les 50 données journalières de concentration en ozone (*cf.* Cornillon et al. (2019)). La variable à expliquer est la concentration en ozone notée `O3` et la variable explicative est la température notée `T12`.

```
# Chargement des données
import pandas as pd
from plydata import *
donnee = pd.read_csv('./donnee/ozone.txt',sep=';',index_col=0)
```

Le tableau 1.1 donne les 10 premières mesures effectuées.

Table 1.1 – 10 données de température à 12 h et teneur en ozone

	O3	T12
19960422	63.6	13.4
19960429	89.6	15.0
19960506	79.0	7.9
19960514	81.2	13.1
19960521	88.0	14.1
19960528	68.4	16.7
19960605	139.0	26.8
19960612	78.2	18.4
19960619	113.8	27.2
19960627	41.8	20.6

Nous allons chercher à expliquer le maximum de `O3` de la journée par la température à 12 h. D'un point de vue pratique, le but de cette régression est double :

1. Ajuster un modèle pour expliquer la concentration en O3 en fonction de T12 ;
2. Prédire les valeurs de concentration en O3 pour de nouvelles valeurs de T12.

Avant toute analyse, il est intéressant de représenter les données. Chaque point du graphique 1.1 représente, pour un jour donné, une mesure de la température à T12 h et le pic d'ozone de la journée.

```
# Nuage des points
from plotnine import *
import matplotlib.pyplot as plt
p = (donnee >> ggplot(aes(x="T12",y="O3"))+geom_point(color="black")+
     labs(x='T12',y='O3'))
print(p)
```

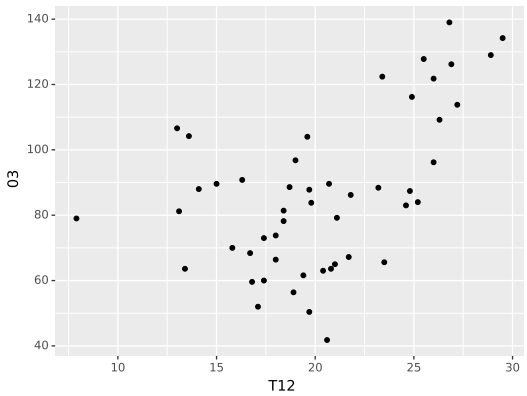


Figure 1.1 – 50 données journalières de température et O3

Ce graphique permet de vérifier visuellement si une régression linéaire est pertinente. Autrement dit il suffit de regarder si le nuage de point s'étire le long d'une droite. Bien qu'ici il semble que le nuage s'étire sur une première droite jusqu'à 22 ou 23°C puis selon une autre droite pour les hautes valeurs de températures, nous pouvons tenter une régression linéaire simple. À l'aide de la librairie statsmodels, on ajuste le modèle de régression :

$$\mathrm{O3}_t = \beta_0 + \beta_1 \mathrm{T12}_t + \varepsilon_t \tag{1.8}$$

```
# Estimation des paramètres
import statsmodels.formula.api as smf
model = smf.ols(formula = 'O3~T12', data=donnee).fit()
model_params = model.summary2().tables[1]
```

Table 1.2 – Coefficients du modèle linéaire : $\mathrm{O3}_t = \beta_0 + \beta_1 \mathrm{T12}_t + \varepsilon_t$

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	31.415	13.0584	2.4057	2e-02	5.1592	57.6707
T12	2.701	0.6266	4.3107	1e-04	1.4412	3.9609

Une fois déterminés les estimateurs $\hat{\beta}_0$ et $\hat{\beta}_1$, nous pouvons estimer la droite de régression par la formule

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x \tag{1.9}$$

On a le modèle ajusté suivant :

$$\widehat{\texttt{O3}} = 31.415 + 2.701\texttt{T12} \tag{1.10}$$

Afin d'examiner la qualité du modèle et des observations, nous traçons la droite ajustée et les observations. Comme il existe une incertitude dans les estimations,nous traçons aussi un intervalle de confiance de la droite (à 95%).

```
p = p + geom_smooth(method='lm',se=True, color='blue')
print(p)
```

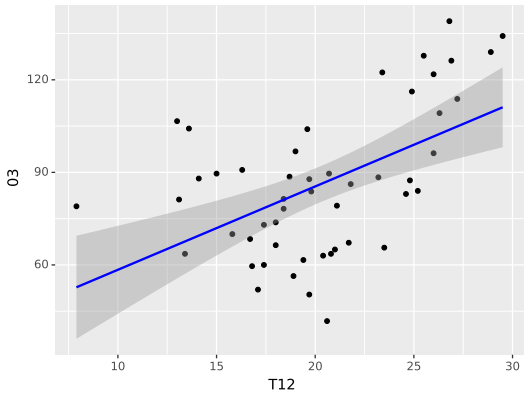


Figure 1.2 – 50 données journalières de température et O3 et l'ajustement linéaire obtenu

Ce graphique permet de vérifier visuellement si une régression est correcte, c'est - à - dire de constater la qualité d'ajustement de notre modèle.

Nous constatons que les observations qui possèdent de faibles valeurs ou de fortes valeurs de températures sont au - dessus de la droite ajustée (Figure 1.2) alors que les observations qui possèdent des valeurs moyennes sont en dessous. Les erreurs ne semblent donc pas identiquement distribuées. Pour s'en assurer il est aussi possible de tracer les résidus. Pour une régression simple, les deux choix sont possibles, mais pour une régression multiple, seul le tracé des résidus sera réalisable. Enfin, l'intervalle de confiance à 95% est éloigné de la droite. Cet intervalle peut être vu comme « le modèle peut être n'importe quelle droite dans cette bande ». Il en découle que la qualité de l'estimation ne semble pas être très bonne.

Cas particulier : modèle sans terme constant

La théorie économique postule parfois des relations dans lesquelles $\beta_0 = 0$: c'est le cas par exemple pour une fonction de production de produit industriel où le facteur de production (unique) nul entraîne une production nulle. L'estimation de β_1 est alors donnée par la formule suivante :

$$\hat{\beta}_1 = \frac{\sum_{t=1}^{t=n} x_t y_t}{\sum_{t=1}^{t=n} x_t^2} \tag{1.11}$$

Nous remarquons qu'il s'agit de l'application de la formule (1.6) dans laquelle $\overline{x}$ et $\overline{y}$ sont nulles. Dans le cas de variables centrées, c'est donc cette formule (1.11) qu'il convient d'employer car le terme constant est nul.

1.1.1.3 Les différentes écritures du modèle : erreur et résidu

Le modèle de régression simple peut s'écrire sous deux formes selon qu'il s'agit du modèle théorique spécifié par l'économiste ou du modèle estimé à partir d'un échantillon.

1. Le modèle théorique spécifié par l'économiste avec ε_t l'erreur inconnue :

$$y_t = \beta_0 + \beta_1 x_t + \varepsilon_t$$

2. Le modèle estimé à partir d'un échantillon d'observations :

$$y_t = \hat{\beta}_0 + \hat{\beta}_1 x_t + \hat{\varepsilon}_t = \hat{y}_t + \hat{\varepsilon}_t$$

avec $\hat{\varepsilon}_t$ le résidu.

Le résidu observé $\hat{\varepsilon}_t$ est donc la différence entre les valeurs observées de la variable à expliquer et les valeurs ajustées à l'aide des estimations des coefficienrs du modèle. $\hat{\varepsilon}_t$ peut donc être calculé suivant la formule :

$$\hat{\varepsilon}_t = y_t - \hat{y}_t \tag{1.12}$$

Proposition 1.1 *Par construction, la somme des résidus est nulle :*

$$\sum_{t=1}^{t=n} \hat{\varepsilon}_t = 0 \tag{1.13}$$

En effet, les résidus sont définis par :

$$\hat{\varepsilon}_t = y_t - \hat{y}_t = y_t - \hat{\beta}_0 - \hat{\beta}_1 x_t = (y_t - \overline{y}) - \hat{\beta}_1 (x_t - \overline{x})$$

En prenant la somme, on obtient :

$$\sum_{t=1}^{t=n} \hat{\varepsilon}_t = \sum_{t=1}^{t=n} (y_t - \overline{y}) - \hat{\beta}_1 \sum_{t=1}^{t=n} (x_t - \overline{x}) = 0$$

Avant de poursuivre, notons que le calcul des estimateurs des moindres carrés est purement déterministe : il ne fait en rien appel aux hypothèses $H2)$, $H3)$ et $H4)$ sur le modèle. Celles - ci vont en fait servir dans la suite à expliciter les propriétés statistiques de ces estimateurs.

1.1.2 Propriétés des estimateurs

Sous les hypothèses $H2)$, $H3)$ et $H4)$ de centrages, décorrélations et homoscédasticités des erreurs ε_t du modèle, on peut déjà donner certaines propriétés des estimateurs $\hat{\beta}_0$ et $\hat{\beta}_1$ des moindres carrés ordinaires.

1.1.2.1 Fonctions linéaires de y_t

Proposition 1.2 *$\hat{\beta}_0$ et $\hat{\beta}_1$ sont des fonctions linéaires de y_t*

En effet, $\hat{\beta}_1$ s'écrit :

$$\hat{\beta}_1 = \frac{\sum_{t=1}^{t=n} (x_t - \overline{x})(y_t - \overline{y})}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} = \frac{\sum_{t=1}^{t=n} y_t (x_t - \overline{x})}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} - \frac{\sum_{t=1}^{t=n} \overline{y} (x_t - \overline{x})}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} = \frac{\sum_{t=1}^{t=n} y_t (x_t - \overline{x})}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2}$$

En posant $\lambda_t = \dfrac{(x_t - \overline{x})}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2}$, on obtient :

$$\hat{\beta}_1 = \sum_{t=1}^{t=n} \lambda_t y_t \tag{1.14}$$

De même, $\hat{\beta}_0$ peut s'écrire comme suit :

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x} = \overline{y} - \overline{x} \sum_{t=1}^{t=n} \lambda_t y_t = \frac{1}{n} \sum_{t=1}^{t=n} y_t - \overline{x} \sum_{t=1}^{t=n} \lambda_t y_t = \sum_{t=1}^{t=n} y_t \left(\frac{1}{n} - \overline{x} \lambda_t \right)$$

En regroupant par rapport à y_t, on obtient :

$$\hat{\beta}_0 = \sum_{t=1}^{t=n} \mu_t y_t \tag{1.15}$$

avec $\mu_t = \left(\dfrac{1}{n} - \overline{x} \lambda_t \right)$.

Proposition 1.3 *Propriétés de λ_t et μ_t*

Propriété 1.1 $\sum_{t=1}^{t=n} \lambda_t = 0$

$$\lambda_t = \frac{(x_t - \overline{x})}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \quad \Longrightarrow \quad \sum_{t=1}^{t=n} \lambda_t = \frac{\sum_{t=1}^{t=n} (x_t - \overline{x})}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} = 0$$

Propriété 1.2 $\sum_{t=1}^{t=n} \lambda_t x_t = 1$

$$\sum_{t=1}^{t=n} \lambda_t x_t = \frac{\sum_{t=1}^{t=n} x_t^2 - \overline{x} \sum_{t=1}^{t=n} x_t}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} = \frac{\sum_{t=1}^{t=n} x_t^2 - n\overline{x}^2}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} = \frac{\sum_{t=1}^{t=n} (x_t - \overline{x})^2}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} = 1$$

Propriété 1.3 $\sum_{t=1}^{t=n} \lambda_t^2 = \frac{1}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2}$

$$\sum_{t=1}^{t=n} \lambda_t^2 = \frac{\sum_{t=1}^{t=n} (x_t - \overline{x})^2}{\left[\sum_{t=1}^{t=n} (x_t - \overline{x})^2\right]^2} = \frac{1}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2}$$

Propriété 1.4 $\sum_{t=1}^{t=n} \mu_t = 1$

$$\sum_{t=1}^{t=n} \mu_t = \sum_{t=1}^{t=n} \left(\frac{1}{n} - \overline{x}\lambda_t\right) = n\left(\frac{1}{n}\right) - \overline{x} \underbrace{\sum_{t=1}^{t=n} \lambda_t}_{=0} = 1$$

Propriété 1.5 $\sum_{t=1}^{t=n} \mu_t x_t = 0$

$$\sum_{t=1}^{t=n} \mu_t x_t = \sum_{t=1}^{t=n} \left(\frac{x_t}{n} - \overline{x}\lambda_t x_t\right) = \underbrace{\sum_{t=1}^{t=n} \frac{x_t}{n}}_{=\overline{x}} - \overline{x} \underbrace{\sum_{t=1}^{t=n} \lambda_t x_t}_{=1} = 0$$

Propriété 1.6 $\sum_{t=1}^{t=n} \mu_t^2 = \frac{1}{n} + \frac{\overline{x}^2}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2}$

$$\sum_{t=1}^{t=n} z_t^2 = \sum_{t=1}^{t=n} \left(\frac{1}{n} - \overline{x}\lambda_t\right)^2 = \sum_{t=1}^{t=n} \frac{1}{n^2} - 2\frac{\overline{x}}{n} \sum_{t=1}^{t=n} \lambda_t + \overline{x}^2 \sum_{t=1}^{t=n} \lambda_t^2 = \frac{1}{n} + \frac{\overline{x}^2}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2}$$

1.1.2.2 Estimateurs sans biais

Proposition 1.4 *Les estimateurs $\hat{\beta}_0$ et $\hat{\beta}_1$ sont des estimateurs sans biais de β_0 et β_1, c'est - à - dire que*

$$\mathbb{E}\left(\hat{\beta}_0\right) = \beta_0 \quad et \quad \mathbb{E}\left(\hat{\beta}_1\right) = \beta_1 \tag{1.16}$$

Pour β_0, on part de l'équation (1.15), on obtient :

$$\mathbb{E}\left(\hat{\beta}_0\right) = \mathbb{E}\left(\sum_{t=1}^{t=n} \mu_t y_t\right) = \mathbb{E}\left[\sum_{t=1}^{t} \mu_t \left(\beta_0 + \beta_1 x_t + \varepsilon_t\right)\right] = \beta_0 + \sum_{t=1}^{t=n} \mu_t \mathbb{E}\left(\varepsilon_t\right) = \beta_0$$

On adopte la même logique pour β_1 en prenant la relation (1.14), on obtient :

$$\mathbb{E}\left(\hat{\beta}_1\right) = \mathbb{E}\left(\sum_{t=1}^{t=n} \lambda_t y_t\right) = \mathbb{E}\left[\sum_{t=1}^{t} \lambda_t \left(\beta_0 + \beta_1 x_t + \varepsilon_t\right)\right] = \beta_1 + \sum_{t=1}^{t=n} \lambda_t \mathbb{E}\left(\varepsilon_t\right) = \beta_1$$

1.1.2.3 Estimateurs convergents

Proposition 1.5 *La variance de $\hat{\beta}_0$ est donnée par*

$$\sigma^2_{\hat{\beta}_0} = V\left(\hat{\beta}_0\right) = \frac{\sigma^2_\varepsilon \sum_{t=1}^{t=n} x_t^2}{n \sum_{t=1}^{t=n} \left(x_t - \overline{x}\right)^2} = \sigma^2_\varepsilon \left(\frac{1}{n} + \frac{\overline{x}^2}{\sum_{t=1}^{t=n} \left(x_t - \overline{x}\right)^2}\right) \tag{1.17}$$

et celle de $\hat{\beta}_1$

$$\sigma^2_{\hat{\beta}_1} = V\left(\hat{\beta}_1\right) = \frac{\sigma^2_\varepsilon}{\sum_{t=1}^{t=n} \left(x_t - \overline{x}\right)^2} \tag{1.18}$$

Leur covariance vaut :

$$Cov\left(\hat{\beta}_0, \hat{\beta}_1\right) = -\frac{\sigma^2_\varepsilon \overline{x}}{\sum_{t=1}^{t=n} \left(x_t - \overline{x}\right)^2} \tag{1.19}$$

En effet, pour $\hat{\beta}_0$, on part de la relation entre $\hat{\beta}_0$ et β_0 :

$$\begin{aligned}\sigma^2_{\widehat{\beta}_0} &= \mathrm{V}\left(\hat{\beta}_0\right) = \mathbb{E}\left[\left(\hat{\beta}_0 - \beta_0\right)^2\right] = \mathbb{E}\left[\left(\sum_{t=1}^{t=n}\lambda_t\varepsilon_t\right)^2\right] = \mathbb{E}\left[\sum_{t=1}^{t=n}\lambda_t^2\varepsilon_t^2 + 2\sum_{t\neq s}\lambda_t\lambda_s\varepsilon_t\varepsilon_s\right] \\ &= \sum_{t=1}^{t=n}\lambda_t^2\underbrace{\mathbb{E}\left(\varepsilon_t^2\right)}_{=\sigma^2_\varepsilon} + 2\sum_{t\neq s}\lambda_t\lambda_s\underbrace{\mathbb{E}\left(\varepsilon_t\varepsilon_s\right)}_{=0} = \sigma^2_\varepsilon\left(\frac{1}{n} + \frac{\overline{x}^2}{\sum_{t=1}^{t=n}\left(x_t - \overline{x}\right)^2}\right)\end{aligned}$$

On applique le même raisonnement pour $\hat{\beta}_1$, on part de la relation entre $\hat{\beta}_1$ et β_1 :

$$\begin{aligned}\sigma^2_{\widehat{\beta}_1} &= \mathrm{V}\left(\hat{\beta}_1\right) = \mathbb{E}\left[\left(\hat{\beta}_1 - \beta_1\right)^2\right] = \mathbb{E}\left[\left(\sum_{t=1}^{t=n}\mu_t\varepsilon_t\right)^2\right] = \mathbb{E}\left[\sum_{t=1}^{t=n}\mu_t^2\varepsilon_t^2 + 2\sum_{t\neq s}\mu_t\mu_s\varepsilon_t\varepsilon_s\right] \\ &= \sum_{t=1}^{t=n}\mu_t^2\underbrace{\mathbb{E}\left(\varepsilon_t^2\right)}_{=\sigma^2_\varepsilon} + 2\sum_{t\neq s}\mu_t\mu_s\underbrace{\mathbb{E}\left(\varepsilon_t\varepsilon_s\right)}_{=0} = \frac{\sigma^2_\varepsilon}{\sum_{t=1}^{t=n}\left(x_t - \overline{x}\right)^2}\end{aligned}$$

Enfin, pour la covariance entre les deux estimateurs :

$$\mathrm{Cov}\left(\hat{\beta}_0, \hat{\beta}_1\right) = \mathrm{Cov}\left(\overline{y} - \hat{\beta}_1\overline{x}, \hat{\beta}_1\right) = \underbrace{\mathrm{Cov}\left(\overline{y}, \hat{\beta}_1\right)}_{=0} - \overline{x}\mathrm{V}\left(\hat{\beta}_1\right) = -\frac{\overline{x}\sigma^2_\varepsilon}{\sum_{t=1}^{t=n}\left(x_t - \overline{x}\right)^2}$$

Proposition 1.6 *Les estimateurs $\hat{\beta}_0$ et $\hat{\beta}_1$ sont des estimateurs convergents, c'est - à - dire que*

$$\lim_{n\to+\infty} V\left(\hat{\beta}_0\right) = 0 \quad \textit{et} \quad \lim_{n\to+\infty} V\left(\hat{\beta}_1\right) = 0 \tag{1.20}$$

Il suffit de remarquer que, pour $\hat{\beta}_0$, on a

$$\mathrm{V}\left(\hat{\beta}_0\right) = \frac{\sigma^2_\varepsilon}{n}\left(1 + \frac{\overline{x}^2}{\mathrm{V}(x)}\right) \quad \Longrightarrow \quad \lim_{n\to+\infty}\mathrm{V}\left(\hat{\beta}_0\right) = \lim_{n\to+\infty}\frac{\sigma^2_\varepsilon}{n}\left(1 + \frac{\overline{x}^2}{\mathrm{V}(x)}\right) = 0$$

Le même raisonnement s'applique à $\hat{\beta}_1$:

$$\mathrm{V}\left(\hat{\beta}_1\right) = \frac{\sigma^2_\varepsilon}{n\mathrm{V}\left(x\right)} \quad \Longrightarrow \quad \lim_{n\to+\infty}\mathrm{V}\left(\hat{\beta}_1\right) = \lim_{n\to+\infty}\frac{\sigma^2_\varepsilon}{n\mathrm{V}\left(x\right)} = 0$$

Remarque 1.3

Dans la réalité, on ne connaît $\mathrm{V}\left(\hat{\beta}_0\right)$ et $\mathrm{V}\left(\hat{\beta}_1\right)$ que par leur estimateur $\hat{\mathrm{V}}\left(\hat{\beta}_0\right)$ et $\hat{\mathrm{V}}\left(\hat{\beta}_1\right)$. Ainsi, la matrice de variance - covariance estimée est :

$$\hat{\sigma}^2_{\hat{\beta}} = \hat{\sigma}^2_{\varepsilon} \begin{pmatrix} \frac{1}{n} + \frac{\overline{x}^2}{\sum_{t=1}^{n} (x_t - \overline{x})^2} & -\frac{\overline{x}}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \\ -\frac{\overline{x}}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} & -\frac{1}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \end{pmatrix} \tag{1.21}$$

où $\hat{\sigma}^2_{\varepsilon}$ est l'estimateur sans biais et convergent de σ^2_{ε} avec :

$$\hat{\sigma}^2_{\varepsilon} = \frac{1}{n-2} \sum_{t=1}^{t=n} \left(y_t - \hat{\beta}_0 - \hat{\beta}_1 x_t \right)^2 \tag{1.22}$$

Proposition 1.7 *La statistique*

$$\hat{\sigma}^2_{\varepsilon} = \frac{1}{n-2} \sum_{t=1}^{t=n} \left(y_t - \hat{\beta}_0 - \hat{\beta}_1 x_t \right)^2 \tag{1.23}$$

est un estimateur sans biais de σ^2_{ε}.

Preuve

Réécrivons les résidus en constatant que $\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x}$ et $\beta_0 = \overline{y} - \hat{\beta}_1 \overline{x} - \overline{\varepsilon}$, ce qui donne :

$$\begin{aligned} \hat{\varepsilon}_t &= y_t - \hat{y}_t = \beta_0 + \beta_1 x_t + \varepsilon_t - \hat{\beta}_0 - \hat{\beta}_1 x_t \\ &= \overline{y} - \hat{\beta}_1 \overline{x} - \overline{\varepsilon} + \beta_1 x_t + \varepsilon_t - \overline{y} - \hat{\beta}_1 \overline{x} - \hat{\beta}_1 x_t \\ &= \left(\beta_1 - \hat{\beta}_1 \right) (x_t - \overline{x}) + (\varepsilon_t - \overline{\varepsilon}) \end{aligned}$$

En développant et en nous servant de l'écriture vue plus haut :

$$\hat{\beta}_1 = \beta_1 + \frac{\sum_{t=1}^{t=n} (x_t - \overline{x}) \varepsilon_t}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2}$$

Nous avons :

$$\begin{aligned} \sum_{t=1}^{t=n} \hat{\varepsilon}^2_t &= \left(\beta_1 - \hat{\beta}_1 \right)^2 \sum_{t=1}^{t=n} (x_t - \overline{x})^2 + \sum_{t=1}^{t=n} (\varepsilon_t - \overline{\varepsilon})^2 + 2 \left(\beta_1 - \hat{\beta}_1 \right) \sum_{t=1}^{t=n} (x_t - \overline{x}) (\varepsilon_t - \overline{\varepsilon}) \\ &= \left(\beta_1 - \hat{\beta}_1 \right)^2 \sum_{t=1}^{t=n} (x_t - \overline{x})^2 + \sum_{t=1}^{t=n} (\varepsilon_t - \overline{\varepsilon})^2 - 2 \left(\beta_1 - \hat{\beta}_1 \right)^2 \sum_{t=1}^{t=n} (x_t - \overline{x})^2 \end{aligned}$$

Prenons - en l'espérance :

$$\mathbb{E}\left(\sum_{t=1}^{t=n}\hat{\varepsilon}_t^2\right) = \mathbb{E}\left(\sum_{t=1}^{t=n}\left(\varepsilon_t - \overline{\varepsilon}\right)^2\right) - \sum_{t=1}^{t=n}\left(x_t - \overline{x}\right)^2 \mathrm{V}\left(\hat{\beta}_1\right) = (n-2)\,\sigma_\varepsilon^2$$

Bien sûr, lorsque n est grand, cet estimateur diffère très peu de l'estimateur empirique de la variance des résidus, à savoir $n^{-1}\sum_{t=1}^{t=n}\hat{\varepsilon}_t^2$.

L'estimation de la variance $\hat{\sigma}_\varepsilon^2$ vaut

```
# variance de l'erreur
scale = model.scale
print("Variance de l'erreur : %.4f"%(scale))
```

```
## Variance de l'erreur : 420.4041
```

On retourne les variances et covariances des estimateurs :

```
# Matrice des variances-covariances des estimateurs
covparams = model.cov_params()
```

Table 1.3 – Matrice des variances - covariances des estimateurs

	Intercept	T12
Intercept	170.522699	-7.9780816
T12	-7.978082	0.3926221

1.1.3 Équation d'analyse de la variance

1.1.3.1 Équation de décomposition de la variance

Cette équation signifie que la variabilité totale est égale à la variabilité expliquée plus la variabilité résiduelle et permet de juger de la qualité de l'ajustement d'un modèle. Elle s'écrit :

$$\begin{array}{lcl} \sum_{t=1}^{t=n}(y_t - \overline{y})^2 & = & \sum_{t=1}^{n}(\hat{y}_t - \overline{y})^2 + \sum_{t=1}^{n}(y_t - \hat{y}_t)^2 \\ SCT & = & SCE + SCR \end{array}$$

où SCT (respectivement SCE et SCR) représente la somme des carrés totale (respectivement expliquée par la méthode et résiduelle).

La tableau 1.4 donne le résumé sur la décomposition de la variance dans la cas d'un modèle de régression linéaire simple.

Les degrés de liberté correspondent au nombre de valeurs que nous pouvons choisir arbitrairement (par exemple, pour la variabilité totale, connaissant $n-1$ valeurs, nous pourrons en déduire la $n-$ième, puisque nous connaissons la moyenne $\overline{y}$).

Table 1.4 – Tableau d'analyse de la variance

Source de variation	Somme des carrés (SC)	Degré de liberté (ddl)	Carrés moyens (SC/ddl)
Exogène (x)	SCE	1	$SCE/1$
Résiduelle	SCR	$n-2$	$SCR/n-2$
Totale	SCT	$n-1$	

```
# Somme des carrés totale - SCT
n = donnee.shape[0]
SCT = n*np.var(donnee["O3"])
# Somme des carrés expliquée
SCE = n*np.var(model.fittedvalues)
# Somme des carrés résiduelle
SCR = n*np.var(model.resid)
var_eq = pd.DataFrame({
  "Source de variance" : ["Exogène (x)","Résiduelle","Totale"],
  "Somme des carrés (SC)" : np.around(np.array([SCE, SCR,SCT]),4),
  "Degré de liberté (ddl)" : np.around(np.array([1, n-2, n-1]),4),
  "Carré moyen (SC/ddl)" : np.around(np.array([SCE/1,SCE/(n-2),np.nan]),4)
})
```

Table 1.5 – Tableau d'analyse de la variance - Données Ozone

Source de variance	Somme des carrés (SC)	Degré de liberté (ddl)	Carré moyen (SC/ddl)
Exogène (x)	7811.825	1	7811.8247
Résiduelle	20179.395	48	162.7463
Totale	27991.220	49	

La formule de décomposition de la variance permet d'introduire le coefficient de détermination de façon naturelle.

1.1.3.2 Coefficient de détermination R^2

C'est un indicateur permettant de mesurer la qualité d'ajustement d'un modèle. Il est défini par la relation :

$$R^2 = \frac{SCE}{SCT} = 1 - \frac{SCR}{SCT} \tag{1.24}$$

En effet, plus la variance expliquée est proche de la variance totale, meilleure est l'ajsutement du nuage de points par la droite des moindres carrés.

Pour un modèle linéaire de régression simple, on a la formule suivante :

$$R^2 = \hat{\beta}_1^2 \frac{\sum_{t=1}^{t=n}(x_t - \overline{x})^2}{\sum_{t=1}^{t=n}(y_t - \overline{y})^2} = \hat{\beta}_1^2 \frac{\mathrm{V}(x)}{\mathrm{V}(y)} \tag{1.25}$$

Propriété 1.7

De façon schématique, on peut différencier les cas suivants :

— Si $R^2 = 1$, le modèle explique tout. Les points de l'échantillon sont parfaitement alignés sur la droite des moindres carrés.

— Si $R^2 = 0$, cela veut dire que $\sum_{t=1}^{t=n} (\hat{y}_t - \overline{y}) = 0$, donc $\hat{y}_t = \overline{y}$ pour tout t. Le modèle de régression linéaire est inadapté puisqu'on ne modélise rien de mieux que la moyenne.

De façon générale, l'interprétation est la suivante : le modèle de régression linéaire permet d'expliquer $100 \times R^2\%$ de la variance totale des données.

```
# Coefficient de détermination
r2squared = model.rsquared
print('Coefficient de détermination : %.4f'%(r2squared))
```

```
## Coefficient de détermination : 0.2791
```

On obtient un score de 28% avec notre modèle. Cette valeur du R^2 est faible : peut-être qu'une régression linéaire simple n'est-elle pas adaptée ici.

Remarque 1.4

On peut aussi voir R^2 comme le carré du coefficient de corrélation empirique entre les x_t et les y_t :

$$R^2 = \left(\frac{\sum_{t=1}^{t=n} (x_t - \overline{x})(y_t - \overline{y})}{\sqrt{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \sqrt{\sum_{t=1}^{t=n} (y_t - \overline{y})^2}} \right)^2 = \rho_{x,y}^2 \tag{1.26}$$

En effet, en partant de l'equation (1.25), on a :

$$\begin{aligned} R^2 &= \hat{\beta}_1^2 \frac{\sum_{t=1}^{t=n} (x_t - \overline{x})^2}{\sum_{t=1}^{t=n} (y_t - \overline{y})^2} = \left(\frac{\sum_{t=1}^{t=n} (x_t - \overline{x})(y_t - \overline{y})}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \right)^2 \times \frac{\sum_{t=1}^{t=n} (x_t - \overline{x})^2}{\sum_{t=1}^{t=n} (y_t - \overline{y})^2} \\ &= \left(\frac{\sum_{t=1}^{t=n} (x_t - \overline{x})(y_t - \overline{y})}{\sqrt{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \sqrt{\sum_{t=1}^{t=n} (y_t - \overline{y})^2}} \right)^2 = \rho_{x,y}^2 \end{aligned}$$

1.1.4 Prévision

Un des buts de la régression est de proposer des prévisions pour la variable à expliquer y. Soit x_{n+1}, la réalisation de la variable exogène à l'instant $n+1$, alors la valeur prévue pour la variable endogène à l'instant $n+1$ est :

$$\widehat{y}_{n+1}^{p} = \widehat{\beta}_0 + \widehat{\beta}_1 x_{n+1} \tag{1.27}$$

Toutefois, la vraie réalisation de la variable endogène à l'instant θ sera donnée par :

$$y_{n+1} = \beta_0 + \beta_1 x_{n+1} + \varepsilon_{n+1} \tag{1.28}$$

avec $\mathbb{E}(\varepsilon_{n+1}) = 0$, $\mathrm{V}(\varepsilon_{n+1}) = \sigma_\varepsilon^2$ et $\mathrm{Cov}(\varepsilon_{n+1}, \varepsilon_t) = 0$ pour $t = 1, \ldots, n$.

En utilisant la notation $\widehat{y}_{n+1}^{p}$, nous souhaitons insister sur la notion de prévision : la valeur pour laquelle nous effectuons la prévision, ici la $(n+1)^e$, n'a pas servi dans le calcul des estimateurs. Remarquons que cette quantité sera différente de la valeur ajustée, notée $\widehat{y}_t$, qui elle fait intervenir la t^e observation.

Deux types d'erreurs vont entacher notre prévision, l'une due à la non - connaissance de ε_{n+1} et l'autre due à l'estimation des paramètres.

Proposition 1.8 *La variance de la valeur prévue de $\widehat{y}_{n+1}^{p}$ vaut*

$$V(\widehat{y}_{n+1}^{p}) = \sigma_\varepsilon^2 \left(\frac{1}{n} + \frac{(x_{n+1} - \overline{x})^2}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \right) \tag{1.29}$$

En effet, en partant de l'équation (1.27), on a :

$$\begin{aligned}
\mathrm{V}(\widehat{y}_{n+1}^{p}) &= \mathrm{V}\left(\widehat{\beta}_0 + \widehat{\beta}_1 x_{n+1}\right) = \mathrm{V}\left(\widehat{\beta}_0\right) + x_{n+1}^2 \mathrm{V}\left(\widehat{\beta}_1\right) + 2x_{n+1}\mathrm{Cov}\left(\widehat{\beta}_0, \widehat{\beta}_1\right) \\
&= \sigma_\varepsilon^2 \left(\frac{1}{n} + \frac{\overline{x}^2}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \right) + x_{n+1}^2 \times \frac{\sigma_\varepsilon^2}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} - 2x_{n+1} \times \frac{\sigma_\varepsilon^2 \overline{x}}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \\
&= \sigma_\varepsilon^2 \left(\frac{1}{n} + \frac{(x_{n+1} - \overline{x})^2}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \right)
\end{aligned}$$

La variance de $\widehat{y}_{n+1}^{p}$ nous donne une idée de la stabilité de l'estimation. En prévision, on s'intéresse généralement à l'erreur que l'on commet entre la vraie valeur à prévoir y_{n+1} et celle que l'on prévoit $\widehat{y}_{n+1}^{p}$. L'erreur peut être simplement résumée par la différence entre ces deux valeurs, c'est ce que nous appelerons **l'erreur de prévision** définie par :

$$\widehat{\varepsilon}_{n+1}^{p} = y_{n+1} - \widehat{y}_{n+1}^{p} \tag{1.30}$$

Proposition 1.9 (erreur de prévision) *L'erreur de prévision, définie par* $\hat{\varepsilon}^p_{n+1} = y_{n+1} - \hat{y}^p_{n+1}$ *satisfait les propriétés suivantes :*

$$\begin{cases} \mathbb{E}\left(\hat{\varepsilon}^p_{n+1}\right) & = 0 \\ V\left(\hat{\varepsilon}^p_{n+1}\right) & = \sigma^2_\varepsilon \left(1 + \dfrac{1}{n} + \dfrac{(x_{n+1} - \overline{x})^2}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \right) \end{cases} \tag{1.31}$$

Pour l'espérance, il suffit d'utiliser le fait que ε_{n+1} est centrée et que les estimateurs $\hat{\beta}_0$ et $\hat{\beta}_1$ sont sans biais :

$$\mathbb{E}\left(\hat{\varepsilon}^p_{n+1}\right) = \mathbb{E}\left(\beta_0 - \hat{\beta}_0\right) + \mathbb{E}\left(\beta_1 - \hat{\beta}_1\right) x_{n+1} + \mathbb{E}\left(\varepsilon_{n+1}\right) = 0$$

Nous obtenons la variance de l'erreur de prévision en nous servant du fait que y_{n+1} est fonction de ε_{n+1} seulement tandis que $\hat{y}^p_{n+1}$ est fonction des autres erreurs $(\varepsilon_t)_{1 \leq t \leq n}$. On obtient bien :

$$\mathrm{V}\left(\hat{\varepsilon}^p_{n+1}\right) = \sigma^2_\varepsilon \left(1 + \frac{1}{n} + \frac{(x_{n+1} - \overline{x})^2}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \right)$$

Remarque 1.5

La variance augmente lorsque x_{n+1} s'éloigne du centre de gravité du nuage. Autrement dit, faire de la prévision lorsque x_{n+1} est « loin » de $\overline{x}$ est périlleux, puisque la variance de l'erreur de prévision peut être très grande ! Ceci s'explique intuitivement par le fait que plus une observation x_{n+1} est éloignée de la moyenne $\overline{x}$ et moins on a d'information sur elle.

Remarque 1.6

Mais, on ne connaît cette variance que par son estimateur défini par :

$$\hat{\mathrm{V}}(\hat{\varepsilon}_{n+1}) = \hat{\sigma}^2_\varepsilon \left(1 + \frac{1}{n} + \frac{(x_{n+1} - \overline{x})^2}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \right) \tag{1.32}$$

```
# Prédiction
forecast = model.get_prediction(exog=donnee['T12'])
prediction = pd.DataFrame(forecast.predicted,columns = ["forecast"])
DPrim = (donnee >> select("O3", "T12")).reset_index(drop=True)
prediction = pd.concat([DPrim,prediction],axis=1)
```

Table 1.6 – Prévision sur les valeurs observées

O3	T12	forecast
63.6	13.4	67.609
89.6	15.0	71.930
79.0	7.9	52.753
81.2	13.1	66.799
88.0	14.1	69.500
68.4	16.7	76.522

```
# Représentation graphique
print((prediction >>ggplot(aes(x = "T12",y = "O3"))+geom_point(color="black")+
       geom_line(aes(y="forecast"),color="black")+labs(x="T12",y="O3")))
```

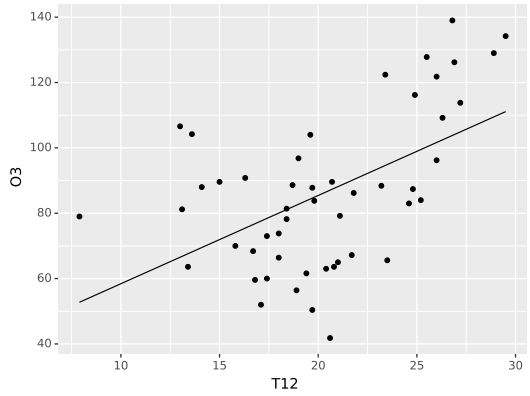


Figure 1.3 – Droite de régression et prévision

1.2 Inférence statistique

Jusqu'à présent, nous avons pu, en choisissant une fonction de coût quadratique, ajuster un modèle de régression, à savoir calculer $\widehat{\beta}_0$ et $\widehat{\beta}_1$. Grâce aux coefficients estimés, nous pouvons donc prédire, pour chaque nouvelle valeur x_{n+1} une valeur de la varianble àexpliquer $\widehat{y}_{n+1}^p$ qui est tout simplement le point sur la droite ajustée correspondant à l'abscisse x_{n+1}. En ajoutant les hypothèses $H3)$ et $H4)$, nous avons pu calculer l'espérance et la variance des estimateurs. Ces propriétés permettent d'appréhender de manière grossidère la qualité des estimateurs proposés.

Cependant, nous souhaitons en général connaître la loi des estimateurs afin de calculer des intervalles ou des régions de confiance ou effectuer des tests. Il faut donc introduire une hypothèse supplémentaire concernant la loi des ε_t. L'hypothèse $H7)$ est définie comme suit :

$$H7) \quad \varepsilon_t \sim \mathcal{N}\left(0, \sigma_\varepsilon^2\right) \quad \text{et} \quad \text{les } \varepsilon_t \text{ sont indépendants}$$

où $\mathcal{N}\left(0, \sigma_\varepsilon^2\right)$ est une loi normale d'espérance nulle et de variance σ_ε^2. Le modèle de régression devient un modèle paramétrique, où les paramètres $\left(\beta_0, \beta_1, \sigma_\varepsilon^2\right)$ sont à valeurs dans $\mathbb{R} \times \mathbb{R} \times \mathbb{R}_+^*$. La loi des ε_t étant connue, les lois des y_t s'en déduisent :

$$\forall t \in \{1, \dots, n\} \quad y_t \sim \mathcal{N}\left(\beta_0 + \beta_1 x_t, \sigma_\varepsilon^2\right)$$

et les y_t sont mutuellement indépendants puisque les ε_t le sont. Nous pouvons donc calculer la vraisemblance de l'échantillon et les estimateurs qui maximisent cette vraisemblance.

1.2.1 Estimation par maximum de vraisemblance (MV)

A côté de la méthode des moindres carrés ordinaires (MCO), la méthode du maximum de vraisemblance (*Maximum Likelihood Method*, en anglais) (*cf.* Jack and John (1999)) permet également d'estimer les paramètres d'un modèle de régression, sous l'hypothèse que la vraie loi de distribution desdits paramètres est connue. Si le principe pour les MCO est de trouver le paramètre qui minimise la somme des carrés des erreurs, la méthode du maximum de vraisemblance cherche par contre à trouver le paramètre à même (ayant une forte probabilité) de reproduire les vraies valeurs de l'échantillon.

Nous supposons désormais que les erreurs suivent une loi normale $\varepsilon_t \sim \mathcal{N}(0, \sigma_\varepsilon^2)$. Ce qui implique que $y_t \sim \mathcal{N}\left(\beta_0 + \beta_1 x_t, \sigma_\varepsilon^2\right)$. La vraisemblance est la densité de l'échantillon vue comme fonction des paramètres. Grâce à l'indépendance des erreurs, les observations sont indépendantes et la vraisemblance s'écrit :

$$\mathrm{L}\left(y_1, \dots, y_n; \beta_0, \beta_1, \sigma_\varepsilon^2\right) = \prod_{t=1}^{t=n} f(y_t) = \left(\frac{1}{2\pi\sigma_\varepsilon^2}\right)^{n/2} \exp\left(-\frac{1}{2\sigma_\varepsilon^2} \sum_{t=1}^{t=n} \left(y_t - \beta_0 - \beta_1 x_t\right)^2\right) \tag{1.33}$$

Lorsqu'on effectue une transformation logarithmique de la fonction de vraisemblance ci-dessus, l'on obtient la fonction dite log-vraisemblance qui servira de base à l'estimation des paramètres $\hat{\beta}_0, \hat{\beta}_1$ et $\hat{\sigma}_\varepsilon^2$. La log-vraisemblance s'écrit :

$$\mathcal{L}(\beta_0, \beta_1, \sigma_\varepsilon^2) = -\frac{n}{2} \ln\left(\sigma_\varepsilon^2\right) - \frac{n}{2} \ln\left(2\pi\right) - \frac{1}{2\sigma_\varepsilon^2} \sum_{t=1}^{t=n} \left(y_t - \beta_0 - \beta_1 x_t\right)^2 \tag{1.34}$$

Pour estimer $\hat{\beta}_0, \hat{\beta}_1$ et $\hat{\sigma}_\varepsilon^2$ par maximum de vraisemblance, la démarche consiste à maximiser la fonction log-vraisemblance ci-dessus, ce qui revient à annuler ses dérivées premières par rapport aux paramètres β_0, β_1 et σ_ε^2. On about à :

$$\begin{cases} \hat{\beta}_1^{\text{EMV}} = \dfrac{\text{Cov}(x_t, y_t)}{\text{V}(x_t)} \\ \hat{\beta}_0^{\text{EMV}} = \overline{y} - \hat{\beta}_1^{\text{EMV}} \overline{x} \\ \hat{\sigma}_{\varepsilon,\text{EMV}}^2 = \dfrac{1}{n} \displaystyle\sum_{t=1}^{t=n} \left(y_t - \hat{\beta}_0^{\text{EMV}} - \hat{\beta}_1^{\text{EMV}} x_t\right)^2 \end{cases}$$

Calculons les dérivées par rapport à β_0, β_1 et σ_ε^2 :

$$\begin{cases} \dfrac{\partial}{\partial\beta_0}\mathcal{L}\left(\beta_0,\beta_1,\sigma_\varepsilon^2\right) &= -\dfrac{1}{\sigma_\varepsilon^2}\displaystyle\sum_{t=1}^{t=n}\left(y_t-\beta_0-\beta_1x_t\right)^2 \\ \dfrac{\partial}{\partial\beta_1}\mathcal{L}\left(\beta_0,\beta_1,\sigma_\varepsilon^2\right) &= -\dfrac{1}{\sigma_\varepsilon^2}\displaystyle\sum_{t=1}^{t=n}x_t\left(y_t-\beta_0-\beta_1x_t\right)^2 \\ \dfrac{\partial}{\partial\sigma_\varepsilon^2}\mathcal{L}\left(\beta_0,\beta_1,\sigma_\varepsilon^2\right) &= -\dfrac{n}{2\sigma_\varepsilon^2}+\dfrac{1}{2\sigma_\varepsilon^4}\displaystyle\sum_{t=1}^{t=n}\left(y_t-\beta_0-\beta_1x_t\right)^2 \end{cases}$$

Les deux premières expressions sont identiques aux équations normales fournies par les MCO. Par contre, la troisième équation est différente de celle obtenue avec les MCO. En effet, on a la relation :

$$\widehat{\sigma}^2_{\varepsilon,\text{EMV}} = \frac{n-2}{n}\widehat{\sigma}^2_{\varepsilon,\text{MCO}} \tag{1.35}$$

Puisque l'estimation MCO de la variance des erreurs est sans biais, celui du maximum de vraisemblance est biaisé, mais reste convergent. Cette dernière propriété garantit la minimisation du biais avec l'accroissement de la taille de l'échantillon.

On définit notre fonction log-vraisemblance en prenant l'opposé de cette fonction.

```
# Fonction log-vraisemblance
import numpy as np
def logvraisemblance(parameters):
  beta0 = parameters[0]
  beta1 = parameters[1]
  sigma = parameters[2]
  for i in np.arange(0, len(x)):
    y_exp = beta0 + beta1*x
    loglik = (len(x)/2 * np.log(2 * np.pi) + len(x)/2 * np.log(sigma ** 2) +
              1 /(2 * sigma ** 2) * sum((y - y_exp) ** 2))
  return loglik
```

On estime nos paramètres grâce à la fonction minimize du package Scipy. Cette fonction recherche le minimum d'une fonction objectif.

```
# Estimation des paramètres par maximum de vraisemblance
from scipy.optimize import minimize
x = donnee.T12.values
y = donnee.O3.values
opt = minimize(logvraisemblance, np.array([1,1,1]), method="L-BFGS-B")
emv_params = pd.DataFrame({"Intercept": opt.x[0],"T12" : opt.x[1],
                           "scale": opt.x[2]**2},index=["EMV"])
```

Table 1.7 – Estimateurs par maximum de vraisemblance

	Intercept	T12	scale
EMV	31.4151	2.701	403.5885

On voit que les valeurs estimées obtenues sont identiques pour les deux premiers paramètres, sauf celle liée à l'estimation de la variance de l'erreur.

Avant de passer aux lois des estimateurs et aux intervalles de confiance qui s'en déduisent, faisons quelques rappels sur les lois usuelles dans ce contexte.

1.2.2 Rappels sur les lois de probabilités usuelles

Outre la loi gaussienne, trois lois seront d'usage dans la suite : la loi du χ^2, laloi de Student et la loi de Fisher.

1.2.2.1 La loi du χ^2

Définition 1.1 *Soit $X_1, \dots, X_n$ des variables aléatoires indépendantes et identiquement distribuées suivant une loi normale centrée et réduite. La loi d ela variable*

$$X = \sum_{i=1}^{i=n} X_i^2 \tag{1.36}$$

est appelée loi du χ^2 à n degrés de liberté, noté $X \sim \chi^2_n$.

On a $\mathbb{E}(X) = n$ et $\mathrm{V}(X) = 2n$. Lorsque n est grand, on sait par le Téorème Central Limite que X suit approximativement une loi normale de moyenne n et de variance $2n$: $X \approx \mathcal{N}(n, 2n)$. Ainsi, pour n grand, environ 95% des valeurs de X se situent dans l'intervalle $\left[n - 2\sqrt{2n}, n + 2\sqrt{2n}\right]$.

```
# Simulation d'une loi de chi2 de degré 50
import scipy.stats as st
x_min, x_max = 0,100
pas = np.linspace(x_min,x_max,1000)
normal = st.norm.pdf(pas,loc=50,scale=np.sqrt(100))
chi2 = st.chi2.pdf(pas,50)
print((pd.DataFrame({"x":pas,"normal":normal,"chi2":chi2}) >>
       ggplot(aes(x="x",y="normal"))+geom_line(color="red")+
       geom_line(aes(y="chi2"),color="blue")+ ylab("f(x)")))
```

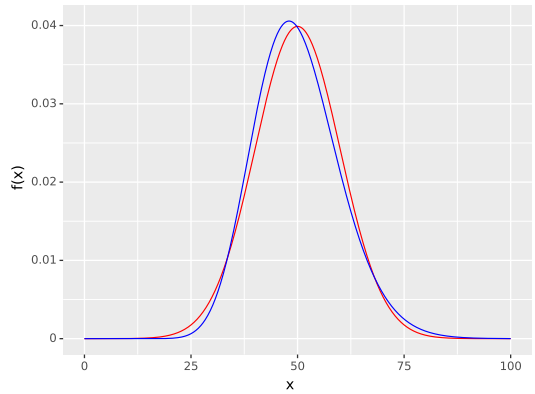


Figure 1.4 – Densité d'un χ^2_{50} (en bleu) et densité d'une $\mathcal{N}(50, 100)$ (en rouge)

1.2.2.2 Loi de Student

Définition 1.2 *Soit X une variable aléatoire suivant une loi normale centrée réduite et Y une variable aléatoire suivant une loi de χ^2 à n degrés de liberté, avec X et Y indépendantes. La loi de la variable*

$$T = \frac{X}{\sqrt{Y/n}} \tag{1.37}$$

est appelée loi de Student à n degrés de liberté et on note $T \sim \mathcal{T}_n$.

Lorsque $n = 1$, T suit une loi de Cauchy et n'a donc pas d'espérance (ni, a fortiori, de variance). Pour $n = 2$, T est centrée mais de variance infinie. Pour $n \geq 3$, T est centrée et de variance $\dfrac{n}{n-2}$.

```
# Simulation d'une loi de Student
x_min, x_max = -10,10
pas = np.linspace(x_min,x_max,1000)
normal = st.norm.pdf(pas)
student = st.t.pdf(pas,3)
print((pd.DataFrame({"x":pas,"normal":normal,"student":student}) >>
       ggplot(aes(x="x",y="normal"))+geom_line(color="red")+
       geom_line(aes(y="student"),color="blue")+ylab("f(x)")))
```

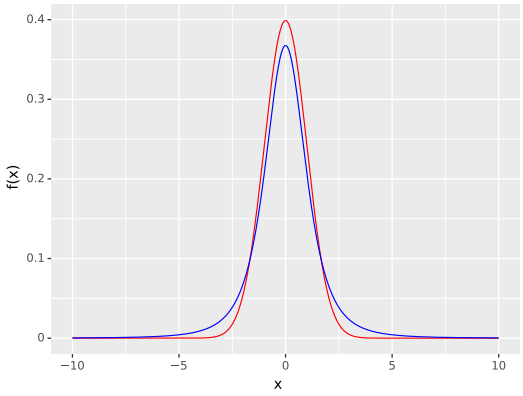


Figure 1.5 – Densité d'un $\mathcal{T}_3$ (en bleu) et densité d'une $\mathcal{N}(0,1)$ (en rouge)

D'autre part, lorsque n devient grand, on sait par la Loi des Grand Nombres que le dénominateur tend presque sûrement vers 1. De fait, on peut montrer que pour n grand, T tend en loi vers une gaussienne centrée réduite : $T \approx \mathcal{N}(0,1)$. Par conséquent, lorsque n sera grand, on pourra remplacer les quantiles d'une loi de Student $\mathcal{T}_n$ par ceux d'une loi $\mathcal{N}(0,1)$.

```
# Simulation d'une loi de Student
student2 = st.t.pdf(pas,50)
print((pd.DataFrame({"x":pas,"normal":normal,"student":student2}) >>
       ggplot(aes(x="x",y="normal"))+geom_line(color="red")+
       geom_line(aes(y="student"),color="blue")+ylab("f(x)")))
```

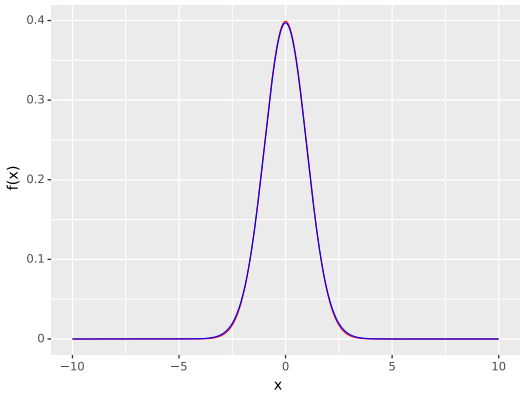


Figure 1.6 – Densité d'un $\mathcal{T}_{50}$ (en bleu) et densité d'une $\mathcal{N}(0,1)$ (en rouge)

1.2.2.3 Loi de Fisher

Définition 1.3 *Soit X une variable aléatoire suivant une loi du χ^2 à n degrés de liberté et Y une variable aléatoire suivant une loi de χ^2 à m degrés de liberté, avec X et Y indépendantes. La loi de la variable*

$$F = \frac{X/n}{Y/m} \tag{1.38}$$

est appelée loi de Fisher à (n,m) degrés de liberté et on note $F \sim \mathcal{F}_{n,m}$.

Pour $m > 2$, la variance d'une loi de Fisher $\mathcal{F}_{n,m}$ est $\frac{m}{m-2}$. Dans la suite, typiquement, m sera grand, de sorte qu'à nouveau lalOi des Grands Nombres implique que Y/m tend vers 1. Dans ce cas, F peut se voir comme un χ^2 normalisé par son degré de liberté : $F \approx \chi^2_n/n$. Ceci est illustrée (Figure 1.7) pour $n = 1$ et $m = 10$.

```
n, m = 2,10
pas = np.linspace(0,7,100)
chi2_normalized = st.chi2.pdf(pas,n)/n
fisher = st.f.pdf(pas,n,m)
print((pd.DataFrame({"x":pas,"fisher":fisher,"chi2_normalized":chi2_normalized}) >>
       ggplot(aes(x="x",y="fisher"))+geom_line(color="red")+
       geom_line(aes(y="chi2_normalized"),color="blue")+ylab("f(x)")))
```

1.2.3 Loi des estimateurs

Nous allons maintenant voir comment les lois précédentes interviennent dans nos estimateurs. Afin de faciliter la lecture de cette partie, considérons les notations suivantes :

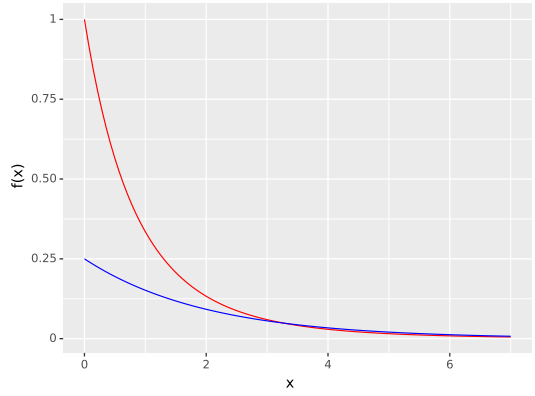


Figure 1.7 – Densité d'une $\mathcal{F}_{2,10}$ (en rouge) et densité d'un $\chi^2_2/2$ (en bleu)

$$\begin{aligned}
\sigma^2_{\widehat{\beta}_0} &= \sigma^2_\varepsilon \left(\frac{1}{n} + \frac{\overline{x}^2}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \right), & \hat{\sigma}^2_{\widehat{\beta}_0} &= \hat{\sigma}^2_\varepsilon \left(\frac{1}{n} + \frac{\overline{x}^2}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \right) \\
\sigma^2_{\widehat{\beta}_1} &= \frac{\sigma^2_\varepsilon}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2}, & \hat{\sigma}^2_{\widehat{\beta}_1} &= \frac{\hat{\sigma}^2_\varepsilon}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2}
\end{aligned}$$

où $\hat{\sigma}^2_\varepsilon = (n-2)^{-1} \sum_{t=1}^{t=n} \hat{\varepsilon}_t$. Notons que les estimateurs de la colonne de gauche ne sont pas réellement des estimateurs.En effet puisque σ^2_ε est inconnu, c'est estimateurs ne sont pas calculables avec les données. Cependant ec sont eux qui interviennent dans les lois des estimateurs $\widehat{\beta}_0$ et $\widehat{\beta}_1$. Les estimateurs donnés dans la colonne de droite sont ceuw qui sont utilisés (et utilisables) et ils consistent simplement à remplacer σ^2_ε par $\hat{\sigma}^2_\varepsilon$.

Les lois des estimateurs sont données dans la proposition suivante.

Proposition 1.10 (Lois des estimateurs : variance connue) *Les lois des estimateurs des moindres carrés ordinaires sont :*

1. $\hat{\beta}_0 \sim \mathcal{N}\left(\beta_0, \sigma^2_{\widehat{\beta}_0}\right)$.
2. $\hat{\beta}_1 \sim \mathcal{N}\left(\beta_1, \sigma^2_{\widehat{\beta}_1}\right)$.
3. $\hat{\beta} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix} \sim \mathcal{N}\left(\beta, \Omega_\beta\right)$ *avec* $\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}$ *et* $\Omega_\beta = \dfrac{\sigma_\varepsilon}{\sum_{t=1}^{t=n} (x_t - \overline{x})^2} \begin{pmatrix} n^{-1} \sum_{t=1}^{t=n} x_t & -\overline{x} \\ -\overline{x} & 1 \end{pmatrix}$
4. $\dfrac{n-2}{\sigma^2_\varepsilon} \hat{\sigma}^2_\varepsilon$ *suit une loi du* χ^2 *à* $(n-2)$ *degrés de liberté.*
5. $\left(\hat{\beta}_0, \hat{\beta}_1\right)$ *et* $\hat{\sigma}^2_\varepsilon$ *sont indépendants.*

Le problème des propriétés ci - dessus vient de ce qu'elles font intervenir la variance théorique σ_ε^2, généralement inconnue. La façon naturelle de procéder est de la remplacer par son estimateur $\hat{\sigma}_\varepsilon^2$. Les lois intervenant dans les estimateurs s'en trouvent de fait légèrement modifiées.

Proposition 1.11 (Lois des estimateurs : variance estimée) *Lorsque σ_ε^2 est estimée par $\hat{\sigma}_\varepsilon^2$,nous avons :*

1. $\dfrac{\hat{\beta}_0 - \beta_0}{\hat{\sigma}_{\widehat{\beta}_0}} \sim \mathcal{T}_{n-2}$ *où* $\mathcal{T}_{n-2}$ *est une loi de Student à* $(n-2)$ *degrés de liberté.*
2. $\dfrac{\hat{\beta}_0 - \beta_0}{\hat{\sigma}_{\widehat{\beta}_0}} \sim \mathcal{T}_{n-2}$.
3. $\dfrac{1}{2\hat{\sigma}_\varepsilon^2}\left(\hat{\beta}-\beta\right)' V^{-1}\left(\hat{\beta}-\beta\right) \sim \mathcal{F}_{2,n-2}$ *où* $\mathcal{F}_{2,n-2}$ *est une loi de Fisher à* 2 *et* $n-2$ *degrés de liberté et* $V = \dfrac{1}{\sigma_\varepsilon^2}\Omega_\beta$.

1.2.4 Intervalle et région de confiance

Ces dernières propriétés nous permettent de donner des intervalles de confiance (IC) ou des régions de confiance (RC) des estimateurs. En effet, la valeur ponctuelle d'un estimateur est de peu d'intérêt en général et il est intéressant de lui associer un intervalle de confiance. Les résultats sont donnés pour un α en général, en pratique on prend typiquement $\alpha = 0.05$.

1.2.4.1 Intervalle de confiance pour un coefficient

Un intervalle de confiance du coefficient β_i au risque α est donné par :

$$IC_{1-\alpha}(\beta_i) = \left[\hat{\beta}_i - \hat{\sigma}_{\widehat{\beta}_i} t_{n-2}^{1-\alpha/2}; \hat{\beta}_i + \hat{\sigma}_{\widehat{\beta}_i} t_{n-2}^{1-\alpha/2}\right] \tag{1.39}$$

$t_{n-2}^{1-\alpha/2}$ représente le quantile de niveau $(1-\alpha/2)$ d'une loi de Student $\mathcal{T}_{n-2}$.

Pour les données « Concentration en Ozone \fg[], si nous intéressons au rôle des variables, nous pouvons calculer les intervalles de confiance des paramètres.

```
# Intervalle de confiance des paramètres
confint = model.conf_int(alpha=0.05)
confint.columns = ["[0.025","0.975]"]
```

Table 1.8 – Intervalle de confiance des paramètres

	[0.025	0.975]
Intercept	5.159232	57.67071
T12	1.441180	3.96089

La première ligne correspond à l'intervalle de confiance de β_0 et la seconde correspond à l'intervalle de confiance de β_1. L'intervalle de confiance à 95% sur l'ordonnée à l'origine est étendu (52.5). Cela provient des erreurs (l'estimation de σ est de 20.5), mais

surtout du fait que les températures sont en moyenne très loin de 0. Cependant ce coefficient en fait pas très souvent l'objet d'interprétation. L'autre IC à 95% est moins étendu (2.5). Nous constatons qu'il semble exister un effet de la température sur les pics d'ozone, bien que l'on pose la question de la validité de l'hypothèse linéaire.

1.2.4.2 Région de confiance simultanée des paramètres

Une région de confiance simultanée des deux paramètres inconnus $\beta = (\beta_0, \beta_1)'$ est donnée par l'équation suivante :

$$\frac{1}{2\hat{\sigma}^2}\left[n(\hat{\beta}_0-\beta_0)^2+2n\overline{x}(\hat{\beta}_0-\beta_0)(\hat{\beta}_1-\beta_1)+\sum_{t=1}^{t=n}x_t^2(\hat{\beta}_1-\beta_1)\right]\leq f_{2,n-2}^{1-\alpha} \tag{1.40}$$

où $f_{2,n-2}^{1-\alpha}$ représente le quantile de niveau $(1-\alpha)$ d'une loi de Fisher à 2 et $n-2$ degrés de liberté.

Exemple 1.2 *Concentration en Ozone - Région de confiance*

Il est possible de tracer la région de confiance simultanée des deux paramètres β_0 et β_1, ce qui est rarement fait en pratique. Nous pouvons la comparer aux intervalles de confiance au mêm degré de confiance. Cette comparaison illustre uniquement la différence entre intervalle simple et région de confiance. En général, l'utilisateur de la méthode choisit l'ne ou l'autre forme. Pour cette comparaison, nous utilisons les commandes suivantes :

```
# Simulation d'une loi normale multivariée
sim_norm = st.multivariate_normal.rvs(model.params,model.cov_params(),100,
                                      np.random.seed(123))
x_min, x_max = confint.iloc[:,0],confint.iloc[:,1]
xx_min,xx_max = np.min(sim_norm,axis=0),np.max(sim_norm,axis=0)
# Région de confiance
print((pd.DataFrame(sim_norm,columns=["Intercept","T12"]) >>
     ggplot(aes(x="Intercept",y="T12"))+
     geom_rect(aes(xmin=x_min[0],xmax=x_max[0],ymin=x_min[1],ymax=x_max[1]),
               alpha=0.0,color="black",linetype ='dashed')+
     geom_rect(aes(xmin=xx_min[0],xmax=xx_max[0],ymin=xx_min[1],ymax=xx_max[1]),
               alpha=0.0,color="black",linetype ='-')+
     geom_point(aes(x = model.params[0],y=model.params[1]),shape="+")+
     stat_ellipse()+labs(x="$\\beta_{0}$",y="$\\beta_{1}$")))
```

Les axes de l'ellipse ne sont pas parallèles aux axes du graphique, les deux estimateurs sont corrélés. Nous retrouvons que la corrélation entre les deux estimateurs est toujours négative (ou nulle), le plus axe de l'ellipse ayant une pente négative. Nous observons bien sûr une différence entre le rectangle de confiance, juxtaposition des deux intervalles de confiance et l'ellipse.

1.2.4.3 Intervalle de confiance pour la variance

Un intervalle de confiance de σ^2 est donné par :

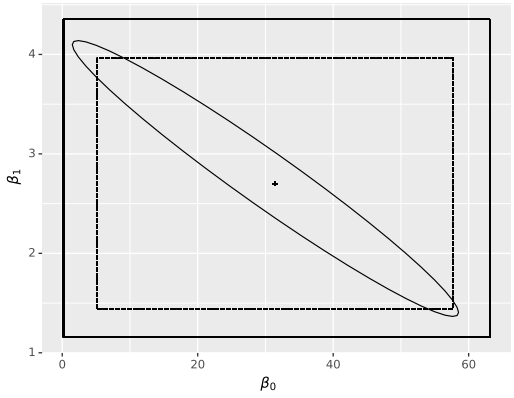


Figure 1.8 – Région de confiance simultanée des deux paramètres

$$IC(\sigma_\varepsilon^2) = \left[\frac{(n-2)}{\chi_1^2}\hat{\sigma}_\varepsilon^2; \frac{(n-2)}{\chi_2^2}\hat{\sigma}_\varepsilon^2\right] \tag{1.41}$$

avec χ_1^2 ayant la probabilité $\alpha/2$ d'être dépassée et χ_2^2 la probabilité $1-\alpha/2$ d'être dépassée à $n-p-1$ degrés de liberté.

1.2.4.4 Intervalle de confiance pour la droite de régression

Nous pouvons également donner un intervalle de confiance de la droite de régression.

Proposition 1.12 (IC pour $\mathbb{E}(y_t)$) *Un intervalle de confiance $\mathbb{E}(y_t) = \beta_0 + \beta_1 x_t$ est donné par*

$$\left[\hat{y}_{t'} \pm t_{n-2}^{1-\alpha/2}\hat{\sigma}_\varepsilon \sqrt{\frac{1}{n} + \frac{(x_{t'} - \overline{x})^2}{\sum_{t=1}^{t=n}(x_t - \overline{x})^2}}\right] \tag{1.42}$$

En calculant les intervalles de confiance pour tous les points de la droite, nous obtenons une hyperbole de confiance. En effet, lorsque $x_{t'}$ est proche de $\overline{x}$, le terme dominant de la variance est $1/n$, mais dès que $x_{t'}$ s'éloigne de $\overline{x}$, le terme dominant est le terme au carré.

1.2.4.5 Intervalle de confiance pour la prévision

En matière de prévision dans le cas d'erreurs gaussiennes, les résultats obtenus en 1.1.4 pour l'espérance et la variance sont toujours valables. De plus, puisque $\hat{y}_{n+1}$ est linéaire en $\hat{\beta}_0$ et $\hat{\beta}_1$ et ε_{n+1}, on peut préciser sa loi :

$$y_{n+1} - \hat{y}_{n+1} \sim \mathcal{N}\left(0, \sigma_\varepsilon^2\left(1 + \frac{1}{n} + \frac{(x_{n+1} - \overline{x})^2}{\sum_{t=1}^{t=n}(x_t - \overline{x})^2}\right)\right) \tag{1.43}$$

A nouveau on ne connaît pas σ_ε^2 et on l'estime donc par $\hat{\sigma}_\varepsilon^2$. Comme $(y_{n+1} - \hat{y}_{n+1})$ et $\frac{(n-2)}{\hat{\sigma}_\varepsilon^2}\sigma_\varepsilon^2$ sont indépendants, on peut énoncer un résultat donnant des intervalles de confiance pour y_{n+1}.

Proposition 1.13 (Loi et intervalle de confiance pour la prédiction) *Avec les hypothèses précédentes, on a :*

$$\frac{y_{n+1} - \hat{y}_{n+1}}{\hat{\sigma}_\varepsilon \sqrt{1 + \frac{1}{n} + \frac{(x_{n+1} - \overline{x})^2}{\sum_{t=1}^{t=n}(x_t - \overline{x})^2}}} \sim \mathcal{T}_{n-2} \tag{1.44}$$

d'où l'on déduit l'intervalle de confiance suivant pour y_{n+1} :

$$\left[\hat{y}_{n+1} \pm t_{n-2}^{1-\alpha/2}\hat{\sigma}_\varepsilon \sqrt{1 + \frac{1}{n} + \frac{(x_{n+1} - \overline{x})^2}{\sum_{t=1}^{t=n}(x_t - \overline{x})^2}}\right] \tag{1.45}$$

Plus le point à prévoir admet pour abscisse x_{n+1} une valeur éloignée de $\overline{x}$, plus l'intervalle de coniance sera grand.

Plus précisément, la courbe décrite par les limites de ces intervalles de confiance lorsque x_{n+1} varie est une hyperbole d'axes $x = \overline{x}$ et $y = \hat{\beta}_0 + \hat{\beta}_1 x$. Pour s'en persuader, il suffit d'effectuer le changement de variables

$$\begin{cases} X &= x - \overline{x} \\ Y &= y - \left(\hat{\beta}_0 + \hat{\beta}_1 x\right) \end{cases}$$

d'où il ressort qu'un point (X, Y) est dans la région de confiance ci - dessus si et seulement si

$$\frac{X^2}{a^2} - \frac{Y^2}{b^2} \leq 1$$

avec

$$\begin{cases} a &= \left(1 + \frac{1}{n}\right)\left[\hat{\sigma} t_{n-2}^{1-\alpha/2}\right]^2 \\ b &= \left(1 + \frac{1}{n}\right)\sum_{t=1}^{t=n}(x_t - \overline{x})^2 \end{cases}$$

Ce qui définit bien l'intérieur d'une hyperbole. En particulier, le centre de gravité hyperbole est tout bonnement le centre de gravité du nuage de points.

Pour les données « Concentration en Ozone », dans une optique de prévision, il est nécessaire de s'intérer à la qualité de la prévision.Cette qualité peut être envisagée de manière succincte grâceà l'intervalle de confiance des prévisions. Afin de bien les distinguer de celui dela droite, nous figurons les deux sur un même graphique.

```
# Calcul des intervalles de confiance
frame= forecast.summary_frame(alpha=0.05)
```

Table 1.9 – Prévision et intervalle de confiance

mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
67.609	5.216	57.121	78.097	25.070	110.148
71.930	4.418	63.047	80.814	29.759	114.102
52.753	8.305	36.055	69.451	8.274	97.232
66.799	5.374	55.994	77.603	24.181	109.416
69.500	4.858	59.732	79.267	27.133	111.866
76.522	3.681	69.120	83.924	34.637	118.407

```
# Concaténation
DPrim = (donnee >> select("O3", "T12")).reset_index(drop=True)
DPrim = pd.concat([DPrim,frame],axis=1)
# Représentation graphique
print((DPrim >>ggplot(aes(x="T12",y="O3"))+geom_point(color="black")+
     geom_line(aes(y="mean"),color="black")+
     geom_line(aes(y="mean_ci_lower"),color="blue",linetype = "dashed")+
     geom_line(aes(y="mean_ci_upper"),color="blue",linetype = "dashed")+
     geom_line(aes(y="obs_ci_lower"),color="red",linetype = "dotted")+
     geom_line(aes(y="obs_ci_upper"),color="red",linetype = "dotted")+
     labs(x="T12",y="O3")))
```

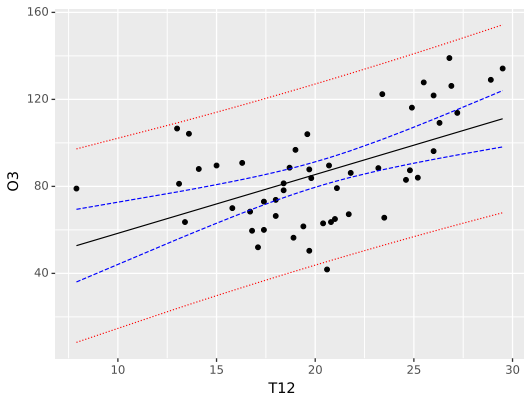


Figure 1.9 – Droite de régression et intervalle de confiance pour y (en rouge) et pour $\mathbb{E}(y)$ (en bleu)

Afin d'illustrer les équations des intervalles de confiance pour les prévisions et la droite ajustée, nous remarquons bien évidemment que l'intervalle de confiance des prévisions est plus grand que l'intervalle de confiance de la droite de régression. L'intervalle de confiance de la droite de régression admet une forme hyperbolique.

1.2.5 Tests de significativité

Cette construction est facilitée par l'hypothèse de normalité des erreurs. En effet, on admet que $\varepsilon_t \sim \mathcal{N}(0, \sigma_\varepsilon^2)$. Cette hypothèse n'est indispensable pour obtenir des estimateurs convergents des paramètres du modèle. Mais elle est utilisée pour la construction des tests statistiques et intervalle de confiance. Les trois tests suivantes sont équivalents et portent sur la significativité du modèle linéaire simple :

	Hypothèses	
Test 1	$H_0 : \beta_1 = 0$	$H_1 : \beta_1 \neq 0$
Test 2	$H_0 : \rho_{x,y} = 0$	$H_1 : \rho_{x,y} \neq 0$
Test 3	$H_0 : SCE = 0$	$H_1 = SCE \neq 0$

Le premier test porte sur la pente de la droite de régression, le deuxième test sur le coefficient de corrélation entre x et y et, enfin, le troisième a pour but de juger si la somme des carrés expliqués est significative, ces trois tests néanmoins répondent à la même interrogation.

1.2.5.1 Test sur la pente de la droite de régression

Il porte sur le coefficient β_1 (test de significativité de β_1). Pour effectuer ce test, on admet que :

$$\frac{\hat{\beta}_1 - \overline{a}}{\sigma_{\widehat{\beta}_1}} \sim \mathcal{N}(0,1) \tag{1.46}$$

Cela est valable aussi bien pour β_0 que pour β_1. $\overline{a}$ représente un nombre donné. Dans le test de significativité d'un coefficient β_1, $\overline{a} = 0$. On a donc que :

$$\frac{\hat{\beta}_1}{\sigma_{\widehat{\beta}_1}} \sim \mathcal{N}(0,1) \tag{1.47}$$

$\hat{\sigma}_{\widehat{\beta}_1}$ est l'estimateur de $\sigma_{\widehat{\beta}_1}$. D'où :

$$t_{cal} = \frac{\hat{\beta}_1}{\hat{\sigma}_{\widehat{\beta}_1}} \sim \mathcal{T}_{n-2} \tag{1.48}$$

On rejette l'hypothèse nulle (H_0) au risque α si $|t_{n-2}^{1-\alpha/2}| > t_{cal}$. Rejeter H_0 signifie que le coefficient β_1 est significativement différent de zéro et dans le cas d'un modèle linéaire simple, cela signifie que le modèle est significatif.

```
#Test de significativité des paramètres
ttest = pd.concat([model.tvalues,model.pvalues],axis=1)
ttest.columns = ['t','P>|t|']
```

P>|t| est appelée probabilité critique ou « p-value ». Elle représente la probabilité, pour la statistique de test sous H_0, de dépasser la valeur estimée. Ici, nous rejetons l'hypothèse H_0 pour les deux paramètres estimés au niveau $\alpha = 5\%$.

Dans le cadre de la régression simple, cela permet d'effectuer de manière rapide un choix de variable pertinente. En toute rigueur, si pour les deux paramètres l'hypothèse

Table 1.10 – Test T-student de significativité des paramètres

	t	P>|t|
Intercept	2.4057	2e-02
T12	4.3107	1e-04

H_0 est acceptée, il est nécessaire de reprendre un modèle en supprimant le paramètre dont la probabilité critique est la plus proche de 1. Dans ce cas - là, dès la phase de représentation des données, de grosses doutes doivent apparaître sur l'intérêt de la régression linéaire simple.

1.2.5.2 Test du coefficient de corrélation linéaire

C'est le test de significativité du coefficient de corrélation linéaire $\rho_{x,y}$. Pour l'effectuer, on :

— Calcule d'abord le coefficient de corrélation empirique : $\rho_{xy} = \dfrac{\text{Cov}(x,y)}{\sigma_x \sigma_y}$

— Calcule la statistique du test : $t_{cal} = \dfrac{\rho_{x,y}}{\sqrt{\dfrac{1-\rho_{x,y}^2}{n-2}}}$

On rejette l'hypothèse nulle H_0 au risque α si $|t_{n-2}^{1-\alpha/2}| > t_{cal}$.

```
# Test du coefficent de corrélation de Pearson
from scipy.stats.stats import pearsonr
def corr_test(x,y,**kwargs):
  res = pearsonr(x,y,**kwargs)
  df = pd.DataFrame({"statistic" : res[0], "pvalue":res[1]},
                    index=["pearson test"])
  return df
# Application
corrtest = corr_test(donnee['T12'],donnee['O3'])
```

Table 1.11 – Test du coefficent de corrélation de Pearson

	statistic	pvalue
pearson test	0.5282814	8.04e-05

1.2.5.3 Test de significativité global

C'est le test de significativité du modèle encore appelé test par analyse de la variance. Pour effectuer ce test, la statistique calculée est celle de Fisher :

$$F_{cal} = \frac{\dfrac{SCE}{ddl_{SCE}}}{\dfrac{SCR}{ddl_{SCR}}} = \frac{\dfrac{SCE}{1}}{\dfrac{SCR}{n-2}} = \frac{\dfrac{R^2}{1}}{\dfrac{1-R^2}{n-2}} \tag{1.49}$$

La statistique F_{cal} est le rapport de la somme des carrés expliquée par x_t sur la somme des carrés des résidus, chacune de ces sommes étant divisée par son degré de liberté respectif. Ainsi, si la variance expliquée est significativement supérieure à la variance résiduelle, la variable x_t est considérée comme étant une variable réellement explicative.

F_{cal} suit une statistique de Fisher à 1 et $n-2$ degrés de liberté. Nous rejettons l'hypothèse nulle H_0 au seuil α si F_{cal} est supérieur à la valeur lue sur la table de la loi de Fischer à 1 et $n-2$ degrés de liberté. Dans le cas contraire, nous acceptons l'hypothèse d'égalité des variances, la variable x_t n'est pas explicative de la variable x_t.

```
# Test de Fisher
import numpy as np
ftest = pd.DataFrame(np.transpose([model.df_model,model.df_resid,
                                   model.fvalue,model.f_pvalue])).T
ftest.columns = ['d1','d2','f','pvalue']
ftest.index = ['f-test']
```

Table 1.12 – Test F de Fisher

	d1	d2	f	pvalue
f-test	1	48	18.58171	8.04e-05

Remarque 1.7

Avec les 3 tests ci-dessus, le rejet de H_0 signifie que le modèle est significatif. On peut aussi montrer que $F_{cal} = (t_{cal})^2$ dans le cas d'un modèle linéaire de régression simple.

Preuve

$$F_{cal} = (t_{cal})^2 = \left(\frac{\hat{\beta}_1}{\hat{\sigma}_{\hat{\beta}_1}}\right)^2 = \frac{\hat{\beta}_1^2}{\hat{\sigma}_\varepsilon^2 / \sum_{t=1}^{t=n}(x_t - \overline{x})^2} = \frac{\hat{\beta}_1^2 \sum_{t=1}^{t=n}(x_t - \overline{x})^2}{\sum_{t=1}^{t=n}\hat{\varepsilon}_t/(n-2)}$$

1.2.5.4 Test portant sur un paramètre

On rappelle que le modèle considéré est défini par l'équation (1.1). Nous voulons tester au risque α l'hypothèse nulle $H_0 : \beta_i = \overline{a}$ (où $\overline{a}$ est une valeur connue et $i = 0, 1$) contre une hypothèse alternative. Pour effectuer ce test, la statistique à calculer est :

$$t_{cal} = \frac{\hat{\beta}_i - \overline{a}}{\hat{\sigma}_{\hat{\beta}_i}} \sim \mathcal{T}_{n-2} \tag{1.50}$$

Le rejet de décision est indiqué dans le tableau :

Table 1.13 – Règle de décision

Hypothèse		Rejeter H_0 au risque α
$H_0 : \beta_i = \overline{a}$	$H_1 : \beta_i \neq \overline{a}$	$\lvert t_{n-2}^{1-\alpha/2} \rvert > t_{cal}$
$H_0 : \beta_i = \overline{a}$	$H_1 : \beta_i > \overline{a}$	$t_{cal} > t_{n-2}^{1-\alpha}$
$H_0 : \beta_i = \overline{a}$	$H_1 : \beta_i < \overline{a}$	$t_{cal} < -t_{n-2}^{1-\alpha}$

où $t_{n-2}^{1-\alpha/2}$ et $t_{n-2}^{1-\alpha}$ sont les quantiles d'ordre respectifs $1-\alpha/2$ et $1-\alpha$ de la loi de Student à $n-2$ degrés de libertés (n est le nombre d'observation).

2 Modèle linéaire de régression multiple

Sommaire

Le modèle linéaire de régression multiple est une extension du modèle linéaire simple (*cf.* Bourbonnais et al. (2021)). Ici, plusieurs variables exogènes expliquent le comportement de la variable endogène.

2.1 Spécification du modèle

2.1.1 Présentation du modèle

Spécifié en séries temporelles, on a le modèle suivant :

$$y_t = \beta_0 + \beta_1 x_{1t} + \cdots + \beta_p x_{pt} + \varepsilon_t \quad \forall t = 1, \dots, n \tag{2.1}$$

où y_t est la variable à expliquer au temps t et les $x_{jt} (j = 1, \dots, p)$ les p variables explicatives au temps t. $\beta_0, \dots, \beta_p$ les $(p+1)$ paramètres du modèle. Comme au chapitre précédent, ε_t représente l'erreur de spécification, c'est-à-dire la différence entre le modèle vrai et le modèle spécifié. n est le nombre d'observation.

Si on était dans un univers déterministe, la loi en question serait parfaitement vérifiée, mais ce n'est pas le cas pour plusieurs raisons :

— Les phénomènes économiques sont caractérisés par l'interdépendance entre nombreux éléments, ce qui entraîne que les variables explicatives susceptibles d'exercer une influence sur le variable expliquée sont nombreuses. Mais l'effet des variables qui ont été omises explique qu'il y ait des écarts entre la réalité observée et le résultat de la fonction ;
— Les phénomènes économiques sont mis en œuvre par des individus qui n'ont eux-mêmes pas un comportement déterminé. C'est donc leur libre arbitre qui fait que parfois ils n'agissent pas comme prévue et on n'obtient pas les résultats escomptés.

Les lois économiques sont donc seulement vraies en moyenne. Elles ont le caractère de lois statistiques. On choisit de traduire les écarts entre la réalité observée et les résultats de l'estimation par une variable aléatoire notée ε_t encore appelée terme d'erreur.

Compte tenu de l'existence de ce terme, la loi qui va gouverner la variable expliquée (y_t) n'est pas une loi mathématique mais une loi statistique et il faudra se donner une distribution pour caractériser ε_t.

L'estimation des coefficients $\beta_0, \dots, \beta_p$ fournit des valeurs qui seront elles-mêmes des variables aléatoires ayant une distribution de probabilité que l'on devra spécifier selon la distribution de ε_t.

On dispose d'un ensemble d'observation (données) portant à la fois sur la variable expliquée et sur les variables explicatives. Sachant que la relation a un caractère probabiliste, il n'est pas possible de déterminer de façon unique les vraies valeurs des paramètres.

Les paramètres estimés sont désignés par $\hat{\beta}_0, \dots, \hat{\beta}_p$ et les connaissant on pourra remonter aux valeurs vraies des paramètres.

Pour que cette opération d'inférence statistique soit possible, il faut qu'un certains nombres d'hypothèses soit vérifié, sinon, on aura toujours un résultat numérique sans valeur car il ne permet pas de porter un jugement sur les vraies valeurs des coefficients recherchés.

2.1.2 Le modèle sous forme matricielle

L'écriture précédente du modèle est d'un maniement peu pratique. Afin d'en alléger l'écriture et de faciliter l'expression de certains résultats, on a habituellement recours aux notations matricielles. Ainsi, en écrivant le modèle observation par observation, on obtient :

$$\begin{cases} y_1 & = \beta_0 + \beta_1 x_{11} + \cdots + \beta_p x_{p1} + \varepsilon_1 \\ \vdots & \\ y_n & = \beta_0 + \beta_1 x_{1n} + \cdots + \beta_p x_{pn} + \varepsilon_n \end{cases} \tag{2.2}$$

Soit, sous forme matricielle, ce modèle s'écrit :

$$\underset{(n,1)}{Y} = \underset{(n,p+1)}{X} \underset{(p+1,1)}{\beta} + \underset{(n,1)}{\varepsilon} \tag{2.3}$$

avec

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad X = \begin{pmatrix} 1 & \cdots & x_{p1} \\ \vdots & & \vdots \\ 1 & \vdots & x_{pn} \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

Nous remarquons la première colonne de la matrice X, composée de 1, qui correspond au coefficient β_0 (coeffcient du terme constant).

La dimension de la matrice X est donc de n lignes et $(p+1)$ colonnes (p étant le nombre de variables explicatives réelles, c'est - à - dire constante exclue).

L'écriture sous forme matricielle rend plus aisée la manipulation du modèle linéaire multiple.

2.1.3 Hypothèses de base

Les hypothèses en question portent sur les variables ainsi que sur le terme d'erreur (ε_t).

2.1.3.1 Les hypothèses classiques (hypothèses stochastiques)

Hypothèse 2.1 *Les variables sont observées sans erreur*

Cela suppose que la collecte des informations statistiques a été faite sans erreur. On peut cependant avoir une petite idée sur la fiabilité des chiffres utilise car le degré de précision de la méthode d'estimation est dans une certaine mesure limité par la qualité des données. Il ne sert à rien d'utiliser des méthodes sophistiquées sur des données douteuses.

Hypothèse 2.2 *Le terme d'erreur a une espérance mathématique nulle :* $\mathbf{E}(\varepsilon) = 0$

C'est-à-dire qu'on se trompe tantôt dans un sens et tantôt dans l'autre sens, l'erreur n'est donc pas systématique.

Hypothèse 2.3 *Le terme d'erreur a une variance qui est constante :* $V(\varepsilon) = \sigma_\varepsilon^2 I_n$

C'est ce qu'on appelle l'homoscédasticité c'est-à-dire la marge d'erreur est supposée identique en tout point de l'échantillon. C'est une hypothèse parfois contestable dans la mesure où on a des raisons de penser que la variance peut augmenter ou diminuer sur l'ensemble des observations. Cette hypothèse peut être testée et quand on a repéré l'hétéroscédasticité, on peut la corriger et obtenir néanmoins des bons estimateurs.

Hypothèse 2.4 *Les termes d'erreur sont indépendants les uns des autres dans le temps :* $Cov(\varepsilon_t, \varepsilon_s) = 0, \quad \forall t \neq s$

Cette hypothèse traduit le fait que l'erreur n'est pas systématique mais aussi le fait que la dispersion est constante sur l'ensemble de l'échantillon (non corrélation des erreurs). Cette hypothèse peut être testée et lorsqu'elle n'est vérifiée, on parle d'auto corrélation des erreurs. Cette situation est due très souvent à la mauvaise spécification du modèle. En effet, une spécification défectueuse est de nature à entraîner les erreurs systématiques. La mauvaise spécification peut être due à l'omission d'une variable importante dans la modélisation, aux choix erronés du type de fonction qui relie la variable expliquée aux variables explicatives.

Hypothèse 2.5 *Les variables explicatives* $x_{1t}, \ldots, x_{pt}$ *sont indépendantes des termes de l'erreur :* $Cov(X, \varepsilon) = 0$

Dans le cas contraire, on va dire que les variables explicatives sont entachées des erreurs et on utilise la méthode des variables instrumentales (MVI) pour corriger ce problème.

2.1.3.2 Les hypothèses structurelles

Hypothèse 2.6 *Les variables explicatives $x_{1t}, \dots, x_{pt}$ ne sont pas linéairement dépendantes (absence de colinéarité des variables explicatives)*

Ceci implique que la matrice $X^{'}X$ est régulière et que la matrice inverse $(X^{'}X)^{-1}$ existe. Si deux ou plusieurs variables explicatives sont fortement liées, on parle de colinéarité des variables explicatives.

Hypothèse 2.7 $(X^{'}X)\ /\ n$ *tend vers une matrice finie non singulière*

Hypothèse 2.8 n *(le nombre d'observations) est supérieur à* $p+1$ *(le nombre de séries explicatives).*

2.2 Estimation et propriétés des estimateurs

2.2.1 Estimation des coefficients de régression

Le modèle sous forme matricielle à p variables explicatives et n observations s'écrit :

$$Y = X\beta + \varepsilon \tag{2.4}$$

Pour estimer le vecteur β composé des coefficients $\beta_0, \dots, \beta_p$, nous appliquerons la méthode des moindres carrés ordinaires qui consiste à minimiser la somme des carrés des erreurs, soit :

$$\min_{\beta} \varepsilon^{'}\varepsilon = \min_{\beta}(Y - X\beta)^{'}(Y - X\beta) \tag{2.5}$$

Ce programme admet son minimum lorsque la dérivée partielle est nulle. On obtient l'estimateur suivant :

$$\hat{\beta} = (X^{'}X)^{-1}X^{'}Y \tag{2.6}$$

Preuve

$$\begin{aligned} \mathcal{Q}(\beta) &= (Y - X\beta)^{'}(Y - X\beta) = Y^{'}Y - Y^{'}X\beta - \beta^{'}X^{'}Y - \beta^{'}X^{'}X\beta \\ &= Y^{'}Y - 2\beta^{'}X^{'}Y + \beta^{'}X^{'}X\beta \end{aligned}$$

Nous différencions $\mathcal{Q}$ par rapport à β :

$$\frac{\partial}{\partial\beta}\mathcal{Q}(\beta) = -2X^{'}Y + 2X^{'}X\hat{\beta} = 0 \quad \Longrightarrow \quad \hat{\beta} = \left(X^{'}X\right)^{-1}X^{'}Y$$

Pour s'assurer que ce point $\hat{\beta}$ est bien un minimum strict, il faut que la dérivée seconde soit une matrice définie positive. Or la dérivée seconde s'écrit

$$\frac{\partial^2}{\partial \beta^2} \mathcal{Q}(\beta) = 2X^{'}X$$

et X est de plein rang donc $X^{'}X$ est inversible et n'a pas de valeur propre nulle. La matrice $X^{'}X$ est donc définie positive et $\hat{\beta}$ est bien un minimum strict.

Les équations issues de la relation $X^{'}X\hat{\beta} = X^{'}Y$ sont appelées équations normales.

Remarque 2.1

Cette solution est réalisable si la matrice carrée $X^{'}X$ de dimension $(p+1, p+1)$ est inversible. La matrice $X^{'}X$ est la matrice des produits croisés des variables explicatives ; en cas de colinéarité parfaite entre deux variables ecplicatives, la matrice $X^{'}X$ est singulière et la méthode des moindres carrés ordinaires défaillante.

Le modèle estimé s'écrit :

$$y_t = \hat{\beta}_0 + \hat{\beta}_1 x_{1,t} + \cdots + \hat{\beta}_p x_{p,t} + \hat{\varepsilon}_t \tag{2.7}$$

avec $\hat{\varepsilon}_t = y_t - \hat{y}_t$ où $\hat{\varepsilon}_t$ est le résidu, c'est - à - dire l'écart entre la valeur observée de la variable à expliquer et sa valeur estimée (ajustée).

Exemple 2.1

La modélisation de la concentration d'ozone dans l'atmosphère évoquée au chapitre 1 est relativement simpliste. En effet, des variables météorologiques autres que la température peuvent expliquer cette concentration, comme par exemple le rayonnement, la précipitation ou encore le vent qui déplace les masses d'air. Certaines météorologiques susceptibles d'avoir une influence sur la concentration d'ozone sont également mesurées.

Nous cherchons à expliquer l'ozone (`03`) par deux variables explicatives, la température à 12h (`T12`) et le vent (`Vx`). Le vent est mesuré en degré (direction) et mètre par seconde (vitesse).

$$03 = f(T12, Vx) \tag{2.8}$$

Nous chargeons nos données.

```
# Chargement des données
import pandas as pd
donnee = pd.read_csv('./donnee/ozone.txt',sep=';',index_col=0)
```

Le tableau 2.1 donne les 10 premières mesures effectuées.

Faisons une représentation en 3D.

```
# Nuage de points en 3D
from mpl_toolkits import mplot3d
fig = plt.figure(figsize = (10,10))
```

Table 2.1 – 10 données de température à 12 h, teneur en ozone et vent

	O3	T12	Vx
19960422	63.6	13.4	9.35
19960429	89.6	15.0	5.40
19960506	79.0	7.9	19.30
19960514	81.2	13.1	12.60
19960521	88.0	14.1	-20.30
19960528	68.4	16.7	-3.69
19960605	139.0	26.8	8.27
19960612	78.2	18.4	4.93
19960619	113.8	27.2	-4.93
19960627	41.8	20.6	-3.38

```
axe = plt.axes(projection = "3d")
axe.scatter3D(donnee['T12'],donnee['Vx'],donnee['O3'],color="black")
axe.set(xlabel='T12',ylabel='Vx',zlabel='O3');
plt.show()
```

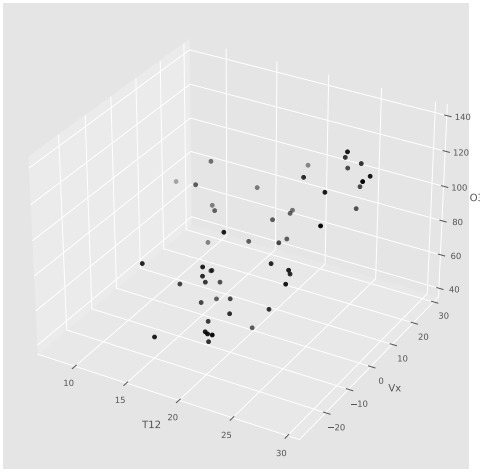


Figure 2.1 – Nuage de points de 03 en fonction de T12 et Vx

Il est très difficile de voir si une régression est adaptée, ce qui signifie ici que les points ne doivent pas être très éloignés d'un plan commun. Le modèle de régression est :

$$\texttt{O3} = \beta_0 + \beta_1 \texttt{T12} + \beta_2 \texttt{Vx} + \varepsilon \tag{2.9}$$

```
# Estimation des paramètres
import statsmodels.formula.api as smf
model = smf.ols(formula = 'O3~T12+Vx',data=donnee).fit()
# Extraction des résultats
model_params = model.summary2().tables[1]
```

On a le modèle ajusté suivant :

$$\widehat{\texttt{O3}} = 35.4530 + 2.5380\texttt{T12} + 0.8736\texttt{Vx} \tag{2.10}$$

Table 2.2 – Coefficients du modèle linéaire : $O3_t = \beta_0 + \beta_1 T12_t + \beta_2 Vx + \varepsilon_t$

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	35.4530	10.7446	3.2996	0.0019	13.8377	57.0683
T12	2.5380	0.5151	4.9270	0.0000	1.5017	3.5744
Vx	0.8736	0.1772	4.9309	0.0000	0.5172	1.2300

A l'issue de cette phase d'estimation, nous pouvons tracer notre modèle, le plan d'équation $z = 35.453 + 2.538x + 0.8736y$.

```
xx, yy = np.meshgrid(donnee["T12"],donnee["Vx"])
z = 35.4530 + 2.5380*xx + 0.8736*yy
# Représentation graphique
fig = plt.figure(figsize = (10,10))
axe = plt.axes(projection = "3d")
axe.scatter3D(donnee['T12'],donnee['Vx'],donnee['O3'],color="black");
axe.plot_surface(xx,yy,z,alpha=0.01);
axe.set(xlabel='T12',ylabel='Vx',zlabel='O3');
plt.show()
```

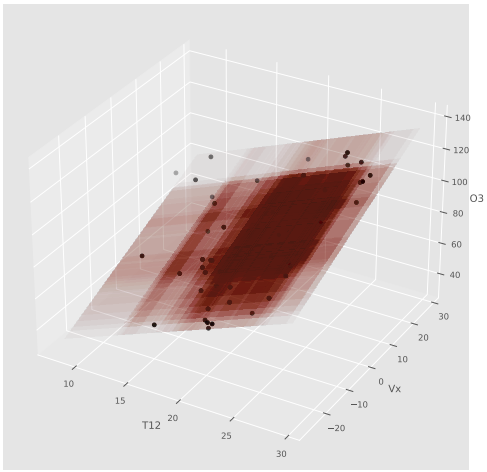


Figure 2.2 – Représentation des données et hyperplan.

2.2.1.1 Effet de la variation d'une seule des variables explicatives

Soit le modèle estimé :

$$y_t = \hat{\beta}_0 + \hat{\beta}_1 x_{1,t} + \cdots + \hat{\beta}_p x_{p,t} + \hat{\varepsilon}_t$$

Si la variable x_j passe de la valeur $x_{j,t}$ à $\left(x_{j,t} + \Delta x_{j,t}\right)$, toutes choses étant égales par ailleurs (les $p-1$ autres variables restant constantes), alors la variable à expliquer varie de $\hat{\beta}_j \Delta x_j$:

$$\Delta\hat{y}_t = \hat{\beta}_j \Delta x_{j,t} \tag{2.11}$$

Les coefficients s'interprètent donc directement en terme de propension marginale.

Exemple 2.2 *Effet de la température sur la concentration en ozone*

Rappelons que dans le cadre du modèle linéaire simpke, le modèle estimé s'ecrit :

$$\widehat{\texttt{O3}} = 31.415 + 2.701\texttt{T12} \tag{2.12}$$

Le coefficient associé à la température (T12) est significatif sur les deux modèles, par conséquent on peut faire une comparaison sur les niveaux de variation.

Table 2.3 – Effet de variation de $T2$

Variable	Modèle simple	Modèle multiple
T12	2.701	2.5380

L'effet est plus grand dans le cas du modèle simple que dans le cas du modèle multiple.

Nous avons ajouté la variable Vx au modèle présenté dans le chapitre 1, cet ajout est - il pertinent ? Afin de répondre correctement à cette question nous devons envisager de construire soit des procédures générales de choix de modèles, soit un test entre le modèle de la régression simple $\texttt{O3} = \beta_0 + \beta_1\texttt{T12} + \varepsilon$ et le modèle plus complexe $\texttt{O3} = \beta_0 + \beta_1\texttt{T12} + \beta_2\texttt{Vx} + \varepsilon$.

2.2.2 Propriétés des estimateurs

Le statisticien cherche vérifier que les estimateurs des moindres carrés ordinaires que nous avons construits admettent de bonnes propriétés au sens statistique. Dans notre cadre de travail, cela peut se résumer en deux parties : l'estimateur des moindres carrés ordinaires est - il sans biais et est - il de variance minimale dans sa classe d'estimateurs ?

2.2.2.1 Estimateurs sans biais

Le modèle sous forme matricielle peut s'écrire, comme pour le modèle de régression simple, de différentes manières :

$$\left.\begin{array}{l} Y = X\beta \\ Y = X\hat{\beta} + \hat{\varepsilon} \\ \widehat{Y} = X\hat{\beta} \end{array}\right\} \rightarrow \quad \hat{\varepsilon} = Y - \widehat{Y}$$

Nous obtenons :

$$\begin{array}{rcl} \hat{\beta} & = & (X'X)^{-1}X'Y = (X'X)^{-1}X'\left(X\beta + \varepsilon\right) \\ & = & (X'X)^{-1}X'(X\beta) + (X'X)^{-1}X'\varepsilon \\ & = & \beta + (X'X)^{-1}X'\varepsilon \end{array}$$

d'où $\mathbb{E}\left(\hat{\beta}\right) = \beta + (X'X)^{-1}X'\mathbb{E}(\varepsilon) = \beta$.

L'estimateur des moindres carrés ordinaires est donc sans biais. D'où la proposition suivante.

Proposition 2.1 (Estimateur sans biais) *L'estimateur $\hat{\beta}$ obtenu est un estimateur sans biais de β, c'est - à - dire*

$$\mathbb{E}\left(\hat{\beta}\right) = \beta \tag{2.13}$$

2.2.2.2 Estimateur convergent

Comme en régression simple, l'estimateur obtenu est sans biais. On obtient de plus une expression très simple pour sa matrice de covariance $\mathrm{V}\left(\hat{\beta}\right)$. On rappelle que la matrice de covariance du vecteur aléatoire $\hat{\beta}$, ou matrice de variance - covariance est par définition :

$$\mathrm{V}\left(\hat{\beta}\right) = \mathbb{E}\left[\left(\hat{\beta}-\beta\right)\left(\hat{\beta}-\beta\right)'\right] \tag{2.14}$$

or $\hat{\beta}-\beta = (X'X)^{-1}X'\varepsilon$ et $\left(\hat{\beta}-\beta\right)' = \varepsilon' X\left(X'X\right)^{-1}$ puisque $\left(X'X\right)^{-1}$ est symétrique.

$$\left(\hat{\beta}-\beta\right)\left(\hat{\beta}-\beta\right)' = \left(X'X\right)^{-1}X'\varepsilon\varepsilon' X\left(X'X\right)^{-1}$$

d'où

$$\mathrm{V}\left(\hat{\beta}\right) = \left(X'X\right)^{-1}X'\mathbb{E}\left(\varepsilon\varepsilon'\right)X\left(X'X\right)^{-1} = \sigma_\varepsilon^2\left(X'X\right)^{-1}$$

Proposition 2.2 (Matrice de covariance) *La variance de $\hat{\beta}$ vaut*

$$V\left(\hat{\beta}\right) = \sigma_\varepsilon^2\left(X'X\right)^{-1} \tag{2.15}$$

Proposition 2.3 (Estimateur convergent) *L'estimateur $\hat{\beta}$ est un estimateur convergent, c'est - à - dire*

$$\lim_{n\to+\infty} V\left(\hat{\beta}\right) = 0 \tag{2.16}$$

Il suffit de remarquer que

$$\mathrm{V}\left(\hat{\beta}\right) = \frac{\sigma_\varepsilon^2}{n}\left(\frac{\left(X'X\right)^{-1}}{n}\right)^{-1} \quad \Longrightarrow \quad \lim_{n\to+\infty}\mathrm{V}\left(\hat{\beta}\right) = 0$$

L'estimateur est donc convergent.

L'estimateur des moindres carrés ordinaires est optimal en un certain sens. C'est ce que précise le résultat suivant, généralisation de celui vu en régression linéaire simple.

Theorème 2.1 (Gauss - Markov) *L'estimateur $\hat{\beta}$ est moindres carrés ordinaires est de variance minimale parmi les estimateurs linéaires sans biais de β.*

En effet, l'estimateur des moindres carrés ordinaires est qualifié de BLUE (*Best Linear Unbiased Estimator*), car il s'agit du meilleur estimateur linéaire sans biais (au sens qu'il fournit les variances les plus faibles pour les estimateurs).

2.2.2.3 Résidu et variance résiduelle

Les résidus sont définis par la relation suivante :

$$\hat{\varepsilon} = Y - \widehat{Y} \tag{2.17}$$

En nous servant du modèle, $Y = X\beta + \varepsilon$, nous avons une autre écriture des résidus :

$$\hat{\varepsilon} = Y - X\hat{\beta} = Y - X\left(X'X\right)^{-1}X'Y = \left[I - X\left(X'X\right)^{-1}X'\right]Y = PY$$

avec $P = I - X\left(X'X\right)^{-1}X'$. Cette matrice P est symétrique et idempotente $(P^2 = P)$. Elle est aussi telle que $PX = 0$ et $P\hat{\varepsilon} = \hat{\varepsilon}$.

Comme en régression linéaire simple, un estimateur « naturel » de la variance résiduelle est donné par :

$$\frac{1}{n}\sum_{t=1}^{t=n}\left(y_t - \hat{y}_t\right)^2 = \frac{1}{n}\sum_{t=1}^{t=n}\varepsilon_t^2 = \frac{1}{n}||\varepsilon||^2$$

Malheureusement cet estimateur est biaisé. Ce biais est néanmoins facile à corriger, comme le montre le résultat suivant. C'est une généralisation du résultat obtenu en régression linéaire simple, en remplaçant $n-2$ par $n-p-1$. On définit donc

$$\hat{\sigma}_\varepsilon^2 = \frac{1}{n-p-1}\sum_{t=1}^{t=n}\hat{\varepsilon}_t^2 = \frac{1}{n-p-1}\hat{\varepsilon}'\hat{\varepsilon} = \frac{1}{n-p-1}||\hat{\varepsilon}||^2 \tag{2.18}$$

Proposition 2.4 ($\hat{\sigma}_\varepsilon^2$ sans biais) *La statistique $\hat{\sigma}_\varepsilon^2$ est un estimateur sans biais de σ_ε^2.*

On a $\hat{\varepsilon} = PY = P(X\beta + \varepsilon) = P\varepsilon$ puisque $PX = 0$. D'où l'espérance de l'erreur est donnée par :

$$\mathbb{E}\left(\hat{\varepsilon}'\hat{\varepsilon}\right) = \mathbb{E}\left(\varepsilon' P' P\varepsilon\right) = \mathbb{E}\left(\varepsilon' P\varepsilon\right)$$

En utilisant le fait que la trace d'un scalaire est un scalaire, ilvient :

$$\begin{aligned}
\mathbb{E}\left(\varepsilon' P\varepsilon\right) &= \mathbb{E}\left[\mathrm{Tr}\left(\varepsilon' P\varepsilon\right)\right] = \mathbb{E}\left[\mathrm{Tr}\left(\varepsilon\varepsilon' P\right)\right] \\
&= \mathbb{E}\left(\varepsilon\varepsilon'\right)\mathrm{Tr}(P) = \sigma_\varepsilon^2 \mathrm{Tr}(P) = \sigma_\varepsilon^2 \mathrm{Tr}\left(I - X\left(X'X\right)^{-1}X'\right) \\
&= \sigma_\varepsilon^2 \mathrm{Tr}\left(I\right) - \sigma_\varepsilon^2 \mathrm{Tr}\left(X\left(X'X\right)^{-1}X'\right) = \sigma_\varepsilon^2 \mathrm{Tr}\left(I\right) - \sigma_\varepsilon^2 \mathrm{Tr}\left(\left(X'X\right)^{-1}X'X\right) \\
&= \sigma_\varepsilon^2\left[n - (p+1)\right] = \sigma_\varepsilon^2\left(n - p - 1\right)
\end{aligned}$$

car $\text{Tr}(I) = n$ et $\text{Tr}\left(\left(X'X\right)^{-1}X'X\right) = p+1$ car $\left(X'X\right)$ est de dimension $(p+1, p+1)$.

D'où l'estimateur de la variance de l'erreur ou variance résiduelle s'écrit donc :

$$\hat{\sigma}_\varepsilon^2 = \frac{1}{n-p-1}\sum_{t=1}^{t=n}\hat{\varepsilon}_t^2 \tag{2.19}$$

```
# variance de l'erreur
scale = model.scale
print("Variance de l'erreur : %.4f"%(scale))
```

```
## Variance de l'erreur : 282.9659
```

L'estimation de $\hat{\sigma}_\varepsilon$ vaut ici 16.82 et nous avons $n = 50$ pour $p = 2$ variables, ce qui donne $n - p - 1 = 47$ (degrés de liberté).

A partir de cet estimateur de la variance résiduelle, nous obtenons immédiatement un estimateur de la variance de $\hat{\beta}$ en remplaçant σ_ε^2 par son estimateur :

$$\hat{\text{V}}\left(\hat{\beta}\right) = \hat{\sigma}_{\hat{\beta}}^2 = \hat{\sigma}_\varepsilon^2\left(X'X\right)^{-1} \tag{2.20}$$

Nous avons donc un estimateur de l'écart - type de l'estimateur $\hat{\beta}_j$ de chaque coefficient de la régression β_j

$$\hat{\sigma}_{\hat{\beta}_j} = \sqrt{\hat{\sigma}_\varepsilon^2\left[\left(X'X\right)^{-1}\right]_{j+1,j+1}} \tag{2.21}$$

L'écriture $\left[\left(X'X\right)^{-1}\right]_{j+1,j+1}$ signifie le $(j+1)-$ème terme diagonale de la matrice $\left(X'X\right)^{-1}$ et non l'inverse du $(j+1)-$ième terme diagonal de la matrice $\left(X'X\right)$. Afin d'alléger les écritures, nous écrirons $\left(X'X\right)^{-1}_{j+1,j+1}$ au lieu de $\left[\left(X'X\right)^{-1}\right]_{j+1,j+1}$ et parfois $\hat{\sigma}_j$ pour $\hat{\sigma}_{\hat{\beta}_j}$.

On retourne les variances et covariances des estimateurs :

```
# Matrice des variances-covariances des estimateurs
covparams = model.cov_params()
```

Table 2.4 – Matrice des variances - covariances des estimateurs

	Intercept	T12	Vx
Intercept	115.4461971	-5.396963	0.1450821
T12	-5.3969629	0.265359	-0.0058560
Vx	0.1450821	-0.005856	0.0313864

2.2.3 L'analyse de la variance

L'analyse de la variance permet d'appréhender l'influence des variables exogènes sur la variable endogène.

2.2.3.1 Équation d'analyse de la variance

Cette équation de décomposition de la variance qui permet de juger de la qualité de l'ajustement d'un modèle stipule que la variabilité totale est égale à la variabilité expliquée plus la variabilité des résidus. Elle s'écrit :

$$SCT = SCE + SCR \tag{2.22}$$

avec $SCT = y^{'}y - n\overline{y}^2 = n \times \mathrm{V}(y)$; $SCE = \hat{y}^{'}\hat{y} - n\overline{y}^2$ et $SCR = (y-\hat{y})^{'}(y-\hat{y}) = \hat{\varepsilon}^{'}\hat{\varepsilon}$.

Cette équation permet d'apprécier la qualité de l'ajustement du modèle considéré ; en effet, plus la variance expliquée (SCE) est « proche » de la variance totale (SCT) (SCE $\longrightarrow$ SCT), meilleur est l'ajustement global du modèle.

La tableau 2.5 donne le résumé sur la décomposition de la variance dans la cas d'un modèle de régression linéaire multiple.

Table 2.5 – Analyse de la variance pour une régression multiple

Source de variation	Somme des carrés (SC)	Degré de liberté (ddl)	Carrés moyens (SC/ddl)
Exogène	SCE	p	SCE/k
Résiduelle	SCR	$n-p-1$	$SCR/n-p-1$
Totale	SCT	$n-1$	

```
# Somme des carrés totale - SCT
n = donnee.shape[0]
p = 2
SCT = n*np.var(donnee["O3"])
# Somme des carrés expliquée
SCE = n*np.var(model.fittedvalues)
# Somme des carrés résiduelle
SCR = n*np.var(model.resid)
var_eq = pd.DataFrame({
  "Source de variance" : ["Exogène","Résiduelle","Totale"],
  "Somme des carrés (SC)" : np.around(np.array([SCE, SCR,SCT]),4),
  "Degré de liberté (ddl)" : np.around(np.array([1, n-p-1, n-1]),4),
  "Carré moyen (SC/ddl)" : np.around(np.array([SCE/1,SCE/(n-p-1),np.nan]),4)
})
```

Table 2.6 – Tableau d'analyse de la variance - Données Ozone

Source de variance	Somme des carrés (SC)	Degré de liberté (ddl)	Carré moyen (SC/ddl)
Exogène	14691.82	1	14691.8215
Résiduelle	13299.40	47	312.5919
Totale	27991.22	49	

Mais puisque ces valeurs dépendent des unités de mesure, on préfère utiliser un nombre sans dimension pour juger de la qualité d'un ajustement : le coefficient de détermination multiple.

2.2.3.2 Coefficient de détermination

L'équation (2.22) va nous permettre de juger de la qualité de l'ajustement d'un modèle ; en effet, plus la variance expliquée est « proche » de la variance totale, meilleur est l'ajustement global du modèle. C'est pourquoi nous calculons le rapport SCE sur SCT :

$$R^2 = \frac{SCE}{SCT} = \frac{\sum_{t=1}^{t=n} \left(\hat{y}_t - \overline{y}\right)^2}{\sum_{t=1}^{t=n} \left(y_t - \overline{y}\right)^2} = 1 - \frac{\sum_{t=1}^{t=n} \varepsilon_t^2}{\sum_{t=1}^{t=n} \left(y_t - \overline{y}\right)^2} = 1 - \frac{SCR}{SCT} \tag{2.23}$$

R^2 mesure la proportion de la variance de la variable endogène expliquée par la régression de cette dernière sur la variable exogène. Plus précisément, R^2 montre le rôle joué par l'ensemble des variables exogènes sur l'évolution de la variable endogène. Il d'autant meilleur qu'il est proche de 1.

```
# Coefficient de détermination
r2squared = model.rsquared
print('Coefficient de détermination : %.4f'%(r2squared))
```

```
## Coefficient de détermination : 0.5249
```

2.2.3.3 Coefficient de détermination ajustée

Toutefois, le coefficient de détermination multiple seul est insuffisant pour juger de la qualité d'un modèle. Il masque l'influence des variables exogènes considérées isolément sur le comportement de l'endogène. Il ne peut donc pas se substituer au test des paramètres du modèle. En outre, ce coefficient ne tient compte ni du nombre d'observations, ni du nombre de variables explicatives dans le modèle.

C'est compte tenu de ces insuffisances que l'on a convenu de proposer un coefficient de détermination noté $\overline{R}^2$, appelé « R^2 corrigé ou ajusté », défini par :

$$\overline{R}^2 = 1 - \frac{n-1}{n-p-1}\left(1 - R^2\right) \tag{2.24}$$

```
# Coefficient de détermination adjusté
adjr2squared = model.rsquared_adj
print('Coefficient de détermination adj. : %.4f'%(adjr2squared))
```

```
## Coefficient de détermination adj. : 0.5047
```

2.2.4 Prévision

Un des buts de la régression est de proposer des prévisions pour la variable à expliquer y lorsque nous avons de nouvelles valeurs de x. Soit une nouvelle valeur $x_{n+1}^{'}\left(x_{n+1,1}, \ldots, x_{n+1,p}\right)$, nous voulons prédire y_{n+1}. Le modèle spécifié s'écrit :

$$Y = X\beta + \varepsilon \tag{2.25}$$

Le modèle estimé à l'aide d'un échantillon de taille n observations s'écrit :

$$Y = X\hat{\beta} + \hat{\varepsilon} \tag{2.26}$$

Or

$$y_{n+1} = x_{n+1}^{'}\beta + \varepsilon_{n+1}$$

avec $\mathbb{E}\left(\varepsilon_{n+1}\right) = 0$, $\mathrm{V}\left(\varepsilon_{n+1}\right) = \sigma_\varepsilon^2$ et $\mathrm{Cov}\left(\varepsilon_{n+1}, \varepsilon_t\right) = 0$ pour $t = 1, \ldots, n$. Nous pouvons prédire la valeur correspondante grâce ua modèle ajusté :

$$\widehat{y}_{n+1}^{p} = \hat{\beta}_0 + \sum_{j=1}^{j=p} \hat{\beta}_j x_{j,n+1} = x_{n+1}^{'}\hat{\beta} \tag{2.27}$$

Deux types d'erreurs vont entacher la prévision, la première due à l'incertitude sur ε_{n+1} et l'autre due à l'estimation.

Proposition 2.5 (Variance de la prévision $\widehat{y}_{n+1}^{p}$) *La variance de valeur prévue de $\widehat{y}_{n+1}^{p}$ vaut :*

$$V(\widehat{y}_{n+1}^{p}) = V\left(x_{n+1}^{'}\hat{\beta}\right) = x_{n+1}^{'} V\left(\hat{\beta}\right) x_{n+1} = \sigma_\varepsilon^2 x_{n+1}^{'} \left(X^{'}X\right)^{-1} x_{n+1}$$

La variance de $\widehat{y}_{n+1}^{p}$ nous donne une idée de la stabilité de l'estimation. En prévision, on s'intéresse généralement à l'erreur que l'on commet entre la vraie valeur à prévoir y_{n+1} et celle que l'on prévoit $\widehat{y}_{n+1}^{p}$. L'erreur peut être simplement résumée par la différence entre ces deux valeurs, c'est ce que nous appelerons **l'erreur de prévision** définie par :

$$\widehat{\varepsilon}_{n+1}^{p} = y_{n+1} - \widehat{y}_{n+1}^{p} \tag{2.28}$$

Proposition 2.6 (Erreur de prévision) *L'erreur de prévision, définie par $\widehat{\varepsilon}_{n+1}^{p} = y_{n+1} - \widehat{y}_{n+1}^{p}$ satisfait les propriétés suivantes :*

$$\begin{cases} \mathbb{E}\left(\widehat{\varepsilon}_{n+1}^{p}\right) & = 0 \\ V(\widehat{\varepsilon}_{n+1}^{p}) & = \sigma_\varepsilon^2 \left[1 + x_{n+1}^{'} \left(X^{'}X\right)^{-1} x_{n+1}\right] \end{cases} \tag{2.29}$$

Calculons la variance de l'erreur de prévision $\widehat{\varepsilon}_{n+1}^{p} = y_{t+h} - \widehat{y}_{t+h}^{p}$:

$$\begin{aligned} \mathrm{V}\left(\widehat{\varepsilon}_{n+1}^{p}\right) &= \mathrm{V}\left(x_{n+1}^{'}\beta + \varepsilon_{n+1} - x_{n+1}^{'}\hat{\beta}\right) = \sigma_\varepsilon^2 + \sigma_\varepsilon^2 x_{n+1}^{'} \left(X^{'}X\right)^{-1} x_{n+1} \\ &= \sigma_\varepsilon^2 \left[1 + x_{n+1}^{'} \left(X^{'}X\right)^{-1} x_{n+1}\right] \end{aligned}$$

Nous retrouvons bien l'incertitude due aux erreurs σ_ε^2 sur laquelle vient s'ajouter l'incertitude d'estimation.

On calcule la valeur prédicte pour les valeurs `T12 = 5` et `Vx=0`.

```
# Prévision
forecast = model.get_prediction(exog=dict(T12=5,Vx=0))
prediction = pd.DataFrame(forecast.predicted,columns = ["forecast"])
```

Table 2.7 – Prévision sur les valeurs T12=5 et VX=0

forecast
48.143

Remarque 2.2

Puiosque l'estimateur $\hat{\beta}$ est un estimateur sans biais de β et l'espérance de ε vaut 0, les espérances de y_{n+1} et $\hat{y}^p_{n+1}$ sont identiques. La variance de l'erreur de prévision s'écrit :

$$\mathrm{V}\left(y_{n+1} - \hat{y}^p_{n+1}\right) = \mathbb{E}\left[y_{n+1} - \hat{y}^p_{n+1} - \mathbb{E}\left(y_{n+1}\right) + \mathbb{E}\left(\hat{y}^p_{n+1}\right)\right]^2 = \mathbb{E}\left(y_{n+1} - \hat{y}^p_{n+1}\right)^2 \quad (2.30)$$

Nous voyons donc ici que la variance de l'erreur de prévision est mesurée par l'erreur quadratique moyenne de prévision (EQMP) ou FMSE (*Forecast Mean Squared Error*).

2.3 Inférence dans le modèle gaussien

Nous allons désormais supposer que les erreurs suivent une loi normale, c'est - à - dire

$$\varepsilon \sim \mathcal{N}\left(0, \sigma^2_\varepsilon I_n\right) \quad (2.31)$$

C'est grâce à cette hypothèse que l'inférence statistique peut se réaliser car c'est elle qui va préciser la distribution statistique des estimateurs $\hat{\beta}$. Elle joue donc un rôle important essentiel bien qu'elle soit difficile à vérifier. Il existe des tests de normalité, mais ils ne sont applicables sur les termes d'erreur qui par définition demeure inconnu.

L'hypothèse gaussienne va nous permettre de calculer la vraisemblance et donc les estimateurs du maximum de vraisemblance (EMV). Cette hypothèse va nous permettre également de calculer des régions de confiance et de proposer des tests.

2.3.1 Estimateur du maximum de vraisemblance

Calculons la vraisemblance de l'échantillon. La vraisemblance est la densité de l'échantillon vue comme fonction des paramètres. Grâce à l'indépendance des erreurs, les observations sont indépendantes et la vraisemblance s'écrit :

$$\mathrm{L}\left(\beta, \sigma^2_\varepsilon\right) = \left(\frac{1}{2\pi\sigma^2}\right)^{n/2} \exp\left\{-\frac{1}{2\sigma^2_\varepsilon}\sum_{t=1}^{t=n}\left(y_t - \beta_0 - \sum_{j=1}^{j=p}\beta_j x_{tj}\right)^2\right\} \quad (2.32)$$

La fonction log-vraisemblance associée s'écrit :

$$\mathcal{L}\left(\beta, \sigma_\varepsilon^2\right) = -\frac{n}{2}\log\left(2\pi\sigma_\varepsilon^2\right) - \frac{1}{2\sigma_\varepsilon^2}\left\|Y - X\beta\right\|^2 \tag{2.33}$$

Nous obtenons

$$\begin{cases} \dfrac{\partial}{\partial\beta}\mathcal{L}\left(\beta, \sigma_\varepsilon^2\right) & = \dfrac{1}{2\sigma_\varepsilon^2}\dfrac{\partial}{\partial\beta}\left(\left\|Y - X\beta\right\|^2\right) \\ \dfrac{\partial}{\partial\sigma_\varepsilon^2}\mathcal{L}\left(\beta, \sigma_\varepsilon^2\right) & = -\dfrac{n}{2\sigma_\varepsilon^2} + \dfrac{1}{2\sigma_\varepsilon^4}\left\|Y - X\beta\right\|^2 \end{cases}$$

A partir de la première équation du système, nous avons évidemment $\hat{\beta}_{\text{EMV}} = \hat{\beta}$ et à partir de la seconde, nous avons

$$\hat{\sigma}^2_{\text{EMV}} = \frac{1}{n}\left\|Y - X\hat{\beta}_{\text{EMV}}\right\|^2 = \frac{1}{n}\left\|Y - X\hat{\beta}\right\|^2 \tag{2.34}$$

et donc que $\hat{\sigma}^2_{\text{EMV}} = \dfrac{n-p-1}{n}\hat{\sigma}^2_\varepsilon$. L'estimateur du maximum de vraisemblance est donc biaisé.

```
# fonction log-vraisemblance
import numpy as np
def logvraisemblance(parameters):
  beta0 = parameters[0]
  betap = parameters[1:-1]
  sigma = parameters[-1]
  for i in np.arange(0, len(x)):
    y_exp = beta0+ np.dot(x,betap)
    loglik = (len(x)/2 * np.log(2 * np.pi) + len(x)/2 * np.log(sigma ** 2) +
              1 /(2 * sigma ** 2) * np.linalg.norm(y-y_exp)**2)
  return loglik

# Estimation des paramètres par maximum de vraisemblance
from scipy.optimize import minimize
x = donnee[['T12','Vx']].values
y = donnee.O3.values
opt = minimize(logvraisemblance,np.array([1,1,1,1]), method="L-BFGS-B")
emv_params = pd.DataFrame({"Intercept": opt.x[0], "T12": opt.x[0],"Vx":opt.x[2],
                           "scale" : opt.x[3]**2},index=["EMV"])
```

Table 2.8 – Estimateurs par maximum de vraisemblance

	Intercept	T12	Vx	scale
EMV	35.453	35.453	0.8736	265.988

Les valeurs estimées des paramètres sont identiques à celles obtenues les MCO, sauf celle de la variance de l'erreur. On s'attendait à ce résultat.

2.3.2 Lois des estimateurs

Grâce à l'hypothèse gaussienne, nous pouvons établier la loi des estimateurs. Nous commençons cette section par un rappel sur les vecteurs gaussiens.

2.3.2.1 Quelques rappels

Un vecteur aléatoire Y de $\mathbb{R}^n$ est dit gaussien si toute combinaison linéaire de ses composantes est une variable aléatoire gaussienne.Ce vecteur admet alors une espérance $\mathbb{E}[Y] = \mu$ et une matrice de variance - covariance $\Sigma_Y = \mathbb{E}\left[(Y-\mu)(Y-\mu)^{'}\right]$ qui caractérisent complètement sa loi. On note dans ce cas $Y \sim \mathcal{N}(\mu, \Sigma_Y)$. On montre alors que les composantes d'un vecteur gaussien $Y = [Y_1, ..., Y_n]^{'}$ sont indépendantes si et seulement si Σ_Y est diagonale.

Soit $Y \sim \mathcal{N}(\mu, \Sigma_Y)$ un vecteur gaussien. Il admet une densité f sur $\mathbb{R}^n$ si et seulement si sa matrice de dispersion Σ_Y est inversible, auquel cas :

$$f(y) = \frac{1}{(2\pi)^{n/2}\sqrt{\det(\Sigma_Y)}} \exp\left\{-\frac{1}{2}(y-\mu)^{'}\Sigma_Y(y-\mu)\right\} \tag{2.35}$$

Dans ce cas, on montre aussi la propriété suivante.

Proposition 2.7 (Vecteur gaussien et Loi du χ^2) *Soit $Y \sim \mathcal{N}(\mu, \Sigma_Y)$ un vecteur gaussien. Si Σ_Y est inversible, alors*

$$(Y-\mu)^{'}\Sigma_Y(Y-\mu) \sim \chi^2_n \tag{2.36}$$

loi de χ^2 à n degrés de liberté.

2.3.2.2 Nouvelles propriétés

Notons au préalable que, pour ce qui nous concerne, la gaussianité des résidus implqiue celle du vecteur Y :

$$\varepsilon \sim \mathcal{N}(0, \sigma^2_\varepsilon I_n) \quad \Longrightarrow \quad Y = X\beta + \varepsilon \sim \mathcal{N}(X\beta, \sigma^2_\varepsilon I_n) \tag{2.37}$$

Proposition 2.8 (Lois des estimateurs avec variance connue) *Sous les hypothèses, nous avons :*

1. *$\hat{\beta}$ est un vecteur gaussien de moyenne β et de variance $\sigma^2_\varepsilon \left(X^{'}X\right)^{-1}$:*

$$\hat{\beta} \sim \mathcal{N}\left(\beta, \sigma^2_\varepsilon \left(X^{'}X\right)^{-1}\right) \tag{2.38}$$

2. *$\hat{\beta}$ et $\hat{\sigma}^2_\varepsilon$ sont indépendants ;*
3. $\dfrac{n-p-1}{\sigma^2_\varepsilon}\hat{\sigma}^2_\varepsilon \sim \chi^2_{n-p-1}$.

Proposition 2.9 (Lois des estimateurs avec variance inconnue) *Sous les hypothèses*

1. *pour* $j = 0, \dots, p$*, nous avons*

$$T_j = \frac{\hat{\beta}_j - \beta_j}{\hat{\sigma}_j} \sim \mathcal{T}_{n-p-1} \tag{2.39}$$

2. *Soit* R *une matrice de taille* $q \times (p+1)$ *de rang* $q\,(q \leq (p+1))$ *alors la variable aléatoire*

$$\frac{1}{q\hat{\sigma}_\varepsilon^2}\left(\hat{\beta}-\beta\right)' R' \left[R\left(X'X\right)^{-1} R'\right]^{-1} R\left(\hat{\beta}-\beta\right) \sim \mathcal{F}_{q,n-p-1} \tag{2.40}$$

De ces résultats vont découler les régions de confiance.

2.3.3 Intervalles et régions de confiance

Les logiciels et certains ouvrages donnent des intervalles de confiance pour les paramètres pris séparément. Cependant ces intervalles de confiance ne tiennent pas compte de la dépendance des paramètres, ce qui conduirait à construire plutôt des régions de confiance (RC).

2.3.3.1 Intervalles de confiance

Les intervalles classiques de confiance relatifs aux paramètres du modèle s'effectuent suivant la même logique qu'avec le modèle linéaire simple ; le degré de liberté, qui était de $(n-2)$, est maintenant à $(n-p-1)$.

Proposition 2.10 (IC des paramètres) *On a les résultats suivants :*

1. *Un intervalle de confiance, de niveau* $1-\alpha$*, pour un* β_j *pour* $j = 0, \dots, p$ *est donné par*

$$\left[\hat{\beta}_j - t_{n-p-1}^{1-\alpha/2}\hat{\sigma}_j, \hat{\beta}_j + t_{n-p-1}^{1-\alpha/2}\hat{\sigma}_j\right] \tag{2.41}$$

2. *Un intervalle de confiance, de niveau* $1-\alpha$*, pour* σ_ε^2 *est donné par*

$$IC(\sigma_\varepsilon^2) = \left[\frac{(n-p-1)}{\chi_1^2}\hat{\sigma}_\varepsilon^2; \frac{(n-p-1)}{\chi_2^2}\hat{\sigma}_\varepsilon^2\right] \tag{2.42}$$

avec χ_1^2 *ayant la probabilité* $\alpha/2$ *d'être dépassée et* χ_2^2 *la probabilité* $1-\alpha/2$ *d'être dépassée à* $n-p-1$ *degrés de liberté. On rappelle que :*

$$\frac{(n-p-1)\hat{\sigma}_\varepsilon^2}{\sigma_\varepsilon^2} \sim \chi^2_{(n-p-1)} \tag{2.43}$$

Exemple 2.3

Table 2.9 – 10 données de température à 12 h, teneur en ozone, vent et nébulosité

	O3	T12	Vx	Ne12
19960422	63.6	13.4	9.35	7
19960429	89.6	15.0	5.40	4
19960506	79.0	7.9	19.30	8
19960514	81.2	13.1	12.60	7
19960521	88.0	14.1	-20.30	6
19960528	68.4	16.7	-3.69	7
19960605	139.0	26.8	8.27	1
19960612	78.2	18.4	4.93	7
19960619	113.8	27.2	-4.93	6
19960627	41.8	20.6	-3.38	8

Nous traitons les 50 données journalières concernant la concentration en ozone. La variable à expliquer est la concentration en ozone notée O3 et les variables explicatives sont la température notée T12, le vent noté Vx et la nébulosité noté Ne12.

Le modèle de régression devient :

$$\texttt{O3} = \beta_0 + \beta_1 \texttt{T12} + \beta_2 \texttt{Vx} + \beta_3 \texttt{Ne12} + \varepsilon \tag{2.44}$$

```
# Estimation des paramètres
model2 = smf.ols(formula = "O3~T12+Vx+Ne12",data=donnee).fit()
# Extraction des résultats
model2_params = model2.summary2().tables[1]
```

Table 2.10 – Coefficients du modèle linéaire : $\texttt{O3}_t = \beta_0 + \beta_1 \texttt{T12}_t + \beta_2 \texttt{Vx} + \beta_3 \texttt{Ne12} + \varepsilon_t$

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	84.5473	13.6067	6.2136	0.0000	57.1584	111.9362
T12	1.3150	0.4974	2.6438	0.0112	0.3138	2.3163
Vx	0.4864	0.1675	2.9033	0.0057	0.1492	0.8237
Ne12	-4.8934	1.0270	-4.7647	0.0000	-6.9606	-2.8261

On a le modèle ajusté suivant :

$$\widehat{\texttt{O3}} = 84.5473 + 1.3150\texttt{T12} + 0.4864\texttt{Vx} - 4.8934\texttt{Ne12} \tag{2.45}$$

On construit les intervalles de confiance pour nos paramètres.

```
# Intervalle de confiance des paramètres
confint = model2.conf_int(alpha=0.05)
confint.columns = ["[0.025","0.975]"]
```

Table 2.11 – Intervalle de confiance des paramètres

	[0.025	0.975]
Intercept	57.1584152	111.9362500
T12	0.3138112	2.3162807
Vx	0.1491857	0.8237055
Ne12	-6.9606085	-2.8261372

2.3.3.2 Régions de confiance

Proposition 2.11 (RC des paramètres) *Une région de confiance pour q paramètres de niveau $1-\alpha$ est donnée*

1. *Lorsque σ_ε est connue, par*

$$RC_\alpha(R\beta) = \left\{ R\beta \in \mathbb{R}^q, \frac{1}{\sigma_\varepsilon^2}\left(\hat{\beta}-\beta\right)' R' \left[R\left(X'X\right)^{-1}R'\right]^{-1} R\left(\hat{\beta}-\beta\right) \sim \chi^2_q(1-\alpha) \right\} \tag{2.46}$$

2. *Lorsque σ_ε est inconnue, par*

$$RC_\alpha(R\beta) = \left\{ R\beta \in \mathbb{R}^q, \frac{1}{q\hat{\sigma}_\varepsilon^2}\left(\hat{\beta}-\beta\right)' R' \left[R\left(X'X\right)^{-1}R'\right]^{-1} R\left(\hat{\beta}-\beta\right) \sim \mathcal{F}^{1-\alpha}_{q,n-p-1} \right\} \tag{2.47}$$

où R est la matrice de taille $q\times(p+1)$ dont tous les éléments sont nuls sauf les $[R]_{i+1,j+1}$ qui valent 1.

Donons un exemple illustrant le second point de la proposition précédente.

Exemple 2.4

Considérons le cas $p=1$ et $q=2$, la matrice $R=I_2$, de sorte que

$$R\left(\hat{\beta}-\beta\right) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} \hat{\beta}_0-\beta_0 \\ \hat{\beta}_1-\beta_1 \end{pmatrix} = \begin{pmatrix} \hat{\beta}_0-\beta_0 \\ \hat{\beta}_1-\beta_1 \end{pmatrix}$$

Si la constante fait partie du modèle, X est la matrice $(n\times 2)$ dont la première colonne est uniquement composée de 1 et la seconde est composée des x_i, si bien que

$$X'X = \begin{pmatrix} n & \sum_{t=1}^{t=n} x_t \\ \sum_{t=1}^{t=n} x_t & \sum_{t=1}^{t=n} x_t^2 \end{pmatrix} = \begin{pmatrix} n & n\overline{x} \\ n\overline{x} & \sum_{t=1}^{t=n} x_t^2 \end{pmatrix}$$

Et le deuxième point de la proposition précédente s'écrit :

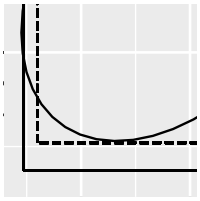
$$\frac{1}{2\hat{\sigma}_\varepsilon^2}\left[n\left(\hat{\beta}_0-\beta_0\right)^2 + 2n\overline{x}\left(\hat{\beta}_0-\beta_0\right)\left(\hat{\beta}_1-\beta_1\right) + \sum_{t=1}^{t=n} x_t^2\left(\hat{\beta}_1-\beta_1\right)^2\right] \sim \mathcal{F}_{2,n-2}$$

qui est exactement l'équation permettant de construire une ellipse de confiance pour $\beta=(\beta_0,\beta_1)$. En effet, cette région de confiance est une ellipse qui tient compte de la corrélation entre $\hat{\beta}_0$ et β_1.

Exemple 2.5 *Concentration en ozone*

Nous allons dessiner les régions de confiance de tous les couples de paramètres et les comparer graphiquemeent aux intervalles de confiance pour chaque paramètre pris indépendamment (ellipse *versus* rectange). Nous choisissons un intervalle de confiance à 95% pour chaque paramètre et une région de confiance à 95%.

```
# Simulation d'une loi normale multivariée
from plotnine import *
import patchworklib as pw
import scipy.stats as st
sim_norm = st.multivariate_normal.rvs(model2.params,model2.cov_params(),100,
                                      np.random.seed(123))
def plot_ellipse(i1,i2,name1,name2):
  x_min, x_max = confint.iloc[[i1,i2],0],confint.iloc[[i1,i2],1]
  xx_min = np.min(sim_norm[:,[i1,i2]],axis=0)
  xx_max = np.max(sim_norm[:,[i1,i2]],axis=0)
  # Région de confiance
  p = (pd.DataFrame(sim_norm[:,[i1,i2]],columns=[f"{name1}",f"{name2}"]) >>
       ggplot(aes(x=f"{name1}",y=f"{name2}"))+
       geom_rect(aes(xmin=x_min[0],xmax=x_max[0],ymin=x_min[1],ymax=x_max[1]),
                 alpha=0.0,color="black",linetype ='dashed')+
       geom_rect(aes(xmin=xx_min[0],xmax=xx_max[0],ymin=xx_min[1],ymax=xx_max[1]),
                 alpha=0.0,color="black",linetype ='-')+
       stat_ellipse()+labs(x=f"$\\beta_{i1}$",y=f"$\\beta_{i2}$"))
  return p
# Application
g1 = pw.load_ggplot(plot_ellipse(0,1,"Intercept","T12"),figsize=(2,3))
g2 = pw.load_ggplot(plot_ellipse(0,2,"Intercept","Vx"),figsize=(2,3))
g3 = pw.load_ggplot(plot_ellipse(0,3,"Intercept","Ne12"),figsize=(2,3))
g4 = pw.load_ggplot(plot_ellipse(1,2,"T12","Vx"),figsize=(2,3))
g5 = pw.load_ggplot(plot_ellipse(1,3,"T12","Ne12"),figsize=(2,3))
g6 = pw.load_ggplot(plot_ellipse(2,3,"Vx","Ne12"),figsize=(2,3))
# Run once
# g = (g1|g2|g3)/(g4|g5|g6)
# g.savefig("figure.jpg")
```



Afin d'observer la corrélation entre les paramètres, nous pouvons regarder l'orientation du grand axe de l'ellipse. Si cet axe n'est pas parallèle aux axes du repère, il y a corrélation. Ainsi, nous observons que $\hat{\beta}_0$ et $\hat{\beta}_1$ sont fortement corrélés. Il en est de même avec $\left(\hat{\beta}_0, \hat{\beta}_2\right)$ et $\left(\hat{\beta}_1, \hat{\beta}_2\right)$.

Enfin rappelons que nous pouvons calculer un intervalle de confiance à 95% pour $\hat{\sigma}^2_\varepsilon$ avec les commandes suivantes :

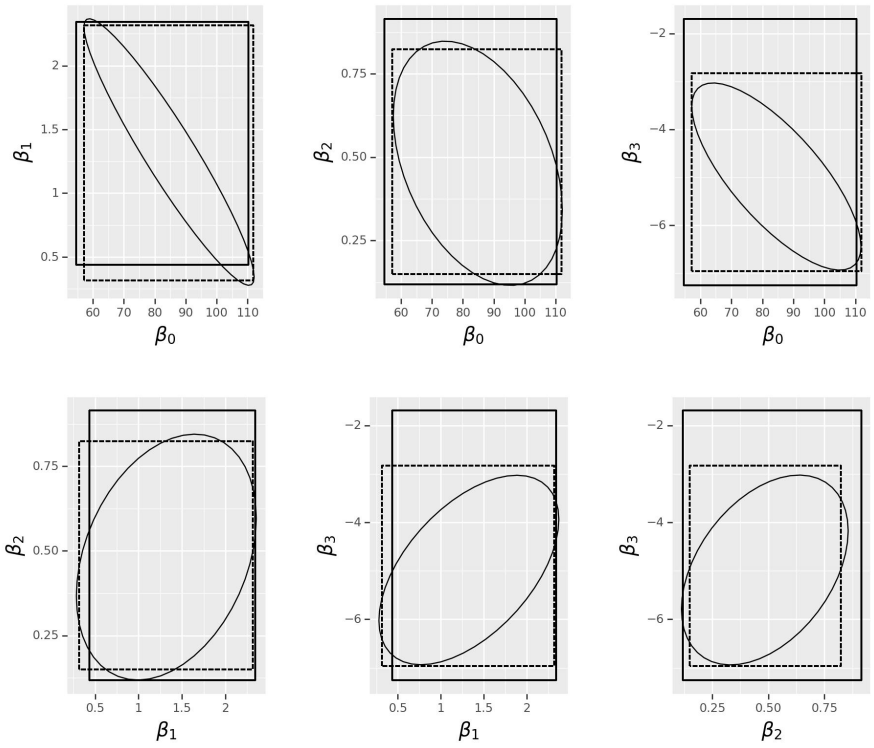


Figure 2.3 – Régions de confiance et rectangle des couples de paramètres

```
# IC pour la variance de l'erreur
(model2.scale*model2.df_resid/st.chi2.ppf(0.975,model2.df_resid),
 model2.scale*model2.df_resid/st.chi2.ppf(0.025,model2.df_resid))
```

```
## (133.66985879803255, 305.37055837566083)
```

2.3.4 Prévision

Soit $x_{n+1}^{'} = (x_{n+1,1}, \dots, x_{n+1,p})$ une nouvelle valeur pour laquelle nous voulons prédire la variable à expliquer y_{n+1} définie par :

$$y_{n+1} = x_{n+1}^{'}\beta + \varepsilon_{n+1}$$

avec $\varepsilon_{n+1} \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ indépendant des $(\varepsilon_t)_{1 \leq t \leq n}$. A partir des n observations précédentes, nous avons pu calculer un estimateur $\hat{\beta}$ de β. Nous nous servons de cet estimateur pour prévoir y_{n+1} par

$$\hat{y}_{n+1}^p = x_{n+1}^{'}\hat{\beta}$$

Pour quantifier l'erreur de prévision $\hat{\varepsilon}_{n+1}^p = y_{n+1} - \hat{y}_{n+1}^p$, on utilise la décomposition :

$$y_{n+1} - \hat{y}_{n+1}^p = x_{n+1}^{'}\beta + \varepsilon_{n+1} - x_{n+1}^{'}\hat{\beta} = x_{n+1}^{'}\left(\beta - \hat{\beta}\right) + \varepsilon_{n+1}$$

qui est la somme de deux variables aléatoires gaussiennes indépendantes puisque $\hat{\beta}$ est construit à partir des $(\varepsilon_t)_{1 \leq t \leq n}$. On en déduit que $(y_{n+1} - \hat{y}_{n+1}^p)$ est une variable gaussienne, dont moyenne et variance ont été calculées précédemment, ce qui donne :

$$y_{n+1} - \hat{y}^p_{n+1} \sim \mathcal{N}\left(0, \sigma^2_\varepsilon \left[1 + x'_{n+1}\left(X'X\right)^{-1} x_{n+1}\right]\right)$$

Nous pouvons maintenant donner un intervalle de confiance pour y_{n++1}.

Proposition 2.12 (IC pour la prévision) *Un intervalle de confiance de niveau* $1-\alpha$ *pour* y_{n+1} *est donné par :*

$$\left[x'_{n+1}\hat{\beta} - t^{1-\alpha/2}_{n-p-1}\sigma_\varepsilon\sqrt{1 + x'_{n+1}\left(X'X\right)^{-1}x_{n+1}}, x'_{n+1}\hat{\beta} + t^{1-\alpha/2}_{n-p-1}\sigma_\varepsilon\sqrt{1 + x'_{n+1}\left(X'X\right)^{-1}x_{n+1}}\right] \tag{2.48}$$

Reprenons l'exemple de la concentration en ozone. Nous avons modélisé les pics d'ozone par `T12`, `Vx` et `Ne12`.

```
# Prévision sur les valeurs observées
forecast = model2.get_prediction(exog=dict(T12=5,Vx=20,Ne12=12))
frame = forecast.summary_frame(alpha=0.05)
```

Table 2.12 – Prévision

mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
42.131	9.091	23.832	60.43	8.677	75.585

La colonne `mean` indique la valeur prédicte définie par 2.27.

```
# Valeur prédicte
valpred = forecast.predicted_mean
print('Valeur prédicte : %.4f'%(valpred))
```

```
## Valeur prédicte : 42.1310
```

La colonne `mean_se` donne l'écart-type de l'erreur due à l'estimation des paramètres du modèle :

$$\hat{\sigma}\sqrt{x'_{n+1}(X'X)^{-1}x_{n+1}} \tag{2.49}$$

```
# Ecart-type de l'estimation de la moyenne
se_mean = forecast.se_mean
print("Ecart-type de l'estimation de la moyenne : %.4f"%(se_mean))
```

```
## Ecart-type de l'estimation de la moyenne : 9.0909
```

Les colonnes `mean_ci_lower` et `mean_ci_upper` sont les bornes inférieures et supérieures de l'intervalle de confiance pour l'estimation de la moyenne (équation **??**).

```
# Intervalle de confiance de l'estimation de la moyenne
ic_mean_pred = forecast.conf_int(alpha=0.05)
print('Intervalle de confiance :',ic_mean_pred[0])
```

```
## Intervalle de confiance : [23.83204766 60.42995084]
```

Enfin, les colonnes `obs_ci_lower` et `obs_ci_upper` sont les bornes inférieures et supérieures de l'intervalle de confiance de prédiction. En particulier, la dernière valeur correspond au calcul de l'équation

$$\hat{\sigma}\sqrt{1 + x'_{n+1}(X'X)^{-1}x_{n+1}} \tag{2.50}$$

On vérifie :

```
# qpred contient la borne inférieure et supérieure
qpred = st.t.interval(alpha=0.95,df=model2.df_resid)
ic_pred = forecast.predicted_mean + qpred*forecast.se_obs
print('Intervalle de confiance :',ic_pred)
```

```
## Intervalle de confiance : [ 8.67679131 75.5852072 ]
```

2.3.5 Tests d'hypothèses

Reprenons l'exemple de la prévision des pics d'ozone. Nous avons décider de modéliser les pics d'ozone `O3` par la température à midi `T12`, le vent `Vx` et la nébulosité à midi `Ne12`. Il paraît alors raisonnable de se poser par exemple les questions suivantes :

1. Est ce que la valeur de `O3` est influencée par la variable `Vx` ?
2. Y a - t - il un effet nébulosité ?
3. Est ce que la valeur de `O3` est influencée par `Vx` et `T12` ?

Rappelons le modèle utilisé est le suivant :

$$\texttt{O3}_t = \beta_0 + \beta_1 \texttt{T12} + \beta_2 \texttt{Vx} + \beta_3 \texttt{Ne12} + \varepsilon_t$$

Nous pouvons expliciter les trois questions précédentes en terme de test d'hypothèses :

1. correspond à $H_0 : \beta_2 = 0$, contre $H_1 : \beta_2 \neq 0$.
2. correspond à $H_0 : \beta_3 = 0$, contre $H_1 : \beta_3 \neq 0$.
3. correspond à $H_0 : \beta_1 = \beta_2 = 0$ contre $H_1 : \beta_1 \neq 0$ ou $\beta_2 \neq 0$.

Ces tests d'hypothèses reviennent à tester la nullité d'un ou plusieurs paramètres en même temps. Si l'on teste plusieurs paramètres à la fois, on parle de test de nullité simulatnée des coefficients. Ceci signifie que, sous l'hypothèse H_0, certains coefficients sont nuls, donc les variables correspondant à ceux - ci ne sont pas utiles pour la modélisation du phénomène. Ce cas de figure revient à comparer deux modèles emboîtés, l'un étant un cas particulier de l'autre.

2.3.5.1 Test de comparaison d'un ensemble de paramètres à un ensemble de valeur

Il s'agit de tester simultanément l'égalité d'un sous ensemble de coefficients de régression à des valeurs fixées. Ce sont des tests du genre :

$$\begin{cases} H_0 : \beta_q = \overline{\beta}_q \\ H_1 : \beta_q \neq \overline{\beta}_q \end{cases}$$

q étant le nombre de coefficients retenus, c'est-à-dire la dimension de chacun des vecteurs β_q. Pour effectuer ce test, on calcule la statistique de Fisher suivante :

$$F_0 = \frac{1}{q}\left(\hat{\beta}_q - \overline{\beta}_q\right)' \widehat{\Omega}^{-1}_{\hat{\beta}_q}\left(\hat{\beta}_q - \overline{\beta}_q\right) \tag{2.51}$$

On rejette l'hypothèse nulle au risque α si $F_0 > F_\alpha(q, n-p-1)$.

Exemple 2.6 *Concentration*

On souhaite répondre à la troisième question à savoir : Est ce que la valeur de `03` est influencée par `Vx` et `T12`? Ce qui correspond à $\beta_1 = \beta_2 = 0$. L'hypothèse nulle peut s'écrire sous une forme matricielle :

$$\mathrm{H}_0 : R\beta = 0 \tag{2.52}$$

où β est le vecteur des coefficients et R la matrice des contraintes définie par :

$$R = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

En effet, en développent, nous avons :

$$\mathrm{H}_0 : \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \Longleftrightarrow \quad \mathrm{H}_0 : \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

```
# Test de nullité simultanée des paramètres
import numpy as np
# Matrice des contraintes
R = np.array([[0,1,0,0],[0,0,1,0]])
# Test statistique
ftest = model2.f_test(R)
res_ftest = pd.DataFrame({"valeur de F" : ftest.fvalue,"DDL num.":ftest.df_num,
                          "DDL denom" : ftest.df_denom,"Pr>F" : ftest.pvalue},
                          index = ["F test"])
```

Table 2.13 – Test de nullité simultanée de T2 et Vx - Test de Fisher

	valeur de F	DDL num.	DDL denom	Pr>F
F test	6.42	2	46	0.003

2.3.5.2 Test de Student de significativité d'un coefficient

Nous voulons tester $H_0 : \beta_j = 0$ contre $H_1 : \beta_1 = 0$, appelé test bilatéral de significativité de β_j. La statistique de test :

$$T_{cal} = \frac{\hat{\beta}_j}{\hat{\sigma}_j}$$

où $\hat{\sigma}_j$ est l'écart - type estimé de $\hat{\beta}_j$. Sous l'hypothèse nulle H_0, cette statistique suit une loi de Student à $(n-p-1)$ degrés de liberté. Nous rejetons H_0 si la statistique de test, T_{cal} est telle que :

$$|T_{cal}| > t_{n-p-1}^{1-\alpha/2}$$

où $t_{n-p-1}^{1-\alpha/2}$ est le quantile d'order $(1-\alpha/2)$ d'une loi de Student à $(n-p-1)$ degrés de liberté.

Pour notre exemple, cette situation correspond aux question 1) $(H_0 : \beta_2 = 0)$ et 2) $(H_0 : \beta_3 = 0)$.

Nous allons effectuer un test de significativité des paramètres.

```
#Test de significativité des paramètres
ttest = pd.concat([model2.tvalues,model2.pvalues],axis=1)
ttest.columns = ['t','P>|t|']
```

Table 2.14 – Test T-student de significativité des paramètres

	t	P>|t|
Intercept	6.213643	0.0000001
T12	2.643786	0.0111742
Vx	2.903293	0.0056541
Ne12	-4.764744	0.0000193

Pour tous les coefficients pris séparément, nous refusons au seuil $\alpha = 5\%$ l'hypothèse $H_0 : \beta_j = 0$. Tous les coeffcients sont significativement non nuls et il ne semble donc pas utile de supprimer une variable explicative.

2.3.5.3 Test de significativité globale du modèle

Si des connaissances a *priori* du phénomène assurent l'existence d'un terme constant dans la régression, alors pour tester l'influences des autres régresseurs (non constants) sur la réponde Y, on regarde si $\mathbb{E}[Y] = \beta_0$. En d'autres termes, on teste si tous les coeffcients sont nuls, excepté la constante.

Dans ce cas, les hypothèses sont :

$$\begin{cases} H_0 : SCE = 0 \\ H_1 : SCE \neq 0 \end{cases}$$

La statistique à utiliser est celle de Fisher et se calcule comme suit :

$$F_{cal} = \frac{\frac{SCE}{ddl_{SCE}}}{\frac{SCR}{ddl_{SCR}}} = \frac{\frac{SCE}{p}}{\frac{SCR}{n-p-1}} = \frac{\frac{R^2}{p}}{\frac{1-R^2}{n-p-1}} \tag{2.53}$$

On rejette l'hypothèse nulle au risque α si $F_{cal} > F_{\alpha}(p, n-p-1)$.

```
# Test de significativité globale
ftest = model2.mse_model/model2.mse_resid
print('f-statistic : %.2f'%(ftest))
```

```
## f-statistic : 32.87
```

```
# Test de Fisher
ftest = pd.DataFrame(np.transpose([model2.fvalue,model2.df_model,
                                   model2.df_resid,model2.f_pvalue])).T
ftest.columns = ["Valeur de F","DDL num.", "DDL denom.", "Pr>F"]
ftest.index = ['F-test']
```

Table 2.15 – Test F de Fisher

	Valeur de F	DDL num.	DDL denom.	Pr>F
F-test	32.86621	3	46	0

Nous refusons à nouveau H_0.

Remarque 2.3

Si nous comparons ce modèle au modèle vu en début du chapitre à l'aide d'un test F entre ces deux modèles emboîtés, nous avons :

```
# Test des modèles emboîtés
import statsmodels.api as sm
anova_test = sm.stats.anova_lm(model,model2,test="F",typ="I")
```

Table 2.16 – Test des modèles emboîtés

df_resid	ssr	df_diff	ss_diff	F	Pr(>F)
47	13299.399	0			
46	8904.622	1	4394.777	22.70279	1.93e-05

Nous retrouvons que le test F entre ces deux modèles est équivalent au test T de nullité du coefficient de la variable `Ne12` dans le modèle `model2` (les deux probabilités critiques valent $1.93\text{e}-05$).

En conclusion, il semble que les 3 variables `T12`, `Vx` et `Ne12` soient explicatives de l'ozone.

3

Multicolinéarité des variables explicatives

Sommaire

Dans ce chapitre, nous examinons comment gérer l'abondance de l'information. En effet, il est fréquent, lorsque l'économiste spécifie un modèle, qu'il hésite à intégrer telle ou telle variable explicative. La question essentielle qu'il se pose est la suivante : comment déterminer le mix optimal des variables explicatives ? Formulée en terme statistique, cette question se résume à trouver les variables explicatives qui maximisent leur coefficient de corrélation avec la série à expliquer, tout en étant les moins corrélées entre elles.

Dans un modèle de régression, lorsque deux variables explicatives ou d'avantage sont fortement corrélées, on est en présence d'un cas de colinéarité (ou de multicolinéarité). Il est alors difficiel sinon impossible d'isoler l'effet particulier de chacun de ces variables explicatives sur la variable expliquée. En pareille circonstance, les coefficients des moindres carrés ordinaires estimés peuvent être sans signification statistique (ou même être affectés d'un signe érroné) alors que le coefficient de détermination R^2 atteint une valeur très élevée ($R^2 > 90\%$). En effet, une multicolinéarité s'avère problématique, car elle peut augmenter la variance des coefficients de régression et les rendre instables et difficiles à interpréter. Les conséquences des coefficients instables peuvent être les suivantes :

— les coefficients peuvent sembler non siginificatifs, même lorsqu'une relation significative existe entre le prédicteur et la réponse ;
— les coeffcients des prédicteurs fortement corrélés varieront considérablement d'un échantillon à l'autre ;
— lorsque des termes d'un modèle sont fortement corrélés, la suppression de l'un de ces termes aura une incidence considérable sur les coefficients estimés des autres. Les coefficients des termes fortement corréls peuvent même présenter le mauvais signe.

La multicolinéarité n'a aucune incidence sur l'adéquation de l'ajustement, ni sur la qualité de la prévision. Cependant, les coefficients individuels associés à chaque variable explicative ne peuvent pas être interprétés de façon fiable.

Certaines méthodes permettent quelques fois de surmonter oud u moins de réduire

l'inconvénient de la multicolinéarité : la réunion des données ; l'utilisation d'une nouvelle information indépendante ; la transformation de la relation fonctionnelle ; l'abadon d'une des variables fortement corrélée.

3.1 Définition de la multicolinéarité

3.1.1 Type de multicolinéarité

Il existe deux types de multicolinéarité : la multicolinéarité stricte ou parfaite et la multicolinéarité approchée ou imparfaite. Dans les deux cas, l'hypothèse d'indépendance des variables explicatives est remise en question.

3.1.1.1 Multicolinéarité stricte

Au sens strict, on parle de multicolinéarité parfaite lorsqu'une des variables explicatives d'un modèle est une combinaison linéaire d'une ou plusieurs autres variables explicatives introduites dans le même modèle. De façon générale, elle se traduit par la relation déterministe du type :

$$\beta_i x_{i,t} + \beta_{j,t} x_{j,t} = \beta_k x_{k,t} \quad \text{avec} \quad \beta_k = \beta_i + \beta_j \tag{3.1}$$

Dans les faites, une multicolinéarité parfaite n'est quasiment jamais observée. En effet, c'est une situation extrêment rare qui est liée à une erreur de conception du modèle ou de définition des variables. Ainsi, en cas de multicolinéarité parfaite, la machine affiche « *error* » car la matrice $\left(X^{'}X\right)^{-1}$ n'existe pas car $(X^{'}X)$ n'est pas inversible. En effet, la matrice X aura au moins deux colonnes linéairement dépendantes et dont $(X^{'}X)$ aura au moins deux lignes et deux colonnes linéairement dépendantes et sera une matrice singulière c'est - à - dire $\det\left(X^{'}X\right) = 0$ et dans ce cas il est impossible de calculer l'inverse.

3.1.1.2 Multicolinéarité approchée

Dans un modèle linéaire ordinaire à n observations et p variables explicatives, une situation de multicolinéarité approchée apparaît lorsque certains vecteurs colonnes de X, la matrice $(n \times p)$ des variables exogènes, forment un système de vecteurs « presque » liés, et qu'alors la matrice carrée $\left(X^{'}X\right)$ a un certain nombre de valeurs propres très petites, qui pèsent numériquement sur son inversion.

Dans ce cas, la relation n'est plus déterministe mais plus large et s'écrit :

$$\beta_i x_{i,t} + \beta_{j,t} x_{j,t} = \beta_k x_{k,t} + \gamma \quad \text{avec} \quad \beta_k = \beta_i + \beta_j, \gamma \in \mathbb{R} \tag{3.2}$$

3.1.1.3 Le rôle des valeurs propres sur la matrice des corrélations

On rappelle que dans le cas d'une régression linéaire multiple ; la matrice de variances et covariances de $\widehat{\beta}$ est égale à $\mathrm{V}\left(\widehat{\beta}\right) = \sigma_\varepsilon^2 \left(X^{'}X\right)$. Si les prédicteurs sont très corréles entre eux alors $(X^{'}X)$ est mal conditionnée (déterminant proche de 0) et son inverse

aura des termes élevés. Dans ce cas, les paramètres du mdoèle seront estimés avec imprécision et les prédictions pourront être entachés d'erreurs considérables même si R^2 a une valeur élevée.

Si on suppose que les données sont centrées et réduites (ce qui ne change pas le R^2 ni les valeurs prévues), alors la matrice $(X^{'}X)$ est égale à

$$X^{'}X = nR \tag{3.3}$$

où R est la matrice des corrélations entre les prédicteurs.

Sous cette condition, on a donc :

$$\mathrm{V}\left(\widehat{\beta}\right) = \frac{\sigma_{\varepsilon}^2}{n}R^{-1} \quad \text{et} \quad \sigma_{\widehat{\beta}_j}^2 = \frac{\sigma_{\varepsilon}^2}{n}R_{jj}^{-1}$$

Posons $R = U\Lambda U^{'}$ où Δ est la matrice diagonale des valeurs propres et U la matrice des vecteurs propres de R. On a donc $R^{-1} : U\Delta^{-1}U^{'}$. On en déduit donc :

$$\sigma_{\widehat{\beta}_j}^2 = \frac{\sigma_{\varepsilon}^2}{n}\sum_{k=1}^{k=p}\frac{\left(v_{jk}\right)^2}{\lambda_k} \tag{3.4}$$

On voit donc que $\sigma_{\widehat{\beta}_j}^2$ dépend des inverses de valeurs propres de R : lorsqu'il y a forte colinéarité entre les prédicteurs les dernières valeurs propres sont proches de 0 d'où l'instabilité des β_j.

3.2 Mesures de la colinéarité

Il existe différentes mesures de la multicolinéarité.

3.2.1 Détection de la colinéarité

La colinéarité fausse l'inférence statistique, notamment parce qu'elle gonfle la variance estimée des coefficients. Il existe plusieurs techniques pour détecter la multicolinéarité entre les variables explicatives, dont le test de Klein et le test de Farrar-Glauber.

3.2.1.1 Test de Klein

Il ne s'agit pas d'un test à proprement parler mais plutôt d'un indicateur simple pour détecter rapidement les situations à problème. (*cf.* Bourbonnais et al. (2021)). La procédure du test de Klein and Nakamura (1962) consiste à comparer le coefficient de détermination R^2 calculé sur le modèle à p variables :

$$Y = X\beta + \varepsilon \tag{3.5}$$

et les coefficients de détermination (ou de corrélation linéaire de Pearson ρ_{x_i,x_j}^2) entre les variables explicatives considérées deux à deux pour $i \neq j$.

Ce test s'effectue en trois étapes :

1. Estimation du modèle (3.5) et calcul du R^2.

$$Y = X\hat{\beta} + \hat{\varepsilon} \quad \text{et} \quad R^2 = \frac{SCE}{SCT} = 1 - \frac{SCR}{SCT}$$

2. Calcul de la matrice des coeffcients de corrélation linéaire de Pearson entre les variables explicatives, prises deux à deux, soit :

$$\begin{pmatrix} 1 & \rho_{x_1x_2} & \cdots & \rho_{x_1x_p} \\ \rho_{x_2x_1} & 1 & \cdots & \rho_{x_2x_p} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{x_px_1} & \rho_{x_px_2} & \cdots & 1 \end{pmatrix}$$

3. Comparer, enfin, le R^2 de la régression aux différents coefficients de corrélation.

Il y a présomption de multicolinéarité si au moins un des $\rho_{x_ix_j}$ élevé au carré est supérieur au R^2.

Exemple 3.1 *Données CONSO*

Nous traitons les données automobiles. le fichier décrit $(n = 25)$ automobiles à partir de leur cylindrée, leur puissance, leur poids et leur consommation.

```
# Chargement des données
import pandas as pd
cars = pd.read_excel("./donnee/voitures.xlsx")
```

Table 3.1 – Données automobiles - 6 observations

modele	cylindree	puissance	poids	conso
Maserati Ghibli GT	2789	209	1485	14.5
Daihatsu Cuore	846	32	650	5.7
Toyota Corolla	1331	55	1010	7.1
Fort Escort 1.4i PT	1390	54	1110	8.6
Mazda Hachtback V	2497	122	1330	10.8
Volvo 960 Kombi aut	2473	125	1570	12.7
Toyota Previa salon	2438	97	1800	12.8
Renault Safrane 2.2. V	2165	101	1500	11.7
Honda Civic Joker 1.4	1396	66	1140	7.7
VW Golt 2.0 GTI	1984	85	1155	9.5
Suzuki Swift 1.0 GLS	993	39	790	5.8
Lancia K 3.0 LS	2958	150	1550	11.9
Mercedes S 600	5987	300	2250	18.7
Volvo 850 2.5	2435	106	1370	10.8
VW Polo 1.4 60	1390	44	955	6.5
Hyundai Sonata 3000	2972	107	1400	11.7
Opel Corsa 1.2i Eco	1195	33	895	6.8
Opel Astra 1.6i 16V	1597	74	1080	7.4
Peugeot 306 Xs 108	1761	74	1100	9.0
Mitsubishi Galant	1998	66	1300	7.6
Citroen ZX Volcane	1998	89	1140	8.8
Peugeot 806 2.0	1998	89	1560	10.8
Fiat Panda Mambo L	899	29	730	6.1
Seat Alhambra 2.0	1984	85	1635	11.6
Ford Fiesta 1.2 Zetec	1242	55	940	6.6

Nous souhaitons expliquer et prédire la consommation des automobiles (Y : conso) à partir de $(p = 3)$ caractéristiques (X_1 : cylindrée, X_2 : puissance, X_3 : poids). L'équation de régression s'écrit :

$$\texttt{cons} = \beta_0 + \beta_1 \texttt{cylindrée} + \beta_2 \texttt{puissance} + \beta_3 \texttt{poids} + \varepsilon \tag{3.6}$$

```
# Estimation des paramètres
import statsmodels.formula.api as smf
model = smf.ols("conso ~ cylindree + puissance + poids",data=cars).fit()
# Tableau des coefficients
tab_summary = model.summary2().tables[1]
```

Table 3.2 – Coefficients du modèle de régression

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	1.2998	0.6141	2.1167	0.0464	0.0228	2.5769
cylindree	-0.0005	0.0005	-0.9533	0.3513	-0.0015	0.0006
puissance	0.0303	0.0075	4.0516	0.0006	0.0147	0.0458
poids	0.0052	0.0008	6.4473	0.0000	0.0035	0.0069

Le modèle, avec un $R^2 = 0.9567$. est globalement significatif à 5% ($F = 154.68$, avec Prob(F-statistic) $= 1.78689 \times 10^{-14}$).

```
# Avertissement
print(model.summary2())
```

```
##                   Results: Ordinary least squares
## ================================================================
## Model:              OLS              Adj. R-squared:     0.951
## Dependent Variable: conso            AIC:                56.8006
## Date:               2023-09-21 20:22 BIC:                61.6761
## No. Observations:   25               Log-Likelihood:     -24.400
## Df Model:           3                F-statistic:        154.7
## Df Residuals:       21               Prob (F-statistic): 1.79e-14
## R-squared:          0.957            Scale:              0.49091
## ----------------------------------------------------------------
##                 Coef.   Std.Err.     t     P>|t|    [0.025  0.975]
## ----------------------------------------------------------------
## Intercept       1.2998    0.6141   2.1167  0.0464   0.0228  2.5769
## cylindree      -0.0005    0.0005  -0.9533  0.3513  -0.0015  0.0006
## puissance       0.0303    0.0075   4.0516  0.0006   0.0147  0.0458
## poids           0.0052    0.0008   6.4473  0.0000   0.0035  0.0069
## ----------------------------------------------------------------
## Omnibus:               0.799        Durbin-Watson:          2.419
## Prob(Omnibus):         0.671        Jarque-Bera (JB):       0.772
## Skew:                  -0.369       Prob(JB):               0.680
## Kurtosis:              2.558        Condition No.:          11415
## ================================================================
## Notes:
## [1] Standard Errors assume that the covariance matrix of the
## errors is correctly specified.
```

```
## [2] The condition number is large, 1.14e+04. This might indicate
## that there are strong multicollinearity or other numerical
## problems.
```

Un avertissement (warnings) nous alerte néanmoins. On suspecte un problème de colinéarité entre les exogènes. Nous isolons les exogènes dans une matrice.

```
# Matrice des exogènes
cars_exog = cars[["cylindree","puissance","poids"]]
mc = cars_exog.corr(method="pearson")
```

Table 3.3 – Matrice des corrélations des exogènes

	cylindree	puissance	poids
cylindree	1.0000	0.947	0.8748
puissance	0.9470	1.000	0.8280
poids	0.8748	0.828	1.0000

Analysons tout d'abord cette matrice des corrélations. Un règle simple de vérification est de comparer la valeur absolue des corrélations croisées entre exogènes avec 0.8 : $|\rho_{x_i x_j}| > 0.8$. On voit que la cylindrée et la puissance du moteur sont liées, on comprend très bien pourquoi. Elles sont également liées avec le poids des véhicules. Oui, les véhicules lours ont généralement besoin de plus de puissance.

Appliquons le test de Klein. Elle est plus intéressante car elle tient compte des caractéristiques numériques du modèle. Dans notre cas, $R^2 = 0.9567$.

```
# Règle de Klein
mc2 = mc.apply(lambda x : x**2, axis=0)
```

Table 3.4 – Matrice des corrélations des exogènes au carré

	cylindree	puissance	poids
cylindree	1.0000	0.8969	0.7653
puissance	0.8969	1.0000	0.6855
poids	0.7653	0.6855	1.0000

Aucune des valeurs n'excède le R^2, mais il y a des similarités inquiétantes néanmoins.

Remarque 3.1

Le test de Klein n'est pas un test statistique au sens test d'hypothèses, mais simplement un critère de présomption de multicolinéarité. C'est pourquoi il doit être complété par le test de Farrar et Glauber qui est bien un test statistique.

3.2.1.2 Test de Farrar - Glauber

Le test de Farrar and Glauber (1967) teste les hypothèses suivantes :

$$\begin{cases} H_0 : & \text{absence de multicolinéarité} \\ H_1 : & \text{présence de multicolinéarite} \end{cases}$$

Ce test est basé sur la statistique du χ^2, calculée à partir de l'échantillon comme suit :

$$FG = -\left[n - 1 - \frac{1}{6}(2K + 5)\right] \ln|R| \tag{3.7}$$

où n est la taille de l'échantillon ; K est le nombre de variables explicatives (terme constant inclus, $K = p + 1$) ; $|R|$ est le déterminant de la matrice des coefficients de corrélation linéaire de Pearson entre variables explicatives.

Sous l'hypothèse nulle, cette statistique suit une loi de χ^2 à $\frac{1}{2}K(K-1)$ degrés de liberté. Si $FG > \chi^2$ lu dans la table à $\frac{1}{2}K(K-1)$ degrés de liberté au seuil α choisi, alors on rejette l'hypothèse nulle ; il y a présomption de multicolinéarité.

```
# Statistique de Farrar - Glauber
import scipy.stats as st
n = cars.shape[0]
K = len(model.params)
FG = -(n-1-(1/6)*(2*K+5))*np.log(np.linalg.det(mc))
# Degré de liberté
ddl = (1/2)*K*(K-1)
# Valeur critique
res_fg =pd.DataFrame({"statistic" : FG, "DDL" : ddl,
                      "Critical value" : st.chi2.ppf(0.95,ddl)},index=["FG test"])
```

Table 3.5 – Test de Farrar - Glauber

	statistic	DDL	Critical value
FG test	81.24697	6	12.59159

Puisque $FG > \chi^2$, nous rejetons l'hypothèse H_0, il y a présomption de multicolinéarité.

Ces deux tests conduisent donc à des résultats différents, cependant le test de Farrar - Glauber, dont le fondement théorique est plus affirmé, semble devoir être privilégié.

3.2.1.3 Facteur d'inflation de la variance

Les deux tests précédents ne détectent que la colinéarité bivariée. Pour évaluer la multicolinéarité, il faudrait effectuer la régression de chaque exogène X_j avec les $(p-1)$ autres exogènes, puis étudier le coefficient de détermination R_j^2.

L'approche la plus classique consiste à examiner les facteurs d'inflation de la variance (FIV) ou *variance inflation factor* (VIF) en anglais. Les FIV permettent d'évaluer la liaison d'une exogène avec l'ensemble des autres variables. Ils estiment de combien la variance d'un coefficient est « augmentée » en raison d'une relation linéaire avec d'autres prédicteurs.

Définition 3.1 (Facteur d'inflation de la variance) *On appelle facteur d'inflation de la variance (VIF) la quantité*

$$V_j = \frac{1}{1 - R_j^2} \tag{3.8}$$

où R_j^2 est le coeffcient de détermination de la regréssion de X_j avec les $(p-1)$ autres variables explicatives.

On parle de *facteur d'inflation* car nous avons la relation suivante :

$$\mathrm{V}\left(\hat{\beta}_j\right) = \frac{\sigma_\varepsilon^2}{n} V_j$$

L'écart - type de l'estimation est multiplié par un facteur $\sqrt{V_j}$: plus V_j sera élevé, plus la variance $\mathrm{V}\left(\hat{\beta}_j\right)$ de l'estimation sera forte. L'estimation $\hat{\beta}_j$ sera donc très instable, il aura moins de chances d'être significatif dans le test de nullité du coefficient dans la régression.

A partir de quelle valeur de V_j doit - on s'inquiéter ? Si les variables étaient 2 à 2 indépendantes , $V_j = 1$ et $\mathrm{V}\left(\hat{\beta}_j\right) = \frac{\sigma_\varepsilon^2}{n}$. Nous pourrions obtenir les ceofficients de la régression multiple à partir de p régressions simples. Une règle usuelle de détection de la colinéarité est de prendre un seuil où l'on multiplierait d'un facteur de 2 l'écart - type de l'estimation. On décide qu'il y a un problème de colinéarité lorsque

$$V_j \geq 4 \tag{3.9}$$

Certains utilisent une règle moins contraignante et préfèrent les seuils 5 ou même 10 c'est - à - dire la multicolinéarité n'est signalée que si elle est vraiment élevée.

Remarque 3.2

La quantité $1-R_j^2$ est appelée *tolérance* : plus elle est faible, plus la variable X_j souffre de colinéarité. En dérivant la règle de détection du VIF, on s'inquiéterait dès que la tolérance est inférieure à 0.25.

```
# Variance Inflation Factor
def create_formula(y,x):
  return y + ' ~ ' + ' + '.join(x)

from plydata import *
vif = pd.DataFrame(index=cars_exog.columns,columns=["Rsquared"]).astype("float")
for name in cars_exog.columns:
  X, Y = cars_exog.drop(columns=name), cars_exog[name]
  formule = create_formula(Y.name,X.columns)
  vif.loc[name,:] = smf.ols(formule,data=cars_exog).fit().rsquared
# Calcul du VIF
vif = vif >> mutate(VIF="1/(1-Rsquared)")
```

Table 3.6 – Variance Inflation Factor (par régression)

	Rsquared	VIF
cylindree	0.9230330	12.992577
puissance	0.8968698	9.696485
poids	0.7653084	4.260911

Calculer p régressions croisées, chaque variable X_j contre les $(p-1)$ autres pour obtenir les R_j^2 et donc V_j, serait vite fastidieux. Nous pouvons profiter des calculs existants pour produire le VIF. En effet, si C est la matrice des corrélations entre les exogènes de taille $(p \times p)$, la quantité V_j peut être lue à la coordonnée de la diagonale principale de la matrice C^{-1}.

```
# Approche 1 - VIF
VIF = pd.DataFrame({"VIF" : np.diag(np.linalg.inv(mc))})
VIF.index = cars_exog.columns
```

Table 3.7 – Variance Inflation Factor (par inversion de la matrice des corrélations)

	VIF
cylindree	12.992577
puissance	9.696485
poids	4.260911

On peut vérifier nos résultats en utilisant la fonction variance_inflation_factor de Statsmodels.

```
from patsy import dmatrices
from statsmodels.stats.outliers_influence import variance_inflation_factor
features = "+".join(cars_exog.columns)
# get y and X dataframes based on this regression:
y, X = dmatrices("conso ~" + features,cars, return_type='dataframe')
# VIF - Statsmodels
VIF             = pd.DataFrame()
VIF['feature']  = X.columns
VIF['VIF']      = [variance_inflation_factor(X.values,i) for i in range(X.shape[1])]
```

Table 3.8 – Variance Inflation Factor (Statsmodels)

feature	VIF
Intercept	19.203540
cylindree	12.992577
puissance	9.696485
poids	4.260911

Une variable pose problème dès lors que $VIF > 4$ (en étant très restrictif). Nous avons manifestement des problèmes dans nos données.

Proposition 3.1 *Cohérence des signes*

Il existe une autre approche très simple pour détecter la colinéarité, comparer les signes des coeffcients de la régression avec le signe des corrélations simples entre les exogènes et l'endogène. La procédure est la suivante :

1. Calculer normalement la régression linéaire multiple $Y = X\beta + \varepsilon$ et stocker les signes des coeffcients estimés $\widehat{\beta}$.
2. Calculer les corrélations croisées entre chaque variable exogène X_j et l'endogène y : ρ_{y,x_j}.

Il y a présomption de colinéarité s'il existe des situations où sign $\left(\hat{\beta}_j\right) \neq$ sign $\left(\rho_{y,x_j}\right)$. En effet, cela indique que les autres variables perturbent la relation entre Y et X_j.

Exemple 3.2 *Application aux données CONSO*

Nous calculons les corrélations simples entre chaque exogène et l'endogène. Nous comparons les résultats avec les coeffcients de la régression.

```
# Corrélation entre exogène et endogène
res = pd.DataFrame()
res["coef"] = model.params[1:]
res["corr"] = [np.corrcoef(cars[x],cars["conso"],
                           rowvar=False)[0,1] for x in cars_exog.columns]
```

Table 3.9 – Comparison des corrélations individuelles et des coefficients - Données CONSO

	coef	corr
cylindree	-0.0004740	0.9151557
puissance	0.0302691	0.9254514
poids	0.0052011	0.9421241

Il y a un conflit pour la variable cylindree.

3.2.2 Traitement de la multicolinéarité

Les méthodes de sélection de variables sont une des réponses possibles au problème de la multicolinéarité, mais elles peuvent conduire à l'élimination de variables significativement liées à y. Il est alors difficile de proposer à l'utilisateur un modèle qui ne tient pas compte de variances pourtant influentes et ne permet pas de quantifier l'effet de leurs variations sur la réponse y (*cf.* Saporta (2006)).

D'autres techniques sont préconisées telles que : la régression sur composantes principales ; la régression PLS et etc.

4

Violation des hypothèses sur les erreurs

Sommaire

Ce chapitre est consacré aux problèmes particuliers liés au non - respect des hypothèses. Nous nous attachons particulièrement à deux formes classiques : l'autocorrélation des erreurs et l'hétéroscédasticité. L'étude de ces deux phénomènes nous permet de définir un nouvel estimateur, celui des moindres carrés généralisés (GLS, *Generalized Least Squares*), utilisé lorsque la matrice des variances et covariances de l'erreur ne répond plus aux hypothèses classiques. Pour chacun de ces deux cas, nous présentons les méthodes d'investigation : tests de détection, conséquences et procédures d'estimation.

4.1 Autocorrélation des erreurs

Dans le chapitre **??**, l'hypothèse 4 les termes d'erreur sont indépendants les uns des autres dans le temps si et seulement si $\mathbb{E}\left(\varepsilon_t, \varepsilon_{t'}\right) = 0, \quad \forall t = t^{'}$. Dans le cas contraire, c'est-à-dire $\mathbb{E}(\varepsilon_t, \varepsilon_{t'}) \neq 0$, la méthode des moindres carrés ordinaires donne toujours des estimateurs sans biais, mais la variance n'est plus minimale. On parle alors **d'autocorrélation des termes de l'erreur**. Dans ce cas, la matrice des variances et covariances des paramètres s'écrit :

$$\mathrm{V}\left(\widehat{\beta}\right) = \left(X^{'}X\right)^{-1} X^{'}\mathbb{E}\left(\varepsilon\varepsilon^{'}\right) X \left(X^{'}X\right)^{-1} = \left(X^{'}X\right)^{-1}\left(X^{'}\mathrm{V}(\varepsilon)X\right)\left(X^{'}X\right)^{-1} \tag{4.1}$$

$\widehat{\beta}$ est ici un estimateur dont la première diagonale de la matrice des variances et covariances est supérieure à celle de $\sigma_\varepsilon^2\left(X^{'}X\right)^{-1}$ du modèle classique. Le problème est donc de trouver un estimateur des paramètres du modèle ayant les mêmes propriétés que l'estimateur des moindres carrés ordinaires : cet estimateur est celui des moindres carrés généralisés (MCG).

4.1.1 Estimateur des Moindres Carrés Généralisés

Partant du modèle linéaire général

$$Y = X\beta + \varepsilon \tag{4.2}$$

dans lequel la matrice des variances et covariances des termes de l'erreur est $\mathbb{E}\left(\varepsilon_t \varepsilon_t^{'}\right) = \Omega_\varepsilon \neq \sigma_\varepsilon^2 I$ où Ω_ε est de dimension (n, n), on démontre que l'estimateur $\hat{\beta}$ qui a les mêmes propriétés que l'estimateur des moindres carrés ordinaires (sans biais, fonction de Y et à variance minimale) est donné par :

$$\hat{\beta} = \left(X^{'}\Omega_\varepsilon^{-1}X\right)^{-1}\left(X^{'}\Omega_\varepsilon^{-1}Y\right) \tag{4.3}$$

avec :

$$\mathrm{V}\left(\hat{\beta}\right) = \left(X^{-1}\Omega_\varepsilon^{-1}X\right)^{-1} \tag{4.4}$$

Cet estimateur est appelé estimateur des Moindres Carrés Généralisés (MCG) ou encore estimateur de Aitken.

Remarque 4.1

— Lorsque les hypothèses classiques sont satisfaites, on retrouve l'estimateur des moindres carrés ordinaires :

$$\hat{\beta} = \left(X^{'}\Omega_\varepsilon^{-1}X\right)^{-1}\left(X^{'}\Omega_\varepsilon^{-1}Y\right) = \left(X^{'}\left(\frac{1}{\sigma_\varepsilon^2}I\right)X\right)^{-1}\left(X^{'}\left(\frac{1}{\sigma_\varepsilon^2}I\right)Y\right) = \left(X^{'}X\right)^{-1}\left(X^{'}Y\right)$$

— L'utilisation de l'estimateur des moindres carrés généralisés nécessite la connaissance de Ω_ε. Or dans la pratique cette matrice est inconnue. Les formules ci-dessus ne sont donc pas utilisables, sauf si l'on connait une approximation de Ω_ε. Il est ainsi nécessaire de développer des procédures d'estimation opérationnelles.

4.1.2 Causes de l'autocorrélation des erreurs

Lorsque nous travaillons avec des données longitudinales (ou en séries temporelles), la date définit naturellement l'ordonnancement des observations. Il est important de vérifier que les résidus sont produits de manière totalement aléatoire. Si l'on conclut au rejet de cette hypothèse, les résidus sont produits par un processus quelconque, l'hypothèse d'indéendance des erreurs est rejetée, la méthode des moindres caryrés ordinaires n'est plus BLUE : elle est certe non - biaisée, mais n'est plus à variance minimale, et la matrice de variance et covariance n'est plus estimée de manière convergente, les tests de significativité ne sont plus opérants.

La situation ci-dessus est celle dans laquelle les termes de l'erreur ne sont pas indépendants au cours du temps. On va introduire une hypothèse selon laquelle « le terme d'erreur dépend linéairement de sa propre valeur passée ». Cette hypothèse se traduit par l'expression :

$$\varepsilon_t = \rho\varepsilon_{t-1} + \mu_t \tag{4.5}$$

Dans cette expression, le coefficient ρ est le coefficient d'autocorrélation et μ_t est le nouveau terme d'erreur qui satisfait aux propriétés suivantes :

$$\begin{cases} \mathbb{E}(\mu_t) = 0 \\ \mathrm{V}(\mu_t) = \sigma_\mu^2 \quad \forall t \\ \mu_t \sim \mathcal{N}(0, \sigma_\mu^2) \end{cases}$$

Le terme d'erreur ε_t de l'équation initiale $\left(y_t = \beta_0 + \sum_{j=1}^{j=p} \beta_j x_{j,t} + \varepsilon_t \right)$ comprend dont une partie systématique qui est $\rho\varepsilon_{t-1}$ et une partie purement aléatoire μ_t.

La relation (4.5) est possible mais n'est pas forcement la seule car :

— Elle est linéaire
— Elle est une autocorrélation de premier order, c'est - à - dire elle ne dépend que de la période antérieure.

L'analyse graphique du résidu peut donner une première information sur l'existence ou non d'éventuelle autocorrélation : plus on observera des régularités au niveau des signes d'erreur, plus on constatera une autocorrélation. Il existe deux types d'autocorrélation : lorsqu'on a une sucession positive ou négative des résidus, cette reproduction entraine une autocorrélation positive $(\rho < 0)$ (*cf.* Figure 4.1).

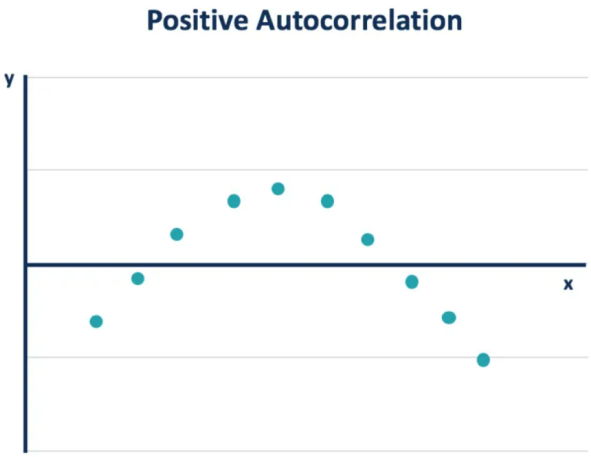


Figure 4.1 – Autocorrélation positive

Au contraire, si les résidus sont alternés ; c'est - à - dire qu'il se produit des fluctuations positives et négatives, on a une autocorrélation négative $(\rho > 0)$ (*cf.* Figure 4.2).

L'autocorrélation est un problème statistique qui peut être observé pour des raisons ci-après (*cf.* Bourbonnais et al. (2021)) :

— Mauvaise spécification du modèle (c'est la raison la plus importante) : les relations entre la variable endogène et les variables explicatives ne sont pas linéaires et s'expriment sous une autre forme que celle du modèle estimé (logarithmes, différences premières,...) ;

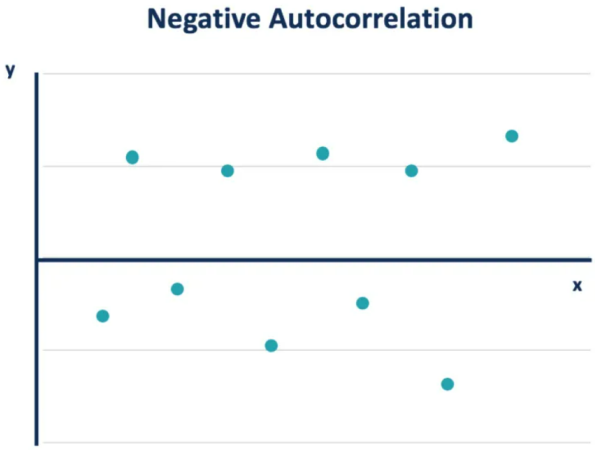


Figure 4.2 – Autocorrélation negative

— L'absence d'une variable explicative pertinente (importante) dont l'explication résiduelle permettrait de « blanchir » les erreurs ;
— Un lissage par moyenne mobile ou une interpolation des données crée une autocorrélation artificielle des erreurs due à l'usage de ces deux opérateurs.

Exemple 4.1 *Concentration en Ozone*

Revenons à l'exemple de la prévision des pics d'ozone. Nous expliquons le pic d'ozone O3 par 6 variables : la teneur maximum en ozone la veille (O3v), la température prévue par Méto France à 6h (T6), à midi (T12), une variable synthétique (la projection du vent sur l'axe est - ouest notée Vx) et enfin les nébulosités prévues à midi (Ne12) et à 15h (Ne15). Nous avons pour ce travail $n = 1014$ observations.

Notre modèle s'écrit :

$$O3 = \tag{4.6}$$

```
# Chargement des données
import pandas as pd
ozone = pd.read_csv('./donnee/ozone_long.txt',sep=';')
```

Table 4.1 – 50 données de température à 12 h et teneur en ozone complétées

O3	T6	T12	Ne12	Ne15	Vx	O3v
47.6	10.1	13.3	8	8	-3.4641	62.2
56.2	9.5	13.8	7	0	0.0000	47.6
61.8	3.6	16.8	2	2	-0.3420	56.2
50.8	9.5	11.4	7	7	-0.5209	61.8
59.8	9.8	13.8	8	8	-0.9848	50.8
53.4	8.9	12.6	8	8	-1.8794	59.8

Notre modèle s'écrit :

$$\texttt{O3} = \beta_0 + \beta_1 \texttt{T6} + \beta_2 \texttt{T12} + \beta_3 \texttt{Ne12} + \beta_4 \texttt{Ne15} + \beta_5 \texttt{Vx} + \beta_6 \texttt{O3v} \tag{4.7}$$

```
# Estimation des paramètres
import statsmodels.formula.api as smf
model = smf.ols(formula='O3~T6+T12+Ne12+Ne15+Vx+O3v',data=ozone).fit()
coef_table = model.summary2().tables[1]
```

Table 4.2 – Coefficients du modèle de régression

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	34.0895	3.1749	10.7371	0.0000	27.8593	40.3198
T6	-1.5935	0.1842	-8.6491	0.0000	-1.9550	-1.2319
T12	2.0775	0.1717	12.1011	0.0000	1.7406	2.4144
Ne12	-0.9538	0.3234	-2.9497	0.0033	-1.5884	-0.3193
Ne15	-0.7738	0.2780	-2.7837	0.0055	-1.3193	-0.2283
Vx	0.7108	0.1398	5.0842	0.0000	0.4364	0.9851
O3v	0.4548	0.0205	22.2005	0.0000	0.4146	0.4949

Nous représentons les résidus en fonction du numéro d'observation, numéros qui sont dans l'ordre chronologique.

```
from plotnine import *
resid = pd.DataFrame(model.resid,columns=["resid"],index=ozone.index)
print((ggplot(resid,aes(x=resid.index,y="resid"))+geom_point()+
       labs(x="Index",y="Résidus")))
```

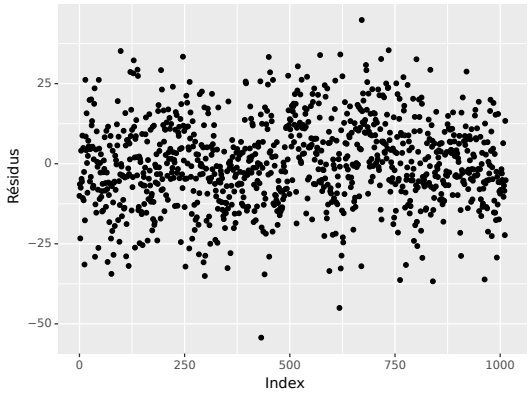


Figure 4.3 – Résidus du modèle (4.7)

4.1.3 Détection de l'autocorrélation des erreurs

La détection de l'autocorrélation des résidus peut s'effectuer visuellement à l'aide du graphique des résidus (*cf.* Figure 4.3). Elle peut également s'appuyer sur des techniques statistiques. Il existe un grand nombre de tests d'absence d'autocorrélation. Les plus connus étant ceux de Box and Pierce (1970), Ljung and Box (1978) et de Durbin and Watson (1950).

4.1.3.1 Tests de Box-Pierce et de Ljung - Box

Les tests de Box - Pierce et de Ljung - Box sont appelés des **tests de Portmanteau**. 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 :

$$H_0 : \hat{\rho}_{\hat{\varepsilon}}(1) = \cdots = \hat{\rho}_{\hat{\varepsilon}}(H) = 0 \tag{4.8}$$

La statistique de test de Box and Pierce (1970) s'écrit :

$$Q_{BP}(H) = T \sum_{j=1}^{j=H} \hat{\rho}^2_{\hat{\varepsilon}}(j) \tag{4.9}$$

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.

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 :

$$Q_{JB}(H) = T(T+2) \sum_{j=1}^{j=H} \frac{\hat{\rho}^2_{\hat{\varepsilon}}(j)}{T-j} \tag{4.10}$$

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 4.2 *Statistique Box - Pierce et de Ljung - Box*

Sous Python, la fonction acorr_ljungbox teste l'autocorrélation en utilisant la méthode Ljung - Box. Pour obtenir également les statistiques de Box - Pierce, il faut fixer le paramète `boxpierce` à `True`.

```
# Test de Box - Pierce
from statsmodels.stats.diagnostic import acorr_ljungbox
bp_lb = pd.concat((acorr_ljungbox(model.resid,lags=[x],boxpierce=True,
                    return_df=True) for x in np.arange(1,11)),axis=0)
```

Table 4.3 – Statistique de Box - Pierce et de Ljung - Box

	Ljung - Box		Box - Pierce	
Lag	statistic	pvalue	statistic	pvalue
1	7.766519	0.0053224	7.743586	0.0053904
2	10.779372	0.0045634	10.744578	0.0046435
3	28.231966	0.0000032	28.111283	0.0000034
4	43.602831	0.0000000	43.391375	0.0000000
5	60.570435	0.0000000	60.242076	0.0000000
6	75.926351	0.0000000	75.477080	0.0000000
7	87.835207	0.0000000	87.280444	0.0000000
8	103.061545	0.0000000	102.356916	0.0000000
9	110.490176	0.0000000	109.705119	0.0000000
10	119.321433	0.0000000	118.432070	0.0000000

Toutes les p-values étant inférieures au seuil critique de 5%, on conclut que les résidus estimés sont autocorrélés au sens du test de Box - Pierce.

Remarque 4.2

Dans le cas d'un résidu hétéroscédastique (*cf.* section 4.2), 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 $\widehat{\rho}_{\widehat{\varepsilon}}(i)$ par la variance asymptotique $\widehat{\sigma}^2_{\widehat{\rho}_{\widehat{\varepsilon}}(i)}$ calculée dans le cas de l'hétéroscédasticité par Lo and MacKinlay (1989) selon la formule :

$$\widehat{\sigma}^2_{\widehat{\rho}_{\widehat{\varepsilon}}(i)} = \frac{\sum_{t=i+1}^{t=T} \widehat{\varepsilon}^2_t \widehat{\varepsilon}^2_{t-i}}{\sum_{t=i+1}^{t=T} \widehat{\varepsilon}^2_t} \tag{4.11}$$

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 4.3

Plusieurs versions du test de portmanteau ont été proposés 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 :

$$Q_{LM}(H) = \frac{H(H+1)}{2T} + Q_{BP}(H) = \frac{H(H+1)}{2T} + T\sum_{j=1}^{j=H} \widehat{\rho}^2_{\widehat{\varepsilon}}(j) \tag{4.12}$$

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 :

$$Q_M(H) = T(T+2)\sum_{j=1}^{j=H} \frac{\widehat{\pi}^2_{\widehat{\varepsilon}}(j)}{T-j} \tag{4.13}$$

avec $\widehat{\pi}(j) = \widehat{\phi}_{j,j}$. 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 :

$$D = T\left(1 - \left|\widehat{R}_H\right|^{1/H}\right) \tag{4.14}$$

où

$$\widehat{R}_H = \begin{pmatrix} 1 & \hat{\rho}_{\hat{\varepsilon}_t}(1) & \cdots & \hat{\rho}_{\hat{\varepsilon}_t}(H) \\ \hat{\rho}_{\hat{\varepsilon}_t}(1) & 1 & \cdots & \hat{\rho}_{\hat{\varepsilon}_t}(H-1) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\rho}_{\hat{\varepsilon}_t}(H) & \hat{\rho}_{\hat{\varepsilon}_t}(H-1) & \cdots & 1 \end{pmatrix} \tag{4.15}$$

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 asymtotique du test développé par Peña and Rodriguez (2002) est très lente (*cf.* Lin and McLeod (2006)).

Fisher (2011) a proposé de nouvelles versions pondérées du test de Box and Pierce (1970) et Monti (1994), les statistiques de test sont les suivantes :

$$Q_{WL}(H) = T(T+2)\sum_{j=1}^{j=H} \frac{H-j+1}{H(T-j)}\hat{\rho}^2_{\hat{\varepsilon}}(j) \tag{4.16}$$

et

$$Q_{WM}(H) = T(T+2)\sum_{j=1}^{j=H} \frac{H-j+1}{H(T-j)}\hat{\pi}^2_{\hat{\varepsilon}}(j) \tag{4.17}$$

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.

4.1.3.2 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 $t-2, t-3, \ldots$. On considère le processus suivant décrivant une autocorrélation à l'ordre 1 des résidus :

$$\hat{\varepsilon}_t = \rho\hat{\varepsilon}_{t-1} + \nu_t \tag{4.18}$$

ν_t est un bruit blanc et $\hat{\varepsilon}_t$ désigne les résidus estimés. Les hypothèses du test sont :

$$\begin{cases} H_0 : \rho = 0 & \text{(absence d'autocorrélation à l'ordre 1 des résidus)} \\ H_1 : \rho \neq 0 & \text{(présence d'autocorrélation à l'ordre 1 des résidus)} \end{cases} \tag{4.19}$$

La statistique de Durbin-Watson, noté DW, est donnée par :

$$DW = \frac{\sum_{t=2}^{t=T}(\hat{\varepsilon}_t - \hat{\varepsilon}_{t-1})^2}{\sum_{t=1}^{t=T}\hat{\varepsilon}_t^2} \tag{4.20}$$

On peut réécrire cette statistique de Burbin - Watson comme suit :

$$DW = \frac{\sum_{t=2}^{t=T}(\hat{\varepsilon}_t - \hat{\varepsilon}_{t-1})^2}{\sum_{t=1}^{t=T}\hat{\varepsilon}_t^2} = 2\,(1-\hat{\rho}) + \frac{\hat{\varepsilon}_1^2 + \hat{\varepsilon}_T^2}{\sum_{t=1}^{t=T}\hat{\varepsilon}_t^2} \tag{4.21}$$

avec $\hat{\rho} = \dfrac{\sum_{t=2}^{t=T}\hat{\varepsilon}_t\hat{\varepsilon}_{t-1}}{\sum_{t=1}^{t=T}\hat{\varepsilon}_t^2}$, l'estimateur du coeffcient d'autocorrélation d'ordre 1.

Cette statistique DW varie en 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. 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.

Exemple 4.3 *Test de Durbin - Watson sur les résidus*

On demande de calculer la statistique de Durbin - Watson.

```
# Test de Durbin - Watson
from statsmodels.stats.stattools import durbin_watson
dw_res = pd.DataFrame(columns=["DW","rho"],index=["DW test"]).astype("float")
dw = durbin_watson(model.resid)
dw_res["DW"] = dw
dw_res["rho"] = 1 - (dw/2)
```

Table 4.4 – Statistique de Durbin - Watson

	DW	rho
DW test	1.824528	0.0877357

Le test de Durbin et Watson est un test pésomptif d'indépendance des erreurs du fait qu'il utilise les résidus ; de plus, il ne teste qu'une autocorrélation d'ordre 1. Son

utilisation est encadrée par certaines conditions : la régression doit compoter un terme constante ; les variables explicatives sont certaines (non stochastiques), en particulier elles ne doivent pas comporter l'endogène retardée.

Remarque 4.4

Il est important de noter que le test de Durbin et Watson 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 :

$$h = \hat{\rho}\sqrt{\frac{T}{1 - T \times \hat{\sigma}^2_{\phi_1}}} \tag{4.22}$$

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 $T \times \hat{\sigma}^2_{\phi_1} < 1$.

4.1.3.3 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 :

$$m = \frac{1}{T}\left(T(T+1) - \sum_{i=1}^{i=3} T_i^2\right) \tag{4.23}$$

où T_i est le nombre de runs respectivement positifs, négatifs ou nuls $(i = 1, 2, 3)$.

La variance de m est :

$$\sigma^2_m = \frac{1}{T^2(T-1)}\left\{\left(\sum_{i=1}^{i=3} T_i^2\right)\left(\sum_{i=1}^{i=3} T_i^2 + T(T+1)\right) - 2T\sum_{i=1}^{i=3} T_i^3 - T^3\right\} \tag{4.24}$$

Pour T grand la distribution est considérée comme normale et on démontre que :

$$Z = \frac{R + 0.5 - m}{\sigma_m} \longrightarrow \mathcal{N}(0;1) \tag{4.25}$$

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 4.4 *Application du runs test*

On demande d'appliquer le runs test sur les résidus du estimé. 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 = pd.DataFrame(columns =["z-stat","p-value"],index = ["Run test"])
rtest.loc["Run test",:] = runs_test(model.resid)
rtest.iloc[:,1] = scientific_format(digits=3)(rtest.iloc[:,1])
```

Table 4.5 – Runs test sur les résidus

	z-stat	p-value
Run test	-3.440196	5.813e-04

Nous avons une p-value qui est inférieure au seuil critique de 5%. Pour ce modèle, on rejette l'hypothèse nulle d'indépendance des valeurs résiduelles : les résidus sont autocorrelés.

4.1.3.4 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 :

$$VN = \frac{T}{T-1} \times \frac{\sum_{t=1}^{t=T-1} \left(\hat{\varepsilon}_t - \hat{\varepsilon}_{t+1}\right)^2}{\sum_{t=1}^{t=T} \left(\hat{\varepsilon}_t - \bar{\hat{\varepsilon}}\right)^2} \tag{4.26}$$

Sous H_0, la statistique $z_1 = (VN - \mu)/\sigma$ suit asymptotiquement une loi normale standard avec :

$$\mu = \frac{2T}{T-1} \quad \text{et} \quad \sigma^2 = 4\frac{T^2(T-2)}{(T^2-1)(T-1)^2} \tag{4.27}$$

Dans le cas biaisé (original), la statistique du test est :

$$VN = \frac{\sum_{t=1}^{t=T-1} \left(\hat{\varepsilon}_t - \hat{\varepsilon}_{t+1}\right)^2}{\sum_{t=1}^{t=T} \left(\hat{\varepsilon}_t - \bar{\hat{\varepsilon}}\right)^2} \tag{4.28}$$

Sous H_0, la statistique $z_2 = (VN-2)/\sigma$ suit asymptotiquement une loi normale standard avec :

$$\sigma^2 = 4\frac{(T-2)}{T^2-1} \tag{4.29}$$

On rejette l'hypothèse nulle d'indépendance si la pvalue est inférieure à la valeur critique α.

Exemple 4.5 *Application du test de Von - Neumann sur la série des résidus*

On applique le test d'indépendence de Von - Neumann sur les séries des résidus. 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
import scipy.stats as st
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 = pd.DataFrame(columns=["VN","z","pvalue"],index=["VN test"])
vntest.loc["VN test",:] = VonNeumannTest(model.resid)
vntest.iloc[:,2] = scientific_format(digits=3)(vntest.iloc[:,2])
```

Table 4.6 – Test de Von - Neumann

	VN	z	pvalue
VN test	1.82633	-2.79656	5.165e-03

Au seuil de 5%, la pvalue est inférieure à cette valeur critique, par conséquent, on rejette l'hypothèse nulle d'indépendance des valeurs des résidus.

4.1.3.5 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 :

$$RVN = \frac{\sum_{i=1}^{i=T-1} (R_i - R_{i+1})^2}{\sum_{j=1}^{j=T} (R_j - \mu_R)^2} \tag{4.30}$$

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 4.7 – 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

$$\sigma^2 = \frac{4}{5} \times \frac{(T-2)(5T^2 - 2T - 9)}{T(T^2-1)(T-1)} \tag{4.31}$$

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

$$p = \frac{5}{2} \times \frac{T(T^2-1)(T-1)}{(T-2)(5T^2-2T-9)} - \frac{1}{2} \tag{4.32}$$

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 4.6 *Application du test de rang de Von - Neumann*

Nous implémentons ici une fonction `VonNeumannRatioTest` identique à la fonction BartelsRankTest 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 = pd.DataFrame(columns=["RVN","z","pvalue"],index=["RVN test"])
vnrtest.loc["RVN test",:] = VonNeumannRatioTest(model.resid,methode="normal")
vnrtest.iloc[:,2] = scientific_format(digits=3)(vnrtest.iloc[:,2])
```

Table 4.8 – Test de rang de Von - Neumann

	RVN	z	pvalue
RVN test	1.756692	-3.876546	1.059e-04

Au seuil de 5%, la pvalue est inférieure à cette valeur critique, par conséquent, on rejette l'hypothèse nulle d'indépendance des valeurs des résidus.

4.1.3.6 Test de Breusch - Godfrey

Ce test, fondé sur un test de Fisher de nullité de coefficients ou de Multiplicateur de Lagrange (« *LM Test*), permet de tester une autocorrélation d'un ordre supérieur à 1 et reste valide en présence de la variable dépendante décalée en tant que variable explicative. L'idée générale de ce test réside dans la recherche d'une relation significative entre le résidu et ce même résidu décalé.

Une autocorrélation des erreurs d'un ordre p s'écrit :

$$\varepsilon_t = \rho_1 \varepsilon_{t-1} + \rho_2 \varepsilon_{t-2} + \cdots + \rho_p \varepsilon_{t-p} + \mu_t \tag{4.33}$$

Soit le modèle de régression multiple à erreurs autocorrélés d'ordre p :

$$y_t = \beta_0 + \beta_1 x_{1,t} + \beta_2 x_{2,t} + \cdots + \beta_p x_{p,t} + \rho_1 \varepsilon_{t-1} + \rho_2 \varepsilon_{t-2} + \cdots + \rho_p \varepsilon_{t-p} + \mu_t \quad (4.34)$$

Ce test est mené en trois étapes :

1. Estimation par la méthode des moindres carrés ordinaires du modèle linéaire et calcul du résidu $\hat{\varepsilon}_t$, puisque les erreurs sont inconnues, le test porte sur les résidus.
2. Estimation par les moindres carrés ordinaires de l'équation intermédiaire :

$$\hat{\varepsilon}_t + \beta_0 + \beta_1 x_{1,t} + \beta_2 x_{2,t} + \cdots + \beta_p x_{p,t} + \rho_1 \hat{\varepsilon}_{t-1} + \rho_2 \hat{\varepsilon}_{t-2} + \cdots + \rho_p \hat{\varepsilon}_{t-p} + \mu_t \quad (4.35)$$

   Soit n le nombre d'observations disponibles (attention chaque décalage entraine la perte d'une observation) pour estimer les paramètres du modèles et R^2 le coeffcient de détermination. Certains auteurs préconisent, afin de ne pas perdre d'observations, de mettre 0 les premières valeurs du résidu décalé. La différence n'est perceptible que pour des petits échantillons.
3. Test d'hypothèses sur l'équation intermédiaire.

   L'hypothèse nulle d'absence d'autocorrélation des erreurs est :

$$H0 : \rho_1 = \rho_2 = \cdots = \rho_p = 0$$

   Si on refuse l'hypothèse nulle, alors il existe un risque d'autocorrélation des erreurs d'ordre p.

Pour mener ce test, nous avons deux possibilités :

— Soit effecture un test de Fisher classique de nullité des coefficients ρ_i
— Soit recourir à la statistique LM qui est distribuée comme un χ^2 à p degrés de liberté. Si

$$n \times R^2 > \chi^2(p)$$

lu dans la tableau au seuil α, on rejette l'hypothèse d'indépendance des erreurs.

Exemple 4.7 *Test de Breusch - Godfrey*

Sous Python, la fonction acorr_breusch_godfrey teste l'autocorrélation des erreurs à partir du test de Breusch - Godfrey.

```
# Test de Breusch - Godfrey
from statsmodels.stats.diagnostic import acorr_breusch_godfrey
bg=pd.DataFrame(np.zeros((10,4)),columns=["LM","LM-pvalue","F-stat","F-pvalue"])
for i in np.arange(10):
  bg.iloc[i,:] = acorr_breusch_godfrey(model,nlags=i+1)
bg.insert(0,"lag",np.arange(1,11))
```

Au seuil de 5%, toutes les pvalues sont inférieures au seuil, on rejette l'hypothèse d'indépendance des valeurs résiduelles.

Table 4.9 – Test de Breusch - Godfrey

Lag	statistic	pvalue	statistic	pvalue
1	13.69878	0.0002146	13.776822	0.0002171
2	22.42523	0.0000135	11.364424	0.0000132
3	46.63450	0.0000000	16.133520	0.0000000
4	59.26229	0.0000000	15.564504	0.0000000
5	71.06032	0.0000000	15.102225	0.0000000
6	77.74300	0.0000000	13.853166	0.0000000
7	82.42093	0.0000000	12.639204	0.0000000
8	87.45749	0.0000000	11.787105	0.0000000
9	88.44061	0.0000000	10.595842	0.0000000
10	89.23981	0.0000000	9.621098	0.0000000

4.1.4 Conséquences de l'autocorrélation

La principale conséquence de l'autocorrélation est que $\mathbb{E}\left(\varepsilon\varepsilon^{'}\right) = \Omega_{\varepsilon} \neq \sigma_{\varepsilon}^{2} I$. En effet, les termes de l'erreur n'étant plus indépendant au cours du temps, la matrice des variances et covariances ne peut plus être une matrice diagonale : $\mathbb{E}\left(\varepsilon_t \varepsilon_{t'}^{'}\right) = 0 \quad \forall t \neq t^{'}$.

On traduit l'autocorrélation d'ordre 1 par la relation (4.5). Cette relation permet de calculer les coavrainces entre les termes de l'erreur. On peut donc écrire :

$$\mathbb{E}\left(\varepsilon_t \varepsilon_{t-1}\right) = \mathbb{E}\left[\left(\rho\varepsilon_{t-1} + \mu_t\right)\varepsilon_{t-1}\right] = \rho\mathbb{E}\left(\varepsilon_{t-1}^{2}\right) + \mathbb{E}\left(\varepsilon_{t-1}\mu_t\right)$$

La vriance de ε_t étant homoscédastique, nous avons : $\mathbb{E}\left(\varepsilon_t^{2}\right) = \mathbb{E}\left(\varepsilon_t^{2}\right) = \sigma_{\varepsilon}^{2}$ et $\mathbb{E}\left(\varepsilon_t \mu_t\right) = 0$ car μ_t et ε_t sont indépendants.

Soit :

$$\mathbb{E}\left(\varepsilon_t \varepsilon_{t-1}\right) = \rho\sigma_{\varepsilon}^{2}$$

Plus généralement, on montre que l'auto covariance d'ordre h est donnée par la relation :

$$\mathbb{E}\left(\varepsilon_t \varepsilon_{t-h}\right) = \rho^{h}\sigma_{\varepsilon}^{2} \tag{4.36}$$

La matrice des variances et covariancs de l'erreur devient (en remplaçant $\mathbb{E}\left(\varepsilon_t \varepsilon_{t-h}\right)$ par $\rho^{h}\sigma_{\varepsilon}^{2}$)

$$\Omega_{\varepsilon} = \mathbb{E}\left(\varepsilon\varepsilon^{'}\right) = \sigma_{\varepsilon}^{2}\begin{pmatrix} 1 & \rho & \rho^{2} & \cdots & \rho^{n-1} \\ \rho & 1 & \rho & \cdots & \rho^{n-2} \\ \rho^{2} & \rho & 1 & \cdots & \rho n - 3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{n-1} & \rho^{n-2} & \rho^{n-3} & \cdots & 1 \end{pmatrix} = \sigma_{\varepsilon}^{2} W$$

La matrice W fait que l'estimation par la méthode des moindres carrés ordinaires n'est plus correcte car $\mathbb{E}\left(\varepsilon\varepsilon^{'}\right) \neq \sigma_{\varepsilon}^{2} I$. Le biais résultant de l'application directe des moindres carrés ordinaires porte uniquement sur la variance des estimateurs c'est - à - dire que les estimateurs eux - mêmes démeurent sans biais $\mathbb{E}\left(\widehat{\beta}\right) = \beta$ car la matrice

des variances et covariances n'intervient pas dans le calcul de ces estimateurs et donc ils n'ont pas à être affectés.

Par contre, la matrice des variances et covariances va être affecté. En effet, on a :

$$\mathrm{V}\left(\hat{\beta}\right) = \left(X^{'}X\right)^{-1} X^{'}\mathbb{E}\left(\varepsilon\varepsilon^{'}\right) X \left(X^{'}X\right)^{-1} = \sigma_{\varepsilon}^{2}\left(X^{'}X\right)^{-1}\left(X^{'}WX\right)\left(X^{'}X\right)^{-1}$$

avec $\sigma_{\varepsilon}^{2} = \dfrac{\sigma_{\mu}^{2}}{1-\rho^{2}}$.

L'estimateur des moindres carrés généralisés est égal à :

$$\hat{\beta} = \left(X^{'}\Omega_{\varepsilon}^{-1}X\right)^{-1}\left(X^{'}\Omega_{\varepsilon}^{-1}Y\right) \tag{4.37}$$

avec

$$\Omega_{\varepsilon}^{-1} = \frac{1}{\sigma_{\mu}^{2}}\begin{pmatrix} 1 & -\rho & 0 & 0 & \cdots & \cdots & 0 \\ -\rho & 1+\rho^{2} & -\rho & 0 & \cdots & \cdots & 0 \\ 0 & -\rho & 1+\rho^{2} & -\rho & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & & & 0 \\ 0 & \cdots & & & \cdots & 1+\rho^{2} & -\rho \\ 0 & \cdots & & & 0 & -\rho & 1 \end{pmatrix}$$

Cependant, nous ne connaissons ni le terme ρ, ni la variance σ_{μ}^{2} ; nous allons donc chercher une transformation matricielle M telle que le modèle

$$MY = MX\beta + M\varepsilon \tag{4.38}$$

ait ses erreurs indépendantes et homoscédastiques.

Soit :

$$\mathbb{E}\left(M\varepsilon(M\varepsilon)^{'}\right) = \mathbb{E}\left(M\varepsilon\varepsilon^{'}M^{'}\right) = M\mathbb{E}\left(\varepsilon\varepsilon^{'}\right)M^{'}$$

Dans ce cas, on peut déterminer l'estimateur BLUE de β par la méthode des moindres carrés ordinaires :

$$\hat{\beta} = \left((MX)^{'}MX\right)^{-1}(MX)^{'}MY = \left(X^{'}M^{'}MX\right)^{-1}X^{'}M^{'}MY \tag{4.39}$$

4.1.5 Correction de l'autocorrélation des erreurs

Il existe plusieurs méthodes qui ont comme principe commun de passer par une élimination préalable de l'erreur ε_{t}. Soit le modèle à deux variables explicatives suivant :

$$\begin{cases} y_{t} &= \beta_{0} + \beta_{1}x_{1,t} + \beta_{2}x_{2,t} + \varepsilon_{t} \\ \varepsilon_{t} &= \rho\varepsilon_{t-1} + \mu_{t} \end{cases}$$

En combinant les deux equations du système ci - dessus, on aboutit au modèle suivant :

$$y_t = \rho y_{t-1} + \beta_0 (1-\rho) + \beta_1 (x_{1,t} - \rho x_{1,t-1}) + \beta_2 (x_{2,t} - \rho x_{2,t-1}) + \mu_t \quad (4.40)$$

Plus généralement, pour un modèle de régression multiple à p variables explicatives, on aura :

$$y_t - \rho y_{t-1} = \beta_0 (1-\rho) + \beta_1 (x_{1,t} - \rho x_{1,t-1}) + \beta_2 (x_{2,t} - \rho x_{2,t-1}) + \cdots + \beta_p (x_{p,t} - \rho x_{p,t-1}) + \mu_t \quad (4.41)$$

Le terme aléatoire μ_t répond aux hypothèses d'application de la méthode des moindres carrés ordinaires, nous pouvons donc utiliser lesprocédures d'estimation des moindres carrés ordinaires sur les variables transformées. Les estimations des coefficients calculées à partir du modèle en différence s'interprètent directement comme étant les coefficients du modèle initial,sauf en ce qui concerne le terme constant :

$$\beta_0 = \frac{b_0}{1-\rho}$$

Ainsi, si nous connaissons la technique d'estimation permettant de lever l'autocorrelation d'ordre 1,nous pouvons l'appliquer de manière opérationnelle, il convient d'estimer le paramètre ρ. C'est l'objet des procédures d'estimations ssuivantes.

4.1.5.1 La méthode de Durbin et Watson en deux étapes

A l'étape 1, on estime les paramètres du modèle ci - après :

$$y_t = \rho y_{t-1} + \beta_0 (1-\rho) + \beta_1 (x_{1,t} - \rho x_{1,t-1}) + \beta_2 (x_{2,t} - \rho x_{2,t-1}) + \cdots + \beta_p (x_{p,t} - \rho x_{p,t-1}) + \mu_t \quad (4.42)$$

L'estimation de cette équation fournit les valeurs pour $\hat{\rho}, \hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$. Mais les estimateurs sont liés entre eux par une contrainte qui est que le coefficient de $x_{j,t-1}$ est égal au produit des coefficients de y_{t-1} et de $x_{j,t}$, pour tout j. Or dans la procédure d'estimation rien ne permet de tenircompte de cette contrainte„ les estimateurs obtenus ne pourront dont pas être retenu, ils constituent un ensemble d'estimateurs non cohérent. Le seul estimateur cohérent est l'estimateur de $\hat{\rho}$.

A partir de cet estimateur, il est possible d'avoir les meilleurs estimateurs de $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$. L'equation (4.42) devient :

$$\tilde{y}_t = b_0 + \beta_1 \tilde{x}_{1,t} + \beta_2 \tilde{x}_{2,t} + \cdots + \beta_p \tilde{x}_{p,t} + \mu_t \quad (4.43)$$

avec : $\tilde{y}_t = y_t - \hat{\rho} y_{t-1}$, $\tilde{x}_{j,t} = x_{j,t} - \hat{\rho} x_{j,t-1} \quad \forall j$ et $b_0 = \beta_0(1-\hat{\rho})$.

A l'étape 2, on estime le modèle définit par l'équation (4.43) en utilisant la méthode des moindres carrés ordinaires. Cette estimation fournit les vraies valeurs $\hat{b}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ avec μ_t un bruit blanc. On dit alors qu'on a corrigé l'autocorrélation des erreurs.

Exemple 4.8 *Application de la méthode de Durbin et Watson en 2 étapes*

En reprenant le modèle (4.6) dans lequel nous avons détecté une autocorrélation des erreurs, on demande d'en corriger par la méthode en deux étapes de Durbin et Watson.

4.1.5.2 La méthode de Cochrane - Orcutt

C'est une méthode itérative qui consiste à partir de l'équation (4.41) précédente en se donnant une valeur arbitraire de ρ et d'en déduire les valeurs $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$ par la méthode des moindres carrés ordinaires.

Connaissant les valeurs $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$, on revient sur une nouvelle valeur ρ et on réestime le modèle. On trouve encore d'autre valeur $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$. Et ainsi de suite. La procédure s'arrêttera lorsque les valeurs de ρ , $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$ ne changeront plus significativement.

Soit $\hat{\rho}_0 = \hat{\rho}$ la valeur d'amorçage de la procédure. Connaissant ρ_0, effectue une régression linéaire sur les quasi - différences :

$$y_t - \hat{\rho}_0 y_{t-1} = b_0 + \beta_1 \left(x_{1,t} - \hat{\rho}_0 x_{1,t-1}\right) + \cdots + \beta_p \left(x_{p,t} - \hat{\rho}_0 x_{p,t-1}\right) + \mu_t$$

Les paramètres estimés sont alors $\hat{\beta}_1, \cdots, \hat{\beta}_1$ et $\hat{\beta}_0 = \hat{b}_0 / (1 - \hat{\rho}_0)$.

A partir des nouveaux résidus d'estimations $\hat{\varepsilon}_t^1$, nous recalculons une nouvelle valeur de ρ_1 soit $\hat{\rho}_1$:

$$\hat{\varepsilon}_t^1 = y_t - \hat{\beta}_0 - \hat{\beta}_1 x_{1,t} - \cdots - \hat{\beta}_p x_{p,t}$$

et

$$\hat{\rho}_1 = \frac{\sum_{t=2}^{t=n} \hat{\varepsilon}_t^1 \hat{\varepsilon}_{t-1}^1}{\sum_{t=1}^{t=n} \left(\hat{\varepsilon}_t^1\right)^2}$$

On régrèsse sur les quasi-différences par la méthode des moindres carrés ordinaires :

$$y_t - \hat{\rho}_1 y_{t-1} = b_0 + \beta_1 \left(x_{1,t} - \hat{\rho}_1 x_{1,t-1}\right) + \cdots + \beta_p \left(x_{p,t} - \hat{\rho}_1 x_{p,t-1}\right) + \mu_t$$

puis nous calculons un nouveau résidu $\hat{\varepsilon}_t^2$ àpartir de la nouvelle estimation des coeffcients, ce qui nous permet d'obtenir un $\hat{\rho}_2$. Et ainsi de suite jusqu'à la stabilité des coeffcients β_j estimés.

Srgan a montré que cette procédure est convergente, c'est - à - dire qu'au bout d'un certains nombres d'itérations,les valeurs de $\rho, \beta_0, \beta_1, \cdots, \beta_p$ vont se stabiliser.

Il existe deux types de programmes possibles pour décider de l'arrêt de la procédure, à savoir :

— On se fixe a priori un nombre d'itération, c'est - à - dire lorsque ce nombre est atteint on arrête la procédure (généralement 5 itérations environs sont recommandées).

— On se fixe un critère de divergence sur ρ, c'est - à - dire dès que les deux valeurs successives de ρ diffèrent de moins de critère (de cette valeur seuil), on arrête la procédure. Généralement, la valeur seuil est de 1%

On dit alors qu'on a corrigé l'autocorrélation des erreurs.

En générale, la convergence est assez rapide et elle sera d'autant plus rapide lorsqu'on aura bien choisi la valeur d'amorçage. Comme la statistique de Durbin et Watson (DW) est en rapport direct avec la valeur de $DW \approx 2-2\rho$, il est commode de prendre comme valeur d'amorçage ρ_0 tel que :

$$\hat{\rho}_0 = \hat{\rho} = 1 - \frac{1}{2}DW$$

Exemple 4.9 *Méthode Cochrane - Orcutt sur les données d'Ozone*

En reprenant le modèle (4.6) dans lequel nous avons détecté une autocorrélation des erreurs, on demande d'en corriger par la méthode de cochrane - Orcutt.

Nous implémentons l'algorithme de Cochrane - Orcutt. Tout d'abord, nous construisons une fonction pour estimer le modèle :

$$\left(y_t - \hat{\rho}^{(i)} y_{t-1}\right) = \beta_0 \left(1 - \hat{\rho}^{(i)}\right) + \beta_1 \left(x_{1,t} - \hat{\rho}^{(i)} x_{1,t-1}\right) + \cdots + \beta_p \left(x_{p,t} - \hat{\rho}^{(i)} x_{p,t-1}\right) + \mu_t$$

à chaque itération i. Nous insérons également l'algorithme de **prais - winsten** qui est une version modifiée de Cochrane - Orcutt. Contrairement à Cochrane - Orcutt, l'algorithme de prais - winsten ne supprime pas la première observation.

```
# cochrane-orcutt / prais-winsten with given AR(1) rho,
# derived from ols model, default to cochrane-orcutt
import statsmodels.api as sm

def ols_ar1(model,rho,drop1=True):
  x = model.model.exog
  y = model.model.endog
  ystar = y[1:]-rho*y[:-1]
  xstar = x[1:,]-rho*x[:-1,]
  if drop1 == False:
    ystar = np.append(np.sqrt(1-rho**2)*y[0],ystar)
    xstar = np.append([np.sqrt(1-rho**2)*x[0,]],xstar,axis=0)
  model_ar1 = sm.OLS(ystar,xstar).fit()
  return(model_ar1)
```

Nous construisons la fonction d'itération :

```
# cochrane-orcutt / prais-winsten iterative procedure
# default to cochrane-orcutt (drop1=True)
def OLSAR1(model,drop1=True,rtol=0.0001,verbose=False):
  x = model.model.exog
  y = model.model.endog
```

```
  e = y - (x @ model.params)
  e1 = e[:-1]; e0 = e[1:]
  rho0 = np.dot(e1,e[1:])/np.dot(e1,e1)
  rdiff = 1.0
  allrho = list()
  i = 0
  while(rdiff>rtol):
    model1 = ols_ar1(model,rho0,drop1)
    e = y - (x @ model1.params)
    e1 = e[:-1]; e0 = e[1:]
    rho1 = np.dot(e1,e[1:])/np.dot(e1,e1)
    rdiff = np.sqrt((rho1-rho0)**2)
    rho0 = rho1
    i = i + 1
    if verbose:
      print(f"Iteration {i} avec Rho = {rho0}")
    allrho.append(rho0)
  model1 = ols_ar1(model,rho0,drop1)
  return(model1,allrho,rho0)
```

Nous testons nos fonctions :

```
# Application - Cochrane et Orcutt
model_olsar1, allrho, optimal_rho = OLSAR1(model,drop1=True,verbose=False)
coef_olsar1 = model_olsar1.summary2().tables[1]
```

Table 4.10 – Coefficients du modèle estimé (Cochrane-Orcutt)

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
x1	41.6824	3.6420	11.4449	0.0000	34.5356	48.8293
x2	-1.5459	0.1873	-8.2517	0.0000	-1.9135	-1.1783
x3	2.6086	0.1726	15.1152	0.0000	2.2699	2.9472
x4	-0.7019	0.3015	-2.3280	0.0201	-1.2935	-0.1102
x5	-1.0609	0.2667	-3.9772	0.0001	-1.5843	-0.5374
x6	0.7612	0.1516	5.0218	0.0000	0.4637	1.0586
x7	0.2342	0.0233	10.0397	0.0000	0.1884	0.2799

Visualisons la structure d'évolution de ρ.

```
# Différentes valeurs de rho
from plotnine import *

allrho = pd.DataFrame(allrho,columns=["rho"])
allrho.insert(0,"iteration",np.arange(1,len(allrho)+1))
print((ggplot(allrho,aes(x="iteration",y="rho"))+geom_point()+geom_line()))
```

Nous avons une courbe croissante en fonction de $\hat{\rho}$. On peut voir que l'algorithme a convergé après 53 itérations.

Illustrons l'approche basée sur la procédure itérative de Prais - Winsten.

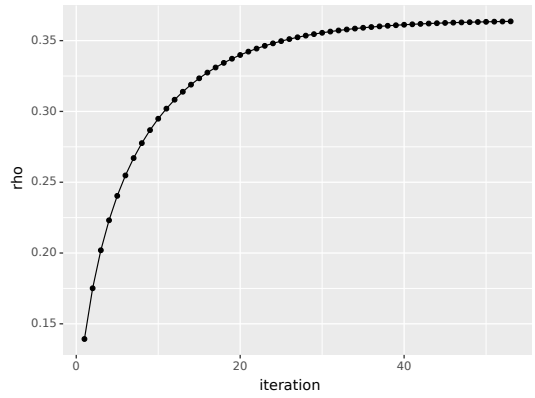


Figure 4.4 – Evolution de $\hat{\rho}$

```
# Application - Prais et Winsten
model_olsar1bis, allrho2, optimal_rho2 = OLSAR1(model,drop1=False,verbose=False)
coef_olsar1bis = model_olsar1bis.summary2().tables[1]
```

Table 4.11 – Coefficients du modèle estimé (Prais - Winsten)

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
x1	41.5399	3.6348	11.4285	0.0000	34.4073	48.6724
x2	-1.5481	0.1873	-8.2653	0.0000	-1.9157	-1.1806
x3	2.6078	0.1725	15.1170	0.0000	2.2693	2.9463
x4	-0.7047	0.3017	-2.3358	0.0197	-1.2966	-0.1127
x5	-1.0625	0.2668	-3.9821	0.0001	-1.5861	-0.5389
x6	0.7626	0.1515	5.0343	0.0000	0.4654	1.0599
x7	0.2364	0.0233	10.1464	0.0000	0.1907	0.2821

Sous Python, la fonction GLSAR de Statsmodels permet d'effectuer de déterminer la valeur optimale de ρ par la méthode de Cochrane - Orcutt. Cependant, elle se base sur l'algorithme FGLS (Feasible Generalized Least Squares).

```
# Procédure de Cochrane - Orcutt - Statsmodels
import statsmodels.api as sm

ar_gls = sm.GLSAR(model.model.endog,model.model.exog)
results = ar_gls.iterative_fit(maxiter = 100)
coef_argls = results.summary2().tables[1]
```

Table 4.12 – Coefficients du modèle estimé (Cochrane-Orcutt-GLSAR)

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	41.7388	3.6470	11.4447	0.0000	34.5822	48.8954
x1	-1.5432	0.1873	-8.2372	0.0000	-1.9108	-1.1755
x2	2.6125	0.1726	15.1386	0.0000	2.2738	2.9511
x3	-0.6988	0.3012	-2.3198	0.0206	-1.2899	-0.1077
x4	-1.0627	0.2666	-3.9869	0.0001	-1.5858	-0.5397
x5	0.7611	0.1516	5.0204	0.0000	0.4636	1.0586
x6	0.2321	0.0233	9.9425	0.0000	0.1863	0.2779

La convergence du modèle après :

```
print ('Iterations used = %d Converged %s' % (results.iter,results.converged))
```

```
## Iterations used = 66 Converged True
```

Le tableau 4.13 donne les valeurs optimales de ρ suivant les trois procédures itératives :

```
# Regroupement des valeurs
rho = pd.DataFrame({"Cochrane - Orcutt" : optimal_rho,
                    "Prais - Winster" : optimal_rho2,
                    "GLSAR - statsmodels" : ar_gls.rho},index= ["rho"])
```

Table 4.13 – Valeurs de $\hat{\rho}$

	Cochrane - Orcutt	Prais - Winster	GLSAR - statsmodels
rho	0.3636	0.3609	0.3665

4.1.5.3 La méthode de Hildreth - Lu

C'est une méthode d'estimation par « balayage ». Elle consiste toujours à partir de l'équation (4.41) puis de faire une série d'estimation pour des valeurs croissantes ou décroissantes de ρ. Par exemple, on sait que $\rho \in [0;1]$, on peut décider de prendre $\rho = 0.1; 0.2; ...; 0.9$ sur l'intervalle $\rho \in [0;1]$ et régresser pour toutes les valeurs successives de ρ avec un pas fixé ici égal à 0.1. On retient la valeur de ρ qui minimise la somme des carrés des résidus $\left(\sum_{t=1}^{t=T} \hat{\varepsilon}_t\right)$.

Remarque 4.5

Il est possible d'affiner la valeur estimée de ρ en réemployant la même procédure sur un intervalle restreint et avec un pas plus fin (par exemple 0.01). Egalement, il est à noter que cette technique est optimale selon le critère des moindres carrés puisque l'on retirn le ρ qui minimise la somme des carrés des résidus.

Les programmes de régression utilisent fréquemment la méthode de Cochrane - Orcutt pour corriger l'autocorrélation des erreurs. Dans certains cas, il peut arriver que la procédure ne suffise pas pour corriger l'autocorrélation, cela peut être dû à un problème de spécification du modèle retenu.

A côté d'une réflexion sur la forme de l'équation, il est alors utile d'explorer d'autres formes possibles d'autocorrélation, en partir lorsque l'ordre de l'autocorrélation est supérieurà 1.

Exemple 4.10 *Concentration en Ozone*

En reprenant le modèle (4.6) dans lequel nous avons détecté une autocorrélation des erreurs, on demande d'en corriger par la méthode de Hildreth - Lu.

Nous implémentons l'algorithme de Hildreth - Lu :

```
# cochrane-orcutt / prais-winsten with given AR(1) rho,
# derived from ols model, default to cochrane-orcutt (drop=True)
def ols_ar1(model,rho,drop1=True):
  x = model.model.exog
  y = model.model.endog
  ystar = y[1:]-rho*y[:-1]
  xstar = x[1:,]-rho*x[:-1,]
  if drop1 == False:
    ystar = np.append(np.sqrt(1-rho**2)*y[0],ystar)
    xstar = np.append([np.sqrt(1-rho**2)*x[0,]],xstar,axis=0)
  model_ar1 = sm.OLS(ystar,xstar).fit()
  return(model_ar1)

# hildreth-lu grid search procedure
def OLSAR1_hl(model,drop1=True,rtol=0.00001,verbose=False):
  r0 = 0; s0 = 1
  while s0 > 1.0e-5:
    rho = np.arange(r0-0.9*s0, r0+0.9*s0, s0/10)
    SSR = np.ones(np.shape(rho))
    j = 0
    for i in rho:
      model1 = ols_ar1(model,i,drop1)
      SSR[j] = model1.ssr
      j = j+1
      tab = pd.DataFrame({'rho': rho, 'SSR': SSR})
      r0 = tab['rho'][tab['SSR']==tab['SSR'].min()].values
      s0 = s0/10
      if verbose:
        print('Rho = ', r0)
  model1 = ols_ar1(model,i,drop1)
  return(model1, r0)
```

Testons notre fonction :

```
# Application - Hildreth - Lu
model_olsar_hl, optimal_rho_hl = OLSAR1_hl(model,drop1=True,verbose=False)
coef_olsar_hl = model_olsar_hl.summary2().tables[1]
```

Table 4.14 – Coefficients du modèle estimé (Hildreth - Lu)

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
x1	45.3170	4.9442	9.1656	0.0000	35.6148	55.0191
x2	-0.9024	0.1875	-4.8115	0.0000	-1.2704	-0.5344
x3	3.0533	0.1762	17.3273	0.0000	2.7075	3.3990
x4	-0.2140	0.2615	-0.8181	0.4135	-0.7272	0.2992
x5	-1.1274	0.2356	-4.7847	0.0000	-1.5898	-0.6650
x6	0.6062	0.1476	4.1084	0.0000	0.3167	0.8958
x7	-0.0376	0.0247	-1.5222	0.1283	-0.0860	0.0109

Appliquons la procédure itérative de Prais - Winsten :

```
# Application - Hildreth - Lu
model_olsar_hl2, optimal_rho_hl2 = OLSAR1_hl(model,drop1=False,verbose=False)
coef_olsar_hl2 = model_olsar_hl2.summary2().tables[1]
```

Table 4.15 – Coefficients du modèle estimé (Prais - Winsten)

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
x1	45.1030	4.9301	9.1484	0.0000	35.4284	54.7775
x2	-0.9011	0.1875	-4.8066	0.0000	-1.2690	-0.5332
x3	3.0563	0.1761	17.3570	0.0000	2.7108	3.4018
x4	-0.2139	0.2614	-0.8181	0.4135	-0.7269	0.2991
x5	-1.1292	0.2355	-4.7941	0.0000	-1.5914	-0.6670
x6	0.6072	0.1475	4.1168	0.0000	0.3178	0.8967
x7	-0.0373	0.0247	-1.5108	0.1312	-0.0857	0.0111

Le tableau 4.16 donne les valeurs optimales de ρ suivant les deux procédures itératives :

```
# Regroupement des valeurs
rho_hl = pd.DataFrame({"Hidreth - Lu" : optimal_rho_hl,
                       "Prais - Winster" : optimal_rho_hl2},index= ["rho"])
```

Table 4.16 – Valeurs de $\hat{\rho}$

	Hidreth - Lu	Prais - Winster
rho	0.4	0.4

Les valeurs sont identiques.

4.1.6 Prévision avec autocorrélation des erreurs

On considère le modèle de régression linéaire multiple

$$y_t = \beta_0 + \beta_1 x_{1,t} + \beta_{2,t} + \cdots + \beta_p x_{p,t} + \varepsilon_t, \quad t = 1, 2, \dots, n \tag{4.44}$$

où $\varepsilon_t = \rho\varepsilon_{t-1} + \mu_t$ avec $|\rho| < 1$ et $\Omega_\varepsilon = \mathbb{E}\left(\varepsilon\varepsilon^{'}\right) = \sigma_{\varepsilon'}$.
En combinant les deux équations, on a :

$$y_t^* = b_0 + \beta_1 x_{1,t}^* + \beta_2 x_{2,t}^* + \cdots + \beta_p x_{p,t}^* + \mu_t$$

où $y_t^* = y_t - \rho y_{t-1}$ et $x_{j,t}^* = x_{j,t} - \rho x_{j,t-1}$ pour tout $j = 1, \dots, p$. Cette relation satisfait les hypothèses du modèle de régresson linéaire. L'estimateur obtenu est celui des moindres carrés généralisés $\widehat{\beta}_{MCG}$.

Soit $x_{n+1}^{'} = (x_{n+1,1}, \dots, x_{n+1,p})$ une nouvelle valeur pour laquelle nous voulons prédire la variable à expliquer y_{n+1} définie par :

$$y_{n+1} = x_{n+1}^{'}\beta + \varepsilon_{n+1}$$

La prédiction optimale est donnée par :

$$\hat{y}^*_{n+1} = x^*_{n+1}\hat{\beta}_{MCG}$$

qui s'écrit en fonction des données originales comme suit :

$$\hat{y}_{n+1} = x_{n+1}\hat{\beta}_{MCG} + \rho\left(y_n - x_n\hat{\beta}_{MCG}\right)$$

Le second terme de cette équation est une estimation de l'espérance conditionnelle de ε_{n+1} car :

$$\mathbb{E}\left(\varepsilon_{n+1}|\varepsilon_n\right) = \rho\varepsilon_n = \rho\left(y_n - x_n\beta\right)$$

Ce second terme est estimé par $\rho\left(y_n - x_n\hat{\beta}_{MCG}\right)$. La variance de l'erreur de prévision est donnée par :

$$s^2_f = s^2_\varepsilon\left[1 + x^*_{n+1}\left(X^{*'}X\right)^{-1}x_{n+1}\right] \tag{4.45}$$

où $s^2_\varepsilon = \dfrac{\left(y^* - X^*\hat{\beta}_{MCG}\right)'\left(y^* - X^*\hat{\beta}_{MCG}\right)}{n-p-1}$.

4.2 L'hétéroscédasticité

Dans la théorie classique des moindres carrés ordinaires, on suppose que les erreurs sont homoscédastiques c'est - à - dire que la variance du terme d'erreur est constante pour toutes les valeurs des variables indépendantes. Si cette hypothèse n'est pas satisfaite, les estimations des paramètres ne présentent pas de biais mais elles sont inefficaces c'est - à - dire elles ne sont pas optimales. Il existe dans la classe des estimateurs linéaires sans biais un estimateur de variance plus faible. Le niveau de signification et la puissance des tests usuels ainsi que les intervalles de confiance sont alors fortement affectiés : c'est le problème de l'hétéroscédasticité.

4.2.1 Définition et causes de l'hétéroscédasticité

4.2.1.1 Définition

Par définition, il y a hétéroscédasticité lorsque la variance du terme de l'erreur (σ^2_ε) n'est pas constante sur l'ensemble des observations (au cours du temps ou sur l'ensemble de l'échantillon). S'il n'y a pas en plus autocorrélation des erreurs. La matrice des variances et covariances des termes de l'erreur (Ω_ε) reste une matrice diagonale, c'est - à - dire :

$$\Omega_\varepsilon = \mathbb{E}\left(\varepsilon\varepsilon'\right) = \begin{pmatrix} \mathbb{E}\left(\varepsilon_1^2\right) & \mathbb{E}\left(\varepsilon_1\varepsilon_2\right) & \cdots & \mathbb{E}\left(\varepsilon_1\varepsilon_n\right) \\ \mathbb{E}\left(\varepsilon_2\varepsilon_1\right) & \mathbb{E}\left(\varepsilon_1^2\right) & \cdots & \mathbb{E}\left(\varepsilon_2\varepsilon_n\right) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{E}\left(\varepsilon_n\varepsilon_1\right) & \mathbb{E}\left(\varepsilon_n\varepsilon_2\right) & \cdots & \mathbb{E}\left(\varepsilon_n^2\right) \end{pmatrix} = \begin{pmatrix} \sigma^2_{\varepsilon_1} & 0 & \cdots & 0 \\ 0 & \sigma^2_{\varepsilon_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2_{\varepsilon_n} \end{pmatrix} \tag{4.46}$$

Mais les éléments diagonaux en cas d'hétéroscédasticité ne sont plus égaux entre eux $\left(\sigma^2_{\varepsilon_1} \neq \sigma^2_{\varepsilon_2} \neq \cdots \neq \sigma^2_{\varepsilon_n}\right)$. Cette matrice va donc s'écrire :

$$\Omega_\varepsilon = \mathbb{E}\left(\varepsilon\varepsilon'\right) = \sigma^2_\varepsilon V \neq \sigma^2_\varepsilon \mathbb{I}_n \tag{4.47}$$

ce qui entraîne une remise en cause de l'hypothèse d'homoscédasticité , $\mathbb{E}\left(\varepsilon^2_t\right) = \sigma^2_\varepsilon$, $\forall t$. Ce cas de figure se rencontre davantage dans des modèles spécifiés en coupe instantanée ou alors lorsque les observations sont représentatives de moyennes.

Lorsque se pose le poblème d'hétéroscédasticité, les conséquences sont les mêmes que celles de l'autocorrélation des erreurs, à savoir :

— $\Omega_\varepsilon \neq \sigma^2_\varepsilon \mathbb{I}_n$
— Les estimateurs obtenus par la méthode des moindres carrés ordinaires demeurent sans biais mais ne sont plus à variance minimale.

4.2.1.2 Les causes de l'hétéroscédasticité

Les causes de l'hétéroscédasticité sont multiples :

— Les observations représentent les moyennes calculées sur des échantillons de tailles différentes.
— La répétition d'une même valeur de la variable endogène pour des valeurs différentes d'une variable exogène.
— Les erreurs liées aux valeurs prises par une variable exogène (par exemple : dans un modèle en coupe instantanée, la variance de la consommation croît avec le revenu disponible).

4.2.2 Détection de l'hétéroscédasticité

Il existe différents tests permettant de faire apparaître l'existence éventuelle de l'hétéroscédasticité.

4.2.2.1 Le test de Goldfeld - Quandt

Ce test par de l'idée que s'il y a hétéroscédasticité ayant un caractère systématique, c'est aux extrémités du nuage de point qu'elle sera visible. Il faut donc exclusivement s'intéresser aux extrémités du nuage de point. Ce test n'est valable que si l'une des variables est la cause de l'hétéroscédasticité et que le nombre d'observations est important. Il se déroule en deux étapes.

Etape 1 : Suppression de D observations situées au centre de l'échantillon

Ainsi, si on a n observations dans l'échantillon, on ne garde que $\dfrac{n-D}{2}$ observations aux deux extrémités (*cf.* Figure 4.5).

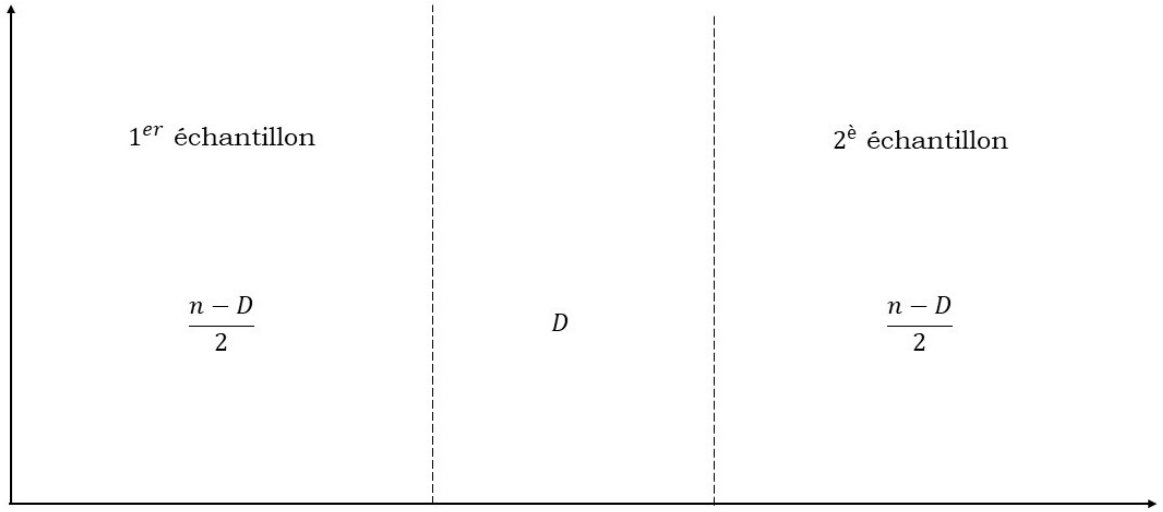


Figure 4.5 – Représentation graphique

Etape 2 : Régression sur chacune de ces deux extrémités et test

On effectue la régression sur les deux sous - échantillons et on estimé la somme des carrés des résidus (SCR). On note SCR_1 (resp. SCR_2) la somme des carrés des résidus sur le premier échantillon (resp. le second échantillon).

On vérifie ensuite si la variance résiduelle est la même pour ces deux régressions, pour cela on procède à un test d'analyse de la variance qui s'appuie sur le rapport des deux variances en plaçant au numérateur la plus forte variance. Les hypothèses du test sont :

$$\begin{cases} H_0 & : \text{homoscédasticité} \\ H_1 & : \text{hétéroscédasticité} \end{cases}$$

La statistique utilisée pour effectuer le test est celle de Fisher définie par :

$$F^* = \begin{cases} \dfrac{SCR_1/ddl1}{SCR_2/ddl2} & \text{si} \quad SCR_1 > SCR_2 \\ \dfrac{SCR_2/ddl2}{SCR_1/ddl1} & \text{si} \quad SCR_2 > SCR_1 \end{cases} \tag{4.48}$$

Sous l'hypothèse nulle, cette statistique suit une loi de Fisher à $\dfrac{n-D}{2} - p - 1$ et $\dfrac{n-D}{2} - p - 1$ degrés de liberté. Si F^* est supérieure à la statistique lue à $\dfrac{n-D}{2} - p - 1$ et $\dfrac{n-D}{2} - p - 1$ degrés de liberté, on rejette l'hypothèse nulle et on conclut que le modèle est hétéroscédasticité.

Remarque 4.6

Si les variances sont vraiment différentes, le rapport F^* sera élevé et donc, plus la valeur du rapport augmente, plus on a des chances de conclure H_1, c'est - à - dire l'hétéroscédasticité.

Exemple 4.11

Un directeur de la production d'une unité de construction automobile désire déterminer une relation entre le nombre de défauts constatés (y_i) et le temps de vérification (x_i) d'une automobile, selon le modèle suivant :

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

Pour ce faire, il procède à un test sur 30 véhicule qu'il regroupe en 6 classes de 5 voitures en demandant à chaque chef d'atelier de passer un nombre d'heures de vérification fixé. Les résultats sont consignés dans le tableau 4.20 :

Table 4.17 – Relation entre temps de vérification et nombre de défauts

y_i					x_i
4	5	6	8	8	4
6	11	13	15	17	3.5
9	13	14	15	21	2
6	13	16	23	26	1.5
11	15	17	22	34	1
7	21	23	28	38	0.5

```
# Chargement des données
import pandas as pd
vehicule = pd.read_excel("./donnee/vehicule.xlsx",index_col=0)
```

Sous Python, la fonction het_goldfeldquandt permet d'effectuer le test d'hétéroscédasticité de Goldfeld et Quandt. La valeur D doit être approximativement égale au quart du nombre d'observations totales, soit $30/4 \approx 8$.

```
# Test de Goldfeld et Quandt
import statsmodels.api as sm
def goldfeldquandt(y,x,**kwargs):
  gq = sm.stats.diagnostic.het_goldfeldquandt(y,x,**kwargs)
  res = pd.DataFrame({"statistic" : gq[0],"p-value":gq[1]},
                     index=["Goldfeld-Quandt"])
  return res
y = vehicule["y"]
x = sm.add_constant(vehicule["x"])
gq = goldfeldquandt(y,x,drop=8)
```

Table 4.18 – Test de Goldfeld et Quandt

	statistic	p-value
Goldfeld-Quandt	7.1215	0.0204

La p-value étant inférieure au seuil de 5%, l'hypothèse d'homoscédasticité H_0 est rejetée, par conséquent le modèle est donc hétéroscédastique.

4.2.2.2 Le test de Gleisjer

Il permet non seulement de déceler une éventuelle hétéroscédasticité, mais aussi d'identifier la forme que revêt cet hétéroscédasticité. Ce test est fondé sur la relation entre le résidu de l'estimation par les moindres carrés ordinaires effectuée sur le modèle de base et la variable explicative supposée êttre la cause de l'hétéroscédasticité.

Etape 1 : Régression de y_t sur les $x_{j,t}$ par les mco

On a la régression suivante : $Y = X\beta + \varepsilon$. Le vecteur des résidus $\hat{\varepsilon}$ est alors connu.

```
# Modèle général
import statsmodels.formula.api as smf
model = smf.ols("y~x",data=vehicule).fit()
coef_model = model.summary2().tables[1]
```

Table 4.19 – Coefficients du modèle de régression

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	24.0944	2.3977	10.0490	0e+00	19.1830	29.0059
x	-4.1253	0.9823	-4.1998	2e-04	-6.1374	-2.1132

Le modèle, avec un $R^2 = 0.3865$. est globalement significatif à 5% ($F = 17.64$, avec Prob(F-statistic) $=2.4540698 \times 10^{-4}$).

Etape 2 : Régression de la valeur obsolue $|\hat{\varepsilon}|$ sur x_j

Le tableau 4.20 donne les différents modèles en fonction du type de l'hétéroscédasticité.

Table 4.20 – Forme de l'hétéroscédasticité

Type	Modèle	Forme $\sigma^2_{\hat{\varepsilon}_j}$
1	$\|\hat{\varepsilon}_j\| = \alpha_0 + \alpha_1 x_j + \nu_j$	$k^2 x_j^2$ $(k \in \mathbb{R})$
2	$\|\hat{\varepsilon}_j\| = \alpha_0 + \alpha_1 x_j^{1/2} + \nu_j$	$k^2 x_j$ $(k \in \mathbb{R})$
3	$\|\hat{\varepsilon}_j\| = \alpha_0 + \alpha_1 x_j^{-1} + \nu_j$	$k^2 x_j^{-2}$ $(k \in \mathbb{R})$

— On estime les paramètres du modèle $|\hat{\varepsilon}_j| = \alpha_0 + \alpha_1 x_j + \nu_j$

```
# fonction
def form(x,p):
  return x**p
# Creation d'un dataframe
df = pd.concat([model.resid,vehicule.x],axis=1)
df.columns = ["resid","x"]
# Modèle de type 1
model_typ1 = smf.ols("abs(resid)~x",data=df).fit()
coef_typ1 = model_typ1.summary2().tables[1]
```

Le modèle, avec un $R^2 = 0.1895$. est globalement significatif à 5% ($F = 6.55$, avec Prob(F-statistic) $= 0.0161929$).

Table 4.21 – Coefficients du modèle de régression - type 1 : $|\hat{\varepsilon}_j| = \alpha_0 + \alpha_1 x_j + \nu_j$

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	8.0901	1.3996	5.7802	0.0000	5.2231	10.9571
x	-1.4673	0.5734	-2.5589	0.0162	-2.6418	-0.2927

— On estime les paramètres du modèle $|\hat{\varepsilon}_j| = \alpha_0 + \alpha_1 x_j^{1/2} + \nu_j$

```
# Modèle de type 2
model_typ2 = smf.ols("abs(resid)~form(x,1/2)",data=df).fit()
coef_typ2 = model_typ2.summary2().tables[1]
```

Table 4.22 – Coefficients du modèle de régression - type 2 : $|\hat{\varepsilon}_j| = \alpha_0 + \alpha_1 x_j^{1/2} + \nu_j$

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	10.7138	2.3019	4.6542	0.0001	5.9985	15.4291
form(x, 1 / 2)	-4.1479	1.5948	-2.6008	0.0147	-7.4147	-0.8810

Le modèle, avec un $R^2 = 0.1946$. est globalement significatif à 5% ($F = 6.76$, avec Prob(F-statistic) $= 0.0146879$).

— On estime les paramètres du modèle $|\hat{\varepsilon}_j| = \alpha_0 + \alpha_1 x_j^{-1} + \nu_j$

```
# Modèle de type 2
model_typ3 = smf.ols("abs(resid)~form(x,-1)",data=df).fit()
coef_typ3 = model_typ3.summary2().tables[1]
```

Table 4.23 – Coefficients du modèle de régression - type 3 : $|\hat{\varepsilon}_j| = \alpha_0 + \alpha_1 x_j^{-1} + \nu_j$

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	2.7944	1.2232	2.2845	0.0301	0.2888	5.3000
form(x, -1)	2.8568	1.2400	2.3039	0.0289	0.3168	5.3968

Le modèle, avec un $R^2 = 0.1594$. est globalement significatif à 5% ($F = 5.31$, avec Prob(F-statistic) $= 0.0288642$).

Etape 3 : Test

L'hypothèse d'homoscédasticité est rejetée si le coefficient α_1 d'une des trois spécifications est significativement différent de zéro. Lorsque exceptionnellement, tous les t de student de α_1 sont significativement différent de 0, on revient la spécification de l'hétéroscédasticité donc le t de student est le plus élevé.

En général lorsqu'on détecte une hétéroscédasticité de type : $\sigma^2_{\hat{\varepsilon}_t} = k^2 f\left(x_{j,t}\right)$, il convient de diviser toutes les données des différentes variables (explicatives et expliquée) par la racine carré de $f\left(x_{j,t}\right)$ soit $\sqrt{f\left(x_{j,t}\right)}$ pour retrouver un modèle homoscédastique.

Les trois t de Student empiriques sont supérieurs à 1.96, l'hétéroscédasticité est donc détectée. La forme retenue est la 2 (toutefois, la forme 1 est très proche.)

4.2.2.3 Le test de White

Le test de White est très proche de celui de Gleisjer, il est fondé sur la relation entre le carré des résidus et une ou plusieurs variables explicatives en niveau et à des puissances différentes dans une même équation de régression. Il teste les hypothèses suivantes :

$$\begin{cases} H_0 & : \text{homoscédasticité} \\ H_1 & : \text{hétéroscédasticité} \end{cases}$$

Le test de White présente l'avantage qu'il ne nécessite pas que l'on spécifie les variables qui sont à la cause de l'hétéroscédasticité. Pour tester H_0, ce teste peut se faire de deux façons ci - après :

1. **Test de White avec termes croisés**

   Il est basé sur l'estimation du modèle :

$$\hat{\varepsilon}_t^2 = \alpha_0 + \sum_{j=1}^{j=p} \alpha_j x_{j,t} + \sum_{j=1}^{j=p} \beta_j x_{j,t}^2 + \sum_{j \neq k} \theta_{i,j} x_{j,t} x_{k,t} + \mu_t$$

2. **Test de White sans termes croisés**

   Il est basé sur l'estimation du modèle :

$$\hat{\varepsilon}_t^2 = \alpha_0 + \sum_{j=1}^{j=p} \alpha_j x_{j,t} + \sum_{j=1}^{j=p} \beta_j x_{j,t}^2 + \mu_t$$

Dans les deux tests (avec ou sans termes croisés), $x_{j,t}$ sont les variables explicatives, $\hat{\varepsilon}_t$ sont les résidus issus de l'estimation par les moindres carrés ordinaires du modèle $Y = X\beta + \varepsilon$ et μ_t le terme d'erreur.

Soit n lenombre d'observations et R^2 le coefficient de déterminantion. Si l'un de ces coefficients de régression est significativement différent de 0, alors on accepte l'hypothèse d'hétéroscédasticité. Nous pouvons procéder àce test soit à l'aide d'un test de Fisher classique de nullité de coeffcients, dans ce cas, si refuse l'hypothèse nulle, alors il existe un risque d'hétéroscédasticité. Soit recourir à la statistique LM qui est donnée par :

$$Lm = n \times R^2 \tag{4.49}$$

Sous H_0, cette statistique suit une loi de χ^2 à K degrés de libertés où K est le nombre de régresseurs (exogènes) dans l'expression estimée. Si $Lm > \chi^2(K)$ lu dans la table au seuil α, on rejette l'hypothèse d'homoscédasticité des erreurs.

Exemple 4.12 *Application du test de White*

On souhaite tester l'hétéroscédasticité à l'aide du test de White.

```
# Test de White
model_white = smf.ols("form(resid,2)~x+form(x,2)",data=df).fit()
coef_white = model_white.summary2().tables[1]
```

Table 4.24 – Coefficients du modèle de régression : $\hat{\varepsilon}_j^2 = \alpha_0 + \alpha_1 x_j + b_1 x_j^2 + \nu_j$

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	136.0182	43.7037	3.1123	0.0044	46.3456	225.6908
x	-78.5815	48.2937	-1.6272	0.1153	-177.6719	20.5089
form(x, 2)	11.9844	10.2580	1.1683	0.2529	-9.0634	33.0321

Le tableau 4.25 donne les valeurs des statistiques F et LM aisni que les valeurs critiques :

```
import scipy.stats as st
# Statistique de Fisher
fvalue, fpvalue = model_white.fvalue, model_white.f_pvalue
# Statistique LM
LM = df.shape[0]*model_white.rsquared
LM_crit = st.chi2.ppf(0.95,2)
white_res = pd.DataFrame({"Fisher" : fvalue, "fpvalue" :fpvalue,"LM" : LM,
                          "lm_crit" : LM_crit},index=["White test"])
```

Table 4.25 – Test de White

	Fisher		LM	
	F	pvalue	LM	pvalue
White test	3.9564	0.0311	6.7993	5.9915

Nous sommes, dans les deux cas, amenés à rejeter l'hypothèse H_0 pour un seuil de 5%. Le modèle est donc hétéroscédastique.

Remarque 4.7

Sous Python, la fonction het_white permet d'effectuer le test de White.

```
# White test - Statsmodels
from statsmodels.stats.diagnostic import het_white
model = smf.ols("y~x",data=vehicule).fit()
res = het_white(model.resid,  model.model.exog)
labels = ['LM Statistic','LM Statistic p-value','F-Statistic','F-Test p-value']
white_test = pd.DataFrame(dict(zip(labels,res)),index=["white test"])
```

Table 4.26 – Test de White

	LM		F	
	LM	pvalue	F	pvalue
white test	6.7993	0.0334	3.9564	0.0311

Toutes les pvalues étant inférieures au seuil de 5%, on rejette l'hypothèse nulle d'homoscédasticité. Le modèle est hétéroscédastique.

4.2.2.4 Le test ARCH

Les modèles de type ARCH (*AutoRegressive Conditionnal Heteroscedasticity*) permettent de modéliser des chroniques qui ont une volatilité (ou variance ou variabilité) instantanée qui dépend du passé. Il est ainsi possible d'élaborer une prévision dynamique de la chronique en termes de moyenne et de variance. Les hypothèses à formuler pour ce test sont :

$$\begin{cases} H_0 & : \text{homoscédasticité} \\ H_1 & : \text{hétéroscédasticité} \end{cases}$$

Etape 1 : Calcul de $\hat{\varepsilon}_t$ le résidu du modèle de régression

Soit,

$$\hat{\varepsilon} = Y - X\hat{\beta}$$

Etape 2 : Calcul de $\hat{\varepsilon}_t^2$

Etape 3 : Régression autorégressive des résisus sur p retards

Partant des résidus $\hat{\varepsilon}_t$ issus de l'estimation du modèle général, la détection de l'hétéroscédasticité par le test ARCH se fait en régressant le carré des résidus $\hat{\varepsilon}_t$ sur leurs décalages puissances au carré, soit :

$$\hat{\varepsilon}_t^2 = \omega + \sum_{h=1}^{h=p} \alpha_h \hat{\varepsilon}_{t-h}^2 + \nu_t$$

Etape 4 : Test

Ce test est fondé soit sur un test de Fisher classique, soit sur le test du multiplicateur de Lagrange (LM) qui est donnée par :

$$LM = n \times R^2$$

où n est le nombre d'observations servant le calcul de la régression de l'étape 3 et R^2 est le coefficient de détermination de l'étape 3.

Sous H_0, cette statistique suit une loi de χ^2 à p degrés de liberté. Si LM est supérieure à la statistique lue dans la table au seuil α, on rejette l'hypothèse d'homoscédasticité des erreurs.

Sous Python, la fonction het_arch permet d'effectuer le test ARCH.

```
# Test ARCH
from statsmodels.stats.diagnostic import het_arch
arch_res = het_arch(model.resid)
arch_test = pd.DataFrame(dict(zip(labels,arch_res)),index=["ARCH test"])
```

Table 4.27 – Test ARCH

	LM		Fisher	
	LM	pvalue	F	pvalue
ARCH test	15.7759	0.015	5.435	0.0027

Toutes les pvalues étant inférieures au seuil de 5%, on rejette l'hypothèse nulle d'homoscédasticité. Le modèle est hétéroscédastique.

4.2.3 Correction de l'hétéroscédasticité

4.2.3.1 Estimateur des MCG

L'estimateur BLUE (*Best Linear Unbiased Estimator*) du modèle hétéroscédastique est alors celui des moindres carrés généralisés :

$$\begin{cases} \hat{\beta} & = \left(X^{'}\Omega_{\varepsilon}^{-1}X\right)^{-1}\left(X^{'}\Omega_{\varepsilon}^{-1}Y\right) \\ \Omega_{\varepsilon} & = \left(X^{'}\Omega_{\varepsilon}^{-1}X\right)^{-1} \end{cases}$$

Il n'existe pas une méthodologie unique de correction, à la différence de l'autocorrélation d'ordre 1 es erreurs, mais des méthodes que l'on applique en fonction de la cause présumée de l'hétéroscédasticité. Mais en général, la démarche consiste à déterminer une transformation des valeurs de la variable endogène (à expliquer) et des variables exogènes (explicatives) afin de se ramener à un modèle homoscédastique (à variances constantes).

4.2.3.2 Correction de l'hétéroscédasticité par MCG

L'hétéroscédasticité ne concerne que la matrice des variances et covariances des termes de l'erreur et n'affecte pas les estimateurs des moindres carrés ordinaires qui démeurent sans biais. Par contre, la variance de ces mêmes estimateurs et les tests de student sont biaisés.

Pour y remédier, on utilise la méthode proposé par Aitken (1930) appelée moindres carrés généralisés (GLS, *Generalized Least Square*) qui permet d'obtenir les estimateurs sans biais que que soit la matrice $\mathbb{E}\left(\varepsilon\varepsilon^{'}\right)$. Supposons que $\mathbb{E}\left(\varepsilon\varepsilon^{'}\right) = \sigma_{\varepsilon}^{2}V$ avec $V \neq \mathbb{I}_{n}$, V étant définie positive ($\sqrt{V}$ existe), c'est - à - dire que pour tout vecteur x, alors $x^{'}Vx > 0$. Cette propriété joue le rôle que la condition qu'une variable soit positive admet une racine carrée réelle.

On peut mettre V sous la forme d'un produit de 2 matrices M et $M^{'}$ telle que $V = MM^{'}$. Par construction V est symétrique ($V = V^{'}$). De même, M est une matrice régulière, c'est - à - dire qu'elle admet une matrice inverse M^{-1}.

Si $V = MM^{'}$ alors $V^{-1} = \left(MM^{'}\right)^{-1} = M^{-1}M^{'-1}$. On note aussi que $M^{-1}VM^{'-1} = \mathbb{I}$.

Cette propriété importante qui permet de corriger l'hétéroscédasticité est réalisable parce que V est à la fois une matrice positive et symétrique. On part du modèle :

$$Y = X\beta + \varepsilon$$

on pré-multiplie chacune des variables par M^{-1} et on obtient :

$$Y^* = X^*\beta + \varepsilon^*$$

avec $Y^* = M^{-1}Y$, $X^* = M^{-1}X$ et $\varepsilon^* = M^{-1}\varepsilon$.

Le nouveau terme de l'erreur ε^* a pour matrice de variance et covariances Ω_{ε^*} définie par :

$$\mathbb{E}\left(\varepsilon^*\varepsilon^{*'}\right) = \mathbb{E}\left(M^{-1}\varepsilon\varepsilon^{-1}M'^{-1}\right) = M^{-1}\mathbb{E}\left(\varepsilon\varepsilon'\right)M'^{-1} = M^{-1}\left(\sigma_\varepsilon^2 V\right)M'^{-1} = \sigma_\varepsilon^2\mathbb{I}_n$$

Ce nouveau terme d'erreur respecte les propriétés de non autocorrélation et d'homoscédasticité.

L'estimateur $\hat{\beta}$ dans ce cas est défini par :

$$\hat{\beta} = \left(X^{*'}X^*\right)^{-1}X^{*'}Y^* = \left(X'M'^{-1}M^{-1}X\right)^{-1}\left(X'M'^{-1}M^{-1}\right)$$
$$= \left(X'V^{-1}X\right)^{-1}X'V^{-1}Y \quad (4.50)$$

L'estimateur $\hat{\beta}$ est ici sans biais et de variance constante. Il est appelé estimateur de Aitken.

Le tableau 4.28 donne les informations sur les estimateurs des moindres carrés ordinaires et celui des moindres carrés généralisés

Table 4.28 – Comparaison entre MCO et MCG

Méthode	$\widehat{\beta}$	$\Omega_{\widehat{\beta}}$
Moindres Carrés Ordinaires	$\left(X'X\right)^{-1}X'Y$	$\sigma_\varepsilon\left(X'X\right)^{-1}$
Moindres Carrés Généralisés	$\left(X'V^{-1}X\right)^{-1}X'V^{-1}Y$	$\sigma_\varepsilon\left(X'V^{-1}X\right)^{-1}$

Propriété 4.1 *Propriétés de l'estimateur des MCG*

1. $\hat{\beta}_{MCG}$ ***est un estimateur sans biais***

   En effet, on a :

$$\begin{aligned}\hat{\beta}_{MCG} &= \left(X'V^{-1}X\right)^{-1}X'V^{-1}Y = \left(X'V^{-1}X\right)^{-1}X'V^{-1}\left(X\beta+\varepsilon\right)\\ &= \left(X'V^{-1}X\right)^{-1}X'V^{-1}X\beta + \left(X'V^{-1}X\right)^{-1}X'V^{-1}\varepsilon = \beta + \left(X'V^{-1}X\right)^{-1}X'V^{-1}\varepsilon\end{aligned}$$

   En prenant l'espérance mathématique, on a :

$$\mathbb{E}\left(\hat{\beta}_{MCG}\right) = \mathbb{E}\left(\beta\right) + \left(X'V^{-1}X\right)^{-1}X'V^{-1}\mathbb{E}\left(\varepsilon\right) = \beta$$

2. *Variance de* $\hat{\beta}_{MCG}$

   On part de $\beta + \left(X'V^{-1}X\right)^{-1}X'V^{-1}\varepsilon$ *:*

$$\begin{aligned}\Omega_{\widehat{\beta}_{MCG}} &= \mathbb{E}\left[\left(\hat{\beta}_{MCG}-\beta\right)\left(\hat{\beta}_{MCG}-\beta\right)^{'}\right] = \mathbb{E}\left[\left(X^{'}V^{-1}X\right)^{-1}X^{'}V^{-1}\varepsilon\varepsilon^{'}V^{-1}X\left(X^{'}V^{-1}X\right)^{-1}\right]\\ &= \left(X^{'}V^{-1}X\right)^{-1}X^{'}V^{-1}\mathbb{E}\left(\varepsilon\varepsilon^{'}\right)V^{-1}X\left(X^{'}V^{-1}X\right)^{-1}\\ &= \left(X^{'}V^{-1}X\right)^{-1}X^{'}V^{-1}\sigma_{\varepsilon}^{2}VV^{-1}X\left(X^{'}V^{-1}X\right)^{-1}\\ &= \sigma_{\varepsilon}^{2}\left(X^{'}V^{-1}X\right)^{-1}\end{aligned}$$

La matrice de $\hat{\beta}_{MCG}$ est donnée par les moindres carrés généralisés ou encore par les moindres carrés ordinaires sur les données transformées :

$$\Omega_{\widehat{\beta}_{MCG}} = \sigma_{\varepsilon}^{2}\left(X^{*'}X^{*}\right)^{-1} \tag{4.51}$$

Le problème est maintenant de connaître la matrice M qui permettra de déduire V et V^{-1}. Le calcul de la matrice des variances et covariances de $\hat{\beta}_{MCG}$ suppose d'avoir des informations préalables sur la matrice V, mais dans la plupart des cas, ces informations ne sont pas disponibles. C'est pourquoi la méthode des moindres carrés généralisés n'est applicable que dans un nombre limité de cas.

Lorsque l'hétéroscédasticité existe, la méthode de Gleisjer fourni l'information dont on a besoin sur la variable explicative perturbatrice. La matrice M pour une variable perturbatrice $x_{j,t}$ va s'écrire :

$$M = \begin{pmatrix}\sqrt{x_{j,1}} & 0 & \cdots & 0\\ 0 & \sqrt{x_{j,2}} & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \sqrt{x_{j,n}}\end{pmatrix}$$

et la matrice M^{-1} sera donnée par la relation :

$$M = \begin{pmatrix}1/\sqrt{x_{j,1}} & 0 & \cdots & 0\\ 0 & 1/\sqrt{x_{j,2}} & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & 1/\sqrt{x_{j,n}}\end{pmatrix}$$

Plus simplement, l'application des moindres carrés généralisés lorsqu'on a détecté la variable perturbatrice $x_{j,t}$ revient à diviser toutes les observations des différentes variables par $\sqrt{x_{j,t}}$. Et, quand les variables sont transformées, on applique simplement les moindres carrés ordinaires sur les variables transformées pour corriger l'hétéroscédasticité.

Exemple 4.13 *Correction de l'hétéroscédasticité*

Les quatres tests sont concordants : le modèle est hétéroscédastique, il convient donc d'en corriger les effets. Le test de Gleisjer a mis en évidence une relation de type 2. Le modèle estimé par les moindres carrés ordinaires est le suiivant :

$$Z_t = \alpha_1 X_t + \alpha_2 T_t + \varepsilon_t \tag{4.52}$$

avec $Z_t = y_t/\sqrt{x_t}$, $X_t = 1/\sqrt{x_t}$ et $T_t = x_t/\sqrt{x_t}$. ε_t répond aux hypothèses classiques. Il est à noter l'absence de la constante.

```
# Mutate
from plydata import define
vehicule=define(vehicule,Z= 'y/np.sqrt(x)',X='1/np.sqrt(x)',T='x/np.sqrt(x)')
# Modèle
gls_model = smf.ols("Z~X+T-1",data=vehicule).fit()
coef_gls = gls_model.summary2().tables[1]
```

Table 4.29 – Coefficients du modèle de régression : $Z_t = \alpha_1 X_t + \alpha_2 T_t + \varepsilon_t$

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
X	24.9415	2.5035	9.9628	0.0000	19.8133	30.0696
T	-4.5319	1.5355	-2.9514	0.0063	-7.6772	-1.3866

Les coefficients du modèle initial sont $\hat{\beta}_0 = \hat{\alpha}_1 =$ 24.9415 et $\hat{\beta}_1 = \hat{\alpha}_2 =$ -4.5319. Le modèle estimé est donc : $y_t =$ 24.9415 -4.5319 $x_t + \hat{\varepsilon}_t$.

Il existe bien une influence significative du temps de vérification sur le nombre de défauts constaté, chaque heure de vérification permet de supprimer en moyenne 4.5 defauts.

5 Diagnostics des résidus

Sommaire

Ce chapitre a pour but de présenter quelques tests diagnostics qui sont utilisés couramment en économétrie appliquée. Ces tests diagnostics ont été conçus pour détecter des problèmes reliés aux hypothèses statistiques de base du modèle de régression linéaire. Si ces hypothèses ne sont pas confirmées, les conclusionss auxquelles on arrive en estimant un modèle de régression peuvent être trompeuses.

Le critère des moindres carrés, comme celui de la vraisemblance, appliqué à une distribution gaussienne douteuse, est très sensible à des observations atypiques, hors « norme » (outliers), c'est-à-dire qui présentent des valeurs trop singulières. Un diagnostic doit être établi dans le cadre spécifique du modèle recherché afin d'identifier les observations **influentes**, c'est-à-dire celles dont une faible variation du $p+1$-uplet $(x_{1i}, \ldots, x_{pi}, y_i)$ introduisent une modification importante des caractéristiques du modèle.

Ces observations repérées, il n'y a pas de remède universel : supprimer une valeur aberrante, corriger une erreur de mesure, construire une estimation robuste (en norme $\mathcal{L}_1$), ne rien faire..., cela dépend du contexte et doit être négocié avec le commanditaire de l'étude.

Pour illustrer ce chapitre, nous considèreons les données Concentration en ozone disponible dans le livre de Cornillon et al. (2019). Nous expliquons le pic d'ozone (`O3`) par 6 variables : la teneur maximum en ozone la veille (`O3v`), la tempértaure prévue par Météo France à 6h (`T6`), à midi (`T12`), une variable synthétique (la projection du vent sur l'axe est-ouest notée `Vx`) et enfin les nébulosités prévues à midi (`Ne12`) et à 15h (`Ne15`). Nous avons pour ce travail $n = 1024$ observations.

```
# Chargement des données
import pandas as pd
donnee = pd.read_csv('./donnee/ozone_long.txt',sep=';')
```

On ajuste le modèle suivant :

$$\texttt{O3} = \beta_0 + \beta_1 \texttt{T6} + \beta_2 \texttt{T12} + \beta_3 \texttt{Ne12} + \beta_4 \texttt{Ne15} + \beta_5 \texttt{Vx} + \varepsilon \tag{5.1}$$

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

model = smf.ols(formula = 'O3~T6+T12+Ne12+Ne15+Vx+O3v',data=donnee).fit()
coef_model = model.summary2().tables[1]
```

Table 5.1 – Coefficients du modèle linéaire : $\texttt{O3} = \beta_0 + \beta_1 \texttt{T6} + \beta_2 \texttt{T12} + \beta_3 \texttt{Ne12} + \beta_4 \texttt{Ne15} + \beta_5 \texttt{Vx} + \varepsilon$

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	34.0895495	3.1749235	10.737125	0.0000000	27.8593255	40.3197734
T6	-1.5934597	0.1842351	-8.649057	0.0000000	-1.9549884	-1.2319311
T12	2.0775021	0.1716793	12.101063	0.0000000	1.7406119	2.4143922
Ne12	-0.9538260	0.3233660	-2.949679	0.0032546	-1.5883745	-0.3192775
Ne15	-0.7738052	0.2779763	-2.783710	0.0054747	-1.3192843	-0.2283261
Vx	0.7107781	0.1398006	5.084229	0.0000004	0.4364443	0.9851119
O3v	0.4547521	0.0204838	22.200524	0.0000000	0.4145562	0.4949480

```
# Influence du model
from statsmodels.stats.outliers_influence import OLSInfluence
influ = OLSInfluence(model)
```

5.1 Analyse des résidus

L'examen des résidus constitue une étape primordiale de la régression linéaire. Cette étape estt essentiellement sur les méthodes graphiques, et il est donc difficile d'avoir des règles strictes de décision.

5.1.1 Différents résidus

Différents types de résidus sont définis afin d'affiner leurs propriétés.

5.1.1.1 Les résidus théoriques

Les résidus théoriques correspondent à la différence entre les valeur observé et la va leur ajustée de la cible.

$$\hat{\varepsilon}_i = y_i - \hat{y}_i \tag{5.2}$$

avec $\hat{y}_i = \hat{\beta}_0 + \sum_{j=1}^{j=p} \hat{\beta}_j x_{j,i}$.

```
# Residus
import numpy as np
import matplotlib.pyplot as plt
from plotnine import *

n = len(donnee)
resid = pd.DataFrame({"obs" : np.arange(1,n+1),"resid" : model.resid})
```

```
print((ggplot(resid,aes(x="obs",y="resid"))+geom_point()+
       labs(x="individus",y="residus",title="Représentation des résidus")))
```

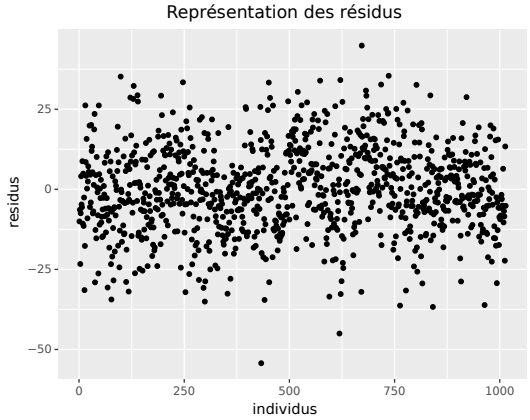


Figure 5.1 – Nuage de points des résidus

5.1.1.2 Les résidus$_{(i)}$

on a :

$$\hat{\varepsilon}_{(i)i} = y_i - \widehat{y_{(i)i}} = \frac{\hat{\varepsilon}}{1 - h_{ii}} \tag{5.3}$$

où $\widehat{y_{(i)i}}$ est la prévision de y_i, calculée sans la i-ème observation.

```
# résidus press
resid_press = pd.DataFrame({"obs" : np.arange(1,n+1),"residp":influ.resid_press})
print((ggplot(resid_press,aes(x="obs",y="residp"))+geom_point()+
       labs(x="individus",y="résidus",title="Représentation des résidus PRESS")))
```

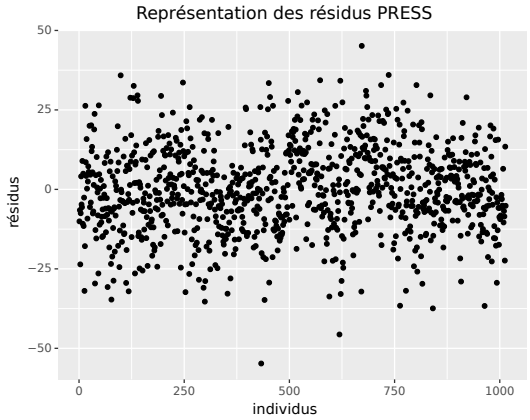


Figure 5.2 – Résidus PRESS

Ce type de résidu conduit à la définition du PRESS(*predicted residual sum of squares*) dit de Allen :

$$\text{PRESS} = \frac{1}{n}\sum_{i=1}^{i=n}\hat{\varepsilon}^2_{(i)i} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\hat{\varepsilon}}{1-h_{ii}}\right)^2 \tag{5.4}$$

C'est une estimation sans biais de la qualité de prévision d'un modèle car une même observation n'est pas utilisée, à la fois, pour estimer le modèle et l'erreur de prévision. Le PRESS est très utile pour comparer les qualités prédictives de plusieurs modèles.

```
# Valeur du PRESS
press = influ.ess_press/n
print('PRESS : %.4f'%(press))
```

```
## PRESS : 182.8723
```

5.1.1.3 Résidus normalisés

Afin d'éliminer la non - homogénéité des variances des résidus estimées, nous préférons utiliser les résidus normalisés définis par :

$$r_i = \frac{\hat{\varepsilon}_i}{\sigma\sqrt{1-h_{ii}}} \tag{5.5}$$

où h_{ij} est l'élément (i,j) de la matrice de projection.

5.1.1.4 Les résidus standardisés

Nous ne connaissons pas σ car inconnu, par conséquent nous remplaçons σ par $\hat{\sigma}$, nous obtenons les résidus standardisés :

$$t_i = \frac{\hat{\varepsilon}_i}{\hat{\sigma}_\varepsilon\sqrt{1-h_{ii}}} \tag{5.6}$$

```
# Résidus standardisés
```

Ces résidus ne sont pas indépendants par construction puisque la variance résiduelle estimée $\hat{\sigma}^2_\varepsilon$ a été estimée avec toutes les données. Ils ne peuvent donc pas être représentatifs d'une absence/présence de structuration par autocorrélation. Cependant, ils possèdent la même variance unité. Ils ont donc utiles afin de détecter des valeurs importantes de résidus. Puisqu'ils possèdent une même variance, ils sont donc dans la même « bande » de largeur constante qu'une règle « empirique » habituelle fixe à ± 2, car 2 est proche du quantile à 97.5% d'une loi normale.

5.1.1.5 Résidus studentisés

La standardisation « interne » dépend de $\hat{\varepsilon}_i$ dans le calcul de $\hat{\sigma}_\varepsilon$ estimation de $\text{V}(\hat{\varepsilon}_i)$. Une estimation non biaisé de cette variance est basée sur :

$$\hat{\sigma}^2_{\varepsilon(i)} = \frac{1}{n-3}\left[(n-p-1)\hat{\sigma}^2_{\varepsilon} - \frac{\hat{\varepsilon}^2}{1-h_{ii}}\right] \tag{5.7}$$

qui ne tient compte de la i-ème observation. On défini alors les résidus studentisés par :

$$t_i = \frac{\hat{\varepsilon}}{\hat{\sigma}_{\varepsilon(i)}\sqrt{1-h_{ii}}} \tag{5.8}$$

Ces résidus sont obtenus par validation croisée. Sous hypothèse de normalité des erreurs, on montre que ces résidus suivent une loi de Student à $(n-p-1)$ degrés de liberté.

Les résidus studentisés (*cf.* Figure 5.3) font apparaître une structuration presque négligeable en forme de sinusoïde en fonction du numéro des observations, ou du temps, le sobservations étant rangées par date de mesure. Ceci peut paraître normal puisque nous avons des variables mesurées dans le temps et cette légère variation peut être vue comme une autocorrélation (éventuelle) des résidus.

```
# Résidus studentisés
from statsmodels.nonparametric.smoothers_lowess import lowess

x = np.linspace(1,n,n)
y = np.repeat(2,n)
filtered = lowess(influ.resid_studentized_external,x)
resid = pd.DataFrame({"x": x,"y":y,"resids" : influ.resid_studentized_external})
print((ggplot(resid,aes(x="x",y="resids"))+geom_point()+
        geom_line(resid,aes(x="x",y="y"),color="green")+
        geom_line(resid,aes(x="x",y="-y"),color="green")+
        geom_line(aes(x=filtered[:,0],y=filtered[:,1]),color="red")+
        labs(x="Individu",y="Résidu studentisé par VC")))
```

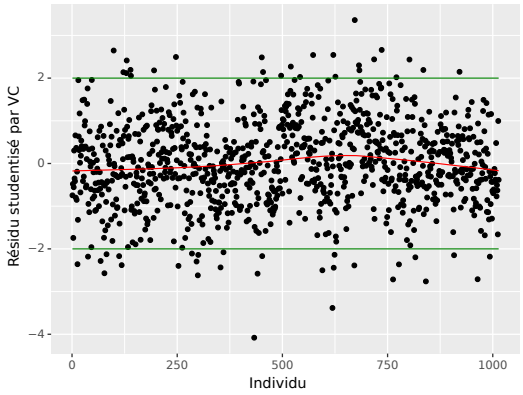


Figure 5.3 – Résidus studentisés par VC du modèle de régression à 6 variables

Comme nous sommes en présence de 1014 observations, il est normal que certaines valeurs dépassent ce seuil ponctuellement $(-2, 2)$. En fait, en moyenne 5% des individus sont à l'extérieur de la bande matérialisée par les lignes en pointillés vertes. La

courbe en noire est un lissage des valeurs des résidus studentisés. On observe une légère courbure autour du jour 600 mais rien de dramatique.

```
# Valeurs abérrantes
ab_val = donnee.iloc[np.where(np.abs(influ.resid_studentized_external) > 3)]
```

Table 5.2 – Valeurs abérrante

	O3	T6	T12	Ne12	Ne15	Vx	O3v
432	40.2	15.1	21.1	8	7	-4.3301	124.8
618	76.2	16.1	26.7	6	3	-0.6840	144.8
671	88.8	9.9	10.9	8	8	-3.0000	41.6

Les observations suspectes correspondent au jour $432, 618$ et 671. Ces observations sont donc mal expliquées par le modèle à 6 variables. Il faut regarder plus précisément ces observations pour comprendre pourquoi ces points sont aberrants. Une valeur aberrante n'est pas nécessairement une observation que l'on doit écarter de l'échantillon. C'est surtout le cas s'il s'agit d'un point levier.

5.1.2 Tests statistiques

5.1.2.1 Analyse de la normalité

L'hypothèse de normalité est examiné d'abord graphiquement en affichant le graphique quantile-quantile aussi appelé `QQ-plot`. Le `QQ-plot` est le nuage de points $(p_i, \varepsilon_{(i)})$ où :

— $\varepsilon_{(i)}$ le i^{e} plus petit élément de n réalisations $\varepsilon_1, \dots, \varepsilon_n$ d'une variable aléatoire ε et

— p_i vaut $\Phi^{-1}\left(\dfrac{i}{n+1}\right)$ où Φ est la fonction de répartition d'une $\mathcal{N}(0,1)$.

Ci-dessous (*cf.* Figure 5.4) est représenté le graphe quantile-quantile des quantiles empriques en ordonné par rapport aux quantiles théorique d'une loi $\mathcal{N}(0,1)$. Le choix de la version studentisé des résidus est nécessaire pour que chaque valeur soit normalisée et que la comparaison avec les quantiles empiriques fassent sens. On retrouve ici les 3 observations aberrantes de la section précédente : ce sont les 3 points les plus éloignés de la première bissectrice. Cependant, l'hypothèse de normalité reste très raisonnable.

```
## QQ-plot
import statsmodels.api as sm
fig, axe = plt.subplots(figsize=(16,6))
fig = sm.qqplot(influ.resid_studentized_external,line='45',ax=axe);
axe.set_title('QQ-plot');
plt.show()
```

```
## meta NOT subset; don't know how to subset; dropped
```

Si $\varepsilon \sim \mathcal{N}(0,1)$, le graphe obtenu est proche de la droite d'équation $y = x$. Cette analyse graphique peut être formalisée à l'aide du test de Shapiro-Wilk.

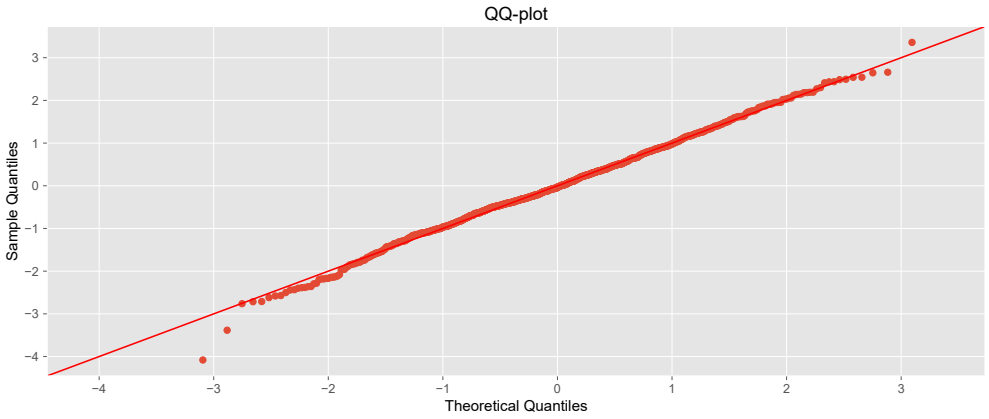


Figure 5.4 – QQ-plot des résidus studentisés

```
# Test de Shapiro
from scipy import stats
test = stats.shapiro(influ.resid_studentized_external)
shapiro = pd.DataFrame({"statistic":test[0],"pvalue":test[1]},
                       index=["shapiro Test"])
```

Table 5.3 – Test de normalité de Shapiro

	statistic	pvalue
shapiro Test	0.9976093	0.147101

Comme la pvalue est supérieur au seuil critique de 5%, on accepte l'hypothèse nulle de normalité des résidus.

5.1.2.2 Analyse de l'homoscédasticité

Dans le modèle linéaire (possiblement en absence de normalité), nous avons $\text{Cov}(\widehat{Y}, \widehat{\varepsilon}) = 0$ (qui équivaut à l'indépendance dans le modèle gaussien) de sorte que le nuage de points $(\widehat{y}_i, t_i^*)$ne devrait pas exhiber de structure particulière. Un nuage de points en forme de trompette révèle souvent une hétéroscédasticité alors qu'une forme courbe de type banane indique plutôt l'existence d'une non-linéarité.

À gauche sont représentés les résidus studentisés contre les valeurs ajustées de la variable réponse. On retrouve ces 3 observations un peu en dehors de la norme. Cependant, on observe pas de structure particulière. On observe un phénomène très similaire sur le graphique de droite représentant la valeur absolue des résidus studentisés en fonction de la valeur ajustée de la variable réponse. Le lissage permet de mieux objectiver le fait qu'il n'y ait pas de structure.

```
# Analyse de l'hétéroscédasticité
fig, (axes1,axes2) = plt.subplots(1,2,figsize=(16,6))
fig.subplots_adjust(wspace=0.5)
filtered = lowess(abs(influ.resid_studentized_external),model.fittedvalues)
axes1.scatter(model.fittedvalues,influ.resid_studentized_external);
axes2.scatter(model.fittedvalues,np.abs(influ.resid_studentized_external));
```

```
axes2.plot(filtered[:,0],filtered[:,1],color = 'black');
axes1.set(xlabel="Valeur ajustée Y",ylabel="Résidu studentisé par VC",
          title="Résidus studentisés \n versus valeurs ajustées");
axes2.set(xlabel="Valeur ajustée Y",ylabel="Résidus studentisés en valeur abs.",
          title="Résidus studentisés versus \n valeurs ajustées en valeur abs.");
plt.tight_layout()
plt.show()
```

```
## meta NOT subset; don't know how to subset; dropped
```

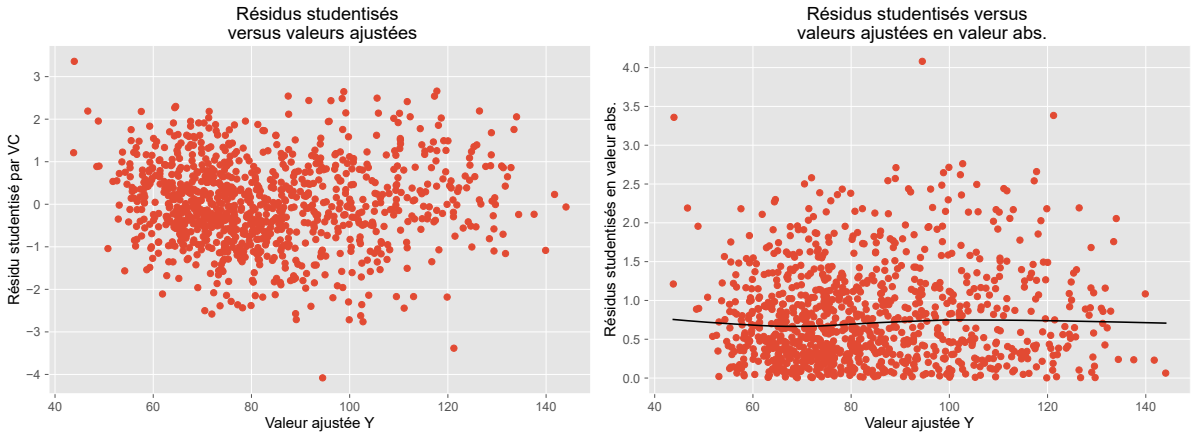


Figure 5.5 – Analyse de l'hétéroscédasticité

5.1.2.3 Analyse de la structure des résidus

Il a déjà été procédé à une première analyse de la structure des résidus dans la section précédente. On peut affiner cette analyse en représentant le graphique des résidus studentisés en fonction de chaque variables explicatives.

```
exog_name = model.model.exog_names[1:]
fig = plt.figure(figsize=(16,7))
for i, col in enumerate(exog_name):
  ax =fig.add_subplot(2,3,i+1)
  filtered = lowess(influ.resid_studentized_external,donnee[col])
  ax.scatter(donnee[col],influ.resid_studentized_external);
  ax.plot(filtered[:,0],filtered[:,1],color = 'black')
  ax.set(xlabel=col,title="Résidus stud.",ylabel=f"Résidus versus {col}");
plt.tight_layout()
plt.show()
```

```
## meta NOT subset; don't know how to subset; dropped
```

Les graphiques pour les variables T6, Ne12, Ne15 n'exhibe pas de structure particulière si bien que l'on peut supposer que l'hypothèse linéaire en ces variables est raisonnable ainsi que l'hypothèse d'homoscédasticité.

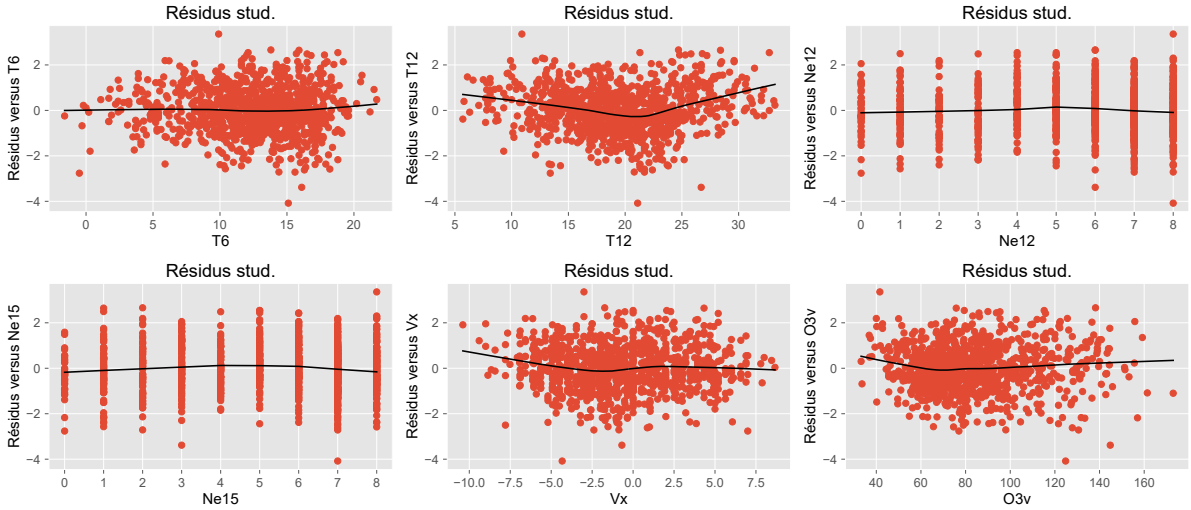


Figure 5.6 – Résidus partiels pour les 6 variables explicatives. Le trait continu représente le résumé lissé des données par le lisseur loess.

En revanche pour les variables Vx, O3v et plus particulièrement T12 on observe un changement de régime matérialisé par une brisure de la courbe de lissage. Cela peut être dû à une hétéroscédasticité inhérente au phénomène physique ou à une erreur de modèle. Typiquement, on pourrait tenter de décomposer T12 en deux variables qui modéliserait le lien avec la température à midi dans chacun des régime inférieure ou supérieure à 20°C. Ceci peut-être fait à l'aide d'indicatrice.

5.1.3 Autres mesures de diagnostiques

5.1.3.1 Effet levier et observations influentes

Une autre indication est donnée par l'éloignement des x_{ji} par rapport à leur moyenne $\overline{x}_j$. En effet, écrivons le vecteur prédicteur $\hat{Y}$ comme combinaison linéaires des observations :

$$\hat{y} = X\hat{\beta} = X(X'X)^{-1}X'Y = Hy, \quad \text{avec} \quad H = X(X'X)^{-1}X' \tag{5.9}$$

H est une matrice de format $n \times n$ et ses éléments diagonaux $h_{ii} (i = 1, \dots, n)$ mesurent ainsi l'impact ou l'importance du rôle que joue y_i dans l'estimation de $\hat{y}_i$. Nous pouvons écrire :

$$\hat{y}_i = \sum_{j=1}^{j=n} h_{ij} y_j = h_{ii} y_i + \sum_{j \neq i} h_{ij} y_j \tag{5.10}$$

Ainsi, h_{ii} représente le poids de l'observation i sur sa propre estimation. La matrice H et ses coefficients $\left(h_{ij}\right)_{i,j=1,\dots,n}$ vérifient :

1. $\text{Trace}(H) = \sum_{i=1}^{n} h_{ii} = p$, la moyenne des h_{ii} vaut donc p/n. Ainsi, si h_{ii} est « grand » y_i influe fortement sur $\hat{y}_i$.

2. Trace $(H) =$ Trace $(HH) = \sum_{i,j=1}^{n} h_{ij}^2 = p$
3. $\forall i = 1, \dots, n \; h_{ii} \in [0, 1]$
4. $\forall i, j = 1, \dots, n \; h_{ij} \in \left[\frac{1}{2}; \frac{1}{2}\right]$
5. Si $h_{ii} = 1$ alors $h_{ij} = 0 \; \forall j \neq i$
6. Si $h_{ii} = 0$ alors $h_{ij} = 0 \; \forall j \neq i$

Nous avons les cas extrêmes suivants :

— Si $h_{ii} = 1$, $\hat{y}_i$ est entièrement déterminée par y_i car $h_{ij} = 0 \; \forall j \neq i$;
— Si $h_{ii} = 0$, $\hat{y}_i$ n'a pas d'influence sur $\hat{y}_i$ (qui vaut alors 0).

Un point est un point levier si les valeurs h_{ii} de H dépasse les valeurs suivantes :

— $h_{ii} > 2p/n$ selon Hoaglin et Welsch (1978)
— $h_{ii} > 3p/n$ pour $p > 6$ et $n - p - 1 > 12$ selon Velleman et Welsch (1981)
— $h_{ii} > 1/2$ selon Huber (1981)

Cette notion de levier h_{ii} correspond à l'éloignement du centre de gravité de la i^e ligne X : plus le point est éloigné, plus la valeur des h_{ii} augmente. Un point levier n'est pas un point aberrant car il se situe dans le prolongement de la droite de régression et donc son résidu est faible.

```
# Points leviers
data = pd.DataFrame({"x":x,"hii":influ.hat_diag_factor})
print((ggplot(data,aes(x="x",y="hii"))+geom_point()+
       geom_hline(yintercept = [2*7/n,3*7/n],color=["green","blue"])+
       labs(x="Individu",y="$h_{ii}$",title="Points leviers")))
```

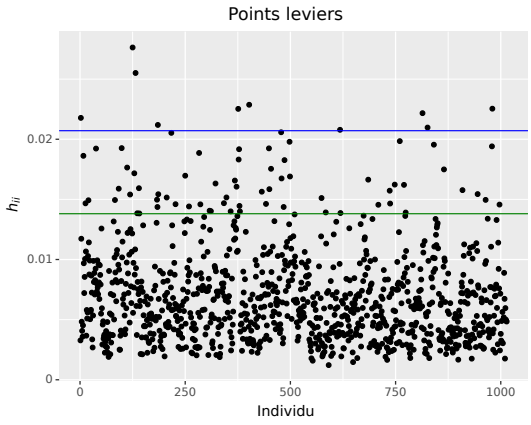


Figure 5.7 – Points leviers

5.1.3.2 Distance de Cook

La distance de Cook mesure l'influence de l'observation i sur l'estimation du paramètre β. La distance de Cook associée à l'observation i est définie par :

$$\mathrm{C}_i = \frac{1}{(p+1)\hat{\sigma}^2}\left(\hat{\beta}_{(i)} - \hat{\beta}'\right)\left(X'X\right)\left(\hat{\beta}_{(i)} - \hat{\beta}'\right) = \frac{\left(\hat{y}_i - x_i'\hat{\beta}_{(i)}\right)^2}{(p+1)\hat{\sigma}^2} \tag{5.11}$$

Il est cependant possible de la réexprimer de manière plus concise et plus simple à calculer comme :

$$\mathrm{C}_i = \frac{1}{p+1}\frac{h_{ii}}{1-h_{ii}} = \frac{h_{ii}}{(p+1)(1-h_{ii})^2}\frac{\hat{\varepsilon}_i^2}{\hat{\sigma}^2} \tag{5.12}$$

In est important de noter que la loi de C_i n'est pas une loi de Fisher car $\hat{\beta}_{(i)}$ n'est pas indépendant de $\hat{\beta}$ et $\hat{\sigma}^2$. Pour autant, si $\hat{\beta}_{(i)}$ est remplacée par le vrai paramètre (inconnu) $\beta_0 \in \mathbb{R}^p$, la variable aléatoire

$$\frac{1}{(p+1)\hat{\sigma}^2}\left(\hat{\beta}_0 - \hat{\beta}'\right)\left(X'X\right)\left(\hat{\beta}_0 - \hat{\beta}'\right) \tag{5.13}$$

suit une loi de Fisher $f_{p+1,n-p-1}$. Comme précédemment, une observation i dépassant le seuil fixé doit amener à se poser des questions de conserver ou non cet individu dans l'échantillon. Les mêmes précautions doivent cependant être observées que pour l'écart des observations aberrantes.

```
#Distance de Cook
yok = np.repeat(0.1,n)
cd, pval = influ.cooks_distance
data = pd.DataFrame({"x":x,"cd":cd})
print((ggplot(data,aes(x="x",y="cd"))+geom_point()+
       geom_hline(yintercept = 0.1, color="green")+
       labs(x="Individu",y="Distance de Cook",title="Distance de Cooks")))
```

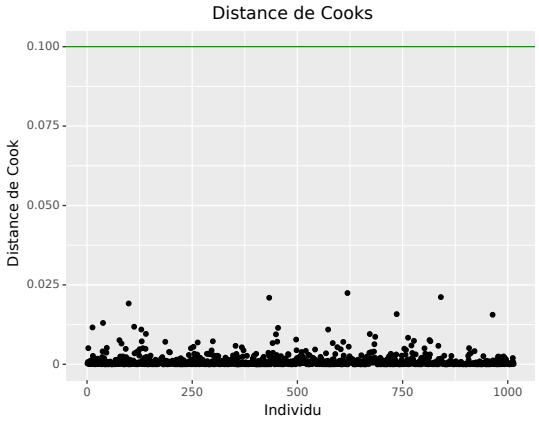


Figure 5.8 – Distance de Cook

En ce qui concerne les observations influentes (*cf.* Figure 5.7), aucune observation ne montre une distance de Cook nettement supérieure aux autres et il ne semble pas y avoir d'observation très influente. De plus, le seuil $f_{p,n-p-1}(0.1) = 0.4$ est supérieur à toutes les observations. Au niveau des points leviers, nous observons que beaucoup d'individus statistiques sont plus grands que le seuil indicatif de $2p/n$, 8 seulement

sont au - dessus du seuil $3p/n$ et enfin aucun n'est aux environs de 0.5. De manière plus générale, aucune observation ne montre un h_{ii} très différent des autres. En conclusion, nous conservons toutes les informations.

5.1.3.3 DFFITS

Une autre mesure d'influence est fréquemment utilisée, il s'agit de l'écart de Welsh-Kuh. Cette mesure est souvent appelée DFFITS dans les logiciels statistiques. L'écart de Welsh-Kuh associé à l'observation i est défini par :

$$\text{WK}_i = |t_i^\bullet| \sqrt{\frac{h_{ii}}{1-h_{ii}}} \tag{5.14}$$

où $t_i^\bullet$ est le résidu studentisé obtenu par validation croisée. L'écart de Welsh-Kuh, au facteur $1/(p+1)$ près, est la distance de Cook dans laquelle l'estimation de $\hat{\sigma}^2$ se fait à l'aide $\hat{\sigma}_{(i)}$. Le seuil recommandé pour WK_i est $2\sqrt{\frac{p+1}{n-p-1}}$.

Cette quantité permet d'évaluer l'écart standardisé entre l'estimation bâtie sur toutes les observations et l'estimation bâtie sur toutes les observations sauf la i^e. Cet écart de Welsh - Kuh mesure ainsi l'influence simultanée d'une observation sur l'estimation des paramètres β et σ^2.

```
# DFFITS
obs, dffits = influ.dffits
data = pd.DataFrame({"x" : x, "obs" : abs(obs)})
print((ggplot(data,aes(x="x",y="obs"))+geom_point()+
       geom_hline(yintercept = [dffits,1.5*dffits], color="green")+
       labs(x="Individu",y="Écart de Welsh-Kuh",title="Mesure DFFITS")))
```

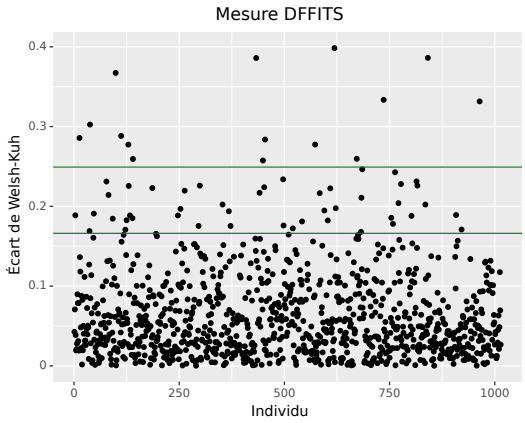


Figure 5.9 – Mesures DFFITS

5.2 Sélection des variables

La sélection de modèle peut être vue comme la recherche du modèle optimal au sens d'un critère choisi, parmi tous les modèles possibles. Cela peut être vu comme une

optimisation d'une fonction objectif (le critère). Pour cela et à l'image des possibilités en optimisation, on peut soit faire une recherche exhaustive car le nombre de modèles possibles est fini, soit partir d'un point de départ et utiliser une méthode d'optimisation de la fonction objectif (recherche pas à pas).

5.2.1 Recherche exhaustive

Lorsque tous les modèles avec p variables sont possibles, il y a 2^p possibilités et donc cette méthode n'est pas envisageable si p est grand. Des techniques algorithmiques permettent cependant de minimiseer le nombre de calculs à effectuer et permettent d'envisager cette possibilité dans des cas de taille modérée.

Remarquons que ce type de recherche n'a aucun sens lorsque l'on souhaite utiliser des tests puisque cette procédure compare uniquement deux modèles emboîtés l'un dans l'autre.

5.2.2 Recherche pas à pas

Ce type de recherche est obligatoire pour les tests puisque l'on ne peut tester que des modèles emboités. Par constre, elle ne permet en général que de trouver un optimum local. Il est bien de répéter cette procédure à partir de différents points de départ. Pour les autres critères, ce type de recherche n'est à conseiller que lorsque la recherche exhaustive n'est pas possible (n grand, p grand, etc...).

5.2.2.1 Méthode ascendante

Appelée *forward selection*, son principe est le suivant : A chaque pas, une variable est ajoutée au modèle.

— Si la méthode ascendante utilise un test F, nous rajoutons la variable X_i dont la probabilité critique (*fp-value*) associée à la statistique partielle de test de Fisher qui compare les 2 modèles est minimale. Nous nous arrêtons lorsque toutes les variables sont intégrées ou lorsque la probabilité critique est plus grande qu'une valeur seuil.
— Si la méthode ascendante utilise un critère de choix, nous ajoutons la variable X_i dont l'ajout au modèle conduit à l'optimisation de la plus grande du critère de choix. Nous nous arrêtons lorsque toutes les variables sont intégrées ou lorsqu'aucune variable ne permet l'optimisation du critère de choix.

Contrairement à `R`, la librairie `statsmodels` n'a pas de procédure toute faite pour la sélection de variables. On définit donc une fonction pour l'implémenter. Voici un code pour la procédure `forward` basée sur le R^2 ajusté (équivalent à la minimisation de la variance). Le lecteur peut écrire une fonction équivalente pour les C_p de Mallow, l'AIC et le BIC. Nous présentons ici un code modifié de celui téléchargé ici.

```
# Forward selection
def forward_selection(data, response, verbose = True):
    remaining = set(data.columns)
    remaining.remove(response)
    selected = []
```

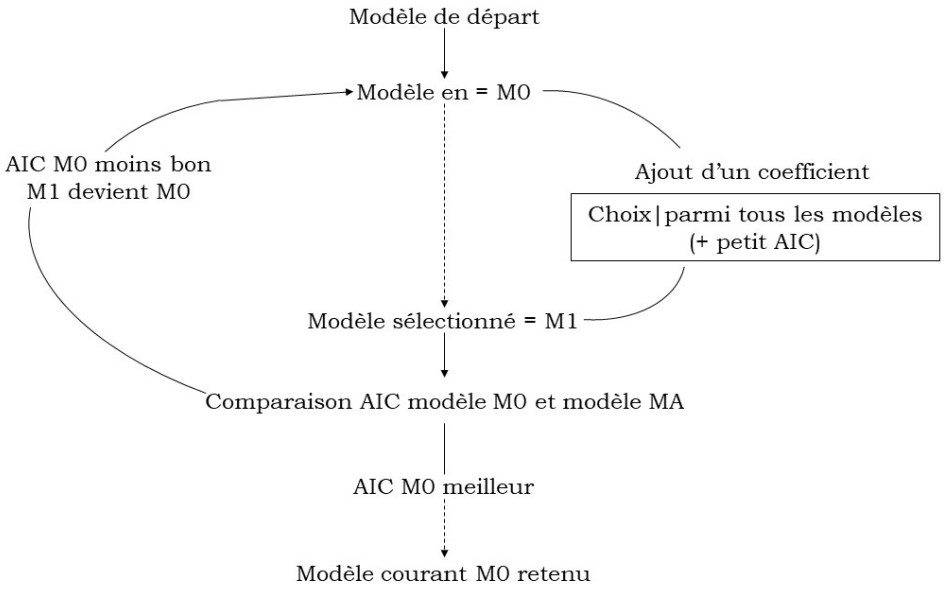


Figure 5.10 – Technique ascendante utilisant l'AIC

```
    current_score, best_new_score = 0.0, 0.0
    while remaining and current_score == best_new_score:
        scores_with_candidates = []
        for candidate in remaining:
            formula = "{} ~ {} + 1".format(response,
                                           ' + '.join(selected + [candidate]))
            score = smf.ols(formula, data).fit().rsquared_adj
            scores_with_candidates.append((score, candidate))
        scores_with_candidates.sort()
        best_new_score, best_candidate = scores_with_candidates.pop()
        if current_score < best_new_score:
            remaining.remove(best_candidate)
            selected.append(best_candidate)
            current_score = best_new_score
            if verbose:
                print('Add {:10} with Adj. R-squared {:.6}'.format(best_candidate,
                                                                   best_new_score))

    formula = "{} ~ {} + 1".format(response,' + '.join(selected))
    model = smf.ols(formula, data).fit()
    return model

# Application
forward_model = forward_selection(donnee, 'O3')
```

```
## Add O3v         with Adj. R-squared 0.472306
## Add Ne12        with Adj. R-squared 0.598518
## Add T12         with Adj. R-squared 0.62222
## Add T6          with Adj. R-squared 0.655414
```

```
## Add Vx          with Adj. R-squared 0.664948
## Add Ne15        with Adj. R-squared 0.667177
```

```
print(forward_model.model.formula)
```

```
## O3 ~ O3v + Ne12 + T12 + T6 + Vx + Ne15 + 1
```

On remarque qu'aucune variable explicative n'a été supprimée. Basons nous à présent sur la p-value :

```
# Méthode ascendante basée sur la pvalue
import statsmodels.api as sm

def forward_regression(X, y,threshold_in,verbose=False):
    initial_list = []
    included = list(initial_list)
    while True:
        changed=False
        excluded = list(set(X.columns)-set(included))
        new_pval = pd.Series(index=excluded)
        for new_column in excluded:
            model = sm.OLS(y,
                  sm.add_constant(pd.DataFrame(X[included+[new_column]]))).fit()
            new_pval[new_column] = model.pvalues[new_column]
        best_pval = new_pval.min()
        if best_pval < threshold_in:
            best_feature = new_pval.idxmin()
            included.append(best_feature)
            changed=True
            if verbose:
                print('Add  {:30} with p-value {:.6}'.format(best_feature,
                                                             best_pval))

        if not changed:
            break
    return included
# Application
forward_reg = forward_regression(donnee[["O3v","Ne12","T12","T6","Vx","Ne15"]],
                                 donnee["O3"],0.01)
print(forward_reg)
```

```
## ['O3v', 'Ne12', 'T12', 'T6', 'Vx', 'Ne15']
```

5.2.2.2 Méthode descendante

Appelée *backward selection*, à la première étape toutes les variables sont intégrées au modèle :

— Si la méthode descendante utilise un test F, nous éliminons ensuite la variable X_i dont la valeur p, associée à la statistique partielle de test de Fisher, est la plus

grande. Nous nous arrêtons lorsque toutes les variables sont retirées du modèle ou lorsque la valeur p est plus petite qu'une valeur seuil.

— Si la méthode descendante utilise un critère de choix, nous retirons la variable X_i dont le retrait du modèle conduit à l'augmentation la plus grande du critère de choix. Nous nous arrêtons lorsque toutes les variables sont retirées ou lorsque qu'aucune variable ne permet l'augmentation du critère de choix.

```
# Mathode descendante
def backward_regression(X, y,threshold_out,verbose=False):
    included=list(X.columns)
    while True:
        changed=False
        model = sm.OLS(y, sm.add_constant(pd.DataFrame(X[included]))).fit()
        # use all coefs except intercept
        pvalues = model.pvalues.iloc[1:]
        worst_pval = pvalues.max() # null if pvalues is empty
        if worst_pval > threshold_out:
            changed=True
            worst_feature = pvalues.idxmax()
            included.remove(worst_feature)
            if verbose:
                print('Drop {:30} with p-value {:.6}'.format(worst_feature,
                                                             worst_pval))
        if not changed:
            break
    return included
# Application
backward_reg=backward_regression(donnee[["O3v","Ne12","T12","T6","Vx","Ne15"]],
                                 donnee["O3"],0.01)
print(backward_reg)
```

```
## ['O3v', 'Ne12', 'T12', 'T6', 'Vx', 'Ne15']
```

5.2.2.3 Méthode progressive

Appelée *stepwise selection*, le principe est le même que pour la méthode ascendante, sauf que l'on peut éliminer des variables déjà introduites. En effet, il peut arriver que des variables introduites en début ne soient plus significatives après introduction de nouvelles variables. Remarquons qu'en général la variable « constante », constituée de 1 et associée au coefficient « moyenne générale », est en général traitée à part et elle est toujours présente dans le modèle.

On retrouve https ://github.com/AakkashVijayakumar/stepwise-regression des codes pour les méthodes forward et backward basées sur la p-value. Ou encore en utilisant le package stepwise-regression.

6 Le modèle linéaire général

Sommaire

Jusqu'à présent, les variables explicatives étaients quantitatives continues. Or, il arrive fréquemment que certaines variables explicatives soient des variables qualitatives. Le modèle linéaire général est une extension de la régression linéaire multiple au cas où il y a des variables explicatives qualitatives. Dans ce cas, pouvons-nous appliquer la méthode des moindres carrés vue dans les chapitres précédents ?

Nous reprenons l'exemple d'ozone où nous souhaitons expliquer la concentration en ozone `O3` en fonction de la température `T12` et de la direction du vent `vent`, variable qualitative prenant 4 modalités : `NORD`, `SUD`, `EST` et `OUEST`.

```
# Chargement des données
import pandas as pd

donnee = pd.read_csv('./donnee/ozone.txt',sep=';',index_col=0)
```

Table 6.1 – Tableau de données brutes

	O3	T12	T15	Ne12	N12	S12	E12	W12	Vx	O3v	nebulosite	vent
19960422	63.6	13.4	15.0	7	0	0	3	0	9.35	95.6	NUAGE	EST
19960429	89.6	15.0	15.7	4	3	0	0	0	5.40	100.2	SOLEIL	NORD
19960506	79.0	7.9	10.1	8	0	0	7	0	19.30	105.6	NUAGE	EST
19960514	81.2	13.1	11.7	7	7	0	0	0	12.60	95.2	NUAGE	NORD
19960521	88.0	14.1	16.0	6	0	0	0	6	-20.30	82.8	NUAGE	OUEST
19960528	68.4	16.7	18.1	7	0	3	0	0	-3.69	71.4	NUAGE	SUD

Nous avons 2 variables explicatives : température à 12 h (`T12`) et direction du vent (`vent`). Pouvons-nous effectuer une régression multiple ? Comment utiliser la variable `vent` ? Dans cet exemple simple, nous pouvons représenter les données avec en abscisse la température, en ordonnée la concentration en ozone et couleur la direction du vent.

```
#Nuage
from plotnine import *
p = (ggplot(donnee,aes(x="T12",y="O3",fill="vent", linetype="vent"))+
     geom_point(aes(shape="vent",color="vent"))+
     geom_smooth(method="lm",se=False)+
     theme(legend_position=(0.2,0.7),legend_direction="vertical"))
print(p)
```

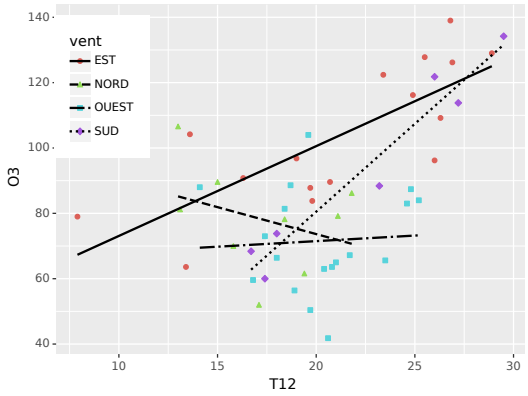


Figure 6.1 – Nuage de points et régression simple pour chaque direction du vent

Les pentes des différentes régressions sont différentes, il semble que la modélisation de la concentration de l'ozone en fonction de la température dépende de la variable vent. Ainsi, la direction du vent pourrait avoir un effet sur la concentration en ozone, mais cela est difficile à observer.

6.1 Modélisation du problème

6.1.1 Modélisation

En présence des variables qualitatives dans le modèle, il faut procéder à deux opérations.

— Chaque variable qualitative est remplacée par les variables indicatrices de ses modalités. Par exemple, la variable `vent` est remplacée par les variables `EST,OUEST,NORD,SUD`.
— On peut remarquer que la somme des variables indicatrices associées à une variable qualitative est égale à 1. Ceci provoque une redondance entre les variables explicatives qu'il faut prendre en compte. Pour résoudre ce problème, on doit supprimer une variable indicatrice de modalité par variable qualitative. Le choix de la modalité à supprimée est arbitraire et n'a aucune influence sur le calcul des valeurs prédictes. En général, on supprime la variable indicatrice de la dernière modalité de chaque variable qualitative, ou bien la première. Les modèles supprimées servent de référence.

Ici, nous supprimons la modalité `EST`. Le modèle étudié s'écrit :

$$\texttt{O3} = \beta_0 + \beta_1 \texttt{T12} + \beta_2 1_{\{\text{vent=NORD}\}} + \beta_3 1_{\{\text{vent=OUEST}\}} + \beta_4 1_{\{\text{vent=SUD}\}} + \varepsilon \tag{6.1}$$

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

modelcomplet = smf.ols(formula='O3~T12+vent',data=donnee).fit()
coef_table = modelcomplet.summary2().tables[1]
```

Table 6.2 – Coefficients du modèle de régression

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	52.3484	12.3370	4.2432	0.0001	27.5005	77.1964
vent[T.NORD]	-15.8292	7.3121	-2.1648	0.0357	-30.5564	-1.1019
vent[T.OUEST]	-29.9384	5.7761	-5.1831	0.0000	-41.5721	-18.3047
vent[T.SUD]	-12.8550	7.6239	-1.6861	0.0987	-28.2102	2.5003
T12	2.4300	0.5476	4.4377	0.0001	1.3271	3.5329

Le modèle estimée s'écrit :

$$\widehat{\texttt{O3}} = 52.35 + 2.43 \times \texttt{T12} - 15.83 \times 1_{\{\text{vent=NORD}\}} - 29.94 \times 1_{\{\text{vent=OUEST}\}} - 12.86 \times 1_{\{\text{vent=SUD}\}} \tag{6.2}$$

En choisissant un risque $\alpha = 5\%$, les niveaux de signification donnés montrent que la variable T12 est significative, ainsi que les variables NORD et OUEST. La variable SUD n'est pas significative.

```
# Prédiction
prediction = modelcomplet.get_prediction(exog=donnee[['T12','vent']])
frame = prediction.summary_frame(alpha=0.05)
frame.index = donnee.index
```

Table 6.3 – Prédiction

	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
19960422	84.91091	5.977468	72.87168	96.95015	49.10767	120.71415
19960429	72.96981	5.707649	61.47402	84.46561	37.34563	108.59399
19960506	71.54572	8.396836	54.63362	88.45782	33.82374	109.26770
19960514	68.35274	6.012801	56.24234	80.46315	32.52551	104.17998
19960521	56.67356	5.181691	46.23710	67.11002	21.37698	91.97014
19960528	80.07508	7.097521	65.77994	94.37022	43.45159	116.69857

Le graphique de la concentration en ozone observée *versus* la concentration en ozone calculée est donné dans la figure 6.2.

```
# Graphique
df = pd.concat([donnee["O3"],frame['mean']],axis=1)
print((ggplot(df,aes(x="mean",y="O3"))+geom_point(color="blue")))
```

6.1.2 Tests sur les variables explicatives

Nous allons tester la significativité de nos variables explicatives.

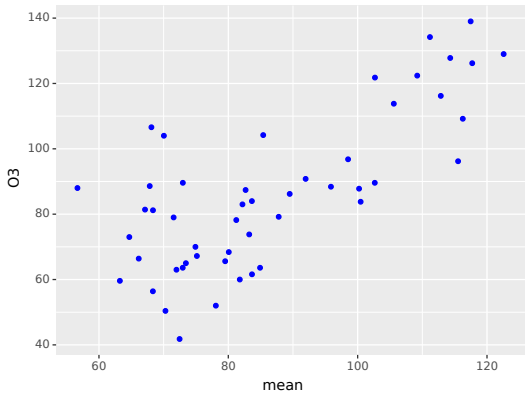


Figure 6.2 – Concentration en ozone observée versus calculée

6.1.2.1 Tests sur les variables quantitatives

Il s'agit d'étudier sur le modèle (6.1) le test d'hypothèse :

$$\begin{cases} \mathrm{H}_0 : \beta_1 = 0 \\ \mathrm{H}_1 : \beta_1 \neq 0 \end{cases}$$

On utilise la statistique :

$$\mathrm{F} = \frac{\text{SCE(modèle complet) - SCE(modèle complet sauf T12)}}{\text{SCR(modèle complet)}/(n-k-1)} \tag{6.3}$$

On régresse O3 sur vent, c'est-à-dire le modèle suivant :

$$\texttt{O3} = \alpha_0 + \alpha_1 1_{\{\text{vent=NORD}\}} + \alpha_2 1_{\{\text{vent=OUEST}\}} + \alpha_3 1_{\{\text{vent=SUD}\}} + \eta \tag{6.4}$$

```
# Modèle sans température T12
modelsanst12 = smf.ols(formula='O3~vent',data=donnee).fit()
coefsanst12 = modelsanst12.summary2().tables[1]
```

Table 6.4 – Coefficients du modèle de régression sans température T12

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	103.8500	4.9634	20.9233	0.0000	93.8593	113.8407
vent[T.NORD]	-25.5611	8.2723	-3.0900	0.0034	-42.2123	-8.9099
vent[T.OUEST]	-32.2722	6.8215	-4.7310	0.0000	-46.0032	-18.5413
vent[T.SUD]	-9.5071	8.9969	-1.0567	0.2962	-27.6169	8.6026

Le modèle estimé s'écrit :

$$\widehat{\texttt{O3}} = 103.85 - 25.56 \times 1_{\{\text{vent=NORD}\}} - 32.27 \times 1_{\{\text{vent=OUEST}\}} - 9.51 \times 1_{\{\text{vent=SUD}\}} \tag{6.5}$$

On calcule la statistique F :

```
# Test sur le coefficient de T12
import scipy.stats as stats

u1 = modelcomplet.mse_model - modelsanst12.mse_model
v1 = modelcomplet.mse_resid/modelcomplet.df_resid
fstats1 = u1/v1
print('f-statistic : %.4f'%(fstats1))
```

```
## f-statistic : 89.6262
```

Cette statistique F est le carré de la statistique de t de Student à $n-k-1$ dégres de liberté. Ainsi, le niveau de significativité du F est exactement celui du t.

```
# Statistique t de Student
tstats = np.sqrt(fstats1)
pvalue1 = 2*(1-stats.t.cdf(np.abs(tstats),df=modelcomplet.df_resid))
print("""t-statistic : %.4f \npvalue : %.4f"""%(tstats,pvalue1))
```

```
## t-statistic : 9.4671
## pvalue : 0.0000
```

On ne peut rejeter l'hypothèse H_0. La température à 12 h est significative.

6.1.2.2 Tests sur les variables qualitatives

On étudie sur le modèle (6.1) le test d'hypothèse :

$$\begin{cases} H_0 : \beta_2 = \beta_3 = \beta_4 = 0 \\ H_1 : \text{Au moins un de ces } \beta_j \neq 0 \end{cases}$$

On utilise la statistique :

$$F = \frac{1}{3}\frac{(\text{SCE(modèle complet) - SCE(modèle complet sauf vent)})}{\text{SCR(modèle complet)}/(n-k-1)} \tag{6.6}$$

Le chiffre 3 au dénominateur correspond au nombre de paramètres à tester.

```
# Modèle sans vent
modelsansvent = smf.ols(formula='O3~T12',data=donnee).fit()
coefsansvent = modelsansvent.summary2().tables[1]
```

Table 6.5 – Coefficients du modèle de régression sans vent

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	31.415	13.0584	2.4057	2e-02	5.1592	57.6707
T12	2.701	0.6266	4.3107	1e-04	1.4412	3.9609

Le modèle estimé s'écrit :

$$\widehat{\texttt{O3}} = 31.42 + 2.7 \times \texttt{T12} \tag{6.7}$$

On calcule la statistique F :

```
# Test sur vent
u2 = modelcomplet.mse_model - modelsansvent.mse_model
v2 = modelcomplet.mse_resid/modelcomplet.df_resid
fstats2 = u2/(3*v2)
pvalue2 = stats.f.cdf(fstats2,3,modelcomplet.df_resid)
print('f-statistic : %.4f \npvalue : %.4f'%(fstats2,pvalue2))
```

```
## f-statistic : -212.3169
## pvalue : 0.0000
```

La direction du vent est donc globalement significative.

6.2 Analyse de la covariance

6.2.1 Position du problème

Dans l'approche par analyse de la covariance, la démarche la plus naturelle consiste à effectuer K [1] régressions différentes, une pour chaque catégorie. Cela donne en termes de modélisation :

$$\texttt{O3}_i^j = \beta_0^j + \beta_1^j \texttt{T12}^j + \varepsilon_i^j, \quad i = 1, \dots, n_j \tag{6.8}$$

où n_j est le nombre total d'individus appartenant au groupe $j (j = \texttt{EST, NORD,OUEST,SUD})$. Trois cas de figures sont envisageables :

Cas 1 : La pente est identique d'une droite à l'autre

Dans ce cas, la température à 12 h intervient de la même façon d'une direction du vent à l'autre. Ce qui donne en terme de modélisation :

$$\texttt{O3}_i^j = \beta_0^j + \beta_1 \texttt{T12}^j + \varepsilon_i^j, \quad i = 1, \dots, n_j \tag{6.9}$$

Graphiquement, on a :

```
# Graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot([0,10],[0.1,.5],linestyle='--',color = 'red');
axe.plot([0,10],[0.25,0.65],color = 'blue');
axe.plot([0,10],[0.4,0.8],linestyle='--',color = 'green');
plt.annotate('b0=0.1', xy =(10.1,.5),xytext =(10.1,.5));
plt.annotate('b0=0.25', xy =(10.1,.65),xytext =(10.1,.65));
```

1. K est le nombre de modalités de la variable qualitative

```
plt.annotate('b0=0.4', xy =(10.1,.8),xytext =(10.1,.8));
axe.set(xlabel="x",ylabel="y",title="Pentes parallèles",xlim=[0,11],ylim=[0,1]);
plt.show()
```

```
## meta NOT subset; don't know how to subset; dropped
```

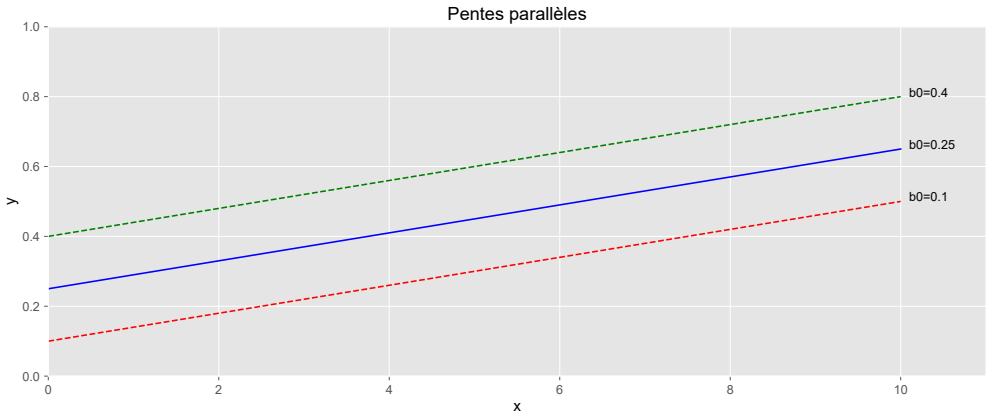


Figure 6.3 – Trois droites de régression fictives parallèles

Cas 2 : L'ordonnée à l'origine est la même d'une droite à l'autre

Dans ce cas, nous avons en termes de modélisation :

$$\texttt{03}_i^j = \beta_0 + \beta_1^j \texttt{T12}^j + \varepsilon_i^j, \quad i = 1, \dots, n_j \tag{6.10}$$

et graphiquement :

```
# Graphique
fig, axe = plt.subplots(figsize=(16,6))
axe.plot([0,10],[0.1,.5],linestyle='--',color = 'red');
axe.plot([0,10],[0.1,0.65],color = 'blue');
axe.plot([0,10],[0.1,0.8],linestyle='--',color = 'green');
plt.annotate('b1=1', xy =(10.1,.5),xytext =(10.1,.5));
plt.annotate('b1=2', xy =(10.1,.65),xytext =(10.1,.65));
plt.annotate('b1=3', xy =(10.1,.8),xytext =(10.1,.8));
plt.annotate('b0', xy =(0,.1),xytext =(-.3,.1));
axe.set(xlabel="x",ylabel="y",title="Ordonnée identique",xlim=[0,11],ylim=[0,1]);
plt.show()
```

```
## meta NOT subset; don't know how to subset; dropped
```

Dans le modèle (6.9), le coefficient β_1 est le même dans toutes les directions du vent. Cependant, si nous effectuons 4 régressions distinctes, comment trouverons-nous la même estimation de β_1 ? De même, dans le modèle (6.10), comment allons-nous procéder pour obtenir le même estimateur de β_0 dans chaque direction en effectuant 4 régressions distinctes ? Par conséquent, il semble raisonnable de n'effectuer qu'une seule régression.

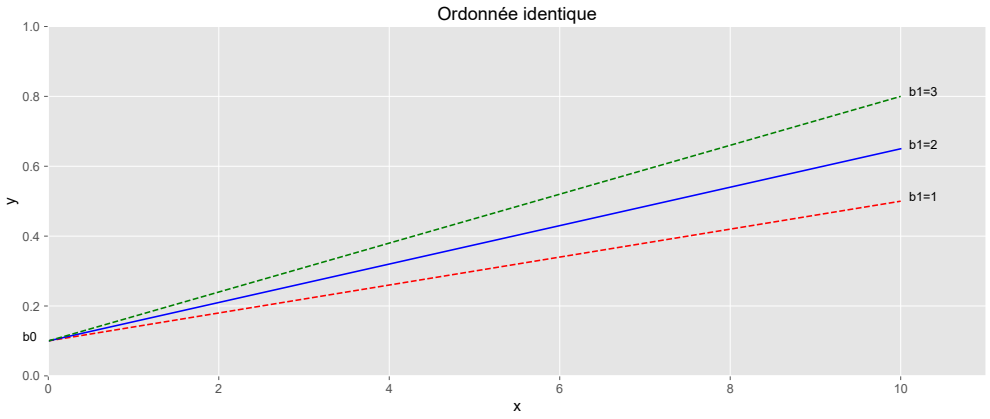


Figure 6.4 – Trois droites de régression fictives ayant la même ordonnée à l'origine

```
# Modèle par direction du vent
params = pd.DataFrame().astype("float")
for lab in np.unique(donnee["vent"]):
  model = smf.ols('O3~T12',data=donnee[donnee.vent==lab]).fit()
  summary = model.summary2().tables[1]
  params = pd.concat([params,summary],axis=0)
params = params.round(4).reset_index().rename(columns={"index":"params"})
```

Table 6.6 – Estimation des paramètres sur les 4 régressions

vent	params	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
EST	Intercept	45.6090	13.3549	3.4152	0.0042	16.9657	74.2523
	T12	2.7480	0.6078	4.5213	0.0005	1.4444	4.0516
NORD	Intercept	106.6345	30.6592	3.4781	0.0103	34.1370	179.1320
	T12	-1.6491	1.7562	-0.9390	0.3790	-5.8017	2.5036
OUEST	Intercept	64.6840	27.1131	2.3857	0.0298	7.2069	122.1612
	T12	0.3407	1.3267	0.2568	0.8006	-2.4717	3.1531
SUD	Intercept	-27.0602	13.8646	-1.9517	0.1084	-62.7002	8.5799
	T12	5.3786	0.6006	8.9550	0.0003	3.8347	6.9226

On voit que pour les directions du vent `NORD` et `OUEST`, le coefficient β_1 n'est pas significatif.

6.2.2 Hypothèse gaussienne

Sous l'hypothèse de normalité des résidus, nous pouvons tester toutes les hypothèses linéaires possibles. Un des principaux objectifs de l'analyse de la covariance est de savoir si les variables explicatives influent sur la variable à expliquer. Les deux tests à effectuer sont :

1. **Le test d'égalité des pentes**

$$\begin{cases} \mathrm{H}_0 : \beta_1^{\mathrm{EST}} = \beta_1^{\mathrm{NORD}} = \beta_1^{\mathrm{OUEST}} = \beta_1^{\mathrm{SUD}} = \beta_1 \\ \mathrm{H}_1 : \exists (i,j) : \beta_1^i \neq \beta_1^j \end{cases}$$

2. **Le test d'égalité des ordonnées à l'origine**

$$\begin{cases} \mathrm{H}_0 : \beta_0^{\mathrm{EST}} = \beta_0^{\mathrm{NORD}} = \beta_0^{\mathrm{OUEST}} = \beta_0^{\mathrm{SUD}} = \beta_0 \\ \mathrm{H}_1 : \exists (i,j) : \beta_0^i \neq \beta_0^j \end{cases}$$

La statistique de test vaut donc :

$$\mathrm{F} = \frac{||\widehat{Y} - \widehat{Y}_0||^2/(K-1)}{||Y - \widehat{Y}||^2/(n-K)} \tag{6.11}$$

L'hypothèse H_0 est rejetée en faveur de H_1 si l'observation de la statistique F est supérieure à $f_{K-1,n-K}(1-\alpha)$, la valeur α étant la probabilité de rejeter à tort H_0 ou erreur de première espèce et nous conclurons à l'effet du facteur explicatif.

Nous écrivons le modèle avec interaction :

```
# modèle avec interaction sans constante
modelinter = smf.ols(formula ='O3~vent+vent:T12-1',data=donnee).fit()
coefmodelinter = modelinter.summary2().tables[1]
```

Table 6.7 – Coefficients du modèle avec interaction sans constante

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
vent[EST]	45.6090	13.9343	3.2731	0.0021	17.4884	73.7295
vent[NORD]	106.6345	28.0341	3.8037	0.0005	50.0594	163.2095
vent[OUEST]	64.6840	24.6208	2.6272	0.0120	14.9972	114.3709
vent[SUD]	-27.0602	26.5389	-1.0196	0.3137	-80.6179	26.4976
vent[EST] :T12	2.7480	0.6342	4.3333	0.0001	1.4682	4.0278
vent[NORD] :T12	-1.6491	1.6058	-1.0269	0.3103	-4.8897	1.5916
vent[OUEST] :T12	0.3407	1.2047	0.2828	0.7787	-2.0905	2.7719
vent[SUD] :T12	5.3786	1.1497	4.6783	0.0000	3.0585	7.6988

Nous enlevons la constante en écrivant -1. Ensuite, il faut conserver une ordonnée à l'origine différente pour chacune des modalités du `vent`, ce qui est représenté par le facteur `vent` (ou une interaction de la variable 1 avec `vent`). Ensuite, nous ajoutons un coefficient directeur différent pour chacun des modalités du vent, ce qui est représenté par la variable `T12` en interaction avec `vent`. Cela donne :

Si dans l'écriture du modèle, la constante est conservée, le logiciel va prendre comme modalité de référence la première modalité (définie par ordre lexicographique). Cela donne :

```
# modèle avec interaction avec constante
modelinter2 = smf.ols(formula ='O3~vent+vent:T12',data=donnee).fit()
coefmodelinter2 = modelinter2.summary2().tables[1]
```

Les coefficients des ordonnées à l'origine sont des effets différentiels par rapport à la modalité de référence (`EST`), exemple $61.02 + 45.60 = 106.62$ valeur de `vent[EST]` dans l'écriture précédente.

Le modèle avec une seule pente peut s'écrire :

```
# Modèle avec une seule pente
model2 = smf.ols(formula='O3~vent+T12',data=donnee).fit()
model2b = smf.ols(formula='O3~-1+vent+T12',data=donnee).fit()
```

Table 6.8 – Coefficients du modèle avec interaction avec constante

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	45.6090	13.9343	3.2731	0.0021	17.4884	73.7295
vent[T.NORD]	61.0255	31.3061	1.9493	0.0580	-2.1528	124.2038
vent[T.OUEST]	19.0751	28.2905	0.6743	0.5038	-38.0174	76.1675
vent[T.SUD]	-72.6691	29.9746	-2.4244	0.0197	-133.1604	-12.1779
vent[EST] :T12	2.7480	0.6342	4.3333	0.0001	1.4682	4.0278
vent[NORD] :T12	-1.6491	1.6058	-1.0269	0.3103	-4.8897	1.5916
vent[OUEST] :T12	0.3407	1.2047	0.2828	0.7787	-2.0905	2.7719
vent[SUD] :T12	5.3786	1.1497	4.6783	0.0000	3.0585	7.6988

Le modèle avec une seule ordonnée à l'origine peut s'écrire :

```
# Modèle avec une seule ordonnée à l'origine
model3 = smf.ols(formula='O3~vent:T12',data=donnee).fit()
```

A présent, choisissons la meilleure modélisation :

— **égalité des pentes**

Nous effectuons un test entre les modèles `modelinter2` et `model2`.

```
# Test d'égalité des pentes
import statsmodels.api as sm
anova1 = sm.stats.anova_lm(model2,modelinter2,test="F", typ="I")
```

Table 6.9 – Test d'égalité des pentes

df_resid	ssr	df_diff	ss_diff	F	Pr(>F)
45	12611.951	0			
42	9087.431	3	3524.521	5.4298	0.003

nous concluons donc à l'effet du vent sur les pentes. Nous aurions obtenu les mêmes résultats avec `model2b` contre `modelinter2` ou `model2` contre `model2b` ou encore `model2b` contre `modelinter`.

— **égalités des ordonnées à l'origine**

Nous effectuons un test entre le modèle `model3` et le modèle `modelinter2`.

```
# Test d'égalité des ordonnées à l'origine
anova2 = sm.stats.anova_lm(model3,modelinter2,test="F", typ="I")
```

Table 6.10 – Test d'égalité des ordonnées à l'origine

df_resid	ssr	df_diff	ss_diff	F	Pr(>F)
45	11864.057	0			
42	9087.431	3	2776.626	4.2776	0.0101

Nous concluons donc à l'effet du vent sur les ordonnées à l'origine.

Enfin le graphique de résidus (*cf.* Figure 6.5) obtenu avec

```
# Graphique des résidus
from statsmodels.stats.outliers_influence import OLSInfluence

influ = OLSInfluence(model2)
df = pd.concat([model2.fittedvalues,influ.resid_studentized_external],axis=1)
df.columns = ["y_hat","student_resid"]
p = (ggplot(df,aes(x="y_hat",y="student_resid"))+geom_point()+
    labs(x="$\widehat{Y}$",y="résidus"))
print(p)
```

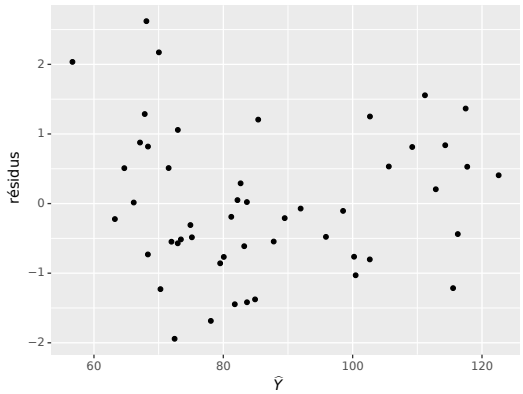


Figure 6.5 – Résidus studentisés

ne fait apparaître ni structure ni point aberrant.

Par contre, si on analyse la structure des résidus par modalité de `Vent` grâce à la commande :

```
# Structure des résidus par modalité de vent
df["vent"] = donnee["vent"]
p = (ggplot(df,aes(x="y_hat",y="student_resid"))+geom_point()+
      facet_wrap("~vent")+labs(x="$\widehat{Y}$",y="résidus"))
print(p)
```

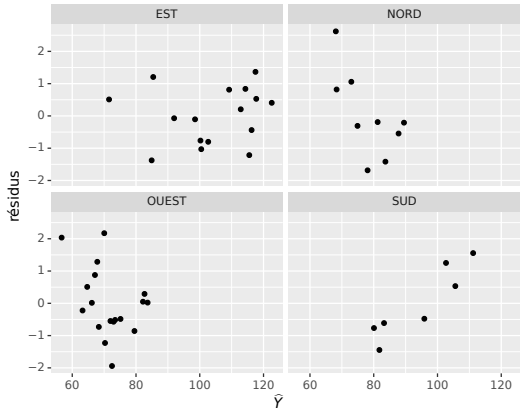


Figure 6.6 – Résidus studentisés par niveau de vent.

on constate une structuration des résidus pour la modalité SUD. Cependant cette structuration n'est constatée qu'avec 7 individus, ce qui semble trop peu pour que cette conclusion soit fiable.

Remarque 6.1

Pour cet exemple, nous conservons dont le modèle complet.

```
# modèle complet
model0 = smf.ols(formula='O3~vent+T12+vent:T12',data=donnee).fit()
coefmodel0 = model0.summary2().tables[1]
```

Table 6.11 – Coefficients du modèle complet avec interaction

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	45.6090	13.9343	3.2731	0.0021	17.4884	73.7295
vent[T.NORD]	61.0255	31.3061	1.9493	0.0580	-2.1528	124.2038
vent[T.OUEST]	19.0751	28.2905	0.6743	0.5038	-38.0174	76.1675
vent[T.SUD]	-72.6691	29.9746	-2.4244	0.0197	-133.1604	-12.1779
T12	2.7480	0.6342	4.3333	0.0001	1.4682	4.0278
vent[T.NORD] :T12	-4.3971	1.7265	-2.5468	0.0146	-7.8813	-0.9129
vent[T.OUEST] :T12	-2.4073	1.3614	-1.7682	0.0843	-5.1548	0.3402
vent[T.SUD] :T12	2.6306	1.3130	2.0035	0.0516	-0.0191	5.2803

`Intercept` et `T12` sont bien les valeurs de l'ordonnée à l'origine et de la pente pour le vent d'EST.

Remarque 6.2

Dans le modèle complet, nous utilisons bien les 3 variables `vent`, `T12` et leur interaction. En effet, en écrivant ainsi, le logiciel, pourra impose des contraintes lors de l'inversion de la matrice X.

7 Analyse de la variance

Sommaire

En statistique, l'analyse de la variance (ANOVA, *ANalysis Of VAriance*) est un ensemble de méthodes statistiques utilisées pour vérifier sir les moyennes des groupes proviennent d'une même population. Les groupes cooréspondent aux modalités d'une variable qualitative et les moyennes sont calculées à partir d'une variable continue.

Ce test s'applique lorsque l'on mesure une ou plusieurs variables explicatives catégorielles (appelées alors faceturs de variabilité, leurs différentes modalités étant parfois appelées « niveaux ») qui ont de l'influence sur la loi d'une variable continue à expliquer. On parle d'analyse à un facteur lorsque l'analyse porte sur un modèle décrit par un seul facteur de variabilité, d'analyse à deux facteurs ou d'analyse multifactorielle sinon.

7.1 Exemple introductif

On dispose de 30 chambres de même type avec leur frais de location. On dispose également des quartiers pour les chambres : A, B et C. Les frais de location des chambres se trouvent dans le tableau ci-après :

Table 7.1 – Frais de location par quartier

Quartier	Frais de location
A	20, 15, 21, 20, 15, 16, 18, 17, 15, 12, 21, 20
B	50, 60, 65, 45, 40, 70, 65, 30
C	35, 35, 32, 40, 45, 25, 20, 21, 30, 29

On cherche à expliquer le loyer (variable quantitative) par le type de quartier (variable qualitative).

```
# Create dataset
A = [20, 15, 21, 20, 15, 16, 18, 17, 15, 12, 21, 20]
B = [50, 60, 65, 45, 40, 70, 65, 30]
C = [35, 35, 32, 40, 45, 25, 20, 21, 30, 29]
```

Nos listes créées n'ont pas la même longueur.

```
# Longueur des listes
import pandas as pd
longueur=pd.DataFrame({"A":len(A),"B":len(B),"C":len(C)},index=["Effectif"])
```

Table 7.2 – Nombres d'observations par quartier

	A	B	C
Effectif	12	8	10

Nous créons notre variable quartier qui contiendra le type de quartier du logement.

```
# Regroupement des quartiers
q_A, q_B, q_C = ['A']*len(A), ['B']*len(B), ['C']*len(C)

# Somme
frais, quartier = A + B + C, q_A + q_B + q_C
```

On crée notre dataframe :

```
# création du dataframe
df = pd.DataFrame({'quartier': quartier, 'montant': frais})
```

Table 7.3 – Données de location

quartier	montant
A	20
A	15
A	21
A	20
A	15
A	16
A	18
A	17
A	15
A	12
A	21
A	20
B	50
B	60
B	65
B	45
B	40
B	70
B	65
B	30
C	35
C	35
C	32
C	40
C	45
C	25
C	20
C	21
C	30
C	29

On visualise le boxplot de notre variable par quartier.

```
# Représentation graphique - Box plot
from plotnine import *
p = (ggplot(df,aes(x="quartier",y="montant",color="quartier"))+
     geom_boxplot(size=1,outlier_shape=1,outlier_color="black",outlier_size=3)+
     geom_jitter(alpha=0.7,size=1.5)+
     labs(title='Boxplot des frais de location par quartier')+
     theme(legend_position=(0.2,0.6),legend_direction="vertical"))
print(p)
```

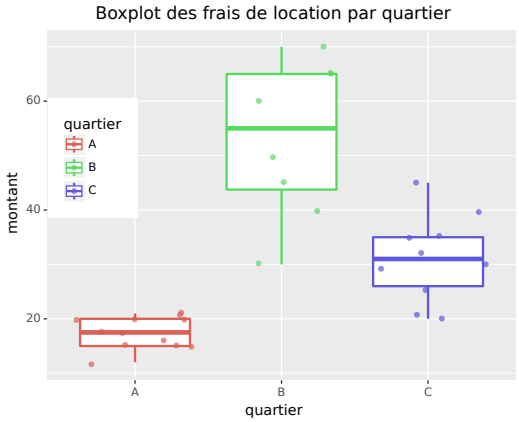


Figure 7.1 – Boxplot des frais de location par quartier

7.2 Le modèle d'analyse de la variance à 1 facteur

7.2.1 Notations

On appellera :

- y : la variable quantitative ;
- y_{ik} : la valeur de la variable y pour l'observation i du groupe k ;
- K : le nombre total de groupe ;
- n_k : le nombre d'observations dans le groupe k ;
- n : le nombre total d'observations. Il correspond à $n = \sum_{k=1}^{K} n_k$

On a les indicateurs de tendances centrales suivantes :

7.2.1.1 Moyenne par groupes

La moyenne par groupe est calculée avec la formule :

$$\overline{y}_k = \frac{1}{n_k} \sum_{i=1}^{i=n_k} y_{ik}, \quad \forall k \tag{7.1}$$

C'est la moyenne de y dans le groupe k.

```
# Moyenne par quartier
import numpy as np
mean = df.groupby("quartier").mean().T
```

Table 7.4 – Frais moyen de location par quartier

	A	B	C
montant	17.5	53.125	31.2

7.2.1.2 Moyenne générale

La moyenne générale ou moyenne globale est calculée avec la formule ci-après :

$$\overline{y} = \frac{1}{n} \sum_{k=1}^{k=K} \sum_{i=1}^{i=n_k} y_{ik} \tag{7.2}$$

Pour nos données, on a :

```
# Moyenne générale
mean.loc[:,'gen. mean'] = np.mean(df['montant'])
```

Table 7.5 – Frais moyen de location par quartier et général

	A	B	C	gen. mean
montant	17.5	53.125	31.2	31.5667

Propriété 7.1 *La moyenne globale est une moyenne des moyennes des groupes pondérées par leurs effectifs. En effet, on a :*

$$\overline{y} = \frac{1}{n} \sum_{k=1}^{k=K} n_k \overline{y}_k$$

On calcule également les indicateurs de dispersion.

7.2.1.3 Variance du groupe k

La variance du groupe k est calculée comme suit :

$$\sigma_k^2(y) = \frac{1}{n_k} \sum_{i=1}^{i=n_k} \left(y_{ik} - \overline{y}_k\right)^2 \quad \forall k \tag{7.3}$$

C'est la variance de y à l'intérieur du groupe k.

```
# Variance par groupe
variance = df.groupby("quartier").var(ddof=0).T
```

Table 7.6 – Variance par quartier

	A	B	C
montant	7.9167	174.6094	57.16

7.2.1.4 Variance globale

Sa formule est :

$$\sigma^2(y) = \frac{1}{n} \sum_{k=1}^{k=K} \sum_{i=1}^{i=n_k} (y_{ik} - \overline{y})^2 \tag{7.4}$$

```
# Variance globale
variance.loc[:,'gen. variance'] = np.var(df['montant'])
```

Table 7.7 – Variance par quartier et générale

	A	B	C	gen. variance
montant	7.9167	174.6094	57.16	271.9122

7.2.2 Le modèle et ses hypothèses

Le modèle ANOVA à 1 facteur s'écrit simplement :

$$y_{ik} = \alpha + \beta_k + \varepsilon_{ik} \tag{7.5}$$

Autrement dit, le montant du loyer est la resultante de trois effets :

— Un effet commun à tous les chambres : α
— Un effet spécifique au groupe et commun aux individus du même groupe : β_k
— Un effet de perturbation propre à chaque individu : ε_{ik}

Remarque 7.1

Le modèle a trop de paramètres inconnus pour être estimable de manière bien déterminée. On rajoute donc la contrainte suivante :

$$\sum_{k=1}^{k=K} \frac{n_k}{n} \beta_k = 0 \tag{7.6}$$

C'est-à-dire que les effets spécifiques sont en moyenne nuls. Ils sont vus comme des effets différentiels à l'effet global.

Les ε_{ik} sont considérés comme des perturbations aléatoires. Ils matérialisent l'effet éventuel de tous les facteurs influençant y que l'on n'a pas pris en compte dans le modèle.

Hypothèse 7.1 (Hypothèses du modèle) *On suppose que les ε_{ik} sont des aléas de moyenne nulle, indépendants et identiquement distribués et supposés gaussien.*

7.2.3 Les estimateurs du modèle

Les estimateurs des différents modèles sont calculés de la manière suivante :

- Effet commun : $\hat{\alpha} = \overline{y}$;
- Effet spécifique du groupe k : $\hat{b}_k = \hat{y}_k - \overline{y}$;
- Perturbation : $\hat{\varepsilon}_{ik} = y_{ik} - \overline{y}_k$

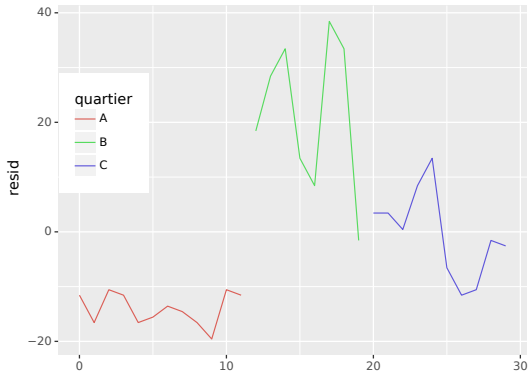
```
# Estimation des paramètres
def anova_params(df,x,y):
  # Common effet
  common_effet = df[y].mean()
  # Specific effet
  specific_effet = df.groupby(x).mean().apply(lambda x : x-common_effet,axis=0).T
  # Resid
  resid = df[y] - common_effet
  # Store all result
  res = specific_effet.copy()
  # Insert commmon effet
  res.insert(0,"alpha",common_effet)
  return res,resid.to_frame("resid")

# Application
aov_params, resid = anova_params(df,x="quartier",y="montant")
```

Table 7.8 – Paramètres estimés

	alpha	betaA	betaB	betaC
montant	31.5667	-14.0667	21.5583	-0.3667

On visualise les résidus :

```
# Représentation des résidus
print((ggplot(resid,aes(x=resid.index,y="resid",color=df.quartier))+
       geom_line()+
       theme(legend_position=(0.2,0.6),legend_direction="vertical")))
```



7.2.4 Équation de décomposition de la variance

Elle consiste à écrire la variance totale sous forme de variance intra groupe et variance inter groupe.

$$
\begin{aligned}
\sigma^2(y) &= \frac{1}{n}\sum_{k=1}^{k=K}\sum_{i=1}^{i=n_k}(y_{ik}-\overline{y})^2 = \frac{1}{n}\sum_{k=1}^{k=K}\sum_{i=1}^{i=n_k}\left[(y_{ik}-\overline{y}_k)+(\overline{y}_k-\overline{y})\right]^2 \\
&= \frac{1}{n}\sum_{k=1}^{k=K}\sum_{i=1}^{i=n_k}\left[(y_{ik}-\overline{y}_k)^2+(\overline{y}_k-\overline{y})^2+2\,(y_{ik}-\overline{y}_k)(\overline{y}_k-\overline{y})\right]^2 \\
&= \frac{1}{n}\sum_{k=1}^{k=K}\sum_{i=1}^{i=n_k}(y_{ik}-\overline{y}_k)^2+\frac{1}{n}\sum_{k=1}^{k=K}\sum_{i=1}^{i=n_k}(\overline{y}_k-\overline{y})^2+\frac{2}{n}\sum_{k=1}^{K}\sum_{i=1}^{i=n_k}(y_{ik}-\overline{y}_k)(\overline{y}_k-\overline{y}) \\
&= \frac{1}{n}\sum_{k=1}^{k=K}n_k\left(\frac{1}{n_k}\sum_{i=1}^{i=n_k}(y_{ik}-\overline{y}_k)^2\right)+\frac{1}{n}\sum_{k=1}^{k=K}n_k\,(\overline{y}_k-\overline{y})^2 \\
&\quad +\frac{2}{n}\sum_{k=1}^{k=K}(\overline{y}_k-\overline{y})\underbrace{\left[\sum_{i=1}^{i=n_k}(y_{ik}-\overline{y}_k)\right]}_{=0} \\
&= \frac{1}{n}\sum_{k=1}^{k=K}n_k\sigma_k^2(y)+\frac{1}{n}\sum_{k=1}^{k=K}n_k(\overline{y}_k-\overline{y})^2
\end{aligned}
$$

On fait la remarque suivante :

- $\frac{1}{n}\sum_{k=1}^{k=K}n_k\sigma_k^2(y)$ est la moyenne des variances internes aux groupes pondérées par leurs effectifs. Il s'agit donc d'une variance interne moyenne ou **variance dans les groupes**.
- $\frac{1}{n}\sum_{k=1}^{k=K}n_k\,(\overline{y}_k-\overline{y})^2$ est la variance des moyennes des différents groupes, soit une **variance entre les groupes**.

Ainsi, on a :

$$\textbf{variance totale = variance intra groupe + variance inter groupe} \tag{7.7}$$

où :

— **La variance intra** représente la moyenne des variances (conditionnelles) et quantifie la part de variabilité intrinsèque de y : $\sigma^2_{\text{intra}}(y) = \frac{1}{n}\sum_{k=1}^{k=K} n_k \sigma^2_k(y)$

— **La variance inter** représente la variance des moyennes (conditionnelles) et mesure l'hétérogénéité des populations : $\sigma^2_{\text{inter}}(y) = \frac{1}{n}\sum_{k=1}^{k=K} n_k \left(\overline{y}_k - \overline{y}\right)^2$

```
# Equation de décomposition de la variance
def variance_decomposition(categories, values):
    cat = np.unique(categories, return_inverse=True)[1]
    values = np.array(values)

    ssw = 0
    ssb = 0
    for i in np.unique(cat):
        subgroup = values[np.argwhere(cat == i).flatten()]
        ssw += np.sum((subgroup-np.mean(subgroup))**2)
        ssb += len(subgroup)*(np.mean(subgroup)-np.mean(values))**2

    return ssw/values.shape[0], ssb/values.shape[0]
ssw, ssb = variance_decomposition(df['quartier'], df['montant'])
res_var = pd.DataFrame({"intra groupe":ssw,"inter groupe":ssb,"general":ssw+ssb},
                       index=["variance"])
```

Table 7.9 – Décomposition de la variance

	intra groupe	inter groupe	general
variance	68.7825	203.1297	271.9122

7.2.5 Le critère et le test

L'idée est la suivante :

— Si le rapport $\dfrac{\text{variance inter}}{\text{variance intra}}$ est **grand**, on admet la significativité de l'influence dépisté du facteur groupe sur la variable y.

7.2.5.1 Formulation du test

Les hypothèses du test sont :

$$\begin{cases} H_0 & : \text{l'influence du groupe n'est pas significative} \\ H_1 & : \text{l'influence du groupe est significative} \end{cases}$$

Sous l'hypothèse que le facteur de groupe n'induit aucune différence sur y entre les groupes (c'est-à-dire *aucun effet spécifique*, autrement dit $\beta_k = 0$), la statistique de test est :

$$F_{stat} = \frac{\frac{1}{K-1} \times \text{variance inter}}{\frac{1}{n-K} \times \text{variance intra}} \tag{7.8}$$

Cette statistique suit une loi de Fisher à $K-1$ et $n-K$ degrés de liberté noté $F(K-1, n-K)$. On compare cette statistique F au fractile d'ordre 95% de la loi de Fisher $\mathcal{F}_{0.96}(K-1, n-K)$. On rejette l'hypothèse nulle si F_{stats} est supérieure à la statistique de Fisher lue à $K-1$ et $n-K$ degrés de liberté.

Le tableau 7.10 donne un résumé de l'analyse de variance pour un facteur.

Table 7.10 – Équation de décomposition de la variance

Source de variation	degré de liberté	Somme des carrés	Carré moyen	Stat. f	Significativité
Inter	$K-1$	$n \times \sigma^2_{\text{inter}}(y)$	$\frac{n \times \sigma^2_{\text{inter}}(y)}{K-1}$	F_{stat}	$pvalue$
Intra	$n-K$	$n \times \sigma^2_{\text{intra}}(y)$	$\frac{n \times \sigma^2_{\text{intra}}(y)}{n-K}$		
Totale	$nK-1$	$n \times \sigma^2(y)$			

```
# Calcul de F stat
n, K= df.shape[0], len(np.unique(df['quartier']))

# numérateur et dénominateur et F-stat
num, den = ssb/(K-1), ssw/(n-K)
f_stat = num/den

# calcul de la pvalue
import scipy.stats as st
ddl1, ddl2 = K-1, n-K
pvalue = st.f.sf(f_stat, ddl1, ddl2)
# Stockage des résultats
res_aov = pd.DataFrame({"sum sq intra":n*ssw,"sum sq inter":n*ssb,
                        "statistic":f_stat,"ddl num" : ddl1,"ddl denom":ddl2,
                        "pvalue":pvalue},index=["Fisher test"])
```

Table 7.11 – Test de Fisher - Analyse de la variance

	sum sq intra	sum sq inter	statistic	ddl num	ddl denom	pvalue
Fisher test	2063.475	6093.892	39.8684	2	27	0

Nous avons une pvalue significative ($p < 0.05$). En conclusion, il y a une différence significative entre les quartiers.

7.2.5.2 Mise en oeuvre avec statsmodels

Statsmodels dispose d'une fonction anova_lm pour réaliser une analyse de la variance.

```
# get ANOVA table as R like output
import statsmodels.api as sm
from statsmodels.formula.api import ols

# Ordinary Least Squares (OLS) model
model = ols('montant ~ quartier', data=df).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
```

Table 7.12 – Analyse de la variance - Statsmodels

	sum_sq	df	F	PR(>F)
quartier	6093.892	2	39.8684	0
Residual	2063.475	27		

Statsmodels nous renvoie la table réduite identique à celle du tableau 7.10. On voit que les résultats sont identiques à ceux calculés manuellement.

Bibliographie

Bartels, Robert. 1982. “The Rank Version of von Neumann’s Ratio Test for Randomness.” *Journal of the American Statistical Association* 77 (377) : 40–46.

Bourbonnais, Régis et al. 2021. *Econométrie*. Dunod.

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.

Bradley, James V. 1968. “Distribution-Free Statistical Tests.”

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.

Cornillon, Pierre-André, Nicolas Hengartner, Eric Matzner-Løber, and Laurent Rouvière. 2019. *Régression Avec r-2e édition*. EDP sciences.

Cromwell, Jeff B, and Michel Terraza. 1994. *Multivariate Tests for Time Series Models*. 100. Sage.

Danioko, Sidy, Jianwei Zheng, Kyle Anderson, Alexander Barrett, and Cyril S Rakovski. 2022. “A Novel Correction for the Adjusted Box-Pierce Test.”

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.

Farrar, Donald E, and Robert R Glauber. 1967. “Multicollinearity in Regression Analysis : The Problem Revisited.” *The Review of Economic and Statistics*, 92–107.

Fisher, Thomas J. 2011. “Testing Adequacy of Arma Models Using a Weighted Portmanteau Test on the Residual Autocorrelations.” *Contributed Paper* 327 : 2011.

Heiss, Florian, and Daniel Brunner. 2020. *Using Python for Introductory Econometrics*. Independently published New York.

Jack, Johnston, and Dinardo John. 1999. “Méthodes économétriques.” *Éd ECONOMICA, 4ème éd.*

Klein, Lawrence R, and Mitsugu Nakamura. 1962. “Singularity in the Equation Systems of Econometrics : Some Aspects of the Problem of Multicollinearity.” *International Economic Review* 3 (3) : 274–99.

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.

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.

Peña, Daniel, and Julio Rodriguez. 2002. "A Powerful Portmanteau Test of Lack of Fit for Time Series." *Journal of the American Statistical Association* 97 (458) : 601–10.

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.

Sheppard, Kevin. 2014. "Introduction to Python for Econometrics, Statistics and Data Analysis." University of Oxford.

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.

Econométrie du modèle linéaire :

Théorie et application sous Python

Le modèle linéaire est souvent le premier outil de statistique inférentielle mis en œuvre car il suit après l'étude descriptive des données. Cet ouvrage présente les bases de l'économétrie du modèle linéaire. Il met l'accent sur les méthodes et les compétences indispensables à tout étudiant ou tout praticien désireux d'appliquer le modèle linéaire.

Ce manuel s'adresse aux étudiants de Licence et Master de Mathématiques, de Mathématiques appliquées, d'Economie et d'Econométrie, ainsi qu'aux élèves des écoles d'ingénieurs. Il s'adresse également aux professionnels, praticiens de l'économétrie du modèle linéaire.

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 en France et Ingénieur Statisticien Economiste (ISE) de l'école nationale de la statistique et de l'analyse économique (ENSAE) de Dakar au Sénégal. Par ailleurs, il est titulaire d'un master recherche en économie quantitative obtenu à la faculté des sciences économiques et de gestion appliquée de l'Université de Douala au Cameroun. 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.