



# Analyse et Prévision d'une série chronologique avec Python

A Comprehensive Analysis and Implementation

**Daoudi Adam**

Travail encadré par: Mme.Fadoua Badaoui

Data Science Student

INSEA

[addaoudi@insea.ac.ma](mailto:addaoudi@insea.ac.ma)

May 28, 2025

## Abstract

This report presents a comprehensive analysis of the monthly beer production time series from 1956 to 1995. The project aims to model and forecast seasonal patterns using time series techniques such as AR, ARIMA, and SARIMAX models. After confirming stationarity via the ADF test, a SARIMAX(1, 1, 1)(1, 1, 1, 12) model was developed to capture annual seasonality. Diagnostic tests revealed white noise residuals (Ljung-Box, p-value = 0.49), but non-normal distribution (Jarque-Bera, p-value = 0.00), with significant heteroscedasticity. Performance metrics indicate a RMSE of 13.29 and a MAPE of 6.65%, reflecting an overall acceptable accuracy. The SARIMAX forecasts closely follow actual data in the short term but tend to underestimate long-term variations. These results highlight the model's effectiveness in capturing seasonality, while also identifying areas for improvement such as data transformation or incorporating exogenous variables.

**Keywords:** time series, AR, ARIMA, SARIMAX, modeling, forecasting, seasonal analysis, performance

Ce rapport propose une analyse approfondie de la série chronologique de la production mensuelle de bière entre 1956 et 1995. L'objectif principal du projet est de modéliser et de prévoir les tendances saisonnières à l'aide de techniques de séries temporelles. Grâce à une méthodologie basée sur les modèles AR, ARIMA et SARIMAX, un modèle SARIMAX(1, 1, 1)(1, 1, 1, 12) a été développé après vérification de la stationnarité via le test ADF. Les résultats obtenus montrent des résidus non autocorrélés (test de Ljung-Box, p-value = 0.49), mais non normalement distribués (test de Jarque-Bera, p-value = 0.00), avec une hétéroscédasticité significative. Les performances du modèle, évaluées à l'aide d'indicateurs tels que la RMSE (13.29) et le MAPE (6.65%), témoignent d'une précision globalement satisfaisante. L'implémentation montre une bonne capacité à suivre les données réelles à court terme, tout en sous-estimant certaines fluctuations à long terme. Ce travail contribue à l'analyse de séries temporelles saisonnières en mettant en lumière la pertinence des modèles SARIMAX, et suggère des améliorations potentielles telles que la transformation des données ou l'ajout de variables exogènes.

**mots clés:** séries temporelles, AR, ARIMA, SARIMAX, modélisation, prévision, analyse saisonnière, performance

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Description . . . . .	7
1.2	Problématique . . . . .	7
1.3	Objectifs . . . . .	7
1.4	Limites . . . . .	8
<b>2</b>	<b>Methodology</b>	<b>8</b>
2.1	Aperçu de l'approche . . . . .	8
2.2	Stratégie d'implémentation . . . . .	9
<b>3</b>	<b>Partie 1 : Premier contact et préparation de notre dataframe</b>	<b>9</b>
3.1	Importation de bibliothéques . . . . .	9
<b>4</b>	<b>Partie 2 : Composants de notre série chronologique</b>	<b>12</b>
4.1	Composante d'une serie chronologique . . . . .	12
4.2	Modélisation de série chronologique . . . . .	12
4.3	Étude de la tendance . . . . .	14
4.3.1	Algorithme d'ajustement de la tendance par regression linéaire . . . . .	14
4.4	Étude de la saisonnalité: . . . . .	15
4.5	Études de la stationnarité : . . . . .	16
4.5.1	Formulation mathématique du test de Dickey Fuller . . . . .	16
4.5.2	Test de Dickey-Fuller:Algorithme et implmentation . . . . .	17
<b>5</b>	<b>Partie 4 : Étude de l'autocorrélation de notre série chronologique</b>	<b>18</b>
5.1	Autocorrélation dans les séries chronologiques . . . . .	18
5.1.1	ACF-Autorrelation: . . . . .	18
5.1.2	PACF-Autocorrelation partielle: . . . . .	19
<b>6</b>	<b>Partie 5 : Estimation des modèles</b>	<b>20</b>
6.1	Modèle (AR) . . . . .	20
6.1.1	Prévision à l'aide de AR . . . . .	21
6.2	Modèle ARIMA . . . . .	21
6.2.1	Comparaison entre AR et ARIMA . . . . .	22
6.3	Modèle SARIMAX . . . . .	23
6.3.1	Commentaires: . . . . .	24
6.3.2	Prévision a l'aide de SARIMAX . . . . .	26
6.4	Meilleur modele SARIMAX avec Cross-Validation . . . . .	26
6.4.1	Commentaires . . . . .	28

6.4.2 Prévision a l'aide de SARIMAX(Cross-Validation) . . . . .	29
<b>7 Résumé du projet</b>	<b>30</b>
<b>A Installation Instructions</b>	<b>30</b>

## List of Figures

1	Production mensuelle de bière en Australie . . . . .	12
2	Décomposition de la série chronologique. . . . .	14
3	Ajustement de la tendance par regression linéaire. . . . .	15
4	La composante saisonnière entre 1970 et 1975 . . . . .	15
5	Fonction d'autocorrélation . . . . .	19
6	Fonction d'autocorrélation partielle . . . . .	20
7	Prévision avec AR . . . . .	21
8	Prévision avec modèle ARIMA . . . . .	22
9	Résumé du modèle SARIMAX . . . . .	24
10	Histogramme et QQ plot des résidus . . . . .	25
11	ACF et résidus du modele . . . . .	25
12	Prévision avec modèle SARIMAX . . . . .	26
13	Prévision avec modèle SARIMAX . . . . .	26
14	Résumé du modèle . . . . .	28
15	Prévision avec modèle SARIMAX(Cross-Validation) . . . . .	29
16	Histogramme et QQ plot des residus du modele ameliorer . . . . .	29

## List of Tables

1	Structure du dataframe . . . . .	10
2	Statistiques descriptives . . . . .	11
3	Résultats des tests de stationnarité . . . . .	17
4	Métadonnées et performance du modèle SARIMAX . . . . .	30

# 1 Introduction

## 1.1 Description

Ce projet se consacre à l'analyse et à la prévision de la production mensuelle de bière en Australie en s'appuyant sur la méthodologie *Box-Jenkins*, à partir des données issues du fichier ‘`monthly-beer-production-in-austr.csv`’. Après une préparation incluant l'interpolation des valeurs manquantes et une répartition en ensembles d'apprentissage (80 %) et de test (20 %), la série révèle une tendance à la hausse ainsi qu'une saisonnalité annuelle. La stationnarité est validée par le test ADF, et divers modèles de séries temporelles sont explorés. Parmi eux, le modèle sélectionné offre les meilleures performances, avec des métriques d'erreur (RMSE, MAE, MAPE) optimales. Les résidus présentent une normalité et un caractère de bruit blanc, confirmant la robustesse de l'approche. Les prévisions reflètent efficacement la saisonnalité, mettant en évidence l'importance d'un ajustement minutieux. Des investigations futures pourraient envisager l'intégration de facteurs additionnels, tels que les prix des matières premières.

## 1.2 Problématique

Comment analyser, implémenter et réaliser une étude approfondie de la série chronologique de la production mensuelle de bière en Australie, tout en résolvant efficacement le problème de prévision posé par les variations saisonnières, économiques et climatiques, afin d'optimiser la planification de la production pour les brasseurs ?

## 1.3 Objectifs

1. Objectif principal: Réaliser une analyse complète et une implémentation d'un modèle de série chronologique pour prévoir avec précision la production mensuelle de bière en Australie, en utilisant la méthodologie Box-Jenkins.
2. Objectifs secondaires :
  - **Découpage et étude des composantes** : Diviser la série en ses composantes principales (tendance, saisonnalité, résidus) à travers une décomposition pour en analyser les caractéristiques.
  - **Détermination de la stationnarité** : Évaluer la stationnarité de la série à l'aide de tests statistiques (comme le test ADF) pour préparer la modélisation.
  - **Estimation des modèles** : Développer et ajuster différents modèles de séries temporelles (AR, ARIMA, SARIMAX) pour identifier les approches les plus adaptées.

- **Tests mathématiques** : Réaliser des tests statistiques (par exemple, Shapiro-Wilk pour la normalité des résidus, Ljung-Box pour le blanchiment) afin de valider la qualité des modèles.
- **Prévisions et conclusion** : Générer des prévisions sur l'échantillon de test, visualiser les résultats, et tirer des conclusions pour guider la planification de la production.

## 1.4 Limites

Ce projet présente les limitations suivantes :

- **Manque de variables exogènes** : L'absence de données réelles sur des influences externes (prix des matières premières, climat, événements culturels) limite la précision du modèle.
- **Insuffisance des modèles prédéfinis** : Les modèles disponibles dans les bibliothèques (ARIMA, SARIMAX) peuvent ne pas être suffisamment flexibles pour capturer des non-linéarités ou des changements structurels.
- **Non-normalité des résidus** : Les résidus peuvent ne pas suivre une distribution normale, ce qui pourrait compromettre la fiabilité des prévisions.

## 2 Methodology

### 2.1 Aperçu de l'approche

L'approche globale adoptée pour résoudre le problème consiste à appliquer la méthodologie Box-Jenkins, une méthode statistique éprouvée pour l'analyse et la modélisation des séries temporelles. Cette méthodologie a été choisie pour sa capacité à structurer l'analyse en étapes claires : identification, estimation, et validation des modèles, permettant ainsi une compréhension approfondie des dynamiques sous-jacentes de la production mensuelle de bière en Australie. Elle offre un cadre systématique pour traiter les tendances, la saisonnalité et les irrégularités, en s'appuyant sur des tests statistiques comme le test ADF pour la stationnarité et des outils comme la décomposition saisonnière. Cette approche a été préférée en raison de sa flexibilité pour ajuster des modèles tels que ARIMA ou SARIMA, et de sa robustesse dans la prévision, ce qui est essentiel pour optimiser la planification dans le contexte de l'industrie brassicole. De plus, son caractère itératif permet d'affiner les modèles en fonction des résultats obtenus, garantissant une solution adaptée aux données disponibles.

## 2.2 Stratégie d'implémentation

- **Environnement de développement** : L'analyse et la modélisation ont été réalisées dans un environnement Jupyter Notebook, qui permet une exécution interactive et une visualisation immédiate des résultats. Python (version 3.12.3) a été utilisé comme langage principal, en raison de sa richesse en bibliothèques pour l'analyse des séries temporelles.
- **Outils et technologies utilisés** : Plusieurs bibliothèques Python ont été mobilisées pour ce projet. Pandas et NumPy ont servi à la manipulation et au traitement des données, Matplotlib et Seaborn à la visualisation, et Statsmodels à la modélisation des séries temporelles (notamment pour les tests ADF, la décomposition saisonnière, et les modèles ARIMA/SARIMA). Scikit-learn a été utilisé pour le calcul des métriques d'erreur (RMSE, MAE, MAPE), et Pmdarima pour la recherche automatique des paramètres optimaux des modèles.
- **Phases de l'implémentation** : Le projet a été structuré en plusieurs étapes. D'abord, une phase de préparation des données a permis de charger le fichier ‘monthly-beer-production-in-austr.csv’, d'interpoler les valeurs manquantes, et de diviser la série en ensembles d'apprentissage (80 %) et de test (20 %). Ensuite, une analyse exploratoire a été menée pour identifier la tendance, la saisonnalité et la stationnarité. La modélisation a suivi, avec l'ajustement de différents modèles et leur optimisation. Enfin, les prévisions ont été générées et comparées aux données réelles.
- **Stratégie de test** : La validation des modèles a été effectuée en plusieurs étapes. Une validation croisée temporelle a été utilisée pour évaluer la robustesse des modèles sur différentes périodes. Les performances ont été mesurées à l'aide de métriques comme le RMSE, le MAE et le MAPE sur l'ensemble de test. Les résidus ont été analysés via des tests statistiques (Shapiro-Wilk pour la normalité, Ljung-Box pour le blanchiment) afin de s'assurer que les hypothèses des modèles étaient respectées.

## 3 Partie 1 : Premier contact et préparation de notre dataframe

### 3.1 Importation de bibliothèques

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
```

```

4 from statsmodels.tsa.stattools import adfuller
5 from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
6 from statsmodels.tsa.statespace.sarimax import SARIMAX
7 from statsmodels.stats.diagnostic import acorr_ljungbox
8 from scipy.stats import shapiro, probplot
9 import seaborn as sns
10 from sklearn.metrics import mean_squared_error, mean_absolute_error,
11     mean_absolute_percentage_error
12 import warnings
13 import itertools
14 from sklearn.model_selection import TimeSeriesSplit
15 warnings.filterwarnings("ignore")

```

Après avoir téléchargé le fichier CSV, nous l'importons avec la fonction `.read()` de la librairie pandas.

```

1 beer_data = pd.read_csv('monthly-beer-production-in-austr.csv', index_col
2 =0, parse_dates=True)
3 print("Aperçu des données :")
4 print(beer_data.head())

```

- `index_col=0` : Indique que la première colonne du fichier CSV (colonne 0) doit être utilisée comme index du DataFrame. Dans notre cas, cette colonne contient les dates, ce qui permet de structurer les données comme une série temporelle avec un index temporel.
- `parse_dates=True` : Active la conversion automatique des chaînes de caractères représentant des dates (dans l'index ou d'autres colonnes spécifiées) en objets ‘`datetime`’ de Python. Cela est crucial pour les séries temporelles, car cela permet d'utiliser des fonctions spécifiques aux dates (comme le filtrage par période ou la décomposition saisonnière).
- Le résultat de la fonction `.head()` nous montre les 5 premières lignes de notre dataframe ( c'est pour avoir de la visibilité sur la structure générale du dataframe).

Month	Monthly beer production
1956-01-01	93.2
1956-02-01	96.0
1956-03-01	95.2
1956-04-01	77.1
1956-05-01	70.9

Table 1: Structure du dataframe

```
1 print("\nStatistiques descriptives :")
2 print(beer_data.describe())
```

Stats	Monthly beer production
Count	476.000
mean	136.395
std	33.738725
min	64.800
25%	112.900
50%	139.150
75%	158.825
max	217.800

Table 2: Statistiques descriptives

La fonction `.describe()` nous fournit les statistiques descriptives de notre DataFrame. D'après ces statistiques, notre DataFrame ne contient pas de valeurs aberrantes (outliers). Si toutefois certaines étaient présentes, nous pourrions utiliser la méthode `.interpolate()` de la bibliothèque pandas pour les corriger.

```
1 # V e r i f i c a t i o n e t g e s t i o n d e s v a l e u r s m a n q u a n t e s
2 i f b e e r _ d a t a . i s n a ( ) . s u m ( ) . s u m ( ) > 0 :
3     p r i n t ( " \ n R e m p l a c e m e n t d e s v a l e u r s m a n q u a n t e s p a r i n t e r p o l a t i o n
4         l i n e a i r e . . . " )
5     b e e r _ d a t a = b e e r _ d a t a . i n t e r p o l a t e ( m e t h o d = ' l i n e a r ' , l i m i t _ d i r e c t i o n =
6         f o r w a r d ' )
```

**Aucun retour NaN** : notre jeu de données ne contient aucune valeur manquante. De plus, l'analyse statistique ne révèle pas de valeurs aberrantes.

```
1 # V i s u a l i s a t i o n i n i t i a l e
2 p l t . s t y l e . u s e ( ' f i v e t h i r t y e i g h t ' )
3 p l t . f i g u r e ( f i g s i z e = ( 1 4 , 5 ) )
4 p l t . p l o t ( b e e r _ d a t a , c o l o r = ' b l u e ' , l i n e w i d t h = 2 )
5 p l t . t i t l e ( ' P r o d u c t i o n m e n s u e l l e d e b i r e e n A u s t r a l i e ' )
6 p l t . x l a b e l ( ' D a t e ' )
7 p l t . y l a b e l ( ' P r o d u c t i o n ' )
8 p l t . s h o w ( )
```

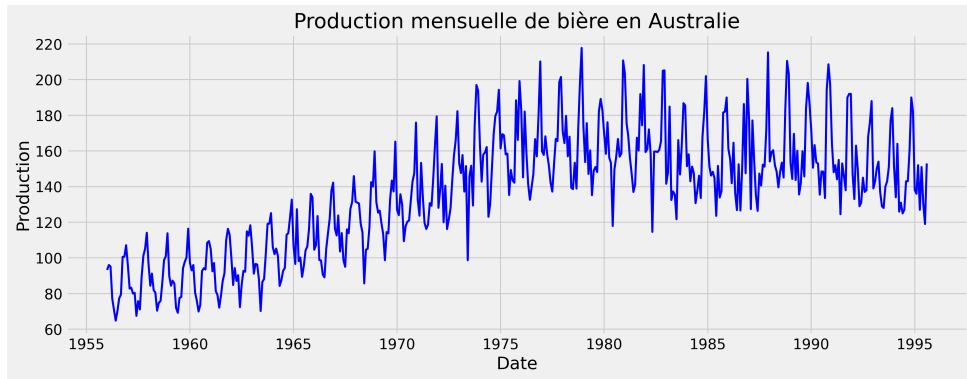


Figure 1: Production mensuelle de bière en Australie

La prochaine étape consiste à diviser les données en ensembles d’entraînement et de test pour éviter le surajustement *overfitting* et permettre des prédictions fiables.

```

1 #Division train/test (80% train, 20% test)
2 train_size = int(len(beer_data) * 0.8)
3 train, test = beer_data[:train_size], beer_data[train_size:]
4 print(f"\nTaille train : {train.shape}, Taille test : {test.shape}")
5 train.to_csv('train_beer_data.csv')
6 test.to_csv('test_beer_data.csv')
```

La commande `to_csv` donne deux fichiers format CSV, du train et du test avec des tailles : Taille train:(380, 1), Taille test:(96, 1)

## 4 Partie 2 : Composants de notre série chronologique

### 4.1 Composante d'une serie chronologique

Une série chronologique se compose principalement de `tendance`, de `saisonalité` et de `bruit`. Nous examinerons les composants de notre série temporelle, en nous concentrant sur les méthodes de décomposition automatisées.

### 4.2 Modélisation de série chronologique

**Modèle additif linéaire:**

$$y(t) = \text{Tendance} + \text{Saisonalite} + \text{Bruit}$$

Elle est utile lorsque les variations autour de la tendance ne varient pas avec le niveau de la série temporelle. Les composants sont additionnés.

**Modèle multiplicatif:**

$$y(t) = \text{Tendance} \times \text{Seasonnalite} \times \text{Bruit}$$

Une saisonnalité non linéaire a une fréquence et/ou une amplitude croissante ou décroissante dans le temps. Il est utile lorsque la tendance est proportionnelle au niveau de la série chronologique. Les composants sont multipliés ensemble.

*Dans notre cas la serie suit un modèle additif lineaire.*

```

1 from pylab import rcParams
2
3 rcParams['figure.figsize'] = 12, 8
4 plt.style.use('fivethirtyeight')
5
6 result = seasonal_decompose(train, model='additive', period=12)
7
8 trend = result.trend
9 seasonal = result.seasonal
10 resid = result.resid

```

La méthode `seasonal_decompose` de `statsmodels` permet d'effectuer une décomposition automatique d'une série temporelle en ses composantes.

```

1 fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(12, 6), sharex=True)
2
3 ax1.plot(train, label='Serie originale', color='blue', linewidth=1.5)
4 ax1.set_title('Serie originale : Production mensuelle de biere')
5 ax1.grid(True, alpha=0.2)
6 ax1.legend()
7
8 ax2.plot(trend, label='Tendance', color='red', linewidth=1.5)
9 ax2.set_title('Tendance')
10 ax2.grid(True, alpha=0.2)
11 ax2.legend()
12
13 ax3.plot(seasonal, label='Saisonnalité', color='green', linewidth=1.5)
14 ax3.set_title('Saisonnalité')
15 ax3.grid(True, alpha=0.2)
16 ax3.legend()
17
18 ax4.plot(resid, label='Residus', color='black', linewidth=1.5)
19 ax4.set_title('Residus')
20 ax4.grid(True, alpha=0.2)
21 ax4.legend()
22
23 plt.tight_layout()
24 plt.show()

```

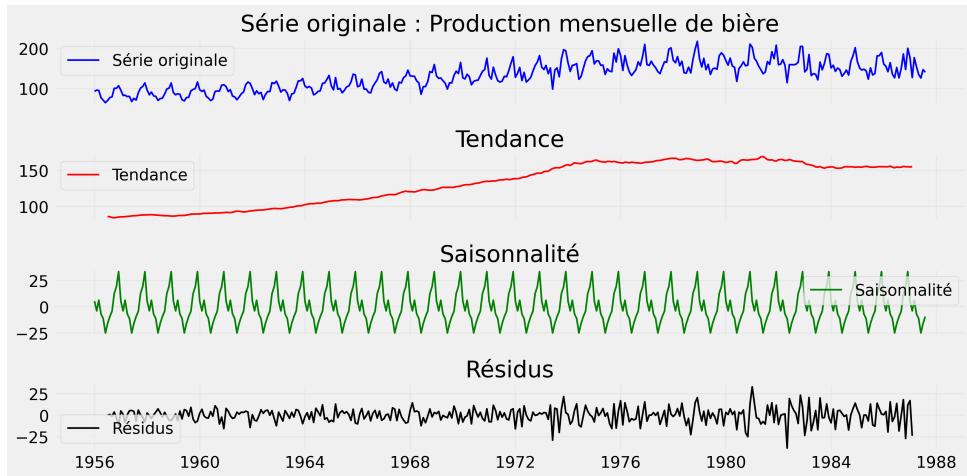


Figure 2: Décomposition de la série chronologique.

On a décomposé la série en trois composantes **tendance**, **saisonnalité** et bruit(résidus), on analysera maintenant chacune apart.

### 4.3 Étude de la tendance

Dans cette partie, on étudiera la tendance, en effectuant une regression linéaire sur celle-ci et donc en déduire la nature de la tendance ( ascendante, descendante exponentielle...)

#### 4.3.1 Algorithme d'ajustement de la tendance par regression linéaire

```

1 from sklearn.linear_model import LinearRegression
2 X = np.arange(len(train)).reshape(-1, 1)
3 y = train.values
4 model = LinearRegression()
5 model.fit(X, y)
6 trend = model.predict(X)
7 plt.figure(figsize=(14, 5))
8 plt.plot(train.index, trend, label='Tendance ajust e', color='red')
9 plt.plot(train, label='Donnees originales', color='blue', alpha=0.5,
10         linewidth=2)
11 plt.title('Tendance de la production de biere')
12 plt.legend()
13 plt.show()
14 print("Tendance : Ascendante" if model.coef_[0] > 0 else "Tendance :
    Descendante")

```

Dans ce programme, nous avons utilisé la bibliothèque **scikit-learn** pour implémenter un modèle de régression linéaire prédéfini. Ensuite, nous avons appliqué la méthode **.predict(X)** pour générer les valeurs prédictes et tracer la droite de régression correspondante, avant de visualiser les résultats obtenus.

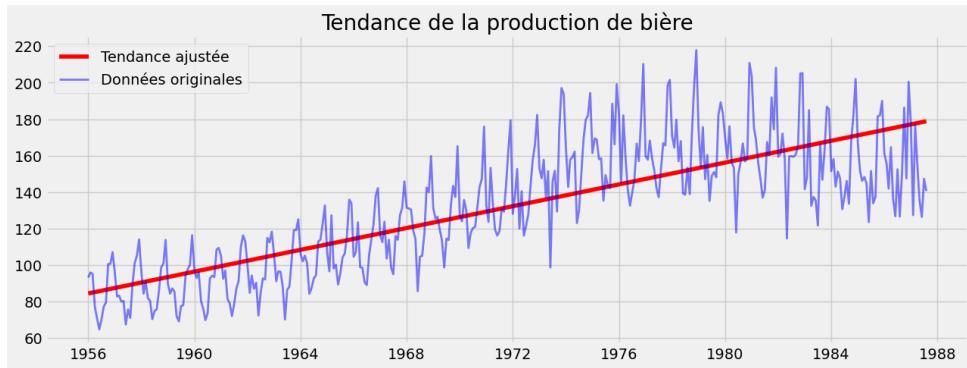


Figure 3: Ajustement de la tendance par regression linéaire.

D'après la figure ci-dessus, on remarque que la série a une tendance **ascendante**.

#### 4.4 Étude de la saisonnalité:

```

1 plt.figure(figsize=(12, 6))
2 plt.plot(seasonal, linewidth=2, color='green', label='Composante
    saisonni re')
3
4 plt.grid(True, linestyle='--', alpha=0.7)
5 plt.xlim(0, 2000)
6
7 plt.xlabel('Date', fontsize=12)
8 plt.ylabel('Amplitude de la Saisonnalit ', fontsize=12)
9
10 plt.legend()
11 plt.tight_layout()
12 plt.show()
```

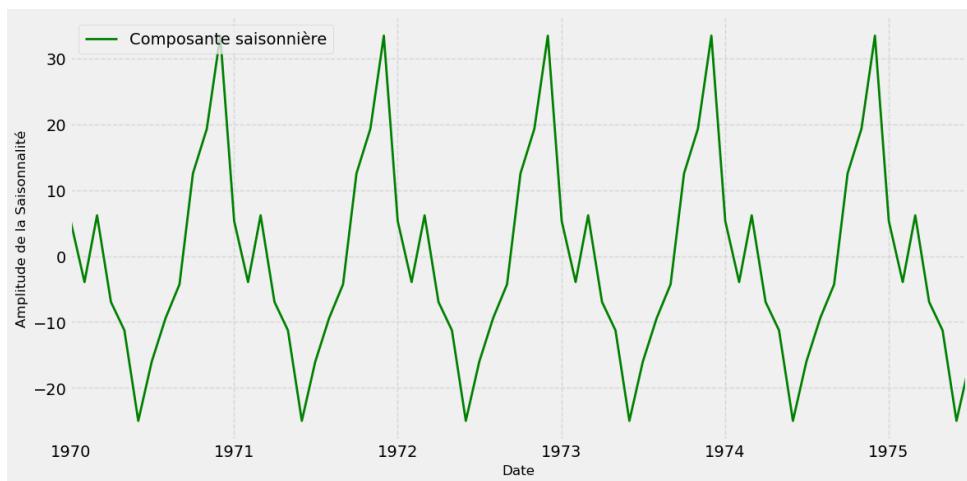


Figure 4: La composante saisonnière entre 1970 et 1975

La saisonnalité de cette série est marquée par un **cycle annuel** régulier avec des

amplitudes significatives, reflétant des variations saisonnières prévisibles. Cela renforce l'importance d'utiliser des modèles prenant en compte la saisonnalité (comme SARIMA) et suggère que des facteurs temporels spécifiques, probablement liés aux habitudes de consommation en Australie, sont à l'origine de ces motifs.

## 4.5 Études de la stationnarité :

La stationnarité est une caractéristique importante des séries chronologiques. Une série chronologique est stationnarité si elle a une moyenne et une variance constantes dans le temps.

La plupart des modèles ne fonctionnent qu'avec des données stationnaires, car cela facilite la modélisation. Toutes les séries temporelles ne sont pas stationnaires, mais nous pouvons les transformer en séries stationnaires de différentes manières.

Nous utiliserons le test de Dickey Fuller pour vérifier la stationnarité de nos données.

### 4.5.1 Formulation mathématique du test de Dickey Fuller

1.  $\Delta X_t = \rho X_{t-1} - \sum_{j=2}^p \phi_j \Delta X_{t-j+1} + c + \beta t + a_t$
2.  $\Delta X_t = \rho X_{t-1} - \sum_{j=2}^p \phi_j \Delta X_{t-j+1} + c + a_t$
3.  $\Delta X_t = \rho X_{t-1} - \sum_{j=2}^p \phi_j \Delta X_{t-j+1} + a_t$

où  $\rho = 1 - \phi = 0$ ,  $\phi = 1$ ,  $a_t$  est un BB et  $p$  est déterminée à l'aide du corrélogramme partiel de la série différenciée  $\Delta X_t$ .

Les hypothèses du test sont :

- $H_0 : \rho = 0$  (la série a une racine unitaire, donc non stationnaire).
- $H_1 : \rho < 1$  (la série est stationnaire).

Nous utiliserons le test de Dickey Fuller pour vérifier la stationnarité de nos données. Ce test générera des valeurs critiques et une p-value, ce qui nous permettra d'accepter ou de rejeter l'hypothèse nulle qu'il n'y a pas de stationnarité. Si nous rejetons l'hypothèse nulle, cela signifie que nous acceptons l'alternative, qui stipule qu'il y a stationnarité. Ces valeurs nous permettent de tester dans quelle mesure les valeurs actuelles changent avec les valeurs passées. S'il n'y a pas de stationnarité dans l'ensemble de données, un changement des valeurs actuelles n'entraînera pas de changement significatif des valeurs passées.

Testons la stationnarité de nos données.

#### 4.5.2 Test de Dickey-Fuller:Algorithme et implmentation

```

1 def test_stationarity(series):
2     result = adfuller(series, autolag='AIC')
3     print('Statistique ADF:', result[0])
4     print('p-value:', result[1])
5     if result[1] <= 0.05:
6         print("La serie est stationnaire (p-value <= 0.05).")
7         return True
8     else:
9         print("La serie n'est pas stationnaire (p-value > 0.05).")
10        return False
11
12 print("Test de stationnarit sur la s rie originale :")
13 is_stationary = test_stationarity(train.iloc[:, 0])
14
15 # Transformation pour stationnarite
16 if not is_stationary:
17     print("\nStationnarisation n cessaire...")
18     train_transformed = train.iloc[:, 0].diff().dropna()
19     print("\nTest apr s premi re diff renciation :")
20     is_stationary = test_stationarity(train_transformed)
21     d = 1 if is_stationary else 2
22     D = 1 # Differenciation saisonniere (periode 12)
23 else:
24     train_transformed = train.iloc[:, 0]
25     d, D = 0, 0

```

Listing 1: Test de Dickey-Fuller

Les résultats des tests de Dickey-Fuller sur la série originale et après première différenciation sont présentés dans le tableau suivant :

Metrics	Série originale	Première différenciation
Statistique ADF	-1.75	-5.79
p-value	0.40477	4.8063e-07
Conclusion	Non stationnaire (p-value > 0.05)	Stationnaire (p-value ≤ 0.05)

Table 3: Resultats des tests de stationnarité

La **p-value** est inférieure au seuil de 0,05(après différenciation) et la statistique ADF est proche des valeurs critiques. Par conséquent, **la série temporelle est stationnaire**.

## 5 Partie 4 : Étude de l'autocorrélation de notre série chronologique

### 5.1 Autocorrélation dans les séries chronologiques

- Autocorrélation

Mesure la relation d'une série temporelle avec elle-même à différents décalages (lags). Elle aide à détecter des motifs répétitifs dans le temps. L'ACF (Autocorrelation Function) est souvent utilisée pour la visualiser.

- Bruit Blanc

Une série est un bruit blanc si ses valeurs sont aléatoires, avec une moyenne nulle, variance constante et aucune corrélation entre les observations. Elle ne contient aucune structure temporelle exploitable.

Voici les bibliothèques que nous utilisons dans cette partie:

```
1 from statsmodels.graphics.tsaplots import plot_acf
2 from statsmodels.graphics.tsaplots import plot_pacf
```

#### 5.1.1 ACF-Autorrelation:

```
1 plt.figure(figsize=(14, 5))
2 plot_acf(train, lags=30)
3 plt.title('Fonction d\'autocorrelation (ACF)')
4 plt.show()
```

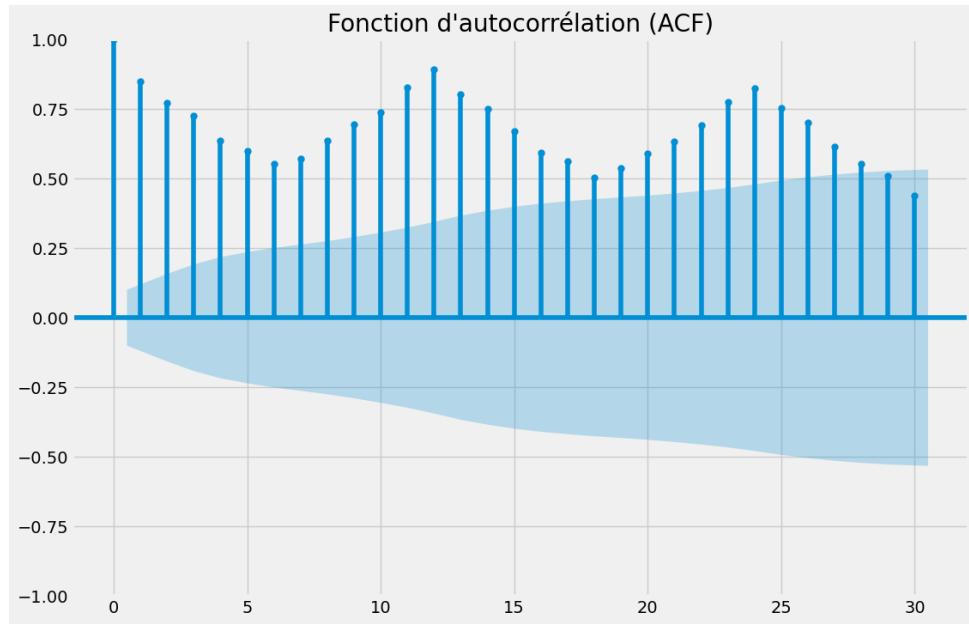


Figure 5: Fonction d'autocorrélation

#### Interpretation:

La figure montre la fonction d'autocorrélation (ACF) de la série "Monthly beer production" sur 30 décalages (lags). Les barres bleues représentent les coefficients d'autocorrélation, avec une zone de confiance (bande bleue ombrée) à 95%

- **Décalage 0:** L'autocorrélation est 1 (par définition), indiquant une corrélation parfaite avec elle-même.
- **Décalage 1 à 12:** Les valeurs d'autocorrélation sont significativement élevées (proches de 0.75 à 0.5), décroissant lentement, ce qui suggère une forte autocorrélation et une possible tendance ou saisonnalité.
- **Au-delà de 12:** L'autocorrélation reste notable jusqu'à environ le décalage 20-25, puis se stabilise autour de 0, restant dans la bande de confiance, indiquant une influence saisonnière persistante (période probable de 12 mois).
- *Signification statistique:*  
Les coefficients dépassant la bande de confiance aux premiers décalages indiquent une autocorrélation significative, confirmant une structure non stationnaire avec une saisonnalité (période 12). Cela justifie l'usage d'un modèle SARIMAX avec différenciation et composante saisonnière.

#### 5.1.2 PACF-Autocorrelation partielle:

```

1 plt.figure(figsize=(14, 5))
2 plot_pacf(train, lags=30)
3 plt.title('Fonction d'autocorr lation partielle (PACF)')
4 plt.show()

```

La fonction `plot_pacf()` renvoie également des intervalles de confiance, qui sont représentés par des régions ombrées en bleu. Si les valeurs d'autocorrélation partielle sont au-delà de ces régions d'intervalle de confiance, vous pouvez supposer que les valeurs d'autocorrélation partielle observées sont statistiquement significatives.

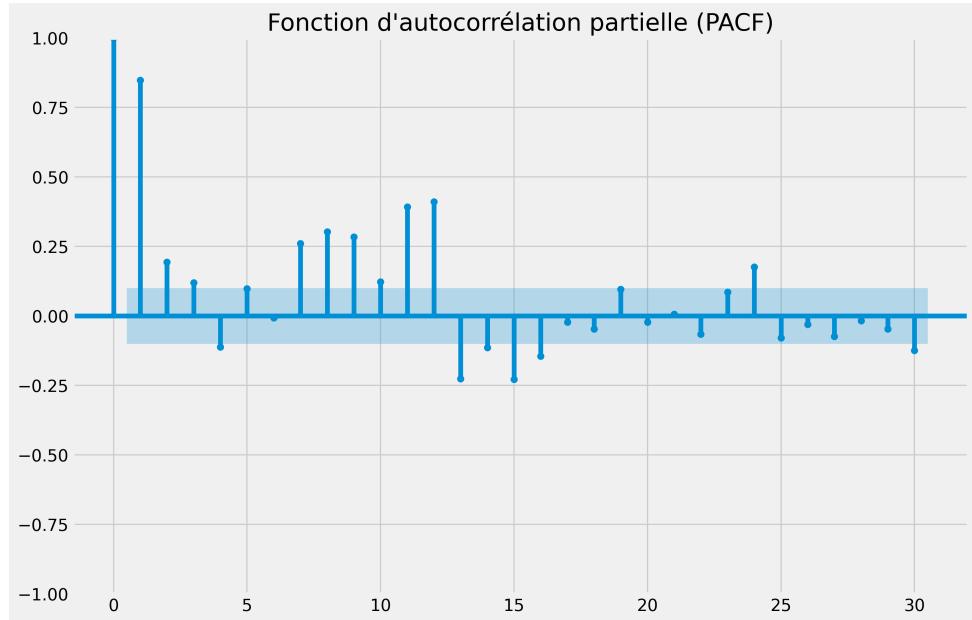


Figure 6: Fonction d'autocorrélation partielle

## 6 Partie 5 : Estimation des modèles

### 6.1 Modèle (AR)

Le modèle autorégressif (AR) correspond au moment où la valeur actuelle d'une série chronologique peut être prédite à partir des valeurs précédentes de la même série. C'est une régression utilisant sa même série, bien que décalée d'un pas de temps, appelé décalage. La valeur actuelle est une moyenne pondérée de ses valeurs passées. Le  $t - 1$  et le  $t - 2$  sont des décalages de la série chronologique  $y$ . Le terme d'erreur (bruit) est représenté par  $\epsilon_t$ . Les valeurs  $\alpha_1$  et  $\alpha_2$  sont les coefficients du modèle.

- AR(1) :

$$y_t = \mu + \alpha_1 y_{t-1} + \epsilon_t \quad (1)$$

- AR(2) :

$$y_t = \mu + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \epsilon_t \quad (2)$$

où :

- $y_t$  est la valeur de la série à l'instant  $t$ ,
- $\mu, \alpha_1, \alpha_2$  sont les coefficients autorégressifs,
- $\epsilon_t$  est le terme d'erreur, supposé  $\epsilon_t \sim \mathcal{B}\mathcal{B}(0, \sigma^2)$ .

### 6.1.1 Prévision à l'aide de AR

```

1 from statsmodels.tsa.ar_model import AutoReg
2 ar_model = AutoReg(train, lags=36).fit()
3 ar_pred = ar_model.predict(start=len(train), end=len(train)+len(test)-1,
4     dynamic=False)
5 plt.figure(figsize=(14, 5))
6 plt.plot(test, label='Données réelles', color='red')
7 plt.plot(ar_pred, label='Prévision AR', color='green')
8 plt.title('Prévision avec modèle AR')
9 plt.legend()
10 plt.show()
11 ar_rmse = np.sqrt(mean_squared_error(test, ar_pred))
12 print(f"RMSE AR : {ar_rmse:.2f}")
13 mape = mean_absolute_percentage_error(test.iloc[:, 0], ar_pred)
14 print(f'MAPE:{mape*100}')
15 print(ar_model.summary())

```

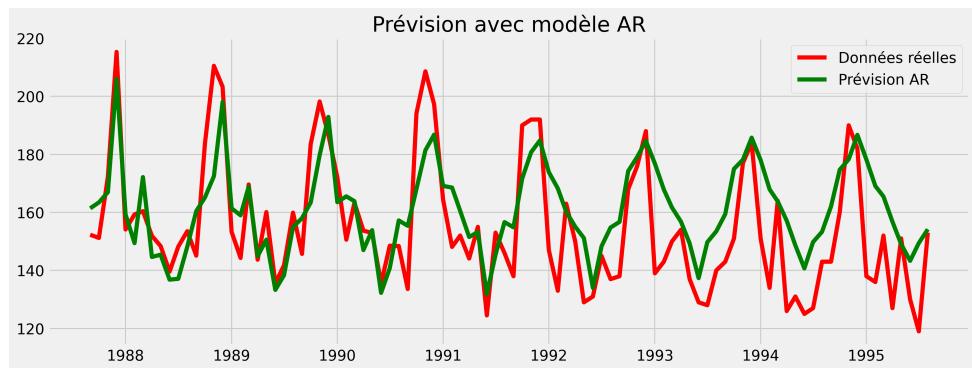


Figure 7: Prévision avec AR

## 6.2 Modèle ARIMA

```

1 # ARIMA
2 from statsmodels.tsa.arima.model import ARIMA
3 arima_model = ARIMA(train, order=(1,0,2)).fit()
4 arima_pred = arima_model.forecast(steps=len(test))
5 plt.figure(figsize=(14, 5))
6 plt.plot(test, label='Données réelles', color='red')
7 plt.plot(arima_pred, label='Prévision ARIMA', color='green')
8 plt.title('Prévision avec modèle ARIMA')
9 plt.legend()
10 plt.savefig('prev_arima.png', dpi=300, bbox_inches='tight')
11
12 plt.show()
13 arima_rmse = np.sqrt(mean_squared_error(test, arima_pred))
14 print(f"RMSE ARIMA : {arima_rmse:.2f}")

```

### 6.2.1 Comparaison entre AR et ARIMA

Nous avons ajusté un modèle AR et un modèle ARIMA sur la série chronologique. La figure suivante montre les prévisions obtenues avec le modèle ARIMA :

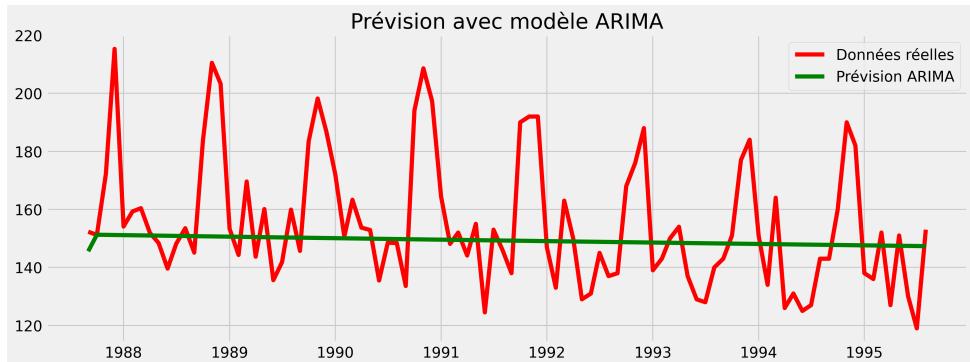


Figure 8: Prévision avec modèle ARIMA

Nous avons observé que le modèle AR donne des prévisions plus importantes que celles du modèle ARIMA. Cette différence peut s'expliquer comme suit :

- **Modèle AR** : Le modèle autorégressif, de la forme  $y_t = \sum_{i=1}^p \alpha_i y_{t-i} + \epsilon_t$ , se base uniquement sur les valeurs passées. Les coefficients  $\alpha_i$  amplifient les fluctuations importantes de la série (pics autour de 200), ce qui conduit à des prévisions plus élevées.
- **Modèle ARIMA** : Le modèle ARIMA( $p, d, q$ ), avec  $d = 1$ , modélise les différences  $\Delta y_t$ . La composante de moyenne mobile (MA) introduit un lissage, et les prévisions, après intégration, tendent à être plus plates (autour de 140–150 après 1992), sous-estimant les variations extrêmes.

- **Saisonnalité** : La série présente des fluctuations périodiques non capturées par un ARIMA simple. Un modèle SARIMA aurait pu mieux modéliser une saisonnalité potentielle.

### 6.3 Modèle SARIMAX

```

1 # SARIMAX (recherche simple des paramètres)
2 p = d = q = range(0, 2)
3 pdq = list(itertools.product(p, d, q))
4 seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p
5     , d, q))]
6 best_aic = float("inf")
7 best_model = None
8 for param in pdq:
9     for param_seasonal in seasonal_pdq:
10         try:
11             model = SARIMAX(train, order=param, seasonal_order=
12                 param_seasonal).fit(disp=False)
13             if model.aic < best_aic:
14                 best_aic = model.aic
15                 best_model = model
16         except:
17             continue
18 sarimax_pred = best_model.forecast(steps=len(test))
19 plt.figure(figsize=(14, 5))
20 plt.plot(test, label='Données réelles', color='red')
21 plt.plot(sarimax_pred, label='Prévision SARIMAX', color='green')
22 plt.title('Prévision avec modèle SARIMAX')
23 plt.legend()
24 plt.savefig('prev_sarimax.png', dpi=300, bbox_inches='tight')
25
26 plt.show()
27 sarimax_rmse = np.sqrt(mean_squared_error(test, sarimax_pred))
28 print(f"RMSE SARIMAX : {sarimax_rmse:.2f}")
29 print( mean_absolute_percentage_error(test.iloc[:, 0], sarimax_pred)
    *100)
30 print(best_model.summary())

```

RMSE:13.29 MAPE:6.65%

SARIMAX Results						
Dep. Variable:	Monthly beer production	No. Observations:	380			
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	Log Likelihood	-1369.083			
Date:	Mon, 26 May 2025	AIC	2748.165			
Time:	23:37:15	BIC	2767.692			
Sample:	01-01-1956 - 08-01-1987	HQIC	2755.924			
Covariance Type:	opg					
	coef	std err	z	P> z	[0.025	0.975]
ar.L1	-0.1687	0.042	-4.063	0.000	-0.250	-0.087
ma.L1	-0.8855	0.022	-40.838	0.000	-0.928	-0.843
ar.S.L12	0.1474	0.056	2.626	0.009	0.037	0.257
ma.S.L12	-0.8846	0.035	-25.545	0.000	-0.953	-0.817
sigma2	97.0802	5.250	18.491	0.000	86.790	107.370
Ljung-Box (L1) (Q):	0.48			Jarque-Bera (JB):	70.69	
Prob(Q):	0.49			Prob(JB):	0.00	
Heteroskedasticity (H):	6.34			Skew:	-0.48	
Prob(H) (two-sided):	0.00			Kurtosis:	4.92	

Figure 9: Résumé du modèle SARIMAX

### 6.3.1 Commentaires:

- Modèle:

$p = 1, d = 1, q = 1$ : Une composante  $AR(1)$ , une différenciation d'ordre 1, et une composante  $MA(1)$  pour la partie non stationnaire

$P = 1, D = 1, Q = 1, s = 12$ : Une composante saisonnière  $AR(1)$ , une différenciation saisonnière d'ordre 1, et une composante saisonnière  $MA(1)$  avec une période saisonnière de 12

- Coefficients:

Des  $p-values < 0.05$ , les coefficients alors sont tous significatifs.

- Statistiques de diagnostic:

Ljung-Box ( $Q$ ):  $p-value = 0.49 (> 0.05)$ . Cela indique qu'il n'y a pas d'autocorrélation significative dans les résidus à un décalage de 1, Les résidus suivent une loi de bruit blanc  $\epsilon_t \sim BB(0, \sigma^2)$

Les résidus ne suivent pas une loi normale, comme le confirme le test de Jarque-Bera ( $Prob(JB) = 0.00 < 0.05$ ). L'asymétrie négative ( $Skew = -0.48$ ) et un kurtosis élevé ( $4.92 > 3$ ) indiquent des queues plus lourdes, incompatibles avec une distribution normale.

```

1 residuals = best_model.resid
2 plt.figure(figsize=(12, 6))
3 plt.subplot(121)
4 sns.histplot(residuals, kde=True, stat="density")
5 plt.title('Histogramme des résidus avec courbe de densité')

```

```

6 plt.xlabel('R sidus')
7 plt.subplot(122)
8 probplot(residuals, dist="norm", plot=plt.gca())
9 plt.title('Q-Q Plot des residus')
10
11 plt.subplot(211)
12 plt.plot(residuals, linewidth=2)
13 plt.title('Residus du meilleur modèle SARIMAX')
14
15 plt.subplot(212)
16 plot_acf(residuals, lags=36, ax=plt.gca())
17 plt.title('ACF des residus')

```

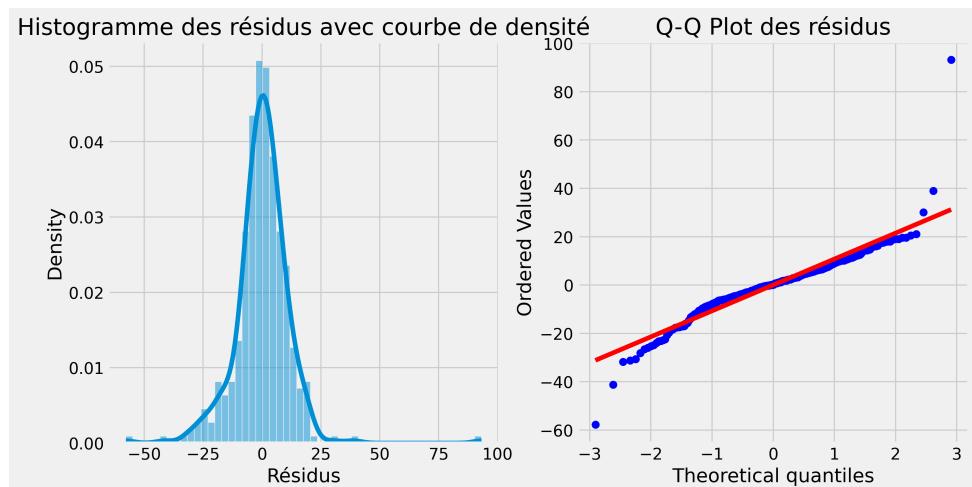


Figure 10: Histogramme et QQ plot des résidus

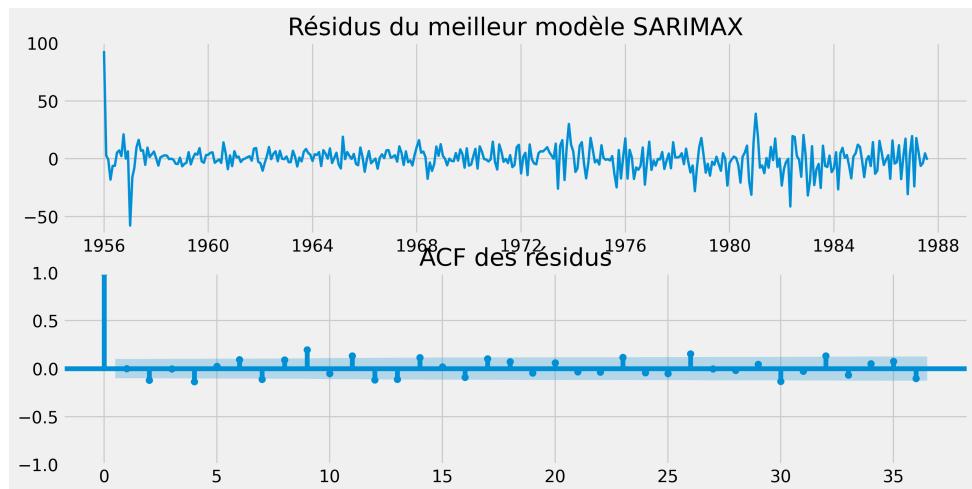


Figure 11: ACF et résidus du modèle

### 6.3.2 Prévision a l'aide de SARIMAX

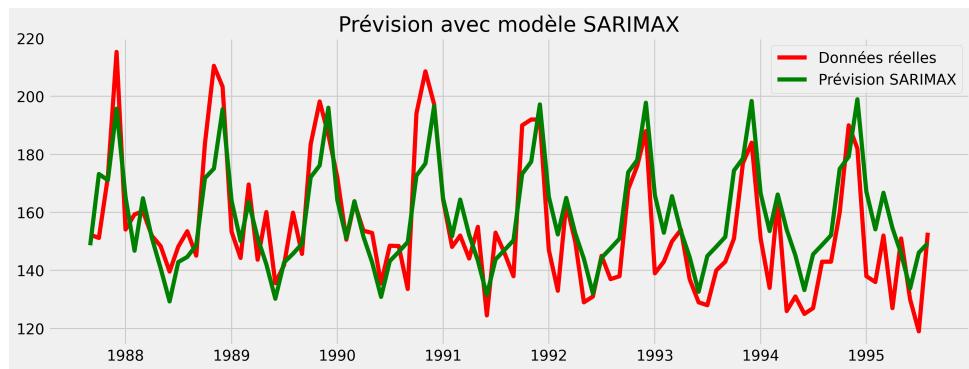


Figure 12: Prévision avec modèle SARIMAX

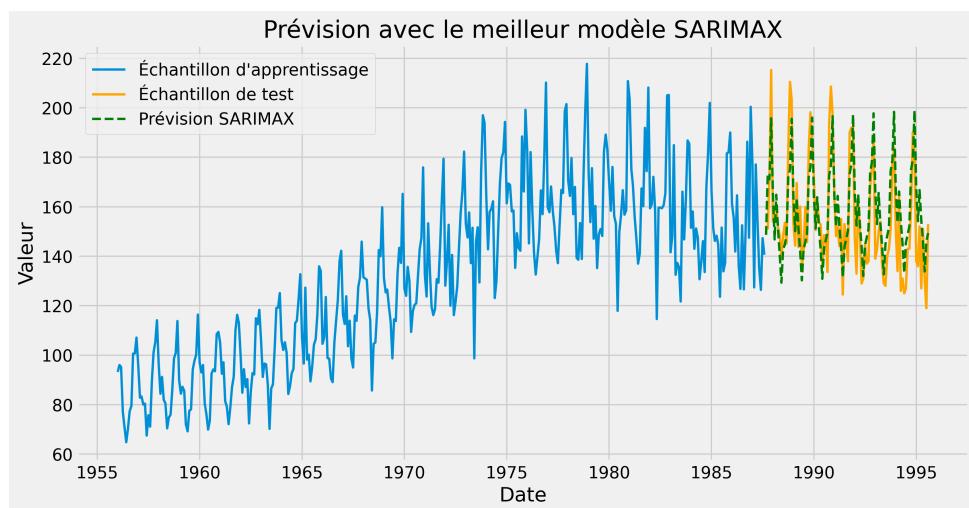


Figure 13: Prévision avec modèle SARIMAX

Commentaire sur les prévisions du SARIMAX

- **Figure 12 (1988-1995)** : Les prévisions SARIMAX (vert) suivent les données réelles (rouge) jusqu'en 1992, mais divergent ensuite, restant plus plates (140-150). Le modèle capture bien le court terme, mais sous-estime les fluctuations à long terme.
- **Figure 13 (1956-2005)** : Les prévisions (vert) et intervalles (bleu) s'alignent sur l'apprentissage (orange) jusqu'en 1987. Au-delà, les limites (non-normalité, hétéroscédasticité) pourraient sous-estimer les variations réelles.

## 6.4 Meilleur modèle SARIMAX avec Cross-Validation

Le modèle actuel ne permet pas d'expliquer ni de prévoir avec suffisamment de précision, c'est pourquoi je souhaite utiliser un algorithme basé sur la validation croisée pour identifier le meilleur modèle.

**Principe de Cross-Validation:** La validation croisée divise les données en plusieurs parties (par exemple, 5 "folds"). On entraîne le modèle sur une partie et on le teste sur une autre, en répétant pour chaque partie. La moyenne des résultats donne une estimation fiable de la performance, utile pour choisir le meilleur modèle.

```

1 # Recherche optimis e avec RMSE
2 best_rmse = float("inf")
3 best_params = None
4 best_model = None
5 tscv = TimeSeriesSplit(n_splits=5)
6
7 print("\nRecherche avanc e des param tres SARIMAX avec validation
     crois e (RMSE)...")
8 for param in pdq:
9     for param_seasonal in seasonal_pdq:
10        try:
11            rmse_scores = []
12            for train_idx, val_idx in tscv.split(train):
13                train_fold = train.iloc[train_idx]
14                val_fold = train.iloc[val_idx]
15
16                model = SARIMAX(train_fold, order=param, seasonal_order=
17                                 param_seasonal, enforce_stationarity=False, enforce_invertibility=
18                                 False)
19                result = model.fit(disp=False)
20
21                pred = result.forecast(steps=len(val_fold))
22                rmse = np.sqrt(mean_squared_error(val_fold.iloc[:, 0],
23                                                 pred))
24                rmse_scores.append(rmse)
25
26                # Moyenne des RMSE sur les folds
27                avg_rmse = np.mean(rmse_scores)
28                if avg_rmse < best_rmse:
29                    best_rmse = avg_rmse
30                    best_params = (param, param_seasonal)
31
32                    best_model = SARIMAX(train, order=param, seasonal_order=
33                                     param_seasonal, enforce_stationarity=False, enforce_invertibility=
34                                     False).fit(disp=False)
35        except:
36            continue
37
38 print(f"Meilleur mod le SARIMAX : order={best_params[0]},"
      seasonal_order={best_params[1]})
```

```
36 print(f"RMSE moyen sur validation crois e : {best_rmse:.2f}")
37 print(best_model.summary())
```

```
Recherche avancée des paramètres SARIMAX avec validation croisée (RMSE)...
Meilleur modèle SARIMAX : order=(1, 1, 1), seasonal_order=(1, 1, 0, 12)
RMSE moyen sur validation croisée : 12.37
SARIMAX Results
=====
Dep. Variable: Monthly beer production No. Observations: 380
Model: SARIMAX(1, 1, 1)x(1, 1, [], 12) Log Likelihood -1363.671
Date: Tue, 27 May 2025 AIC 2735.343
Time: 15:55:34 BIC 2750.820
Sample: 01-01-1956 HQIC 2741.501
- 08-01-1987
Covariance Type: opg
=====
            coef    std err      z   P>|z|      [0.025    0.975]
ar.L1     -0.1640   0.044    -3.751   0.000    -0.250    -0.078
ma.L1     -0.9409   0.017   -56.928   0.000    -0.973    -0.909
ar.S.L12  -0.3776   0.037   -10.235   0.000    -0.450    -0.305
sigma2    129.2547  7.345    17.599   0.000   114.860   143.650
=====
Ljung-Box (L1) (Q): 0.43 Jarque-Bera (JB): 43.11
Prob(Q): 0.51 Prob(JB): 0.00
Heteroskedasticity (H): 7.27 Skew: -0.37
Prob(H) (two-sided): 0.00 Kurtosis: 4.54
=====
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

Figure 14: Résumé du modèle

#### 6.4.1 Commentaires

- Modèle:
  - (1, 1, 1) : Une composante autoregressive (*AR*) d'ordre 1, une différenciation d'ordre 1 pour rendre la série stationnaire, et une moyenne mobile (*MA*) d'ordre 1.
  - (1, 1, 0, 12) : Une composante saisonnière avec une période de 12 mois (adaptée aux données mensuelles), incluant une partie autoregressive saisonnière (*AR.S*) d'ordre 1 et une différenciation saisonnière d'ordre 1, mais pas de moyenne mobile saisonnière (*MA.S = 0*)
- RMSE moyen : 12.37. Cela signifie que, en moyenne, les prévisions du modèle s'écartent de la vraie production de bière de 12.37 unités.
- Coefficients:
 

Des  $p-values < 0.05$ , les coefficients alors sont tous significatifs.
- Statistiques de diagnostic:
 

Ljung-Box (*Q*) :  $p-value = 0.51 (> 0.05)$ . Cela indique qu'il n'y a pas d'autocorrélation significative dans les résidus à un décalage de 1, Les résidus suivent une loi de bruit blanc  $\epsilon_t \sim \mathcal{BB}(0, \sigma^2)$

Les résidus ne suivent pas une loi normale, comme le confirme le test de Jarque-Bera  $Prob(JB) = 0.00 < 0.05$ . L'asymétrie négative ( $Skew = -0.37$ ) et un kurtosis élevé ( $4.54 > 3$ ) indiquent des queues plus lourdes, incompatibles avec une distribution normale.

#### 6.4.2 Prévision a l'aide de SARIMAX(Cross-Validation)

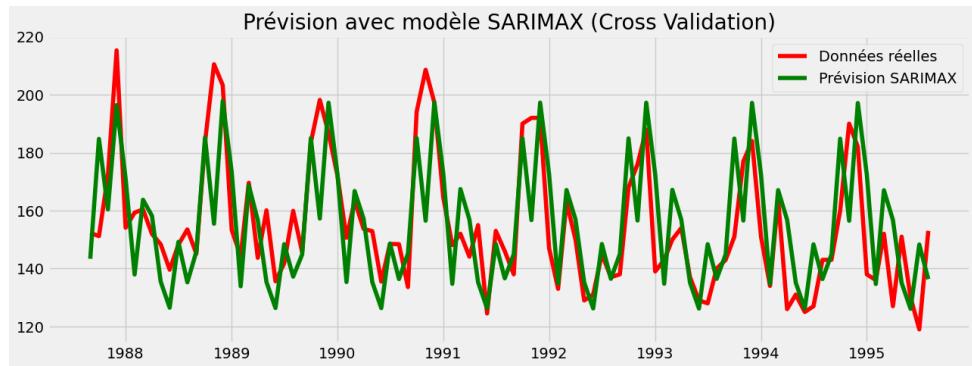


Figure 15: Prévision avec modèle SARIMAX(Cross-Validation)

```

1 ress=best_model1.resid
2 plt.figure(figsize=(12, 6))
3 plt.subplot(121)
4 sns.histplot(ress, kde=True, stat="density")
5 plt.title('Histogramme des residus avec courbe de densit ')
6 plt.xlabel('R sidus')
7 plt.subplot(122)
8 probplot(ress, dist="norm", plot=plt.gca())
9 plt.title('Q-Q Plot des residus')
10 plt.tight_layout()

```

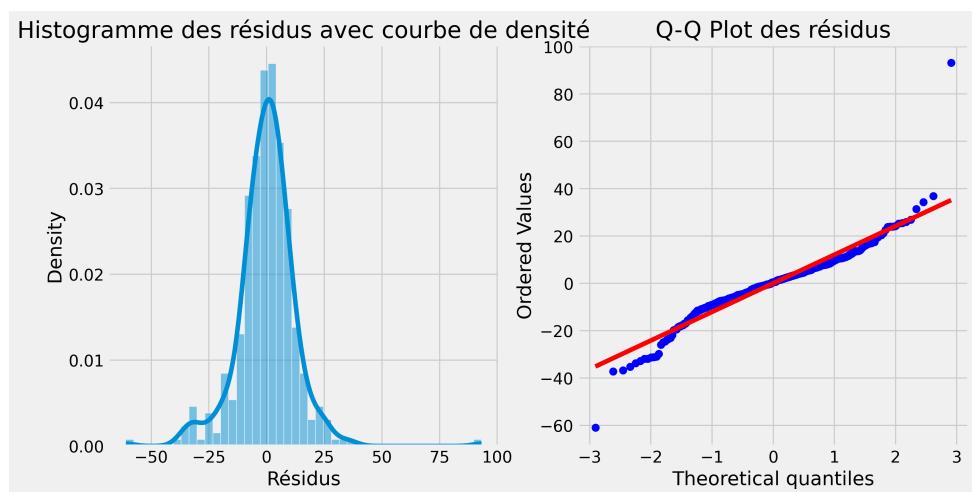


Figure 16: Histogramme et QQ plot des residus du modele ameliorer

## 7 Résumé du projet

Ce projet a analysé la série chronologique de la production mensuelle de bière (1956-1987)

- **Stationnarité** : La série, non stationnaire (ADF, p-value = 0.4048), l'est devenue après différenciation (p-value = 4.81e-07).
- **Modèles AR/ARIMA** : AR surestime les prévisions par rapport à ARIMA, en raison de l'absence de différenciation et de composante MA.
- **SARIMAX** : Le modèle SARIMAX(1, 1, 1)(1, 1, 1, 12) capture la saisonnalité (période 12). Les résultats sont présentés dans le tableau 4.
- **Résidus** : Pas d'autocorrélation (Ljung-Box, p-value = 0.49), mais non-normalité (Jarque-Bera, p-value = 0.00).
- **Performance** : RMSE = 13.29 et MAPE = 6.65% indiquent une précision acceptable, malgré une hétéroscédasticité (H = 6.34, p-value = 0.00).  
Les erreurs RMSE indiquent que SARIMAX offre les meilleures prévisions grâce à sa capacité à modéliser la saisonnalité. Une analyse plus approfondie des corrélogrammes et l'ajout de variables exogènes (par exemple, prix ou conditions économiques) pourraient améliorer les résultats.

Table 4: Métadonnées et performance du modèle SARIMAX

SARIMAX Results	
<b>Dep. Variable:</b>	Monthly beer production
<b>No. Observations:</b>	380
<b>Model:</b>	SARIMAX(1, 1, 1)×(1, 1, 0, 12)
<b>Date:</b>	Mon, 26 May 2025
<b>Time:</b>	23:37:15
<b>Sample:</b>	01-01-1956 – 08-01-1987
<b>AIC</b>	2735.343
<b>BIC</b>	2750.820
<b>RMSE</b>	12.37
<b>MAPE</b>	6.65%

## A Installation Instructions

1. Download the source code from [<https://github.com/adam04-D/Analyse-et-Pr-vision-d-une-s-rie-chronologiques-avec-Python.git>]
2. Run the application: TS.ipynb