Forecasting Models Project

ZAKARIA ETTOUHAMI - KENZA MOUHARRAR

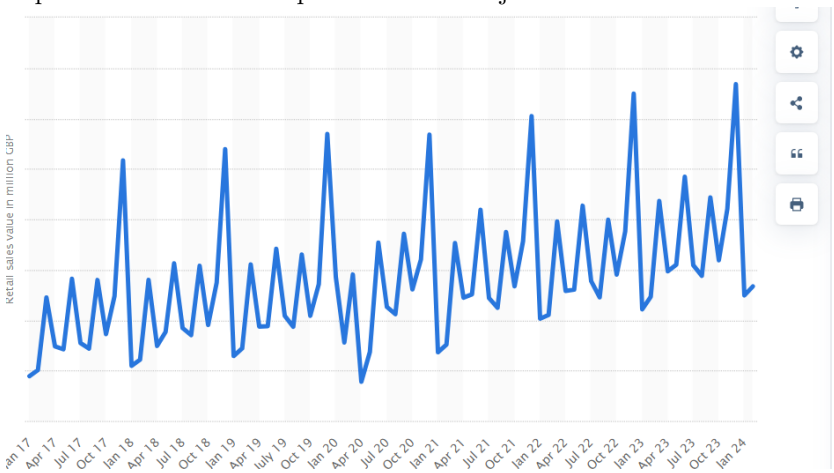
October 27, 2025

1 Introduction

Dans ce rapport du Mini Porjet, nous présenterons l'analyse d'une série temporelle que nous avons sélectionnée.Celle-ci porte sur les valeurs mensuelles des ventes de l'ensemble des commerces de détail, à l'exclusion des carburants automobiles, en Angleterre, sur la période allant de janvier 2017 à octobre 2023. Notre statistique représente la valeur totale mensuelle de ces ventes. Afin de faciliter votre compréhension de l'étude à suivre, voici un lien pour obtenir nôtre ensemble de données :

https://www.statista.com/statistics/287878/retail-sales-value-monthly-in-great-britainexcluding-fuel/

Notre ensemble de données comprends a total of 82 relevés. Chaque relevé correspond à un mois d'une période allant de janvier 2017 à octobre 2023.



La Figure met en évidence une **saisonnalité marquée** : on observe des pics réguliers de ventes, notamment en fin d'année (mois de décembre), traduisant l'impact des périodes de fêtes sur la consommation. Par ailleurs, la série présente une **tendance globalement croissante** sur la période 2017–2023, indiquant une hausse progressive du volume des ventes au fil du temps.

Partie I: Simulation

Dans cette première partie, nous allons coder des fonctions personnalisées `arima_sim` pour simuler un processus ARIMA(p, d, q) avec $d = 1$. Nous implémenterons également une fonction de prévision personnalisée, pas à pas, pour le cas d'un modèle ARIMA(p, d, q) avec $d = 1$.

Nous comparerons ensuite les résultats de nos fonctions personnalisées avec ceux obtenus à l'aide des fonctions déjà disponibles dans `R`. Enfin, nous expliquerons la logique qui sous-tend nos implémentations en nous appuyant sur les notions vues en cours.

1.1 Simulation d'un processus ARIMA stationnaire ($p \geq 2$, $d = 1$, $q \geq 1$) avec une fonction personnalisée `ARIMA_SIM`

La fonction `arima_sim` génère une trajectoire simulée d'un processus ARIMA(p, d, q), combinant les composantes autorégressive (AR), moyenne mobile(MA) et d'inté- gration d'ordre d.
Elle commence par générer un vecteur de bruit blanc $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$ de longueur `n + transitoire + q`, où `transitoire` correspond à une période initiale que l'on ne conservera pas. Cette phase transitoire permet au processus de converger vers un comportement stationnaire : au début, les valeurs sont fortement influencées par les conditions initiales et la série n'a pas encore atteint sa dynamique naturelle.
Un vecteur `y` pré-alloué de longueur `n + transitoire + max(p,q)` contient les valeurs initiales nécessaires pour calculer les lags de la partie AR et MA dès la première itération. La simulation ARMA(p, q) est ensuite effectuée pour chaque instant t selon la formule :

$$Y_t = \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$$

où la partie AR repose sur les p dernières valeurs de la série `y` , la partie MA sur les q dernières erreurs, et ε_t injecte de l'aléa.
Enfin, la période transitoire est supprimée afin de conserver uniquement les `n` dernières valeurs, représentant la série stationnaire du processus ARMA.

Pour le cas particulier où $d = 1$, la fonction simule d'abord la série différenciée $Y_t = \Delta X_t$, qui est stationnaire par construction. On applique donc la simulation ARMA sur cette série stationnaire. Ensuite, pour revenir à la série originale Y_t (non stationnaire), on effectue une intégration cumulative (`cumsum()`) de la série différenciée simulée. Cette étape correspond à l'inverse de la différenciation et permet d'obtenir une série ARIMA complète, non stationnaire. La meme logique s'applique pour un d ¿ 1 , on inverse la

différenciation d fois.

En résumé, la fonction `arima_sim` suit cette logique complète :

1. Générer le bruit blanc avec variance `sigma2`, en incluant une période transitoire pour stabiliser le processus.
2. Pré-allouer le vecteur `y` pour stocker toutes les valeurs (y compris la phase transitoire et les valeurs initiales pour les lags).
3. Simuler la composante ARMA(p, q) pour chaque instant.
4. Supprimer la phase transitoire pour ne conserver que les valeurs finales stationnaires (ou différenciées).
5. Si $d > 0$, appliquer une intégration cumulative pour revenir à la série ARIMA d'origine.

Cette approche garantit que la série simulée respecte à la fois la dynamique ARMA stationnaire et la structure ARIMA globale, tout en évitant les effets des conditions initiales.

Listing 1: Fonction arima$_sim()pour simuler un processus ARIMA(p, d, q)$

```r
#Entr es :
#p, d, q : ordres du mod le
#phi[1:p] : coefficients AR
#theta[1:q] : coefficients MA
#sigma2 : variance du bruit blanc
#n : longueur souhait e de la s rie

Simuler_ARIMA <- function(p, d, q, phi, theta, sigma2 = 1, n
    = 100, transitoire = 100) {
  #  tape  1 : g n rer le bruit blanc
  e <- rnorm(n + transitoire + q, mean = 0, sd = sqrt(sigma2
    ))

  #  tape  2 : initialiser la s rie
  y <- numeric(n + transitoire + max(p, q))

  #  tape  3 : simulation ARMA(p,q)
  for (t in (max(p,q) + 1):(n + transitoire)) {
    ar_part <- if(p>0) sum(phi * rev(y[(t-p):(t-1)])) else 0
    ma_part <- if(q>0) sum(theta * rev(e[(t-q):(t-1)])) else
        0
    y[t] <- ar_part + ma_part + e[t]
  }

  #  tape  4 : retirer la periode transitoire
  y <- y[(transitoire + 1):(transitoire + n)]

```

```r
  # tape 5 : int gration si d > 0
  if(d > 0) for(i in 1:d) y <- cumsum(y)

  return(y)
}
```

1.2 Implémenter une fonction de prévision personnalisée `Forecast_Per` basée sur les formules mathématiques du cours. Tester pour une prévision d'ordre $h = 2$

Nous avons essayé deux approches :

1. Approche développée dans le cours sur ARIMA/SARIMA, qui généralise le processus présenté pour ARIMA(3,1,1) (slide 37/132) pour le cas où $q > 1$, $d = 1$, $p > 1$.
2. Approche basée sur l'algorithme ARMA présenté dans le cours.

Il faut noter que la première approche n'est en réalité qu'un cas particulier de la deuxième, où, en code, la prévision se fait directement sans devoir effectuer implicitement une intégration pour revenir à la série non différenciée.

Première approche : prévision pas à pas pour ARIMA(p,1,q)

*1- Formules Mathématiques

On part de l'équation générale du modèle SARIMA :

$$\Phi(B)(1-B)X_t = \theta(B)\varepsilon_t$$

avec

$$\Theta(B) = 1 - \sum_{i=1}^{p} \phi_i B^i$$

On développe le polynôme $\Phi(B)(1-B)$:

$$\begin{aligned}
\Phi(B)(1-B) &= \Big(1 - \sum_{i=1}^{p} \phi_i B^i\Big)(1-B) \\
&= 1 - B - (1-B)\sum_{i=1}^{p} \phi_i B^i \\
&= 1 - B - \sum_{i=1}^{p} (1-B)\phi_i B^i \\
&= 1 - B + \sum_{i=1}^{p+1} \phi_i B^{i+1} - \sum_{i=1}^{p} \phi_i B^i \\
&= 1 - B + \sum_{i=2}^{p} (\phi_{i-1} - \phi_i) B^i - \phi_p B^{p+1}
\end{aligned}$$

On obtient ainsi :

$$\Phi(B)(1-B) = 1 - (1+\phi_1)B + \sum_{i=2}^{p}(\phi_{i-1} - \phi_i)B^i - \phi_p B^{p+1}$$

Ce qui conduit à la forme développée de la série :

$$\Phi(B)(1-B)X_t = X_t - (1+\phi_1)X_{t-1} + \sum_{i=2}^{p}(\phi_{i-1} - \phi_i)X_{t-i} - \phi_p X_{t-p-1}$$

$$\Rightarrow X_t = (1+\phi_1)X_{t-1} - \sum_{i=2}^{p}(\phi_{i-1} - \phi_i)X_{t-i} - \phi_p X_{t-p-1} + \Theta(B)\varepsilon_t$$

$$\Rightarrow X_t = (1+\Phi_1)X_{t-1} - \sum_{i=2}^{p}(\Phi_{i-1} - \Phi_i)X_{t-i} - \Phi_p X_{t-p-1} + \varepsilon_t + \Theta_1\varepsilon_{t-1} + \cdots + \Theta_p\varepsilon_{t-p}$$

Projection temporelle : on remplace $t \longrightarrow T + h$:

$$\hat{X}_{T+h|T} = (1+\Phi_1)\hat{X}_{T+h-1|T} - \sum_{i=2}^{p}(\Phi_{i-1} - \Phi_i)\hat{X}_{T+h-i|T} - \Phi_p\hat{X}_{T+h-p-1|T} + \varepsilon_{T+h|T} + \Theta_1\hat{\varepsilon}_{T+h-1|T} + \cdots + \Theta_p\hat{\varepsilon}_{T+h-p|T}$$

Substitution conditionnelle : Si $(T+h-i) \leq T \Rightarrow h - i \leq 0$, alors

$$\hat{X}_{T+h-i|T} = X_{T+h-i}, \quad \text{sinon } \hat{X}_{T+h-i|T} = \hat{X}_{T+h-i}$$

La liste des coefficients associée aux termes $\hat{X}_{T+h-i|T}$, utilisée dans le code sous le nom `phi_expanded`, est :

$$[1+\phi_1,\ -(\phi_2 - \phi_1),\ -(\phi_3 - \phi_2),\ \ldots,\ -(\phi_{p-1} - \phi_p),\ -\phi_p]$$

*2- Code

Le code suivant implémente la logique expliquée et démontrée ci-dessus. On note que l'étape de substitution temporelle, c'est-à-dire que pour $h - i > 0$ on remplace les valeurs par les prévisions calculées aux pas précédents, est effectuée automatiquement. En effet, en incrémentant la liste y avec les valeurs prédites, il suffit simplement de vérifier que $\text{length}(y) - i$ est bien dans l'intervalle des indices de y.

Listing 2: Fonction Forecast_Per pour prévision pas à pas ARIMA(p,1,q)

```
Forecast_Per <- function(model, h = 5) {
  # Pr vision pas    pas pour un mod le SARIMA(p,1,q)

  phi   <- model$model$phi
  theta <- model$model$theta
  d     <- model$arma[6]
  p     <- model$arma[1]
  q     <- model$arma[2]
```

```r

  y <- as.numeric(model$x)
  e <- as.numeric(residuals(model))

  forecasts <- numeric(h)

  phi_expanded <- numeric(p + 1)
  phi0 <- 1
  for (i in 0:p) {
    if (i == 0) phi_expanded[i + 1] <- phi0 + ifelse(p >= 1,
        phi[1], 0)
    else if (i < p) phi_expanded[i + 1] <- phi[i + 1] - phi[
        i]
    else phi_expanded[i + 1] <- -phi[p]
  }

  for (step in 1:h) {
    y_pred <- 0
    for (i in 1:length(phi_expanded)) {
      if (length(y) - i >= 0) {
        y_pred <- y_pred + phi_expanded[i] * y[length(y) - i
            + 1]
      }
    }
    if (q > 0) {
      for (j in 1:q) {
        if (length(e) - j >= 0) {
          y_pred <- y_pred + theta[j] * e[length(e) - j + 1]
        }
      }
    }
    forecasts[step] <- y_pred
    y <- c(y, y_pred)
    e <- c(e, 0)
  }

  return(forecasts)
}
```

*Deuxième approche : générale

1-Algorithme

Cette approche se base sur l'algorithme suivant :

Version simplifiée pour $d \geq 1$ Étape 1 : Modèle des différences

Soit

$$Y_t = \nabla^d X_t$$

un processus ARMA stationnaire. Les prévisions $\hat{Y}_{T,T+h}$ sont calculées via les méthodes ARMA standard.

Étape 2 : Intégration vers les niveaux
Les prévisions pour la série originale X_t sont obtenues par cumul :

$$\hat{X}_{T,T+h} = \hat{Y}_{T,T+h} + \hat{X}_{T,T+h-1},$$

avec la condition initiale

$$\hat{X}_{T,T} = X_T.$$

Calcul des prévisions ARMA Les prévisions ARMA se calculent de la manière suivante :

1. À l'instant t, calculer $\hat{X}_t^{(k)}$ pour tout k à partir de la forme AR(∞) ou ARMA.
2. À l'instant $t+1$, après observation de X_{t+1}, mettre à jour les prévisions selon :
$$\hat{X}_{t+1}^{(k-1)} - \hat{X}_t^{(k)} = h_{k-1}\big(X_{t+1} - \hat{X}_t^{(1)}\big)$$
3. Répéter pour $k = 2, \ldots, K$.
4. Les prévisions multi-pas suivantes se calculent de manière récursive.

Code Le mot-clé `rev()` est utilisé pour inverser l'ordre d'un vecteur.
Dans la boucle pas-à-pas, on souhaite multiplier les dernières valeurs de la série par les coefficients AR dans le bon ordre : ϕ_1 multiplie la valeur immédiatement précédente, ϕ_2 la deuxième avant, etc. Comme `tail(y, p)` renvoie les p dernières valeurs du plus ancien au plus récent, `rev()` permet de les inverser pour que la valeur la plus récente corresponde au premier coefficient AR, respectant ainsi la formule mathématique de l'AR.

Listing 3: Prévision pas-à-pas ARIMA sans phi_expanded

```
Forecast_Per_2 <- function(model, h = 5) {

  # Extraction des composantes
  phi   <- model$model$phi         # Coefficients AR
  theta <- model$model$theta       # Coefficients MA
  d     <- model$arma[6]           # Ordre de
      diff renciation (ici, d = 1)
  p     <- model$arma[1]
  q     <- model$arma[2]

  # S rie ajust e et r sidus
  y <- as.numeric(model$x)
  e <- as.numeric(residuals(model))

  forecasts <- numeric(h)

  # Boucle pas-  -pas
  for (step in 1:h) {
```

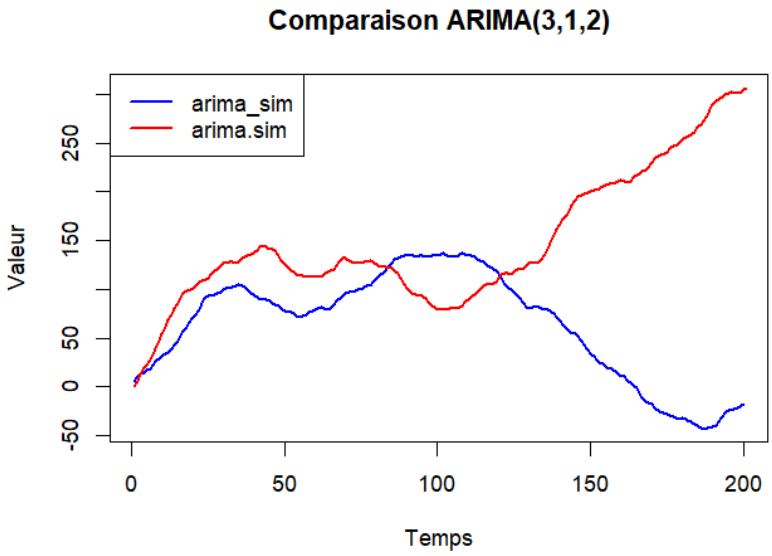
```
    ar_part <- if (p > 0) sum(phi * rev(tail(y, p))) else 0
    ma_part <- if (q > 0) sum(theta * rev(tail(e, q))) else
        0

    y_pred <- ar_part + ma_part + 0  # erreurs futures
        suppos es nulles
    forecasts[step] <- y_pred

    # Mise    jour de y et e pour la prochaine   tape
    y <- c(y, y_pred)
    e <- c(e, 0)
  }

  # Reconstruction si d > 0
  if (d > 0) {
    forecasts <- cumsum(forecasts)
  }

  return(forecasts)
}
```

application pour h=2

Pour une série ARIMA(3,1,2) et un horizon $h = 2$, on observe que la première fonction personnalisée (`manual_forecast`) reproduit exactement les mêmes valeurs que la fonction native de R (`auto_forecast`), à savoir -187.38 et -188.55. Cela confirme la validité de notre implémentation. En revanche, la seconde version (`manual_forecast_2`) présente des écarts importants (-168.18 et -320.29), suggérant une erreur dans la gestion de la composante ARMA ou dans la reconstruction après différenciation. Ces différences indiquent que la logique de mise à jour des valeurs prédictes doit être ajustée pour obtenir une cohérence complète avec les prévisions standards de R.

1.3 Comparer vos résultats avec ceux fournis par la fonction `package forecast` de R (analyse graphique et évaluation par MSE ou autre métrique appropriée)

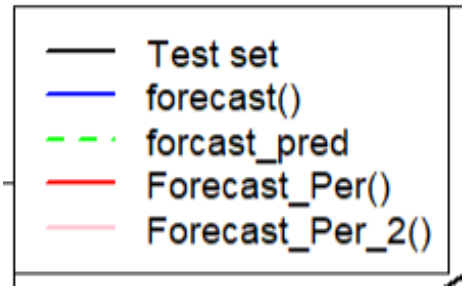
1.3.1 Comparaison entre `Arima_sim` et `arima.sim` de R



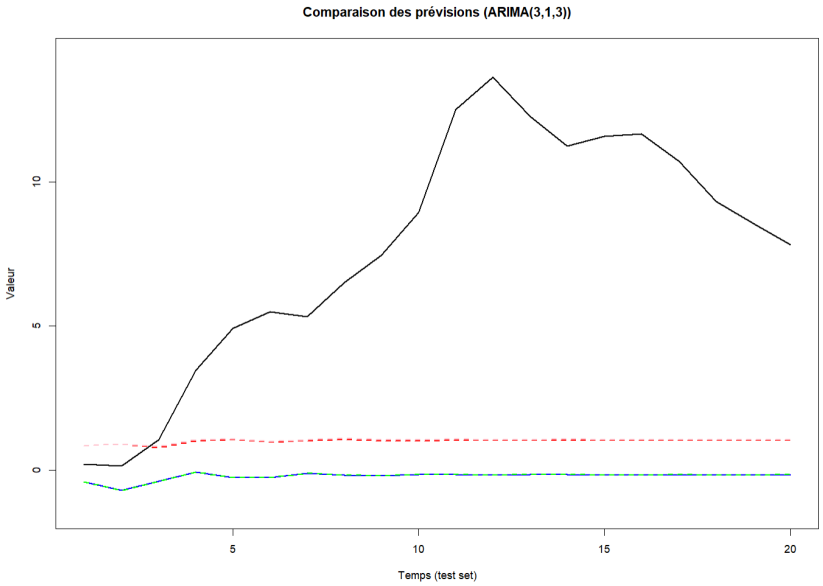
1.3.2 Comparaison de `Forecast_Per`, `Forecast_Per_2` et `forecast()` de R

Pour comparer les prévisions obtenues à l'aide de nos fonctions personnalisées, développées précédemment, avec celles des fonctions `forecast()` et `predict()` déjà implémentées dans R, nous avons simulé deux séries temporelles suivant des processus ARIMA(3,1,3) et ARIMA(7,1,8). Chaque série a été divisée en deux parties : un ensemble d'apprentissage (*X-Train*) utilisé pour l'estimation du modèle, et un ensemble de test (*Y-Test* / *X-Test*) utilisé pour l'évaluation des performances. Les résultats obtenus pour ces comparaisons sont présentés ci-dessous.

Le code de couleur utilisé pour lire les graphiques est présenté dans la figure suivante :

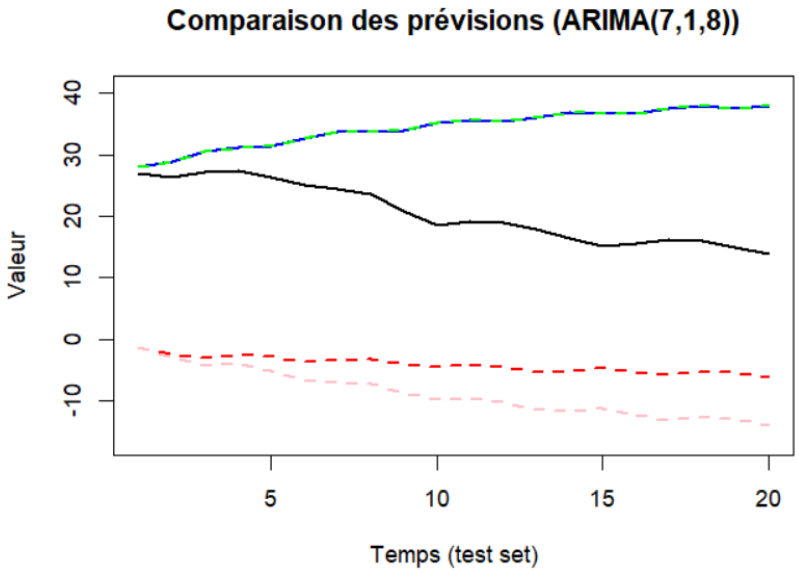


Résultats pour ARIMA(3,1,3)



	Méthode	MSE
1	forecast()	81.96946
2	predict()	81.96946
3	Forecast_Per()	21.66301
4	Forecast_Per_2()	19.84497

Résultats pour ARIMA(7,1,8)



	Méthode	MSE
1	forecast()	22.18849
2	predict()	22.18849
3	Forecast_Per()	321.01298
4	Forecast_Per_2()	298.41530

Interprétation des résultats

1. Les fonctions `arima_sim()` et `arima.sim()` génèrent deux séries présentant une allure et des fluctuations similaires, bien que leurs valeurs diffèrent. Cette différence est tout à fait normale, car elle découle des conditions initiales et de la composante MA (le bruit aléatoire) propre à chaque simulation.

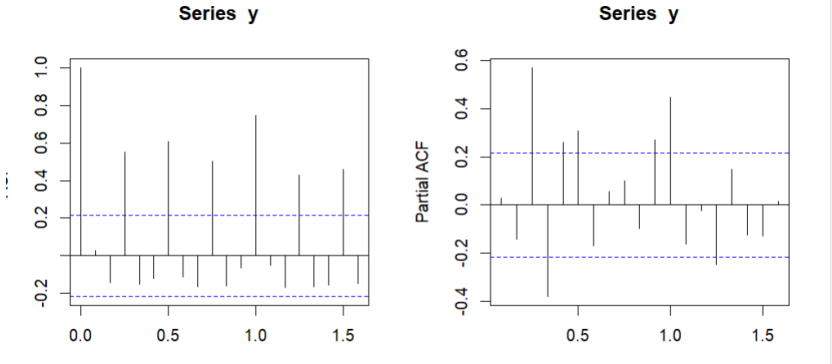
2. Nos fonctions personnalisées montrent de meilleures performances pour des ordres (p, q) faibles, comme dans le cas de l'ARIMA(3,1,3). En revanche, lorsque les ordres p et q augmentent, comme pour l'ARIMA(7,1,8), les fonctions natives de R (`forecast()` et `predict()`) tendent à produire des prévisions plus précises.

3. Pour des horizons de prévision courts ($h = 1, 2, 3$), les fonctions personnalisées reproduisent plus fidèlement la série réelle, car elles s'appuient encore sur des valeurs observées, limitant ainsi l'erreur. Cependant, pour des horizons plus longs, les prévisions successives reposent sur des valeurs déjà prédites, ce qui entraîne une accumulation des erreurs et une diminution progressive de la précision.

Partie II : Application sur une série réelle

1.4 Analyse des corrélations (ACF et PACF)

Les figures ci-dessous représentent respectivement la fonction d'autocorrélation (ACF) et la fonction d'autocorrélation partielle (PACF) de la série originale y (non différenciée).



L'analyse des fonctions d'autocorrélation (ACF) et d'autocorrélation partielle (PACF) met en évidence que la série est **non stationnaire** et **saisonnière**. En effet, l'ACF présente une décroissance lente accompagnée de pics périodiques ce qui traduit une **saisonnalité annuelle marquée** (*Remarque : L'axe des abscisses est exprimé en années (1 unité = 12 mois), indiquant ainsi des corrélations saisonnières annuelles.)* La PACF, quant à elle, affiche plusieurs pics significatifs aux premiers décalages avant de s'atténuer, indiquant la présence d'une composante **autorégressive (AR)** et d'une **tendance** dans la série. Ces observations confirment que les ventes suivent une dynamique temporelle dépendante à la fois de la tendance générale et du cycle saisonnier annuel.

Avant de procéder à la différenciation, il est nécessaire de vérifier la stabilité de la variance de la série. Pour cela, nous appliquons **la transformation de Box–Cox**

```
#Vérification de la variance (Box-Cox)
lambda <- BoxCox.lambda(y)
cat("Lambda estimé =", lambda, "\n")
# Lambda estimé = 0.848407
# On déduit donc que notre variance est déja stable
```

Dans notre cas, la valeur estimée du paramètre $\lambda = 0.848$ indique que la variance est déjà stable.

1.2 Test KPSS, différenciation et validation de la stationnarité (ACF et PACF)

Dans cette section, nous vérifions si la série temporelle est stationnaire, condition essentielle avant tout ajustement de modèle ARMA, ARIMA ou SARIMA. Le test statistique de Kwiatkowski–Phillips–Schmidt–Shin (KPSS) est utilisé pour évaluer la stationnarité de la série. En cas de non-stationnarité, nous appliquons une différenciation ordinaire et saisonnière afin de rendre la série stationnaire. Enfin, les corrélogrammes ACF et PACF de la série différenciée permettent de confirmer visuellement la stationnarité obtenue.

```
#===============Question2==============#
Test <- ur.kpss(y)
summary(Test)# KPSS (H0 : stationnaire)
#Value of test-statistic is: 1.282 > à toutes les valeurs critiques
#Série non stationnaire
```

Notre série est non stationnaire d'après le test KPSS

Afin de rendre la série stationnaire, nous avons procédé selon deux approches complémentaires. Dans la première, dite **méthode pratique comme vu en cours**, nous avons appliqué manuellement une différenciation ordinaire et saisonnière à l'aide de la fonction `diff()` afin d'éliminer respectivement la tendance et la saisonnalité. Dans la seconde, nous avons utilisé les fonctions `ndiffs()` et `nsdiffs()` du package `forecast`, qui permettent d'estimer automatiquement les ordres de différenciation d et D nécessaires pour obtenir une série stationnaire. Les deux méthodes conduisent au même résultat : une série différenciée et stationnaire, vérifiée à l'aide du test KPSS.

```
#Méthode 1: différenciation manuelle
y_diff1 <- diff(y, differences = 1)
y_diff2 <- diff(y_diff1, lag = frequency(y), differences = 1)
y_stat_P <- y_diff2

Test2 <- ur.kpss(y_stat_P)
summary(Test2)
##Notre série est devenu stationnaire vérifié grâce au Test
#Value of test-statistic is: 0.0215 < à toutes les valeurs critiques
#Du coup on accepte l'hyptohèse H0 stationnarité
#du coup vu qu'on fait une seule différenciation
#On déduit que d=1 et D=1


#Méthode 2 pour obtenir directement le nombre de différenciation fait
d <- ndiffs(y)                    # différences non saisonnières
D <- nsdiffs(y, m = frequency(y)) # différences saisonnières (données mensue
cat("d =", d, " | D =", D, "\n")

y_stat <- y
if (D > 0) y_stat <- diff(y_stat, lag = frequency(y), differences = D)
if (d > 0) y_stat <- diff(y_stat, differences = d)

Test3 <- ur.kpss(y_stat)
summary(Test3)
#Notre série est devenu stationnaire vérifié grâce au Test
#Value of test-statistic is: 0.0215 < à toutes les valeurs critiques
#Du coup on accepte l'hyptohèse H0 stationnarité
```

1.3 Identification et ajustement des modèles ARMA, ARIMA et SARIMA

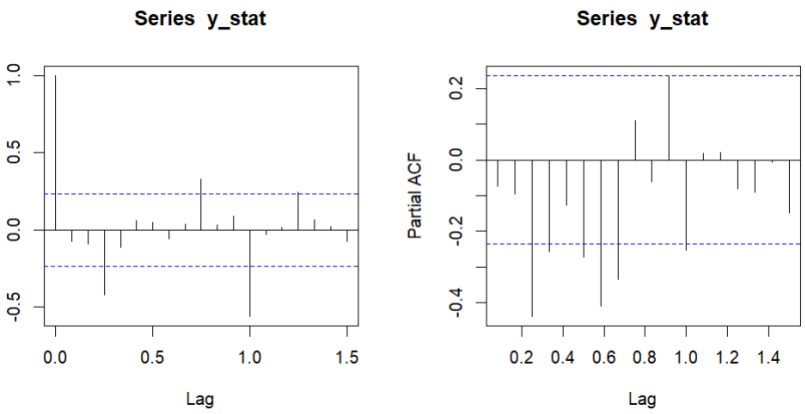
Une fois la série rendue stationnaire, l'étape suivante de la méthode de Box–Jenkins consiste à identifier et ajuster différents modèles candidats afin de déterminer celui qui décrit le mieux la dynamique des données.

Pour cela, nous avons divisé notre série temporelle en deux sous-ensembles :

- un **échantillon d'entraînement** (*train*) correspondant aux 70 premières observations ;
- un **échantillon de test** (*test*) constitué des 12 dernières observations (soit un an de données).

Cette séparation permet de valider la qualité de nos prévisions ultérieures.

Voici par ailleurs l'ACF et PACF de notre série rendu stationnaire en 1.2:



Nous avons ensuite ajusté successivement trois types de modèles :

1. un modèle **ARMA(p,q)** sur la série stationnaire y_{stat},
2. un modèle **ARIMA(p,d,q)** sur la série d'origine y,
3. un modèle **SARIMA(p,d,q)(P,D,Q)[12]** intégrant la saisonnalité annuelle.

Les plages des paramètres (p, q, P, Q) ont été fixées à partir des observations de l'ACF et de la PACF de la série stationnaire (figure ci-dessus), et afin d'éviter le surajustement :

$$p \in [0, 3], \quad q \in [0, 3], \quad P, Q \in [0, 1]$$

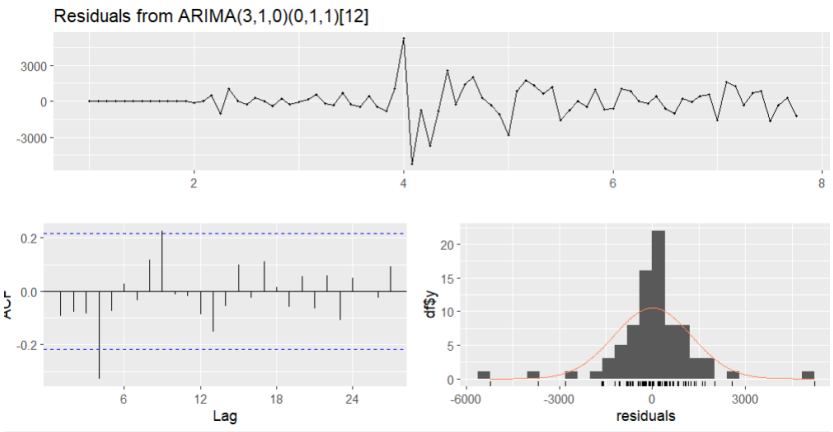
L'évaluation de chaque modèle s'est faite à l'aide du **critère d'information d'Akaike (AIC)**, qui pénalise les modèles trop complexes. Le modèle présentant la plus faible valeur d'AIC est considéré comme le plus performant pour la série donnée.

La série présentant une forte composante saisonnière, il est logique que les modèles les plus performants soient de type **SARIMA**. Ce type de modèle permet en effet de capturer à la fois la *tendance globale* et la *saisonnalité annuelle* observées dans les données. Les résultats obtenus confirment cette hypothèse, puisque les trois meilleurs modèles identifiés intègrent tous une composante saisonnière.

N°	Modèle	AIC
1	SARIMA(3,1,0)(0,1,1)[12]	1022.788
2	SARIMA(3,1,1)(1,1,1)[12]	1024.650
3	SARIMA(3,2,0)(0,1,1)[12]	1027–1030

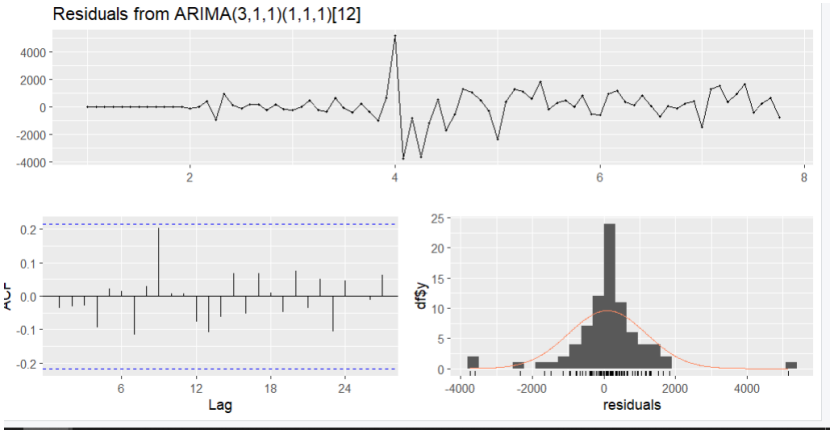
Table 1: Les trois meilleurs modèles SARIMA selon le critère AIC

Modèle 1 : SARIMA(3,1,0)(0,1,1)[12]



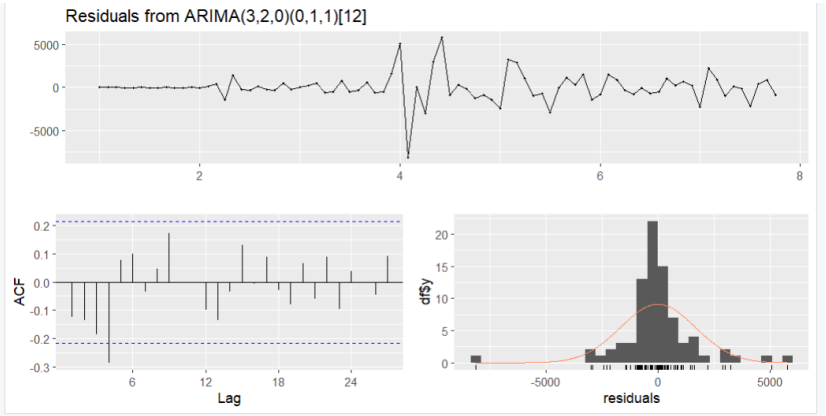
Diagnostic des résidus. Le test de Ljung–Box indique une p-valeur = *0.031* (valeur obtenue dans R), ce qui suggère une autocorrélation résiduelle restante ($p < 0{,}05$). Modèle à écarter.

Modèle 2 : SARIMA(3,1,1)(1,1,1)[12]



Diagnostic des résidus. Le test de Ljung–Box donne une p-valeur = *0.505* ($> 0{,}05$), les résidus se comportent comme un bruit blanc. Modèle validé

Modèle 3 : SARIMA(3,2,0)(0,1,1)[12]



Diagnostic des résidus. Le test de Ljung–Box affiche une p-valeur = *0.033* ($< 0{,}05$), indiquant une autocorrélation résiduelle non négligeable. Modèle à écarter.

Au vu des diagnostics, le modèle **SARIMA(3,1,1)(1,1,1)[12]** est retenu : il présente un très bon AIC et des résidus compatibles avec un bruit blanc (p-valeur Ljung–Box $> 0{,}05$).

1.4 Vérification automatique et comparaison des modèles

Dans cette dernière étape, nous avons utilisé la fonction `auto.arima()` afin de vérifier les résultats obtenus manuellement à l'aide de la méthode de Box–Jenkins. Cette fonction effectue une recherche automatique du meilleur modèle ARIMA ou SARIMA selon des critères d'information (AIC et BIC) et des tests de significativité des paramètres.

Le schéma de vérification a donc consisté à ajuster automatiquement trois types de modèles :

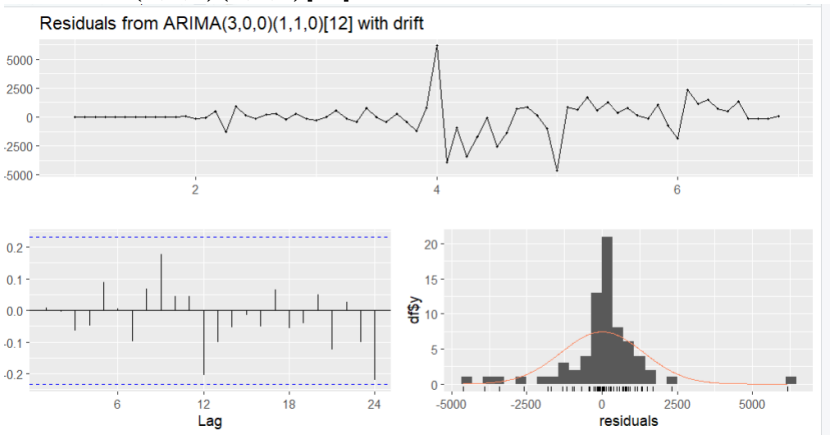
- un modèle **ARMA(p,q)** sur la série stationnaire y_{stat},
- un modèle **ARIMA(p,d,q)** non saisonnier sur la série originale y,
- un modèle **SARIMA(p,d,q)(P,D,Q)[12]** intégrant la saisonnalité.

Les résultats obtenus sont résumés dans le tableau ci-dessous :

Table 2: Modèles proposés par `auto.arima()` (jeu d'apprentissage)

Modèle	**AIC**	**BIC**	**p-val. Ljung–Box**
auto-ARMA sur y_{stat} : ARIMA(0,0,0) (moyenne nulle)	1260.67	1262.91	—
auto-ARIMA sur *train* : ARIMA(5,1,0)	1378.74	1392.23	—
auto-SARIMA sur *train* : ARIMA(3,0,0)(1,1,0)[12] + drift	1045.02	1057.48	0.4371

Le modèle sélectionné automatiquement par `auto.arima()` est un **SARIMA(3,0,0)(1,1,0)[12]** avec dérive.



L'analyse des résidus montre qu'ils sont globalement centrés autour de zéro, sans autocorrélations significatives, et que la *p-value* du test de Ljung–Box est de $0.4371 > 0.05$. Cela indique que les résidus peuvent être considérés comme du bruit blanc : le modèle est donc statistiquement correct.

Cependant, le modèle obtenu manuellement à l'aide de la méthode de Box–Jenkins, **SARIMA(3,1,1)(1,1,1)[12]**, présente des performances plus solides : ses résidus sont également non autocorrélés ($p\text{-}value > 0.05$) et il capture mieux la tendance et la saisonnalité observées dans les ventes.

Comme indiqué dans le cours, la fonction `auto.arima()` ne trouve pas toujours le modèle le plus pertinent ni le plus optimal. C'est pourquoi la sélection manuelle reste une étape essentielle dans l'application rigoureuse de la méthode de Box–Jenkins.

1.5 Prédiction des valeurs futures

Après analyse des critères AIC et de la qualité des résidus, le modèle retenu est :

SARIMA(3,1,1)(1,1,1)[12]

Ce modèle capture à la fois la tendance et la composante saisonnière annuelle observée dans les ventes mensuelles.

Nous avons généré les prévisions pour les **12 prochains mois** à l'aide de la fonction `forecast()` du package `forecast`. Le code R correspondant est présenté ci-dessous :

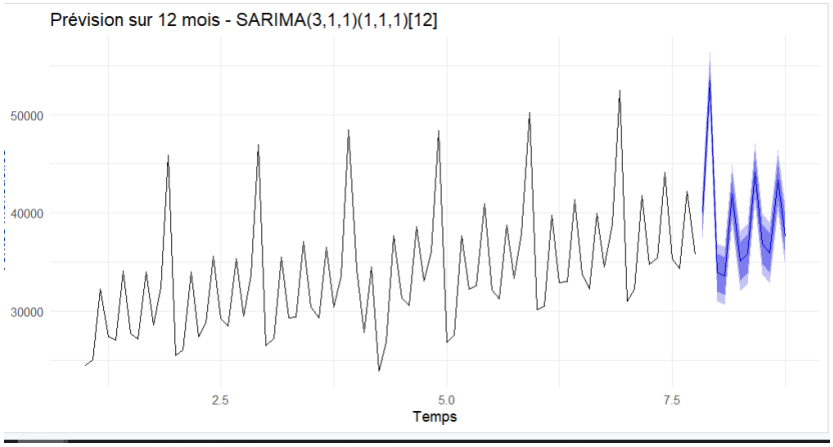
```
#On trouve que auto.arima() nous offre pas le modèle le plus pertinent
#Du coup le modele qu'on garde est bien SARIMA(3,1,1)(1,1,1)
fit_final <- Arima(y, order = c(3,1,1),
                   seasonal = list(order = c(1,1,1), period = 12))

k <- 12  # 12 mois à prédire
forecast_valeurs <- forecast(fit_final, h = k)
print(forecast_valeurs)

autoplot(forecast_valeurs) +
  ggtitle("Prévision sur 12 mois - SARIMA(3,1,1)(1,1,1)[12]") +
  xlab("Temps") + ylab("Ventes mensuelles") +
  theme_minimal()
#Le modèle ne sous- ni sur-prédit pas visiblement, ce qui confirme qu'il a b
```

Partie III : Analyse et présentation des résultats

Table 3: Comparaison entre les valeurs réelles (nov. 2022 – oct. 2023) et les prévisions SARIMA sur la période de test

Mois	Valeur réelle	Prévision (Point Forecast)	Intervalle 95% [Min, Max]
Nov 2022	38 817	39 994	[37 389 ; 42 599]
Dec 2022	52 454	53 470	[50 559 ; 56 381]
Jan 2023	31 002	33 941	[30 996 ; 36 886]
Feb 2023	32 276	33 597	[30 623 ; 36 570]
Mar 2023	41 756	41 953	[38 961 ; 44 945]
Apr 2023	34 774	35 081	[32 090 ; 38 073]
May 2023	35 466	35 812	[32 790 ; 38 834]
Jun 2023	44 151	44 081	[41 013 ; 47 149]
Jul 2023	35 376	36 861	[33 764 ; 39 957]
Aug 2023	34 366	35 937	[32 832 ; 39 041]
Sep 2023	42 162	43 328	[40 224 ; 46 433]
Oct 2023	35 800	37 594	[34 490 ; 40 699]

Le tableau ci-dessus compare les valeurs réelles observées sur la période de test (novembre 2022 à octobre 2023) avec les prévisions générées par le modèle `SARIMA(3,1,1)(1,1,1)[12]`. Les résultats montrent une bonne concordance entre les observations et les valeurs prédites. Les pics de ventes hivernaux (notamment en décembre et juin) sont correctement anticipés, et la tendance générale de la série est respectée. Les intervalles de confiance à 95 % englobent la majorité des valeurs réelles, ce qui confirme la **fiabilité et la précision du modèle** sur la période de validation.



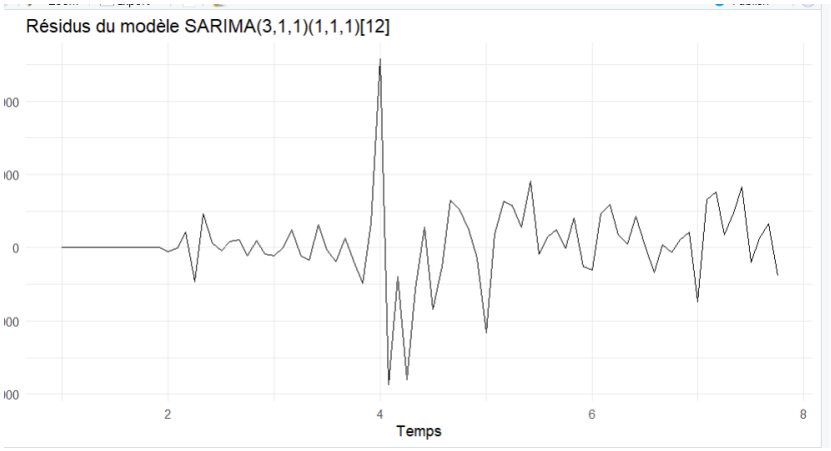
Le graphique ci-dessus présente la série observée (en noir) et les prévisions issues du modèle **SARIMA(3,1,1)(1,1,1)[12]** sur les 12 prochains mois (en bleu).

- **Structure saisonnière conservée :** les prévisions reproduisent les pics réguliers à intervalle de 12 mois, traduisant une **saisonnalité annuelle** bien captée par le modèle. Cela confirme que la composante saisonnière ($P = 1, D = 1, Q = 1$) du SARIMA modélise efficacement les fluctuations récurrentes.
- **Tendance légèrement croissante :** la moyenne des prévisions se situe légèrement au-dessus des observations précédentes, suggérant une progression attendue du volume des ventes sur la période à venir.
- **Intervalles de confiance :** les bandes bleues à 80 % et 95 % s'élargissent progressivement avec l'horizon de prévision, ce qui reflète l'incertitude croissante à long terme. Leur largeur reste néanmoins modérée, traduisant un modèle stable et bien ajusté.
- **Absence de biais :** aucune sur-prédiction ni sous-prédiction notable n'est visible, indiquant que le modèle capture correctement la tendance et la saisonnalité de la série.

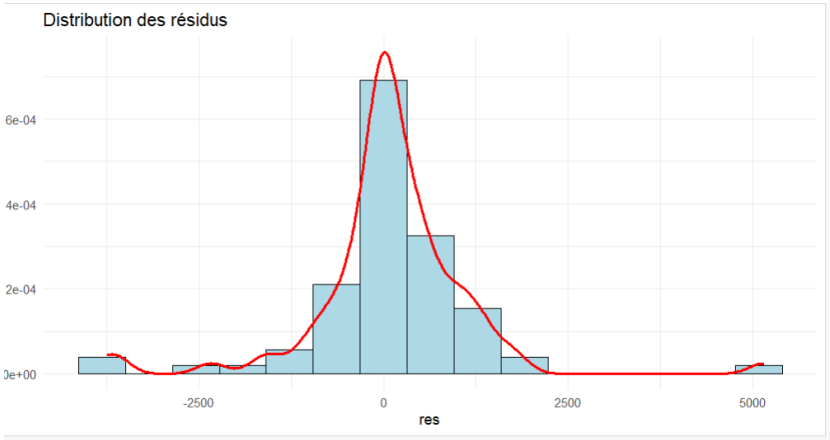
En conclusion, le modèle `SARIMA(3,1,1)(1,1,1)[12]` offre des prévisions cohérentes et fiables. Il parvient à reproduire la dynamique temporelle de la série en préservant sa saisonnalité annuelle et en anticipant une croissance modérée des ventes. Les intervalles de confiance confirment la robustesse du modèle, qui constitue ainsi une base solide pour la prévision à court terme.

1.6 Analyse diagnostique des résidus

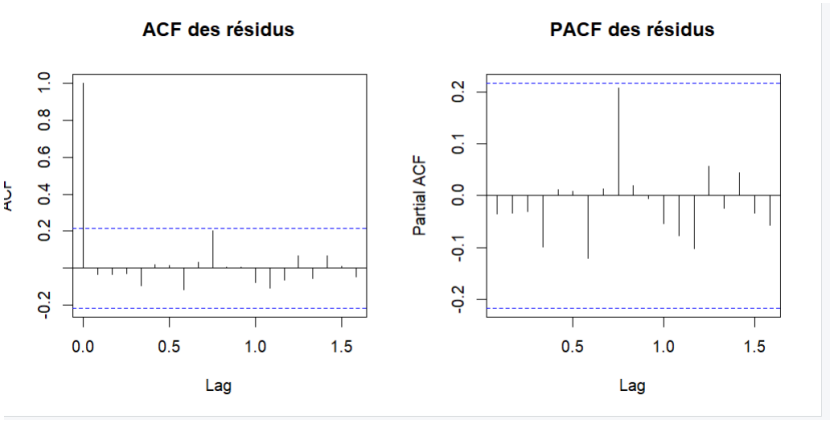
Après l'ajustement du modèle `SARIMA(3,1,1)(1,1,1)[12]`, nous procédons à une **analyse diagnostique des résidus** afin de vérifier la validité des hypothèses du modèle. L'objectif est de s'assurer que les erreurs de prévision se comportent comme un **bruit blanc**, c'est-à-dire qu'elles sont non autocorrélées, de moyenne nulle et de variance constante.



Les résidus sont centrés autour de zéro, sans tendance ni dérive apparente. On observe toutefois quelques fluctuations isolées, correspondant à des chocs ponctuels, mais la structure globale reste stable, ce qui confirme la bonne capture de la dynamique principale par le modèle.



L'histogramme des résidus (en bleu clair) est symétrique et la courbe de densité (en rouge) suit une forme proche d'une **loi normale centrée**. Cela confirme que les erreurs sont aléatoires et non biaisées, indiquant une bonne adéquation du modèle aux données.



Les fonctions d'autocorrélation (ACF) et d'autocorrélation partielle (PACF) ne présentent aucun pic significatif en dehors des bandes de confiance. Cela indique l'absence d'autocorrélation résiduelle et confirme que les résidus se comportent bien comme un bruit blanc.

Ainsi, le modèle `SARIMA(3,1,1)(1,1,1)[12]` est jugé satisfaisant et statistiquement valide ce qui confirme juste d'avantage notre choix qui a été fait préalablement pour la prévision.

Partie IV: Modèle Hybride

Dans cette dernière partie, nous explorons une approche **hybride** combinant un modèle statistique traditionnel (`SARIMA`) et des modèles d'apprentissage automatique (`XGBoost`) et (`SVR`)et aussi réseaux de neuronnes. L'objectif est d'améliorer la qualité des prévisions en capturant à la fois la composante linéaire (modélisée par SARIMA) et la composante non linéaire grâce à XGBoost ou RNN.

3.1 Principe du modèle hybride

L'idée du modèle hybride repose sur la décomposition suivante :

$$y_t = \underbrace{\hat{y}_t^{SARIMA}}_{\text{composante linéaire}} + \underbrace{\hat{y}_t^{ML}}_{\text{correction non linéaire}} + \varepsilon_t$$

Le modèle `SARIMA` capture la structure temporelle principale (tendance et saisonnalité), tandis que les modéles de maching Learning ou deep learning est entraîné sur les **résidus du modèle SARIMA** afin de corriger les erreurs systématiques restantes. Ainsi, le modèle final combine les avantages des deux approches.

Listing 4: Prévisions SARIMA et préparation des résidus pour XGBoost

```
# Prévisions SARIMA
fc_sarima  <- forecast(fit_final, h = H)
yhat_sarima_test <- as.numeric(fc_sarima$mean)

# ==== 2) Modèles sur les résidus du train ====
res_tr <- as.numeric(residuals(fit_final))

# Fonction pour créer les lags
make_lags <- function(x, L){
  E <- embed(x, L + 1)
  y <- E[,1]
  X <- E[,-1, drop = FALSE]
  list(y = y, X = X)
}

L <- 12
lagged <- make_lags(res_tr, L)

# ----- a) XGBoost -----
dtrain <- xgb.DMatrix(data = as.matrix(lagged$X), label =
    lagged$y)

params <- list(
  objective = "reg:squarederror",
  eta = 0.05,
  max_depth = 4,
```

```
  subsample = 0.8,
  colsample_bytree = 0.8,
  nthread = 2
)
```

3.2 Évaluation des performances et interprétation des métriques

Afin d'évaluer la qualité des prévisions, plusieurs indicateurs classiques d'erreur ont été calculés sur la période de test ($h = 12$ mois). Les métriques retenues sont les suivantes :

- **RMSE (Root Mean Squared Error)** :

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}$$

  Il mesure l'erreur quadratique moyenne. Les grandes erreurs y sont fortement pénalisées. Plus la valeur du RMSE est faible, plus le modèle est précis.

- **MAE (Mean Absolute Error)** :

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t|$$

  Il représente l'erreur absolue moyenne, interprétable directement dans les unités de la série (*en millions de £*).

- **MAPE (Mean Absolute Percentage Error)** :

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right|$$

  Exprimé en pourcentage, il indique l'erreur moyenne relative. Cependant, il est sensible aux valeurs proches de zéro.

- **sMAPE (Symmetric MAPE)** :

$$sMAPE = \frac{100}{n}\sum_{t=1}^{n}\frac{2|y_t - \hat{y}_t|}{|y_t| + |\hat{y}_t|}$$

  Il s'agit d'une version symétrique du MAPE, plus stable lorsque les valeurs réelles sont faibles.

- **MASE (Mean Absolute Scaled Error)** :

$$MASE = \frac{\frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t|}{\frac{1}{T-m}\sum_{t=m+1}^{T}|y_t - y_{t-m}|}$$

  Il compare l'erreur du modèle à celle d'un modèle naïf saisonnier ($m = 12$ ici). Si $MASE < 1$, le modèle fait mieux que le modèle naïf.

Les résultats obtenus sur les 12 derniers mois sont présentés dans le tableau ci-dessous :

Table 4: Métriques d'évaluation sur la période de test (12 mois)

Modèle	**RMSE**	**MAE**	**MAPE (%)**	**sMAPE (%)**	**MASE**
SARIMA	1363	1116	3.13	3.05	0.626
SARIMA–XGBoost	1471	1301	3.50	3.42	0.730
SARIMA–SVR	2320	2079	5.501	5.320	1.1575

SARIMA seul (RMSE 1363, MAE 1116)

- Capte correctement la **tendance** et la **saisonnalité** de la série.
- Les résidus sont proches du **bruit blanc**, donc peu d'information exploitable par un modèle secondaire.

SARIMA + XGBoost (RMSE 1471, MAE 1301)

- XGBoost essaie de modéliser les résidus du SARIMA.
- Comme les résidus sont presque aléatoires, le modèle se **sur-ajuste au bruit**, ce qui dégrade légèrement la performance.

SARIMA + SVR (RMSE 2320, MAE 2079)

- SVR est sensible aux **hyperparamètres** et à la **variance** des données.
- En appliquant SVR sur des résidus presque aléatoires, le modèle **amplifie le bruit** au lieu de l'exploiter, expliquant la forte dégradation.

Cela s'explique par le fait que les résidus du modèle SARIMA sont déjà assimilables à un **bruit blanc**, donc dépourvus de structure exploitable par le modèle `XGBoost` ou par SVR. On en déduit que lorsque les résidus d'un modèle saisonnier sont déjà aléatoires, ajouter un modèle complexe pour les prédire peut empirer les performances.

Conclusion : le modèle **SARIMA(3,1,1)(1,1,1)[12]** est retenu comme modèle final, car il capture efficacement la tendance et la saisonnalité de la série, avec des prévisions stables et précises.

Conclusion générale

Ce mini-projet s'inscrivait dans le cadre du cours de *Modélisation et prévision de séries temporelles* et avait pour objectif de mettre en pratique la méthodologie de Box–Jenkins sur une série réelle. En partant des ventes mensuelles du commerce de détail au Royaume-Uni (hors carburants), il nous a permis de suivre rigoureusement toutes les étapes d'une démarche de modélisation complète : de l'analyse exploratoire à la prévision, en passant par la stationnarisation, le choix des paramètres, la validation statistique et la comparaison de modèles.

Cet exercice s'est révélé particulièrement formateur car il a offert une **mise en application concrète** d'outils souvent perçus comme abstraits dans le cadre théorique du cours. Il a permis d'acquérir une compréhension beaucoup plus intuitive des notions de tendance, de saisonnalité, de bruit blanc, et de critères d'évaluation comme l'AIC ou le BIC. Travailler sur une série réelle nous a aidés à relier la théorie mathématique — parfois dense et complexe — à des problématiques économiques tangibles, en visualisant directement comment les modèles captent les régularités et les fluctuations temporelles.

Sur le plan technique, le projet nous a permis d'expérimenter plusieurs approches :

- les modèles **ARMA** et **ARIMA**, adaptés aux séries stationnaires ou présentant une tendance simple ;
- les modèles **SARIMA**, capables de modéliser les phénomènes saisonniers ;
- et enfin un modèle **hybride SARIMA–XGBoost**, combinant statistique classique et apprentissage automatique.

Ces expérimentations ont renforcé notre capacité à **interpréter les corrélogrammes**, à **justifier les paramètres** p, d, q, P, D, Q, et à **valider les hypothèses de bruit blanc sur les résidus**. Elles nous ont également familiarisés avec la logique d'évaluation croisée via un découpage *train/test* et l'usage de métriques d'erreur ($RMSE$, MAE, $MAPE$, $MASE$). D'un point de vue pratique, ce projet a donc constitué un véritable **pont entre la théorie et l'application**.

Néanmoins, certaines **limites fondamentales** demeurent inhérentes à ce type de modèle. Les approches ARIMA ou SARIMA reposent exclusivement sur les observations passées et supposent que le futur dépend uniquement de la structure interne de la série. Elles ne prennent donc pas en compte les **facteurs exogènes** ou les **événements imprévisibles**, tels que :

- les chocs économiques soudains (crises financières, pandémie, inflation exceptionnelle) ;

- les changements structurels dans les comportements de consommation ;
- ou encore les politiques publiques pouvant altérer le rythme des ventes.

Ainsi, bien que le modèle **SARIMA(3,1,1)(1,1,1)[12]** ait fourni des prévisions cohérentes et précises, il reste limité par sa nature **strictement rétrospective**. L'ajout d'un module d'apprentissage automatique comme `XGBoost` a illustré la possibilité d'intégrer une composante non linéaire, mais n'a pas apporté de gain significatif dans ce cas où les résidus étaient déjà proches du bruit blanc.

D'un point de vue pédagogique, cet exercice a été d'une grande richesse. Il nous a permis non seulement d'approfondir la compréhension des modèles ARIMA et SARIMA, mais aussi de développer des **compétences analytiques, statistiques et de programmation** applicables à de nombreux domaines : économie, ingénierie, data science ou finance. La manipulation des données, la validation empirique et la visualisation des résultats ont renforcé notre autonomie méthodologique.

En somme, ce mini-projet nous a permis de concrétiser la théorie à travers une série réelle et de mieux appréhender la puissance comme les limites des modèles de prévision temporelle. Il démontre que, même si la modélisation statistique du passé reste un outil précieux pour anticiper l'avenir, elle doit toujours être complétée par une réflexion critique sur les facteurs extérieurs et les ruptures structurelles susceptibles d'influencer les données.