



République Tunisienne
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
Université de Carthage
École Supérieure de la Statistique et de
l'Analyse de l'Information



RAPPORT DE PROJET DE FIN D'ANNÉE

Modélisation et prévision de la demande du tourisme international

Par GHADA ABDERRAHIM
SAFA BOUZAYENE



Encadrant : Mme Amira GASMI
Année Universitaire : 2022 - 2023

Remerciement

Le travail présenté dans cette période a été effectué dans le cadre du projet de fin d'année à l'École Supérieure de la Statistique et de l'Analyse de l'Information de Tunis.

Au terme de ce projet, nous souhaitons exprimer notre profonde gratitude et notre immense respect envers Madame Amira Guesmi pour sa disponibilité, ses avis éclairés et ses judicieux conseils.

Nous tenons également à exprimer notre reconnaissance à tous les enseignants et administrateurs de l'École Supérieure de la Statistique et de l'Analyse de l'Information de Tunis pour leur soutien et leur contribution à notre formation et notre projet.

Table des matières

1	Cadre général du projet	1
1	Contexte et problématique	1
2	Objectif	2
3	Conclusion	2
2	Environnement de travail	3
1	Environnement matériel	3
2	Environnement logiciel	3
3	Méthodologie	5
1	Prétraitement et analyse de la série temporelle	5
1.1	Chargement des données	5
1.2	Analyse de données	6
1.3	Détection des outliers	6
1.4	Conversion des données en série temporelle	7
1.5	Visualisation de la serie	7
2	Détection de la saisonnalité	8
2.1	Analyse graphique de la saisonnalité	8
2.2	Test de saisonnalité	10
2.3	Conclusion	11
3	Désaisonnalisation	11
3.1	Rapport à la moyenne mobile	11
3.2	Différenciation saisonnière	12
4	Détection de la stationnarité	15
4.1	Test ADF sur la série brute	16
4.2	Test ADF sur la série non saisonnière	18
5	La modélisation	20
5.1	Modèle de régression linéaire	20
5.2	Modèle ARIMA	22
5.3	Modèle de lissage exponentiel simple	25
5.4	Modèle SARIMA	27

5.5	Modèle de lissage exponentiel saisonnier	29
5.6	Modèle combiné	31
6	Prévision	32
7	Conclusion	34
Bibliographie		35
1	Logiciel R	36
2	Logiciel Python	47

Table des figures

3.1	Python : Les 5 premières lignes de la série	5
3.2	Python : Statistiques descriptives	6
3.3	Python : Boxplot	6
3.4	La série temporelle : Tourists	7
3.5	Python : Visualisation de la série temporelle	7
3.6	R : Seasonal Plot	8
3.7	R : Polar Seasonal Plot	9
3.8	R : Graphique de corrélation avec un décalage de 12 mois	9
3.9	Eviews : ACF et PACF	10
3.10	R : isSeasonal test	11
3.11	isSeasonal	12
3.12	EViews : Test de Canova & Hansen	13
3.13	EViews/R : Hegy-test	14
3.14	R : Série désaisonnalisée	15
3.15	R : Seasonal plot	15
3.16	EViews : ADF-test sur la série en niveau	16
3.17	EViews : ADF-test sur la série en différence	17
3.18	R : ADF-test sur la série non saisonnière	18
3.19	R : Ordre de différenciation	18
3.20	R : Serie différenciée	19
3.21	R : test de Phillips-Perron	19
3.22	R : Modèle de régression linéaire	21
3.23	R : Synthèse du modèle	21
3.24	R : Comparaison de la série observée et des valeurs ajustées du modèle	21
3.25	Résultats du test de Portmanteau	22
3.26	R : ACF et PACF de la série	23
3.27	Série temporelle avec prévision de test	23
3.28	R : Test de portmanteau	23
3.29	R : différenciation et filtre de différenciation inverses	24
3.30	Python : Lissage exponentiel simple	26

3.31	Test de portmanteau	26
3.32	R : Modèle SARIMA	28
3.33	test de Portmanteau	28
3.34	Détermination de la meilleure combinaison	29
3.35	Python : Lissage exponentiel saisonnier	30
3.36	test de Portmanteau	30
3.37	Modèle combiné	31
3.38	Le test de portmanteau	32

Chapitre 1

Cadre général du projet

1 Contexte et problématique

Le tourisme est un secteur clé de l'économie thaïlandaise depuis de nombreuses années. Selon les données du ministère du Tourisme et des Sports, la Thaïlande a accueilli environ 15,9 millions de touristes en 2010, mais ce chiffre a considérablement augmenté au fil des années pour atteindre 32,6 millions de touristes en 2016, soit une augmentation de plus de 100 % par rapport à 2010. En outre, selon les données de l'Organisation mondiale du tourisme, la Thaïlande est également devenue un choix de destination de plus en plus populaire pour les touristes internationaux au cours de cette période. En 2010, la Thaïlande se classait au 18ème rang mondial en termes de nombre de touristes internationaux, mais elle a progressé pour atteindre la 10ème place en 2016, dépassant des pays comme la Malaisie, l'Autriche et la Grèce.

Le tourisme en provenance d'Europe a également contribué de manière significative à la croissance du secteur touristique en Thaïlande au cours des dernières années. Selon les données du Ministère du Tourisme et des Sports, en 2016, la Thaïlande a accueilli environ 6,6 millions de visiteurs européens, représentant une augmentation de 6,4 % par rapport à l'année précédente.

En lien avec cette forte croissance du tourisme en provenance d'Europe vers la Thaïlande, notre choix de base de données consiste en une série mensuelle représentant le nombre d'arrivées de touristes européens en Thaïlande de 2010 à 2016. Cette série de données peut nous aider à mieux comprendre l'évolution du tourisme en provenance d'Europe vers la Thaïlande au cours de cette période de forte croissance.

2 Objectif

La modélisation et la prévision de la demande du tourisme international est un enjeu majeur pour les acteurs de l'industrie touristique, les gouvernements et les organisations internationales. Dans ce projet, nous nous intéressons à la modélisation de la demande du tourisme international pour la Thaïlande, plus précisément en ce qui concerne le nombre d'arrivées de touristes européens, la modélisation et la prévision du tourisme international sont essentielles pour permettre une gestion efficace du secteur du tourisme.

Les méthodes de modélisation et de prévision peuvent aider les acteurs de l'industrie touristique, les gouvernements et les organisations internationales à comprendre les tendances du marché, à anticiper les fluctuations saisonnières et les événements externes, et à prendre des décisions éclairées pour le développement touristique.

3 Conclusion





Dans ce chapitre, nous avons présenté le cadre général du projet de fin d'année intitulé "Modélisation et prévision de la demande du tourisme international". Nous avons expliqué le contexte et la problématique, ainsi que les objectifs et les motivations de ce projet. Dans le chapitre suivant, nous allons présenter la méthodologie qui sera utilisée pour réaliser ce projet.

Chapitre 2

Environnement de travail

Dans ce chapitre, nous présentons l'environnement de travail que nous avons utilisé pour mener à bien notre projet. Cet environnement est composé à la fois d'un environnement matériel et d'un environnement logiciel.

1 Environnement matériel

PC 1 :	Système d'exploitation :	Mémoire vive :	Processeur :
		 20 GO	

Ces ordinateurs ont été choisis pour leur puissance de calcul et leur capacité à traiter des données lourdes.

2 Environnement logiciel

La réalisation de ce projet oblige une utilisation des logiciels qui seront installés et utilisés tout au long du travail :



R est une plateforme open source pour l'analyse statistique et la science des données, qui est soutenue par la R Foundation for Statistical Computing. Il s'agit à la fois d'un langage de programmation et d'un logiciel qui offre une variété de fonctions statistiques et graphiques pour l'analyse de données. La R Foundation for Statistical Computing est une organisation à but non lucratif qui développe et maintient R, et encourage l'utilisation de ce logiciel dans la communauté scientifique.



Nous avons choisi EViews pour sa précision dans l'application des tests statistiques.

EViews est un logiciel populaire pour la génération de graphiques.



Python offre une grande variété de bibliothèques spécialisées dans la modélisation et la prévision, elles offrent des outils pour effectuer des analyses statistiques et des modélisations de séries temporelles.

Chapitre 3

Méthodologie

Dans ce chapitre, nous détaillons les différentes étapes que nous avons suivies pour réaliser notre projet de modélisation et prévision de la demande du tourisme international.

1 Prétraitement et analyse de la série temporelle

1.1 Chargement des données

La première étape de notre méthodologie consiste à charger les données sur le tourisme international à partir de la base de données nommée "touriste". Le tableau ci-dessous montre un aperçu des cinq premières lignes de la base de données :

year tourists		
month		
2010-01-01	2010	569275
2010-02-01	2010	526584
2010-03-01	2010	482138
2010-04-01	2010	311384
2010-05-01	2010	192360

FIGURE 3.1 – Python : Les 5 premières lignes de la série

La base de données contient des informations sur les arrivées mensuelles de touristes de l'Europe vers Thaïlande pour la période allant de janvier

2010 jusqu'à décembre 2016.

1.2 Analyse de données

Nous avons effectué quelques statistiques descriptives pour mieux comprendre nos données :

	year	month	tourists
count	84.000000	84.000000	84.000000
mean	2013.000000	6.500000	471341.857143
std	2.012012	3.472786	174081.667755
min	2010.000000	1.000000	192360.000000
25%	2011.000000	3.750000	312719.000000
50%	2013.000000	6.500000	433864.500000
75%	2015.000000	9.250000	623084.250000
max	2016.000000	12.000000	849006.000000

FIGURE 3.2 – Python : Statistiques descriptives

Ces données représentent un tableau avec 84 entrées pour les années allant de 2010 à 2016 et pour chaque mois de l'année. La colonne "tourists" représente le nombre d'arrivées de touristes européens en Thaïlande pour chaque mois donné. Les statistiques de base indiquent que la moyenne du nombre de touristes est d'environ 471 342 avec un écart-type de 174 082, tandis que le minimum et le maximum sont respectivement de 192 360 et 849 006. Les quartiles indiquent que la moitié des données se situe entre 312 719 et 623 084 touristes. Les données montrent également que le nombre de touristes varie selon les mois de l'année, avec des pics durant la haute saison touristique et des creux durant la saison basse.

1.3 Détection des outliers

Nous avons également effectué une analyse pour détecter la présence d'éventuels outliers dans nos données. Le graphique ci-dessous montre la distribution des données :

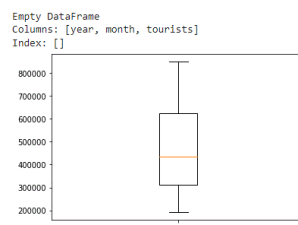


FIGURE 3.3 – Python : Boxplot

Nous pouvons conclure qu'il n'y a pas de valeurs aberrantes dans notre base de données.

1.4 Conversion des données en série temporelle

Nous avons ensuite converti nos données en une série temporelle. Le graphe ci-dessous montre la série temporelle obtenue :

```
month
2010-01-01    569275
2010-02-01    526584
2010-03-01    482138
2010-04-01    311384
2010-05-01    192360
...
2016-08-01    437962
2016-09-01    313290
2016-10-01    447962
2016-11-01    608558
2016-12-01    777957
Name: tourists, Length: 84, dtype: int64
```

FIGURE 3.4 – La série temporelle : Tourists

1.5 Visualisation de la serie

En nous concentrant sur la période de 2010 à 2016, nous avons créé un graphe montrant l'évolution du nombre de touristes au fil du temps. Cette représentation graphique permet de visualiser les variations de la série dans le temps et d'identifier les tendances et les saisons touristiques.

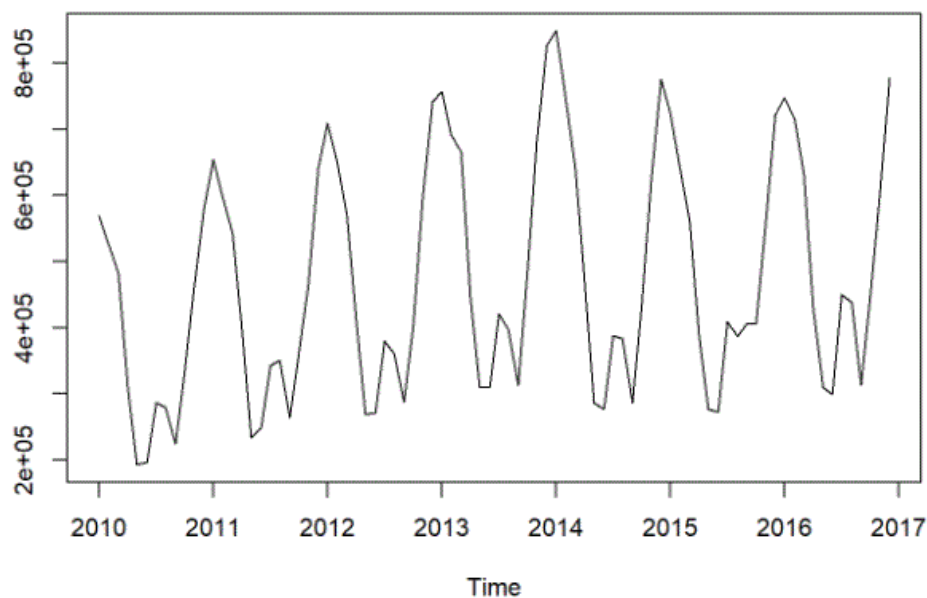


FIGURE 3.5 – Python : Visualisation de la série temporelle

Le graphe ci-dessus montre clairement une tendance à la hausse du nombre de touristes entre 2010 et 2016, avec une légère diminution en 2015 et 2016. Nous pouvons également observer une saisonnalité avec une

augmentation du nombre de touristes pendant les mois d'hiver , suivi d'une baisse pendant les mois d'été.

2 Détection de la saisonnalité

2.1 Analyse graphique de la saisonnalité

Pour détecter la saisonnalité dans notre série temporelle, nous avons utilisé plusieurs graphes :

a. Seasonal plot

Le seasonal plot est un type de graphique qui montre la tendance saisonnière dans les données en affichant les valeurs pour chaque saison sur une même ligne, pour chaque année.

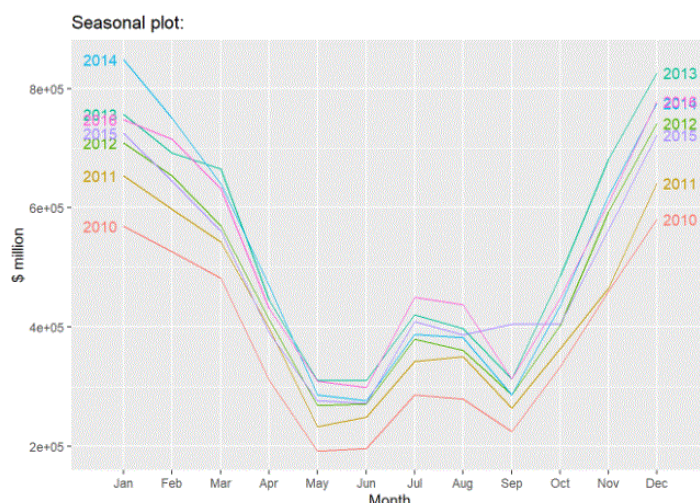


FIGURE 3.6 – R : Seasonal Plot

D'après ce graphe, nous pouvons observer une évolution similaire pour chaque année, ce qui peut indiquer la présence d'une variation saisonnière dans notre série temporelle.

b. Polar Seasonal plot

Ce type de graphique représente les variations saisonnières dans les données sous forme de motifs circulaires.

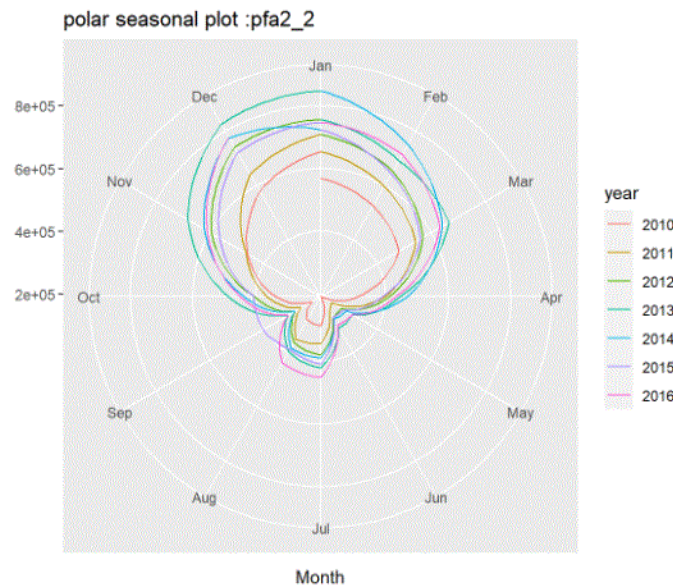


FIGURE 3.7 – R : Polar Seasonal Plot

D'après ce graphe, nous pouvons observer un motif régulier, ce qui peut indiquer qu'il existe une variation saisonnière dans les données. Les variations saisonnières sont des fluctuations qui se répètent à intervalles réguliers, dans notre cas l'intervalle est l'année.

c. Graphique de corrélation

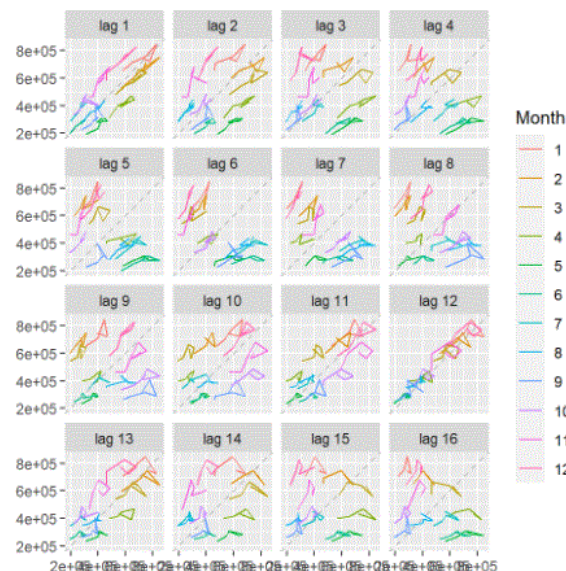


FIGURE 3.8 – R : Graphique de corrélation avec un décalage de 12 mois

Nous avons observé une forte corrélation avec un décalage de 12 mois, ce qui peut indiquer que la série de données présente une saisonnalité annuelle ou un ordre saisonnier de 12 mois.

d. Les autocorrélations

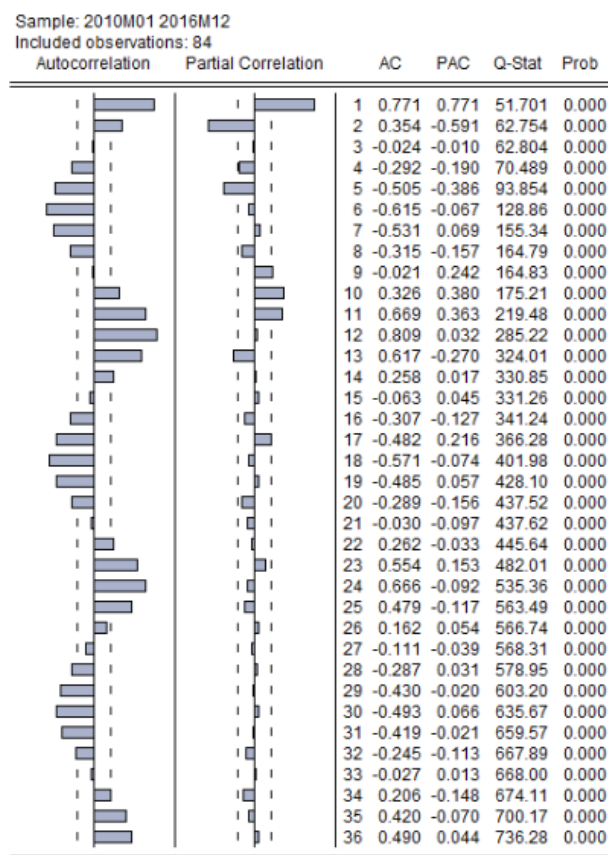


FIGURE 3.9 – Eviews : ACF et PACF

En examinant l'ACF (autocorrelation function) et le PACF (partial autocorrelation function), il est possible de détecter graphiquement la présence de saisonnalité et d'identifier les modèles ARIMA appropriés. Le corrélogramme d'une série saisonnière a souvent une forme sinusoïdale.

2.2 Test de saisonnalité

Ensuite, nous avons choisi la fonction `isSeasonal` pour déterminer si notre série temporelle est saisonnière ou non. Le test `isSeasonal` est une méthode couramment utilisée pour déterminer si une série temporelle est saisonnière ou non.

Cette méthode repose sur l'hypothèse nulle selon laquelle la série n'est pas saisonnière.

Elle utilise des techniques de décomposition de la série en composantes de tendance, saisonnières et résiduelles.

```
library(seastests)

# Charger la série chronologique

# Tester si la série est saisonnière
is_seasonal <- isSeasonal(pfa2_2)

# Afficher le résultat
if (is_seasonal) {
  print("La série est saisonnière.")
} else {
  print("La série n'est pas saisonnière.")
}

## [1] "La série est saisonnière."
```

FIGURE 3.10 – R : isSeasonal test

Dans notre cas, l'application de la méthode isSeasonal nous a permis de conclure que notre série est saisonnière, ce qui suggère qu'elle présente des variations saisonnières significatives.

2.3 Conclusion

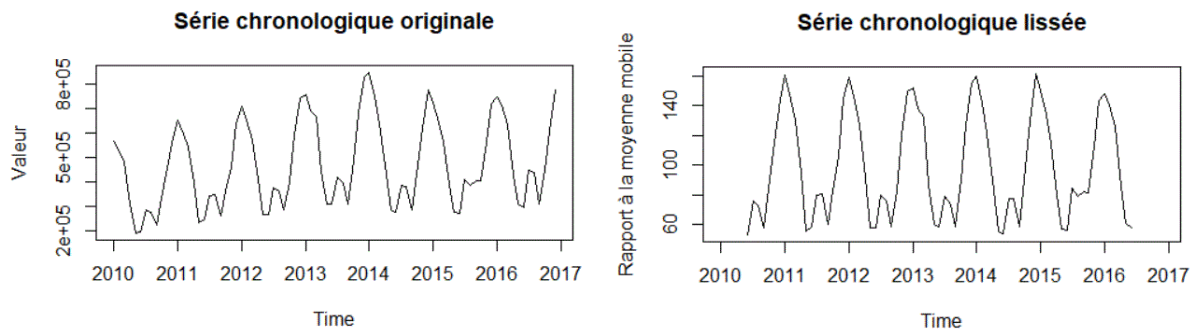
En conclusion, nous pouvons confirmer la présence de saisonnalité dans nos données.

3 Désaisonnalisation

3.1 Rapport à la moyenne mobile

La méthode du ratio à la moyenne mobile consiste à diviser les valeurs mensuelles de la série étudiée (X_t) par la figure de la moyenne mobile correspondante pour chaque mois (MA_t), et à l'exprimer en % pour générer le ratio à la moyenne mobile :

$$M_{\text{ratio}} = \frac{X_t}{MA_t} \cdot 100 \quad (3.1)$$



Cependant, malgré cette méthode, la saisonnalité peut encore rester dans la série. Pour vérifier si la série est saisonnière, nous avons utilisé la fonction `isSeasonal` pour confirmer la présence de saisonnalité dans la série. Notez que même après avoir appliqué la méthode de rapport à la moyenne mobile, la série peut toujours présenter une saisonnalité.

```
library(seastests)

# Charger la série chronologique

# Tester si la série est saisonnière
is_seasonal <- isSeasonal(ratio)

# Afficher le résultat
if (is_seasonal) {
  print("La série est saisonnière.")
} else {
  print("La série n'est pas saisonnière.")
}

## [1] "La série est saisonnière."
```

FIGURE 3.11 – `isSeasonal`

3.2 Différenciation saisonnière

3.2.1 Détection de la présence de racines unitaires saisonnières

a. Test de Canova and Hansen

Ce test permet de détecter la présence d'une racine unitaire saisonnière dans la série de données. Il a pour hypothèses :

- (H_0) Inexistence d'une racine unitaire saisonnière à une fréquence spécifiée.
- (H_1) Existence d'une racine unitaire saisonnière.

Seasonal Unit Root Test for TOURISTS_SA
Method: Canova-Hansen
Null Hypothesis: No unit root at specified frequencies
Periodicity (Seasons): 12
Non-Seasonal Deterministics: Constant
Seasonal Deterministics: Seasonal dummies
Sample Size: 84

		Significance Level		
Joint LM Statistic	K	1%	5%	10%
1.959357	11	3.270	2.750	2.490

Note: The K restrictions used in the regression are identified below.

FIGURE 3.12 – EViews : Test de Canova & Hansen

Ce test a donné une statistique égale à 1.956 inférieur à la statistique critique au seuil de 5% qui égale à 2.750 et donc on ne peut qu'accepter l'hypothèse alternative et conclure l'existence de la racine unitaire saisonnière.

b. Test de Hegy

Le test de traditional Hegy (Hylleberg, Engle, Granger, Yoo) permet de déterminer la présence ou non d'une racine unitaire saisonnière dans la série de données. il a pour hypothèses :

(H_0) :La série avait une racine unitaire saisonnière à une fréquence spécifiée.

(H_1) :Il n'y avait pas de racine unitaire saisonnière.

Method: Traditional HEGY
 Null Hypothesis: Unit root at specified frequency
 Periodicity (Seasons): 12
 Non-Seasonal Deterministics: Constant
 Seasonal Deterministics: Seasonal dummies
 Lag Selection: 0 (Automatic: AIC, maxlags=12)
 Sample Size: 72

		Significance Level		
	Test Stat.	1%	5%	10%
Frequency 0	-2.093553			
n=60		-3.36	-2.79	-2.51
n=80		-3.35	-2.80	-2.51
n=72*		-3.35	-2.80	-2.51
Frequency 2PI/12 and 22PI/12	4.982002			
n=60		2.69	1.15	0.74
n=80		2.71	1.14	0.73
n=72*		2.70	1.14	0.73
Frequency 4PI/12 and 20PI/12	8.450512			
n=60		2.69	1.15	0.74
n=80		2.71	1.14	0.73
n=72*		2.70	1.14	0.73
Frequency 6PI/12 and 18PI/12	8.076930			
n=60		2.69	1.15	0.74
n=80		2.71	1.14	0.73
n=72*		2.70	1.14	0.73
Frequency 8PI/12 and 16PI/12	7.454870			
n=60		2.69	1.15	0.74
n=80		2.71	1.14	0.73
n=72*		2.70	1.14	0.73
Frequency 10PI/12 and 14PI/12	6.139117			
n=60		2.69	1.15	0.74
n=80		2.71	1.14	0.73
n=72*		2.70	1.14	0.73
Frequency PI	-2.443863			
n=60		-3.36	-2.79	-2.51
n=80		-3.35	-2.80	-2.51
n=72*		-3.35	-2.80	-2.51

```

HEGY test for unit roots
data: pfa2_2
      statistic p-value
t_1      -1.9062  0.0345 *
t_2      -2.4471  0.0464 *
F_3:4      5.0638  0.0421 *
F_5:6      8.4991  0.0037 **
F_7:8      8.1168  0.0048 **
F_9:10     7.4474  0.0076 **
F_11:12    6.1716  0.0191 *
F_12:12    11.9479  0 ***
F_1:12     11.5594 3e-04 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Deterministic terms: constant + trend + seasonal dummies
Lag selection criterion and order: fixed, 0
P-values: based on response surface regressions

```

FIGURE 3.13 – EViews/R : HEGY-test

Nous avons constaté, à partir de l'output d'EViews et de R, la présence d'une racine unitaire saisonnière à toutes les fréquences, car les t-statistiques sont supérieures à la valeur critique calculée au seuil de 5

Ainsi, nous avons conclu que l'hypothèse nulle était vraie et que notre série a une racine unitaire saisonnière.

3.2.2 Le filtre de différenciation

D'après les résultats précédents, on doit appliquer le filtre suivant pour retirer les fluctuations saisonnières :

$$\Delta s = (1 - L) \sum_{i=0}^{11} L^i \quad (3.2)$$

.

3.2.3 Vérification de la désaisonnalisation

En observant le graphique ci-dessous de la série différenciée et le résultat du test isSeason, on peut conclure que notre série est devenue non saisonnière et on peut confirmer ce résultat en utilisant le Seasonal plot et en le

comparant au plot de la base initiale, on voit une évolution aléatoire pour chaque année, ce qui pourrait suggérer l'absence de variations saisonnières dans notre série temporelle.



FIGURE 3.14 – R : Série désaisonnalisée

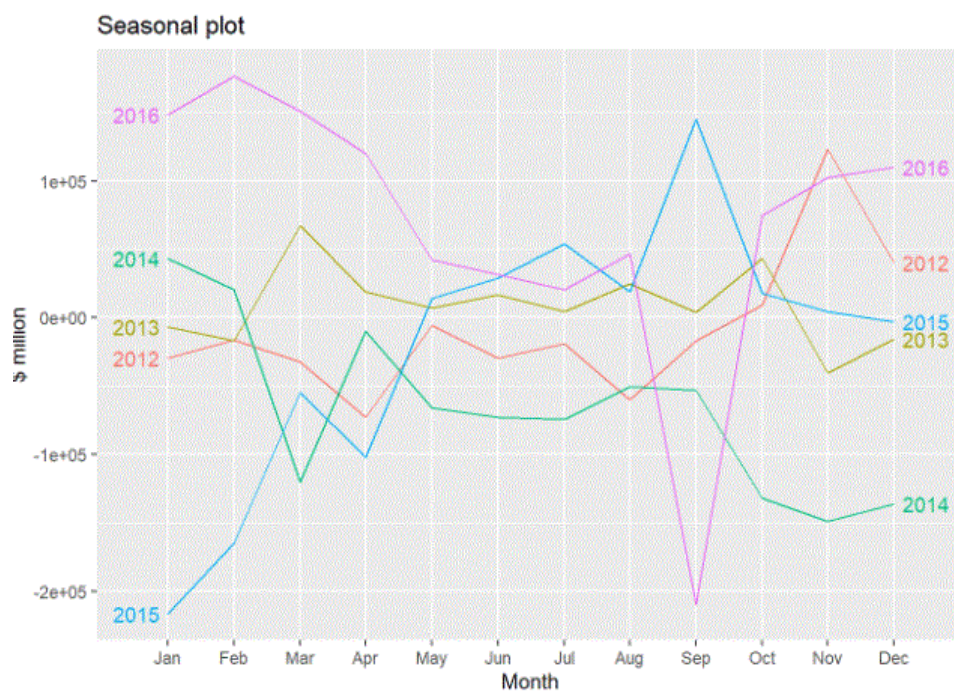


FIGURE 3.15 – R : Seasonal plot

4 Détection de la stationnarité

La stationnarité est une propriété clé des séries temporelles qui indique que les propriétés statistiques d'une série ne changent pas avec le temps. Cette propriété est importante car elle permet de simplifier considérablement l'analyse et la modélisation de la série temporelle.

Si une série n'est pas stationnaire, il peut être difficile de faire des prévisions précises ou de comprendre les relations entre les variables.

Pour cela, on a effectué le test le plus puissant qui est le test de Dickey-Fuller Augmenté (ADF) qui a pour hypothèses :

- (H_0) : La série temporelle n'est pas stationnaire (elle présente une racine unitaire).
- (H_1) : La série temporelle est stationnaire.

4.1 Test ADF sur la série brute

Il est nécessaire d'appliquer le test ADF sur la série brute car nous avons besoin de cette information pour la modélisation à venir.

Null Hypothesis: TOURISTS has a unit root Exogenous: Constant, Linear Trend Lag Length: 11 (Automatic - based on SIC, maxlag=11)					Null Hypothesis: TOURISTS has a unit root Exogenous: Constant Lag Length: 11 (Automatic - based on SIC, maxlag=11)				
			t-Statistic	Prob.*				t-Statistic	Prob.*
Augmented Dickey-Fuller test statistic					Augmented Dickey-Fuller test statistic				
Test critical values:					Test critical values:				
1% level					1% level				
5% level					5% level				
10% level					10% level				
*Mackinnon (1996) one-sided p-values.					*Mackinnon (1996) one-sided p-values.				
Augmented Dickey-Fuller Test Equation Dependent Variable: D(TOURISTS) Method: Least Squares Date: 03/28/23 Time: 00:05 Sample (adjusted): 2011M01 2016M12 Included observations: 72 after adjustments					Augmented Dickey-Fuller Test Equation Dependent Variable: D(TOURISTS) Method: Least Squares Date: 03/28/23 Time: 00:14 Sample (adjusted): 2011M01 2016M12 Included observations: 72 after adjustments				
Variable	Coefficient	Std. Error	t-Statistic	Prob.	Variable	Coefficient	Std. Error	t-Statistic	Prob.
TOURISTS(-1)	-0.271379	0.176191	-1.522971	0.1332	TOURISTS(-1)	-0.336190	0.120678	-2.785850	0.0072
D(TOURISTS(-1))	-0.173063	0.196717	-0.879752	0.3826	D(TOURISTS(-1))	-0.107088	0.144265	-0.742302	0.4608
D(TOURISTS(-2))	-0.475106	0.173026	-2.745833	0.0080	D(TOURISTS(-2))	-0.417486	0.127636	-3.270916	0.0018
D(TOURISTS(-3))	-0.487492	0.165392	-2.947487	0.0046	D(TOURISTS(-3))	-0.433047	0.123132	-3.516920	0.0008
D(TOURISTS(-4))	-0.474047	0.154863	-3.061073	0.0033	D(TOURISTS(-4))	-0.426801	0.121481	-3.513313	0.0009
D(TOURISTS(-5))	-0.399337	0.149888	-2.664244	0.0100	D(TOURISTS(-5))	-0.356464	0.121800	-2.926627	0.0049
D(TOURISTS(-6))	-0.649930	0.125993	-5.158463	0.0000	D(TOURISTS(-6))	-0.613003	0.101109	-6.062798	0.0000
D(TOURISTS(-7))	-0.489910	0.131861	-3.563675	0.0007	D(TOURISTS(-7))	-0.437892	0.114321	-3.830362	0.0003
D(TOURISTS(-8))	-0.570787	0.121816	-4.685661	0.0000	D(TOURISTS(-8))	-0.543869	0.108419	-5.016383	0.0000
D(TOURISTS(-9))	-0.581820	0.112793	-5.158302	0.0000	D(TOURISTS(-9))	-0.561305	0.104299	-5.381705	0.0000
D(TOURISTS(-10))	-0.713345	0.101068	-7.058082	0.0000	D(TOURISTS(-10))	-0.699415	0.096483	-7.249077	0.0000
D(TOURISTS(-11))	-0.343847	0.127632	-2.694054	0.0092	D(TOURISTS(-11))	-0.331336	0.124325	-2.665089	0.0099
C	153756.2	72864.15	2.110177	0.0392	C	175430.1	58003.86	3.024456	0.0037
@TREND("2010M01")	-190.1694	382.5800	-0.497071	0.6210					
R-squared	0.883133	Mean dependent var	2741.542		R-squared	0.882635	Mean dependent var	2741.542	
Adjusted R-squared	0.856939	S.D. dependent var	115584.1		Adjusted R-squared	0.858764	S.D. dependent var	115584.1	
S.E. of regression	43717.90	Akaike info criterion	24.38157		S.E. of regression	43438.05	Akaike info criterion	24.35804	
Sum squared resid	1.11E+11	Schwarz criterion	24.82425		Sum squared resid	1.11E+11	Schwarz criterion	24.76911	
Log likelihood	-863.7365	Hannan-Quinn criter.	24.55780		Log likelihood	-863.8895	Hannan-Quinn criter.	24.52169	
F-statistic	33.71463	Durbin-Watson stat	1.719069		F-statistic	36.97545	Durbin-Watson stat	1.722744	
Prob(F-statistic)	0.000000								

FIGURE 3.16 – EViews : ADF-test sur la série en niveau

D'après la première figure présentée ci-dessus, on peut remarquer que la t-statistic=-1.5229 est supérieur à la t-calculée (5%) = -3.4734, en plus la p-valeur = 0.8126 > 5%, on peut donc accepter l'hypothèse nulle et conclure que notre série contient une racine unité. En d'autres termes, on

peut aboutir à la non-significativité de la tendance de la série puisque la p-valeur de la tendance qui est égale à 0.621 est supérieur au seuil de 5%.

En somme, on a une série non stationnaire avec une tendance non significative c'est à dire un processus DS.

On refait donc le test sur la série en niveau mais sans tendance : D'après la 2ème figure, on peut remarquer que la p-value de la constante = 0.0037 < 5% et donc la série est affectée par une constante alors notre série est non stationnaire du type DS avec dérive.

Stationnarisation de la série brute

On applique une autre fois le test ADF mais cette fois-ci sur la série en différence :

Null Hypothesis: D(TOURISTS) has a unit root				
Exogenous: Constant				
Lag Length: 11 (Automatic - based on AIC, maxlag=11)				
			t-Statistic	Prob.*
Augmented Dickey-Fuller test statistic			-3.792808	0.0046
Test critical values:	1% level		-3.525618	
	5% level		-2.902953	
	10% level		-2.588902	
*MacKinnon (1996) one-sided p-values.				
Augmented Dickey-Fuller Test Equation				
Dependent Variable: D(TOURISTS,2)				
Method: Least Squares				
Date: 04/03/23 Time: 23:59				
Sample (adjusted): 2011M02 2016M12				
Included observations: 71 after adjustments				
Variable	Coefficient	Std. Error	t-Statistic	Prob.
D(TOURISTS(-1))	-4.255438	1.121976	-3.792808	0.0004
D(TOURISTS(-1),2)	3.022316	1.035971	2.917377	0.0050
D(TOURISTS(-2),2)	2.649691	0.941358	2.814754	0.0067
D(TOURISTS(-3),2)	2.246841	0.849986	2.643386	0.0105
D(TOURISTS(-4),2)	1.904190	0.755895	2.519119	0.0145
D(TOURISTS(-5),2)	1.620090	0.662428	2.445686	0.0175
D(TOURISTS(-6),2)	1.167423	0.568319	2.054169	0.0445
D(TOURISTS(-7),2)	0.866212	0.477177	1.815284	0.0747
D(TOURISTS(-8),2)	0.499819	0.385025	1.298147	0.1994
D(TOURISTS(-9),2)	0.167566	0.290500	0.576821	0.5663
D(TOURISTS(-10),2)	-0.280800	0.200749	-1.398764	0.1672
D(TOURISTS(-11),2)	-0.427861	0.123360	-3.468393	0.0010
C	7674.019	5419.957	1.415882	0.1622
R-squared	0.892294	Mean dependent var		1351.310
Adjusted R-squared	0.870010	S.D. dependent var		115427.3
S.E. of regression	41616.38	Akaike info criterion		24.27434
Sum squared resid	1.00E+11	Schwarz criterion		24.68863
Log likelihood	-848.7389	Hannan-Quinn criter.		24.43909
F-statistic	40.04175	Durbin-Watson stat		2.196570
Prob(F-statistic)	0.000000			

FIGURE 3.17 – EViews : ADF-test sur la série en différence

D'après ce résultat, on peut observer que la t-statistic=-3.7928 est devenue inférieure à la t-calculée (5%)=-2.90295 et la p-value = 0.0046 < 5% , on accepte donc l'hypothèse alternative et on conclut que notre série est

devenue stationnaire avec un ordre de différenciation égale à 1, par la suite, notre variable "Tourists" est intégrée d'ordre 1.

4.2 Test ADF sur la série non saisonnière

De meme, on applique le test ADF sur la série ajustée de la saisonnalité :

```
##
## Title:
## Augmented Dickey-Fuller Test
##
## Test Results:
## PARAMETER:
## Lag Order: 12
## STATISTIC:
## Dickey-Fuller: -2.3513
## P VALUE:
## 0.4328
##
```

FIGURE 3.18 – R : ADF-test sur la série non saisonnière

D'après la figure ci-dessus, on a obtenu une p-valeur de 0.4328 qui est plus élevée que 5% , on accepte donc l'hypothèse nulle qui consiste en la non-stationnarité de la série.

Stationnarisation de la série non saisonnière

On commence tout d'abord par déterminer l'ordre de différenciation nécessaire pour obtenir une série stationnaire. En utilisant la fonction `ndiffs`, on a déterminé que l'ordre de différenciation était de 1.

```
# Utiliser la fonction ndiffs pour déterminer l'ordre de différenciation
on
diff_order <- ndiffs(diff.x, test="adf")
cat("L'ordre de différenciation nécessaire est ", diff_order)

## L'ordre de différenciation nécessaire est 1
```

FIGURE 3.19 – R : Ordre de différenciation

Nous avons ensuite appliqué la différenciation d'ordre 1 en utilisant la fonction `diff()` et par la suite nous avons visualisé notre nouvelle série différenciée à l'aide d'un graphique.

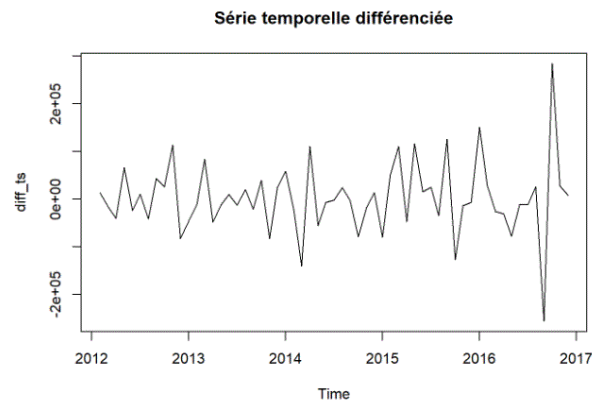


FIGURE 3.20 – R : Serie différenciée

Vérification de la stationnarité

Nous avons vérifié la stationnarité de la série à l'aide du test Phillips-Perron. Ce test est utilisé pour tester l'hypothèse nulle de non-stationnarité de la série.

```
library(tseries)
# Effectuer le test de Phillips-Perron
pp_test <- pp.test(diff_ts)

## Phillips-Perron Unit Root Test
## alternative: stationary
##
## Type 1: no drift no trend
## lag Z_rho p.value
## 3 -71.6 0.01
## ----
## Type 2: with drift no trend
## lag Z_rho p.value
## 3 -71.6 0.01
## ----
## Type 3: with drift and trend
## lag Z_rho p.value
## 3 -71.7 0.01
## -----
## Note: p-value = 0.01 means p.value <= 0.01
```

FIGURE 3.21 – R : test de Phillips-Perron

Le résultat du test Phillips-Perron nous a permis de confirmer que notre série est stationnaire après avoir appliqué la différenciation d'ordre 1, puisque les résultats pour les trois types de tests sont similaires, avec une p-value très faible (≤ 0.01). Cela suggère que la série temporelle est stationnaire, car la p-value est inférieure à la valeur du seuil généralement utilisée de 0.05.

Ainsi, nous concluons que la série est stationnaire.

5 La modélisation

La modélisation est une méthode essentielle pour prédire les tendances futures et comprendre les comportements des variables étudiées.

En économie, il existe de nombreuses techniques de modélisation telles que la régression, l'analyse de séries temporelles, et les méthodes de lissage. Dans ce projet, nous envisageons de construire plusieurs modèles pour prédire les variations saisonnières de la demande de tourisme thaïlandais.

Nous utiliserons les méthodes suivantes : [Le modèle de régression](#), [le modèle ARIMA](#), [le lissage exponentiel simple](#), [le modèle SARIMA](#), et [le lissage exponentiel saisonnier](#).

Chacune de ces méthodes a ses avantages et inconvénients et nous comparerons leurs performances pour déterminer celle qui fournit les prévisions les plus précises pour notre étude.

5.1 Modèle de régression linéaire

Nous avons commencé par le modèle de régression linéaire, qui est le plus simple. On essaiera de créer le modèle suivant :

$$Touristes_t = a + \sum_{i=1}^{11} \alpha_i \times 1_{\{mois=i\}} + \epsilon_t \quad (3.3)$$

La première étape consiste à créer deux ensembles de données distincts : un ensemble de formation pour entraîner le modèle (train) un ensemble de test pour évaluer la performance du modèle (test).

```
train <- window(pfa2_2, end = c(2015,12))
test <- window(pfa2_2, start = c(2016,1))
```

Ensuite, nous avons utilisé la fonction `tslm` pour ajuster le modèle avec saisonnalité :

```
library(TSA)
mod_trend_lin=tslm(pfa2_2~season(pfa2_2))
summary(mod_trend_lin)
```

FIGURE 3.22 – R : Modèle de régression linéaire

Dans cette fonction, "season(pfa2_2)" représente les indicatrices des différentes saisons. Ensuite, à l'aide de la fonction summary(), on peut observer les différentes valeurs estimées des différents paramètres :

```
Residuals:
    Min       1Q   Median       3Q      Max
-146592  -26737    4938   38372  133139

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    715867     23994   29.835  < 2e-16 ***
season(pfa2_2)February  -61543     33933   -1.814  0.073895 .
season(pfa2_2)March    -130895     33933   -3.857  0.000247 ***
season(pfa2_2)April    -307390     33933   -9.059  1.62e-13 ***
season(pfa2_2)May     -447835     33933  -13.198  < 2e-16 ***
season(pfa2_2)June    -448335     33933  -13.212  < 2e-16 ***
season(pfa2_2)July    -333364     33933   -9.824  6.21e-15 ***
season(pfa2_2)August  -345083     33933  -10.170  1.44e-15 ***
season(pfa2_2)September -416873     33933  -12.285  < 2e-16 ***
season(pfa2_2)October  -305203     33933   -8.994  2.14e-13 ***
season(pfa2_2)November -145625     33933   -4.292  5.44e-05 ***
season(pfa2_2)December    7842      33933    0.231  0.817880

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 63480 on 72 degrees of freedom
Multiple R-squared:  0.8846,    Adjusted R-squared:  0.867
F-statistic: 50.19 on 11 and 72 DF,  p-value: < 2.2e-16
```

FIGURE 3.23 – R : Synthèse du modèle

On a une bonne qualité d'ajustement du modèle puisque R^2 égale à 0.8846.

Enfin, nous avons représenté les valeurs ajustées du modèle sur le même graphe que notre série :

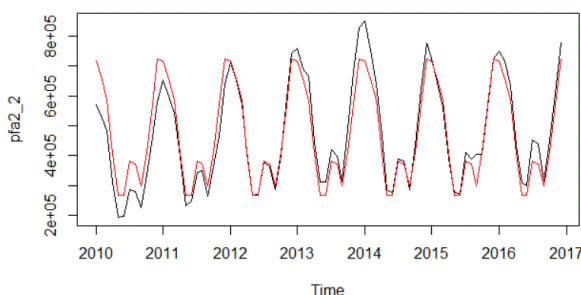


FIGURE 3.24 – R : Comparaison de la série observée et des valeurs ajustées du modèle

Ensuite, nous avons effectué un test de Portmanteau pour vérifier si les résidus du modèle de régression sont corrélés. Le test de Portmanteau est un test statistique qui vérifie si les corrélations d'ordre supérieur entre les résidus sont nulles. Les hypothèses nulle et alternative du test de Portmanteau sont respectivement :

- H0 : Les corrélations des résidus sont nulles
- H1 : Il existe au moins une corrélation des résidus différente de zéro

Box-Ljung test

```
data: resid
X-squared = 163.95, df = 10, p-value < 2.2e-16
```

FIGURE 3.25 – Résultats du test de Portmanteau

Comme nous pouvons le voir, la p-valeur est inférieure au seuil de 0,05, ce qui suggère qu'il existe une corrélation entre les résidus, indiquant que le modèle de régression ne peut pas être adéquat pour les données observées.

5.2 Modèle ARIMA

Après avoir réalisé le modèle de régression, nous avons procédé à la modélisation ARIMA :

ARIMA (AutoRegressive Integrated Moving Average) est un modèle qui combine une partie AR (AutoRegressive) et une partie MA (Moving Average) :

$$Y_t = c + \sum_{i=1}^p \varphi_i Y_{t-i} + \sum_{j=1}^q \theta_j \varepsilon_{t-j} + \varepsilon_t \quad (3.4)$$

Avec :

Y_t est la variable dépendante,

c est une constante,

φ_i sont les coefficients d'autorégression de l'ordre p ,

Y_{t-i} sont les valeurs passées de la série,

θ_j sont les coefficients de la moyenne mobile de l'ordre q ,

ε_{t-j} sont les résidus passés,

ε_t est le résidu courant.

Nous avons utilisé l'ACF (Autocorrelation Function) et le PACF (Partial Autocorrelation Function) pour déterminer les ordres p et q du modèle. Pour cela, nous avons tracé les graphes de l'ACF et du PACF (voir les figures ci-dessous).

Et à partir de la première valeur significative de l'ACF et le PACF qui se situe au-delà de l'intervalle de confiance, nous avons sélectionné $p=1$ et $q=1$ comme ordres appropriés pour notre modèle.

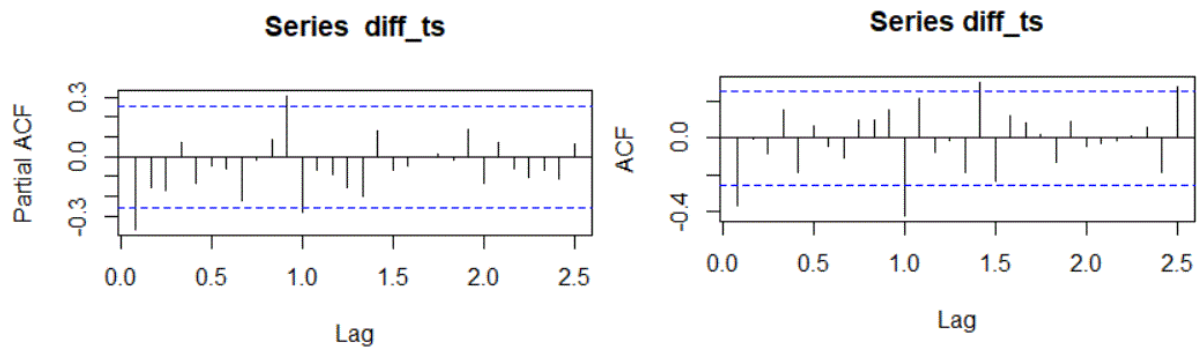


FIGURE 3.26 – R : ACF et PACF de la série

On a ensuite procédé à un modèle $\text{ARIMA}(1,0,1)$:

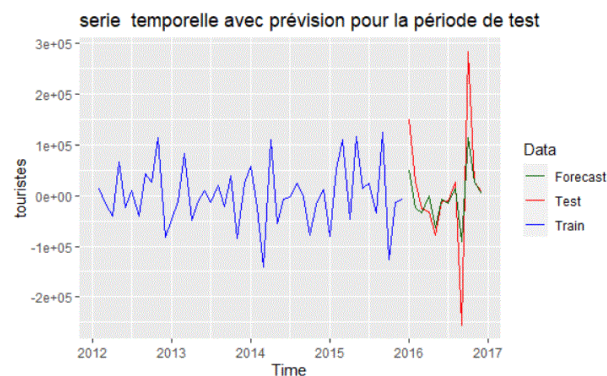


FIGURE 3.27 – Série temporelle avec prévision de test

Nous avons également effectué un test de portmanteau pour évaluer la présence de corrélation résiduelle dans le modèle $\text{ARMA}(1,1)$. Les résultats du test sont présentés dans l'image ci-dessous.

Box-Ljung test

```
data: resid
X-squared = 6.8285, df = 10, p-value = 0.7415
```

FIGURE 3.28 – R : Test de portmanteau

la p-valeur du test de portmanteau égale à 0.7415 qui est supérieure au seuil de 5%, cela suggère que les résidus ne sont pas corrélés de manière significative, ce qui indique que le modèle est adéquat pour les données observées.

Après avoir validé notre modèle et vérifié son adéquation, il est recommandé de revenir à la base initiale pour confirmer les résultats et assurer la fiabilité des prévisions.

Pour revenir à la série de données initiale, nous avons appliqué les filtres inverses de saisonnalité et de stationnarité que nous avons utilisés lors de la phase de prétraitement des données.

```

```{r}
Appliquer la différenciation cumulée pour inverser la différenciation
diff_inv <- cumsum(diff(serie, lag = 12, differences = 1)) + serie[12]
Appliquer le filtre inverse pour réintroduire la saisonnalité
saison <- stats::filter(diff_inv, filter = c(rep(1, 11), 0, 0), sides = 1) + mean(serie)
```

```

FIGURE 3.29 – R : différenciation et filtre de différenciation inverses

Pour évaluer la performance prévisionnelle de notre modèle ARMA(1,1), nous avons utilisé différents critères tels que le MAE (Mean Absolute Error), le MAPE (Mean Absolute Percentage Error) et le RMSE (Root Mean Squared Error) pour les horizons de prévision $h=1, 6$ et 12 .

1. Le MAE : représente la moyenne de la valeur absolue des erreurs de prévision :

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

où n est le nombre d'observations, y_i est la valeur réelle de la i -ème observation et \hat{y}_i est la valeur prédite correspondante.

2. Le MAPE : mesure l'erreur de prévision en pourcentage de la valeur réelle :

$$MAPE = \frac{1}{n \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right|} \times 100\%$$

où n est le nombre d'observations, y_i est la valeur réelle de la i -ème observation et \hat{y}_i est la valeur prédite correspondante.

3. Le RMSE : mesure l'écart quadratique moyen entre les valeurs prédites et réelles :

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Où : y_i est la valeur réelle de la série à l'instant i , \hat{y}_i est la valeur prédite de la série à l'instant i , n est le nombre total d'observations dans la série temporelle.

Les résultats de ces critères sont présentés dans le tableau ci-dessous :

| MAE | MAPE | RMSE |
|----------------|--------------|----------------|
| H=1, 100377.18 | H=1, 66.60% | H=1, 100377.18 |
| H=6, 35273.95 | H=6, 72.9% | H=6, 48630.12 |
| H=12, 47324.36 | H=12, 58.25% | H=12, 76706.33 |

d'après le tableau ce modèle de prévision présente des valeurs grandes pour MAE, MAPE et RMSE, on peut le considérer alors comme étant non précis et donc il n'est pas performant.

5.3 Modèle de lissage exponentiel simple

La méthode de lissage exponentiel simple est une technique de prévision couramment utilisée qui consiste à attribuer des poids décroissants aux observations passées d'une série chronologique.

Le principe est de calculer une moyenne pondérée des observations passées.

La formule de lissage exponentiel simple est donnée par :

$$\hat{y}_{t+1} = \alpha y_t + (1 - \alpha) \hat{y}_t \quad (3.5)$$

Où \hat{y}_{t+1} est la valeur prédite pour le temps $t + 1$, y_t est la valeur observée pour le temps t , et \hat{y}_t est la prédiction pour le temps t basée sur la moyenne pondérée des observations passées, α est un paramètre de lissage qui prend des valeurs entre 0 et 1 et contrôle la rapidité à laquelle les poids décroissent exponentiellement.

Nous avons essayé cette méthode pour prédire notre variable explicative "touristes" et avons utilisé la fonction "HoltWinters(train, alpha=0.8)" dans R pour ajuster le modèle de lissage exponentiel simple.

Ensuite, nous avons représenté les valeurs prédites du modèle sur le même graphe que notre série chronologique "test" pour évaluer sa performance.

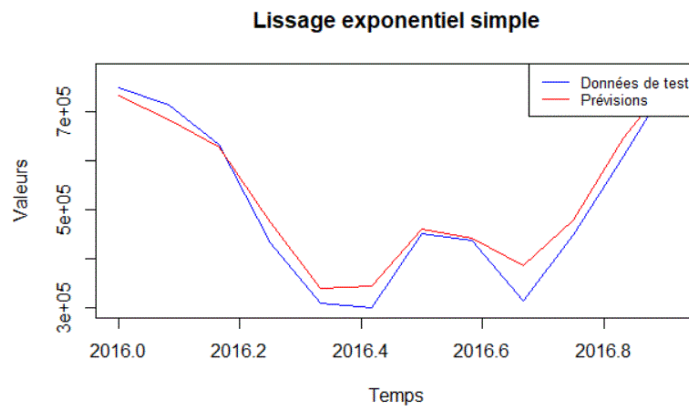


FIGURE 3.30 – Python : Lissage exponentiel simple

Nous avons également effectué un test de portmanteau pour évaluer la présence de corrélation résiduelle dans le modèle lissage exponentiel simple. Les résultats du test sont présentés dans l'image ci-dessous.

```
Box-Ljung test

data: resid
X-squared = 7.368, df = 10, p-value = 0.6903
```

FIGURE 3.31 – Test de portmanteau

Le résultat du test montre que la p-value est supérieure à 0,05, ce qui suggère que le modèle est adéquat pour les données observées.

Par conséquent, ce modèle peut être utilisé pour faire des prévisions à court terme pour la variable explicative étudiée.

Après la validation et la vérification de l'adéquation de notre modèle, nous avons procédé à la récupération de la série de données initiale en appliquant les filtres inverses de saisonnalité et de stationnarité utilisés lors de la phase de prétraitement des données.

Pour évaluer la performance prévisionnelle de notre modèle de lissage exponentiel simple, nous avons utilisé les mêmes critères que pour le modèle ARMA(1,1) tels que le MAE, le MAPE et le RMSE pour l'horizon de prévision $h=12$.

Les résultats de ces critères sont également présentés dans le tableau ci-dessous :

| MAE | MAPE | RMSE |
|----------------|-------------|----------------|
| H=12, 27570.79 | H=12, 6.85% | H=12, 34122.53 |

Le modèle de lissage exponentiel simple que nous avons utilisé a donné une MAPE (Mean Absolute Percentage Error) de 6,85%.

Cela indique que le modèle peut produire des erreurs de prévision d'environ 6,85% pour les 12 prochaines périodes.

Donc il peut être considéré comme un niveau d'erreur raisonnablement faible, ce qui suggère que le modèle est capable de prédire avec précision les valeurs futures de la variable explicative.

5.4 Modèle SARIMA

Par la suite, nous avons effectué un modèle SARIMA sur notre série brute après la différenciation d'ordre 1 (voir page 16). En fait, le modèle SARIMA (Seasonal Autoregressive Integrated Moving Average) est une extension de la formule de ARIMA pour modéliser les données saisonnières. La formule de SARIMA est définie par :

$$\phi_p(B)(1-B)^d\Phi_P(B)\Delta^D\delta_t = \theta_q(B)(1-B)^d\Theta_Q(B)\epsilon_t \quad (3.6)$$

où :

- B est l'opérateur de retard,
- d est le degré de différenciation,
- ϕ_p et θ_q sont les polynômes autorégressif et moyenne mobile de degré p et q , respectivement,
- Φ_P et Θ_Q sont les polynômes autorégressif et moyenne mobile saisonniers de degré P et Q , respectivement,
- Δ^D est l'opérateur de différenciation saisonnière d'ordre D ,
- δ_t est la série temporelle modélisée,
- ϵ_t est un bruit blanc gaussien.

En utilisant la fonction `auto.arima`, nous avons trouvé que le meilleur modèle était celui d'ordre

$(3,0,0)(1,1,0)[12]$.

Ce modèle est composé d'un polynôme autorégressif d'ordre 3, d'un polynôme autorégressif saisonnier d'ordre 1, d'un terme de différenciation saisonnière d'ordre 1, la saisonnalité a été fixée à 12, ce qui indique que la série suit un cycle saisonnier annuel.

Pour visualiser les performances du modèle sélectionné, nous avons créé un graphique qui affiche les valeurs prédites sur la partie test de notre série chronologique. Ce graphique est présenté ci-dessous :

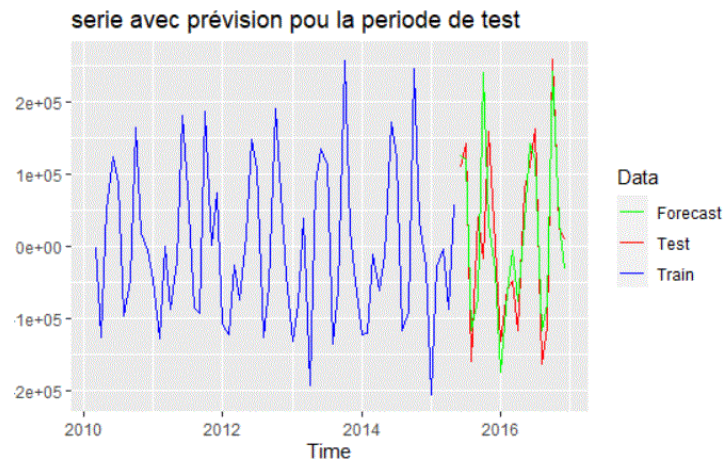


FIGURE 3.32 – R : Modèle SARIMA

Nous pouvons observer que les valeurs prédites suivent assez bien la tendance des valeurs réelles de la série.

Nous avons également réalisé un test de portmanteau pour évaluer la présence de corrélation résiduelle dans le modèle SARIMA. Les résultats de ce test sont présentés dans l'image ci-dessous :

Box-Ljung test

```
data: resid
X-squared = 9.3675, df = 10, p-value = 0.4976
```

FIGURE 3.33 – test de Portmanteau

Selon le résultat du test, la valeur de la p-value (0,4976) est supérieure à 0,05, ce qui indique que le modèle SARIMA est adéquat pour les données observées. Nous pouvons donc considérer que les résidus du modèle ne présentent pas de corrélation significative.

Ensuite, nous avons récupéré la série de données initiale et calculé les critères d'information utilisés pour sélectionner ce modèle qui sont présentés dans le tableau suivant :

| MAE | MAPE | RMSE |
|----------|-------|----------|
| 21328,99 | 5,43% | 30980,85 |

Le MAPE pour le modèle SARIMA est de 5,43%, ce qui est relativement faible. Cela indique que le modèle est considéré comme précis pour la prévision de notre série chronologique.

Ainsi, ce modèle peut être utilisé pour faire des prévisions pour la variable explicative étudiée.

5.5 Modèle de lissage exponentiel saisonnier

Le lissage exponentiel saisonnier est une méthode de prévision qui permet de tenir compte des variations saisonnières. La formule du modèle de lissage exponentiel saisonnier est la suivante :

$$\hat{y}_{t+h|t} = l_t + hb_t + s_{t-m+(h-1 \bmod m)+1}$$

$$l_t = \alpha(y_t - s_{t-m}) + (1 - \alpha)(l_{t-1} + b_{t-1})$$

$$b_t = \beta(l_t - l_{t-1}) + (1 - \beta)b_{t-1}$$

$$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1 - \gamma)s_{t-m} \quad (3.7)$$

où

l_t représente le niveau au temps t .

b_t la tendance au temps t .

$s_{t+h-m(k+1)}$ la composante saisonnière pour la période h .

m la longueur de la saison.

k l'indice de la période de la saison.

Nous avons utilisé cette méthode pour prédire notre variable explicative "touristes" en utilisant la fonction "HoltWinters(train, seasonal="multiplicative", alpha=0.2, beta=0.1, gamma=0.3)" dans R pour ajuster le modèle de lissage exponentiel saisonnier avec :

- "alpha" est réglé sur 0.2, ce qui représente le coefficient de lissage pour la moyenne mobile exponentielle de niveau.
- "beta" est réglé sur 0.1, ce qui représente le coefficient de lissage pour la moyenne mobile exponentielle de tendance.
- "gamma" est réglé sur 0.3, ce qui représente le coefficient de lissage pour la moyenne mobile exponentielle de saisonnalité.

Il faut bien noter qu'on a obtenu cette combinaison des paramètres après tester différentes possibilités pour trouver celle qui donne les prévisions les plus précises pour notre série chronologique en se basant sur l'erreur quadratique moyenne (MSE) et c'est à l'aide de la fonction ets() suivante :

```
call:
ets(y = pfa2_2, model = "ZZZ", opt.crit = "mse")

Smoothing parameters:
alpha = 0.2
beta  = 0.1
gamma = 0.3
```

FIGURE 3.34 – Détermination de la meilleure combinaison

Avec :

-l'option "ZZZ" pour le paramètre "model", ce qui signifie que la fonction

a choisi les paramètres de manière automatique pour les composantes de niveau, de tendance et de saisonnalité.

Ensuite, nous avons représenté les valeurs prédites du modèle sur le même graphe que notre série chronologique "test" pour évaluer sa performance.

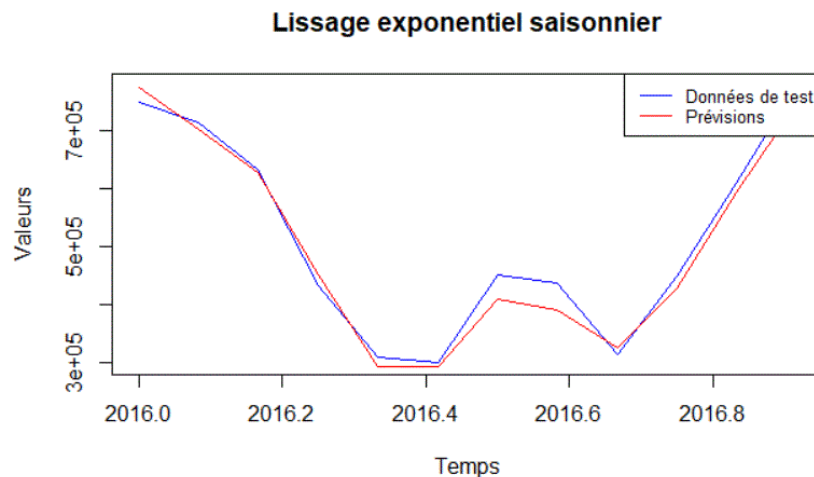


FIGURE 3.35 – Python : Lissage exponentiel saisonnier

Nous avons effectué un test de portmanteau afin d'évaluer la présence de corrélation résiduelle dans le modèle de lissage exponentiel saisonnier.

Les résultats de ce test sont présentés dans l'image ci-dessous.

Box-Ljung test

```
data: resid
X-squared = 26.891, df = 10, p-value = 0.002709
```

FIGURE 3.36 – test de Portmanteau

Comme nous pouvons le voir, la p-valeur (0.002709) est inférieure au seuil de 0,05, ce qui suggère qu'il existe une corrélation entre les résidus, indiquant que le modèle de Lissage exponentiel saisonnier ne peut pas être adéquat pour les données observées.

Finalement, nous avons évalué la performance prévisionnelle du modèle en utilisant différents critères pour un horizon de prévision de 12 périodes. Les résultats de ces critères sont présentés dans le tableau ci-dessous.

| MAE | MAPE | RMSE |
|---------------|------------|----------------|
| H=12, 21155.2 | H=12, 4.4% | H=12, 24522.57 |

Bien que le modèle de lissage exponentiel saisonnier semble donner une erreur de prévision relativement faible, avec une MAPE de 4,4%, nous ne pouvons pas conclure que ce modèle est adéquat pour la prédiction de la série chronologique étudiée.

En effet, le test de portmanteau que nous avons réalisé révèle la présence de corrélation résiduelle dans le modèle, ce qui suggère que ce modèle ne peut pas être utilisé pour la prédiction.

5.6 Modèle combiné

Nous allons maintenant combiner deux modèles, soit un modèle $ARIMA(1,0,1)$ et un modèle de lissage exponentiel simple, afin d'obtenir une prédiction plus précise. Pour ce faire, nous allons utiliser la formule suivante pour la combinaison des prévisions :

$$Combined_{Forecast} = 0.5 * ARIMA(1,0,1) + 0.5 * Holt - Winters$$

où $ARIMA(1,0,1)$ représente le modèle ARIMA et $Holt - Winters$ le modèle de lissage exponentiel simple.

Ensuite, nous avons représenté les valeurs prédites du modèle sur le même graphique que notre série chronologique "test" pour évaluer sa performance.

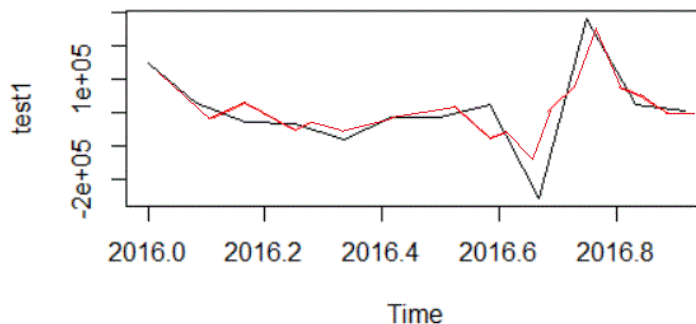


FIGURE 3.37 – Modèle combiné

Nous avons effectué un test de portmanteau afin d'évaluer la présence de corrélation résiduelle dans le modèle de lissage exponentiel saisonnier. Les résultats de ce test sont présentés dans l'image ci-dessous :

```

Box-Ljung test

data: resid
X-squared = 26.891, df = 10, p-value = 0.082709

```

FIGURE 3.38 – Le test de portmanteau

Le test de portmanteau qui indique une p-value sup rieure   0.05, ce qui signifie que la s rie r siduelle est al atoire et que notre mod le de pr diction combin  est valide.

Enfin, nous avons proc d    l  valuation de la performance pr visionnelle du mod le en utilisant divers indicateurs pour une horizon de pr vision de 12 p riodes. Les r sultats de cette  valuation sont pr sent s dans le tableau ci-dessous :

| MAE | MAPE | RMSE |
|---------|------|----------|
| 23255.2 | 9.4% | 24751.57 |

Le MAPE (Mean Absolute Percentage Error) de notre mod le est de 9.4%, ce qui signifie que les pr visions du mod le ont en moyenne une erreur relative de 9.4%.

Cela peut  tre consid r  comme un bon r sultat pour la performance pr visionnelle du mod le.

→ Conclusion

En conclusion, nous avons s lectionn  les mod les ARIMA(1,1), SARIMA(3,0,0)(1,1,0)[12], le lissage exponentiel simple et le modele combin  (ARIMA(1,1) et le lissage exponentiel simple) et pour pr dire notre variable d'int r t, en raison de leur performance  lev e lors des tests de portmanteau.

6 Pr vision

Apr s avoir effectu  la mod lisation et s lectionn  les meilleurs mod les pour notre s rie temporelle, nous pouvons maintenant passer   la phase de pr vision.

L'objectif est de faire des pr visions pour les 12 prochaines p riodes en utilisant les mod les $ARIMA(1,1)$, $SARIMA(3,0,0)(1,1,0)$ [12], le lissage exponentiel simple et le mod le combin  comme pr sent  ci-dessous :

| Date | $ARIMA(1,0,1)$ | $SARIMA(3,0,0)(1,1,0)$ | Lissage Expo. Simple | Mod le combin  |
|------------|----------------|------------------------|----------------------|----------------|
| 2017-01-01 | 783251.2 | 714065.5 | 761485.6 | 796244.8 |
| 2017-02-01 | 709814.7 | 637124.7 | 702950.7 | 700411.4 |
| 2017-03-01 | 635963.8 | 556684.2 | 617066.6 | 655121.1 |
| 2017-04-01 | 457132.0 | 382884.3 | 428197.6 | 487536.2 |
| 2017-05-01 | 295212.2 | 247516.0 | 307712.9 | 312245.3 |
| 2017-06-01 | 297016.6 | 243125.1 | 298086.6 | 315006.2 |
| 2017-07-01 | 415045.6 | 368266.4 | 441287.2 | 455288.6 |
| 2017-08-01 | 396261.5 | 351383.6 | 423048.7 | 431198.7 |
| 2017-09-01 | 330249.6 | 321658.3 | 364439.2 | 386355.1 |
| 2017-10-01 | 431708.1 | 387442.7 | 434440.4 | 435408.7 |
| 2017-11-01 | 600604.2 | 556036.4 | 593088.9 | 621012.5 |
| 2017-12-01 | 755881.7 | 711927.8 | 756679.7 | 767741.8 |

TABLE 3.1 – Pr visions pour les douze prochaines p riodes

D'apr s ce tableau, on constate que les quatre mod les de pr vision $ARIMA(1,0,1)$, $SARIMA(3,0,0)(1,1,0)$, le lissage exponentiel simple et le mod le combin  ont des pr visions diff rentes pour chaque mois.

Cependant, on peut remarquer que les quatre mod les ont des r sultats assez proches pour la plupart des mois, bien que le mod le $SARIMA(3,0,0)(1,1,0)$ donne des pr visions l g rement inf rieures aux trois autres mod les pour certains mois.

En fin de compte, la performance de chaque mod le d pendra de la pr cision de la m thode de pr vision choisie pour la s rie temporelle en question.

7 Conclusion

En conclusion, la réalisation de ce projet de fin d'année sur la prévision et la modélisation à travers l'utilisation de différents modèles a été une expérience très enrichissante.

Nous avons acquis de nouvelles compétences et des connaissances précieuses dans ce domaine, notamment en matière de collecte et d'analyse de données, ainsi que dans l'utilisation de divers modèles pour prédire les résultats futurs.

Nous avons également appris l'importance de la mise en place d'une méthodologie rigoureuse et de l'utilisation d'outils appropriés pour obtenir des résultats précis et fiables.

Nous sommes fiers des résultats que nous avons obtenus et nous sommes convaincus que cette expérience nous sera bénéfique pour notre future.

Bibliographie

- Gasmi, A. (2013). Seasonal adjustment versus seasonality modelling : Effect on tourism demand forecasting. *Advances in Management Applied Economics*, 3(4), 119-132.
- Meng, X., He, C., Fleyeh, H. (2012). Testing seasonal unit roots in data at any frequency : an HEGY approach. *Working Papers in Transport, Tourism, Information Technology and Microdata Analysis*, (2012 :08). ISSN : 1650-5581.
- Goude, Y. (2021-2022). Modélisation de séries stationnaires : MAP-STA2 : Séries chronologiques [cours]. EDF
- Fritz, R. G., Brandon, C., Xander, J. (1984). Combining time-series and econometric forecast of tourism activity. *Annals of Tourism Research*, 11(2), 219-229
- Lagnoux, A. (s. d.). Série chronologique : Prévision par lissage exponentiel [cours]. ISMAG1 - MI00141X, Université Toulouse - Jean Jaurès
- Roche, A. (2019). Modèles ARIMA et SARIMA, prédiction et choix de modèle [cours]. Executive Master Statistique et Big Data, Université de Paris-Saclay.
- Butler, R. (1998). Seasonality in tourism : Issues and implications. *The Tourist Review*, 53(1), 3-11. ISSN 0251-3102.
- Armstrong, J. S. (2001). Combining forecasts. *Marketing Papers Wharton Faculty Research*, University of Pennsylvania, ScholarlyCommons.

Annexe

1 Logiciel R

```
library(dplyr)
library(lubridate)
library(tsibble)
library(tidyverse)
library(ggplot2)
library(gridExtra)
library(grid)
library(ggthemes)
library(corrplot)
library(lmtest)
library(sandwich)
library(TSA)
library(tseries)
library(car)
library(aTSA)
library(forecast)
library(astsa)
library(urca)
library(seastests)
library(TSstudio)
library("cowplot")
library(readxl)
pfa2_2 <- read_excel("C:/Users/Ghada/Downloads/pfa2 (2).xlsx")
head(pfa2_2)
View(pfa2_2)

train=as.vector((t(as.matrix(pfa2_2[3]))))
plot(train)
pfa2_2=ts(train,start=c(2010,1),frequency = 12);
pfa2_2
```

```

plot(pfa2_2)

ggseasonplot(pfa2_2, year.labels=TRUE, year.labels.left=TRUE) +
  ylab("$ million") +
  ggtitle("Seasonal plot:")

library(ggseas)
ggseasonplot(pfa2_2, polar =TRUE)+ylab(" ") +
  ggtitle("polar seasonal plot :pfa2_2")

pfa2_2_pret <- window (pfa2_2,start=1965)
gglagplot(pfa2_2_pret)

  library(seastests)

# Charger la série chronologique

# Tester si la série est saisonnière
is_seasonal <- isSeasonal(pfa2_2)

# Afficher le résultat
if (is_seasonal) {
  print("La série est saisonnière.")
} else {
  print("La série n'est pas saisonnière.")
}

  # Calculer la moyenne mobile de la série chronologique
ma <- rollmean(pfa2_2, k = 12, fill = NA)
  # pour une moyenne mobile d'ordre 3

# Calculer le rapport entre la série et la moyenne mobile
ratio <- (pfa2_2/ ma)*100

# Afficher la série originale et la série lissée
plot(pfa2_2, main = "Série chronologique originale", ylab = "Valeur")
plot(ma, main = "Moyenne mobile d'ordre 3", ylab = "Valeur")
plot(ratio, main = "Série chronologique lissée",
ylab = "Rapport à la moyenne mobile")

diff.x <- diff(pfa2_2, differences = 2, lag = 12)

```

```

# différence première à un an

# Afficher la série différenciée
plot(diff.x, main = "Série chronologique différenciée",
ylab = "Valeur")

    ggseasonplot(diff.x, year.labels=TRUE, year.labels.left=TRUE) +
    ylab("$ million") +
    ggtitle("Seasonal plot")

#o=en comparant ce plot au plot initial il est possible de constater
une évolution aléatoire pour chaque année, ce qui pourrait suggérer
l'absence de variations saisonnières dans notre série chronologique.

ggseasonplot(pfa2_2, year.labels=TRUE, year.labels.left=TRUE) +
    ylab("touristes") +
    ggtitle("serie brut")library(fUnitRoots)

    adfTest(diff.x, lags = 12, type = "ct")

    # Utiliser la fonction ndiffs pour déterminer l'ordre
    de différenciation
diff_order <- ndiffs(diff.x, test="adf")
cat("L'ordre de différenciation nécessaire est ", diff_order)
# Appliquer une différenciation d'ordre 1
diff_ts <- diff(diff.x, differences=1)
plot(diff_ts, main="Série temporelle différenciée")

    diff_order1 <- ndiffs(diff_ts, test="adf")
cat("L'ordre de différenciation nécessaire est ", diff_order1)
cat("
    ")
library(tseries)
# Effectuer le test de Phillips-Perron
pp_test <- pp.test(diff_ts)

# Afficher les résultats du test
print(summary(pp_test))

    train <- window(pfa2_2, end = c(2015,12))
test <- window(pfa2_2, start = c(2016,1))

```

```

library(TSA)
mod_trend_lin=tslm(pfa2_2~season(pfa2_2))
summary(mod_trend_lin)

plot(pfa2_2)
lines(ts(mod_trend_lin$fitted.values,start=c(2010,1),
frequency=12),col="red")

library("tseries")

resid <- residuals(mod_trend_lin)

portmanteau_test <- Box.test(resid, lag=10, type="Ljung-Box")

print(portmanteau_test)

pred <- predict(mod_trend_lin, newdata = test)
pred_lin <- tail(pred,36)
h1=pred_lin[1]
h2=pred_lin[1:6]
h3=pred_lin[1:12]

MAE1 <- mean(abs(h1 - test[1]))
MAE2 <- mean(abs(h2 - test[1:6]))
MAE3 <- mean(abs(h3 - test[1:12]))

mape_1month <- mean(abs((test[1] - h1)/test[1])) * 100
mape_6months <- mean(abs((test[1:6] - h2)/test[1:6])) * 100
mape_12months <- mean(abs((test[1:12] - h3)/test[1:12])) * 100

RMSE1 <- sqrt(mean((h1 - test[1])^2))
RMSE2 <- sqrt(mean((h2 - test[1:6])^2))
RMSE3 <- sqrt(mean((h3 - test[1:12])^2))

# Afficher les mesures d'erreur dans un tableau
accuracy_table <- data.frame(

```

```

MAE=c(MAE1,MAE2,MAE3),
MAPE = c(mape_1month, mape_6months, mape_12months),
RMSE=c(RMSE1,RMSE2,RMSE3)
)
rownames(accuracy_table) <- c("h = 1", "h = 6", "h = 12")
accuracy_table

# Plot the autocorrelation function (ACF)
acf(diff_ts,lag=30)
# Plot the partial autocorrelation function (PACF)
pacf(diff_ts,lag=30)

train1<- window(diff_ts, end = c(2015,12))
test1 <- window(diff_ts, start = c(2016,1),end= c(2016,12))
library(forecast)

mod1 <- auto.arima(train1, max.p = 3, max.q = 3, max.P = 2, max.Q = 2)

library(ggplot2)
# Prévoir la série temporelle de test à l'aide
du modèle ARIMA sélectionné
forecast_arimax <- forecast(mod1, h = 10)
# Tracer les séries temporelles de train et de test, ainsi que
la prévision pour la période de test
ggplot() +
  geom_line(aes(x = index(train1), y = coredata(train1),
    color = "Train")) +
  geom_line(aes(x = index(test1), y = coredata(test1),
    color = "Test")) +
  geom_line(aes(x = index(forecast_arimax$mean),
    y = forecast_arimax$mean, color = "Forecast")) +
  scale_color_manual(values = c("Train" = "blue",
    "Test" = "red", "Forecast" = "darkgreen")) +
  labs(x = "Time", y = "touristes", color = "Data") +
  ggtitle("serie temporelle avec prévision
pour la période de test")

library("tseries")

```

```

resid <- residuals(mod1)

portmanteau_test <- Box.test(resid, lag=10, type="Ljung-Box")

print(portmanteau_test)

# Prédire les valeurs pour 1 mois, 6 mois et 24 mois à l'aide
# du modèle ARIMA ajusté
pred_1month <- forecast(mod1, h = 1)
pred_6months <- forecast(mod1, h = 6)
pred_12months <- forecast(mod1, h = 12)

# Calculer les mesures d'erreur pour les prévisions
mape_1month <- mean(abs((test[1] - pred_1month$mean)/test[1])) * 100
mape_6months <- mean(abs((test[1:6] - pred_6months$mean)
/test[1:6])) * 100
mape_12months <- mean(abs((test[1:12] - pred_12months$mean)
/test[1:12])) * 100

MAE1 <- mean(abs(pred_1month$mean - test[1]))
MAE2 <- mean(abs(pred_6months$mean - test[1:6]))
MAE3 <- mean(abs(pred_12months$mean - test[1:12]))

RMSE1 <- sqrt(mean((pred_1month$mean - test[1])^2))
RMSE2 <- sqrt(mean((pred_6months$mean - test[1:6])^2))
RMSE3 <- sqrt(mean((pred_12months$mean - test[1:12])^2))

# Afficher les mesures d'erreur dans un tableau
accuracy_table <- data.frame(
  MAE=c(MAE1,MAE2,MAE3),
  MAPE = c(mape_1month, mape_6months, mape_12months),
  RMSE=c(RMSE1,RMSE2,RMSE3)
)
rownames(accuracy_table) <- c("h = 1", "h = 6", "h = 12")
accuracy_table

```

```

# Appliquer la différenciation cumulée pour
inverser la différenciation
diff_inv <- cumsum(diff(diff_ts, lag = 12,
differences = 1)) + diff_ts[12]
# Appliquer le filtre inverse pour réintroduire la saisonnalité
saison <- stats::filter(diff_inv, filter = c(rep(1, 11), 0, 0),
sides = 1) + mean(diff_ts)

train <- window(pfa2_2, end = c(2015,12))
test <- window(pfa2_2, start = c(2016,1))
mod1 <- auto.arima(train1, max.p = 3, max.q = 3, max.P = 2,
max.Q = 2)

library(ggplot2)

# Prévoir la série temporelle de test à l'aide du modèle
ARIMA sélectionné
forecast_arimax <- forecast(mod1, h = length(test)+12)
# Créer la dataframe des prévisions
df <- data.frame(date = seq(as.Date("2016-01-01"),
by = "month", length.out = 24),

predicted_values = forecast_arimax$mean)
df

library(forecast)
# Appliquer le lissage exponentiel simple avec un
coefficient de lissage de 0,2
model <- HoltWinters(train1, alpha=0.8)
# Prédire les valeurs futures
predictions <- forecast(model, h=length(test1)+12)
# Afficher les données d'origine et les prévisions
plot(test, main="Lissage exponentiel simple",
xlab="Temps", ylab="Valeurs")
lines(test, col="blue")
lines(predictions$mean, col="red")
legend("topright", legend=c("Données de test",
"Prévisions"), col=c( "blue", "red"), lty=1:1, cex=0.8)

```



```

library("tseries")
resid <- residuals(model)
portmanteau_test <- Box.test(resid, lag=10,
type="Ljung-Box")
print(portmanteau_test)

mae <- mean(abs(predictions$mean - test1))
mape <- mean(abs((predictions$mean - test1) / test1))
* 100
rmse <- sqrt(mean((predictions$mean - test1)^2))
# Afficher les résultats dans un tableau
table <- data.frame(MAE = mae, MAPE = mape,
RMSE = rmse)
print(table)

predictions <- forecast(model, h=24)
# Créer un data frame avec les prévisions et la date
df <- data.frame(date = seq(as.Date("2016-01-01"),
by = "month", length.out = 24),
predicted_values = predictions$mean)

library(urroot)
hegy.test(pfa2_2 ,deterministic=c(1,1,1))

library(fUnitRoots)
adfTest(pfa2_2, lags = 12, type = "ct")

ts<-diff(pfa2_2, differences=1)
adfTest(ts, lags = 12, type = "ct")
ts1<-diff(ts, differences=1)
adfTest(ts1, lags = 12, type = "ct")

library(urca)
result_dhf <- ur.df(pfa2_2, type = "drift",
lags = 12, selectlags = "AIC")
# Afficher les résultats
summary(result_dhf)

library(forecast)

```

```

train <- window(ts1, end = c(2015,5))
test <- window(ts1, start = c(2015,6))
mod <- auto.arima(train, max.p = 3, max.q = 3,
max.P = 2, max.Q = 2, seasonal = TRUE)
summary(mod)

library(ggplot2)
# Prévoir la série temporelle de test à l'aide du modèle
sARIMA sélectionné
forecast_arimax <- forecast(mod, h = length(test))
# Tracer les séries temporelles de train et de test,
ainsi que la prévision pour la période de test
ggplot() +
  geom_line(aes(x = index(train), y = coredata(train),
color = "Train")) +
  geom_line(aes(x = index(test), y = coredata(test),
color = "Test")) +
  geom_line(aes(x = index(forecast_arimax$mean),
y = forecast_arimax$mean, color = "Forecast")) +
  scale_color_manual(values = c("Train" = "blue"
, "Test" = "red", "Forecast" = "green")) +
  labs(x = "Time", y = "SNCF", color = "Data") +
  ggtitle("serie avec prévision pou la periode de test")

library("tseries")
resid <- residuals(mod)
portmanteau_test <- Box.test(resid, lag=10,
type="Ljung-Box")
print(portmanteau_test)

# Définir la longueur de la période future à prévoir
future_length <- 12

# Prévoir la période future à l'aide du modèle sARIMA
sélectionné
forecast_future <- forecast(mod, h = future_length+
length(test))

# Tracer les séries temporelles de train et de test,

```

```

ainsi que la prévision pour la période de test et de
future
ggplot() +
  geom_line(aes(x = index(train), y = coredata(train),
    color = "Train")) +
  geom_line(aes(x = index(test), y = coredata(test),
    color = "Test")) +
  geom_line(aes(x = index(forecast_arimax$mean), y =
    forecast_arimax$mean, color = "Forecast (Test)")) +
  geom_line(aes(x = index(forecast_future$mean), y =
    forecast_future$mean, color = "Forecast (Future)")) +
  scale_color_manual(values = c("Train" = "blue", "Test"
    = "red", "Forecast (Test)" = "green", "Forecast (Future)" =
    "orange")) +
  labs(x = "Time", y = "serie", color = "Data") +
  ggtitle("Série avec prévision pour la période de test
    et de future")

library(forecast)

# Diviser la série en données d'entraînement et de test
train <- window(pfa2_2, end = c(2015,12))
test <- window(pfa2_2, start = c(2016,1))

# Former un modèle ARIMA sur les données d'entraînement
model <- arima(pfa2_2, order=c(3,0,0), seasonal=c(1,1,0))

# Faire des prévisions sur les données de test en

utilisant le modèle
formé
forecast_raw <- predict(model, n.ahead = 24)

# Créer un tableau de données à partir des prévisions
forecast_table <- data.frame(mois = as.Date(time

(forecast_raw$pred)),
tourists = forecast_raw$pred)

```

```

# Afficher le tableau de données
forecast_table

# Créer un graphique pour les prévisions
plot(forecast_raw$pred, col = "red")

library(forecast)

# Charger les données
train <- window(pfa2_2, end = c(2015,12))
test <- window(pfa2_2, start = c(2016,1))

# Appliquer le lissage exponentiel saisonnier avec un
coefficient de lissage de 0,2
model1 <- HoltWinters(train, seasonal="multiplicative", alpha=0.2,
beta=0.1, gamma=0.3)

# Prédire les valeurs futures
predictions <- forecast(model1, h=24)

# Afficher les données d'origine et les prévisions
plot(test, main="Lissage exponentiel saisonnier",
xlab="Temps", ylab="Valeurs")
lines(test, col="blue")
lines(predictions$mean, col="red")
legend("topright", legend=c( "Données de test", "Prévisions"),
col=c( "blue", "red"), lty=1:1, cex=0.8)

# Calculer l'erreur de prévision

mae <- mean(abs(predictions$mean - test))
mape <- mean(abs((predictions$mean - test) / test)) * 100
rmse <- sqrt(mean((predictions$mean - test)^2))

# Afficher les résultats dans un tableau
table <- data.frame(MAE = mae, MAPE = mape, RMSE = rmse)
print(table)

```

```

library("tseries")

resid <- residuals(model1)

portmanteau_test <- Box.test(resid, lag=10, type="Ljung-Box")

print(portmanteau_test)

combined_pred <- (predictions$mean + forecast_arimax$mean) / 2
combined_forecast <- ts(combined_pred, start = c(2016, 1),
frequency = 12)
plot(test1)
lines(combined_forecast, col = "red")

mae <- mean(abs(combined_forecast - test))
mape <- mean(abs((combined_forecast - test) / test)) * 100
rmse <- sqrt(mean((combined_forecast - test)^2))

# Afficher les résultats dans un tableau
table <- data.frame(MAE = mae, MAPE = mape, RMSE = rmse)
print(table)
resid <- residuals(combined_forecast)
portmanteau_test <- Box.test(resid, lag = 10, type = "Ljung-Box")
print(portmanteau_test)

resid <- residuals(combined_forecast)
portmanteau_test <- Box.test(resid, lag = 10, type = "Ljung-Box")
print(portmanteau_test)

```

2 Logiciel Python

```

import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_excel("pfa2.xlsx")

```

```

    # Combiner les colonnes année et mois en une
    seule colonne de temps
df['month'] = pd.to_datetime(df['year'].astype(str) + '-' +
df['month'].astype(str), format='%Y-%m')

# Définir la colonne que vous souhaitez utiliser comme index temporel
df = df.set_index('month')

# Convertir le DataFrame en série temporelle
ts = df['tourists']

print(df.describe())

import pandas as pd
import matplotlib.pyplot as plt

Q1 = df["tourists"].quantile(0.25)    #312719.0
Q3 = df["tourists"].quantile(0.75)    #623084.25
IQR = Q3 - Q1    #310365.25

outliers = df[(df["tourists"] < (Q1 - 1.5 * IQR)) |
(df["tourists"] > (Q3 + 1.5 * IQR))]
print(outliers)                        #-152828.875

plt.boxplot(df["tourists"])
plt.show()

import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.holtwinters import SimpleExpSmoothing

# Appliquer le lissage exponentiel simple avec
un coefficient de lissage de 0,2
model = SimpleExpSmoothing(train_data).fit(smoothing_level=0.2)

# Prédire les valeurs futures
predictions = model.forecast(12)

```

```

# Afficher les données d'origine et les prévisions
plt.plot(train_data, label='Données d\'entraînement')
plt.plot(test_data, label='Données de test')
plt.plot(predictions, label='Prévisions')
plt.legend()
plt.show()

    train_data1, test_data1 = train_test_split(ts,
        test_size=0.2, shuffle=False)
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import ExponentialSmoothing

# Extraire la partie saisonnière

# Modéliser la partie saisonnière avec une méthode
de lissage exponentiel saisonnier
model3 = ExponentialSmoothing(ts, trend='add',
seasonal='add', seasonal_periods=12).fit()
print(model3.params)
# Afficher le type de modèle
print(model3.model)
# Prédire la partie saisonnière pour les 12 prochains mois
forecast = model3.predict(start=len(train_data1),
end=len(train_data1)+len(test_data1)-1)

# Visualiser la partie saisonnière et la prédiction
pour les 12 prochains mois
plt.plot(ts, label='notre base')
plt.plot(forecast, label='Prédiction')
plt.legend()
plt.show()
print(forecast)

    for param in pdq:
        for seasonal_param in pdq_x_QDQs:
            try:
                mod = sm.tsa.statespace.SARIMAX(tourist,
                    order=param,

```

```

        seasonal_order=seasonal_param,
        enforce_stationarity=True,
        enforce_invertibility=False)
    results = mod.fit()
    print('ARIMA{}x{} - AIC:{}'.format(param,
        seasonal_param, results.aic))
except:
    continue

train_data1, test_data1 = train_test_split(ts,
test_size=0.2,

shuffle=False)
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import ExponentialSmoothing

# Extraire la partie saisonnière

# Modéliser la partie saisonnière avec une méthode
de lissage exponentiel saisonnier
model3 = ExponentialSmoothing(ts, trend='add',
seasonal='add', seasonal_periods=12).fit()
print(model3.params)
# Afficher le type de modèle
print(model3.model)
# Prédire la partie saisonnière pour les 12 prochains mois
forecast = model3.predict(start=len(train_data1),
end=len(train_data1)+len(test_data1)-1)

# Visualiser la partie saisonnière et la prédiction
pour les 12 prochains mois
plt.plot(ts, label='notre base')
plt.plot(forecast, label='Prédiction')
plt.legend()
plt.show()
print(forecast)

# Calculer les résidus

```



```
residuals = model3.resid

# Effectuer le test de portmanteau
ljung_box = sm.stats.acorr_ljungbox(residuals,
lags=[10])
print(ljung_box)
```