



PROJET 9

*Prédiction
de la
Demande
en Electricité*

Introduction

Ce projet a été réalisé :

- Dans le cadre de la formation DATA ANALYST d'OpenClassRooms
- Avec l'aide/support de M. César Clavé (Mentor Openclassrooms)
- Contexte → Société spécialisée dans les énergies renouvelables (**ENERCOOP**)
- Mission → Mettre en adéquation l'offre et la demande en électricité des utilisateurs.
- Proposition → Conception de Modèles Holt-Winters & SARIMA

Sources Données

- Consommation Electrique → RTE France (réseau de Transport de l'électricité)
- DJU (correct effet meteo) → Cegibat



Sommaire

- ❑ Introduction (remerciements)
- ❑ Data Import - Analyse descriptive
 - Consommation Electrique ARA (2014-2019)
 - Données DJU (Cegibat)
 - Synthèse chiffrée, choix périmètre
- ❑ Correction de l'effet de T°
 - Régression Linéaire avec data DJU sur data conso
 - Visualisation superposée Avant/Après
- ❑ Désaisonnalisation Consommation - MA
 - Décomposition Data en : Tendence, Saisonnalité, Résidus
 - Correction avec Moyennes Mobiles
 - Visualisation superposée Avant/Après
- ❑ Prévision Consommation – Méthode Holt Winters
 - Présentation Méthode
 - Application – Analyse à posteriori
 - Evaluation/Performance du Modèle de Prédiction (HT)
- ❑ Prévision Consommation – Méthode SARIMA
 - Présentation Méthode
 - Application – Analyse à posteriori
 - Evaluation/Performance du Modèle de Prédiction (SARIMA)
- ❑ Conclusion / Questions
- ❑ Annexes

Data Import - Analyse descriptive



1^{er} Contrôle visuel (rapide)
(re)Mise en forme de
données



Nettoyage données (en Python)
«Cleaning» détaillé dans un
notebook jupyter



DataSet Principal → ara(0) (consommation électrique)

Filtre

5 années ½

Région Auvergne Rhône-Alpes

2014-01 → 2019-05 (65 Mois)

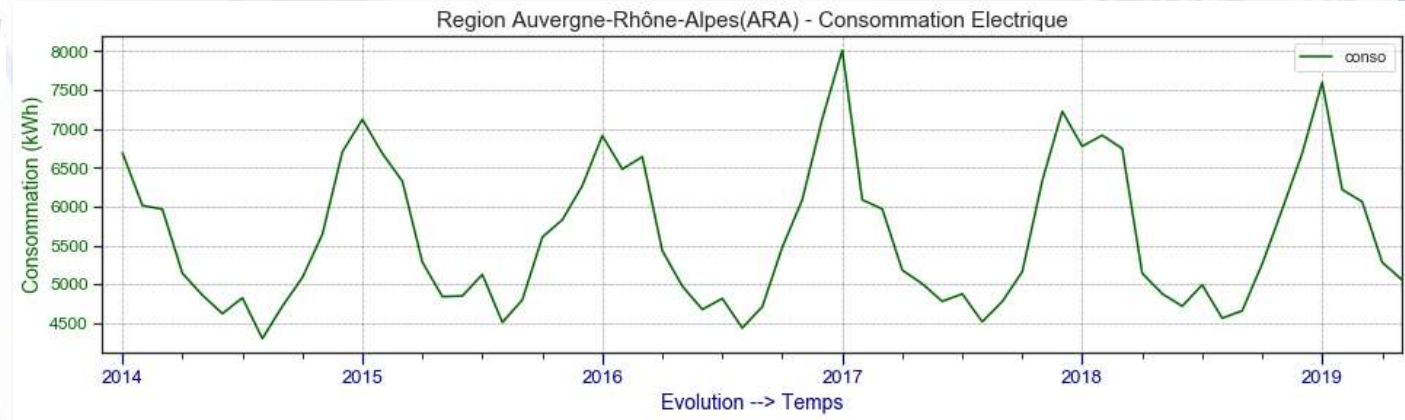


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65 entries, 0 to 64
Data columns (total 4 columns):
 ara_date    65 non-null datetime64[ns]
 conso       65 non-null int64
 annee       65 non-null object
 mois        65 non-null object
dtypes: datetime64[ns](1), int64(1), object(2)
memory usage: 2.1+ KB
(65, 4)
```

	ara_date	conso	annee	mois
0	2014-01-01	6693	2014	01
1	2014-02-01	6013	2014	02
...
63	2019-04-01	5281	2019	04
64	2019-05-01	5054	2019	05

65 rows × 4 columns

	A	B	C	D	E
1	Mois	Qualité	Territoire	Production totale	Consommation totale
79	2013-03	Données définitives	Auvergne-Rhône-Alpes	11129	0
92	2013-04	Données définitives	Auvergne-Rhône-Alpes	10532	0
105	2013-05	Données définitives	Auvergne-Rhône-Alpes	11456	0
118	2013-06	Données définitives	Auvergne-Rhône-Alpes	9750	0
131	2013-07	Données définitives	Auvergne-Rhône-Alpes	9153	0
144	2013-08	Données définitives	Auvergne-Rhône-Alpes	8779	0
157	2013-09	Données définitives	Auvergne-Rhône-Alpes	8236	0
170	2013-10	Données définitives	Auvergne-Rhône-Alpes	8876	0
183	2013-11	Données définitives	Auvergne-Rhône-Alpes	10312	0
196	2013-12	Données définitives	Auvergne-Rhône-Alpes	12264	0
209	2014-01	Données définitives	Auvergne-Rhône-Alpes	13001	6693
222	2014-02	Données définitives	Auvergne-Rhône-Alpes	11323	6013
235	2014-03	Données définitives	Auvergne-Rhône-Alpes	10828	5965
248	2014-04	Données définitives	Auvergne-Rhône-Alpes	9143	5139
261	2014-05	Données définitives	Auvergne-Rhône-Alpes	9448	4858
274	2014-06	Données définitives	Auvergne-Rhône-Alpes	8705	4621
287	2014-07	Données définitives	Auvergne-Rhône-Alpes	9473	4823





Calcul des DJU



Le degré jour est une valeur représentative de l'écart entre la T° moyenne d'une journée et un seuil de température préétabli (18 °C dans le cas des DJU ou Degré Jour Unifié).

Sommés sur une période, ils permettent de calculer les besoins de chauffage et de climatisation d'un bâtiment.

DJU base 18° d'un Jour J

$$= 18 - [(T^{\circ}\text{max} - T^{\circ}\text{min}) / 2]$$

CONSTATATION :

DJU plus importants en hiver qu'en été

Données d'entrée

Station météo	LYON-BRON (69)
Usage	Chauffage
Méthode de calcul	Météo
Température de référence	18°C
Date de début	31/12/2011
Date de fin	30/05/2019

Résultats

	Jan	Fév	Mar	Avr	Mai	Jun	Jui	Aoû	Sep	Oct	Nov	Déc	Total
2019	445,9	308,7	237,6	170,4	106,6	0	0	0	0	0	0	0	1 269,2
2018	301,4	430,4	291,8	97,8	47,2	0,3	0	1,5	13,3	122,8	264,9	355,7	1 926,9
2017	532,2	271,1	205,4	180,8	80,7	3,7	1,3	2	49,1	110	328,5	409,8	2 174,3
2016	344	319,4	318,4	176,8	93,7	8,3	4,4	0,3	14,5	191,2	283,1	466,4	2 220,3
2015	419,5	393,2	276,1	143,4	49,1	0	0	0	50,2	188,6	249,5	289,7	2 059,1
2014	334,1	290,1	244,4	134,1	82,9	3	8,8	3,7	20,2	73,7	222,8	393,9	1 811,5
2013	453,1	452,7	327,6	190,4	150,9	30,8	0	1,2	27,1	88,1	337,5	412,8	2 472
2012	409	517,9	215,6	191,2	68,4	9,6	5,6	2,6	40,5	122	266,8	380,7	2 229,7



DataSet complémentaire → dju

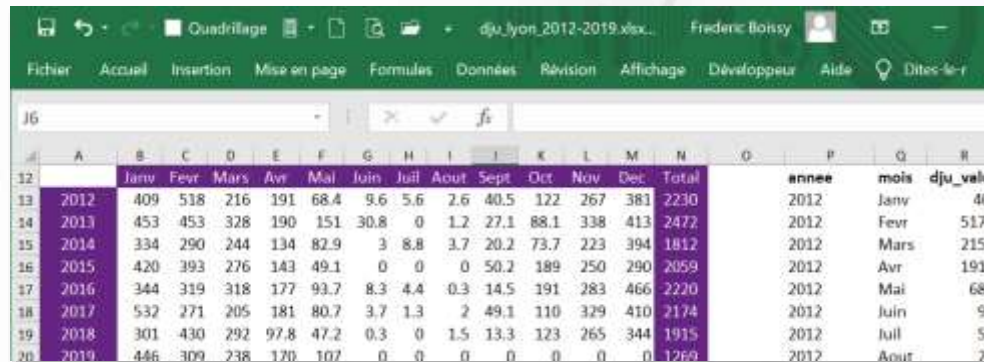
Filtre
5 années ½

Station LYON
2014-01 → 2019-05 (65 Mois)

Station de relevé Dju choisie

→ **Ville de LYON**

(Essais avec Evian, Clermont, Chambéry)

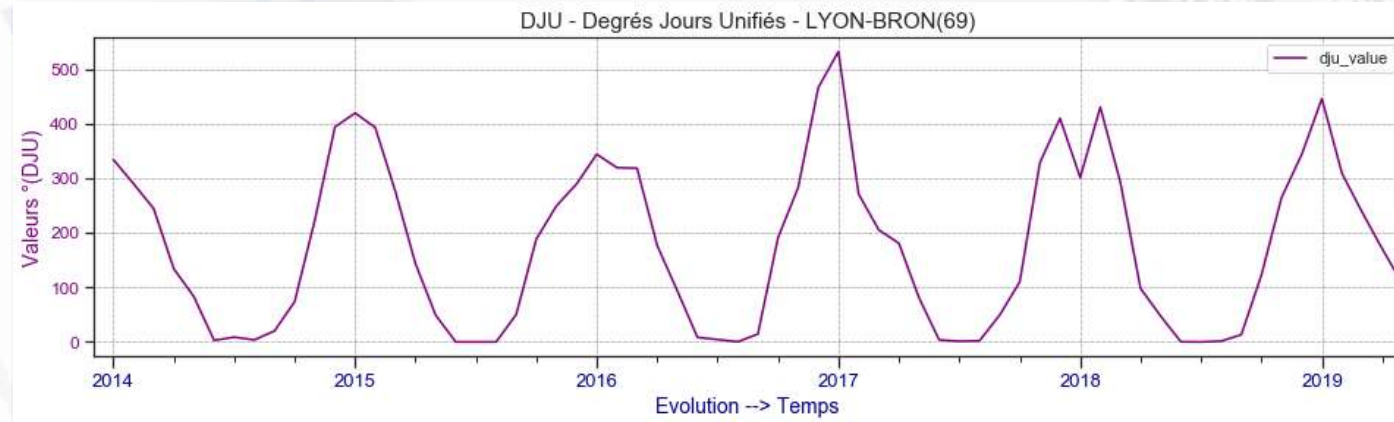


	Janv	Fevr	Mars	Avr	Mai	Juin	Juil	Aout	Sept	Oct	Nov	Dec	Total
2012	409	518	216	191	68.4	9.6	5.6	2.6	40.5	122	267	381	2230
2013	453	453	328	190	151	30.8	0	1.2	27.1	88.1	338	413	2472
2014	334	290	244	134	82.9	3	8.8	3.7	20.2	73.7	223	394	1812
2015	420	393	276	143	49.1	0	0	0	50.2	189	250	290	2059
2016	344	319	318	177	93.7	8.3	4.4	0.3	14.5	191	283	466	2220
2017	532	271	205	181	80.7	3.7	1.3	2	49.1	110	329	410	2174
2018	301	430	292	97.8	47.2	0.3	0	1.5	13.3	123	265	344	1915
2019	446	309	238	120	107	0	0	0	0	0	0	0	1269

	Index	annee	mois	dju_mois	dju_value
0	24	2014	Janv	2014-01	474.700
1	25	2014	Fevr	2014-02	410.700
...
63	87	2019	Avr	2019-04	293.700
64	88	2019	Mai	2019-05	260.700

65 rows × 5 columns

	A	B	C	D
	dju_date	dju_value	annee	mois
2	2012-01	500.1	2012	Janv
3	2012-02	623.3	2012	Fevr
4	2012-03	331.3	2012	Mars
5	2012-04	294.6	2012	Avr
6	2012-05	142.9	2012	Mai
7	2012-06	51.4	2012	Juin
8	2012-07	27.7	2012	Juil
9	2012-08	20.4	2012	Aout
10	2012-09	123	2012	Sept
11	2012-10	243.1	2012	Oct
12	2012-11	375.8	2012	Nov
13	2012-12	499.8	2012	Dec
14	2013-01	550.2	2013	Janv
15	2013-02	543.9	2013	Fevr
16	2013-03	475.3	2013	Mars
17	2013-04	282	2013	Avr
18	2013-05	247.2	2013	Mai



Correction de l'effet de la température (T°)

... sur la consommation (du au chauffage électrique par exemple)
Correction des données de consommation à l'aide
→ d'une régression linéaire

$$CONSO = \alpha + \beta \cdot DJU_{(Value)} + \epsilon_{(Residus)}$$

Outils (librairies) → statsmodels

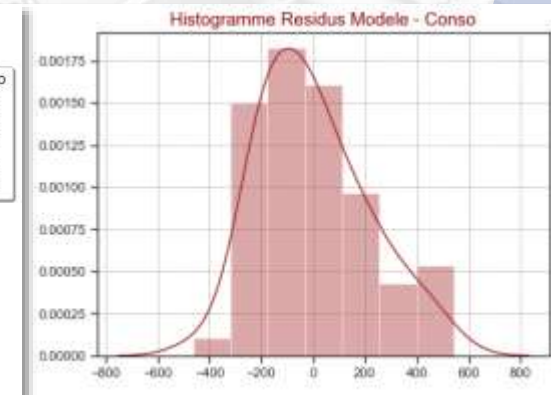
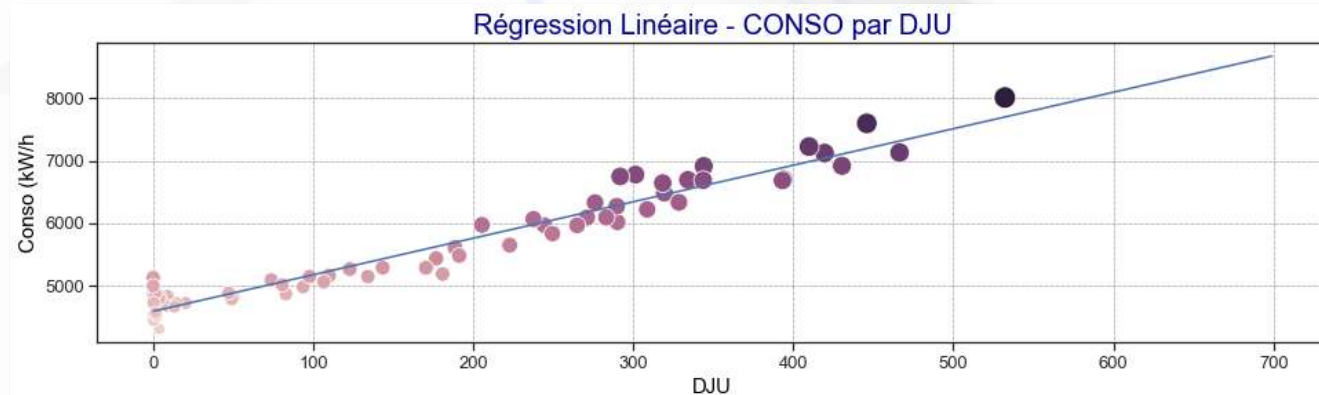
OLS Regression Results						
=====						
Dep. Variable:	conso		R-squared:	0.947		
Model:	OLS		Adj. R-squared:	0.946		
Method:	Least Squares		F-statistic:	1123.		
Date:	Wed, 11 Sep 2019		Prob (F-statistic):	7.12e-42		
Time:	15:15:35		Log-Likelihood:	-439.62		
No. Observations:	65		AIC:	883.2		
Df Residuals:	63		BIC:	887.6		
Df Model:	1					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]
dju_value	5.8603	0.175	33.518	0.000	5.511	6.210
intercept	4582.7200	40.557	112.994	0.000	4501.673	4663.767

Vérification → scikit-learn

Var. Expliquée : R² = 94.69 (%)
Coefficients : α = [5.86033664] β = 4582.7200207488595

Le coefficient R² explique la corrélation entre les deux séries :

→ Corrélation démontrée R² = 0,946



Correction de l'effet de la température (T°)

ARA2 - Nouvelle Série Corrigée de l'effet de Température (T°)

ARA0
Date

2014-01-01	6693
2014-02-01	6013
2014-03-01	5965
2014-04-01	5139
2014-05-01	4858
...	...
2019-01-01	7599
2019-02-01	6218
2019-03-01	6063
2019-04-01	5281
2019-05-01	5054

65 rows x 1 columns

ARA2
Date

2014-01-01	4,735.062
2014-02-01	4,312.916
2014-03-01	4,532.734
2014-04-01	4,353.129
2014-05-01	4,372.178
...	...
2019-01-01	4,985.876
2019-02-01	4,408.914
2019-03-01	4,670.584
2019-04-01	4,282.399
2019-05-01	4,429.288

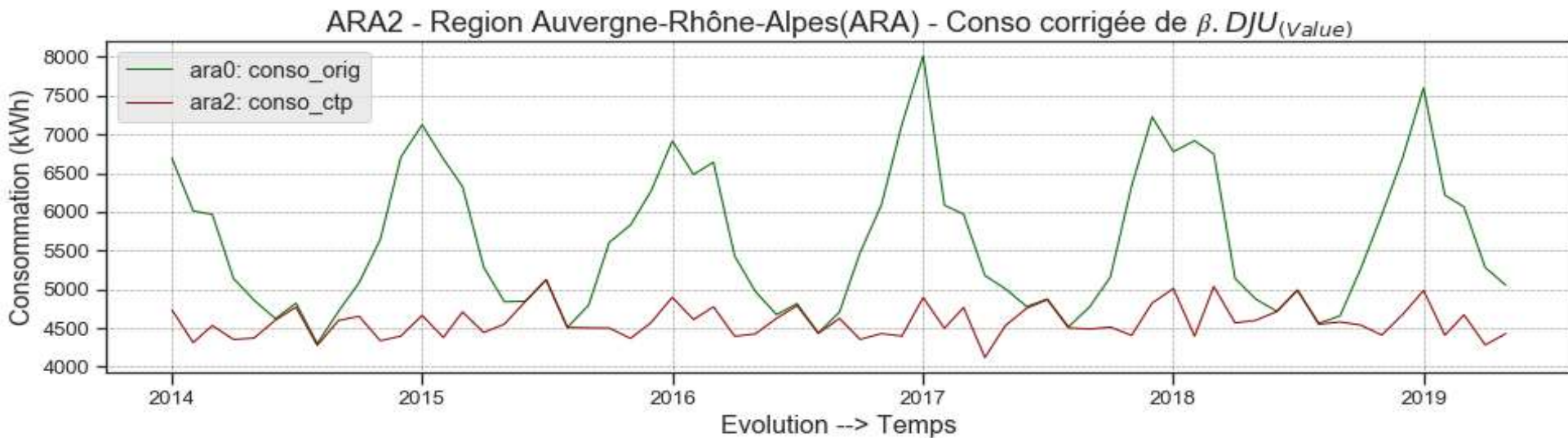
65 rows x 1 columns

```
ara_ctp = ara0['conso'] -  $\alpha$ *np.asarray(dju['dju_value'])
ara_ctp = ara_ctp.rename("conso_ctp")
ara2 = pd.DataFrame(ara_ctp)
```

En retirant la composante $\beta.DJU(Value)$ pour ne conserver que le modèle de régression linéaire, on obtient la courbe rouge ci-dessous :

Ara2 Conso_ctp

Consommation électrique corrigée de l'effet de Température T°



Après correction de l'effet de T° , on utilisera la méthode :

→ **Moyennes Mobiles (moving average)**

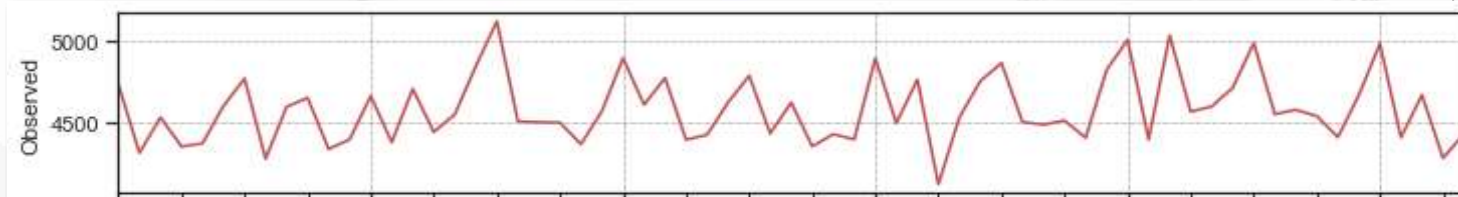
Avant de passer à cette opération, il est intéressant d'observer les différentes **composantes** de la Série **ARA2**

Utilisation de la fonction → **seasonal_decompose** de la librairie « statsmodel »

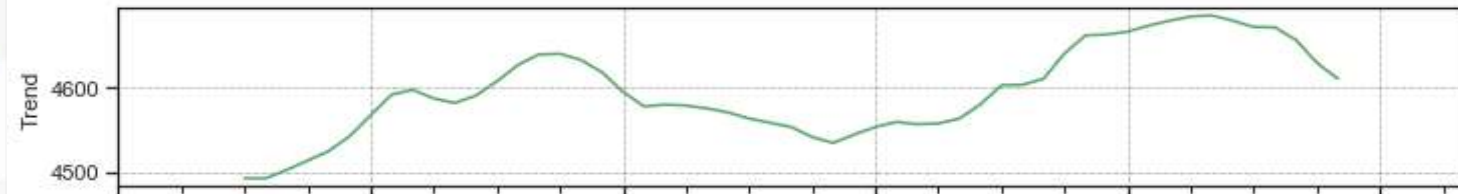
```
from statsmodels.tsa.seasonal import seasonal_decompose
decomp_ara2_add = seasonal_decompose(ara2, model='additive', freq=12)
```

Annexes → Modèle Additif / Multiplicatif

Données brutes



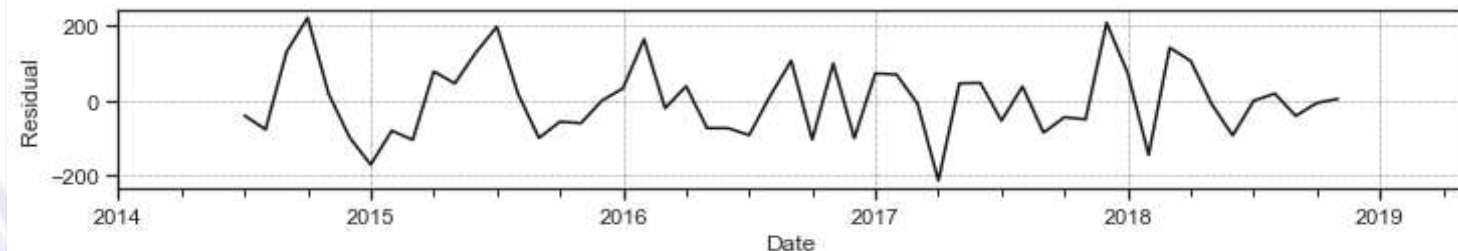
Tendance
(estimation croissante)



Saisonnalité
(mesures espacées
De 12mois)

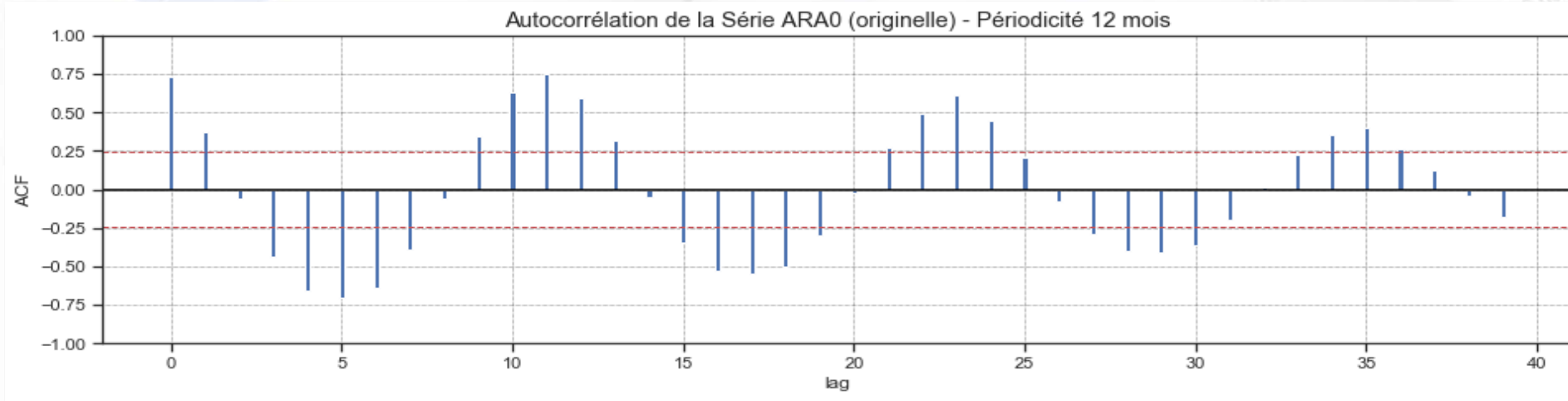


Résidus
Bruit blanc, après retrait de la
saisonnalité et de la Tendance.
Partie des variables qui ne sont pas
expliquées par les 2 premières.



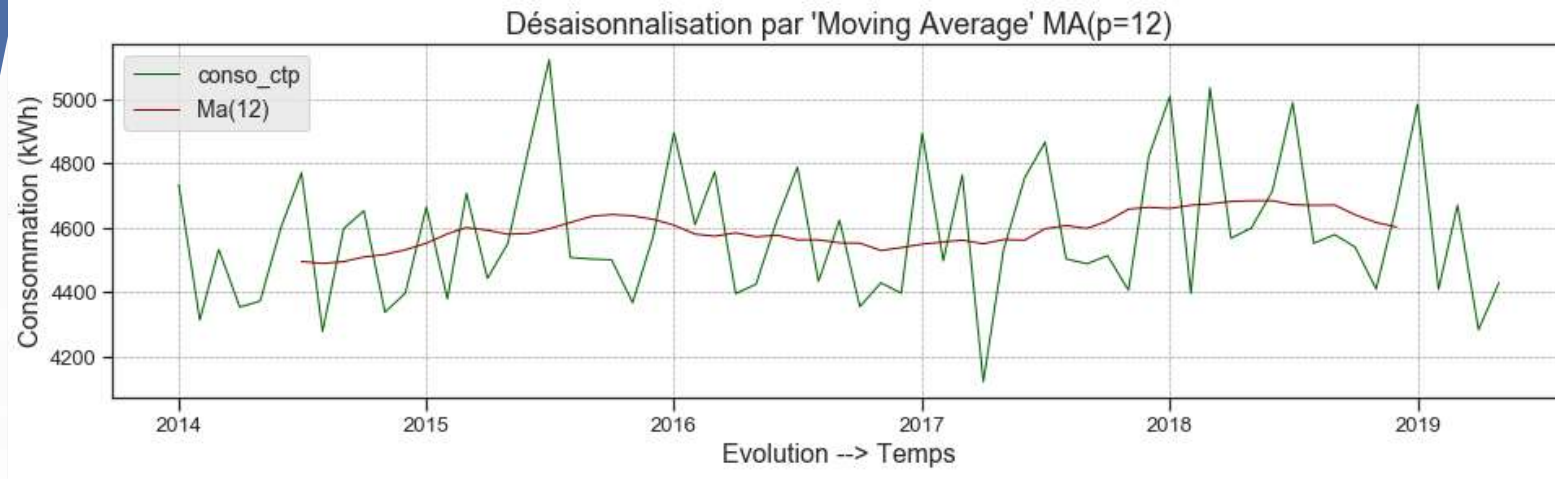
Détermination **Ordre(p)** pour le modèle **MA(p)**

Graphe **Autocorrélation** pour retrouver la saisonnalité.
La corrélation maximale positive obtenue est sur 12 mois.
Avec le Lag exprimé en unité « mensuelle ».

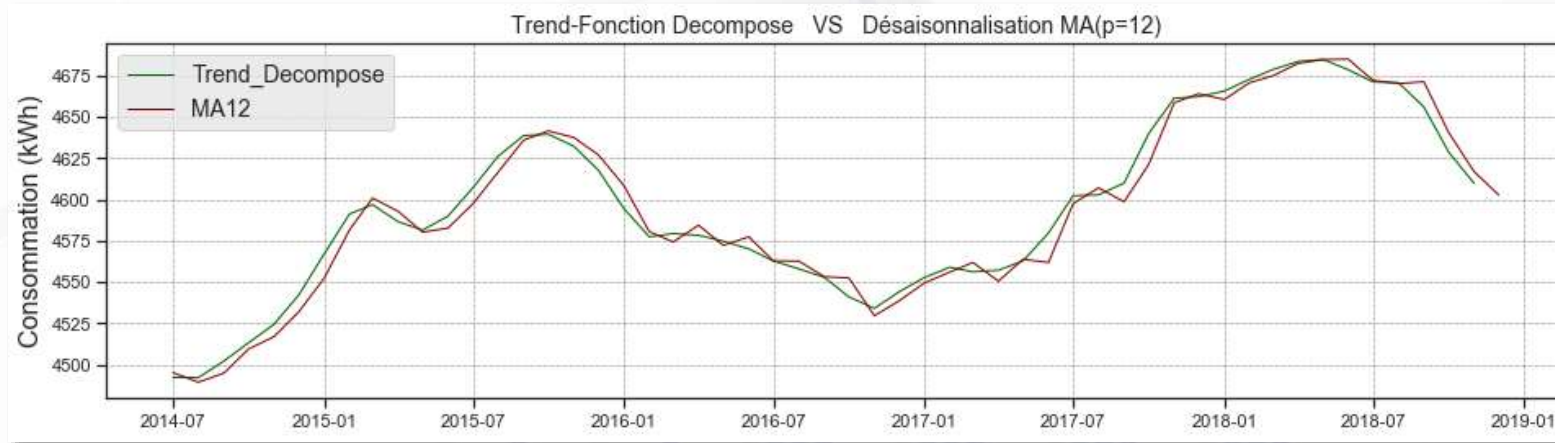


Utilisation de cette corrélation maxi de 12 mois avec **Pandas.Rolling**
→ Moyenne glissante sur cette période (12mois)

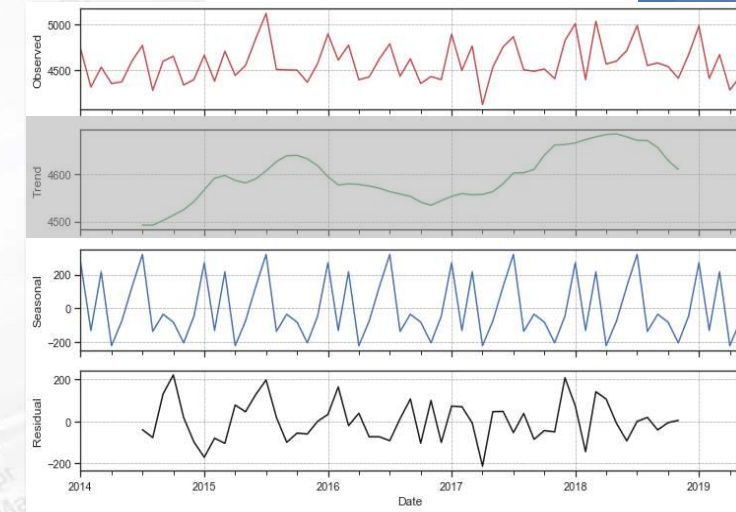
```
# Tail-rolling average transform  
rolling = ara2.rolling(window=12, center=True)  
ma12 = rolling.mean()
```



Donc on estime ici que la désaisonnalisation a été efficace car on retrouve la décomposition de la tendance vue avec la fonction précédente (seasonal_decompose).



seasonal_decompose



Préambule → Notion de **Stationnarité**

Un processus X_t sera considéré (*faiblement*) **Stationnaire** si

- son Espérance(moyenne arith.) μ est constante dans le temps :
 $\mathbb{E}(X_t) = \mu, \forall t \in \mathbb{Z}$
- ses Autocovariances sont constantes dans le temps :
 $\forall (t, h) \in \mathbb{Z}, \text{Cov}(X_t, X_{t-h})$ ne dépend que de l'intervalle séparant les 2 instants h , pas de l'instant t

Pour vérifier qu'une Série est **Stationnaire** :

→ Test de **Dickey-Fuller**

Hypothèse Nulle H_0 : la série n'est pas stationnaire au seuil de Risque de 1ère espèce $\alpha=5\%$

→ Etude des **autocorrélations** (simples et partielles)

Test **adfuller** (statsmodel)

```
from statsmodels.tsa.stattools import adfuller
print("> Is the ARA2 - data stationary ?")
dfctest = adfuller(ara2.conso_ctp.diff().dropna(), autolag='AIC')
print("Test statistic = {:.3f}".format(dfctest[0]))
print("P-value = {:.3f}".format(dfctest[1]))
print("Critical values :")
for k, v in dfctest[4].items():
    print("\t{}: {} - The Time Series AR2 data is {} stationary with {}% confidence".format(k, v, "not" if v < dfctest[0] else "", 100-int(k[:1])))
```

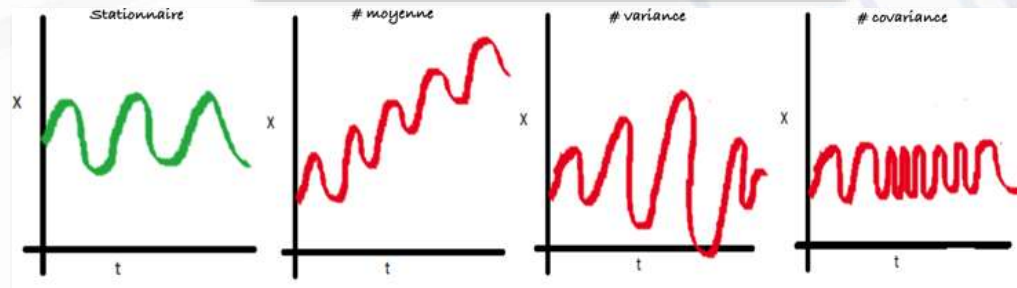
Contrôle stationnarité **ARA0 – série origine**

La série **est stationnaire** au seuil de risque $\alpha=5\%$

```
> Is the ARA0 data stationary ?
Test statistic = -7.725
P-value = 0.000
Critical values :
1%: -3.560242358792829 - The Times Series ARA0 data is stationary with 99% confidence
5%: -2.9178502070837 - The Times Series ARA0 data is stationary with 95% confidence
10%: -2.5967964150943397 - The Times Series ARA0 data is stationary with 90% confidence
```

$P_{value} = 0.0$

Exemple de Séries **non-stationnaire**



Contrôle stationnarité **ARA2 – série corrigée (T*)**

La série **n'est pas** stationnaire au seuil de risque $\alpha=5\%$

```
> Is the ARA2 - data stationary ?
Test statistic = -2.415
P-value = 0.138
Critical values :
1%: -3.562878534649522 - The Time Series AR2 data is not stationary with 99% confidence
5%: -2.918973284023669 - The Time Series AR2 data is not stationary with 95% confidence
10%: -2.597393446745562 - The Time Series AR2 data is not stationary with 90% confidence
```

$P_{value} = 0.138$

→ La Méthode *Holt-Winters*

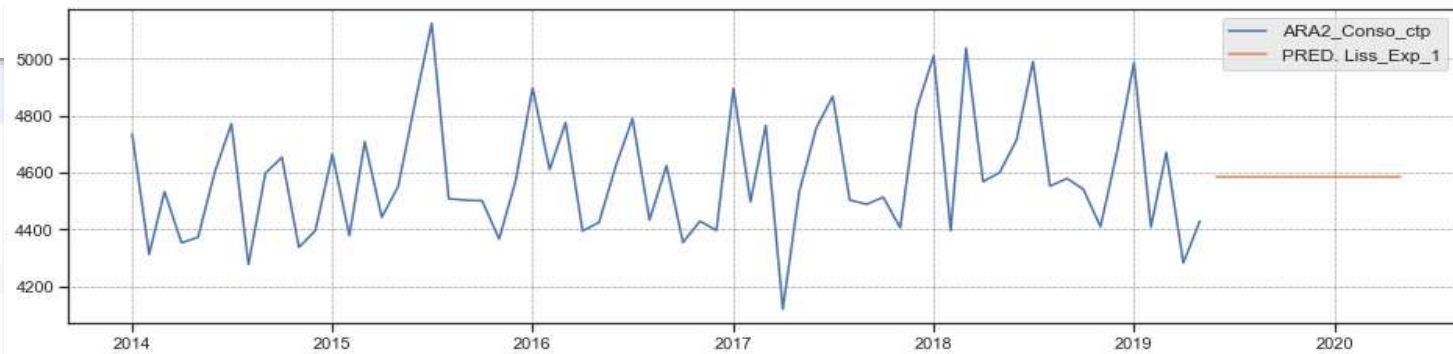
Effectue des « lissages » exponentiels sur séries TP. Elle utilise 3 paramètres :

- α : avec lequel on va pondérer les valeurs de la série (lissage de la moyenne)
- β : avec lequel on va pondérer les valeurs de tendance de la série
- γ : avec lequel on va pondérer les valeurs de la saisonnalité

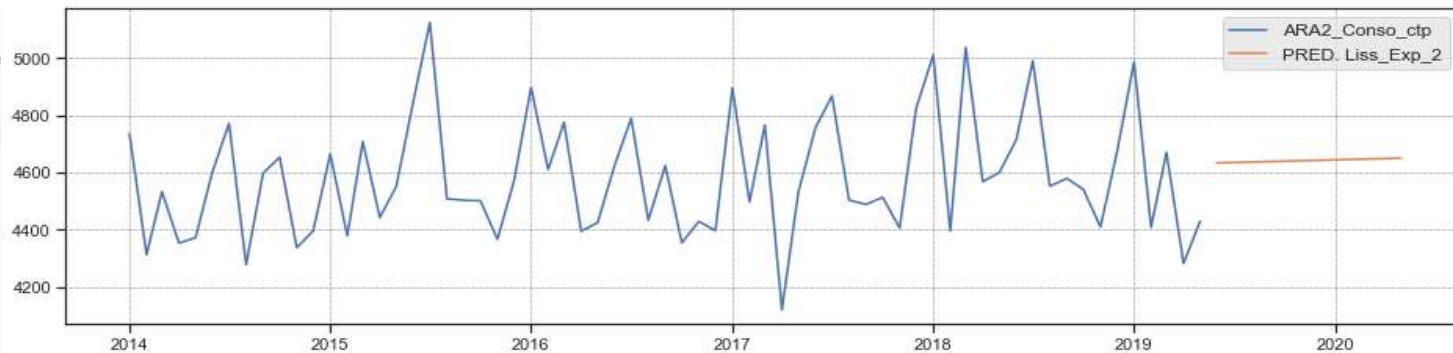
Détermination optimale de α , β et γ avec le package « *exponentials smoothing* » (lib. statsmodels)

```
from statsmodels.tsa.api import ExponentialSmoothing
les = ExponentialSmoothing(np.asarray(ara2['conso_ctp']), trend=None, seasonal=None).fit()
les_pred = les.forecast(12)
les = ExponentialSmoothing(np.asarray(ara2['conso_ctp']), trend='add', seasonal=None).fit()
les_pred = les.forecast(12)
```

Lissage Exponentiel *Simple*

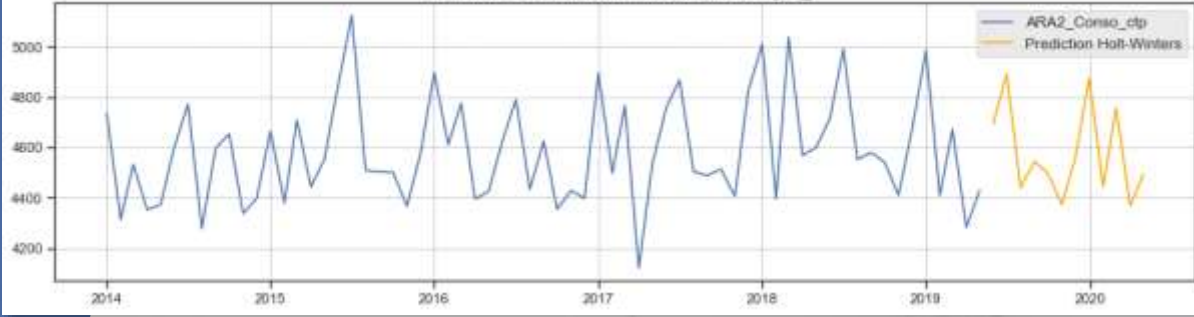


Lissage Exponentiel *Double*

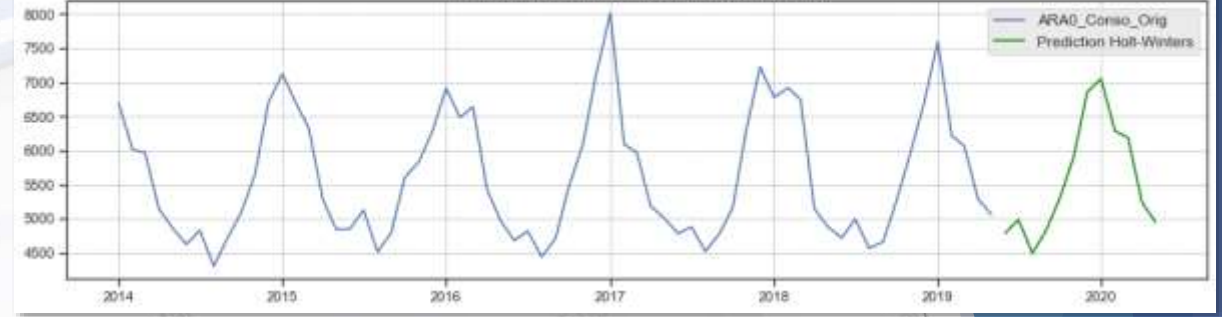


Méthode Holt-Winters Triple Lissage Exponentiel

Prédiction Holt-Winters sur Série ARA2 corrigée_T*



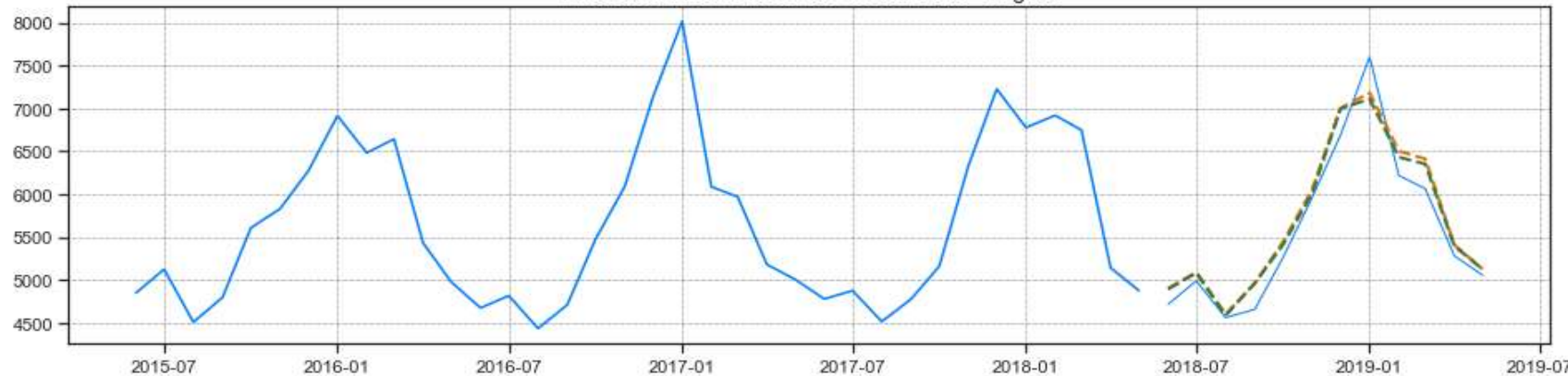
Prédiction Holt-Winters sur Série ARA0 - origine



Analyse a Posteriori Contrôle Prédiction VS 12 derniers Mois Observés (Réels)

— Vraie Valeur OBS - ARA0
 - - w/o damping -- RMSE=1230.18, MAPE(%)=17.65, AIC=627.58, BIC=659.10
 - - damped -- RMSE=1215.35, MAPE(%)=17.37, AIC=632.88, BIC=666.38

Prédiction Holt-Winters sur Série ARA0 - origine



w/o Damping

RMSE₁ = 1230,18
MAPE₁ = 17,65%
AIC₁ = 627,58
BIC₁ = 659,10

Damped

RMSE₂ = 1230,18
MAPE₂ = 17,65%
AIC₂ = 627,58
BIC₂ = 659,10

Estimateurs

AIC (critère d'Akaike)
 BIC (critère de Schwartz)

$$AIC(p, q) = \ln(\hat{\sigma}^2) + 2 \frac{p+q}{T}$$

$$BIC(p, q) = \ln(\hat{\sigma}^2) + (p+q) \frac{\ln(T)}{T}$$

$$RMSE = \sqrt{\frac{1}{T} \sum_{t=1}^T (x_t - \hat{x}_t)^2}$$

$$MAPE = \frac{1}{T} \sum_{t=1}^T \left| \frac{x_t - \hat{x}_t}{x_t} \right|$$



Evaluation de notre modèle de Prédiction :

En retirant de la Série Originelle ARA0 la série générée par Holt-Winters, on obtient les **résidus**, qui eux, « normalement » devront être décorrélés des notions de saisonnalité et de tendance. Ce sont des stochastiques.

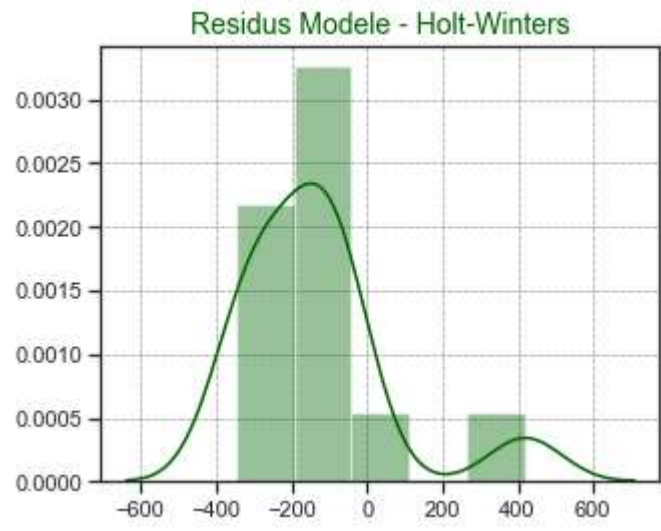
Test Shapiro-Wilk (normalité des résidus)

Problème

Ici, l'hypothèse de normalité est remise en cause
→ **p-value faible (0,01) < 5%**

```
from scipy.stats import shapiro
hw_resid = test['conso'] - pred
print("TEST Shapiro P-value = ", shapiro(hw_resid)[1])
```

TEST Shapiro P-value = 0.010640964843332767



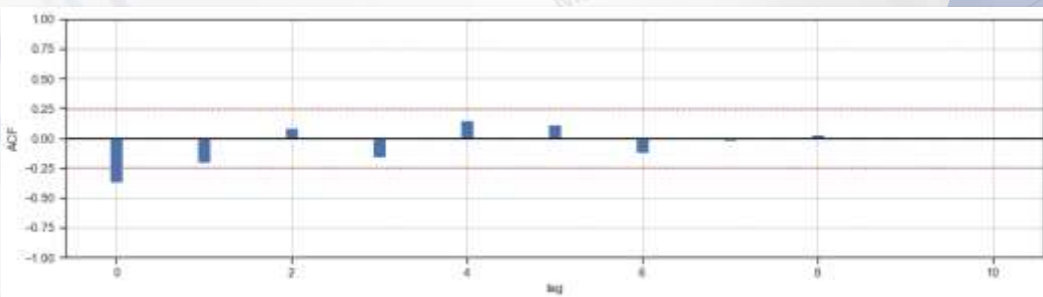
Test Box-Ljung (Blancheur des Résidus - Reste saisonnier ?)

Problème

Rejet de l'hypothèse nulle H_0 d'indépendance des résidus.
→ Les **p_value < 5%**

```
from ljungbox import *
h, pV, Q, cV = lbqtest(hw_resid, range(1, 10), alpha=0.1)
print('lag p-value Q c-value rejectH0')
for i in range(len(h)):
    print("%-2d %10.3f %10.3f %10.3f %s" % (i+1, pV[i], Q[i], cV[i], str(h[i])))
```

lag	p-value	Q	c-value	rejectH0
1	0.000	14.848	2.706	True
2	0.000	31.084	4.605	True
3	0.000	45.005	6.251	True
4	0.000	56.988	7.779	True
5	0.000	57.125	9.236	True
6	0.000	57.125	10.645	True
7	0.000	57.125	12.017	True
8	0.000	57.125	13.362	True
9	0.000	57.125	14.684	True



Prévision Consommation - Méthode SARIMA

→ Méthode **SARIMA** (*Seasonal Auto Regressive Integrated Moving Average*)

GENERALITES

SARIMA est utilisé pour les séries non stationnaires

Ce modèle permet d'identifier les tendances et la saisonnalité.

Le SARIMA se compose d'autres modèles de prévision :

- **AR** Modèle Autorégressif
- **MA** Modèle des moyennes mobiles
- **ARIMA** Modèle pour séries non stationnaires
(i = integrated, d = différenciation)
- **SARIMA** Modèle pour séries non stationnaires

PROCEDURE

- 1) Stationnarisation (éventuellement)
→ DickeyFuller, ACF, PACF
- 2) Identification de modèles potentiels
- 3) Estimation / Vérification des modèles
- 4) Choix définitif du modèle
- 5) Prévision / Analyse a posteriori
- 6) Etudes des Résidus

Modèle SARIMA → Configuration

Nécessite la sélection d'hyperparamètres pour la tendance et les éléments saisonniers de la série.

Composantes Tendanciellles (non saisonnières)
3 éléments de tendance doivent être configurés.
Les mêmes que pour le modèle ARIMA :

- (**p**) l'ordre AR (autorégressif)
- (**d**) le degré de différenciation
- (**q**) l'ordre MA (moving average - moyenne mobile)

Composantes Saisonnières

4 éléments « saisonniers » qui ne font pas partie de l'ARIMA doivent être configurés :

- (**P**) l'ordre AR saisonnier
- (**D**) le degré de différenciation saisonnier
- (**Q**) l'ordre MA saisonnier
- (**m**) Le nombre de pas « timestep » pour 1 seule période saisonnière (12 ici)

SARIMA(p, d, q)(P, D, Q)m

Python

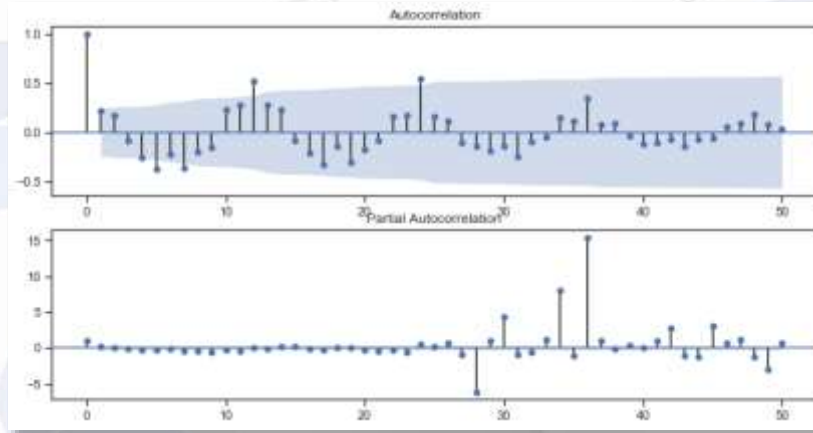
Fonction SARIMAX (librairie Statsmodels)

statsmodels.tsa.statespace.sarimax.SARIMAX



○ Contrôle Stationnarité

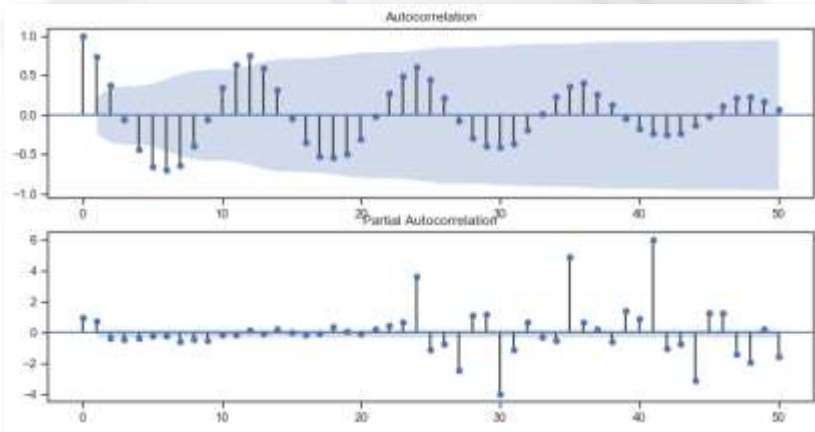
La sortie ACF de la série ARA0 présente une décroissance lente vers 0, ce qui traduit un problème de non-stationnarité.



1

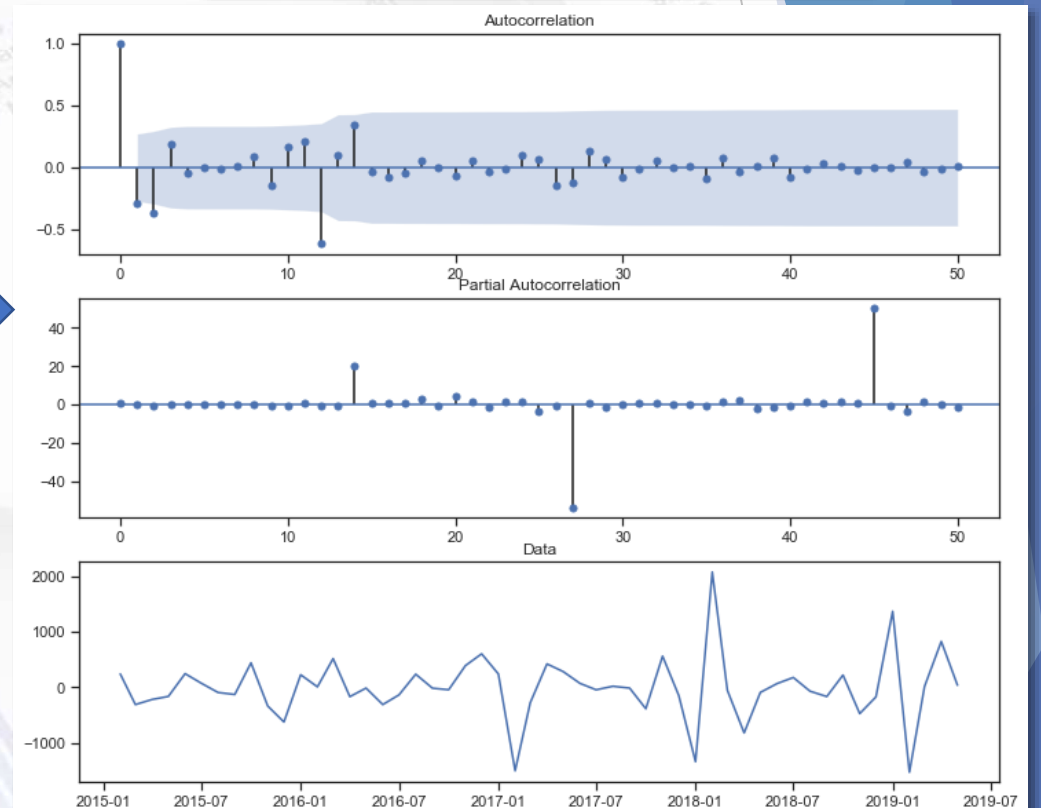
```
ara0_dif1 = ara0['conso'] - ara0['conso'].shift(1)
```

On effectue donc une 1^{ère} Différenciation ($I-B$) en tendance.



On effectue donc une Différenciation en saisonnalité ($I-B^{12}$).

```
ara0_dif_1_12 = ara0_dif1 - ara0_dif1.shift(12)
```



Prévision Consommation - Méthode SARIMA

○ Détermination Termes manquants (p, q, P, D, Q)m $d = 1$

- Création d'un algorithme de recherche des meilleures combinaisons possibles pour les termes AR & MA.
- Limitation → Valeurs [0 ou 1]

```
# Creation d'un DataFrame vide pour récupérer les résultats
# des meilleurs estimations de paramètres de modèles SARIMA
# selon la valeur du critère d'Akaike : AIC_value
# Plus la valeur est faible, meilleur le modèle est.
dfsarima = pd.DataFrame()
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            sarima_p1 = pd.Series(str(param))
            sarima_p2 = pd.Series(str(param_seasonal))
            mod = sm.tsa.statespace.SARIMAX(np.asarray(ara0['conso']),
                                           order=param,seasonal_order=param_seasonal,
                                           enforce_invertibility=False)

            results = mod.fit()
            aic_value = pd.Series(results.aic)
            bic_value = pd.Series(results.bic)
            ligne = pd.concat([sarima_p1, sarima_p2, aic_value, bic_value], axis=1)
            ligne.columns = ['SARIMA_p1', 'SARIMA_p2', 'AIC_Value', 'BIC_Value']
            dfsarima = dfsarima.append(ligne, ignore_index = True)
        except :
            continue

dfsarima = dfsarima.sort_values(by="AIC_Value").reset_index(drop=True)
pd.options.display.max_rows=10
dfsarima.head(10)
```

$p=0, q=1$
 $P=1, D=1, Q=1$
 $m=12$

	SARIMA_p1	SARIMA_p2	AIC_Value	BIC_Value
0	(0, 1, 1)	(1, 1, 1, 12)	761.614	769.419
1	(0, 1, 1)	(1, 1, 0, 12)	763.497	769.351
2	(1, 1, 1)	(1, 1, 0, 12)	763.586	771.391
3	(1, 1, 1)	(0, 1, 1, 12)	764.534	772.339
4	(0, 1, 1)	(0, 1, 1, 12)	765.221	771.075
5	(1, 1, 0)	(1, 1, 1, 12)	769.939	777.744
6	(0, 1, 0)	(1, 1, 1, 12)	771.138	776.992
7	(1, 1, 1)	(1, 1, 1, 12)	773.986	783.743

Modèle **SARIMA** choisi → (0,1,1)(1,1,1)12

Statespace Model Results

```
=====
Dep. Variable:          y          No. Observations:          65
Model:                SARIMAX(0, 1, 1)x(1, 1, 1, 12)      Log Likelihood          -376.807
Date:                  Thu, 12 Sep 2019                    AIC              761.614
Time:                  19:16:10                            BIC              769.419
Sample:                0                                    HQIC              764.686
                    - 65
Covariance Type:      opg

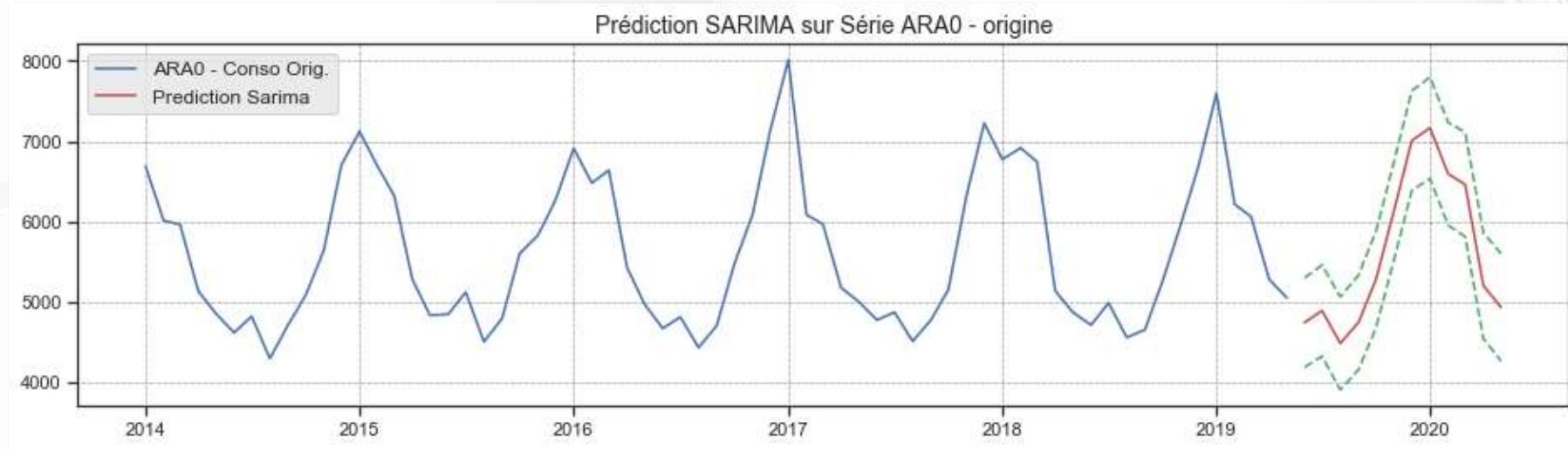
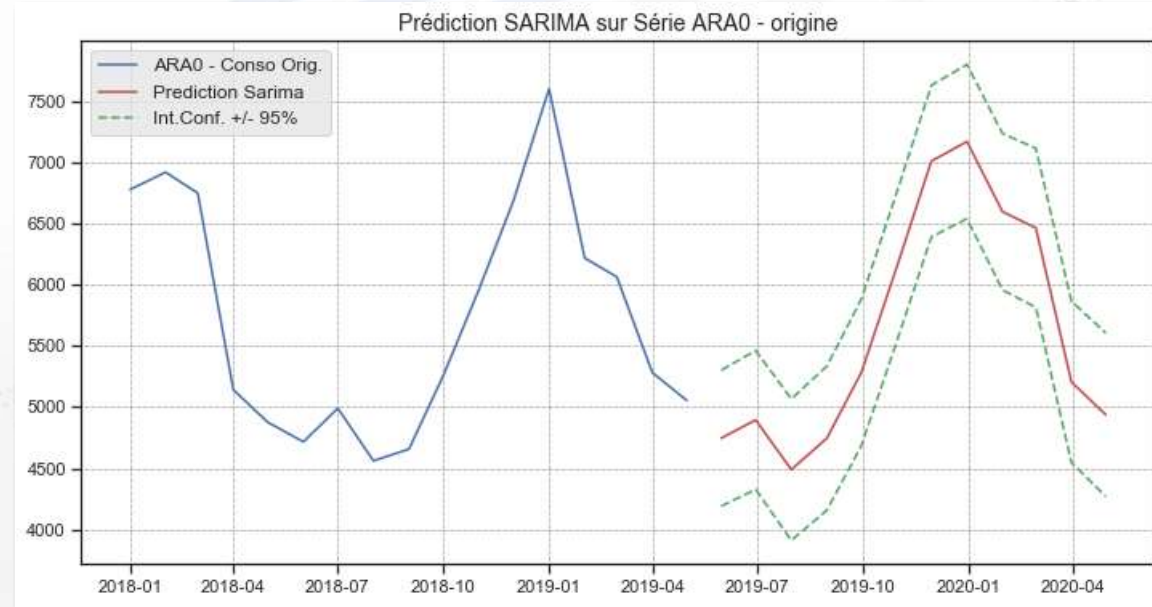
=====
              coef      std err      z      P>|z|      [0.025      0.975]
-----
ma.L1          -1.2520      0.119     -10.545      0.000     -1.485     -1.019
ar.S.L12        -0.4652      0.236      -1.972      0.049     -0.928     -0.003
ma.S.L12        -0.4315      0.343      -1.257      0.209     -1.104      0.241
sigma2          5.121e+04    9567.134      5.352      0.000     3.25e+04     7e+04

Ljung-Box (Q):                31.24    Jarque-Bera (JB):                3.34
Prob(Q):                      0.84    Prob(JB):                      0.19
Heteroskedasticity (H):        0.89    Skew:                          0.34
Prob(H) (two-sided):          0.81    Kurtosis:                      4.04
=====

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
Retard : p-value
6 : 0.954886085573103
12 : 0.940211841290871
18 : 0.8945604161685343
24 : 0.9122886298713837
30 : 0.9286217788127267
36 : 0.9403939608882159
```



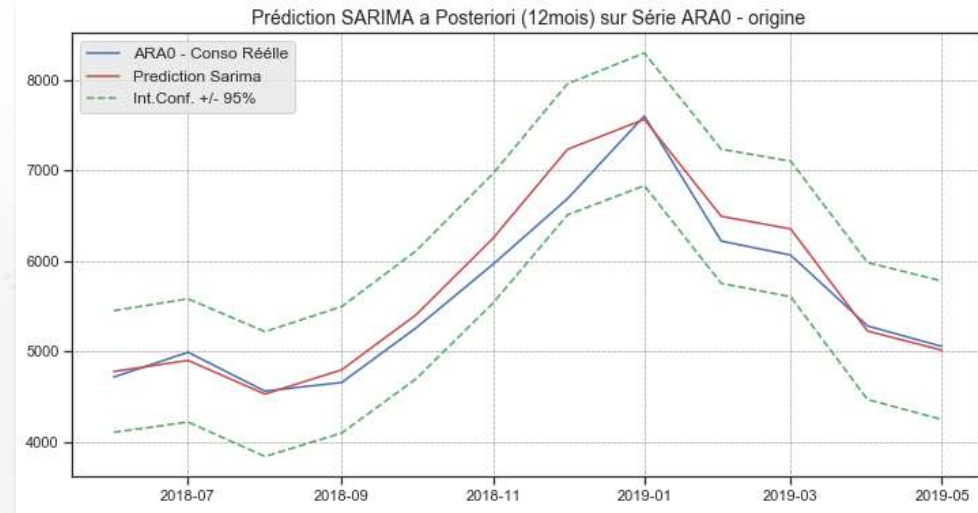
Méthode SARIMA
Prédiction avec
Modèle $(0,1,1)(1,1,1)_{12}$



SARIMA - Analyse a Posteriori Contrôle Prédiction VS 12 derniers Mois Observés (Réels)

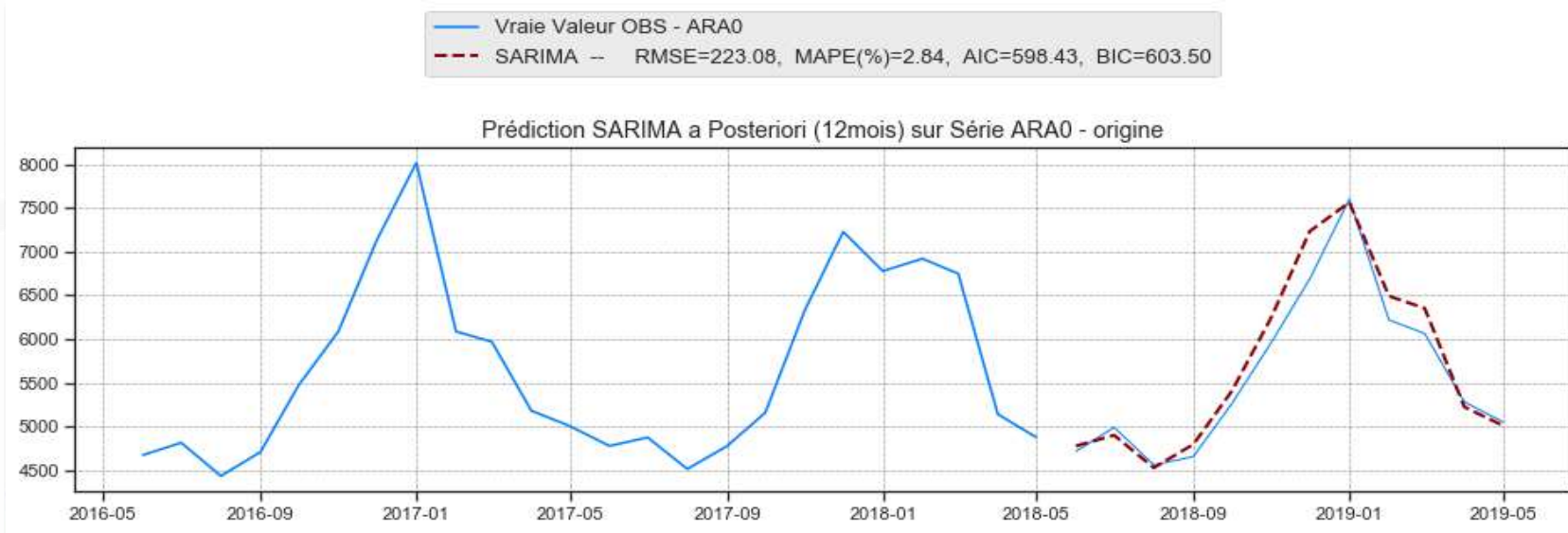
SARIMA Meilleur Modèle
→ Modèle (0,1,1)(1,1,1)12

SARIMA_p1	SARIMA_p2	AIC_Value	BIC_Value
(0, 1, 1)	(1, 1, 0, 12)	598.434	603.500
(1, 1, 1)	(1, 1, 0, 12)	598.931	605.686
(1, 1, 0)	(1, 1, 0, 12)	607.589	612.656



Résultats Holt-Winters

RMSE₂ = 1230,18
MAPE₂ = 17,65%
AIC₂ = 627,58
BIC₂ = 659,10



Résultats SARIMA

RMSE = 223,08
MAPE = 2,84%
AIC = 598,43
BIC = 603,50



Evaluation de notre modèle de Prédiction SARIMA → Test des RESIDUS

Test Shapiro-Wilk (normalité des résidus)

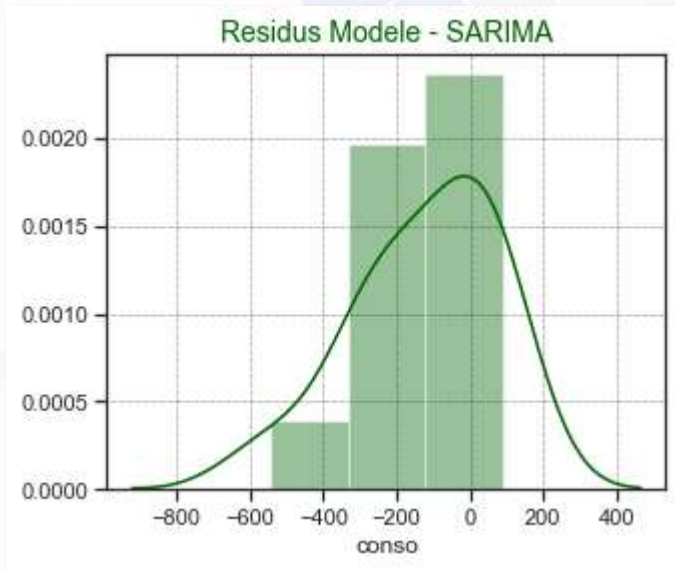
Hypothèse de normalité validé au seuil de 5%

Mais attention tout de même

→ $p\text{-value} = 0,1222 > 5\%$

```
sarima_resid = test['conso'] - sar_pred
print("TEST Shapiro P-value = ", shapiro(sarima_resid)[1])
```

TEST Shapiro P-value = 0.12229303270578384



Test Box-Ljung (Blancheur des Résidus - Reste saisonnier ?)

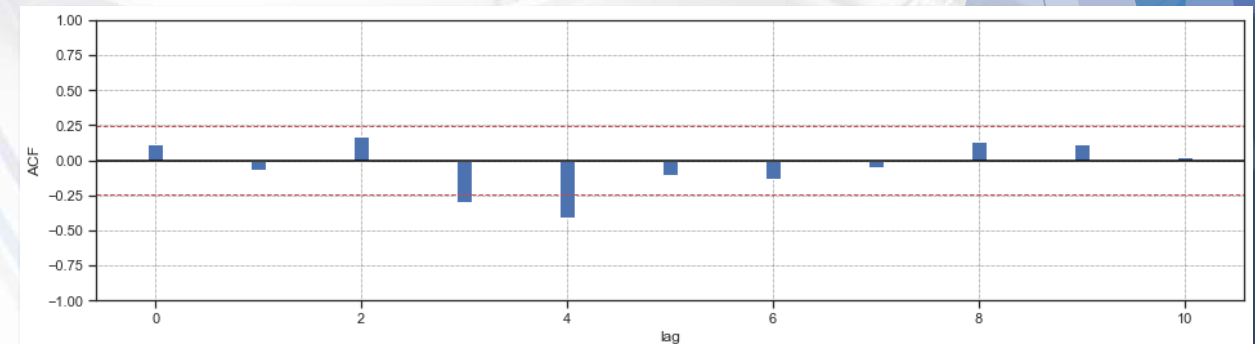
Problème

Rejet de l'hypothèse nulle H_0 d'indépendance des résidus.

→ Les $p\text{-value} < 5\%$

```
from ljungbox import *
h, pV, Q, cV = lbqtest(sarima_resid, range(1, 10), alpha=0.1)
print('lag p-value Q c-value rejectH0')
for i in range(len(h)):
    print("%-2d %10.3f %10.3f %10.3f %s" % (i+1, pV[i], Q[i], cV[i], str(h[i])))
```

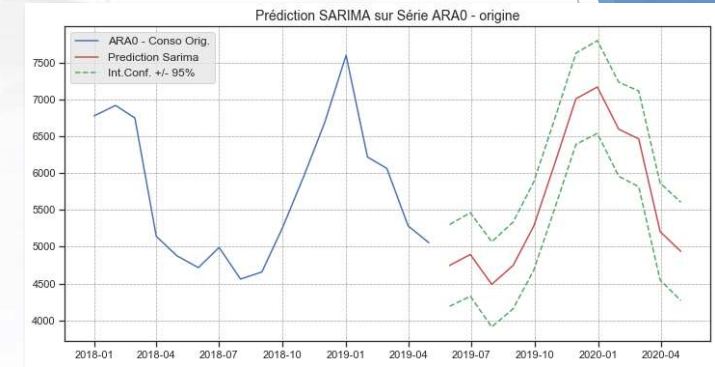
lag	p-value	Q	c-value	rejectH0
1	0.000	13.068	2.706	True
2	0.000	22.187	4.605	True
3	0.000	29.182	6.251	True
4	0.000	35.710	7.779	True
5	0.000	41.568	9.236	True
6	0.000	41.568	10.645	True
7	0.000	41.568	12.017	True
8	0.000	41.568	13.362	True
9	0.000	41.568	14.684	True



Projet 9

Prédictions demande Electricité

→ Le modèle **SARIMA** présenté semble le plus approprié pour établir notre prédiction de consommation électrique



Merci de votre attention !

Pour plus d'informations, vous pouvez
me contacter en suivant le lien
directement ci-dessous

frederic.boissy@gmail.com

Annexes – Modélisations « add » & « mult »

Explications sur 2 types de modélisation "déterministes" :

Ces modèles relèvent de la Statistique Descriptive.

Ils ne font intervenir que de manière sous-jacente le calcul des probabilités et consistent à supposer que l'observation de la série à la date t est une fonction du temps t et d'une variable ϵt centrée faisant office d'erreur au modèle, représentant la différence entre la réalité et le modèle proposé.

Soient :

X_t , la série temporelle,

Z_t , la composante tendancielle,

S_t , la composante saisonnière,

ϵt , le résidu, erreur ou écart du modèle

MODELE ADDITIF

$$X_t = Z_t + S_t + \epsilon t$$

t variant de $t = 1 \dots$ à $T, T+1, \dots$

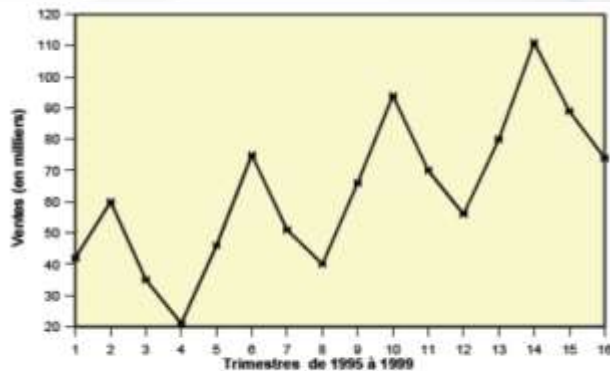


FIGURE 6 – Modèle additif. Amplitude constante autour de la tendance

MODELE MULTIPLICATIF

$$X_t = Z_t(1 + S_t)(1 + \epsilon t)$$

t variant de $t = 1 \dots$ à $T, T+1, \dots$

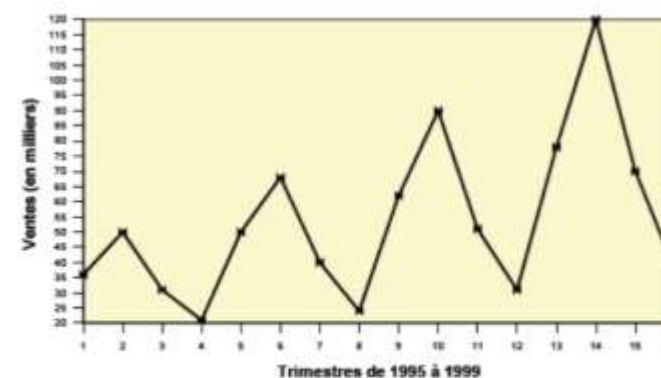


FIGURE 7 – Modèle multiplicatif. Amplitude proportionnelle à la tendance

