

Projet 9 : Prédisez la demande électricité

Année 2020-2021



Présentation des données

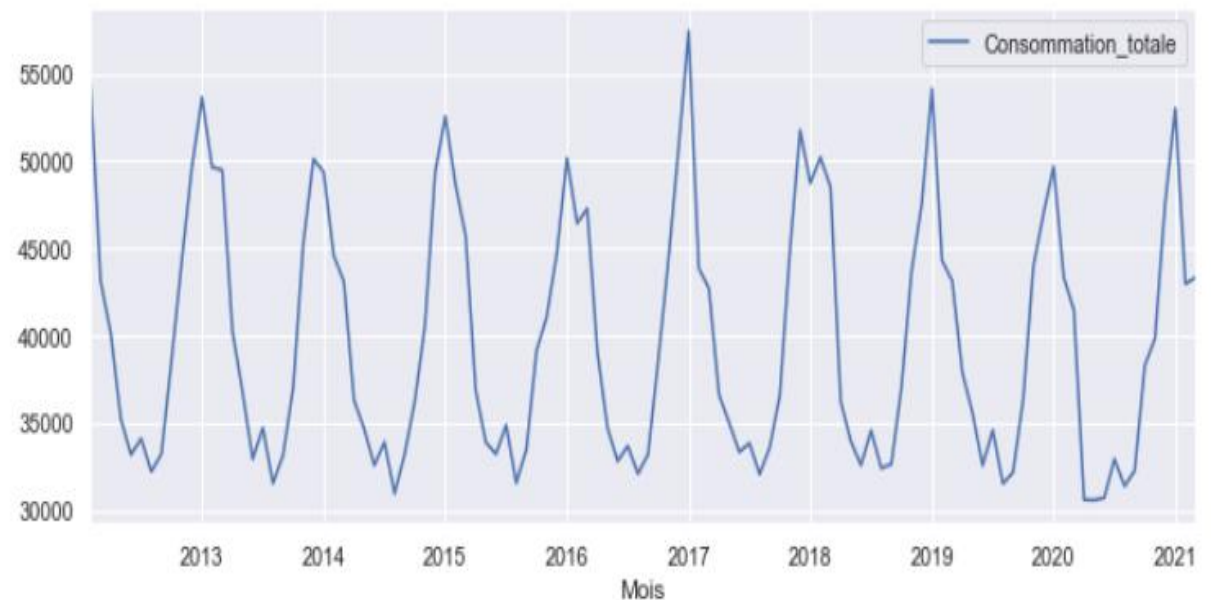
Notre consommation

```
df = pd.read_csv("Data/consommation.csv",  
                 encoding = "ISO-8859-1",  
                 engine='python',  
                 sep="\t")  
df=df[['Mois', 'Territoire', 'Consommation totale']]
```

Janvier 2009 à Juin 2021

2 colonnes et 110 lignes

	Mois	Consommation_totale
0	2012-02-01	54476
1	2012-03-01	43156
2	2012-04-01	40176
3	2012-05-01	35257
4	2012-06-01	33219



Jeu de météo

Les données météo qui sont utilisées pour
corriger l'effet température proviennent de :
CEGIBAT
île de France

2. Sélectionnez la méthode de calcul ([en savoir plus sur les méthodes de calculs](#))

☒ Météo ☐ Professionnels de l'énergie

3. Sélectionnez le type d'usage

☒ Chauffage ☐ Climatisation

4. Sélectionnez la température de référence



5. Période de chauffage

Date de début Date de fin

Les DJU sont proposés sur la période du 01/01/2009 au 15/06/2021.

2. Sélectionnez la méthode de calcul ([en savoir plus sur les méthodes de calculs](#))

☒ Météo ☐ Professionnels de l'énergie

3. Sélectionnez le type d'usage

☐ Chauffage ☒ Climatisation

4. Sélectionnez la température de référence



5. Période de chauffage

Date de début Date de fin

Les DJU sont proposés sur la période du 01/01/2009 au 15/06/2021.

Chauffage
Janvier 2009 à Juin 2021

Climatisation
Janvier 2009 à Juin 2021

Création d'un nouveau Jeu de données

But : avoir les données dans un même data frame

	Année	01	02	03	04	05	06	07	08	09	10	11	12	Total
0	2021	0	0	0.2	0.9	5.1	47.6	0.0	0.0	0.0	0.0	0.0	0	53.8
1	2020	0	0	0.0	11.5	28.6	57.5	96.6	157.0	72.3	0.0	0.0	0	423.4
2	2019	0	0	0.0	5.7	3.2	100.1	152.3	115.6	22.6	4.0	0.0	0	403.3
3	2018	0	0	0.0	16.1	42.9	74.6	198.7	126.3	33.2	17.0	0.0	0	508.7
4	2017	0	0	0.0	0.0	47.2	111.9	108.3	80.5	8.5	2.7	0.0	0	358.9
5	2016	0	0	0.0	0.0	8.5	37.0	105.6	121.4	59.1	0.0	0.0	0	331.5
6	2015	0	0	0.0	2.3	2.8	64.8	139.1	122.4	1.0	0.0	0.3	0	332.6
7	2014	0	0	0.0	0.0	2.2	41.3	101.1	35.7	45.8	7.1	0.0	0	233.0
8	2013	0	0	0.0	3.8	0.0	29.9	152.1	89.2	27.1	4.3	0.0	0	306.2
9	2012	0	0	0.0	0.0	30.1	25.6	60.4	117.9	20.4	1.4	0.0	0	255.6
10	2011	0	0	0.0	12.6	21.6	54.1	30.8	74.7	49.8	10.9	0.0	0	254.5
11	2010	0	0	0.0	3.3	16.1	64.3	134.7	59.2	8.0	4.3	0.0	0	289.7
12	2009	0	0	0.0	0.0	12.3	45.1	86.7	117.9	15.2	4.8	0.0	0	281.9

```
meteo_newformat={'mois':[],'climatisation':[]}
```

```

Project Name Here
for Année in meteo.index.values:
    for mois in meteo.columns:
        meteo_newformat['mois'].append(f"{Année}-{mois}-01")
        meteo_newformat['climatisation'].append(meteo.loc[Année,mois])

```

```

meteo_newformat=pd.DataFrame(meteo_newformat)
meteo_newformat['mois']=pd.to_datetime(meteo_newformat['mois'])

```

Climatisation

Degré Jour Unifié : est la différence entre la température extérieure et une température de référence qui permet de réaliser des estimations de consommations d'énergie

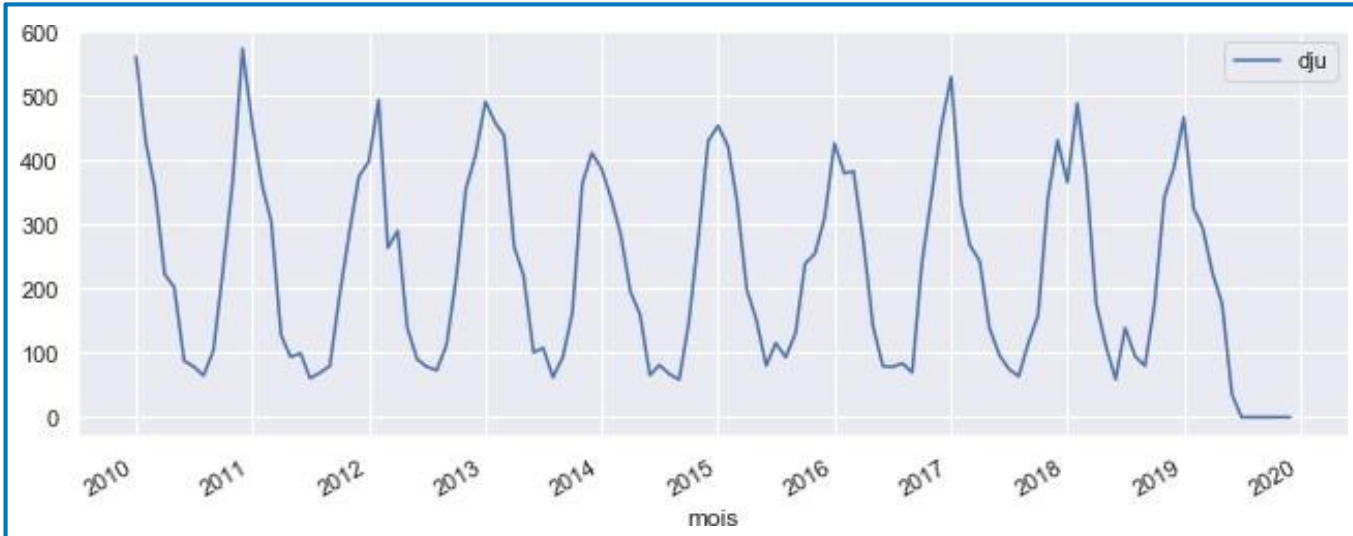
```
dju['dju']=dju['chauffage']+dju['climatisation']
dju.head()
```

	mois	climatisation	chauffage	dju
0	2019-01-01	0.0	466.9	466.9
1	2019-02-01	0.0	324.3	324.3
2	2019-03-01	0.0	295.1	295.1
3	2019-04-01	0.7	223.5	224.2
4	2019-05-01	0.0	176.8	176.8

111	2019-04-01	38078	2019-04-01	0.7	223.5	224.2
112	2019-05-01	35599	2019-05-01	0.0	176.8	176.8
113	2019-06-01	32571	2019-06-01	9.6	26.0	35.6
114	2019-07-01	34606	2019-07-01	0.0	0.0	0.0

115 rows × 6 columns

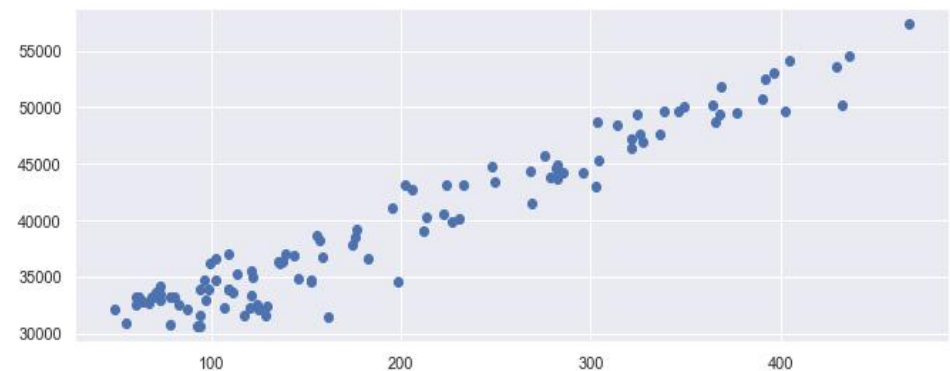
Nettoyage
Du jeu de données



Coefficient de Pearson

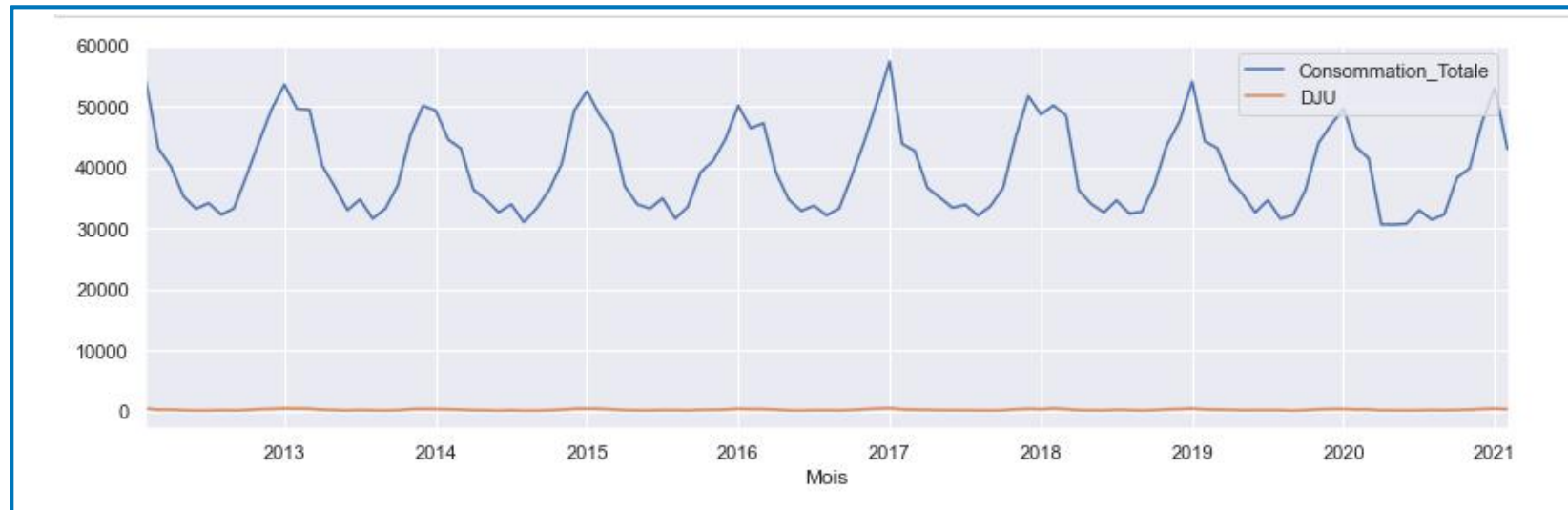
	Consommation_totale	dju	t
Consommation_totale	1.0	0.968367476650693	-0.062116312769860425
dju	0.968367476650693	1.0	0.003472066833825364
t	-0.062116312769860425	0.003472066833825364	1.0

La Courbe entre Dju et consommation

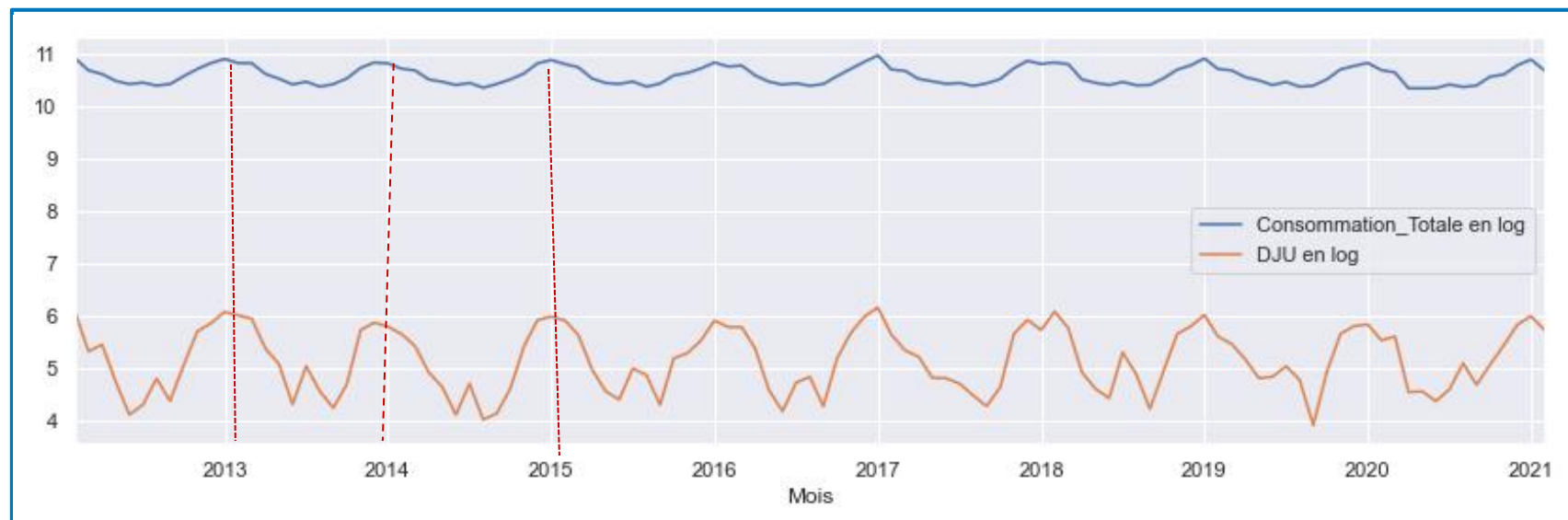


Comparaison

Consommation totale et DJU



Une forte corrélation



Correction de la DJU

Correction de l'effet température

L'effet température va influencer notre consommation donc nous allons corriger nos données températures grâce à la régression



y=notre variable cible

x=nos variables explicatives
t (temps) et dju

```
y=Elec['Consommation_totale']
x=Elec.drop(columns=['Consommation_totale'])
```

```
reg = linear_model.LinearRegression(fit_intercept=True)
reg.fit(x, y)
regression=reg.fit(x, y)
regression
```

```
LinearRegression()
```

```
print(reg.coef_,reg.intercept_)
#Ici nous avons le coefficient du dju et le 2ème coeff correspond à t=temps
#Les coeff sont affichés ds cet ordre puisque ils sont ds cet ordre ds le dataset
```

```
[ 60.35192562 -14.68978545] 28433.900809390787
```

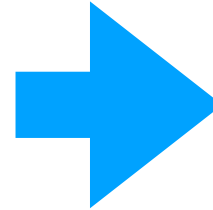
```
#Le coefficient du dju
c = reg.coef_[0]
c
```

```
60.351925623451834
```

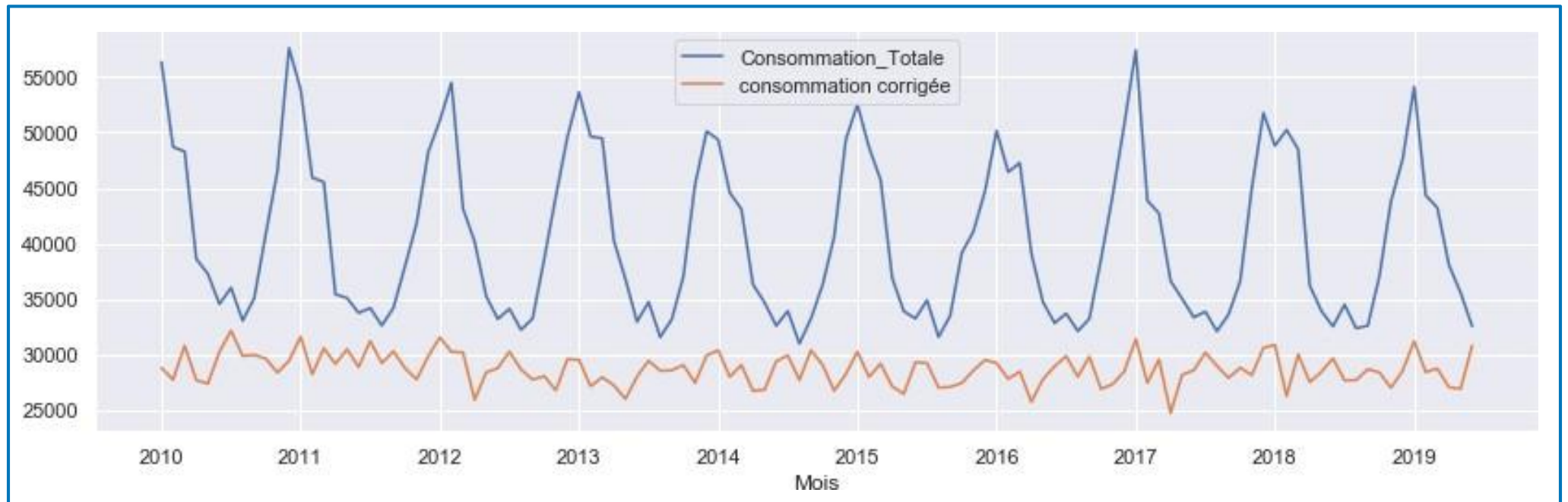
```
# calcule de serie corrigée
serie_corrige = Elec['Consommation_totale'] - Elec['dju']*c
```

Après notre régression

Mois	Consommation_totale	dju	t
2010-01-01	56342	561.2	1
2010-02-01	48698	427.4	2
2010-03-01	48294	356.5	3
2010-04-01	38637	222.6	4
2010-05-01	37284	201.6	5



serie_corrigee	
Mois	
2010-01-01	28830.713774
2010-02-01	27745.890711
2010-03-01	30817.568176
2010-04-01	27724.647338
2010-05-01	27401.114570

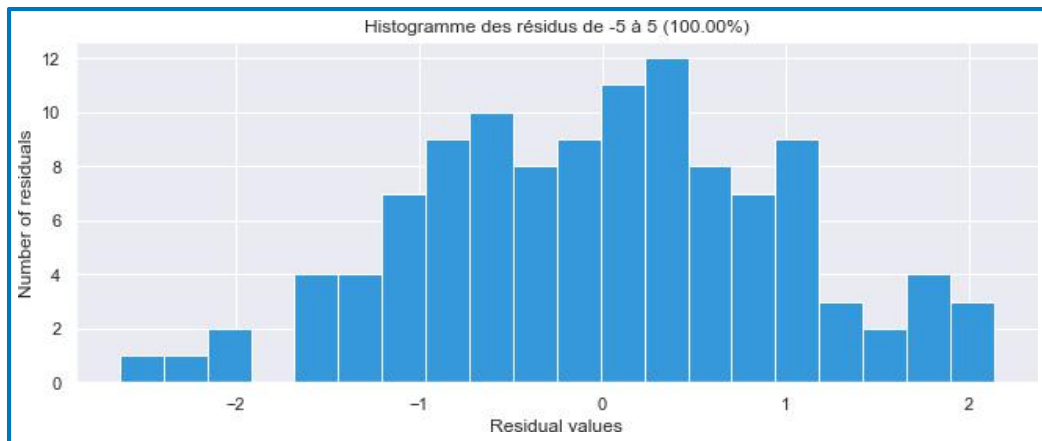


Test de nos résidus

H0 : nos résidus suivent une loi normale : $0,05 < p\text{-value}$
 H1 : les résidus ne suivent pas une loi normale : $0,05 > p\text{-value}$

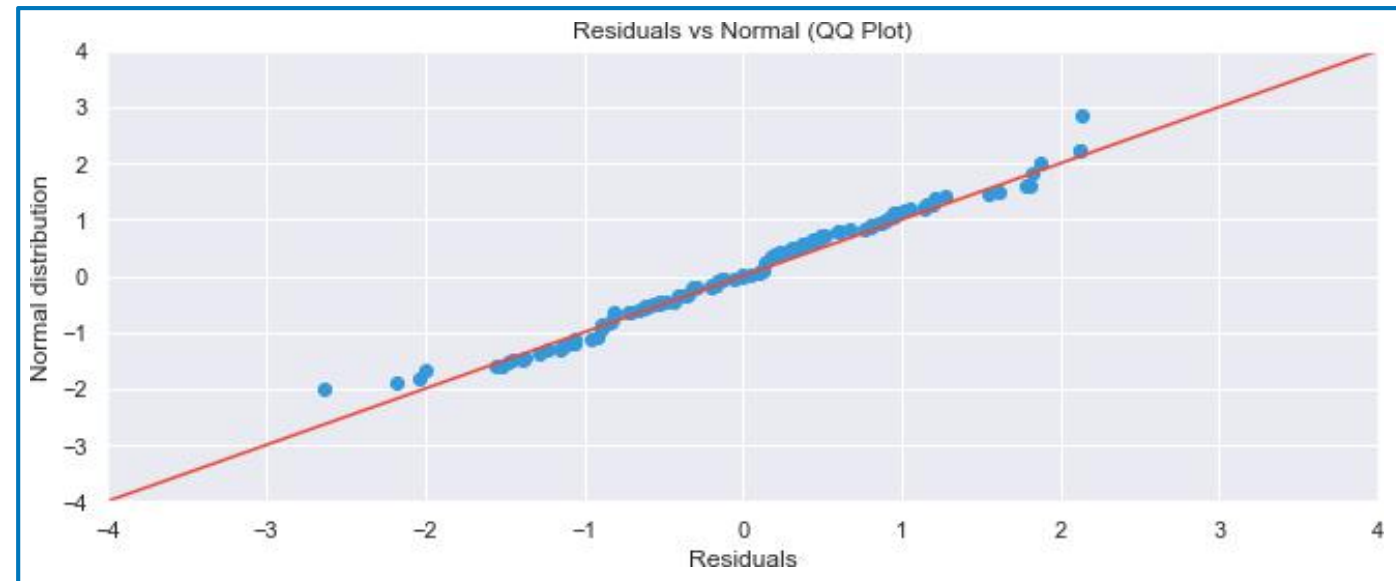
```
from scipy.stats import shapiro
shapiro(reg_multip.resid)
```

(0.9929186701774597, 0.8298814296722412)



Nos résidus sont assez bien répartis

Les valeurs sont sur la bissectrice



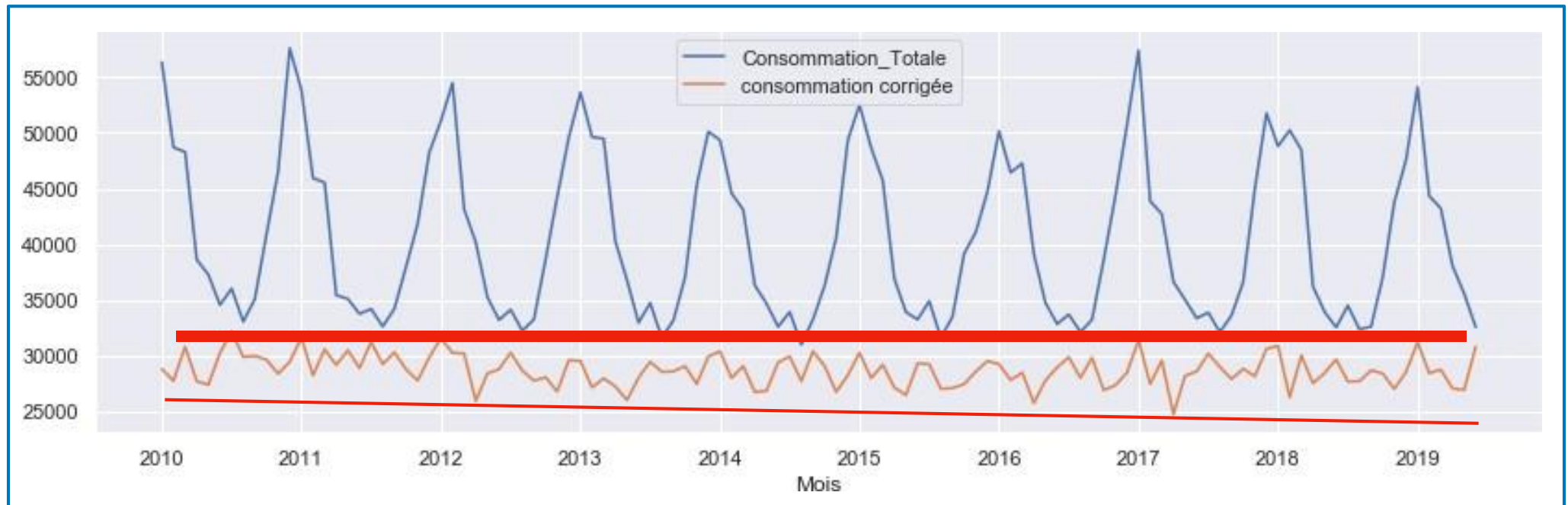
Notre modèle

Modèle additif

Les deux droites sont parallèles

Modèle multiplicatif

Les deux droites sont sécantes



Modèle additif

Les deux droites sont parallèles

Moyennes mobiles

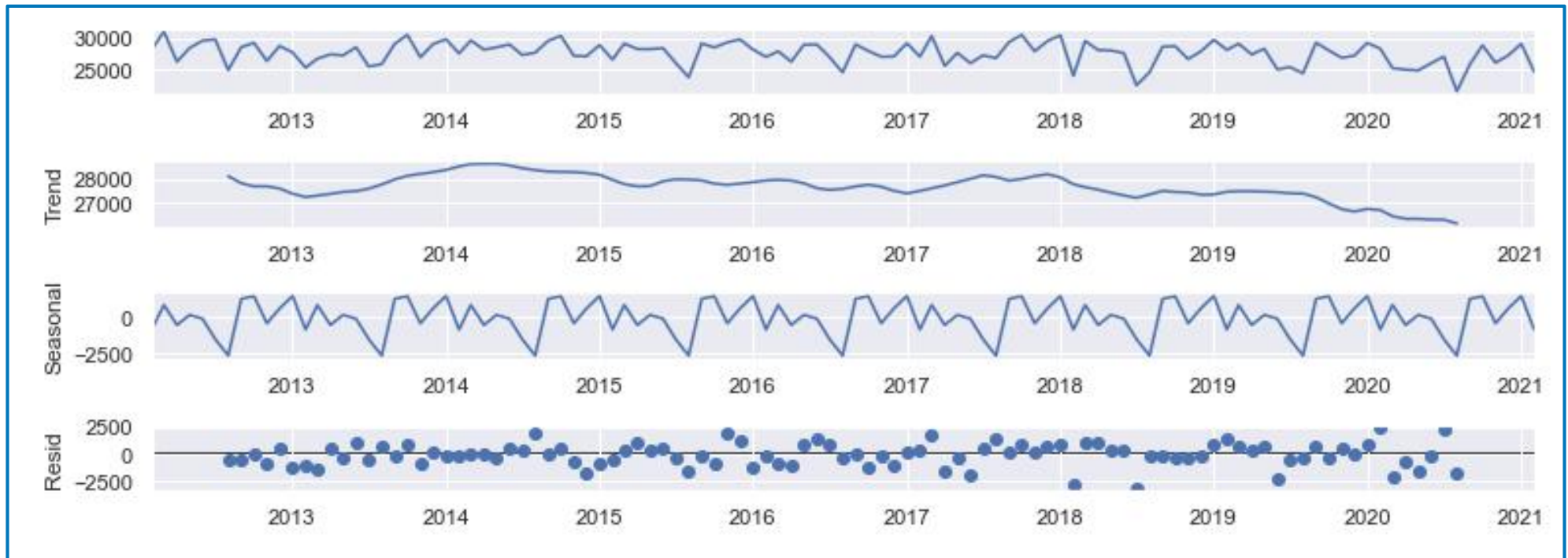
Moyenne mobile: modèle additif

01 Observé

03 Saisonnalité

02 Tendance

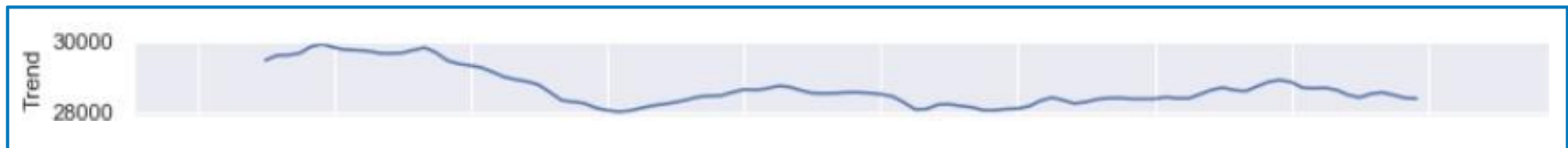
04 Résidus



Moyenne mobile

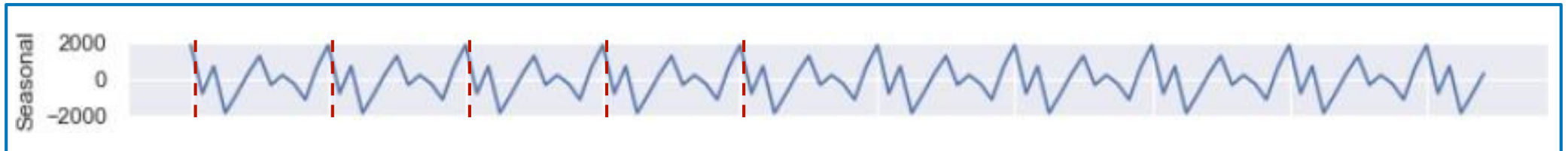


Ici nous pouvons voir notre série de base corrigée avec le DJU



Ici nous pouvons voir qu'il n'y a pas de tendance globale qui semble se dessiner, la série n'est pas en augmentation ou en baisse constante régulière, il n'y a donc pas de tendance

Moyenne mobile



Ici nous pouvons voir que une période se dessine pour la saisonnalité qui est d'un carreau soit 12 mois. Donc nous avons une période de 12 mois.

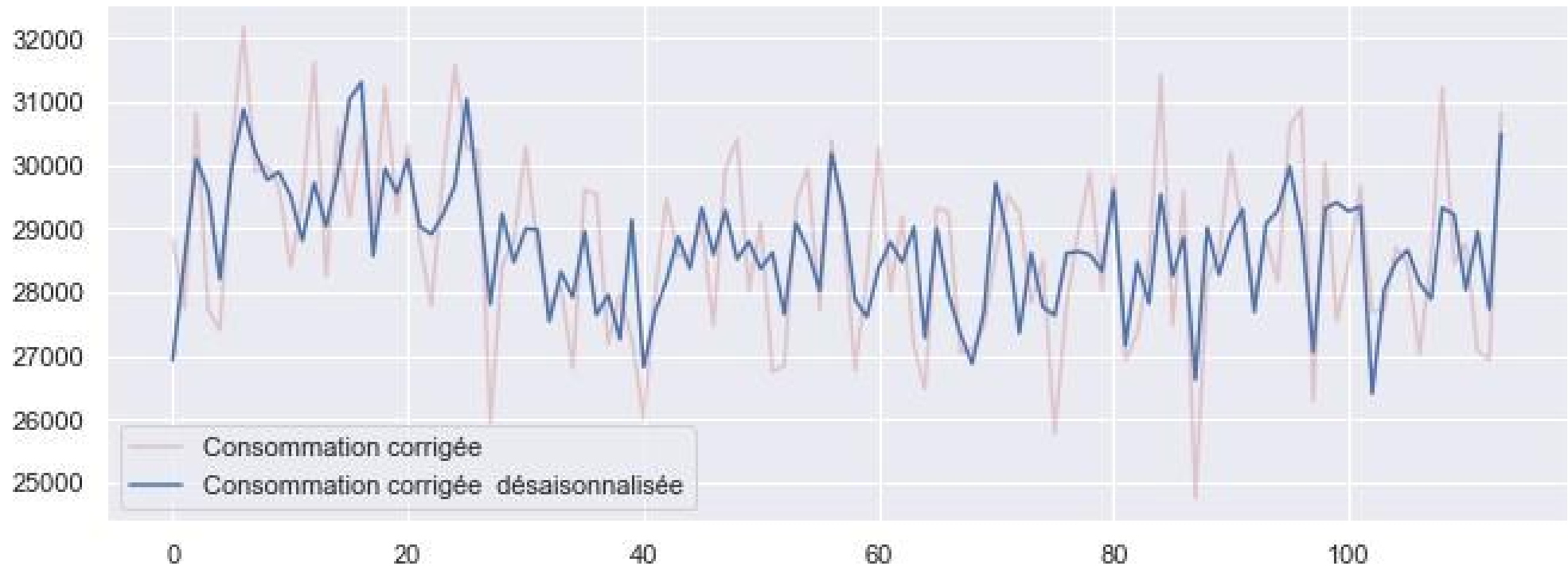
periode = S



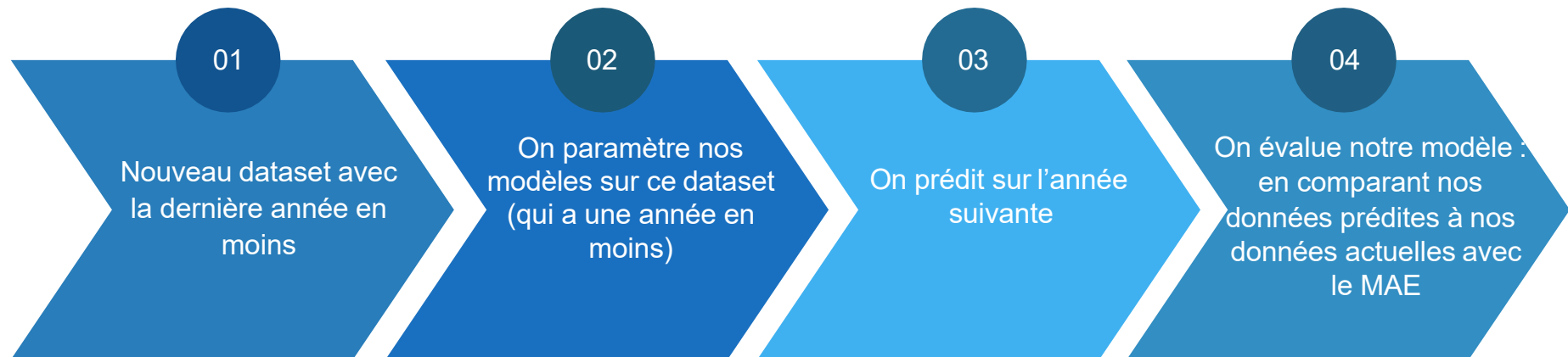
Résidus (le reste) = série observée - (tendance + saisonnalité)

Moyenne mobile

```
serie_corr_Elec['corrigee-saison']=serie_corrige.values-decomp_x.seasonal.values
```



Méthodologie pour nos modèles



Holt-Winter



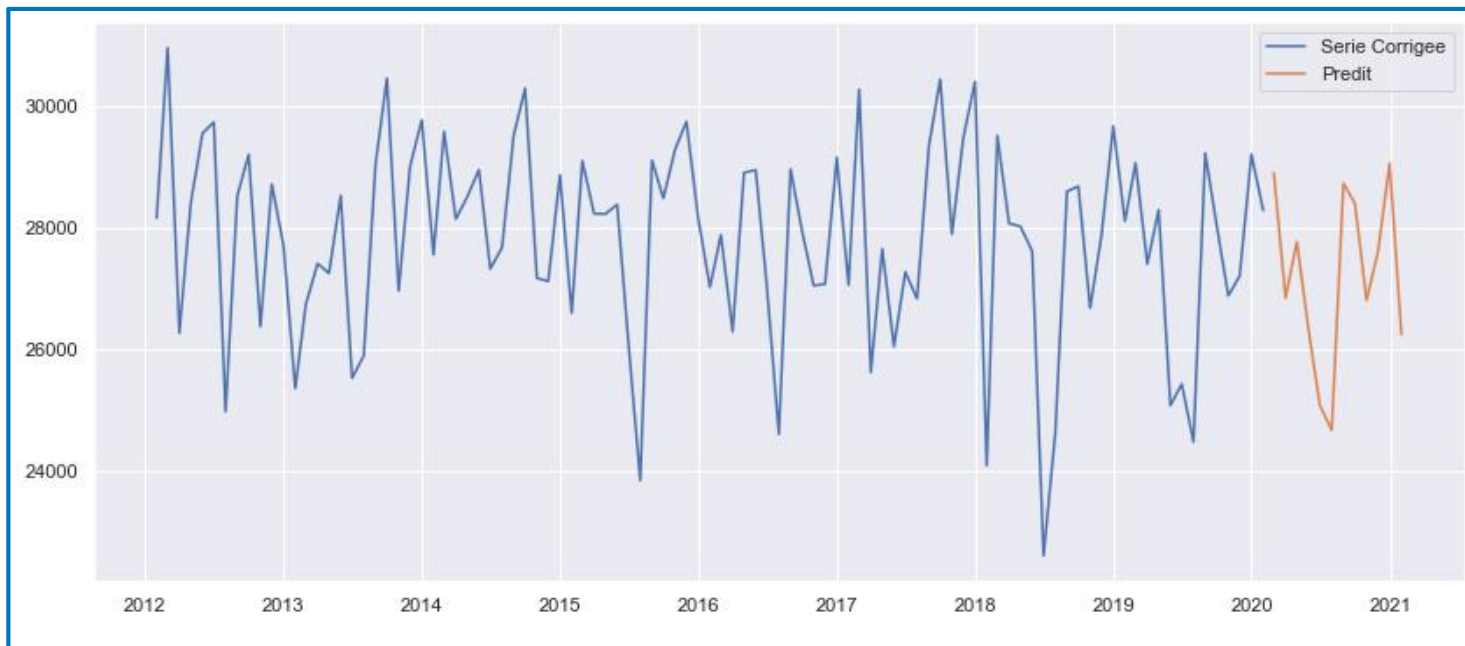
Modèle additif

```
from statsmodels.tsa.api import ExponentialSmoothing

y = np.asarray(x_tronc["corrige"])
hw = ExponentialSmoothing(y,
                          seasonal_periods=12, trend='add',
                          seasonal='add').fit()

hw_pred = hw.forecast(12)
```

```
plt.figure(figsize=(14,6))
plt.plot(x_tronc['Mois'], x_tronc['corrige'], label='Serie Corrige')
plt.plot(pd.date_range(x_tronc.Mois[x_tronc.shape[0]-1],
                      periods=12, freq='M'), hw_pred, label='Predit')
plt.legend()
```



Prédiction année -1

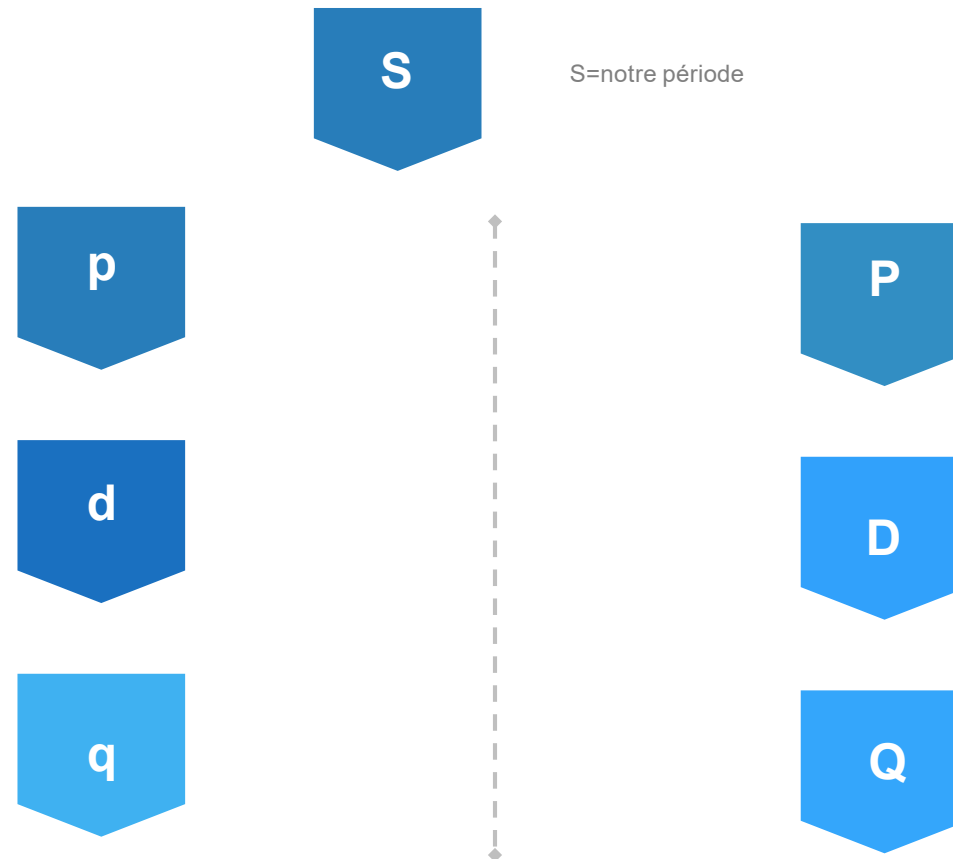
```
MAE = np.abs(x_a_prevoir['corrige'] - x_a_prevoir['prediction']).mean()
MAE
```

1620.9409644067662

Prévision de consommation

SARIMA

Sarima : (p,d,q)(P,D,Q) [S]



Tests

```
kpss(serie_corr_df['corrige'])[1]
```

```
/anaconda3/lib/python3.6/site-packag  
behavior of using lags=None will cha  
as lags='legacy', and so a sample-si  
t will change to be the same as lags  
od. To silence this warning, either  
warn(msg, FutureWarning)
```

```
0.08517684021448321
```

KPSS : H0 la série est stationnaire

P_values>5% : accepte H0

Ici on accepte H0 mais la P-value très proche
du seuil critique

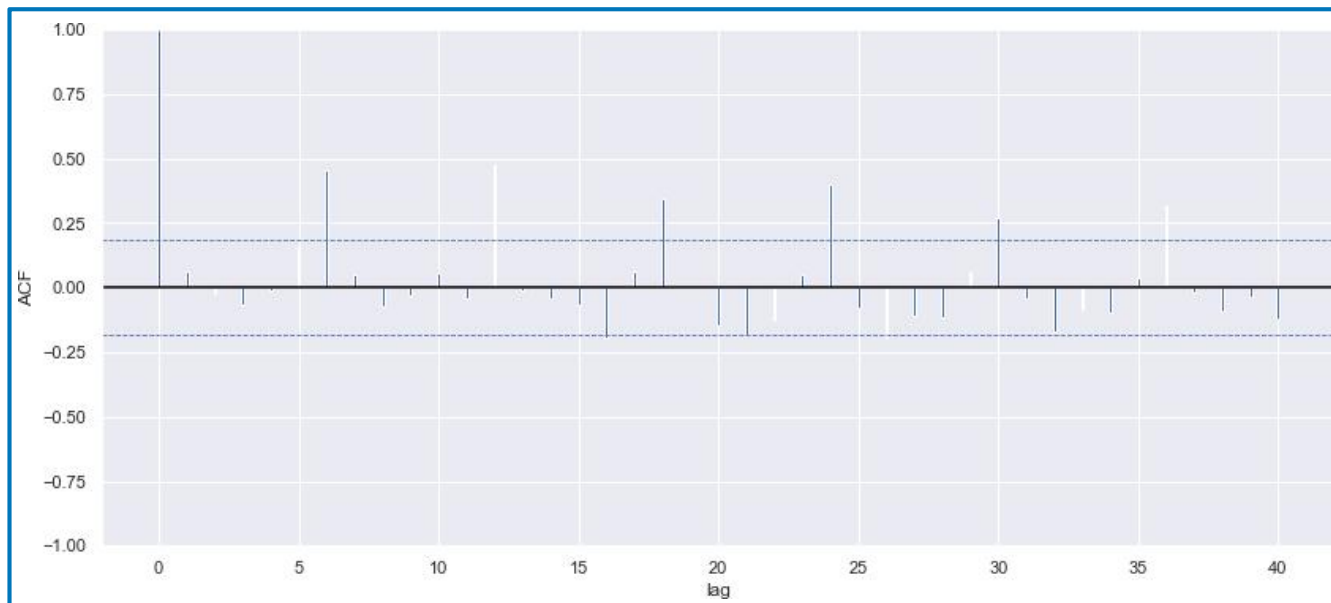
```
adfuller(serie_corr_df['corrige'])[1]
```

```
0.3682866645959178
```

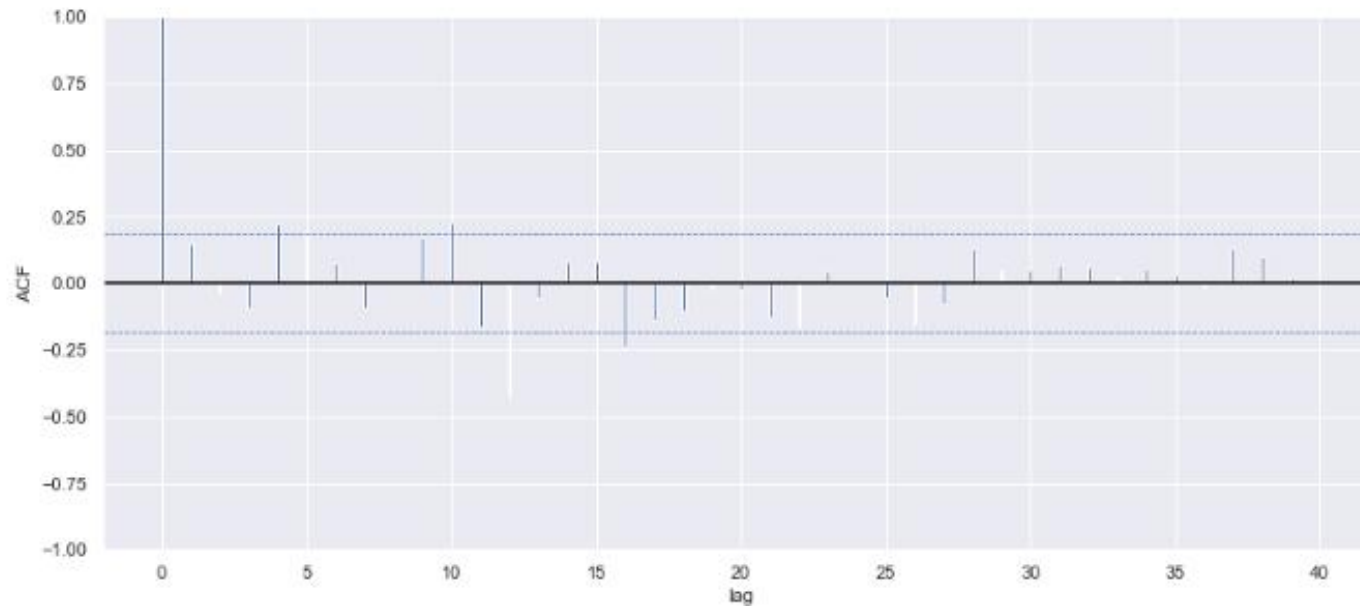
AD Fuller : H0 la série n'est pas stationnaire

P_values>5% : accepte H0

Ici on accepte H0 donc notre série n'est pas
stationnaire



```
y_dif1 = serie_corr_df['corrige'] - serie_corr_df['corrige'].shift(12)
plot_sortie_acf(acf(np.asarray(y_dif1[13:]])), y_len)
```



```
adfuller(y_dif1[12:])[1]
#adfuller : H0 la serie n'est pas stationnaire'
```

0.005691344568372587

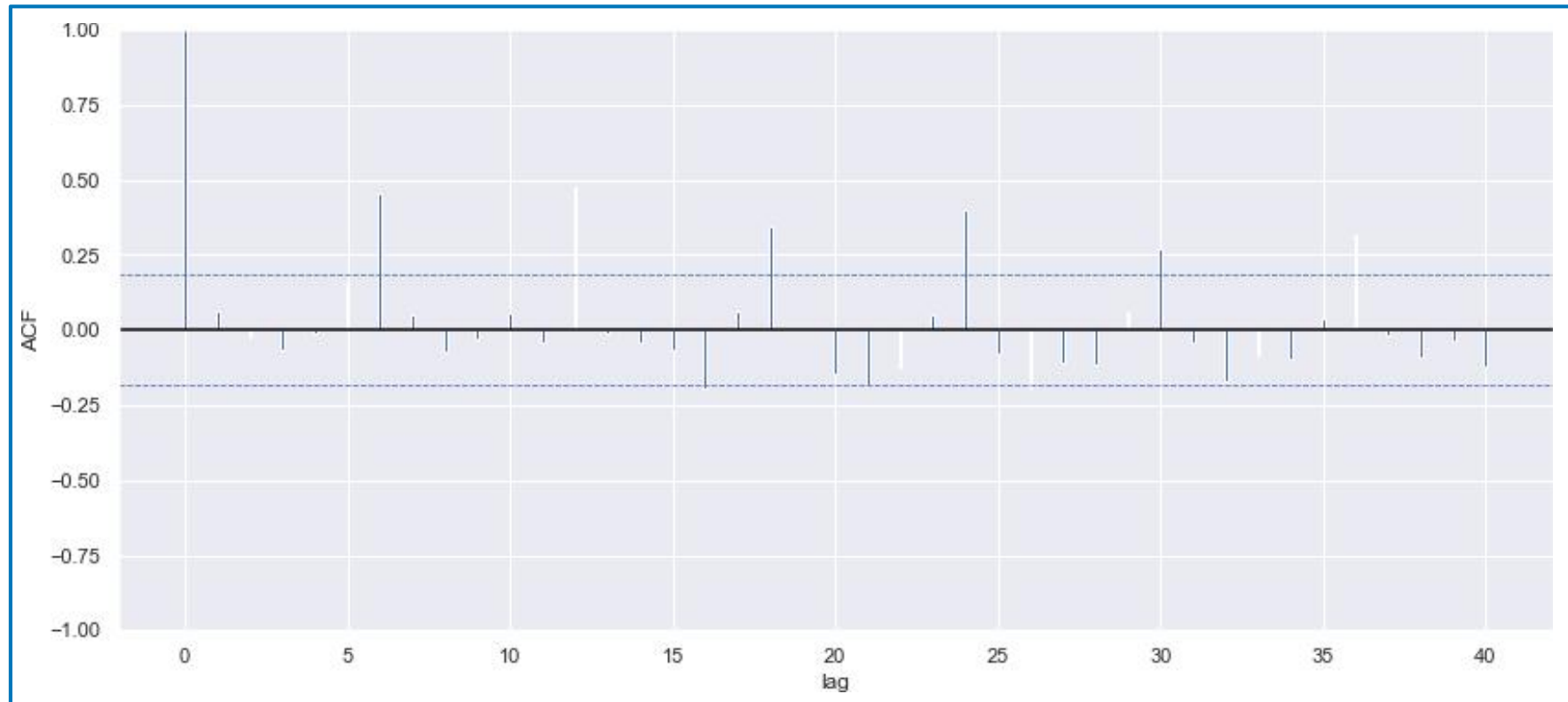
AD Fuller : H0 la série n'est pas stationnaire
P_values<5% : accepte H1
Ici on accepte H1 donc notre série est stationnaire

```
kpss(y_dif1[12:])[1]
/anaconda3/lib/python3.6/s
behavior of using lags=Nor
as lags='legacy', and so a
t will change to be the se
od. To silence this warnin
warn(msg, FutureWarning)
/anaconda3/lib/python3.6/s
g: p-value is greater than
warn("p-value is greater
```

0.1

KPSS : H0 la série est stationnaire
P_values>5% : accepte H0
Ici on accepte H0 donc la série est stationnaire

Sarima paramètre : d

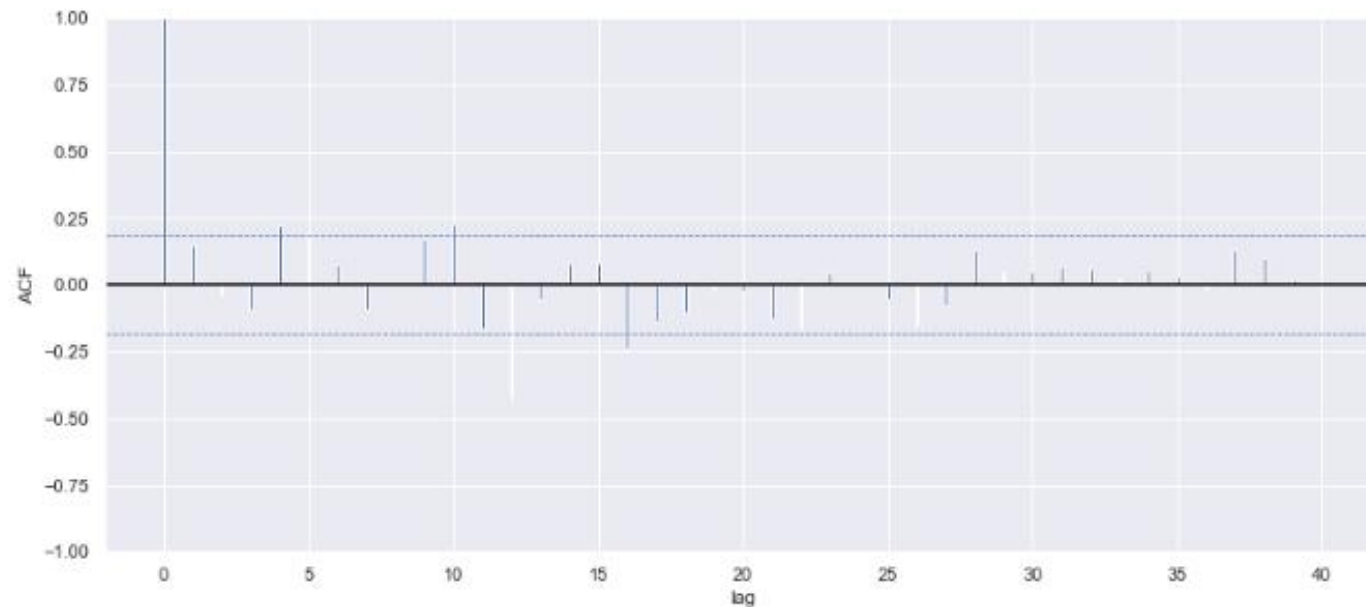


Décroissante lente au niveau des lags 12, on n'effectue pas de différentiation en tendance donc $d=0$ mais une différenciation en saisonnalité donc $D=1$

$d=0$

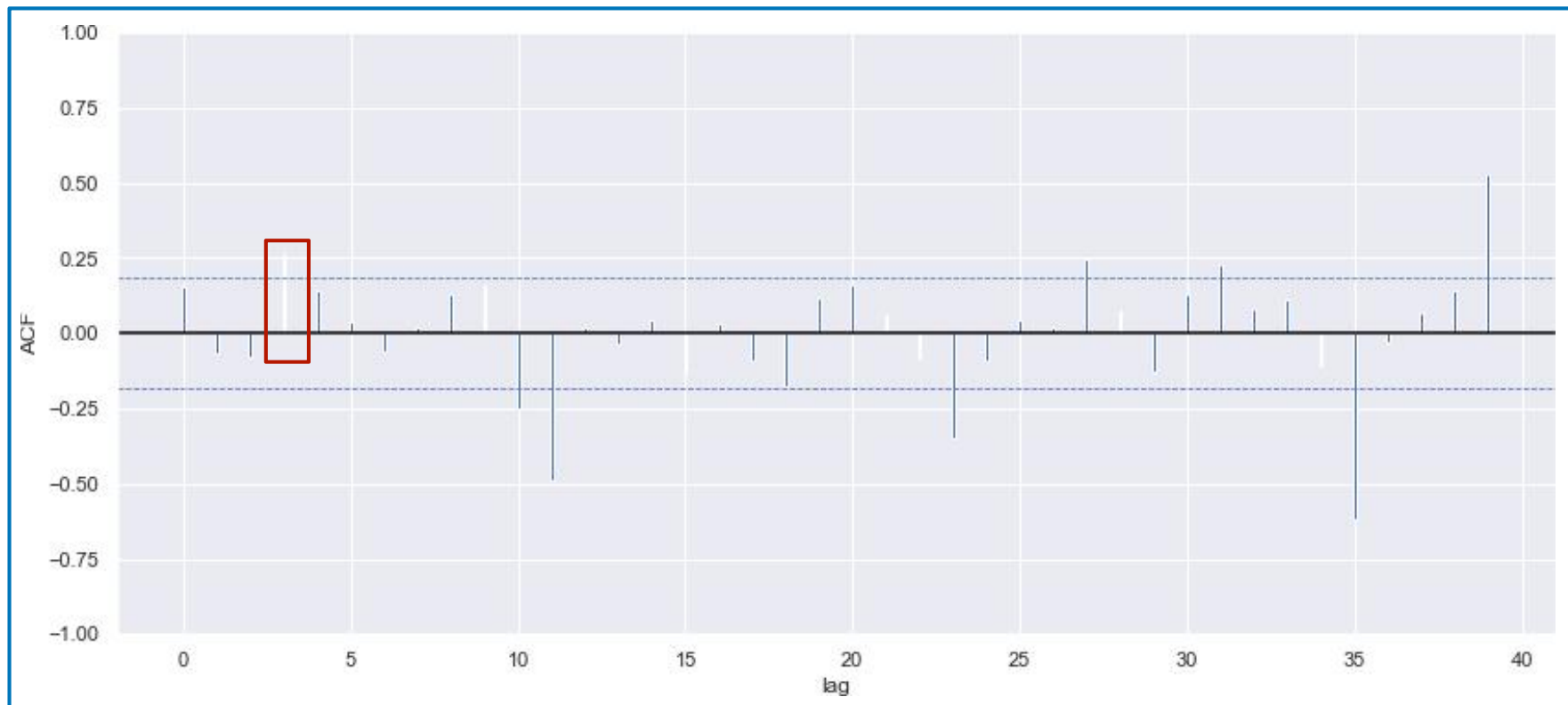
Sarima paramètre : D

```
y_dif1 = serie_corr_df['corrige'] - serie_corr_df['corrige'].shift(12)  
plot_sortie_acf(acf(np.asarray(y_dif1[13:])), y_len)
```



D=1

Sarima paramètre : p

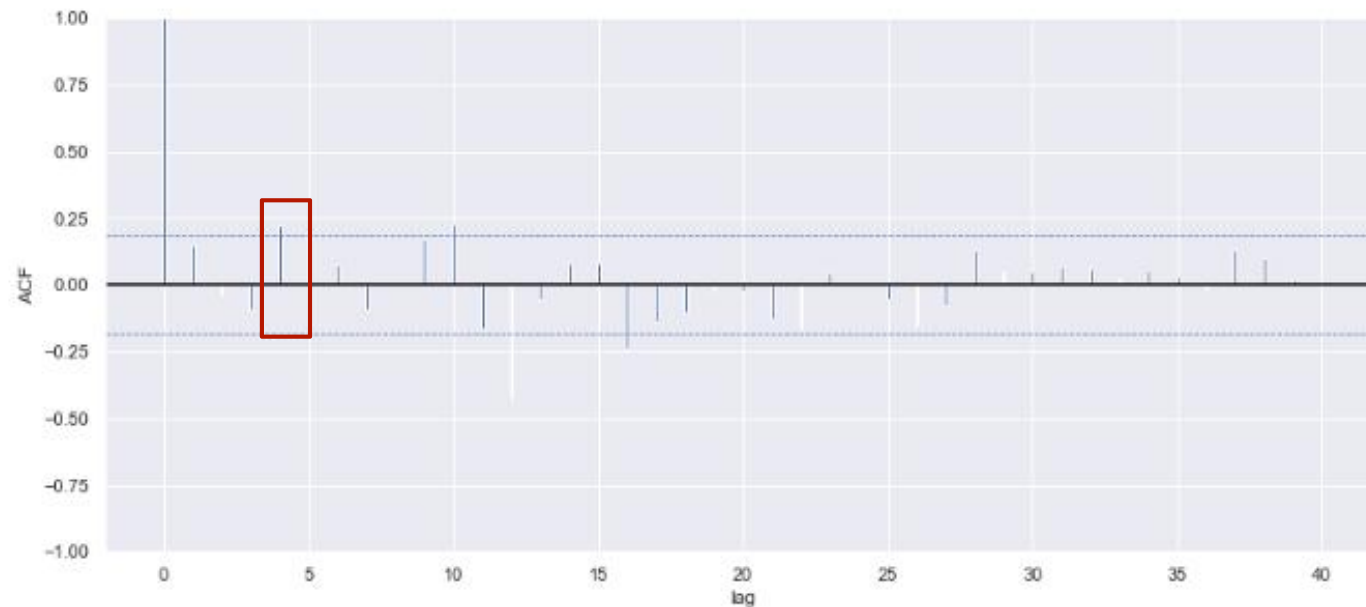


premier lag où le PACF dépasse le seuil de significativité

p=1

Sarima paramètre : q

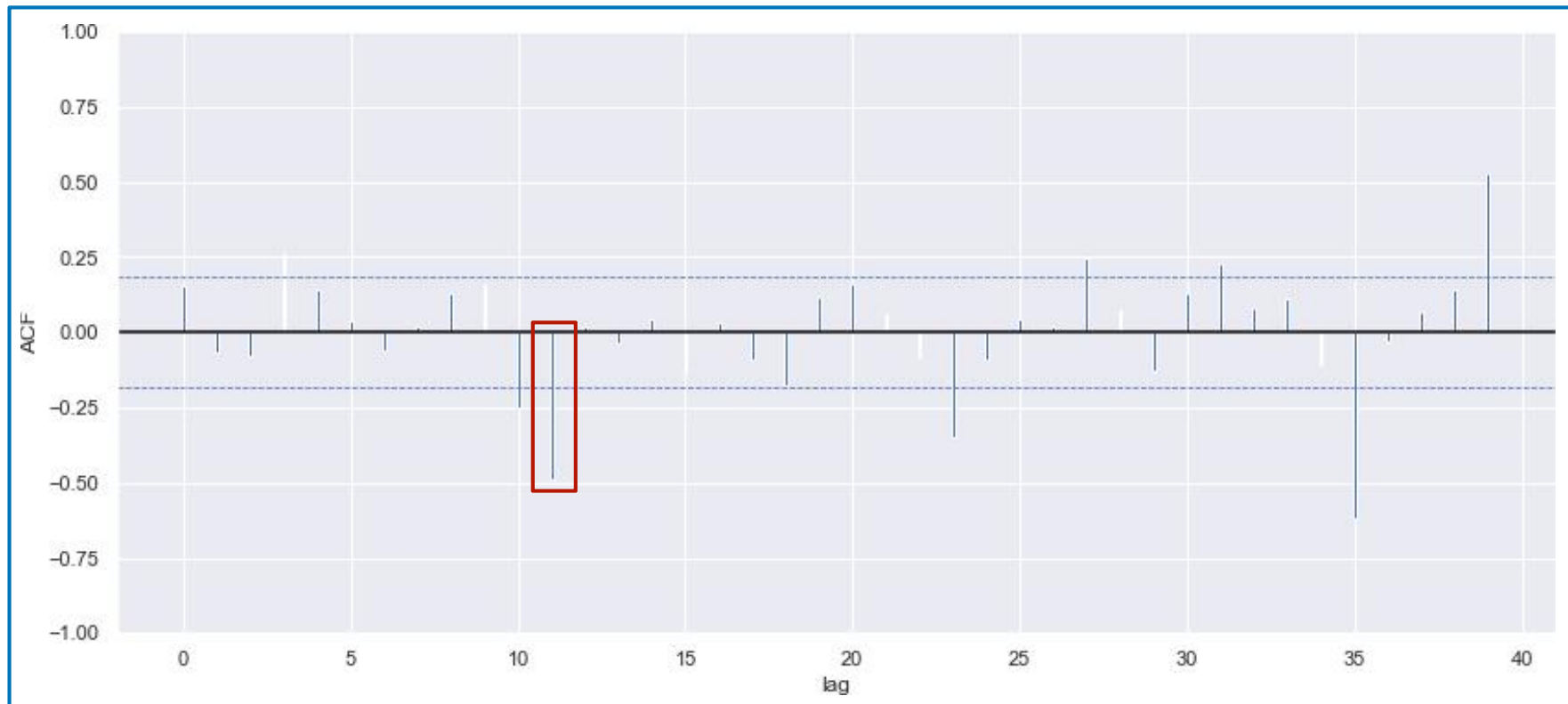
```
y_dif1 = serie_corr_df['corrige'] - serie_corr_df['corrige'].shift(12)  
plot_sortie_acf(acf(np.asarray(y_dif1[13:])), y_len)
```



premier lag où l'ACF dépasse le seuil de significativité

q=1

Sarima paramètre : P



On regarde les lags saisonniers qui sortent du seuil de significativité sur le PACF

P=1

Sarima paramètre : Q

```
y_dif1 = serie_corr_df['corrige'] - serie_corr_df['corrige'].shift(12)  
plot_sortie_acf(acf(np.asarray(y_dif1[13:])), y_len)
```



On regarde les lags saisonniers qui sortent du seuil de significativité

Q=1

SARIMA : modèle 1

```

from statsmodels.tsa.statespace.sarimax import *
from statsmodels.stats.diagnostic import acorr_ljungbox

modell = SARIMAX(np.asarray(x_tronc["corrigee"]), order=(1,0,1), seasonal_order=(1,1,1,12))
results1 = modell.fit()
print(results1.summary())

#print('Retard : p-value')
#for elt in [6, 12, 18, 24, 30, 36]:
#    #print('{} : {}'.format(elt, acorr_ljungbox(results1.resid, lags=elt)[1].mean()))

```

Statespace Model Results

```

=====
Dep. Variable:          y      No. Observations:          102
Model:                SARIMAX(1, 0, 1)x(1, 1, 1, 12)      Log Likelihood          -766.227
Date:                  Sat, 14 Dec 2019                  AIC              1542.454
Time:                  10:18:06                          BIC              1554.953
Sample:                0                                HQIC              1547.494
                    - 102
Covariance Type:      opg
=====

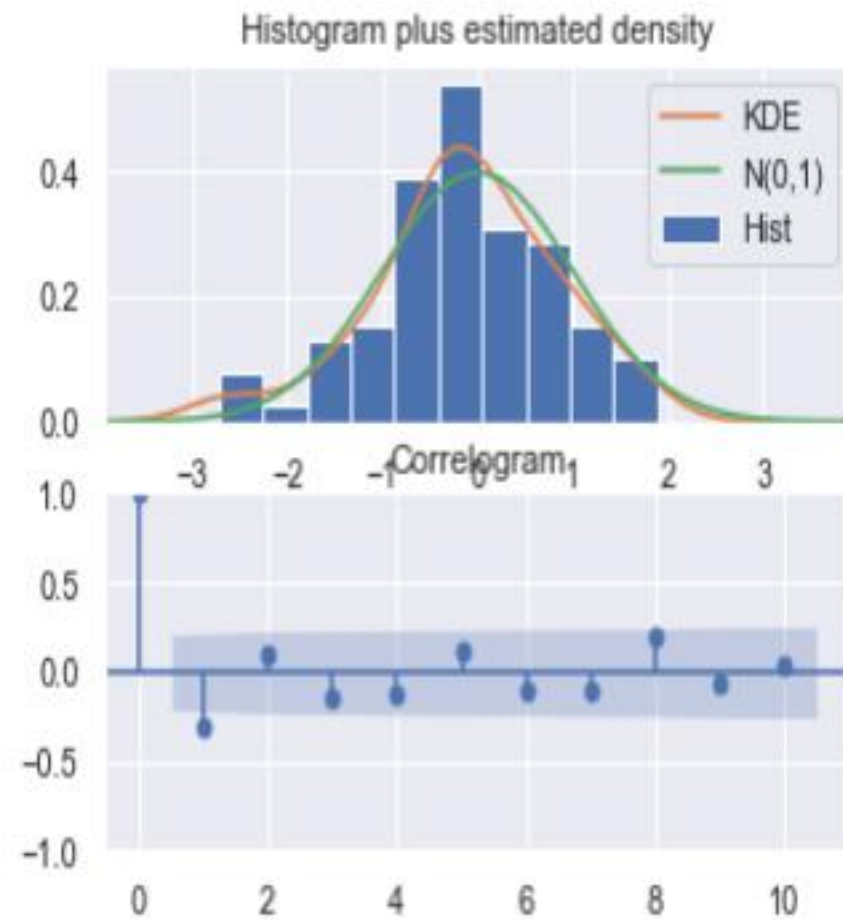
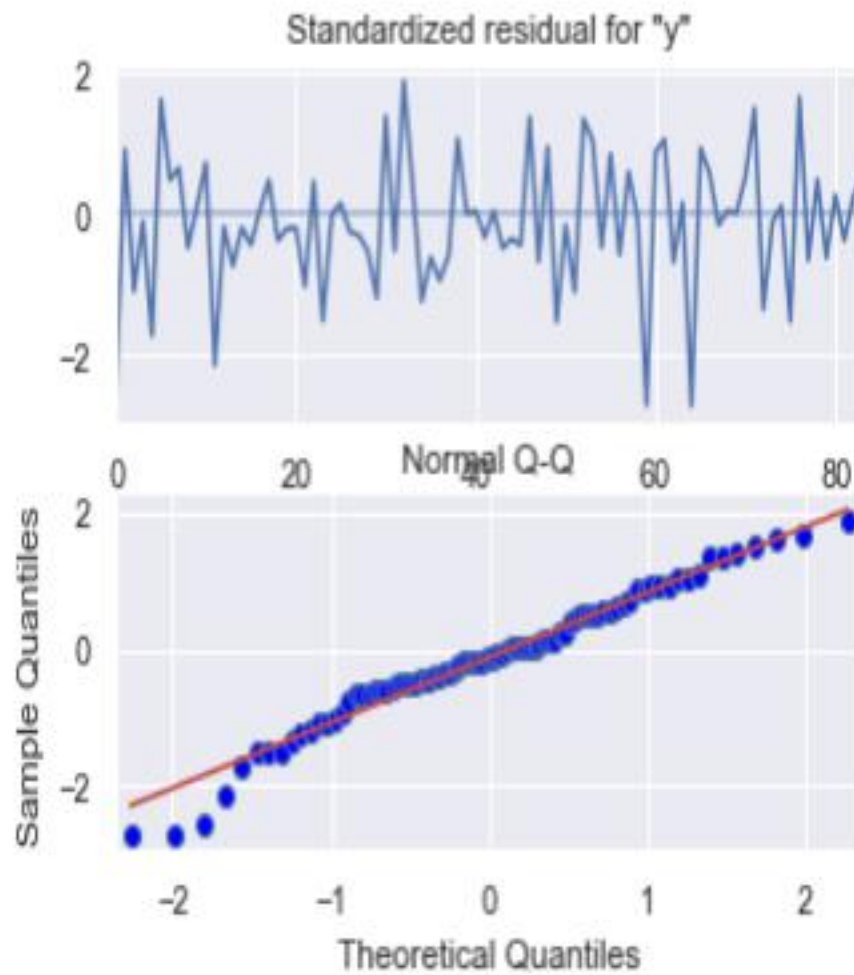
```

	coef	std err	z	P> z	[0.025	0.975]
ar.L1	0.9489	0.031	30.241	0.000	0.887	1.010
ma.L1	-0.9064	0.036	-25.291	0.000	-0.977	-0.836
ar.S.L12	0.3682	0.131	2.815	0.005	0.112	0.625
ma.S.L12	-0.6148	0.166	-3.710	0.000	-0.940	-0.290
sigma2	1.719e+06	2.35e-09	7.3e+14	0.000	1.72e+06	1.72e+06

```

=====
Ljung-Box (Q):          48.39      Jarque-Bera (JB):          0.92
Prob(Q):                0.17      Prob(JB):                0.63
Heteroskedasticity (H): 0.74      Skew:                    -0.22
Prob(H) (two-sided):    0.42      Kurtosis:                2.77
=====

```



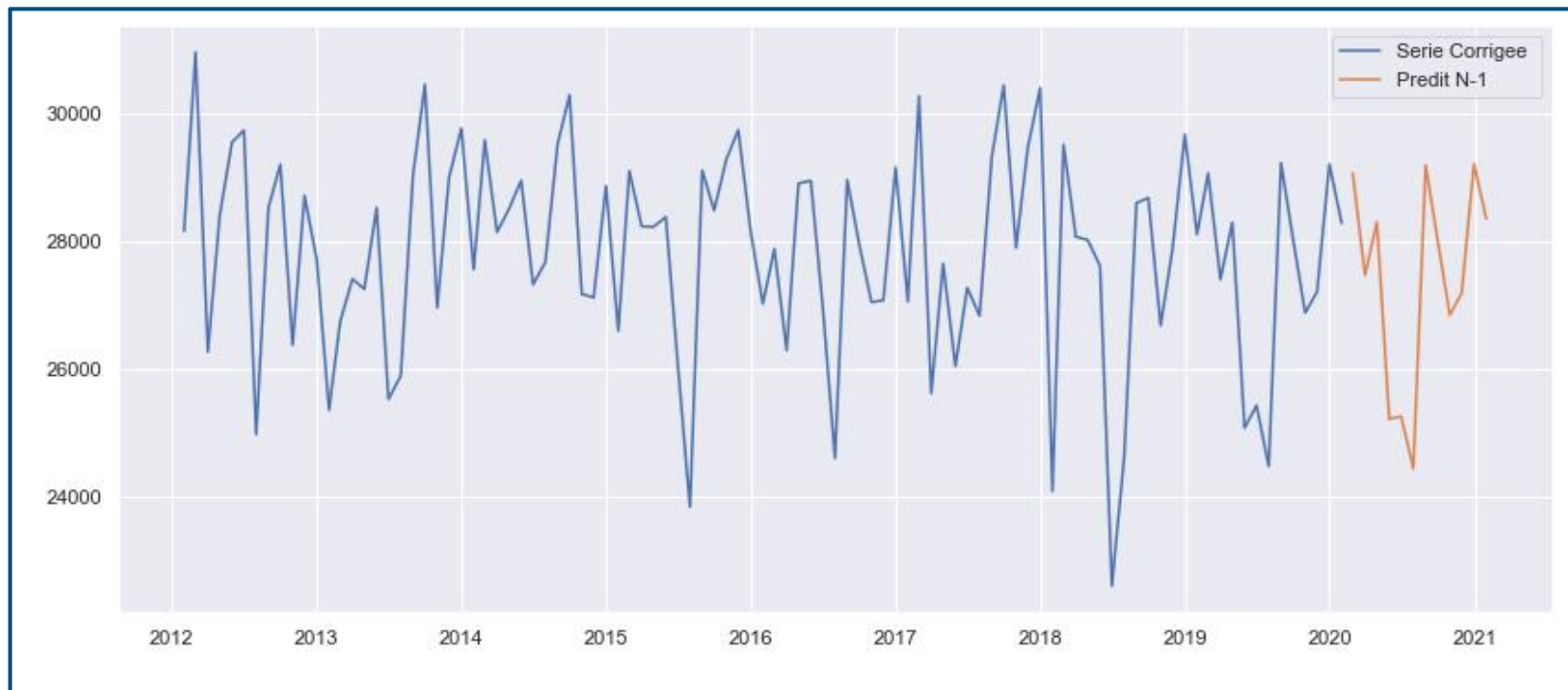
Modélisation du

```

model1 = SARIMAX(np.asarray(x_tronc["corrige"]), order=(1,0,1), seasonal_order=(1,1,1,12))
results1 = model1.fit()
sarima_pred=results1.forecast(12)

plt.figure(figsize=(14,6))
plt.plot(x_tronc['Mois'],x_tronc['corrige'], label='Serie Corrige')
plt.plot(pd.date_range(x_tronc.Mois[x_tronc.shape[0]-1],
                      periods=12, freq='M'), sarima_pred, label='Predit corrigée sans dju')
plt.legend()

```



```

pred_model2tronc = results1.get_forecast(12)
pred_tronc = pred_model2tronc.predicted_mean
MAE_sarima_model2=np.abs(x_a_prevoir['corrige']-pred_tronc).mean()
MAE_sarima_model2

```

1976.4174959276188

Sarima N°2

```

: import pmdarima as pm

# Seasonal - fit stepwise auto-ARIMA
smodel = pm.auto_arima(x_tronc["corrigee"],
                      test='adf',
                      m=12,
                      seasonal=True,
                      d=0, D=1, trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)

smodel.summary()

```

Dep. Variable:	y	No. Observations:	102
Model:	SARIMAX(1, 0, 0)x(0, 1, 0, 12)	Log Likelihood	-772.426
Date:	Sat, 14 Dec 2019	AIC	1550.853
Time:	10:18:12	BIC	1558.352
Sample:	0	HQIC	1553.877
	- 102		
Covariance Type:	opg		

	coef	std err	z	P> z	[0.025	0.975]
intercept	-71.7965	138.702	-0.518	0.605	-343.648	200.055
ar.L1	0.1588	0.099	1.605	0.108	-0.035	0.353
sigma2	1.684e+06	2.58e+05	6.536	0.000	1.18e+06	2.19e+06

Ljung-Box (Q):	64.54	Jarque-Bera (JB):	0.00
Prob(Q):	0.01	Prob(JB):	1.00
Heteroskedasticity (H):	0.83	Skew:	0.01
Prob(H) (two-sided):	0.61	Kurtosis:	2.98

```

: MAE_sarima_model_optim=np.abs(x_a_prevoir['corrigee']-pred_tronc).mean()
MAE_sarima_model_optim

: 1246.9854384675446

```

Comparaison de nos modèles

Modèle additif

HOLT-WINTERS

```
MAE=np.abs(x_a_prevoir['corrige']-x_a_prevoir['prediction']).mean()  
MAE
```

1620.9409644067662

SARIMA

```
pred_model2tronc = results1.get_forecast(12)  
pred_tronc = pred_model2tronc.predicted_mean  
MAE_sarima_model2=np.abs(x_a_prevoir['corrige']-pred_tronc).mean()  
MAE_sarima_model2
```

1976.4174959276188

Holt-Winter : année+1

