Prédiction de la demande d’électricité en France

Chez Enercoop société coopérative spécialisée dans les énergies renouvelables.

Projet 9 - Nalron Mai 2020
OpenClassrooms
ENSAE-ENSAI Formation continue

Langage utilisé Python sur Jupiter Notebook

Qu'est-ce qu'une énergie renouvelable ?

Une énergie est dite renouvelable lorsqu'elle provient de sources que la nature renouvelle en permanence, par opposition à une énergie non renouvelable dont les stocks s'épuisent.

Les énergies renouvelables proviennent de 2 grandes sources naturelles : le Soleil (à l'origine du cycle de l'eau, des marées, du vent et de la croissance des végétaux) et la Terre (qui dégage de la chaleur).

Surnommées « énergies propres » ou « énergies vertes », leur exploitation engendre très peu de déchets et d'émissions polluantes mais leur pouvoir énergétique est beaucoup plus faible que celui des énergies non renouvelables.

Source : EDF

À la force du vent

Appelée aussi « énergie du vent », la solution des éoliennes est la plus utilisée en Europe. Celles-ci permettent de transformer la force du vent en électricité grâce à un générateur.

- Une éolienne de 2 MW fournit de l'électricité pour **2 000 PERSONNES**, chauffage compris.

Toute la puissance du soleil

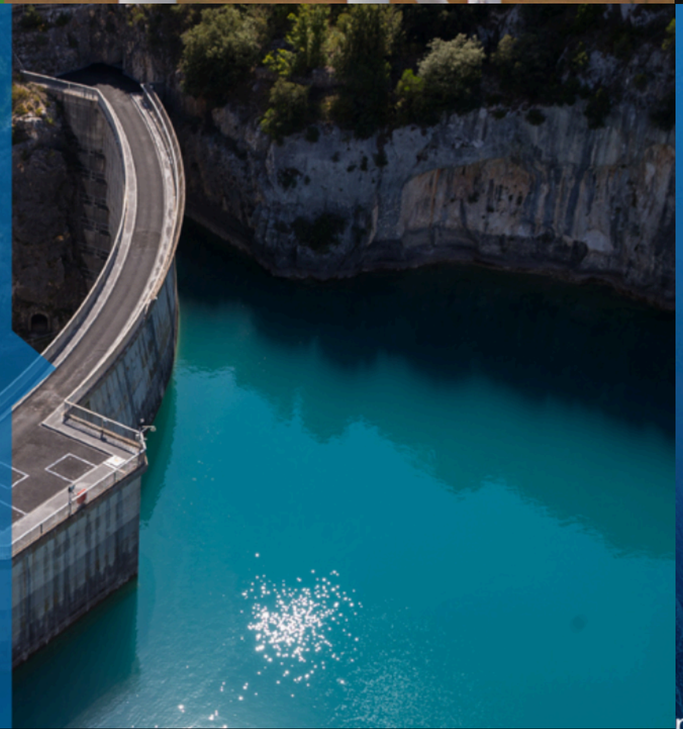
L'utilisation de cellules photovoltaïques, essentiellement composées de silicium, permet de capter et transformer la lumière du soleil en électricité.

- **480 CELLULES MONOCRISTALLINES** équipent la smartflower, un tournesol photovoltaïque géant qui suit la course du soleil.

La force de l'eau au coeur du procédé

Exploitée en France depuis la fin du 19e siècle, l'hydroélectricité utilise la force de l'eau (fleuves, rivières…) pour fabriquer de l'électricité dans les centrales hydroélectriques, alimentées en partie par des barrages.

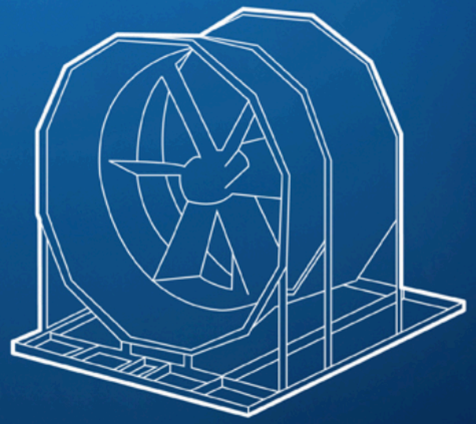
- La France métropolitaine compte **2 250 INSTALLATIONS HYDROÉLECTRIQUES.**



Quand le courant marin fait tourner les turbines

Saviez-vous que les courants océaniques ou issus des marées peuvent permettre la production d'électricité ? En effet, la rotation des turbines installées en mer entraîne un alternateur produisant un courant électrique variable.

- En France, il y a une seule usine marémotrice, la Rance. Elle produit chaque année **4% DE L'ÉLECTRICITÉ CONSOMMÉE** en Bretagne.



Votre fournisseur d'électricité est-il vraiment vert ?

Retrouvez les analyses détaillées sur le guide en ligne → **guide-electricite-verte.fr**

Vraiment verts

Ce sont les meilleurs à l'heure actuelle. **Vous faites le choix d'une électricité vraiment renouvelable,** sans énergies fossiles ni énergie nucléaire.

1er EX-ÆQUO enercoop

1er EX-ÆQUO planète OUI

Urban Solar

ilek

plüm ENERGIE

En bonne voie

Ils sont **loin d'être parfaits mais font des efforts** pour acheter et produire des énergies renouvelables, ou investir dans ce secteur.

mint energie

ekWateur

alterna électricité - gaz

Sélia

GEG

À la traîne

Ils font **encore trop peu d'efforts, voire aucun,** en matière d'électricité renouvelable.

énergem fournisseur de confiance

Cdiscount · Énergie

greenyellow

lucia

énergies E.Leclerc

OHM ÉNERGIE

Butagaz

proxelia

ÉNERGIES DU SANTERRE

Vraiment mauvais

Ils appartiennent à de grands groupes qui **polluent la planète en investissant dans des énergies sales et dangereuses** (charbon, gaz de schiste, nucléaire...).

IBERDROLA

VATTENFALL

happe

ENGIE

TOTAL direct energie

sowee

EDF

eni

Traitement des données

Données mensuelles de consommation totale d'électricité en énergie:
http://www.rte-france.com/fr/eco2mix/eco2mix-telechargement

Données météo pour corriger les données de l'effet température:
https://cegibat.grdf.fr/simulateur/calcul-dju

Pré-traitement des données

Données de consommation totale

- Chargement des données
- Restriction sur le pays France
- Projection sur la consommation
- Suppression des NaN
- Indexation selon le time

```python
#Chargement du dataset 'consommation.csv'
data_conso = pd.read_csv('p9_02_dataset/consommation.csv', sep=';', parse_dates=[0])
data_conso.head(10)
```

	Mois	Qualité	Territoire	Production totale	Production nucléaire	Production thermique totale	Production thermique charbon	Production thermique fioul	Production thermique gaz	Production hydraulique	...	Consommation totale	Solde exportateur
0	0000-00	Données consolidées	Grand-Est	11346	8643.0	1120	22.0	2	1095	565	...	4545	6711.0
1	0000-00	Données consolidées	Nouvelle-Aquitaine	5289	4179.0	164	NaN	0	164	419	...	4578	711.0
2	0000-00	Données consolidées	Auvergne-Rhône-Alpes	11622	8382.0	334	6.0	11	316	2630	...	6834	4483.0
3	0000-00	Données consolidées	Bourgogne-Franche-Comté	467	NaN	124	NaN	0	123	89	...	2188	-1724.0
4	0000-00	Données consolidées	Bretagne	535	NaN	182	NaN	4	178	59	...	2443	-1915.0
5	0000-00	Données consolidées	Centre-Val de Loire	7434	6951.0	87	NaN	0	87	17	...	2031	5402.0
6	0000-00	Données consolidées	France	54572	37352.0	5470	157.0	149	5163	6350	...	49559	4252.0
7	0000-00	Données consolidées	Ile-de-France	393	NaN	259	0.0	6	252	3	...	7539	-7145.0

```python
#Création d'un jeu de données mensuelles de consommation totale d'électricité en énergie
#Restriction sur la modalité 'France' et projection sur les Series pertinentes
conso = data_conso.copy()
conso = conso[['Mois', 'Territoire', 'Consommation totale']]
conso = conso[conso['Territoire'] == 'France']
conso.drop('Territoire', axis=1, inplace=True)
conso.rename(columns={'Mois': 'mois', 'Consommation totale': 'consommation'}, inplace=True)
#Changement du type de Series en datetime
#Changement d'index pour que la série temporelle soit reconnue
conso['mois'] = pd.to_datetime(conso['mois'])
conso.set_index('mois', inplace=True)
```

mois	consommation
2012-01-01	51086
2012-02-01	54476
2012-03-01	43156
2012-04-01	40176
2012-05-01	35257
...	...

Pré-traitement des données

Données des DJU

- Chargement des données
- Suppression des lignes inutiles
- Transformation par itération
- Matrice de série temporelle

Une jointure par méthode merge interne est nécessaire pour la création d'un échantillon complet.

```python
#Chargement du dataset 'dju.xlsx'
data_dju = pd.read_excel('p9_02_dataset/dju.xlsx', header=None, skiprows=11)
data_dju.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	NaN	JAN	FÉV	MAR	AVR	MAI	JUN	JUI	AOÛ	SEP	OCT	NOV	DÉC	Total
1	2019.0	404.9	268.3	233.1	168.5	117.9	14	0	0	0	0	0	0	1206.6
2	2018.0	303.4	432.6	314.3	119.7	55.9	8.1	0	3.3	34.3	122.4	282.5	325.9	2002.2
3	2017.0	467.9	278.4	206.1	182.6	75	9.4	1	6.8	62.6	99.4	282.6	369	2040.6
4	2016.0	364.4	321.6	321.1	212.1	88.1	27.5	5.7	3.2	11.7	176	285.6	390.8	2207.3

```python
#Suppression des deux premières lignes et dernière Series
data_dju.drop([0, 1], axis=0, inplace=True)
data_dju.drop([13], axis=1, inplace=True)

#La première Serie est considérée comme index
data_dju[0] = data_dju[0].astype(int)
data_dju.set_index([0], inplace=True)
```

```python
#Transformation des données
dju = {'mois':[],'dju_chauffage':[]}

for year in data_dju.index.values:
    for month in data_dju.columns:
        dju['mois'].append(f"{year}-{month}")
        dju['dju_chauffage'].append(data_dju.loc[year,month])

dju = pd.DataFrame(dju)
dju['mois'] = pd.to_datetime(dju['mois'])
```

Échantillon de travail

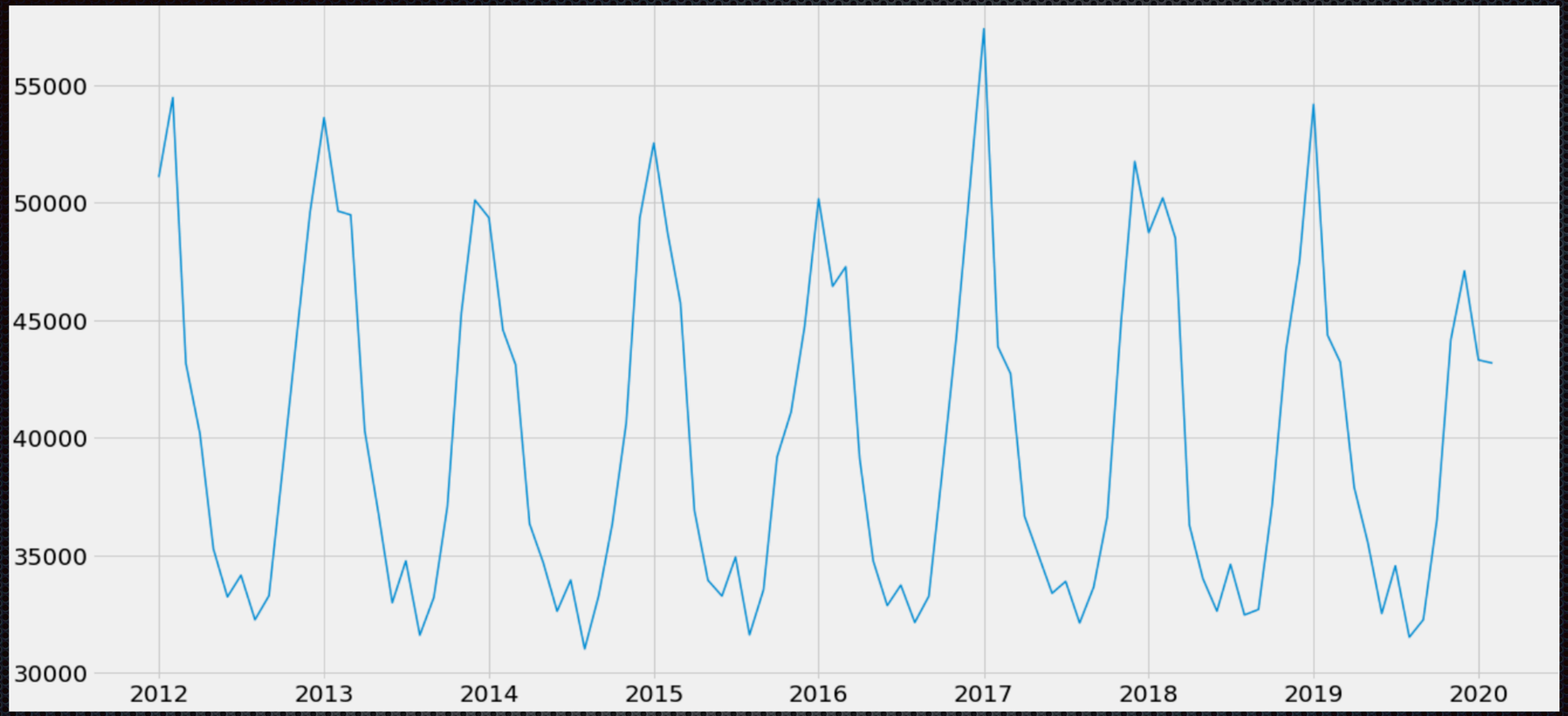
A partir des deux précédentes sources de données, un échantillon a été constitué.

- Années : 2012 à 2018
- Nombre d’observations : 84
- Aucune valeur aberrante
- Aucune donnée manquante
- Aucun doublon

mois	consommation	dju_chauffage
2012-01-01	51086	336.0
2012-02-01	54476	435.9
2012-03-01	43156	201.9
2012-04-01	40176	230.3
2012-05-01	35257	83.3
...	...	...
2018-08-01	32451	3.3
2018-09-01	32683	34.3
2018-10-01	37102	122.4
2018-11-01	43738	282.5
2018-12-01	47553	325.9

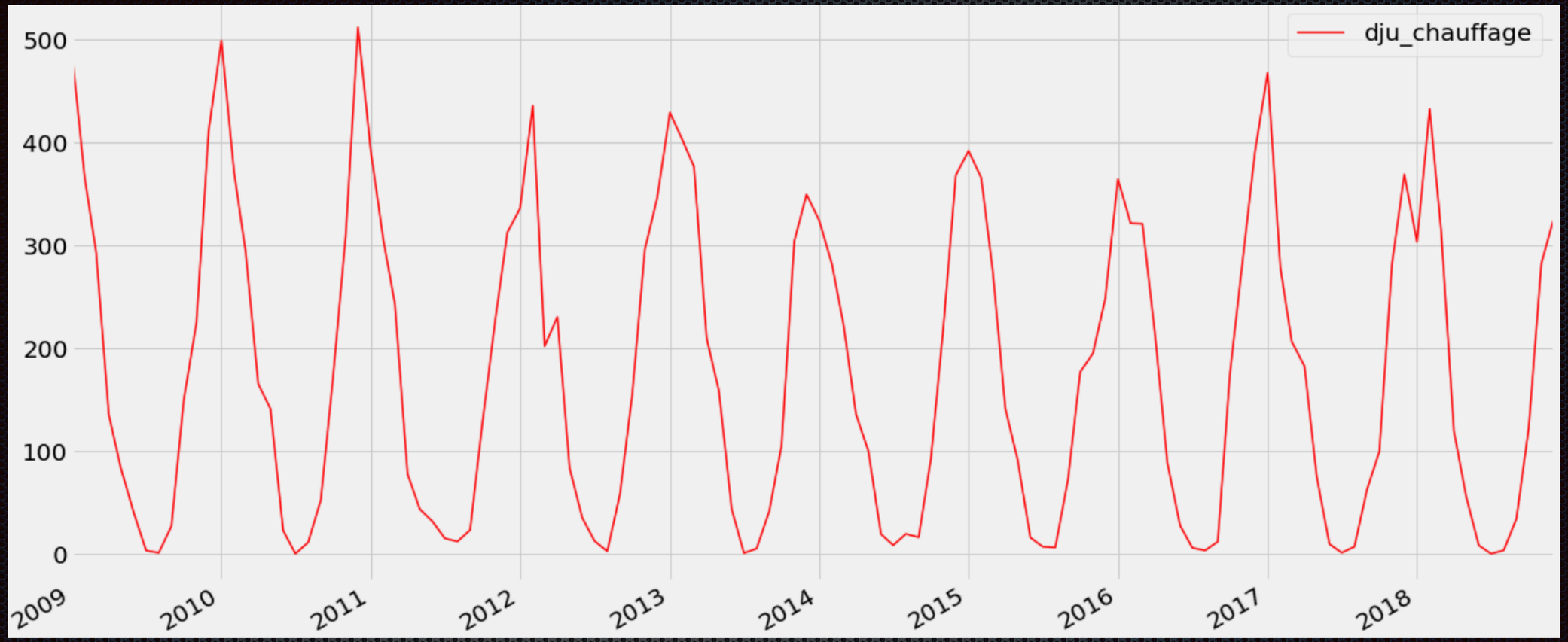
84 rows × 2 columns

Évolution de la consommation d’électricité en France



Cycles réguliers au cours du temps et de même amplitude. Une correction des données de consommation mensuelles de l'effet température (dues au chauffage électrique) sera faite à l’aide de la régression linéaire.

Évolution des Degrés Jour Unifiés chauffage électrique



Série temporelle avec saisonnalité liée aux conditions climatiques. Cycles mensuels fonction de la température, et d'autres facteurs exogènes non pris en compte dans l'étude comme l'humidité, impact de la pollution, etc…

Correction de l’effet température

L'effet température (due au chauffage électrique) sera corrigé par **Régression linéaire**.

Régression linéaire - Statsmodels

La modélisation a été faite à partir de **Statsmodels**, mais également par Sckit-learn à des fins de comparaison.

La qualité du modèle est estimée à 95% (coefficient de détermination R²).

Besoin de vérification de l'hypothèse de normalité…

```python
#Préparation des données pour établir la Régression linéaire
y = df['consommation']
x = sm.add_constant(df['dju_chauffage'])

#Nouvelle Régression linéaire par Statsmodels
#Instanciation du modèle et entrainement
reg = sm.OLS(y, x).fit()
print(reg.summary())
```

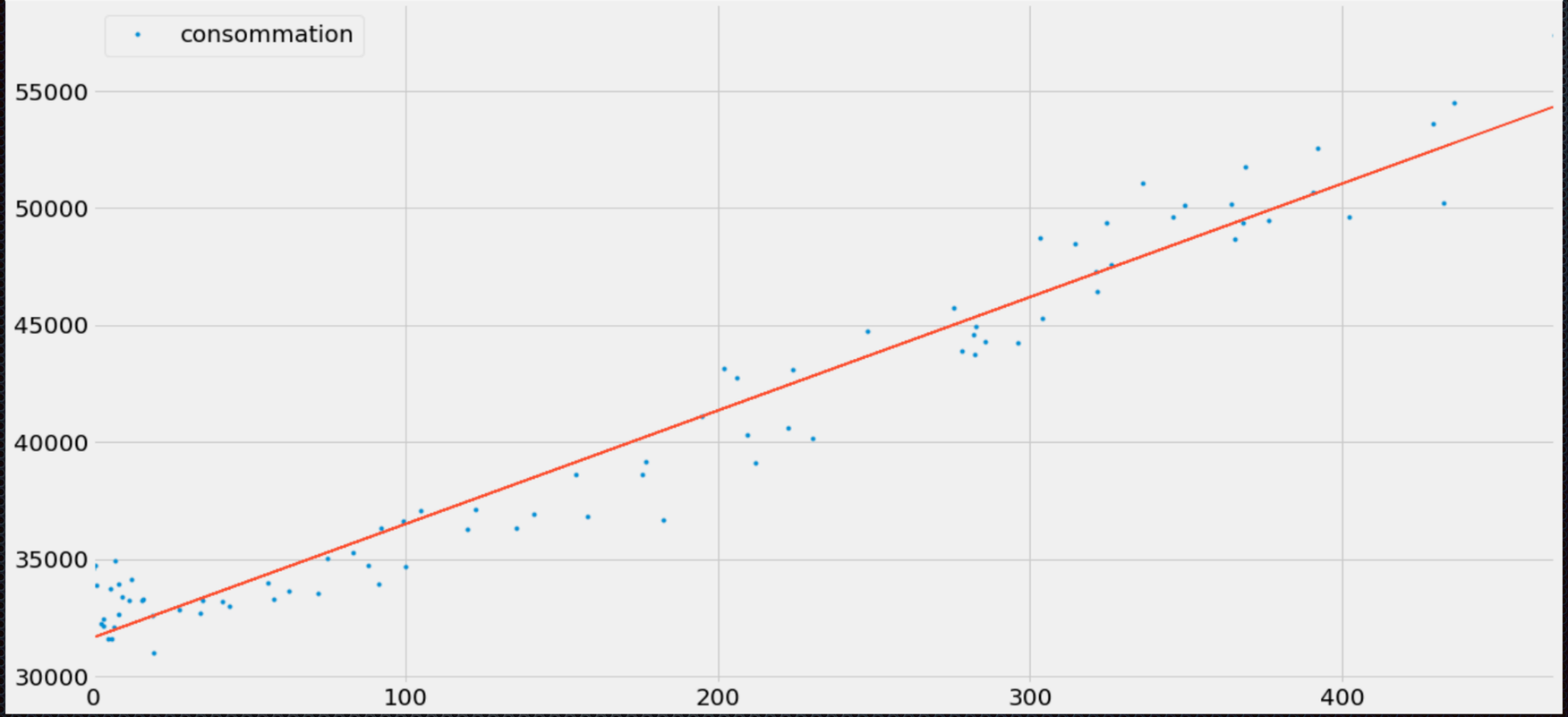
```
                            OLS Regression Results
==============================================================================
Dep. Variable:           consommation   R-squared:                       0.954
Model:                            OLS   Adj. R-squared:                  0.954
Method:                 Least Squares   F-statistic:                     1704.
Date:                Tue, 12 May 2020   Prob (F-statistic):           1.23e-56
Time:                        13:04:51   Log-Likelihood:                -734.60
No. Observations:                  84   AIC:                             1473.
Df Residuals:                      82   BIC:                             1478.
Df Model:                           1
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const           3.164e+04    265.008    119.405      0.000    3.11e+04    3.22e+04
dju_chauffage     48.4919      1.175     41.282      0.000      46.155      50.829
==============================================================================
Omnibus:                        0.871   Durbin-Watson:                   1.921
Prob(Omnibus):                  0.647   Jarque-Bera (JB):                0.922
Skew:                           0.119   Prob(JB):                        0.631
Kurtosis:                       2.546   Cond. No.                         356.
==============================================================================
```

```python
#Coefficient de régression linéaire
reg.params['dju_chauffage']
```

```
48.49191639056929
```

Plot Régression linéaire

```
#Représentation de la Régression linéaire obtenue
df.plot(x='dju_chauffage', y='consommation', style='.')
plt.plot(df['dju_chauffage'], reg.predict(sm.add_constant(df['dju_chauffage'])))

plt.savefig("p9_03_graphic/regression_lineaire_conso_dju.png")
plt.show()
```

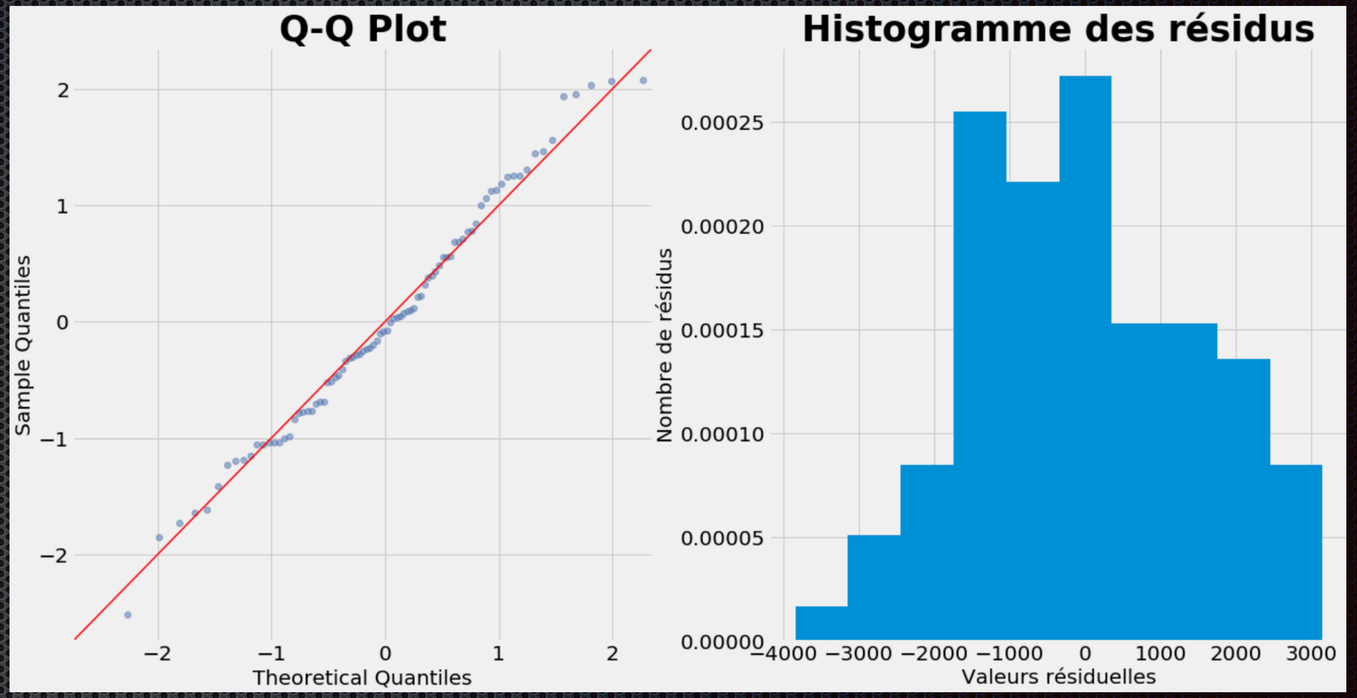


Le modèle de régression linéaire illustre la relation dont nous parlons.
Visuellement on peut distinguer la corrélation entre nos deux variables continues.

Test statistique du modèle

Aux vues de ces graphes, il ne semble pas illusoire de penser que les résidus issus de la modélisation sont gaussiens.

La précédente intuition est validée par la statistique et la p-value du test de Shapiro-Wilk.



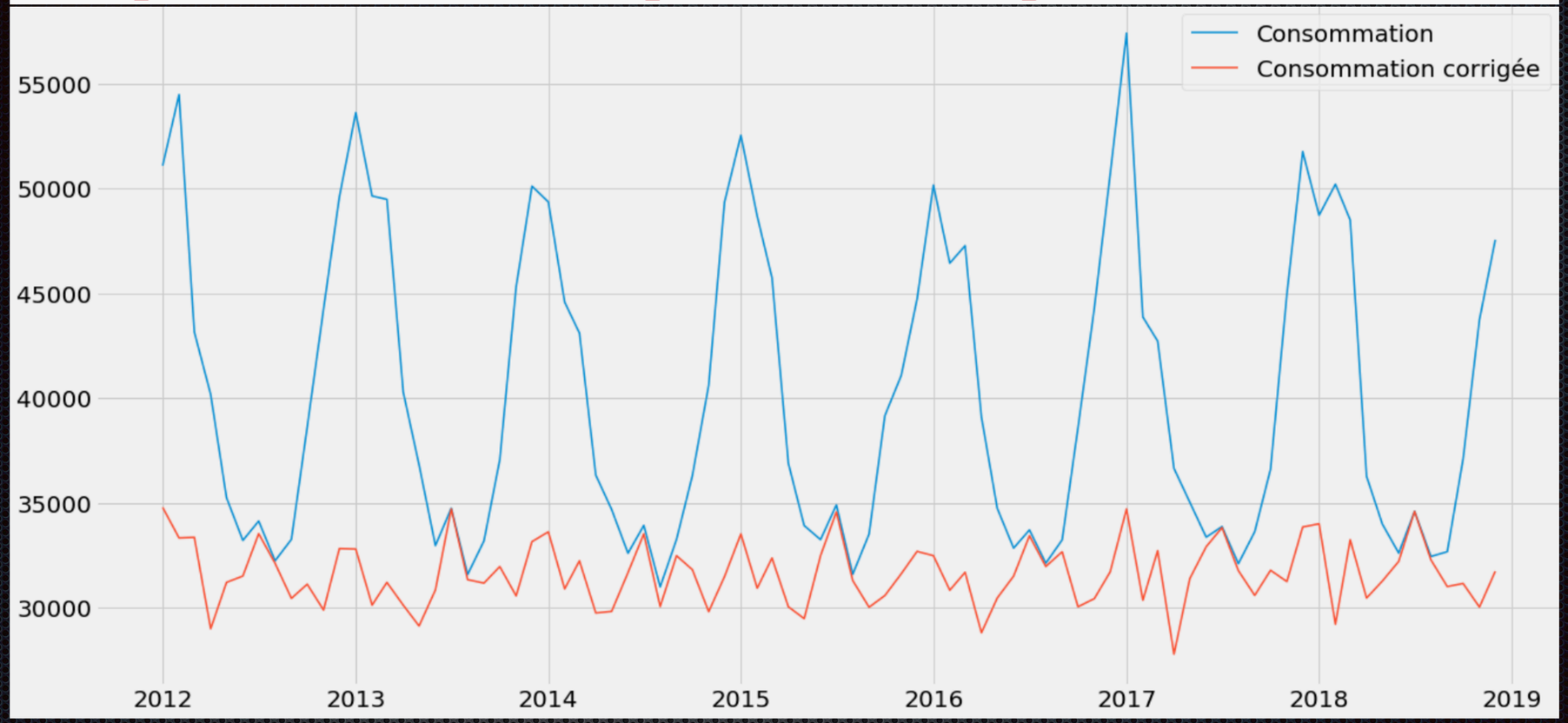
On ne rejette pas l'hypothèse de normalité des résidus.

```
#Test de Shapiro-Wilk
shapiro(reg.resid)
```

```
(0.9864242076873779, 0.5266931056976318)
```

Consommation en électricité

```
#Calcul de la correction : l'effet DJU est retranché à la consommation totale
df['conso_corr'] = df['consommation'] - df['dju_chauffage']*reg.params['dju_chauffage']
```



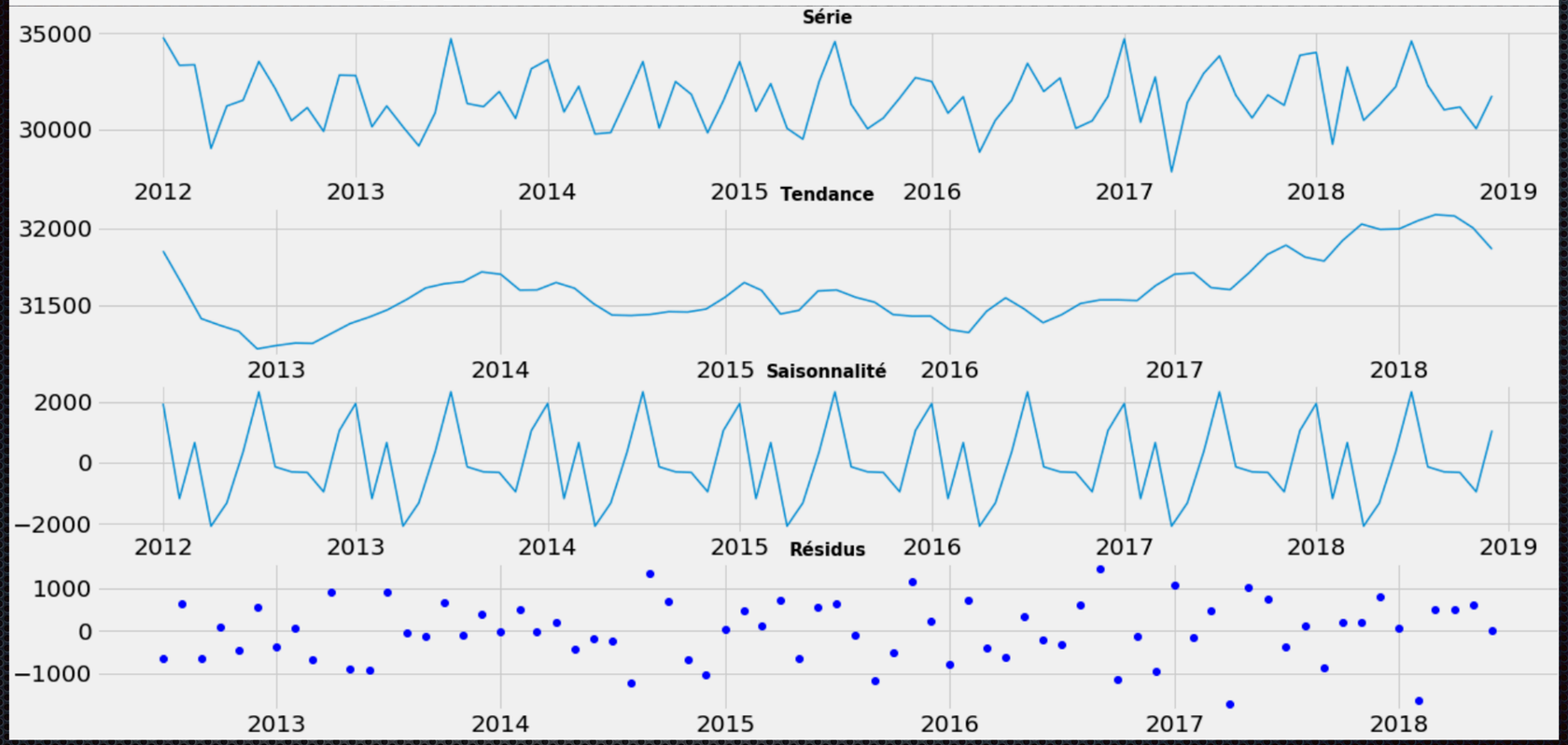
La série temporelle de la consommation en électricité corrigée de l'effet température (dues au chauffage électrique) nous servira pour la désaisonnalisation et pour les futures prévisions.

Désaisonnalisation de la consommation corrigée

Les moyennes mobiles seront utilisées pour transformer le signal temporel. Tendance invariante, absorber la saisonnalité et réduire le résidu.

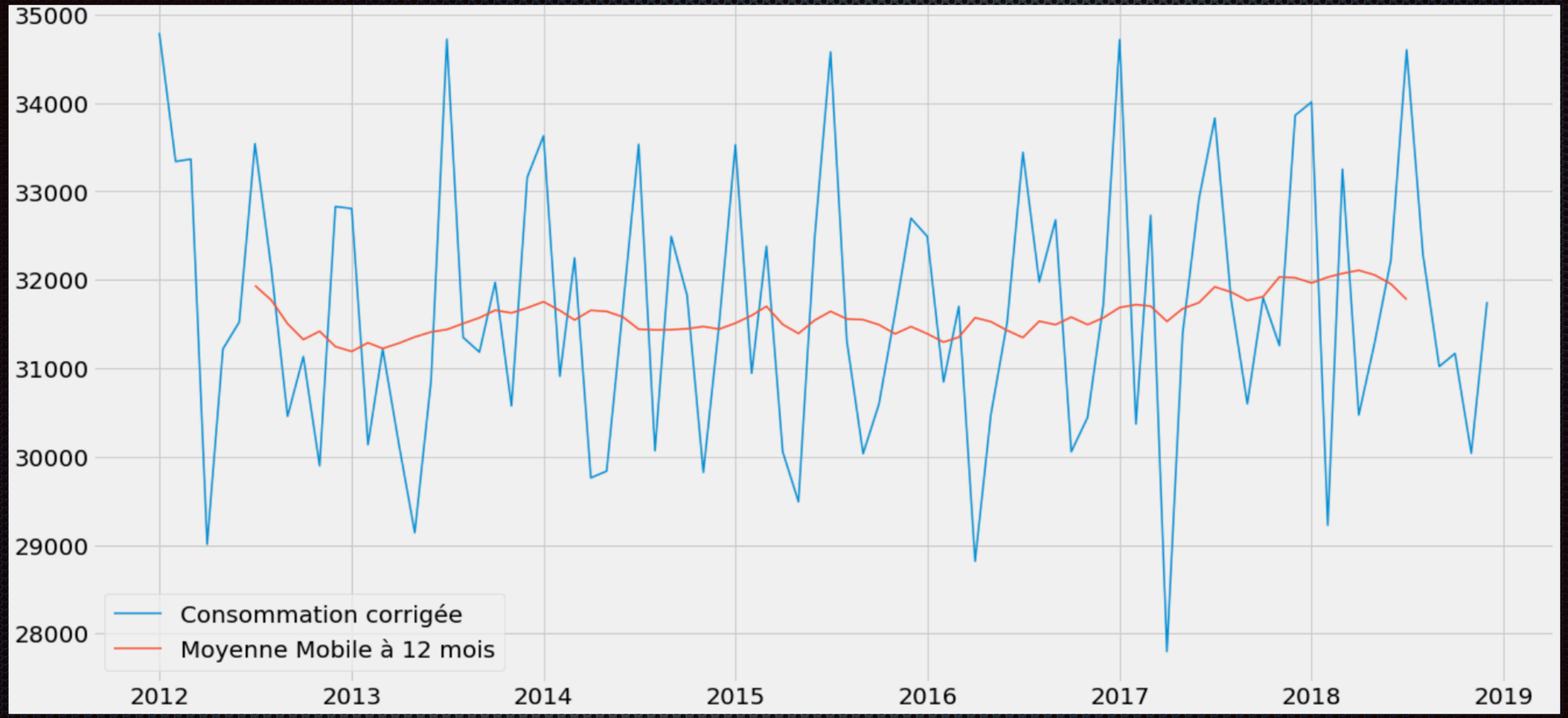
Décomposition de la série temporelle

```python
from statsmodels.tsa.seasonal import seasonal_decompose
import matplotlib.pyplot as plt

decomposition = seasonal_decompose(y,  model='additive')
```



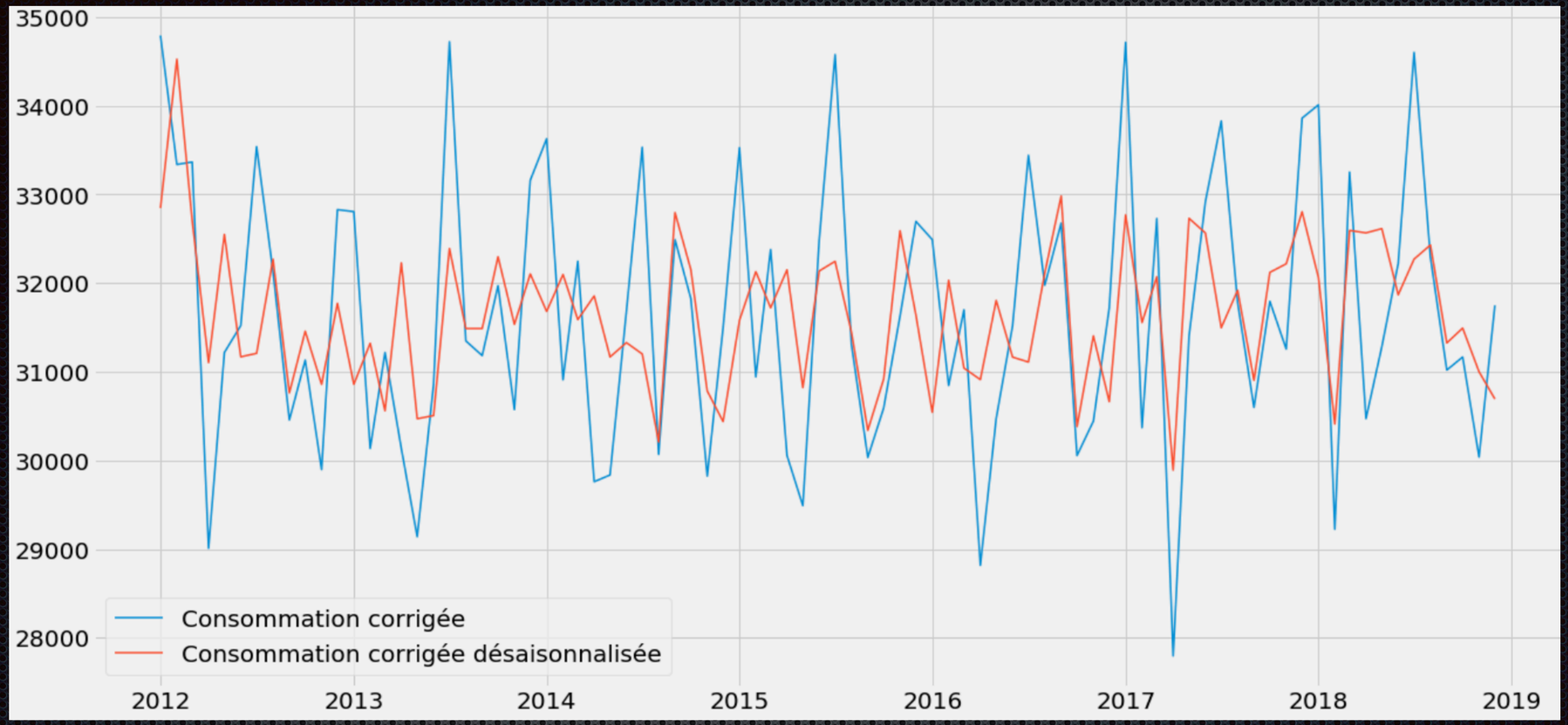
La méthode *seasonal_decompose* permet la décomposition de la série, **la saisonnalité peut alors être retranchée**.

Moyenne mobile 12 mois



Pré-représentation graphique de la série temporelle par sa Moyenne mobile 12 mois. **Quelle représentation de la consommation électrique hors-saisonnalité ?**

Consommation désaisonnalisée



La série temporelle désaisonnalisée reste stable suivant un modèle additif. Des prévisions pourront être réalisées par des techniques économétriques traditionnelles.

L’ électricité ne se stockant pas, la production globale sur le réseau électrique français doit, à tout moment, être strictement égale à la consommation.

Toute modification de la demande ou de la production d’électricité en un point du réseau se répercute instantanément sur tout le système électrique.

Celui-ci doit donc s’adapter en permanence pour satisfaire l’équilibre offre-demande.

Prévision de la consommation

La méthode de Holt-Winters (lissage exponentiel) permet de prendre en compte tendance et saisonnalité.

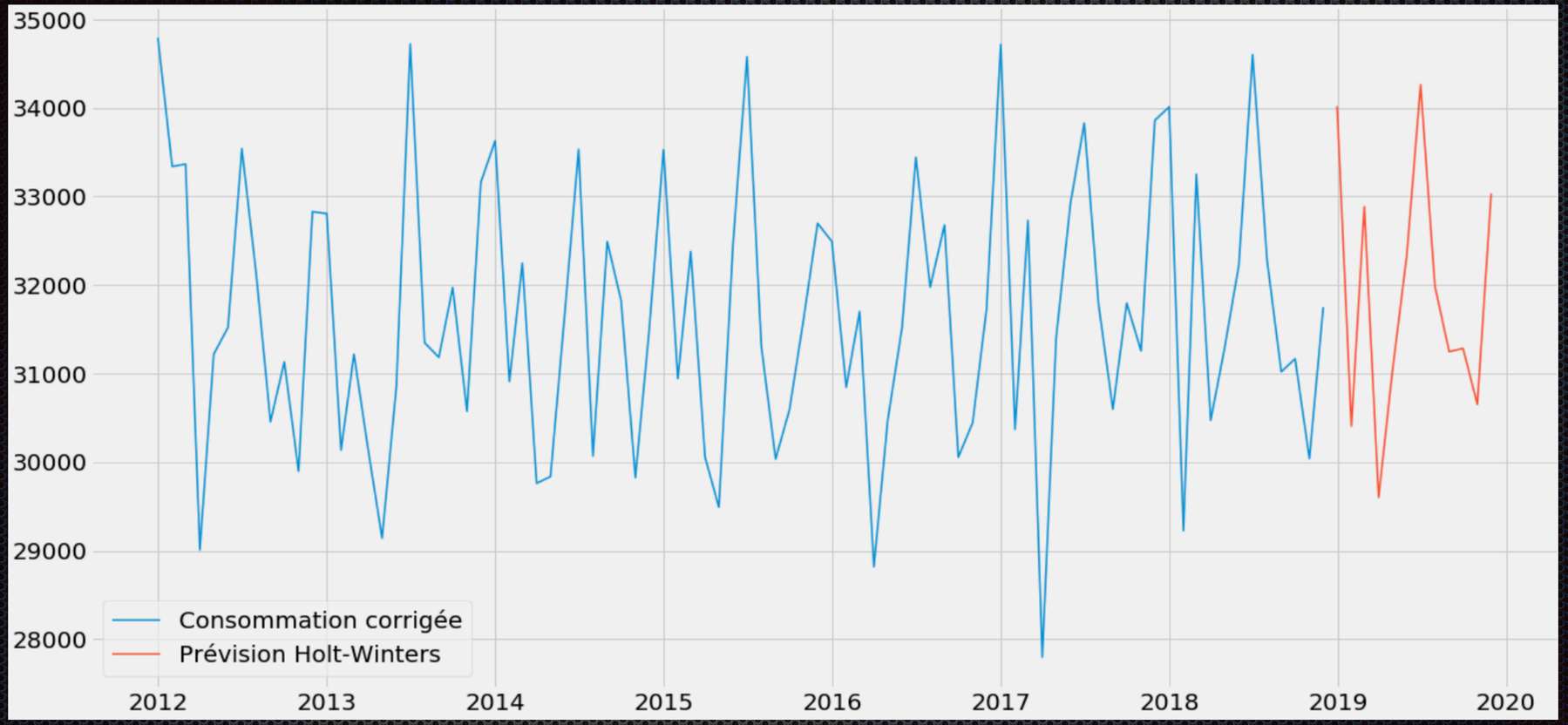
Modèle de prévision Holt-Winters

```
from statsmodels.tsa.api import ExponentialSmoothing

y = np.asarray(df["conso_corr"])
hw = ExponentialSmoothing(y, seasonal_periods=12, trend='add', seasonal='add').fit()
hw_pred = hw.forecast(12)

plt.plot(df["conso_corr"], label='Consommation corrigée')
plt.plot(pd.date_range(df.index[len(y)-1], periods=12, freq='M'), hw_pred, label='Prévision Holt-Winters')

plt.title("Consommation d'électricité en France - Prévision 2019")

plt.savefig("p9_03_graphic/prevision2019_holt-winters.png")
plt.legend()
plt.show()
```

Le lissage exponentiel simple ou double ne tient pas compte de la saisonnalité. La méthode Holt-Winters conserve la saisonnalité, largement retenu dans notre contexte.

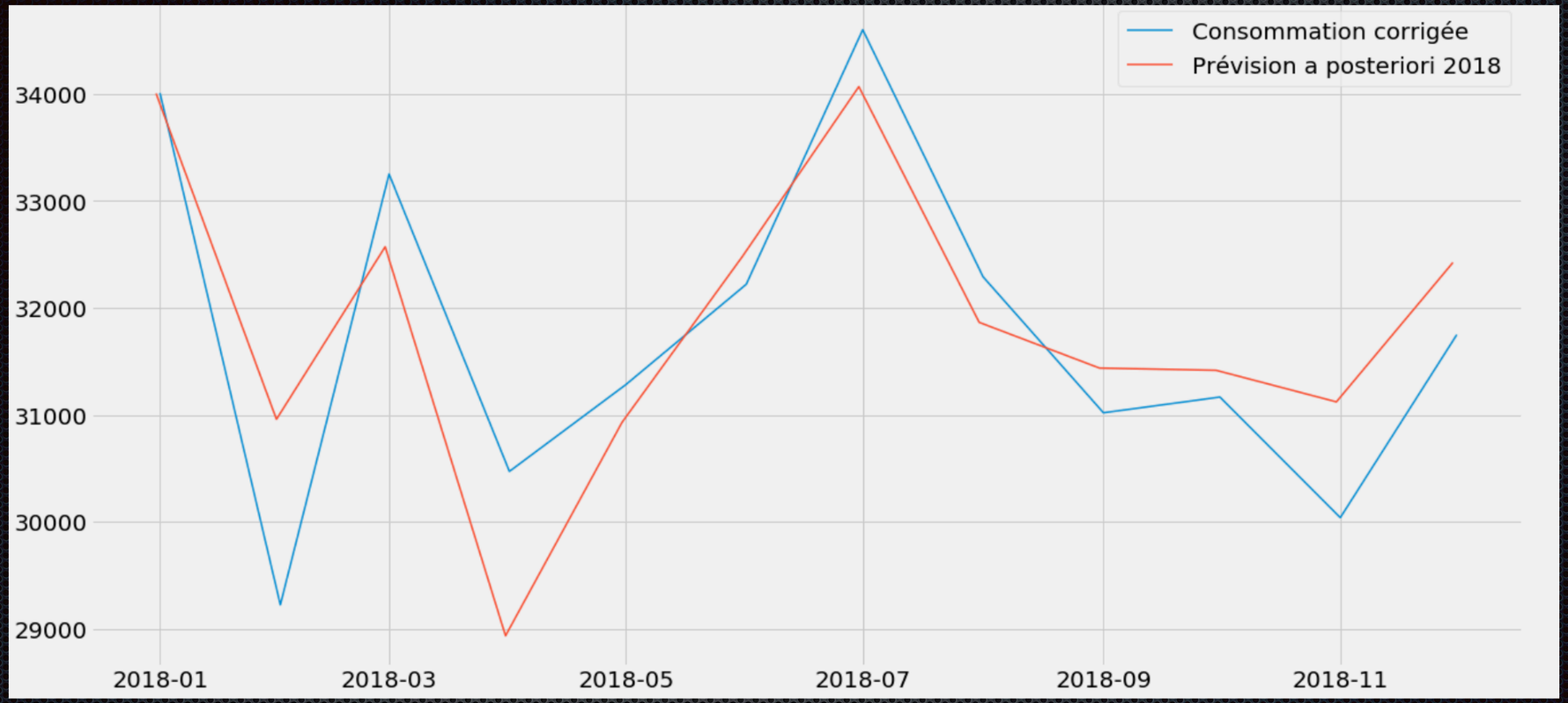
Que donne cette prévision?

Prévision 2019 Holt-Winters



La forte saisonnalité de la série est potentiellement prévisible par Holt-Winters. Le lissage exponentiel simple ou double, n’aurait fourni qu’une constante et qu’une droite. **A posteriori que donne cette prévision?**

Prévision a posteriori 12 mois HW

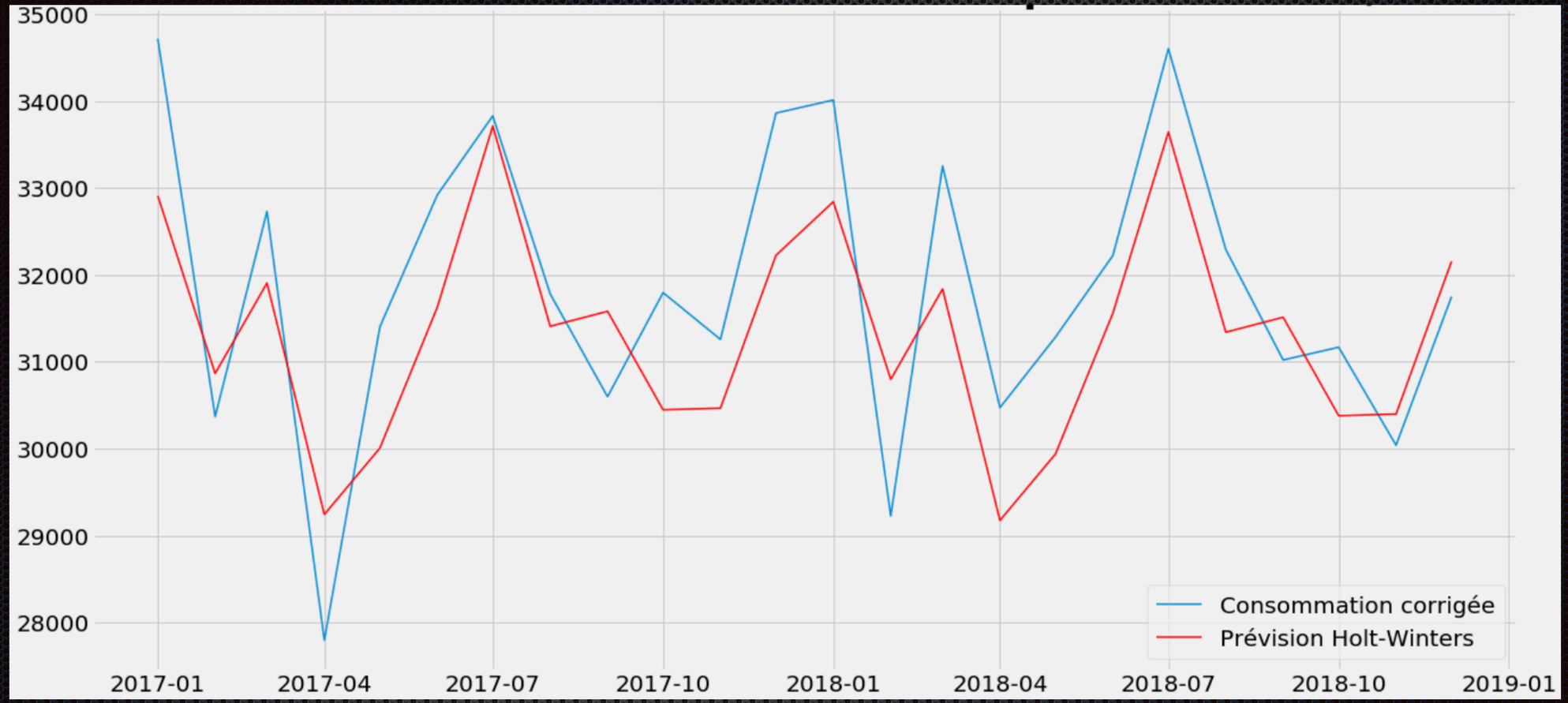



```
#Calcul du MAPE
mape = round((np.abs(1-hw_pred/x_prevision["conso_corr"])).mean()*100, 2)
mape
```

```
2.14
```

La prévision ressort intéressante avec un MAPE (Mean Absolute Percentage) de 2%. **Une prévision a posteriori sur 24 mois serait-elle toujours performante?**

Prévision a posteriori 24 mois HW



MAPE :3.14

Sur du long terme le modèle reste robuste et performant.

Qu'elle serait la prévision avec la méthode SARIMA?

Prévision de la consommation

La méthode SARIMA permet de modéliser notre série qui présente une forte saisonnalité.

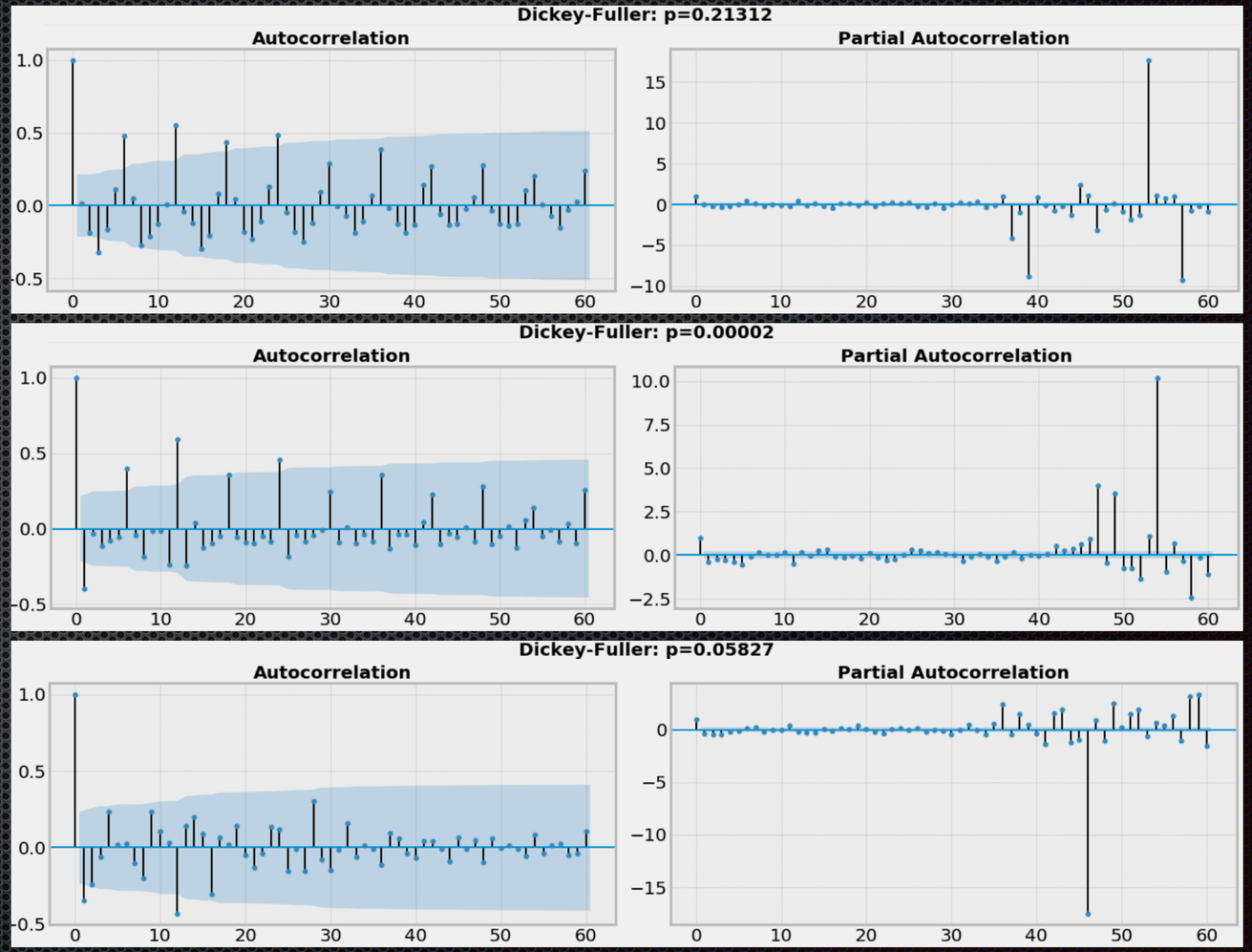
Stationnarisation par différenciation

L'autocorrélogramme simple par sa décroissance lente vers 0 valide un problème de non-stationnarité.

La sortie ACF après une première différenciation présente toujours une décroissance pour les multiples de 12.

Une différenciation d'ordre 12 retourne des résultats satisfaisants.

Test de Dickey-Fuller rejette l'hypothèse nulle de non-stationnarité au seuil de 5%.



Identification des modèles

L'identification des combinaisons est faite par minimisation du critère d'information d'Akaike AIC, pour déterminer les ordres optimaux.

Mais pas seulement, la fonction donne également la possibilité d'optimiser les modèles avec un test de Ljung–Box pour la blancheur du résidu et pvalues significatives.

```python
#Détermination du meilleur modèle de prévision
DUREE_PREDITE = 12 #prédiction sur 12 mois
y = df['conso_corr']

#Calcul de toutes les combinaisons de p, d et q
pdq_list = list(itertools.product(range(3),
                                  range(2),
                                  range(3)
          ))

#Calcul de toutes les combinaisons de P, D et Q
PDQ_list = [ (*val, 12) for val in list(itertools.product(
                                range(3),
                                range(2),
                                range(3))
           )]


#Nombre de configurations total pour un modèle SARIMA
n = len(pdq_list) * len(PDQ_list)

#Dataframe pour stockage des valeurs
results_SARIMA = pd.DataFrame({'modele':[], 'aic':[], 'bic':[], 'ljungbox':[], 'pvalues':[]})
counter = 1
s_time = time()

#Itération sur toutes les combinaisons
for pdq in pdq_list:
    for PDQ in PDQ_list:
        try:
            #Instanciation et entrainement du modèle SARIMA
            mod = SARIMAX(np.asarray(y), order = pdq, seasonal_order = PDQ,
                          enforce_stationarity=False, enforce_invertibility=False)
            result = mod.fit()

            #Intégration des indicateurs dans le précédent dataframe
            results_SARIMA = results_SARIMA.append({'modele':f"{pdq}x{PDQ}",
                                'aic': result.aic,
                                'bic': result.bic,
                                'ljungbox': acorr_ljungbox(result.resid, lags=12)[1].mean(),
                                'pvalues': result.pvalues.min()},
                                ignore_index=True)
        except:
            continue

        clear_output(' ')
        print(f"{counter}/{n} - {time() - s_time:.0f}s")
        counter += 1
```

Estimation du modèle

Sur les modèles les plus performants selon le critère Akaike AIC (et BIC), **le modèle (1, 1, 1)x(2, 1, 1, 12)** ressort comme étant le plus optimal **sur les 324 modèles proposés**.

Le résidu est bien un bruit blanc, vérification établie à l'aide du test de Ljung–Box.

```
#Sélection des modèles SARIMA les plus optimaux selon les tests de significativité des paramètres
#et de blancheur du résidu sont validés au niveau 5%
results_SARIMA_filter = results_SARIMA[(results_SARIMA['ljungbox'] > 0.05)]
results_SARIMA_filter = results_SARIMA_filter[results_SARIMA_filter['pvalues'] < 0.05]
```

	modele	aic	bic	ljungbox	pvalues	rank_aic	rank_bic
95	(0, 1, 2)x(0, 1, 2, 12)	738.406	747.326	0.908576	8.30316e-08	1	1
101	(0, 1, 2)x(1, 1, 2, 12)	739.429	750.134	0.940724	6.27032e-07	2	2
203	(1, 1, 2)x(0, 1, 2, 12)	739.864	750.57	0.880271	2.89554e-07	3	3
107	(0, 1, 2)x(2, 1, 2, 12)	741.166	753.655	0.943071	8.33273e-15	4	4
209	(1, 1, 2)x(1, 1, 2, 12)	741.212	753.702	0.90234	0.00180701	5	5
311	(2, 1, 2)x(0, 1, 2, 12)	741.536	754.025	0.916196	2.1131e-06	6	6
215	(1, 1, 2)x(2, 1, 2, 12)	743.026	757.3	0.919117	0.00205488	7	7
317	(2, 1, 2)x(1, 1, 2, 12)	744.456	758.73	0.0607745	1.33613e-05	8	8
323	(2, 1, 2)x(2, 1, 2, 12)	748.675	764.732	0.91642	0.0245875	9	11
83	(0, 1, 1)x(1, 1, 2, 12)	755.117	764.15	0.962951	0.016185	10	10
77	(0, 1, 1)x(0, 1, 2, 12)	755.436	762.663	0.916439	6.38855e-08	11	9
191	(1, 1, 1)x(1, 1, 2, 12)	755.729	766.569	0.854975	1.61005e-30	12	13
304	(2, 1, 1)x(2, 1, 1, 12)	755.979	768.626	0.782397	6.76281e-26	13	17
299	(2, 1, 1)x(1, 1, 2, 12)	755.989	768.635	0.772647	7.93003e-29	14	18
89	(0, 1, 1)x(2, 1, 2, 12)	756.649	767.489	0.963819	0	15	14
185	(1, 1, 1)x(0, 1, 2, 12)	756.943	765.976	0.853698	2.71175e-19	16	12
286	(2, 1, 0)x(2, 1, 1, 12)	766.154	776.994	0.168032	6.10613e-14	23	23
287	(2, 1, 0)x(2, 1, 2, 12)	768.766	781.413	0.227699	6.44333e-07	24	24
196	(1, 1, 1)x(2, 1, 1, 12)	770.994	781.966	0.879901	0	25	25
195	(1, 1, 1)x(2, 1, 0, 12)	773.066	782.209	0.898275	0	26	26
213	(1, 1, 2)x(2, 1, 0, 12)	773.454	784.426	0.48952	0	27	27
214	(1, 1, 2)x(2, 1, 1, 12)	773.565	786.365	0.933777	0.0020434	28	28
281	(2, 1, 0)x(1, 1, 2, 12)	783.519	794.491	0.191207	3.22234e-06	29	29
275	(2, 1, 0)x(0, 1, 2, 12)	785.573	794.717	0.243175	3.11585e-07	30	30

Choix du modèle

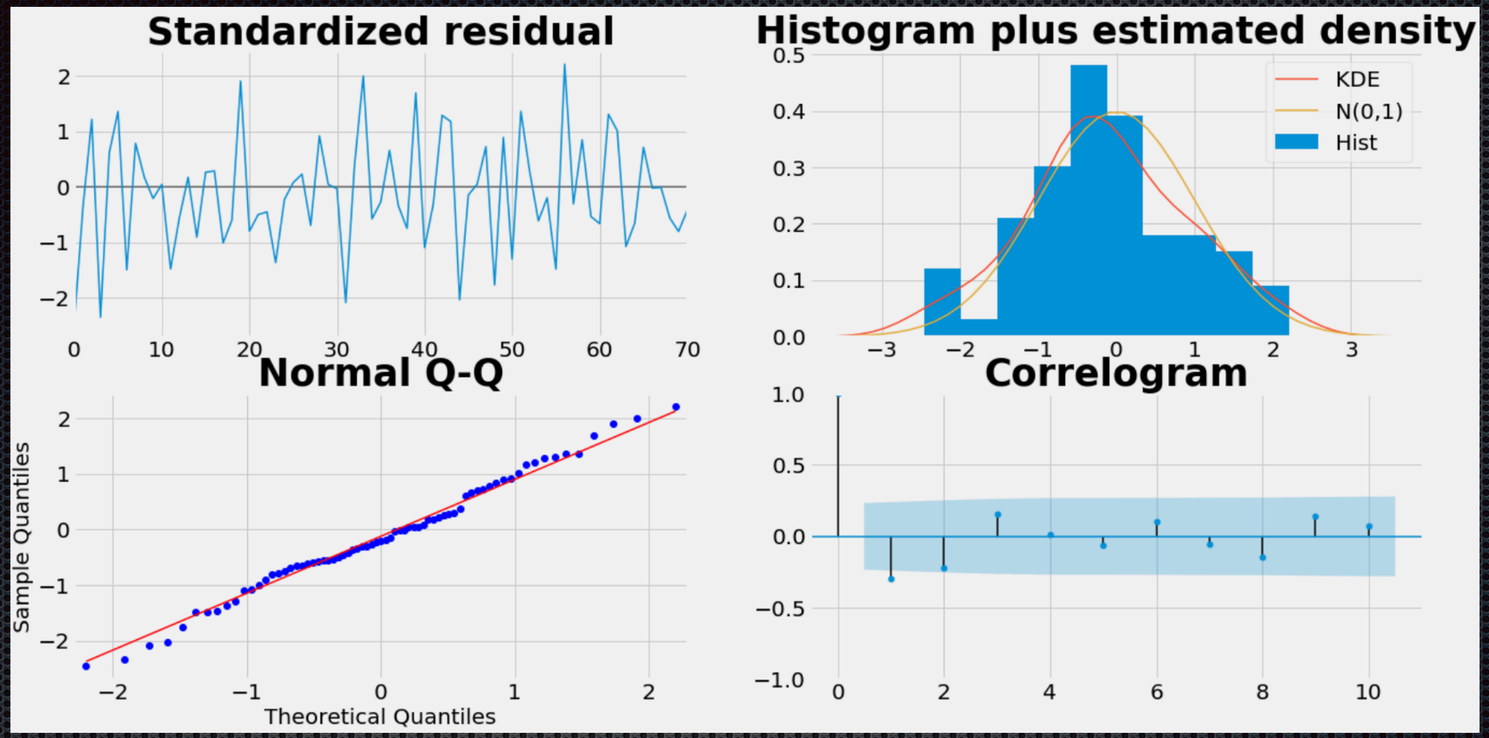
Le modèle choisi paraît pertinent, les paramètres sont significatifs, la blancheur du résidu est validé au niveau 5%.

Un test de normalité des résidus peut-être judicieux afin de vérifier la distribution gaussienne.

```python
#Instanciation et entrainement du modèle SARIMA via statsmodels
model = SARIMAX(np.asarray(y), order=(1,1,1), seasonal_order=(2,1,1,12))
results = model.fit()
print(results.summary())
```

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                                     y   No. Observations:                   84
Model:             SARIMAX(1, 1, 1)x(2, 1, 1, 12)   Log Likelihood                -615.377
Date:                            Wed, 20 May 2020   AIC                           1242.754
Time:                                    12:21:24   BIC                           1256.331
Sample:                                         0   HQIC                          1248.153
                                             - 84
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.5840      0.155      3.760      0.000       0.280       0.888
ma.L1         -0.7093      0.149     -4.762      0.000      -1.001      -0.417
ar.S.L12      -1.1129      0.042    -26.731      0.000      -1.194      -1.031
ar.S.L24      -0.2502      0.066     -3.792      0.000      -0.380      -0.121
ma.S.L12       1.0000      0.195      5.131      0.000       0.618       1.382
sigma2      1.701e+06   1.15e-07   1.48e+13      0.000     1.7e+06     1.7e+06
===================================================================================
Ljung-Box (Q):                       61.02   Jarque-Bera (JB):                 0.03
Prob(Q):                              0.02   Prob(JB):                         0.98
Heteroskedasticity (H):               0.77   Skew:                            -0.02
Prob(H) (two-sided):                  0.53   Kurtosis:                         2.91
===================================================================================
```

Normalité du résidu



La représentation "Standardized residual" et "Correlogram" confirment qu'il n'y a pas de corrélation des résidus. Les résidus sont normalement distribués KDE vs distribution normale - N (0,1). La distribution ordonnée des résidus représentée par le du QQ plot est globalement satisfaisante. A noter des écarts en queue de distribution.

Test de normalité du résidu

L'hypothèse de normalité est remise en cause par Shapiro (p-value < 0.05).

L'hypothèse de normalité n'est pas remise en cause par Jarque-Bera.

Le test de Jarque-Bera complète l'intuition des petites divergences vers les queues de distribution du Q-Q plot.

L'échantillon étant de taille suffisante, les résultats obtenus par le modèle ne sont pas absurdes.

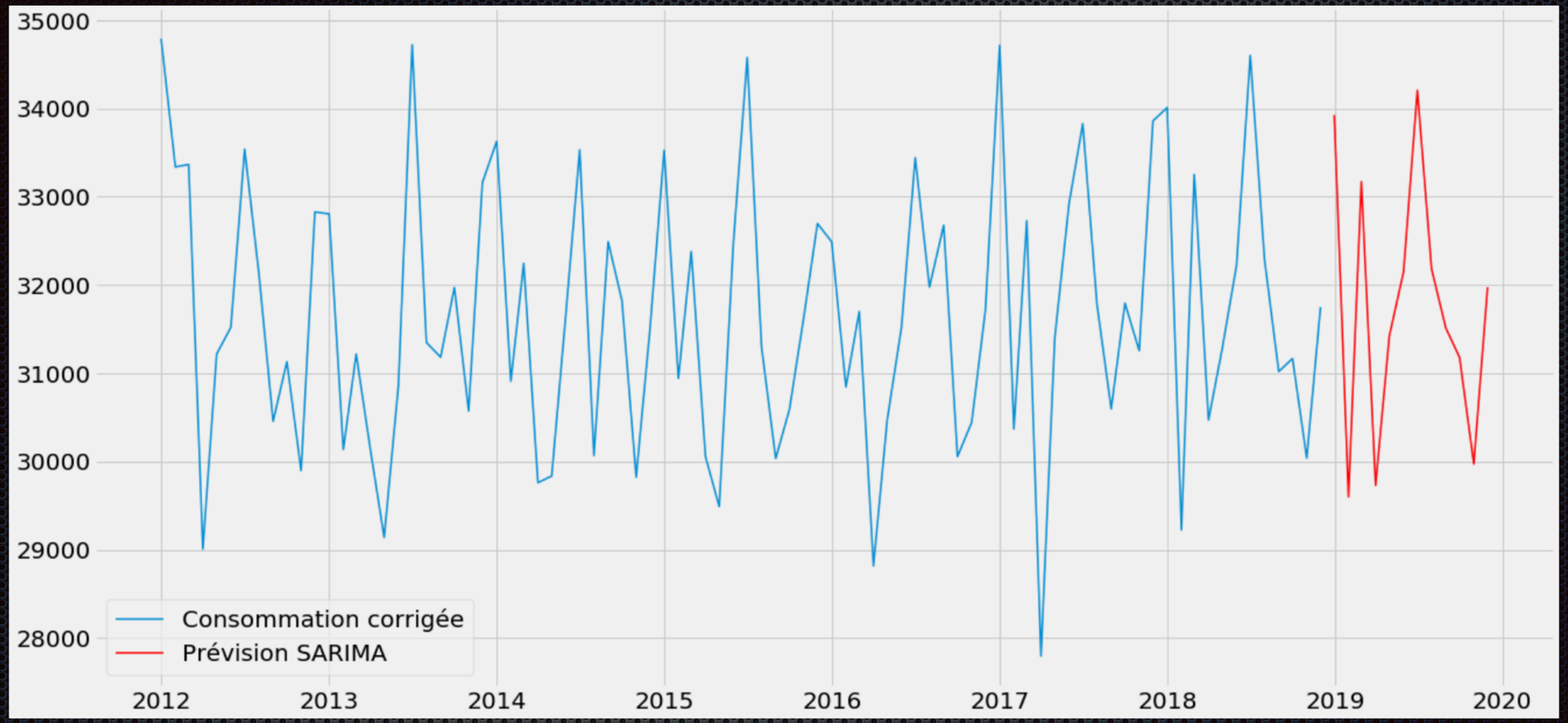
```
#Test de Shapiro-Wilk
from scipy.stats import shapiro
shapiro(results.resid)
```

```
2.3092423659955535e-16
```

```
#Test de Jarque-Bera
results.test_normality("jarquebera")[0][1]
```

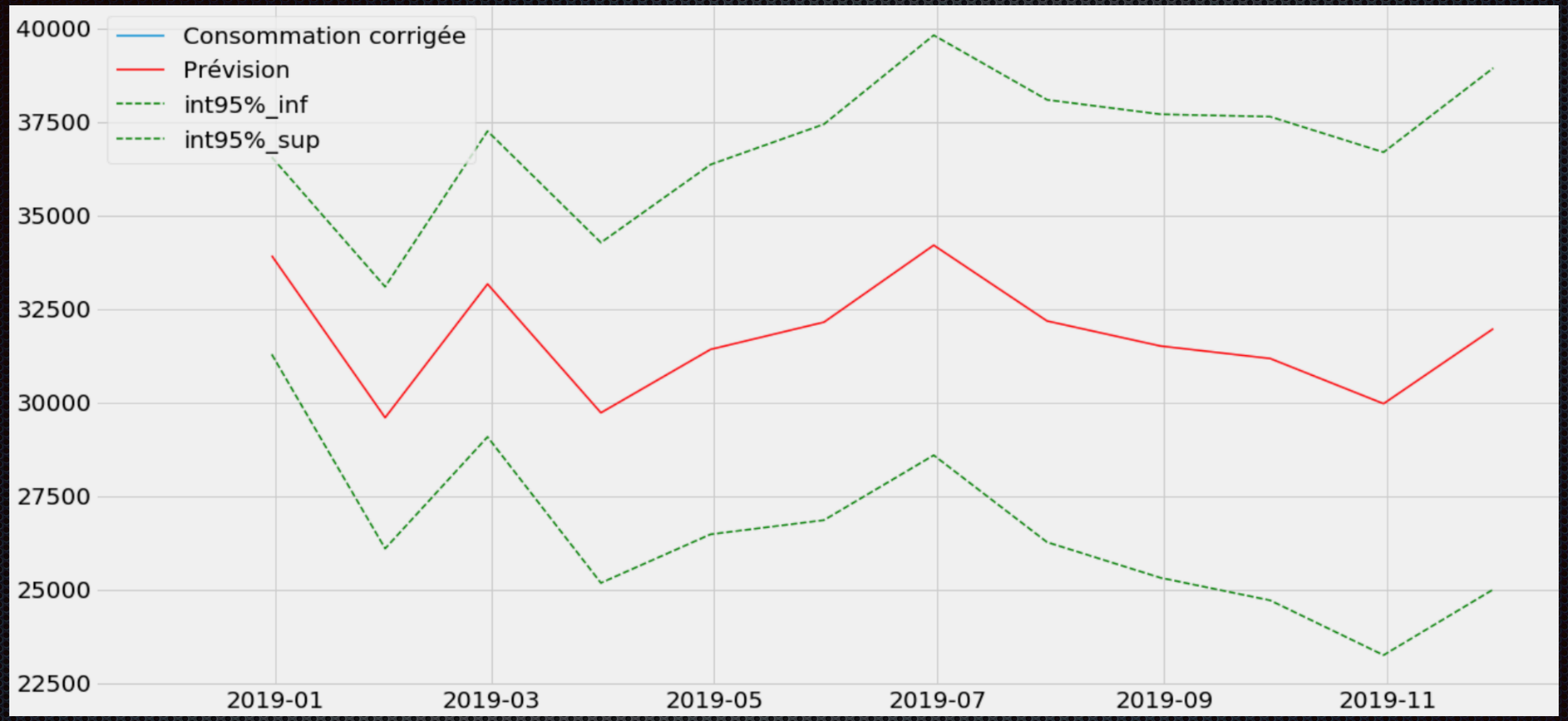
```
0.932255211653555
```

Prévision de la consommation 2019



Scénario plausible à l'horizon d'une prévision à 12 mois. Les extrêmes pics et creux semblent mieux être pris en compte que précédemment dans Holt-Winters.

Intervalles de confiance prévision



La normalité des erreurs est importante pour la fiabilité de nos intervalles de confiance. Notons un niveau d'incertitude plus marqué sur du long terme.

A posteriori, quel écart entre les prévisions et les réalisations?

Quelle qualité prédictive ?

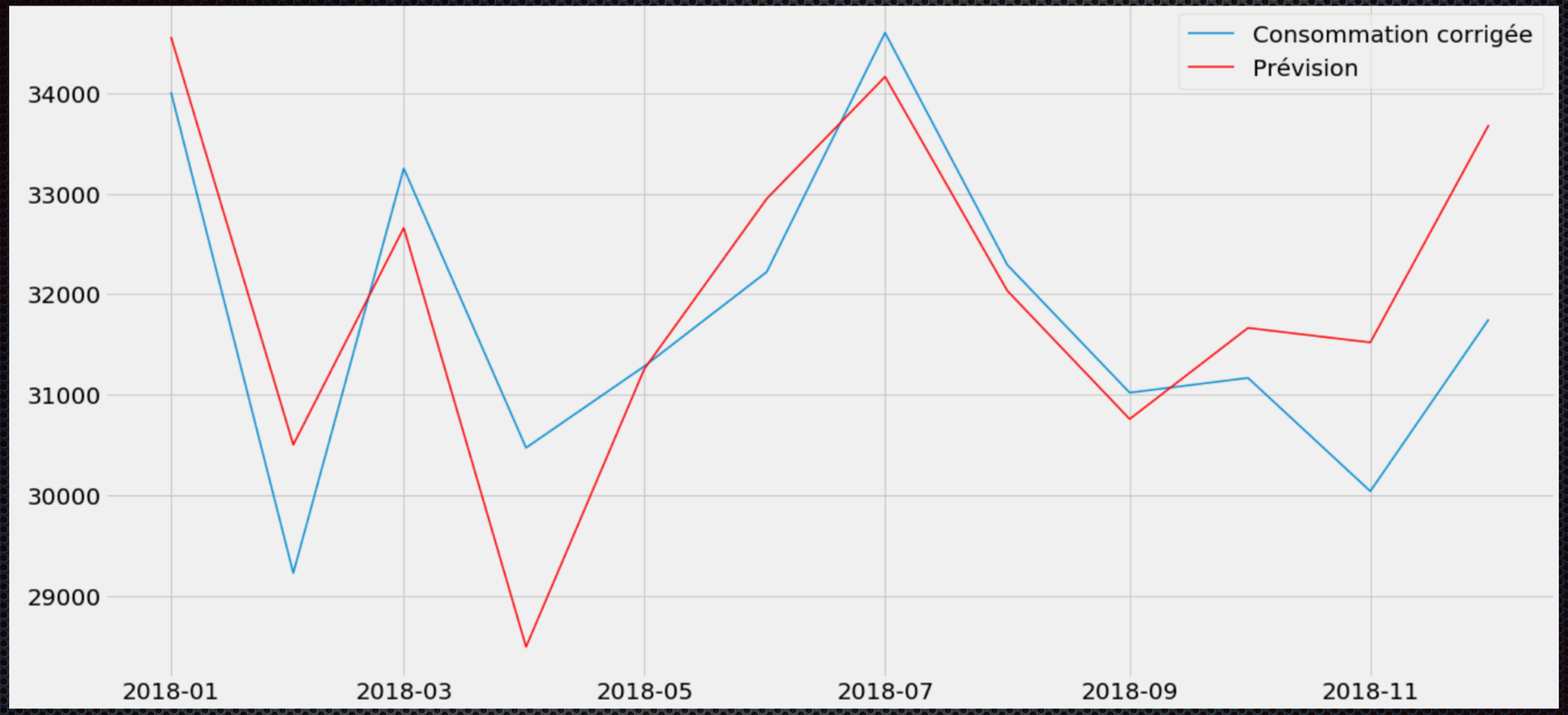
Une étude a posteriori permet d'estimer la qualité prédictive du modèle proposé.

Le modèle est toujours validé par la significativité des valeurs, ainsi que la blancheur du résidu testé par Ljung-Box niveau 5%.

```
model_tronc = SARIMAX(np.asarray(df_tronc['conso_corr']), order=(1,1,1), seasonal_order=(2,1,1,12))
results_tronc = model_tronc.fit()
print(results_tronc.summary())

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

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                                   y   No. Observations:                   72
Model:             SARIMAX(1, 1, 1)x(2, 1, 1, 12)   Log Likelihood                -514.263
Date:                            Wed, 20 May 2020   AIC                           1040.526
Time:                                    14:28:41   BIC                           1052.991
Sample:                                         0   HQIC                          1045.392
                                             - 72
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.5952      0.189      3.142      0.002       0.224       0.966
ma.L1         -0.7049      0.181     -3.895      0.000      -1.060      -0.350
ar.S.L12      -1.1023      0.049    -22.309      0.000      -1.199      -1.005
ar.S.L24      -0.2225      0.084     -2.640      0.008      -0.388      -0.057
ma.S.L12       1.0000      0.222      4.509      0.000       0.565       1.435
sigma2      1.922e+06   1.15e-07   1.67e+13      0.000    1.92e+06    1.92e+06
===================================================================================
Ljung-Box (Q):                       66.71   Jarque-Bera (JB):                 0.14
Prob(Q):                              0.01   Prob(JB):                         0.93
Heteroskedasticity (H):               1.05   Skew:                            -0.05
Prob(H) (two-sided):                  0.91   Kurtosis:                         2.78
===================================================================================
Retard : p-value
6 : 0.9083765747881715
12 : 0.8733066304636646
18 : 0.6656138154599217
24 : 0.6424416381017937
30 : 0.6826866656285108
36 : 0.7268380487409898
```

Prévision a posteriori 12 mois




```
mape = round((np.abs(1-pred_tronc/x_prevision["conso_corr"])).mean()*100, 2)
mape
```

```
2.68
```

Le prévision a posteriori permet de valider avec appui du critère d'erreur que les écarts entre les prévisions et les réalisations sont acceptables.

La prévision à court terme reste un facteur essentiel pour assurer l'équilibre entre la production et la consommation durant la journée.

Notons que les intervalles de confiance basés sur la normalité des erreurs ont un niveau d'incertitude plus élevé sur du long terme.

Dans la réalité, l'estimation ponctuelle reste non biaisée, les modèles SARIMA se révèlent être les plus performants sur ces prévisions.

Ces modèles n'intègrent pas de variables exogènes, dans un contexte métier il est donc nécessaire de trouver un biais pour lier des variables climatiques et des signaux de prix.