enercoop
L'énergie militante

Prévoir la demande en électricité

Jérôme Chalain – juillet 2020

Plan

1. Contexte

2. **Présentation** des données utilisées (sources, pré-traitements, choix, etc.)

3. **Correction** des données de l'effet température via une régression linéaire

4. **Désaisonnalisation** de la consommation obtenue après correction grâce aux moyennes mobiles

5. **Prévision** de la consommation (corrigée de l'effet température) sur un an
- en utilisant la méthode de Holt Winters (lissage exponentiel)
- puis la méthode SARIMA

6. Conclusion

Contexte

Contexte

- Enercoop, société spécialisée dans les énergies renouvelables.

- **Problématique :**
  - La plupart de ces énergies renouvelables est intermittente → difficile de prévoir les capacités de production d'électricité.
  - La demande en électricité des utilisateurs varie au cours du temps, et dépend de paramètres comme la météo (température, luminosité, etc.)

- Il s’agit donc de rechercher la meilleure adéquation possible entre offre et demande ! Pour cela, nous avons un certain nombre d’outils à notre disposition.

Présentation des données

Présentation des données utilisées (source, pré-traitements, etc.)

1. Données mensuelles de consommation totale d'électricité

Source : site de RTE, société publique gérant le réseau de transport d'électricité de France.
De 01/2012 à 03/2020.

```
conso.head()
```

	Mois	Qualité	Territoire	Production totale	Production nucléaire	Production thermique totale	Production thermique charbon	Production thermique fioul	Production thermique gaz	Production hydraulique	...	Production bioénergies	Consommation totale	expor
0	0000-00	Données consolidées	Grand-Est	11346	8643.0	1120	22.0	2	1095	565	...	86	4545	
1	0000-00	Données consolidées	Nouvelle-Aquitaine	5289	4179.0	164	NaN	0	164	419	...	120	4578	
2	0000-00	Données consolidées	Auvergne-Rhône-Alpes	11622	8382.0	334	6.0	11	316	2630	...	104	6834	
3	0000-00	Données consolidées	Bourgogne-Franche-Comté	467	NaN	124	NaN	0	123	89	...	31	2188	

```
conso.shape
```

```
(1156, 22)
```

Présentation des données utilisées (source, pré-traitements, etc.)

1. Données mensuelles de consommation totale d'électricité

```
# Suppression des lignes avec des mois incohérents, des lignes autres que "France" et
# des colonnes autres que "Mois" et "conso totale"
conso = conso.loc[(conso['Mois']!='0000-00') & (conso['Territoire']=='France'), ['Mois', 'Consommation totale']]
```

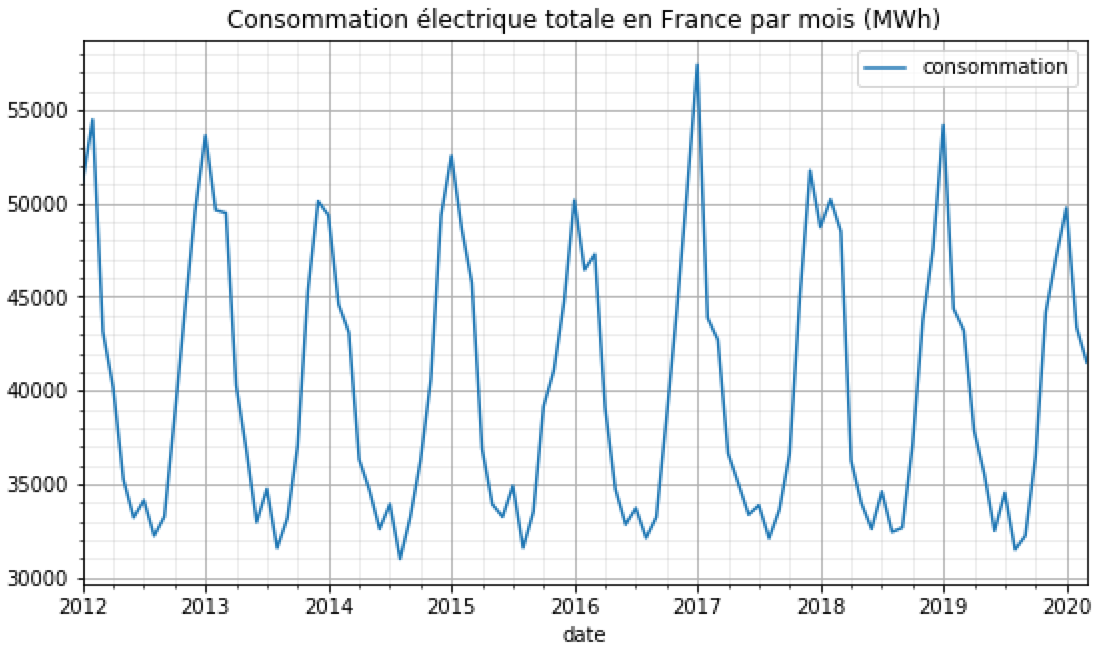
```
conso.head()
```

	Mois	Qualité	Territoire	Production totale	Production nucléaire	Production thermique totale	Production thermique charbon	Production thermique fioul	Production thermique gaz	Production hydraulique	...	Production bioénergies	Consommation totale	expor
0	0000-00	Données consolidées	Grand-Est	11346	8643.0	1120	22.0	2	1095	565	...	86	4545	
1	0000-00	Données consolidées	Nouvelle-Aquitaine	5289	4179.0	164	NaN	0	164	419	...	120	4578	
2	0000-00	Données consolidées	Auvergne-Rhône-Alpes	11622	8382.0	334	6.0	11	316	2630	...	104	6834	
3	0000-00	Données consolidées	Bourgogne-Franche-Comté	467	NaN	124	NaN	0	123	89	...	31	2188	

Présentation des données utilisées (source, pré-traitements, etc.)

1. Données mensuelles de consommation totale d'électricité

	Mois	Consommation totale
0	2012-01	51086
1	2012-02	54476
2	2012-03	43156
3	2012-04	40176
4	2012-05	35257



On observe une saisonnalité de période 12 et une tendance stable

2. Données météo pour corriger les données de conso mensuelle de l'effet température

- Le degré jour unifié (**DJU**) est la somme des écarts entre une température de référence (généralement 18 degrés) et la température moyenne journalière.
- Le DJU pour chaque mois est la somme des DJU quotidiens.

	Année	JAN	FÉV	MAR	AVR	MAI	JUN	JUI	AOÛ	SEP	OCT	NOV	DÉC	Total
0	2019	404.9	268.3	233.1	168.5	117.9	14.0	0.0	0.0	0.0	0.0	0.0	0.0	1206.6
1	2018	303.4	432.6	314.3	119.7	55.9	8.1	0.0	3.3	34.3	122.4	282.5	325.9	2002.2
2	2017	467.9	278.4	206.1	182.6	75.0	9.4	1.0	6.8	62.6	99.4	282.6	369.0	2040.6
3	2016	364.4	321.6	321.1	212.1	88.1	27.5	5.7	3.2	11.7	176.0	285.6	390.8	2207.3
4	2015	392.0	365.7	275.5	141.1	91.5	15.8	6.9	6.1	71.9	176.9	195.0	248.1	1986.2
5	2014	324.4	281.9	223.9	135.5	100.2	19.1	8.3	19.3	16.0	92.3	222.6	368.2	1811.5
6	2013	429.2	402.2	376.6	209.5	158.4	43.6	0.6	5.0	41.5	105.0	303.9	349.5	2424.8
7	2012	336.0	435.9	201.9	230.3	83.3	35.0	12.4	2.4	58.0	154.6	296.2	345.9	2191.5
8	2011	392.0	304.8	243.1	77.6	43.4	31.4	15.0	11.9	23.2	127.6	226.6	312.7	1809.0
9	2010	499.2	371.4	294.5	165.3	140.9	22.6	0.0	11.1	52.3	172.2	310.0	512.0	2551.1
10	2009	486.8	365.7	293.2	135.1	82.2	39.8	3.1	0.9	26.9	149.6	224.7	411.8	2219.7

Source : GRDF/CEGIBAT*
*centre d'expertise de GRDF

DJU

```python
dju = pd.melt(dju, id_vars='annee', var_name='Mois', value_name='dju')
```

	annee	Mois	dju
0	2019	01	404.9

```python
# Concaténation
dju['date'] = dju['annee'] + '-' + dju['Mois']
```

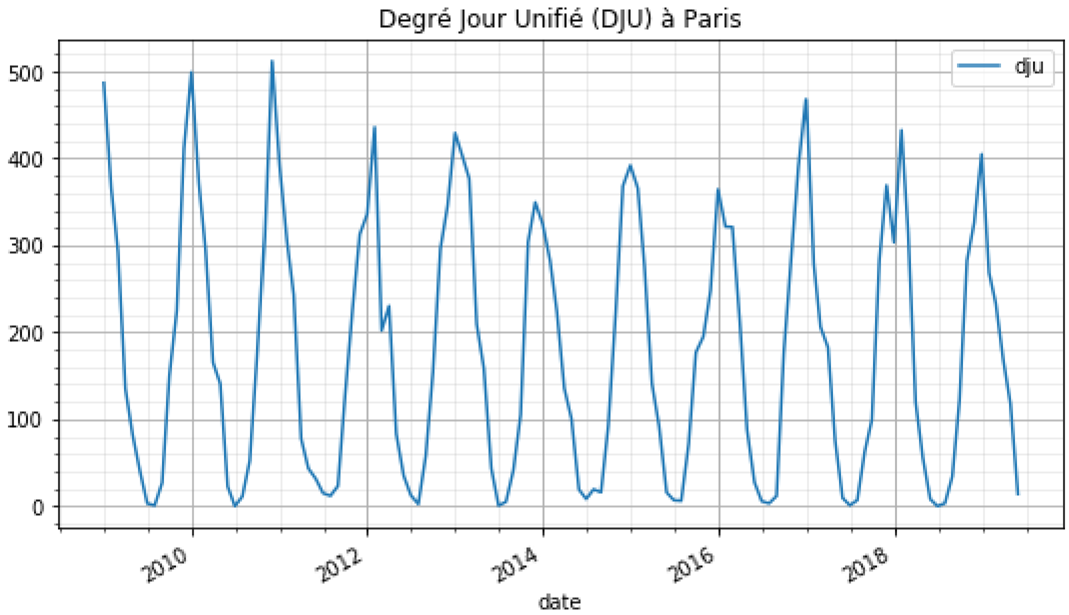
	dju	date
0	404.9	2019-01-01
1	303.4	2018-01-01
2	467.9	2017-01-01
3	364.4	2016-01-01
4	392.0	2015-01-01

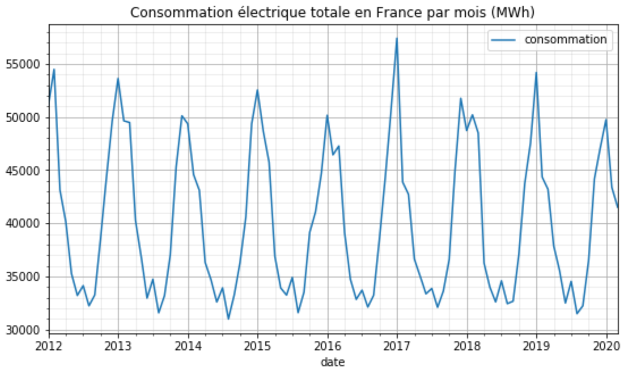
	Année	JAN	FÉV	MAR	AVR	MAI	JUN	JUI	AOÛ	SEP	OCT	NOV	DÉC	Total
0	2019	404.9	268.3	233.1	168.5	117.9	14.0	0.0	0.0	0.0	0.0	0.0	0.0	1206.6
1	2018	303.4	432.6	314.3	119.7	55.9	8.1	0.0	3.3	34.3	122.4	282.5	325.9	2002.2
2	2017	467.9	278.4	206.1	182.6	75.0	9.4	1.0	6.8	62.6	99.4	282.6	369.0	2040.6
3	2016	364.4	321.6	321.1	212.1	88.1	27.5	5.7	3.2	11.7	176.0	285.6	390.8	2207.3
4	2015	392.0	365.7	275.5	141.1	91.5	15.8	6.9	6.1	71.9	176.9	195.0	248.1	1986.2
5	2014	324.4	281.9	223.9	135.5	100.2	19.1	8.3	19.3	16.0	92.3	222.6	368.2	1811.5
6	2013	429.2	402.2	376.6	209.5	158.4	43.6	0.6	5.0	41.5	105.0	303.9	349.5	2424.8
7	2012	336.0	435.9	201.9	230.3	83.3	35.0	12.4	2.4	58.0	154.6	296.2	345.9	2191.5
8	2011	392.0	304.8	243.1	77.6	43.4	31.4	15.0	11.9	23.2	127.6	226.6	312.7	1809.0
9	2010	499.2	371.4	294.5	165.3	140.9	22.6	0.0	11.1	52.3	172.2	310.0	512.0	2551.1
10	2009	486.8	365.7	293.2	135.1	82.2	39.8	3.1	0.9	26.9	149.6	224.7	411.8	2219.7

```python
dju = dju[dju['date'] < '2019-07-01']
```

Présentation des données utilisées (source, pré-traitements, etc.)

Données météo pour corriger les données de conso. mensuelle de l'effet température





On observe aussi une saisonnalité de période 12.
La tendance est semblable à celle de la consommation d'énergie : les 2 variables semblent corrélées.

Préparation des données – tableau avec consommation et dju

```
# Jointure des 2 df via inner permet de ne garder que les données qui existent des 2 cotés.
data = pd.merge(conso, dju, left_on='mois', right_on='date', how='inner')
```

	Mois	Consommation totale
0	2012-01	51086
1	2012-02	54476
2	2012-03	43156
3	2012-04	40176
4	2012-05	35257

+

	dju	date
0	404.9	2019-01-01
1	303.4	2018-01-01
2	467.9	2017-01-01
3	364.4	2016-01-01
4	392.0	2015-01-01

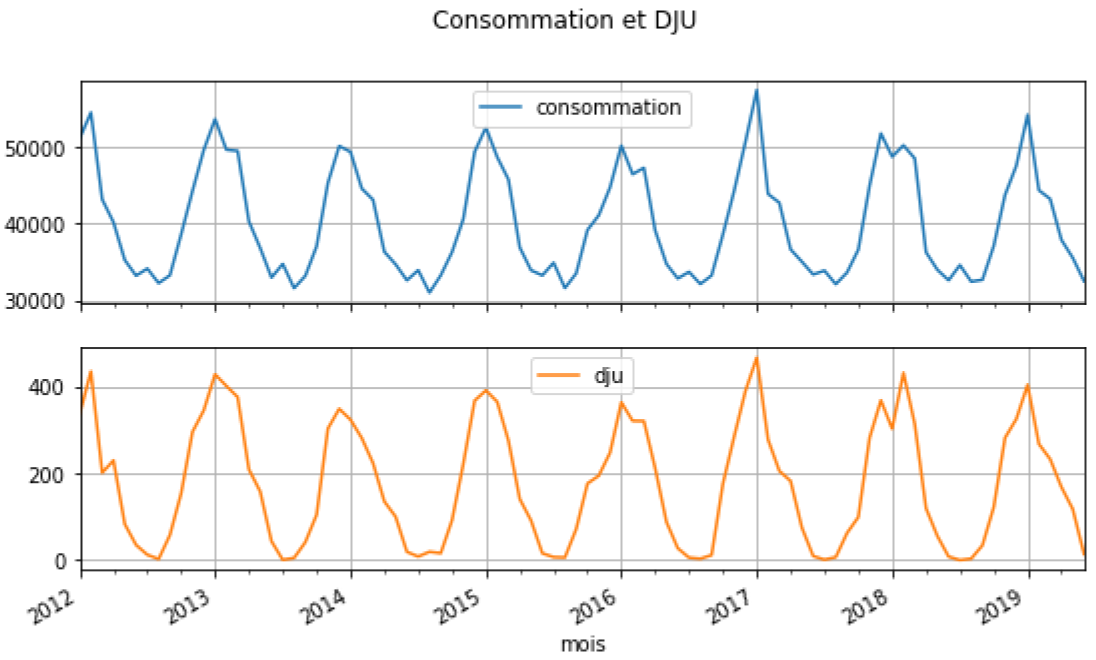
=

	consommation	dju
mois		
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

Dates : de janvier 2012 à juin 2019

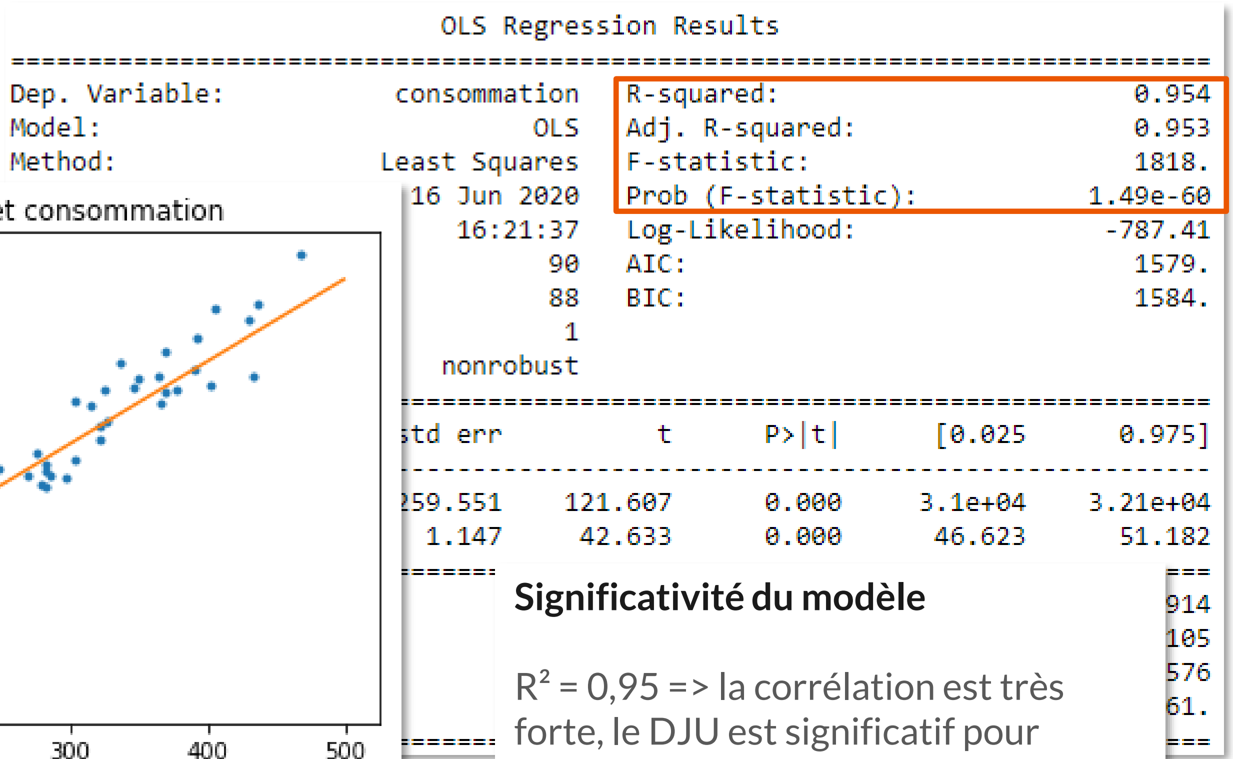
Préparation des données – représentation consommation et DJU

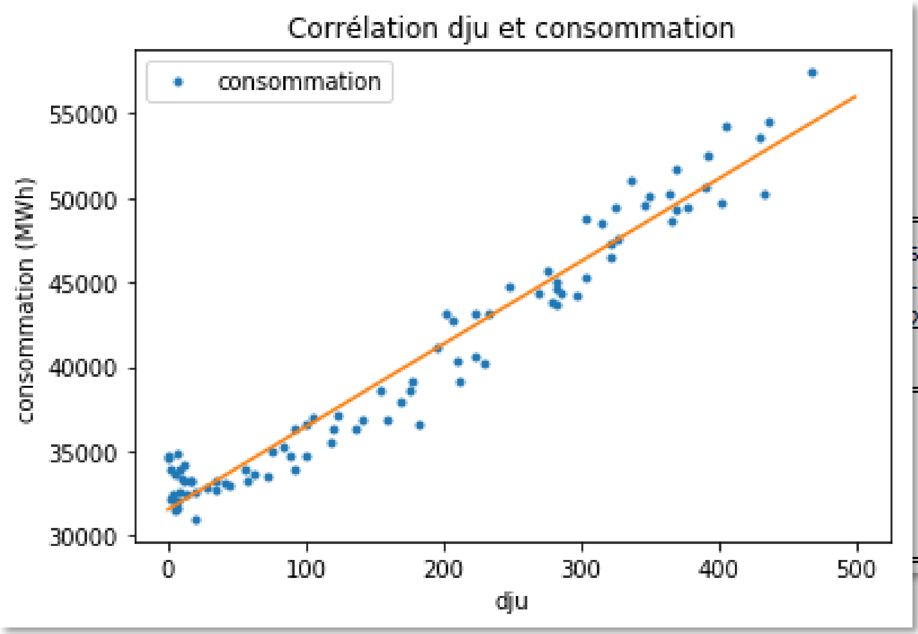
	consommation	dju
mois		
2012-01-01	51086	336.0
2012-02-01	54476	435.9
2012-03-01	43156	201.9
2012-04-01	40176	230.3
2012-05-01	35257	83.3



Correction de l'effet température via une régression linéaire

Visualisation de la corrélation entre les 2 variables


```
                            OLS Regression Results
==============================================================================
Dep. Variable:           consommation   R-squared:                       0.954
Model:                            OLS   Adj. R-squared:                  0.953
Method:                 Least Squares   F-statistic:                     1818.
                         16 Jun 2020   Prob (F-statistic):           1.49e-60
                            16:21:37   Log-Likelihood:                -787.41
                                  90   AIC:                             1579.
                                  88   BIC:                             1584.
                                   1
                           nonrobust
==============================================================================
 std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
259.551     121.607      0.000    3.1e+04    3.21e+04
  1.147      42.633      0.000     46.623      51.182
==============================================================================
                                                                          914
                                                                          105
                                                                          576
                                                                          61.
==============================================================================
```

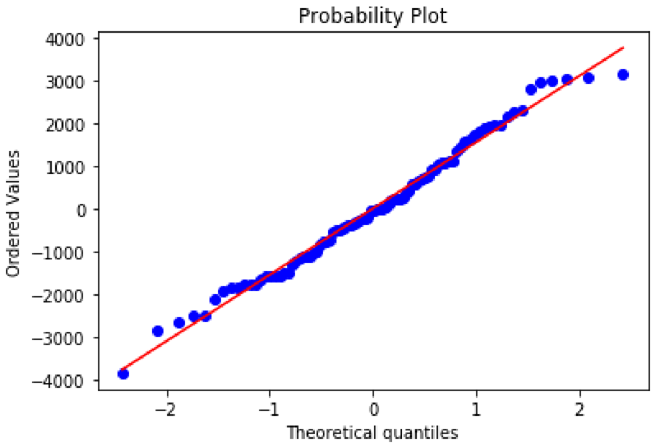


Significativité du modèle

$R^2 = 0{,}95$ => la corrélation est très forte, le DJU est significatif pour expliquer la consommation électrique.

Analyse des résidus pour vérifier la significativité de notre modèle

Normalité
OK



Shapiro & Jarque-Bera : p-value > 0,05
=> On ne peut pas rejeter H0 selon laquelle la distribution des résidus est conforme à une distribution normale.

Résidus indépendants (non corrélés)
OK

Test de Durbin-Watson = 1.914

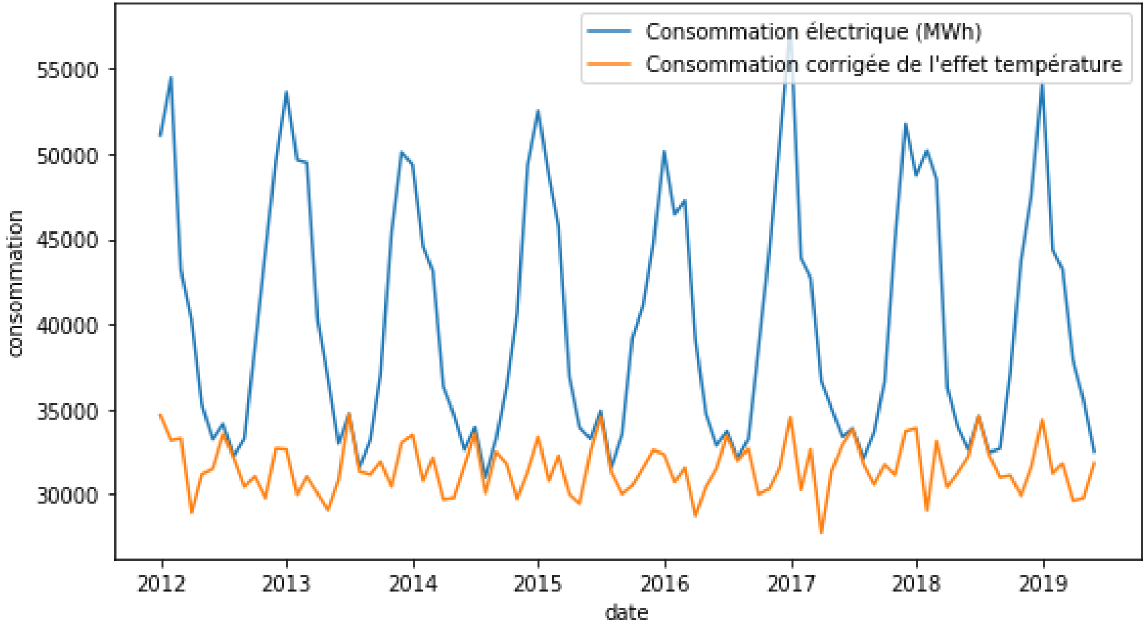
Variance (homoscédasticité : la variance des résidus est stable)
OK

Test de Breusch Pagan : p-value > 0.05 => nous ne pouvons pas rejeter l’hypothèse nulle d’homoscédasticité des résidus.

Représentation graphique

consommation corrigée = consommation – (DJU x coefficient(DJU))

```
# Je retire l'effet température de la consommation électrique
data['conso_corr'] = data['consommation'] - (data['dju'] * results.params['dju'])
```



Désaisonnalisation
grâce aux moyennes mobiles

Utilisation de « *seasonal_decompose* » du package statsmodels

L'algorithme est basé sur les moyennes mobiles.

Choix du modèle additif étant donné que l'écart entre le mini et la maxi sur l'année reste à peu près stable au cours du temps

```
decomp_cc = seasonal_decompose(data.conso_corr, model='additive')
decomp_cc_graph = decomp_cc.plot()
```

```
data.head()
```

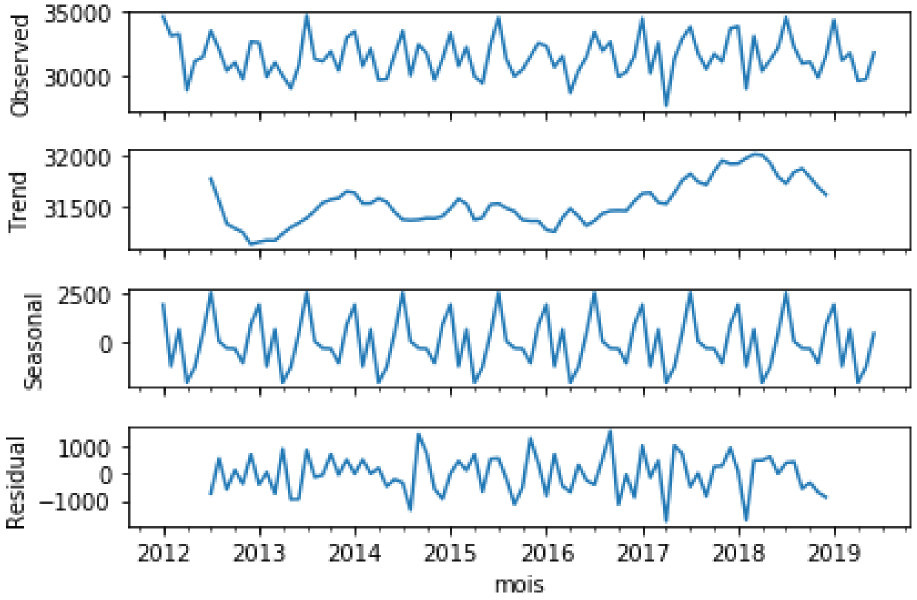
	consommation	dju	conso_corr
mois			
2012-01-01	51086	336.0	34654.752443
2012-02-01	54476	435.9	33159.390447
2012-03-01	43156	201.9	33282.580709
2012-04-01	40176	230.3	28913.749071
2012-05-01	35257	83.3	31183.419877

Utilisation de « *seasonal_decompose* » du package statsmodels

```
decomp_cc = seasonal_decompose(data.conso_corr, model='additive')
decomp_cc_graph = decomp_cc.plot()
```

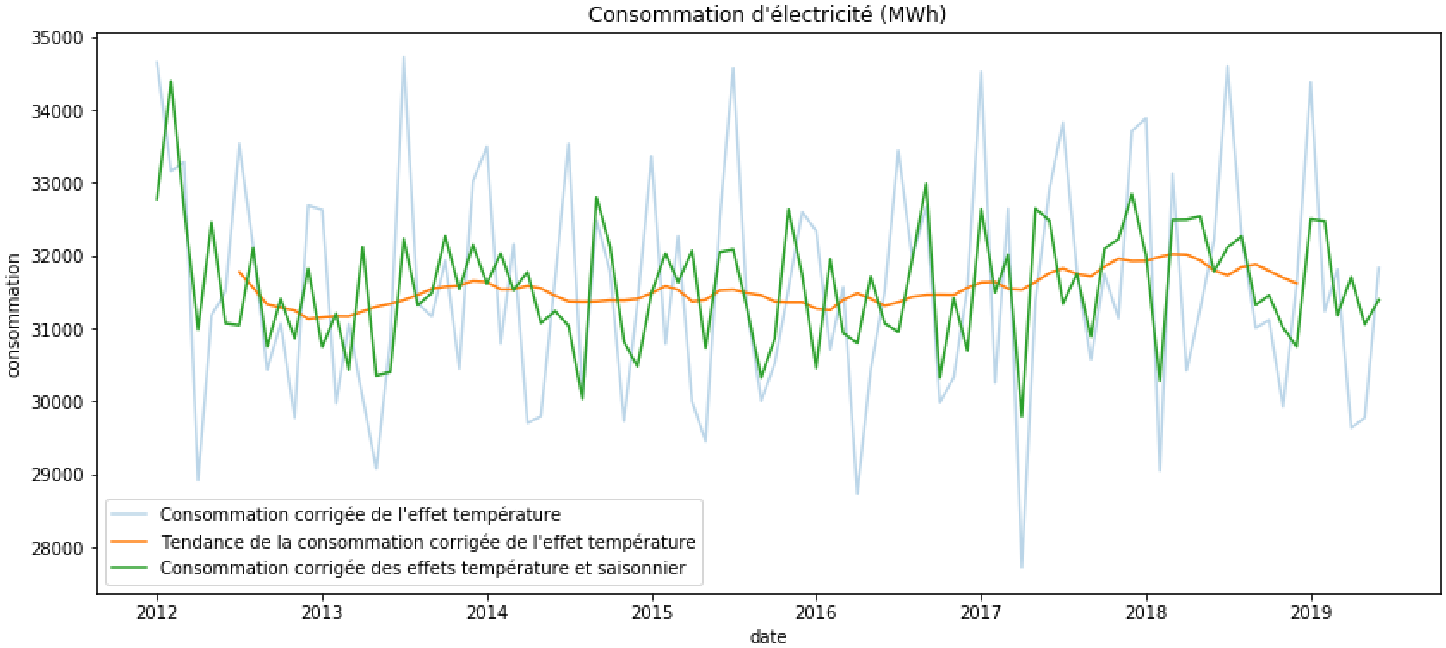
Nous obtenons donc :

- la tendance (traduit le niveau moyen de la série temporelle)
- la saisonnalité (l'ensemble des fluctualités que l'on retrouve dans la série temporelle, quelque chose de périodique)
- les résidus (le bruit, la part que l'on ne peut expliquer)



Représentation graphique

```
plt.plot(data.conso_corr, label="Consommation corrigée de l'effet température", alpha=0.30)
plt.plot(decomp_cc.trend, label="Tendance de la consommation corrigée de l'effet température")
plt.plot(data.conso_corr - decomp_cc.seasonal, label="Consommation corrigée des effets température et saisonnier")
```



Prévision de la consommation
via les méthodes
Holt-Winters et SARIMA

Prévision de la consommation

1. via la méthode Holt-Winters

Méthode de Holt-Winters (lissage exponentiel) : précisions théoriques

Le lissage exponentiel est une méthode empirique de lissage et de prévision de données chronologiques affectées d'aléas.

Comme dans la méthode des moyennes mobiles, chaque donnée est lissée successivement en partant de la valeur initiale.

Mais alors que la moyenne mobile accorde le même poids à toutes les observations passées à l'intérieur d'une certaine fenêtre, le lissage exponentiel donne aux observations passées un poids décroissant exponentiellement avec leur ancienneté.

(source : Wikipedia)

Exemple de lissage exponentiel simple.
Données brutes : températures moyennes quotidiennes à la station météo de Paris-Montsouris (France) du 01/01/1960 au 29/02/1960.
Données lissées avec le facteur α = 0,1.

Méthode de Holt-Winters (lissage exponentiel) : précisions théoriques

La méthode de Holt-Winters est une méthode de lissage exponentiel double.

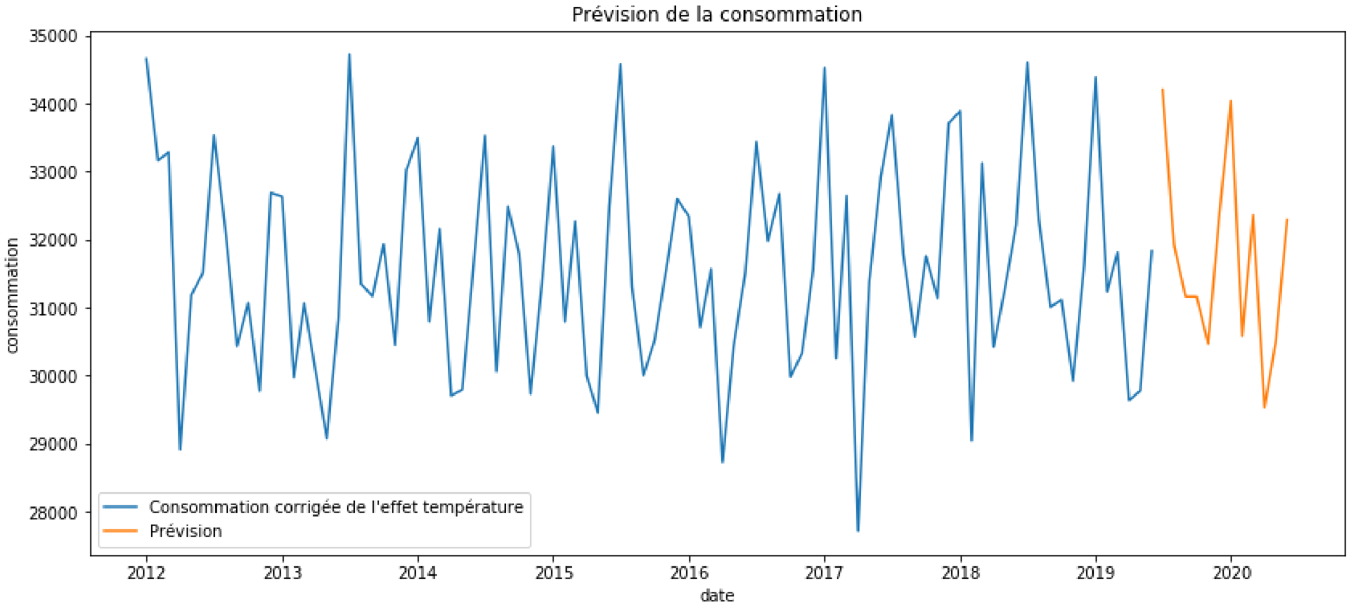
Le lissage exponentiel simple ne donne pas de bons résultats lorsque les données brutes présentent une ou des tendances. Les valeurs lissées présentent une sous-estimation ou une surestimation systématique selon le sens de la tendance.

Les méthodes de lissage exponentiel double ont pour objet de lisser le niveau des données (c'est-à-dire d'éliminer les variations aléatoires) et de lisser la tendance, c'est-à-dire d'éliminer l'effet de la tendance sur les valeurs lissées.

(source : Wikipedia)

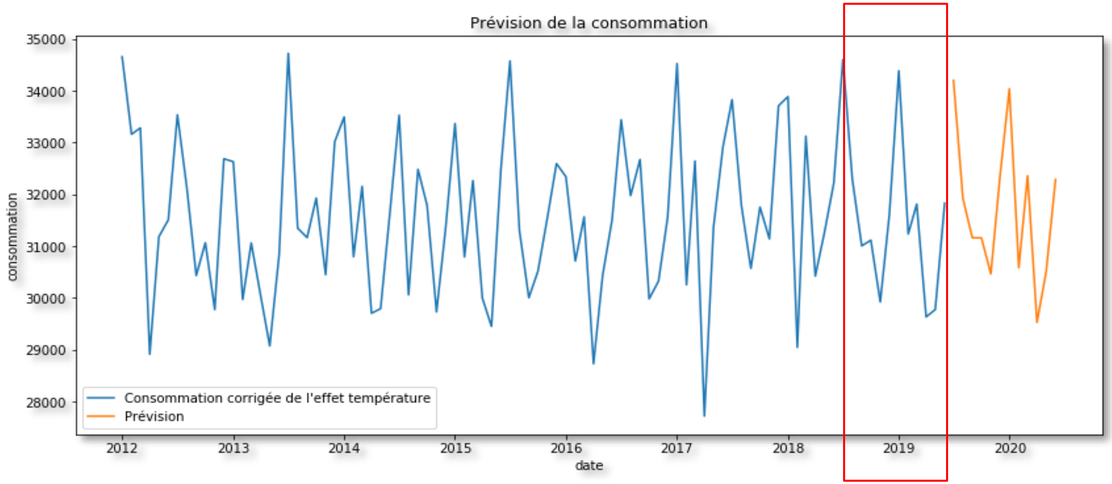
Méthode de Holt-Winters (lissage exponentiel ou *exponential smoothing*)

```
y = data.conso_corr
# Modèle
hw_meth = ExponentialSmoothing(np.asarray(y), seasonal_periods=12,trend='add', seasonal='add').fit()
# Prévision
prev_hw_meth = hw_meth.forecast(12)
```



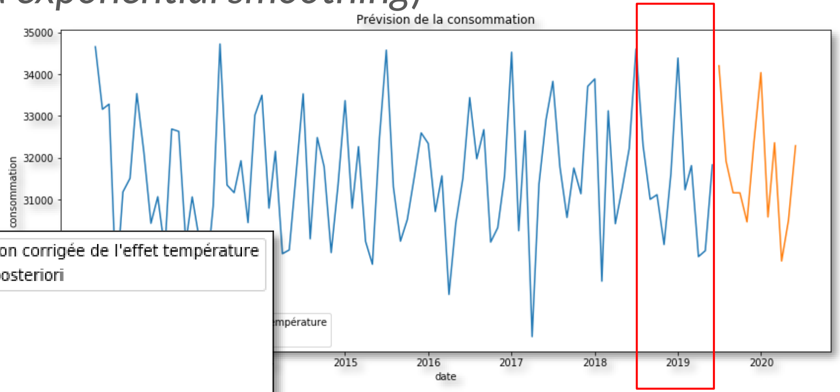
Méthode de Holt-Winters (lissage exponentiel ou *exponential smoothing*)

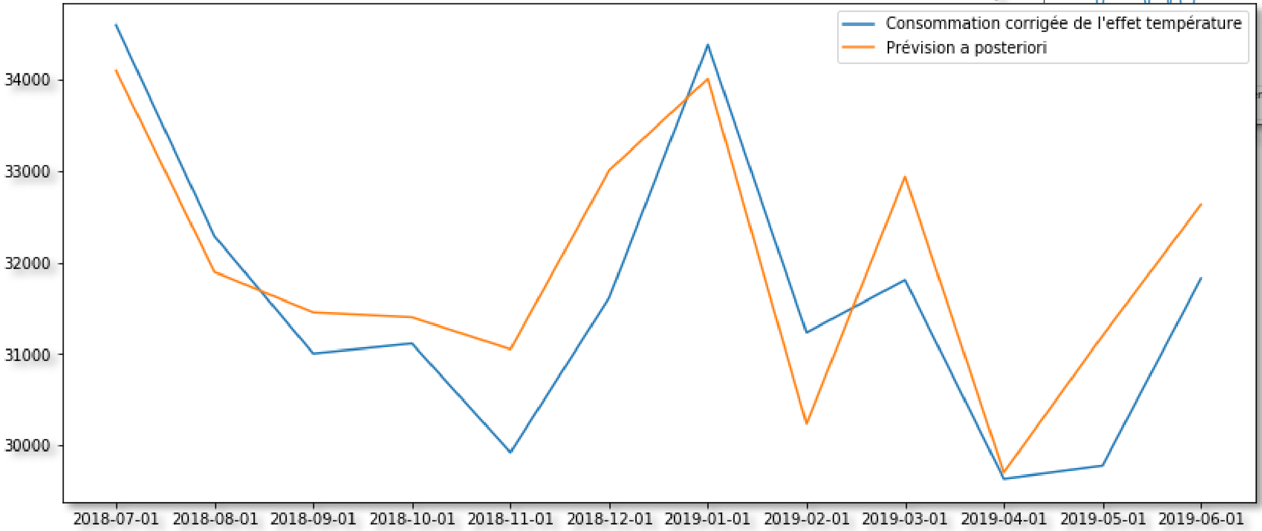
Il est possible de faire une prévision *a posteriori*.




```
# nous allons afficher les n derniers mois
nb_mois = 12
plt.figure(figsize=(14,6))
plt.plot(y[-nb_mois:], label="Consommation corrigée de l'effet température")
plt.plot(prev_hw_meth_post.iloc[-nb_mois:], label='Prévision a posteriori')
```

Méthode de Holt-Winters (lissage exponentiel ou *exponential smoothing*)





Nous constatons un résultat très correct.

RMSE* = 866.46
Root Mean Squared Error

MAPE** = 2.33 %
Mean Absolute Percentage Error ou erreur absolue moyenne en %

**RMSE : racine carrée de la moyenne arithmétique des carrés des écarts entre prévisions du modèle et observations.*
***MAPE : moyenne des écarts en valeur absolue par rapport aux valeurs observées.*

Prévision de la consommation

2. via la méthode SARIMA

Méthode SARIMA : point théorique

ARMA, ARIMA, SARIMA ?

Les processus ARIMA et SARIMA sont la généralisation des modèles ARMA pour des processus non stationnaires, admettant une tendance (ARIMA), ou encore une tendance et une saisonnalité (SARIMA).

Méthode SARIMA (*Seasonal AutoRegressive Integrated Moving Average)*

Que veut dire SARIMA ?

$$\text{SARIMA}(p,d,q).\underbrace{(P,D,Q)}_{\text{partie saisonnière}}s$$

p : nombre de termes autorégressifs (AR)
d : nombre de différences non saisonnières
q : nombre de termes moyens mobiles (MA)

AutoRegressive : définit une relation linéaire entre un instant t et les p instants qui précèdent.

Moving Average : définit une relation entre une perturbation décorrélée entre un instant t et les q instants qui précèdent.

Seasonal : Cette décomposition est aussi faite pour la partie saisonnière, selon les paramètres P et Q.

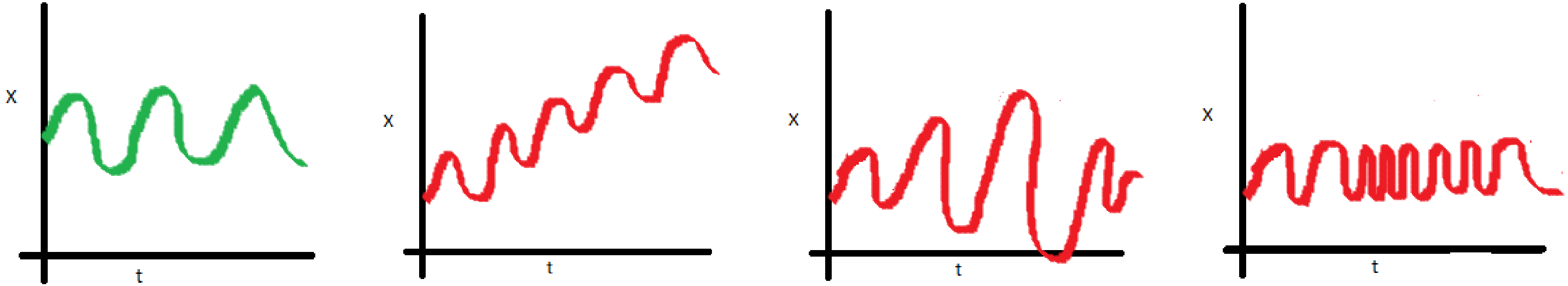
Integrated : Stationnarisation incluse.

Méthode SARIMA (*Seasonal Autoregressive Moving Average)*

Un processus est **stationnaire** (la structure du processus reste la même avec le temps) si :

- son espérance est constante à travers le temps (pas de tendance)
- sa variance est constante à travers le temps
- les autocorrélations entre deux moments séparés dans le temps sont constantes (la position dans le temps ne joue pas de rôle)

Si une série temporelle est stationnaire et présente un comportement particulier pendant un intervalle de temps donné, elle présentera le même comportement à un moment ultérieur.
Rendre notre processus stationnaire nous permettra d'appliquer des méthodes de régression.



cf. https://www.seanabu.com/2016/03/22/time-series-seasonal-ARIMA-model-in-python/

Méthode SARIMA (*Seasonal Autoregressive Moving Average)*

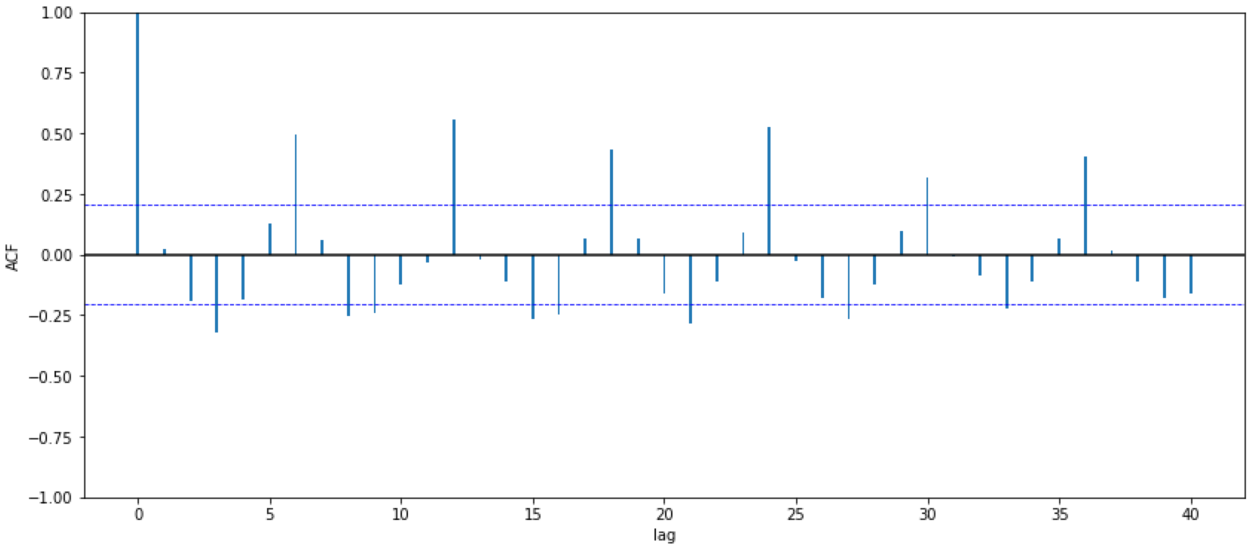
La démarche adoptée est la suivante :

1. stationnarisation
2. identification *a priori* de modèles potentiels
3. estimation des modèles potentiels
4. vérification des modèles potentiels
5. choix définitif d'un modèle
6. prévision à l'aide du modèle choisi
7. analyse *a posteriori* de la prévision

Stationnarisation : autocorrélogramme simple

Diagramme de corrélation simple sans appliquer de différenciations à y :

```
plot_sortie_acf(acf(np.asarray(y)), len(y))
```



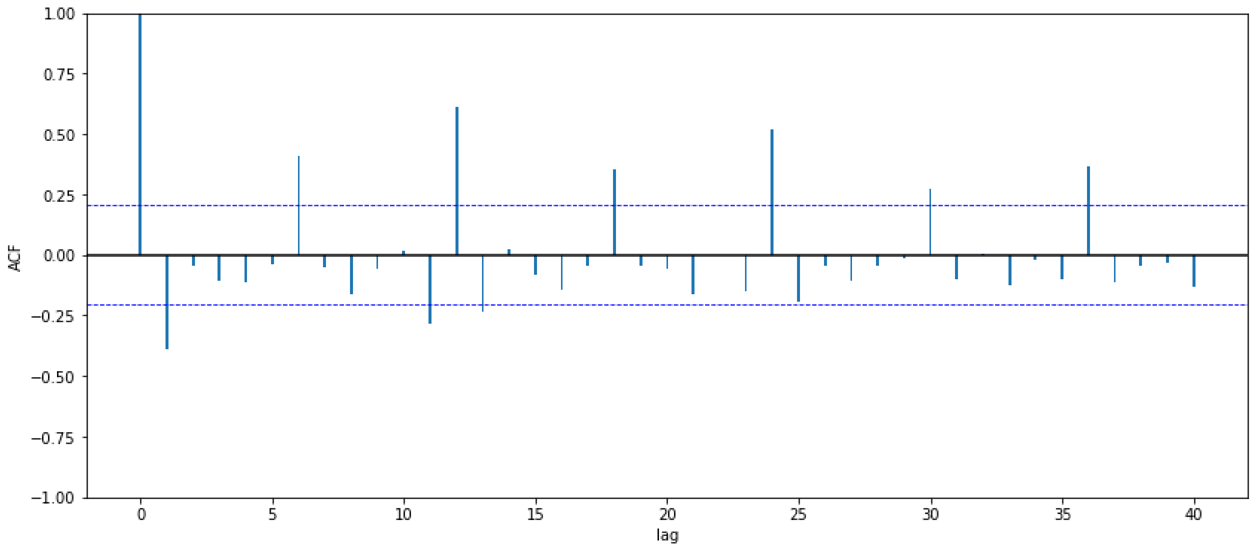
*Les **pointillées horizontaux**, sont les intervalles de confiance du coefficient de corrélation égal à 0.*
*Les **traits verticaux** représentent les coefficients de corrélation entre les résidus de chaque point et ceux des points de la ligne suivante (lag=1), ou ceux séparés de deux lignes (lag=2) etc.*

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

Autocorrélogramme simple

En effectuant une 1ère différenciation (I−B), la sortie ACF présente encore une décroissance lente vers 0.

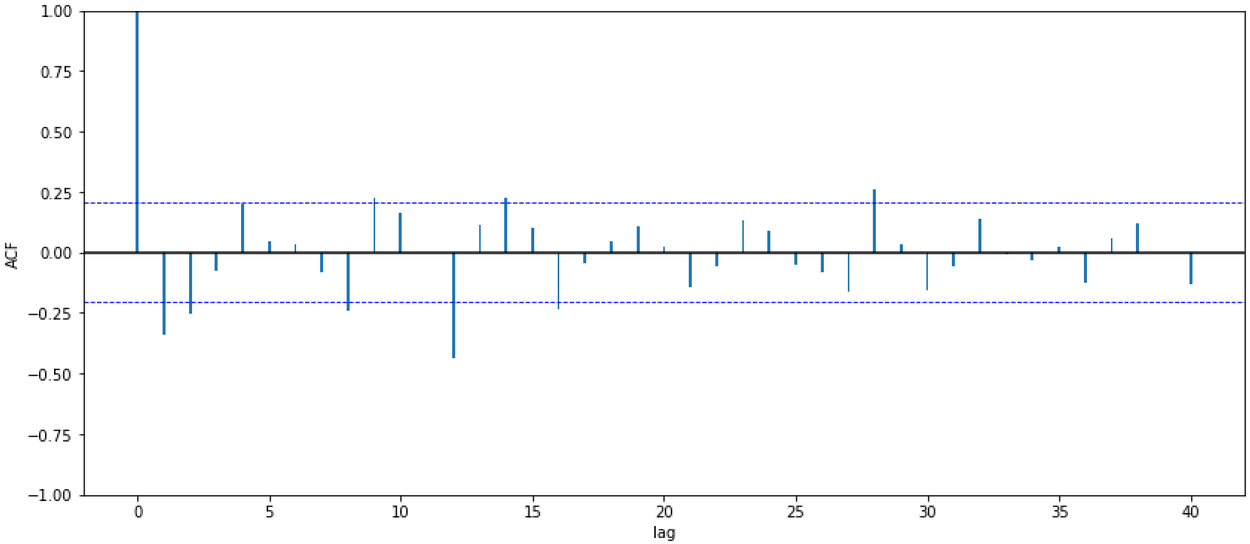
```
y_dif1 = (y - y.shift(1))
plot_sortie_acf(acf(np.asarray(y_dif1[1:])), len(y))
```



Autocorrélogramme simple

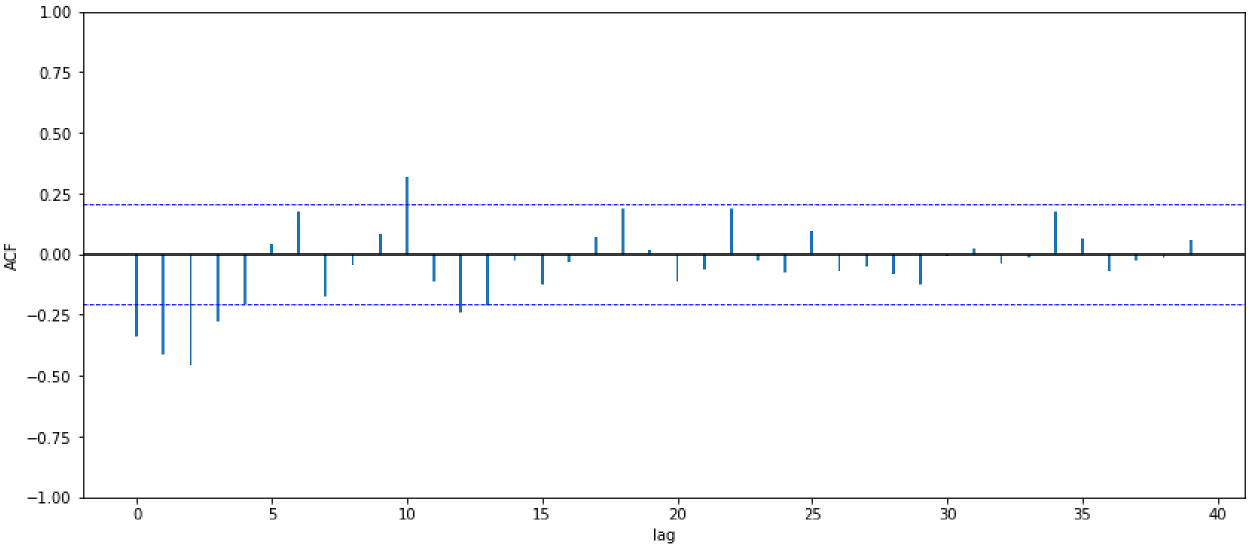
En effectuant une 2ème différenciation, la sortie semble pouvoir être interprétée comme un autocorrélogramme simple empirique

```
y_dif_1_12 = y_dif1 - y_dif1.shift(12)
plot_sortie_acf(acf(np.asarray(y_dif_1_12[13:])), len(y))
```



Autocorrélogramme partiel (*partial ACF*)

```
plot_sortie_acf(pacf(np.asarray(y_dif_1_12[13:]), method='ldb'), len(y), pacf=True)
```



On ne constate plus de pics significatifs (au dessus des pointillés, qui représentent le seuil à 5%) à intervalle régulier.
La différenciation effectuée semble suffire pour stationnariser la série temporelle.

Recherche des paramètres p,d,q et P,D,Q

Au vu des autocorrélogrammes empiriques simples et partiels, on peut estimer plusieurs modèles.

J'ai commencé par (1,1,1)x(1,1,1) puis (1,1,1)x(0,1,1) qui ne sont pas satisfaisants.

Le modèle (0,1,1)x(0,1,1) a des paramètres significatifs.

Ce résultat est confirmé via un petit algorithme d'estimation des paramètres. En effet, le modèle obtient le plus faible AIC*.

```
grid_search_results.sort_values('AIC').head(5)
```

	modele	AIC	BIC
27	(0, 1, 1)x(0, 1, 1, 12)	1051.903408	1058.332812
31	(0, 1, 1)x(1, 1, 1, 12)	1053.092932	1061.665471
59	(1, 1, 1)x(0, 1, 1, 12)	1053.109104	1061.681643
63	(1, 1, 1)x(1, 1, 1, 12)	1065.013496	1075.729169
15	(0, 0, 1)x(1, 1, 1, 12)	1068.159660	1076.795193

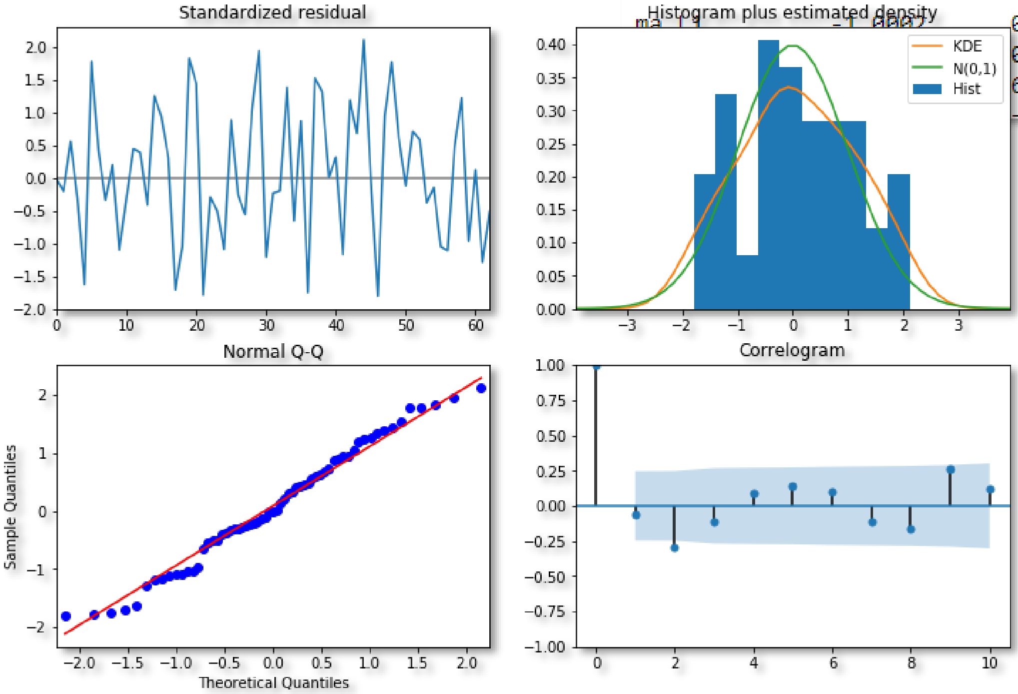
**According Peterson, T. (2014) the AIC (Akaike information criterion) is an estimator of the relative quality of statistical models for a given set of data. Given a collection of models for the data, AIC estimates the quality of each model, relative to each of the other models. The low AIC value the better.*
Cf. https://towardsdatascience.com/how-to-forecast-sales-with-python-using-sarima-model-ba600992fa7d

Prévision de la consommation – méthode SARIMA

SARIMA(0,1,1)(0,1,1)$_{12}$

Les tests de significativité des paramètres et de blancheur du résidu sont validés au niveau 5%.

	coef	std err	z	P>|z|	[0.025	0.975]
[illegible]	[illegible]	0.218	-4.595	0.000	-1.427	-0.574
		0.074	-7.984	0.000	-0.737	-0.446
		6e-07	3.29e+12	0.000	8.45e+05	8.45e+05



Test de Ljung-Box **OK** :
les observations sont aléatoires et indépendantes
(= bruit blanc)

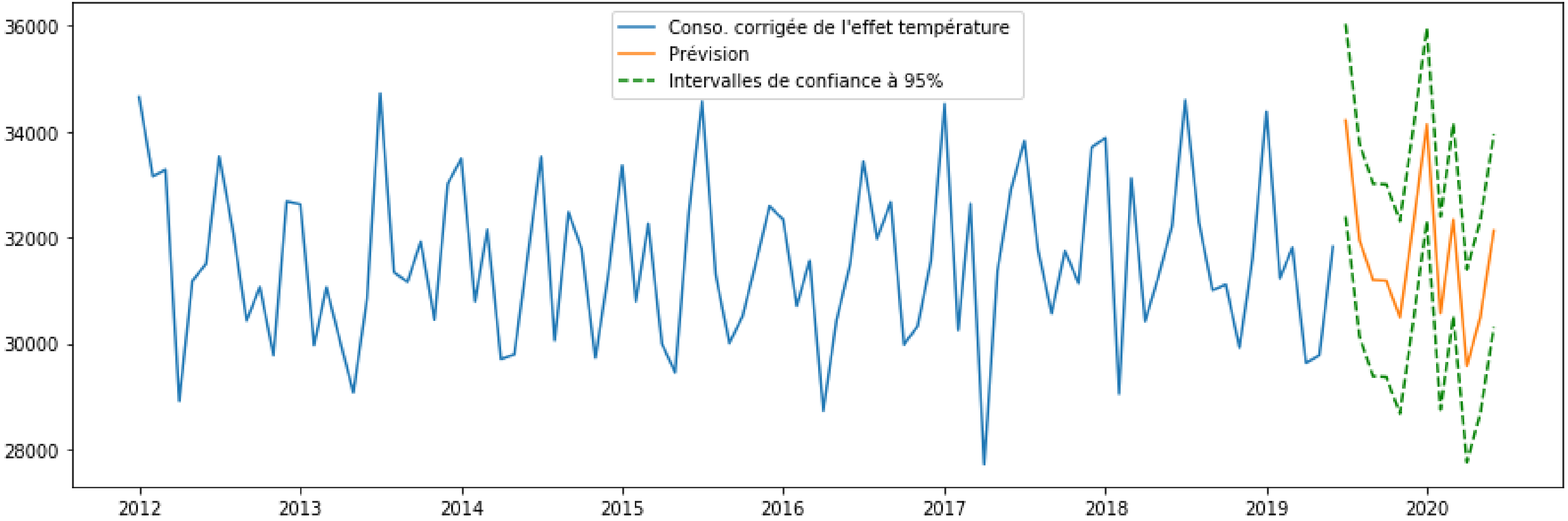
```
Retard : p-value
6 : 0.8838651270870385
12 : 0.8630423846058252
18 : 0.6460444353343495
24 : 0.6149421035413486
30 : 0.6505425230138061
36 : 0.6978968339029064
```

Analyse des résidus :
- QQ plot → OK
- KDE : distribution normale → OK
- Homoscédasticité → OK
- Indépendance → OK.

Prévision de la consommation – méthode SARIMA

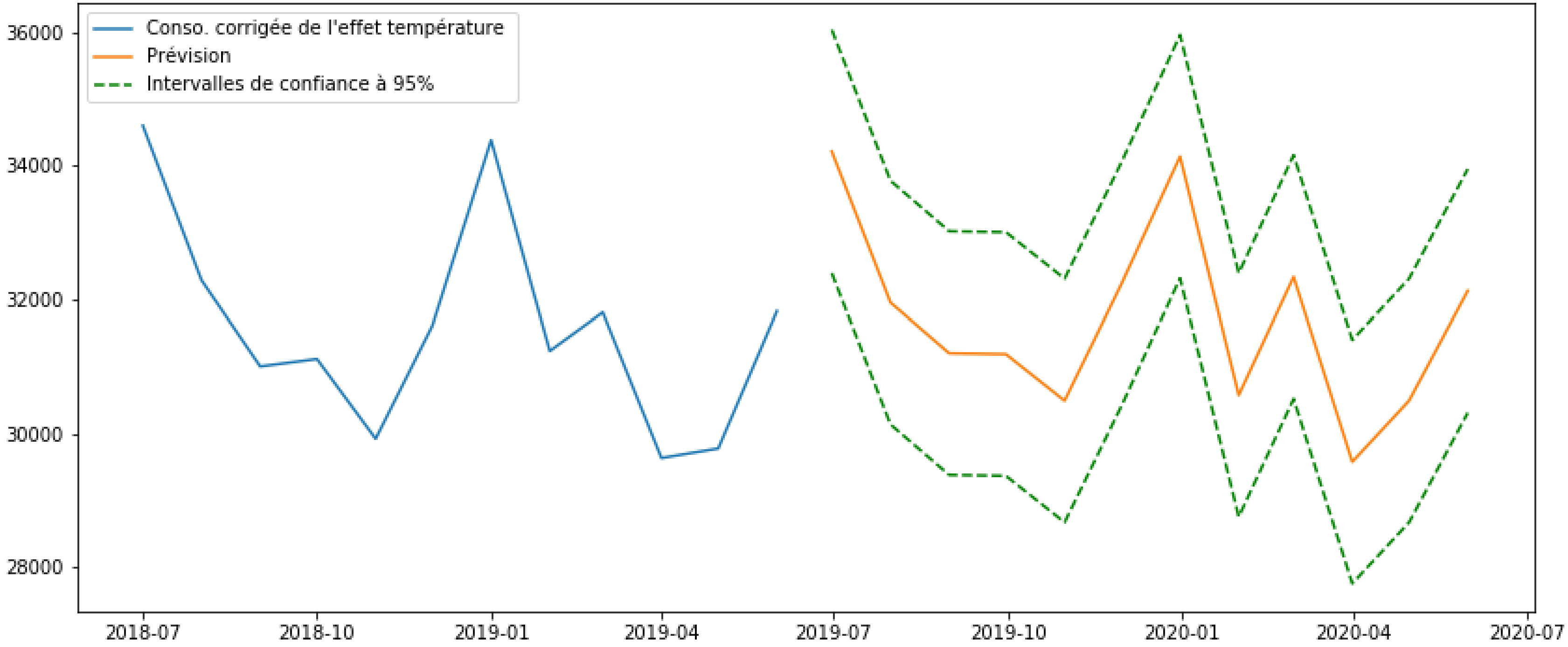
SARIMA(0,1,1)(0,1,1)12 – Prévision à l'aide du modèle retenu

```python
# Nous allons faire une prévision sur un an
pred_model1 = results1.get_forecast(12)
pred = pred_model1.predicted_mean
pred_l = [elt[0] for elt in pred_model1.conf_int(alpha=0.05)]
pred_u = [elt[1] for elt in pred_model1.conf_int(alpha=0.05)]
```



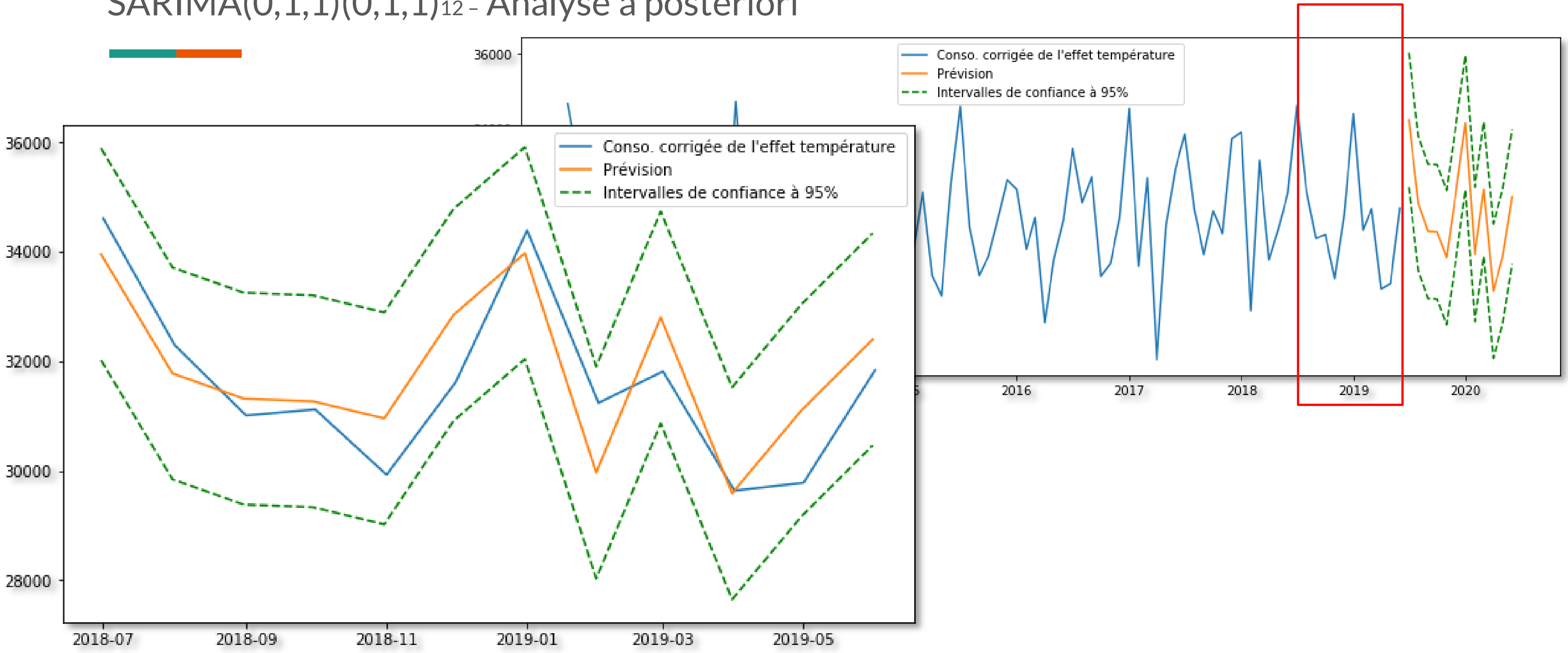
Prévision de la consommation – méthode SARIMA

SARIMA(0,1,1)(0,1,1)$_{12}$ – Prévision à l'aide du modèle retenu (zoom)



Prévision de la consommation – méthode SARIMA

SARIMA(0,1,1)(0,1,1)12 – Analyse a posteriori

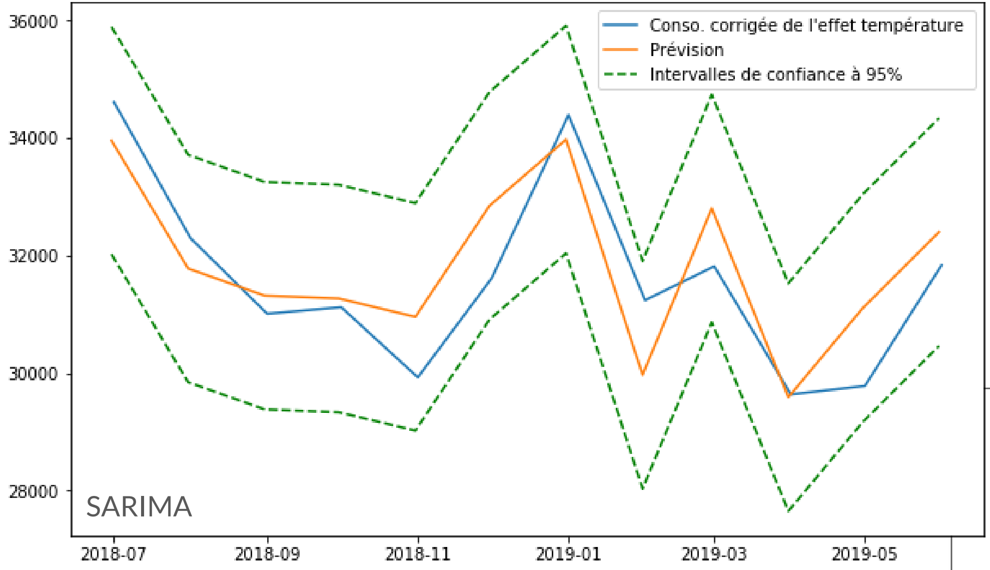


Notre modèle englobe bien de mi-2018 à juin 2019 sur un intervalle de confiance à 95%

L'intervalle de confiance à 95% est basé sur les données antérieures à 07/2018.

Prévision de la consommation – méthode SARIMA

SARIMA(0,1,1)(0,1,1)12 – Analyse a posteriori



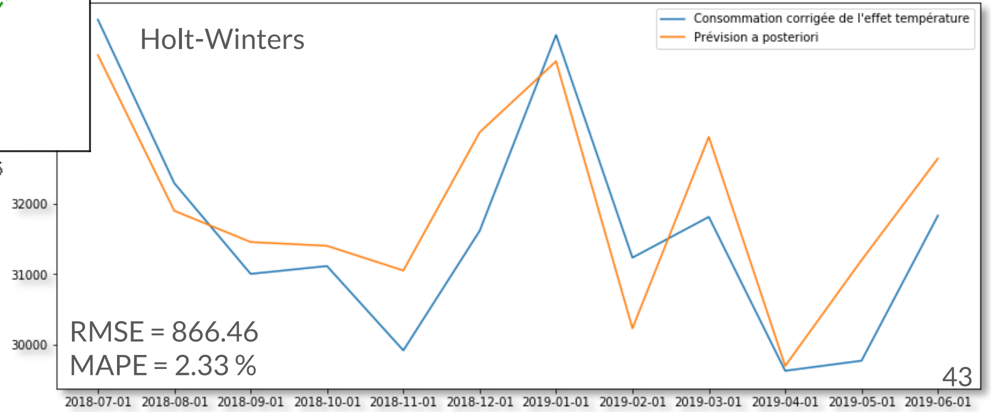
RMSE = 446,18

Root Mean Squared Error

MAPE = 1,23 %

Mean Average Percentage Error ou erreur relative absolue moyenne

Résultats meilleurs qu'avec Holt-Winters



Conclusion

Conclusion

Nous pouvons effectivement **améliorer l'adéquation entre l'offre et la demande**.

Pour cela, nous avons à notre disposition :

- La **régression linéaire** pour corriger les données de consommation mensuelles de l'effet température;
- Les **moyennes mobiles** (désaisonnalisation de la consommation après correction)
- Les **méthodes Holt Winters et SARIMA** (prévision de la consommation corrigée de l'effet température), sachant que la méthode SARIMA nous donne les meilleurs résultats.

Merci de votre attention!