



DataScientest • com

*Rapport Technique d'évaluation*

# PROJET ENERGIE

Promotion Data Scientist BootCamp  
JUIN 2021 – SEPTEMBRE 2021

Participants :

Samira FLICI

Joëlda KPODJA

Alexandre RASCHIA

Mentor :

Frédéric FRANCINE

## Table des matières

Introduction .....	2
I. Objectifs du projet .....	2
II. Aspects Techniques.....	3
III. Data sets .....	5
IV. Dataviz Equilibre Consommation / Production .....	10
V. Modèles de prédition de la Consommation.....	22
A. Modèle SARIMA.....	22
B. Modèles de Régression régularisée .....	27
1. Modèle Ridge.....	32
2. Modèle Lasso.....	35
3. Modèle ElasticNet .....	39
VI. Dataviz Sources de Production .....	48
VII. Vie du Projet .....	60
A. Description des travaux.....	60
B. Difficultés rencontrées .....	61
C. Description des livrables – fichiers de code .....	61
VIII. Conclusion .....	62
IX. ANNEXES.....	0
A. Diagramme de GANTT .....	0
B. Bibliographie .....	0

# Introduction

Le sujet de ce projet porte sur l'analyse des données de production et de consommation du réseau électrique français. La source du jeu de données est celle de l'ODRE (Open Data Réseaux Energies) avec un accès à toutes les informations de consommation et de production par Région et par filière jour par jour (toutes les 1/2 heures) depuis 2013.

## I. Objectifs du projet

Les objectifs du projet s'expriment à travers trois questionnements que nous nous sommes posés et auxquels nous avons voulu répondre dans un analyse détaillée présentée dans ce rapport. Les trois problématiques se résument ainsi :

- Comment est assuré l'équilibre entre consommation et production au niveau national et régional ?
- Sommes-nous capables de prédire correctement la consommation avec un(des) modèle(s) de machine learning afin de prévoir les besoins de production ?
- Quelles sont les sources d'énergies au niveau national et régional qui contribuent à satisfaire les besoins d'électricité et dans quelles proportions ?

### Expertise du groupe sur le sujet et les problématiques adressées :

**Samira** : Economètre de formation, j'ai été pendant une dizaine d'années data analyst et chef de projets, dans le marketing, les média ou le conseil. Le sujet du projet m'a permis de mettre en application mes connaissances en traitement de la donnée avec les nouveaux outils appris pendant la formation et d'en apprendre plus à travers différentes lectures sur les problématiques de l'énergie.

**Joëlda** : Étudiante en Mathématiques et Informatique, l'étude et la réalisation du projet centré sur la production et la consommation d'énergie m'a permis de mettre en pratique de manière concrète les connaissances acquises pendant mon cursus surtout au niveau théorique et comment les exploiter en une expérience supplémentaire en réalisation de projet.

**Alexandre** : ingénieur de formation avec 20 ans d'expériences dans l'industrie aéronautique, la gestion de projet et le conseil, le sujet de la fourniture d'électricité et des économies d'énergies a été étudié à titre personnel à travers des lectures diverses sur la transition énergétique et sa faisabilité.

Afin de récupérer des données complémentaires qui nous semblaient pertinentes pour les modèles, nous avons contacté l'INSEE pour collecter 2 types de données : la population et les entreprises réparties sur le territoire.

## II. Aspects Techniques

Dans ce chapitre, nous proposons d'aborder succinctement les aspects techniques du réseau électrique français.

Le développement des usages électriques depuis le milieu du XXe siècle a abouti à la construction d'un système de production centralisé, associé à un réseau électrique interconnecté et maillé à l'échelle nationale et continentale.

Ce réseau électrique est constitué de 3 types de réseaux :

- Les réseaux de transport sont basés sur une structure de réseau maillée (« autoroutes de l'énergie »). Ils sont à haute tension (de 50 kV à 400 kV) et ont pour but de transporter l'énergie des grands centres de production vers les régions consommatrices d'électricité. Les grandes puissances transitées imposent des lignes électriques de forte capacité de transit, ainsi qu'une structure maillée (ou interconnectée).
  - Les réseaux de répartition (haute tension de l'ordre de 30 à 150 kV) ont pour but d'assurer à l'échelle régionale la fourniture d'électricité. L'énergie y est injectée essentiellement par le réseau de transport via des transformateurs, mais également par des centrales électriques de moyennes puissances (inférieures à environ 100 MW). Les réseaux de répartition sont distribués de manière assez homogène sur le territoire d'une région.
  - Les réseaux publics de distribution d'électricité desservent en moyenne et basse tension (20 kV et 400 V), selon une architecture en arborescence, les consommateurs finaux et les clients domestiques et professionnels (commerçants, artisans, petites industries). Leur longueur cumulée dépasse 1,3 million de kilomètres. L'interface entre les réseaux moyenne et basse tension est assurée par quelque 700 000 « postes de distribution ».
- Le développement de la production d'énergie décentralisée (éolien, photovoltaïque, etc.) et de nouveaux usages (autoproduction, électromobilité, etc.) modifient le rôle des réseaux de distribution qui deviennent collecteurs de l'énergie produite par les plus petites installations de production.

Le bon fonctionnement du réseau de transport repose sur des équilibres instantanés et sur le respect de très nombreuses contraintes techniques évoluant au cours du temps. Ainsi, sa gestion est complexe et délicate.

En France, RTE est le propriétaire et le gestionnaire du réseau public de transport d'électricité dont la longueur cumulée atteint 100 000 kilomètres environ.

## RTE, Responsable de l'équilibrage du système électrique

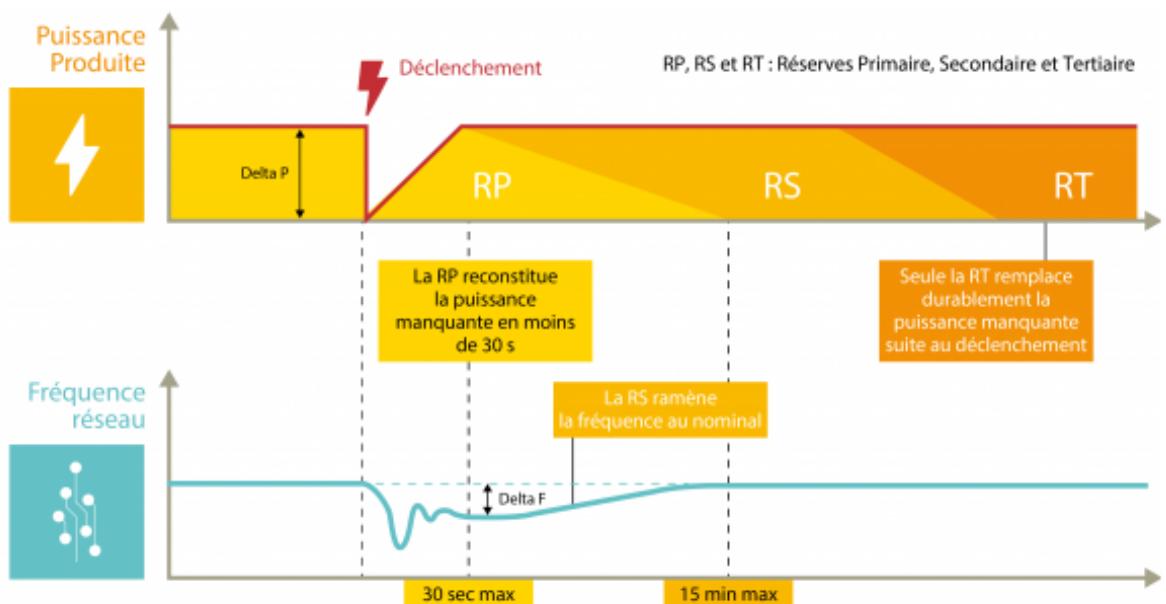
Gestionnaire du réseau public de transport d'électricité, RTE doit :

- Assurer à tout instant l'équilibre entre la production et la consommation d'électricité ;
- Résoudre les congestions sur le réseau de transport.

Dans ce but, RTE constitue et active des réserves d'équilibrage fournies par les acteurs d'ajustement : producteurs, consommateurs, autres acteurs susceptibles d'injecter ou de soutirer de l'énergie sur le réseau.

RTE dispose de trois types de réserves - primaire, secondaire, tertiaire - pour résorber les déséquilibres entre la production et la consommation d'électricité.

## Fonctionnement des réserves pour rétablir l'équilibre production / consommation



Note : la Puissance produite et la Fréquence du réseau sont directement liées.

En cas différence de Puissance (Delta P) entre la Production et la Consommation, les réserves primaires (RP) permettent de reconstituer la puissance manquante en moins de 30 secondes, puis les réserves secondaires (RS) ramènent la fréquence au nominal (50Hz) et enfin les réserves tertiaires remplacent la puissance manquante dans la durée.

Ce système permet d'éviter un black-out du réseau qui priverait d'électricité une partie ou l'ensemble de la France.

De plus, l'équilibrage du réseau est effectué entre régions limitrophes (françaises ou autres pays si besoin) ; l'électricité étant consommé au plus près de sa source de production pour des raisons de pertes en ligne.

### III. Data sets

Descriptif du jeu de données initiale ainsi que des jeux complémentaires utilisés dans le projet.

#### **Jeu de données initial**

##### **Données éCO2mix régionales consolidées et définitives (janvier 2013 à juin 2021)**

*Format : fichier csv*

<https://opendata.reseaux-energies.fr/explore/dataset/eco2mix-regional>

Ce jeu de données, présente les données régionales consolidées depuis janvier 2020 et définitives (de janvier 2013 à décembre 2019) issues de l'application éCO2mix. Elles sont élaborées à partir des comptages et complétées par des forfaits.

Ce jeu de données contient 1 787 328 lignes au total.

Chaque entrée regroupe les informations suivantes :

- Code INSEE Région
- Région
- Nature des données
- Date
- Heure
- Date-Heure
- Consommation (MW)
- Production Thermique (MW)
- Production Nucléaire (MW)
- Production Eolien (MW)
- Production Solaire (MW)
- Production Hydraulique (MW)
- Pompage système STEP (MW)
- Production Bioénergies (MW)
- Echanges physiques (MW) : solde des échanges avec les régions limitrophes
- Différents Flux physiques
- TCO (%) pour chaque type de source (thermique, nucléaire, etc)
- TCH (%) pour chaque type de source (thermique, nucléaire, etc)

Notes explicatives :

- ✓ Les puissances données en MW sont des puissances instantanées.
- ✓ La Production Thermique correspond à la puissance produite par les centrales thermiques à flamme. Ces centrales peuvent être des centrales à charbon, des centrales fioul-vapeur, des turbines à combustion ou des cycles combinés à gaz.
- ✓ La Production Nucléaire correspond à la puissance produite par les centrales nucléaire.
- ✓ La Production Eolien correspond à la puissance produite par les parcs d'éoliennes.
- ✓ La Production Solaire correspond à la puissance produite par les centrales solaires des fournisseurs d'énergie et les installations photovoltaïques des particuliers qui réinjectent sur le réseau.
- ✓ La Production Hydraulique correspond à la puissance produite par les centrales hydrauliques de type « lac » (barrages), au « fil de l'eau » ou « éclusée ».
- ✓ Le Pompage STEP correspond à la puissance stockée par les Stations de Transfert d'Énergie par Pompage. Elle est négative lorsqu'elle pompe l'eau en aval pour la stocker dans la retenue en amont (stockage de l'énergie électrique en énergie potentielle) et est restituée en Puissance hydraulique lorsque nécessaire.
- ✓ La production Bioénergies correspond à la puissance produite à partir de la biomasse et réinjecté sur le réseau (bois-énergie, biocarburant, biogaz, ...)
- ✓ Les Echanges physiques représentent le solde des échanges avec les régions limitrophes (Exportateur si négatif et Importateur si positif).
- ✓ Les flux physiques ne contiennent malheureusement pas de données pertinentes à exploiter.
- ✓ Le TCO représente le taux de couverture en % du type de production par rapport à la production totale.
- ✓ Le TCH représente le taux de charge en % du type de production par rapport à sa puissance max.

## Jeux de données complémentaires

### **Température quotidienne régionale (depuis janvier 2016 à juin 2021)**

*Format : fichier csv*

<https://opendata.reseaux-energies.fr/explore/dataset/temperature-quotidienne-regionale>

Ce jeu de données présente les températures minimales, maximales et moyennes quotidiennes (en degré Celsius), par région administrative française, du 1er janvier 2016 à aujourd'hui.

Il est basé sur les mesures officielles du réseau de stations météorologiques françaises. La mise à jour de ce jeu de données est mensuelle.

Ce jeu de données contient 26494 lignes au total.

Chaque entrée regroupe les informations suivantes :

- Date
- Code INSEE Région
- Région
- Température Min(°C)
- Température Max(°C)
- Température Moy(°C)

### **Estimation de population par région, sexe et grande classe d'âge - Années 1975 à 2021**

*Format : fichier xlsx*

<https://www.insee.fr/fr/statistiques/1893198>

Ce jeu de données brutes présente l'estimation de population par région, sexe et classe d'âge. Les données sont organisées par années (1 onglet excel / 1 année).

Pour des facilités d'utilisation et les besoins du projet, nous avons regroupé toutes les données dont nous avions besoin dans un fichier csv pour plus de commodité.

Les données s'échelonnent de 2016 à 2021, pour une synchronisation avec les données de températures.

Le jeu de données « raffiné » pour la population se présente donc ainsi :

Région	2016	2017	2018	2019	2020	2021
Auvergne-Rhône-Alpes	7916889	7948287	7994459	8030533	8064146	8090442
Bourgogne-Franche-Comté	2818338	2811423	2807807	2801577	2794517	2784858
Bretagne	3306529	3318904	3335414	3347004	3358524	3371158
Centre-Val de Loire	2577866	2576252	2572853	2569510	2565726	2561451
Corse	330455	334938	338554	342256	345867	349269
Grand Est	5555186	5549586	5550389	5543407	5536002	5522476
Hauts-de-France	6006870	6003815	6004108	5995908	5987795	5975757
Île-de-France	12117132	12174880	12213447	12252917	12291557	12324261
Normandie	3335929	3330478	3327477	3320832	3313432	3305218
Nouvelle-Aquitaine	5935603	5956978	5979778	5999253	6018424	6039092
Occitanie	5808435	5845102	5885496	5918981	5951850	5985697
Pays de la Loire	3737632	3757600	3781423	3800348	3818421	3837166
Provence-Alpes-Côte d'Azur	5021928	5030890	5052832	5065696	5077582	5088998
France métropolitaine	64468792	64639133	64844037	64988222	65123843	65235843

## Démographie des entreprises et des établissements pour l'année X

<https://www.insee.fr/fr/statistiques?taille=100&debut=0&theme=38&categorie=3>

Les années 2016, 2017 et 2018 ont été récupérées et traitées pour une synchronisation avec le jeu de données températures.

De plus, les années 2019 et au-delà sont soit partielles, soit inexistantes.

Pour exemple – année 2016 (le traitement similaire a été appliqué à 2017 et 2018):

### [Démographie des entreprises et des établissements pour l'année 2016](#)

Fichier "Stocks d'établissements"

Format : fichiers dbf

Ces fichiers portent sur les établissements et les organismes en activité au 31 décembre 2015 dont le siège est situé en France métropolitaine et dans les départements d'outre-mer (y compris Mayotte).

Les fichiers de stocks d'établissements en activité au 31 décembre 2016 contiennent 3 107 674 enregistrements regroupant les caractéristiques des 6 067 881 établissements concernés.

La quantité d'information comprise dans ce fichier ne permet pas d'être traité avec Excel. En effet, Excel est limité à 1 048 576 lignes. Après quelques recherches, nous avons finalement réussi à traiter les données brutes avec Access en formulant des requêtes ensuite exploitablessous Excel afin de préparer un fichier csv.

Etapes de traitement :

- Ouverture et requête/filtrage Access
- Import sous Excel
- Traitement des données par région, secteur d'activité et par taille
- Jeu de données final enregistré en csv pour l'année 2016

Chaque enregistrement regroupe les informations suivantes :

- Région
- Sect\_Prim\_Micro : nombre d'entreprises de taille Micro dans le secteur Primaire
- Sect\_Prim\_PME : nombre de Petites et Moyennes Ent. dans le secteur Primaire
- Sect\_Prim\_ETI : nombre d'Ent. de Taille Intermédiaire dans le secteur Primaire
- Sect\_Prim\_GE : nombre de Grandes Entreprises dans le secteur Primaire
- Sect\_Sec\_Micro : nombre d'entreprises de taille Micro dans le secteur Secondaire
- Sect\_Sec\_PME : nombre de Petites et Moyennes Ent. dans le secteur Secondaire
- Sect\_Sec\_ETI : nombre d'Ent. de Taille Intermédiaire dans le secteur Secondaire
- Sect\_Sec\_GE : nombre de Grandes Entreprises dans le secteur Secondaire
- Sect\_Ter\_Micro : nombre d'entreprises de taille Micro dans le secteur Tertiaire
- Sect\_Ter\_PME : nombre de Petites et Moyennes Ent. dans le secteur Tertiaire
- Sect\_Ter\_ETI : nombre d'Ent. de Taille Intermédiaire dans le secteur Tertiaire
- Sect\_Ter\_GE : nombre de Grandes Entreprises dans le secteur Tertiaire

Pour rappel :

Taille : Micro (0 à 9 salariés), PME (10 à 249), ETI (250 à 4999), GE (5000 et au-delà)

Secteur primaire : Agriculture, sylviculture et pêche

Secteur secondaire : Industrie manufacturière, construction, industries extractives et autres

Secteur tertiaire : Commerce de tous types, information et communication, finance et assurances, immobilier, administration publique, enseignement, autres services, etc.

## Données géospatiales des régions en France

Les données ont été récupérées sur un repository maintenu par Grégoire David sur lequel nous avons téléchargé le fichier.

Fichier "regions-version-simplifiee"

Format : fichiers GeoJson

Les fichiers [GeoJSON](#) encodent les données géographiques des régions, et peuvent être facilement manipulés avec [geopandas](#).

Ci-dessous un aperçu des variables du fichier :

	code	nom	geometry
0	11	Île-de-France	POLYGON ((2.59052 49.07965, 2.63327 49.10838, ...
1	24	Centre-Val de Loire	POLYGON ((2.87463 47.52042, 2.88845 47.50943, ...
2	27	Bourgogne-Franche-Comté	POLYGON ((3.62942 46.74946, 3.57569 46.74952, ...
3	28	Normandie	POLYGON ((-1.11962 49.35557, -1.07822 49.38849...)
4	32	Hauts-de-France	POLYGON ((4.04797 49.40564, 4.03991 49.39740, ...

Les différentes variables sont les suivantes :

- code : correspond au code INSEE de chaque région de France
- nom : nom des différentes régions de France
- geometry : La géométrie de chaque département est stockée comme un polygone, avec la longitude et la latitude de chacun des points du polygone.

Voici un aperçu de la Ile-de-France, pour cela nous sélectionnons une ligne dans le dataframe qui nous permet d'évaluer la géométrie :

```
sf.loc[0].geometry
```



## IV. Dataviz Equilibre Consommation / Production

L'objectif de ce chapitre est de répondre à la question suivante :

- Comment est assuré l'équilibre entre consommation et production au niveau national et régional ?

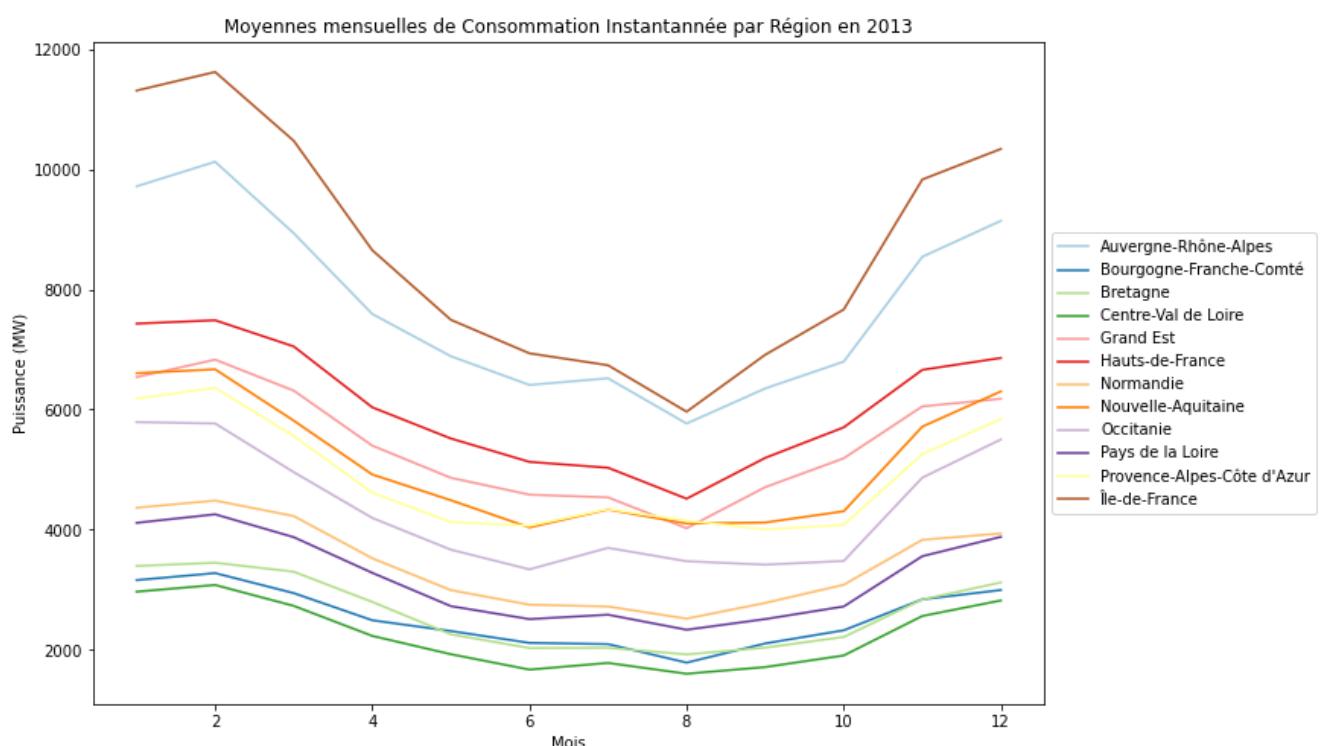
Pour cela nous avons tout d'abord effectué les actions suivantes sur le jeu de données initial :

- Importation du jeu de données
- Tri par Date-Heure et Région
- Suppression des colonnes inutiles (Flux, Code INSEE, Nature)
- Traitement de la variable Date-Heure avec pd.to\_datetime
- Remplacement des NaN par 0 sur les colonnes Productions et Consommation (l'absence de valeur sur ces colonnes peut s'interpréter par un 0)
- Création de la colonne Production qui additionne l'ensemble des productions par ligne (Date-Heure-Région)
- Création des colonnes Jour, Mois, Année pour de futurs affichages et/ou tris
- Création d'un dataframe par année (ex : Ener\_sorted\_2020)

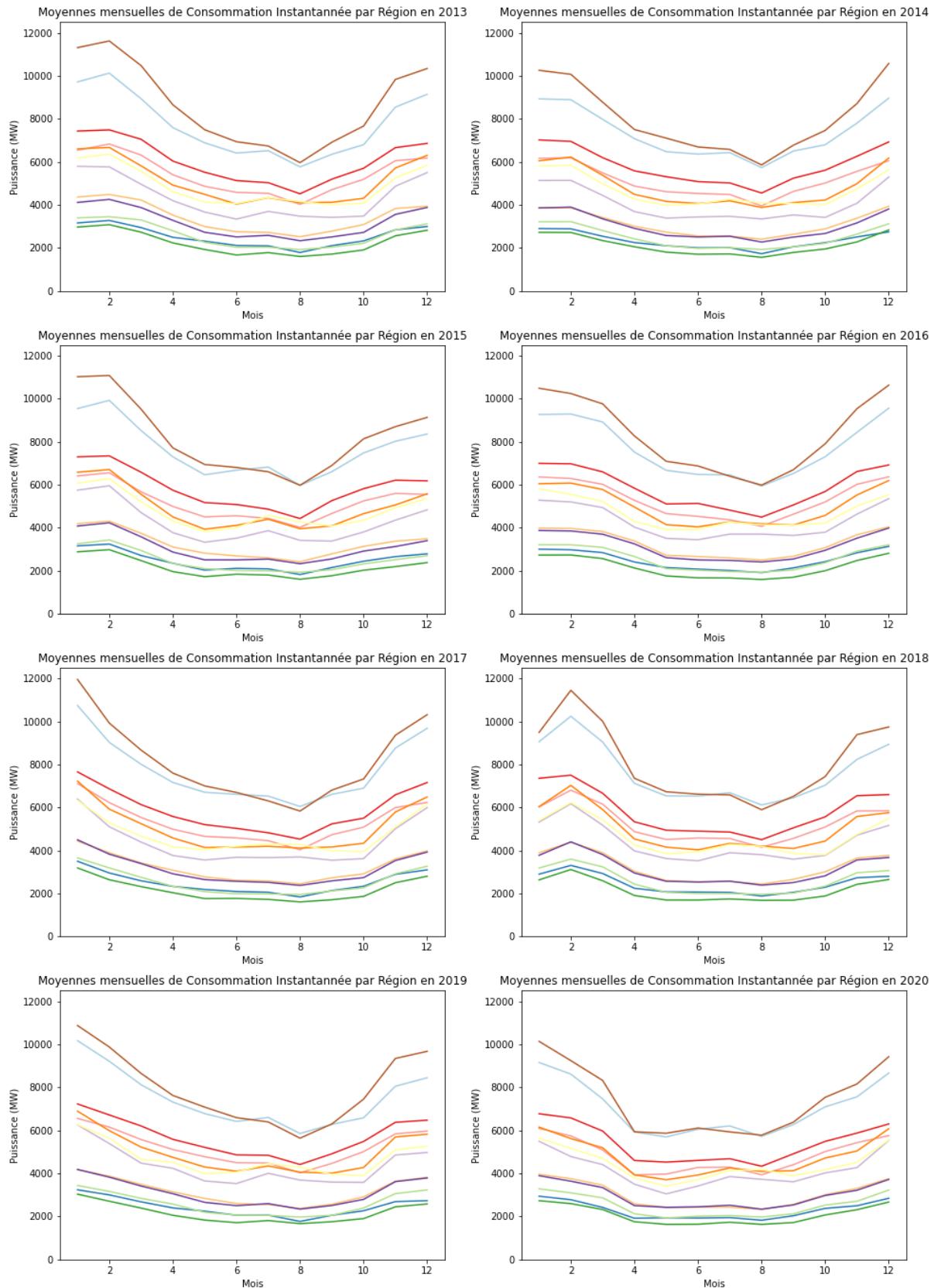
Nous pouvons à présent commencer à effectuer la Dataviz en analysant la Consommation.

### **Consommation Régionale (Puissance)**

Nous souhaitons tout d'abord observer les moyennes mensuelles de consommation instantanée par région en 2013



Nous obtenons une tendance reproduite sur toutes les régions. Observons maintenant la même information sur les autres années.



#### Observations :

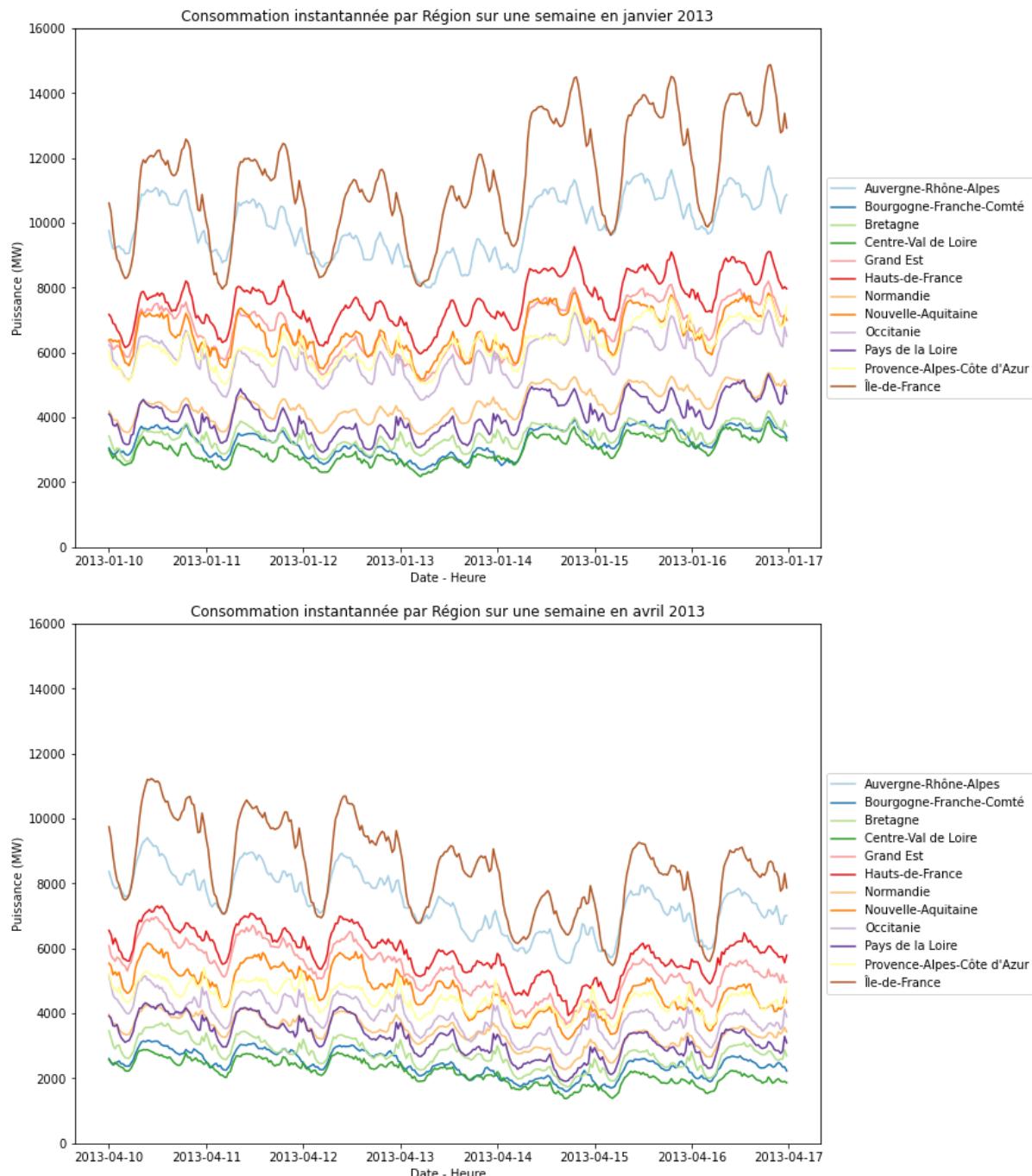
On note donc sur ces graphiques des variations saisonnières avec des valeurs hautes en hiver et basses en été.

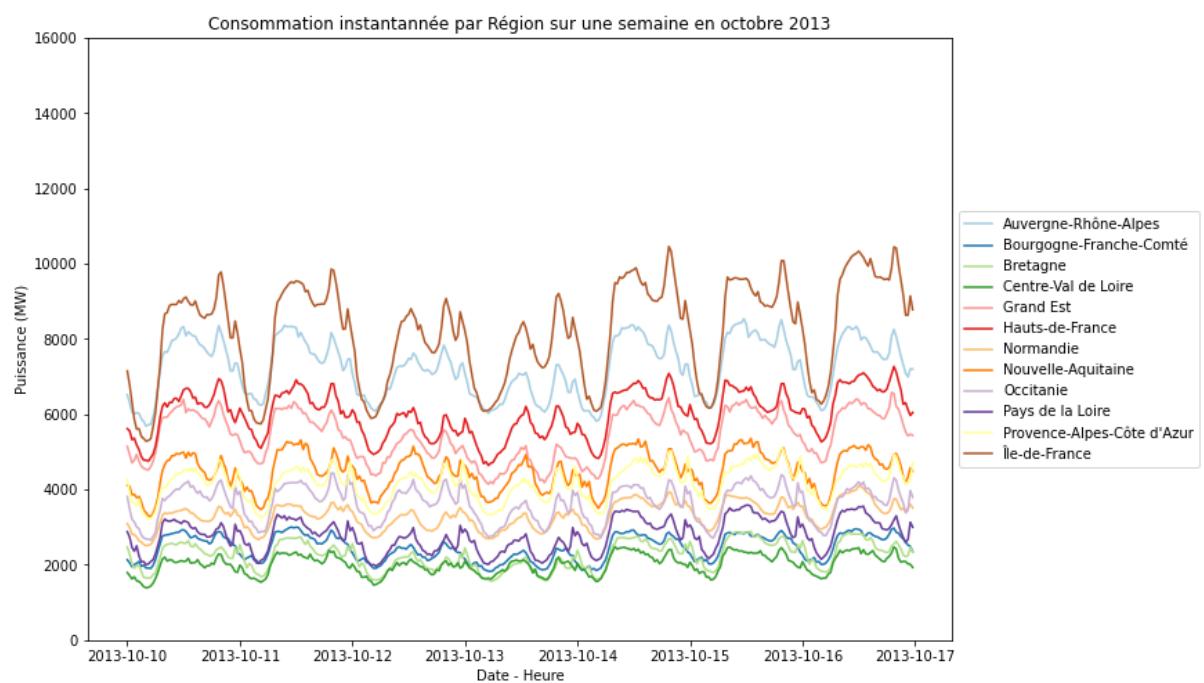
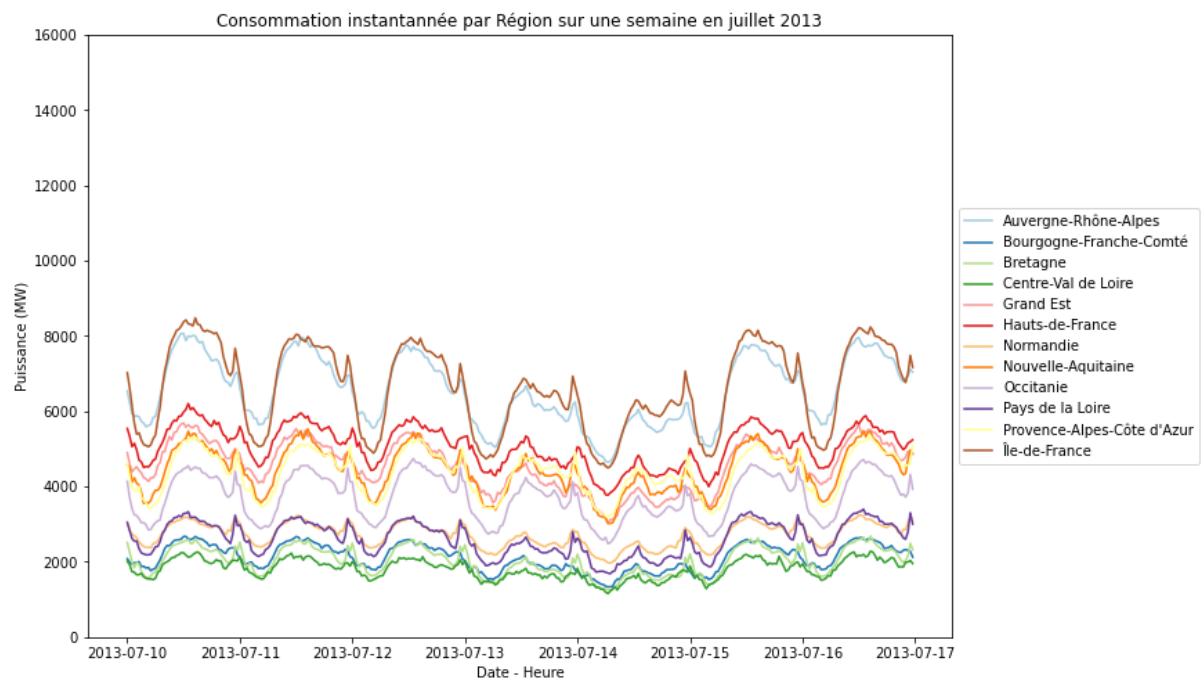
Ces variations de consommation s'expliquent ainsi :

- le besoin de chauffage en hiver chez les particuliers et dans les entreprises augmente la consommation d'électricité (en 2017, environ 40% de la population se chauffe avec de l'électricité : convecteurs, radiateurs à inertie, panneaux rayonnants, planchers chauffants, pompes à chaleur),
- la baisse de l'activité des entreprises (Industries et Tertiaires) en été pour les congés ; cumulée à l'absence de chauffage créent le creux caractéristique de la période d'été.

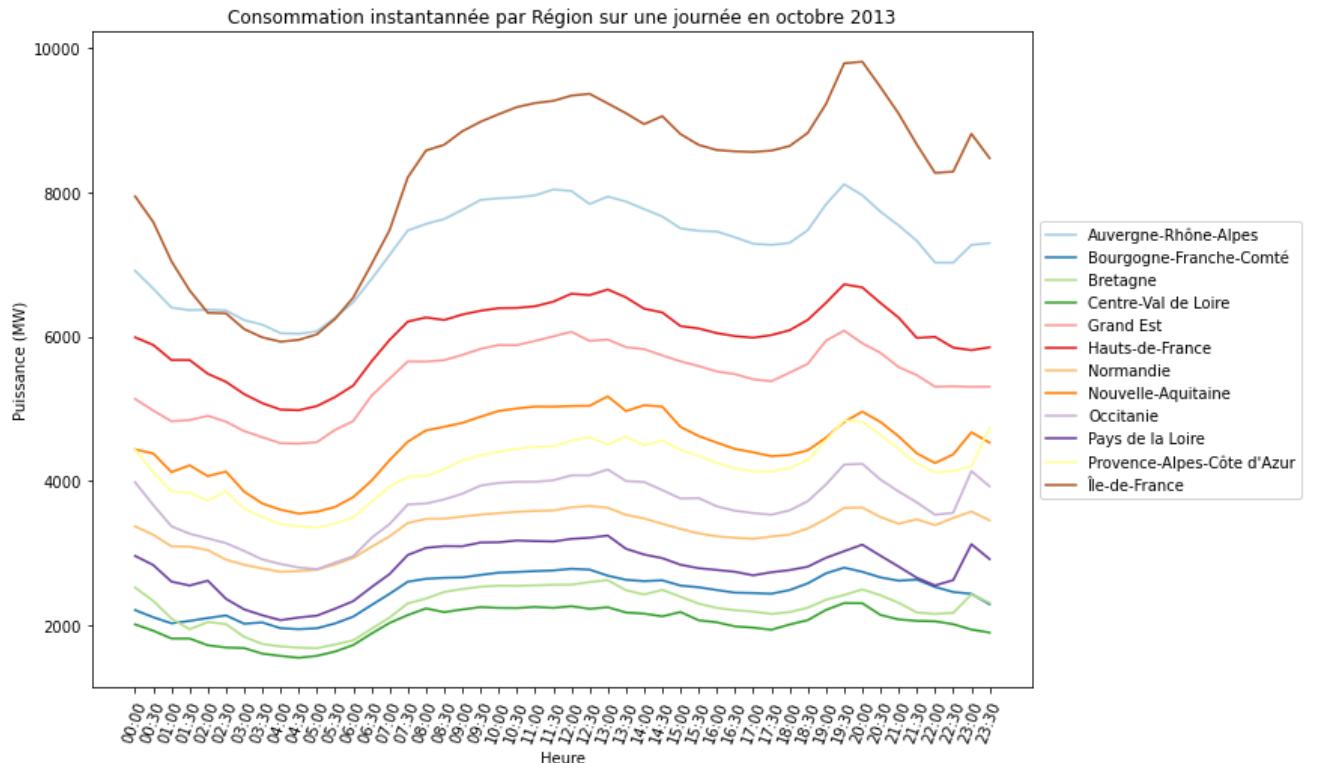
De plus, on observe sur le graphique de l'année 2020 les effets du confinement du mois de mi-mars produisant les effets de baisse d'activité comme ceux observés durant les congés d'été.

Nous allons maintenant observer la Consommation instantanée par région sur une semaine à plusieurs périodes de l'année.





Afin de compléter ces graphiques, nous allons zoomer sur 1 journée en Octobre



### Observations :

Quelle que soit la semaine observée dans l'année (janvier, avril, juillet et octobre) et quelle que soit la région, on observe un schéma récurrent de consommation quotidienne.

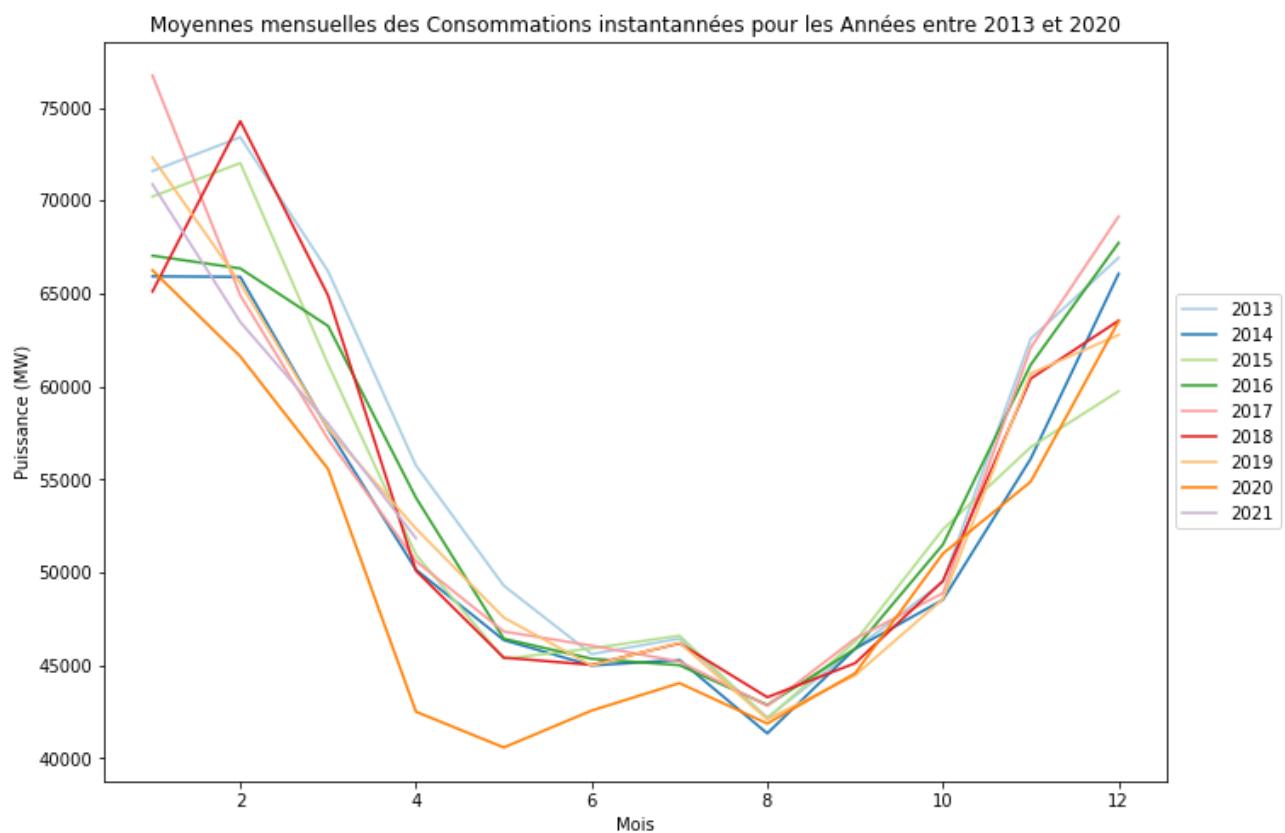
- ✓ Le minimum de consommation se situe aux alentours de 04h00 lorsque la majorité de la population dort.
- ✓ Ensuite la consommation croît fortement jusqu'à 08h00 ce qui correspond à la montée en charge du matin : le réveil et l'allumage des premiers appareils électriques et chauffages/chauffe-eau.
- ✓ Par la suite, la consommation croît plus modérément avec la mise en route des activités industrielles et de bureau jusqu'à un pic entre 12h00 et 13h00.
- ✓ Pour l'après-midi, la consommation décroît légèrement jusqu'aux environs de 18h00.
- ✓ Par la suite, le pic de consommation quotidien est observé entre 18h00 et 22h00 lorsque les français rentrent chez eux et allument alors simultanément le chauffage, la lumière et les autres appareils électroménagers (lave-linge, lave-vaisselle, télévision, ...)
- ✓ Un dernier pic apparaît aux alentours de 23h00 et correspond probablement au lancement d'appareils de type lave-linge, sèche-linge et lave-vaisselle pendant les heures « creuses ».

### **Consommation Nationale (Puissance)**

Nous souhaitons maintenant observer les moyennes mensuelles de consommation instantanée nationale sur plusieurs années

Pour ce faire, nous réalisons un nouveau dataframe à l'aide d'un groupby sommant la consommation.

```
Ener_Conso_Nat = Ener_sorted.groupby(["Année","Mois","Jour","Heure"])["Consommation (MW)"].sum().reset_index()
```



#### Observations :

On note une certaine stabilité dans la consommation instantanée moyennée par mois.

On retrouve cependant l'effet du confinement et de l'arrêt de l'activité économique qui apparaît clairement à partir de mars sur la courbe de l'année 2020.

L'analyse de la quantité d'énergie (MW.h) consommée par mois pourra nous donner une indication plus précise sur la tendance annuelle.

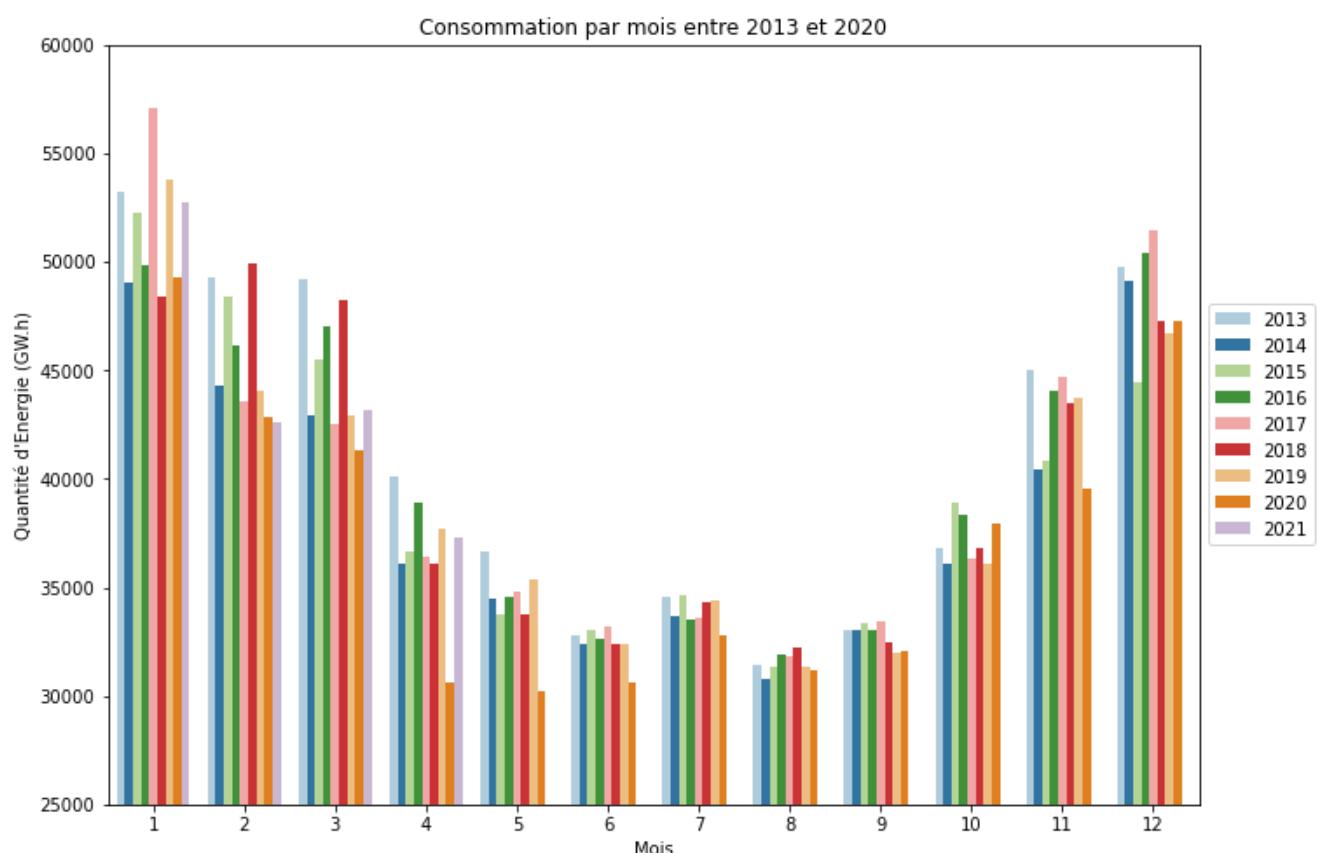
## **Quantité d'Energie - Consommation Nationale**

La Quantité d'Energie correspond à ce qui se trouve sur la facture de votre fournisseur, c'est-à-dire des kW.h. Cela correspond à l'intégration à l'heure de la Puissance Instantanée mesurée.

Pour calculer cette quantité, nous utilisons un groupby sommant la consommation que nous divisons par 2 (intégration car 1 valeur toutes les ½ heures) puis par 1000 pour convertir les kW.h en MW.h.

```
QEner_Conso_Nat_M      =      Ener_sorted.groupby(["Année","Mois"])["Consommation  
(MW)"].sum()/2/1000
```

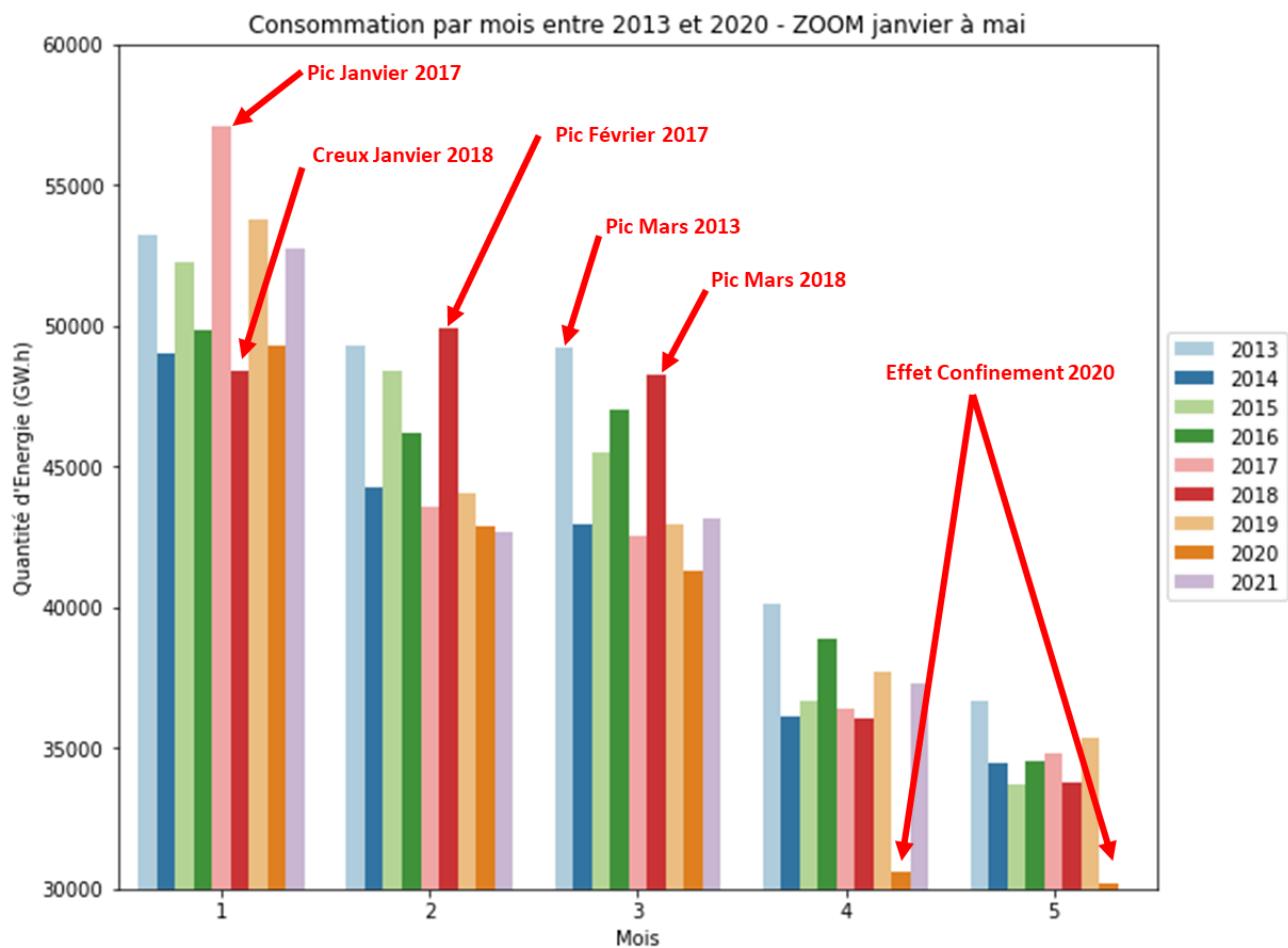
Nous pouvons donc afficher la Quantité d'Energie consommée par mois entre 2013 et 2020



### Observations :

On observe toujours le même profil de consommation quelle que soit l'année observée.

Nous proposons de réaliser un zoom afin d'expliquer plus en détails quelques effets observables.



#### ***Effet Confinement COVID-19 :***

On note un écart important à la baisse sur les mois d'avril et mai 2020 par rapport aux mêmes mois des autres années.

#### ***Effet Température :***

Pic sur Janvier 2017

archive meteo-france : <http://www.meteo-france.fr/actualites/45600575-janvier-2017-froid-et-peu-arrose>

« Les températures ont été 2 à 4 °C inférieures aux normales excepté sur les régions bordant la Manche et la Méditerranée. Les gelées ont été très fréquentes sur l'ensemble du pays notamment dans le Nord-Est, avec plus de 20 jours de gel. Du 17 au 26 janvier, les températures sont restées particulièrement glaciales, avec des journées sans dégel particulièrement sur le quart nord-est. Comme en 2009 et 2010, ce mois de janvier a été froid avec une température moyenne inférieure de 1.9 °C aux normales. »

Pic sur Février 2018

archive meteo-france : <https://www.meteocontact.fr/climatologie/france/fevrier-2018>

« Ce mois de février a été bien froid, plus ensoleillé au nord et maussade au sud que la normale et inégal en termes de précipitations. Le fait marquant de ce mois reste les températures particulièrement basses notamment à la fin février. Sur l'ensemble du pays, les températures ont donc été anormalement basses. En Corse, cette anomalie est moins

prononcée car on a mesuré -0.1°C à Ajaccio (2A), -0.5°C à l'Île-Rousse (2B) et -0.65°C à Figari (2A). Mais on a constaté l'effet inverse à Limoges (87) qui atteint -3.45°C, à Orléans (45) et Creil (60) avec -3.25°C, et à Aurillac (15), à Châteaudun (41) et à Pontoise (95) avec -3.2°C. De telles températures font de ce mois de février le plus froid depuis 2012. »

### Pic sur Mars 2013

archive meteo-france : <https://meteofrance.com/magazine/meteo-histoire/les-grands-evenements/11-au-15-mars-2013-des-chutes-de-neige-exceptionnelles>

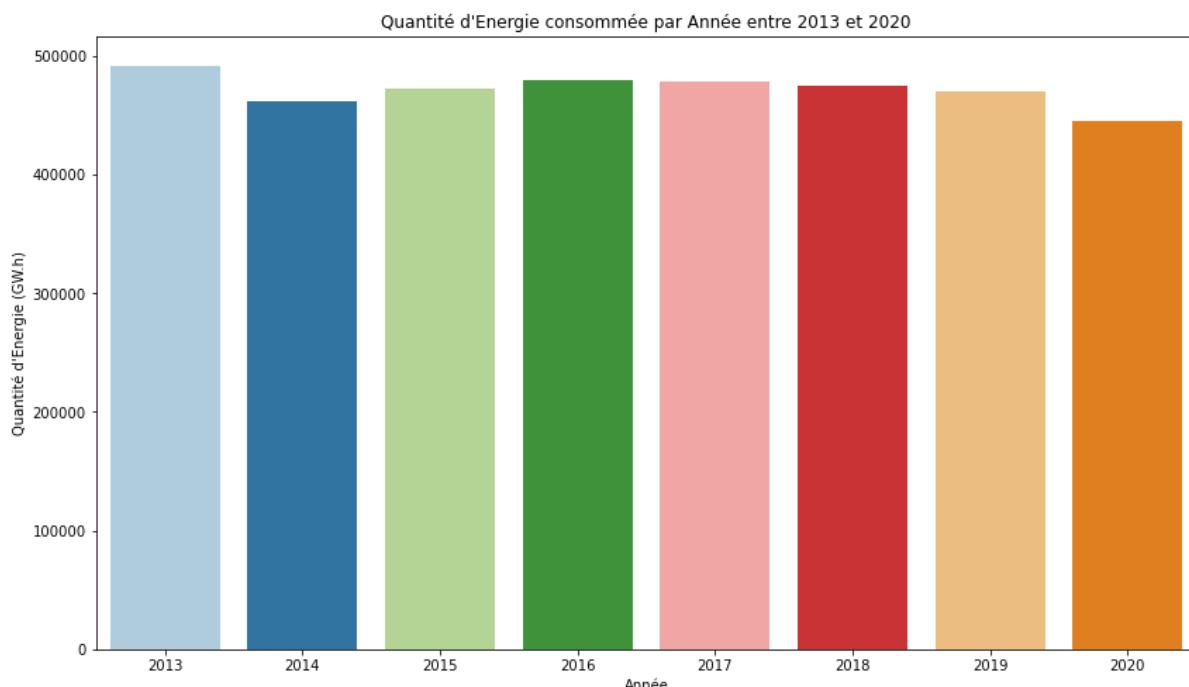
« Un froid record : un épisode hivernal tardif est survenu du 11 au 15 mars et a concerné la quasi-totalité du pays. Les températures ont battu des records de froid sur un grand quart nord-ouest de la France, de la Normandie à la Beauce, à la Champagne-Ardenne et au Nord - Pas-de-Calais, avec des températures localement inférieures à -10 °C le 13 mars. Des records mensuels de températures minimales ont été localement battus. »

### Pic sur Mars 2018

archive meteo-france : <http://www.meteofrance.fr/actualites/61051700-mars-2018-agite>

« Durant ce mois de mars particulièrement maussade, l'hiver a joué les prolongations avec de fréquents passages perturbés et des épisodes de neige tardifs. La température, en moyenne de 8,2 °C sur le mois et sur le pays, a été inférieure à la normale\* de 0,5 °C. Ce mois de mars a été plus froid que janvier, qui avait bénéficié d'une douceur exceptionnelle avec 8,4 °C en moyenne sur le pays. Voir consommation de janvier exceptionnellement basse. »

Nous affichons maintenant la Quantité d'Energie consommée par année entre 2013 et 2020.



### Observations :

On note une certaine stabilité dans la consommation avec peut-être une légère baisse tendancielle depuis 2017 qu'il reste cependant à confirmer. Cette baisse pourrait être due aux programmes d'économies d'énergies lancées ces dernières années.

On note aussi un écart plus important en 2020 avec l'effet COVID 19.

Enfin, l'année 2014 apparaît très en dessous des autres années hors COVID avec une

possible explication du côté des Températures : <http://www.meteofrance.fr/climat-passe-et-futur/bilans-climatiques/bilan-2014/bilan-climatique-provisoire-de-l-annee-2014>

« 2014 : une année exceptionnellement chaude La température moyenne annuelle en 2014 sur la France a dépassé de 1,2 °C la normale\*, positionnant l'année au premier rang des années les plus chaudes depuis 1900, devant 2011 (+ 1.1 °C) et 2003 (+ 1.0 °C). »

## Production Nationale vs Consommation

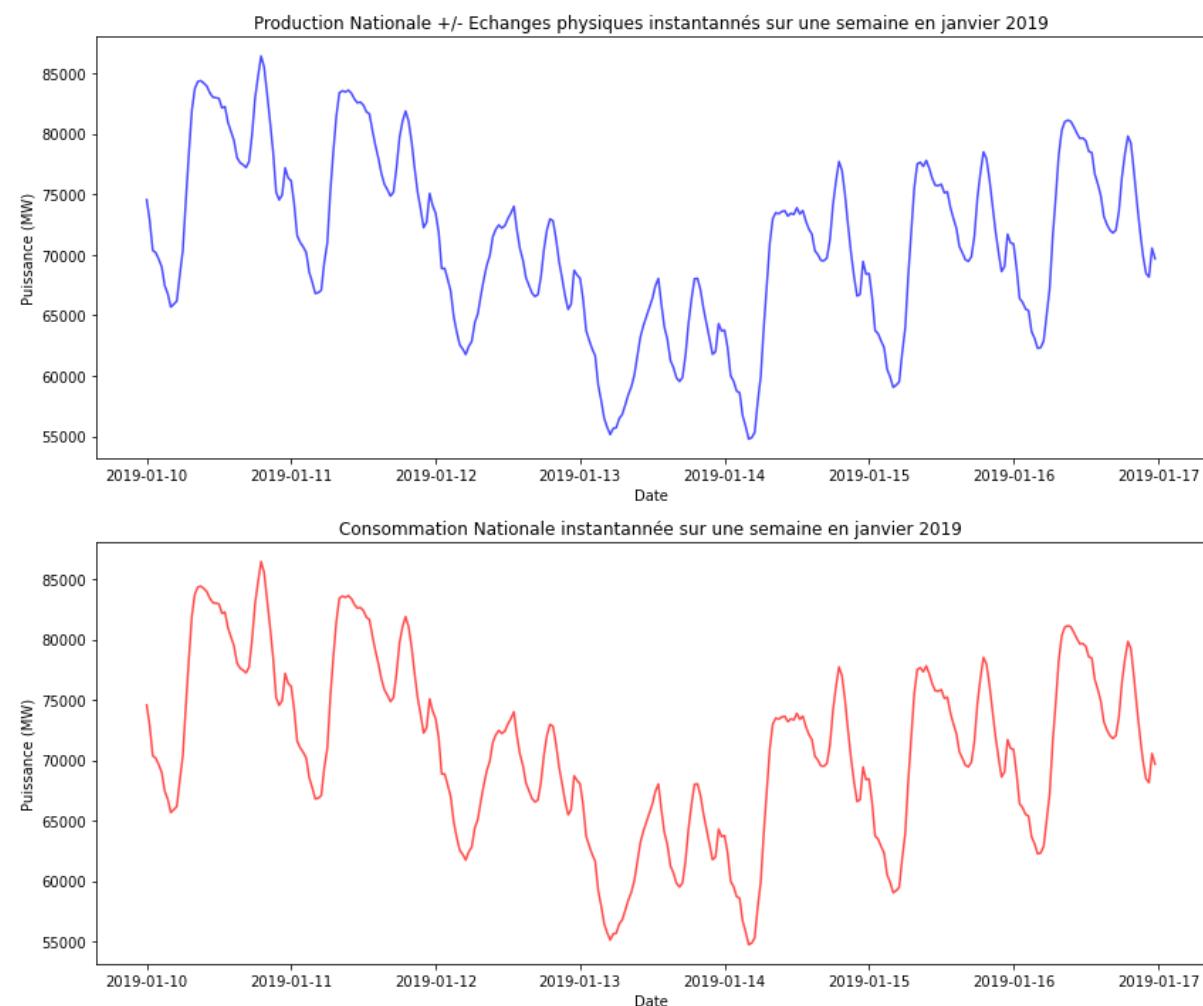
Nous allons maintenant observer la Production Nationale +/- les échanges physiques au regard de la Consommation.

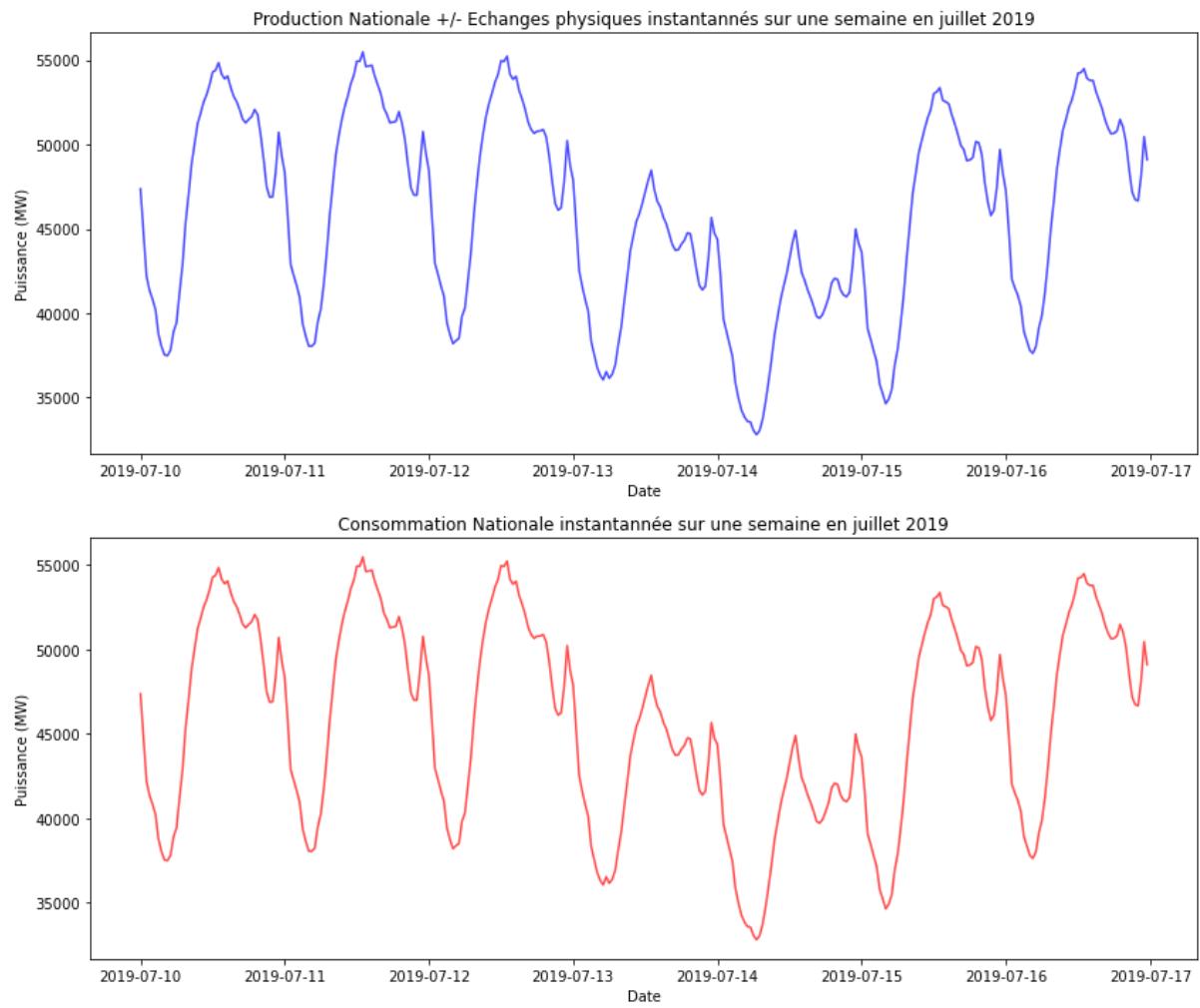
Pour ce faire, nous appliquons un groupby sommant les variables sur la Date et l'Heure

`Ener_Prod_Nat = Ener_sorted.groupby(["Date", "Heure"]).sum().reset_index()`

Puis nous créons un dataframe par année (ex : `Ener_Prod_Nat_2019`).

Afin d'établir le comportement de la Production +/- les échanges physiques au regard de la Consommation, nous choisissons d'afficher ces variables sur 1 semaine en janvier et en juillet 2019.





#### Observations :

La Production Nationale +/- les échanges physiques avec les pays limitrophes et la Consommation Nationale sont parfaitement synchronisées. L'équilibre est géré continuellement en fonction des prévisions et de la demande réelle.

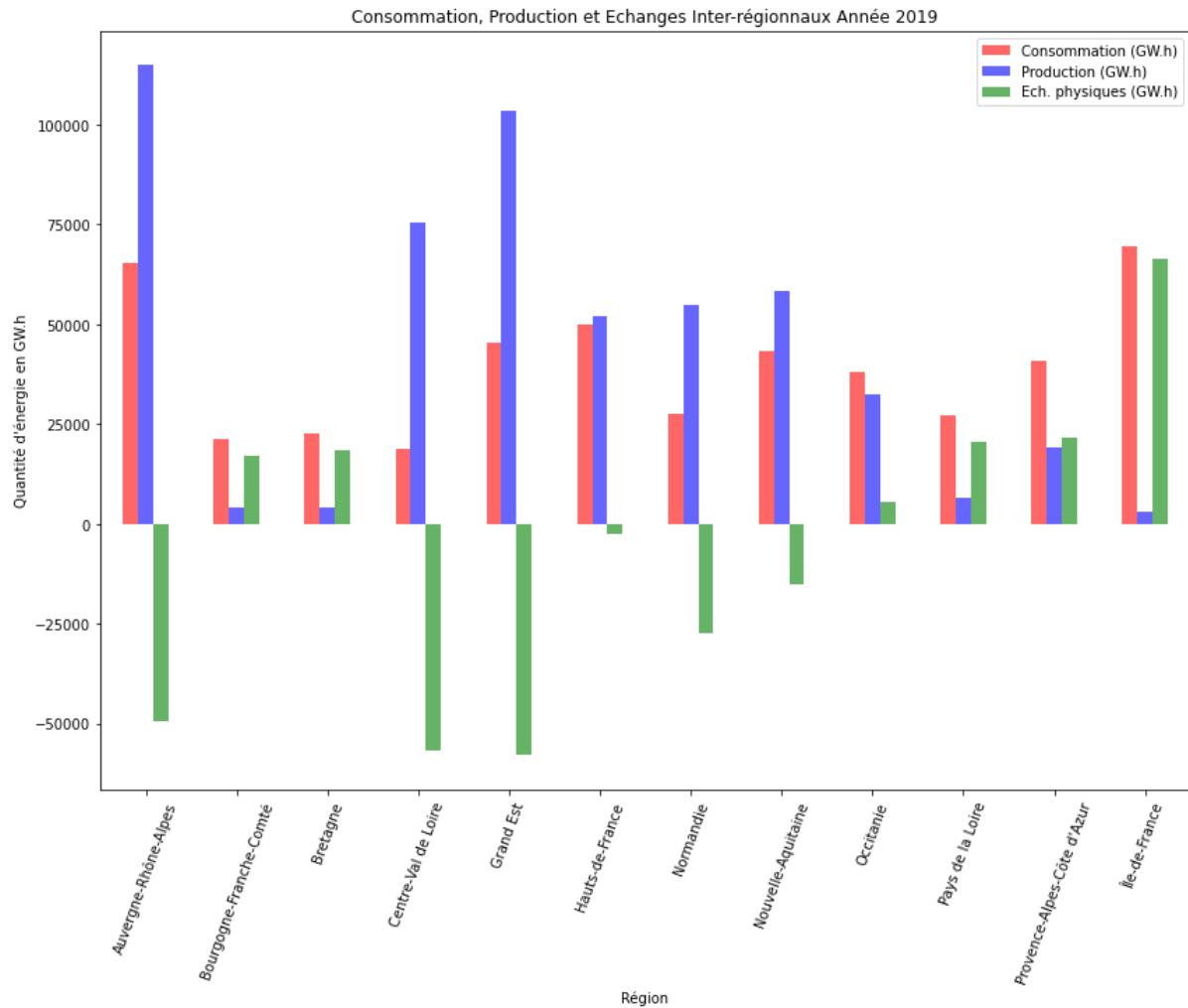
Sur une journée, la consommation nationale instantanée peut varier de plus de 15000 MW sans que cela ne pose de problème pour la production ; ce qui démontre des mécanismes d'équilibrage efficaces et des moyens de production flexibles.

## Production vs Consommation par Région

Nous proposons de visualiser les quantités d'énergie produites et consommées annuellement par Région ainsi que les échanges physiques, le tout en GW.h.

Pour ce faire, nous appliquons un groupby sommant les variables sur chaque Région.

`QEner_2019_som = QEner_2019_temp.groupby('Région').sum()/2/1000`



### Observations :

On observe 3 grandes classes de Région en France :

- Les régions « consommatrices » produisant peu ou très peu comme Bourgogne-Franche-Comté, Bretagne, Pays de la Loire, Provence-Alpes-Côte d'Azur, et surtout Île-de-France.
- Les régions « équilibrées » comme Hauts-de-France et Occitanie
- Les régions « Productrices » distribuant leurs excédents comme Auvergne-Rhône-Alpes, Centre-Val de Loire, Grand Est, Normandie et Nouvelle-Aquitaine.

La position géographique de la Bretagne combinée à sa faible production explique qu'elle soit identifiée comme un point faible du réseau électrique impliquant des coupures locales en cas de problème sur le réseau breton.

# V. Modèles de prédition de la Consommation

L'objectif de ce chapitre est donc de répondre à notre deuxième problématique :

- Sommes-nous capables de prédire correctement la consommation avec un(des) modèle(s) de machine learning afin de prévoir les besoins de production ?

Pour cela, nous proposons plusieurs approches d'algorithme de machine learning :

- Un modèle d'analyse de séries temporelles SARIMA
- Plusieurs modèles de régression en commençant par un Ridge et un Lasso, pour terminer sur un ElasticNet

## A. Modèle SARIMA

Pourquoi un SARIMA ?

Nous avons choisi de travailler sur un modèle de séries temporelles pour pouvoir analyser dans notre série la dépendance dans le temps et le comportement saisonnier qui peut apparaître.

Cela fait suite aux éléments de datavisualisation que nous avons vu précédemment.

Nous avons clairement constaté un effet saisonnier dans notre série.

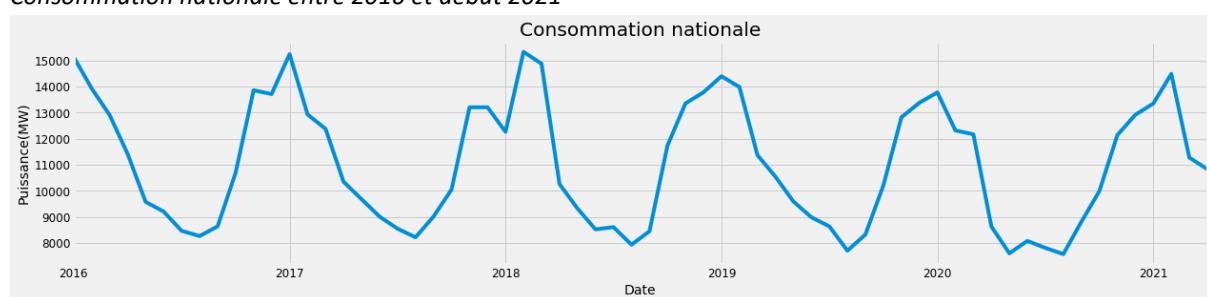
On peut grapher ci-dessous nos données brutes de consommation entre 2016 et 2021.

Nous avons choisi de restreindre le jeu de données en partant de l'année 2016.

Plus précisément il s'agit, ici de la représentation de la consommation nationale maximale mensuelle en méga watt, plus pertinente que la consommation moyenne mensuelle si l'on souhaite établir une prédition utile pour anticiper les besoins et la disponibilité des installation.

### • Etude de la série chronologique

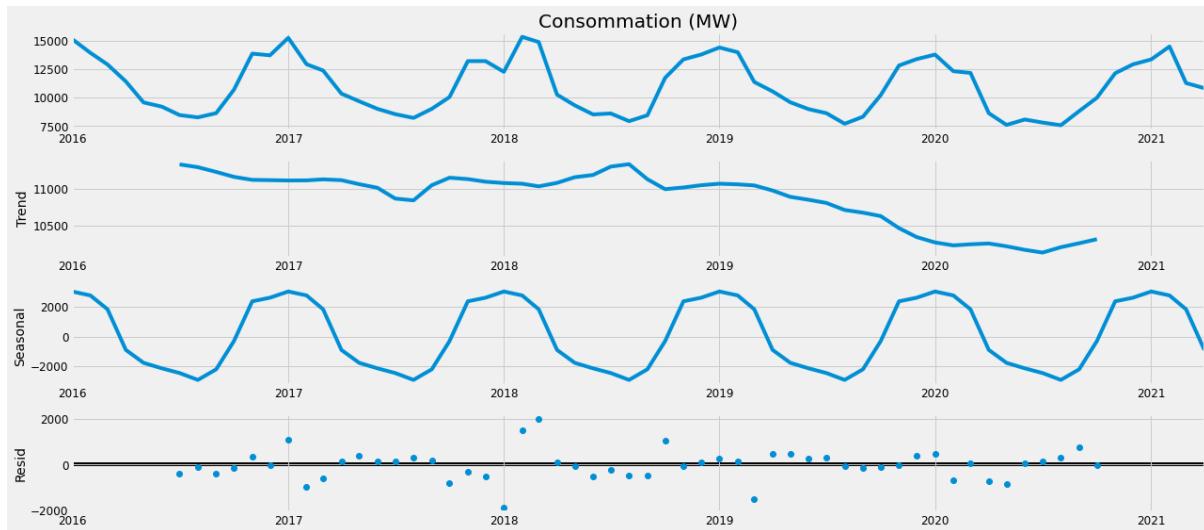
*Consommation nationale entre 2016 et début 2021*



### Observations :

En analysant le graphique, nous pouvons observer que la série chronologique présente une saisonnalité annuelle. Les pics s'observent en début et fin d'année (période hivernale).

Nous pouvons décomposer la série chronologique en trois composantes distinctes : la tendance, la saisonnalité et le bruit à l'aide de la commande « `sm.tsa.seasonal_decompose` » de la bibliothèque pylab :



### Observations :

Ici, la décomposition est additive, en effet on n'observe pas de tendance claire et continue dans le temps, à la hausse ou à la baisse.

Aussi, on confirme la présence d'une saisonnalité annuelle.

Le choix du modèle SARIMA ( Seasonal Autoregressive Integrated Moving Average) permet de contrôler la saisonnalité en l'incluant directement en tant que caractéristique (variable X) dans notre modèle au lieu de transformer notre variable Y en changement annuel.

### • Définition du modèle SARIMA

Nous pouvons utiliser l'implémentation de SARIMA par statsmodels :

```
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(Ener_model_series, order=param, seasonal_order=param_seasonal,
                                              enforce_stationarity=False, enforce_invertibility=False)
            results = mod.fit()
            print('SARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

(pdq correspond à une liste de paramètres entre 0 et 2)

Les arguments clés de la fonction SARIMAX sont:

- Les données brutes , stockées dans un dataframe appelé Ener\_model\_series.  
Ici nous avons fait un « resample des données » pour avoir une « series » contenant uniquement la dimension de temps et la variable de consommation à l'aide de la commande suivante :  
`Ener_model=Ener_model.resample('MS', on='Date').max()`

Les données temporelles sont « resamplées » en partant du début de chaque mois (paramètre *MS :month start*) en prenant la puissance maximale de chaque mois.

Les données sont ainsi stockées dans une « series » panda pour pouvoir appliquer un modèle de séries temporelles, ici le SARIMA

```
Date
2016-01-01    15148.0
2016-02-01    13934.0
2016-03-01    12912.0
2016-04-01    11421.0
2016-05-01    9578.0
2016-06-01    9207.0
2016-07-01    8467.0
2016-08-01    8267.0
2016-09-01    8638.0
2016-10-01    10694.0
Freq: MS, Name: Consommation (MW), dtype: float64
```

- L'argument `seasonal_order` est similaire à `order`, sauf qu'il sert à spécifier la saisonnalité.

Ici, nous avons une partie de la sortie des différentes combinaisons de paramètres et donc différents modèles avec les critères AIC pour chaque modèle testé.

```
SARIMA(0, 1, 1)x(0, 0, 12)12 - AIC:1069.8804630673058
SARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:859.3264306462654
SARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:847.7591878934422
SARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:604.7949299175158
SARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:885.9964527657263
SARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:825.0352341986597
SARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:651.0587718535857
SARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:606.8150231422378
```

Le critère AIC (critère d'information d'Akaike) est un estimateur de la qualité relative des modèles statistiques pour un ensemble de données. Le critère AIC estime la qualité de chaque modèle, par rapport à chacun des autres modèles. Plus la valeur de l'AIC est faible plus le modèle sera pertinent. Notre sortie suggère que SARIMAX (0, 1, 1)x(0, 1, 1, 12) avec une valeur AIC de 604.79 est la meilleure combinaison, nous considérerons cette option comme optimale.

- **Ajustement du modèle**

En ajustant les paramètres précédents à notre modèle nous obtenons les résultats suivants :

```

SARIMAX Results
=====
Dep. Variable: Consommation (MW) No. Observations: 64
Model: SARIMAX(0, 1, 1)x(0, 1, 1, 12) Log Likelihood -299.397
Date: Mon, 23 Aug 2021 AIC 604.795
Time: 15:56:01 BIC 609.628
Sample: 01-01-2016 HQIC 606.499
- 04-01-2021
Covariance Type: opg
=====

            coef    std err        z      P>|z|      [0.025]     [0.975]
--- 
ma.L1     -1.1143   0.112    -9.956    0.000     -1.334    -0.895
ma.S.L12  -0.8517   0.356    -2.392    0.017     -1.549    -0.154
sigma2    3.77e+05  2.08e+05   1.812    0.070    -3.08e+04  7.85e+05
--- 

Ljung-Box (L1) (Q): 0.00 Jarque-Bera (JB): 0.24
Prob(Q): 1.00 Prob(JB): 0.89
Heteroskedasticity (H): 1.04 Skew: 0.03
Prob(H) (two-sided): 0.95 Kurtosis: 3.39
=====
```

### Observations :

Les paramètres du modèle sont significatifs. Nous devons maintenant vérifier que le résidu est un bruit blanc et qu'il est distribué normalement.

Les tests statistiques nous indiquent les propriétés des résidus :

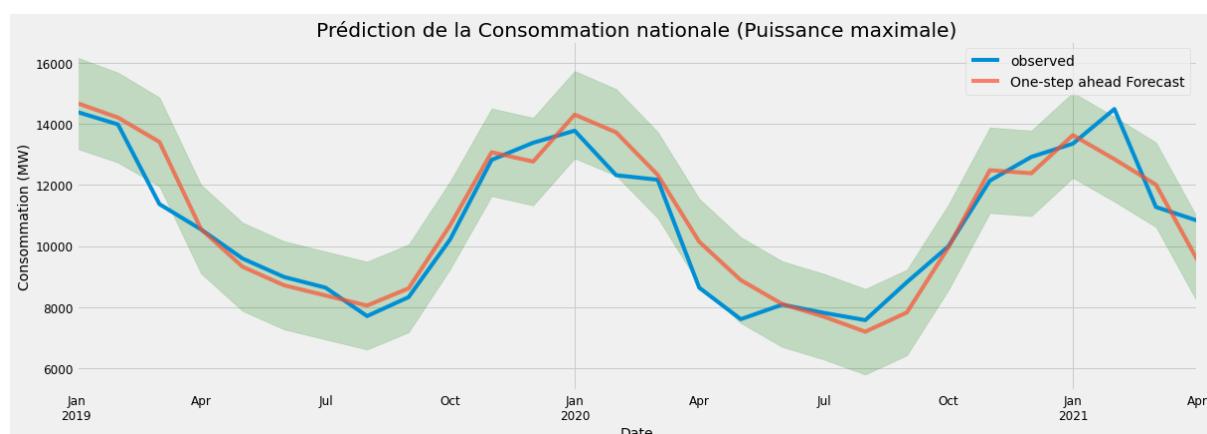
Le test de [Ljung-Box](#) est un test de blancheur. C'est un test statistique qui vise à rejeter ou non l'hypothèse H<sub>0</sub> : Le résidu est un bruit blanc. Ici on lit sur la ligne Prob(Q) que la p-valeur de ce test est de 1, donc on ne rejette pas l'hypothèse.

Le test de [Jarque-Bera](#) est un test de normalité. C'est un test statistique qui vise à rejeter ou non l'hypothèse H<sub>0</sub> : Le résidu suit une distribution normale. Ici on lit sur la ligne Prob (JB) que la p-valeur du test est de 0.89. On ne rejette donc pas l'hypothèse.

Le résidu vérifie les hypothèses que l'on a faites à priori. On peut donc conclure que le modèle SARIMA(0,1,1)(0,1,1)12 est satisfaisant.

Nous allons maintenant utiliser ce modèle pour faire une prédition sur la consommation d'électricité. Les prédictions s'effectuent à l'aide de la méthode [predict](#) appliquée à un modèle ajusté.

Cette étape consiste à comparer les valeurs réelles avec les prédictions prévues.



La RMSE est de 770.9

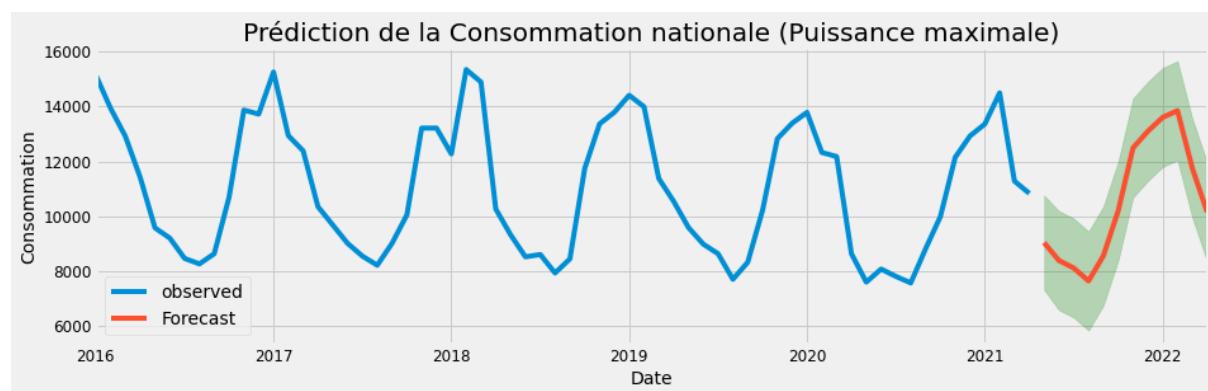
#### Observations :

Nos prédictions correspondent très bien aux vraies valeurs. La commande "pred = results.get\_prediction(start=pd.to\_datetime('2019-01-01'))" détermine la période qu'on prévoirait en comparant avec les données réelles.

Dans le graphique suivant, nous prévoyons la consommation d'électricité pour les 12 prochains mois.

#### Observations :

Le pic annuel que l'on retrouve en période hivernale se retrouve sur l'année 2022. Il s'agit ici d'un « forecast », nous faisons une prédition sur l'année 2022 en nous basant sur les données de notre série temporelle.



#### Conclusion :

Notre modèle SARIMA a de bonnes performances et nous arrivons à de bonnes prédictions de la consommation d'électricité en France.

## B. Modèles de Régression régularisée

L'approche par Régression nous semble être une deuxième approche possible au regard des data que nous avons pu récupérer en jeux de données complémentaires.

En effet, nous avons pu voir l'impact de la température sur des variations de consommation. Nous supposons également que les données de population influencent la consommation dans la mesure où l'électricité est consommé pour des besoins humains.

A cela s'ajoute enfin le profil économique d'une région qui peut se traduire par la représentation des établissements d'entreprises en fonction de leur taille et leur secteur d'activité dans le sens où l'activité économique est très probablement liée à la consommation d'électricité.

### Que nous apporte une approche par Régression régularisée ?

Nous avons vu que la régularisation permet d'éviter le sur-apprentissage d'une régression linéaire en rajoutant à la fonction objective (la somme des carrés des erreurs) un terme de régularisation qui mesure la complexité du modèle.

En effet les méthodes de régularisation peuvent améliorer l'erreur de prédiction du modèle en réduisant la variabilité des estimations du coefficient de régression à l'aide d'une réduction des estimations tendant vers 0.

La régression régularisée équilibre la même minimisation de la somme des erreurs au carré avec un terme de pénalité sur la taille des coefficients et tend à produire des modèles plus simples qui sont moins enclins au surajustement.

Elle est donc utilisée pour améliorer la stabilité, réduire l'impact de la colinéarité et améliorer l'efficacité et la généralisation du calcul.

### Etape d'intégration des data pour former un dataframe unique

Tout d'abord nous traitons le dataframe initial :

- Import du fichier 'eco2mix-regional-cons-def.csv'
- Tri par Date, Heure, Région
- Suppression des colonnes des Production, Echanges, Flux, TCOs et TCHs
- Traitement des NaN

Pour arriver au résultat dataframe **Ener\_light** ci-dessous.

	Région	Date	Heure	Date - Heure	Consommation (MW)
0	Auvergne-Rhône-Alpes	2013-01-01	00:00	2013-01-01T00:00:00+01:00	0.0
1	Bourgogne-Franche-Comté	2013-01-01	00:00	2013-01-01T00:00:00+01:00	0.0

Ensuite nous préparons les dataframe complémentaires.

- Importation des données de température dans le datframe **Temp\_per\_day**.

	Date	Région	TMin (°C)	TMax (°C)	TMoy (°C)
0	2016-01-01	Auvergne-Rhône-Alpes	2.96	10.56	6.76
1	2016-01-01	Bourgogne-Franche-Comté	2.22	9.52	5.87
2	2016-01-01	Bretagne	3.00	12.27	7.64
3	2016-01-01	Centre-Val de Loire	1.52	10.73	6.12
4	2016-01-01	Corse	9.95	15.10	12.52

- Importation des données de population dans le datframe **Pop\_per\_year** avec ensuite l'application de la fonction melt car les colonnes représentaient les années.

	Région	Année	Population
0	Auvergne-Rhône-Alpes	2016	7916889
1	Bourgogne-Franche-Comté	2016	2818338
2	Bretagne	2016	3306529
3	Centre-Val de Loire	2016	2577866
4	Corse	2016	330455
5	France métropolitaine	2016	64468792

- Importation des données entreprises dans 3 dataframes (1 par année) puis concaténation pour obtenir plus qu'un seul dataframe **Ent**.

	Région	Année	Sect_Prim_Micro	Sect_Prim_PME	Sect_Prim_ETI	Sect_Prim_GE	Sect_Sec_Micro	Sect_Sec_PME	Sect_Sec_ETI	Sect_Sec_GE
0	Auvergne-Rhône-Alpes	2016	82294	224	2	0	116792	11355	272	1
1	Bourgogne-Franche-Comté	2016	38683	192	2	0	34867	3725	95	1
2	Bretagne	2016	41964	259	1	0	37801	4092	104	0
3	Centre-Val de Loire	2016	30868	132	2	0	30032	3165	111	0
4	Grand Est	2016	62576	249	2	0	64204	6644	196	1

	Sect_Ter_Micro	Sect_Ter_PME	Sect_Ter_ETI	Sect_Ter_GE
	493816	35827	682	3
	140904	11153	217	2
	165130	13708	272	1
	120209	10464	214	0
	268042	22263	462	3

Tous les jeux de données sont importés. Il nous faut donc les finalisés et les fusionnés.

- Récupération des pics de puissance quotidien sur la consommation avec un groupby max en créant le dataframe **Energy**
- Insertion de l'année à partir de la Date
- Sélection des data sur les années 2016 à 2018 pour la synchronisation avec les autres dataframes (notamment les entreprises)

	Date	Année	Région	Consommation (MW)
0	2016-01-01	2016	Auvergne-Rhône-Alpes	8283.0
1	2016-01-01	2016	Bourgogne-Franche-Comté	2658.0
2	2016-01-01	2016	Bretagne	3344.0
3	2016-01-01	2016	Centre-Val de Loire	2446.0
4	2016-01-01	2016	Grand Est	5401.0
...	...	...	...	...
13147	2018-12-31	2018	Nouvelle-Aquitaine	7208.0
13148	2018-12-31	2018	Occitanie	6339.0
13149	2018-12-31	2018	Pays de la Loire	3970.0
13150	2018-12-31	2018	Provence-Alpes-Côte d'Azur	6502.0
13151	2018-12-31	2018	Île-de-France	10388.0

13152 rows × 4 columns

Nous procédons maintenant à la fusion des données en prenant comme référence du merge les enregistrements du dataframe initial.

- Fusion du dataframe **Temp\_per\_day** avec **Energy** dans **data\_fusion**
- Fusion du dataframe **Pop\_per\_year** avec **data\_fusion** dans **data\_fusion\_2**
- Fusion du dataframe **Ent** dans **data\_fusion\_2** dans **data**
- Suppression des variables Année, Tmin et Tmax

Le dataframe ***data*** est prêt pour les modèles :

	Date	Région	Consommation (MW)	TMoy (°C)	Population	Sect_Prim_Micro	Sect_Prim_PME	Sect_Prim_ETI	Sect_Prim_GE
0	2016-01-01	Auvergne-Rhône-Alpes	8283.0	6.76	7916889	82294	224	2	0
1	2016-01-01	Bourgogne-Franche-Comté	2658.0	5.87	2818338	38683	192	2	0
2	2016-01-01	Bretagne	3344.0	7.64	3306529	41964	259	1	0
3	2016-01-01	Centre-Val de Loire	2446.0	6.12	2577866	30868	132	2	0
4	2016-01-01	Grand Est	5401.0	4.06	5555186	62576	249	2	0
Sect_Sec_Micro	Sect_Sec_PME	Sect_Sec_ETI	Sect_Sec_GE	Sect_Ter_Micro	Sect_Ter_PME	Sect_Ter_ETI	Sect_Ter_GE		
116792	11355	272	1	493816	35827	682	3		
34867	3725	95	1	140904	11153	217	2		
37801	4092	104	0	165130	13708	272	1		
30032	3165	111	0	120209	10464	214	0		
64204	6644	196	1	268042	22263	462	3		

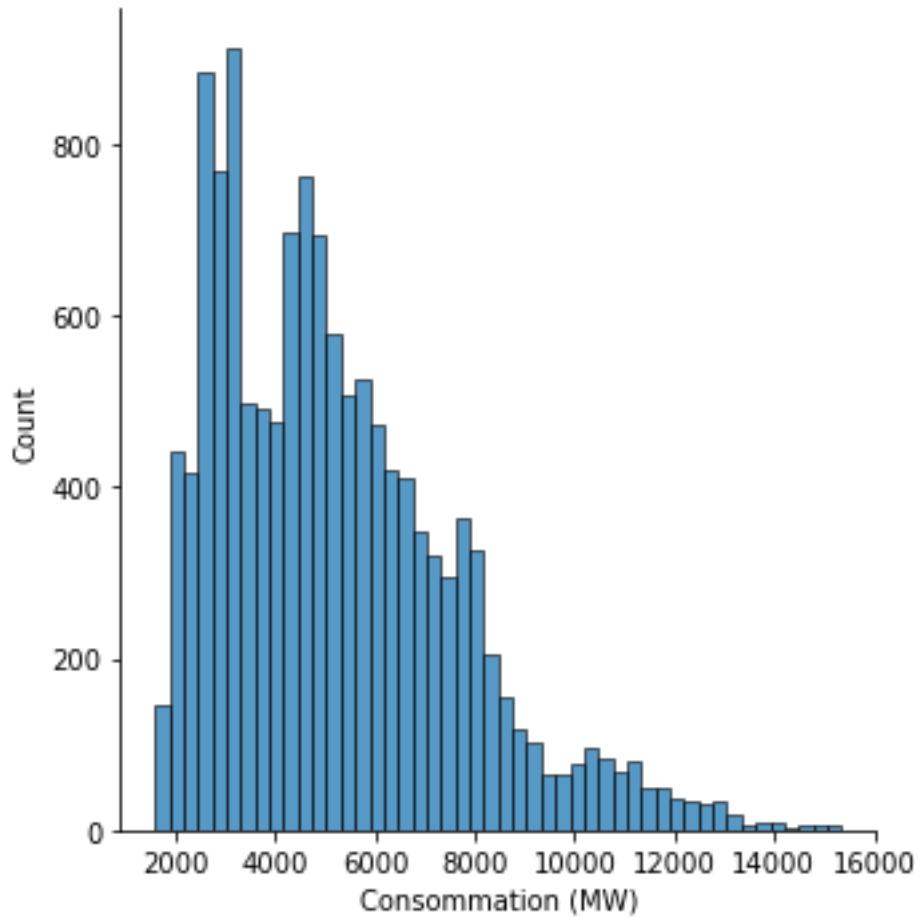
Vérification des formats et NaN du dataframe ***data***:

```

Int64Index: 13152 entries, 0 to 13151
Data columns (total 17 columns):
 #   Column           Non-Null Count Dtype
 ---  -----
 0   Date             13152 non-null  object
 1   Région          13152 non-null  object
 2   Consommation (MW) 13152 non-null  float64
 3   TMoy (°C)        13152 non-null  float64
 4   Population       13152 non-null  int64
 5   Sect_Prim_Micro 13152 non-null  int64
 6   Sect_Prim_PME    13152 non-null  int64
 7   Sect_Prim_ETI    13152 non-null  int64
 8   Sect_Prim_GE     13152 non-null  int64
 9   Sect_Sec_Micro   13152 non-null  int64
 10  Sect_Sec_PME    13152 non-null  int64
 11  Sect_Sec_ETI    13152 non-null  int64
 12  Sect_Sec_GE     13152 non-null  int64
 13  Sect_Ter_Micro   13152 non-null  int64
 14  Sect_Ter_PME    13152 non-null  int64
 15  Sect_Ter_ETI    13152 non-null  int64
 16  Sect_Ter_GE     13152 non-null  int64

```

Nous visualisons la distribution de la variable cible et nous observons une distribution bimodale :



Avec un extract du min et du max pour observer l'amplitude de la variable cible à prédire.

la consommation min est de : 1597.0  
la consommation max est de : 15338.0

## 1. Modèle Ridge

La régression Ridge est une méthode d'ajustement de modèle qui est utilisée pour analyser les données qui souffrent de multicollinéarité, corrélations entre les variables prédictives. Cette méthode effectue une régularisation L2, norme euclidienne. Lorsque le problème de la multicollinéarité se pose, les moindres carrés ne sont pas biaisés et les variances sont importantes, ce qui fait que les valeurs prédictives sont très éloignées des valeurs réelles.

C'est donc un moyen de créer un modèle parcimonieux lorsque le nombre de variables prédictrices dans un ensemble dépasse le nombre d'observations.

Dans la régression Ridge, la première étape consiste à normaliser les variables (dépendantes et indépendantes) en soustrayant leurs moyennes et en les divisant par leurs écarts types. Cela pose un problème de notation puisque nous devons indiquer d'une manière ou d'une autre si les variables d'une formule particulière sont normalisées ou non.

En ce qui concerne la normalisation, tous les calculs de régression Ridge sont basés sur des variables normalisées. Lorsque les coefficients de régression finaux sont affichés, ils sont réajustés dans leur échelle d'origine. Cependant, la trace de la crête est sur une échelle standardisée.

```
# Normalisation
scaler = preprocessing.StandardScaler()

features[features.columns] = pd.DataFrame(scaler.fit_transform(features), index=features.index)
target_scaled = scaler.fit_transform(target)
```

Nous devons à présent trouver un coefficient de régularisation adapté. Pour rappel, l'objectif est de biaiser un peu la prédiction, afin de diminuer l'erreur standard.

On appelle ce coefficient alpha ; nous devons en tester un certain nombre afin de trouver celui qui est optimal.

Nous pouvons tester toutes les régressions Ridges avec les différentes valeurs de l'hyperparamètre  $\alpha$  et récupérer les poids des différents coefficients de la régression associées ainsi que l'erreur quadratique : la valeur de alpha diminue les poids de tous les paramètres de la régression, ce qui signifie qu'ils ne sont pas appris automatiquement par le modèle, mais qu'ils doivent être définis manuellement.

Nous effectuons une recherche sur grille pour trouver les valeurs alpha optimales.

En effet, pour obtenir le meilleur modèle prédictif possible, il est préférable d'utiliser la classe RidgeCV (Cross Validation) à qui l'on donne dans le paramètre alphas une liste de valeurs pour  $\alpha$  que la fonction parcourra pour créer et évaluer plusieurs modèles par validation croisée, puis sélectionner le  $\alpha$  entraînant les meilleures performances.

```
from sklearn.linear_model import RidgeCV

ridge_reg = RidgeCV(alphas= [0.001, 0.01, 0.1, 0.3, 0.7, 1, 10, 50, 100])
ridge_reg.fit(X_train, y_train)

RidgeCV(alphas=array([1.e-03, 1.e-02, 1.e-01, 3.e-01, 7.e-01, 1.e+00, 1.e+01, 5.e+01,
1.e+02]))
```

Le compromis entre le biais et la variance est généralement compliqué lorsqu'il s'agit de construire des modèles de régression Ridge sur un ensemble de données réelles. Cependant, la tendance générale que l'on doit retenir est la suivante :

- Le biais augmente lorsque  $\lambda$  augmente.
- La variance diminue lorsque  $\lambda$  augmente.

Nous obtenons donc le  $\alpha$  retenu par le modèle grâce à l'attribut `alpha_` et également le score du modèle sur l'ensemble d'entraînement puis sur l'ensemble de test.

```
print('alpha sélectionné par c-v :', ridge_reg.alpha_)
print('score train :', ridge_reg.score(X_train, y_train))
print('score test :', ridge_reg.score(X_test, y_test))

alpha sélectionné par c-v : 0.01
score train : 0.9137455443512946
score test : 0.9134069504892897
```

En Stockant dans `ridge_pred_train` les valeurs ajustées du modèle et dans `ridge_pred_test` les prédictions du modèle pour `X_test`, nous pouvons afficher l'erreur quadratique moyenne (MSE) de prédiction pour `ridge_pred_train` et `ridge_pred_test`, grâce à la fonction `mean_squared_error`.

```
ridge_pred_train = ridge_reg.predict(X_train)
ridge_pred_test = ridge_reg.predict(X_test)

print('mse train :', mean_squared_error(ridge_pred_train, y_train))
print('mse test :', mean_squared_error(ridge_pred_test, y_test))

mse train : 0.08608326412842206
mse test : 0.0872148926808324
```

Ainsi avec un  $\alpha$  sélectionné de 0.01 par exemple nous obtenons un score de 0.9137 avec l'échantillon train pour un MSE de 0.086 et un score de 0.9134 avec l'échantillon test pour un MSE de 0.087.

En calculant les moyennes et écart-types nous pouvons obtenir et afficher nos prédictions avec le modèle RIDGE.

```
# création des variables moyenne et écart-type
target = pd.DataFrame(np.c_[data['Consommation (MW)']], columns = ['Consommation (MW)'], index=data.index)
scaler = preprocessing.StandardScaler().fit(target)
moy = scaler.mean_[-1]
ec = scaler.scale_[-1]
print('moyenne :', moy)
print('écart-type :', ec)

moyenne : 5204.196243917275
écart-type : 2396.648593119176

# Affichage des consommations observées et des consommations prédictes par le RIDGE

pd.DataFrame({'Consommations_observées_(MW)' : (ridge_y_test_resh*ec)+moy,
              'Consommations_prédites_(MW)' : (ridge_pred_test_2*ec)+moy},
              index=X_test.index).head(50)
```

	Consommations_observées_(MW)	Consommations_predites_(MW)
2017-11-21 - Centre-Val de Loire	2955.0	2928.454460
2017-03-28 - Pays de la Loire	3770.0	4020.517078
2017-11-26 - Bretagne	3437.0	4080.891342
2018-05-26 - Provence-Alpes-Côte d'Azur	4477.0	4701.574885
2018-12-10 - Bretagne	3668.0	3691.433664
2016-02-28 - Bretagne	3764.0	3976.881665
2016-03-30 - Île-de-France	10854.0	9900.433302
2016-04-26 - Hauts-de-France	6989.0	7300.193617
2016-08-14 - Centre-Val de Loire	1644.0	1071.740695
2016-12-25 - Hauts-de-France	6082.0	6843.177829
2016-10-14 - Île-de-France	10200.0	9616.067923
2017-05-20 - Île-de-France	7355.0	9601.768883
2017-12-25 - Pays de la Loire	3835.0	4604.481696
2017-04-03 - Centre-Val de Loire	2413.0	2419.389651
2017-04-09 - Pays de la Loire	2936.0	3284.213863
2017-10-24 - Provence-Alpes-Côte d'Azur	4715.0	5237.838733
2017-02-07 - Auvergne-Rhône-Alpes	10932.0	9422.637070
2017-03-17 - Bretagne	3090.0	3457.572586
2017-03-23 - Provence-Alpes-Côte d'Azur	5303.0	5526.012578
2018-01-31 - Île-de-France	11580.0	10176.985237
2017-11-08 - Auvergne-Rhône-Alpes	10237.0	9543.238459
2017-04-11 - Provence-Alpes-Côte d'Azur	4744.0	5307.660590
2018-04-26 - Île-de-France	8070.0	9760.593074
2017-01-30 - Bourgogne-Franche-Comté	3723.0	3833.693501

## 2. Modèle Lasso

La régression Lasso est une technique de régularisation. Elle est utilisée sur les méthodes de régression pour une prédiction plus précise. Ce modèle utilise le rétrécissement.

Cela consiste à réduire les valeurs des données vers un point central comme la moyenne. La procédure lasso favorise les modèles simples et épars (c'est-à-dire les modèles comportant moins de paramètres).

Ce type particulier de régression est bien adapté aux modèles présentant des niveaux élevés de multicollinéarité ou lorsque l'on souhaite automatiser certaines parties de la sélection du modèle, comme la sélection des variables/élimination des paramètres.

La régression Lasso utilise la technique de régularisation L1. Elle est utilisée lorsque nous avons un plus grand nombre de caractéristiques car elle effectue automatiquement la sélection des caractéristiques.

Nous créons un modèle de régression Lasso, `lasso_r`, avec  $\alpha=1$  (valeur par défaut sur scikit-learn), ajusté sur les données d'entraînement.

```
# TEST LASSO
from sklearn.linear_model import Lasso

lasso_r = Lasso(alpha=1)

lasso_r.fit(X_train, y_train)
```

Lasso(alpha=1)

Les coefficients estimés étant récupérables, comme pour toute régression linéaire, nous pouvons afficher les coefficients estimés par le modèle.

Tous les coefficients étant nuls, nous déduisons que la valeur  $\alpha=1.0$  ne convient pas à nos données.

```
: lasso_r.coef_
array([-0.,  0.,  0., -0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.])
```

$\lambda$  dénote la quantité de rétrécissement.

$\lambda = 0$  implique que toutes les caractéristiques sont considérées et c'est équivalent à la régression linéaire où seule la somme résiduelle des carrés est considérée pour construire un modèle prédictif.

$\lambda = \infty$  implique qu'aucune caractéristique n'est considérée, c'est-à-dire que lorsque  $\lambda$  se rapproche de l'infini, il élimine de plus en plus de caractéristiques.

Le biais augmente avec l'augmentation de  $\lambda$  la variance augmente avec la diminution de  $\lambda$

Nous créons donc un nouveau modèle de régression Lasso, lasso\_reg, avec  $\alpha=0.1$  (valeur par défaut sur scikit-learn), ajusté sur les données d'entraînement.

```
lasso_reg = Lasso(alpha=0.1)

lasso_reg.fit(X_train, y_train)

Lasso(alpha=0.1)
```

En stockant dans lasso\_pred\_train les valeurs ajustées du modèle et dans lasso\_pred\_test les prédictions du modèle pour X\_test , nous pouvons afficher l'erreur quadratique moyenne de prédiction pour lasso\_pred\_train et lasso\_pred\_test, grâce à la fonction mean\_squared\_error.

Nous pouvons donc obtenir le score du modèle sur l'ensemble d'apprentissage qui est de 0.849 pour l'échantillon train avec un MSE de 0.150 et celui de l'ensemble de test qui est de 0.846 pour l'échantillon test avec un MSE de 0.154.

```
: print('score train :', lasso_reg.score(X_train, y_train))
print('score test :', lasso_reg.score(X_test, y_test))

score train : 0.8496275602847103
score test : 0.846641219327567

: lasso_pred_train = lasso_reg.predict(X_train)
lasso_pred_test = lasso_reg.predict(X_test)

print('mse train :', mean_squared_error(lasso_pred_train, y_train))
print('mse test :', mean_squared_error(lasso_pred_test, y_test))

mse train : 0.15007399152070117
mse test : 0.15446008280786128
```

Nous allons utiliser la fonction lasso\_path permettant de produire les coefficients estimés correspondants aux différents  $\alpha$  qu'elle reçoit en arguments.

Elle prend comme argument les données de l'échantillon d'apprentissage et la variable cible associée et retourne les  $\alpha$  testés dans l'ordre décroissant, ainsi que la matrice des coefficients de taille  $p \times m$  où  $p$  désigne le nombre de variables explicatives, et  $m$  le nombre de valeurs de  $\alpha$  essayées.

La fonction de scikit-learn permet de sélectionner automatiquement un jeu de valeurs  $\alpha$  à tester, mais il est possible également de les spécifier explicitement dans l'argument alphas.

```
from sklearn.linear_model import LassoPath

mes_alphas = (0.001, 0.01, 0.02, 0.025, 0.05, 0.1, 0.25, 0.5, 0.8, 1.0)

alpha_path, coefs_lasso, _ = LassoPath(X_train, y_train, alphas=mes_alphas)

coefs_lasso.shape

(1, 13, 10)
```

Nous cherchons à savoir laquelle des différentes valeurs de  $\alpha$  est la plus performante en prédiction.

Il existe, tout comme pour la régression Ridge, une classe LassoCV qui permet de trouver le  $\alpha$  optimal en fonction de la performance prédictive, par validation croisée.

La fonction retourne les  $\alpha$  testés dans l'attribut alphas\_, le  $\alpha$  retenu dans l'attribut alpha\_, mais également dans l'attribut model.mse\_path\_ la matrice contenant pour chaque  $\alpha$  la valeur du MSE pour chaque échantillon de cross-validation obtenu.

```
from sklearn.linear_model import LassoCV

model_lasso = LassoCV(cv=10).fit(X_train, y_train)

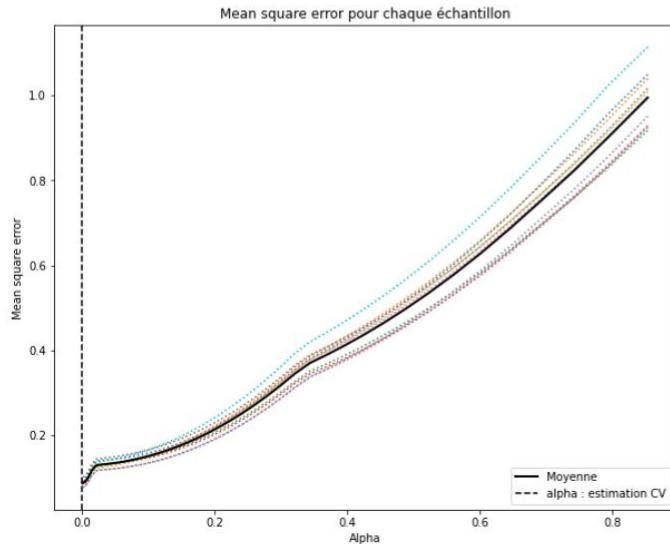
alphas = model_lasso.alphas_
```

```
plt.figure(figsize=(10, 8))

plt.plot(alphas, model_lasso.mse_path_, ':')
plt.plot(alphas, model_lasso.mse_path_.mean(axis=1), 'k', label='Moyenne', linewidth=2)

plt.axvline(model_lasso.alpha_, linestyle='--', color='k', label='alpha : estimation CV')

plt.xlabel('Alpha')
plt.ylabel('Mean square error')
plt.title('Mean square error pour chaque échantillon')
plt.legend();
```



Nous obtenons une courbe représentant les erreurs MSE en fonction des valeurs de  $\alpha$  pour chaque échantillon de la validation croisée. Une courbe représentant la moyenne des scores MSE sur tous les échantillons, en fonction des valeurs de  $\alpha$  testés , et une ligne verticale sur la valeur de  $\alpha$  sélectionnée par le modèle.

Nous obtenons donc le MSE de 0.088 obtenu sur l'échantillon de test par le modèle model\_lasso avec un score de 0.912.

```
pred_test = model_lasso.predict(X_test)

print('score test :', model_lasso.score(X_test, y_test))
print('mse test :', mean_squared_error(pred_test, y_test))

score test : 0.912288745132827
mse test : 0.08834112810861874
```

En procédant de la même manière qu'avec notre modèle RIDGE et en calculant les moyennes et écart-types nous pouvons obtenir et afficher nos prédictions avec le modèle LASSO.

	Consommations_observées_(MW)	Consommations_predites_(MW)
2017-11-21 - Centre-Val de Loire	2955.0	2997.064301
2017-03-28 - Pays de la Loire	3770.0	4045.981455
2017-11-26 - Bretagne	3437.0	4198.031678
2018-05-26 - Provence-Alpes-Côte d'Azur	4477.0	4575.488454
2018-12-10 - Bretagne	3668.0	3748.766946
2016-02-28 - Bretagne	3764.0	4086.574676
2016-03-30 - Île-de-France	10854.0	9930.032201
2016-04-26 - Hauts-de-France	6989.0	7266.313689
2016-08-14 - Centre-Val de Loire	1644.0	1118.660303
2016-12-25 - Hauts-de-France	6082.0	6810.963714
2016-10-14 - Île-de-France	10200.0	9646.703328
2017-05-20 - Île-de-France	7355.0	9733.472106
2017-12-25 - Pays de la Loire	3835.0	4627.817534
2017-04-03 - Centre-Val de Loire	2413.0	2489.855024
2017-04-09 - Pays de la Loire	2936.0	3312.362052
2017-10-24 - Provence-Alpes-Côte d'Azur	4715.0	5232.088630
2017-02-07 - Auvergne-Rhône-Alpes	10932.0	9475.917181
2017-03-17 - Bretagne	3090.0	3576.984907
2017-03-23 - Provence-Alpes-Côte d'Azur	5303.0	5519.212086
2018-01-31 - Île-de-France	11580.0	10075.744959
2017-11-08 - Auvergne-Rhône-Alpes	10237.0	9596.078980
2017-04-11 - Provence-Alpes-Côte d'Azur	4744.0	5301.655987
2018-04-26 - Île-de-France	8070.0	9660.870538
2017-01-30 - Bourgogne-Franche-Comté	3723.0	3825.526760
2018-02-24 - Île-de-France	12045.0	10842.250749

Généralement la régression Lasso est une technique de régularisation utilisée pour une prédiction plus précise.

Le Lasso est considéré comme meilleur que le Ridge car il ne sélectionne que certaines caractéristiques et réduit à zéro les coefficients des autres.

La régression Lasso utilise le rétrécissement, où les valeurs des données sont réduites vers un point central tel que la valeur moyenne.

Les scores obtenus sont très bons, nos modèles semblent performants.

### 3. Modèle ElasticNet

Pourquoi tester un ElasticNet ?

Dans les paragraphes ci-dessus, nous avons vu que la régression Ridge utilise la pénalité L2 et la régression Lasso utilise la pénalité L1.

Le modèle ElasticNet offre pour sa part une combinaison linéaire des pénalités L1 et L2.

Les avantages de cette combinaison sont : la conservation de la capacité de sélection de variables de Lasso avec exclusion des variables non pertinentes, le partage des poids entre variables corrélées et pas de sélection arbitraire

Nous effectuons en premier lieu une visualisation de la distribution des variables à l'aide d'un sns.pairplot(). Le résultat nous montre une distribution bimodale pour la Tmoy et aucun schéma de distribution particulier pour les autres features.

Nous procédons alors à la normalisation du dataframe à l'aide d'un scaler MinMax sur l'ensemble du dataframe.

```
# Normalisation avec le scaler MinMax
# pas de loi normale sur les features en dehors d'une distribution bimodale sur Tmoy
scaler = preprocessing.MinMaxScaler()

data_scaled = pd.DataFrame(scaler.fit_transform(data), index=data.index, columns=data.columns)
```

Nous procédons ensuite à la sélection des features et de la target ‘Consommation (MW)’.

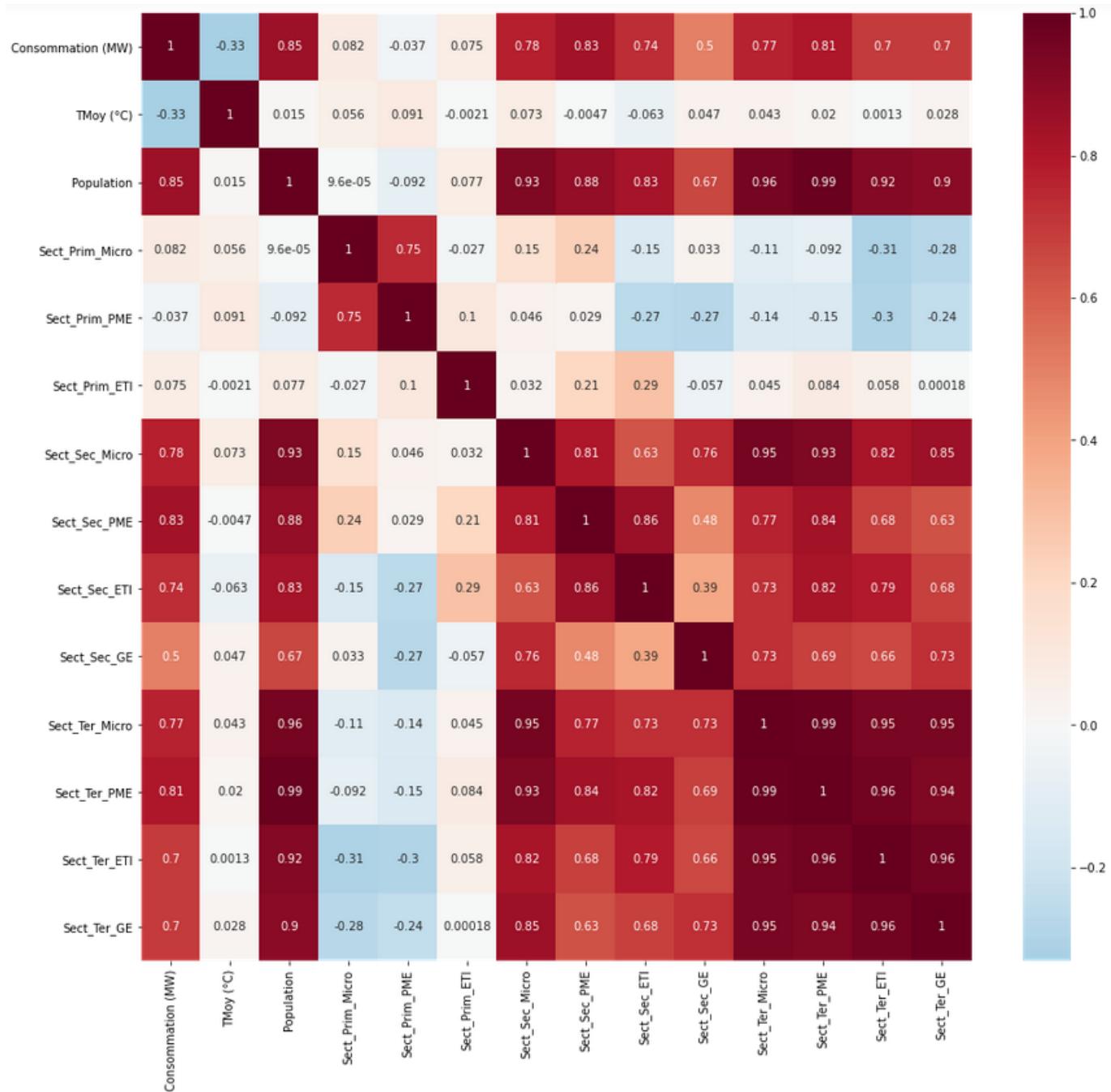
```
# Selection des features et de la target
features = data_scaled.drop(['Consommation (MW)'], axis=1)
target = data_scaled['Consommation (MW)']
```

Puis à la séparation du jeu de données en 2 échantillons train et test tout en gardant la notion de temporalité à l'aide de l'argument Shuffle=False.

```
# Séparation des échantillon train et test avec option Shuffle=False
X_train, X_test, y_train, y_test = train_test_split(features, target_scaled,
                                                    test_size=0.2, random_state=44, shuffle=False)
```

Dans un premier temps, nous allons réaliser une analyse préliminaire par heatmap afin d'observer les corrélations entre chaque colonne du jeu de données.

```
# Analyse préliminaire par heatmap pour observer les correlations entre chaque colonne de data
plt.figure(figsize=(16,15))
sns.heatmap(data.corr(), annot=True, cmap='RdBu_r', center=0)
plt.show()
```



Il semble y avoir des corrélations entre la Consommation et la population, la température et les données des entreprises.

Nous passons alors à l'étape création du modèle en utilisant la validation croisée et entraînement.

```
# Création du modèle ElasticNet
model = ElasticNetCV(cv=10, l1_ratio=(0.1, 0.25, 0.5, 0.7, 0.75, 0.8, 0.85, 0.9, 0.99),
                      alphas=(0.001, 0.01, 0.02, 0.025, 0.05, 0.1, 0.25, 0.5, 0.8, 1.0),
                      max_iter=1000000000)

# Entrainement du modèle
model.fit(X_train, y_train)
```

Nous pouvons alors afficher l'intercept et les coefficients de régression pour observer les variables explicatives ayant un impact sur la variable cible.

```
# Affichage de l'intercept et des coeff estimées pour chaque variable

coeffs = list(model.coef_)
coeffs.insert(0, model.intercept_)
feats = list(features.columns)
feats.insert(0, 'intercept')

pd.DataFrame({'valeur estimée': coeffs}, index=feats)
```

valeur estimée	
intercept	0.272522
TMoy (°C)	-0.332831
Population	0.525907
Sect_Prim_Micro	-0.076689
Sect_Prim_PME	0.016418
Sect_Prim_ETI	-0.031440
Sect_Sec_Micro	0.110990
Sect_Sec_PME	0.184293
Sect_Sec_ETI	-0.000000
Sect_Sec_GE	-0.028805
Sect_Ter_Micro	-0.000000
Sect_Ter_PME	0.000000
Sect_Ter_ETI	-0.298532
Sect_Ter_GE	0.057827

On remarque que la population, la Température moyenne et le nombre d'ETI du secteur tertiaire ont une influence importante sur le modèle.

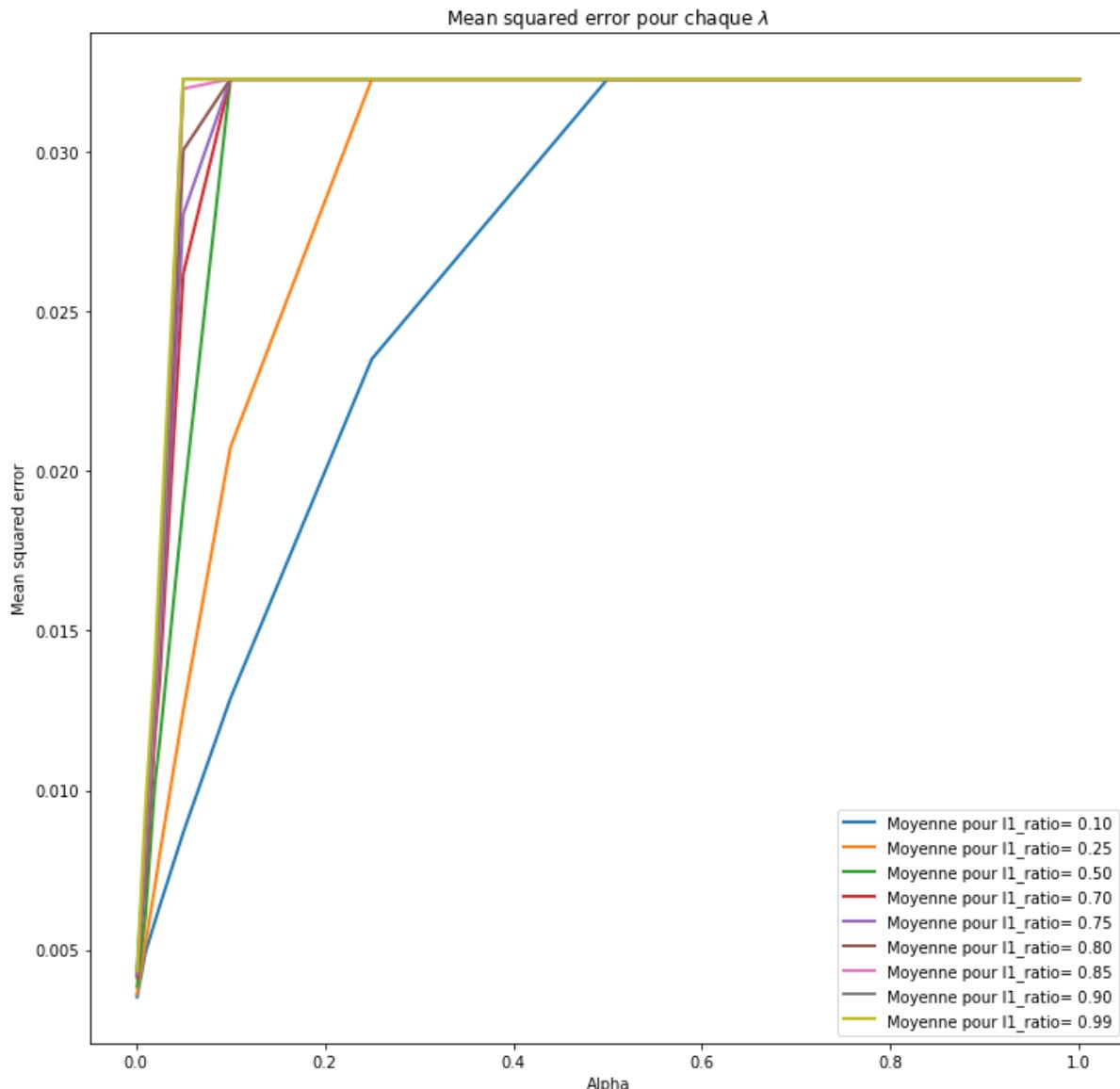
D'autre part, on remarque également l'influence dans une moindre mesure des micro-entreprises du secteur primaire, les micro-entreprises du secteur secondaire ainsi que les grandes entreprises du secteur tertiaire.

Nous allons donc à présent afficher la MSE en fonction des alphas pour chaque L1\_ratio

```
# Affichage de la moyenne des erreurs MSE en fonction des alphas pour chaque l1_ratio
alphas = model.alphas_
plt.figure(figsize=(12,12))

for i in range(model.mse_path_.shape[0]):
    plt.plot(alphas, model.mse_path_[i,:,:].mean(axis=1),
              label='Moyenne pour l1_ratio= %.2f' %model.l1_ratio[i], linewidth=2)

plt.xlabel('Alpha')
plt.ylabel('Mean squared error')
plt.title('Mean squared error pour chaque $\lambda$')
plt.legend()
plt.show()
```



La MSE augmente rapidement quelque soit le L1 tout en plafonnant à une valeur très faible.

Affichons l'alpha retenu par le modèle, ainsi que les métriques MSE et le score pour les 2 échantillons.

```
] print('alpha sélectionné par le modèle :', model.alpha_)

alpha sélectionné par le modèle : 0.001

[i]: # MSE (Mean Squared Error) pour Train et test
print('MSE train :', mean_squared_error(y_train, pred_train))
print('MSE test :', mean_squared_error(y_test, pred_test))

MSE train : 0.0030919514617136893
MSE test : 0.0030690568974159403

[i]: # score R2 (coefficient de détermination) pour les 2 échantillons
print('R2 score train :', model.score(X_train, y_train))
print('R2 score test :', model.score(X_test, y_test))

R2 score train : 0.9019682570064667
R2 score test : 0.877495745045526
```

Nous obtenons une MSE de 0.003 et un R<sup>2</sup> score de 0.901 sur l'échantillon train, et une MSE de 0.003 et un R<sup>2</sup> score de 0.877 sur l'échantillon de test.

Les résultats sont plutôt bons, le modèle semble performant.

Nous allons donc compléter les métriques avec l'erreur de pourcentage absolu moyen (MAPE).

```
] # MAPE (Mean Absolute Percentage Error) pour les 2 échantillons
print('MAPE train :', mean_absolute_percentage_error(y_train, pred_train))
print('MAPE test :', mean_absolute_percentage_error(y_test, pred_test))

MAPE train : 18447978725.81969
MAPE test : 0.29526505260550207
```

Nous obtenons une MAPE extrêmement grande sur l'échantillon train et un MAPE de 0.29% sur l'échantillon test. Le résultat sur l'échantillon de test est très bon.

Cependant la valeur de la MAPE sur l'échantillon train est anormalement grande.

Après investigations sur la méthode de calcul de la MAPE et contrôle des valeurs de l'échantillon train, il apparaît que y\_train inclus une valeur à 0 ce qui impose au calcul de la MAPE l'utilisation d'un Epsilon au dénominateur qui du coup va biaiser le résultat de la MAPE pour le y\_train. Concernant le y\_test, la vérification montre que l'échantillon est exempt de 0.

```
] y_train[y_train['Consommation (MW)'] == 0]
# raison du MAPE train très haut
```

Consommation (MW)	
2016-06-26 - Centre-Val de Loire	0.0

```
] y_test[y_test['Consommation (MW)'] == 0]
```

Consommation (MW)	
-------------------	--

Comme pour les modèles Ridge et Lasso, nous allons afficher les valeurs observées et les valeurs prédictives sur l'échantillon de test. Pour cela, nous devons tout d'abord de-scaler les valeurs de pred\_test et y\_test qui ont été converties en dataframes.

```
)]: # création des variables min et max
min = scaler.data_min_[0]
max = scaler.data_max_[0]
print('min :', min)
print('max :', max)

min : 1597.0
max : 15338.0

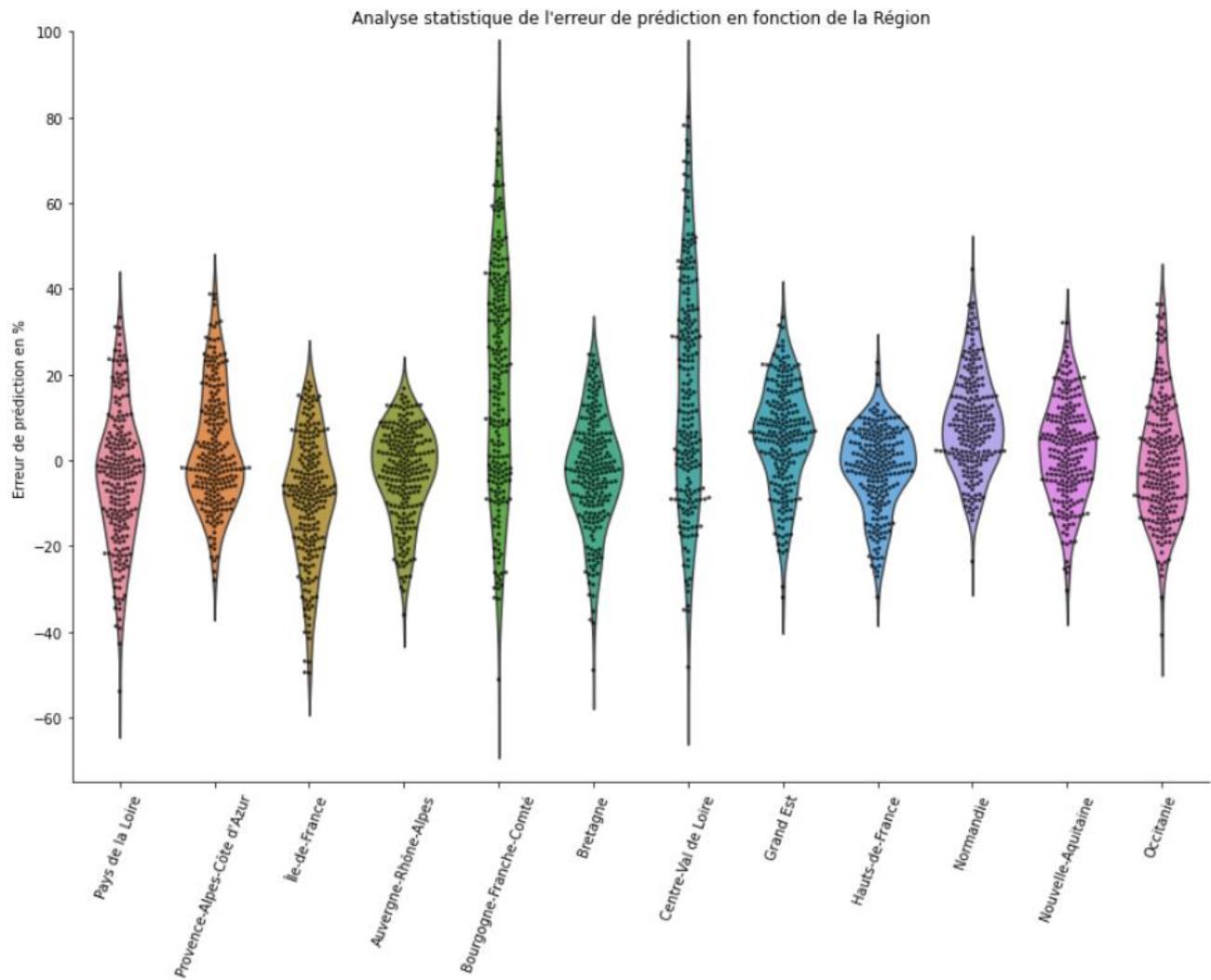
): # Affichage des consommations observées et des consommations prédictives par le modèle Elastic Net.

pd.DataFrame({'Consommations_observées_(MW)': (y_test['Consommation (MW)']*(max-min))+min,
               'Consommations_predites_(MW)' : (pred_test['Consommation (MW)']*(max-min))+min},
               index=y_test.index).head(50)
```

	Consommations_observées_(MW)	Consommations_predites_(MW)
2018-05-26 - Pays de la Loire	2864.0	3021.913812
2018-05-26 - Provence-Alpes-Côte d'Azur	4477.0	4260.155424
2018-05-26 - Île-de-France	7306.0	8891.363771
2018-05-27 - Auvergne-Rhône-Alpes	6349.0	7699.869620
2018-05-27 - Bourgogne-Franche-Comté	2027.0	1707.477080
2018-05-27 - Bretagne	2240.0	2656.671360
2018-05-27 - Centre-Val de Loire	1723.0	1583.897992
2018-05-27 - Grand Est	4481.0	4287.135874
2018-05-27 - Hauts-de-France	4976.0	4641.899148
2018-05-27 - Normandie	2742.0	2511.789572
2018-05-27 - Nouvelle-Aquitaine	3982.0	5214.392946
2018-05-27 - Occitanie	3775.0	4509.161773
2018-05-27 - Pays de la Loire	2793.0	3255.891117

Certaines valeurs semblent présenter des écarts de prédiction importants. Afin de pouvoir analyser ces écarts, nous allons calculer l'erreur de prédiction en pourcentage dans la variable pct\_error puis nous afficherons l'analyse statistique par région afin de visualiser des différences possibles entre régions.

	Région	Date	Consommations_observées_(MW)	Consommations_predites_(MW)	pct_error
0	Pays de la Loire	2018-05-26	2864.0	3021.913812	5.513750
1	Provence-Alpes-Côte d'Azur	2018-05-26	4477.0	4260.155424	-4.843524
2	Île-de-France	2018-05-26	7306.0	8891.363771	21.699477
3	Auvergne-Rhône-Alpes	2018-05-27	6349.0	7699.869620	21.276888
4	Bourgogne-Franche-Comté	2018-05-27	2027.0	1707.477080	-15.763341



#### Observations :

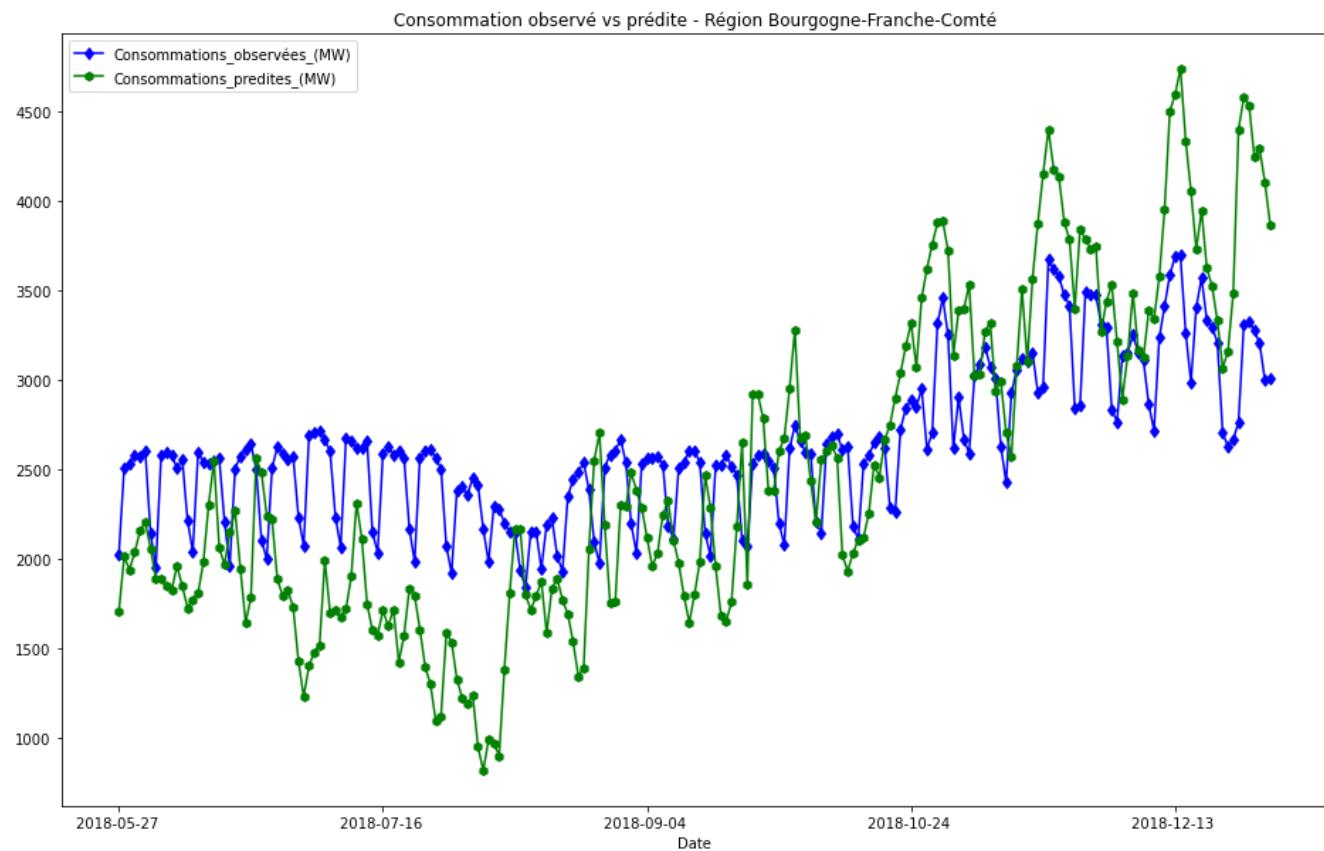
Cette analyse statistique laisse apparaître plusieurs profils de régions :

- Des régions avec une distribution « compacte » telles que l'Auvergne-Rhône-Alpes ou les Hauts-de-France.
- Des régions avec une distribution « étendue » comme la Bourgogne-Franche-Comté ou le Centre-Val de Loire.
- Des régions avec une distribution « intermédiaire » comme la Normandie ou l'Île-de-France.

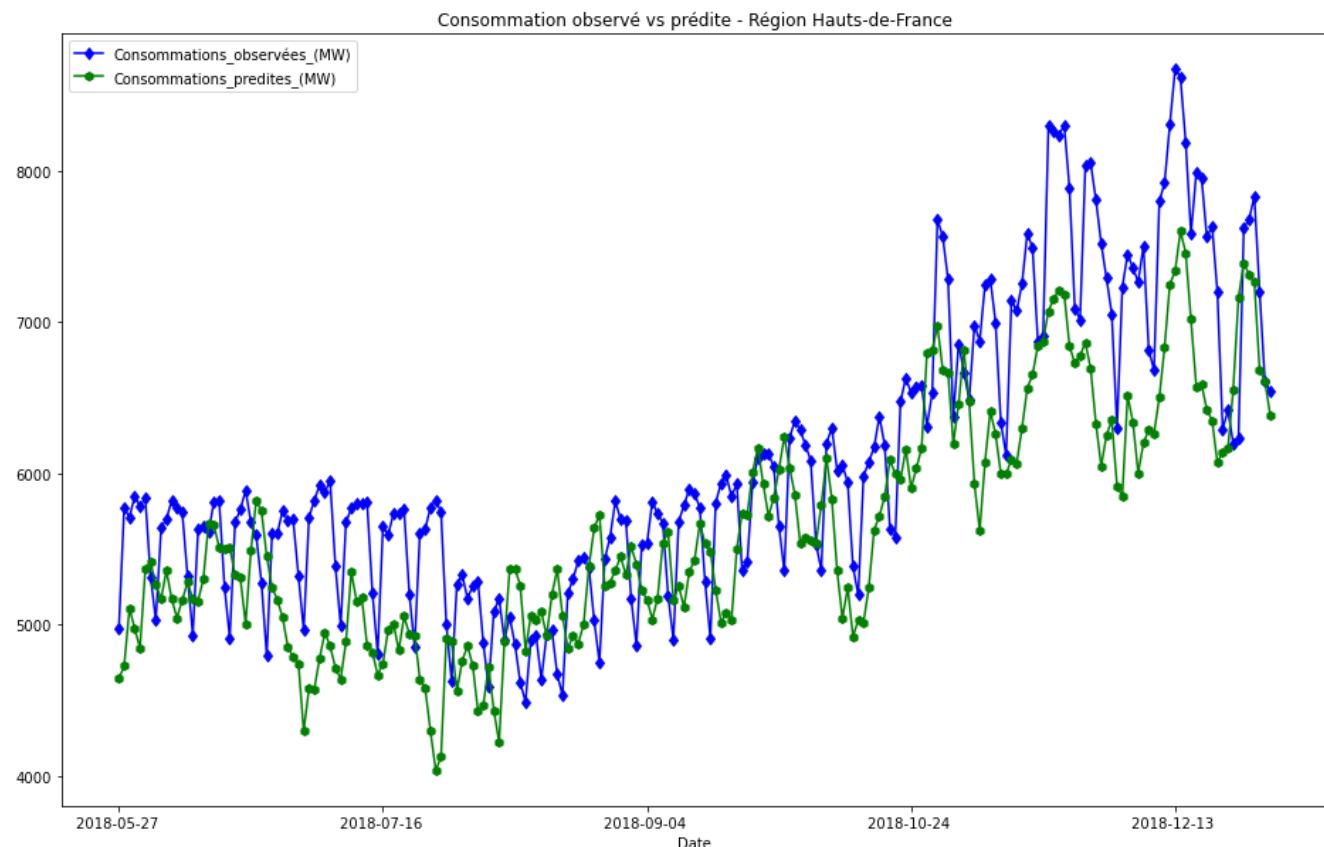
De plus, on observe que quelque soit le profil, les erreurs de prédiction sont nombreuses et loin d'être faibles (supérieur à 5%).

Afin de finaliser l'analyse de la qualité des prédictions, nous allons sélectionner 2 régions avec des profils différents pour visualiser les valeurs prédites et les valeurs observées sur tout l'échantillon de test.

## Région Bourgogne-Franche-Comté :



## Région Hauts-de-France :



On note que quelque soit le profil, les prédictions ne sont pas bonnes avec des écarts importants entre les prédictions et le réel.

## Conclusion sur les modèles de Régression régularisée :

Malgré des métriques exceptionnellement bonnes et qui laissent penser que les modèles sont performants, il apparaît que les prédictions sont totalement en écart avec les valeurs observées.

Cette divergence entre métriques et performance pose une problématique importante sur laquelle nous ne pouvons émettre que des hypothèses.

Hypothèse 1 : le type de modèle serait inadapté à la relation entre la variable cible et les variables explicatives (cf pairplot dans notebook). Il nous faudrait donc choisir d'autres modèles à tester (ex : Random Forest).

Hypothèse 2 : les variables explicatives utilisées pour les modèles sont non suffisamment représentatives ou portent des biais.

Par exemple, la Température moyenne pour une région est une valeur portant une information de faible qualité pour notre modèle (une région est beaucoup trop étendue pour que la température soit représentative, il faudrait peut-être un gradient de température sur la région en fonction de la densité de population).

D'autre part, les data de population ou d'entreprises ne sont données que pour chaque année et non quotidiennement, cela pose donc un problème de fréquence et donc une distribution particulière de la variable.

De façon plus générale, d'autres features moins « évidentes » auraient peut-être été plus pertinentes pour la prédiction.

Par ailleurs, nous avons vérifié (voir les notebooks jupyter) s'il n'y avait pas un biais lié au scaler entre le Standard scaler et le MinMax scaler sur 2 modèles Elastic Net. Le résultat est que le MinMax minimise légèrement les erreurs avec des outliers max à 60% au lieu de 80%. Mais le problème reste entier dans la divergence entre les métriques et les résultats de prédiction.

Cela démontre qu'il ne faut jamais partir du principe que les métriques sont suffisantes à qualifier un modèle mais qu'une vérification par comparaison prédiction vs observé est absolument nécessaire afin de valider le modèle.

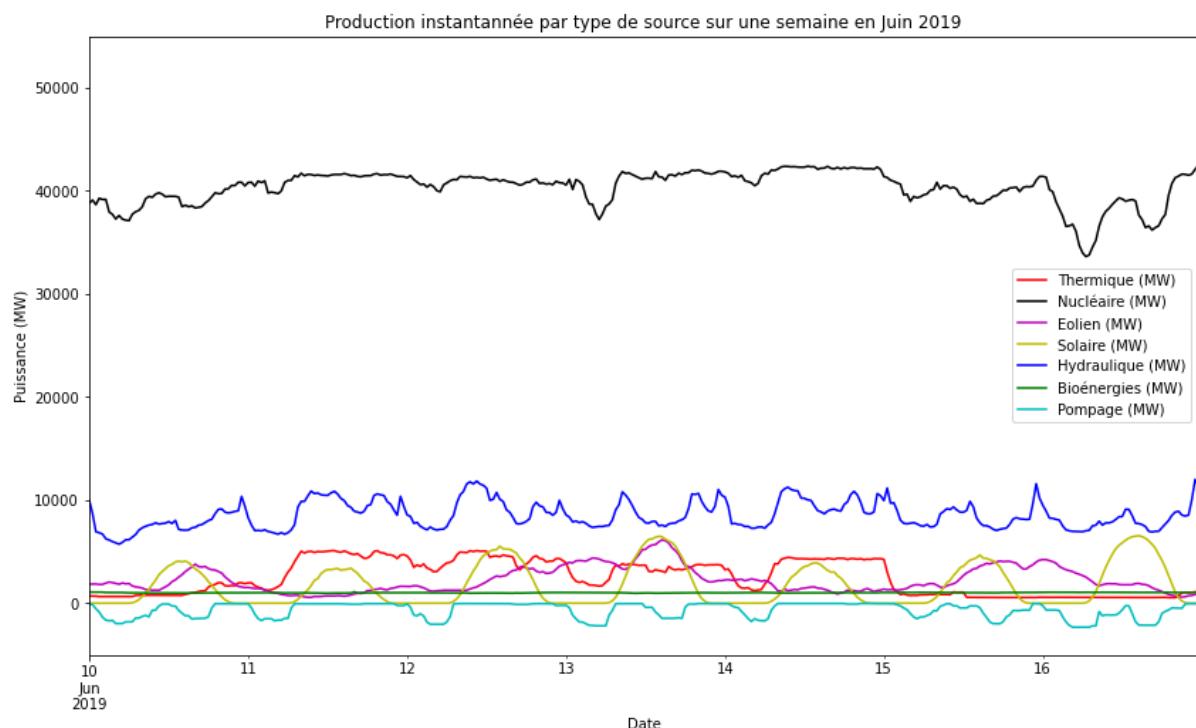
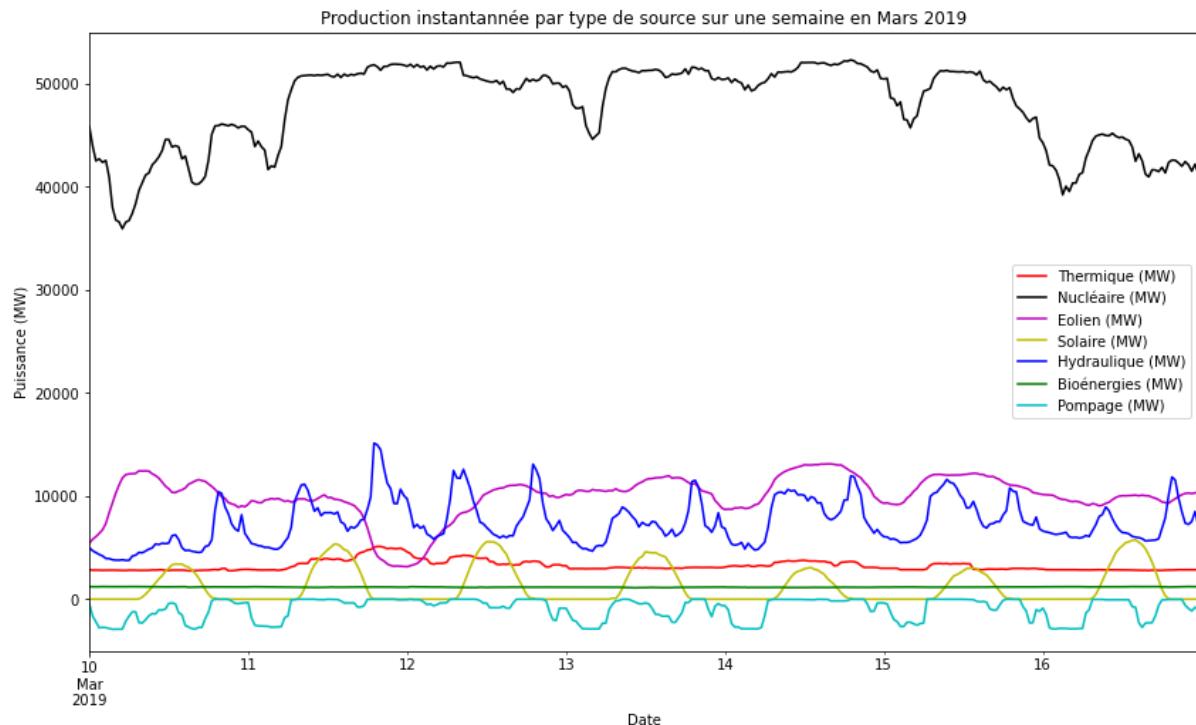
## VI. Dataviz Sources de Production

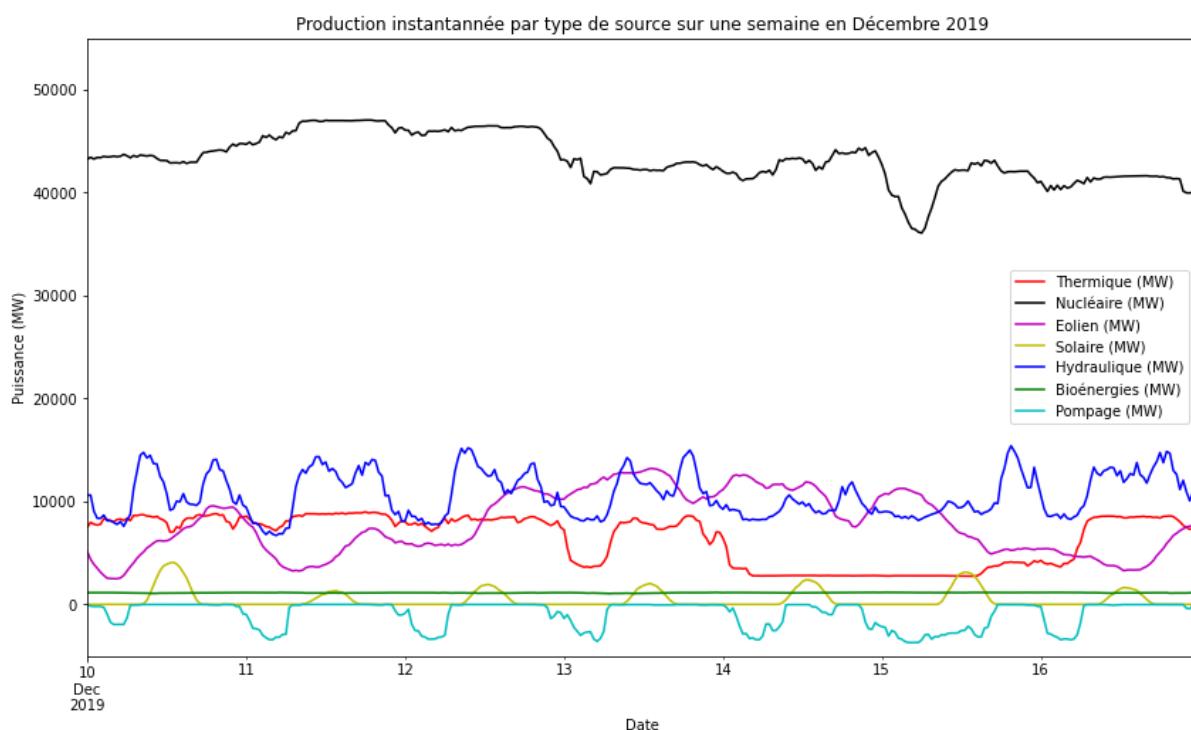
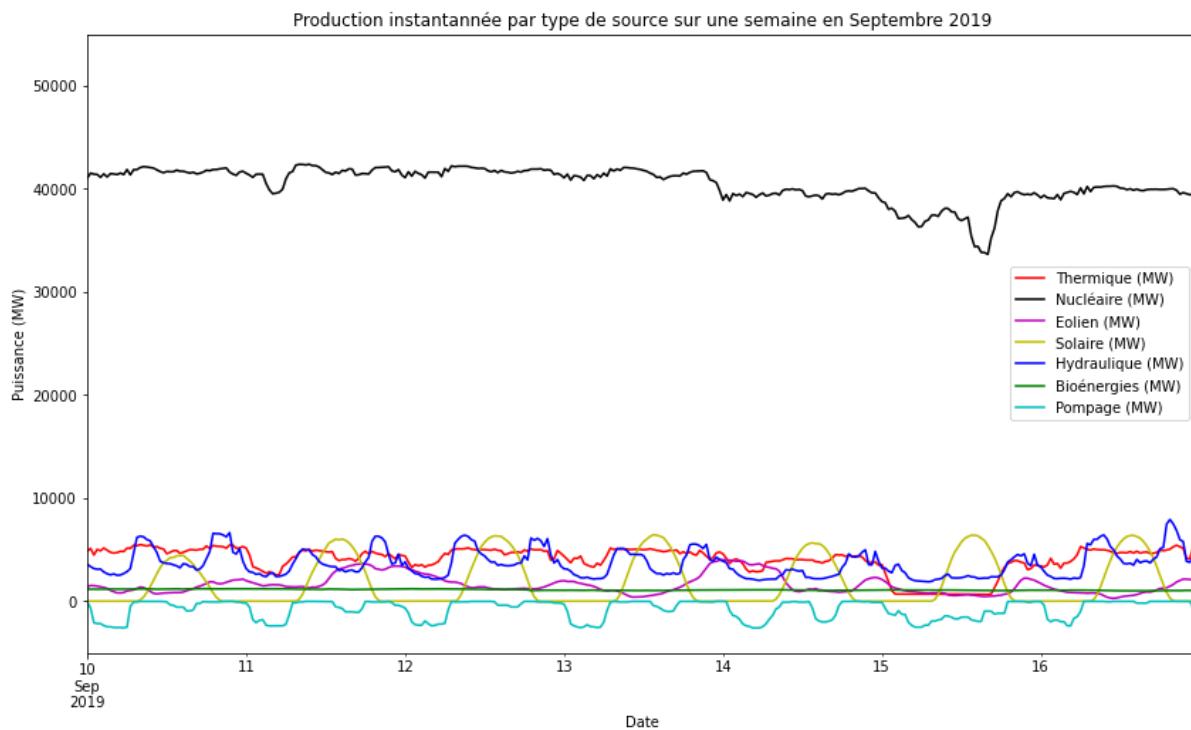
L'objectif de ce chapitre est enfin de répondre à notre troisième problématique :

- Quelles sont les sources d'énergies au niveau national et régional qui contribuent à satisfaire les besoins d'électricité et dans quelles proportions ?

Dans un premier temps, nous allons observer la Production Nationale instantanée par type de source sur une semaine en Mars Juin, Septembre et Décembre 2019.

Note : l'échelle de Puissance est conservé entre les graphiques pour comparaison.





#### Observations :

On note un assemblage complexe de multiples productions avec des comportements et des utilités distinctes selon le type de source :

- Les sources non contrôlables telles que les productions Eolienne et Solaire qui sont qualifiées d'Énergies Renouvelables Intermittentes pas RTE (EnRi).
- Les sources contrôlables et peu flexibles telles que les productions Nucléaire, Thermique et Bioénergies
- Les sources contrôlables et capables de flexibilité extrême comme l'hydraulique en production ou son pendant le pompage STEP permettant de stocker l'énergie.

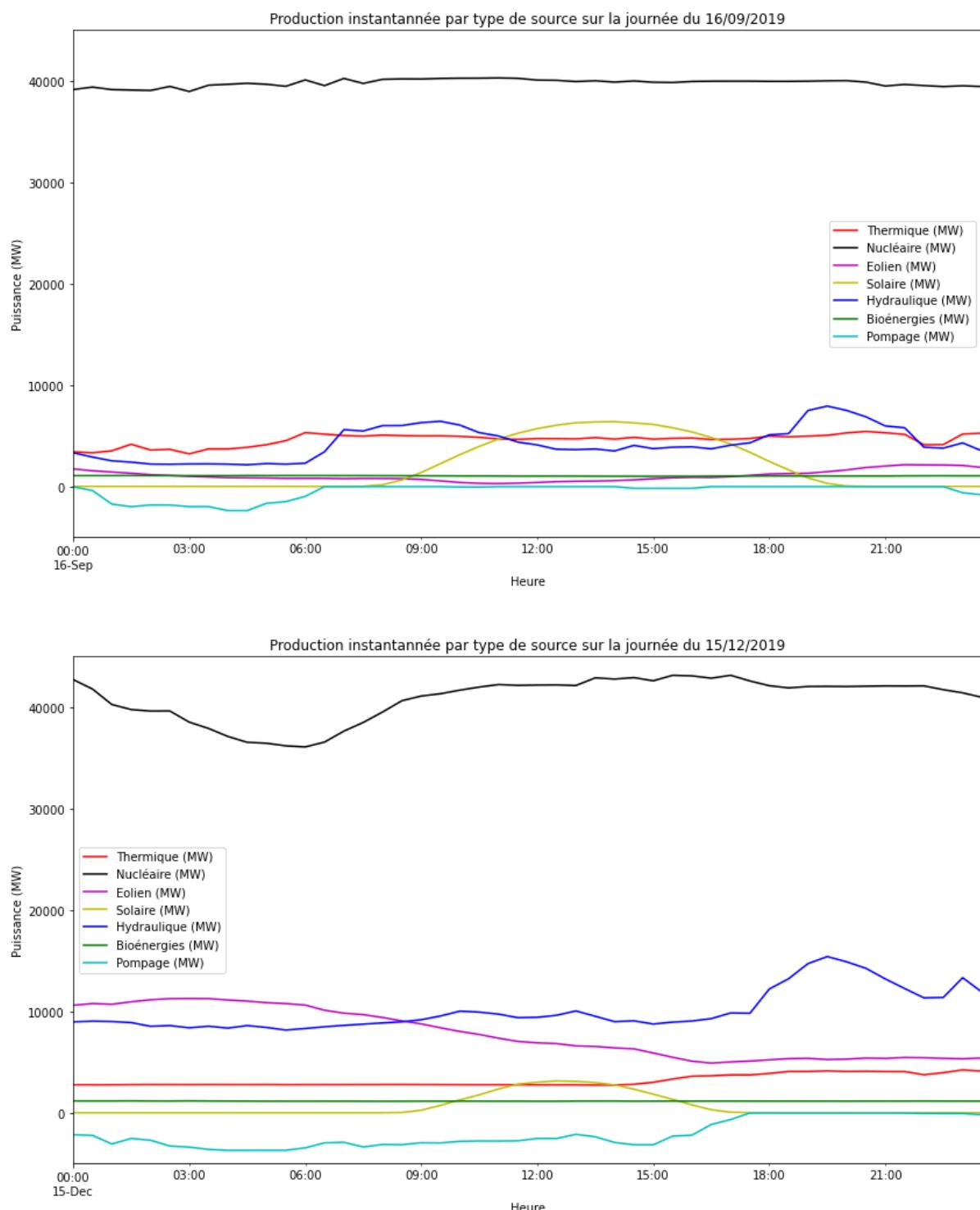
La production nucléaire avec environ 40000 à 50000 MW sert de socle à la production nationale et est relativement flexible.

Ensuite on peut observer la production Solaire en forme de cloche pendant la journée (en fonction du rayonnement solaire).

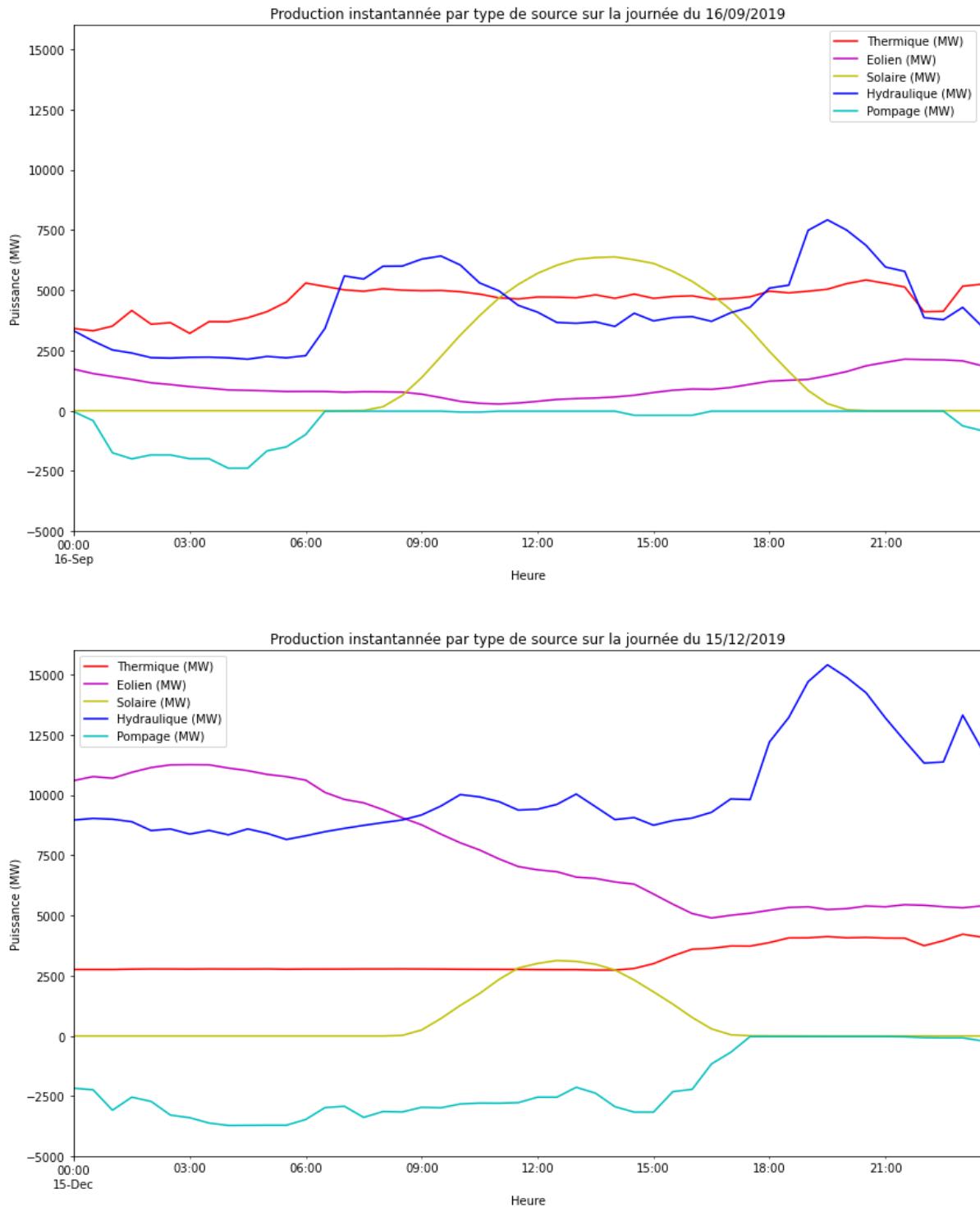
La production Eolienne est elle très volatile en fonction de la vitesse des vents.

Enfin la grande variabilité nécessaire à la production pour répondre aux besoins de consommation est fournie par l'Hydraulique et le pompage STEP.

En zoomant sur 2 journées des semaines précédemment affichées (le 16/09/2019 et le 15/12/2019), on peut confirmer les observations exprimées ci-dessus.



Enfin nous proposons de faire un focus sur les 2 mêmes journées en enlevant le Nucléaire pour éviter un « écrasement » d'échelle de Puissance.



Note :

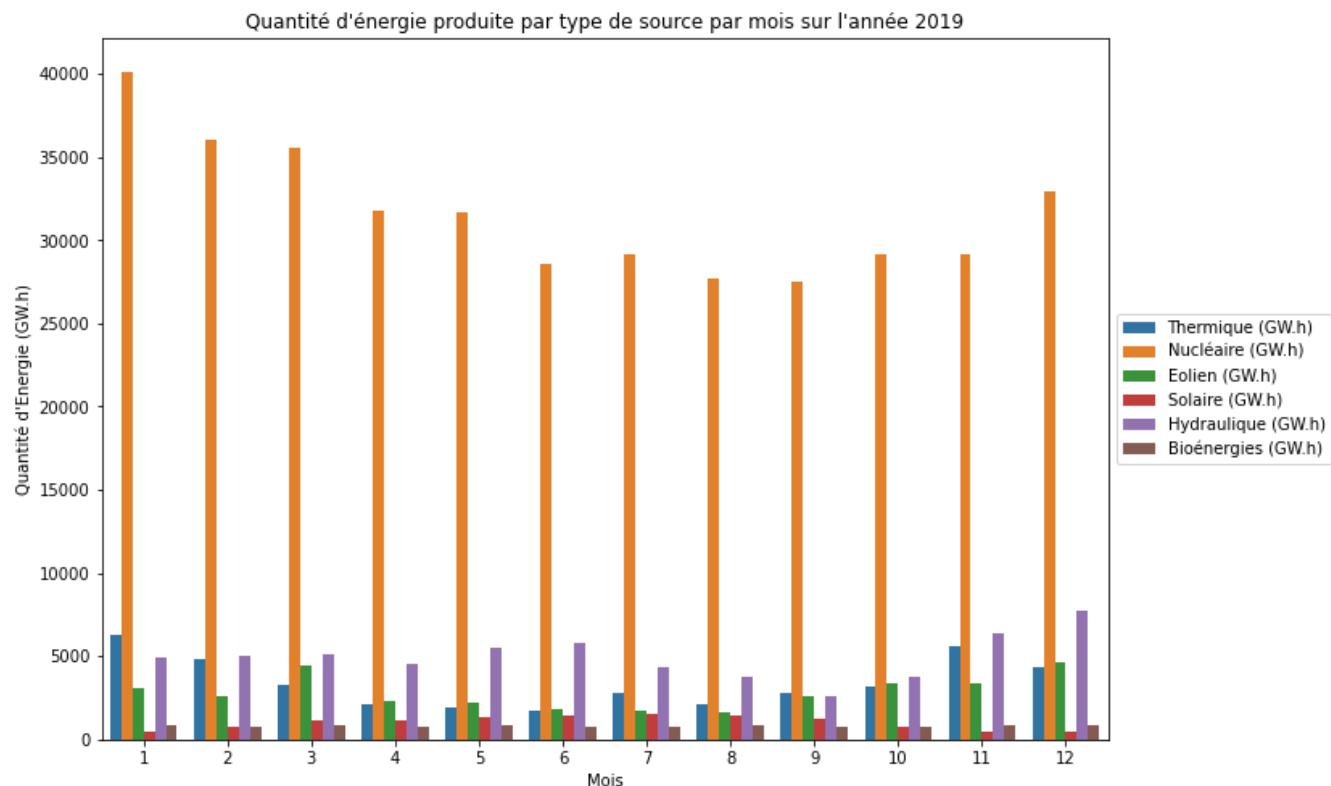
On visualise mieux la variabilité des productions.

On peut également noter la forte différence de production pour l'Eolien et le Solaire en fonction de la journée et de la saison.

Production Eolienne : Septembre de 300 à 2000 MW / Décembre de 5000 à 11500 MW

Production Solaire : Septembre pic à plus de 6000 MW / Décembre pic à 3000 MW

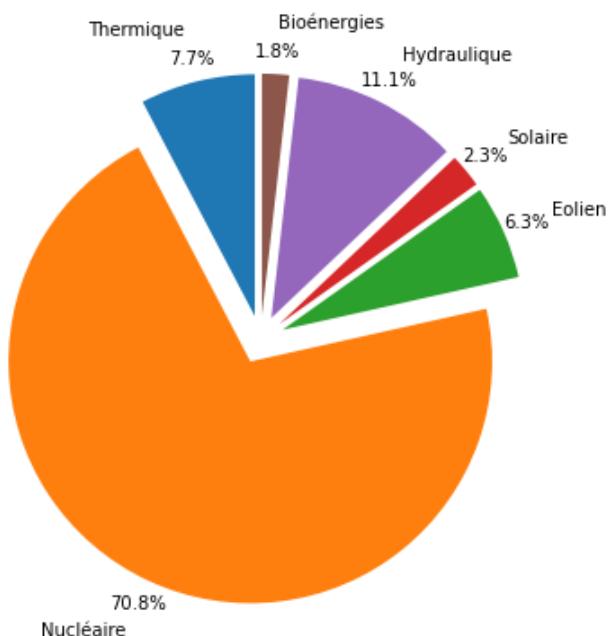
Attachons-nous maintenant à analyser la Quantité d'Energie produite par mois sur l'année 2019.



#### Observations :

On retrouve la prépondérance du Nucléaire sur toutes les autres énergies avec une production variant de 27000 à 40000 GW.h selon les besoins. Au global, on retrouve la saisonnalité pour répondre à la demande de consommation déjà observée précédemment.

#### Répartition de la production sur l'année 2019 en fonction du type d'énergie :



#### Observations :

- 70.8% provient des centrales nucléaires,
- 7.7% provient de centrales Thermiques,
- 21.5% provient d'énergies renouvelables (Hydraulique, Eolien, Solaire, Bioénergies).

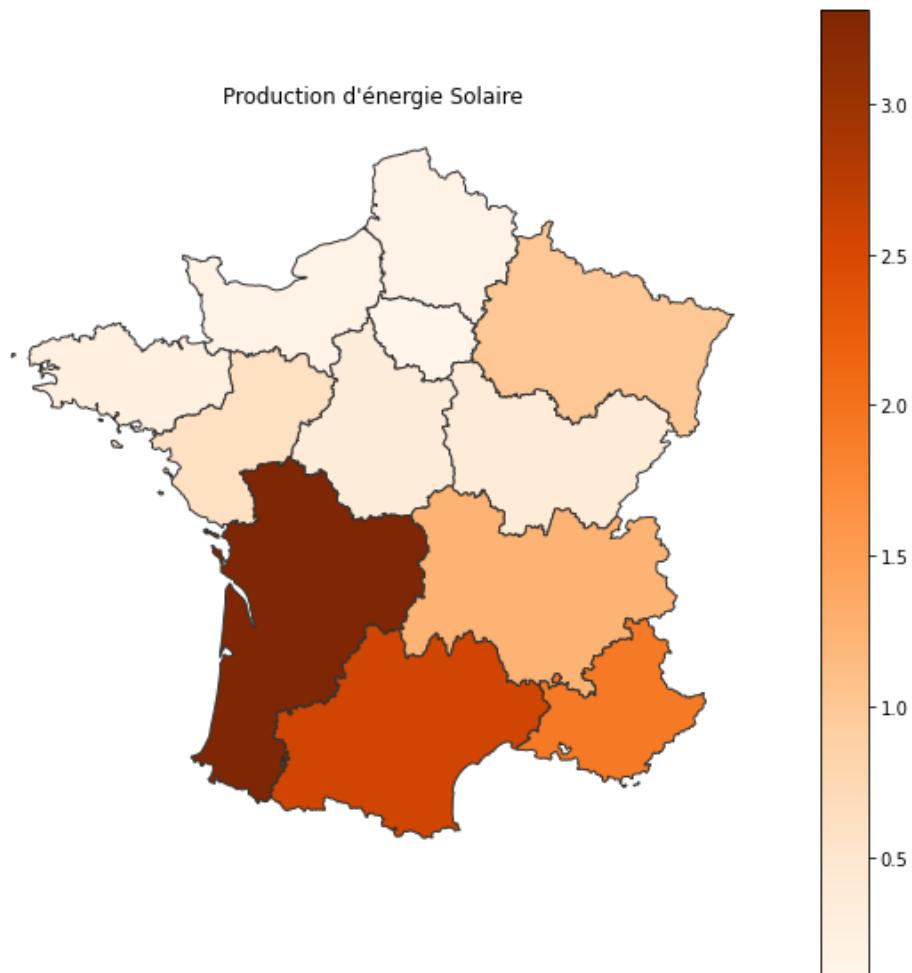
Nous proposons maintenant d'observer les Régions contributrices en fonction des types d'énergie.

Pour répondre à cette question, nous avons choisi d'illustrer notre propos à l'aide de cartes choroplèthes permettant de représenter des données quantitatives, ici les différentes productions par type de source d'énergie par région.

Pour cela nous avons utilisé la librairie « GeoPandas » qui permet de travailler avec des données géospatiales (ici des données de géolocalisation des régions).

Nous représentons ci-dessous la production par région (en TWh) des différentes énergies pour l'année 2019.

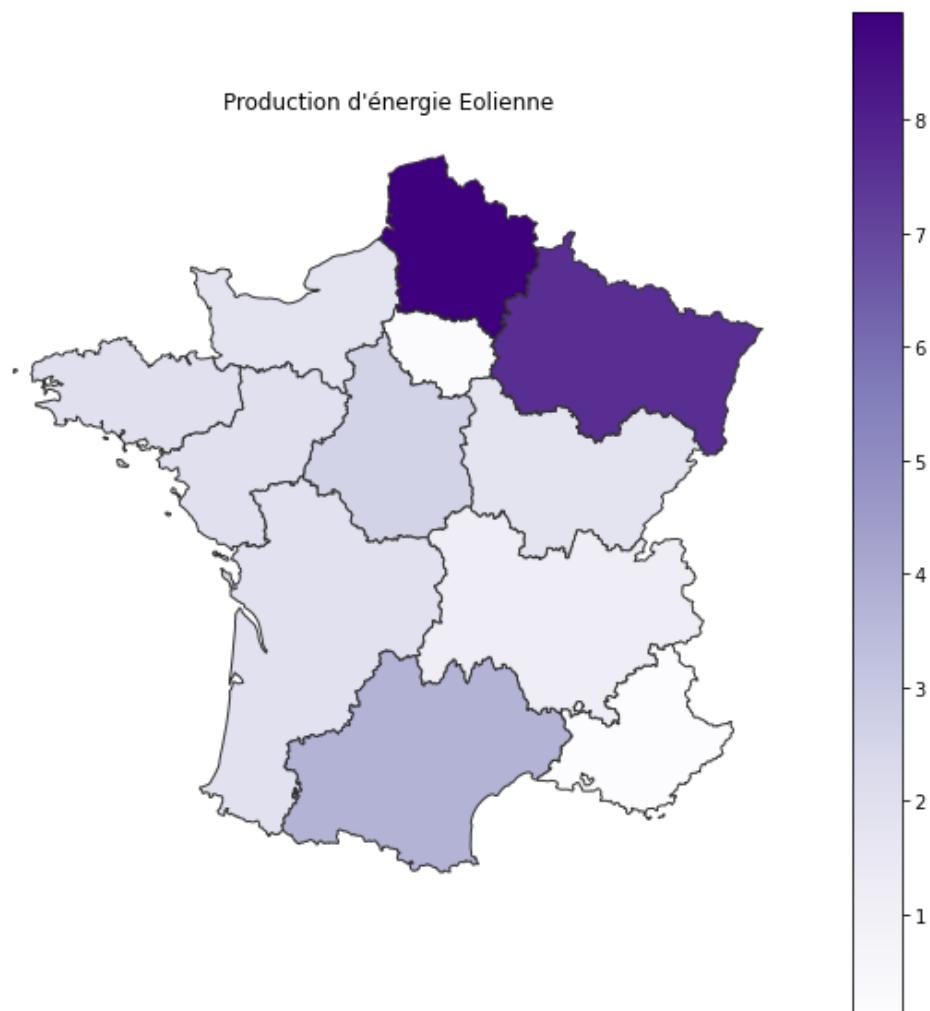
### ***Production d'électricité en TWh à partir de l'Energie Solaire***



#### Observations :

- La Nouvelle-Aquitaine, l'Occitanie et la région Provence-Alpes-Côte d'Azur sont les régions produisant le plus d'énergie solaire, ce qui paraît logique du point de vue géographique (rayonnement solaire) et en rapport avec le fait que ces 3 régions totalisent plus de 60% de la puissance installée en métropole avec 5931 MW sur les 9283 MW en 2019.

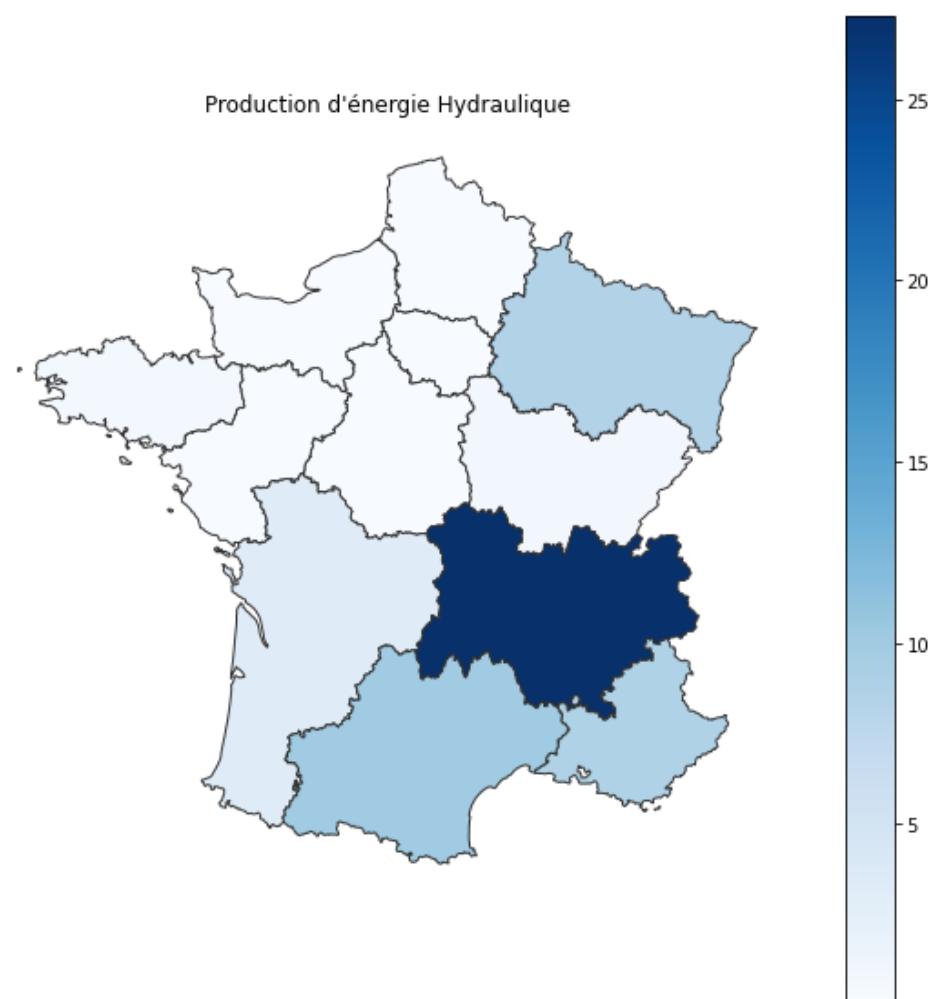
### Production d'électricité en TWh à partir de l'Energie Eolienne



#### Observations :

- La majorité de l'électricité produite par le parc éolien provient des régions Grand Est et Hauts-de-France.  
Au-delà des aspects climatiques (gisement de vents), le développement de parcs éoliens est lié aux tarifs d'achat de l'électricité produite et aux démarches plus ou moins lourdes du processus d'implantation. C'est ainsi que les Hauts-de-France et le Grand-Est concentrent la moitié du Parc Eolien avec en 2019, 8166 MW installée sur 16476 MW en métropole.

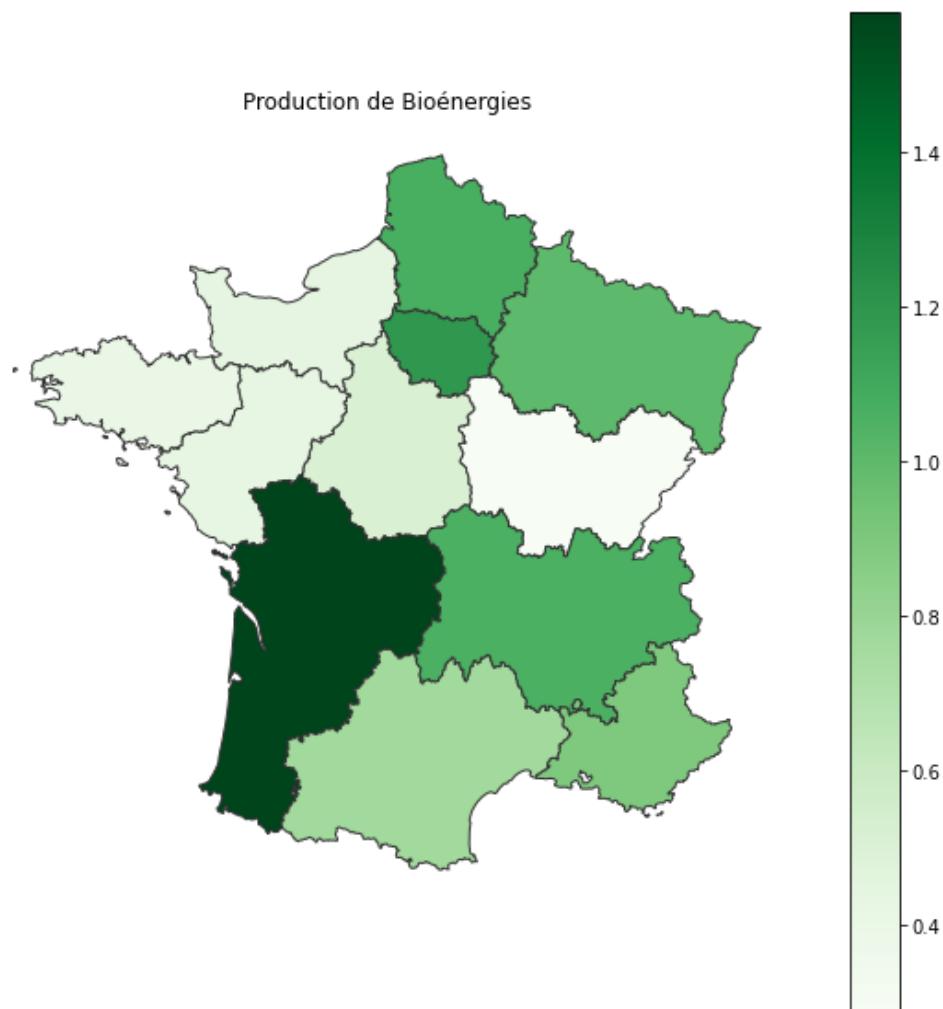
### Production d'électricité en TWh à partir de l'Energie Hydraulique



Observations :

- L'hydroélectricité, première source d'électricité à partir d'énergie renouvelable en France est produite par 3 régions : l'Auvergne-Rhône-Alpes, l'Occitanie et la région Provence-Alpes Côte d'Azur. La région Auvergne-Rhône-Alpes constitue à elle seule 45% de la puissance installée en 2019 et si l'on prend les 3 régions les plus productrices, on monte alors à 20302 MW installée sur 25553 MW en métropole, soit près de 80% !

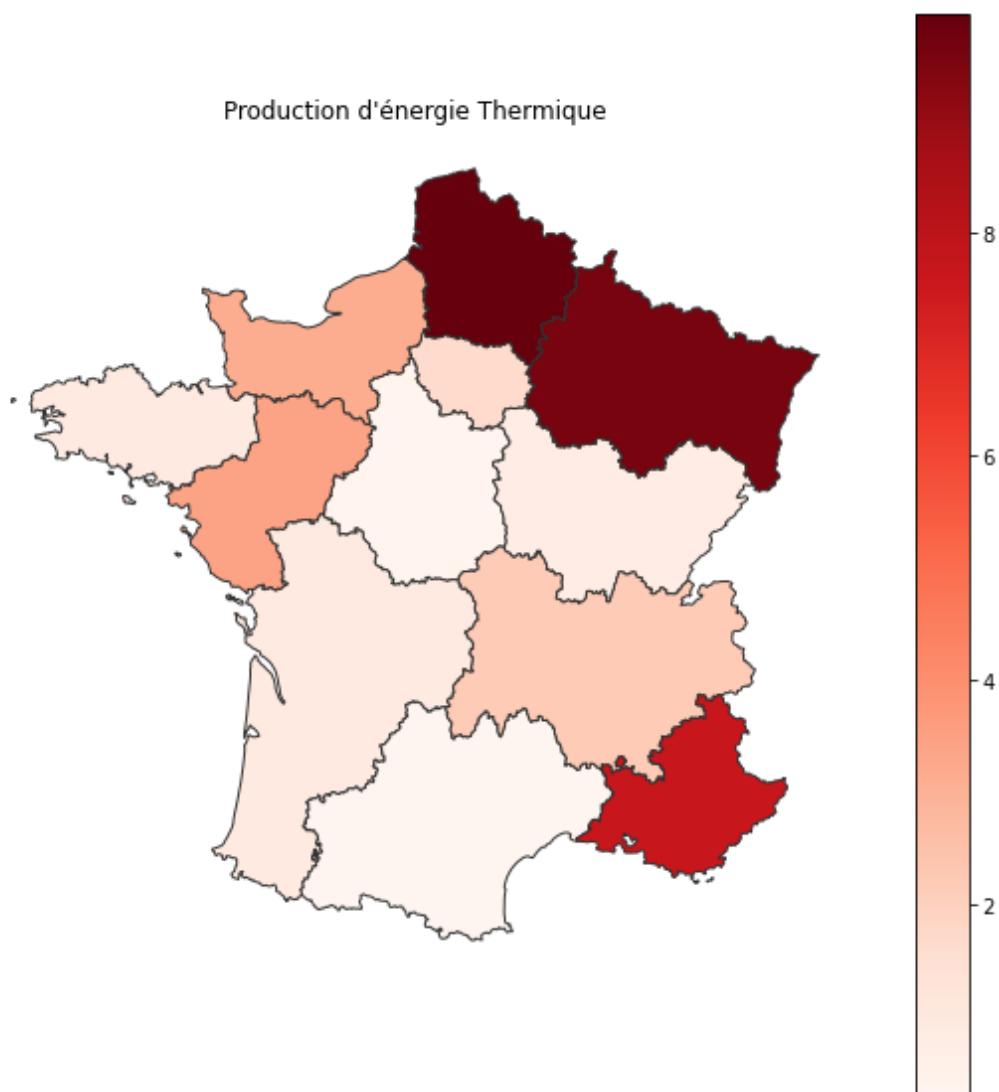
### **Production d'électricité en TWh à partir des Bioénergies**



#### Observations :

- L'électricité issue des bioénergies est produite principalement par la Nouvelle-Aquitaine, l'Auvergne Rhône-Alpes et l'Ile-de-France. On notera la 3<sup>ème</sup> place de l'Ile de France qui s'explique par la nature même de cette énergie : dit dans le jargon technique « issue de la biomasse ». Dans les faits, en France, 10% de la production d'électricité issue de la biomasse provient du biogaz (méthanisation) et les 90% restants sont issues de la combustion de matières de types bois, végétaux, déchets agricoles, ordures ménagères organiques.  
Si on y regarde donc de plus près pour l'Ile de France, l'usine d'incinération des déchets urbains Dalkia Wastenergy (filiale de Dalkia, elle-même filiale d'EDF) d'Ivry-sur Seine traite les déchets ménagers de plus de 5 millions d'habitants (soit plus de 690 000 t par an). Cela explique donc la 3<sup>ème</sup> position de la région.

### ***Production d'électricité en TWh à partir de l'Energie Thermique***

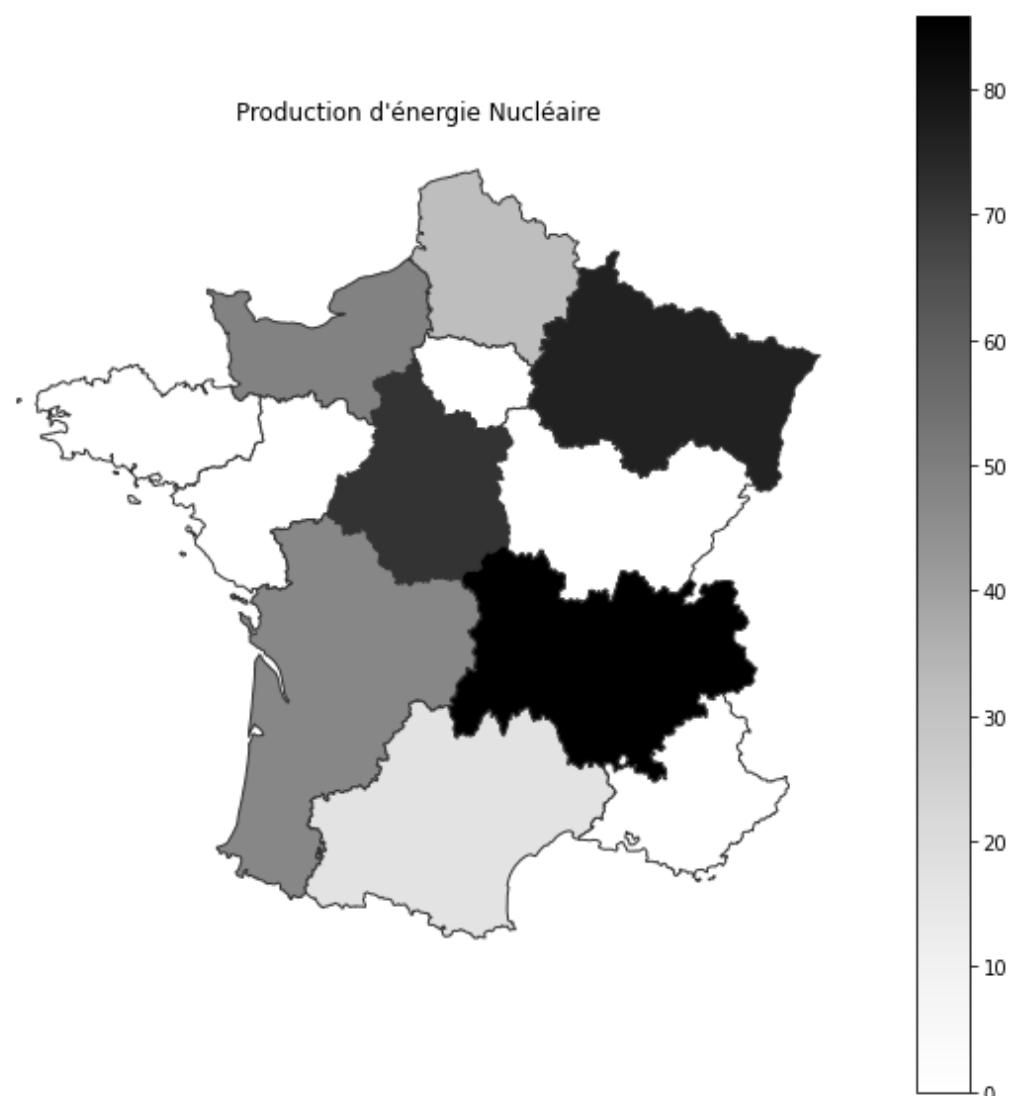


#### Observations :

- Les centrales thermiques produisent le plus dans les régions Hauts-de-France, Grand Est et Provence-Alpes-Côte d'Azur. Cependant, les 13 centrales thermiques (3 à charbon et 10 à cycle combiné de gaz) présentes en métropole sont réparties sur seulement 5 régions pour un total de 8577 MW installées :
  - Provence-Alpes-Côte d'Azur / 2873 MW
  - Grand Est / 2321 MW
  - Pays de la Loire / 1608 MW
  - Hauts-de-France / 1363 MW
  - Auvergne-Rhône-Alpes / 412 MW

Il semble donc que les centrales thermiques soient sollicitées indépendamment de leur puissance mais effectivement comme appoint temporaire comme nous l'avons vu dans les analyses précédentes.

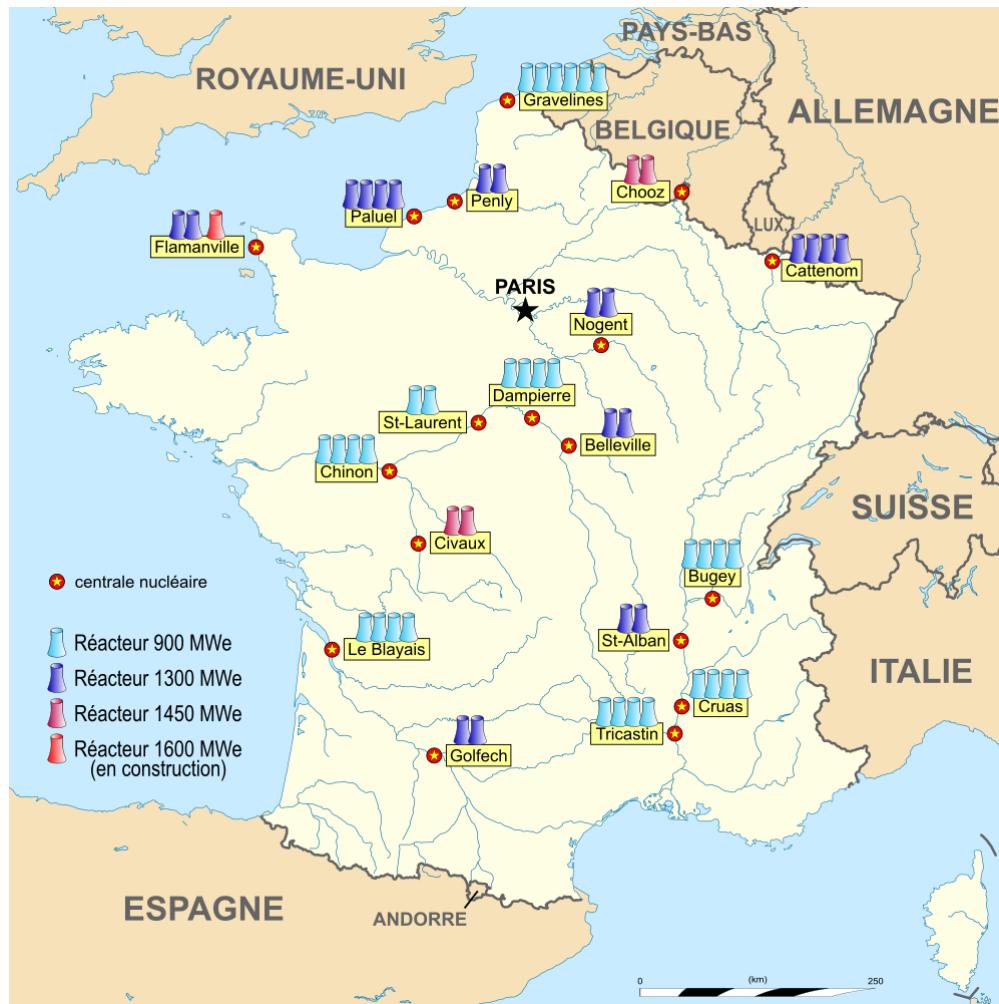
### **Production d'électricité en TWh à partir de l'Energie Nucléaire**



#### Observations :

- Les régions Auvergne Rhône-Alpes, le Centre-Val de Loire et le Grand Est sont les 3 plus grandes régions productrices du parc nucléaire français. Elles produisent à elles seules plus de 60% de l'électricité nucléaire.  
En effet, pour fonctionner les centrales nucléaires ont besoin d'une source d'eau froide et sont donc situées en bord de mer ou près d'un cours d'eau ayant un débit suffisant. Les choix des sites se font aussi en fonction des besoins en électricité : plusieurs réacteurs sont situés dans la Vallée du Rhône à proximité des sites industriels de la région Rhône-Alpes, d'autres en Normandie et en bord de Loire près de la région parisienne.

Voici donc la répartition des Centrales françaises sur le territoire :



Le tableau ci-dessous présente le détail des Puissances nettes installées. On retrouve bien une similitude entre Production et Puissance installée dans les régions productrices.

Région	Centrales	Nombre de réacteurs	Puissance nette installée en GW
Auvergne-Rhône-Alpes	Bugey, Cruas, Saint-Alban, Tricastin	14	13,4
Centre-Val de Loire	Belleville, Chinon, Dampierre, Saint-Laurent	12	11,6
Grand Est	Cattenom, Chooz, Nogent	8	10,7
Normandie	Flamanville, Paluel, Penly	8 (+1)	10,4 (+1,6)
Nouvelle-Aquitaine	Le Blayais, Civaux	6	6,5
Occitanie	Golfech	2	2,6
Hauts-de-France	Gravelines	6	5,4
	<b>TOTAL</b>	<b>56 (+1)</b>	<b>60,6 (+1,6)</b>

# VII. Vie du Projet

## A. Description des travaux

La description des travaux réalisés dans ce projet est décrite dans le tableau ci-dessous. Vous y trouverez les tâches, les livrables associés, l'allocation des ressources et la target date pour chaque tâche.

Le GANTT se trouve en annexe.

n°	Tâche	Livrable	Qui	Target
1	Réaliser la Data exploration	Notebook d'explo et invest techniques	Samira/Alex/Elianne	01/07/2021
2	Etablir les objectifs du projet et structurer le projet	problématiques et plan projet	Samira/Alex/Elianne	13/07/2021
3	Réaliser une première itération de modèles à tester	Notebooks modèles SARIMA	Samira / Elianne	16/07/2021
4	Réaliser une deuxième itération de modèles à tester	Notebooks modèles SARIMA	Samira / Elianne	22/07/2021
5	Réaliser une troisième itération de modèles à tester	Notebooks modèles SARIMA	Samira / Elianne	29/07/2021
6	Recherche de data complémentaires	Fichiers xls et autres formats	Alexandre	29/07/2021
7	Mise en forme et preprocessing des data complémentaires (INSEE)	Fichiers csv	Alexandre	16/08/2021
8	Réaliser la Dataviz de la problématique 1	Notebook	Alexandre	20/08/2021
9	Finaliser la Dataviz de la problématique 1	Notebook	Alexandre	25/08/2021
10	Finaliser le modèle SARIMA avec la Pmax mensuelle	modèle SARIMA	Samira	23/08/2021
11	Réaliser le modèle Elastic Net avec la Pmax quotidienne	modèle Elastic Net	Alex	23/08/2021
12	Réaliser les modèles Ridge et Lasso avec la Pmax quotidienne	modèles Ridge et Lasso	Elianne	23/08/2021
13	Préparer la trame du rapport	Template Rapport prêt à remplir	Elianne	20/08/2021
14	Rédiger l'introduction sur le jeu de données et la partie technique	Chapitre introduction	Alexandre	24/08/2021
15	Rédiger le chapitre Dataviz Problématique 1	Chapitre DV Prob 1 rédigé	Alexandre	25/08/2021
16	Rédiger le chapitre Problématique 2 avec les modèles	Chapitre Modèles Prob 2 rédigé	Samira/Alex/Elianne	25/08/2021
17	Réaliser la dataviz du chapitre 3 sur les aspects production	Notebook jupyter dataviz Chapitre 3	Samira/Alex	23/08/2021
18	Rédiger le chapitre Problématique 3 sur les aspects production	Chapitre DV Prob 3 rédigé	Samira/Alex	25/08/2021
19	Rédiger la Conclusion	Chapitre conclusion	Samira/Alex/Elianne	25/08/2021
20	Relecture et correction du rapport	Rapport vérifié et prêt pour livraison	Samira/Alex/Elianne	26/08/2021
21	Verification et correction des fichiers de code	Fichiers code vérifiés et prêts pour livraison	Samira/Alex/Elianne	26/08/2021
22	Préparer la trame de la présentation soutenance	Template de la présentation	Samira	23/08/2021
23	Rédaction de la soutenance grâce aux éléments du rapport	Soutenance prête	Samira/Alex/Elianne	01/09/2021
24	Préparer le fichier spyder pour intégration streamlit	fichier prêt à intégrer	Elianne	28/08/2021
25	Integration du code dans le spyder	Fichier intégré	Elianne	30/08/2021
26	Vérification et ajustements du streamlit	Streamlit prêt pour présentation	Samira/Alex/Elianne	01/09/2021
27	Vérification et correction de la soutenance	présentation de la soutenance prête	Samira/Alex/Elianne	01/09/2021
28	Entrainement à la soutenance	Equipe prête pour soutenance	Samira/Alex/Elianne	01/09/2021

## B. Difficultés rencontrées

### **Conversion / Manipulation de format de date**

- Nous avons eu à traiter les data dans la mesure où nous avions des données horodatées toutes les 30 minutes. Il a fallu retraiter les dates dans leurs formats pour pouvoir travailler dessus.

### **Jeux de données complémentaires**

- Il nous a fallu ajouter d'autres jeux de données à notre étude pour que cela apporte un peu de complexité et de précision dans nos résultats. Leurs traitements a été plus fastidieux notamment avec un passage sur Access.

A travers les résultats de certains de nos modèles, nous avons compris qu'il nous aurait fallu des jeux de données plus complets et précis ou différents pour produire des prévisions de meilleure qualité.

### **Geopanda et données géospatiales**

- L'installation du module Geopandas a été impossible sur Anaconda même en créant un environnement neutre sur lequel l'installer.  
Il a fallu passer par google collab et chrome pour pouvoir l'installer et afficher des cartes choroplèthe. Notre fichier de données étant volumineux il fallait à chaque fois une trentaine de minutes pour le charger dans le collab.  
Peut-être qu'un module sur GeoPandas pourrait enrichir la formation et aider les prochains étudiants.

## C. Description des livrables – fichiers de code

### **Fichiers disponibles sur le Git**

Rapport projet : [\*Rapport projet Py-nergie bc\\_ds\\_juin-sept\\_2021 vfinale\*](#)

Notebook Dataviz problématique 1 et 3 : [\*Projet Py-nergie Dataviz finale\*](#)

Notebook Dataviz problématique 3 sous Google collab : [\*Carte\\_regions\\_final\\_v2\*](#)

Notebook modèle SARIMA (plus test modèle polynomial) : [\*Projet\\_Py\\_nergie\\_Sarimax final\*](#)  
*(Nous avons laissé dans ce notebook, un test sur une régression polynomiale que nous n'avons pas reporté dans le rapport car il nous aurait fallu plus de temps pour l'améliorer)*

Notebook sur les modèles Ridge et Lasso : [\*projet-nergie-final-ridge-lasso-newversion1\*](#)

Notebook sur le modèle Elastic Net : [\*Py-nergie Model Elastic Net v3 - MinMax scaler\*](#)  
*(pour info, Notebook sur le modèle Elastic Net avec Standard scaler dispo avec le nom : [\*Py-nergie Model Elastic Net v3 - Std scaler\*](#))*

## VIII. Conclusion

Le projet nous a permis de mettre en application les compétences acquises au cours de la formation autour d'un vrai sujet.

Notamment, la dataviz qui nous a permis de mettre en pratique les différentes librairies et même au-delà avec Geopandas.

Nous avons aussi pu améliorer nos connaissances en python en passant par toutes les étapes d'un projet de data scientist : la collecte des données, le nettoyage et le traitement, le choix des modèles, et enfin l'interprétation des résultats des différents algorithmes de Machine Learning.

Nous avons aussi découvert Github et Streamlit.

De plus, cela a été l'occasion de réaliser un travail de groupe où chacun a pu apporter son expertise et son expérience.

Enfin l'étude du sujet nous a apporté une connaissance approfondie dans le domaine de l'énergie en France, de l'équilibrage des réseaux électriques et des spécificités liées aux différents types d'énergies.

## **IX. ANNEXES**

## A. Diagramme de GANTT

## **PARTIE 1**

## PARTIE 2

## B. Bibliographie

### ***BIBLIOGRAPHIE SUJET TECHNIQUE RESEAU ELECTRIQUE***

- <https://www.cre.fr/Electricite/Reseaux-d-electricite/Presentation-des-reseaux-d-electricite>
- <http://modules-pedagogiques.cre.fr/m1/index2.html>
- <https://www.rte-france.com/chaque-seconde-courant-passe/equilibrer-loffre-et-la-demande-delectricite>
- <https://www.rte-france.com/eco2mix>
- <https://www.ecologie.gouv.fr/securite-dapprovisionnement-en-electricite>
- <https://www.rte-france.com/analyses-tendances-et-prospectives/les-bilans-previsionnels>
- <https://www.rte-france.com/analyses-tendances-et-prospectives/bilan-previsionnel-2050-futurs-energetiques>
- <https://www.energuide.be/fr/questions-reponses/pourquoi-le-reseau-electrique-doit-il-rester-en-equilibre/2136/>

### ***BIBLIOGRAPHIE ÉLABORATION RIDGE ET LASSO***

Pour comprendre les raisons pour lesquelles les modèles RIDGE et LASSO peuvent servir pour établir nos prédictions nous avons eu besoin de plusieurs documents de différentes sources.

Comprendre les principes de Régression Regularisée d'un point de vue mathématique :

- [https://eric.univ-lyon2.fr/~ricco/cours/slides/regularized\\_regression.pdf](https://eric.univ-lyon2.fr/~ricco/cours/slides/regularized_regression.pdf)
- <https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b>

### **APPLICATION :**

Exemple d'application des modèles RIDGE et LASSO sur des données et leur comparaison :

- <https://www.mygreatlearning.com/blog/understanding-of-lasso-regression/>
- <https://www.mygreatlearning.com/blog/what-is-ridge-regression/>
- <https://openclassrooms.com/fr/courses/4444646-entrainez-un-modele-predictif-lineaire/4507811-tp-comparez-le-comportement-du-lasso-et-de-la-regression-ridge>
- <https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/>
- <https://www.analyticsvidhya.com/blog/2016/01/ridge-lasso-regression-python-complete-tutorial/>