Projet de Modélisation Prédictive

Arthur Reidenbach et Raphael Bordas

Mars 2024

Sommaire

Code lié à ce projet disponible sur ce dépôt GitHub.

1 Motivation

Avec la part croissante des énergies renouvelables dans le mix électrique en France, il est indispensable de pouvoir prédire la demande nette de production électrique, c'est-à-dire la charge globale du réseau (`Load` dans le jeu de données) moins les productions renouvelables (`Solar_power` et `Wind_power`). L'augmentation de la demande brute avec les voitures électriques est également un enjeu majeur des futures prédictions de consommation, dont les caractéristiques majeurs vont ainsi être amenées à changer. Une rupture, dont l'importance est encore inconnue, est ainsi à prévoir à la fois sur les modalités de production et les habitudes de consommation. C'est dans ce cadre que le jeu de données fourni se propose de prédire la demande nette lors de la période de sobriété énergétique de septembre 2022 à octobre 2023, en apprenant sur les données de la dernière décennie (2013 - 2022).

2 Nettoyage des données et analyse exploratoire

2.1 Choix des données d'entraînement

Les données test ne comportant pas la demande nette, nous procédons par validation sur un sous-échantillon du jeu d'entraînement original (`Data0` dans le code). Plusieurs méthodes peuvent être envisagées :

1. Entraînement entre 2013 et 2021 inclus, validation entre le 01/01/2022 et le 01/09/2022.
2. Entraînement entre 2018 et 2021 inclus, validation entre le 01/01/2022 et le 01/09/2022.
3. Entraînement entre le 01/01/2018 et le 01/09/2022, validation sur 365 points aléatoirement choisis entre le 01/01/2021 et le 01/09/2022 et exclus de l'entraînement

La motivation pour entraîner les modèles à partir de 2018 (méthodes 2 et 3) vient essentiellement du changement de comportement de certaines séries temporelles : augmentation de la production éolienne observable sur la Figure 1 (panel de gauche), redéfinition du calcul de la nébulosité pondérée par la production (`Nebulosity_weighted`) à partir de 2018 (Figure 1, panel de droite). De manière générale, il a été observé de meilleures performances à modèles identiques lorsque l'entraînement commence en 2018.

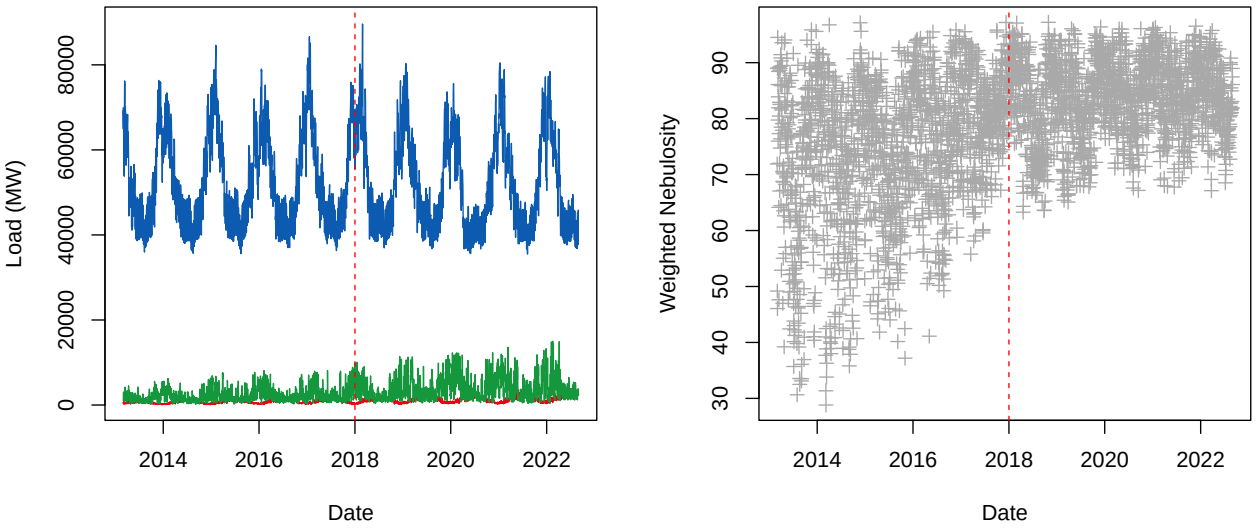

Figure 1: Evolution of some time series before and after 2018 (red line on Jan. 1st 2018)

De plus, il est également intéressant de prendre des points de validation aléatoire, pour éviter une évaluation biaisée de la performance : la méthode 2 n'évalue la performance d'un modèle que sur 3 saisons. Or, ainsi que le montre la Figure 2, la série temporelle d'intérêt présente une saisonnalité annuelle très marquée. Il est donc primordial d'évaluer également le modèle sur octobre-novembre-décembre.

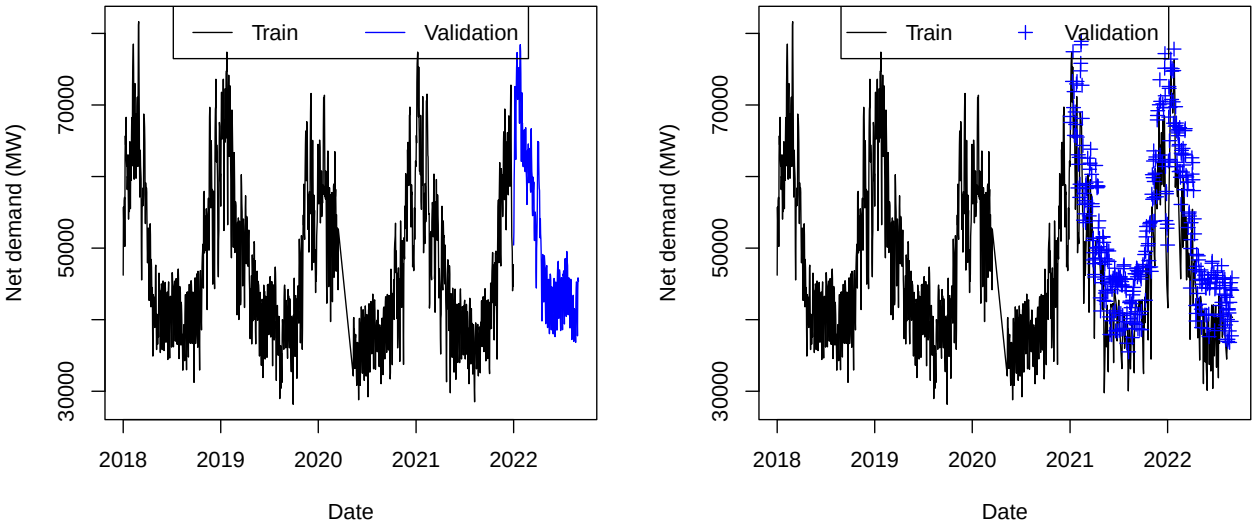

Figure 2: Two versions of the training and validation datasets

2.2 Données externes

La série temporelle d'intérêt a la particularité de couvrir toute la période de la pandémie du Covid-19 en 2020-2021. En effet, d'un point de vue descriptif, il est observé une baisse drastique et anormale de la

consommation (Figure 3). Deux solutions ont alors été testé pour prendre en compte ces déviations par rapport à la tendance avant et après le Covid :

- un codage *one-hot* des trois confinements: du 17/03/2020 au 11/05/2020, du 30/10/2020 au 15/12/2020 et du 03/04/2021 au 03/05/2021. 1 dénote alors la présence d'un confinement, 0 son absence.
- un indice de confinement (*strengency index*[1]) pour prendre en compte l'effet de la pandémie du Covid-19 sur la consommation électrique en France. Dans ce cadre, nous avons testé d'une part l'incorporation de cet indice comme covariable et d'autre part l'exclusion de l'entraînement des périodes avec un indice supérieur à 80 (correspondant au premier confinement du printemps 2020). C'est cette dernière solution qui a été retenu dans les modèles finaux (voir `train_data2` et `val_data2` dans le code).

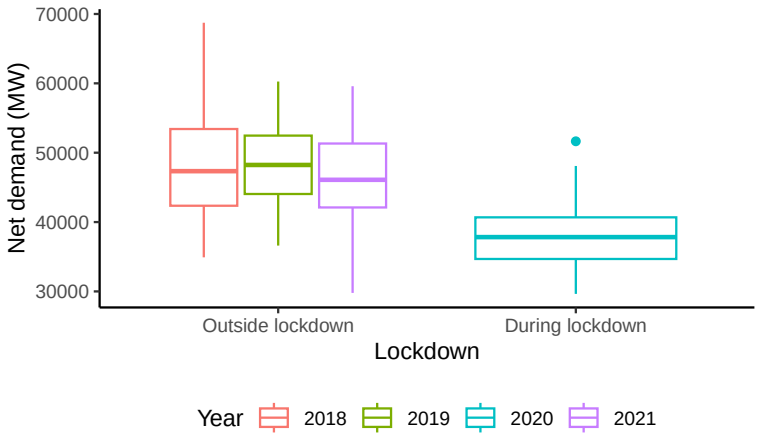

Figure 3: Impact of Covid over the net demand. Comparing time series between March 17th and May 11th of each year.

2.3 Sélection *a priori* de variables

La Figure 4 résume l'analyse exploratoire de quelques variables clés pour la prédiction de la demande nette : la charge du réseau du jour précédent (`Load.1`) semble être très fortement corrélée à la demande, mais être aussi dépendante du jour de la semaine (`WeekDays3`). La température (panel droit) semble avoir plutôt une relation polynomiale à la demande nette, selon le jour de l'année (`toy`). En ce qui concerne le jour de la semaine, nous avons travaillé avec 7 facteurs (`WeekDays`), dans les premiers modèles, avant de n'utiliser que `WeekDays3` qui regroupe les jours de la semaine avec un comportement de demande très similaire (mardi, mercredi et jeudi, noté `WorkDay`).

3 Modélisation

Nous ne reportons pas ici l'intégralité des modèles testés (voir les différents scripts sur le GitHub du projet pour cela), mais montrons plutôt les principales étapes ayant menées aux modèles finaux, ainsi que les points les plus pertinents.

3.1 Modèles linéaires

Pour montrer l'importance des variables a priori sélectionnées, nous avons commencé par des modèles linéaires simples, multiples et quantiles. Notre stratégie était alors la suivante :

1. Apprendre le modèle sur le jeu d'entrainement (`train_data` ou `train_data2` selon la méthode 2 ou 3 respectivement, voir section sur Choix des données d'entrainement)

[1]Hale, T., Angrist, N., Goldszmidt, R. et al. A global panel database of pandemic policies (Oxford COVID-19 Government Response Tracker). Nat Hum Behav 5, 529–538 (2021). https://doi.org/10.1038/s41562-021-01079-8

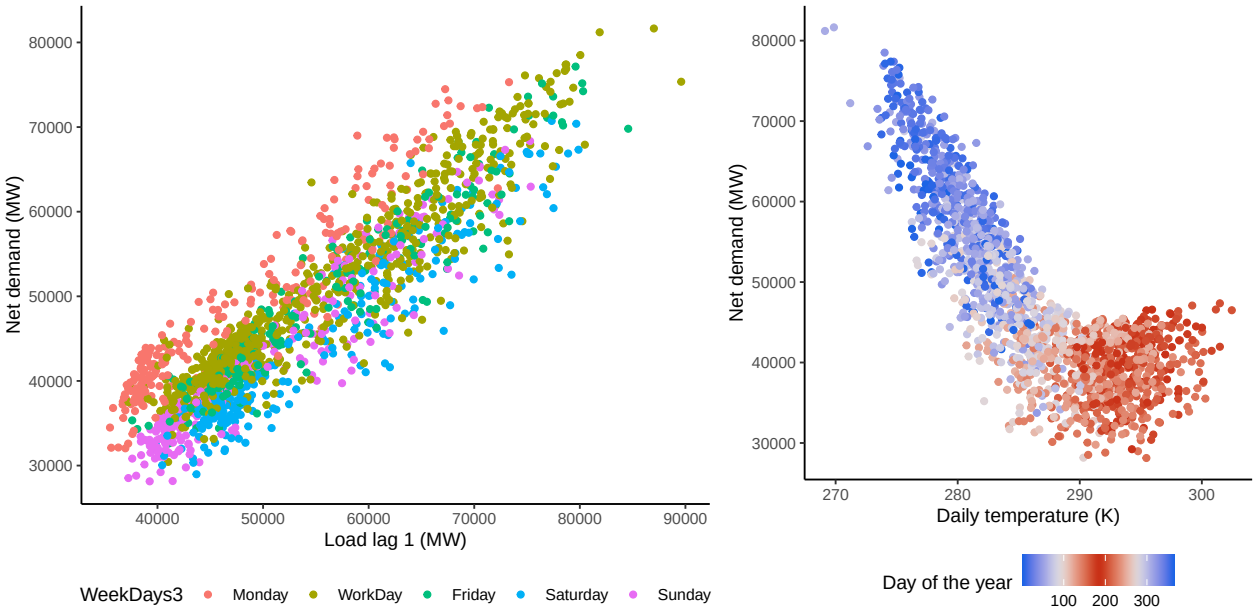

Figure 4: Relation between the net demand and some well selected variables. WeekDays3 denotes the grouping of Tues, Wed and Thur as WorkDay. In Day of the Year, 0 is Jan 1st, 365 Dec 31st.

2. Calculer l'erreur test (RMSE) sur le jeu de validation
3. Ajouter le quantile 0.95 des résidus aux prédictions
4. Calculer la pinball loss sur le jeu de validation

Si le modèle est performant :

4. Validation croisée 8-plis sur la concaténation entrainement-validation pour prédire `Data1`
5. Répéter 3 et 4 sur les prédictions de la validation croisée
6. Ajouter le quantile 0.95 calculé sur les résidus aux prédictions du test
7. Soumission Kaggle si le modèle était jugé suffisament performant

3.1.1 Importance du jour de la semaine et de la température

Comme mis en évidence sur la figure 4, la relation entre Température et Demande nette n'est pas strictement linéaire. Pour prendre en compte ce phénomène, il a d'abord été introduit des polynômes tronqués à Temp = 280 et Temp = 290, menant à ce premier modèle de référence :

```
mod1 <- lm(Net_demand ~ WeekDays3 + Temp + Temp_trunc1 + Temp_trunc2 + stringency_index,
           data=Data0[sel_a,])
```

Les résidus sont supposés suivre une distribution normale, et il est donc possible d'ajouter le quantile 0.95 à la prédiction :

```
res <- Data0[sel_a,]$Net_demand - mod1$fitted.values
quant <- qnorm(0.95, mean = mean(res), sd = sd(res))
mod1.forecast <- mod1.forecast + quant
```

Ce type de premiers modèles produit une pinball loss autour de 400 (tableau 1) et n'a donc pas été soumis sur Kaggle.

3.1.2 Prendre en compte la saisonnalité

Notre première tentative pour prendre en compte la saisonnalité des données, à la fois mensuelle et hebdomadaire, a reposé sur une analyse de Fourier. Pour mettre en évidence les saisonnalités, on peut à la fois regarder l'auto-corrélation de la série temporelle demande nette et le periodigramme (Figure 5) : la demande est bien dépendante du jour de la semaine et sur le périodigramme on retrouve à la fois les saisonnalités annuelles (premier pic) et hebdomadaires (second pic et ses harmoniques).

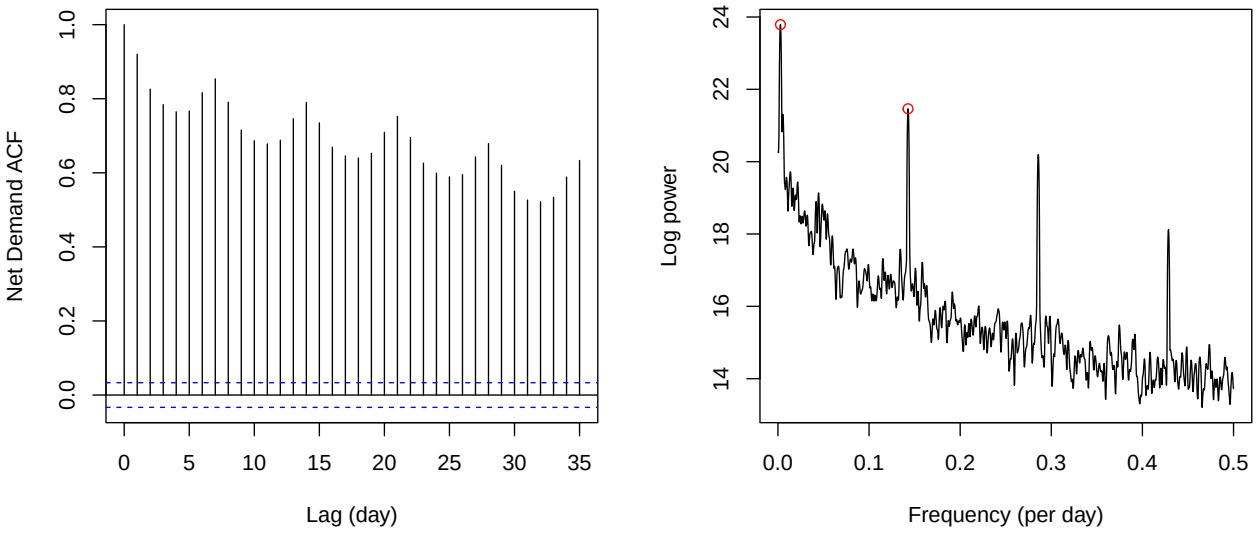

Figure 5: Auto-correlation of the net demand (left) and its periodigram (right). On the right panel, spectral peaks are located at $f = 0.0028$ and $f = 0.1427$, corresponding respectively to annual and weekly frequency (sampling frequency is 1/day)

```
mod1 <- lm(
    Net_demand ~ WeekDays3 + Temp + Temp_trunc1 + Temp_trunc2  +
      stringency_index +
      cos1 + cos2 + cos3 + cos4 + cos5 + cos6 + cos7 + cos8 +
      sin1 + sin2 + sin3 + sin4 + sin5 + sin6 + sin7 + sin8,
    data = Data0[sel_a,]
  )
```

Avec $\sin_i = \sin(\omega t i)$, $\omega = \frac{2\pi}{365}$ et t la date numérique au format numérique. Pour choisir i, on minimise l'erreur de prédiction (RMSE) de modèles emboités comportant de plus en plus de cycles :

$$i = \text{argmin}_{j\in\{1\ldots50\}}\text{RMSE}(\text{lm}(\cos_1 + \cdots + \cos_j + \sin_1 + \cdots + \sin_j))$$

3.1.3 Régression quantile

Dans ce data challenge, nous cherchons à prédire le quantile 0.95, il est donc intéressant de comparer la performance entre les modèles précédents et une prédiction plus directe des quantiles :

```
mod.rq <- rq(Net_demand ~ WeekDays + Temp + Temp_trunc1 + Temp_trunc2 + stringency_index,
             data = train_data,
             tau = 0.95)
```

3.1.4 Résultats des modèles linéaires

Les performances des modèles sont résumées dans le tableau 1. Tous les modèles linéaires de cette section ont été entrainé sur `Data0`.

Table 1: Results of the linear models

Model	RMSE (Validation)	Pinball loss (Validation)
Linear	3896	391
Cyclic	3196	340
Quantile	NA	322

Ainsi la prise en compte des saisonnalités est importante pour améliorer les performances du modèle, et un travail sur les quantiles indispensables pour faire baisser la pinball loss. De ce point de vue-là, nous n'avons pas pu descendre en dessous de 300 sur le jeu de validation en utilisant uniquement des modèles linéaires. Etant donné la performance des GAM décris ci-dessous, aucun des trois types de modèles linéaires n'a été soumis sur Kaggle.

3.2 GAM

3.2.1 Modèle de départ

Le travail réalisé sur les modèles linéaires, avec d'une part les tendances polynomiales prises en compte par des polynômes tronqués et d'autre part les effets de saisonnalités (cycles), suggère qu'un modèle additif généralisé (GAM) serait plus performant. En couplant cette nouvelle modélisation à d'autres facteurs météorologiques (le vent et la nébulosité), la performance de prédiction a été grandement accru.

Nous retenons pour cela toutes les variables signiticatives des modèles linéaires. Nous ajoutons également les données solaires et de l'élolien. Le choix d'une base cyclique pour la variable `toy` s'explique par son cycle annuelle : elle varie entre 0 et 1 chaque année. Le modèle de départ était donc le suivant :

```
equation <- Net_demand ~ WeekDays +
  s(Temp, k = 10, bs = "cr") +
  s(toy, k = 30, bs = "cc") +
  s(Load.1, bs = 'cr', k = 15) +
  s(Load.7, bs = 'cr') +
  s(Wind_weighted, k = 10, bs = "cr") +
  te(as.numeric(Date), Nebulosity_weighted, k = c(4, 15)) +
  s(Temp_s99, k = 10, bs = 'cr') +
  s(Solar_power.1, bs = 'cr', k = 5) +
  s(Wind_power.1, bs = 'cr', k = 5)
md.gam.1 <- gam(equation, data = train_data)
```

3.2.2 Déterminer les noeuds

Pour déterminer les nœuds, nous regardons le résumé d'un GAM : si l'`edf` est proche de la valeur k spécifiée dans l'équation, alors nous augmentons la dimension de la base jusqu'à ce l'`edf` n'augmente plus. Cette méthode, bien que très empirique, a permis d'améliorer le modèle, en s'écartant bien souvent de la valeur par défaut $k = 10$.

3.2.3 Déterminer les intéractions

Pour s'assurer de la pertinence du choix des interactions dans les GAM, un test ANOVA est réalisé. Par exemple, montrons l'intérêt d'ajouter une interaction de la date avec le vent au modèle précédent. C'est-à-dire utiliser `te(as.numeric(Date), Wind_weighted, k = c(4, 10))` au lieu de `s(Wind_weighted, k = 10, bs = "cr")`.

Nous comparons pour cela les modèles construits avec les équations suivantes :

```
equation.1 <- Net_demand ~ WeekDays +
  s(Temp, k = 10, bs="cr") +
```

Table 2: Comparing GAM models with and without interaction for the weighted wind. Equation are defined in the attached R code snippet in the text.

Equation	sqrt(GCV) score	Residual Deviance
Equation 1	1314.424	2570086066
Equation 2	1287.628	2449799066

```
  s(toy, k = 30, bs="cc")+
  s(Load.1, bs='cr', k = 15) +
  s(Load.7, bs='cr') +
  # no interaction between Wind and Time :
  s(Wind_weighted, k = 10, bs = "cr") +
  s(Time, k = 4, bs = "cr")  +
  te(Time, Nebulosity_weighted, k=c(4,10)) +
  s(Temp_s99,k=10, bs='cr') +
  s(Solar_power.1, bs='cr', k = 5) +
  s(Wind_power.1, bs='cr', k = 5)

equation.2 <- Net_demand ~ WeekDays +
  s(Temp, k = 10, bs="cr") +
  s(toy, k = 30, bs="cc")+
  s(Load.1, bs='cr', k = 15) +
  s(Load.7, bs='cr') +
  # interaction between Wind and Time through a smooth tensor product
  te(Time, Wind_weighted, k=c(4,10)) +
  te(Time, Nebulosity_weighted, k=c(4,10)) +
  s(Temp_s99,k=10, bs='cr') +
  s(Solar_power.1, bs='cr', k = 5) +
  s(Wind_power.1, bs='cr', k = 5)
```

Le tableau 2 résume les indicateurs de comparaison : l'introduction de l'interaction fait baisser le GCV et la déviance résiduelle. De plus un test du χ^2 sur la déviance a une p-value inférieure à 1.10^{-12}.

3.2.4 Ajustement des résidus pour la prédiction du quantile 0.95

De la même façon que pour les modèles linéaires, le quantile 0.95 des résidus du modèle est ajouté à la prédiction. Pour autant, certains modèles (dont celui de la sous-section suivante) ont des scores plus faibles sur le jeu de données privé du Kaggle sans cette correction.

A noter si la distribution des résidus semble normal, ainsi que montré sur la figure 6, on observe également les valeurs extrêmes s'écartant significativement d'une distribution théorique. Or il s'agit ici de prédire le quantile 0.95. Un travail supplémentaire sur les résidus est donc pertinent.

Pour essayer de comprendre ce phénomène, au lieu de modéliser le quantile 0.95 par :

```
m <- mean(gam$residuals),
s <- sd(gam$residuals)
quant <- qnorm(0.95, mean = m, sd = s)
```

nous avons également utilisé une régression quantile linéaire sur les résidus :

```
mod.rq <- rq(
  residuals ~ Date + toy + Temp +
    BH + Load.1 + Load.7 + Temp_s99 +
    WeekDays + Wind_weighted + Nebulosity_weighted +
```

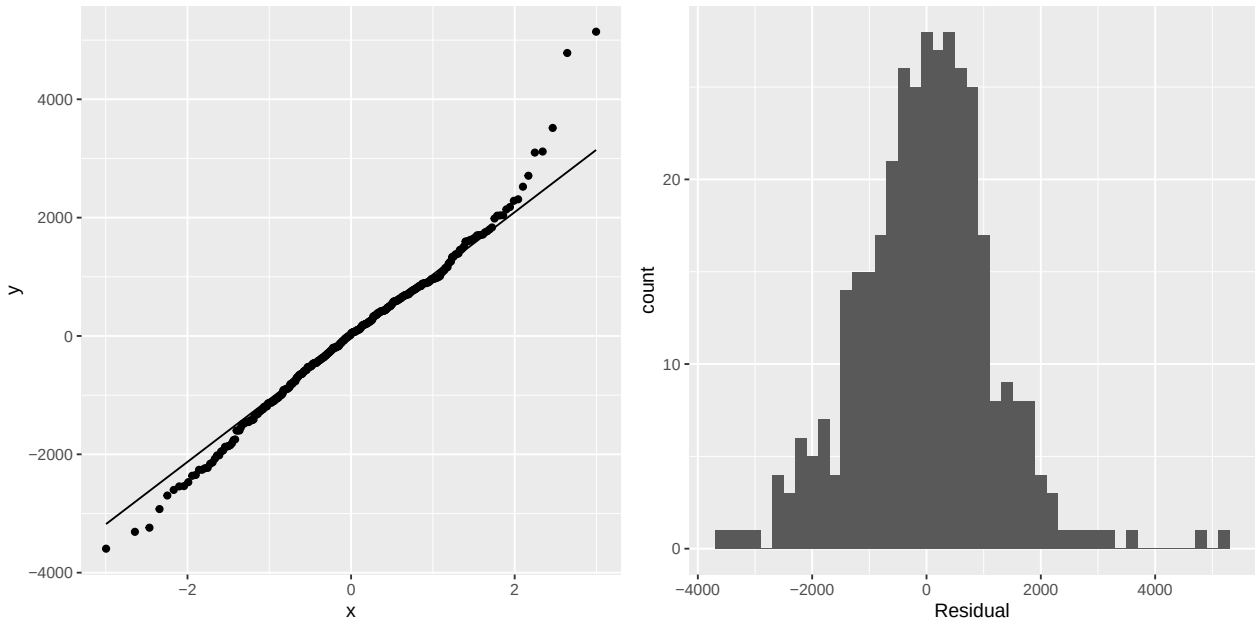

Figure 6: Residual distribution of a typical GAM used in the project.

```
    Date + Wind_power.1 + Solar_power.1,
  data = quantile_data,
  tau = 0.95
)
quantnew <- predict(mod.rq, newdata = quantile_newdata)
```

Cependant, avec l'inclusion de certaines variables, la matrice de design peut devenir singulière. Il n'est pas possible de construire de cette manière-là le meilleur modèle GAM trouvé et de comparer sa performance sans correction des résidus et une correction par régression quantile.

Dans cet esprit, nous avons donc testé une modélisation directe du quantile 0.95 à l'aide de modèle GAM-quantile (package `qgam`) en reprenant les équations identiques aux modèles GAM classiques.

3.2.5 Résultats des GAM

Avant de montrer le meilleur modèle, montrons l'intérêt des diverses investigations sur les résidus. En reprenant l'équation suivante :

```
Net_demand ~ WeekDays +
  s(Temp, k = 10, bs = "cr") +
  s(toy, k = 30, bs = "cc") +
  s(Load.1, bs = 'cr', k = 15) +
  s(Load.7, bs = 'cr') +
  s(Wind_weighted, k = 10, bs = "cr") +
  te(as.numeric(Date), Nebulosity_weighted, k = c(4, 15)) +
  s(Temp_s99, k = 10, bs = 'cr') +
  s(Solar_power.1, bs = 'cr', k = 5) +
  s(Wind_power.1, bs = 'cr', k = 5)
```

```
## Net_demand ~ WeekDays + s(Temp, k = 10, bs = "cr") + s(toy, k = 30,
##     bs = "cc") + s(Load.1, bs = "cr", k = 15) + s(Load.7, bs = "cr") +
##     s(Wind_weighted, k = 10, bs = "cr") + te(as.numeric(Date),
```

Table 3: Results of some representative GAM models. The same equation is used for the 4 models (see main text)

Model	sqrt(GCV)	Pinball loss (Validation)	Kaggle (private set)
GAM no quant	1055.3	442	139
GAM qnorm	1055.3	132	559
GAM rq	1055.3	107	184
Q-GAM	103.0	105	204

```
##     Nebulosity_weighted, k = c(4, 15)) + s(Temp_s99, k = 10,
##     bs = "cr") + s(Solar_power.1, bs = "cr", k = 5) + s(Wind_power.1,
##     bs = "cr", k = 5)
```

Il vient les résultats du tableau 3, montrant la performance des GAM sans correction de quantile pour les soumissions Kaggle[2].

Pour conclure sur cette section, nous rapportons ici le meilleur modèle GAM obtenu lors du data challenge (score de 122 sur le jeu de données privé sur Kaggle) :

3.3 Aggrégation d'experts

Notre dernière démarche, à la suite de ce travail sur les GAM, a été de chercher à agréger les prédictions de divers experts : GAM et boosting. Ainsi que discuté plus haut, des GAM aux équations légèrement différentes peuvent donner des performances similaires. La question est alors de savoir si une agrégation de GAM, couplée à des modèles de régression boosté (package `gbm`) peuvent encore accroître ces performances.

Notre première agrégation reprenait donc des GAM étudié en amont (notamment pour le comportement des résidus) et un modèle boosté comme suit :

```
gbm1 <- gbm(
  equation,
  distribution = "gaussian" ,
  data = Data0[sel_a,],
  n.trees = Ntree,
  interaction.depth = 10,
  n.minobsinnode = 5,
  shrinkage = 0.02,
  bag.fraction = 0.5,
  train.fraction = 1,
  keep.data = FALSE,
  n.cores = 8
)
```

Le package `operaa` ensuite été utilisé pour agréger les prédictions selon une fonction de perte MSE et une combinaison convexe :

```
opera::oracle(Data0$Net_demand[sel_b],
              experts,
              model = "convex",
              loss.type = "square")
```

```
Call:
oracle.default(Y = Data0$Net_demand[sel_b], experts = experts,
```

[2]Ces résultats sont ici de re-soumission tardive pour s'assurer d'avoir des modèles comparables avec des équations strictement identiques et entraînement sur la même version du jeu de données (voir discussion sur le Choix des données d'entraînement). Il peut donc y avoir des différences de scores.

```
    model = "convex", loss.type = "square")

Coefficients:
   gbm gam1  gam2     gam3
 0.193 0.24 0.567 4.95e-21

                      rmse   mape
Best expert oracle:   1320 0.0208
Uniform combination:  1270 0.0203
Best convex oracle:   1230 0.0201
```

A partir de ces résultats, il peut être judicieux de ne choisir que les gam, qui regroupent plus de 80% du poids des prédictions. En agrégation ainsi `gam1` et `gam2` nous obtenons notre deuxième meilleure prédiction sur le jeu de données privé de Kaggle (score de 132).

4 Discussion

A partir d'une sélection restrainte de variables météorologiques (température, vent, nébulosité), temporelles (dates) et de production en décalage (notamment la veille), nous avons cherché à prédire la demande nette d'électricité en France, de manière journalière. De tous les modèles testés, les GAM se sont révélés les plus performants, se généralisant bien sur le jeu de données privé Kaggle. Si le choix des variables explicatives est déterminant, il ne faut pas négliger la sélection des données d'entrainement et une refléxion sur la gestion des résidus et leur distribution. Ces deux derniers éléments, une fois les bonnes variables identifiées, semblent en effet être la clé pour améliorer la performance des prédictions journalières.

En ce qui concerne les perspectives de ce projet de modéliser, il serait intéresser d'étudier d'autres distributions que la distribution normale. En effet, tous les modèles testés reposaient sur l'hypothèse gaussienne. En ayant en tête la queue de distribution des résidus plus ou moins longue selon les modèles, une distribution gamme peut par exemple être pertinente.

Enfin, les processus auto-regressif[3] pourrait être un deuxième élement d'amélioration des prévisions de la demande nette, toujours dans cette perspective de mieux prendre en compte les spécificités de la distribution des résidus des modèles GAM.

[3] une première tentative est décrite à la fin du script `_03_gam.R` mais n'a pas été suffisament concluante pour être reportée ici ou faire l'object d'une soumission Kaggle (en terme de qualité prédictive par rapport aux meilleurs gam sans autorégression : pinball loss de 204 contre 146 respectivement, sur le jeu de données `val_data2`).