

Modélisation Prédictive Rapport

Valentin Gözl, Laura Fuentes

February 28, 2023

Contents

Introduction	1
Mise en place du jeu de données	1
Choix du type de modèle	2
Linear models	3
Modèles additifs généralisés	4
Gam Check	5
ARIMA	7
Filtre Kalman	8
Pipeline basée sur le modèle qgam	9
Agrégation d'experts	9
Conclusion	10

Introduction

Dans ce projet, nous avons considéré un jeu de données regroupant différentes variables en rapport avec la consommation énergétique en France pendant la période de 2012 à 2020. Notre but est de construire un modèle qui permet de prédire la consommation française en énergie pendant la période du Covid. Pour cela, nous effectuerons un choix de variables, et complexifierons au fur et à mesure l'équation et le modèle choisi. Enfin, nous mettrons en place une pipeline et agrégation d'experts pour aboutir à un modèle plus robuste et performant.

Mise en place du jeu de données

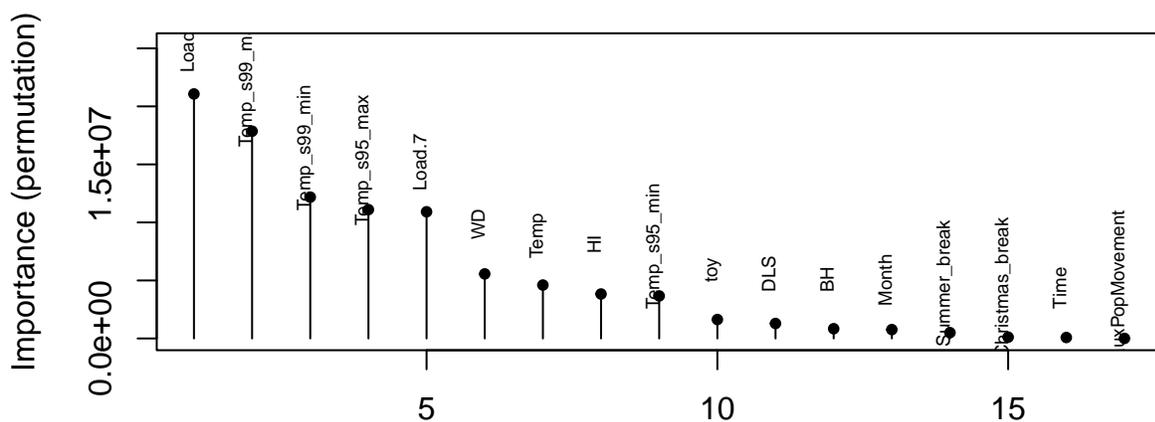
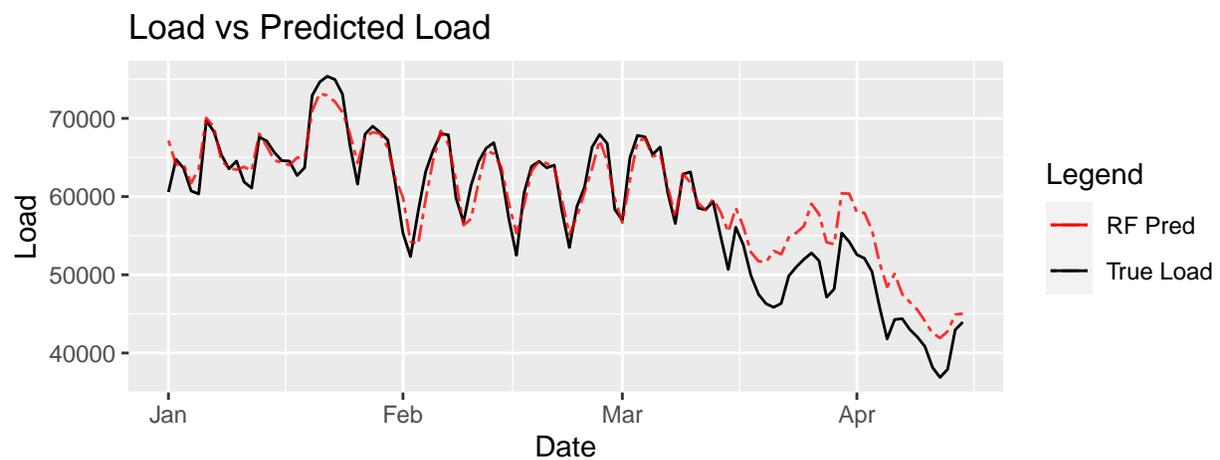
Le premier réflexe est de télécharger l'ensemble des packages et diviser le set train en deux pour pouvoir tester nos modèles avant de les soumettre. Nous avons ainsi choisi la période de 2012 - 2019 comme train et 2019-(15/04/2020) comme test.

Choix du type de modèle

Pour commencer, nous avons créé de nouvelles variables à partir de variables déjà existantes. C'est le cas de *WeekDays2*, *GRI_factor* et *WD*. Nous avons d'autre part récupéré des données des différents mouvements sociaux et le pourcentage de population mobilisée. La variable *Movement* prend des données de la SNCF et met les jours de grève à 1 et 0 sinon. Ensuite, nous avons créé une variable mesurant la température ressentie. Le problème de cette dernière concernait les nombreuses valeurs NA's, ainsi que la représentativité au niveau national des stations météorologiques constituant les données. C'est pour cela que nous ne l'avons pas utilisé au final.

Pour comprendre quelles variables sont plus significatives et argumenter le choix, nous avons effectué une forêt aléatoire, et regardé l'importance des variables. On a aussi vu que la RMSE associée n'est pas très bonne.

```
## [1] 3140
```



Nous pouvons ainsi bien remarquer que les variables à plus forte importance sont: *Load.1*, *Load.7*, les variables relatives à la température, *WeekDays*, *WD*, *BH*, *toy*, *Summer_break*, *DLS* and *Christmas_break*.

Nous avons également vérifié que la variable *Movement* (taux de la population participant à la manifestation) n'était pas explicative à l'aide de ANOVA.

```
##      Res.Df      RSS      Df      Sum of Sq
## Min.    :2931  Min.    :3.954e+10  Min.    :1  Min.    :4987977
## 1st Qu.:2931  1st Qu.:3.954e+10  1st Qu.:1  1st Qu.:4987977
## Median :2932  Median :3.954e+10  Median :1  Median :4987977
## Mean   :2932  Mean   :3.954e+10  Mean   :1  Mean   :4987977
## 3rd Qu.:2932  3rd Qu.:3.954e+10  3rd Qu.:1  3rd Qu.:4987977
## Max.   :2932  Max.   :3.954e+10  Max.   :1  Max.   :4987977
##
##      F      Pr(>F)
## Min.    :0.3698  Min.    :0.5432
## 1st Qu.:0.3698  1st Qu.:0.5432
## Median :0.3698  Median :0.5432
## Mean   :0.3698  Mean   :0.5432
## 3rd Qu.:0.3698  3rd Qu.:0.5432
## Max.   :0.3698  Max.   :0.5432
## NA's    :1      NA's    :1
```

Linear models

Simple linear model

Pour comprendre et appréhender le cadre de l'étude, nous commencerons par effectuer un modèle simple. C'est-à-dire un modèle linéaire avec les co-variables choisies précédemment. Nous avons considéré que la consommation de la veille changeait en fonction du jour de la semaine. C'est pour cela que nous avons décidé de créer une fonction de la consommation de la veille en fonction de chaque catégorie de *WeekDays*.

```
formula <- "Load ~Load.1 + WeekDays + Load.7 + Temp + Temp_s99_max +
  Temp_s99_min + WeekDays + WD + BH+ toy + Summer_break + DLS +
  Christmas_break + HI"
slm <- lm(formula)%>%as.formula, data=Data0)
pred1 = predict(slm, newdata=Data1)
rmse(Data1$Load, pred1)
```

```
## [1] 2153
```

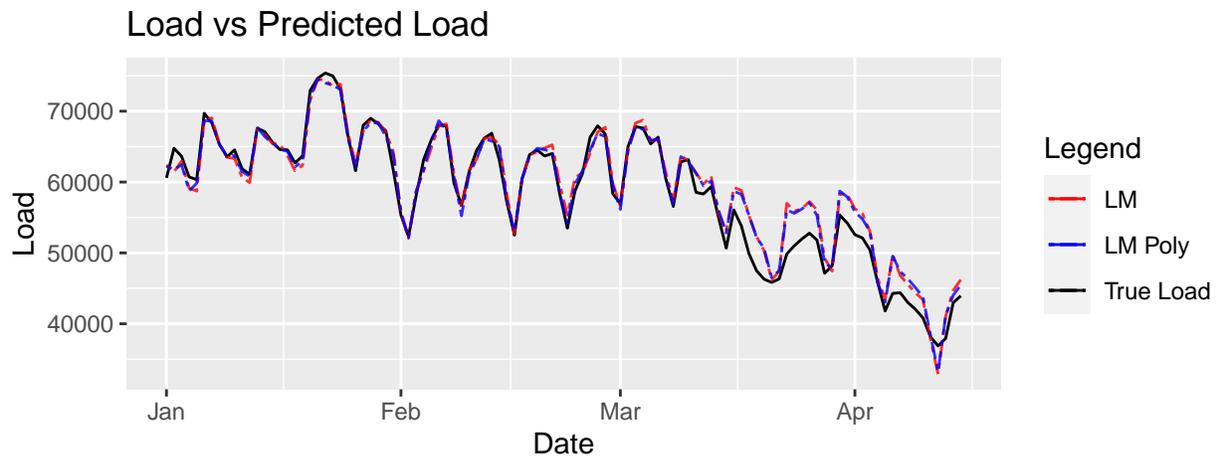
Polynomial transformations

Nous avons mis en place des transformations polynomiales sur le modèle linéaire comme une première approche de complexification du modèle. Pour cela, nous avons ajouté les co-variables relatives à la température et *Load.7* au carré.

```
formula <- "Load ~Load.1 + WeekDays + Load.7 + I(Load.7^2) + Temp + I(Temp^2) +
  Temp_s99_max + I(Temp_s99_max^2) + Temp_s99_min + WeekDays + WD + BH+ toy +
  Summer_break + DLS + Christmas_break + HI "
slm <- lm(formula)%>%as.formula, data=Data0)
pred2 = predict(slm, newdata=Data1)
rmse(Data1$Load, pred2)
```

```
## [1] 2000
```

Dans cette figure, nous observons que le modèle linéaire polynomial a une meilleure performance en début janvier et au niveau des pics pendant le Covid. Ces résultats se voient aussi au niveau du RMSE.



Modèles additifs généralisés

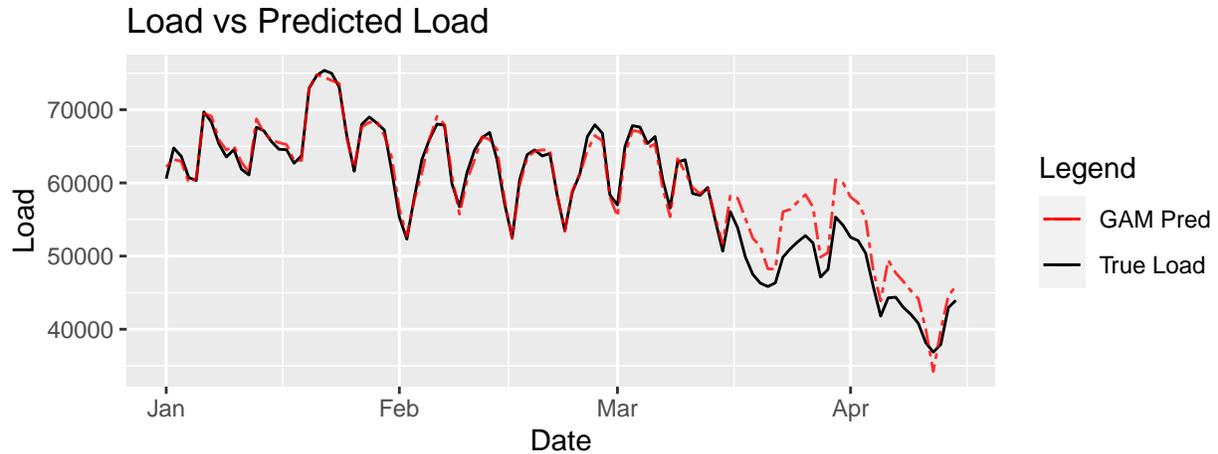
Choix de la partie linéaire et spline

Dans la suite, nous avons considéré la mise en place des modèles additifs généralisés (GAM). Pour cela, nous avons d'abord modifié notre équation. Nous avons distingué les variables à mettre dans la partie linéaire du modèle, puis dans la partie spline. Nous avons ainsi intégré les variables qualitatives ainsi que la consommation de la veille en fonction du jour de la semaine dans la partie linéaire. Dans la partie spline, nous avons ajouté les variables ayant une notion de temporalité comme *Load.7* ou les températures. Nous avons également regroupé dans une même spline des variables ayant une relation logique, comme c'était le cas de la température et le temps. Enfin, nous avons également créé une fonction spline pour chaque jour de la semaine pour la variable *toy* pour ne pas négliger l'effet des jours de la semaine sur la consommation annuelle.

```
equation <- "Load ~ Load.1:as.factor(WeekDays) + HI + BH + Christmas_break +  
  Summer_break + DLS + s(Temp) + s(Temp_s99_max, Temp_s99_min)+ s(Load.7) +  
  s(Time, k=7) + s(toy, k =30, bs = 'cc', by=as.factor(WD))+ s(Temp, Time, k=20)"  
  
gam<-gam(equation%>%as.formula, data=Data0)  
gam.forecast <- predict(gam, newdata=Data1)  
rmse(Data1$Load, gam.forecast)
```

```
## [1] 2316
```

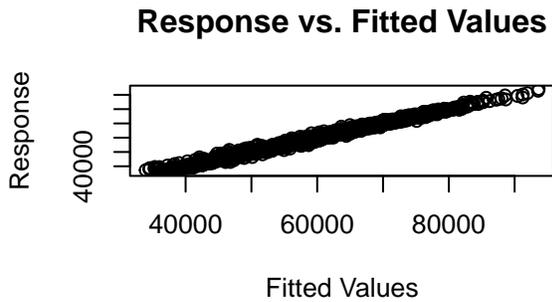
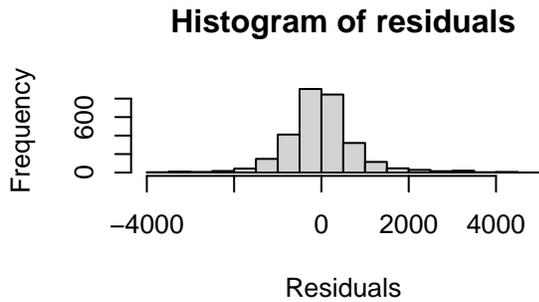
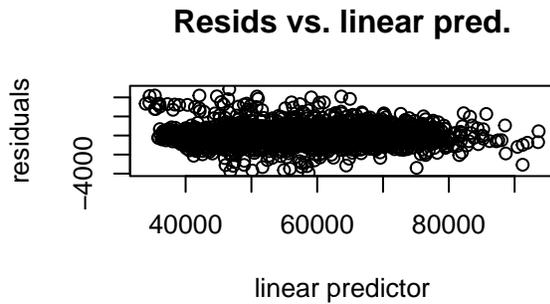
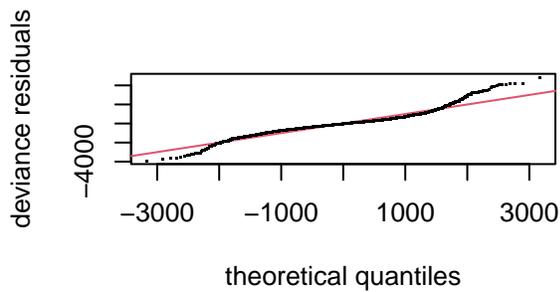
Même si la RMSE du validation set de GAM est moins bonne que celle des modèles linéaires, les résultats sur le test set diffèrent.



Pour améliorer le rendement du modèle, nous avons tenté de comprendre l'origine des erreurs à partir des courbes de consommation. Nous avons constaté que les erreurs commencent à s'accroître au niveau du mois de mars 2020, correspondant au début de la période Covid. Ceci s'explique du fait que la variable *GouvernementResponseIndex* comprend des valeurs nulles pendant des années, et celles-ci explosent dans une courte période d'un mois, laissant peu de temps d'entraînement sur la pandémie. Pour simuler le comportement de la population pendant le confinement avec les données que l'on avait déjà, nous avons pensé aux samedis. En effet, nous avons émis l'hypothèse qu'un jour de confinement était comparable en matière de consommation à un jour de weekend comme un samedi. Dans cet esprit, nous avons créé la variable *WD*, qui modifie le jour de la semaine à samedi s'il y a confinement (la *GouvernementResponseIndex* ≥ 70), et maintient les jours de la semaine inchangés sinon. Une autre idée serait d'utiliser l'apprentissage en ligne, un concept que nous avons utilisé par la suite. L'apprentissage en ligne nous permet de prendre en compte des données d'entraînement de l'époque contemporaine avec des mises à jour.

Gam Check

Nous avons également utilisé la fonction `gam.check` pour améliorer le rendement du modèle `gam`. Celle-ci nous a permis d'ajuster la dimension des bases des splines. Nous avons ainsi incrémenté les valeurs de `k` quand la `p`-valeur était très petite. Pour la variable *toy*, utiliser un `k` très grand faisait tourner le modèle trop longtemps et risquait d'overfitter. Nous avons donc pris `k=30` même si la `p`-valeur était encore petite. Enfin, nous avons également vérifié que les résidus étaient bien gaussiens à chaque fois à partir de l'histogramme issu du plot. Nous pouvons voir que les résidus sont en fait distribués de manière gaussienne. Le QQ-plot n'est pas une ligne parfaite, ce qui montre que le modèle a encore un certain potentiel pour s'améliorer.



```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 13 iterations.
## The RMS GCV score gradient at convergence was 0.7674903 .
## The Hessian was positive definite.
## Model rank = 278 / 279
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##
##          k'   edf k-index p-value
## s(Temp)          9.00  6.33   1.01   0.69
## s(Temp_s99_max,Temp_s99_min) 29.00 23.96   1.02   0.82
## s(Load.7)         9.00  4.89   0.97   0.03 *
## s(Time)           6.00  1.00   0.84 <2e-16 ***
## s(toy):as.factor(WD)Friday    28.00 17.48   0.83 <2e-16 ***
## s(toy):as.factor(WD)Monday    28.00 24.04   0.83 <2e-16 ***
## s(toy):as.factor(WD)Saturday  28.00 20.73   0.83 <2e-16 ***
## s(toy):as.factor(WD)Sunday    28.00 24.55   0.83 <2e-16 ***
## s(toy):as.factor(WD)Thursday  28.00 21.27   0.83 <2e-16 ***
## s(toy):as.factor(WD)Tuesday   28.00 22.68   0.83 <2e-16 ***
## s(toy):as.factor(WD)Wednesday 28.00 22.30   0.83 <2e-16 ***
## s(Temp,Time)        17.00 11.24   0.86 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

GAM et régression quantile: qgam

Dans la suite nous utiliserons le package `qgam`, et en particulier la fonction `qgam`. Celle-ci ajuste un modèle additif ainsi qu'une régression quantile sur un unique quantile. On utilise ici quasiment la même équation qu'auparavant, il suffit juste d'ajuster la variable "qu", correspondant au quantile. Après plusieurs essais, nous avons remarqué qu'on obtient de meilleurs résultats pour un 'qu' autour de 0.4. En effet, en fixant le quantile à 0.4, on change la fonction de perte. Nous introduisons ainsi un biais permettant de mieux s'ajuster aux données lors de la période du Covid.

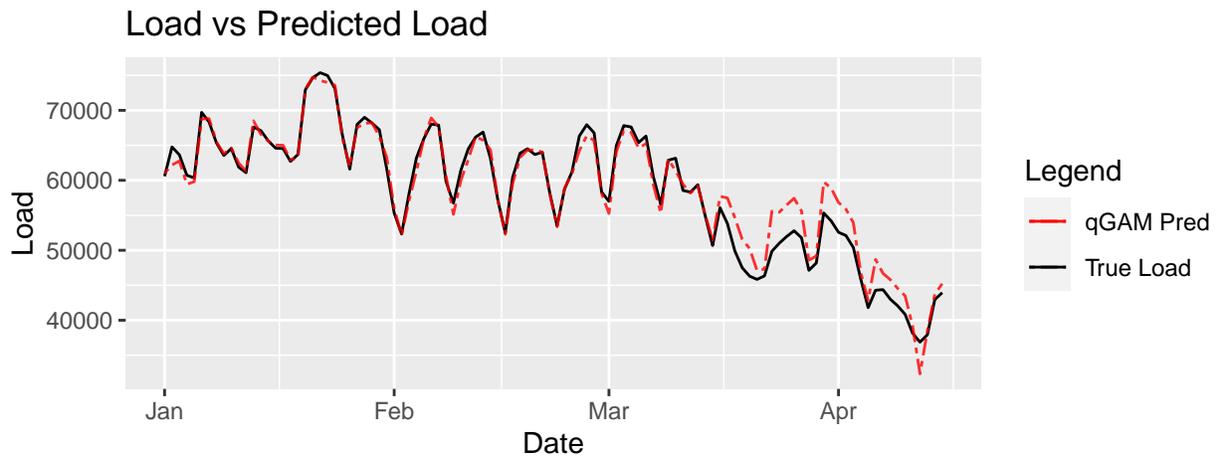
```
gam9<-qgam(equation%>%as.formula, data=Data0, qu=0.4)
```

```
## Estimating learning rate. Each dot corresponds to a loss evaluation.  
## qu = 0.4.....done
```

```
gam9.forecast <- predict(gam9, newdata=Data1)  
rmse(Data1$Load, gam9.forecast)
```

```
## [1] 1955
```

Nous pouvons observer que la prédiction issue du `qgam` s'améliore nettement pendant la période Covid par rapport aux autres modèles. Cette observation est aussi remarquable au niveau du RMSE, qui diminue de 300 par rapport au résultat de GAM.



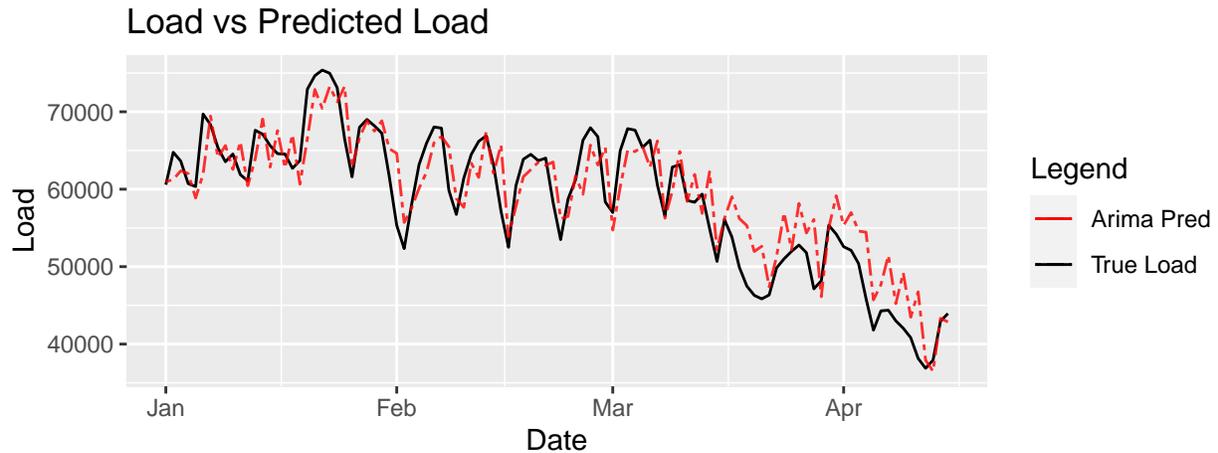
ARIMA

ARIMA (Autoregressive Integrated Moving Average) est un modèle de série chronologique qui utilise les valeurs et les erreurs passées pour prévoir les valeurs futures d'une série. Le modèle peut être ajusté pour capturer les tendances et la saisonnalité, et implique une différenciation des données pour les rendre stationnaires. Le modèle ARIMA est largement utilisé pour faire des prévisions sur la base de tendances historiques.

```
arima.fit <- forecast::Arima(gam9.forecast,  
                             order = c(1,1,2), seasonal = c(0,0,2))  
arima.predict <- fitted(arima.fit)  
rmse(arima.predict, Data1$Load)
```

```
## [1] 3969
```

Le résultat n'est pas bon. En revanche, on constate que la nature des prédictions est bien différente par rapport aux autres modèles. Ceci sera donc intéressant à considérer pour l'agrégation d'experts.



Filtre Kalman

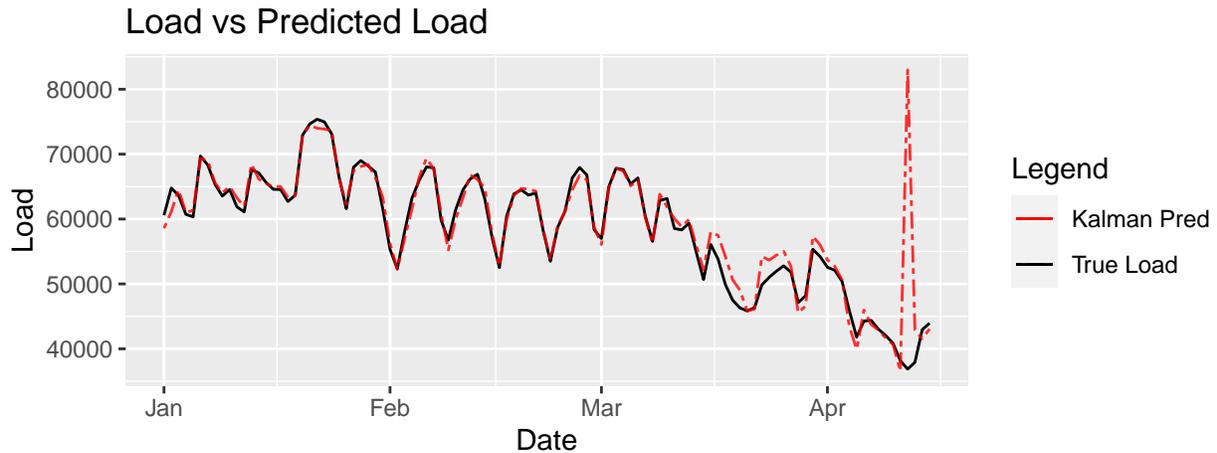
Une fois le modèle qgam amélioré, nous avons décidé d'implémenter l'online-learning. Cette méthode nous permet d'aborder le problème des données d'entraînement insuffisantes pour la période Covid. En effet, le filtre Kalman met à jour l'estimation des poids séquentiellement, au fur et mesure des prédictions, ce qui résout partiellement le problème de données Covid insuffisantes. Pour simplifier la situation Covid, nous avons créé la variable *GRI_factor*, étant une variable catégorielle à trois facteurs (none, medium, high). Ceci nous permet de faire une mise au point de la situation Covid avec de l'online-learning. Malheureusement, les résultats étaient meilleurs sans cette variable. C'est pour cela que nous avons décidé de continuer avec le modèle qgam précédent pour l'online learning.

```
#####online learning
# static
X <- predict(gam9, newdata=Data_train, type='terms')
y = Data_train$Load
###scaling columns
for (j in 1:ncol(X)){
  X[,j] <- (X[,j]-mean(X[,j])) / sd(X[,j])
}
X <- cbind(X,1)
d <- ncol(X)

#dynamic
ssm <- viking::statespace(X, y)
gam9.kalman.static <- ssm$pred_mean%>%tail(nrow(Data1))
#ssm_dyn <- viking::select_Kalman_variances(ssm, X[sel_a,],
#y[sel_a], q_list = 2^(-30:0), p1 = 1, ncores = 6)
#saveRDS(ssm_dyn, "../Results/Kalman_filter_Data_train.RDS")
ssm_dyn = readRDS("../Results/Kalman_filter_Data_train.RDS")
ssm_dyn <- predict(ssm_dyn, X, y, type='model', compute_smooth = TRUE)
gam9.kalman.Dyn <- ssm_dyn$pred_mean%>%tail(nrow(Data1))
rmse(Data1$Load, gam9.kalman.Dyn)
```

```
## [1] 4700
```

Comme pour le Arima, les résultats ne sont pas optimaux, mais à nouveau, les prédictions sont différentes et intéressantes pour l'agrégation d'experts.



Pipeline basée sur le modèle qgam

Étant arrivés au bout des améliorations de qgam et tout en ayant tenté arima et le filtre kalman, nous avons considéré d'autres modèles vus en cours pour comparer leurs performances. Nous avons ainsi étudié les résidus par blocs et appliqué des forêts aléatoires, ce qui va nous permettre d'améliorer le modèle. Ensuite, nous avons amélioré davantage la performance à l'aide de Arima.

```
refit <- Arima(ts_res_forecast, model=fit.arima.res)
prevARIMA.res <- tail(refit$fitted, nrow(Data1))
gam9.arima.forecast <- gam9.forecast + prevARIMA.res
```

Agrégation d'experts

Comme dernière méthode, nous avons décidé d'implémenter une agrégation d'experts pour extraire une combinaison de prédicteurs qui puissent améliorer davantage la performance du modèle. Pour cela, nous avons regroupé l'ensemble des prédicteurs dans une variable experts. Dans cette agrégation d'experts, nous avons utilisé les différents modèles, notamment: le modèle linéaire polynomial, gam, qgam (avec et sans arima), une forêt aléatoire comprenant toutes les variables, un filtre kalman et la pipeline. Le modèle choisi est MLpol, celui-ci nous permet d'ajuster les coefficients des modèles de façon variable sur chaque partie.

```
experts <- cbind(gam9.forecast, pred2,rf.forecast, gam.forecast,
               gam9.kalman.Dyn, arima.predict,
               as.numeric(gam9.arima.forecast))%>%as.matrix
nom_exp <- c("qgam", "polynomial_lm", "rf",
            "gam", "kalman", "arima", "pipeline")
colnames(experts) <- nom_exp

rmse_exp <- apply(experts, 2, rmse, y=Data1$Load)

cumsum_exp <- apply(Data1$Load-experts, 2, cumsum)

#Correction du biais
expertsM2000 <- experts-2000
```

```

expertsP2000 <- experts+2000
experts <- cbind(experts, expertsM2000, expertsP2000)
colnames(experts) <-c(nom_exp, paste0(nom_exp, "M"), paste0(nom_exp, "P"))
cumsum_exp <- apply(Data1$Load-experts, 2, cumsum)

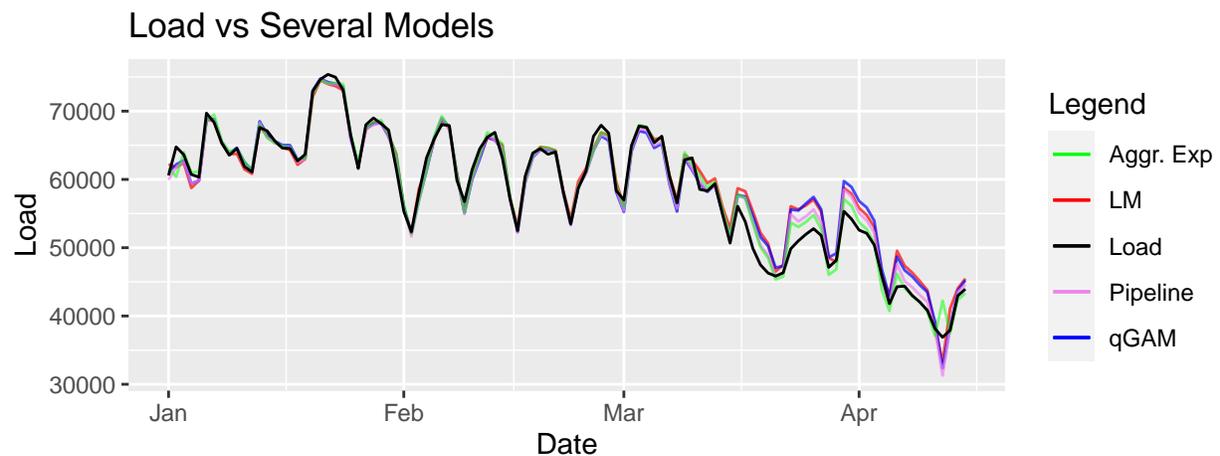
par(mfrow=c(1,1))
agg <- mixture(Y = Data1$Load, experts = experts, loss.gradient=TRUE)
#summary(agg)
####PLOT EXPERT AGGREGATION
#plot(agg)
or <- oracle(Y=Data1$Load, experts)

par(mfrow=c(1,1))
rmse(agg$prediction, y=Data1$Load)

```

```
## [1] 1341
```

Nous pouvons voir dans la figure, que le modèle le plus performant est effectivement issu de l'agrégation d'experts. En effet, nous avons des modèles avec des prédictions très différentes. L'agrégation de ces modèles, avec mise à jour des coefficients dans le temps, se traduit par une amélioration nette de la performance. Comme on peut le voir sur la figure, lors de l'apparition du Covid, nos prédictions sont moins performantes. Avec l'agrégation d'experts nous avons réussi à améliorer nos prédictions, même si dans cet entraînement (sel_a), l'accès aux périodes Covid étaient extrêmement restreintes.



Conclusion

Dans ce travail, nous avons testé plusieurs modèles différents pour prédire la consommation en électricité de la France. Nous avons vu que nos modèles plus simples donnaient déjà des résultats exploitables. Leur complexification donnait des résultats encore plus prometteurs. Le plus grand défi a été le fait que la période de test a été fixée en plus grande partie pendant la pandémie du Covid. Un autre défi, était le fait qu'au niveau des jeux de données train issu de sel_a, on n'avait pas encore du Covid. Ceci créait un biais entre les scores prévus et ceux sur Kaggle. En effet, en entraînant le jeu de données train tout entier, on avait un mois de Covid, ce qui améliorait nettement nos résultats. Enfin, notre meilleur prédicteur était issu d'un modèle additif généralisé avec régression quantile. L'utilisation de techniques telles que l'agrégation d'experts et l'apprentissage en ligne ont encore amélioré notre performance. Ceci nous a permis d'améliorer la performance de notre meilleur prédicteur et aboutir enfin au modèle final.