

Modélisation de saisonnalités complexes avec GAMs

Leonardo Martins Bianco - Guillaume Lambert

05/03/2021

Introduction

Nous avons choisi d'étudier des données météorologiques de la **National Oceanic and Atmospheric Administration (NOAA)**, une agence états-unienne responsable de la préservation, la surveillance, l'évaluation et l'accès public aux données et informations météorologiques. Les données que nous avons utilisées font parties des **Local Climatological Data (LCD)**, regroupant des données par heure, jour et mois sur un ensemble d'environ 1600 stations météorologiques aux Etats-Unis.

Même si les données semblent simples, la haute fréquence des observations implique un caractère de saisonnalité complexe. C'est-à-dire que les observations horaires donnent une saisonnalité journalière ("locale"), mais aussi une saisonnalité annuelle ("globale").

Les méthodes classiques, en particulier celles implémentées pour l'objet `ts`, ne sont pas capables de traiter ces complexités saisonnales. On présente ici deux alternatives. Plus récemment (2012) (Hyndman, n.d.b) une nouvelle classe `msts` ("Multiple seasonality time series") a été implémentée pour traiter ces difficultés. On va l'utiliser dans ce projet pour le comparer à un modèle GAM. On fera une étude de la complexité du modèle GAM et on expliquera comment choisir la base selon une étude graphique et une étude des performances. On verra que les résultats sont considérablement meilleurs que ceux obtenus par `msts` et en moins de temps. On terminera notre travail par l'étude des résidus de notre modèle GAM pour déterminer son efficacité.

Présentation et mise en forme des données

Nous avons choisi la station météorologique **JFK INTERNATIONAL AIRPORT, NY US**. Elle est située à 3.4 mètres en hauteur et ses coordonnées latitude/longitude sont 40.63915° , -73.76401° .

La période de mesure est de 01-07-1948 à 22-02-2021. On télécharge les données sur la période 01-01-2010 à 31-12-2019, ce qui représente un nombre conséquent d'observations permettant une bonne étude des données. On importe les données :

```
data_brute <- read_delim("meteo.csv", delim=",")
```

Ce jeu de données est très dense en informations : il y a 124 variables. La première chose à faire est de comprendre les données et les mettre en forme.

La première variable **STATION** est le numéro de la station météorologique que l'on étudie : 74486094789. Etant donné que l'on en étudie des données issues d'une seule station, on peut se séparer de cette variable. La deuxième variable **DATE** étant primordiale dans l'étude de séries chronologiques, on la garde. On garde également la variable **SOURCE** qui va nous permettre de nettoyer les données. Finalement, on extrait également les variables par heure et par jour.

On remarque que lorsque **SOURCE** = 6, la ligne correspondante est consacrée aux valeurs journalières et donc les valeurs horaires sont par défaut. On doit donc prendre les lignes telles que **SOURCE** \neq 6.

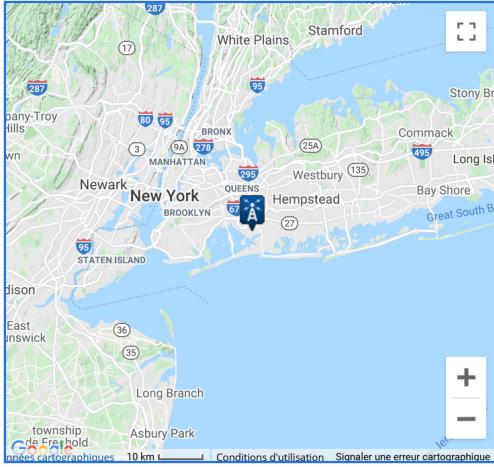


Figure 1: Emplacement de la station météorologique

```
data_brute <- data_brute[which(data_brute$SOURCE != 6),]
```

D'autre part, on remarque que la première date de notre jeu de données est *2010-01-01 00:51:00* et que parmi les lignes où **SOURCE** = 7, il y a une incrémentation d'une heure à partir de *2012-01-01 00:51:00*. C'est donc ces lignes qui nous intéressent car on souhaite étudier la température au fil des heures. Les lignes où **SOURCE** = 4 correspondent à des dates avec heures "rondes" (par exemple *2012-01-01 01:00:00*), elles ne nous intéressent donc pas. On retire donc les lignes où **SOURCE** = 4, puis on retire les lignes où **SOURCE** = 7 avec des dates non incrémentées de la date de départ *2010-01-01 00:51:00*.

```
data_brute <- data_brute[which(data_brute$SOURCE != 4),]
data_brute <- data_brute[which(minute(data_brute$DATE)==51),]
```

Concernant la variable **HourlyDryBulbTemperature**, la documentation nous donne l'indication suivante : "This is the dry-bulb temperature and is commonly used as the standard air temperature reported". Cette colonne correspond donc bien à ce que l'on cherche. On prend donc les colonnes **DATE** et **HourlyDryBulbTemperature**.

Vérifions le type des deux variables :

```
# Vérifions le type des deux variables :
str(data$DATE)
```

```
##  POSIXct[1:87098], format: "2010-01-01 00:51:00" "2010-01-01 01:51:00" "2010-01-01 02:51:00" ...
```

```
str(data$HourlyDryBulbTemperature)
```

```
##  num [1:87098] 33 33 33 33 33 33 33 33 34 35 ...
```

La variable **DATE** est bien sous format d'une date *POSIXct* et la variable **HourlyDryBulbTemperature** est sous le format numérique *num*. Aucune valeur d'un autre type est observée et le nombre total d'observations est le même pour les 2 variables : 87098.

On convertit ensuite les températures de degrés Fahrenheit en degrés Celsius et on arrondi les valeurs à 4 chiffres décimaux.

Avant de procéder à l'analyse descriptive des données, on vérifie que notre jeu de données est complet.

```
data[!complete.cases(data),]
```

```
##          DATE HourlyDryBulbTemperature
```

```

## 72766 2018-05-04 22:51:00 NA
## 79909 2019-03-01 11:51:00 NA

```

Seules 2 valeurs de la température sont manquantes. On décide de remplacer ces 2 valeurs par la température de l'heure suivante. On assume ce choix car la température varie peu d'une heure à l'autre et cela concerne 2 observations sur 87098.

On renseigne ensuite le fuseau horaire *EST: Eastern Standard Time (North America)* à la variable **DATE** :

```
data$DATE <- with_tz(data$DATE, "EST")
```

Finalement, pour créer nos modèles et faire des prédictions, on sépare les données que l'on a traité en deux : une partie d'entraînement du modèle **trainData** et une partie de prévision **testData**. On ajoute aussi des variables auxiliaires temporelles : **Heure** l'heure de la journée, **NumJour** le numéro du jour dans l'année et **NumSem** le numéro de la semaine dans l'année.

Analyse descriptive des données

Après avoir mis en forme les données, la prochaine étape de notre étude est l'analyse descriptive de nos données. Tout d'abord, calculons les statistiques de base :

```
summary(data[-c(3, 4, 5)])
```

```

##           DATE                  HourlyDryBulbTemperature
##   Min.   :2009-12-31 19:51:00   Min.   :-17.222
##   1st Qu.:2012-06-28 21:06:00   1st Qu.:  5.556
##   Median :2014-12-25 16:21:00   Median : 13.333
##   Mean    :2014-12-27 02:41:00   Mean    : 13.009
##   3rd Qu.:2017-06-23 13:36:00   3rd Qu.: 21.111
##   Max.    :2019-12-31 18:51:00   Max.    : 38.889

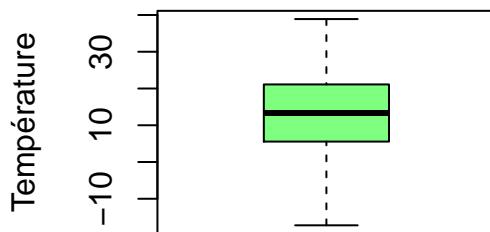
```

La variable **DATE** varie de *2010-01-01 00:51:00* à *2019-12-31 23:51:00* avec une incrémentation d'une heure à chaque observations.

La variable **HourlyDryBulbTemperature** prend ses valeurs dans l'intervalle $[-17.222, 38.889]$. La moyenne et la médiane sont très proches : 13.009 et 13.333 respectivement. Ainsi, les valeurs extrêmes ne semblent pas déformer la moyenne à première vue.

Représentons la boîte à moustaches de la température :

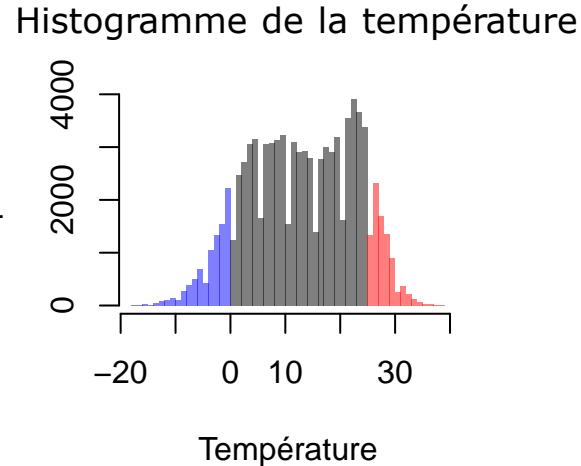
Boîte à moustaches de la température



On retrouve visuellement le fait que la moyenne et la médiane sont proches. On remarque un phénomène intéressant concernant les quartiles : le premier quartile (des valeurs les plus basses) est plus grand que le dernier quartile (des valeurs les plus hautes). On en déduit que les 25% des températures les plus basses balayent plus de valeurs que les 25% des températures les plus hautes. D'autre part, on remarque que les premier et quatrième quartiles sont plus grands que les deuxième et troisième. Cela veut dire que la moitié des

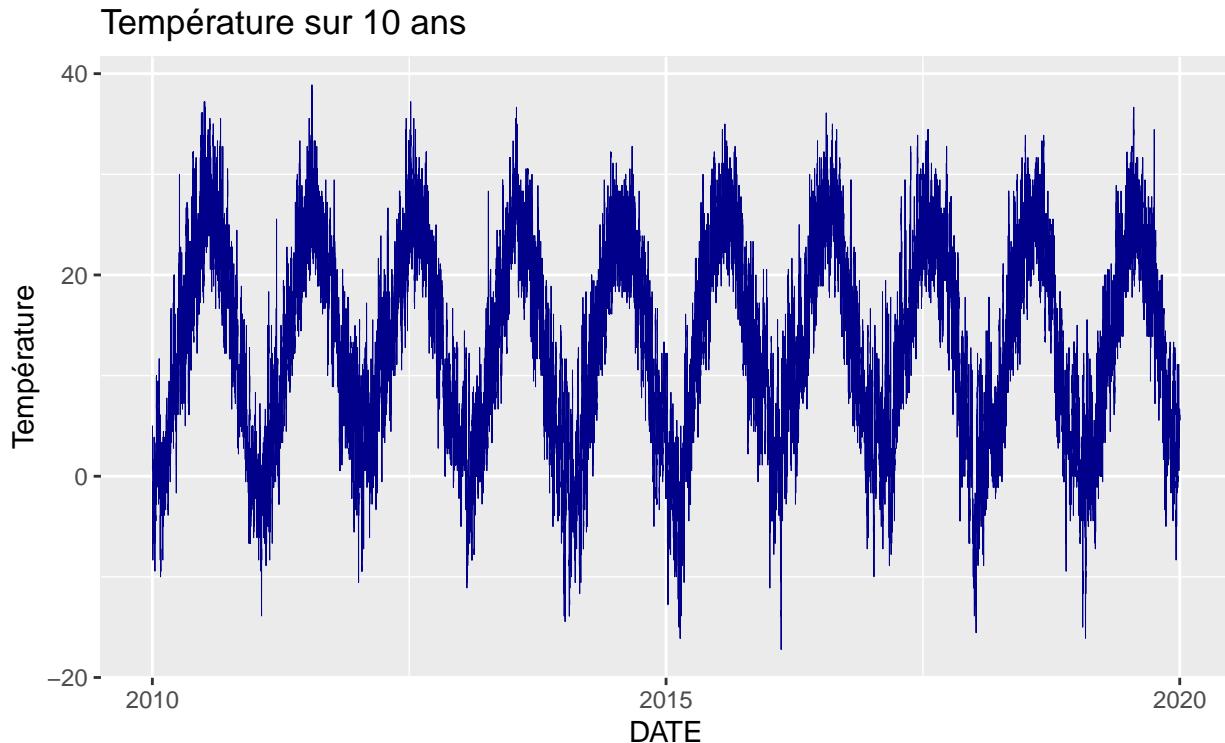
températures intermédiaires sont concentrées autour de la médiane, par rapport à la moitié des températures “extrêmes”.

On peut aussi visualiser cette répartition sur l'histogramme suivant, où on représente en bleu les températures inférieures à 0 degrés et en rouge les température supérieures à 25 degrés :



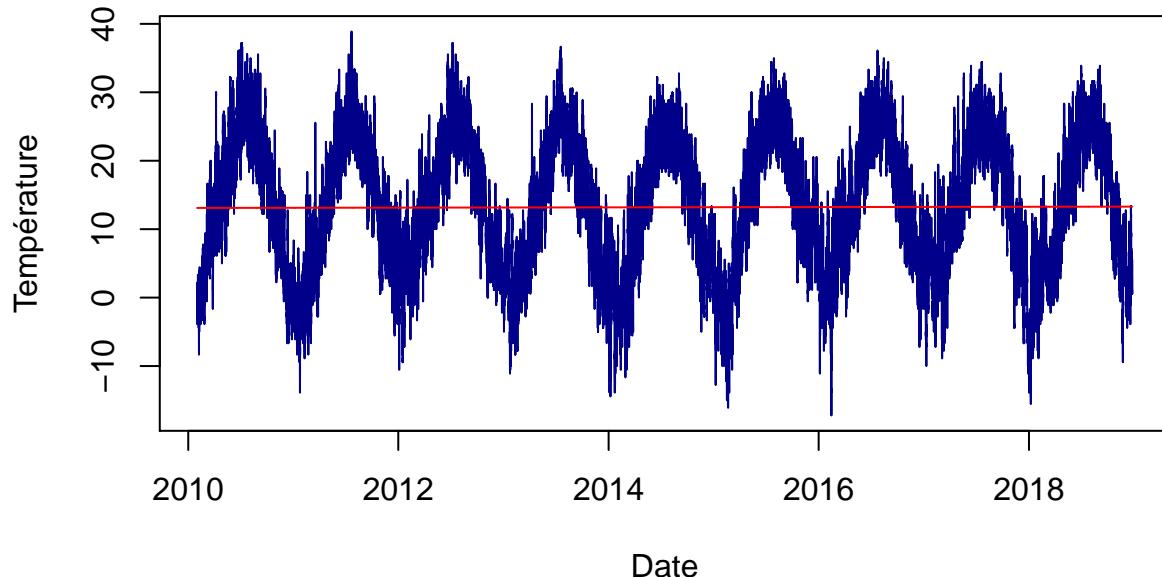
On remarque que la répartition des températures a une allure gaussienne.

On va maintenant étudier la tendance et la saisonnalité apparentes de la température par rapport au temps. Représentons la variable **HourlyDryBulbTemperature** par rapport à la variable **DATE** :



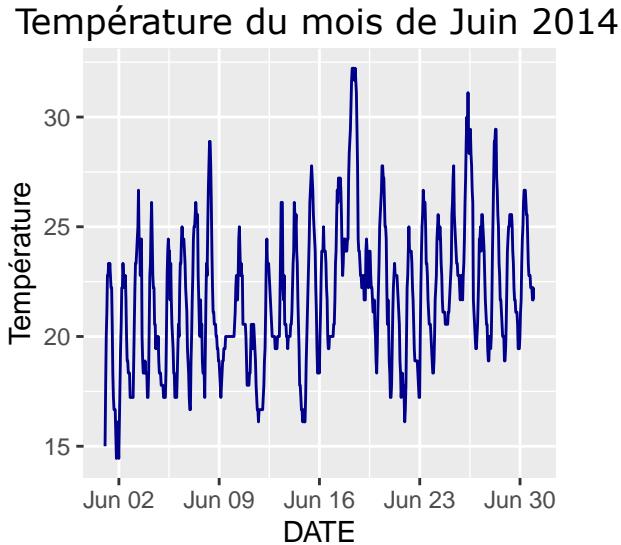
On ne distingue pas avec assurance une tendance. On se dit intuitivement qu'elle sera affine avec une pente très faible. On choisit donc de modéliser la tendance par une régression avec un intercept (pour capturer l'ordonnée à l'origine) et une variable **t** numérotant les observations (pour capturer la pente) :

Estimation de la tendance par régression linéaire



Le résultat est cohérent avec nos attentes. La valeur du coefficient associé à la pente est vraiment très faible (de l'ordre de 10^{-6}) mais la variable **t** est significative dans le test de significtivité de Student. Le coefficient associé à l'intercept est 13,1, ce qui est en accord avec la moyenne de la température que l'on a calculée précédemment (13.009).

Concernant la saisonnalité, on en remarque une annuelle avec un pic positif au milieu de l'année (l'été) et deux pics négatifs au début et à la fin de l'année (l'hiver). L'échelle des données étant grande, on ne distingue pas de possibles saisonnalités plus courtes. On représente donc la variable **HourlyDryBulbTemperature** sur le mois de juin 2014 :



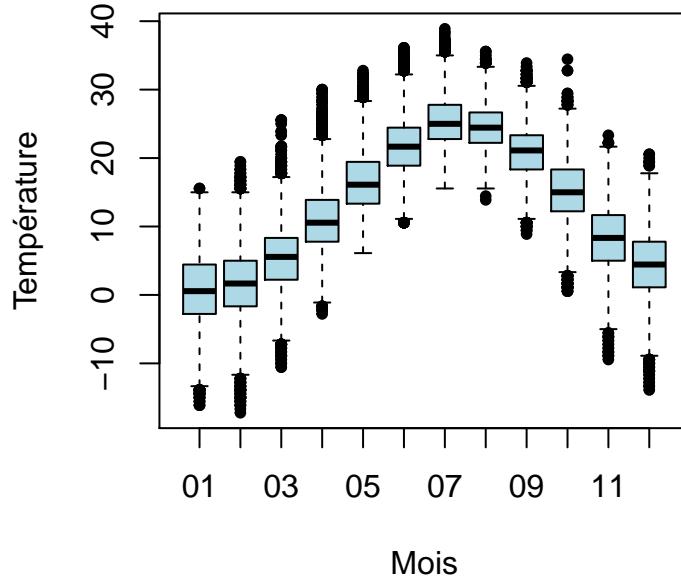
On remarque ici une nouvelle saisonnalité : une saisonnalité journalière avec un pic positif à la moitié de la journée (le midi) et des pics négatifs au début et à la fin de la journée (le matin et le soir).

La variable **HourlyDryBulbTemperature** semble donc posséder 2 saisonnalités : une annuelle et une journalière. On peut de plus les interpréter par les saisons et le cycle jour/nuit.

On veut analyser ces deux saisonnalités en faisant 2 croisements par facteur : le premier par mois pour la

saisonnalité annuelle et le second par heure pour la saisonnalité journalière.

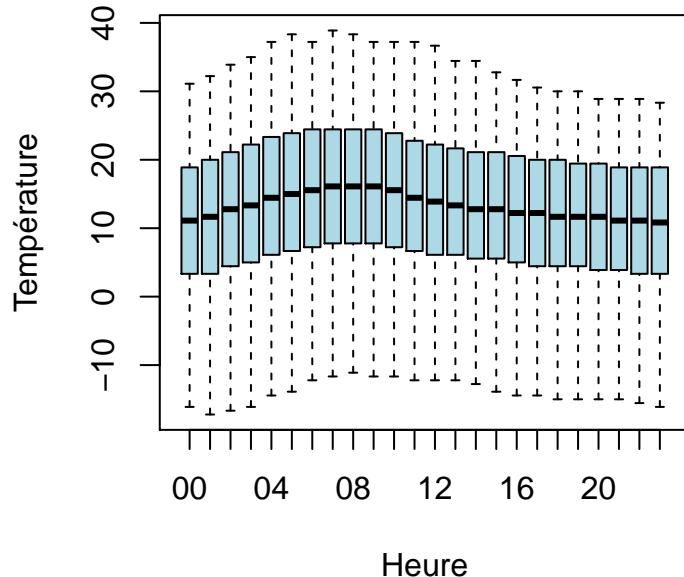
Température annuelle par des boîtes à moustaches mensuelles



On observe de nouveau la saisonnalité de la température selon les mois et les saisons de l'année. Ce que l'on peut noter de nouveau est que la plage des valeurs prises en hiver est bien plus étendue que celle des valeurs prises l'été. On observe également la présence de valeurs aberrantes que l'on peut aussi deviner sur le graphique de la température sur 10 ans.

Représentons cette fois-ci la température selon les heures :

Température journalière par des boîtes à moustaches horaires

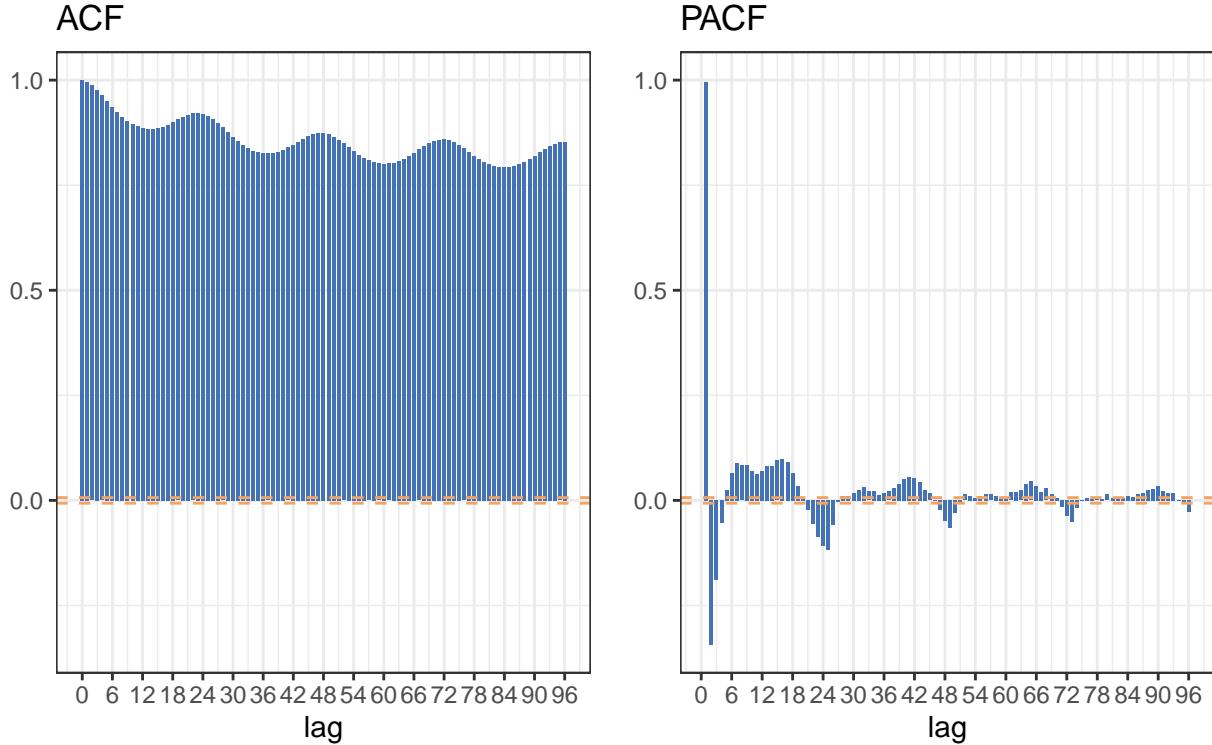


On observe bien la saisonnalité journalière avec des valeurs hautes de la température l'après-midi, lorsque le Soleil est levé. Ce graphique ne nous donne pas beaucoup plus d'informations et on s'attendait à une plus grande variance de la température le long des journées. Cependant, la moyenne selon les heures étant calculée sur tous les mois, on mélange les hautes températures de l'été avec les basses températures de l'hiver, d'où la

faible variance.

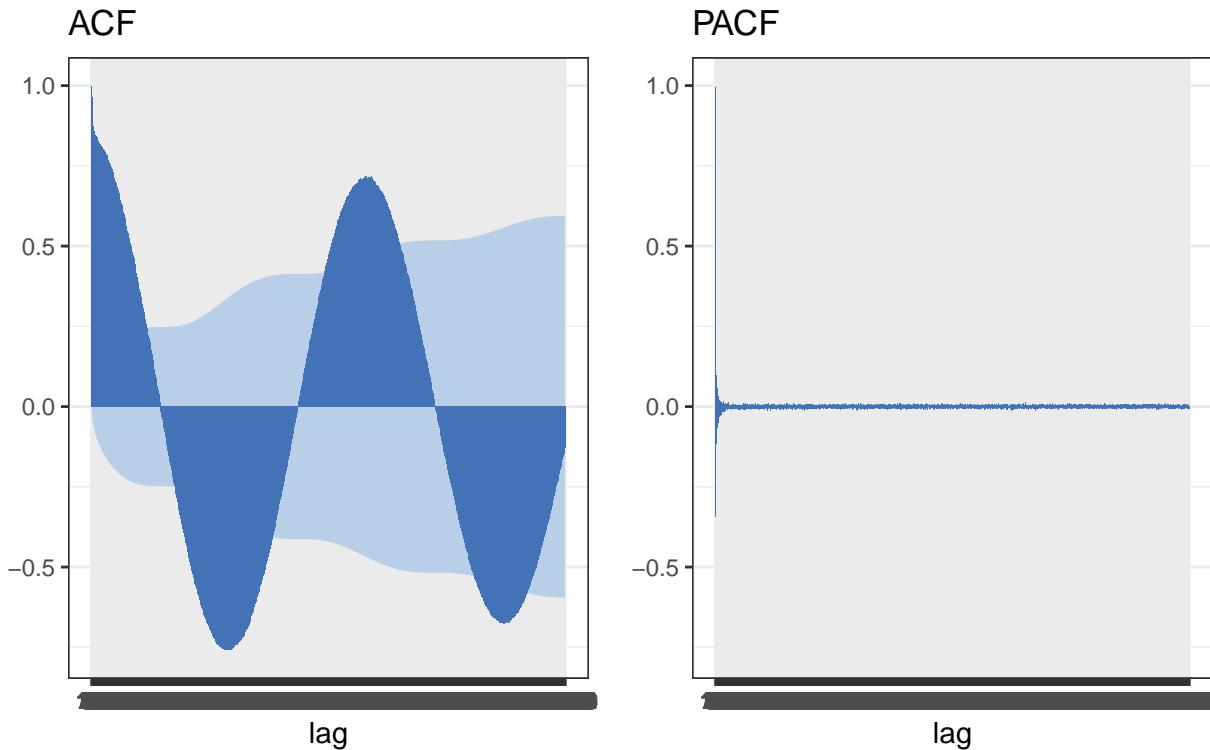
La dernière étape de notre analyse descriptive est l'étude de l'autocorrélogramme où l'on doit distinguer une auto-corrélation toutes les 24 valeurs pour la saisonnalité journalière et une auto-corrélation toutes les $24 \times 365.25 = 8766$ valeurs pour la saisonnalité annuelle. On représente aussi, à côté de chaque graphe d'autocorrélation, un graphe de l'autocorrélation partielle. Voici l'interprétation de cette quantité : on peut projeter une série chronologique sur ses valeurs passées selon la formule $P(Y_t) = \sum_{j=1}^k b_{k,j} Y_{t-j}$. L'autocorrélation partielle évaluée en k correspond au coefficient $b_{k,k}$. Une autre interprétation de l'autocorrélation partielle en k est que l'on analyse la corrélation entre X_1 et X_k en retirant les interférences des termes intermédiaires X_2, \dots, X_{k-1} . C'est donc une espèce de "débruitage", et on voit bien que son graphe est moins chargé que le graphe acf.

Pour observer l'auto-corrélation journalière, on représente l'autocorrélogramme sur 96 valeurs (4 jours) :



On observe une très forte auto-corrélation sur ces 4 jours avec des valeurs d'auto-corrélation proches de 1. On voit aussi la saisonnalité journalière comme prévu. Etant donné que cette saisonnalité est incluse dans la saisonnalité annuelle, l'auto-corrélation journalière suit une tendance, ici à la baisse.

Pour observer l'auto-corrélation annuelle, on représente l'autocorrélogramme sur $8766 \times 2 = 17532$ valeurs (2 ans) :



On observe la saisonnalité annuelle comme prévu toutes les 8766 valeurs. Les valeurs des hivers sont auto-correlées positivement et les valeurs des étés sont auto-correlées négativement, ce qui correspond bien à l'allure de la saisonnalité annuelle.

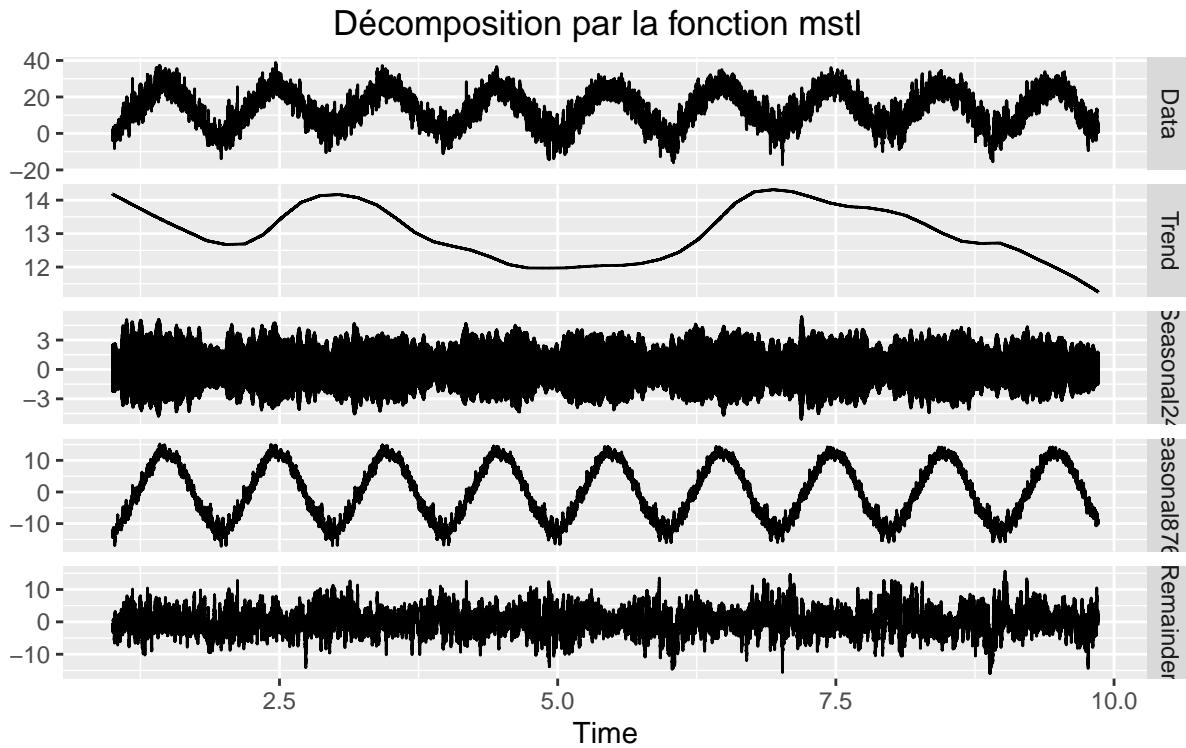
Enfin, concernant la stationnarité de la température, les 2 saisonnalités impliquent que l'espérance de la température dépend du temps et donc la température n'est pas stationnaire. On peut de plus l'observer sur les 2 autocorrélogrammes où l'auto-corrélation est très forte avec des maximums globaux pour les lags multiples de 365.25×24 (saisonnalité annuelle) et des maximums locaux pour les lags multiples de 24 (saisonnalité journalière).

Modèles MSTS

La première approche est d'utiliser la classe `msts` du package `forecast` pour le logiciel R. C'est une classe développée en 2011 pour traiter les séries chronologiques ayant des saisonnalités multiples. On va donc utiliser ce modèle "tout prêt" pour le comparer à un modèle GAM.

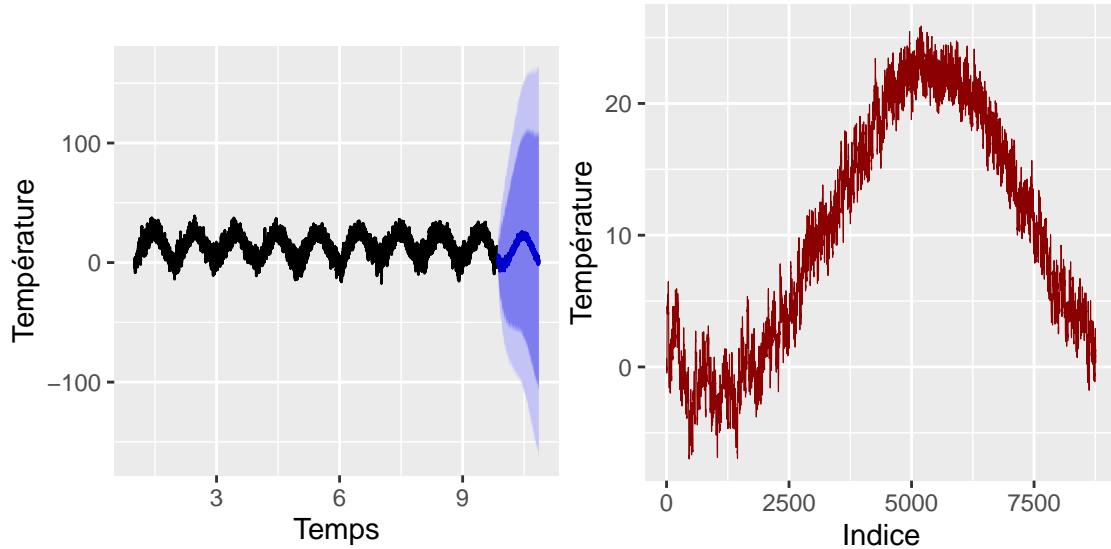
```
trainDataMSTS <- msts(trainData$HourlyDryBulbTemperature, seasonal.periods = c(24, 24*365))
```

Voici une décomposition des données faite par la fonction `mst1`.



On voit que les saisonnalités journalières et annuelles sont bien capturées, et que les résidus sont centrés et à peu près de variance constante, ce qui est une bonne indication visuelle de stationnarité.

On réalise ensuite une prévision en utilisant ce modèle. Observons les graphes de la prévision en comparaison aux données d'entraînement et le graphe de la partie prédictive. On voit que les 2 saisonnalités sont présentes dans la prévision.



On peut calculer l'erreur RMSE entre la prévision et les données de test. Le résultat est $\text{RMSE} = 5.379212$. On comparera ce résultat avec celui du modèle GAM que l'on construit dès maintenant.

Modèle GAM

Les modèles additifs généralisés (GAM) ont été introduits dans (Hastie and Tibshirani 1990) et son but est d'ajouter à une régression linéaire des termes non-linéaires. L'objectif est donc de modéliser des relations non-linéaires.

La théorie

Mathématiquement on écrit un GAM comme

$$y_i = \beta_0 + X_i\beta + f_1(x_{1,i}) + f_2(x_{2,i}) + f_3(x_{3,i}, x_{4,i}) + \dots + \varepsilon_i$$

On voit bien que la partie linéaire est toujours présente avec le terme $X_i\beta$ où X_i est le plan formé des variables $(x_{k,i})_k$. Les fonctions f_j sont supposées lisses, dépendantes d'au plus 2 variables et elles sont estimées selon un choix de base. Les résidus ε_i doivent être i.i.d., centrés, et de variance constante (mais pas nécessairement gaussiens).

On garde beaucoup de similarités avec l'approche de la régression linéaire : on va estimer les bons coefficients β et les fonctions f_j en minimisant un problème de moindres-carrés. On pourrait proposer

$$\min_{\beta, f_j} \|y - \beta_0 - X\beta - f_1(x_1) - f_2(x_2) + \dots\|^2$$

Cependant, comme dans le cas linéaire, il convient de “régulariser” ce problème. On veut éviter la surapprentissage et avoir des fonctions qui n'oscillent pas trop fortement, i.e., limiter le niveau de non-linéarité de notre modèle. Cette seconde considération nous suggère de faire une régularisation de type Ridge sur les dérivées secondes des fonctions f_j :

$$\min_{\beta, f_j} \|y - \beta_0 - X\beta - f_1(x_1) - f_2(x_2) + \dots\|^2 + \lambda_1 \int f_1''(x)^2 dx + \lambda_2 \int f_2''(x)^2 dx + \dots$$

On va expliquer plus tard comment choisir les paramètres λ_i de régularisation.

On observe déjà que les fonctions f_j seront estimées selon un choix de base de fonctions non-linéaires. Il y a plusieurs choix possibles et cela change en fonction des données. Notamment, on utilisera des splines cubiques de manière générale, et des splines cycliques si nécessaire pour modéliser des parties saisonnales. Quand on choisit des splines, c'est nécessaire aussi de spécifier le nombre de noeuds à utiliser.

Finalement, voici une liste des inconnues du problème : on doit choisir la base de fonctions non-linéaires, le choix du nombre k de noeuds, on doit estimer les coefficients des f_j dans cette base, les coefficients β de la partie linéaire du modèle, et finalement les paramètres de régularisation λ_i .

Validation croisée

On décrit maintenant la procédure de validation croisée (“cross-validation” en anglais). Cette procédure est très utile pour l'estimation des paramètres du modèle, comme les paramètres de pénalisation λ_i , et le nombre de noeuds k dans la base de splines. Il y a plein d'autres applications, comme la sélection de variables, la prévision du score dans une compétition comme Kaggle et le choix de modèles.

Validation croisée ordinaire Elle consiste à partitionner le jeu de données d'entraînement en k parties, puis entraîner un modèle sur $k - 1$ des parties et faire une prédiction dans la partie restante. Cela estime l'erreur de prédiction du modèle. Cependant on observe que il y a une combinatoire non-triviale dans cette procédure, et donc cette approche est en générale lente.

Validation croisée généralisée On décrit maintenant une idée originale introduite dans (Golub, Heath, and Wahba 1979). Dans cet article, ils déduisent un estimateur du paramètre de régularisation λ (dans le cas d'une régularisation Ridge spécifiquement).

Cette quantité peut être aussi vue comme un estimateur du score de la validation croisée ordinaire. Ainsi, on s'approche d'un tel score sans avoir directement analysé chaque sous-modèle : on minimise numériquement une fonction, ce qui est beaucoup moins cher numériquement.

L'estimateur qu'ils trouvent est

$$V_g(\lambda) = n\|y - X\hat{\beta}_\lambda\|^2 / (n - \text{tr}(F_\lambda))^2$$

où $\hat{\beta}_\lambda = F_\lambda \hat{\beta}_0$, avec

$$\hat{\beta}_0 = (X^T X)^{-1} X^T y \quad \text{et} \quad F_\lambda = \left(X^T X + \sum \lambda_j S_j \right)^{-1} X^T X$$

On remarque que $\hat{\beta}_0$ est l'estimation de β dans le cas non-régularisé.

Implémentation

Approche naïve

On pourrait écrire un modèle GAM en ajoutant comme variables l'heure de la journée **Heure**, le numéro de la semaine dans l'année **NumSem** et le numéro du jour dans l'année **NumJour**. Les relations entre chacune de ces variables et la température étant non-linéaires, on utilise des fonctions *smooth s()*. Les splines choisis sont des splines cubiques cycliques. On les choisit cycliques car cela entraîne la continuité entre le début et la fin du *fitting*, ce qui est primordial lorsqu'on étudie des phénomènes cycliques comme la température.

```
# Modèle avec toutes les variables
modelGAM <- gam(HourlyDryBulbTemperature ~
  s(Heure, bs = 'cc', k = 24) +
  s(NumSem, bs = 'cc', k = 52) +
  s(NumJour, bs = 'cc', k = 365),
  data = as.data.frame(trainData))
```

On voit que si l'on essaye d'utiliser la variable **NumJour**, qui indique à quel jour de l'année on est, on a des problèmes dans le choix du nombre de noeuds k . Soit on prend k petit, mais ainsi on perd de l'information sur la modélisation et la prédiction, soit on prend k grand (idéalement $k_{\max} = 365$) et le temps d'exécution du code est très long. On aimerait alors utiliser une autre variable en limitant les pertes sur les performances du modèle.

Sélection de variables

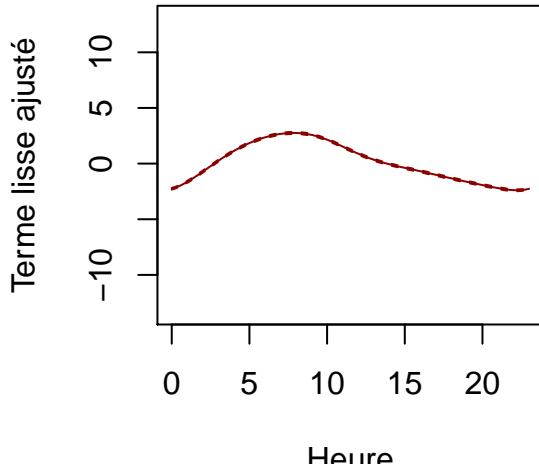
Une idée est d'observer que dans une semaine donnée, l'évolution de la température sur une journée ne diffère pas des autres journées de la semaine. C'est-à-dire que dans une semaine fixée, la saisonnalité journalière possède une tendance constante égale à la moyenne de la semaine en question. Cela indique que les jours de cette semaine sont équivalents, dans le sens que pour estimer la température à 15h, peu importe si c'est lundi, jeudi ou dimanche : si on connaît la moyenne de la semaine, il faut juste connaître la moyenne de température à 15h. On assume donc le choix de remplacer le numéro du jour dans l'année par le numéro de la semaine dans l'année. On vient de donner une explication visuelle et intuitive, et on donnera une justification basée sur les performances dans la suite.

Ceci dit, on a ≈ 52 semaines dans une année, et 24 heures par jour. Ainsi, le nombre total de noeuds que l'on aura besoin si l'on utilise que ces deux variables est la somme $52 + 24 = 76$, beaucoup moins que les 365 précédents (rien que pour le numéro du jour), et associés à deux variables différentes. On crée donc notre modèle GAM que l'on appelle **modelGAM** et on étudie son résumé :

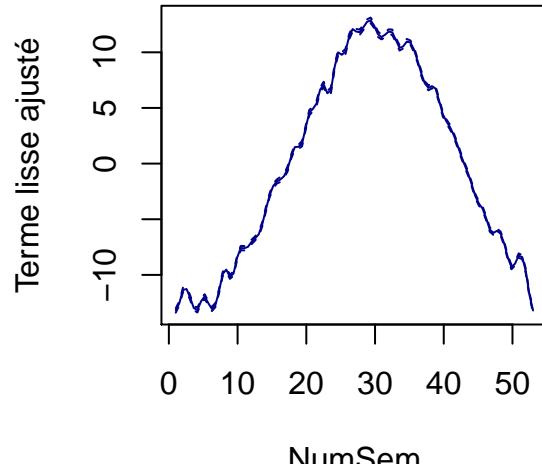
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## HourlyDryBulbTemperature ~ s(Heure, bs = "cc", k = 24) + s(NumSem,
##     bs = "cc", k = 52)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.19709   0.01421   928.8 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df    F p-value
## s(Heure) 10.26    22 690.2 <2e-16 ***
## s(NumSem) 49.60    50 7277.8 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.83  Deviance explained =  83%
## GCV = 15.678  Scale est. = 15.666  n = 77595
```

Soulignons quelques statistiques clés : on a $R_{\text{ajusté}} = 0.83$, et le score de validation croisée généralisée (qui estime la performance de prévision) est $GCV = 15.678$. On note que cette valeur est essentiellement la même que celle obtenue dans le modèle GAM avec plus de variables (et qui prend beaucoup plus de temps pour s'exécuter). Les tests de significativité de Student sont tous très significatifs, ce qui est un bon point pour notre modèle. On trace ensuite les fonctions *smooth* de notre base afin de les étudier :

Terme lisse s(Heure)



Terme lisse s(NumSem)

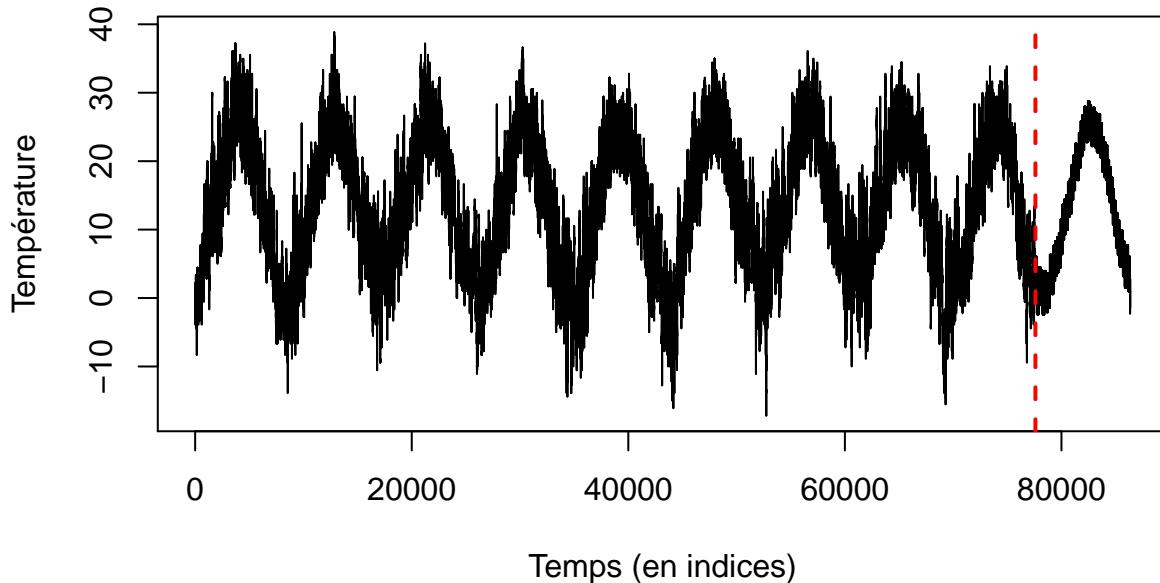


Ces graphes représentent les fonctions lisses ajustées $f(\text{Heure})$ et $f(\text{NumSem})$ selon la base de spline cycliques choisie dans le modèle GAM. Le premier graphe nous montre que la température semble se concentrer à la

fin du matin chaque jour, et que la température est plus haute à la moitié de l'année (ce qui correspond à l'été). On regarde bien le caractère cyclique par le fait que dans les deux graphes la température à la fin est la même que la température au début. Finalement, on voit aussi que les intervalles de confiance sont très ajustés aux courbes, ce qui signifie que ce résultat est précis.

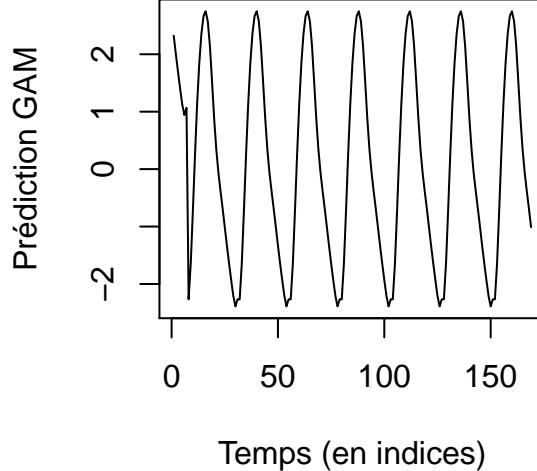
On crée des variables auxiliaires **Temps** pour la prédiction, qui sont juste une numérotation des observations des données d'entraînement **trainData** et des données de test **testData**. C'est une manière pratique de manipuler le temps sans utiliser les variables du type **POSIXt**. Après avoir ajusté le modèle, on fait la prédiction et on trace les données d'entraînement suivies de la prédiction.

Prédiction d'un an avec le modèle GAM

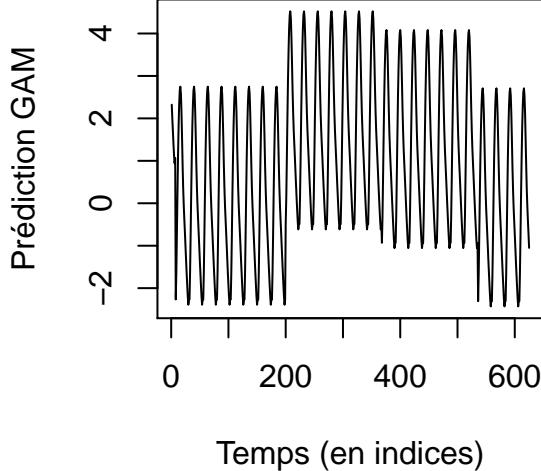


Faisons un graphe avec une échelle plus petite pour confirmer l'existence de la saisonnalité journalière dans ce modèle.

Prédiction d'une semaine



Prédiction d'un mois



Le plot d'une semaine révèle bien que l'on a une saisonnalité journalière. Le plot d'un mois montre bien que, à chaque semaine, on ne différencie pas les jours les uns des autres, mais d'une semaine à l'autre on change la

température moyenne. La structure hebdomadaire de la modélisation où la température moyenne ne change pas n'a aucun sens concret de manière locale mais elle est significative de manière globale. Cela est dû à l'utilisation de la variable **NumSem** au lieu de la variable **NumJour**.

Comparaison avec msts

On calcule le score RMSE par rapport aux données de test et on obtient 3.80328. Or, on voit en comparant le RMSE des modèles GAM et MSTS que le modèle GAM performe considérablement mieux : 3.80328 (GAM) contre 5.37921 (MSTS). Sur le temps d'exécution, on a 7.21889s (GAM) et 4.27370s (MSTS). Par rapport au nombre des données considérées (presque 100.000 observations) on considère que même si le modèle GAM prend quelques secondes de plus, étant donné sa performance, il est compétitif avec le modèle MSTS.

Amélioration du modèle : intéraction entre variables ?

L'une des manières de compléter un modèle **GAM** est l'ajout de fonctions représentant les interactions entre les variables. Plus précisément, on cherche à savoir s'il existe des interactions non-linéaires entre les variables sur l'estimateur du modèle **GAM**. Si c'est le cas, on peut ajouter dans le modèle une fonction *smooth* $s(x_1, x_2)$. On obtient alors l'équation suivante dans un GAM à 2 variables :

$$y = \beta_1 + \beta_2 x_1 + \beta_3 x_2 + \beta_4 f_1(x_1) + \beta_5 f_2(x_2) + \beta_6 f_3(x_1, x_2)$$

On peut d'ailleurs utiliser un produit tensoriel pour représenter les interactions se produisant à différentes échelles. On aura alors $f_3(x_1, x_2) = f_{3,1}(x_1)f_{3,2}(x_2)$.

Un point important à noter dans l'ajout de fonctions représentant l'intéraction entre variables est le caractère linéaire de l'intéraction. Si l'intéraction est linéaire, alors on obtient $f_3(x_1, x_2) = f_{3,1}(x_1) + f_{3,2}(x_2)$ et alors l'intérêt de la fonction bivariée en (x_1, x_2) disparaît. En effet, on pourra alors réécrire le modèle **GAM** comme

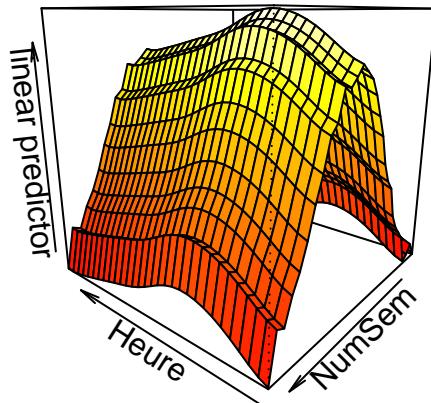
$$y = \beta_1 + \beta_2 x_1 + \beta_3 x_2 + \beta_4 \underbrace{(f_1(x_1) + f_{3,1}(x_1))}_{=g_1(x_1)} + \beta_5 \underbrace{(f_2(x_2) + f_{3,2}(x_2))}_{=g_2(x_2)}$$

Revenons à notre étude. Notre modèle GAM possède pour l'instant un intercept et une fonction *smooth* pour chaque variable. On cherche maintenant à étudier s'il existe une interaction non-linéaire entre les variables **Heure** et **NumSem** sur la température. Intuitivement, on se doute qu'il existe une interaction entre ces deux variables sur la température. En effet, la température sur une journée peut varier différemment selon la saison. Si c'est le cas, il faut que cette différence soit assez importante pour que l'ajout d'un terme d'interaction dans le GAM soit significatif en terme d'amélioration des performances. Par exemple, on peut se demander si la température évolue de la même manière sur une journée en hiver et en été. Pour étudier cette interaction, on va procéder en 2 étapes :

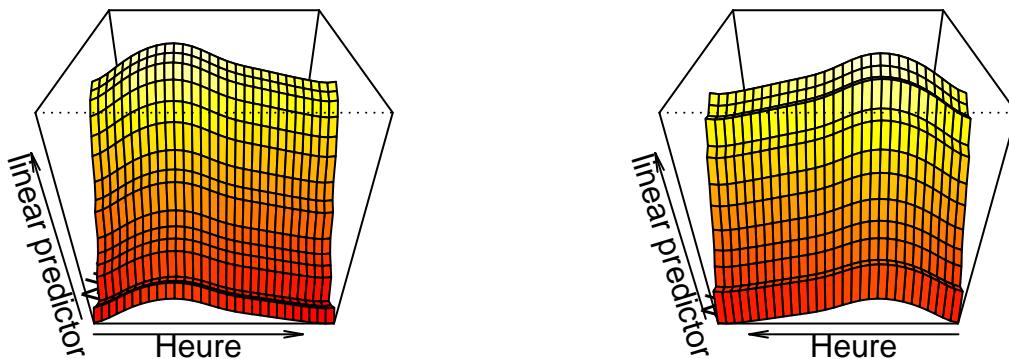
- une étude graphique grâce à la fonction *vis.gam*
- une étude des performances des modèles GAMs avec ou sans terme d'interaction

Etude graphique :

Interaction entre les variables sur l'estimateur

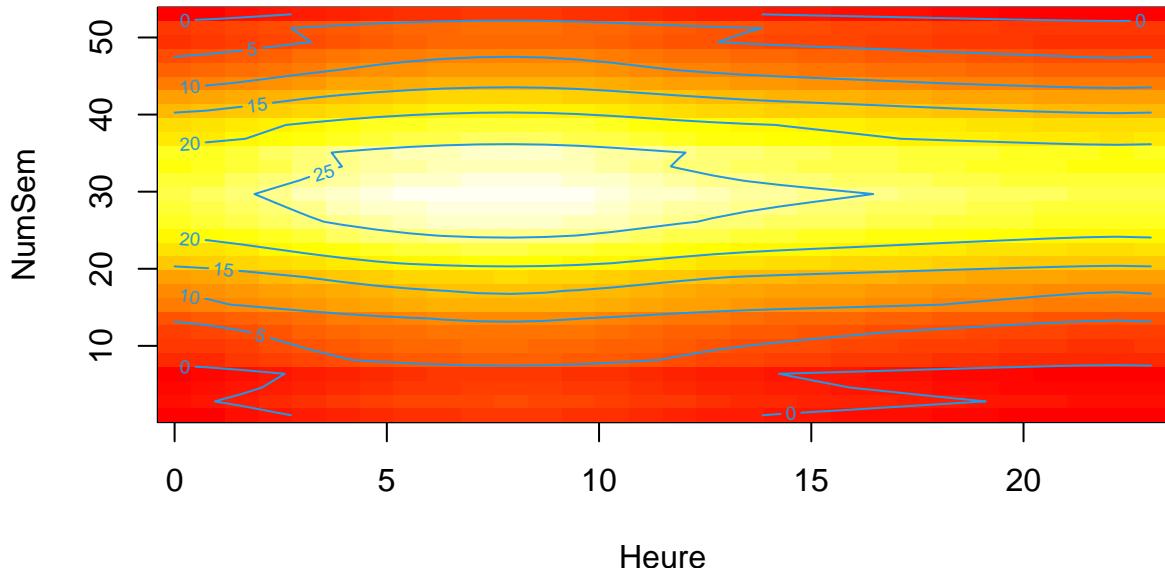


On remarque qu'il existe bien une interaction des variables **Heure** et **NumSem** sur l'estimateur du modèle. On observe un pic positif et une symétrie par rapport au centre des valeurs de **NumSem** qui correspond à l'été. De même, on observe un pic un peu avant le centre des valeurs de **Heure**, ce qui correspond au lever et au coucher du Soleil. On dessine le même graphique selon deux angles différents afin d'avoir un aperçu global :



Comme dit précédemment, on voit bien qu'il y a une interaction dans notre modèle. Il reste maintenant à voir si cette interaction est linéaire dans le sens où on peut la décomposer comme une fonction de **Heure** sommée à une fonction de **NumSem**. On voit sur les 2 graphiques ci-dessus que pour une valeur de **NumSem** fixée et selon les valeurs de **Heure**, les valeurs de l'estimateur semblent évoluées de la même manière. On peut aussi l'observer lorsque l'on trace les lignes de niveaux de l'estimateur :

Lignes de niveau de l'estimateur selon les variables



On voit que la saisonnalité journalière est presque indépendante du numéro de la semaine. On aurait d'ailleurs le même résultat en remplaçant les numéros de la semaine par le numéro des jours. Visuellement, l'interaction de **Heure** et **NumSem** sur l'estimateur semble donc linéaire. Poursuivons notre étude par la comparaison des performances des modèles.

Etude des performances :

Nous avons présenté ici un raisonnement descriptif sur des graphiques mais on peut aussi utiliser un argument basé sur des performances du modèle : le temps de création du GAM (en secondes et arrondi au millième), le R-ajusté et le GCV du GAM, et le RMSE de la prédiction test. On note **Heure** par **H** et **NumSem** par **NS**. Voici les résultats :

Performances			
Modèle	$s(H)+s(NS)$	$s(H)+s(NS)+s(H,NS)$	$s(H)+s(NS)+ti(H,NS)$
Temps de création du GAM (en s)	2.551	6.096	8.333
R-ajusté	0.830	0.832	0.832
Score GCV	15.673	15.487	15.502
Score RMSE	3.790	3.780	3.785

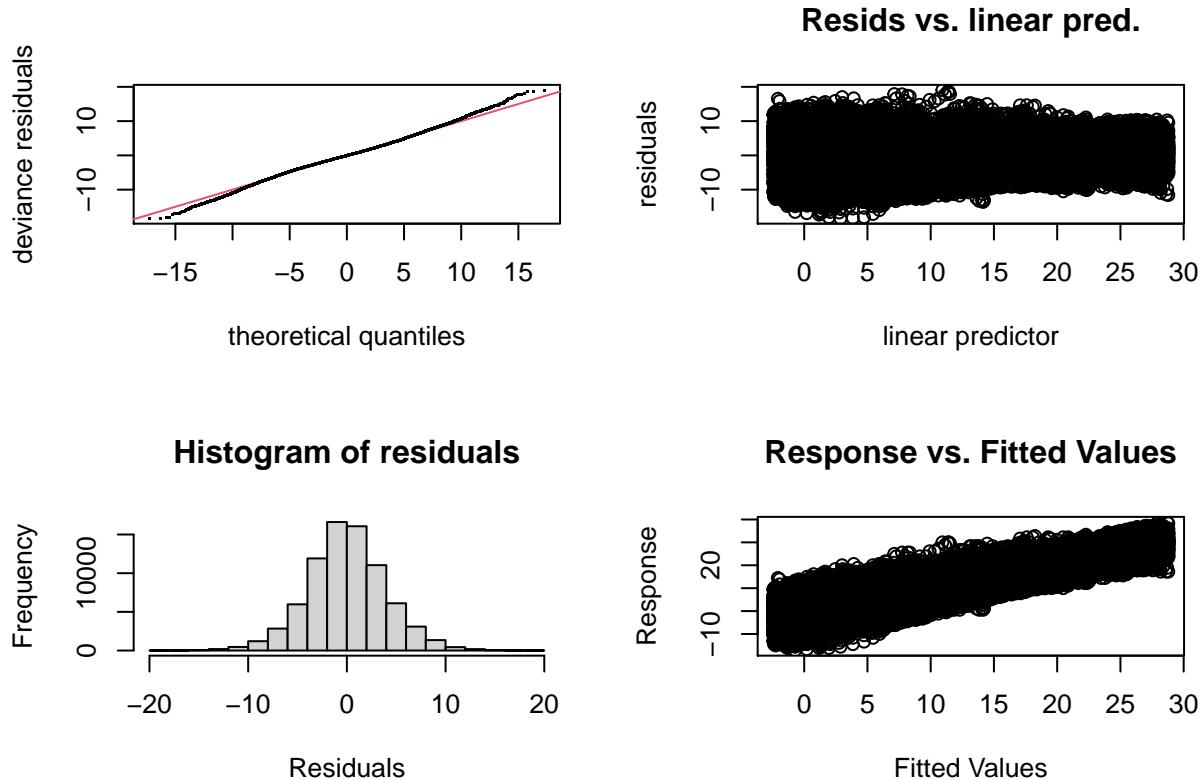
On voit tout d'abord que le temps de création du modèle GAM est doublé voire triplé lorsqu'on ajoute un terme d'interaction. Les modèles avec interaction sont donc plus coûteux à construire et c'est normal car on ajoute une fonction à notre base. Le gain sur le R-ajusté est très faible : 2 millièmes. De même, le gain sur le GCV est de l'ordre du dizième et le gain sur le RMSE est de l'ordre du centième. L'apport d'un terme d'interaction est très peu significatif alors que le temps de calcul est bien plus important. Cela confirme notre étude descriptive sur l'interaction entre les variables que l'on peut considérer comme linéaire. Notre modèle choisi est donc celui sans terme d'interaction.

Remarque : Si dans notre jeu de données on n'observe pas d'interaction non-linéaire, cela ne veut pas dire qu'il n'existe pas de telle interaction dans un autre jeu de données où on mesure la température dans un autre endroit. On ne pourra donc pas admettre ce résultat pour n'importe quel jeu de données de température. On

s'attendait d'ailleurs à observer une interaction non-linéaire dans ce jeu de données car on pensait que la température journalière variait différemment selon la saison et cela de manière assez importante pour avoir une meilleure modélisation. Il serait intéressant d'étudier cela en prenant des données de température dans d'autres endroits.

Vérification du modèle et étude des résidus

Maintenant que nous avons défini notre modèle, on peut étudier les résidus du modèle pour déterminer sa qualité. La fonction *gam.check* nous fournit un test sur la dimension des splines et des graphiques intéressants sur les résidus :



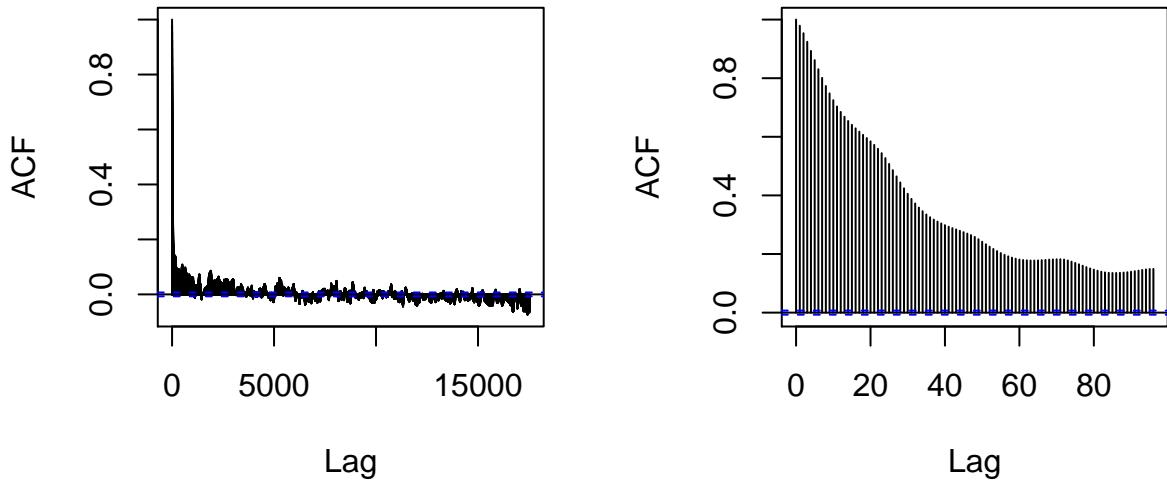
Le test sur les valeurs des paramètres **k** nous indique que ceux que l'on a renseigné dans les splines de notre modèle ne sont pas trop petits.

L'allure de l'histogramme est très satisfaisante car il semble représenter que les résidus suivent une loi normale centrée. Le QQ-plot renforce cette intuition mais les queues sont un peu trop épaisses de part et d'autre. C'est un phénomène de *Kurtosis*. Il y a donc une trop grande variance des deux côtés de la loi normale.

Le graphique “Resids vs. linear pred.” montre que l’homoscédasticité des résidus n’est pas parfaitement respectée avec une plus grande variance pour le premier quart des valeurs de *linear predictor*. L’allure générale est cependant satisfaisante pour le reste des valeurs *linear predictor*. De plus, la variance est centrée sur l’ensemble des valeurs.

On peut faire la même analyse du graphique “Response vs. Fitted Values”. En effet, la relation entre les deux grandeurs est clairement linéaire et la pente est proche de 1. Cependant, la pente n'est pas exactement 1 et on observe ici aussi une plus grande variance pour le premier quart des valeurs de *Fitted Values*.

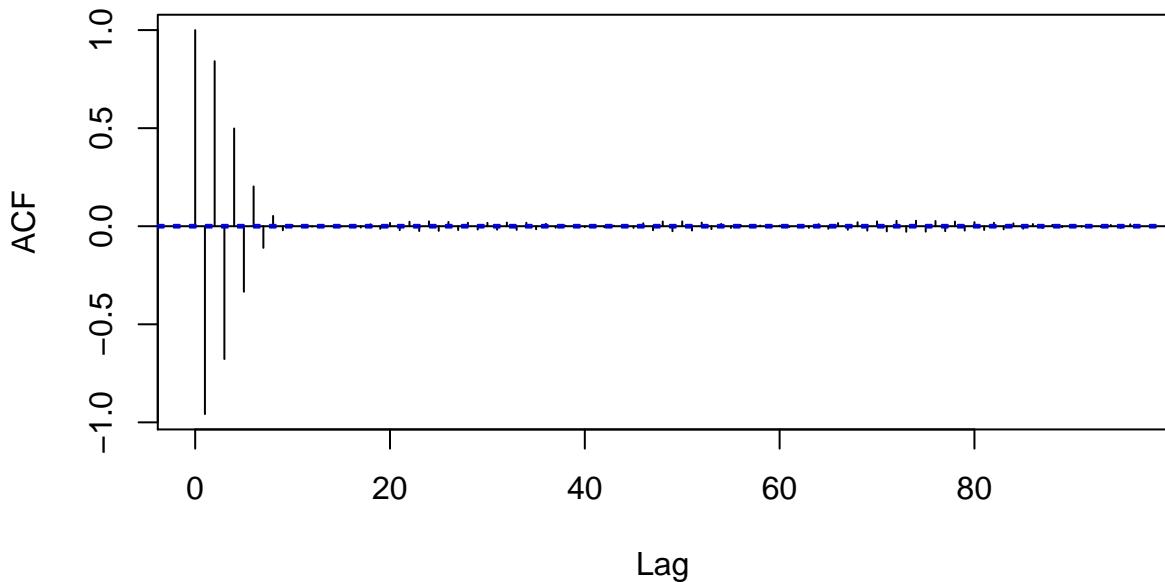
On conclut l'étude des résidus par l'observation de son caractère stationnaire. Lors de l'étude de la stationnarité de la température, on avait remarqué que les 2 saisonnalités étaient bien présentes sur les autocorrélogrammes. On représente donc 2 autocorrélogrammes avec une échelle journalière (4 jours) puis annuelle (2 ans) pour voir si les 2 saisonnalités sont toujours présentes dans les résidus :



Sur le premier graphique, on ne distingue plus la forte autocorrélation annuelle. Cela veut dire que notre modèle modélise correctement la saisonnalité annuelle et on pouvait s'y attendre car cette saisonnalité est très régulière dans le temps et sa fréquence est faible sur le jeu de données.

Sur le second graphique, on distingue la saisonnalité journalière des résidus, ce qui est problématique. Cela veut dire que notre modèle ne parvient pas à modéliser correctement cette saisonnalité. Ce phénomène est peut-être dû à l'irrégularité de la saisonnalité journalière et sa forte fréquence, ou bien à un manque d'informations dans notre modèle.

De manière générale, ces deux autocorrélogrammes montrent que les résidus ne sont pas stationnaires. On observe que l'autocorrélation diminue fortement à l'échelle annuel mais elle reste trop haute et n'appartient pas à l'intervalle de confiance. La raison de ce mauvais comportement semble être la mauvaise modélisation de la saisonnalité journalière. Pour déterminer cette intuition, on représente l'autocorrélogramme des résidus où on applique la fonction `diff()` avec le paramètre `differences = 24` pour retirer la saisonnalité journalière.

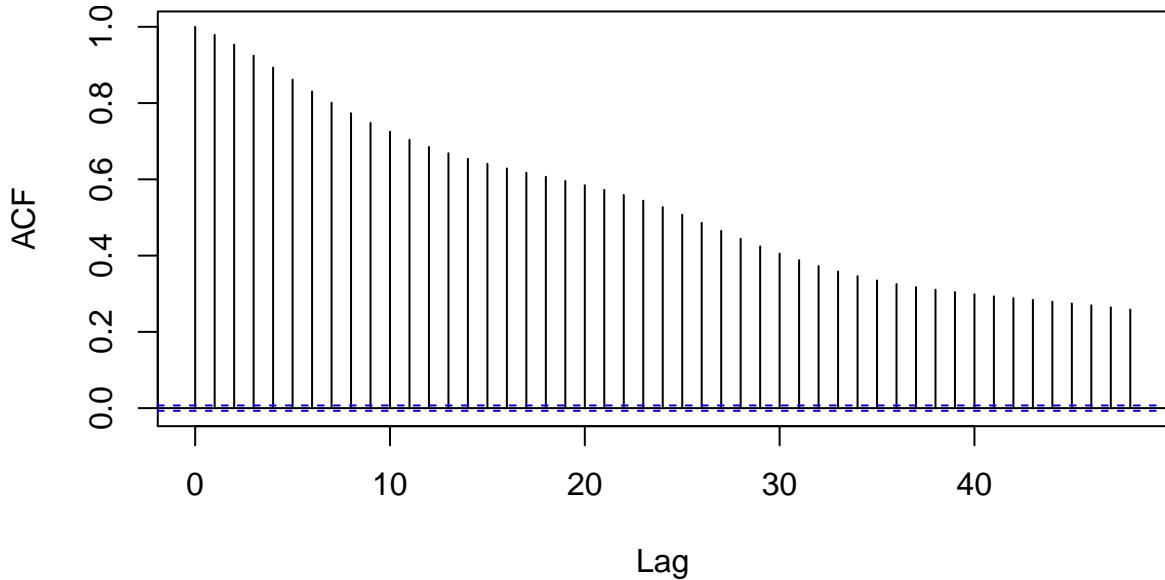


Ces 2 graphiques confirment notre intuition. L'autocorrélation diminue de manière exponentielle et devient très faible. Les valeurs des ACF sont proches de l'intervalle de confiance. Lorsqu'elles sont incluses dans l'intervalle de confiance, on peut les considérer comme nulles et donc affirmer la stationnarité. Cependant, on distingue encore des valeurs de l'ACF en dehors de l'intervalle de confiance lorsque les valeurs de Lag approchent les multiples de 24. La saisonnalité journalière est donc encore présente dans les résidus mais de manière très faible et les valeurs de l'ACF sont beaucoup plus faibles que précédemment. On peut presque

conclure la stationnarité des résidus après application de la fonction `diff()`.

On conclut notre étude du caractère stationnaire des résidus par la non stationnarité à cause de la mauvaise modélisation de la saisonnalité journalière. Cependant, nos résidus restent satisfaisant car la correction de ce problème pourrait conclure la stationnarité des résidus. Nous avons tenté de résoudre ce problème en ajoutant des variables, en modifiant les splines du GAM et même en ajoutant des termes d'interaction entre les variables. En effet, ce dernier pourrait régler le problème car si la variable **Heure** intéragie avec la variable **NumSem**, alors notre modèle ne prend pas en compte cette information et cela pourrait expliquer le mauvais comportement journalier. Cependant, l'ajout des termes d'interaction n'a pas amélioré la stationnarité des résidus (ni les observations précédentes sur leur répartition gaussienne).

Finalement, on construit un nouveau modèle **resModelGAM** sur les résidus pour voir s'il peut modéliser la saisonnalité journalière restante. On l'étudie en utilisant son résumé et son autocorrélogramme.



L'intercept et la fonction smooth `s(Heure,bs='cc',k=24)` ne sont pas significatifs dans les tests de significativités de Student. Le R-ajusté et le score GCV sont très mauvais et l'autocorrélogramme montre que le modèle **resModelGAM** ne parvient pas à modéliser la saisonnalité journalière. De plus, comme précédemment, en utilisant la fonction `diff()` avec le paramètre `differences = 24`, on obtient la stationnarité. On conclut que le problème ne vient pas directement de notre modèle **modelGAM** mais peut-être de la trop haute fréquence de la saisonnalité journalière ou bien d'un manque d'information dans notre modèle.

Conclusion

Dans ce projet nous avons étudié des données avec saisonnalités complexes. Cela se caractérise par l'existence de deux saisonnalités en deux échelles très différentes (une étant journalière, l'autre annuelle).

Notre première approche est l'utilisation de la classe *MSTS* du package *forecast* dans R. Nous avons ainsi obtenu des résultats qui modélisent bien les deux saisonnalités, et ce modèle a été capable de faire une prévision d'un an avec les variations journalières présentes. Ensuite nous avons créé un modèle additif généralisé (GAM) avec 2 variables, chacune pour modéliser une saisonnalité. Après un choix de variables pertinent, nous avons réussi à faire une modélisation qui modélise les deux saisonnalités ainsi que faire une prévision avec un score RMSE meilleur que le modèle MSTS.

Lors de la création de notre modèle GAM, nous avons présenté une réflexion et une justification de chacun de nos choix : choix des variables, choix des splines, de leur paramètres et aussi l'étude de l'interaction de nos 2 variables sur le modèle. Nous avons vu que l'interaction est trop linéaire pour qu'elle ait un impact

significatif sur les performances du modèles. Enfin, nous avons étudié les résidus afin d'évaluer l'efficacité du modèle. Le comportement des résidus est satisfaisant avec une répartition presque gaussienne et on peut presque les considérer comme un bruit blanc. Cependant, les résidus ne sont pas parfaits et nous avons vu que c'est surtout la saisonnalité journalière qui n'est pas parfaitement modélisée par le modèle qui provoque le comportement moyen des résidus. Une piste de résolution de ce problème est l'utilisation d'autres modèles et d'autres outils afin de modéliser d'une meilleure manière nos données.

Références

- Golub, G., H. Heath, and G. Wahba. 1979. “Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter.” *Technometrics* 21: 215–24.
- Hastie, Trevor, and Robert Tibshirani. 1990. *Generalized Additive Models*. Wiley Online Library.
- Hyndman, Rob. n.d.a. “Complex Seasonality.” <https://otexts.com/fpp2/complexseasonality.html>.
- . n.d.b. “Forecast’ Package Changelog.” <https://pkg.robjhyndman.com/forecast/news/index.html>.
- Marra, Giampiero, and Simon N. Wood. 2011. “Practical Variable Selection for Generalized Additive Models.” *Computational Statistics & Data Analysis* 55 (7): 2372–87. [https://doi.org/https://doi.org/10.1016/j.csda.2011.02.004](https://doi.org/10.1016/j.csda.2011.02.004).