Université Claude Bernard Lyon 1

MASTER 1 MATHÉMATIQUES APPLIQUÉES, STATISTIQUE

2024 - 2025

Projet en Mathématiques appliquées

Prévision du facteur de charge éolien chez RTE

Encadrante : Cécile Mercadier

Etudiants : Komlan Katakou et Jana Taleb

Remerciements

Nous tenons à exprimer nos chaleureux remerciements à notre encadrante, ***Mme Cécile Mercadier****, pour ses conseils avisés, son soutien constant et ses précieuses orientations tout au long de ce projet. Son expertise et sa disponibilité nous ont été d'une grande aide pour surmonter les défis techniques auxquels nous nous sommes heurtés.*

Nous adressons également nos sincères remerciements à ***Mr Christophe Poquet****, responsable du Master 1, et à tous les enseignants de cette formation.*

Enfin, nous souhaitons remercier nos camarades de promotion pour les échanges constructifs que nous avons eus et le soutien mutuel durant ce semestre.

Table des matières

Première partie

Introduction

Chapitre 1

Pourquoi ce projet ?

L'énergie éolienne est un pilier de la transition énergétique. Mais, derrière les pales qui tournent, se cachent une mécanique complexe, des indicateurs techniques et de gros volumes de données dont la bonne compréhension est primordiale pour prévoir la production et le rendement des installations éoliennes.

Ce projet a pour but d'expliquer la production éolienne française et de prédire le facteur de charge (indice de rendement) du parc éolien français à partir des données météorologiques et géographiques en mobilisant des techniques de modélisation statistique.

Le présent rapport comporte :

- un bref aperçu du fonctionnement des éoliennes
- une présentation de la problématique de RTE
- un survol des notions théoriques indispensables
- une étude descriptive des données mises à notre disposition
- une analyse et une explication de la méthodologie et du code fourni par RTE
- quelques suggestions d'amélioration pour avoir un modèle qui gagne en prédictions précises et robustes.

Il est à noter que ce sujet a fait l'objet d'un stage court de six semaines en fin d'année universitaire 2023-2024. Il est important de signaler que nous n'avons jamais disposé de la solution élaborée lors de ce stage. Notre point de départ est le même que lors de ce stage. Nous disposons des codes et des données de RTE avec pour objectif de les comprendre et de les améliorer, et ce, dans une direction assez libre.

Chapitre 2

Comprendre la problématique de RTE

2.1 Description d'une éolienne

Une éolienne convertit l'énergie cinétique du vent en électricité. Elle est constituée de plusieurs éléments mécaniques et électriques, décrits ci-après et visibles sur la Figure 2.1.

Le **rotor**, situé à l'avant de l'éolienne, est composé de 2 ou 3 pales fixées à un **moyeu**. Il capte l'énergie du vent et la transmet à l'**axe de rotation**. Cet axe entraîne une **génératrice** qui transforme l'énergie mécanique en électricité. Un **multiplicateur de vitesse** peut être présent pour augmenter la vitesse de rotation de l'axe, sauf dans les éoliennes à entraînement direct.

L'ensemble technique (génératrice, multiplicateur, système de contrôle) est logé dans la **nacelle**, placée en haut du **mât**. Ce mât, généralement composé de 3 à 5 tronçons en acier ou en béton, soutient l'éolienne à une hauteur optimale, là où le vent est plus régulier et puissant.

À la base du mât se trouvent les **armoires de commande** et souvent un **transformateur** électrique. Le mât est ancré dans des **fondations** en béton armé dimensionnées selon le type de sol et les contraintes mécaniques.

Le rotor tourne généralement dans le sens horaire lorsqu'on regarde l'éolienne de face. La zone balayée par les pales est appelée **surplomb**. Les réseaux électriques enterrés relient les éoliennes entre elles et au réseau général, conformément à la réglementation française, qui impose l'enfouissement de toutes les lignes du parc éolien.

2.2 La production éolienne

Le vent fait tourner les pales, entraînant un axe qui transmet l'énergie mécanique au générateur via un multiplicateur. L'électricité produite est ensuite acheminée vers le réseau via un transformateur. La production réelle dépend de :

- la vitesse du vent (très variable dans le temps et dans l'espace)
- la disponibilité technique propre à chaque modèle d'éolienne.

2.3 Régime de fonctionnement : Courbe de puissance

La production d'une éolienne dépend évidemment du vent qui souffle plus fort au niveau de sa nacelle qu'au niveau du sol.

Quand le vent est trop faible le rotor ne reçoit pas assez d'énergie cinétique et l'éolienne ne tourne pas. En l'absence de vent ou pour un vent très faible, bien que le rotor puisse être lé-

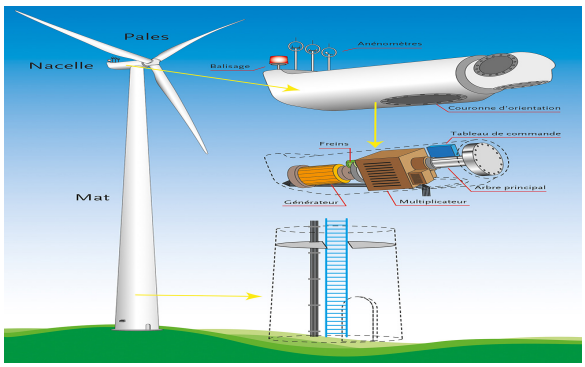


FIGURE 2.1 – Description d'une éolienne

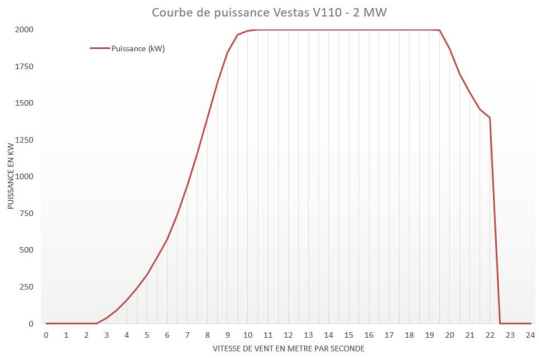


FIGURE 2.2 – Courbe de puissance d'une éolienne - Source : www.eolise.fr

gèrement en mouvement, la production électrique est donc nulle. Les éoliennes commencent à produire de l'électricité à partir de leur vitesse de démarrage, généralement à 3 m/s, l'équivalent d'une petite brise sur l'échelle de Beaufort.

Dès que la vitesse du vent atteint le seuil de démarrage, l'éolienne commence à produire de l'électricité. La puissance éolienne étant proportionnelle au cube de la vitesse du vent, la puissance produite augmente très vite avec la vitesse du vent et l'éolienne atteint sa puissance nominale (le maximum qu'elle puisse produire) pour sa vitesse optimale de vent. Cette vitesse se situe selon les modèles entre 10 et 13 m/s (**probablement 12 m/s pour les éoliennes de RTE**, voir section 6.2). A partir de cette vitesse, l'éolienne produit à sa puissance maximale.

Enfin, les éoliennes sont prévues pour fonctionner jusqu'à une vitesse maximum de vent, dite vitesse d'arrêt. Si la vitesse du vent dépasse cette limite, elles ne peuvent plus produire d'électricité en toute sécurité et sans provoquer une usure accélérée des composants. La vitesse d'arrêt se situe selon les modèles aux alentours de 22 à 30 m/s (79 à 108 km/h) soit « tempête » sur l'échelle de Beaufort. Lorsque le vent dépasse cette vitesse d'arrêt, l'éolienne positionne automatiquement son rotor face au vent et ses pales « en drapeau ». Les freins sont serrés et dans cette configuration très rare, l'éolienne peut résister à des vents allant jusqu'à des tornades.

Le régime de fonctionnement des éoliennes peut être visualisé à travers un graphique appelé courbe de puissance. Elle trace la puissance produite en fonction de la vitesse du vent. Par exemple, pour le modèle d'éolienne *Vestas V110* (modèle très fréquent en France) de 2 MW de puissance nominale , la courbe de puissance est donnée par la Figure 2.2.

2.4 Puissance installée

La puissance installée d'une éolienne (ou d'un parc éolien) est la puissance électrique maximale que l'éolienne peut produire dans des conditions de vent optimales, définie par le constructeur. Dans le cas de notre étude, c'est la puissance électrique que le parc éolien devrait produire si la vitesse du vent était toujours de 12 m/s en tout temps et en toute zone. C'est une valeur théorique (et non la production effective) qui sert de référence technique pour estimer la capacité de production.

2.5 Facteur de charge

Le facteur de charge est un indicateur officiel utilisé par les organismes énergétiques dont RTE pour évaluer le rendement des installations éoliennes. Il est défini comme le ratio entre la production réelle et la production maximale théorique de l'éolienne (c'est-à-dire sa production si

le vent soufflait 100 % du temps à la vitesse optimale de 12 m/s).
En effet, il est fréquent qu'une éolienne ne produise de l'électricité que pendant une partie du temps. Mais cette production n'est pas non plus tout le temps au niveau maximal que l'éolienne pourrait atteindre, faute de vent suffisant. Et puisque l'éolienne ne tourne pas toujours à plein régime, elle ne produit qu'une portion de sa capacité maximale. Si on fait le quotient entre le niveau de production électrique d'une éolienne sur une année par rapport à ce qu'elle pourrait produire si le vent était toujours idéal, on obtient le fameux *facteur de charge*.

$$\boxed{\text{FC} = \frac{\text{Production réelle}}{\text{Puissance installée}}}$$

A titre d'exemple, le facteur de charge moyen du parc éolien français varie généralement entre 21 et 25 % selon les années.

Le facteur de charge est donc un indicateur clé de performance d'une installation éolienne reflétant l'adéquation entre l'équipement et le potentiel éolien local.

2.6 Justification de la démarche choisie

Le choix de modéliser le ratio `FC` (plutôt que la production éolienne toute brute) repose sur des considérations physiques et techniques : à 12 m/s, les éoliennes atteignent leur puissance maximale. En deça de ce seuil, elles produisent une énergie très faible et au-delà de la vitesse d'arrêt, elles ne fonctionnent plus. Ainsi, la puissance installée incarne le potentiel idéal de production, servant de référence pour évaluer la performance réelle par le biais du facteur de charge.

En plus d'être un indicateur universel de la performance d'une installation éolienne, le facteur de charge présente aussi de bonnes propriétés statistiques de stationnarité et de faible variablité, ce qui en fait une variable plus robuste à modéliser.

Deuxième partie

Fondements théoriques

Chapitre 3

Modèles Linéaires Généralisés (GLMs)

Ceci est un résumé du chapitre 2 du livre (WOOD 2017).

3.1 Présentation du modèle

Les modèles linéaires généralisés (GLMs) sont une extension des modèles linéaires classiques autorisant une distribution de la variable expliquée Y non nécessairement normale et une relation non linéaire entre l'espérance $\mathbb{E}(Y)$ et les prédicteurs via une **fonction de lien**. On dispose :

- d'une loi pour les variables réponses $(Y_i)_{1\leq i\leq n}$ appartenant à la famille exponentielle
- de n variables explicatives $(X_i)_{1\leq i\leq n}$
- d'une fonction de lien g qui relie linéairement les espérances $\mu_i = \mathbb{E}(Y_i)$ aux prédicteurs.

Le modèle s'écrit :

$$\boxed{g(\mu_i) = g(\mathbb{E}(Y_i)) = X_i\beta} \tag{3.1}$$

où X_i est la $i-$ème ligne (associée à l'observation i) de la matrice $X \in \mathcal{M}_{n,p}(\mathbb{R})$ du modèle et $\beta \in \mathbb{R}^p$ est le vecteur des paramètres du modèle. La fonction g est supposée monotone et suffisamment lisse . Comme exemple, le cas $g(\mu) = \mu$ correspond à celui du modèle linéaire gaussien ordinaire, le cas $g(\mu) = \log(\mu)$ est celui d'une régression de Poisson et le cas $g(\mu) = \log\left(\frac{\mu}{1-\mu}\right)$ est celui de la régression logistique. La densité des Y_i s'écrit :

$$f(y_i|\theta_i,\phi) = \exp\left(\frac{y_i\theta_i - b_i(\theta_i)}{a_i(\phi)} + c_i(y_i,\phi)\right),$$

avec ϕ paramètre de dispersion et θ le paramètre canonique de la famille exponentielle. Dans bon nombre de cas comme dans la suite de ce rapport, on considère les fonctions a_i de la forme $a_i(\phi) = \frac{\phi}{\omega_i}$.
Grâce à des calculs simples impliquant les équations du score et l'expression de l'information de Fisher dans un modèle régulier (c'est le cas de la famille exponentielle), on obtient une expression des moments :

$$\mathbb{E}(Y_i) = \mu_i = b'(\theta_i) \quad \text{et} \quad Var(Y_i) = b''(\theta_i)a(\phi), \quad \forall i \in \{1,\dots,n\}$$

3.2 Estimation des paramètres

L'estimation du vecteur de paramètres β se fait par **maximum de vraisemblance**. On veut maximiser :

$$\ell(\beta) = \sum_{i=1}^{n}\left[\frac{y_i\theta_i - b(\theta_i)}{\phi} + c_i(y_i,\phi)\right].$$

On annule les dérivées partielles de ℓ pour la détermination de β, ce qui aboutit à :

$$\frac{\partial \ell}{\partial \beta_j} = \sum_{i=1}^{n} \frac{(y_i - \mu_i)}{V(\mu_i)} \cdot \frac{\partial \mu_i}{\partial \beta_j} = 0, \quad \text{où} \quad V(\mu_i) = \frac{Var(Y_i)}{\phi}.$$

Si les $V(\mu_i)$ étaient connus et indépendants de β, ces équations seraient celles obtenues en cherchant β par moindres carrés pondérés avec pour fonction objectif $S = \sum_{i=1}^{n} \frac{(y_i-\mu_i)^2}{V(\mu_i)}$. On obtient ainsi un algorithme (connu sous le nom d'IRLS) d'approximation de β par moindres carrés pondérés. En notant $\beta^{[k]}$ l'estimation à l'itération k de β, $\eta_i^{[k]} = X_i\beta^{[k]}$ et $\mu_i^{[k]} = g^{-1}(\eta_i^{[k]})$ et en initialisant avec un $\beta^{[0]}$, on réitère jusqu'à convergence les étapes :

- Calculer les $V(\mu_i^{[k]})$ associés au $\beta^{[k]}$ actuel
- Minimiser S en β avec $V(\mu_i)$ fixé égal à $V(\mu_i^{[k]})$, $\forall i \in \{1, \ldots, n\}$
- Incrémenter k : $k \leftarrow k + 1$

3.3 Mesure de l'ajustement et sélection de modèle

Une fois les paramètres estimés, plusieurs critères permettent d'évaluer la qualité du modèle :

Critère d'information d'Akaike (AIC)

L'objectif de l'AIC est de trouver un équilibre entre la qualité de l'ajustement (vraisemblance élevée) et la complexité du modèle (en évitant un nombre de paramètres trop grand). Cela consiste à minimiser la quantité :

$$AIC = -2\ell(\hat{\beta}) + 2k, \quad \text{où} \quad k \quad \text{est le nombre de paramètres identifiables du modèle.}$$

Comparaison de modèles par test d'hypothèse

Ce test est basé sur la notion de déviance. La déviance est une mesure de l'ajustement d'un modèle aux données par rapport à un modèle saturé et est définie par :

$D = 2[\ell(\hat{\beta}_{max}) - \ell(\hat{\beta})]\phi$ où $\ell(\hat{\beta}_{max})$ est la log-vraisemblance maximisée pour le modèle saturé et

$l(\hat{\beta})$ est la log-vraisemblance maximisée pour le modèle ajusté.
Pour comparer un modèle restreint à p_0 paramètres (hypothèse H_0) et un modèle complet à p_1 paramètres (hypothèse H_1), on peut se servir de la statistique de test :

$$F = \frac{(D_0 - D_1)/(p_1 - p_0)}{D_1/(n - p_1)} \sim \text{Fisher}(p_1 - p_0, n - p_1), \quad \text{car, sous } H_0, \quad \frac{D_0 - D_1}{\phi} \sim \chi^2_{p_1-p_0}.$$

où D_0 est la déviance du modèle restreint et D_1 celle du modèle complet. Une *p-value* faible indique un meilleur ajustement pour le modèle complet et donne donc des raisons de rejeter H_0.

Remarque : Un modèle plus complexe réduit toujours la déviance, mais cela peut entraîner un sur-ajustement. C'est pourquoi des critères comme l'AIC sont souvent utilisés.

Statistique de Pearson et estimation de ϕ

Un estimateur de ϕ est donné par : $\hat{\phi} = \frac{1}{n-p} \sum_{i=1}^{n} \frac{(y_i-\hat{\mu}_i)^2}{V(\hat{\mu}_i)}$. Cet estimateur fait intervenir la statistique de Pearson :

$$X^2 = \sum_{i=1}^{n} \frac{(y_i - \hat{\mu}_i)^2}{V(\hat{\mu}_i)} \sim \chi^2_{n-p}$$

qui est utilisée pour l'ajustement global du modèle. Si la valeur observée est élevée (très grande devant l'espérance $n - p$ de la chi-deux) , cela indique un mauvais ajustement du modèle.

Résidus de Pearson

Pour un bon modèle, les résidus de Pearson définis par $\hat{\epsilon}_i^p = \frac{y_i - \hat{\mu}_i}{\sqrt{V(\hat{\mu}_i)}}$ doivent être approximativement centrés et de variance ϕ. Ils ne doivent présenter aucune tendance en moyenne ou en variance quand on les trace contre les valeurs ajustées.

3.4 Implémentation des GLMs dans R

On se propose ici de modéliser la proportion du nombre de patients déclenchant une crise cardiaque en fonction du niveau de concentration de l'enzyme créatine-kinase (CK) dans le sang. Notons n le nombre de patients et, pour $i \in \{1, \dots, n\}$, p_i la proportion de patients au niveau x_i ayant déclenché une crise cardiaque.

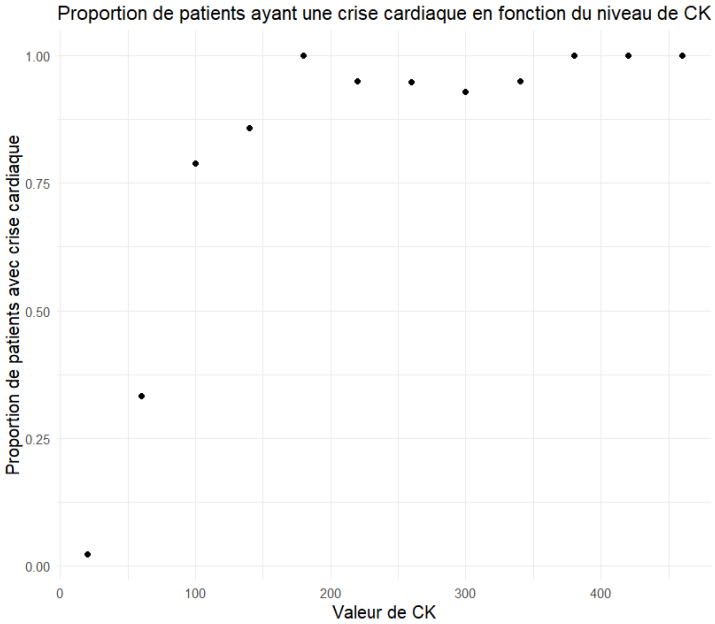


FIGURE 3.1 – Proportion de patients ayant déclenché une crise cardiaque en fonction du niveau de CK

Se basant sur la figure 3.1 qui a une allure sigmoide, un modèle convenable serait de la forme :

$$\mathbb{E}(p_i) = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}} \quad , \quad \forall i \in \{1, \dots, n\} \tag{3.2}$$

Si μ_i et N_i désignent respectivement le nombre moyen de crises cardiaques et le nombre total de patients au niveau x_i, en prenant g la fonction *logit*, on a :

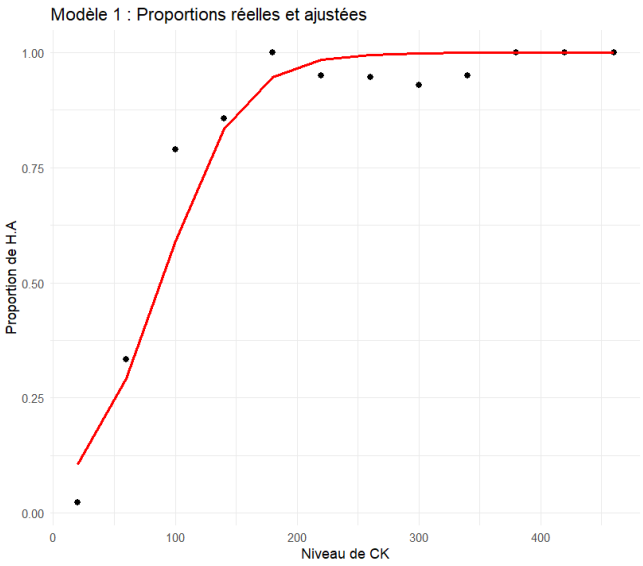
$$g(\mu_i) = g(\mathbb{E}(p_i N_i)) = \log\left(\frac{\mu_i}{N_i - \mu_i}\right) = \beta_0 + \beta_1 x_i \quad , \quad \forall i \in \{1, \dots, n\} \tag{3.3}$$

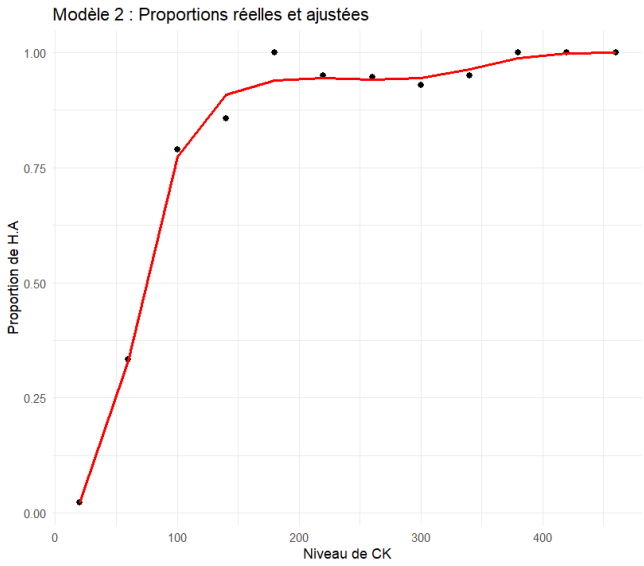
et on a alors un modèle linéaire en les paramètres, via la fonction g.

Le code R du listing 3.1 saisit les données et ajuste deux modèles logistiques via la fonction `glm()` de `R`.

Listing 3.1 – Modélisation de la proportion d'une crise cardiaque selon le niveau de CK

```
# Niveaux de CK
ck <- c(20, 60, 100, 140, 180, 220, 260, 300, 340, 380, 420, 460)

# Nombre de patients ayant declenché une crise cardiaque pour chaque niveau
ha <- c(2, 13, 30, 30, 21, 19, 18, 13, 19, 15, 7, 8)  # ha = heart attack

# Nombre de patients n'ayant pas declenché de crise
ok <- c(88, 26, 8, 5, 0, 1, 1, 1, 1, 0, 0, 0)

# Construction du dataframe
heart <- data.frame(ck, ha, ok)

# Ajustement d'un modèle simple
modele1 <- glm(cbind(ha, ok) ~ ck, family = binomial, data = heart)

# Affichage du résumé du modèle
summary(modele1)

# Tracé des valeurs ajustées
p <- heart$ha/(heart$ha + heart$ok)  # Proportion réelle de H.A

plot(heart$ck , p ,xlab="Niveau␣de␣CK", ylab="Proportion␣de␣H.A")

lines(heart$ck, fitted(modele1))

# Ajustement d'un deuxième modèle
modele2 <- glm(cbind(ha, ok) ~ ck + I(ck^2) + I(ck^3), family=binomial,
data=heart)

# Affichage du résumé du modèle
summary(modele2)

# Tracé des valeurs ajustées
plot(heart$ck , p ,xlab="Niveau␣de␣CK", ylab="Proportion␣de␣H.A")

lines(heart$ck, fitted(modele2))
```






```
Call:
glm(formula = cbind(ha, ok) ~ ck, family = binomial, data = heart)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.758358   0.336696  -8.192 2.56e-16 ***
ck           0.031244   0.003619   8.633  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 271.712  on 11  degrees of freedom
Residual deviance:  36.929  on 10  degrees of freedom
AIC: 62.334
```

```
Call:
glm(formula = cbind(ha, ok) ~ ck + I(ck^2) + I(ck^3), family = binomial,
    data = heart)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.786e+00  9.268e-01  -6.243 4.30e-10 ***
ck           1.102e-01  2.139e-02   5.153 2.57e-07 ***
I(ck^2)     -4.649e-04  1.381e-04  -3.367  0.00076 ***
I(ck^3)      6.448e-07  2.544e-07   2.535  0.01125 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 271.7124  on 11  degrees of freedom
Residual deviance:   4.2525  on  8  degrees of freedom
AIC: 33.658
```

Comparaison des modèles

Le premier modèle montre un faible ajustement confirmé par les résultats du `summary(modele1)`, notamment la déviance résiduelle de 36.93 qui est très grande devant l'espérance 10 d'une χ^2 à 10 degrés de liberté dont elle est censée suivre la loi. Un graphique supplémentaire (omis dans ce rapport) obtenu par la commande `plot(modele1)` montre une tendance en moyenne des résidus et suggère un ajustement cubique que met en œuvre le `modele2`. On a alors une courbe qui s'ajuste bien aux données, une valeur de l'AIC ayant considérablement diminué (de 62.33 à 33.66) et une déviance résiduelle de 4.25 qui est une valeur raisonnable pour une χ^2_{10}.

D'ailleurs, en comparant les deux modèles via la fonction `anova()` qui implémente un test de rapport de vraisemblance généralisé, on obtient une p-value de l'ordre de 10^{-8}, ce qui confirme une meilleure adéquation du `modele2` aux données.

Conclusion

Ce chapitre définit un cadre rigoureux pour les modèles linéaires généralisés. Il met en évidence la généralisation du modèle linéaire classique, l'estimation des paramètres par la méthode IRLS, l'évaluation du modèle via de nombreuses méthodes et montre une application des GLMs sous le logiciel R à divers types de données où l'on peut, selon le cas, se ramener à un modèle log-linéaire, une régression logistique ou de Poisson. Le chapitre termine en exposant des résultats sur le maximum de vraisemblance (consistance, normalité asymptotique,...) que nous avons étudiés en cours de Statistique paramétrique.

Chapitre 4

Modèles Additifs Généralisés (GAMs)

Ceci est un résumé du chapitre 3 du livre (WOOD 2017).

4.1 Présentation du modèle

Les GAMs généralisent les GLMs en autorisant des relations **non linéaires** entre les covariables (variables prédictives) et la variable réponse. Leur structure générale est :

$$\boxed{g(\mathbb{E}(Y_i)) = X_i^*\theta + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip})} \tag{4.1}$$

- g est la fonction de lien
- $(Y_i)_{1\leq i\leq n}$ est la variable réponse, de loi appartenant à la famille exponentielle
- X_i^* est la i-ème ligne de la matrice X^* représentant la partie strictement paramétrique du modèle
- θ est le vecteur des paramètres du modèle
- Les $(f_j)_{1\leq j\leq p}$ sont des fonctions lisses des prédicteurs $(x_k)_{1\leq k\leq n}$

La matrice X^* permet de décrire la dépendance linéaire tout en autorisant le modèle à capturer des effets non linéaires avec les fonctions lisses f_j.

4.2 Fonctions de Lissage Univariées

Considérons le cas univarié où la réponse y est expliquée par une seule covariable x. On a :

$$y_i = f(x_i) + \varepsilon_i, \quad \text{avec les} \quad \varepsilon_i \quad \text{i.i.d. de loi} \quad \mathcal{N}(0, \sigma^2) \tag{4.2}$$

L'équation (4.2) n'est qu'une réécriture de (4.1) où les informations des parties linéaire et non-linéaire sont renseignées dans f. L'estimation de f dans le cadre du modèle linéaire requiert l'écriture de (4.2) de sorte qu'il devienne linéaire en les paramètres. Cela peut être fait en décomposant f dans une base de fonctions $(b_j)_{1\leq j\leq q}$, comme suit :

$$f(x) = \sum_{j=1}^{q} \beta_j b_j(x) \tag{4.3}$$

En substituant (4.3) dans (4.2), on obtient le modèle $y_i = \sum_{j=1}^{q} \beta_j b_j(x_i) + \varepsilon_i$, linéaire en les paramètres $(\beta_j)_{1\leq j\leq q}$. Une manière naturelle d'estimer une fonction continue est d'utiliser une approximation polynomiale mais les polynômes de degré élevé ont tendance à produire des oscillations indésirables et leur extrapolation peut être très irréaliste. C'est pourquoi on leur préfère les **splines de régression.**

4.2.1 Représentation en bases de Splines

Une spline de régression est une fonction lisse utilisée pour approximer une relation non linéaire. On utilisera principalement les splines cubiques qui sont des fonctions composées de morceaux de polynômes de degré 3, raccordés en des points appelés noeuds, de manière à assurer une continuité des dérivées première et seconde en ces noeuds. La fonction f se décompose ainsi en une combinaison linéaire de splines :

$$f(x) = \sum_{j=1}^{q} \beta_j b_j(x) \tag{4.4}$$

Le nombre et la position des noeuds influent sur la souplesse de la spline et donc sur la flexibilité du modèle. Par exemple, un nombre élevé de noeuds présente des risques de surajustement. En notant $\{x_i^*, i = 1, \ldots, q-2\}$ l'ensemble des noeuds, on a une base de splines cubiques donnée par : $b_1(x) = 1, b_2(x) = x \quad \text{et} \quad \forall j \in \{1, \ldots, q-2\}, b_j(x) = R(x, x_j^*)$, avec :

$$\begin{aligned} R(x,z) =& \frac{1}{4}\left(\left(z-\frac{1}{2}\right)^2 - \frac{1}{12}\right)\left(\left(x-\frac{1}{2}\right)^2 - \frac{1}{12}\right) \\ & - \left(|x-z| - \frac{1}{2}\right)^4 - \frac{1}{2}\left(|x-z| - \frac{1}{2}\right)^2 + \frac{7}{240} \end{aligned} \tag{4.5}$$

4.2.2 Contrôle du degré de lissage : Pénalisation

Le choix du degré de lissage et du nombre de noeuds est crucial pour l'ajustement du modèle. Une méthode adéquate est celle de l'ajout d'une pénalité sur la rugosité de la spline : plutôt que d'ajuster le nombre de noeuds, ce qui affecte la dimension de la base, on fixe un nombre de noeuds légèrement supérieur à ce que l'on estime être nécessaire puis on ajoute une pénalité qui décourage les vibrations excessives dans la spline. Cette pénalité est ajoutée à l'objectif des moindres carrés et pénalise les modèles à grandes fluctuations, i.e. les grandes dérivées secondes de la spline.

Ainsi, au lieu de minimiser $||Y - X\beta||^2$, on minimise plutôt $||Y - X\beta||^2 + \lambda\beta^T S\beta$. Le deuxième terme de cette noivelle fonction objectif (à λ près) n'est autre que l'intégrale de la dérivée seconde de f qui s'écrit ici comme une forme quadratique en β car f est linéaire en les β_j.

Connaissant λ, on montre que l'objectif des moindres carrés pénalisés se minimise en

$$\hat{\beta} = (X^T X + \lambda S)^{-1} X^T Y \tag{4.6}$$

On note au passage la matrice d'influence du modèle $A = X(X^T X + \lambda S)^{-1} X^T$. En pratique, l'expression (4.6) de $\hat{\beta}$ est peu usitée. On se sert plutôt de la décomposition :

$$\left\| \begin{bmatrix} Y \\ 0_q \end{bmatrix} - \begin{bmatrix} X \\ \sqrt{\lambda} B \end{bmatrix} \beta \right\|^2 = \|Y - X\beta\|^2 + \lambda\beta^T S\beta, \quad \text{avec} \quad B \quad \text{telle que} \quad B^T B = S,$$

dont on déduit que le problème des moindres carrés pénalisés n'est autre que celui des moindres carrés non pénalisés où la matrice du modèle est augmentée de $\sqrt{\lambda}B$ et les données Y_i augmentés de q zéros.

Donc, on peut estimer le modèle à λ connu.

4.3 Sélection du paramètre de lissage : Validation croisée

Si f et $\hat{f}$ désignent respectivement la vraie fonction et son estimation, un bon critère serait de minimiser en λ la quantité

$$M(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(\hat{f}_i(\lambda) - f_i)^2 \tag{4.7}$$

où la dépendance en λ des $\hat{f}_i$ et de M sera omise par simplicité dans la suite .

M reste inconnue à cause de f mais on peut en trouver une estimation. Pour cela, on définit le **score de validation croisée ordinaire** :

$$\nu_o = \frac{1}{n}\sum_{i=1}^{n}(\hat{f}_i^{[-i]} - y_i)^2$$

où $\hat{f}_i^{[-i]}$ représente l'estimation de f avec toutes les données sauf l'observation i. En utilisant le caractère centré des ε_i et l'approximation $\hat{f}_i^{[-i]} \approx \hat{f}$, on obtient :

$$\mathbb{E}(\nu_o) \approx \mathbb{E}(M) + \sigma^2 \tag{4.8}$$

Choisir λ pour minimiser ν_o (et donc M grâce à (4.8)) est appelé **validation croisée ordinaire**. On peut montrer que

$$\nu_o = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}_i)^2/(1 - A_{ii})^2 \tag{4.9}$$

Une autre quantité est le score de **validation croisée généralisée**

$$\mathrm{GCV}(\lambda) = \frac{n\sum_{i=1}^{n}(y_i - \hat{f}_i)^2}{(n - \mathrm{tr}(A))^2}, \tag{4.10}$$

qui est facilement minimisable pour trouver le λ optimal.

4.4 Modèles Additifs

Un modèle additif combine plusieurs fonctions de lissage, par exemple pour un cas bivarié

$$y_i = f_1(x_i) + f_2(z_i) + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2). \tag{4.11}$$

Ce modèle repose sur l'hypothèse d'additivité (le caractère non couplé) des effets de x et z, ce qui est une hypothèse assez forte. De plus, la non-identifiabilité apparaît lorsque $f_1(x)$ et $f_2(z)$ contiennent des constantes arbitraires. En imposant $\gamma_1 = 0$ dans

$$f_1(x) = \delta_1 + \delta_2 x + \sum_{j=1}^{q_1-2}\delta_{j+2}R(x, x_j^*), \quad f_2(z) = \gamma_1 + \gamma_2 z + \sum_{j=1}^{q_2-2}\gamma_{j+2}R(z, z_j^*) \tag{4.12}$$

on s'assure de l'identifiabilité. La i-ème ligne de la matrice du modèle devient :

$$X_i = \left[1, \quad x_i, \quad R(x_i, x_j^*), \quad z_i, \quad R(z_i, z_j^*)\right]^T, \quad \text{où} \quad j = 2, \ldots, q_1 \quad \text{puis de nouveau} \quad j = 2, \ldots, q_2$$

La rugosité de l'ajustement est évaluée par $\lambda_1\beta^T S_1\beta + \lambda_2\beta^T S_2\beta$.
En posant $S = \lambda_1 S_1 + \lambda_2 S_2$, l'objectif devient :

$$\left\| \begin{bmatrix} Y \\ 0_q \end{bmatrix} - \begin{bmatrix} X \\ B \end{bmatrix} \beta \right\|^2, \quad \text{avec} \quad B \quad \text{telle que} \quad B^T B = S \tag{4.13}$$

L'estimation se ramène alors à une régression linéaire classique, ajustable via la fonction `lm()` de `R`. Les paramètres de lissage λ_1 et λ_2 optimaux sont obtenus par validation croisée généralisée.

4.5 Modèles Additifs Généralisés

Les modèles additifs généralisés (GAMs) sont une extension des modèles additifs, tout comme les GLMs le sont pour les modèles linéaires ordinaires. Dans ce cas, le prédicteur linéaire prédit une transformation lisse et monotone de l'espérance de la réponse, qui peut suivre une distribution quelconque de la famille exponentielle. Ils s'écrivent :

$$g(\mathbb{E}(Y_i)) = f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}), \quad Y_i \sim \text{famille exponentielle} \tag{4.14}$$

Contrairement aux modèles additifs classiques, qui sont ajustables par moindres carrés pénalisés, les GAMs nécessitent une maximisation de vraisemblance pénalisée. En pratique, cela est accompli par la version pénalisée de l'algorithme IRLS (vu au chapitre 3), qui réitère jusqu'à convergence les étapes suivantes :

1. À partir des estimations actuelles des paramètres $\beta^{[k]}$, calculer $\mu^{[k]}$ et définir :

$$w_i \propto \frac{1}{V(\mu_i^{[k]})g'(\mu_i^{[k]})}, \quad z_i = g(\mu_i^{[k]}) + (Y_i - \mu_i^{[k]})g'(\mu_i^{[k]}), \text{avec} \quad \text{Var}(Y_i) = V(\mu^{[k]}) \tag{4.15}$$

2. Minimiser en β la quantité pénalisée suivante :

$$||W^{1/2}(z - X\beta)||^2 + \beta^T S\beta \tag{4.16}$$

où W est la matrice diagonale contenant les poids w_i et S la matrice de pénalité du lissage.

Pour l'optimisation des paramètres de lissage, on peut encore procéder par validation croisée généralisée.

Conclusion

Ce chapitre introduit les modèles additifs généralisés en présentant leur grande flexibilité par rapport aux modèles linéaires généralisés . En utilisant des splines de régression pour capturer les effets non linéaires tout en contrôlant la complexité avec des pénalités sur le lissage par validation croisée, l'ajustement d'un GAM apparaît comme un compromis entre fidélité aux données et régularisation des fonctions lisses, ce qui permet d'éviter un surajustement et d'obtenir des modèles interprétables.

Troisième partie

Données

Chapitre 5

Etude descriptive des données

Dans cette partie est présentée une analyse exploratoire de quelques uns des jeux de données mis à disposition. Ils sont de trois types :

- Les données géographiques renseignées dans le jeu de données `ZC_grille_01_elargie.csv`
- Les données météorologiques contenues dans six jeux de données et couvrant la période 2017 - 2022
- Les données de production éolienne sur la période 2017 - 2022

5.1 Jeu de données `ZC_grille_01_elargie.csv`

5.1.1 Description

Le jeu de données `ZC_grille_01_elargie.csv` contient des informations géographiques sur les sites de mesure de la vitesse du vent en France métropolitaine, ainsi que leur classification en **10 zones climatiques distinctes**.

Il se compose de **6 351 observations** de **3 variables**, décrites dans le tableau ci-après.

Nom de la variable	Description	Valeurs observées
`Lon`	Longitude du site en degrés.	$[-4.7, 8.1]$
`Lat`	Latitude du site en degrés.	$[42.4, 51.0]$
`ZC.adapte`	Zone climatique attribuée au site.	$[\![1, 10]\!]$

TABLEAU 5.1 – Structure du jeu de données `ZC_grille_01_elargie.csv`.

Cette classification regroupe les sites en tenant compte des disparités climatiques régionales et des différences de conditions météorologiques, influençant directement la production éolienne. Cependant, l'algorithme ayant conduit à cette classification (valeur de l'étiquette `ZC.adapte`) n'a pas été fourni dans les informations disponibles : elle est donnée telle quelle.

5.1.2 Visualisation graphique

Une première visualisation peut être faite pour représenter le contenu de ce jeu de données : une carte de la France montrant la répartition des zones climatiques (Figure 5.1) et des histo-

grammes (Figure 5.2) montrant le nombre de sites par zone.

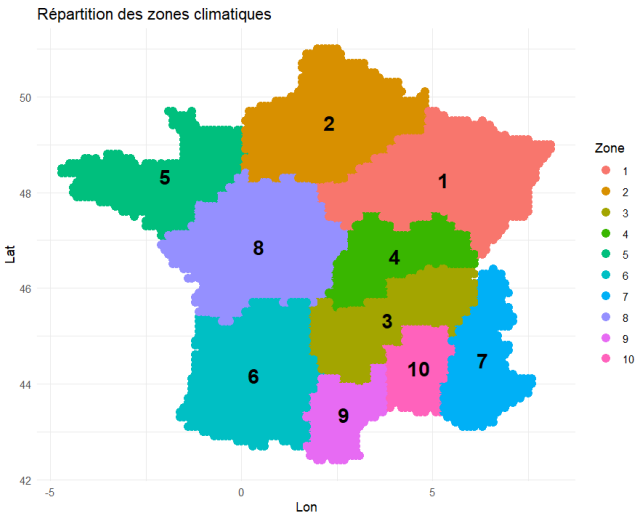


FIGURE 5.1 – Répartition des zones climatiques

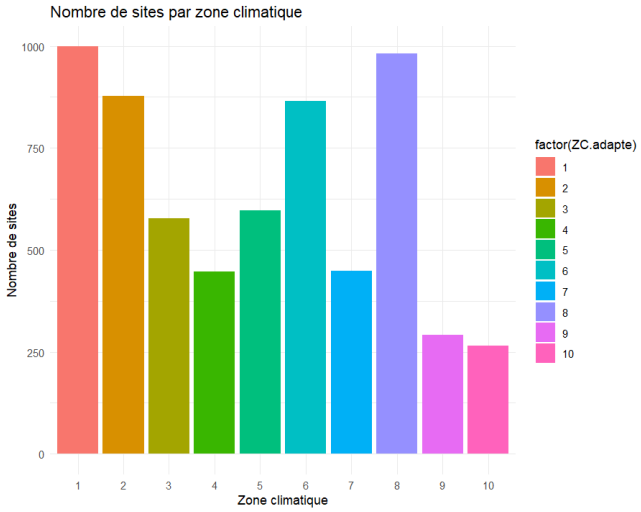


FIGURE 5.2 – Nombre de sites par zone climatique

5.2 Jeu de données `wind_2017.nc`

Plusieurs fichiers de vent (`wind _2017 à _2022`) sont fournis. Nous présentons le premier.

Ce fichier est de l'extension `.nc` et nous utilisons les fonctions du package `ncdf4` (REARTHSYSSCI 2020) pour le manipuler.

5.2.1 Introduction au format NetCDF et au package `ncdf4` de `R`

Le format `NetCDF` (Network Common Data Form) est un format largement utilisé pour travailler sur des données climatiques, mais il est aussi utilisé dans d'autres domaines produisant de grandes quantités de données multidimensionnelles. Les fichiers NetCDF sont dits autodescriptifs, car ils contiennent des **métadonnées** qui décrivent leur contenu (par exemple, la disposition des coordonnées de latitude et longitude, les noms et unités des variables, ainsi que les attributs tels que les codes de valeurs manquantes, ...).

Les fichiers NetCDF présentent l'avantage d'une grande flexibilité au niveau machine, permettant notamment leur transfert entre serveurs et ordinateurs sous différents systèmes d'exploitation sans nécessiter de conversion.

Il existe deux versions de NetCDF : la version `netCDF4` qui est présentée ici est la plus répandue. Elle prend en charge des ensembles de données très grands et inclut de multiples fonctionnalités. `R` offre la capacité de lire et d'écrire des fichiers `NetCDF` grâce au package `ncdf4`.

5.2.2 Ouverture et Description du fichier

La fonction `nc_open()` permet de charger le fichier pour des manipulations ultérieures. En affichant l'objet qu'il retourne, on a une idée globale de la structure du fichier.

```
wind2017 <- nc_open("~/data/wind_era5/wind_2017.nc")
print(wind2017)
```

```
2 variables (excluding dimension variables):
   short u100[longitude,latitude,time]
       scale_factor: 0.000829374955909171
       add_offset: 4.05863110414802
       _FillValue: -32767
       missing_value: -32767
       units: m s**-1
       long_name: 100 metre U wind component
   short v100[longitude,latitude,time]
       scale_factor: 0.000801986852202 34
       add_offset: -0.96512553413655
       _FillValue: -32767
       missing_value: -32767
       units: m s**-1
       long_name: 100 metre V wind component

3 dimensions:
   longitude  Size:71
       units: degrees_east
       long_name: longitude
   latitude  Size:51
       units: degrees_north
       long_name: latitude
   time  Size:8760
       units: hours since 1900-01-01 00:00:00.0
       long_name: time
       calendar: gregorian

2 global attributes:
   Conventions: CF-1.6
   history: 2024-05-07 08:06:55 GMT by grib_to_netcdf-2.28.1: /opt/ecmwf/mars-client/bin/grib_to_netcdf -S param -o /ca
che/data5/adaptor.mars.internal-1715069164.9778883-16503-4-fcb6fde8-f2f3-4002-a465-626e3bd52de2.nc /cache/tmp/fcb6fde8-f2f3-
4002-a465-626e3bd52de2-adaptor.mars.internal-1715068475.248474-16503-4-tmp.grib
```

FIGURE 5.3 – Description du fichier `wind_2017.nc`

D'après la sortie de la commande `nc_open()` (Figure 5.3), on peut décrire le contenu. du fichier `wind_2017.nc`. Il contient des données sur la vitesse du vent à 100 mètres d'altitude pour l'année 2017. Ces données sont observées en deux variables :

- `u100` : Composante U (direction Est-Ouest) du vent à 100 mètres, mesurée en m/s
- `v100` : Composante V (direction Nord-Sud) du vent à 100 mètres, mesurée en m/s

Chacune des variables est observée sur une grille à trois dimensions :

- 71 points de longitude (de $-5.0°$ à $9.0°$)
- 51 points de latitude (de $42.0°$ à $52.0°$)
- 8760 dates (correspondant aux heures de l'année 2017)

Les données manquantes sont codées par la valeur `-32767`. Notons que le fichier ne contient pas les données brutes mais plutôt normalisées grâce à une échelle (`scale_factor`) et un décalage (`add_offset`) pour chacune des variables ; les vraies valeurs des composantes U et V du vent sont donc obtenues en multipliant par l'échelle et en ajoutant le décalage appropriés aux données normalisées.

5.2.3 Autres manipulations

Toutes les informations que nous avons lues dans la sortie produite par `nc_open(wind2017)` peuvent être récupérées grâce aux deux fonctions que nous introduisons ici : `ncvar_get()` et `ncatt_get()`.

Accès aux dimensions du fichier

Pour obtenir des informations (attributs) sur les dimensions, telles que leurs noms longs ou leurs unités, on se sert de la fonction `ncatt_get()`.

```
time_units <- ncatt_get(wind2017, "time", "units")
time_name <- ncatt_get(wind2017, "time", "long_name")
time_calend <- ncatt_get(wind2017, "time", "calendar")
```

Accès aux variables

On se sert de la fonction `ncvar_get()` pour récupérer les valeurs d'une variable et de `ncatt_get()` pour récupérer les attributs de la variable d'intérêt :

```
lon <- ncvar_get(wind2017, "longitude")
u100 <- ncvar_get(wind2017, "u100")
u100_attrs <- ncatt_get(wind2017, "u100")
u100_units <- ncatt_get(wind2017, "u100", "units")
u100_fill <- ncatt_get(wind2017, "u100", "_FillValue")
```

On peut par exemple vérifier que `u100` a bien les bonnes dimensions et essayer d'afficher l'unité des mesures de `u100` :

```
print(dim(u100))
print(u100_units$value)
```

```
[1] 71   51   8760
[1] "m s**-1"
```

Fermeture du fichier

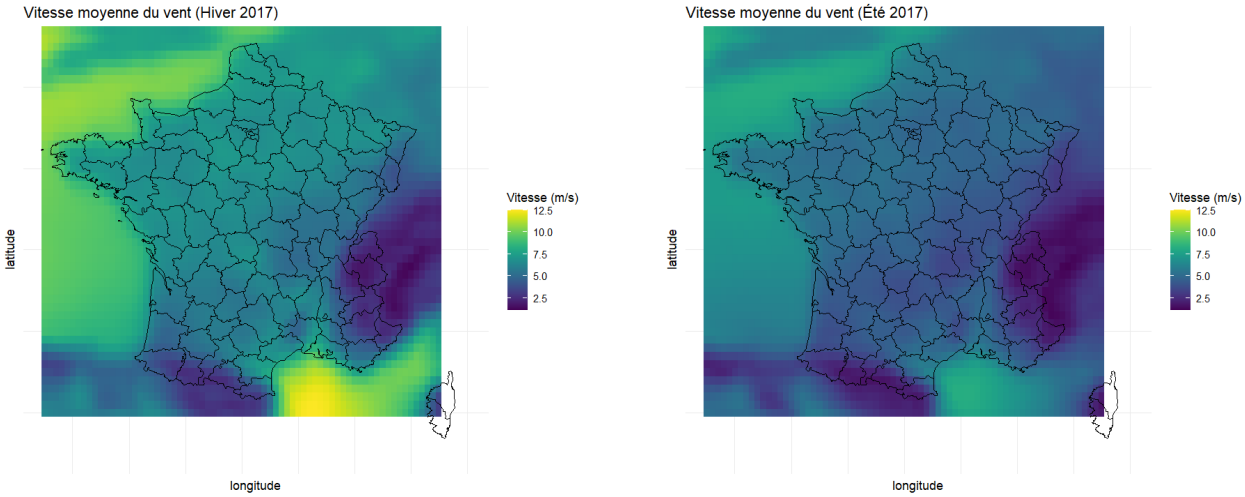
```
nc_close(wind2017)
```

Une fois les données extraites, il est important de fermer le fichier NetCDF avec la fonction `nc_close()` pour libérer les ressources mémoire utilisées.

5.3 Etude descriptive de la vitesse du vent

5.3.1 Approche saisonnière

Dans l'exploration des données, nous avons remarqué des effets saisonniers sur la vitesse du vent. Pour cela, nous proposons de visualiser la répartition de la vitesse du vent sur tout le territoire français à différentes périodes de l'année, par exemple, les moyennes hivernales et estivales. C'est ce que montrent les *heatmaps* ci-dessous.



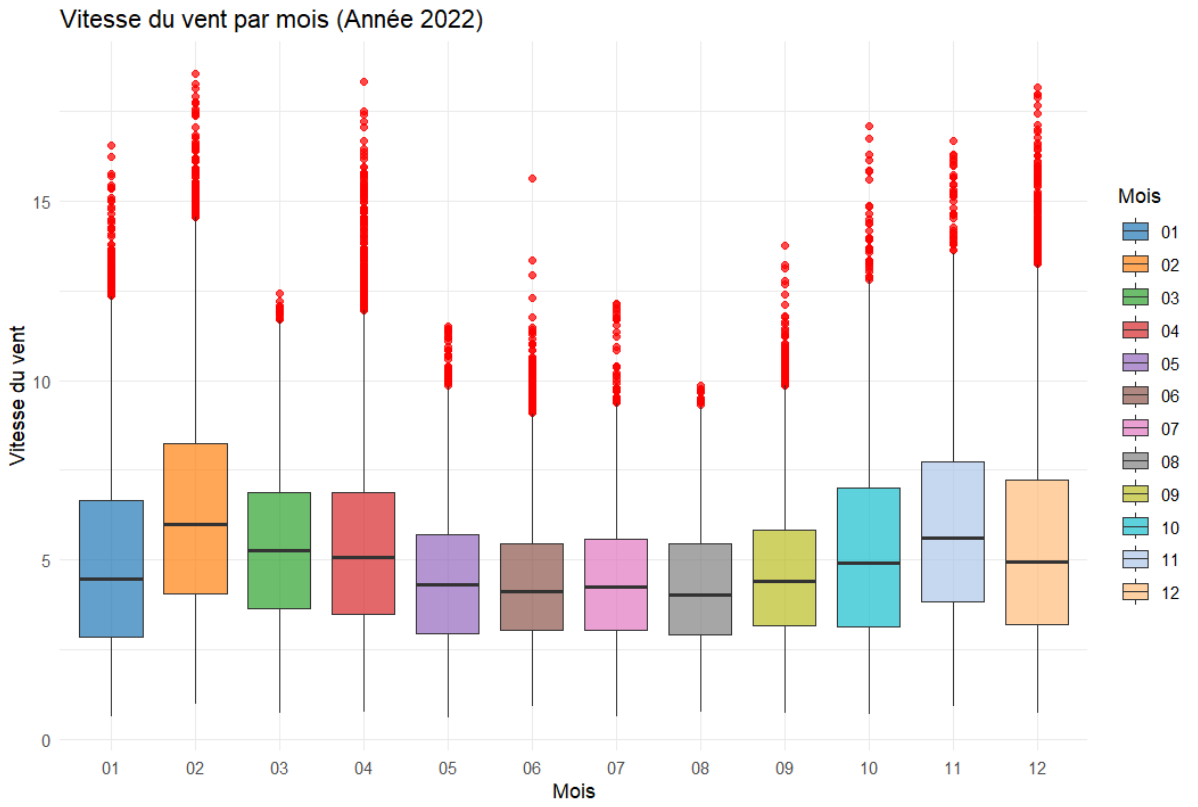


FIGURE 5.4 – Vitesse du vent en 2022

Ces représentations donnent lieu à de plusieurs constats :

- En hiver, la vitesse du vent est élevée sur la plus grande partie du territoire, à l'exception de la côte sud-ouest et de la frontière italo-suisse.
- En été, les vitesses sont plus faibles surtout dans la moitié sud et restent relativement élevées dans la moitié nord.
- En revanche, toute l'année, on a de très grandes vitesses de vent sur tout le pourtour méditerranéen.

Ces constats sont corroborés par les Figures 5.4, 5.5 et 5.6 qui offrent une analyse comparative de la distribution de la vitesse du vent selon la période de l'année.

5.3.2 Approche zonale

Les graphiques 5.5 et 5.6 révèlent bien entendu une tendance saisonnière claire, avec une augmentation notable de la vitesse du vent durant les mois d'hiver et une diminution pendant l'été. Cependant, au-delà de ces variations saisonnières, un autre phénomène est observable : des zones spécifiques se distinguent par des vitesses de vent particulièrement faibles, tandis que d'autres présentent des vitesses nettement plus élevées. Ce dernier effet semble être indépendant des saisons, suggérant que des facteurs locaux ou régionaux influencent de manière significative la vitesse du vent dans certaines régions.

Les histogrammes de la Figure 5.7 illustrent au mieux la distribution de la vitesse du vent par zone.

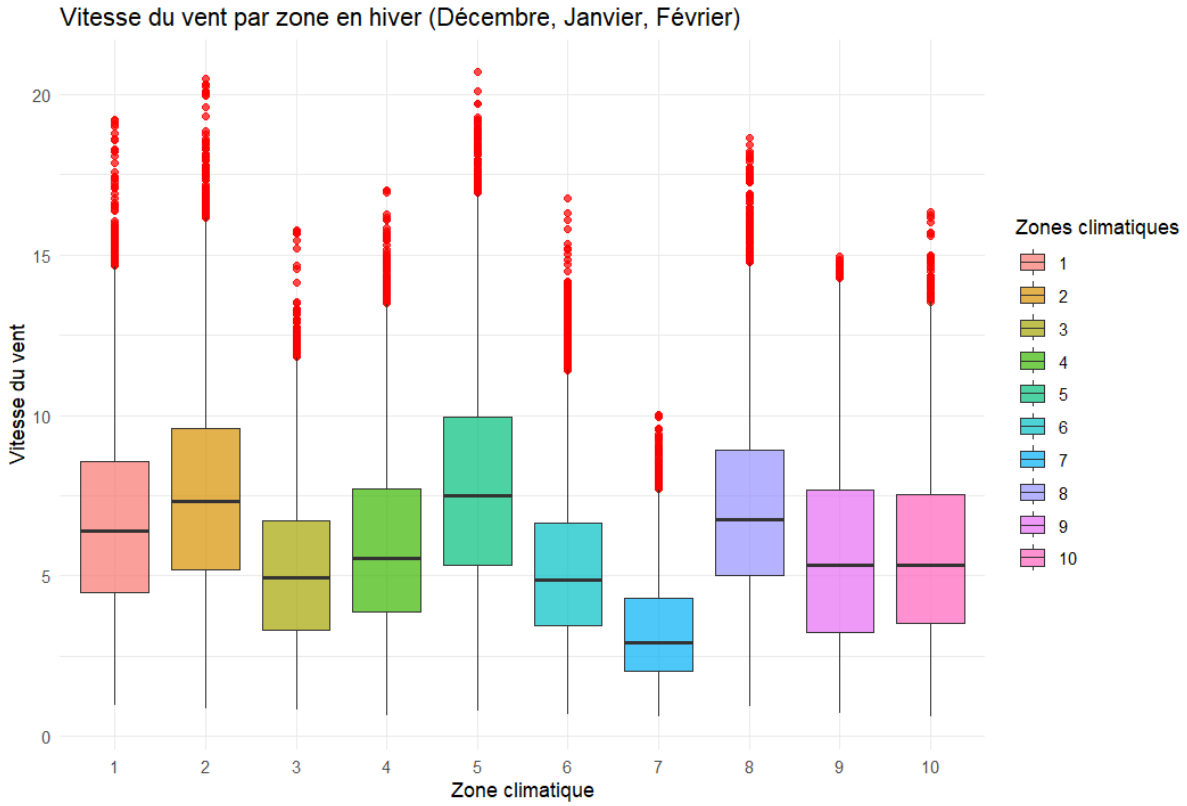


FIGURE 5.5 – Vitesse du vent en hiver

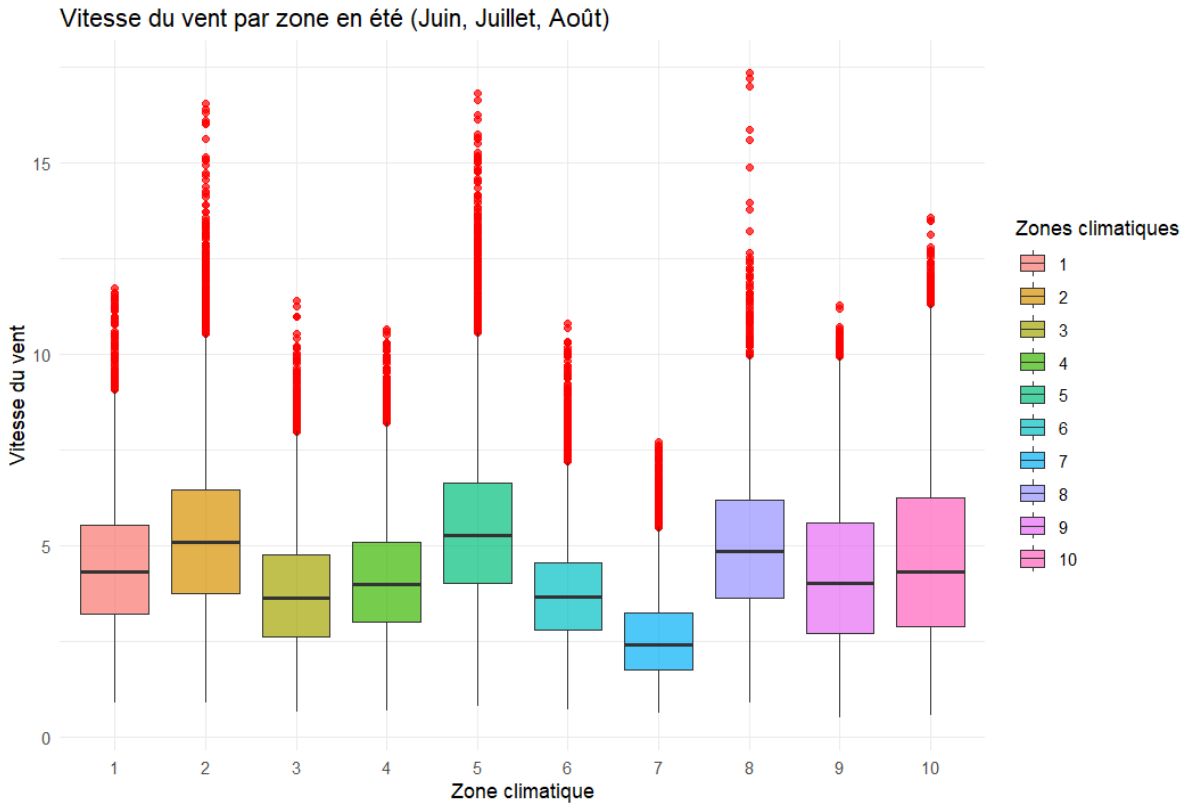


FIGURE 5.6 – Vitesse du vent en été

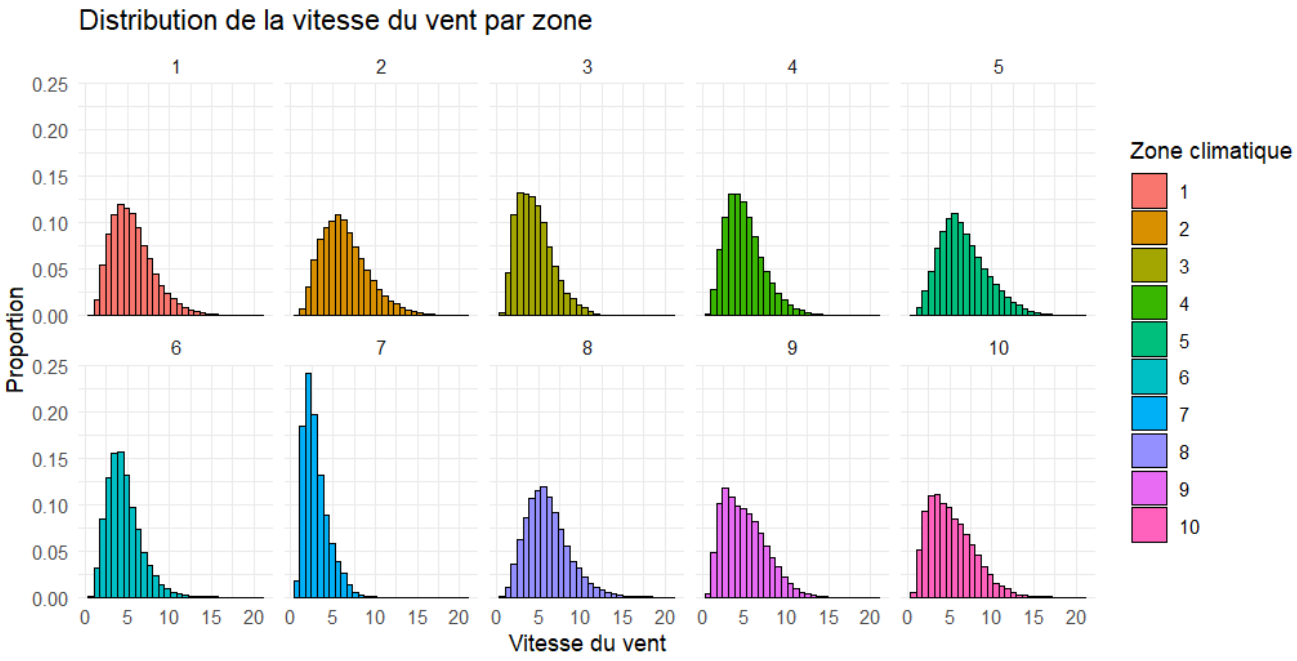


FIGURE 5.7 – Vitesse du vent par zone

Tests statistiques

Afin de confirmer les constats suggérés par les histogrammes, deux tests d'hypothèses peuvent être mis en oeuvre pour tester :

- (H_0) : La distribution de la vitesse du vent est identique sur toutes les zones
  contre
- (H_1) : Il y a une différence significative entre les zones

Test de Kruskal - Wallis et Test de Dunn

- Le test (non paramétrique) de Kruskal-Wallis est une extension du test de Mann-Whitney pour plus de deux groupes. Il est utilisé pour tester si plusieurs échantillons indépendants proviennent de la même distribution. Appliqué sur les données de vent des dix zones (avec une filtrattion temporelle de façon à obtenir des échantillons «indépendants»), il donne le résultat du Listing 5.1.

Listing 5.1 – Résultat du test de Kruskal-Wallis sur la variable `mean_ff100` selon `ZC.adapte`

```
kruskal_result <- kruskal.test(mean_ff100 ~ factor(ZC.adapte), data = df_
    aggregated)
print(kruskal_result)

        Kruskal-Wallis rank sum test

data:  mean_ff100 by factor(ZC.adapte)
Kruskal-Wallis chi-squared = 4690.4, df = 9, p-value < 2.2e-16
```

  On obtient une p-value presque nulle, ce qui permet de rejeter l'hypothèse nulle selon laquelle les distributions de la vitesse du vent sur les dix zones sont identiques.
- Le test de Dunn est utilisé après un test de Kruskal-Wallis pour effectuer des comparaisons multiples entre les groupes. Cependant, dans les tests multiples, le taux d'erreur de

première espèce augmente, donc il est important d'ajuster les p-values. Une solution courante est donnée par la méthode de Bonferroni. En appliquant ce test sur nos données, on obtient généralement de petites p-values comme le montre la *heatmap* de la Figure 5.8.

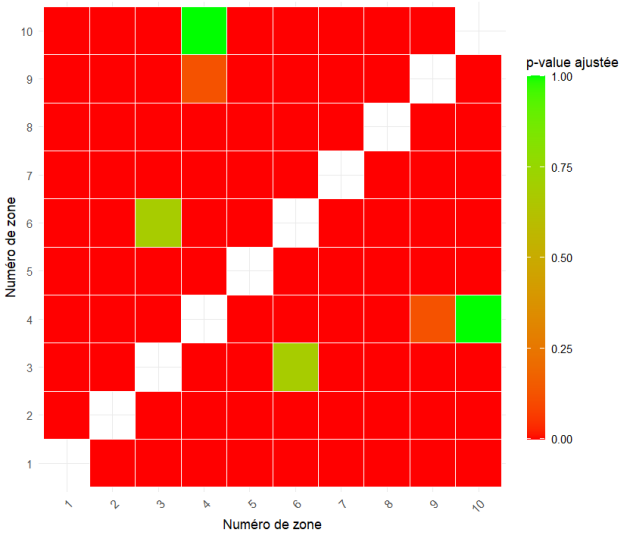


FIGURE 5.8 – Résultats du Test de Dunn

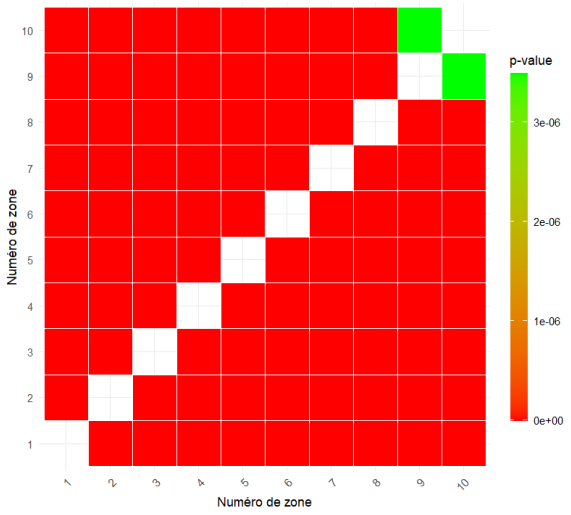


FIGURE 5.9 – Test de Kolmogorov - Smirnov

Test de Kolmogorov - Smirnov

Le test de Kolmogorov-Smirnov est utilisé pour tester si deux échantillons proviennent de la même distribution. En l'appliquant sur chacun des couples de zones, on peut synthétiser les résultats avec les p-values représentées sur la *heatmap* de la Figure 5.9.

Synthèse

Le test de Kolmogorov - Smirnov permet de voir qu'il y a des différences significatives entre les distributions de la vitesse du vent sur les dix zones. Cela est confirmé par les résultats du test de Dunn selon lesquels il existe des différences significatives mais moins marquées entre les zones 3 et 6 d'une part et entre 4 et 10 d'autre part.

Conclusion

L'étude menée dans ce chapitre révèle une double influence : une variabilité saisonnière marquée et un effet de zone persistant sur la vitesse du vent. Il faudra donc exploiter les dynamiques locales et saisonnières mises en évidence pour améliorer la précision des prévisions dans la partie modélisation.

Quatrième partie

Prévision de la production éolienne : Modélisation statistique

Chapitre 6

Analyse du code et de la méthodologie de RTE

Ce chapitre est consacré à l'analyse détaillée du code fourni par RTE, à la compréhension des principes techniques sous-jacents à la méthodologie adoptée ainsi qu'à la modélisation statistique proprement dite. On dispose de :

- la quantité de production éolienne (`eolien`) enregistrée toutes les demi-heures ou tous les quarts d'heure entre 2017 et 2022
- une grille de coordonnées géographiques découpée en 10 zones (résolution : 6351×6351 points)
- la vitesse du vent (`ff_100`) mesurée toutes les heures sur la période 2017 - 2022 en chaque point d'une grille de résolution 71×51 points.

et on cherche à prédire le facteur de charge, `FC = eolien / pi`, où `pi` est la puissance installée (à estimer au préalable).

6.1 Partie 1 : Préparation des données

Chargement des bibliothèques

Le code commence par charger les bibliothèques utiles :

- `dplyr`, `purrr` : Manipulation et agrégation des données
- `tidyr`, `janitor` : Nettoyage des données
- `data.table` : Lecture et manipulation des données
- `lubridate` : Gestion des dates et heures
- `mgcv` : Modélisation GAM
- `ggplot2` : Visualisation graphique

Lecture et agrégation des données de production

```
chemin <- "data/eco2mix/"
df_prod0 <- map(list.files(chemin, pattern = "xls$", full.names = TRUE),
    function(ff) {
  log_info(ff)
  tmp <- read.table(ff, fill = TRUE, header = TRUE, quote = "", sep = "\t",
                    row.names = NULL, fileEncoding = "latin1")
```

```
  names(tmp) = c(names(tmp[,-1]),"dumb")
  clean_names(tmp) %>%
    select(perimetre, nature, date, heures, eolien) %>%
    drop_na()
}) %>%
  rbindlist(use.names = TRUE)
```

Les fichiers du dossier `eco2_mix` contenant entre autres la quantité d'énergie électrique produite par l'éolien sur la période 2017 - 2022 sont chargés . Chaque fichier est ouvert avec la fonction `read.table()` du package `data.table` et les noms de colonnes sont corrigés (avec la fonction `clean_names()` du package `janitor`). Les colonnes `perimetre, nature, date, heures, eolien` sont sélectionnées (fonction `select()` du package `dplyr()`) et les valeurs manquantes sont supprimées à l'aide de la commande `drop_na()`.

Création de la colonne `date_cible`

```
df_prod0[,date_cible:=with_tz(force_tz(ymd_hm(paste(date, heures)), "Europe/
    Paris"), "UTC")]
```

A cette ligne, on crée une nouvelle colonne `date_cible` dans `df_prod0` : on combine les colonnes `date` et `heures` au format "année-mois-jour heure : minute" via la fonction `paste()`. On convertit cette chaîne en un objet Date_et_Heure en spécifiant le fuseau horaire "Europe/Paris" à l'aide de la fonction `force_tz()`, puis le tout est transposé au fuseau horaire UTC à l'aide de la fonction `with_tz()` (package `lubridate`).

Agrégation horaire et visualisation de la production éolienne

```
df_prod <- df_prod0[, .(eolien = mean(eolien)), by = .(date_cible = ceiling_
    date(date_cible, unit = "hour"))]
ggplot(data = df_prod) + geom_line(aes(x = date_cible, y = eolien))
```

Les données sont ensuite agrégées par heure, en arrondissant chaque enregistrement à l'heure la plus proche supérieurement, puis en prenant la moyenne horaire.

On trace ensuite le graphique montrant l'évolution de la production éolienne sur toute la période 2017 - 2022 (Figure 6.1).
On observe un pic saisonnier (annuel) de la production éolienne, avec une baisse notable durant l'été, ce qui est la conséquence directe du comportement saisonnier du vent mis en évidence au chapitre précédent. De plus, la production éolienne présente une tendance clairement croissante sur toute la période, ce qui est dû aux politiques publiques de transition énergétique.

Grille des zones et visualisation

```
df_grille <- fread("data/ZC_grille_01_elargie.csv")
dt_dpt_for_plot <- ggplot2::map_data("france") %>% filter(!stringr::str_detect(
    region, "Corse"))
ggplot(df_grille) +
  geom_raster(aes(x = Lon, y = Lat, fill = factor(ZC.adapte)), alpha = 0.3,
      show.legend = FALSE) + theme_minimal() +
  scale_alpha(range = c(0.2, 0.5), na.value = 0) +
  geom_polygon(aes(long, lat, group = group), fill = NA, colour = "grey70",
      data = dt_dpt_for_plot, alpha = 0.5) +
  geom_text(aes(x = Lon, y = Lat, label = ZC.adapte), df_grille[, .(Lon = mean(
      Lon), Lat = mean(Lat)), by = .(ZC.adapte)]) +
  coord_quickmap()
```

Le fichier `ZC_grille_01_elargie.csv` est chargé à l'aide de la fonction `fread()` du package `data.table`. On renseigne dans `dt_dpt_for_plot` les données cartographiques de la France sans la Corse. Ces données sont utilisées pour visualiser la grille des zones sur une carte à l'aide de la fonction `geom_raster()` tout en ajustant les contours et le fond de carte avec `geom_polygon()`. Ensuite, les numéros des zones contenus dans la colonne `ZC.adapte` sont affichées au centre des zones via `geom_text()` et les zones sont coloriées en fonction de la valeur de l'étiquette `ZC.adapte` (Figure 6.2).

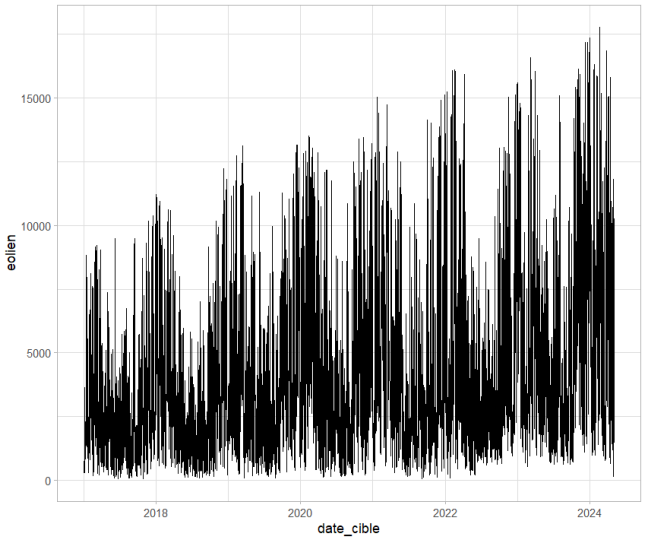


FIGURE 6.1 – Production éolienne

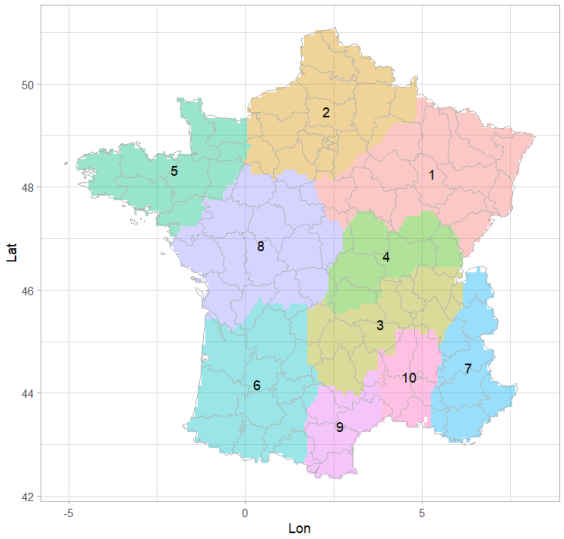


FIGURE 6.2 – Grille des zones

Traitement des données météorologiques

```
df_meteo_zone <- map(list.files("data/wind_era5", "nc", full.names = TRUE),
    function(ff) {
  fid <- ncdf4::nc_open(ff)
  file_dump <- fid$dim

  origin_date <- strsplit(split = "␣since␣", x = file_dump$time$units)[[1]][2]

  ## définition des index
  indexes_names <- list(longitude = as.character(round(file_dump$longitude$vals
      , 2)),
    latitude = as.character(round(file_dump$latitude$vals, 2)),
    time =  as.character(file_dump$time$vals))

  ## extraction array
  array_meteo0 <- sqrt(ncdf4::ncvar_get(nc = fid, varid = "u100")^2+ncdf4::
      ncvar_get(nc = fid, varid = "v100")^2)
  dimnames(array_meteo0) <- indexes_names

  ## conversion data.table
  df_meteo0 <- as.data.frame.table(array_meteo0, stringsAsFactors = FALSE,
      responseName = 'ff100') %>% setDT()
  df_meteo0[,date_cible := ymd_hms(origin_date) + dhours(as.numeric(time))]
  df_meteo0$time <- NULL

  ncdf4::nc_close(fid)

  ## agrégation zonale
  df_meteo_zone_ff <- merge(df_meteo0 %>%
    mutate(longitude = round(as.numeric(longitude), 2),
```

```
    latitude = round(as.numeric(latitude), 2)),
    df_grille %>% mutate(longitude=round(Lon, 2), latitude = round(Lat, 2)),
    by = c("longitude", "latitude"))[
    ,.(ff100 = mean(ff100)), by = .(date_cible, ZC.adapte)]
  df_meteo_zone_ff
}) %>% rbindlist(use.names=TRUE)

df_meteo_zone[(year(date_cible) == 2020) & (month(date_cible) == 1)] %>%
  ggplot() + geom_line(aes(x=date_cible, y=ff100, color = factor(ZC.adapte)))+
  theme_light()

saveRDS(df_meteo_zone %>% pivot_wider(date_cible, names_from = ZC.adapte,
    values_from = ff100, names_prefix = "ff100_"), "data/input_model/df_meteo_
    zone_wide.RDS")})
```

La fonction `map()` du package `purrr` est utilisée pour appliquer une série de manipulations aux fichiers de l'extension `.nc` du dossier `wind_era5`. Chaque fichier est ouvert avec la fonction `nc_open(ff)` du package `ncdf4`. Ces fichiers contiennent des mesures des composantes U et V du vent. La vitesse du vent, calculée en prenant la racine carrée de la somme des carrés des composantes U et V, est stockée dans `array_meteo0`.

Les informations de longitude, latitude et temps sont extraites et utilisées pour nommer les dimensions de `array_meteo0`, puis cet array est converti en un `data.table` avec la fonction `as.data.frame.table()`. La colonne `time` est transformée en un format de date et heure avec la fonction `ymd_hms()` de `lubridate` et ajustée selon l'origine de la date contenue dans les fichiers du dossier `wind_era`.

Une fois la donnée traitée pour chaque fichier, le code ferme le fichier avec `ncdf4::nc_close` et procède à l'agrégation des données par zone. Cette agrégation consiste à fusionner les données de vent avec le jeu de données (`df_grille`), contenant des informations sur les coordonnées géographiques, en arrondissant la longitude et la latitude à deux décimales. Ensuite, la vitesse moyenne (`ff100`) du vent est calculée par date (`date_cible`) et zone (`ZC.adapte`).

Les résultats sont ensuite combinés dans un grand `data.table` via `rbindlist()` et on obtient un tableau complet `df_meteo_zone` à trois colonnes (`date_cible, ZC.adapte, ff100`) avec les données de vent agrégées sur toutes les zones et dates. Enfin, une visualisation graphique de la vitesse du vent est réalisée pour le mois de janvier 2020 (Figure 6.3), et les données sont sauvegardées dans un format large à l'aide de la fonction `pivot_wider()`).

6.2 Estimation de la puissance installée

Un modèle additif généralisé est utilisé pour prédire la production éolienne et estimer la puissance installée du parc éolien à partir de la vitesse du vent et du temps écoulé depuis début 2017.

```
df_meteo_zone_ff <- merge(df_meteo0 %>%
    mutate(longitude = round(as.numeric(longitude), 2),
    latitude = round(as.numeric(latitude), 2)),
    df_grille %>% mutate(longitude = round(Lon, 2), latitude = round(Lat, 2)),
    by = c("longitude", "latitude"))[
    , .(ff100 = mean(ff100)), by = .(date_cible, ZC.adapte)]
mod_gam_noFC_linklog <- bam(eolien ~ s(ff100_fr) + s(time_since_beginning, k =
    7), data = df_fr, family = gaussian(link = "log"), discrete = TRUE)
df_fr[, pi := exp(predict(mod_gam_noFC_linklog, newdata = df_fr %>% mutate(
    ff100_fr = 12)))]
ggplot(df_fr) + geom_line(aes(x = date_cible, y = pi, color = "PI")) +
  geom_line(aes(x = date_cible, y = eolien, color = "prod"))
saveRDS(df_fr[, .(date_cible, pi)], "data/input_model/df_pi.RDS")
```

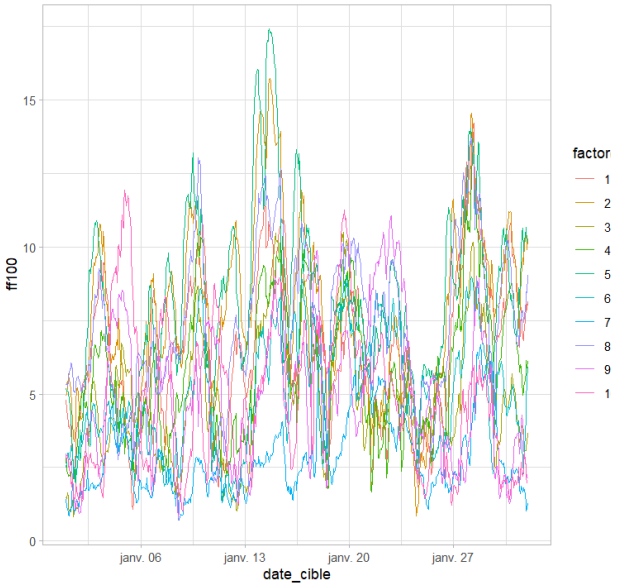


FIGURE 6.3 – Vitesse du vent par zone (Janvier 2020)

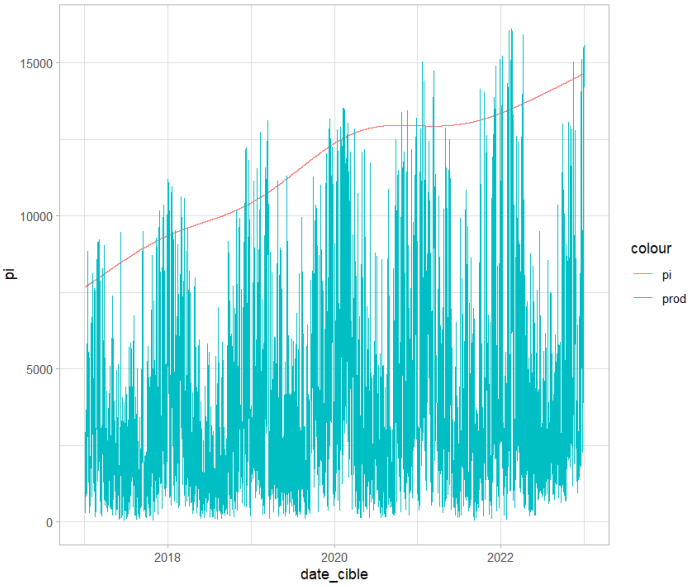


FIGURE 6.4 – Production éolienne et Puissance installée

On commence par fusionner les données de vent (`df_meteo_zone`) et les données de production (`df_prod`) par `date_cible`. Pour ce faire, on se sert de la fonction `merge()` avec un calcul préalable de la vitesse moyenne du vent (`ff100_fr`) par date. Une fois les données fusionnées, une nouvelle colonne `time_since_beginning` est ajoutée pour renseigner le temps écoulé depuis le 1er janvier 2017.

Ensuite, un modèle GAM avec lien logarithmique est ajusté sur les données, à l'aide de la fonction `bam()` du package `mgcv`. Ce modèle utilise la production éolienne (`eolien`) comme variable réponse et deux prédicteurs non linéaires : `ff100_fr` et `time_since_beginning`, en spécifiant un lissage dans une base de splines à 7 éléments.

Après l'ajustement, on calcule une estimation de la puissance installée `pi` en faisant la prédiction (sous le modèle) de la production éolienne pour une vitesse de vent fixée égale à 12 m/s.

La Figure 6.4 trace l'évolution de la production éolienne ainsi que celle de la puissance installée `pi` de tout le parc éolien sur la période 2017 - 2022. Le nuage de points de la Figure 6.5 montre l'influence de la vitesse du vent (`ff100_fr`) sur le ratio `eolien/pi`. La même chose est représentée sur la Figure 6.6 mais avec un aspect lissé et on remarque à chaque fois que le ratio `eolien/pi` est généralement compris entre 0 et 1 (c'est une proportion !).

6.3 Partie 2 : Prévision du facteur de charge

Métrique d'évaluation

```
rmse <- function(obs, prev, na_rm = TRUE) {
  sqrt(mean((prev - obs)^2, na.rm = na_rm))}
```

Une fonction `rmse()` est définie pour calculer la racine carrée de l'erreur quadratique moyenne et sera utilisée ultérieurement pour quantifier la différence entre les prédictions du modèle et les vraies valeurs.

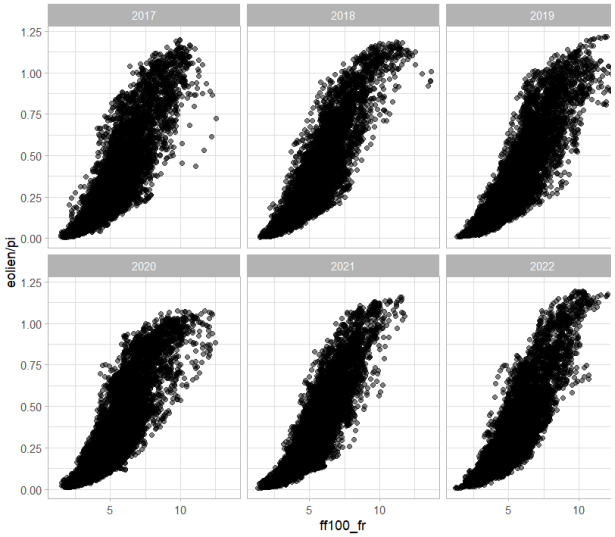


FIGURE 6.5 – Ratio `eolien/pi`

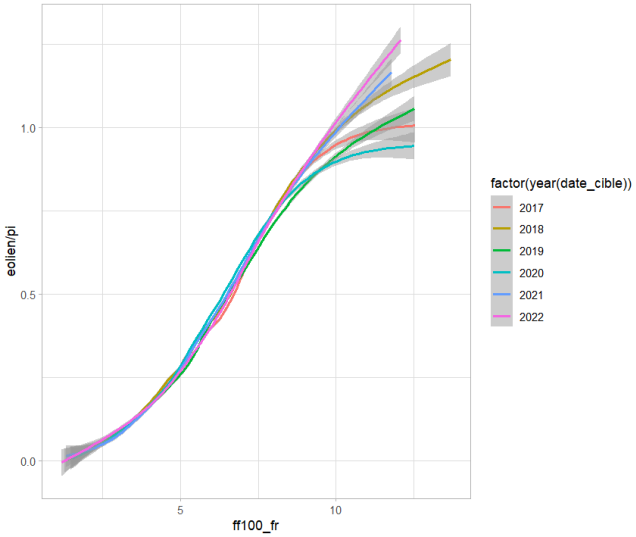


FIGURE 6.6 – Ratio lissé

Fusion des données

```
df_input <- merge(readRDS("data/input_model/df_pi.RDS"),
  readRDS("data/input_model/df_prod.RDS"), by = "date_cible")%>%
mutate(FC = eolien/pi) %>%
merge(readRDS("data/input_model/df_meteo_zone_wide.RDS"), by = "date_cible")
```

Les trois fichiers RDS :

- `df_pi` : contenant la puissance installée prédite
- `df_prod` : contenant les données de production (notamment la variable `eolien`)
- `df_meteo_zone_wide` : contenant les vitesses de vent sur les différentes zones

sont fusionnés via la variable `date_cible`.

Modélisation statistique

```
# Formule GAM avec effets non-linéaires
modgam_10z_formula <- "FC ~ s(ff100_1) + s(ff100_2) + ... + s(ff100_10)"
modgam_10z <- bam(as.formula(modgam_10z_formula), data = df_input, discrete =
    TRUE)
```

A cette étape, un modèle GAM est ajusté pour expliquer le facteur de charge à partir des vitesses de vent sur les 10 zones.

Validation croisée

```
df_prev <- map(unique(year(df_input$date_cible)), function(yy_test) {
  df_input[, train_test := fifelse(
    year(date_cible) == yy_test, "test", "train")]
  if(yy_test != 2022) {
    df_input[year(date_cible) == 2022, train_test := "out"]}
  mod <- bam(as.formula(modgam_10z_formula),
             data = df_input[train_test == "train"], discrete = TRUE)
  df_input[train_test == "test"] %>%
    mutate(prev = predict(mod, newdata = .))
}) %>% rbindlist()
```

```
# Calcul du RMSE
df_prev_pca[, .(rmse = rmse(FC, prev))]
```

- Pour chaque année `yy_test`, un modèle GAM est ajusté sur les années d'entraînement, puis le facteur de charge est prédit sur l'année `yy_test`.
- L'année 2022 est exclue si elle n'est pas l'année de test.
- Les prévisions sont agrégées dans un tableau `df_prev` les unes à la suite des autres et la RMSE (erreur quadratique moyenne) est calculée sur toutes les prédictions (Listing 6.1).

Listing 6.1 – Calcul du RMSE entre les colonnes `FC` et `prev`

```
df_prev[, .(rmse = rmse(FC, prev))]
          rmse
         <num>
1: 0.04707577
```

Analyse des résidus

```
ggplot(df_prev) +
  geom_point(aes(x = ff100_1, y = prev - FC), alpha = 0.5) +
  facet_wrap(~year(date_cible)) +
  labs(title = "Distribution des résidus")
```

Le code de RTE termine en proposant à travers un nuage de points (Figure 6.7) une visualisation des résidus du modèle en fonction de la vitesse du vent sur une zone.

Il serait aussi intéressant de visualiser la répartition de ces résidus par zone de vent à travers des histogrammes (Figure 6.8) pour vérifier s'ils sont bien centés autour de zéro, ou encore d'examiner l'évolution temporelle de ces résidus afin de s'assurer qu'ils sont bien dignes de cette appellation, c'est-à-dire, qu'ils ne contiennent plus de l'information non capturée par le modèle.

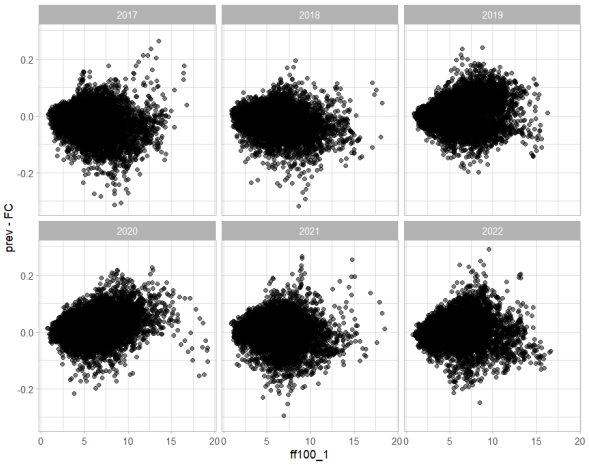


FIGURE 6.7 – Résidus du modèle

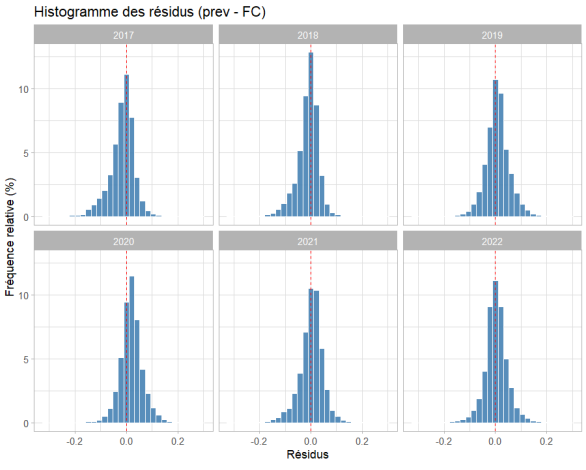


FIGURE 6.8 – Histogramme des résidus

```
#Histogrammes
ggplot(df_prev) +
  geom_histogram(aes(x = prev - FC), bins = 30, fill = "steelblue", color = "
      white", alpha = 0.7) +
  geom_vline(xintercept = 0, linetype = "dashed", color = "red")+
  facet_wrap(~ year(date_cible)) +
```

```
  labs(title = "Histogramme des résidus (prev - FC)",
       x = "Résidus", y = "Fréquence") +
  theme_minimal()
```

Archivage des résultats

```
saveRDS(df_prev %>% select(date_cible, prev),
  "data/output/df_prev_ref.RDS")
```

Après les visualisations et la validation du modèle, les prédictions par date sont sauvegardées en format RDS pour une référence ultérieure.

6.4 Complément : le format RDS

Pour la gestion et le stockage des données et des modèles dans le code de RTE, le format `RDS` (`R Data Serialization`) a été privilégié.

Il s'agit d'un format natif de `R` permettant de sauvegarder et de charger des objets `R` de manière efficace. Contrairement au format `RData`, qui peut contenir plusieurs objets, un fichier `RDS` ne stocke qu'un seul objet, ce qui le rend plus adapté au cadre d'une utilisation modulaire où il est nécessaire de charger et d'utiliser chaque objet de manière indépendante.

Quand utiliser RDS ?

- Pour stocker des modèles statistiques ou des données transformées sans charger tout un environnement.
- Lorsqu'on veut charger un objet sans écraser des variables existantes dans l'environnement.
- Pour partager un objet entre différentes sessions R.

Sauvegarder un objet en `RDS`

Pour enregistrer un objet `R` au format `RDS`, on utilise la fonction `saveRDS()`.

Syntaxe

```
saveRDS(objet, file = "nom_fichier.rds")
```

Exemple

```
# Exemple
notre_liste <- list(a = 1:10, b = matrix(1:9, nrow = 3), c = "texte")
# Sauvegarde en format RDS
saveRDS(notre_liste, file = "notre_liste.rds")
```

Charger un fichier RDS

Pour charger un fichier **RDS**, on utilise la fonction `readRDS()`.

Syntaxe

```
objet <- readRDS(file = "nom_fichier.rds")
```

Exemple

```
fichier <- readRDS("data_list.rds") # Chargement de l'objet sauvegardé
print(fichier) # Affichage
```

Remarque : Contrairement à la fonction `load()` (qui charge directement un fichier `.RData` dans l'environnement `R`), `readRDS()` ne restaure pas l'objet directement mais le retourne. Il est donc nécessaire d'assigner le résultat à une variable.

Différence entre `RDS` et `RData`

TABLEAU 6.1 – Comparaison entre RDS et RData

Caractéristique	**RDS (saveRDS/readRDS)**	**RData (save/load)**
Nombre d'objets	Un seul objet par fichier	Plusieurs objets
Chargement	L'objet doit être assigné	Charge les objets directement dans l'environnement
Usage recommandé	Stockage et manipulation modulaire	Sauvegarde de plusieurs objets

Chapitre 7

Amélioration des performances

Le modèle de RTE présenté dans le chapitre précédent commet une erreur quadratique moyenne (RMSE) de 0,047. L'objectif de ce dernier chapitre est de proposer quelques approches visant à réduire cette erreur. La section 7.1 présente quelques premières tentatives infructueuses tandis que les deux sections suivantes proposent des améliorations effectives ainsi que le code `R` qui a servi à les mettre en oeuvre.

7.1 Premières tentatives d'amélioration

7.1.1 Modèle à cinq zones

Une première approche est d'essayer de voir l'impact du nombre de zones climatiques sur la qualité des prévisions. Un découpage préeexistant est consultable en ligne sur le site de Météo France (MÉTÉO FRANCE 2021) et montrée à la Figure 7.1.

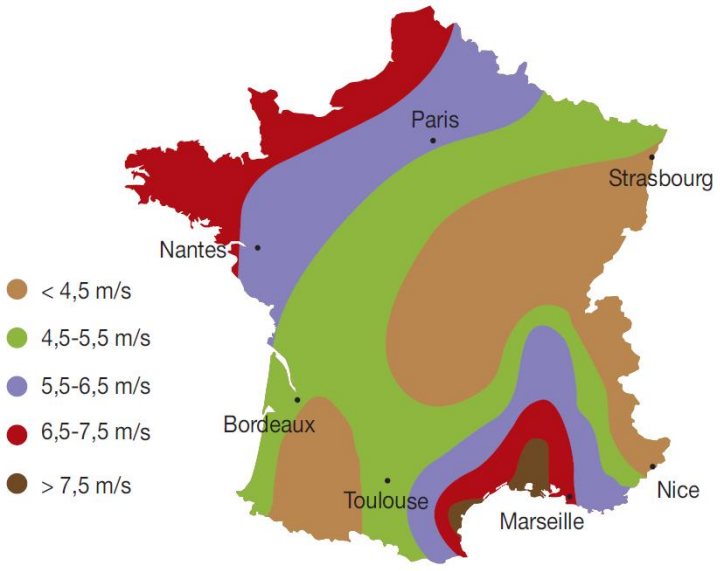


FIGURE 7.1 – Zones de vent à 50 m d'altitude

Il s'agit d'une classification des zones selon la vitesse du vent à 50 mètres d'altitude.
Or, nous disposons plutôt de la vitesse à 100 mètres. Pour cela, on peut se référer à l'article (CHEMEUROPE CONTRIBUTORS 2020) qui donne une estimation de la vitesse u_x du vent à une altitude z_x la connaissant à une certaine altitude z_r :

$$u_x = u_r \left(\frac{z_x}{z_r} \right)^{\alpha}$$

où α est un paramètre dépendant de la stabilité atmosphérique.

Dans les conditions standard, on utilise généralement : $\alpha \approx \frac{1}{7} \approx 0{,}143$

En appliquant cette formule sur la colonne `ff100` de `df_meteo_zone` et en reprenant l'agrégation par zone des données de production comme dans le pipeline du bloc de code 6.1, nous obtenons le dataframe du Listing 7.1, avec notamment la colonne `ff50` de la vitesse du vent à 50 mètres d'altitude. Le fichier `df_input` sur lequel se base la modélisation GAM dans la partie 6.3 du code de RTE est reconstruit et ressemble finalement à celui du Listing 7.2.

Listing 7.1 – Vitesses reconstruites à 50 m pour chaque zone géographique

```
# Affichage des premières lignes de df_meteo_zone
> print(head(df_meteo_zone))
            date_cible ZCnew     ff50
                <POSc> <num>    <num>
1: 2017-01-01 00:00:00     5 8.438999
2: 2017-01-01 00:00:00     4 6.938337
3: 2017-01-01 00:00:00     3 6.025828
4: 2017-01-01 00:00:00     2 4.978374
```

Listing 7.2 – Table d'entrée pour les prédictions : vitesses à 50 m par zone

```
# Affichage des premières lignes de df_input
> head(df_input)
Key: <date_cible>
            date_cible       pi eolien          FC   ff50_5   ff50_4   ff50_3
                ff50_2   ff50_1
                <POSc>    <num>  <num>       <num>    <num>    <num>    <num>
                       <num>    <num>
1: 2017-01-01 00:00:00 6657.806   331.0 0.04971608 8.438999 6.938337 6.025828
    4.978374 2.332300
2: 2017-01-01 01:00:00 6657.806   363.5 0.05459757 8.508322 6.947371 6.014358
    4.989444 2.352779
3: 2017-01-01 02:00:00 6657.806   347.0 0.05211927 8.493902 6.942245 5.992140
    4.962379 2.349422
4: 2017-01-01 03:00:00 6657.806   358.5 0.05384657 8.501388 6.973097 5.993196
    4.982221 2.325246
```

On refait, à partir de cette nouvelle décomposition, la modélisation GAM faite dans le code de RTE et on obtient les résultats ci-après :

Listing 7.3 – Résumé du modèle GAM à cinq covariables (`ff50_1` à `ff50_5`)

```
summary(modgam_5z)
Family: gaussian
Link function: identity
Formula:
FC ~ s(ff50_1) + s(ff50_2) + s(ff50_3) + s(ff50_4) + s(ff50_5)
Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.48169    0.00429   112.3   <2e-16 ***
Approximate significance of smooth terms:
            edf Ref.df      F p-value
s(ff50_1) 6.449  7.463  250.6  <2e-16 ***
s(ff50_2) 6.860  7.798 1335.0  <2e-16 ***
s(ff50_3) 7.611  8.406 1597.6  <2e-16 ***
s(ff50_4) 7.924  8.615 1583.9  <2e-16 ***
s(ff50_5) 8.702  8.954 2423.0  <2e-16 ***
```

```
R-sq.(adj) =  0.665   Deviance explained = 66.5%
fREML = -23848  Scale est. = 0.023078  n = 51437
```

Listing 7.4 – RMSE du modèle GAM à 5 zones

```
df_prev[,.(rmse = rmse(FC, prev))]
        rmse
       <num>
1: 0.1532901
```

On s'aperçoit donc que ce modèle n'améliore pas l'erreur quadratique moyenne, puisque la valeur 0,15 reste supérieure à 0,047.

7.1.2 Transformation des prédicteurs

De par nos recherches, nous savons maintenant que la production éolienne est proportionnelle au cube de la vitesse du vent. Donc, une approche convenable serait plutôt d'expliquer le facteur de charge à partir du cube de la vitesse du vent sur les 5 zones. Cette fois-ci, dans la modélisation GAM, nous choisissons plutôt la fonction *identité* pour fonction de lien car il s'agit maintenant d'un lien physique.

Et ce modèle fait mieux que ceux que celui déployé précédemment, sans pour autant surpasser la performance de celui de RTE car la valeur 0,047 de référence reste en dessous de la valeur 0,102 obtenue (Listings 7.5 et 7.6).

Listing 7.5 – Résumé du modèle avec transformation cubique des prédicteurs

```
summary(modgam_10z)
Family: gaussian
Link function: identity
Formula:
FC ~ s(ff100_1^3) + s(ff100_2^3) + s(ff100_3^3) + s(ff100_4^3) +
    s(ff100_5^3) + s(ff100_6^3) + s(ff100_7^3) + s(ff100_8^3) +
    s(ff100_9^3) + s(ff100_10^3)
Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.61816    0.00107   577.9   <2e-16 ***
Approximate significance of smooth terms:
              edf Ref.df       F p-value
s(ff100_1)  8.643  8.937 1915.22  <2e-16 ***
s(ff100_2)  8.762  8.972 6612.81  <2e-16 ***
s(ff100_3)  7.013  7.869   58.92  <2e-16 ***
s(ff100_4)  8.869  8.990   61.49  <2e-16 ***
s(ff100_5)  8.137  8.692  260.30  <2e-16 ***
s(ff100_6)  7.592  8.329   21.92  <2e-16 ***
s(ff100_7)  7.960  8.581   31.98  <2e-16 ***
s(ff100_8)  8.287  8.792  700.19  <2e-16 ***
s(ff100_9)  7.658  8.408  584.06  <2e-16 ***
s(ff100_10) 6.390  7.387   30.07  <2e-16 ***
R-sq.(adj) =  0.967   Deviance explained = 96.7%
fREML = -87318  Scale est. = 0.0020592  n = 52298
```

Listing 7.6 – RMSE du modèle GAM à 10 covariables transformées

```
df_prev[,.(rmse = rmse(FC, prev))]
         rmse
        <num>
1: 0.04707577
```

7.2 Amélioration par traitement de la multicolinéarité

7.2.1 Le constat

Constat sur l'inflation de la variance

Le travail fait jusqu'ici n'a pas tenu compte de la possible dépendance spatiale entre les covariables. Pourtant, dans un contexte de modélisation comme celui-ci, il est peu réaliste de supposer que les vitesses du vent sont indépendantes entre zones voisines. Cette corrélation spatiale se traduit souvent, dans les données, par une forte redondance linéaire entre certaines variables explicatives.

Cette configuration où il existe une forte corrélation linéaire entre deux ou plusieurs variables explicatives, connue sous le nom de *multicolinéarité*, peut sérieusement nuire à l'interprétabilité et à l'estimation fiable des effets individuels de chaque covariable. Elle rend les coefficients instables et tend à gonfler artificiellement les variances des estimateurs.

Pour diagnostiquer ce problème, on introduit un indicateur classique : le VIF (**Variance Inflation Factor**). Le VIF permet de quantifier la redondance d'une variable par rapport à l'ensemble des autres. Un VIF élevé (généralement supérieur à 10) signale une covariable fortement corrélée à une ou plusieurs autres, et donc susceptible d'introduire de la multicolinéarité.

```
# Ajout et Standardisation de la variable temps (valeurs trop grandes)
df_input[, time_since_beginning := df_fr$time_since_beginning]
df_input[, time_std := scale(time_since_beginning)]

library(car) # Calcul du VIF
vif(lm(FC ~ ff100_1 + ff100_2 + ff100_3 + ff100_4 + ff100_5 + ff100_6 +
         ff100_7 + ff100_8 + ff100_9 + ff100_10 + time_std, data = df_input))
```

Listing 7.7 – Résultats du calcul des VIF

```
vif(lm(FC ~ ff100_1 + ff100_2 + ff100_3 + ff100_4 + ff100_5 + ff100_6 +
         ff100_7 + ff100_8 + ff100_9 + ff100_10 + time_std, data = df_input))

  ff100_1   ff100_2   ff100_3   ff100_4   ff100_5   ff100_6   ff100_7   ff100_8
        ff100_9  ff100_10  time_std
> 8.125736  5.979385  7.091473 10.998787  4.698589  3.632378  1.965175
    7.049576  2.976454  2.714259  1.009080
```

On obtient des valeurs généralement élevées du VIF, notamment pour la variable `ff100_4` (VIF dépassant le seuil critique de 10) qui est donc fortement corrélée à d'autres covariables.

Matrice de corrélation linéaire

```
cor_matrix <- cor(df_input[, c("ff100_1","ff100_2","ff100_3", "ff100_4","ff100_
    5","ff100_6", "ff100_7", "ff100_8","ff100_9", "ff100_10", "time_std")])
cor_heatmap <- reshape2::melt(cor_matrix)
ggplot(cor_heatmap, aes(Var1, Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_viridis(option = "plasma", limits = c(-1, 1)) +
  theme_minimal(base_size = 12) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
    panel.grid = element_blank()) +
  labs(title = "Matrice de Corrélation linéaire",
    x = NULL, y = NULL,fill = "Corrélation")
```

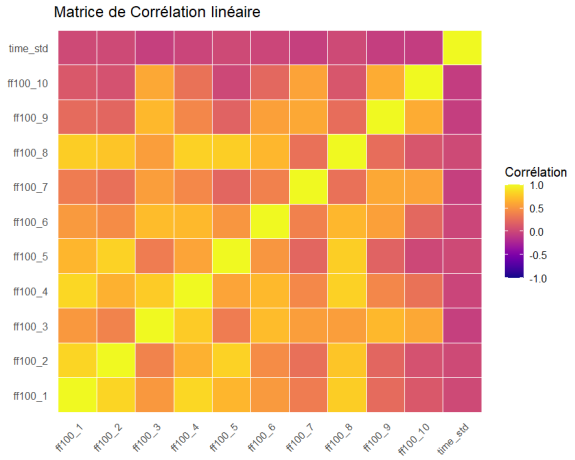


FIGURE 7.2 – Matrice de corrélation linéaire

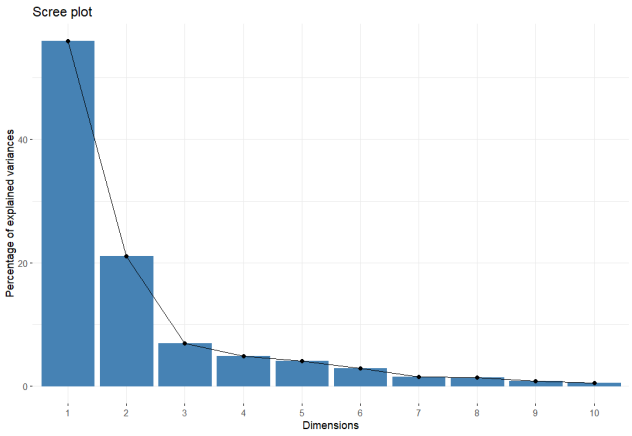


FIGURE 7.3 – Scree plot de l'ACP

La visualisation de la matrice (Figure 7.2) montre une très forte corrélation linéaire (de plus de 80%) entre `ff100_1` et `ff100_2`; `ff100_1` et `ff100_4`; `ff100_2` et `ff100_5`; ...

Il faudra donc traiter cette multicolinéarité pour être sûr d'avoir un modèle rigoureux et stable.

7.2.2 Décorrélation par analyse en composantes principales

Les effets dûs à la multicolinéarité peuvent être atténués en appliquant une analyse en composantes principales (ACP). Cette méthode permet de transformer les variables initialement corrélées en un nouvel ensemble de variables orthogonales (les composantes principales) maximisant la quantité d'information contenue dans les données.

```
library(FactoMineR)
library(factoextra)
# Normalisation des variables
pca_data <- df_input[, .(ff100_1, ff100_2, ff100_3, ff100_4, ff100_5, ff100_6,
    ff100_7, ff100_8, ff100_9, ff100_10)]
pca_data <- as.data.frame(scale(pca_data))
# ACP
pca_result <- PCA(pca_data, scale.unit = TRUE, graph = FALSE)
# Pourcentages de variance expliquée (cumulée)
cumsum(pca_result$eig[, 2])
# Visualisation des taux de variance expliquée
fviz_eig(pca_result)
```

Listing 7.8 – Taux cumulés de variance expliquée (ACP)

```
cumsum(pca_result$eig[, 2])
   comp 1    comp 2    comp 3    comp 4    comp 5    comp 6    comp 7    comp 8
          comp 9   comp 10
> 55.93725  76.96682  83.88728  88.72029  92.79804  95.65066  97.19969
    98.65897  99.49507 100.00000
```

```
# Modèle GAM basé sur les premières composantes principales
## Sélection des composantes (>90% de variance expliquée)
pca_scores <- as.data.frame(pca_result$ind$coord[, 1:5])
# Ajout des scores PCA au dataframe de la modélisation
df_input <- cbind(df_input, pca_scores)
modgam_pca_formula <- "FC ~ s(Dim.1) + s(Dim.2) + s(Dim.3) + s(Dim.4)+s(Dim.5)
    + s(time_std)"
modgam_pca <- bam(as.formula(modgam_pca_formula), data = df_input)
# Validation croisée
```

```
df_prev_pca <- map(unique(year(df_input$date_cible)), function(yy_test) {
  df_input[, train_test := fifelse(year(date_cible) == yy_test, "test", "train"
      )]
  if (yy_test != 2022) {df_input[year(date_cible) == 2022, train_test := "out"
      ]}
  modgam_pca <- bam(as.formula(modgam_pca_formula), data = df_input[train_test
      == "train"], discrete = TRUE)
  df_input[train_test == "test"] %>% mutate(prev = predict(modgam_pca, newdata
      = .))
}) %>% rbindlist()
# Calcul du RMSE
df_prev_pca[, .(rmse = rmse(FC, prev))]
```

Listing 7.9 – Calcul du RMSE pour le modèle basé sur l'ACP

```
df_prev_pca[, .(rmse = rmse(FC, prev))]
         rmse
        <num>
1: 0.07552623
```

On s'aperçoit que ce modèle basé sur les composantes principales issues de l'ACP, même s'il fait un peu mieux que les précédents, ne surpasse toujours pas la performance du modèle proposé par RTE, vu que la valeur 0,047 de référence est en-dessous du RMSE actuel de 0,0755.

7.2.3 Amélioration par régularisation (Ridge, Lasso, Elastic-Net)

Afin d'améliorer davantage la capacité prédictive du modèle tout en contrôlant sa complexité, nous explorons désormais des approches basées sur la régularisation, en particulier les méthodes Ridge, Lasso et Elastic-Net, qui permettent de limiter le surapprentissage et de gérer la multicolinéarité.

```
library(glmnet)
years <- unique(year(df_input$date_cible))
results <- list()
for (yy_test in years) {
  df_input[, train_test := fcase(
    year(date_cible) == yy_test, "test",
    year(date_cible) == 2022 & yy_test != 2022, "out", default = "train")]
  features <- setdiff(names(df_input), c("date_cible", "FC", "train_test"))

  x_train <- as.matrix(df_input[train_test == "train", ..features])
  y_train <- df_input[train_test == "train", FC]

  x_test <- as.matrix(df_input[train_test == "test", ..features])
  y_test <- df_input[train_test == "test", FC]

  models <- list(ridge = cv.glmnet(x_train, y_train, alpha = 0),      # Ridge
    lasso = cv.glmnet(x_train, y_train, alpha = 1),      # Lasso
    elastic = cv.glmnet(x_train, y_train, alpha = 0.5)  # Elastic Net)
  # Prédictions des modèles
  preds <- data.table(date_cible = df_input[train_test == "test", date_cible],
      FC_true = y_test,
    ridge = predict(models$ridge, newx = x_test, s = "lambda.min")[, 1],
    lasso = predict(models$lasso, newx = x_test, s = "lambda.min")[, 1],
    elastic = predict(models$elastic, newx = x_test, s = "lambda.min")[, 1]
  )
  results[[as.character(yy_test)]] <- preds
}
# Résulats et calcul des RMSE
final <- rbindlist(results, fill = TRUE)
erreurs <- final[, .(rmse_ridge = rmse(FC_true, ridge), rmse_lasso = rmse(FC_
    true, lasso),
```

```
rmse_elastic = rmse(FC_true, elastic)), by = year(date_cible)]
# Affichage des RMSE
print(c(mean(erreurs$rmse_elastic),mean(erreurs$rmse_lasso),mean(erreurs$rmse_
    ridge)))
```

Listing 7.10 – RMSE des modèles de régularisation

```
print(c(mean(erreurs$rmse_elastic), mean(erreurs$rmse_lasso), mean(erreurs$rmse
    _ridge)))
[1] 0.04066285 0.04087566 0.04532456
```

On obtient, pour les modèles de régularisation de type Elastic-Net et de type Lasso, une amélioration (quoique modeste) du RMSE par rapport au modèle proposé par RTE.

7.3 Amélioration nette par redécoupage des zones

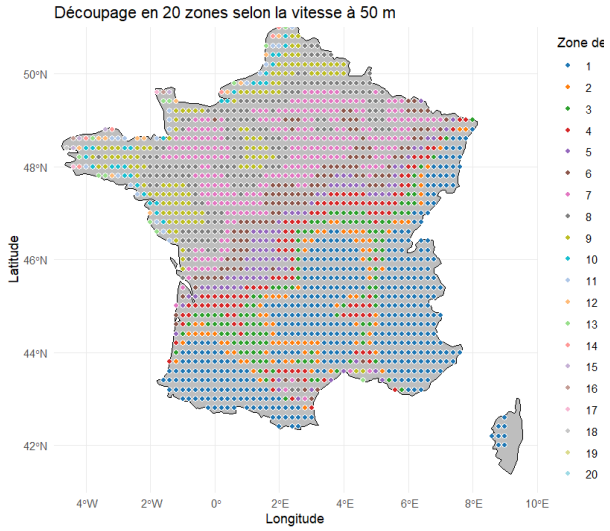


FIGURE 7.4 – Nouveau découpage

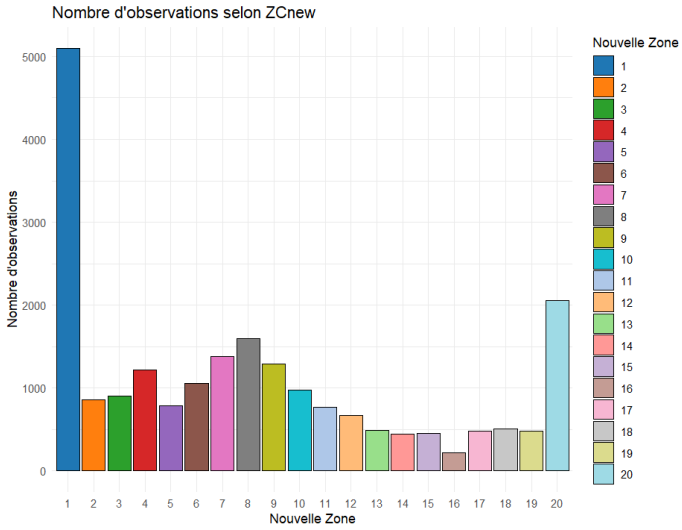


FIGURE 7.5 – Nombre de sites par zone

Reprenons la carte de la section 7.1.1 et redéfinissons le découpage géographique de manière plus fine (20 zones) comme sur la Figure 7.4. Extrapolons ensuite la vitesse du vent à 50 m d'altitude comme fait précédemment avec le modèle à 5 zones et reprenons la modélisation du facteur de charge comme dans le code de RTE. Le code est donné ci-après.

```
# Traitement des fichiers NetCDF
df_meteo_zone <- map(list.files("data/wind_era5", "nc", full.names = TRUE),
    function(ff){
  # Ouverture du fichier NetCDF
  fid <- ncdf4::nc_open(ff)
  file_dump <- fid$dim
  # Extraction de la date d'origine
  origin_date <- strsplit(split = "␣since␣", x = file_dump$time$units)[[1]][2]
  # Définition des index
  indexes_names <- list(longitude = as.character(round(file_dump$longitude$vals
      , 2)),
    latitude = as.character(round(file_dump$latitude$vals, 2)),
    time = as.character(file_dump$time$vals))
  # Extraction des données
  array_meteo0 <- sqrt(ncdf4::ncvar_get(nc = fid, varid = "u100")^2 + ncdf4::
      ncvar_get(nc = fid, varid = "v100")^2)
  dimnames(array_meteo0) <- indexes_names
```

```
  # Conversion en data.table
  df_meteo0 <- as.data.frame.table(array_meteo0, stringsAsFactors = FALSE,
      responseName = 'ff100') %>% setDT()
  df_meteo0[, date_cible := ymd_hms(origin_date) + dhours(as.numeric(time))]
  df_meteo0$time <- NULL
  # Fermeture du fichier NetCDF
  ncdf4::nc_close(fid)
  # Calcul de ff50 à partir de ff100
  df_meteo0[, ff50 := as.numeric(ff100 * (0.5^0.2))]
  # Application du rezonage (ZCnew) en fonction de ff50
  df_meteo0[, ZCnew := fcase(
    ff50 <= 4.5, 1,
    ff50 > 4.5 & ff50 <= 4.7, 2,
    ff50 > 4.7 & ff50 <= 4.9, 3,
    #............ Ainsi de suite
  ff50> 7.7 & ff50 <= 7.9, 18,
  ff50> 7.9 & ff50 <= 8.1, 19,
  ff50> 8.1 , 20)]
  # Agrégation zonale
  df_meteo_zone_ff <- df_meteo0[
    , .(ff50 = mean(ff50, na.rm = TRUE)),
    by = .(date_cible, ZCnew)]
  df_meteo_zone_ff
}) %>% rbindlist(use.names = TRUE)
# Sauvegarde sous format wide
saveRDS(df_meteo_zone %>%
          pivot_wider(id_cols = date_cible, names_from = ZCnew, values_from =
              ff50, names_prefix = "ff50_"),  "data/input_model/df_meteo_zone_
              wide.RDS")

# Préparation des données pour la modélisation
df_fr <- merge(df_meteo_zone[, .(ff50_fr = mean(ff50, na.rm = TRUE)), by = .(
    date_cible)], df_prod, by = "date_cible")
df_fr[, time_since_beginning := as.numeric(difftime(date_cible, ymd_hms("
    2017-01-01 00:00:00")))]

# Modèle GAM de eolien en fonction de ff50
mod_gam_noFC_linklog <- bam(
  eolien ~ s(ff50_fr, k = 10) + s(time_since_beginning, k = 7),
  data = df_fr, family = gaussian(link = "log"), discrete = TRUE)

# Estimation de pi (Puissance installée)
df_fr[,pi := exp(predict(mod_gam_noFC_linklog, newdata = df_fr %>% mutate(ff50_
    fr = 12*(0.5^0.2))))]

# Sauvegarde
saveRDS(df_fr[,.(date_cible, pi, eolien_pred)], "ata/input_model/df_pi.RDS")

# Chargement des fichiers pour la modélisation
df_input <- merge(readRDS("data/input_model/df_pi.RDS"), readRDS("data/input_
    model/df_prod.RDS"), by = "date_cible") %>%
  mutate(FC=eolien/pi) %>%
  merge(readRDS("data/input_model/df_meteo_zone_wide.RDS"), by = "date_cible")

# Modèle GAM à 20 zones
modgam_20z_formula <- "FC ~ s(ff50_1) + s(ff50_2) + s(ff50_3) + s(ff50_4)+s(
    ff50_5)+s(ff50_6)+s(ff50_7)+s(ff50_8)+s(ff50_9)+s(ff50_10)+s(ff50_11)+s(ff50
    _12)+s(ff50_13)+s(ff50_14)+s(ff50_15)+s(ff50_16)+s(ff50_17)+s(ff50_18)+s(
    ff50_19)+s(ff50_20)"

# GAM avec validation croisée par blocs annuels
df_prev <- map(unique(year(df_input$date_cible)), function(yy_test){
  df_input[,train_test := fifelse(year(date_cible) == yy_test, "test", "train")
```

```
    ]
  if(yy_test != 2022){df_input[year(date_cible) == 2022, train_test := "out"]}
  modgam_20z <- bam(as.formula(modgam_20z_formula), data=df_input[train_test ==
      "train"], discrete = TRUE)
  df_input[train_test == "test"] %>% mutate(prev = predict(modgam_20z, newdata
      = .))
}) %>% rbindlist()
## Calcul du RMSE
df_prev[,.(rmse = rmse(FC, prev))]
```

Le résultat final de ce modèle à 20 zones est donné au Listing 7.11.

Listing 7.11 – RMSE du modèle à 20 zones

```
df_prev[,.(rmse = rmse(FC, prev))]
          rmse
         <num>
1: 0.000425589
```

Et on obtient pour ce dernier modèle une diminution considérable et très surprenante de l'erreur de prédiction moyenne qui vaut finalement 0.0004, soit **100 fois mieux que le modèle de RTE!**

Cinquième partie

Conclusion

Nous y sommes !

Ce projet a constitué pour nous une première immersion concrète dans la résolution d'une problématique relevant d'un contexte industriel. Il nous a amenés à savoir manipuler des données massives de natures différentes (numériques, cartographiques, ...), des fichiers à divers extensions (`.csv`, `.xls`, `.nc`, `.RDS`, ...), des packages `R` variés pour diverses tâches, à comprendre du code `R` écrit par une tierce personne, à déployer un modèle statistique rigoureux pour répondre à la problématique posée et enfin à produire un rapport scientifique bien documenté et bien référencé avec des illustrations pertinentes.

Plutôt que de nous limiter à l'approche adoptée par RTE pour la prévision du rendement éolien, nous nous sommes évertués à l'améliorer tant au niveau du découpage zonal que de la modélisation statistique. Après quelques tentatives infructueuses, nous sommes parvenus à un résultat très intéressant.

Cependant, si nous avions disposé de davantage de temps, nous aurions pu approfondir plusieurs autres approches. Par exemple, différentes stratégies pour un redécoupage géographique (avec des zones connexes) plus fin, afin de mieux capturer les spécificités locales des données du vent, ce qui aurait potentiellement permis d'affiner davantage la précision des prédictions. Nous aurions aimé tester d'autres modèles d'apprentissage automatique (comme les forêts aléatoires) ou d'apprentissage profond (comme les réseaux de neurones récurrents qui sont parfaitement adaptés à la prévision de données à forte dépendance temporelle ou spatiale). Nous aurions pu tenter aussi une renormalisation déterministe de la production éolienne afin de prédire le facteur de charge à partir d'une estimation plus fiable de la puissance installée, ne provenant pas elle-même d'un modèle statistique. Enfin, l'intégration de données supplémentaires d'autres covariables, telles que la température, l'humidité, l'altitude des installations, aurait enrichi notre analyse et offert une perspective plus large sur la dynamique de la production éolienne en France.

Bibliographie

ADJENGUE, Luc (2021). *Test de Kruskal-Wallis*. URL : http://www.presses-polytechnique.ca/redirect.html.

CHEMEUROPE CONTRIBUTORS (2020). *Wind profile power law*. URL : https://www.chemeurope.com/en/encyclopedia/Wind_profile_power_law.html.

CRAN CONTRIBUTORS (2024). *ncdf4 : Interface to Unidata netCDF (Version 4 or Earlier) Format Data Files*. URL : https://cran.r-project.org/web/packages/ncdf4/index.html.

DATACAMP (2023). *Un guide sur le VIF et le traitement de la multicolinéarité*. URL : https://www.datacamp.com/fr/tutorial/variance-inflation-factor.

EOLISE (2022a). *Comment fonctionne une éolienne ?* URL : https://eolise.fr/vos-questions/comprendre-leolien/le-fonctionnement-de-lenergie-eolienne/comment-fonctionne-une-eolienne/.

– (2022b). *Qu'est-ce que la courbe de puissance d'une éolienne ?* URL : https://eolise.fr/vos-questions/comprendre-leolien/le-fonctionnement-de-lenergie-eolienne/quest-ce-que-la-courbe-de-puissance-dune-eolienne/.

– (2023). *Le facteur de charge, c'est quoi au juste ?* URL : https://eolise.fr/vos-questions/comprendre-leolien/le-fonctionnement-de-lenergie-eolienne/le-facteur-de-charge-cest-quoi-au-juste/.

HASTIE, Trevor, Robert TIBSHIRANI et Jerome FRIEDMAN (2013). *An Introduction to Statistical Learning*. Springer. URL : https://www.statlearning.com/.

INSTITUT MATHÉMATIQUE DE TOULOUSE (2014). *Tests non paramétriques*. URL : https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-inf-np.pdf.

MÉTÉO FRANCE (2021). *Carte des zones de vent à 50m d'altitude*. URL : http://hqe.guidenr.fr/cible-13-hqe/ventilation-naturelle-vent.php.

R DOCUMENTATION (2020). *RDS Format*. URL : https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/readRDS.

REARTHSYSSCI (2020). *Le package ncdf4*. URL : https://pjbartlein.github.io/REarthSysSci/netCDF.html.

RTE (2024). *Évolution du parc éolien et du facteur de charge en France*. URL : https://analysesetdonnees.rte-france.com/production/eolien.

WOOD, Simon N. (2017). *Generalized Additive Models*. Second. Texts in Statistical Science. Chapman et Hall/CRC. DOI : 10.1201/9781315370279. URL : https://www.taylorfrancis.com/books/mono/10.1201/9781315370279/generalized-additive-models-simon-wood.

ZOU, Hui et Trevor HASTIE (2005). « Regularization and variable selection via the elastic net ». In : *Journal of the Royal Statistical Society : Series B (Statistical Methodology)* 67.2, p. 301-320. URL : https://hastie.su.domains/Papers/B67.2%20%282005%29%20301-320%20Zou%20%26%20Hastie.pdf.