

Introduction aux méthodes quantitatives en sciences sociales avec R

Philippe Apparicio et Jérémy Gelb

2020-10-19

Table des matières

Préface	9
Comment lire ce livre	9
Structure du livre	9
Remerciements	9
1 Régressions linéaires généralisées (GLM)	11
1.1 Qu'est qu'un modèle GLM?	11
1.1.1 Formulation d'un GLM	11
1.1.2 Autres distributions et rôle de la fonction de Lien	12
1.1.3 Conditions d'application	15
1.1.4 Résidus et déviance	15
1.1.5 Vérifier l'ajustement	16
1.1.6 Comparer deux modèles GLM	21
1.2 Les principaux modèles GLM	22
1.2.1 Les modèles GLM pour des variables qualitatives	22
1.2.2 Les modèles GLM pour des variables de comptage	68
1.2.3 Les modèles GLM pour des variables continues	103
2 Régressions à effets mixtes (GLMM) et régression multiniveaux	113
2.1 Principes de base des GLMM	113
2.2 Application à une variable continue	113
2.3 Régressions multiniveaux	113
3 Modèles généralisés additifs	115
3.1 Principes de base des GAM	115
3.2 Extensions des GAM	115
3.2.1 GAMM	115
3.2.2 GAMMAR	115
Première partie Analyses exploratoires multivariées	117
4 Méthodes factorielles	119
4.1 Un petit historique	119
4.2 Analyses de composantes principales (ACP)	119
4.3 Analyses factorielles de correspondances (AFC)	119
4.4 Analyses factorielles de correspondances multiples (AFM)	119
4.5 Analyses factorielles de correspondances mixte	119

5 Méthodes de classification non-supervisées	121
5.1 Un aperçu sur la multitude des méthodes de classifications	121
5.2 Combinaisons méthodes factorielles et méthodes de classifications	121
5.3 Classification ascendantes hiérachiques	121
5.4 Nuées dynamiques	121
5.4.1 k-means	121
5.4.2 k-median	121
5.4.3 Les extensions en logique floues : c-means, c-median	121
6 Conclusion	123
6.1 Synthèse des méthodes abordées	123
6.2 Autres méthodes non abordées	123
6.2.1 Régressions par quantile	123
6.2.2 Régressions par panel	123
6.2.3 Régressions par Tobit	123
6.2.4 Analyses de survie	123
6.2.5 Équations structurelles	123
6.3 Deux écoles de statistique inférielle : fréquentiste et bayésienne	123
6.4 Éthique en méthodes quantitatives	123
7 Annexes	125
7.1 Tableau des valeurs critiques de χ^2	125
7.2 Tableau des valeurs critiques de F	125
7.3 Tableau des valeurs critiques de t	126

Liste des tableaux

1.1	Principaux pseudo R^2	17
1.2	Exemple de matrice confusion	20
1.3	Exemple de matrice confusion multinomiale	21
1.4	Indicateurs de qualité de prédiction	21
1.5	Interprétation des valeurs du coefficient de Kappa	22
1.6	Carte d'identité du modèle logistique binomial	23
1.7	Variable indépendantes utilisées pour prédire le mode de transport le plus utilisé	25
1.8	Matrice de confusion pour le modèle binomial	34
1.9	Matrice de confusion pour le modèle binomial	35
1.10	Résultats du modèle binomial	37
1.11	Carte d'identité du modèle probit binomial	41
1.12	Carte d'identité du modèle logistique des cotes proportionnelles	42
1.13	Variable indépendantes utilisées pour prédire la catégorie de prix de logements Airbnb	44
1.14	Coefficients du modèle logistique des cotes proportionnelles	53
1.15	Carte d'identité du modèle probit binomial	56
1.16	Variable indépendantes utilisées dans le modèle logistique multinomial	58
1.17	Coefficients du modèle multinomial A VS B	67
1.18	Coefficients du modèle multinomial A VS C	67
1.19	Coefficients du modèle multinomial A VS D	68
1.20	Carte d'identité du modèle de Poisson	69
1.21	Variable indépendantes utilisées dans le modèle de Poisson	71
1.22	Résultats du modèle de quasi-Poisson	82
1.23	Carte d'identité du modèle Négatif Binomial	83
1.24	Résultats du modèle Négatif Binomial	91
1.25	Carte d'identité du modèle de Poisson avec excès fixe de zéros	91
1.26	Carte d'identité du modèle de Poisson avec excès ajusté de zéros	93
1.27	Résultats de la partie Poisson du modèle de Poisson avec excès de zéros ajusté	101
1.28	Résultats de la partie logistique du modèle de Poisson avec excès de zéros ajusté	102
1.29	Carte d'identité du modèle Gaussien	104
1.30	Résultats du modèle Gaussien	111
7.1	Distribution des valeurs critiques du khi2	126
7.2	Distribution des valeurs critiques de F avec p=0,05	127
7.3	Distribution des valeurs critiques de t	128

Table des figures

1	Cargo, le plus beau	10
1.1	Exemple de données issues d'une distribution de Bernoulli	13
1.2	Ajustement d'une droite de régression aux données issues d'une distribution de Bernoulli	14
1.3	Utilisation de la fonction de lien logistique	14
1.4	Distances de Cook pour le modèle binomial avec toutes les observations	26
1.5	Distances de Cook pour le modèle binomial sans les valeurs aberrantes	28
1.6	Distribution des résidus simulé pour le modèle binomial	29
1.7	Diagnostic des résidus simulés par le package DHARMA	30
1.8	Point d'équilibre entre sensibilité et spécificité	33
1.9	Rapports de cote pour les différents pays de l'UE	36
1.10	Rapports de cote pour les différents lieux de résidence	38
1.11	Impact de l'âge sur la probabilité d'utiliser le vélo comme moyen de déplacement pour son trajet le plus fréquent	40
1.12	Distribution des prix des logements Airbnb	43
1.13	Distances de Cook pour le modèle logistique des cotes proportionnelles	46
1.14	Diagnostic général des résidus simulés du modèle des cotes proportionnelles	47
1.15	Diagnostic variable par variable des résidus simulés du modèle des cotes proportionnelles	48
1.16	Diagnostic variable par variable des résidus simulés du modèle des cotes proportionnelles (arpès correction)	49
1.17	Prédiction de la probabilité d'appartenance aux trois catégories de prix en fonction de la densité de végétation	55
1.18	Distances de Cook pour le modèle logistique multinomial	60
1.19	Diagnositc général des résidus simulés pour le modèle multinomial	62
1.20	Diagnositc par variable des résidus simulés pour le modèle multinomial	63
1.21	Diagnositc général des résidus simulés pour le modèle multinomial (version 3)	64
1.22	Diagnositc général des résidus simulés pour le modèle multinomial (version 4)	65
1.23	Distribution originale du nombre d'accident par intersection	72
1.24	Distance de Cook pour le modèle de Poisson	73
1.25	Distance de Cook pour le modèle de Poisson après avoir retiré les valeurs aberrentes	74
1.26	Représentation de la sur-dispersion des données dans le modèle de Poisson	76
1.27	Comparaison de la distribution originale et des simulations pour le modèle de quasi-Poisson	78
1.28	Comparaison de la distribution originale et des simulations pour le modèle de Poisson	78
1.29	Analyse globale des résidus simulés pour le modèle de quasi-Poisson	79
1.30	Comparaison des résidus simulés et de chaque variable indépendante	80
1.31	Distances de Cook pour le modèle négatif binomial	84
1.32	Distances de Cook pour le modèle négatif binomial (après avoir retiré deux observations fortement influentes)	86
1.33	Diagnostic général des résidus simulés pour le modèle Négatif Binomial	87

1.34 Comparaison de la distribution originale et des simulations pour le modèle Négatif binomial	88
1.35 Représentation de la sur-dispersion des données dans le modèle de Poisson	89
1.36 Diagnostic général des résidus simulés du modèle de Poisson avec excès de zéros ajusté .	94
1.37 Diagnostic général des résidus simulés du modèle de Poisson avec excès de zéros ajusté (sans valeurs aberrantes)	97
1.38 Comparaison de la distribution originale et des simulations pour le modèle de Poisson avec excès de zéros ajusté	98
1.39 Processus de sélection d'un modèle pour une variable de comptage	103
1.40 Distances de Cook pour le modèle Gaussien	105
1.41 Distances de Cook pour le modèle Gaussien après suppression des observations influentes	107
1.42 Diagnostic général des résidus simulés pour le modèle Gaussien	108
1.43 Comparaison de la distribution originale de la variable et des simulations issues du modèle	108
1.44 Comparaison de la distribution originale de la variable et des simulations issues du modèle	110

Préface

Comment lire ce livre

Si vous googlez l'expression « comment lire un livre ? », vous trouverez une multitude de conseils et astuces. Pour ce livre, nous conseillons de le lire de gauche à droite et page par page. Plus sérieusement, il comprend plusieurs types de blocs de texte qui, on l'espère, faciliteront la lecture.



Bloc packages : habituellement localisé en début du chapitre, il comprend la liste des packages R utilisés pour un chapitre.



Bloc objectif : comprend une description des objectifs d'une section.



Bloc notes : comprend une information secondaire sur une notion, un élément, une idée abordée dans une section.



Bloc pour aller plus loin : peut comprendre des références ou des extensions d'une méthode statistique abordée dans une section.



Bloc astuce : décrit un élément qui vous facilera la vie : une propriété statistique, un *package*, une fonction, une syntaxe R.



Bloc attention : comprend une notion ou un élément important à bien maîtriser.

Structure du livre

À écrire plus tard.

Remerciements

Note au beau Cargo (chien Mira) qui nous supporte dans l'écriture du livre !



FIG. 1 : Cargo, le plus beau

Chapitre 1

Régressions linéaires généralisées (GLM)



Les modèles linéaires généralisés

Dans cette section nous présenterons les modèles linéaires généralisés plus communément appelés GLM (generalized linear model en anglais). Il s'agit d'une extension directe des modèles de régression linéaire (LM) par la méthode des moindres carrés ordinaire décrite dans le chapitre précédent. Pour aborder cette section sereinement, il est important d'avoir bien compris le concept de distribution présenté dans la section (section ??). A la fin de cette section, vous serez en mesure de :

- comprendre la distinction entre un modèle LM classique et un GLM
- identifier les composantes d'un GLM
- interpréter les résultats d'un GLM
- effectuer les diagnostic d'un GLM

1.1 Qu'est qu'un modèle GLM ?

Nous avons vu qu'une régression linéaire multiple (LM) ne peut être appliquée que si la variable dépendante analysée est continue et si elle est normalement distribuée une fois les variables indépendantes contrôlées. Il s'agit d'une limite très importante puisqu'elle ne peut être utilisée pour modéliser et prédire des variables binaires, multinomiales, de comptage, ordinaires ou plus simplement des données anormalement distribuées. Une seconde limite importante des LM est que l'influence des variables indépendantes sur la variable dépendante ne peut être que linéaire. L'augmentation d'une unité de X conduit à une augmentation (ou diminution) de β (coefficients de régression) unités de Y , ce qui n'est pas toujours représentatif des phénomènes étudiés. Afin de dépasser ces contraintes, [Nelder and Wedderburn \(1972\)](#) ont proposé une extension des modèles LM, soit les modèles linéaires généralisés (GLM).

1.1.1 Formulation d'un GLM

Puisqu'un modèle GLM est une extension des modèles LM, il est possible de traduire un modèle LM sous forme d'un GLM. Nous utilisons ce point de départ pour détailler la morphologie d'un GLM. Nous avons vu dans la section précédente qu'un modèle LM correspond à la formule suivante (notation matricielle) :

$$Y = \beta_0 + X\beta + \epsilon \quad (1.1)$$

Avec β_0 la constante (intercept) et β un vecteur de coefficients de régression pour les k variables indépendantes (X).

D'après cette formule, nous modélisons la variable Y avec une équation de régression linéaire et un terme d'erreur que l'on estime être normalement distribué. Nous pouvons reformuler ce simple LM sous forme d'un GLM avec l'écriture suivante :

$$\begin{aligned} Y &\sim \text{Normal}(\mu, \sigma) \\ g(\mu) &= \beta_0 + \beta X \\ g(x) &= x \end{aligned} \tag{1.2}$$

Pas de panique ! Cette écriture se lit comme suit : La variable Y est issue d'une distribution normale ($Y \sim \text{Normal}$) avec deux paramètres : μ (sa moyenne) et σ (son écart type). μ varie en fonction d'une équation de régression linéaire : $\beta_0 + \beta X$, transformée par une fonction de lien g (définie plus tard). Dans ce cas précis, la fonction de lien est appelée fonction identitaire puisqu'elle n'applique aucune transformation ($g(x) = x$). Vous noterez ici que le second paramètre de la distribution normale σ (paramètre de dispersion) est fixé et ne dépend donc pas des variables indépendantes à la différence de μ . Dans ce modèle spécifiquement, les paramètres à estimer sont : σ , β_0 , et β . Notez que dans la notation traditionnelle, la fonction de lien est appliquée au paramètre modélisé. Il est possible de renverser cette notation en utilisant la réciproque (g') de la fonction de lien (g) :

$$g(\mu) = \beta_0 + \beta X \iff \mu = g'(\beta_0 + \beta X) \text{ si } g'(g(x)) = x \tag{1.3}$$

Dans un modèle GLM, la distribution attendue de la variable Y est déclarée de façon explicite, ainsi que la façon dont nos variables indépendantes influencent cette distribution. Ici, c'est la moyenne (μ) de la distribution qui est modélisée, on s'intéresse donc au changement moyen de Y provoqué par les variables X .

Avec cet exemple, nous voyons les deux composantes supplémentaires d'un modèle GLM :

- La distribution supposée de la variable Y (ici la distribution normale)
- Une fonction de lien associant l'équation de régression formée par les variables indépendantes et un paramètre de la distribution retenue (ici la fonction identitaire et le paramètre μ).

Notez également que l'estimation des paramètres d'un modèle GLM (ici β_0 , βX et σ) ne se fait plus avec la méthode des moindres carrés vue pour les modèles LM. À la place, la méthode par maximum de vraisemblance (maximum likelihood) est le plus souvent utilisée, mais certains packages utilisent également la méthode des moments (method of moments). Dans les deux cas, ces méthodes nécessitent des échantillons plus grands que la méthode des moindres carrés.

1.1.2 Autres distributions et rôle de la fonction de Lien

À première vue, on pourrait se demander pourquoi rajouter ces deux éléments puisqu'ils ne font que complexifier le modèle. Prenons donc un exemple appliqué au cas d'une variable binaire pour souligner la capacité de généralisation des modèles GLM. Admettons que nous souhaitons modéliser / prédire la probabilité qu'un cycliste décède lors d'une collision avec un véhicule motorisé. Notre variable dépendante est donc binaire (0 = survie, 1 = décès), et nous souhaitons la prédire avec trois variables continues que sont : la vitesse de déplacement du cycliste ($x1$), la vitesse de déplacement du véhicule ($x2$) et la masse du véhicule ($x3$). Puisque Y n'est pas continue, il ne fait aucun sens d'assumer qu'elle est issue d'une distribution normale. Cependant, il est naturel de supposer qu'elle provient d'une distribution de Bernoulli (pour rappel, une distribution de Bernoulli permet de modéliser un phénomène ayant deux issues possibles comme un lancer de pièce de monnaie). Plus spécifiquement, nous pourrions formuler

l'hypothèse que nos trois variables x_1 , x_2 et x_3 influencent le paramètre p (la probabilité d'occurrence de l'évènement) d'une distribution de Bernoulli. Avec ces premières hypothèses, nous pouvons écrire le modèle suivant :

$$\begin{aligned} Y &\sim \text{Bernoulli}(p) \\ g(p) &= \beta_0 + \beta X \\ g(x) &= x \end{aligned} \tag{1.4}$$

Toutefois, le résultat n'est pas entièrement satisfaisant. En effet, p est une probabilité et, par nature, ce paramètre devrait être compris entre 0 et 1 (entre 0 et 100% de chance de décès, ni plus ni moins). L'équation de régression que nous utilisons actuellement peut produire des résultats compris en $+\infty$ et $-\infty$ pour p puisque rien ne contraint la somme $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$ à être comprise entre 0 et 1. Il est possible de visualiser le problème soulevé par cette situation avec les figures suivantes. Admettons que nous ayons observé une variable Y binaire et que nous savons qu'elle est influencée par une variable X , qui plus elle augmente, plus la chance que Y soit 1 augmente :

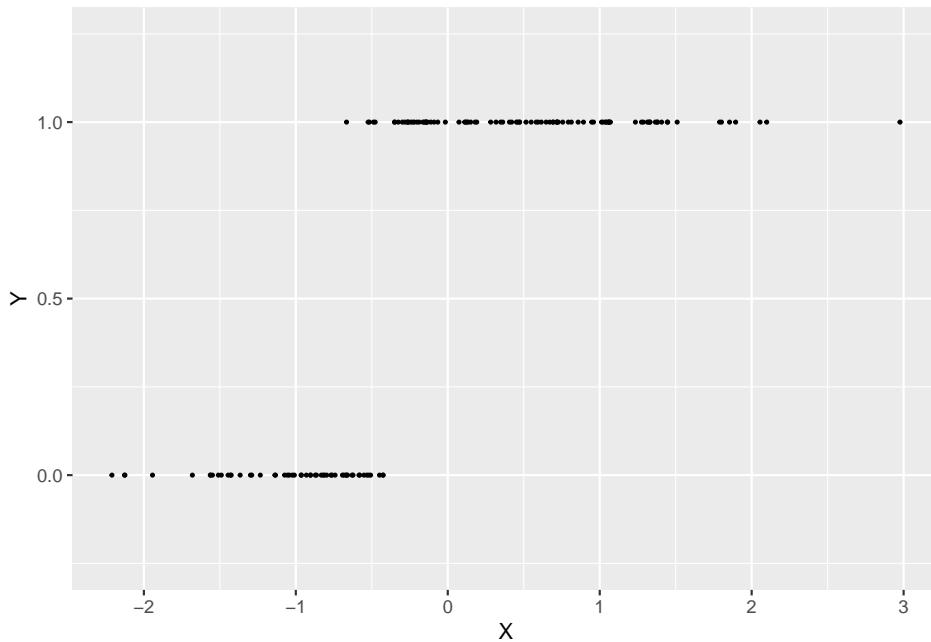


FIG. 1.1 : Exemple de données issues d'une distribution de Bernoulli

Si l'on utilise l'équation de régression actuelle, cela revient à trouver la droite la mieux ajustée passant dans ce nuage de points :

Ce modèle semble bien cerner l'influence positive de X sur Y , mais la droite est au final très éloignée de chaque point, indiquant un faible ajustement du modèle. De plus, la droite prédit des probabilités négatives lorsque X est inférieur à -2.5 et des probabilités supérieures à 1 quand X est supérieur à 1. Elle est donc loin de bien représenter les données.

C'est ici qu'intervient la fonction de lien. La fonction identitaire n'est pas satisfaisante, nous devons la remplacer par une fonction qui conditionnera la somme $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$ pour donner un résultat entre 0 et 1. Une candidate toute désignée est la fonction *sigmoidale*, plus souvent appelée la fonction *logistique* !

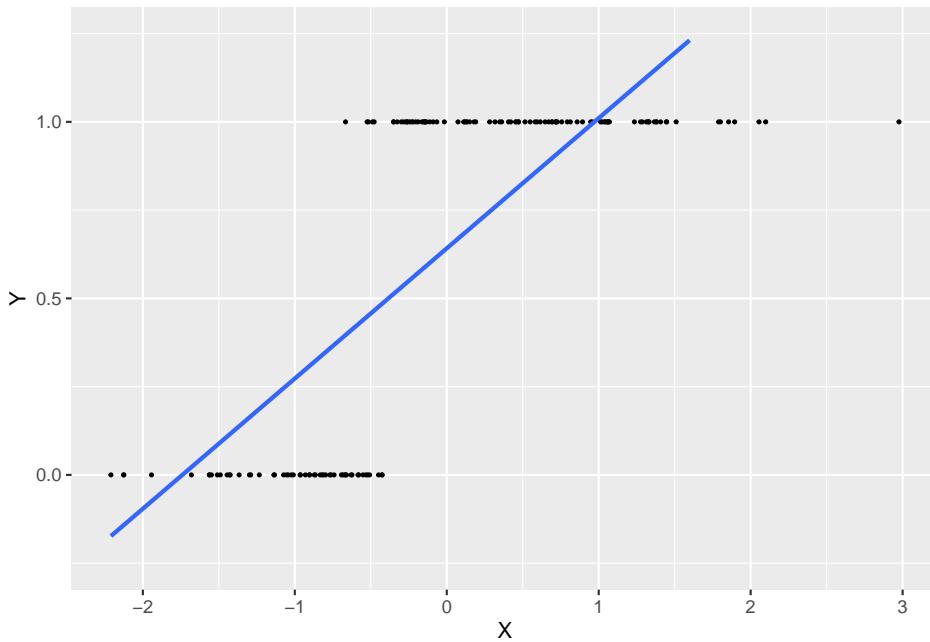


FIG. 1.2 : Ajustement d'une droite de régression aux données issues d'une distribution de Bernoulli

$$Y \sim \text{Bernoulli}(p)$$

$$S(p) = \beta_0 + \beta X$$

$$S(x) = \frac{e^x}{e^x + 1}$$

(1.5)

La fonction logistique prend la forme d'un S. Plus la valeur entrée dans la fonction est grande, plus le résultat produit par la fonction est proche de 1 et inversement. Si l'on reprend l'exemple précédent, on obtient le modèle suivant :

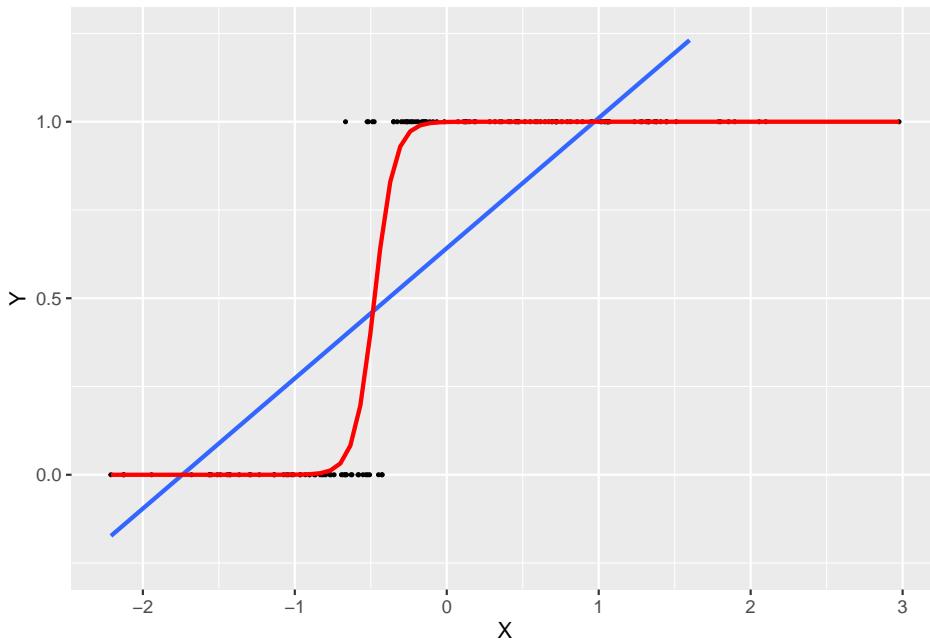


FIG. 1.3 : Utilisation de la fonction de lien logistique

Une fois cette fonction insérée dans le modèle, on constate qu'une augmentation de la somme $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$ conduit à une augmentation de la probabilité p et inversement, et que cet impact est non linéaire. Nous avons donc maintenant un GLM permettant de prédire la probabilité d'un décès lors d'un accident en combinant une distribution et une fonction de lien adéquates.

1.1.3 Conditions d'application

La famille des GLM englobe de (très) nombreux modèles du fait de la diversité de distributions existantes et des fonctions de liens utilisables. Cependant, certaines combinaisons sont plus souvent utilisées que d'autres. Nous présentons donc dans les prochaines sections les modèles GLM les plus communs. Les conditions d'applications varient d'un modèle à l'autre en fonction du choix de la distribution, il existe cependant quelques conditions d'application communes à tous ces modèles :

- L'indépendance des observations (et donc des erreurs).
- L'absence de valeurs aberrantes / fortement influentes.
- L'absence de multicolinéarité entre les variables indépendantes

Ces trois conditions sont également valables pour les modèles LM comme nous l'avons mentionné dans la section précédente. La distance de *cook* peut ainsi être utilisée pour détecter les potentielles valeurs aberrantes et le facteur d'inflation de la variance (*VIF*) pour détecter la multicolinéarité. Les conditions d'applications particulières seront détaillées dans les sections dédiées à chaque modèle.

1.1.4 Résidus et déviance

Dans la section sur la régression linéaire simple, nous avions présenté la notion de résidu, soit l'écart entre les valeurs prédictes par le modèle et la valeur réelle de Y . Pour un modèle GLM, ces résidus traditionnels ne sont pas très informatifs si la variable à modéliser est binaire, multinomiale ou même de comptage. Lorsque l'on travaille avec des GLM, on préférera utiliser trois autres formes des résidus, soit les résidus de Pearson, les résidus de déviance et les résidus simulés.

Les résidus de Pearson sont une forme ajustée des résidus classiques : on soustrait à la valeur observée la valeur attendue divisée par la racine carrée de la variance modélisée. Leur formule varie donc d'un modèle à l'autre puisque l'expression de la variance change en fonction de la distribution du modèle. Pour un modèle GLM Gaussien, cela donne :

$$r_i = \frac{y_i - \mu_i}{\sigma} \quad (1.6)$$

Pour un modèle GLM de Bernoulli :

$$r_i = \frac{y_i - p_i}{\sqrt{p_i(1-p_i)}} \quad (1.7)$$

Les résidus de déviance sont basés sur le concept de *likelihood* présenté dans la section ???. Pour rappel, le *likelihood*, ou la vraisemblance d'un modèle, correspond à la probabilité conjointe d'avoir observé les données Y selon le modèle étudié. Pour des raisons mathématiques (voir section ??), on préfère généralement calculer le *log likelihood*. Plus cette valeur est forte, moins le modèle se trompe. Cette interprétation est donc inverse à celle des résidus classiques, c'est pourquoi on multiplie le *log likelihood* par -2 pour retrouver une interprétation intuitive. Ainsi, pour chaque observation i , on peut calculer

$$d_i = -2 * \log(P(y_i|M_e)) \quad (1.8)$$

Avec d_i le résidu de déviance, et $P(y_i|M_e)$ la probabilité d'avoir observé la valeur y_i selon le modèle étudié (M_e).

La somme de tous ces résidus est appelée la déviance totale du modèle.

$$D(M_e) = \sum_{i=1}^n -2 * \log(P(y_i|M_e)) \quad (1.9)$$

Il s'agit donc d'une quantité représentant à quel point le modèle est erroné vis-à-vis des données. Notez qu'en tant que telle, la déviance n'a pas d'interprétation directe, en revanche, elle est utilisée pour calculer des mesures d'ajustements des modèles GLM.

Les résidus simulés sont une avancée récente dans le monde des GLM, ils fournissent une définition et une interprétation harmonisée des résidus pour l'ensemble des modèles GLM. Dans la section sur les LM (ref), nous avions vu comment interpréter les graphiques des résidus pour détecter d'éventuels problèmes dans le modèle. Cependant, cette technique est bien plus compliquée à mettre en œuvre pour les GLM puisque la forme attendue des résidus varie en fonction de la distribution choisie pour modéliser Y . La façon la plus efficace de procéder est d'interpréter les graphiques des résidus simulés qui ont la particularité d'être **identiquement distribués quelque soit le modèle GLM construit**. Ces résidus simulés sont compris entre 0 et 1, et sont calculés de la manière suivante :

- À partir du modèle GLM construit, simuler S fois (généralement 1000) une variable Y' avec autant d'observation (n) que Y . Cette variable simulée est une combinaison de la prédiction du modèle (coefficients et variables indépendantes) et de sa dispersion (variance). Ces simulations représentent des variations vraisemblables de la variable Y si le modèle est correctement spécifié. En d'autres termes, si le modèle représente bien le phénomène à l'origine de la variable Y , alors les simulations Y' issues du modèle devraient être proche de la variable Y originale. Pour une explication plus détaillée de ce que signifie simuler des données à partir d'un modèle, référez-vous au bloc attention ci-dessous.
- Pour chaque observation, on obtient ainsi S valeurs formant une distribution, Ds_i , des valeurs simulées par le modèle pour cette observation.
- Pour chacune de ces distributions, on calcule la probabilité cumulative d'observer la vrai valeur Y_i d'après la distribution Ds_i . Cette valeur est comprise entre 0 (toutes les valeurs simulées sont plus grandes que Y_i) et 1 (toutes les valeurs simulées sont inférieures à Y_i).



Si le modèle est correctement spécifié, le résultat attendu est que la distribution de ces résidus est uniforme. En effet, il y a autant de chance que les simulations produisent des résultats supérieurs ou inférieurs à Y_i si le modèle représente bien le phénomène (Dunn and Smyth, 1996, Gelman and Hill (2006))). Si la distribution des résidus ne suit pas une loi uniforme, cela signifie que le modèle échoue à reproduire le phénomène à l'origine de Y ce qui doit nous alerter sur sa pertinence.

1.1.5 Vérifier l'ajustement

Il existe trois façons de vérifier l'ajustement d'un modèle GLM :

- Utiliser des mesures d'ajustement (AIC, pseudo R2, déviance expliquée, etc.)
- Comparer la distribution de la variable originale et des prédictions
- Comparer les prédictions du modèle avec les valeurs originales

Notez d'emblée que vérifier la qualité d'ajustement d'un modèle (ajustement aux données originales) ne revient pas à vérifier la validité d'un modèle (respect des conditions d'application). Cependant, ces dernières sont généralement liées car un modèle mal ajusté a peu de chance d'être valide et inversement.

1.1.5.1 Les mesures d'ajustement

Les mesures d'ajustement sont des indicateurs plus ou moins arbitraires dont le principal intérêt est de faciliter la comparaison entre plusieurs modèles similaires. Il est nécessaire de les reporter car dans certains cas, ils peuvent indiquer des modèles très mal ajustés.

1.1.5.1.1 La déviance expliquée

Rappelons que la déviance d'un modèle est une quantité représentant à quel point le modèle est erroné. L'idée de l'indicateur de la déviance expliquée est d'estimer le pourcentage de la déviance maximale observable dans les données que le modèle est parvenu à expliquer. La déviance maximale observable dans les données est obtenue en utilisant la déviance totale du modèle nul (noté M_n , soit un modèle dans lequel aucune variable indépendante n'est ajouté et ne comportant qu'un intercept). Cette déviance est maximale puisqu'aucun prédicteur n'est présent dans le modèle. On calcule ensuite le pourcentage de cette déviance totale qui a été contrôlée par le modèle étudié (M_e).

$$\text{déviance expliquée} = \frac{D(M_n) - D(M_e)}{D(M_n)} = 1 - \frac{D(M_e)}{D(M_n)} \quad (1.10)$$

Il s'agit donc d'un simple calcul de pourcentage entre la déviance maximale ($D(M_n)$) et la déviance expliquée par le modèle étudié ($D(M_n) - D(M_e)$). Cet indicateur est compris entre 0 et 1, plus il est petit, plus la capacité de prédiction du modèle est faible. Attention, cet indicateur ne tient pas compte de la complexité du modèle. Ajouter une variable indépendante supplémentaire ne peut qu'augmenter la déviance expliquée, ce qui ne signifie pas que la complexification du modèle soit justifiée (**voir section sur le principe de parcimonie**).

1.1.5.1.2 Les pseudo R^2

Le R^2 est une mesure d'ajustement représentant la part de la variance expliquée dans un modèle linéaire classique. Cette mesure n'est pas directement transposable au cas des GLM puisqu'ils peuvent être appliqués à des variables non continues et anormalement distribuées. Toutefois, il existe des mesures semblables appelées pseudo R^2 , remplissant un rôle similaire. Notez cependant qu'ils ne peuvent pas être interprétés comme le R^2 classique : **Ils ne représentent pas la part de la variance expliquée**. Ils sont compris dans l'intervalle 0 et 1. Plus la valeur s'approche de 1, plus l'ajustement est élevé.

TAB. 1.1 : Principaux pseudo R^2

Nom	Formule	Commentaire
MacFadden	$1 - \frac{\text{loglike}(M_e)}{\text{loglike}(M_n)}$	Le rapport des loglikelihood, très proche de la déviance expliquée
MacFadden ajusté	$1 - \frac{\text{loglike}(M_e) - K}{\text{loglike}(M_n)}$	Version ajustée du R2 de MacFadden tenant compte du nombre de paramètre (K) dans le modèle
Efron	$1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	Rapport entre la somme des résidus classiques au carré (numérateur) et de la somme des écarts au carré à la moyenne (dénominateur). Notez que pour un GLM Gaussien, ce pseudo R2 est identique au R2 classique
Cox & Snell	$1 - e^{-\frac{2}{n}(\text{loglike}(M_e) - \text{loglike}(M_n))}$	Transformation de la déviance afin de la mettre sur une échelle de 0 à 1 (mais ne pouvant atteindre exactement 1)
Nagelkerke	$\frac{1 - e^{-\frac{2}{n}(\text{loglike}(M_e) - \text{loglike}(M_n))}}{1 - e^{-\frac{2 * \text{loglike}(M_n)}{n}}}$	Ajustement du R2 de Cox et Snell pour que l'échelle de valeurs possibles puisse comporter 1 (attention car les valeurs de ce R2 tendent à être toujours plus fortes que les autres)

En dehors du pseudo R^2 de MacFadden ajusté, aucune de ces mesures ne tient compte de la complexité du modèle. Il est cependant intéressant de les reporter car des valeurs très faibles indiquent vraisemblablement un modèle avec une moindre capacité informative. À l'inverse, des valeurs trop fortes pourraient également indiquer un problème de sur-ajustement (**voir encadre sur le principe de parcimonie**).

1.1.5.1.3 Le critère d'information d'Akaike (AIC)

Probablement l'indicateur le plus répandu, sa formule est relativement simple car il s'agit seulement d'un ajustement de la déviance :

$$AIC = D(M_e) + 2K \quad (1.11)$$

Avec K le nombre de paramètres à estimer dans le modèle (coefficients, paramètres de distribution, etc.).

Le *AIC* n'a pas d'interprétation directe, mais permet de comparer deux modèles imbriqués (**voir section X pour la définition**). Plus le *AIC* est petit, mieux le modèle est ajusté. L'idée derrière cet indicateur est relativement simple. Si la déviance D est grande, alors le modèle est mal ajusté. Ajouter des paramètres (des coefficients pour de nouvelles variables X par exemple) ne peut que réduire D , mais cette réduction n'est pas forcément suffisamment grande pour justifier la complexification du modèle. Le *AIC* pondère donc D en lui ajoutant 2 fois le nombre de paramètres du modèle. Un modèle plus simple (avec moins de paramètres) parvenant à une même déviance est préférable à un modèle complexe (principe de parcimonie ou du rasoir d'Ockham), ce que permet de «quantifier» le *AIC*. Attention, le *AIC* **ne peut pas être utilisé pour comparer des modèles non imbriqués**. Notez que d'autres indicateurs similaires comme le *WAIC*, le *BIC* et le *DIC* sont utilisés dans un contexte d'inférence Bayésienne. Notez simplement que ces indicateurs sont conceptuellement proches du *AIC* et s'interprètent (à peu de choses près) de la même façon.

1.1.5.2 Comparer les distributions originale et prédictes

Une façon rapide de vérifier si un modèle est mal ajusté est de comparer la forme de la distribution originale et celle capturée par le modèle. L'idée est la suivante, si le modèle est bien ajusté aux données, il est possible de se servir de celui-ci pour générer de nouvelles données dont la distribution ressemble à celle des données originales. Si une différence importante est observable, alors les résultats du modèle ne sont pas fiables car le modèle échoue à reproduire le phénomène étudié. Cette lecture graphique ne permet pas de s'assurer que le modèle est valide ou bien ajusté, mais simplement d'écartier rapidement les mauvais candidats. Notez que cette méthode ne s'applique pas lorsque la variable modélisée est binaire, multinomiale ou ordinaire. Le graphique à réaliser comprend donc : la distribution de la variable dépendante Y (représentée avec un histogramme ou un graphique de densité) et plusieurs distributions simulées à partir du modèle. Cette approche est plus répandue dans la statistique bayésienne, mais elle reste pertinente dans l'approche fréquentiste. Il est rare de reporter ces figures, mais elles doivent faire partie de votre diagnostic.

Distinction entre simulation et prédition :

Notez ici que **simuler des données** à partir d'un modèle et **effectuer des prédictions** à partir d'un modèle sont deux opérations différentes. Prédire une valeur à partir d'un modèle revient simplement à appliquer son équation de régression à des données. Si l'on réutilise les mêmes données, la prédition renvoie toujours le même résultat, il s'agit de la partie systématique du modèle. Pour illustrer cela, admettons que nous ayons ajusté un modèle GLM de type gaussien (fonction de lien identitaire) avec trois variables continues X_1 , X_2 et

X_3 et des coefficients respectifs de 0,5, 1,2 et 1,8 ainsi qu'un intercept de 7. Nous pouvons utiliser ces valeurs pour prédire la valeur attendue de Y quand $X_1 = 3$, $X_2 = 5$ et $X_3 = 5$:

$$\text{Prediction} = 7 + 3 * 0.5 + 5 * 1.2 + 1.8 * 5 = 23.5$$

En revanche, simuler des données à partir d'un modèle revient à ajouter la dimension stochastique (aléatoire) du modèle. Puisque notre modèle GLM est gaussien, il comporte un paramètre σ (son écart type), admettons pour cet exemple qu'il soit de 1,2. Ainsi avec les données précédentes, il est possible de simuler un ensemble infini de valeurs dont la distribution est la suivante : $Normal(\mu = 23,5, \sigma = 1,2)$. 95% du temps, ces valeurs simulées se trouveront dans l'intervalle 21,1-25,9 ($\mu - 2\sigma; \mu + 2\sigma$), puisque cette distribution est normale. Les valeurs simulées dépendent donc de la distribution choisie pour le modèle et de l'ensemble des paramètres du modèle, pas seulement de l'équation de régression.

Si vous aviez à ne retenir qu'une seule phrase de ce bloc, retenez que la prédiction ne se réfère qu'à la partie systématique du modèle (équation de régression), alors que la simulation incorpore la partie stochastique (aléatoire) de la distribution du modèle. Deux prédictions effectuées sur des données identiques donnent des résultats identiques, ce qui n'est que rarement le cas pour la simulation.

1.1.5.3 Comparer les prédictions du modèle avec les valeurs originales

Dans le meilleur des mondes, les prédictions d'un modèle devraient être proches des valeurs réelles observées. Si ce n'est pas le cas, alors le modèle n'est pas fiable et ses paramètres ne sont pas informatifs. Dépendamment de la nature de la variable modélisée (quantitative ou qualitative), plusieurs approches peuvent être utilisées pour quantifier l'écart entre valeurs réelles et valeurs prédictives.

1.1.5.3.1 Pour une variable quantitative

La mesure la plus couramment utilisée pour une variable continue est l'erreur moyenne quadratique (Root Mean Square Error, RMSE en anglais).

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}} \quad (1.12)$$

Il s'agit donc de la racine carrée de la moyenne des écarts au carré entre valeurs réelles et prédictions. Le RMSE est exprimé dans la même unité que la donnée originale et nous donne donc une indication sur l'erreur moyenne de la prédiction du modèle. Admettons par exemple que nous modélisions les niveaux de bruit environnemental en ville en décibels et que notre modèle de régression ait un RMSE de 3,5. Cela signifierait qu'en moyenne notre modèle se trompe de 3,5 décibels (erreur pouvant être négative ou positive) ce qui serait énorme (3 décibels correspondent à une multiplication par deux de l'intensité sonore) et nous amènerait à reconsidérer la fiabilité du modèle. Notez que l'usage d'une moyenne quadratique plutôt qu'une moyenne arithmétique permet de donner plus d'impact aux larges erreurs et donc de pénaliser davantage des modèles faisant parfois des grosses erreurs de prédiction. Le RMSE est donc très sensible à la présence de valeurs aberrantes. À la place de la moyenne quadratique, il est possible d'utiliser la simple moyenne arithmétique des valeurs absolues des erreurs (MAE). Cette mesure est cependant moins souvent utilisée :

$$MAE = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n} \quad (1.13)$$

Ces deux mesures peuvent être utilisées pour comparer la capacité de prédiction de deux modèles appliqués aux mêmes données, même s'ils ne sont pas imbriqués. Elles ne permettent cependant pas de

prendre en compte de la complexité du modèle. Un modèle plus complexe aura toujours un *RMSE* et un *MAE* plus petits.

1.1.5.3.2 Pour une variable qualitative

Lorsque l'on modélise une variable qualitative, une erreur revient à prédire la mauvaise catégorie pour une observation. Il ainsi possible de compter pour un modèle le nombre de bonnes et de mauvaises prédictions et d'organiser cette information dans une **matrice de confusion**. Cette dernière prend la forme suivante pour un modèle binaire :

En colonnes du tableau 1.2, nous avons les catégories observées et en lignes les catégories prédictes. La diagonale représente les prédictions correctes. Dans le cas présent, le modèle a bien catégorisé 35 (15 + 20) observations sur 43, soit une précision totale de 81,4%. Il a en revanche mal classifié huit (18,6%). 5 A ont été catégorisés comme des B, soit 20% des A, et seuls trois B ont été catégorisé comme des A (13%).

Le matrice ci-dessus 1.2 ne comporte que deux catégories possibles, la variable Y modélisée était donc une variable binaire. Il est possible d'étendre facilement le concept de matrice de confusion au cas des variables avec plus de deux modalités (multinomiale). Le tableau 1.3 est un exemple de matrice de confusion multinomiale.

Trois mesures pour chaque catégorie peuvent être utilisée pour déterminer la capacité de prédiction du modèle :

- La précision (precision en anglais), soit le nombre de fois où une classe a été correctement prédicté, divisée par le nombre de fois où la classe a été prédicté
- Le rappel (recall en anglais), soit le nombre de fois où une classe a été correctement prédicté, divisée par le nombre de fois où elle se trouve dans les données originales
- Le score $F1$, soit la moyenne harmonique entre la précision et le rappel :

$$F1 = 2 * \frac{\text{précision} * \text{rappel}}{\text{précision} + \text{rappel}} \quad (1.14)$$

Il est possible de calculer les moyennes pondérées des différents indicateurs (macro-indicateurs) afin de disposer d'une valeur d'ensemble pour le modèle. La pondération est faite en fonction du nombre de cas réel de chaque catégorie, l'idée étant qu'il est moins grave d'avoir des indicateurs plus faibles pour des catégories moins fréquentes. Cependant, il est tout à fait possible que cette pondération ne soit pas souhaitable. C'est par exemple le cas dans de nombreuses études en santé portant sur des maladies rares ou l'attention est concentrée sur ces catégories peu fréquentes. Le coefficient de Kappa (indicateur de 0 à 1) peut aussi être utilisé pour quantifier la fidélité générale de la prédiction du modèle. Pour l'interprétation du coefficient de Kappa, référez-vous au tableau 1.5. Enfin, un test statistique basé sur la distribution binomiale peut être utilisé pour vérifier que le modèle atteint un niveau de précision supérieur au seuil de non-information. Ce seuil correspond à la proportion de la modalité la plus présente dans le jeu de donnée. Dans la matrice de confusion utilisée ci-dessus, ce seuil est de 30,5% (catégorie D), ce qui signifie qu'un modèle prédisant tout le temps la catégorie D aurait une précision de 30,5% pour cette catégorie.

TAB. 1.2 : Exemple de matrice confusion

Valeur prédictive / Valeur réelle	A	B	Total (%)
A	15	3	18 (41,9)
B	5	20	25 (51,1)
Total (%)	20 (46,6)	23 (53,5)	43 (81,4)

TAB. 1.3 : Exemple de matrice confusion multinomiale

Valeur prédictive / Valeur réelle	A	B	C	D	Total (%)
A	15	3	1	5	24 (18,7)
B	5	20	2	12	39 (30,4)
C	2	10	25	8	45 (35,2)
D	1	0	5	14	20 (15,6)
Total (%)	23 (18,1)	33 (25,7)	33 (25,7)	39 (30,5)	128

Il est donc nécessaire que notre modèle face mieux que ce seuil. Dans le cas de la matrice de confusion précédente, nous obtenons donc les valeurs suivantes :

Dans le cas de la matrice de confusion du tableau 1.3, nous obtenons donc les valeurs affichées dans le tableau 1.4

A la lecture du tableau 1.4, nous remarquons que :

- la catégorie D est la moins bien prédite des quatre catégories (faible précision et faible rappel)
- La catégorie C a une forte précision mais un faible rappel, ce qui signifie que de nombreuses observations étant initialement des A, B ou D ont été prédites comme des C. Ce constat est également vrai pour la catégorie B.
- Le coefficient de Kappa indique un accord modéré entre les valeurs originales et la prédiction
- La probabilité que la précision du modèle ne dépasse pas le seuil de non information est inférieure à 0.001, indiquant que le modèle a une précision supérieure à ce seuil.

1.1.6 Comparer deux modèles GLM

Comme nous avons pu le voir dans la section sur les régressions linéaires classiques, il est courant de comparer plusieurs modèles imbriqués. Cette procédure permet de déterminer si l'ajout d'une ou de plusieurs variables contribuent à significativement améliorer le modèle. Il est possible d'appliquer la même démarche aux GLM à l'aide du test de rapport de vraisemblance (*likelihood ratio test*). L'idée derrière ce test est de comparer le likelihood de deux modèles GLM imbriqués, la valeur de ce test se calcule avec l'équation suivante :

$$LR = 2(\text{loglik}(M_2) - \text{loglik}(M_1)) \quad (1.15)$$

Avec M_2 un modèle reprenant toutes les variables du modèle M_1 , impliquant donc que $\text{loglik}(M_2) >= \text{loglik}(M_1)$.

L'idée derrière ce test est que le modèle M_2 comporte plus de paramètre que le modèle M_1 et devrait donc être mieux ajusté aux données. Si c'est bien le cas, la différence entre les *loglikelihood* de deux modèles devrait être supérieure à 0. La valeur calculée LR suit une distribution du Chi2 avec un nombre de degré

TAB. 1.4 : Indicateurs de qualité de prédiction

	précision	rappel	F1
A	65,2	31,3	42,3
B	60,6	25,6	36,0
C	75,8	27,8	40,7
D	35,9	35,0	35,4
macro	57,8	30,0	38,2
Kappa	0.44		
Valeur de p (précision > NIR)	<0.0001		

TAB. 1.5 : Interprétation des valeurs du coefficient de Kappa

K	Interprétation
< 0	Désaccord
0 - 0,20	Accord très faible
0,21 - 0,40	Accord faible
0,41 - 0,60	Accord modéré
0,61 - 0,80	Accord fort
0,81 - 1	Accord presque parfait

de liberté égal au nombre de paramètres supplémentaire dans le modèle M_2 comparativement à M_1 . Avec ces deux informations, il est possible de déterminer la valeur de p associée à ce test et de déterminer si M_2 est significativement mieux ajusté que M_1 aux données. Notez qu'il existe aussi deux autres tests (test de Wald et test de Lagrange) ayant la même fonction. Il s'agit dans les deux cas d'approximations du test de rapport des vraisemblance dont la puissance statistique est inférieure au test de rapport de vraisemblance (Neyman et al., 1933).

1.2 Les principaux modèles GLM

Dans cette section, nous décrions les principaux modèles GLM utilisés. Il en existe de nombreuses variantes que nous ne pouvons pas toutes mentionner ici. L'objectif est donc de comprendre les rouages de ces modèles afin de pouvoir en cas de besoin reporter ces connaissances sur des modèles plus spécifiques. Pour faciliter la lecture de cette section, nous vous proposons une carte d'identité de chacun des modèles présentés. Elles contiennent l'ensemble des informations pertinentes à retenir pour chaque modèle.

1.2.1 Les modèles GLM pour des variables qualitatives

Nous abordons en premier les principaux GLM utilisés pour modéliser des variables binaires, multinomiales et ordinaires. Prenez bien le temps de saisir le fonctionnement du modèle logistique binomial car il sert de base pour les trois autres modèles présentés.

1.2.1.1 Le modèle logistique binomial

Le modèle logistique binomial est une généralisation du modèle de Bernoulli que nous avons présenté dans l'introduction de cette section. Le modèle logistique binomiale couvre donc deux cas de figure :

1. La variable observée est binaire (0 ou 1). Dans ce cas, le modèle logistique binomiale devient un simple modèle de Bernoulli.
2. La variable observée est un comptage (nombre de réussite) et on dispose d'une autre variable avec le nombre de réplications de l'expérience. Par exemple, pour chaque intersection d'un réseau routier, nous pourrions avoir le nombre de décès à vélo (variable Y de comptage) et le nombre de collisions vélo / automobile (variable quantifiant le nombre d'expérience, chaque collision étant une expérience). Spécifiquement, on tente de prédire le paramètre p de la distribution Binomiale à l'aide de notre équation de régression et la fonction logistique comme fonction de lien.

1.2.1.1.1 Interprétation des paramètres

Les seuls paramètres à estimer du modèle sont les coefficients β_0 et β . La fonction de lien logistique transforme la valeur de ces coefficients, en conséquence, ils ne peuvent plus être interprétés simplement.

TAB. 1.6 : Carte d'identité du modèle logistique binomial

Type de variable dépendante	Variante binaire (0 ou 1) ou comptage de réussite à une expérience (ex : 3 réussites sur 5 expériences)
Distribution utilisée	Binomiale
Formulation	$Y \sim Binomial(p)$ $g(p) = \beta_0 + \beta X$ $g(x) = \log(\frac{x}{1-x})$
Fonction de lien	logistique
Paramètre modélisé	p
Paramètres à estimer	β_0, β
Conditions d'applications	Non-séparation complète, Absence de surdispersion ou sousdispersion

β_0 et β sont des logarithmes de rapports de cote (*log odd ratio*). Le rapport de cote est relativement facile à interpréter. Pour l'obtenir il suffit d'utiliser la fonction exponentielle (l'inverse de la fonction logarithmique) pour passer des log rapport de cote à de simples rapport de cote. Donc si $exp(\beta)$ est inférieur à 1, il réduit la probabilité d'observer l'évènement et inversement si $exp(\beta)$ est supérieur à 1.

Par exemple, admettons que nous ayons eu un coefficient β_1 de 1,2 pour une variable X_1 dans une régression logistique. Il est nécessaire d'utiliser son exponentiel pour l'interpréter de façon intuitive. $exp(1, 2) = 3,32$, ce qui signifie que lorsque l'on augmente X_1 d'une unité, on multiplie les chances par 3,32 d'observer 1 plutôt que 0 comme valeur de Y. Admettons maintenant que β_1 vaut -1,2, on calcule donc $exp(-1, 2) = 0,30$, ce qui signifie qu'à chaque augmentation d'une unité de X_1 , on multiplie les chances par 0,30 d'observer 1 plutôt que 0 comme valeur de Y. En d'autres termes, on divise par 3,33 ($1/0,30 = 3,33$) les chances d'observer 1 plutôt que 0, soit une diminution de 70% ($1 - 0,3 = 0,7$) des chances d'observer 1 plutôt que 0.



Les rapports de cotes

Le rapport de cote, ou rapport des chances est une mesure utilisée pour exprimer l'effet d'un facteur sur une probabilité beaucoup utilisé dans le domaine de la santé mais aussi des paris. Prenons un exemple concret avec le port du casque à vélo. Si sur 100 accidents impliquant des cyclistes portant un casque on observe seulement 3 cas de blessures graves à la tête, contre 15 dans un second groupe de 100 cyclistes ne portant pas de casques, on peut calculer le rapport de cote suivant :

$$\frac{p(1-q)}{q(1-p)} = \frac{0,15 * (1-0,03)}{0,03 * (1-0,15)} = 5,71 \quad (1.16)$$

avec p la probabilité d'observer le phénomène (ici la blessure grave à la tête) dans le groupe 1 (ici les cyclistes sans casque) et q la probabilité d'observer le phénomène dans le groupe 2 (ici les cyclistes avec un casque). Ce rapport de cote indique que les cyclistes sans casques ont 5,71 fois plus de chances de se blesser gravement à la tête lors d'un accident que ceux portant un casque.

1.2.1.1.2 Les conditions d'application

La non-séparation complète signifie qu'aucune des variables X ne à elle seule être capable de parfaitement distinguer les deux catégories 0 et 1 de la variable Y. Dans un tel cas de figure, les algorithmes d'ajustement utilisés pour estimer les paramètres des modèles sont incapables de converger. Notez aussi l'absurdité de créer un modèle pour prédire une variable Y si une variable X est capable à elle seule de la prédire à coup sûr. Ce problème est appelé un effet de Hauck-Donner, il est assez facile de le repérer car la plupart du temps les fonctions de R signalent ce problème (message d'erreur sur la convergence). Sinon, des

valeurs extrêmement élevées ou faibles pour certains rapports de cote peuvent aussi indiquer un effet de Hauck-Donner.

La surdispersion est un problème spécifique aux distributions n'ayant pas de paramètre de dispersion (binomiale, poisson, exponentielle, etc...), pour ces dernières, la variance dépend directement de la moyenne. On parle de surdispersion lorsque dans un modèle les résidus (ou erreurs) sont plus dispersés de ce que suppose la distribution utilisée. À l'inverse, il est aussi possible (mais rare) d'observer de cas de sous-dispersion (lorsque la dispersion des résidus est plus petite que ce que suppose la distribution choisie). Ce cas de figure se produit généralement lorsque le modèle parvient à réaliser une prédiction trop précise pour être fiable. Si vous rencontrez une forte sous-dispersion, cela signifie souvent que l'un de vos prédicteurs provoque une séparation complète. La meilleure option dans ce cas est de supprimer le prédicteur en question du modèle. La variance attendue d'une distribution binomiale est : $nb * p * (1 - p)$, soit le produit entre le nombre de tirage, la probabilité de réussite et la probabilité d'échec. À titre d'exemple, si l'on considère une distribution binomiale avec un seul tirage et 50% de chances de réussite, sa variance serait : $1 * 0,5 * (1 - 0,5) = 0,25$.

On peut observer de la surdispersion dans un modèle binomial logistique pour plusieurs raisons :

- Il manque des variables importantes dans le modèle, conduisant à un mauvais ajustement et donc une surdispersion des erreurs
- Les observations ne sont pas indépendantes, impliquant qu'une partie de la variance n'est pas contrôlée et augmente les erreurs
- La probabilité de succès de chaque expérience varie d'une répétition à l'autre (différentes distributions)

La conséquence directe de la surdispersion est la sous-estimation de la variance des coefficients de régression. En d'autres termes, la surdispersion conduire à sous-estimer notre incertitude quant aux coefficients obtenus et réduire les valeurs de p calculées pour ces coefficients. Les risques de trouver des résultats significatifs à cause des fluctuations d'échantillonnage augmentent.

Pour détecter de la surdispersion ou de la sousdispersion dans un modèle logistique binomial, il est possible d'observer les résidus de déviance du modèle. Ces derniers sont supposés suivre une distribution du Chi2 avec $n - k$ degrés de libertés (avec n le nombre d'observation et k le nombre de coefficients dans le modèle). Par conséquent, la somme des résidus de déviance d'un modèle logistique binomiale divisée par le nombre de degré de libertés devrait être proche de 1. Une légère déviation (jusqu'à 0,15 au dessus ou au dessous de 1) n'est pas alarmante, au delà il est nécessaire d'ajuster le modèle.

Notez que si la variable Y modélisée est exactement binaire (chaque expérience est indépendante et n'est composée que d'un seul tirage) et que le modèle utilise donc une distribution de Bernoulli, le test précédent pour détecter une éventuelle surdispersion n'est pas valide. [Hilbe \(2009\)](#) parle de surdispersion implicite pour le modèle de Bernoulli et recommande notamment de toujours ajuster les erreurs standards des modèles utilisant des distribution de Bernoulli, Binomiale et Poisson. Pour cela, il est possible d'utiliser ce que l'on appelle des quasi-distribution, ou des estimateurs robustes ([Zeileis, 2004](#)).

1.2.1.1.3 Exemple appliqué dans R

Présentation des données

Pour illustrer le modèle logistique binomial, nous utilisons ici un jeu de données proposé par l'Union Européenne : l'enquête de déplacement sur la demande pour des systèmes de transports innovants¹. Pour

¹<https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/P82V9X>

cette enquête, un échantillon de 1000 individus représentatifs de la population a été sélectionné dans chacun des 26 États membres de l'UE, soit un total de 26000 observations. Pour chaque individu, ont été collectées des informations socio-professionnelles, son mode de transport le plus fréquent, le temps de trajets de son déplacement le plus fréquent et son niveau de sensibilité à la cause environnementale. Nous modélisons ici la probabilité qu'un individu déclare utiliser le plus fréquemment le vélo comme moyen de transport. Les variables explicatives sont résumées dans le tableau suivant. Il existe bien évidemment un grand nombre de facteurs au niveau individuel qui impactent la prise de décision sur le mode de transport. Les résultats de ce modèle ne doivent donc pas être pris avec un grand sérieux, il s'agit d'un exemple pédagogique.

Vérification des conditions d'applications

La première étape de la vérification des conditions d'application est de calculer les valeurs du facteur d'inflation de variance (VIF) pour s'assurer de l'absence de multicolinéarité trop forte entre les variables indépendantes.

```
library(car)

# Chargement des données
dfenquete <- read.csv("data/glm/enquete_transport_UE.csv")
dfenquete$Pays <- relevel(as.factor(dfenquete$Pays), ref = "Germany")

# Vérification du VIF

model1 <- glm(y ~
  Pays + Sexe + Age + Education + StatutEmploi + Revenu +
  Residence + Duree + ConsEnv,
  family = binomial(link="logit"),
  data = dfenquete
)
vif(model1)
```

TAB. 1.7 : Variable indépendantes utilisées pour prédire le mode de transport le plus utilisé

Nom de la variable	signification	Type de variable	Mesure
Pays	Pays de résidence	Variable multinomiale	Le nom d'un des 26 pays membres de l'UE
Sexe	Sexe biologique	Variable binaire	Homme ou Femme
Age	L'âge biologique	Variable continue	L'âge en nombre d'années variant de 16 à 84 ans dans le jeu de donnée
Education	Niveau d'éducation maximum atteint	Variable multinomiale	Premier cycle , Secondaire inférieur (classes supérieures de l'école élémentaire) , Secondaire , Troisième cycle
StatutEmploi	Employé ou non	Variable binaire	Employé ou non
Revenu	Niveau de revenu autodéclaré	Variable multinomiale	Très faible revenu , faible revenu , revenu moyen , revenu élevé , revenu très élevé et sans réponse
Residence	Lieu de résidence	Variable multinomiale	Zone rurale , petite ou moyenne ville (moins de 250 000 habitants) , Grande ville (entre 250 000 et 1 million d'habitants) , Aire métropolitaine (plus d'un million d'habitants)
Duree	Durée du voyage le plus fréquent en minutes autodéclarée	Variable continue	Nombre de minute
ConsEnv	Préoccupation environnementale	Variable ordinaire	Échelle de likert et 1 à 10

```

##          GVIF Df GVIF^(1/(2*Df))
## Pays      1.794797 27     1.010890
## Sexe      1.028618  1     1.014208
## Age       1.060256  1     1.029687
## Education 1.428872  3     1.061285
## StatutEmploi 1.151879  1     1.073256
## Revenu     1.220934  5     1.020162
## Residence  1.130526  3     1.020658
## Duree      1.042638  1     1.021096
## ConsEnv    1.090987  1     1.044503

```

L'ensemble des valeurs de VIF sont inférieures à 5, indiquant l'absence de multicolinéarité excessive dans le modèle. La seconde étape de vérification est le calcul des distances de Cook et l'identification d'éventuelles valeurs aberrantes

```

# calcul et représentation des distances de Cook
cookd <- data.frame(
  dist = cooks.distance(model1),
  oid = 1:nrow(dfenquete)
)

ggplot(cookd) +
  geom_point(aes(x = oid, y = dist), color = rgb(0.1,0.1,0.1,0.4), size = 1) +
  geom_hline(yintercept = 0.002, color = "red") +
  labs(x = "observations",
       y = "distance de Cook") +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank())

```

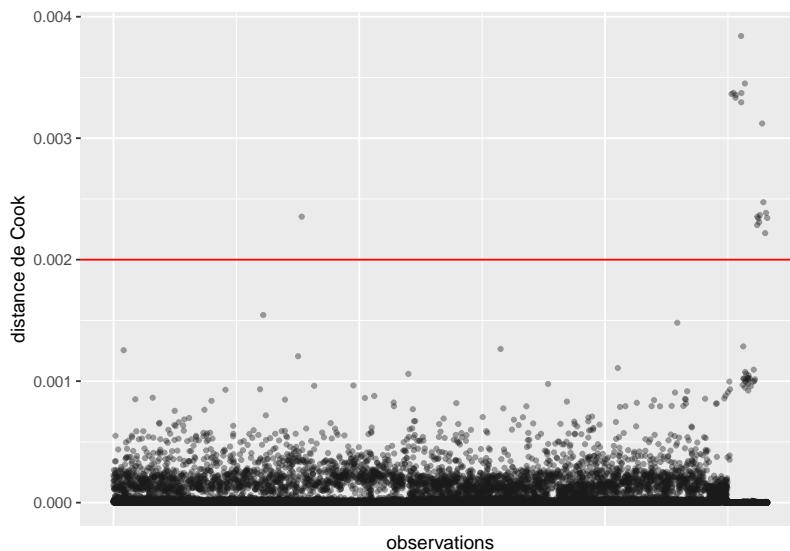


FIG. 1.4 : Distances de Cook pour le modèle binomial avec toutes les observations

Le calcul de la distance de Cook révèle un ensemble d'observations se démarquant nettement des autres (délimitées dans le graphique par la ligne rouge). Nous les isolons dans un premier temps pour les observer.

```

#isoler les observations avec de très fortes valeurs de Cook
#valeur seuil choisie : 0.002

cas_etranges <- subset(dfenquete, cookd$dist>=0.002)
cat(nrow(cas_etranges), 'observations se démarquant dans le modèle')

## 19 observations se démarquant dans le modèle

print(cas_etranges)

##          X y      Pays Sexe Age      Education Statut_emploi
## 7660    7660 1 Slovakia homme  50      universite   Employed
## 25150  25150 1      Malta homme  16      secondaire Not Employed
## 25227  25227 1      Malta femme 53      secondaire inferieur Not Employed
## 25309  25309 1      Malta femme 32      secondaire   Employed
## 25322  25322 1      Malta homme 38      universite   Employed
## 25536  25536 1      Malta homme 27      universite   Employed
## 25541  25541 1      Malta homme 38      secondaire inferieur Employed
## 25549  25549 1      Malta homme 31      universite   Employed
## 25690  25690 1 Luxembourg homme 32      universite   Employed
## 26190  26190 1      Cyprus homme 24      secondaire Not Employed
## 26201  26201 1      Cyprus homme 25      secondaire   Employed
## 26244  26244 1      Cyprus homme 32      secondaire   Employed
## 26269  26269 1      Cyprus homme 60      secondaire Not Employed
## 26303  26303 1      Cyprus homme 59      secondaire Not Employed
## 26393  26393 1      Cyprus homme 30      premier cycle Employed
## 26444  26444 1      Cyprus femme 52      universite   Employed
## 26516  26516 1      Cyprus homme 21      universite Not Employed
## 26549  26549 1      Cyprus homme 28      universite   Employed
## 26600  26600 1      Cyprus homme 36      secondaire   Employed
##          Revenu      Residence Duree mode_pref StatutEmploi ConsEnv
## 7660     moyen      zone rurale  775     velo     employe     7
## 25150    moyen      zone rurale   15     velo     sans emploi   3
## 25227    moyen      zone rurale  45     marche    sans emploi   5
## 25309    moyen petite-moyenne ville  25     marche    employe    4
## 25322    eleve      zone rurale  30     marche    employe   10
## 25536    tres eleve petite-moyenne ville  14     velo     employe   10
## 25541    moyen      zone rurale   5     marche    employe    8
## 25549    sans reponse petite-moyenne ville  60     velo     employe   10
## 25690    tres eleve petite-moyenne ville  720     velo     employe    6
## 26190    moyen      grande ville  20     velo     sans emploi   5
## 26201    faible      zone rurale  20     velo     employe    5
## 26244    tres faible petite-moyenne ville  18     velo     employe    4
## 26269    moyen      petite-moyenne ville  5     velo     sans emploi   7
## 26303    moyen      zone rurale   7     velo     sans emploi   8
## 26393    tres eleve petite-moyenne ville  61     velo     employe    5
## 26444    eleve      petite-moyenne ville 120     velo     employe    3
## 26516    moyen      petite-moyenne ville  25     velo     sans emploi   8
## 26549    tres faible petite-moyenne ville  15     velo     employe    2

```

```
## 26600      moyen petite-moyenne ville     8      velo      employe      1
```

En observant les 19 cas étranges, nous remarquons que la plupart des observations proviennent de Malte et de Chypre. Ces deux petites îles constituent des cas particuliers en Europe et devraient vraisemblablement faire l'objet d'une analyse séparée. Nous décidons donc de les retirer du jeu de données. Deux autres observations étranges sont observable en Slovaquie et au Luxembourg. Dans les deux cas, les répondants ont renseigné des temps de trajets fantaisiste de respectivement 775 et 720 minutes. Nous les retirons donc également de l'analyse.

```
#Retirer les observations aberrantes
dfenquete2 <- subset(dfenquete, (dfenquete$Pays %in% c("Malta", "Cyprus")) == F &
                     dfenquete$Duree < 400)

#Reajuster le modèle
model2 <- glm(y ~
                 Pays + Sexe + Age + Education + StatutEmploi + Revenu +
                 Residence + Duree + ConsEnv,
                 family = binomial(link="logit"),
                 data = dfenquete2)

#Recalculer la distance de Cook
cookd <- data.frame(
  dist = cooks.distance(model2),
  oid = 1:nrow(dfenquete2)
)

ggplot(cookd) +
  geom_point(aes(x = oid, y = dist), color = rgb(0.1,0.1,0.1,0.4), size = 1) +
  labs(x = "observations",
       y = "distance de Cook") +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank())
```

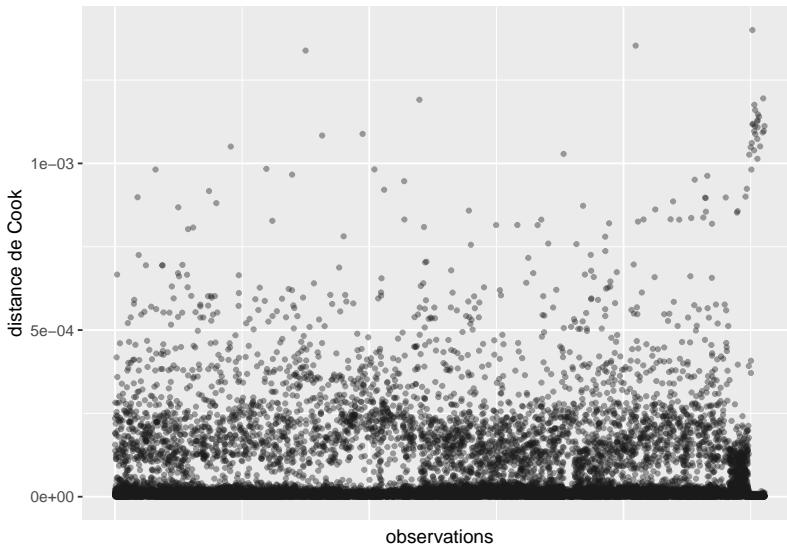


FIG. 1.5 : Distances de Cook pour le modèle binomial sans les valeurs aberrantes

Après avoir retiré ces valeurs aberrantes, nous n'observons plus de nouveaux cas singuliers avec les distances de Cook.

La prochaine étape de vérification des conditions d'application est l'analyse des résidus simulés. Nous commençons donc par calculer ces résidus et afficher leur histogramme.

```
library(DHARMa)

#Extraire les probabilités prédites par le modèle
probs <- predict(model2, type = "response")

#Calculer 1000 simulations à partir du modèle ajusté
sims <- lapply(1:length(probs), function(i){
  p <- probs[[i]]
  vals <- rbinom(n = 1000, size = 1, prob = p)
})
matsim <- do.call(rbind, sims)

#utiliser le package DHARMA pour calculer les résidus simulés
sim_res <- createDHARMA(simulatedResponse = matsim,
                           observedResponse = dfenquete2$y,
                           fittedPredictedResponse = probs,
                           integerResponse = T)

ggplot() +
  geom_histogram(aes(x = residuals(sim_res)),
                 bins = 30, fill = "white", color = rgb(0.3,0.3,0.3))
```

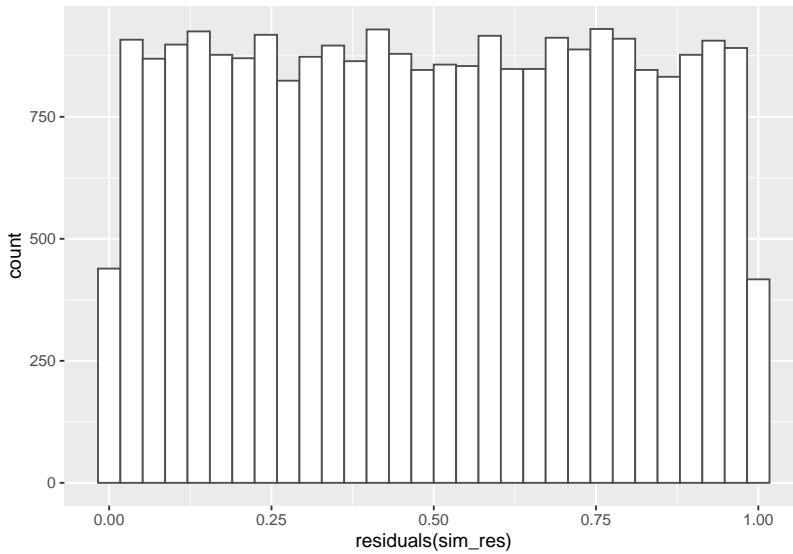


FIG. 1.6 : Distribution des résidus simulé pour le modèle binomial

L'histogramme indique clairement que les résidus simulés suivent une distribution uniforme. Il est possible d'aller plus loin dans le diagnostic en utilisant la fonction `plot` sur l'objet `sim_res`. La partie de droite de la figure ainsi obtenue (figure 1.7) est un diagramme de quantiles (ou QQ plot). Les points du graphique sont supposés suivre une ligne droite matérialisée par la ligne rouge. Une déviation de cette ligne indique un éloignement des résidus de leur distribution attendue. Trois tests sont également réalisés par la fonction.

- Le premier (KS test) permet justement de tester si les points dévient significativement de la ligne droite. Dans notre cas, la valeur de p n'est pas significative, indiquant que les résidus ne dévient pas de la distribution uniforme.

- Le second test permet de vérifier la présence de sur ou sous dispersion. Dans notre cas ce test n'est à nouveau pas significatif, n'indiquant donc ni surdispersion ni sousdispersion.
- Le dernier test permet de vérifier si des valeurs aberrantes sont présentes dans les résidus. Une valeur non significative indique une absence de valeur aberrantes.

Le second graphique permet de comparer les résidus et les valeurs prédictes. L'idéal est donc d'observer une ligne droite horizontale rouge au milieu du graphique qui indiquerait une absence de relation entre les valeurs prédictes et les résidus (ce que nous observons bien ici).

```
plot(sim_res)
```

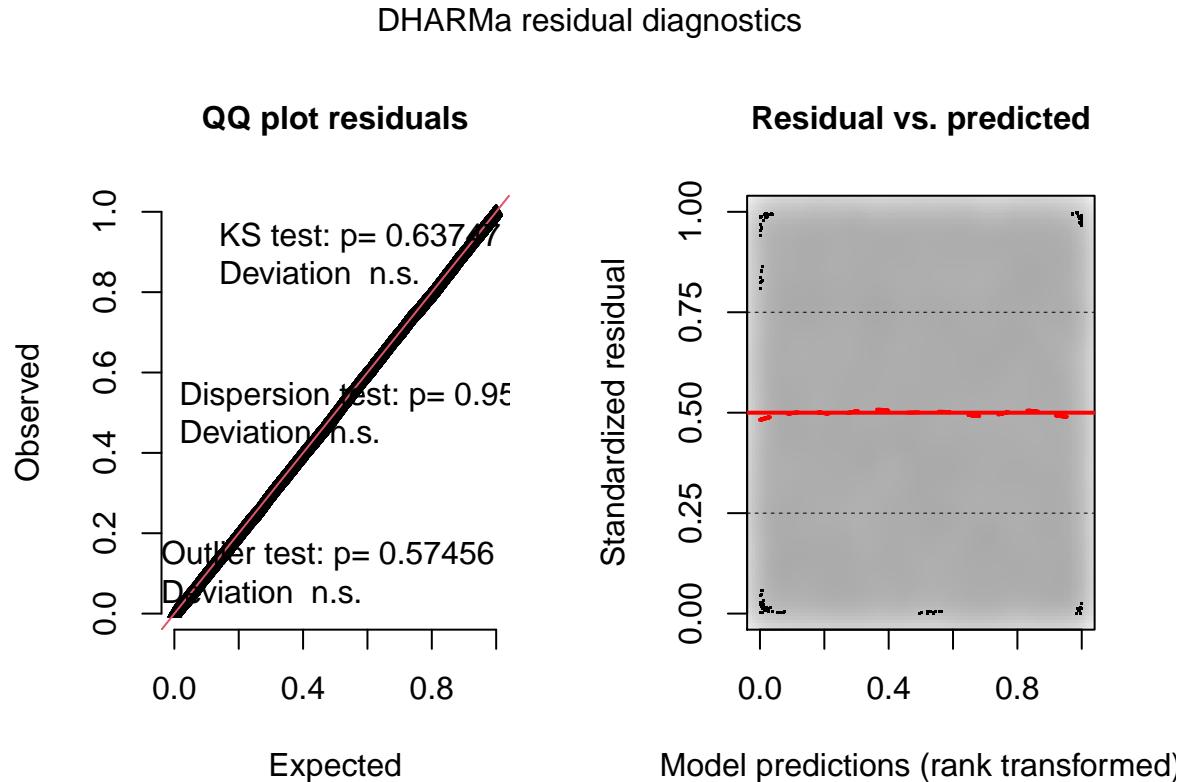


FIG. 1.7 : Diagnostic des résidus simulés par le package DHARMA

L'analyse approfondie des résidus nous permet donc de conclure que le modèle respecte les conditions d'application et que nous pouvons passer à la vérification de la qualité d'ajustement du modèle.

Vérification de la qualité d'ajustement

Pour calculer les différents R² d'un modèle GLM, nous proposons la fonction suivante :

```
rsqs <- function(loglike.full, loglike.null, full.deviance, null.deviance, nb.params, n){
  #calcul de la déviance expliquée
  explained_dev <- 1 - (full.deviance / null.deviance)

  K <- nb.params
  #R2 de MacFadden ajusté
  r2_faddenadj <- 1 - (loglike.full - K) / loglike.null

  Lm <- loglike.full
```

```

Ln <- loglike.null
#R2 de Cox and Snell
Rcs <- 1 - exp((-2/n) * (Lm-Ln))
#R2 de Nagelkerke
Rn <- Rcs / (1-exp(2*Ln/n))
return(
  list("deviance expliquée" = explained_dev,
       "MacFadden ajuste" = r2_faddenadj,
       "Cox and Snell" = Rcs,
       "Nagelkerke" = Rn
  )
)
}

```

Nous pourrons l'utiliser pour l'ensemble des modèles GLM de la section. Dans le cas du modèle binomial nous obtenons :

```

#Ajuster un modèle null avec seulement un intercept
model2.null <- glm(y ~1,
                     family = binomial(link="logit"),
                     data = dfenquete2)

#calculer les R2
rsqs(loglike.full = as.numeric(logLik(model2)), # loglikelihood du modèle complet
      loglike.null = as.numeric(logLik(model2.null)), # loglikelihood du modèle null
      full.deviance = deviance(model2), # deviance du modèle complet
      null.deviance = deviance(model2.null), # deviance du modèle null
      nb.params = model2$rank, # nombre de paramètres dans le modèle
      n = nrow(dfenquete2) # nombre d'observations
    )

## `$`deviance expliquée`
## [1] 0.0876057
##
## `$`MacFadden ajuste`
## [1] 0.08357379
##
## `$`Cox and Snell`
## [1] 0.0689509
##
## `$Nagelkerke
## [1] 0.1236597

```

La déviance expliquée par le modèle est de 8,8%, les pseudos R2 de McFadden (ajusté), Efron et Nagelkerke sont respectivement 0,084, 0,069 et 0,12. L'ensemble de ces valeurs sont relativement faibles et indiquent qu'une large partie de la variabilité de Y reste inexpliquée.

Pour vérifier la qualité de prédiction du modèle, nous devons comparer les catégories prédictes et les catégories réelles de notre variable dépendante et construire une matrice de confusion. Cependant, un modèle GLM binomiale prédit **la probabilité d'appartenance au groupe 1** (ici les personnes utilisant le vélo pour effectuer leur déplacement le plus fréquent). Pour convertir ces probabilités prédictes en catégories prédictes, il faut choisir une probabilité seuil au-delà de laquelle on considère que la valeur attendue est 1 (cycliste) plutôt que 0 (autre). Un exemple naïf serait de prendre le seuil 0,5, ce qui

signifierait que si le modèle prédit qu'une observation a au moins 50% de chance d'être un cycliste, alors nous l'attribuons à cette catégorie. Cependant, cette méthode donne des résultats peu intéressants lorsque l'échantillon est déséquilibré. Dans notre cas, les cyclistes sont beaucoup moins nombreux que les autres usagers (3616 contre 21931), utiliser un seuil de 50% reviendrait donc à très certainement manquer beaucoup de cyclistes dans notre prédition. Le choix de ce seuil est généralement obtenu en trouvant le point d'équilibre entre la sensibilité (proportion de 1 correctement identifiés) et la spécificité (proportion de 0 correctement identifiés). Ce point d'équilibre est identifiable graphiquement en calculant la spécificité et la sensibilité de la prédition selon toutes les valeurs possibles du seuil.



```
library(ROCR)

#Obtention des prédition du modéle
prob <- predict(model2, type = "response")

#calcul de la sensibilité et de la spécificité (package ROCR)
predictions <- prediction(prob, dfenquete2$y)

sens <- data.frame(x=unlist(performance(predictions, "sens")@x.values),
                     y=unlist(performance(predictions, "sens")@y.values))
spec <- data.frame(x=unlist(performance(predictions, "spec")@x.values),
                     y=unlist(performance(predictions, "spec")@y.values))

# trouver numériquement la valeur seuil (minimiser la différence absolue
# entre sensibilité et spécificité)
real <- dfenquete2$y

find_cutoff <- function(seuil){
  pred <- ifelse(prob>seuil,1,0)
  sensi <- sum(real==1 & pred==1) / sum(real==1)
  spec <- sum(real==0 & pred==0) / sum(real==0)
  return(abs(sensi-spec))
}

prob_seuil <- optimize(find_cutoff,interval = c(0,1), maximum = F)$minimum

cat("Le seuil de probabilité à retenir équilibrant la Sensibilité et la Spécificité est",prob_seuil)
```

```
## Le seuil de probabilité à retenir équilibrant la Sensibilité et la Spécificité est 0.14785
```

```
# affichage du graphique

ggplot() +
  geom_line(data = sens, mapping = aes(x = x, y = y)) +
  geom_line(data = spec, mapping = aes(x = x,y = y,col="red")) +
  scale_y_continuous(sec.axis = sec_axis(~., name = "Spécificité")) +
  labs(x='Probabilité seuil', y="Sensibilité") +
  geom_vline(xintercept = prob_seuil, color = "black", linetype = "dashed") +
  annotate(geom = "text", x = prob_seuil, y = 0.01, label = round(prob_seuil,3))+
  theme(axis.title.y.right = element_text(colour = "red"), legend.position="none")
```

On constate avec la figure 1.8 que si la valeur seuil choisi est 0 %, alors la prédition a une sensibilité parfaite (le modèle prédit toujours 1, donc tous les 1 sont détectés) et à l'inverse si le seuil choisi est 100% alors la prédition à une spécificité parfaite (le modèle prédit toujours 0, donc tous les 0 sont détectés).

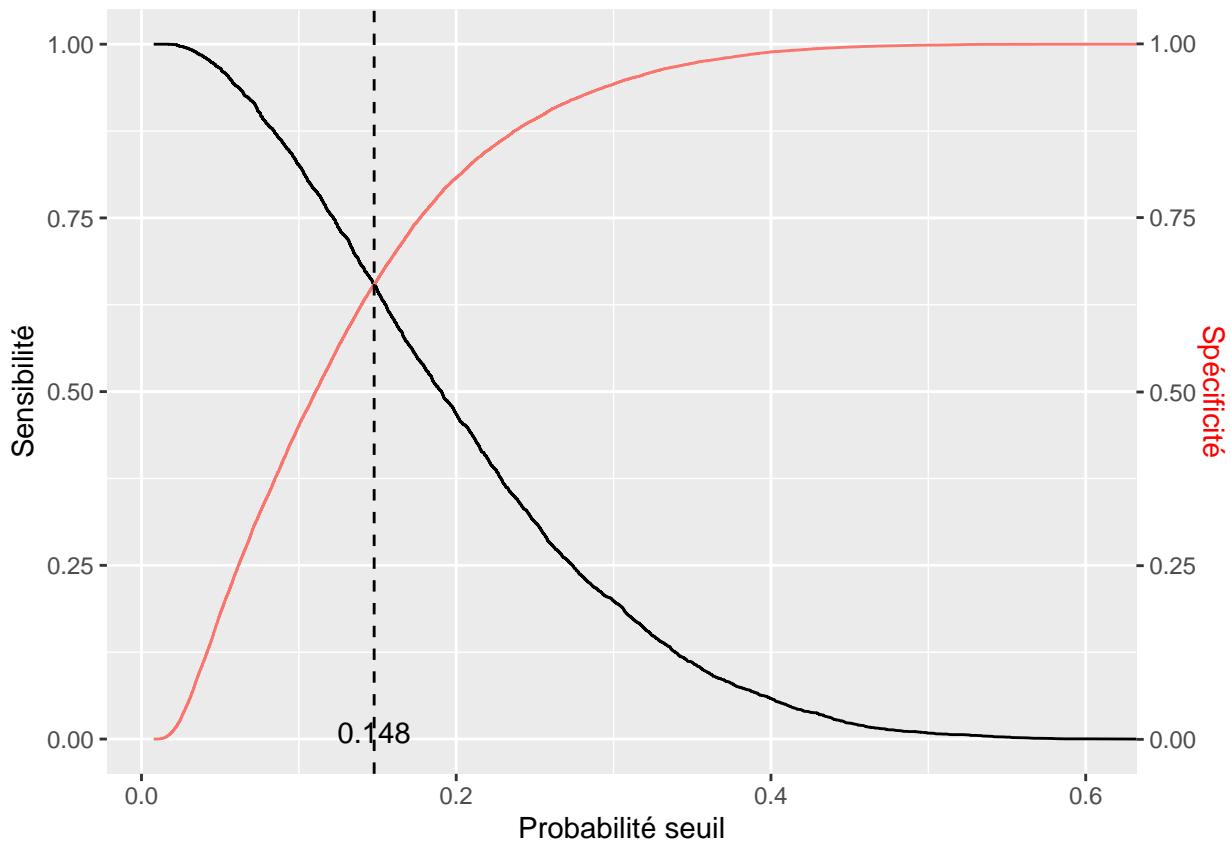


FIG. 1.8 : Point d'équilibre entre sensibilité et spécificité

Dans notre cas, la valeur d'équilibre est d'environ 0,148, donc si le modèle prédit une probabilité au moins égale à 14,8% qu'un individu utilise le vélo pour son déplacement le plus fréquent, nous l'attribuerons à cette catégorie. Avec ce seuil, nous pouvons convertir les probabilités prédictes en classes prédictes et construire notre matrice de confusion.

```

library(caret) # pour la matrice de confusion

#calcul des catégories prédictes
ypred <- ifelse(predict(model2,type="response")>0.148,1,0)
info <- confusionMatrix(as.factor(dfenquete2$y), as.factor(ypred))

# affichage des valeurs brutes de la matrice de confusion
print(info)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction      0      1
##               0 14355  7576
##               1 1251   2365
##
##                   Accuracy : 0.6545
##                   95% CI : (0.6486, 0.6603)
##       No Information Rate : 0.6109

```

```

##      P-Value [Acc > NIR] : < 2.2e-16
##
##          Kappa : 0.1783
##
## McNemar's Test P-Value : < 2.2e-16
##
##          Sensitivity : 0.9198
##          Specificity : 0.2379
##          Pos Pred Value : 0.6546
##          Neg Pred Value : 0.6540
##          Prevalence : 0.6109
##          Detection Rate : 0.5619
##          Detection Prevalence : 0.8585
##          Balanced Accuracy : 0.5789
##
## 'Positive' Class : 0
##

```

Les résultats proposés par le package **caret** sont exhaustifs, nous vous proposons ici une façon de les présenter dans deux tableaux. Le premier (1.8) présente la matrice de confusion et le second (1.9) les indicateurs de qualité de prédiction.

D'après ces indicateurs, nous constatons que le modèle a une capacité de prédiction relativement faible, mais tout de même significativement supérieur au seuil de non information. La valeur de rappel pour la catégorie 1 (cycliste) est faible, indiquant que le modèle a manqué un nombre important de cyclistes lors de sa prédiction.

Interprétation des résultats du modèle

L'interprétation des résultats d'un modèle binomial passe par la lecture des rapports de cotes et de leurs intervalles de confiance. Nous commençons donc par calculer la version robuste des erreurs standards des coefficients :

```

library(sandwich) #pour calculer les erreurs standards robustes

covModel2 <- vcovHC(model2, type = "HC0") # méthode HC0, basée sur les résidus
stdErrRobuste <- sqrt(diag(covModel2)) # extraire la diagonale

#extraction des coefficients
coeffs <- model2$coefficients
#recalcule des scores Z
zvalRobuste <- coeffs / stdErrRobuste
#recalcule des valeurs de P
pvalRobuste <- 2 * pnorm(abs(zvalRobuste), lower.tail = FALSE)
#calcule des rapports de cote

```

TAB. 1.8 : Matrice de confusion pour le modèle binomial

	0 (réel)	1 (réel)	Total	%
0 (prédict)	14355	7576	21931	85.8
1 (prédict)	1251	2365	3616	14.2
Total	15606	9941	25547	
%	61.1	38.9		

TAB. 1.9 : Matrice de confusion pour le modèle binomial

	Précision	Rappel	F1
0	0.65	0.92	0.76
1	0.65	0.24	0.35
macro	0.65	0.65	0.6
Kappa	0.18		
Valeur de p (précision > NIR)	0		

```

oddRatio <- exp(coeffs)
#calcule des intervalles de confiance à 95% des rapports de cote
lowerBound <- exp(coeffs - 1.96 * stdErrRobuste) 
upperBound <- exp(coeffs + 1.96 * stdErrRobuste)
#etoiles pour les valeurs de p
starsp <- case_when(pvalRobuste <= 0.001 ~ "***",
                      pvalRobuste > 0.001 & pvalRobuste <= 0.01 ~ "**",
                      pvalRobuste > 0.01 & pvalRobuste <= 0.05 ~ "*",
                      pvalRobuste > 0.05 & pvalRobuste <= 0.1 ~ ".",
                      TRUE ~ "")
)

#compilation des resultats dans un tableau
tableau_binom <- data.frame(
  coefficients = coeffs,
  rap.cote = oddRatio,
  err.std = stdErrRobuste,
  score.z = zvalRobuste,
  p.val = pvalRobuste,
  rap.cote.2.5 = lowerBound,
  rap.cote.97.5 = upperBound,
  sign = starsp
)

```

Considérant que la variable Pays a 24 modalités, il est plus judicieux de présenter ces rapports de cote sous forme d'un graphique. Nous avons choisi l'Allemagne comme catégorie de référence puisqu'elle fait partie des pays avec une importante part modale pour le vélo sans constituer un cas extrême comme le Danemark.

```

#isoler les ligne du tableau recapitulatif pour les pays
paysdf <- subset(tableau_binom, grepl("Pays", row.names(tableau_binom), fixed = T))
paysdf$Pays <- gsub("Pays", "", row.names(paysdf), fixed=T)

ggplot(data = paysdf) +
  geom_vline(xintercept = 1, color = "red") #afficher la valeur de reference
  geom_errorbarh(aes(xmin = rap.cote.2.5, xmax = rap.cote.97.5, y = reorder(Pays, rap.cote)), height = 0) +
  geom_point(aes(x = rap.cote, y = reorder(Pays, rap.cote))) +
  geom_text(aes(x = rap.cote.97.5, y = reorder(Pays, rap.cote), label = paste("RC : ", round(rap.cote, 2), sep=""))),
  labs(x = "Rapports de cote",
       y = "Pays (référence : Allemagne)")

```

Dans la figure 1.9, la barre horizontale pour chaque pays représente l'intervalle de confiance de son rapport de cote (le point), plus cette ligne est longue, plus grande est l'incertitude autours de ce paramètre.

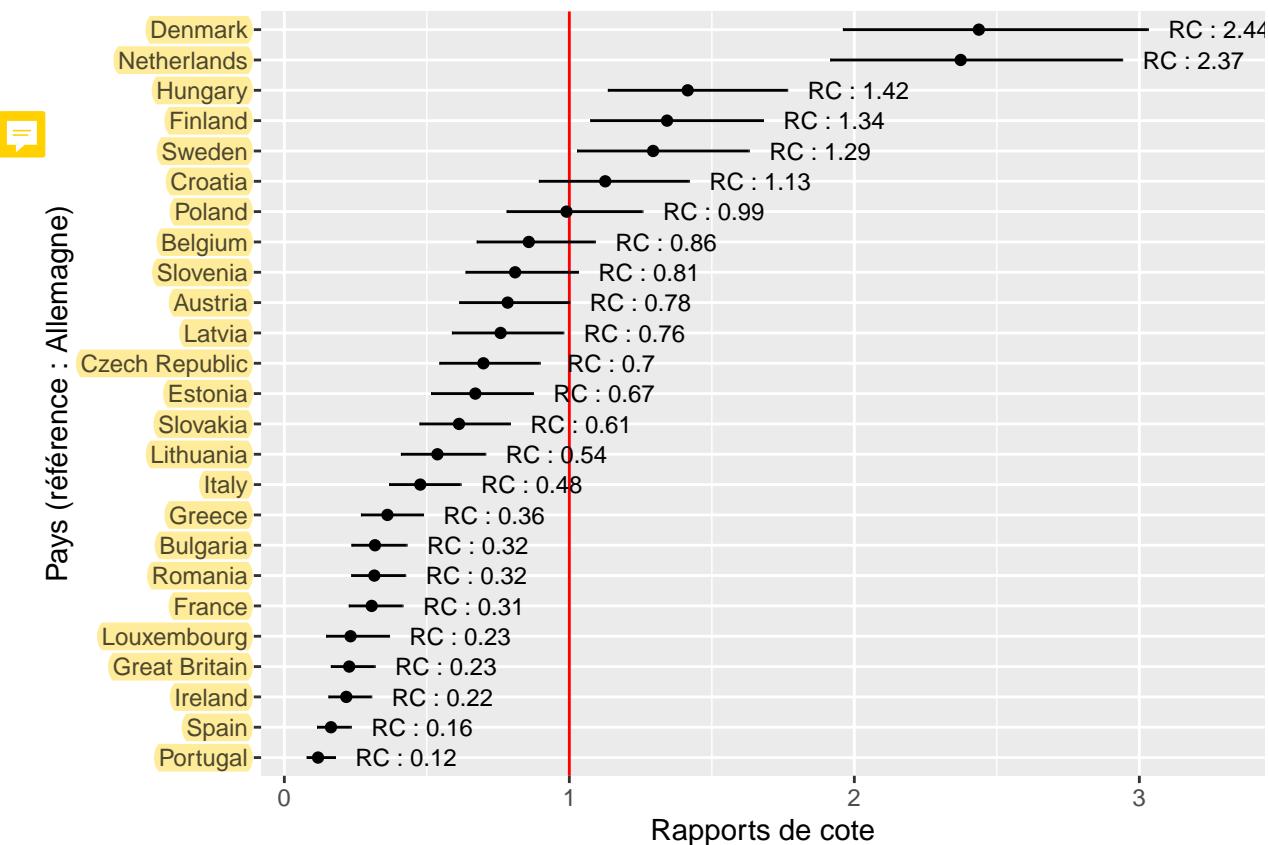


FIG. 1.9 : Rapports de cote pour les différents pays de l'UE

Lorsque les lignes de deux pays se chevauchent, cela signifie qu'il n'y a pas de différence significative au seuil 0,05 entre les rapports de cotes des deux pays. La ligne rouge tracée à $x = 1$, représente le rapport de cote du pays de référence (ici l'Allemagne). On constate ainsi que comparativement à un individu vivant en Allemagne, ceux vivant au Danemark et au Pays Bas ont 2,4 fois plus de chance d'utiliser le vélo pour leur déplacement le plus fréquent. Les Pays de l'Ouest de l'UE (Grèce, Italie, France, Luxembourg, Royaume Uni, Espagne, Portugal ...) ont en revanche des rapports de cotes plus faibles. En France, les chances qu'un individu utilise le vélo pour son trajet le plus fréquent sont 3,22 ($1/0.31$) fois plus faibles que si l'individu vivait en Allemagne.

Pour le reste des coefficients et rapport de cote, nous les rapportons dans le tableau (1.10).

Les chances pour un individu d'utiliser le vélo pour son trajet le plus fréquent sont augmentées de 45% s'il s'agit d'un homme plutôt qu'une femme. Pour l'âge, nous constatons un effet relativement faible puisque chaque année supplémentaire réduit les chances qu'un individu utilise le vélo comme mode de transport pour son trajet le plus fréquent de 0,9% ($(0.991-1)*100$). Le fait d'être sans emploi augmente les chances d'utiliser le vélo de 29% comparativement au fait d'avoir un emploi. Concernant le niveau d'éducation, seul le coefficient pour le groupe des personnes de la catégorie secondaire inférieur est significatif, indiquant que les personnes de ce groupe ont 35% de chances en plus d'utiliser le vélo comme mode de transport pour leur déplacement le plus fréquent comparativement aux personnes du groupe premier cycle. Pour le revenu, seul le groupe avec de très faibles revenus se distingue significativement du groupe avec une revenu élevé avec un rapport de cote de 1,27, soit 27% de chances en plus d'utiliser le vélo.

Comparativement à ceux vivant dans une aire métropolitaine, les personnes vivant dans des petites, moyennes et grandes villes ont des chances accrues d'utiliser le vélo comme mode de déplacement

TAB. 1.10 : Résultats du modèle binomial

Variable	Coefficient	Rapport de cote	P	RC 2,5%	RC 97,5%
Intercept	-2.497	0.082	<0.001	0.058	0.118
<i>Sexe</i>					
ref : femme	–	–	–	–	–
homme	0.372	1.451	<0.001	1.347	1.562
Age	-0.009	0.991	<0.001	0.988	0.994
<i>Education</i>					
ref : premier cycle	–	–	–	–	–
secondaire	0.193	1.213	0.066	0.987	1.490
secondaire inferieur	0.301	1.351	0.008	1.081	1.687
universite	0.146	1.157	0.177	0.936	1.432
<i>StatutEmploi</i>					
ref : employe	–	–	–	–	–
sans emploi	0.257	1.293	<0.001	1.190	1.405
<i>Revenu</i>					
ref : elevé	–	–	–	–	–
faible	0.077	1.080	0.286	0.938	1.244
moyen	0.042	1.043	0.523	0.917	1.185
sans reponse	0.217	1.242	0.034	1.016	1.517
tres elevé	-0.120	0.887	0.524	0.613	1.283
tres faible	0.240	1.271	0.006	1.073	1.505
<i>Residence</i>					
ref : aire metropolitaine	–	–	–	–	–
grande ville	0.273	1.314	<0.001	1.146	1.507
petite-moyenne ville	0.277	1.319	<0.001	1.169	1.487
zone rurale	-0.119	0.888	0.087	0.775	1.017
Duree	-0.001	0.999	0.326	0.998	1.001
ConsEnv	0.102	1.108	<0.001	1.087	1.130

pour leur trajet le plus fréquent. En revanche, on ne peut observer aucune différence entre la probabilité d'utiliser le vélo dans une métropole et en zone rurale. La figure 1.10 permet de clairement visualiser cette situation. Rappelons que la référence est la situation : vivre dans une région métropolitaine, et est représentée par la ligne verticale rouge. Plusieurs pistes d'interprétations peuvent être envisagées pour ce résultat :

- En métropole et dans les zones rurales, les distances domicile-travail tendent à être plus grandes que dans les petites, moyennes et grandes villes.
- En métropole, le système de transport en commun est davantage développé et entre donc en concurrence avec les modes de transport actifs.

```
#isoler les ligne du tableau recapitulatif pour les lieux de résidence
residdf <- subset(tableau_binom, grep("Residence", row.names(tableau_binom), fixed = T))
residdf$resid <- gsub("Residence", "", row.names(residdf), fixed=T)

ggplot(data = residdf) +
  geom_vline(xintercept = 1, color = "red") + #afficher la valeur de référence
  geom_errorbarh(aes(xmin = rap.cote.2.5, xmax = rap.cote.97.5, y = resid), height = 0) +
  geom_point(aes(x = rap.cote, y = resid)) +
  geom_text(aes(x = rap.cote.97.5, y = resid, label = paste("RC : ", round(rap.cote,2), sep=""))), size = 3, nudge_x = 0)
```

Il est aussi intéressant de noter que la durée des trajets ne semble pas impacter la probabilité d'utiliser

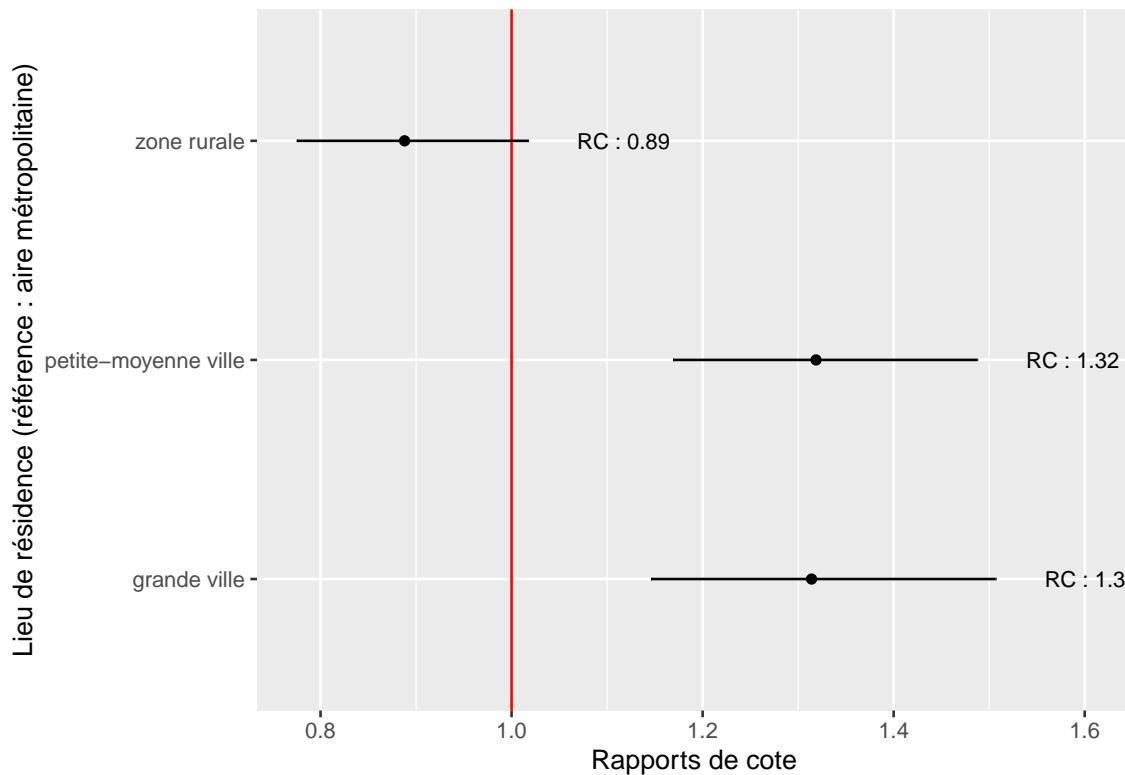


FIG. 1.10 : Rapports de cote pour les différents lieux de résidence

le vélo. Enfin, une conscience environnementale plus affirmée semble être associée avec une probabilité supérieure d'utiliser le vélo pour son déplacement le plus fréquent, avec une augmentation des chances de 11% ((1 - 0.108)*100) pour chaque point supplémentaire sur l'échelle de likert.

Afin de simplifier la présentation de certains résultats, il est possible de calculer exactement les prédictions réalisées par le modèle. Un bon exemple ici est le cas de la variable âge, quelle différence peut-on attendre entre deux individus identiques ayant seulement une différence d'âge de 15 ans ?

Prenons comme individu un homme de 30 ans, vivant dans une grande ville allemande, ayant un niveau d'éducation de niveau secondaire, employé, dans la tranche de revenu moyen, déclarant effectuer un trajet de 45 minutes et ayant rapporté un niveau de conscience environnementale de 5 (sur 10). Nous pouvons prédire la probabilité qu'il utilise le vélo pour son trajet le plus fréquent en utilisant la formule suivante :

$$\text{logit}(p) = -2,497 + 1 * 0,372 + 30 * -0,009 + 1 * 0,193 + 1 * 0,042 + 1 * 0,273 + 45 * -0,001 + 5 * 0,102$$

$$p = \exp(-2,497 + 1 * 0,372 + 30 * -0,009 + 1 * 0,193 + 1 * 0,042 + 1 * 0,273 + 45 * -0,001 + 5 * 0,102) / (1 + \exp(-2,497 + 1 * 0,372 + 30 * -0,009 + 1 * 0,193 + 1 * 0,042 + 1 * 0,273 + 45 * -0,001 + 5 * 0,102)) = 0,194$$

Il y aurait donc 19,4% de chances pour que cette personne soit un cycliste. Cette probabilité dépasse le seuil que nous avons sélectionnée, cet individu serait donc classé comme un cycliste. Si on augmente son âge de 15 ans, nous obtenons :

$$p = \exp(-2,497 + 1 * 0,372 + 45 * -0,009 + 1 * 0,193 + 1 * 0,042 + 1 * 0,273 + 45 * -0,001 + 5 * 0,102) / (1 + \exp(-2,497 + 1 * 0,372 + 45 * -0,009 + 1 * 0,193 + 1 * 0,042 + 1 * 0,273 + 45 * -0,001 + 5 * 0,102)) = 0,174$$

Soit une réduction de 2 points de pourcentages. Il est également possible de représenter cette évolution sur un graphique pour montrer l'effet sur l'étendue des valeurs possibles. Sur ces graphiques des effets marginaux, il est essentiel de représenter notre incertitude quant à notre prédiction. En temps normal, la fonction `predict` calcule directement l'erreur standard de la prédiction et cette dernière peut être utilisée pour calculer l'intervalle de confiance de la prédiction. Cependant, nous voulons ici utiliser nos erreurs standards robustes. Nous devons donc procéder par simulation pour déterminer l'intervalle de confiance à 95% de nos prédictions. Cette opération nécessite de réaliser plusieurs opérations manuellement dans R.

```
# créer un jeu de données fictif pour la prédiction
mat <- model.matrix(model2$terms, model2$model)
age2seq <- seq(20,80)
mat2 <- matrix(mat[1,], nrow=length(age2seq), ncol=length(mat[1,]), byrow=TRUE)
colnames(mat2) <- colnames(mat)
mat2[, "Age"] <- age2seq
mat2[, "PaysBelgium"] <- 0
mat2[, "Duree"] <- 45
mat2[, "ConsEnv"] <- 5
mat2[, "StatutEmployeans emploi"] <- 0
mat2[, "Residencegrande ville"] <- 1
mat2[, "Educationsecondaire"] <- 1
mat2[, "Sexehomme"] <- 1
mat2[, "Revenumoyen"] <- 1
mat2[, "Revenufaible"] <- 0

# calculer la prédiction comme un log de rapport de cote (avec les erreurs standards)
# en multipliant les coefficient par les valeurs des données fictives
coeffs <- model2$coefficients
pred <- coeffs %*% t(mat2)

# simulation de prédictions (toujours en log de rapport de cote)

# étape 1 : simuler 1000 valeurs pour chaque coefficient
sim_coeffs <- lapply(1:length(coeffs), function(i){
  coef <- coeffs[[i]]
  std.err <- stdErrRobuste[[i]]
  vals <- rnorm(n = 1000, mean = coef, sd = std.err)
  return(vals)
})
mat_sim_coeffs <- do.call(rbind, sim_coeffs)

# étape 2 : effectuer les prédictions à partir des coefficients simulés
sim_preds <- lapply(1:ncol(mat_sim_coeffs), function(i){
  temp_coefs <- mat_sim_coeffs[, i]
  temp_pred <- as.vector(temp_coefs %*% t(mat2))
  return(temp_pred)
})

mat_sim_preds <- do.call(cbind, sim_preds)

# étape 3 : extraire les intervalles de confiance pour les simulations
intervals <- apply(mat_sim_preds, MARGIN = 1, FUN = function(vec){
  return(quantile(vec, probs = c(0.05, 0.95)))
})
```

```

# etape 4 : recuperer tous ces elements dans un dataframe
df <- data.frame(
  Age = seq(20,80),
  pred = as.vector(pred),
  lower = as.vector(intervals[1,]),
  upper = as.vector(intervals[2,])
)

# etape 5 : appliquer l'inverse de la fonction de lien pour
# obtenir les predictions en termes de probabilite
ilink <- family(model2)$linkinv

df$prob_pred <- ilink(df$pred)
df$prob_lower <- ilink(df$lower)
df$prob_upper <- ilink(df$upper)

# etape 6 : representer le tout sur un graphique
ggplot(df) +
  geom_ribbon(aes(x = Age, ymax = prob_upper, ymin = prob_lower), fill = rgb(0.1,0.1,0.1,0.4)) +
  geom_path(aes(x = Age, y = prob_pred), color = "blue", size = 1) +
  geom_hline(yintercept = 0.15, linetype = "dashed", size = 0.7) +
  labs(x = "Âge", y = "Probabilité prédictée (intervale de confiance 5% - 95%)")

```

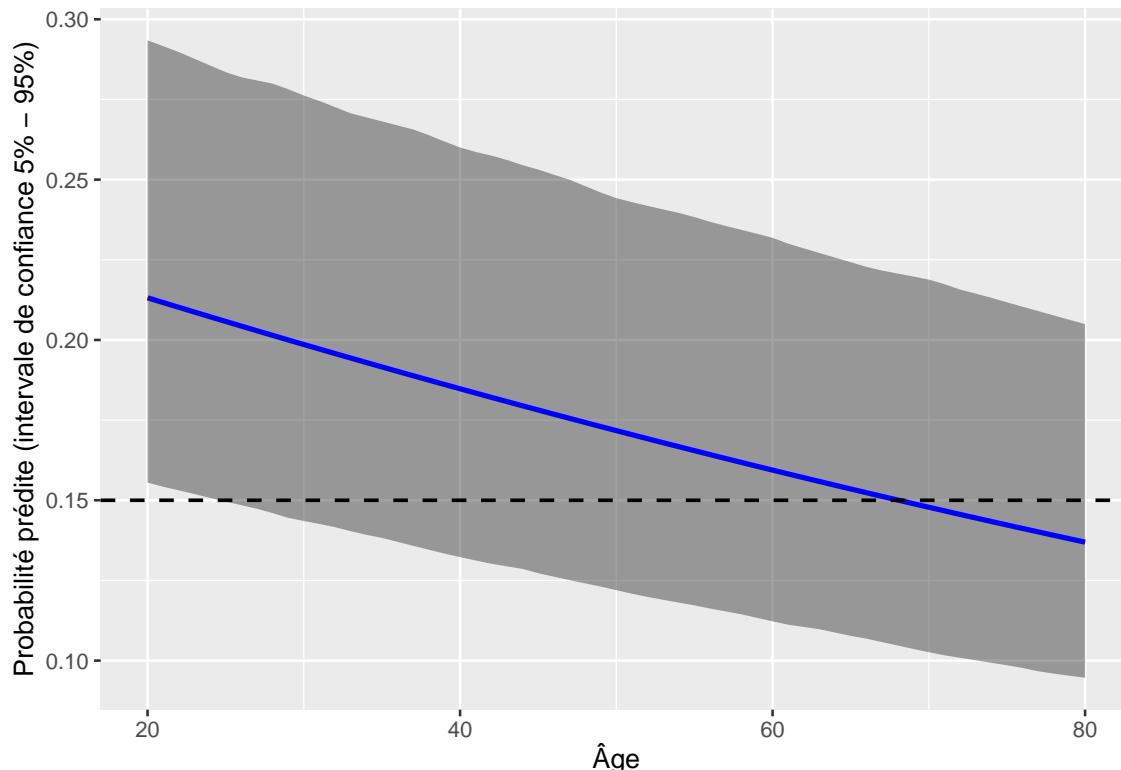


FIG. 1.11 : Impact de l'âge sur la probabilité d'utiliser le vélo comme moyen de déplacement pour son trajet le plus fréquent

La figure 1.11 permet de bien constater la diminution de la probabilité d'utiliser le vélo pour son trajet le plus fréquent avec l'âge, mais cette réduction est relativement ténue. Dans le cas utilisé en exemple, l'individu ne serait plus classé cycliste qu'après 67 ans.

1.2.1.2 Le modèle probit binomial

Le modèle GLM probit binomial est pour ainsi dire le frère du modèle logistique binomial. La seule différence entre les deux réside dans l'utilisation de deux fonction de liens différentes probit et logistique. La fonction de lien probit (Φ) correspond à la fonction cumulative de la distribution normale et a également une forme de S. Cette version du modèle est plus souvent utilisée par les économistes. Le principal changement réside dans l'interprétation des coefficients β_0 et β . Du fait de la transformation probit, ces derniers indiquent le changement en termes de scores Z de la probabilité modélisée. Vous conviendrez qu'il ne s'agit pas d'une échelle très intuitive, la plupart du temps, seuls la significativité et le signe (positif ou négatif) des coefficients sont interprétés.

1.2.1.3 Le modèle logistique des cotes proportionnelles

Le modèle logistique des cotes proportionnelles (aussi appelé modèle logistique cumulatif) est utilisé pour modéliser une variable qualitative ordinale. Un bon exemple de ce type de variable est une échelle de satisfaction : très insatisfait, satisfait, mitigé, insatisfait, très insatisfait, qui peut être recodée avec des valeurs numériques : 4, 3, 2, 1, 0 (ces échelons étant notés j). Il n'existe pas à proprement parler de distribution pour représenter ces données, mais avec une petite astuce, il est possible de simplement utiliser la distribution binomiale. Cette astuce est l'hypothèse de la proportionnalité des cotes. Cette hypothèse suppose que le passage de la catégorie 0 à la catégorie 1 est proportionnel au passage de la catégorie 1 à la catégorie 2 et ainsi de suite. Si cette hypothèse est respectée, alors les coefficients du modèle pourront autant décrire le passage de la catégorie satisfait à très satisfait que de la catégorie insatisfait à mitigé. Si cette hypothèse n'est pas respectée, il faudrait des coefficients différents pour représenter les passages d'une cagoterie à l'autre (ce qui est le cas pour le modèle multinomial présenté juste après).

Ainsi, dans le modèle logistique binomial vu précédemment on modélise la probabilité d'observer un événement $P(Y = 1)$. Dans un modèle logistique ordinal, on modélise la probabilité cumulative d'observer l'échelon j de notre variable ordinale $P(Y \leq j)$. L'intérêt de cette reformulation est que l'on conserve la facilité d'interprétation du modèle logistique binomial classique avec les rapports de cotes, à ceci près qu'ils représentent maintenant la probabilité de passer à un échelon supérieur de Y . La différence pratique est que notre modèle se retrouve avec autant d'intercepts qu'il n'y a de catégorie à prédire moins un, chacun de ces intercepts contrôlant pour la probabilité de base de passer de la catégorie j à la catégorie $j+1$.

1.2.1.3.1 Conditions d'application

Les conditions d'application sont les mêmes que pour un modèle binomial, avec bien sûr l'ajout de l'hypothèse sur la proportionnalité des cotes. Selon cette hypothèse, l'impact de chaque variable

TAB. 1.11 : Carte d'identité du modèle probit binomial

Type de variable dépendante	Variable binaire (0 ou 1) ou comptage de réussite à une expérience (ex : 3 réussites sur 5 expériences)
Distribution utilisée	Binomiale
Formulation	$Y \sim Binomial(p)$ $g(p) = \beta_0 + \beta X$ $g(x) = \Phi^{-1}(x)$
Fonction de lien	probit
Paramètre modélisé	p
Paramètres à estimer	β_0, β
Conditions d'applications	Non-séparation complète, Absence de surdispersion ou sousdispersion

TAB. 1.12 : Carte d'identité du modèle logistique des cotes proportionnelles

Type de variable dépendante	Variable qualitative ordonnées avec j catégories
Distribution utilisée	Binomiale
Formulation	$Y \sim Binomial(p)$ $g(p \leq j) = \beta_0 j + \beta X$ $g(x) = \log(\frac{x}{1-x})$
Fonction de lien	logistique
Paramètre modélisé	p
Paramètres à estimer	β et $j-1$ constantes β_0
Conditions d'applications	Non-séparation complète, Absence de surdispersion ou sousdispersion, Proportionnalité des cotes

indépendante est identique sur la probabilité de passer d'un échelon de la variable Y au suivant. Afin de tester cette condition, deux approches sont envisageables :

1. Utiliser l'approche de Brant (Brant, 1990). Il s'agit d'un test statistique comparant les résultats du modèle ordinal avec ceux d'une série de modèles logistiques binomials (1 pour chaque catégorie possible de Y).
2. Ajuster un modèle ordinal sans l'hypothèse de proportionnalité des cotes et effectuer un test de ratio des likelihood pour vérifier si le premier est significativement mieux ajusté.

Si certaines variables ne respectent pas cette condition d'application, trois options sont possibles pour y remédier :

1. Supprimer la variable du modèle (à éviter si cette variable est importante dans votre cadre théorique)
2. Autoriser la variable à avoir un effet différent entre chaque pallier (possible avec le package **VGAM**).
3. Changer de modèle et opter pour un modèle des catégories adjacentes. Il s'agit du cas particulier où toutes les variables sont autorisées à changer à chaque niveau. Ne pas confondre ce dernier modèle et le modèle multinomial que nous présentons ensuite, le modèle des catégories adjacentes continue à prédire la probabilité de passer à une catégorie supérieure.

1.2.1.3.2 Exemple appliqué dans R

Pour cet exemple, nous allons analyser un jeu de données proposé par Inside Airbnb, une organisation sans but lucratif collectant des données des annonces sur le site d'Airbnb pour alimenter le débat sur l'impact de cette société sur les quartiers. Plus spécifiquement, nous utilisons le jeu de données pour Montréal, compilé le 30 juin 2020². Nous modélisons ici le prix par nuit des logements, ce type d'exercice est appelé modélisation hédonique. Il est particulièrement utilisé en économie urbaine pour évaluer les déterminants du marché immobilier et prédire son évolution. Le cas de Airbnb a déjà été étudié dans plusieurs articles (Teubner et al., 2017, ; Wang and Nicolau, 2017, ; Zhang et al., 2017), il en ressort notamment que le niveau de confiance inspiré par l'hôte, les caractéristiques intrinsèques du logement et sa localisation sont les principaux prédicteurs de son prix. Nous construirons donc notre modèle sur cette base. Notez que nous avons décidé de retirer les logements avec des prix supérieur à 250\$ qui constituent des cas particuliers et qui devraient faire l'objet d'une analyse à part entière. Nous avons également retiré les observations pour lesquelles certaines données sont manquantes, et obtenons un nombre final de 9051 observations.

La distribution originale du prix des logements dans notre jeu de données est présentée à la figure 1.12.

²<http://insideairbnb.com/get-the-data.html>

```
# Charger le jeu de données
data_airbnb <- read.csv("data/glm/airbnb_data.csv")

# Afficher la distribution du prix
ggplot(data = data_airbnb) +
  geom_histogram(aes(x = price), bins = 30, color = "white", fill = "#1d3557", size = 0.02)
```

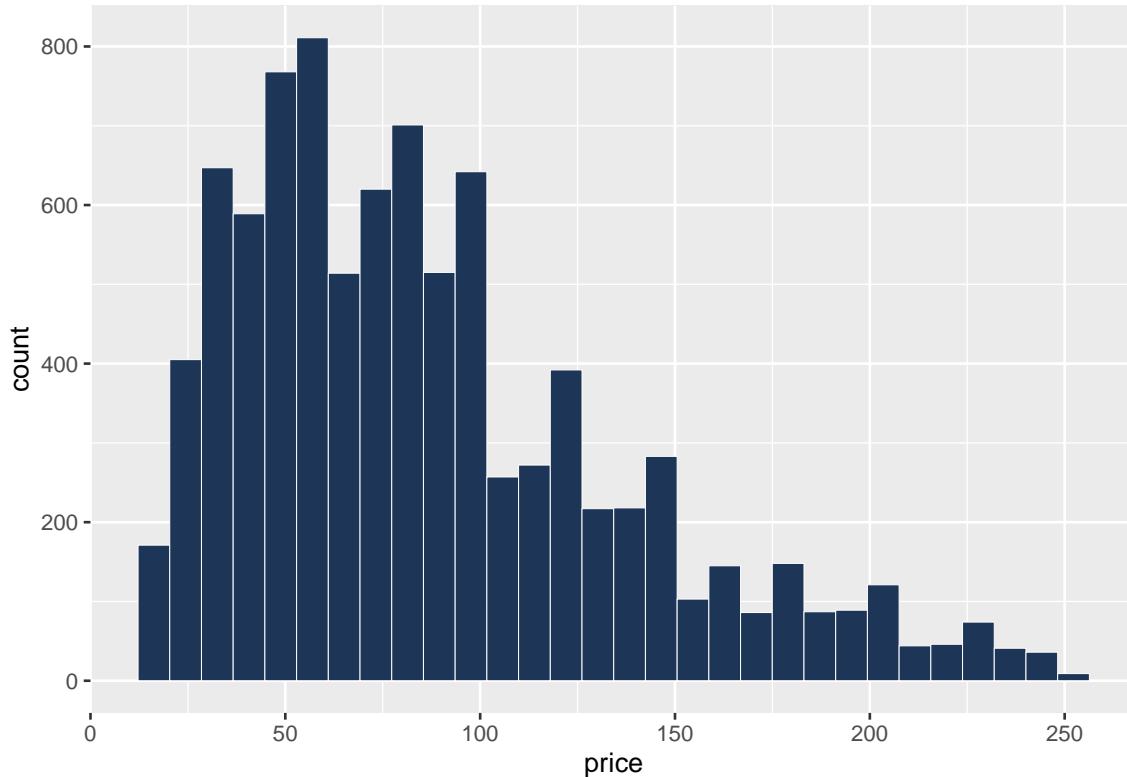


FIG. 1.12 : Distribution des prix des logements Airbnb

Nous avons ensuite découpé le prix des logements en trois catégories : inférieur à 50\$, entre 50\$ et 100\$ et entre 100\$ et 250\$. Ces catégories forment une variable nominale de trois échelons que nous modélisons à partir de trois catégories de variables :

- Les caractéristiques propres au logement
- Les caractéristiques environnementales autour du logement
- Les notes obtenues par le logement sur le site d'Airbnb

```
# afficher le nombre de logement par catégories
table(data_airbnb$fac_price_cat)
```

```
##  
##      1     2     3  
## 2212 3911 2928
```

Le tableau 1.13 suivant présente l'ensemble des variables utilisées dans le modèle.

Vérification des conditions d'applications

Avant d'ajuster le modèle, nous commençons par vérifier si nos variables indépendantes ne sont pas

TAB. 1.13 : Variable indépendantes utilisées pour prédire la catégorie de prix de logements Airbnb

Nom de la variable	signification	Type de variable	Mesure
Lit	Nombre de lit dans le logement	Variable de comptage	Nombre de lit dans le logement
Jardin	Présence d'un jardin ou d'une arrière cours	Variable binaire	Oui ou Non
Privee	Le logement est entièrement à disposition du locataire ou seulement une pièce	Variable binaire	Privé ou Partagé
Parking	Une place de parking gratuite est disponible sur la rue	Variable binaire	Oui ou Non
Accueil	L'hôte accueille personnellement les locataires	Variable binaire	Oui ou Non
Vegetation	Végétation dans les environs du logement	Variable continue	Pourcentage de surface végétale dans un rayon de 500m autour du logement
Metro	Présence d'une station de métro à proximité du logement	Variable binaire	Présence d'une station de métro dans un rayon de 500m autour du logement
Commercial	Commerce dans les environs du logement	Variable continue	Pourcentage de surface dédiée au commerce (mode d'occupation du sol) dans un rayon de 1 km autour du logement
Note	Évaluation de la qualité du logement par les usagers	Variable ordinaire	Score obtenu par le logement sur une échelle allant de 1 (très mauvais) à 5 (parfait)
Propriétaire	Nombre total de logements détenus par l'hôte sur Airbnb	Variable de comptage	Nombre total de logements détenus par l'hôte sur Airbnb

marquées par une multicolinéarité excessive.

```
# notez que la fonction vif ne s'intéresse qu'au variable indépendante
# vous pouvez donc utiliser la fonction glm avec la fonction vif quelque
# soit le glm que vous construisez
vif(glm(price ~ beds +
  Garden_or_backyard + Host_greets_you + Free_street_parking +
  prt_veg_500m + has_metro_500m + commercial_1km +host_total_listings_count +
  private + cat_review, data = data_airbnb))
```

##	beds	Garden_or_backyard	Host_greets_you
##	1.123595	1.079324	1.046884
##	Free_street_parking	prt_veg_500m	has_metro_500m
##	1.142189	1.532536	1.239516
##	commercial_1km	host_total_listings_count	private
##	1.225301	1.058232	1.143932
##	cat_review		
##	1.015778		

L'ensemble des valeurs de VIF sont inférieures à 2 indiquant une absence de multicolinéarité. Nous pouvons donc à présent ajuster le modèle et passer à l'analyse des distances de Cook. Pour ajuster le modèle, nous utilisons le package **VGAM** et la fonction **vglm** qui nous donnent accès à la famille cumulative pour ajuster des modèles logistiques ordinaires.

```
library(VGAM)

modele <- vglm(fac_price_cat ~ beds +
  Garden_or_backyard + Free_street_parking +
```

```

prt_veg_500m + has_metro_500m + commercial_1km +
private + cat_review + host_total_listings_count ,
family = cumulative(link="logitlink", # fonction de lien
parallel = TRUE, # cote proportionnelle
reverse = TRUE),
data = data_airbnb, model = T)

```

Notez que puisque notre variable Y a trois catégories différentes et que nous modélisons la probabilité de passer à une catégotie supérieure, chaque observation a deux (3-1) valeurs de résidus différentes. Nous pouvons donc calculer deux distances de Cook différentes que nous devons analyser conjointement. Malheureusement, la fonction `cook.distance` ne fonctionne pas avec les objets `vglm`, nous devons donc les calculer manuellement.

```

# extraction des residus
res <- residuals(modele, type = "pearson")
# extraction de la hat matrix (necessaire pour calculer la distance de Cook)
hat <- hatvaluesvlm(modele)

# calcul des distances de Cook
cooks <- lapply(1:ncol(res), function(i){
  r <- res[,i]
  h <- hat[,i]
  cook <- (r/(1 - h))^2 * h/(1 * modele@rank)
})

# structuration dans un dataframe
matcook <- data.frame(do.call(cbind, cooks))
names(matcook) <- c("dist1","dist2")
matcook$oid <- 1:nrow(matcook)

# afficher les distances de Cook
plot1 <- ggplot(data = matcook) +
  geom_point(aes(x = oid, y = dist1), size = 0.2, color = rgb(0.1,0.1,0.1,0.4)) +
  labs(x = "", y = "", subtitle = "distance de Cook P(Y>=2)")+
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())

plot2 <- ggplot(data = matcook) +
  geom_point(aes(x = oid, y = dist2), size = 0.2, color = rgb(0.1,0.1,0.1,0.4)) +
  labs(x = "", y = "", subtitle = "distance de Cook P(Y>=3)")+
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())

ggarrange(plot1, plot2, ncol = 2, nrow = 1)

```

Les distances de Cook nous permettent d'identifier quelques potentiels outliers, mais ils semblent être différents d'un graphique à l'autre. Nous décidons donc de ne pas retirer d'observations à ce stade et de passer à l'analyse des résidus simulés. Pour effectuer des simulations à partir de ce modèle, nous nous basons sur les probabilités d'appartenances prédites par le modèle

```

# extraire les probabilites predite
predicted <- predict(modele,type = "response")

```

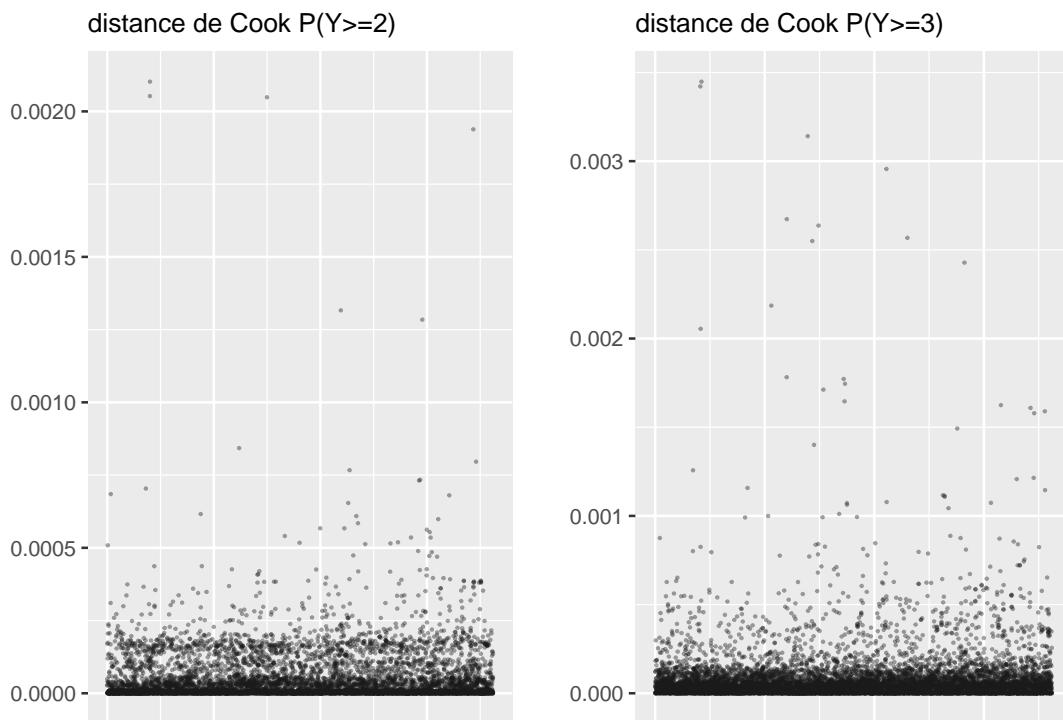


FIG. 1.13 : Distances de Cook pour le modèle logistique des cotes proportionnelles

```
round(head(predicted,n = 4),3)
```

```
##      1     2     3
## 1 0.706 0.267 0.028
## 2 0.073 0.461 0.466
## 3 0.687 0.283 0.030
## 4 0.049 0.383 0.568
```

On constate ainsi que pour la première observation, la probabilité prédictée d'appartenir au groupe 1 est de 69,4%, 27,7% pour le groupe 2 et 2,9% pour le groupe 3. Si nous effectuons 1000 simulations, on peut s'attendre à ce qu'en moyenne, sur ces 1000 simulations, 694 indiqueront 1 comme catégorie prédictée, 277 indiqueront 2 et seulement 29 indiqueront 3.

```
# Nous effectuerons 1000 simulations
nsim <- 1000

# lancement des simulations pour chaque observation (lignes dans predicted)
simulations <- lapply(1:nrow(predicted), function(i){
  probs <- predicted[i,]
  sims <- sample(c(1,2,3), size = nsim, replace = T, prob = probs)
  return(sims)
})

# combiner les predictions dans un tableau
matsim <- do.call(rbind, simulations)

# observons si nos simulations sont proches de ce que nous attendions
table(matsim[1,])
```

```
##  
##   1   2   3  
## 724 248 28
```

À partir de ces simulations de prédition, nous pouvons entamer le diagnostic des résidus simulés grâce au package **DHARMA**

```
# extraction de la prédiction moyenne du modèle  
pred_cat <- unique(data_airbnb$fac_price_cat)[max.col(predicted)]  
  
# préparer les données avec le package DHARMA  
sim_res <- createDHARMA(simulatedResponse = matsim,  
                           observedResponse = as.numeric(data_airbnb$fac_price_cat),  
                           fittedPredictedResponse = as.numeric(pred_cat),  
                           integerResponse = T)  
  
# Afficher le graphique de diagnostic général  
plot(sim_res)
```

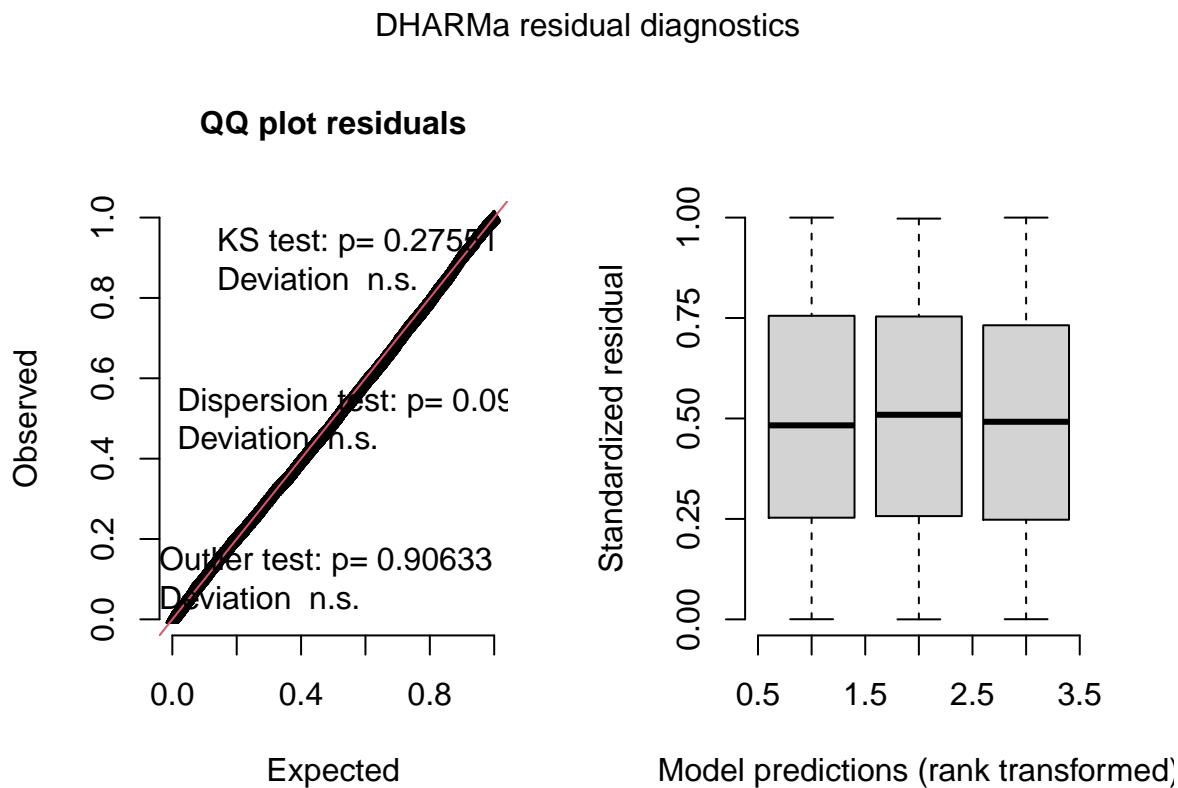


FIG. 1.14 : Diagnostic général des résidus simulés du modèle des cotes proportionnelles

La figure 1.14 nous indique que les résidus simulés suivent bien une distribution uniforme et qu'aucune valeur aberrante n'est observable. Pour affiner notre diagnostic, nous vérifions également si aucune relation ne semble exister entre chaque variable indépendante et les résidus.

```
# préparons un plot multiple
par(mfrow=c(3,4))
vars <- c("beds", "Garden_or_backyard", "Host_greets_you",
        "Free_street_parking", "prt_veg_500m",
        "has_metro_500m", "commercial_1km", "private",
        "cat_review", "host_total_listings_count")

for(v in vars){
  plotResiduals(sim_res, data_airbnb[[v]], main = v)
}
```

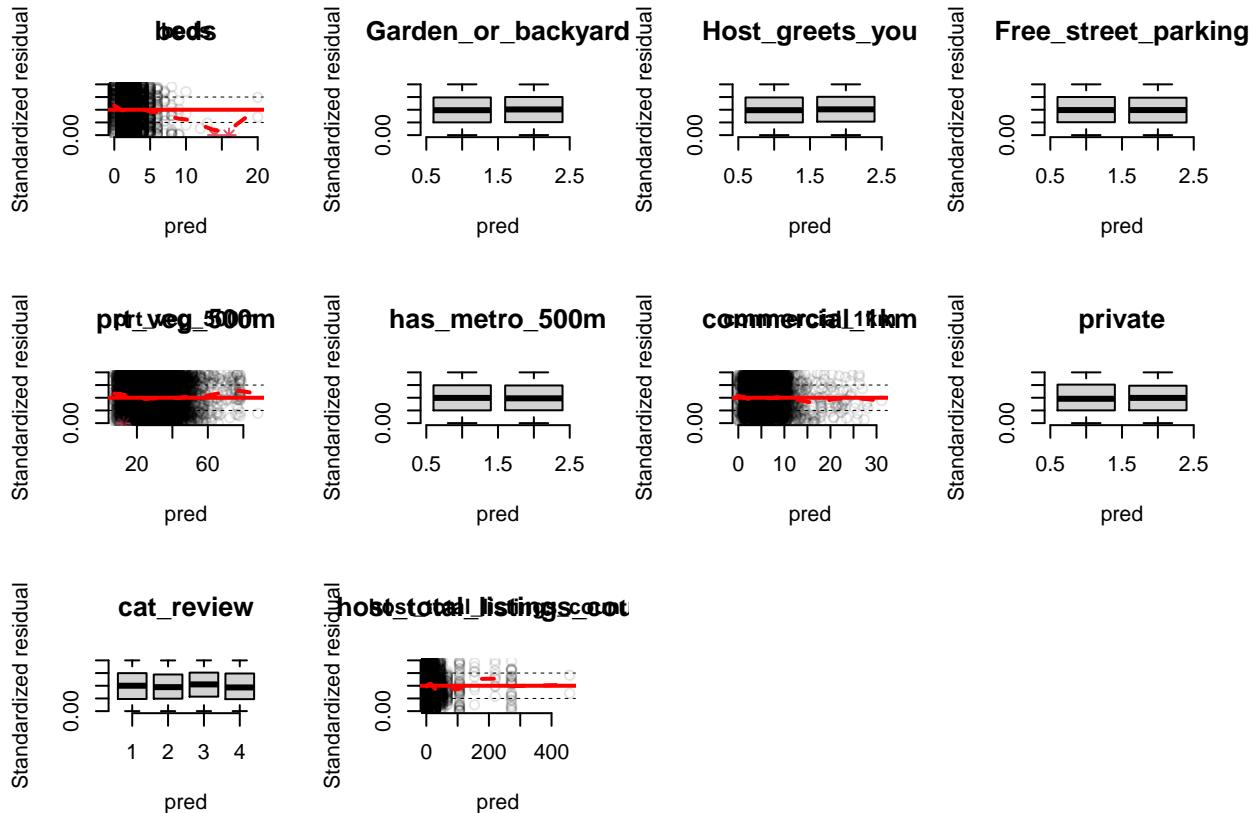


FIG. 1.15 : Diagnostic variable par variable des résidus simulés du modèle des cotés proportionnelles

La figure 1.15 indique qu'aucune relation marquée n'existe entre nos variables indépendantes et nos résidus simulés, sauf pour la variable nombre de lits. En effet, nous pouvons constater que les résidus ont tendance à être toujours plus faible quand le nombre de lit augmente. Cet effet est sûrement lié au fait qu'au delà de cinq lits, le logement en question est vraisemblablement un dortoir. Il pourrait être judicieux de retirer ces observations de l'analyse, considérant qu'elles sont peu nombreuses et constituent une forme de logement particulière.

```
data_airbnb2 <- subset(data_airbnb, data_airbnb$beds <= 5)

modele2 <- vglm(fac_price_cat ~ beds +
                  Garden_or_backyard + Free_street_parking +
                  prt_veg_500m + has_metro_500m + commercial_1km +
                  private + cat_review + host_total_listings_count ,
                  family = cumulative(link="logitlink", # fonction de lien
```

```
parallel = TRUE, # cote proportionnelle
reverse = TRUE),
data = data_airbnb2, model = T)
```

Nous pouvons ensuite recalculer les résidus simulés pour observer si cette tendance a été corrigée. La figure 1.16 montre qu'une bonne partie du problème a été corrigé, cependant il semble tout de même que les résidus soient plus fort pour les logements avec un seul lit..

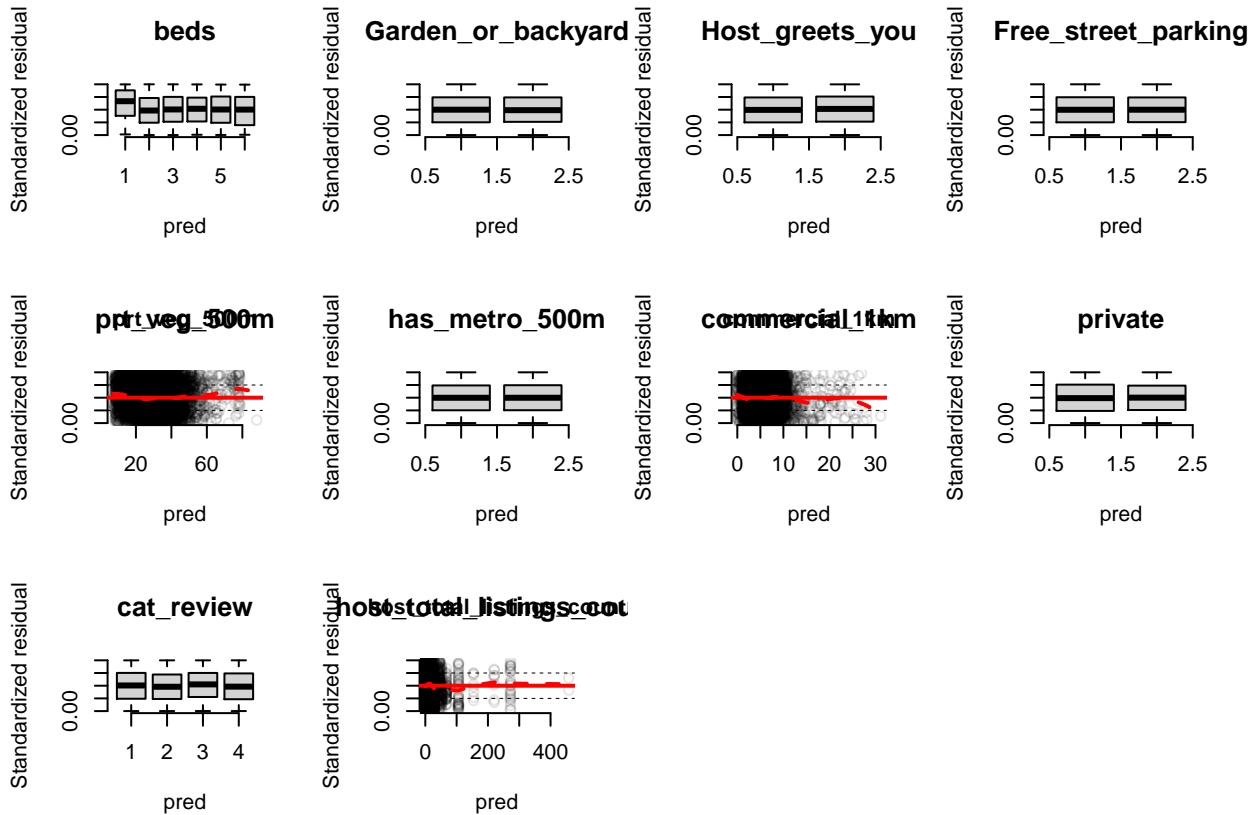


FIG. 1.16 : Diagnostic variable par variable des résidus simulés du modèle des cotes proportionnelles (après correction)

La prochaine étape du diagnostic est de vérifier si nous n'avons pas de séparation parfaite provoquée par un de nos prédicteurs. Le package **VGAM** propose pour cela la fonction `hdeff`.

```
tests <- hdeff(modele2)
problem <- tests[tests == TRUE]
problem
```

```
## named logical(0)
```

La fonction nous informe qu'aucun de nos prédicteurs ne provoque de séparation parfaite : toutes les valeurs renvoyées par la fonction `hdeff` sont égales à `FALSE`.

Il ne nous reste donc plus qu'à vérifier que l'hypothèse de proportionnalité des cotes est respectée, soit que l'impact de chacune des variables indépendante est bien le même pour passer de la catégorie 1 à 2 que pour passer des catégories 2 à 3. Pour cela, deux approches sont possibles : le test de Brant ou la réalisation d'une séquence de test de rapport de vraisemblance.

Le package **brant** propose une implémentation du test de Brant, mais elle ne peut être appliquée qu'à des modèles construits avec la fonction **polr** du package **MASS**. Nous avons donc récupérer le code source de la fonction **brant** du package **brant** et apporté quelques modifications pour qu'elle soit utilisable sur un objet **vglm**. Cette nouvelle fonction appelée **brant.vglm** est disponible dans le code source de ce livre.

```
tableau_brant <- round(brant.vglm(modele2), 3)
```

```
## -----
## Test for      X2  df  probability
## -----
## Omnibus       113.72  9   0
## beds          0.63   1   0.43
## Garden_or_backyard YES 0.16   1   0.69
## Free_street_parking YES 0.84   1   0.36
## prt_veg_500m   6.49   1   0.01
## has_metro_500m YES 0.02   1   0.89
## commercial_1km YES 0.01   1   0.92
## privateEntier 93.56   1   0
## cat_review     0.63   1   0.43
## host_total_listings_count 1.51   1   0.22
## -----
## H0: Parallel Regression Assumption holds
```

Ce premier tableau nous indique que seule la variable indiquant si le logement est disponible en entier ou partagé contrevient à l'hypothèse de proportionnalité des cotes (la seule valeur de p significative). Pour confirmer cette observation, nous pouvons réaliser un ensemble de test de rapport de vraisemblance. Pour chaque variable du modèle, nous allons créer un second modèle dans lequel cette variable est autorisée à varier pour chaque catégorie et comparer les niveaux d'ajustement des modèles. Nous avons implémenté cette procédure dans la fonction **parallel.likelihoodtest.vglm** présente dans le code source de ce livre.

```
tableau_likelihood <- parallel.likelihoodtest.vglm(modele2)
```

```
## |
```

```
print(tableau_likelihood)
```

	variable	non_parallel	AIC	loglikelihood	p.val	loglikelihood	ratio	test
## 1	beds	15304	-7640				0.271	
## 2	Garden_or_backyard	15305	-7641				0.417	
## 3	Free_street_parking	15301	-7639				0.079	
## 4	prt_veg_500m	15296	-7636				0.007	
## 5	has_metro_500m	15303	-7640				0.191	
## 6	commercial_1km	15305	-7640				0.271	
## 7	private	15215	-7595				0.000	
## 8	cat_review	15306	-7641				0.430	
## 9	host_total_listings_count	15304	-7640				0.257	

Les résultats de cette seconde série de tests confirment les précédents, la variable concernant le type de

logement doit être autorisée à varier en fonction de la catégorie. Ce second tableau nous indique que la variable concernant la densité de végétation pourrait aussi être amenée à varier en fonction du groupe, mais ce changement a un effet très marginal (différence de AIC de seulement 8 points). Nous ajustons donc un nouveau modèle autorisant la variable `private` à changer en fonction de la catégorie prédite.

```
modele3 <- vglm(fac_price_cat ~ beds +
  Garden_or_backyard + Host_greets_you + Free_street_parking +
  prt_veg_500m + has_metro_500m + commercial_1km +
  private + cat_review + host_total_listings_count,
  family = cumulative(link="logitlink",
  parallel = FALSE ~ private ,
  reverse = TRUE),
  data = data_airbnb2, model = T)
```

Vérifier l'ajustement du modèle

Maintenant que toutes les conditions d'application ont été passées en revue, nous pouvons passer à la vérification de l'ajustement du modèle

```
modelefull <- vglm(fac_price_cat ~ 1,
  family = cumulative(link="logitlink",
  parallel = TRUE,
  reverse = TRUE),
  data = data_airbnb2, model = T)

rsqs(loglike.full = logLik(modele3),
  loglike.null = logLik(modelefull),
  full.deviance = deviance(modele3),
  null.deviance = deviance(modelefull),
  nb.params = modele3@rank,
  n = nrow(data_airbnb2)
)

## `$`deviance expliquée`
## [1] 0.2087098
##
## `$`MacFadden ajusté`
## [1] 0.2073552
##
## `$`Cox and Snell`
## [1] 0.3606848
##
## $Nagelkerke
## [1] 0.4085924
```

Le modèle final parvient à expliquer 21% de la déviance originale, il obtient un R² ajusté de MacFadden de 0,21, et des R² de Cox et Snell et Nagelkerke de respectivement 0,36 et 0,41. Construisons à présent la matrice de confusion de la prédiction du modèle (nous utilisons ici la fonction `nice_confusion_matrix` également disponible dans le code source de ce livre)

```
preds_probs <- fitted(modele3)
pred_cat <- c(1,2,3)[max.col(preds_probs)]
```

```
library(caret)
matrices <- nice_confusion_matrix(data_airbnb2$fac_price_cat,pred_cat)
```

afficher la matrice de confusion
`print(matrices$confusion_matrix)`

##	rowsnames	1	2	3	rs	rp
## colsnames	""	"1 (reel)"	"2 (reel)"	"3 (reel)"	"Total"	"%"
## 1	"1 (redit)"	"1576"	"736"	"168"	"2480"	"27.7"
## 2	"2 (redit)"	"579"	"2385"	"1331"	"4295"	"48"
## 3	"3 (redit)"	"49"	"771"	"1360"	"2180"	"24.3"
##	"Total"	"2204"	"3892"	"2859"	"8955"	NA
##	"%"	"24.6"	"43.5"	"31.9"	NA	NA

afficher les indicateurs de qualité de prédiction
`print(matrices$indicators)`

##	rnames	precision	rappel	F1
## 1	"1"	"0.64"	"0.72"	"0.67"
## 2	"2"	"0.56"	"0.61"	"0.58"
## 3	"3"	"0.62"	"0.48"	"0.54"
## macro_scores	"macro"	"0.6"	"0.59"	"0.59"
##	"Kappa"	"0.37"	NA	NA
##	"Valeur de p (précision > NIR)"	"0"	NA	NA

Le modèle a une précision totale de 61% (61% des observations ont été correctement prédictes). La catégorie 1 a de loin la meilleure précision (72%) et la 3 la pire (48%) ce qui indique qu'il manque vraisemblablement des variables indépendantes contribuant à prédire les prix des logements les plus chers. Le coefficient de Kappa indique un niveau d'accord entre modéré et faible, mais le modèle parvient à une prédiction significativement supérieure au seuil de non-information. Si l'ajustement du modèle est imparfait, il est suffisamment fiable pour nous donner des renseignements pertinents sur le phénomène étudié.

Interprétation des résultats

L'ensemble des coefficients du modèle sont accessibles via la fonction `summary`. À partir des coefficients et de leurs erreurs standards, il est possible de calculer les rapports de cotes ainsi que leurs intervalles de confiances.

```
tableau <- summary(modele3)$coeff
rappCote <- exp(tableau[,1])
rappCote2.5 <- exp(tableau[,1] - 1.96 * tableau[,2])
rappCote97.5 <- exp(tableau[,1] + 1.96 * tableau[,2])
tableau <- cbind(tableau, rappCote, rappCote2.5, rappCote97.5)
print(round(tableau,3))
```

##	Estimate	Std. Error	z value	Pr(> z)	rappCote
## (Intercept):1	-1.203	0.137	-8.768	0.000	0.300
## (Intercept):2	-3.240	0.151	-21.516	0.000	0.039
## beds	0.748	0.027	28.143	0.000	2.113
## Garden_or_backyardYES	0.120	0.067	1.795	0.073	1.128

```

## Host_greets_youYES      0.094    0.060   1.548    0.122   1.098
## Free_street_parkingYES -0.015    0.046  -0.320    0.749   0.985
## prt_veg_500m            -0.026   0.003 -10.182   0.000   0.975
## has_metro_500mYES       0.063    0.048   1.326    0.185   1.065
## commercial_1km           0.008    0.008   0.955    0.340   1.008
## privateEntier:1          2.445    0.062  39.241   0.000  11.533
## privateEntier:2          1.576    0.081  19.573   0.000   4.834
## cat_review                0.200    0.021   9.563   0.000   1.222
## host_total_listings_count -0.002   0.001  -2.094   0.036   0.998
##                                     rappCote2.5 rappCote97.5

## (Intercept):1             0.230    0.393
## (Intercept):2               0.029    0.053
## beds                      2.006    2.226
## Garden_or_backyardYES     0.989    1.287
## Host_greets_youYES        0.975    1.236
## Free_street_parkingYES    0.900    1.078
## prt_veg_500m               0.970    0.980
## has_metro_500mYES          0.970    1.170
## commercial_1km              0.992    1.024
## privateEntier:1            10.207   13.031
## privateEntier:2              4.128    5.660
## cat_review                  1.173    1.273
## host_total_listings_count   0.996    1.000

```

Pour faciliter la lecture des résultats, nous proposons le tableau X

TAB. 1.14 : Coefficients du modèle logistique des cotes proportionnelles

variable	coefficient	RC	val.p	RC 2,5%	RC 97,5%	sign.
(Intercept):1	-1.20	0.300	<0.001	0.230	0.395	***
(Intercept):2	-3.24	0.039	<0.001	0.029	0.052	***
beds	0.75	2.113	<0.001	2.014	2.226	***
Garden_or_backyard						
ref : NO	-	-	-	-	-	-
YES	0.12	1.128	0.073	0.990	1.284	.
Host_greets_you						
ref : NO	-	-	-	-	-	-
YES	0.09	1.098	0.122	0.980	1.234	
Free_street_parking						
ref : NO	-	-	-	-	-	-
YES	-0.01	0.985	0.749	0.896	1.083	
prt_veg_500m	-0.03	0.975	<0.001	0.970	0.980	***
has_metro_500m						
ref : NO	-	-	-	-	-	-
YES	0.06	1.065	0.185	0.970	1.174	
commercial_1km	0.01	1.008	0.340	0.990	1.020	
cat_review	0.20	1.222	<0.001	1.174	1.271	***
host_total_listings_count	<0.01	0.998	0.036	1.000	1.000	*
Effets par niveau						
<i>private</i>						
ref : Chambre	-	-	-	-	-	-
Entier :1	2.45	11.533	<0.001	10.176	13.066	***
Entier :2	1.58	4.834	<0.001	4.137	5.641	***

Sans surprise, chaque lit supplémentaire contribue à augmenter les chances que le logement soit dans une catégorie de prix supérieure (multiplication par deux à chaque lit supplémentaire). En revanche, la présence d'un parking gratuit, d'un jardin et l'accueil en personne par l'hôte n'ont pas d'impact significatifs. Comme l'indiquaient les articles mentionnés en début de section, les revues positives augmentent la probabilité d'appartenir à une catégorie supérieure de prix. Pour chaque point supplémentaire sur l'échelle de 1 à 5 augmente la probabilité d'appartenir à une catégorie de prix supérieure de 22,2\$. Il est intéressant de noter que le fait de disposer du logement entier plutôt que d'une simple chambre augmente davantage les chances de passer du groupe de prix 1 à 2 (multiplication par 2,45) que du groupe 2 à 3 (multiplication par 1,58). Il semblerait également que si l'hôte possède plusieurs logements, la probabilité d'avoir une classe de prix supérieure diminue légèrement. Cependant l'effet est trop petit pour pouvoir se livrer à des interprétations.

Les variables environnementales ont peu d'impact : le pourcentage de surface commerciale dans un rayon d'un kilomètre et la présence d'une station de métro ne sont pas significative. En revanche, une augmentation de la surface végétale dans un rayon de 500m tend à réduire la probabilité d'appartenir à une classe supérieure. Notre hypothèse concernant ce résultat est que cette variable représente un effet associé à la position des Airbnb, les plus centraux ayant tendance à être plus chers mais avec un environnement moins vert et inversement. Pour l'illustrer, prédisons les probabilités d'appartenance aux différents niveaux de prix d'un logement avec les caractéristiques suivantes : entièrement privé, 2 lits, un jardin, une place de parking gratuite, l'hôte ne dispose que d'un logement sur Airbnb et accueille les arrivant en personne, surface commercial dans un rayon de 1km : 10%, catégorie de revue : 2, absence de métro dans un rayon de 500m.

```
# créer un jeu de données pour effectuer des predictions
df <- data.frame(
  prt_veg_500m = seq(5,90),
  beds = 2,
  Garden_or_backyard = "YES",
  Host_greets_you = "YES",
  Free_street_parking = "YES",
  has_metro_500m = "NO",
  commercial_1km = 10,
  private = "Entier",
  cat_review = 2,
  host_total_listings_count = 1
)

# Effectuer les predictions (dans l'échelle log)
preds <- predict(modele3, newdata = df, type = "link", se.fit=T)

# Définir l'inverse de la fonction de lien
ilink <- function(x){exp(x)/(1+exp(x))}

# Calculer les probabilités et leurs intervalles de confiance
df[["P[Y>=2]]] <- ilink(preds$fitted.values[,1])
df[["P[Y>=2] 2,5%"]] <- ilink(preds$fitted.values[,1] - 1.96 * preds$se.fit[,1])
df[["P[Y>=2] 97,5%"]] <- ilink(preds$fitted.values[,1] + 1.96 * preds$se.fit[,1])

df[["P[Y>=3]]] <- ilink(preds$fitted.values[,2])
df[["P[Y>=3] 2,5%"]] <- ilink(preds$fitted.values[,2] - 1.96 * preds$se.fit[,2])
df[["P[Y>=3] 97,5%"]] <- ilink(preds$fitted.values[,2] + 1.96 * preds$se.fit[,2])

df[["P[Y=1]]] = 1-df[["P[Y>=2]]"]
```

```

df[["P[Y=1] 2,5" ]] = 1-df[["P[Y>=2] 2,5"]]
df[["P[Y=1] 97,5" ]] = 1-df[["P[Y>=2] 97,5"]]

# afficher les résultats

ggplot(data = df) +
  geom_ribbon(aes(x = prt_veg_500m,
                  ymin = `P[Y>=2] 2,5`,
                  ymax = `P[Y>=2] 97,5`), fill ="#f94144", alpha = 0.4) +
  geom_path(aes(x = prt_veg_500m, y = `P[Y>=2]`, color="Y2")) +
  geom_ribbon(aes(x = prt_veg_500m,
                  ymin = `P[Y>=3] 2,5`,
                  ymax = `P[Y>=3] 97,5`), fill ="#90be6d", alpha = 0.4) +
  geom_path(aes(x = prt_veg_500m, y = `P[Y>=3]`, color = "Y3" )) +
  geom_ribbon(aes(x = prt_veg_500m,
                  ymin = `P[Y=1] 2,5`,
                  ymax = `P[Y=1] 97,5`), fill ="#277da1" , alpha = 0.4) +
  geom_path(aes(x = prt_veg_500m, y = `P[Y=1]`, color = "Y1")) +
  scale_color_manual(name = "Probabilité prédictes",
                     breaks = c("Y1", "Y2", "Y3"),
                     labels = c("P[Y=1]", "P[Y>=2]", "P[Y>=3]"),
                     values = c("Y2" = "#f94144", "Y3" = "#90be6d",
                               "Y1" = "#277da1")) +
  labs(x = "Densité de végétation (%)",
       y = "Probabilité")

```

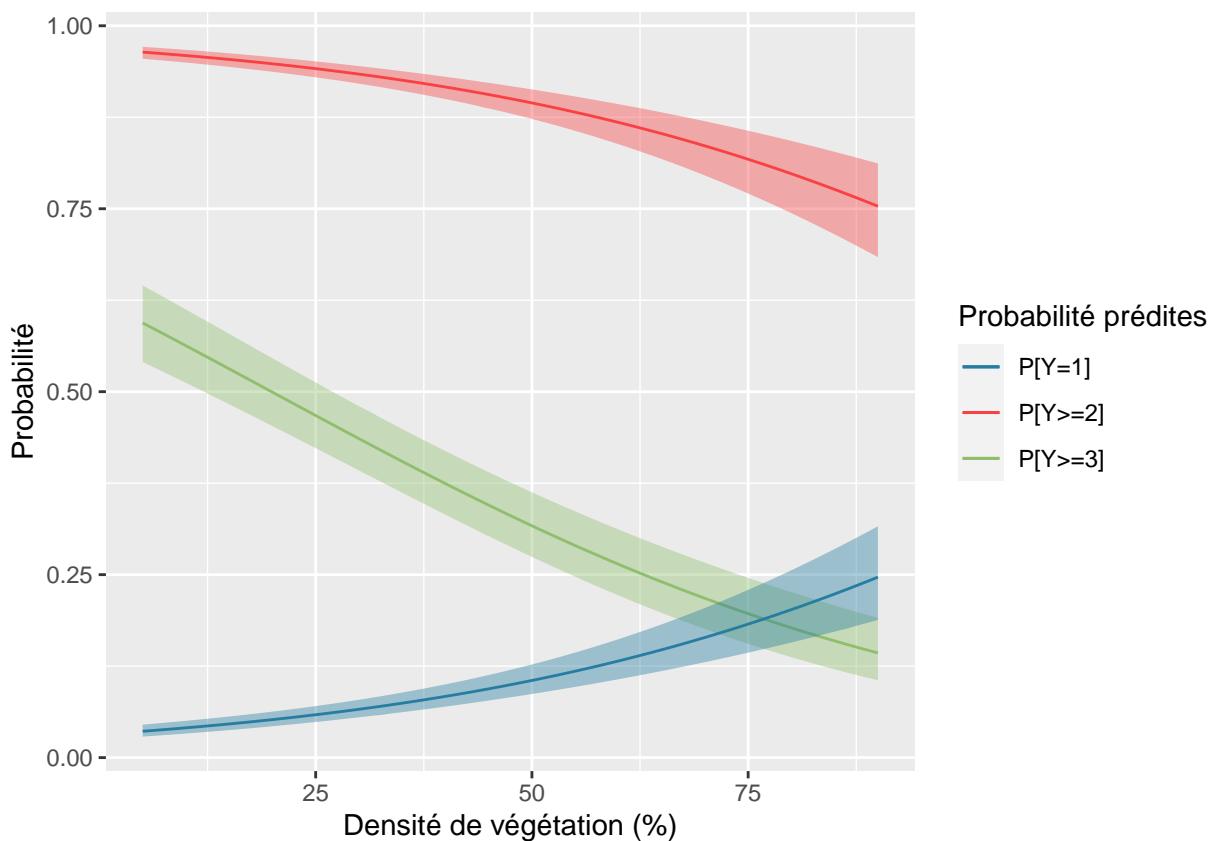


FIG. 1.17 : Prédiction de la probabilité d'appartenance aux trois catégories de prix en fonction de la densité de végétation

On constate sur la figure 1.17 effet que les probabilités d'appartenir aux niveaux 2 et 3 diminuent à mesure qu'augmente le pourcentage de végétation. La probabilité d'appartenir à la classe 2 et plus (en rouge), passe de plus de 95% en cas d'absence de végétation, à environ 75% avec 80% de végétation dans un rayon de 500m. Comme vous pouvez le constater, la probabilité $P[Y = 1]$ est la symétrie de $P[Y \geq 2]$ puisque $P[Y = 1] + P[Y \geq 2] = 1$.

1.2.1.4 Le modèle logistique multinomial

La régression logistique multinomiale est utilisée pour modéliser une variable Y qualitative multinomiale, c'est-à-dire une variable dont les catégories ne peuvent pas être ordonnées. Dans le modèle précédent, nous avons vu qu'il était possible de modéliser une variable ordinaire avec une distribution binomiale en formulant l'hypothèse de la proportionnalité des cotes. Avec une variable multinomiale, cette hypothèse ne tient plus car les catégories ne sont plus ordonnées. Il faut donc formuler le modèle différemment. L'idée derrière un modèle multinomial est de choisir une catégorie de référence, puis de modéliser les probabilités d'appartenir à chaque autre catégorie plutôt qu'à cette catégorie de référence. Si nous avons K catégories possibles dans notre variable Y , nous obtenons $K-1$ comparaisons. Chaque comparaison est modélisée avec sa propre équation ce qui génère de nombreux paramètres. Par exemple, admettons que notre variable Y ait 5 catégories et que nous disposons de 6 variables X prédictives. Nous avons ainsi 4 (5-1) équations de régression, multiplié par 7 paramètres (6 coefficients et un intercept), soit 28 coefficients à analyser. Considérant cette tendance à la multiplication des coefficients, il est fréquent de recourir à une méthode appelée Anlayse de type 3 pour limiter au maximum le nombre de variables indépendantes dans le modèle. L'idée de cette méthode est de recalculer plusieurs version du modèle dans lesquelles une variable indépendante est retirée, puis de réaliser un test de rapport de vraisemblance en comparant ce nouveau modèle au modèle complet pour vérifier si la variable en question améliore significativement le modèle. Il est alors possible de retirer toutes les variables dont l'apport est négligeable si elles étaient également peu intéressantes du point de vue théorique.

1.2.1.4.1 Conditions d'applications

Les conditions d'application sont les mêmes que pour un modèle binomial, avec l'ajout de l'hypothèse sur l'indépendance des alternatives non pertinentes. Cette dernière suppose que le choix entre deux catégories est indépendant des catégories proposées. Voici un exemple simple pour illustrer cette hypothèse : admettons que nous disposons d'une trentaine de personnes et que nous leur demandions la couleur de leurs yeux. Cette variable ne serait pas affectée par la présence de nouvelles couleurs. En revanche, si nous leur demandions de choisir un mode de transport parmi une liste pour se rendre à leur lieu de travail, leur réponse serait nécessairement affectée par la liste des modes de transport

TAB. 1.15 : Carte d'identité du modèle probit binomial

Type de variable dépendante	Variable qualitative multinomiale avec K catégories
Distribution utilisée	Binomiale
Formulation	$Y \sim Binomial(p)$ $g(p = k \text{ avec } ref = a) = \beta_{0k} + \beta X_k$ $g(x) = \frac{\log(x)}{1-x}$
Fonction de lien	logistique
Paramètre modélisé	p
Paramètres à estimer	β_{0k}, β_k pour $k \in [2, \dots, K]$
Conditions d'applications	Non-séparation complète, Absence de surdispersion ou sousdispersion, Indépendance des alternatives non pertinentes

disponibles. Les tests développés pour vérifier cette hypothèse sont connus pour leur faible fiabilité³. Il est plus pertinent de décider théoriquement si cette hypothèse est valide ou non. Dans le cas contraire, il est possible d'utiliser une classe de modèle logistique plus rare : le modèle logistique imbriqué.

Notez également que le grand nombre de paramètres dans ce type de modèle implique de disposer d'un plus grand nombre d'observation afin de disposer de suffisamment d'information pour ajuster tous les paramètres.

Pour vérifier la présence de surdispersion, il est possible dans le cas du modèle multinomial de calculer le rapport entre le Chi2 de Pearson et le nombre de degré de liberté du modèle. Si ce rapport est supérieur à 1 (des valeurs jusqu'à 1,2 ne sont pas problématiques) alors le modèle souffre de surdispersion (SAS Institute Inc, 2020a). Le Chi2 de Pearson permet de comparer les catégories observées dans la variable Y et les catégories prédites par le modèle. Il est possible de le calculer avec la formule suivante :

$$\chi^2 = \sum_k \frac{(O_k - E_k)^2}{E_k} \quad (1.17)$$

Avec O_k la proportion d'observations dans le groupe k et E_k la proportion de valeurs prédites dans le groupe k .

Le ratio est ensuite calculé comme suit ainsi : $\frac{\chi^2}{(N-p)(K-1)}$, Avec N le nombre d'observation et K le nombre de groupe dans la variable Y .



Le modèle logistique imbriqué

Du fait de sa proximité avec les modèles à effets mixtes que nous aborderons plus tard (section X), nous ne détaillons pas ici le modèle logistique imbriqué, mais présentons plutôt son principe général. Il s'agit d'une généralisation du modèle logistique multinomial basé sur l'idée que certaines catégories pourraient être regroupées dans des « nids » (nest en anglais). Dans ces groupes, les erreurs peuvent être corrélées, indiquant ainsi que si une catégorie est manquante, une autre catégorie du même groupe sera préférée. Un paramètre λ contrôle spécifiquement cette corrélation et permet donc de mesurer sa force une fois le modèle ajusté. Il peut donc être pertinent de comparer un modèle imbriqué à un modèle multinomial pour déterminer lequel des deux est le mieux ajusté aux données.

1.2.1.4.2 Exemple appliqué dans R

Pour cet exemple, nous reproduisons une partie de l'analyse effectuée dans l'étude de McFadden (2016). Cet article s'intéresse aux écarts entre les croyances des individus et les connaissances scientifiques sur les sujets des OGM et du réchauffement climatique. Les auteurs utilisent pour cela des données issue d'un enquête auprès de 961 individus formant un échantillon représentatif de la population des États Unis. Les données issue de cette enquête sont téléchargeable sur le site de l'éditeur⁴, ce qui nous permet ici de reproduire l'analyse effectuée par les auteurs. Deux questions sont centrales dans l'enquête : "Dans quelle mesure êtes vous en accord ou en désaccord avec la phrase suivante : Les plantations génétiquement modifiée sont sans danger pour la consommation" et "Dans quelle mesure êtes vous en accord ou en désaccord avec la phrase suivante : La Terre se réchauffe du fait des activités humaines". Pour ces deux questions, les répondants devaient sélectionner leur degré d'accord sur une échelle de Likert allant de 1 (fortement en désaccord) à 5 (fortement en accord). Les réponses à ces deux questions ont été utilisées pour former une variable multinomiale à quatre modalités :

- A : les individus étant en accord avec les deux propositions

³<https://statisticalhorizons.com/ia>

⁴<https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0166140>

- B : les individus en désaccord sur les OGM, mais en accord sur le réchauffement climatique
- C : les individus en accord sur les OGM, mais en désaccord sur le réchauffement climatique
- D : les individus en désaccord avec les deux propositions

Un modèle logistique multinomial a été utilisée pour déterminer quels facteurs contribuent à la probabilité d'appartenir à ces différentes catégories. Les variables indépendantes présentes dans le modèle sont détaillées dans le tableau 1.16. Les auteurs avaient notamment conclus que :

- Les effets des connaissances (réelles ou perçues) n'étaient pas uniforme et pouvaient varier en fonction du sujet.
- L'orientation politique avait une influence significative sur les croyances.
- Les répondants avec de plus hauts résultats au test de cognition CRT avaient plus souvent des opinions divergentes de la communauté scientifique.

Vérification des conditions d'applications

Avant d'ajuster le modèle, nous commençons par vérifier l'absence de multicolinéarité excessive entre nos variables indépendantes.

```
data_quest <- read.csv("data/glm/enquete_PublicOpinion_vs_Science.csv")

# choix des valeurs de références dans les facteurs
data_quest$Parti <- relevel(as.factor(data_quest$Parti), ref = "Democrate")
data_quest$Sexe <- relevel(as.factor(data_quest$Sexe), ref = "homme")

vif(glm(SCIGM ~ PercepOGM + PercepRechClim + ConnaisOGM + ConnaisRechClim +
        CRT + Parti + AGE + Sexe + Revenu, data = data_quest))

##                                     GVIF Df GVIF^(1/(2*Df))
## PercepOGM      1.092693  1     1.045320
## PercepRechClim 1.177495  1     1.085125
## ConnaisOGM     1.150662  1     1.072689
## ConnaisRechClim 1.158438  1     1.076307
```

TAB. 1.16 : Variable indépendantes utilisées dans le modèle logistique multinomial

Nom de la variable	signification	Type de variable	Mesure
PercepOGM	La recherche scientifique supporte ma vision sur la sécurité des plantes OGM	Variable ordinaire	Échelle de Likert de 1 (fortement en désaccord) à 5 (fortement en accord)
PercepRechClim	La recherche scientifique supporte ma vision sur le réchauffement climatique	Variable ordinaire	Échelle de Likert de 1 (fortement en désaccord) à 5 (fortement en accord)
ConnaisOGM	Niveau de connaissance sur les OGM	Variable ordinaire	Nombre de réponses juste sur 3 questions portant sur les OGM
ConnaisRechClim	Niveau de connaissance sur le réchauffement climatique	Variable ordinaire	Nombre de réponses juste sur 3 questions portant sur le réchauffement climatique
CRT	Score obtenu au Cognitive Reflection Test, utilisé pour déterminer la propension à faire preuve d'esprit d'analyse plutôt que choisir des réponses intuitives	Variable ordinaire	Nombre de réponses juste sur 3 questions pièges
Parti	Orientation politique du répondant	Variable multinomiale	Républicain, Démocrate et autre
Sexe	Sexe du répondant	Variable binaire	Femme ou Homme
Age	Âge du répondant	Variable continue	Âge du répondant
Revenu	Niveau de revenu du répondant	Variable ordinaire	Échelle de 1 (moins de 20 000) 8(140000 et plus)

```

## CRT           1.155371  1      1.074882
## Parti        1.130817  2      1.031212
## AGE          1.071655  1      1.035208
## Sexe         1.064918  1      1.031949
## Revenu       1.049499  1      1.024451

```

L'ensemble des valeurs de VIF sont inférieure à 2, indiquant bien une absence de multicolinéarité. La seconde étape est d'ajuster le modèle et de vérifier l'absence de sur ou sousdispersion. Pour ajuster le modèle, nous utilisons à nouveau la fonction `vglm` du package **VGAM**, avec le paramètre `family = multinomial()`.

```

# ajustement du modèle
modele <- vglm(Y ~ PercepOGM + PercepRechClim + ConnaisOGM + ConnaisRechClim +
                  CRT + Parti + AGE + Sexe + Revenu,
                  data = data_quest,
                  family = multinomial(refLevel="A"), model = T)

# calcule du Chi2 de Pearson
pred <- predict(modele, type= "response")
cat_predict <- colnames(pred)[max.col(pred)]

freq_real <- table(data_quest$Y)
freq_pred <- table(cat_predict)

chi2 <- sum(((freq_real - freq_pred)**2) / freq_pred)

N <- nrow(data_quest)
p <- modele@rank
r <- length(freq_real)
ratio <- chi2 / ((N-p)*(r-1))
print(ratio)

## [1] 0.1243563

```

Le ratio entre la statistique de Pearson et le nombre de degré de liberté du modèle n'indique pas de présence de surdispersion dans le modèle.

La prochaine étape de la vérification des conditions d'application est l'observation des distances de Cook. À nouveau, puisque le modèle évalue la probabilité d'appartenir à $K-1$ catégorie, nous pouvons calculer $K-1$ résidus par observations et par extension $K-1$ distances de Cook. Aucune observation ne semble se détacher nettement dans la figure 1.18. Nous décidons donc pour le moment de conserver toutes les observations.

```

# extraction des residus
res <- residuals(modele, type = "pearson")
# extraction de la hat matrix (necessaire pour calculer la distance de Cook)
hat <- hatvalues(lm(modele))

# calcul des distances de Cook
vals <- c("A","B","C","D")

cooks <- lapply(1:ncol(res),function(i){
  r <- res[,i]
  h <- hat[,i]

```

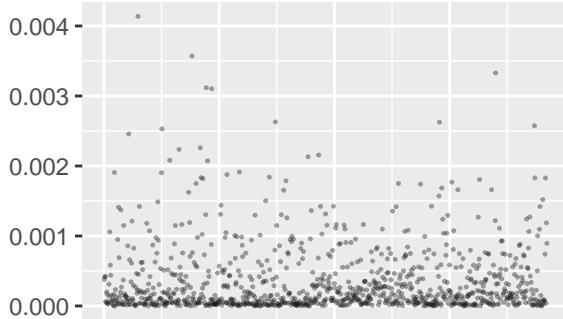
```

cook <- (r/(1 - h))^2 * h/(1 * modele@rank)
df <- data.frame(
  oid = 1:length(cook),
  cook = cook
)
plot <- ggplot(data = df) +
  geom_point(aes(x = oid, y = cook), size = 0.2, color = rgb(0.1,0.1,0.1,0.4)) +
  labs(x = "", y = "", subtitle = paste("distance de Cook P(",vals[[1]]," VS ",vals[[i+1]],")",sep="")) +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())
return(plot)
})

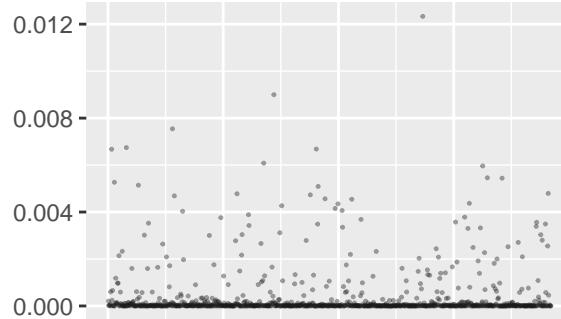
ggarrange(plotlist = cooks, ncol = 2, nrow = 2)

```

distance de Cook P(A VS B)



distance de Cook P(A VS C)



distance de Cook P(A VS D)

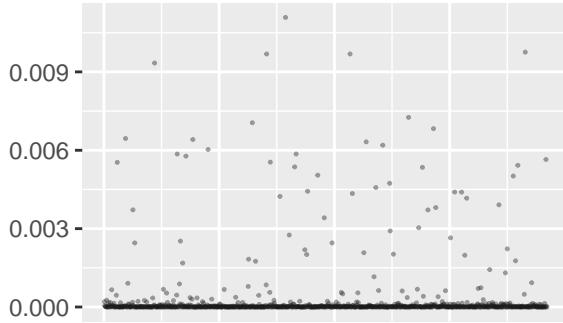


FIG. 1.18 : Distances de Cook pour le modèle logistique multinomial

Avant de passer à l'analyse de résidus simulés, il est pertinent de réaliser une analyse de type 3 afin de retirer les variables indépendantes dont l'apport au modèle est négligeable. La fonction `AnalyseType3` (disponible dans le code source de ce livre) permet d'effectuer cette opération automatiquement pour un objet de type `vglm`.

```
tableau <- AnalyseType3(modele, data_quest)
```

```
## |
## Type 3 Analysis of Effects
## ****
```

```

## AIC model complet : 1855
## loglikelihood model complet : 1789
##   variable retiree  AIC loglikelihood p.val
## 1      Percep0GM 1879      1819      0
## 2    PercepRechClim 1941      1881      0
## 3     Connais0GM 1910      1850      0
## 4 ConnaisRechClim 1862      1802 0.0059
## 5          CRT 1860      1800 0.0125
## 6        Parti 1879      1825      0
## 7         AGE 1852      1792 0.4469
## 8        Sexe 1875      1815      0
## 9       Revenu 1850      1790 0.7718

```

L'analyse de type 3 nous permet de déterminer que l'âge et le revenu sont deux variables dont la contribution au modèle est marginale. Nous décidons donc de les retirer pour alléger les tableaux de coefficients que nous présenterons plus loin. Nous pouvons également en conclure que ces deux variables ne jouent aucun rôle dans la propension à être en désaccord avec la recherche scientifique. Nous réajustons le modèle en conséquence.

```

modele2 <- vglm(Y ~ Percep0GM + PercepRechClim + Connais0GM + ConnaisRechClim +
                  CRT + Parti + Sexe,
                  data = data_quest,
                  family = multinomial(refLevel="A"), model = T)

```

Nous pouvons à présent passer à l'analyse des résidus simulés. Le calcul de ces derniers est très similaire au cas d'un modèle logistique de cote proportionnelle présenté dans la section précédente. La figure 1.19 indique que les résidus suivent bien une distribution uniforme, et qu'aucune valeur aberrante n'est observable.

```

# extraction des prédictions (en probabilité)
predicted <- predict(modele2, type = "response")

nsim <- 1000
# lancement des simulations pour chaque observation (chaque lignes dans predicted)
simualtions <- lapply(1:nrow(predicted), function(i){
  probs <- predicted[i,]
  sims <- sample(c(1,2,3,4), size = nsim, replace = T, prob = probs)
  return(sims)
})
matsim <- do.call(rbind, simualtions)

# on conserver la catégorie avec la plus haute probabilité
# comme catégorie prédite
pred_cat <- colnames(predicted)[max.col(predicted)]

# on convertit les catégories en nombres entiers (nécessaire pour le package DHARMA)

real <- case_when(data_quest$Y == "A" ~ 1,
                    data_quest$Y == "B" ~ 2,
                    data_quest$Y == "C" ~ 3,
                    data_quest$Y == "D" ~ 4)

```

```

pred <- case_when(pred_cat == "A" ~ 1,
                   pred_cat == "B" ~ 2,
                   pred_cat == "C" ~ 3,
                   pred_cat == "D" ~ 4)

# on calcule les résidus simulés
sim_res <- createDHARMA(simulatedResponse = matsim,
                          observedResponse = real,
                          fittedPredictedResponse = pred,
                          integerResponse = T)

DHARMA:::plotQQunif(sim_res)

```

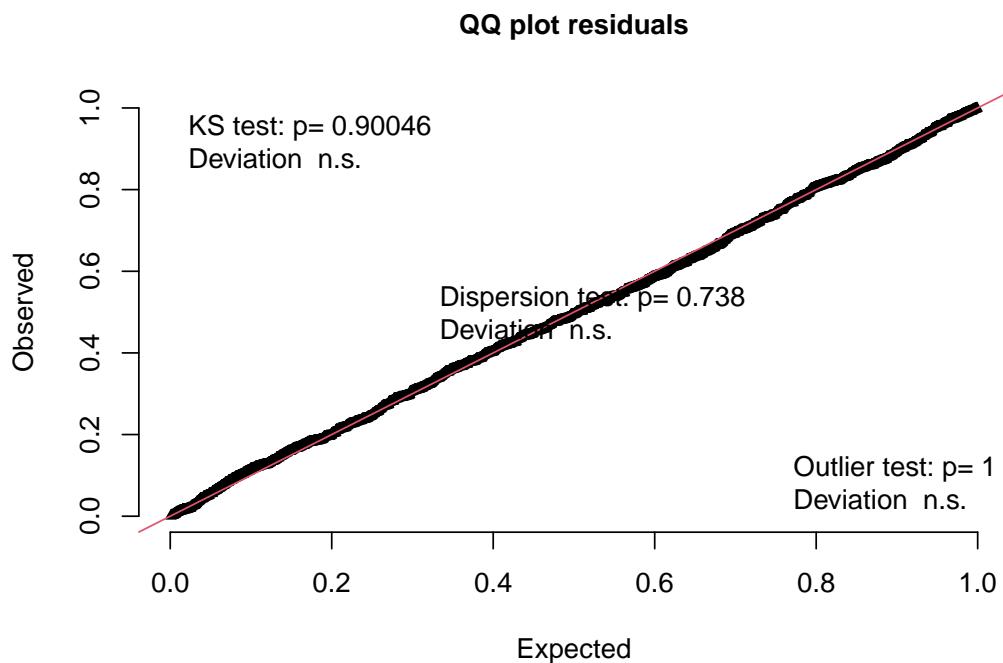


FIG. 1.19 : Diagnositc général des résidus simulés pour le modèle multinomial

La figure 1.20 permet d'affiner le diagnostic en s'assurant de l'absence de relation entre les variables indépendantes et les résidus. Il est possible de remarquer des irrégularités pour les variables de perceptions (premier en second panneau). Dans les deux cas, la catégorie 1 (fort désaccord) se démarque nettement des autres catégories. Nous proposons donc de les recoder comme des variables binaires : en désaccord / pas en désaccord pour minimiser cet effet.

```

par(mfrow=c(3,3))
vars <- c("PercepOGM", "PercepRechClim", "ConnaisOGM",
        "ConnaisRechClim", "CRT", "Parti", "Sexe")

for(v in vars){
  plotResiduals(sim_res, data_quest[[v]], main = v)
}

```

Nous réajustons donc le modèle et recalculons nos résidus ajustés (masqué ici pour alléger le document). La figure 1.21 nous confirme que le problème a été corrigé.

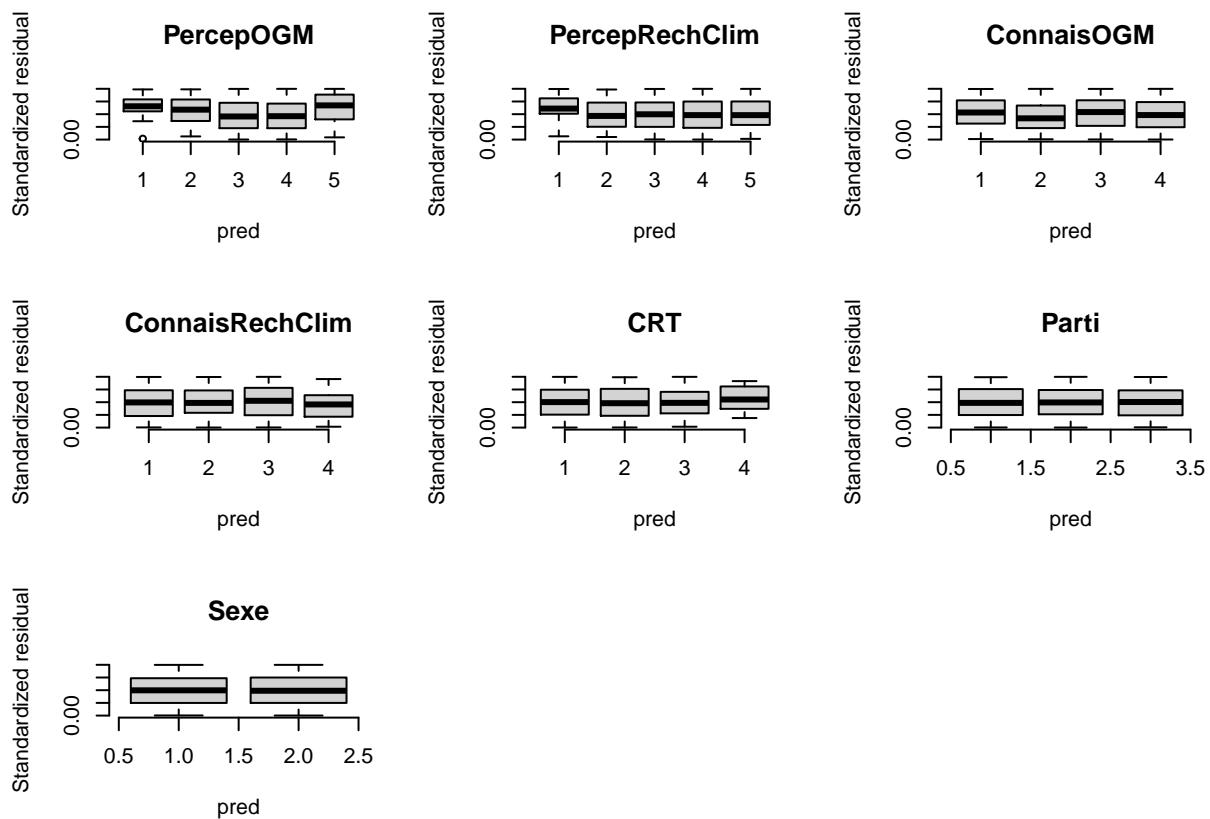


FIG. 1.20 : Diagnositc par variable des résidus simulés pour le modèle multinomial

```
# convertir les variable ordinale et variables binaires
data_quest$PercepOGMDes <- ifelse(data_quest$PercepOGM %in% c(1,2), 1,0)
data_quest$PercepRechClimDes <- ifelse(data_quest$PercepRechClim %in% c(1,2), 1,0)

# réajuster le modèle
modele3 <- vglm(Y ~ PercepOGMDes + PercepRechClimDes +
  ConnaisOGM + ConnaisRechClim +
  CRT + Parti + Sexe,
  data = data_quest,
  family = multinomial(refLevel="A"), model = T)
```

Profitons du fait que nous utilisons le package **VGAM** pour vérifier l'absence d'effet de hauck-Donner qui indiquerait que des variables indépendantes provoquent des séparations parfaites.

```
test <- hdeff(modele3)
test[test==TRUE]

## (Intercept):2 (Intercept):3
##          TRUE          TRUE
```

La fonction nous informe que les intercepts (constantes) permettant de comparer le groupe C au groupe A, et le groupe D au groupe A provoquent des séparations parfaites. Ceci s'explique notamment par le faible nombre d'observations tombant dans ces catégories. Considérant que comparativement à la catégorie A, être dans les catégories B, C, ou D signifie remettre en cause au moins un consensus scientifique, il peut être raisonnable de fixer l'intercept pour qu'il soit le même pour les trois comparaisons. Ainsi, les

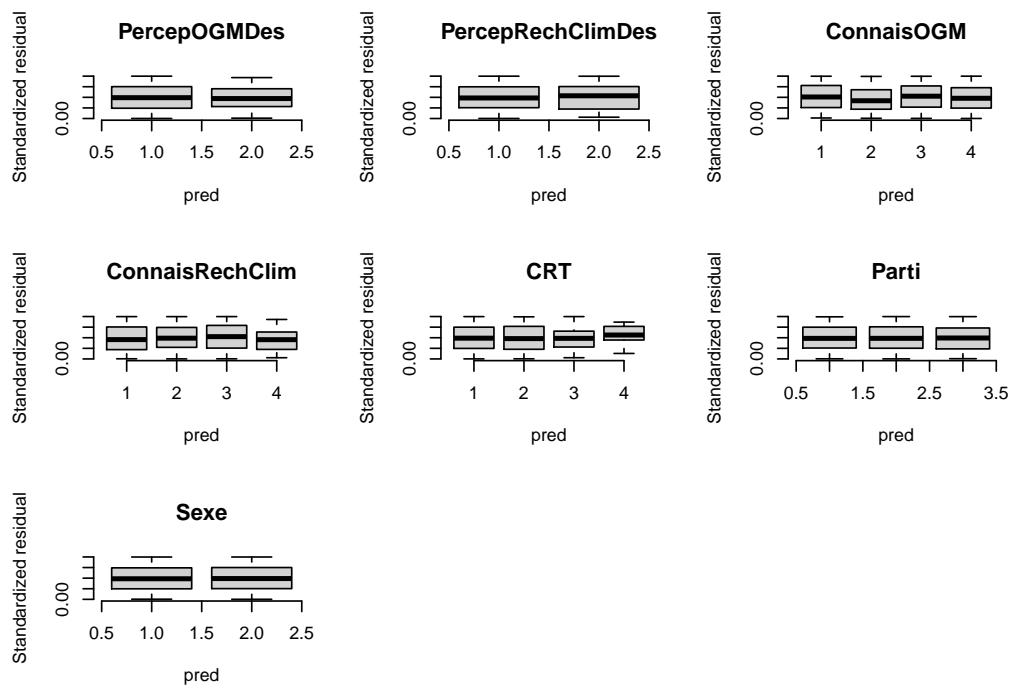


FIG. 1.21 : Diagnositc général des résidus simulés pour le modèle multinomial (version 3)

chances de passer de A à un autre groupe ne dépendrait pas du groupe en question, mais uniquement des prédicteurs. Pour cela, nous pouvons forcer le modèle à n'ajuster qu'un seul **intercept** avec la syntaxe suivante :

```
modele4 <- vglm(Y ~ PercepOGMDes + PercepRechClimDes +
  ConnaisOGM + ConnaisRechClim +
  CRT + Parti + Sexe,
  data = data_quest,
  family = multinomial(refLevel="A",
                        parallel = TRUE ~1), model = T)
test <- hdeff(modele4)
test[test==TRUE]
## named logical(0)
```

Nous n'avons donc plus de séparation complète, et les résidus simulés de la quatrième version du modèle sont toujours acceptables (figure 1.22). Ils notent cependant un début de surdispersion, mais le test n'est pas significatif au seuil 0,01.

Vérifier l'ajustement du modèle

Puisque les conditions d'application du modèle sont respectées, nous pouvons à présent vérifier sa qualité d'ajustement

```
modelenull <- vglm(Y ~ 1 ,
  data = data_quest,
  family = multinomial(refLevel="A",
                        parallel = TRUE ~ 1 + CRT),
  model = T)
```

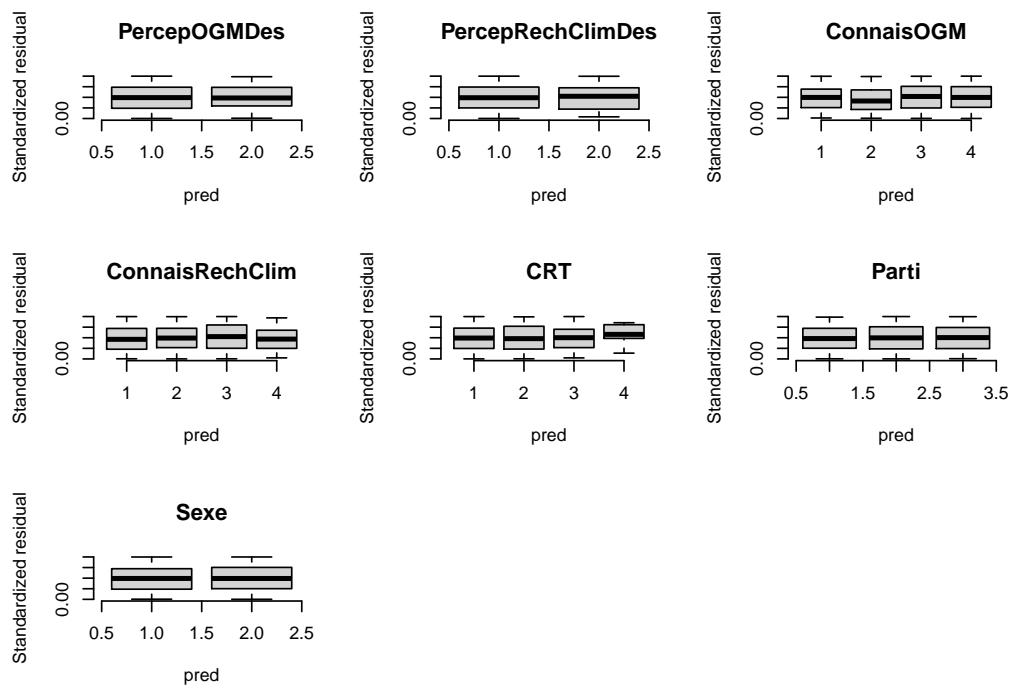


FIG. 1.22 : Diagnositc général des résidus simulés pour le modèle multinomial (version 4)

```
rsqs(loglike.full = logLik(modele4),
loglike.null = logLik(modele4),
full.deviance = deviance(modele4),
null.deviance = deviance(modele4),
nb.params = modele4@rank,
n = nrow(data_quest)
)
```

```
## `$deviance expliquee`
## [1] 0.1750071
##
## `$MacFadden ajuste`
## [1] 0.1530715
##
## `$`Cox and Snell`
## [1] 0.3397248
##
## `$Nagelkerke
## [1] 0.3746843
```

Le modèle parvient à expliquer 17,5% de la déviance totale, il obtient un R2 ajusté de MacFadden de 0,15, et des R2 de Cox et Snell et de Nagelkerke de respectivement 0,34 et 0,37.

Passons à la construction de la matrice de confusion pour analyser la capacité de prédiction du modèle

```
preds <- predict(modele4, type = "response")
pred_cats <- colnames(preds)[max.col(preds)]
```

```

real <- data_quest$Y

matrices <- nice_confusion_matrix(real, pred_cats)

# afficher la matrice de confusion
print(matrices$confusion_matrix)

##          rowsnames   A      B      C      D      rs
## colsnames "" "A (reel)" "B (reel)" "C (reel)" "D (reel)" "Total"
## A          "A (predit)" "482"    "168"    "75"    "29"    "754"
## B          "B (predit)" "31"     "88"     "9"     "20"    "148"
## C          "C (predit)" "10"     "6"     "18"     "6"     "40"
## D          "D (predit)" "3"     "4"     "3"     "9"     "19"
##          "Total"      "526"    "266"    "105"    "64"    "961"
##          "%"         "54.7"   "27.7"   "10.9"   "6.7"   NA
##          rp
## colsnames "%"
## A          "78.5"
## B          "15.4"
## C          "4.2"
## D          "2"
##          NA
##          NA

# afficher les indicateurs de qualité de prédiction
print(matrices$indicators)

##          rnames      precision      rappel      F1
## A          "A"        "0.64"       "0.92"       "0.75"
## B          "B"        "0.59"       "0.33"       "0.43"
## C          "C"        "0.45"       "0.17"       "0.25"
## D          "D"        "0.47"       "0.14"       "0.22"
## macro_scores "macro"    "0.6"        "0.62"       "0.57"
##          "Kappa"      "0.27"       NA          NA
##          "Valeur de p (précision > NIR)" "0"        NA          NA

```

La précision globale du modèle est de 60% et dépasse significativement le seuil de non information. L'indicateur de Kappa indique un accord modéré entre la prédiction et les valeurs réelle. Les catégories C et D sont les catégories avec la plus faible précision, indiquant ainsi que le modèle a tendance à manquer les prédictions pour les individus en désaccord avec la consensus scientifique sur le réchauffement climatique. Les indices de rappel sont également très faibles pour les catégories B, C et D, indiquant qu'assez peu d'observations appartenant originellement à ces groupes ont bien été classé dans ces groupes. La capacité de prédiction du modèle est donc relativement faible.

Interprétation des résultats

Puisque nous disposons de quatre catégories dans notre variable Y, nous obtenons au final 3 tableaux de coefficients. Il est possible de visualiser l'ensemble des coefficients du modèle avec la fonction `summary`, nous proposons les tableau 1.17, 1.18 et 1.19 pour présenter l'ensemble des résultats.

Le tableau 1.17 compare donc le groupe A (en accord avec la recherche scientifique sur les deux



TAB. 1.17 : Coefficients du modèle multinomial A VS B

variable	coefficient	RC	val.p	RC 2,5%	RC 97,5%	sign.
PercepOGMDes	1.94	6.989	<0.001	4.221	11.588	***
PercepRechClimDes	0.43	1.532	0.305	0.677	3.456	
ConnaisOGM	-0.33	0.718	<0.001	0.607	0.852	***
ConnaisRechClim	0.18	1.197	0.084	0.980	1.462	.
CRT	0.35	1.417	0.016	1.073	1.878	*
<i>Parti</i>						
ref : Democrat	—	—	—	—	—	—
autre	0.66	1.934	0.001	1.310	2.858	***
Republicain	0.65	1.910	0.003	1.259	2.915	**
<i>Sexe</i>						
ref : homme	—	—	—	—	—	—
femme	1.28	3.581	<0.001	2.535	5.053	***

sujets) et le groupe B (en désaccord sur la question des OGM). Les résultats indiquent que le fait de se percevoir en désaccord avec le consensus scientifique sur la question des OGM multiple par sept les chances d'appartenir au groupe B comparativement au groupe A. Cependant pour chaque bonne réponse supplémentaire sur les questions testant les connaissances sur les OGM, les chances d'appartenir au groupe B comparativement au groupe A diminue de 28%. Ainsi, un individu ayant répondu correctement aux trois questions aurait ses chances réduites de 63% d'appartenir au groupe B ($\exp(-0.33*3)$). Il est intéressant de noter que les variables concernant le réchauffement climatique n'ont pas d'effet significatif ici. La variable CRT indique qu'à chaque bonne réponse supplémentaire au test de Cognition, les chances d'appartenir au groupe B augment de 42%. Un individu qui aurait répondu juste au trois question du test aurait donc 2,9 fois plus de chances d'appartenir au groupe B qu'au groupe A. Concernant le parti politique, comparativement à une personne se déclarant plus proche du parti démocrate, les personnes proches du parti républicain ou d'un autre parti ont près de 2 fois plus de chances d'appartenir au groupe B. Enfin, une femme, comparativement à un homme à 3,6 fois plus de chance d'appartenir au groupe B.

Le tableau 1.18 compare les groupes A et C (en désaccord sur le réchauffement climatique). Il est intéressant de noter ici que se percevoir en désaccord avec la recherche scientifique est associé avec une forte augmentation des chances d'appartenir au groupe C. Cependant, les un plus grand nombre de bonnes réponses aux question sur le réchauffement climatique est également associé avec une augmentation des chances (30% à chaque bonne réponse supplémentaire) d'appartenir au groupe C. Le CRT n'a cette fois-ci pas d'impact. Se déclarer proche du parti républicain, comparativement au

**TAB. 1.18 :** Coefficients du modèle multinomial A VS C

variable	coefficient	RC	val.p	RC 2,5%	RC 97,5%	sign.
PercepOGMDes	-0.18	0.834	0.705	0.326	2.138	
PercepRechClimDes	2.06	7.821	<0.001	3.819	16.119	***
ConnaisOGM	-0.11	0.896	0.287	0.733	1.094	
ConnaisRechClim	0.28	1.323	0.024	1.041	1.682	*
CRT	0.24	1.275	0.149	0.914	1.768	
<i>Parti</i>						
ref : Democrat	—	—	—	—	—	—
autre	-0.19	0.828	0.496	0.482	1.433	
Republicain	0.92	2.512	<0.001	1.584	4.015	***
<i>Sexe</i>						
ref : homme	—	—	—	—	—	—
femme :1	-0.45	0.640	0.040	0.419	0.980	*

parti démocrate, multiplie les chances par 2,5 d'appartenir au groupe C. Comparativement au tableau précédent, le fait d'être une femme diminue les chances de 36% d'appartenir au groupe C.

Le dernier tableau 1.19 compare le groupe A au groupe D (en désaccord sur les deux sujets). Les variables les plus importantes sont une fois encore le fait de se sentir en désaccord avec la recherche scientifique et le degré de connaissance sur les OGM. La variable concernant le parti politique est significative au seuil 0,05 et exprime toujours une tendance accrue pour les individus du parti républicain à appartenir au groupe D.

Nos propres conclusions corroborent celles de l'article original. Une des conclusions intéressante est que le rejet du consensus scientifique ne semble pas nécessairement être associé à un déficit d'information ni à une plus faible capacité analytique, mais relèverait d'avantage d'une polarisation politique. Notez que cette littérature sur les croyances et la confiance dans la recherche est complexe, si le sujet vous intéresse, la discussion de l'article de [McFadden \(2016\)](#) est un bon point de départ.

1.2.1.5 Conclusion sur les modèles pour des variables qualitatives

Nous avons pu voir dans cette section les trois principales formes de modèles GLM pour modéliser une variable binaire (modèle binomial), une variable ordinaire (modèle de cotes proportionnel) et une variable multinomiale (modèle multinomial). Pour ces trois modèles, nous avons vu que la distribution utilisée est toujours la distribution binomiale et la fonction de lien la fonction logistique. Les coefficients obtenus s'interprètent comme des rapports de cote, une fois qu'ils sont transformés avec la fonction exponentielle. Nous avons également vu le modèle binomial probit, une variante du modèle binomial logistique utilisant la fonction probit comme fonction de lien. Sachez qu'il est également possible d'utiliser la fonction de lien probit pour le modèle des cotes proportionnelles et le modèle multinomial.

1.2.2 Les modèles GLM pour des variables de comptage

Dans cette section, nous présentons les principaux modèles utilisés pour modéliser des variables de comptage. Il peut s'agir de variable comme le nombre d'accident à une intersection, le nombre de cafés par quartier, le nombre de cas de certaines maladie par secteurs de recensement, etc.

1.2.2.1 Le modèle de Poisson

Le modèle GLM de base pour modéliser une variable de comptage est le modèle de Poisson. Pour rappel, la distribution de Poisson a un seul paramètre : (λ) . Il représente le nombre moyen d'événements

TAB. 1.19 : Coefficients du modèle multinomial A VS D

variable	coefficient	RC	val.p	RC 2,5%	RC 97,5%	sign.
PercepOGMDes	1.50	4.488	<0.001	2.270	8.935	***
PercepRechClimDes	2.44	11.501	<0.001	5.312	25.028	***
ConnaisOGM	-0.48	0.621	<0.001	0.492	0.787	***
ConnaisRechClim	0.13	1.140	0.399	0.844	1.553	
CRT	0.34	1.409	0.119	0.914	2.160	
<i>Parti</i>						
ref : Democrate	-	-	-	-	-	-
autre	0.19	1.211	0.531	0.664	2.203	
Republicain	0.63	1.872	0.038	1.041	3.387	*
<i>Sexe</i>						
ref : homme	-	-	-	-	-	-
femme :1	-0.11	0.896	0.664	0.543	1.477	

observés sur l'intervalle de temps retenu, ainsi que la variance de la distribution. En conséquence, λ doit être un nombre strictement positif. On ne peut pas observer un nombre négatif de phénomène. Il est donc nécessaire d'utiliser une fonction de lien pour contraindre l'équation de régression sur l'intervalle $[0, +\infty]$. La fonction la plus utilisée est le logarithme naturel (\log) dont la réciproque est la fonction exponentielle (\exp).

1.2.2.1.1 Interprétation des paramètres

Les coefficients du modèle expriment l'impact du changement d'une unité des variables X sur λ (le nombre de cas) dans l'échelle logarithmique (log-scale). Pour rappel, l'échelle logarithmique est multiplicative, ce qui complique quelque peu la tâche lorsque l'on tente de revenir à l'échelle originale des données. En effet, si l'on convertit les coefficients dans leur échelle originale avec la fonction exponentielle, leur effet n'est plus additif, mais multiplicatif. Prenons un exemple concret, admettons que nous ayons ajusté un modèle de poisson à une variable de comptage Y avec deux variables x_1 et x_2 et que nous ayons obtenus les coefficients suivants :

$$\beta_0 = 1,8; \beta_1 = 0,5; \beta_2 = -1,5$$

L'interprétation basique (sur l'échelle log) est la suivante : Une augmentation de 1 unité de la variable x_1 est associée avec une augmentation de 0.5 du log du nombre de cas attendu. Une augmentation de 1 unité de la variable x_2 est associée avec une réduction de 1.5 unités du log du nombre de cas attendu. Avec la conversion avec la fonction exponentielle, on obtient :

- $\exp(0.5) = 1.649$, soit une multiplication par 1.649 du nombre de cas attendu à chaque augmentation d'une unité de x_1 .
- $\exp(-1.5) = 0.223$, soit une division par 4.54 du nombre de cas attendu à chaque augmentation d'une unité de x_2 .

Utilisons maintenant notre équation pour effectuer une prédiction si $x_1 = 1$ et $x_2 = 1$.

$$\lambda = \exp(1,8 + 0,5 * 1 + -1,5 * 1) = 2,225$$

Si nous augmentons x_1 d'une unité, nous obtenons :

$$\lambda = \exp(1,8 + 0,5 * 2 + -1,5 * 1) = 3,670$$

En ayant augmenté d'une unité x_1 , nous avons multiplié notre résultat par $1,649 : 2,225 * 1,649 = 3,670$

Notes que ces effets se multiplient entre eux ! Si nous augmentons à la fois x_1 et x_2 d'une unité chacun, nous obtenons : $\lambda = \exp(1,8 + 0,5 * 2 + -1,5 * 2) = 0,818$, ce qui correspond bien à $2,225 * 1,649 * 0,223$ (effet de x_1) * $0,223$ (effet de x_2) = 0,818

TAB. 1.20 : Carte d'identité du modèle de Poisson

Type de variable dépendante	Variable de comptage
Distribution utilisée	Poisson
Formulation	$Y \sim Poisson(\lambda)$ $g(\lambda) = \beta_0 + \beta X$ $g(x) = \log(x)$
Fonction de lien	\log
Paramètre modélisé	λ
Paramètres à estimer	β_0, β
Conditions d'applications	Absence d'excès de zéros, Absence de surdispersion ou sousdispersion

Il existe des fonctions dans R qui s'occupe de calculer ces prédictions à partir des équations des modèles pour vous. Il est cependant essentiel de bien saisir ce comportement multiplicatif induit par la fonction de lien log.

1.2.2.1.2 Conditions d'applications

Puisque la distribution de Poisson n'a qu'un seul paramètre, le modèle GLM de Poisson est exposé au même problème potentiel de sur-dispersion que les modèles binomiaux de la section précédente. Référez-vous à la section 1.2.1.1.2 pour davantage de détails sur le problème posé par la sur-dispersion. Pour détecter une potentielle surdispersion dans un modèle de Poisson, il est possible dans un premier temps de calculer le ratio entre la déviance du modèle et son nombre de degré de liberté (SAS Institute Inc, 2020b). Ce ratio doit être proche de 1, s'il est plus grand, le modèle souffre de surdispersion.

$$\hat{\phi} = \frac{D(\text{modele})}{N - p} \quad (1.18)$$

Avec N le nombre d'observations et p le nombre de paramètres. Il est également possible de tester formellement si la surdispersion est significative avec un test de dispersion.

La question de l'excès de zéros avait été abordée dans la section présentant les distributions (@ref(#sectpoissonzero)). Il s'agit d'une situation où un plus grand nombre de zéros sont présents dans les données que ce que suppose la distribution de Poisson. Dans ce cas de figure, il vaut mieux utiliser la distribution de Poisson avec excès de zéros. Il arrive parfois que les problèmes de surdispersion soient provoqués par des excès de zéro. Une façon simple de vérifier si nous avons un trop grand nombre de zéros est de comparer les quantités prédites par le modèle de poisson avec les quantités réelle (variable Y).

1.2.2.1.3 Exemple appliqué dans R

Pour cet exemple, nous allons reproduire l'analyse effectuée dans l'article de Cloutier et al. (2014). L'enjeu de cette étude était de modéliser le nombre de piétons blessés autour de plus de 500 carrefours dans les quartiers centraux de Montréal. Pour cela, trois types de variables étaient utilisées : des variables décrivant l'intersection, des variables décrivant les activités humaines dans un rayon de 1km autour de l'intersection et des variables représentant le trafic routier autour de l'intersection. Un impact direct de ce type d'étude est bien évidemment l'établissement de meilleures pratiques d'aménagement réduisant les risques encourus par les piétons lors de leurs déplacements en ville. Le tableau 1.21 présente l'ensemble des variables utilisées dans l'analyse.

La distribution originale de la variable est décrite à la figure 1.23. Les barres grises représente la distribution du nombre d'accident et les barres rouges une distribution de Poisson ajustée sans prédicteur (modèle nul). Ce premier graphique peut laisser penser qu'un modèle de Poisson ne sera pas nécessairement le plus adapté considérant le grand nombre d'intersection pour lesquelles nous n'avons aucun accident. Cependant, rappelons que la variable Y n'a pas besoin de suivre une distribution de Poisson. Dans un modèle GLM, l'hypothèse que nous formulons est que la variable Y **conditionnée par les variables X** suit une certaine distribution (ici Poisson).

```
# chargement des données
data_accidents <- read.csv("data/glm/accident_pietons.csv")

# ajustement d'une distribution de Poisson sans predicteur
library(fitdistrplus)
```

TAB. 1.21 : Variable indépendantes utilisées dans le modèle de Poisson

Nom de la variable	signification	Type de variable	Mesure
Feux_auto	Présence de feux de circulation	Variable binaire	0 = absence ; 1 = présence
Feux_piet	Présence de feux de traversée pour les piétons	Variable binaire	0 = absence ; 1 = présence
Pass_piet	Présence d'un passage piéton	Variable binaire	0 = absence ; 1 = présence
Terreplein	Présence d'un terre-plein central	Variable binaire	0 = absence ; 1 = présence
Apaisement	Présence de mesure d'apaisements de la circulation	Variable binaire	0 = absence ; 1 = présence
LogEmploi	Logarithme du nombre d'emploi dans un rayon d'un kilomètre	Variable continue	Logarithme du nombre d'emploi. Utilisation du logarithme car la variable est fortement asymétrique
Densite_pop	Densité de population dans un rayon d'un kilomètre	Variable continue	Habitants / hectare
Entropie	Diversité de l'occupation du sol dans un rayon d'un kilomètre	Variable continue	Mesure de 0 à 1 ; 0 = spécialisation parfait ; 1 = diversité parfaite
DensiteInter	Densité d'intersection dans un rayon d'un kilomètre (connexité)	Variable continue	Nombre d'intersection par km ²
Artere	Présence d'une artère à l'intersection	Variable binaire	0 = absence ; 1 = présence
Long_arterePS	Longueur d'artère dans un rayon d'un kilomètre	Variable continue	Exprimée en mètres
NB_voies5	Présence d'une cinq voies à l'intersection	Variable binaire	0 = absence ; 1 = présence

```

model_poisson <- fitdist(data_accidents$Nbr_acci, distr = "pois")

# création d'un graphique pour comparer les deux distributions
dfpoisson <- data.frame(x=c(0:19),
                        y=dpois(0:19, model_poisson$estimate)
)

counts <- data.frame(table(data_accidents$Nbr_acci))
names(counts) <- c("nb_accident", 'frequence')
counts$nb_accident <- as.numeric(as.character(counts$nb_accident))
counts$prop <- counts$frequence / sum(counts$frequence)

ggplot() +
  geom_bar(aes(x=nb_accident, weight = prop, fill = "real"), width = 0.6, data = counts) +
  geom_bar(aes(x=x, weight = y, fill = "adj"), width = 0.15, data = dfpoisson) +
  scale_x_continuous(limits = c(-0.5,7), breaks = c(0:7)) +
  scale_fill_manual(name = "", 
                     breaks = c("real","adj"),
                     labels = c("distribution originale", "distribution de Poisson"),
                     values = c("real" = rgb(0.4,0.4,0.4), "adj" = "red")) +
  labs(subtitle = "",
        x = "nombre d'accidents",
        y = "")

```

** Vérification des conditions d'application **

Comme précédemment, notre première étape est de vérifier l'absence de multicolinéarité avec la fonction **vif** du package **car**.

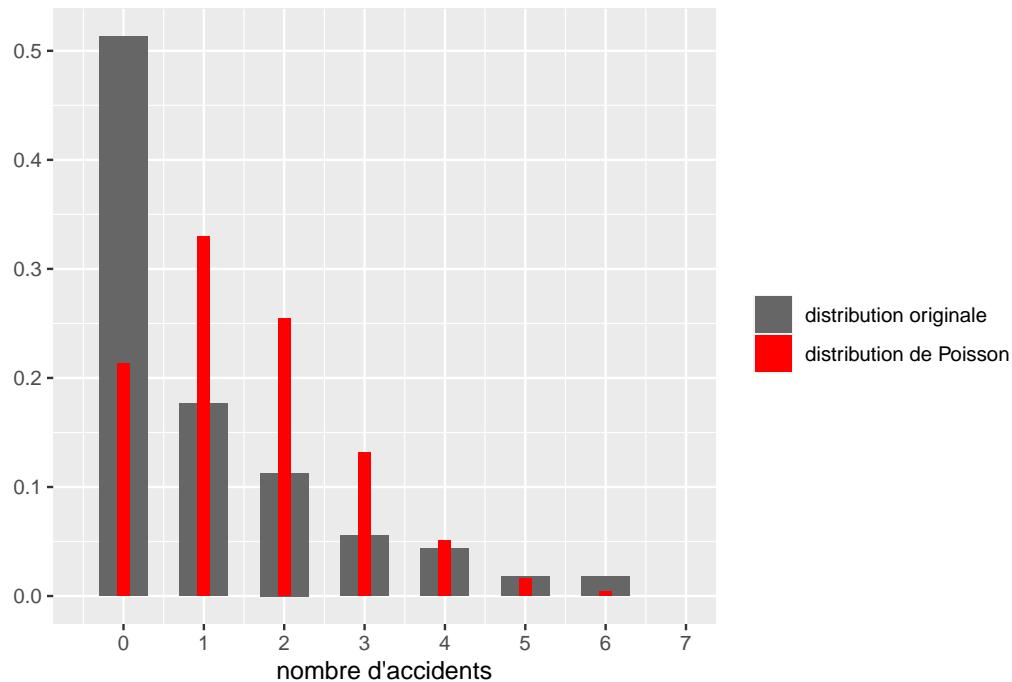


FIG. 1.23 : Distribution originale du nombre d'accident par intersection

```
vif(glm(Nbr_acci ~ Feux_auto + Feux_piet + Pass_piet + Terreplein + Apaisement +
         LogEmploi + Densite_pop + Entropie + DensiteInter +
         Long_arterePS + Artere + NB_voies5,
         family = poisson(link="log"),
         data = data_accidents))
```

```
##          Feux_auto      Feux_piet      Pass_piet      Terreplein      Apaisement
## 2.861708     1.518668     2.321213     1.221683     1.059722
##          LogEmploi    Densite_pop      Entropie  DensiteInter Long_arterePS
## 4.763692     1.153096     1.770904     2.040457     4.467841
##          Artere      NB_voies5
## 1.887728     1.520514
```

Toutes les valeurs de VIF sont inférieures à 5, notons tout de même que le logarithme de l'emploi et la longueur d'artère dans un rayon d'un kilomètre ont des valeurs de VIF proches de cinq. La second étape du diagnostic est le calcul et l'affichage des distances de Cook.

```
# ajustement d'une première version du modèle
modele <- glm(Nbr_acci ~ Feux_auto + Feux_piet + Pass_piet + Terreplein + Apaisement +
               LogEmploi + Densite_pop + Entropie + DensiteInter +
               Long_arterePS + Artere + NB_voies5,
               family = poisson(link="log"),
               data = data_accidents)

# calcule des distances de Cook
cooksd <- cooks.distance(modele)
df <- data.frame(
  cook = cooksd,
  oid = 1:length(cooksd))
```

```
)
ggplot(data = df) +
  geom_point(aes(x = oid, y = cook), size = 0.5, alpha = 0.5) +
  labs(x = "",  

       y = "distance de Cook")
```

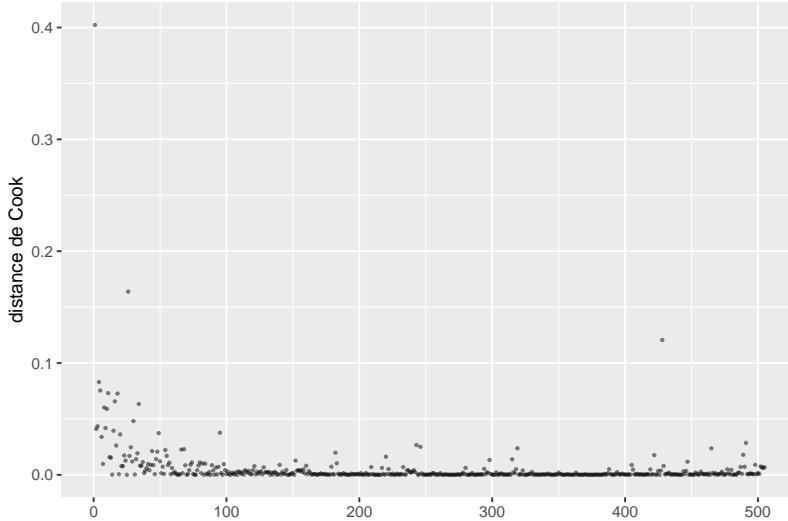


FIG. 1.24 : Distance de Cook pour le modèle de Poisson

La figure 1.24 nous montre que trois observations ont des valeurs extrêmement fortes pour leur distance de Cook. Nous les isolons dans un premier temps pour les observer.

```
cas_étrange <- subset(data_accidents, cooksd>0.1)
print(cas_étrange)

##          X NumInter NOUNIK Nbr_acci Feux_auto Feux_piet Pass_piet Terreplein
## 1      1     7839     419      19       1       1       1       1
## 26    26     8442     136       7       0       0       1       0
## 428   428     8823     458       0       1       1       1       0
##          Apaisement EmpTotBuffer Densite_pop Entropie DensiteInter Long_arterePS
## 1          0     7208.538    5980.923  0.8073926    42.41597   6955.00
## 26          0     1342.625    2751.012  0.0000000    73.35344   2849.66
## 428         1    13122.560   14148.827  0.6643891   109.25066  4634.20
##          Artere NB_voies5 log_acci catego_acci catego_acci2 Arret VAG sum_app
## 1          1           1  2.995732           1       1       0       1       4
## 26          0           0  2.079442           1       1       1       1       3
## 428         0           0  0.000000           0       0       0       0       3
##          LogEmploi AccOrdinal PopHa
## 1     8.883021           2  5.980923
## 26    7.202382           2  2.751012
## 428   9.482088           0 14.148827
```

Les deux premiers cas sont des intersections avec de nombreux accidents (respectivement 19 et 7), qui risquent de perturber les estimations du modèle. Le troisième cas ne comprend en revanche aucun accident. Puisqu'il ne s'agit que de trois observations et que leurs distances de Cook sont très nettement

supérieures aux autres, nous les retirons du modèle.

```
data2 <- subset(data_accidents, cooksd<0.1)

# ajustement d'une première version du modèle
modele <- glm(Nbr_acci ~ Feux_auto + Feux_piet + Pass_piet + Terreplein + Apaisement +
                LogEmploi + Densite_pop + Entropie + DensiteInter +
                Long_arterePS + Artere + NB_voies5,
                family = poisson(link="log"),
                data = data2)

# calcule des distances de Cook
cooksd <- cooks.distance(modele)
df <- data.frame(
  cook = cooksd,
  oid = 1:length(cooksd)
)

ggplot(data = df) +
  geom_point(aes(x = oid, y = cook), size = 0.5, alpha = 0.5) +
  labs(x = "", y = "distance de Cook")
```

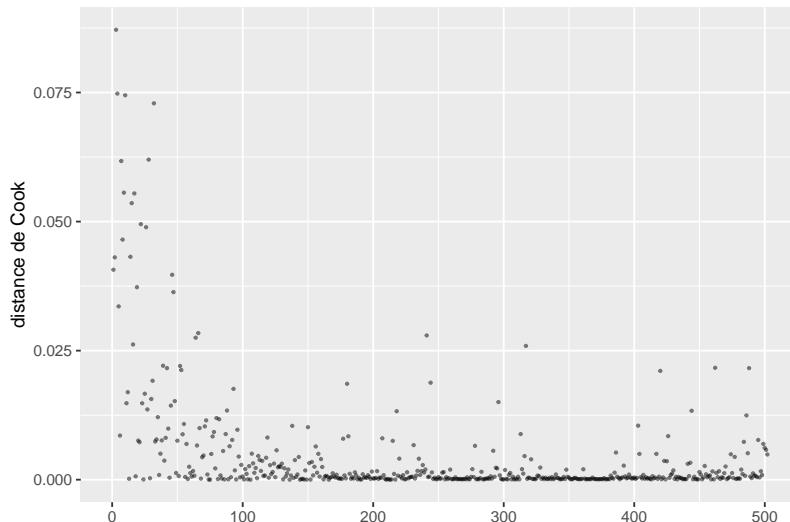


FIG. 1.25 : Distance de Cook pour le modèle de Poisson après avoir retiré les valeurs aberrantes

La figure 1.25 montre que nous n'avons plus d'observations fortement influentes dans le modèle après avoir retiré les trois observations identifiées précédemment. Nous devons à présent vérifier l'absence de surdispersion.

```
# calcule du rapport entre déviance et nombre de degré de liberté du modèle
deviance(modele)/(nrow(data2) - modele$rank)
```

```
## [1] 1.674691
```

Le rapport entre la déviance et le nombre de degrés de liberté du modèle est nettement supérieure à 1, indiquant une surdispersion excessive. Nous pouvons confirmer ce résultats avec la fonction dispersiontest du package **AER**.

```

library(AER)
# test de surdispersion
dispersiontest(modele)

## 
## Overdispersion test
##
## data: modele
## z = 5.2737, p-value = 6.686e-08
## alternative hypothesis: true dispersion is greater than 1
## sample estimates:
## dispersion
## 1.891565

```

contrairement à la forme classique d'un modèle de poisson pour laquelle la dispersion attendue est de 1, le test nous indique qu'une dispersion de 1,89 serait mieux ajustée aux données.

Il est également possible de constater graphiquement cet écart avec un graphique représentant les valeurs réelles, les valeurs prédites, ainsi que la variance (sous forme de barres d'erreurs) attendue par le modèle (figure : 1.26). Nous constatons ainsi que les valeurs réelles ont largement tendance à dépasser la variance attendue par le modèle, surtout pour les valeurs les plus faibles de la distribution.

```

# extraction des prédition du modèle
lambdas <- predict(modele, type = "response")

# création d'un dataframe pour contenir la prédition et les vraie valeurs
df1 <- data.frame(
  lambdas = lambdas,
  reals = data2$Nbr_acci
)

# calcul de l'intervalle de confiance à 95% selon la distribution de Poisson
# et stockage dans un second dataframe
seqa <- seq(0,round(max(lambdas)),1)
df2 <- data.frame(
  lambdas = seqa,
  lower = qpois(p = 0.025, lambda = seqa),
  upper = qpois(p = 0.975, lambda = seqa)
)

# affichage des valeurs réelles et prédites (en rouge)
# et de leur variance selon le modèle (en noir)
ggplot() +
  geom_errorbar(data = df2,
    mapping = aes(x = lambdas, ymin = lower, ymax = upper),
    width = 0.2, color = rgb(0.4,0.4,0.4)) +
  geom_point(data = df1,
    mapping = aes(x = lambdas, y = reals),
    color = "red", size = 0.5) +
  labs(x = 'valeurs prédites',
    y = "valeurs réelles")

```

Pour tenir compte de cette particularité des données, nous modifions légèrement le modèle pour

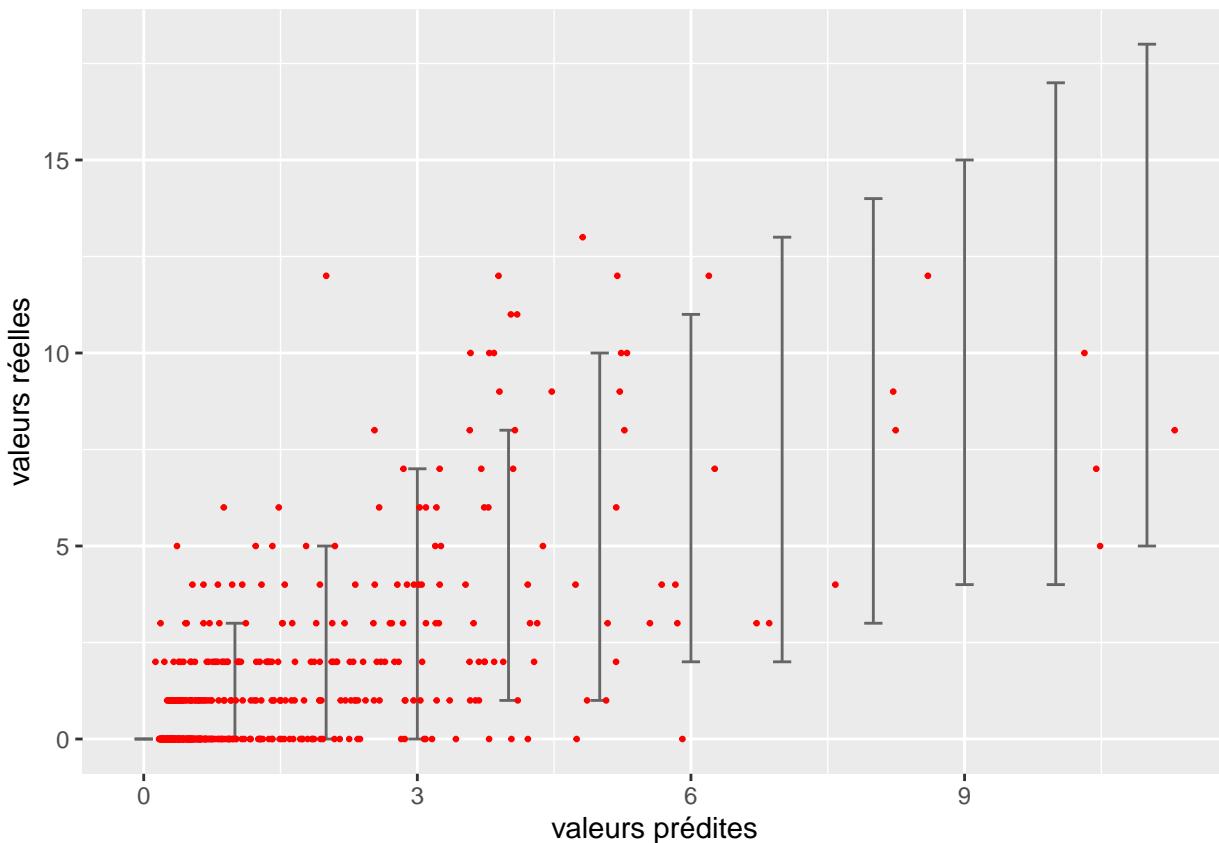


FIG. 1.26 : Représentation de la sur-dispersion des données dans le modèle de Poisson

utiliser une distribution de quasi-Poisson, intégrant spécifiquement un paramètre de dispersion. Cet ajustement ne modifie pas l'estimation des coefficient du modèle, mais modifie le calcul des erreurs standards et par extension les valeurs de p pour les rendre moins sensibles au problème de surdispersion. Une autre approche aurait été de calculer une version robuste des erreurs standards avec le package `sandwich` comme nous l'avons fait dans la section sur le modèle binomial (`?sectexemplebinom`). Après réajustement du modèle, le nouveau paramètre de dispersion estimé est de 1,92.

Les quasi-distributions

Les distributions Binomiale et Poisson ne disposent chacune que d'un paramètre décrivant à la fois leur variance et leur espérance. Elles manquent donc de flexibilité et échouent parfois à représenter fidèlement des données avec une forte variance. Il existe donc des distributions alternatives, respectivement les distributions quasi-binomiale et quasi-poisson ajoutant chacune un paramètre supplémentaire pour contrôler la variance. Bien que cette solution soit attrayante, il ne faut pas perdre de vue que la sur ou la sous dispersion peuvent être causées par l'absence de certaines variables explicatives, la sur représentation de zéros, ou encore une séparation parfaite de la variable dépendante causée par une variable indépendante.

```
modele2 <- glm(Nbr_acci ~ Feux_auto + Feux_piet + Pass_piet + Terreplein + Apaisement +
  LogEmploi + Densite_pop + Entropie + DensiteInter +
  Long_arterePS + Artere + NB_voies5,
  family = quasipoisson(link="log"),
  data = data2)
```

Nous pouvons à présent comparer la distribution originale des données et les simulations issues du

modèle. Notez que contrairement à la distribution de Poisson simple, il n'existe pas dans R de fonction pour simuler des valeurs issues d'une distribution de quasi-Poisson. Il est cependant possible d'exploiter sa proximité théorique avec la distribution Négative Binomiale pour définir notre propre fonction de simulation. La figure 1.27 permet de comparer la distribution originale (en gris) et l'intervalle de confiance à 95% des simulations issus des simulations (en rouge). Nous remarquons que le modèle semble capturer efficacement la forme générale de la distribution originale. À titre de comparaison, nous pouvons effectuer le même exercice avec la distribution de Poisson classique (le code n'est pas montré pour éviter les répétitions). La figure 1.28 montre qu'un simple modèle de Poisson est très éloigné de la distribution originale de Y .

```
# definition d'une fonction pour simuler des données quasi-poisson
rqpois <- function(n, lambda, disp) {
  rnbinom(n = n, mu = lambda, size = lambda/(disp-1))
}

# extraction des valeurs predites par le modele
preds <- predict(modele2, type="response")

# generation de 1000 simulation pour chaque prediction
disp <- 1.918757 # trouvable dans le summary(modele2)
nsim <- 1000
cols <- lapply(1:length(preds), function(i){
  lambda <- preds[[i]]
  sims <- round(rqpois(n = nsim, lambda = lambda, disp = disp))
  return(sims)
})
mat_sims <- do.call(rbind, cols)

# preparation des donnees pour le graphique (valeurs reelles)
counts <- data.frame(table(data2$Nbr_acc))
names(counts) <- c("nb_accident", "frequence")
counts$nb_accident <- as.numeric(as.character(counts$nb_accident))
counts$prop <- counts$frequence / sum(counts$frequence)

# preparation des donnees pour le graphique (valeurs simulees)
df1 <- data.frame(count = 0:25)

count_sims <- lapply(1:nsim, function(i){
  sim <- mat_sims[,i]
  cnt <- data.frame(table(sim))
  df2 <- merge(df1, cnt, by.x="count", by.y = "sim", all.x = T, all.y=F)
  df2$Freq <- ifelse(is.na(df2$Freq), 0, df2$Freq)
  return(df2$Freq)
})

count_sims <- do.call(cbind, count_sims)

df_sims <- data.frame(
  val = 0:25,
  med = apply(count_sims, MARGIN = 1, median),
  lower = apply(count_sims, MARGIN = 1, quantile, probs = 0.025),
  upper = apply(count_sims, MARGIN = 1, quantile, probs = 0.975)
)
```

```
ggplot() +
  geom_bar(aes(x=nb_accident, weight = frequence), width = 0.6, data = counts) +
  geom_errorbar(aes(x = val, ymin = lower, ymax = upper),
                data = df_sims, color = "red", width = 0.6) +
  geom_point(aes(x = val, y = med), color = "red", size = 1.3, data = df_sims) +
  scale_x_continuous(limits = c(-0.5,7), breaks = c(0:7)) +
  xlim(-1,12) +
  labs(subtitle = "",
       x = "nombre d'accidents",
       y = "nombre d'occurrence")
```

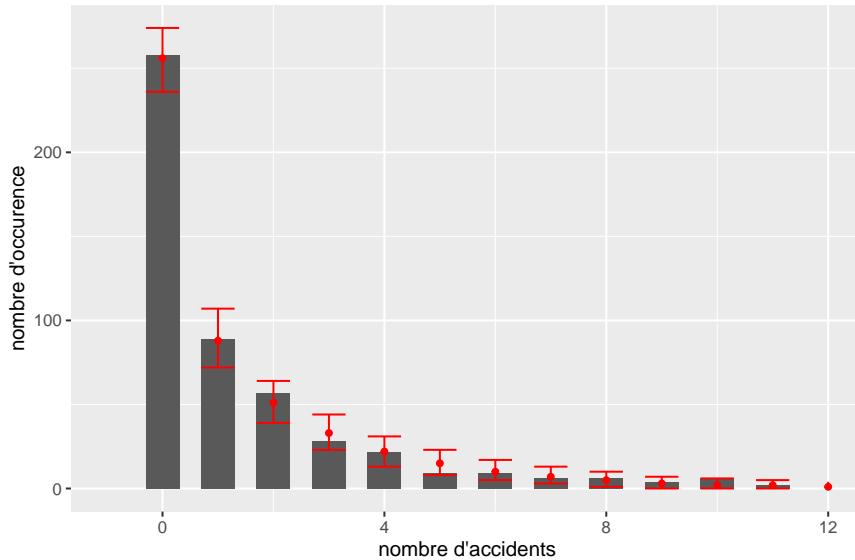


FIG. 1.27 : Comparaison de la distribution originale et des simulations pour le modèle de quasi-Poisson

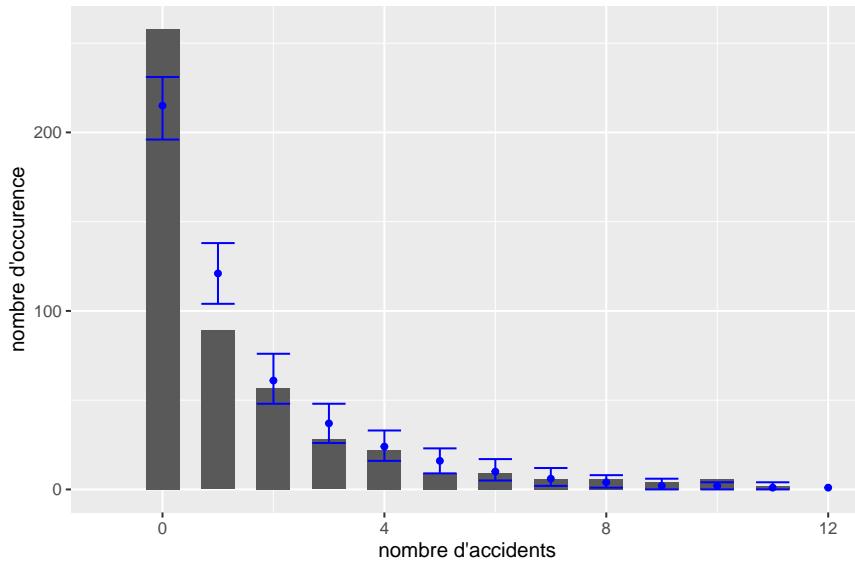


FIG. 1.28 : Comparaison de la distribution originale et des simulations pour le modèle de Poisson

La prochaine étape du diagnostic est l'analyse des résidus simulés. La figure 1.29 indique que les résidus du modèle suivent bien une distribution uniforme et qu'aucune valeur aberrante n'est observable.

```
# génération de 1000 simulation pour chaque prediction
disp <- 1.918757 # trouvable dans le summary(modele2)
nsim <- 1000
cols <- lapply(1:length(preds), function(i){
  lambda <- preds[[i]]
  sims <- rpois(n = nsim, lambda = lambda, disp = disp)
  return(sims)
})
mat_sims <- do.call(rbind, cols)

sim_res <- createDHARMa(simulatedResponse = mat_sims,
                         observedResponse = data2$Nbr_acci,
                         fittedPredictedResponse = modele2$fitted.values,
                         integerResponse = T)

plot(sim_res)
```

DHARMA residual diagnostics

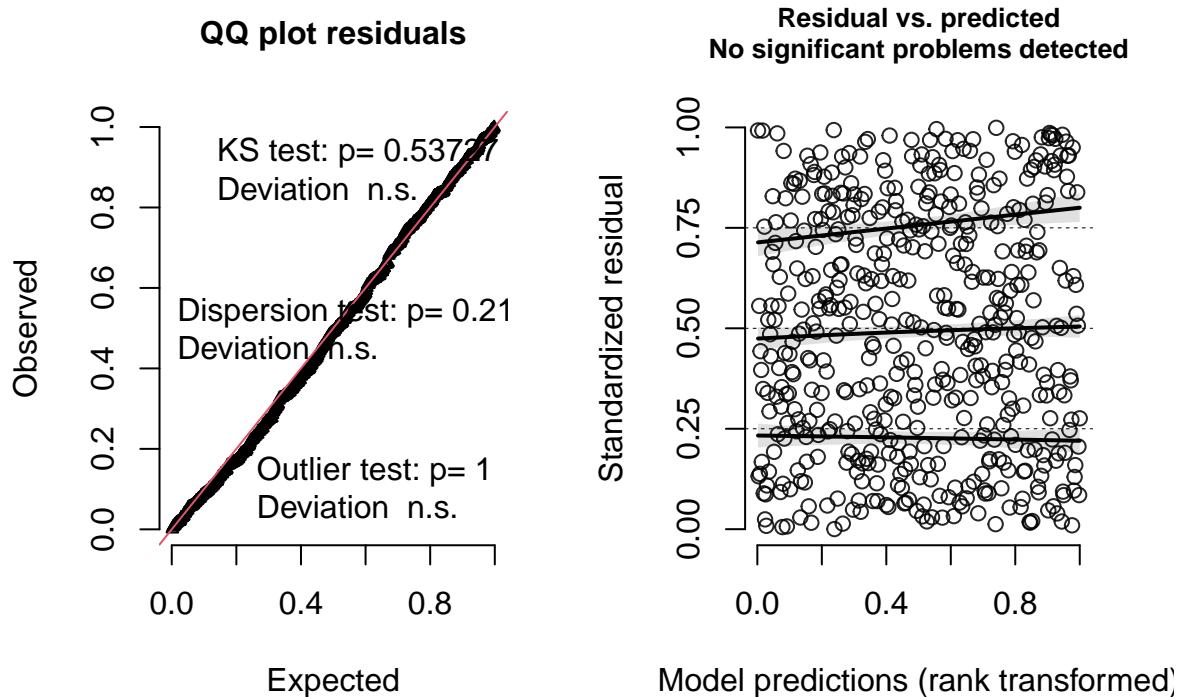


FIG. 1.29 : Analyse globale des résidus simulés pour le modèle de quasi-Poisson

Pour affiner notre diagnostic, nous pouvons également comparer les résidus simulés et chaque variable indépendante. La figure 1.30 n'indique aucune relation problématique entre nos variables indépendantes et les résidus.

```
par(mfrow=c(3,4))
vars <- c("Feux_auto", "Feux_piet", "Pass_piet", "Terreplein", "Apaisement",
        "LogEmploi", "Densite_pop", "Entropie", "DensiteInter",
        "Long_arterePS", "Artere", "NB_voies5")

for(v in vars){
```

```
plotResiduals(sim_res, data2[[v]], main = v)
}
```

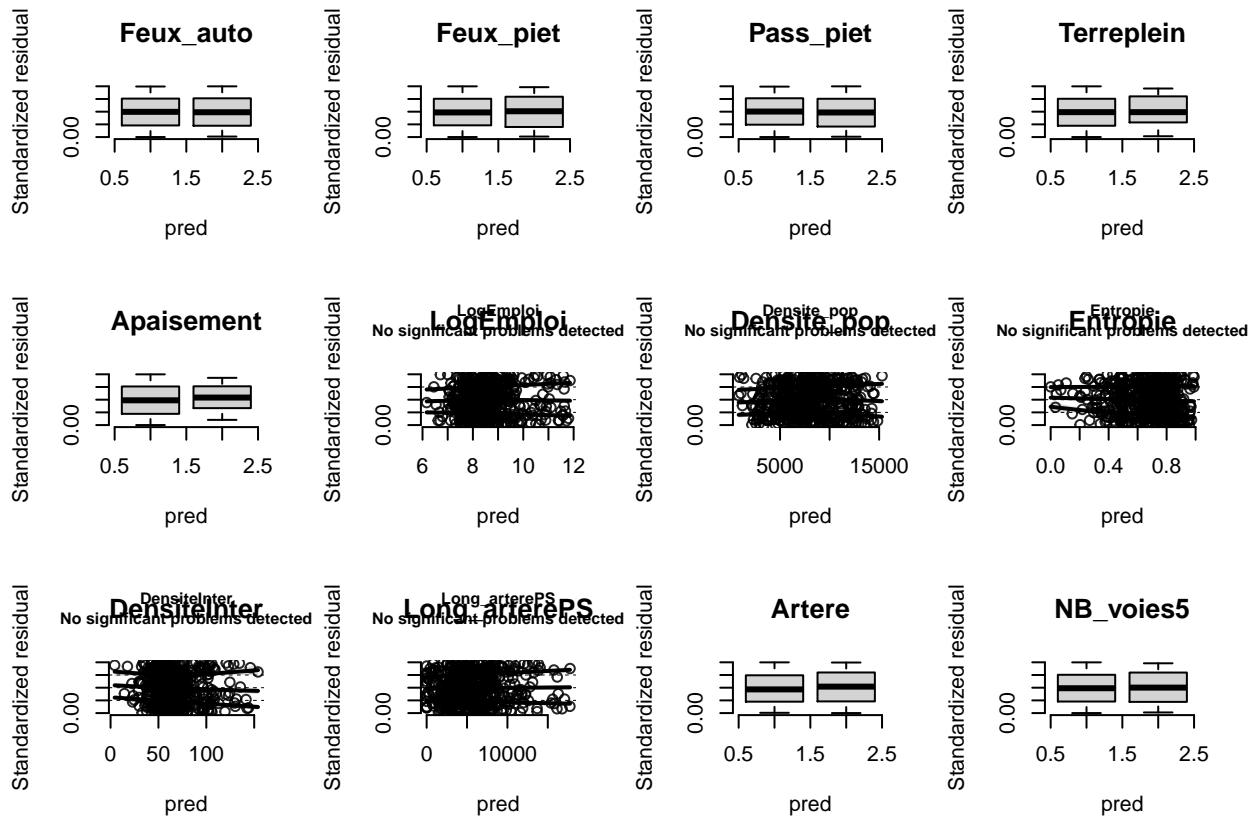


FIG. 1.30 : Comparaison des résidus simulés et de chaque variable indépendante

Maintenant que l'ensemble des diagnostics a été effectué, nous pouvons passer à la vérification de la qualité d'ajustement.

Vérifier la qualité d'ajustement

Pour le calcul des pseudo R^2 , notez qu'il n'existe pas à proprement parler de loglikelihood pour les quasi-distributions. Pour contourner ce problème, il est possible d'utiliser le loglikelihood d'un simple modèle de Poisson (puisque les coefficients ne changent pas), mais il est important de garder à l'esprit que ces pseudo R^2 seront d'autant plus faible que la surdispersion originale était forte

```
modelnull <- glm(Nbr_acci ~ 1,
                   family = poisson(link="log"),
                   data = data2)

rsqs(loglike.full = logLik(modele),
      loglike.null = logLik(modelnull),
      full.deviance = deviance(modele),
      null.deviance = deviance(modelnull),
      nb.params = modele$rank,
      n = nrow(data2))

## `$deviance expliquee'
## [1] 0.4805998
```

```

## 
## $`MacFadden ajuste`
## 'log Lik.' 0.3258375 (df=13)
## 
## $`Cox and Snell`
## 'log Lik.' 0.7789704 (df=13)
## 
## $Nagelkerke
## 'log Lik.' 0.787958 (df=13)

```

Le modèle parvient ainsi à expliquer 48% de la déviance totale, il obtient un R² ajusté de MacFadden de 0,33 et un R² de Cox et Snell de 0,78.

```

# calcul du RMSE
sqrt(mean((predict(modele2, type = "response") - data2$Nbr_acci)**2))

## [1] 1.838026

```

```

# nombre moyen d'accidents
mean(data2$Nbr_acci)

```

```
## [1] 1.503984
```

L'erreur quadratique moyenne du modèle est de 1,86, ce que signifie qu'en moyenne le modèle se trompe d'environ deux accidents pour chaque intersection. Cette valeur est relativement élevée si nous la comparons avec le nombre moyen d'accident : 1,5. Ceci s'explique certainement pas le grand nombre de zéros dans la variable Y qui tendent à tirer les prédictions vers le bas.

Interprétation des résultats

L'ensemble des coefficients du modèle sont accessibles via la fonction `summary`. Puisque la fonction de lien du modèle est la fonction `log`, il est pertinent de convertir les coefficients avec la fonction `exp` afin de pouvoir les interpréter sur l'échelle originale (nombre d'accident) plutôt que l'échelle `log` (`log(nombre d'accident)`). N'oubliez pas que ces effets sont multiplicatifs une fois transformés avec la fonction `exp`. Nous pouvons également utiliser les erreurs standards pour calculer des intervalles de confiance à 95% des exponentiels des coefficients. Le tableau 1.22 présente l'ensemble des informations pertinentes pour l'interprétation des résultats.

```

# calcule des coefficients en exponentiel et des intervalles de confiance
tableau <- summary(modele2)$coefficients

coeffs <- tableau[,1]
err.std <- tableau[,2]
expcoeff <- exp(coeffs)
exp2.5 <- exp(coeffs - 1.96*err.std)
exp975 <- exp(coeffs - 1.96*err.std)
pvals <- tableau[,4]

cbind(coeffs,err.std,expcoeff,exp2.5,exp975,pvals)

```

	coeffs	err.std	expcoeff	exp2.5	exp975
## (Intercept)	-3.677249e+00	6.534370e-01	0.02529246	0.00702707	0.00702707

```

## Feux_auto      1.097796e+00 2.201903e-01 2.99755257 1.94687022 1.94687022
## Feux_piet      3.282706e-01 1.244404e-01 1.38856462 1.08802851 1.08802851
## Pass_piet      3.395237e-01 2.349762e-01 1.40427862 0.88600764 0.88600764
## Terreplein     -3.590382e-01 2.171316e-01 0.69834770 0.45629486 0.45629486
## Apaisement     2.877035e-01 2.031770e-01 1.33336184 0.89536535 0.89536535
## LogEmploi      2.315611e-01 9.671167e-02 1.26056636 1.04290097 1.04290097
## Densite_pop    1.021922e-04 1.757475e-05 1.000010220 1.000006775 1.000006775
## Entropie       -4.166887e-01 3.779268e-01 0.65922613 0.31429439 0.31429439
## DensiteInter   2.228111e-03 2.700742e-03 1.00223059 0.99693935 0.99693935
## Long_arterePS  1.093341e-05 2.680656e-05 1.000001093 0.99995839 0.99995839
## Artere         2.848286e-02 1.426850e-01 1.02889238 0.77788248 0.77788248
## NB_voies5      6.382048e-01 1.287830e-01 1.89307928 1.47077578 1.47077578
##                               pvals
## (Intercept) 3.082557e-08
## Feux_auto   8.588714e-07
## Feux_piet   8.606077e-03
## Pass_piet   1.491186e-01
## Terreplein  9.886053e-02
## Apaisement  1.574051e-01
## LogEmploi   1.702569e-02
## Densite_pop 1.097698e-08
## Entropie    2.707587e-01
## DensiteInter 4.097742e-01
## Long_arterePS 6.835526e-01
## Artere      8.418604e-01
## NB_voies5   9.952088e-07

```

Parmi les variables décrivant les aménagements de l'intersection, nous constatons que la présence d'un feu de circulation et d'un feu de traversée pour les piétons multiplient le nombre attendu d'accidents à une intersection par 3 et 1,39. À contraria, la présence d'un passage piéton, d'un terre-plein ou de mesures d'apaisement n'ont pas d'effets significatifs (valeurs de $p > 0.05$). Concernant les variables décrivant l'environnement à proximité des intersections, nous observons que la concentration d'emploi et la densité de population contribuent toutes les deux à augmenter le nombre d'accident à une intersection, bien que leurs effets soient limités. Enfin, la présence d'une rue à cinq voies à l'intersection augmente le nombre d'accidents attendu à l'intersection de 89%. Nous ne détaillons pas plus loin les résultats car nous utilisons

TAB. 1.22 : Résultats du modèle de quasi-Poisson

Variable	Coeff.	exp(Coeff.)	Val.p	IC 2,5% exp(Coeff.)	IC 97,5% exp(Coeff.)	Sign.
Intercept	-3.68	0.03	<0.001	0.01	0.09	***
Feux_auto	1.10	3	<0.001	1.97	4.66	***
Feux_piet	0.33	1.39	0.009	1.09	1.79	**
Pass_piet	0.34	1.40	0.149	0.88	2.20	
Terreplein	-0.36	0.70	0.099	0.44	1.05	.
Apaisement	0.29	1.33	0.157	0.88	1.95	
LogEmploi	0.23	1.26	0.017	1.04	1.52	*
Densite_pop	<0.01	1.00	<0.001	1.00	1.00	***
Entropie	-0.42	0.66	0.271	0.32	1.39	
DensiteInter	<0.01	1.00	0.410	1.00	1.01	
Long_arterePS	<0.01	1.00	0.684	1.00	1.00	
Artere	0.03	1.03	0.842	0.78	1.36	
NB_voies5	0.64	1.89	<0.001	1.46	2.44	***

le même jeu de données dans les prochaines sections.

1.2.2.2 Le modèle Négatif Binomial

Dans le cas où une variable de comptage serait marquée par une sur ou sous dispersion, la distribution de Poisson n'est pas en mesure de capturer efficacement sa variance. Pour contourner ce problème, il est possible d'utiliser la distribution Négative Binomiale plutôt que la distribution de Poisson. Cette distribution peut être décrite comme une généralisation de la distribution de Poisson : elle inclut un second paramètre θ contrôlant la dispersion. L'intérêt premier de ce changement de distribution est que l'interprétation des paramètres est la même pour les deux modèles, tout en contrôlant directement l'effet d'une potentielle sur-dispersion.

1.2.2.2.1 Conditions d'applications

Les conditions d'application d'un modèle Négatif Binomial sont presque les mêmes que celle d'un modèle de Poisson. La seule différence est que la condition d'absence de sur ou sous-dispersion est remplacée une condition de respect du lien moyenne-variance. En effet, dans un modèle binomial, le paramètre de dispersion θ est combiné avec μ pour exprimer la variance de la distribution. Dans le package `mgcv` que nous utilisons dans l'exemple, le lien entre μ , θ et la variance est le suivant :

$$\text{variance} = \mu + \mu^{\frac{2}{\theta}} \quad (1.19)$$

Il s'agit donc d'un modèle hétéroscédastique, sa variance n'est pas fixe, mais fonction de sa propre moyenne. Si la moyenne augmente, la variance augmente (comme pour un modèle de Poisson), et l'intensité de cette augmentation est contrôlée par le paramètre θ . Si cette condition n'est pas respectée, l'analyse des résidus simulés révèlera un problème de dispersion.

1.2.2.2.2 Exemple appliqué dans R

Dans l'exemple précédent avec le modèle de Poisson, nous avions observé une certaine sur-dispersion que nous avions contournée en utilisant un modèle de quasi-poisson. Dans l'article original, les auteurs avaient opté pour un modèle Négatif Binomial, ce que proposons de faire ici. Les variables utilisées sont les même que pour le modèle de Poisson. Nous utilisons le package `mgcv` et sa fonction `gam` pour ajuster le modèle.

Vérification des conditions d'application

Nous avions vu précédemment que nos variables indépendantes n'étaient pas marquées par une multicolinéarité forte. Il n'est pas nécessaire de recalculer les valeurs de VIF puisque nous utilisons les mêmes données. La première étape du diagnostic est donc de calculer les distances de Cook.

TAB. 1.23 : Carte d'identité du modèle Négatif Binomial

Type de variable dépendante	Variable de comptage
Distribution utilisée	Négative Binomiale
Formulation	$Y \sim NB(\mu, \theta)$ $g(\mu) = \beta_0 + \beta X$ $g(x) = \log(x)$
Fonction de lien	\log
Paramètre modélisé	μ
Paramètres à estimer	β_0, β et θ
Conditions d'applications	Absence d'excès de zéros, Respect du lien variance-moyenne

```

library(mgcv)

# chargement des données
data_accidents <- read.csv("data/glm/accident_pietons.csv")

# ajustement d'une première version du modèle
modelnb <- gam(Nbr_acci ~ Feux_auto + Feux_piet + Pass_piet +
                 Terreplein + Apaisement +
                 LogEmploi + Densite_pop + Entropie + DensiteInter +
                 Long_arterePS + Artere + NB_voies5,
                 family = nb(link="log"),
                 data = data_accidents)

# calcul et affichage des distances de Cook
cooksd <- cooks.distance(modelnb)

df <- data.frame(
  cook = cooksd,
  oid = 1:length(cooksd)
)

ggplot(data = df)+
  geom_point(aes(x = oid, y = cook), size = 0.5, color = rgb(0.4,0.4,0.4)) +
  labs(x = "", y = "distance de Cook")+
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())

```

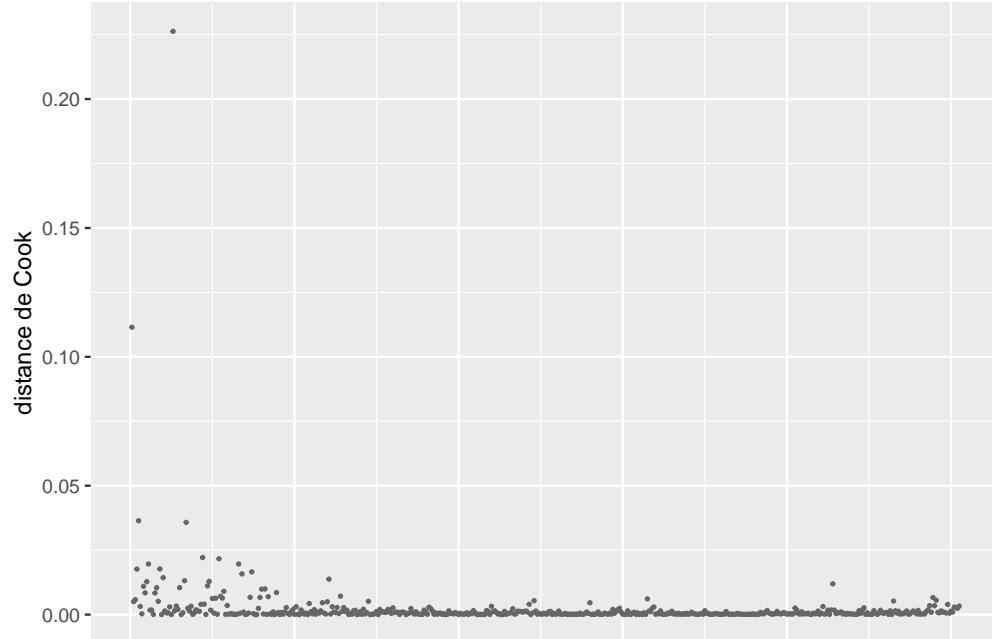


FIG. 1.31 : Distances de Cook pour le modèle négatif binomial

Nous observons dans la figure 1.31 que deux observations se distinguent très nettement des autres.

```

cas_etrange <- subset(data_accidents, cooksd > 0.03)
print(cas_etrange)

```

```

##      X NumInter NOUNIK Nbr_acci Feux_auto Feux_piet Pass_piet Terreplein
## 1    1     7839    419      19       1       1       1       1
## 5    5     8054    433      12       1       0       1       0
## 26   26     8442    136      7        0       0       1       0
## 34   34     8049    129      6        0       0       1       0
##      Apaisement EmpTotBuffer Densite_pop Entropie DensiteInter Long_arterePS
## 1          0     7208.538  5980.923 0.8073926  42.41597  6955.00
## 5          0     8585.350  8655.430 0.7607852  89.11495  6412.27
## 26         0     1342.625 2751.012 0.0000000  73.35344  2849.66
## 34         0    12516.410  8950.942 0.4300549  74.91879  8443.01
##      Artere NB_voies5 log_acci catego_acci catego_acci2 Arret VAG sum_app
## 1      1     1 2.995732           1       1       0       1       4
## 5      0     0 2.564949           1       1       0       1       4
## 26     0     0 2.079442           1       1       1       1       3
## 34     1     0 1.945910           1       1       1       0       3
##      LogEmploi AccOrdinal PopHa
## 1  8.883021           2 5.980923
## 5  9.057813           2 8.655430
## 26 7.202382           2 2.751012
## 34 9.434796           2 8.950942

```

Il s'agit à nouveau des quatre observations avec un grand nombre d'accidents. Nous décidons de les retirer du jeu de données pour ne pas fausser les résultats concernant l'ensemble des autres intersections. Dans une analyse plus détaillée, il serait judicieux de chercher à comprendre pourquoi ces quatre observations sont particulièrement accidentogènes.

```

data2 <- subset(data_accidents, cooksd < 0.03)

# ajustement d'une première version du modèle
modelnb <- gam(Nbr_acci ~ Feux_auto + Feux_piet + Pass_piet +
                 Terreplein + Apaisement +
                 LogEmploi + Densite_pop + Entropie + DensiteInter +
                 Long_arterePS + Artere + NB_voies5,
                 family = nb(link="log"),
                 data = data2)

# calcul et affichage des distances de Cook
cooksd <- cooks.distance(modelnb)

df <- data.frame(
  cook = cooksd,
  oid = 1:length(cooksd)
)

ggplot(data = df) +
  geom_point(aes(x = oid, y = cook), size = 0.5, color = rgb(0.4,0.4,0.4)) +
  labs(x = "", y = "distance de Cook") +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())

```

Après avoir retiré ces deux observations, les distances de Cook (figure 1.31) ne révèlent plus d'observations fortement influentes dans le modèle. La prochaine étape du diagnostic est donc d'analyser les résidus simulés

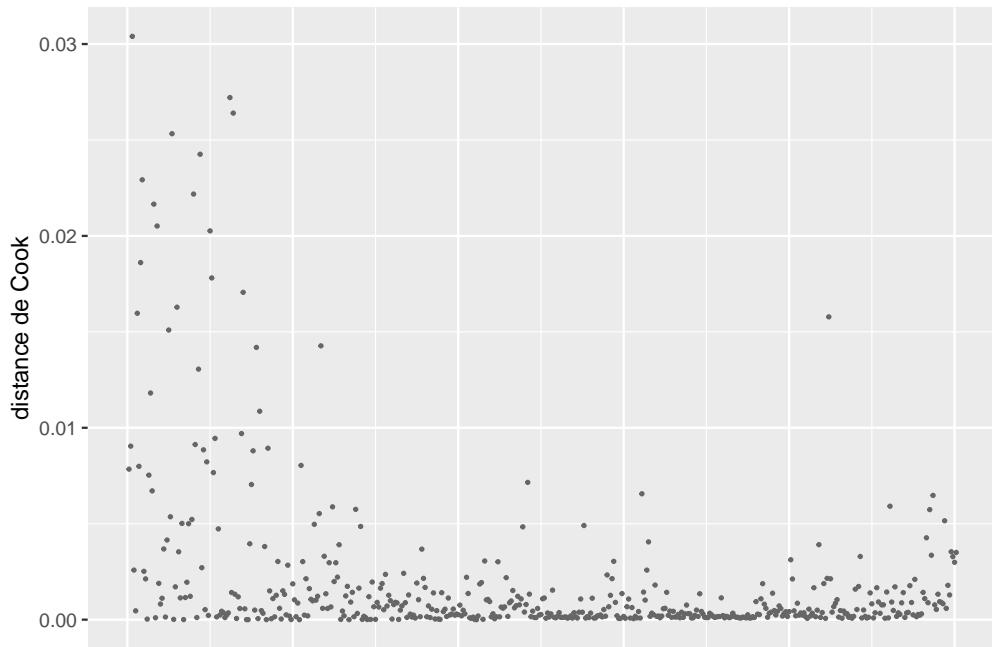


FIG. 1.32 : Distances de Cook pour le modèle négatif binomial (après avoir retiré deux observations fortement influentes)

```

# extraction de la valeur de theta
theta <- modelnb$family$getTheta(T)
nsim <- 1000
# extraction des valeurs predites par le modele
mus <- predict(modelnb, type = "response")

# calcul des simulations
cols <- lapply(1:length(mus), function(i){
  mu <- mus[[i]]
  sims <- rnbineg(n = nsim, mu = mu, size = theta)
  return(sims)
})
mat_sims <- do.call(rbind, cols)

# calcul des residus simulés
sim_res <- createDHARMA(simulatedResponse = mat_sims,
                           observedResponse = data2$Nbr_acci,
                           fittedPredictedResponse = mus,
                           integerResponse = T)
# affichage du diagnostic
plot(sim_res)

```

La figure 1.33 présentant le diagnostic des résidus simulés montre que ces derniers suivent bien une distribution uniforme, aucun problème de dispersion ni de valeur aberrantes. La figure 1.34 permet de comparer la distribution originale de la variable Y et les simulations issues du modèle (intervalles de confiance représentés en bleu). On constate que le modèle parvient bien à reproduire la distribution originale, et ce même pour les valeurs les plus à droites de la distribution.

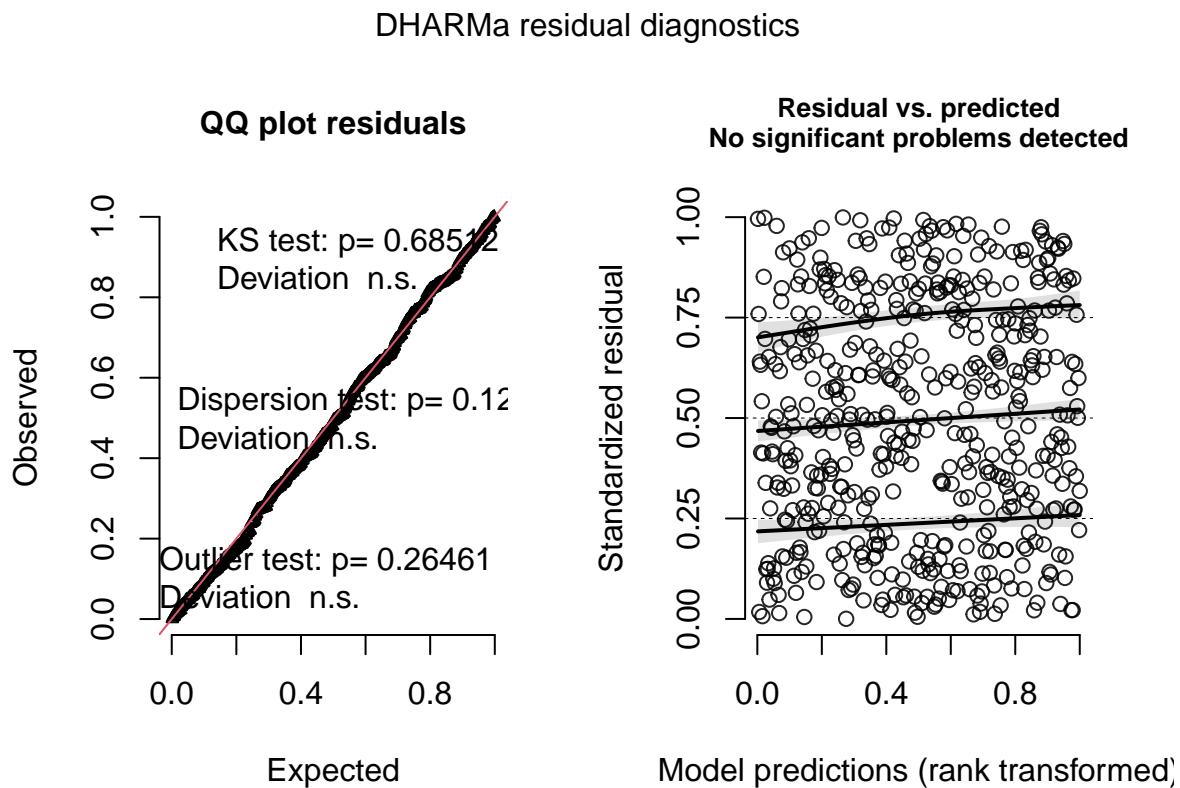


FIG. 1.33 : Diagnostic général des résidus simulés pour le modèle Négatif Binomial

```
# extraction des valeurs predites par le modele
mus <- predict(modelnb, type="response")

# generation de 1000 simulation pour chaque prediction
theta <- modelnb$family$getTheta(T)
nsim <- 1000
cols <- lapply(1:length(mus), function(i){
  mu <- mus[[i]]
  sims <- round(rnbinom(n = nsim, mu = mu, size = theta))
  return(sims)
})
mat_sims <- do.call(rbind, cols)

# preparation des donnees pour le graphique (valeurs reelles)
counts <- data.frame(table(data2$Nbr_acc))
names(counts) <- c("nb_accident", 'frequence')
counts$nb_accident <- as.numeric(as.character(counts$nb_accident))
counts$prop <- counts$frequence / sum(counts$frequence)

# preparation des donnees pour le graphique (valeurs simulees)
df1 <- data.frame(count = 0:25)

count_sims <- lapply(1:nsim, function(i){
  sim <- mat_sims[,i]
  cnt <- data.frame(table(sim))
  df2 <- merge(df1,cnt, by.x="count", by.y = "sim", all.x = T, all.y=F)
  df2$Freq <- ifelse(is.na(df2$Freq), 0,df2$Freq)
})
```

```

return(df2$Freq)
})

count_sims <- do.call(cbind, count_sims)

df_sims <- data.frame(
  val = 0:25,
  med = apply(count_sims, MARGIN = 1, median),
  lower = apply(count_sims, MARGIN = 1, quantile, probs = 0.025),
  upper = apply(count_sims, MARGIN = 1, quantile, probs = 0.975)
)

# affichage du graphique
ggplot() +
  geom_bar(aes(x=nb_accident, weight = frequence), width = 0.6, data = counts) +
  geom_errorbar(aes(x = val, ymin = lower, ymax = upper),
                data = df_sims, color = "blue", width = 0.6) +
  geom_point(aes(x = val, y = med), color = "blue", size = 1.3, data = df_sims) +
  scale_x_continuous(limits = c(-0.5,7), breaks = c(0:7)) +
  xlim(-1,12) +
  labs(subtitle = "",
       x = "nombre d'accidents",
       y = "nombre d'occurrence")

```

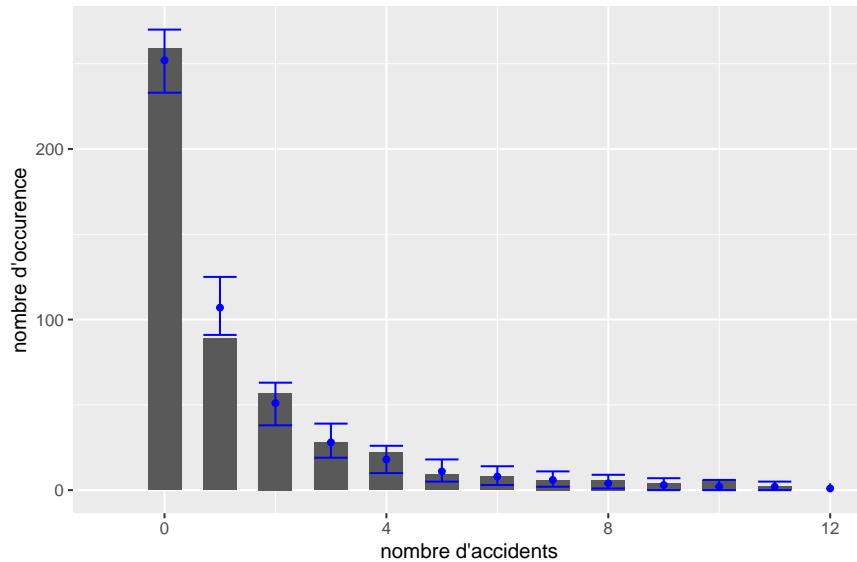


FIG. 1.34 : Comparaison de la distribution originale et des simulations pour le modèle Négatif binomial

À titre de comparaison, nous pouvons à nouveau réaliser le graphique permettant de visualiser si la variance attendue par le modèle est proche de celle effectivement observée dans les données. Nous avons constaté avec ce graphique lorsque nous ajustions un modèle de Poisson que la variance des données était trop grande comparativement à celle attendue par le modèle (@ref :surdispoiss)

```

# extraction des préditions du modèle
mus <- predict(modelnb, type = "response")

# création d'un dataframe pour contenir la prédition et les vraie valeurs

```

```

df1 <- data.frame(
  mus = mus,
  reals = data2$Nbr_acci
)

# calcul de l'intervalle de confiance à 95% selon la distribution de Poisson
# et stockage dans un second dataframe
seqa <- seq(0,round(max(mus)),1)
df2 <- data.frame(
  mus = seqa,
  lower = qnbinom(p = 0.025, mu = seqa, size = theta),
  upper = qnbinom(p = 0.975, mu = seqa, size = theta)
)

# affichage des valeurs réelles et prédites (en rouge)
# et de leur variance selon le modèle (en noir)
ggplot() +
  geom_errorbar(data = df2,
    mapping = aes(x = mus, ymin = lower, ymax = upper),
    width = 0.2, color = rgb(0.4,0.4,0.4)) +
  geom_point(data = df1,
    mapping = aes(x = mus, y = reals),
    color ="red", size = 0.5) +
  labs(x = 'valeurs prédites',
       y = "valeurs réelles")

```

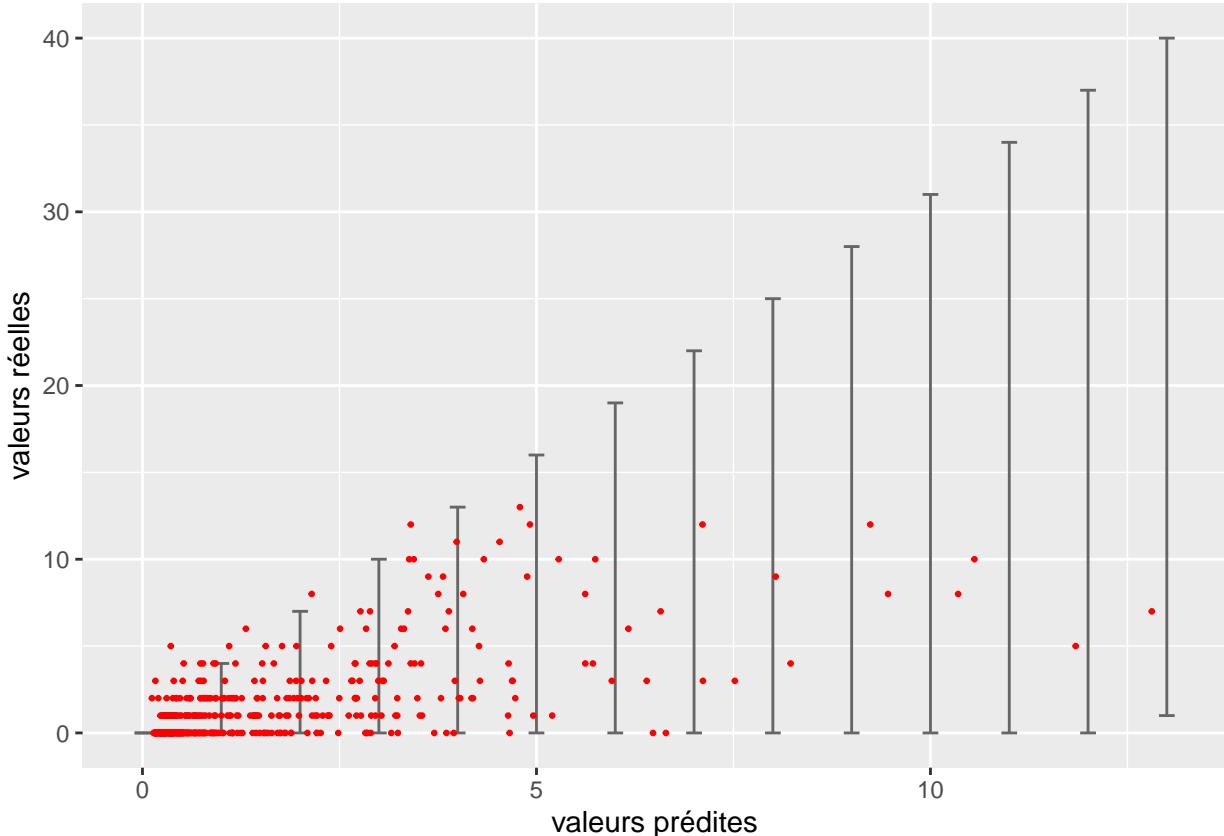


FIG. 1.35 : Représentation de la sur-dispersion des données dans le modèle de Poisson

Nous pouvons ainsi constater à la figure (@ref :surdispnb) que le modèle Négatif Binomial autorise une variance bien plus large que le modèle de Poisson et est ainsi mieux ajusté aux données.

Vérification de la qualité d'ajustement

```
# calcul des pseudo R2
rsqs(loglike.full = logLik(modelnb),
      loglike.null = logLik(modelnull),
      full.deviance = deviance(modelnb),
      null.deviance = modelnb$null.deviance,
      nb.params = modelnb$rank,
      n = nrow(data2))

## `$deviance expliquee`
## [1] 0.458052
##
## $`MacFadden ajuste`
## 'log Lik.' 0.384353 (df=14)
##
## $`Cox and Snell`
## 'log Lik.' 0.8304773 (df=14)
##
## $Nagelkerke
## 'log Lik.' 0.8399731 (df=14)

# calcul du RMSE
sqrt(mean((predict(modelnb, type = "response") - data2$Nbr_acci)**2))

## [1] 1.825278
```

Le modèle parvient à expliquer 45% de la déviance , il obtient un R^2 ajusté de McFadden de 0.14 et un R^2 de Nagelkerke de 0.42. L'erreur moyenne quadratique de la prédiction est de 1,9, ce qui est identique au modèle de Poisson ajusté précédemment.

Interprétation des résultats

Il est possible d'accéder à l'ensemble des coefficients du modèle via la fonction `summary`. À nouveau, les coefficient doivent être convertis avec la fonction exponentielle (du fait de la fonction de lien `log`) et interprétés comme des effets multiplicatifs.

Le tableau 1.24 présente les coefficients estimés par le modèle. Les résultats sont très similaires à ceux du modèle de quasi-Poisson original. Nous notons cependant que la variable présence d'un feu pour piéton n'est plus significative au seuil 0,05.

1.2.2.3 Le modèle de poisson avec excès fixe de zéros

Dans le cas où la variable Y comprendrait significativement plus de zéros que ce que suppose une distribution de poisson, il est possible d'utiliser la distribution de poisson avec excès de zéros. Pour rappel, cette distribution ajoute un paramètre p contrôlant pour la proportion de zéros dans la distribution. Du point de vue conceptuel, cela revient à formuler l'hypothèse suivante : dans les données que l'on a observé, deux processus distincts sont à l'œuvre. Le premier est issu d'une distribution de poisson et le second produit des zéros qui s'ajoutent aux données. Les zéros produits par la distribution

TAB. 1.24 : Résultats du modèle Négatif Binomial

Variable	Coeff.	exp(Coeff.)	Val.p	IC 2,5% exp(Coeff.)	IC 97,5% exp(Coeff.)	Sign.
Intercept	-3.88	0.02	<0.001	0.01	0.08	***
Feux_auto	1.13	3.10	<0.001	2.03	4.71	***
Feux_piet	0.35	1.42	0.016	1.06	1.90	*
Pass_piet	0.22	1.24	0.300	0.83	1.88	
Terreplein	-0.34	0.71	0.155	0.44	1.14	
Apaisement	0.24	1.27	0.315	0.79	2.03	
LogEmploi	0.23	1.26	0.025	1.03	1.55	*
Densite_pop	<0.01	1.00	<0.001	1.00	1.00	***
Entropie	-0.17	0.84	0.669	0.38	1.86	
DensiteInter	<0.01	1.00	0.925	0.99	1.01	
Long_arterePS	<0.01	1.00	0.587	1.00	1.00	
Artere	0.11	1.11	0.497	0.82	1.51	
NB_voies5	0.70	2.01	<0.001	1.48	2.75	***

de poisson sont appelés les vrais zéros, alors que ceux produits par le second phénomènes sont appelés les faux zéros.

Dans cette formulation, p est fixé. Nous n'avons donc aucune information sur ce qui produit les zéros supplémentaires, mais seulement leur proportion totale dans le jeu de données.

1.2.2.3.1 Interprétation des paramètres

L'interprétation des paramètres est identique à celle d'un modèle de Poisson. Le paramètre p représente la proportion de zéros dans la variable Y une fois que sont contrôlés les variables indépendantes.

1.2.2.3.2 Exemple appliqué dans R

La variable de comptage des accidents des piétons que nous avons utilisées dans les deux exemples précédents semble être une bonne candidate pour une distribution de poisson avec excès de zéros. En effet, nous avions pu constater une sur-dispersion dans le modèle de Poisson original, ainsi qu'un nombre important d'intersections sans accident. Tentons donc d'améliorer notre modèle en ajustant un excès fixe de zéros. Nous utilisons la fonction `gamlss` du package `gamlss`.

```
library(gamlss)

modelzi <- gamlss(formula = Nbr_acci ~ Feux_auto + Feux_piet + Pass_piet +
                    Terreplein + Apaisement + LogEmploi + Densite_pop +
                    Entropie + DensiteInter + Long_arterePS + Artere + NB_voies5,
```

TAB. 1.25 : Carte d'identité du modèle de Poisson avec excès fixe de zéros

Type de variable dépendante	Variable de comptage
Distribution utilisée	Poisson avec excès de zéros
Formulation	$Y \sim ZIP(\mu, \theta)$ $g(\lambda) = \beta_0 + \beta X$ $g(x) = \log(x)$
Fonction de lien	\log
Paramètre modélisé	λ
Paramètres à estimer	β_0, β et p
Conditions d'applications	Absence de surdispersion

```

    sigma.formula = ~1,
    family = ZIP(mu.link = "log", sigma.link="logit"),
    data = data_accidents)

## GAMLSS-RS iteration 1: Global Deviance = 1516.53
## GAMLSS-RS iteration 2: Global Deviance = 1514.656
## GAMLSS-RS iteration 3: Global Deviance = 1514.52
## GAMLSS-RS iteration 4: Global Deviance = 1514.508
## GAMLSS-RS iteration 5: Global Deviance = 1514.506
## GAMLSS-RS iteration 6: Global Deviance = 1514.506

modelnull <- glm(formula = Nbr_acci ~ 1,
                  family = poisson(link="log"),
                  data = data_accidents)

# calcule de la deviance expliquée

1 - deviance(modelzi) / deviance(modelnull)

## [1] 0.08267513

# calcul de la probabilité de base p d'être un faux 0
# en appliquant l'inverse de la fonction logistique
exp(-1.4376) / (1+exp(-1.4376))

## [1] 0.1919173

```

Nous constatons immédiatement que le modèle avec excès fixe de zéro est peu ajusté aux données. Cette version du modèle ne parvient à capturer que 8% de la déviance, ce qui s'explique facilement car nous n'avons donné aucune variable au modèle pour distinguer les vrais et les faux zéros. Pour cela, nous devons passer au prochain modèle : poisson avec excès ajusté de zéros. Notons tout de même que d'après ce modèle, 19% des observations seraient des faux zéros.

1.2.2.4 Le modèle de poisson avec excès ajusté de zéros

Nous avons vu dans le modèle précédent que l'excès de zéro était conceptualisé comme la combinaison de deux phénomènes, l'un issu d'une distribution de poisson que l'on souhaite modéliser, et le second générant des zéros supplémentaires. Il est possible d'aller plus loin que de simplement contrôler la proportion de zéro supplémentaire en modélisant explicitement ce second processus en ajoutant une deuxième équation au modèle. Cette deuxième équation a pour enjeu de modéliser p (la proportion de 0) à l'aide de variables indépendantes, ces dernières pouvant se retrouver dans les deux parties du modèle. Pour résumer, un modèle de Poisson avec excès ajusté de zéro revient à combiner un modèle logistique binomial avec un modèle de Poisson. La partie binaire du modèle s'occupe des faux zéros, et la partie Poisson du reste des données. L'idée étant que pour chaque observation, le modèle évalue sa probabilité d'être un faux zéro (partie binomiale), avant d'évaluer son impact dans le modèle de Poisson.

1.2.2.4.1 Interprétation des paramètres

L'interprétation des paramètres β_0 et β est identique à celle d'un modèle de Poisson. Les paramètres α_0 et α est identique à celle d'un modèle binomial. Plus spécifiquement, ces derniers paramètres modélisent

TAB. 1.26 : Carte d'identité du modèle de Poisson avec excès ajusté de zéros

Type de variable dépendante	Variable de comptage
Distribution utilisée	Poisson avec excès de zéros
Formulation	$Y \sim ZIP(\mu, \theta)$ $g(\lambda) = \beta_0 + \beta X$ $s(p) = \alpha_0 + \alpha_X$ $g(x) = \log(x)$ $s(x) = \log\left(\frac{x}{1-x}\right)$
Fonction de lien	log et logistique
Paramètre modélisé	λ et p
Paramètres à estimer	β_0, β, α_0 et α
Conditions d'applications	Absence de surdispersion

la probabilité d'observer des valeurs supérieures à 0.

1.2.2.4.2 Exemple appliqu  

Nous avions vu dans l'exemple précédent que l'utilisation du modèle avec excès fixe de zéros pour les données d'accident des pi  tons aux intersections ne donnait pas de r  sultats satisfaisants. Nous tentons ici d'améliorer le mod  le en ajoutant les pr  dicteurs significatifs du mod  le de Poisson dans la seconde quation de r  gression destin  e dtecter les z  ros.

V  rification des conditions d'applications

Pour un mod  le de Poisson avec exc  s de 0, il n'est pas possible de calculer de distances de Cook. Nous devons donc directement passer  l'analyse des r  siduals simul  s

```
# ajuster une premi  re version du mod  le
modelza <- gamlss(formula = Nbr_acci ~ Feux_auto + Feux_piet + Pass_piet +
                    Terreplein + Apaisement + LogEmploi + Densite_pop +
                    Entropie + DensiteInter + Long_arterePS + Artere + NB_voies5,
                    sigma.formula = ~1 + Feux_auto + Feux_piet + Densite_pop + NB_voies5,
                    family = ZIP(mu.link = "log", sigma.link="logit"),
                    data = data_accidents)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 1505.155
## GAMLSS-RS iteration 2: Global Deviance = 1488.658
## GAMLSS-RS iteration 3: Global Deviance = 1483.304
## GAMLSS-RS iteration 4: Global Deviance = 1482.085
## GAMLSS-RS iteration 5: Global Deviance = 1481.868
## GAMLSS-RS iteration 6: Global Deviance = 1481.832
## GAMLSS-RS iteration 7: Global Deviance = 1481.827
## GAMLSS-RS iteration 8: Global Deviance = 1481.825
## GAMLSS-RS iteration 9: Global Deviance = 1481.825
```

```
# extraire la pr  dition des valeurs lambda
lambdas <- predict(modelza, type = "response", what = "mu")

# extraire la pr  dition des valeurs p
ps <- predict(modelza, type = "response", what = "sigma")

# calculer la combinaison de ces deux elements
```

```

preds <- lambdas * ps

# effectuer les 1000 simulations
nsim <- 1000
cols <- lapply(1:length(lambdas), function(i){
  lambda <- lambdas[[i]]
  p <- ps[[i]]
  sims <- rZIP(n = nsim, mu = lambda, sigma = p)
  return(sims)
})
mat_sims <- do.call(rbind, cols)

# calculer les residus simulés
sim_res <- createDHARMA(simulatedResponse = mat_sims,
                           observedResponse = data_accidents$Nbr_acci,
                           fittedPredictedResponse = preds,
                           integerResponse = T)

plot(sim_res)

```

DHARMA residual diagnostics

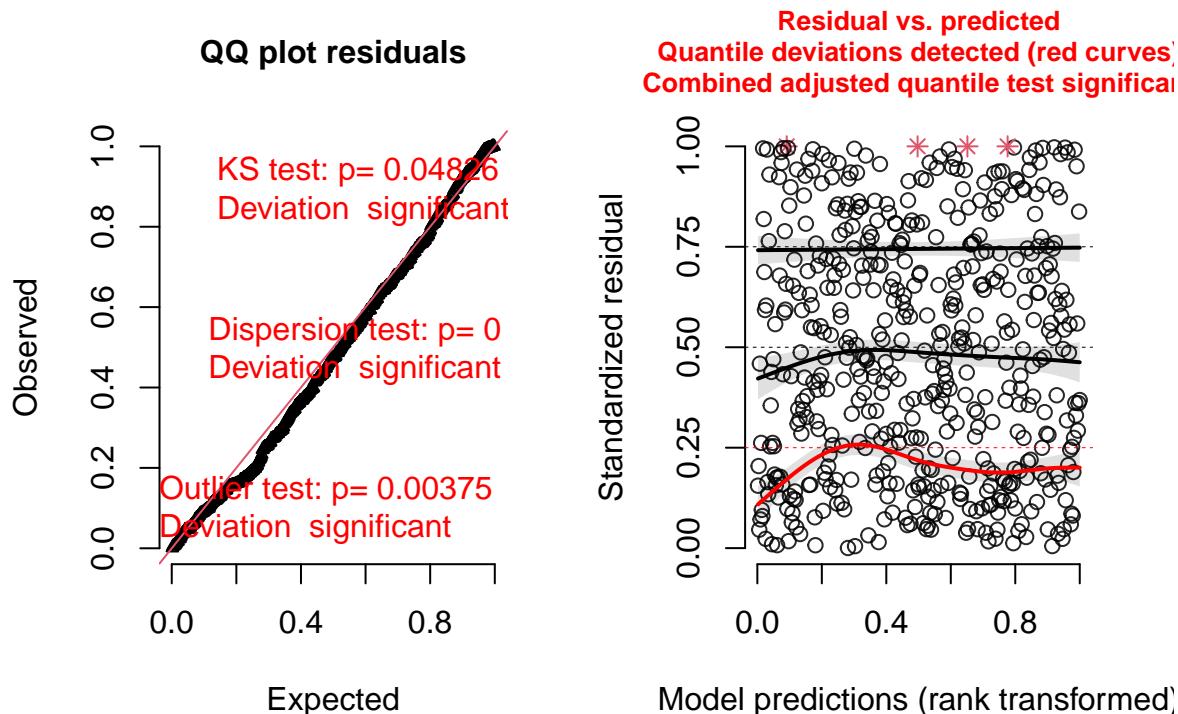


FIG. 1.36 : Diagnostic général des résidus simulés du modèle de Poisson avec excès de zéros ajusté

La figure 1.36 indique deux problèmes importants dans le modèle : la présence de valeurs aberrantes ainsi qu'un potentiel problème de dispersion. nous commençons donc par identifier ces valeurs aberrantes

```

#identification des outliers
isOutlier <- outliers(sim_res, return = "logical", lowerQuantile = 0.001,
                      upperQuantile = 0.999)
cas_étrange <- subset(data_accidents, isOutlier)

```

```
print(cas_étrange)
```

```
##          X NumInter NOUNIK Nbr_acci Feux_auto Feux_piet Pass_piet Terreplein
## 1      1    7839     419      19       1       1       1       1
## 5      5    8054     433      12       1       0       1       0
## 26     26    8442     136       7       0       0       1       0
## 44     44    5676      78       5       0       0       0       0
## 482    482   14429     583       0       0       0       0       0
##          Apaisement EmpTotBuffer Densite_pop Entropie DensiteInter Long_arterePS
## 1          0    7208.538    5980.923  0.8073926    42.41597   6955.00
## 5          0    8585.350    8655.430  0.7607852    89.11495   6412.27
## 26         0   1342.625   2751.012  0.0000000   73.35344   2849.66
## 44         0   4998.519   8090.478  0.7879618   66.86856   4517.65
## 482        0   1813.911   8988.260  0.4486079   60.93742   3821.78
##          Artere NB_voies5 log_acci catego_acci catego_acci2 Arret VAG sum_app
## 1          1      1 2.995732           1       1       0       1       4
## 5          0      0 2.564949           1       1       0       1       4
## 26         0      0 2.079442           1       1       1       1       3
## 44         0      0 1.791759           1       1       1       1       4
## 482        1      0 0.000000           0       0       0       0       3
##          LogEmploi AccOrdinal PopHa
## 1      8.883021           2 5.980923
## 5      9.057813           2 8.655430
## 26     7.202382           2 2.751012
## 44     8.516897           2 8.090478
## 482    7.503240           0 8.988260
```

Nous retirons des données six observations pouvant avoir une certaine influence sur le modèle. Après réajustement, la figure 1.37 nous informe que nous n'avons plus de valeurs aberrantes restantes ni de fort problème de dispersion. En revanche, le premier quantile des résidus tant à être plus faible que ce que l'on pourrait attendre d'une distribution uniforme. Cette observation laisse penser que le modèle a du mal à bien identifier les faux zéros. Ce résultat n'est pas étonnant car aucune variable n'avait été identifiée à cette fin dans l'article originale (Cloutier et al., 2014) qui utilisait un modèle Négatif Binomial.

```
data2 <- subset(data_accidents, isOutlier==FALSE)

# ajuster une première version du modèle
modelza <- gamlss(formula = Nbr_acci ~ Feux_auto + Feux_piet + Pass_piet +
                    Terreplein + Apaisement + LogEmploi + Densite_pop +
                    Entropie + DensiteInter + Long_arterePS + Artere + NB_voies5,
                    sigma.formula = ~1 + Feux_auto + Feux_piet + Densite_pop + NB_voies5,
                    family = ZIP(mu.link = "log", sigma.link="logit"),
                    data = data2)

## GAMLSS-RS iteration 1: Global Deviance = 1390.884
## GAMLSS-RS iteration 2: Global Deviance = 1381.199
## GAMLSS-RS iteration 3: Global Deviance = 1378.319
## GAMLSS-RS iteration 4: Global Deviance = 1377.422
## GAMLSS-RS iteration 5: Global Deviance = 1377.118
## GAMLSS-RS iteration 6: Global Deviance = 1377.015
```

```

## GAMLSS-RS iteration 7: Global Deviance = 1376.982
## GAMLSS-RS iteration 8: Global Deviance = 1376.972
## GAMLSS-RS iteration 9: Global Deviance = 1376.968
## GAMLSS-RS iteration 10: Global Deviance = 1376.967
## GAMLSS-RS iteration 11: Global Deviance = 1376.966

# extraire la prédiction des valeurs lambda
lambdas <- predict(modelza, type = "response", what = "mu")

# extraire la prédiction des valeurs p
ps <- predict(modelza, type = "response", what = "sigma")

# calculer la combinaison de ces deux éléments
preds <- lambdas * ps

# effectuer les 1000 simulations
nsim <- 1000
cols <- lapply(1:length(lambdas), function(i){
  lambda <- lambdas[[i]]
  p <- ps[[i]]
  sims <- rZIP(n = nsim, mu = lambda, sigma = p)
  return(sims)
})
mat_sims <- do.call(rbind, cols)

# calculer les résidus simulés
sim_res <- createDHARMA(simulatedResponse = mat_sims,
                           observedResponse = data2$Nbr_acci,
                           fittedPredictedResponse = preds,
                           integerResponse = T)
plot(sim_res)

```

Nous pouvons une fois encore comparer des simulations issues du modèle et la distribution originale de la variable Y . La figure 1.38 montre clairement que les simulations du modèle (en bleu) sont très éloignées dans la distribution originale (en gris), ce qui remet directement en question la pertinence de ce modèle.

```

# extraire la prédiction des valeurs lambda
lambdas <- predict(modelza, type = "response", what = "mu")

# extraire la prédiction des valeurs p
ps <- predict(modelza, type = "response", what = "sigma")

# génération de 1000 simulation pour chaque prédiction
nsim <- 1000
cols <- lapply(1:length(lambdas), function(i){
  lambda <- lambdas[[i]]
  p <- ps[[1]]
  sims <- round(rZIP(nsim, mu=lambda, sigma = p))
  return(sims)
})
mat_sims <- do.call(rbind, cols)

# préparation des données pour le graphique (valeurs réelles)
counts <- data.frame(table(data2$Nbr_acci))

```

DHARMA residual diagnostics

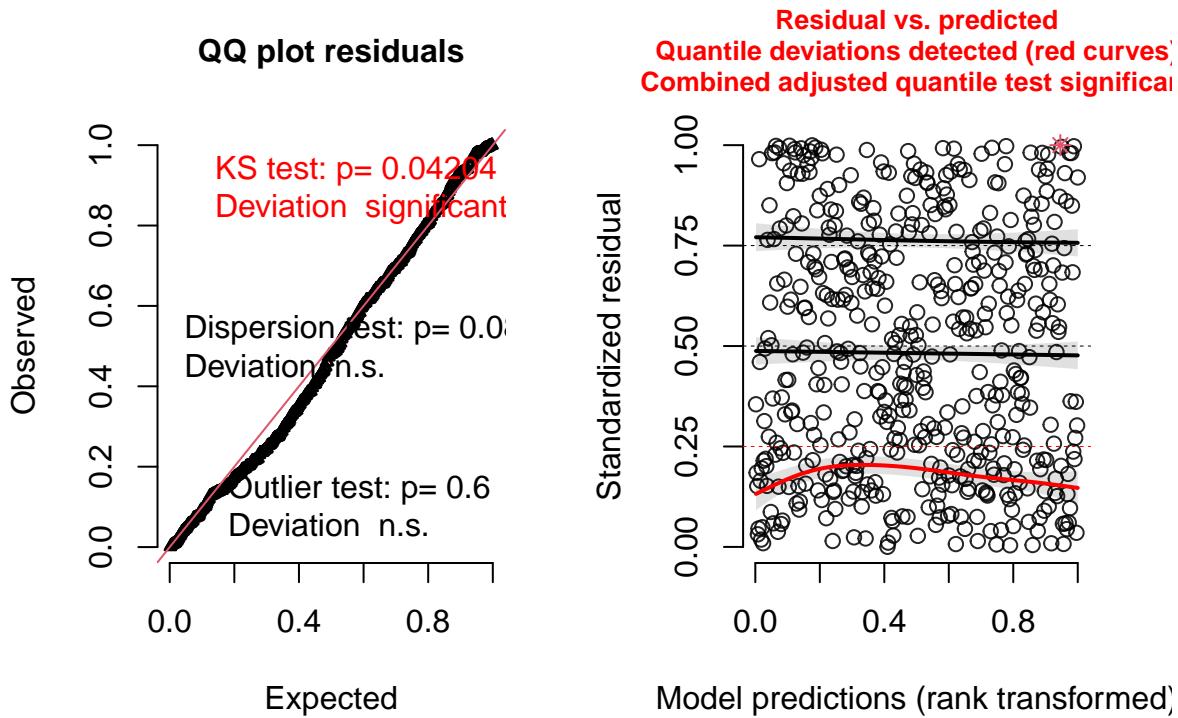


FIG. 1.37 : Diagnostic général des résidus simulés du modèle de Poisson avec excès de zéros ajusté (sans valeurs aberrantes)

```

names(counts) <- c("nb_accident", "frequence")
counts$nb_accident <- as.numeric(as.character(counts$nb_accident))
counts$prop <- counts$frequence / sum(counts$frequence)

# préparation des données pour le graphique (valeurs simulées)
df1 <- data.frame(count = 0:25)

count_sims <- lapply(1:nsim, function(i){
  sim <- mat_sims[,i]
  cnt <- data.frame(table(sim))
  df2 <- merge(df1,cnt, by.x="count", by.y = "sim", all.x = T, all.y=F)
  df2$Freq <- ifelse(is.na(df2$Freq), 0,df2$Freq)
  return(df2$Freq)
})

count_sims <- do.call(cbind,count_sims)

df_sims <- data.frame(
  val = 0:25,
  med = apply(count_sims, MARGIN = 1, median),
  lower = apply(count_sims, MARGIN = 1, quantile, probs = 0.025),
  upper = apply(count_sims, MARGIN = 1, quantile, probs = 0.975)
)

# affichage du graphique
ggplot() +

```

```

geom_bar(aes(x=nb_accident, weight = frequence), width = 0.6, data = counts) +
  geom_errorbar(aes(x = val, ymin = lower, ymax = upper),
    data = df_sims, color = "blue", width = 0.6) +
  geom_point(aes(x = val, y = med), color = "blue", size = 1.3, data = df_sims) +
  scale_x_continuous(limits = c(-0.5,7), breaks = c(0:7)) +
  xlim(-1,12) +
  labs(subtitle = "",
       x = "nombre d'accidents",
       y = "nombre d'occurrence")

```

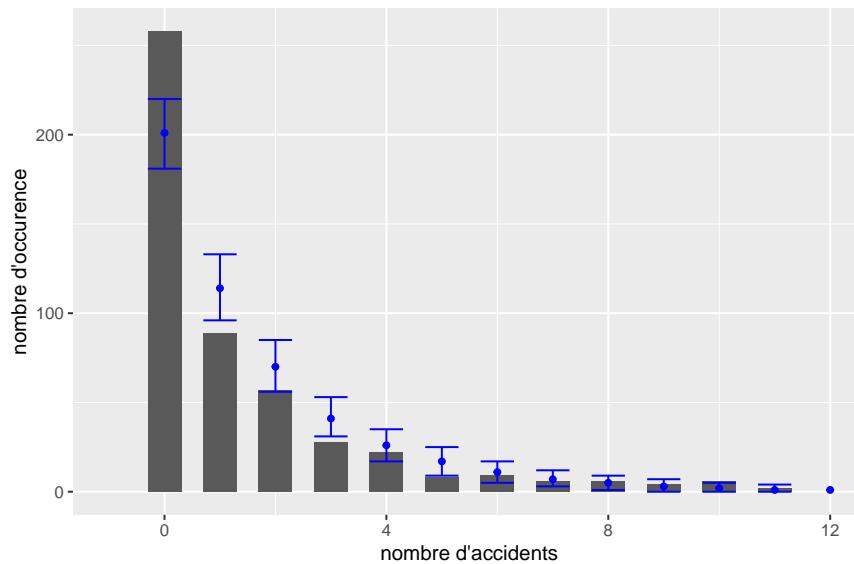


FIG. 1.38 : Comparaison de la distribution originale et des simulations pour le modèle de Poisson avec excès de zéros ajusté

Vérifier la qualité d'ajustement

```

modelenull <- glm(Nbr_acci ~ 1,
                    family = poisson(link="log"),
                    data = data2)

# calcul des R2
rsqs(loglike.full = logLik(modelza),
      loglike.null = logLik(modelenull),
      full.deviance = deviance(modelza),
      null.deviance = deviance(modelenull),
      nb.params = modelza$sigma.df + modelza$mu.df,
      n = nrow(data2)
      )

## `$deviance expliquee`
## [1] 0.1073371
##
## `$`MacFadden ajuste`
## 'log Lik.' 0.3588483 (df=18)
##
## `$`Cox and Snell`
```

```

## 'log Lik.' 0.8086509 (df=18)
##
## $Nagelkerke
## 'log Lik.' 0.8186255 (df=18)

# calcul du RMSE
sqrt(mean((preds - data2$Nbr_acci)**2))

## [1] 2.635322

```

Le modèle avec excès de zéro ajusté ne parvient à expliquer que 11% de la déviance totale. Il obtient cependant des valeurs de R² assez hautes (MacFadden ajusté : 0,36, Nagerlkerke : 0,82). Son RMSE est cependant très élevé (2,6), comparativement à celui que nous avions obtenu avec le modèle Négatif Binomial (1,9). Considérant ces éléments, ce modèle est nettement moins informatif que le modèle Négatif Binomial et ne devrait pas être retenu. Nous allons cependant montré ici comment interpréter ces résultats.

Interprétation des résultats

L'ensemble des coefficients du modèle sont accessible avec la fonction `summary`. Les coefficients dédiés à la partie Poisson (appelée mu dans le résumé) doivent être analysés et interprétés de la même manière que s'ils provenaient d'un modèle de Poisson. Les coefficients appartenant à la partie logistique (appelé sigma dans le résumé) doivent être analysés et interprété de la même manière que s'ils provenaient d'un modèle logistique.

```

# extraction des résultats
base_table <- summary(modelza)

## ****
## Family: c("ZIP", "Poisson Zero Inflated")
##
## Call: gammelss(formula = Nbr_acci ~ Feux_auto + Feux_piet +
##     Pass_piet + Terreplein + Apaisement + LogEmploi +
##     Densite_pop + Entropie + DensiteInter + Long_arterePS +
##     Artere + NB_voies5, sigma.formula = ~1 + Feux_auto +
##     Feux_piet + Densite_pop + NB_voies5, family = ZIP(mu.link = "log",
##     sigma.link = "logit"), data = data2)
##
## Fitting method: RS()
##
## -----
## Mu link function: log
## Mu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.609e+00 5.419e-01 -4.814 1.98e-06 ***
## Feux_auto    5.779e-01 1.903e-01  3.036  0.00253 **
## Feux_piet    4.207e-01 1.048e-01  4.012 6.97e-05 ***
## Pass_piet    3.995e-01 1.938e-01  2.061  0.03987 *
## Terreplein   -3.348e-01 1.666e-01 -2.010  0.04502 *
## Apaisement    2.246e-01 1.535e-01  1.463  0.14405
## LogEmploi    1.663e-01 7.509e-02  2.215  0.02723 *

```

```

## Densite_pop    8.610e-05  1.501e-05   5.735  1.72e-08 ***
## Entropie      -2.894e-01  2.950e-01  -0.981   0.32698
## DensiteInter   4.035e-03  2.052e-03   1.967   0.04980 *
## Long_arterePS  1.100e-05  2.024e-05   0.544   0.58698
## Artere        8.629e-02  1.117e-01   0.772   0.44035
## NB_voies5     4.295e-01  1.017e-01   4.224   2.87e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -----
## Sigma link function: logit
## Sigma Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.004e+00  5.530e-01   1.816  0.07005 .
## Feux_auto   -1.716e+00  6.031e-01  -2.845  0.00463 **
## Feux_piet    2.661e-01  7.228e-01   0.368  0.71292
## Densite_pop -1.170e-04  5.469e-05  -2.140  0.03286 *
## NB_voies5   -1.831e+00  1.227e+00  -1.493  0.13606
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -----
## No. of observations in the fit: 500
## Degrees of Freedom for the fit: 18
##       Residual Deg. of Freedom: 482
##                           at cycle: 11
##
## Global Deviance: 1376.967
##          AIC: 1412.967
##          SBC: 1488.829
## ****

```

```

# multiplication par 1000 des coefficient de population
# (effet pour 1000 habitants)
base_table[8,1] <- 1000 * base_table[8,1]
base_table[8,2] <- 1000 * base_table[8,2]
base_table[17,1] <- 1000 * base_table[17,1]
base_table[17,2] <- 1000 * base_table[17,2]

# multiplication par 1000 des coefficient de longueur artere
# (effet pour 1 km)
base_table[11,1] <- 1000 * base_table[11,1]
base_table[11,2] <- 1000 * base_table[11,2]

# calcul des exponentiels des predicteurs
# et des intervalles de confiance
expcoeff <- exp(base_table[,1])
expcoeff2.5 <- exp(base_table[,1] - 1.96 * base_table[,2])
expcoeff97.5 <- exp(base_table[,1] + 1.96 * base_table[,2])

base_table <- cbind(base_table, expcoeff, expcoeff2.5,expcoeff97.5)

```

```

#calculer une colonne indiquant le niveau de significativite
sign <- case_when(
  base_table[,4] < 0.001 ~ "***",
  base_table[,4] >= 0.001 & base_table[,4]<0.01 ~ "**",
  base_table[,4] >= 0.01 & base_table[,4]<0.05 ~ "*",
  base_table[,4] >= 0.05 & base_table[,4]<0.1 ~ ".",
  TRUE ~ ""
)

# arrondir : 3 decimales
base_table <- round(base_table,3)

# enlever les colonnes de valeurs de t et d'erreur standard
base_table <- base_table[,c(1,4,5,6,7)]
base_table <- cbind(base_table, sign)

# remplacer les 0 dans la colonne pval
base_table[,2] <- ifelse(base_table[,2]=="0","<0.001",base_table[,2])

# separer le tout en deux tableaux
part_poiss <- base_table[1:13,]
part_logit <- base_table[14:18,]

# mettre les bons noms de colonne
colnames(part_poiss) <- c("Coeff.", "Val.p", "Exp(Coeff.)", "IC 2,5% exp(Coeff.)", "IC 97,5% exp(Coeff.)", "Sign.")

colnames(part_logit) <- c("Coeff.", "Val.p", "RC", "IC 2,5% RC", "IC 97,5% RC", "Sign.")

```

Nous rapportons les résultats de ce modèle de Poisson avec excès de zéro ajusté dans les tableau 1.27 et 1.28.

Nous observons ainsi que la présence d'un feu de circulation divise les chances de ne pas observer d'accident à une intersection par 5. La densité de population a également un effet positif sur la probabilité d'observer des accidents à une intersection. Pour chaque 1000 habitants supplémentaire à proximité de l'intersection, les chances de ne pas observer d'accident sont réduites de 11%.

Concernant les coefficients pour la partie Poisson du modèle, nous observons que à nouveau, la présence

Tab. 1.27 : Résultats de la partie Poisson du modèle de Poisson avec excès de zéros ajusté

Variable	Coeff.	Val.p	Exp(Coeff.)	IC 2,5% exp(Coeff.)	IC 97,5% exp(Coeff.)	Sign.
(Intercept)	-2.609	<0.001	0.074	0.025	0.213	***
Feux_auto	0.578	0.003	1.782	1.227	2.588	**
Feux_piet	0.421	<0.001	1.523	1.24	1.87	***
Pass_piet	0.399	0.04	1.491	1.02	2.18	*
Terreplein	-0.335	0.045	0.715	0.516	0.992	*
Apaisement	0.225	0.144	1.252	0.927	1.691	
LogEmploi	0.166	0.027	1.181	1.019	1.368	*
Densite_pop	0.086	<0.001	1.09	1.058	1.122	***
Entropie	-0.289	0.327	0.749	0.42	1.335	
DensiteInter	0.004	0.05	1.004	1	1.008	*
Long_arterePS	0.011	0.587	1.011	0.972	1.052	
Artere	0.086	0.44	1.09	0.876	1.357	
NB_voies5	0.429	<0.001	1.536	1.259	1.875	***

TAB. 1.28 : Résultats de la partie logistique du modèle de Poisson avec excès de zéros ajusté

Variable	Coeff.	Val.p	RC	IC 2,5% RC	IC 97,5% RC	Sign.
(Intercept)	1.004	0.07	2.729	0.923	8.067	.
Feux_auto	-1.716	0.005	0.18	0.055	0.586	**
Feux_piet	0.266	0.713	1.305	0.316	5.381	
Densite_pop	-0.117	0.033	0.89	0.799	0.99	*
NB_voies5	-1.831	0.136	0.16	0.014	1.773	

d'un feu de circulation et d'un feu pour piéton contribuent à multiplier respectivement par 2 et 1,5 le nombre attendu d'accident à une intersection. De même, la présence d'un axe de circulation à 5 voies augmente de 57% le nombre d'accident. Enfin, la densité de population est aussi associée à une augmentation du nombre d'accident. Pour 1000 habitant supplémentaire autours de l'intersection, on augmente le nombre d'accident attendu de 9%.

1.2.2.5 Conclusion sur les modèles destinés à des variables de comptage

Nous avons vu dans cette section que modéliser une variable de comptage ne doit pas toujours être fait avec une simple distribution de Poisson. Il est nécessaire de tenir compte de la potentielle sur ou sous dispersion ainsi que de l'excès de 0. Nous n'avons cependant pas couvert tous les cas, il est en effet possible d'ajuster des modèles avec une distribution négative binomiale avec excès de zéros (avec le package `gamlss`), ainsi que des modèles de Hurdle. Ces derniers ont une approche différente de celle proposée par les distributions ajustées pour tenir compte de l'excès de zéro que nous détaillons dans l'encadré pour aller plus loin ci-dessous. Le processus de sélection du modèle peut être résumé avec la figure 1.39. Notez que même en suivant cette procédure, rien ne garantit que votre modèle final reflète bien les données que vous étudiez. L'analyse approfondie des résidus et des prédictions du modèle est la seule façon de déterminer si oui ou non le modèle est fiable.



: modèle de Hurdle VS modèle avec excès de zéro

Les modèles de Hurdle sont une autre catégorie de modèle GLM. Ils peuvent être décrits avec la formulation suivante :

$$\begin{cases} Y \sim \text{Binomial}(p) \text{ si } y = 0 \\ \text{logit}(p) = \beta_a X_a \\ Y \sim \text{TrPoisson}(\lambda) \text{ si } y > 0 \\ \log(\lambda) = \beta_b X_b \end{cases} \quad (1.20)$$

On constate qu'un modèle de Hurdle utilise deux distributions, la première est une distribution Binomiale dont l'objectif est de prédire si les observations sont à 0 ou au-dessus de 0. La seconde est une distribution strictement positive (supérieure à 0), il peut s'agir d'une distribution tronquée de Poisson, tronquée Négative Binomiale, Gamma, log-Normale ou autre, dépendamment du phénomène modélisé. Puisque le modèle fait appel à deux distributions, deux équations de régressions sont utilisées, l'une pour prédire p (la probabilité d'observer une valeur au-dessus de 0) et l'autre l'espérance (moyenne) de la seconde distribution.

En d'autres termes, un modèle de Hurdle modélise les données à zéro et les données au-delà de 0 comme deux processus différents (chacun avec sa propre distribution). Cette approche se distingue des modèles avec excès de zéros qui utilisent une seule distribution pour décrire l'ensemble des données. D'après un modèle avec excès de zéro, il existe des vrais et des faux zéros que l'on tente de distinguer. Dans un modèle de Hurdle, l'idée est que les zéros constituent une limite. On modélise la probabilité de dépasser cette limite, et ensuite la magnitude du dépassement de cette limite.

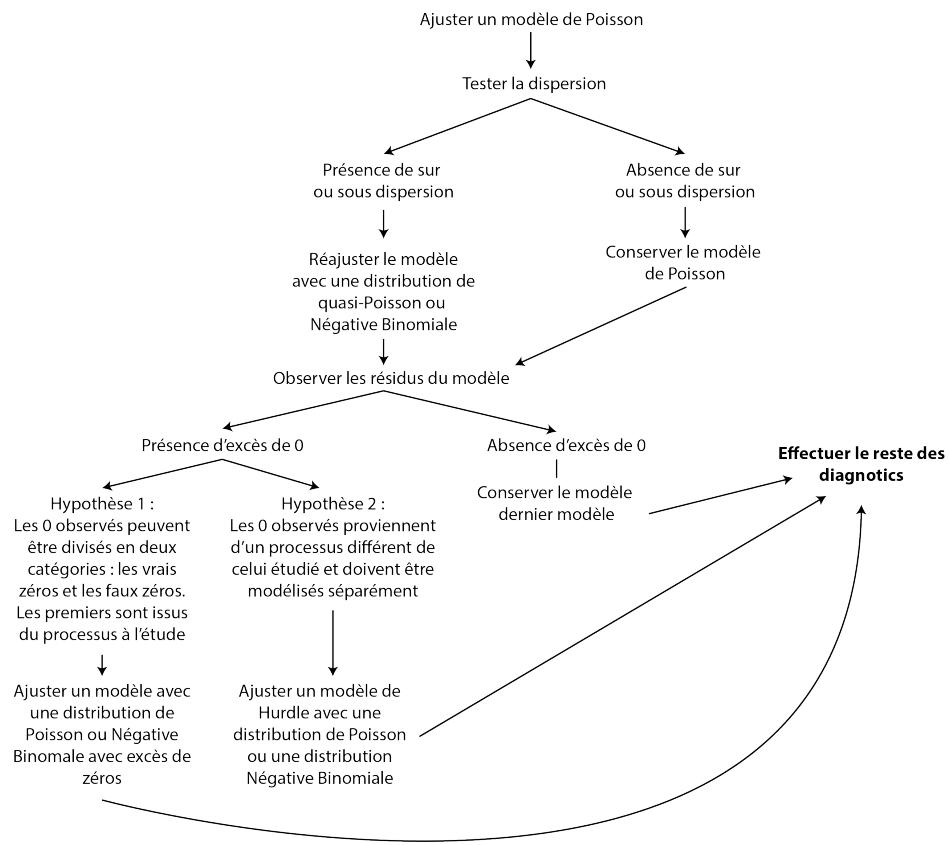


FIG. 1.39 : Processus de sélection d'un modèle pour une variable de comptage

Prenons un exemple pour rendre la distinction plus concrète. Admettons que nous utilisons un capteur capable de mesurer la concentration de particules fines dans l'air. D'après les spécifications du fabricant, le capteur est capable de mesurer des taux de concentration à partir de $0,001 \text{ ug/m}^3$. Dans une ville avec des niveaux de concentration très faibles, il sera très fréquent pour le capteur de renvoyer zéro. Considérant ce phénomène, il serait judicieux de modéliser le processus avec un modèle de Hurdle Gamma puisque les 0 représentent une limite qui n'a pas été franchie : le seuil de détection du capteur. On va donc traiter différemment les secteurs sous ce seuil, et au-dessus de seuil. Si l'on reprend notre exemple sur les accidents des piétons à des intersections, il est plus judicieux dans ce cas de modéliser le phénomène avec un modèle avec excès de zéro puisque l'on peut observer zéro accident sur une intersection dangereuse (vrai zéro), et zéro accident sur une intersection sur laquelle aucune piéton ne traverse jamais (faux zéro).

1.2.3 Les modèles GLM pour des variables continues

Comme nous avons pu le voir dans la section ??, il existe un grand nombre de distributions permettant de décrire une grande diversité de variables continues. Il serait fastidieux de tous les présenter, nous allons seulement revenir sur les plus fréquents.

1.2.3.1 Le modèle GLM Gaussien

Comme nous l'avons vu en introduction, il s'agit du GLM le plus simple puisqu'il correspond à la transposition de la régression linéaire classique (des moindres carrés) dans la forme des modèles

généralisés.

1.2.3.1.1 Conditions d'applications

Les conditions d'applications sont les mêmes que celles pour une régression linéaire classique. La condition de l'Homoscédasticité (homogénéité de la variance), est dûe au fait que la variance du modèle est contrôlée par un seul paramètre fixe $\text{var}(y) = \sigma$ (l'écart type de la distribution normale). À titre de comparaison, rappelons que dans un modèle de Poisson, la variance est égale à la moyenne ($\text{var}(y) = E(y)$) alors que dans un modèle Négatif Binomial, la variance est fonction de la moyenne et d'un paramètre θ ($\text{var}(y) = E(y) + E(y)^{\frac{2}{\theta}}$). Pour ces deux exemples, la variance augmente au fur et à mesure que la moyenne augmente.

1.2.3.1.2 Interprétation des paramètres

L'interprétation des paramètres est la même que pour une régression linéaire classique :

- β_0 : il s'agit de l'intercept, donc de la moyenne attendue de la variable Y lorsque les valeurs de toutes les variables X sont 0.
- β : les coefficients de régressions, ils quantifient l'impact d'une augmentation d'une unité des variables X sur la moyenne de la variable Y .
- σ : l'écart type de Y après avoir contrôlé les variables X . Il peut s'interpréter comme l'incertitude restante après modélisation de la moyenne de Y . Concrètement, si vous utilisez votre équation de régression pour prédire une nouvelle valeur de Y : \hat{Y} , l'intervalle de confiance à 95% de cette prédiction est $(\hat{Y} - 3\sigma; \hat{Y} + 3\sigma)$. Vous noterez donc que plus σ est grand, plus grande est l'incertitude de la prédiction.

1.2.3.1.3 Exemple appliqué dans R

Pour cet exemple, nous reprenons le modèle LM que nous avions présenté dans la section ???. Pour rappel, l'objectif est de modéliser la densité végétale dans les secteurs de recensement de Montréal. Pour cela, nous utilisons des variables caractérisant des populations vulnérables au niveau physiologique ou socio-économique, tout en contrôlant l'impact de l'environnement urbain. Parmi ces dernières, l'âge médian des bâtiments est ajouté au modèle avec une polynomiale d'ordre deux, et la densité d'habitant transformé avec la fonction logarithme.

Vérification des conditions d'application

La première étape de la vérification des conditions d'application est bien sûr la vérification de l'absence de multicolinéarité.

TAB. 1.29 : Carte d'identité du modèle Gaussien

Type de variable dépendante	Variable continue dans l'intervalle $]-\infty; +\infty[$
Distribution utilisée	Normal
Formulation	$Y \sim Normal(\mu, \sigma)$ $g(\mu) = \beta_0 + \beta X$ $g(x) = x$
Fonction de lien	identitaire
Paramètre modélisé	μ
Paramètres à estimer	β_0, β et σ
Conditions d'applications	Homoscédasticité

```
# chargement des données
load("data/lm/DataVegetation.RData")

# calcul du VIF
vif(glm(VegPct ~ log(HABHA)+poly(AgeMedian,2) +
        Pct_014+Pct_65P+Pct_MV+Pct_FR, data = DataFinal))
```

	GVIF	Df	GVIF ^{(1/(2*Df))}
## log(HABHA)	1.289495	1	1.135559
## poly(AgeMedian, 2)	1.387429	2	1.085307
## Pct_014	1.517957	1	1.232054
## Pct_65P	1.304094	1	1.141969
## Pct_MV	1.480275	1	1.216666
## Pct_FR	1.729646	1	1.315160

Puisque l'ensemble des valeurs de VIF sont inférieures à deux, nos données ne sont pas marquées par une multicolinéarité problématique. La seconde étape du diagnostic est de calculer et d'afficher les distances de Cook.

```
# ajustement du modèle
modele <- glm(VegPct ~ log(HABHA)+poly(AgeMedian,2) +
                 Pct_014+Pct_65P+Pct_MV+Pct_FR, data = DataFinal)

# calcul des distances de Cook
cooksd <- cooks.distance(modele)

# affichage des valeurs
df <- data.frame(
  cook = cooksd,
  oid = 1:length(cooksd)
)

ggplot(data = df) +
  geom_point(aes(x = oid, y = cook), color = rgb(0.4,0.4,0.4,0.7), size = 1) +
  labs(x="", y = "distance de Cook")
```

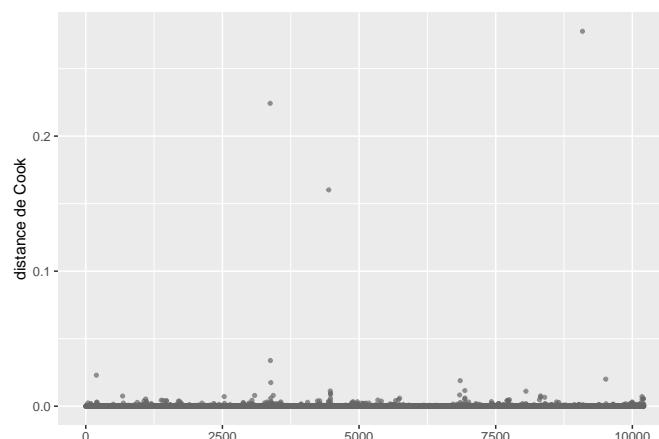


FIG. 1.40 : Distances de Cook pour le modèle Gaussien

La figure 1.40 indique clairement que trois observations sont très influentes dans le modèle.

```
# selection des cas étranges
cas_étranges <- subset(DataFinal, cooksd > 0.03)
print(cas_étranges)

##          IDIDU VegPct ArbPct V250Pct V500Pct A250Pct A500Pct      HABHA
## 3374 2466125715 10.481   5.478  18.987  13.744   2.908   2.704 74.835867
## 3378 2466125725  0.000   0.000  12.709  12.505   2.116   2.324 88.006946
## 4446 2466169704 23.162   5.209  31.437  31.535   8.672   9.108 313.142733
## 9088 2466307709 85.767  27.583  78.195  83.492  42.999  51.074  2.070472
##          AgeMedian Pct_014 Pct_65P Pct_MV Pct_FR DistCBDkm SDRIDU SDRNOM
## 3374        226     4.76    14.29   23.81   14.29      0.748 2466023 Montréal
## 3378        206     6.25    12.50   25.00   12.50      0.706 2466023 Montréal
## 4446        206    14.40    16.87   53.50   42.39      8.678 2466023 Montréal
## 9088        207    12.00    24.00   12.00   16.00     28.440 2466023 Montréal
```

Il s'agit de quatre observations dans Montréal avec des logements très anciens : plus de 200 ans alors que la moyenne est de 52 ans pour le reste de la zone d'étude. Le fait que nous ayons dans le modèle une polynomiale d'ordre 2 pour cette variable intensifie l'impact de ces valeurs extrêmes. En conséquence nous décidons de simplement les supprimer. Nous verrons plus tard qu'une alternative envisageable est de changer la distribution du modèle pour une distribution de Student (plus robuste aux valeurs extrêmes).

```
# retirer les cas étranges
DataFinal2 <- subset(DataFinal, cooksd < 0.03)

# ajustement du modèle
modele <- glm(VegPct ~ log(HABHA)+poly(AgeMedian,2)+Pct_014+Pct_65P+Pct_MV+Pct_FR, data = DataFinal2,
family = gaussian())

# calcul des distances de Cook
cooksd <- cooks.distance(modele)

# affichage des valeurs
df <- data.frame(
  cook = cooksd,
  oid = 1:length(cooksd)
)

ggplot(data = df) +
  geom_point(aes(x = oid, y = cook), color = rgb(0.4,0.4,0.4,0.7), size = 1) +
  labs(x="", y = "distance de Cook")
```

Une fois ces observations retirée, les nouvelles distances de Cook (figure 1.41) ne révèlent plus d'observations fortement influentes. Nous pouvons passer à l'analyse des résidus similés. La figure 1.42 nous montre cependant que nos résidus devient significativement d'une distribution uniforme, que des valeurs aberrantes sont encore présente et qu'il existe un lien entre résidus et prédiction dans le modèle.

```
# extraction des prédictions du modèle
mus <- predict(modele, type = 'response')
modsigma <- sigma(modele)
```

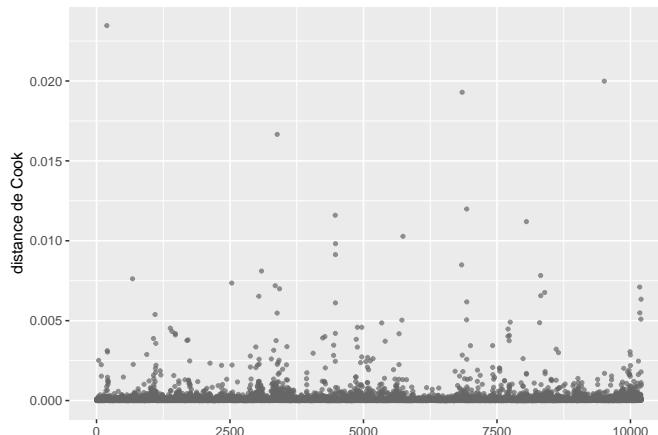


FIG. 1.41 : Distances de Cook pour le modèle Gaussien après suppression des observations influentes

```
# extraction de l'écart type du modèle

# génération de 1000 simulation pour chaque prediction
nsim <- 1000
cols <- lapply(1:length(mus), function(i){
  mu <- mus[[i]]
  sims <- rnorm(nsim, mean=mu, sd = modsigma)
  return(sims)
})
mat_sims <- do.call(rbind, cols)

# calculer les résidus simulés
sim_res <- createDHARMA(simulatedResponse = mat_sims,
                           observedResponse = DataFinal2$VegPct,
                           fittedPredictedResponse = mus,
                           integerResponse = F)
plot(sim_res)
```

Pour mieux cerner le problème que nous avons ici, nous pouvons dans un premier temps comparer la distribution originale des données et les simulations issues du modèle. La figure 1.43 montre clairement le problème : la distribution normale est mal ajustée aux données. Ces dernières sont légèrement asymétriques et ne peuvent pas être inférieures à zéro, ce que la distribution normale ne parvient pas à reproduire.

```
df <- reshape2::melt(mat_sims[,1:30])

ggplot() +
  geom_histogram(data = DataFinal2, mapping = aes(x = VegPct, y = ..density..), color = "black", fill = "white", binwidth = 0.05) +
  geom_density(data = df, aes(x = value, group = Var2), color = rgb(0.4,0.4,0.4,0.4), fill = rgb(0,0,0,0))
```

Il est également possible de vérifier si la condition d'homogénéité de la variance s'applique bien aux données :

```
# extraction des prédictions du modèle
mus <- predict(modele, type = "response")
```

DHARMA residual diagnostics

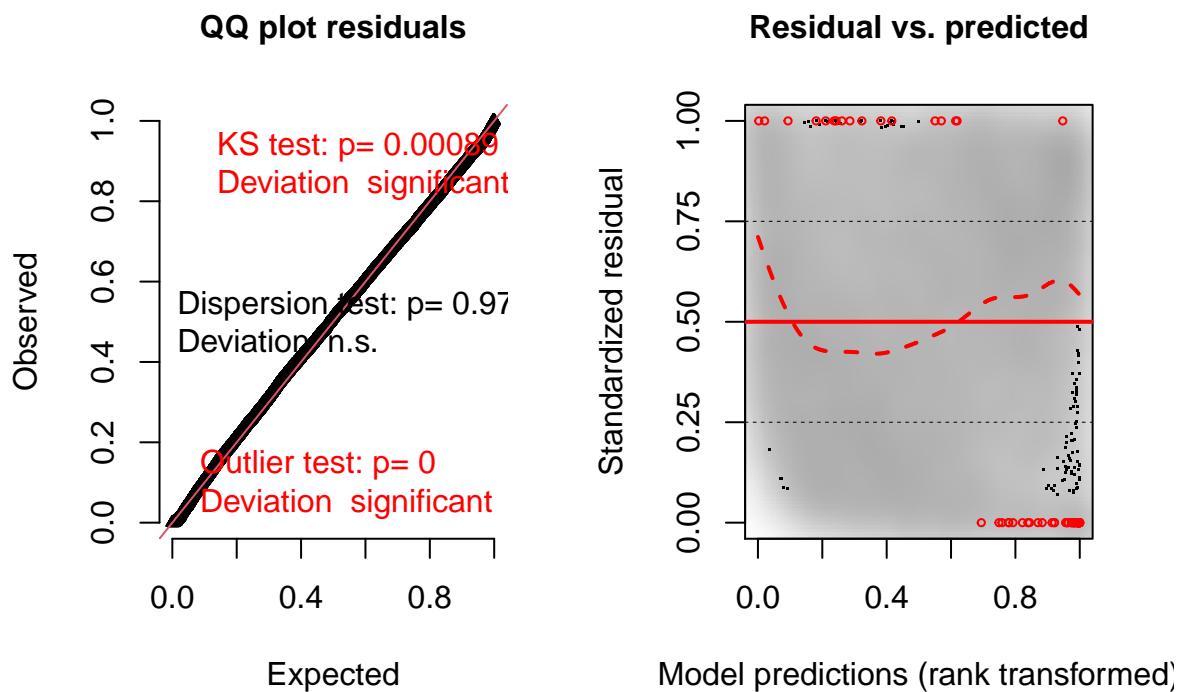


FIG. 1.42 : Diagnostic général des résidus simulés pour le modèle Gaussien

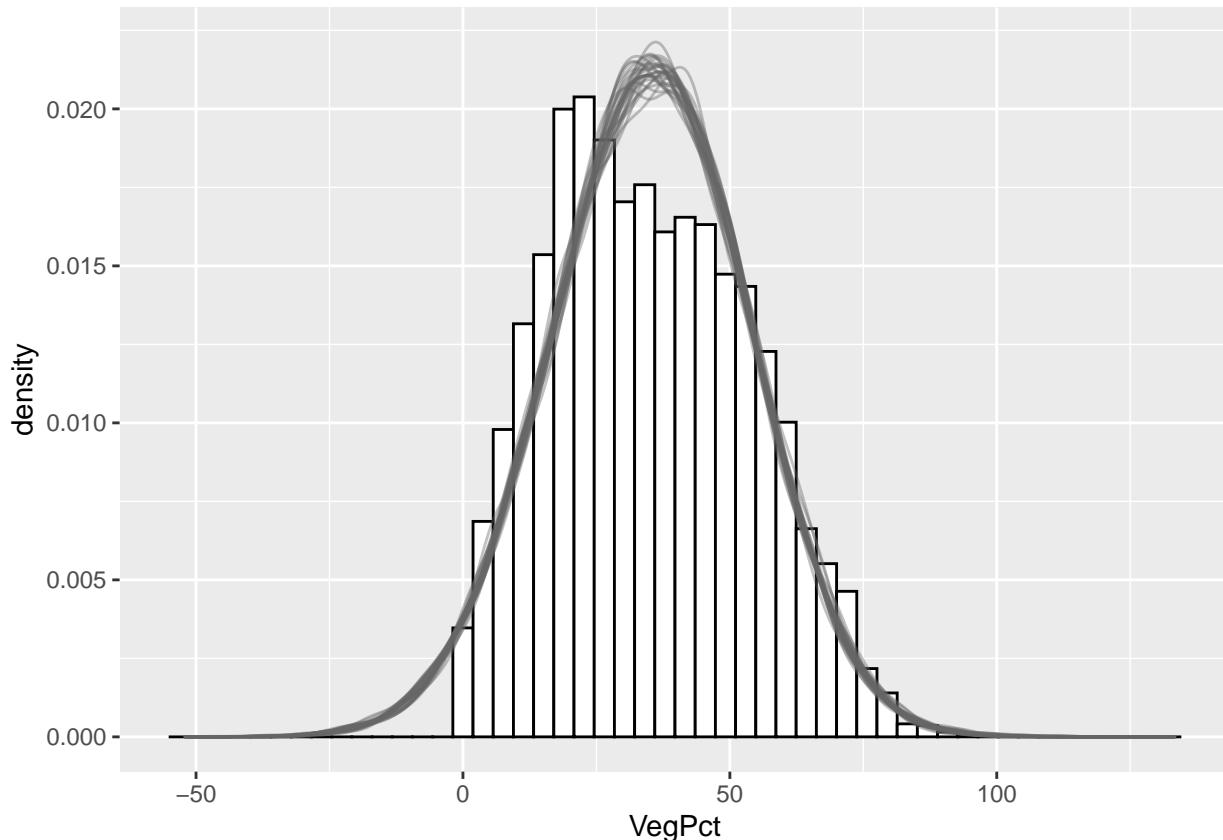


FIG. 1.43 : Comparaison de la distribution originale de la variable et des simulations issues du modèle

```

sigma_model <- sigma(modele)

# création d'un dataframe pour contenir la prédiction et les vraie valeurs
df1 <- data.frame(
  mus = mus,
  reals = DataFinal2$VegPct
)

# calcul de l'intervalle de confiance à 95% selon la distribution de Poisson
# et stockage dans un second dataframe
seqa <- seq(0,100,10)
df2 <- data.frame(
  mus = seqa,
  lower = qnorm(p = 0.025, mean = seqa, sd = sigma_model),
  upper = qnorm(p = 0.975, mean = seqa, sd = sigma_model)
)

# affichage des valeurs réelles et prédites (en rouge)
# et de leur variance selon le modèle (en noir)
ggplot() +
  geom_point(data = df1,
    mapping = aes(x = mus, y = reals),
    color = "red", size = 0.5) +
  geom_errorbar(data = df2,
    mapping = aes(x = mus, ymin = lower, ymax = upper),
    width = 0.2, color = rgb(0.4,0.4,0.4)) +
  labs(x = "valeurs prédites",
    y = "valeurs réelles")

```

À nouveau, on peut constater que le modèle s'attendrait à trouver des valeurs négative pour la concentration de végétation, ce que n'est pas possible dans notre cas. En revanche, il semblerait que la variance soit bien homogène puisque la dispersion des observations semble à peu près suivre la dispersion attendue par le modèle (en noir).

Malgré ces différents constats indiquant clairement qu'un modèle Gaussien est un choix sous-optimal pour ces données, nous allons poursuivre l'analyse de ce modèle.

Vérification de la qualité d'ajustement

```

# ajustement d'un modèle nul
modelenull <- glm(VegPct ~ 1,
  data = DataFinal2,
  family = gaussian())

# calcul des pseudo R2
rsqs(loglike.full = logLik(modele),
  loglike.null = logLik(modelenull),
  full.deviance = deviance(modele),
  null.deviance = deviance(modelenull),
  nb.params = modele$rank,
  n = nrow(DataFinal2)
)

## `$`deviance expliquee`
```

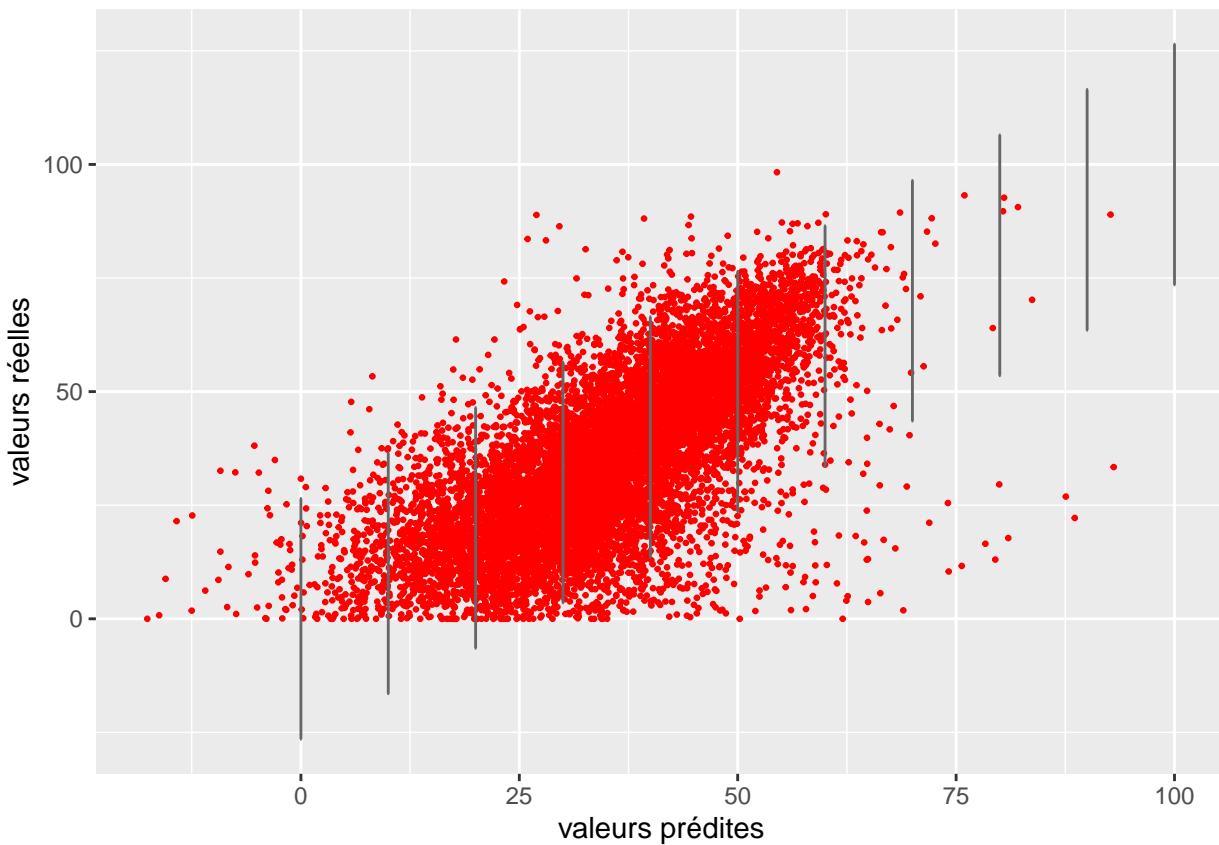


FIG. 1.44 : Comparaison de la distribution originale de la variable et des simulations issues du modèle

```
## [1] 0.4706321
##
## $`MacFadden ajuste`
## 'log Lik.' 0.07310662 (df=9)
##
## $`Cox and Snell`
## 'log Lik.' 0.4706321 (df=9)
##
## $Nagelkerke
## 'log Lik.' 0.4707122 (df=9)
```

Le modèle parvient à expliquer 47% de la déviance totale, mais obtient un R^2 ajusté de MacFadden de seulement 0,07.

```
# calcul du RMSE
sqrt(mean((predict(modele, type = "response") - DataFinal2$VegPct)**2))
```

```
## [1] 13.49885
```

L'erreur quadratique moyenne est de 13,5 points de pourcentage, ce qui indique que le modèle a une assez faible capacité prédictive.

Interprétation des résultats

L'ensemble des coefficients du modèle sont accessibles via la fonction `summary`, nous proposons le tableau

1.30 qui présente l'ensemble des résultats du modèle.

TAB. 1.30 : Résultats du modèle Gaussien

Variable	Coeff.	Err.std	Val.z	val.p	IC coeff 2,5%	IC coeff 97,5%	Sign.
Intercept	53.606	1.00	53.64	<0.001	51.647	55.565	***
AgeMedian ordre 1	2.732	15.56	0.18	0.861	-27.772	33.237	
AgeMedian ordre 2	-320.869	14	-22.91	<0.001	-348.318	-293.420	***
Pct_014	0.915	0.03	29.31	<0.001	0.853	0.976	***
Pct_65P	0.280	0.02	15.05	<0.001	0.243	0.316	***
Pct_MV	-0.042	0.01	-4.19	<0.001	-0.061	-0.022	***
Pct_FR	-0.340	0.01	-30.94	<0.001	-0.362	-0.318	***

Chapitre 2

Régressions à effets mixtes (GLMM) et régression multiniveaux

2.1 Principes de base des GLMM

Notion de pseudo-réplications

Effets aléatoire *versus* effets fixes

2.2 Application à une variable continue

2.3 Régressions multiniveaux

Chapitre 3

Modèles généralisés additifs

3.1 Principes de base des GAM

3.2 Extensions des GAM

3.2.1 GAMM

3.2.2 GAMMAR

Première partie

Analyses exploratoires multivariées

Chapitre 4

Méthodes factorielles

4.1 Un petit historique

4.2 Analyses de composantes principales (ACP)

4.3 Analyses factorielles de correspondances (AFC)

4.4 Analyses factorielles de correspondances multiples (AFM)

4.5 Analyses factorielles de correspondances mixte

Chapitre 5

Méthodes de classification non-supervisées

5.1 Un aperçu sur la multitude des méthodes de classifications

5.2 Combinaisons méthodes factorielles et méthodes de classifications

5.3 Classification ascendantes hiérachiques

5.4 Nuées dynamiques

5.4.1 k-means

5.4.2 k-median

5.4.3 Les extensions en logique floues : c-means, c-median

Chapitre 6

Conclusion

6.1 Synthèse des méthodes abordées

Road map method

6.2 Autres méthodes non abordées

6.2.1 Régressions par quantile

6.2.2 Régressions par panel

6.2.3 Régressions par Tobit

6.2.4 Analyses de survie

6.2.5 Équations structurelles

6.3 Deux écoles de statistique inférielle : fréquentiste et bayésienne

6.4 Éthique en méthodes quantitatives

Chapitre 7

Annexes

7.1 Tableau des valeurs critiques de χ^2

La courte syntaxe  ci-dessous permet de générer un tableau avec les valeurs critiques du χ^2 pour différents degrés de signification (valeurs de p).

```
library(stargazer)

# vecteur pour les degrés de liberté de 1 à 30, puis 40 et 50
dl <- c(1:30, 40, 50, 100, 250, 500)
# la fonction qchisq permet d'obtenir la valeur théorique en fonction
# d'une valeur de p et d'un nombre de degrés de liberté
tableChi2 <- cbind(dl,
                     p0.05 = round(qchisq(p=0.95, df=dl, lower.tail = FALSE),3),
                     p0.01 = round(qchisq(p=0.99, df=dl, lower.tail = FALSE),3),
                     p0.001 = round(qchisq(p=0.999, df=dl, lower.tail = FALSE),3))
# Impression du tableau avec la library stargazer
stargazer(tableChi2, type="text", summary=FALSE, rownames=FALSE, align = TRUE, digits = 3,
          title="Distribution des valeurs critiques du Khi2")
```

7.2 Tableau des valeurs critiques de F

La courte syntaxe  ci-dessous permet de générer un tableau avec les valeurs critiques de F avec $p=0,05$.

```
library(stargazer)

dl1 <- c(1:10, 15, 20, 25, 30, 50, 100, 500)
dl2 <- c(1:30, 40, 50, 100)
matrice <- matrix(ncol=length(dl1), nrow=length(dl2), byrow = TRUE)
for(r in 1:length(dl1)){
  for(c in 1:length(dl2)){
    matrice[c,r] <- round(qf(p=0.05, dl1[r], dl2[c], lower.tail = FALSE),2)
  }
}
tableF_p0.05 <- as.data.frame(matrice, row.names = paste0("dl2=",dl2, sep=""))
```

TAB. 7.1 : Distribution des valeurs critiques du khi2

dl	p=0,05	p=0,01	p=0,001
1	0,004	0,000	0,000
2	0,103	0,020	0,002
3	0,352	0,115	0,024
4	0,711	0,297	0,091
5	1,145	0,554	0,210
6	1,635	0,872	0,381
7	2,167	1,239	0,598
8	2,733	1,646	0,857
9	3,325	2,088	1,152
10	3,940	2,558	1,479
11	4,575	3,053	1,834
12	5,226	3,571	2,214
13	5,892	4,107	2,617
14	6,571	4,660	3,041
15	7,261	5,229	3,483
16	7,962	5,812	3,942
17	8,672	6,408	4,416
18	9,390	7,015	4,905
19	10,117	7,633	5,407
20	10,851	8,260	5,921
21	11,591	8,897	6,447
22	12,338	9,542	6,983
23	13,091	10,196	7,529
24	13,848	10,856	8,085
25	14,611	11,524	8,649
26	15,379	12,198	9,222
27	16,151	12,879	9,803
28	16,928	13,565	10,391
29	17,708	14,256	10,986
30	18,493	14,953	11,588
40	26,509	22,164	17,916
50	34,764	29,707	24,674
100	77,929	70,065	61,918
250	214,392	200,939	186,554
500	449,147	429,388	407,947

```
colnames(tableF_p0.05) <- paste0("dl",dl, sep="")

stargazer(tableF_p0.05, type="text", summary=FALSE, rownames=FALSE, align = TRUE, digits = 3,
           title="Distribution des valeurs critiques de F avec p=0,05")
```

7.3 Tableau des valeurs critiques de t

La courte syntaxe  ci-dessous permet de générer un tableau avec les valeurs critiques de T avec $p=0,05$, 0,01 et 0,01.

```
library(stargazer)

# vecteur pour les degrés de liberté de 1 à 30, puis 40 et 50
dl <- c(1:30, 40, 50, 60, 70, 80, 90, 100, 250, 500, 1000, 2500)
# la fonction qchisq permet d'obtenir la valeur théorique en fonction
```

TAB. 7.2 : Distribution des valeurs critiques de F avec p=0,05

dl1.1	dl1.2	dl1.3	dl1.4	dl1.5	dl1.6	dl1.7	dl1.8	dl1.9	dl1.10	dl1.15	dl1.20	dl1.25	dl1.30	dl1.50	dl1.100	d
161,45	199,50	215,71	224,58	230,16	233,99	236,77	238,88	240,54	241,88	245,95	248,01	249,26	250,10	251,77	253,04	2
18,51	19,00	19,16	19,25	19,30	19,33	19,35	19,37	19,38	19,40	19,43	19,45	19,46	19,46	19,48	19,49	
10,13	9,55	9,28	9,12	9,01	8,94	8,89	8,85	8,81	8,79	8,70	8,66	8,63	8,62	8,58	8,55	
7,71	6,94	6,59	6,39	6,26	6,16	6,09	6,04	6,00	5,96	5,86	5,80	5,77	5,75	5,70	5,66	
6,61	5,79	5,41	5,19	5,05	4,95	4,88	4,82	4,77	4,74	4,62	4,56	4,52	4,50	4,44	4,41	
5,99	5,14	4,76	4,53	4,39	4,28	4,21	4,15	4,10	4,06	3,94	3,87	3,83	3,81	3,75	3,71	
5,59	4,74	4,35	4,12	3,97	3,87	3,79	3,73	3,68	3,64	3,51	3,44	3,40	3,38	3,32	3,27	
5,32	4,46	4,07	3,84	3,69	3,58	3,50	3,44	3,39	3,35	3,22	3,15	3,11	3,08	3,02	2,97	
5,12	4,26	3,86	3,63	3,48	3,37	3,29	3,23	3,18	3,14	3,01	2,94	2,89	2,86	2,80	2,76	
4,96	4,10	3,71	3,48	3,33	3,22	3,14	3,07	3,02	2,98	2,85	2,77	2,73	2,70	2,64	2,59	
4,84	3,98	3,59	3,36	3,20	3,09	3,01	2,95	2,90	2,85	2,72	2,65	2,60	2,57	2,51	2,46	
4,75	3,89	3,49	3,26	3,11	3,00	2,91	2,85	2,80	2,75	2,62	2,54	2,50	2,47	2,40	2,35	
4,67	3,81	3,41	3,18	3,03	2,92	2,83	2,77	2,71	2,67	2,53	2,46	2,41	2,38	2,31	2,26	
4,60	3,74	3,34	3,11	2,96	2,85	2,76	2,70	2,65	2,60	2,46	2,39	2,34	2,31	2,24	2,19	
4,54	3,68	3,29	3,06	2,90	2,79	2,71	2,64	2,59	2,54	2,40	2,33	2,28	2,25	2,18	2,12	
4,49	3,63	3,24	3,01	2,85	2,74	2,66	2,59	2,54	2,49	2,35	2,28	2,23	2,19	2,12	2,07	
4,45	3,59	3,20	2,96	2,81	2,70	2,61	2,55	2,49	2,45	2,31	2,23	2,18	2,15	2,08	2,02	
4,41	3,55	3,16	2,93	2,77	2,66	2,58	2,51	2,46	2,41	2,27	2,19	2,14	2,11	2,04	1,98	
4,38	3,52	3,13	2,90	2,74	2,63	2,54	2,48	2,42	2,38	2,23	2,16	2,11	2,07	2,00	1,94	
4,35	3,49	3,10	2,87	2,71	2,60	2,51	2,45	2,39	2,35	2,20	2,12	2,07	2,04	1,97	1,91	
4,32	3,47	3,07	2,84	2,68	2,57	2,49	2,42	2,37	2,32	2,18	2,10	2,05	2,01	1,94	1,88	
4,30	3,44	3,05	2,82	2,66	2,55	2,46	2,40	2,34	2,30	2,15	2,07	2,02	1,98	1,91	1,85	
4,28	3,42	3,03	2,80	2,64	2,53	2,44	2,37	2,32	2,27	2,13	2,05	2,00	1,96	1,88	1,82	
4,26	3,40	3,01	2,78	2,62	2,51	2,42	2,36	2,30	2,25	2,11	2,03	1,97	1,94	1,86	1,80	
4,24	3,39	2,99	2,76	2,60	2,49	2,40	2,34	2,28	2,24	2,09	2,01	1,96	1,92	1,84	1,78	
4,23	3,37	2,98	2,74	2,59	2,47	2,39	2,32	2,27	2,22	2,07	1,99	1,94	1,90	1,82	1,76	
4,21	3,35	2,96	2,73	2,57	2,46	2,37	2,31	2,25	2,20	2,06	1,97	1,92	1,88	1,81	1,74	
4,20	3,34	2,95	2,71	2,56	2,45	2,36	2,29	2,24	2,19	2,04	1,96	1,91	1,87	1,79	1,73	
4,18	3,33	2,93	2,70	2,55	2,43	2,35	2,28	2,22	2,18	2,03	1,94	1,89	1,85	1,77	1,71	
4,17	3,32	2,92	2,69	2,53	2,42	2,33	2,27	2,21	2,16	2,01	1,93	1,88	1,84	1,76	1,70	
4,08	3,23	2,84	2,61	2,45	2,34	2,25	2,18	2,12	2,08	1,92	1,84	1,78	1,74	1,66	1,59	
4,03	3,18	2,79	2,56	2,40	2,29	2,20	2,13	2,07	2,03	1,87	1,78	1,73	1,69	1,60	1,52	
3,94	3,09	2,70	2,46	2,31	2,19	2,10	2,03	1,97	1,93	1,77	1,68	1,62	1,57	1,48	1,39	

```
# d'une valeur de p et d'un nombre de degrés de liberté
tableT <- cbind(dl,
  p0.10 = round(qt(p=1 - (0.10/2), df=dl),2),
  p0.05 = round(qt(p=1 - (0.05/2), df=dl),2),
  p0.01 = round(qt(p=1 - (0.01/2), df=dl),2),
  p0.001 = round(qt(p=1 - (0.001/2), df=dl),2))

# Impression du tableau avec la library stargazer
stargazer(tableT, type="text", summary=FALSE, rownames=FALSE, align = TRUE, digits = 2,
  title="Distribution des valeurs critiques de t")
```

TAB. 7.3 : Distribution des valeurs critiques de t

dl	p0.10	p0.05	p0.01	p0.001
1	6,31	12,71	63,66	636,62
2	2,92	4,30	9,92	31,60
3	2,35	3,18	5,84	12,92
4	2,13	2,78	4,60	8,61
5	2,02	2,57	4,03	6,87
6	1,94	2,45	3,71	5,96
7	1,89	2,36	3,50	5,41
8	1,86	2,31	3,36	5,04
9	1,83	2,26	3,25	4,78
10	1,81	2,23	3,17	4,59
11	1,80	2,20	3,11	4,44
12	1,78	2,18	3,05	4,32
13	1,77	2,16	3,01	4,22
14	1,76	2,14	2,98	4,14
15	1,75	2,13	2,95	4,07
16	1,75	2,12	2,92	4,01
17	1,74	2,11	2,90	3,97
18	1,73	2,10	2,88	3,92
19	1,73	2,09	2,86	3,88
20	1,72	2,09	2,85	3,85
21	1,72	2,08	2,83	3,82
22	1,72	2,07	2,82	3,79
23	1,71	2,07	2,81	3,77
24	1,71	2,06	2,80	3,75
25	1,71	2,06	2,79	3,73
26	1,71	2,06	2,78	3,71
27	1,70	2,05	2,77	3,69
28	1,70	2,05	2,76	3,67
29	1,70	2,05	2,76	3,66
30	1,70	2,04	2,75	3,65
40	1,68	2,02	2,70	3,55
50	1,68	2,01	2,68	3,50
60	1,67	2,00	2,66	3,46
70	1,67	1,99	2,65	3,44
80	1,66	1,99	2,64	3,42
90	1,66	1,99	2,63	3,40
100	1,66	1,98	2,63	3,39
250	1,65	1,97	2,60	3,33
500	1,65	1,96	2,59	3,31
1 000	1,65	1,96	2,58	3,30
2 500	1,65	1,96	2,58	3,29

Bibliographie

- Brant, R. (1990). Assessing proportionality in the proportional odds model for ordinal logistic regression. *Biometrics*, pages 1171–1178.
- Cloutier, M.-s., Tremblay, M., Morency, P., and Apparicio, P. (2014). Carrefours en milieu urbain : quels risques pour les piétons ? exemple empirique des quartiers centraux de montréal, canada.
- Dunn, P. K. and Smyth, G. K. (1996). Randomized quantile residuals. *Journal of Computational and Graphical Statistics*, 5(3) :236–244.
- Gelman, A. and Hill, J. (2006). *Data analysis using regression and multilevel/hierarchical models*. Cambridge university press.
- Hilbe, J. M. (2009). *Logistic regression models*. CRC press.
- McFadden, B. R. (2016). Examining the gap between science and public opinion about genetically modified food and global warming. *PloS one*, 11(11) :e0166140.
- Nelder, J. A. and Wedderburn, R. W. M. (1972). Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3) :370–384.
- Neyman, J., Pearson, E. S., and Pearson, K. (1933). Ix. on the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706) :289–337.
- SAS Institute Inc (2020a). SAS/STAT 15.2 User's Guide modeling multinomial overdispersion. https://documentation.sas.com/?docsetId=statug&docsetTarget=statug_fmm_examples04.htm&docsetVersion=15.2&locale=en. Accessed : 2020-10-15.
- SAS Institute Inc (2020b). SAS/STAT 15.2 User's Guide poisson regression | sas annotated output. <https://stats.idre.ucla.edu/sas/output/poisson-regression/>. Accessed : 2020-10-15.
- Teubner, T., Hawlitschek, F., and Dann, D. (2017). Price determinants on airbnb : How reputation pays off in the sharing economy. *Journal of Self-Governance & Management Economics*, 5(4).
- Wang, D. and Nicolau, J. L. (2017). Price determinants of sharing economy based accommodation rental : A study of listings from 33 cities on airbnb. com. *International Journal of Hospitality Management*, 62 :120–131.
- Zeileis, A. (2004). Econometric computing with hc and hac covariance matrix estimators.
- Zhang, Z., Chen, R. J., Han, L. D., and Yang, L. (2017). Key factors affecting the price of airbnb listings : A geographically weighted approach. *Sustainability*, 9(9) :1635.