Étude de cas - GLM et valeurs manquantes

Romane Lacoste-Badie, Margaux Touzé et Camille Loisel

02/03/2022

Contents

1 Modèles linéaires généralisés (GLM) avec R

1.1 Introduction

L'objectif de cette première partie est d'étudier le jeu de données sur les mites Oribatid mis à disposition par P. Legendre et D. Borcard et décrit dans leur article (Borcard and Legendre 1994). Tout particulièrement, nous essaierons de construire différents modèles linéaires simples puis généralisés pour décrire trois des variables de ce jeu de données.

Pour leur étude, Legendre et Borcard ont choisi une zone située à Saint-Hippolyte, Canada, qu'ils ont divisée en 70 "coeurs". Pour chaque coeur, ils ont récupéré des données environnementales et sur la faune. Le jeu de données que l'on étudie est donc constitué de 70 observations qui correspondent aux coeurs, et de 9 variables décrites ci-dessous.

```
str(mites)
```

```
## 'data.frame':    70 obs. of  9 variables:
##  $ Galumna   : int  8 3 1 1 2 1 1 1 2 5 ...
##  $ pa        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ totalabund: int  140 268 186 286 199 209 162 126 123 166 ...
##  $ prop      : num  0.05714 0.01119 0.00538 0.0035 0.01005 ...
##  $ SubsDens  : num  39.2 55 46.1 48.2 23.6 ...
##  $ WatrCont  : num  350 435 372 360 204 ...
##  $ Substrate : chr  "Sphagn1" "Litter" "Interface" "Sphagn1" ...
##  $ Shrub     : chr  "Few" "Few" "Few" "Few" ...
##  $ Topo      : chr  "Hummock" "Hummock" "Hummock" "Hummock" ...
```

- **Galumna** (entier) : nombre de mites *Galumna sp.*
- **pa** (0 ou 1) : présence (1) ou absence (0) de mites *Galumna sp.*
- **totalabund** (entier) : nombre total de mites
- **prop** (réel entre 0 et 1) : proportion de mites *Galumna sp.*
- **SubsDens** (réel) : densité du substrat en $g.L^{-1}$ de matière sèche non comprimée
- **WatrCont** (réel) : contenu en eau du substrat en $g.L^{-1}$
- **Substrate** (facteur) : type de substrat, facteur à 7 modalités
- **Shrub** (facteur) : buissons, facteur à trois modalités
- **Topo** (facteur) : microtopographie, facteur à 2 modalités

Nous allons maintenant représenter ces données afin de mieux les appréhender.

Commençons par les variables que l'on va chercher à expliquer par la suite : `Galumna`, `pa` et `prop.`

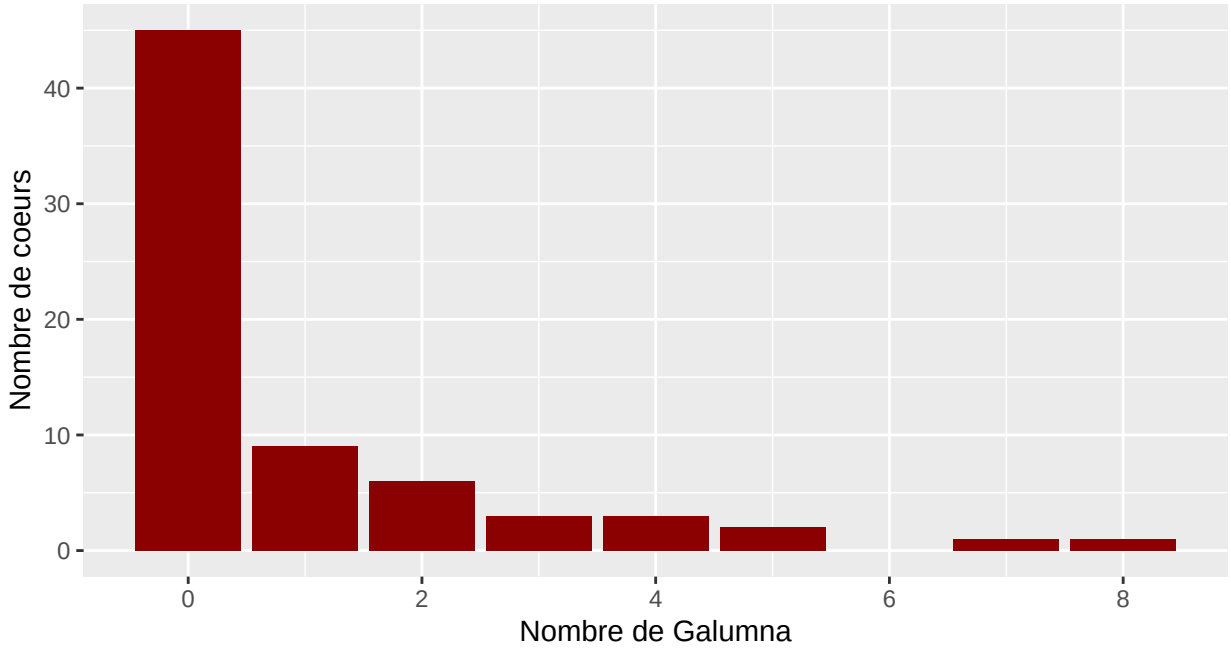


Figure 1: Distribution de Galumna

On remarque que plus de la moitié des coeurs étudiés ne contiennent pas de mites Galumna. La valeur maximale de Galumna contenues dans un coeur est 8.

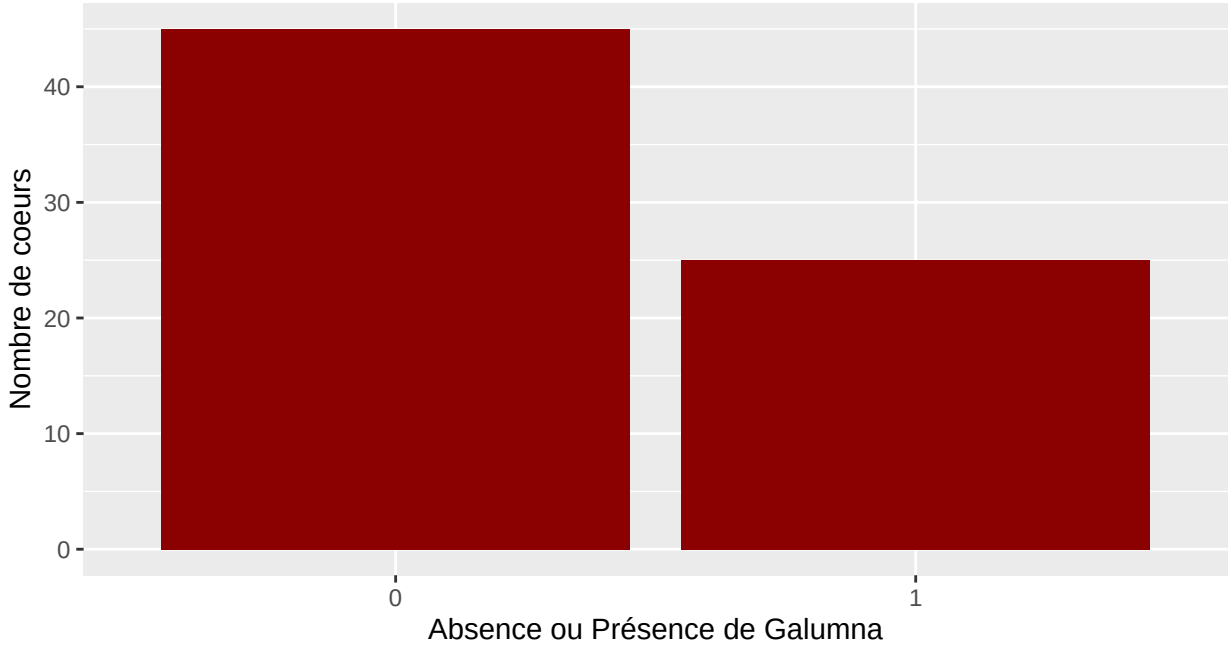


Figure 2: Distribution de pa

On voit ci-dessus plus précisément que 45 coeurs ne contiennent pas de Galumna et 25 coeurs en contiennent.

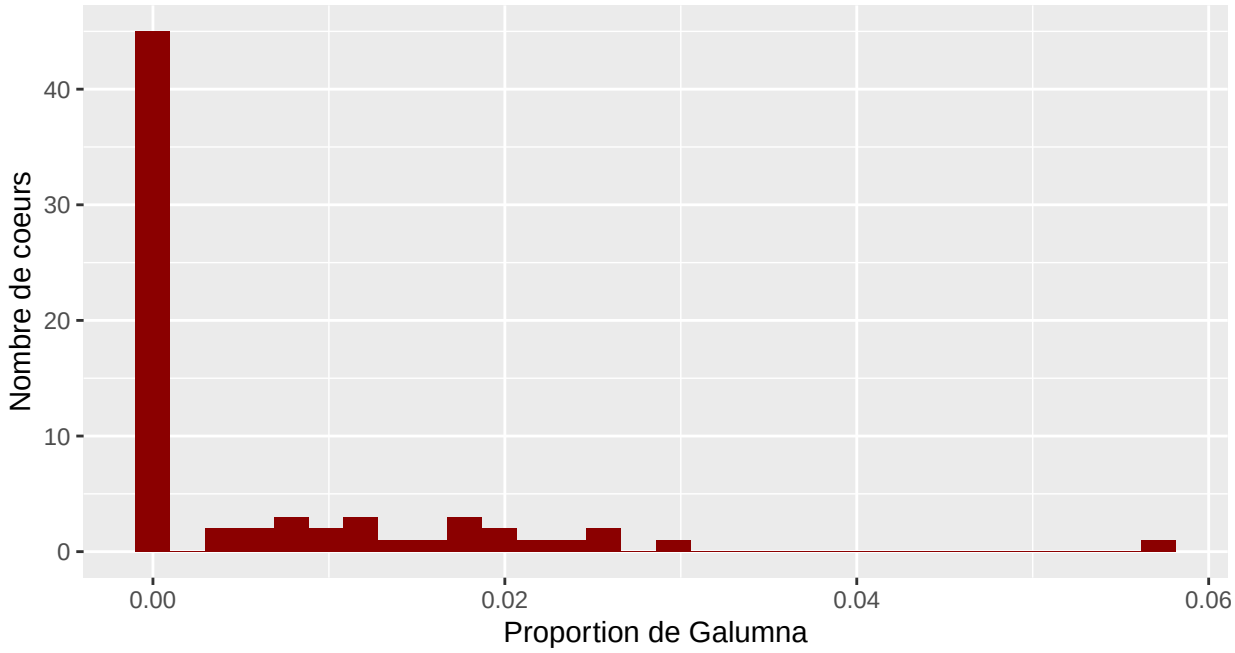


Figure 3: Distribution de prop

Naturellement, on remarque que 45 coeurs ont une proportion de Galumna de 0%. Pour les autres coeurs, la proportion varie entre 0 et 0.06%.

Passons maintenant aux variables explicatives.

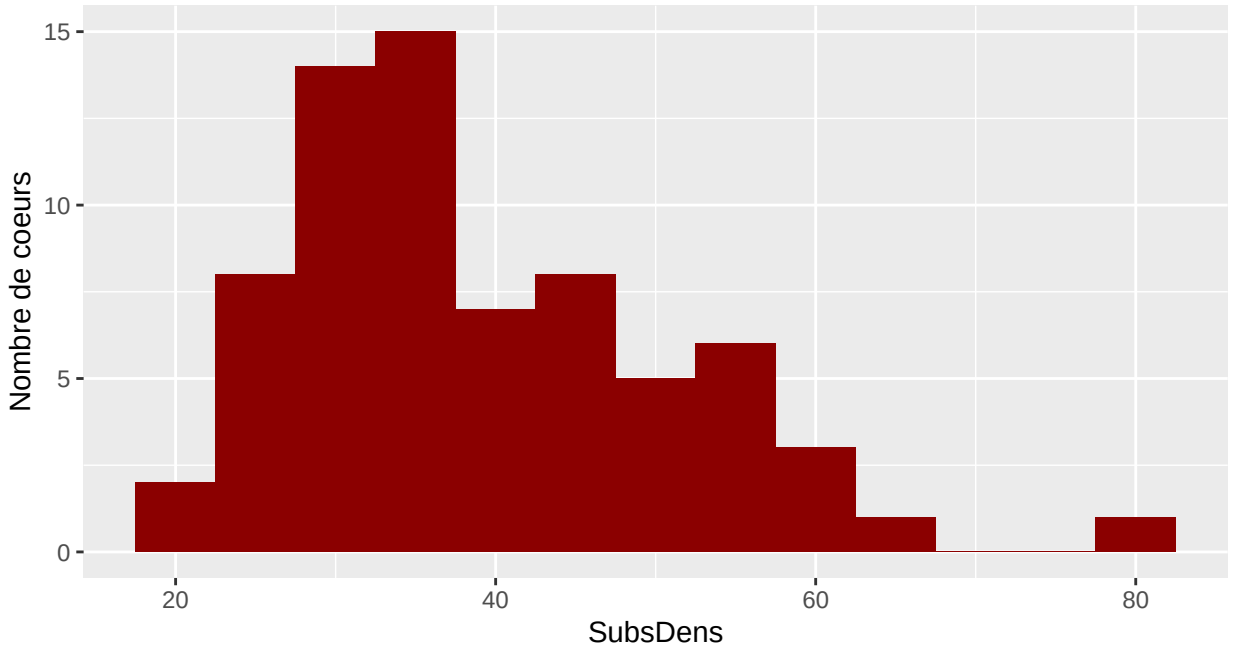


Figure 4: Distribution de SubsDens

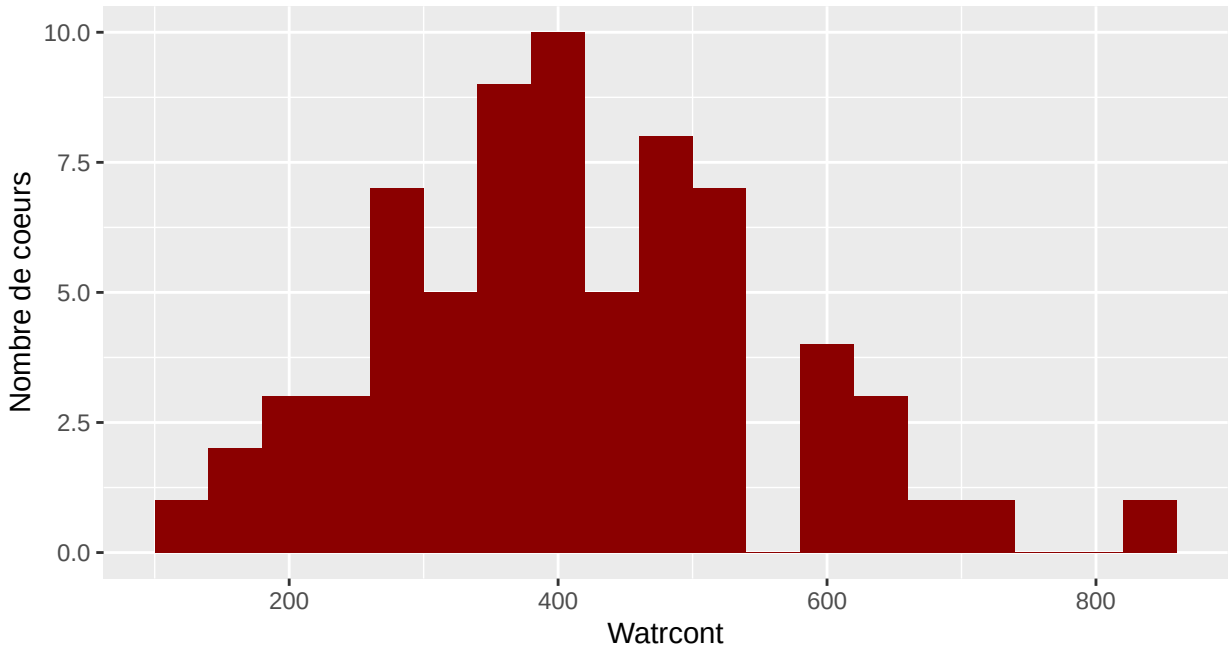


Figure 5: Distribution de WatrCont

Nous pouvons observer la distribution des variables quantitatives SubsDens et WatrCont. Cependant, ces variables étant déterministes, on ne fait pas d'hypothèse de normalité. Donc nous n'avons pas besoin de "vérifier" qu'elles suivent une loi normale, ni d'effectuer des transformations pour que cela soit le cas.

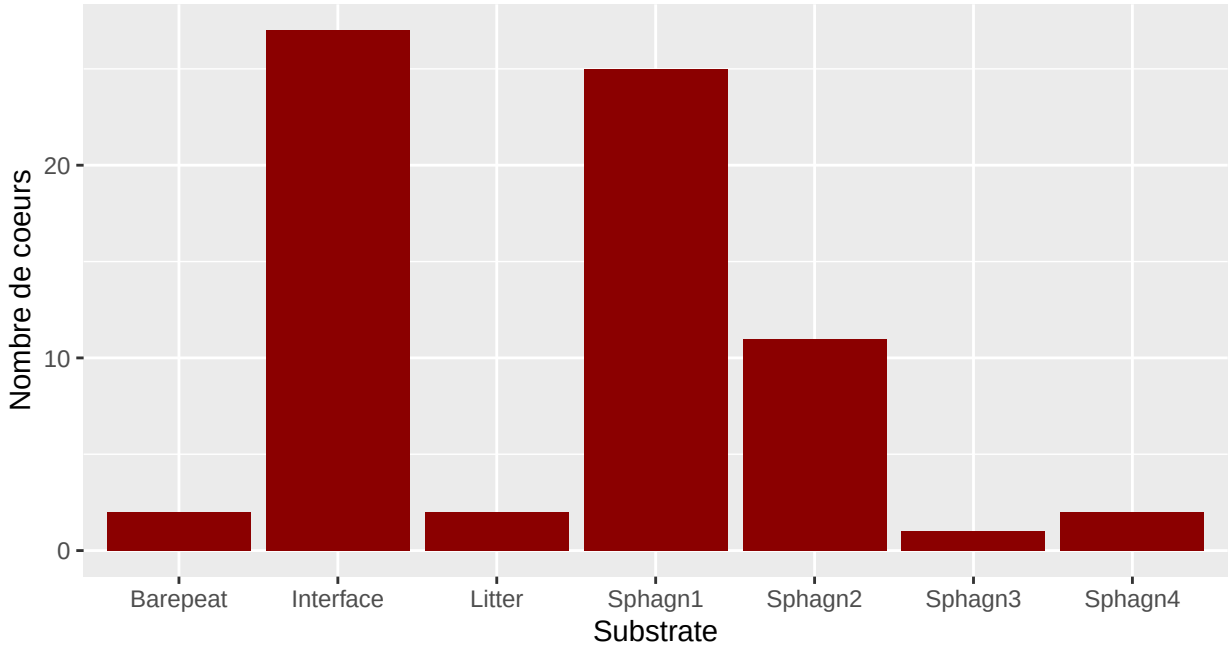


Figure 6: Distribution de Substrate

Les deux types de substrat majoritaires sont "Interface" et "Sphagn1", qui sont chacun le substrat d'environ 25 coeurs sur les 70 totaux.

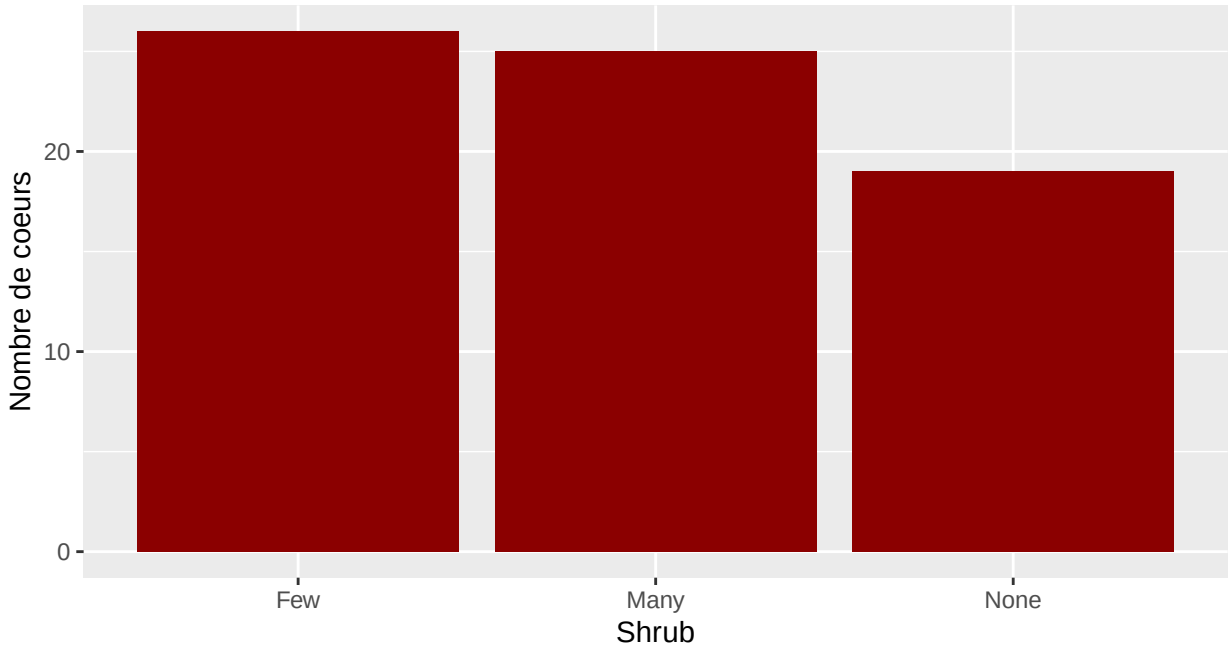


Figure 7: Distribution de Shrub

La variable `Shrub` représente les buissons du coeur. On voit qu'un peu moins d'une vingtaine de coeurs ne contient pas de buisson. Puis environ 25 coeurs contiennent beaucoup de buissons et peu de buissons.

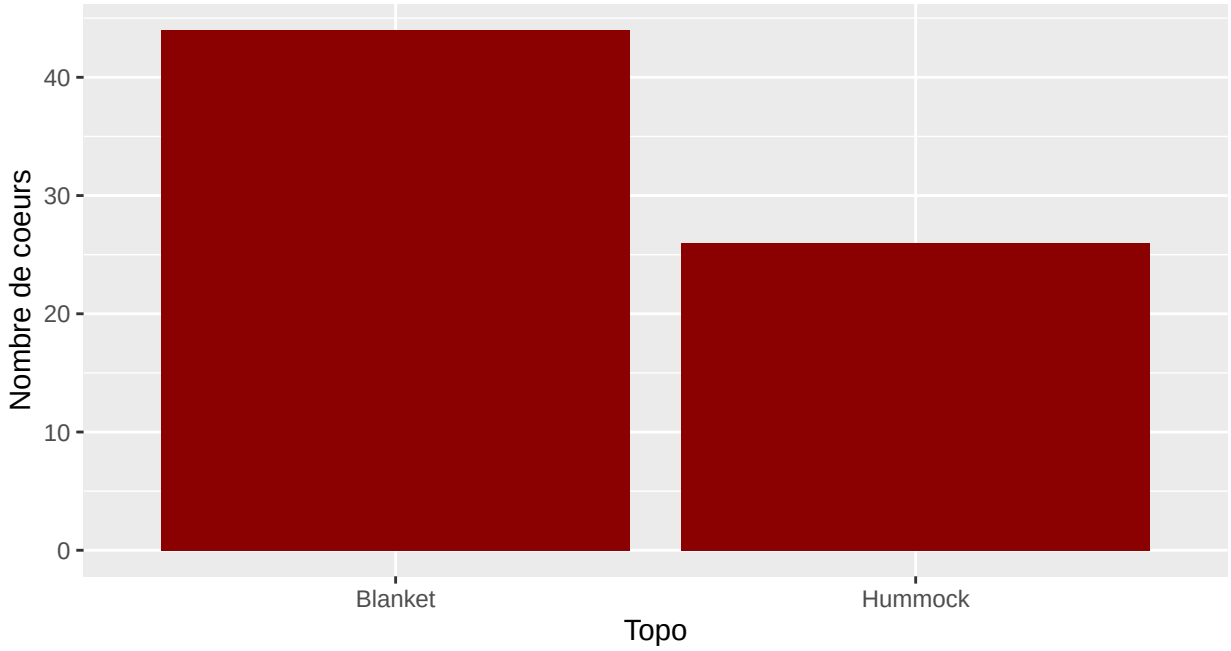


Figure 8: Distribution de Topo

Finalement, on voit qu'environ 45 coeurs ont une topographie de type "Blanket", les autres ont une topographie de type "Hummock".
Ce premier coup d'oeil aux données nous a permis de remarquer quelque chose d'important : plus de la moitié des coeurs ne contiennent pas de Galumna. Cela siginifie, pour ceux-ci, que la proportion de Galumna est nulle, que la variable Galumna est égale à 0 et que la variable occurence de Galumna est également égale à 0. Il est important de garder cela à l'esprit pour la future construction de nos modèles.

1.2 LM Gaussien

Dans un premier temps, nous allons essayer d'ajuster un modèle linéaire gaussien pour chaque variable réponse que l'on cherche à expliquer : `Galumna`, `prop`, `pa`. Mais un tel modèle doit vérifier plusieurs hypothèses. Le premier travail consiste donc à regarder si ces hypothèses sont vérifiées.

1.2.1 Vérification des hypothèses

Rappelons les hypothèses du modèle linéaire gaussien :

- sur la forme du modèle :
  (H1) **linéarité** du modèle

- sur les erreurs ϵ_i qui sont :
  (H2) **centrées** : $\mathbb{E}(\epsilon_i) = 0, \forall i = 1, ..n$, cette condition est toujours vérifiée par les moindres carrés ordinaires (MCO), technique de résolution de la régression linéaire
  (H3) **homoscédastiques** : $\mathbb{V}(\epsilon_i) = \sigma^2, \forall i$
  (H4) **non-autocorrélées** : $cor(\epsilon_i, \epsilon_i^{'}) = 0, \forall i \neq i'$
  (H5) **normales** : $\epsilon_i \sim N(0, \sigma^2)$

- sur les variables explicatives $X_1, ..., X_p$ qui sont :
  (H6) **non aléatoires** : la théorie se généralise facilement pour les variables aléatoires
  (H7) **non multicolinéaires** : les variables $X_1, ..., X_p$ sont linéairement indépendantes, ce qui garantit l'unicité de l'estimateur MCO.

Pour chaque modèle expliquant `Galumna`, `prop` et `pa`, nous regarderons si toutes ces hypothèses sont vérifiées, sauf la **2** et la **6** qui sont toujours vérifiées.

- **(H1) Linéarité du modèle**

Nous allons observer la linéarité du modèle dans un premier temps par représentation graphique. Nous représenterons chaque variable à expliquer en fonction des variables `WatrCont` et `SubsDens` qui sont les deux seules variables quantitatives explicatives.

Variable réponse Galumna :

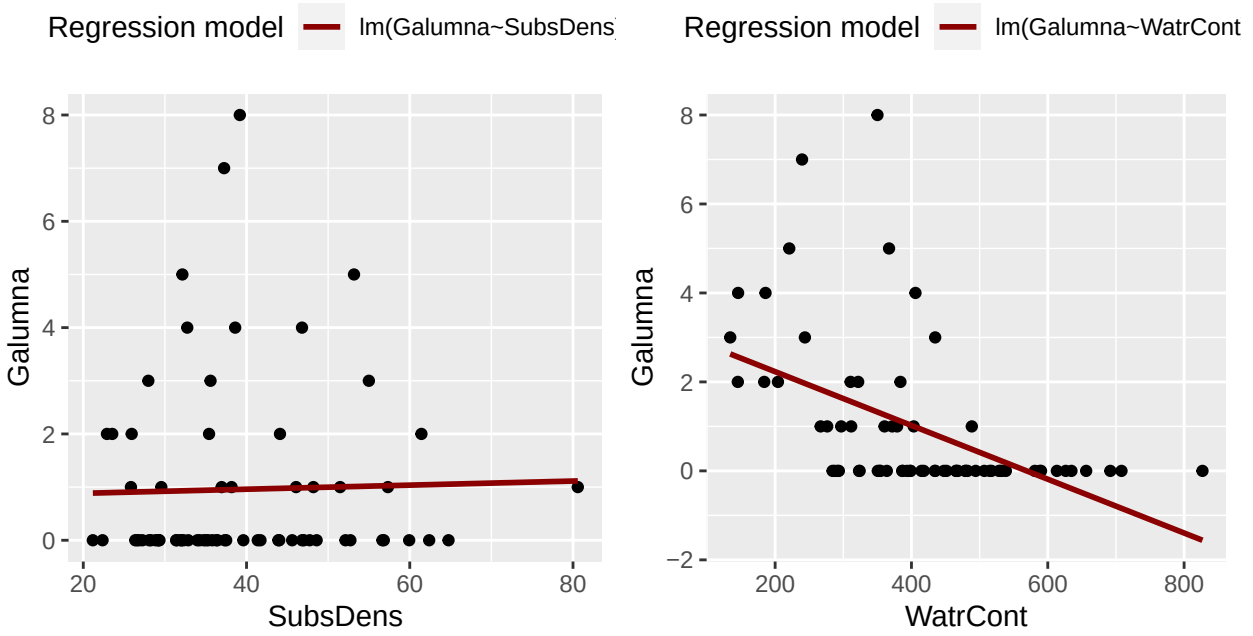


Figure 9: Modèles de régression sur la variable réponse Galumna

On ne voit pas un lien linéaire évident entre `Galumna` et `SubsDens`, ni entre `Galumna` et `WatrCont.` On voit que la quantité importante de points sur l'axe des abscisses (pour lesquels `Galumna`=0) semble beaucoup impacter la droite de régression.

Variable réponse pa :

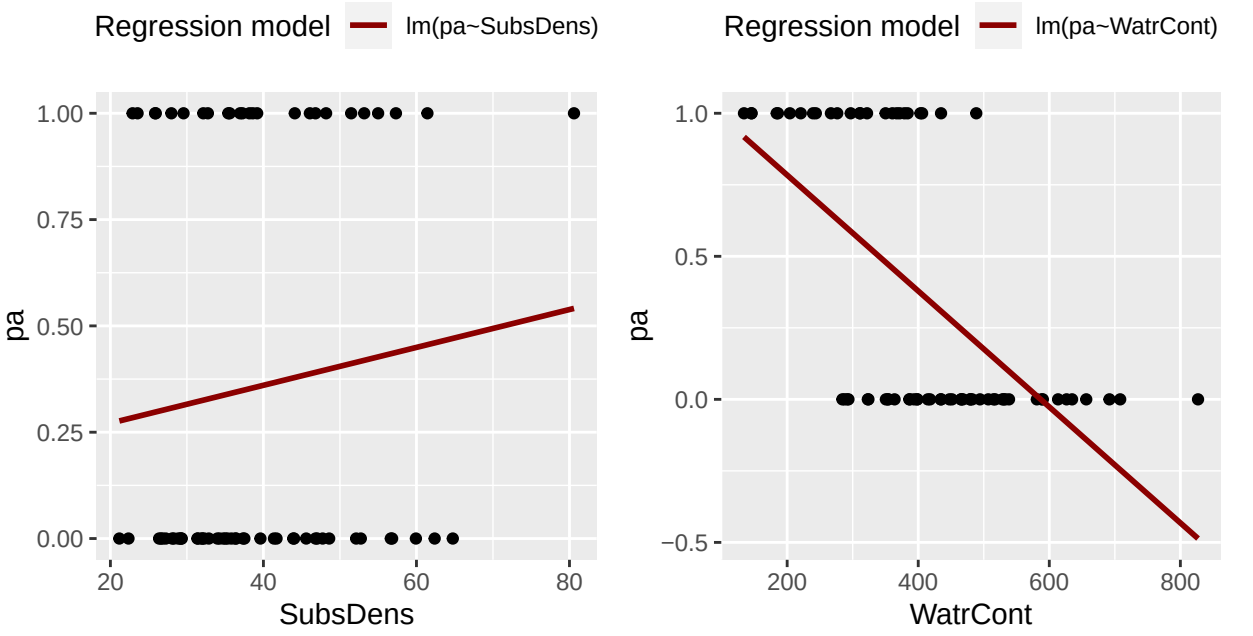


Figure 10: Modèles de régression sur la variable réponse pa

On ne voit pas non plus un lien linéaire entre la variable `pa` et les variables explicatives. De plus, comme c'est une variable binaire, il semblerait plus naturel de penser à une régression logistique qu'à une régression linéaire.

Variable réponse prop :

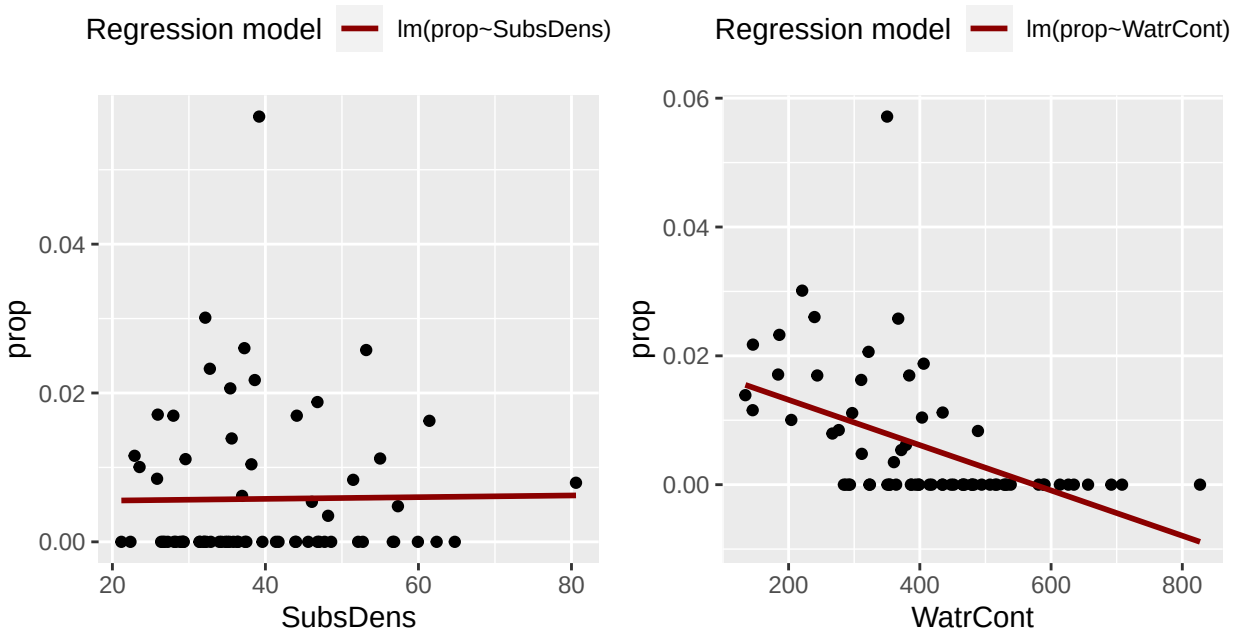


Figure 11: Modèles de régression sur la variable réponse prop

On fait le même constat pour la variable `prop`, il ne semble pas y avoir un lien linéaire particulier et la présence de beaucoup de points sur l'axe des abscisses semble influencer fortement la droite de régression.

Pour pousser notre analyse plus loin, nous allons effectuer un test de Harvey-Collier sur les modèles qui expliquent nos variables réponses par les deux variables explicatives quantitatives (`SubsDens` et `WatrCont`). Le test de Harvey-Collier (fonction `lmtest:harvtest`) consiste à faire un test de Student sur les résidus récursifs. Si la réelle relation entre les variables n'est pas linéaire mais convexe ou concave, la moyenne des résidus récursifs devrait être significativement différente de 0.

Table 1: Harvey-Collier Test

	p-value Harvey-Collier Test
Galumna~WatrCont+SubsDens	0.1604845
pa~WatrCont+SubsDens	0.0005844
prop~WatrCont+SubsDens	0.1092010

Au seuil de 5%, on peut conclure que :

- la relation entre la variable `Galumna` et les variables `WatrCont` et `SubsDens` est linéaire
- la relation entre la variable `pa` et les variables `WatrCont` et `SubsDens` n'est pas linéaire
- la relation entre la variable `prop` et les variables `WatrCont` et `SubsDens` est linéaire

À ce stade de l'analyse, on peut déjà conclure qu'il ne sera pas possible d'ajuster un modèle linéaire sur le variable `pa` car l'hypothèse de linéarité n'est pas vérifiée.

- **(H3) Erreurs** ϵ_i homoscédastiques

Pour la suite de la vérification des hypothèses, on ajuste un modèle complet sur chacune de nos variables réponses (i.e. un modèle qui fait intervenir toutes les variables explicatives à notre disposition).

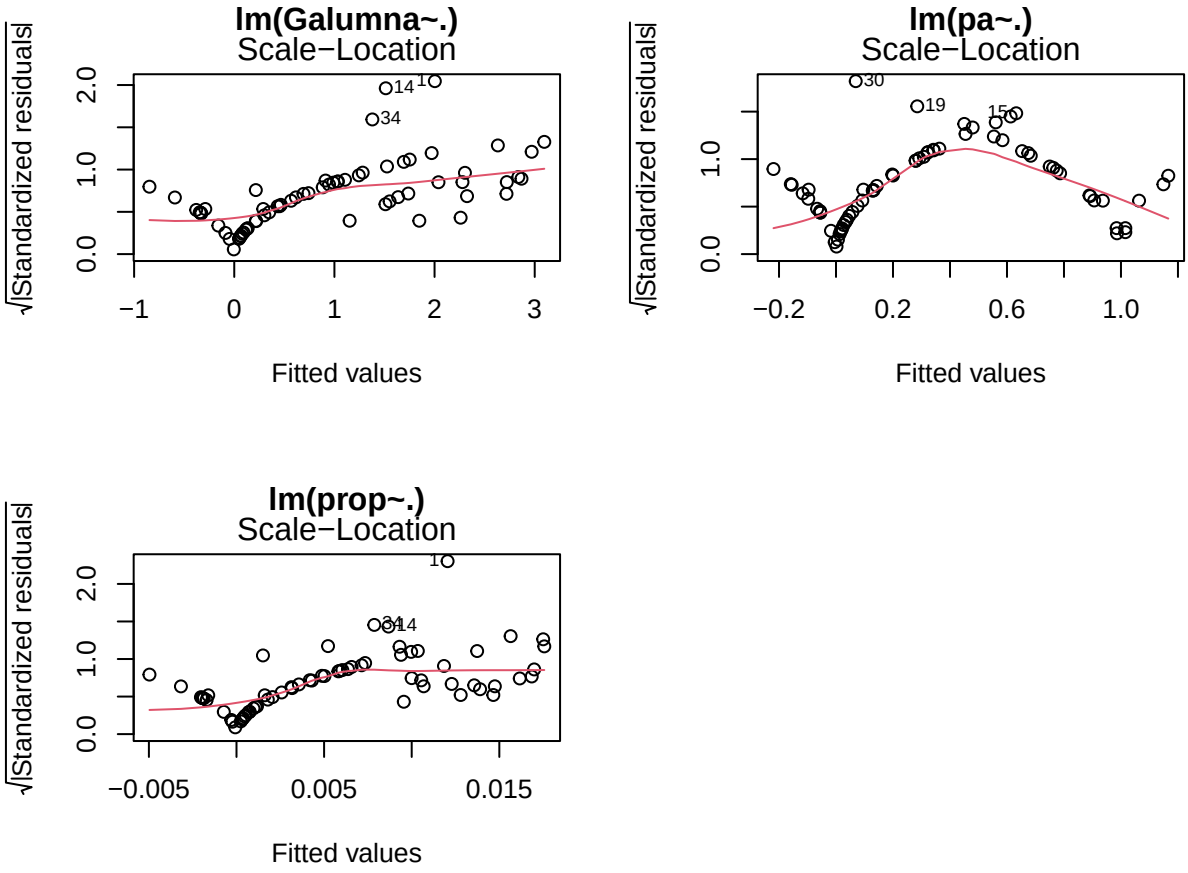


Figure 12: Variances des résidus des modèles de régression linéaire

Le type de graphique représenté ci-dessus est utilisé pour évaluer l'homoscédasticité des résidus. Si leur variance est bien constante, alors la courbe rouge doit former une ligne à peu près horizontale. On voit que cette hypothèse est plutôt bien respectée pour les modèles expliquant les variables `Galumna` et `prop`. Par contre, l'hypothèse n'est manifestement pas vérifiée pour le modèle expliquant la variable `pa`.

La vérification de cette hypothèse peut également se faire par un test de Breusch-Pagan (fonction `lmtest:bptest`) pour lequel on a H_0 : le bruit est homoscédastique, i.e. $\sigma_i^2 = \sigma^2$ pour tout i.

Table 2: Breusch-Pagan Test

	p-value Bruesch-Pagan Test
Galumna~WatrCont+SubsDens	0.6924990
pa~WatrCont+SubsDens	0.0226197
prop~WatrCont+SubsDens	0.8163084

Ce test confirme la conclusion basée sur les graphiques : on rejette l'hypothèse d'homoscédasticité des résidus pour le modèle expliquant `pa`, mais on l'accepte pour les deux autres modèles.

- **(H4) Erreurs** ϵ_i non corrélées

Afin de tester la non-autocorrélation des résidus du modèle linéaire, nous allons effectuer un test de Durbin-Watson (fonction `lmtest:dwtest`). Ce test cherche à évaluer la significativité du coefficient ρ dans la formule : $\epsilon_t = \rho\epsilon_{t-1} + u_t$ où ϵ_t est le résidu estimé du modèle et u_t un bruit blanc avec le test de Wald. L'hypothèse nulle est donc $H_0 : \rho = 0$, il y a non-autocorrélation des résidus.

Table 3: Durbin-Watson Test

	p-value Durbin-Watson Test
Galumna~WatrCont+SubsDens	0.0301950
pa~WatrCont+SubsDens	0.3649320
prop~WatrCont+SubsDens	0.0080226

On observe une p-value inférieure à 0.05 pour les modèles expliquant `Galumna` et `prop.` On peut dire qu'au seuil de 5%, on considère que l'hypothèse de non-autocorrélation des erreurs n'est pas respectée pour ces deux modèles. Elle est cependant vérifiée pour le modèle linéaire expliquant `pa.`
À ce stade de l'analyse, nous avons constaté que les hypothèses 1 et 3 ne sont pas vérifiées pour le modèle linéaire expliquant `pa`. L'hypothèse 4 n'est quant à elle pas vérifiée pour les modèles linéaires expliquant `Galumna` et `prop.` On pourrait donc s'arrêter et dire qu'il n'est pas possible d'ajuster un modèle linéaire sur ces trois variables. Par curiosité, nous allons continuer à regarder si les hypothèses suivantes sont vérifiées, bien que l'issue concernant les modèles linéaires reste la même.

- **(H5) Normalité des erreurs** ϵ_i

Dans le cadre d'un modèle linéaire, on suppose $\epsilon_i \sim N(0, \sigma^2)$. Une façon de vérifier cette hypothèse est de tracer les QQ-plot de nos modèles linéaires.

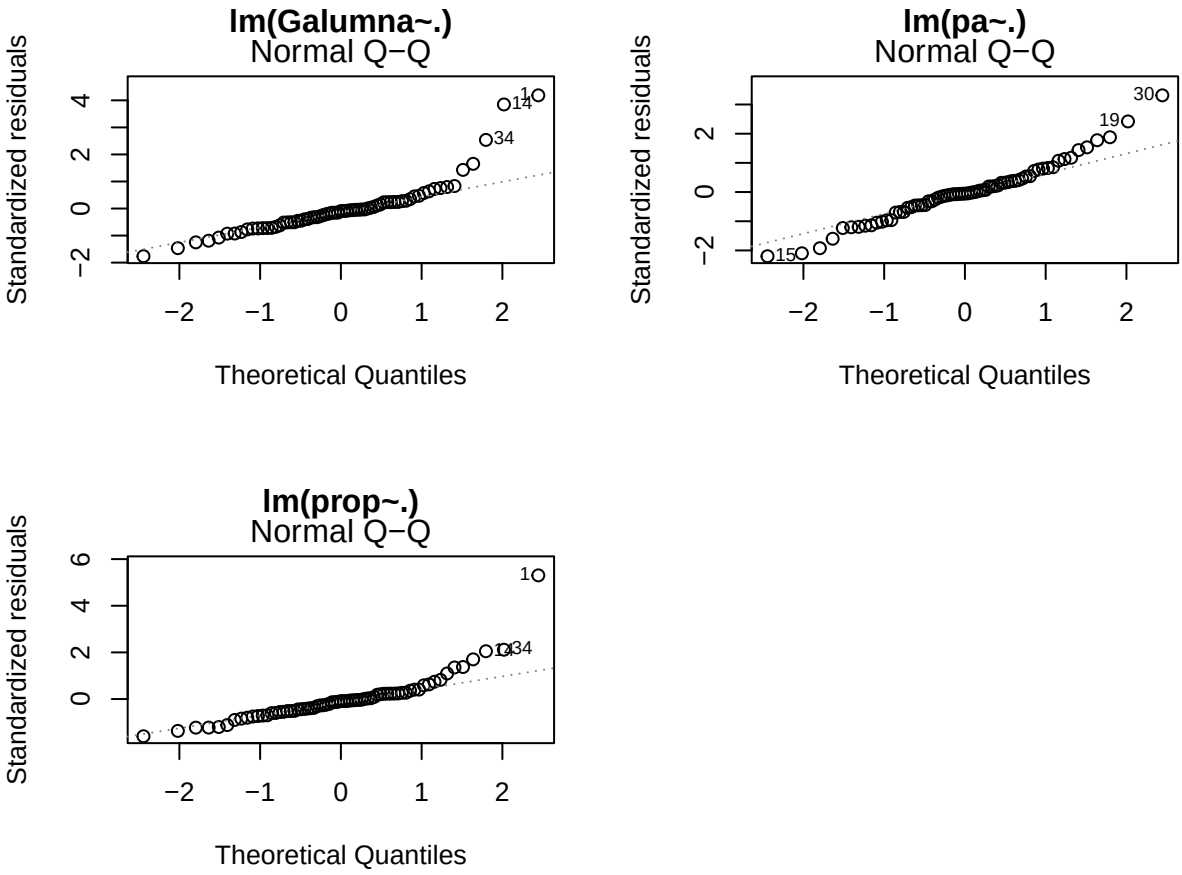


Figure 13: QQ-plot des modèles de régression linéaire

On voit que les points de nos différents modèles suivent à peu près tous la droite de normalité. Cependant, on remarque de nombreux points déviant de cette droite. Il est difficile de tirer une conclusion à partir de ces graphiques seuls. Il est également possible de réaliser un test de Shapiro-Wilk (fonction `stats:shapiro.test`). Il teste l'hypothèse nulle selon laquelle un échantillon $x_1, ..., x_n$ est issu d'une population normalement distribuée. Ici, notre échantillon sera les $\epsilon_1, ..., \epsilon_n$ du modèle linéaire considéré.

Table 4: Shapiro-Wilk Test

	p-value Shapiro-Wilk Test
Galumna~WatrCont+SubsDens	0.0000000
pa~WatrCont+SubsDens	0.0473534
prop~WatrCont+SubsDens	0.0000000

On observe une très petite p-value pour les trois modèles linéaires. On considère que l'hypothèse de normalité des résidus n'est pas vérifiée pour nos trois modèles.

- **(H7) Non-multicolinéarité des variables explicatives**

La dernière hypothèse qui doit être vérifiée lorsque l'on ajuste un modèle linéaire est la non-multicolinéarité des variables explicatives. Pour la vérifier, on peut faire de l'analyse bivariée, c'est-à-dire vérifier si les variables semblent être corrélées deux à deux, ou encore calculer les VIF.

Corrélations SubsDens~ :

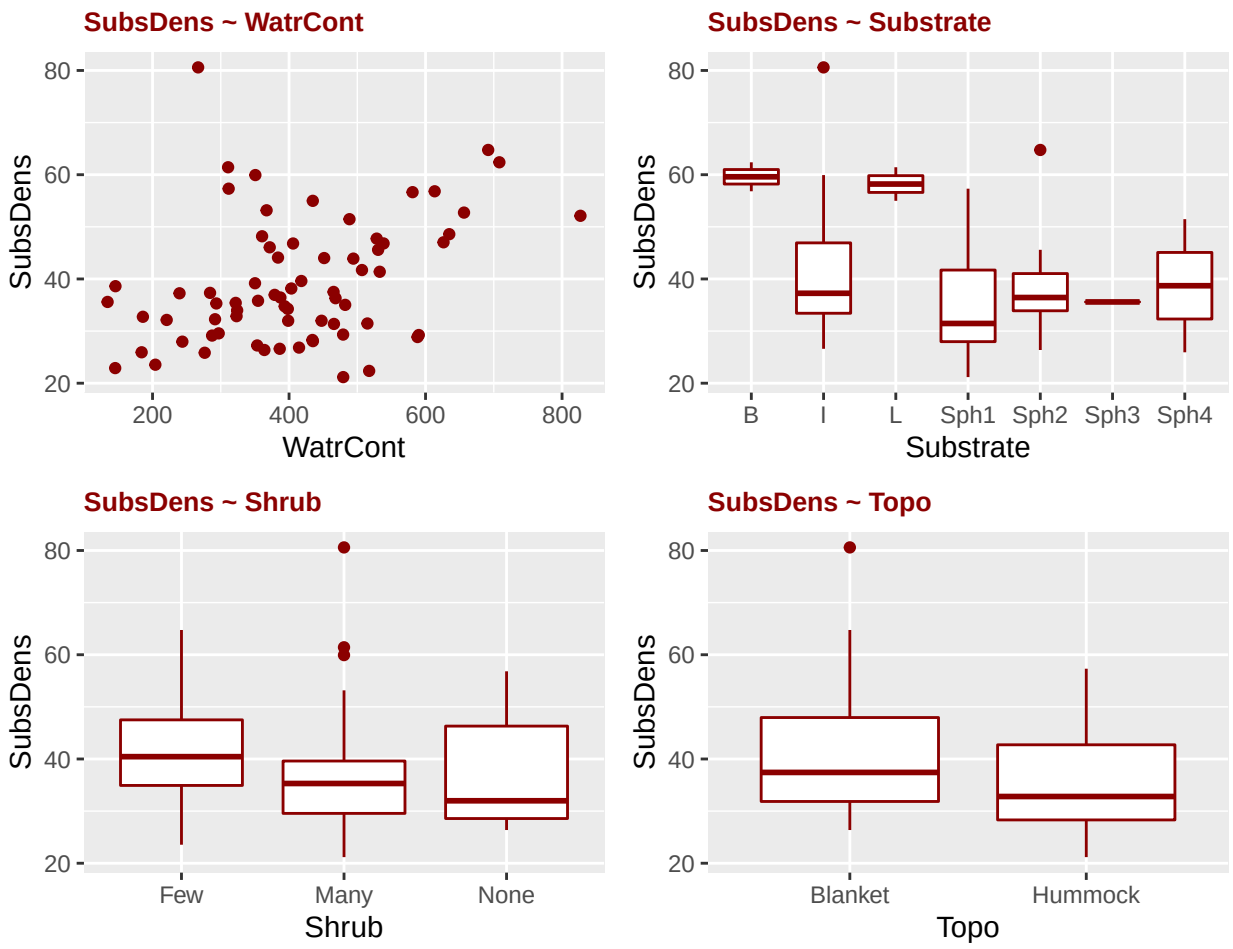


Figure 14: Corrélations avec la variable SubsDens

Il ne semble pas y avoir de corrélation particulière entre les variables quantitatives `WatrCont` et `SubsDens`, les points sont répartis de manière assez aléatoire. Il ne semble pas y avoir de corrélation entre `SubsDens` et les variables `Shrub` et `Topo` non plus : la moyenne de `SubsDens` ne semble pas varier significativement selon le facteur `Shrub` ou `Topo`. Cependant, le type de substrat (variable `Substrate`) semble influencer significativement la densité du substrat (`SubsDens`), cela paraît assez logique.

Corrélations WatrCont~ :

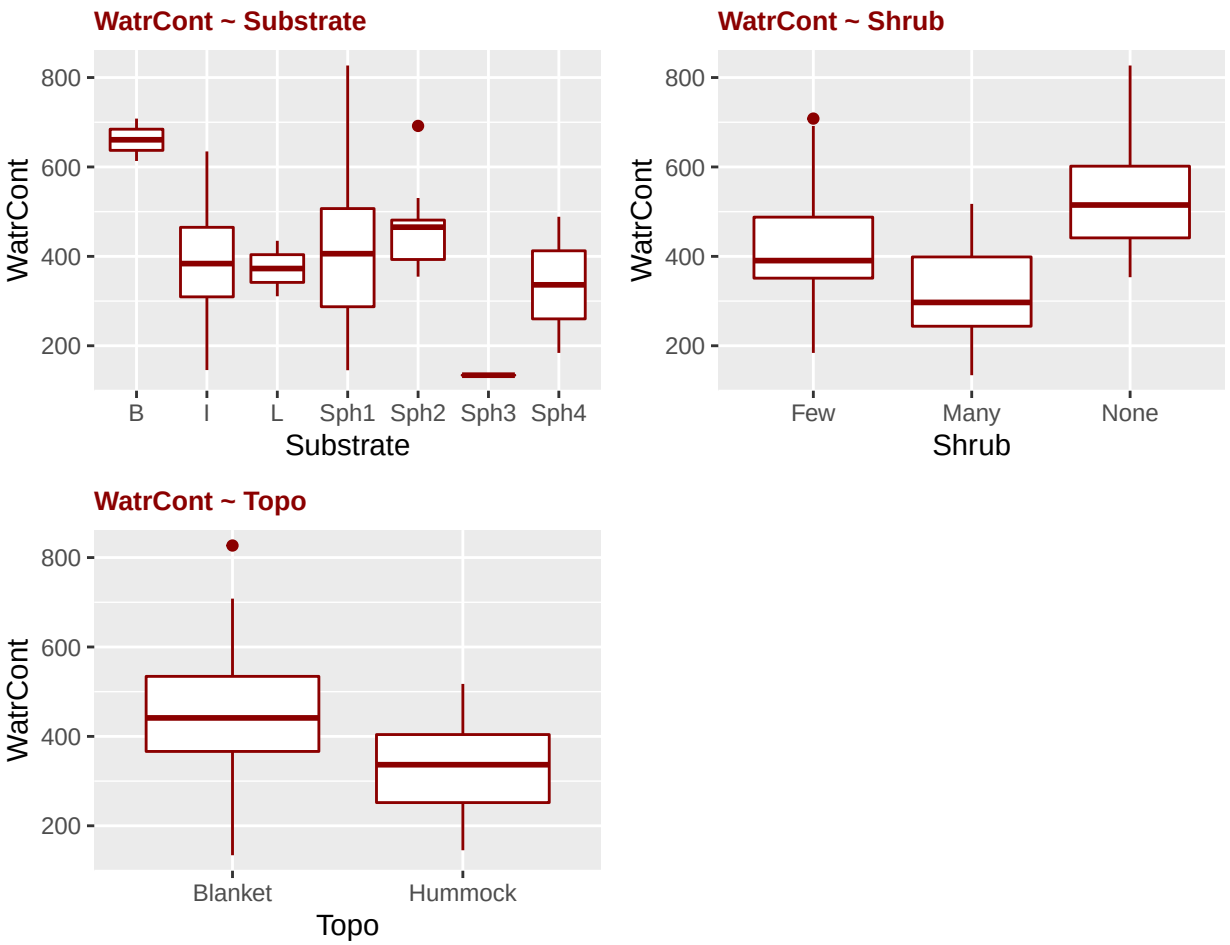


Figure 15: Corrélations avec la variable WatrCont

Les variables `Substrate` et `Shrub` semblent avoir une influence significative sur la quantité d'eau (variable `WatrCont`). Cependant, il est difficile de conclure qu'il y a présence ou absence de corrélation entre `WatrCont` et `Topo.`

Corrélations Substrate~ :

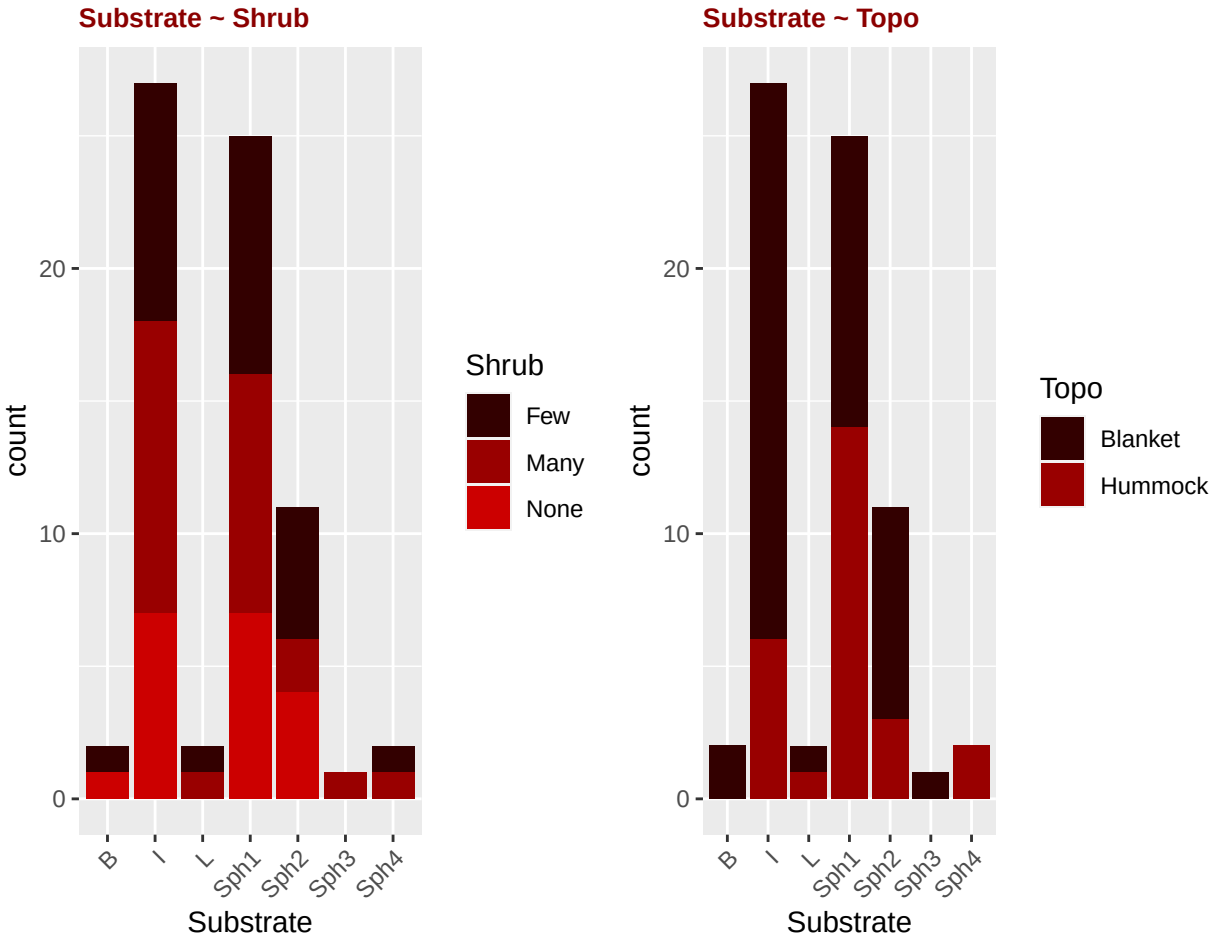


Figure 16: Corrélations avec la variable Substrate

Du fait du peu de donées que nous avons à disposition, il est difficile de conclure à partir des graphiques.

Corrélations Shrub~ :

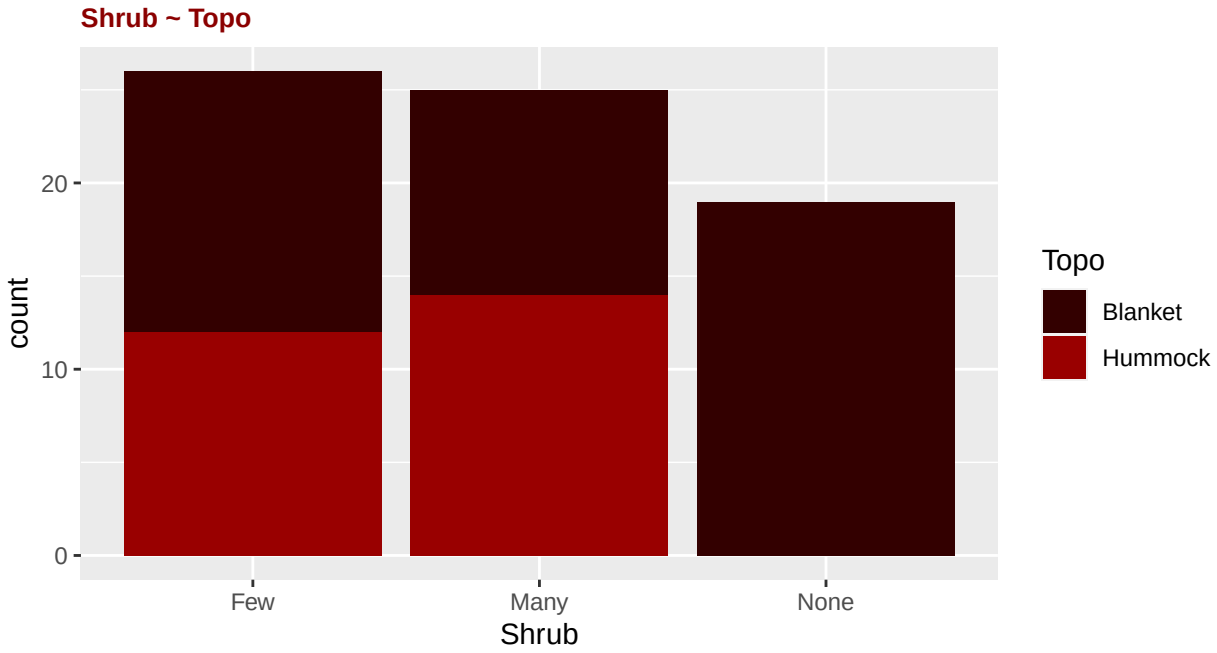


Figure 17: Corrélations avec la variable Shrub

Une nouvelle fois, au vu du faible nombre d'observations, il serait risqué de conclure uniquement à partir du graphique ci-dessus.

Calcul des VIF :

Table 5: VIF

	lm(Galumna~.)	lm(pa~.)	lm(prop~.)
SubsDens	1.683771	1.683771	1.683771
WatrCont	2.214753	2.214753	2.214753
Substrate	1.961711	1.961711	1.961711
Shrub	2.069477	2.069477	2.069477
Topo	1.804865	1.804865	1.804865

Les VIF sont par définition toujours supérieurs à 1. On considère que le VIF d'une variable devient trop élevé lorsque sa valeur dépasse 5. Ce n'est pas le cas pour notre modèle. On peut donc conclure grâce aux VIF, et à notre analyse bivariée qu'il y a non-multicolinéarité entre les variables explicatives.

Conclusion : après analyse de toutes les hypothèses, on conclut qu'au moins une des hypothèses n'est pas vérifiée pour chacun de nos modèles linéaires. Il est donc impossible d'ajuster un modèle linéaire sur nos données pour expliquer les variables `Galumna`, `pa` et `prop.` Cependant, on peut essayer d'effectuer des transformations sur les données afin qu'elles vérifient les hypothèses.

1.2.2 Transformations sur les données

Transformer les données peut parfois régler les problèmes posés par les hypothèses du modèle linéaire gaussien, et notamment la non-normalité et l'hétéroscédasticité des résidus. Cependant, ces transformations présentent des inconvénients :

- elles changent la variable réponse, ce qui peut compliquer l'interprétation
- elles ne peuvent pas toujours améliorer la linéarité et l'homogénéité de la variance en même temps
- les bornes de l'espace d'échantillonnage changent

Dans le cadre de notre étude, nous essaierons des transformations adaptées aux données en proportion pour notre variable `prop`. Ensuite, nous essaierons une transformation plus générale : celle de Box & Cox.

1.2.2.1 Transformation pour les données en proportion

Deux des principales transformations qui permettent de stabiliser la variance pour les données en proportion sont la transformation *logit* et la transformation *arcsin*. Cependant, pour des données écologiques, la transformation *arcsin* est souvent préférée car il est courant d'avoir des proportions à 0% ou 100%, ce qui pose problème pour effectuer une transformation *logit*. En effet, cela ferait donnerait lieu à des valeurs de $-\infty$ pour des proportions égales à 0. On rappelle que c'est notre cas : nous avions remarqué, lors de notre analyse préliminaire des données, que plusieurs observations avaient une valeur de 0 pour la variable prop. Il est donc impossible ici d'utiliser une transformation *logit* sur les données.

Transformation Arcsin :

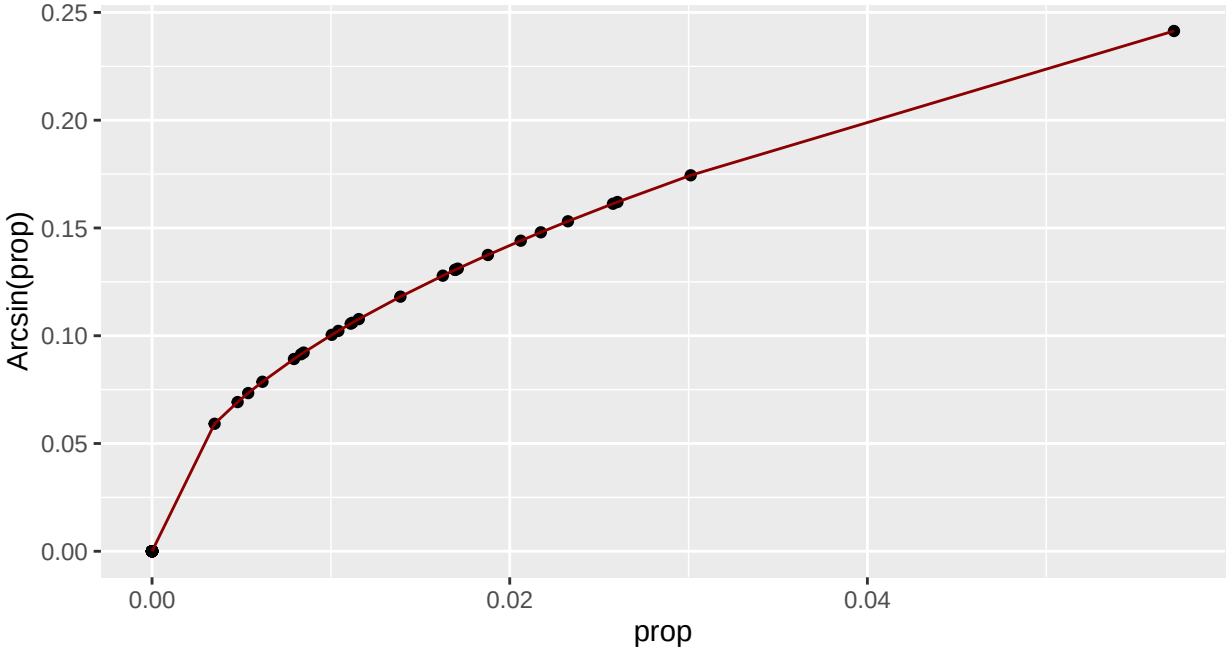


Figure 18: Transformation Arcsin

On voit ci-dessus que la transformation *arcsin* semble finalement assez proche d'une transformation linéaire. Regardons ce que cela provoque sur les hypothèses du modèle linéaire gaussien.
Rappelons que nous avions conclu, avant transformation, que l'hypothèse d'homoscédasticité des résidus était vérifiée. Cependant, ce n'était pas le cas pour l'hypothèse de normalité des résidus. Regardons ce qu'il en est après transformation.

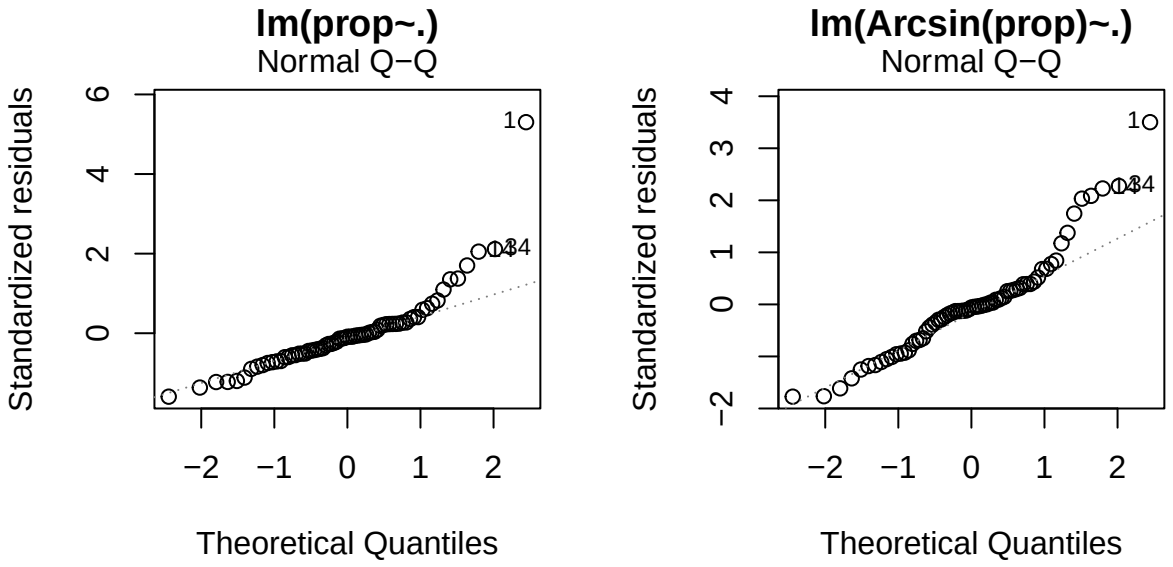


Figure 19: QQ-plot : à gauche sur données prop, à droite sur données prop après transformation Arcsin

Graphiquement, la transformation *arcsin* ne semble pas avoir amélioré la normalité des résidus. Vérifions avec un test de Shapiro-Wilk :

```
## p-value Shapiro-Wilk Test : 0.0002163341
```

Une nouvelle fois, on rejette H_0 et on conclut que l'hypothèse de normalité des résidus n'est toujours pas vérifiée, même après une transformation *arcsin*.

1.2.2.2 Transformation de Box-Cox

La tranformation de Box & Cox est une transformation non linéaire souvent utilisée pour rendre les données normales. L'objectif est donc d'obtenir une distribution normale des données après transformation, et une variance constante. La transformation s'obtient comme suit :

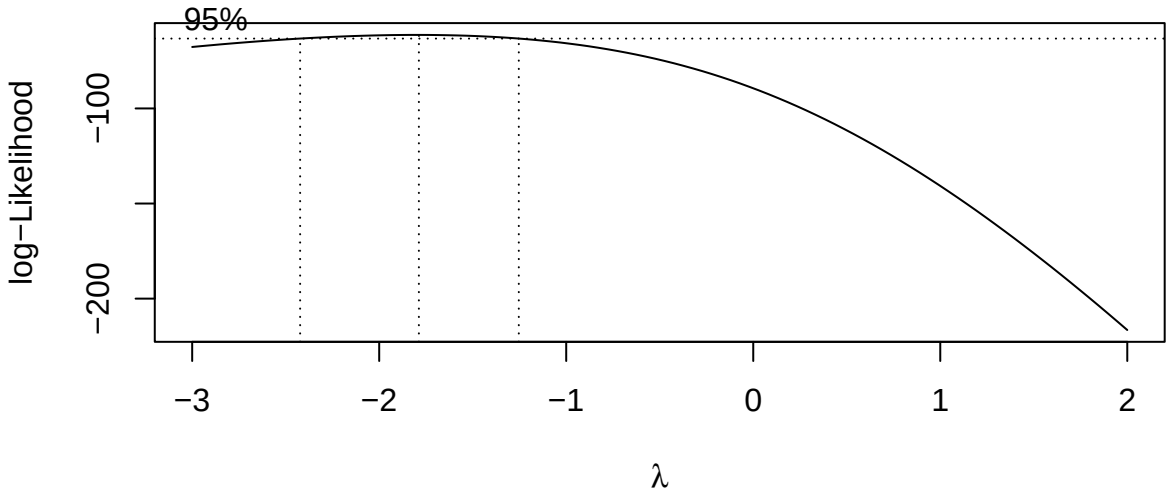
$$\{y \in \mathbb{R}_*^+, \lambda \in \mathbb{R}\} : y^* = f(y, \lambda) = \begin{cases} \frac{y^\lambda - 1}{\lambda} & (\lambda \neq 0) \\ \log(y) & (\lambda = 0) \end{cases}$$

Dans le cas où $\lambda = 0$, on retrouve une transformation logarithmique classique. Dans le cas où $\lambda = 1$, cela revient à ne pas faire de transformation et de conserver notre variable d'origine à une translation près. Pour un échantillon de n observations, on applique cette même transformation à chaque valeur. Attention, la transformation n'est pas définie pour les valeurs négatives ou nulles de la variable. Dans ce cas de figure, Box & Cox proposent de translater toutes les valeurs :

$$\{y \in \mathbb{R}_*^+, \lambda_1 \in \mathbb{R}, \lambda_2 \in \mathbb{R}\} : y^* = f(y, \lambda_1, \lambda_2) = \begin{cases} \frac{(y+\lambda_2)^{\lambda_1} - 1}{\lambda} & (\lambda_1 \neq 0) \\ \log(y + \lambda_2) & (\lambda_1 = 0) \end{cases}$$

Il faut choisir λ_2 tel que $\forall y, y + \lambda_2 > 0$. Dans notre cas, on fixera $\lambda_2 = 1$ et on utilisera la fonction `MASS::boxcox` pour estimer λ_1. Appliquons maintenant cette transformation à chacune de nos variables réponses, puis re-vérifions les hypothèses du modèle linéaire gaussien, et plus particulièrement l'hypothèse de normalité qui était rejetée pour nos trois modèles et qui est supposée avoir été améliorée par la transformation.

Variable réponse Galumna :




```
## Lambda_1 optimal :  -1.787879
```

Si la valeur 0 avait été contenue par l'intervalle de confiance pour λ_1, il aurait été judicieux de simplement effectuer une transformation logarithmique. Ici, ce n'est pas le cas, on va donc réajuster un modèle sur les données transformées par Box & Cox avec le λ_1 qui maximise la log-vraisemblance (ici $\lambda_1 = -1.8$).

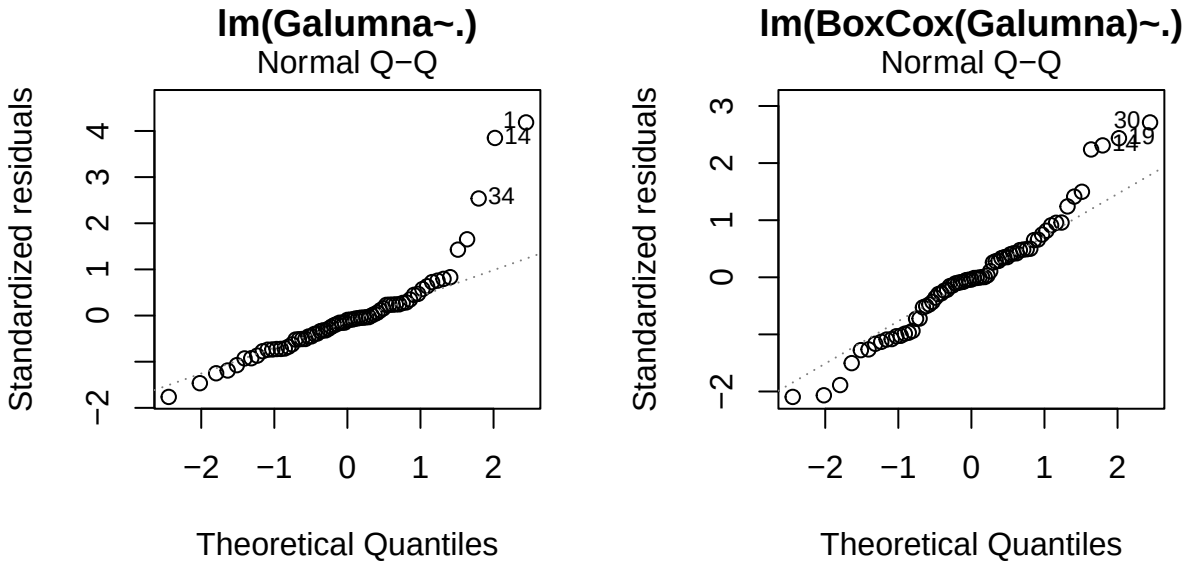
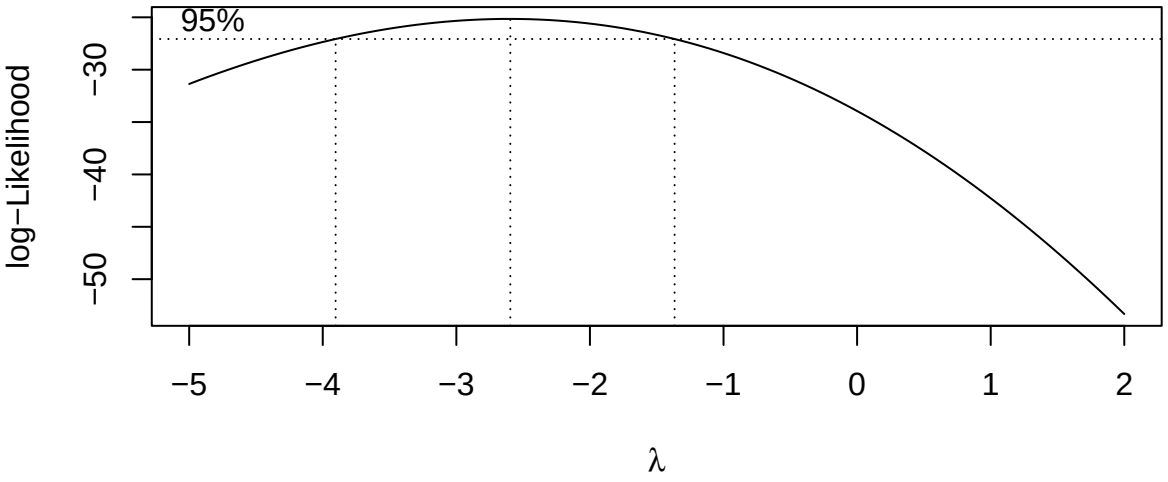


Figure 20: QQ-plot : à gauche sur données Galumna, à droite sur données Galumna après transformation Box Cox

```
## p-value pour le test de Shapiro-Wilk :  0.02953752
```

Graphiquement, la transformation de Box & Cox ne semble pas avoir amélioré la normalité des résidus. Pour le test de Shapiro-Wilk, on obtient une p-value inférieure à 5% donc on rejette l'hypothèse de normalité des résidus. La transformation de Box & Cox pour la variable Galumna n'a pas réglé le problème des hypothèses du modèle linéaire gaussien non vérifiées.

Variable réponse pa :




```
## Lambda_1 optimal :  -2.59596
```

On réajuste un modèle sur les données transformées par Box & Cox avec $\lambda_1 = -2.6$.

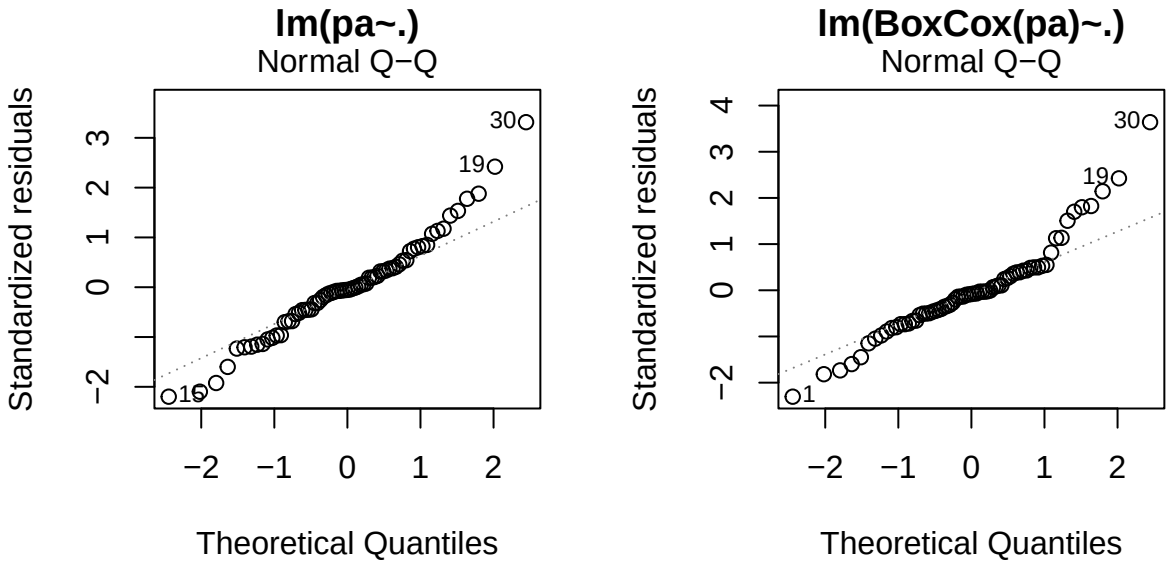
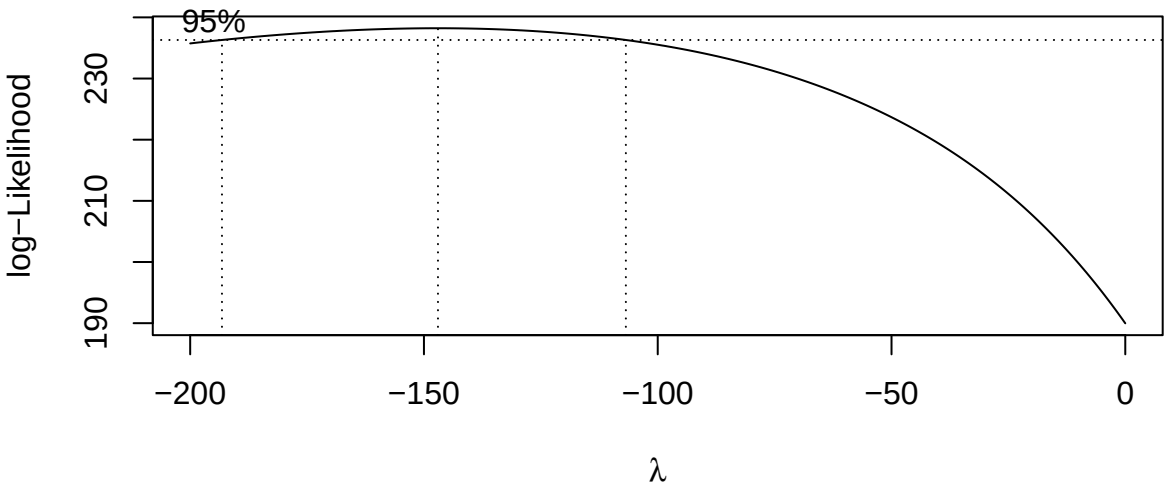


Figure 21: QQ-plot : à gauche sur données pa, à droite sur données pa après transformation Box Cox

```
## p-value pour le test de Shapiro-Wilk :  0.0005016299
```

Une nouvelle fois, graphiquement la transformation ne semble pas avoir amélioré la normalité des résidus. Pour le test de Shapiro-Wilk, on obtient une p-value très faible donc on rejette une fois de plus l'hypohtèse de normalité des résidus.

Variable réponse prop :




```
## Lambda_1 optimal :  -147
```

On réajuste un modèle sur les données transformées par Box & Cox avec $\lambda_1 = -147$. Il s'agit d'une valeur de λ_1 très élevée comparée aux valeurs précédents, mais c'est celle qui maximise la log-vraisemblance.

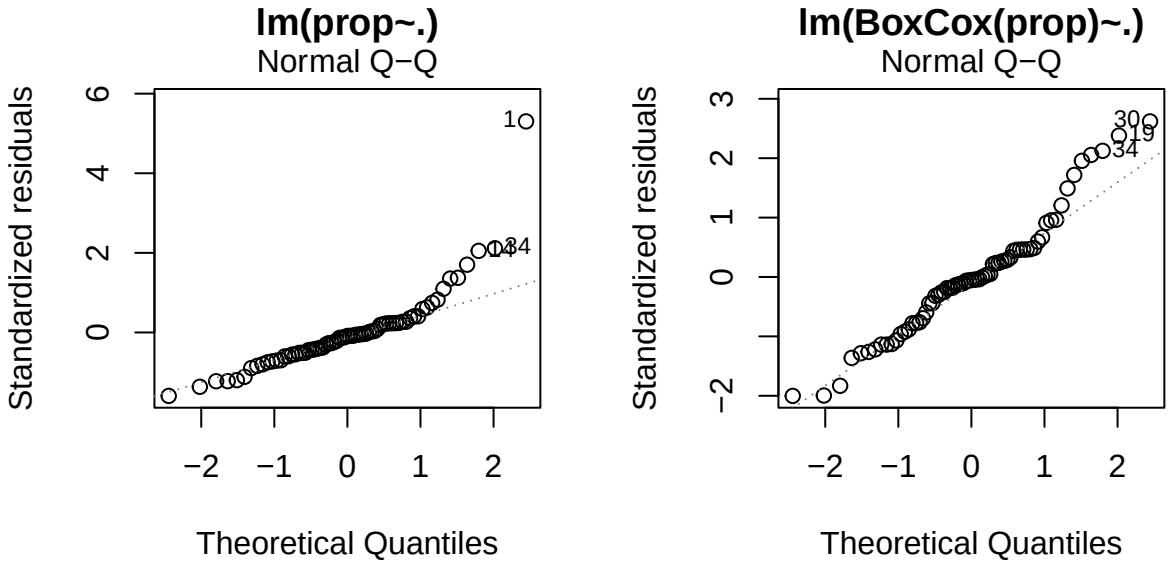


Figure 22: QQ-plot : à gauche sur données prop, à droite sur données prop après transformation de Box Cox

```
## p-value pour le test de Shapiro-Wilk :  0.02953752
```

Graphiquement, la transformation semble avoir légèrement amélioré la normalité des résidus (ce n'est pas la même échelle pour y pour les deux graphiques). Cependant, d'après le test de Shapiro-Wilk, on rejette l'hypothèse de normalité.

Conclusion :

Dans cette première partie de l'analyse du jeu de données "mites", nous avons tenté d'ajuster un modèle linéaire gaussien sur trois variables explicatives. Cependant, l'une des difficultés lorsque l'on ajuste un modèle linéaire est qu'il doit vérifier un certain nombre d'hypothèses. Dans notre cas, il y avait toujours au moins une hypothèse qui n'était pas vérifiée. Nous avons donc mis en place des transformations de données, adaptées à nos variables, afin que le modèle vérifie les hypothèses. Malheureusement, ces transformations n'ont pas permis de valider les hypothèses, en particulier celle de normalité des résidus.
Dans la pratique, il existe de nombreuses autres distributions de probabilité pour décrire les données : Poisson, Bernouilli, Binomiale... Nous allons donc essayer une autre approche pour ajuster un modèle sur nos données : les modèles linéaires généralisés.
Un modèle linéaire généralisé a quatre composantes :

1. les variables réponses $Y_1, Y_2, \ldots, Y_n$ sont supposées suivre la même distrubtion appartenant à la **famille exponentielle**
2. un vecteur de paramètres β à estimer et des variables explicatives $X = [1, x^1, \ldots, x^p]$
3. une fonction de lien g telle que
   $g(\mu_i) = x_i^T \beta$ avec $E(Y_i) = \mu_i$
4. une fonction de variance $V(\mu_i)$ qui décrit comment la variance $Var(Y_i)$ dépend de l'espérance $E(Y_i) = \mu_i$:

   $$Var(Y_i) = \phi V(\mu_i)$$

   où ϕ est un paramètre de dispersion ou facteur d'échelle (constante) et V une fonction de variance.
   Finalement, le modèle linéaire gaussien est un cas particulier du modèle linéaire généralisé où l'on a $Y = X\beta + \epsilon$ avec les ϵ_i i.i.d. suivant $N(0, \sigma^2)$. Dans ce cas, $\mu_i = E(Y_i) = x_i^T \beta$ et la fonction de lien g est la fonction identité $g(\mu_i) = \mu_i = x_i^T \beta$.

Sous R, la fonction utilisée pour les modèles linéaires généralisés est `glm`.

Nous allons donc essayer des modèles linéaires généralisés avec différentes distributions et fonctions de lien pour nos données :

- une régression logistique binaire sur la variable `pa`
- une régression logistique sur la variable `prop`
- une régression de Poisson sur la variable `Galumna`

1.3 GLM Régression Logistique binaire sur la variable d'occurrence (pa)

Objectif : expliquer l'occurence `pa`, déterminer le meilleur modèle, interpréter les coefficients, puis évaluer l'ajustement de ce modèle et son pouvoir prédictif.

La variable binaire est un type de variable commun dans les données en écologie ou en santé : on observe un phénomène Y ou son absence. Dans notre cas, on veut savoir si l'occurence de mites Galumna varie en fonction de l'environnement.

Ici, le modèle linéaire généralisé est composé de :

1. la variable réponse $Y_i, i = 1, ..., n$ qui suit une distribution de Bernouilli de paramètre π_i
2. un vecteur de paramètre β à estimer et des variables explictives $X = [1, x^1, \ldots, x^p]$
3. la fonction de lien g nommée *logit*. Elle est définie de $[0, 1]$ sur $\mathbb{R}$ par $g(\pi) = ln(\frac{\pi}{1-\pi})$, soit $\pi_i = g^{-1}(x_i^T\beta) = \frac{\exp(x_i^T\beta)}{1+\exp(x_i^T\beta)}$

Ainsi, on suppose ici que y_i suit une loi Binomiale $B(1, \frac{\exp(x_i^T\beta)}{1+\exp(x_i^T\beta)})$.

Sous R, on indique en paramètre de la fonction `family="binomial"` et il n'est pas nécessaire de préciser la fonction de lien car par défaut, pour la famille binomiale, la fonction de lien est *logit*.

1.3.1 Recherche du meilleur modèle

- **Méthode "à la main"**

Pour débuter, on construit le modèle complet (avec toutes les variables explicatives) pour déterminer lesquelles sont significatives :

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## Call:
## glm(formula = pa ~ WatrCont + SubsDens + Topo + Shrub + Substrate,
##     family = "binomial", data = mites)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.96119  -0.20137  -0.00004   0.11385   2.26516
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -1.430e+01  1.022e+04  -0.001   0.9989
## WatrCont           -2.162e-02  8.505e-03  -2.542   0.0110 *
## SubsDens            1.625e-01  7.690e-02   2.114   0.0345 *
## TopoHummock         2.135e+00  1.046e+00   2.040   0.0413 *
## ShrubMany           3.918e-01  1.057e+00   0.371   0.7109
## ShrubNone          -1.585e+01  3.627e+03  -0.004   0.9965
## SubstrateInterface  1.351e+01  1.022e+04   0.001   0.9989
## SubstrateLitter     3.251e+01  1.563e+04   0.002   0.9983
## SubstrateSphagn1    1.523e+01  1.022e+04   0.001   0.9988
## SubstrateSphagn2    1.393e+01  1.022e+04   0.001   0.9989
## SubstrateSphagn3    3.158e+01  2.047e+04   0.002   0.9988
## SubstrateSphagn4    3.387e+01  1.577e+04   0.002   0.9983
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 91.246  on 69  degrees of freedom
## Residual deviance: 29.897  on 58  degrees of freedom
## AIC: 53.897
##
## Number of Fisher Scoring iterations: 19
```

Lorsque l'on effectue le `summary` du modèle complet, on obtient un warning qui nous dit qu'une ou plusieurs observations du jeu de données sont prédites à 0 ou 1, ce qui ne veut pas forcément dire que la régression logistique est mauvaise. On rappelle que dans notre jeu de données on avait des proportions nulles. On décide donc d'ignorer le warning. La sortie du `summary` nous permet de voir quelles variables sont significatives pour le test de Wald. Celui-ci permet de tester la nullité d'un coefficient : $H_0 : \beta_j = 0$. Sous H_0, on a :

$$\sqrt{n}(I(\hat{\beta}))^{\frac{1}{2}}_{jj}(\hat{\beta}_j) = \frac{\hat{\beta}_j}{\sqrt{\hat{Var}(\hat{\beta}_j)}} \sim N(0,1)$$

Ici, les variables significatives semblent être WatrCont, SubsDens et Topo.

On peut alors tester un modèle uniquement avec celles-ci :

```
##
## Call:
## glm(formula = pa ~ WatrCont + SubsDens + Topo, family = "binomial",
##     data = mites)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1471  -0.2356  -0.1091   0.3515   2.3526
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.582892   2.223158  -0.262 0.793174
## WatrCont    -0.021623   0.006252  -3.459 0.000543 ***
## SubsDens     0.172951   0.064303   2.690 0.007153 **
## TopoHummock  2.737763   0.940022   2.912 0.003586 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 91.246  on 69  degrees of freedom
## Residual deviance: 34.475  on 66  degrees of freedom
## AIC: 42.475
##
## Number of Fisher Scoring iterations: 7
```

Ce modèle semble mieux car les variables sont toutes vraiment significatives.

Afin de s'assurer de la significativité du retrait des autres variables, nous pouvons réaliser un test de modèles emboîtés. L'idée est de tester le modèle complet

$$Y = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p + \epsilon$$

contre le sous-modèle

$$Y = \beta_0 + \beta_1 X_1 + ... + \beta_{p-q} X_{p-q} + \epsilon$$

dans lequel on n'a pas pris en compte les q dernières variables. Cela revient à tester dans le modèle global $H_0 : \beta_{p-q+1} = ... = \beta_p = 0$ contre H_1 : le contraire. Si $rg(X) = p$ et $\epsilon \sim N(0, \sigma^2 I_n)$ alors sous H_0 :

$$F = \frac{n-p}{q} \frac{SCR_c - SCR}{SCR} \sim F(q, n-p)$$

où SCR est la somme des carrés des résidus dans le modèle complet et SCR_c la somme des carrés des résidus du sous-modèle. On l'effectue avec la fonction `stats::anova`.

```
## Analysis of Deviance Table
##
## Model 1: pa ~ WatrCont + SubsDens + Topo
## Model 2: pa ~ WatrCont + SubsDens + Topo + Shrub + Substrate
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1        66     34.475
## 2        58     29.897  8   4.5781   0.8016
```

La grande p-value nous indique que les variables présentes dans le plus grand modèle mais pas dans les plus petit ne sont pas significatives. Avant de confirmer le choix de ce modèle, regardons les valeurs du critère BIC (critère explicatif) pour chacun des deux modèles.

Table 6: BIC

Petit modèle	Grand modèle
51.4692	80.87911

En accord avec les résultats précédents, le critère BIC est inférieur pour le petit modèle. Cela confirme notre choix de garder uniquement les variables `WatrCont`, `SubsDens` et `Topo.`

Nous pouvons comparer ce modèle obtenu avec celui déterminé par une sélection automatique.

- **Méthode de sélection automatique**

Nous effectuons une sélection stepwise descendante, toujours avec le critère BIC (option `k=log(n)` dans la fonction `step`). Celle-ci part du plus gros modèle et tente d'enlever les variables les moins significtives afin de retrouver le meilleur modèle.

```
## Modèle retenu par la fonction step :
```

```
##
## Call:  glm(formula = pa ~ WatrCont + SubsDens + Topo, family = "binomial",
##     data = mites)
##
## Coefficients:
## (Intercept)     WatrCont     SubsDens  TopoHummock
##    -0.58289     -0.02162      0.17295      2.73776
##
## Degrees of Freedom: 69 Total (i.e. Null);  66 Residual
## Null Deviance:       91.25
## Residual Deviance: 34.48     AIC: 42.48
```

La sélection automatique retient également le modèle contenant les variables `WatrCont`, `SubsDens` et `Topo` car c'est celui pour lequel le BIC est le plus faible.

Par conséquent, nous retenons le modèle suivant pour la suite :

$$logit(\hat{\pi}) = -0.583 - 0.022 * WatrCont + 0.173 * SubsDens + 2.738 * TopoHummock \quad (1)$$

1.3.2 Interprétation des coefficients

La sortie de notre modèle indique que le contenu en eau, la densité du substrat et la topographie sont associés significativement à l'occurrence de mites. Cependant, on peut également interpréter les coefficients de la pente grâce aux odds-ratio. L'odd-ratio est défini comme le rapport des probabilités d'apparition de l'évènement $Y = 1$ contre $Y = 0$. L'odd est la quantité $\frac{\pi_i}{1-\pi_i}$. Dans le modèle logistique, on définit plus précisément l'odd par :

$$\frac{\pi_i}{1-\pi_i} = \exp(x_i^T \beta) = \exp(\beta_0 + \beta_1 x_1^i + ... + \beta_p x_i^p)$$

et si on considère deux individus i_1 et i_2 dont la valeur des covariables ne diffère que de 1 pour la j-ième covariable, soit $x_{i_1}^j - x_{i_2}^j = 1$, on calcule l'odd-ratio avec :

$$\frac{\pi_{i_1}}{1-\pi_{i_1}} / \frac{\pi_{i_2}}{1-\pi_{i_2}} = \exp(\beta_j)$$

On dira alors qu'une augmentation de 1 de la variable j entraîne une multiplication de l'odds-ratio de $\exp(\beta_j)$.

Table 7: Odds-ratios du modèle expliquant pa

	Odds-ratios
Intercept	0.5582817
WatrCont	0.9786086
SubsDens	1.1888083
TopoHummock	15.4523718

Grâce à ces coefficients, nous pouvons déterminer le pourcentage de probabilité de la présence de mites, lorsque que l'on augmente l'une des variables. En effet, pour chaque augmentation d'une unité du contenu en eau (`WatrCont`), nous avons 2.2% ($0.978 * 100 - 100$) de risque en moins d'avoir une présence de mites, tandis ce que pour une unité de densité du substrat (`SubsDens`) en plus, le risque est plus élevé de 18%. Enfin, lorsque de la topographie est de type Hummock, il y a énormément de probabilité d'avoir des mites, pusique celle-ci augmente de plus de 1400%.

1.3.3 Validation du modèle

Afin de valider notre modèle, nous allons à présent évaluer son ajustement, ainsi que ses pouvoirs explicatifs et de classfication.

- **Evaluation de l'ajustement du modèle**
  - Effet levier

  Dans un premier temps, nous pouvons nous intéresser aux points leviers, c'est à dire les points qui influencent fortement sur leur propre estimation.

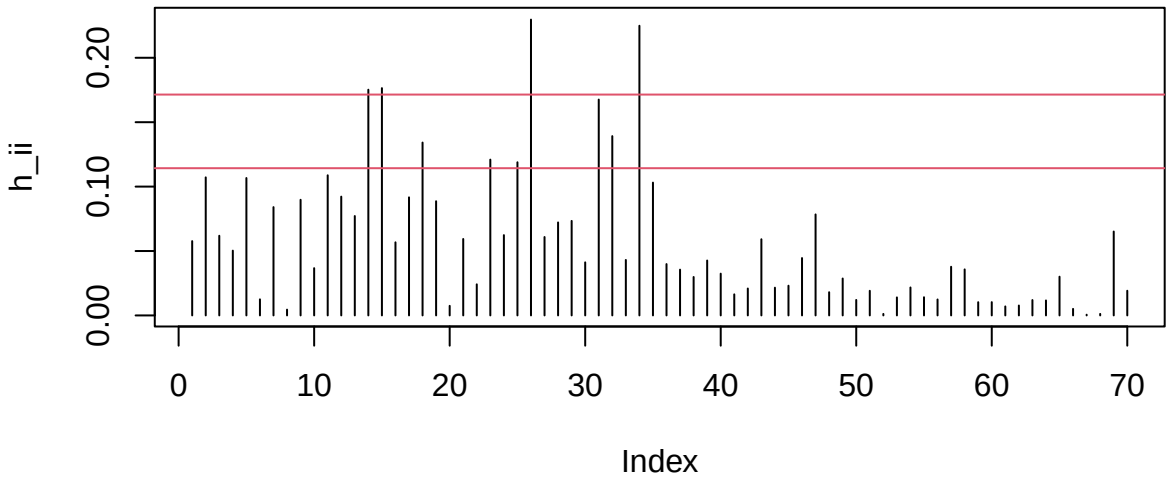


Figure 23: Points leviers du modèle expliquant pa

On remarque sur le graphique précédent que quatre observations peuvent être considérées comme points leviers. Une connaissance plus approfondie du jeu de données nous permettrait de comprendre plus en détails ces résultats.

– Points influents

Regardons ensuite les points influents sur le modèle. Ce sont des points importants car leur présence joue beaucoup sur l'estimation des coefficients. Pour les déterminer, on peut représenter leur distance de Cook, c'est à dire la distance entre le vecteur des coefficients estimés avec toutes les observations et celui estimé avec toutes les observations sauf une.

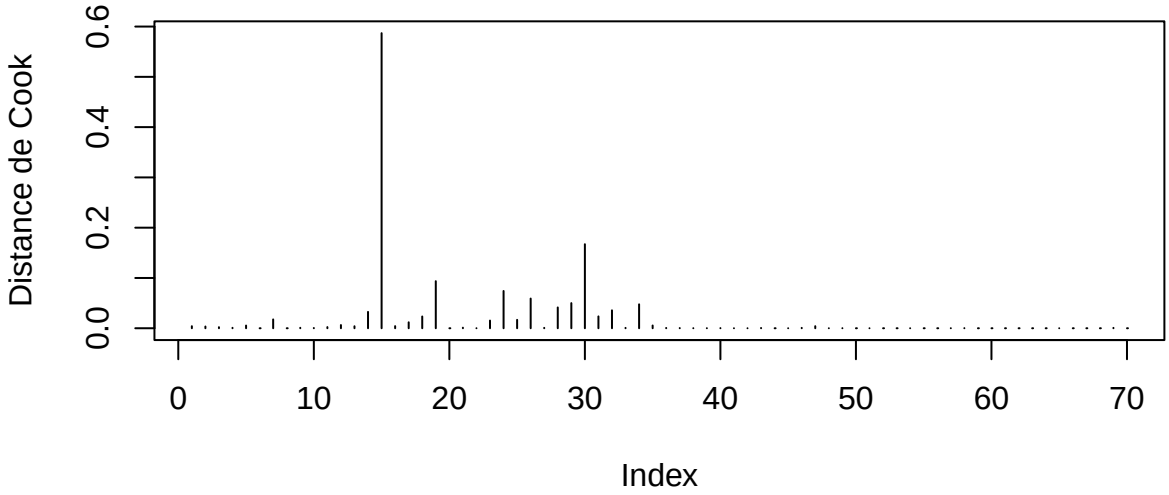


Figure 24: Points influents du modèle expliquant pa

Sur le graphique obtenu, on remarque notamment l'observation 15 qui a une distance de Cook nettement supérieure aux autres. Cela signifie que c'est un point influent. Là encore, il serait necessaire de connaître plus en détails notre jeu de données pour pouvoir l'expliquer et notammant déterminer si c'est un point levier, un point abberrant ou les deux. Nous pouvons tout de même remarquer qu'il figurait déja dans les points à effet levier relevés précédemment.

– Analyse des résidus

Afin de tester la qualité de l'ajustement de notre modèle, on peut utiliser le test de Pearson basé sur les résidus de Pearson. Ce test a pour hypothèse nulle H_0 : le modèle ajuste correctement les données. On définit les résidus de Pearson comme $\hat{r}_{P,i} = \frac{y_i - \hat{y}_i}{\sqrt{Var(\hat{y}_i)}}$ et sous H_0, on a $\sum_{i=1}^{n} \hat{r}_{P,i}^2 \sim \chi^2(n - (p+1))$.

```
## p-value pour le test de Pearson :  0.9835327
```

La p-value étant très grande, on admet que le modèle est très bien adapté aux données. On peut tout de même visualiser ces résidus de Pearson.

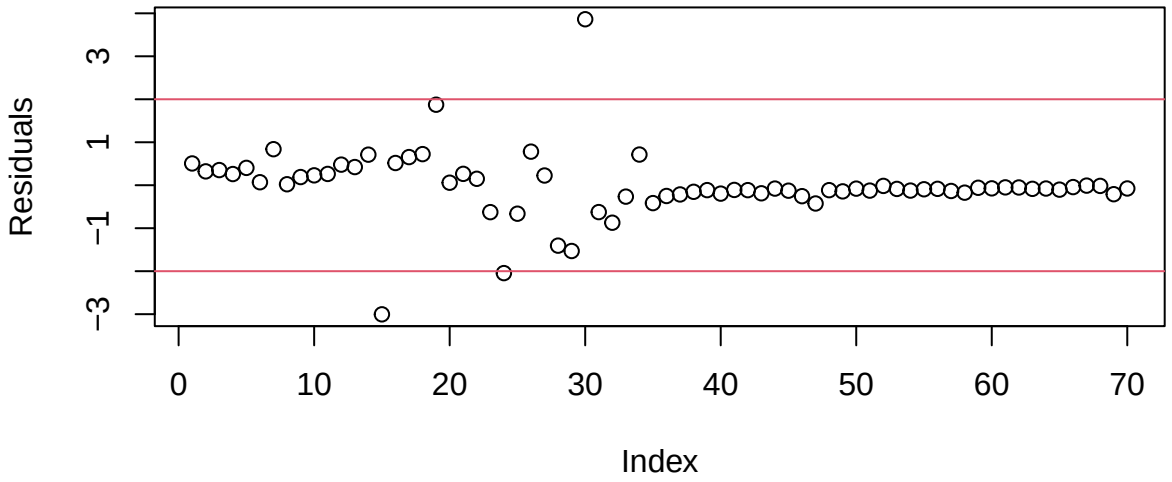


Figure 25: Résidus de Pearson du modèle expliquant pa

Nous remarquons deux individus ayant des résidus élevés. Une nouvelle fois, nous retrouvons l'individu 15. Il serait peut-être judicieux de l'enlever pour la suite. Cependant, le jeu de données n'étant pas très grand, nous décidons de le conserver afin de ne pas le réduire davantage.

– Test d'Hosmer-Lemeshow

Par ailleurs, on peut également effectuer un test d'Hosmer-Lemeshow permettant d'évaluer la pertinence d'un modèle de régression logistique pour lequel on a des données individuelles (pour plus d'informations voir (Hosmer and Lemeshow 2000))

```
## p-value pour le test de Hosmer-Lemeshow :  0.9529158
```

De même, la très grande p-value nous indique que le modèle est bien adapté aux données.

Avant de valider définitivement le choix de ce modèle, il faut évaluer son pouvoir explicatif ainsi que son pouvoir de classification.

- Evaluation du pouvoir explicatif du modèle

Afin d'évaluer le pouvoir explicatif de notre modèle, on peut calculer son Pseudo-R^2. Il en existe plusieurs. Ici, nous allons déterminer celui de McFadden (1973) correspondant au quotient entre la différence des déviances nulles et résiduelles et la déviance nulle.

```
## pseudo-R2 :  0.6221724
```

D'après ce résultat, on en conclut que notre modèle explique plus de 62.2% des données. Regardons ensuite s'il est capable de bien classer ces données.

- Evaluation du pouvoir de classification du modèle

Pour évaluer le pouvoir de classification de notre modèle, nous allons juger ses prédictions grâce à une courbe ROC. Afin de ne pas prédire sur des individus ayant servis à la construction du modèle, nous effectuons une validation croisée. En effet, pour chaque individu i, nous estimons le modèle sans cet individu puis nous

prédisons la valeur de `pa` pour celui-ci. Nous obtenons alors les prédictions de `pa` pour tous les individus et nous pouvons tracer la courbe ROC.

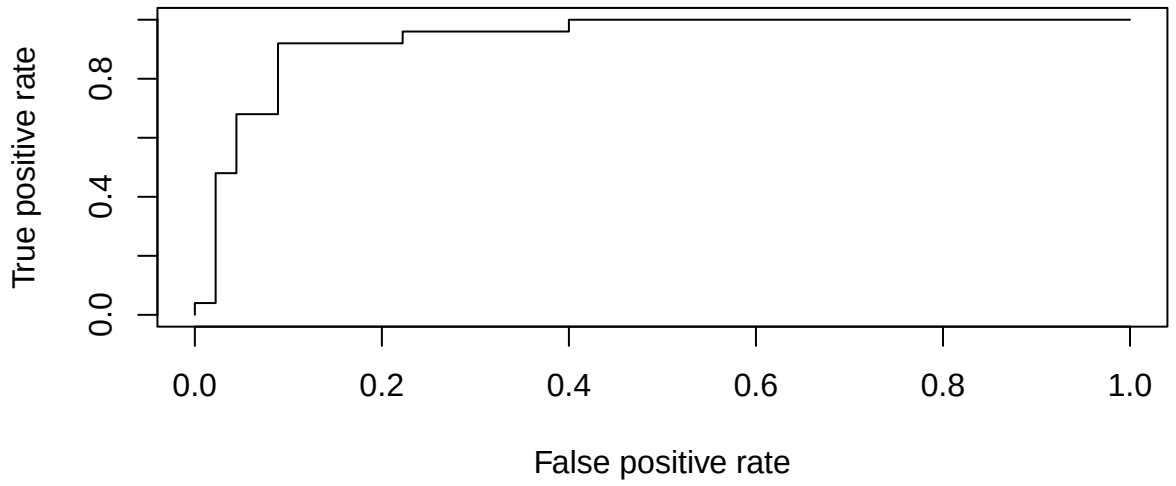


Figure 26: Courbe ROC

Cette courbe résume le taux de vrais positifs en fonction du taux de faux positifs. Elle est d'autant meilleure qu'elle s'éloigne de la diagonale. Ici, nous avons une courbe largement au dessus de la diagonale. Nous pouvons d'ailleurs calculer son aire sous la courbe (AUC).

```
## AUC :  0.9351111
```

L'aire sous la courbe est proche de 1 : le pouvoir prédictif de notre modèle est alors très bon.

Enfin, nous pouvons résumer les bonnes et mauvaises prédictions de notre modèle grâce à une matrice de confusion. Pour cela, il est intéressant de déterminer en amont le seuil à partir duquel on admet la présence de mites. On trace alors la courbe du taux d'erreur en fonction des différents seuils.

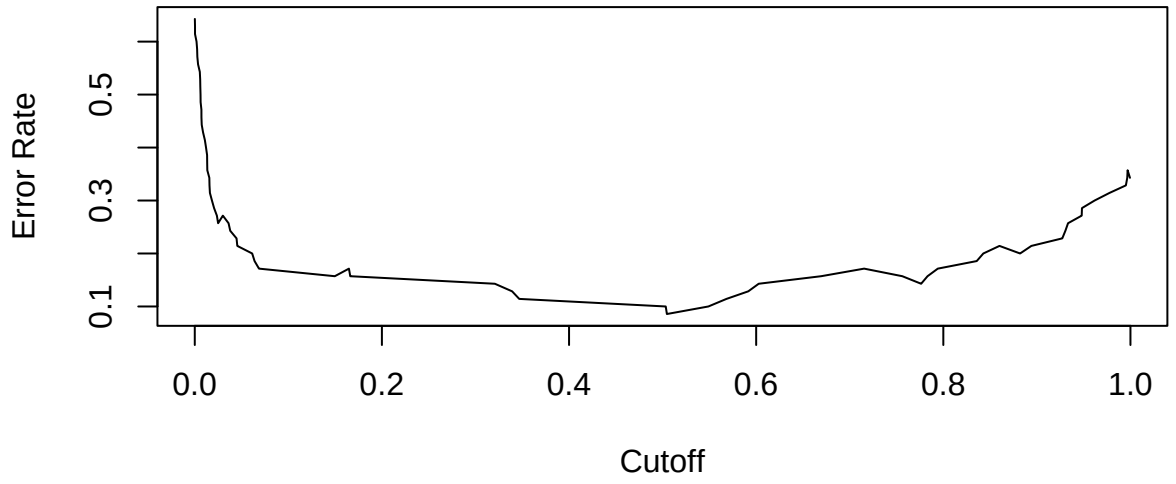


Figure 27: Taux d'erreur en fonction du seuil

Nous remarquons que le taux d'erreur le plus faible est pour un seuil de 0.5. C'est alors le seuil que nous allons choisir pour étblir notre matrice de confusion.

```
##
```

```
##          0  1
##   FALSE 40  2
##   TRUE   5 23
```

D'après cette matrice, nous avons prédit quarante fois l'abscence de mites et vingt-trois fois leur présence en ayant raison. Cependant, le modèle s'est trompé sept fois : cinq fois en ayant prédit l'abscence de mites à tort et deux fois leur présence à tort. Par conséquent, le taux d'erreur est de $\frac{7}{70} = 0.1$. On peut en conclure que ce modèle a de bonnes qualités de classification.

Conclusion : Nous en déduisons que le modèle (1) de régression logistique expliquant l'occurrence de mites grâce au contenu en eau, à la densité du substrat et à la topographie obtient de bons résultats, tant sur son ajustement que sur son pouvoir de classification.

1.4 GLM Régression Logistique sur les données agrégées (prop)

Objectif : expliquer la fréquence relative `prop`, déterminer le meilleur modèle pour cette variable réponse, interpréter les coefficients, puis évaluer l'ajustement de ce modèle.

La variable `prop` est, comme son nom l'indique, une variable en proportion. Malgré qu'il ne s'agisse pas d'une variable binaire, ce cas est proche d'une régression logistique. En effet, cette fois-ci, au lieu de prédire une valeur binaire échec ou succès (0 ou 1), on veut prédire la proportion de succès $P_i = Y_i/n_i$. Dans notre cas, la valeur n_i est la variable `totalabund` qui représente le nombre total de mites, et la variable Y_i est le nombre de succès, ici le nombre de mites qui sont des Galumna (représenté par la variable `Galumna` elle-même). En clair, nous avons : `prop` = `Galumna/totalabund`. On a $\mathbb{E}(Y_i) = n_i\pi_i$ et donc $\mathbb{E}(P_i) = \pi_i$. Et on modélise les probabilités π_i par $g(\pi_i) = x_i^T\beta$ où x_i est le vecteur des variables explicatives, β est un vecteur de paramètres et g est la fonction de lien. La fonction de lien pour la régression logistique est la fonction *logit*.

Sous R, on peut utiliser la fonction `glm` de manière semblable à lorsque l'on fait une régression logistique classique, mais en précisant cette fois des poids a priori.

1.4.1 Recherche du meilleur modèle

On suivra la même méthode que dans la partie précédente pour chercher le meilleur modèle : une recherche à la main dans un premier temps, puis une méthode de sélection automatique.

- **Méthode "à la main"**

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## Call:
## glm(formula = prop ~ WatrCont + SubsDens + Topo + Shrub + Substrate,
##     family = "binomial", data = mites, weights = totalabund)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4587  -0.9228  -0.1041   0.0000   4.1062
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -2.028e+01  6.816e+03  -0.003 0.997626
## WatrCont           -5.215e-03  1.508e-03  -3.458 0.000544 ***
## SubsDens            1.962e-02  1.133e-02   1.732 0.083321 .
## TopoHummock         4.437e-01  3.585e-01   1.238 0.215847
## ShrubMany           2.483e-01  3.029e-01   0.820 0.412226
## ShrubNone          -1.780e+01  2.340e+03  -0.008 0.993931
## SubstrateInterface  1.598e+01  6.816e+03   0.002 0.998129
## SubstrateLitter     1.646e+01  6.816e+03   0.002 0.998073
## SubstrateSphagn1    1.616e+01  6.816e+03   0.002 0.998108
## SubstrateSphagn2    1.488e+01  6.816e+03   0.002 0.998258
## SubstrateSphagn3    1.577e+01  6.816e+03   0.002 0.998154
## SubstrateSphagn4    1.627e+01  6.816e+03   0.002 0.998095
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 140.70  on 69  degrees of freedom
## Residual deviance:  67.73  on 58  degrees of freedom
## AIC: 158.48
```

```
##
## Number of Fisher Scoring iterations: 19
```

On voit qu'il n'y que très peu de variables significatives. On créé un modèle avec les deux variables les plus significatives du modèle complet : `WatrCont` et `SubsDens`.

```
##
## Call:
## glm(formula = prop ~ WatrCont + SubsDens, family = "binomial",
##     data = mites, weights = totalabund)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6469  -1.0054  -0.6332  -0.1536   4.6133
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.400771   0.430215  -7.905 2.68e-15 ***
## WatrCont    -0.007220   0.001099  -6.568 5.10e-11 ***
## SubsDens     0.022387   0.009185   2.437   0.0148 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 140.702  on 69  degrees of freedom
## Residual deviance:  85.387  on 67  degrees of freedom
## AIC: 158.14
##
## Number of Fisher Scoring iterations: 5
```

Cette fois-ci on voit que toutes les variables, ainsi que la constante sont significatives pour le test de Wald. Pour rappel, ce test permet de tester la nullité d'un coefficient : $H_0 : \beta_j = 0$.

- **Méthode de sélection automatique**

On effectue maintenant une recherche automatique du meilleur modèle en partant du plus gros modèle et en se basant sur le critère BIC, critère explicatif.

```
## [1] "Modèle retenu par la fonction step: "
```

```
##
## Call:  glm(formula = prop ~ WatrCont + SubsDens + Topo, family = "binomial",
##     data = mites, weights = totalabund)
##
## Coefficients:
## (Intercept)     WatrCont     SubsDens  TopoHummock
##   -4.229742    -0.006823     0.029161     0.728616
##
## Degrees of Freedom: 69 Total (i.e. Null);  66 Residual
## Null Deviance:       140.7
## Residual Deviance: 78.18     AIC: 152.9
```

Avec une procédure de sélection backward, on garde 3 variables : `WatrCont`, `SubsDens` et `Topo`.

Pour cette étude, on se place dans un cadre plus explicatif que prédictif. On décide donc de comparer ces trois modèles avec le BIC :

Table 8: BIC des différents modèles

Modèle complet	Modèle 2 variables	Modèle 3 variables
185.4658	164.8866	161.9289

Le modèle à trois variables est celui qui minimise le BIC. On peut tout de même effectuer un test de modèles emboîtés afin de regarder si la troisième variable `Topo` est bien significative. On teste donc le modèle à 2 variables contre celui à 3 variables.

```
## Analysis of Deviance Table
##
## Model 1: prop ~ WatrCont + SubsDens
## Model 2: prop ~ WatrCont + SubsDens + Topo
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1        67     85.387
## 2        66     78.181  1   7.2063 0.007265 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La p-value est inférieure à 0.05 donc la variable `Topo` est significative.

On effectue un summary du modèle à 3 variables.

```
##
## Call:
## glm(formula = prop ~ WatrCont + SubsDens + Topo, family = "binomial",
##     data = mites, weights = totalabund)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7781  -0.8947  -0.5749  -0.1380   4.1291
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.229742   0.542002  -7.804 6.00e-15 ***
## WatrCont    -0.006823   0.001157  -5.896 3.73e-09 ***
## SubsDens     0.029161   0.009880   2.951  0.00316 **
## TopoHummock  0.728616   0.280072   2.602  0.00928 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 140.702  on 69  degrees of freedom
## Residual deviance:  78.181  on 66  degrees of freedom
## AIC: 152.93
##
## Number of Fisher Scoring iterations: 5
```

On remarque que toutes les variables sont significatives d'après le test de Wald.

On retient finalement le modèle suivant :

$$logit(\hat{\pi}) = -4.23 - 0.006 * WatrCont + 0.029 * SubsDens + 0.729 * TopoHummock \quad (2)$$

1.4.2 Interprétation des coefficients

On peut interpréter les coefficients du modèle pour nos trois variables grâce à leurs odds-ratios. L'odd-ratio pour la variable j est égal à $\exp(\beta_j)$.

Table 9: Odds-ratios du modèle expliquant prop

	Odds-ratios
Intercept	0.0145561
WatrCont	0.9932006
SubsDens	1.0295906
TopoHummock	2.0722106

L'odd-ratio pour la variable `WatrCont` est inférieur à 1. De plus, sa p-value associée au test de Wald est inférieure à 5% donc on peut dire que la proportion de mites est significativement moins élevée si la quantité d'eau dans le sol augmente. L'odd-ratio de `SubsDens` est quant à lui supérieur à 1, et sa p-value pour le test de Wald inférieure à 5%. Cela signifie que la proportion de mites sera significativement plus importante si la densité du substrat augmente.
Enfin, l'odd-ratio pour `Topo` vaut 2.07 et sa p-value est inférieure à 5%. Donc on peut conclure que si la topographie est de type Hummock, la proportion de mites sera significativement plus élevée.

1.4.3 Validation du modèle

Enfin, nous devons maintenant regarder si notre modèle est "bon". Nous allons, pour cela, évaluer sa qualité d'ajustement et son pouvoir explicatif.

- **Evaluation de l'ajustement du modèle**
  - Effet levier

  On regarde dans un premier temps les points leviers (qui influencent fortement leur estimation).

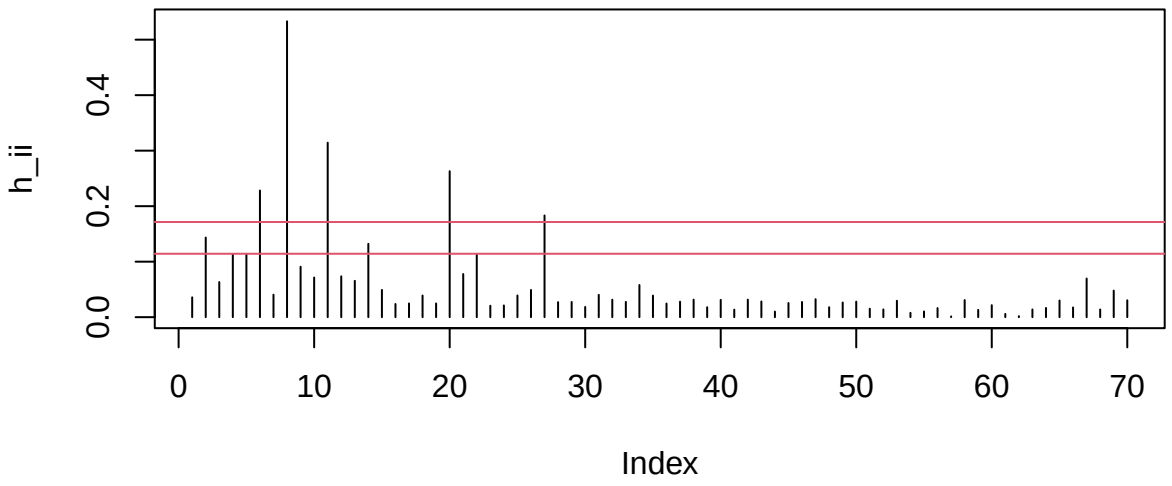


Figure 28: Points leviers du modèle expliquant prop

  On voit que 5 observations peuvent être déclarées comme "points leviers".

  - Points influents

  Pour les observer, on représente leur distance de Cook.

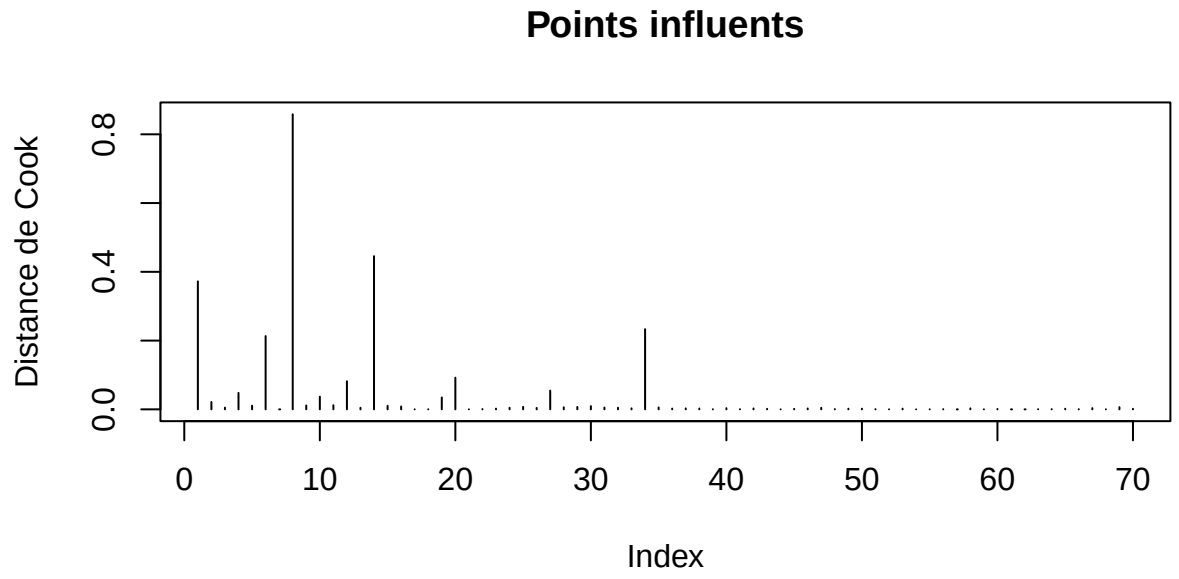


Figure 29: Points influents du modèle expliquant prop

On relève plusieurs points influents. Notamment la 8ème observation, qui était déjà un point levier. Généralement, si un point est influent, il est soit levier, soit aberrant, soit les deux.

– Analyse des résidus

Finalement, nous allons analyser les résidus. On effectue pour cela un test de Pearson ou test global de la qualité de l'ajustement basé sur les résidus de Pearson.

```
## p-value pour le test de Pearson :  0.004671346
```

La p-value est très petite. Cela signifie que d'après le test de Pearson, notre modèle n'ajuste pas très bien les données. Cependant, nous ne voyons pas comment le changer afin qu'il soit meilleur à ce niveau là. Nous continuons avec ce modèle, tout en gardant à l'esprit qu'il présente des défauts d'ajustement.

On finit par représenter les résidus.

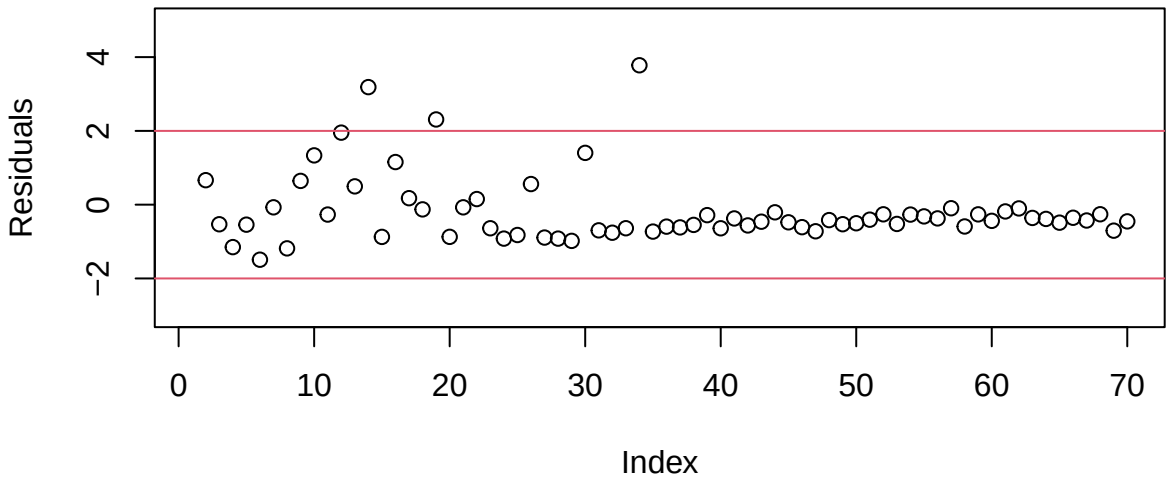


Figure 30: Résidus de Pearson du modèle expliquant prop

On voit que les résidus sont répartis de façon homogène autour de l'axe des abscisses. Ils sont pour la

plupart proches de 0. Cependant, trois points semblent avoir une valeur de résidu élevée. Peut-être correspondent-ils à des points leviers ou aberrants observés précédemment. Nous pourrions les enlver du jeu de données afin de construire un modèle qui ajusterait encore mieux les données mais nous décidons de les garder car le jeu de données ne contient pas beaucoup d'individus (70) et nous ne voulons pas le réduire davantage. De plus, nous n'avons pas une connaissance très poussée de nos données donc nous ne pourrions pas justifier le retrait de ces individus pour notre étude.

- **Evaluation du pouvoir explicatif du modèle**

Pour finir la validation du modèle, on calcule le pseudo-R^2, qui nous donne la variance expliquée par notre modèle :

```
## pseudo-R2 :  0.4443505
```

Notre modèle explique 44% de la variance. Ce score n'est pas très élevé. Cependant, en régression logistique, les valeurs faibles de pseudo-R^2 sont souvent courantes.

Conclusion : Nous en déduisons que le modèle (2) de régression logistique expliquant la proportion de mites Galumna grâce au contenu en eau, à la densité du substrat et à la topographie obtient de bons résultats.

1.5 GLM Régression Poisson sur la variable d'abondance (Galumna)

Objectif : modéliser l'abondance de l'espèce Galumna en fonction des caractéristiques du substrat (son contenu en eau `WatrCont` et sa densité `SubsDens`) et, si nécessaire, des autres variables environnementales.

Pour ce faire, nous allons utiliser un modèle linéaire généralisé avec une distribution adaptée à notre problème de comptage : la distribution de Poisson.

Pour une distribution Poisson $Y \sim Pois(\lambda)$ avec le lien *log* par défaut nous avons $log(Y) = \eta(.)$ avec l'inverse de ce lien $\lambda = e^{\eta(.)}$, où $\eta(.)$ représente le prédicteur linéaire.

Comme pour la régression logistique, la régression de Poisson utilise la fonction `glm`. Il faut spécifier la famille "poisson" et (optionnellement) le lien *log*. En effet, le logarithme est la foncton de lien par défaut pour la régression de Poisson sur R.

1.5.1 Recherche du meilleur modèle

- **Méthode "à la main"**

Nous allons commencer par ajuster un modèle complet et ensuite effectuer une sélection de variables.

```
## Warning: glm.fit: fitted rates numerically 0 occurred
```

```
##
## Call:
## glm(formula = Galumna ~ WatrCont + SubsDens + Topo + Shrub +
##     Substrate, family = poisson(link = "log"), data = mites)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7489  -0.9030  -0.2186  -0.0001   3.5092
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -1.520e+01  4.390e+03  -0.003 0.997238
## WatrCont           -5.778e-03  1.525e-03  -3.790 0.000151 ***
## SubsDens            2.429e-02  1.082e-02   2.245 0.024756 *
## TopoHummock         5.901e-01  3.518e-01   1.677 0.093447 .
## ShrubMany           4.092e-02  2.979e-01   0.137 0.890745
## ShrubNone          -1.761e+01  1.945e+03  -0.009 0.992778
## SubstrateInterface  1.589e+01  4.390e+03   0.004 0.997111
## SubstrateLitter     1.652e+01  4.390e+03   0.004 0.996996
## SubstrateSphagn1    1.620e+01  4.390e+03   0.004 0.997056
## SubstrateSphagn2    1.449e+01  4.390e+03   0.003 0.997366
## SubstrateSphagn3    1.617e+01  4.390e+03   0.004 0.997062
## SubstrateSphagn4    1.585e+01  4.390e+03   0.004 0.997119
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 168.254  on 69  degrees of freedom
## Residual deviance:  77.559  on 58  degrees of freedom
## AIC: 168.72
##
## Number of Fisher Scoring iterations: 17
```

Nous observons dans le `summary` de notre modèle complet qu'il n'y a que très peu de variables significatives. On créé un modèle avec les deux variables significatives du modèle complet : `WatrCont` et `SubsDens`.

```
##
## Call:
## glm(formula = Galumna ~ WatrCont + SubsDens, family = poisson(link = "log"),
##     data = mites)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0851  -0.9488  -0.6573  -0.1344   4.5579
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.885689   0.425194   4.435 9.21e-06 ***
## WatrCont    -0.008304   0.001118  -7.426 1.12e-13 ***
## SubsDens     0.023367   0.008380   2.788   0.0053 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 168.25  on 69  degrees of freedom
## Residual deviance: 101.49  on 67  degrees of freedom
## AIC: 174.65
##
## Number of Fisher Scoring iterations: 6
```

On observe que toutes les variables, ainsi que la constante, sont significatives pour le test de Wald.

- **Recherche automatique**

Avant de confirmer ce modèle, nous allons effectuer une recherche automatique du meilleur modèle en partant du plus gros modèle et en se basant sur le critère BIC, critère explicatif.

```
## [1] "Modèle retenu par la fonction step : "
```

```
##
## Call:  glm(formula = Galumna ~ WatrCont + SubsDens + Topo, family = poisson(link = "log"),
##     data = mites)
##
## Coefficients:
## (Intercept)     WatrCont     SubsDens  TopoHummock
##    0.710300    -0.007539     0.033870     0.891830
##
## Degrees of Freedom: 69 Total (i.e. Null);  66 Residual
## Null Deviance:       168.3
## Residual Deviance: 91.7  AIC: 166.9
```

Avec une procédure de sélection backward, on garde 3 variables : `WatrCont`, `SubsDens` et `Topo`.

Nous avons donc un nouveau modèle :

```
##
## Call:
## glm(formula = Galumna ~ WatrCont + SubsDens + Topo, family = poisson(link = "log"),
##     data = mites)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8676  -0.8221  -0.5927  -0.3579   3.9134
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.710300   0.584683   1.215 0.224424
## WatrCont    -0.007539   0.001178  -6.400 1.56e-10 ***
## SubsDens     0.033870   0.009373   3.614 0.000302 ***
## TopoHummock  0.891830   0.295424   3.019 0.002538 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 168.254  on 69  degrees of freedom
## Residual deviance:  91.699  on 66  degrees of freedom
## AIC: 166.86
##
## Number of Fisher Scoring iterations: 6
```

On voit que toutes les variables sont significatives pour le test de Wald.

Pour cette étude, on se place dans un cadre plus explicatif que prédictif. On décide donc de comparer ces trois modèles avec le BIC :

Table 10: BIC des différents modèles

Modèle complet	Modèle à 2 variables	Modèle à 3 variables
195.7031	181.3957	175.855

Le modèle à trois variables est celui qui minimise le BIC. On peut tout de même effectuer un test de modèles emboîtés afin de regarder si la troisième variable `Topo` est bien significative. On teste donc le modèle à 2 variables contre celui à 3 variables.

```
## Analysis of Deviance Table
##
## Model 1: Galumna ~ WatrCont + SubsDens + Topo
## Model 2: Galumna ~ WatrCont + SubsDens
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1        66     91.699
## 2        67    101.488 -1  -9.7891 0.001755 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La p-value est inférieure à 0.05 donc la variable `Topo` est significative.

Le modele sélectionné s'écrit donc :

$$log(Y) = 0.710 - 0.008 * WatrCont + 0.034 * SubsDens + 0.892 * TopoHummock \quad (3)$$

1.5.2 Interprétation des coefficients

Nous allons maintenant interpréter les coefficients de notre modèle. Nous allons determiner l'effet de chaque coefficient du modèle sur le prédicteur linéaire $\eta = \beta_0 + \beta_1 * WatrCont + \beta_2 * SubsDens + \beta_3 * TopoHummock$.

- Le coefficient `WatrCont` qui indique le contenu en eau, indique que η diminue de 0.008 pour chaque augmentation d'une unité de `WatrCont` si la variable `SubsDens` reste constante. Nous avons donc le facteur $e^{-0.008} = 0.99$ qui correspond à une perte de 1% de l'abondance de Galumna par unité de contenu en eau (`WatrCont`) supplémentaire.
- Le coefficient `SubsDens` qui indique la densité, a un changement additif de 0.023 qui correspond au facteur $e^{0.034} = 1.034585$. On en conclut que la densité n'influe pas sur l'abondance de Galumna.
- Le coefficient `TopoHummock` qui représente la topographie de type Hummock, a un changement additif de 0.892 qui correspond au facteur $e^{0.892} = 2.440005$. Cela correspond a une augmentation de 250% de l'abondance de Galumna pour une topographie de type Hummock.

1.5.3 Validation du modèle

Enfin, nous passons à la phase de validation du modèle. Nous allons, pour cela, regarder s'il y a de la surdispersion dans ce modèle, puis évaluer sa qualité d'ajustement et son pouvoir explicatif.

- **Analyse de la surdispersion**

Pour une distribution de Poisson, $Var(Y) = \mu = E(Y)$. En pratique, on constate que la variance des données dépasse souvent μ, indiquant une surdispersion dans les paramètres du modèle.
Par définition, lorsque la déviance résiduelle est supérieure au nombre de degrés résiduels, le modèle est surdispersé. On peut estimer un paramètre de surdispersion

$$\phi = \frac{\text{Déviance résiduelle}}{\text{Degré de liberté résiduels}}$$

Nous avons ici :

```
cat("phi = ", glm.poi3$deviance/glm.poi3$df.residual)
```

```
## phi =  1.38938
```

Cette valeur est bien supérieure à 1 : il y a de la surdispersion dans le modèle. Pour palier cela, nous pouvons utiliser la famille `quasipoisson` dans notre modèle. Corriger pour la surdispersion ne va pas affecter l'estimation des paramètres, mais leur significativité. En fait, les écarts-types des paramètres seront multipliés par $\sqrt{\phi}$. Autrement dit, la surdispersion n'introduit pas de biais, mais augmente l'incertitude sur les valeurs des coefficients.

```
##
## Call:
## glm(formula = Galumna ~ WatrCont + SubsDens + Topo, family = quasipoisson,
##     data = mites)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8676  -0.8221  -0.5927  -0.3579   3.9134
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.710300   0.808202   0.879   0.3827
## WatrCont    -0.007539   0.001628  -4.630 1.77e-05 ***
## SubsDens     0.033870   0.012956   2.614   0.0111 *
## TopoHummock  0.891830   0.408361   2.184   0.0325 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 1.910725)
##
##     Null deviance: 168.254  on 69  degrees of freedom
```

```
## Residual deviance:  91.699  on 66  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 6
```

On voit que les coefficients ne changent pas, contrairement aux erreurs-types.

- **Évaluation de l'ajustement du modèle**
  - Effet levier

  On regarde une nouvelle fois les points leviers.

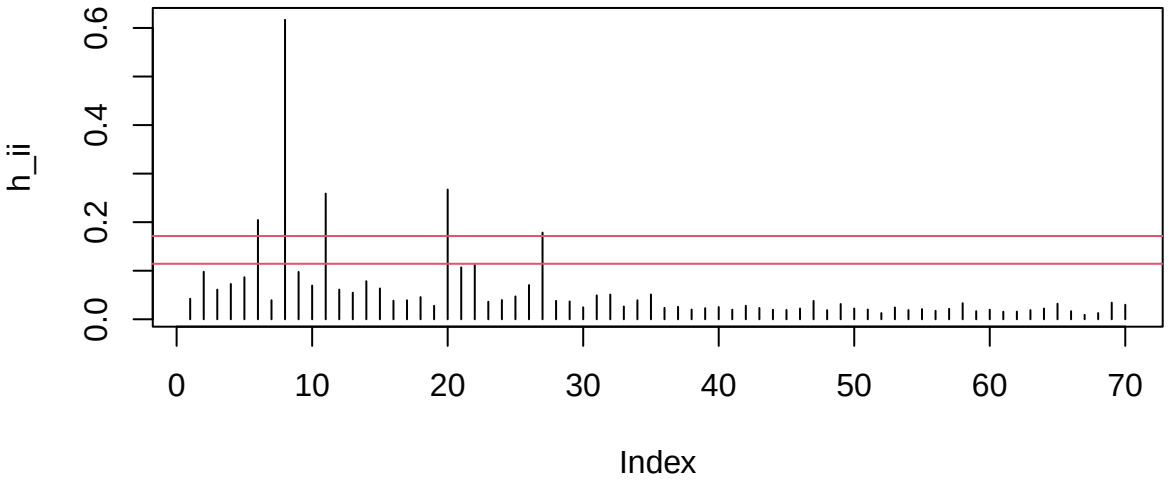


Figure 31: Points leviers du modèle expliquant Galumna

Cinq observations peuvent être considérées comme points leviers. Une connaissance plus approfondie du jeu de données nous permettrait de comprendre plus en détail ces résultats.

  - Points influents

Regardons ensuite les points influents sur le modèle.

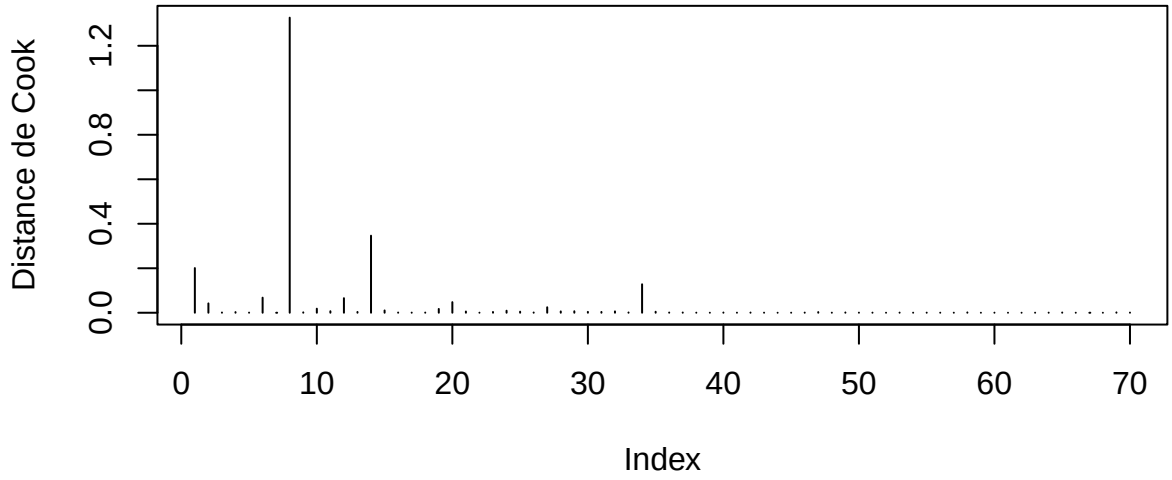


Figure 32: Points influents du modèle expliquant Galumna

Sur le graphique obtenu, on remarque notamment l'observation 8 qui a une distance de Cook nettement supérieure aux autres. Cela signifie que c'est un point influent. Là encore, il serait necessaire de connaître plus en détails notre jeu de données pour pouvoir l'expliquer et notammant déterminer si c'est un point levier, un point abberrant ou les deux.

– Analyse des résidus

Pour tester la qualité de l'ajustement du modèle, on effectue un test de Pearson.

```
## p-value pour le test de Pearson :  1.20652e-05
```

La p-value est très petite. Cela signifie que d'après le test de Pearson, notre modèle n'ajuste pas très bien les données. Cependant, nous ne voyons pas comment le changer afin qu'il soit meilleur à ce niveau là. Nous continuons avec ce modèle, tout en gardant à l'esprit qu'il présente des défauts d'ajustement.

- **Evaluation du pouvoir explicatif du modèle**

Calculons maintenant le Pseudo-R^2 du modèle.

```
## pseudo-R2 :  0.4549952
```

D'après ce résultat, on en conclut que notre modèle explique plus de 45% des données.

Ainsi, nous en déduisons que le modèle (3) de régression quasi-Poisson expliquant l'abondance de Galumna grâce au contenu en eau, à la densité du substrat et à la topographie est un bon modèle explicatif pour nos données.

Conclusion :
Le but initial de notre étude était de décrire le jeu de données sur les mites Oribatid. Pour cela, nous avons commencé par faire de l'analyse descriptive des variables, individuellement, puis collectivement. Ensuite, nous avons tenté d'ajuster un modèle linéaire pour expliquer trois variables du jeu de données. Nous avons été confrontées à un problème : aucun modèle, même après transformation des données, ne satisfaisait toutes les hypothèses du modèle linéaire gaussien. Nous avons donc choisi d'autres distributions (non gaussiennes), mieux adaptées à nos données, et ajusté un modèle linéaire généralisé pour chacune d'entre elles. Finalement, nous obtenons trois modèles qui nous permettent d'expliquer nos trois variables. Nous jugeons nos modèles plutôt bons, bien que leur qualité globale puisse être discutée.

2 Données manquantes

2.1 Scénarios NA sur un jeu de données simulées

2.1.1 Simulations

Dans le but d'étudier différents scénarios d'imputation de données manquantes, nous commençons par simuler $n = 100$ réalisations d'un vecteur gaussien (X, Y) de moyenne $\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ et de variance $\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$. Pour cela, on utilise la fonction `rmvnorm` du package `mvtnorm`.

Ensuite, nous ajoutons des données manquantes sur Y selon plusieurs mécanismes, afin de les comparer par la suite.

- Mécanisme **MCAR** (**M**issing **C**ompletely **A**t **R**andom)

Le premier mécanisme que nous utilisons est appelé MCAR. Pour celui-ci, les données sont manquantes de façon totalement aléatoire, c'est-à-dire que la probabilité d'abscence est la même pour toutes les observations. Dans notre cas, on choisit $P(M = 0) = 0.35$.

Voici les informations concernant le jeu de données obtenu :

```
##        X                   Y
##  Min.   :-2.73811   Min.   :-1.90801
##  1st Qu.:-0.78230   1st Qu.:-0.75489
##  Median : 0.01703   Median : 0.01495
##  Mean   :-0.08483   Mean   :-0.07061
##  3rd Qu.: 0.58113   3rd Qu.: 0.80906
##  Max.   : 2.20992   Max.   : 2.03330
##                     NA's   :34
```

- Mécanisme **MAR** (**M**issing **A**t **R**andom)

Ensuite, nous utilisons le mécanisme MAR. Contrairement au précédent, la probabilité d'abscence est liée à une ou plusieurs autres variables observées. Elle ne dépend cependant pas des valeurs manquantes. Ici, on choisit $P(M = 0|X) = \mathcal{B}(\phi(1.2X - 0.5))$ où $\phi(.)$ désigne la fonction de répartition d'une loi normale $\mathcal{N}(0, 1)$.

On obtient un jeu de données ayant les caractéristiques suivantes pour chacune de ses variables :

```
##        X                   Y
##  Min.   :-2.73811   Min.   :-2.2075
##  1st Qu.:-0.78230   1st Qu.:-0.9604
##  Median : 0.01703   Median :-0.2701
##  Mean   :-0.08483   Mean   :-0.2493
##  3rd Qu.: 0.58113   3rd Qu.: 0.5985
##  Max.   : 2.20992   Max.   : 1.6661
##                     NA's   :36
```

- Mécanisme **MNAR** (**M**issing **N**ot **A**t **R**andom)

Enfin, nous utilisons le mécanisme MNAR. Cette fois-ci, les données sont manquantes de façon non aléatoire, c'est-à-dire qu'elles dépendent des variables en question. En effet, la probabilité qu'une valeur soit manquante dépend d'une ou plusieurs données non observées pour les autres variables, ici X. Dans notre cas, on a $P(M = 0|Y) = \mathcal{B}(\phi(1.2X - 0.5))$, où $\phi(.)$ désigne toujours la fonction de répartition d'une loi normale $\mathcal{N}(0, 1)$.

Le jeu de données obtenu contient toujours deux variables vérifiant les informations suivantes :

```
##        X                   Y
##  Min.   :-2.73811   Min.   :-2.20751
##  1st Qu.:-0.78230   1st Qu.:-1.19347
##  Median : 0.01703   Median :-0.69551
```

```
##  Mean   :-0.08483   Mean   :-0.61411
##  3rd Qu.: 0.58113   3rd Qu.: 0.04182
##  Max.   : 2.20992   Max.   : 0.95414
##                     NA's   :39
```

Notons que ce dernier type de données manquantes est plus complexe à traiter par la suite.

2.1.2 Analyse

Maintenant que nous avons ajouté de trois manières différentes des valeurs manquantes sur la variable Y, nous allons analyser les jeux de données obtenus. Pour cela, nous commençons par les visualiser en entier. Nous utilisons la fonction `vis_miss` du package `visdat`.

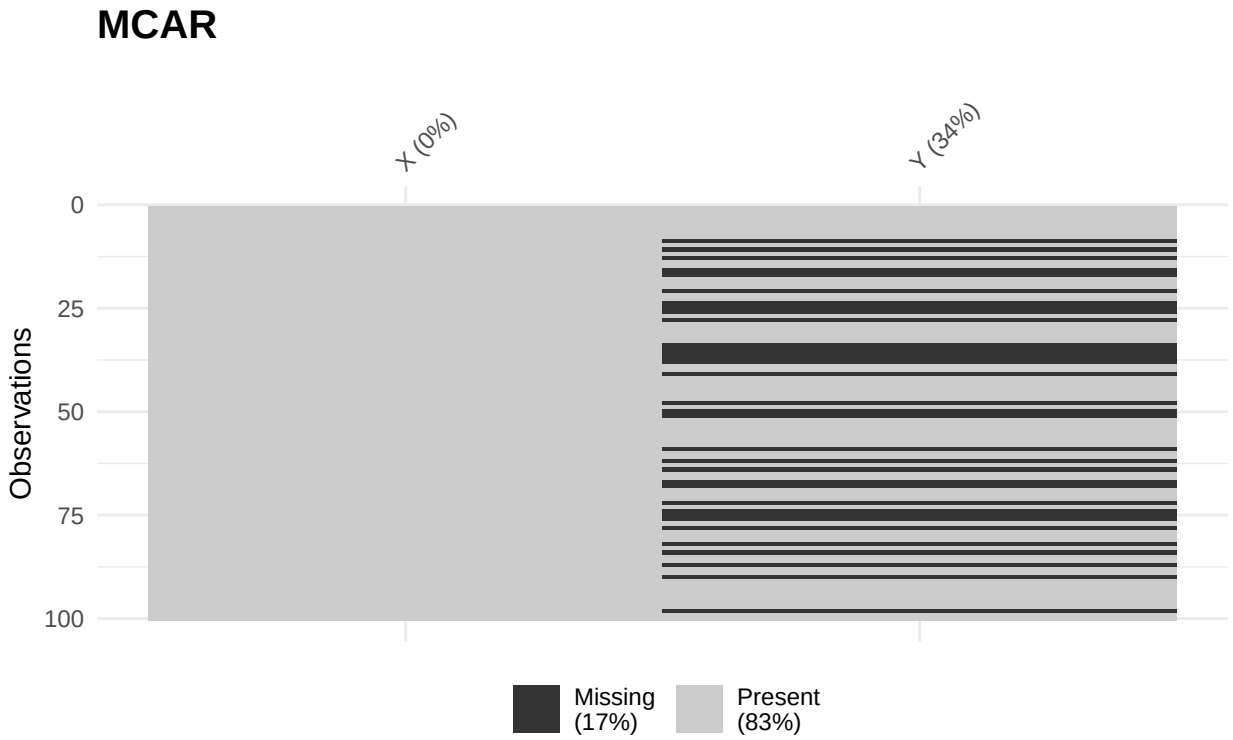


Figure 33: Pourcentage de données manquantes MCAR

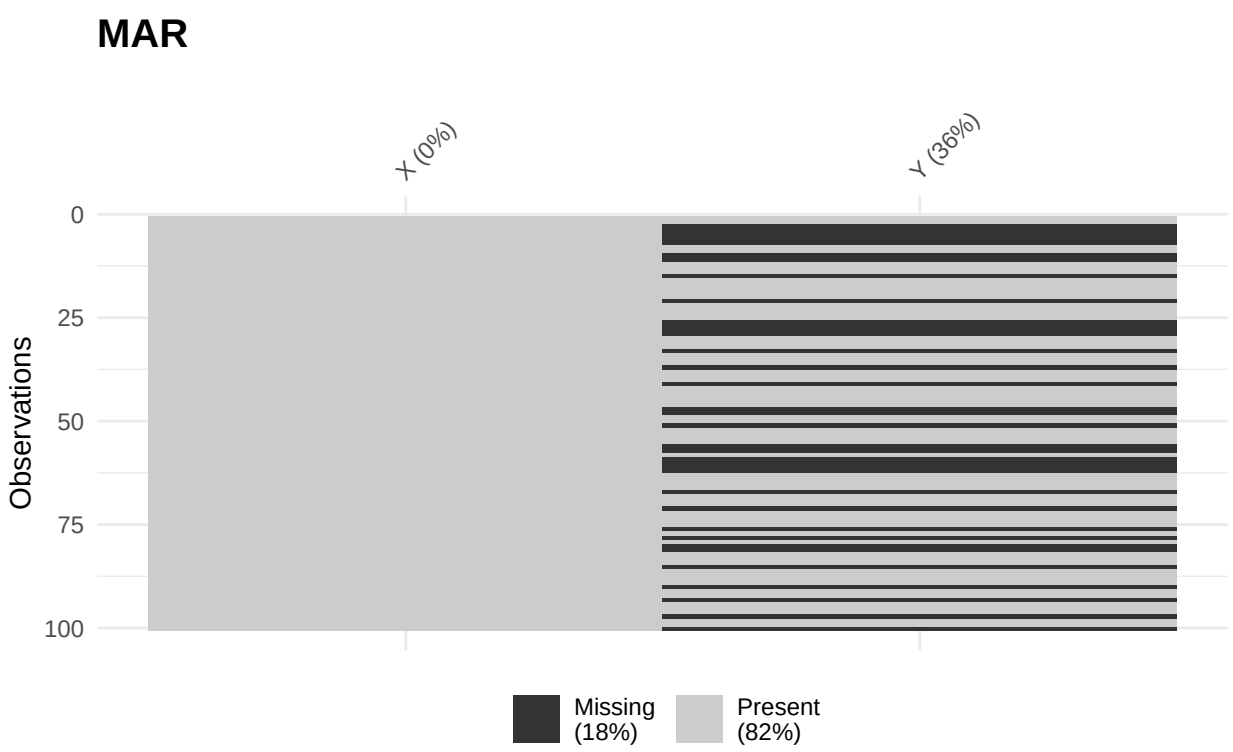


Figure 34: Pourcentage de données manquantes MAR

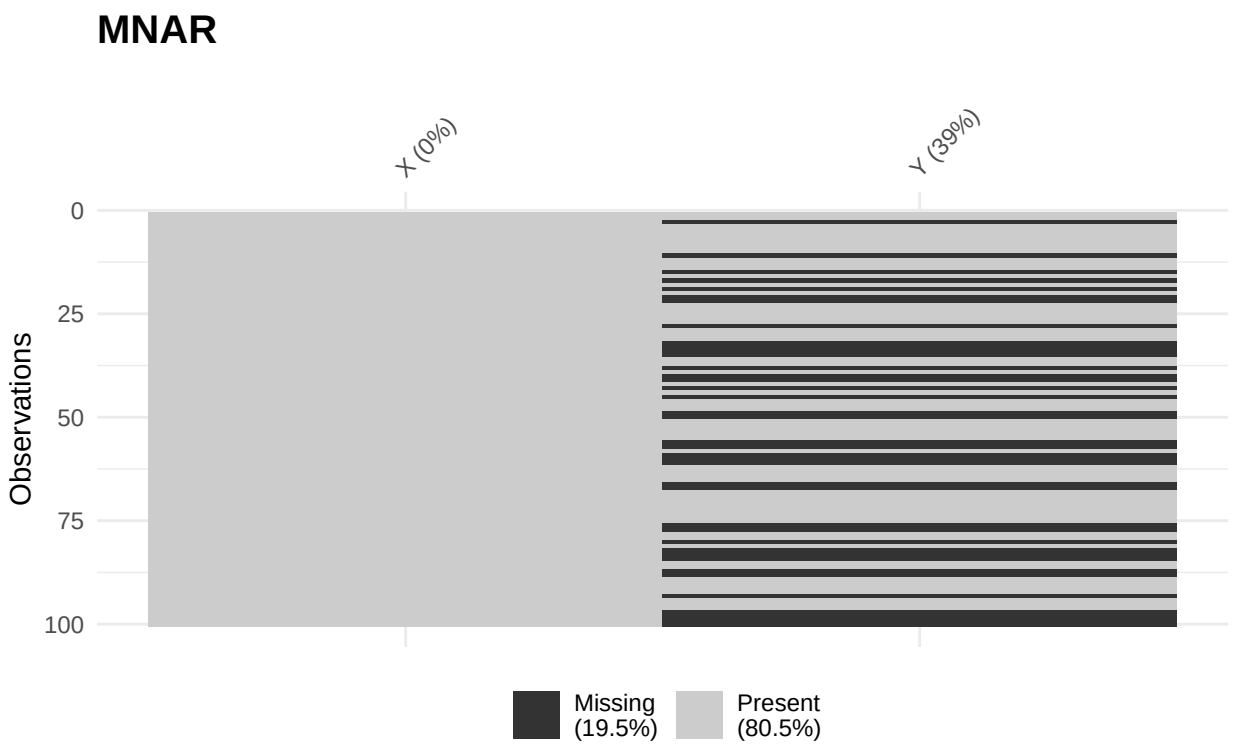


Figure 35: Pourcentage de données manquantes MNAR

Nous remarquons que les pourcentages de données manquantes sont globalement équivalents. En effet, elles

représentent environ 35% de la variable Y, équivalent à 18% de la totalité du jeu de données. Cependant, elles ne sont pas réparties de la même façon dans les différents cas.

Nous pouvons alors regarder plus en détails comment sont reparties les données manquantes de Y selon les valeurs de X pour chacun des jeux de données. Pour cela, nous utilisons la fonction `geom_miss_point` disponible dans `ggplot`.

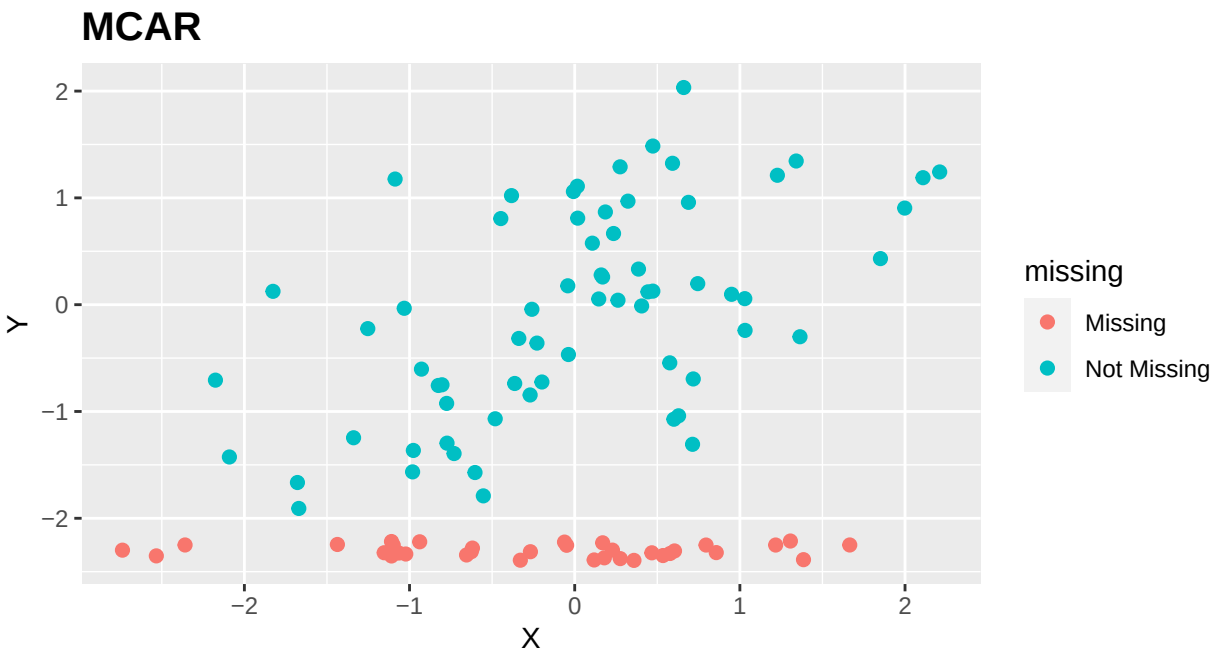


Figure 36: Répartition des données manquantes MCAR

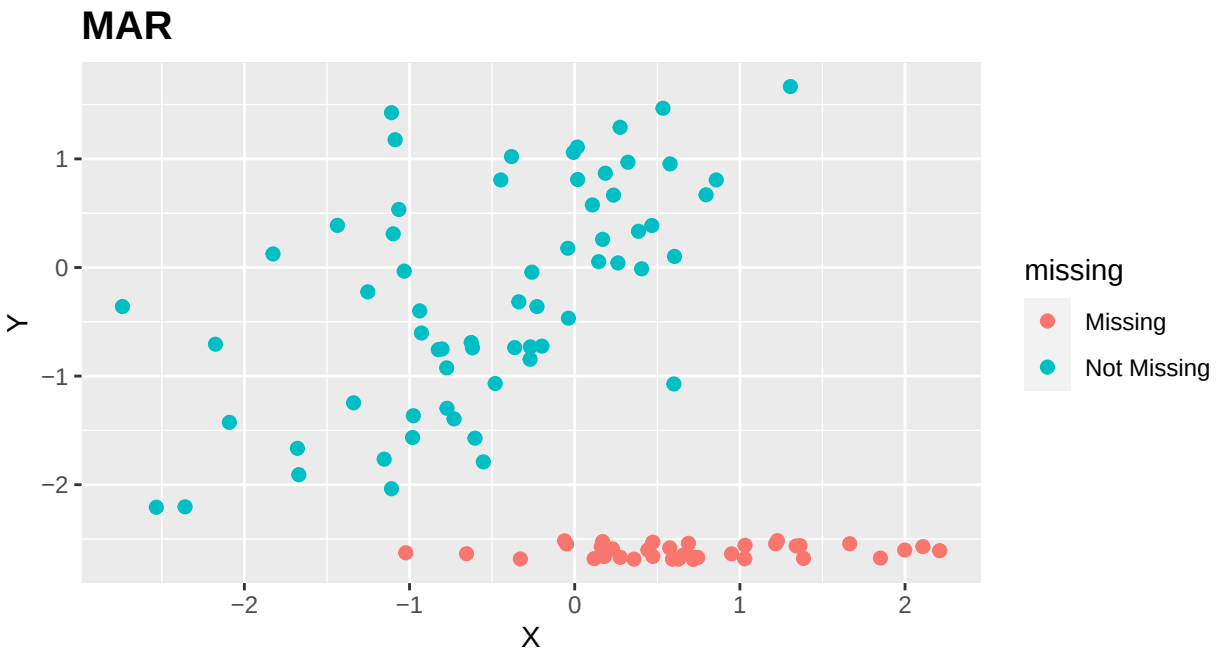


Figure 37: Répartition des données manquantes MAR

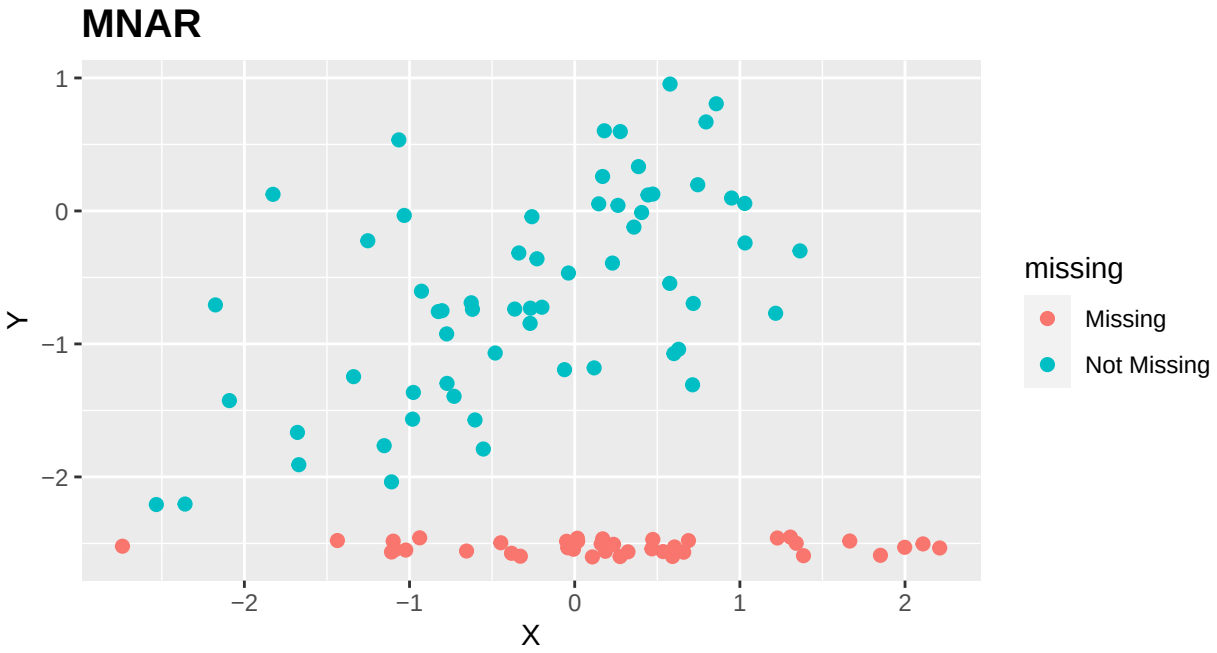


Figure 38: Répartition des données manquantes MNAR

Nous remarquons que pour la deuxième configuration, les valeurs manquantes sont plutôt associées à un X positif, tandis que dans les deux autres cas, elles sont reparties dans les négatifs également.

2.1.3 Estimation de la moyenne de Y et de l'intervalle de confiance associé

Nous voulons à présent étudier nos jeux de données et déterminer la moyenne de la variable Y, ainsi que l'intervalle de confiance associé pour chacun d'entre eux. Nous pourrons ensuite comparer les résultats obtenus.

Pour ce faire, plusieurs méthodes seront utilisées. En effet, dans un premier temps, nous allons utiliser uniquement les données non manquantes de la variable (cas complets). Puis, nous testerons les différentes méthodes d'imputation suivantes pour les valeurs non renseignées :

– L'imputation simple par la moyenne.

– L'imputation simple par régression stochastique.

– L'imputation multiple par régression stochastique.

– L'imputation multiple par régression.

- Analyse des cas complets

Comme évoqué, nous débutons par une analyse des cas complets. Cela consiste à supprimer les observations pour lesquelles la valeur de Y est manquante. C'est la méthode la plus simple. Cependant, elle est déconseillée lorsque le nombre de données manquantes est trop élevé car nous perdrions trop d'informations. De plus, pour les jeux de données où les valeurs manquantes ne sont pas type MCAR, retirer des observations va induire un biais dans l'analyse puisque l'ensemble des observations pour lesquelles des données sont manquantes n'est pas forcément représentatif de l'échantillon initial.

Dans notre cas, nous avons environ 30% de Na dans nos différents jeux de données. De plus, deux d'entre eux comportent des valeurs manquantes qui ne sont pas de type MCAR. Il ne serait donc pas judicieux de procéder ainsi. Cependant, l'objectif ici étant de comparer plusieurs méthodes, nous allons tout de même le réaliser. Pour cela, nous utilisons la fonction `na.omit`.

Nous pouvons regarder la dimension de nos nouveaux jeux de données “complets”.

Table 11: Dimension du jeu de données complet

	MCAR	MAR	MNAR
Observations	66	64	61
Variables	2	2	2

La dimension initiale était de 100 observations. Après suppression, nous en gardons nettement moins. Rappelons que les lignes retirées ne sont pas les mêmes pour chacun des jeux de données car les valeurs manquantes ne se situaient pas au même endroit.

Nous pouvons à présent estimer la moyenne de Y. Pour cela, nous réalisons une régression simple sur la variable, grâce à la fonction `lm` de la librairie `stats`, et nous récupérons l'intercept. Les moyennes obtenues sont présentes dans le tableau suivant :

Table 12: Moyenne de Y des cas complets

MCAR	MAR	MNAR
-0.0706092	-0.2493365	-0.61411

On remarque que les moyennes sont très différentes selon les jeux de données. La première est celle qui se rapporche le plus de 0, la vraie valeur.

Regardons maintenant les intervalles de confiance associés :

Table 13: Intervalle de confiance associé à la moyenne de Y des cas complets

	2.5%	97.5%
MCAR	-0.3093260	0.1681076
MAR	-0.5050327	0.0063597
MNAR	-0.8157242	-0.4124958

Nous remarquons ici que les deux premiers jeux de données, correspondants aux types MCAR et MAR, ont de “bons” résultats. En effet, contrairement au troisième, le 0 est présent dans l'intervalle. Cependant, le premier semble plus centré autour de la vraie valeur alors que le deuxième la contient de peu. Cela confirme ce que nous avions évoqué précédemment sur le biais que provoque cette méthode lorsque les données manquantes ne sont pas de type MCAR.

- Imputation simple par la moyenne

Pour éviter de supprimer les observations où se trouvent des données manquantes, on peut décider d'imputer des valeurs à ces endroits. Plusieurs méthodes existent pour déterminer la valeur à indiquer. La première que nous testons consiste à imputer le Na par la moyenne de la variable. Toutes les données manquantes seront donc “remplacées” par la même valeur.

Pour l'estimation de la moyenne de la variable ensuite, nous obtenons forcément les mêmes résultats qu'auparavant :

Table 14: Moyenne de Y après imputation simple par la moyenne

MCAR	MAR	MNAR
-0.0706092	-0.2493365	-0.61411

Regardons ce que donnent les intervalles de confiance associés :

Table 15: Intervalle de confiance associé à la moyenne de Y après imputation simple par la moyenne

	2.5%	97.5%
MCAR	-0.2267352	0.0855168
MAR	-0.4113632	-0.0873098
MNAR	-0.7357114	-0.4925086

Les intervalles de confiance ont changés. Il y a maintenant uniquement dans le premier cas (MCAR) que 0 se situe dans l'intervalle. En effet, pour les deux autres mécansimes, l'intervalle s'est décalé dans les négatifs. De plus, leur longueur a diminué. Ainsi, ils offrent plus de précision dans leur estimation.

- Imputation simple par régression stochastique

La deuxième méthode d'imputation simple que nous testons est l'imputation par régression stochastique. Celle-ci consiste à "remplacer" une donnée non observée par une valeur prédite obtenue en régressant la variable manquante sur d'autres variables, à laquelle on ajoute une valeur résiduelle aléatoire. Cela permet de préserver les relations entre les variables, mais aussi d'avoir l'avantage d'une composante aléatoire.

Concrètement, pour réaliser cette méthode, il nous faut tirer aléatoirement une valeur autour de celle prédite par le modèle de régression. Pour cela, nous utilisons la fonction `mice` avec la méthode `norm.nob`.

Nous visualisons ensuite les données pour s'assurer que la distribution des données imputées est globalement la même que celle des données initiales.

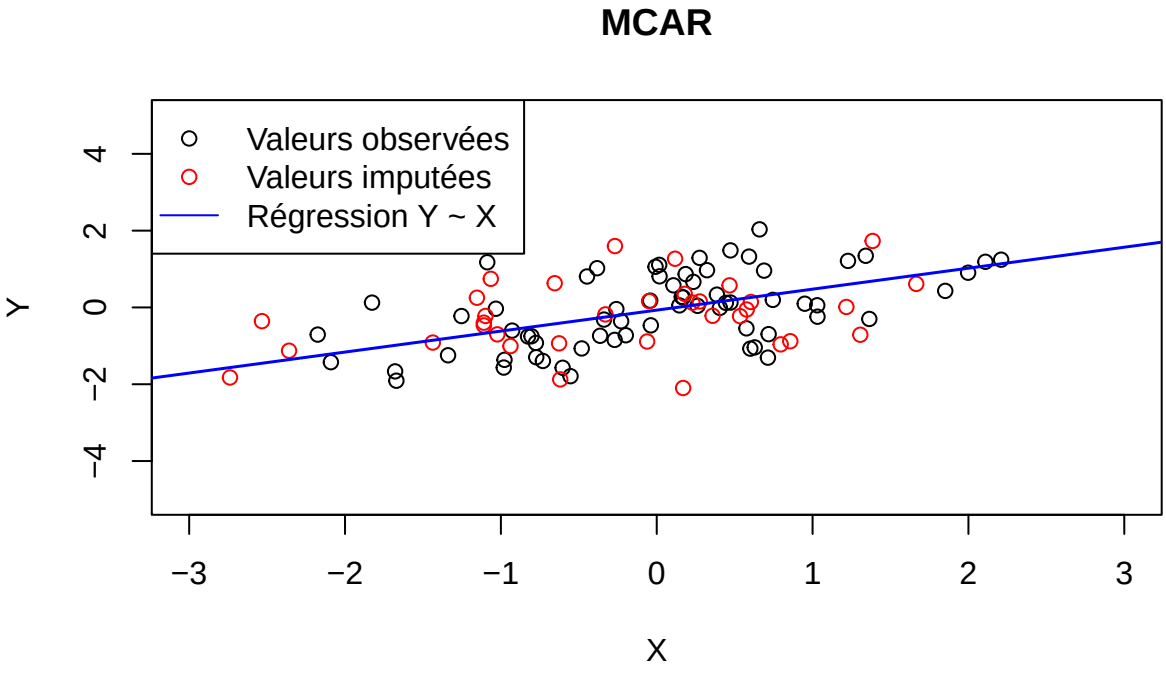


Figure 39: MCAR - Imputation simple par régression stochastique

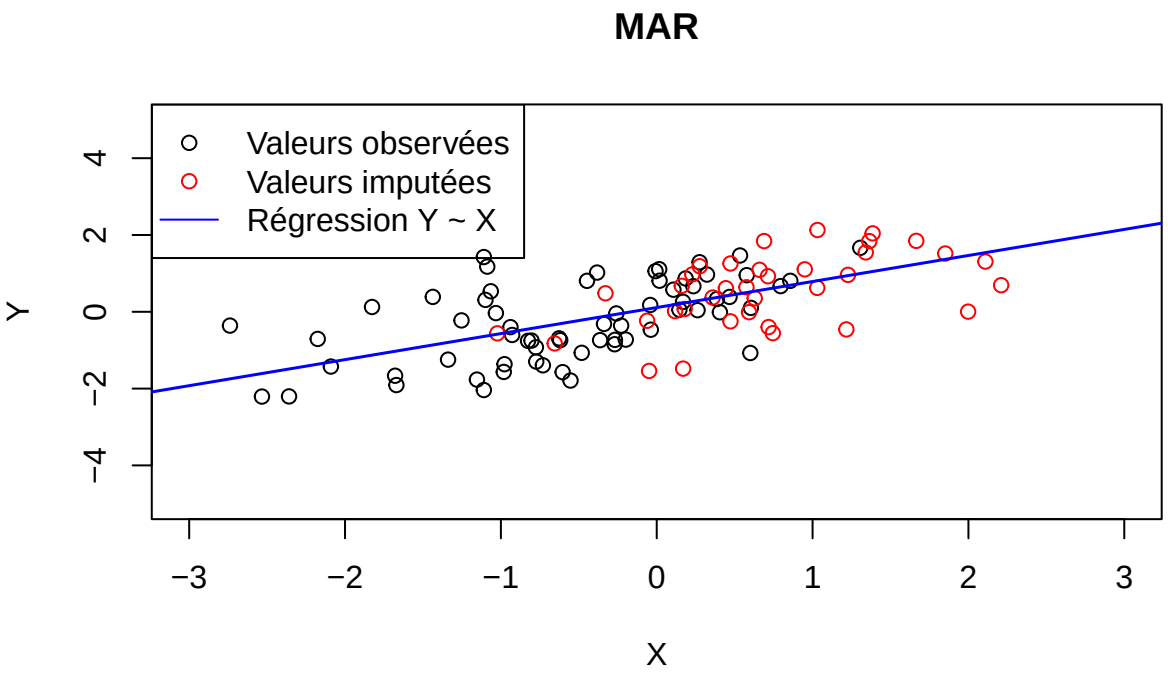


Figure 40: MAR - Imputation simple par régression stochastique

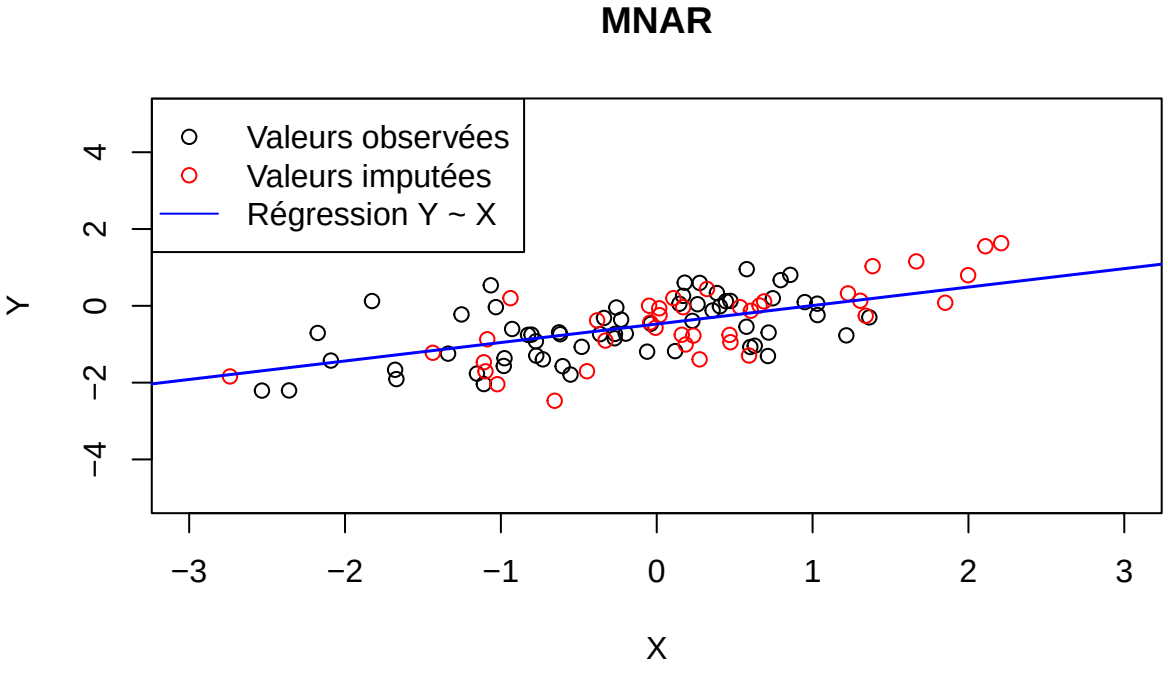


Figure 41: MNAR - Imputation simple par régression stochastique

Les différentes imputations semblent réalistes car elles suivent une distribution semblable à celles des données déjà existantes.

Nous estimons ensuite les moyennes et obtenons les résultats suivants :

Table 16: Moyenne de Y après imputation simple par régression stochastique

MCAR	MAR	MNAR
-0.1233357	0.0385562	-0.5310062

Cette fois-ci, c'est le deuxième cas (MAR) qui obtient le meilleur résultat car la moyenne obtenue est très proche de celle attendue.

Nous pouvons regarder ce que donnent les intervalles de confiance associés :

Table 17: Intervalle de confiance associé à la moyenne de Y après imputation simple par régression stochastique

	2.5%	97.5%
MCAR	-0.3110478	0.0643764
MAR	-0.1727160	0.2498284
MNAR	-0.7017427	-0.3602698

Les intervalles de confiance obtenus avec cette méthode d'imputation semblent meilleurs que ceux des méthodes précédemment testées. En effet, les deux premiers contiennent 0 et le troisième s'en rapproche. Cependant, leur longueur est plus grande que pour l'imputation avec la moyenne. Ils sont donc moins précis.

- Imputation multiple par régression stochastique

La méthode précédente ayant des résultats encourageants, nous souhaitons poursuivre son étude. En effet, une valeur unique ne pouvant pas refléter l'incertitude sur la prédiction, nous allons tester de renouveller vingt fois l'imputation avant de combiner les résultats. C'est ce qu'on appelle l'imputation multiple. Pour la réaliser, nous utilisons toujours la fonction `mice` mais en précisant l'argument `m=20`.

Nous obtenons alors 20 jeu de données imputés. Regardons les valeurs pour chacun d'eux, dans les trois cas que nous étudions :

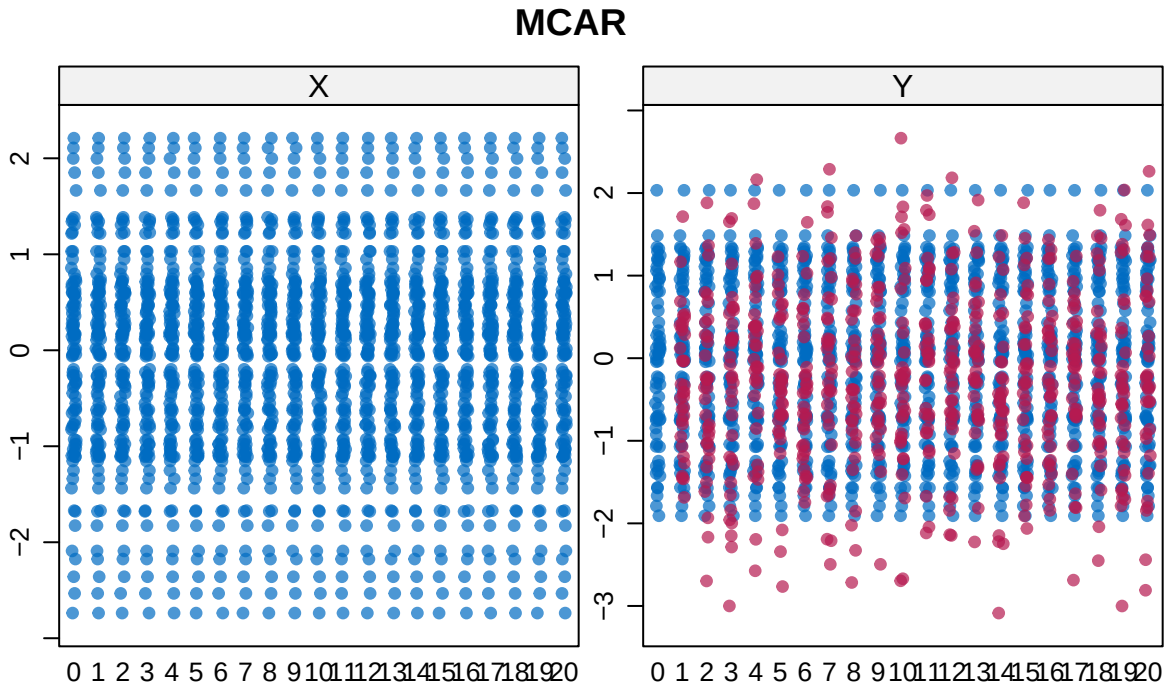


Figure 42: MCAR - Imputation multiple par régression stochastique

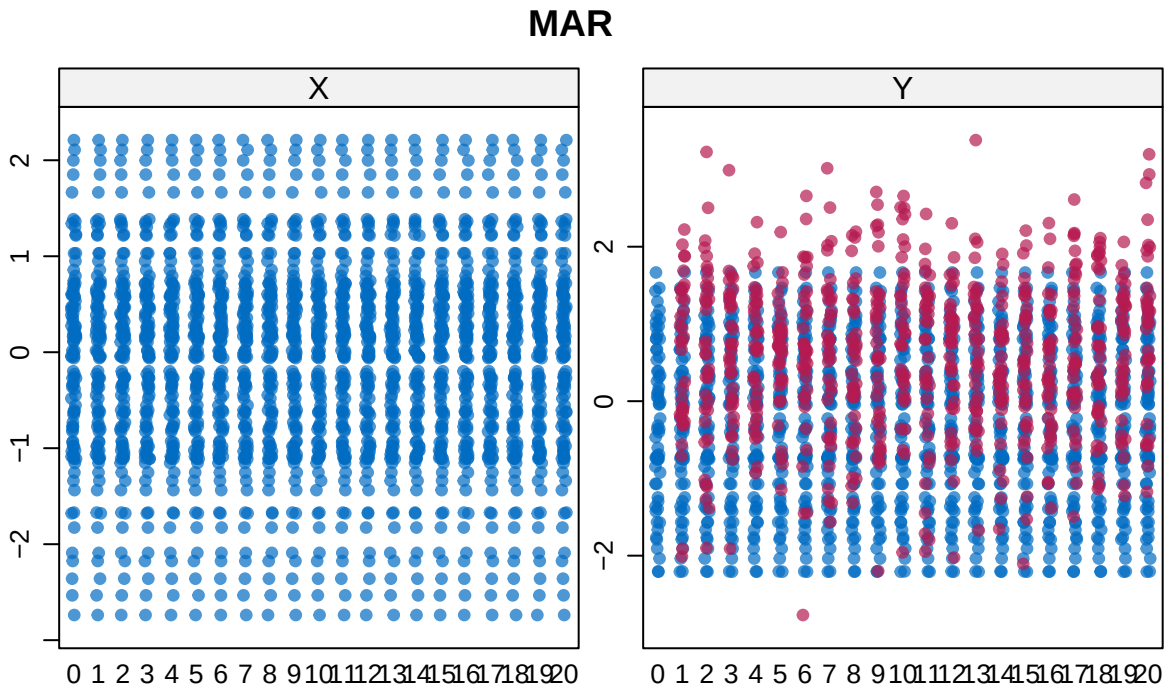


Figure 43: MAR - Imputation multiple par régression stochastique

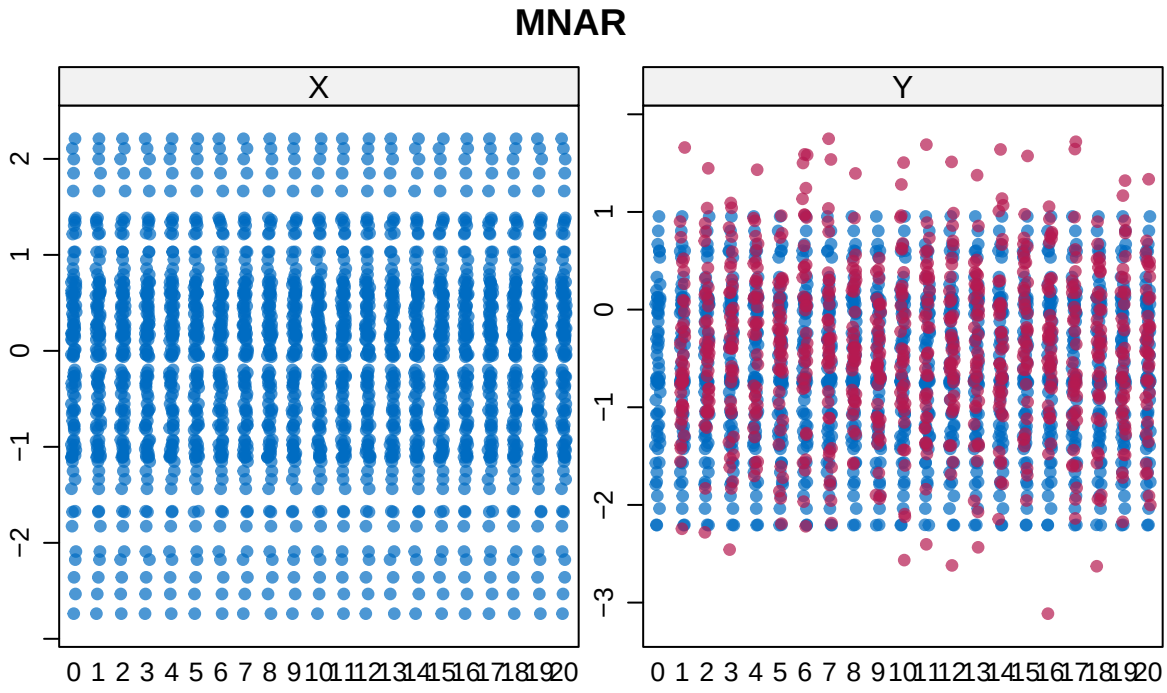


Figure 44: MNAR - Imputation multiple par régression stochastique

Les données imputées se trouvant en rouge, nous remarquons que chaque imputation est différente, mais respecte la distribution des données initiales.

Il faut maintenant analyser et combiner les données en utilisant les fonctions `with` et `pool` et on récupère la moyenne estimée.

Table 18: Moyenne de Y après combinaison des 20 imputations par régression stochastique

MCAR	MAR	MNAR
-0.1323226	0.0532383	-0.5287197

Une nouvelle fois, le mécanisme MAR obtient le meilleur résultat, même si il s'éloigne légèrement plus de 0. Cependant, l'incertitude de l'imputation des données manquantes étant prise en compte, on peut considérer que ce sont des résultats plus "justes".

Regardons les intervalles de confiance associés :

Table 19: Intervalle de confiance associé à la moyenne de Y après 20 imputations par régression stochastique

	2.5%	97.5%
MCAR	-0.3543063	0.0896611
MAR	-0.1855427	0.2920193
MNAR	-0.7095834	-0.3478559

Les conclusions sont les mêmes que pour les moyennes. En effet, les intervalles sont un peu plus grands mais sont sensiblement répartis de la même façon par rapport à 0 que pour l'imputation simple.

- Imputation multiple par régression

Enfin, nous allons tester de réaliser une nouvelle fois une imputation multiple. En effet, nous réalisons vingt imputations, mais cette fois-ci par régression linéaire. Nous le précisons dans le code en indiquant `method=norm` dans la fonction `mice`.

Nous obtenons les distributions suivantes :

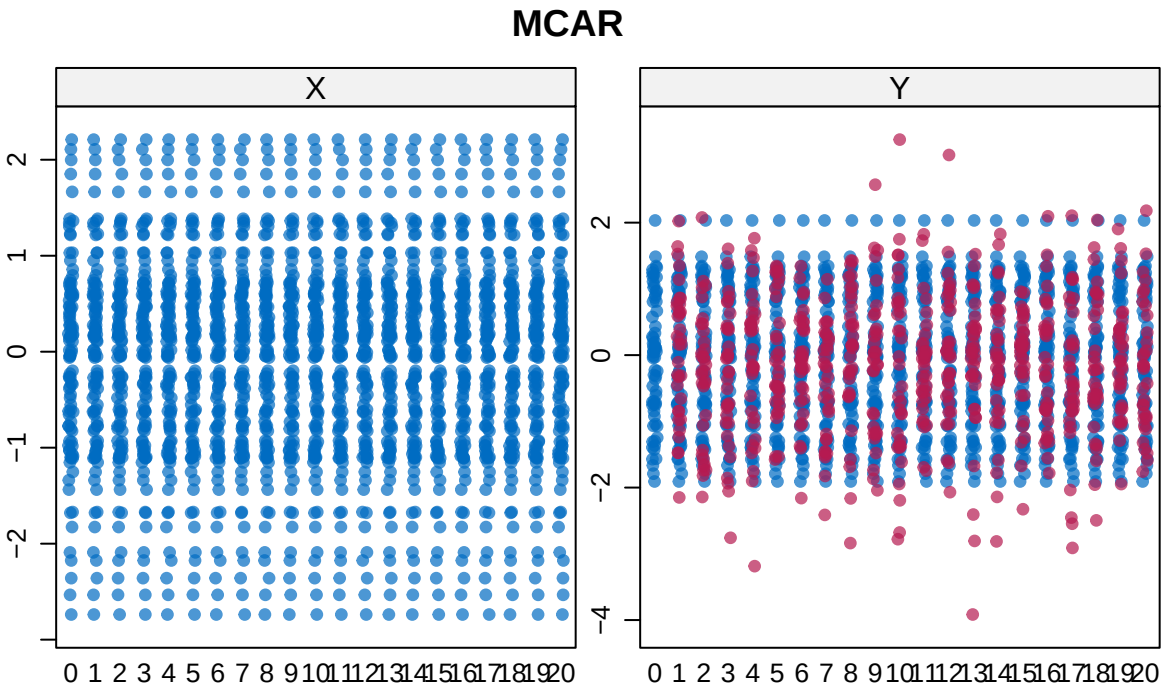


Figure 45: MCAR - Imputation multiple par régression

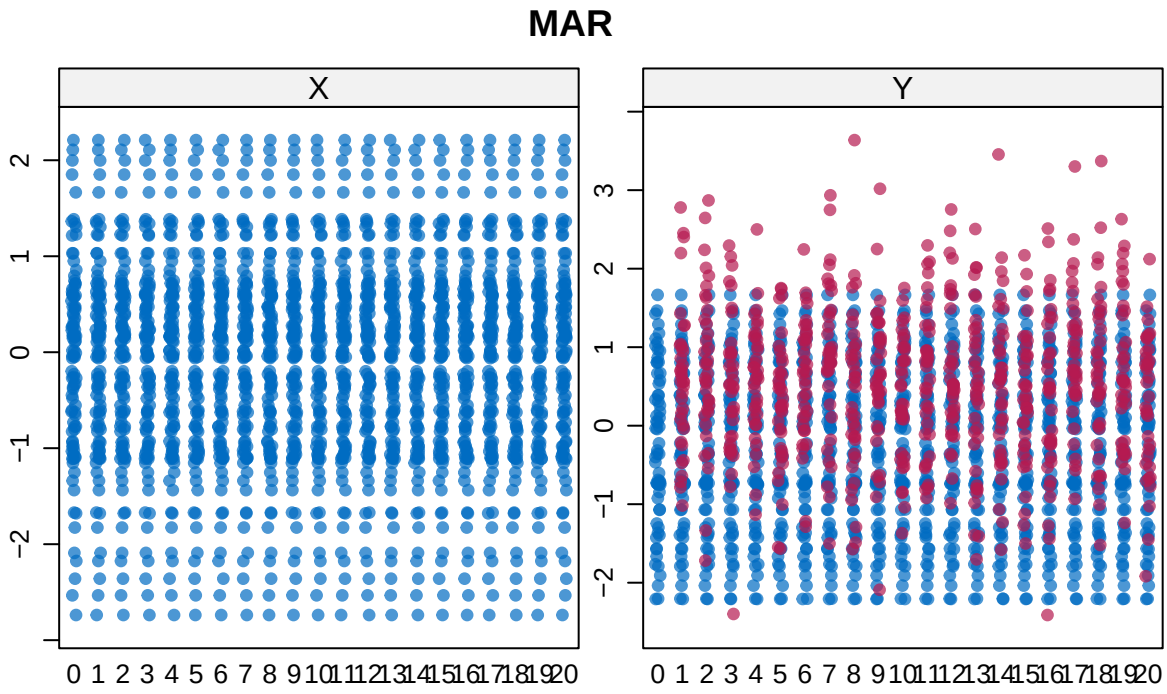


Figure 46: MAR - Imputation multiple par régression

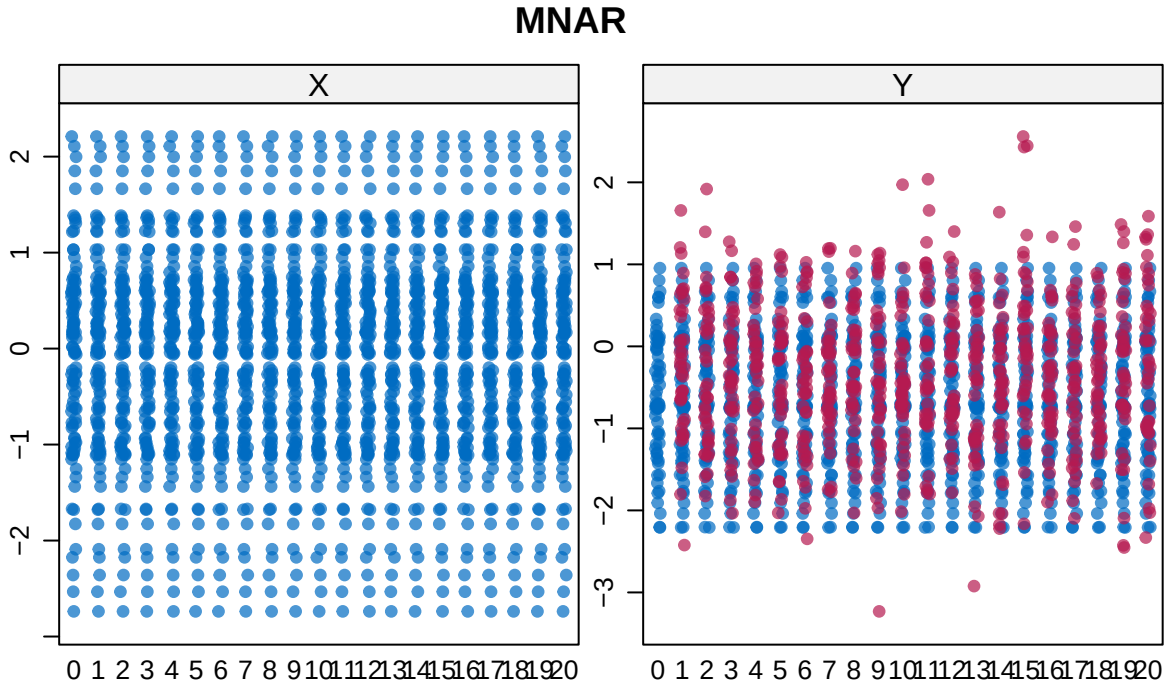


Figure 47: MNAR - Imputation multiple par régression MNAR

Nous calculons ensuite les moyennes et nous obtenons les résultats suivants :

Table 20: Moyenne de Y après combinaison des 20 imputations par régression

MCAR	MAR	MNAR
-0.1006362	0.0341727	-0.5099331

Le deuxième jeu de données obtient encore de bons résultats. L'imputation par régression semble donc mieux adaptée pour le mécanisme MAR.

Regardons si cela se confirme pour les intervalles de confiance associés aux moyennes :

Table 21: Intervalle de confiance associé à la moyenne de Y après 20 imputations par régression

	2.5%	97.5%
MCAR	-0.3169001	0.1156277
MAR	-0.2167796	0.2851250
MNAR	-0.7158475	-0.3040187

Nous observons que la vraie valeur est située dans les deux premiers intervalles uniquement. De plus, le deuxième semble être plus centré autour de 0.

2.1.4 Comparaison des stratégies

Afin de comparer chacune des méthodes précédentes, nous allons recommencer toutes les procédures 200 fois. Cela nous permettra d'estimer le biais commis sur l'estimation de la moyenne ainsi que la longueur moyenne et le taux de couverture de l'intervalle de confiance.

#"'{r, echo = F} set.seed(123) mean.mcar.cc <- c(1:200) mean.mar.cc <- c(1:200) mean.mnar.cc <- c(1:200)

mean.mcar.im <- c(1:200) mean.mar.im <- c(1:200) mean.mnar.im <- c(1:200)

mean.mcar.irs <- c(1:200) mean.mar.irs <- c(1:200) mean.mnar.irs <- c(1:200)

mean.mcar.irs20 <- c(1:200) mean.mar.irs20 <- c(1:200) mean.mnar.irs20 <- c(1:200)

mean.mcar.ir20 <- c(1:200) mean.mar.ir20 <- c(1:200) mean.mnar.ir20 <- c(1:200)

IC.mcar.cc <- matrix(ncol = 2, nrow = 200) IC.mar.cc <- matrix(ncol = 2, nrow = 200) IC.mnar.cc <- matrix(ncol = 2, nrow = 200)

IC.mcar.im <- matrix(ncol = 2, nrow = 200) IC.mar.im <- matrix(ncol = 2, nrow = 200) IC.mnar.im <- matrix(ncol = 2, nrow = 200)

IC.mcar.irs <- matrix(ncol = 2, nrow = 200) IC.mar.irs <- matrix(ncol = 2, nrow = 200) IC.mnar.irs <- matrix(ncol = 2, nrow = 200)

IC.mcar.irs20 <- matrix(ncol = 2, nrow = 200) IC.mar.irs20 <- matrix(ncol = 2, nrow = 200) IC.mnar.irs20 <- matrix(ncol = 2, nrow = 200)

IC.mcar.ir20 <- matrix(ncol = 2, nrow = 200) IC.mar.ir20 <- matrix(ncol = 2, nrow = 200) IC.mnar.ir20 <- matrix(ncol = 2, nrow = 200)

for (i in 1:200){ don <- rmvnorm(n=n, mean = mu, sigma = matcov) colnames(don) <- c("X", "Y")

#MCAR don.ismcar = don ismcar <- sample(c(T,F), size = n, prob = c(0.35, 0.65), replace = TRUE) don.ismcar[ismcar, "Y"] <- NA don.ismcar <- as.data.frame(don.ismcar)

#MAR don.ismar <- don ismar <- sapply(don.ismar[,"X"], FUN=function(xx){ prob <- pnorm(1.2*xx-.5) res <- sample(c(T,F), size = 1, prob = c(prob,1-prob)) return (res) }) don.ismar[ismar,"Y"] <- NA don.ismar <- as.data.frame(don.ismar)

#MNAR don.ismnar <- don ismnar <- sapply(don.ismnar[,“Y”], FUN=function(xx){ prob <- pnorm(1.2*xx-.5) res <- sample(c(T,F), size = 1, prob = c(prob,1-prob)) return (res) }) don.ismnar[ismnar,“Y”] <- NA don.ismnar <- as.data.frame(don.ismnar)

#Cas complet don.ismcar.cc <- na.omit(don.ismcar) don.ismar.cc <- na.omit(don.ismar) don.ismnar.cc <- na.omit(don.ismnar)

lm.mcar.cc <- lm(don.ismcar.cc$Y$ 1)$lm.mar.cc <- lm(don.ismar.cc$Y~1) lm.mnar.cc <- lm(don.ismnar.cc$Y~1)

mean.mcar.cc[i] <- lm.mcar.cc$coefficients$mean.mar.cc[i] <- lm.mar.cc$coefficients mean.mnar.cc[i] <- lm.mnar.cc$coefficients

IC.mcar.cc[i,] <- confint(lm.mcar.cc) IC.mar.cc[i,] <- confint(lm.mar.cc) IC.mnar.cc[i,] <- confint(lm.mnar.cc)

#Imputation par la moyenne don.ismcar.im <- na.aggregate(don.ismcar, FUN = mean) don.ismar.im <- na.aggregate(don.ismar, FUN = mean) don.ismnar.im <- na.aggregate(don.ismnar, FUN = mean)

lm.mcar.im <- lm(don.ismcar.im$Y$ 1)$lm.mar.im <- lm(don.ismar.im$Y~1) lm.mnar.im <- lm(don.ismnar.im$Y~1)

mean.mcar.im[i] <- lm.mcar.im$coefficients$mean.mar.im[i] <- lm.mar.im$coefficients mean.mnar.im[i] <- lm.mnar.im$coefficients

IC.mcar.im[i,] <- confint(lm.mcar.im) IC.mar.im[i,] <- confint(lm.mar.im) IC.mnar.im[i,] <- confint(lm.mnar.im)

#Imputation par régression stochastique don.ismcar.irs <- complete(mice(don.ismcar, method = “norm.nob”, m = 1, print = FALSE)) don.ismar.irs <- complete(mice(don.ismar, method = “norm.nob”, m = 1, print = FALSE)) don.ismnar.irs <- complete(mice(don.ismnar, method = “norm.nob”, m = 1, print = FALSE))

lm.mcar.irs <- lm(don.ismcar.irs$Y$ 1)$lm.mar.irs <- lm(don.ismar.irs$Y~1) lm.mnar.irs <- lm(don.ismnar.irs$Y~1)

mean.mcar.irs[i] <- lm.mcar.irs$coefficients$mean.mar.irs[i] <- lm.mar.irs$coefficients mean.mnar.irs[i] <- lm.mnar.irs$coefficients

IC.mcar.irs[i,] <- confint(lm.mcar.irs) IC.mar.irs[i,] <- confint(lm.mar.irs) IC.mnar.irs[i,] <- confint(lm.mnar.irs)

#20 imputations par régression stochastique don.ismcar.irs20 <- mice(don.ismcar, method = “norm.nob”, m = 20, print = FALSE) don.ismar.irs20 <- mice(don.ismar, method = “norm.nob”, m = 20, print = FALSE) don.ismnar.irs20 <- mice(don.ismnar, method = “norm.nob”, m = 20, print = FALSE)

fit_mcar.irs20 <- with(don.ismcar.irs20, lm(Y~1)) fitpool_mcar.irs20 <- pool(fit_mcar.irs20)

fit_mar.irs20 <- with(don.ismar.irs20, lm(Y~1)) fitpool_mar.irs20 <- pool(fit_mar.irs20)

fit_mnar.irs20 <- with(don.ismnar.irs20, lm(Y~1)) fitpool_mnar.irs20 <- pool(fit_mnar.irs20)

mean.mcar.irs20[i] <- summary(fitpool_mcar.irs20, conf.int = TRUE)[2][1,1] mean.mar.irs20[i] <- summary(fitpool_mar.irs20, conf.int = TRUE)[2][1,1] mean.mnar.irs20[i] <- summary(fitpool_mnar.irs20, conf.int = TRUE)[2][1,1]

inf_mcar.irs20 <- summary(fitpool_mcar.irs20, conf.int = TRUE)[7] sup_mcar.irs20 <- summary(fitpool_mcar.irs20, conf.int = TRUE)[8]

inf_mar.irs20 <- summary(fitpool_mar.irs20, conf.int = TRUE)[7] sup_mar.irs20 <- summary(fitpool_mar.irs20, conf.int = TRUE)[8]

inf_mnar.irs20 <- summary(fitpool_mnar.irs20, conf.int = TRUE)[7] sup_mnar.irs20 <- summary(fitpool_mnar.irs20, conf.int = TRUE)[8]

IC.mcar.irs20[i,] <- matrix(c(inf_mcar.irs20[1,1], sup_mcar.irs20[1,1]), ncol = 2) IC.mar.irs20[i,] <- matrix(c(inf_mar.irs20[1,1], sup_mar.irs20[1,1]), ncol = 2) IC.mnar.irs20[i,] <- matrix(c(inf_mnar.irs20[1,1], sup_mnar.irs20[1,1]), ncol = 2)

#20 imputations par régression don.ismcar.ir20 <- mice(don.ismcar, method = “norm”, m = 20, print = FALSE) don.ismar.ir20 <- mice(don.ismar, method = “norm”, m = 20, print = FALSE) don.ismnar.ir20 <- mice(don.ismnar, method = “norm”, m = 20, print = FALSE)

fit_mcar.ir20 <- with(don.ismcar.ir20, lm(Y~1)) fitpool_mcar.ir20 <- pool(fit_mcar.ir20)

fit_mar.ir20 <- with(don.ismar.ir20, lm(Y~1)) fitpool_mar.ir20 <- pool(fit_mar.ir20)

fit_mnar.ir20 <- with(don.ismnar.ir20, lm(Y~1)) fitpool_mnar.ir20 <- pool(fit_mnar.ir20)

mean.mcar.ir20[i] <- summary(fitpool_mcar.ir20, conf.int = TRUE)[2][1,1] mean.mar.ir20[i] <- summary(fitpool_mar.ir20, conf.int = TRUE)[2][1,1] mean.mnar.ir20[i] <- summary(fitpool_mnar.ir20, conf.int = TRUE)[2][1,1]

inf_mcar.ir20 <- summary(fitpool_mcar.ir20, conf.int = TRUE)[7] sup_mcar.ir20 <- summary(fitpool_mcar.ir20, conf.int = TRUE)[8]

inf_mar.ir20 <- summary(fitpool_mar.ir20, conf.int = TRUE)[7] sup_mar.ir20 <- summary(fitpool_mar.ir20, conf.int = TRUE)[8]

inf_mnar.ir20 <- summary(fitpool_mnar.ir20, conf.int = TRUE)[7] sup_mnar.ir20 <- summary(fitpool_mnar.ir20, conf.int = TRUE)[8]

IC.mcar.ir20[i,] <- matrix(c(inf_mcar.ir20[1,1], sup_mcar.ir20[1,1]), ncol = 2) IC.mar.ir20[i,] <- matrix(c(inf_mar.ir20[1,1], sup_mar.ir20[1,1]), ncol = 2) IC.mnar.ir20[i,] <- matrix(c(inf_mnar.ir20[1,1], sup_mnar.ir20[1,1]), ncol = 2) }

```
-   Biais commis sur l'estimation de la moyenne

Nous savons que le biais s'écrit de la forme $Biais(\hat{\theta})=\mathbb{E}(\hat{\theta}) - \theta$, ou

Commençons donc par regarder l'évolution de la moyenne sur ces $200$ imputations pour chacune des méthod

\begin{figure}[H]
\includegraphics{TP_mites_files/figure-latex/unnamed-chunk-115-1} \caption{Evolution de la moyenne de Y
\end{figure}

\begin{figure}[H]
\includegraphics{TP_mites_files/figure-latex/unnamed-chunk-116-1} \caption{Evolution de la moyenne de Y
\end{figure}

\begin{figure}[H]
\includegraphics{TP_mites_files/figure-latex/unnamed-chunk-117-1} \caption{Evolution de la moyenne de Y
\end{figure}

\begin{figure}[H]
\includegraphics{TP_mites_files/figure-latex/unnamed-chunk-118-1} \caption{Evolution de la moyenne de Y
\end{figure}

\begin{figure}[H]
\includegraphics{TP_mites_files/figure-latex/unnamed-chunk-119-1} \caption{Evolution de la moyenne de Y
\end{figure}

Tout d'abord, nous remarquons que les moyennes obtenues avec le mécanisme MCAR sont toujours autour de $

-   Longueur moyenne de l'intervalle de confiance

Regardons maintenant comment évolue l'intervalle de confiance associé à la moyenne. Pour ce faire, nous
```

```
#```{r, echo = F}
l.mcar.cc <- mean(IC.mcar.cc[,2]-IC.mcar.cc[,1])
l.mar.cc <- mean(IC.mar.cc[,2]-IC.mar.cc[,1])
l.mnar.cc <- mean(IC.mnar.cc[,2]-IC.mnar.cc[,1])

l.mcar.im <- mean(IC.mcar.im[,2]-IC.mcar.im[,1])
l.mar.im <- mean(IC.mar.im[,2]-IC.mar.im[,1])
l.mnar.im <- mean(IC.mnar.im[,2]-IC.mnar.im[,1])

l.mcar.irs <- mean(IC.mcar.irs[,2]-IC.mcar.irs[,1])
l.mar.irs <- mean(IC.mar.irs[,2]-IC.mar.irs[,1])
l.mnar.irs <- mean(IC.mnar.irs[,2]-IC.mnar.irs[,1])

l.mcar.irs20 <- mean(IC.mcar.irs20[,2]-IC.mcar.irs20[,1])
l.mar.irs20 <- mean(IC.mar.irs20[,2]-IC.mar.irs20[,1])
l.mnar.irs20 <- mean(IC.mnar.irs20[,2]-IC.mnar.irs20[,1])

l.mcar.ir20 <- mean(IC.mcar.ir20[,2]-IC.mcar.ir20[,1])
l.mar.ir20 <- mean(IC.mar.ir20[,2]-IC.mar.ir20[,1])
l.mnar.ir20 <- mean(IC.mnar.ir20[,2]-IC.mnar.ir20[,1])

L <- rbind(data.frame('1'= l.mcar.cc, '2' = l.mar.cc, '3' = l.mnar.cc),data.frame('1'= l.mcar.im,'2'= l

colnames(L) <- c("MCAR", "MAR", "MNAR")
rownames(L) <- c("Cas complets", "Imputation simple par la moyenne", "Imputation simple par régression s

kable(data.frame(L), col.names = c("MCAR", "MAR", "MNAR"), position="H", caption = "Longueur moyenne de
```

Nous observons que l'imputation par la moyenne obtient les longueurs d'intervalle les plus faibles, suivie par l'imputation simple par régression stochastique. Cependant, il ne faut pas oublier que dans ces cas, on ne tient pas compte du fait que des valeurs ont été remplacées et on fait comme si c'était de vraies observations. Cela va en général sous-évaluer la variabilité dans les données et peut expliquer des intervalles de confiance plus courts.

Les imputations multiples obtiennent donc des intervalles plus grands mais tentent de mieux reproduire la variabilité des données. Lorsque l'on compare les deux imputations de ce type effectuées, nous remarquons que celle par régression stochastique semble être plus performante car fournit un intervalle plus précis. De plus, les résultats sont également meilleurs que lorsque l'on a étudié uniquement les cas complets.

- Taux de couverture de l'intervalle de confiance

Enfin, nous allons analyser le taux de couverture des intervalles de confiance. En effet, la valeur attendue de la moyenne étant 0, nous regardons combien de fois celle-ci est comprise dans l'intervalle obtenu. C'est ce que nous résumons en pourcentage dans le tableau suivant :

#"'{r, echo=F} tc.mcar.cc <- (sum(sign(IC.mcar.cc[,1]*IC.mcar.cc[,2])<0)*100/200) tc.mar.cc <- (sum(sign(IC.mar.cc[,1]*IC.mar.cc[,2])<0)*100/200) tc.mnar.cc <- (sum(sign(IC.mnar.cc[,1]*IC.mnar.cc[,2])<0)*100/200)

tc.mcar.im <- (sum(sign(IC.mcar.im[,1]*IC.mcar.im[,2])<0)*100/200) tc.mar.im <- (sum(sign(IC.mar.im[,1]*IC.mar.im[,2])<0)*10
tc.mnar.im <- (sum(sign(IC.mnar.im[,1]*IC.mnar.im[,2])<0)*100/200)

tc.mcar.irs <- (sum(sign(IC.mcar.irs[,1]*IC.mcar.irs[,2])<0)*100/200) tc.mar.irs <- (sum(sign(IC.mar.irs[,1]*IC.mar.irs[,2])<0)*10
tc.mnar.irs <- (sum(sign(IC.mnar.irs[,1]*IC.mnar.irs[,2])<0)*100/200)

tc.mcar.irs20 <- (sum(sign(IC.mcar.irs20[,1]*IC.mcar.irs20[,2])<0)*100/200) tc.mar.irs20 <- (sum(sign(IC.mar.irs20[,1]*IC.mar.irs
tc.mnar.irs20 <- (sum(sign(IC.mnar.irs20[,1]*IC.mnar.irs20[,2])<0)*100/200)

tc.mcar.ir20 <- (sum(sign(IC.mcar.ir20[,1]*IC.mcar.ir20[,2])<0)*100/200) tc.mar.ir20 <- (sum(sign(IC.mar.ir20[,1]*IC.mar.ir20[,2
tc.mnar.ir20 <- (sum(sign(IC.mnar.ir20[,1]*IC.mnar.ir20[,2])<0)*100/200)

L <- rbind(data.frame('1'= tc.mcar.cc, '2' = tc.mar.cc, '3' = tc.mnar.cc),data.frame('1'= tc.mcar.im,'2'= tc.mar.im,'3' = tc.mnar.im),data.frame('1'= tc.mcar.irs,'2'= tc.mar.irs,'3' = tc.mnar.irs), data.frame('1'= tc.mcar.irs20,'2'= tc.mar.irs20,'3' = tc.mnar.irs20), data.frame('1'= tc.mcar.ir20,'2'= tc.mar.ir20,'3' = tc.mnar.ir20))

colnames(L) <- c("MCAR", "MAR", "MNAR") rownames(L) <- c("Cas complets", "Imputation par la moyenne", "Imputation simple par régression stochastqiue", "Imputation multiple par régression stochastique", "Imputation multiple par régression")

```
#```{r, echo=F}
kable(data.frame(L), position="H", caption = "Taux de couverture de l'intervalle de confiance")
```

Premièrement, ce tableau confirme les graphiques analysés plus haut. En effet, nous remarquons de manière évidente que le jeu de données associé au mécanisme MNAR n'a quasiment pas permis d'obtenir des intervalles de confiance contenant la vraie valeur que l'on cherche à estimer (la moyenne). De plus, concernant les mécanismes MCAR et MAR, nous voyons une différence pour les cas complets et l'imputation par la moyenne, puis les résultats sont semblables pour les autres. Notons tout de même que, contrairement à la longeur de l'intervalle, la taux de couverture semble être meilleur pour l'imputation multiple par régression que celle par régression stochastique.

Conclusion : Nous avons comparé trois mécanismes de répartition des données manquantes (MCAR, MAR et MNAR) sur un jeu de données où nous avons réalisé une analyse des cas complets, deux imputations simples (par la moyenne et par régression stochastique) et deux imputations multiples (par régression stochastique et régression). Après différentes études, nous en concluons de bons résultats pour le mécanisme MCAR ainsi que pour le mécanisme MAR sur les imputations par régression. Quand aux différentes méthodes d'imputation, les résultats des impuations simples semblaient meilleurs mais avec toutes les limites qu'ils comportent sur la variabilité des données. Ainsi, une imputation multiple est donc conseillée. Nous avons obtenu de meilleurs résultats lorsqu'elle était par régression stochastique pour la longueur de l'intervalle. Cependant, pour le taux de couverture, l'imputation multiple par régression était légèrement plus performante.

2.2 Scénarios NA sur le jeu de données `mites`

Nous allons dans un premier temps générer 10% de données manquantes sur notre jeu de données `mites` suivant les trois catégories utilisées précédemment, MCAR, MAR et MNAR. Nous allons utiliser la fonction `ampute` du package `mice`. Pour utiliser cette fonction, nous allons uniquement nous concentrer sur les variables quantitatives de notre jeu de données `mites` et ainsi supprimer les variables qualitatives, `Substrate`, `Shrub` et `Topo` et la variable binaire `pa`. Nous obtenons donc un nouveau jeu de données.

```
str(mites1)
```

```
## 'data.frame':    70 obs. of  5 variables:
##  $ Galumna   : int  8 3 1 1 2 1 1 1 2 5 ...
##  $ totalabund: int  140 268 186 286 199 209 162 126 123 166 ...
##  $ prop      : num  0.05714 0.01119 0.00538 0.0035 0.01005 ...
##  $ SubsDens  : num  39.2 55 46.1 48.2 23.6 ...
##  $ WatrCont  : num  350 435 372 360 204 ...
```

L'utilisation de la fonction `ampute` nécessite une attention particulière. En effet, dans l'aide de la fonction nous avons que l'argument `prop` signifie "un scalaire spécifiant la proportion de données manquantes". Cependant, nous avons remarqué que l'utilisation seule de cet argument ne nous donne pas 10% de données manquantes sur notre jeu de données. Dans notre exemple ci dessous nous avons seulement 8 données manquantes.

```
amp.mcar1 = ampute(mites1, prop = 0.10, mech = "MCAR")
countNA(amp.mcar1$amp)
```

```
## [1] 8
```

En cherchant un peu plus loin, nous avons remarqué que l'argument `prop` signifie la proportion de lignes incomplètes dans notre jeu de données. Nous avons 70 observations pour 5 variables soit 350 données (~350 cellules). Nous avons donc pensé qu'il était préférable de générer 10% de données manquantes sur l'ensemble de nos données, soit 10% de 350 données, ce qui equivaut à +/- 35 données manquantes. Pour cela, nous devons utiliser l'argument `bycases` comme `FALSE`. En effet, l'argument `bycases` étant `FALSE` signifie que la proportion de données manquantes est définie en terme de cellules et non en terme de lignes, qui est la valeur par défault. Pour spécifier les 3 différents méchanismes MCAR, MAR et MNAR, nous utilisons l'argument `mech`.

- Mécanisme **MCAR** (**M**issing **C**ompletely **A**t **R**andom)

```
amp.mcar = ampute(mites1, prop = 0.10, mech = "MCAR", bycases = FALSE)
countNA(amp.mcar$amp)
```

```
## [1] 37
```

- Mécanisme **MAR** (**M**issing **A**t **R**andom)

```
amp.mar = ampute(mites1, prop = 0.10, mech = "MAR", bycases = FALSE)
countNA(amp.mar$amp)
```

```
## [1] 35
```

- Mécanisme **MNAR** (**M**issing **N**ot **A**t **R**andom)

```
amp.mnar = ampute(mites1, prop = 0.10, mech = "MNAR", bycases = FALSE)
countNA(amp.mnar$amp)
```

```
## [1] 27
```

Table 22: Proportion de données manquantes

MCAR	MAR	MNAR
0.1057143	0.1	0.0771429

Nous avons donc une proportion de ~10% de données manquantes générées.

2.2.1 Analyse exploratoire

Nous allons maintenant explorer et analyser les trois dispositifs de données manquantes créés précédemment. Dans un premier temps, nous ferons une analyse univariée, puis une analyse bivariée et pour finir nous ferons une analyse multidimentionnelle.

Analyse univariée

Nous utilisons la fonction `aggr` du package `VIM` pour l'analyse univariée.

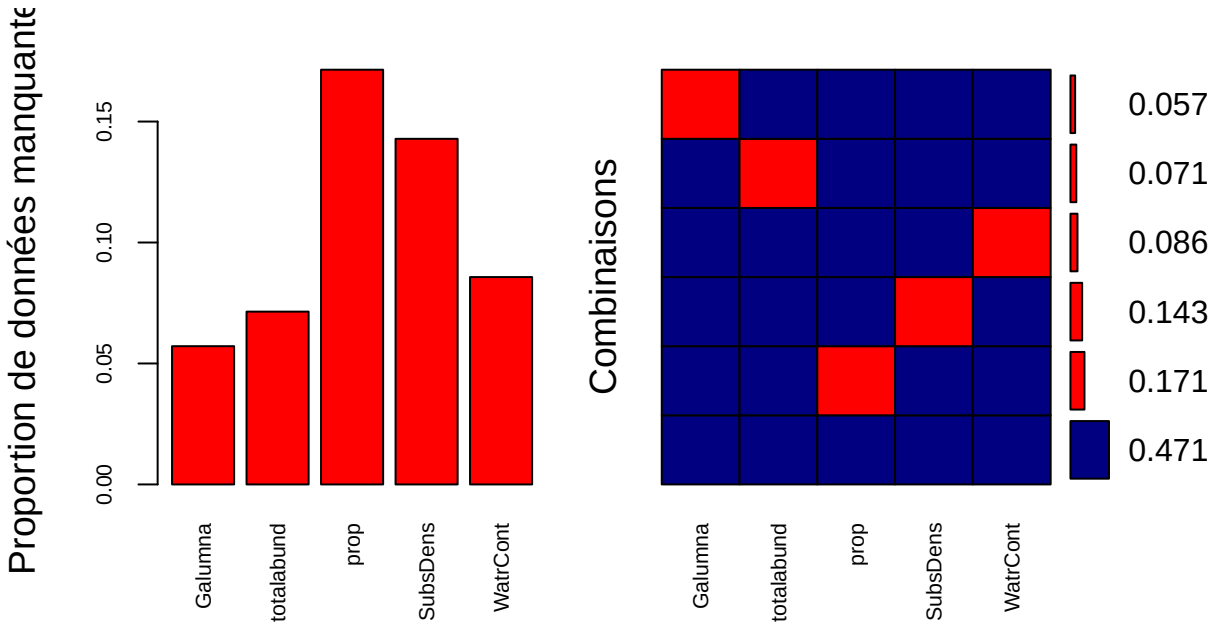


Figure 48: MCAR

Les colonnes représentent les variables, les lignes représentent les types de "situations" des données manquantes générées dans le jeu de données. La ligne entière bleue représente toutes les lignes dans le data frame qui ne comportent pas de données manquantes, soit 51,4% des lignes dans notre cas. Une case rouge représente toutes les lignes dans le data frame qui sont manquantes pour cette variable. Dans notre cas nous avons par exemple, 10% de lignes qui contiennent des données manquantes pour la variable `WatrCont`.

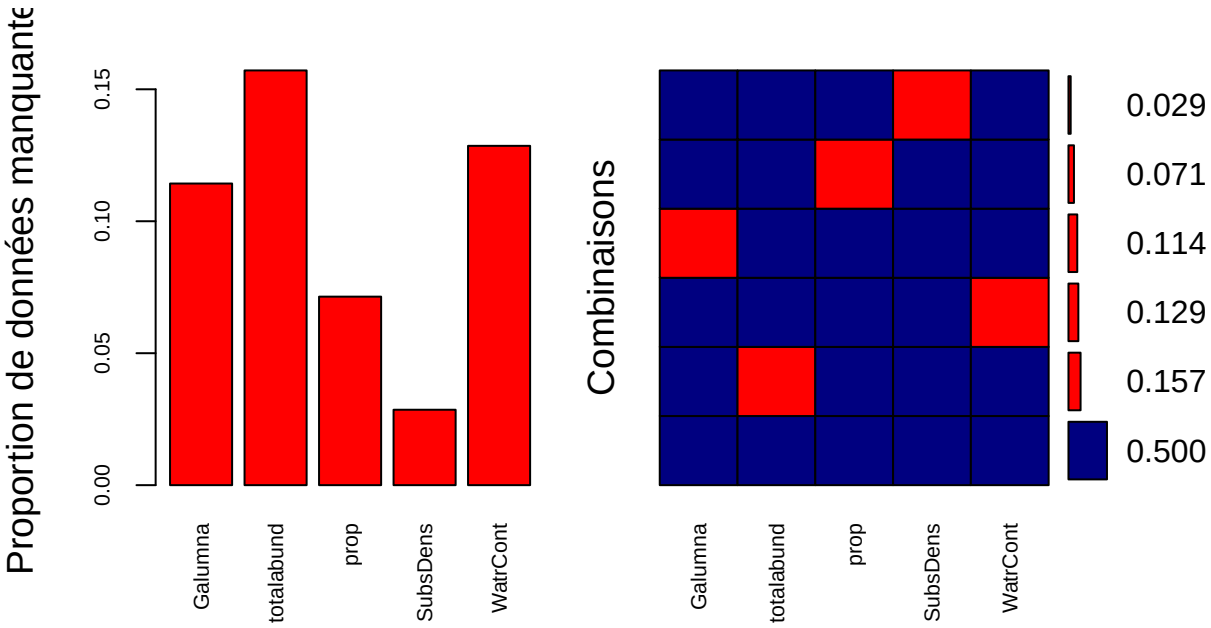


Figure 49: MAR

Pour le dispositif MAR, on observe que les variables `Galumna` et `totalabund` ont le plus de données manquantes, soit plus de 12% de données manquantes chacunes en comparaison avec la variable `SubsDens` qui a seulement moins de 6% de données manquantes. L'analyse bivariée sera donc plus intéressante dans ce cas.

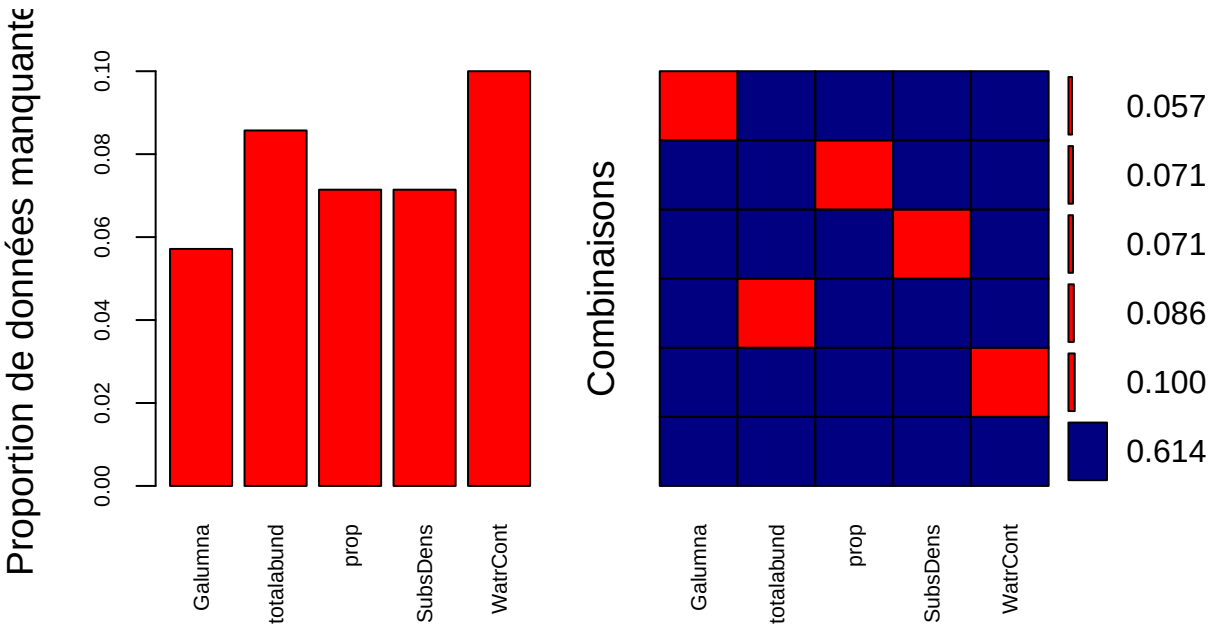


Figure 50: MNAR

Dans ce dispositif, MNAR, on remarque que la proportion de données manquantes pour chaque variable est assez similaire. Nous avons pour chaque variable, une proportion de + ou - 10% de données manquantes.

Analyse bivariée

Pour l'analyse bivariée, nous allons utiliser la fonction `CramerV` du package `DescTools`. Le test de Cramer est utilisé pour mesurer l'intensité des relations entre les variables. Nous avons ainsi que plus le V de Cramer se rapproche de 1, plus l'intensité de la relation est forte. Nous utilisons la fonction `table` dans un premier temps pour créer un tableau de contingence entre deux variables que nous avons selectionnée. Il est important de spécifier l'argument `useNA` pour inclure les données manquantes dans notre tableau.

Nous avons vu que dans le dispositif MCAR, les variables `SubsDens` et `prop` sont les deux variables ayant le plus de données manquantes. Nous allons donc voir si ces 2 variables ont une forte relation. Nous allons, de plus, tester sur différentes paires de variables.

Table 23: V de Cramer (MCAR)

SubsDens/Prop	Galumna/Prop	Galumna/SubsDens	WatrCont/totalabund
0.94348	0.8084864	0.952732	0.952732

Table 24: V de Cramer (MAR)

SubsDens/Prop	Galumna/Prop	Galumna/WatrCont	SubsDens/totalabund
0.9871395	0.8939834	0.8565389	0.9819805

Table 25: V de Cramer (MNAR)

SubsDens/Prop	Galumna/Prop	Galumna/WatrCont	SubsDens/totalabund
0.9239588	0.8713478	0.9645061	0.9612692

Le V de Cramer est très proche de 1 pour chaque paire de variables et chaque dispositif, ce qui signifie que les variables ont une forte relation 2 à 2.

Analyse multidimentionnelle

Nous utiliserons la fonction `MCA` du package `FactoMineR` pour l'analyse multidimensionnelle. L'analyse des correspondances multiples est une technique descriptive visant à résumer l'information contenue dans un grand nombre de variables afin de faciliter l'interprétention des corrélations existantes entre ces différentes variables. On cherche à savoir quelles sont les variables corrélées entre elles. L'analyse se concentre sur ses deux premiers axes qui constitueront un bon résumé des variations observables dans le jeu de données. Le nom des variables finissant par "-m" signifie les données manquantes pour cette variable, contrairement au nom des variables se termiant par "-o".

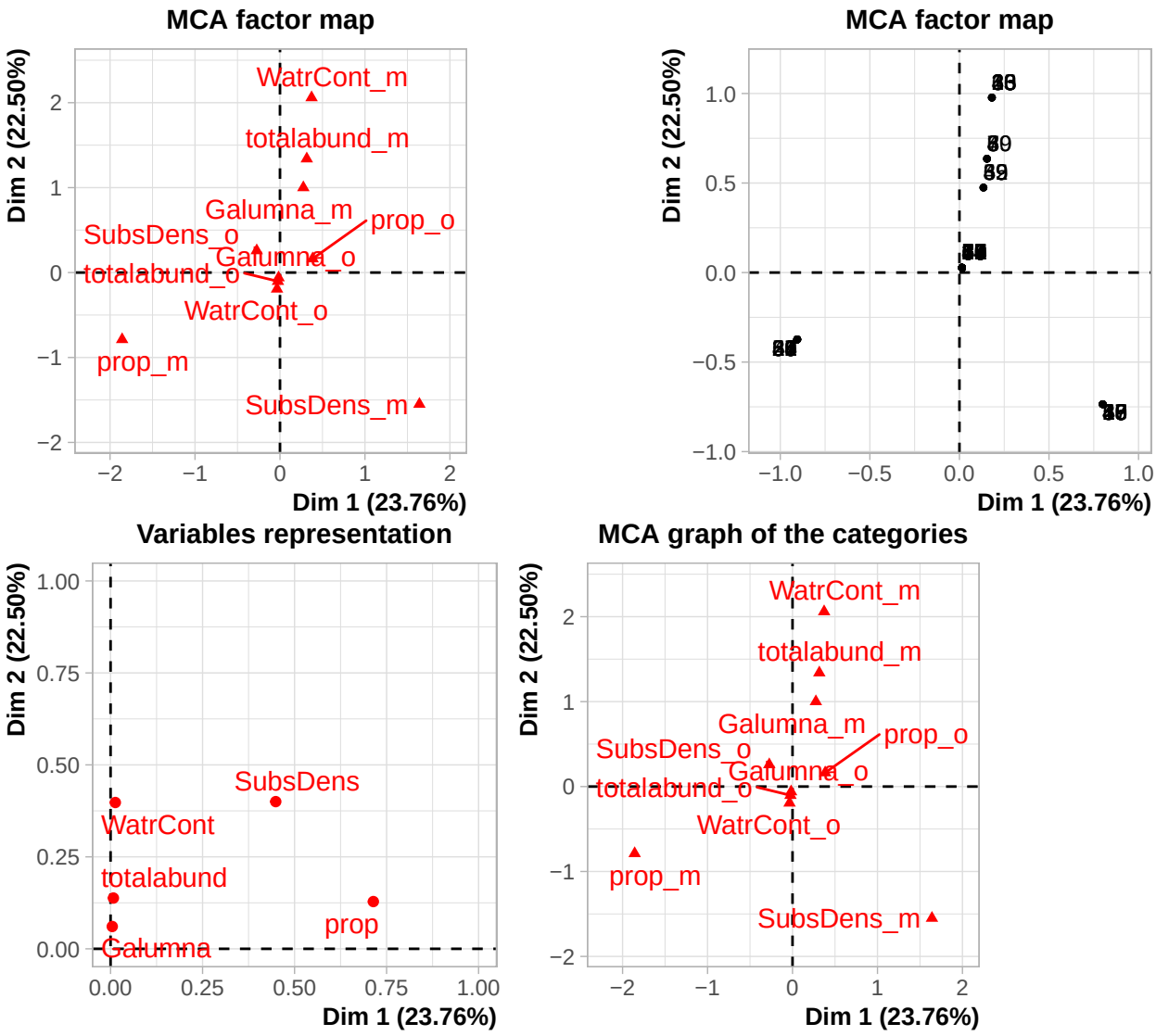


Figure 51: MCAR

```
## Warning: ggrepel: 4 unlabeled data points (too many overlaps). Consider increasing max.overlaps
## ggrepel: 4 unlabeled data points (too many overlaps). Consider increasing max.overlaps
```

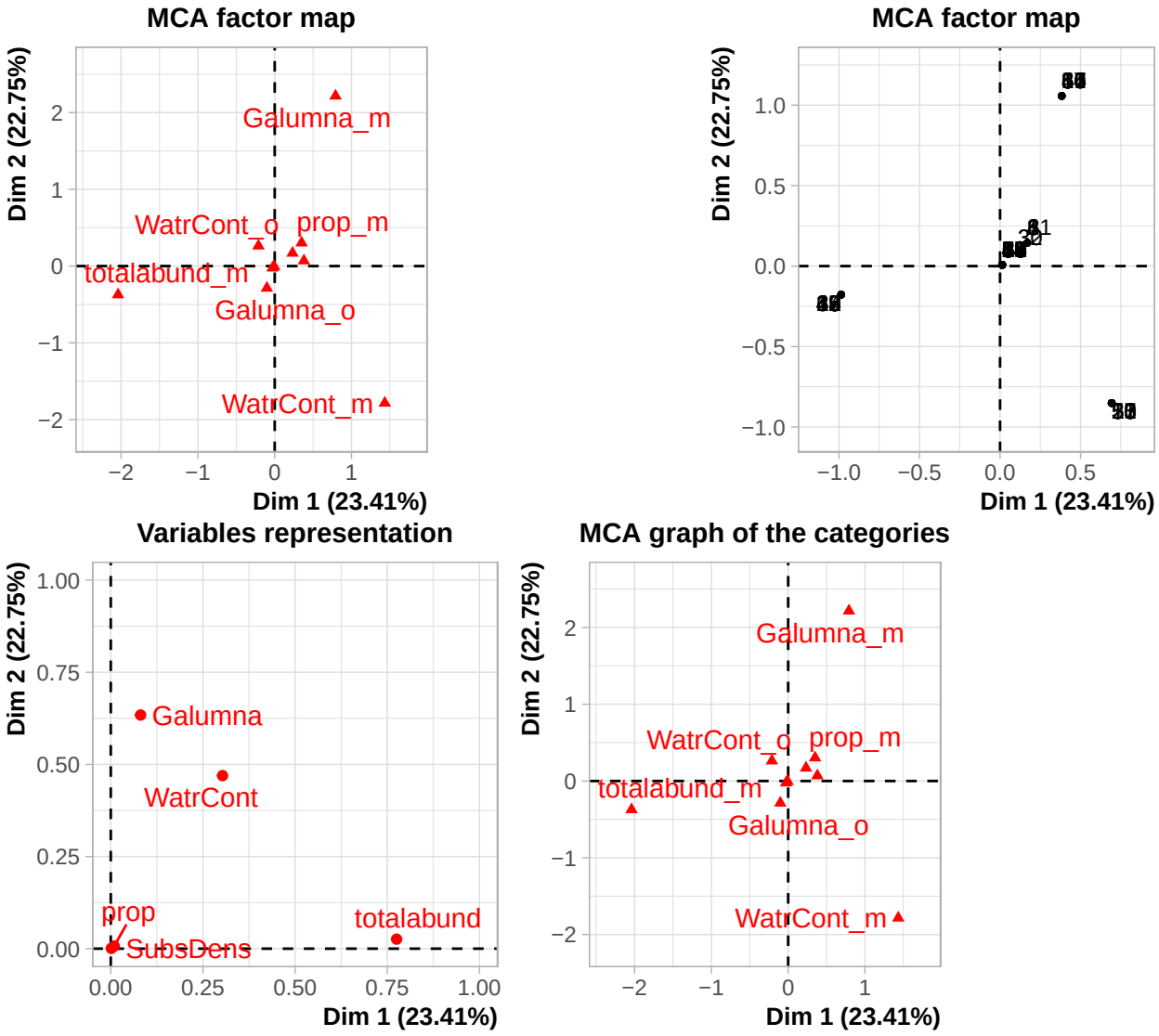


Figure 52: MAR

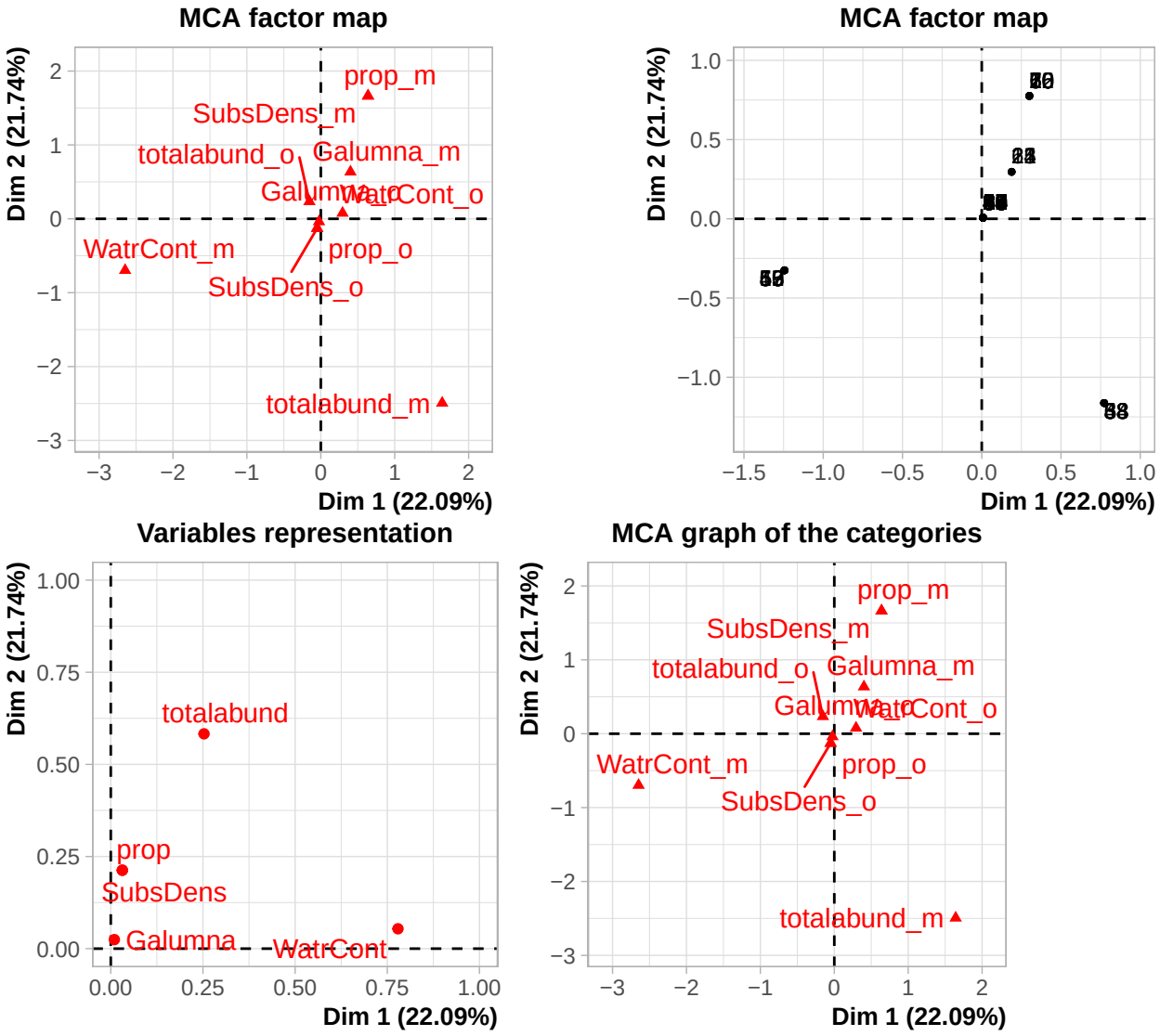


Figure 53: MNAR

2.2.2 Imputer les données manquantes

Nous allons maintenant imputer les données manquantes sur les trois dispositifs que nous avons crées précédement. Dans un premier temps, nous effecturons des imputations simples (par la moyenne, la médiane, méthode LOVF...) puis nous terminerons sur une imputation multiple.

Imputation simple

2.2.2.1 Par la moyenne

Nous utilisons la fonction `impute` du package `Hmisc`.

MCAR

```
set.seed(123)
dat.moy.mcar=impute(amp.mcar$amp, what=mean)
```

MAR

```
set.seed(123)
dat.moy.mar=impute(amp.mar$amp, what=mean)
```

MNAR

```
set.seed(123)
dat.moy.mnar=impute(amp.mnar$amp, what=mean)
```

2.2.2.2 Par la médiane

Nous utilisons aussi la fonction `impute` du package `Hmisc`.

MCAR

```
dat.med.mcar=impute(amp.mcar$amp, what=median)
```

MAR

```
dat.med.mar=impute(amp.mar$amp, what=median)
```

MNAR

```
dat.med.mnar=impute(amp.mnar$amp, what=median)
```

2.2.2.3 LOVF (Last observation carries forward)

Cette méthode remplace la valeur manquante par la dernière valeur disponible. Cependant, cette méthode peut donner des résultats biaisées, même en présence de MCAR. Nous utilisons la fonction `na.locf` du package `zoo`.

MCAR

```
dat.locf=na.locf(amp.mcar$amp,na.rm=FALSE)
dat.locf=na.locf(dat.locf,na.rm=FALSE,
   fromLast=TRUE)
```

MAR

```
dat.locf=na.locf(amp.mar$amp,na.rm=FALSE)
dat.locf=na.locf(dat.locf,na.rm=FALSE,
   fromLast=TRUE)
```

MNAR

```
dat.locf=na.locf(amp.mnar$amp,na.rm=FALSE)
dat.locf=na.locf(dat.locf,na.rm=FALSE,
   fromLast=TRUE)
```

2.2.2.4 k plus proches voisins kNN

L'idée de cette méthode est de calculer les distances entre observations, et d'attribuer aux données manquantes la moyenne des valeurs observées chez les k plus proches voisins. Nous utilisons la fonction `kNN` du package `VIM`.

MCAR

```
dat.kNN=kNN(amp.mcar$amp, k=5, imp_var=FALSE)
```

MAR

```
dat.kNN=kNN(amp.mar$amp, k=5, imp_var=FALSE)
```

MNAR

```
dat.kNN=kNN(amp.mnar$amp, k=5, imp_var=FALSE)
```

2.2.2.5 Loess (Régression locale)

Cette méthode consiste à construire un polynôme de degré faible par méthode des moindres carrés pondérés en fonction de la proximité entre les observations jugées semblables à celle sur laquelle nous voulons procéder à l'imputation.

MCAR

```
dat.imputed=rbind(colnames(amp.mcar$amp),amp.mcar$amp)
indices=1:nrow(amp.mcar$amp)
dat.loess= apply(dat.imputed, 2, function(j) {
predict(locfit(j[-1] ~ indices), indices)
})
head(dat.loess)
```

```
##       Galumna totalabund        prop SubsDens WatrCont
## [1,] 3.120188   216.4955 0.006840352 49.82224 378.4502
## [2,] 3.066104   210.4263 0.007804113 48.76281 364.7685
## [3,] 3.010413   204.5353 0.008729467 47.73602 352.0085
## [4,] 2.953147   198.8246 0.009575387 46.74302 340.1673
## [5,] 2.894339   193.2959 0.010343271 45.78496 329.2421
## [6,] 2.834022   187.9512 0.011034518 44.86298 319.2303
```

Avec cette méthode nous remarquons que les variables `Galumna` et `totalabund` ne sont plus des valeurs entières pour les données manquantes imputées. Nous ne prenons donc pas cette méthode en considération dans notre analyse.

2.2.2.6 missForests

MCAR

```
dat.missForest<-missForest(amp.mcar$amp,maxiter=10,
        ntree = 200, variablewise = TRUE)$ximp
```

```
##   missForest iteration 1 in progress...done!
##   missForest iteration 2 in progress...done!
##   missForest iteration 3 in progress...done!
```

```
head(dat.missForest)
```

```
##   Galumna totalabund        prop SubsDens WatrCont
## 1       8   140.0000 0.019043610   39.180   350.15
## 2       3   268.0000 0.011194030   43.568   434.81
## 3       1   128.0172 0.005376344   46.070   371.72
## 4       1   115.5748 0.003496503   48.190   360.50
## 5       2   199.0000 0.010050251   23.550   204.13
## 6       1   209.0000 0.009069712   57.320   311.55
```

Comme pour la méthode Loess, nous remarquons que les variables `Galumna` et `totalabund` ne sont plus des valeurs entières pour les données manquantes imputées.

A noter que la méthode SVD, décomposition en valeurs singulières, ne fonctionnent pas dans notre cas. Il y a trop de données manquantes, cela introduit un biais important dans le calcul de base de décomposition.

Imputation multiple

Dans l'imputation simple, une valeur unique ne peut pas refléter l'incertitude sur la prédiction. Une autre approche est l'imputation multiple, qui consiste à imputer successivement plusieurs valeurs à chaque donnée manquante. Plusieurs jeux de données complétés sont ainsi générés, respectant les caractéristiques de la distribution des données observées (variabilité et corrélation entre les variables). Des analyses standards sont ensuite menées séparément sur chaque jeu de données complétés, puis leurs résultats sont combinés pour fournir un résultat global.

Nous allons suivre la même démarche que nous avons fait sur les données générées dans la partie précédente toujours en utilisant la fonction `mice`.

- Imputation multiple par régression stochastique

Nous utilisons l'argument `m=20` ce qui signifie que nous obtenons 20 jeu de données imputées.

```
## Warning: Number of logged events: 400
```

```
## Warning: Number of logged events: 100
```

```
## Warning: Number of logged events: 330
```

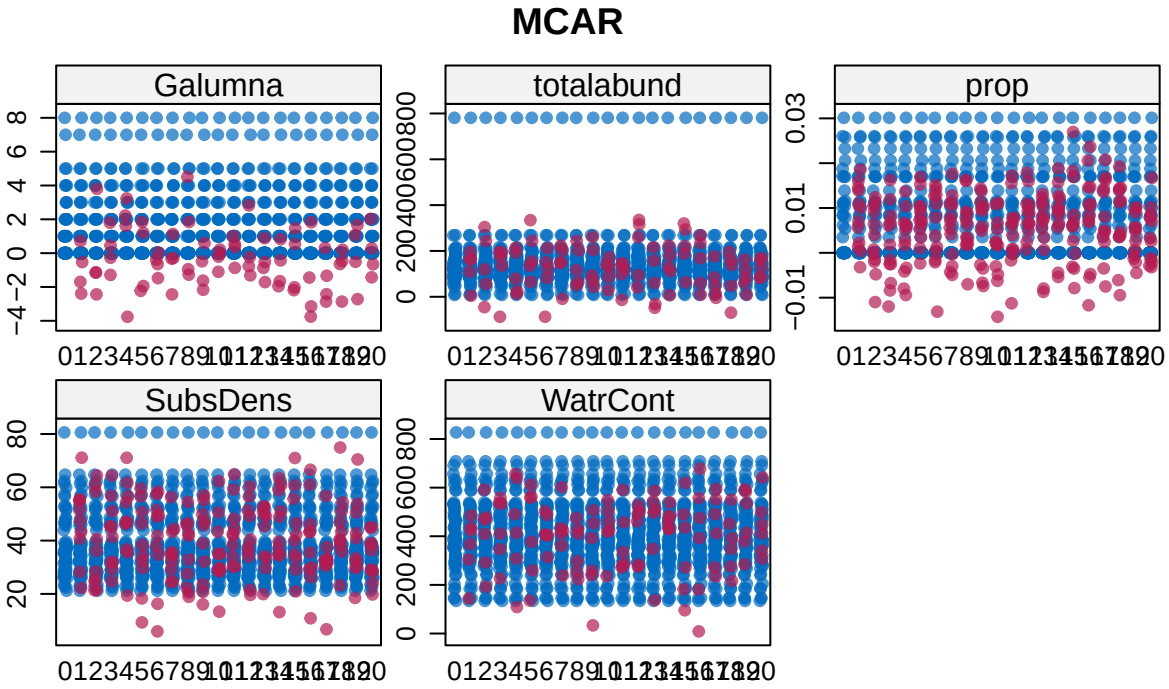


Figure 54: MCAR - Imputation multiple par régression stochastique

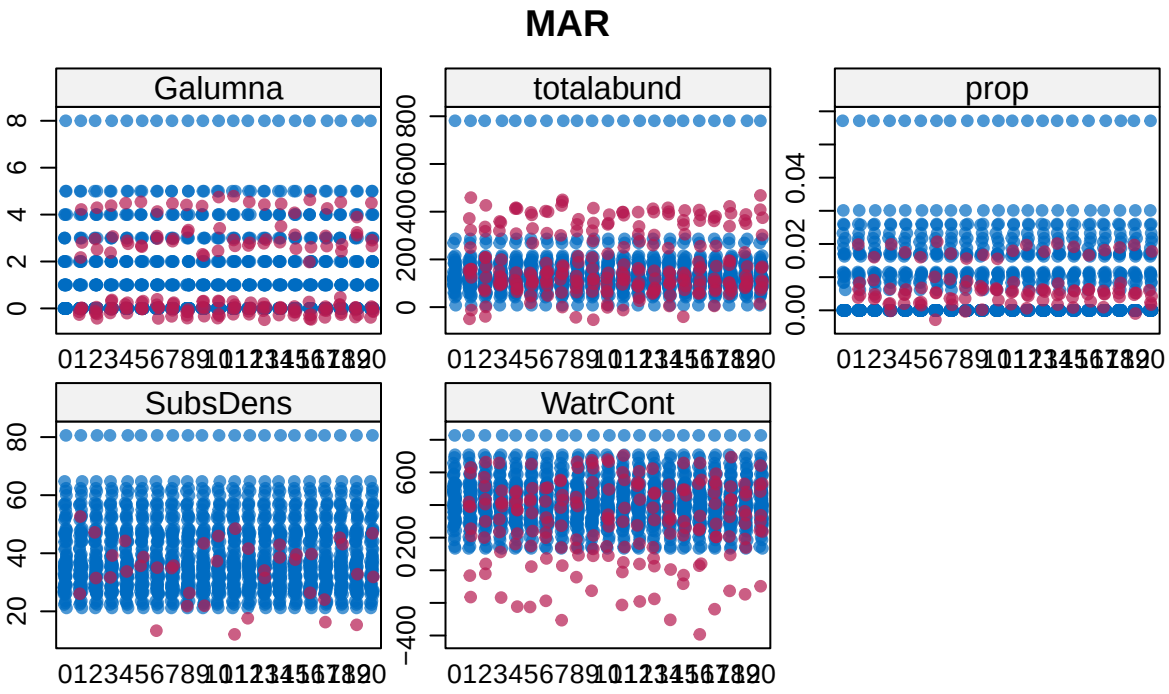


Figure 55: MAR - Imputation multiple par régression stochastique

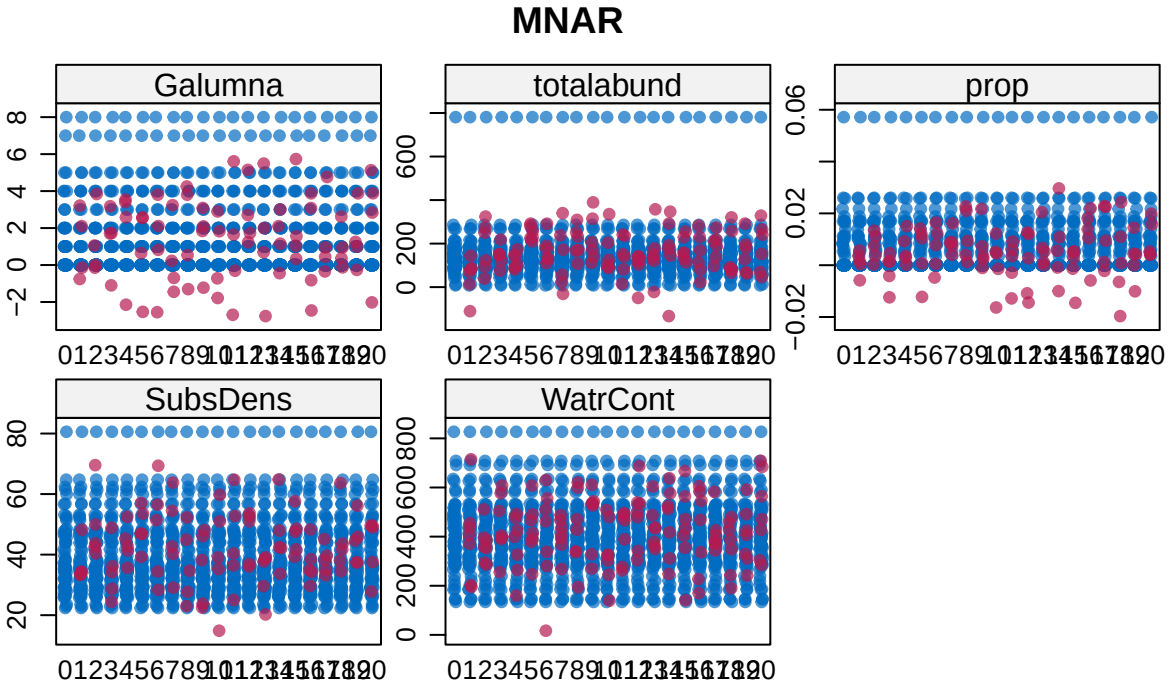


Figure 56: MNAR - Imputation multiple par régression stochastique

Les données imputées sont en rouges. On remarque des différences dans les 3 dispositifs. En effet, dans le dispositif MNAR et plus particulièrement pour la variable `prop`, les données imputées ne respectent pas les données initiales. En comparaison, les dispositifs MCAR et MAR pour la variable `prop` respectent bien la distribution des données initiales. D'un point de vue global, les deux dispositifs MCAR et MAR respectent

assez bien la distribution des données initiales.

Nous allons maintenant, identiquement à la partie précédente, analyse et combiner les données en utilisant les fonctions `with` et `pool` pour récuperer la moyenne estimée.

```
## [1] 0.005764294
```

Table 26: Moyenne de prop après combinaison des 20 imputations par régression stochastique

MCAR	MAR	MNAR
0.0052608	0.0057761	0.0050767

En comparaison avec la moyenne sur la variable `prop` des données initiales du jeu de données `mites`, nous avons que le dispositif MAR et MCAR obtiennent les meilleurs résultats.

```
## [1] 0.9571429
```

Table 27: Moyenne de Galumna après combinaison des 20 imputations par régression stochastique

MCAR	MAR	MNAR
0.9356925	0.9684451	0.9929529

Pour la variable `Galumna`, le resultat n'est pas similaire, dans ce cas le dispositif MNAR a le meilleur résultat.

```
## [1] 140
```

Table 28: Moyenne de totalabund après combinaison des 20 imputations par régression stochastique

MCAR	MAR	MNAR
136.1812	145.1413	140.1408

Pour ces 3 variables différentes, ils n'y a pas de dispositif qui se démarquent des autres. Cependant, le dispositif MAR semble être plutot bon sur chaque variable tandis que par exemple le dispositif MNAR est mauvais pour la variable `prop`. Nous retenons donc le dispositif MAR pour l'imputation multiple par régression stochastique.

Conclusion

Nous avons utilisé le jeu de données `mites` pour générer des données manquantes suivant les 3 catégories MCAR, MAR et MNAR. Nous avons ensuite analyser ces 3 dispositifs dans une analyse univariée, une analyse bivariée et une analyse multidimentionnelle. Pour finir, nous avons imputer les données manquantes avec plusieurs méthodes différentes, de l'imputation simple à l'imputation multiple par régression stochastique.

3 Références bibliographiques

https://delladata.fr/
https://mran.microsoft.com/snapshot/2017-04-24/web/packages/mice/vignettes/ampute.html
https://www.rdocumentation.org/

Bellanger, Lise. 2022. "Cours d'Étude de Cas."

Borcard, D., and P. Legendre. 1994. "Environmental Control and Spatial Structure in Ecological Communities: An Example Using Oribatid Mites (Acari, Oribatei)." *Environmental and Ecological Statistics* 1: 37–61.

Hosmer, D. W., and S. Lemeshow. 2000. "Applied Logistic Regression." *2nd Edition, John Wiley & Sons, Inc., New York*.