Biostatistiques appliquées avec R

Manuel de laboratoire pour le cours BIO4558

Julien Martin

2024-01-29

Table of contents

Préface

Les exercices de laboratoire que vous retrouverez dans les pages qui suivent sont conçus de manière à vous permettre de développer une expérience pratique en analyse de données à l'aide d'un logiciel (R). R est un logiciel très puissant, mais comme tous les logiciels, il a des limites. En particulier il ne peut réfléchir à votre place, vous dire si l'analyse que vous tentez d'effectuer est appropriée ou sensée, ou interpréter biologiquement les résultats.

Quelques points importants à retenir

- Avant de commencer une analyse statistique, il faut d'abord vous familiariser son fonctionnement. Cela ne veut pas dire que vous devez connaître les outils mathématiques qui la sous-tendent, mais vous devriez au moins comprendre les principes utilisés lors de cette analyse. Avant de faire un exercice de laboratoire, lisez donc la section correspondante dans les notes de cours. Sans cette lecture préalable, il est très probable que les résultats produits par le logiciel, même si l'analyse a été effectuée correctement, seront indéchiffrables.

- Les laboratoires sont conçus pour compléter les cours théoriques et vice versa. À cause des contraintes d'horaires, il se pourrait que le cours et le laboratoire ne soient pas parfaitement synchronisés. N'hésitez donc pas à poser des questions sur le labo en classe ou des questions théoriques au laboratoire.

- Travaillez sur les exercices de laboratoire à votre propre rythme. Certains exercices prennent beaucoup moins de temps que d'autres et il n'est pas nécessaire de compléter un exercice par séance de laboratoire. En fait deux séances de laboratoire sont prévues pour certains des exercices. Même si vous n'êtes pas notés sur les exercices de laboratoire, soyez conscient que ces exercices sont essentiels. Si vous ne les faites pas, il est très peu probable que vous serez capable de compléter les devoirs et le projet de session. Prenez donc ces exercices de laboratoire au sérieux !

- Les 2 premier laboratoires sont conçu pour vous permettre d'acquérir ou de réviser le minimum de connaissances requises pour vous permettre de réaliser les exercices de laboratoires avec R. Il y a presque toujours de multiples façons de faire les choses avec R et vous ne trouverez ici que des méthodes simples. Ceux et celles d'entre vous qui y sont enclins pourront trouver en ligne des instructions plus détaillées et complexes. En particulier, je vous conseille :

  - R pour les débutants http://cran.r-project.org/doc/contrib/Paradis-rdebuts_fr.pdf
  - An introduction to R http://cran.r-project.org/doc/manuals/R-intro.html
  - Si vous préférez des manuels, le site web de CRAN en garde une liste commentée à : http://www.r-project.org/doc/bib/R-books.html
  - Une liste impressionnante de très bon livre sur R https://www.bigbookofr.com/
  - Finalement, comme aide-mémoire à garder sous la main, je vous recommande R reference card par Tom Short http://cran.r-project.org/doc/contrib/Short-refcard.pdf

Qu'est-ce que R et pourquoi l'utiliser dans ce cours?

R est un logiciel libre et multi-plateforme formant un système statistique et graphique. R est également un langage de programmation spécialisé pour les statistiques.

R a deux très grands avantages pour ce cours, et un inconvénient embêtant initialement mais qui vous forcera à acquérir des excellentes habitudes de travail. Le premier avantage est que vous pouvez tous l'installer sur votre (ou vos) ordinateurs personnel gratuitement. C'est important parce que c'est à l'usage que vous apprendrez et maîtriserez réellement les biostatistiques et cela implique que vous devez avoir un accès facile et illimité à un logiciel statistique. Le deuxième avantage est que R peut tout faire en statistiques. R est conçu pour être extensible et est devenu l'outil de prédilection des statisticiens mondialement. La question n'est plus : " Est-ce que R peut faire ceci? ", mais devient" Comment faire ceci avec R ". Et la recherche internet est votre ami. Aucun autre logiciel n'offre ces deux avantages.

L'inconvénient embêtant initialement est que l'on doit opérer R en tapant des instructions (ou en copiant des sections de code) plutôt qu'en utilisant des menus et en cliquant sur différentes options. Si on ne sait pas quelle commande taper, rien ne se passe. Ce n'est donc pas facile d'utilisation à priori. Cependant, il est possible d'apprendre rapidement à faire certaines des opérations de base (ouvrir un fichier de données, faire un graphique pour examiner ces données, effectuer un test statistique simple). Et une fois que l'on comprend le principe de la chose, on peut assez facilement trouver sur le web des exemples d'analyses ou de graphiques plus complexes et adapter le code à nos propres besoins. C'est ce que vous ferez dans le premier laboratoire pour vous familiariser avec R.

Pourquoi cet inconvénient est-il d'une certaine façon un avantage? Parce que vous allez sauver du temps en fin de compte. Garanti. Croyez-moi, on ne fait jamais une analyse une seule fois. En cours de route, on découvre des erreurs d'entrée de données, ou que l'on doit faire l'analyse séparément pour des sous-groupes, ou on obtient des données supplémentaires, ou on fait une erreur. On doit alors recommencer l'analyse. Avec une interface graphique et des menus, cela implique recommencer à cliquer ici, entre des paramètres dans des boîtes et sélectionner des boutons. Chaque fois avec possibilité d'erreur. Avec une série de commandes écrites, il suffit de corriger ce qui doit l'être puis de copier-coller l'ensemble pour répéter instantanément. Et vous avez la possibilité de parfaitement documenter ce que vous avez fait. C'est comme cela que les professionnels travaillent et offrent une assurance de qualité de leurs résultats.

Installation des logiciels nécessaires

R

Pour installer R sur un nouvel ordinateur, allez au site http://cran.r-project.org/. Vous y trouverez des versions compilées (binaries) ou non (sources) pour votre système d'exploitation de prédilection (Windows, MacOS, Linux).

Note : R a déjà été installé sur les ordinateurs du laboratoire (la version pourrait être un peu plus ancienne, mais cela devrait être sans conséquences).

Rstudio

RStudio est un environnement de développement intégré (IDE) créé spécifiquement pour travailler avec R. Sa popularité connaît une progression foudroyante depuis 2014. Il permet de consulter dans une interface conviviale ses fichiers de script, la ligne de commande R, les rubriques d'aide, les graphiques, etc.

RStudio est disponible à l'identique pour les plateformes Windows, OS X et Linux. Pour une utilisation locale sur son poste de travail, on installera la version libre (Open Source) de RStudio Desktop depuis le site https://www.rstudio.com/products/rstudio/download/

Paquets pour R

- Rmarkdown
- tinytex

Ces 2 paquets devrait être installé automatiquement avec RStudio, mais pas toujours. Je vous recommande donc de les installer manuellement. Pour ce faire, simplement copier-coller le texte suivant dans le terminal R.

```
install.packages(c("rmarkdown", "tinytex"))
```

G*Power

G*Power est un programme gratuit, développé par des psychologues de l'Université de Dusseldorf en Allemagne. Le programme existe en version Mac et Windows. Il peut cependant être utilisé sous linux via Wine. G*Power vous permettra d'effectuer une analyse de puissance pour la majorité des tests que nous verrons au cours de la session sans avoir à effectuer des calculs complexes ou farfouiller dans des tableaux ou des figures décrivant des distributions ou des courbes de puissance.

Téléchargez le programme sur le site https://www.psychologie.hhu.de/arbeitsgruppen/allgemeine-psychologie-und-arbeitspsychologie/gpower.html

Instructions générales pour les laboratoires

- Apporter une clé USB ou son équivalent à chaque séance de laboratoire pour sauvegarder votre travail.
- Lire l'exercice de laboratoire AVANT la séance, lire le code R correspondant et préparer vos questions sur le code.
- Durant les pré-labs, écouter les instructions et posez vos questions au moment approprié.
- Faites les exercices du manuel de laboratoire à votre rythme, en équipe, puis je vous recommande de commencer (compléter?) le devoir. Profitez de la présence du démonstrateur et du prof…
- Pendant vos analyses, copiez-collez des fragments de sorties de R dans un document (par exemple dans votre traitement de texte favori) et annotez abondamment.
- Ne tapez pas directement vos commandes dans R mais plutôt dans un script. Vous pourrez ainsi refaire le labo instantanément, récupérer des fragments de code, ou plus facilement identifier les erreurs dans vos analyses.
- Créez votre propre librairie de fragments de codes (snippets). Annotez-là abondamment. Vous vous en féliciterez plus tard.

Notes sur le manuel

Vous trouverez dans le manuel des explications sur la théorie, du code R, des explications sur R et des exercises.

Le manuel essaie aussi de mettre en évidence le texte de différentes manières.

Exercice

Avec des sections à vous de jouer, ui indique un exercise à faire, idéalement sans regarder la solution qui se trouve plus bas.

Solution

aves les réponses

Warning

des avertissements

Important

des points importants

Note

des notes

Licence

Figure 1.: Licence Creative Commons

1. Introduction à R

Après avoir complété cet exercice de laboratoire, vous pourrez :

- Ouvrir des fichiers de données R déjà existants
- Importer des ensembles de données rectangulaires
- Exporter des donnes de R vers un fichier texte
- Vérifier si les données ont été correctement importées
- Examiner la distribution des observations d'une variable
- Examiner visuellement et tester la normalité d'une variable
- Calculer des statistiques descriptives d'une variable
- Effectuer des transformations de données

1.1. Paquets et données requises pour le labo

Ce laboratoire nécessite:

- les paquets R:
  - questionr
  - ggplot2
- les fichiers de données
  - ErablesGatineau.csv
  - sturgeon.csv

1.2. Premier pas avec R

Une fois R et RStudio installés sur votre machine, nous n'allons pas lancer R mais plutôt RStudio.

RStudio n'est pas à proprement parler une interface graphique qui permettrait d'utiliser R de manière "classique" via la souris, des menus et des boîtes de dialogue. Il s'agit plutôt de ce qu'on appelle un *Environnement de développement intégré* (IDE) qui facilite l'utilisation de R et le développement de scripts.

1.2.1. La console

1.2.1.1. L'invite de commandes

Au premier lancement de RStudio, l'écran principal est découpé en trois grandes zones :

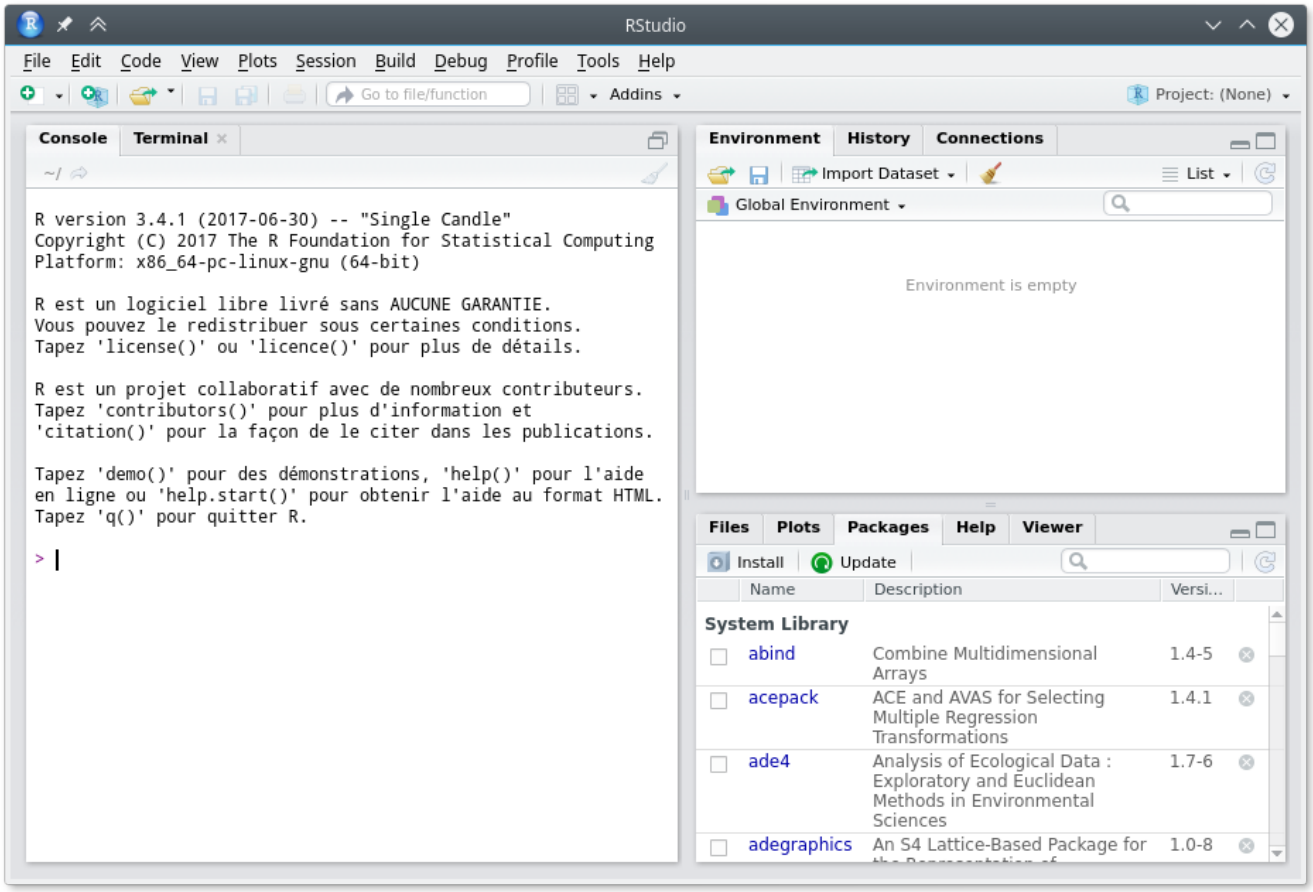


Figure 1.1.: Interface de RStudio

La zone de gauche se nomme *Console*. À son démarrage, RStudio a lancé une nouvelle session de R et c'est dans cette fenêtre que nous allons pouvoir interagir avec lui.

La *Console* doit normalement afficher un texte de bienvenue ressemblant à ceci :

```
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.
```

suivi d'une ligne commençant par le caractère `>` et sur laquelle devrait se trouver votre curseur. Cette ligne est appelée l'*invite de commande* (ou *prompt* en anglais). Elle signifie que R est disponible et en attente de votre prochaine commande.

Nous pouvons tout de suite lui fournir une première commande, en saisissant le texte suivant puis en appuyant sur `Entrée` :

```
2 + 2
```

```
[1] 4
```

R nous répond immédiatement, et nous pouvons constater avec soulagement qu'il sait faire des additions à un chiffre[1]. On peut donc continuer avec d'autres opérations :

```
5 - 7
```

```
[1] -2
```

```
4 * 12
```

```
[1] 48
```

```
-10 / 3
```

```
[1] -3.333333
```

```
5^2
```

```
[1] 25
```

Cette dernière opération utilise le symbole `^` qui représente l'opération *puissance*. `5^2` signifie donc "5 au carré", soit 25.

1.2.1.2. Précisions concernant la saisie des commandes

Lorsqu'on saisit une commande, les espaces autour des opérateurs n'ont pas d'importance. Les trois commandes suivantes sont donc équivalentes, mais on privilégie en général la deuxième pour des raisons de lisibilité du code.

[1] On peut ignorer pour le moment la présence du `[1]` en début de ligne.

```
10+2
10 + 2
10       +       2
```

Quand vous êtes dans la console, vous pouvez utiliser les flèches vers le haut et vers le bas pour naviguer dans l'historique des commandes que vous avez tapées précédemment. Vous pouvez à tout moment modifier la commande affichée, et l'exécuter en appuyant sur `Entrée`.

Enfin, il peut arriver qu'on saisisse une commande de manière incomplète : oubli d'une parenthèse, faute de frappe, etc. Dans ce cas, R remplace l'invite de commande habituel par un signe `+` :

```
4 *
+
```

Cela signifie qu'il "attend la suite". On peut alors soit compléter la commande sur cette nouvelle ligne et appuyer sur `Entrée`, soit, si on est perdu, tout annuler et revenir à l'invite de commandes normal en appuyant sur `Esc` ou `Échap`.

1.2.2. Objets

1.2.2.1. Objets simples

Faire des calculs c'est bien, mais il serait intéressant de pouvoir stocker un résultat quelque part pour pouvoir le réutiliser ultérieurement sans avoir à faire du copier/coller.

Pour conserver le résultat d'une opération, on peut le stocker dans un *objet* à l'aide de l'opérateur d'assignation `<-`. Cette "flèche" stocke ce qu'il y a à sa droite dans un objet dont le nom est indiqué à sa gauche.

Prenons tout de suite un exemple :

```
x <- 2
```

Cette commande peut se lire *"prend la valeur 2 et mets la dans un objet qui s'appelle* **`x`***"*.

Si on exécute une commande comportant juste le nom d'un objet, R affiche son contenu :

```
x
```

```
[1] 2
```

On voit donc que notre objet `x` contient bien la valeur 2.

On peut évidemment réutiliser cet objet dans d'autres opérations. R le remplacera alors par sa valeur :

```
x + 4
```

```
[1] 6
```

On peut créer autant d'objets qu'on le souhaite.

```
x <- 2
y <- 5
resultat <- x + y
resultat
```

```
[1] 7
```

> **!** Important
>
> Les noms d'objets peuvent contenir des lettres, des chiffres, les symboles `.` et `_`. Ils ne peuvent pas commencer par un chiffre. Attention, R fait la différence entre minuscules et majuscules dans les noms d'objets, ce qui signifie que `x` et `X` seront deux objets différents, tout comme `resultat` et `Resultat`.
> De manière générale, il est préférable d'éviter les majuscules (pour les risques d'erreur) et les caractères accentués (pour des questions d'encodage) dans les noms d'objets.
> De même, il faut essayer de trouver un équilibre entre clarté du nom (comprendre à quoi sert l'objet, ce qu'il contient) et sa longueur. Par exemple, on préfèrera comme nom d'objet `taille_conj1` à `taille_du_conjoint_numero_1` (trop long) ou à `t1` (pas assez explicite).

Quand on assigne une nouvelle valeur à un objet déjà existant, la valeur précédente est perdue. Les objets n'ont pas de mémoire.

```
x <- 2
x <- 5
x
```

```
[1] 5
```

De la même manière, assigner un objet à un autre ne crée pas de "lien" entre les deux. Cela copie juste la valeur de l'objet de droite dans celui de gauche :

```
x <- 1
y <- 3
x <- y
x
```

```
[1] 3
```

```
## Si on modifie y, cela ne modifie pas x
y <- 4
x
```

```
[1] 3
```

On le verra, les objets peuvent contenir tout un tas d'informations. Jusqu'ici on n'a stocké que des nombres, mais ils peuvent aussi contenir des chaînes de caractères (du texte), qu'on délimite avec des guillemets simples ou doubles (' ou ") :

```
chien <- "Chihuahua"
chien
```

```
[1] "Chihuahua"
```

1.2.3. Vecteurs

Imaginons maintenant qu'on a demandé la taille en centimètres de 5 personnes et qu'on souhaite calculer leur taille moyenne. On pourrait créer autant d'objets que de tailles et faire l'opération mathématique qui va bien :

```
taille1 <- 156
taille2 <- 164
taille3 <- 197
taille4 <- 147
taille5 <- 173
(taille1 + taille2 + taille3 + taille4 + taille5) / 5
```

```
[1] 167.4
```

Cette manière de faire n'est évidemment pas pratique du tout. On va plutôt stocker l'ensemble de nos tailles dans un seul objet, de type *vecteur*, avec la syntaxe suivante :

```
tailles <- c(156, 164, 197, 147, 173)
```

Si on affiche le contenu de cet objet, on voit qu'il contient bien l'ensemble des tailles saisies :

```
tailles
```

```
[1] 156 164 197 147 173
```

Un *vecteur* dans R est un objet qui peut contenir plusieurs informations du même type, potentiellement en très grand nombre.

L'avantage d'un vecteur est que lorsqu'on lui applique une opération, celle-ci s'applique à toutes les valeurs qu'il contient. Ainsi, si on veut la taille en mètres plutôt qu'en centimètres, on peut faire :

```
tailles_m <- tailles / 100
tailles_m
```

```
[1] 1.56 1.64 1.97 1.47 1.73
```

Cela fonctionne pour toutes les opérations de base :

```
tailles + 10
```

```
[1] 166 174 207 157 183
```

```
tailles^2
```

```
[1] 24336 26896 38809 21609 29929
```

Imaginons maintenant qu'on a aussi demandé aux cinq mêmes personnes leur poids en kilos. On peut alors créer un deuxième vecteur :

```
poids <- c(45, 59, 110, 44, 88)
```

On peut alors effectuer des calculs utilisant nos deux vecteurs `tailles` et `poids`. On peut par exemple calculer l'indice de masse corporelle (IMC) de chacun de nos enquêtés en divisant leur poids en kilo par leur taille en mètre au carré :

```
imc <- poids / (tailles / 100) ^ 2
imc
```

```
[1] 18.49112 21.93635 28.34394 20.36189 29.40292
```

Un vecteur peut contenir des nombres, mais il peut aussi contenir du texte. Imaginons qu'on a demandé aux 5 mêmes personnes leur niveau de diplôme : on peut regrouper l'information dans un vecteur de *chaînes de caractères*. Une chaîne de caractère contient du texte libre, délimité par des guillemets simples ou doubles :

```
diplome <- c("PHD", "Bac", "MSc", "MSc", "Bac")
diplome
```

```
[1] "PHD" "Bac" "MSc" "MSc" "Bac"
```

L'opérateur `:`, lui, permet de générer rapidement un vecteur comprenant tous les nombres entre deux valeurs, opération assez courante sous R :

```
x <- 1:10
x
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

Enfin, notons qu'on peut accéder à un élément particulier d'un vecteur en faisant suivre le nom du vecteur de crochets contenant le numéro de l'élément désiré. Par exemple :

```
diplome[2]
```

```
[1] "Bac"
```

Cette opération, qui utilise l'opérateur `[]`, permet donc la sélection d'éléments d'un vecteur.

Dernière remarque, si on affiche dans la console un vecteur avec beaucoup d'éléments, ceux-ci seront répartis sur plusieurs lignes. Par exemple, si on a un vecteur de 50 nombres on peut obtenir quelque chose comme :

```
 [1] 294 425 339 914 114 896 716 648 915 587 181 926 489
[14] 848 583 182 662 888 417 133 146 322 400 698 506 944
[27] 237 324 333 443 487 658 793 288 897 588 697 439 697
[40] 914 694 126 969 744 927 337 439 226 704 635
```

On remarque que R ajoute systématiquement un nombre entre crochets au début de chaque ligne : il s'agit en fait de la position du premier élément de la ligne dans le vecteur. Ainsi, le 848 de la deuxième ligne est le 14e élément du vecteur, le 914 de la dernière ligne est le 40e, etc.

Ceci explique le `[1]` qu'on obtient quand on affiche un simple nombre et permet de constater que pour R, un nombre est un vecteur à un seul élément :

```
[1] 4
```

1.2.4. Fonctions

1.2.4.1. Principe

Nous savons désormais effectuer des opérations arithmétiques de base sur des nombres et des vecteurs, et stocker des valeurs dans des objets pour pouvoir les réutiliser plus tard.

Pour aller plus loin, nous devons aborder les *fonctions* qui sont, avec les objets, un deuxième concept de base de R. On utilise des fonctions pour effectuer des calculs, obtenir des résultats et accomplir des actions.

Formellement, une fonction a un *nom*, elle prend en entrée entre parenthèses un ou plusieurs *arguments* (ou *paramètres*), et retourne un *résultat*.

Prenons tout de suite un exemple. Si on veut connaître le nombre d'éléments du vecteur `tailles` que nous avons construit précédemment, on peut utiliser la fonction `length`, de cette manière :

```
length(tailles)
```

```
[1] 5
```

Ici, `length` est le nom de la fonction, on l'appelle en lui passant un argument entre parenthèses (en l'occurrence notre vecteur `tailles`), et elle nous renvoie un résultat, à savoir le nombre d'éléments du vecteur passé en paramètre.

Autre exemple, les fonctions `min` et `max` retournent respectivement les valeurs minimales et maximales d'un vecteur de nombres :

```
min(tailles)
```

```
[1] 147
```

```
max(tailles)
```

```
[1] 197
```

La fonction `mean` calcule et retourne la moyenne d'un vecteur de nombres :

```
mean(tailles)
```

```
[1] 167.4
```

La fonction `sum` retourne la somme de tous les éléments du vecteur :

```
sum(tailles)
```

```
[1] 837
```

Jusqu'à présent on n'a vu que des fonctions qui calculent et retournent un unique nombre. Mais une fonction peut renvoyer d'autres types de résultats. Par exemple, la fonction `range` (étendue) renvoie un vecteur de deux nombres, le minimum et le maximum :

```
range(tailles)
```

```
[1] 147 197
```

Ou encore, la fonction `unique`, qui supprime toutes les valeurs en double dans un vecteur, qu'il s'agisse de nombres ou de chaînes de caractères :

```
diplome
```

```
[1] "PHD" "Bac" "MSc" "MSc" "Bac"
```

```
unique(diplome)
```

```
[1] "PHD" "Bac" "MSc"
```

1.2.4.2. Arguments

Une fonction peut prendre plusieurs arguments, dans ce cas on les indique toujours entre parenthèses, séparés par des virgules.

On a déjà rencontré un exemple de fonction acceptant plusieurs arguments : la fonction `c`, qui combine l'ensemble de ses arguments en un vecteur[2] :

```
tailles <- c(156, 164, 197, 181, 173)
```

Ici, `c` est appelée en lui passant cinq arguments, les cinq tailles séparées par des virgules, et elle renvoie un vecteur numérique regroupant ces cinq valeurs.

Supposons maintenant que dans notre vecteur `tailles` nous avons une valeur manquante (une personne a refusé de répondre, ou notre mètre mesureur était en panne). On symbolise celle-ci dans R avec le code interne `NA` :

[2] `c` est l'abbréviation de *combine*, son nom est très court car on l'utilise très souvent

```
tailles <- c(156, 164, 197, NA, 173)
tailles
```

```
[1] 156 164 197  NA 173
```

> **Note**
>
> `NA` est l'abbréviation de *Not available*, non disponible. Cette valeur particulière peut être utilisée pour indiquer une valeur manquante, qu'il s'agisse d'un nombre, d'une chaîne de caractères, etc.

Si je calcule maintenant la taille moyenne à l'aide de la fonction `mean`, j'obtiens :

```
mean(tailles)
```

```
[1] NA
```

En effet, R considère par défaut qu'il ne peut pas calculer la moyenne si une des valeurs n'est pas disponible. Il considère alors que cette moyenne est elle-même "non disponible" et renvoie donc comme résultat `NA`.

On peut cependant indiquer à `mean` d'effectuer le calcul en ignorant les valeurs manquantes. Ceci se fait en ajoutant un argument supplémentaire, nommé `na.rm` (abbréviation de *NA remove*, "enlever les NA"), et de lui attribuer la valeur `TRUE` (code interne de R signifiant *vrai*) :

```
mean(tailles, na.rm = TRUE)
```

```
[1] 172.5
```

Positionner le paramètre `na.rm` à `TRUE` indique à la fonction `mean` de ne pas tenir compte des valeurs manquantes dans le calcul.

Si on ne dit rien à la fonction `mean`, cet argument a une valeur par défaut, en l'occurrence `FALSE` (faux), qui fait qu'il ne supprime pas les valeurs manquantes. Les deux commandes suivantes sont donc rigoureusement équivalentes :

```
mean(tailles)
```

```
[1] NA
```

```
mean(tailles, na.rm = FALSE)
```

```
[1] NA
```

> **Note**
>
> Lorsqu'on passe un argument à une fonction de cette manière, c'est-à-dire sous la forme `nom = valeur`, on parle d'*argument nommé*.

1.2.4.3. Aide sur une fonction

Il est fréquent de ne pas savoir (ou d'avoir oublié) quels sont les arguments d'une fonction, ou comment ils se nomment. On peut à tout moment faire appel à l'aide intégrée à R en passant le nom de la fonction (entre guillemets) à la fonction `help` :

```
help("mean")
```

On peut aussi utiliser le raccourci `?mean`.

Ces deux commandes affichent une page (en anglais) décrivant la fonction, ses paramètres, son résultat, le tout accompagné de diverses notes, références et exemples. Ces pages d'aide contiennent à peu près tout ce que vous pourrez chercher à savoir, mais elles ne sont pas toujours d'une lecture aisée.

Dans RStudio, les pages d'aide en ligne s'ouvriront par défaut dans la zone en bas à droite, sous l'onglet *Help*. Un clic sur l'icône en forme de maison vous affichera la page d'accueil de l'aide.

1.2.5. Regrouper ses commandes dans des scripts

Jusqu'ici on a utilisé R de manière "interactive", en saisissant des commandes directement dans la console. Ça n'est cependant pas la manière dont on va utiliser R au quotidien, pour une raison simple : lorsque R redémarre, tout ce qui a été effectué dans la console est perdu.

Plutôt que de saisir nos commandes dans la console, on va donc les regrouper dans des scripts (de simples fichiers texte), qui vont garder une trace de toutes les opérations effectuées, et ce sont ces scripts, sauvegardés régulièrement, qui seront le "coeur" de notre travail. C'est en rouvrant les scripts et en réexécutant les commandes qu'ils contiennent qu'on pourra "reproduire" les données, leur traitement, les analyses et leurs résultats.

Pour créer un script, il suffit de sélectionner le menu *File*, puis *New file* et *R script*. Une quatrième zone apparaît alors en haut à gauche de l'interface de RStudio. On peut enregistrer notre script à tout moment dans un fichier avec l'extension `.R`, en cliquant sur l'icône de disquette ou en choissant *File* puis *Save*.

Un script est un fichier texte brut, qui s'édite de la manière habituelle. À la différence de la console, quand on appuie sur `Entrée`, cela n'exécute pas la commande en cours mais insère un saut de ligne (comme on pouvait s'y attendre).

Pour exécuter une commande saisie dans un script, il suffit de positionner le curseur sur la ligne de la commande en question, et de cliquer sur le bouton *Run* dans la barre d'outils juste au-dessus de la zone d'édition du script. On peut aussi utiliser le raccourci clavier `Ctrl + Entrée` (`Cmd + Entrée` sous Mac). On peut enfin sélectionner plusieurs lignes avec la souris ou le clavier et cliquer sur *Run* (ou utiliser le raccourci clavier), et l'ensemble des lignes est exécuté d'un coup.

Au final, un script pourra ressembler à quelque chose comme ça :

```
tailles <- c(156, 164, 197, 147, 173)
poids <- c(45, 59, 110, 44, 88)

mean(tailles)
mean(poids)

imc <- poids / (tailles / 100) ^ 2
min(imc)
max(imc)
```

1.2.5.1. Commentaires

Les commentaires sont un élément très important d'un script. Il s'agit de texte libre, ignoré par R, et qui permet de décrire les étapes du script, sa logique, les raisons pour lesquelles on a procédé de telle ou telle manière… Il est primordial de documenter ses scripts à l'aide de commentaires, car il est très facile de ne plus se retrouver dans un programme qu'on a produit soi-même, même après une courte interruption.

Pour ajouter un commentaire, il suffit de le faire précéder d'un ou plusieurs symboles `#`. En effet, dès que R rencontre ce caractère, il ignore tout ce qui se trouve derrière, jussqu'à la fin de la ligne.

On peut donc documenter le script précédent :

```
# Saisie des tailles et poids des enquêtés
tailles <- c(156, 164, 197, 147, 173)
poids <- c(45, 59, 110, 44, 88)

# Calcul des tailles et poids moyens
mean(tailles)
mean(poids)

# Calcul de l'IMC (poids en kilo divisé par les tailles en mètre au carré)
imc <- poids / (tailles / 100) ^ 2
# Valeurs extrêmes de l'IMC
min(imc)
max(imc)
```

1.2.6. Installer et charger des extensions (*packages*)

R étant un logiciel libre, il bénéficie d'un développement communautaire riche et dynamique. L'installation de base de R permet de faire énormément de choses, mais le langage dispose en plus d'un système d'extensions permettant d'ajouter facilement de nouvelles fonctionnalités. La plupart des extensions sont développées et maintenues par la communauté des utilisateurs de R, et diffusées via un réseau de serveurs nommé CRAN (*Comprehensive R Archive Network*).

Pour installer une extension, si on dispose d'une connexion Internet, on peut utiliser le bouton *Install* de l'onglet *Packages* de RStudio.

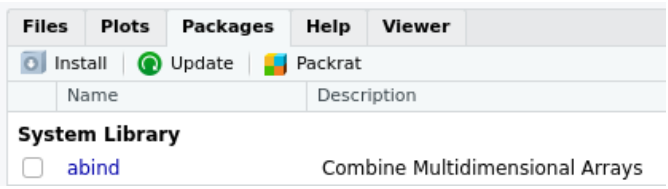


Figure 1.2.: Installer une extension

Il suffit alors d'indiquer le nom de l'extension dans le champ *Package* et de cliquer sur *Install*.

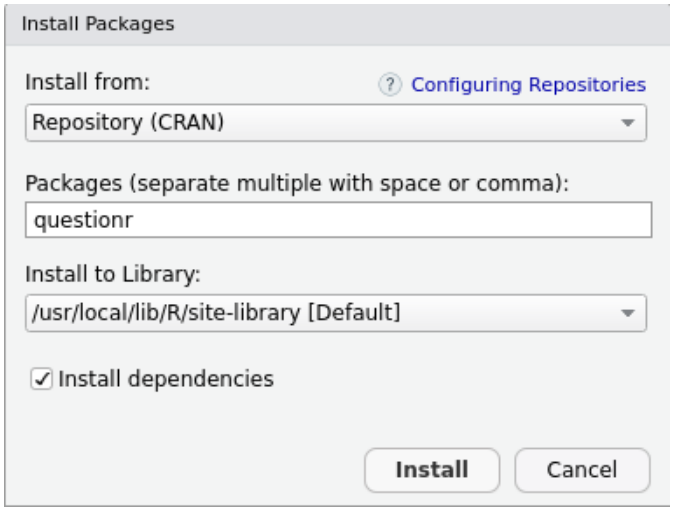


Figure 1.3.: Installation une extension

images/screenshots/rstudio_package_install.png On peut aussi installer des extensions en utilisant la fonction `install.packages()` directement dans la console. Par exemple, pour installer le *package* `questionr` on peut exécuter la commande :

```
install.packages("questionr")
```

Installer une extension via l'une des deux méthodes précédentes va télécharger l'ensemble des fichiers nécessaires depuis l'une des machines du CRAN, puis installer tout ça sur le disque dur de votre ordinateur. Vous n'avez besoin de le faire qu'une fois, comme vous le faites pour installer un programme sur votre Mac ou PC.

Une fois l'extension installée, il faut la "charger" avant de pouvoir utiliser les fonctions qu'elle propose. Ceci se fait avec la fonction `library`. Par exemple, pour pouvoir utiliser les fonctions de `questionr`, vous devrez exécuter la commande suivante :

```
library(questionr)
```

Ainsi, bien souvent, on regroupe en début de script toute une série d'appels à `library` qui permettent de charger tous les packages utilisés dans le script. Quelque chose comme :

```
library(readxl)
library(ggplot2)
library(questionr)
```

Si vous essayez d'exécuter une fonction d'une extension et que vous obtenez le message d'erreur `impossible de trouver la fonction`, c'est certainement parce que vous n'avez pas exécuté la commande `library` correspondante.

1.2.7. Exercices

1.2.7.1. Exercice 1

Construire le vecteur `x` suivant :

```
[1] 120 134 256  12
```

Solution

```
x <- c(120, 134, 256, 12)
```

Utiliser ce vecteur `x` pour générer les deux vecteurs suivants :

```
[1] 220 234 356 112
```

```
[1] 240 268 512  24
```

Solution

```
x + 100
x * 2
```

1.2.7.2. Exercice 2

On a demandé à 4 ménages le revenu des deux conjoints, et le nombre de personnes du ménage :

```
conjoint1 <- c(1200, 1180, 1750, 2100)
conjoint2 <- c(1450, 1870, 1690, 0)
nb_personnes <- c(4, 2, 3, 2)
```

Calculer le revenu total de chaque ménage, puis diviser par le nombre de personnes pour obtenir le revenu par personne de chaque ménage.

Solution

```
revenu_total <- conjoint1 + conjoint2
revenu_total / nb_personnes
```

1.2.7.3. Exercice 3

Dans l'exercice précédent, calculer le revenu minimum et maximum parmi ceux du premier conjoint.

```
conjoint1 <- c(1200, 1180, 1750, 2100)
```

Solution

```
range(conjoint1)
```

Recommencer avec les revenus suivants, parmi lesquels l'un des enquetés n'a pas voulu répondre :

```
conjoint1 <- c(1200, 1180, 1750, NA)
```

Solution

```
range(conjoint1, na.rm = TRUE)
```

1.2.7.4. Exercice 4

Les deux vecteurs suivants représentent les précipitations (en mm) et la température (en °C) moyennes sur la ville de Lyon, pour chaque mois de l'année, entre 1981 et 2010 :

```
temperature <- c(3.4, 4.8, 8.4, 11.4, 15.8, 19.4, 22.2, 21.6, 17.6, 13.4, 7.6, 4.4)
precipitations <- c(47.2, 44.1, 50.4, 74.9, 90.8, 75.6, 63.7, 62, 87.5, 98.6, 81.9, 55.2)
```

Calculer la température moyenne sur l'année.

Calculer la quantité totale de précipitations sur l'année.

Solution

```
mean(temperature)
sum(precipitations)
```

À quoi correspond et comment peut-on interpréter le résultat de la fonction suivante ? Vous pouvez vous aider de la page d'aide de la fonction si nécessaire.

```
cumsum(precipitations)
```

```
 [1]  47.2  91.3 141.7 216.6 307.4 383.0 446.7 508.7 596.2 694.8 776.7 831.9
```

Même question pour :

```
diff(temperature)
```

```
 [1]  1.4  3.6  3.0  4.4  3.6  2.8 -0.6 -4.0 -4.2 -5.8 -3.2
```

Solution

`cumsum(precipitations)` correspond à la somme cumulée des précipitations sur l'année. Par exemple, la 6e valeur du vecteur résultat correspond au total de précipitations de janvier à juin.
`diff(temperature)` correspond à la différence de température d'un mois sur l'autre. Par exemple, la 2e valeur de ce vecteur correspond à l'écart de température entre le mois de février et le mois de janvier.

1.2.7.5. Exercice 5

On a relevé les notes en maths, anglais et sport d'une classe de 6 élèves et on a stocké ces données dans trois vecteurs :

```
maths <- c(12, 16, 8, 18, 6, 10)
anglais <- c(14, 9, 13, 15, 17, 11)
sport <- c(18, 11, 14, 10, 8, 12)
```

Calculer la moyenne des élèves de la classe en anglais.

Calculer la moyenne générale de chaque élève.

Solution

```
mean(anglais)
(maths + anglais + sport) / 3
```

Essayez de comprendre le résultat des deux fonctions suivantes (vous pouvez vous aider de la page d'aide de ces fonctions) :

```
pmin(maths, anglais, sport)
```

```
[1] 12  9  8 10  6 10
```

```
pmax(maths, anglais, sport)
```

```
[1] 18 16 14 18 17 12
```

Solution

`pmin` et `pmax` renvoient les minimum et maximum "parallèles" des trois vecteurs passés en argument. Ainsi, `pmin` renvoie pour chaque élève la note minimale dans les trois matières, et `pmax` la note maximale.

1.3. Premier travail avec des données

1.3.1. Jeu de données d'exemple

Dans cette partie nous allons (enfin) travailler sur des "vraies" données, et utiliser un jeu de données présent dans l'extension `questionr`. Nous devons donc avant toute chose installer cette extension.

Pour installer ce package, deux possibilités :

- Dans l'onglet *Packages* de la zone de l'écran en bas à droite, cliquez sur le bouton *Install*. Dans le dialogue qui s'ouvre, entrez "questionr" dans le champ *Packages* puis cliquez sur *Install*.
- Saisissez directement la commande suivante dans la console : `install.packages("questionr")`

Dans les deux cas, tout un tas de messages devraient s'afficher dans la console. Attendez que l'invite de commandes `>` apparaisse à nouveau.

Pour plus d'informations sur les extensions et leur installation, voir la section @ref(packages).

Le jeu de données que nous allons utiliser est un extrait de l'enquête *Histoire de vie* réalisée par l'INSEE en 2003. Il contient 2000 individus et 20 variables.

Pour pouvoir utiliser ces données, il faut d'abord charger l'extension `questionr` (après l'avoir installée, bien entendu) :

```
library(questionr)
```

L'utilisation de `library` permet de rendre "disponibles", dans notre session R, les fonctions et jeux de données inclus dans l'extension.

Nous devons ensuite indiquer à R que nous souhaitons accéder au jeu de données à l'aide de la commande `data` :

```
data(hdv2003)
```

Cette commande ne renvoie aucun résultat particulier (sauf en cas d'erreur), mais vous devriez voir apparaître dans l'onglet *Environment* de RStudio un nouvel objet nommé `hdv2003` :

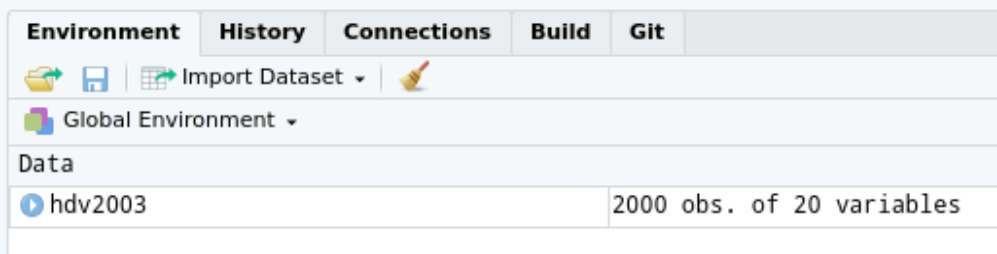


Figure 1.4.: Onglet Environment

Cet objet est d'un type nouveau : il s'agit d'un tableau de données.

1.3.2. Tableau de données (*data frame*)

Un *data frame* (ou tableau de données, ou table) est un type d'objet R qui contient des données au format tabulaire, avec les observations en ligne et les variables en colonnes, comme dans une feuille de tableur de type LibreOffice ou Excel.

Si on se contente d'exécuter le nom de notre tableau de données :

```
hdv2003
```

R va, comme à son habitude, nous l'afficher dans la console, ce qui est tout sauf utile.

Une autre manière d'afficher le contenu du tableau est de cliquer sur l'icône en forme de tableau à droite du nom de l'objet dans l'onglet *Environment* :

Figure 1.5.: Icone view

Ou d'utiliser la fonction `View` :

```
View(hdv2003)
```

Dans les deux cas votre tableau devrait s'afficher dans RStudio avec une interface de type tableur :

Filter

	id	age	sexe	nivetud	poids	occup	qualif	freres.sc
1	1	28	Femme	Enseignement superieur y compris technique sup...	2634.3982	Exerce une profession	Employe	
2	2	23	Femme	NA	9738.3958	Etudiant, eleve	NA	
3	3	59	Homme	Derniere annee d'etudes primaires	3994.1025	Exerce une profession	Technicien	
4	4	34	Homme	Enseignement superieur y compris technique sup...	5731.6615	Exerce une profession	Technicien	
5	5	71	Femme	Derniere annee d'etudes primaires	4329.0940	Retraite	Employe	
6	6	35	Femme	Enseignement technique ou professionnel court	8674.6994	Exerce une profession	Employe	
7	7	60	Femme	Derniere annee d'etudes primaires	6165.8035	Au foyer	Ouvrier qualifie	
8	8	47	Homme	Enseignement technique ou professionnel court	12891.6408	Exerce une profession	Ouvrier qualifie	
9	9	20	Femme	NA	7808.8721	Etudiant, eleve	NA	
10	10	28	Homme	Enseignement technique ou professionnel long	2277.1605	Exerce une profession	Autre	
11	11	65	Femme	Enseignement superieur y compris technique sup...	704.3227	Retraite	Employe	
12	12	47	Homme	2eme cycle	6697.8682	Exerce une profession	Ouvrier qualifie	
13	13	63	Femme	Derniere annee d'etudes primaires	7118.4659	Retraite	Employe	
14	14	67	Femme	Enseignement technique ou professionnel court	586.7714	Exerce une profession	NA	
15	15	76	Femme	A arrete ses etudes, avant la derniere annee d'et...	11042.0774	Retraite	NA	
16	16	49	Femme	Enseignement technique ou professionnel court	9958.2287	Exerce une profession	Employe	
17	17	62	Homme	Enseignement superieur y compris technique sup...	4836.1393	Retraite	Cadre	
18	18	20	Femme	NA	1551.4846	Etudiant, eleve	NA	

Showing 1 to 19 of 2,000 entries

Figure 1.6.: Interface View

Il est important de comprendre que l'objet `hdv2003` contient *l'intégralité* des données du tableau. On voit donc qu'un objet peut contenir des données de types très différents (simple nombre, texte, vecteur, tableau de données entier), et être potentiellement de très grande taille[3].

> **Note**
>
> Sous R, on peut importer ou créer autant de tableaux de données qu'on le souhaite, dans les limites des capacités de sa machine.

Un *data frame* peut être manipulé comme les autres objets vus précédemment. On peut par exemple faire :

[3]La seule limite pour la taille d'un objet étant la mémoire vive (RAM) de la machine sur laquelle tourne la session R.

```
d <- hdv2003
```

ce qui va entraîner la copie de l'ensemble de nos données dans un nouvel objet nommé `d`. Ceci peut paraître parfaitement inutile mais a en fait l'avantage de fournir un objet avec un nom beaucoup plus court, ce qui diminuera la quantité de texte à saisir par la suite.

Pour résumer, comme nous avons désormais décidé de saisir nos commandes dans un script et non plus directement dans la console, les premières lignes de notre fichier de travail sur les données de l'enquête *Histoire de vie* pourraient donc ressembler à ceci :

```
## Chargement des extensions nécessaires
library(questionr)

## Jeu de données hdv2003
data(hdv2003)
d <- hdv2003
```

1.3.2.1. Structure du tableau

Un tableau étant un objet comme un autre, on peut lui appliquer des fonctions. Par exemple, `nrow` et `ncol` retournent le nombre de lignes et de colonnes du tableau :

```
nrow(d)
```

```
[1] 2000
```

```
ncol(d)
```

```
[1] 20
```

La fonction `dim` renvoie ses dimensions, donc les deux nombres précédents :

```
dim(d)
```

```
[1] 2000   20
```

La fonction `names` retourne les noms des colonnes du tableau, c'est-à-dire la liste de nos *variables* :

```
names(d)
```

```
 [1] "id"            "age"           "sexe"          "nivetud"
 [5] "poids"         "occup"         "qualif"        "freres.soeurs"
 [9] "clso"          "relig"         "trav.imp"      "trav.satisf"
[13] "hard.rock"     "lecture.bd"    "peche.chasse"  "cuisine"
[17] "bricol"        "cinema"        "sport"         "heures.tv"
```

Enfin, la fonction `str` renvoie un descriptif plus détaillé de la structure du tableau. Elle liste les différentes variables, indique leur type [4] et affiche les premières valeurs :

```
str(d)
```

```
'data.frame':   2000 obs. of  20 variables:
 $ id           : int  1 2 3 4 5 6 7 8 9 10 ...
 $ age          : int  28 23 59 34 71 35 60 47 20 28 ...
 $ sexe         : Factor w/ 2 levels "Homme","Femme": 2 2 1 1 2 2 2 1 2 1 ...
 $ nivetud      : Factor w/ 8 levels "N'a jamais fait d'etudes",..: 8 NA 3 8 3 6 3 6 NA 7 ...
 $ poids        : num  2634 9738 3994 5732 4329 ...
 $ occup        : Factor w/ 7 levels "Exerce une profession",..: 1 3 1 1 4 1 6 1 3 1 ...
 $ qualif       : Factor w/ 7 levels "Ouvrier specialise",..: 6 NA 3 3 6 6 2 2 NA 7 ...
 $ freres.soeurs: int  8 2 2 1 0 5 1 5 4 2 ...
 $ clso         : Factor w/ 3 levels "Oui","Non","Ne sait pas": 1 1 2 2 1 2 1 2 1 2 ...
 $ relig        : Factor w/ 6 levels "Pratiquant regulier",..: 4 4 4 3 1 4 3 4 3 2 ...
 $ trav.imp     : Factor w/ 4 levels "Le plus important",..: 4 NA 2 3 NA 1 NA 4 NA 3 ...
 $ trav.satisf  : Factor w/ 3 levels "Satisfaction",..: 2 NA 3 1 NA 3 NA 2 NA 1 ...
 $ hard.rock    : Factor w/ 2 levels "Non","Oui": 1 1 1 1 1 1 1 1 1 1 ...
 $ lecture.bd   : Factor w/ 2 levels "Non","Oui": 1 1 1 1 1 1 1 1 1 1 ...
 $ peche.chasse : Factor w/ 2 levels "Non","Oui": 1 1 1 1 1 1 2 2 1 1 ...
 $ cuisine      : Factor w/ 2 levels "Non","Oui": 2 1 1 2 1 1 2 2 1 1 ...
 $ bricol       : Factor w/ 2 levels "Non","Oui": 1 1 1 2 1 1 1 2 1 1 ...
 $ cinema       : Factor w/ 2 levels "Non","Oui": 1 2 1 2 1 2 1 1 2 2 ...
 $ sport        : Factor w/ 2 levels "Non","Oui": 1 2 2 2 1 2 1 1 1 2 ...
 $ heures.tv    : num  0 1 0 2 3 2 2.9 1 2 2 ...
```

Sous RStudio, on peut afficher à tout moment la structure d'un objet en cliquant sur l'icône de triangle sur fond bleu à gauche du nom de l'objet dans l'onglet *Environment* :

[4] Les différents types de variables seront décrits plus en détail dans le chapitre @ref(vectorfactor) sur les recodages.

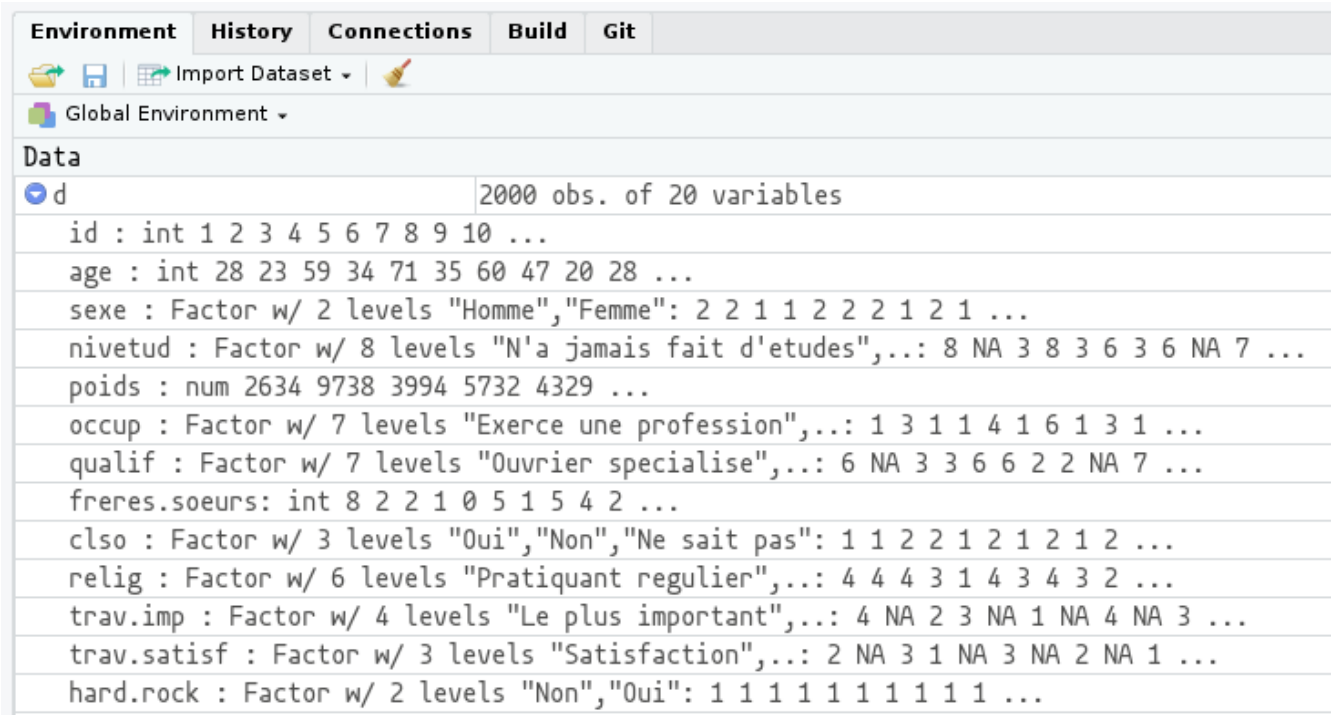


Figure 1.7.: Structure d'un objet

1.3.2.2. Accéder aux variables d'un tableau

Une opération très importante est l'accès aux variables du tableau (à ses colonnes) pour pouvoir les manipuler, effectuer des calculs, etc. On utilise pour cela l'opérateur `$`, qui permet d'accéder aux colonnes du tableau. Ainsi, si l'on tape :

```
d$sexe
```

```
  [1] Femme Femme Homme Homme Femme Femme Femme Homme Femme Homme Femme Homme
 [13] Femme Femme Femme Femme Homme Femme Homme Femme Femme Homme Femme Femme
 [25] Femme Homme Femme Homme Homme Homme Homme Homme Homme Homme Femme Femme
 [37] Homme Femme Femme Homme Femme Homme Homme Femme Femme Homme Femme Femme
 [49] Femme Femme Homme Femme Homme Femme Homme Femme Femme Femme Homme Femme
 [61] Femme Homme Homme Homme Homme Femme Homme Homme Femme Femme Homme Homme
 [73] Femme Femme Femme Femme Homme Femme Femme Femme Femme Femme Femme Homme
 [85] Homme Femme Homme Homme Homme Homme Homme Femme Homme Femme Femme Femme
 [97] Homme Homme Femme Femme Femme Homme Femme Homme Homme Femme Femme Femme
[109] Femme Homme Homme Homme Homme Homme Femme Homme Homme Femme Homme Homme
[121] Femme Femme Femme Homme Femme Femme Homme Femme Femme Homme Femme Homme
[133] Femme Femme Femme Homme Homme Homme Homme Homme Homme Homme Homme Femme
[145] Homme Homme Homme Femme Femme Femme Homme Femme Femme Femme Femme Homme
[157] Femme Homme Homme Homme Femme Homme Femme Homme Femme Homme Homme Femme
[169] Femme Femme Homme Femme Homme Femme Femme Femme Homme Homme Homme Femme
[181] Homme Femme Femme Homme Homme Femme Femme Femme Femme Femme Homme Homme
[193] Femme Homme Homme Femme Homme Femme Homme Femme
```

```
 [ reached getOption("max.print") -- omitted 1800 entries ]
Levels: Homme Femme
```

R va nous afficher l'ensemble des valeurs de notre variable `sexe` dans la console, ce qui est à nouveau fort peu utile. Mais cela nous permet de constater que `d$sexe` est un vecteur de chaînes de caractères tels qu'on en a déjà rencontré précédemment.

La fonction `table$colonne` renvoie donc la colonne nommée `colonne` du tableau `table`, c'est-à-dire un vecteur, en général de nombres ou de chaînes de caractères.

Si on souhaite afficher seulement les premières ou dernières valeurs d'une variable, on peut utiliser les fonctions `head` et `tail` :

```
head(d$age)
```

```
[1] 28 23 59 34 71 35
```

```
tail(d$age, 10)
```

```
 [1] 52 42 50 41 46 45 46 24 24 66
```

Le deuxième argument numérique permet d'indiquer le nombre de valeurs à afficher.

1.3.2.3. Créer une nouvelle variable

On peut aussi utiliser l'opérateur `$` pour créer une nouvelle variable dans notre tableau : pour cela, il suffit de lui assigner une valeur.

Par exemple, la variable `heures.tv` contient le nombre d'heures passées quotidiennement devant la télé :

```
head(d$heures.tv, 10)
```

```
 [1] 0.0 1.0 0.0 2.0 3.0 2.0 2.9 1.0 2.0 2.0
```

On peut vouloir créer une nouvelle variable dans notre tableau qui contienne la même durée mais en minutes. On va donc créer une nouvelle variables `minutes.tv` de la manière suivante :

```
d$minutes.tv <- d$heures.tv * 60
```

On peut alors constater, soit visuellement soit dans la console, qu'une nouvelle variable (une nouvelle colonne) a bien été ajoutée au tableau :

```
head(d$minutes.tv)
```

```
[1]   0  60   0 120 180 120
```

1.3.3. Analyse univariée

On a donc désormais accès à un tableau de données `d`, dont les lignes sont des observations (des individus enquêtés), et les colonnes des variables (des caractéristiques de chacun de ces individus), et on sait accéder à ces variables grâce à l'opérateur `$`.

Si on souhaite analyser ces variables, les méthodes et fonctions utilisées seront différentes selon qu'il s'agit d'une variable *quantitative* (variable numérique pouvant prendre un grand nombre de valeurs : l'âge, le revenu, un pourcentage…) ou d'une variable *qualitative* (variable pouvant prendre un nombre limité de valeurs appelées modalités : le sexe, la profession, le dernier diplôme obtenu, etc.).

1.3.3.1. Analyser une variable quantitative

Une variable quantitative est une variable de type numérique (un nombre) qui peut prendre un grand nombre de valeurs. On en a plusieurs dans notre jeu de données, notamment l'âge (variable `age`) ou le nombre d'heures passées devant la télé (`heures.tv`).

1.3.3.1.1. Indicateurs de centralité

Caractériser une variable quantitative, c'est essayer de décrire la manière dont ses valeurs se répartissent, ou se distribuent.

Pour cela on peut commencer par regarder les valeurs extrêmes, avec les fonctions `min`, `max` ou `range` :

```
min(d$age)
```

```
[1] 18
```

```
max(d$age)
```

```
[1] 97
```

```
range(d$age)
```

```
[1] 18 97
```

On peut aussi calculer des indicateurs de *centralité* : ceux-ci indiquent autour de quel nombre se répartissent les valeurs de la variable. Il y en a plusieurs, le plus connu étant la moyenne, qu'on peut calculer avec la fonction `mean` :

```
mean(d$age)
```

```
[1] 48.157
```

Il existe aussi la médiane, qui est la valeur qui sépare notre population en deux : on a la moitié de nos observations en-dessous, et la moitié au-dessus. Elle se calcule avec la fonction `median` :

```
median(d$age)
```

```
[1] 48
```

Une différence entre les deux indicateurs est que la médiane est beaucoup moins sensible aux valeurs "extrêmes" : on dit qu'elle est plus *robuste*. Ainsi, en 2013, le salaire net *moyen* des salariés à temps plein en France était de 2202 euros, tandis que le salaire net *médian* n'était que de 1772 euros. La différence étant due à des très hauts salaires qui "tirent" la moyenne vers le haut.

1.3.3.1.2. Indicateurs de dispersion

Les indicateurs de dispersion permettent de mesurer si les valeurs sont plutôt regroupées ou au contraire plutôt dispersées.

L'indicateur le plus simple est l'étendue de la distribution, qui décrit l'écart maximal observé entre les observations :

```
max(d$age) - min(d$age)
```

```
[1] 79
```

Les indicateurs de dispersion les plus utilisés sont la variance ou, de manière équivalente, l'écart-type (qui est égal à la racine carrée de la variance). On obtient la première avec la fonction `var`, et le second avec `sd` (abbréviation de *standard deviation*) :

```
var(d$age)
```

```
[1] 287.0249
```

```r
sd(d$age)
```

```
[1] 16.94181
```

Plus la variance ou l'écart-type sont élevés, plus les valeurs sont dispersées autour de la moyenne. À l'inverse, plus ils sont faibles et plus les valeurs sont regroupées.

Une autre manière de mesurer la dispersion est de calculer les quartiles :

- le premier quartile est la valeur pour laquelle on a 25% des observations en dessous et 75% au dessus
- le deuxième quartile est la valeur pour laquelle on a 50% des observations en dessous et 50% au dessus (c'est donc la médiane)
- le troisième quartile est la valeur pour laquelle on a 75% des observations en dessous et 25% au dessus

On peut les calculer avec la fonction `quantile` :

```r
### Premier quartile
quantile(d$age, prob = 0.25)
```

```
25%
 35
```

```r
## Troisième quartile
quantile(d$age, prob = 0.75)
```

```
75%
 60
```

`quantile` prend deux arguments principaux : le vecteur dont on veut calculer le quantile, et un argument `prob` qui indique quel quantile on souhaite obtenir. `prob` prend une valeur entre 0 et 1 : 0.5 est la médiane, 0.25 le premier quartile, 0.1 le premier décile, etc.

Notons enfin que la fonction `summary` permet d'obtenir d'un coup plusieurs indicateurs classiques :

```r
summary(d$age)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  18.00   35.00   48.00   48.16   60.00   97.00
```

1.3.3.1.3. Représentation graphique

L'outil le plus utile pour étudier la distribution des valeurs d'une variable quantitative reste la représentation graphique.

La représentation la plus courante est sans doute l'histogramme. On peut l'obtenir avec la fonction `hist` :

```
hist(d$age)
```

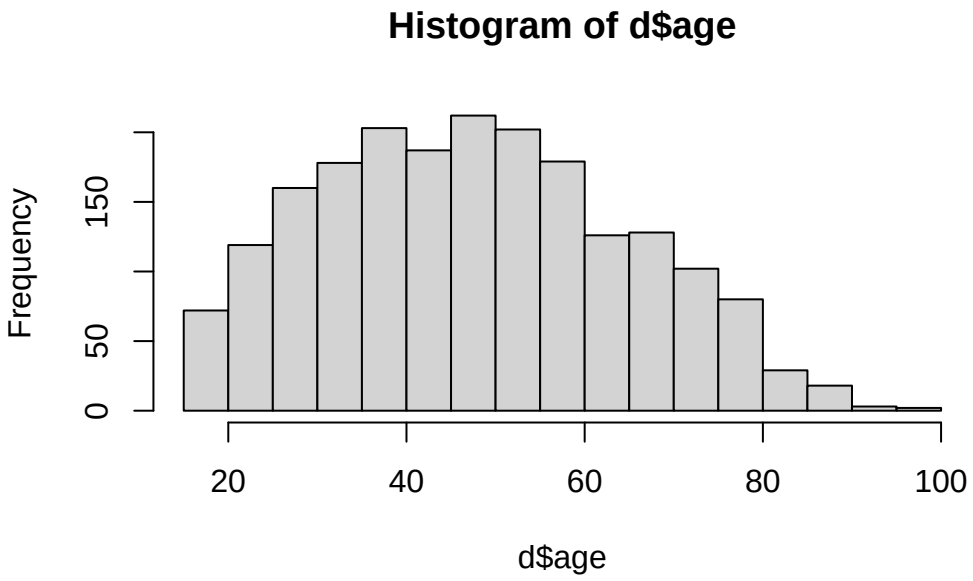


Figure 1.8.: Histogramme de l'age par défaut

Cette fonction n'a pas pour effet direct d'effectuer un calcul ou de nous renvoyer un résultat : elle génère un graphique qui va s'afficher dans l'onglet *Plots* de RStudio.

On peut personnaliser l'apparence de l'histogramme en ajoutant des arguments supplémentaires à la fonction `hist`. L'argument le plus important est `breaks`, qui permet d'indiquer le nombre de classes que l'on souhaite.

```
hist(d$age, breaks = 10, main = "")
```

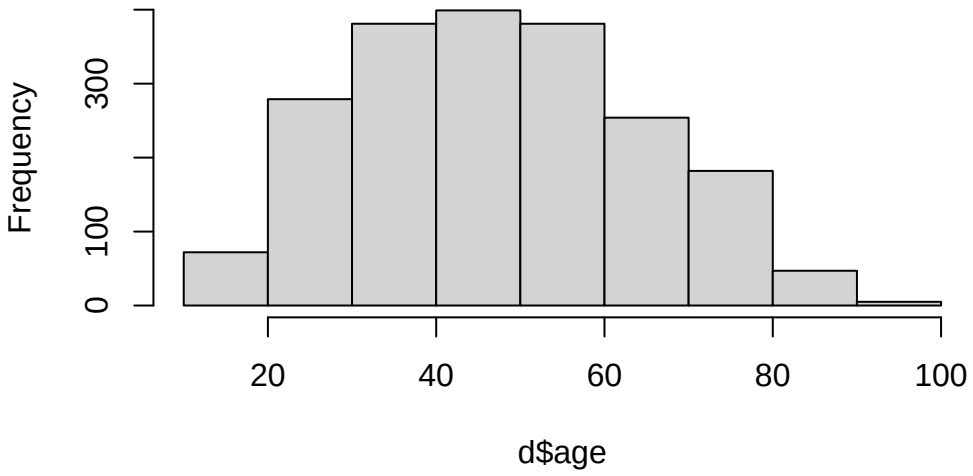


Figure 1.9.: Histogramme de l'age avec 10 classes

```
hist(d$age, breaks = 70, main = "")
```

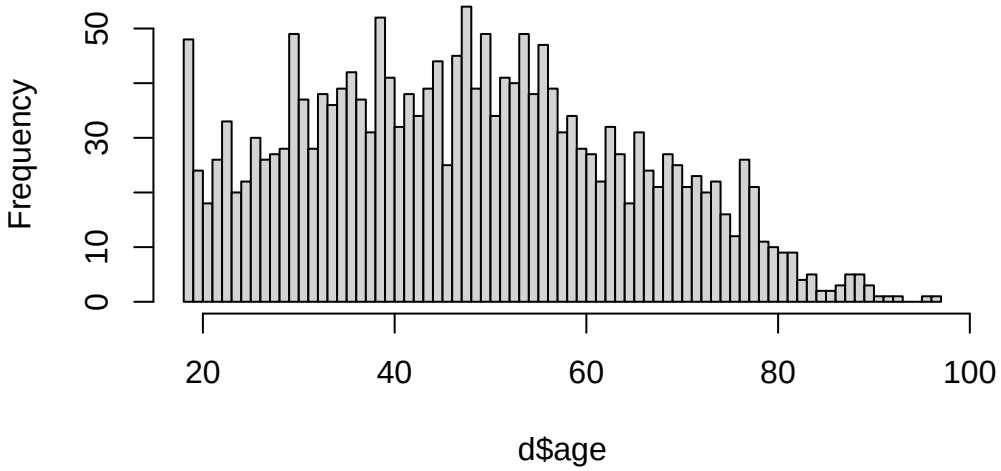


Figure 1.10.: Histogramme de l'age avec 70 classes

Le choix d'un "bon" nombre de classes pour un histogramme n'est pas un problème simple : si on a trop peu de classes, on risque d'effacer quasiment toutes les variations, et si on en a trop on risque d'avoir trop de détails et de masquer les grandes tendances.

Les arguments de `hist` permettent également de modifier la présentation du graphique. On peut ainsi changer la couleur des barres avec `col`[5], le titre avec `main`, les étiquettes des axes avec `xlab` et `ylab`, etc. :

```
hist(d$age,
  col = "skyblue",
  main = "Répartition des âges des enquêtés",
  xlab = "Âge",
  ylab = "Effectif"
)
```

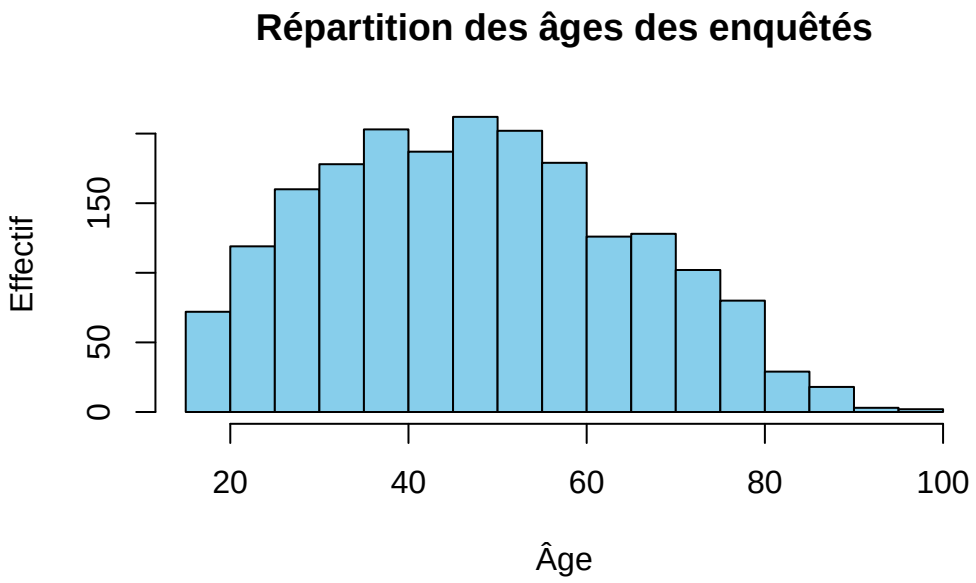


Figure 1.11.: Histogramme modifié

La fonction `hist` fait partie des fonctions graphique de base de R. On verra plus en détail d'autres fonctions graphiques avec l'extension `ggplot2` qui permet la production et la personnalisation de graphiques complexes.

1.3.3.2. Analyser une variable qualitative

Une variable qualitative est une variable qui ne peut prendre qu'un nombre limité de valeurs, appelées modalités. Dans notre jeu de données on trouvera par exemple le sexe (`sexe`), le niveau d'études (`nivetud`), la catégorie socio-professionnelle (`qualif`)…

À noter qu'une variable qualitative peut tout-à-fait être numérique, et que certaines variables peuvent être traitées soit comme quantitatives, soit comme qualitatives : c'est le cas par exemple du nombre d'enfants ou du nombre de frères et soeurs.

[5]Les différentes manières de spécifier des couleurs sont indiquées dans l'encadré de la section @ref(scalecolor).

1.3.3.2.1. Tri à plat

L'outil le plus utilisé pour représenter la répartition des valeurs d'une variable qualitative est le *tri à plat* : il s'agit simplement de compter, pour chacune des valeurs possibles de la variable (pour chacune des modalités), le nombre d'observations ayant cette valeur. Un tri à plat s'obtient sous R à l'aide de la fonction `table` :

```
table(d$sexe)
```

```
Homme Femme
  899  1101
```

Ce tableau nous indique donc que parmi nos enquêtés on trouve 899 hommes et 1101 femmes.

```
table(d$qualif)
```

```
      Ouvrier specialise         Ouvrier qualifie                Technicien
                     203                      292                        86
Profession intermediaire                   Cadre                   Employe
                     160                      260                       594
                  Autre
                     58
```

Un tableau de ce type peut être affiché ou stocké dans un objet, et on peut à son tour lui appliquer des fonctions. Par exemple, la fonction `sort` permet de trier le tableau selon la valeur de l'effectif. On peut donc faire :

```
tab <- table(d$qualif)
sort(tab)
```

```
                  Autre                Technicien Profession intermediaire
                     58                        86                       160
      Ouvrier specialise                     Cadre         Ouvrier qualifie
                     203                       260                       292
                Employe
                    594
```

> **!** Important
>
> Attention, par défaut la fonction `table` n'affiche pas les valeurs manquantes (`NA`). Si on souhaite les inclure il faut utiliser l'argument `useNA = "always"`, soit : `table(d$qualif, useNA = "always")`.

À noter qu'on peut aussi appliquer `summary` à une variable qualitative. Le résultat est également le tri à plat de la variable, avec en plus le nombre de valeurs manquantes éventuelles :

```
summary(d$qualif)
```

```
      Ouvrier specialise         Ouvrier qualifie                Technicien
                     203                      292                        86
Profession intermediaire                   Cadre                   Employe
                     160                      260                       594
                   Autre                   NA's
                      58                      347
```

Par défaut ces tris à plat sont en effectifs et ne sont donc pas toujours très lisibles, notamment quand on a des effectifs importants. On leur rajoute donc en général la répartition en pourcentages. Pour cela, nous allons utiliser la fonction `freq` de l'extension `questionr`, qui devra donc avoir précédemment été chargée avec `library(questionr)` :

```
## À rajouter en haut de script et à exécuter
library(questionr)
```

On peut alors utiliser la fonction :

```
freq(d$qualif)
```

```
                           n    % val%
Ouvrier specialise       203 10.2 12.3
Ouvrier qualifie         292 14.6 17.7
Technicien                86  4.3  5.2
Profession intermediaire 160  8.0  9.7
Cadre                    260 13.0 15.7
Employe                  594 29.7 35.9
Autre                     58  2.9  3.5
NA                       347 17.3   NA
```

La colonne `n` représente les effectifs de chaque catégorie, la colonne `%` le pourcentage, et la colonne `val%` le pourcentage calculé sur les valeurs valides, donc en excluant les `NA`. Une ligne a également été rajoutée pour indiquer le nombre et la proportion de `NA`.

`freq` accepte un certain nombre d'arguments pour personnaliser son affichage. Par exemple :

- `valid` indique si on souhaite ou non afficher les pourcentages sur les valeurs valides
- `cum` indique si on souhaite ou non afficher les pourcentages cumulés
- `total` permet d'ajouter une ligne avec les effectifs totaux
- `sort` permet de trier le tableau par fréquence croissante (`sort="inc"`) ou décroissante (`sort="dec"`).

```
freq(d$qualif, valid = FALSE, total = TRUE, sort = "dec")
```

```
                           n     %
Employe                  594  29.7
Ouvrier qualifie         292  14.6
Cadre                    260  13.0
Ouvrier specialise       203  10.2
Profession intermediaire 160   8.0
Technicien                86   4.3
Autre                     58   2.9
NA                       347  17.3
Total                   2000 100.0
```

1.3.3.2.2. Représentations graphiques

On peut représenter graphiquement le tri à plat d'une variable qualitative avec un diagramme en barres, obtenu avec la fonction `barplot`. Attention, contrairement à `hist` cette fonction ne s'applique pas directement à la variable mais au résultat du tri à plat de cette variable, calculé avec `table`. Il faut donc procéder en deux étapes :

```
tab <- table(d$clso)
barplot(tab)
```

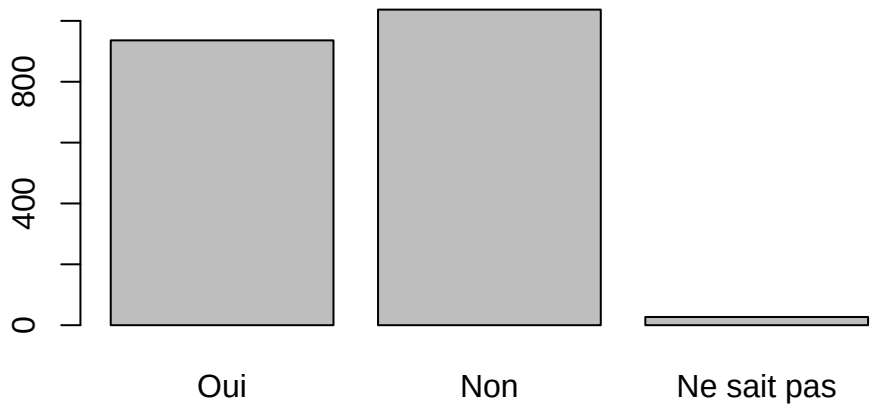


Figure 1.12.: Graphique en barre

On peut aussi trier le tri à plat avec la fonction `sort` avant de le représenter graphiquement, ce qui peut faciliter la lecture du graphique :

```
barplot(sort(tab))
```

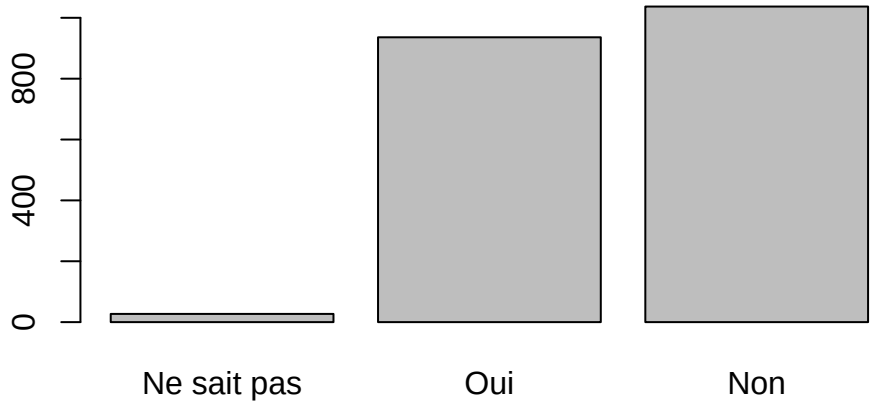


Figure 1.13.: Graphique en barre trié

Une alternative au graphique en barres est le *diagramme de Cleveland*, qu'on peut obtenir avec la fonction `dotchart`. Celle-ci s'applique elle aussi au tri à plat de la variable calculé avec `table`.

```
dotchart(table(d$qualif))
```

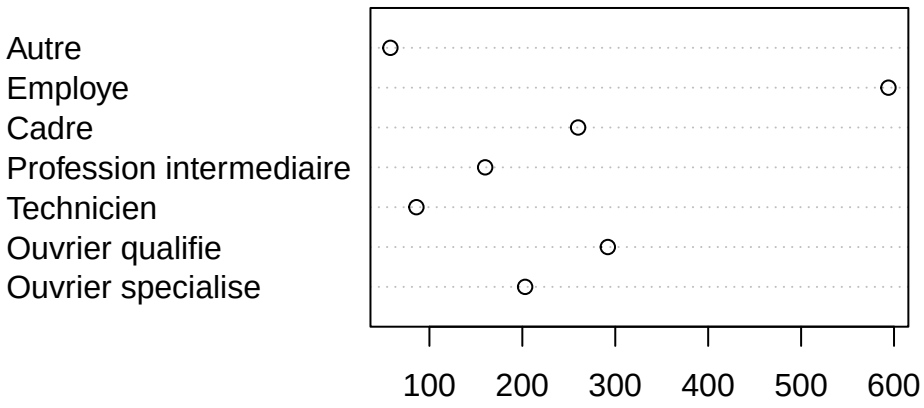


Figure 1.14.: Graphique de Cleveland

Là aussi, pour améliorer la lisibilité du graphique il est préférable de trier le tri à plat de la variable avant de le représenter :

```
dotchart(sort(table(d$qualif)))
```

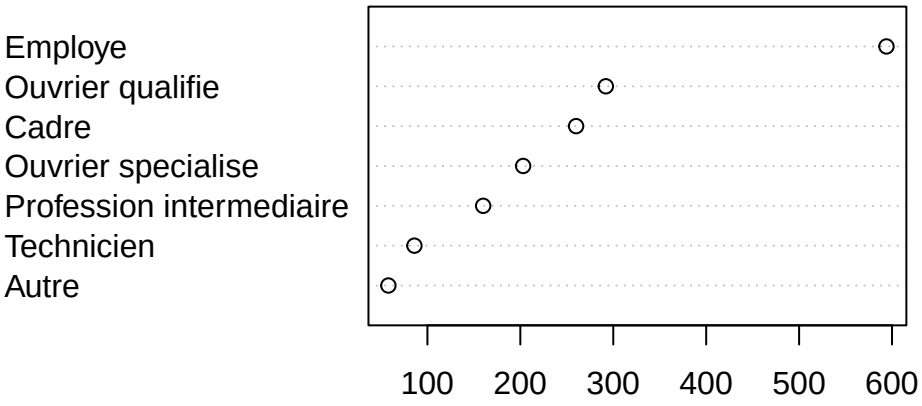


Figure 1.15.: Graphique de Cleveland trié

1.3.4. Exercices

Exercice 1

Créer un nouveau script qui effectue les actions suivantes :

- charger l'extension `questionr`
- charger le jeu de données nommé `hdv2003`
- copier le jeu de données dans un nouvel objet nommé `df`
- afficher les dimensions et la liste des variables de `df`

Solution

```
library(questionr)

data(hdv2003)
df <- hdv2003

dim(df)
names(df)
```

Exercice 2

On souhaite étudier la répartition du temps passé devant la télévision par les enquêtés (variable `heures.tv`). Pour cela, affichez les principaux indicateurs de cette variable : valeur minimale, maximale, moyenne, médiane et écart-type. Représentez ensuite sa distribution par un histogramme en 10 classes.

Solution

```
summary(df$heures.tv)
sd(df$heures.tv)

hist(df$heures.tv, breaks = 10)
```

Exercice 3

On s'intéresse maintenant à l'importance accordée par les enquêtés à leur travail (variable `trav.imp`). Faites un tri à plat des effectifs des modalités de cette variable avec la commande `table`.

Solution

```
table(df$trav.imp)
```

Faites un tri à plat affichant à la fois les effectifs et les pourcentages de chaque modalité. Y'a-t-il des valeurs manquantes ?

Solution

```
freq(df$trav.imp)
```

Représentez graphiquement les effectifs des modalités à l'aide d'un graphique en barres.

Solution

```
tab <- sort(table(df$trav.imp))
barplot(tab)
```

Utilisez l'argument `col` de la fonction `barplot` pour modifier la couleur du graphique en `tomato`.

Solution

```
barplot(tab, col = "tomato")
```

Tapez `colors()` dans la console pour afficher l'ensemble des noms de couleurs disponibles dans R. Testez chaque couleur une à une pour trouver votre couleur préférée.

Solution

C'est une blague, hein ! Cela dit `moccasin` ou `palevioletred` sont pas mal, si vous voulez essayer :-)

1.4. Analyse de 2 variables

Faire une analyse bivariée, c'est étudier la relation entre deux variables : sont-elles liées ? les valeurs de l'une influencent-elles les valeurs de l'autre ? ou sont-elles au contraire indépendantes ?

À noter qu'on va parler ici d'influence ou de lien, mais pas de relation de cause à effet : les outils présentés permettent de visualiser ou de déterminer une relation, mais des liens de causalité proprement dit sont plus difficiles à mettre en évidence. Il faut en effet vérifier que c'est bien telle variable qui influence telle autre et pas l'inverse, qu'il n'y a pas de "variable cachée", etc.

Là encore, le type d'analyse ou de visualisation est déterminé par la nature qualitative ou quantitative des deux variables.

1.4.1. Croisement de deux variables qualitatives

1.4.1.1. Tableaux croisés

On va continuer à travailler avec le jeu de données tiré de l'enquête *Histoire de vie* inclus dans l'extension `questionr`. On commence donc par charger l'extension, le jeu de données, et à le renommer en un nom plus court pour gagner un peu de temps de saisie au clavier :

```
library(questionr)
data(hdv2003)
d <- hdv2003
```

Quand on veut croiser deux variables qualitatives, on fait un *tableau croisé*. Comme pour un tri à plat ceci s'obtient avec la fonction `table` de R, mais à laquelle on passe cette fois deux variables en argument. Par exemple, si on veut croiser la catégorie socio-professionnelle et le sexe des enquêtés :

```
table(d$qualif, d$sexe)
```

```
                          Homme Femme
  Ouvrier specialise         96   107
  Ouvrier qualifie          229    63
  Technicien                 66    20
  Profession intermediaire   88    72
  Cadre                     145   115
  Employe                    96   498
  Autre                      21    37
```

Pour pouvoir interpréter ce tableau on doit passer du tableau en effectifs au tableau en pourcentages ligne ou colonne. Pour cela, on peut utiliser les fonctions `lprop` et `cprop` de l'extension `questionr`, qu'on applique au tableau croisé précédent.

Pour calculer les pourcentages ligne :

```
tab <- table(d$qualif, d$sexe)
lprop(tab)
```

```
                         Homme Femme Total
  Ouvrier specialise      47.3  52.7 100.0
  Ouvrier qualifie        78.4  21.6 100.0
  Technicien              76.7  23.3 100.0
  Profession intermediaire 55.0  45.0 100.0
  Cadre                   55.8  44.2 100.0
  Employe                 16.2  83.8 100.0
  Autre                   36.2  63.8 100.0
  All                     44.8  55.2 100.0
```

Et pour les pourcentages colonne :

```
cprop(tab)
```

```
                         Homme Femme All
  Ouvrier specialise      13.0  11.7  12.3
  Ouvrier qualifie        30.9   6.9  17.7
  Technicien               8.9   2.2   5.2
  Profession intermediaire 11.9   7.9   9.7
  Cadre                   19.6  12.6  15.7
  Employe                 13.0  54.6  35.9
  Autre                    2.8   4.1   3.5
  Total                  100.0 100.0 100.0
```

> **Note**
>
> Pour savoir si on doit faire des pourcentages ligne ou colonne, on pourra se référer à l'article suivant : http://alain-leger.lescigales.org/textes/lignecolonne.pdf
> En résumé, quand on fait un tableau croisé, celui-ci est parfaitement symétrique : on peut inverser les lignes et les colonnes, ça ne change pas son interprétation. Par contre, on a toujours en tête un "sens" de lecture dans le sens où on considère que l'une des variables *dépend* de l'autre. Par exemple, si on croise sexe et type de profession, on dira que le type de profession dépend du sexe, et non l'inverse : le type de profession est alors la variable *dépendante* (à expliquer), et le sexe la variable *indépendante* (explicative).
> Pour faciliter la lecture d'un tableau croisé, il est recommandé de **faire les pourcentages sur**

la variable indépendante. Dans notre exemple, la variable indépendante est le sexe, elle est en colonne, on calcule donc les pourcentages colonnes qui permettent de comparer directement, pour chaque sexe, la répartition des catégories socio-professionnelles.

1.4.1.2. Représentation graphique

Il est possible de faire une représentation graphique d'un tableau croisé, par exemple avec la fonction `mosaicplot` :

```
mosaicplot(tab)
```

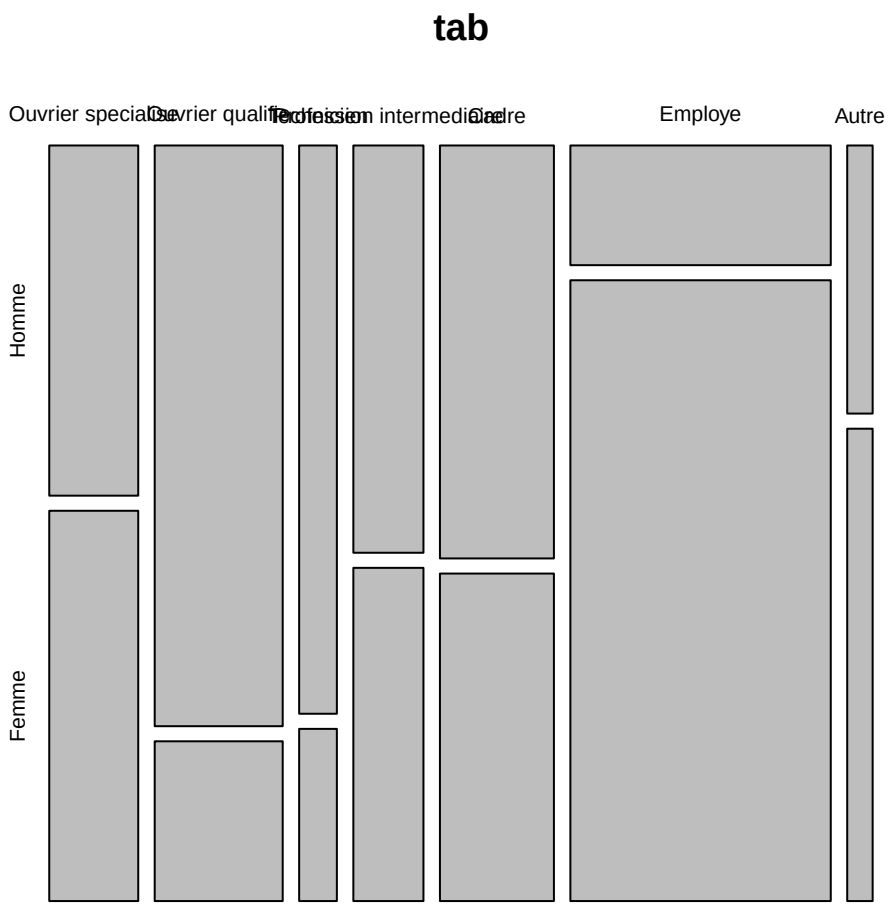


Figure 1.16.: Graphique mosaique

On peut améliorer ce graphique en colorant les cases selon les résidus du test du ^{2} (argument `shade = TRUE`) et en orientant verticalement les labels de colonnes (argument `las = 3`) :

```
mosaicplot(tab, las = 3, shade = TRUE)
```

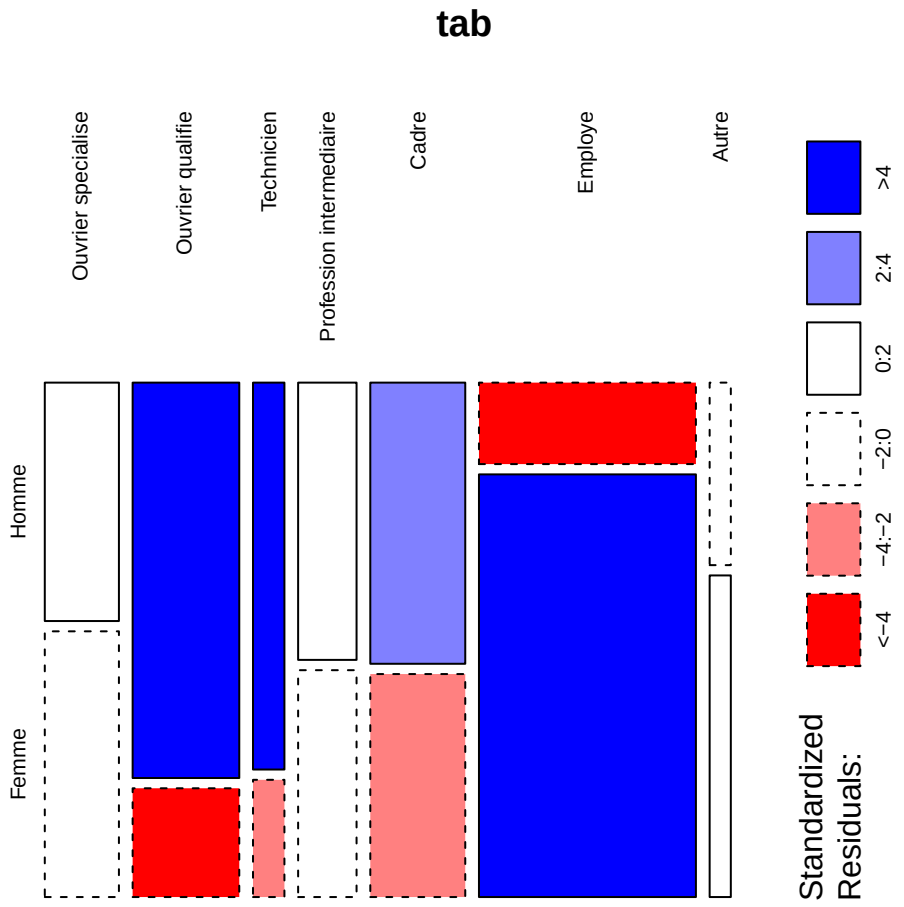


Figure 1.17.: Graphique mosaique modifié

Chaque rectangle de ce graphique représente une case de tableau. Sa largeur correspond au pourcentage des modalités en colonnes (il y'a beaucoup d'employés et d'ouvriers et très peu d'"autres"). Sa hauteur correspond aux pourcentages colonnes : la proportion d'hommes chez les cadres est plus élevée que chez les employés. Enfin, la couleur de la case correspond au résidu du test du ^{2} correspondant : les cases en rouge sont sous-représentées, les cases en bleu sur-représentées, et les cases blanches sont proches des effectifs attendus sous l'hypothèse d'indépendance.

1.4.2. Croisement d'une variable quantitative et d'une variable qualitative

1.4.2.1. Représentation graphique

Croiser une variable quantitative et une variable qualitative, c'est essayer de voir si les valeurs de la variable quantitative se répartissent différemment selon la catégorie d'appartenance de la variable qualitative.

Pour cela, l'idéal est de commencer par une représentation graphique de type "boîte à moustache" à l'aide de la fonction `boxplot`. Par exemple, si on veut visualiser la répartition des âges selon la pratique ou non d'un sport, on va utiliser la syntaxe suivante :

```
boxplot(age ~ sport, data = d)
```

> **Note**
>
> Cette syntaxe de `boxplot` utilise une nouvelle notation de type "formule". Celle-ci est utilisée notamment pour la spécification des modèles de régression. Ici le `~` peut se lire comme "en fonction de" : on veut représenter le boxplot de l'âge en fonction du sport.

Ce qui va nous donner le résultat suivant :

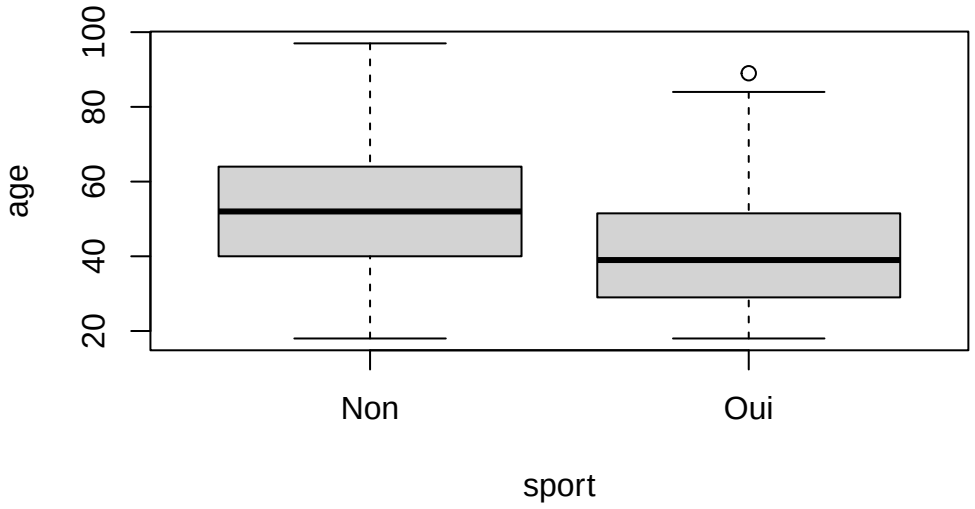


Figure 1.18.: Graphique en boites à moustaches

> **Note**
>
> L'interprétation d'un boxplot est la suivante : Les bords inférieurs et supérieurs du carré central représentent le premier et le troisième quartile de la variable représentée sur l'axe vertical. On a donc 50% de nos observations dans cet intervalle. Le trait horizontal dans le carré représente la médiane. Enfin, des "moustaches" s'étendent de chaque côté du carré, jusqu'aux valeurs minimales

> et maximales, avec une exception : si des valeurs sont éloignées du carré de plus de 1,5 fois l'écart interquartile (la hauteur du carré), alors on les représente sous forme de points (symbolisant des valeurs considérées comme "extrêmes").

Dans le graphique ci-dessus, on voit que ceux qui ont pratiqué un sport au cours des douze derniers mois ont l'air d'être sensiblement plus jeunes que les autres.

1.4.2.2. Calculs d'indicateurs

On peut aussi vouloir comparer certains indicateurs (moyenne, médiane) d'une variable quantitative selon les modalités d'une variable qualitative. Si on reprend l'exemple précédent, on peut calculer la moyenne d'âge pour ceux qui pratiquent un sport et pour ceux qui n'en pratiquent pas.

Une première méthode pour cela est d'extraire de notre population autant de sous-populations qu'il y a de modalités dans la variable qualitative. On peut le faire notamment avec la fonction `subset`.

On applique `subset` pour créer deux sous-populations, stockées dans deux nouveaux tableaux de données :

```
d_sport <- subset(d, sport == "Oui")
d_nonsport <- subset(d, sport == "Non")
```

On peut ensuite utiliser ces deux nouveaux tableaux de données comme on en a l'habitude, et calculer les deux moyennes d'âge :

```
mean(d_sport$age)
```

```
[1] 40.92531
```

```
mean(d_nonsport$age)
```

```
[1] 52.25137
```

Une autre possibilité est d'utiliser la fonction `tapply`, qui prend en paramètre une variable quantitative, une variable qualitative et une fonction, puis applique automatiquement la fonction aux valeurs de la variables quantitative pour chaque niveau de la variable qualitative :

```
tapply(d$age, d$sport, mean)
```

```
     Non      Oui
52.25137 40.92531
```

1.4.3. Croisement de deux variables quantitatives

Le jeu de données `hdv2003` comportant assez peu de variables quantitatives, on va s'intéresser maintenant à un autre jeu de données comportant des informations du recensement de la population de 2012. On le charge avec :

```
data(rp2012)
```

Un nouveau tableau de données `rp2012` devrait apparaître dans votre environnement. Celui-ci comprend les 5170 communes de France métropolitaine de plus de 2000 habitants, et une soixantaine de variables telles que le département, la population, le taux de chômage, etc. Pour une description plus complète et une liste des variables, voir section @ref(rp2012).

1.4.3.1. Représentation graphique

Quand on croise deux variables quantitatives, l'idéal est de faire une représentation graphique sous forme de nuage de points à l'aide de la fonction `plot`. On va représenter le croisement entre le pourcentage de cadres et le pourcentage de propriétaires dans la commune :

```
plot(rp2012$cadres, rp2012$proprio)
```

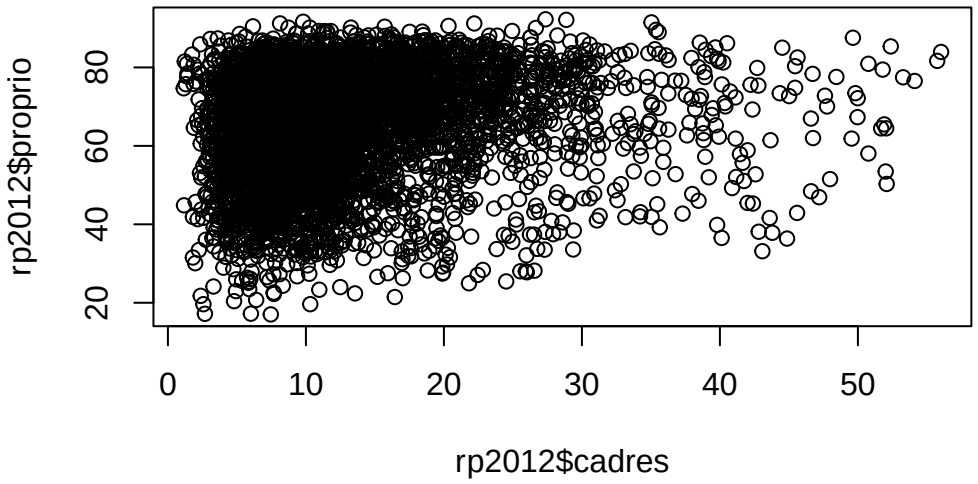


Figure 1.19.: Graphique du pourcentage de propriétaire en fonction du pourcentage de cadre

Une représentation graphique est l'idéal pour visualiser l'existence d'un lien entre les deux variables. Voici quelques exemples d'interprétation :

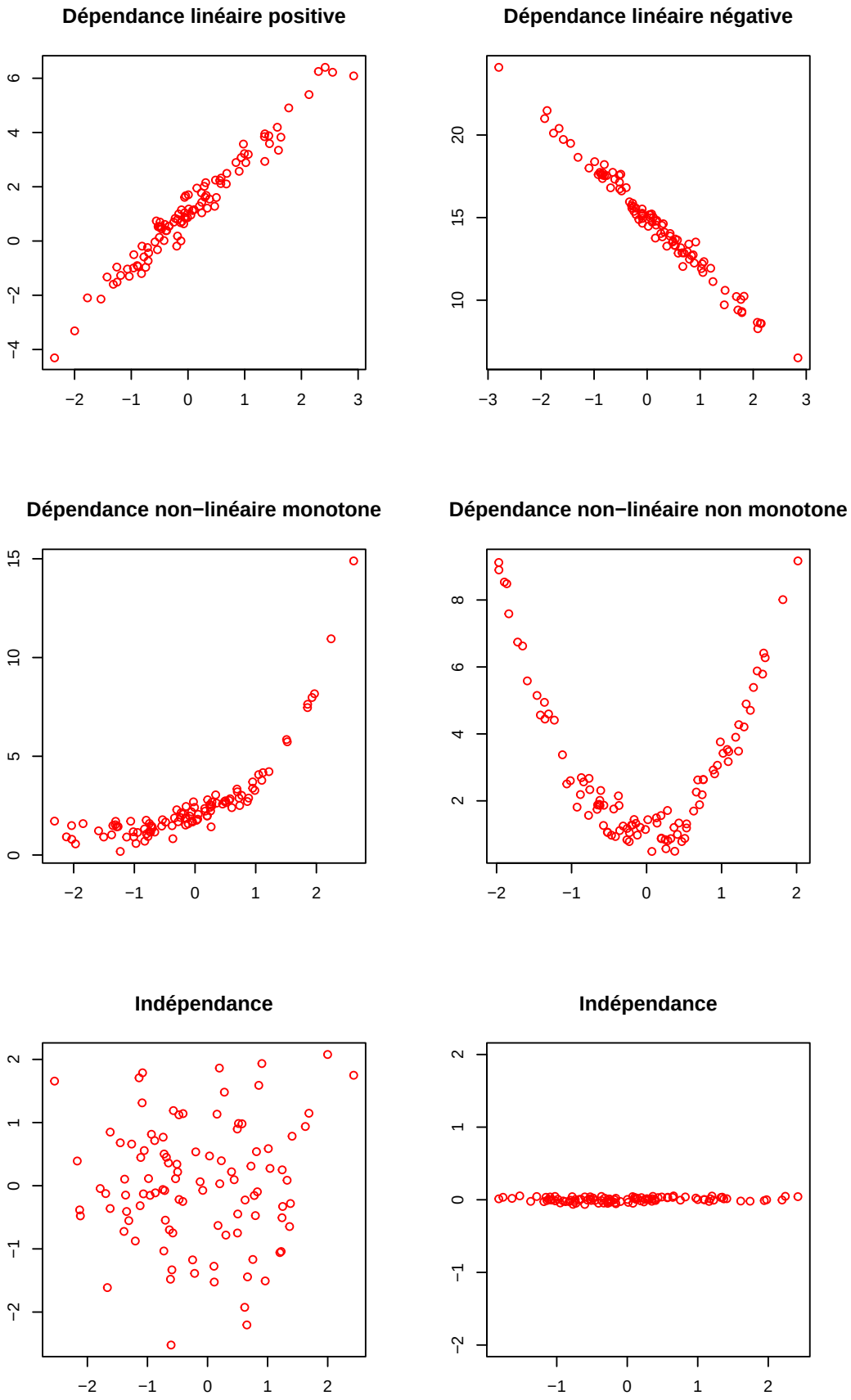


Figure 1.20.: Illustration des relations bivariées

Dans ce premier graphique généré sur nos données, il semble difficile de mettre en évidence une relation de dépendance. Si par contre on croise le pourcentage de cadres et celui de diplômés du supérieur, on obtient une belle relation de dépendance linéaire.

```
plot(dipl_sup ~ cadres, data = rp2012)
```

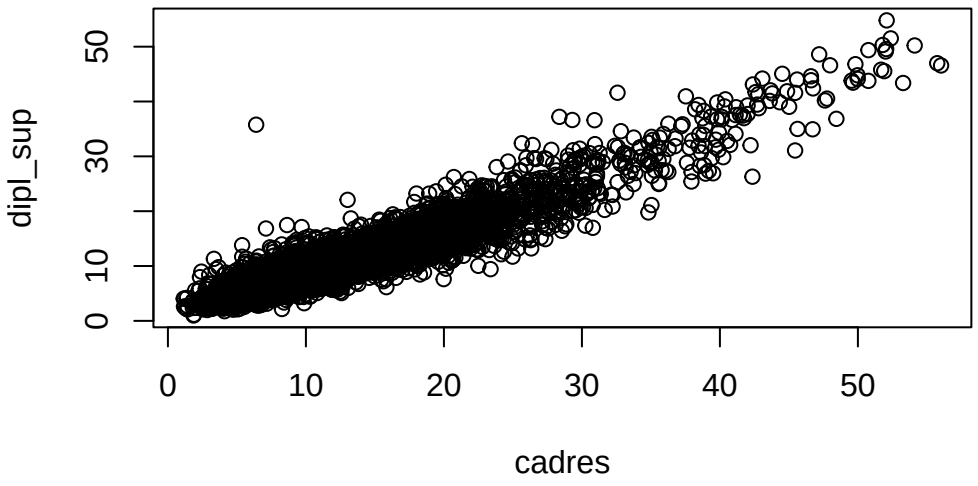


Figure 1.21.: Relation entre le nombre de personnes diplomées à l'université et le nombre de cadre

1.4.4. Exercices

Exercice 1

Dans le jeu de données `hdv2003`, faire le tableau croisé entre la catégorie socio-professionnelle (variable `qualif`) et le fait de croire ou non en l'existence des classes sociales (variable `clso`). Identifier la variable indépendante et la variable dépendante, et calculer les pourcentages ligne ou colonne. Interpréter le résultat.

Solution

```
library(questionr)
data(hdv2003)
tab <- table(hdv2003$qualif, hdv2003$clso)

## Ici la variable indépendante est `qualif`, on calcule donc
## les pourcentages lignes
lprop(tab)
```

Représenter ce tableau croisé sous la forme d'un `mosaicplot` en colorant les cases selon les résidus du test du ^{2}.

Solution

```
mosaicplot(tab, shade = TRUE)
```

Exercice 2

Toujours sur le jeu de données `hdv2003`, faire le boxplot qui croise le nombre d'heures passées devant la télévision (variable `heures.tv`) avec le statut d'occupation (variable `occup`).

Solution

```
boxplot(hdv2003$heures.tv ~ hdv2003$occup)
```

Calculer la durée moyenne devant la télévision en fonction du statut d'occupation à l'aide de `tapply`.

Solution

```
tapply(hdv2003$heures.tv, hdv2003$occup, mean, na.rm = TRUE)
```

Exercice 3

Sur le jeu de données `rp2012`, représenter le nuage de points croisant le pourcentage de personnes sans diplôme (variable `dipl_aucun`) et le pourcentage de propriétaires (variable `proprio`).

Solution

```
library(questionr)
data(rp2012)
plot(rp2012$dipl_aucun, rp2012$proprio)
## ou
plot(proprio ~ dipl_aucun, data = rp2012)
```

1.5. Gérer les données

1.5.1. Importer et exporter des données

Il existe de multiple format four sauvegarder les données, les 2 plus utiles sont `.csv` et `.Rdata`. Les fichiers `.csv` sont utilisés pour stocker des données. Ils sont ouvrables par les éditeurs de texte (e.g. Word, Writer, atom, …) et les tableurs (e.g. MS Excel, LO Calc). Ils sont lus avec la fonction `read.csv` et créés

avec `write.csv`. Les fichiers `.Rdata` sont utilisés pour stocker n'importe quel objet R pas uniquement des données. Cependant, ces fichiers ne peuvent être lus et utilisés que par R. Ces fichiers sont lus avec la fonction `load` et créés avec la fonction `save`.

Les données pour les exercices de laboratoire et pour les devoirs vous sont fournies en format `.csv`.

1.5.1.1. Répertoire de travail

> **Warning**
>
> Une des erreurs les plus communes lorsque l'on débute avec R est lié au chargement des données et la lecture de fichier externe à R.

Un message d'erreur typique est:

```
Error in file(file, "rt") : cannot open the connection
In addition: Warning message:
In file(file, "rt") :
  cannot open file 'ou_est_mon_fichier.csv': No such file or directory
```

L'erreur est du au fait que R ne sache pas où trouver le fichier. Par défaut lorsqu'on ouvre R, R utilise le dossier utilisateur sur l'ordinateur comme dossier de travail. Cela signifie que R va cherhcer à lire les fichiers dans ce dossier et écrire les nouveaux fichiers dans ce dossier. Ceci n'est pas toujours pratique surtout lorsque l'on débute avec R. Pour lire/écrire un fichier dans un endroit particulier sur l'ordinateur, il faut spécifier à R le chemin de cet endroit. Cela peut ce faire de 3 manières différentes:

1. avec la fonction `file.choose()`. La fonction ouvrira une boîte de dialogue vous permettant d'aller choisir un fichier sur votre ordinateur. Si cette option semble très attirante de part sa simplicité, je ne recommande pas de s'en servir car elle ne permet pas de reproduire l'analyse facilement. En effet, elle nécessite de choisir le document chaque fois que l'on souhaite l'utiliser.
2. en spécifiant le chemin complet du fichier dans la commande. Par example `"/home/julien/Documents/cours/Bl`
   C'est assez long à taper et surtout cela ne permet pas de facilement utliser le code sur un autre ordinateur.
3. en spécifiant un répertoire de travail avec la fonction `setwd()`. Ceci indique à R de chercher et d'écrire les fichiers dans un dossier en particulier. Le chemin des fichiers est toujours interprété de manière relative au répertoire de travail. Cela à l'avantage de pouvoir facilement utiliser le même code sur plusieurs ordinateur ssi la structure du dossier est la même.

POur connaitre le répertoire de travail de R il faut utiliser la fonction `getwd()`. La fonction `setwd()` permet de spécifier le chemin du dossier à utiliser comme répertoire de travail.

> **Note**
>
> Si vous ouvrez `RStudio` en double-cliquant sur un fichier `.R` alors `Rstudio` utlisera le dossier où ce fichier est présent comme répertoire de travail. Plutôt pratique car cela évite d'avoir à utiliser la fonction `setwd()`.

> **Important**
>
> Pour l'ensemble des laboratoire du cours, je suggère de créer un dossier dans lequel seront sauvegardés tous les scripts d'analyses et de sauvegardés tous les fichiers de données dans un sous dossier `data`. Le code du labo est structuré de cette manière. C'est pourquoi tous les codes de chargement ou d'écriture de données seront du type `data/mon_fichier.xxx`.

1.5.1.2. Ouvrir un fichier de données en format `.Rdata`

Pour ouvrir ces fichiers, vous pouvez cliquer dessus et laisser votre système d'exploitation démarrer une nouvelle session de R ou encore, à partir de la console de R, utliser la fonction `load` avec le nom et le chemin du fichier de données. Par example, pour ouvrir le fichier `ErablesGatineau.Rdata` qui se situe dans le dossier `data` du dossier de travail, il faut taper:

```
load("data/ErablesGatineau.Rdata")
```

1.5.1.3. Ouvrir un fichier de données en format `.csv`

Pour importer ces données en format `.csv` dans R, il faut utiliser la commande `read.csv()`. Par exemple, pour créer un objet R `erables` qui contient les données du fichier `ErablesGatineau.csv`, il faut utiliser la commande suivant.

```
erables <- read.csv("data/ErablesGatineau.csv")
```

> **Warning**
>
> Attention si vous travaillez dans une langue utilisant la virgule au lieu du point décimal. Par défaut, R utilise le point décimal et vous n'obtiendrez pas le résultat escompté. Il existe une version modifiée de `read.csv()` appelée `read.csv2()` qui règle ce problème. Googlez-la si vous en avez besoin.

Pour vérifier si les données ont bel et bien été lues, vous pouvez lister les objets en mémoire avec la fonction `ls()` ou en obtenir une liste avec une description plus détaillée avec `ls.str()`.

> **Note**
>
> Je vous déconseille cependant, la fonction `ls.str()` car elle peut produire des sorties extrèmementn longue si vous avez beaucoup d'objet dans l'environnement R. Je vous suggère donc d'utliser `ls()` et ensuite `str()` sur l'objet qui vous intéresse.

```
ls()
```

```
 [1] "anglais"        "chien"          "d"              "d_nonsport"
 [5] "d_sport"        "diplome"        "erables"        "hdv2003"
 [9] "imc"            "maths"          "p"              "poids"
[13] "precipitations" "reg"            "resultat"       "rp2012"
[17] "s"              "sport"          "tab"            "taille1"
[21] "taille2"        "taille3"        "taille4"        "taille5"
[25] "tailles"        "tailles_m"      "temperature"    "title"
[29] "x"              "y"
```

```
str(erables)
```

```
'data.frame':   100 obs. of  3 variables:
 $ station: chr  "A" "A" "A" "A" ...
 $ diam   : num  22.4 36.1 44.4 24.6 17.7 ...
 $ biom   : num  732 1171 673 1552 504 ...
```

R confirme avoir en mémoire l'objet `erables`. `erables` est un tableau de données rectangulaire (data.frame) contenant 100 observations (lignes) de 3 variables (colonnes): `station`, une variable de type Facteur avec 2 niveaux, et `diam` et `biom` qui sont 2 variables numériques.

1.5.1.4. Entrer des données

R n'est pas un environnement idéal pour entrer des données. C'est possible, mais la syntaxe est lourde et peut inciter à s'arracher les cheveux. Utilisez votre chiffrier préféré pour faire l'entrée de données. Ce sera plus efficace et moins frustrant.

1.5.1.5. Nettoyer/corriger des données

Une autre opération qui peut être frustrante en R. Mon conseil : ne le faites pas là. Retournez au fichier original, faites la correction, puis re-exportez les données vers R. Il est finalement plus simple de refaire exécuter les quelques lignes de code par la machine. Vous aurez à la fin une seule version (corrigée) de vos données et un code qui vous permet de refaire votre analyse.

1.5.1.6. Exporter des données à partir de R.

Vous pouvez utiliser la fonction,

```
write.csv(mydata, file = "outfilename.csv", row.names = FALSE)
```

où `mydata` est le nom du base de données à exporter et `outfilename.csv` est le nom du fichier à produire. Notez que ce fichier sera créé dans le répertoire de travail (qui peut être changé par le menu à `File>Change dir`, ou par la commande `setwd()`)

1.5.2. Examen préliminaire des données

La première étape de toute analyse est l'examen des données. Elle nous permet de découvrir si on a bien importé les données, si les nombres enregistrés sont possibles, si toutes les données ont bien été lues, etc. L'examen préliminaire des données permet souvent aussi d'identifier des observations suspectes, possiblement dûes à des erreurs d'entrée de donnée. Finalement, l'examen graphique préliminaire permet en général de visualiser les tendances principales qui seront confirmées par l'analyse statistique en tant que telle. Le fichier `sturgeon.csv` contient les données d'une étude effectuée sur les esturgeons de la rivière Saskatchewan. Ces données ont été récoltées, entre autres, pour examiner comment la taille des esturgeons varie entre les sexes (sex), les sites (location), et les années (year).

- Chargez les données du fichier `sturgeon.csv` dans un objet `sturgeon`.
- Pour obtenir un aperçu des éléments du fichier qui ont été chargés en mémoire, taper la commande `str(sturgeon)`.

```
sturgeon <- read.csv("data/sturgeon.csv")
str(sturgeon)
```

```
'data.frame':   186 obs. of  9 variables:
 $ fklngth : num  37 50.2 28.9 50.2 45.6 ...
 $ totlngth: num  40.7 54.1 31.3 53.1 49.5 ...
 $ drlngth : num  23.6 31.5 17.3 32.3 32.1 ...
 $ rdwght  : num  15.95 NA 6.49 NA 29.92 ...
 $ age     : int  11 24 7 23 20 23 20 7 23 19 ...
 $ girth   : num  40.5 53.5 31 52.5 50 54.2 48 28.5 44 39 ...
 $ sex     : chr  "MALE" "FEMALE" "MALE" "FEMALE" ...
 $ location: chr  "THE_PAS" "THE_PAS" "THE_PAS" "THE_PAS" ...
 $ year    : int  1978 1978 1978 1978 1978 1978 1978 1978 1978 1978 ...
```

1.5.2.1. Sommaire statistique

Pour un sommaire du contenu du base de données appelé sturgeon qui est en mémoire, taper la commande

```
summary(sturgeon)
```

```
    fklngth         totlngth        drlngth          rdwght      
 Min.   :24.96   Min.   :28.15   Min.   :14.33   Min.   : 4.73  
 1st Qu.:41.00   1st Qu.:43.66   1st Qu.:25.00   1st Qu.:18.09  
 Median :44.06   Median :47.32   Median :27.00   Median :23.10  
 Mean   :44.15   Mean   :47.45   Mean   :27.29   Mean   :24.87  
 3rd Qu.:48.00   3rd Qu.:51.97   3rd Qu.:29.72   3rd Qu.:30.27  
 Max.   :66.85   Max.   :72.05   Max.   :41.93   Max.   :93.72  
                 NA's   :85      NA's   :13      NA's   :4      
      age            girth           sex              location  
```

```
 Min.   : 7.00   Min.   :11.50   Length:186         Length:186
 1st Qu.:17.00   1st Qu.:40.00   Class :character   Class :character
 Median :20.00   Median :44.00   Mode  :character   Mode  :character
 Mean   :20.24   Mean   :44.33
 3rd Qu.:23.50   3rd Qu.:48.80
 Max.   :55.00   Max.   :73.70
 NA's   :11      NA's   :85
      year
 Min.   :1978
 1st Qu.:1979
 Median :1979
 Mean   :1979
 3rd Qu.:1980
 Max.   :1980
```

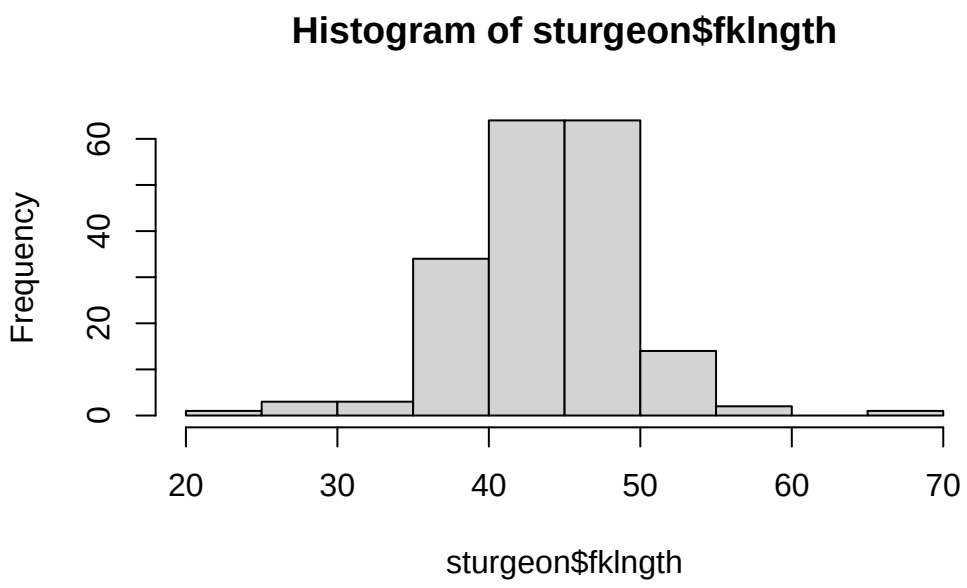
Pour chaque variable, R donne le minimum, le maximum, la médiane qui est la valeur au milieu de la liste des observations ordonnées (appelée le 50 ième percentile), ici, la 93 ième valeur des 186 observations, les valeurs au premier (25%) et troisième quartile (75%), et si il y a des valeurs manquantes dans la colonne. Notez que plusieurs des variables ont des observations manquantes (NA). Donc, seules les variables fklngth (longueur à la fourche), sex, location et year ont 186 observations.

> ⚠ Warning
>
> **Attention aux valeurs manquantes** Plusieurs fonctions de R y réagissent mal et on doit souvent faire les analyses sur des sous- ensembles sans valeur manquante, par des commandes ou des options dans les commandes. On y reviendra, mais prenez l'habitude de noter mentalement si il y a des données manquantes et de vous en rappeler en faisant l'analyse.

1.5.2.2. Histogramme, densité de probabilité empirique, boxplot et examen visuel de la normalité

Examinons maintenant de plus près la distribution de fklngth. La commande `hist()` permet de tracer un histogramme de la variable fklngth dans le base de données sturgeon.

```
hist(sturgeon$fklngth)
```



Les données semblent suivre approximativement une distribution normale.

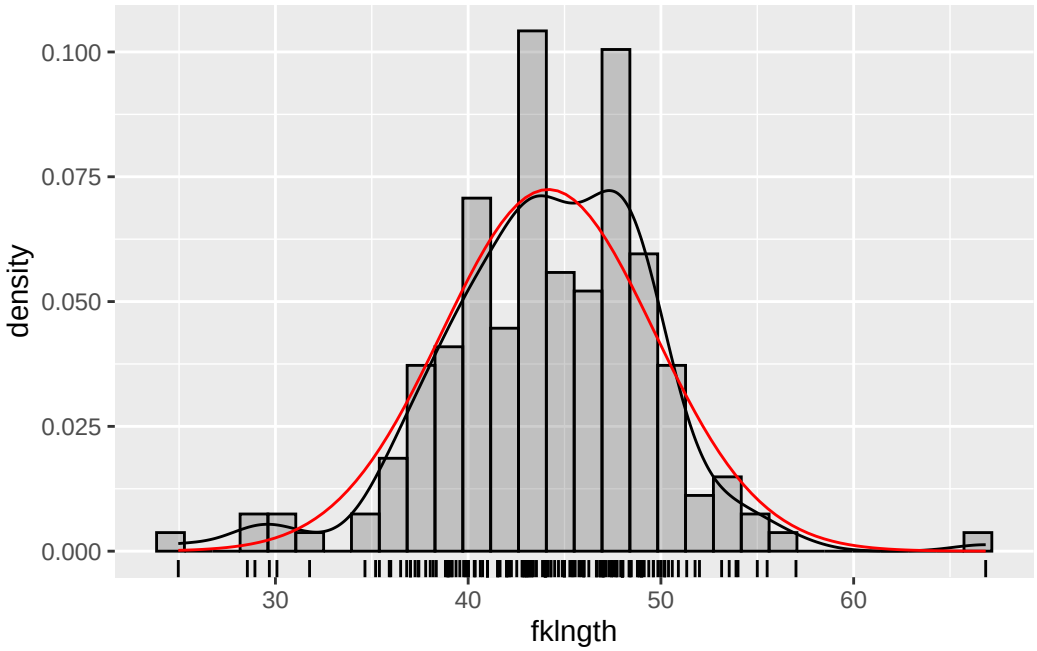
> **Note**
>
> Cette syntaxe peut paraître un peu lourde puisqu'on doit ajouter le préfixe `sturgeon$` devant chaque nom de variable. On pourrait se faciliter la tâche en utilisant la commande `attach()` **mais cela est fortement déconseillé** et jamais utilisé dans ce document.

Cet histogramme est la représentation classique. Mais les histogrammes ne sont pas parfaits. Leur forme dépend en partie du nombre de catégories utilisées, surtout pour les petits échantillons. On peut faire mieux, particulièrement si on est intéressé à comparer visuellement la distribution des observations à une distribution normale. Mais il faut programmer un peu (ou savoir copier-coller…). Le code suivant est un histogramme fait avec l'extension `ggplot2`.

> **Exercice**
>
> Copiez-collez le code suivant dans une nouvelle fenêtre script (File->New script, ou Ctrl-n dans Windows), puis exécutez le.

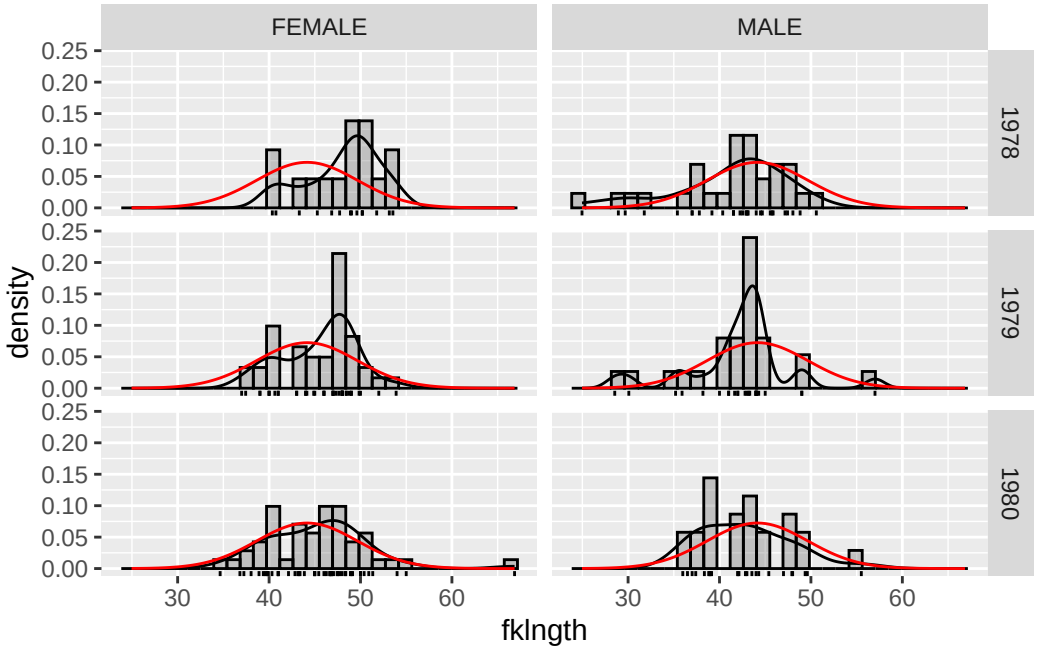
```
## Chargez l'extension ggplot si besoin
library(ggplot2)
##  créer un graphique `mygraph` utilisant les données de "sturgeon"
## et définir l'axe des X comme la longueur `fklngth`
mygraph <- ggplot(data = sturgeon, aes(x = fklngth))

## ajouter différentes parties au graphique
mygraph <- mygraph +
  ## histogramme semi-transparent
  geom_histogram(aes(y = ..density..), bins = 30, color = "black", alpha = 0.3) +
```

```
  ##  line de densité
  geom_density() +
  ##  localisation des observations
  geom_rug() +
  ##  courbe de distribution normale approximé au données
  stat_function(
    fun = dnorm,
    args = list(
      mean = mean(sturgeon$fklngth),
      sd = sd(sturgeon$fklngth)
    ),
    color = "red"
  )

## montrer le graphique
mygraph
```



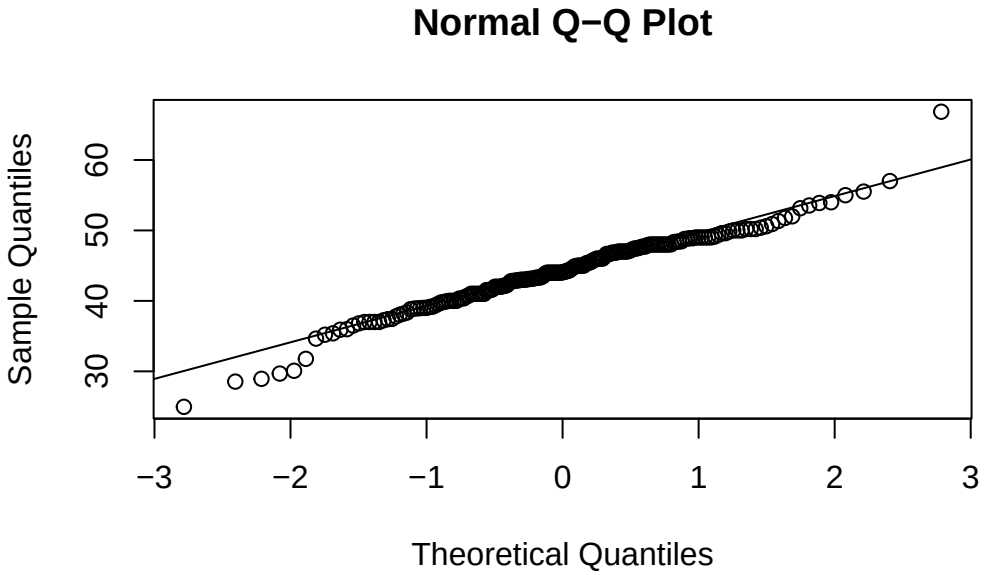
Chaque observation est représentée par une barre sous l'axe des x (rug). En rouge est la distribution normale de données avec la même moyenne et écart-type que les observations. Et l'autre ligne est la densité de probabilité empirique, « lissée » à partir des observations. Si vous êtes plus aventureux, vous pouvez examiner la distribution des observations de fklngth par sous-groupes (par exemple sex et year) avec :

```
mygraph + facet_grid(year ~ sex)
```



Chaque panneau illustre la distribution pour un sexe cette année-là, et la courbe en rouge récurrente représente la distribution normale pour l'ensemble des données. Cette courbe peut servir à mieux évaluer visuellement les différences entre les panneaux. Une autre façon d'évaluer la normalité de données visuellement est de faire un QQ plot avec la paire de commandes `qqnorm()` et `qqline()`.

```
qqnorm(sturgeon$fklngth)
qqline(sturgeon$fklngth)
```



Des données parfaitement normales suivraient la ligne droite diagonale. Ici, il y a des déviations dans les queues de la distribution, et un peu à droite du centre. Comparez cette représentation à celle des deux

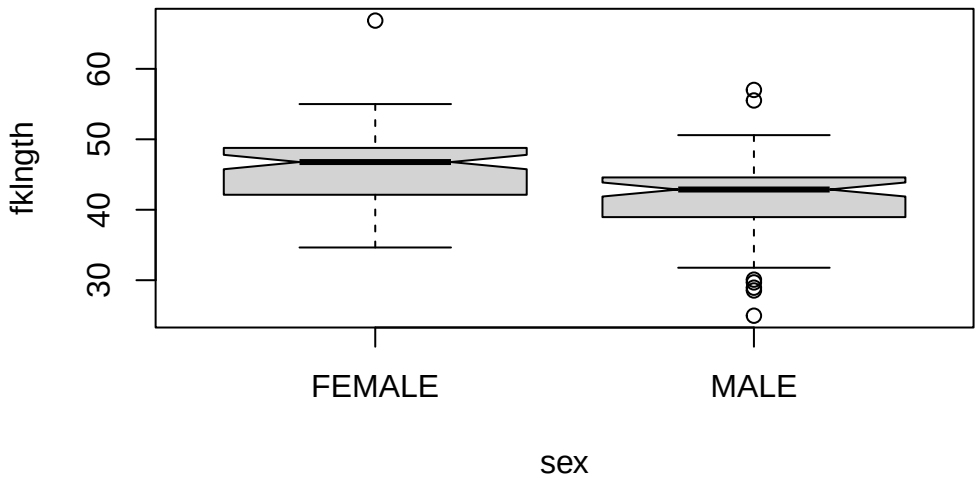
graphiques précédents. Vous conviendrez sans doute avec moi qu'il est plus facile de visualiser comment la distribution dévie de la normalité sur les histogrammes et les graphiques de la densité empirique de probabilité que sur les QQ plots. Ceci dit, les QQ plots sont souvent utilisés et vous devriez être capable de les interpréter. De plus, on peut facilement éprouver statistiquement l'hypothèse que les données sont distribuées normalement avec R par la commande `shapiro.test()` qui calcule une statistique (W) qui est une mesure de la tendance des points d'un QQ plot à former une ligne parfaite. Si oui, alors W=1. Si W s'éloigne de 1 (vers 0), alors les données s'éloignent de la normalité. Ici,

```
shapiro.test(sturgeon$fklngth)
```

```
    Shapiro-Wilk normality test

data:  sturgeon$fklngth
W = 0.97225, p-value = 0.0009285
```

W n'est pas très loin de 1, mais suffisamment pour que la différence soit significative. L'examen visuel des grands échantillons est souvent compliqué par le fait que plusieurs points se superposent et qu'il devient plus difficile de bien visualiser la tendance centrale. Les boxplots avec "moustaches" (box and whiskers plots) offrent une alternative intéressante. La commande `boxplot()` peut produire un boxplot de fklngth pour chaque niveau de sex, et ajoute les coches.

```
boxplot(fklngth ~ sex, data = sturgeon, notch = TRUE)
```



La ligne un peu plus épaisse dans la boîte de la Figure indique la médiane. La coche est proportionnelle à l'incertitude quant à la position de la médiane. On peut visuellement interpréter approximativement les différences entre médianes en examinant si il y a chevauchement entre les coches (ici, il n'y a pas

chevauchement, et on conclurait provisoirement que la médiane de fklngth pour les femelles est supérieure à celle des mâles). Les boîtes s'étendent du premier au troisième quartile (du 25ième au 75ième percentile si vous préférez), Les barres (moustaches ou whiskers) au-dessus et en dessous des boîtes s'étendent soit de la valeur minimum à la valeur maximum, ou, si il y a des valeurs extrêmes, de la plus petite à la plus grande valeur à l'intérieur de 1.5x la largeur de l'étendue interquartile . Enfin, les observations qui excèdent les limites des moustaches (donc à plus de 1.5x l'étendue interquartile de chaque côté de la médiane) sont indiquées par des symboles.Ce sont des valeurs qui pourraient être considérées comme extrêmes et possiblement aberrantes.

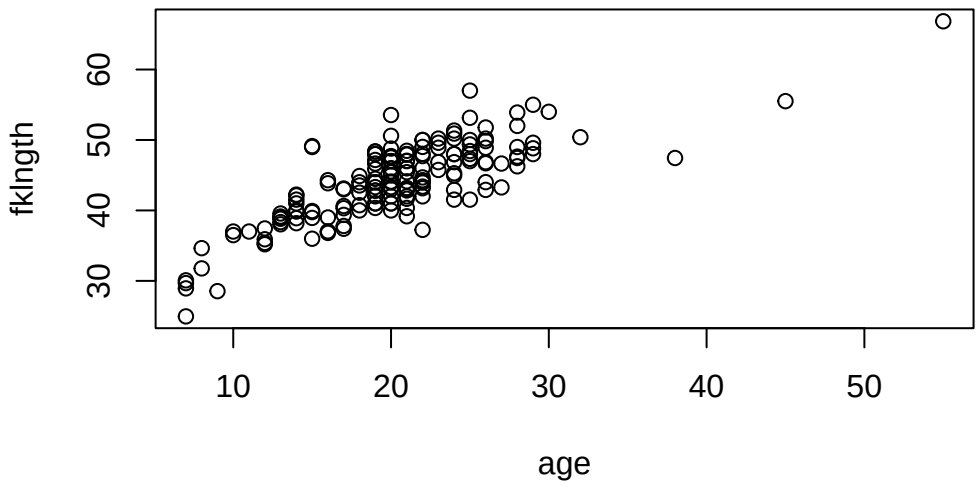
1.5.2.3. Diagrammes de dispersion bivariés

En plus des graphiques pour chacune des variables séparément, il est très souvent intéressant de jeter un coup d'oeil aux diagrammes de dispersion . La commande `plot(y~x)` permet de faire le graphique de y sur l'axe vertical (l'ordonnée) en fonction de x sur l'axe horizontal (l'abscisse).

> Exercice
>
> Faites un graphique de fklngth en fonction de age avec la commande plot.

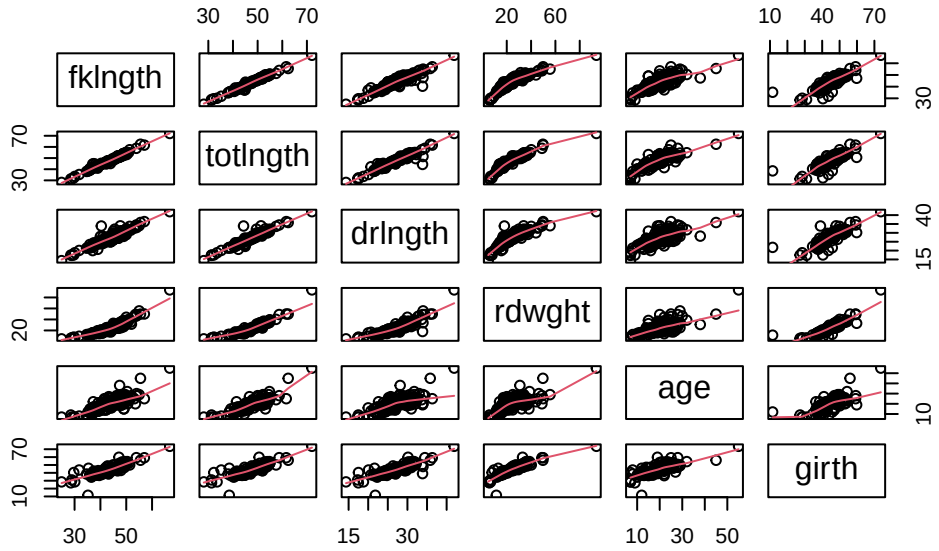
Vous devriez obtenir:

```
plot(fklngth ~ age, data = sturgeon)
```



R a une fonction qui permet la création des graphiques de dispersion de toutes les paires de variables (`pairs()`). Une des option de ¬ est l'ajout d'une trace lowess qui indique la tendance de la relation entre les variables. Pour obtenir la matrice de ces graphiques avec la trace lowess pour toutes les variable dans sturgeon, entrer la commande `pairs(sturgeon[,1:6], panel=panel.smooth)` et vous devriez obtenir

```
pairs(sturgeon[, 1:6], panel = panel.smooth)
```



1.5.3. Créer des sous-ensembles de cas

Il arrive fréquemment qu'une analyse se concentre sur un sous-ensemble des observations contenues dans un fichier de données. Les cas sont d'habitude sélectionnés selon un critère en particulier. Pour utiliser un sous-ensemble de vos données en créant un graphique ou en performant une analyse, on peut utiliser la commande `subset()`. Par exemple, pour créer un sous ensemble des données du tableau sturgeon qui ne contient que les femelles capturées en 1978, on peut écrire :

```
sturgeon_female_1978 <- subset(sturgeon, sex == "FEMALE" & year == "1978")
sturgeon_female_1978
```

```
     fklngth totlngth  drlngth rdwght age girth    sex   location year
2   50.19685 54.13386 31.49606     NA  24  53.5 FEMALE    THE_PAS 1978
4   50.19685 53.14961 32.28346     NA  23  52.5 FEMALE    THE_PAS 1978
6   49.60630 53.93701 31.10236  35.86  23  54.2 FEMALE    THE_PAS 1978
7   47.71654 51.37795 33.97638  33.88  20  48.0 FEMALE    THE_PAS 1978
15  48.89764 53.93701 29.92126  35.86  23  52.5 FEMALE    THE_PAS 1978
105 46.85039       NA 28.34646  23.90  24    NA FEMALE CUMBERLAND 1978
106 40.74803       NA 24.80315  17.50  18    NA FEMALE CUMBERLAND 1978
107 40.35433       NA 25.59055  20.90  21    NA FEMALE CUMBERLAND 1978
109 43.30709       NA 27.95276  24.10  19    NA FEMALE CUMBERLAND 1978
113 53.54331       NA 33.85827  48.90  20    NA FEMALE CUMBERLAND 1978
114 51.77165       NA 31.49606  35.30  26    NA FEMALE CUMBERLAND 1978
116 45.27559       NA 26.57480  23.70  24    NA FEMALE CUMBERLAND 1978
```

```
118 53.14961       NA 32.67717  45.30  25    NA FEMALE CUMBERLAND 1978
119 50.19685       NA 32.08661  33.90  26    NA FEMALE CUMBERLAND 1978
123 49.01575       NA 29.13386  37.50  22    NA FEMALE CUMBERLAND 1978
```

Warning

Dans ces comparaisons, il faut toujours utiliser `==` pour égal à. Dans ce contexte, si vous utilisez `=` seulement, vous n'obtiendrez pas ce que vous désirez. Dans le tableau qui suit se trouve une liste de commandes communes que vous allez probablement utiliser pour créer des expressions en R.

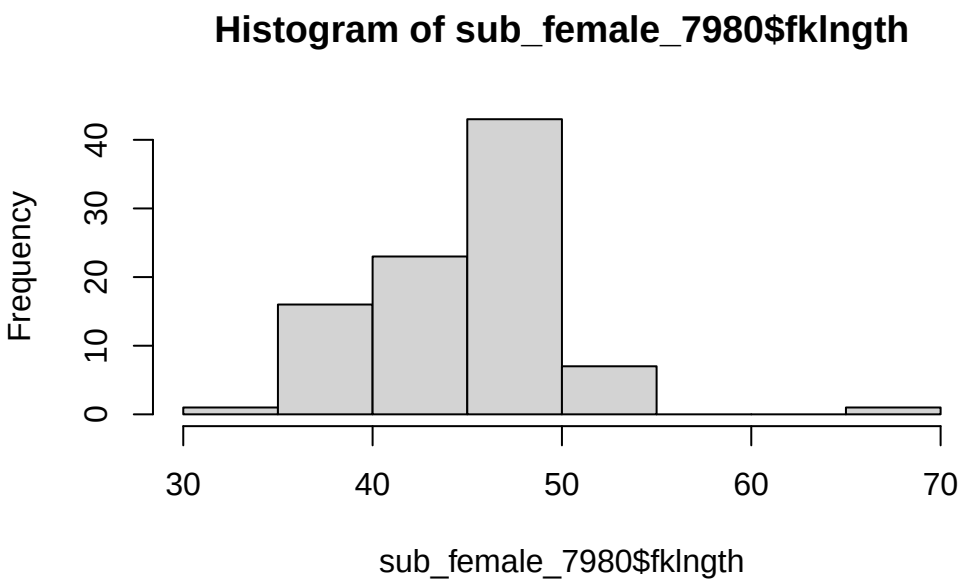
Operateur	Explication	Operateur	Explication
==	Égal à	!=	Pas égal à
>	Plus que	<	Moins que
>=	Plus que ou égal à	<=	Moins que ou égal à
&	Et vectorisé	|	Ou vectorisé
&&	Et contrôle	||	Ou contrôle
!	Pas		

Exercice

En utilisant les commandes `subset()` et `hist()`, essayez de faire un histogramme pour le sous-ensemble de cas correspondant aux femelles capturées en 1979 et 1980 (donc `sex == "FEMALE" & (year == "1979" | year == "1980")`)

Solution

```
sub_female_7980 <- subset(sturgeon, sex == "FEMALE" & (year == "1979" | year == "1980"))
hist(sub_female_7980$fklngth)
```



1.5.4. Transformations de données

Il est très fréquemment nécessaire d'effectuer des transformations mathématiques sur les données brutes pour mieux satisfaire aux conditions d'application de tests statistiques. R étant aussi un langage de programmation complet, il peut donc effectuer les transformations désirées. Les fonctions les plus fréquemment utilisées sont:

- `log()`
- `sqrt()`
- `ifelse()`

On peut employer ces fonctions directement dans les lignes de commandes, ou encore créer de nouvelles variables orphelines ou faisant partie d'un data.frame. Par exemple, pour faire un graphique du logarithme décimal de fklngth en fonction de l'âge, on peut écrire

```
plot(log(fklngth) ~ age, data = sturgeon)
```

Pour créer une variable orpheline (i.e. non incluse dans le data.frame) appelée logfklngth et contenant le logarithme décimal de fklngth, on peut écrire

::: {.cell}

```
logfklngth <- log10(sturgeon$fklngth)
```

:::

Si on veut ajouter cette variable transformée à un tableau de données (data.frame), alors, on doit préfixer le nom de la variable par le nom du base de données et du symbole `$`, par exemple, pour ajouter une variable nommée `lfkl` contenant le log10 de fklngth au tableau sturgeon, on peut écrire:

```
sturgeon$logfkl <- log10(sturgeon$fklngth)
```

N'oubliez pas de sauvegarder ce tableau modifié si vous voulez avoir accès à cette nouvelle variable dans le futur. Pour les transformations conditionnelles, on peut utiliser la fonction `ifelse()`. Par exemple, pour créer une nouvelle variable appelée dummy qui sera égale à 1 pour les mâles et 0 pour les femelles, on peut écrire:

```
sturgeon$dummy <- ifelse(sturgeon$sex == "MALE", 1, 0)
```

1.5.5. Exercice sur R

Vous trouverez dans le fichier `salmonella.csv`, des valeurs numériques du ratio d'infection des cellules par la salmonelle dans deux milieux (IN VITRO et IN VIVO) et pour trois souches différentes de salmonelles. Examinez les données pour le ratio et faites des graphiques pour évaluer la normalité de la distribution des ratios pour la souche SAUVAGE dans les 2 milieux combinés et produire un graphique.

Solution

```r
## Charger les données
salmonella <- read.csv("data/salmonella.csv")

## creer le graph en utilisant juste la souche sauvage et définir x
mygraph <- ggplot(subset(salmonella, souche == "SAUVAGE"), aes(x = ratio))
## ajouter des composants graphiques
mygraph <- mygraph +
  # line densité
  geom_density() +
  # position des observations
  geom_rug() +
  # histogramme
  geom_histogram(aes(y = ..density..),
    bins = 30,
    color = "black",
    alpha = 0.3
  ) +
  # distribution normal ajustée
  stat_function(
    fun = dnorm,
    args = list(
      mean = mean(subset(salmonella, souche == "SAUVAGE")$ratio),
      sd = sd(subset(salmonella, souche == "SAUVAGE")$ratio)
    ),
    color = "red"
  )
## faire le graphique
mygraph
```

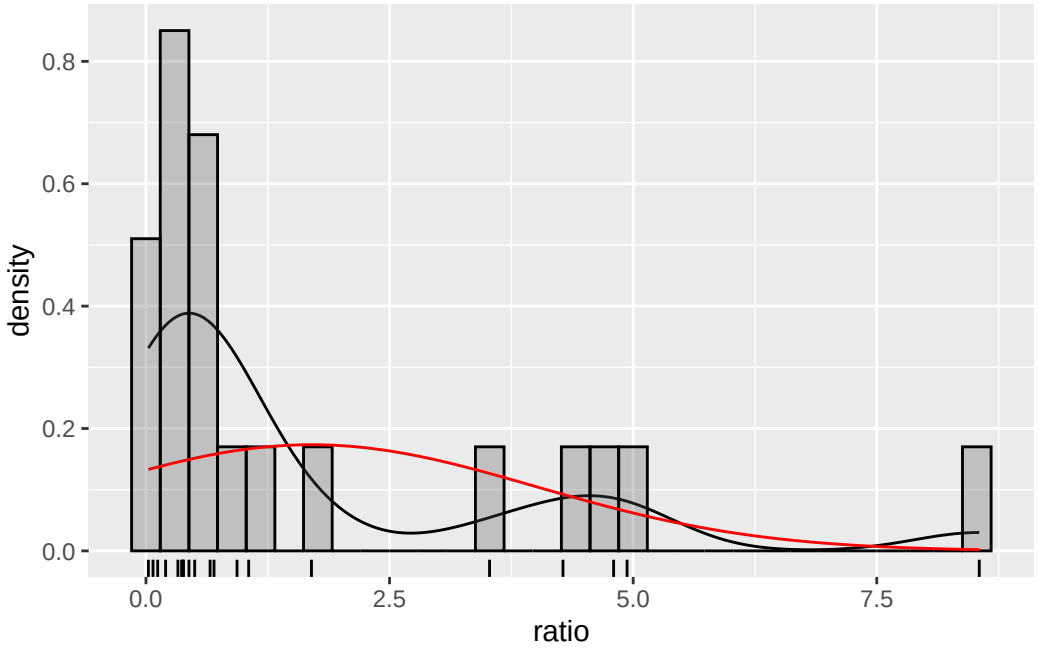


Figure 1.22.: Distibution des ratios d'infections par la souche sauvage de salmonelle

2. Analyse de puissance avec R et G*Power

Après avoir complété cet exercice de laboratoire, vous devriez :

- Pouvoir calculer la puissance d'un test de t avec R et G*Power
- Pouvoir calculer l'effectif requis pour obtenir la puissance désirée avec un test de t
- Pouvoir calculer la taille de l'effet détectable par un test de t étant donné l'effectif, la puissance et α
- Comprendre comment la puissance change lorsque l'effectif augmente, la taille de l'effet change, ou lorsque α diminue
- Comprendre comment la puissance est affectée lorsque l'on passe d'un test bilatéral à un test unilatéral

2.1. La théorie

2.1.1. Qu'est-ce que la puissance?

La puissance est la probabilité de rejeter l'hypothèse nulle quand elle est fausse.

2.1.2. Pourquoi faire une analyse de puissance?

Évaluer l'évidence

L'analyse de puissance effectuée après avoir accepté une hypothèse nulle permet de calculer la probabilité que l'hypothèse nulle soit rejetée si elle était fausse et que la taille de l'effet était d'une valeur donnée. Ce type d'analyse a posteriori est très commun.

Planifier de meilleures expériences

L'analyse de puissance effectuée avant de réaliser une expérience (le plus souvent après une expérience préliminaire cependant), permet de déterminer le nombre d'observations nécessaires pour détecter un effet d'une taille donnée à un niveau fixe de probabilité (la puissance). Ce type d'analyse a priori devrait être réalisé plus souvent.

Estimer la "limite de détection" statistique

L'effort d'échantillonnage est souvent déterminé à l'avance (par exemple lorsque vous héritez de données récoltées par quelqu'un d'autre), ou très sévèrement limité (lorsque les contraintes logistiques prévalent). Que ce soit a priori ou a posteriori l'analyse de puissance vous permet d'estimer, pour un effort d'échantillonnage donné et un niveau de puissance fixe, quelle est la taille minimale de l'effet qui peut être détecté (comme étant statistiquement significatif).

2.1.3. Facteurs qui affectent la puissance

Il y a 3 facteurs qui affectent la puissance d'un test statistique.

Le critère de décision

La puissance dépend de α, le seuil de probabilité auquel on rejette l'hypothèse nulle. Si ce seuil est très strict (*i.e.* si α est fixé à une valeur très basse, comme 0.1% ou p = 0.001), alors la puissance sera plus faible que si le seuil était moins strict.

La taille de l'échantillon

Plus l'échantillon est grand, plus la puissance est élevée. La capacité d'un test à détecter de petites différences comme étant statistiquement significatives augmente avec une augmentation du nombre d'observations.

La taille de l'effet

Plus la taille de l'effet est grande, plus un test a de puissance. Pour un échantillon de taille fixe, la capacité d'un test à détecter un effet comme étant statistiquement significatif est plus élevée si l'effet est grand que s'il est petit. La taille de l'effet est en fait une mesure du degré de fausseté de l'hypothèse nulle.

2.2. Qu'est ce que G*Power?

G*Power est un programme gratuit, développé par des psychologues de l'Université de Dusseldorf en Allemagne. Le programme existe en version Mac et Windows. Il peut cependant être utilisé sous linux via Wine. G*Power vous permettra d'effectuer une analyse de puissance pour la majorité des tests que nous verrons au cours de la session sans avoir à effectuer des calculs complexes ou farfouiller dans des tableaux ou des figures décrivant des distributions ou des courbes de puissance. Il est possible de faire tous les analyses de G*power avec R, mais cela est nettement plus complexes, car il faut tous coder à la main. Dans les cas les plus simple le code R est aussi fourni. G*power est vraiment un outil très utile que vous devrez maîtriser.

> Exercice
>
> Téléchargez le programme **ici** et installez-le sur votre ordi et votre station de travail au laboratoire (si ce n'est déjà fait).

2.3. Comment utiliser G*Power

2.3.1. Principe général

L'utilisation de G*Power implique généralement en trois étapes:

1. Choisir le test approprié

2. Choisir l'un des 5 types d'analyses de puissance disponibles
3. Inscrire les valeurs des paramètres requis et cliquer sur Calculate

2.3.2. Types d'analyses de puissance disponibles

A priori

Calcule l'effectif requis pour une valeur de α, β et de taille d'effet donnée. Ce type d'analyse est utile à l'étape de planification des expériences.

Compromis

Calcule α et β pour un rapport β/α donné, un effectif fixe, et une taille d'effet donnée. Ce type d'analyse est plus rarement utilisé (je ne l'ai jamais fait), mais peut être utile lorsque le rapport β/α est d'intérêt, par exemple lorsque le coût d'une erreur de type I et de type II peut être quantifié.

Critère

Calcule α pour β, effectif et taille d'effet donné. En pratique, je vois peu d'utilité pour ce type de calcul. Contactez-moi si vous en voyez une!

Post-hoc

Calcule la puissance (1 - β) pour α, une taille d'effet et un effectif donné. Très utilisée pour interpréter les résultats d'une analyse statistique non-significative, mais seulement si l'on utilise une taille d'effet biologiquement significative (et non la taille d'effet observée). Peu pertinente lorsque le test est significatif.

Sensitivité

Calcule la taille d'effet détectable pour une valeur d'α, β et un effectif donné. Très utile également au stade de planification des expériences.

2.3.3. Comment calculer la taille de l'effet G*Power permet de faire une analyse de puissance pour de nombreux tests statistiques

L'indice de la taille de l'effet qui est utilisé par G*Power pour les calculs dépend du test. Notez que d'autres logiciels peuvent utiliser des indices différents et il est important de vérifier que l'indice que l'on utilise est celui qui convient. G*Power vous facilite la tâche et permet de calculer la taille de l'effet en inscrivant seulement les valeurs pertinentes dans la fenêtre de calcul. Le tableau suivant donne les indices utilisés par G*Power pour les différents tests.

Test	Taille d'effet	Formule
test de t sur des moyennes	d	$d = \frac{\lvert\mu_1 - \mu_2\rvert}{\sqrt{({s_1}^2 + {s_2}^2)/2}}$
test de t pour des corrélations	r	
autres tests de t	d	$d = \frac{\mu}{\sigma}$
test F (ANOVA)	f	$f = \frac{\sqrt{\frac{\sum_{i=1}^{k}(\mu_i - \mu)^2}{k}}}{\sigma}$
autres test F	f^2	$f^2 = \frac{{R_p}^2}{1 - {R_p}^2}$ R_p est le coefficient de corrélation partiel
test Chi-carré	w	$w = \sqrt{\sum_{i=1}^{m} \frac{(p_{0i} - p_{1i})^2}{p_{0i}}}$ p_{0i} p_{1i} sont les proportions de la catégorie i prédites par l'hypothèse nulle et alternative respectivement

2.4. Puissance pour un test de t comparant deux moyennes

> **!** Important
>
> L'ensemble des analyses de puissance décrites après peuvent être réalisé avec 2 fonctions dans R.
>
> - `pwr.t.test()`: lorsque les tailles d'échantillons sont identiques
> - `pwr.t2n.test()`: lorsque les échantillons ont des tailles différentes

L'objectif de cette séance de laboratoire est de vous familiariser avec G*Power et de vous aider à comprendre comment les quatre paramètres des analyses de puissance (α, β, effectif et taille de l'effet) sont reliés entre eux. On examinera seulement un des nombreux tests, le test de t permettant de comparer deux moyennes indépendantes. C'est le test le plus communément utilisé par les biologistes, vous l'avez tous déjà utilisé, et il conviendra très bien pour les besoins de la cause. Ce que vous apprendrez aujourd'hui s'appliquera à toutes les autres analyses de puissance que vous effectuerez à l'avenir.

Jaynie Stephenson a étudié la productivité des ruisseaux de la région d'Ottawa. Elle a, entre autres, quantifié la biomasse des poissons dans 18 ruisseaux sur le Bouclier Canadien d'une part, et dans 18 autres ruisseaux de la vallée de la rivière des Outaouais et de la rivière Rideau d'autre part. Elle a observé une biomasse plus faible dans les ruisseaux de la vallée (2.64 g/m^2, écart-type=3.28) que dans ceux du Bouclier (3.31 g/m^2, écart-type=2.79). En faisant un test de t pour éprouver l'hypothèse nulle que la biomasse des poissons est la même dans les deux régions, elle obtient:

```
Pooled-Variance Two-Sample t-Test
t = -0.5746, df = 34, p-value = 0.5693
```

Elle accepte l'hypothèse nulle (puisque p est plus élevé que 0.05) conclue donc que la biomasse moyenne des poissons est la même dans ces deux régions.

2.4.1. Analyse post-hoc

Compte tenu des valeurs des moyennes observées et de leur écart- type, on peut utiliser G*Power pour calculer la puissance du test de t bilatéral pour deux moyennes indépendantes et pour la taille d'effet (i.e. la différence entre la biomasse entre les deux régions, pondérée par les écarts-type) à $\alpha = 0.05$.

Démarrer G*Power.

1. À **Test family**, choisir: t tests
2. À **Statistical test**, choisir: Means: Difference between two inde- pendent means (two groups)
3. À **Type of power analysis**, choisir: Post hoc: Compute achieved power - given α, sample size, and effect size
4. Dans **Input Parameters**,

- à la boîte **Tail(s)**, choisir: Two,
- vérifier que α **err prob** est égal à 0.05
- inscrire 18 pour **Sample size group** 1 et 2
- pour calculer la taille d'effet (Effect size d), cliquer sur le bouton **Determine =>**

5. Dans la fenêtre qui s'ouvre à droite, sélectionner **n1 = n2**

- entrer les moyennes (**Mean group** 1 et 2)
- entrer les écarts types (**SDs group** 1 et 2)
- cliquer sur le bouton **Calculate and transfer to main window**

6. Cliquer sur le bouton Calculate dans la fenêtre principale et vous devriez obtenir ceci:

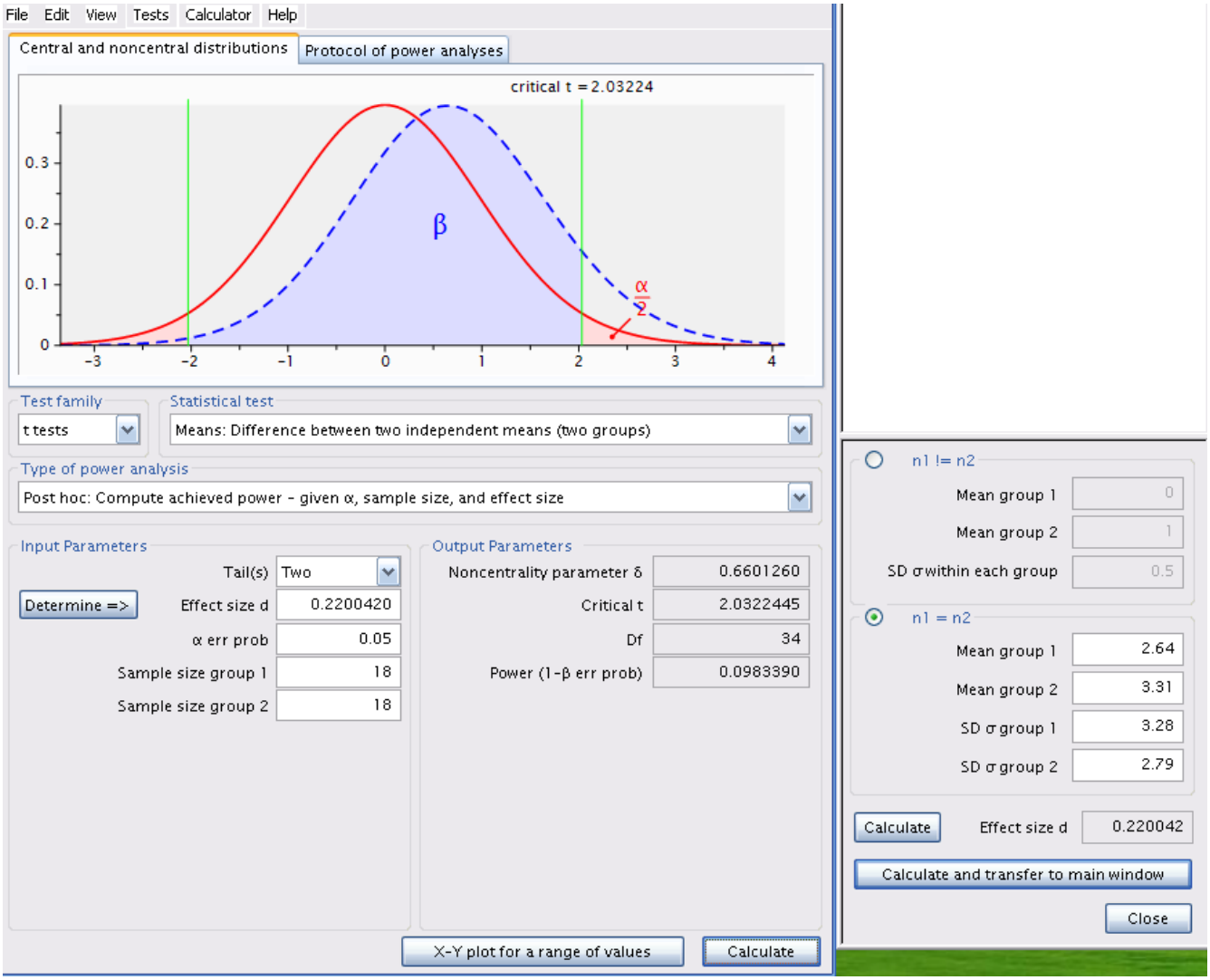


Figure 2.1.: Analyse post-hoc avec la taille d'effet estimée

La même analyse peut être faites en utlisant R. Pour l'analyse avec R, il faut définir d'abord calculer la taille d'effet `d` pour un test de t comparant deux moyennes, puis utiliser la fonction `pwr.t.test` de l'extension `pwr`. Le plus simple comme nous allons estimer la taille d'effet `d` plusieurs fois et de créer une petite fonction qui estime `d` basé sur les paramètres nécessaires.

```
# charger l'extension pwr
library(pwr)
# définir une fonction pour d
d <- function(u1, u2, sd1, sd2) {
  abs(u1 - u2) / sqrt((sd1^2 + sd2^2) / 2)
}

# analyse de puissance
pwr.t.test(n = 18, d = d(u1 = 2.64, sd1 = 3.28, u2 = 3.31, sd2 = 2.79), sig.level = 0.05, type =
```

```
     Two-sample t test power calculation

              n = 18
              d = 0.220042
      sig.level = 0.05
          power = 0.09833902
    alternative = two.sided

NOTE: n is number in *each* group
```

```r
# graphique comme g*Power
x <- seq(-4, 4, length = 200)
plot(x, dnorm(x), type = "l", col = "red", lwd = 2)
qc <- qt(0.025, 16)
abline(v = qc, col = "green")
abline(v = -qc, col = "green")
lines(x, dnorm(x, mean = (3.31 - 2.64)), type = "l", col = "blue", lwd = 2)

# power corresponds to the shaded area
y <- dnorm(x, mean = (3.31 - 2.64))
polygon(c(x[x <= -qc], -qc), c(y[x <= -qc], 0), col = rgb(red = 0, green = 0.2, blue = 1, alpha =
```

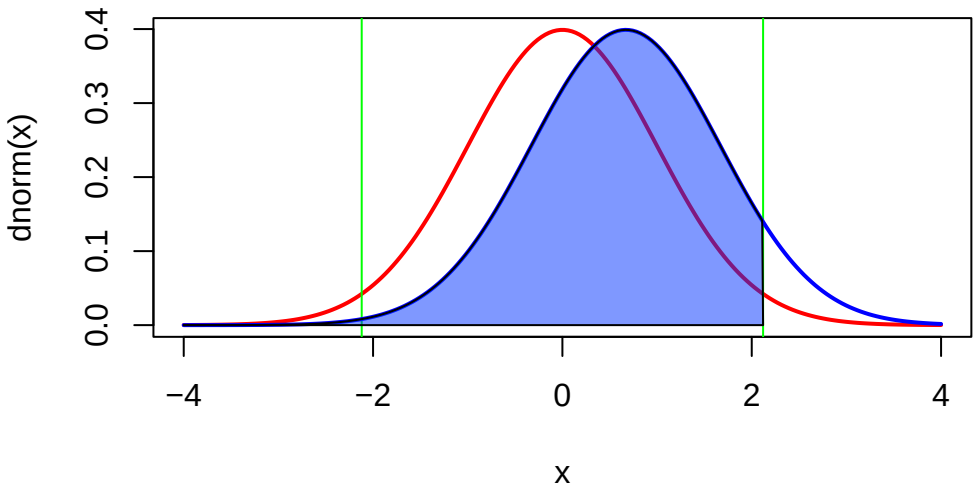


Figure 2.2.: Graphique de l'analyse de puissance dans R

Étudions un peu la figure @ref(fig:gpower-1).

- La courbe de gauche, en rouge, correspond à la distribution de la statistique t si H_0 est vraie (i.e si les deux moyennes étaient égales) compte tenu de l'effectif (18 dans chaque région) et des écarts-types observés.

- Les lignes verticales vertes correspondent aux valeurs critiques de t pour une valeur $\alpha = 0.05$ et un effectif total de 36 (2x18).
- Les régions ombrées en rose correspondent aux zones de rejet de H_0. Si Jaynie avait obtenu une valeur de t en dehors de l'intervalle délimité par les valeurs critiques allant de -2.03224 à 2.03224, alors elle aurait rejeté H_0, l'hypothèse nulle d'égalité des deux moyennes. En fait, elle a obtenu une valeur de t égale à -0.5746 et conclu que la biomasse est la même dans les deux régions.
- La courbe de droite, en bleu, correspond à la distribution de la sta- tistique t si H_1 est vraie (ici H_1 correspond à une différence de biomasse entre les deux régions de $3.33 - 2.64 = 0.69 g/m^2$, compte tenu des écarts-types observés). Cette distribution correspond à ce qu'on devrait s'attendre à observer si H_1 était vraie et que l'on répétait un grand nombre de fois les mesures dans des échantillons aléatoires de 18 ruisseaux des deux régions en calculant la statistique t à chaque fois. En moyenne, on observerait une valeur de t d'environ 0.6.
- Notez que la distribution de droite chevauche considérablement celle de gauche, et une bonne partie de la surface sous la courbe de droite se retrouve à l'intérieur de l'intervalle d'acceptation de H_0, délimité par les deux lignes vertes et allant de -2.03224 à 2.03224. Cette proportion, correspondant à la partie ombrée en bleu sous la courbe de droite et dénoté par β correspond au risque d'erreur de type II qui est d'accepter H_0 quand H_1 est vraie.
- La puissance est simplement $1 - \beta$, et est ici de 0.098339. Donc, si la biomasse différait de $0.69 g/m^2$ entre les deux régions, Jaynie n'avait que 9.8% des chances d'être capable de détecter une différence statistiquement significative à $\alpha = 5$ en échantillonnant 18 ruisseaux de chaque région.

Récapitulons: La différence de biomasse entre les deux régions n'est pas statistiquement significative d'après le test de t. C'est donc que cette différence est relativement petite compte tenu de la précision des mesures. Il n'est donc pas très surprenant que la puissance, i.e. la probabilité de détecter une différence significative, soit faible. Toute cette analyse ne nous informe pas beaucoup.

Une analyse de puissance post hoc avec la taille de l'effet observé n'est pas très utile. On la fera plutôt pour une taille d'effet autre que celle observée quand H 0 est acceptée. Quelle taille d'effet utiliser? C'est la biologie du système étudié qui peut nous guider. Par exemple, en ce qui concerne la biomasse des poissons, on pourrait s'attendre à ce qu'une différence de biomasse du simple au double (disons de 2.64 à 5.28 g/m^2) ait des conséquences écologiques. On voudrait s'assurer que Jaynie avait de bonnes chances de détecter une différence aussi grande que celle-là avant d'accepter ses conclusions que la biomasse est la même entre les deux régions. Quelles étaient les chances de Jaynie de détecter une différence de 2.64 g/m^2 entre les deux régions? G*Power peut nous le dire.

Exercice

Changer la moyenne du groupe 2 à 5.28, recalculer la taille d'effet, et cliquer sur Calculate pour obtenir (@ref(fig:gpower-2)).

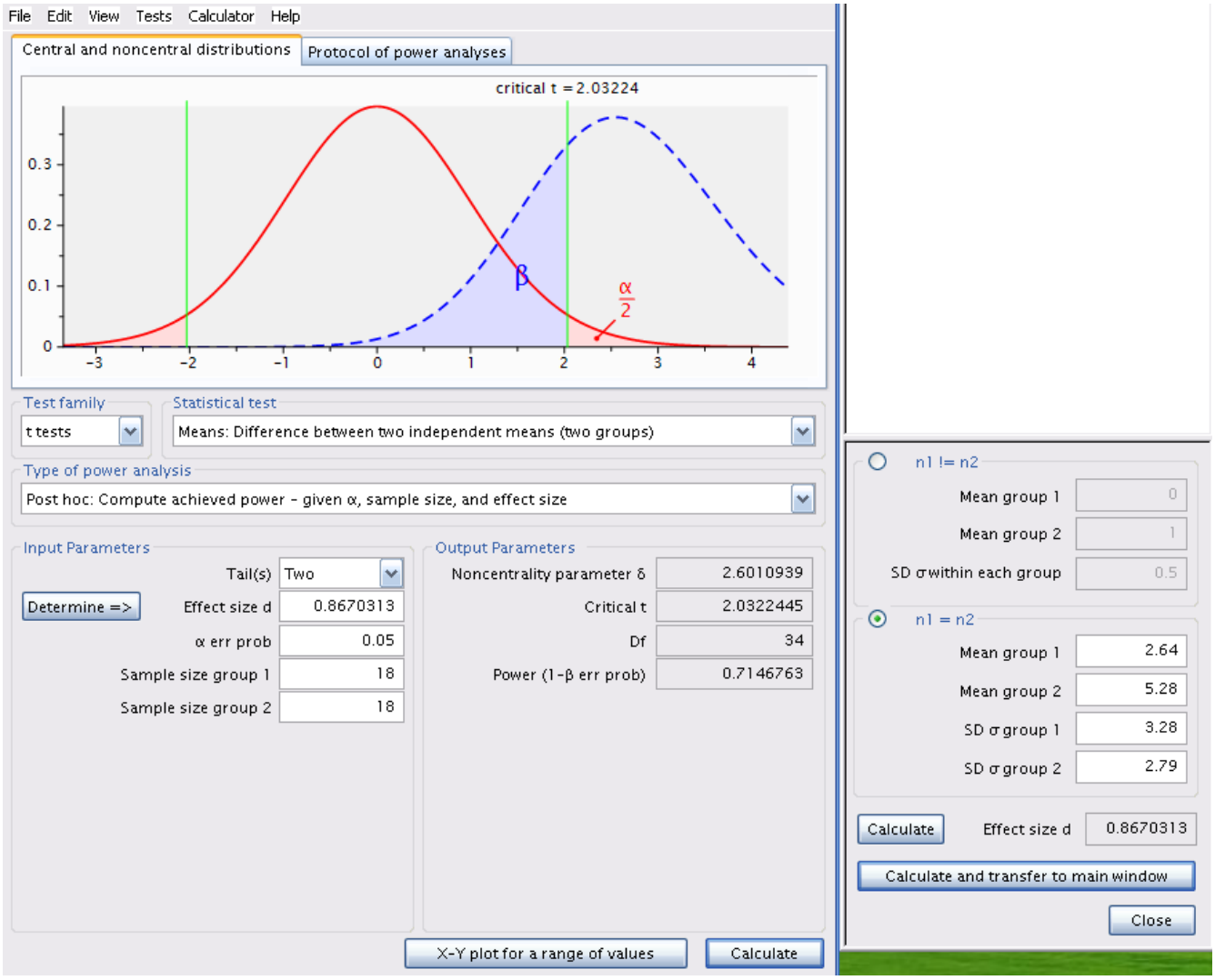


Figure 2.3.: Analyse post-hoc avec une taille d'effet différente

```
pwr.t.test(n = 18, d = d(u1 = 2.64, sd1 = 3.28, u2 = 5.28, sd2 = 2.79), sig.level = 0.05, type =
```

```
     Two-sample t test power calculation

              n = 18
              d = 0.8670313
      sig.level = 0.05
          power = 0.7146763
    alternative = two.sided

NOTE: n is number in *each* group
```

La puissance est de 0.71, donc Jaynie avait une chance raisonnable de détecter une différence du simple au double avec 18 ruisseaux dans chaque région.

Notez que cette analyse de puissance post hoc pour une taille d'effet jugée biologiquement significative est bien plus informative que l'analyse précédente pour la taille d'effet observée (qui est celle effectuée par défaut par bien des néophytes et de trop nombreux logiciels qui essaient de penser pour nous). En effet, Jaynie n'a pu détecter de différences significatives entre les deux régions. Cela pourrait être pour deux raisons: soit qu'il n'y a pas de différences entre les régions, ou soit parce que la précision des mesures est si faible et l'effort d'échantillonnage était si limité qu'il était très peu probable de détecter même d'énormes différences. La deuxième analyse de puissance permet d'éliminer cette seconde possibilité puisque Jaynie avait 71% des chances de détecter une différence du simple au double.

2.4.2. Analyse à priori

Supposons qu'on puisse défendre la position qu'une différence de biomasse observée par Jaynie entre les deux régions, $3.31 - 2.64 = 0.67 g/m^2$, soit écologiquement signifiante. On devrait donc planifier la prochaine saison d'échantillonnage de manière à avoir de bonnes chances de détecter une différence de cette taille. Combien de ruisseaux Jaynie devrait-elle échantillonner pour avoir 80% des chances de la détecter (compte tenu de la variabilité observée)?

Exercice

Changer le type d'analyse de puissance dans G*Power à **A priori**: **Compute sample size - given α, power, and effect size.** Assurez-vous que les valeurs pour les moyennes et les écarts-type soient celles qu'a obtenu Jaynie, recalculez la taille de l'effet, et inscrivez 0.8 pour la puissance.

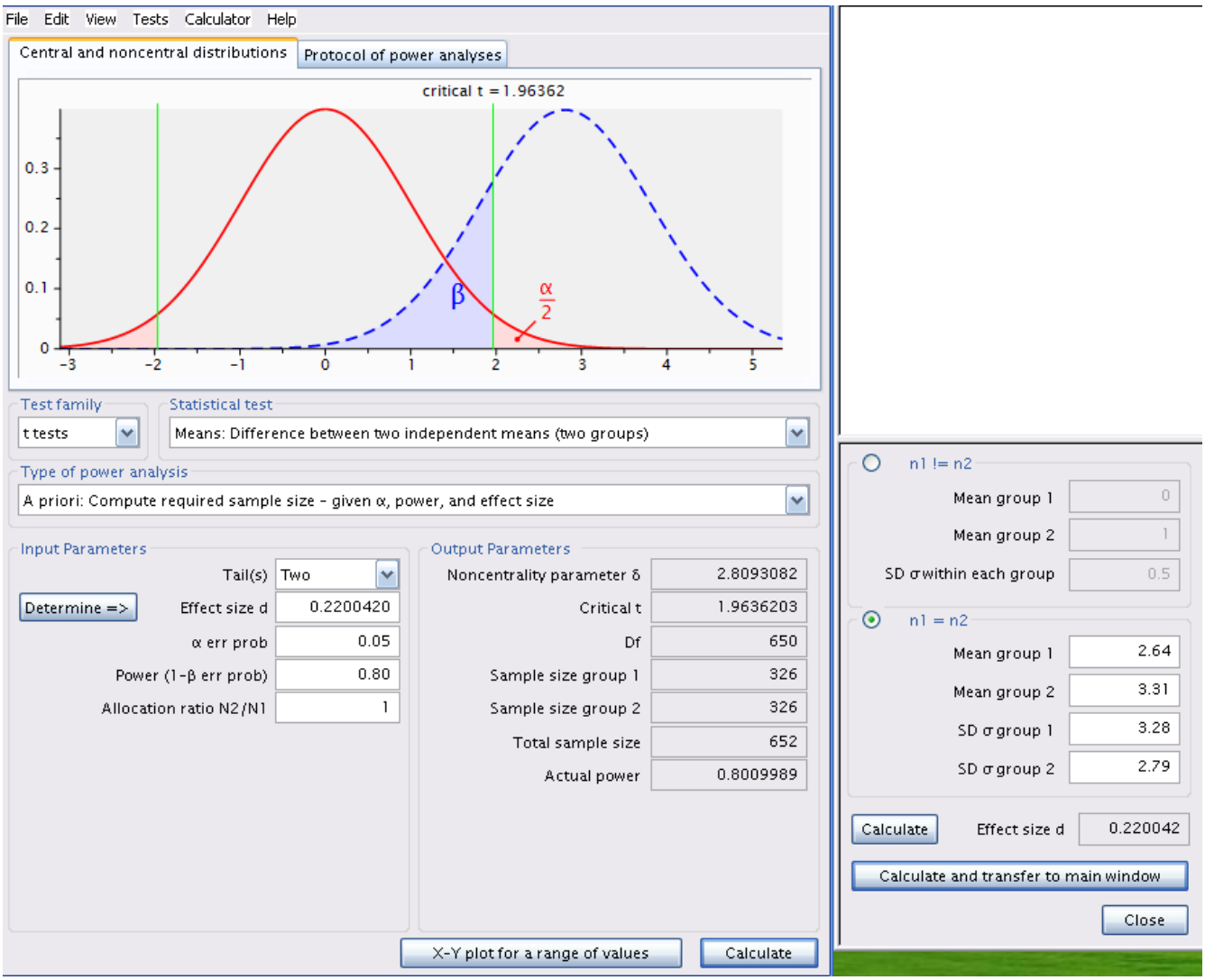


Figure 2.4.: Analyse à priori

```
pwr.t.test(power = 0.8, d = d(u1 = 2.64, sd1 = 3.28, u2 = 3.31, sd2 = 2.79), sig.level = 0.05, ty
```

```
     Two-sample t test power calculation

              n = 325.1723
              d = 0.220042
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

NOTE: n is number in *each* group
```

Ouch! Il faudrait échantillonner 326 ruisseaux dans chaque région! Cela coûterait une fortune et exigerait de nombreuses équipes de travail. Sans cela, on ne pourrait échantillonner que quelques dizaines de

ruisseaux, et il serait peu probable que l'on puisse détecter une si faible différence de biomasse entre les deux régions. Ce serait vraisemblablement en vain et pourrait être considéré comme une perte de temps: pourquoi tant d'efforts et de dépenses si les chances de succès sont si faibles.

Si on refait le même calcul pour une puissance de 95%, on obtient 538 ruisseaux par région. Augmenter la puissance ça demande plus d'effort.

```
pwr.t.test(power = 0.95, d = d(u1 = 2.64, sd1 = 3.28, u2 = 3.31, sd2 = 2.79), sig.level = 0.05, t
```

```
     Two-sample t test power calculation

              n = 537.7286
              d = 0.220042
      sig.level = 0.05
          power = 0.95
    alternative = two.sided

NOTE: n is number in *each* group
```

2.4.3. Analyse de sensitivité - Calculer la taille d'effet détectable

Compte tenu de la variabilité observée, d'un effort d'échantillonnage de 18 ruisseaux par région, et en conservant $\alpha = 0.05$, quelle est la taille d'effet que Jaynie pouvait détecter avec 80% de chances $\beta = 0.2$)?

> Exercice
>
> Changez le type d'analyse dans G*Power à **Sensitivity**: **Compute required effect size - given α, power, and sample size** et assurez-vous que la taille des échantillons est de 18 dans chaque région.

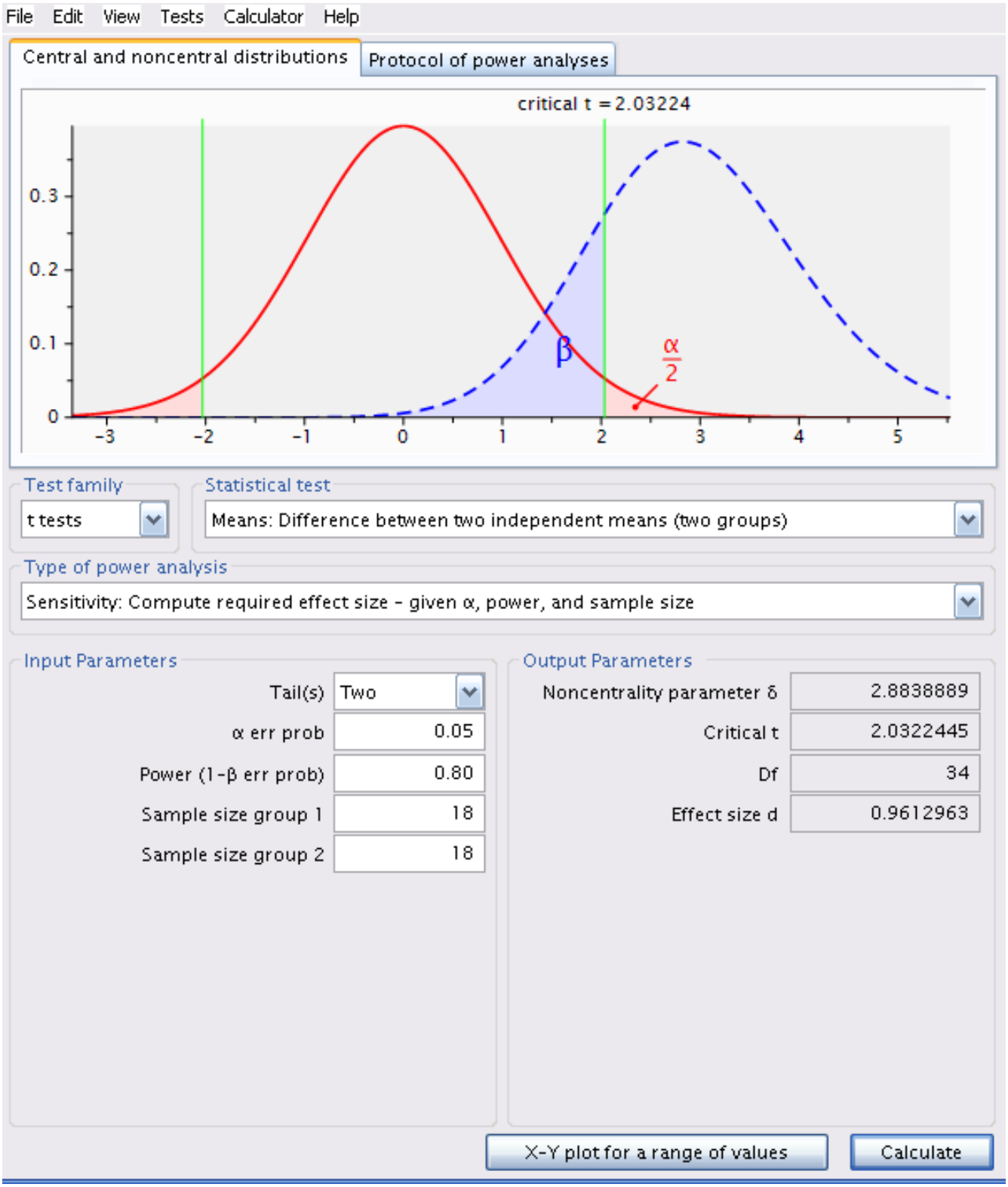


Figure 2.5.: Analyse de sensitivité

```
pwr.t.test(power = 0.8, n = 18, sig.level = 0.05, type = "two.sample")
```

```
     Two-sample t test power calculation

              n = 18
              d = 0.9612854
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

NOTE: n is number in *each* group
```

La taille d'effet détectable pour cette taille d'échantillon, $\alpha = 0.05$ et $\beta = 0.2$ (ou une puissance de 80%) est de 0.961296.

> **Warning**
>
> cette valeur est l'indice `d` de la taille de l'effet et est pondérée par la variabilité des mesures.

Dans ce cas ci, `d` est approximativement égal à

$$d = \frac{|\bar{X}_1 \bar{X}_2|}{\sqrt{\frac{{s_1}^2 + {s_2}^2}{2}}}$$

Pour convertir cette valeur de d sans unités en une valeur de différence de biomasse détectable (i.e $|\bar{X}_1 \bar{X}_2|$), il suffit de multiplier `d` par le dénominateur de l'équation.

$$|\bar{X}_1 \bar{X}_2| = d * \sqrt{\frac{{s_1}^2 + {s_2}^2}{2}}$$

Dans R, on peut estimer cela avec:

```
pwr.t.test(power = 0.8, n = 18, sig.level = 0.05, type = "two.sample")$d * sqrt((3.28^2 + 2.79^2)
```

```
[1] 2.926992
```

Donc, avec 18 ruisseaux dans chaque région, pour $\alpha = 0.05$ et $\beta = 0.2$ (une puissance de 80%), Jaynie pouvait détecter une différence de biomasse de $2.93 g/m^2$ entre les régions, un peu plus que du simple au double.

2.5. Points à retenir

- L'analyse de puissance post hoc n'est pertinente que lorsque l'on a accepté l'hypothèse nulle. Il est en effet impossible de faire une erreur de type II quand on rejette H_0.
- Avec de très grands échantillons, on a une puissance quasi infinie et on peut détecter statistiquement de très petites différences qui ne sont pas nécessairement biologiquement significatives.
- En utilisant un critère de signification plus strict (α<0.05) on diminue notre puissance.
- En voulant maximiser la puissance, on augmente l'effort requis, à moins d'utiliser une valeur critique plus libérale ($\alpha > 0.05$)
- Le choix de β est quelque peu arbitraire. On considère que $\beta = 0.2$ (puissance de 80%) est relativement élevé.

3. Corrélation et régression linéaire simple

Après avoir complété cet exercice de laboratoire, vous devriez être en mesure de :

- Utiliser R pour produire un diagramme de dispersion pour illus- trer la relation entre deux variables avec trace lowess
- Utiliser R pour faire des transformations simples
- Utiliser R pour calculer le coefficient de corrélation de Pearson entre deux variables et en évaluer sa signification statistique
- Utiliser R pour calculer la corrélation de rang entre des paires de variables avec le r de Spearman et le tau de Kendall
- Utiliser R pour évaluer la signification de corrélations dans une matrice de corrélation en utilisant les probabilités ajustées par la méthode de Bonferroni.
- Utiliser R pour faire une régression linéaire simple.
- Utiliser R pour évaluer si un ensemble de données remplit les conditions d'application d'une analyse de régression simple.
- Quantifier la taille de l'effet d'une régression simple et effectuer une analyse de puissance avec G*Power.

3.1. Extensions R et données requises pour le labo

Ce laboratoire nécessite:

- les paquets R:
  - car
  - lmtest
  - boot
  - pwr
  - ggplot
- les fichiers de données
  - sturgeon.csv

Il ne faut pas oublier de charge les extensions avec `library()` et de les installer au besoin avec `install.packages()` Pour les données, il faut les lire et les assigner à un objet R.

```
library(car)
library(lmtest)
library(boot)
library(ggplot2)
library(pwr)
```

```
library(performance)

sturgeon <- read.csv("data/sturgeon.csv")
```

> **Note**
>
> Notez que la ligne de code pour lire les données considère que le fichier de données se trouve dans un dossier `data` au sein de votre répertoire de travail. Si ce n'est pas le cas veuillez ajuster la ligne de commande.

3.2. Diagrammes de dispersion

Les analyses de corrélation et de régression devraient toujours commencer par un examen des données.C'est une étape critique qui sert à évaluer si ce type d'analyse est approprié pour un ensemble de données. Supposons que nous sommes intéressés à évaluer si la longueur d'esturgeons mâles dans la région de *The Pas* covarie avec leur poids. Pour répondre à cette question, regardons d'abord la corrélation entre la longueur et le poids. Souvenez-vous qu'une des conditions d'application de l'analyse de corrélation est que la relation entre les deux variables est linéaire. Pour évaluer cela, commençons par faire un diagramme de dispersion.

- Les données sur les esturgeons son disponibles dans le fichier `sturgeon.csv`. Après avoir chargé les données dnas un objet `sturgeon`, faites un diagramme de dispersion avec une droite de régression et une courbe LOWESS de la longueur en fonction du poids.

```
sturgeon <- read.csv("data/sturgeon.csv")
str(sturgeon)
```

```
'data.frame':   186 obs. of  9 variables:
 $ fklngth : num  37 50.2 28.9 50.2 45.6 ...
 $ totlngth: num  40.7 54.1 31.3 53.1 49.5 ...
 $ drlngth : num  23.6 31.5 17.3 32.3 32.1 ...
 $ rdwght  : num  15.95 NA 6.49 NA 29.92 ...
 $ age     : int  11 24 7 23 20 23 20 7 23 19 ...
 $ girth   : num  40.5 53.5 31 52.5 50 54.2 48 28.5 44 39 ...
 $ sex     : chr  "MALE" "FEMALE" "MALE" "FEMALE" ...
 $ location: chr  "THE_PAS" "THE_PAS" "THE_PAS" "THE_PAS" ...
 $ year    : int  1978 1978 1978 1978 1978 1978 1978 1978 1978 1978 ...
```

```
mygraph <- ggplot(
  data = sturgeon[!is.na(sturgeon$rdwght), ], # source of data
  aes(x = fklngth, y = rdwght)
)
```

```
# plot data points, regression, loess trace
mygraph <- mygraph +
  stat_smooth(method = lm, se = FALSE, color = "green") + # add linear regression, but no SE shad
  stat_smooth(color = "red", se = FALSE) + # add loess
  geom_point() # add data points

mygraph # display graph
```

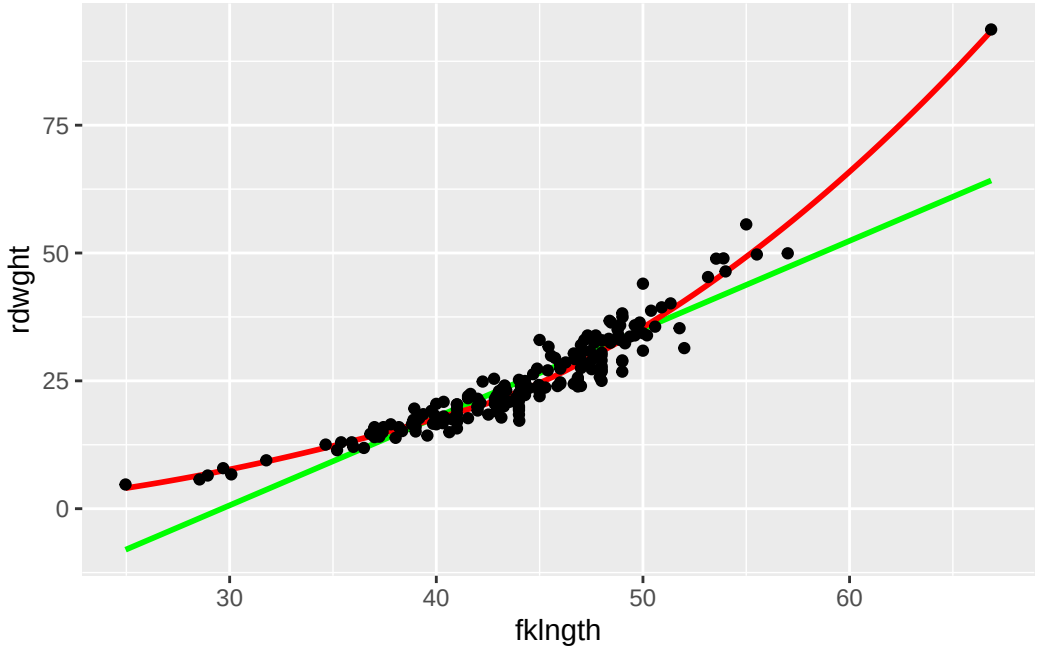


Figure 3.1.: Graphique du poids en fonction de la longueur des esturgeons

- Est-ce que la dispersion des points suggère une bonne corrélation entre les deux variables? Est-ce que la relation semble linéaire?

Ce graphique suggère une tendance plus curvilinéaire que linéaire. Malgré tout, il semble y avoir une forte corrélation entre les deux variables.

- Refaites le diagramme de dispersion avec des axes logarithmiques.

```
# apply log transformation on defined graph
mygraph + scale_x_log10() + scale_y_log10()
```

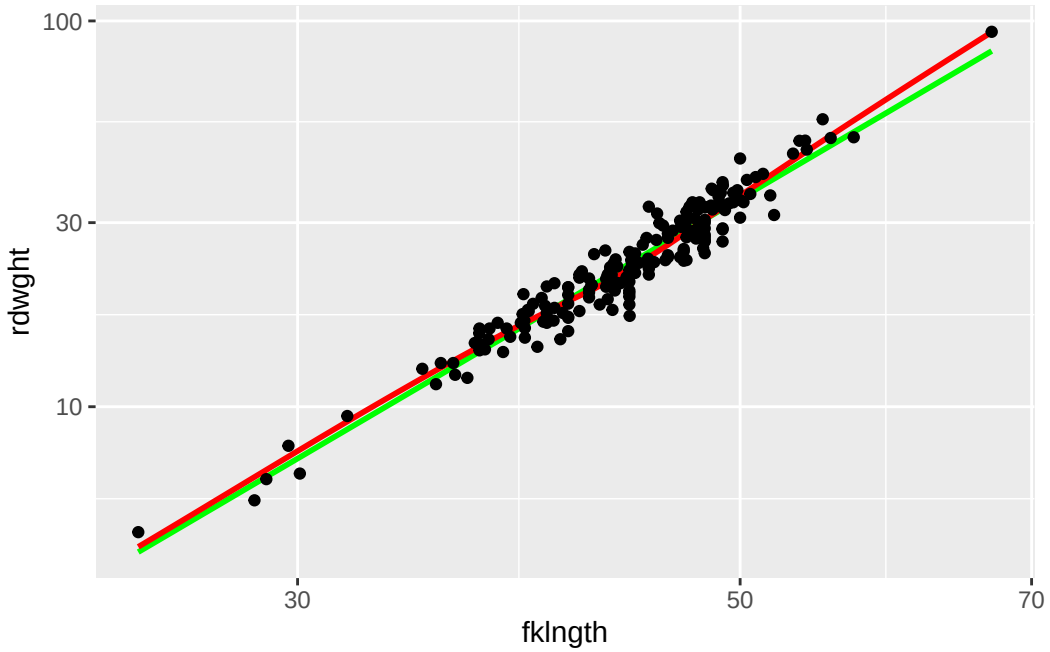


Figure 3.2.: Graphique poids-longueur avec une échelle log

Comparez les diagrammes de dispersion avant et après transformation (Figures @ref(fig:stur-2) et @ref(fig:stur-log)). Comme l'analyse de corrélation présuppose une relation linéaire entre les variables, on devrait donc privilégier l'analyse sur les données log-transformées.

3.3. Transformations et le coefficient de corrélation

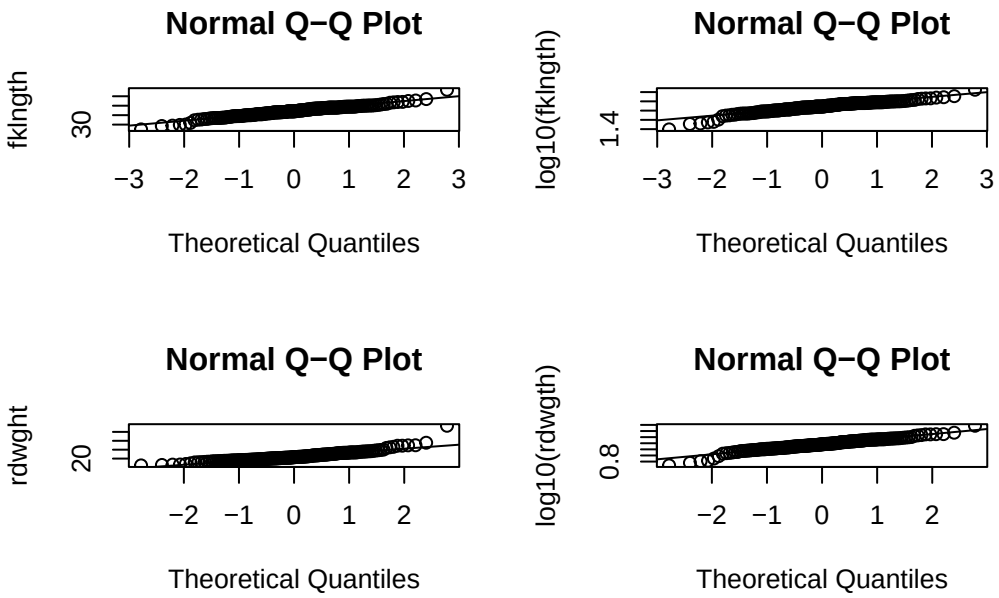
Une autre condition préalable à l'analyse de corrélation est que les deux variables concernées suivent une distribution normale bidimensionnelle. On peut aisément vérifier la normalité de chacune des 2 variables séparément tel que décrit dans le laboratoire précédent. Si les deux variables sont normalement distribuées, on présume généralement qu'elles suivent une distribution normale bidimensionnelle lorsqu'analysées simultanément (notez que ce n'est pas toujours le cas cependant).

- Examinez la distribution des quatre variables (les deux variables originales et les variables transformées). Que concluez-vous de l'inspection visuelle de ces graphiques ?

Les figures ci-dessous sont les diagrammes de probabilité (`qqplot()`). Le code pour produire des graphiques multiples sur une page, comme on voit ci-dessous, est:

```
par(mfrow = c(2, 2)) # divise le graphique en 4 sections
qqnorm(sturgeon$fklngth, ylab = "fklngth")
qqline(sturgeon$fklngth)
qqnorm(log10(sturgeon$fklngth), ylab = "log10(fklngth)")
qqline(log10(sturgeon$fklngth))
qqnorm(sturgeon$rdwght, ylab = "rdwght")
qqline(sturgeon$rdwght)
```

```
qqnorm(log10(sturgeon$rdwght), ylab = "log10(rdwgth)")
qqline(log10(sturgeon$rdwght))
```



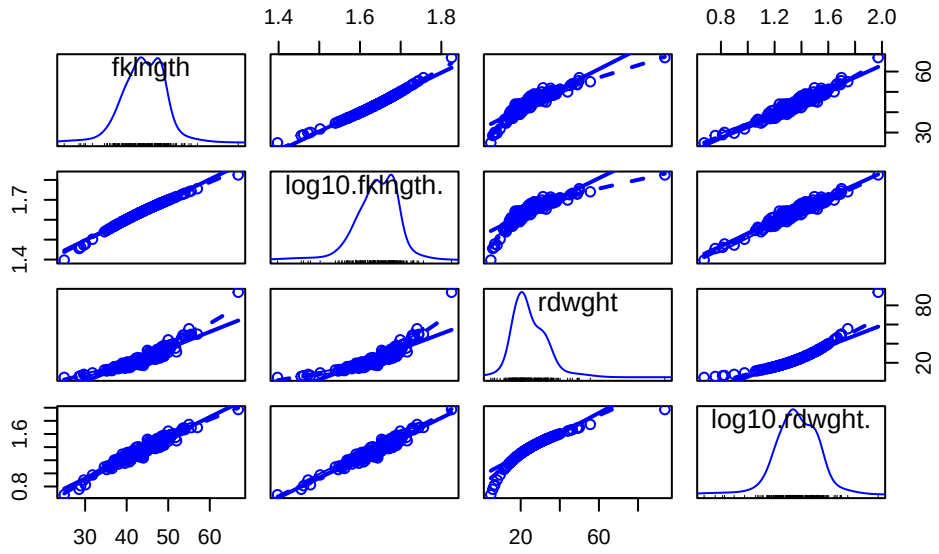

```
par(mfrow = c(1, 1)) # redéfinie la zone de graphique par défaut
```

Il n'y a pas grand-chose à redire: aucune des distributions n'est parfaitement normale, mais les déviations semblent mineures.

- Générez une matrice de graphiques de dispersion améliorés en utilisant la commande `scatterplotMatrix` de la librairie `car`.

```
scatterplotMatrix(
  ~ fklngth + log10(fklngth) + rdwght + log10(rdwght),
  data = sturgeon,
  smooth = TRUE, diagonal = "density"
)
```

```
Warning in applyDefaults(diagonal, defaults = list(method = "adaptiveDensity"),
: unnamed diag arguments, will be ignored
```



- Ensuite, calculez le coefficient de corrélation de Pearson entre chaque paire (variables originales et logtransformées) en utilisant la commande `cor()`. Avant de commencer, on va cependant ajouter les variables transformées au tableau de données sturgeon:

```
sturgeon$lfklngth <- with(sturgeon, log10(fklngth))
sturgeon$lrdwght <- log10(sturgeon$rdwght)
```

Vous pouvez ensuite obtenir la matrice de corrélation par:

```
cor(sturgeon[, c("fklngth", "lfklngth", "lrdwght", "rdwght")], use = "complete.obs")
```

Fréquemment, il y a des données manquantes dans un échantillon. En choisissant `use="complete.obs"`, toutes les lignes du fichier pour lesquelles les variables ne sont pas toutes mesurées sont éliminées. Dans ce cas, toutes les corrélations seront calculées avec le même nombre de cas. Par contre, en utilisant `use="pairwise.complete.obs"`, R élimine une observation que lorsqu'un des deux membres de la paire a une valeur manquante. Dans ce cas, si les données manquantes pour différentes variables se retrouvent dans un groupe différent d'observation, les corrélations ne seront pas nécessairement calculées sur le même nombre de cas ni sur le même sous-ensemble de cas. En général, vous devriez utiliser l'option `use="complete.obs"` à moins que vous ayez un très grand nombre de données manquantes et que cette façon de procéder élimine la plus grande partie de vos observations.

Pourquoi la corrélation entre les variables originales est-elle la plus faible des trois ?

```
cor(sturgeon[, c("fklngth", "lfklngth", "lrdwght", "rdwght")], use = "complete.obs")
```

```
           fklngth  lfklngth   lrdwght    rdwght
fklngth  1.0000000 0.9921435 0.9645108 0.9175435
lfklngth 0.9921435 1.0000000 0.9670139 0.8756203
lrdwght  0.9645108 0.9670139 1.0000000 0.9265513
rdwght   0.9175435 0.8756203 0.9265513 1.0000000
```

Il y a plusieurs choses à noter ici.

- Premièrement, la corrélation entre la longueur à la fourche et le poids rond est élevée, peu importe la transformation: les poissons lourds ont tendance à être longs.
- Deuxièmement, la corrélation est plus forte pour les données transformées que pour les données brutes.

Pourquoi? Parce que le coefficient de corrélation est inversement proportionnel au bruit autour de la relation linéaire. Si la relation est curvilinéaire (comme dans le cas des données non transformées), le bruit est plus grand que si la relation est parfaitement linéaire. Par conséquent, la corrélation est plus faible.

3.4. Matrices de corrélations et correction de Bonferroni

Une pratique courante est d'examiner une matrice de corrélation à la recherche des associations significatives. Comme exemple, essayons de tester si la corrélation entre lfklngth et rdwght est significative (c'est le plus faible coefficient de corrélation de cette matrice).

- Estimer la correlation entre la longueur (fklngth) et le poids (rdwght) des esturgeons:

```
cor.test(
  sturgeon$lfklngth, sturgeon$rdwght,
  alternative = "two.sided",
  method = "pearson"
)
```

```
    Pearson's product-moment correlation

data:  sturgeon$lfklngth and sturgeon$rdwght
t = 24.322, df = 180, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.8367345 0.9057199
sample estimates:
      cor
0.8756203
```

On voit ici que la corrélation est hautement significative ($p < 2.2e-16$),ce qui n'est pas surprenant étant donné la valeur du coefficient de corrélation (0.8756). Il est important de réaliser que si une matrice contient un grand nombre de corrélations, il n'est pas surprenant d'en trouver au moins une qui soit

"significative". En effet, on s'attend à en trouver 5% en moyenne lorsqu'il n'y a en fait aucune corrélation entre les paires de moyennes. Une façon de corriger pour cette tendance est d'ajuster le niveau α critique auquel on attribue une signification statistique en divisant α par le nombre k de corrélations qui sont examinées : $\alpha' = \alpha/k$ (ajustement de Bonferroni). Si initialement $\alpha = 0.05$ et qu'il y a 10 corrélations qui sont examinées, alors $\alpha' = 0.005$. Donc, afin de rejeter l'hypothèse nulle, la valeur de p devra être plus petite que α', en l'occurrence 0.005. Dans l'exemple qui précède, on devrait donc ajuster α critique en divisant par le nombre total de corrélations dans la matrice (6 dans ce cas, donc $\alpha' = 0.00833$). Cette correction modifie-t-elle votre conclusion quant à la corrélation entre `lkfl` et `rdwght`?

3.5. Corrélations non paramétriques: *r* de Spearman et τ de Kendall

L'analyse faite à la section précédente avec les esturgeons suggère que l'une des conditions préalables à l'analyse de corrélation, soit la distribution normale bidimensionnelle de données, pourrait ne pas être remplie pour `fklngth` et `rdwght`, ni pour les paires de variables transformées. La recherche d'une transformation appropriée peut parfois être difficile. Pire encore, pour certaines distributions il n'existe pas de transformation qui va normaliser les données. Dans ces cas-là, la meilleure option est de faire une analyse non paramétrique qui ne présume ni de la normalité ni de la linéarité. Ces analyses sont basées sur les rangs. Les deux plus communes sont le coefficient de rang de Spearman et le τ (tau) de Kendall.

- Dans R, testez la corrélation entre `fklngth` et `rdwght` en utilisant Spearman et Kendall's .

```
cor.test(
  sturgeon$lfklngth, sturgeon$rdwght,
  alternative = "two.sided",
  method = "spearman"
)
```

```
Warning in cor.test.default(sturgeon$lfklngth, sturgeon$rdwght, alternative =
"two.sided", : Cannot compute exact p-value with ties
```

```

    Spearman's rank correlation rho

data:  sturgeon$lfklngth and sturgeon$rdwght
S = 47971, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.9522546
```

```
cor.test(
  sturgeon$lfklngth, sturgeon$rdwght,
  alternative = "two.sided",
  method = "kendall"
)
```

```
    Kendall's rank correlation tau

data:  sturgeon$lfklngth and sturgeon$rdwght
z = 16.358, p-value < 2.2e-16
alternative hypothesis: true tau is not equal to 0
sample estimates:
      tau
0.8208065
```

Comparer les résultats de cette analyse à l'analyse paramétrique. Pourquoi y-a-t'il une différence ?

Calculez les corrélations non paramétriques sur les paires de variables transformées. Vous devriez voir tout de suite que les corrélations des données transformées et non transformées sont identiques puisque dans les deux cas la corrélation est calculée à partir des rangs qui ne sont pas affectés par la transformation.

Notez que les corrélations obtenues avec le tau de Kendall (0.820) sont plus faibles que celles du coefficient de Spearman (0.952). Le tau de Kendall pondère un peu plus les grandes différences entre les rangs alors que le coefficient de Spearman donne le même poids à chaque paire d'observations. En général, on préfère le tau de Kendall lorsqu'il y a plus d'incertitude quant aux rangs qui sont près les uns des autres.

Les esturgeons de cet échantillon ont été capturés à l'aide de filets et d'hameçons d'une taille fixe. Quel impact cette méthode de capture peut-elle avoir eu sur la forme de la distribution de `fklngth` et `rdwght`? Compte tenu de ces circonstances, l'analyse de corrélation est-elle appropriée ?

Rappelez-vous que l'analyse de corrélation présume aussi que chaque variable est échantillonnée aléatoirement. Dans le cas de nos esturgeons, ce n'est pas le cas: les hameçons appâtés et les filets ne capturent pas de petits esturgeons (et c'est pourquoi il n'y en a pas dans l'échantillon). Il faut donc réaliser que les coefficients de corrélation obtenus dans cette analyse ne reflètent pas nécessairement ceux de la population totale des esturgeons.

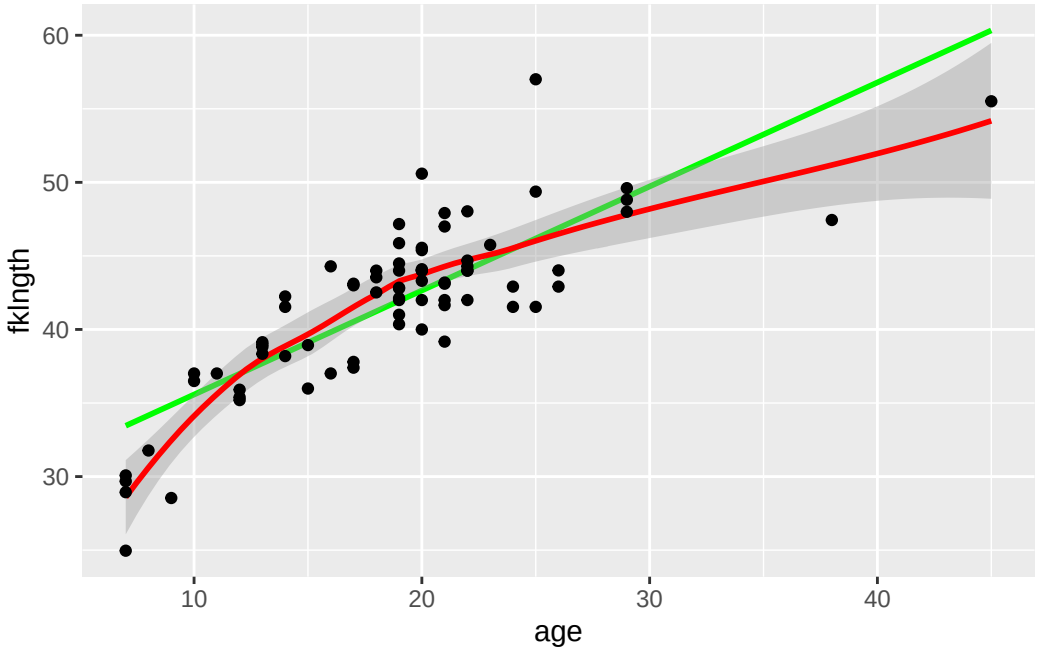
3.6. Régression linéaire simple

L'analyse de corrélation vise à décrire comment deux variables covarient. L'analyse de régression vise plutôt à produire un modèle permettant de prédire une variable (la variable dépendante) par l'autre (la variable indépendante).

Comme pour l'analyse de corrélation, on devrait commencer en examinant des graphiques. Puisque l'on est intéressé à quantifier la relation entre deux variables, un graphique de la variable dépendante (Y) en fonction de la variable indépendante (X) est tout à fait approprié.

- Le fichier `sturgeon.csv` contient des données d'un inventaire des esturgeons récoltés en 1978-1980 à Cumberland House en Saskatchewan et à The Pas au Manitoba. Faites un diagramme de dispersion de `fklngth` (la variable dépendante) en fonction de `age` (la variable indépendante) pour les esturgeons mâles unqiuement et ajoutez-y une régression linéaire et une trace lowess. Que concluez-vous de ce diagramme de dispersion ?

```
sturgeon.male <- subset(sturgeon, subset = sex == "MALE")
mygraph <- ggplot(
  data = sturgeon.male, # source of data
  aes(x = age, y = fklngth)
) # aesthetics: y=fklngth, x=rdwght
# plot data points, regression, loess trace
mygraph <- mygraph +
  stat_smooth(method = lm, se = FALSE, color = "green") + # add linear regression, but no SE shad
  stat_smooth(color = "red") + # add loess
  geom_point() # add data points
mygraph # display graph
```



Ce graphique suggère que la relation n'est pas linéaire.

Supposons que nous désirions estimer le taux de croissance des esturgeons mâles. Un estimé (peut-être pas terrible...) du taux de croissance peut être obtenu en calculant la pente de la régression de la longueur à la fourche sur l'âge.

Ajustons d'abord une régression avec la commande lm() et sauvons ces résultats dans un objet appelé `RegModel.1`.

```
RegModel.1 <- lm(fklngth ~ age, data = sturgeon.male)
```

Rien n'apparait à l'écran, c'est normal ne vous inquiétez pas, tout a été sauvegardé en mémoire. Pour voir les résultats, tapez:

```r
summary(RegModel.1)
```

```
Call:
lm(formula = fklngth ~ age, data = sturgeon.male)

Residuals:
    Min      1Q  Median      3Q     Max
-8.4936 -2.2263  0.1849  1.7526 10.8234

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 28.50359    1.16873   24.39   <2e-16 ***
age          0.70724    0.05888   12.01   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.307 on 73 degrees of freedom
  (5 observations deleted due to missingness)
Multiple R-squared:  0.664, Adjusted R-squared:  0.6594
F-statistic: 144.3 on 1 and 73 DF,  p-value: < 2.2e-16
```

la sortie R donne:

1. `Call:` Le modèle qui a été ajusté et les données utilisées.
2. `Residuals:` Un sommaire statistique des résidus autour du modèle estimé.
3. `Coefficients:` Valeurs estimées des paramètres du modèle, erreurs-types, statistiques t et probabilités associées.
4. `Residual standard error:` Racine carrée de la variance résiduelle.
5. `Multiple R-squared:` Coefficient de détermination. Il correspond à la proportion de la variabilité de la variable dépendante qui peut être expliquée par la régression.
6. `Adjusted R-squared:` Le R-carré ajusté tient compte du nombre de paramètres du modèle. Si vous voulez comparer différents modèles qui n'ont pas le même nombre de paramètres, c'est ce qu'il faut utiliser.
7. `F-statistic:` C'est le test de signification omnibus du modèle. Dans le cas de la régression simple, il est équivalent au test sur la pente de la régression.

La régression estimée est donc:

$$Fklngth = 28.50359 + 0.70724 * age$$

Étant donné la valeur significative du test de F ainsi que pour le test de t pour la pente de la droite, on rejette l'hypothèse nulle qu'il n'y a pas de relation entre la taille et l'âge.

3.6.1. Vérifier les conditions d'application de la régression

La régression simple de type I a quatre conditions préalables :

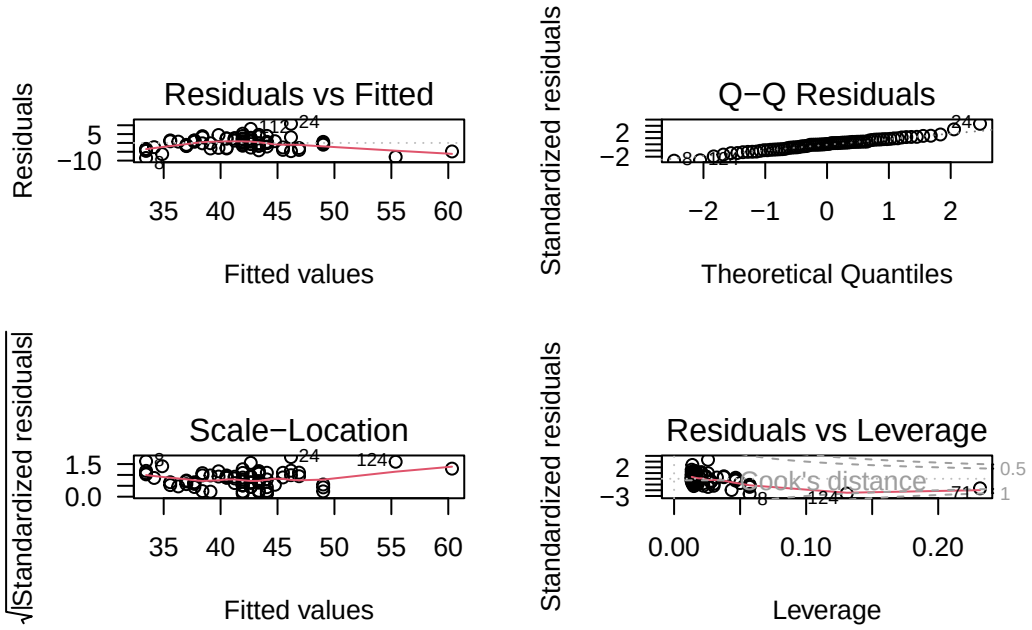
1. il n'y a pas d'erreur de mesure sur la variable indépendante (X)
2. la relation entre Y et X est linéaire
3. les résidus sont normalement distribués
4. la variance des résidus est constante pour toutes les valeurs de la variable indépendante

Procédons maintenant à l'examen post-mortem. La première condition est rarement remplie avec des données biologiques ; il y presque toujours de l'erreur sur X et sur Y. Cela veut dire qu'en général les pentes estimées sont biaisées, mais que les valeurs prédites ne le sont pas. Toutefois, si l'erreur de mesure sur X est petite par rapport à l'étendue des valeurs de X, le résultat de l'analyse n'est pas dramatiquement influencé. Par contre, si l'erreur de mesure est relativement grande (toujours par rapport à l'étendue des valeurs de X), la droite de régression obtenue par la régression de modèle I est un piètre estimé de la relation fonctionnelle entre X et Y. Dans ce cas, il est préférable de passer à la régression de modèle II, malheureusement au-delà du contenu de ce cours. Les autres conditions préalables à l'analyse de régression de modèle I peuvent cependant être vérifiées, ou du moins évaluées visuellement. La commande plot() permet de visualiser des graphiques diagnostiques pour des modèles linéaires.

```
par(mfrow = c(2, 2), las = 1)
plot(RegModel.1)
```

La commande par() est utilisée pour dire à R de tracer 2 rangées et 2 colonnes de graphiques par page (il y a quatre graphiques diagnostiques qui sont générés automatiquement pour les modèles linéaires), et la commande las indique à R d'effectuer une rotation des légendes de l'axe des Y pour qu'elles soient perpendiculaires à l'axe (oui. Je sais. Rien de tout ça n'est évident.)

Vous obtiendrez:

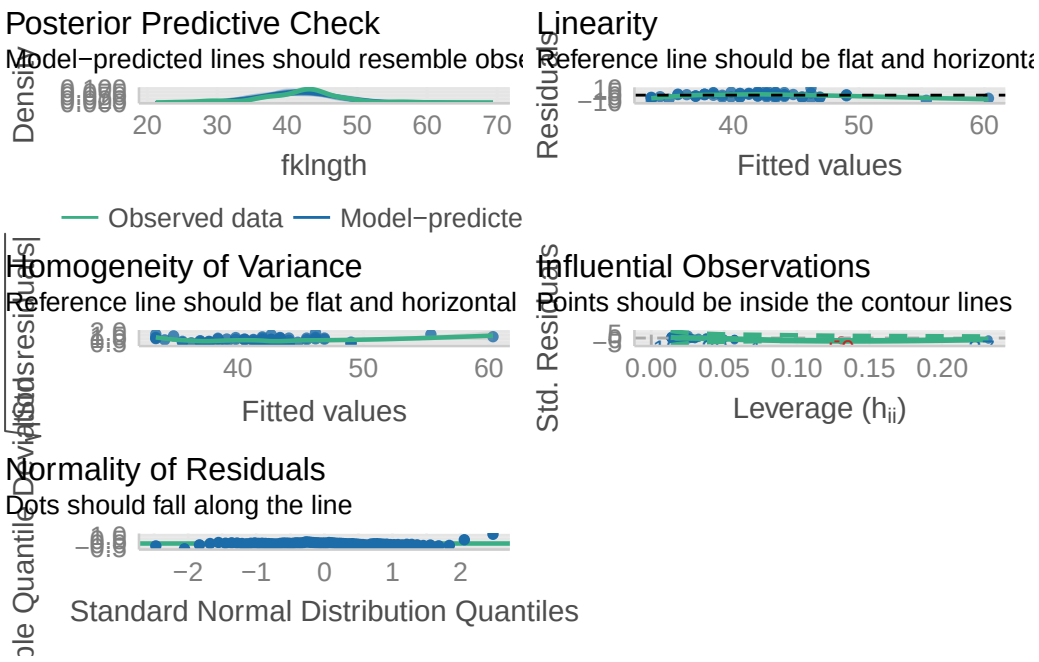


1. En haut à gauche, permet d'évaluer la linéarité, la normalité, et l'homoscédasticité des résidus. Il illustre les déviations autour de la régression en fonction des valeurs prédites. Rappllez-vous que le graphique de fklngth vs age suggère que la relation entre la longueur à la fourche et l'âge n'est pas linéaire. Les très jeunes et très vieux esturgeons sont sous la droite en général, alors que les esturgeons d'âge moyen sont retrouvés généralement au-dessus de la droite de régression. C'est exactement ce que le graphique des résidus en fonction des valeurs prédites illustre. La ligne en rouge est une trace lowess au travers de ce nuage de points. Si la relation était linéaire, la trace lowess serait presque plate et près de 0. La dispersion des résidus permet d'évaluer visuellement leur normalité et hétéroscédasticité; mais ce graphique n'est pas optimal pour évaluer ces propriétés. Les deux graphiques suivants sont supérieurs au premier pour cela.
2. En haut à droite permet d'évaluer la normalité des résidus. C'est un graphique QQ des résidus (QQ plot). Des résidus distribués normalement tomberaient exactement sur la diagonale. Ici, on voit que c'est presque le cas, sauf dans les queues de la distribution.
3. En bas à gauche, intitulé Scale-Location, permet d'évaluer l'homoscedasticité. On y retrouve sur l'ordonnée (l'axe des y) la racine carrée de la valeur absolue des résidus standardisés (résidus divisés par l'écart-type des résidus) en fonction des valeurs prédites. Le graphique aide à déterminer si la variation des résidus est constante ou non. Si les résidus sont homoscédastiques, la valeur moyenne sur l'axe des y ne va pas changer en fonction de la valeur prédite. Ici, il y a une certaine tendance, mais pas une tendance monotone puisqu' il y a d'abord une baisse puis une hausse..; bref, rien qui soit une forte évidence contre la supposition d'homoscédasticité.
4. En bas à droite, montre les résidus en fonction du "leverage" et permet de détecter certaines valeurs extrêmes qui ont une grande influence sur la régression. Le leverage d'un point mesure sa distance des autres points, mais seulement en ce qui concerne les variables indépendantes. Dans le cas d'une régression simple, cela revient à la distance entre le point sur l'axe des x et la moyenne de tous les points sur cet axe. Vous devriez porter une attention particulière aux observations qui ont un leverage plus grand que $2(k+1)/n$, où k est le nombre de variables indépendantes (ici, 1) et n est le nombre d'observations. Dans cet exemple, il y a 75 observations et une variable indépendante et donc les points ayant un leverage plus grand que $4/75 = 0.053$ devrait être considérés avec attention. Le graphique indique également comment la régression changerait si on enlevait un point. Ce changement est mesuré par la distance de Cook, illustrée par les bandes en rouge sur le graphique. Un point ayant une distance de Cook supérieure à 1 a une grande influence.

> ⚠ Warning
>
> Notez que R identifie automatiquement les cas les plus extrèmes sur chacun de ces 4 graphiques. Le fait qu'un point soit identifié ne signifie pas nécessairement que c'est une valeur réellement extrème, ou que vous devez vous en préoccuper. Dans tous les ensembles de données il y aura toujours un résidu plus grand que les autres…

Il est possible d'obtenir des graphiques d'évalutions des conditions d'applications, qui sont plus simple à interpréter et plus joli (avec des couleurs). Il faut utiliser la fonction `model_check()` dans le `performance`.

```
check_model(RegModel.1)
```



Finalement, quel est le verdict concernant la régression linéaire entre fklngth et age ? Elle viole la condition de linéarité, possiblement celle de normalité, remplit la condition d'homoscédasticité, et ne semble pas influencée outre mesure par des valeurs bizarres ou extrêmes.

3.6.2. Tests formels des conditions d'application pour la régression

Personnellement, je n'utilise jamais les tests formels des conditions d'application de la régression et me contente des graphiques des résidus pour guider mes décisions. C'est ce que la plupart des praticiens font. Cependant, lors de mes premières analyses, je n'étais pas toujours certain de bien interpréter les graphiques et j'aurais aimé un indice plus formel ou un test permettant de détecter les violations des conditions d'application de la régression.

Le package `lmtest`, qui ne fait pas partie de l'installation de base, mais qui est disponible sur CRAN, permet de faire plusieurs tests de linéarité et d'homoscédasticité. Et on peut tester la normalité avec le test Shapiro-Wilk test vu précédemment.

Charger le package lmtest de CRAN (et installer le si besoin).

Solution

```
library(lmtest)
```

Exercice

Exécutez les commandes suivantes

```
bptest(RegModel.1)
```

```
    studentized Breusch-Pagan test

data:  RegModel.1
BP = 1.1765, df = 1, p-value = 0.2781
```

Le test Breusch-Pagan test examine si la variabilité des résidus est constantes lorsque les valeurs prédites changent. Une faible valeur de p suggère de l'hétéroscédasticité. Ici, la valeur p est élevée et suggère que la condition d'application d'homoscédasticité est remplie avec ces données.

```
dwtest(RegModel.1)
```

```
    Durbin-Watson test

data:  RegModel.1
DW = 2.242, p-value = 0.8489
alternative hypothesis: true autocorrelation is greater than 0
```

Le test Durbin-Watson permet de détecter l'autocorrélation sérielle des résidus. En l'absence d'autocorrélation (i.e. d'indépendance des résidus) la valeur attendue de la statistique D est 2. Ce test permet d'éprouver l'hypothèse d'indépendance des résidus, mais ne permet de détecter qu'un type particulier de dépendance. Ici, le test ne permet pas de rejeter l'hypothèse d'indépendance.

```
resettest(RegModel.1)
```

```
    RESET test

data:  RegModel.1
RESET = 14.544, df1 = 2, df2 = 71, p-value = 5.082e-06
```

Le test RESET permet d'éprouver la linéarité. Si la relation est linéaire, alors la statistique RESET sera d'environ 1. Ici, la statistique est beaucoup plus élevée (14.54) et hautement significative. Le test confirme la tendance que nous avons détectée visuellement plus haut: la relation n'est pas linéaire.

```
shapiro.test(residuals(RegModel.1))
```

```
    Shapiro-Wilk normality test

data:  residuals(RegModel.1)
W = 0.98037, p-value = 0.2961
```

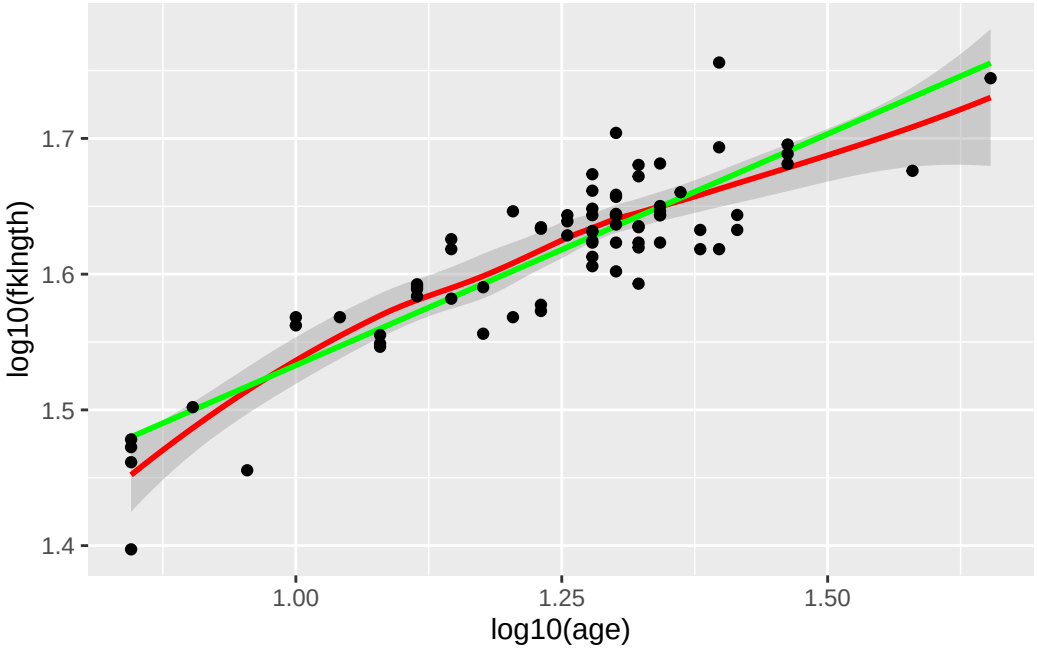
Le test de normalité Shapiro-Wilk sur les résidus confirme que la déviation par rapport à une distribution normale des résidus n'est pas grande.

3.7. Transformation des données en régression

La relation entre `fklngth` et `age` n'étant pas linéaire, on devrait donc essayer de transformer les données pour tenter de les linéariser :

- Voyons ce qu'une transformation log donne:

```
par(mfrow = c(1, 1), las = 1)
ggplot(
  data = sturgeon.male,
  aes(x = log10(age), y = log10(fklngth))
) +
  geom_smooth(color = "red") +
  geom_smooth(method = "lm", se = FALSE, color = "green") +
  geom_point()
```



Ajustons maintenant une régression simple sur ces données transformées.

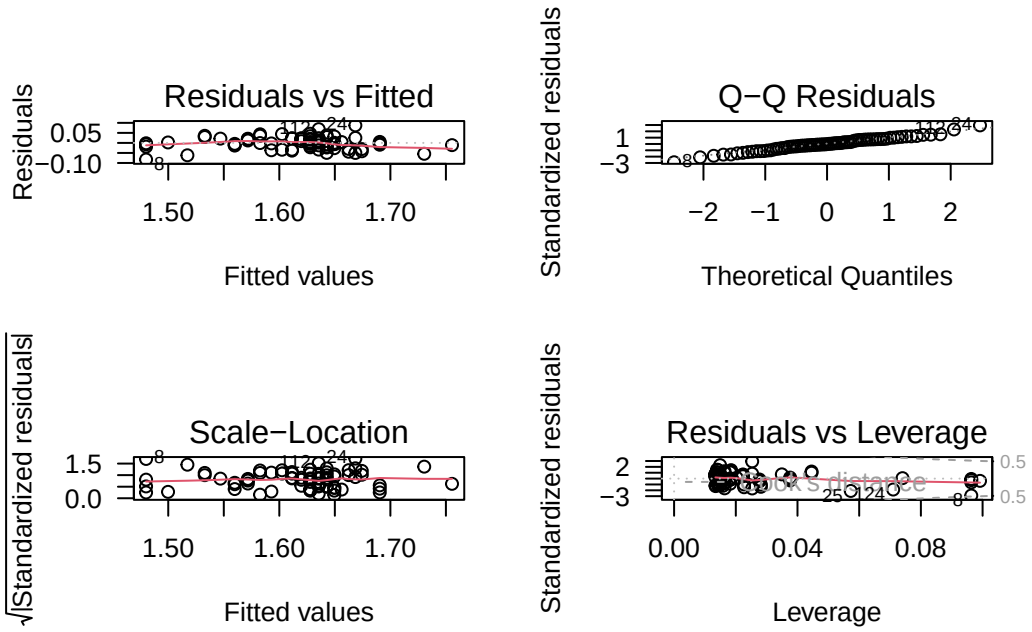
```
RegModel.2 <- lm(log10(fklngth) ~ log10(age), data = sturgeon.male)
summary(RegModel.2)
```

```
Call:
lm(formula = log10(fklngth) ~ log10(age), data = sturgeon.male)

Residuals:
      Min        1Q    Median        3Q       Max
```

```
-0.082794 -0.016837 -0.000719  0.021102  0.087446

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.19199    0.02723   43.77   <2e-16 ***
log10(age)   0.34086    0.02168   15.72   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03015 on 73 degrees of freedom
  (5 observations deleted due to missingness)
Multiple R-squared:  0.772, Adjusted R-squared:  0.7688
F-statistic: 247.1 on 1 and 73 DF,  p-value: < 2.2e-16
```

Examinons maintenant les graphiques diagnostiques:

```
par(mfrow = c(2, 2), las = 1)
plot(RegModel.2)
```



Il y a une certaine amélioration, mais ce n'est pas encore parfait (la perfection n'est pas de ce monde....). Le graphique des résidus en fonction des valeurs prédites suggère encore une certaine non linéarité. Sur le graphique Q-Q les points se retrouvent plus près de la droite diagonale qu'avant, indiquant que les résidus sont encore plus près de la normalité après la transformation log-log. Il n'y a pas d'indice d'hétéroscédasticité. Finalement, même si il reste quelques points avec plus d'influence (leverage) que les autres, aucun n'a une distance de Cook au-delà de 0.5. En résumé, la transformation log a amélioré les choses: relation est plus linéaire, les résidus sont plus normaux, et il y a moins de points avec une influence relativement élevée.Est-ce que les tests formels supportent cette évaluation?

```
bptest(RegModel.2)
```

```
    studentized Breusch-Pagan test

data:  RegModel.2
BP = 0.14282, df = 1, p-value = 0.7055
```

```
dwtest(RegModel.2)
```

```
    Durbin-Watson test

data:  RegModel.2
DW = 2.1777, p-value = 0.6134
alternative hypothesis: true autocorrelation is greater than 0
```

```
resettest(RegModel.2)
```

```
    RESET test

data:  RegModel.2
RESET = 4.4413, df1 = 2, df2 = 71, p-value = 0.01523
```

```
shapiro.test(residuals(RegModel.2))
```

```
    Shapiro-Wilk normality test

data:  residuals(RegModel.2)
W = 0.98998, p-value = 0.8246
```

Oui, les conclusions sont les mêmes: les résidus sont encore homoscédastiques (test Breusch-Pagan), ne sont pas autocorrélés (test Durbin-Watson), sont normaux (test Shapiro-Wilk), et sont plus linéaires (la valeur de P du test RESET est maintenant 0.015, au lieu de 0.000005). Donc la linéarité a augmenté, mais cette condition d'application semble encore légèrement violée.

3.8. Traitement des valeurs extrèmes

Dans cet exemple, il n'y a pas de valeur vraiment extrème. Oui, je sais, R a quand même identifié les observations 8, 24, et 112 dans le dernier graphique diagnostique. Mais ces valeurs sont encore dans la fourchette de valeurs que je juge "acceptables". Mais comment déterminer objectivement ce qui est acceptable? À quel moment juge t'on qu'une valeur extrème est vraiment trop invraisemblable pour ne pas l'exclure? Il n'y a malheureusement pas de règle absolue là-dessus. Les opinions varient, mais je penche vers le conservatisme sur cette question.

Ma position est que, à moins que la valeur soit biologiquement impossible ou clairement une erreur d'entrée de données, je n'élimine pas les valeurs extrêmes et j'utilise toutes mes données dans leur analyse. Pourquoi?

Parce que je veux que mes données reflètent bien la variabilité naturelle ou réelle. C'est d'ailleurs parfois cette variabilité qui est intéressante.

L'approche conservatrice qui consiste à conserver toutes les valeurs extrêmes possibles est possiblement la plus honnête, mais elle peut causer certains problèmes. Ces valeurs extrêmes sont souvent la cause des violations des conditions d'application des tests statistiques. La solution suggérée à ce dilemme est de faire l'analyse avec et sans les valeurs extrêmes et de comparer les conclusions. Dans bien des cas, les conclusions seront qualitativement les mêmes et les tailles d'effet ne seront pas très différentes. Toutefois, dans certains cas, la présence des valeurs extrêmes change complètement les conclusions. Dans ces cas, il faut simplement accepter que les conclusions dépendent entièrement de la présence des valeurs extrêmes et sont donc peu concluantes.

Suivant cette approche comparative, refaisons donc l'analyse après avoir enlevé les observations 8, 24, et 112.

```
RegModel.3 <- lm(log10(fklngth) ~ log10(age), data = sturgeon.male, subset = !(rownames(sturgeon.
summary(RegModel.3)
```

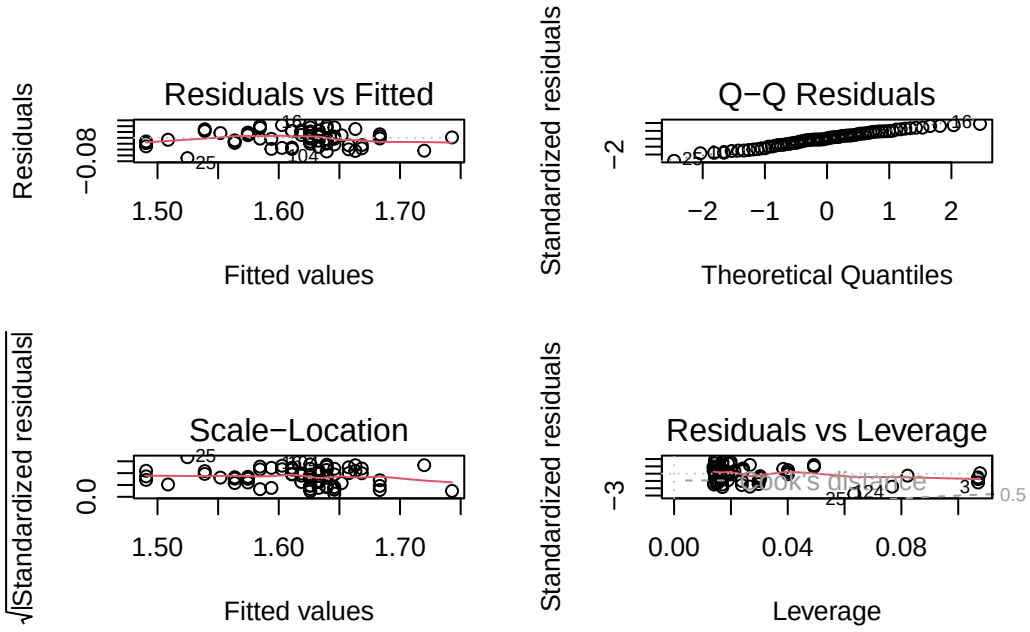
```

Call:
lm(formula = log10(fklngth) ~ log10(age), data = sturgeon.male,
    subset = !(rownames(sturgeon.male) %in% c("8", "24", "112")))

Residuals:
      Min        1Q    Median        3Q       Max
-0.069163 -0.017390  0.000986  0.018590  0.047647

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.22676    0.02431   50.46   <2e-16 ***
log10(age)   0.31219    0.01932   16.16   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02554 on 70 degrees of freedom
  (5 observations deleted due to missingness)
Multiple R-squared:  0.7885,    Adjusted R-squared:  0.7855
```

```
F-statistic:   261 on 1 and 70 DF,  p-value: < 2.2e-16
```

L'ordonnée à l'origine (Intercept), la pente, et le R carré sont presque les mêmes, et la valeur de p est encore astronomiquement petite. Enlever les valeurs extrêmes a peu d'effet dans ce cas.

Les graphiques diagnostiques des résidus et les tests formels des conditions d'application sur ce sous-ensemble de données donnent:

```
par(mfrow = c(2, 2))
plot(RegModel.3)
```




```
sturgeon.male.subset <- subset(sturgeon, subset = !(rownames(sturgeon.male) %in% c("8", "24", "11
bptest(RegModel.3)
```

```
    studentized Breusch-Pagan test

data:  RegModel.3
BP = 0.3001, df = 1, p-value = 0.5838
```

```
dwtest(RegModel.3)
```

```
    Durbin-Watson test

data:  RegModel.3
DW = 2.0171, p-value = 0.5074
alternative hypothesis: true autocorrelation is greater than 0
```

```
resettest(RegModel.3)
```

```
    RESET test

data:  RegModel.3
RESET = 3.407, df1 = 2, df2 = 68, p-value = 0.0389
```

```
shapiro.test(residuals(RegModel.3))
```

```
    Shapiro-Wilk normality test

data:  residuals(RegModel.3)
W = 0.98318, p-value = 0.4502
```

Il n'y a pas vraiment de différence ici non plus avec l'analyse des données en entier. Bref, tout pointe vers la conclusion que les valeurs les plus extrêmes de cet ensemble de donnée n'influencent pas indûment les résultats statistiques.

3.9. Quantifier la taille d'effet et analyse de puissance en régression

L'interprétation biologique des résultats n'est pas la même chose que l'interprétation statistique. Dans l'analyse qui précède, on conclue statistiquement que la taille augmente avec l'âge (puisque la pente est positive et et $p<0.05$). Mais cette augmentation "statistique" de la taille avec l'âge ne donne pas d'information sur la différence de taille entre les jeunes et vieux individus. La pente et un graphique sont plus informatifs à ce sujet que la valeur p. La pente (dans l'espace log-log) est 0.34. Cela veut dire que pour chaque unité d'accroissement de X (log10(age)), il y a une augmentation de 0.34 unités de log10(fklngth). En d'autres mots, quand l'âge est multiplié par 10, la longueur à la fourche est multipliée environ par 2 ($10^{0.34}$). Donc la longueur des esturgeons augmente plus lentement que leur âge (contrairement à mon tour de taille, semble-t-il….). La valeur de la pente (0.34) est un estimé de la taille de l'effet de l'âge sur la longueur.

Il est aussi importnat d'estimer l'intervalle de confiance sur la pente pour pouvoir estimer si l'intervalle inclus ou non que des valeurs biologiquement importantes. Cela peut être fait simplement avec la fonction `confint()`.

```
confint(RegModel.2)
```

```
                2.5 %   97.5 %
(Intercept) 1.1377151 1.246270
log10(age)  0.2976433 0.384068
```

L'intervalle de confiance à 95% de la pente est 0.29-0.38. L,intervalle de confiance est assez faible et éloigné de zéro.

3.9.1. Puissance de détecter une pente donnée

Pour les calculs de puissance avec G*Power vous devrez cependant utiliser une autre métrique de la taille de l'effet, calculée à partir de la pente, de son erreur-type, et de la taille de l'échantillon (ce qui facilite les calculs pour G*Power, mais malheureusement pas pour vous ;-) La métrique (d) est calculée comme:

$$d = \frac{b}{s_b\sqrt{n-k-1}}$$

où b est l'estimé de la pente, s_b est l'erreur type de la pente, n est le nombre d'observations, et k est le nombre de variables indépendantes (1 pour la régression linéaire simple).

Vous pouvez calculer approximativement la puissance avec G*Power pour une valeur de pente que vous jugez assez grande pour mériter d'être détectée. Allez à **Tests: Means: One group: difference from constant**, là, vous devrez remplacer la valeur de b dans l'équation pour la taille d'effet (d) par la pente que vous voudriez détecter, mais utiliser l'erreur type calculée à partir de vos données.

Par exemple, supposons que les ichthyologues considèrent qu'une pente de 0.1 pour la relation entre log10(fklngth) et log10(age) est signifiante biologiquement, et qu'ils désirent estimer la puissance de détecter une telle pente à partir d'un échantillon de 20 esturgeons. Les résultats de la régression log-log nous fournissent ce dont on a besoin:

```
summary(RegModel.2)
```

```

Call:
lm(formula = log10(fklngth) ~ log10(age), data = sturgeon.male)

Residuals:
      Min        1Q    Median        3Q       Max
-0.082794 -0.016837 -0.000719  0.021102  0.087446

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.19199    0.02723   43.77   <2e-16 ***
log10(age)   0.34086    0.02168   15.72   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.03015 on 73 degrees of freedom
  (5 observations deleted due to missingness)
Multiple R-squared:  0.772, Adjusted R-squared:  0.7688
F-statistic: 247.1 on 1 and 73 DF,  p-value: < 2.2e-16
```

L'erreur-type de la pente est 0.02168. Il y avait 75 poissons (n=75) dans l'échantillon de départ. On peut donc calculer la métrique de taille d'effet pour G*Power

$$d = \frac{b}{s_b\sqrt{n-k-1}} = \frac{0.1}{0.02168\sqrt{74-1-1}} = 0.54$$

Armés de cette taille d'effet (une pente présumée de 0.1 et une variabilité autour de la régression similaire à la régression de fklngth vs age), aller à **Tests: Means: One group: difference from constant**, et entrez la valeur calculée de d, alpha, et l'effectif de l'échantillon pour calculer la puissance.

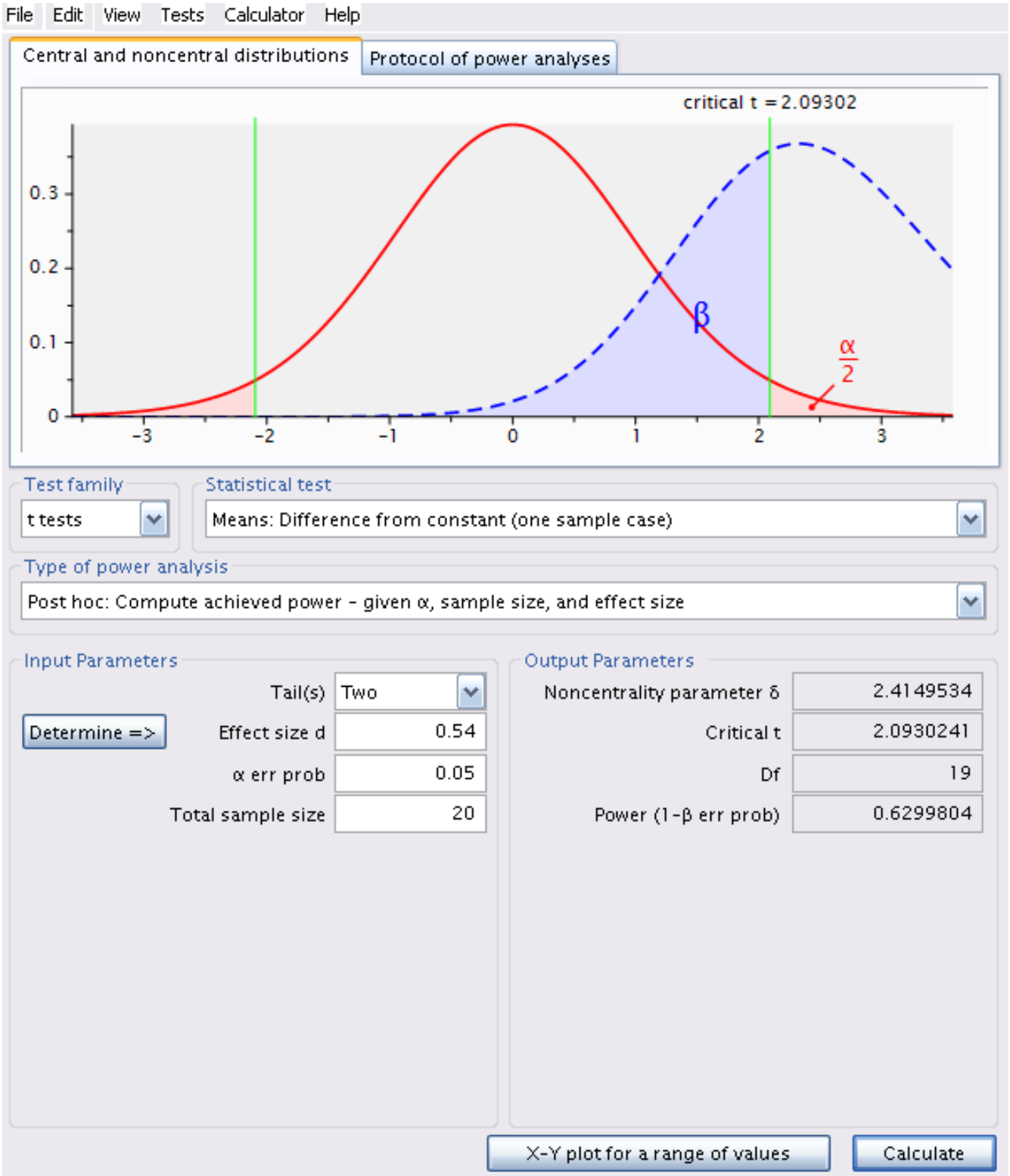


Figure 3.3.: Analyse de puissance pour N = 20 et pente = 0.1

Dans R, il est possible de faire cette analyse avec le code suivant:

```r
library(pwr)

# analyse de puissance
pwr.t.test(n = 20, d = 0.54, sig.level = 0.05, type = "one.sample")
```

```
     One-sample t test power calculation

              n = 20
              d = 0.54
      sig.level = 0.05
          power = 0.6299804
    alternative = two.sided
```

La puissance de détecter une pente comme étant statistiquement significative (au niveau alpha), si la pente est 0.1, que la variabilité résiduelle autour de la régression est semblable à celle de notre échantillon (ce qui revient à une taille d'effet de 0.54, pour un échantillon de 20 esturgeons et alpha=0.05) est de 0.629. Seulement environ 2/3 des échantillons de cette taille détecteraient un effet significatif de l'âge sur `fklngth`.

3.9.2. Effectif requis pour atteindre une puissance désirée (test A-priori)

Pour estimer la taille d'échantillon (effectif) requis pour avoir une puissance de 99% de détecter un effet de l'âge si la pente est 0.1 (sur une échelle log-log), avec alpha=0.05, on utilise la même valeur de d (0.54):

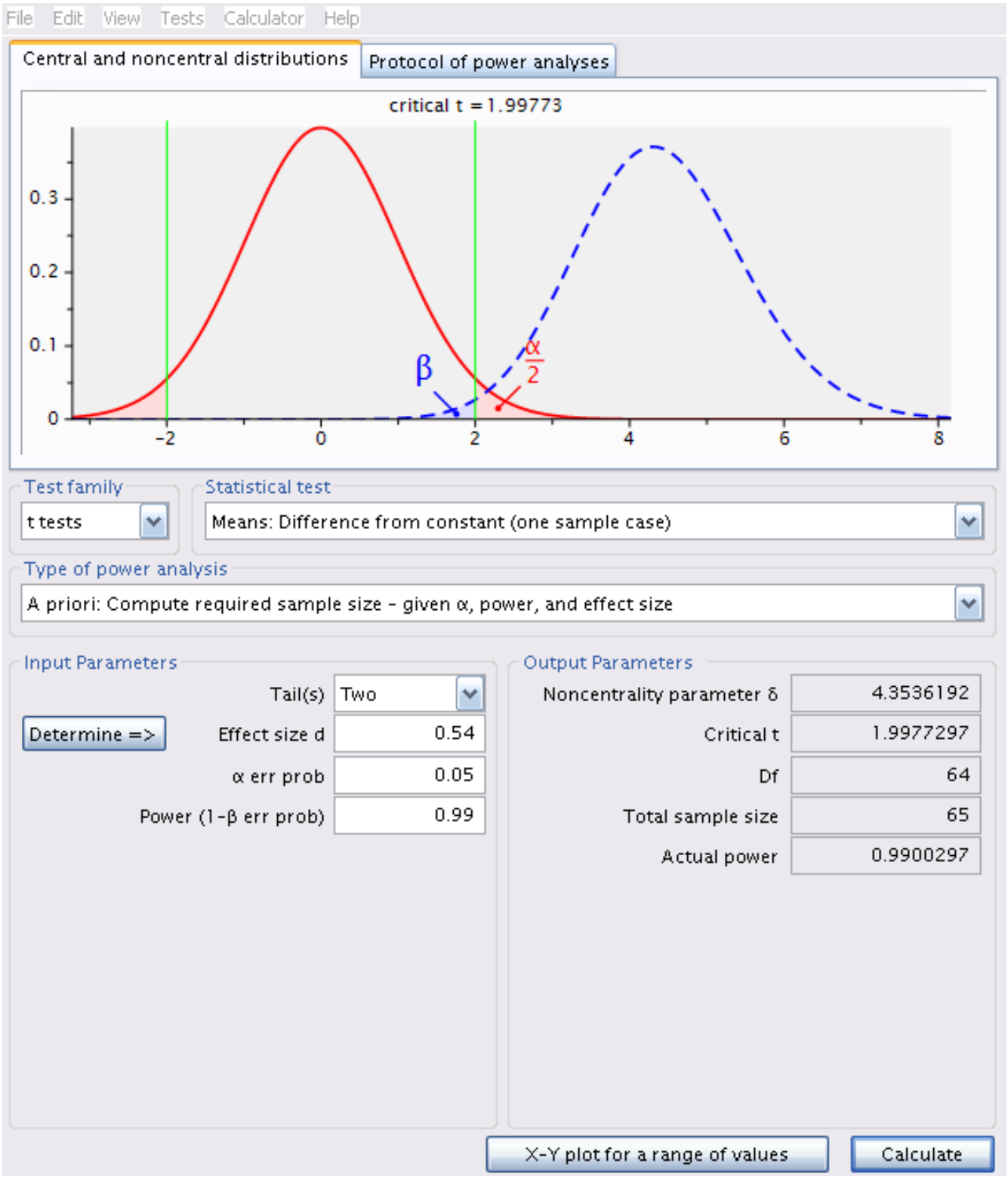


Figure 3.4.: Analyse à priori pour déterminer la taille d'échantillon pour une puissance de 0.99

Dans R, il est possible de faire cette analyse avec le code suivant:

```
library(pwr)

# analyse de puissance
pwr.t.test(n = 65, d = 0.54, sig.level = 0.05, type = "one.sample")
```

```
     One-sample t test power calculation

              n = 65
              d = 0.54
      sig.level = 0.05
          power = 0.9900297
    alternative = two.sided
```

En augmentant la taille de l'échantillon à 65, selon le même scénario que précédemment, la puissance augmente à 99%.

3.10. Bootstrap en régression simple avec R

Un test non paramétrique pour l'ordonnée à l'origine et la pente d'une régression simple peut être effectué par bootstrap.

```
# charger le paquet boot
library(boot)
# obtenir les poids de régression
bs <- function(formula, data, indices) {
  d <- data[indices, ] # allows boot to select sample
  fit <- lm(formula, data = d)
  return(coef(fit))
}
# bootstrapping with 1000 replications
results <- boot(
  data = sturgeon.male,
  statistic = bs,
  R = 1000, formula = log10(fklngth) ~ log10(age)
)
# view results
results
```

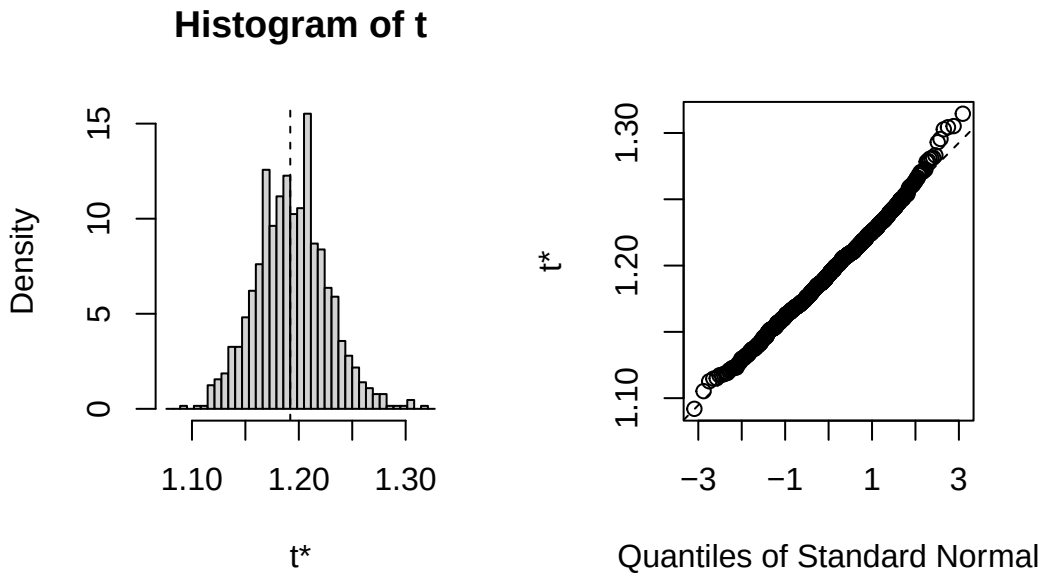
```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = sturgeon.male, statistic = bs, R = 1000, formula = log10(fklngth) ~
```

```
    log10(age))


Bootstrap Statistics :
     original       bias    std. error
t1* 1.1919926  0.001645875  0.03298142
t2* 0.3408557 -0.001145732  0.02610819
```

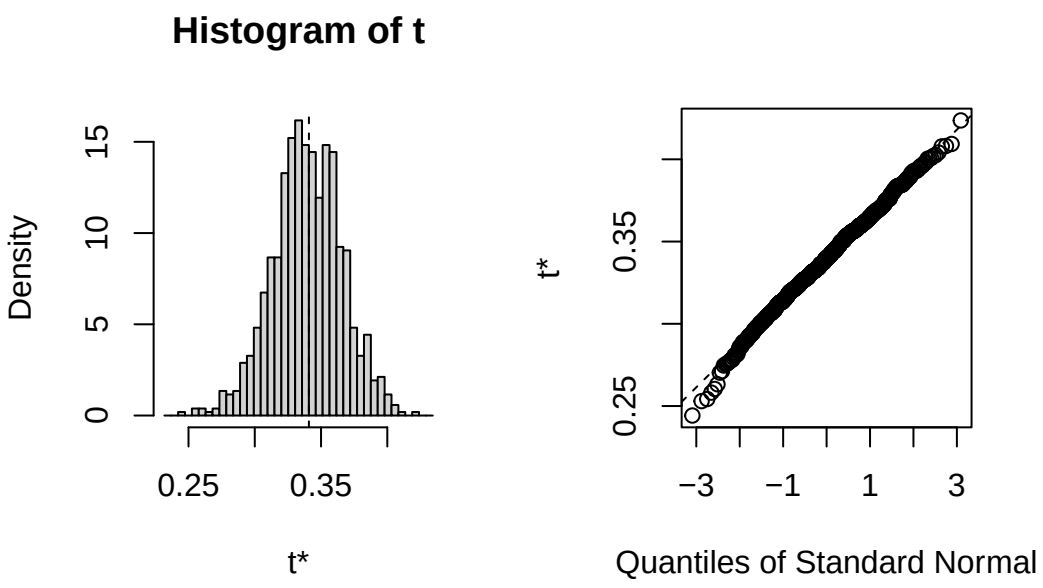
Pour chaque paramètre du modèle (ici l'ordonnée à l'origine est appelée t1* et la pente de la régression t2*), R imprime :

1. `original` la valeur estimée sur tout l'échantillon
2. `bias` la différence entre la valeur moyenne des estimés par bootstrap et la valeur originale sur tout l'échantillon
3. `std. error` l'erreur-type de l'estimé bootstrap

```
par(mfrow = c(2, 2))
plot(results, index = 1) # intercept
```




```
plot(results, index = 2) # log10(age)
```



La distribution des estimés obtenus par bootstrap est assez normale dans cet exemple, avec de petites déviations dans les queuee de la distribution (là où ça compte pour les intervalles de confiance…). On pourrait utiliser l'erreur-type des estimés bootstrap pour calculer un intervalle de confiance symétrique (moyenne +- t ET). Cependant, comme R peut facilement calculer des intervalles de confiance qui corrigent pour le biais (BCa) ou encore des intervalle empiriques à partir des distributions simulées (méthode Percentile) il peut être aussi simple de les calculer selon les 3 méthodes:

```
# interval de confiance pour l'ordonnée à l'origine
boot.ci(results, type = "all", index = 1)
```

```
Warning in boot.ci(results, type = "all", index = 1): bootstrap variances
needed for studentized intervals


BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "all", index = 1)

Intervals :
Level      Normal              Basic
95%   ( 1.126,  1.255 )   ( 1.122,  1.254 )

Level     Percentile            BCa
95%   ( 1.130,  1.262 )   ( 1.122,  1.251 )
Calculations and Intervals on Original Scale
```

```r
# intervalle de confiance pour la pente
boot.ci(results, type = "all", index = 2)
```

```
Warning in boot.ci(results, type = "all", index = 2): bootstrap variances
needed for studentized intervals


BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "all", index = 2)

Intervals :
Level      Normal              Basic
95%   ( 0.2908,  0.3932 )   ( 0.2899,  0.3954 )

Level     Percentile            BCa
95%   ( 0.2863,  0.3918 )   ( 0.2942,  0.3979 )
Calculations and Intervals on Original Scale
```

Ici, les 4 types d'intervalles de confiance que R a calculé sont essentiellement semblables. Si les données avaient violé plus sévèrement les conditions d'application de la régression (normalité, homoscedasticité), alors les différentes méthodes (Normal, Basic, Percentile, et BCa) auraient divergé un peu plus. Lequel choisir alors? BCa est celui qui est préféré de la majorité des praticiens, présentement.

4. Comparaison de deux échantillons

Après avoir complété cet exercice de laboratoire, vous devriez pouvoir:

- Utiliser R pour examiner visuellement vos données
- Utiliser R pour comparer les moyennes de deux échantillons tirés de populations normales
- Utiliser R pour comparer les moyennes de deux échantillons tirés de populations qui ne sont pas normales
- Utiliser R pour comparer les moyennes de deux échantillons appariés.

4.1. Paquets et données requises pour le labo

Ce laboratoire nécessite:

- les paquets R:
  - car
  - lmtest
  - boot
  - lmPerm
- les fichiers de données
  - sturgeon.csv
  - skulldat_2020.csv

```
library(car)
library(lmtest)
library(boot)
library(lmPerm)
library(ggplot2)
sturgeon <- read.csv("data/sturgeon.csv")
skull <- read.csv("data/skulldat_2020.csv")
```

4.2. Examen visuel des données

Une des premières étapes dans toute analyse de données est l'examen visuel des données par des graphiques et statistiques sommaires pour détecter les distributions sous-jacentes, les valeurs extrêmes et les tendances dans vos données. Cela commence souvent avec des graphiques de vos données (histogrammes, diagrammes de probabilité, Box plots, etc.) qui vous permettent d'évaluer si vos données sont normales, si elles sont corrélées les unes aux autres, ou s'il y a des valeurs suspectes dans le fichier.

Supposons que l'on veuille comparer la distribution en taille des esturgeons de The Pas et Cumberland House. La variable `fklngth` dans le fichier `sturgeon.csv` représente la longueur (en cm) à la fourche de chaque poisson mesurée de l'extrémité de la tête à la base de la fourche de la nageoire caudale. Pour commencer, examinons si cette variable est normalement distribuée. On ne va pas tester pour la normalité à ce stade-ci; la présomption de normalité dans les analyses paramétriques s'applique aux résidus et non aux données brutes. Cependant, si les données brutes ne sont pas normales, vous avez d'habitude une très bonne raison de soupçonner que les résidus vont aussi ne pas avoir une distribution normale.

Une excellente façon de comparer visuellement une distribution à la distribution normale est de superposer un histogramme des données observées à une courbe normale. Pour ce faire, il faut procéder en deux étapes :

1. indiquer à R que nous voulons créer un histogramme superposé à une courbe normale
2. spécifier qu'on veut que les graphiques soient faits pour les deux sites

- En utilisant les données du fichier `sturgeon.csv`, générez les histogrammes et les approximations des distributions normales ajustées aux données de `fklngth` à `The Pas` et `Cumberland House`.

```
# use "sturgeon" dataframe to make plot called mygraph
# and define x axis as representing fklngth
mygraph <- ggplot(
  data = sturgeon,
  aes(x = fklngth)
) +
  xlab("Fork length (cm)")
# add data to the mygraph ggplot
mygraph <- mygraph +
  geom_density() + # add data density smooth
  geom_rug() + # add rug (bars at the bottom of the plot)
  geom_histogram( # add black semitransparent histogram
    aes(y = ..density..),
    color = "black", alpha = 0.3
  ) +
  # add normal curve in red, with mean and sd from fklength
  stat_function(
    fun = dnorm,
    args = list(
      mean = mean(sturgeon$fklngth),
      sd = sd(sturgeon$fklngth)
    ),
    color = "red"
  )
# display graph, by location
mygraph + facet_grid(. ~ location)
```

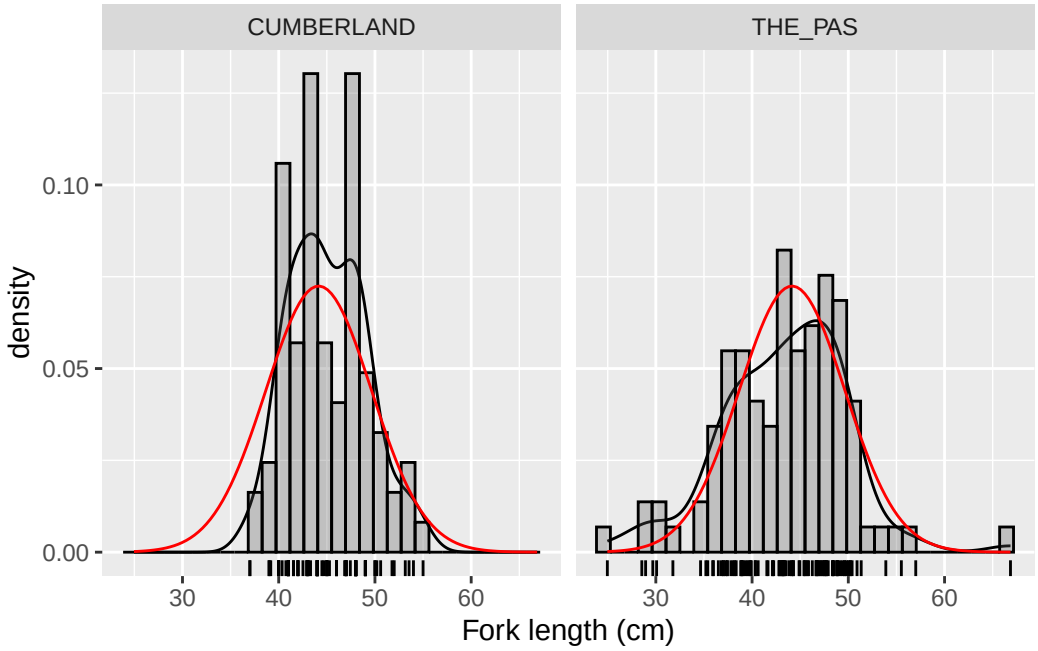


Figure 4.1.: Distribution de la longueur des esturgeons

Examinez ce graphique et essayez de déterminer si ces deux échantillons sont normalement distribués. À mon avis, cette variable est approximativement normalement distribuée dans les deux échantillons. Puisque ce qui nous intéresse est de comparer la taille des poissons de deux sites différents, c'est probablement une bonne idée de créer un graphique qui compare les deux groupes de données. Un Box plot convient très bien pour cette tâche.

- Tracez un boxplot de `fklngth` groupé par `location`. Que concluez-vous quant à la différence entre les deux sites?

```
ggplot(data = sturgeon, aes(
  x = location,
  y = fklngth
)) +
  geom_boxplot(notch = TRUE)
```

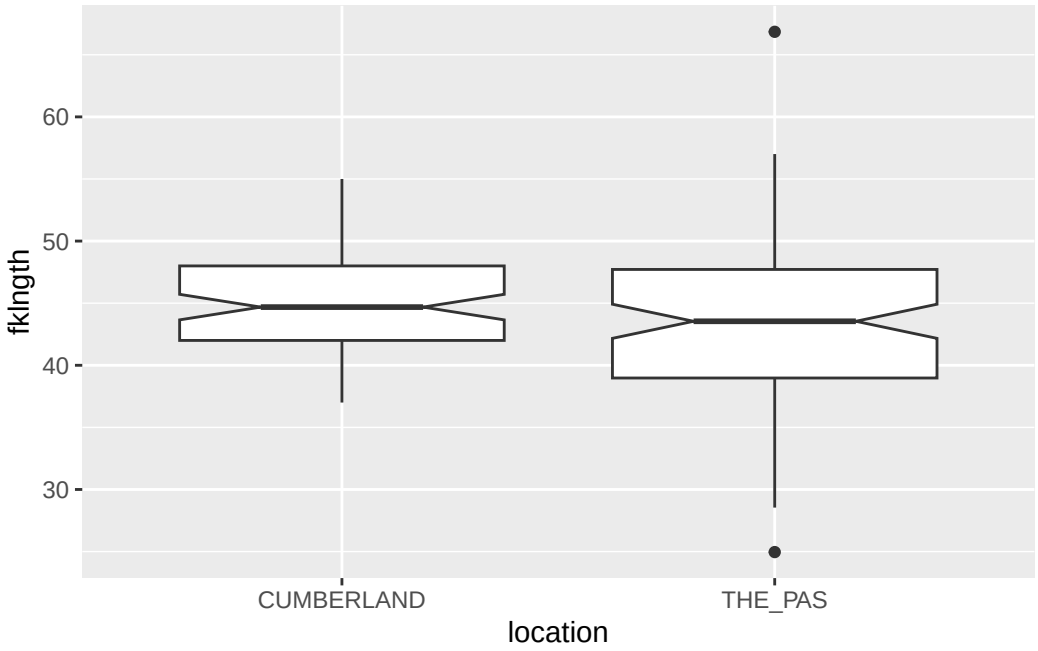


Figure 4.2.: Boxplot de la longueur des esturgeons

Il n'y a pas de grande différence de taille entre les deux sites, mais la taille des poissons à The Pas est plus variable ayant une plus large étendue de taille et des valeurs extrêmes (définies par les valeurs qui sont $>$ 1.5*l'étendue interquartile) à chaque bout de la distribution.

4.3. Comparer les moyennes de deux échantillons indépendants

Éprouvez l'hypothèse nulle d'égalité de la longueur à la fourche à The Pas et Cumberland House de 3 manières différentes:

1. paramétriques supposant des variances égales
2. paramétriques supposant des variances différentes
3. non-paramétrique (pas de conditions d'aplications sur la distribution et la variance)

Que concluez-vous?

```
# t-test assuming equal variances
t.test(
  fklngth ~ location,
  data = sturgeon,
  alternative = "two.sided",
  var.equal = TRUE
)
```

```
    Two Sample t-test
```

```
data:  fklngth by location
t = 2.1359, df = 184, p-value = 0.03401
alternative hypothesis: true difference in means between group CUMBERLAND and group THE_PAS is no
95 percent confidence interval:
 0.1308307 3.2982615
sample estimates:
mean in group CUMBERLAND    mean in group THE_PAS
                45.08439                 43.36984
```

```r
# t-test assuming unequal variances
t.test(
  fklngth ~ location,
  data = sturgeon,
  alternative = "two.sided",
  var.equal = FALSE
)
```

```
    Welch Two Sample t-test

data:  fklngth by location
t = 2.2201, df = 169.8, p-value = 0.02774
alternative hypothesis: true difference in means between group CUMBERLAND and group THE_PAS is no
95 percent confidence interval:
 0.1900117 3.2390804
sample estimates:
mean in group CUMBERLAND    mean in group THE_PAS
                45.08439                 43.36984
```

```r
# test non paramétrique
wilcox.test(
  fklngth ~ location,
  data = sturgeon,
  alternative = "two.sided"
)
```

```
    Wilcoxon rank sum test with continuity correction

data:  fklngth by location
W = 4973, p-value = 0.06296
alternative hypothesis: true location shift is not equal to 0
```

En se fiant au test de t, on rejette donc l'hypothèse nulle. Il y a une différence significative entre les deux moyennes des longueurs à la fourche.

Notez que si l'on se fie au test de *Wilcoxon*, il faut accepter l'hypothèse nulle. Les deux tests mènent donc à des conclusions contradictoires. La différence significative obtenue par le test de t peut provenir en partie d'une violation des conditions d'application du test (normalité et homoscédasticité). D'un autre côté, l'absence de différence significative selon le test de Wilcoxon pourrait être due au fait que, pour un effectif donné, la puissance du test non paramétrique est inférieure à celle du test paramétrique correspondant. Compte tenu 1) des valeurs de p obtenues pour les deux tests, et 2) le fait que pour des grands échantillons (des effectifs de 84 et 101 sont considérés grands) le test de t est considéré robuste, il est raisonnable de rejeter l'hypothèse nulle.

Avant d'accepter les résultats du test de t et de rejeter l'hypothèse nulle qu'il n'y a pas de différences de taille entre les deux sites, il est important de déterminer si les données remplissent les conditions de normalité des résidus et d'égalité des variances. L'examen préliminaire suggérait que les données sont à peu près normales mais qu'il y avait peut-être des problèmes avec les variances (puisque l'étendue des données pour The Pas était beaucoup plus grande que celle pour Cumberland). On peut examiner ces conditions d'application plus en détail en examinant les résidus d'un modèle linéaire et en utilisant les graphiques diagnostiques:

```
m1 <- lm(fklngth ~ location, data = sturgeon)
par(mfrow = c(2, 2))
plot(m1)
```

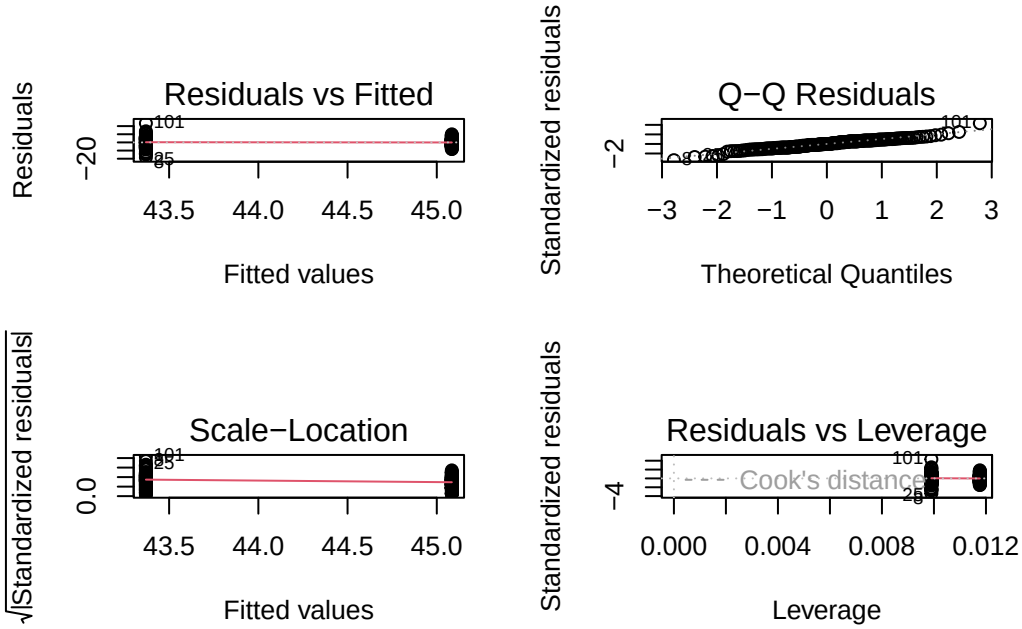


Figure 4.3.: Condition d'application du modèle linéaire

Le premier graphique ci-dessus montre comment les résidus se distribuent autour des valeurs prédites (les moyennes) pour chaque site et permette de juger si il semble y avoir un problème de normalité ou d'homoscédasticité. Si les variances étaient égales dans les deux sites, l'étendue verticale des résidus

tendrait à être la même. Sur le graphique, on voit que l'étendue des résidus est plus grande à gauche (le site où la taille moyenne est la plus faible), ce qui suggère un possible problème d'homogénéité des variances. On peut tester cela plus formellement en comparant la moyenne de la valeur absolue des résidus.(on y reviendra; c'est le test de Levene).

Le second graphique est un graphique de probabilité (graphique Q-Q) des résidus. Comme ici, les points tombent près de la diagonale, il ne semble pas y avoir de problème important avec la normalité. On peut faire un test formel de la condition de normalité par le test de Shapiro- Wilk:

```
shapiro.test(residuals(m1))
```

```
    Shapiro-Wilk normality test

data:  residuals(m1)
W = 0.97469, p-value = 0.001857
```

Hummm. Ce test indique que les résidus ne sont pas normaux, ce qui contredit notre évaluation visuelle. Cependant, puisque (a) la distribution des résidus ne s'éloigne pas beaucoup de la normalité et (b) le nombre d'observations à chaque site est raisonnablement grand (i.e. >30), on n'a pas à être trop inquiet quant à l'impact de cette violation de normalité sur la fiabilité du test.

Qu'en est-il de l'égalité des variances?

```
library(car)
leveneTest(m1)
```

```
Warning in leveneTest.default(y = y, group = group, ...): group coerced to
factor.

Levene's Test for Homogeneity of Variance (center = median)
       Df F value    Pr(>F)
group   1  11.514 0.0008456 ***
      184
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
bptest(m1)
```

```
    studentized Breusch-Pagan test

data:  m1
BP = 8.8015, df = 1, p-value = 0.00301
```

Les résultats qui précédents proviennent de 3 des tests disponibles en R (dans les package car et lmtest) qui éprouvent l'hypothèse de l'égalité des variances dans des tests de t ou des modèles linéaires ayant uniquement des variables indépendantes discontinues ou catégoriques. Il est redondant de faire les 2 tests. Si ils sont présentés ici, c'est que ces 2 tests sont usuels et qu'il n'y a pas consensus quant au meilleur des deux. Le test de Levene est le plus connu et utilisé. Il compare la moyenne des valeurs absolues des résidus dans les deux groupes. Le test Breusch-Pagan a l'avantage d'être applicable à une plus large gamme de modèles linéaires (il peut être utilisé également avec des variables indépendantes continues, comme en régression). Ici, les deux tests mènent à la même conclusion: la variance diffère entre les deux sites.

Sur la base de ces résultats, on peut conclure qu'il y a évidence (mais faible) pour rejeter l'hypothèse nulle qu'il n'y a pas de différence dans la taille de poissons entre les deux sites. On a utilisé une modification du test de t pour tenir compte du fait que les variances ne sont pas égales et nous sommes satisfaits que la condition de normalité des résidus a été remplie. Alors, fklngth à Cumberland est plus grande que fklngth à The Pas.

4.4. Bootstrap et tests de permutation pour comparer deux moyennes

4.4.1. Bootstrap

Le bootstrap et les tests de permutation peuvent être utilisés pour comparer les moyennes (ou d'autres statistiques). Le principe général est simple et peut être effectué de diverses façons. Ici j'utilise certains des outils disponibles et le fait qu'une comparaison de moyenne peut être représentée par un modèle linéaire. On pourra utiliser un programme similaire plus tard quand on ajustera des modèles plus complexes.

```
library(boot)
```

La première section sert à définir une fonction (ici appelée bs) qui extraie les coefficients d'un modèle ajusté :

```
# function to obtain model coefficients for each iteration
bs <- function(formula, data, indices) {
  d <- data[indices, ]
  fit <- lm(formula, data = d)
  return(coef(fit))
}
```

La deuxième section avec la commande boot() fait le gros du travail: on prend les données dans sturgeon, on les bootstrap $R = 1000$ fois, et chaque fois on ajuste le modèle `fklngth` vs `location` et on garde les valeurs calculées par la fonction `bs`.

```
# bootstrapping with 1000 replications
results <- boot(
  data = sturgeon, statistic = bs, R = 1000,
  formula = fklngth ~ location
)
# view results
results
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = sturgeon, statistic = bs, R = 1000, formula = fklngth ~
    location)


Bootstrap Statistics :
     original        bias    std. error
t1* 45.084391  0.0005660241   0.4082100
t2* -1.714546 -0.0122179717   0.7721073
```

On obtient les estimés originaux pour les deux coefficients du modèle: la moyenne pour le premier (alphabétiquement) site soit Cumberland, et la différence entre les deux moyennes à Cumberland et The Pas. C'est ce second paramètre, la différence entre les moyennes, qui nous intéresse.

```
plot(results, index = 2)
```

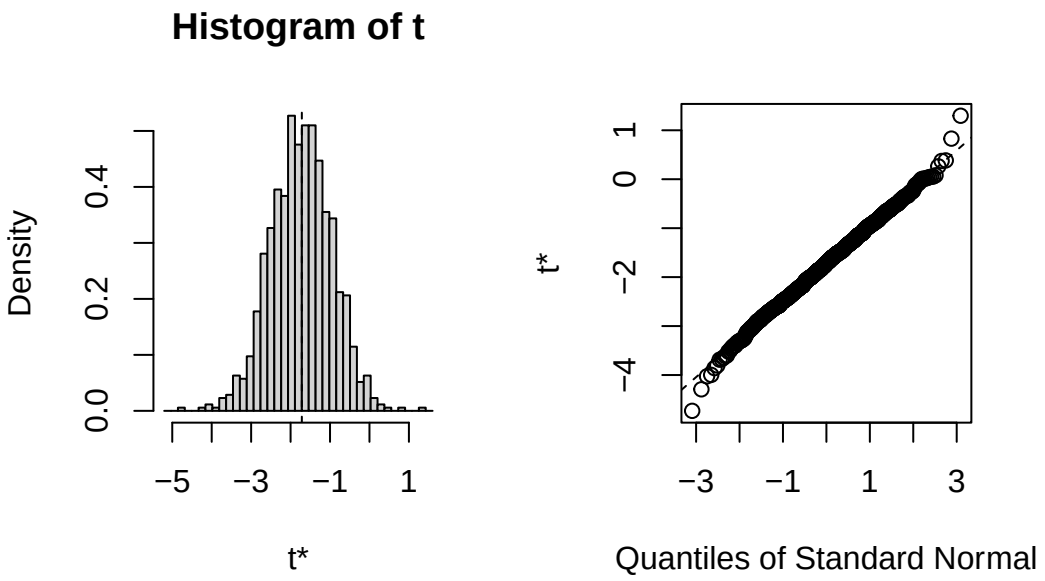


Figure 4.4.: Normalité des estimés de la différence des moyennes par bootstrap

```
# get 95% confidence intervals
boot.ci(results, type = "bca", index = 2)
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
```

```
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 2)

Intervals :
Level       BCa
95%   (-3.295, -0.257 )
Calculations and Intervals on Original Scale
```

Comme l'intervalle de confiance n'inclue pas 0, on conclue que les moyennes ne sont pas les mêmes.

4.4.2. Permutation

Les tests de permutation pour les modèles linéaires peuvent être effectués à l'aide du package `lmPerm`:

```
m1Perm <- lmp(
  fklngth ~ location,
  data = sturgeon,
  perm = "Prob"
)
```

```
[1] "Settings:  unique SS "
```

La fonction `lmp()` fait tout le travail pour nous. Ici, cette fonction est effectuée avec l'option perm pour choisir la règle utilisée pour stopper les calculs. L'option Probs arrête les permutations quand la déviation standard estimée pour la p-valeur tombe sous un seuil déterminé. C'est l'une des nombreuses règles qui peuvent possiblement être utilisées pour ne faire les permutations que sur un sous-ensemble des permutations possibles (ce qui prendrait souvent trrrrrès longtemps).

```
summary(m1Perm)
```

```
Call:
lmp(formula = fklngth ~ location, data = sturgeon, perm = "Prob")

Residuals:
      Min        1Q    Median        3Q       Max
-18.40921  -3.75370  -0.08439   3.76598  23.48055

Coefficients:
          Estimate Iter Pr(Prob)
location1   0.8573 5000   0.0088 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 5.454 on 184 degrees of freedom
Multiple R-Squared: 0.02419,    Adjusted R-squared: 0.01889
F-statistic: 4.562 on 1 and 184 DF,  p-value: 0.03401
```

1. `Iter`: la règle a limité à 5000 permutations le calcul. Notez que ce nombre va varier à chaque fois que vous tournerez cet petit bout de code. Ce sont des résultats obtenus par permutations aléatoires, donc vous devez vous attendre à de la variabilité. .
2. `Pr(Prob)`: La p-valeur estimée pour H0 est 0.0088. La différence observée pour fklngth between entre les deux sites était plus grande que les valeurs permutées environ (1 - 0.0088= 99.1%) des 5000 permutations. Notez que 5000 permutations ce n'est pas un si grand nombre de permutations que ça, et donc les faibles valeurs de p ne sont pas très précises. Si vous voulez des valeurs précises de p, vous devrez faire plus de permutations. Vous pouvez ajuster 2 paramètres: maxIter, le nombre maximal de permutations (défaut 5000) et Ca, le seuil de précision désiré qui arrête les permutations quand l'erreur- type de p est plus petite que Ca*p (défaut=0.1)
3. `F-statistic`: Le reste est la sortie standard pour un modèle ajusté à des données, avec le test paramétrique. Ici, la p-valeur, présumant que toutes les conditions d'application sont remplies, est 0.034.

4.5. Comparer les moyennes de deux échantillons appariés

> **Warning**
>
> Pour la section suivante veuillez télécharger le nouveau fichier `skulldat_2020.csv` qui a été récemment ajouté sur Brightspace.

Dans certaines expériences les mêmes individus sont mesurés deux fois, par exemple avant et après un traitement ou encore à deux moments au cours de leur développement. Les mesures obtenues lors de ces deux événements ne sont pas indépendantes, et des comparaisons de ces mesures appariées doivent être faites.

Le fichier `skulldat_2020.csv` contient des mesures de la partie inférieure du visage de jeunes filles d'Amérique du Nord prises à 5 ans, puis à 6 ans (données de Newman and Meredith, 1956).

- Pour débuter, éprouvons l'hypothèse que la largeur de la figure est la même à 5 ans et à 6 ans en assumant que les mesures viennent d'échantillons indépendants.

```r
skull <- read.csv("data/skulldat_2020.csv")
t.test(width ~ age,
  data = skull,
  alternative = "two.sided",
  paired = FALSE
)
```

```
    Welch Two Sample t-test
```

```
data:  width by age
t = -1.7812, df = 27.93, p-value = 0.08576
alternative hypothesis: true difference in means between group 5 and group 6 is not equal to 0
95 percent confidence interval:
 -0.43002624  0.03002624
sample estimates:
mean in group 5 mean in group 6
       7.461333        7.661333
```

Jusqu'à maintenant, nous avons spécifié le test de t en utilisant une notation de type `formule` avec `y ~ x` où `y` est la variable pour laquelle on souhaite comparer les moyennes et `x` correspond à une variable définissant les groupes. Cela marche bien lorsque les données de sont pas pairées et sont présentées dans un format de type *long* où les données prise dans une même catégorie ou sur une même personne sont simplement les unes en-dessous des autres avec des colonnes indiquant l'appartenance des mesures aux différentes catégories (voir la structure de `skull` par exemple) Dans l'objet `skull`, il y a 3 colonnes:

- `width`: largeur de la tête
- `age`: age lors de la mesure
- `id`: identité de la personne

```
head(skull)
```

```
  width age id
1  7.33   5  1
2  7.53   6  1
3  7.49   5  2
4  7.70   6  2
5  7.27   5  3
6  7.46   6  3
```

Quand les données sont pairées, il faut indiquer comment elle doivent être associées. Dans notre exemple, elles sont pairées par individu. Le format de données de type *long* indique cet appariement via la colonne `id`. Cependant, la fonction `t.test` ne permet pas de le prendre en compte. Il faut donc transformer les données en format de type *large* ou *horizontale* ou il y a une colonne différente pour chaque catégorie. Dans notre example, on souhaite avoir un fichier avec une colonne de mesure par age et où chaque ligne correspond à une personne différente. On peut modifier le format des données avec le code suivant.

```
skull_h <- data.frame(id = unique(skull$id))
skull_h$width5 <- skull$width[match(skull_h$id, skull$id) & skull$age == 5]
skull_h$width6 <- skull$width[match(skull_h$id, skull$id) & skull$age == 6]
head(skull_h)
```

```
  id width5 width6
1  1   7.33   7.53
2  2   7.49   7.70
3  3   7.27   7.46
```

```
4  4   7.93   8.21
5  5   7.56   7.81
6  6   7.81   8.01
```

Maintenant, effectuons le test apparié qui est approprié: Que conclure? Comment les résultats diffèrent-ils de la première analyse? Pourquoi?

```
t.test(skull_h$width5, skull_h$width6,
  alternative = "two.sided",
  paired = TRUE
)
```

```
    Paired t-test

data:  skull_h$width5 and skull_h$width6
t = -19.72, df = 14, p-value = 1.301e-11
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -0.2217521 -0.1782479
sample estimates:
mean difference
           -0.2
```

La première analyse a comme supposition que les deux échantillons de filles de 5 et 6 ans sont indépendants, alors que la deuxième analyse a comme supposition que la même fille a été mesurée deux fois, une fois à 5 ans, et la deuxième fois à 6 ans.

Notez que, dans le premier cas, on accepte l'hypothèse nulle, mais que le test apparié rejette l'hypothèse nulle. Donc, le test qui est approprié (le test apparié) indique un effet très significatif de l'âge, mais le test inapproprié suggère que l'âge n'importe pas. C'est parce qu'il y a une très forte corrélation entre la largeur du visage à 5 et 6 ans:

```
graphskull <- ggplot(data = skull_h, aes(x = width5, y = width6)) +
  geom_point() +
  labs(x = "Skull width at age 5", y = "Skull width at age 6") +
  geom_smooth() +
  scale_fill_continuous(low = "lavenderblush", high = "red")
graphskull
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

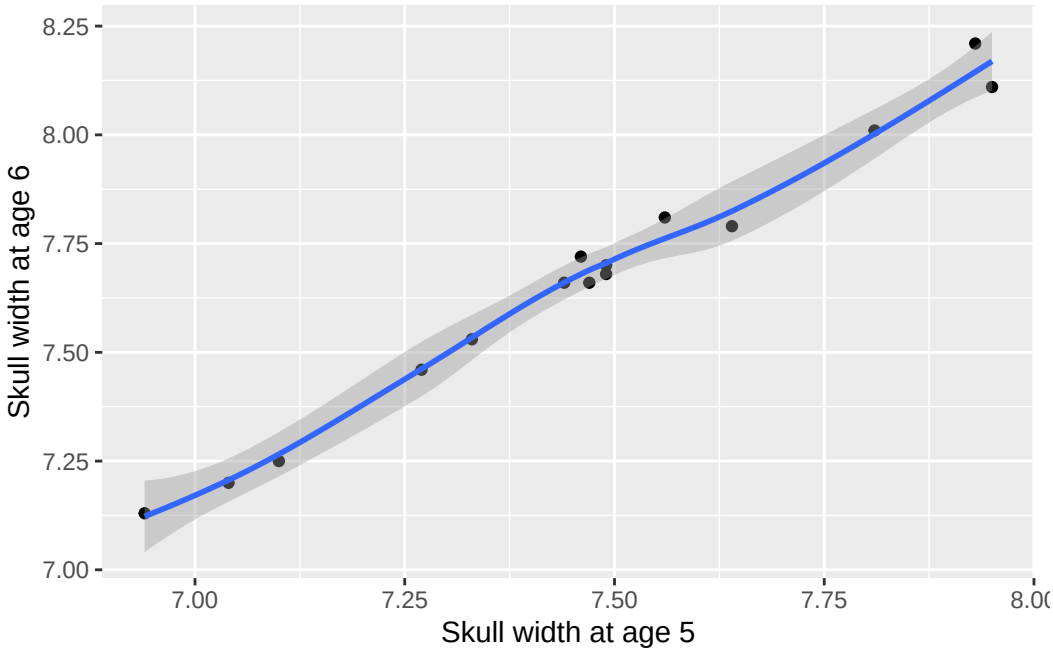


Figure 4.5.: Relation entre la taille de la tête à 5 et 6 ans

Avec $r = 0.9930841$. En présence d'une si forte corrélation, l'erreur-type de la différence appariée de largeur du visage entre 5 et 6 ans est beaucoup plus petit que l'erreur-type de la différence entre la largeur moyenne à 5 ans et la largeur moyenne à 6 ans. Par conséquent, la statistique t associée est beaucoup plus élevée pour le test apparié, la puissance du test est plus grande, et la valeur de p plus petite.

- Répétez l'analyse en utilisant l'alternative nonparamétrique, le test Wil-coxon signed-rank. (Que concluez-vous?

```
wilcox.test(skull_h$width5, skull_h$width6,
  alternative = "two.sided",
  paired = TRUE
)
```

```
Warning in wilcox.test.default(skull_h$width5, skull_h$width6, alternative =
"two.sided", : cannot compute exact p-value with ties
```

```
    Wilcoxon signed rank test with continuity correction

data:  skull_h$width5 and skull_h$width6
V = 0, p-value = 0.0007193
alternative hypothesis: true location shift is not equal to 0
```

Donc on tire la même conclusion qu'avec le test de t apparié et conclue qu'il y a des différences significatives entre la taille des crânes de filles âgées de 5 et 6 ans (quelle surprise!).

Mais, attendez une minute! On a utilisé des tests bilatéraux ici mais, compte tenu de s connaissances sur la croissance des enfants, une hypothèse unilatérale serait préférable. Ceci peut être accommodé en modifiant l'option "alternative". On utilise l'hypothèse alternative pour décider entre "less" ou "greater". Ici on s'attends que si il y a une différence, width5 va être inférieur à width6, donc on utiliserait "less".

```
t.test(skull_h$width5, skull_h$width6,
  alternative = "less",
  paired = TRUE
)
```

```
    Paired t-test

data:  skull_h$width5 and skull_h$width6
t = -19.72, df = 14, p-value = 6.507e-12
alternative hypothesis: true mean difference is less than 0
95 percent confidence interval:
       -Inf -0.1821371
sample estimates:
mean difference
           -0.2
```

```
wilcox.test(skull_h$width5, skull_h$width6,
  alternative = "less",
  paired = TRUE
)
```

```
Warning in wilcox.test.default(skull_h$width5, skull_h$width6, alternative =
"less", : cannot compute exact p-value with ties
```

```
    Wilcoxon signed rank test with continuity correction

data:  skull_h$width5 and skull_h$width6
V = 0, p-value = 0.0003597
alternative hypothesis: true location shift is less than 0
```

Pour estimer la puissance d'un test de t avec R, il faut utiliser la fonction `power.t.test()`. Il faut spécifier l'argument `type = "paired"`, utiliser la moyenne et l'écart-type de la différence au sein des paires dans les arguments `delta` et `sd`.

```r
skull_h$diff <- skull_h$width6 - skull_h$width5
power.t.test(
  n = 15,
  delta = mean(skull_h$diff),
  sd = sd(skull_h$diff),
  type = "paired"
)
```

```
     Paired t test power calculation

              n = 15
          delta = 0.2
             sd = 0.03927922
      sig.level = 0.05
          power = 1
    alternative = two.sided

NOTE: n is number of *pairs*, sd is std.dev. of *differences* within pairs
```

4.6. Références

Bumpus, H.C. (1898) The elimination of the unfit as illustrated by the introduced sparrow, Passer domesticus. Biological Lectures, Woods Hole Biology Laboratory, Woods Hole, 11 th Lecture: 209 - 226.

Newman, K.J. and H.V. Meredith. (1956) Individual growth in skele- tal bigonial diameter during the childhood period from 5 to 11 years of age. Amer. J. Anat. 99: 157 - 187.

5. ANOVA à un critère de classification

Après avoir complété cet exercice de laboratoire, vous devriez être capable de :

- Utiliser R pour effectuer une analyse de variance paramétrique à un critère de classification, suivie de comparaisons multiples
- Utiliser R pour vérifier si les conditions d'application de l'ANOVA paramétrique sont remplies
- Utiliser R pour faire une ANOVA à un critère de classification non-paramétrique
- Utiliser R pour transformer des données de manière à mieux remplir les conditions d'application de l'ANOVA paramétrique.

5.1. Paquets et données requises pour le labo

Ce laboratoire nécessite:

- les paquets R:
  - ggplot2
  - multcomp
  - car
- les fichiers de données
  - Dam10dat.csv

```
library(ggplot2)
library(car)
library(multcomp)
```

5.2. ANOVA à un critère de classification et comparaisons multiples

L'ANOVA à un critère de classification est l'analogue du test de t pour des comparaisons de moyennes de plus de deux échantillons. Les conditions d'application du test sont essentiellement les mêmes, et lorsque appliqué à deux échantillons ce test est mathématiquement équivalent au test de t.

En 1961-1962, le barrage Grand Rapids était construit sur la rivière Saskatchewan en amont de Cumberland House. On croit que durant la construction plusieurs gros esturgeons restèrent prisonniers dans des sections peu profondes et moururent. Des inventaires de la population d'esturgeons furent faits en 1954, 1958, 1965 et 1966. Au cours de ces inventaires, la longueur à la fourche (frklngth) furent mesurées (pas nécessairement sur chaque poisson cependant). Ces données sont dans le fichier `Dam10dat.csv`.

5.2.1. Visualiser les données

- À partir des données, vous devez d'abord changer le type de donnée de la variable `year`, pour que R traite `year` comme une variable discontinue (factor) plutôt que continue.

```
Dam10dat <- read.csv("data/Dam10dat.csv")
Dam10dat$year <- as.factor(Dam10dat$year)
str(Dam10dat)
```

```
'data.frame':   118 obs. of  21 variables:
 $ year    : Factor w/ 4 levels "1954","1958",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ fklngth : num  45 50 39 46 54.5 49 42.5 49 56 54 ...
 $ totlngth: num  49 NA 43 50.5 NA 51.7 45.5 52 60.2 58.5 ...
 $ drlngth : logi  NA NA NA NA NA NA ...
 $ drwght  : num  16 20.5 10 17.5 19.7 21.3 9.5 23.7 31 27.3 ...
 $ rdwght  : num  24.5 33 15.5 28.5 32.5 35.5 15.3 40.5 51.5 43 ...
 $ sex     : int  1 1 1 2 1 2 1 1 1 1 ...
 $ age     : int  24 33 17 31 37 44 23 34 33 47 ...
 $ lfkl    : num  1.65 1.7 1.59 1.66 1.74 ...
 $ ltotl   : num  1.69 NA 1.63 1.7 NA ...
 $ ldrl    : logi  NA NA NA NA NA NA ...
 $ ldrwght : num  1.2 1.31 1 1.24 1.29 ...
 $ lrdwght : num  1.39 1.52 1.19 1.45 1.51 ...
 $ lage    : num  1.38 1.52 1.23 1.49 1.57 ...
 $ rage    : int  4 6 3 6 7 7 4 6 6 7 ...
 $ ryear   : int  1954 1954 1954 1954 1954 1954 1954 1954 1954 1954 ...
 $ ryear2  : int  1958 1958 1958 1958 1958 1958 1958 1958 1958 1958 ...
 $ ryear3  : int  1966 1966 1966 1966 1966 1966 1966 1966 1966 1966 ...
 $ location: int  1 1 1 1 1 1 1 1 1 1 ...
 $ girth   : logi  NA NA NA NA NA NA ...
 $ lgirth  : logi  NA NA NA NA NA NA ...
```

- Ensuite, visualisez les données comme dans le labo pour les tests de t. Créez un histogramme avec ligne de densité et un Box plot par année. Que vous révèlent ces données?

```
mygraph <- ggplot(Dam10dat, aes(x = fklngth)) +
  labs(x = "Fork length (cm)") +
  geom_density() +
  geom_rug() +
  geom_histogram(aes(y = ..density..),
    color = "black",
    alpha = 0.3
  ) +
  stat_function(
    fun = dnorm,
    args = list(
```

```
      mean = mean(Dam10dat$fklngth),
      sd = sd(Dam10dat$fklngth)
    ),
    color = "red"
  )

# display graph, by year
mygraph + facet_wrap(~year, ncol = 2)
```

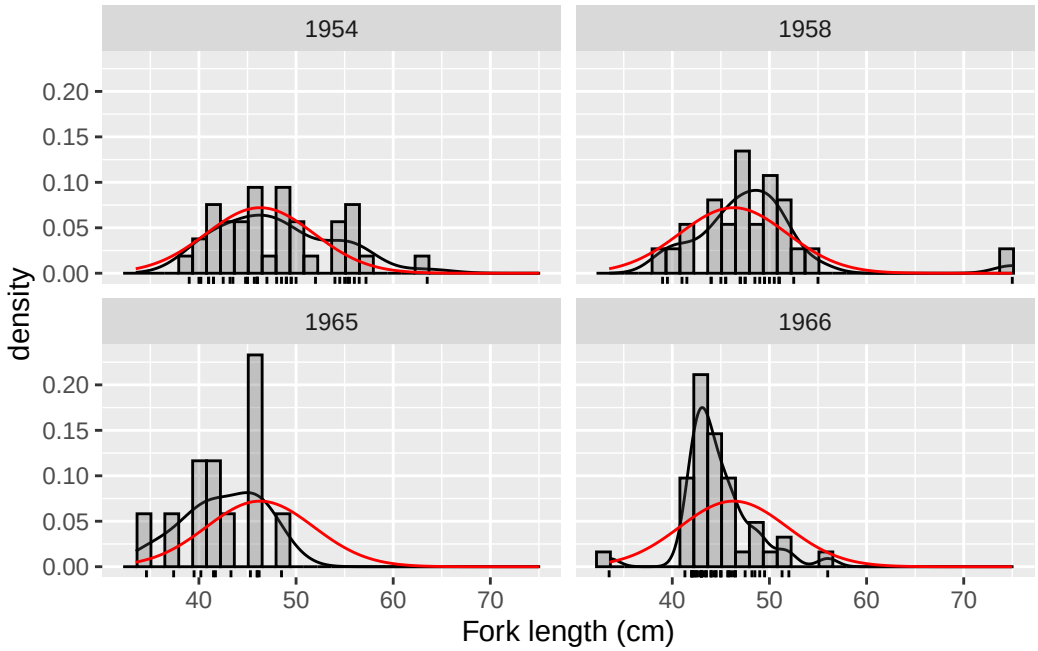


Figure 5.1.: Distribution de la longueur des esturgeons par année

```
boxplot(fklngth ~ year, data = Dam10dat)
```

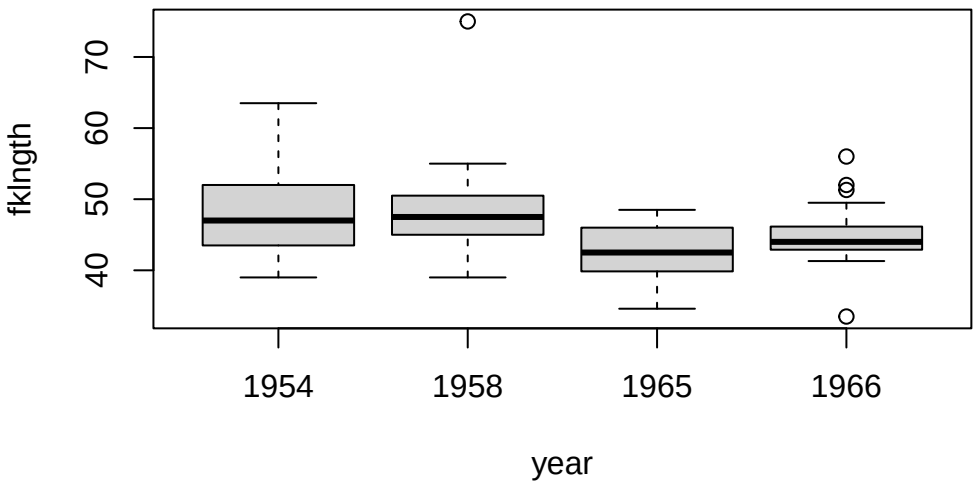


Figure 5.2.: Boxplot de la longueur pas annéee

Il semble que la taille des esturgeons est un peu plus petite après la construction du barrage, mais les données sont très variables et les effets ne sont pas parfaitement clairs. Il y a peut-être des problèmes de normalité avec les échantillons de 1954 et 1966, et il y a probablement des valeurs extrêmes dans les échantillons de 1958 et 1966. On va continuer en testant les conditions d'application de l'ANOVA. Il faut d'abord faire l'analyse et examiner les résidus.

5.2.2. Vérifier les conditions d'application de l'ANOVA paramétrique

L'ANOVA paramétrique a trois conditions principales d'application :

1. les résidus sont normalement distribués,
2. la variance des résidus est égale dans tous les traitements (homoscédasticité) et
3. les résidus sont indépendants les uns des autres.

Ces conditions doivent être remplies avant qu'on puisse se fier aux résultats de l'ANOVA paramétrique.

- Faites une ANOVA à un critère de classification sur fklngth par année et produisez les graphiques diagnostiques

```
# Fit anova model and plot residual diagnostics
anova.model1 <- lm(fklngth ~ year, data = Dam10dat)
par(mfrow = c(2, 2))
plot(anova.model1)
```

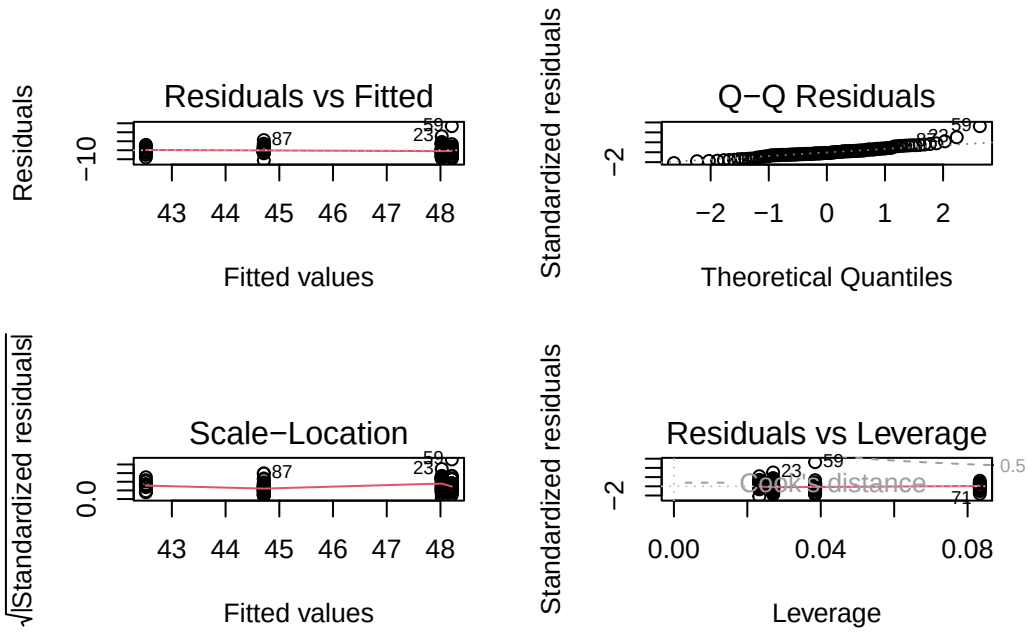


Figure 5.3.: Conditions d'applications de l'ANOVA

> **Warning**
>
> Faire attention dans le cadre d'une ANOVA à ce que la variable indépendante soit bien un facteur **factor**. Si la variable indépendante est reconnu comme du texte **character** alors vous n'obtiendrez que 3 graphiques et un message d'erreur du type:
> 'hat values (leverages) are all = 0.1
> and there are no factor predictors; no plot no. 5'

D'après les graphiques, on peut douter de la normalité et de l'homogénéité des variances. Notez qu'il y a un point qui ressort vraiment avec une forte valeur résiduelle (cas numéro 59) et qu'il ne s'aligne pas bien avec les autres valeurs: c'est la valeur extrême qui avait été détectée plus tôt. Ce point fera sans doute gonfler la variance résiduelle du groupe auquel il appartient.

Des tests formels nous confirmeront ou infirmeront nos conclusions faites à partir de ces graphiques.

- Faites un test de normalité sur les résidus de l'ANOVA.

```
shapiro.test(residuals(anova.model1))
```

```
    Shapiro-Wilk normality test

data:  residuals(anova.model1)
W = 0.91571, p-value = 1.63e-06
```

Ce test confirme nos soupçons: les résidus ne sont pas distribués normalement. Il faut cependant garder à l'esprit que la puissance est grande et que même de petites déviations de la normalité sont suffisantes pour rejeter l'hypothèse nulle.

- Ensuite, éprouvez l'hypothèse d'égalité des variances (homoscedasticité):

```
leveneTest(fklngth ~ year, data = Dam10dat)
```

```
Levene's Test for Homogeneity of Variance (center = median)
       Df F value  Pr(>F)
group   3  2.8159 0.04234 *
      114
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La valeur de p vous dit que vous pouvez rejeter l'hypothèse nulle qu'il n'y a aucune différence dans les variances entre les années. Alors, nous concluons que les variances ne sont pas homogènes.

5.2.3. Faire l'ANOVA

- Faites une ANOVA de fklnght en choisissant / en présumant pour l'instant que les conditions d'application sont suffisamment remplies. Que concluez-vous?

```
summary(anova.model1)
```

```
Call:
lm(formula = fklngth ~ year, data = Dam10dat)

Residuals:
     Min       1Q   Median       3Q      Max
-11.2116  -2.6866  -0.7116   2.2103  26.7885

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  48.0243     0.8566  56.061  < 2e-16 ***
year1958      0.1872     1.3335   0.140  0.88859
year1965     -5.5077     1.7310  -3.182  0.00189 **
year1966     -3.3127     1.1684  -2.835  0.00542 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.211 on 114 degrees of freedom
Multiple R-squared:  0.1355,    Adjusted R-squared:  0.1128
F-statistic: 5.957 on 3 and 114 DF,  p-value: 0.0008246
```

- *Coefficients: Estimates* Les 4 coefficients peuvent être utilisés pour obtenir les valeurs prédites par le modèle (i.e. les moyennes de chaque groupe). La fklngth moyenne de la première année (1954) est 48.0243. Les coefficients pour les 3 autres années sont la différence entre la moyenne de l'année en question et la moyenne de 1954. La moyenne pour 1965 est 48.0243-5.5077=42.5166. Pour chaque coefficient, on a également accès à l'erreur-type, une valeur de t et la probabilité qui lui est associée (H0 que le coefficient est 0). Les poissons étaient plus petits après la construction du barrage qu'en 1954. Vous devez prendre ces p-valeurs avec un grain de sel, car elles ne sont pas corrigées pour les comparaisons multiples et. En général, je porte peu d'attention à cette partie des résultats imprimés et me concentre sur ce qui suit.
- *Residual standard error*: La racine carrée de la variance des résidus (valeurs observées moins valeurs prédites) qui correspond à la variabilité inexpliquée par le modèle (variation de la taille des poissons capturés la même année).
- *Mutiple R-squared* Le R-carré est la proportion de la variabilité de la variable dépendante qui peut être expliquée par le modèle. Ici, le modèle explique 13.5% de la variabilité. Les différences de taille d'une année à l'autre sont relativement petites lorsqu'on les compare à la variation de taille entre les poissons capturés la même année.

4. *F-Statistic* La p-valeur associée au test "omnibus" que toutes les moyennes sont égales. Ici, p est beaucoup plus petit que 0.05 et on rejetterait H0 pour conclure que fklngth varie selon les années.

La commande `anova()` produit le tableau d'ANOVA standard qui contient la plupart de cette information:

```
anova(anova.model1)
```

```
Analysis of Variance Table

Response: fklngth
           Df  Sum Sq Mean Sq F value    Pr(>F)
year        3  485.26 161.755  5.9574 0.0008246 ***
Residuals 114 3095.30  27.152
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La variabilité totale de `fklngth`, mesurée par la somme des carrés des écarts (Sum sq) est partitionnée en ce qui peut être expliqué par l'année (485.26) et la variabilité résiduelle inexpliquée (3095.30). L'année explique bien (485.26/(3095.30+485.26)=.1355 or 13.55% de la variabilité). Le carré moyen des résidus (Residual Mean Sq) est leur variance.

5.2.4. Les comparaisons multiples

- La fonction `pairwise.t.test()` peut être utilisée pour comparer des moyennes et ajuster (ou non, si désiré) les probabilités pour le nombre de comparaisons en utilisant l'une des options pour p.adj:

Compare toutes les moyennes sans ajuster les probabilités

```
pairwise.t.test(Dam10dat$fklngth, Dam10dat$year,
  p.adj = "none"
)
```

```
    Pairwise comparisons using t tests with pooled SD

data:  Dam10dat$fklngth and Dam10dat$year

     1954   1958   1965
1958 0.8886 -      -
1965 0.0019 0.0022 -
1966 0.0054 0.0079 0.1996

P value adjustment method: none
```

Option `bonf` ajuste les p-valeurs avec la correction de Bonferroni. Ici, il y a 6 valeurs de p calculées, et la correction de Bonferroni revient à simplement multiplier la p-valeur par 6 (sauf si le résultat est supérieur à 1. Si tel est le cas, la `p-value` ajustée est 1).

```
pairwise.t.test(Dam10dat$fklngth, Dam10dat$year,
  p.adj = "bonf"
)
```

```
    Pairwise comparisons using t tests with pooled SD

data:  Dam10dat$fklngth and Dam10dat$year

     1954  1958  1965
1958 1.000 -     -
1965 0.011 0.013 -
1966 0.033 0.047 1.000

P value adjustment method: bonferroni
```

Option “holm” is est la correction séquentielle de Bonferroni dans laquelle les p-valeurs sont ordonnées de (i=1) la plus faible à (N) la plus grande. La correction pour les p-valeurs est (N-i+1). Ici, il y a N=6 paires de moyennes qui sont comparées. La plus petite valeur de p non corrigée est 0.0019 pour 1954 vs 1965. La p-valeur corrigée est donc $0.0019 * (6 - 1 + 1) = 0.011$. La seconde plus petite p-valeur est 0.0022. Sa p-valeur corrigée est 0.0022*(6-2+1)=0.011. Pour la p-valeur la plus élevée, la correction est (N-N+1)=1, donc la p-valeur corrigée est égale à la p-valeur brute.

```
pairwise.t.test(Dam10dat$fklngth, Dam10dat$year,
  p.adj = "holm"
)
```

```
    Pairwise comparisons using t tests with pooled SD

data:  Dam10dat$fklngth and Dam10dat$year

     1954  1958  1965
1958 0.889 -     -
1965 0.011 0.011 -
1966 0.022 0.024 0.399

P value adjustment method: holm
```

L'option "fdr" sert à contrôler le "false discovery rate".

```
pairwise.t.test(Dam10dat$fklngth, Dam10dat$year,
  p.adj = "fdr"
)
```

```
    Pairwise comparisons using t tests with pooled SD

data:  Dam10dat$fklngth and Dam10dat$year

     1954   1958   1965
1958 0.8886 -      -
1965 0.0066 0.0066 -
1966 0.0108 0.0119 0.2395

P value adjustment method: fdr
```

Les quatre méthodes mènent ici à la même conclusion: les poissons sont plus gros après la construction du barrage et toutes les comparaisons entre les années 50 et 60 sont significatives alors que les différences entre 54 et 58 ou 65 et 66 ne le sont pas. La conclusion ne dépend pas du choix de méthode.

Dans d'autres situations, vous pourriez obtenir des résultats contradictoires. Alors, quelle méthode choisir? Les p-valeurs qui ne sont pas corrigées sont certainement suspectes lorsqu'il y a plusieurs comparaisons. D'un autre coté, la correction de Bonferroni est conservatrice et le devient encore plus lorsqu'il y a de très nombreuses comparaisons. Des travaux récents suggèrent que la correction fdr est un bon compromis lorsqu'il y a beaucoup de comparaisons.

La méthode de Tukey est l'une des plus populaires et est facile à utiliser en R (notez cependant qu'il y a un petit bug qui se manifeste quand la variable indépendante peut ressembler à un nombre plutôt qu'un facteur, ce qui explique la petite pirouette avec `paste0` dans le code):

```
Dam10dat$myyear <- as.factor(paste0("m", Dam10dat$year))
TukeyHSD(aov(fklngth ~ myyear, data = Dam10dat))
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = fklngth ~ myyear, data = Dam10dat)

$myyear
                   diff        lwr        upr     p adj
m1958-m1954  0.1872141  -3.289570  3.6639986 0.9990071
m1965-m1954 -5.5076577 -10.021034 -0.9942809 0.0100528
m1966-m1954 -3.3126964  -6.359223 -0.2661701 0.0274077
m1965-m1958 -5.6948718 -10.436304 -0.9534397 0.0116943
m1966-m1958 -3.4999106  -6.875104 -0.1247171 0.0390011
m1966-m1965  2.1949612  -2.240630  6.6305526 0.5710111
```

```
plot(TukeyHSD(aov(fklngth ~ myyear, data = Dam10dat)))
```

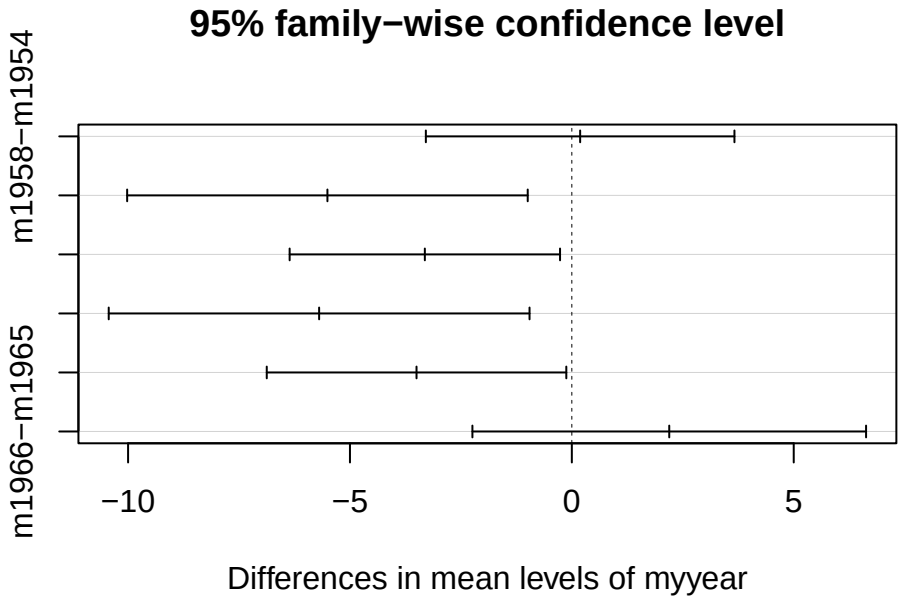


Figure 5.4.: Différence anuelles dans la longueur des esturgeons

Les intervalles de confiance, corrigés pour les comparaisons multiples par la méthode de Tukey, sont illustrés pour les différences entre années. Malheureusement les légendes ne sont pas complètes, mais l'ordre est le même que dans le tableau précédent.

Le package multcomp peut produire de meilleurs graphiques, mais requiert un peu plus de code:

```
# Alternative way to compute Tukey multiple comparisons
# set up a one-way ANOVA
```

```
anova.fkl.vs.year <- aov(aov(fklngth ~ myyear, data = Dam10dat))
# set up all-pairs comparisons for factor `year'

meandiff <- glht(anova.fkl.vs.year, linfct = mcp(
  myyear =
    "Tukey"
))
confint(meandiff)
```

```
     Simultaneous Confidence Intervals

Multiple Comparisons of Means: Tukey Contrasts


Fit: aov(formula = aov(fklngth ~ myyear, data = Dam10dat))

Quantile = 2.5937
95% family-wise confidence level


Linear Hypotheses:
                   Estimate lwr      upr
m1958 - m1954 == 0   0.1872  -3.2714   3.6458
m1965 - m1954 == 0  -5.5077  -9.9974  -1.0179
m1966 - m1954 == 0  -3.3127  -6.3433  -0.2821
m1965 - m1958 == 0  -5.6949 -10.4115  -0.9783
m1966 - m1958 == 0  -3.4999  -6.8574  -0.1424
m1966 - m1965 == 0   2.1950  -2.2174   6.6073
```

```
plot(meandiff)
```

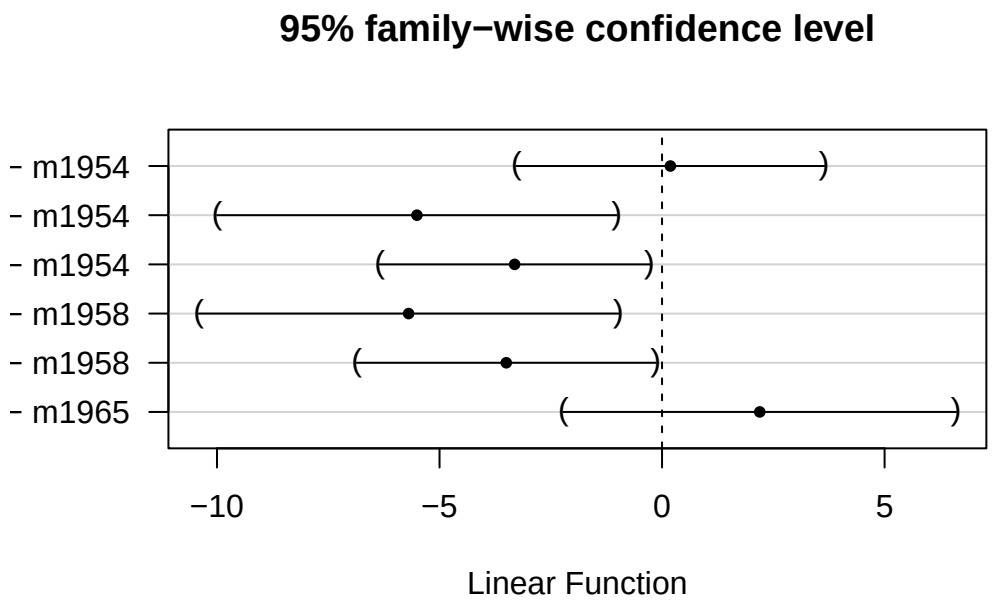


Figure 5.5.: Différence anuelles dans la longueur des esturgeons

C'est un peu mieux, mais ce qui le serait encore plus c'est un graphique des moyennes, avec leurs intervalles de confiance ajustés pour les comparaisons multiples:

```
# Compute and plot means and Tukey CI
means <- glht(
  anova.fkl.vs.year,
  linfct = mcp(myyear = "Tukey")
)
cimeans <- cld(means)
# use sufficiently large upper margin
# plot
old.par <- par(mai = c(1, 1, 1.25, 1))
plot(cimeans)
```

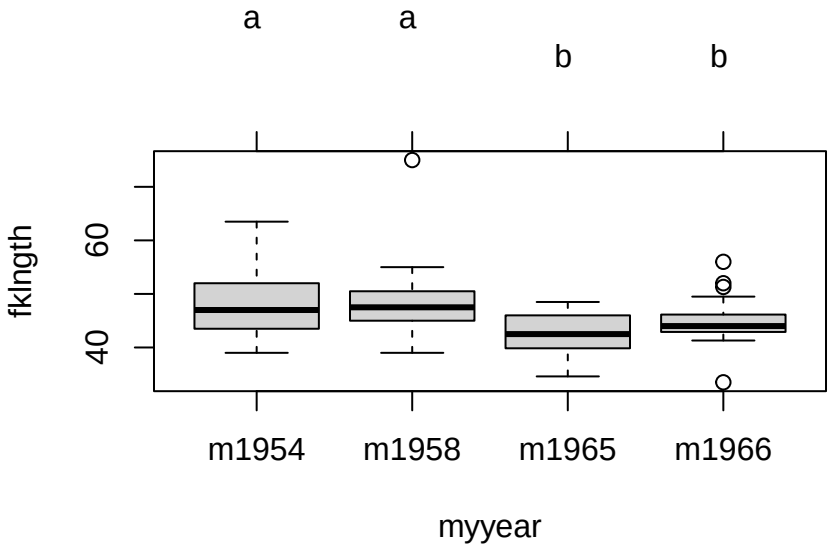


Figure 5.6.: Différence anuelles dans la longueur des esturgeons

Notez les lettres au dessus du graphique: les années étiquetées avec la même lettre ne diffèrent pas significativement l'une de l'autre.

5.3. Transformations de données et ANOVA non-paramétrique

Dans l'exemple précédent sur les différences annuelles de la variable fklgnth, on a noté que les conditions d'application de l'ANOVA n'étaient pas remplies. Si les données ne remplissent pas les conditions de l'ANOVA paramétrique, il y a 3 options :

1. Ne rien faire. Si les effectifs dans chaque groupe sont grands, on peut relaxer les conditions d'application car l'ANOVA est alors assez robuste aux violations de normalité (mais moins aux violations d'homoscedasticité),
2. on peut transformer les données
3. on peut faire une analyse non-paramétrique.

- Refaites l'ANOVA de la section précédente après avoir transformé en faisant le logarithme à la base de 10. Avec les données transformées, est-ce que les problèmes qui avaient été identifiés disparaissent ?

```
# Fit anova model on log10 of fklngth and plot residual diagnostics
par(mfrow = c(2, 2))
anova.model2 <- lm(log10(fklngth) ~ year, data = Dam10dat)
plot(anova.model2)
```

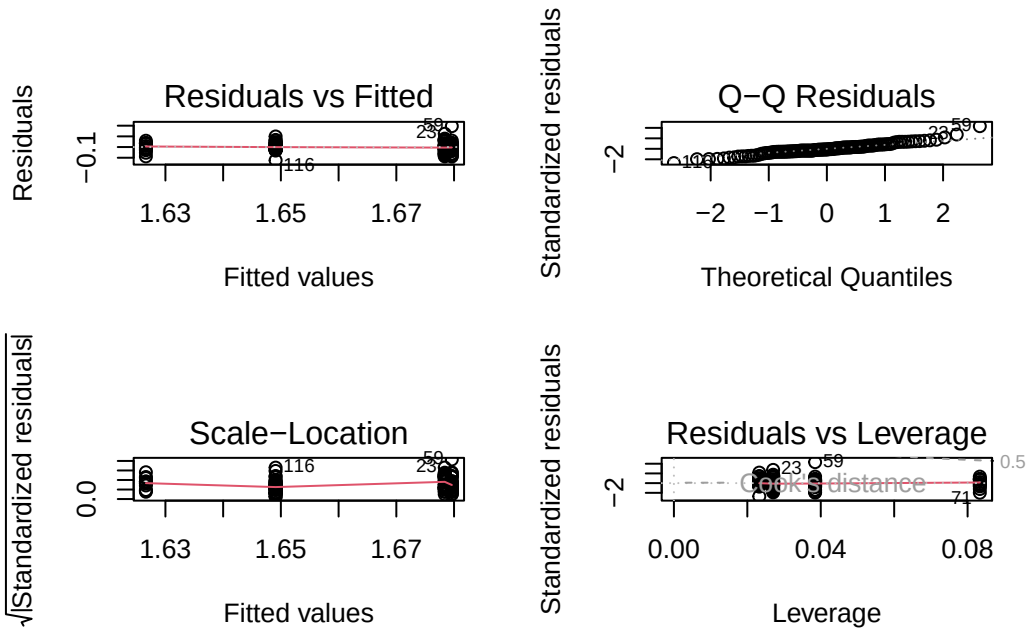


Figure 5.7.: Conditions d'application de l'ANOVA

Les graphiques diagnostiques des résidus donnent:

Les graphiques sont à peine mieux ici. Si on fait le test Wilk-Shapiro sur les résidus, on obtient:

```
shapiro.test(residuals(anova.model2))
```

```
    Shapiro-Wilk normality test

data:  residuals(anova.model2)
W = 0.96199, p-value = 0.002048
```

Alors, on a toujours des problèmes avec la normalité et on est juste sur le seuil de décision pour l'égalité des variances. Vous avez le choix à ce point:

1. essayer de trouver une autre transformation pour mieux rencontrer les conditions d'application
2. assumer que les données sont rencontrent suffisamment les conditions d'application
3. faire une ANOVA non-paramétrique.

- L'analogue non-paramétrique de l'ANOVA à un critère de classification le plus employé est le test de Kruskall-Wallis. Faites ce test sur `fklngth` et comparez les résultats à ceux de l'analyse paramétrique. Que concluez-vous?

```
kruskal.test(fklngth ~ year, data = Dam10dat)
```

```
    Kruskal-Wallis rank sum test

data:  fklngth by year
Kruskal-Wallis chi-squared = 15.731, df = 3, p-value = 0.001288
```

La conclusion est donc la même qu'avec l'ANOVA paramétrique: on rejette l'hypothèse nulle que le rang moyen est le même pour chaque année. Donc, même si les conditions d'application de l'analyse paramétrique n'étaient pas parfaitement rencontrées, les conclusions sont les mêmes, ce qui illustre la robustesse de l'ANOVA paramétrique.

5.4. Examen des valeurs extrêmes

Vous devriez avoir remarqué au cours des analyses précédentes qu'il y avait peut-être des valeurs extrêmes dans les données. Ces points étaient évidents dans le Box Plot de fklngth by year et ont été notés comme les points 59, 23, et 87 dans les diagrammes de probabilité des résidus et dans le diagramme de dispersion des résidus et des valeurs estimées. En général, vous devez avoir de très bonnes raisons pour enlever des valeurs extrêmes de la base de données (i.e. vous savez qu'il y a eu une erreur avec un cas). Cependant, il est quand même toujours valable de voir comment l'analyse change en enlevant des valeurs extrêmes de la base de données.

- Répétez l'ANOVA originale sur fklngth et year mais faites le avec un sous-ensemble de données sans les valeurs extrêmes. Est-ce que les conclusions ont changé?

```
Damsubset <- Dam10dat[-c(23, 59, 87), ] # removes obs 23, 59 and 87
aov.Damsubset <- aov(fklngth ~ as.factor(year), Damsubset)
summary(aov.Damsubset)
```

```
                 Df Sum Sq Mean Sq F value   Pr(>F)
as.factor(year)   3  367.5  122.50   6.894 0.000267 ***
Residuals       111 1972.4   17.77
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
shapiro.test(residuals(aov.Damsubset))
```

```
    Shapiro-Wilk normality test

data:  residuals(aov.Damsubset)
W = 0.98533, p-value = 0.2448
```

```
leveneTest(fklngth ~ year, Damsubset)
```

```
Levene's Test for Homogeneity of Variance (center = median)
       Df F value   Pr(>F)
group   3  4.6237 0.004367 **
      111
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

L'élimination de trois valeurs extrêmes améliore un peu les choses, mais ce n'est pas parfait. On a toujours une problème avec les variances, mais les résidus sont maintenant normaux. Cependant, le fait que la conclusion qu'on tire de l'ANOVA originale ne change pas en enlevant les points renforce le fait qu'on n'a pas une bonne raison pour enlever les points.

5.5. Test de permutation

Commande R pour un test de permutation d'une ANOVA à un critère de classification.

```
############################################################
# Permutation Test for one-way ANOVA
# modified from code written by David C. Howell
# http://www.uvm.edu/~dhowell/StatPages/More_Stuff/Permutation%20Anova/PermTestsAnova.html
# set desired number of permutations
nreps <- 500
# to simplify reuse of this code, copy desired dataframe to mydata
mydata <- Dam10dat
# copy model formula to myformula
myformula <- as.formula("fklngth ~ year")
# copy dependent variable vector to mydep
mydep <- mydata$fklngth
# copy independent variable vector to myindep
myindep <- as.factor(mydata$year)
################################################
# You should not need to modify code chunk below
################################################
# Compute observed F value for original sample
mod1 <- lm(myformula, data = mydata) # Standard Anova
ANOVA <- summary(aov(mod1)) # Save summary to variable
observedF <- ANOVA[[1]]$"F value"[1] # Save observed F value
# Print standard ANOVA results
cat(
  " The standard ANOVA for these data follows ",
  "\n"
)
```

```
print(ANOVA, "\n")
cat("\n")
cat("\n")
print("Resampling as in Manly with unrestricted sampling of observations. ")

# Now start resampling
Fboot <- numeric(nreps) # initalize vector to receive permuted
values
Fboot[1] <- observedF
for (i in 2:nreps) {
  newdependent <- sample(mydep, length(mydep)) # randomize dep
  var
  mod2 <- lm(newdependent ~ myindep) # refit model
  b <- summary(aov(mod2))
  Fboot[i] <- b[[1]]$"F value"[1] # store F stats
}
permprob <- length(Fboot[Fboot >= observedF]) / nreps
cat(
  " The permutation probability value is: ", permprob,
  "\n"
)
# end of code chunk for permutation
```

Version lmPerm du test de permutation.

```
## lmPerm version of permutation test
library(lmPerm)
# for generality, copy desired dataframe to mydata
# and model formula to myformula
mydata <- Dam10dat
myformula <- as.formula("fklngth ~ year")
# Fit desired model on the desired dataframe
mymodel <- lm(myformula, data = mydata)
# Calculate permutation p-value
anova(lmp(myformula, data = mydata, perm = "Prob", center = FALSE, Ca = 0.001))
```

6. ANOVA à critères multiples : plans factoriels et hiérarchiques

Après avoir complété cet exercice de laboratoire, vous devriez être capable de:

- Utiliser R pour faire une ANOVA paramétrique d'un plan factoriel avec deux facteurs de classification et réplication
- Utiliser R pour faire une ANOVA paramétrique d'un plan factoriel avec deux facteurs de classification sans réplication
- Utiliser R pour faire une ANOVA paramétrique d'un plan hiérarchique avec réplication
- Utiliser R pour faire une ANOVA non paramétrique avec deux facteurs de classification
- Utiliser R pour faire des comparaisons multiples

Il existe une très grande variété de plans (designs) d'ANOVA que R peut analyser. Cet exercice n'est qu'une introduction aux plans les plus communs.

6.1. Paquets et données requises pour le labo

Ce laboratoire nécessite:

- les paquets R:
  - multicomp
  - car
  - tidyverse
- les fichiers de données
  - Stu2wdat.csv
  - Stu2mdat.csv
  - nr2wdat.csv
  - nestdat.csv
  - wmcdat2.csv
  - wmc2dat2.csv

```
library(multcomp)
library(car)
library(tidyverse)
```

6.2. Plan factoriel à deux facteurs de classification et réplication

Il est fréquent de vouloir analyser l'effet de plusieurs facteurs simultanément. L'ANOVA factorielle à deux critères de classification permet d'examiner deux facteurs à la fois, mais la même approche peut être utilisée pour 3, 4 ou même 5 facteurs quoique l'interprétation des résultats devienne beaucoup plus complexe.

Supposons que vous êtes intéressés par l'effet de deux facteurs : site (location, Cumberland House ou The Pas) et sexe (sex, mâle ou femelle) sur la taille des esturgeons. Comme l'effectif n'est pas le même pour tous les groupes, c'est un plan qui n'est pas balancé. Notez aussi qu'il y a des valeurs manquantes pour certaines variables, ce qui veut dire que chaque mesure n'a pas été effectuée sur chaque poisson.

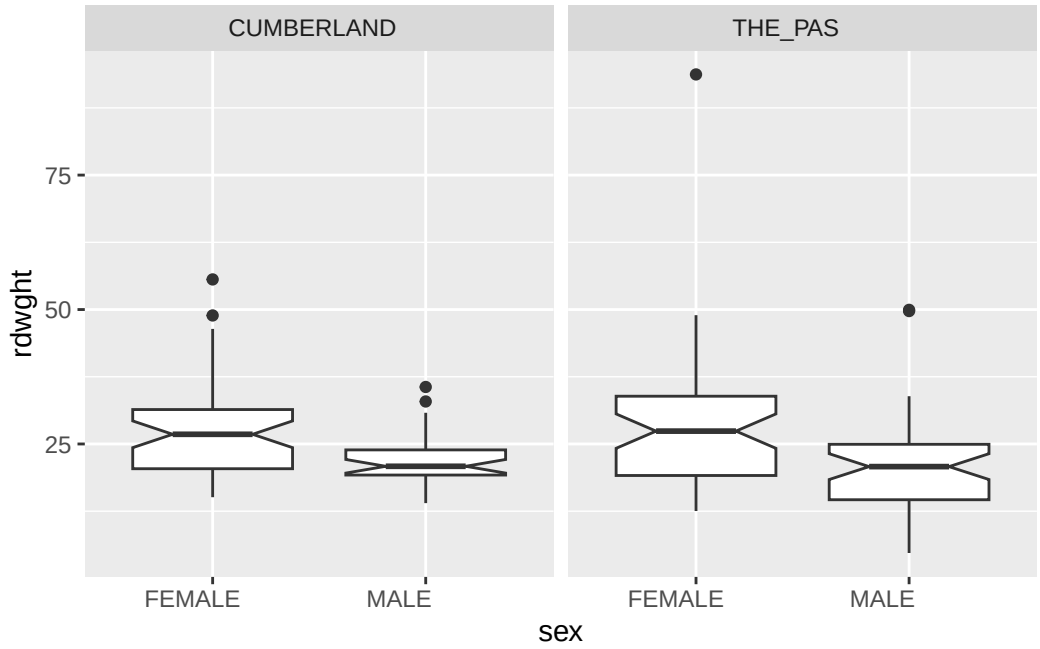
6.2.1. ANOVA à effets fixes

- Examinez d'abord les données en faisant des box plots de rdwght pour sex et location des données du fichier `Stu2wdat.csv` .

Solution

```
Stu2wdat <- read.csv("data/Stu2wdat.csv")

ggplot(Stu2wdat, aes(x = sex, y = rdwght)) +
geom_boxplot(notch = TRUE) +
facet_grid(~location)
```

```
Warning: Removed 4 rows containing non-finite values (`stat_boxplot()`).
```



Les graphiques montrent qu'aux deux sites les femelles sont probablement plus grandes que les mâles, mais que les tailles ne varient pas beaucoup d'un site à l'autre. La présence de valeurs extrêmes sur ces graphiques suggère qu'il y aura peut-être des problèmes avec la condition de normalité des résidus.

- Générez les statistiques sommaires pour RDWGHT par sex et Location.

```
Stu2wdat <- read.csv("data/Stu2wdat.csv")
aggregate(rdwght ~ sex + location, data = Stu2wdat, FUN = "summary")
```

```
           sex     location rdwght.Min. rdwght.1st Qu. rdwght.Median
1 FEMALE       CUMBERLAND      15.10000       20.40000      26.80000
2 MALE         CUMBERLAND      14.00000       19.22500      20.85000
3 FEMALE       THE_PAS         12.54000       19.14000      27.39000
4 MALE         THE_PAS          4.73000       14.63000      20.79000
  rdwght.Mean rdwght.3rd Qu. rdwght.Max.
1    27.37347       31.40000    55.60000
2    22.14118       23.90000    35.60000
3    27.97717       33.88000    93.72000
4    20.64652       24.94250    49.94000
```

Ces résultats supportent l'interprétation des box plots: Les femelles sont plus grosses que les mâles, et la différence de taille entre les deuxsites sont petites.

- À l'aide du fichier Stu2wdat.csv faites une ANOVA factorielle à deux critères de classification:

```
# Fit anova model and plot residual diagnostics
# but first, save current par and set graphic page to hold 4 graphs
opar <- par(mfrow = c(2, 2))
anova.model1 <- lm(rdwght ~ sex + location + sex:location,
  contrasts = list(sex = contr.sum, location = contr.sum),
  data = Stu2wdat
)
anova(anova.model1)
```

```
Analysis of Variance Table

Response: rdwght
              Df  Sum Sq Mean Sq F value    Pr(>F)
sex            1  1839.6 1839.55 18.6785 2.569e-05 ***
location       1     4.3    4.26  0.0433    0.8355
sex:location   1    48.7   48.69  0.4944    0.4829
Residuals    178 17530.4   98.49
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

> **Warning**
>
> Attention, R imprime les sommes des carrés séquentielles (Type I) les carrés moyens et probabilités associés. Vous ne pouvez pas vous y fier si votre plan d'expérience n'est pas parfaitement balancé. Dans cet exemple, le nombre de poissons capturés change selon le site et le sexe et le plan d'expérience n'est donc pas balancé.

Vous devez extraire les sommes de carrés partielles (Type III). Le moyen le plus simple que j'ai trouvé est d'utiliser la fonction Anova()du package `car` (notez la différence subtile, Anova() n'est pas la même chose que anova(), R est impitoyable et distingue les majuscules des minuscules). Malheureusement, Anova() ne suffit pas; il faut également spécifier le type de contraste dans le modèle avecl'argument `contrasts = list(sex = contr.sum,location = contr.sum)`

```
library(car)
Anova(anova.model1, type = 3)
```

```
Anova Table (Type III tests)

Response: rdwght
             Sum Sq  Df   F value    Pr(>F)
(Intercept)  106507   1 1081.4552 < 2.2e-16 ***
sex            1745   1   17.7220 4.051e-05 ***
location          9   1    0.0891    0.7656
sex:location     49   1    0.4944    0.4829
Residuals     17530 178
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Suite à l'ANOVA, on accepte deux hypothèses nulles: (1) que l'effet du sexe ne varie pas entre les sites (pas d'interaction significative) et (2) qu'il n'y a pas de différence de taille des esturgeons (peu importe le sexe) entre les deux sites. D'un autre coté, on rejette l'hypothèse nulle qu'il n'y a pas de différence de taille entre les esturgeons mâles et les femelles, tel que suggéré par les graphiques.

```
par(mfrow = c(2, 2))
plot(anova.model1)
```

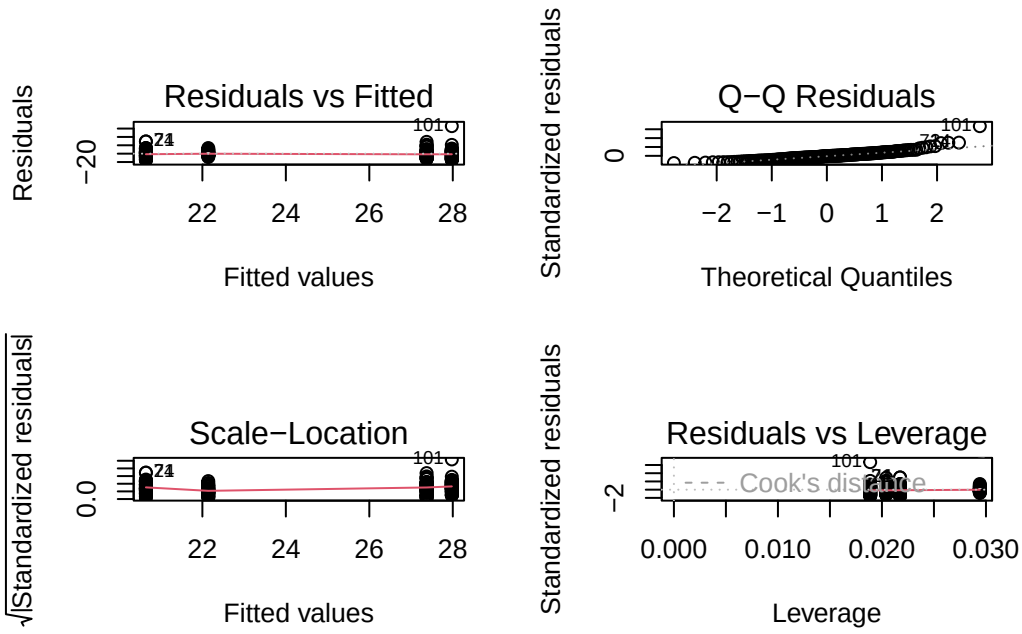


Figure 6.1.: Conditions d'application ANOVA model1

Cependant, on ne peut se fier à ces résultats sans vérifier si les conditions d'application de l'ANOVA étaient remplies. Un examen des graphiques des résidus, en haut, montre que les résidus semblent être distribués plus ou moins normalement, si ce n'est des 3 valeurs extrêmes qui sont notées sur le diagramme de probabilité (cas 101, 24,& 71). D'après le graphique des résidus vs les valeurs prédites, on voit que l'étendue des résidus est plus ou moins égale pour les valeurs estimées, sauf encore pour 2 ou 3 cas. Si on éprouve la normalité, on obtient:

```
shapiro.test(residuals(anova.model1))
```

```
    Shapiro-Wilk normality test

data:  residuals(anova.model1)
W = 0.87213, p-value = 2.619e-11
```

Alors, il y a évidence que les résidus ne sont pas distribués normalement.

Nous allons utiliser le test de Levene pour examiner l'homoscédasticité des résidus, de la même façon qu'on a fait pour l'ANOVA à un critère de classification.

```
library(car)
leveneTest(rdwght ~ sex * location, data = Stu2wdat)
```

```
Levene's Test for Homogeneity of Variance (center = median)
       Df F value  Pr(>F)
group   3  3.8526 0.01055 *
      178
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Si les résidus étaient homoscédastiques, on accepterait l'hypothèse nulle que le absres moyen ne varie pas entre les niveaux de sexe et location (i.e., sexloc). Le tableau d'ANOVA ci-dessus montre que l'hypothèse est rejetée. Il y a donc évidence d'hétéroscédasticité. En bref, nous avons donc plusieurs conditions d'application qui ne sont pas respectées. La question qui reste est: ces violations sont-elles suffisantes pour invalider nos conclusions ?

Exercice

Répétez la même analyse avec les données du fichier `stu2mdat.csv` . Que concluez-vous? Supposons que vous vouliez comparer la taille des mâles et des femelles. Comment cette comparaison diffère entre les deux ensembles de données ?

Solution

```r
Stu2mdat <- read.csv("data/Stu2mdat.csv")
anova.model2 <- lm(
  formula = rdwght ~ sex + location + sex:location,
  contrasts = list(sex = contr.sum, location = contr.sum),
  data = Stu2mdat
)
summary(anova.model2)
Anova(anova.model2, type = 3)
```

```
Call:
lm(formula = rdwght ~ sex + location + sex:location, data = Stu2mdat,
    contrasts = list(sex = contr.sum, location = contr.sum))

Residuals:
    Min      1Q  Median      3Q     Max
-15.917  -6.017  -0.580   4.445  65.743

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     24.5346     0.7461  32.885  < 2e-16 ***
sex1            -0.5246     0.7461  -0.703    0.483
location1        0.2227     0.7461   0.299    0.766
sex1:location1   3.1407     0.7461   4.210 4.05e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 9.924 on 178 degrees of freedom
  (4 observations deleted due to missingness)
Multiple R-squared:  0.09744,   Adjusted R-squared:  0.08223
F-statistic: 6.405 on 3 and 178 DF,  p-value: 0.0003817
```

Notez que cette fois les femelles sont plus grandes que les mâles à Cumberland House, mais que c'est le contraire à The Pas. Quel est le résultat de l'ANOVA (n'oubliez pas qu'il faut des Type III sums of squares pour les résultats)?

```
Anova Table (Type III tests)

Response: rdwght
             Sum Sq  Df   F value    Pr(>F)
(Intercept)  106507   1 1081.4552 < 2.2e-16 ***
sex             49   1    0.4944    0.4829
location         9   1    0.0891    0.7656
sex:location  1745   1   17.7220 4.051e-05 ***
Residuals    17530 178
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Dans ce cas, le terme de l'interaction (sex:location) est maintenant significatif mais les effets principaux ne le sont pas.

- Vous trouverez utile ici de créer des graphiques pour les deux fichiers de données pour comparer les interactions entre sex et location. Le graphique d'interaction montre les relations entre les moyennes de chaque combinaison de facteurs (appelées aussi les moyennes des ce$lules).Générez un graphique illustrant les intéractions en utilisant la fonction allEffects du package effects :

```
library(effects)
allEffects(anova.model1)
```

```
 model: rdwght ~ sex + location + sex:location

 sex*location effect
             location
sex            CUMBERLAND   THE_PAS
  FEMALE          27.37347     27.97717
  MALE            22.14118     20.64652
```

```
plot(allEffects(anova.model1), "sex:location")
```

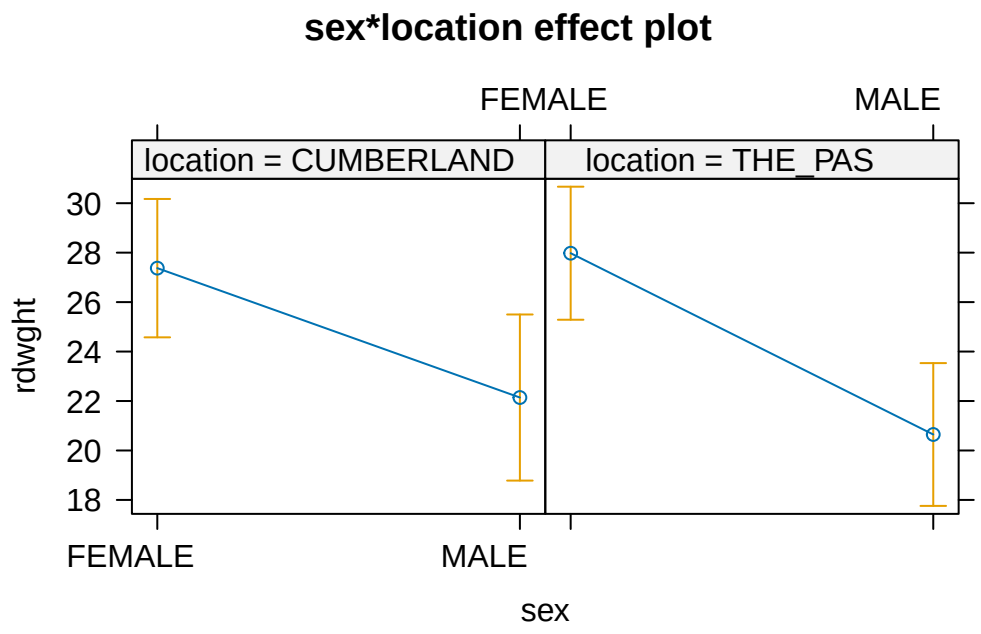


Figure 6.2.: Effet du sexe et du lieu sur le poids des esturgeons

```
allEffects(anova.model2)
```

```
 model: rdwght ~ sex + location + sex:location

 sex*location effect
              location
sex            CUMBERLAND   THE_PAS
  FEMALE           27.37347     20.64652
  MALE             22.14118     27.97717
```

```
plot(allEffects(anova.model2), "sex:location")
```

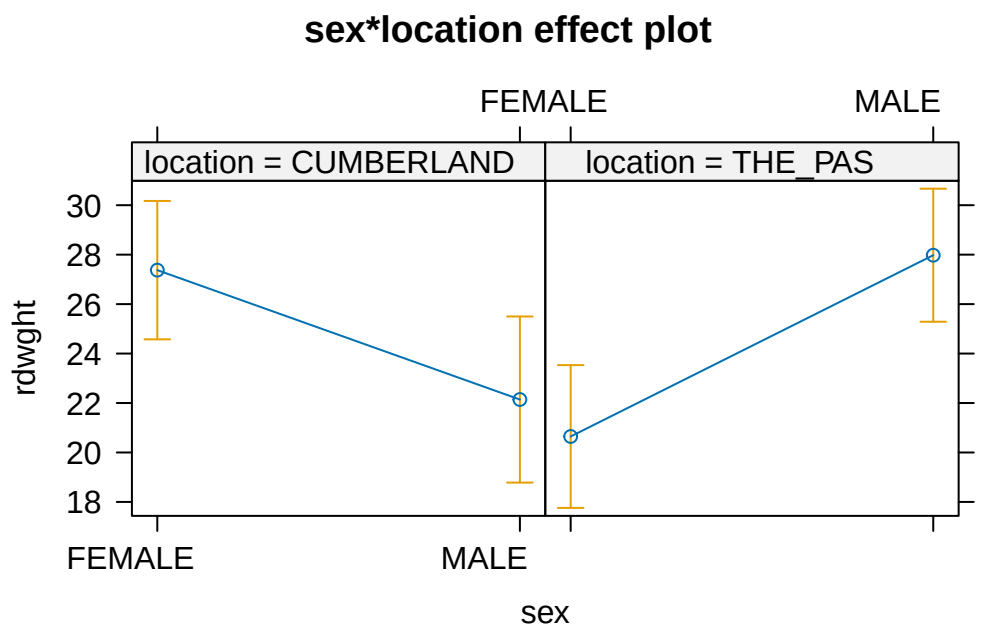


Figure 6.3.: Effet du sexe et du lieu sur le poids des esturgeons

Il y a une différence importante entre les résultats obtenus avec `stu2wdat` et `stu2mdat`. Dans le premier cas, puisqu'il n'y a pas d'interaction, on peut regrouper les données des deux niveaux d'un facteur (le site, par exemple) pour éprouver l'hypothèse d'un effet de l'autre facteur (le sexe). En fait, si on fait cela et que l'on calcule une ANOVA à un critère de classification (sex), on obtient:

```
Anova(aov(rdwght ~ sex, data = Stu2wdat), type = 3)
```

```
Anova Table (Type III tests)

Response: rdwght
            Sum Sq  Df F value    Pr(>F)
(Intercept)  78191   1 800.440 < 2.2e-16 ***
sex           1840   1  18.831 2.377e-05 ***
Residuals    17583 180
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notez que la somme des carrés des résidus (17583) est presque égale à celle du modèle complet (17530) de l'ANOVA factorielle à deux facteurs. C'est parce que dans cette anova factorielle, le terme d'interaction et le terme représentant l'effet du site n'expliquent qu'une partie infime de la variabilité. D'un autre coté, si on essaie le même truc avec `stu2mdat`, on obtient:

```
Anova(aov(rdwght ~ sex, data = Stu2mdat), type = 3)
```

```
Anova Table (Type III tests)

Response: rdwght
            Sum Sq  Df  F value Pr(>F)
(Intercept)  55251   1 515.0435 <2e-16 ***
sex            113   1   1.0571 0.3053
Residuals    19309 180
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Ici la somme des carrées des résidus (19309) est beaucoup plus grande que celle de l'ANOVA factorielle (175306) parce qu'une partie importante de la variabilité expliquée par le modèle est associée à l'interaction. Notez que si on n'avait fait que cette analyse, on conclurait que les esturgeons mâles et femelles ont la même taille. Mais en fait leur taille diffère; seulement la différence est à l'avantage des mâles à un site et à l'avantage des femelles à l'autre. Il est donc délicat d'interpréter l'effet principal (sexe) en présence d'une interaction significative…

6.2.2. ANOVA à effets mixtes

Les analyses qui précèdent négligent un point important: `location` pourrait être traité comme un facteur aléatoire et `sex` est fixe. Par conséquent le modèle approprié d'ANOVA est de type mixte.

Notez que dans toutes les analyses qui précèdent, R a traité cette ANOVA comme si elle etait de type effet fixe seulement, et les termes principaux et celui d'interaction ont été testés en utilisant le carré moyen des résidus comme dénominateur des tests de F. Cependant, pour une ANOVA de type mixte, ces effets devraient être testés en utilisant le carré moyen du terme d'interaction, ou en combinant la somme des carrés de l'erreur et de l'interaction (selon le statisticien consulté!).

En utilisant `Stu2wdat`, refaites un tableau d'ANOVA pour RDWGHT en considérant location comme facteur aléatoire et sex comme un facteur fixe. Pour ce faire, vous devrez recalculer les valeurs de F pour sex et location en utilisant le carré moyen de l'interaction sex:location au lieu du carré moyen des résidus comme dénominateur. Le mieux c'est de le faire à la mitaine ent travaillant avec les Type III Sums of squares du tableau d'ANOVA.

Solution

```
Anova(anova.model1, type = 3)
```

```
Anova Table (Type III tests)

Response: rdwght
             Sum Sq  Df    F value    Pr(>F)
(Intercept)  106507   1 1081.4552 < 2.2e-16 ***
sex            1745   1   17.7220 4.051e-05 ***
location          9   1    0.0891    0.7656
sex:location     49   1    0.4944    0.4829
Residuals     17530 178
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Pour `sex`, la nouvelle valeur de F (le rapport des carrés moyens) est de

$$F = \frac{(1745/1)}{(49/1)} = 35.6$$

Pour obtenir la valeur de p correspondant à cette statistique `F`, il faut utilisez la fonction de probabilité de la distribtuion de F `pf(F, df1, df2, lower.tail = FALSE)`, où F est la valeur de F calculée, et `df1` et `df2` sont les degrés de liberté du numérateur (sex) et dénominateur(SEX:location).

```
pf(35.6, 1, 1, lower.tail = FALSE)
```

```
[1] 0.1057152
```

Notez que maintenant la valeur de p pour sex n'est plus significative. C'est parce que le carré moyen de l'erreur dans l'ANOVA initiale est plus petit que celui associé à l'interaction, mais surtout parce que le nombre de degrés de liberté pour le dénominateur du test de F est passé de 178 à 1 seulement. En général, c'est beaucoup plus difficile d'obtenir des résultats significatifs quand les degrés de liberté pour le dénominateur sont petits.

> **Note**
>
> Les modèles mixtes qui sont une généralisation de l'ANOVA à effets mixtes sont maintenant extrêmement bien développé et sont à favoriser lors d'analyse incluant des effets dit aléatoires.

6.3. Plan factoriel à deux facteurs de classification sans réplication

Dans certains plans d'expérience il n'y a pas de réplicats pour chaque combinaison de facteurs, par exemple parce qu'il serait trop coûteux de faire plus d'une observation. L'ANOVA à deux critères de classification est quand même possible dans ces circonstances, mais il y a une limitation importante.

> **Warning**
>
> Comme il n'y a pas de réplicats, on ne peut estimer la variance du terme d'erreur. En effet on ne peut qu'estimer la somme des carrés associés à chacun des facteurs principaux, et la quantité de variabilité qui reste (Remainder Mean Square) représente la somme de la variabilité attribuable à l'interaction et au terme d'erreur. Cela a une implication importante. Dans le cas d'un modèle avec uniquement des effets fixes ou pour l'effet aléatoire d'un modèle d'ANOVA mixtes on ne peut tester les effets principaux que si on est sur qu'il n'y a pas d'interaction.

Un limnologiste qui étudie Round Lake dans le Parc Algonquin prend une seule mesure de température (temp,en degrés C) à 10 profondeurs différentes (depth, en m) à quatre dates (date) au cours de l'été. Ses données sont au fichier `nr2wdat.csv`.

- Effectuez une ANOVA à deux critères de classification en utilisant temp comme variable dépendante et date et depth comme variables indépendantes (vous devez changer le type de données pour DEPTH pour que R traite cette variable comme un facteur et non pas une var$able continue).

```
nr2wdat <- read.csv("data/nr2wdat.csv")
nr2wdat$depth <- as.factor(nr2wdat$depth)
anova.model4 <- lm(temp ~ date + depth, data = nr2wdat)
Anova(anova.model4, type = 3)
```

```
Anova Table (Type III tests)

Response: temp
             Sum Sq Df  F value    Pr(>F)
(Intercept) 1511.99  1 125.5652 1.170e-11 ***
date         591.15  3  16.3641 2.935e-06 ***
depth       1082.82  9   9.9916 1.450e-06 ***
Residuals    325.12 27
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Si on suppose que c'est un modèle d'ANOVA mixte (date aléatoire, Depth fixe), que concluez vous? (Indice: faites un graphique d'interaction température en fonction de la profondeur et la date, pour voir ce qui se passe).

```
interaction.plot(nr2wdat$depth, nr2wdat$date, nr2wdat$temp)
```

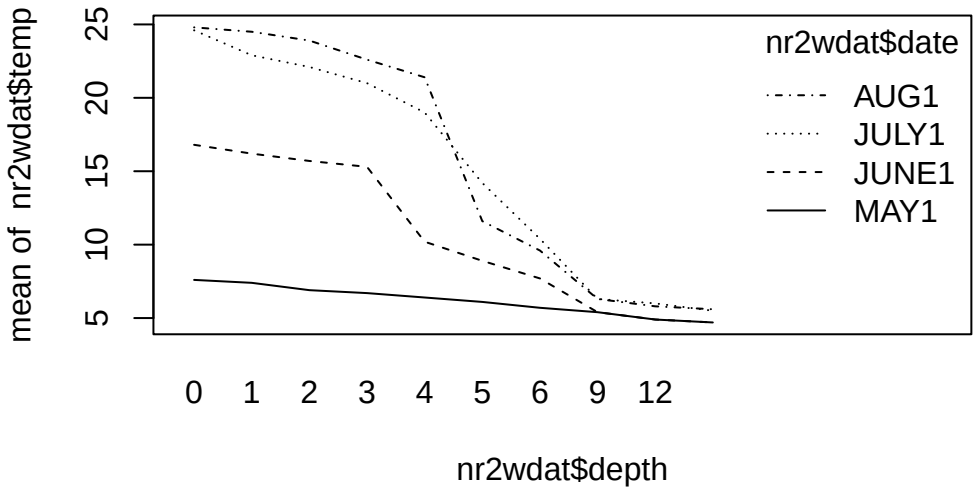


Figure 6.4.: Effet du mois et de la profondeur sur la température

La température diminue significativement en profondeur. Pour tester l'effet du mois (le facteur aléatoire), on doit présumer qu'il n'y a pas d'interaction entre la profondeur et le mois (donc que l'effet de la profondeur sur la température est le même à chaque mois). C'est peu probable: si vous faites un graphique de la température en fonction de la profondeur pour chaque mois, vous observerez que le profil de température change au fur et à mesure du développement de la thermocline. Bref, comme le profil change au cours de l'été, ce modèle ne fait pas de très bonnes prédictions.

Jetez un coup d'oeil sur les graphiques des résidus:

```
par(mfrow = c(2, 2))
plot(anova.model4)
```

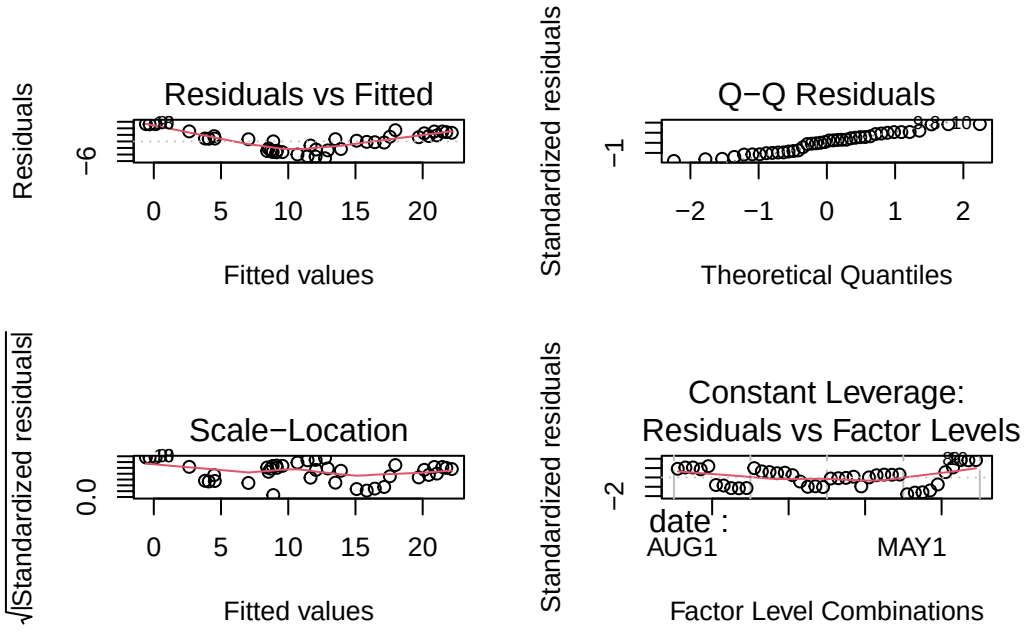


Figure 6.5.: Conditions d'applications du modèle anova.model4

```
shapiro.test(residuals(anova.model4))
```

```
    Shapiro-Wilk normality test

data:  residuals(anova.model4)
W = 0.95968, p-value = 0.1634
```

Le test de normalité sur les résidus donne p = 0.16, donc l'hypothèse de normalité ne semble pas être sérieusement en doute. Pour l'égalité des variances, on peut seulement comparer entre les mois en utilisant les profondeurs comme réplicats (ou l'inverse). En utilisant les profondeurs comme réplicats, on obtient:

```r
leveneTest(temp ~ date, data = nr2wdat)
```

```
Warning in leveneTest.default(y = y, group = group, ...): group coerced to
factor.
```

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value    Pr(>F)
group  3  17.979 2.679e-07 ***
      36
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Il y a donc un problème d'hétéroscédasticité, comme on peut très bien voir dans le graphique des résidus vs les valeurs estimées. Cette analyse n'est donc pas très satisfaisante: il y a des violations des conditions d'application et il semble y avoir une interaction entre depth et date qui pourrait invalider l'analyse.

6.4. Plans hiérarchiques

Un design expérimental fréquent implique la division de chaque groupe du facteur majeur en sous-groupes aléatoires. Par exemple, une généticienne intéressée par l'effet du génotype sur la résistance à la dessiccation chez la drosophile effectue une expérience. Pour chaque génotype (facteur principal) elle prépare trois chambres de croissance (sous-groupes) avec une température et humidité contrôlées. Dans chaque chambre de croissance, elle place cinq larves, puis mesure le nombre d'heures pendant lesquelles chaque larve survit. Les données ont donc un structure hiérarchique. Il ya des observations répétées dans chaque chambre au sein de chaque génotype.

- Le fichier `nestdat.csv` contient les résultats d'une expérience semblable. Il contient trois variables : genotype, chamber et survival. Effectuez une ANOVA hiérarchique avec survival comme variable dépendante et genotype et chamber/genotype comme variables indépendantes.

```r
nestdat <- read.csv("data/nestdat.csv")
nestdat$chamber <- as.factor(nestdat$chamber)
nestdat$genotype <- as.factor(nestdat$genotype)
anova.nested <- lm(survival ~ genotype / chamber, data = nestdat)
```

Que concluez-vous de cette analyse ? Que devrait être la prochaine étape ? (Indice: si l'effet de Chamber / genotype n'est pas significatif, vous pouvez augmenter la puissance des comparaisons entre génotypes en regroupant les chambres de chaque génotype.). Faites-le ! N'oubliez pas de vérifier les conditions d'applications de l'ANOVA!

Solution

```
anova(anova.nested)
```

```
Analysis of Variance Table

Response: survival
                 Df  Sum Sq Mean Sq  F value Pr(>F)
genotype          2 2952.22 1476.11 292.6081 <2e-16 ***
genotype:chamber  6   40.65    6.78   1.3432 0.2639
Residuals        36  181.61    5.04
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
par(mfrow = c(2, 2))
plot(anova.nested)
```

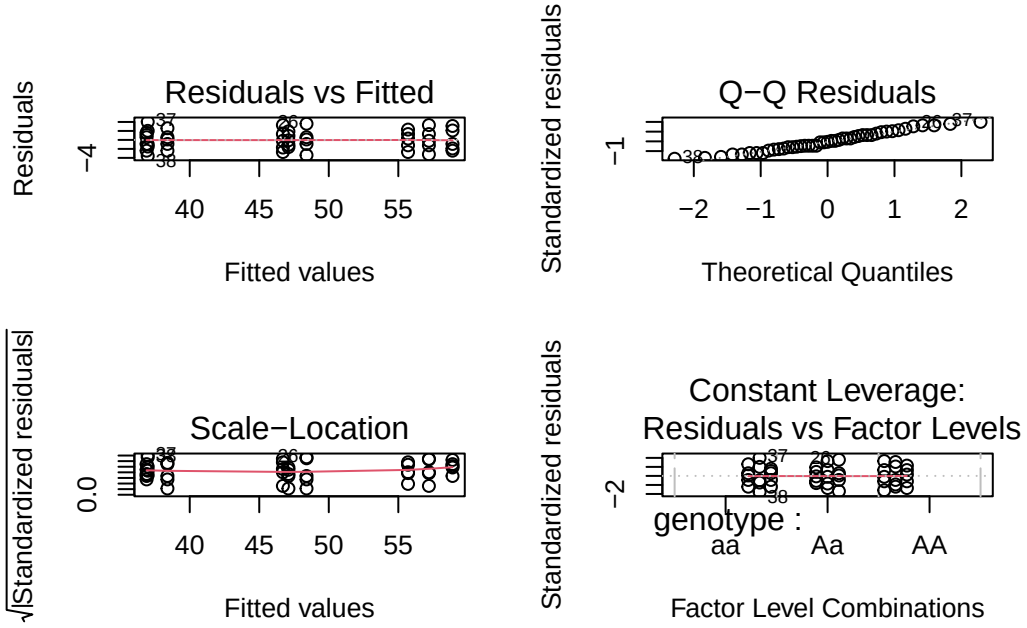


Figure 6.6.: Conditions d'applications du modèle anova.nested

On conclue de cette analyse que la variation entre les chambres de croissance n'est pas significative, mais qu'on doit rejeter l'hypothèse nulle que tous les génotypes ont la même résistance à la dessiccation.

Comme l'effet hiérarchique chamber / genotype n'est pas significatif, on peut regrouper les observations pour augmenter le nombre de degrés de liberté:

```
anova.simple <- lm(survival ~ genotype, data = nestdat)
anova(anova.simple)
```

```
Analysis of Variance Table

Response: survival
          Df  Sum Sq Mean Sq F value    Pr(>F)
genotype   2 2952.22 1476.11  278.93 < 2.2e-16 ***
Residuals 42  222.26    5.29
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Donc on conclue qu'il y a une variation significative de résistance à la dessiccation entre les trois génotypes.

Le graphique de `survival` en fonction du `génotype` suggère que la résistance à la dessiccation varie entre chaque génotype. On peut combiner cela avec un test de Tukey.

```
par(mfrow = c(1, 1))
# Compute and plot means and Tukey CI
means <- glht(anova.simple, linfct = mcp(
  genotype =
    "Tukey"
))
cimeans <- cld(means)
# use sufficiently large upper margin
old.par <- par(mai = c(1, 1, 1.25, 1))
# plot
plot(cimeans, las = 1) # las option to put y-axis labels as God intended them
```

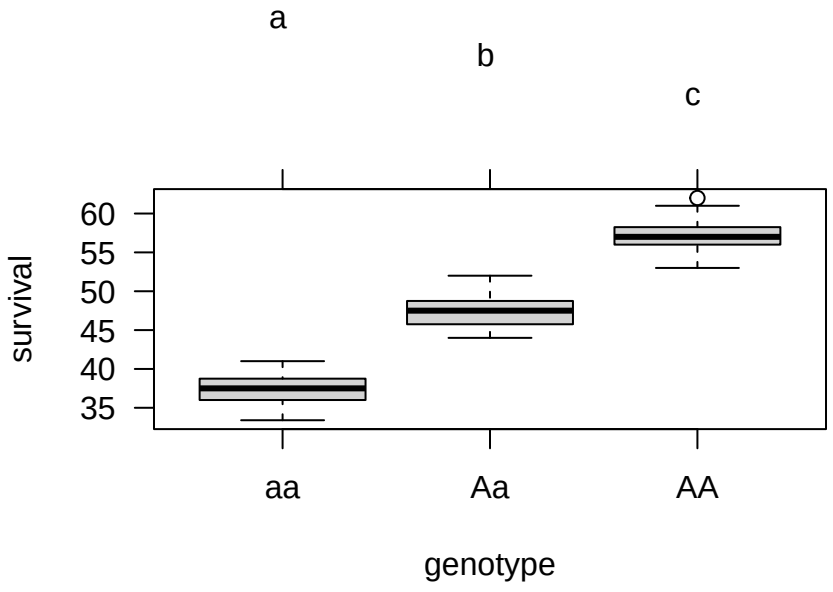


Figure 6.7.: Effet du genotype sur la résistance à la dessication avec un test de Tukey

On conclue donc que la résistance à la dessiccation (R), telle que mesurée par la survie dans des conditions chaudes et sèches, varie significativement entre les trois génotypes avec R(AA) > R(Aa) > R(aa).

Cependant, avant d'accepter cette conclusion, il faut éprouver les conditions d'application du test. Voici les diagnostics des résidus pour l'ANOVA à un critère de classification (non hiérarchique):

Solution

```
par(mfrow = c(2, 2))
plot(anova.simple)
```

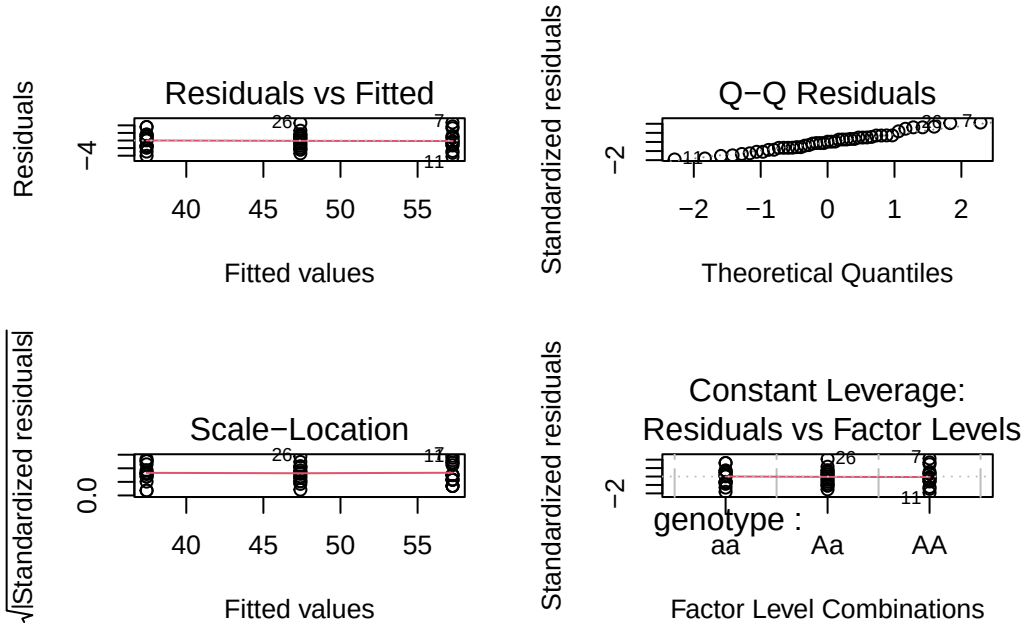


Figure 6.8.: Conditions d'applications du modèle anova.simple

Donc, toutes les conditions d'application semblent être remplies, et on peut donc accepter les conclusions. Notez que si l'on compare le carré moyen des résidus de l'ANOVA hiérarchique et de l'ANOVA à un critère de classification (5.045 vs 5.292), ils sont presque identiques. Cela n'est pas surprenant compte tenu de la faible variabilité associée aux chambres de croissance pour chaque génotype.

6.5. ANOVA non paramétrique avec deux facteurs de classification

L'ANOVA non paramétrique à deux critères de classification est une extension de celle à un critère de classification vue précédemment. Elle débute par une ANOVA faite sur les données transformées en rangs. Elle peut se faire sur des données avec ou sans réplicats.

À partir du fichier `stu2wdat.csv`, effectuez une ANOVA non paramétrique à deux facteurs de classification pour examiner l'effet de `sex` et `location` sur `rank(rdwght)`.

```
aov.rank <- aov(
  rank(rdwght) ~ sex * location,
  contrasts = list(
    sex = contr.sum, location = contr.sum
  ),
  data = Stu2wdat
)
```

L'extension de Schreirer-Ray-Hare au test de Kruskall-Wallis se fait ensuite à la main. Il faut d'abord calculer la statistique H égale au rapport de la somme des carrées de l'effet testé, divisée par le carré moyen total. On calcule la statistique H pour chacun des termes. Les statistiques H sont ensuite comparées à une distribution théorique de χ^2 (chi-carré) en utilisant la commande `pchisq(H, df, lower.tail = FALSE)`, où `H` et `df` sont les statistiques H calculées et les degrés de libertés, respectivement.

Testez l'effet de sex et location sur rdwght. Que concluez-vous ? Comment ce résultat se compare-t-il à celui obtenu en faisant l'ANOVA paramétrique faite précédemment ?

```
Anova(aov.rank, type = 3)
```

```
Anova Table (Type III tests)

Response: rank(rdwght)
              Sum Sq  Df  F value    Pr(>F)
(Intercept)  1499862   1 577.8673 < 2.2e-16 ***
sex            58394   1  22.4979 4.237e-06 ***
location        1128   1   0.4347    0.5105
sex:location    1230   1   0.4738    0.4921
Residuals     472383 182
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Pour calculer l'extension Schreirer-Ray-Hare au test de Kruskall-Wallis, on doit d'abord calculer le carré moyen total (MS), i.e. la variance des données transformées en rang. Ici, on a 186 observations, donc des rangs; 1, 2, 3, … 186. La variance de cette série de 186 valeurs peut être calculée simplement par `var(1:186)`.

Donc on peut calculer la statistique H pour chaque terme:

```
Hsex <- 58394 / var(1:186)
Hlocation <- 1128 / var(1:186)
Hsexloc <- 1230 / var(1:186)
```

Et convertir ces statistiques en valeur de ps:

```
# sex
Hsex
```

```
[1] 20.14628
```

```
pchisq(Hsex, 1, lower.tail = FALSE)
```

```
[1] 7.173954e-06
```

```
# location
Hlocation
```

```
[1] 0.3891668
```

```
pchisq(Hlocation, 1, lower.tail = FALSE)
```

```
[1] 0.5327377
```

```
# sex:location
Hsexloc
```

```
[1] 0.4243574
```

```
pchisq(Hsexloc, 1, lower.tail = FALSE)
```

```
[1] 0.5147707
```

Ces résultats sont semblables aux résultats de l'ANOVA non-paramétrique à deux critères de classification. Malgré la puissance réduite, il y a encore un effet significatif du sexe, mais ni interaction ni effet du site.

Il y a toutefois une différence importante. Rappelez-vous que dans l'ANOVA paramétrique il y avait un effet significatif de sex en considérant le problème comme un modèle ANOVA à effet fixe. Cependant, si on traite le problème comme un modèle d'ANOVA à effet mixte l'effet significatif de `sex` peut en principe disparaître parce que le nombre de degré de liberté (dl) associés au carré moyen (CM) de l'interaction est plus faible que le nombre de dl du CM de l'erreur du modèle à effet fixes. Dans ce cas ci, cependant, le CM de l'interaction est environ la moitié du CM de l'erreur. Par conséquent, l'effet significatif de `sex` pourrait devenir encore plus significatif si le problème est analysé (comme il se doit) comme une ANOVA mixte. Encore une fois on peut voir l'importance de spécifier le modèle adéquat en ANOVA.

6.6. Comparaisons multiples

Les épreuves d'hypothèses subséquentes en ANOVA à plus d'un critère de classification dépendent des résultats initiaux de l'ANOVA. Si vous êtes intéressés à comparer des effets moyens d'un facteur pour tous les niveaux d'un autre facteur (par exemple l'effet du sexe sur la taille des esturgeons peu importe d'où ils viennent), alors vous pouvez procéder exactement tel que décrit dans la section sur les comparaisons multiples suivant l'ANOVA à un critère de classification. Pour comparer les moyennes des cellules entre elles, il faut spécifier l'interaction comme variable qui représente le groupe.

Le fichier `wmcdat2.csv` contient des mesures de consommation d'oxygène, `o2cons`, de deux espèces, `species`, d'un mollusque (une patelle) à trois concentrations différentes d'eau de mer, `conc`. Ces données sont présentées à la p. 332 de Sokal et Rohlf 1995.

- Effectuez une ANOVA factorielle à deux critères de classification sur ces données en utilisant `o2cons` comme variable dépendante et `species` et `conc` comme les facteurs (il va probablement falloir changer le type de données de variable conc à facteur). Que concluez-vous ?

Solution

```
wmcdat2 <- read.csv("data/wmcdat2.csv")
wmcdat2$species <- as.factor(wmcdat2$species)
wmcdat2$conc <- as.factor(wmcdat2$conc)
anova.model5 <- lm(o2cons ~ species * conc, data = wmcdat2)
Anova(anova.model5, type = 3)
```

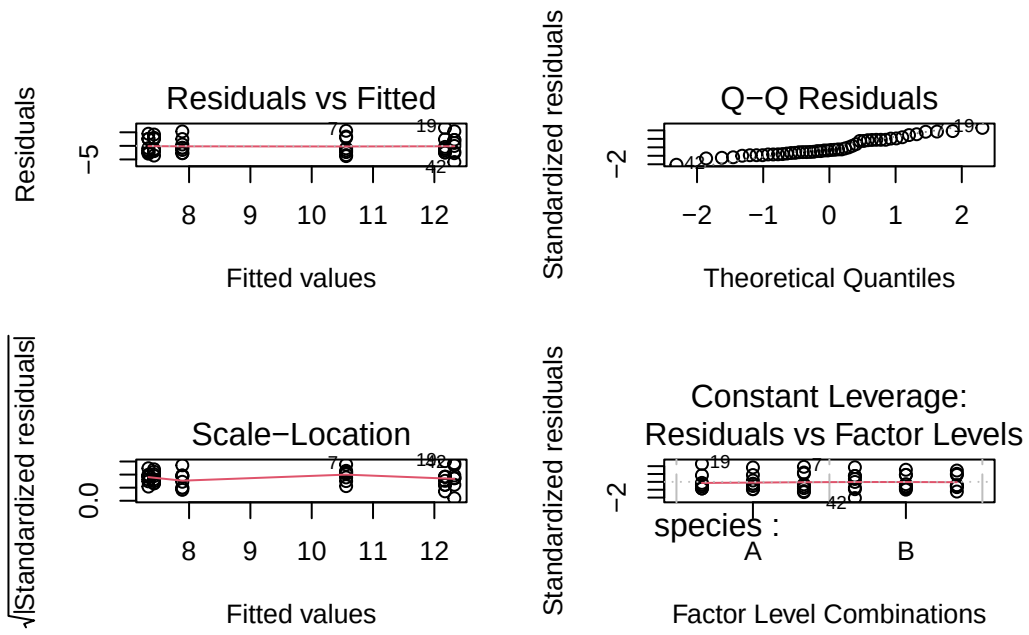
```
Anova Table (Type III tests)

Response: o2cons
              Sum Sq Df  F value    Pr(>F)
(Intercept)  1185.60  1 124.0165 4.101e-14 ***
species         0.09  1   0.0097   0.92189
conc           74.90  2   3.9172   0.02755 *
species:conc   23.93  2   1.2514   0.29656
Residuals     401.52 42
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Comme l'effectif dans chaque cellule est relativement petit, il faudrait idéalement refaire cette analyse avec une ANOVA non-paramétrique. Pour le moment, contentons nous de la version paramétrique.

Examinons les graphiques diagnostiques:

Solution

```
par(mfrow = c(2, 2))
plot(anova.model5)
```



Les variances semblent donc égales. Le test de normalité donne:

Solution

```
shapiro.test(residuals(anova.model5))
```

```
    Shapiro-Wilk normality test

data:  residuals(anova.model5)
W = 0.93692, p-value = 0.01238
```

Il y a donc évidence de non-normalité, mais à part ça tout semble aller. Comme l'ANOVA est relativement robuste à la non-normalité, on va regarder de l'autre coté. (Si vous voulez être plus confiants, vous pouvez tourner une ANOVA non paramétrique. Vous arriverez aux mêmes conclusions.)

- À la suite des résultats que vous venez d'obtenir, quelles moyennes voudriez-vous comparer ? Pourquoi?

Solution

ajouter une explication ici

On conclue donc qu'il n'y a pas de différence entre les espèces et que l'effet de la concentration ne dépends pas de l'espèce (il n'y a pas d'interaction). Par conséquent, les seules comparaisons justifiables sont entre les concentrations:

```
# fit simplified model
anova.model6 <- aov(o2cons ~ conc, data = wmcdat2)
# Make Tukey multiple comparisons
TukeyHSD(anova.model6)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = o2cons ~ conc, data = wmcdat2)

$conc
           diff       lwr        upr     p adj
75-50  -4.63625 -7.321998 -1.9505018 0.0003793
100-50 -3.25500 -5.940748 -0.5692518 0.0141313
100-75  1.38125 -1.304498  4.0669982 0.4325855
```

```
par(mfrow = c(1, 1))
# Graph of all comparisons for conc
tuk <- glht(anova.model6, linfct = mcp(conc = "Tukey"))
# extract information
tuk.cld <- cld(tuk)
# use sufficiently large upper margin
old.par <- par(mai = c(1, 1, 1.25, 1))
# plot
plot(tuk.cld)
```

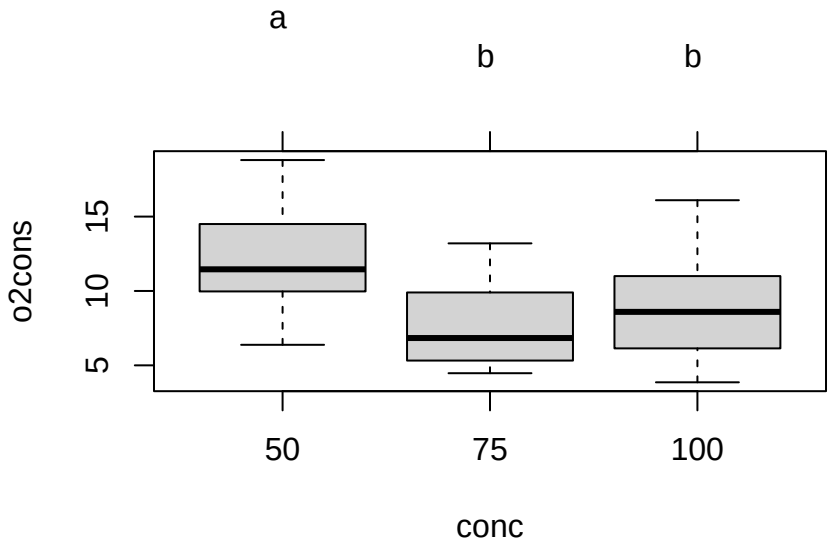


Figure 6.9.: Comparaison de Tukey des moyennes de consommation d'oxygèn en fonction del la concentration

```
par(old.par)
```

Il y a donc une différence de consommation d'oxygène significative lorsque la salinité est réduite de 50%, mais pas à 25% de réduction.

- Répétez les deux analyses précédentes sur les données du fichier `wmc2dat2.csv`. Comment les résultats se comparent-ils à ceux obt$nus précédemment ?

Solution

```
wmc2dat2 <- read.csv("data/wmc2dat2.csv")
wmc2dat2$species <- as.factor(wmc2dat2$species)
wmc2dat2$conc <- as.factor(wmc2dat2$conc)
anova.model7 <- lm(o2cons ~ species * conc, data = wmc2dat2)
```

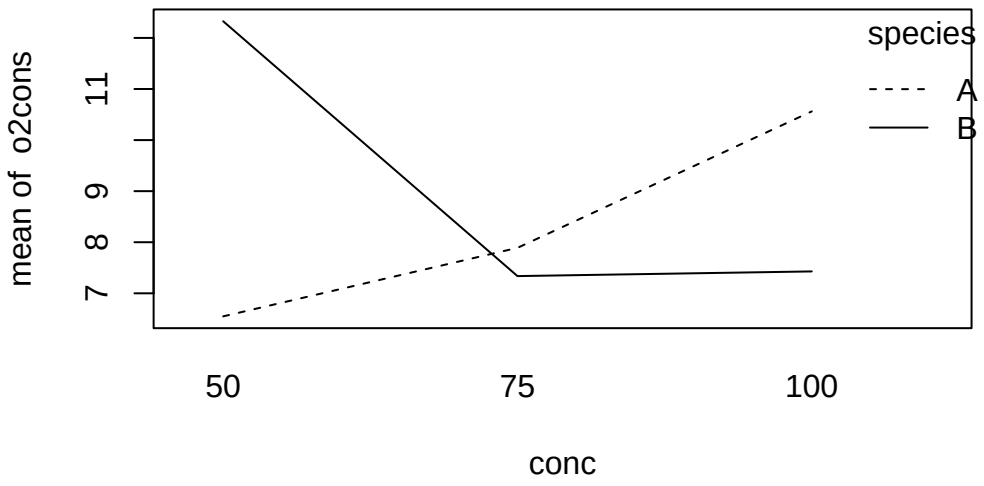
En utilisant `wmc2dat2.csv`, on obtient:

```
Anova Table (Type III tests)

Response: o2cons
             Sum Sq Df F value    Pr(>F)
(Intercept)  343.09  1 36.2132 3.745e-07 ***
species      133.52  1 14.0929 0.0005286 ***
```

```
conc          66.76  2  3.5232 0.0385011 *
species:conc 168.15  2  8.8742 0.0006101 ***
Residuals    397.91 42
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Dans ce cas ci, il y a une interaction significative, et il n'est par conséquent pas approprié de comparer les moyennes regroupées par espèce ou concentration. Ceci est clairement visualisé par un graphique d'interaction:

```
with(wmc2dat2, interaction.plot(conc, species, o2cons))
```



- Toujours en utilisant les données de `wmc2dat2.csv`, comparez les 6 moyennes avec l'ajustement Bonferonni. Pour ce faire, il sera utile de créer une nouvelle variable qui combine species et conc:

```
wmc2dat2$species.conc <- as.factor(paste0(wmc2dat2$species, wmc2dat2$conc))
```

ensuite on peut faire les comparaisons de Bonferroni:

```
with(wmc2dat2, pairwise.t.test(o2cons, species.conc, p.adj = "bonf"))
```

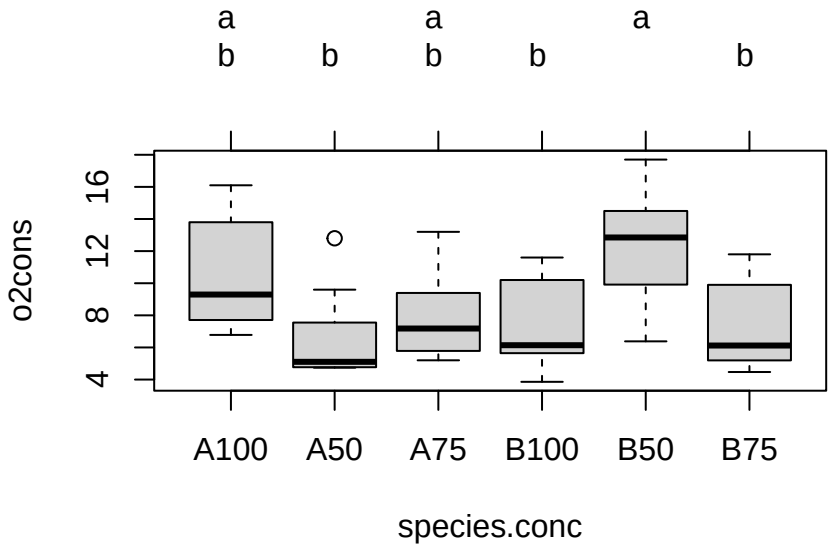
```
    Pairwise comparisons using t tests with pooled SD

data:  o2cons and species.conc
```

```
     A100   A50    A75    B100   B50
A50  0.1887 -      -      -      -
A75  1.0000 1.0000 -      -      -
B100 0.7223 1.0000 1.0000 -      -
B50  1.0000 0.0079 0.0929 0.0412 -
B75  0.6340 1.0000 1.0000 1.0000 0.0350

P value adjustment method: bonferroni
```

Ces comparaisons sont un peu plus difficiles à interpréter, mais l'analyse examine essentiellement les différences entre les concentrations de l'eau dans l'espèce A (nommé adj1) et pour les différences entre les concentrations dans l'espèce B (nommé adj2). Cette analyse indique que la différence principale est entre la concentration de 50% pour l'espèce B et les concentrations de 75 et 100% de l'espèce B, tandis qu'il n'y a aucunes différences significatives pour l'espèce A.

Je trouve ces tableaux de résultats peu satisfaisants parce qu'ils indiquent seulement les valeur de ps sans indices de la taille de l'effet. On peut obtenir à la fois le résultat des tests de comparaison multiple et un indice de la taille de l'effet à l'aide du code suivant:

```r
# fit one-way anova comparing all combinations of species.conc combinations
anova.modelx <- aov(o2cons ~ species.conc, data = wmc2dat2)
tuk2 <- glht(anova.modelx, linfct = mcp(species.conc = "Tukey"))
# extract information
tuk2.cld <- cld(tuk2)
# use sufficiently large upper margin
old.par <- par(mai = c(1, 1, 1.25, 1))
# plot
plot(tuk2.cld)
```



```
par(old.par)
```

Dans cette analyse on a utilisé le CM = 9.474 du modèle d'ANOVA pour comparer les moyennes. En ce faisant, on présume qu'il s'agit d'une situation d'ANOVA à effet fixes, ce qui n'est peut-être pas le cas (conc est certainement fixe, mais species peut être fixe ou aléatoire).

6.7. Test de permutation pour l'ANOVA à deux facteurs de classification

Quand les données ne rencontrent pas les conditions d'application des tests paramétriques d'ANOVA à un ou plusieurs facteurs de classification, il est possible d'utiliser les tests de permutation comme une alternative aux tests non-paramétriques pour calculer des p-valeurs. Le code suivant est pour un modèle I d'une ANOVA à deux facteurs de classification. Je vous laisse le soin d'adapter ce code pour d'autres modèles. (J'offre même des points boni pour une solution élégante pour des modèles à plusieurs facteurs de classification).

```
############################################################
# Permutation test for two way ANOVA
# Ter Braak creates residuals from cell means and then permutes across
# all cells
# This can be accomplished by taking residuals from the full model
# modified from code written by David C. Howell
# http://www.uvm.edu/~dhowell/StatPages/More_Stuff/Permutation%20Anova/PermTestsAnova.html
nreps <- 500
dependent <- Stu2wdat$rdwght
factor1 <- as.factor(Stu2wdat$sex)
factor2 <- as.factor(Stu2wdat$location)
my.dataframe <- data.frame(dependent, factor1, factor2)
my.dataframe.noNA <- my.dataframe[complete.cases(my.dataframe), ]
mod <- lm(dependent ~ factor1 + factor2 + factor1:factor2,
  data = my.dataframe.noNA
)
res <- mod$residuals
TBint <- numeric(nreps)
TB1 <- numeric(nreps)
TB2 <- numeric(nreps)
ANOVA <- summary(aov(mod))
cat(
  " The standard ANOVA for these data follows ",
  "\n"
)
F1 <- ANOVA[[1]]$"F value"[1]
F2 <- ANOVA[[1]]$"F value"[2]
Finteract <- ANOVA[[1]]$"F value"[3]
print(ANOVA)
cat("\n")
cat("\n")
```

```
TBint[1] <- Finteract
for (i in 2:nreps) {
  newdat <- sample(res, length(res), replace = FALSE)
  modb <- summary(aov(newdat ~ factor1 + factor2 +
    factor1:factor2,
  data = my.dataframe.noNA
  ))
  TBint[i] <- modb[[1]]$"F value"[3]
  TB1[i] <- modb[[1]]$"F value"[1]
  TB2[i] <- modb[[1]]$"F value"[2]
}
probInt <- length(TBint[TBint >= Finteract]) / nreps
prob1 <- length(TB1[TB1 >= F1]) / nreps
prob2 <- length(TB2[TB1 >= F2]) / nreps
cat("\n")
cat("\n")
print("Resampling as in ter Braak with unrestricted sampling
of cell residuals. ")
cat(
  "The probability for the effect of Interaction is ",
  probInt, "\n"
)
cat(
  "The probability for the effect of Factor 1 is ",
  prob1, "\n"
)
cat(
  "The probability for the effect of Factor 2 is ",
  prob2, "\n"
)
```

Si vous avez la chance d'avoir accès au package lmPerm, vous pouvez effectuer le test de permutation beaucoup plus rapidement et facilement:

```
#############################################################################
## lmPerm version of permutation test
library(lmPerm)
# for generality, copy desired dataframe to mydata
# and model formula to myformula
mydata <- Stu2wdat
myformula <- as.formula("rdwght ~ sex+location+sex:location")
# Fit desired model on the desired dataframe
mymodel <- lm(myformula, data = mydata)
# Calculate permutation p-value
anova(lmp(myformula, data = mydata, perm = "Prob", center = FALSE, Ca = 0.001))
```

6.8. Bootstrap pour l'ANOVA à deux facteurs de classification

Dans la plupart des cas, les tests de permutation seront plus appropriés que le bootstrap pour les designs d'ANOVA. J'ai quand même un bout de code qui pourra servir si vous en avez besoin:

```
############################################################
############
# Bootstrap for two-way ANOVA
# You possibly want to edit bootfunction.mod1 to return other values
# Here it returns the standard coefficients of the fitted model
# Requires boot library
#
nreps <- 5000
dependent <- Stu2wdat$rdwght
factor1 <- as.factor(Stu2wdat$sex)
factor2 <- as.factor(Stu2wdat$location)
my.dataframe <- data.frame(dependent, factor1, factor2)
my.dataframe.noNA <- my.dataframe[complete.cases(my.dataframe), ]
library(boot)
# Fit model on observed data
mod1 <- aov(dependent ~ factor1 + factor2 + factor1:factor2,
  data = my.dataframe.noNA
)



# Bootstrap 1000 time using the residuals bootstraping methods to
# keep the same unequal number of observations for each level of the indep. var.
fit <- fitted(mod1)
e <- residuals(mod1)
X <- model.matrix(mod1)
bootfunction.mod1 <- function(data, indices) {
  y <- fit + e[indices]
  bootmod <- lm(y ~ X)
  coefficients(bootmod)
}
bootresults <- boot(my.dataframe.noNA, bootfunction.mod1,
  R = 1000
)
bootresults
## Calculate 90% CI and plot bootstrap estimates separately for each model parameter
boot.ci(bootresults, conf = 0.9, index = 1)
plot(bootresults, index = 1)
boot.ci(bootresults, conf = 0.9, index = 3)
plot(bootresults, index = 3)
boot.ci(bootresults, conf = 0.9, index = 4)
plot(bootresults, index = 4)
boot.ci(bootresults, conf = 0.9, index = 5)
plot(bootresults, index = 5)
```

7. Régression multiple

Après avoir complété cet exercice de laboratoire, vous devriez pouvoir :

- Utiliser R pour ajuster une régression multiple et comparer des modèles selon l'approche inférentielle et celle de la théorie de l'information
- Utiliser R pour éprouver des hypothèses sur l'effet des variables indépendantes sur la variable dépendante.
- Utiliser R pour évaluer la multicolinéarité entre les variables indépendantes et en évaluer ses effets
- Utiliser R pour effectuer une régression curvilinéaire (polynomiale).

7.1. Paquets et données requises pour le labo

Ce laboratoire nécessite:

- les paquets R:
  - ggplot2
  - car
  - lmtest
  - simpleboot
  - boot
  - MuMIn
- les fichiers de données
  - Mregdat.csv

7.2. Conseils généraux

Les variables qui intéressent les biologistes sont généralement influencées par plusieurs facteurs, et une description exacte ou une prédiction de la variable dépendante requiert que plus d'une variable soit incluse dans le modèle. La régression multiple permet de quantifier l'effet de plusieurs variables continues sur la variable dépendante.

Il est important de réaliser que la maîtrise de la régression multiple ne s'acquiert pas instantanément. Les débutants doivent garder à l'esprit plusieurs points importants :

1. Un modèle de régression multiple peut être hautement significatif même si aucun des termes pris isolément ne l'est (ceci est causé par la multicolinéarité),
2. Un modèle peut ne pas être significatif alors que l'un ou plusieurs des termes le sont (ceci est un signe d'un modèle trop complexe ("overfitting")) et,
3. À moins que les variables indépendantes soient parfaitement orthogonales (c'est-à-dire qu'il n'y ait aucune corrélation entre elles et donc pas de multicolinéarité) les diverses approches de sélection des variables indépendantes peuvent mener à des modèles différents.

7.3. Premières régressions multiples

Le fichier `Mregdat.csv` contient des données de richesse spécifique de quatre groupes d'organismes dans 30 marais de la région Ottawa-Cornwall-Kingston. Les variables sont:

- la richesse spécifique:
  - des oiseaux (bird, et son logarithme base 10 logbird)
  - des mammifères (mammal, logmam)
  - des amphibiens et reptiles (herptile, logherp)
  - des vertébrés (totsp, logtot)
- les coordonnées des sites (lat, long)
- la superficie du marais (logarea)
- le pourcentage du marais inondé toute l'année (swamp)
- le pourcentage des terres couvertes par des forêts dans un rayon de 1km du marais (cpfor2)
- la densité des routes pavées (en m/ha) dans un rayon de 1km du marais (thtden).

Nous allons nous concentrer sur les amphibiens et les reptiles (herptile) pour cet exemple, il est donc avisé d'examiner la distribution de cette variable et les corrélations avec les variables indépendantes potentielles:

```
mydata <- read.csv("data/Mregdat.csv")
scatterplotMatrix(
  ~ logherp + logarea + cpfor2 + thtden + swamp,
  regLine = TRUE, smooth = TRUE, diagonal = TRUE,
  data = mydata
)
```

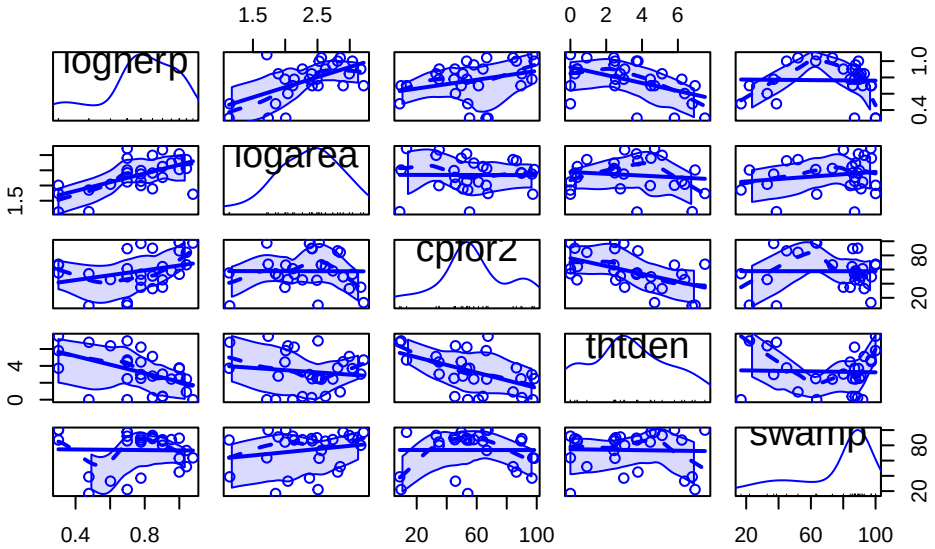


Figure 7.1.: Matrice de rélation et densité pour la richesse spécifique des amphibiens et reptiles

- En utilisant les données de ce fichier, faites la régression simple de `logherp` en fonction de `logarea` . Que concluez-vous à partir de cette analyse?

```r
model.loga <- lm(logherp ~ logarea, data = mydata)
summary(model.loga)
```

```
Call:
lm(formula = logherp ~ logarea, data = mydata)

Residuals:
     Min       1Q   Median       3Q      Max
-0.38082 -0.09265  0.00763  0.10409  0.46977

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.18503    0.15725   1.177 0.249996
logarea      0.24736    0.06536   3.784 0.000818 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1856 on 26 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.3552,    Adjusted R-squared:  0.3304
F-statistic: 14.32 on 1 and 26 DF,  p-value: 0.0008185
```

```r
par(mfrow = c(2, 2))
plot(model.loga)
```

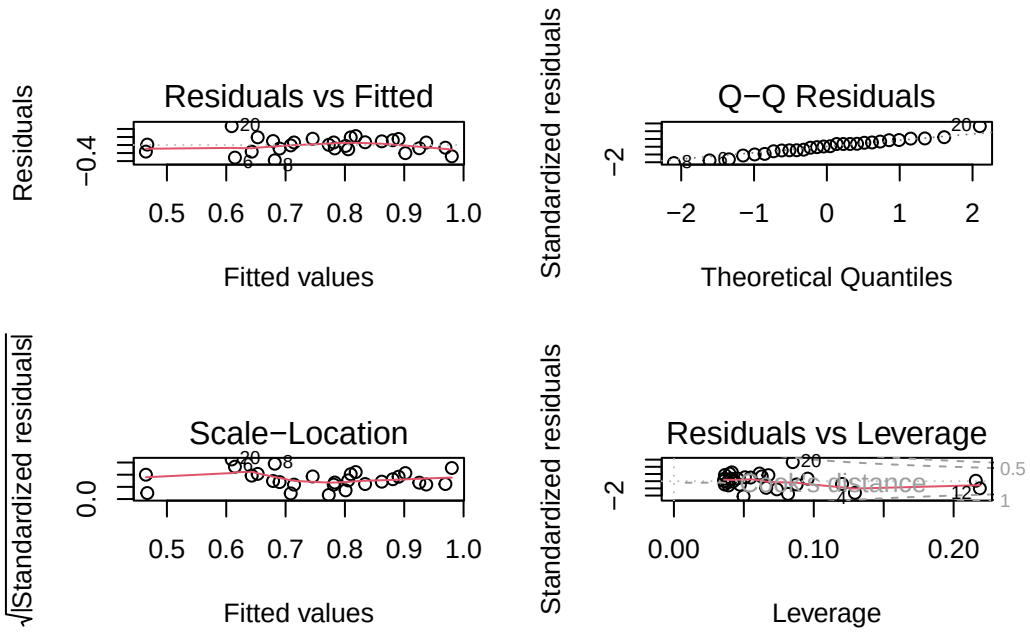


Figure 7.2.: Conditions d'applications de la régression de *logherp* sur *logarea*

Il semble donc y avoir une relation positive entre la richesse spécifique des reptiles et des amphibiens et la surface des marais. La régression n'explique cependant qu'environ le tiers de la variabilité (R 2 =0.355). L'analyse des résidus indique qu'il n'y a pas de problème avec la normalité, l'homoscédasticité, ni l'indépendance.

- Faites ensuite la régression de `logherp` en fonction de `cpfor2` . Que concluez-vous?

Solution

```
model.logcp <- lm(logherp ~ cpfor2, data = mydata)
summary(model.logcp)
```

```
Call:
lm(formula = logherp ~ cpfor2, data = mydata)

Residuals:
     Min       1Q   Median       3Q      Max
-0.49095 -0.10266  0.05881  0.16027  0.25159

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.609197   0.104233   5.845 3.68e-06 ***
cpfor2      0.002706   0.001658   1.632    0.115
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.2202 on 26 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.09289,   Adjusted R-squared:  0.058
F-statistic: 2.662 on 1 and 26 DF,  p-value: 0.1148
```

Ici, on doit accepter l'hypothèse nulle et conclure qu'il n'y a pas de relation entre la richesse spécifique dans les marais (`logherp`) et la proportion de forêts sur les terres adjacentes (`cpfor2`). Qu'est-ce qui arrive quand on fait une régression avec les 2 variables indépendantes?

- Refaites la régression de logherp enfonction de logarea et cpfor2 à la fois. Que concluez-vous?

Solution

```
model.mcp <- lm(logherp ~ logarea + cpfor2, data = mydata)
summary(model.mcp)
```

```
Call:
lm(formula = logherp ~ logarea + cpfor2, data = mydata)

Residuals:
     Min       1Q   Median       3Q      Max
-0.40438 -0.11512  0.01774  0.08187  0.36179

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.027058   0.166749   0.162 0.872398
logarea     0.247789   0.061603   4.022 0.000468 ***
cpfor2      0.002724   0.001318   2.067 0.049232 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.175 on 25 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.4493,    Adjusted R-squared:  0.4052
F-statistic:  10.2 on 2 and 25 DF,  p-value: 0.0005774
```

On voit donc qu'on peut rejeter les 2 hypothèses nulles que la pente de la régression de `logherp` sur `logarea` est zéro et que la pente de la régression de `logherp` sur `cpfor2` est zéro. Pourquoi `cpfor2` devient-il un facteur significatif dans la régression multiple alors qu'il n'est pas significatif dans la régression simple? Parce qu'il est parfois nécessaire de contrôler pour l'effet d'une variable pour pouvoir détecter les effets plus subtils d'autres variables. Ici, il y a une relation significative entre `logherp` et `logarea` qui masque l'effet de `cpfor2` sur `logherp` . Lorsque le modèle tient compte des deux variables explicatives, il devient possible de détecter l'effet de `cpfor2` .

- Ajustez un autre modèle, cette fois en remplaçant `cpfor2` par `thtden` (logherp ~ logarea + thtden). Que concluez-vous?

Solution

```
model.mden <- lm(logherp ~ logarea + thtden, data = mydata)
summary(model.mden)
```

```
Call:
lm(formula = logherp ~ logarea + thtden, data = mydata)

Residuals:
     Min       1Q   Median       3Q      Max
-0.31583 -0.12326  0.02095  0.13201  0.31674

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.37634    0.14926   2.521 0.018437 *
logarea      0.22504    0.05701   3.947 0.000567 ***
thtden      -0.04196    0.01345  -3.118 0.004535 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1606 on 25 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.5358,    Adjusted R-squared:  0.4986
F-statistic: 14.43 on 2 and 25 DF,  p-value: 6.829e-05
```

On rejette donc l'hypothèse nulle que la richesse spécifique n'est pas influencée par la taille des marais (`logarea`) ni par la densité des routes (`thtden`). Notez qu'ici il y a une relation négative significative entre la richesse spécifique des amphibiens et reptiles et la densité des routes sur les terres adjacentes, tandis que la relation est positive pour la taille des marais et pour la densité des forêts (`cpfor2` ; résultat de la dernière régression).

Le R^2 de ce modèle est plus élevé que pour le précédent, reflétant une corrélation plus forte entre logherp et thtden qu'entre `logherp` et `cpfor2` .

La richesse spécifique des reptiles et amphibiens semble donc reliée à la surface de marais (`logarea`), la densité des routes (`thtden`), et possiblement au couvert forestier sur les terres adjacentes aux marais (`cpfor2`). Cependant, les trois variables ne sont peut-être pas nécessaires dans un modèle prédictif. Si deux des trois variables (disons `cpfor2` et `thtden`) sont parfaitement corrélées, alors l'effet de `thtden` ne serait rien de plus que celui de `cpfor2` (et vice-versa) et un modèle incluant l'une des deux variables ferait des prédictions identiques à un modèle incluant ces deux variables (en plus de `logarea`).

- Estimez un modèle de régression avec `logherp` comme variable dépendante et `logarea`, `cpfor2` et `thtden` comme variables indépendantes. Que concluez-vous?

Solution

```
model.mtri <- lm(logherp ~ logarea + cpfor2 + thtden, data = mydata)
summary(model.mtri)
```

```
Call:
lm(formula = logherp ~ logarea + cpfor2 + thtden, data = mydata)

Residuals:
     Min       1Q   Median       3Q      Max
-0.30729 -0.13779  0.02627  0.11441  0.29582

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.284765   0.191420   1.488 0.149867
logarea      0.228490   0.057647   3.964 0.000578 ***
cpfor2       0.001095   0.001414   0.774 0.446516
thtden      -0.035794   0.015726  -2.276 0.032055 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1619 on 24 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.5471,    Adjusted R-squared:  0.4904
F-statistic: 9.662 on 3 and 24 DF,  p-value: 0.0002291
```

Plusieurs choses sont à noter ici: 1. Tel que prédit, le coefficient de régression pour `cpfor2` n'est plus significativement différent de 0. Une fois que la variabilité attribuable à `logarea` et `thtden` est enlevée, il ne reste qu'une fraction nonsignificative de la variabilité attribuable à `cpfor2` . 2. Le R^2 pour ce modèle(0.547) n'est que légèrement supérieur au R^2 du modèle avec seulement `logarea` et `thtden` (.536), ce qui confirme que `cpfor2` n'explique pas grand-chose de plus.

Notez aussi que même si le coefficient de régression pour `thtden` n'a pas beaucoup changé par rapport à ce qui avait été estimé lorsque seul `thtden` et `logarea` étaient dans le modèle (0-.036 vs -0.042), l'erreur type pour l'estimé du coefficient est plus grand, et ce modèle plus complexe mène à un estimé moins précis. Si la corrélation entre `thtden` et `cpfor2` était plus forte, la décroissance de la précision serait encore plus grande.

On peut comparer les deux derniers modèles (i.e., le modèle incluant les 3 variables et celui avec seulement logarea and thtden) pour décider lequel privilégier.

```
anova(model.mtri, model.mden)
```

```
Analysis of Variance Table

Model 1: logherp ~ logarea + cpfor2 + thtden
```

```
Model 2: logherp ~ logarea + thtden
  Res.Df     RSS Df Sum of Sq     F Pr(>F)
1     24 0.62937
2     25 0.64508 -1 -0.015708 0.599 0.4465
```

Cette comparaison révèle que le modèle à 3 variables ne fait pas de prédictions significativement meilleures que le modèle avec seulement Logarea et thtden . Ce résultat n'est pas surprenant puisque le test de signification pour cpfor2 dans le modèle complet indique qu'il faut accepter l'hypothèse nulle.

À la suite de cette analyse, on doit conclure que:

1. Le meilleur modèle est celui incluant thtden et logarea .
2. Il y a une relation négative entre la richesse spécifique des amphibiens et reptiles et la densité des routes sur les terres adjacentes.
3. Il y a une relation positive entre la richesse spécifique et la taille des marais.

Notez que le "meilleur" modèle n'est pas nécessairement le modèle parfait, seulement le meilleur n'utilisant que ces trois variables indépendantes. Il est évident qu'il y a d'autres facteurs qui contrôlent la richesse spécifique dans les marais puisque, même le "meilleur" modèle n'explique que la moitié de la variabilité.

7.4. Régression multiple pas-à-pas (stepwise)

Quand le nombre de variables prédictives est restreint, comme dans l'exemple précédent, il est aisé de comparer manuellement les modèles pour sélectionner le plus adéquat. Cependant, lorsque le nombre de variables indépendantes augmente, cette approche n'est rapidement plus utilisable et il est alors utile d'utiliser une méthode automatisée.

La sélection pas à pas avec R utilise le Critère Informatif de Akaike (Akaike Information Criterion, $AIC = 2 * ln(RSS) + 2K$ où K le nombre de variables indépendantes, n est le nombre d'observations, et RSS est la somme des carrés des résidus) comme mesure de la qualité d'ajustement des modèles. Cette mesure favorise la précision des prédictions et pénalise la complexité. **Lorsque l'on compare des modèles par AIC, le modèle avec le plus petit AIC est le modèle à préférer.**

- Utiliser la fonction stepAIC pour activer la sélection pas à pas des variables indépendantes sur le modèles de régression incluant `logarea`, `cpfor2` et `thtden`:

```
# Stepwise Regression
library(MASS)
step.mtri <- stepAIC(model.mtri, direction = "both")
```

```
Start:  AIC=-98.27
logherp ~ logarea + cpfor2 + thtden

           Df Sum of Sq     RSS     AIC
- cpfor2    1   0.01571 0.64508 -99.576
<none>                 0.62937 -98.267
- thtden    1   0.13585 0.76522 -94.794
- logarea   1   0.41198 1.04135 -86.167
```

```
Step:  AIC=-99.58
logherp ~ logarea + thtden

          Df Sum of Sq     RSS     AIC
<none>                 0.64508 -99.576
+ cpfor2   1   0.01571 0.62937 -98.267
- thtden   1   0.25092 0.89600 -92.376
- logarea  1   0.40204 1.04712 -88.013
```

```
step.mtri$anova # display results
```

```
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
logherp ~ logarea + cpfor2 + thtden

Final Model:
logherp ~ logarea + thtden

      Step Df   Deviance Resid. Df Resid. Dev       AIC
1                               24  0.6293717 -98.26666
2 - cpfor2  1 0.01570813        25  0.6450798 -99.57640
```

R nous donne:

1. L'ajustement (mesuré par AIC) du modèle complet en premier lieu.
2. L'AIC des modèles dans lesquels une variable a été enlevée du modèle complet. Notez que c'est seulement en enlevant cpfor2 du modèle qu'on peut réduire l'AIC
3. La valeur de AIC pour les modèles auxquels on enlève ou on ajoute une variable au modèle sélectionné à la première étape.(i.e. logherp ~ logarea + thtden). Notez qu'aucun des modèles n'a un AIC inférieur à ce modèle.

Au lieu de débuter par le modèle complet (saturé) et enlever des termes, on peut commencer par le modèle nul et ajouter des termes:

```
# Forward selection approach
model.null <- lm(logherp ~ 1, data = mydata)
step.f <- stepAIC(
  model.null,
  scope = ~ . + logarea + cpfor2 + thtden, direction = "forward"
)
```

```
Start:  AIC=-82.09
logherp ~ 1

          Df Sum of Sq    RSS     AIC
+ logarea  1   0.49352 0.8960 -92.376
+ thtden   1   0.34241 1.0471 -88.013
+ cpfor2   1   0.12907 1.2605 -82.820
<none>                 1.3895 -82.091

Step:  AIC=-92.38
logherp ~ logarea

         Df Sum of Sq     RSS     AIC
+ thtden  1   0.25093 0.64508 -99.576
+ cpfor2  1   0.13078 0.76522 -94.794
<none>                0.89600 -92.376

Step:  AIC=-99.58
logherp ~ logarea + thtden

         Df Sum of Sq     RSS     AIC
<none>                0.64508 -99.576
+ cpfor2  1  0.015708 0.62937 -98.267
```

```
step.f$anova # display results
```

```
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
logherp ~ 1

Final Model:
logherp ~ logarea + thtden

       Step Df  Deviance Resid. Df Resid. Dev       AIC
1                               27  1.3895281 -82.09073
2 + logarea  1 0.4935233        26  0.8960048 -92.37639
3  + thtden  1 0.2509250        25  0.6450798 -99.57640
```

Le résultat final est le même, mais la trajectoire est différente. Dans ce cas, R débute avec le modèle le plus simple et ajoute une variable indépendante à chaque étape, sélectionnant la variable minimisant AIC à cette étape. Le modèle de départ a donc seulement une ordonnée à l'origine. Puis, `logarea` est ajouté, suivi de `thtden`. `cpfor2` n'est pas ajouté au modèle, car son addition fait augmenter l'AIC.

Il est recommandé de comparer le résultat final de plusieurs approches. Si le modèle retenu diffère selon l'approche utilisée, c'est un signe que le "meilleur" modèle est possiblement difficile à identifier et que vous

devriez être circonspects dans vos inférences. Dans notre exemple, pas de problème: toutes les méthodes convergent sur le même modèle final.

Pour conclure cette section, quelques conseils concernant les méthodes automatisées de sélection des variables indépendantes:

1. Les différentes méthodes de sélection des variables indépendantes peuvent mener à des modèles différents. Il est souvent utile d'essayer plus d'une méthode et de comparer les résultats. Si les résultats diffèrent, c'est presque toujours à cause de multicolinéarité entre les variables indépendantes.
2. Attention à la régression pas-à-pas. Les auteurs de SYSTAT en disent:

   > Stepwise regression is probably the most abused computerized statistical technique ever devised. If you think you need automated stepwise regression to solve a particular problem, you probably don't. Professional statisticians rarely use automated stepwise regression because it does not necessarily find the "best" fitting model, the "real" model, or alternative "plausible" models. Furthermore, the order in which variables enter or leave a stepwise program is usually of no theoretical significance. You are always better off thinking about why a model could generate your data and then testing that model.

En bref, on abuse trop souvent de cette technique.

3. Il faut toujours garder à l'esprit que l'existence d'une régression significative n'est pas suffisante pour prouver une relation causale.

7.5. Détecter la multicolinéarité

La multicolinéarité est la présence de corrélations entre les variables indépendantes. Lorsqu'elle est extrême (multicolinéarité parfaite) elle empêche l'estimation des modèles statistiques. Lorsqu'elle est grande ou modérée, elle réduit la puissance de détection de l'effet des variables indépendantes individuellement, mais elle n'empêche pas le modèle de faire des prédictions.

Un des indices les plus utilisés pour quantifier la multicolinéarité et le facteur d'inflation de la variance (VIF, variance inflation factor). Le fichier d'aide du package HH explique ainsi son calcul:

> A simple diagnostic of collinearity is the variance inflation factor, VIF one for each regression coefficient (other than the intercept). Since the condition of collinearity involves the predictors but not the response, this measure is a function of the X's but not of Y. The VIF for predictor i is $1/(1 - R_i^2)$, where R_i^2 is the R^2 from a regression of predictor i against the remaining predictors. If R_i^2 is close to 1, this means that predictor i is well explained by a linear function of the remaining predictors, and, therefore, the presence of predictor i in the model is redundant. Values of VIF exceeding 5 are considered evidence of collinearity: The information carried by a predictor having such a VIF is contained in a subset of the remaining predictors. If, however, all of a model's regression coefficients differ significantly from 0 (p-value $< .05$), a somewhat larger VIF may be tolerable.

Bref, les VIF indiquent de combien l'incertitude de chaque coefficient de régression est augmentée par la multicolinéarité.

Attrappe. Il y a plusieurs fonctions vif() (j'en connais au moins trois dans les extensions `car`, `HH` et `DAAG`), et je ne sais pas en quoi elles diffèrent.

On peut calculer les VIF avec la fonction `vif()` de l'extension `car`: :

```
library(car)
vif(model.mtri)
```

```
 logarea   cpfor2   thtden
1.022127 1.344455 1.365970
```

Ici, il n'y a pas d'évidence de multicolinéarité car toutes les valeurs de VIF sont près de 1.

7.6. Régression polynomiale

La régression requiert la linéarité de la relation entre les variables dépendante et indépendante(s). Lorsque la relation n'est pas linéaire, il est parfois possible de linéariser la relation en effectuant une transformation sur une ou plusieurs variables. Cependant, dans bien des cas il est impossible de transformer les axes pour rendre la relation linéaire. On doit alors utiliser une forme ou l'autre de régression nonlinéaire. La forme la plus simple de régression non-linéaire est la régression polynomiale dans laquelle les variables indépendantes sont à une puissance plus grande que 1 (Ex : X^2 ou X^3).

- Faites un diagramme de dispersion des résidus (residual) de la régression `logherp ~ logarea` en fonction de swamp .

Solution

```
# problème avec les données de manquantes dans logherp
mysub <- subset(mydata, !is.na(logherp))
# ajouter les résidus dans les donnée
mysub$resloga <- residuals(model.loga)
ggplot(data = mysub, aes(y = resloga, x = swamp)) +
  geom_point() +
  geom_smooth()
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

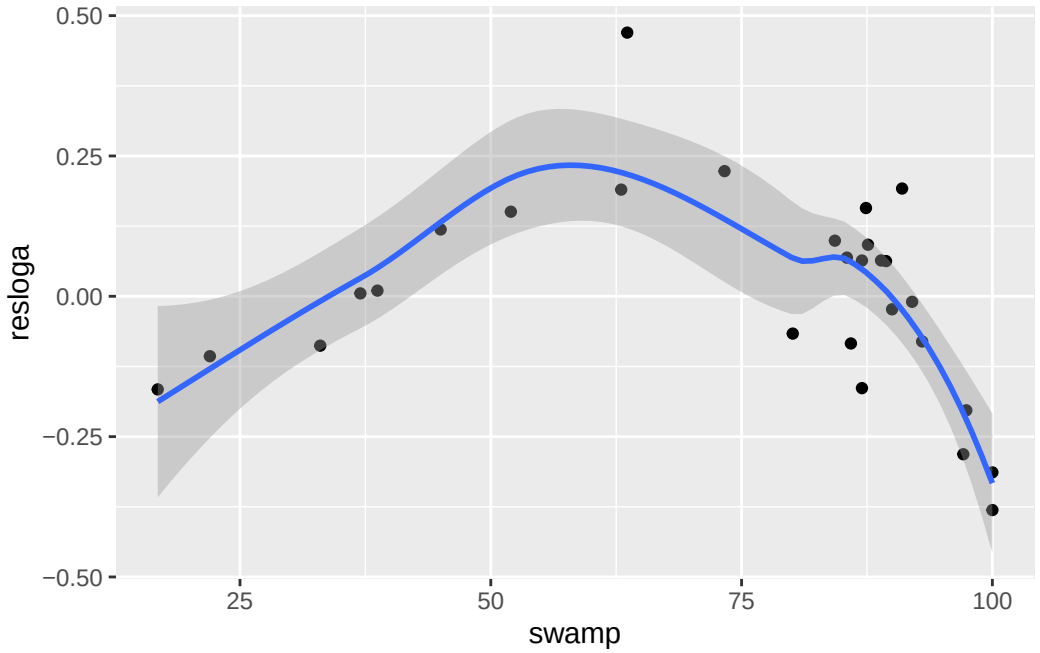


Figure 7.3.: Relation entre swamp et les résidus de la régression entre logherp et logarea

- L'examen de ce graphique suggère qu'il y a une forte relation entre les deux variables, mais qu'elle n'est pas linéaire. Essayez de faire une régression de residual sur swamp . Quelle est votre conclusion?

Solution

```
model.resloga <- lm(resloga ~ swamp, mysub)
summary(model.resloga)
```

```
Call:
lm(formula = resloga ~ swamp, data = mysub)

Residuals:
     Min       1Q   Median       3Q      Max
-0.35088 -0.13819  0.00313  0.10849  0.45802

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.084571   0.109265   0.774    0.446
swamp       -0.001145   0.001403  -0.816    0.422

Residual standard error: 0.1833 on 26 degrees of freedom
Multiple R-squared:  0.02498,   Adjusted R-squared:  -0.01252
F-statistic: 0.666 on 1 and 26 DF,  p-value: 0.4219
```

En deux mots, l'ajustement est épouvantable! Malgré le fait que le graphique suggère une relation très forte entre les deux variables. Cependant, cette relation n'est pas linéaire… (ce qui est également apparent si vous examinez les résidus du modèle linéaire).

- Refaites la régression d'en haut, mais cette fois incluez un terme pour représenter $swamp^2$. L'expression devrait apparaître comme: $Y\ X + I(X^2)$. Que concluez-vous? Qu'est-ce que l'examen des résidus de cette régression multiple révèle?

Solution

```
model.resloga2 <- lm(resloga ~ swamp + I(swamp^2), mysub)
summary(model.resloga2)
```

```

Call:
lm(formula = resloga ~ swamp + I(swamp^2), data = mysub)

Residuals:
      Min        1Q    Median        3Q       Max
-0.181185 -0.085350  0.007377  0.067327  0.242455

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -7.804e-01  1.569e-01  -4.975 3.97e-05 ***
swamp        3.398e-02  5.767e-03   5.892 3.79e-06 ***
I(swamp^2)  -2.852e-04  4.624e-05  -6.166 1.90e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1177 on 25 degrees of freedom
Multiple R-squared:  0.6132,    Adjusted R-squared:  0.5823
F-statistic: 19.82 on 2 and 25 DF,  p-value: 6.972e-06
```

Il devient évident que si on corrige la richesse spécifique pour la taille des marais, une fraction importante de la variabilité résiduelle peut être associée à swamp, selon une relation quadratique. Si vous examinez les résidus, vous observerez que l'ajustement est nettement meilleur qu'avec le modèle linéaire.

Solution

```
par(mfrow = c(2, 2))
plot(model.resloga2)
```

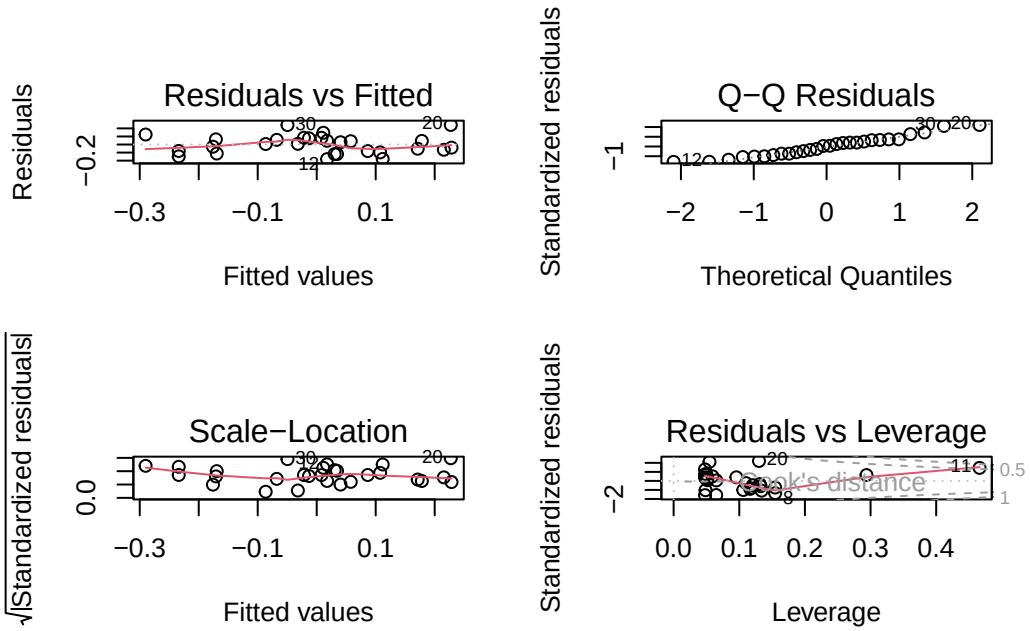


Figure 7.4.: Relation

- En vous basant sur les résultats de la dernière analyse, comment suggérez-vous de modifier le modèle de régression multiple? Quel est, d'après vous, le meilleur modèle? Pourquoi? Ordonnez les différents facteurs en ordre croissant de leur effet sur la richesse spécifique des reptiles.

Suite à ces analyses, il semble opportun d'essayer d'ajuster un modèle incluant logarea, thtden, cpfor2, swamp et swamp^2 :

Solution

```
model.poly1 <- lm(
  logherp ~ logarea + cpfor2 + thtden + swamp + I(swamp^2),
  data = mydata
)
summary(model.poly1)
```

```
Call:
lm(formula = logherp ~ logarea + cpfor2 + thtden + swamp + I(swamp^2),
    data = mydata)

Residuals:
      Min        1Q    Median        3Q       Max
-0.201797 -0.056170 -0.002072  0.051814  0.205626

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept) -3.203e-01  1.813e-01  -1.766   0.0912 .
logarea      2.202e-01  3.893e-02   5.656 1.09e-05 ***
cpfor2      -7.864e-04  9.955e-04  -0.790   0.4380
thtden      -2.929e-02  1.048e-02  -2.795   0.0106 *
swamp        3.113e-02  5.898e-03   5.277 2.70e-05 ***
I(swamp^2)  -2.618e-04  4.727e-05  -5.538 1.45e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1072 on 22 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.8181,    Adjusted R-squared:  0.7767
F-statistic: 19.78 on 5 and 22 DF,  p-value: 1.774e-07
```

Les résultats de cette analyse suggèrent qu'on devrait probablement exclure `cpfor2` du modèle:

Solution

```r
model.poly2 <- lm(
  logherp ~ logarea + thtden + swamp + I(swamp^2),
  data = mydata
)
summary(model.poly2)
```

```

Call:
lm(formula = logherp ~ logarea + thtden + swamp + I(swamp^2),
    data = mydata)

Residuals:
     Min       1Q   Median       3Q      Max
-0.19621 -0.05444 -0.01202  0.07116  0.21295

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.461e-01  1.769e-01  -1.957   0.0626 .
logarea      2.232e-01  3.842e-02   5.810 6.40e-06 ***
thtden      -2.570e-02  9.364e-03  -2.744   0.0116 *
swamp        2.956e-02  5.510e-03   5.365 1.89e-05 ***
I(swamp^2)  -2.491e-04  4.409e-05  -5.649 9.46e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1063 on 23 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.8129,    Adjusted R-squared:  0.7804
F-statistic: 24.98 on 4 and 23 DF,  p-value: 4.405e-08
```

Est-ce qu'il y a possiblement un problème de multicolinéarité?

```
vif(model.poly2)
```

```
   logarea     thtden      swamp I(swamp^2)
  1.053193   1.123491  45.845845  45.656453
```

Les valeurs d'inflation de la variance (VIF) pour les deux termes de swamp sont beaucoup plus élevés que le seuil de 5. Cependant, c'est la norme pour les termes polynomiaux et on ne doit pas s'en préoccuper outre mesure, surtout quand les deux termes sont hautement significatifs dans le modèle. Les fortes valeurs de VIF indiquent que les coefficients pour ces deux termes ne sont pas estimés précisément, mais leur utilisation dans le modèle permet tout de même de faire de bonnes prédictions (i.e. ils décrivent la réponse à swamp).

7.7. Vérifier les conditions d'application de modèles de régression multiple

Toutes les techniques de sélection des modèles présument que les conditions d'applications (indépendance, normalité, homoscédasticité, linéarité) sont remplies. Comme il y a un grand nombre de modèles qui peuvent être ajustés, il peut paraître quasi impossible de vérifier si les conditions sont remplies à chaque étape de construction. Cependant, il est souvent suffisant d'examiner les résidus du modèle complet (saturé) puis du modèle final. Les termes qui ne contribuent pas significativement à l'ajustement n'affectent pas beaucoup les résidus et donc les résidus du modèle final sont généralement similaires à ceux du modèle complet.

Examinons donc les graphiques diagnostiques du modèle final:

Solution

```
par(mfrow = c(2, 2))
plot(model.poly2)
```

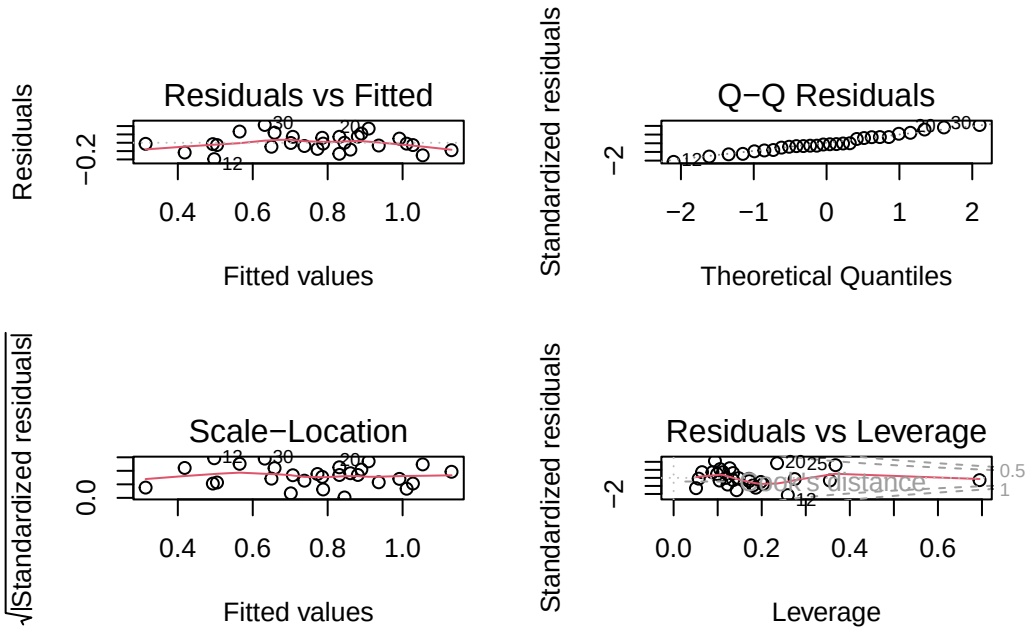


Figure 7.5.: Conditions d'application du modèle `model.poly2`

Tout semble acceptable dans ce cas. Pour convaincre les sceptiques, on peut faire les tests formels des conditions d'application:

```
shapiro.test(residuals(model.poly2))
```

```
    Shapiro-Wilk normality test

data:  residuals(model.poly2)
W = 0.9837, p-value = 0.9278
```

Les résidus ne dévient pas significativement de la normalité. Bien.

```
library(lmtest)
bptest(model.poly2)
```

```
    studentized Breusch-Pagan test

data:  model.poly2
BP = 3.8415, df = 4, p-value = 0.4279
```

Pas de déviation d'homoscédasticité non plus. Bien.

```
dwtest(model.poly2)
```

```
    Durbin-Watson test

data:  model.poly2
DW = 1.725, p-value = 0.2095
alternative hypothesis: true autocorrelation is greater than 0
```

Pas de corrélation sérielle des résidus, donc pas d'évidence de nonindépendance.

```
resettest(model.poly2, type = "regressor", data = mydata)
```

```
    RESET test

data:  model.poly2
RESET = 0.9823, df1 = 8, df2 = 15, p-value = 0.4859
```

Et finalement, pas de déviation significative de la linéarité. Donc tout semble acceptable.

7.8. Visualiser la taille d'effet

Les coefficients de la régression multiple peuvent mesurer la taille d'effet, quoiqu'il puisse être nécessaire de les standardiser pour qu'ils ne soient pas influencés par les unités de mesure. Mais un graphique est souvent plus informatif. Dans ce contexte, les graphiques des résidus partiels (appelés components+residual plots dans R) sont particulièrement utiles. Ces graphique illustrent comment la variable dépendante, corrigée pour l'effet des autres variables dans le modèle, varie avec chacune des variables indépendantes du modèle. Voyons voir:

```
# Evaluate visually linearity and effect size
# component + residual plot
crPlots(model.poly2)
```

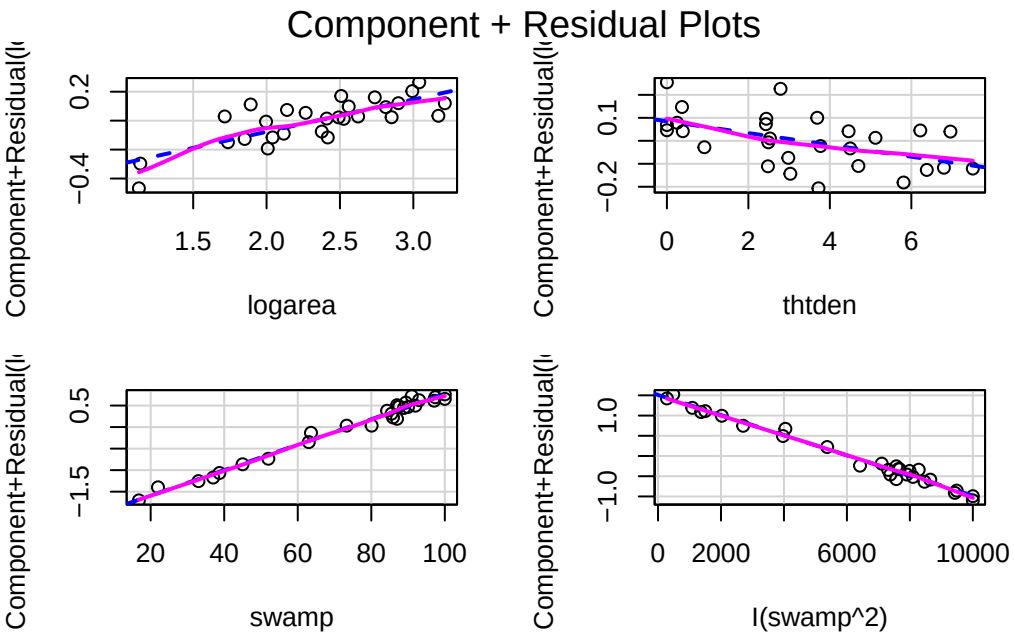


Figure 7.6.: Graphiques de résidus partiels du modèle `model.poly2`

Notez que l'échelle de l'axe des y varie sur chaque graphique. Pour thtden, la variable dépendante (log10(richesse des herptiles)) varie d'environ 0.4 unités entre la valeur minimum et maximum de thtden. Pour logarea, la variation est d'environ 0.6 unité log. Pour swamp, l'interprétation est plus compliquée parce qu'il y a deux termes qui quantifient son effet, et que ces termes ont des signes opposés (positif pour swamp et négatif pour swamp^2) ce qui donne une relation curvilinéaire de type parabole. Le graphique ne permet pas de bien visualiser cela. Ceci dit, ces graphique n'indiquent pas vraiment de violation de linéarité.

Pour illustrer ce qui serait visible sur ces graphiques si il y avait une déviation de linéarité, enlevons le terme du second degré pour swamp, puis on va refaire ces graphiques et effectuer le test RESET.

Solution

```
model.nopoly <- lm(
  logherp ~ logarea + thtden + swamp,
  data = mydata
)
crPlots(model.nopoly)
```

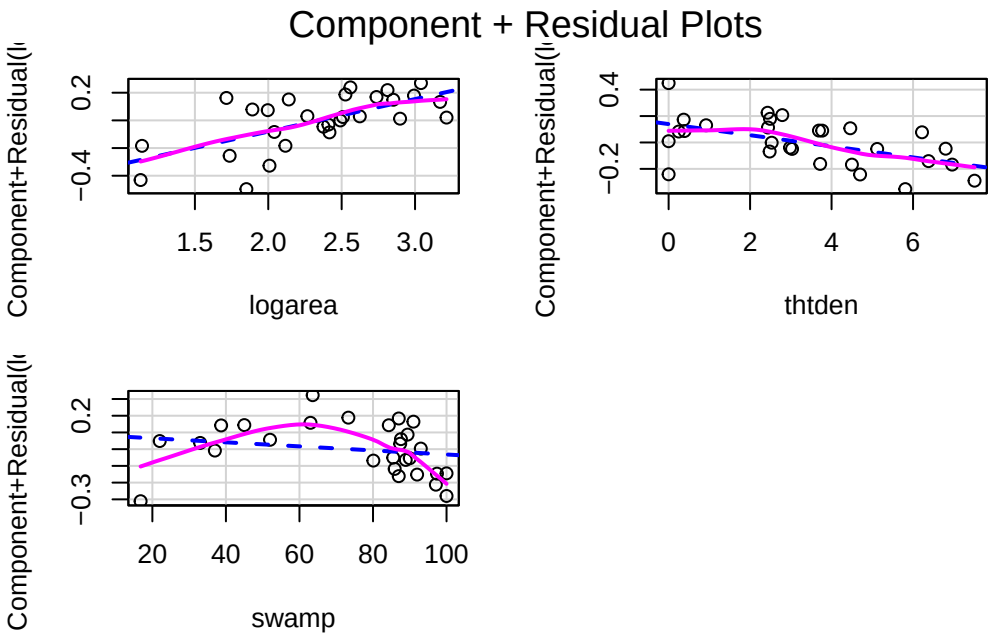


Figure 7.7.: Graphiques de résidus partiels du modèle `model.nopoly`

La relation non-linéaire avec swamp devient évidente. Et le test RESET détecte bien cette non-linéarité:

```
resettest(model.nopoly, type = "regressor")
```

```
    RESET test

data:  model.nopoly
RESET = 6.7588, df1 = 6, df2 = 18, p-value = 0.0007066
```

7.9. Tester la présence d'interactions

Lorsqu'il y a plusieurs variables indépendantes, vous devriez toujours garder à l'esprit la possibilité d'interactions. Dans la majorité des situations de régression multiple cela n'est pas évident parce que l'addition de termes d'interaction augmente la multicolinéarité des termes du modèle, et parce qu'il n'y a souvent pas assez d'observations pour éprouver toutes les interactions ou que les observations ne sont pas suffisamment balancées pour faire des tests puissants pour les interactions. Retournons à notre modèle "final" et voyons ce qui se passe si on essaie d'ajuster un modèle saturé avec toutes les interactions:

```
fullmodel.withinteractions <- lm(
  logherp ~ logarea * cpfor2 * thtden * swamp * I(swamp^2),
  data = mydata
)
summary(fullmodel.withinteractions)
```

```
Call:
lm(formula = logherp ~ logarea * cpfor2 * thtden * swamp * I(swamp^2),
    data = mydata)

Residuals:
ALL 28 residuals are 0: no residual degrees of freedom!

Coefficients: (4 not defined because of singularities)
                                        Estimate Std. Error t value Pr(>|t|)
(Intercept)                           -5.948e+03        NaN     NaN      NaN
logarea                                3.293e+03        NaN     NaN      NaN
cpfor2                                 7.080e+01        NaN     NaN      NaN
thtden                                 9.223e+02        NaN     NaN      NaN
swamp                                  1.176e+02        NaN     NaN      NaN
I(swamp^2)                            -3.517e-01        NaN     NaN      NaN
logarea:cpfor2                        -3.771e+01        NaN     NaN      NaN
logarea:thtden                        -4.781e+02        NaN     NaN      NaN
cpfor2:thtden                         -1.115e+01        NaN     NaN      NaN
logarea:swamp                         -7.876e+01        NaN     NaN      NaN
cpfor2:swamp                          -1.401e+00        NaN     NaN      NaN
thtden:swamp                          -1.920e+01        NaN     NaN      NaN
logarea:I(swamp^2)                     5.105e-01        NaN     NaN      NaN
cpfor2:I(swamp^2)                      3.825e-03        NaN     NaN      NaN
thtden:I(swamp^2)                      7.826e-02        NaN     NaN      NaN
swamp:I(swamp^2)                      -2.455e-03        NaN     NaN      NaN
logarea:cpfor2:thtden                  5.359e+00        NaN     NaN      NaN
logarea:cpfor2:swamp                   8.743e-01        NaN     NaN      NaN
logarea:thtden:swamp                   1.080e+01        NaN     NaN      NaN
cpfor2:thtden:swamp                    2.620e-01        NaN     NaN      NaN
logarea:cpfor2:I(swamp^2)             -5.065e-03        NaN     NaN      NaN
logarea:thtden:I(swamp^2)             -6.125e-02        NaN     NaN      NaN
cpfor2:thtden:I(swamp^2)              -1.551e-03        NaN     NaN      NaN
logarea:swamp:I(swamp^2)              -4.640e-04        NaN     NaN      NaN
cpfor2:swamp:I(swamp^2)                3.352e-05        NaN     NaN      NaN
thtden:swamp:I(swamp^2)                2.439e-04        NaN     NaN      NaN
logarea:cpfor2:thtden:swamp           -1.235e-01        NaN     NaN      NaN
logarea:cpfor2:thtden:I(swamp^2)       7.166e-04        NaN     NaN      NaN
logarea:cpfor2:swamp:I(swamp^2)               NA         NA      NA       NA
logarea:thtden:swamp:I(swamp^2)               NA         NA      NA       NA
cpfor2:thtden:swamp:I(swamp^2)                NA         NA      NA       NA
logarea:cpfor2:thtden:swamp:I(swamp^2)        NA         NA      NA       NA

Residual standard error: NaN on 0 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:      1, Adjusted R-squared:    NaN
F-statistic:   NaN on 27 and 0 DF,  p-value: NA
```

Notez les coefficients manquants aux dernières lignes: on ne peut inclure les 32 termes si on a seulement 28 observations. Il manque des observations, le R carré est 1, et le modèle “prédit” parfaitement les

données.

Si on essaie une méthode automatique pour identifier le "meilleur" modèle dans ce gâchis, R refuse:

```
step(fullmodel.withinteractions)
```

```
Error in step(fullmodel.withinteractions): AIC is -infinity for this model, so 'step' cannot proc
```

Bon, est-ce qu'on oublie tout ça et qu'on accepte le modèle final sans ce soucier des interactions? Non, pas encore. Il y a un compromis possible: comparer notre modèle "final" à un modèle qui contient au moins un sous-ensemble des interactions, par exemple toutes les interactions du second degré, pour éprouver si l'addition de ces interactions améliore beaucoup l'ajustement du modèle.

```
full.model.2ndinteractions <- lm(
  logherp ~ logarea + cpfor2 + thtden + swamp + I(swamp^2)
    + logarea:cpfor2
    + logarea:thtden
    + logarea:swamp
    + cpfor2:thtden
    + cpfor2:swamp
    + thtden:swamp,
  data = mydata
)
summary(full.model.2ndinteractions)
```

```
Call:
lm(formula = logherp ~ logarea + cpfor2 + thtden + swamp + I(swamp^2) +
    logarea:cpfor2 + logarea:thtden + logarea:swamp + cpfor2:thtden +
    cpfor2:swamp + thtden:swamp, data = mydata)

Residuals:
      Min        1Q    Median        3Q       Max
-0.216880 -0.036534  0.003506  0.042990  0.175490

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     4.339e-01  6.325e-01   0.686 0.502581
logarea        -1.254e-01  2.684e-01  -0.467 0.646654
cpfor2         -9.344e-03  7.205e-03  -1.297 0.213032
thtden         -1.833e-01  9.035e-02  -2.028 0.059504 .
swamp           3.569e-02  7.861e-03   4.540 0.000334 ***
I(swamp^2)     -3.090e-04  7.109e-05  -4.347 0.000500 ***
logarea:cpfor2  2.582e-03  2.577e-03   1.002 0.331132
logarea:thtden  7.017e-02  3.359e-02   2.089 0.053036 .
logarea:swamp  -5.290e-04  2.249e-03  -0.235 0.816981
```

```
cpfor2:thtden  -2.095e-04  6.120e-04  -0.342 0.736544
cpfor2:swamp    4.651e-05  5.431e-05   0.856 0.404390
thtden:swamp    2.248e-04  4.764e-04   0.472 0.643336
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.108 on 16 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.8658,    Adjusted R-squared:  0.7735
F-statistic: 9.382 on 11 and 16 DF,  p-value: 4.829e-05
```

Ce modèle s'ajuste un peu mieux aux données que les modèle "final" (il explique 86.6% de la variance de logherp, comparé à 81.2% pour le modèle "final" sans interactions), mais il compte deux fois plus de paramètres. De plus, si vous examinez les coefficients, il se passe d'étranges choses: le signe pour logare a changé par exemple. C'est un des symptômes de la multicolinéarité. Allons voir les facteurs d'inflation de la variance:

```
vif(full.model.2ndinteractions)
```

```
there are higher-order terms (interactions) in this model
consider setting type = 'predictor'; see ?vif
```

```
       logarea         cpfor2         thtden          swamp     I(swamp^2)
      49.86060       78.49622      101.42437       90.47389      115.08457
logarea:cpfor2 logarea:thtden  logarea:swamp  cpfor2:thtden   cpfor2:swamp
      66.97792       71.69894       67.27034       14.66814       29.41422
  thtden:swamp
      20.04410
```

Aie! tous les VIF sont plus grands que 5, pas seulement les termes incluant swamp. Cette forte multicolinéarité empêche de quantifier avec précision l'effet de ces interactions. De plus, ce modèle avec interactions n'est pas plus informatif que le modèle "final" puisque son AIC est plus élevé (souvenez-vous qu'on privilégie le modèle avec la valeur d'AIC la plus basse):

```
AIC(model.poly1)
```

```
[1] -38.3433
```

```
AIC(full.model.2ndinteractions)
```

```
[1] -34.86123
```

On peut également utiliser la fonction anova() pour comparer l'ajustement des deux modèles et vérifier si l'addition des termes d'intération améliore significativement l'ajustement:

```
anova(model.poly1, full.model.2ndinteractions)
```

```
Analysis of Variance Table

Model 1: logherp ~ logarea + cpfor2 + thtden + swamp + I(swamp^2)
Model 2: logherp ~ logarea + cpfor2 + thtden + swamp + I(swamp^2) + logarea:cpfor2 +
    logarea:thtden + logarea:swamp + cpfor2:thtden + cpfor2:swamp +
    thtden:swamp
  Res.Df     RSS Df Sum of Sq      F Pr(>F)
1     22 0.25282
2     16 0.18651  6  0.066314 0.9481  0.489
```

Ici, l'addition des termes d'interaction ne réduit pas significativement la variabilité résiduelle du modèle "complet". Qu'en est-il de la si on compare le modèle avec interaction et notre modèle "final"?

```
anova(model.poly2, full.model.2ndinteractions)
```

```
Analysis of Variance Table

Model 1: logherp ~ logarea + thtden + swamp + I(swamp^2)
Model 2: logherp ~ logarea + cpfor2 + thtden + swamp + I(swamp^2) + logarea:cpfor2 +
    logarea:thtden + logarea:swamp + cpfor2:thtden + cpfor2:swamp +
    thtden:swamp
  Res.Df     RSS Df Sum of Sq      F Pr(>F)
1     23 0.25999
2     16 0.18651  7  0.073486 0.9006 0.5294
```

Le test indique que ces deux modèles ont des variances résiduelles comparables, et donc que l'addition des termes d'interaction et de cpfor2 au modèle final n'apporte pas grand chose.

7.10. Recherche du meilleur modèle fondée sur la théorie de l'information

Une des principales critiques des méthodes pas-à-pas (stepwise) est que les valeurs de p ne sont pas strictement interprétables à cause du grand nombre de tests qui sont implicites dans le processus. C'est le problème des comparaisons ou tests multiples: en construisant un modèle linéaire (comme une régression multiple) à partir d'un grand nombre de variables et de leurs interactions, il y a tellement de combinaisons possibles qu'un ajustement de Bonferroni rendrait les tests trop conservateurs.

Une alternative, défendue par Burnham et Anderson (2002, Model selection and multimodel inference: a practical information-theoretic approach. 2nd ed), est d'utiliser l'AIC (ou mieux encore AICc qui est plus approprié quand le nombre d'observations est inférieur à 40 fois le nombre de variables indépendantes) pour ordonner les modèles et identifier un sousensemble de modèles qui sont les meilleurs. On peut ensuite

calculer les moyennes des coefficients pondérées par la probabilité que chacun des modèles soit le meilleur pour obtenir des coefficients qui sont plus robustes et moins sensibles à la multicolinéarité.

L'approche de comparaison par *AIC* a d'abord été développé pour comparer un ensemble de modèle préalablement défini basé sur les connaissance du sytème et les hypothèses biologiques. Cependant, certains ont développé une approche plutôt brutale et sans scrupule de faire tous les modèles possibles et de les comparer par *AIC*. Cette approche a été suivie dans le package `MuMIn`. Les comparaisons de modèle par `AIC`doivent être faites en utilisant exactement le même jeu de données pour chaque modèle. Il faut donc s'arrurer d'enlever les données manquantes et de spécifier dans la fonction `lm` de ne pas marcher s'il y a des données manquantes.

> **Note**
>
> Je ne supporte pas l'approche stepwise ni l'approche par AIC. Je déteste l'approche par la fonction `dredge()` qui selon moi va à l'encontre de la philosophie des AIC et de la parsimonie. Je soutiens de dévelooper un modèle basé sur des hypothèses biologiques et de reporter ce modèle avec tous les effets significatifs ou non.

```
# refaire le modèle en s'assurant qu'il n'y a pas de "NA" et en spécificant na.action
full.model.2ndinteractions <- update(
  full.model.2ndinteractions,
  . ~ .,
  data = mysub,
  na.action = "na.fail"
)

library(MuMIn)
dd <- dredge(full.model.2ndinteractions)
```

```
Fixed term is "(Intercept)"
```

L'objet `dd` contient tous les modèles possibles (i.e. ceux qui ont toutes les combinaisons possibles) en utilisant les termes du modèle full.model.2ndinteractions ajusté précédemment. On peut ensuite extraire de l'objet dd le sous-ensemble de modèles qui ont un AICc semblable au meilleur modèle (Burnham et Anderson suggèrent que les modèles qui dévient par plus de 7 unités d'AICc du meilleur modèle ont peu de support empirique).

```
# get models within 2 units of AICc from the best model
top.models.1 <- get.models(dd, subset = delta < 2)
avgmodel1 <- model.avg(top.models.1) # compute average parameters
summary(avgmodel1) # display averaged model
```

```
Call:
model.avg(object = top.models.1)
```

```
Component model call:
lm(formula = logherp ~ <2 unique rhs>, data = mysub, na.action =
     na.fail)

Component models:
      df logLik   AICc delta weight
12345  7  27.78 -35.95  0.00   0.55
1234   6  25.78 -35.56  0.39   0.45

Term codes:
    I(swamp^2)        logarea          swamp         thtden logarea:thtden
             1              2              3              4              5

Model-averaged coefficients:
(full average)
                 Estimate Std. Error Adjusted SE z value Pr(>|z|)
(Intercept)    -2.145e-01  2.308e-01   2.406e-01   0.891    0.373
logarea         1.356e-01  1.089e-01   1.119e-01   1.213    0.225
swamp           3.180e-02  5.971e-03   6.273e-03   5.070    4e-07 ***
I(swamp^2)     -2.669e-04  4.770e-05   5.011e-05   5.326    1e-07 ***
thtden         -6.985e-02  5.233e-02   5.361e-02   1.303    0.193
logarea:thtden  2.131e-02  2.487e-02   2.545e-02   0.837    0.403

(conditional average)
                 Estimate Std. Error Adjusted SE z value Pr(>|z|)
(Intercept)    -2.145e-01  2.308e-01   2.406e-01   0.891   0.3727
logarea         1.356e-01  1.089e-01   1.119e-01   1.213   0.2253
swamp           3.180e-02  5.971e-03   6.273e-03   5.070    4e-07 ***
I(swamp^2)     -2.669e-04  4.770e-05   5.011e-05   5.326    1e-07 ***
thtden         -6.985e-02  5.233e-02   5.361e-02   1.303   0.1927
logarea:thtden  3.882e-02  2.114e-02   2.237e-02   1.735   0.0827 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
confint(avgmodel1) # display CI for averaged coefficients
```

```
                      2.5 %       97.5 %
(Intercept)    -0.6860022996  0.257064603
logarea        -0.0836067896  0.354883299
swamp           0.0195105703  0.044099316
I(swamp^2)     -0.0003650809 -0.000168656
thtden         -0.1749296690  0.035236794
logarea:thtden -0.0050266778  0.082666701
```

1. La liste des modèles qui sont à 4 unités ou moins de l'AICc du meilleur modèle. Les variables dans chaque modèle sont codées et on retrouve la clé en dessous du tableau.

2. Pour chaque modèle, en plus de l'AICc, le poids Akaike est calculé. C'est un estimé de la probabilité que ce modèle est le meilleur. Ici on voit que le premier modèle (le meilleur) a seulement 34% des chance d'être vraiment le meilleur.
3. À partir de ce sous-ensemble de modèles, la moyenne pondérée des coefficients (en utilisant les poids Akaike) est calculée, avec in IC à 95%. Notez que les termes absents d'un modèle sont considérés avoir un coefficient de 0 pour ce terme.

7.11. Bootstrap et régression multiple

Quand les données ne rencontrent pas les conditions d'application de normalité et d'homoscédasticité et que les transformations n'arrivent pas à corriger ces violations, le bootstrap peut être utilisé pour calculer des intervalles de confiance pour les coefficients. Si la distribution des coefficients bootstrappés est symétrique et approximativement normale, on peut utiliser les percentiles empiriques pour calculer les limites de confiance.

Le code qui suit, utilisant le package `simpleboot`, a été conçu pour être facilement modifiable et calcule les limites des IC à partir des percentiles empiriques.

```
############################################################
#######
# Bootstrap analysis the simple way with library simpleboot
# Define model to be bootstrapped and the data source used
mymodel <- lm(logherp ~ logarea + thtden + swamp + I(swamp^2), data = mydata)
# Set the number of bootstrap iterations
nboot <- 1000
library(simpleboot)
# R is the number of bootstrap iterations
# Setting rows to FALSE indicates resampling of residuals
mysimpleboot <- lm.boot(mymodel, R = nboot, rows = FALSE)
# Extract bootstrap coefficients
myresults <- sapply(mysimpleboot$boot.list, function(x) x$coef)
# Transpose matrix so that lines are bootstrap iterations and columns are coefficients
tmyresults <- t(myresults)
```

Vous pouvez ensuite faire des graphiques pour voir les résultats. Lorsque vous tournerez ce code, il y aura une pause pour vous permettre d'examiner la distribution pour chaque coefficient du modèle sur des graphiques:

```
# Plot histograms of bootstrapped coefficients
ncoefs <- length(data.frame(tmyresults))
par(mfrow = c(1, 2), mai = c(0.5, 0.5, 0.5, 0.5), ask = TRUE)
for (i in 1:ncoefs) {
  lab <- colnames(tmyresults)[i]
  x <- tmyresults[, i]
  plot(density(x), main = lab, xlab = "")
  abline(v = mymodel$coef[i], col = "red")
```

```
  abline(v = quantile(x, c(0.025, 0.975)))
  hist(x, main = lab, xlab = "")
  abline(v = quantile(x, c(0.025, 0.975)))
  abline(v = mymodel$coef[i], col = "red")
}
```

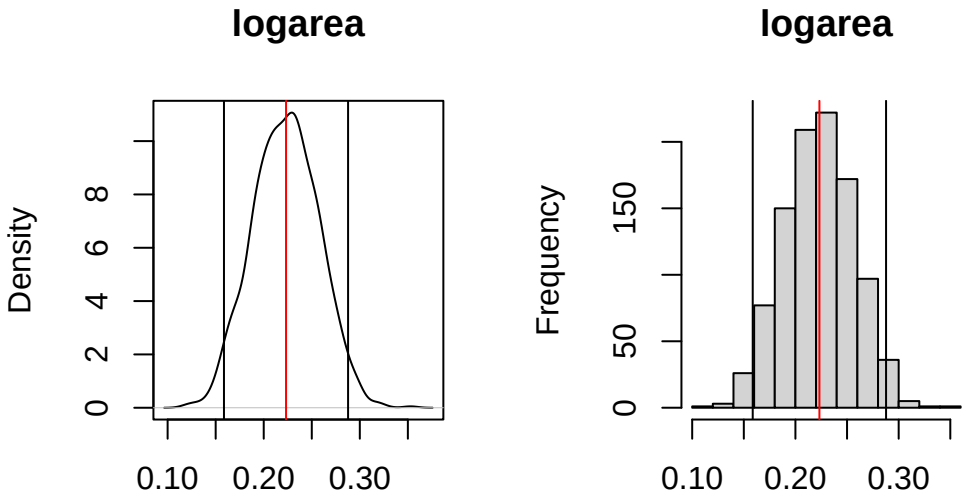


Figure 7.8.: Distribution des estimé par bootstrap pour `logarea`

Le graphique de droite illustre la densité lissée (kernel density) et celui de gauche est l'histogramme des estimés bootstrap du coefficient. La ligne rouge sur le graphique indique la valeur du coefficient ordinaire (pas bootstrap) et les deux lignes verticales noires marquent les limites de l'intervalle de confiance à 95%. Ici l'IC ne contient pas 0, et donc on peut conclure que l'effet de logarea sur logherp est significativement positif.

Les limites précises peuvent être obtenues par:

```
# Display empirical bootstrap quantiles (not corrected for bias)
p <- c(0.005, 0.01, 0.025, 0.05, 0.95, 0.975, 0.99, 0.995)
apply(tmyresults, 2, quantile, p)
```

```
      (Intercept)   logarea       thtden      swamp    I(swamp^2)
0.5%  -0.70550322 0.1442407 -0.047779168 0.01625423 -0.0003465268
1%    -0.68144031 0.1509525 -0.045588841 0.01776112 -0.0003411190
2.5%  -0.63946155 0.1586440 -0.042294775 0.01948596 -0.0003261057
5%    -0.60577998 0.1668945 -0.039841585 0.02126944 -0.0003176463
95%   -0.08008776 0.2778039 -0.012125998 0.03814283 -0.0001830763
97.5% -0.02568194 0.2877844 -0.010144740 0.03926291 -0.0001699843
```

```
99%    0.03955076 0.2983717 -0.005979965 0.04078311 -0.0001529431
99.5%  0.07723605 0.3024344 -0.003992079 0.04159421 -0.0001447103
```

Ces intervalles de confiances ne sont pas fiables si la distribution des estimés bootstrap n'est pas Gaussienne. Dans ce cas, il vaut mieux calculer des coefficients non-biaisés (bias-corrected accelerated confidence limits, BCa):

```
##################################################
# Bootstrap analysis in multiple regression with BCa confidence intervals
# Preferable when parameter distribution is far from normal
# Bootstrap 95% BCa CI for regression coefficients

library(boot)
# function to obtain regression coefficients for each iteration
bs <- function(formula, data, indices) {
  d <- data[indices, ] # allows boot to select sample
  fit <- lm(formula, data = d)
  return(coef(fit))
}
# bootstrapping with 1000 replications
results <- boot(
  data = mydata, statistic = bs, R = 1000,
  formula = logherp ~ logarea + thtden + swamp + I(swamp^2)
)
# view results
```

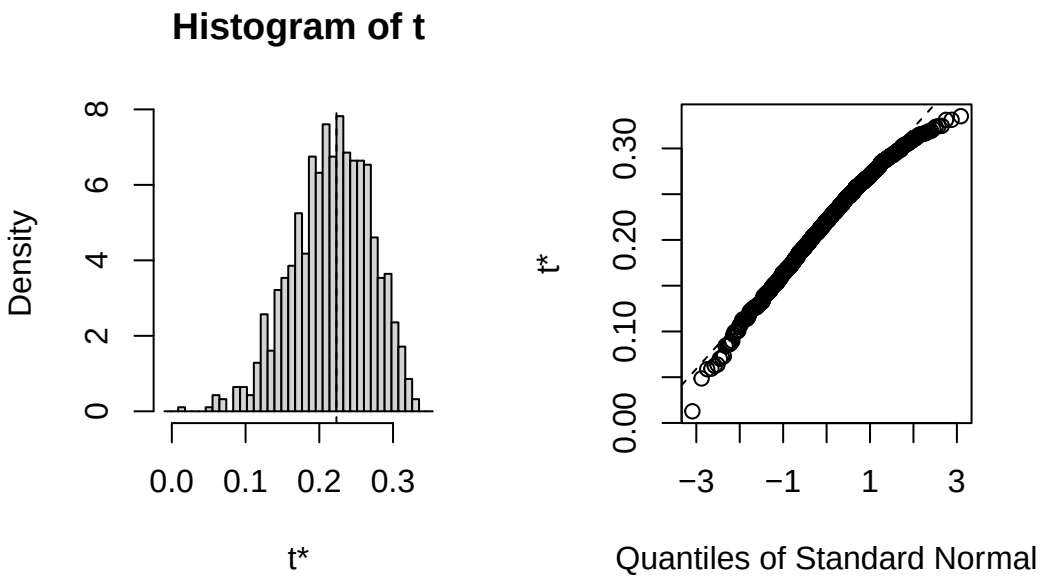
Pour obtenir les résultats, le code suivant va produire le graphique standard pour chaque coefficient, et les estimés BCa pour l'intervalle de confiance

```
plot(results, index = 1) # intercept
plot(results, index = 2) # logarea
plot(results, index = 3) # thtden
plot(results, index = 4) # swamp
plot(results, index = 5) # swamp^2

# get 95% confidence intervals
boot.ci(results, type = "bca", index = 1)
boot.ci(results, type = "bca", index = 2)
boot.ci(results, type = "bca", index = 3)
boot.ci(results, type = "bca", index = 4)
boot.ci(results, type = "bca", index = 5)
```

Pour logarea, cela donne:

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates
```

```
CALL :
boot.ci(boot.out = results, type = "bca", index = 2)

Intervals :
Level       BCa
95%   ( 0.1235,  0.3161 )
Calculations and Intervals on Original Scale
```



Notez que l'intervalle BCa va de 0.12 à 0.32, alors que l'intervalle standard était de 0.16 à 0.29. L'intervalle BCa est ici plus grand du côté inférieur et plus petit du côté supérieur comme il se doit compte tenu de la distribution non-Gaussienne et asymétrique des estimés bootstrap.

7.12. Test de permutation

Les tests de permutations sont plus rarement effectués en régression multiple que le bootstrap. Voici un fragment de code pour le faire tout de même.

```
############################################################
##########
# Permutation in multiple regression
#
# using lmperm library
library(lmPerm)
# Fit desired model on the desired dataframe
mymodel <- lm(logherp ~ logarea + thtden + swamp + I(swamp^2),
  data = mydata
```

```
)
mymodelProb <- lmp(
  logherp ~ logarea + thtden + swamp + I(swamp^2),
  data = mydata, perm = "Prob"
)
summary(mymodel)
summary(mymodelProb)
```

8. ANCOVA et modèle linéaire général

Après avoir complété cet exercice de laboratoire, vous devriez pouvoir:

- Utiliser R pour faire une analyse de covariance (ANCOVA) et ajuster des modèles qui ont des variables indépendantes continues et discontinues (modèle linéaire général)
- Utiliser R pour vérifier les conditions préalables à ces modèles
- Utiliser R pour comparer l'ajustement de modèles statistiques
- Utiliser R pour faire des tests de bootstrap et de permutation sur des modèles avec des variables indépendantes continues et discontinues.

8.1. Paquets et données requises pour le labo

Ce laboratoire nécessite:

- les paquets R:
  - ggplot2
  - car
  - lmtest
- les fichiers de données
  - anc1dat.csv
  - anc3dat.csv

8.2. Modèle linéaire général

Les modèles linéaires généraux ou General Linear Model en anglais sont différent des modèles linéaires généralisés (ou generalized linear model, GLM). Les modèles linéaires généraux sont des modèles statistiques de la forme $Y = B\mathbf{X} + E$, ou Y est un vecteur contenant la variable dépendante continue, B est un vecteur des paramètres estimés, $\mathbf{X}$ et la matrice des différents variables indépendantes et E est un vecteur de résidus homoscédastiques et normalement distribués. Tous les tests que nous avons étudiés à date (test de t, régression linéaire simple, ANOVA à un facteur de classification, ANOVA à plusieurs facteurs de classification et régression multiple) sont formulés ainsi. Notez que tous les modèles que nous avons rencontrés à ce jour ne contiennent qu'un type de variable indépendante (soit continue ou discontinue). Dans cet exercice de laboratoire, vous allez ajuster des modèles qui ont les deux types de variables indépendantes.

8.3. ANCOVA

ANCOVA est l'abréviation pour l'analyse de covariance. C'est un type de modèle linéaire général dans lequel il y a une (ou plusieurs) variable indépendante continue (parfois appelé la covariable) et une (ou plusieurs) variable indépendante discontinue. Dans la présentation traditionnelle de l'ANCOVA dans les manuels de biostatistique, le modèle ANCOVA ne contient pas de termes d'interaction entre les variables continues et discontinues. Par conséquent, on doit précéder l'ajustement de ce modèle (réduit parce que sans terme d'interaction), par un test de signification de l'interaction qui correspond à éprouver l'égalité des pentes (coefficients pour la ou les variables continues) entre les différents niveaux de la ou les variables discontinues (i.e un test d'homogénéité des pentes).

8.4. Homogénéité des pentes

Pour répondre à de nombreuses questions biologiques, il est nécessaire de déterminer si deux (ou plus de deux) régressions diffèrent significativement. Par exemple, pour comparer l'efficacité de deux insecticides on doit comparer la relation entre leur dose et la mortalité. Ou encore, pour comparer le taux de croissance des mâles et des femelles on doit comparer la relation entre la taille et l'âge des mâles et des femelles.

Comme chaque régression linéaire est décrite par deux paramètres, la pente et l'ordonnée à l'origine, on doit considérer les deux dans la comparaison. Le modèle d'ANCOVA, à strictement parler, n'éprouve que l'hypothèse d'égalité des ordonnées à l'origine. Cependant, avant d'ajuster ce modèle, il faut éprouver l'hypothèse d'égalité des pentes (homogénéité des pentes).

8.4.1. Cas 1 - La taille en fonction de l'âge (exemple avec pente commune)

> **Exercice**
>
> En utilisant les données du fichier `anc1dat.csv`, éprouvez l'hypothèse que le taux de croissance des esturgeons mâles et femelles de The Pas est le même (données de 1978-1980). Comme mesure du taux de croissance, nous allons utiliser la pente de la régression du log 10 de la longueur à la fourche, lfkl, sur le log 10 de l'âge, l'age.

Commençons par examiner les données. Pour faciliter la comparaison, il serait utile de tracer la droite de régression et la trace lowess pour ainsi plus facilement évaluer la linéarité. On peut aussi ajouter un peu de trucs R pour obtenir des légendes plus complètes (remarquez l'utilisation de la commande expression() pour obtenir des indices):

```
anc1dat <- read.csv("data/anc1dat.csv")
anc1dat$sex <- as.factor(anc1dat$sex)
myplot <- ggplot(data = anc1dat, aes(x=lage,    y=log10(fklngth)))+facet_grid(.~sex)+geom_point()
myplot <- myplot+
  stat_smooth(method = lm, se=FALSE)+
  stat_smooth(se=FALSE, color="red") +
  labs(
    y = expression(log[10]~(Fork~length)),
    x = expression(log[10]~(Age))
```

```
)
myplot
```

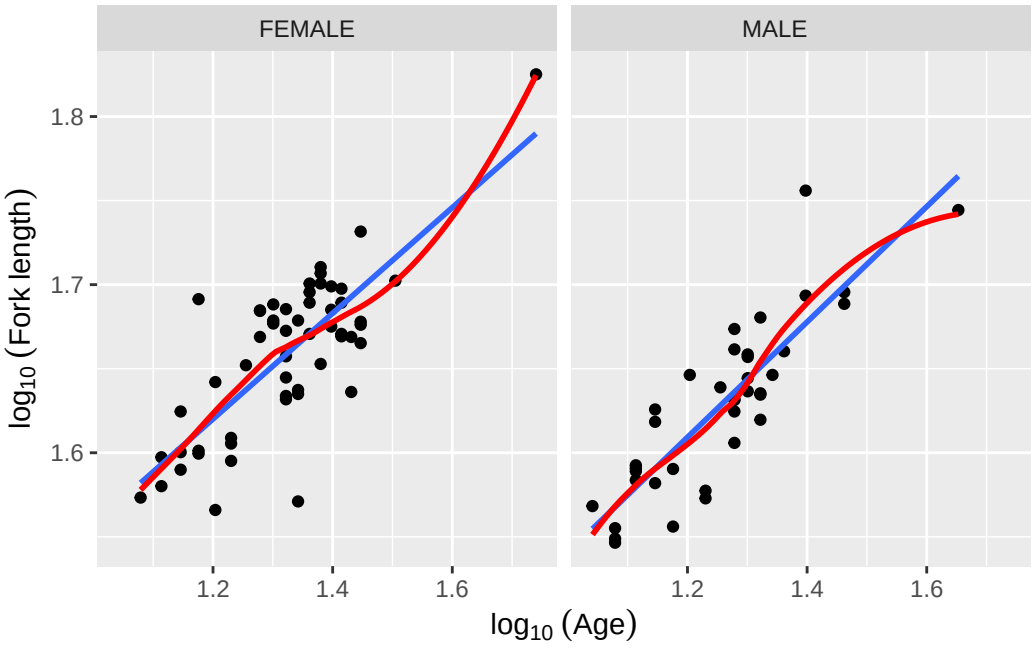


Figure 8.1.: Longueur des esturgeons en fonction de l'age

La transformation log-log rend la relation linéaire et, à première vue, il ne semble pas y avoir de problème évident avec les conditions d'application. Ajustons donc le modèle complet avec l'interaction:

```
model.full1<-lm(lfkl ~ sex + lage + sex:lage, data = anc1dat)
Anova(model.full1, type = 3)
```

```
Anova Table (Type III tests)

Response: lfkl
             Sum Sq Df  F value    Pr(>F)
(Intercept) 0.64444  1 794.8182 < 2.2e-16 ***
sex         0.00041  1   0.5043    0.4795
lage        0.07259  1  89.5312 4.588e-15 ***
sex:lage    0.00027  1   0.3367    0.5632
Residuals   0.07135 88
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Probabilité que le terme lage*sex n'affecte pas la longueur à la fourche (i.e. que la pente ne diffère pas entre les sexes, et que la différence de taille entre les mâles et femelles ne varie pas avec l'âge)

Attrape. Notez que j'ai utilisé la fonction Anova() du package car avec un "a" majuscule au lieu de la fonction native anova() (avec un "a" minuscule") associée aux objets produits par lm() pour obtenir les sommes de carrés de type III. Ces sommes des carrés des écarts de type III (partiels) sont calculées comme si la variable était entrée la dernière dans le modèle et correspondent à la différence entre la variance expliquée par le modèle complet et par le modèle dans lequel seule cette variable est omise. La fonction native anova() donne les sommes des carrés séquentielles, calculées au fur et à mesure que chaque variable est ajoutée au modèle nul avec seulement une ordonnée à l'origine. Dans de rares cas, les sommes des carrés de type I et III sont égales (quand le design est parfaitement orthogonal ou balancé). Dans la vaste majorité des cas, les sommes des carrés de type I et III sont différentes et je vous conseille de toujours utiliser les sommes des carrés de type III dans vos analyses.

À partir de cette analyse, on devrait accepter les hypothèses nulles (1) d'égalité des pentes pour les deux sexes, et (2) que les ordonnées à l'origine sont les mêmes pour les deux sexes. Mais, avant d'accepter ces conclusions, il faut vérifier si les données rencontrent les conditions d'application, comme d'habitude…

Solution

```
par(mfrow = c(2,2))
plot(model.full1)
```

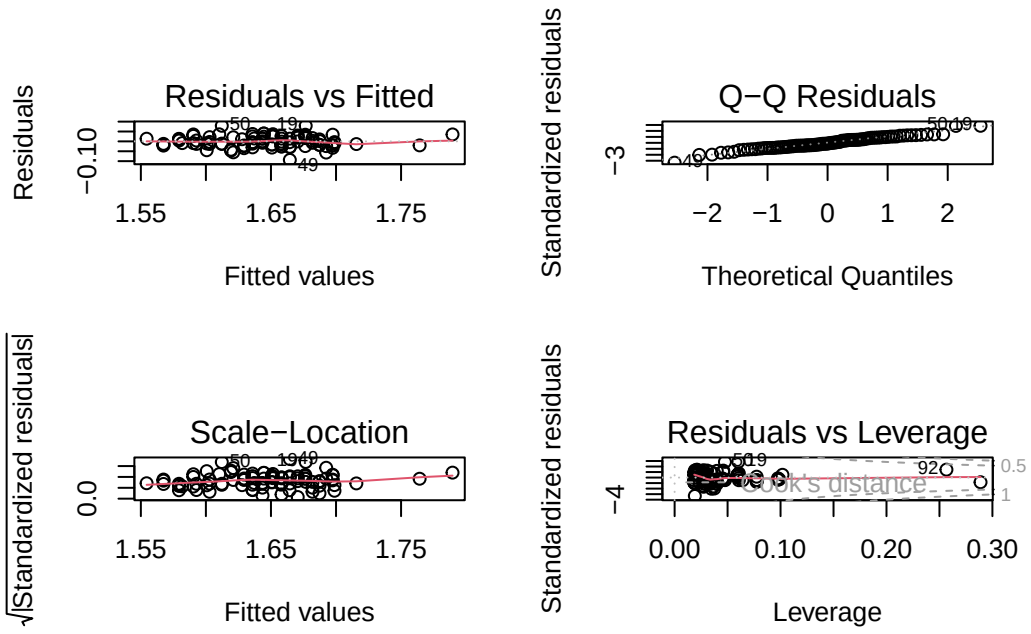


Figure 8.2.: Conditions d'applications du modèle `model.full1`

En ce qui concerne la normalité, ça a l'air d'aller quoiqu'il y a quelques points, en haut à droite, qui dévient de la droite. Si on effectue le test de Wilk-Shapiro (W = .9764, p = 0.09329), on confirme que les résidus ne dévient pas significativement de la normalité. Les résidus semblent homoscédastiques, mais si vous voulez vous en assurer, vous pouvez l'éprouver par un des tests formels. Ici j'utilise le test Breusch-Pagan, qui est approprié quand certaines des variables indépendantes sont continues (Le test de Levene n'est approprié que lorsqu'il n'y a que des variables discontinues).

```
bptest(model.full1)
```

```
    studentized Breusch-Pagan test

data:  model.full1
BP = 0.99979, df = 3, p-value = 0.8013
```

Comme l'hypothèse nulle de ce test est que les résidus sont homoscédastiques, et que p est relativement élevé, le test confirme l'évaluation visuelle. De plus, il n'y a pas de tendance évidente dans les résidus, suggérant qu'il n'y a pas de problème de linéarité. Ce qui peut également être éprouvé formellement:

```
resettest(model.full1, power = 2:3, type = "regressor", data = anc1dat)
```

```
    RESET test

data:  model.full1
RESET = 0.59861, df1 = 2, df2 = 86, p-value = 0.5519
```

La dernière condition d'application est qu'il n'y a pas d'erreur de mesure sur la variable indépendante continue. On ne peut vraiment éprouver cette condition„ mais on sait que des estimés indépendants de l'âge des poissons obtenus par différents chercheurs donnent des âges qui concordent avec moins de 1-2 ans d'écart., ce qui est inférieur au 10% de la fourchette observée des âges et donc acceptable pour des analyses de modèles de type I (attention ici on ne parle pas des SC de type I, je sais, c'est facile de confondre…)

Vous noterez qu'il y a une observation qui a un résidu normalisé (studentized residual) qui est élevé, i.e. une valeur extrême (cas numéro 49). Éliminez-la de l'ensemble de données et refaites l'analyse. Vos conclusions changent-elles?

```
model.full.no49<-lm(lfkl ~ sex + lage + sex:lage, data = anc1dat[c(-49),])
Anova(model.full.no49, type=3)
```

```
Anova Table (Type III tests)

Response: lfkl
             Sum Sq Df  F value Pr(>F)
(Intercept) 0.64255  1 895.9394 <2e-16 ***
sex         0.00038  1   0.5273 0.4697
lage        0.07378  1 102.8746 <2e-16 ***
sex:lage    0.00022  1   0.3135 0.5770
Residuals   0.06239 87
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La conclusion ne change pas après avoir enlevé la valeur extrême. Comme on n'a pas de bonne raison d'éliminer cette valeur, il est probablement mieux de la conserver. Un examen des conditions d'application après avoir enlevé cette valeur révèle qu'elles sont toutes rencontrées.

8.4.2. Cas 2 - Taille en fonction de l'âge (exemple avec des pentes différentes)

> Exercice
>
> Le fichier `anc3dat.csv` contient des données sur des esturgeons mâles de deux sites (locate) : Lake of the Woods dans le Nord-Ouest de l'Ontario et Chruchill River dans le Nord du Manitoba. En utilisant la même procédure, éprouvez l'hypothèse que la pente de la régression de lfkl sur lage est la même aux deux sites (alors Locate est la variable en catégories et non pas sex). Que concluez-vous?

```
anc3dat <- read.csv("data/anc3dat.csv")
myplot <- ggplot(data = anc3dat, aes(x=lage, y = log10(fklngth))) +
  facet_grid(.~locate) +
  geom_point() +
  stat_smooth(method = lm, se=FALSE)+
  stat_smooth(se=FALSE, color="red")+
  labs(
    y = expression(log[10]~(Fork~length)),
    x = expression(log[10]~(Age))
)
myplot
```

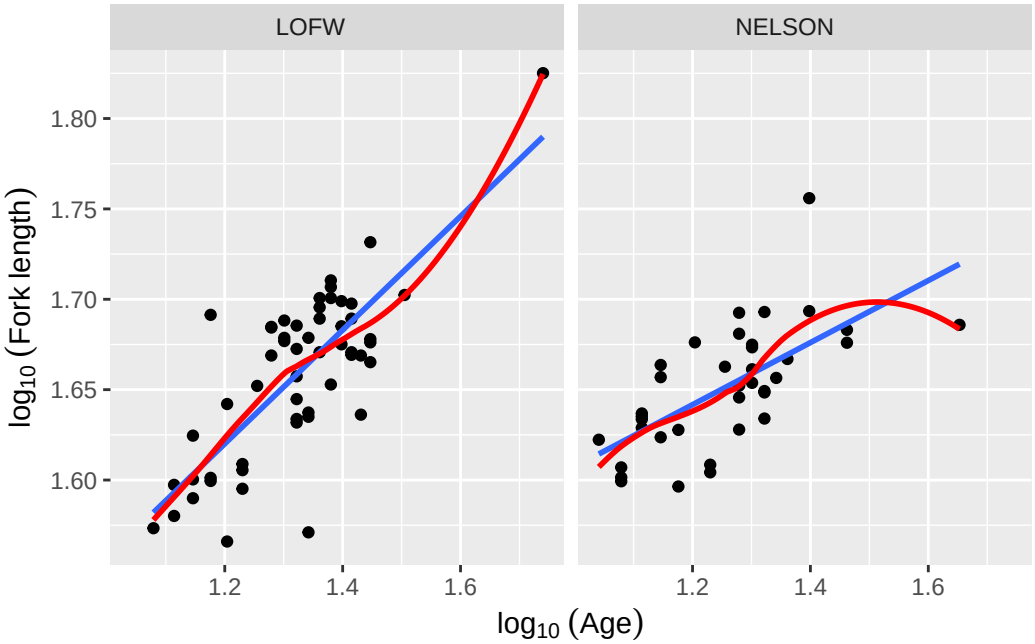


Figure 8.3.: Longueur des esturgeons en fonction de l'age d'après `anc3dat`

```
model.full2<-lm(lfkl ~ lage + locate + lage:locate, data = anc3dat)
Anova(model.full2, type = 3)
```

```
Anova Table (Type III tests)

Response: lfkl
             Sum Sq Df  F value    Pr(>F)
(Intercept) 0.62951  1 1078.632 < 2.2e-16 ***
lage        0.07773  1  133.185 < 2.2e-16 ***
locate      0.00968  1   16.591 0.0001012 ***
lage:locate 0.00909  1   15.575 0.0001592 ***
Residuals   0.05136 88
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Ici, on rejette les hypothèses nulles (1) que les pentes sont les mêmes dans les deux sites et (2) que les ordonnées à l'origine sont égales. En d'autres mots, si on veut prédire la longueur à la fourche d'un esturgeon à un âge donné précisément, il faut savoir de quel site il provient. Puisque les pentes diffèrent, il faut estimer des régressions séparées.

Mais avant d'accepter ces conclusions, on doit se convaincre que les conditions d'application sont rencontrées:

Solution

```
par(mfrow = c(2,2))
plot(model.full2)
```

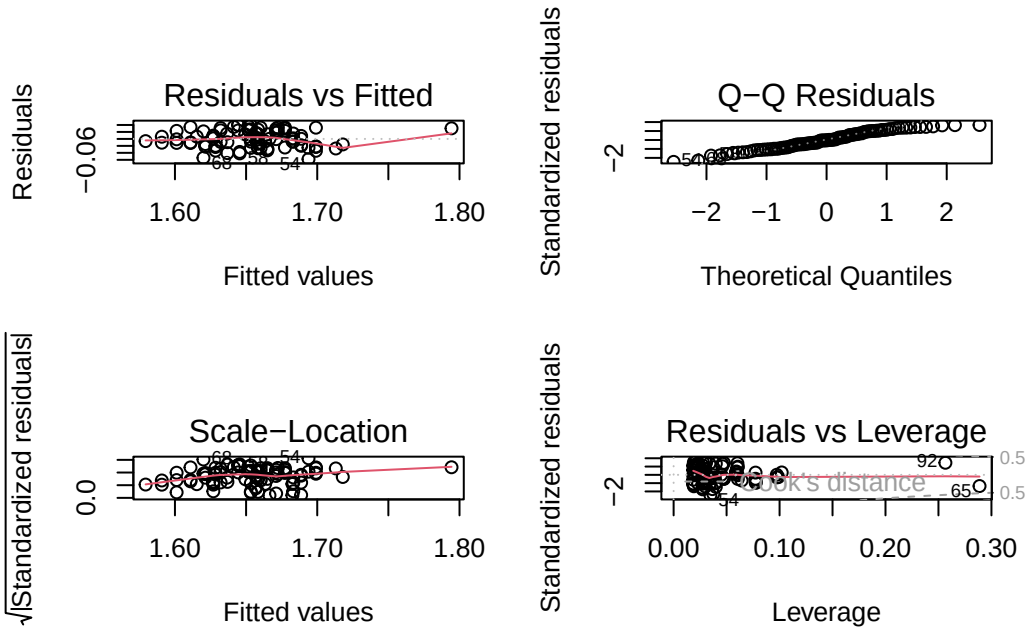


Figure 8.4.: Conditions d'applications du modèle `model.full2`

Si on examine les résidus selon les méthodes habituelles, on voit qu'il n'y a pas de problème de linéarité, ni d'homoscédasticité (BP = 2.8721, p = 0.4118). Cependant, le test de Wilk-Shapiro est significatif (W=0.97, p = 0.03). Étant donné la taille assez grande de l'échantillon (N=92), ce test a beaucoup de puissance, même si la déviation de normalité ne semble pas très grande. Compte-tenu de la robustesse relative des LM, de la taille de l'échantillon, on ne devrait pas ^tre trop inquiet de cette déviation de normalité.

Donc, comme les conditions des LM sont suffisamment remplies, on peut accepter les résultats donnés par R. Tous les termes sont significatifs (location, lage, interaction). Ce modèle complet est équivalent à ajuster des régressions séparées pour chaque site. Pour obtenir les coefficients, on peut ajuster des régressions simples sur chaque sous-ensemble, ou extraire les coefficients ajustés du modèle complet:

```
model.full2
```

```
Call:
lm(formula = lfkl ~ lage + locate + lage:locate, data = anc3dat)

Coefficients:
            (Intercept)                    lage       locateNELSON
                 1.2284                  0.3253                  0.2207
lage:locateNELSON
                -0.1656
```

Par défaut, la variable `locate` est encodée comme 0 pour le site qui vient le premier en ordre alphabétique (LofW) et 1 pour l'autre (Nelson). Les régressions pour chaque site deviennent donc:

Pour LofW:

$$\begin{aligned} lfkl &= 1.2284 + 0.3253 \times lage + 0.2207 \times 0 - 0.1656 \times 0 \times lage \\ &= 1.2284 + 0.3253 \times lage \end{aligned}$$

Pour Nelson:

$$\begin{aligned} lfkl &= 1.2284 + 0.3253 \times lage + 0.2207 \times 1 - 0.1656 \times 1 \times lage \\ &= 1.4491 + 0.1597 \times lage \end{aligned}$$

Vous pouvez vérifier en ajustant séparément les régressions pour chaque site:

```
by(anc3dat, anc3dat$locate,function(x) lm(lfkl~lage, data=x))
```

```
anc3dat$locate: LOFW

Call:
lm(formula = lfkl ~ lage, data = x)

Coefficients:
(Intercept)         lage
     1.2284       0.3253

------------------------------------------------------------
anc3dat$locate: NELSON

Call:
lm(formula = lfkl ~ lage, data = x)

Coefficients:
(Intercept)         lage
     1.4491       0.1597
```

8.5. Le modèle d'ANCOVA

Si le test d'homogénéité des pentes indique qu'elles diffèrent, alors on devrait estimer des régressions individuelles pour chaque niveau des variables discontinues. Cependant, si on accepte l'hypothèse d'égalité des pentes, l'étape suivante est de comparer les ordonnées à l'origine. Selon la "vieille école" i.e. l'approche traditionnelle, on ajuste un modèle avec la variable catégorique et la variable continue, mais sans interaction (le modèle ANCOVA sensus stricto) et on utilise la somme des carrés des écarts de type III, disons avec la fonction `Anova()`. C'est ce que la majorité des manuels de biostatistiques présentent.

L'autre approche consiste à utiliser les résultats de l'analyse du modèle complet, et tester la signification de chaque terme à partir des sommes des carrés partiels. C'est plus rapide, mais moins puissant. Dans la plupart des cas, cette perte de puissance n'est pas trop préoccupante, sauf lorsque le modèle est très complexe et contient de nombreuses interactions non-significatives. Je vous suggère d'utiliser l'approche simplifiée, et de n'utiliser l'approche traditionnelle que lorsque vous acceptez l'hypothèse d'égalité des ordonnées à l'origine. Pourquoi?

Puisque l'approche simplifiée est moins puissante, si vous rejetez quand même H0, alors votre conclusion ne changera pas, mais sera seulement renforcée, en utilisant l'approche traditionnelle.

Ici, je vais comparer l'approche traditionnelle et l'approche simplifiée. Rappelez-vous que vous voulez évaluer l'égalité des ordonnées à l'origine **après avoir déterminé que les pentes étaient égales**. Éprouver l'égalité des ordonnées à l'origine quand les pentes diffèrent (ou, si vous préférez, quand il y a une interaction) est rarement sensé, peut facilement être mal interprété, et ne devrait être effectué que rarement.

De retour aux données de anc1dat.csv, en comparant la relation entre lfkl et lage entre les sexes, nous avions obtenu les résultats suivants pour le modèle complet avec interactions:

```
Anova(model.full1, type = 3)
```

```
Anova Table (Type III tests)

Response: lfkl
             Sum Sq Df  F value    Pr(>F)
(Intercept) 0.64444  1 794.8182 < 2.2e-16 ***
sex         0.00041  1   0.5043    0.4795
lage        0.07259  1  89.5312 4.588e-15 ***
sex:lage    0.00027  1   0.3367    0.5632
Residuals   0.07135 88
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

On avait déjà conclu que la pente ne varie pas entre les sexes (i.e. l'interaction n'est pas significative). Notez que la p-valeur associée au sexe (0.4795) n'est pas significative non plus.

De l'autre côté, selon l'approche traditionnelle, l'inférence quand à l'effet du sexe se fait à partir du modèle réduit (le modèle ANCOVA sensus stricto):

```
model.ancova <- lm(lfkl ~ sex + lage, data = anc1dat)
Anova(model.ancova, type = 3)
```

```
Anova Table (Type III tests)

Response: lfkl
             Sum Sq Df   F value Pr(>F)
(Intercept) 1.13480  1 1410.1232 <2e-16 ***
sex         0.00149  1    1.8513 0.1771
lage        0.14338  1  178.1627 <2e-16 ***
Residuals   0.07162 89
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
summary(model.ancova)
```

```
Call:
lm(formula = lfkl ~ sex + lage, data = anc1dat)

Residuals:
      Min        1Q    Median        3Q       Max
-0.093992 -0.018457 -0.000876  0.022491  0.081161

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      1.225533   0.032636  37.552   <2e-16 ***
sexMALE         -0.008473   0.006228  -1.361    0.177
lage             0.327253   0.024517  13.348   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02837 on 89 degrees of freedom
Multiple R-squared:  0.696, Adjusted R-squared:  0.6892
F-statistic: 101.9 on 2 and 89 DF,  p-value: < 2.2e-16
```

Dans ce modèle, sex n'est pas significatif et on conclue donc que l'ordonnée à l'origine ne diffère pas entre les sexes. Notez que la pvaleur est plus petite (0.1771 vs 0.4795), ce qui reflète la puissance accrue de l'approche traditionnelle. Toutefois, les conclusions sont les mêmes: les ordonnées à l'origine ne diffèrent pas.

Exercice

En examinant les graphiques diagnostiques, vous noterez qu'il y a trois observations dont la valeur absolue du résidu est grande (cas 19, 49, et 50). Ces observations pourraient avoir un effet disproportionné sur les résultats de l'analyse. Éliminez-les et refaites l'analyse. Les conclusions changent-elles ?

```r
model.ancova.nooutliers <- lm(lfkl ~ sex + lage, data = anc1dat[c(-49, -50, -19),])
Anova(model.ancova.nooutliers, type = 3)
```

```
Anova Table (Type III tests)

Response: lfkl
             Sum Sq Df   F value  Pr(>F)
(Intercept) 1.09160  1 1896.5204 < 2e-16 ***
sex         0.00232  1    4.0374 0.04764 *
lage        0.13992  1  243.0946 < 2e-16 ***
Residuals   0.04950 86
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(model.ancova.nooutliers)
```

```
Call:
lm(formula = lfkl ~ sex + lage, data = anc1dat[c(-49, -50, -19),
    ])

Residuals:
      Min        1Q    Median        3Q       Max
-0.058397 -0.018469 -0.000976  0.020696  0.040288

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      1.224000   0.028106  43.549   <2e-16 ***
sexMALE         -0.010823   0.005386  -2.009   0.0476 *
lage             0.328604   0.021076  15.591   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02399 on 86 degrees of freedom
Multiple R-squared:  0.7706,    Adjusted R-squared:  0.7653
F-statistic: 144.4 on 2 and 86 DF,  p-value: < 2.2e-16
```

Ouch! Les résultats changent. Il faudrait donc rejeter l'hypothèse nulle et conclure que les ordonnées à l'origine diffèrent! Une conclusion qualitativement différente de celle obtenue en considérant toutes les données. Pourquoi? Il y a deux raisons possibles : (1) les valeurs extrêmes influencent beaucoup les régressions ou (2) l'exclusion des valeurs extrêmes permet d'augmenter la puissance de détection d'une différence. La première explication est moins plausible parce que les valeurs extrêmes n'avaient pas une grande influence (leverage faible). Alors, la deuxième explication est plus plausible et vous pouvez le vérifier en faisant des régressions pour chaque sexe sans et avec les valeurs extrêmes. Si vous le faites, vous noterez que les ordonnées à l'origine pour chaque sexe ne changent presque pas alors que leurs erreurs-types changent beaucoup.

> Exercice
>
> Ajustez une régression simple entre lfkl et lage pour l'ensemble complet de données et aussi pour le sous-ensemble sans les 3 valeurs déviantes. Comparez ces modèles avec les modèles d'ANCOVA ajustés précédemment. Que concluez-vous ? Quel modèle, d'après vous, a le meilleur ajustement aux données ? Pourquoi ?

Le modèle en excluant les valeurs extrêmes:

```
model.linear.nooutliers<-lm(lfkl ~ lage,data = anc1dat[c(-49, -50, -19),])
summary(model.linear.nooutliers)
```

```
Call:
lm(formula = lfkl ~ lage, data = anc1dat[c(-49, -50, -19), ])

Residuals:
      Min        1Q    Median        3Q       Max
-0.055567 -0.017809 -0.002944  0.021272  0.044972

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.20378    0.02670   45.09   <2e-16 ***
lage         0.34075    0.02054   16.59   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02441 on 87 degrees of freedom
Multiple R-squared:  0.7598,    Adjusted R-squared:  0.7571
F-statistic: 275.2 on 1 and 87 DF,  p-value: < 2.2e-16
```

Pour la régression simple (sans les valeurs extrêmes) on obtient un R 2 de 0.76 et une erreur-type des résidus de 0.02441, En comparant à l'erreur-type des résidus du modèle d'ANCOVA (0.02399) on réalise que la qualité des prédictions est essentiellement la même, même en ajustant des ordonnées à l'origine différentes pour chaque groupe. Par conséquent, les bénéfices de l'inclusion d'un terme pour les différentes ordonnées à l'origine sont faibles alors que le coût, en terme de complexité du modèle, est élevé (33% d'augmentation du nombre de termes pour un très faible amélioration de la qualité d'ajustement). Si vous examinez les résidus de ce modèle, vous trouverez qu'ils sont à peu près O.K.)

Si on ajuste une régression simple sur toutes les données, on obtient:

```r
model.linear<-lm(lfkl ~ lage, data = anc1dat)
summary(model.linear)
```

```
Call:
lm(formula = lfkl ~ lage, data = anc1dat)

Residuals:
      Min        1Q    Median        3Q       Max
-0.090915 -0.018975 -0.002587  0.021270  0.085273

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.21064    0.03089   39.19   <2e-16 ***
lage         0.33606    0.02376   14.14   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0285 on 90 degrees of freedom
```

```
Multiple R-squared:  0.6897,    Adjusted R-squared:  0.6863
F-statistic: 200.1 on 1 and 90 DF,  p-value: < 2.2e-16
```

Encore une fois, l'erreur-type des résidus (0.0285) pour cette régression unique est semblable à la variance du modèle d'ANCOVA (0.02837) et le modèle simplifié prédit presque aussi bien que le modèle plus complexe. (Ici encore, toutes les conditions d'application semblent remplies, si ce n'est de la valeur extrême).

Donc, dans les deux cas (avec ou sans les valeurs extrêmes), l'addition d'un terme supplémentaire pour le sexe n'ajoute pas grand-chose. Il semble donc que le meilleur modèle soit celui de la régression simple. Un estimé raisonnablement précis de la taille des esturgeons peut être obtenu de la régression commune sur l'ensemble des résultats.

Note: Il est fréquent que l'élimination de valeurs extrêmes en fasse apparaître d'autres. C'est parce que ces valeurs extrêmes dépendent de la variabilité résiduelle. Si on élimine les valeurs les plus déviantes, la variabilité résiduelle diminue, et certaines observations qui n'étaient pas si déviantes que cela deviennent proportionnellement plus déviantes. Notez aussi qu'en éliminant des valeurs extrêmes, l'effectif diminue et que la puissance décroît. Il faut donc être prudent.

8.6. Comparer l'ajustement de modèles

Comme vous venez de le voir, le processus d'ajustement de modèles est itératif. La plupart du temps il y a plus d'un modèle qui peut être ajusté aux données et c'est à vous de choisir celui qui est le meilleur compromis entre la qualité d'ajustement (qu'on essaie de maximiser) et la complexité (qu'on essaie de minimiser). La stratégie de base en ajustant des modèles linéaires (ANOVA, régression, ANCOVA) est de privilégier le modèle le plus simple si la qualité d'ajustement n'est pas significativement plus mauvaise. R peut calculer une statistique F vous permettant de comparer l'ajustement de deux modèles. Dans ce cas, l'hypothèse nulle est que la qualité d'ajustement ne diffère pas entre les deux modèles.

> Exercice
>
> En utilisant les données de `anc1dat` comparez l'ajustement du modèle ANCOVA et de la régression commune:

```
anova(model.ancova,model.linear)
```

```
Analysis of Variance Table

Model 1: lfkl ~ sex + lage
Model 2: lfkl ~ lage
  Res.Df      RSS Df  Sum of Sq      F Pr(>F)
1     89 0.071623
2     90 0.073113 -1 -0.0014899 1.8513 0.1771
```

La fonction anova() utilise la différence entre la somme des carrés des deux modèles et la divise par la différence entre le nombre de degrés de liberté pour obtenir un carré moyen. Ce carré moyen est utilisé au numérateur et est divisé par la variance résiduelle du modèle le plus complexe pour obtenir la statistique

F. Dans ce cas-ci, le test de F n'est pas significatif, et on conclut que les deux modèles ont une qualité d'ajustement équivalente, et qu'on devrait donc privilégier le modèle le plus simple, la régression linéaire simple.

Exercice

Refaites le même processus avec le données de `anc3dat`, ajustez le modèle complet avec interaction (LFKL~LAGE+LOCATE+LAGE:LOCATE) et sans interaction (LFKL~LAGE+LOCATE), Comparez l'ajustement des deux modèles, que concluez vous?

Solution

```
model.full.anc3dat<-lm(lfkl ~ lage + locate + lage:locate, data = anc3dat)
model.ancova.anc3dat<-lm(lfkl ~ lage + locate, data = anc3dat)
anova(model.full.anc3dat,model.ancova.anc3dat)
```

```
Analysis of Variance Table

Model 1: lfkl ~ lage + locate + lage:locate
Model 2: lfkl ~ lage + locate
  Res.Df      RSS Df  Sum of Sq      F    Pr(>F)
1     88 0.051358
2     89 0.060448 -1 -0.0090901 15.575 0.0001592 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Cette fois-ci, le modèle plus complexe s'ajuste significativement mieux aux données. (Pas surprenant puisque nous avions précédemment conclu que l'interaction est significative avec ces données.)

8.7. Bootstrap

```
#################################################################
######
# Bootstrap analysis
# Bootstrap analysis BCa confidence intervals
# Preferable when parameter distribution is far from normal
# Bootstrap 95% BCa CI for regression coefficients
library(boot)

# To simplify future modifications of the code in this file,
# copy the data to a generic mydata dataframe
mydata <- anc3dat

# create a myformula variable containing the formula for the model to be fitted
```

```
myformula <- as.formula(lfkl ~ lage + locate + lage:locate)

# function to obtain regression coefficients for each iteration
bs <- function(formula, data, indices) {
  d <- data[indices, ]
  fit <- lm(formula, data = d)
  return(coef(fit))
}
# bootstrapping with 1000 replications
results <- boot(data = mydata, statistic = bs, R = 1000, formula = myformula)

# view results
results
boot_res <- summary(results)
rownames(boot_res) <- names(results$t0)
boot_res

op <- par(ask = TRUE)
for (i in 1:length(results$t0)) {
  plot(results, index = i)
  title(names(results$t0)[i])
}
par(op)

# get 95% confidence intervals
for (i in 1:length(results$t0)) {
  cat("\n", names(results$t0)[i],"\n")
  print(boot.ci(results, type = "bca", index = i))
}
```

8.8. Permutation test

```
############################################################
##########
# Permutation test
#
# using lmperm library
# To simplify future modifications of the code in this file,
# copy the data to a generic mydata dataframe
mydata<-anc3dat
# create a myformula variable containing the formula for the
# model to be fitted
myformula<-as.formula(lfkl ~ lage + locate + lage:locate)
require(lmPerm2)
# Fit desired model on the desired dataframe
```

```
mymodel <- lm(myformula, data = mydata)
# Calculate p-values for each term by permutation
# Note that lmp centers numeric variable by default, so to
# get results that are
# consistent with standard models, it is necessary to set
center=FALSE
mymodelProb <- lmp(myformula, data = mydata, center=FALSE,
perm = "Prob")
summary(mymodel)
summary(mymodelProb)
```

Références

Paquets R

Package	Version	Citation
base	4.3.2	R Core Team (2023)
bayesplot	1.10.0	Gabry et al. (2019); Gabry and Mahr (2022)
boot	1.3.28.1	(**boot1997?**); (**boot2022?**)
car	3.1.2	Fox and Weisberg (2019a)
effects	4.2.2	Fox (2003); Fox and Hong (2009); Fox and Weisberg (2018); Fox and Weisberg (2019b)
emo	0.0.0.9000	Wickham et al. (2023b)
factoextra	1.0.7	Kassambara and Mundt (2020)
knitr	1.45	Xie (2014); Xie (2015); Xie (2023a)
lattice	0.22.5	(**lattice?**)
lmPerm	2.1.0	Wheeler and Torchiano (2016)
lmtest	0.9.40	Zeileis and Hothorn (2002)
MASS	7.3.60	(**MASS?**)
multcomp	1.4.25	Hothorn et al. (2008)
MuMIn	1.47.5	Bartoń (2023)
palmerpenguins	0.1.1	Horst et al. (2020)
patchwork	1.2.0	Pedersen (2024)
performance	0.10.8	Lüdecke et al. (2021)
pwr	1.3.0	Champely (2020)
questionr	0.7.8	Barnier et al. (2023)
rmarkdown	2.25	Xie et al. (2018); Xie et al. (2020); Allaire et al. (2023)
simpleboot	1.1.7	Peng (2019)
svglite	2.1.3	Wickham et al. (2023c)
tidybayes	3.0.6	Kay (2023a)
tidyverse	2.0.0	Wickham et al. (2019)
vcd	1.4.12	Meyer et al. (2006); Zeileis et al. (2007); Meyer et al. (2023b)
vcdExtra	0.8.5	Friendly (2023)

Package	Version	Citation
abind	1.4.5	Plate and Heiberger (2016)
arrayhelpers	1.1.0	Beleites (2020)
askpass	1.2.0	Ooms (2023a)
backports	1.4.1	Lang and R Core Team (2021)
base64enc	0.1.3	Urbanek (2015)
bayestestR	0.13.1	Makowski et al. (2019)
bit	4.0.5	Oehlschlägel and Ripley (2022)

Package	Version	Citation
bit64	4.0.5	Oehlschlägel and Silvestri (2020)
blob	1.2.4	Wickham (2023a)
brio	1.1.4	Hester and Csárdi (2023)
bslib	0.6.1	Sievert et al. (2023)
ca	0.71.1	Nenadic and Greenacre (2007)
cachem	1.0.8	Chang (2023a)
callr	3.7.3	Csárdi and Chang (2022)
carData	3.0.5	Fox et al. (2022)
cellranger	1.1.0	Bryan (2016)
checkmate	2.3.1	Lang (2017)
classInt	0.4.10	Bivand (2023)
clipr	0.8.0	Lincoln (2022)
coda	0.19.4	Plummer et al. (2006)
colorspace	2.1.0	Zeileis et al. (2009); Stauffer et al. (2009); Zeileis et al. (2020a)
commonmark	1.9.0	Ooms (2023b)
corrplot	0.92	Wei and Simko (2021)
cowplot	1.1.2	Wilke (2023a)
cpp11	0.4.7	Vaughan et al. (2023)
crayon	1.5.2	Csárdi (2022)
crosstalk	1.2.1	Cheng and Sievert (2023)
curl	5.2.0	Ooms (2023c)
data.table	1.14.10	Barrett et al. (2023)
datawizard	0.9.1	Patil et al. (2022)
DBI	1.2.1	R Special Interest Group on Databases (R-SIG-DB) et al. (2024)
dendextend	1.17.1	Galili (2015)
desc	1.4.3	Csárdi et al. (2023b)
diffobj	0.3.5	Gaslam (2021)
digest	0.6.34	Antoine Lucas et al. (2024)
distributional	0.3.2	O'Hara-Wild et al. (2023)
DT	0.31	Xie et al. (2023)
e1071	1.7.14	Meyer et al. (2023a)
ellipse	0.5.0	Murdoch and Chow (2023)
ellipsis	0.3.2	Wickham (2021)
emmeans	1.9.0	Lenth (2023)
estimability	1.4.1	Lenth (2022)
evaluate	0.23	Wickham and Xie (2023)
FactoMineR	2.9	Lê et al. (2008)
fansi	1.0.6	Gaslam (2023)
farver	2.1.1	Pedersen et al. (2022)
fastmap	1.1.1	Chang (2023b)
flashClust	1.1.2	Langfelder and Horvath (2012)
fontawesome	0.5.2	Iannone (2023)
fs	1.6.3	Hester et al. (2023b)
gargle	1.5.2	Bryan et al. (2023)
generics	0.1.3	Wickham et al. (2022a)
ggdist	3.3.1	Kay (2023b); Kay (2024)
ggpubr	0.6.0	Kassambara (2023a)
ggrepel	0.9.5	Slowikowski (2024)

Package	Version	Citation
ggridges	0.5.5	Wilke (2023b)
ggsci	3.0.0	Xiao (2023)
ggsignif	0.6.4	Constantin and Patil (2021)
glue	1.7.0	Hester and Bryan (2024)
gnm	1.1.5	Turner and Firth (2023)
gridExtra	2.3	Auguie (2017)
gtable	0.3.4	Wickham and Pedersen (2023)
here	1.0.1	Müller (2020)
highr	0.10	Xie and Qiu (2022)
htmltools	0.5.7	Cheng et al. (2023c)
htmlwidgets	1.6.4	Vaidyanathan et al. (2023)
httpuv	1.6.13	Cheng et al. (2023a)
ids	1.0.1	FitzJohn (2017)
insight	0.19.7	Lüdecke et al. (2019)
isoband	0.2.7	Wickham et al. (2022b)
jquerylib	0.1.4	Sievert and Cheng (2021)
labeling	0.4.3	Justin Talbot (2023)
labelled	2.12.0	Larmarange (2023)
later	1.3.2	Chang and Cheng (2023)
lazyeval	0.2.2	Wickham (2019)
leaps	3.1	Fortran code by Alan Miller (2020)
lifecycle	1.0.4	Henry and Wickham (2023)
lme4	1.1.35.1	Bates et al. (2015)
MatrixModels	0.5.3	Bates and Maechler (2023)
matrixStats	1.2.0	Bengtsson (2023a)
memoise	2.0.1	Wickham et al. (2021)
mime	0.12	Xie (2021)
miniUI	0.1.1.1	Cheng (2018)
minqa	1.2.6	Bates et al. (2023)
mitools	2.4	Lumley (2019)
multcompView	0.1.9	Graves et al. (2023)
munsell	0.5.0	Wickham (2018)
mvtnorm	1.2.4	Genz and Bretz (2009)
nloptr	2.0.3	Johnson (?)
numDeriv	2016.8.1.1	Gilbert and Varadhan (2019)
openssl	2.1.1	Ooms (2023d)
pbkrtest	0.5.2	Halekoh and Højsgaard (2014)
pkgbuild	1.4.3	Wickham et al. (2023e)
pkgconfig	2.0.3	Csárdi (2019)
pkgload	1.3.3	Wickham et al. (2023a)
plyr	1.8.9	Wickham (2011a)
polynom	1.4.1	Venables et al. (2022)
posterior	1.5.0	Vehtari et al. (2021); Bürkner et al. (2023)
praise	1.0.0	Csardi and Sorhus (2015)
prettyunits	1.2.0	Csardi (2023a)
processx	3.8.3	Csárdi and Chang (2023)
progress	1.2.3	Csárdi and FitzJohn (2023)
promises	1.2.1	Cheng (2023)

Package	Version	Citation
proxy	0.4.27	Meyer and Buchta (2022)
ps	1.7.5	Loden et al. (2023)
quadprog	1.5.8	Berwin A. Turlach R port by Andreas Weingessel <Andreas.Weingessel@ci.tuwien.ac.at> Fortran contributions from Cleve Moler dpodi/LINPACK) (2019)
quantreg	5.97	Koenker (2023)
qvcalc	1.0.3	Firth (2023)
R.cache	0.16.0	Bengtsson (2022)
R.methodsS3	1.8.2	Bengtsson (2003a)
R.oo	1.25.0	Bengtsson (2003b)
R.utils	2.12.3	Bengtsson (2023b)
R6	2.5.1	Chang (2021)
rappdirs	0.3.3	Ratnakumar et al. (2021)
RColorBrewer	1.1.3	Neuwirth (2022)
Rcpp	1.0.12	Eddelbuettel and François (2011); Eddelbuettel (2013); Eddelbuettel and Balamuta (2018); Eddelbuettel et al. (2024)
RcppEigen	0.3.3.9.4	Bates and Eddelbuettel (2013)
relimp	1.0.5	Heather Turner and Fox (2016)
rematch	2.0.0	Csardi (2023b)
rematch2	2.1.2	Csárdi (2020)
remotes	2.4.2.1	Csárdi et al. (2023a)
renv	1.0.3	Ushey and Wickham (2023)
reshape2	1.4.4	Wickham (2007)
rprojroot	2.0.4	Müller (2023)
rstatix	0.7.2	Kassambara (2023b)
sandwich	3.1.0	Zeileis (2004); Zeileis (2006); Zeileis et al. (2020b)
sass	0.4.8	Cheng et al. (2023b)
scales	1.3.0	Wickham et al. (2023f)
scatterplot3d	0.3.44	Ligges and Mächler (2003)
selectr	0.4.2	Potter (2012)
shiny	1.8.0	Chang et al. (2023)
sourcetools	0.1.7.1	Ushey (2023)
SparseM	1.81	Koenker (2021)
stringi	1.8.3	Gagolewski (2022)
styler	1.10.2	Müller and Walthert (2023)
survey	4.2.1	Lumley (2004); Lumley (2010); Lumley (2023)
svUnit	1.0.6	Grosjean (2023)
sys	3.4.2	Ooms (2023e)
systemfonts	1.0.5	Pedersen et al. (2023)
tensorA	0.36.2.1	van den Boogaart (2023)
testthat	3.2.1	Wickham (2011b)
textshaping	0.3.7	Pedersen (2023)
TH.data	1.1.2	Hothorn (2023)
tidyselect	1.2.0	Henry and Wickham (2022)
timechange	0.2.0	Spinu (2023)
tinytex	0.49	Xie (2019); Xie (2023b)
tzdb	0.4.0	Vaughan (2023)
utf8	1.2.4	Perry (2023)

Package	Version	Citation
uuid	1.1.1	Urbanek and Ts'o (2023)
vctrs	0.6.5	Wickham et al. (2023d)
viridis	0.6.4	Garnier et al. (2023a)
viridisLite	0.4.2	Garnier et al. (2023b)
vroom	1.6.5	Hester et al. (2023a)
waldo	0.5.2	Wickham (2023b)
withr	3.0.0	Hester et al. (2024)
xfun	0.41	Xie (2023c)
xtable	1.8.4	Dahl et al. (2019)
yaml	2.3.8	Garbett et al. (2023)
zoo	1.8.12	Zeileis and Grothendieck (2005)

Bibliography

Allaire, J., Y. Xie, C. Dervieux, J. McPherson, J. Luraschi, K. Ushey, A. Atkins, H. Wickham, J. Cheng, W. Chang, and R. Iannone. 2023. rmarkdown: Dynamic documents for r.

Antoine Lucas, D. E. with contributions by, J. Tuszynski, H. Bengtsson, S. Urbanek, M. Frasca, B. Lewis, M. Stokely, H. Muehleisen, D. Murdoch, J. Hester, W. Wu, Q. Kou, T. Onkelinx, M. Lang, V. Simko, K. Hornik, R. Neal, K. Bell, M. de Queljoe, I. Suruceanu, B. Denney, D. Schumacher, W. Chang, D. Attali, and M. Chirico. 2024. digest: Create compact hash digests of r objects.

Auguie, B. 2017. gridExtra: Miscellaneous functions for "Grid" graphics.

Barnier, J., F. Briatte, and J. Larmarange. 2023. questionr: Functions to make surveys processing easier.

Barrett, T., M. Dowle, and A. Srinivasan. 2023. data.table: Extension of "data.frame".

Bartoń, K. 2023. MuMIn: Multi-model inference.

Bates, D., and D. Eddelbuettel. 2013. Fast and elegant numerical linear algebra using the RcppEigen package. Journal of Statistical Software 52:1–24.

Bates, D., M. Mächler, B. Bolker, and S. Walker. 2015. Fitting linear mixed-effects models using lme4. Journal of Statistical Software 67:1–48.

Bates, D., and M. Maechler. 2023. MatrixModels: Modelling with sparse and dense matrices.

Bates, D., K. M. Mullen, J. C. Nash, and R. Varadhan. 2023. minqa: Derivative-free optimization algorithms by quadratic approximation.

Beleites, C. 2020. arrayhelpers: Convenience functions for arrays.

Bengtsson, H. 2003a. The R.oo package - object-oriented programming with references using standard R code. *in* K. Hornik, F. Leisch, and A. Zeileis, editors. Proceedings of the 3rd international workshop on distributed statistical computing (DSC 2003). https://www.r-project.org/conferences/DSC-2003/Proceedings/, Vienna, Austria.

Bengtsson, H. 2003b. The R.oo package - object-oriented programming with references using standard R code. *in* K. Hornik, F. Leisch, and A. Zeileis, editors. Proceedings of the 3rd international workshop on distributed statistical computing (DSC 2003). https://www.r-project.org/conferences/DSC-2003/Proceedings/, Vienna, Austria.

Bengtsson, H. 2022. R.cache: Fast and light-weight caching (memoization) of objects and results to speed up computations.

Bengtsson, H. 2023a. matrixStats: Functions that apply to rows and columns of matrices (and to vectors).

Bengtsson, H. 2023b. R.utils: Various programming utilities.

Berwin A. Turlach R port by Andreas Weingessel <Andreas.Weingessel@ci.tuwien.ac.at> Fortran contributions from Cleve Moler dpodi/LINPACK), S. original by. 2019. quadprog: Functions to solve quadratic programming problems.

Bivand, R. 2023. classInt: Choose univariate class intervals.
Bryan, J. 2016. cellranger: Translate spreadsheet cell ranges to rows and columns.
Bryan, J., C. Citro, and H. Wickham. 2023. gargle: Utilities for working with google APIs.
Bürkner, P.-C., J. Gabry, M. Kay, and A. Vehtari. 2023. posterior: Tools for working with posterior distributions.
Champely, S. 2020. pwr: Basic functions for power analysis.
Chang, W. 2021. R6: Encapsulated classes with reference semantics.
Chang, W. 2023a. cachem: Cache r objects with automatic pruning.
Chang, W. 2023b. fastmap: Fast data structures.
Chang, W., and J. Cheng. 2023. later: Utilities for scheduling functions to execute later with event loops.
Chang, W., J. Cheng, J. Allaire, C. Sievert, B. Schloerke, Y. Xie, J. Allen, J. McPherson, A. Dipert, and B. Borges. 2023. shiny: Web application framework for r.
Cheng, J. 2018. miniUI: Shiny UI widgets for small screens.
Cheng, J. 2023. promises: Abstractions for promise-based asynchronous programming.
Cheng, J., W. Chang, S. Reid, J. Brown, B. Trower, and A. Peslyak. 2023a. httpuv: HTTP and WebSocket server library.
Cheng, J., T. Mastny, R. Iannone, B. Schloerke, and C. Sievert. 2023b. sass: Syntactically awesome style sheets ("Sass").
Cheng, J., and C. Sievert. 2023. crosstalk: Inter-widget interactivity for HTML widgets.
Cheng, J., C. Sievert, B. Schloerke, W. Chang, Y. Xie, and J. Allen. 2023c. htmltools: Tools for HTML.
Constantin, A.-E., and I. Patil. 2021. ggsignif: R package for displaying significance brackets for "ggplot2". PsyArxiv.
Csardi, G. 2023a. prettyunits: Pretty, human readable formatting of quantities.
Csardi, G. 2023b. rematch: Match regular expressions with a nicer "API".
Csardi, G., and S. Sorhus. 2015. praise: Praise users.
Csárdi, G. 2019. pkgconfig: Private configuration for "R" packages.
Csárdi, G. 2020. rematch2: Tidy output from regular expression matching.
Csárdi, G. 2022. crayon: Colored terminal output.
Csárdi, G., and W. Chang. 2022. callr: Call r from r.
Csárdi, G., and W. Chang. 2023. processx: Execute and control system processes.
Csárdi, G., and R. FitzJohn. 2023. progress: Terminal progress bars.
Csárdi, G., J. Hester, H. Wickham, W. Chang, M. Morgan, and D. Tenenbaum. 2023a. remotes: R package installation from remote repositories, including "GitHub".
Csárdi, G., K. Müller, and J. Hester. 2023b. desc: Manipulate DESCRIPTION files.
Dahl, D. B., D. Scott, C. Roosen, A. Magnusson, and J. Swinton. 2019. xtable: Export tables to LaTeX or HTML.
Eddelbuettel, D. 2013. Seamless R and C++ integration with Rcpp. Springer, New York.
Eddelbuettel, D., and J. J. Balamuta. 2018. Extending R with C++: A Brief Introduction to Rcpp. The American Statistician 72:28–36.
Eddelbuettel, D., R. Francois, J. Allaire, K. Ushey, Q. Kou, N. Russell, I. Ucar, D. Bates, and J. Chambers. 2024. Rcpp: Seamless r and c++ integration.
Eddelbuettel, D., and R. François. 2011. Rcpp: Seamless R and C++ integration. Journal of Statistical Software 40:1–18.
Firth, D. 2023. qvcalc: Quasi variances for factor effects in statistical models.
FitzJohn, R. 2017. ids: Generate random identifiers.
Fortran code by Alan Miller, T. L. based on. 2020. leaps: Regression subset selection.
Fox, J. 2003. Effect displays in R for generalised linear models. Journal of Statistical Software 8:1–27.
Fox, J., and J. Hong. 2009. Effect displays in R for multinomial and proportional-odds logit models: Extensions to the effects package. Journal of Statistical Software 32:1–24.
Fox, J., and S. Weisberg. 2018. Visualizing fit and lack of fit in complex regression models with predictor

effect plots and partial residuals. Journal of Statistical Software 87:1–27.
Fox, J., and S. Weisberg. 2019a. An R companion to applied regression. Third. Sage, Thousand Oaks CA.
Fox, J., and S. Weisberg. 2019b. An r companion to applied regression. 3rd edition. Sage, Thousand Oaks CA.
Fox, J., S. Weisberg, and B. Price. 2022. carData: Companion to applied regression data sets.
Friendly, M. 2023. vcdExtra: "vcd" extensions and additions.
Gabry, J., and T. Mahr. 2022. bayesplot: Plotting for bayesian models.
Gabry, J., D. Simpson, A. Vehtari, M. Betancourt, and A. Gelman. 2019. Visualization in bayesian workflow. J. R. Stat. Soc. A 182:389–402.
Gagolewski, M. 2022. stringi: Fast and portable character string processing in R. Journal of Statistical Software 103:1–59.
Galili, T. 2015. dendextend: An r package for visualizing, adjusting, and comparing trees of hierarchical clustering. Bioinformatics.
Garbett, S. P., J. Stephens, K. Simonov, Y. Xie, Z. Dong, H. Wickham, J. Horner, reikoch, W. Beasley, B. O'Connor, G. R. Warnes, M. Quinn, and Z. N. Kamvar. 2023. yaml: Methods to convert r data to YAML and back.
Garnier, Simon, Ross, Noam, Rudis, Robert, Camargo, A. Pedro, Sciaini, Marco, Scherer, and Cédric. 2023a. viridis(Lite) - colorblind-friendly color maps for r.
Garnier, Simon, Ross, Noam, Rudis, Robert, Camargo, A. Pedro, Sciaini, Marco, Scherer, and Cédric. 2023b. viridis(Lite) - colorblind-friendly color maps for r.
Gaslam, B. 2021. diffobj: Diffs for r objects.
Gaslam, B. 2023. fansi: ANSI control sequence aware string functions.
Genz, A., and F. Bretz. 2009. Computation of multivariate normal and t probabilities. Springer-Verlag, Heidelberg.
Gilbert, P., and R. Varadhan. 2019. numDeriv: Accurate numerical derivatives.
Graves, S., H.-P. Piepho, and L. S. with help from Sundar Dorai-Raj. 2023. multcompView: Visualizations of paired comparisons.
Grosjean, P. 2023. SciViews-r. UMONS, MONS, Belgium.
Halekoh, U., and S. Højsgaard. 2014. A kenward-roger approximation and parametric bootstrap methods for tests in linear mixed models – the R package pbkrtest. Journal of Statistical Software 59:1–30.
Heather Turner, D. F. with contributions from, and J. Fox. 2016. relimp: Relative contribution of effects in a regression model.
Henry, L., and H. Wickham. 2022. tidyselect: Select from a set of strings.
Henry, L., and H. Wickham. 2023. lifecycle: Manage the life cycle of your package functions.
Hester, J., and J. Bryan. 2024. glue: Interpreted string literals.
Hester, J., and G. Csárdi. 2023. brio: Basic r input output.
Hester, J., L. Henry, K. Müller, K. Ushey, H. Wickham, and W. Chang. 2024. withr: Run code "With" temporarily modified global state.
Hester, J., H. Wickham, and J. Bryan. 2023a. vroom: Read and write rectangular text data quickly.
Hester, J., H. Wickham, and G. Csárdi. 2023b. fs: Cross-platform file system operations based on "libuv".
Horst, A. M., A. P. Hill, and K. B. Gorman. 2020. palmerpenguins: Palmer archipelago (antarctica) penguin data.
Hothorn, T. 2023. TH.data: TH's data archive.
Hothorn, T., F. Bretz, and P. Westfall. 2008. Simultaneous inference in general parametric models. Biometrical Journal 50:346–363.
Iannone, R. 2023. fontawesome: Easily work with "Font Awesome" icons.
Johnson, S. G. ? The NLopt nonlinear-optimization package. ? ?
Justin Talbot. 2023. labeling: Axis labeling.
Kassambara, A. 2023a. ggpubr: "ggplot2" based publication ready plots.

Kassambara, A. 2023b. rstatix: Pipe-friendly framework for basic statistical tests.
Kassambara, A., and F. Mundt. 2020. factoextra: Extract and visualize the results of multivariate data analyses.
Kay, M. 2023b. ggdist: Visualizations of distributions and uncertainty.
Kay, M. 2023a. tidybayes: Tidy data and geoms for Bayesian models.
Kay, M. 2024. ggdist: Visualizations of distributions and uncertainty in the grammar of graphics. IEEE Transactions on Visualization and Computer Graphics:1–11.
Koenker, R. 2021. SparseM: Sparse linear algebra.
Koenker, R. 2023. quantreg: Quantile regression.
Lang, M. 2017. checkmate: Fast argument checks for defensive R programming. The R Journal 9:437–445.
Lang, M., and R Core Team. 2021. backports: Reimplementations of functions introduced since r-3.0.0.
Langfelder, P., and S. Horvath. 2012. Fast R functions for robust correlations and hierarchical clustering. Journal of Statistical Software 46:1–17.
Larmarange, J. 2023. labelled: Manipulating labelled data.
Lê, S., J. Josse, and F. Husson. 2008. FactoMineR: A package for multivariate analysis. Journal of Statistical Software 25:1–18.
Lenth, R. 2022. estimability: Tools for assessing estimability of linear predictions.
Lenth, R. V. 2023. emmeans: Estimated marginal means, aka least-squares means.
Ligges, U., and M. Mächler. 2003. Scatterplot3d - an r package for visualizing multivariate data. Journal of Statistical Software 8:1–20.
Lincoln, M. 2022. clipr: Read and write from the system clipboard.
Loden, J., D. Daeschler, G. Rodola', and G. Csárdi. 2023. ps: List, query, manipulate system processes.
Lüdecke, D., M. S. Ben-Shachar, I. Patil, P. Waggoner, and D. Makowski. 2021. performance: An R package for assessment, comparison and testing of statistical models. Journal of Open Source Software 6:3139.
Lüdecke, D., P. Waggoner, and D. Makowski. 2019. insight: A unified interface to access information from model objects in R. Journal of Open Source Software 4:1412.
Lumley, T. 2004. Analysis of complex survey samples. Journal of Statistical Software 9:1–19.
Lumley, T. 2010. Complex surveys: A guide to analysis using r: A guide to analysis using r. John Wiley; Sons.
Lumley, T. 2019. mitools: Tools for multiple imputation of missing data.
Lumley, T. 2023. survey: Analysis of complex survey samples.
Makowski, D., M. S. Ben-Shachar, and D. Lüdecke. 2019. bayestestR: Describing effects and their uncertainty, existence and significance within the bayesian framework. Journal of Open Source Software 4:1541.
Meyer, D., and C. Buchta. 2022. proxy: Distance and similarity measures.
Meyer, D., E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch. 2023a. e1071: Misc functions of the department of statistics, probability theory group (formerly: E1071), TU wien.
Meyer, D., A. Zeileis, and K. Hornik. 2006. The strucplot framework: Visualizing multi-way contingency tables with vcd. Journal of Statistical Software 17:1–48.
Meyer, D., A. Zeileis, K. Hornik, and M. Friendly. 2023b. vcd: Visualizing categorical data.
Müller, K. 2020. here: A simpler way to find your files.
Müller, K. 2023. rprojroot: Finding files in project subdirectories.
Müller, K., and L. Walthert. 2023. styler: Non-invasive pretty printing of r code.
Murdoch, D., and E. D. Chow. 2023. ellipse: Functions for drawing ellipses and ellipse-like confidence regions.
Nenadic, O., and M. Greenacre. 2007. Correspondence analysis in r, with two- and three-dimensional graphics: The ca package. Journal of Statistical Software 20:1–13.
Neuwirth, E. 2022. RColorBrewer: ColorBrewer palettes.
O'Hara-Wild, M., M. Kay, and A. Hayes. 2023. distributional: Vectorised probability distributions.

Oehlschlägel, J., and B. Ripley. 2022. bit: Classes and methods for fast memory-efficient boolean selections.
Oehlschlägel, J., and L. Silvestri. 2020. bit64: A S3 class for vectors of 64bit integers.
Ooms, J. 2023a. askpass: Password entry utilities for r, git, and SSH.
Ooms, J. 2023b. commonmark: High performance CommonMark and github markdown rendering in r.
Ooms, J. 2023c. curl: A modern and flexible web client for r.
Ooms, J. 2023d. openssl: Toolkit for encryption, signatures and certificates based on OpenSSL.
Ooms, J. 2023e. sys: Powerful and reliable tools for running system commands in r.
Patil, I., D. Makowski, M. S. Ben-Shachar, B. M. Wiernik, E. Bacher, and D. Lüdecke. 2022. datawizard: An R package for easy data preparation and statistical transformations. Journal of Open Source Software 7:4684.
Pedersen, T. L. 2023. textshaping: Bindings to the "HarfBuzz" and "Fribidi" libraries for text shaping.
Pedersen, T. L. 2024. patchwork: The composer of plots.
Pedersen, T. L., B. Nicolae, and R. François. 2022. farver: High performance colour space manipulation.
Pedersen, T. L., J. Ooms, and D. Govett. 2023. systemfonts: System native font finding.
Peng, R. D. 2019. simpleboot: Simple bootstrap routines.
Perry, P. O. 2023. utf8: Unicode text processing.
Plate, T., and R. Heiberger. 2016. abind: Combine multidimensional arrays.
Plummer, M., N. Best, K. Cowles, and K. Vines. 2006. CODA: Convergence diagnosis and output analysis for MCMC. R News 6:7–11.
Potter, S. 2012. Introducing the selectr package. The University of Auckland, Auckland, New Zealand.
R Core Team. 2023. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.
R Special Interest Group on Databases (R-SIG-DB), H. Wickham, and K. Müller. 2024. DBI: R database interface.
Ratnakumar, S., T. Mick, and T. Davis. 2021. rappdirs: Application directories: Determine where to save data, caches, and logs.
Sievert, C., and J. Cheng. 2021. jquerylib: Obtain "jQuery" as an HTML dependency object.
Sievert, C., J. Cheng, and G. Aden-Buie. 2023. bslib: Custom "Bootstrap" "Sass" themes for "shiny" and "rmarkdown".
Slowikowski, K. 2024. ggrepel: Automatically position non-overlapping text labels with "ggplot2".
Spinu, V. 2023. timechange: Efficient manipulation of date-times.
Stauffer, R., G. J. Mayr, M. Dabernig, and A. Zeileis. 2009. Somewhere over the rainbow: How to make effective use of colors in meteorological visualizations. Bulletin of the American Meteorological Society 96:203–216.
Turner, H., and D. Firth. 2023. gnm: Generalized nonlinear models.
Urbanek, S. 2015. base64enc: Tools for base64 encoding.
Urbanek, S., and T. Ts'o. 2023. uuid: Tools for generating and handling of UUIDs.
Ushey, K. 2023. sourcetools: Tools for reading, tokenizing and parsing r code.
Ushey, K., and H. Wickham. 2023. renv: Project environments.
Vaidyanathan, R., Y. Xie, J. Allaire, J. Cheng, C. Sievert, and K. Russell. 2023. htmlwidgets: HTML widgets for r.
van den Boogaart, K. G. 2023. tensorA: Advanced tensor arithmetic with named indices.
Vaughan, D. 2023. tzdb: Time zone database information.
Vaughan, D., J. Hester, and R. François. 2023. cpp11: A c++11 interface for r's c interface.
Vehtari, A., A. Gelman, D. Simpson, B. Carpenter, and P.-C. Bürkner. 2021. Rank-normalization, folding, and localization: An improved rhat for assessing convergence of MCMC (with discussion). Bayesian Analysis.
Venables, B., K. Hornik, and M. Maechler. 2022. polynom: A collection of functions to implement a class for univariate polynomial manipulations.

Wei, T., and V. Simko. 2021. R package "corrplot": Visualization of a correlation matrix.
Wheeler, B., and M. Torchiano. 2016. lmPerm: Permutation tests for linear models.
Wickham, C. 2018. munsell: Utilities for using munsell colours.
Wickham, H. 2007. Reshaping data with the reshape package. Journal of Statistical Software 21:1–20.
Wickham, H. 2011a. The split-apply-combine strategy for data analysis. Journal of Statistical Software 40:1–29.
Wickham, H. 2011b. testthat: Get started with testing. The R Journal 3:5–10.
Wickham, H. 2019. lazyeval: Lazy (non-standard) evaluation.
Wickham, H. 2021. ellipsis: Tools for working with ...
Wickham, H. 2023a. blob: A simple S3 class for representing vectors of binary data ("BLOBS").
Wickham, H. 2023b. waldo: Find differences between r objects.
Wickham, H., M. Averick, J. Bryan, W. Chang, L. D. McGowan, R. François, G. Grolemund, A. Hayes, L. Henry, J. Hester, M. Kuhn, T. L. Pedersen, E. Miller, S. M. Bache, K. Müller, J. Ooms, D. Robinson, D. P. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, and H. Yutani. 2019. Welcome to the tidyverse. Journal of Open Source Software 4:1686.
Wickham, H., W. Chang, J. Hester, and L. Henry. 2023a. pkgload: Simulate package installation and attach.
Wickham, H., R. François, and L. D'Agostino McGowan. 2023b. emo: Easily insert "Emoji".
Wickham, H., L. Henry, T. L. Pedersen, T. J. Luciani, M. Decorde, and V. Lise. 2023c. svglite: An "SVG" graphics device.
Wickham, H., L. Henry, and D. Vaughan. 2023d. vctrs: Vector helpers.
Wickham, H., J. Hester, W. Chang, K. Müller, and D. Cook. 2021. memoise: "Memoisation" of functions.
Wickham, H., J. Hester, and G. Csárdi. 2023e. pkgbuild: Find tools needed to build r packages.
Wickham, H., M. Kuhn, and D. Vaughan. 2022a. generics: Common S3 generics not provided by base r methods related to model fitting.
Wickham, H., and T. L. Pedersen. 2023. gtable: Arrange "Grobs" in tables.
Wickham, H., T. L. Pedersen, and D. Seidel. 2023f. scales: Scale functions for visualization.
Wickham, H., C. O. Wilke, and T. L. Pedersen. 2022b. isoband: Generate isolines and isobands from regularly spaced elevation grids.
Wickham, H., and Y. Xie. 2023. evaluate: Parsing and evaluation tools that provide more details than the default.
Wilke, C. O. 2023a. cowplot: Streamlined plot theme and plot annotations for "ggplot2".
Wilke, C. O. 2023b. ggridges: Ridgeline plots in "ggplot2".
Xiao, N. 2023. ggsci: Scientific journal and sci-fi themed color palettes for "ggplot2".
Xie, Y. 2014. knitr: A comprehensive tool for reproducible research in R. *in* V. Stodden, F. Leisch, and R. D. Peng, editors. Implementing reproducible computational research. Chapman; Hall/CRC.
Xie, Y. 2015. Dynamic documents with R and knitr. 2nd edition. Chapman; Hall/CRC, Boca Raton, Florida.
Xie, Y. 2019. TinyTeX: A lightweight, cross-platform, and easy-to-maintain LaTeX distribution based on TeX live. TUGboat 40:30–32.
Xie, Y. 2021. mime: Map filenames to MIME types.
Xie, Y. 2023a. knitr: A general-purpose package for dynamic report generation in r.
Xie, Y. 2023b. tinytex: Helper functions to install and maintain TeX live, and compile LaTeX documents.
Xie, Y. 2023c. xfun: Supporting functions for packages maintained by "Yihui Xie".
Xie, Y., J. J. Allaire, and G. Grolemund. 2018. R markdown: The definitive guide. Chapman; Hall/CRC, Boca Raton, Florida.
Xie, Y., J. Cheng, and X. Tan. 2023. DT: A wrapper of the JavaScript library "DataTables".
Xie, Y., C. Dervieux, and E. Riederer. 2020. R markdown cookbook. Chapman; Hall/CRC, Boca Raton, Florida.
Xie, Y., and Y. Qiu. 2022. highr: Syntax highlighting for r source code.

Zeileis, A. 2004. Econometric computing with HC and HAC covariance matrix estimators. Journal of Statistical Software 11:1–17.

Zeileis, A. 2006. Object-oriented computation of sandwich estimators. Journal of Statistical Software 16:1–16.

Zeileis, A., J. C. Fisher, K. Hornik, R. Ihaka, C. D. McWhite, P. Murrell, R. Stauffer, and C. O. Wilke. 2020a. colorspace: A toolbox for manipulating and assessing colors and palettes. Journal of Statistical Software 96:1–49.

Zeileis, A., and G. Grothendieck. 2005. zoo: S3 infrastructure for regular and irregular time series. Journal of Statistical Software 14:1–27.

Zeileis, A., K. Hornik, and P. Murrell. 2009. Escaping RGBland: Selecting colors for statistical graphics. Computational Statistics & Data Analysis 53:3259–3270.

Zeileis, A., and T. Hothorn. 2002. Diagnostic checking in regression relationships. R News 2:7–10.

Zeileis, A., S. Köll, and N. Graham. 2020b. Various versatile variances: An object-oriented implementation of clustered covariances in R. Journal of Statistical Software 95:1–36.

Zeileis, A., D. Meyer, and K. Hornik. 2007. Residual-based shadings for visualizing (conditional) independence. Journal of Computational and Graphical Statistics 16:507–525.

Bibliography

Allaire, J., Y. Xie, C. Dervieux, J. McPherson, J. Luraschi, K. Ushey, A. Atkins, H. Wickham, J. Cheng, W. Chang, and R. Iannone. 2023. rmarkdown: Dynamic documents for r.

Antoine Lucas, D. E. with contributions by, J. Tuszynski, H. Bengtsson, S. Urbanek, M. Frasca, B. Lewis, M. Stokely, H. Muehleisen, D. Murdoch, J. Hester, W. Wu, Q. Kou, T. Onkelinx, M. Lang, V. Simko, K. Hornik, R. Neal, K. Bell, M. de Queljoe, I. Suruceanu, B. Denney, D. Schumacher, W. Chang, D. Attali, and M. Chirico. 2024. digest: Create compact hash digests of r objects.

Auguie, B. 2017. gridExtra: Miscellaneous functions for "Grid" graphics.

Barnier, J., F. Briatte, and J. Larmarange. 2023. questionr: Functions to make surveys processing easier.

Barrett, T., M. Dowle, and A. Srinivasan. 2023. data.table: Extension of "data.frame".

Bartoń, K. 2023. MuMIn: Multi-model inference.

Bates, D., and D. Eddelbuettel. 2013. Fast and elegant numerical linear algebra using the RcppEigen package. Journal of Statistical Software 52:1–24.

Bates, D., M. Mächler, B. Bolker, and S. Walker. 2015. Fitting linear mixed-effects models using lme4. Journal of Statistical Software 67:1–48.

Bates, D., and M. Maechler. 2023. MatrixModels: Modelling with sparse and dense matrices.

Bates, D., K. M. Mullen, J. C. Nash, and R. Varadhan. 2023. minqa: Derivative-free optimization algorithms by quadratic approximation.

Beleites, C. 2020. arrayhelpers: Convenience functions for arrays.

Bengtsson, H. 2003a. The R.oo package - object-oriented programming with references using standard R code. *in* K. Hornik, F. Leisch, and A. Zeileis, editors. Proceedings of the 3rd international workshop on distributed statistical computing (DSC 2003). https://www.r-project.org/conferences/DSC-2003/Proceedings/, Vienna, Austria.

Bengtsson, H. 2003b. The R.oo package - object-oriented programming with references using standard R code. *in* K. Hornik, F. Leisch, and A. Zeileis, editors. Proceedings of the 3rd international workshop on distributed statistical computing (DSC 2003). https://www.r-project.org/conferences/DSC-2003/Proceedings/, Vienna, Austria.

Bengtsson, H. 2022. R.cache: Fast and light-weight caching (memoization) of objects and results to speed up computations.

Bengtsson, H. 2023a. matrixStats: Functions that apply to rows and columns of matrices (and to vectors).

Bengtsson, H. 2023b. R.utils: Various programming utilities.

Berwin A. Turlach R port by Andreas Weingessel <Andreas.Weingessel@ci.tuwien.ac.at> Fortran contributions from Cleve Moler dpodi/LINPACK), S. original by. 2019. quadprog: Functions to solve quadratic programming problems.

Bivand, R. 2023. classInt: Choose univariate class intervals.

Bryan, J. 2016. cellranger: Translate spreadsheet cell ranges to rows and columns.

Bryan, J., C. Citro, and H. Wickham. 2023. gargle: Utilities for working with google APIs.

Bürkner, P.-C., J. Gabry, M. Kay, and A. Vehtari. 2023. posterior: Tools for working with posterior distributions.

Champely, S. 2020. pwr: Basic functions for power analysis.

Chang, W. 2021. R6: Encapsulated classes with reference semantics.

Chang, W. 2023a. cachem: Cache r objects with automatic pruning.

Chang, W. 2023b. fastmap: Fast data structures.

Chang, W., and J. Cheng. 2023. later: Utilities for scheduling functions to execute later with event loops.

Chang, W., J. Cheng, J. Allaire, C. Sievert, B. Schloerke, Y. Xie, J. Allen, J. McPherson, A. Dipert, and B. Borges. 2023. shiny: Web application framework for r.

Cheng, J. 2018. miniUI: Shiny UI widgets for small screens.

Cheng, J. 2023. promises: Abstractions for promise-based asynchronous programming.

Cheng, J., W. Chang, S. Reid, J. Brown, B. Trower, and A. Peslyak. 2023a. httpuv: HTTP and WebSocket server library.

Cheng, J., T. Mastny, R. Iannone, B. Schloerke, and C. Sievert. 2023b. sass: Syntactically awesome style sheets ("Sass").

Cheng, J., and C. Sievert. 2023. crosstalk: Inter-widget interactivity for HTML widgets.

Cheng, J., C. Sievert, B. Schloerke, W. Chang, Y. Xie, and J. Allen. 2023c. htmltools: Tools for HTML.

Constantin, A.-E., and I. Patil. 2021. ggsignif: R package for displaying significance brackets for "ggplot2". PsyArxiv.

Csardi, G. 2023a. prettyunits: Pretty, human readable formatting of quantities.

Csardi, G. 2023b. rematch: Match regular expressions with a nicer "API".

Csardi, G., and S. Sorhus. 2015. praise: Praise users.

Csárdi, G. 2019. pkgconfig: Private configuration for "R" packages.

Csárdi, G. 2020. rematch2: Tidy output from regular expression matching.

Csárdi, G. 2022. crayon: Colored terminal output.

Csárdi, G., and W. Chang. 2022. callr: Call r from r.

Csárdi, G., and W. Chang. 2023. processx: Execute and control system processes.

Csárdi, G., and R. FitzJohn. 2023. progress: Terminal progress bars.

Csárdi, G., J. Hester, H. Wickham, W. Chang, M. Morgan, and D. Tenenbaum. 2023a. remotes: R package installation from remote repositories, including "GitHub".

Csárdi, G., K. Müller, and J. Hester. 2023b. desc: Manipulate DESCRIPTION files.

Dahl, D. B., D. Scott, C. Roosen, A. Magnusson, and J. Swinton. 2019. xtable: Export tables to LaTeX or HTML.

Eddelbuettel, D. 2013. Seamless R and C++ integration with Rcpp. Springer, New York.

Eddelbuettel, D., and J. J. Balamuta. 2018. Extending R with C++: A Brief Introduction to Rcpp. The American Statistician 72:28–36.

Eddelbuettel, D., R. Francois, J. Allaire, K. Ushey, Q. Kou, N. Russell, I. Ucar, D. Bates, and J. Chambers. 2024. Rcpp: Seamless r and c++ integration.

Eddelbuettel, D., and R. François. 2011. Rcpp: Seamless R and C++ integration. Journal of Statistical Software 40:1–18.

Firth, D. 2023. qvcalc: Quasi variances for factor effects in statistical models.

FitzJohn, R. 2017. ids: Generate random identifiers.

Fortran code by Alan Miller, T. L. based on. 2020. leaps: Regression subset selection.

Fox, J. 2003. Effect displays in R for generalised linear models. Journal of Statistical Software 8:1–27.

Fox, J., and J. Hong. 2009. Effect displays in R for multinomial and proportional-odds logit models: Extensions to the effects package. Journal of Statistical Software 32:1–24.

Fox, J., and S. Weisberg. 2018. Visualizing fit and lack of fit in complex regression models with predictor effect plots and partial residuals. Journal of Statistical Software 87:1–27.

Fox, J., and S. Weisberg. 2019a. An R companion to applied regression. Third. Sage, Thousand Oaks CA.

Fox, J., and S. Weisberg. 2019b. An r companion to applied regression. 3rd edition. Sage, Thousand Oaks CA.

Fox, J., S. Weisberg, and B. Price. 2022. carData: Companion to applied regression data sets.

Friendly, M. 2023. vcdExtra: "vcd" extensions and additions.

Gabry, J., and T. Mahr. 2022. bayesplot: Plotting for bayesian models.

Gabry, J., D. Simpson, A. Vehtari, M. Betancourt, and A. Gelman. 2019. Visualization in bayesian workflow. J. R. Stat. Soc. A 182:389–402.

Gagolewski, M. 2022. stringi: Fast and portable character string processing in R. Journal of Statistical Software 103:1–59.

Galili, T. 2015. dendextend: An r package for visualizing, adjusting, and comparing trees of hierarchical clustering. Bioinformatics.

Garbett, S. P., J. Stephens, K. Simonov, Y. Xie, Z. Dong, H. Wickham, J. Horner, reikoch, W. Beasley, B. O'Connor, G. R. Warnes, M. Quinn, and Z. N. Kamvar. 2023. yaml: Methods to convert r data to YAML and back.

Garnier, Simon, Ross, Noam, Rudis, Robert, Camargo, A. Pedro, Sciaini, Marco, Scherer, and Cédric. 2023a. viridis(Lite) - colorblind-friendly color maps for r.

Garnier, Simon, Ross, Noam, Rudis, Robert, Camargo, A. Pedro, Sciaini, Marco, Scherer, and Cédric. 2023b. viridis(Lite) - colorblind-friendly color maps for r.

Gaslam, B. 2021. diffobj: Diffs for r objects.

Gaslam, B. 2023. fansi: ANSI control sequence aware string functions.

Genz, A., and F. Bretz. 2009. Computation of multivariate normal and t probabilities. Springer-Verlag, Heidelberg.

Gilbert, P., and R. Varadhan. 2019. numDeriv: Accurate numerical derivatives.

Graves, S., H.-P. Piepho, and L. S. with help from Sundar Dorai-Raj. 2023. multcompView: Visualizations of paired comparisons.

Grosjean, P. 2023. SciViews-r. UMONS, MONS, Belgium.

Halekoh, U., and S. Højsgaard. 2014. A kenward-roger approximation and parametric bootstrap methods for tests in linear mixed models – the R package pbkrtest. Journal of Statistical Software 59:1–30.

Heather Turner, D. F. with contributions from, and J. Fox. 2016. relimp: Relative contribution of effects in a regression model.

Henry, L., and H. Wickham. 2022. tidyselect: Select from a set of strings.

Henry, L., and H. Wickham. 2023. lifecycle: Manage the life cycle of your package functions.

Hester, J., and J. Bryan. 2024. glue: Interpreted string literals.

Hester, J., and G. Csárdi. 2023. brio: Basic r input output.

Hester, J., L. Henry, K. Müller, K. Ushey, H. Wickham, and W. Chang. 2024. withr: Run code "With" temporarily modified global state.

Hester, J., H. Wickham, and J. Bryan. 2023a. vroom: Read and write rectangular text data quickly.

Hester, J., H. Wickham, and G. Csárdi. 2023b. fs: Cross-platform file system operations based on "libuv".

Horst, A. M., A. P. Hill, and K. B. Gorman. 2020. palmerpenguins: Palmer archipelago (antarctica) penguin data.

Hothorn, T. 2023. TH.data: TH's data archive.

Hothorn, T., F. Bretz, and P. Westfall. 2008. Simultaneous inference in general parametric models. Biometrical Journal 50:346–363.

Iannone, R. 2023. fontawesome: Easily work with "Font Awesome" icons.

Johnson, S. G. ? The NLopt nonlinear-optimization package. ? ?
Justin Talbot. 2023. labeling: Axis labeling.
Kassambara, A. 2023a. ggpubr: "ggplot2" based publication ready plots.
Kassambara, A. 2023b. rstatix: Pipe-friendly framework for basic statistical tests.
Kassambara, A., and F. Mundt. 2020. factoextra: Extract and visualize the results of multivariate data analyses.
Kay, M. 2023b. ggdist: Visualizations of distributions and uncertainty.
Kay, M. 2023a. tidybayes: Tidy data and geoms for Bayesian models.
Kay, M. 2024. ggdist: Visualizations of distributions and uncertainty in the grammar of graphics. IEEE Transactions on Visualization and Computer Graphics:1–11.
Koenker, R. 2021. SparseM: Sparse linear algebra.
Koenker, R. 2023. quantreg: Quantile regression.
Lang, M. 2017. checkmate: Fast argument checks for defensive R programming. The R Journal 9:437–445.
Lang, M., and R Core Team. 2021. backports: Reimplementations of functions introduced since r-3.0.0.
Langfelder, P., and S. Horvath. 2012. Fast R functions for robust correlations and hierarchical clustering. Journal of Statistical Software 46:1–17.
Larmarange, J. 2023. labelled: Manipulating labelled data.
Lê, S., J. Josse, and F. Husson. 2008. FactoMineR: A package for multivariate analysis. Journal of Statistical Software 25:1–18.
Lenth, R. 2022. estimability: Tools for assessing estimability of linear predictions.
Lenth, R. V. 2023. emmeans: Estimated marginal means, aka least-squares means.
Ligges, U., and M. Mächler. 2003. Scatterplot3d - an r package for visualizing multivariate data. Journal of Statistical Software 8:1–20.
Lincoln, M. 2022. clipr: Read and write from the system clipboard.
Loden, J., D. Daeschler, G. Rodola', and G. Csárdi. 2023. ps: List, query, manipulate system processes.
Lüdecke, D., M. S. Ben-Shachar, I. Patil, P. Waggoner, and D. Makowski. 2021. performance: An R package for assessment, comparison and testing of statistical models. Journal of Open Source Software 6:3139.
Lüdecke, D., P. Waggoner, and D. Makowski. 2019. insight: A unified interface to access information from model objects in R. Journal of Open Source Software 4:1412.
Lumley, T. 2004. Analysis of complex survey samples. Journal of Statistical Software 9:1–19.
Lumley, T. 2010. Complex surveys: A guide to analysis using r: A guide to analysis using r. John Wiley; Sons.
Lumley, T. 2019. mitools: Tools for multiple imputation of missing data.
Lumley, T. 2023. survey: Analysis of complex survey samples.
Makowski, D., M. S. Ben-Shachar, and D. Lüdecke. 2019. bayestestR: Describing effects and their uncertainty, existence and significance within the bayesian framework. Journal of Open Source Software 4:1541.
Meyer, D., and C. Buchta. 2022. proxy: Distance and similarity measures.
Meyer, D., E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch. 2023a. e1071: Misc functions of the department of statistics, probability theory group (formerly: E1071), TU wien.
Meyer, D., A. Zeileis, and K. Hornik. 2006. The strucplot framework: Visualizing multi-way contingency tables with vcd. Journal of Statistical Software 17:1–48.
Meyer, D., A. Zeileis, K. Hornik, and M. Friendly. 2023b. vcd: Visualizing categorical data.
Müller, K. 2020. here: A simpler way to find your files.
Müller, K. 2023. rprojroot: Finding files in project subdirectories.
Müller, K., and L. Walthert. 2023. styler: Non-invasive pretty printing of r code.
Murdoch, D., and E. D. Chow. 2023. ellipse: Functions for drawing ellipses and ellipse-like confidence regions.
Nenadic, O., and M. Greenacre. 2007. Correspondence analysis in r, with two- and three-dimensional

graphics: The ca package. Journal of Statistical Software 20:1–13.
Neuwirth, E. 2022. RColorBrewer: ColorBrewer palettes.
O'Hara-Wild, M., M. Kay, and A. Hayes. 2023. distributional: Vectorised probability distributions.
Oehlschlägel, J., and B. Ripley. 2022. bit: Classes and methods for fast memory-efficient boolean selections.
Oehlschlägel, J., and L. Silvestri. 2020. bit64: A S3 class for vectors of 64bit integers.
Ooms, J. 2023a. askpass: Password entry utilities for r, git, and SSH.
Ooms, J. 2023b. commonmark: High performance CommonMark and github markdown rendering in r.
Ooms, J. 2023c. curl: A modern and flexible web client for r.
Ooms, J. 2023d. openssl: Toolkit for encryption, signatures and certificates based on OpenSSL.
Ooms, J. 2023e. sys: Powerful and reliable tools for running system commands in r.
Patil, I., D. Makowski, M. S. Ben-Shachar, B. M. Wiernik, E. Bacher, and D. Lüdecke. 2022. datawizard: An R package for easy data preparation and statistical transformations. Journal of Open Source Software 7:4684.
Pedersen, T. L. 2023. textshaping: Bindings to the "HarfBuzz" and "Fribidi" libraries for text shaping.
Pedersen, T. L. 2024. patchwork: The composer of plots.
Pedersen, T. L., B. Nicolae, and R. François. 2022. farver: High performance colour space manipulation.
Pedersen, T. L., J. Ooms, and D. Govett. 2023. systemfonts: System native font finding.
Peng, R. D. 2019. simpleboot: Simple bootstrap routines.
Perry, P. O. 2023. utf8: Unicode text processing.
Plate, T., and R. Heiberger. 2016. abind: Combine multidimensional arrays.
Plummer, M., N. Best, K. Cowles, and K. Vines. 2006. CODA: Convergence diagnosis and output analysis for MCMC. R News 6:7–11.
Potter, S. 2012. Introducing the selectr package. The University of Auckland, Auckland, New Zealand.
R Core Team. 2023. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.
R Special Interest Group on Databases (R-SIG-DB), H. Wickham, and K. Müller. 2024. DBI: R database interface.
Ratnakumar, S., T. Mick, and T. Davis. 2021. rappdirs: Application directories: Determine where to save data, caches, and logs.
Sievert, C., and J. Cheng. 2021. jquerylib: Obtain "jQuery" as an HTML dependency object.
Sievert, C., J. Cheng, and G. Aden-Buie. 2023. bslib: Custom "Bootstrap" "Sass" themes for "shiny" and "rmarkdown".
Slowikowski, K. 2024. ggrepel: Automatically position non-overlapping text labels with "ggplot2".
Spinu, V. 2023. timechange: Efficient manipulation of date-times.
Stauffer, R., G. J. Mayr, M. Dabernig, and A. Zeileis. 2009. Somewhere over the rainbow: How to make effective use of colors in meteorological visualizations. Bulletin of the American Meteorological Society 96:203–216.
Turner, H., and D. Firth. 2023. gnm: Generalized nonlinear models.
Urbanek, S. 2015. base64enc: Tools for base64 encoding.
Urbanek, S., and T. Ts'o. 2023. uuid: Tools for generating and handling of UUIDs.
Ushey, K. 2023. sourcetools: Tools for reading, tokenizing and parsing r code.
Ushey, K., and H. Wickham. 2023. renv: Project environments.
Vaidyanathan, R., Y. Xie, J. Allaire, J. Cheng, C. Sievert, and K. Russell. 2023. htmlwidgets: HTML widgets for r.
van den Boogaart, K. G. 2023. tensorA: Advanced tensor arithmetic with named indices.
Vaughan, D. 2023. tzdb: Time zone database information.
Vaughan, D., J. Hester, and R. François. 2023. cpp11: A c++11 interface for r's c interface.
Vehtari, A., A. Gelman, D. Simpson, B. Carpenter, and P.-C. Bürkner. 2021. Rank-normalization, folding, and localization: An improved rhat for assessing convergence of MCMC (with discussion). Bayesian

Analysis.
Venables, B., K. Hornik, and M. Maechler. 2022. polynom: A collection of functions to implement a class for univariate polynomial manipulations.
Wei, T., and V. Simko. 2021. R package "corrplot": Visualization of a correlation matrix.
Wheeler, B., and M. Torchiano. 2016. lmPerm: Permutation tests for linear models.
Wickham, C. 2018. munsell: Utilities for using munsell colours.
Wickham, H. 2007. Reshaping data with the reshape package. Journal of Statistical Software 21:1–20.
Wickham, H. 2011a. The split-apply-combine strategy for data analysis. Journal of Statistical Software 40:1–29.
Wickham, H. 2011b. testthat: Get started with testing. The R Journal 3:5–10.
Wickham, H. 2019. lazyeval: Lazy (non-standard) evaluation.
Wickham, H. 2021. ellipsis: Tools for working with ...
Wickham, H. 2023a. blob: A simple S3 class for representing vectors of binary data ("BLOBS").
Wickham, H. 2023b. waldo: Find differences between r objects.
Wickham, H., M. Averick, J. Bryan, W. Chang, L. D. McGowan, R. François, G. Grolemund, A. Hayes, L. Henry, J. Hester, M. Kuhn, T. L. Pedersen, E. Miller, S. M. Bache, K. Müller, J. Ooms, D. Robinson, D. P. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, and H. Yutani. 2019. Welcome to the tidyverse. Journal of Open Source Software 4:1686.
Wickham, H., W. Chang, J. Hester, and L. Henry. 2023a. pkgload: Simulate package installation and attach.
Wickham, H., R. François, and L. D'Agostino McGowan. 2023b. emo: Easily insert "Emoji".
Wickham, H., L. Henry, T. L. Pedersen, T. J. Luciani, M. Decorde, and V. Lise. 2023c. svglite: An "SVG" graphics device.
Wickham, H., L. Henry, and D. Vaughan. 2023d. vctrs: Vector helpers.
Wickham, H., J. Hester, W. Chang, K. Müller, and D. Cook. 2021. memoise: "Memoisation" of functions.
Wickham, H., J. Hester, and G. Csárdi. 2023e. pkgbuild: Find tools needed to build r packages.
Wickham, H., M. Kuhn, and D. Vaughan. 2022a. generics: Common S3 generics not provided by base r methods related to model fitting.
Wickham, H., and T. L. Pedersen. 2023. gtable: Arrange "Grobs" in tables.
Wickham, H., T. L. Pedersen, and D. Seidel. 2023f. scales: Scale functions for visualization.
Wickham, H., C. O. Wilke, and T. L. Pedersen. 2022b. isoband: Generate isolines and isobands from regularly spaced elevation grids.
Wickham, H., and Y. Xie. 2023. evaluate: Parsing and evaluation tools that provide more details than the default.
Wilke, C. O. 2023a. cowplot: Streamlined plot theme and plot annotations for "ggplot2".
Wilke, C. O. 2023b. ggridges: Ridgeline plots in "ggplot2".
Xiao, N. 2023. ggsci: Scientific journal and sci-fi themed color palettes for "ggplot2".
Xie, Y. 2014. knitr: A comprehensive tool for reproducible research in R. *in* V. Stodden, F. Leisch, and R. D. Peng, editors. Implementing reproducible computational research. Chapman; Hall/CRC.
Xie, Y. 2015. Dynamic documents with R and knitr. 2nd edition. Chapman; Hall/CRC, Boca Raton, Florida.
Xie, Y. 2019. TinyTeX: A lightweight, cross-platform, and easy-to-maintain LaTeX distribution based on TeX live. TUGboat 40:30–32.
Xie, Y. 2021. mime: Map filenames to MIME types.
Xie, Y. 2023a. knitr: A general-purpose package for dynamic report generation in r.
Xie, Y. 2023b. tinytex: Helper functions to install and maintain TeX live, and compile LaTeX documents.
Xie, Y. 2023c. xfun: Supporting functions for packages maintained by "Yihui Xie".
Xie, Y., J. J. Allaire, and G. Grolemund. 2018. R markdown: The definitive guide. Chapman; Hall/CRC, Boca Raton, Florida.
Xie, Y., J. Cheng, and X. Tan. 2023. DT: A wrapper of the JavaScript library "DataTables".

Xie, Y., C. Dervieux, and E. Riederer. 2020. R markdown cookbook. Chapman; Hall/CRC, Boca Raton, Florida.

Xie, Y., and Y. Qiu. 2022. highr: Syntax highlighting for r source code.

Zeileis, A. 2004. Econometric computing with HC and HAC covariance matrix estimators. Journal of Statistical Software 11:1–17.

Zeileis, A. 2006. Object-oriented computation of sandwich estimators. Journal of Statistical Software 16:1–16.

Zeileis, A., J. C. Fisher, K. Hornik, R. Ihaka, C. D. McWhite, P. Murrell, R. Stauffer, and C. O. Wilke. 2020a. colorspace: A toolbox for manipulating and assessing colors and palettes. Journal of Statistical Software 96:1–49.

Zeileis, A., and G. Grothendieck. 2005. zoo: S3 infrastructure for regular and irregular time series. Journal of Statistical Software 14:1–27.

Zeileis, A., K. Hornik, and P. Murrell. 2009. Escaping RGBland: Selecting colors for statistical graphics. Computational Statistics & Data Analysis 53:3259–3270.

Zeileis, A., and T. Hothorn. 2002. Diagnostic checking in regression relationships. R News 2:7–10.

Zeileis, A., S. Köll, and N. Graham. 2020b. Various versatile variances: An object-oriented implementation of clustered covariances in R. Journal of Statistical Software 95:1–36.

Zeileis, A., D. Meyer, and K. Hornik. 2007. Residual-based shadings for visualizing (conditional) independence. Journal of Computational and Graphical Statistics 16:507–525.

A. Note sur les devoirs et Rmarkdown

Pour les devoirs, veuillez soumettre les rapports en utilisant le fichier Rmarkdown fourni. Le fichier inclue l'énoncé du devoir, les questions et la structure pour intégrer le code R, les sorties R et vos interprétations.

Après avoir modifié et sauvegardé le fichier `Rmd`, il faut générer le rapport dynamique en format `pdf`. Pour ce faire, il faut avoir installer une distribution de latex. Le plus simple est d'installer `tinytex`. Pour ce faire il faut installer l'extension R `tinytex` et à partir de decette extension installer la distribution latex `tinytex`.

```
install.packages("tinytex")
tinytex::install_tinytex()
```

B. Petit guide pour Rmarkdown

B.1. Syntaxe Markdown

Ceci est un petit document qui résume la syntaxe de Markdown

On utilise # (6 max.) pour créer des titres (section) :

```
### Titre 3

#### Titre 4

#### Titre 5

##### Titre 6
```

Donne

B.1.1. Titre 3

B.1.1.1. Titre 4

B.1.1.2. Titre 5

B.1.1.2.1. Titre 6

On entoure le texte des symboles * ou _ pour mettre le texte en italique ou en gras. Un symbole pour mettre en italique: `*Texte en Italique*` *Texte en Italique*

`_Texte en Italique_` *Texte en Italique*

Deux symboles pour mettre en gras: `**Texte gras**` **Texte gras**

`__Texte gras__` **Texte gras**

On utilise une combinaison des 2 charactère pour obtenir du **texte gras et *italique***.

Pour créer un liste on utilise le symbole - suivit d'une espace.

- Il doit y avoir une ligne vide avant la liste.
- On doit ajouter quelques espaces après chaque ligne.
  - Pour ajouter des niveau à la liste on indente (tab) 2 fois, puis on ajoute le symbole + suivit d'un espace.
  - Ici encore, on doit ajouter quelques espaces après chaque lignes.
- Il doit y avoir une ligne vide après la liste.

On peut aussi créer des liste avec des numéro ou des lettre.

1. On doit simplement mettre un point après le chiffre ou la lettre, suivit d'un espace

    a. Encore une fois, il peut y avoir plusieurs niveaux ii). Mais pas plus de 3 niveau

Pour écrire des équation mathématique, on entoure le texte du symbole $

$K = 0.5$

$\log(x + k)$

Pour faire un tableau, on fait comme suit (noter que l'alignement des symbole est sans importance) :

```
colonne 1  |colonne 2|colonne 3|colonne 4
---------|--------|--------|-----------
cellulle | cellulle |cellulle| cellulle
cellulle   | cellulle |cellulle| cellulle
cellulle | cellulle |cellulle| cellulle
```

On peut utiliser le symbole > pour indenter un paragraphe (le résultat est différent entre PDF et HTML) :

> En HTML, ça produit un résultat étrange … > mais bon …

On peut écrire dans la même police que celle du code en entourant le texte avec le symbole ‘ : `variable`

Si l'on veut utilisé un symbole dans le texte sans qu'il modifie le texte, il faut mettre le symbole \ devant.

Pour inséré un lien, on entoure le texte avec [ ], et on met l'addresse dans des paranthèse () immédiatement à coté: R Markdown

On peut tracer un ligne horizontale en répétant le symbole - 3 fois ou plus :

B.2. À propos de LaTex

Ce qui suit concerne uniquement ceux qui on installer LaTex. Les fonction LaTex serve à modifier le texte, mais ne fonctionne que lorsqu'on produit un document PDF (ça fonctionne peut-être aussi pour les document word). Ils en existe beaucoup trop de fonction pour toutes les énumérer ici. Google est votre ami.

Voici quelques exemple:

```
\emph{texte italique}
\textcolor{red}{texte rouge}
\texttt{même police que le code de R}
```

Le nom de la prochaine fonction est assez descriptif

```
\pagebreak
```

B.3. Concernant le code dans notebook

Vous devez écrire le code à l'intérieur de bloc de code, sinon il ne sera pas interpréter comme du code, mais comme du texte. Vous pouvez taper directement la notation pour créer un bloc code ou utiliser les raccourci clavier de RStudio Pour créer un nouveau bloc de code, taper ```` ```{r} ```` pour ouvrir le bloc et ```` ``` ```` pour le fermer. Sinon, appuyer sur *Ctrl+Alt+I* Dans Rstudio, pour éxécuter le code à l'intérieur d'un bloc appuyer sur *Ctrl+Shift+Enter* ou cliquer sur le boutton *Run* (dans le bloc, en haut à droite).

Charger les données et les packages dans le premier bloc de code.

````
```{r} #ouverture du bloc de code
code
``` #fermeture du bloc code
````

Lorsque vous exécuter du code dans un bloc, le résultat apparait directement en dessous de la du bloc (si votre document est un R Notebook). L'option `fig.cap` fonctionnement seulement lorsqu'on génère un PDF.

Lorsque vous sauvegarder un notebook, un fichier HTLM contenant le code et les résultats est sauvegarder en même temps. Cliquer sur le bouton *Preview* ou appuyer sur *Ctrl+Shift+K* pour voir le fichier HTML.

````
```{r, fig.cap = "Exemple de graphique tiré du fichier d'aide de la fonction stat_bin"}
ggplot(diamonds, aes(carat)) +
  geom_histogram()

ggplot(diamonds, aes(carat)) +
  geom_histogram(binwidth = 0.01)

ggplot(diamonds, aes(carat)) +
  geom_histogram(bins = 200)
```
````

```
ggplot(diamonds, aes(carat)) +
  geom_histogram()
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

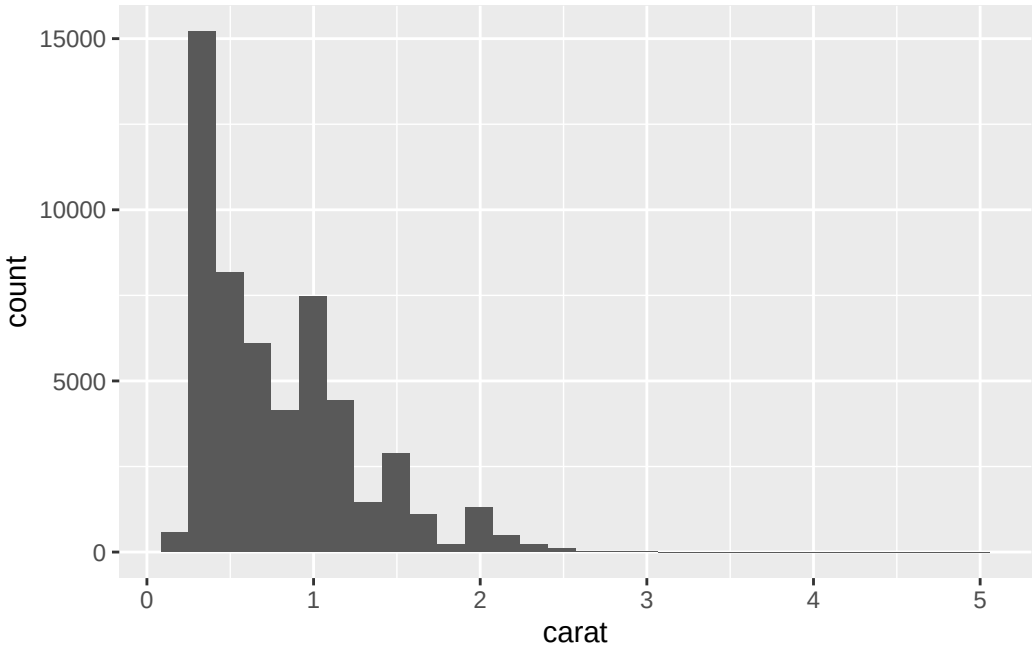


Figure B.1.: Exemple de graphique tiré du fichier d'aide de la fonction stat_bin

```
ggplot(diamonds, aes(carat)) +
  geom_histogram(binwidth = 0.01)
```

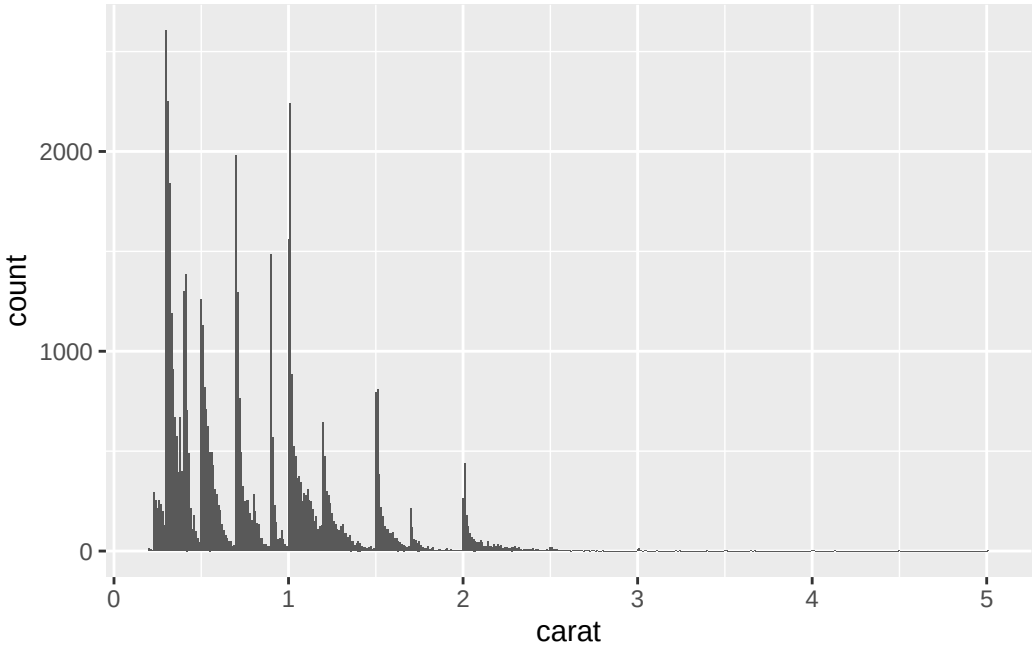


Figure B.2.: Exemple de graphique tiré du fichier d'aide de la fonction stat_bin

```
ggplot(diamonds, aes(carat)) +
  geom_histogram(bins = 200)
```

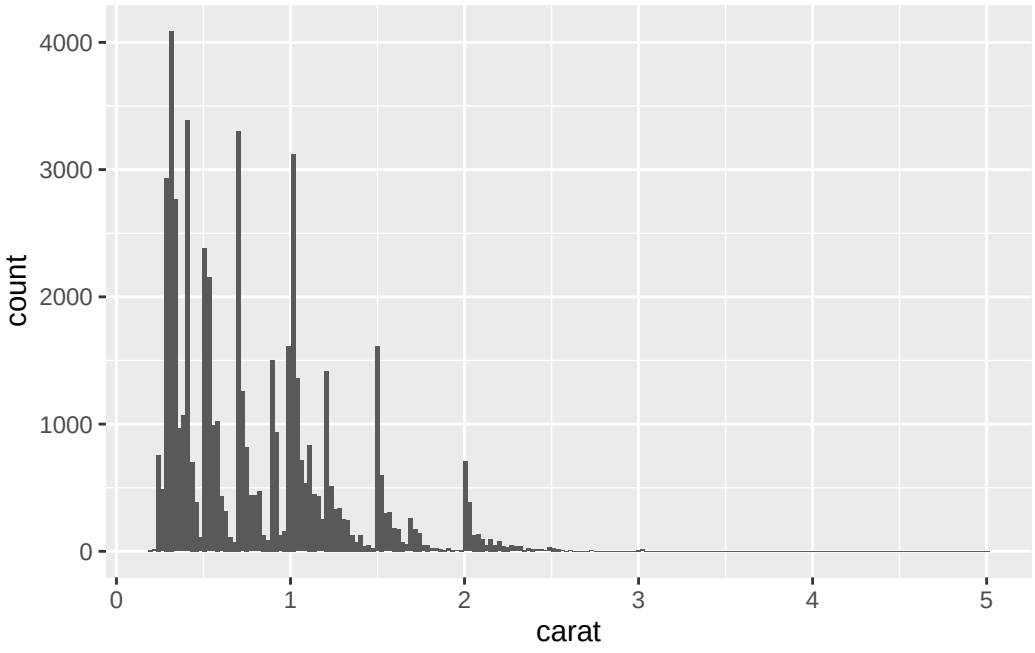


Figure B.3.: Exemple de graphique tiré du fichier d'aide de la fonction stat_bin

C. Resources pour en apprendre plus sur rmarkdown

https://statistique-et-logiciel-r.com/guide-de-demarrage-en-r-markdown/

https://rmarkdown.rstudio.com/