L'analyse de données écologiques avec R

Charles A. Martin

2025-04-04

Table des matières

Bienvenue

Ce site web contient l'ensemble des notes de cours, exercices et exemples accumulés au fil des années pour les cours Biologie Quantitative (STT1039) et Interprétation des données écologiques (ECL1012) du baccalauréat en biologie-écologie.

Le matériel est rassemblé ici sous forme de site web et de livre électronique[1] afin d'assurer une pérennité à ce contenu au-delà des 15 semaines de cours. Le contenu sera bonifié et amélioré au fil du temps, mais demeurera en tout temps un texte d'introduction aux différents concepts.

Ce livre ne remplacera jamais la consultation de volumes de statistiques de référence ou d'articles scientifiques. En ce sens, il ne devrait jamais être cité tel quel dans une publication.

Merci à l'avance et bonne lecture!

Charles.

[1] Version PDF

1. Introduction

1.1. Pourquoi les statistiques?

Ah la grande question, pourquoi avons-nous besoin de cours de statistiques dans un baccalauréat de biologie-écologie? Ce n'est probablement pas ce qui vous a amené à choisir cette formation. Et pourtant, on vous impose non seulement un, mais deux cours de statistiques!

La raison de ce choix est simple : la nature est extrêmement variable. Chaque phénomène que l'on observe, chaque animal que l'on mesure, chaque arbre que l'on coupe est différent des autres. Mais pourtant, votre aurez besoin pour travailler de connaître certaines généralités : la paruline jaune se retrouve surtout dans les bosquets près des cours d'eau, le phosphore augmente la quantité de cyanobactéries dans les lacs, le CO2 dans l'atmosphère cause les changements climatiques, etc.

Comment arrive-t-on à ces conclusions si tout est si variable?

1.2. Exemple concret

Vous achetez un sachet de semences de coriandre. Une fois à la maison, vous prenez deux pots identiques que vous remplissez de la même quantité de terre. Vous ajoutez la même quantité d'eau et vous installez les pots côte à côte devant la même fenêtre. Vous prenez ensuite votre

sachet de graines, vous séparez le contenu en deux parties égales, et vous en plantez une moitié dans un pot et l'autre moitié dans l'autre.

Trois semaines plus tard, votre coriandre a assez poussé pour la récolter. Par curiosité, vous pesez la récolte des deux pots pour la comparer. Inévitablement les récoltes ne seront pas parfaitement égales. Dans ce cas-ci, la récolte du premier pot est 9 g plus importante que l'autre.

Quelque temps plus tard, un de vos amis vous dit que votre coriandre poussera vraiment mieux si vous ajoutez une cuillère à thé d'engrais à votre eau. Vous répétez donc l'expérience, mais cette-fois, vous ajoutez de l'engrais dans la préparation d'un des pots. Vous récoltez votre coriandre quelques semaines plus tard, et votre pot avec l'engrais contient 10 g de plus de coriandre que son voisin.

Est-ce que c'est l'engrais a fait effet ou vous avez été chanceuse et que la variabilité normale entre les récoltes a favorisé par hasard le pot dans lequel vous aviez mis de l'engrais?

Et si vous recommencez l'expérience 10 fois, et que 7 des 10 fois le pot avec l'engrais vous donne une plus grande récolte, est-ce que l'engrais fait une différence ou c'est encore la chance?

Ce genre de question est beaucoup plus complexe à résoudre qu'il n'y paraît, et vous n'aurez pas de trop de deux cours de 45 heures pour assimiler toutes les nuances.

Sachez en terminant qu'une fois sur le marché du travail, la majorité des étudiants de programmes de biologie et d'écologie auraient souhaité (eh oui!) avoir plus de cours sur les méthodes d'analyse de données[1].

[1] https://peerj.com/articles/285/

1.3. Pourquoi le logiciel R?

La question qui suit normalement le questionnement sur les statistiques est : pourquoi faut-il utiliser R? Il me semble que Excel ça fait le travail non? La réponse est que oui, Excel faisait probablement le travail pour ce que vous avez eu à faire dans le passé, mais il ne sera pas complètement approprié pour ce que vous aurez à affronter dans le premier cours, et surtout pas dans le deuxième.

Il faut cependant l'avouer, R n'est pas un logiciel facile d'approche. Lorsque vous lancez R la première fois, vous obtenez essentiellement un écran vide, avec un curseur qui clignote, qui attend que vous saisissiez des commandes. Comme les ordinateurs fonctionnaient au début des années 1980... on est vraiment loin de l'iPad! R est essentiellement un langage de programmation. Pour le pire, mais aussi pour le meilleur.

Les avantages d'utiliser R sont nombreux. Le premier, est qu'il est gratuit! Chaque fois que quelqu'un vous mentionnera que les choses sont plus simples dans Systat ou un autre logiciel du même genre qui possède une interface graphique poussée, gardez en mémoire que ce genre de logiciel coûte souvent plusieurs centaines, voire des milliers de dollars.

Mais outre les arguments financiers, R de par sa nature de langage de programmation ouvert, est extrêmement versatile. En quelques secondes, vous pouvez avoir au bout de vos doigts, dans le même logiciel, le nécessaire pour créer des cartes interactives[2], trouver des visages dans une image[3], mesurer l'ouverture dans une canopée forestière[4] ou détecter des chants d'oiseaux dans un MP3[5].

Il existe de nombreux autres langages de programmation avec des capacités statistiques que nous pourrions utiliser outre que R, entre autres

[2] https://numerilab.io/en/workshops/R_GIS_and_leaflet
[3] https://rive-numeri-lab.github.io/workshops/ML
[4] https://github.com/cmartin/ImageAnalysisPrimer
[5] https://besjournals.onlinelibrary.wiley.com/doi/full/10.1111/2041-210X.12624

Python ou Julia. Par contre, peu de langages ont des communautés en ligne aussi accueillantes et ouvertes que celles organisées autour de R. Dans le monde parfois cruel des communautés virtuelles, il est rassurant pour une débutante de savoir que les interactions à propos de R sont les plus agréables sur le site de questions en ligne Stack Overflow[6]. Vous pouvez aussi en quelques clics trouver des conférences et des formations gratuites sur un ensemble de sujets[7], souvent organisées par des femmes ou des minorités visibles[8].

Entre autres, la réponse de la communauté R a été exemplaire face aux comportements inappropriés d'un dirigeant de DataCamp et la tentative de la compagnie camoufler l'histoire[9]. Dans un monde informatique empreint de machisme et souvent contrôlé par des hommes blancs occidentaux, cet aspect n'est pas à négliger.

1.4. Faciliter les choses

Apprendre à programmer n'est pas simple. C'est un métier en soi. Chaque fois que vous aurez des ennuis de programmation, rappelez-vous que **c'est normal d'en arracher**. Lors de ma première session en informatique au CÉGEP, nous avons passé les deux premiers mois de la session à programmer sur papier, sans toucher à l'ordinateur, pour s'assurer de bien maîtriser les concepts de base. Nous n'avons malheureusement pas ce temps disponible ici. Sachez néanmoins qu'il est normal d'avoir une fenêtre de Google ouverte à côté de celle de R pour chercher des trucs. Même après 10 ans de R, je dois constamment consulter internet pour retrouver des trucs que j'ai oublié comment faire. On vous en demande

[6] https://hackernoon.com/which-programming-languages-have-the-happiest-and-angriest-commenters-ebe91b3852ed#.5j0qkm174
[7] https://satrdays.org/
[8] https://rladies.org/
[9] https://www.computerworld.com/article/3389684/r-community-blasts-datacamp-response-to-execs-inappropriate-behavior.html

beaucoup d'apprendre à la fois les statistiques et la programmation, en deux sessions de 45 heures. Mon travail sera de vous faciliter les choses le plus possible pour que vous y arriviez. J'y travaillerai avec vous sur trois aspects.

D'abord, au niveau de votre interaction avec le logiciel R. Plutôt que de travailler avec R pur, je vous ferai travailler avec le logiciel RStudio. Ce dernier est une interface, une coquille en quelque sorte, qui emballe R pour le rendre plus accessible et convivial. Toutes les commandes que nous lancerons dans RStudio auraient fonctionné dans R pur aussi, mais nous aurons aussi accès à une quantité d'autres outils qui n'auraient pas été disponibles sinon.

Deuxièmement, je vais tenter de vous simplifier le code R à exécuter le plus possible. R n'est pas un langage de programmation très contraignant, ce qui signifie en pratique qu'il existe des dizaines de façons différentes d'effectuer la même tâche. Plutôt que de vous enseigner la façon de base de faire les choses, je vous conseillerai souvent d'utiliser des librairies de code supplémentaires. En particulier, nous discuterons souvent des libraries ggplot2[10] et dplyr[11], que nous utiliserons pour faire nos graphiques et manipuler nos bases de données. Ces deux librairies de code sont axées sur les mêmes principes : plutôt que d'écrire du code cryptique pensé pour l'ordinateur, elles nous permettent d'écrire de façon naturelle et lisible les opérations que nous désirons effectuer sur nos données. Ils définissent des grammaires de manipulation des données et de création de graphiques.

J'utilise quotidiennement ces deux librairies de code et je ne saurais m'en passer. Je me sentirais mal de vous faire utiliser la façon de base, quand ce n'est pas celle que j'utilise au quotidien. J'ai souvent, par le passé, initié des gens à R à l'aide de ces deux librairies avec grand succès. Sachez néanmoins que cette approche est un peu controversée. Plusieurs prônent plutôt une maîtrise du langage de base avant de passer aux

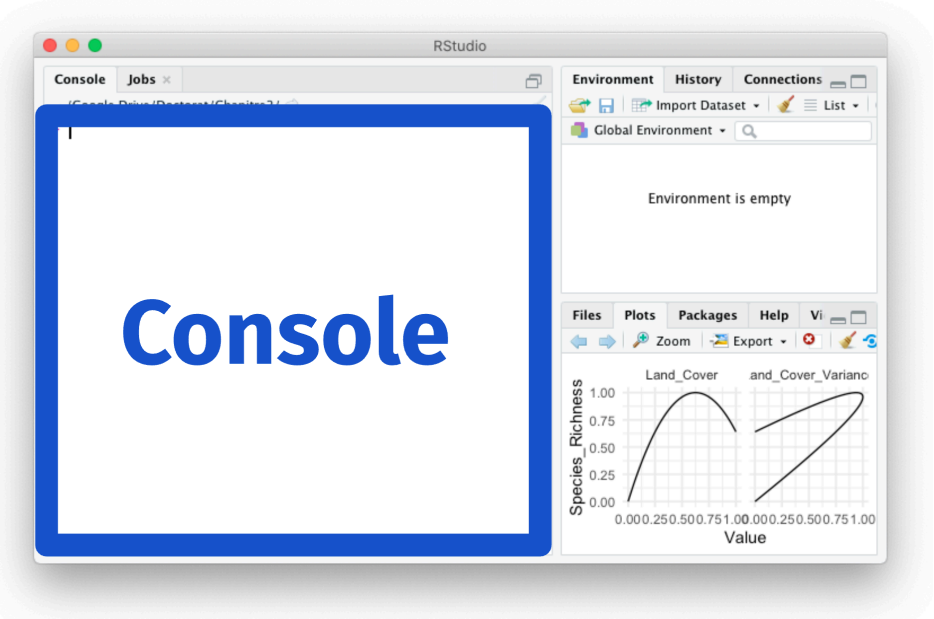
[10] https://ggplot2.tidyverse.org/
[11] https://dplyr.tidyverse.org/

librairies de code additionnelles. Personnellement, dans le cadre d'un cours de statistiques au baccalauréat, je crois que l'accent devrait être mis sur la facilité. Sur une accumulation de petites victoires. J'espère que cette approche vous conviendra.

Troisièmement, j'appliquerai aussi cette approche de facilitation à la partie théorique. Je ne suis pas un mathématicien. Je suis biologiste, j'ai déjà été programmeur, j'ai suivi beaucoup de cours de statistiques (bac, maîtrise, doctorat, formations intensives, etc.), j'ai dépanné des tonnes de gens dans leurs analyses statistiques. Mon point de vue sera surtout celui d'un praticien. Je vous présenterai à l'occasion certaines formules mathématiques, lorsque ce sera nécessaire et pertinent, mais j'essaierai surtout de vous fournir l'intuition menant à la compréhension du concept. Je ne vous demanderai jamais de me réciter une formule par cœur. Mon but sera de m'assurer que vous soyez capables de discuter des concepts, verbalement, avec vos collègues et vos supérieurs.

1.5. Labo : Vos premiers pas dans R

Si vous lancez le logiciel RStudio, vous devriez voir une fenêtre qui ressemble à celle-ci :



Il y a évidemment beaucoup d'éléments dans cette fenêtre. Mais ce qui nous intéresse pour le moment, c'est la console. La console est l'endroit où vous pouvez entrer des commandes et en recevoir le résultat. Vous tapez le code de votre commande, vous appuyez sur Entrée (ENTER) et R vous répond. Essayez par exemple d'entrer ce code :

```
1+1
```

R devrait vous répondre :

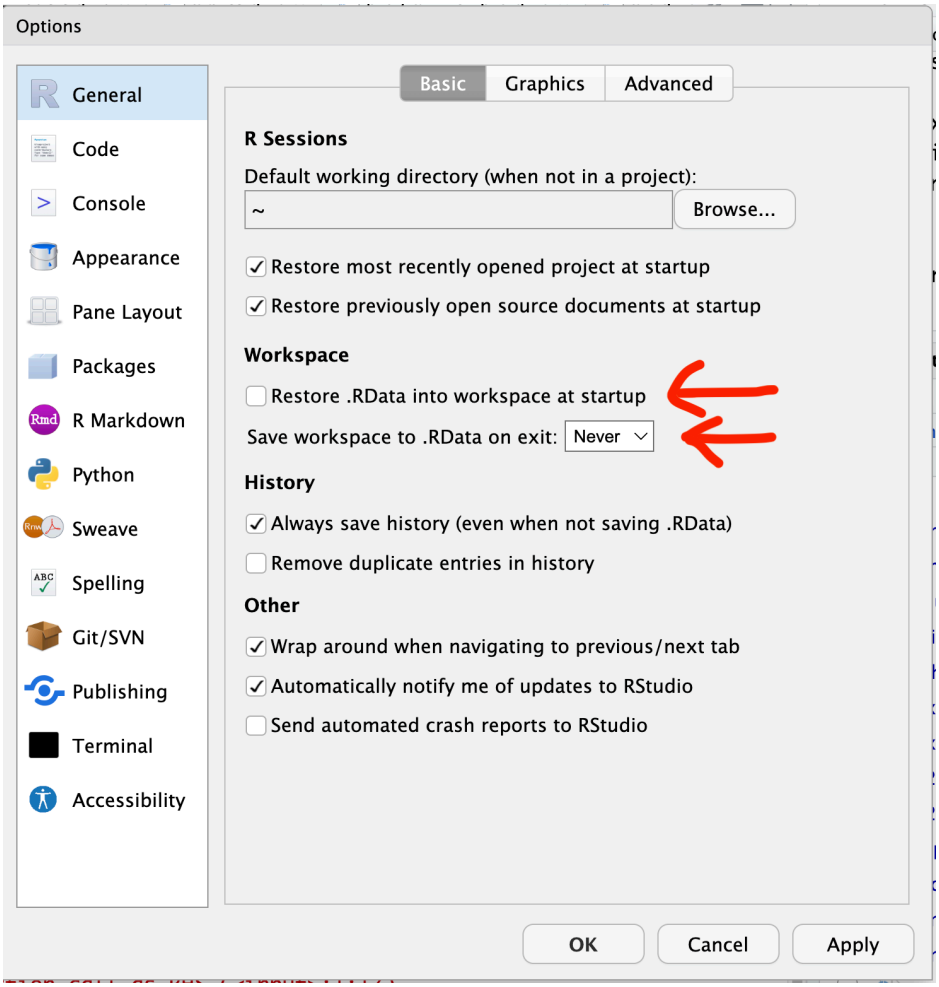
```
[1] 2
```

Notez que le symbole ">" dans votre console indique que R est prêt à recevoir une commande. Si jamais la console ne vous présente pas le symbole ">", appuyez sur la touche Échap (ESC) de votre clavier ou sur bouton "Stop" rouge dans l'interface de RStudio.

1.6. Options à changer

Bien que le logiciel RStudio soit un excellent logiciel, il y a quelques options pour lesquelles le choix des développeurs ne s'accorde pas directement avec les meilleures pratiques actuelles. C'est pourquoi je vous propose de changer 3 options pour vous faciliter la vie dans vos analyses. Ces options peuvent être trouvées dans le menu Outils > Options Globales (Tools > Global Options).

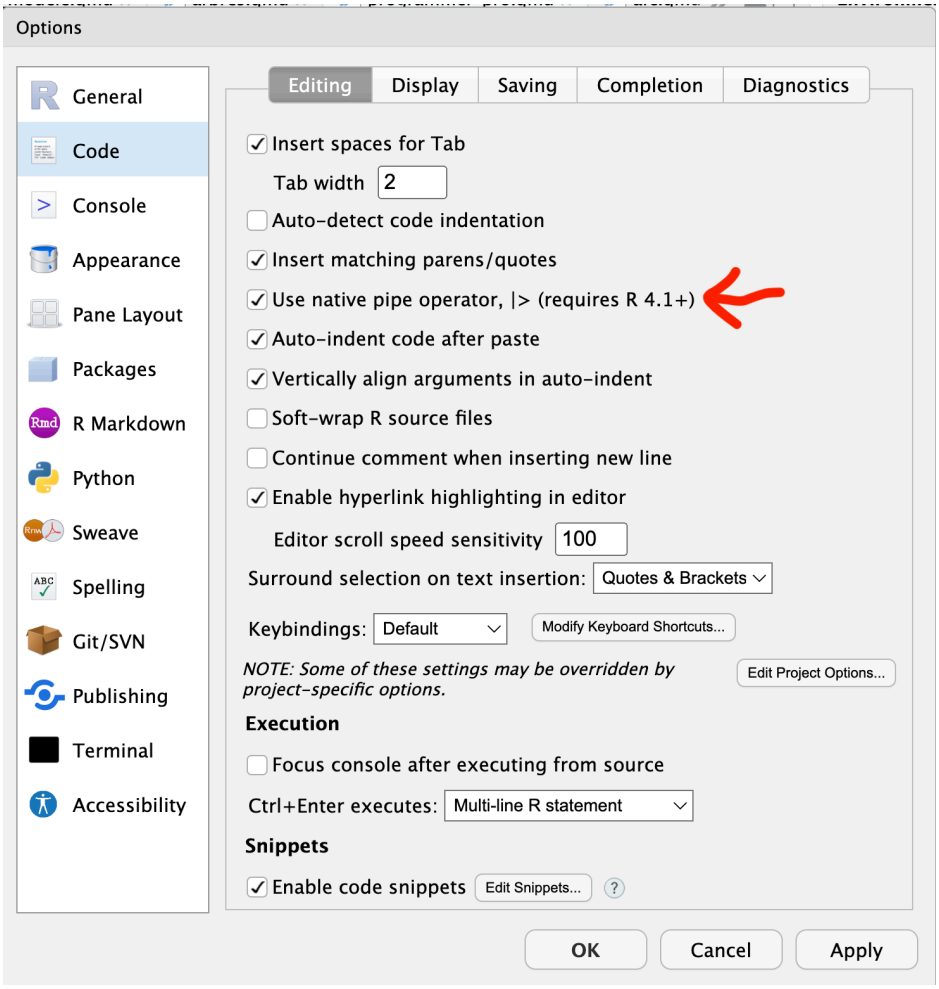
Tout d'abord, dans l'onglet Général, je vous propose de modifier les deux options suivantes :



Cela vous permettra d'avoir une session de R complètement propre/vide à redémarrage. Vous vous éviterez bien des ennuis de problèmes intermittants / non-reproductibles de cette façon.

Ensuite, dans l'onglet Code, je vous propose de cocher l'option permettant d'utiliser l'opérateur d'enchaînement (plus de détails à la

Section 6.3) natif de R plutôt que celui de la librairie magrittr. Cette option est décochée par défaut pour éviter les problèmes de compatiblité avec les versions plus vieilles de R, mais comme vous débutez, cela ne devra pas poser de problème pour vous.



Voilà, vous êtes maintenant prêtes à vous attaquer à vos premières no-

tions de visualisation de données!

partie I.

Les données

2. Les manchots de l'archipel Palmer

Comme vous le découvrirez bien assez tôt, charger nos propres données dans R n'est pas la tâche la plus simple lorsque l'on débute avec R. C'est pourquoi le début de ce livre sera axé sur un jeu de données déjà prêt pour nous, qui pourra nos accompagner au fil des chapitres et des apprentissages. Nous utiliserons pour cela des données provenant de la librairie de code `palmerpenguins`.

Ces données ont été recueillies sur 3 îles de l'archipel Palmer en Antarctique, de 2007 à 2009. Elles contiennent une séries de mesures morphologiques sur 3 espèces de manchots différentes présentes dans cet archipel, soit :

- Le manchot à jugulaire (*Chinstrap*)
- Le manchot papou (*Gentoo*) et
- Le manchot Adélie

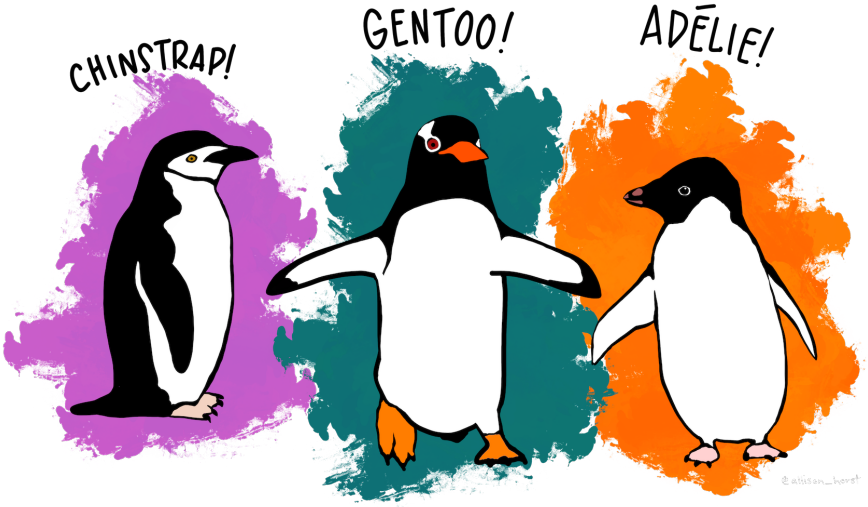


Figure 2.1.: Les manchots de l'archipel Palmer. Illustration par @allison_horst.

Afin de limiter la charger mentale de passer constamment du noms français au nom anglais et vice-versa dans ce livre, nous utiliserons systématiquement les noms anglais puisque ce sont ceux que nous verrons dans les tableaux de données et dans les sorties des modèles.

Les données disponibles pour chaque individu sont :

- Le nom de l'espèce (*species*)
- L'île où a été effectuée la mesure (*island*)
- La longueur du bec (*bill_length_mm*)
- L'épaisseur du bec (*bill_depth_mm*)
- La longueur des nagoires (*flipper_length_mm*)
- Le poids (*body_mass_g*)
- Le sexe (*sex*) et enfin
- L'année (*year*).

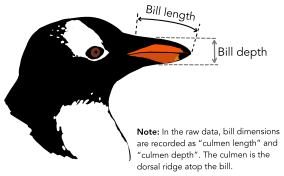


Figure 2.2.: Les mesures du bec des manchots de Palmer. Illustration par @allison_horst.

2.1. Les librairies de code dans R

Bien que nous n'ayons pas besoin nous-même de saisir les données concernant ces manchots, ces dernières ne sont pas incluses avec le logiciel R de base.

Comme nous l'avons vu dans l'introduction, R fait partie d'un écosystème complexe où beaucoup de gens contribuent du code ou des données gratuitement. Cela rend notre travail beaucoup plus agréable et rapide, mais implique aussi que l'on doit constamment installer ou mettre à jour ces contributions externes.

La façon classique de distribuer publiquement du code ou des données est de les inclure dans une librairie (*package*) déposée sur le serveur CRAN[1]. Cette organisation a pour mission d'effectuer un contrôle de qualité et d'offrir une infrastructure permettant de téléchager facilement ces librairies. Une librairie peut contenir du code R, des données, ou les deux.

[1] https://cran.r-project.org/

2.2. Labo : Installation et activation de la librairie `palmerpenguins`

Pour utiliser une librairie provenant de CRAN, il faut suivre deux étapes importantes soit :

1. Télécharger la librairie sur votre ordinateur

Cette étape n'a besoin d'être effectuée qu'une seule fois. C'est équivalent d'installer un nouveau logiciel sur votre ordinateur ou une nouvelle appli sur votre téléphone.

On appelle la fonction `install.packages`, et on lui passe comme argument, entre guillemets, le nom de la librairie désirée :

```
install.packages("palmerpenguins")
```

Notez que pour que cela fonctionne, votre ordinateur doit être connecté à internet pour toute la durée de l'installation.

> Astuce
>
> Cette opération n'a pas besoin d'être répétée à chaque fois que vous ouvrez R. Une fois la librairie installée, elle y est pour toujours (du moins, tant que vous ne changez pas d'ordinateur ou de version de R.)

L'installation de librairies ne fait donc pas partie du processus d'analyse de données comme tel.

2. Activation de la librairie

Pour le moment, la librairie `palmerpenguins` est sur le disque dur de votre ordinateur, mais elle n'est pas encore prête à être utilisée par R. Comme vous aurez rapidement des dizaines, voire des centaines de

librairies installées sur votre ordinateur, R trouve beaucoup plus poli d'attendre que vous lui demandiez explicitement avant de charger une librairie en mémoire.

Donc, chaque fois que vous voudrez utiliser une librairie, vous devrez l'activer à l'aide de la commande `library` :

```
library(palmerpenguins)
```

> **Mise en garde**
>
> Portez une attention particulière au fait que l'on doit mettre des guillemets autour du nom de la librairie au moment de l'installation, mais que ces derniers sont optionnels au moment de l'activation. Cette façon de faire a été maintes fois critiquée, mais demeure à ce jour le standard [2]

Maintenant que le contenu de la librairie est prêt à être utilisé, on peut, par exemple, jeter un coup d'oeil aux premières lignes du tableau de données `penguins` avec la fonction `head` :

```
head(penguins)
```

```
# A tibble: 6 x 8
  species island    bill_length_mm bill_depth_mm
  <fct>   <fct>              <dbl>         <dbl>
1 Adelie  Torgersen           39.1          18.7
2 Adelie  Torgersen           39.5          17.4
3 Adelie  Torgersen           40.3          18
4 Adelie  Torgersen           NA            NA
5 Adelie  Torgersen           36.7          19.3
6 Adelie  Torgersen           39.3          20.6
```

[2] https://stackoverflow.com/a/33709826/373222

```
# i 4 more variables: flipper_length_mm <int>,
#   body_mass_g <int>, sex <fct>, year <int>
```

Maintenant que nous connaissons un peu mieux les données avec lesquelles nous travaillerons, les prochains chapitres seront consacrées plus en détails, à leur exploration, leur visualisation, leur manipulation et leur description.

3. Voir les données

3.1. Piloter à vue

Une des choses les plus importantes que vous aurez à faire lors de vos analyses statistiques est de visualiser les données. Pierre Magnan, qui m'a enseigné le cours de Biologie Quantitative à l'époque, aimait bien comparer les statistiques au pilotage d'avion. Il répétait toujours que voir les données, c'est comme regarder par la fenêtre de l'avion pour voir ce qu'il y a autour. Avant de regarder les chiffres sur les cadrans, il est primordial de voir où l'on va, de connaître les obstacles. De même, en statistiques, il est important de toujours voir ce que l'on fait. Si vos yeux voient une chose et que les chiffres en disent une autre, croyez d'abord vos yeux et doutez des chiffres.

À titre d'illustration, observez ces quatre relations classiques décrites par Anscombe :

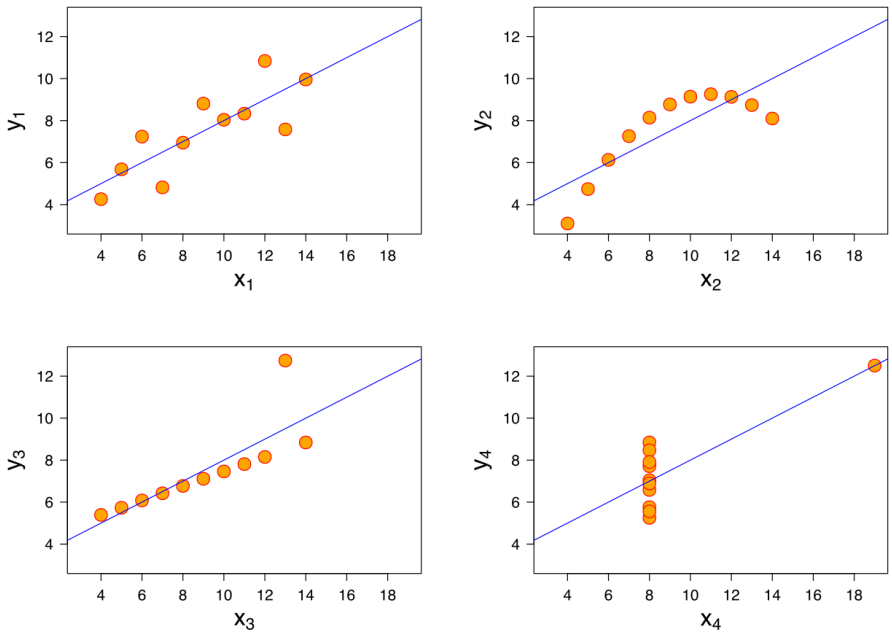


Figure 3.1.: Anscombe.svg: Schutz Avenue, CC BY-SA 3.0, via Wikimedia Commons

Bien que ces relations correspondent à des réalités complètement différentes, la grande majorité de leurs propriétés statistiques sont identiques. Elles présentent, entre autres, la même moyenne des X, la même moyenne des Y, la même variance des X et la même variance des Y (voir Section 5.2). Elles ont aussi la même corrélation (voir Chapitre 17), et la pente calculée par une régression linéaire donnerait exactement le même résultat (voir Chapitre 18). Gardez bien cette illustration en tête. Elle doit être un rappel constant que calculer des statistiques sans voir les données peut être extrêmement hasardeux, voire même dangereux

Il peut évidemment arriver des cas pointus, dans des modélisations com-

plexes, où l'on pourra trouver des choses qui nous avaient échappé au moment de l'exploration visuelle, mais comme pour le pilotage d'avion aux instruments dans le brouillard, il faut avoir beaucoup d'expérience au pilotage à vue avant d'en arriver là.

3.2. Les différents types des données

Avant de lancer R pour regarder des données, nous aurons besoin de définir les différents types de données que nous pourrons rencontrer, car nous y reviendrons fréquemment.

La première distinction à faire est entre les données quantitatives et les données qualitatives. Une **donnée quantitative** est une donnée à laquelle nous pouvons associer un chiffre : un arbre mesurant 30 cm de diamètre, un oiseau pesant 12 g, un nid contenant 6 oeufs. À l'inverse, une **donnée qualitative** n'a pas de chiffre associé : la couleur de vos yeux, la ville où vous êtes né, le nom du lac où un poisson a été capturé. Il s'agit de la distinction la plus importante.

Remarquez que souvent, les variables qualitatives sont des raccourcis, des généralisations, qu'il serait plus juste ou nuancé de décrire sur un gradient. La couleur des yeux, pourrait par exemple être décrite plus précisément à l'aide d'un triplet de variables quantitatives décrivant la quantité de rouge, de vert et de bleu (le fameux RGB[1]) plutôt que par un nom de couleur.

On peut par la suite diviser les données quantitatives en deux groupes : quantitatives continues et quantitatives discrètes. Les **données quantitatives continues** sont celles où, si notre instrument de mesure était plus précis, nous aurions pu ajouter des décimales à notre nombre. Pensons à un oiseau pesant 12 g. Avec une balance plus précise, nous aurions peut-être su qu'il pesait 12,5 g ou 12,47 g. Les **données quantitatives discrètes**

[1] https://fr.wikipedia.org/wiki/Rouge_vert_bleu

quant à elles ne peuvent être rien d'autres que des entiers. Par exemple le nombre d'œufs dans un nid, le nombre de graines dans un sac, etc.

Notez qu'il est possible, avec un peu de mauvaise foi (ou de malchance!) d'obtenir 0,5 œuf dans un nid. Il peut être aussi possible de passer d'une échelle qualitative à une échelle quantitative et vice versa, par exemple en passant de présence/absence à 0/1 ou l'inverse. Mais nous ne nous compliquerons pas la vie pour le moment avec ces cas plus pointus.

3.3. Labo : Préparatifs

Plutôt que d'utiliser les graphiques de base de R, nous utiliserons la très populaire librairie ggplot2. Cette dernière a été conçue pour implémenter ce que les auteurs ont nommé la grammaire des graphiques. Autrement dit, leur but est de nous fournir un vocabulaire nous permettant de décrire ce que l'on veut présenter dans nos graphiques.

Comme à la section suivante, nous devrons donc installer cette librairie externe avant de commencer à travailler :

```
install.packages("ggplot2")
```

Pour commencer à travailler, nous devrons ensuite activer cette librairie, de même que celle contenant nos données de manchots :

```
library(ggplot2)
library(palmerpenguins)
```

> **Astuce**
>
> Bien qu'il puisse être tentant (et rapide!) de copier-coller le code fournit dans ce livre, je vous conseille fortement d'écrire le code vous même dans la console de R. Vous retiendez beaucoup mieux l'information. Vous ferez bien sûr quelques erreurs supplémentaires se faisant, mais apprendre à reconnaître et corriger nos erreurs fait aussi partie des compétences à acquérir pour être confortable dans R. Autant le faire dès le début, dans des exemples simples.

3.4. Labo : Les types de données dans R

Les types de données décrits ci-hauts ont leurs équivalent dans le logiciel R. Voyons d'abord ce qui se produit lorsque l'on envoit le nom d'un tableau de données dans la console de R :

```
penguins
```

```
# A tibble: 344 x 8
   species island    bill_length_mm bill_depth_mm
   <fct>   <fct>              <dbl>         <dbl>
 1 Adelie  Torgersen           39.1          18.7
 2 Adelie  Torgersen           39.5          17.4
 3 Adelie  Torgersen           40.3          18
 4 Adelie  Torgersen           NA            NA
 5 Adelie  Torgersen           36.7          19.3
 6 Adelie  Torgersen           39.3          20.6
 7 Adelie  Torgersen           38.9          17.8
 8 Adelie  Torgersen           39.2          19.6
 9 Adelie  Torgersen           34.1          18.1
10 Adelie  Torgersen           42            20.2
```

```
# i 334 more rows
# i 4 more variables: flipper_length_mm <int>,
#   body_mass_g <int>, sex <fct>, year <int>
```

La première ligne nous informe que notre objet est de type `tibble`. Les tibbles sont un type de tableau de données dans R. Vous verrez aussi souvent des tableaux de données de type `data.frame`, qui se traitent de façon très semblable.

Cette ligne nous informe ensuite que notre tableau contient 344 lignes et 8 colonnes.

Les deux lignes suivantes nous donnent ensuite la liste des colonnes (*species*,*island*, *bill_length_mm*, etc.) et le type associé à chaque des colonnes (*fct*, *fct*, *dbl*, etc.)

Enfin, on a les valeurs pour chacune des lignes et colonnes du tableau.

Par défaut, un objet de type tibble ne nous montre qu'un aperçu intéressant qui entre correctement dans notre écran.

C'est pourquoi vous voyez au bas du tableau la mention `334 more rows` et `4 more variables ...`.

Pour les tableaux dédiés à l'analyse, R contient les 3 mêmes types de données que décrits plus haut, soit

- `double (dbl)` pour les données quantitatives continues
- `integer (int)` pour les données quantitatives discrètes et
- `factor (fct)` pour les données qualitatives.

Vous verrez aussi parfois le type `character (chr)` pour des données textuelles, qui ne sont pas encore considérées comme des variables qualitatives (nous en reparlerons au Chapitre 15).

Vous pouvez connaître le type de toutes les variables d'un tableau de données avec la fonction `str` (pour structure) :

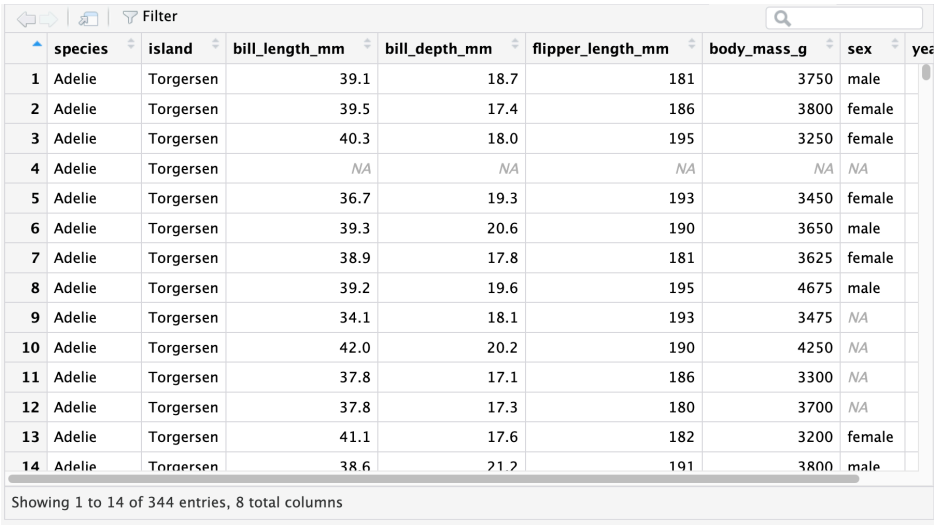
```
str(penguins)
```

```
tibble [344 x 8] (S3: tbl_df/tbl/data.frame)
 $ species          : Factor w/ 3 levels
 "Adelie","Chinstrap",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ island           : Factor w/ 3 levels
 "Biscoe","Dream",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ bill_length_mm   : num [1:344] 39.1 39.5 40.3 NA 36.7
 39.3 38.9 39.2 34.1 42 ...
 $ bill_depth_mm    : num [1:344] 18.7 17.4 18 NA 19.3
 20.6 17.8 19.6 18.1 20.2 ...
 $ flipper_length_mm: int [1:344] 181 186 195 NA 193 190
 181 195 193 190 ...
 $ body_mass_g      : int [1:344] 3750 3800 3250 NA 3450
 3650 3625 4675 3475 4250 ...
 $ sex              : Factor w/ 2 levels
 "female","male": 2 1 1 NA 1 2 1 2 NA NA ...
 $ year             : int [1:344] 2007 2007 2007 2007
 2007 2007 2007 2007 2007 2007 ...
```

Enfin, vous pouvez aussi explorer vos données de façon plus conventionnelle avec la fonction View :

```
View(penguins)
```

3. Voir les données



Filter

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	yea
1	Adelie	Torgersen	39.1	18.7	181	3750	male	
2	Adelie	Torgersen	39.5	17.4	186	3800	female	
3	Adelie	Torgersen	40.3	18.0	195	3250	female	
4	Adelie	Torgersen	NA	NA	NA	NA	NA	
5	Adelie	Torgersen	36.7	19.3	193	3450	female	
6	Adelie	Torgersen	39.3	20.6	190	3650	male	
7	Adelie	Torgersen	38.9	17.8	181	3625	female	
8	Adelie	Torgersen	39.2	19.6	195	4675	male	
9	Adelie	Torgersen	34.1	18.1	193	3475	NA	
10	Adelie	Torgersen	42.0	20.2	190	4250	NA	
11	Adelie	Torgersen	37.8	17.1	186	3300	NA	
12	Adelie	Torgersen	37.8	17.3	180	3700	NA	
13	Adelie	Torgersen	41.1	17.6	182	3200	female	
14	Adelie	Torgersen	38.6	21.2	191	3800	male	

Showing 1 to 14 of 344 entries, 8 total columns

Il est important de mentionner à ce moment-ci que R est un langage de programmation **sensible à la casse**, c'est-à-dire qu'il faut faire attention aux majuscules et aux minuscules lorsque l'on entre des commandes. Jusqu'à ce point, nous avons tout écrit en minuscules, mais la fonction View, elle, doit être écrite avec la première lettre en majuscule!

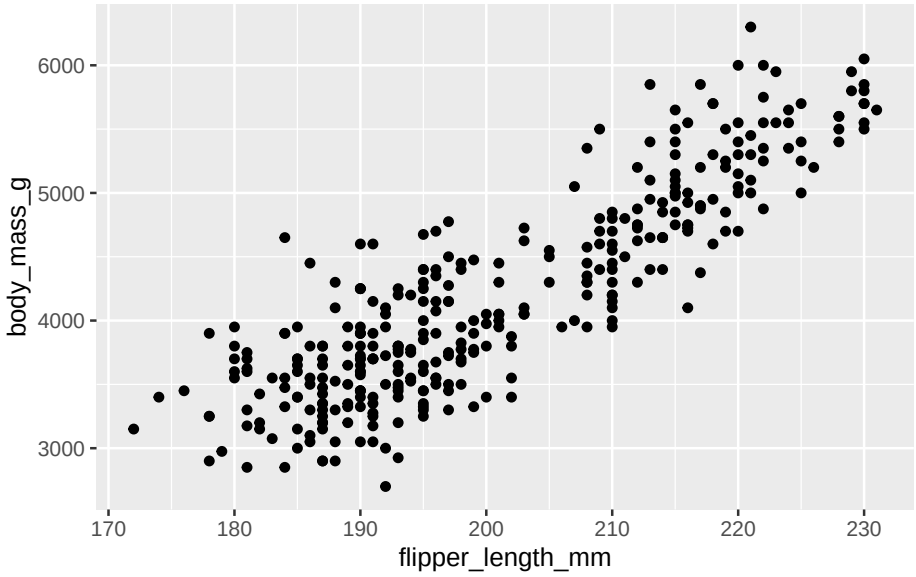
Observez bien la structure de la commande précédente. Elle contient deux éléments, le premier (`View`) est le nom de la **fonction** que nous voulons lancer. Les fonctions sont toujours accompagnées de parenthèses. Les parenthèses contiennent les **arguments** que l'on veut donner à la fonction, pour contrôler son fonctionnement. Ici, nous lui avons passé l'objet penguins. Nous avons donc demandé à R de voir (*to view*) notre tableau de données de manchots (`penguins`).

3.5. Labo : Un premier graphique

Maintenant que ces préliminaires sont derrière nous, lançons le code qui créera notre premier graphique :

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = flipper_length_mm, y =
   ↪  body_mass_g))
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```



Ce graphique nous montre la relation entre deux variables de notre tableau de données soit la longueur des ailes en X et le poids en Y.

3. Voir les données

Analysons tranquillement les morceaux de code qui ont permis de construire ce graphique. Sur la première ligne, on reconnaît une fonction (`ggplot`) et un argument (`data = penguins`). Tous vos graphiques de ggplot démarreront par cette fonction. C'est elle qui crée la base du graphique. L'argument `data=penguins` informe R que dans ce graphique, nous utiliserons des variables provenant du tableau de données `penguins` Contrairement à notre premier exemple avec View, notre argument `penguins` possède un nom (`data=`) qui indique à la fonction `ggplot` à quoi cet objet servira. Dans certains cas, on peut se permettre de ne pas nommer l'argument, mais nous commencerons par tous les nommer, question de clarté.

Cette première ligne se termine par le symbole `+`, pour indiquer à R que nous voulons ajouter des choses à ce graphique. Sur la ligne suivante, nous appelons la fonction `geom_point`. Cette fonction ajoute une couche à notre graphique. Toutes les **couches graphiques** de `ggplot` débutent par le préfixe `geom_`. Nous verrons plus loin qu'il en existe des dizaines d'autres. Dans ce cas particulier, nous avons demandé une couche de points. L'intérieur de la parenthèse sert ensuite à expliquer à R quoi mettre dans notre couche de points. L'argument `mapping` nous permet d'associer (*to map*) des propriétés du graphiques à des variables de notre tableau de données. On informe R qu'en X, nous voulons les valeurs de la variable `flipper_length_mm`, et en Y, nous voulons les valeurs de la variable `body_mass_g`.

La structure d'un graphique avec ggplot2 consiste donc à un appel à la fonction `ggplot`, auquel on associe notre tableau de données. On peut ensuite ajouter une (ou plusieurs) couche graphique (nommées `geom_`) dans laquelle on connecte les propriétés du graphique à celles du tableau de données. Il est capital de bien comprendre cette structure car elle se répètera pour tous vos graphiques avec cette librairie.

Vous aurez aussi peut-être remarqué que R, après avoir fait le graphique nous indique le message suivant :

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```

Pour le moment, cet avertissement est normal. R nous avertit qu'il y a 2 des lignes de notre tableau de données qui n'ont pas pu être affichées parce que une des variables demandée contenait des données manquantes. Nous verrons au Chapitre 4 comment filtrer ces données pour éliminer ce genre de messages.

3.6. Labo : Les propriétés graphiques

Nous avons vu dans notre premier exemple deux propriétés graphiques, soit les coordonnées X et Y d'un point. Il en existe plusieurs autres. Pour la couche de points (`geom_point`), vous avez accès, entre autres, à :

- `color` (la couleur du point)
- `size` (la taille du point)
- `alpha` (l'opacité, allant de 0 à 1)
- `shape` (maximum 6 formes)
- `fill` (la couleur de remplissage de certaines formes)

Pour obtenir de l'information sur une fonction dans R, vous pouvez taper l'opérateur "?" suivi du nom de votre fonction dans la console, p. ex. :

```
?geom_point
```

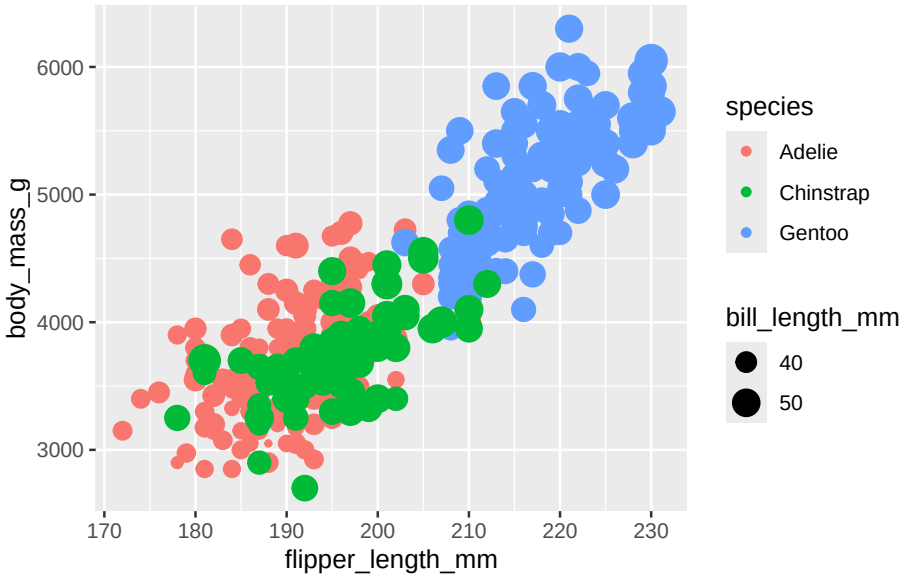
RStudio vous affiche alors dans le panneau Help l'aide de la fonction. Vous voyez entre autres dans cette aide la section Arguments, qui vous donne la liste de tous les arguments qui peuvent être acceptés par cette fonction. Pour les couches de ggplot2, vous avez aussi accès à une section

Aesthetics, qui contient la liste des propriétés de la couche auxquelles vous avez accès dans le mapping.

Mettons ces connaissances en pratique dans un deuxième graphique :

```
ggplot(data = penguins) +
  geom_point(mapping = aes(
    x = flipper_length_mm,
    y = body_mass_g,
    color = species,
    size = bill_length_mm
  )
)
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```

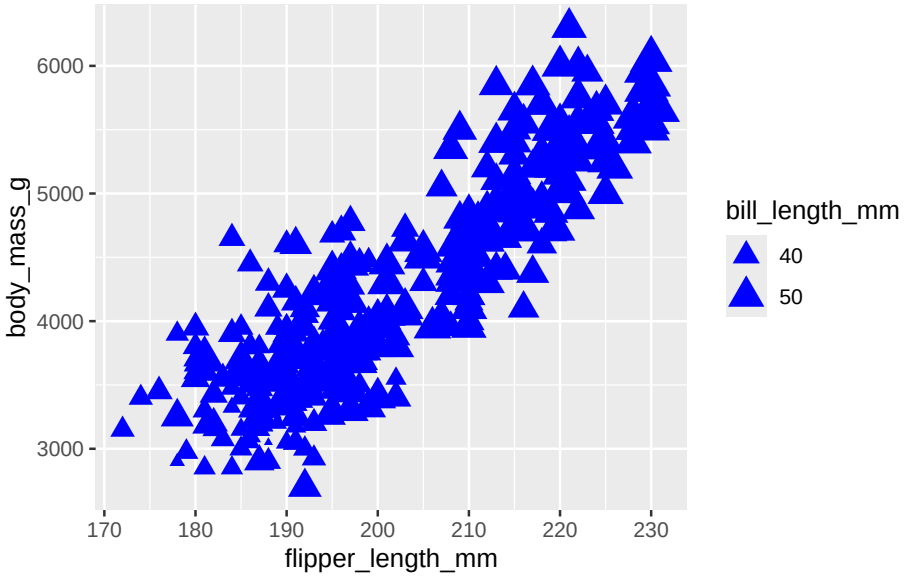


Dans cet exemple, nous avons utilisé 4 propriétés graphiques : X et Y sont associés, comme dans l'exemple précédent, à `flipper_length_mm` et `body_mass_g`, mais l'espèce (`species`) est maintenant associée à la couleur et la longueur du bec (`bill_length_mm`) est associée à la taille du point. Nous affichons donc 4 informations différentes par point. Notez que ggplot2 a ajouté pour vous les légendes pour les propriétés graphiques supplémentaires.

Si jamais vous voudriez modifier une des propriétés graphiques pour l'ensemble des points plutôt que de se baser sur les valeurs d'une variable, il faut simplement sortir cet argument du mapping, et mentionner directement la valeur désirée, p. ex. :

```
ggplot(data = penguins) +
  geom_point(
    mapping = aes(
      x = flipper_length_mm,
      y = body_mass_g,
      size = bill_length_mm
    ),
    color = "blue",
    shape = "triangle"
)
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```



Observez bien que X et Y sont toujours dans la parenthèse du mapping, alors que color et shape sont maintenant à l'extérieur. Remarquez que le nom de la couleur doit être mentionné entre guillemets et inscrit en anglais. La plupart des noms de couleurs qui vous viendront en tête sont déjà définis dans R, mais en cas de doutes, vous pouvez facilement trouver la liste des couleurs disponibles sur internet[2].

Pour les formes, elles peuvent aussi être nommées avec leur nom anglais, par exemple "square open", "cross", etc. La liste complète des noms de formes peut se trouver ici : https://github.com/tidyverse/ggplot2/pull/2338. Cette façon de faire, plus récente, est par contre moins connue et moins documentée que celle qui a longtemps prévalu, soit d'utiliser des numéros de formes[3].

[2] https://sape.inf.usi.ch/quick-reference/ggplot2/colour
[3] http://www.sthda.com/english/wiki/ggplot2-point-shapes#point-shapes-in-r

3.7. Labo : Choisir la bonne couche graphique

Il existe dans ggplot2 des dizaines de couches permettant de concevoir à peu près tous les graphiques imaginables. La difficulté consistera surtout à déterminer quelle couche graphique utiliser dans chaque situation, selon le type de données et ce que l'on veut montrer. Nous en verrons cinq principales, qui devraient satisfaire l'ensemble de nos besoins.

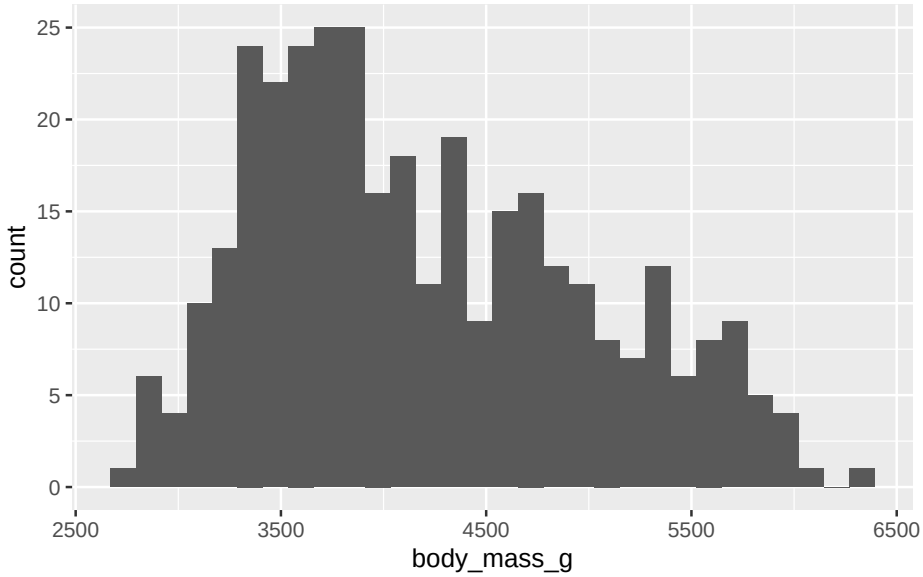
La distribution d'une variable quantitative

La première couche graphique que nous explorerons est **l'histogramme de fréquences**. L'histogramme est utile pour visualiser la distribution des valeurs d'une variable quantitative dans notre tableau de données. Il permet de savoir quelles valeurs sont communes et quelles valeurs sont rares. On peut, avec ggplot2 créer un histogramme en utilisant la couche graphique `geom_histogram` :

```
ggplot(data = penguins) +
  geom_histogram(mapping = aes(x = body_mass_g))
```

```
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```

```
Warning: Removed 2 rows containing non-finite outside
the scale
range (`stat_bin()`).
```

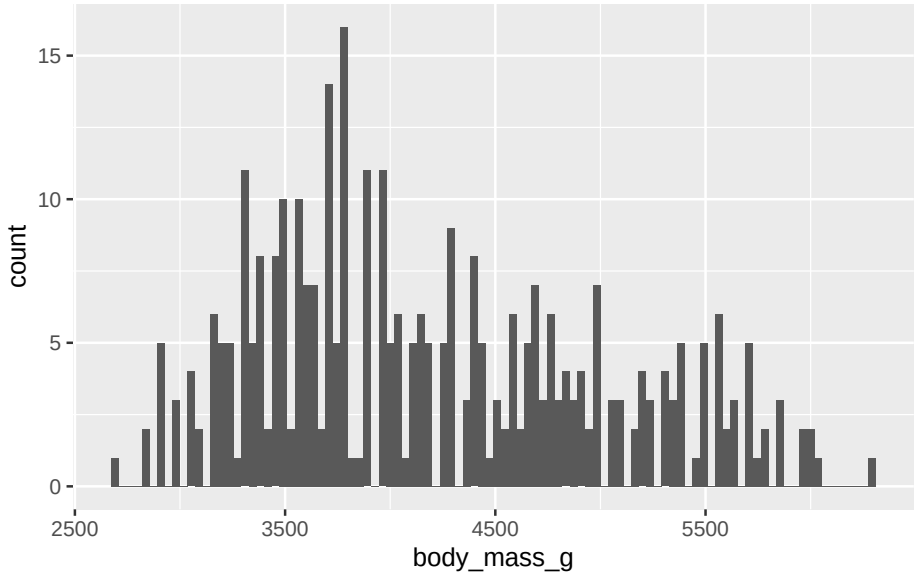


Dans un histogramme, l'axe des X est découpé par classes (*bins*), de largeurs égales. La hauteur de chaque bande (axe des Y) correspond au nombre d'observations de chaque classe de variable (leur **fréquence**). Dans notre exemple, la première classe (à gauche) contient par exemple 1 valeur, la deuxième en compte 6 et la plus commune en compte 25.

Notez que par défaut, ggplot2 choisit de créer 30 classes. C'est un nombre tout à fait arbitraire, qu'il vaut la peine de modifier pour voir si notre impression des données change. On peut modifier le nombre de classes avec l'argument `bins`, qu'il faut prendre soin de mettre à l'extérieur du mapping, p. ex. :

```
ggplot(data = penguins) +
    geom_histogram(
        mapping = aes(x = body_mass_g),
        bins = 100
    )
```

```
Warning: Removed 2 rows containing non-finite outside
the scale
range (`stat_bin()`).
```



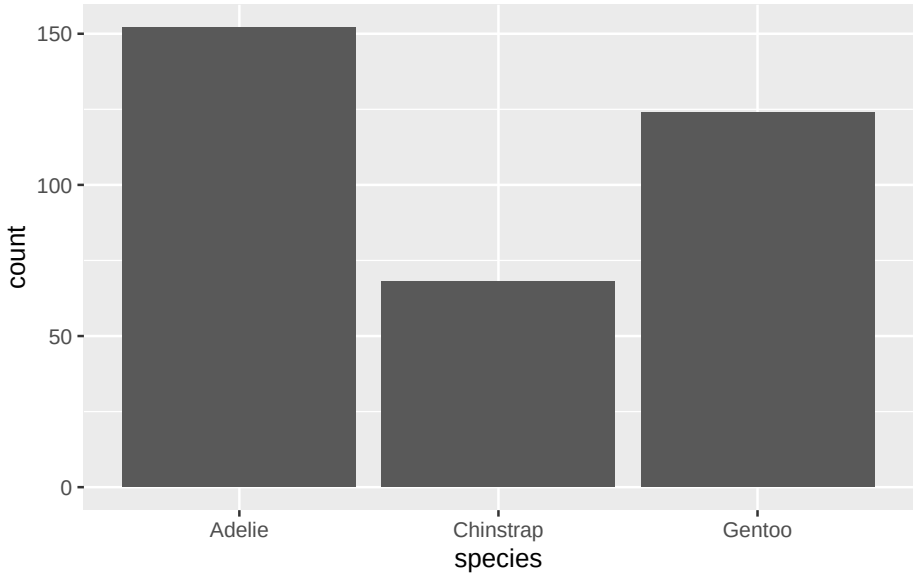
Vous verrez aussi parfois un histogramme avec un axe des Y différent, qui affiche les fréquences relatives plutôt que les fréquences absolues. La **fréquence relative** se définissant pour chaque classe comme le nombre d'observations dans cette classe, divisé par le nombre total d'observations.

La distribution d'une variable qualitative

Le deuxième graphique que nous verrons est très semblable à l'histogramme, il se nomme diagramme à bandes. Le **diagramme à bandes** permet de voir comment se répartissent les observations d'une variable qualitative (plutôt que quantitative dans le cas de l'histogramme). La couche de diagramme à bandes dans ggplot2 se nomme `geom_bar` :

3. Voir les données

```
ggplot(data = penguins) +
  geom_bar(mapping = aes(x = species))
```



On peut voir dans ce graphique que l'espèce la plus commune dans ces données est le manchot Adélie, et la plus rare est le manchot Chinstrap.

Notez enfin que dans le cas du diagramme à bandes, il existe visuellement un espace entre les bandes, alors qu'elles sont collées les unes sur les autres dans l'histogramme.

Prenez garde que la couche `geom_bar` s'attend à calculer elle-même le nombre d'observations pour chacune des bandes. Pour qu'elle fonctionne correctement, vos données doivent être organisées comme ceci :

species
Adelie
Chinstrap
Adelie
...

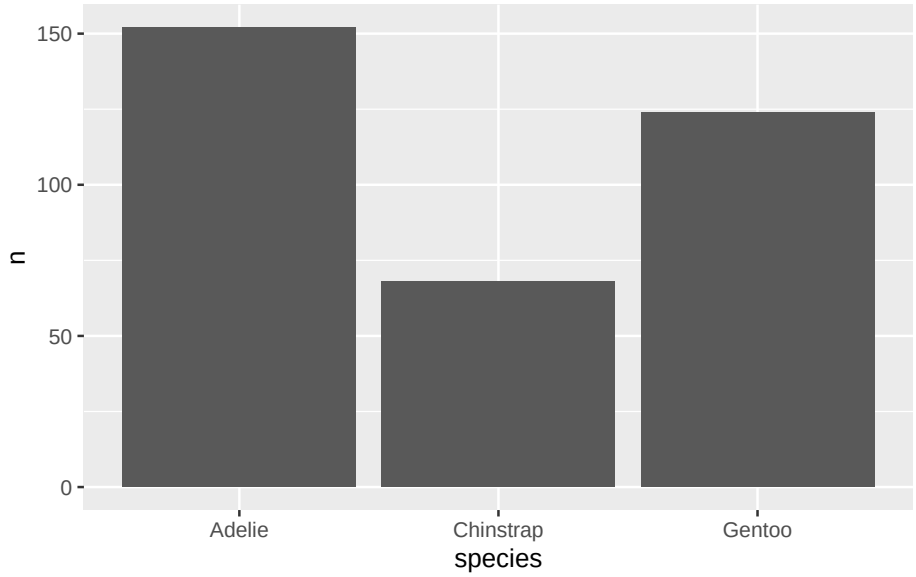
Si jamais la fréquence pour chaque bande est déjà calculée, par exemple comme cela :

species	n
Adelie	152
Chinstrap	68
Gentoo	124
...	...

Il faut plutôt utiliser la couche `geom_col` :

```
ggplot(data = tableau_deja_calcule) +
  geom_col(mapping = aes(x = species, y = n))
```

3. Voir les données



Si jamais vous voulez faire fonctionner l'exemple précédent, vous devez d'abord créer un tableau de données nommé `tableau_deja_calcule`, par exemple comme ceci :

```
library(dplyr)
tableau_deja_calcule <-
  penguins |>
  count(species)
```

Ce genre de code, très compact, sera détaillé à la Section 6.3 et nécessite l'installation de la librairie dplyr.

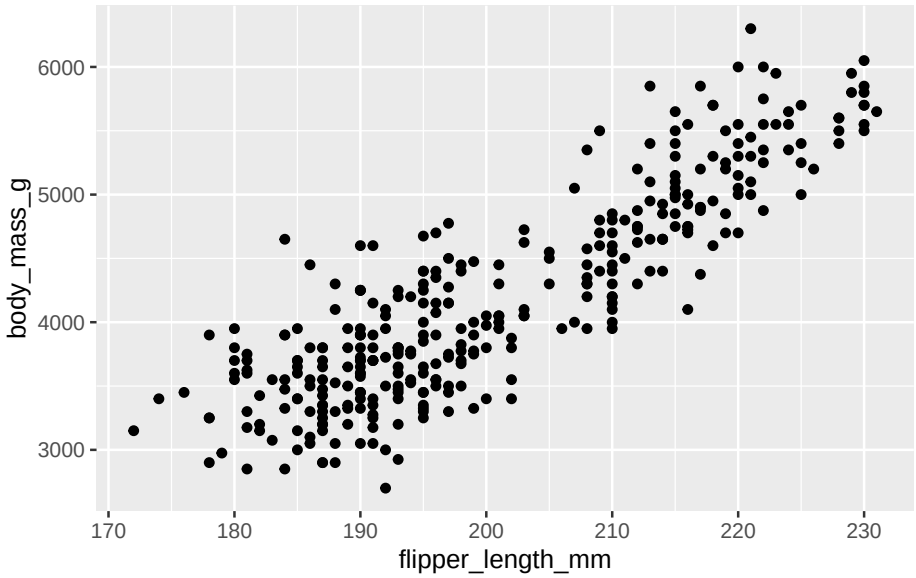
Ces deux couches (histogramme et diagramme à bandes) graphiques nous étaient utiles pour observer nos variables une à la fois. Nous allons maintenant voir quelles couches graphiques sont disponibles pour regarder l'association ou la relation entre deux variables différentes.

La relation entre deux variables quantitatives

Si nous voulons regarder l'association entre deux variables quantitatives, la façon classique de le faire est à l'aide d'un **nuage de points** (*scatterplot*). La couche graphique pour les nuages de points dans ggplot2 est le `geom_point`, dont nous avons vu plusieurs exemples au début de ce chapitre. Par exemple :

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = flipper_length_mm, y =
  ↪  body_mass_g))
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```



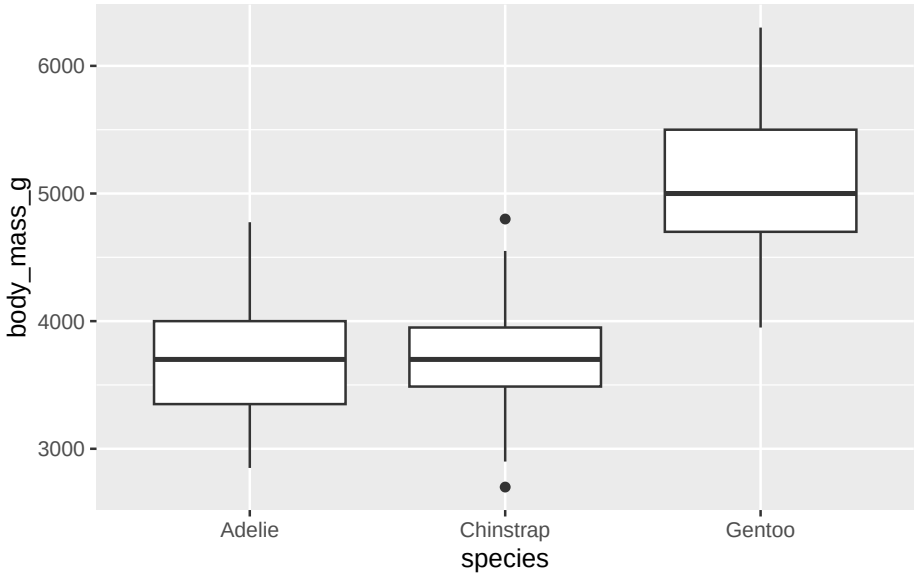
Dans ce graphique, la position en X d'un point représente sa valeur pour la variable `flipper_length_mm`, et en Y sa valeur pour la variable `body_mass_g`. Ce graphique nous permet de constater que plus les manchots ont de longues ailes (lorsque l'on se déplace vers la droite), plus ils sont lourds (on se déplace vers le haut). Lorsqu'une valeur augmente, l'autre augmente aussi.

La relation entre une variable qualitative et une quantitative

Si plutôt que d'avoir deux variables quantitatives nous voulions voir l'association entre une variable quantitative et une variable qualitative, il faudrait plutôt utiliser le **diagramme à moustaches** (*box and whisker plot*). La couche de ggplot2 pour tracer un diagramme à moustaches se nomme `geom_boxplot` :

```
ggplot(data = penguins) +
  geom_boxplot(mapping = aes(x = species, y =
  ↪  body_mass_g))
```

```
Warning: Removed 2 rows containing non-finite outside
the scale
range (`stat_boxplot()`).
```



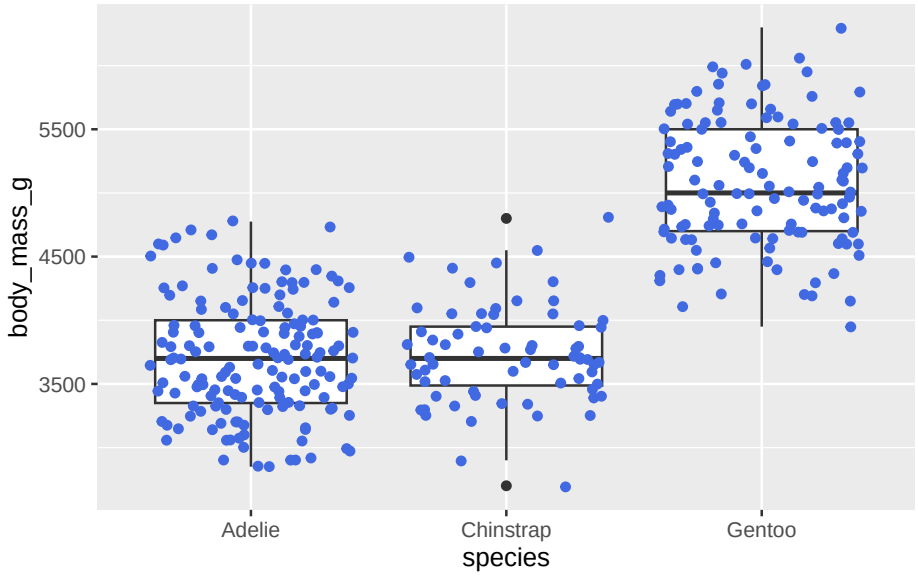
Bien qu'il soit extrêmement populaire en écologie, le diagramme à moustache est un des graphiques qui est souvent le moins bien compris et le moins standardisé. Nous prendrons donc le temps de nous y attarder un peu plus.

Remarquez d'abord qu'en X, nous avons les valeurs d'une variable qualitative (`species`) et en Y les valeurs d'une variable quantitative (`body_mass_g`). Ce que nous observons donc dans le graphique est la répartition des valeurs de poids, groupées par espèce.

De façon classique, la boîte d'un diagramme à moustache est formée par le premier et le troisième quartile. Autrement dit, la taille de la boîte nous indique entre quelles valeurs on retrouve la moitié des observations de notre variable. La ligne horizontale à l'intérieur de la boîte (souvent près du centre, mais pas nécessairement) représente la médiane des données. Elle nous indique que la moitié des valeurs sont plus grandes et la moitié plus petites que ce chiffre.

Les traits verticaux de part et d'autre de la boîte (qu'on appelle les moustaches) ne représentent pas toujours la même chose d'un logiciel à l'autre. Dans ggplot2, ils nous montrent la dernière observation qui se trouve à moins de 1,5 écart interquartile (le nom technique pour la taille de la boîte), à partir de la fin de la boîte. Cette définition n'est pas simple à comprendre, mais n'est pas si importante au bout du compte. Ce qui nous intéresse surtout, c'est de voir si au-delà des moustaches, on retrouve des points individuels. Ces points individuels (comme pour les manchots Chinstrap dans le graphique) nous indiquent des points qui n'entrent pas dans cette définition, et sont donc considérés comme des valeurs potentiellement aberrantes, qu'il vaudra la peine de vérifier et gérer correctement (nous y reviendrons).

Voici une visualisation des points à l'intérieur d'un diagramme à moustache pour vous aider à visualiser leur fonctionnement :



Le diagramme à moustache nous permet donc de constater que les man-

chots Gentoo sont plus lourds que les deux autres espèces, qui ont des poids très semblables.

Attention : vous entendrez souvent des gens discuter de leur diagramme à moustaches en parlant de moyennes, d'écart-type etc. Ces valeurs ne sont pas directement dans le graphique (sauf de rares exceptions), mais le graphique en donne quand même une bonne idée.

La relation entre deux variables qualitatives

Enfin, le dernier cas de figure à explorer est le cas où nous voulons voir si deux variables qualitatives pourraient être reliées ou associées. On pourrait par exemple se demander si le ratio mâle/femelle est différent selon les espèces de manchots.

Ce type de question est souvent exploré par un **tableau de contingence**, c'est-à-dire un tableau avec en X une variable qualitative, en Y une autre variable qualitative et dans chaque cellule le décompte pour chaque combinaison. Pour la question précédente, le tableau de contingence pourrait ressembler à ceci :

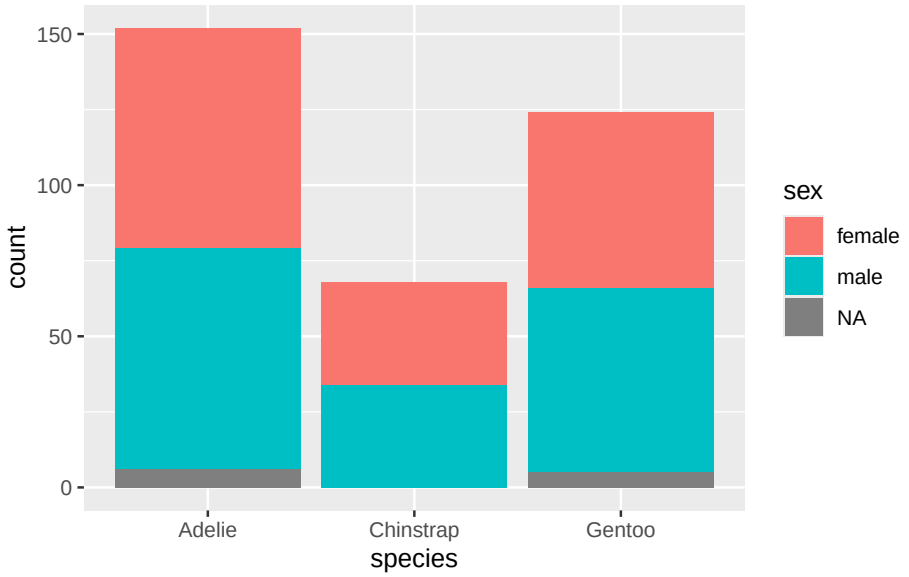
	female	male
Adelie	73	73
Chinstrap	34	34
Gentoo	58	61

En observant ce tableau, on peut constater que le ratio mâle/femelle est très proche de 50/50, et qu'il ne semble pas différer entre les espèces.

Pour les curieuses, le code pour obtenir ces chiffres sera présenté à la section Section 4.9.

On pourrait aussi observer la même information à l'aide d'un **diagramme à bandes**, auquel on intégrerait une composante de couleur de remplissage (fill) :

```
ggplot(data = penguins) +
  geom_bar(mapping = aes(x = species, fill = sex))
```



Ce graphique nous représente visuellement la même réalité : les bandes mâles et femelles sont sensiblement de la même taille pour chacune des espèces. Remarquez aussi la présence de certains individus pour lesquelles on ne connaît pas le sexe (**NA**)

3.8. Labo : Visualiser l'incertitude

Nous avons vu plus haut qu'un diagramme à moustaches permet de visualiser comment se comparent les valeurs d'une variable quantitative entre différents groupes.

Une autre façon d'approcher les mêmes données serait d'illustrer la moyenne et une mesure de variabilité, par exemple l'écart-type (nous y reviendrons à la Section 5.2). On utiliserait alors les fameuses **barres d'erreurs**.

De façon générale, les barres d'erreurs ne sont pas la façon la plus parlante d'illustrer vos données[4], mais comme elles sont encore fréquemment utilisées, il est important que vous sachiez comment les utiliser.

Donc, nous allons tout d'abord se construire un petit tableau qui contiendra la moyenne et l'écart-type du poids du corps des différentes espèces de manchots de Palmer. Nous verrons en détail à la section Section 6.4 ce qui se passe dans ce petit bout de code :

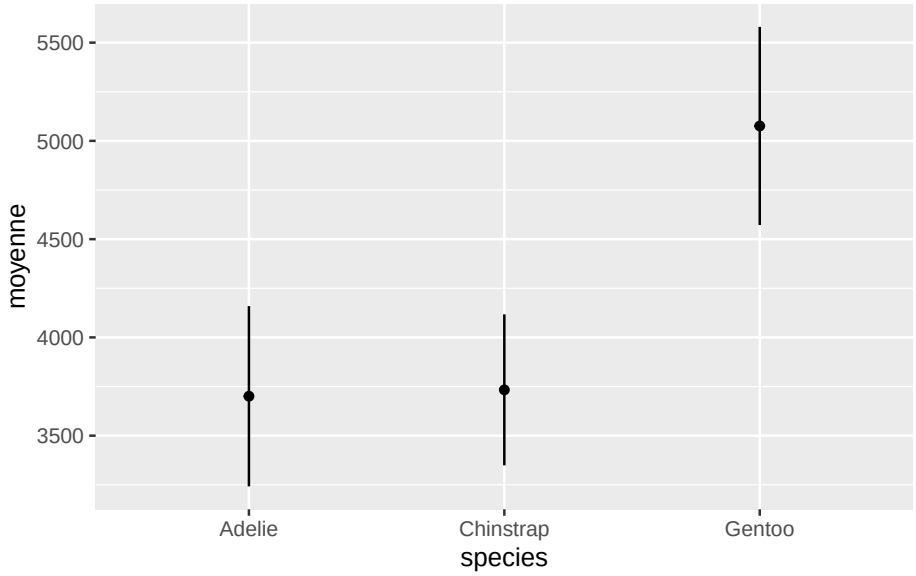
```
library(dplyr)
pour_barres <-
  penguins |>
  group_by(species) |>
  summarize (
    moyenne = mean(body_mass_g, na.rm = TRUE),
    ecart_type = sd(body_mass_g, na.rm = TRUE)
  )

pour_barres
```

```
# A tibble: 3 x 3
  species   moyenne ecart_type
  <fct>       <dbl>      <dbl>
1 Adelie      3701.       459.
2 Chinstrap   3733.       384.
3 Gentoo      5076.       504.
```

[4] https://gorelik.net/2019/10/07/error-bars-in-bar-charts-you-probably-shouldnt/

La façon la plus simple d'afficher des barres d'erreurs est de les ajouter à une couche de points avec la couche `geom_linerange` :

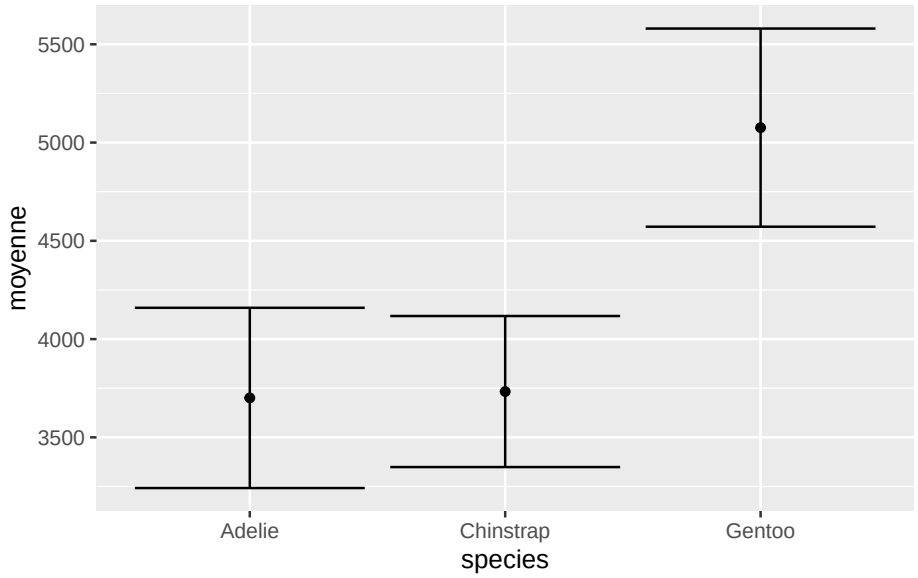
```
ggplot(data = pour_barres, mapping = aes(x = species, y
↪  = moyenne)) +
  geom_point() +
  geom_linerange(aes(
    ymin = moyenne - ecart_type,
    ymax = moyenne + ecart_type)
  )
```



Il faut spécifier manuellement jusqu'où doit aller le haut et le bas de la barre. Ici, on a choisi un écart-type au-dessus et en-dessous de la moyenne.

Vous verrez souvent aussi des moustaches plutôt que des barres simples, que vous pouvez reproduire avec la couche `geom_errorbar` :

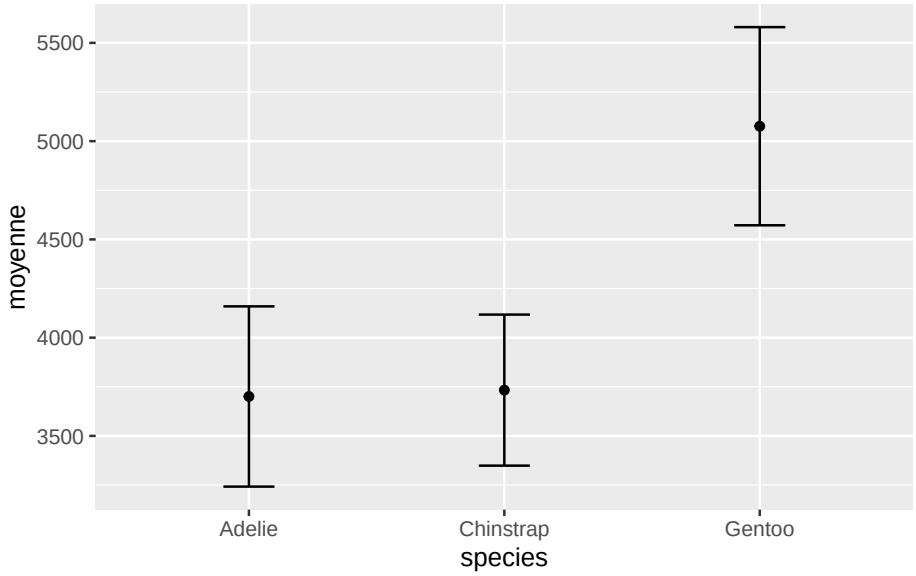
```
ggplot(data = pour_barres, mapping = aes(x = species, y
↪  = moyenne)) +
  geom_point() +
  geom_errorbar(aes(
    ymin = moyenne - ecart_type,
    ymax = moyenne + ecart_type)
  )
```



Comme la taille des moustaches est particulièrement désagréable, il peut être utile de contrôler leur largeur avec l'argument `width` :

```
ggplot(data = pour_barres, mapping = aes(x = species, y
↪  = moyenne)) +
  geom_point() +
  geom_errorbar(aes(
```

```
    ymin = moyenne - ecart_type,
    ymax = moyenne + ecart_type),
    width = 0.2
  )
```

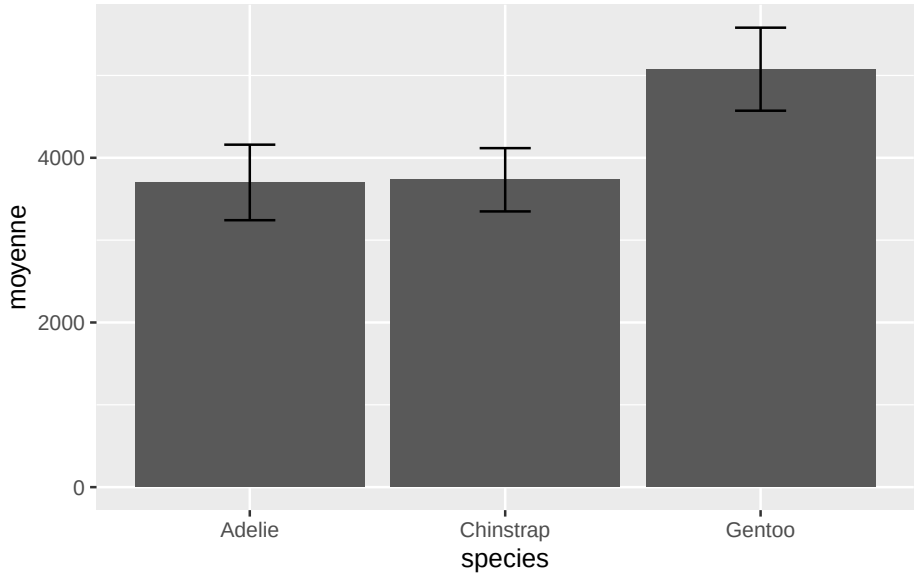


Enfin, ces barres d'erreur peuvent être ajoutées à n'importe quel type de graphique, par exemple sur un diagramme à bandes.

Bien que peu recommandés par les spécialistes en visualisation, le diagramme à bandes avec des barres d'erreurs est néanmoins très utilisé, entre autres en biologie médicale.

```
ggplot(data = pour_barres, mapping = aes(x = species, y
↪  = moyenne)) +
  geom_col() +
  geom_errorbar(aes(
```

```
    ymin = moyenne - ecart_type,
    ymax = moyenne + ecart_type),
    width = 0.2
  )
```



Remarquez que dans ce cas, il faut utiliser un `geom_col` puisque nos valeurs sont déjà pré-calculées.

! Important

Bien que tous les exemples ci-haut aient utilisé l'écart-type comme mesure d'erreur, les barres d'erreur peuvent être utilisées pour illustrer plusieurs choses différentes. Entre autres :

- L'écart-type
- L'erreur-type

- 2 x l'écart-type
- L'intervalle de confiance

Soyez très attentives à bien communiquer ce que représentent vos barres d'erreurs et assurez-vous de bien comprendre ce qu'elles représentent dans vos lectures, car elles peuvent varier d'un auteur à l'autre...

3.9. Exercices

En vous inspirant de la Section 3.6, créez un graphique avec en X la longueur du bec (`bill_length_mm`) et en Y la profondeur du bec (`bill_depth_mm`). Remplacez les points par des carrés. La couleur de chaque carré devra correspondre à l'île où a été mesuré l'individu (`island`). Comment sont reliées ces deux variables?

En adaptant le bon exemple dans la section Section 3.7, affichez un diagramme à moustaches présentant la distribution des longueurs d'aile (`flipper_length_mm`) pour chacune des espèces. Une fois ce graphique réussi, adaptez-le pour que chaque espèce possède sa propre couleur de remplissage dans le diagramme. Quelle espèces possède les ailes les plus longues?

Enfin, en prenant exemple sur la section Section 3.8, créez un graphique montrant la moyenne et l'écart-type de la longueur des ailes (`flipper_length_mm`). Comme on peut utiliser plusieurs couches différentes pour afficher cette information, reprenez ensuite ce même graphique, mais avec une couche différente. Quelle est votre façon préférée d'afficher cette information?

3.10. En résumé

Donc, si on résume comment choisir le bon graphique selon ce que l'on veut explorer, nous avons vu :

- Distribution des valeurs d'une seule variable :
  - Quantitative : Histogramme de fréquences
  - Qualitative : Diagramme à bandes
- Relation ou association entre deux variables :
  - Quantitative + Quantitative = Nuage de points
  - Quantitative + Qualitative = Diagramme à moustaches
  - Qualitative + Qualitative = Tableau de contingence OU diagramme à bande avec couleur de remplissage.

4. Manipuler les données

4.1. Des données propres et bien organisées

Contrairement à ce que l'on peut s'attendre lorsque l'on commence à faire des statistiques, la majorité de votre temps ne sera pas passée à effectuer des tests statistiques. La grande majorité de votre temps sera plutôt passée à nettoyer et organiser vos données.

À la différence de Excel où l'on peut organiser nos données plus ou moins comme bon nous semble, les logiciels spécialisés dans l'analyse de données comme R s'attendent à ce que les données soient dans un format particulier, toujours structuré de la même façon : on parlera de **données propres et bien organisées** (*tidy data*). Pour savoir si vos données sont propres et bien organisées, elles doivent répondre à trois grands principes :

- Chaque variable est représentée par une colonne
- Chaque observation est une ligne
- Chaque type d'observation est dans son propre tableau de données

Voici un exemple de données bien organisées, décrivant des sites forestiers :

site	couvert_pct	pente_degres	traitement
A	20	3	Éclaircie
B	60	2	Contrôle

site	couvert_pct	pente_degres	traitement
C	10	15	Éclaircie

4.2. Labo : Préparatifs

Il existe dans R de base un langage de manipulation de données, basé sur la façon dont les programmeurs manipulent leurs données. Selon l'avis de plusieurs (moi le premier!), cette approche n'est pas très appropriée lorsque notre but n'est pas de devenir programmeur, mais simplement de pouvoir jouer avec nos données le plus simplement possible. C'est avec l'idée de corriger cette lacune qu'a été créée la librairie de code dplyr (prononcez di-playeur, comme *data-pliers*, "pinces à données"), dont le but est de rendre les manipulations de données plus lisibles et intuitives. C'est cette bibliothèque que nous utiliserons pour manipuler les données.

Comme pour ggplot2, utilisez le code suivant pour installer la librairie dplyr :

```
install.packages("dplyr")
```

Cette opération pourrait prendre quelques minutes. Par la suite, n'oubliez pas d'activer la librairie dplyr chaque fois vous aurez besoin de l'utiliser, avec le code suivant :

```
library(dplyr)
```

```
Attaching package: 'dplyr'
```

```
The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

Notez qu'à ce point, vous obtiendrez une série d'avertissements pour vous expliquer que dplyr cache certains objets de R (*"the following objects are masked..."*). Il s'agit simplement d'un avertissement, nous indiquant que certaines fonctions de dplyr ont le même nom que des fonctions de base de R, et que ces dernières ont été cachées par dplyr.

Pour nos exemples de ce chapitre, nous utiliserons à nouveau le tableau de données `penguins`, n'oubliez donc pas sa librairie avant de commencer.

```
library(palmerpenguins)
```

Vous êtes maintenant prêtes à attaquer vos données!

4.3. Cinq opérations de base

Bien que la manipulation de données puisse devenir extrêmement complexe et pointue, elle peut en général se résumer à cinq opérations de base :

- Filtrer des observations
- Trier les observations (i.e. changer l'ordre)
- Ajouter des variables
- Sélectionner des variables

- Résumer les données.

La grande majorité de ce que vous pourriez faire avec vos données peut se résumer à la combinaison de ces opérations.

4.4. Labo : Filtrer des observations

La fonction de dplyr permettant de filtrer les observations se nomme… `filter`! Elle reçoit deux arguments, le premier étant le tableau de données sur lequel appliquer le filtre, et le deuxième décrivant la condition que devront respecter les observations pour être conservées. P. ex. pour conserver uniquement les manchots dont le poids du corps est de plus de 4 kg, nous écrierions le code suivant :

```
filter(penguins, body_mass_g > 4000)
```

Suite à cette commande, R vous répond avec un aperçu du tableau de données modifié :

```
# A tibble: 172 x 8
   species island    bill_length_mm bill_depth_mm
   <fct>   <fct>              <dbl>         <dbl>
 1 Adelie  Torgersen           39.2          19.6
 2 Adelie  Torgersen           42            20.2
 3 Adelie  Torgersen           34.6          21.1
 4 Adelie  Torgersen           42.5          20.7
 5 Adelie  Torgersen           46            21.5
 6 Adelie  Dream              39.2          21.1
 7 Adelie  Dream              39.8          19.1
 8 Adelie  Dream              44.1          19.7
 9 Adelie  Dream              39.6          18.8
10 Adelie  Dream              42.3          21.2
```

```
# i 162 more rows
# i 4 more variables: flipper_length_mm <int>,
#   body_mass_g <int>, sex <fct>, year <int>
```

Vous constatez que ce tableau de données ne contient que 172 lignes plutôt que 344 initialement.

> ⚠ Avertissement
>
> Mais attention, le tableau de données penguins original n'a pas été modifié par votre commande.

Il s'agit d'un principe extrêmement important à comprendre lorsque l'on travaille avec R. À moins qu'on le demande clairement, R ne touche pas à notre tableau de données original. Il nous affiche notre résultat, puis il l'oublie (c'est un peu bête, mais c'est comme ça!).

Pour conserver le résultat d'une opération dans R, il faut assigner ce résultat à un objet avec l'**opérateur d'assignation** (<-). P. ex. on pourrait se créer un tableau de données de petits manchots avec la commande suivante :

```
petits_manchots <- filter(penguins, body_mass_g < 4000)
```

On peut lire une telle commande comme : filtrer le tableau de données `penguins`, puis prendre le résultat et le pousser (<-) dans l'objet `petits_manchots`. Nous avons à ce moment deux objets dans notre environnement de travail : `penguins`, qui contient tous les manchots, et `petits_manchots`, qui en contient un sous-ensemble. Comme expliqué au Chapitre 3, on peut utiliser la commande View pour voir ces deux tableaux :

```
View(penguins)
View(petits_manchots)
```

4.5. Labo : Particularités des filtres

R, comme tous les autres langages de programmation, a certaines particularités qu'il est important de connaître. Nous en discuterons ici de quelques unes qui vous permettront d'éviter bien des ennuis au moment de filtrer les données.

> **Astuce**
>
> La première chose à savoir, est que le symbole = ne veut pas dire "égal"

(Encore là, c'est bête, mais c'est comme ça!). Il est utilisé dans R comme synonyme de <-, l'opérateur d'assignation. Pour tester l'égalité entre deux valeurs, il faut plutôt utiliser l'opérateur ==, comme ceci :

```
filter(penguins, species == "Adelie")
```

```
# A tibble: 152 x 8
   species island    bill_length_mm bill_depth_mm
   <fct>   <fct>              <dbl>         <dbl>
 1 Adelie  Torgersen           39.1          18.7
 2 Adelie  Torgersen           39.5          17.4
 3 Adelie  Torgersen           40.3          18
 4 Adelie  Torgersen           NA            NA
 5 Adelie  Torgersen           36.7          19.3
 6 Adelie  Torgersen           39.3          20.6
 7 Adelie  Torgersen           38.9          17.8
```

```
 8 Adelie  Torgersen           39.2          19.6
 9 Adelie  Torgersen           34.1          18.1
10 Adelie  Torgersen           42            20.2
# i 142 more rows
# i 4 more variables: flipper_length_mm <int>,
#   body_mass_g <int>, sex <fct>, year <int>
```

Vous obtenez ainsi la liste de tous les manchots Adélie de votre tableau de données.

> **Astuce**
>
> Deuxième particularité à connaître : l'opérateur `==` ne fonctionne PAS pour trouver les valeurs manquantes.

Dans R, les valeurs manquantes sont désignées par le code `NA` (sans guillemets). Il pourrait être tentant de faire :

```
filter(penguins, sex == NA)
```

pour trouver les manchots dont on ne connaît pas le sexe, mais cette opération n'aura pas le résultat attendu.

La logique de R est que, si je ne connais pas l'âge de Jacques, et que je ne connais pas l'âge de Jean, si on demande est-ce que Jean et Jacques ont le même âge, la réponse n'est pas oui, mais plutôt "je ne sais pas". Il existe donc dans R une fonction spéciale qui permet de vérifier si une valeur est manquante, qui se nomme is.na :

```
filter(penguins, is.na(sex))
```

```
# A tibble: 11 x 8
   species island    bill_length_mm bill_depth_mm
   <fct>   <fct>              <dbl>         <dbl>
```

```
 1 Adelie  Torgersen           NA            NA
 2 Adelie  Torgersen           34.1          18.1
 3 Adelie  Torgersen           42            20.2
 4 Adelie  Torgersen           37.8          17.1
 5 Adelie  Torgersen           37.8          17.3
 6 Adelie  Dream               37.5          18.9
 7 Gentoo  Biscoe              44.5          14.3
 8 Gentoo  Biscoe              46.2          14.4
 9 Gentoo  Biscoe              47.3          13.8
10 Gentoo  Biscoe              44.5          15.7
11 Gentoo  Biscoe              NA            NA
# i 4 more variables: flipper_length_mm <int>,
#   body_mass_g <int>, sex <fct>, year <int>
```

> **Astuce**
>
> Une autre particularité intéressante est que l'on peut inverser une condition, à l'aide du point d'exclamation.

On peut, par exemple, chercher la liste de tous les manchots pour lesquels on connaît le sexe avec la commande :

```
filter(penguins, !is.na(sex))
```

```
# A tibble: 333 x 8
   species island    bill_length_mm bill_depth_mm
   <fct>   <fct>              <dbl>         <dbl>
 1 Adelie  Torgersen           39.1          18.7
 2 Adelie  Torgersen           39.5          17.4
 3 Adelie  Torgersen           40.3          18
 4 Adelie  Torgersen           36.7          19.3
 5 Adelie  Torgersen           39.3          20.6
 6 Adelie  Torgersen           38.9          17.8
```

```
 7 Adelie  Torgersen           39.2          19.6
 8 Adelie  Torgersen           41.1          17.6
 9 Adelie  Torgersen           38.6          21.2
10 Adelie  Torgersen           34.6          21.1
# i 323 more rows
# i 4 more variables: flipper_length_mm <int>,
#   body_mass_g <int>, sex <fct>, year <int>
```

Une fois ces subtilités comprises, vous serez probablement contente d'apprendre qu'il existe une façon plus lisible d'éliminer les lignes contenant des données manquantes. Vous aurez par contre besoin d'installer et activer une librairie de code supplémentaire nommée tidyr (prononcer tayediyeur, comme dans "mieux organisé"), qui a été conçue spécifiquement pour nettoyer les données.

```
install.packages("tidyr")
```

```
library(tidyr)
```

Une fois la librairie activée, vous pouvez l'utiliser pour créer un tableau sans les lignes où des valeurs étaient manquantes dans la colonne, comme ceci :

```
drop_na(penguins, sex)
```

```
# A tibble: 333 x 8
   species island    bill_length_mm bill_depth_mm
   <fct>   <fct>              <dbl>         <dbl>
 1 Adelie  Torgersen           39.1          18.7
 2 Adelie  Torgersen           39.5          17.4
 3 Adelie  Torgersen           40.3          18
 4 Adelie  Torgersen           36.7          19.3
```

```
 5 Adelie  Torgersen           39.3          20.6
 6 Adelie  Torgersen           38.9          17.8
 7 Adelie  Torgersen           39.2          19.6
 8 Adelie  Torgersen           41.1          17.6
 9 Adelie  Torgersen           38.6          21.2
10 Adelie  Torgersen           34.6          21.1
# i 323 more rows
# i 4 more variables: flipper_length_mm <int>,
#   body_mass_g <int>, sex <fct>, year <int>
```

Vous pouvez aussi, avec la même fonction, éliminer toutes les lignes qui contiennent des données manquantes dans l'une ou l'autre des colonnes, pour conserver uniquement les lignes dont l'information est complète. Dans ce cas, vous n'avez qu'à ne pas nommer de colonne comme deuxième argument :

```
drop_na(penguins)
```

```
# A tibble: 333 x 8
   species island    bill_length_mm bill_depth_mm
   <fct>   <fct>              <dbl>         <dbl>
 1 Adelie  Torgersen           39.1          18.7
 2 Adelie  Torgersen           39.5          17.4
 3 Adelie  Torgersen           40.3          18
 4 Adelie  Torgersen           36.7          19.3
 5 Adelie  Torgersen           39.3          20.6
 6 Adelie  Torgersen           38.9          17.8
 7 Adelie  Torgersen           39.2          19.6
 8 Adelie  Torgersen           41.1          17.6
 9 Adelie  Torgersen           38.6          21.2
10 Adelie  Torgersen           34.6          21.1
# i 323 more rows
# i 4 more variables: flipper_length_mm <int>,
#   body_mass_g <int>, sex <fct>, year <int>
```

Assurez-vous avant d'effectuer ce genre d'opération que seules les colonnes qui vous intéressent sont présentes dans le tableau de données. Pensez par exemple que si vous avez une colonne de notes qui ne contient rien sur la majorité des lignes, appeler `drop_na` de cette façon éliminera toutes les lignes qui n'ont pas de notes...

> **Astuce**
>
> La dernière particularité concernant les filtres est que l'on peut combiner plusieurs conditions avec des OU ou des ET.

Pour se faire, on énumère chacune des conditions, en les séparant par le bon opérateur, respectivement `|` pour les OU et `&` pour les ET. Par exemple, pour avoir tous les manchots Adélie et Gentoo, on ferait comme ceci :

```
filter(penguins, species == "Adelie" | species ==
↪  "Gentoo")
```

```
# A tibble: 276 x 8
   species island    bill_length_mm bill_depth_mm
   <fct>   <fct>              <dbl>         <dbl>
 1 Adelie  Torgersen           39.1          18.7
 2 Adelie  Torgersen           39.5          17.4
 3 Adelie  Torgersen           40.3          18
 4 Adelie  Torgersen           NA            NA
 5 Adelie  Torgersen           36.7          19.3
 6 Adelie  Torgersen           39.3          20.6
 7 Adelie  Torgersen           38.9          17.8
 8 Adelie  Torgersen           39.2          19.6
 9 Adelie  Torgersen           34.1          18.1
10 Adelie  Torgersen           42            20.2
# i 266 more rows
```

```
# i 4 more variables: flipper_length_mm <int>,
#   body_mass_g <int>, sex <fct>, year <int>
```

Remarquez que, verbalement, on aurait tendance à dire “je veux les Adélie et les Gentoo”, mais qu’en termes d’algèbre booléen, on doit dire : “je veux les lignes où `species` a la valeur Adelie ou la valeur Gentoo”.

Enfin, rappelez-vous que cette fonction, comme toutes les autres, ne modifie pas le tableau de données original. Vous devez assigner le contenu à un nouvel objet si vous voulez conserver le résultat de l’opération.

4.6. Labo : Trier les observations

Dans R, lorsque nous demandons de voir le contenu d’un tableau de données, les observations seront toujours présentées dans le même ordre, celui dans lequel les observations ont été saisies dans le tableau. Pour modifier cet ordre, il faut utiliser la fonction arrange, comme ceci :

```
arrange(penguins, body_mass_g)
```

```
# A tibble: 344 x 8
   species   island    bill_length_mm bill_depth_mm
   <fct>     <fct>              <dbl>         <dbl>
 1 Chinstrap Dream               46.9          16.6
 2 Adelie    Biscoe              36.5          16.6
 3 Adelie    Biscoe              36.4          17.1
 4 Adelie    Biscoe              34.5          18.1
 5 Adelie    Dream               33.1          16.1
 6 Adelie    Torgersen           38.6          17
 7 Chinstrap Dream               43.2          16.6
 8 Adelie    Biscoe              37.9          18.6
 9 Adelie    Dream               37.5          18.9
```

```
10 Adelie    Dream               37            16.9
# i 334 more rows
# i 4 more variables: flipper_length_mm <int>,
#   body_mass_g <int>, sex <fct>, year <int>
```

Cette fonction reçoit deux arguments, soit le tableau de données sur lequel s'appliquer, et le nom de la colonne sur laquelle trier. Si on ne dit rien, R nous trie notre tableau en ordre croissant. Rappelez-vous cependant que l'ordre dans votre tableau original ne sera pas affecté par cette opération. Vous devrez assigner le résultat à un nouvel objet pour conserver cet ordre et l'utiliser ailleurs :

```
manchots_tries <- arrange(penguins, body_mass_g)
```

Si au contraire vous voulez obtenir vos données en ordre décroissant, il faut apporter une petite modification au code, comme ceci :

```
arrange(penguins, desc(body_mass_g))
```

```
# A tibble: 344 x 8
   species island bill_length_mm bill_depth_mm
   <fct>   <fct>           <dbl>         <dbl>
 1 Gentoo  Biscoe           49.2          15.2
 2 Gentoo  Biscoe           59.6          17
 3 Gentoo  Biscoe           51.1          16.3
 4 Gentoo  Biscoe           48.8          16.2
 5 Gentoo  Biscoe           45.2          16.4
 6 Gentoo  Biscoe           49.8          15.9
 7 Gentoo  Biscoe           48.4          14.6
 8 Gentoo  Biscoe           49.3          15.7
 9 Gentoo  Biscoe           55.1          16
10 Gentoo  Biscoe           49.5          16.2
# i 334 more rows
```

```
# i 4 more variables: flipper_length_mm <int>,
#   body_mass_g <int>, sex <fct>, year <int>
```

4.7. Labo : Sélectionner certaines colonnes

Parfois il arrivera que votre tableau de données contiendra beaucoup d'informations, qui ne sont pas nécessairement intéressantes pour la question que vous voulez traiter. Il convient alors de simplifier le tableau de données en sélectionnant uniquement certaines variables, comme ceci :

```
select(penguins, body_mass_g, flipper_length_mm, sex)
```

```
# A tibble: 344 x 3
   body_mass_g flipper_length_mm sex
         <int>             <int> <fct>
 1        3750               181 male
 2        3800               186 female
 3        3250               195 female
 4          NA                NA <NA>
 5        3450               193 female
 6        3650               190 male
 7        3625               181 female
 8        4675               195 male
 9        3475               193 <NA>
10        4250               190 <NA>
# i 334 more rows
```

La commande `select` reçoit une série d'arguments. Le premier est toujours le tableau de données, puis ensuite, les autres sont le nom de

toutes les colonnes que l'on désire sélectionner. Dans l'exemple précédent, nous avons créé un tableau de données contenant trois colonnes : `body_mass_g`, `flipper_length_mm` et `sex`.

On peut aussi sélectionner tout, sauf certaines variables, en utilisant un `-` devant chaque nom, comme ceci :

```
select(penguins, -body_mass_g, -bill_length_mm, -sex)
```

```
# A tibble: 344 x 5
   species island bill_depth_mm flipper_length_mm  year
   <fct>   <fct>          <dbl>             <int> <int>
 1 Adelie  Torge~          18.7               181  2007
 2 Adelie  Torge~          17.4               186  2007
 3 Adelie  Torge~          18                 195  2007
 4 Adelie  Torge~          NA                  NA  2007
 5 Adelie  Torge~          19.3               193  2007
 6 Adelie  Torge~          20.6               190  2007
 7 Adelie  Torge~          17.8               181  2007
 8 Adelie  Torge~          19.6               195  2007
 9 Adelie  Torge~          18.1               193  2007
10 Adelie  Torge~          20.2               190  2007
# i 334 more rows
```

4.8. Labo : Ajouter des variables

Il est possible d'ajouter des variables à votre tableau de données, sur le même principe que les formules dans Excel.

La fonction qui permet d'ajouter une variable à un tableau de données se nomme mutate. Il s'agit, tristement, du nom de fonction le moins intuitif de la librairie dplyr. Essayez d'imaginer que votre tableau subit une mutation, qu'il lui pousse un nouveau morceau... une nouvelle variable.

Pour ajouter une colonne de poids des manchots en kg plutôt qu'en g, nous pouvons utiliser le code suivant :

```
mutate(penguins, body_mass_kg = body_mass_g / 1000)
```

```
# A tibble: 344 x 9
   species island    bill_length_mm bill_depth_mm
   <fct>   <fct>              <dbl>         <dbl>
 1 Adelie  Torgersen           39.1          18.7
 2 Adelie  Torgersen           39.5          17.4
 3 Adelie  Torgersen           40.3          18
 4 Adelie  Torgersen           NA            NA
 5 Adelie  Torgersen           36.7          19.3
 6 Adelie  Torgersen           39.3          20.6
 7 Adelie  Torgersen           38.9          17.8
 8 Adelie  Torgersen           39.2          19.6
 9 Adelie  Torgersen           34.1          18.1
10 Adelie  Torgersen           42            20.2
# i 334 more rows
# i 5 more variables: flipper_length_mm <int>,
#   body_mass_g <int>, sex <fct>, year <int>,
#   body_mass_kg <dbl>
```

La fonction mutate doit recevoir comme premier argument le nom du tableau de données sur lequel travailler, puis le nom de la nouvelle colonne, suivi d'un = et du détail du calcul pour construire la variable.

Malheureusement, les nouvelles colonnes sont ajoutés à la fin (à droite) du tableau et donc, pas visibles ici.

On peut cependant apercevoir notre nouvelle colonne avec la fonction `View`, ou sinon, on peut utiliser un argument supplémentaire de la fonction mutate, pour lui dire d'insérer notre nouvelle colonne à gauche plutôt qu'à droite :

```
mutate(
  penguins,
  body_mass_kg = body_mass_g / 1000,
  .before = 1
)
```

```
# A tibble: 344 x 9
   body_mass_kg species island    bill_length_mm
          <dbl> <fct>   <fct>              <dbl>
 1         3.75 Adelie  Torgersen           39.1
 2         3.8  Adelie  Torgersen           39.5
 3         3.25 Adelie  Torgersen           40.3
 4        NA    Adelie  Torgersen           NA
 5         3.45 Adelie  Torgersen           36.7
 6         3.65 Adelie  Torgersen           39.3
 7         3.62 Adelie  Torgersen           38.9
 8         4.68 Adelie  Torgersen           39.2
 9         3.48 Adelie  Torgersen           34.1
10         4.25 Adelie  Torgersen           42
# i 334 more rows
# i 5 more variables: bill_depth_mm <dbl>,
#   flipper_length_mm <int>, body_mass_g <int>,
#   sex <fct>, year <int>
```

Notez qu'il est possible d'utiliser plus d'une variable à l'intérieur du calcul. On pourrait p. ex. ajouter une colonne contenant le rapport entre la longueur et l'épaisseur du bec, comme ceci :

de poids relatif du cerveau, comme ceci :

```
mutate(
  penguins,
  body_mass_kg = body_mass_g / 1000,
```

```
  ratio_bec = bill_length_mm / bill_depth_mm,
  .before = 1
)
```

```
# A tibble: 344 x 10
   body_mass_kg ratio_bec species island bill_length_mm
          <dbl>     <dbl> <fct>   <fct>           <dbl>
 1         3.75      2.09 Adelie  Torge~           39.1
 2         3.8       2.27 Adelie  Torge~           39.5
 3         3.25      2.24 Adelie  Torge~           40.3
 4        NA        NA    Adelie  Torge~           NA
 5         3.45      1.90 Adelie  Torge~           36.7
 6         3.65      1.91 Adelie  Torge~           39.3
 7         3.62      2.19 Adelie  Torge~           38.9
 8         4.68      2    Adelie  Torge~           39.2
 9         3.48      1.88 Adelie  Torge~           34.1
10         4.25      2.08 Adelie  Torge~           42
# i 334 more rows
# i 5 more variables: bill_depth_mm <dbl>,
#   flipper_length_mm <int>, body_mass_g <int>,
#   sex <fct>, year <int>
```

Plus ce ratio est élevé, plus les manchots ont des becs proportionnellement étroits.

Rappelez-vous, comme pour les opérations précédentes, que le tableau original n'est pas modifié par la commande. Il faut utiliser l'opérateur d'assignation (<-) pour conserver le résultat dans un nouvel objet.

4.9. Labo : Résumer les données

Résumer les données est une opération que l'on fait tout naturellement, souvent sans s'en rendre compte : compter le nombre d'observations, calculer la moyenne d'une variable, etc. Dans la librairie dplyr, la fonction qui permet d'effectuer ces opérations se nomme `summarize`. Si l'on voulait connaître le poids moyen des manchots dans notre tableau de données, nous pourrions lancer ceci :

```
summarize (penguins, poids_moyen = mean(body_mass_g))
```

```
# A tibble: 1 x 1
  poids_moyen
        <dbl>
1          NA
```

Ici, R nous répond `NA` puisque la colonne `body_mass_g` contient des données manquantes. Une façon rapide de contourner le problème est de mentionner à la fonction `mean` d'ignorer les données manquantes dans son calcul.

```
summarize (penguins, poids_moyen = mean(body_mass_g,
 ↪  na.rm = TRUE))
```

```
# A tibble: 1 x 1
  poids_moyen
        <dbl>
1       4202.
```

Remarquez que dans un vrai projet, les données manquantes doivent être gérées en amont des analyses. `na.rm` est plus un diachylon (*plaster*) qu'une vraie solution.

La fonction `summarize` attend deux arguments, soit le tableau de données sur lequel appliquer la fonction, puis l'opération à effectuer. Ici, on calcule la moyenne (`mean`) de la variable `body_mass_g`, et on stocke le résultat dans une colonne nommée `poids_moyen`.

Il existe plusieurs fonctions pour résumer les données dans R, donc voici un aperçu non-exhaustif :

- `mean` : calculer la moyenne
- `max` : trouver la valeur la plus élevée
- `min` : trouver la valeur la plus faible
- `sd` : l'écart-type (nous verrons à la Section 5.2 l'utilité de l'écart-type et le détail de son calcul)
- `n` : trouver le nombre de valeurs

Nous en verrons aussi quelques autres dans le Chapitre 5.

Enfin, pour calculer à la fois plusieurs choses, sur plusieurs variables différentes, on peut ajouter des arguments à `summarize` :

```
summarize (
  penguins,
  poids_moyen = mean(body_mass_g, na.rm = TRUE),
  plus_grande_aile = max(flipper_length_mm, na.rm =
   ↪  TRUE),
  nb_observations = n()
)
```

```
# A tibble: 1 x 3
  poids_moyen plus_grande_aile nb_observations
        <dbl>            <int>           <int>
1       4202.              231             344
```

Il faut à ce moment séparer chacun des calculs par une virgule.

Notez que la sortie de la fonction `summarize` est aussi un tableau de données, qui peut être assigné à un objet et manipulé comme tout le reste.

4.10. Exercices

Tout d'abord, partir du tableau de données `penguins`, trouvez en ordre croissant de poids (`body_mass_g`), la liste de tous les manchots Gentoo mâles.

Préparez ensuite un petit tableau de données nommé `petits_pas_adelie` contenant tous les manchots Gentoo et Chinstrap, dont le poids est inférieur à 3000g. Combien de manchots correspondent à cette description?

Produisez ensuite un graphique, au meilleur de votre connaissance, permettant de valider si la réponse précédente a du sens.

Enfin, calculez le poids moyen des manchots Gentoo. Effectuez la même opération pour les manchots Adélie. En moyenne, quelle espèce est la plus lourde?

Notez que vous pouvez, pour cet exercice, créer des petits tableaux de données intermédiaires, par exemple contenant uniquement les Gentoo, pour répondre à la question en plusieurs étapes. Nous verrons à la Section 6.3 comment nous aurions pu répondre à cette question en une seule commande ultra-compacte.

4.11. En résumé

Nous avons donc défini dans ce chapitre cinq opérations de base qui nous permettront de manipuler des tableaux de données, soit :

- `filter` : conserver les observations qui correspondent à certaines conditions
- `arrange` : modifier l'ordre des observations dans un tableau de données (i.e. trier)
- `select` : simplifier un tableau de données en conservant uniquement certaines variables
- `mutate` : ajouter une variable au tableau de données, basée sur un calcul
- `summarize` : résumer les données d'un tableau

Toutes ces fonctions reçoivent comme premier argument le tableau de données sur lequel elles doivent s'appliquer. Et dans tous les cas, le tableau de données original n'est pas modifié, il faut utiliser l'opérateur d'assignation (`<-`) pour conserver le résultat.

5. Décrire les données

Le Chapitre 3 nous a permis de mettre en place différentes façons de voir les données. Lorsque nous désirons comprendre nos données, il est primordial de commencer par cette étape de visualisation. Il est néanmoins utile (et parfois même nécessaire) de pouvoir mettre des chiffres sur ce que l'on voit. Le présent chapitre sera donc dédié à la description des données par des chiffres.

5.1. Mesures de tendance centrale

La tendance centrale est un terme technique pour dire de façon plus simple : autour de quelle valeur tournent nos chiffres. Cela peut sembler anodin, mais il existe en fait plusieurs façons d'y arriver.

Pour une variable quantitative, la façon la plus simple pour décrire cette tendance centrale est la **médiane**. La médiane consiste à déterminer quel est le point milieu de nos données. Nous avons, par définition, 50 % de nos observations qui sont plus petites que la médiane et 50 % qui sont plus grandes. Pour calculer la médiane, il faut trier nos observations par ordre de grandeur. Ensuite, si notre nombre de données est impair, la valeur de la médiane est la valeur en plein centre. Si notre nombre d'observations est pair, la médiane est le point milieu entre les deux nombres les plus au centre.

Si nous avions mesuré les nombres [5, 4, 3, 10, 2], la médiane serait de de 4, et si on avait mesuré [5,4,3,10], elle serait de 4,5.

L'autre façon intuitive de décrire la tendance centrale des données est de calculer la moyenne. Ce que nous appelons couramment "moyenne" est en fait la moyenne arithmétique. Sachez qu'il existe aussi la moyenne géométrique (où l'on multiplie les nombres plutôt que les additionner), mais nous n'entrerons pas dans ces détails ici. Pour calculer la **moyenne**, il faut faire la somme de toutes nos données, puis diviser par le nombre de données.

La moyenne possède plusieurs propriétés intéressantes, entre autres le fait que multiplier la moyenne par le nombre de données, nous redonne toujours la somme des données originales (ce qui n'est pas le cas pour la médiane).

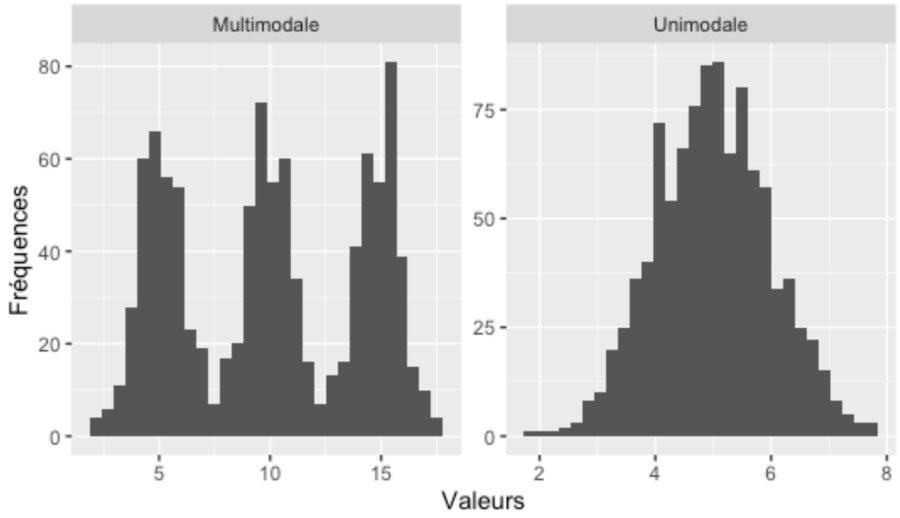
Par contre, la moyenne est plus sensible aux données extrêmes que la médiane. Si vous prenez la suite de nombre suivants : [1, 2 , 3 , 4 , 5 , 6 , 7 , 9 , 1000], la moyenne de ces données sera de 115,2, alors que la médiane sera de 5.

En général, si il n'y a pas de données extrêmes pour tirer la moyenne d'un côté ou l'autre, la moyenne est un meilleur indicateur de tendance centrale, mais en cas d'asymétrie (un côté clairement différent de l'autre), la médiane devient plus intéressante.

Pour les variables qualitatives, vous conviendrez avec moi qu'il est difficile de faire des additions et des divisions! C'est pourquoi, pour ce type de variables, la tendance centrale est habituellement définie par le mode. Le **mode** se trouve en déterminant la valeur la plus commune d'une variable. Par exemple, si une variable contient les valeurs ["A", "B", "A", "B", "B"], le mode de cette variable sera "B".

Le mode peut être aussi utile pour décrire ce que l'on voit dans un histogramme de fréquences. Puisque les données sont groupées en catégories, le mode de l'histogramme sera la bande dans laquelle il y aura le plus d'observations.

On peut aussi étendre la définition du mode, pour nous informer sur la forme d'une distribution dans un histogramme, à savoir si ce dernier est **multimodal** (plusieurs modes) ou **unimodal** (un seul mode).



Notez qu'il faut utiliser notre jugement pour déterminer quels sont les modes et quelles sont des variations normales de forme de l'histogramme puisque la nature est variable.

5.2. Mesures de variabilité (syn. dispersion)

Une fois que nous avons pu décrire la tendance centrale de nos données, l'autre aspect dont nous discuterons abondamment est la variabilité de ces données, c'est-à-dire combien les observations sont différentes les unes des autres. On parle aussi souvent de dispersion des données (autour de la moyenne).

La façon la plus simple de définir la variabilité des données est d'en mesurer l'étendue. On calcule l'**étendue** en trouvant la valeur la plus élevée

et en lui soustrayant la valeur la plus faible. L'étendue est très simple à calculer et à interpréter : c'est la différence entre la plus grande et la plus petite valeur.

Par contre, elle possède aussi un sérieux handicap. De façon probabiliste, plus nous avons d'observations dans notre tableau de données, plus l'étendue sera grande, ce qui est un grave problème, qui rend complexe l'utilisation de cette mesure pour comparer divers jeux de données.

Pour contourner ce problème, les scientifiques calculent en général la variance de leurs données pour décrire leur variabilité. La **variance** peut être décrite comme la moyenne des distances à la moyenne, au carré. C'est un concept un peu abstrait auquel il vaut la peine de s'attarder, parce qu'il reviendra fréquemment. Voici d'ailleurs notre première formule mathématique :

$$\sigma^2 = \sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n-1}$$

Puisqu'il s'agit de notre première équation ensemble, prenons le temps de la décortiquer. σ^2 se prononce sigma au carré, il s'agit du symbole de la variance d'un échantillon (nous reviendrons plus tard sur le concept d'échantillon). Σ (aussi sigma, mais cette fois-ci majuscule) représente une somme. Le $i = 1$ en dessous nous informe que nous devons démarrer notre somme à la première observation de nos données et le petit n nous informe d'arrêter à la dernière (n étant le nombre d'observations). Autrement dit, répéter la partie de droite pour chacune des valeurs et faire la somme de tout ça. Dans la partie de droite, x_i signifie : pour la valeur de x à laquelle nous sommes rendus (on dit parfois la ième valeur). Enfin, $\bar{x}$ se prononce généralement "x-barre" et représente la moyenne de la variable x.

Donc, si l'on résume la formule : on calcule d'abord la moyenne de x. Ensuite, pour chacune des valeurs de x, on leur soustrait la moyenne. On

met ces différences au carré, et on les divise par n-1. Ensuite, quand on a toutes ces valeurs, on en fait la somme. On obtient ainsi la variance.

Si nous avons par exemple les nombres [2, 4, 6]. La première étape est de calculer la moyenne, qui sera de 4. Ensuite, pour chaque nombre on lui soustrait la moyenne [2, 4, 6] - 4 = [-2, 0, 2]. Ensuite, on met chacun des nombres au carré, ce qui devient [4, 0, 4]. Ensuite, on divise chaque nombre par n-1 (donc 2), ce qui devient [2, 0, 2]. Enfin on les additionne pour obtenir une variance de 4.

Plus les valeurs sont différentes de la moyenne, plus la variance sera élevée. P. ex. ces deux séries de nombres ont la même moyenne : [-1, 0, 1] et [-10, 0, 10] soit zéro, mais leurs variances sont respectivement 1 et 100.

Comme discuté précédemment, l'avantage de la variance est qu'elle n'est pas affectée systématiquement par la taille de l'échantillon. Cependant, elle présente aussi un problème majeur au niveau de l'interprétation, soit que ses valeurs sont au carré. Si p. ex. nous mesurons le poids de 3 oiseaux en grammes et obtenons [20, 22, 26], la variance du poids de nos oiseaux sera de 9,33 gramme^2 (e.g. grammes-carrés). Difficile de se représenter mentalement un "gramme-carré" n'est-ce pas? C'est gros ou pas 9 grammes-carrés?

C'est pourquoi, pour discuter de la variabilité des données, on utilise généralement l'**écart-type** (σ), qui est défini comme la racine-carrée de la variance. L'écart-type possède la même robustesse au nombre d'observations que la variance, mais il est lui à la même échelle que les données, donc beaucoup plus facile à interpréter. Dans notre exemple précédent, l'écart-type du poids de nos oiseaux serait de 3,06 grammes.

5.3. Asymétrie

L'asymétrie d'une distribution (si elle est étirée à droite ou à gauche) peut aussi se quantifier, à l'aide d'un chiffre nommé le **coefficient d'asymétrie** (*skew*). Nous n'entrerons pas dans les détails de ce calcul. La chose importante à savoir pour le moment est qu'un coefficient positif est associé à une longue queue à droite, et qu'un coefficient négatif est associé à une longue queue à gauche.

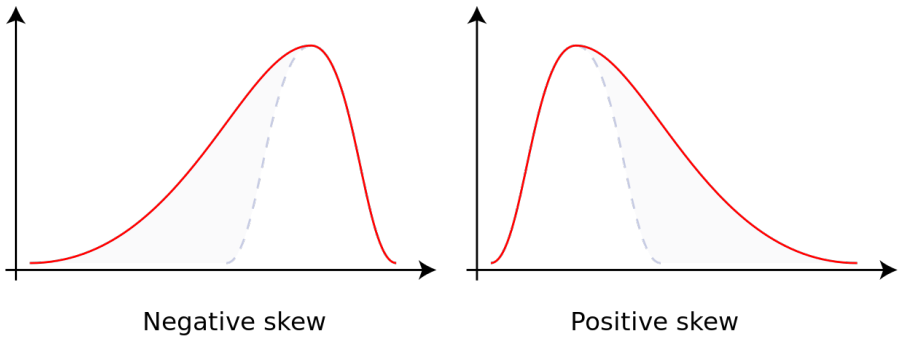


Figure 5.1.: Rodolfo Hermans (Godot) at en.wikipedia., CC BY-SA 3.0, via Wikimedia Commons

De la même façon, il existe aussi une façon de mesurer combien une distribution est pointue ou aplatie. Cette caractéristique se mesure à l'aide du **coefficient d'aplatissement** (*kurtosis*). Un coefficient d'aplatissement > 3 possède plus de données au centre (plus pointue) qu'une distribution normale. On parle d'une courbe **leptokurtique**. Un coefficient d'aplatissement < 3 est plus aplatie qu'une distribution normale (nous expliquerons plus en détail la loi normale au Chapitre 11), on parle alors de courbe **platikurtique**.

5.4. Labo : Décrire les données dans R

La plupart des descripteurs de données définis ci-dessus sont prêts à être utilisés dans R, à l'exception du coefficient d'asymétrie et du coefficient d'aplatissement, que nous ne ré-utiliserons de toute façon pas dans le restant de ce livre. Voici comment les calculer pour la variable `body_mass_g` de notre base de données `penguins` :

```
library(palmerpenguins)
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
summarize(penguins,
    moyenne = mean(body_mass_g, na.rm = TRUE),
    mediane = median(body_mass_g, na.rm = TRUE),
    variance = var(body_mass_g, na.rm = TRUE),
    ecart_type = sd(body_mass_g, na.rm = TRUE),
    etendue = max(body_mass_g, na.rm = TRUE) -
     ↪  min(body_mass_g, na.rm = TRUE)
)
```

```
# A tibble: 1 x 5
  moyenne mediane variance ecart_type etendue
```

5. Décrire les données

```
    <dbl>   <dbl>    <dbl>      <dbl>   <int>
1   4202.    4050  643131.       802.    3600
```

Remarquez que pour calculer la médiane, vous n'avez pas besoin de trier les données au préalable, la fonction `median` s'en occupe elle-même pendant son calcul.

5.5. Exercice : Intuitions quant aux descripteurs de données

Voici les observations concernant trois variables :

A = [0, 5, 10, 15, 20]

B = [10, 15, 20, 25, 30]

C = [8, 9, 10, 11, 12]

Répondez aux questions suivantes, d'abord par intuition (i.e. sans effectuer de calcul).

- Laquelle de ces variables possède la moyenne la plus élevée?
- Laquelle de ces variables possède la variance la plus élevée?
- Quelle est la médiane de chacune des trois variables?
- Laquelle de ces variables possède l'étendue la plus petite?

Puis validez vos réponses à l'aide d'un calcul *manuel* (i.e. avec votre calculatrice).

Ensuite, lancez le code suivant dans R pour créer un tableau de données contenant ces 3 variables :

```r
donnees <- tibble(
  A = c(0, 5, 10, 15, 20),
  B = c(10, 15, 20, 25, 30),
  C = c(8, 9, 10, 11, 12)
)
```

Enfin, en adaptant le code de la section Section 5.4, calculez la moyenne, la médiane, la variance et l'écart-type de chacune de ces variables. Vos résultats devraient être identiques à ceux calculés manuellement...

6. Programmer comme une pro

Une des premières choses à faire pour programmer comme une pro est de mettre son égo de côté. La grande majorité des programmeurs tapissent les murs de leurs cubicules d'aide-mémoires associés aux librairies de code qu'ils emploient. Je vous encourage donc fortement à imprimer les aide-mémoire associés aux libraries ggplot2[1] et dplyr[2] et à les avoir à vos côtés au moment de vous lancer dans R.

6.1. La meta-librairie `tidyverse`

Jusqu'à maintenant, nous avons utilisé pour nos travaux une série de librairies différentes. Nous avons entre autres utilisé les librairies `ggplot2`, `dplyr` et `tidyr`. Si c'est trois librairies travaillent si bien ensemble, c'est qu'elles font partie de la méta-librairie `tidyverse`. Cette dernière comprend une collection d'environ 25 librairies conçues par Hadley Wickham et ses collègues pour rendre le travail dans R plus facile, plus naturel.

Bien qu'elle soit une méta-librairie, `tidyverse` s'installe et s'active comme n'importe quelle librairie :

[1] https://raw.githubusercontent.com/rstudio/cheatsheets/main/data-visualization.pdf

[2] https://raw.githubusercontent.com/rstudio/cheatsheets/main/data-transformation.pdf

```
install.packages("tidyverse")
```

Cette opération pourrait prendre plusieurs minutes étant donné la quantité de librairies à installer.

> **!** Important
>
> N'utilisez la commande `install.packages` qu'une seule fois sur votre ordinateur. Lors des utilisations suivantes, vous n'avez qu'à l'activer avec la commande `library`.

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

Vous voyez dans ce message que l'activation de librairie `tidyverse` active automatiquement les librairies suivantes :

- ggplot2
- dplyr
- tidyr

- tibble
- readr
- forcats
- stringr
- lubridate

Au terme de ce livre, nous aurons utilisé toutes ces librairies, à l'exception de stringr et de lubridate, qui servent respectivement à manipuler du texte et à manipuler des dates. Ces deux actions sont très communes, mais dépassent le cadre de ce livre.

Autrement dit, en activant la librairie tidyverse, vous avez automatiquement votre coffre à outils entier à portée de main.

Vous remarquerez peut-être aussi une deuxième section au message d'activation, soit celui avec liste des conflits. Un conflit existe lorsqu'une fonction dans une librairie porte le même nom qu'une fonction déjà présente dans votre environnement de travail. Et cela arrive fréquemment.

Dans le cas présent, on nous informe que la fonction filter de la librairie dplyr est venue cacher la fonction filter de la librairie stats. Le message nous informe ensuite que, si jamais nous voulons utiliser la fonction filter qui a été écrasée, on peut l'appeler en utilisant stats::filter. Autrement dit, le nom de la librairie, suivi de ::, suivi du nom de la fonction.

Cette façon de faire permet aussi d'utiliser une fonction dans une librairie sans avoir à activer toute la librairie au préalable.

Par exemple, pour enlever les lignes contenant des valeurs manquantes, il serait tout à fait légitime de faire :

```
tidyr::drop_na(tableau)
```

plutôt que

```
library(tidyr)
drop_na(tableau)
```

La métaphore des boîtes

Pour bien comprendre ces nuances, il faut vous imaginer les librairies de R comme des boîtes remplies d'outils.

La fonction `install.packages` est comme un camion d'Amazon qui vous livre la boîte d'outils. Il n'a besoin de livrer la boîte qu'une seule fois, ensuite, elle est chez vous pour toujours.

La fonction `library` prend le contenu d'une des boîtes et le vide sur votre établi. Tous les outils sont maintenant disponibles pour travailler.

Un conflit, c'est comme si en vidant le contenu d'une des boîtes, un des nouveaux outils se déposait par-dessus l'ancien. On peut encore accéder à l'ancien, mais si on ne fait rien de spécial, on attrape le nouveau qui est tombé par-dessus.

Enfin, la notation `::`, permet de prendre un outil dans une des boîtes, de l'utiliser, et ensuite l'outil, attaché avec un élastique, retourne automatiquement dans sa boîte.

Les deux approches ont leurs avantages. Si vous avez besoin de beaucoup d'outils dans la boîte, ça peut être très pratique de vider toute la boîte (i.e. d'utiliser la fonction `library`). Mais si vous n'utilisez qu'un seul outil, une seule fois, peut-être que simplement utiliser l'outil et le retourner dans la boîte sera la meilleure chose à faire (i.e. utiliser `::` sera la façon la plus efficace).

6.2. Labo : Densifier son code ggplot2

Une des choses à savoir à propos des programmeurs est qu'ils sont extrêmement paresseux. Eux vous diraient probablement efficaces, mais d'une façon ou d'une autre, ils détestent perdre leur temps à faire des choses répétitives et cherchent constamment des raccourcis pour accélérer leur travail.

En sachant quelques principes de R supplémentaires, le code R pour nos graphiques pourrait être grandement raccourci, et donc notre risque d'erreur grandement diminué.

La première chose à savoir est que les associations (*mapping*) n'ont pas besoin d'être répétés pour chaque couche graphique, pour autant qu'ils aient été spécifiés à l'appel original de la fonction ggplot. N'oubliez pas de commencer par charger la librairie palmerpenguins pour notre tableau de données :

```
library(palmerpenguins)
```

Et ensuite, entrez le code suivant :

```
ggplot(data = penguins,
       mapping = aes(x = body_mass_g, y =
        ↪  flipper_length_mm)) +
  geom_point() +
  geom_text(mapping = aes(label = species))
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_text()`).
```

Premièrement, ce graphique est très laid! Ce n'est que pour donner un exemple...

Remarquez que le mapping de X et Y est maintenant dans la fonction `ggplot` plutôt que dans `geom_point`.

En travaillant ainsi, les variables `body_mass_g` et `flipper_length_mm` sont automatiquement associées aux coordonnées x et y pour toutes nos couches (`geom_point` et `geom_text`). Nous n'avons donc qu'à spécifier l'association supplémentaire dans notre couche de texte, où les étiquettes de nos données (`label`) proviendront de la variable `species` du tableau de données `penguins`.

Si jamais une propriété est mentionnée globalement et à l'intérieur d'une couche graphique, c'est l'association dans la couche graphique (dans le geom_) qui aura priorité pour cette couche en particulier.

L'autre notion importante à savoir pour densifier votre code encore plus est que dans R, le nom des arguments est optionnel, pour autant que l'on respecte l'ordre prescrit. Pour connaître cet ordre, il faut consulter l'aide de la fonction en question, p. ex.

```
?ggplot
```

Vous trouverez alors une ligne **Usage** qui devrait ressembler à ceci :

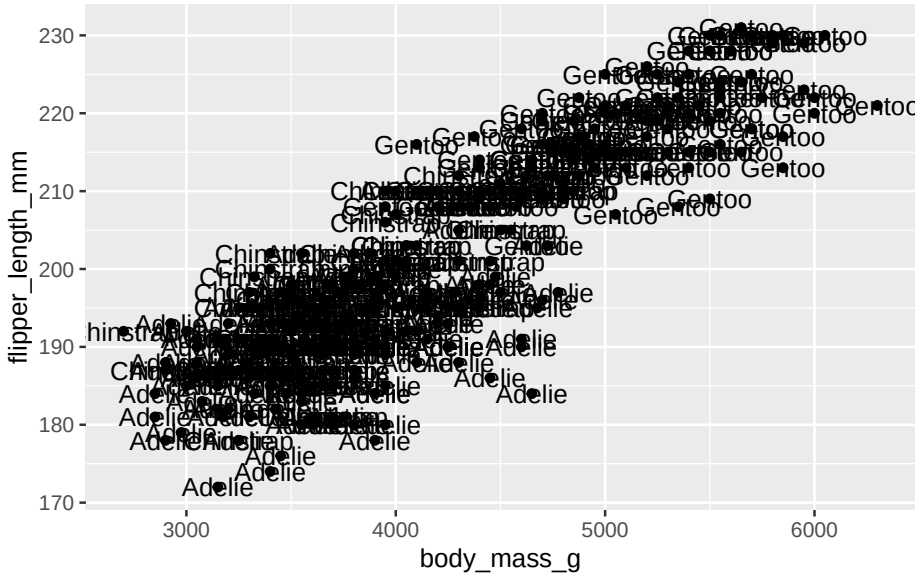
```
ggplot(data = NULL, mapping = aes(), ...,   environment
= parent.frame())
```

La section Usage de l'aide d'une fonction vous indique, entre autres, dans quel ordre R attend les arguments pour la fonction.

Si vous respectez cet ordre, vous n'avez pas besoin de nommer les arguments. Le code précédent pourrait donc être raccourci à ceci :

```
ggplot(penguins,aes(body_mass_g, flipper_length_mm)) +
  geom_point() +
  geom_text(aes(label = species))
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).

Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_text()`).
```



Évidemment, personne ne vous force à utiliser cette façon de faire, mais elle est très commune et utilisée dans la plupart des exemples en ligne. Je l'utiliserais pour le reste des notes de cours.

6.3. Labo : Enchaîner les opérations de dplyr

Dans la préparation de vos données pour leur analyse, vous rencontrerez souvent (presque toujours en fait!) des situations où vous aurez plusieurs opérations à effectuer sur votre tableau de données avant qu'il ne soit prêt pour l'analyse.

On pourrait par exemple démarrer avec notre tableau de données sur les manchots, et vouloir obtenir au bout du compte un tableau contenant l'île et l'année de chacun des mâles Gentoo en ordre croissant de longueur d'aile.

Avec ce que l'on a vu jusqu'à présent, on aurait pu faire comme ceci :

```
a <- filter(penguins, sex == "male" & species ==
 ↪  "Gentoo")
b <- arrange(a, flipper_length_mm)
select(b, island, year)
```

```
# A tibble: 61 x 2
   island  year
   <fct>  <int>
 1 Biscoe  2008
 2 Biscoe  2009
 3 Biscoe  2007
 4 Biscoe  2009
 5 Biscoe  2007
 6 Biscoe  2007
 7 Biscoe  2007
 8 Biscoe  2007
 9 Biscoe  2007
10 Biscoe  2008
# i 51 more rows
```

Vous constatez que pour y arriver, nous utilisons plusieurs objets intermédiaires (a et b) qui n'ont pas de fonction, autre que d'attendre l'opération suivante. L'utilisation de ces objets intermédiaires augmente aussi beaucoup le risque d'erreur.

Nous pourrions bêtement éliminer ces objets intermédiaires, comme ceci :

```
select(arrange(filter(penguins, sex == "male" & species
 ↪  == "Gentoo"), flipper_length_mm), island, year)
```

Ce code accomplit la même tâche, mais vous serez sûrement d'accord, il est vraiment plus difficile à comprendre. Les problèmes sont nombreux. D'abord, le tableau de données duquel on démarre est caché quelque part au milieu du code. Ensuite, il faut lire du centre vers l'extérieur pour comprendre ce que le code fera, ce qui n'est pas très naturel.

Il existe dans R une solution à ce problème, que j'ai traduit comme l'opérateur d'enchaînement (|>, *pipe operator*). Le travail de l'opérateur d'enchaînement est de nous permettre d'écrire du code facile à lire pour nous, et lui s'occupe de le retraduire pour l'ordinateur.

Voyons comment nous aurions pu améliorer notre code à l'aide de cet opérateur, puis nous discuterons un peu de son fonctionnement.

```
penguins |>
  filter(sex == "male" & species == "Gentoo") |>
  arrange(flipper_length_mm) |>
  select(island, year)
```

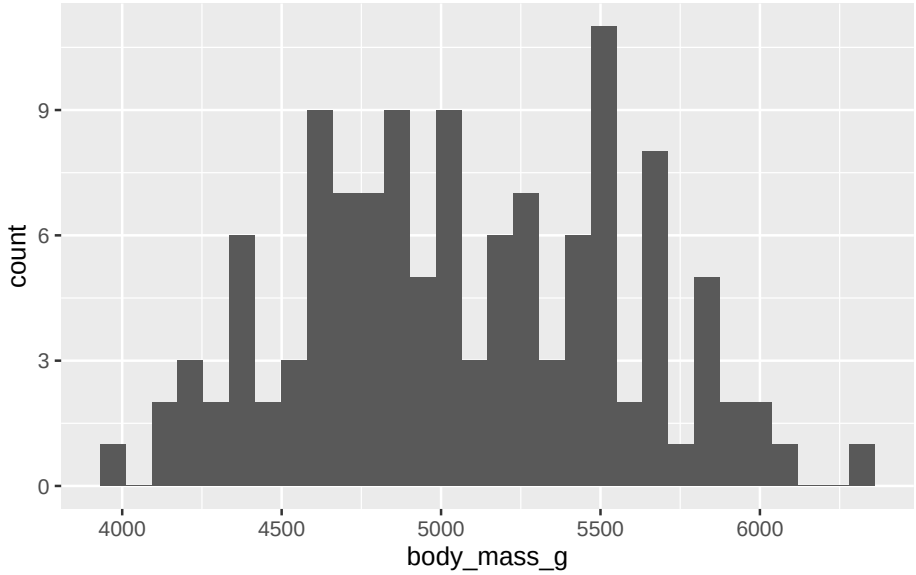
Notre code est maintenant beaucoup plus facile à lire, puisque l'on peut maintenant lire normalement de haut en bas et de gauche à droite. Il décrit de façon beaucoup plus directe ce que l'on voulait faire : démarrer du tableau `penguins`, filtrer, trier et ensuite choisir des colonnes. Le nom du tableau de départ est toujours la première chose dans la chaîne, et ensuite les verbes décrivant nos opérations sont bien mis en évidence.

Si vous observez attentivement le bout de code précédent, vous verrez que pour chacun des verbes (fonctions), nous n'avons pas mentionné sur quel tableau de données travailler. C'est le travail de l'opérateur d'enchaînement : il prend ce qu'on lui fournit à sa gauche, et l'envoie comme premier argument de ce que l'on met à sa droite. On peut donc l'utiliser avec n'importe quelle fonction qui s'attend à recevoir un tableau de données comme premier argument, incluant ggplot :

```
penguins |>
  filter(species == "Gentoo") |>
  ggplot(aes(body_mass_g)) +
  geom_histogram()
```

```
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```

```
Warning: Removed 1 row containing non-finite outside
the scale
range (`stat_bin()`).
```



Le code précédent nous fournit donc, de façon très compacte, un histogramme du poids des manchots Gentoo.

Notez la subtilité suivante : les opérations de dplyr s'enchaînent avec `|>` alors que les couches de ggplot2 s'ajoutent avec l'opérateur `+`. L'auteur de

ces librairies s'excuse à plusieurs reprises dans son livre de cet imbroglio, mais la charge de travail pour établir la constance entre les deux librairies serait trop importante et briserait trop de code pré-existant.

La question qui apparaît dans plusieurs têtes à ce moment est souvent : wow, c'est fou, mais eh, est-ce que je pourrais faire toute mon analyse dans une seule chaîne? La réponse simple est : vous faites ce que vous voulez!

Mais, en général, on recommande de limiter la longueur d'une chaîne à 5-6 opérations qui ont un rapport entre elles. Si vous avez plus d'opérations que cela à faire, il peut être plus avantageux de séparer la chaîne en plusieurs morceaux et de conserver le résultat intermédiaire dans un objet. Particulièrement au moment du débogage, vous serez contents d'avoir travaillé de cette façon.

6.4. Labo : Grouper pour mieux résumer

Nous avons vu au Chapitre 4 qu'il existe une fonction summarize dans dplyr qui nous permet de résumer les données rapidement. Cette fonction est en fait beaucoup plus puissante qu'elle ne le paraissait si on lui combine un opérateur de regroupement. Rappelons-nous d'abord le fonctionnement de la fonction summarize (cette fois en utilisant aussi l'opérateur d'enchaînement) :

```
penguins |>
  summarize(
    poids_moyen = mean(body_mass_g, na.rm = TRUE),
    ecart_type_poids = sd(body_mass_g, na.rm = TRUE)
  )
```

```
# A tibble: 1 x 2
  poids_moyen ecart_type_poids
```

```
        <dbl>             <dbl>
1       4202.              802.
```

Ce bout de code nous permet d'obtenir le poids moyen et l'écart-type du poids de manchots dans le tableau de données.

Si on ajoute l'opérateur de groupement, on pourrait obtenir ces chiffres par espèce :

```
penguins |>
  group_by(species) |>
  summarize(
    poids_moyen = mean(body_mass_g, na.rm = TRUE),
    ecart_type_poids = sd(body_mass_g, na.rm = TRUE)
  )
```

```
# A tibble: 3 x 3
  species   poids_moyen ecart_type_poids
  <fct>           <dbl>            <dbl>
1 Adelie          3701.             459.
2 Chinstrap       3733.             384.
3 Gentoo          5076.             504.
```

Ou même par espèce sur chacune des îles, etc.

```
penguins |>
  group_by(species, island) |>
  summarize(
    poids_moyen = mean(body_mass_g, na.rm = TRUE),
    ecart_type_poids = sd(body_mass_g, na.rm = TRUE)
  )
```

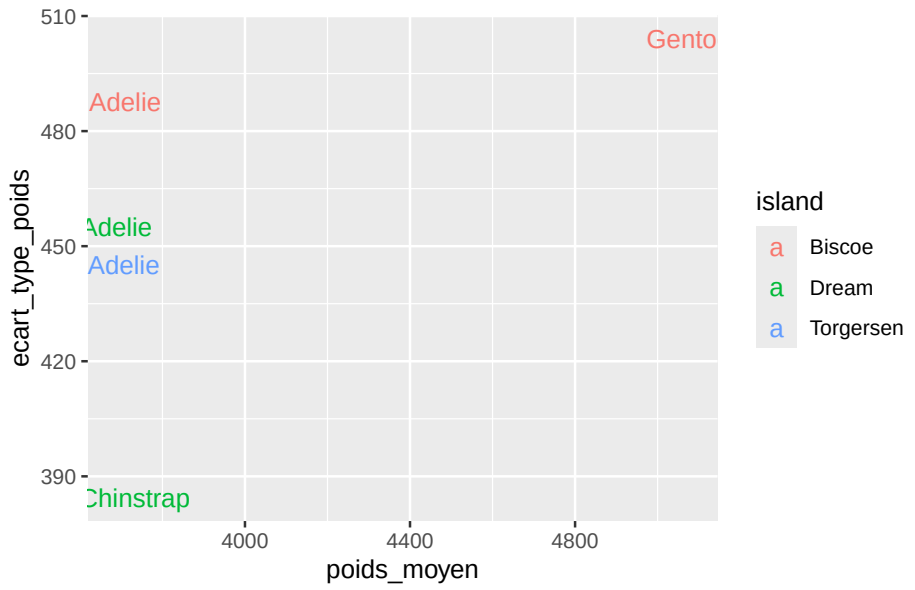
```
`summarise()` has grouped output by 'species'. You can
override using the `.groups` argument.
```

```
# A tibble: 5 x 4
# Groups:   species [3]
  species   island    poids_moyen ecart_type_poids
  <fct>     <fct>           <dbl>            <dbl>
1 Adelie    Biscoe          3710.             488.
2 Adelie    Dream           3688.             455.
3 Adelie    Torgersen       3706.             445.
4 Chinstrap Dream           3733.             384.
5 Gentoo    Biscoe          5076.             504.
```

Notez que l'objet retourné par summarize est lui aussi un tableau de données, qui peut être directement utilisé :

```
penguins |>
  group_by(species, island) |>
  summarize(
    poids_moyen = mean(body_mass_g, na.rm = TRUE),
    ecart_type_poids = sd(body_mass_g, na.rm = TRUE)
  ) |>
  ggplot(aes(poids_moyen, ecart_type_poids)) +
  geom_text(aes(label = species, color = island))
```

```
`summarise()` has grouped output by 'species'. You can
override using the `.groups` argument.
```



C'est donc de cette façon que l'on pouvait obtenir les chiffres du tableau de contingence du Chapitre 3, avec le code suivant :

```
penguins |>
  group_by(sex, species) |>
  summarize(
    n()
  )
```

```
`summarise()` has grouped output by 'sex'. You can
override using the `.groups` argument.

# A tibble: 8 x 3
# Groups:   sex [3]
  sex    species   `n()`
  <fct>  <fct>     <int>
```

```
1 female Adelie       73
2 female Chinstrap    34
3 female Gentoo       58
4 male   Adelie       73
5 male   Chinstrap    34
6 male   Gentoo       61
7 <NA>   Adelie        6
8 <NA>   Gentoo        5
```

Remarquez que pour ce cas particulier, il serait probablement plus efficace d'utiliser la fonction `table` de R de base :

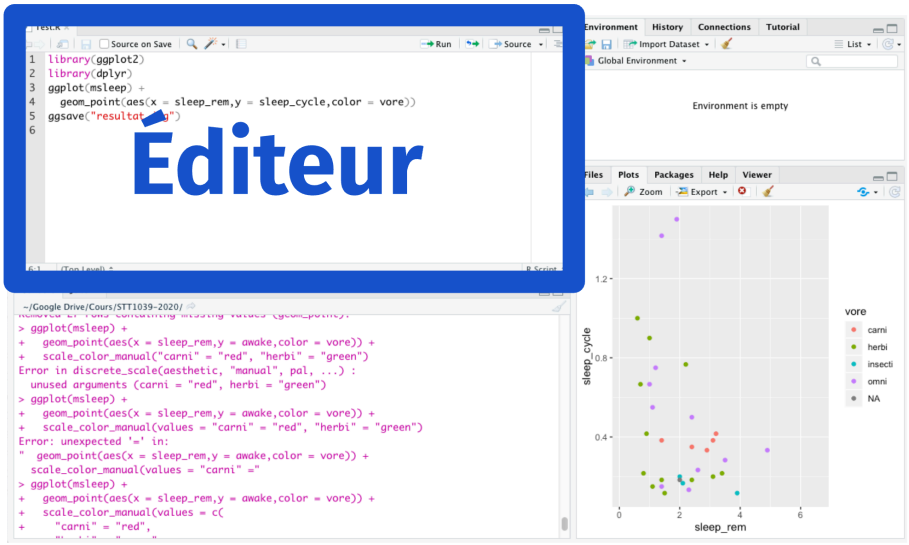
```
table(penguins$sex, penguins$species)
```

```
        Adelie Chinstrap Gentoo
  female     73        34     58
  male       73        34     61
```

6.5. Travailler avec des scripts

Vous avez probablement remarqué, particulièrement avec l'exemple où l'on a enchainé une chaîne de dplyr avec un graphique ggplot, qu'à mesure que notre code se complexifie, il devient de plus en plus ardu de construire nos commandes dans la petite ligne de la console de R. C'est pourquoi la majorité des gens qui travaillent avec R ne codent pas directement dans la console. Ils préparent plutôt des scripts (des séries de commandes) à l'aide d'un éditeur de code. Ils envoient ensuite leurs commandes déjà préparées à la console de R.

Dans RStudio, vous pouvez créer une nouvelle fenêtre de script à l'aide du menu File / New File / R Script. Votre environnement de RStudio se transformera alors comme ceci :



La partie de gauche sera séparée en deux, l'éditeur de code en haut, et la console de R en bas. Vous avez donc maintenant plein d'espace pour préparer vos lignes de code dans l'éditeur. Lorsqu'elles sont prêtes à envoyer à R, assurez que votre curseur est sur la bonne ligne et faites Ctrl+Enter (ou Cmd+Enter sur Mac) et RStudio enverra votre commande à la console. Outre le gain d'espace, l'avantage majeur à travailler de cette façon est que vous pouvez rassembler toutes les lignes de code associées à un projet, et sauvegarder ce script pour le réutiliser plus tard. Vous serez étonnés de voir le nombre de fois où un projet d'analyse s'étire à temps perdu sur plusieurs mois, voire même des années.

Une fois que vous travaillez de cette façon, il devient aussi intéressant d'insérer des lignes de commentaires dans votre code à l'aide du dièse ou hashtag (#). Toutes les lignes précédées d'un # seront considérées par R comme des commentaires et ne seront pas exécutées. Voici un petit exemple de script utilisant cette façon de faire :

```
# Chargement des librairies
library(tidyverse)
library(palmerpenguins)

# Création du premier graphique
ggplot(penguins) +
  geom_point(aes(body_mass_g, flipper_length_mm))

# Sauvegarde du graphique
ggsave("resultat.jpg")
```

Remarquez aussi que les lignes dans l'éditeur de code sont numérotées pour faciliter la discussion. Vous verrez peut-être parfois apparaître des avertissements (par exemple une bulle rouge avec un X) pour vous prévenir que cette ligne contient probablement des erreurs.

6.6. Le clavier est votre ami!

Une des choses que vous remarquerez si vous observez un programmeur travailler et qu'il touche rarement à la souris pendant qu'il code. La majorité du temps, ses doigts demeurent sur le clavier. Encore une fois, il s'agit d'une question d'efficacité. En connaissant quelques raccourcis clavier, votre travail gagnera en vitesse de façon spectaculaire.

Voici ceux que j'utilise le plus souvent :

Raccourci	Windows	Mac
Opérateur d'enchaînement (|>)	Ctrl + Maj + M	Cmd + Maj + M
Opérateur d'assignation (<-)	Alt + -	Option + -

Raccourci	Windows	Mac
Lancer le code R du haut de la page jusqu'au curseur	Ctrl + Alt + B	Cmd + Option + B
Relancer une nouvelle fois le bloc de code que l'on vient d'exécuter	Ctrl + Maj + P	Alt + Cmd + P
Annuler la dernière action dans l'éditeur de code	Ctrl + Z	Cmd + Z

Vous pouvez accéder à la liste complète des raccourcis de RStudio avec la commande Alt + Maj + K (Option + Maj + K sur Mac)

L'autre chose que je vous conseille fortement d'apprendre sur votre clavier est l'emplacement de certaines touches spéciales. Comme la plupart des langages de programmation, R utilise plusieurs caractères spéciaux pour définir certaines opérations. Ces caractères spéciaux sont rarement utilisés dans la vie de tous les jours et nous connaissons rarement leur emplacement sur nos claviers. Néanmoins, puisqu'ils reviennent fréquemment dans R, je vous conseille de les noter sur un post-it collé sur votre écran une fois que vous les aurez trouvés la première fois. Comme votre flow de pensée de programmation sera très fragile, particulièrement dans les débuts, il est préférable de ne pas perdre le fil chaque fois que vous aurez besoin d'écrire un caractère étrange.

Voici en rafale celles dont vous aurez besoin le plus souvent :

```
! # $ % ? & * ( ) [ ] ^ ~ < > | "
```

Notez que comme tous vos claviers sont différents, il m'est impossible ici de vous fournir la bonne façon pour vous d'écrire ces symboles sur votre ordinateur. C'est à vous de les trouver!

7. Saisir des données

Jusqu'à maintenant, chaque fois que nous avons travaillé avec des données, nous avons utilisé des tableaux de données déjà prêts dans R. Nous avons ainsi évité un problème critique : la saisie de nouvelles données! Nous verrons dans ce chapitre trois façons de faire notre propre entrée de données dans R, soit la saisie directe, l'importation de données à partir d'un fichier texte (CSV) et enfin l'importation à partir d'un fichier Excel.

7.1. Labo : La saisie directe

Si vous avez peu de données à entrer, la façon la plus simple de les saisir est probablement d'écrire à la main le code qui créera le tableau de données dans R. La fonction pour créer un nouveau tableau de données dans R se nomme data.frame et s'utilise comme ceci :

```
tableau <- data.frame(
  code_espece  = c("GEBL", "METN", "COAM","CARO"),
  poids_g = c(100, 11, 400,NA),
  decompte = c(12,8,3,1)
)

tableau
```

Et R vous montrera ainsi le tableau nouvellement créé :

```
  code_espece poids_g decompte
1        GEBL     100       12
2        METN      11        8
3        COAM     400        3
4        CARO      NA        1
```

Notez plusieurs choses importantes à propos du code précédent. Pour chacune des variables de notre tableau, la série de valeurs doit être emballée par la fonction `c()` qui sert à coller ensemble des valeurs pour créer ce que R nomme un vecteur (vector). Dans la fonction `c`, chacun des éléments doit être séparé par une virgule. Remarquez aussi que les valeurs qualitatives doivent être emballées chacune par une paire de guillemets. Observez aussi que pour la valeur manquante, il faut saisir le `NA` directement, sans guillemets. Il ne s'agit pas d'une valeur qualitative, mais bien d'un mot-clé réservé de R. Enfin, remarquez que chacune des variables ne doit contenir qu'un seul type de données (soit qualitative ou quantitative), jamais un mélange des deux.

Cette façon de faire par la saisie directe est très pratique pour écrire de petits exemples pour tester un bout de code, ou lorsque vos tableaux de données sont de très petite taille.

7.2. Labo : Les fichiers CSV

La façon classique d'entrer des données dans le logiciel R est d'utiliser des fichiers CSV. Un fichier CSV est un fichier de texte, qui peut être ouvert avec n'importe quelle application, par exemple avec le Bloc Notes de Windows. La particularité d'un fichier CSV est que dans chacune des lignes qui le compose, les valeurs sont séparées par des virgules, d'où son acronyme de CSV : *Comma Separated Values*. Il est possible (mais pas obligatoire), de mentionner le nom dans variables, séparées aussi par des virgules, dans la première ligne du fichier.

Les données de l'exemple sur la saisie directe seraient organisées dans un fichier CSV comme ceci :

```
"GEBL",100,12
"METN",11,8
"COAM",400,3
"CARO",NA,1
```

Vous pouvez télécharger ce fichier d'exemple[1] ou le créer vous-même en copiant-collant le contenu.

> **Mise en garde**
>
> Votre ordinateur vous montrera probablement l'icône de Excel pour ce fichier, mais si vous regardez bien son nom, il se termine par .csv et non .xlsx

Si vous avez vérifié votre fichier à l'aide du Bloc Note et qu'il contient effectivement ce format, il peut être importé à l'aide de la fonction `read.csv`, comme ceci :

```
read.csv("oiseaux.csv")
```

```
Warning in file(file, "rt"): cannot open file
'oiseaux.csv': No such file or directory
```

```
Error in file(file, "rt"): cannot open the connection
```

Et c'est là que les problèmes commencent...

À moins d'un sacré coup de chance, votre ligne `read.csv` n'aura pas fonctionné du premier coup et la prochaine section explorera pourquoi...

[1] https://drive.google.com/file/d/13xxYYbtmFVYiXf0sqBFoCg-B7HLecAo1/view?usp=sharing

7.3. Labo : Le concept de dossier de travail dans R

Le principe d'un dossier de travail est pratiquement aussi vieux que l'informatique. Par contre, la plupart des logiciels modernes font fit de ce concept, ce qui fait que probablement peu d'entre-vous on eu à la comprendre jusqu'ici.

Lorsque vous travaillez dans R et essayez de lire (ou d'écrire) un fichier, R ne fouille pas partout sur votre ordinateur pour le trouver, il regarde à un seul endroit, son dossier de travail (*working directory*).

Vous pouvez demander à R où se trouve actuellement son dossier de travail, à l'aide de la fonction **getwd** (*GET Working Directory*). Sur mon portable personnel, il me répond pour le moment ceci :

```
getwd()
```

```
[1]
"/Users/charlesmartin/Documents/GitHub/NotesDeCours"
```

Ce dossier sera différent sur chacun de vos ordinateurs, et c'est tout à fait normal.

Si votre fichier oiseaux.csv ne se trouve pas dans ce dossier, vous avez alors deux choix. Votre première option est de déplacer votre fichier oiseaux.csv dans le dossier de travail de R. Pour se faire, assurez-vous d'utiliser soit l'Explorateur sur Windows ou le Finder sur Mac.

> **Mise en garde**
>
> N'utilisez surtout PAS la commande "Enregistrer sous..." d'un logiciel (Bloc Note, Excel, etc.) pour déplacer votre fichier.

Le format CSV est relativement fragile à ce genre d'opération et toute re-sauvegarde risque de le rendre inutilisable.

Votre deuxième option est de modifier le dossier de travail de R. Dans RStudio, vous pouvez modifier votre dossier de travail à l'aide de l'item de menu "Session / Set Working Directory / Choose directory ..." et choisir le dossier où se trouve votre fichier de données. Vous pouvez ensuite refaire la commande `read.csv`, qui devrait maintenant fonctionner, comme par magie!

Il est aussi possible de modifier le dossier de travail par programmation, avec la fonction nommée `setwd` (SET Working Directory), qui s'utilise comme ceci :

```
setwd("/Users/charlesmartin/Desktop/")
getwd()
```

```
[1] "/Users/charlesmartin/Desktop"
```

Savoir quel chemin entrer dans la commande `setwd` n'est pas sorcier, mais l'opération demande tout de même une certaine connaissance du système de dossiers de votre ordinateur.

Si la magie n'a pas opéré, n'ayez pas peur de me demander de l'aide pour que je regarde avec vous ce qui se passe. Si ce cours avait été un cours "normal" en présentiel, j'aurais passé une bonne partie du cours à aider les gens à ouvrir le fichier sur leur ordinateur parce que ça n'aurait pas fonctionné. C'est une étape vraiment pas simple.

7.4. L'enfer des fichiers CSV pour une francophone

Les fichiers CSV comportent plusieurs avantages qui en ont fait, au fil du temps, le format par excellence pour plusieurs applications. Ils peuvent être lus par n'importe quel logiciel qui traite du texte, et ce, depuis 50 ans. Ils sont légers, rapides à ouvrir, relativement robustes, etc.

Par contre, pour une francophone qui fait ses biostatistiques au Québec, le portrait est beaucoup moins rose. Faisons une petite liste rapide des problèmes possibles : notre séparateur de décimale (contrairement aux anglophones) est la virgule plutôt que le point. Ça part mal, parce que dans un fichier CSV, la virgule devrait servir à séparer les colonnes. Il existe par contre un second format de fichier, CSV2, qui sépare les valeurs par des points-virgules (et une fonction associée, `read.csv2` dans R).

Pour ajouter à la confusion, si jamais vous essayez d'exporter au format CSV à partir de la version française d'Excel, sachez que même si vous choisissez dans le menu d'Excel le format CSV, il utilisera, sans vous le dire, le format CSV2. Ça peut être un sacré casse-tête quand on essaie d'ouvrir notre fichier par la suite dans R et qu'on croit avoir un vrai CSV entre les mains.

Ensuite, les fichiers textes (comme les CSV) ont été inventés à une époque où l'informatique se déroulait quasi-exclusivement en anglais. Ce qui fait que la façon pour sauvegarder des lettres accentuées a été ajoutée un peu tout croche par la suite, par plein d'organisations différentes. Ce qui fait qu'aujourd'hui, il existe des dizaines de façon différentes de le faire (par exemple UTF-8, latin1, etc). Des solutions modernes comme Excel gardent une trace du type d'encodage utilisé dans le fichier, alors que pour le CSV, cette information est absente. R prend un *guess*, et si il se trompe, c'est à nous manuellement d'essayer des encodages différents jusqu'à ce que l'on tombe sur le bon.

Pour toutes ces raisons, je vous conseille (si vous créez vos propres fichiers de données) de les laisser au format Excel et de les lire directement dans ce format, comme expliqué dans la section suivante.

Remarquez que, si quelqu'un vous envoie un fichier CSV, ouvrez-le comme tel. Vous ne serez pas gagnantes à le transformer d'abord en Excel. Vous vous causerez juste encore plus de problèmes.

7.5. Labo : Les fichiers Excel

La lecture des fichiers Excel à partir de R ne fait pas partie des fonctionnalités de base. Une librairie de code doit donc être activée pour lire ces fichiers :

```
library(readxl)
```

Si jamais vous n'aviez pas installé la librairie `tidyverse` dans les chapitres précédents, vous pouvez installer directement la librairie readxl comme ceci :

```
install.packages("readxl")
```

Et ensuite l'activer avec `library`.

Et elle peut ensuite s'utiliser comme ceci :

```
read_excel("Oiseaux.xlsx")
```

En respectant les mêmes contraintes concernant le dossier de travail que pour les CSV. Vous pouvez télécharger mon fichier Excel[2] pour essayer la fonction sur votre ordinateur.

[2] https://drive.google.com/file/d/1T7aATqpy1WKlLNuZP2uxRRtxi2ZGunbm/view?usp=sharing

7.6. Petits conseils sur la création de vos propres fichiers Excel

Les fichiers Excel, bien que beaucoup plus faciles à gérer que les fichiers CSV, ne sont pas pour autant parfaits. Ils ne vous épargneront pas la tâche de créer de bons fichiers, bien organisés.

Voici quelques règles en rafale pour vous éviter les ennuis :

- Les méta-informations (qui a saisi les données, comment elles ont été mesurées, quand, etc.) ne devraient pas être saisies dans la même feuille que les données.
- Assurez-vous que vos noms de colonnes ne contiennent pas d'espaces ou de lettres accentuées, ne commencent pas par des chiffres et ne sont pas dédoublés.
- N'utilisez jamais la fusion de cellules dans un fichier servant à l'importation de données. R s'attend à recevoir un tableau rectangulaire. Pas une structure étrange ou parfois certaines lignes contiennent moins de colonnes que les autres.
- Assurez-vous, pour chacune de vos colonnes, que Excel est capable d'effectuer correctement des calculs avec vos chiffres. Selon votre version d'Excel, il est possible que vous ayez à utiliser le point plutôt que la virgule comme séparateur de décimales. L'important est d'utiliser celui qui permet à votre version de Excel d'effectuer des calculs. Vos chiffres seront ainsi toujours corrects dans R par la suite.
- Si jamais vous avez des données manquantes à saisir, laissez des blancs dans Excel. Ce sera beaucoup plus simple à gérer que d'insérer des "NA" qui pourraient être mal interprétés comme des valeurs textuelles. R remplacera lui-même vos blancs par des NA.

Enfin, si jamais vous avez à saisir de l'information incluant des noms de gènes, n'utilisez PAS Excel pour saisir vos informations. Plusieurs versions de Excel interprétaient mal certains noms de gènes, qu'elles modifiaient

sans en avertir l'utilisateur, ce qui a causé la réévaluation de plusieurs centaines de découvertes scientifiques ayant analysé des données erronées[3]. Le problème était grave au point où les experts ont simplement décidé de renommer les gènes pour éviter les problèmes à l'avenir[4].

Une autre preuve comme quoi il peut être très dangereux de travailler à l'aveugle sans bien explorer et valider nos données avant de commencer...

7.7. Labo : Toujours bien vérifier les données après la saisie

> **Mise en garde**
>
> Peu importe le format que vous choisirez pour saisir vos données dans R, il est très important de vérifier vos données après l'importation.

Pour se faire, vous avez deux outils principaux. Le premier est la fonction `str` (*STRucture*), qui vous renseigne sur le type de chacune des variables dans votre tableau de données, par exemple comme ceci :

```
library(palmerpenguins)
str(penguins)
```

```
tibble [344 x 8] (S3: tbl_df/tbl/data.frame)
 $ species          : Factor w/ 3 levels
 "Adelie","Chinstrap",..: 1 1 1 1 1 1 1 1 1 1 ...
```

[3] https://www.bbc.com/news/technology-37176926#:~:text=Microsoft's%20Excel%20has%20been%20blamed,altered%20to%20%22September%202%22.

[4] https://www.theverge.com/2020/8/6/21355674/human-genes-rename-microsoft-excel-misreading-dates

```
$ island           : Factor w/ 3 levels
"Biscoe","Dream",..: 3 3 3 3 3 3 3 3 3 3 ...
$ bill_length_mm   : num [1:344] 39.1 39.5 40.3 NA 36.7
39.3 38.9 39.2 34.1 42 ...
$ bill_depth_mm    : num [1:344] 18.7 17.4 18 NA 19.3
20.6 17.8 19.6 18.1 20.2 ...
$ flipper_length_mm: int [1:344] 181 186 195 NA 193 190
181 195 193 190 ...
$ body_mass_g      : int [1:344] 3750 3800 3250 NA 3450
3650 3625 4675 3475 4250 ...
$ sex              : Factor w/ 2 levels
"female","male": 2 1 1 NA 1 2 1 2 NA NA ...
$ year             : int [1:344] 2007 2007 2007 2007
2007 2007 2007 2007 2007 2007 ...
```

L'important dans la sortie de cette fonction est de regarder pour chacune des variables le type que R a choisi de lui attribuer. Ici, nous avons 3 variables qualitatives (fct pour *factor*) , 3 variables quantitatives discrètes (int pour *integer*) et 2 variables quantitatives continues (num pour *numeric*). Si vos données qui devaient être des chiffres ont été chargées en texte (chr ou fct), vous devrez vérifier votre code ou votre fichier.

L'autre fonction importante à lancer après une importation est la fonction `summary`. Cette dernière vous fournit un résumé de vos données :

```
summary(penguins)
```

```
      species          island    bill_length_mm
 Adelie   :152   Biscoe   :168   Min.   :32.10
 Chinstrap: 68   Dream    :124   1st Qu.:39.23
 Gentoo   :124   Torgersen: 52   Median :44.45
                                 Mean   :43.92
                                 3rd Qu.:48.50
                                 Max.   :59.60
```

```
                                NA's   :2
 bill_depth_mm   flipper_length_mm  body_mass_g
 Min.   :13.10   Min.   :172.0     Min.   :2700
 1st Qu.:15.60   1st Qu.:190.0     1st Qu.:3550
 Median :17.30   Median :197.0     Median :4050
 Mean   :17.15   Mean   :200.9     Mean   :4202
 3rd Qu.:18.70   3rd Qu.:213.0     3rd Qu.:4750
 Max.   :21.50   Max.   :231.0     Max.   :6300
 NA's   :2       NA's   :2         NA's   :2
     sex           year
 female:165   Min.   :2007
 male  :168   1st Qu.:2007
 NA's  : 11   Median :2008
              Mean   :2008
              3rd Qu.:2009
              Max.   :2009
```

Les sorties sont peu utiles pour les variables textuelles, mais très importantes pour les variables numériques. Vous voyez pour chacune, entre autres, la valeur minimum et maximum de votre variable, ainsi que sa moyenne et, le cas échéant, le nombre de valeurs manquantes.

Pour chacune de vos variables, prenez toujours quelques secondes pour vous demander si ces chiffres ont du sens. Il peut arriver que R rate le séparateur de décimales ou de milliers, et que vos données soient complètement erronées…

8. Améliorer ses graphiques

Nous avons vu au Chapitre 3 comment explorer rapidement un tableau de données à l'aide de graphiques. Dans le présent chapitre, nous verrons maintenant comment les personnaliser, pour les rendre dans un état acceptable pour être insérés dans un rapport ou un article scientifique.

8.1. Labo : Changer le thème du graphique

Le premier changement que vous serez peut être appelé à apporter à vos graphiques est d'en changer le thème. Avant de vous casser la tête à changer la couleur de fond, enlever la grille, etc., commencez par explorer les thèmes fournis par ggplot2[1].

Outre le thème de base, je vous conseille de commencer par regarder le thème classique, qui ressemble beaucoup aux graphiques de base de R (et donc à ceux que beaucoup de gens s'attendent de voir) :

```
library(tidyverse)
```

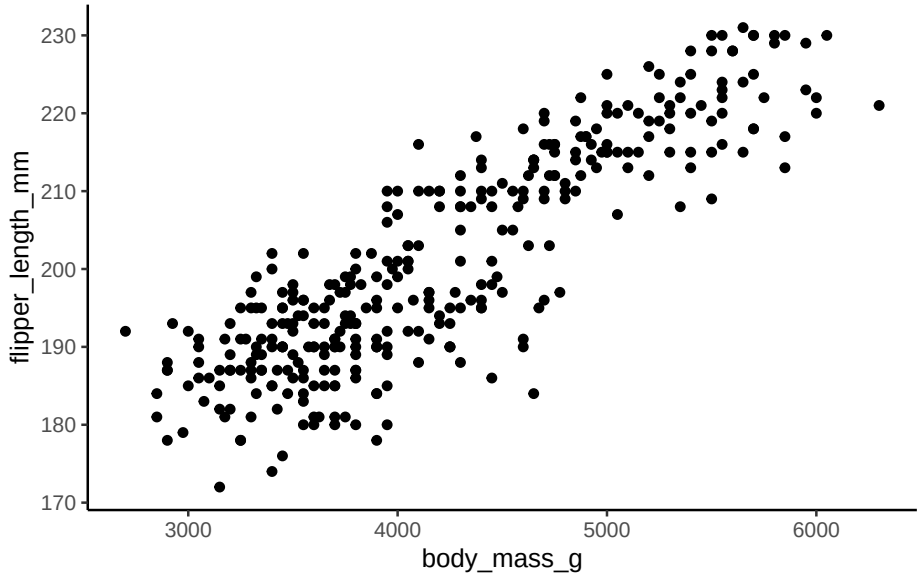
```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
```

[1] https://r4ds.had.co.nz/graphics-for-communication.html#fig:themes

```
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ----------------- tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(palmerpenguins)

ggplot(data = penguins) +
  geom_point(mapping = aes(x = body_mass_g, y =
   ↪  flipper_length_mm)) +
  theme_classic()
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```

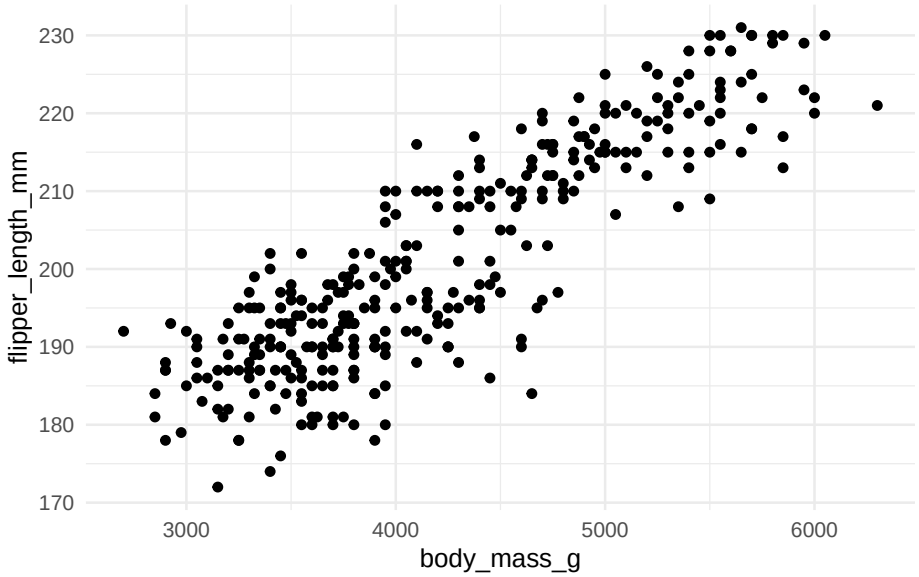


Les thèmes de ggplot2 sont définis comme une couche que l'on ajoute au reste du graphique, à l'aide du +.

De mon côté, j'ai tendance à souvent utiliser le thème minimal, que je trouve aussi clair que celui de base, mais moins chargé :

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = body_mass_g, y =
  ↪  flipper_length_mm)) +
  theme_minimal()
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```



Peu importe le thème que vous choisirez, vous économiserez beaucoup de temps de fignolage en démarrant du thème le plus près possible du résultat que vous recherchez.

Pour les curieux, sachez qu'il existe aussi des librairies de thèmes additionnels qui vous permettent de répéter des thèmes classiques comme ceux du Wall Street Journal et d'Excel[2] alors que d'autres librairies vous permettent de créer un thème basé sur certaines séries télé, par exemple Les Simpsons, Bob L'Éponge ou Game of Thrones[3].

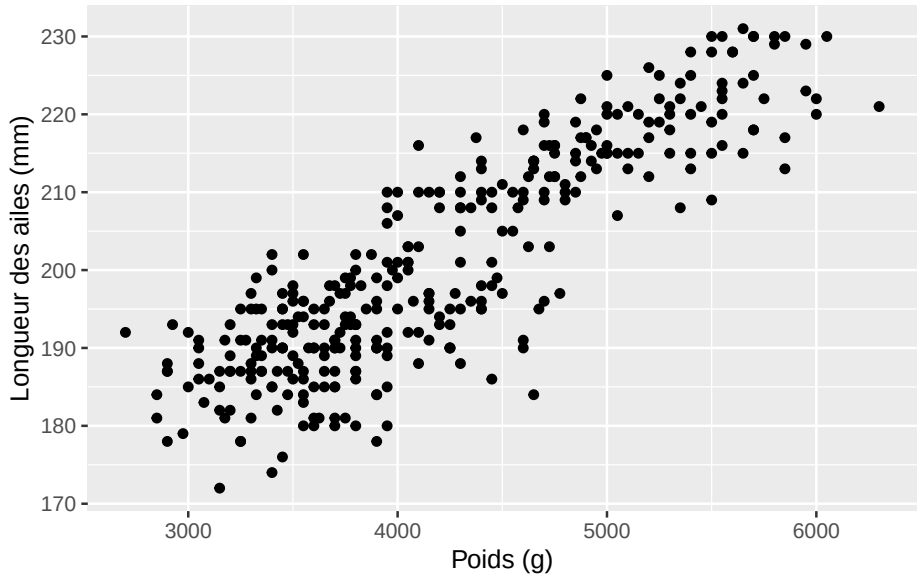
[2] https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/
[3] https://ryo-n7.github.io/2019-05-16-introducing-tvthemes-package/

8.2. Labo : Changer les étiquettes

Un élément important qui fera passer vos graphiques de “exploratoire” à “prêt à publier” est l’ajout d’étiquettes. Les premières, à changer systématiquement, sont celles définissant les axes X et Y. La fonction pour modifier les axes dans ggplot2 se nomme `labs` (*labels*) et s’utilise comme ceci :

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = body_mass_g, y =
  ↪  flipper_length_mm)) +
  labs(x = "Poids (g)", y = "Longueur des ailes (mm)")
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```

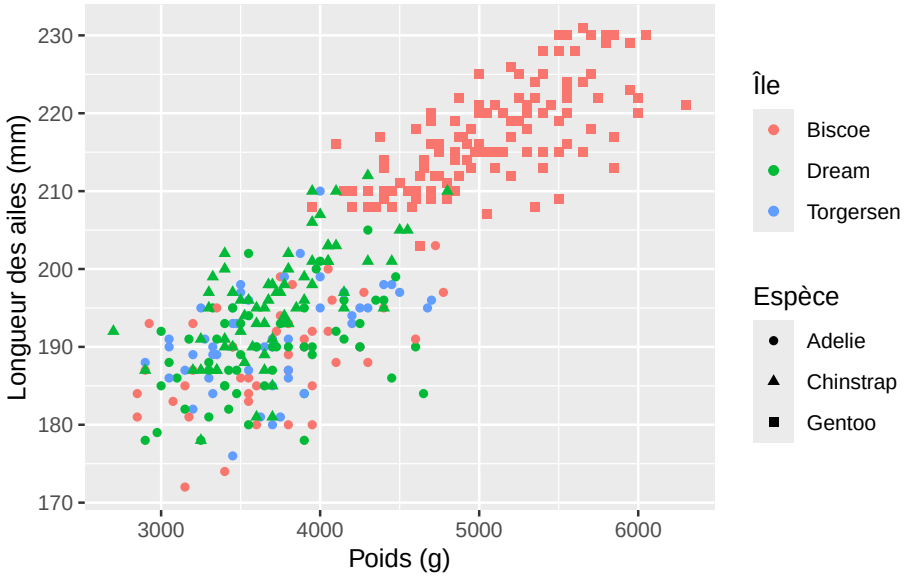


Ces étiquettes doivent être dans la langue utilisée dans le rapport (donc en français pour vos travaux), et mentionner les unités utilisées pour chacune des variables.

La fonction `labs` permet aussi d'attribuer des étiquettes pour les autres propriétés de nos couches graphiques, en ré-utilisant exactement la même terminologie que dans la couche graphique :

```
ggplot(data = penguins) +
  geom_point(mapping = aes(
    x = body_mass_g,
    y = flipper_length_mm,
    color = island,
    shape = species
  )) +
  labs(
    x = "Poids (g)",
    y = "Longueur des ailes (mm)",
    color = "Île",
    shape = "Espèce"
  )
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```



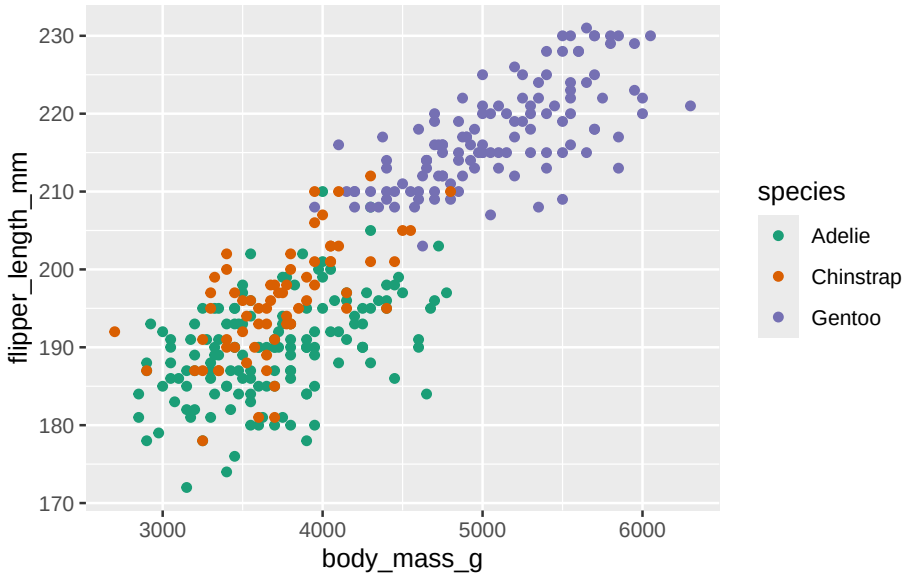
8.3. Labo : Palette de couleurs

Lorsque vous assignez une variable de votre tableau de données à la propriété `color` ou `fill`, il est possible que vous ayez envie (ou besoin) d'utiliser des couleurs autres que celles fournies par ggplot2. Il existe deux stratégies pour y arriver, soit en utilisant des palettes de couleurs pré-fabriquées, ou soit en choisissant une couleur spécifique pour chaque valeur de votre variable.

Pour modifier la palette de couleur d'une propriété graphique, utilisez la fonction `scale_colour_brewer`, comme ceci :

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = body_mass_g, y =
   ↪  flipper_length_mm,color = species)) +
  scale_color_brewer(palette = "Dark2")
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```



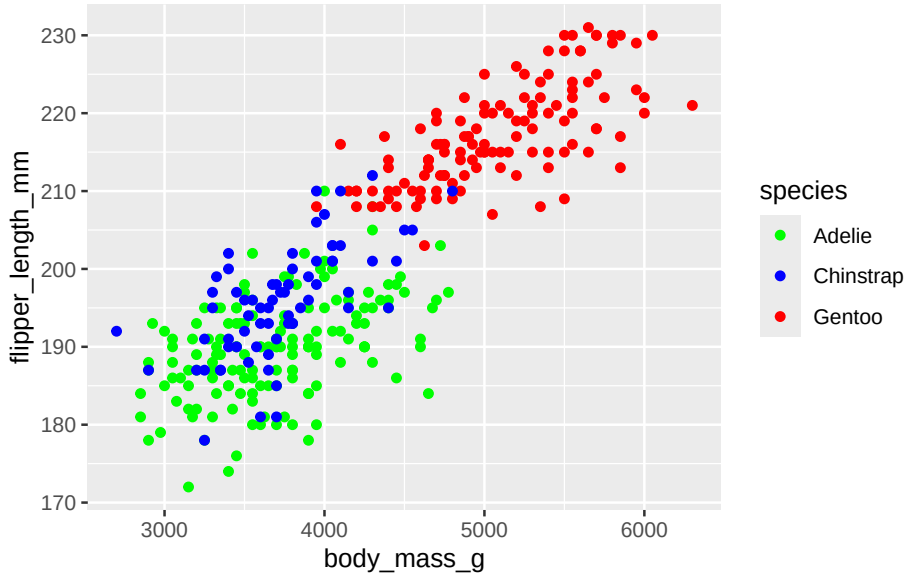
Vous pouvez avoir un aperçu de la liste des palettes de couleurs disponibles dans le livre gratuit R for Data Science[4].

Plutôt que de choisir une palette de couleur pré-fabriquée, vous pouvez aussi choisir vous même les couleurs, à l'aide de la fonction scale_color_manual :

[4] https://r4ds.had.co.nz/graphics-for-communication.html#fig:brewe

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = body_mass_g, y =
   ↪  flipper_length_mm,color = species)) +
  scale_color_manual(values = c(
    "Gentoo" = "red",
    "Adelie" = "green",
    "Chinstrap" = "blue"
  ))
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```



Remarquez que pour cette fonction aussi, les paires de valeurs doivent être enveloppées par la fonction c. Comme mentionné au Chapitre 3, la

plupart des noms de couleurs qui vous viendront à l'esprit sont définis dans R, mais en cas de doute, vous pouvez facilement en trouver des listes sur internet[5]. Aussi, essayez lorsque vous choisissez vos couleurs, d'avoir une petite pensée pour vos collègues daltoniens, qui représentent environ 5 % de la population.

8.4. Labo : Sauvegarder correctement

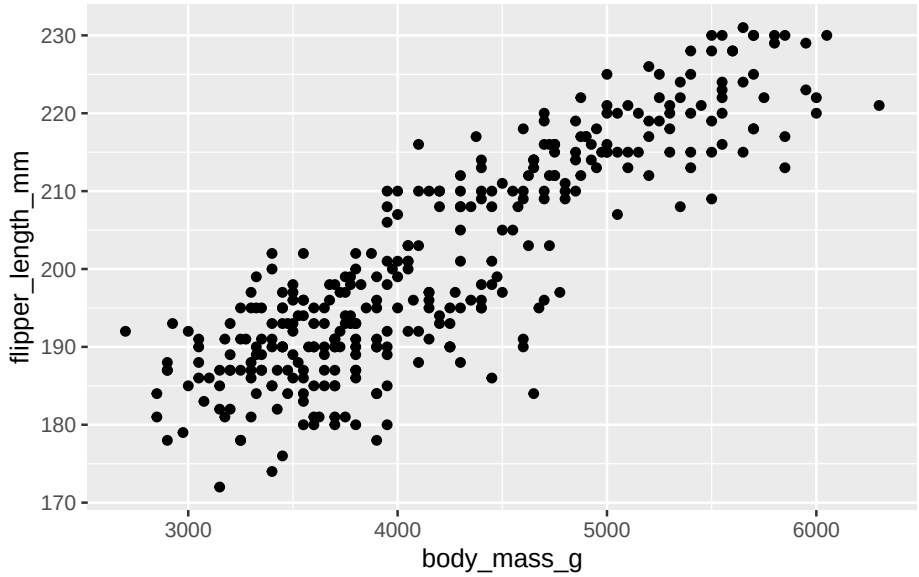
Vous avez probablement remarqué qu'au-dessus de la fenêtre graphique de RStudio, il existe un bouton Export vous permettant de conserver vos graphiques dans des fichiers externes (jpg, pdf, etc.) Bien que très pratique, cette façon de faire comporte aussi certains désavantages, en particulier au niveau de la reproductibilité. Si vous avez à refaire un graphique (et ça arrive plus souvent qu'on le voudrait), vous devrez remettre la fenêtre exactement de la même façon et choisir exactement le même nombre de pixels dans chaque dimensions. De plus, vous n'avez pas un contrôle indépendant de la résolution et des proportions du graphique.

C'est pourquoi je vous propose pour sauvegarder vos graphiques d'utiliser la fonction **ggsave**. Cette fonction sauvegarde dans un fichier le dernier graphique ggplot2 que vous avez fait afficher, comme ceci :

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = body_mass_g, y =
   ↪  flipper_length_mm))
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```

[5] https://derekogle.com/NCGraphing/img/colorbynames.png




```
ggsave("resultat.jpg")
```

```
Saving 5.5 x 3.5 in image
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```

R vous répond alors qu'il a créé un fichier pour vous en nommant ses dimensions.

Si vous allez dans votre dossier de travail (voir la Section 7.3 pour un rappel), vous devriez retrouver un fichier nommé resultat.jpg avec votre graphique à l'intérieur. L'extension choisie (jpg, gif,png,pdf) déterminera le format du fichier de sortie.

La première chose à faire lorsque vous sauvegardez un graphique est de déterminer les proportions que votre graphique doit présenter. Vous

contrôlez ces proportions à l'aide des arguments **width** (largeur) et **height** (hauteur). Par défaut, ggplot2 s'attend à recevoir ces valeurs en pouces :

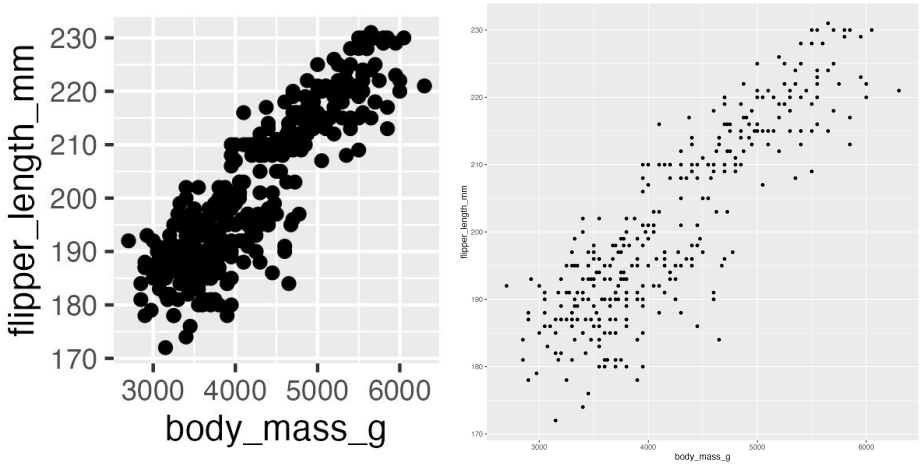
```
ggsave(filename = "2x2.jpg", width = 2, height = 2)
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```

```
ggsave(filename = "8x8.jpg", width = 8, height = 8)
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```

Comparez le résultat de ces deux fonctions :



R essaie d'adapter les proportions des éléments pour qu'ils soient beaux dans un graphique de la taille spécifiée. Dans le cas du 2x2, les éléments

sont beaucoup plus grands toutes proportions gardées, alors que dans le 8x8, ils sont beaucoup plus petits. Vous pouvez aussi expérimenter avec des proportions plus rectangulaires, par exemple 3x5, 1x8, etc...

Une fois que vous avez déterminé la bonne taille pour vos éléments graphiques, vous pouvez contrôler la qualité (la résolution) de votre graphique, à l'aide de l'argument `dpi`. DPI est l'abréviation de *dots per inch*, ou en français points par pouce. Il contrôle la pixellisation de votre graphique.

```
ggsave(filename = "72.jpg", width = 2, height = 2, dpi =
↪  72)
```

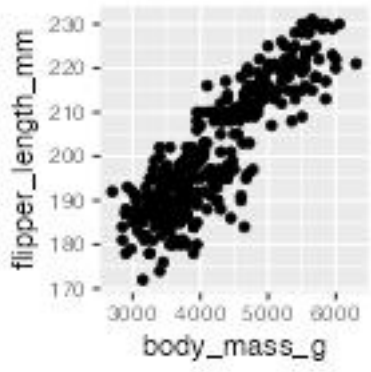
```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```

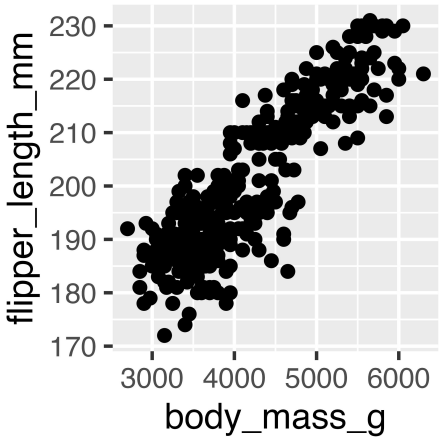
```
ggsave(filename = "1200.jpg", width = 2, height = 2, dpi
↪  = 1200)
```

```
Warning: Removed 2 rows containing missing values or
values
outside the scale range (`geom_point()`).
```

Le premier graphique sera pixellisé, et le deuxième bien net :

La résolution choisie sera une question de compromis. Plus vous augmentez la valeur de DPI, plus votre graphique sera net, mais il sera aussi plus lourd. Si vous pensez imprimer votre graphique, on recommande en général de choisir une résolution de 300 DPI. Moins que ça et ça risque de paraître pixellisé, mais plus que ça votre fichier risque d'être inutilement lourd.


flipper_length_mm
230
220
210
200
190
180
170
3000
4000
5000
6000
body_mass_g


flipper_length_mm
230
220
210
200
190
180
170
3000
4000
5000
6000
body_mass_g

9. Transformer les données

9.1. Introduction

Transformer les données? Non mais Charles, c'est pas comme tricher un peu de transformer nos données? Est-ce qu'on a vraiment le droit de faire ça?

La première chose à savoir pour répondre à cette question est que nos systèmes de mesures sont arbitraires. Ce sont des constructions de notre esprit. Certaines sont à l'échelle directement des données (cm, kg, etc.), mais d'autres sont souvent à des échelles transformées. Entre autres, le pH est défini comme le logarithme négatif de la concentration des ions d'hydrogène (-log[H^+]). Cette échelle est utilisée quotidiennement, partout dans le monde, bien qu'elle soit une mesure transformée.

L'exemple le plus probant que nos systèmes de mesure sont arbitraires est sans doute la température, où les degrés Fahrenheit peuvent être convertis en degrés Celsius par la transformation (F°-32)*5/9=C°. Clairement, il semble y avoir des cas où il est raisonnable de transformer nos données!

L'important pour qu'une transformation soit utilisable dans un contexte statistique est que la transformation soit **monotone** (*monotonic*). C'est-à-dire que l'ordre des observations soit conservé après la transformation. La plus petite donnée doit demeurer la plus petite, la plus grande demeurer la plus grande, etc. Pour autant que cette propriété soit respectée, vous pouvez faire à peu près ce que vous voulez avec les données, sans que cela change les conclusions qualitatives de vos analyses.

Lorsque l'on parle de **conclusion qualitative**, on parle généralement du sens de la relation (A est relié positivement à B, etc) et de son importance (A a un petit effet sur B, C a un grand effet sur B etc.). Évidemment, la conclusion quantitative, le chiffre lui-même définissant notre effet, lui sera différent après la transformation.

J'en profite ici pour vous mentionner que vous pouvez appliquer une transformation différente sur chacune des variables de votre tableau de données. Il n'est pas nécessaire que chaque variable subisse la même transformation. Il est même possible d'avoir certaines variables transformées et d'autres non, cela n'a pas d'importance.

9.2. Pourquoi transformer

La raison majeure pour laquelle nous utilisons des transformations en statistiques est pour forcer des données dans un modèle lorsque ces dernières ne respectent pas les assomptions de l'outil statistique. On peut, par une transformation, normaliser des données qui n'étaient pas normales ou linéariser une relation qui n'était pas linéaire. On peut aussi utiliser une transformation pour diminuer l'effet d'une valeur aberrante sur une analyse, en la rapprochant du nuage de points principal.

9.3. Labo : La transformation logarithmique

La transformation que vous utiliserez sans doute le plus souvent est la transformation logarithmique. On entend par une transformation logarithmique que chacune des observations d'une variable soit remplacée par le logarithme de cette observation. Le base du logarithme n'a pas vraiment d'importance d'un point de vue statistique. Vous pouvez donc utiliser un $\log_{10}$, $\log_e$, $\log_2$, comme vous voulez.

À titre de rappel, le logarithme est la réponse à la question : combien de fois doit-on multiplier la base du logarithme par elle-même pour arriver au nombre attendu. Par exemple, $\log_2(8) = 3$. On doit faire 2 x 2 x 2 pour arriver à 8. De la même façon, $\log_{10}(100)=2$, car 10 x 10 = 100, etc.

Pour ce chapitre, nous travaillerons avec le tableau de données `msleep` fourni avec la librairie ggplot2 plutôt que nos manchots habituels, puisque le tableau `msleep` contient beaucoup de variables qui vaudront la peine d'être transformées. Dans ce tableau, chaque ligne représente une espèce. Nous avons plusieurs colonnes contenant des information sur le temps de sommeil des différentes espèces, mais aussi certaines informations sur des mesures physiologiques, soit le poids du corps (`bodywt`) et le poids du cerveau (`brainwt`)

On peut par exemple se créer une colonne du poids du cerveau et du corps de chaque animal en log base e, comme ceci :

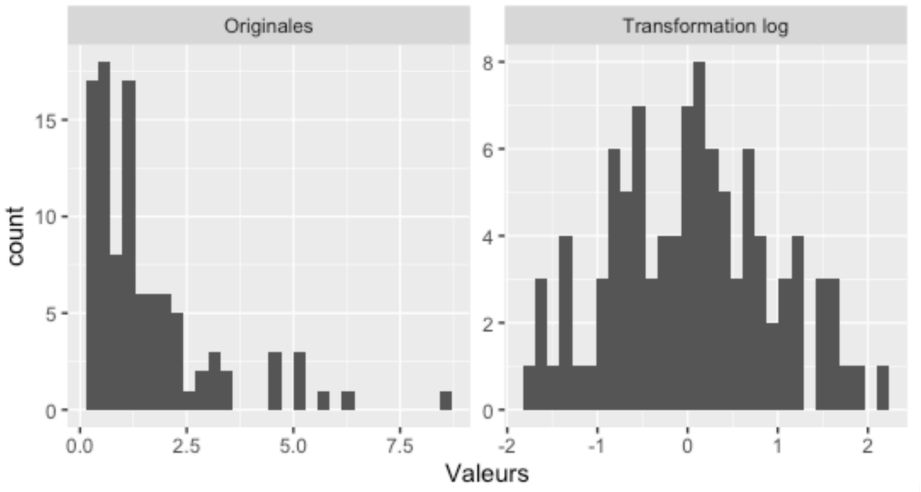
```
library(tidyverse)
msleep |>
  mutate(
    log_bodywt = log(bodywt),
    log_brainwt = log(brainwt)
  )
```

Cependant, avant d'appliquer une transformation, il importe de bien comprendre les maths derrière cette dernière. Par exemple, dans le cas de la transformation log, le log des valeurs <= 0 n'est pas défini. Il est donc important de vérifier que toutes nos valeurs sont > 0 avant d'appliquer cette transformation. Si la condition n'est pas respectée, il est possible de les déplacer légèrement en ajoutant une constante avant la transformation, par exemple comme ceci :

```
msleep |>
  mutate(
    log_bodywt = log(bodywt+1),
    log_brainwt = log(brainwt+1)
  )
```

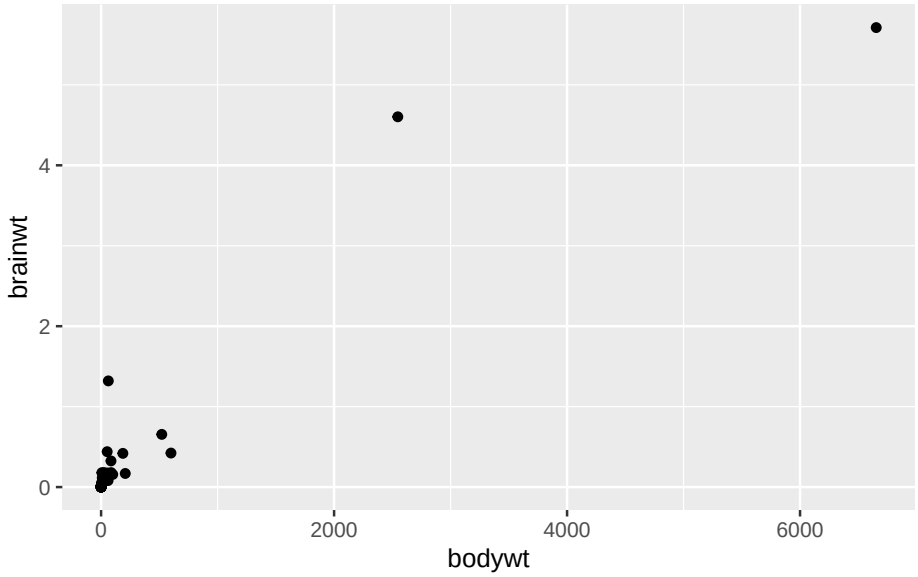
Alors, que fait la transformation log à nos données? Ce qu'il est important de retenir est que cette transformation étire les petites valeurs et compresse les grandes. Si l'on prend une série de valeurs également espacées, voici à quoi elles ressembleront après une transformation log :

Une fois que cette propriété est bien comprise, il devient intuitif de comprendre que cette transformation pourra normaliser (i.e. donner une forme normale) une distribution qui présentait à l'origine une longue queue à droite, comme ceci :

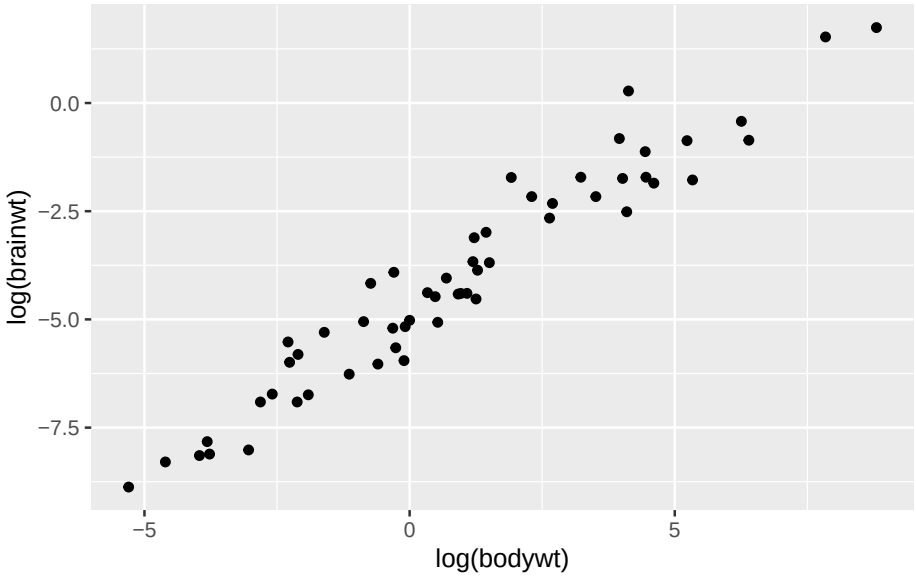


Remarquez que cette transformation log aura aussi pour effet de rapprocher les valeurs que l'on aurait pu considérer comme aberrantes dans les données originales, pour autant que celles-ci étaient à la droite de la distribution.

Dans le même ordre d'idées, la transformation log peut aussi permettre de linéariser une relation, qui autrement aurait été de nature exponentielle. Comparez par exemple les courbes générées par la relation entre brainwt et bodywt dans le tableau de données msleep avec les données originales :



Puis avec les deux variables transformées en log :



Encore une fois, la clé pour comprendre cet effet est de réaliser que la transformation étire les petites données, et compresse les grandes.

9.4. Labo : La transformation racine carrée

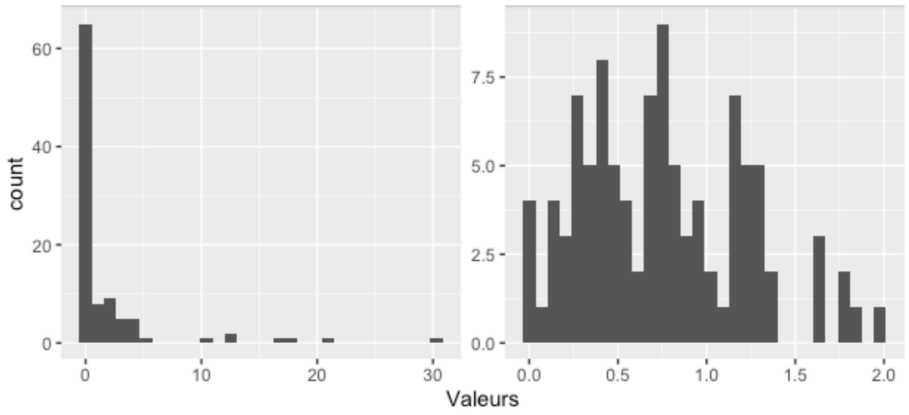
La transformation racine carrée a des effets très semblables à la transformation logarithmique, mais son effet est moins prononcé. Donc, dans un scénario où la distribution de votre variable présentait une longue queue à droite, mais que la transformation log, plutôt que normaliser a créé une longue queue à gauche (i.e. elle sur-corrige), la transformation racine carrée sera peut-être appropriée. Rappelez-vous, cependant, qu'avant d'appliquer une transformation racine carrée, il importe de bien inspecter vos données, puisqu'une racine carrée n'est pas définie pour les nombres négatifs...

Dans R, la transformation racine carrée (`sqrt`; *SQuare RooT*) s'applique comme ceci :

```
msleep |>
  mutate(
    sqrt_bodywt = sqrt(bodywt)
  )
```

Si jamais vos données présentent une longue queue à droite, mais que ni la transformation racine carrée ni la transformation log ne présentent le résultat escompté, rappelez-vous qu'une racine carrée est équivalente à appliquer l'exposant ½ à vos données.

Sachant cela, il est possible d'appliquer en fait n'importe quel exposant fractionnaire à nos données, pour obtenir exactement la transformation voulue. Voyez par exemple ces données présentant une queue très prononcée à droite, qui peuvent être normalisées à l'aide de l'exposant ⅕ (0,2) :



Soyez cependant bien attentif lorsque vous appliquez un exposant fractionnaire à vos données dans R. L'opérateur d'exposant ayant préséance

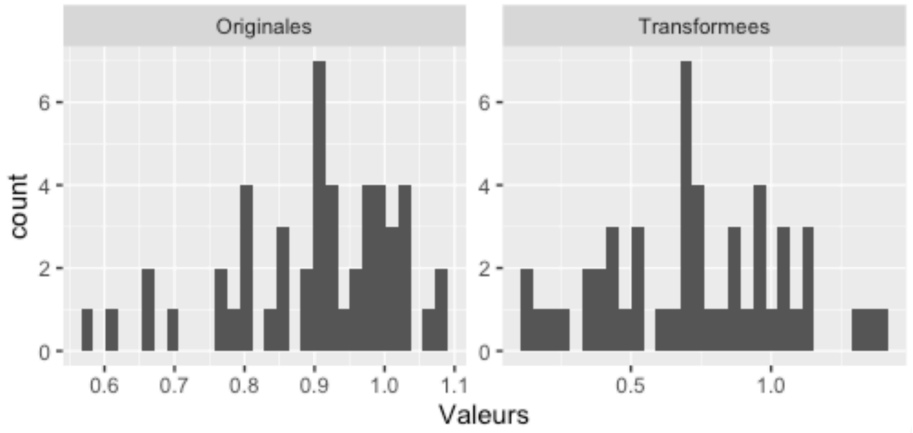
sur celui de division dans l'ordre des opérations, la façon correcte d'appliquer un exposant fractionnaire est à l'aide de parenthèses, comme ceci :

```
msleep |>
  mutate(
    trans_bodywt = bodywt^(1/4)
  )
```

Rappelez-vous aussi en faisant vos transformations que la forme de votre distribution finale ne sera jamais parfaitement normale, ce qui n'est pas nécessairement grave car la plupart des analyses sont robustes à la non-normalité.

9.5. Si la longue queue est à gauche

Toutes les transformations vues jusqu'à présent permettent de corriger des problèmes avec nos données lorsque celles-ci sont asymétriques avec une longue queue à droite. Si la longue queue est à gauche, on peut utiliser le même principe des exposants, mais en utilisant des valeurs supérieures à un. Par exemple, voici l'effet de l'exposant 4 à des données présentant une longue queue à gauche :



9.6. Exercice : Les transformations

Dans le tableau de données msleep, affichez l'histogramme de fréquences de la variable `sleep_rem`, puis tentez de déterminer quelle serait la meilleure transformation pour normaliser cette variable pour une analyse.

Dans le même tableau de données, visualisez la forme de la relation entre les variables `sleep_rem` et (en X) et `sleep_total` (en Y). Vous pouvez ajouter une courbe de lissage (`geom_smooth`) pour mieux voir la forme de la relation. Quelle serait la meilleure transformation à appliquer aux variables pour linéariser cette relation?

NB : ne désespérez pas si vous ne réussissez pas à bien transformer vos variables du premier coup. Il s'agit d'un processus essai-erreur, qui s'accélérera à mesure que vous pratiquerez.

partie II.

Les statistiques

10. Grands principes

Pour pouvoir passer aux prochaines étapes dans notre apprentissage des biostatistiques, nous devrons mettre en place une série de concepts sur lesquels nous pourrons construire. J'ai rassemblé toute cette information ici sous forme de grands principes statistiques. Vous allez probablement vous sentir balayés par une tempête de terminologie et j'en suis désolé. J'ai essayé le plus possible de m'en tenir au strict nécessaire.

10.1. Questions et hypothèses

La chose la plus importante à considérer avant de se lancer dans un projet de statistiques est de se demander : quelle est ma question écologique? Qu'est-ce que je voudrais savoir? La **question écologique** (ou biologique, ou médicale, etc.), au cœur de votre démarche, doit posséder plusieurs caractéristiques. Elle doit, entre autres, être :

- pertinente (bien connectée aux connaissances actuelles),
- claire (sinon les ambiguïtés vont vous rattraper plus tard) et
- réaliste (on ne peut pas tout faire dans un seul projet).

Voici quelques exemples de questions de recherche :

- Comment le paysage adjacent influence-t-il la biodiversité dans les parcs urbains?
- Quels facteurs affectent l'abondance du tangara écarlate?
- Quels effets a la compétition sur le ventre rouge du Nord?

Une fois cette question établie, on peut construire, en se basant sur les connaissances actuelles, une ou des **hypothèses de travail**. Contrairement à la question écologique, l'hypothèse de travail prend position, elle dit : voici comment (au meilleur de ma connaissance) les choses devraient fonctionner.

Voici quelques exemples d'hypothèses de travail :

- La biodiversité des parcs urbains diminue avec la densité de routes
- La présence du tangara écarlate est reliée à la présence de forêts matures.
- Le risque de prédation par l'omble de fontaine entraîne une diminution de la croissance du ventre rouge du nord.

Ces hypothèses de travail peuvent ensuite être traduites en hypothèses statistiques, que l'on pourra tester à l'aide d'expériences. Nous reviendrons spécifiquement sur les hypothèses statistiques au Chapitre 12.

10.2. Populations et échantillons

Une grande partie de la complexité des biostatistiques tient au fait que nous mesurons rarement l'ensemble des individus de notre population d'intérêt. Par exemple, si on s'intéresse aux effets de la présence de forêt mature sur le tangara écarlate, nous ne pourrons pas regarder le comportement de tous les tangaras écarlates en Amérique du nord. De la même façon, notre étude sur la biodiversité des parcs urbains n'étudiera probablement pas l'ensemble des parcs urbains sur la planète.

La **population**, au sens statistique du terme, est l'ensemble des individus (ou des écosystèmes, ou des communautés, etc) auxquels notre question ou notre hypothèse de travail s'intéresse. Elle peut être très spécifique ou très vaste, selon l'ampleur de notre question. Si notre question est de savoir comment réagissent les invertébrés du Lac à la Tortue aux apports

de phosphore, notre population à l'étude sera les invertébrés de ce lac seulement. Si notre question est plus générale : comment réagissent les invertébrés aux apports de phosphore, alors notre population d'intérêt est l'ensemble des invertébrés de la planète!

Vous voyez maintenant pourquoi il est primordial de construire une question réaliste, car cela nous permettra de cerner la population à laquelle nous devrons appliquer notre étude.

Si nous avions les capacités techniques et budgétaires d'étudier tous les individus d'une population, les choses seraient très simples en statistiques. Par exemple, si vous travaillez au MAPAQ et recevez un rapport contenant la productivité de chacun des apiculteurs du Québec, vous avez accès à toute la population d'apiculteurs. Lorsque l'on a accès aux données de toute la population, on parle alors de **recensement**.

Sinon, le sous-ensemble d'individus auxquels vous avez accès pour votre étude devient votre **échantillon**. Il peut être très petit dans certains cas (p. ex. 10 canards capturés dans un marais), ou immense dans d'autres (des millions de feuillets d'oiseaux dans la base de données eBird). Le nombre d'observations ou d'individus dans votre échantillon se nomme la **taille de l'échantillon**, le fameux n. On entend ici par **observation** une ligne dans votre tableau de données, ce qui correspond en général à un individu ou un site ou un écosystème, etc. dépendamment de ce que vous mesurez.

10.3. Conventions

Il existe une convention établie lorsque l'on discute de statistique. Les propriétés numériques (e.g. moyenne, variance, etc.) d'une population sont habituellement nommées **paramètres**, et désignées par une lettre grecque (σ pour l'écart type, μ pour la moyenne, etc.) alors que celles

décrivant des échantillons sont habituellement nommées **statistiques**, et désignées par une lettre romaine (p. ex. s pour l'écart-type).

Je ne suis pas quelqu'un de particulièrement sensible à cette convention, et il est donc possible que j'en déroge à l'occasion. Les esprits plus stricts que le mien risquent par contre de vous demander d'y faire beaucoup plus attention.

10.4. La loi des grands nombres

Pourquoi faire tout un plat de la taille de l'échantillon? À cause de la loi des grands nombres. La loi des grands nombres peut être définie dans un langage simple comme ceci : plus la taille d'un échantillon est grande, plus tout ce que l'on mesure sur cet échantillon (moyenne, variance, corrélation, etc.) va ressembler à la vraie valeur de la population.

Prenons comme illustration le classique : lancer une pièce de monnaie pour jouer à pile ou face. Si notre pièce de monnaie n'est pas truquée, la chance d'obtenir pile est de 50 %, et celle d'obtenir face est aussi de 50 %. Vous êtes d'accord avec moi qu'en lançant dix fois la pièce de monnaie, il peut vous arriver d'avoir exactement 50 % de pile, mais il peut aussi vous arriver d'obtenir 20 % , 40 %, etc. Ce que la loi des grands nombres nous dit, c'est que, plus on lancera notre pièce de monnaie un grand nombre de fois, plus les pourcentages devraient ressembler aux chances réelles de la pièce, soit 50 % de pile. Inversement, plus notre échantillon est petit, plus il pourrait dévier de la vraie valeur de la population.

Cette loi s'applique à tout ce qui peut être échantillonné. Si vous pêchez des truites dans un lac et que le poids moyen de la population de truites est de 4 kg, la moyenne d'un petit échantillon de 5 truites pourrait être facilement de 2 kg ou de 6 kg. Par contre, si vous en pêchez une centaine, leur poids moyen devrait être très très proche de 4 kg.

Notez que dans la réalité, nous ne savons pas à l'avance quelle est la vraie valeur de la population, mais il est tout de même utile de savoir que plus notre échantillon compte d'observations, plus la valeur calculée sur notre échantillon se rapprochera de la vraie valeur de la population.

10.5. Contenu optionnel : l'importance de la question et de la loi des grands nombres

Voici une petite anecdote pour illustrer l'importance de bien réfléchir à la loi des grands nombres et de bien penser notre question avant de commencer à travailler. Cet exemple provient de l'excellent livre de Daniel Kahneman, Thinking Fast and Slow, qui parle du fonctionnement du cerveau humain et des pièges dans lesquels il nous fait parfois tomber.

Afin d'améliorer le système scolaire américain, des experts ont voulu poser la question : quelles sont les caractéristiques des écoles où les élèves réussissent le mieux. Leur plan était de copier le modèle de ces écoles pour créer d'autres succès. Il est rapidement apparu que parmi les meilleures écoles, beaucoup étaient de très petite taille. Beaucoup d'argent a donc été dépensé suite à cette réflexion afin de réduire la taille des écoles pour favoriser la réussite scolaire.

Vous vous en doutez peut-être à ce point, mais la chose que les experts avaient négligée était, évidemment, la loi des grands nombres. Bien sûr, il y avait beaucoup d'écoles de petite taille où les élèves réussissaient mieux, mais il y avait aussi beaucoup d'écoles de petite taille où les élèves réussissaient très mal. Leur déviation de la moyenne était dûe, en grande partie au fait qu'elles représentaient de très petits échantillons.

Un simple retournement de la question, p. ex. en se demandant quels facteurs influencent la réussite scolaire et en observant le rendement de

l'ensemble des écoles auraient évité à la fondation Bill et Melinda Gates de gaspiller beaucoup d'argent[1].

10.6. Caractéristiques d'un bon échantillon

Tous les échantillons ne sont pas égaux. Certaines caractéristiques peuvent faire d'eux de meilleurs, ou de moins bons échantillons. On peut résumer les qualités d'un échantillon à deux choses : sa taille et sa représentativité. Il est facile, une fois que l'on connaît la loi des grands nombres de comprendre l'importance de la taille de l'échantillon, mais qu'en est-il de la représentativité?

La **représentativité** d'un échantillon nous indique qu'il ressemble à la population, qu'il présente les mêmes caractéristiques que cette dernière. Si par exemple notre population compte 30 % de mâles et 70 % de femelles, un échantillon sera représentatif s'il contient environ les mêmes proportions de mâles et femelles. Si notre échantillon contient 10 mâles et 2 femelles, il sera beaucoup moins représentatif de notre population, ce qui pourrait nous amener vers des conclusions erronées par la suite. La représentativité ne se limite pas seulement aux individus. Si l'on pêche tous nos individus dans une seule fosse d'un lac, notre échantillon ne sera peut-être pas représentatif de l'ensemble du lac, qui comprend aussi des zones peu profondes. Lorsque notre échantillon n'est pas représentatif, on dit qu'il est **biaisé**.

Il existe (au moins) deux stratégies pour s'assurer qu'un échantillon est représentatif. La première est d'effectuer un échantillonnage aléatoire. L'**échantillonnage aléatoire** consiste à puiser nos échantillons le plus au hasard possible. On peut par exemple établir une grille sur la surface de notre lac ou de notre forêt et choisir nos emplacements au hasard en

[1] https://marginalrevolution.com/marginalrevolution/2010/09/the-small-schools-myth.html

utilisant un générateur de nombres aléatoires[2] ou en lançant un dé. Plutôt qu'une grille, on peut aussi utiliser un système qui nous indique une direction et un nombre de pas au hasard entre chaque emplacement.

La deuxième stratégie pour former un échantillon représentatif est l'**échantillonnage stratifié**. Ce dernier consiste à construire notre échantillon en fonction de ce que l'on connaît de la population. P. ex. si l'on sait que notre forêt est composée à 20 % de conifères et 80 % de feuillus, nous choisirions nos emplacements pour avoir p. ex. 4 emplacements dans des zones de conifères et 16 dans des peuplements de feuillus. Les échantillons doivent tout de même être pris le plus au hasard possible à l'intérieur de ce système de stratification.

Les deux techniques présentent des avantages et des inconvénients. L'échantillonnage aléatoire peut être extrêmement efficace, en particulier pour éliminer **les effets confondants**, c'est-à-dire des variables autres que celles qui nous intéressent qui pourraient influencer notre étude. Mais pour se faire, nous devons avoir beaucoup d'échantillons. Si nous en avons peu, il est possible que la composante aléatoire nous ait fourni un échantillon biaisé… souvent malheureusement à notre insu. L'échantillonnage stratifié n'est pas parfait non plus, car il peut nous attirer sur de fausses pistes si notre connaissance de la population était erronée, ou si elle nous force à prendre des échantillons qui ne sont pas entièrement indépendants pour respecter notre plan de stratification.

Pour qu'un échantillon soit considéré comme représentatif, il faut aussi que les observations soient indépendantes les unes des autres, afin d'éviter la pseudo-réplication. La **pseudo-réplication**, comme son nom le suggère, consiste à avoir l'impression que nous avons échantillonné des individus indépendants, alors que dans les faits ils ne l'étaient pas. Cette pseudo-réplication peut être parfois très directe (mesurer le même individu plusieurs fois) ou indirecte (p. ex. mesurer deux arbres voisins, qui sont en fait exposés exactement aux mêmes facteurs confondants et compétitionnent un contre l'autre). On peut souvent détecter la

[2] https://www.random.org/

pseudo-réplication en regardant une carte de nos emplacements, et s'apercevoir que certains sont trop près les uns des autres.

La pseudo-réplication devrait être prise très au sérieux, car elle est très complexe à régler après coup. Puisqu'elle gonfle artificiellement notre confiance en nos résultats, elle doit être absolument évitée ou gérée correctement. Il existe des techniques statistiques pour gérer les données non indépendantes, que nous verrons plus en détails au Chapitre 16 et au Chapitre 30.

10.7. L'inférence statistique

Comme nous venons de le voir, en tant que scientifiques, nous n'avons la plupart du temps accès qu'à des échantillons plutôt qu'à la population entière pour effectuer nos analyses. Pourtant, les conclusions que nous devrons apporter ciblent la population entière. Cet acte, de produire des conclusions sur une population à partir d'un échantillon se nomme l'**inférence statistique**. En statistique, cela consiste le plus souvent de façon concrète à évaluer un paramètre d'une population à partir de la statistique mesurée sur un échantillon.

Pour le reste de ce livre, nous nous concentrerons à étudier des méthodes statistiques qui nous permettront d'effectuer ce passage. Vous comprendrez sans doute, après avoir pris connaissance des premières sections du présent chapitre, que nous nagerons toujours dans une certaine incertitude quant à la validité de ce passage. Notre travail sera donc de savoir comment évaluer cette incertitude face à nos conclusions, mais aussi de savoir bien la communiquer aux décideurs ou praticiens qui utiliserons nos résultats.

10.8. Erreurs et puissance

Lorsque nous tenterons d'évaluer un paramètre à partir d'une méthode d'inférence statistique, nous ferons toujours face à un certain risque d'erreur, une probabilité de nous tromper dans nos conclusions. Si p. ex. nous déterminons à l'aide d'échantillons que les mâles bruants chanteurs sont plus grands que les femelles, il se peut que cette conclusion soit fausse. Ce type d'erreur, c'est-à-dire trouver quelque chose lorsqu'il n'y avait rien à trouver, se nomme **erreur de type I**. À l'inverse, il aurait aussi pu arriver que l'on ne trouve pas de différence entre les mâles et les femelles, alors qu'en réalité dans la population, il existait une différence. On parlerait alors d'**erreur de type II**.

Tout dépendant du problème auquel on s'attaque, il peut arriver qu'un type d'erreur soit plus important à surveiller que l'autre.

Si p. ex. vous êtes un juge et que vous ne voulez surtout pas envoyer en prison un innocent qui n'a pas commis de crime. Vous seriez alors très sensible et attentif à l'erreur de type I.

Si vous êtes plutôt un médecin et qu'il vous faut tester si une femme est enceinte avant d'administrer un médicament qui pourrait nuire à sa grossesse, l'erreur clé à surveiller sera l'erreur de type II. On ne veut surtout pas administrer accidentellement un médicament à une femme parce que son test n'a pas réussi à détecter sa grossesse.

Lorsque l'on commence à réfléchir aux erreurs de type II, il est intéressant de définir le concept de puissance statistique. La **puissance statistique** se définit comme la probabilité de trouver un effet, dans les cas où il y en a vraiment un. Les calculs sont relativement complexes et dépassent largement le cadre de ce cours de biostatistiques. Ce qu'il est surtout important de savoir est que la puissance statistique augmente avec la taille de l'échantillon, et aussi avec la taille de l'effet que l'on tente de trouver (i.e. plus l'effet est grand, plus il sera facile à détecter). Au contraire, la puissance diminuera avec la variabilité de l'échantillon.

10.9. Les degrés de liberté

Un autre terme statistique que vous verrez souvent revenir dans les prochains chapitres est celui de degrés de liberté. Les **degrés de liberté** définissent le nombre d'observations qui peuvent toujours varier lorsque l'on a fixé la valeur de certains paramètres. Cette définition, plutôt abstraite, s'explique beaucoup plus facilement à l'aide d'un exemple.

Supposons que nous avons une paire de nombres à propos desquels on ne sait rien. Puisque rien n'est fixé par rapport à ces nombres, nous avons deux degrés de liberté (d.d.l.) Par contre, si nous savons que la somme de nos nombres est 5, il ne nous reste plus qu'un seul d.d.l.. Car du moment que nous connaissons un des nombre, p. ex. 2, l'autre n'est plus libre de varier, on sait automatiquement qu'il serait 3, pour que la somme arrive à 5.

En général, chaque paramètre connu à propos de nos données diminue d'autant les d.d.l. Si nous avons 10 observations et que nous connaissons la moyenne et la variance, il ne nous reste que 8 d.d.l.

Comme nous allons le voir dans les chapitres suivants, les degrés de liberté sont particulièrement importants en statistiques, puisque si nous n'avons pas suffisamment de d.d.l. (i.e. d'observations), cela peut limiter le nombre de paramètres que l'on pourra estimer. On pourrait même en arriver au point où notre test ne peut juste pas s'exécuter. Encore là, plus notre échantillon sera grand, moins ce sera risqué d'arriver!

11. Les lois de probabilité

11.1. L'utilité des lois de probabilité

La première chose qui vous viendra probablement à l'esprit est : c'est quoi une loi de probabilité et à quoi ça peut bien servir? Une loi de probabilité est un outil statistique, une représentation de la réalité. Nous avons vu au Chapitre 3 que l'on peut visualiser la variabilité d'une variable à l'aide d'un histogramme de fréquences. Les lois de probabilité permettent de décrire certaines formes classiques d'histogrammes. Pour ne pas avoir à dire : beaucoup d'observations au milieu, pas beaucoup dans les bouts, les côtés symétriques avec un seul mode. Il peut être avantageux d'avoir un terme qui définit une telle forme, p. ex. ici la distribution normale.

De même, puisqu'une loi de probabilité est définie par des fonctions mathématiques (les fonctions de densité et de répartition), il est possible de l'utiliser pour effectuer des calculs de probabilité et de faire des inférences à partir de certains de ses paramètres.

D'un point de vue sémantique, les lois de probabilités définissent des distributions de fréquences. La loi normale définit la distribution normale, la loi de Poisson définit la distribution de Poisson, etc. On peut donc souvent utiliser un terme ou l'autre (loi ou distribution), puisqu'ils font référence au même concept.

11.2. Structure de la loi normale

La loi de probabilité la plus utilisée en biologie est sans aucun doute la loi normale. Cette dernière suit la forme classique d'une courbe en cloche (*bell curve*), parfois nommée aussi la loi de Gauss.

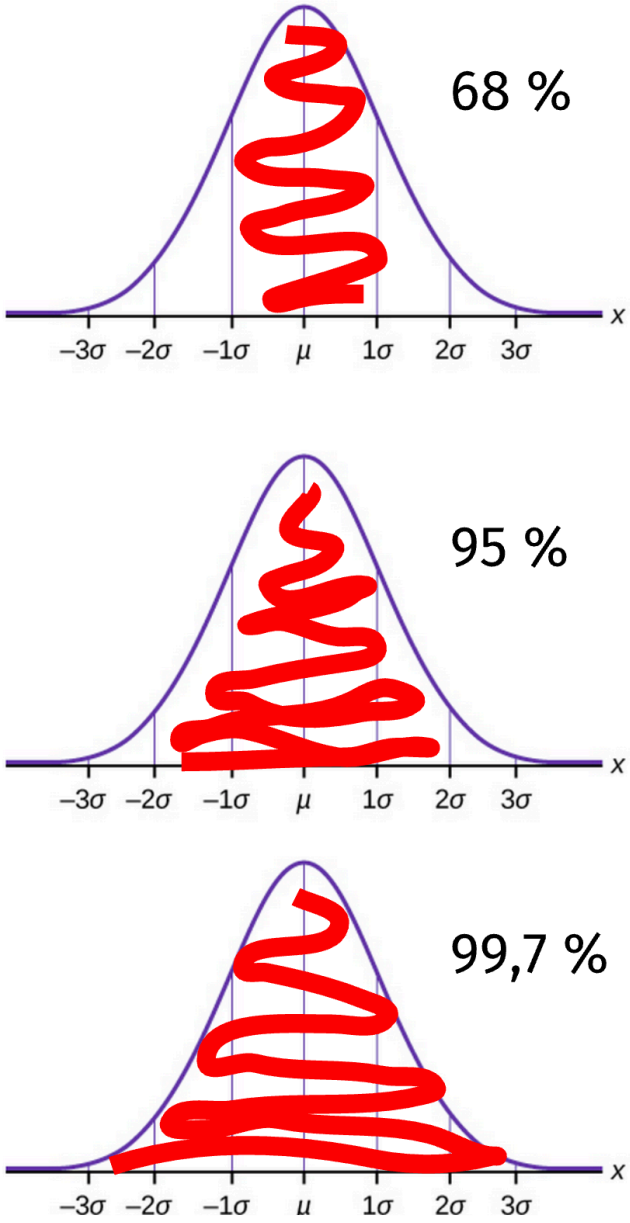


Figure 11.1.: La loi normale, illustrant sa structure et la règle du 68-95-99.7. Le symbole μ (mu) est utilisé ici pour illustrer la moyenne de la population et σ (sigma) un écart-type.

Comme suggéré précédemment, la loi normale se caractérise principalement par la présence d'un seul mode, et d'une distribution symétrique des probabilités de chaque côté. Elle est définie par deux paramètres, soit sa moyenne et sa variance (ou son écart type, dépendant des définitions).

Elle survient fréquemment dans la nature pour décrire la variabilité des phénomènes naturels comme la taille et le poids de la plupart des espèces, la pression artérielle, le Q.I., etc. Lorsqu'un phénomène naturel peut se définir comme la somme de plusieurs événements aléatoires, la forme finale de la distribution risque fort de correspondre à la loi normale. La taille et le poids sont, entre autres, dans cette catégorie car la taille d'un animal adulte dépend généralement de l'activation ou non d'une série de gènes définissant sa croissance.

Il est important de comprendre que ce ne sont pas tous les phénomènes naturels qui produiront des distributions normales. Certains autres, p. ex. les tremblements de terre ou les décès liés à la guerre auront plutôt tendance à présenter une longue queue d'un côté, parce que certains événements sont beaucoup plus rares alors que d'autres sont très communs. Il survient des dizaines de milliers de tremblements de terre de faible magnitude par année et seulement quelques-uns sont plus importants.

Dans la distribution normale, les probabilités de chaque côté de la moyenne sont égales. Si la moyenne du poids des bruants suit une loi normale avec une moyenne de 12 g, il est aussi probable de trouver un bruant de 10 g qu'un bruant de 14 g.

11.3. La règle du 68-95-99,7

Dans la loi normale, la densité de distribution de probabilités autour de la moyenne est toujours organisée de la même façon. Autour de la

moyenne, nous trouverons 68 % des observations à l'intérieur d'un écart-type, 95 % des observations à l'intérieur de deux écarts-types et 99,7 % à l'intérieur de trois écarts-types de la moyenne. On nomme ce phénomène la **loi du 68-95-99,7**.

Connaissant la moyenne et l'écart-type d'un phénomène répondant à la loi normale, il est facile d'évaluer rapidement certaines probabilités en se basant sur cette règle. Si l'on retourne à nos bruants pesant en moyenne 12 g et que l'on sait que l'écart-type de cette distribution est de 1 g, on peut affirmer que 95 % des bruants que nous observerons pèseront entre 10 et 14 g. On peut aussi utiliser cette règle dans l'autre sens. Pour la même population de bruants, si l'on trouve un individu pesant 15 g, on peut affirmer qu'il s'agit d'un phénomène rare, puisque seulement 0,3 % (100 % - 99,7 %) des bruants auront un poids aussi extrême.

Notez qu'il s'agit d'une règle pour faire des calculs "au pif", rapidement sans avoir recours à un ordinateur. Les valeurs calculées seront approximatives, mais suffisantes pour valider un calcul ou avoir une idée de grandeur d'un chiffre en lisant un rapport, par exemple.

11.4. Labo : Calculer des probabilités basées sur la loi normale

Dans le logiciel R, il est possible d'effectuer rapidement des calculs basés sur les probabilités de la distribution normale. Il existe deux fonctions différentes pour le faire, une lorsque l'on recherche la probabilité associée à une valeur (`pnorm`; *Probability NORMal*) et une autre lorsque l'on cherche la valeur associée à une probabilité (`qnorm`; *Quantile NORMal*).

P ex. pour déterminer la probabilité d'observer un bruant de 14 g dans une population dont la moyenne est 12 et l'écart-type est 1, nous utiliserons le code suivant :

```
pnorm(14, mean = 12, sd = 1)
```

```
[1] 0.9772499
```

R nous retourne la valeur 0,977. En se basant sur la loi du 68-95-99,7 expliquée ci-haut, on se serait attendu à obtenir 0,95. Que s'est-il passé?

Ce qu'il faut comprendre est que la fonction `pnorm` calcule les probabilités de façon cumulative à partir de zéro, de gauche à droite. Elle nous informe que notre bruant de 14 g est plus grand que 97,7 % des bruants. Il y a un autre 2,3 % de bruant qui se trouvent à être deux écarts-types plus petits que la moyenne. On peut le voir en faisant :

```
pnorm(10, mean = 12, sd = 1)
```

```
[1] 0.02275013
```

R nous retourne 0,023. Il y a donc 2,3 % de bruants à deux écarts-types sous la moyenne et un autre 2,3 % au-dessus de la moyenne. Il y a donc, effectivement 95,4 % des bruants à l'intérieur de 2 écarts-types de la moyenne.

Notez que nous arrivons à 95,4 %, le vraie proportion de données à deux écarts-types de la moyenne, plutôt que 95 %, qui est un chiffre facile à retenir pour faire des choses "au pif".

Le point important à comprendre ici est que, lorsque l'on discute de probabilités issues d'une distribution, il est important de spécifier si notre probabilité est bilatérale ou unilatérale. Une probabilité **unilatérale** est entièrement d'un côté de la distribution. On peut p. ex. dire que 95 % des bruants sont plus petits que 13,6 g. Au contraire, une probabilité **bilatérale** est partagée de façon égale de chaque côté. On peut p. ex. dire que 95 % des bruants ont un poids entre 10 et 14 g. On aurait alors séparé le 5 % également, 2,5 % de chaque côté de la distribution.

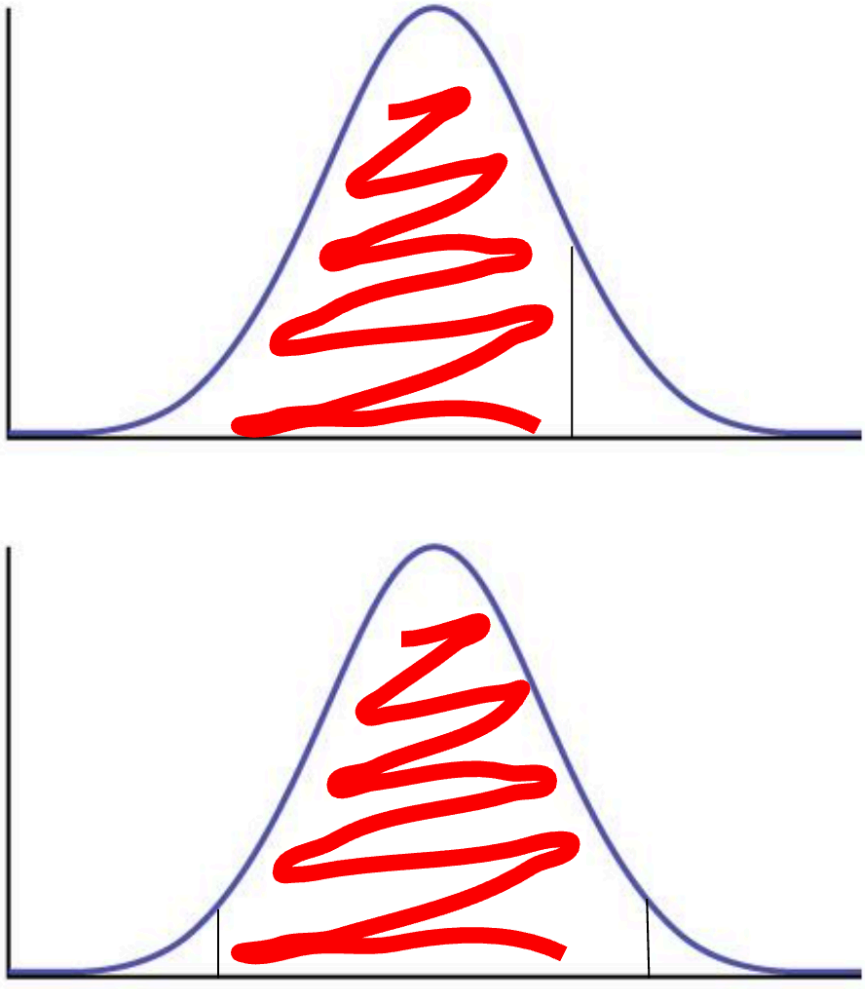


Figure 11.2.: Différence entre une probabilité de 95 % unilatérale (haut) et bilatérale (bas).

Comme mentionné précédemment, on peut utiliser la fonction `qnorm`

pour savoir à quelle valeur correspond une probabilité. Dans notre même distribution de bruants (moyenne 12, écart-type 1), si l'on veut savoir à quoi correspondrait un bruant plus grand que 99 % des autres individus, on peut utiliser le code suivant :

```
qnorm(0.99,mean = 12, sd = 1)
```

```
[1] 14.32635
```

Ce qui nous donne 14,3 g.

11.5. Le théorème central limite

Ce n'est pas un hasard si les phénomènes naturels pouvant être décrits par l'addition de la probabilité d'événements indépendants peuvent être décrits par une loi normale. Il existe un principe statistique sous-jacent à ce phénomène, nommé le théorème central limite. Le **théorème central limite** stipule que des échantillons formés par l'addition d'au moins une vingtaine d'observations aléatoires produiront ensemble une distribution normale, sous certaines conditions[1]. Ce qui est remarquable avec ce phénomène est qu'il est vrai peu importe la forme de la distribution dans laquelle les observations aléatoires sont pigées pour former les échantillons. Même pour un tirage à pile ou face ou un lancé de dé. Si chaque échantillon est formé par la somme de 20 lancers à pile ou face, la distribution de ces échantillons (i.e. si on recommençait nos 20 lancers plusieurs fois) formera toujours une distribution normale.

Notez que le théorème s'applique autant aux moyennes qu'aux additions, puisqu'il s'agit du même principe mathématique.

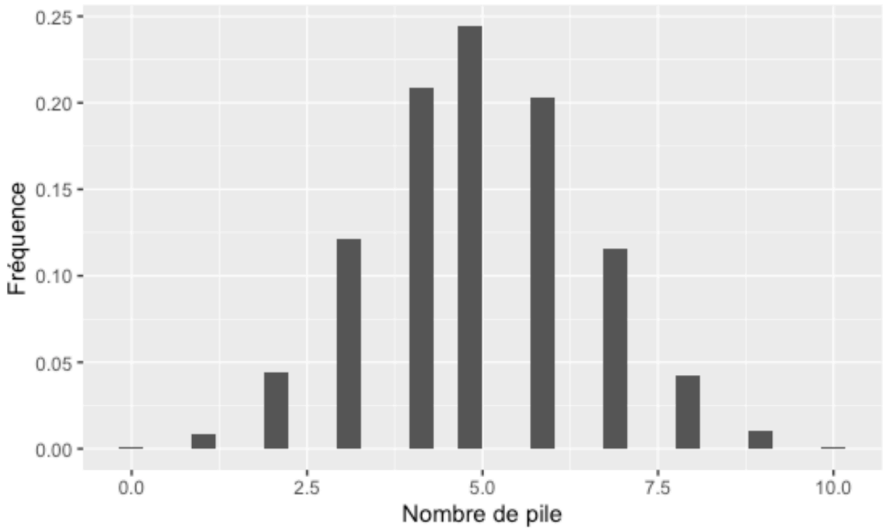
[1] Page, Scott E. The Model Thinker. 2018 p. 62

11.6. Les autres lois

Il existe, évidemment, une série d'autres lois de probabilités que la loi normale. Puisque la majorité des techniques statistiques enseignées dans ce cours exigent que les données suivent une loi normale, nous de verrons ici qu'un bref aperçu du reste des lois.

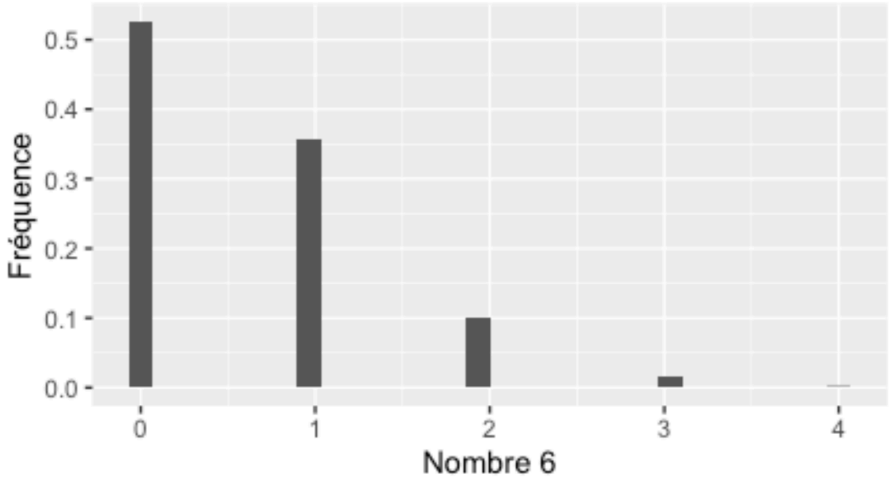
Une des lois statistiques les plus communes que l'on peut rencontrer dans la vie de tous les jours est sans doute la **loi binomiale**. Cette loi décrit le nombre de succès obtenus lorsque l'on répète plusieurs événements ayant la même probabilité de succès. Cette loi nécessite aussi deux paramètres, soit le nombre d'essais et la probabilité de succès d'un essai. Contrairement à la loi normale, la loi binomiale définit des données quantitatives discrètes (des dénombrements) plutôt que quantitatives continues.

La façon classique d'illustrer la loi binomiale est avec le lancer d'une pièce de monnaie. Elle permet, par exemple, de savoir quelles seront les probabilités d'obtenir exactement 5 piles, ou plus de 8 piles lorsque l'on lance 10 fois la pièce :



En écologie, cette distribution se rencontrera souvent lorsque l'on veut étudier le nombre de graines qui germineront, le nombre de poissons qui réussiront à franchir un obstacle, etc.

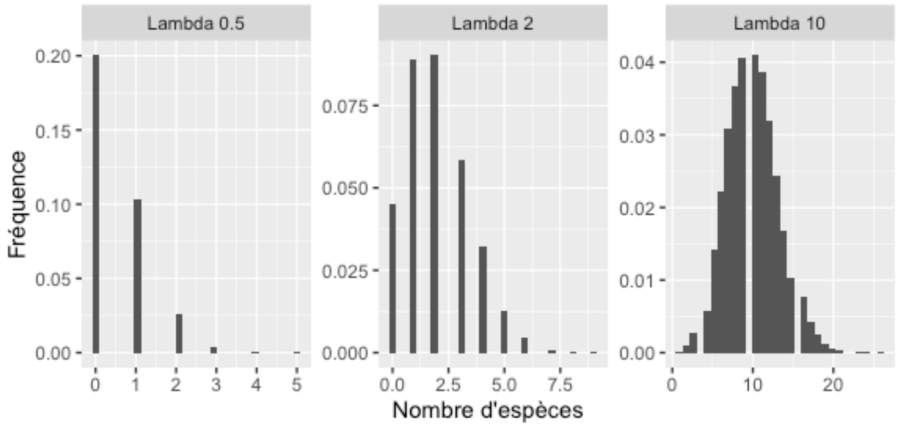
Contrairement à la loi normale, la loi binomiale ne sera pas toujours symétrique. Lorsque les probabilités de succès sont faibles, la distribution viendra s'appuyer à gauche sur la valeur zéro. P. ex. si l'on veut lancer un dé 5 fois, quel pourrait être le nombre de 6 que nous obtiendrons?



Enfin, la dernière loi dont nous discuterons est la **loi de Poisson**. Cette dernière décrit la probabilité qu'un certain nombre d'événements se produisent dans un intervalle de temps (ou d'espace) donné. Elle peut servir à décrire le nombre d'espèces qui seront rencontrées dans une parcelle, le nombre de brins d'herbe dans une pelouse, etc. Le truc pour bien la cerner est qu'elle est parfaite pour décrire le nombre de poissons pêchés dans une journée. Contrairement à la loi binomiale, on ne sait pas à l'avance combien d'essais seront effectués (i.e. on ne sait pas combien il y a de poissons dans le lac).

La propriété particulière de la loi de Poisson est qu'elle est définie par un seul paramètre, nommé λ (lambda), qui décrit à la fois la moyenne et la variance de la distribution (qui se contrôlent avec deux paramètres différents dans la loi normale).

Voici quelques exemples de distribution de la loi Poisson :



Remarquez que, comme pour la distribution binomiale, la distribution Poisson s'appuie à gauche sur la valeur 0 lorsque la valeur de lambda est faible. Au contraire, lorsque la valeur de lambda est élevée, elle est très semblable à la loi normale.

Cette propriété nous permet d'appliquer parfois des tests statistiques associés à la loi normale à ces deux distributions (Poisson et binomiale). Si leur moyenne est suffisamment élevée, la loi normale en est une excellente approximation, pour autant que l'on accepte que nos résultats puissent nous prédire des choses comme 2,5 espèces ou 3,4 œufs.

11.7. Labo : Calculer les probabilités des lois binomiales et Poisson

Les lois binomiale et Poisson sont aussi implémentées dans R, sous la même forme de deux fonctions que la loi normale. Les fonctions `ppois` et `pbinom` nous fournissent des probabilités associées à des valeurs et les fonctions `qpois` et `qbinom` nous fournissent des valeurs associées à des probabilités.

P. ex. si l'on veut connaître les chances d'obtenir 5 fois pile ou moins en lançant 10 pièces de monnaie

```
pbinom(5, size = 10, prob = 0.5)
```

```
[1] 0.6230469
```

La fonction `pbinom` attend trois arguments : le nombre de succès pour lequel on attend la probabilité, le nombre d'essais et la probabilité de succès de chacun des essais. On obtient ici 62 %.

Si on voulait savoir l'inverse, p. ex. combien de fois on obtiendrait le côté pile si on était vraiment malchanceux, disons quelque chose qui arriverait 5 % du temps ou moins :

```
qbinom(0.05, size = 10, prob = 0.5)
```

```
[1] 2
```

R nous répond que ce niveau de malchance correspondrait à uniquement 2 fois pile sur 10 lancers. Autrement dit, 95 % du temps, on en aurait obtenu plus.

Les fonctions associées à la distribution de Poisson s'utilisent exactement de la même façon. Sachant p. ex. que le lambda (i.e. la moyenne) associé au nombre de chênes dans une parcelle de 10 m^2 est de 0,2, le code suivant permettrait de savoir les probabilités d'obtenir un chêne ou moins dans la parcelle :

```
ppois(1, lambda = 0.2)
```

```
[1] 0.9824769
```

R nous répond 98,2 %.

Si on veut plutôt savoir à quoi pourrait ressembler une parcelle exceptionnellement riche, que l'on trouverait uniquement 1 % du temps, on utiliserait le code suivant :

```
qpois(0.99, lambda = 0.2)
```

```
[1] 2
```

Remarquez qu'ici, puisque l'on veut obtenir le 1% à droite de la distribution et que la fonction calcule dà partir de la gauche, on doit faire 1 - 0,01 = 0,99 pour trouver la probabilité à entrer dans la fonction.

Autrement dit, dans ce contexte, une parcelle contenant deux chênes serait particulièrement exceptionnelle.

11.8. Le concept d'intervalle de confiance

Nous avons vu dans le Chapitre 10 qu'à cause du fait que nous ne possédons que des échantillons et jamais d'information complète sur la population, nous devons avoir sans cesse recours à l'inférence statistique. Ce qui signifie que nous devons sans cesse aussi évaluer notre incertitude sur nos mesures.

Grâce au théorème central limite, les mathématiciens savent non seulement que l'on peut décrire la variation autour de la moyenne d'un échantillon à partir de l'écart-type. Mais ils savent aussi que l'incertitude que cette moyenne échantillonnée soit près de la vraie moyenne de la population peut être quantifiée à l'aide d'une seconde mesure, l'**erreur-type**, basée sur l'écart-type :

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$$

Autrement dit, notre incertitude sur notre estimation de la moyenne à partir d'un échantillon (notre inférence) dépend de la variabilité de l'échantillon (qui fait augmenter l'incertitude) et de la taille de l'échantillon (qui fait diminuer l'incertitude).

À partir de l'erreur-type, il est aussi possible de calculer des bornes à l'intérieur desquelles il est probable que la vraie moyenne de la population se trouve, que l'on nomme l'**intervalle de confiance**.

Ces intervalles ont toujours une notion de certitude associée, p. ex. un intervalle de confiance à 95 % confère beaucoup plus de certitude quand à la position de la moyenne qu'un intervalle à 50 %.

Par contre, inévitablement, l'intervalle de confiance à 95 % sera aussi plus large que celui à 50 %; il doit s'élargir pour qu'on soit vraiment sûr que la moyenne est dedans. Plus il est large, plus il a de chances de contenir la vraie moyenne.

Pour calculer un intervalle de confiance, on se base sur les propriétés de la distribution normale, mais l'on utilise l'erreur-type plutôt que l'écart-type pour effectuer les calculs, puisque nous faisons de l'inférence statistique. Je vous propose trois niveaux de précision différents pour effectuer vos calculs.

D'abord, "au pif", on peut se baser sur la loi du 68-95-99,7. Sachant que 95 % des données seront dans les bornes de la moyenne ± 2 erreur-type, il est facile de faire le calcul, souvent même mentalement. Si l'on sait que la moyenne de nos données est de 5 et que notre erreur-type est de 1, on sait que l'intervalle de confiance à 95 % autour de cette moyenne est entre 3 et 7, et que celui à 99 % serait entre 2 et 8 (i.e. moyenne ± 3x l'erreur-type). Ce n'est qu'un estimé grossier, mais qui donne une bonne idée quand même.

Deuxièmement, si notre échantillon était très grand, de l'ordre de centaines d'observations, on pourrait ajouter de la précision en utilisant les vraies valeurs de la loi normale. P. ex. dans la distribution normale, 95 % des données se trouvent à 1,96 écart-type de la moyenne (et non 2,0 comme dans le calcul "au pif").

Enfin, le vrai calcul d'un intervalle de confiance (celui que vous devrez utiliser dans des articles scientifiques, des rapports, etc.) utilise une autre distribution que la loi normale, soit la distribution de T de Student. Cette distribution est définie au Chapitre 12. L'important pour le moment est de savoir qu'elle ressemble beaucoup à la loi normale, mais qu'elle change de forme selon les degrés de liberté de notre échantillon. Avec de grands échantillons, elle se rapproche de la loi normale, mais avec de petits échantillons, elle s'élargit (et donc ajoute de l'incertitude à notre intervalle de confiance). La définition stricte d'un intervalle de confiance à 95 % est donc celle-ci :

$$\bar{x} \pm t_{0,05} \frac{s}{\sqrt{n}}$$

Autrement dit, la valeur de la moyenne ± la valeur de la distribution de T associée à nos degrés de liberté multipliée par l'erreur type.

Notez que ce concept d'erreur-type et d'intervalle de confiance peut s'appliquer aussi à d'autres paramètres de distribution (p. ex. sur la variance).

11.9. Labo : Calculer un intervalle de confiance de façon formelle

Si la moyenne calculée est de 4 et l'erreur-type est de 1,5 et que nous avions 10 observations dans notre échantillon, le calcul serait le suivant

pour obtenir les bornes supérieures et inférieures de l'intervalle :

```
4 + qt(0.975,10-1) * 1.5
```

```
[1] 7.393236
```

```
4 + qt(0.025,10-1) * 1.5
```

```
[1] 0.6067643
```

Il y a plusieurs choses à observer dans ce calcul. Tout d'abord, la fonction `qt` nous fournit la valeur de t associée à une probabilité et à un degré de liberté. Comme notre intervalle de confiance est bilatéral, il faut penser de séparer le 5 % (100 - 95) de chaque côté de la distribution. Ensuite, nos degrés de liberté sont 10-1, car n est notre nombre d'observations et nous n'estimons qu'un seul paramètre, soit la moyenne. Le 1,5 ici est l'erreur type (soit l'écart-type divisé par la racine carrée de la taille de l'échantillon)

11.10. Exercices

Utiliser les lois de probabilité dans R

Sachant qu'une population de truites pèsent en moyenne 4 kg et que la variabilité dans cette population peut être décrite par une loi normale avec un écart-type de 0,5 kg, répondez à ces quelques question :

À l'aide de la règle du 68-95-99.7, calculez **approximativement** dans quel intervalle de poids se trouvent 95 % des truites de ce lac.

Vous avez attrapé dans cette population un poisson qui vous paraît exceptionnellement petit à 1,5 kg. À l'aide du logiciel R, calculez la probabilité d'obtenir un poisson de cette taille ou plus petit.

En vous installant pour pêcher, vous remarquez sur l'emballage de vos hameçons que pour offrir un produit à moindre coût, la compagnie tolère 10 % d'hameçons défectueux. Calculez avec R la probabilité d'obtenir aucun hameçon défectueux dans votre emballage de 12 hameçons.

Sachant qu'en moyenne les amateurs pêchent 4 poissons par jour dans ce lac et que la variabilité autour de ce nombre suit un loi de Poisson, quelle est la probabilité que vous attrapiez 10 poissons ou plus en une journée?

Intervalle de confiance - mise en situation

Vous avez été mandatée pour évaluer la productivité d'une forêt boréale. Vous avez donc mesuré la productivité de 8 parcelles, chacune en tonnes par hectare. Voici les chiffres que vous avez obtenus : [7, 11, 8, 12, 10, 9, 7, 9].

Évaluez à partir de cet échantillon la productivité de cette forêt, et l'intervalle de confiance à 99 % de cette estimation. Vous pouvez effectuer vos calculs à l'aide de R.

11.11. Contenu optionnel : L'interprétation stricte d'un intervalle de confiance

Lorsque l'on regarde un intervalle de confiance, p. ex. 5,1 g ± 2,2 g, on est tentés de se dire qu'il y a 95 % des chances que la vraie moyenne se trouve entre ces deux bornes. Dans la vision stricte des statistiques fréquentistes (celles vues dans ce cours, par opposition à l'approche bayésienne entre autres), cette affirmation est fausse. La vraie moyenne est, ou n'est pas dans cet intervalle, mais on ne le sait pas pour cette fois là en particulier.

Le 95 % (ou tout autre chiffre) associé à l'intervalle s'applique en fait à la technique, à l'outil statistique comme tel. Il nous informe qu'en faisant le

calcul tel qu'expliqué ci-haut, 95 % du temps on obtiendra un intervalle qui contiendra la vraie moyenne de la population. Pour cette fois-ci en particulier, on a aucun moyen de savoir si la technique a fonctionné ou pas.

Autrement dit, imaginons que vous aviez une immense population d'insectes dont vous connaissez la longueur moyenne et que vous preniez 100 échantillons de 5 individus chacun. Si vous calculiez l'intervalle de confiance de la moyenne sur chacun de ces échantillons, vous en trouveriez probablement 5 dont les bornes ne contiendraient pas la vraie moyenne de votre population.

12. Initiation aux tests statistiques

12.1. À quoi servent les tests statistiques

Nous avons parlé au début de ce livre du fait que la nature est intrinsèquement variable. Il devient donc difficile de savoir si ce que nous observons est représentatif ou s'il s'agit de bizarreries dues au hasard pour cet échantillon en particulier. Les tests statistiques nous servent à mettre un chiffre sur cette incertitude. Vous apprendrez dans ce chapitre que l'on considère comme **statistiquement significatif** tout constat qui a moins de 5 % de chance d'être observable par le fruit du hasard seulement.

12.2. Mettre l'emphase sur ce qui est important

Le monde des sciences accorde souvent une grande importance au fait que notre résultat soit statistiquement significatif ou non. Il ne faut cependant jamais perdre de vue que le fait que notre résultat soit significatif ou non ne nous informe pas quant à l'ampleur du phénomène observé. Le chiffre qui nous renseigne sur cette ampleur ce nomme plutôt la **taille de l'effet**.

Si p. ex. on discute de l'apport nutritif d'une proie ingérée par un hibou et que l'on détermine qu'il existe une différence significative entre l'apport

nutritif d'une grenouille et celle d'un mulot, cela ne veut pas nécessairement dire que cette différence soit importante pour le hibou. La différence pourrait être minuscule, de l'ordre de 10 calories (la taille de notre effet), mais tout de même être statistiquement significative puisque l'on aurait échantillonné des milliers de proies et que leur apport est extrêmement constant.

Rappelez-vous, comme nous avons discuté au Chapitre 10, que la puissance statistique (que l'on pourrait redéfinir ici comme la probabilité de trouver un effet significatif) dépend à la fois de la taille de l'échantillon, de la variabilité et de l'ampleur de l'effet que l'on recherche. Hors, pour le hibou, une seule de ces choses compte vraiment : la taille de l'effet, i.e. combien de calories cette proie m'apporte-t-elle de plus par rapport à une autre.

Dans les prochains chapitres, nous discuterons surtout de méthodes permettant de savoir si un constat est statistiquement significatif ou non. Beaucoup de gens y accordent encore beaucoup d'importance, mais sachez qu'il s'agit d'une approche qui tend à changer depuis quelques années. De plus en plus de scientifiques tendent plutôt vers une approche centrée vers l'estimation des tailles d'effet et d'une mesure de confiance associée. Vous verrez d'ailleurs plusieurs de ces approches dans la section de ce livre sur les modèles linéaires (e.g. du Chapitre 27 au Chapitre 31).

Dans cette partie sur les tests statistiques, je vous demanderai simplement que chaque fois que vous arriverez à un résultat, de ne pas vous arrêter à savoir si votre résultat et significatif ou non, mais d'aller plus loin et de (1) rapporter la taille de l'effet, mais aussi de (2) l'interpréter au mieux de votre connaissance sur le sujet. Cela fera déjà de vous d'excellents scientifiques, avec l'esprit allumé nécessaire pour passer aux méthodes plus avancées dans l'avenir.

12.3. Concept d'hypothèse statistique

Au début du Chapitre 10, nous avons discuté de comment développer de bonnes questions écologiques et de comment y associer des hypothèses de travail. Pour effectuer des tests statistiques, vous aurez besoin d'effectuer une étape supplémentaire, soit de construire des hypothèses statistiques, qui seront essentiellement des versions falsifiables statistiquement de nos hypothèses de travail. On pourra affirmer, à l'aide d'un chiffre, si elles sont fausses ou non.

Chacune de vos hypothèses de travail devra être transformée en une paire d'hypothèses statistiques que l'on nomme hypothèse nulle et hypothèse alternative. L'**hypothèse nulle**, souvent nommée H_0, décrit la situation en l'absence de l'effet, du changement et ou de la différence recherchés. L'**hypothèse alternative**, souvent nommée H_1, décrit la différence, le lien ou le changement que l'on recherche.

Si l'on reprend notre hypothèse de travail où l'on avançait que la présence du tangara écarlate devait être reliée à la présence de forêt mature, elle pourrait être traduite en ces deux hypothèses statistiques :

$$H_0 : \mu_{\text{forêt mature}} = \mu_{\text{forêt jeune}}$$
$$H_1 : \mu_{\text{forêt mature}} > \mu_{\text{forêt jeune}}$$

Autrement dit, pour H_0, nous disons que le nombre moyen de tangara sera le même en forêt mature que dans les forêts plus jeunes. L'hypothèse alternative, H_1, est que le nombre de tangara moyen par parcelle sera plus grand dans la forêt mature que dans la jeune forêt. Notez que les hypothèses font référence à la population, et l'on utilise donc le symbole μ pour définir la moyenne.

L'hypothèse H_0 doit être clairement falsifiable. On doit pouvoir montrer qu'elle est fausse. C'est le fondement de toute la démarche des tests d'hypothèses.

12.4. La prise de décision statistique

Une fois ces hypothèses mises en place, il reste maintenant à effectuer le test statistique comme tel. Le processus de prise de décision est relativement simple, mais le pourquoi de la chose mérite une petite réflexion.

Comme nous en avons discuté plusieurs fois depuis le début du cours, lorsque l'on effectue une expérience (i.e. que l'on pige un échantillon dans une population), il existe une variabilité associée à cette procédure. Tous nos échantillons ne seront pas parfaitement identiques. Et donc, dans les faits, H_0 ne sera jamais vraie au sens absolu du terme. Nos deux moyennes ne seront jamais parfaitement égales. Il n'y aura jamais une absence totale de liens entre deux variables. Il s'agit d'un constat clé à comprendre pour la suite.

P. ex. en l'absence de différence pour le tangara entre la forêt mature et la jeune forêt, il y aura toujours une différence entre nos échantillons matures et jeunes. Cette différence sera la plupart du temps petite, mais il pourrait arriver parfois que le hasard fasse que la moyenne entre nos échantillons soit très différente. De la même façon qu'il pourrait arriver, une fois de temps en temps, qu'en lançant 5 pièces de monnaie en l'air, elles retombent toutes du même côté.

Des mathématiciens ont donc préparé pour chacun des tests statistiques un petit calcul pour représenter le phénomène que l'on désire mesurer (différence de moyenne, lien entre deux variables, différence de proportions, etc.), que l'on nomme la **statistique de test**. Par la suite, il ont créé des lois de probabilités qui décrivent les fréquences attendues de ces statistiques de test, si l'hypothèse H_0 est vraie. La statistique de test n'a pas d'interprétation directe, elle est une étape intermédiaire au calcul du test statistique.

Sachant tout ceci, la prise de décision d'un test statistique est plutôt simple : on calcule la statistique de test à partir de nos échantillons, et

on demande à R quelle est la probabilité d'avoir trouvé un tel résultat si H_0 est vraie. C'est ce que l'on appelle la fameuse **valeur de p**.

Si nous avons affaire à quelque chose de vraiment rare (i.e. peu probable si H_0 est vraie, donc une valeur de p faible), on dit que notre test est **statistiquement significatif**. C'est à dire qu'il serait peu probable d'avoir trouvé quelque chose d'aussi clair comme différence/lien/association si H_0 était vraie.

Qu'est-ce qui est assez rare pour être considéré comme statistiquement significatif? Toute probabilité qui est plus rare (i.e. petite) que le **seuil de signification** (α) pré-établit, qui est classiquement placé à 5 %.

Une fois que l'on a déterminé si notre résultat était statistiquement significatif ou non, on peut enfin se prononcer à savoir si l'on peut ou non rejeter l'hypothèse H_0. Certains scientifiques font un lien direct entre rejeter l'hypothèse H_0 et accepter l'hypothèse H_1, mais il faut demeurer prudent. Rejeter H_0 renforce notre croyance dans H_1, mais ce n'est pas une preuve à proprement parler, particulièrement si notre H_1 était un peu tirée par les cheveux.

12.5. Procédure recommandée

Donc, si l'on met tous ces morceaux ensemble, voici comment appliquer un test statistique :

1. Définir l'hypothèse nulle (H_0) et l'hypothèse alternative (H_1)
2. Explorer visuellement les données pour vérifier...
    a) Le respect des assomptions du test
    b) Que l'on peut voir, à l'oeil, l'effet recherché
3. Calculer la statistique de test
4. Obtenir la valeur de p de cette statistique de test dans la distribution appropriée

5. Décider de rejeter ou non l'hypothèse nulle
6. Citer la taille de l'effet et son intervalle de confiance

Lorsque vous effectuerez vos premiers tests statistiques, vous aurez probablement l'impression qu'il s'agit d'une tâche complexe, confuse, etc. C'est tout à fait normal, il y a beaucoup de matériel à digérer. Par contre, si je fais bien mon travail, au fil des semaines, cette tâche devrait devenir de plus en plus claire dans vos têtes, jusqu'à en devenir monotone et prévisible. Si vous en arrivez à ce point, j'aurai atteint mon objectif d'enseignement!

Remarquez que j'ai nommé à l'étape 2.a la vérification des assomptions. Dans d'autres cours ou avec d'autres profs, vous lirez peut-être un intitulé **conditions d'application**. J'essaie le plus possible d'éviter ce terme, car il laisse à penser que si une condition n'est pas remplie (par exemple la normalité), on ne peut strictement pas appliquer le test. Alors que dans les faits, les tests présentent pour la plupart une certaine robustesse à leur non-respect. Le terme **assomption**, quant à lui, fait référence au fait que, quand ils ont inventé le test, les mathématiciens prenaient pour acquis certaines choses à propos des données qu'on allait entrer. Si on entre autre chose, pour certains tests c'est OK jusqu'à un certain point, pour d'autres non.

12.6. Exercice : Les hypothèses

Afin de vous assurer que vous démêlez bien les différentes questions et hypothèses associées aux tests statistiques, placez au bon endroit ces quatre termes statistiques

A. Question,
B. Hypothèse de travail,
C. Hypothèse nulle (H_0),
D. Hypothèse alternative(H_1).

dans les deux mises en situation suivantes :

Effet des néonicotinoïdes sur les abeilles :

1. Il n'y a pas de différence entre le taux de décès moyen des abeilles exposées aux néonicotinoïdes et ceux qui n'y sont pas exposées.
2. Les néonicotinoïdes influencent négativement le taux de survie des abeilles.
3. Quelle est l'influence des néonicotinoïdes sur les populations d'abeilles?
4. Il existe une différence entre le taux de décès moyen des abeilles exposées aux néonicotinoïdes et ceux qui n'y sont pas exposées.

Fonctionnement du syndrome du museau blanc :

1. Il n'existe pas de différence entre le nombre de périodes d'éveil moyen des chauve-souris affectées par le syndrome du museau blanc et celles qui ne le sont pas.
2. Il existe une différence entre le nombre de périodes d'éveil moyen des chauve-souris affectées par le syndrome du museau blanc et celles qui ne le sont pas.
3. Comment le syndrome du museau blanc affecte-t-il la santé des chauve-souris?
4. Le syndrome du museau blanc augmente le nombre de périodes d'éveil pendant l'hibernation

12.7. Le test de T à un échantillon

Pour terminer cette initiation aux tests statistiques nous verrons ensemble un premier test, qui est à mon avis le plus simple et le plus facile à comprendre, c'est pourquoi je l'enseigne en premier. Il s'agit du **test de T à un échantillon**. Ce test permet de déterminer si la moyenne d'un échantillon est significativement différente d'une valeur cible.

Voyons un exemple de mise en situation où l'on peut utiliser le test de T à un échantillon. Vous employez normalement un engrais qui devrait permettre à vos plantes de pousser de 5 cm en une semaine. Vous trouvez un vieux pot de cet engrais, dont vous n'êtes plus certain de l'efficacité. Vous pourriez monter une petite expérience où vous faites pousser 25 plantes à l'aide de l'engrais. Vous pourriez ensuite utiliser le test pour savoir si elles ont effectivement grandit de 5 cm (en moyenne) durant la période ou si elles ont moins grandit (i.e. que votre engrais est périmé).

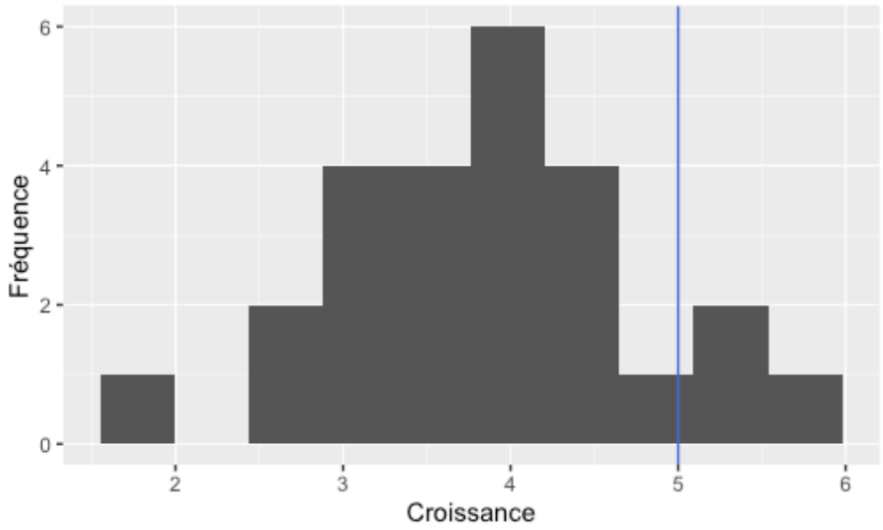
Étape 1 : Définir les hypothèses

Lorsque l'on applique le test de T à un échantillon, notre hypothèse H0 sera que la moyenne de la population est égale à la valeur de référence, p. ex. pour notre mise en situation :

$$H_0 : \mu = 5\ cm$$
$$H_1 : \mu \neq 5\ cm$$

Étape 2 : Explorer visuellement les données

Le test de T à un échantillon assume essentiellement deux choses : que notre échantillon provient d'une distribution normale et que chacune de nos observations sont indépendantes les unes des autres. La façon la plus simple et intuitive de vérifier l'assomption de normalité est d'afficher l'histogramme de notre variable. On peut aussi ajouter à cet histogramme un trait vertical représentant la valeur de référence, afin de voir si à l'oeil nos données s'éloignent de la valeur de référence ou non, comme ceci :



On peut constater principalement deux choses dans ce graphique. D'abord, la forme de nos données correspondent grossièrement à une loi normale. Il faut toujours garder en tête qu'avec de faibles tailles d'échantillon (comme ici avec 25), la forme de la courbe ne sera jamais parfaitement normale. Il faut entraîner notre œil à savoir ce qui est tolérable ou non. D'autant plus que le test de T lui même est robuste au non-respect de cette assomption. Il fournira des résultats fiables même si nos données s'éloignent de la normalité. Malheureusement, il n'existe pas de "seuil de non-normalité" à partir duquel on peut être certain que l'on est OK. Il faut utiliser notre jugement.

La deuxième chose que l'on constate dans ce graphique est que nos données semblent en général sous la barre des 5 cm de croissance. En regardant ce graphique, il faudra s'attendre à trouver un test significatif.

Étape 3 : Calculer la statistique de test

Pour le test de T à un échantillon, les mathématiciens ont défini la sta-

tistique de test comme ceci :

$$t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}}$$

Autrement dit, notre statistique de test (t) est égale à la moyenne de l'échantillon, à laquelle on soustrait notre valeur de référence (ici 5 cm). On divise ensuite cela par l'erreur-type (i.e. notre incertitude par rapport à cette moyenne, voir Chapitre 11).

Comme je vous ai parlé dans d'autres chapitres, il n'est pas très productif d'apprendre par cœur la formule pour la statistique de t. Vous devez par contre comprendre que plus l'échantillon s'éloigne de la valeur de référence, plus t sera grand (en valeur absolue) et plus nos données sont variables, plus t sera petit (en valeur absolue).

Pour notre exemple, la moyenne de notre échantillon est de 3,89 cm et l'écart-type est de 0,91 cm. La statistique de test est donc de -6,1. Le fait que la valeur de t soit négative ou positive n'a pas d'importance dans la procédure de test, l'important est comment la valeur de t s'éloigne de zéro.

Étape 4 : Obtenir la valeur de p.

Après avoir calculé la statistique de test, il faut maintenant déterminer si trouver une telle valeur de t est rare ou non. Les mathématiciens ont déterminé que si on calculait la statistique de t pour une série d'échantillons ayant la même moyenne, la distribution résultante formerait ce qu'ils ont nommé la distribution de T de Student.

Cette distribution est très semblable à la distribution normale, mais sa forme se modifie selon la taille de l'échantillon. Pour y arriver, la distribution de T nécessite deux paramètres, soit la moyenne et les degrés de liberté de notre calcul.

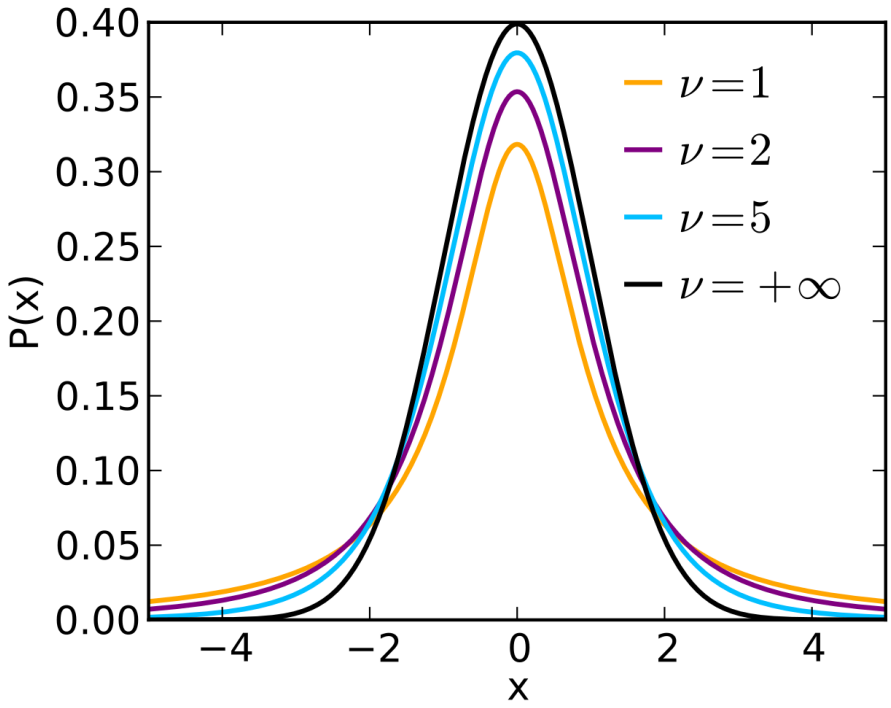


Figure 12.1.: La distribution de T de student, pour 1, 2, 5 et infini degrés de liberté (v). Skbkekas, CC BY 3.0, via Wikimedia Commons

Pour le test de t à un échantillon, les degrés de liberté sont définis comme ceci :

$$d.d.l. = n - 1$$

Pour une valeur de t de -6,1 et 24 degrés de liberté, la chance d'obtenir quelque chose d'aussi différent de la valeur de référence ou encore plus différent, si H_0 est vraie, est de 0,0000026 (notre valeur de p).

Remarquez que je vous fournis la valeur de p sans vous dire exactement

comment elle a été calculée. Comme la formule pour l'obtenir est super complexe [1] et que vous n'aurez jamais à la calculer manuellement, je vous épargne cette étape. J'espère que vous ne m'en voudrez pas trop! Si vous voulez vraiment récupérer cette valeur par vous-même, vous pouvez lancer la fonction R pt, comme ceci avec une valeur de t négative:

```
pt(-6.1,24)*2
```

```
[1] 2.665881e-06
```

Et comme ceci pour une valeur positive :

```
(1-pt(6.1,24))*2
```

```
[1] 2.665881e-06
```

R vous retournera alors la bonne valeur de p pour le t et les degrés de liberté appropriés. Remarquez qu'il faut multiplier la valeur de p par deux, car nous voulons effectuer un test bilatéral, donc en séparant la probabilité de chaque côté de la distribution.

Étape 5 : Rejeter ou non l'hypothèse nulle

À ce point, nous pouvons maintenant prendre notre décision statistique. Ici, puisque notre événement est plus rare que le seuil de signification ($p < 0,05$), on peut affirmer que l'on rejette l'hypothèse nulle. Notre moyenne est significativement différente de la valeur de référence de 5 cm.

Étape 6 : Citer la taille de l'effet et son intervalle de confiance.

Dans le cas du test de T à un échantillon, le chiffre qui nous intéresse est la moyenne de notre échantillon. Dans notre cas, 3,89 cm de croissance. On peut donc juger de l'importance de ce phénomène en constatant que

[1] https://wikimedia.org/api/rest_v1/media/math/render/svg/7fb35627dbb7e3dec4f14d60b0b58ea399966f46

notre échantillon était plus d'un cm plus court que la valeur de référence. Pour nous aider à juger de la confiance à avoir dans ce résultat, il est recommandé de l'accompagner de son intervalle de confiance.

L'intervalle de confiance à 95 % d'une moyenne se calcule comme ceci :

$$\bar{x} = t_{0,05}\frac{s}{\sqrt{n}}$$

Où $t_{0,05}$ la valeur de T associée à la probabilité de 5 %. Il faut faire attention cependant car puisque notre intervalle de confiance est bilatérale, dans les faits, il faut chercher la valeur associée à la probabilité de 0,025.

Pour notre exemple, l'intervalle de confiance autour de notre moyenne est donc de 3,89 ± 0,37, soit entre 3,52 et 4,27 cm. Vous remarquerez que, chaque fois que vous obtiendrez un test significatif au seuil de 5 %, l'intervalle de confiance à 95 % exclura toujours la valeur de référence.

Dans un rapport, vous pourriez écrire ce résultat comme ceci : « La croissance moyenne des plantes était de 3,89 cm ± 0,37 (I.C. 95 %), ce qui était significativement différent des 5 cm attendus avec l'engrais (t_{24} = 6,1, p = 0,00000026). » Remarquez que l'on nomme toujours verbalement le résultat, et on ajoute entre parenthèses l'information statistique pour le supporter.

Vous verrez souvent les valeurs de p significatives inscrites uniquement comme p < 0,05 ou p < 0,01, etc., mais ce format rend difficile la vérification de vos résultats ou leur réplication. À mon avis, il vaut mieux toujours rapporter les valeurs exactes.

12.8. Labo : Le test de T à un échantillon

Maintenant, voyons comment effectuer votre premier test statistique dans R. Vous serez rassuré de voir que tout le détail des étapes 3 et 4 de la démarche présentée ci-haut se résume à une seule ligne de code, qui fait tout pour vous!

Pour notre petit laboratoire, nous tenterons de répondre à la question : est-ce que les manchots Chinstrap de l'archipel Palmer se distinguent des autres populations de manchots Chinstrap, pour lesquels nous savons que le poids moyen est de 4250 g[2]. Étant donné que l'archipel est situé à l'extrême sud de la distribution de l'espèce, notre hypothèse de travail sera que les manchots de l'archipel auront un poids différent du 4250 g de référence.

Étape 1 :

Nos hypothèses statistiques seront donc : $H_0 : \mu = 4250$ g $H_1 : \mu \neq 4250$ g

Étape 2 :

Pour faciliter notre travail, nous allons nous préparer un petit tableau de données contenant uniquement les manchots Chinstrap pour lesquels nous connaissons le poids.

```
library(tidyverse)
```

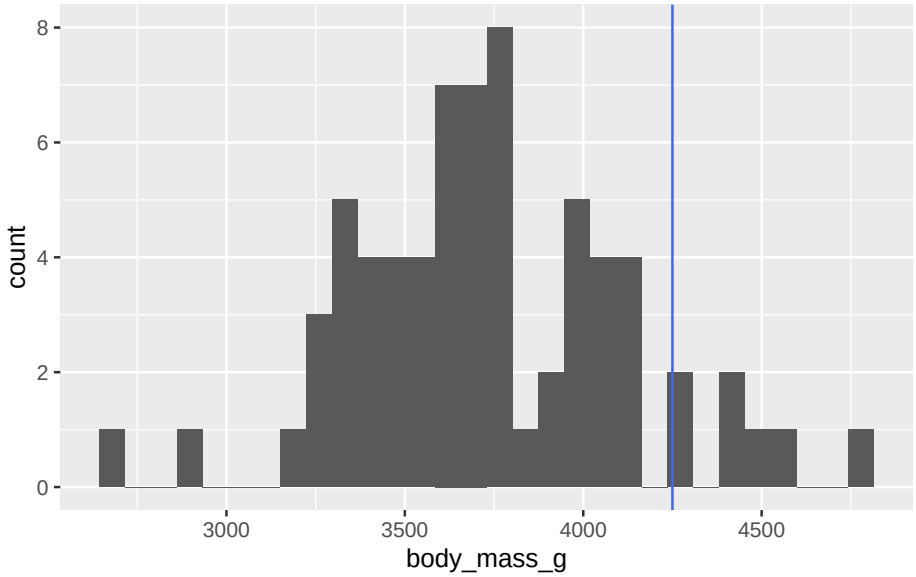
```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ----------------- tidyverse_conflicts() --
```

[2] https://en.wikipedia.org/wiki/Chinstrap_penguin#Description

```
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(palmerpenguins)

chinstrap <- penguins |>
  filter(species == "Chinstrap") |>
  drop_na(body_mass_g)

ggplot(data = chinstrap) +
  geom_histogram(mapping = aes(x = body_mass_g)) +
  geom_vline(xintercept = 4250, color = "royalblue")
```

```
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```



Donc, à première vue, l'assomption de normalité semble respectée pour cette variable. Pour ce que l'on en sait, nos observations sont indépendantes les unes des autres.

Enfin, à première vue, le poids des manchots Chinstrap de l'archipel Palmer semble situé près de 3750, ce qui est bien en dessous de la moyenne de l'espèce. Il faudra sans doute s'attendre à trouver un effet significatif.

Étapes 3 et 4 :

Les étapes 3 et 4 peuvent être résumées en une seule ligne de R, qui va à la fois calculer la statistique de test et déterminer la valeur de p associée, avec les bons degrés de liberté etc. :

```
t.test(chinstrap$body_mass_g, mu = 4250)
```

La fonction `t.test`, contrairement à celles des librairies `ggplot2` et `dplyr`, n'est pas conçue pour travailler avec des tableaux de données. Elle s'attend plutôt à recevoir une simple série de nombres, que l'on nomme **vecteur** dans R. Pour extraire un vecteur d'un tableau de données, R nous offre l'opérateur `$`. Vous pouvez d'ailleurs voir le contenu du vecteur directement, comme ceci :

```
chinstrap$body_mass_g
```

```
 [1] 3500 3900 3650 3525 3725 3950 3250 3750 4150 3700
[11] 3800 3775 3700 4050 3575 4050 3300 3700 3450 4400
[21] 3600 3400 2900 3800 3300 4150 3400 3800 3700 4550
[31] 3200 4300 3350 4100 3600 3900 3850 4800 2700 4500
[41] 3950 3650 3550 3500 3675 4450 3400 4300 3250 3675
[51] 3325 3950 3600 4050 3350 3450 3250 4050 3800 3525
[61] 3950 3650 3650 4000 3400 3775 4100 3775
```

Le deuxième argument (mu) indique à R quelle valeur de référence nous voulons utiliser pour notre test de T à un échantillon. Si on omet cet argument, R assumera que la valeur de référence est zéro.

Après avoir lancé notre commande de test de T, R nous répond ceci :

```
    One Sample t-test

data:  chinstrap$body_mass_g
t = -11.091, df = 67, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 4250
95 percent confidence interval:
 3640.059 3826.117
sample estimates:
mean of x
 3733.088
```

La première ligne nous indique que R a appliqué un test de T à un échantillon (*One Sample t-test*). La deuxième (*data:*) nous indique sur quelles données a été calculé le test. La ligne suivante nous montre la valeur de t calculée, les degrés de liberté (df, pour *Degrees of Freedom*) et la valeur de p (*p-value*) de notre test. La ligne suivante (*alternate hypothesis...*) nous informe de notre hypothèse alternative, soit que la moyenne est différente de 4250. Les dernières lignes nous fournissent l'intervalle de confiance à 95 % de notre moyenne, soit une moyenne de 3733 et un intervalle de confiance allant de 3640 à 3826.

Remarquez la façon dont R nous rapporte la valeur de p de notre test : `< 2.2e-16`.

Tout d'abord, plutôt que nous fournir la valeur en notation décimale, R nous la fournit en **notation scientifique**. 2.2e-16 est sa façon de dire 2.2×10^{-16}, soit 0.00000000000000022 (i.e. on a besoin de 16 zéros pour l'écrire).

Ensuite, bien que R soit capable de calculer très précisement les valeurs de p, la notre est tellement petite que R ne peut pas nous fournir le chiffre exact. Il peut seulement nous dire que c'est plus petit que 2.2×10^{-16}. Mais n'ayant crainte, dans la vraie vie, cela vous arrivera très rarement.

Étape 5 :

Comme notre valeur de p est plus rare que le seuil de signification de 0,05, nous rejetons l'hypothèse nulle H_0 qui stipulait que les manchots Chinstrap de Palmer auraient le même poids que le reste de l'espèce. Nous favorisons donc notre hypothèse H_1 qui avançait que leur poids moyen serait différent de 4250.

Étape 6 : Voici ce que nous pourrions écrire comme résultats pour présenter notre découverte : « Le poids moyen des manchots Chinstrap de l'archipel Palmer était de 3733 g ± 93 g (I.C. 95 %), ce qui était significativement différent des 4250 g attendus si leur poids correspondait à la moyenne de l'espèce (t_{67} = -11,091, $p < 2.2 \times 10^{-16}$). »

Remarquez que 3733 g ± 93 n'est qu'une façon alternative d'écrire l'intervalle 3640 à 3826. Une façon ou l'autre est acceptée comme façon de décrire l'intervalle.

12.9. Contenu optionnel : Appliquer un test statistique *à l'ancienne*.

La section qui suit est à titre informatif seulement. Elle pourrait même entraîner une certaine confusion. Libre à vous de la lire ou non.

À l'époque où les tests statistiques ont été développés, les scientifiques n'avaient pas accès aux ordinateurs ultra-rapides d'aujourd'hui. Calculer la valeur de p exacte comme nous l'avons fait ci-haut en quelques secondes pouvait prendre des heures de calcul manuel. C'est pourquoi les scientifiques ont longtemps utilisé une approche légèrement différente pour appliquer leurs tests, qui est encore parfois enseignée.

Les étapes 1, 2, 3 et 6 du processus étaient les mêmes, mais les étapes 4 et 5 étaient différentes. En fait, l'étape 4 du calcul de la valeur de p était carrément absente. Les scientifiques utilisaient plutôt une série de tables, remplies de chiffres, dans lesquels on pouvait trouver les valeurs de t correspondant à un seuil de signification et à un degré de liberté donné. Voici p. ex. un extrait d'une de ces tables pour la distribution de T de Student :

d.d.l / α	0,05	0,025	...
1	6,314	12,076	...
2	2,920	4,303	...
3	2,353	3,182	...
...	...	...	...

Extrait de la table des valeurs seuils pour la distribution de T de Student.

Les dernières pages des manuels de statistiques étaient remplies de ce genre de tables. Pour les utiliser, le scientifique décidait d'abord du seuil de signification de son expérience (habituellement 0,05) et déterminait les degrés de liberté. Il allait par la suite consulter la table pour savoir quelle valeur de t il devait dépasser (en valeur absolue) pour considérer son test comme significatif. La décision était donc prise en comparant les valeurs de t plutôt qu'en comparant une valeur de p à un seuil de signification.

Comptez-vous chanceux d'apprendre l'analyse statistique aujourd'hui, parce que lorsque j'ai suivi mon cours de biostatistiques il n'y a pas si longtemps (2010), c'était encore la façon de faire qui était enseignée, et une fois sur deux, on perdait des points en se trompant en consultant la table!

12.10. Tests unilatéraux ou bilatéraux

Vous avez probablement remarqué dans les exemples de tests de t ci-haut, que j'ai toujours parlé de regarder si notre échantillon était différent de la valeur seuil, plutôt que spécifiquement chercher si il était plus grand ou plus petit. Il existe une excellente raison derrière cette décision! Lorsque nous effectuons un test qui cherche une différence (quelle que soit la direction), le seuil de signification est réparti de façon égale, moitié-moitié de chaque côté de la distribution. Ces zones sont donc petites, et notre test est réputé être **conservateur**, c'est-à-dire peu à risque d'erreur de type I. Si nous choisissons de regarder uniquement si la moyenne de notre échantillon est plus petite (ou plus grande selon les cas) que notre valeur seuil, alors l'ensemble de la zone de signification se retrouve du même côté :

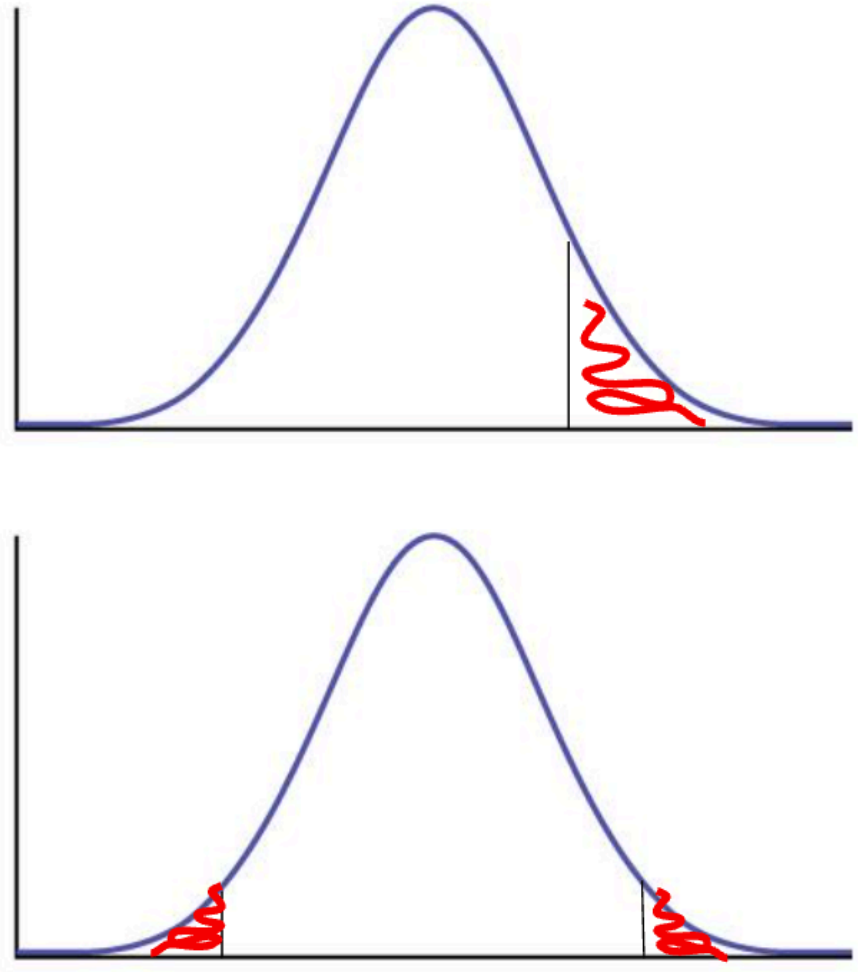


Figure 12.2.: Illustration de la différence de la zone se signification entre un test unilatéral (en-haut) et un test bilatéral (en-bas).

Pour les cas où votre valeur de p est extrêmement faible ou très grande, faire un test unilatéral ou bilatéral ne fera aucune différence. Par contre, si votre résultat se trouve dans la zone qui passe du rouge au blanc entre les deux types de tests, vous pourriez être accusé d'avoir "gonflé" vos résultats si vous utilisez un test unilatéral.

Donc, à moins de raisons conceptuelles et théoriques extrêmement solides, utilisez toujours un test bilatéral. Il s'agit là d'une pratique à bien intégrer à vos habitudes d'analyse de données : en cas de doutes, optez toujours pour la solution la plus conservatrice, c'est-à-dire celle qui minimisera votre risque d'erreur de type I.

Si néanmoins vous désirez effectuer un test de t unilatéral, vous pouvez spécifier de quel côté va votre hypothèse H_1 à l'aide de l'argument nommé alternative :

```
t.test(chinstrap$body_mass_g, my = 4250, alternative =
 ↪  "less")
```

```
    One Sample t-test

data:  chinstrap$body_mass_g
t = 80.096, df = 67, p-value = 1
alternative hypothesis: true mean is less than 0
95 percent confidence interval:
     -Inf 3810.826
sample estimates:
mean of x
 3733.088
```

Si vous vouliez tester spécifique pour plus grand que 4250, inscrivez "greater" plutôt que "less"

12.11. Contenu optionnel : Sur l'importance de la taille de l'effet

Pour terminer ce chapitre, je vais vous raconter une petite anecdote personnelle illustrant bien l'importance d'évaluer la taille de l'effet, même si un résultat est statistiquement significatif.

Lorsque ma conjointe était enceinte de notre deuxième enfant, elle est venue me voir un peu en panique un matin, après avoir lu un article qui disait que les femmes de petite taille avaient plus de risque d'accoucher prématurément[3]. L'article de vulgarisation sur lequel elle était tombée ne mentionnait qu'un seul chiffre, soit la taille de l'échantillon, mais aucune mention sur le risque comme tel, de combien de jours plus courte était la grossesse des femmes de plus petite taille, etc.

Mon esprit de scientifique m'a apporté sur Google Scholar où j'ai retracé l'article original[4]. Dans ce dernier, on insistait beaucoup sur le fait que les résultats étaient extrêmement significatifs, la valeur de p étant de 0,000000151. Par contre, ils insistaient très peu sur le fait que la taille de l'effet n'était que de 0,3 jours de grossesse de moins par tranche de 1 cm de grandeur de la mère. Un petit calcul rapide pour ma conjointe, qui est particulièrement petite (10 cm sous la moyenne des femmes au Canada), nous donne un gros 3 jours de moins de grossesse que la moyenne. Sachant que la grossesse est déjà un phénomène extrêmement variable, qui dure 95 % du temps entre 260 et 300 jours, cette différence de 3 jours était somme toute peu importante.

Si les auteurs de l'étude avaient mis de l'avant la taille de l'effet plutôt que le fait que leur résultat était hautement significatif, ils auraient évité beaucoup de stress inutile (et en fait, on n'aurait peut-être même pas discuté de leur étude dans les journaux grands-public).

[3] https://naitreetgrandir.com/fr/nouvelles/2015/09/07/20150907-grossesse-plus-courte-petites-femmes/

[4] https://journals.plos.org/plosmedicine/article?id=10.1371/journal.pmed.1001865

Morale de l'histoire, mettez toujours, svp, l'emphase sur la taille de l'effet dans vos résultats!

partie III.

Les tests

13. Tests de comparaison de variance

Nous verrons dans les prochains chapitres que plusieurs tests assument l'égalité des variances entre deux groupes. Nous verrons donc dans le présent chapitre les détails d'un test permettant de comparer la variance de deux échantillons. Nous verrons aussi deux stratégies si jamais vous avez à en tester simultanément plus de deux.

13.1. Le test de F

Le test de F est un test destiné à comparer la variance entre deux groupes pour savoir si elle est statistiquement différente. Outre la vérification des assomptions de certains tests mentionnée plus haut, le test de F peut aussi être appliqué à certains problèmes concrets. Il pourrait être utilisé par exemple pour tester l'hypothèse que la taille des oiseaux en milieu urbain est plus homogène (i.e. moins variable) qu'en milieu naturel. Vous pourriez donc avoir capturé 20 oiseaux en milieu urbain et mesuré leur poids, puis 40 en milieu naturel et aussi mesuré leur poids.

Généralement dans la littérature, on utilisera le terme **homoscédastique** pour désigner des jeux de données ayant la même variance et **hétéroscédastique** pour des jeux de données ayant des variances différentes.

Étape 1 : Définir les hypothèses

Les hypothèses statistiques du test de F sont pour l'hypothèse nulle (H_0) que les variances sont égales entre les groupes et l'hypothèse alternative (H_1) est qu'il existe une différence de variance entre les deux groupes.

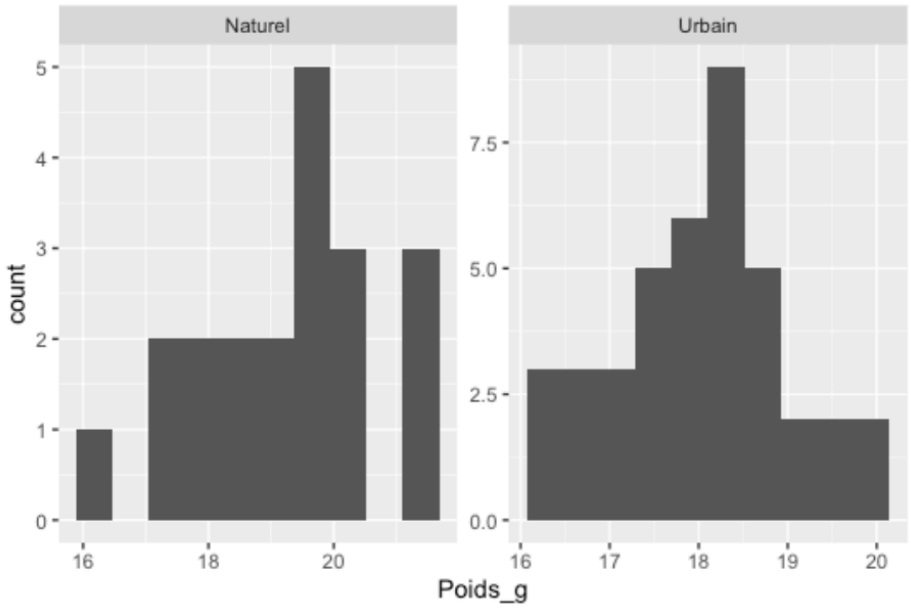
Pour notre exemple sur les oiseaux, nos hypothèses statistiques seraient donc les suivantes :

$$H_0 : \sigma^2_{urbain} = \sigma^2_{naturel}$$
$$H_1 : \sigma^2_{urbain} \neq \sigma^2_{naturel}$$

Remarquez à nouveau que pour les hypothèses, on utilise le symbole grec de la variance (σ^2) et non celui de l'échantillon (s^2).

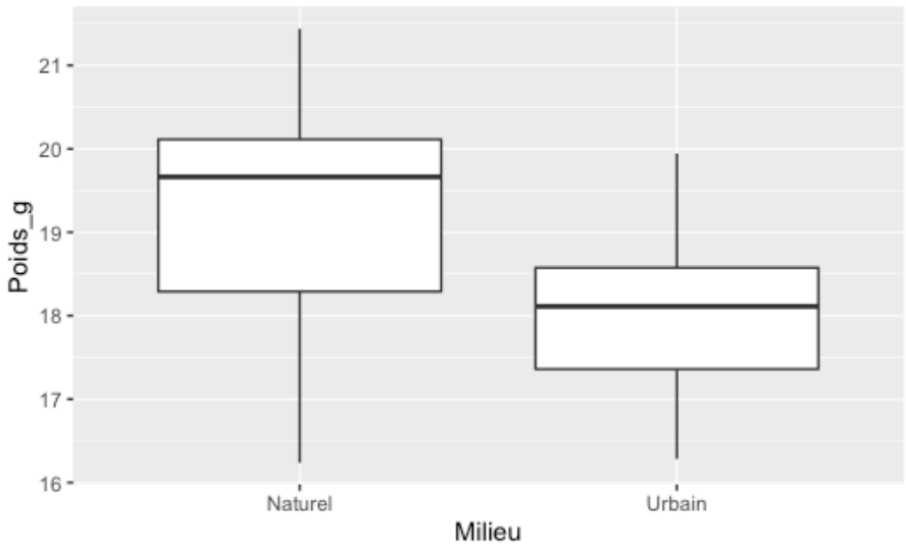
Étape 2 : Explorer visuellement les données.

Le test de F ne compte que deux assomptions. La première, qui est commune à tous les tests, est l'indépendance des observations. Cette assomption est surtout dépendante du design de l'expérience. La deuxième assomption, qui peut quant à elle être vérifiée visuellement, est que les deux échantillons proviennent de populations présentant des distributions normales. La façon la plus simple de vérifier ce fait est de faire un histogramme de fréquence de chacun de nos échantillons. Comme le test de F est relativement sensible aux écarts à la normalité, on peut se permettre d'être relativement strict sur cette assomption.



À première vue, rien ne suggère que les échantillons ne proviendraient pas de distributions normales, on peut donc procéder à la suite.

Une fois les assomptions vérifiées, on peut passer à la visualisation du test comme tel. Pour regarder si un échantillon est plus variable qu'un autre, on peut, entre autres, utiliser un diagramme à moustache et comparer la **taille** des boîtes :



Ici, il faut bien comprendre la nuance entre une différence de moyenne et une différence de variance. Les oiseaux en milieu urbain semblent à première vue plus petits (médiane environ 18 g vs. presque 20 g pour le milieu naturel), mais ce n'est pas cette question qui nous intéresse. Nous nous intéressons à la variabilité (la taille des boîtes). Bien que la boîte présente l'écart interquartile plutôt que la variance, elle donne quand même une bonne idée de la variabilité. Ici, les oiseaux en milieu urbain semblent moins variables (boîte plus petite) que ceux en milieu naturel. Il ne faudrait pas s'étonner que cette différence soit significative puisque l'on observe facilement la différence à l'œil nu.

Étape 3 : Calculer la statistique de test.

La statistique du test de F a été définie par les mathématiciens comme ceci :

$$F = \frac{s_1^2}{s_2^2}$$

Autrement dit, la statistique se calcule comme le ratio de la variance d'un échantillon divisé par la variance du deuxième. On utilise ici le symbole s puisque l'on parle des données de l'échantillon. Dans cette équation, on place toujours la variance la plus élevée au numérateur (en haut) et la plus faible au dénominateur (en bas). Donc, intuitivement, plus nos variances sont différentes l'une de l'autre, plus la valeur de F sera grande. Si nos variances sont absolument égales, F vaudrait 1.

Si dans notre exemple la variance des oiseaux urbains est de 0,886 g^2 et celle des oiseaux en milieu naturel est de 2,10 g^2, notre statistique de F sera de 2,37. Autrement dit, la variance du milieu naturel est un peu plus de 2x plus grande que celle en milieu urbain.

Étape 4 : Obtenir la valeur de p.

Il faut maintenant déterminer si obtenir une telle valeur de F est rare ou non lorsque l'on pige deux échantillons dans des populations de variances égales (notre hypothèse nulle). Les mathématiciens ont décrit la distribution de F exactement pour ce genre de situations. La distribution de F se définit par deux paramètres, soit les degrés de liberté du numérateur et du dénominateur. Et contrairement à la distribution de T, la distribution de F peut être asymétrique :

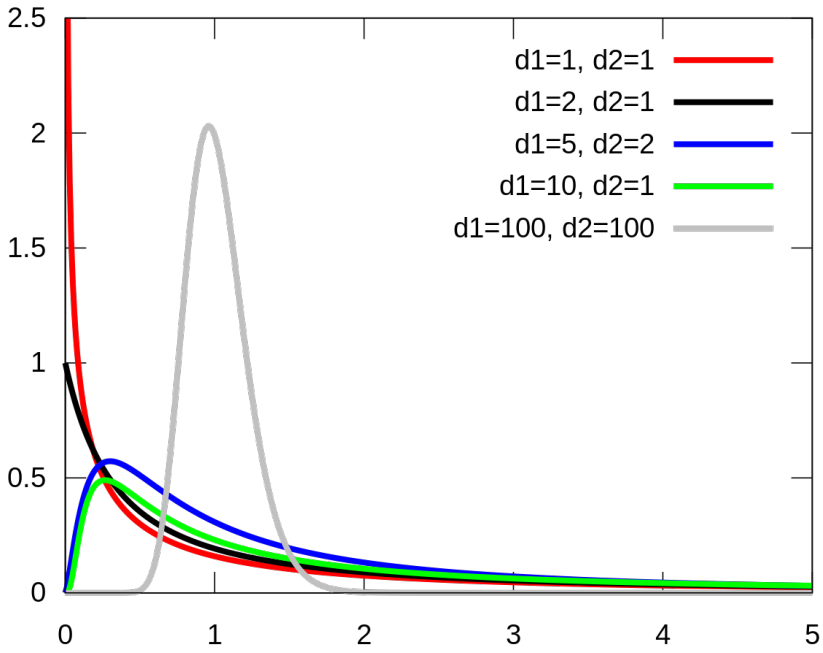


Figure 13.1.: La distribution de F pour différents degrés de liberté (d1 et d2).IkamusumeFan, CC BY-SA 4.0, via Wikimedia Commons

Les degrés de liberté du numérateur et du dénominateur se définissent comme n_1-1 pour le numérateur et n_2-1 pour le dénominateur.

Pour notre exemple, avec une valeur de F de 2,37 et des degrés de liberté de 19 et 39, nous obtenons une valeur de p de 0,023. Vous pourriez récupérer cette valeur manuellement à l'aide de la fonction `pf` de R, mais R fera ce calcul pour vous dans le test.

Étape 5 : Rejeter ou non l'hypothèse nulle

À ce point, nous pouvons maintenant prendre notre décision statistique.

Puisque notre événement est plus rare que le seuil de signification ($p < 0,05$), on peut rejeter l'hypothèse nulle. Les variances de nos deux groupes sont significativement différentes.

Étape 6 : Citer la taille de l'effet et son intervalle de confiance.

Dans le cadre d'une utilisation du test de F pour vérifier les assomptions d'un test, vous pouvez vous arrêter à l'étape précédente. Par contre, si vous avez utilisé le test de F pour tester une hypothèse biologique ou écologique comme dans notre exemple, vous devrez aussi présenter l'intervalle de confiance associé aux résultats.

Dans le cadre du cours, je ne vous demanderai pas de calculer manuellement cet intervalle. Sachez seulement qu'il est représenté dans les résultats de R, et que pour notre exemple, il se situe entre 1,13 et 5,53. Autrement dit, le ratio de la variance de nos deux populations se situe de façon très probable entre un peu plus de 1 (à peine plus grand) et 5x plus grand. Notez que cet intervalle exclut 1 (l'égalité des variances), ce qui explique que notre test soit significatif.

Dans un rapport, vous pourriez écrire ce résultat comme ceci : «Il existait une différence significative de variance du poids des oiseaux entre les milieux urbains et naturels ($F_{19,39} = 2,37$, $p = 0,023$).»

13.2. Labo : Le test de F

Assumons pour ce laboratoire que nous savons que l'île Biscoe de l'archipel Palmer est beaucoup plus diversifiée en termes de niches écologiques que l'île Torgersen. Nous pourrions donc émettre l'hypothèse que le poids des manchots Adélie sur l'île de Biscoe devrait être plus variable que celui de l'île de Torgersen, si cette variabilité affecte l'alimentation des manchots.

Étape 1 :

$$H_0 : \sigma^2_{Biscoe} = \sigma^2_{Torgersen}$$
$$H_1 : \sigma^2_{Biscoe} \neq \sigma^2_{Torgersen}$$

Étape 2 :

Pour faciliter l'exploration visuelle de nos données, nous créerons deux mini-tableaux de données contenant uniquement les manchots Adélie de l'île Biscoe et Torgersen, comme ceci :

```
library(palmerpenguins)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
adelie <- penguins |>
  filter(island %in% c("Biscoe", "Torgersen")) |>
  filter(species == 'Adelie') |>
  drop_na(body_mass_g)

biscoe <- adelie |> filter(island == "Biscoe")
torgersen <- adelie |> filter(island == 'Torgersen')
```

N'oubliez pas d'activer les librairies nécessaires!

Avant de commencer, c'est toujours une bonne idée de se faire un résumé numérique de nos données

```
adelie |>
  group_by(island) |>
  summarize(
    n = n(),
    moyenne = mean(body_mass_g),
    variance = var(body_mass_g)
  )
```

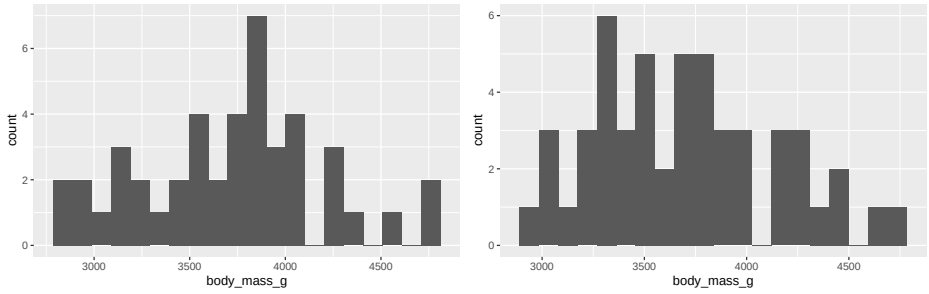
```
# A tibble: 2 x 4
  island        n moyenne variance
  <fct>     <int>   <dbl>    <dbl>
1 Biscoe       44   3710.  237884.
2 Torgersen    51   3706.  198121.
```

Donc, nous avons 44 individus sur l'île Biscoe et 51 sur Torgersen. La variance de l'échantillon est plus élevée sur Biscoe que Torgersen.

Par la suite, on peut facilement créer un histogramme pour chacun des groupes :

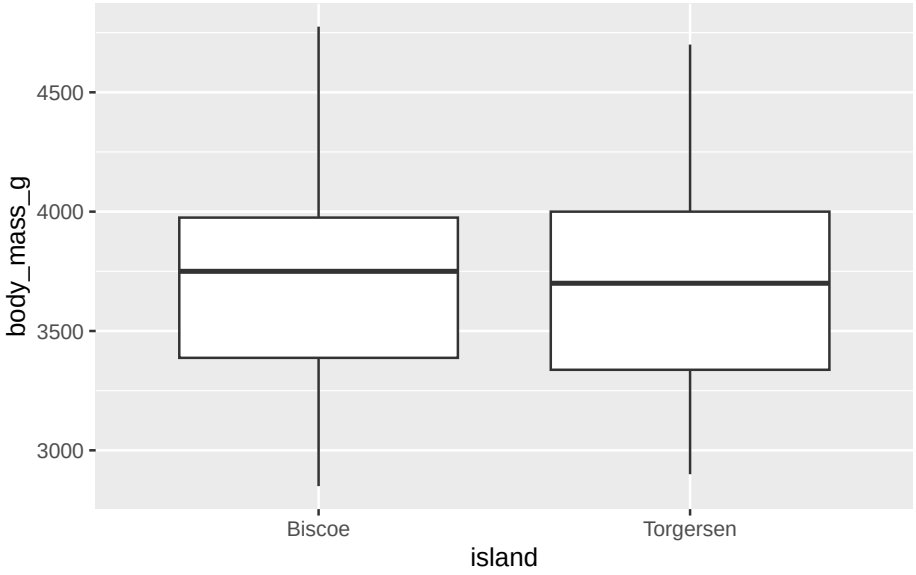
```
biscoe |>
  ggplot() +
  geom_histogram(aes(x = body_mass_g), bins = 20)
torgersen |>
  ggplot() +
  geom_histogram(aes(x = body_mass_g), bins = 20)
```

Nos distributions sont relativement normales, rien pour nous inquiéter ici.



On peut ensuite se représenter visuellement le résultat du test, à l'aide d'un diagramme à moustaches :

```
adelie |>
  ggplot(aes(x = island, y = body_mass_g)) +
  geom_boxplot()
```



À première vue, on ne s'attend pas à trouver une grande différence dans la variabilité de ces deux groupes : la taille des deux boîtes est TRÈS semblable.

Étapes 3 et 4

La fonction pour calculer le test de F dans R se nomme `var.test` (comme "test de variance"). Il existe plusieurs façons de spécifier les données de ce test, mais la plus directe est d'aller chercher (comme pour le test de T) les valeurs de la bonne colonne à l'aide de l'opérateur `$`, comme ceci :

```
var.test(biscoe$body_mass_g, torgersen$body_mass_g)
```

```
    F test to compare two variances

data:  biscoe$body_mass_g and torgersen$body_mass_g
F = 1.2007, num df = 43, denom df = 50, p-value
= 0.5307
alternative hypothesis: true ratio of variances is not
equal to 1
95 percent confidence interval:
 0.6741525 2.1663828
sample estimates:
ratio of variances
          1.200701
```

La première ligne nous indique qu'effectivement, R a appliqué un test de F. Ensuite, il nous rappelle avec quelles données il a fait le test. Puis il nous fournit la statistique de test qu'il a calculée (F, le ratio des deux variances), les degrés de liberté (*num df* et *denom df*) et la valeur de p (*p-value*). Les dernières lignes nous rapportent le ratio des variances et son intervalle de confiance.

Étape 5 :

Comme l'événement observé est relativement commun (p=0,5307), on ne peut pas rejeter l'hypothèse H_0 au seuil de 0,05. Il n'y a pas de différence significative entre la variance du poids des manchots Adélie entre ces deux îles.

Autrement dit, il est très commun d'observer des différences telle qu'un ratio de 1,20 avec une telle taille d'échantillon, si les variances des deux populations sont égales.

On pourrait donc écrire ce résultat comme ceci :

«Il n'existait pas de différence significative entre les variances du poids des manchots Adélie des îles Torgersen et Biscoe ($F_{43,50}$ = 1,20, p = 0,5307).»

13.3. Comparer plus de deux variances

Lorsque nous nous attaquerons à des tests plus complexes permettant de gérer plus de deux échantillons à la fois, vous aurez à vous demander s'il existe une différence de variance significative dans l'ensemble ces derniers. Une façon de faire relativement simple dans ces cas est de tester s'il existe une différence significative entre le groupe ayant la variance la plus élevée et le groupe ayant la variance la moins élevée. Si cette différence est significative, vous savez que l'assomption n'est pas respectée. Si la différence n'est pas significative, vous pouvez assumer que toutes les autres variances ne sont pas différentes les unes des autres.

Si jamais il vous arrive d'avoir à tester des différences de variance pour lesquelles vous voulez regarder plus en détail chacun des échantillons, sachez qu'il existe pour ce genre de situation le test de Levene et le test de Bartlett, mais que ces deux tests dépassent le cadre de ce que nous aurons besoin dans ce livre.

13.4. Exercice : Le test de F

Vous savez, de part vos lectures antérieures, que la forme du corps des poissons est reliée au type de milieu dans lequel ils vivent. Pour tester si cette théorie a des implications pour la biodiversité, vous posez la question à savoir si la taille des poissons est plus variable dans les lacs comprenant des fosses que dans les lacs qui n'ont pas vraiment de zones profondes.

Vous avez capturé et mesuré (en cm) :

- 12 poissons dans un lac sans fosse [28, 22, 22, 28, 26, 23, 25, 21, 24, 21, 26, 24] et
- 8 poissons dans un lac avec fosse [23,20,28,25,25,29,29,22].

Évaluez à l'aide d'un test de F si la variance est différente entre ces deux lacs. Autrement dit, est-ce que la taille des poissons est plus diversifiée dans les lacs comprenant une fosse que dans les lacs n'en comprenant pas?

14. Tests de comparaison de deux moyennes

Il existe plusieurs façons différentes de comparer la moyenne de deux échantillons pour savoir si ceux-ci proviennent de populations ayant des moyennes différentes. Nous en verrons trois dans ce chapitre, soit le test de T à deux échantillons (lorsque les variances sont égales), le test de T de Welch (lorsque les variances sont inégales) et le test de T pairé. Ce chapitre peut paraître un peu long puisqu'il présente trois tests un après l'autre, mais leur principe de fonctionnement étant le même, cela devrait demeurer relativement digeste.

14.1. Le test de T à deux échantillons

Le test de T à deux échantillons s'utilise pour comparer la moyenne de deux échantillons pour savoir si elles sont significativement différentes, lorsque la variance de ces deux échantillons est égale. On pourrait par exemple l'utiliser pour savoir si les oiseaux en milieu urbain sont en moyenne plus gros que les oiseaux en milieu naturel. On pourrait appliquer ce test avec les mêmes données qu'au Chapitre 13, c'est-à-dire par exemple si nous avions capturé 20 oiseaux en milieu urbain et mesuré leur poids, puis 40 en milieu naturel et aussi mesuré leur poids.

Étape 1 : Définir les hypothèses

Les hypothèses statistiques du test de T sont pour l'hypothèse nulle (H_0) que les moyennes sont égales entre les groupes et l'hypothèse alternative (H_1) est qu'il existe une différence de moyenne entre les deux groupes.

Pour notre exemple sur les oiseaux, nos hypothèses statistiques seraient donc les suivantes :

$$H_0 : \mu_{urbain} = \mu_{naturel}$$
$$H_1 : \mu_{urbain} \neq \mu_{naturel}$$

Étape 2 : Explorer visuellement les données

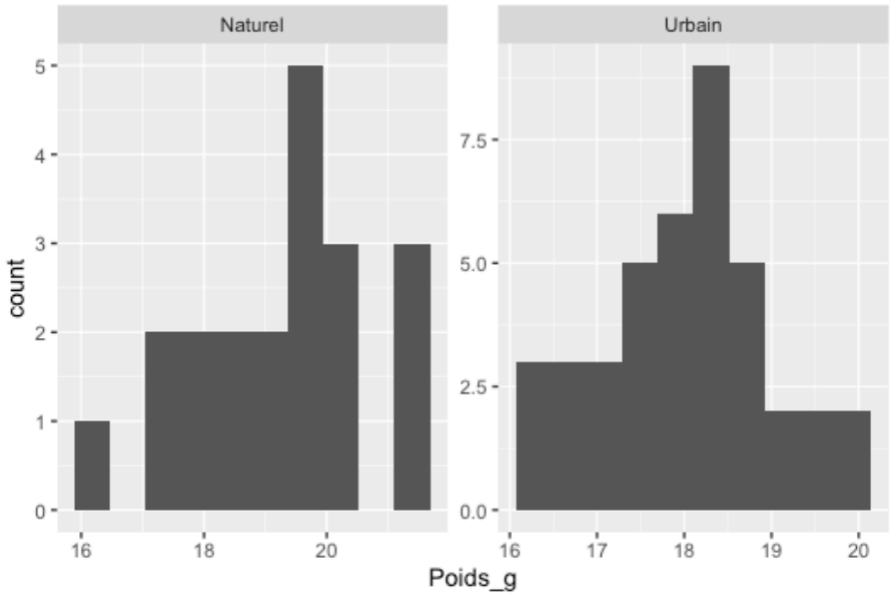
Le test de T à deux échantillons comporte trois assomptions importantes. La première concerne l'égalité des variances. Avant de procéder à l'application d'un test de T à deux échantillons, il faut impérativement déterminer si les variances sont égales ou non à l'aide d'un test de F. Si les variances sont significativement différentes, on devra plutôt utiliser le test de Welch (voir Section 14.2) Comme nous venons de faire ce test au Chapitre 13, nous ne le ré-appliquerons pas ici, mais dans tous vos exercices et toutes vos analyses, vous devrez le faire !

La deuxième assomption importante du test de T à deux échantillons est que les observations sont indépendantes les unes des autres. Un cas classique où cette assomption ne serait PAS respectée serait si vous avez mesuré des individus à deux dates différentes et vous voudriez savoir si leur poids a diminué ou non. Dans ce cas, les observations ne sont pas indépendantes, puisque le même individu est mesuré plusieurs fois. Il faudrait alors appliquer le test de T pairé (voir Section 14.4).

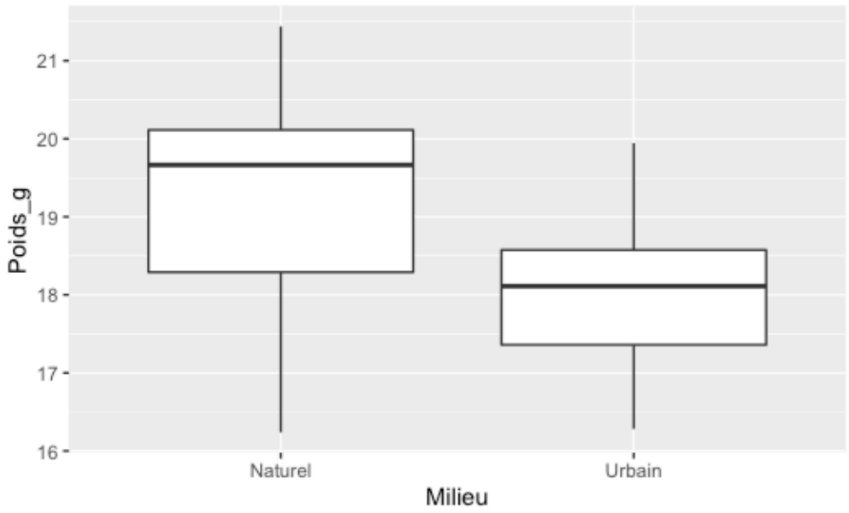
La troisième assomption du test de T à deux échantillons est que les données proviennent d'une distribution normale. Comme le test est relativement robuste par rapport à cette assomption, on peut tolérer une certaine non-normalité. Il n'est pas nécessaire d'utiliser un test de normalité (i.e. Shapiro-Wilk), puisque beaucoup de distributions que ce test

considéreraient comme non-normales seraient tout de même appropriées pour le test de T.

On peut donc se contenter d'une exploration visuelle de nos deux groupes à l'aide d'un histogramme, comme ceci :



Enfin, une fois les assomptions vérifiées, il importe aussi d'aller voir à quoi ressemble la moyenne des deux échantillons, pour se faire une idée avant de commencer, si l'on pense trouver une différence moyenne ou non entre les groupes. La façon idéale d'inspecter une différence de moyenne entre deux groupes et de tracer un diagramme à moustaches de ces données :



Au premier coup d'oeil, on voit une différence de près de 2 g entre nos deux groupes, et cette différence paraît assez claire par rapport à la variabilité pour être statistiquement significative.

Étape 3 : Calculer la statistique de test

Je vous inscris ici les formules pour que vous les ayez sous la main si vous en avez besoin un jour, mais vous n'aurez probablement jamais besoin de les calculer manuellement :

$$T = \frac{\bar{x_1} - \bar{x_1}}{\sqrt{s^2(\frac{1}{n_1} + \frac{1}{n_2})}}$$

Où n_i est le nombre d'observations, x_i la moyenne de l'échantillon i et s^2 est l'estimé de variance commun (*pooled*) calculé comme ceci :

$$s^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}$$

L'important est surtout de comprendre que plus la différence entre les moyennes sera grande, plus la valeur de T sera grande aussi (en valeur absolue). À l'inverse, plus la variabilité des données sera grande, plus la statistique de T sera près de zéro. Les deux ingrédients pour trouver un test de T significatif sont donc une grande différence de moyenne entre les groupes et une petite variabilité entre les individus.

Pour notre exemple, x_1 est 19,27 g et x_2 est 18,03 g. La taille de nos échantillons n_1 et n_2 est respectivement de 40 et 20 individus et la variance de chaque groupe était de 2,10 g^2 et de 0,886 g^2. La valeur de notre statistique de T est donc de 4,01.

Étape 4 : Obtenir la valeur de p

Comme pour le test de T à un échantillon (voir Chapitre 12), la statistique de T calculée ici suit une distribution de T, mais avec n_1+n_2-2 degrés de liberté.

Pour notre exemple, les degrés de liberté sont donc de 58 et la valeur de p est de 0,00017.

Étape 5 : Rejeter ou non l'hypothèse nulle

Nous pouvons maintenant prendre notre décision statistique. Puisqu'une telle valeur de statistique de T est très rarement observée dans de telles circonstances si les moyennes des populations étaient égales (0,00017 est vraiment plus petit que 0,05), alors on peut rejeter notre hypothèse nulle d'aucune différence de moyenne entre les populations. Les moyennes des deux groupes sont significativement différentes.

Étape 6 : Citer la taille de l'effet et son intervalle de confiance

Je vous le rappelle encore une fois, il est très important, après avoir déterminé si notre test est significatif ou non de rapporter la taille de l'effet trouvé (ici la différence entre les deux moyennes) et son intervalle de confiance.

Notre taille d'effet est une simple soustraction des deux moyennes, soit 19,27 g - 18,03 g = 1,24 g.

L'intervalle de confiance de cette différence de moyenne quant à lui se calcule à l'aide de l'erreur-type de la différence et d'une valeur extraite de la distribution de T, soit :

$$(\bar{x_1} - \bar{x_2}) \pm t_{0,05}\sqrt{s^2(\frac{1}{n_1} + \frac{1}{n_2})}$$

Encore une fois, vous n'aurez pas à calculer manuellement cet intervalle de confiance, mais comprenez que plus l'erreur-type sera grand, plus l'intervalle de confiance sera large.

Dans un rapport, vous pourriez écrire ce résultat comme ceci :

«Les oiseaux présentaient une différence significative de poids entre les milieux naturels et urbains (Test de T pour variances égales, T_{58} = 4,01, p = 0,00017) L'intervalle de confiance de cette différence se situait entre 0,62 et 1,87 g supplémentaires chez les oiseaux de milieu naturel (IC 95 %).»

14.2. Le test de Welch

Le test de Welch (ou le test de T de Welch ou le test de T à deux échantillons pour variance inégales) s'applique exactement comme le test de T à deux échantillons, à l'exception de la formule de la statistique de T à l'étape 3 qui serait plutôt comme ceci :

$$T = \frac{\bar{x_1} - \bar{x_2}}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Autrement dit, plutôt que d'utiliser un estimé de la variance commun aux deux échantillons, on utilise chacune des deux valeurs individuellement.

Puisque les variances doivent être estimées individuellement, avec potentiellement des degrés de liberté différents pour chacune, les degrés de liberté effectifs du test doivent être calculés avec une formule un peu compliquée[1], qui donnera un nombre réel plutôt qu'un entier.

Notez qu'il n'est pas incorrect de toujours utiliser le test de Welch, que nos variances soient inégales ou non. Mais le test de T à deux échantillons (i.e. pour variances égales) a plus de puissance statistique, donc plus de chances de trouver un effet significatif. Il est donc avantageux d'utiliser ce dernier lorsque les circonstances le permettent.

14.3. Labo : Le test de T à deux échantillons et le test de Welch

Pour ce laboratoire, nous tenterons de répondre à la question : est-ce que le relief de l'habitat affecte la taille des ailes chez les manchots Adélie. Pour les besoins de l'expérience, nous assumerons que les îles Torgersen et Biscoe ont des reliefs très différents, nous permettant de tester notre hypothèse.

Étape 1 :

$$H_0 = \mu_{Torgersen} = \mu_{Biscoe}$$
$$H_1 = \mu_{Torgersen} \neq \mu_{Biscoe}$$

Étape 2 :

[1] https://wikimedia.org/api/rest_v1/media/math/render/svg/2108692a7e5ce58c5bbbc3a34720411b64a1922e

Comme pour le laboratoire du Chapitre 13, nous commencerons par activer les librairies nécessaires et préparer deux mini-tableaux de données contenant uniquement les observations nécessaires pour appliquer notre test :

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(palmerpenguins)

adelie <-
  penguins |>
  filter(species == "Adelie") |>
  filter(island %in% c("Torgersen", "Biscoe")) |>
  drop_na(flipper_length_mm)

torgersen <- adelie |> filter(island == "Torgersen")
biscoe <- adelie |> filter(island == "Biscoe")
```

D'abord effectuer un test de F

Avant de savoir quel test de T utiliser, il faut regarer si la variance est égale ou non entre nos deux échantillons, pour se faire, il faut appliquer un test de F.

Notre première étape consiste donc à explorer les données avant d'appliquer le test de F :

```
adelie |>
  group_by(island) |>
  summarize(
    n = n(),
    moyenne = mean(flipper_length_mm),
    variance = var(flipper_length_mm)
  )
```

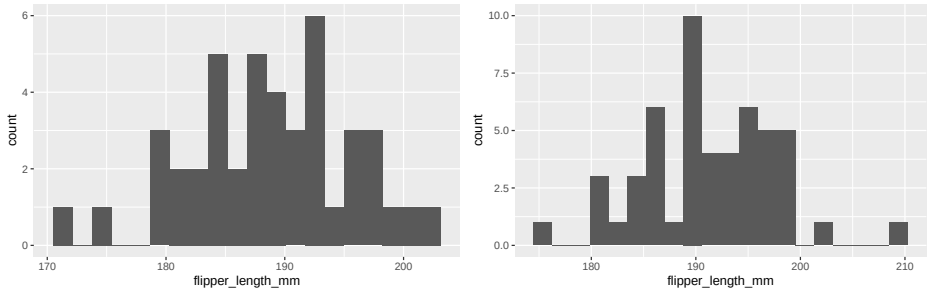
```
# A tibble: 2 x 4
  island        n moyenne variance
  <fct>     <int>   <dbl>    <dbl>
1 Biscoe       44    189.     45.3
2 Torgersen    51    191.     38.8
```

Donc, on a un n raisonnable pour chacune des îles. La variance n'est pas trop différente, mais la moyenne non plus.

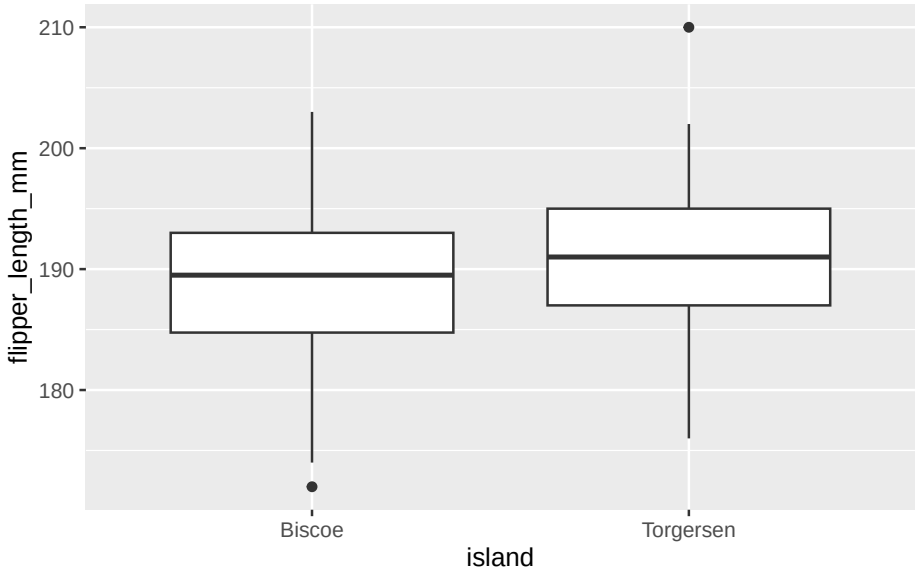
Il est important d'explorer aussi visuellement nos données.

```
biscoe |>
  ggplot() +
  geom_histogram(aes(x = flipper_length_mm), bins = 20)
torgersen |>
  ggplot() +
  geom_histogram(aes(x = flipper_length_mm), bins = 20)
```

Donc, les distributions sont sufisamment normales pour appliquer un test de F et un test de T.




```
adelie |>
  ggplot(aes(island, flipper_length_mm)) +
  geom_boxplot()
```



Visuellement, on ne s'attend pas à trouver de différence de variance (la taille des boîtes et des moustaches est très semblable). Comme vu dans

les résumé numérique, la taille des ailes de manchots sur Torgersen est légèrement plus grande, mais visuellement, elle ne se distingue pas particulièrement.

```
var.test(biscoe$flipper_length_mm,
 ↪  torgersen$flipper_length_mm)
```

```
    F test to compare two variances

data:  biscoe$flipper_length_mm and
torgersen$flipper_length_mm
F = 1.1659, num df = 43, denom df = 50, p-value
= 0.5981
alternative hypothesis: true ratio of variances is not
equal to 1
95 percent confidence interval:
 0.6545884 2.1035135
sample estimates:
ratio of variances
          1.165856
```

Comme notre valeur de p est vraiment plus grande que 0,05, on peut appliquer un test de T à deux échantillons pour variances égales : il n'y a pas de différence de variance entre les deux îles.

Étapes 3 et 4

La fonction pour effectuer le test est la même que le test de T à un échantillon, mais cette fois-ci, il faut lui fournir deux échantillons (eh oui!). Il faut aussi mentionner si on veut que le test s'applique pour des variances égales ou non, avec l'argument `var.equal` que l'on peut mentionner comme `TRUE` ou `FALSE`. Si on indique `TRUE`, R applique le test de T pour variances égales, si on indique `FALSE`, R applique le test de Welch.

```
t.test(torgersen$flipper_length_mm,
 ↪  biscoe$flipper_length_mm, var.equal = TRUE)
```

```
    Two Sample t-test

data:  torgersen$flipper_length_mm and
biscoe$flipper_length_mm
t = 1.8042, df = 93, p-value = 0.07444
alternative hypothesis: true difference in means is not
equal to 0
95 percent confidence interval:
 -0.2416318  5.0428796
sample estimates:
mean of x mean of y
 191.1961  188.7955
```

La première ligne nous informe d'abord de quel test a été utilisé. Comme la fonction `t.test` peut être utilisée pour plusieurs tests différents selon les arguments utilisés, vous comprenez maintenant l'utilité de cette première ligne. La ligne suivante nous rappelle les données utilisées pour le test. La troisième nous fournit la statistique de test (T) ainsi que ses degrés de liberté (df) et la valeur de p associée (ici p=0,074). La ligne suivante nous rappelle quelle est l'hypothèse alternative du test de T. Ensuite, R nous fournit l'intervalle de confiance de la différence de moyenne entre nos deux groupes, soit de -3,66 à 1,93. Autrement dit, l'intervalle de confiance de la différence n'exclut pas la valeur de zéro. Enfin, la dernière ligne nous donne la moyenne de chacun de nos groupes.

Étape 5 :

Comme la valeur de p est plus grande que le seuil de signification (0,07 vs 0,05), notre résultat n'est pas significatif. Il est relativement commun de

trouver une telle différence lorsque les moyennes des deux populations sont égales

Étape 6 :

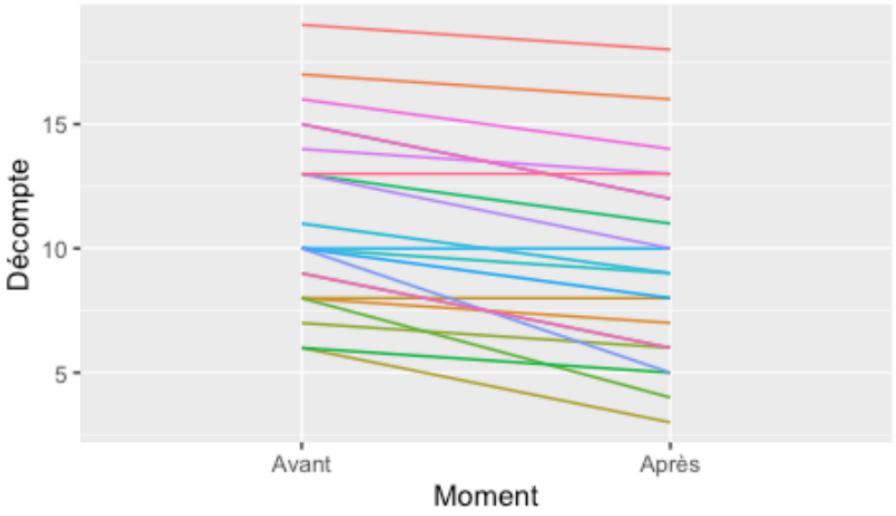
Nous aurions donc pu écrire ce résultat comme ceci : « Il n'y a pas de différence significative dans la taille des ailes des manchots Adélie entre les îles Torgersen et Biscoe (Test de T pour variance égales, T_{93} = 1,80, p = 0,074). L'intervalle de confiance à 95% de la différence entre les deux îles allait de -0,24 à +5,04 mm. ».

14.4. Le test de T pairé

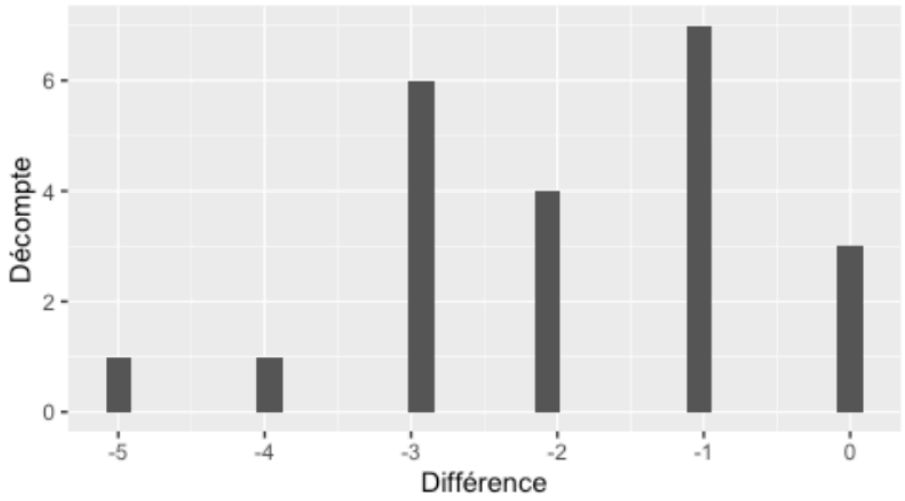
Comme son nom le suggère, le test de T pairé est utilisé lorsque notre expérience est structurée de manière à ce que chacune des mesures dans le premier échantillon corresponde à une mesure dans le deuxième échantillon. Vous verrez aussi parfois le terme "données appariées", ce qui est équivalent.

Ce test est approprié, par exemple, lorsque l'on fait un suivi du poids des individus dans une population, la mesure de départ de chaque individu étant le premier échantillon et la mesure finale devenant le deuxième échantillon. Il peut être aussi utile pour évaluer le résultat d'une expérience où l'on veut voir si un traitement a fonctionné ou non. On pourrait par exemple avoir mesuré le nombre d'insectes sur une série de plantes (n=22), avoir appliqué un traitement insecticide, et vérifié une semaine plus tard le nombre d'insectes présents sur chacune des plantes.

On peut donc s'imaginer, pour chacune des plantes son évolution en nombre d'insectes avant et après le traitement, un peu comme ceci :



Par contre, au moment de calculer le test, on ne s'intéresse pas directement aux valeurs mesurées avant ou après, mais plutôt à la différence entre les deux moments pour chacune des plantes. On aurait donc des valeurs comme -1, 0, -3, +1, etc. pour la différence de nombre d'insectes avant et après. Le test de T pairé regarde si la moyenne de ces différences est différente de zéro ou non. Il faut donc plutôt s'imaginer un graphique comme celui-ci lors que l'on réfléchit au test de T pairé :



On effectue donc essentiellement un test de T à un échantillon basé sur la différence avant-après.

Voyons comment appliquer ce test en se basant sur notre exemple d'insecticide.

Étape 1 : Définir les hypothèses

Comme le test de T pairé s'intéresse à la différence entre le premier et le deuxième échantillon (en pairant les données...), les hypothèses sont aussi structurées de cette façon.

L'hypothèse nulle étant que la différence moyenne est de zéro, et l'hypothèse alternative étant que cette différence est différente de zéro :

$$H_0 : \mu_0 = 0$$
$$H_1 : \mu_0 \neq 0$$

On utilise ici une valeur de référence de zéro par simplicité, mais notez qu'il est souvent plus intéressant d'utiliser une valeur différente, basée

sur votre connaissance scientifique du phénomène à l'étude. Si on avait trouvé une étude qui disait qu'il fallait diminuer d'au moins 10 insectes par plante pour avoir un effet sur sa croissance, il aurait été beaucoup plus intéressant d'utiliser cette valeur comme référence plutôt que zéro. On aurait donc testé si l'insecticide diminue suffisamment le nombre d'insectes pour affecter la croissance de la plante.

Étape 2 : Explorer visuellement les données

La principale assomption du test de T pairé est que la distribution des différences suit une distribution normale. Il faut donc tracer l'histogramme de cette différence pour avoir une idée de la forme de la distribution. Comme nous avons tracé ce graphique à la section précédente, nous ne le retracerons pas ici. En observant rapidement le graphique, on réalise que la moyenne des différences semble bien différente de zéro. On devrait s'attendre à trouver un effet significatif du traitement insecticide, qui ferait diminuer le nombre d'insectes sur les plantes.

Étape 3 : Calculer la statistique de test

Le calcul de la statistique de test est identique à celui du test de T à un échantillon (voir Chapitre 12). Ici la moyenne des différences est de -1,91 insectes, l'écart type est de 1,34 et n=22, donc la statistique de T sera de 6,686 insectes.

Étape 4 : Obtenir la valeur de p Comme pour le test de T à un échantillon, la distribution de la statistique de T devrait suivre une distribution de T de Student avec n-1 degrés de liberté. C'est donc dans cette distribution qu'il faut aller déterminer la valeur de p. Dans notre cas, p= 0,00000012.

Étape 5 : Rejeter ou non l'hypothèse nulle

Comme observer une telle différence est très rare lorsque l'on échantillonne des données telles que stipulées avec l'hypothèse nulle (0,00000012 est vraiment plus petit que 0,05), on rejette l'hypothèse

nulle d'une différence de zéro. On peut donc dire que l'insecticide a un effet significatif sur le nombre d'insectes.

Étape 6 : Citer la taille de l'effet et son intervalle de confiance.

Pour le test de T pairé, la taille de l'effet est l'ampleur de la différence entre nos deux échantillons. Ici, nous avons calculé que cette différence est en moyenne de -1,91 insectes. Avec le même calcul que pour le test de T à un échantillon, on peut calculer l'intervalle de confiance à 95 % de cette différence, qui se situera entre -2,50 et -1,31 insectes.

On peut enfin écrire notre résultat comme ceci : « L'effet moyen de l'insecticide était de -1,91 insecte par plante ± 0,59 (I.C. 95 %), ce qui en fait une différence significativement différente de zéro (t_{22} = 36,686, p = 0,00000012). »

Labo : Le test de T pairé

Puisque le tableau de données `penguins` ne contient pas de données pairées, nous allons devoir, pour cet exemple, se créer un petit tableau permettant d'étudier le gain de poids d'une espèce de bruant à la fin de l'été en préparation pour la migration. Notre question écologique étant de savoir si les bruants prennent effectivement du poids avant la migration ou non.

Nous allons donc activer nos librairies, puis créer manuellement notre tableau de données, à l'aide du code suivant :

```
engraissement <- data.frame(
  poids_debut_g = c(7.1, 3.0, 6.0, 4.0, 4.2),
  poids_fin_g =   c(8.2, 3.5, 5.6, 4.5, 6.1)
) |>
  mutate(difference = poids_fin_g - poids_debut_g)

engraissement
```

```
  poids_debut_g poids_fin_g difference
1           7.1         8.2        1.1
2           3.0         3.5        0.5
3           6.0         5.6       -0.4
4           4.0         4.5        0.5
5           4.2         6.1        1.9
```

Remarquez que dans la même étape où nous créons notre tableau de données, nous créons immédiatement une colonne de différence. Cela facilitera notre travail plus tard.

Étape 1 :

$$H_0 : \mu_{preparation} = 0$$
$$H_1 : \mu_{preparation} \neq 0$$

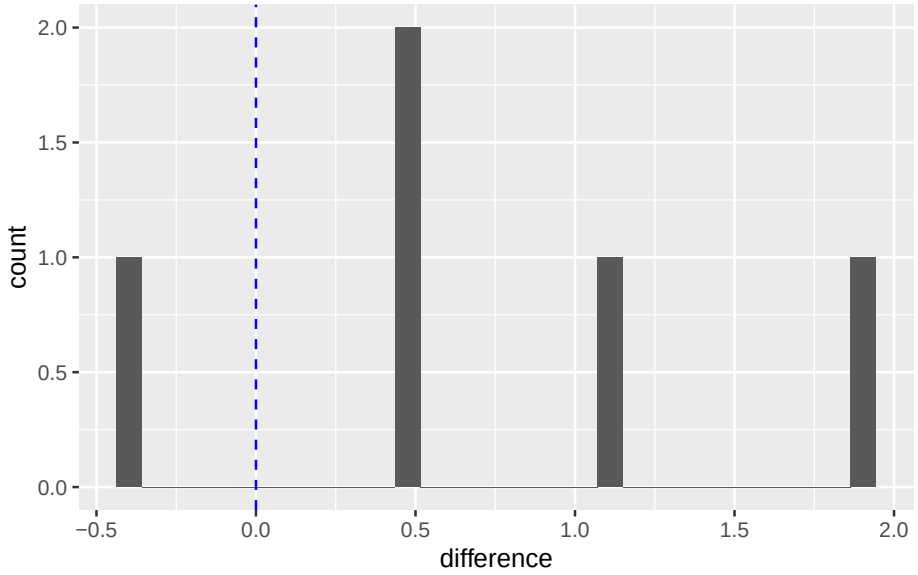
Autrement dit, est-ce que la moyenne de gain de poids durant la préparation à la migration est différente de zéro.

Étape 2 :

Pour le test de T pairé, on peut en un seul graphique valider l'assomption de normalité de la différence, et aussi visualiser la taille de l'effet. Pour se faire, nous créons un histogramme, auquel nous ajoutons une couche de ligne verticale, permettant de voir la valeur de référence (dans notre cas, zéro) :

```
engraissement |>
  ggplot(aes(x = difference)) +
  geom_histogram() +
  geom_vline(xintercept = 0, color = "blue", linetype =
   ↪  "dashed")
```

```
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```



On peut donc constater d'un seul coup d'oeil que nos données semblent normales (pour le peu que l'on en a) et que le poids des bruants, à une exception près, semble avoir augmenté durant la période étudiée.

Étapes 3 et 4 :

Pour calculer le test de T pairé dans R, on utilise la fonction `t.test` exactement comme pour le test de T à un échantillon :

```
t.test(engraissement$difference)
```

```
    One Sample t-test

data:  engraissement$difference
t = 1.8947, df = 4, p-value = 0.131
alternative hypothesis: true mean is not equal to 0
```

```
95 percent confidence interval:
 -0.3350491  1.7750491
sample estimates:
mean of x
     0.72
```

R nous mentionne qu'il a effectué un test de T à un échantillon. C'est correct, c'est ce que nous voulions. Il nous fournit ensuite les mêmes informations qu'au Chapitre 12, mais c'est à nous de les interpréter comme étant la moyenne des différences. Le test nous informe donc que la moyenne des différences de poids pendant la préparation à la migration est de 0,72 g.

Étape 5 :

La sortie de R nous informe aussi que cette différence n'est pas significative au seuil de 0,05 (p=0,131). Il important de nuancer ici le fait que bien l'on trouve une différence claire dans nos données, notre taille d'échantillon est tellement faible (n=5) qu'il serait à peu près impossible de trouver une valeur de p significative comme tel : nous manquons clairement de puissance statistique.

Étape 6 : Nous pourrions donc écrire ce résultat comme ceci : « Le gain de poids moyen des bruants en préparation de la migration était en moyenne de 0,72 g ± 1,06 (I.C. 95 %). Cette différence n'était pas significativement différente de zéro (t_4=1,894, p = 0,131) »

Notez qu'il existe aussi une façon de calculer le test de T pairé sans calculer à l'avance une colonne de différence. Il faut à ce moment passer les deux colonnes à la fonction `t.test` et lui mentionner de nous calculer un test pairé :

```
t.test(engraissement$poids_debut_g,
 ↪  engraissement$poids_fin_g, paired = TRUE)
```

On obtient alors exactement le même résultat :

```
    Paired t-test

data:  engraissement$poids_debut_g and
engraissement$poids_fin_g
t = -1.8947, df = 4, p-value = 0.131
alternative hypothesis: true mean difference is not
equal to 0
95 percent confidence interval:
 -1.7750491  0.3350491
sample estimates:
mean difference
          -0.72
```

14.5. Récapitulatif

Si on fait un petit récapitulatif des tests vus jusqu'à présent, vous êtes maintenant en mesure de gérer les situations suivantes :

- Pour comparer une moyenne à une valeur cible : **Test de T à un échantillon**
- Pour comparer deux moyennes entre-elles : **Test de T à deux échantillons** ou **test de Welch**, selon que la variance est égale ou non.
- Pour comparer la moyenne de données pairées : **Test de T pairé**
- Pour comparer la variance de deux échantillons : **Test de F**.

15. Tests de comparaison de 3+ moyennes

15.1. Différentes façons d'organiser les mêmes données

Les tests statistiques présentés dans les chapitres précédents (le test de F et la famille des tests de T) étaient tous structurés de la même manière, soit que les deux échantillons à comparer étaient soit dans deux tableaux de données différents, ou soit dans deux colonnes différentes du même tableau. Les tests étaient pensés pour **comparer deux variables quantitatives**.

Lorsque l'on a affaire à plus de deux groupes, cette façon de fonctionner peut devenir lourde à gérer. C'est pourquoi normalement, on discute plutôt des tests de comparaison de 3+ moyennes comme **mettant en relation une variable qualitative et une variable quantitative**. Il s'agit en fait des mêmes données, mais présentées différemment.

Par exemple, au Chapitre 14, notre jeu de données concernant la question sur le poids des oiseaux des milieux urbains et naturels aurait pu être présentée en deux tableaux comme ceci :

Par contre, les mêmes observations auraient aussi pu être organisées comme ceci :

Oiseaux en milieu naturel

date	poids_g
2020-09-09	12,5
2020-09-09	13,4
2020-09-10	12,1
...	...

Oiseaux en milieu urbain

date	poids_g
2020-08-01	10,3
2020-08-03	12,6
2020-08-03	9,5
...	...

milieu	date	poids_g
Naturel	2020-09-09	12,5
Naturel	2020-09-09	13,4
Naturel	2020-09-10	12,1
Urbain	2020-08-01	10,3
Urbain	2020-08-03	12,6
Urbain	2020-08-03	9,5
...	...	...

Prenez quelques minutes pour bien comprendre comment on est passé d'un format à l'autre. Il est crucial que cette étape soit claire dans votre tête avant de passer à la suite de ce chapitre.

Ce changement de format de données change aussi la question associée. Plutôt que de se demander si les milieux urbains et naturels sont différents, on se demande maintenant comment le type de milieu influence le poids des oiseaux. Plutôt que de comparer deux variables quantitatives ($\text{poids}_{\text{naturel}}$ et $\text{poids}_{\text{urbain}}$), on met en relation une variable quantitative (poids) et une variable qualitative (milieu).

C'est dans ce dernier format que devront être organisées vos données pour faire bien fonctionner les test du présent chapitre dans le logiciel R. Vous avez évidemment le droit de réfléchir à votre problème comme la comparaison de 3+ variables quantitatives, mais pour le logiciel, vous

devrez traduire votre pensée en mettant en relation une variable qualitative et une quantitative.

Ce changement d'organisation des données entraîne aussi une nouvelle terminologie. Maintenant que nous pensons nos données en termes de cause et d'effet, nous parlerons fréquemment de variable expliquée et de variable explicative. Dans nos modèles statistiques, la **variable expliquée** est toujours celle que nous cherchons à comprendre pourquoi elle varie, alors que la **variable explicative** pourrait fournir une explication pour cette variation. En termes de cause-effet, on s'attend à ce que la variable explicative soit la cause. Dans l'analyse de variance enseignée dans le présent chapitre, la variable expliquée est la variable quantitative et la variable explicative est la variable qualitative.

Vous verrez aussi parfois les appellations de variables **dépendante** (i.e. expliquée) et **indépendante** (i.e. explicative). Les mathématiciens privilégient souvent ces appellations, mais elles sont un peu contre-intuitives pour les praticiens. Je n'utiliserai donc pas ces termes dans mon cours.

15.2. Analyser la variance

Le test statistique que nous verrons dans ce chapitre se nomme l'analyse de variance, et on le désigne habituellement sous son acronyme anglais d'ANOVA (*ANalysis Of VAriance*). Son but est d'aller découper la variance d'une variable quantitative entre deux parties : une que l'on peut attribuer à la variable explicative, et une qui est la variation normale entre les individus (i.e. que l'on ne peut pas expliquer).

Si on reprend notre exemple des différences de poids des oiseaux entre les types de milieux, aucun de nos oiseaux ne pèsera exactement la

même chose que la moyenne. Ils seront tous soit au-dessus ou soit en-dessous. Au Chapitre 5, nous avons donné un nom à cette variation autour de la moyenne : la variance.

Cette variance peut être due à plusieurs choses : différences génétiques entre les individus, âge différent, sexe différent, une partie de hasard, mais aussi une partie due au fait qu'ils étaient dans des milieux différents.

L'ANOVA permet d'aller évaluer la partie de la variation que l'on peut attribuer à notre variable qualitative (le milieu). Elle évalue ensuite si cette partie est grande par rapport à la variation normale entre les individus (celle dûe aux facteurs que nous n'avons pas modélisés comme l'âge, le sexe, etc.). Cette variation normale peut être nommée de beaucoup de façons différentes, entre autres vous verrez le terme **variation intra-groupe**, **bruit** et aussi **résidus** pour la désigner.

L'ANOVA, au final, est un simple test de F (voir Chapitre 13), qui compare la **variation inter-groupe** (celle due à notre variable qualitative) à la variation intra-groupe (le bruit). Elle compare ensuite la valeur de F obtenue à celle que nous aurions pu obtenir si la variable qualitative n'avait aucun effet.

Maintenant que ces principes sont (je l'espère) un peu plus clairs, nous pouvons définir l'ANOVA de façon plus formelle.

15.3. L'ANOVA à un facteur

L'ANOVA à un facteur s'utilise pour comparer deux moyennes ou plus entre elles. Comme expliqué précédemment, elle peut être aussi vue comme un test de la relation entre une variable qualitative et une variable quantitative. Pour illustrer ce test, nous utiliserons à nouveau notre exemple du poids des oiseaux dans différents milieux. Cette fois, nous avons capturé 20 oiseaux en milieu forestier, 20 oiseaux en milieu

urbain et 20 oiseaux en milieu agricole. Nous désirons savoir s'il existe un lien entre le poids des oiseaux et le milieu dans lequel ils vivent.

Étape 1 : Définir les hypothèses

L'hypothèse nulle pour ce test statistique est que les moyennes des différents groupes sont toutes égales entre elles. L'hypothèse alternative est qu'au moins une des moyennes est différente des autres.

$$H_0 : \mu_{urbain} = \mu_{agricole} = \mu_{forestier}$$

$$H_1 : \text{Au moins un des milieux est différent des autres}$$

Étape 2 : Explorer visuellement les données

L'ANOVA à un facteur comporte trois assomptions principales. La première, comme pour tous les autres tests, est que les observations sont indépendantes les unes des autres. Si jamais elles ne le sont pas, il faut utiliser un autre type d'ANOVA, plus approprié (imbriquée, en blocs aléatoires, etc., voir Chapitre 16).

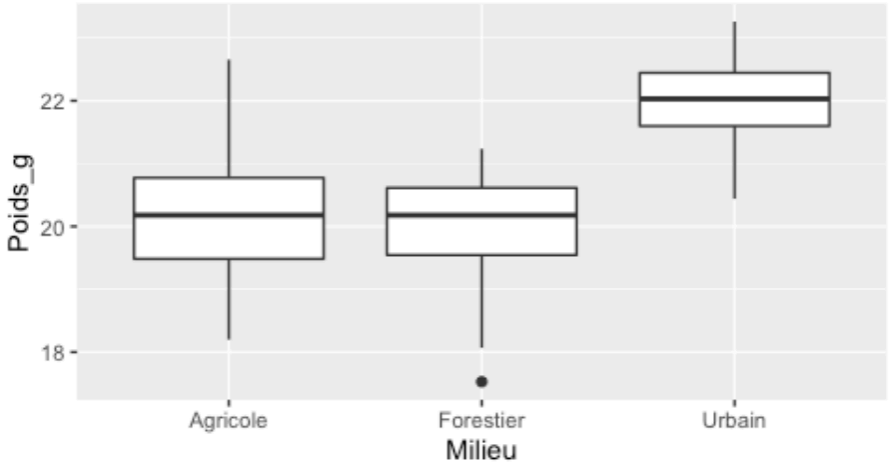
La deuxième assomption est que la variance entre les groupes est égale. Comme nous avons vu au Chapitre 13, on peut, pour tester cette assomption, effectuer un test de F entre le groupe ayant la plus grande variance et le groupe ayant la plus petite. Nous passerons par dessus cette étape ici par souci de brièveté, mais lors de vos travaux ou rapports, vous devez absolument l'effectuer.

La troisième assomption est celle de normalité de la variable quantitative. Ici par contre, il faut être bien attentif lorsque l'on vérifie cette assomption. Il est normal que, avant de débuter l'analyse, la variable quantitative présente plusieurs modes plutôt qu'un seul. L'assomption officielle du test est que **chaque groupe suit une distribution normale**.

On peut donc soit vérifier cette assomption groupe par groupe avant de commencer, ou la vérifier après avoir calculé le test, en vérifiant ce que l'on appelle la **normalité des résidus**. Autrement dit, une fois les différences de moyennes éliminées entre les groupes, ce qui reste devrait être distribué normalement. Je vous enseignerai cette deuxième façon de faire, puisqu'elle sera aussi applicable dans les chapitres suivants pour la régression linéaire, l'ANCOVA et les modèles plus complexes que nous verrons dans la section sur le modèle linéaire.

Il faudra donc penser, après avoir calculé le test, d'aller vérifier les résidus avant de pouvoir procéder à l'interprétation du test. Nous dérogerons donc légèrement des 6 étapes classiques montrées jusqu'à présent pour appliquer un test.

Une fois toutes ces choses vérifiées, il nous reste une chose à faire, qui est d'explorer nos données pour voir si on devrait s'attendre à trouver des différences entre les groupes ou non. La façon classique de le faire est à l'aide d'un diagramme à moustaches, comme ceci :



Après avoir observé ce graphique, on peut s'attendre à ce que l'assomption d'égalité des variances ait été respectée et qu'il existe une

différence de moyenne de poids entre certains groupes, les oiseaux urbains paraissant plus lourds que les oiseaux des autres milieux.

Vous lirez aussi parfois que l'ANOVA nécessite que le nombre d'observations (le n) soit égal entre les groupes. Des tailles d'échantillons différentes n'influenceront pas le calcul ou les résultats de l'ANOVA à un facteur comme nous voyons ici. Ils peuvent cependant compliquer les choses dans les ANOVA plus complexes. On recommande néanmoins d'avoir au moins trois observations pour chacun des groupes.

Étape 3 : Calculer la statistique de test

Contrairement aux tests présentés dans les chapitres précédents, le calcul de l'ANOVA n'est pas aussi direct et nécessite quelques étapes préliminaires avant d'en arriver à la statistique de test. Avant d'en arriver à notre ratio de variance, il faudra commencer par définir le calcul de la variance inter-groupe et de la variance intra-groupe. Pour cela, rappelons-nous d'abord la définition de la variance, telle que vue au Chapitre 5 :

$$\sigma^2 = \sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n-1}$$

Notez que cette formule pourrait être transformée comme ceci et l'on obtiendrait exactement le même résultat :

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}$$

Dans cette deuxième formulation, on clarifie que la formule comporte deux parties distinctes, qui ont chacune un nom et une utilité. La partie au numérateur se nomme la **somme des carrés**. On peut y voir que pour chaque valeur (x_i), on calcule le carré de la différence entre cette valeur et la moyenne. Le dénominateur se nomme quant à lui les degrés de liberté.

Il existe des définitions équivalentes pour la variance inter-groupe et la variance intra-groupe. Cependant, puisqu'il ne s'agit pas de variance à proprement parler, on parle plutôt de **carrés moyens** et on leur attribue généralement leur abréviation anglaise, soit MS pour *mean square*. Les formules sont les suivantes :

$$MS_{inter-groupe} = SS_{inter-groupe}/(k-1)$$

$$MS_{intra-groupe} = SS_{intra-groupe}/k(n-1)$$

Où k est le nombre de niveaux de la variable qualitative (i.e. le nombre de groupes différents) et n le nombre d'échantillons par combinaison de traitement.

Dans ces équations, il nous reste donc à définir $SS_{\text{inter-groupe}}$ et $SS_{\text{intra-groupe}}$, qui sont des sommes des carrés (SS, *Sum of Squares*).

La **somme des carrés inter-groupes** est sans doute la chose la plus abstraite à calculer dans l'ANOVA. On la calcule en remplaçant chaque observation de la variable quantitative par la moyenne de son groupe et en faisant le calcul de la somme des carrés sur ces nouvelles valeurs. Le principe sous-jacent étant que, si les groupes sont très différents les uns des autres, la somme des carrés inter-groupes sera grande. Si les groupes sont très semblables, cette somme des carrés inter-groupes sera très petite.

Enfin, pour calculer la **somme des carrés intra-groupes**, que l'on peut aussi nommer résidus, il faut faire la différence entre chaque observation et la moyenne de son groupe et mettre ces différences au carré avant de les additionner. Plus les mesures dans un groupe sont différentes les unes des autres, plus cette somme des carrés sera grande. Au contraire, si toutes les observations d'un groupe sont égales, cette somme des carrés sera de zéro.

On peut donc ensuite calculer les carrés moyens, comme mentionné ci-haut, et finalement (!) calculer la statistique de F comme étant le ratio

des carrés moyens inter et intra groupe, autrement dit :

$$F = MS_{inter-groupe}/MS_{intra-groupe}$$

Dans le résultat d'une ANOVA, ces informations sont habituellement rassemblées dans un tableau que l'on présente avec les résultats :

Source	Degrés de liberté	Somme des carrés	Carrés moyens	Valeur de F	Valeur de p
Inter-groupe	2	48,88	24,44	25,45	0,0000000013
Intra-groupe(résidus)	57	54,75	0,96		

Comme d'habitude, vous n'aurez pas à faire tout ce calcul en détail. R vous fournira automatiquement le tableau de l'ANOVA comme ci-haut. Cependant, il est important de comprendre que la valeur de F augmentera si les différences entre les moyennes sont grandes, et diminuera si les observations dans un groupe sont très différentes les unes des autres.

Étape 4 : Obtenir la valeur de p

Une fois la valeur de F calculée, nous pouvons aller voir, comme à l'habitude, dans la distribution F quelle serait la probabilité d'avoir trouvé un ratio aussi important si l'hypothèse nulle d'aucune différence entre les groupes était vraie.

Si l'on se rappelle bien, la distribution de F nécessite deux degrés de liberté différents, ceux qui numérateur et ceux du dénominateur. Ici, ils doivent être respectivement de k - 1 et de n - k. Pour notre exemple, notre valeur de F est de 25,45 et nos degrés de liberté sont de 2 et 57, ce qui nous donne une valeur de p de 0,0000000013

Étape 5 : Rejeter ou non l'hypothèse nulle

Comme observer un tel ratio de F serait extrêmement rare si notre hypothèse nulle était vraie, nous pouvons affirmer que nous avons trouvé une différence significative et rejeter l'hypothèse nulle.

Étape 6 : Citer la taille de l'effet et son intervalle de confiance

Si nous avions à citer une intervalle de confiance pour le ratio de F, le calcul serait identique à celui montré au Chapitre 13. Cependant, puisque ce ratio comme tel ne présente que peu d'intérêt biologique, son intervalle de confiance est rarement rapporté.

Nous pourrions donc rapporter ce résultat comme ceci : «Le type milieu de vie influençait de façon significative le poids des oiseaux observés ($F_{2,57}$=25,45, p = 0,0000000013) »

Néanmoins, une question reste en suspens à ce point. Nous avons pu rejeter l'hypothèse nulle que toutes les moyennes étaient égales. Nous savons que le milieu a un effet sur le poids des oiseaux. Cependant, nous ne savons pas à ce point quels milieux sont significativement différents les uns des autres. Nous verrons plus loin dans ce chapitre comment répondre à cette deuxième question.

15.4. Labo : L'ANOVA à un facteur

Pour ce laboratoire, nous irons vers une question un peu plus du côté de la biologie évolutive. Nous nous demanderons si la taille des ailes est différente entre les espèces de manchots ou si ce n'est pas un caractère distinctif.

Étape 1 :

$$H_0 : \mu_{\text{Adélie}} = \mu_{\text{Chinstrap}} = \mu_{\text{Gentoo}}$$

$$H_1 : \text{Au moins deux espèces ont}$$

des tailles d'ailes différentes.

Étape 2 :

Nous préparons d'abord les librairies de code dont nous aurons besoin, puis nous préparerons le tableau de données approprié pour notre test. Comme nous savons que pour certaines observations nous ne connaissons pas la longueur des ailes, il faut prendre le soin d'éliminer ces lignes avant d'effectuer notre analyse :

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(palmerpenguins)

pour_anova <- penguins |>
  drop_na(flipper_length_mm)
```

Puisque l'ANOVA assume que les variances sont égales entre les groupes, il faudra d'abord appliquer un test de F entre la plus petite et la plus grande variance pour s'assurer de pouvoir procéder.

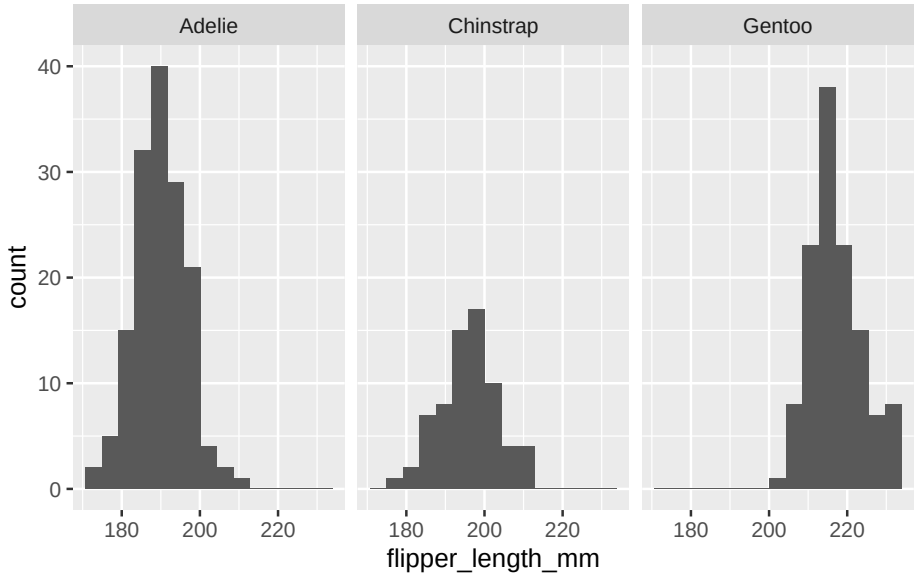
```
pour_anova |>
  group_by(species) |>
  summarize(
    n = n(),
    moyenne = mean(flipper_length_mm),
    variance = var(flipper_length_mm)
  )
```

```
# A tibble: 3 x 4
  species       n moyenne variance
  <fct>     <int>   <dbl>    <dbl>
1 Adelie      151    190.     42.8
2 Chinstrap    68    196.     50.9
3 Gentoo      123    217.     42.1
```

Les deux variances les plus extrêmes sont Gentoo et les Chinstrap. Il faudra donc tester si leurs variances sont significativement différentes.

Commençons tout d'abord par vérifier la distribution des tailles d'ailes des différentes espèces.

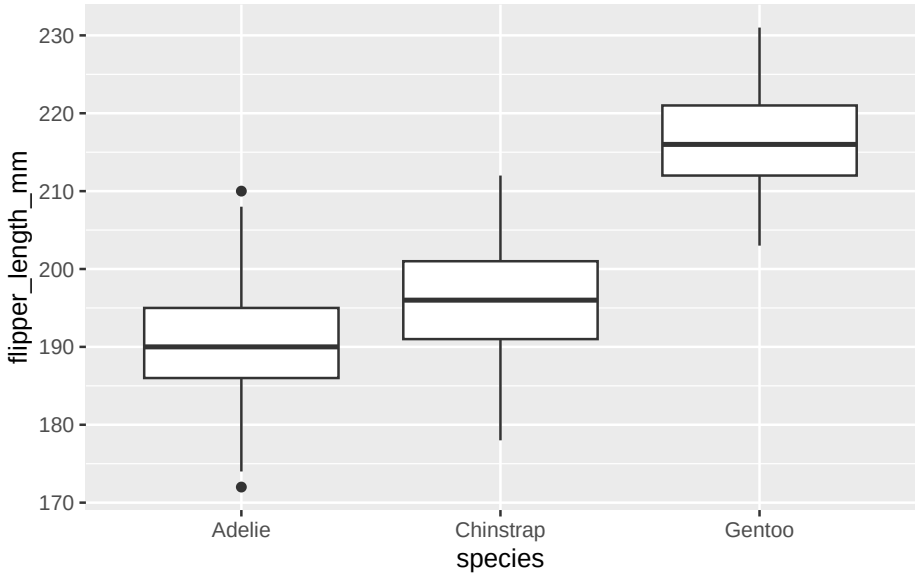
```
pour_anova |>
  ggplot(aes(flipper_length_mm)) +
  geom_histogram(bins = 15) +
  facet_wrap(~species)
```



À première vue, les 3 groupes sont distribués normalement.

Par la suite, nous pouvons explorer visuellement nos données pour voir si on s'attend à trouver des différences de moyennes ou de variances :

```
pour_anova |>
  ggplot(aes(x = species, y = flipper_length_mm)) +
  geom_boxplot()
```



À première vue les variances sont relativement homogènes (taille des boîtes très semblable), mais les moyennes diffèrent beaucoup, en particulier pour les Gentoo qui semblent avoir des ailes vraiment plus grandes.

```
chinstrap <- pour_anova |> filter(species ==
↪  "Chinstrap")
gentoo <- pour_anova |> filter(species == "Gentoo")
var.test(chinstrap$flipper_length_mm,
↪  gentoo$flipper_length_mm)
```

```
    F test to compare two variances

data:  chinstrap$flipper_length_mm and
gentoo$flipper_length_mm
```

```
F = 1.2095, num df = 67, denom df = 122, p-value
= 0.3626
alternative hypothesis: true ratio of variances is not
equal to 1
95 percent confidence interval:
 0.8017046 1.8736162
sample estimates:
ratio of variances
          1.209464
```

Il n'y a pas de différence significative entre les variances des différentes espèces. On peut donc procéder à l'ANOVA à proprement parler.

Étapes 3 et 4 :

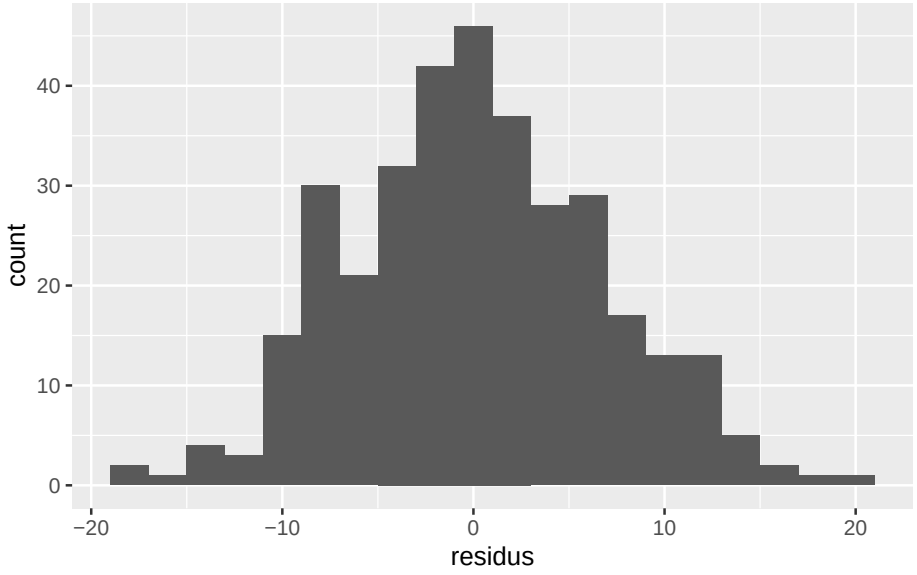
La fonction pour calculer une ANOVA dans R se nomme `aov` (pour *Analysis Of Variance*). Par contre, sa syntaxe est un peu différente des tests que nous avons vus jusqu'à présent. Plutôt que de donner les valeurs de chacun des groupes dans des arguments séparés, il faut utiliser la notation formule de R. Dans cette notation, il faut indiquer d'abord la variable quantitative que nous voulons analyser, puis le symbole `~` (tilde) et ensuite la variable qualitative. Ensuite, il faut utiliser l'argument `data` pour nommer dans quel tableau de données aller chercher ces variables. Pour notre exemple, la commande serait donc :

```
m <- aov(flipper_length_mm ~ species, data = pour_anova)
```

Notez que pour faciliter notre travail dans les étapes subséquentes, on attrape la sortie de l'ANOVA dans un objet nommé `m` plutôt que de l'afficher directement à l'écran. Remarquez aussi que (et c'est vrai pour toutes les formules dans R) la variable expliquée se trouve toujours à gauche du ~ et la (ou les) variable explicative se trouve à droite. Dans votre tête, vous pouvez lire le "~" comme "en fonction de".

La première chose à faire à ce point est de valider la distribution des résidus pour s'assurer qu'ils sont normaux. Pour se faire, le plus simple est d'ajouter une colonne résidus à notre tableau de données à l'aide de la fonction `resid` et ensuite de l'utiliser pour tracer un histogramme :

```
pour_anova <- pour_anova |>
  mutate(residus = resid(m))

pour_anova |>
  ggplot(aes(x = residus)) +
  geom_histogram(bins = 20)
```



Difficile de faire mieux pour la normalité. Remarquez cependant qu'il est pas crucial que les données soient parfaitement normale. L'ANOVA est relativement robuste aux écarts de normalité, mais elle est relativement sensible aux différences de variance entre les groupes.

Si on appelle la fonction `summary` sur cet objet `m`, obtient le tableau d'ANOVA tel que présenté dans l'exemple précédent :

```
summary(m)
```

```
             Df Sum Sq Mean Sq F value Pr(>F)
species       2  52473   26237   594.8 <2e-16 ***
Residuals   339  14953      44
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Tous les éléments mentionnés précédemment s'y trouvent, mais en anglais. L'abréviation Df désigne les degrés de liberté (*Degrees of Freedom*), Sum Sq désigne la somme des carrés, Mean Sq les carrés moyens, F value la valeur de F est Pr la valeur de p. La première ligne (species) désigne la variance inter-groupe et le terme Residuals désigne la variance intra-groupe.

Notez qu'à côté de la valeur de p, si R nous présente une ou plusieurs étoiles (*), c'est une indication supplémentaire que la différence est significative, comme l'indiquent les codes à la fin du résumé.

Étapes 5 et 6 :

On pourrait donc à ce point écrire notre résultat comme ceci : «Les espèces de manchots de l'archipel Palmer avaient des ailes de taille significativement différentes ($F_{2,339}$=594,8, $p < 2 \times 10^{-16}$).»

15.5. Contenu optionnel : ANOVA à un facteur vs. test de T?

À ce point, une question vous vient peut-être en tête. Si ma variable qualitative ne contient que deux groupes, est-ce que je peux quand même utiliser une ANOVA ou je dois absolument utiliser un test de T?

La réponse est que, si les variances sont égales entre vos deux groupes, vous pouvez utiliser une méthode ou l'autre et vous arriverez exactement à la même valeur de p. Ceci est tellement vrai qu'il existe une règle pour convertir une valeur de F en valeur de t et vice-versa, qui dit que $\mathbf{F^2 = t}$.

Par contre, si vos variances sont inégales entre vos groupes, les résultats peuvent différer, puisque bien que l'ANOVA soit robuste à un certain départ de cette condition, elle n'est pas aussi bien adaptée que le test de Welch.

Il existe d'autres petites équivalences de ce genre à découvrir, mais je vais me garder un peu de matériel pour la deuxième partie de ce livre!

15.6. Les tests post hoc

Comme mentionné plus tôt, l'ANOVA peut nous renseigner globalement sur l'effet de notre variable qualitative, mais elle ne peut pas nous informer de quels groupes sont différents les uns des autres. Si l'on désire obtenir ces informations, il faut effectuer un test post hoc, c'est-à-dire après-coup.

La chose extrêmement importante à retenir est que vous devez **toujours effectuer l'ANOVA globale d'abord**. Ensuite, si et seulement si l'ANOVA vous montre une différence significative, vous pouvez lancer les tests post hoc. Il ne faut jamais aller directement avec ces derniers. La raison est fort simple : comme les tests post hoc effectuent beaucoup de

comparaisons, ils augmentent potentiellement votre erreur de type I. Par souci de parcimonie, il faut donc se contenter de les utiliser uniquement au moment approprié.

La première façon de déterminer quels groupes sont différents les uns des autres est d'utiliser de **multiples tests de T**. Pour notre exemple sur les oiseaux, on ferait un test de T entre urbain et forestier, ensuite entre forestier et agricole et finalement entre agricole et urbain. Pour contrer le problème de la multiplication des tests et le gonflement associé de l'erreur de type I, on ajoute généralement à ces tests la correction de Bonferroni. La **correction de Bonferroni** consiste à modifier notre seuil de signification, pour qu'il revienne globalement au seuil attendu.

Par exemple, si nous avions prévu utiliser le seuil classique de $\alpha = 0{,}05$ et que nous voulions faire 3 comparaisons multiples, notre nouveau seuil de signification ajusté par la correction serait de $0{,}05/3 = 0{,}0167$. Donc pour chacun de nos tests de T, la valeur de p devrait être $< 0{,}0167$ pour être considérée comme significative.

Bien que mathématiquement correcte, la correction de Bonferroni est souvent critiquée parce que le nouveau seuil de signification est particulièrement difficile à atteindre, ce qui réduit grandement la puissance de notre procédure statistique.

C'est pourquoi on utilise en général le test de **Tukey HSD** pour effectuer ces comparaisons. Nous ne verrons pas les détails techniques de ce test. L'important est de comprendre que le test calcule une distance seuil (HSD; *Honest Significant Difference*) puis compare la distance entre chaque paire à ce seuil. Il corrige ainsi pour l'augmentation de l'erreur de type I, au même titre que la correction de Bonferroni, mais diminue beaucoup moins la puissance de notre procédure.

15.7. Labo : Le test post-hoc de Tukey HSD

Pour appliquer un test de Tukey HSD dans R, il faut avoir préalablement effectué une ANOVA et avoir conservé le résultat dans un objet.

Comme discuté plus haut, il est primordial d'appliquer les test post-hoc uniquement si l'ANOVA globale indiquait une différence significative entre les groupes.

```
TukeyHSD(m)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = flipper_length_mm ~ species, data =
pour_anova)

$species
                     diff       lwr       upr p adj
Chinstrap-Adelie  5.869887  3.586583  8.153191     0
Gentoo-Adelie    27.233349 25.334376 29.132323     0
Gentoo-Chinstrap 21.363462 19.000841 23.726084     0
```

R nous informe d'abord sur le test appliqué et nous rappelle les détails du modèle d'ANOVA que nous avons utilisé. Ensuite, il nous fournit un tableau des comparaisons par paires qu'il a effectuées.

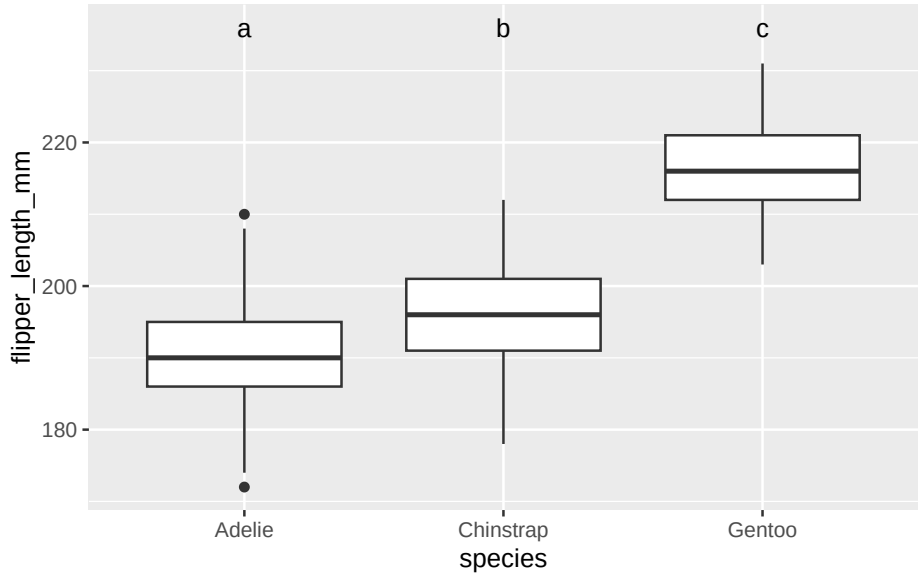
Ce tableau contient trois lignes, qui nous montrent les différences entre Gentoo et Adélie, ensuite Gentoo et Chinstrap, et enfin entre Gentoo et Chinstrap. Pour chacune des paires, le test de Tukey nous fournit la différence moyenne entre les groupes (diff) ainsi que les bornes de l'intervalle de confiance à 95 % de cette différence (lwr et upr) et finalement la valeur de p ajustée de cette différence.

On voit ici que la taille des ailes est significativement différente dans toutes les paires de combinaison. La valeur de p de zéro est une approximation. R veut vraiment nous dire que la valeur est si petite que lui n'arrive pas à la distinguer de 0.

On voit parfois les résultats d'un test de Tukey HSD résumés ainsi : Gentoo > Chinstrap > Adélie. On place les groupes en ordre de grandeur, mais on indique le symbole > uniquement pour séparer les groupes qui sont significativement différents. On relie par un trait souligné les termes qui ne sont pas différents les uns des autres.

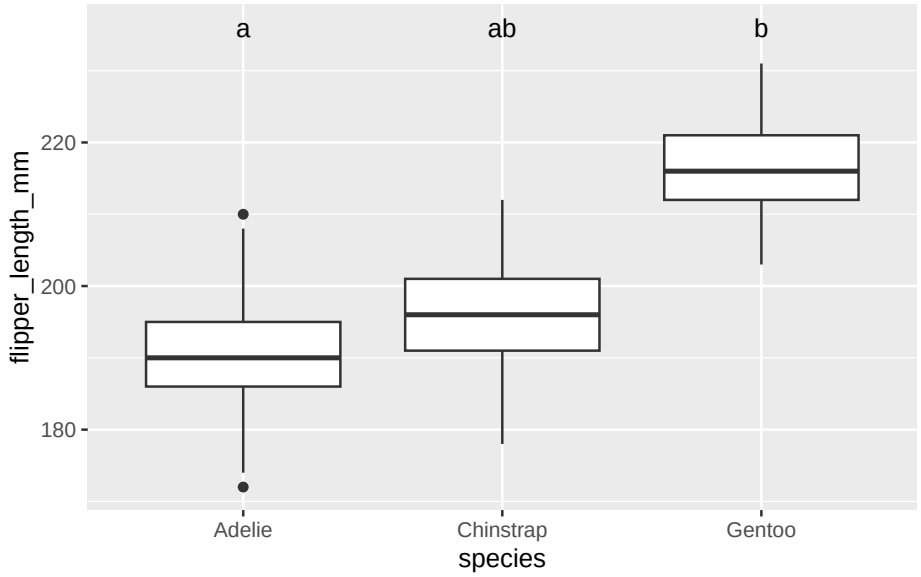
Visuellement, cette information peut-être représentée à l'aide d'une lettre pour chacun des groupe significativement différent d'un autre groupe, comme ceci :

```
etiquettes <-
  data.frame(
    x = c('Adelie','Chinstrap','Gentoo'),
    y = max(pour_anova$flipper_length_mm)+5,
    lettre = c('a','b','c')
  )

pour_anova |>
  ggplot(aes(species, flipper_length_mm)) +
  geom_boxplot()+
  geom_text(data = etiquettes, aes(x = x, y = y,label =
   ↪  lettre))
```



Si le manchot Adélie avait été différent du Gentoo, mais que ni le Gentoo ni le Adélie avait été différent du manchot Chinstrap, les lettres auraient été comme ceci :

```
etiquettes <-
  data.frame(
    x = c('Adelie','Chinstrap','Gentoo'),
    y = max(pour_anova$flipper_length_mm)+5,
    lettre = c('a','ab','b')
  )

pour_anova |>
  ggplot(aes(species, flipper_length_mm)) +
  geom_boxplot()+
  geom_text(data = etiquettes, aes(x = x, y = y,label =
   ↪  lettre))
```



Dans ce genre de graphique, il peut donc arriver qu'un même groupe possède plusieurs lettres, c'est tout à fait normal.

15.8. Exercice : L'ANOVA et le test de Tukey HSD

À partir de la base de données ChickWeight inclue avec R, filtrez les données pour ne conserver que le poids des poussins au dernier jour de de l'expérience.

À partir de ces données, répondez à la question suivante : Existe-t-il une différence de poids des poussins entre les différentes diètes au terme de l'expérience?

Si vous trouvez une différence significative, déterminez quelle(s) diète(s) diffère des autres.

16. L'analyse de variance à plusieurs facteurs

16.1. Introduction

Nous avons vu au Chapitre 15 une technique nommée ANOVA à un facteur. Cette dernière permettait de tester si une variable qualitative avait un effet significatif sur une variable quantitative.

Nous verrons au Chapitre 29 qu'il est possible de combiner des variables qualitatives et quantitatives dans un même modèle statistique. Cette façon de faire est clairement celle que je privilégie, i.e. d'utiliser le modèle linéaire dans la majorité des circonstances, puisqu'elle est beaucoup plus flexible.

Cependant, il existe aussi des extensions de l'ANOVA permettant d'intégrer plusieurs variables explicatives dans ce type de test. Bien que je ne sois pas un pratiquant de cette approche, je vous l'expose tout de même ici, car (1) elle est tout de même utilisée, entre autres en biologie médicale et (2) qu'elle permet une réflexion sur le design d'expérience qui pourra vous être bénéfique de toute façon.

Vous constaterez que la complexité des ANOVA à plusieurs facteurs tient beaucoup, comme dans plusieurs approches statistiques, à la tonne de nouveau vocabulaire à assimiler.

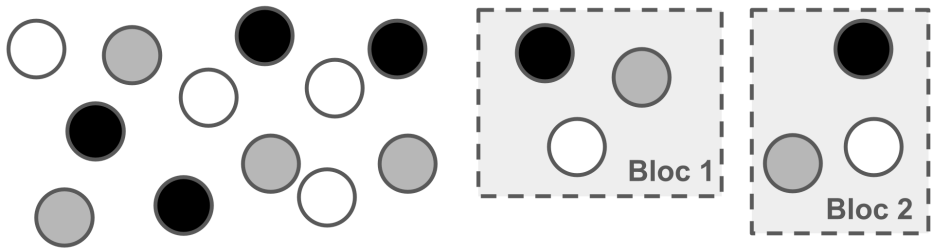
16.2. L'ANOVA en blocs aléatoires

Vous vous rappelez peut-être que dans l'ANOVA à un facteur, chaque observation était indépendante et effectuée sur un individu ou une parcelle différente. Cela faisait partie des assomptions de la technique.

Dans l'ANOVA en blocs aléatoires, différents traitements (ou conditions) sont regroupés physiquement ou spatialement dans un bloc. Chaque traitement est appliqué une seule fois par bloc.

On obtient une ANOVA en blocs aléatoires par exemple si on choisit 10 champs agricoles où effectuer notre expérience, et que dans chaque champ, on choisit un site contrôle, un site où on ajoute de l'engrais et un site où on arrose plus fréquemment.

Voici une comparaison schématique de l'ANOVA à un facteur et de l'ANOVA en blocs aléatoires :



(a) ANOVA à un facteur (a) ANOVA en blocs aléatoires

Ce type de design ajoute, outre l'effet du traitement et l'erreur intragroupe que nous avions dans l'ANOVA à un facteur, un troisième compartiment de variation, soit la variabilité entre les blocs :

Source	Degrés de liberté	Carrés moyens	Ratio de F
Traitements	a-1	$SS_{traitement}/(a-1)$	$MS_{traitement}/MS_{intra}$
Blocs	b-1	$SS_{bloc}/(b-1)$	MS_{bloc}/MS_{intra}
Intra-groupe (résidus)	**(a-1)(b-1)**	$SS_{intra}/(a-1)(b-1)$	

Où a est le nombre de traitements et b est le nombre de blocs.

On peut donc ici tester deux hypothèses avec cette technique :

- (1) Est-ce qu'il y a une différence entre les blocs (rarement une question d'intérêt biologique) et
- (2) est-ce qu'il y a une différence entre les traitements.

Remarquez que le calcul du ratio de F pour l'effet du traitement est le même que dans l'ANOVA à un facteur, sauf pour les degrés de liberté des résidus qui sont réduits, car ils sont "passés" dans l'estimation des effets de bloc.

Ce qui veut dire que si les différences entre les blocs sont grandes, le test sera plus puissant parce que l'on réduit considérablement le compartiment intra (les résidus). Par contre, si les différences entre les blocs sont petites, le test sera moins puissant car on gaspille des degrés de liberté pour estimer l'effet entre les blocs.

16.3. ANOVA pairée et mesures répétées

Vous verrez souvent dans la littérature les termes ANOVA pairée et ANOVA pour mesures répétées. Bien qu'ayant leurs propres noms, ces deux techniques sont des versions spécifiques de l'ANOVA en blocs aléatoires.

Dans le cas de l'analyse pairée, il s'agit d'un design en blocs aléatoires, mais où notre variable qualitative ne possède que deux niveaux. Les

traitements sont donc appliqués "par paire" sur chacun des individus/sites/parcelles, etc.

Dans le cas de l'analyse pour mesure répétées, chaque "bloc" de traitement est représenté par un individu. Les différents moments dans le temps correspondent aux différents niveaux de la variable qualitative. Conceptuellement la question est très différente de l'ANOVA en blocs aléatoires, mais les analyses statistiques sont exactement les mêmes.

16.4. Labo : L'ANOVA en blocs aléatoires

Pour essayer l'ANOVA en blocs aléatoires, nous allons avoir besoin de données appropriées, structurées en blocs. Je vous ai préparé un fichier Excel, contenant trois onglets, avec des données déjà prêtes à être traitées[1]. Remarquez que ces données sont déjà organisées correctement au format long, mais que dans la vraie vie, il est possible que vous ayez à réorganiser vos données avant de procéder.

Nous étudierons donc dans ce chapitre la réponse chimique des plantes à divers types d'attaques[2]. Notre variable expliquée sera dans tous les cas, la concentration en composés phénoliques (mg par g de biomasse sèche). Pour l'ANOVA en blocs aléatoires nous allons regarder l'effet d'une simulation d'herbivorie. Notre variable qualitative aura 3 niveaux, soit Contrôle, Défoliation (enlever toutes les feuilles) et Taille (couper entièrement la tige). Une série de parcelles ont été choisies pour l'expérience, et dans chaque parcelle, nous avons choisi un individu différent pour chacun des traitements.

Commençons par activer les librairies et charger les données :

[1] https://drive.google.com/file/d/1Te6Iofba8CRanItvj7KcizaLoEI3Axq0/view?usp=sharing

[2] Les données sont inventées, mais fortement inspirées de https://www-jstor-org.biblioproxy.uqtr.ca/stable/3545829

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(readxl)
randomized_block <-
 ↪  read_excel("donnees/PourANOVA.xlsx", sheet = 1)
```

Après cette étape, il faudrait normalement inspecter les données, regarder les distributions des variables, etc., mais pour accélérer ce chapitre, assumez que j'ai déjà tout fait ça pour vous.

Pour ajuster une ANOVA en blocs aléatoires, on utilise la fonction **aov** (comme pour l'ANOVA à un facteur), à laquelle on passe une formule, dans laquelle on mentionne notre variable de traitement (Herbivorie) et notre variable de bloc (Parcelle), comme ceci :

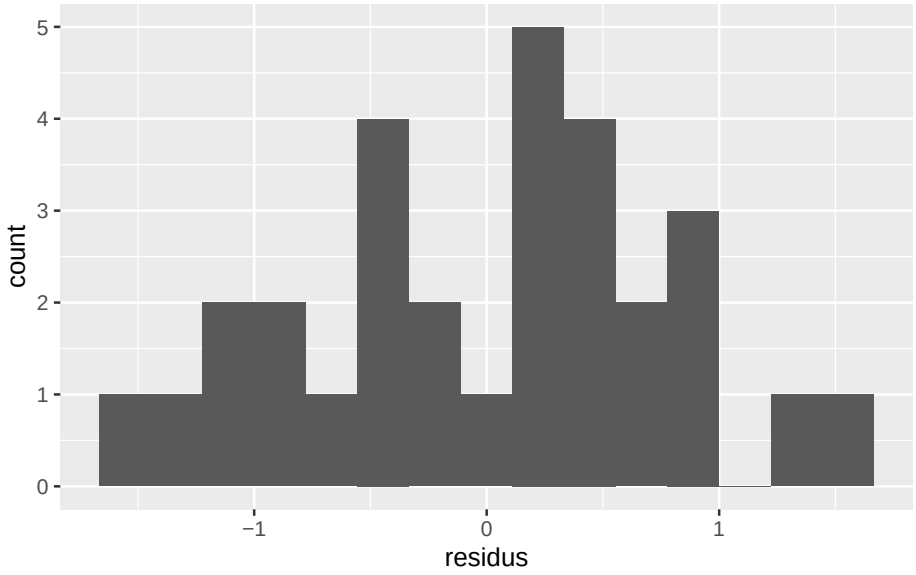
```
modele <- aov(Concentration_Phenols ~ Herbivorie +
 ↪  Parcelle , data = randomized_block)
```

Attrapons maintenant les résidus de ce modèle pour pouvoir les valider :

```
randomized_block <-
  randomized_block |>
  mutate(residus = resid(modele))
```

La normalité des résidus s'évalue de la même façon que d'habitude :

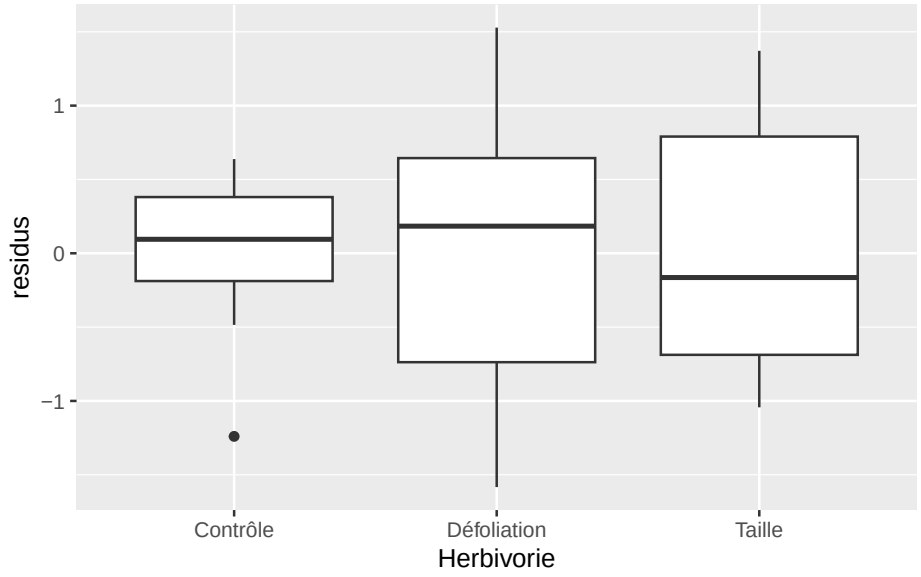
```
ggplot(randomized_block,aes(residus)) +
  geom_histogram(bins = 15)
```



La distribution n'est pas trop mal. Rien d'inquiétant.

Pour l'homogénéité des résidus, on doit faire comme pour l'ANOVA à un facteur, c'est-à-dire procéder à l'aide d'un diagramme à moustache et s'assurer que les boîtes sont relativement semblables entre les différents niveaux de notre variable qualitative :

```
ggplot(randomized_block, aes(Herbivorie, residus)) +
  geom_boxplot()
```



Une fois tout cela vérifié, on peut maintenant regarder les résultats de notre modèle :

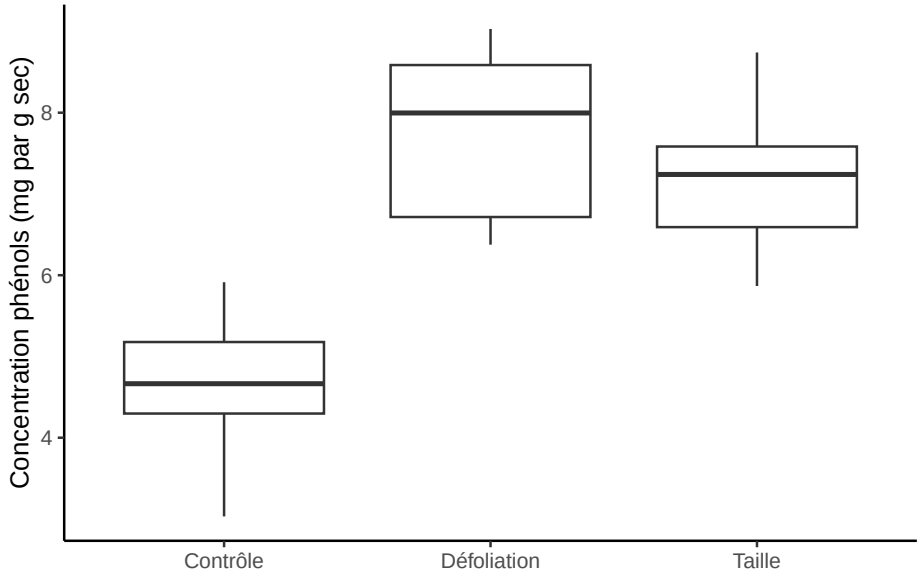
```
summary(modele)
```

```
            Df Sum Sq Mean Sq F value   Pr(>F)
Herbivorie   2  51.68  25.841   25.52 5.57e-06 ***
Parcelle     9   6.29   0.699    0.69    0.709
Residuals   18  18.23   1.013
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Comme discuté ci-haut, l'ANOVA en blocs aléatoires effectue en fait deux tests. Un premier sur notre variable d'intérêt (ici l'herbivorie) et un deuxième sur notre variable de bloc (ici la parcelle). Nos résultats indiquent que la concentration en phénols varie en fonction de notre traitement ($p<0,05$), mais qu'il n'y a pas de différence significative entre les parcelles comme tel ($p>0,05$).

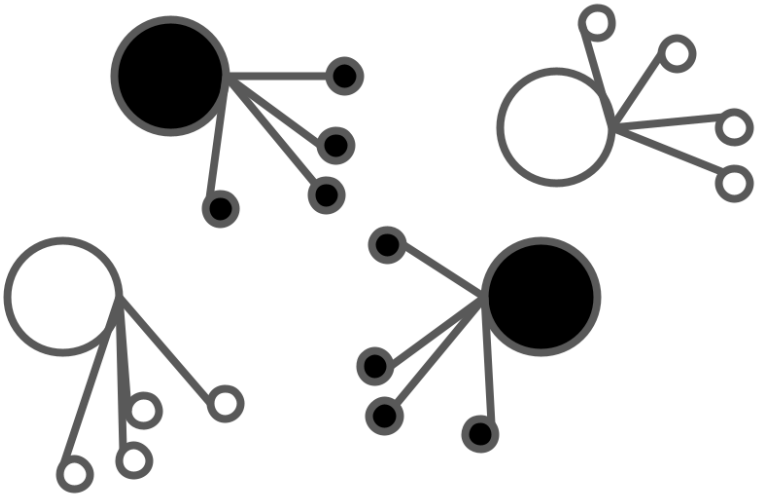
Si on avait à produire un graphique de ces résultats, la meilleure façon serait probablement à l'aide d'un diagramme à moustaches, un peu comme ceci :

```
ggplot(randomized_block,
 ↪  aes(Herbivorie,Concentration_Phenols)) +
  geom_boxplot() +
  theme_classic() +
  labs(x = NULL, y = "Concentration phénols (mg par g
   ↪  sec)")
```



16.5. L'ANOVA imbriquée

Dans un design imbriqué, chaque échantillon (site/parcelle/individu) ne subit qu'un seul des traitements. Par contre, cet échantillon est sous-échantillonné (mesuré) plusieurs fois. Conceptuellement, on pourrait représenter l'ANOVA imbriquée comme ceci :



Remarquez que dans cette façon de faire, le nombre d'échantillons indépendants n'augmente pas, uniquement le nombre d'observations.

Particularités statistiques

Comme pour l'ANOVA en blocs aléatoires, les carrés moyens sont partitionnés en 3 compartiments. Les calculs sont par contre modifiés, avec d'importantes conséquences statistiques. La plus importante étant que le ratio de F pour les traitements n'est plus divisé par les résidus intra mais par les carrés moyens des échantillons.

Source	Degrés de liberté	Carrés moyens	Ratio de F
Traitements	a-1	$SS_{traitement}/(a-1)$	$\mathbf{MS_{traitement}/MS_{échantillons}}$
Échantillons d'un traitement	**a (b-1)**	$SS_{échantillon}/a\ (b-1)$	
Intra-échantillon (résidus)	**a b (n-1)**	SS_{intra} / a b (n-1)	

Où a est le nombre de traitements, b est le nombre d'échantillons indépendants pour chaque niveau de traitement et n est le nombre de sous-échantillons pour chaque échantillon.

Le calcul de l'ANOVA imbriquée est mathématiquement équivalent à calculer la moyenne des sous-échantillons dans chaque échantillon et d'utiliser cette information dans une ANOVA à un facteur classique.

> ⚠ Avertissement
>
> Ce qu'il est important de retenir à propos des ANOVA imbriquées est que les logiciels de statistiques ne peuvent PAS deviner que votre ANOVA est imbriquée plutôt qu'en blocs aléatoires. C'est à vous de le mentionner explicitement dans la formulation de votre analyse.

Oublier de le faire augmente de façon importante la possibilité d'erreur de type I de votre test, puisque vous considererez alors comme indépendants des échantillons qui ne le sont pas. Il s'agirait d'une erreur grave pouvant remettre en cause tous vos résultats.

16.6. Labo : L'ANOVA imbriquée

Pour illustrer l'ANOVA imbriquée, nous allons reprendre l'expérience précédente, mais cette fois-ci, les concentrations de phénols ont été mesurées 5 fois sur chacun des individus.

Charger les données :

```
nested <- read_excel("Donnees/PourANOVA.xlsx", sheet =
↪  2)
```

Préparer un modèle d'ANOVA imbriquée :

```
modele <- aov(Concentration_Phenols ~ Herbivorie +
↪  Error(Individu), data = nested)
```

Remarquez bien l'ajout, où notre variable qui désigne l'identité des individus a été "emballée" de la fonction `Error`. C'est la façon d'expliquer à R que les individus ont été sous-échantillons plusieurs fois, et donc que nos lignes dans notre tableau de données ne sont pas entièrement indépendantes les unes des autres. Il s'agit de la clé pour bien faire les ANOVA imbriquées dans R.

Si on observe les résultats de ce modèle, on remarque que les résultats sont maintenant séparés en deux parties :

```
summary(modele)
```

```
Error: Individu
           Df Sum Sq Mean Sq F value  Pr(>F)
Herbivorie  2  87.14   43.57   35.62 5.3e-05 ***
Residuals   9  11.01    1.22
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Error: Within
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals 48  43.06  0.8972
```

La partie qui nous intéresse de prime abord est la première, où l'on peut constater que la différence entre nos traitements d'herbivorie est significative. La somme des carrés des erreurs utilisées pour ce calcul provient de la différence entre les individus.

L'erreur provenant de la variabilité entre les échantillons provenant d'un même individu a été séparée dans l'autre partie (Error Within) et n'est pas utilisée dans le calcul de la valeur de F pour notre traitement d'herbivorie.

Rappelez-vous que le calcul de l'ANOVA imbriquée fonctionne exactement comme si, pour faire le calcul, nous avions utilisé la moyenne des sous-échantillons plutôt que leurs valeurs individuelles. Pour cette raison, la validation des résidus est un peu plus complexe que dans les autres ANOVA, et nous amènera à faire le calcul des résidus manuellement :

```
pour_validation <- nested |>
  group_by(Individu,Herbivorie) |>
  summarize(Concentration_Phenols =
   ↪  mean(Concentration_Phenols)) |>
  ungroup() |>
  group_by(Herbivorie) |>
  mutate(Prediction = mean(Concentration_Phenols)) |>
  ungroup() |>
  mutate(Residu = Concentration_Phenols - Prediction)
```

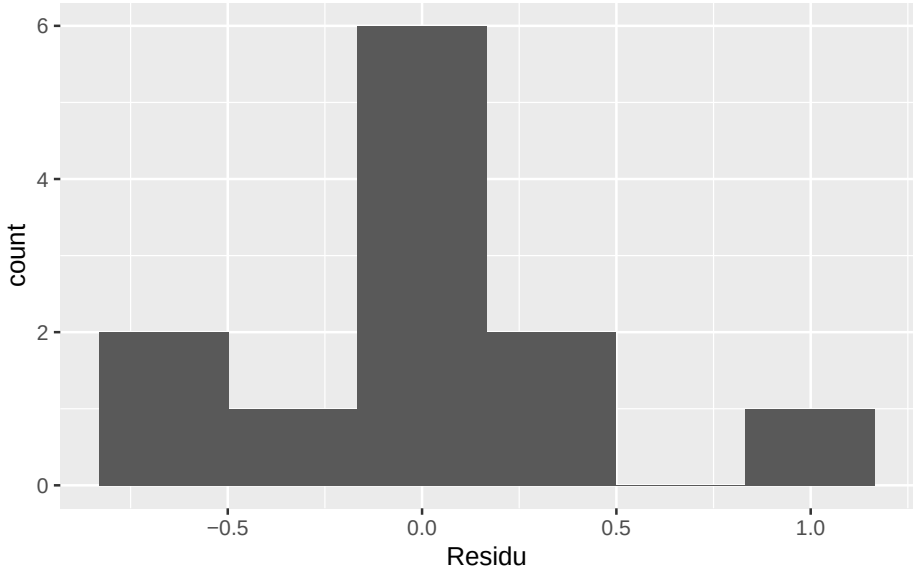
```
`summarise()` has grouped output by 'Individu'. You
can override using the `.groups` argument.
```

Ce n'est pas si important que vous compreniez tous les détails du calcul ci-haut. Il faut seulement être capable de remplacer les noms de variables pour les adapter à vos analyses le moment venu. En gros, la chaîne commence par calculer une moyenne de phénols pour chacun des individus, ensuite, elle ajoute une colonne de moyenne de phénols par niveau d'herbivorie (la prédiction), puis soustrait ces deux valeurs pour obtenir un résidu.

Une fois ce tableau obtenu, on fait les deux graphiques habituels, soit la normalité des résidus, et l'homogénéité entre les groupes. Remarquez

que, bien que nous avions recueilli 60 échantillons dans notre expérience, nous n'avons dans les faits que 12 individus indépendants :

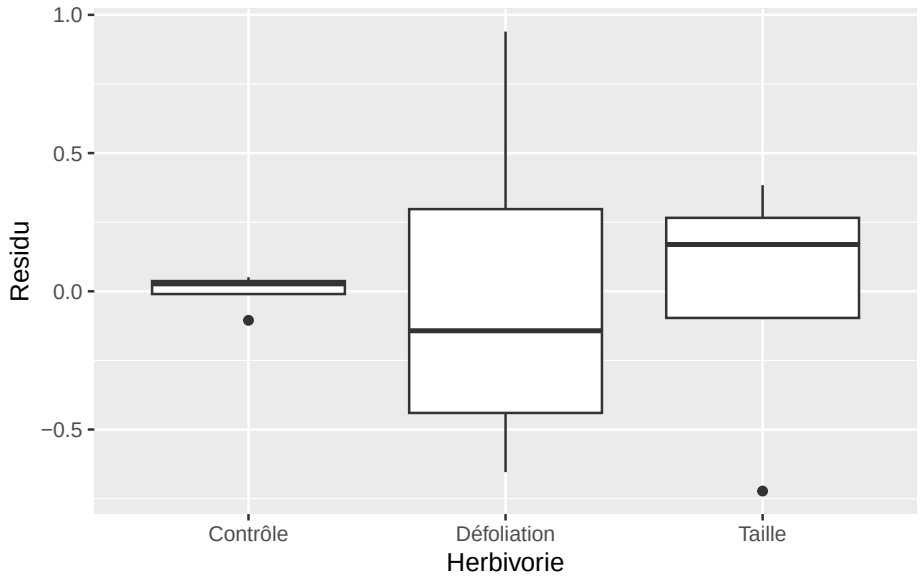
```
ggplot(pour_validation, aes(x = Residu)) +
  geom_histogram(bins = 6)
```



C'est aussi normal qu'on peut l'espérer avec seulement 12 observations.

Ensuite, il faut aussi regarder l'homogénéité de la variance entre les groupes :

```
ggplot(pour_validation, aes(x = Herbivorie, y =
↪  Residu)) +
  geom_boxplot()
```



Ici, c'est loin d'être parfait visuellement. Mais gardez en tête que chaque boîte ne représente en fait que 4 points. Si on observait le même genre de résultats avec 15-20 points par boîte, il y aurait lieu de s'inquiéter.

Dans ce genre de situation, il peut être légitime d'utiliser le test de Levene pour se rassurer :

```
library(car)
```

```
Loading required package: carData
```

```
Attaching package: 'car'
```

```
The following object is masked from 'package:dplyr':

    recode
```

```
The following object is masked from 'package:purrr':

    some
```

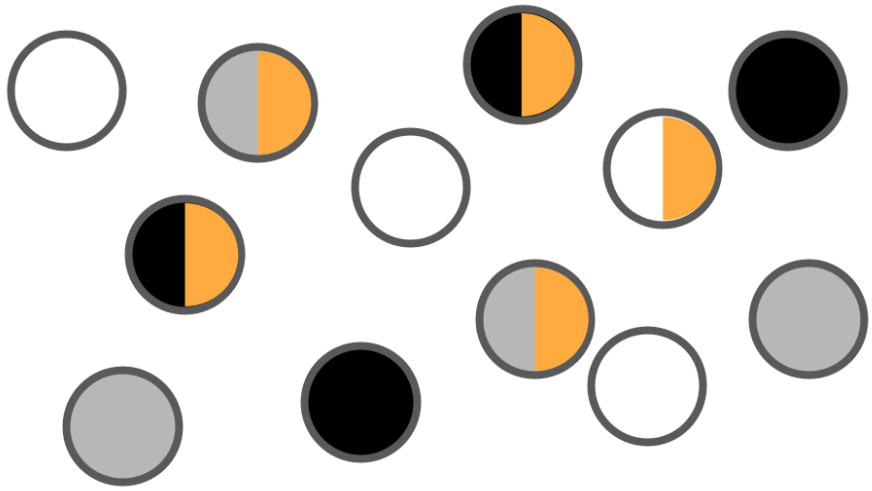
```
leveneTest(Residu~Herbivorie, data = pour_validation)
```

```
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  2  2.0451 0.1853
       9
```

Comme l'hypothèse nulle de ce test est que la variance est homogène entre les groupes et qu'ici on ne peut pas rejeter l'hypothèse nulle (p>0,05), on est donc rassurés, la variance est homogène entre nos groupes.

16.7. L'ANOVA à deux facteurs croisés

L'ANOVA à deux facteurs croisés s'utilise lorsque l'on veut tester simultanément l'effet de deux variables qualitatives sur notre variable expliquée. Chacune des variables peut avoir plusieurs niveaux. Dans la visualisation ci-dessous, on teste une première variable à trois niveaux (noir, gris ou blanc) et une deuxième variable à un niveau (orange ou non) :



Pour que l'analyse à deux facteurs croisés fonctionne correctement, il est important que chaque combinaison de traitements soit présente, et testée le même nombre de fois. Par exemple, pour tester à la fois l'effet d'augmenter ou diminuer le pH dans une culture et l'effet d'ajouter ou non du NaCl, nous aurions pu créer l'expérience suivante :

	Contrôle	NaCl
Contrôle	3 plantes	3 plantes
pH+	3 plantes	3 plantes
pH-	3 plantes	3 plantes

Si on calcule un minimum comme ici de 3 observations par combinaison de traitement, on constate que le nombre d'individus nécessaires devient rapidement élevé. Par exemple, ici, nous avons besoin de 18 individus pour tester nos deux variables.

Le grand avantage de ce genre de design est qu'il permet aussi de tester la présence d'interaction entre nos traitements.

Traitement statistique

L'analyse de la variance de ce modèle à deux facteurs croisés nécessite maintenant la description de 4 compartiments de variance, soit :

Source	Degrés de liberté	Carrés moyens	Ratio de F
Variable A	a-1	$SS_A/(a\text{-}1)$	MS_A/MS_{intra}
Variable B	b-1	$SS_B/(b\text{-}1)$	MS_B/MS_{intra}
Interaction	(a-1)(b-1)	$SS_{AB}/(a\text{-}1)(b\text{-}1)$	MS_{AB}/MS_{intra}
Intra-groupe (résidus)	a b (n-1)	SS_{intra} / a b (n-1)	

Où a est le nombre de niveaux de la variable A, b est le nombre de niveaux de la variable B et n est le nombre de réplicats pour chaque combinaison.

Il est possible de "récupérer" des expériences où le n n'était pas égal dans chacune des cellules, en passant par une approche de modèle linéaire mixte (Chapitre 30) plutôt que d'ANOVA.

16.8. Labo : L'ANOVA à deux facteurs croisés

Pour essayer l'ANOVA à plusieurs facteurs, nous allons reprendre l'expérience précédente où on simulait l'herbivorie sur une série de plantes, mais ici, on ajoutera en plus une deuxième variable dans notre expérience, où la moitié des plantes seront aussi soumises à une expérience où on augmente artificiellement la compétition.

Comme expliqué dans la théorie, il faut penser à l'ANOVA à deux facteurs croisés comme si on roulait deux expériences simultanément. Ici, on en fait une sur la compétition et une sur l'herbivorie.

Donc, on peut charger nos données, qui sont dans la 3e feuille du fichier Excel, puis on ajuste le modèle à l'aide de la fonction `aov`. Comme on

veut savoir si l'effet de l'herbivorie est le même sous forte compétition ou non, on ajoutera aussi un terme d'interaction au modèle, comme ceci :

```
two_way <- read_excel("donnees/PourANOVA.xlsx", sheet =
↪  3)
modele <- aov(Concentration_Phenols ~
↪  Herbivorie+Competition+Herbivorie:Competition, data
↪  = two_way)
```

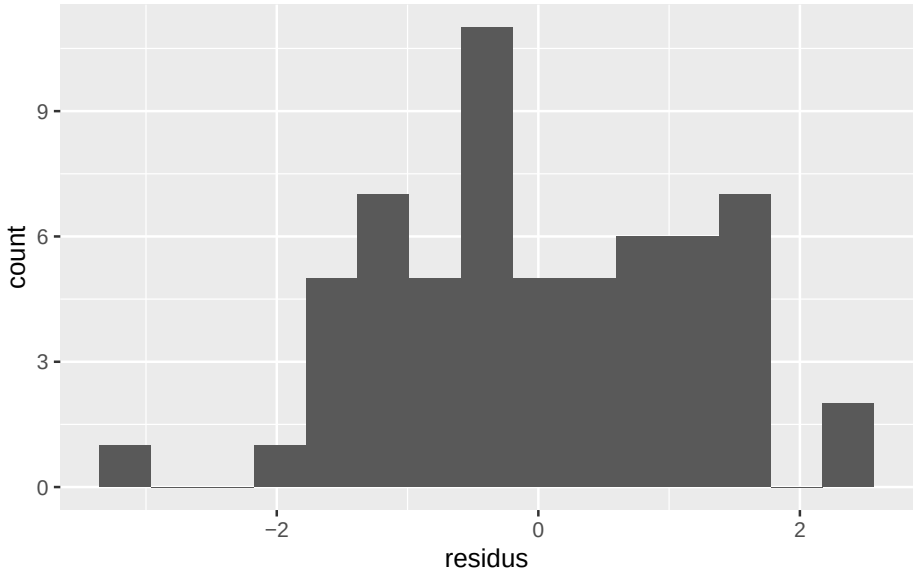
Remarquez que l'on devrait ici aussi inspecter correctement nos données avant de commencer à travailler, mais nous ne le ferons pas ici pour ne pas étirer ce chapitre.

Une fois le modèle ajusté, il faut, comme toujours, l'inspecter avant de procéder à son interprétation. Nous allons conserver, comme pour la régression, les prédictions et les résidus du modèle dans notre tableau de données pour faciliter notre travail :

```
two_way <-
  two_way |>
  mutate(
    residus = resid(modele),
    predictions = predict(modele)
  )
```

Observons tout d'abord la distribution des résidus, qui devrait donner une courbe normale :

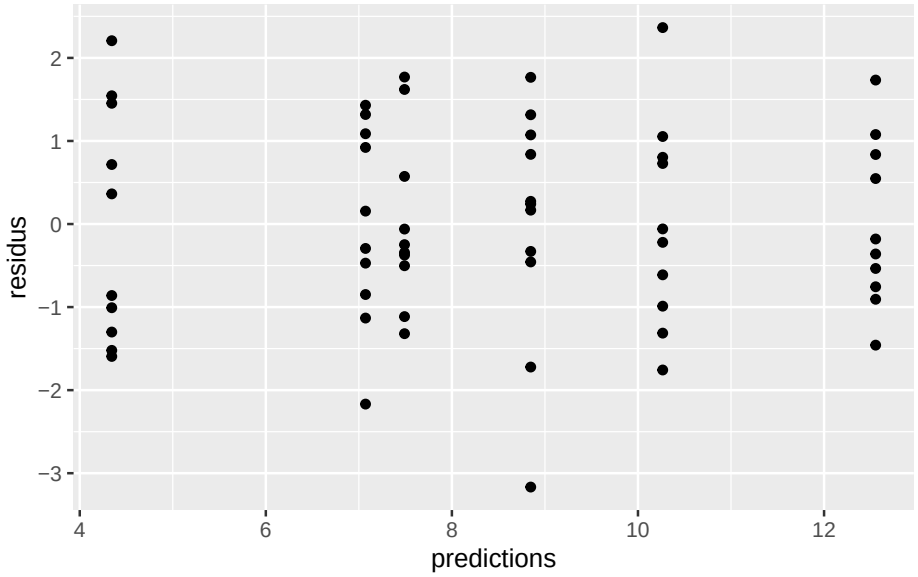
```
ggplot(two_way, aes(residus)) +
  geom_histogram(bins = 15)
```



Pas trop mal.

Maintenant, assurons-nous que les résidus sont homogènes à travers le gradient de prédictions. Autrement dit, vérifions que notre modèle se trompe à peu près de la même façon avec les petites valeurs et les grandes valeurs :

```
ggplot(two_way, aes(predictions, residus)) +
  geom_point()
```



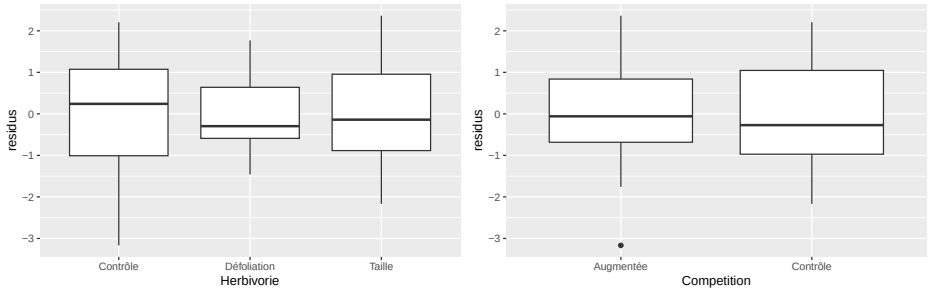
Difficile de faire mieux.

Enfin, il faut aussi vérifier que les erreurs sont équivalentes entre chacun des niveaux de nos variables qualitatives. On peut entrevoir cette information dans le graphique précédent, mais avec une boîte de diagramme à moustache par catégorie, c'est souvent plus facile :

```
ggplot(two_way, aes(Herbivorie, residus)) +
  geom_boxplot()
ggplot(two_way, aes(Competition, residus)) +
  geom_boxplot()
```

Rien d'extrême ici non plus. On peut donc interpréter notre modèle.

```
summary(modele)
```

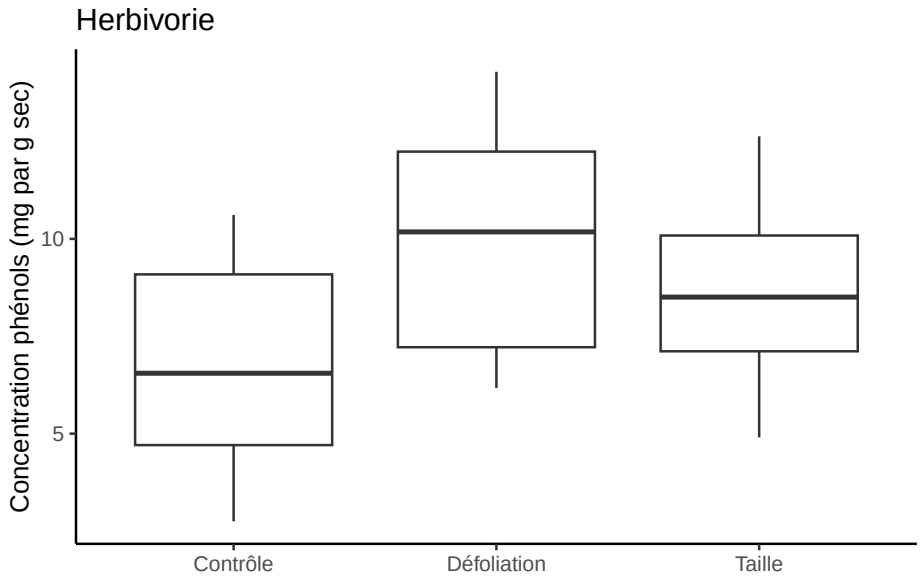



```
                       Df Sum Sq Mean Sq F value
Herbivorie              2 114.51   57.25   37.39
Competition             1 276.21  276.21  180.38
Herbivorie:Competition  2   9.22    4.61    3.01
Residuals              55  84.22    1.53
                         Pr(>F)
Herbivorie             5.58e-11 ***
Competition             < 2e-16 ***
Herbivorie:Competition   0.0575 .
Residuals
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

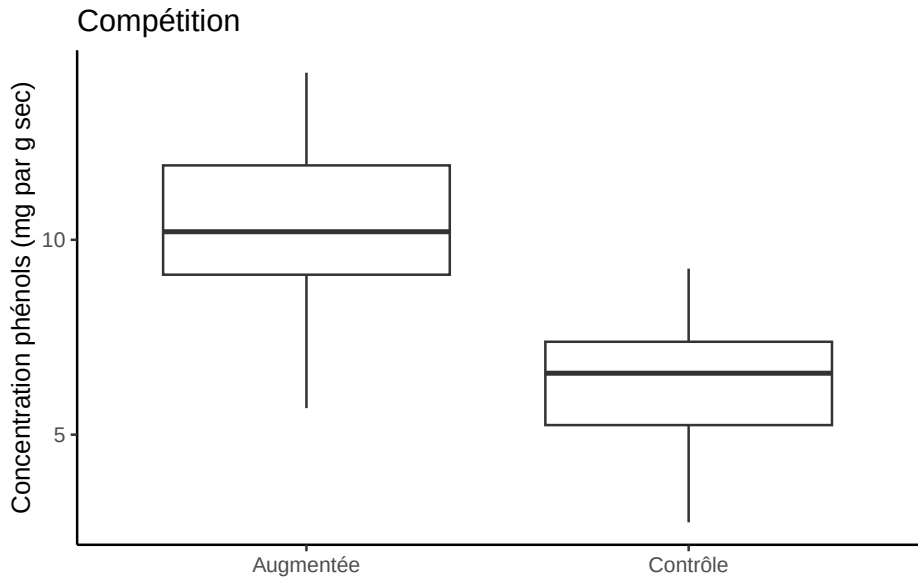
On voit dans ces sorties que notre variable d'herbivorie a un effet significatif (p<0,05). Notre variable de compétition a aussi un effet significatif (p<0,05). Mais que l'interaction entre les deux variables, elle, n'est pas significative. Autrement dit, l'effet de la compétition est le même, peu importe le niveau d'herbivorie et vice-versa, l'effet de l'herbivorie est le même, peu importe la quantité de compétition.

Si on voulait présenter ces résultats dans un rapport, on pourrait probablement le faire à l'aide de deux diagrammes à moustache, comme ceci :

```
ggplot(two_way, aes(Herbivorie, Concentration_Phenols))
↪  +
  geom_boxplot() +
  theme_classic() +
  labs(x = NULL, y = "Concentration phénols (mg par g
   ↪  sec)", title = "Herbivorie")
```




```
ggplot(two_way, aes(Competition,
↪  Concentration_Phenols)) +
  geom_boxplot() +
  theme_classic() +
  labs(x = NULL, y = "Concentration phénols (mg par g
   ↪  sec)", title = "Compétition")
```



16.9. Comparaison des différentes ANOVA

Voici maintenant un petit résumé des avantages et inconvénients des différents types d'ANOVA. Pensez à relire ce résumé avant de démarrer une expérience, afin de choisir la bonne structure, qui maximisera vos chances de détecter l'effet désiré.

ANOVA à un facteur

- Design le plus simple, mais aussi le plus puissant, car il n'y a pas de gaspillage en pseudo-réplication
- Peut s'accommoder de tailles d'échantillons inégales entre les groupes
- Ne tient pas compte de l'hétérogénéité (de l'environnement ou des individus)

- Donc si on a beaucoup de bruit ou de facteurs confondants, on aura moins de puissance statistique

ANOVA en blocs aléatoires

- Façon efficace de tenir compte de l'hétérogénéité
- Utile lorsqu'on est contraint par l'espace ou le temps, p. ex. en considérant chaque site comme un bloc.
- Il y a un coût statistique inutile si la variabilité inter-bloc est faible
- Si on perd un des échantillon dans le bloc, le bloc entier doit être éliminé de l'analyse
- Assume qu'il n'y a pas d'interaction entre les traitements et les blocs (i.e. que l'effet du traitement et le même dans tous les blocs).

ANOVA imbriquée

- Avantage principal : augmente la précision de la réponse de chacun des échantillons; donc on devrait en principe augmenter la puissance du test
- Permet de tester deux hypothèses : la variation entre les traitements et la variation entre les échantillons dans un traitement
- Potentiellement dangereux car souvent analysée de façon incorrecte
- Souvent, l'effort de sous-échantillonner est gaspillé par rapport au gain qui aurait été fait en ajoutant des échantillons indépendants.

ANOVA à deux facteurs croisés

- Comme combiner deux expériences dans une seule
- Utilisation plus efficace du temps et de l'espace
- Toutes les combinaisons de traitements doivent apparaître dans le design de l'expérience
- Permet de séparer les effets simples des interactions
- Désavantage principal : le nombre de combinaisons peut augmenter rapidement, affectant le nombre d'échantillons qu'il est nécessaire de traiter.

- Ce design ne tient pas compte non plus de l'hétérogénéité

16.10. Exercice : L'analyse de variance à plusieurs facteurs

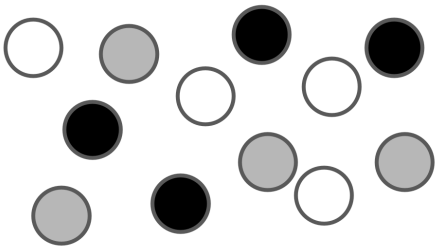
Pour cet exercice, téléchargez d'abord le tableau de données Grillons.csv[3]. Ce tableau contient le résultat d'une expérience visant à déterminer si la réponse immunitaire des mâles grillons diffère de celle des femelles. La réponse immunitaire est mesurée par un score d'encapsulation. Plus ce score est élevé, plus la réponse immunitaire est forte. L'expérience a été effectuée sur 10 grillons mâles et 10 grillons femelles. Pour chaque individu, la réponse immunitaire a été mesurée à trois reprises.

Votre travail consiste donc à :

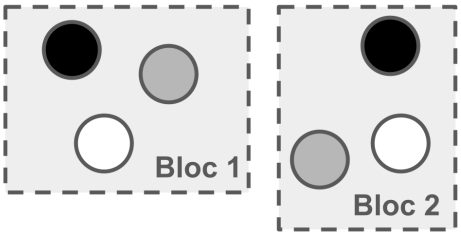
a) Charger, inspecter et préparer les données
b) Déterminer quelle sera l'ANOVA appropriée pour analyser ce jeu de données
c) Appliquer le test d'ANOVA choisi
d) Valider le modèle
e) Interpréter les sorties
f) Préparer un graphique propre des résultats

16.11. Aide mémoire visuel

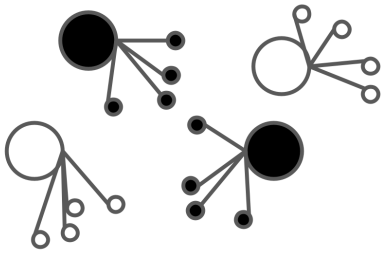
[3] https://drive.google.com/file/d/10h0onFXwaTdUqk3Pw1h_xDRtTCf8pspp/view?usp=sharing



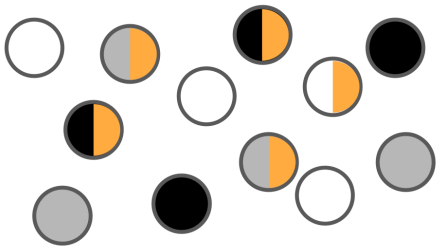
(a) ANOVA à un facteur (*one-way*)



(a) ANOVA en blocs aléatoires (*randomized block*)



(a) ANOVA imbriquée (*nested*)

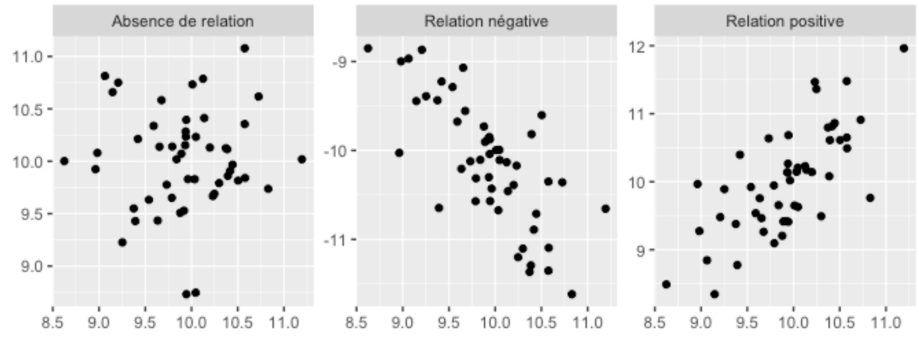


(a) ANOVA à deux facteurs croisés (*two-way, crossed*)

17. La corrélation

Dans ce chapitre, nous verrons une technique pour tester l'association entre deux variables quantitatives. Nous voudrons déterminer s'il existe une relation entre nos deux variables. On entend par **relation** que, si une des variable change, l'autre changera systématiquement dans un sens donné.

Évidemment il y aura aussi du bruit autour de la relation, elle ne sera pas parfaite. On parlera de **relation positive** lorsque les deux variables augmentent simultanément, et de **relation négative** lorsque l'augmentation d'une variable entraîne la diminution d'une autre.



Notez que l'on ne parle pas de changement dans le temps, mais plutôt du portrait de la situation lorsqu'on regarde une série d'observations où on a mesuré les deux variables, un **gradient**. L'analyse des séries temporelles comme tel est un sujet avancé, que nous ne verrons pas dans ce manuel.

17.1. Corrélation de Pearson

Sans doute une des statistiques les plus connues et les plus utilisées, la corrélation de Pearson, permet d'évaluer la force de l'association entre deux variables et d'informer sur le sens (positif ou négatif) de celle-ci. Bien qu'il existe d'autres corrélations (p. ex. celle de Spearman), si quelqu'un vous parle de corrélation sans donner d'autre détail, il vous parle assurément de la corrélation de Pearson.

Avant de vous lancer dans une analyse de corrélation, vous devez vous assurer que vos données correspondent à ce qu'attend la procédure. Outre le fait que vos observations doivent être indépendantes, la corrélation de Pearson assume aussi que vos données présentent une relation linéaire et qu'elles sont distribuées normalement. Si l'une ou l'autre de ces conditions n'est pas remplie, vous pouvez appliquer une méthode alternative, soit la corrélation de Spearman. Nous reviendrons sur cette méthode dans le Chapitre 21.

Le calcul de la corrélation est basé sur une statistique nommée la **covariance**. Je vous épargnerai ici son calcul, mais ce qu'il est important de comprendre est que la covariance mesure combien deux variables varient une avec l'autre. Si les deux variables augmentent ensemble, ce chiffres sera grand et positif. Si lorsqu'une augmente l'autre diminue, ce chiffre sera très négatif, et si les variables ne sont pas reliées, ce chiffre sera près de zéro.

Le problème majeur de la covariance est que sa valeur dépend des unités dans lesquelles ont été mesurées nos données. Si par exemple on change des m pour des km, notre covariance pourrait devenir beaucoup plus petite. Ce genre de problème rend extrêmement complexe les comparaisons entre les jeux de données.

C'est pourquoi le calcul de la corrélation inclut un facteur de correction, qui remet toutes les données à la même échelle. Conceptuellement, on peut donc décrire la corrélation comme ceci :

$$r_{X,Y} = \frac{cov(X,Y)}{\sigma_x \sigma_y}$$

Soit que la corrélation (r) entre X et Y se mesure comme la covariance entre les deux variables, divisée par le produit de leur écart-types.

Ce facteur de correction fait en sorte que la valeur de la corrélation se retrouvera toujours entre -1 et 1. Une valeur de -1 présentant une relation négative parfaite (i.e. sans bruit), 0 une absence de relation et 1 une relation positive parfaite.

Voici, pour vous donner une idée, quelques exemples de relations et les valeurs de corrélation qui leur sont associées :

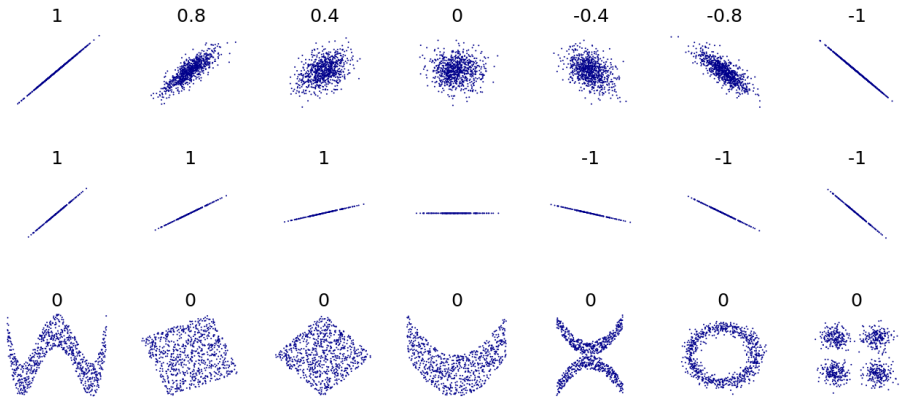


Figure 17.1.: DenisBoigelot, original uploader was Imagecreator, CC0, via Wikimedia Commons

Remarquez particulièrement avec la dernière ligne de cette figure que la corrélation ne mesure que des relations linéaires dans les données. Elle n'est pas appropriée pour mesurer d'autres formes.

Si vous voulez vous entraîner à bien évaluer la force d'une corrélation, sachez qu'il existe des jeux en ligne créées exactement à cette fin[1]. Je vous conseille d'en essayer quelques parties pour vous faire l'œil avant de continuer.

Enfin, il est important de comprendre que trouver une corrélation entre deux variables ne veut PAS dire que ces deux variables sont effectivement reliées par un mécanisme quelconque. Il existe d'ailleurs un adage anglais qui illustre bien ce propos : correlation is not causation. Pour vous donner une idée, allez visiter le site *Spurious Correlations*[2]. Ce site a recensé les corrélations les plus loufoques que l'on pouvait trouver, et en ont même fait un livre. Ils montrent, entre autres, qu'il existe une corrélation de 0,66 entre le nombre de films de Nicolas Cage et le nombre de décès par noyade...

Tout ça pour dire que, soyez prudentes, réfléchissez bien à votre question écologique avant de vous lancer dans toutes sortes de statistiques!

17.2. Tester une corrélation de Pearson

La majorité du temps, la corrélation de Pearson sera utilisée comme tel : on calcule la valeur de r, on l'interprète, et c'est tout. Il pourra cependant arriver que l'on vous demande de savoir si cette valeur de r est significative (i.e. significativement différente de zéro) ou non, particulièrement si votre échantillon est petit ou votre valeur de r plutôt faible. Voici donc la procédure pour tester cette corrélation.

Nous travaillerons la corrélation de Pearson avec un petit exemple où vous avez mesuré la transparence de l'eau (m) dans 24 lacs à l'aide d'un

[1] http://guessthecorrelation.com/
[2] https://www.tylervigen.com/spurious-correlations

disque de Secchi[3] et la densité de poissons par filet installé. Vous voulez savoir si ces deux mesures sont associées ou non.

Étape 1 : Définir les hypothèses

L'hypothèse nulle de ce test est que la corrélation entre les deux variables est de zéro, et l'hypothèse alternative est qu'elle est différente de zéro.

$$H_0 : \rho = 0$$
$$H_1 : \rho \neq 0$$

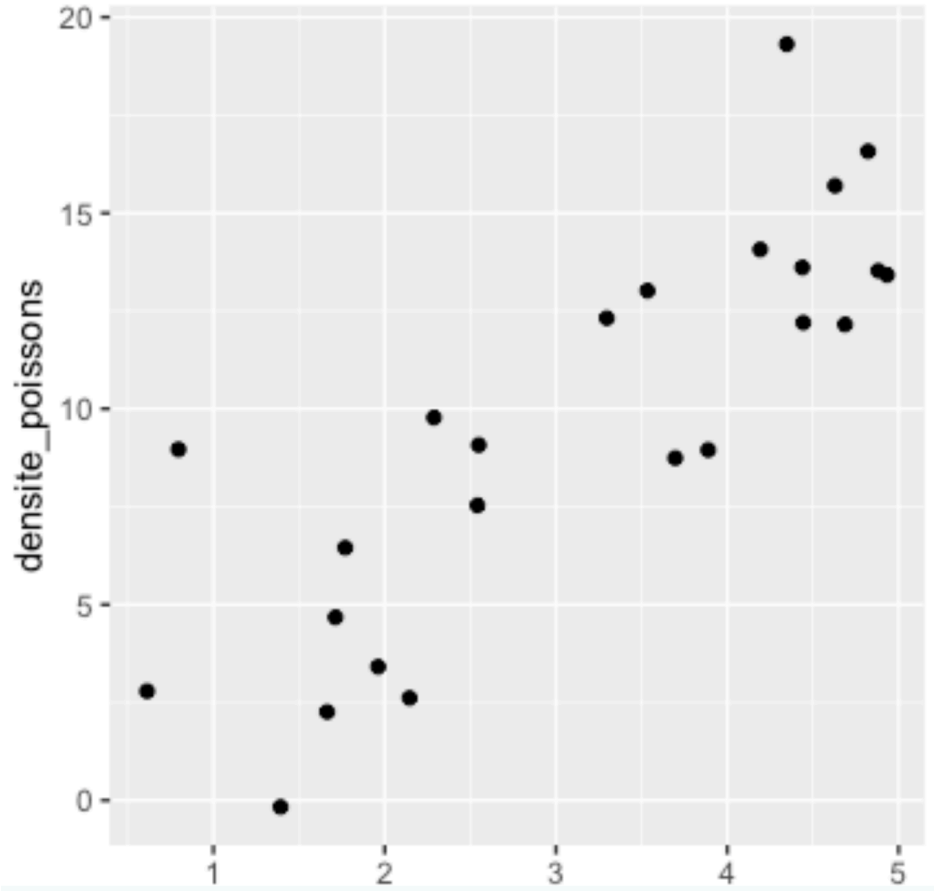
Notez que le symbole de la corrélation, lorsque l'on discute de la valeur de la population est le symbole grec rhô (ρ). Le r étant en général réservé aux échantillons.

Étape 2 : Explorer visuellement les données

Comme nous avons discuté précédemment, il y a deux assomptions que l'on peut vérifier à ce point avec nos données, soit la linéarité de la relation et la normalité des distributions.

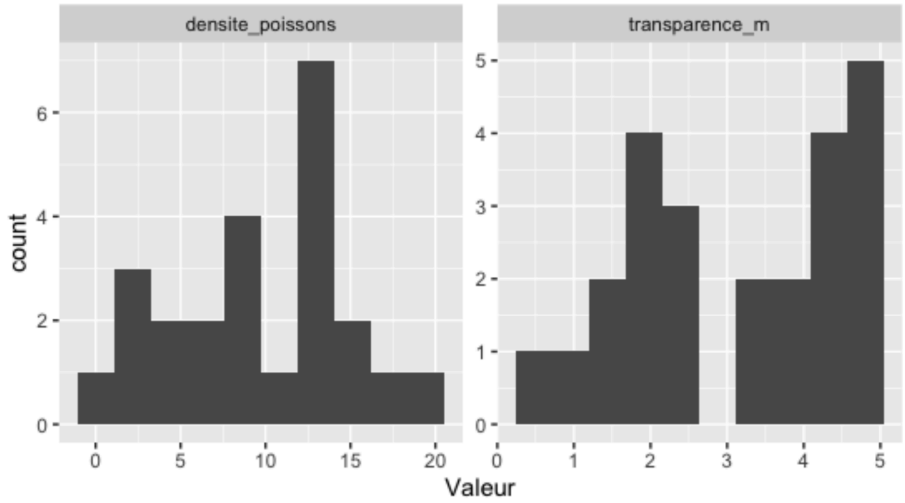
La linéarité de la relation peut être établie à l'aide d'un nuage de points montrant les deux variables d'intérêt :

[3] https://fr.wikipedia.org/wiki/Disque_Secchi



En plus de permettre d'évaluer la linéarité de la relation, ce graphique nous permet aussi de se faire une idée sur la valeur de corrélation que l'on devrait trouver. Ici, "au pif" je dirais que l'on devrait s'attendre à quelque chose qui tourne autour de 0,7.

La normalité des deux variables s'évalue facilement, quant à elle, à l'aide d'histogrammes :



Nos histogrammes ne sont clairement pas parfaits, mais étant donné notre faible taille d'échantillon, on ne peut pas dire qu'ils suggèrent une absence de normalité.

Étape 3 : Calculer la statistique de test

La première étape du test est bien sûr de calculer notre valeur de r. Dans notre exemple, la valeur calculée est de 0,833. Ensuite, le test de cette corrélation s'effectue en calculant une valeur de t, à l'aide de la formule suivante :

$$t = \frac{r\sqrt{n-2}}{\sqrt{(1-r^2)}}$$

Où n est la taille de notre échantillon et r est la valeur de la corrélation. Pour notre exemple, cette valeur de t sera donc de 6,98.

Étape 4 : Obtenir la valeur de p

Comme tous les autres tests jusqu'à présent, le test de corrélation est basé sur la probabilité de trouver une corrélation aussi grande si il n'y avait aucune association entre les populations. Rappelez vous qu'en général, deux populations non reliées ne présenteront pas de corrélation, mais qu'une fois de temps en temps, ça peut arriver, comme pour le nombre de films de Nicolas Cage et le nombre de décès par noyade!

On va donc trouver dans la distribution de t la probabilité associée à la valeur de t trouvée, pour nos n-2 = 22 degrés de liberté. Pour notre exemple, on trouve une valeur de p de 0,000000044.

Étape 5 : Rejeter ou non l'hypothèse nulle

Comme notre valeur de p est plus petite que la valeur seuil de 0,05 (i.e. rare), on considère que notre corrélation est significativement différente de zéro.

Étape 6 : Citer la taille de l'effet et son intervalle de confiance

La taille de l'effet dans cette analyse est la valeur de r trouvée. Vous n'aurez que très rarement à rapporter l'intervalle de confiance de cette corrélation, et dans tous les cas, vous n'aurez pas à la calculer manuellement. Néanmoins, si vous avez besoin d'y arriver, sachez que l'erreur-type de la corrélation se calcule comme suit :

$$s_r = \sqrt{\frac{(1 - r^2)}{(n - 2)}}$$

Dans R, le calcul basé sur cette erreur-type et la valeur de T pour n-2 degrés de liberté et un intervalle à 95 % nous donne un intervalle de confiance situé entre 0,64 et 0,92. Autrement dit, la vraie valeur de la population se situe probablement entre ces deux valeurs.

Comme discuté précédemment, ces valeurs sont rarement rapportées. On écrit habituellement ce genre de résultat comme suit : «La transparence de l'eau et la densité de poissons étaient fortement corrélées (r=0,83)».

Notez que l'adjectif "fortement" doit être utilisé en relation de ce que l'on connaît du système à l'étude. Une corrélation de 0,83 pour une mesure physique où l'on évalue par exemple l'expansion d'un métal avec la chaleur serait considérée comme extrêmement faible. Dans d'autre contextes, par exemple si l'on essaie de trouver à quoi est reliée à la richesse en espèces d'une communauté locale d'oiseaux, alors cette valeur serait extrêmement élevée, puisque l'on trouve rarement des valeurs > 0,30 à cause de la nature aléatoire de la composition des communautés d'une année à l'autre.

Aussi, méfiez-vous toujours des très fortes corrélations, particulièrement si vous arrivez à la valeur r = 1. À tout coup, il s'agit d'une erreur dans les calculs...

17.3. Labo : Corrélation de Pearson

Pour ce petit laboratoire, nous allons explorer la présence d'une association entre la longueur des ailes et la taille du corps chez les manchots de Palmer.

Étape 1 :

$$H_0 = \rho = 0$$
$$H_1 = \rho \neq 0$$

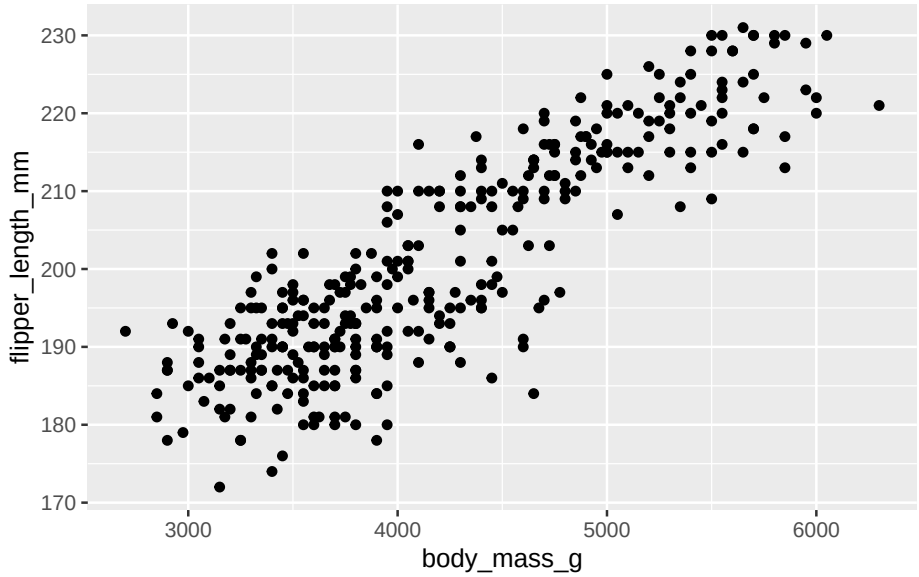
Étape 2 :

Les deux choses importantes à vérifier avant de lancer notre analyse de corrélation sont la linéarité de la relation et la normalité des deux distributions. Voici le code R pour y parvenir :

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(palmerpenguins)

propre <- penguins |>
  drop_na(body_mass_g, flipper_length_mm)

propre |>
  ggplot(aes(body_mass_g, flipper_length_mm)) +
  geom_point()
```

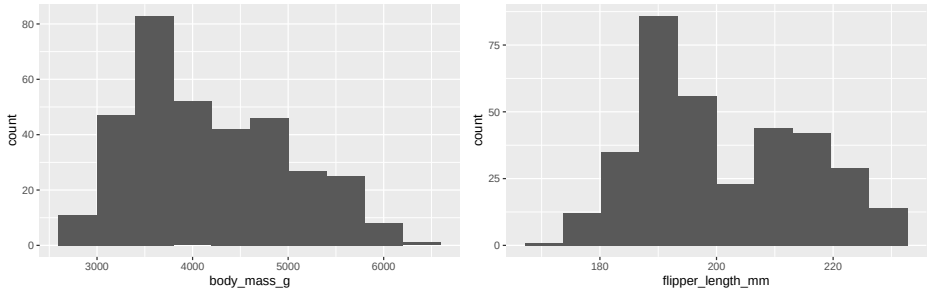


Il serait difficile d'obtenir une relation plus parfaitement linéaire!

Si jamais la relation avait été clairement non-linéaire, il aurait été préférable d'utiliser la corrélation de Spearman (voir Section 21.7) plutôt que celle de Pearson pour effectuer vos calculs.

```
propre |>
  ggplot(aes(body_mass_g)) +
  geom_histogram(bins = 10)
propre |>
  ggplot(aes(flipper_length_mm)) +
  geom_histogram(bins = 10)
```

Les deux distributions sont sufisamment normales pour continuer avec la corrélation de Pearson.



Une fois ces vérifications effectuées, on peut calculer la corrélation à l'aide la fonction **cor**, qui attend deux arguments, soit les deux vecteurs contenant nos données :

```
cor(propre$body_mass_g, propre$flipper_length_mm)
```

```
[1] 0.8712018
```

Donc le poids du corps et la taille des ailes sont fortement corrélés positivement chez les manchots de Palmer (r=0,87).

Enfin, si on veut tester si cette corrélation est significative ou non, on peut utiliser la fonction **cor.test**, qui attend les mêmes arguments :

```
cor.test(propre$body_mass_g, propre$flipper_length_mm)
```

```

    Pearson's product-moment correlation

data:  propre$body_mass_g and propre$flipper_length_mm
t = 32.722, df = 340, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal
to 0
95 percent confidence interval:
```

```
 0.843041 0.894599
sample estimates:
      cor
0.8712018
```

Nous avons donc la valeur de t (32,72), les degrés de liberté (340) et la valeur de p ($<2,2 \times 10^{-16}$) associés à notre valeur de corrélation, ainsi que l'intervalle de confiance à 95 %, qui va de 0,84 à 0,89.

Cette corrélation est donc clairement significativement différente de zéro. Notre intervalle de confiance est particulièrement précis!

17.4. Exercice : Corrélation de Pearson

Sachant que les insectes sont des ectothermes, nous allons voir avec cet exercice si la fréquence de chant (le nombre de stridulations par seconde) d'une espèce de grillons, la némobie striée, est reliée à la température du sol.

Voici les données que vous avez recueillies pour répondre à cette question :

Stridulations/seconde	Température (º F)
20.0	88.6
16.0	71.6
19.8	93.3
18.4	84.3
17.1	80.6
15.5	75.2
14.7	69.7
15.7	71.6
15.4	69.4
16.3	83.3

Stridulations/seconde	Température (º F)
15.0	79.6
17.2	82.6
16.0	80.6
17.0	83.5
14.4	76.3

Testez l'association entre ces deux variables à l'aide de la corrélation de Pearson.

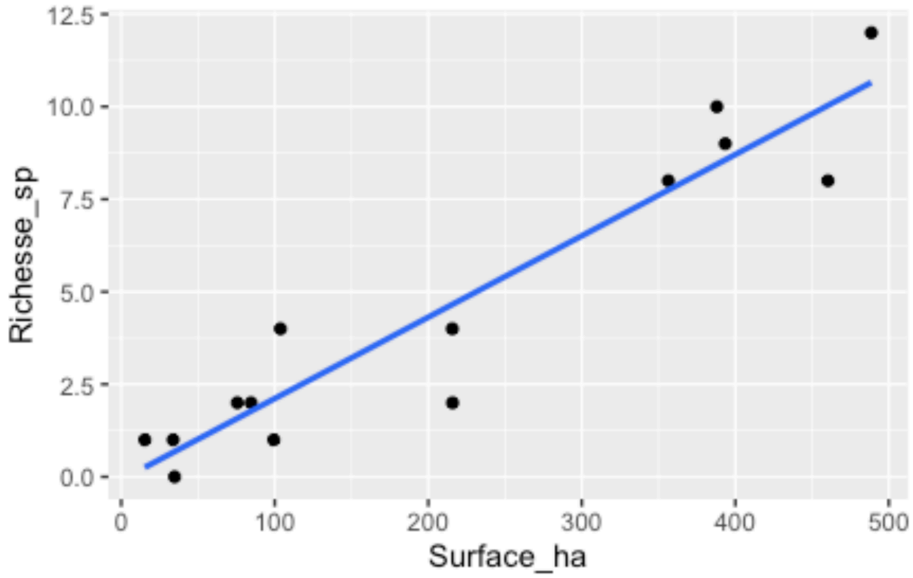
18. La régression linéaire

18.1. Présentation de la régression linéaire

Il vous arrivera souvent dans votre travail d'avoir besoin d'aller plus loin que le fait de savoir que deux variables quantitatives sont corrélées ensemble. Si nous avons en tête un modèle clair de cause à effet, il nous arrivera souvent de vouloir savoir par combien changera une variable expliquée si on modifie la variable explicative. De combien, par exemple, devons-nous réduire nos émissions de carbone pour réduire la température globale de 1°C ? Nous avons besoin de plus qu'un coefficient de corrélation pour répondre à cette question.

Dans ce chapitre, nous irons un peu plus en détail avec la régression linéaire qu'avec d'autres techniques statistiques, puisque beaucoup de techniques que nous verrons en sont des extensions, qui construiront sur les connaissances de ce chapitre. Vous remarquerez que je ne commencerai mon exemple avec des vraies données qu'à la partie Labo du chapitre, pour essayer d'alléger la présentation. J'espère que le message passera quand même bien.

Une question type de la régression linéaire pourrait par exemple être : combien d'espèces d'oiseaux pouvons-nous espérer ajouter à un parc urbain si on ajoute 100 hectares à sa surface protégée? La corrélation ne nous renseigne pas sur ce genre de chiffres. Ce qui nous intéresse ici est de savoir quelle est la **pente** qui relie nos deux variables.

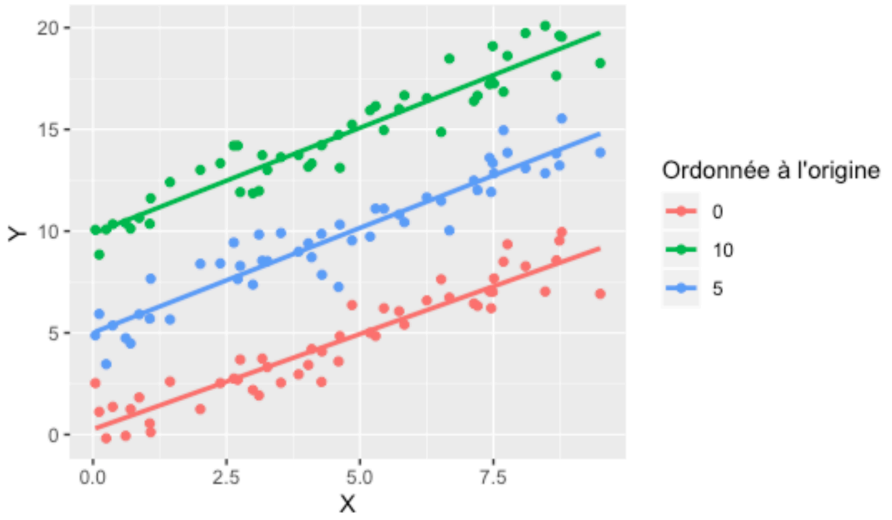


Si nous calculions une pente de 0,02, cela nous dirait que pour chaque hectare supplémentaire protégé, nous gagnons en moyenne 0,02 espèces. Si on ajoute 100 hectares au parc, on peut donc espérer gagner (0,02*100) 2 espèces. Connaître la pente qui relie deux variables nous permet de passer de quelque chose d'intéressant (une corrélation) à quelque chose de pratique, d'appliqué.

Évidemment, on ne peut pas garantir que notre parc gagnera effectivement ces deux espèces. Il y aura aussi du bruit autour de notre relation, c'est-à-dire d'autres variables qui peuvent aussi avoir une influence et dont nous n'avons pas tenu compte dans notre calcul. Il en gagnera peut-être un peu plus, ou peut-être un peu moins.

La régression linéaire comprend toujours un second paramètre (outre la pente), soit l'ordonnée à l'origine. L'**ordonnée à l'origine** nous informe sur la hauteur verticale à laquelle la pente croisera l'axe des Y lorsque X=0. Vous entendrez souvent aussi le terme anglais *intercept* pour désigner

ce second paramètre. Voici un exemple d'une même pente, passant par trois ordonnées à l'origine différentes :



Notez qu'ici nos données s'étendent jusqu'à la valeur X=0, mais dans la plupart des cas, nos données n'iront pas jusque là. L'ordonnée à l'origine est alors le point où la pente croiserait l'axe des Y, si jamais elle s'y rendait.

Vous remarquerez que l'on peut rarement tirer des conclusions biologiques intéressantes de l'ordonnée à l'origine seule, c'est pourquoi nous nous y intéresserons un peu moins dans ce chapitre. Cependant, nous verrons au Chapitre 20 comment comparer l'ordonnée à l'origine entre deux pentes, ce qui peut devenir très intéressant.

18.2. Les assomptions de la régression linéaire

L'estimation de la pente entre deux variables quantitatives peut être effectuée avec une technique nommée la régression linéaire. Avant de la calculer, il importe cependant de s'assurer que nos données correspondent aux assomptions de la technique, qui se comptent au nombre de quatre, soit :

1. La normalité des erreurs
2. L'homogénéité de la variance
3. L'indépendance des observations
4. X est fixé par l'expérimentateur

Assomption 1 : Normalité des erreurs.

Comme pour le modèle d'ANOVA du Chapitre 15, l'assomption de normalité de la régression linéaire s'applique aux résidus du modèle. Il faut donc vérifier cette assomption après avoir effectué nos calculs plutôt que avant.

Les résidus d'une régression linéaire sont définis comme la distance verticale entre chaque point observé et la pente de la régression. Autrement dit, pour chaque observation, de combien notre modèle (i.e. la pente) se trompe par rapport à la réalité observée.

$$rsidu = observation - prdiction$$

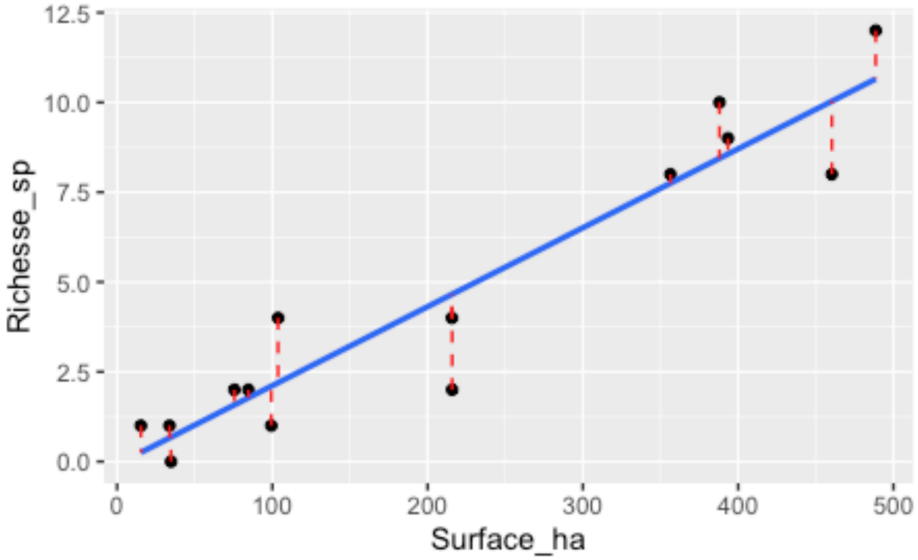


Figure 18.1.: Illustration du concept de résidus d'une régression linéaire. Plus la ligne pointillée rouge est longue, plus le résidu de cette observation est grand.

Plus ces distances sont courtes, plus notre modèle est "bon" (i.e. près de la réalité). Plus ces distances sont grandes, moins notre modèle est bon, plus il est loin de la réalité.

Bien qu'il faille vérifier cette assomption à la fin de la modélisation, il est cependant intéressant de savoir que généralement, les résidus seront distribués normalement si les variables X et Y le sont aussi.

C'est donc toujours une bonne idée de commencer par explorer les histogrammes de fréquences de nos variables avant de débuter l'analyse. Aussi, comme pour beaucoup de tests dans les chapitres précédents, la régression linéaire est relativement robuste aux écarts de normalité. Inutile d'utiliser un test strict de normalité comme celui de Shapiro-Wilk.

Si jamais vos données sont vraiment loin de la normale, n'hésitez pas à utiliser des transformations comme expliquées au Chapitre 9.

Assomption 2 : Homogénéité de la variance

Nous avons énoncé le concept d'homogénéité de la variance au Chapitre 13 sur les tests de comparaison de variance. Dans la régression linéaire, les choses sont cependant un peu plus complexes. On parle ici de variance à travers un gradient plutôt que de variance entre des groupes.

Ce qu'il faut surveiller c'est si le nuage de points a à peu près la même épaisseur sur toute sa largeur ou si il est clairement plus étroit à un bout qu'à un autre.

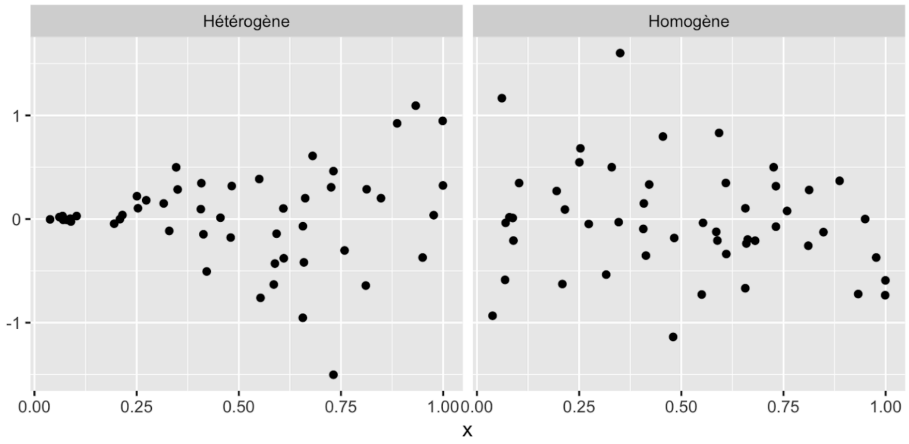


Figure 18.2.: Nuages de point montrant à gauche la variance hétérogène, et à droite la variance homogène.

Ce qu'il est important de savoir est qu'il n'existe pas de test statistique pour déterminer si la variance est suffisamment homogène dans un gradient. Il faudra tranquillement, au fil de vos analyses, apprendre à déter-

miner à l'œil si la variance est suffisamment homogène pour être utilisée dans une régression. Les transformations normalisant et linéarisant les relations ont habituellement aussi pour effet d'homogénéiser la variance. Réglez donc d'abord ces deux problèmes avant de vous attaquer à l'homogénéité de la variance. Si jamais rien n'y fait, sachez qu'il existe aussi une façon de tenir compte de ce changement de variance dans un type de modèle nommé GLS (Generalized Least Squares).

Assomption 3 : Indépendance des observations.

Comme pour toutes les autres techniques vues jusqu'à présent, la régression linéaire assume aussi que les observations sont indépendantes les unes des autres. Ce qui veut dire que la régression linéaire n'est PAS appropriée si on évalue la croissance d'un seul individu au fil du temps, de la richesse en espèces d'arbres dans un parc au fil des années, etc. Il existe des modèles appropriés pour ce genre de question (par exemple les modèles autorégressifs), qui dépassent le cadre de ce livre.

Assomption 4. X est fixé par l'expérimentateur.

Cette quatrième assomption est celle dont on discute habituellement le moins. Lorsque la régression linéaire a été inventée, elle a été pensée pour des cas où la personne préparant l'expérience avait le contrôle sur les valeurs de la variable explicative (i.e. en X), p. ex. en contrôlant le pH dans une éprouvette, la quantité de nutriments dans un aquarium, etc. La régression linéaire assume que X est mesurée parfaitement, et que toute l'erreur dans nos mesures provient de la variable en Y.

Dans la réalité, en biologie-écologie, c'est rarement le cas. La variable en X est souvent un estimé, p. ex. si on met en X la densité d'arbres, le nombre de prédateurs, etc. On aura rarement recensé ces informations, on se basera sur un estimé.

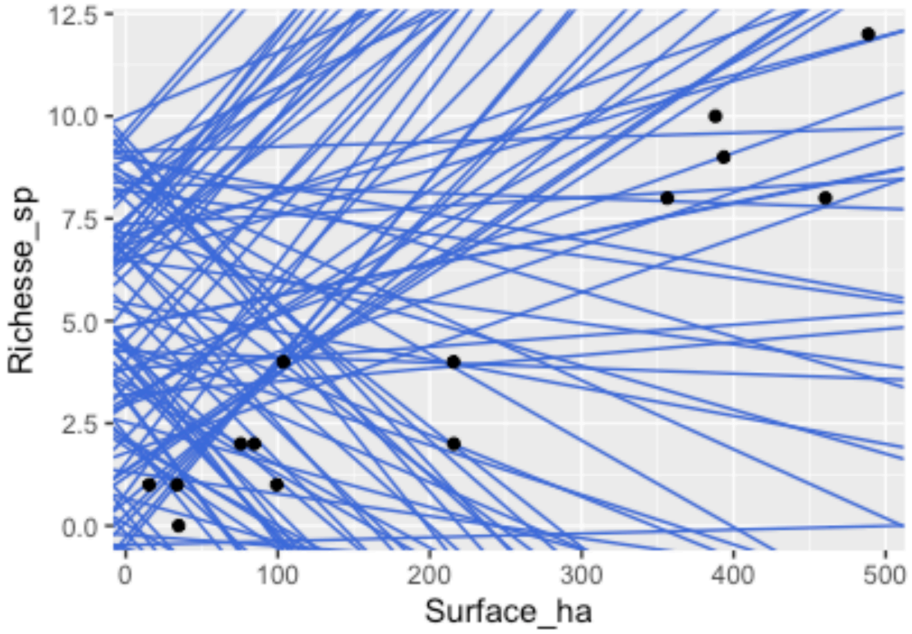
Cette assomption est rarement discutée parce qu'elle n'affecte pas directement l'ampleur des résidus du modèle, ni la précision des prédictions qui résultent de notre modèle. Elle peut cependant affecter la va-

leur comme tel de la pente et de l'ordonnée à l'origine. Faisant changer une au dépens de l'autre.

Cette contrainte devient importante dans les cas où l'interprétation de la valeur de la pente a une importance écologique en soi. Par exemple, si nous étudions la relation entre la longueur et le poids de poissons (en transformant en général nos variables à l'échelle logarithmique), une pente plus petite que 3 nous informera que les poissons ont tendance à s'étirer en grandissant, alors qu'une pente plus grande que 3 nous informe que les poissons deviennent plus costauds en grandissant. Ce genre de relations, que l'on nomme **relations allométriques**, doit être traité avec un autre type de modèle, soit les régressions de type II, que nous ne verrons pas dans ce livre. Si vous n'êtes pas dans ce genre de cas pointus, vous n'avez pas à vous soucier de cette quatrième assomption.

18.3. Les calculs de la régression

La question qu'il convient de se poser à ce point est : comment déterminer où passera notre ligne de régression? Une façon bête d'y arriver serait d'essayer un paquet de valeurs de pentes et d'ordonnées à l'origine :



Pour chacune, on pourrait ensuite calculer les résidus et choisir la combinaison pente + ordonnée à l'origine qui présente les plus petits résidus. Si on lançait quelques milliers de pentes de ce genre, on arriverait toujours à une valeur très proche de la pente idéale… mais ce serait fichtrement long.

Heureusement, il existe une technique nommée la **méthode des moindres carrés**, qui d'un seul petit calcul nous fournit à tout coup la pente et l'ordonnée à l'origine idéale. Elle se nomme ainsi parce que son but est de minimiser la somme des carrés des résidus.

Les résidus sont mis au carré dans le calcul pour deux raisons : d'abord, cela permet d'arriver à une somme intéressante. Si on ne met pas les résidus au carré, comme on a toujours environ la moitié des observations au-dessus de la pente et l'autre moitié en dessous, notre somme des

résidus sera toujours très proche de zéro, ce qui n'apporterait aucune information.

L'autre chose que la mise au carré des résidus permet est de pénaliser les grands résidus de façon plus importante. Mettre un petit nombre au carré ne fait pas beaucoup de différence, mais pour un grand, ça en fait une énorme. Pensez par exemple que $2^2 = 4$, alors que $8^2 = 64$. Un résidu quatre fois plus grand, une fois mis au carré donne une pénalité 16 fois plus grande.

Des mathématiciens ont montré que la pente qui fournira la plus petite somme des résidus (au carré) sera toujours donnée par :

$$b_1 = r\left(\frac{s_y}{s_x}\right)$$

Autrement dit, la meilleure pente (b_1 pour l'échantillon et β_1 pour la population, i.e. la lettre grecque bêta) est toujours trouvée en multipliant la corrélation entre deux variables par le ratio de leurs écart-types.

Ensuite, on peut déterminer l'ordonnée à l'origine de notre régression, comme ceci :

$$b_0 = \bar{y} - b_1\bar{x}$$

Où y-barre et x-barre sont les moyennes de chacune de ces variables. Les paramètres de la régression sont nommés b_0 et b_1 parce vous verrez dans les chapitres sur la modélisation linéaire que l'on peut ajouter des variables supplémentaires à notre modèle, pour tenir compte d'autres facteurs, Ces derniers se nommeraient à ce moment b_2, b_3, etc. Notez que dans ces modèles, les équations pour déterminer les pentes et l'ordonnée à l'origine deviennent cependant beaucoup plus complexes que celles présentées ci-haut.

Si vous connaissez la notation classique d'une pente comme étant :

$$y = ax + b$$

il est facile de faire le parallèle avec cette équation, car pour la régression, la pente est définie comme ceci :

$$y = b_0 + b_1 x$$

Remarquez que nous n'avons malheureusement qu'un seul mot "pente" pour désigner à la fois l'équation de la pente en entier (y=ax+b) et le paramètre de pente comme tel (a). Désolé pour la possible confusion...

18.4. Le coefficient de détermination (r^2)

Une fois que nous avons déterminé quelle était la meilleure pente pour définir nos données, il peut être intéressant de savoir à quel point cette pente est une bonne représentation de la réalité. C'est ce rôle que joue le **coefficient de détermination**, qui est souvent abrégé comme r^2 ou R^2. Ce dernier varie entre 0 et 1, et nous indique à quel point la réalité varie autour de notre pente. Voici quelques exemples de r^2 pour une pente de régression identique où b_1=5 :

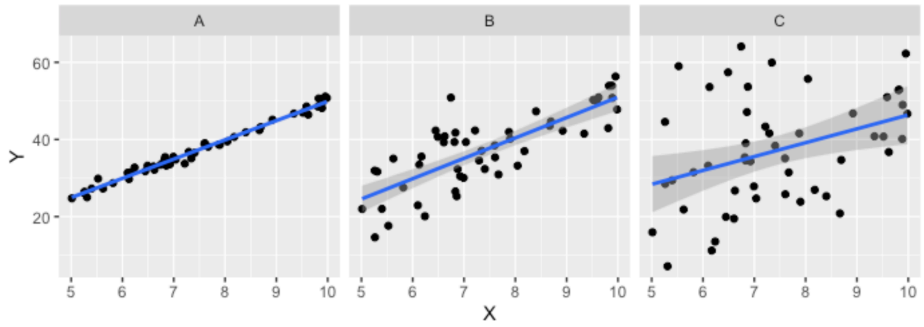


Figure 18.3.: La régression A possède un r^2 de 0,98, la B de 0,62 et la C de 0,14.

Il existe plusieurs façons de calculer le r^2 d'une régression, qui donnent toutes exactement le même résultat. La première (et la plus intuitive dans un contexte de régression simple à deux variables) et de mettre la corrélation entre les deux variables au carré (d'où l'abréviation de r^2...).

Par contre, pour comprendre l'interprétation du r^2, il peut être utile de connaître la deuxième façon de le calculer, soit en faisant 1 moins le rapport entre la variabilité résiduelle (i.e. le bruit dans notre modèle) et la variabilité totale de Y, soit :

$$r^2 = 1 - \frac{SS_{rsidus}}{SS_{totale}}$$

Comme discuté au Chapitre 15, l'abréviation SS désigne ici la somme des carrés (*Sum of Squares*). La partie totale de cette somme des carrés est le numérateur de la variance, soit :

$$SS_{totale} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

Alors que celle des résidus peut s'écrire comme ceci :

$$SS_{rsidus} = \sum_{i=1}^{n}(y_i - \hat{y_i})^2$$

Où $\hat{y}_1$ est la prédiction pour chacune de nos observation. Cette prédiction se calcule à l'aide de l'équation de la régression, dans laquelle on place une à une les valeurs de X (nous reviendrons dans la dernière section de ce chapitre sur le calcul des prédictions).

Une fois cette formule maîtrisée, il est intuitif de comprendre que le r^2 représente la fraction de la variance de Y expliquée par notre modèle. Notre calcul passe par contre par un mini-détour, où on regarde 1 moins la fraction non expliquée (les résidus).

Notez bien la nuance entre les valeurs de r, qui peuvent aller entre -1 et 1 et informent du sens de la relation, alors que le r^2 est limité entre 0 et 1. Il n'informe pas du sens de la relation, mais peut s'interpréter directement comme la fraction de la variance de Y explicable par X, ce que ne permet pas r.

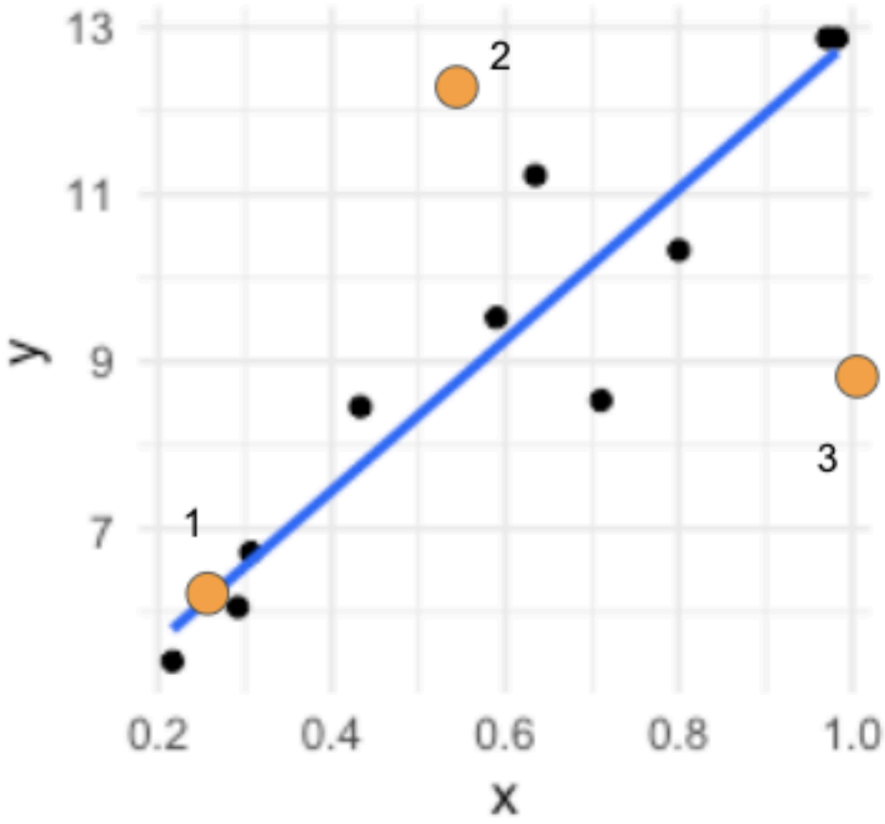
18.5. Inspecter un modèle de régression

Même si les assomptions de la régression sont respectées, il pourrait arriver que votre modèle soit tout de même biaisé ou non représentatif. C'est pourquoi il importe d'inspecter votre modèle avant de l'interpréter.

Pour bien effectuer cette inspection, nous avons besoin de connaître trois concepts. Le premier concept est celui de levier. Une observation aurait un fort **levier** dans votre modèle de régression si sa valeur en X est très éloignée de la moyenne de X.

Le deuxième concept à comprendre est l'**ampleur du résidu**. Comme nous avons discuté plus haut, nos observations ne sont jamais directement sur la pente : plus l'observation en est éloignée, plus l'ampleur de son résidu est grande.

Qu'une observation possède un grand levier, comme tel n'est pas un problème. Qu'une observation possède un grand résidu non plus n'est pas nécessairement un problème. Par contre, si une observation possède à la fois un grand levier ET un grand résidu, alors ça peut devenir un problème, on parle d'une observation avec de l'**influence**. C'est-à-dire qu'à elle seule, elle pourrait faire changer grandement la pente de notre modèle et changer nos conclusions biologiques.



Dans l'illustration précédente, les points 1 et 3 possèdent un fort levier. Les points 2 et 3 possèdent un fort résidu, mais seul le point 3 possède une grande influence, parce qu'il a à la fois un fort levier et un fort résidu.

Pour nous aider à évaluer l'influence d'une observation dans un modèle de régression, il existe une mesure nommée la **distance de Cook**. Cette distance est un chiffre qui intègre à la fois le levier et le résidu. Elle se calcule pour chacune de nos observations. Le calcul est plutôt complexe

et implique de calculer la différence de pente lorsque l'on enlève cette observation du modèle par rapport au modèle complet. Plus la valeur est élevée, plus cette observation est influente. On dit qu'en général si cette valeur est >1 pour une ou plusieurs de nos observations, il faut commencer à s'interroger sur notre modèle.

Et, on fait quoi si ça nous arrive? Comme pour la plupart des problèmes en statistiques, la première chose à faire est d'aller vérifier nos données avec notre carnet de notes. Il aurait pu arriver qu'une donnée soit mal saisie, etc. Ensuite, si la donnée était bonne, l'étape suivante est de reconnaître le problème et de le décrire comme tel dans nos résultats : que telle ou telle observation avait une distance de Cook > 1.

Ensuite, ce que je vous recommande est d'ajuster deux modèles de régression différents, un avec et l'autre sans cette observation problématique. Ainsi, vous pourrez informer votre lecteur de la force de l'influence de cette observation.

Si la pente est sensiblement la même, informez-en votre lecteur. Si ça change drastiquement les résultats, informez-en aussi votre lecteur et discutez des particularités de cette observation.

Est-ce possible que ce soit un individu égaré provenant d'une autre population, d'un échappé de captivité? Voyez cette observation d'un œil curieux, c'est souvent de cette façon que l'on découvre de nouvelles choses en biologie, en se posant des questions sur les exceptions.

18.6. La régression comme test statistique

Jusqu'à présent dans ce chapitre, je vous ai enseigné la régression linéaire surtout comme une technique de modélisation, i.e. une façon d'estimer des paramètres pour représenter la réalité. Elle peut cependant être vue aussi comme un test statistique, c'est ce que nous verrons dans cette section.

Comme pour toutes les techniques statistiques que nous avons vues jusqu'à présent, il pourrait arriver que l'on trouve par hasard dans nos échantillons une pente, alors que dans la population, il n'y en a pas. C'est pourquoi il existe un test pour savoir si la pente (et aussi l'ordonnée à l'origine) sont significativement différents de zéro (i.e. de l'absence de pente).

Il existe deux tests différents pour déterminer si une pente est significative, soit un basé sur une distribution de T et un basé sur distribution de F. Comme ce chapitre commence à s'étirer, nous ne verrons que celle basée sur la distribution de T, mais sachez que l'autre existe aussi.

Notre hypothèse nulle pour la pente sera qu'elle est égale à zéro, et notre hypothèse alternative qu'elle est différente de zéro.

$$H_0 : \beta_1 = 0$$
$$H_1 : \beta_1 \neq 0$$

Les hypothèses s'énonceraient de la même manière si l'on voulait tester l'ordonnée à l'origine :

$$H_0 : \beta_0 = 0$$
$$H_1 : \beta_0 \neq 0$$

La chose à savoir est qu'à l'aide d'un calcul un peu complexe, qui dépasse ce qu'il est raisonnable de faire manuellement dans ce cours, on peut obtenir à l'aide de la méthode des moindres carrés l'erreur-type associée à un paramètre. La statistique de test sera ensuite calculée comme ceci pour la pente :

$$t = \frac{b_1 - \theta}{S_{b_1}}$$

Autrement dit, comme pour le test de T à un échantillon (voir Chapitre 12), la statistique de t se calcule comme le ratio entre notre mesure d'intérêt (ici la pente) et son erreur-type. La valeur de θ (theta) sera zéro si vous voulez savoir si votre pente est différente de zéro, mais ce pourrait aussi être un autre chiffre si vous avez une bonne théorie à tester avec cela.

L'équation s'écrira de la même manière, mais avec b_0 si on veut plutôt tester l'ordonnée à l'origine.

Comme pour un test de t à un échantillon, on va ensuite trouver la probabilité de trouver une telle valeur de t avec n-2 degrés de liberté. Remarquez que l'on fait ici n-2 plutôt que n-1 puisque la régression linéaire estime deux paramètres, soit la pente et l'ordonnée à l'origine.

18.7. Labo : la régression linéaire

Mettons maintenant ensemble tous ces morceaux pour appliquer une régression linéaire à un jeu de données réel.

Sachant qu'il est important de bien pouvoir manoeuvrer dans l'eau pour s'alimenter, notre question sera la suivante : est-ce que le poids des manchots de Palmer augmente avec la taille de leurs ailes.

Remarquez bien qu'on a un sens pour la relation de cause à effet : la taille des ailes causera le poids.

Étape 1 : Définir les hypothèses

Nous nous intéresserons dans notre exemple uniquement à la question de la pente. Nos hypothèses statistiques seront donc les suivantes :

$$H_0 : \beta_1 = 0$$
$$H_0 : \beta_1 \neq 0$$

Autrement dit, notre hypothèse nulle est qu'il n'y a pas de pente qui relie les deux variables, et l'hypothèse alternative est qu'il en existe une.

Étape 2 : Explorer visuellement les données

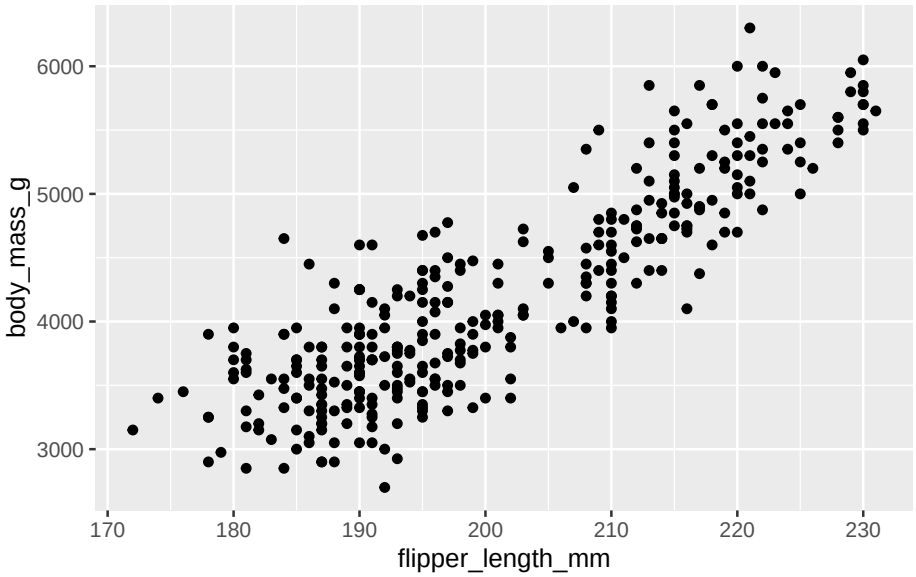
Connaissant les quatre assomptions de la régression linéaire, nous savons que nous ne pouvons pas les tester directement à cette étape. Cependant, il est toujours utile d'observer nos données dans un graphique pour se donner une idée de ce qui nous attend pour la suite. Nous préparerons aussi au préalable une version simplifiée de notre tableau de données de manchots, où nous avons enlevé les lignes contenant des valeurs manquantes pour les colonnes de cette analyse.

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(palmerpenguins)

labo_regression <- penguins |>
  drop_na(body_mass_g, flipper_length_mm)

labo_regression |>
```

```
ggplot(aes(flipper_length_mm, body_mass_g)) +
geom_point()
```



Remarquez d'abord que nous avons mis en X la variable explicative, celle qui causerait (selon notre hypothèse) le changement dans l'autre variable.

On constate (comme nous l'avions fait au Chapitre 17) que cette relation sera clairement linéaire et que la variance, à l'oeil, risque bien d'être homogène.

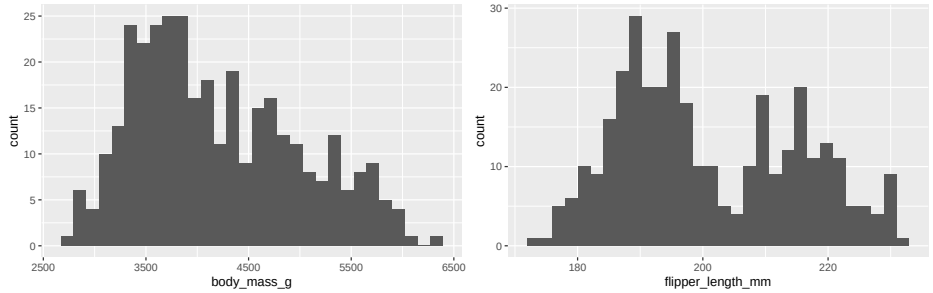
Ensuite, même si ce n'est pas un validation formelle de l'assomption de normalité des résidus, il est important de regarder la distribution de chacune des variables :

```
labo_regression |>
  ggplot(aes(body_mass_g)) +
  geom_histogram()
```

```
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```

```
labo_regression |>
  ggplot(aes(flipper_length_mm)) +
  geom_histogram()
```

```
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```



On constate que la distribution de la longueur des ailes est peut-être bimodale. Il faudrait garder un oeil attentif sur la distribution des résidus (qui sont le test formel de normalité)

Étapes 3 et 4 : Calculer la statistique de test (ici, calculer tout le modèle de régression)

Dans R, la régression linéaire utilise la même notation de formule que celle que nous avons vu pour l'ANOVA, avec le symbole ~ (tilde) qui sépare la partie expliquée de la partie explicative. Donc, pour que ce soit bien

clair, la partie à gauche de la formule est la variable expliquée (le Y de nos graphiques) et la partie à droite est la variable explicative (le X dans nos graphiques).

Notre modèle s'ajusterait donc comme ceci :

```
m <- lm(body_mass_g ~ flipper_length_mm, data =
 ↪  labo_regression)
```

Autrement dit, on calcule une régression linéaire (`lm` pour *Linear Model*) de la taille du corps en fonction de la longueur des ailes, et on conserve le résultat dans l'objet nommé `m`.

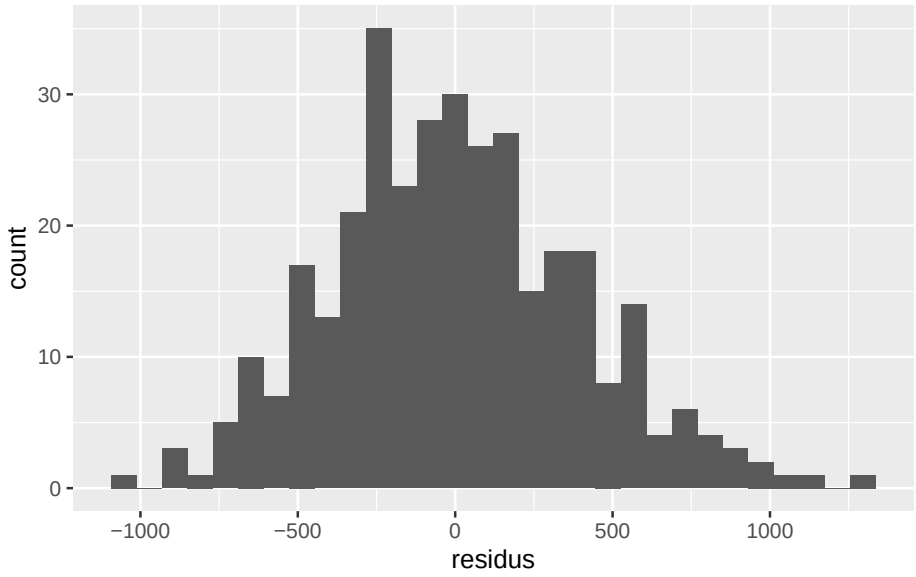
Avant de se lancer dans les résultats, il faut d'abord s'assurer que notre modèle est approprié et non-biaisé. Il faut donc vérifier nos assomptions (normalité des résidus et homogénéité de la variance) et la qualité du modèle (i.e. la présence d'observations influentes). Pour se faire, le plus simple est d'ajouter une colonne à notre base de données contenant le résidu et la prédiction de chacune des observations, comme ceci :

```
labo_regression <-
  labo_regression |>
  mutate(
    residus = resid(m),
    predictions = predict(m)
  )
```

On peut ensuite tracer leur histogramme comme pour n'importe quelle variable :

```
labo_regression |>
  ggplot(aes(residus)) +
  geom_histogram()
```

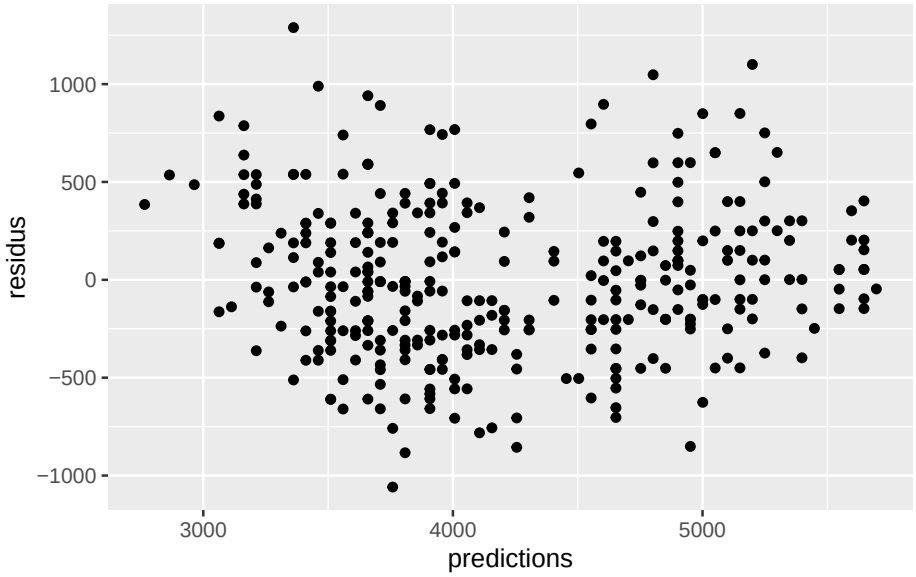
```
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```



On voit que la distribution est clairement normale.

Pour valider l'homogénéité de la variance, on peut tracer un nuage de points avec en X les valeur observées et en Y les résidus, comme ceci :

```
labo_regression |>
  ggplot(aes(predictions, residus)) +
  geom_point()
```



En général, les résidus sont relativement homogènes.

Enfin, il faut aussi vérifier si nous avons des observations influentes qui pourraient changer fortement notre modèle. Pour se faire, il existe dans R la fonction `cooks.distance` qui effectue pour nous le calcul de la distance de Cook pour chacune des observations. On peut ajouter ces valeurs à notre tableau de données aussi, comme cela :

```
labo_regression <-
  labo_regression |>
  mutate(
    D = cooks.distance(m)
  )
```

Et ensuite, appliquer un filtre pour voir si des observations ont une valeur de Cook plus grande que 1 :

```
labo_regression |>
  filter(D > 1)
```

```
# A tibble: 0 x 11
# i 11 variables: species <fct>, island <fct>,
#   bill_length_mm <dbl>, bill_depth_mm <dbl>,
#   flipper_length_mm <int>, body_mass_g <int>,
#   sex <fct>, year <int>, residus <dbl>,
#   predictions <dbl>, D <dbl>
```

Dans notre cas, tout va bien, aucune observation n'a d'influence importante. On peut donc se lancer dans l'interprétation des résultats.

Pour se faire, appelons la fonction `summary` sur notre objet `m` pour avoir un aperçu des calculs :

```
summary(m)
```

```
Call:
lm(formula = body_mass_g ~ flipper_length_mm, data =
labo_regression)

Residuals:
     Min       1Q   Median       3Q      Max
-1058.80  -259.27   -26.88   247.33  1288.69

Coefficients:
                   Estimate Std. Error t value
(Intercept)       -5780.831    305.815  -18.90
flipper_length_mm    49.686      1.518   32.72
                  Pr(>|t|)
(Intercept)         <2e-16 ***
```

```
flipper_length_mm   <2e-16 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 394.3 on 340 degrees of freedom
Multiple R-squared:  0.759, Adjusted R-squared:  0.7583
F-statistic:  1071 on 1 and 340 DF,  p-value: < 2.2e-16
```

Il s'agit probablement de votre sortie de R la plus longue jusqu'à maintenant, mais en y allant morceau par morceau, vous verrez que c'est relativement simple au final. D'abord, la première ligne (*call*) nous rappelle comment la fonction a été appelée, avec quelles variables, quel tableau de données, etc.

Ensuite, la section *Residuals* nous informe de quelques statistiques concernant les résidus. Comme nous les avons déjà explorés visuellement, on peut ignorer cette partie.

Ensuite, la section *Coefficients* nous montre les paramètres estimés par la modèle. La ligne (*Intercept*) correspond à l'ordonnée à l'origine (b_0) et la ligne flipper_length_mm correspond à la pente pour cette variable (b_1). Pour chacun de ces paramètres, R nous fournit quatre nombres, qui sont dans l'ordre : l'estimé du paramètre (*Estimate*), son erreur-type (*Std. Error*), la valeur de t (*t value*) et la valeur de p (*Pr*).

La seule autre information que nous utiliserons de ces sorties est celle nommée Multiple R-squared, qui nous fournit la valeur de r^2 de notre modèle, ici 0,76. À elle seule, la taille des ailes explique 76% de la variabilité dans le poids des manchots.

La colonne Estimate nous fournit donc les estimés des paramètres, soit l'ordonnée à l'origine de notre régression (-5780,8) et sa pente (49,7). L'équation de notre régression est donc :

$$body\_mass\_g = -5780,8 + 49,7 \times flipper\_length\_mm$$

Pour chaque mm d'aile de plus, un manchot pèse 49,7 g supplémentaires.

Remarquez la difficulté d'interpréter l'ordonnée à l'origine. Elle nous dit qu'un manchot avec des ailes de 0 mm pèserait -5780 g.

Étape 5 : Rejeter ou non l'hypothèse nulle.

Puisque la valeur de p associée à notre paramètre de pente est très faible (i.e. rare), soit < 2 x 10^{-16}, on peut rejeter l'hypothèse nulle de l'absence de pente, puisque ce serait très rare d'avoir une pente aussi grande dans notre échantillon si les populations n'étaient pas reliées. Vous pouvez valider votre interprétation de la valeur de p grâce aux petites étoiles à côté de la valeur. Lorsqu'il y a une ou plusieurs étoiles, R considère cette valeur de p comme significative au seuil de 0,05.

Étape 6 : Citer la taille de l'effet et son intervalle de confiance

La taille de l'effet ici est l'ampleur de la pente, donc de 49,7. Vous pouvez évaluer "au pif" son intervalle de confiance à 95 % en faisant 49,7 +/- 2 * 1,51 (l'erreur-type de ce paramètre). L'intervalle de confiance se situerait donc quelque part entre 46,65 et 52,72. Vous pouvez aussi demander à R de calculer le vrai intervalle de confiance pour vous avec la fonction `confint` (*CONFidence INterval*), comme ceci :

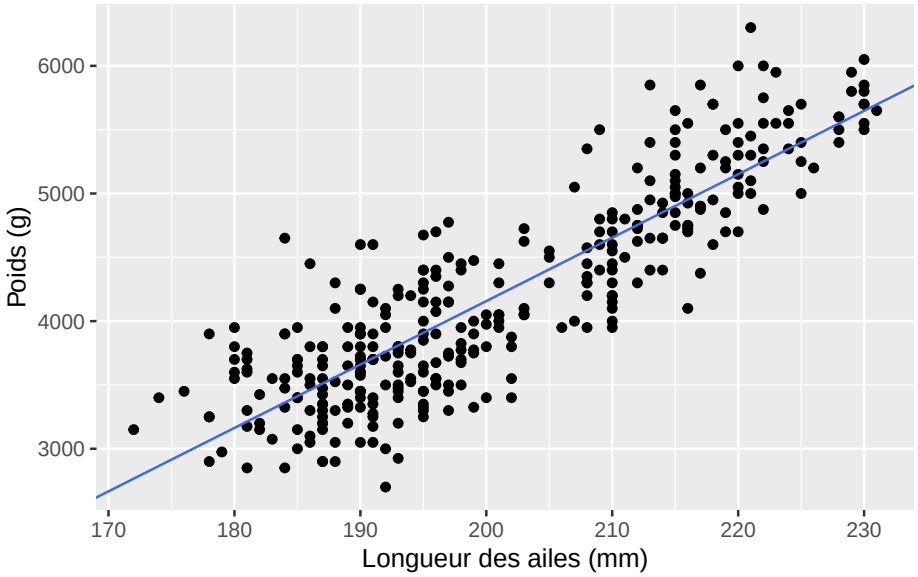
```
confint(m)
```

```
                      2.5 %      97.5 %
(Intercept)       -6382.35801 -5179.30471
flipper_length_mm    46.69892    52.67221
```

Nous pourrions donc écrire notre résultat comme ceci : «La longueur des ailes des manchots de Palmer expliquait de façon significative leurs poids (b_1 = 49,69, t_{340} = 32,72, p < 2 x 10^{16}). L'intervalle de confiance à 95 % de cette pente allait de 46,70 à 52,67. La longueur des ailes expliquait ainsi une partie importante de la variance du poids (r^2 = 0,76)»

Enfin, il peut être intéressant pour un calcul comme la régression linéaire d'illustrer la pente à travers nos données. Pour se faire, on peut tracer à nouveau un nuage de point, et ajouter une couche de pente (`geom_abline`) et lui fournissant les paramètres calculés par notre modèle de régression :

```
labo_regression |>
  ggplot(aes(flipper_length_mm, body_mass_g)) +
  geom_point() +
  geom_abline(slope = 49.69, intercept = -5780.83,
  ↪  color = "royalblue") +
  labs(
    x = "Longueur des ailes (mm)",
    y = "Poids (g)"
  )
```



Remarquez que, comme je l'ai fait ici, si ce graphique est destiné à la publication, il importe de bien nommer les axes et de fournir leurs unités.

18.8. Les prédictions de la régression

À l'aide de l'équation de la régression, il est facile de déterminer la valeur que notre régression prédirait pour une valeur de X donnée. La notation formelle de ces prédictions est celle-ci :

$$\hat{y}_i = b_0 + b_1 x_i$$

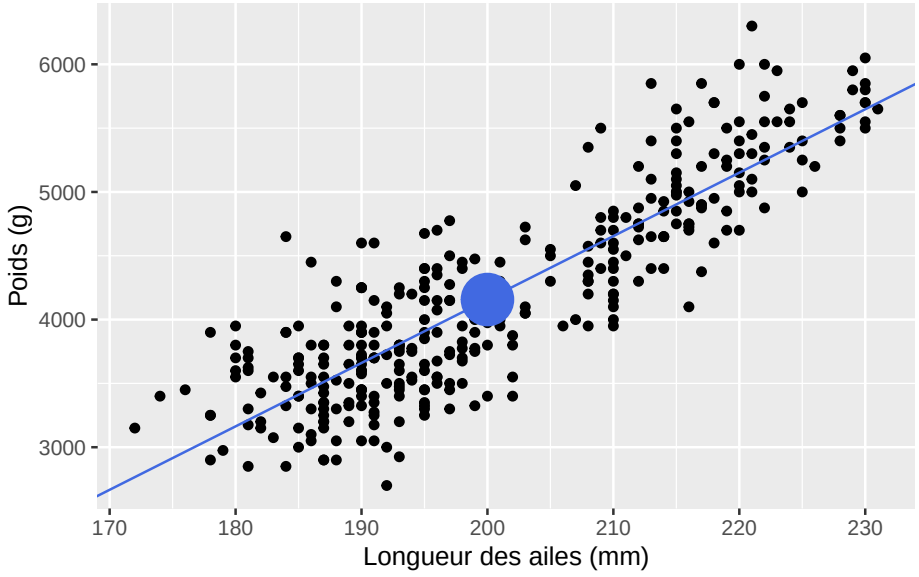
Verbalement, on pourrait décrire cette équation comme : pour prédire une valeur de Y, il faut multiplier la pente par la valeur de X, puis ajouter

l'ordonnée à l'origine. Si on choisit une série de valeurs de X et que l'on prédit leur valeurs pour ensuite les relier, on retrouvera invariablement la pente.

Si on revient à notre exemple en R, si on veut savoir quel serait le poids d'un manchot ayant des ailes de 200 mm, on ferait donc le calcul suivant :

$$Poids = -5780,83 + 49,69 \times 200mm = 4157,17g.$$

Après avoir fait un calcul de ce genre, c'est toujours une bonne idée d'aller se valider avec le graphique pour voir si notre calcul de prédiction est correct. Dans notre cas, le point que l'on prédit se retrouverait ici :



Dans tous les cas, notre prédiction devrait se retrouver directement sur la pente. Si ce n'est pas le cas, on a une erreur soit dans le traçage de la pente, soit dans le calcul de la prédiction.

Ça, c'était la partie facile! La partie plus compliquée maintenant est de savoir, quelle est notre confiance dans cette prédiction?

Remarquez bien d'abord dans le graphique précédent que près de notre point de prédiction pour 200 mm, il existe des manchots ayant des poids allant environ de 3400 g à 4700 g. On pourrait donc créer, à partir de cette information un premier intervalle, que l'on nomme l'**intervalle de prédiction**. Ce dernier nous informe de ce que l'on pourrait trouver vraiment sur le terrain. Si on parle de l'intervalle de prédiction à 95 %, il nous dit que 95 % des observations devraient se trouver entre ces deux bornes. Pour notre exemple, cet intervalle à 95 % irait de 3379 g à 4932 g (je vous épargne ici le calcul, attardez vous au principe et à l'interprétation).

Nous pourrions par contre avancer une autre information à partir de cette valeur de 4157 g. Nous pourrions nous demander à quel point la pente, à cet endroit précis, est proche de la vraie valeur de pente de la population. Vous vous rappelez que plus haut, nous avions vu que la méthode des moindres carrés nous fournissait une erreur-type sur la pente et une erreur-type sur l'ordonnée à l'origine. On serait donc en droit de se demander quelle est notre incertitude sur le 4157 g, compte tenu de ces deux erreurs combinées. On parlerait alors d'**intervalle de confiance de la moyenne**. Cet intervalle dans notre cas se trouve entre 4114,25 g et 4198,31 g.

Remarquez que ce deuxième intervalle est beaucoup plus étroit que le premier, car il ne représente pas la même réalité que le précédent. L'intervalle de prédiction nous informe de la fourchette de valeurs possible de trouver sur le terrain. L'intervalle de confiance de la moyenne lui nous informe sur la certitude de notre pente à cet endroit. L'intervalle de confiance de la moyenne peut être réduit si notre taille d'échantillon augmente, mais l'intervalle de prédiction, lui, devrait demeurer sensiblement le même.

Il est très important, lorsque nous faisons des prédictions à l'aide de notre modèle de régression, de se limiter à l'intervalle de valeurs de X que nous avons utilisé pour ajuster notre modèle de régression. Nous

ne savons pas comment se comporte notre phénomène à l'extérieur de ces bornes. Vous constaterez au fil de votre cheminement en biologie-écologique que peu de phénomènes sont entièrement linéaires. Ils ne le sont souvent que sur une partie du spectre de valeurs, et montrent des accélérations et des ralentissements nonlinéaires dans les extrémités.

18.9. Labo : Prédictions de la régression linéaire

Pour prédire de nouvelles valeurs à partir d'un modèle de régression, il existe dans R la fonction `predict`. Cette dernière s'attend à recevoir au moins deux arguments, soit le modèle de régression déjà ajusté, et un tableau de données sur lequel effectuer les prédictions. Si on continue le laboratoire précédent, on pourrait par exemple faire notre prédiction pour des ailes de 200 mm comme ceci :

```
predict(m, data.frame(flipper_length_mm=200))
```

```
       1
4156.282
```

La clé étant que pour que la prédiction fonctionne bien, notre variable en X doit être écrite exactement comme dans notre tableau de données original.

Remarquez que cette valeur est légèrement différente de celle calculée ci-haut, puisque nous avions arrondi les paramètres, alors que le calcul de R conserve toutes les décimales possibles.

Si l'on veut connaître l'intervalle de prédiction de cette valeur, il faut ajouter un argument supplémentaire pour en informer R, comme ceci :

```
predict(m, data.frame(flipper_length_mm = 200),
 ↪  interval = "prediction")
```

```
       fit      lwr      upr
1 4156.282 3379.612 4932.951
```

La valeur *fit* nous informe de la valeur de notre prédiction, alors que *lwr* (*LoWeR*) et *upr* (*UPpeR*) nous informent des bornes inférieures et supérieures de notre intervalle de prédiction. Par défaut R nous fournit un intervalle à 95 %, mais on peut changer cette valeur à l'aide d'un argument supplémentaire. L'intervalle de prédiction à 99 % s'obtiendrait comme ceci :

```
predict(m, data.frame(flipper_length_mm = 200),
 ↪  interval = "prediction", level = 0.99)
```

```
       fit      lwr      upr
1 4156.282 3133.459 5179.105
```

Enfin, si on veut obtenir l'intervalle de confiance de la moyenne plutôt que l'intervalle de prédiction, il faut changer l'argument `interval` pour ceci :

```
predict(m, data.frame(flipper_length_mm = 200),
 ↪  interval = "confidence")
```

```
       fit      lwr      upr
1 4156.282 4114.257 4198.307
```

18.10. Labo optionnel : tracer l'intervalle de confiance d'un modèle de régression.

Il peut être intéressant de représenter sur notre graphique l'incertitude associée à notre pente. Comme nous l'avons vu dans les sections précédentes, nous pourrions tracer deux intervalles différents, soit l'intervalle de prédiction ou l'intervalle de confiance, mais la stratégie pour le faire est exactement la même.

La clé pour y parvenir est de savoir que si l'on ne fournit pas de nouveau tableau de données à la fonction `predict`, elle nous fournira une valeur pour chacune des données en X de notre tableau original. On pourra ensuite relier chacune de ces prédictions à l'aide d'une ligne pour tracer l'intervalle.

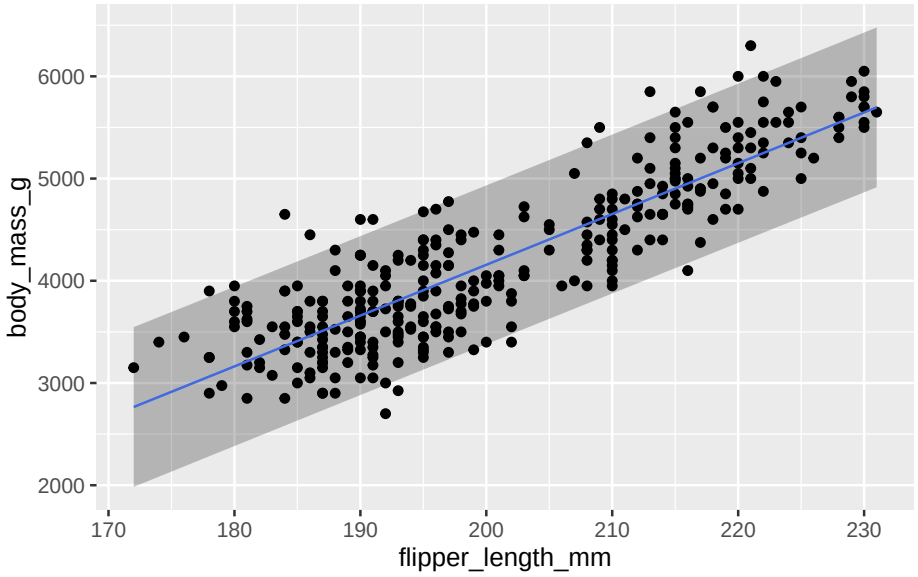
Pour connecter le tableau de données de prédictions avec l'original, nous utiliserons la fonction `bind_cols`, qui connecte les colonnes de plusieurs tableaux ensemble. Soyez par contre très prudents avec cette fonction, car si vos tableaux de sont pas parfaitement dans le même ordre et avec les mêmes lignes, nous n'allez pas connecter les bonnes valeurs ensemble. Une fois les données prêtes, nous utiliserons trois couches de données pour préparer notre graphique, soit un `geom_point` pour les données, un `geom_line` pour la pente et un `geom_ribbon` (i.e. littéralement un ruban) pour l'intervalle de confiance. Remarquez que le `geom_ribbon` reçoit 2 valeurs de Y, celle du haut et celle du bas de la bande. Aussi, remarquez l'ordre dans lequel j'ai ajouté les couches. Si jamais vous ajoutez la couche ribbon en dernier, cette dernière cachera toutes les autres...

Voici donc le code pour l'intervalle de prédiction :

```
labo_regression |>
  bind_cols(predict(m, interval = "prediction")) |>
  ggplot(aes(x = flipper_length_mm, y = body_mass_g)) +
```

```
  geom_ribbon(aes(ymin = lwr, ymax = upr), alpha = 0.3)
   ↪  +
  geom_point() +
  geom_line(aes(y = fit), color = "royalblue")
```

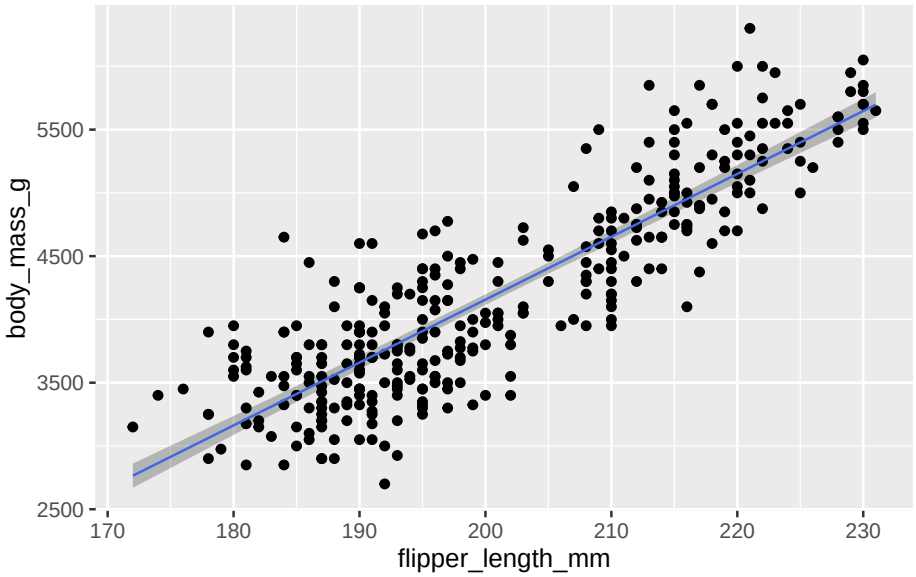
```
Warning in predict.lm(m, interval = "prediction"):
predictions on current data refer to _future_ responses
```



Et celui pour l'intervalle de confiance de la moyenne

```
labo_regression |>
  bind_cols(predict(m, interval = "confidence")) |>
  ggplot(aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_ribbon(aes(ymin = lwr, ymax = upr), alpha = 0.3)
   ↪  +
```

```
  geom_point() +
  geom_line(aes(y = fit), color = "royalblue")
```



Remarquez que l'intervalle de prédiction est beaucoup plus large, puisqu'il doit contenir 95 % des données. L'intervalle de confiance est plus étroit. Notez aussi que ce dernier est plus étroit au centre (on est vraiment certains que la pente passe là) et plus incertain dans les extrémités.

Petit fait intéressant : à tout coup, lorsque notre pente sera significativement différente de zéro, il ne sera pas possible de passer une ligne horizontale dans l'intervalle de confiance sans la toucher ou dépasser.

18.11. Récapitulatif

Si on fait un petit récapitulatif des tests vus jusqu'à présent, vous êtes maintenant en mesure de gérer les situations suivantes :

- Pour comparer une moyenne à une valeur cible : **Test de T à un échantillon**
- Pour comparer deux moyennes entre-elles : **Test de T à deux échantillons** ou **test de Welch**, selon que la variance est égale ou non. Possible aussi de le faire avec l'**ANOVA** si les variances sont égales.
- Pour comparer la moyenne de données pairées : **Test de T pairé**
- Pour comparer la variance de deux échantillons : **Test de F**.
- Pour comparer la moyenne de plus de 3 moyennes : **ANOVA**.
- Pour regarder la relation entre une variable quantitative et une variable qualitative : **ANOVA**.
- Pour vérifier si deux variables quantitatives sont associées : **corrélation**
- Pour quantifier le lien de cause à effet entre deux variables quantitatives : **régression linéaire**.

18.12. Exercice : La régression linéaire

Pour vivre en pratique à quoi peut ressembler la régression linéaire je vous demande, à l'aide d'une analyse de régression linéaire, de quantifier le lien de cause à effet entre la température et la fréquence de stridulation chez la némobie striée. Vous pouvez réutiliser la base de données présentée à la Section 17.4. Par ailleurs, vous savez par vos connaissances de la littérature que la température affecte la fréquence de stridulation, et jamais l'inverse.

Une fois votre modèle prêt, utilisez-le pour prédire la température en degrés Celcius, si jamais une némobie produit 18 stridulations par seconde. Validez ensuite cette prédiction sur un graphique.

Quelle est l'intervalle de confiance à 99% de cette prédiction?

19. La comparaison de proportions

Dans ce chapitre, nous discuterons de l'analyse des variables qualitatives. Vous vous rappelez peut-être qu'au Chapitre 3, nous avons vu qu'une façon de travailler avec les variables qualitatives était de compter le nombre de fois que chacune des combinaisons de valeurs revenait, et d'illustrer ces informations dans un diagramme à bandes ou un tableau de contingence.

On pourrait par exemple se demander si l'efficacité de la vaccination contre la rage chez les ratons-laveurs est affectée par la saison. Pour notre expérience, nous avons bombardé une région de vaccins oraux contre la rage enrobés de nourriture. Nous avons ensuite capturé autant de ratons-laveurs que possible et analysé en laboratoire si ils étaient atteints ou non du virus. La base de données qui en résulterait, selon les principes des données propres, pourrait ressembler à ceci :

Individu	Saison	Rage
A	Printemps	Positif
B	Printemps	Négatif
C	Printemps	Positif
D	Automne	Positif
E	Automne	Négatif
...	...	...

On constate donc que les deux variables qui nous intéressent sont des

variables qualitatives (Saison et Rage). Il faut donc transformer nos données en tableau de contingence pour les analyser, soit comme ceci :

	Positif	Négatif
Automne	2	22
Hiver	27	41

Vous voyez qu'une fois transformées en tableau de contingence, les colonnes et les lignes du nouveau tableau représentent les valeurs de nos variables qualitatives originales.

> ⚠ Avertissement
>
> Remarquez bien que dans un tableau de contingence, chaque individu ne doit apparaître qu'une seule fois et aussi que le contenu de chacune des cellules est une fréquence (i.e. un décompte).

Il ne faut jamais, même si c'est tentant, y inscrire des proportions ou des pourcentages. Nous testerons pour des différences de proportions, mais les données doivent être des fréquences.

Pour avoir une meilleure idée de ce qui se passe dans le tableau de contingence, on calcule souvent des **totaux marginaux**, c'est-à-dire que l'on calcule dans la marge le total pour chacune des lignes et colonnes, comme ceci :

	Positif	Négatif	Total
Automne	2	22	24
Hiver	27	41	68
Total	29	63	92

On constate donc deux choses dans ces totaux marginaux. En général, le nombre de ratons-laveurs capturés à l'automne est plus faible que

celui capturés à l'hiver (24 vs. 68). En général, il y a moins de cas positifs que de cas négatifs à la rage (29 vs. 63). Évidemment, comme la nature est variable, le ratio positif/négatif n'est pas exactement le même entre les deux saisons. De la même façon, le ratio automne/hiver n'est pas exactement le même chez les cas négatifs que chez les cas positifs. Ce qui nous intéressera de savoir ici est de trouver si la proportion de cas positifs varie de façon systématique entre les saisons.

Remarquez que même si cela nous intéresse moins dans ce cas-ci, tester cette association entre les deux variables veut aussi dire que l'on teste si la proportion automne/hiver est différente selon que les ratons-laveurs sont positifs ou négatifs à la rage. L'association va toujours dans les deux sens.

19.1. Le test de khi-carré pour un tableau 2x2

Le test classique pour tester cette question d'association entre deux variables qualitatives est le test de khi-carré. Khi étant la lettre grecque, vous le verrez aussi appelé chi-carré (en anglais khi s'écrit chi), χ^2 ou même khi-deux. Il s'agit toujours du même test!

Nous utiliserons pour illustrer le test de khi-carré l'exemple avancé en introduction, soit de savoir si l'efficacité du vaccin contre la rage pour les ratons-laveurs varie entre les saisons.

Étape 1 : Définir les hypothèses

H_0 : Il n'y a pas d'association entre la saison et le succès du vaccin

H_1 : Il y a une association entre la saison et le succès du vaccin

Remarquez que pour ce test, il n'existe pas de façon mathématique officielle d'écrire les hypothèses (ou du moins, pas dans les 3 manuels de

stats que j'ai consultés). On aurait par contre pu dire pour H_0 que les variables étaient indépendantes et pour H_1 qu'elles étaient dépendantes l'une de l'autre.

Étape 2 : Explorer les données

Comme suggéré précédemment, le test de khi-carré assume d'abord deux choses à propos de notre tableau de contingence. Ce dernier doit absolument contenir des fréquences, et chaque individu ne doit être représenté qu'une seule fois dans le tableau. Notez qu'ici le terme individu est utilisé de façon très générale, si votre étude porte sur les écosystèmes, il faudrait bien entendu que chaque écosystème ne soit représenté qu'une seule fois, etc. Il faut être très attentif à la structure du tableau, en particulier si ce n'est pas nous qui l'avons construit. Pour notre exemple sur la rage, il aurait pu arriver que quelqu'un vous envoie les données sous ce format :

	Cas positifs	Individus testés
Automne	2	24
Hiver	27	68

Ce dernier tableau n'est PAS approprié pour le test de khi-carré, même si à première vue, il est très semblable au précédent. Les individus positifs sont présents à la fois dans les deux colonnes!

Avant de décrire la troisième et dernière assomption du test de khi-carré, il importe de définir le concept de fréquence attendue. Rappelez-vous d'abord que l'hypothèse nulle de notre test est que les deux variables ne sont pas associées entre elles. Les **fréquences attendues** sont donc celles que l'on retrouverait dans le tableau de contingence si jamais les deux variables étaient totalement indépendantes les unes des autres.

Comment trouver ces fréquences attendues? Pour chacune des cellules du tableau de contingence, on multiplie les totaux marginaux de chaque

cellule puis on les divise par le grand total. Par exemple, la fréquence attendue des cas positifs en automne serait de (24 x 29) / 92 = 7,57.

Le tableau des fréquences attendues ressemblerait donc à ceci :

	Positifs	Négatifs
Automne	7,56	16,43
Hiver	21,43	46,57

Ces données correspondent aux fréquences (théoriques) que l'on aurait dû observer en absence de bruit si jamais l'hypothèse nulle de notre test était vraie.

On peut maintenant énoncer la troisième assomption du test de khi-carré, soit que 80 % des cellules du tableau des valeurs attendues doivent contenir des valeurs d'au moins 5, et qu'aucune ne doit contenir une valeur < 1. Lorsque, comme ici, notre tableau de contingence est de 2x2, **chacune** des cellules doit être au moins de 5.

Étape 3 : Calculer la statistique de test

Une fois toutes ces choses énoncées, le calcul de la statistique du test de khi-carré est relativement intuitif :

$$\chi^2 = \sum_{i=1}^{n} \frac{(|O_i - E_i| - 0.5)^2}{E_i}$$

Où O_i est la valeur observée dans chaque cellule et E_i est la valeur attendue. Autrement dit, plus les fréquences observées sont différentes des fréquences attendues, plus la valeur du khi-carré sera élevée. La partie -0,5 se nomme la correction de Yates. Elle sert à éliminer le biais possible lié à l'analyse de tableaux 2x2. Pour des tableaux plus grands comme dans la section suivante, cette correction n'est pas nécessaire.

Si l'on effectue ce calcul pour notre exemple, nous arrivons à une valeur de khi-carré de 6,70.

Étape 4 : Obtenir la valeur de p

Contrairement aux tests des chapitres précédents, mais de façon un peu prévisible si vous avez saisi la nomenclature des tests statistiques, la distribution de notre statistique de test si l'hypothèse nulle est vraie suivra une distribution de... khi-carré! La distribution de khi-carré est particulièrement asymétrique, mais devient de plus en plus symétrique à mesure que les degrés de liberté augmentent (nommés k dans la figure) :

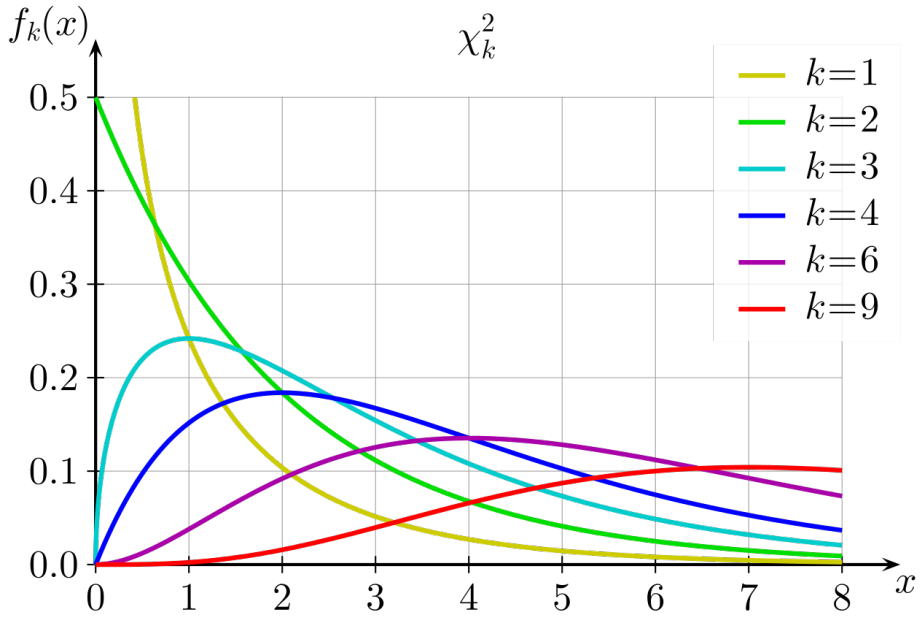


Figure 19.1.: Geek3, CC BY 3.0, via Wikimedia Commons

Les degrés de liberté du test de khi-carré sont définis comme étant (nombre de lignes - 1) x (nombre de colonnes - 1). Pour notre test de 2x2,

nous avons donc un seul degré de liberté.

La probabilité associée à une valeur de khi-carré de 6,70 avec un seul degré de liberté est de 0,009639.

Étape 5 : Rejeter ou non l'hypothèse nulle.

Même si nous avons été chercher notre valeur de p dans une nouvelle distribution, c'est quand même la même poutine que les autres tests pour la suite.

La valeur de p est plus petite que le seuil de 0,05. Il serait donc très rare d'avoir une valeur de khi-carré aussi élevée si jamais nos deux variables n'étaient pas associées. On peut donc dire que l'on rejette l'hypothèse nulle qui stipulait que nos variables n'étaient pas associées.

On pourrait écrire notre résultat comme ceci : «Le succès du vaccin contre la rage chez les ratons-laveurs était significativement relié à la saison où se tenait l'opération de vaccination ($\chi 21 = 6,70$, $p = 0,0096$)».

19.2. Le test de khi-carré pour un tableau autre que 2x2

Le test de khi-carré s'effectue toujours pour évaluer l'indépendance entre **deux** variables qualitatives. Il peut cependant arriver qu'une (ou les deux) variables possèdent plus de deux valeurs possibles. On aurait pu, par exemple, analyser plutôt les données suivantes :

	Positifs	Négatifs	Non-concluant
Printemps	2	22	1
Été	27	41	12
Automne	20	21	3
Hiver	5	15	1

L'ensemble de la procédure demeure exactement le même, sauf que l'on enlève la correction de Yates au moment de faire le calcul du khi-carré, comme ceci :

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

Remarquez qu'au final, le test ne peut pas nous dire dans quelle(s) cellule(s) survient la différence. On ne peut pas savoir si la différence provient des tests négatifs en automne, etc. On ne peut répondre qu'à la question : est-ce que les deux variables sont associées.

19.3. Labo : le test de khi-carré dans R

Jusqu'à présent, toutes les données que nous avons travaillées dans R se retrouvaient dans ce qu'on a appelé des tableaux de données (des objets `data.frame` ou `tibble`). Pour certaines fonctions, nous avions besoin d'accéder directement à une colonne à l'aide du `$`, mais d'une façon ou d'une autre, nos données étaient sous forme de tableaux.

Le test du khi-carré lui s'attend à recevoir un autre format, soit une **matrice de données** (`matrix`). Les matrices de données dans R sont aussi formées de données rectangulaires, mais elles ne peuvent contenir qu'un seul type de données à la fois. Elles doivent contenir soit des chiffres, soit du texte, mais il ne peut pas y avoir de mélange des deux dans une même matrice.

La saisie manuelle de matrice dans R s'effectue avec la fonction `matrix`. Il existe des dizaines de façons différentes d'utiliser cette fonction. Je vous en montrerai une, qui devrait convenir pour tout ce que nous aurons à faire dans ce livre (vous pouvez explorer les autres avec la commande `?matrix` qui vous fournit l'aide de cette fonction). Par exemple

pour entrer l'exemple ci-haut dans R, nous aurions pu faire comme ceci :

```
ratons <- matrix(data = c(2,22,27,41), ncol = 2, byrow =
 ↪  TRUE)
```

On passe donc à la fonction `matrix` 3 arguments. Le premier est la série de nombres à utiliser pour remplir la matrice, le deuxième est le nombre de colonnes (le nombre de lignes sera calculé automatiquement) et le troisième indique à R que l'on fournit nos données ligne par ligne. Si on regarde le contenu de notre matrice raton, on devrait voir ceci :

```
ratons
```

```
     [,1] [,2]
[1,]    2   22
[2,]   27   41
```

Cette méthode fonctionne bien si nous avons déjà calculé notre tableau de contingence. Si ce n'est pas le cas, nous verrons dans l'exemple plus bas comment calculer directement le tableau de contingence à partir de nos données brutes.

Sachant que le ratio mâle-femelle est extrêmement variable chez le manchot royal[1], on peut se demander si ce dernier varie aussi d'une année à l'autre chez les manchots de l'archipel Palmer.

Étape 1 :

H_0 : Il n'y a pas d'association entre l'année et le sexe

H_1 : Il y a une association entre l'année et le sexe

Étape 2 :

[1] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0114052

À cette étape, il faut d'abord préparer notre matrice de données, puisque le tableau de données n'est pas un tableau de contingence. Plutôt que de préparer la matrice manuellement, on peut utiliser la fonction `table` pour la construire. Cette dernière s'attend à recevoir les deux colonnes qualitatives qui formeront notre tableau de contingence. Cependant, on doit d'abord nettoyer nos données, puisque plusieurs individus n'ont pu être sexés lors de l'expérience.

```
library(palmerpenguins)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
pour_chi_carre <- penguins |>
  drop_na(sex)

ma_matrice <- table(pour_chi_carre$sex,
 ↪  pour_chi_carre$year)

ma_matrice
```

```
        2007 2008 2009
```

```
female   51   56   58
male     52   57   59
```

À première vue, la proportion mâle-femelle semble être très constante entre les années, très proche du 1:1, avec chaque fois légèrement plus de mâles.

Comme nous savons que chaque individu n'est représenté qu'une seule fois dans notre tableau de données, on sait que les deux premières assomptions du test sont respectées. La troisième assomption, pour les valeurs attendues >5 sera validée une fois le calcul complété, puisque R nous fournira aussi les valeurs attendues dans notre objet de résultats.

Étapes 3 et 4 :

Le test de khi-carré dans R s'effectue avec la fonction `chisq.test` (*CHI-SQuared test*). On lui passe notre matrice de données et c'est tout! R s'occupe tout seul de calculer les valeurs attendues, la statistique de test et décider d'appliquer ou non la correction de Yates.

```
resultat <- chisq.test(ma_matrice)
```

On conserve le résultat du calcul dans un objet, car on a plusieurs opérations à faire avec.

Tout d'abord, il faut aller consulter le tableau des valeurs attendues pour s'assurer que l'on respecte les conditions d'application du test.

```
resultat$expected
```

```
            2007     2008     2009
female 51.03604 55.99099 57.97297
male   51.96396 57.00901 59.02703
```

Ici, toutes les valeurs sont bien au-delà de 5, donc aucun stress.

On peut maintenant regarder les résultats du test comme tel :

```
resultat
```

```
    Pearson's Chi-squared test

data:  ma_matrice
X-squared = 7.8283e-05, df = 2, p-value = 1
```

Cette sortie contient très peu d'informations comparé à la régression linéaire. Nous avons la valeur de khi-carré calculée (7,8 x 10^{-5}), les degrés de liberté (2) et la valeur de p associée (1).

Étape 5 :

Comme cet événement est relativement commun lorsque notre hypothèse nulle est vraie (p > 0,05), on considère qu'un tel résultat n'est pas significatif. On ne peut PAS rejeter l'hypothèse nulle qu'il n'y a pas d'association entre nos variables.

Étape 6 :

Il n'y a pas de taille d'effet comme tel à rapporter pour le test de khi-carré. On peut cependant écrire notre résultat comme ceci : « Il n'y a pas d'association significative entre l'année et le sexe (χ_2=7,8 x 10^{-5}, p = 1) »

19.4. Labo : le test exact de Fisher

Le test de khi-carré n'est pas le seul test conçu pour analyser des tableaux de contingence. Les mathématiciens en ont aussi conçu d'autres, entre autres, le test exact de Fisher.

Ce dernier est beaucoup plus complexe à calculer. Le principe de base est néanmoins plutôt simple et je vous l'expliquerai ici. Le test de Fisher explore tous les tableaux de contingence qui auraient pu exister en conservant les totaux marginaux de notre tableau original. Par exemple, pour un tableau de contingence de 2x2 avec les observations suivantes :

3	1
1	3

On calcule les totaux marginaux :

3	1	**4**
1	3	**4**
4	**4**	**8**

Et ensuite, R cherchera tous les tableaux permettant de respecter ces totaux, p. ex.

2	2		0	4	
2	2		4	4	

4	0
0	4

Etc.

Une fois tous ces tableaux trouvés, R calcule sur chacun une statistique et détermine ensuite combien rare est notre tableau de valeurs observées, comparé à tous ces tableaux possibles.

Cette procédure permet de toujours obtenir une valeur de p exacte, même quand les valeurs attendues sont < 5 ou < 1. C'est pourquoi certains auteurs recommandent de toujours utiliser ce test, car il serait bon en toutes circonstances. Cependant, avec de grands nombres dans

le tableau, il devient extrêmement complexe à calculer, voir impossible pour un ordinateur ordinaire.

Le test de khi-carré lui a donc longtemps été préféré, parce qu'il est raisonnable à calculer manuellement, ce qui était très important il n'y a même pas 20-30 ans. Le test exact de Fisher serait aussi un peu moins puissant (plus conservateur) et pourrait donc, dans certaines circonstances, rater des liens significatifs que le test de khi-carré aurait trouvés. J'accepterai pour le cours son utilisation, comme bon vous semble en remplacement du test de khi-carré.

Dans R, le test exact de Fisher s'utilise exactement de la même façon que le khi-carré, soit en lui passant une matrice de données :

```
fisher.test(ma_matrice)
```

```
    Fisher's Exact Test for Count Data

data:  ma_matrice
p-value = 1
alternative hypothesis: two.sided
```

On obtient donc avec ce test aussi une valeur de p de 1. Le test de Fisher ne trouve donc pas non plus d'association significative entre nos deux variables.

20. L'analyse de covariance

20.1. Introduction

Les analyses de covariance sont les modèles statistiques les plus complexes que nous étudierons dans la section portant sur les tests statistiques. Contrairement à tout ce que nous avons vu auparavant, ils permettent d'analyser simultanément plusieurs variables explicatives dans un même modèle (mais toujours une seule variable expliquée). Dans sa définition la plus stricte (celle que nous verrons dans le cours), une analyse de covariance (ANCOVA) est un modèle d'ANOVA, mais auquel on ajoute aussi une variable explicative continue. Cette variable explicative, aussi nommée **covariable**, permet de corriger nos mesures pour, par exemple, enlever un effet confondant.

Ce genre de situation peut, par exemple, survenir si l'on veut déterminer si l'ajout de chaux augmente la production de graines chez une espèce d'arbre. Bien que ce qui nous intéresse serait l'effet de l'ajout de chaux, le facteur principal qui contrôle le nombre de graines produites par l'arbre sera sans aucun doute sa taille. Plus l'arbre est grand, plus il produira de graines. Il y aura donc beaucoup de "bruit" dans nos mesures de production de graines. Une grande partie de la variabilité du nombre de graines que nous observerons ne proviendra pas de la variable qui nous intéresse. De là l'utilité de mesurer une covariable (ici la hauteur de l'arbre), qui permettra de tenir compte de ce bruit et de le mettre de côté pour bien évaluer l'effet de la chaux.

20.2. Le modèle statistique

Le modèle classique de l'ANCOVA peut s'écrire de différentes façons, qui reviendront d'une façon ou d'une autre à peu près à ceci :

$$y = b_0 + b_1 x_1 + b_2 x_2$$

Autrement dit, chaque observation (y) est prédite par l'ordonnée à l'origine (b_0) à laquelle on additionne l'effet de la covariable (b_1) et l'effet de la variable catégorique (b_2). L'équation est donc identique à la régression linéaire, mais à laquelle on ajoute un terme b_2 pour l'effet de variable catégorique. L'interprétation des chiffres associés à b_1 et b_2 est relativement directe, soit respectivement le changement de y pour un changement de x_1 et le changement en y pour l'effet de b_2. Par contre, la valeur associée à b_0 est plutôt abstraite et de peu d'intérêt biologique. Il s'agit de la valeur prédite par notre modèle lorsque la covariable vaut zéro et que l'on a pas l'effet de la variable qualitative.

Dans cette notation, la variable x_2 est formée de zéros et de uns, selon que notre variable catégorique présente, pour chaque ligne, la valeur du groupe de référence ou non. Cette transformation se nomme "*dummy coding*" et sera effectuée pour vous par R. Les valeurs transformées en zéro deviendront le niveau de référence, et les autres seront mesurées comme des différences par rapport à ce niveau. Nous y reviendrons plus en détails dans le deuxième cours de stats, les technicalités associées ne sont pas si importantes ici pour le moment.

Ce qu'il est important de comprendre pour le moment, c'est que notre modèle contiendra maintenant 3 paramètres, soit l'ordonnée à l'origine (b_0), la pente associée à la covariable (b_1) et l'effet de la variable catégorique par rapport à son niveau de référence (b_2).

Comme pour la régression linéaire, l'estimation des paramètres de ce modèle s'effectuera avec la méthode des moindres carrés, mais avec un

calcul plus complexe que ce que nous calculerons à la main dans ce premier cours. Nous laisserons R faire les calculs pour nous.

20.3. Les assomptions de l'ANCOVA

La première chose à savoir concernant le modèle d'ANCOVA, est que toutes les assomptions de la régression linéaire s'appliquent aussi à l'ANCOVA.

Avant la modélisation, nous devons donc réfléchir à l'indépendance de nos observations. Après la modélisation, il faudra valider la normalité des erreurs et l'homogénéité des résidus et vérifier si une ou des observations pourraient avoir une forte influence sur notre modèle. L'ANCOVA ajoute une assomption supplémentaire à tout cela, qui est que les pentes soient homogènes entre les deux groupes. Autrement dit, l'ANCOVA, au sens strict, est conçue pour tester des différences d'ordonnée à l'origine, mais pas des différences de pentes.

20.4. Labo : l'ANCOVA

Pour illustrer le calcul de l'ANCOVA, nous passerons directement à un exemple en R, puisque vous n'effectuerez probablement jamais manuellement les calculs associés à l'estimation des paramètres.

Nous analyserons donc un jeu de données où on évaluera comment l'application de chaux influence la production de graines d'une espèce d'arbres, mais en tenant compte dans notre analyse de l'effet de la taille de l'arbre sur sa production de graines. Nous avons mesuré pour notre expérience 28 arbres, 13 au pied desquels on a ajouté de la chaux, et 15 sans traitement de chaux. Nous avons mesuré pour chaque arbre la

production de graines et le diamètre de l'arbre (en cm). Le fichier CSV associé à ces données est disponible avec les notes de cours[1].

Étape 1 : Définir les hypothèses

Bien que notre modèle statistique soit plus complexe que les précédents, notre question d'intérêt et les hypothèses associées sont somme toutes assez simples : est-ce que l'application de chaux influence la production de graines de nos arbres. Autrement dit, est-ce que la moyenne de production de graines avec et sans chaux est la même, une fois que nous tenons en compte les différences de taille des arbres

$$H_0 : \mu_{\text{contrôle (corrigé)}} = \mu_{\text{chaux (corrigé)}}$$
$$H_1 : \mu_{\text{contrôle (corrigé)}} \neq \mu_{\text{chaux (corrigé)}}$$

Étape 2 : Explorer visuellement les données

Comme pour tous les autres outils statistiques, il vaut la peine ici de commencer par regarder à quoi ressemblent nos données avant de se lancer. Nous en profiterons aussi pour charger les librairies nécessaires à notre travail. Nous en profitons aussi pour convertir la variable Traitement en variable catégorique (*factor*).

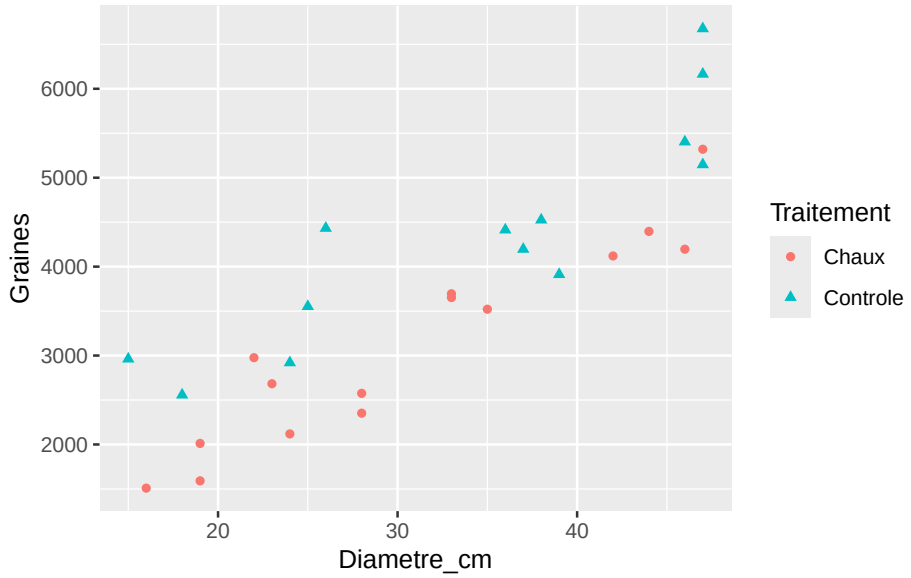
```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------ tidyverse_conflicts() --
```

[1] https://drive.google.com/file/d/1O880ZTYxx31Emy1B40vxIPI6euUDZa11/view?usp=sharing

```
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
arbres <- read.csv("donnees/arbres.csv") |>
  mutate(Traitement = as.factor(Traitement))
  arbres |>
    ggplot(aes(
      x = Diametre_cm,
      y = Graines,
      color = Traitement,
      shape = Traitement
    )) +
    geom_point()
```



Dans ce graphique, on constate qu'en général, nos variables Diamètre et Graines sont reliées de façon linéaire. La variance semble homogène sur l'ensemble du gradient et les pentes, à l'œil, semblent relativement homogènes entre les deux groupes de traitement. On a donc tout ce qu'il nous faut pour lancer l'analyse.

En regardant le graphique, on devrait s'attendre à trouver une différence entre nos traitements, c'est-à-dire que l'ordonnée à l'origine de nos pentes semble clairement différente, le traitement de chaux semblant diminuer la production de graines comparé au contrôle.

Étapes 3 et 4 : Calculer la statistique de test et obtenir la valeur de p

Comme pour la régression linéaire, la façon la plus directe d'estimer les paramètres de l'équation d'ANCOVA (b_0, b_1 et b_2) est par la méthode des moindres carrés. La fonction pour calculer l'ANCOVA dans R est la même que celle pour la régression, soit la fonction lm.

Par contre, nous devons faire une petite modification à notre base de données avant de commencer. Nous avons discuté du fait que le paramètre b0 représente le niveau de référence et b2 l'effet de la variable catégorique par rapport à ce niveau de référence. R ne peut pas deviner tout seul laquelle des valeurs de notre variable qualitative doit être le niveau de référence. Voyons d'abord ce que R avait choisi pour nous :

```
levels(arbres$Traitement)
```

```
[1] "Chaux"    "Controle"
```

Comme "Chaux" est le premier item retourné dans la liste, R croit que nous voulons celui-là comme valeur de référence.

Comme ici il sera plus logique que "Controle" soit notre niveau de référence, nous allons modifier notre variable catégorique pour refléter cela :

```
arbres <- arbres |>
  mutate(Traitement = relevel(Traitement, "Controle"))
```

La fonction relevel permet de changer le niveau de référence d'une variable catégorique. Remarquez que pour que cela modifie vraiment variable, nous devons écraser l'ancien tableau de données avec le nouveau, avec l'opérateur d'assignation. On peut maintenant vérifier que le niveau de référence est le bon :

```
levels(arbres$Traitement)
```

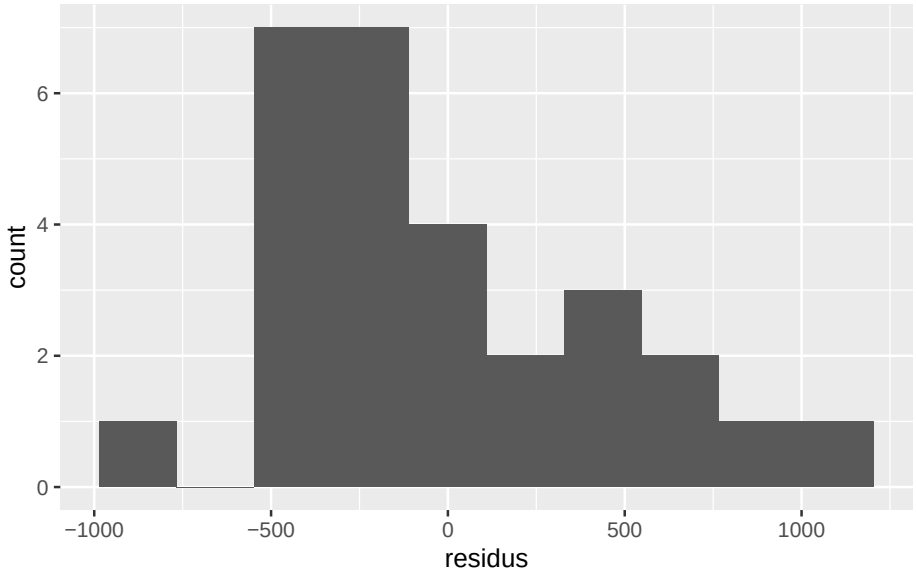
```
[1] "Controle" "Chaux"
```

Nous pouvons maintenant ajuster notre modèle d'ANCOVA comme ceci :

```
m <- lm(Graines~Diametre_cm+Traitement, data = arbres)
```

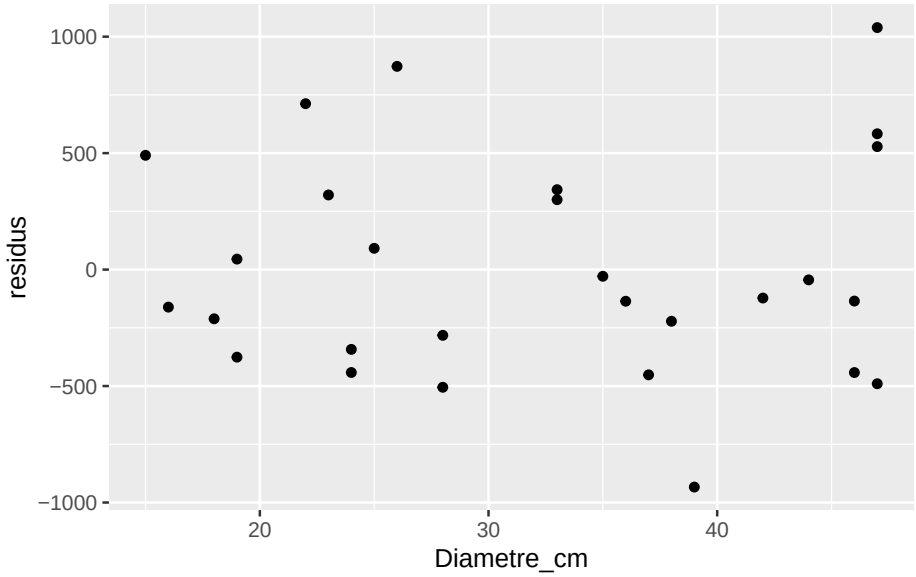
Avant de se lancer dans l'interprétation des sorties de ce modèle, nous devons par contre le valider. Comme pour la régression, nous regarderons d'abord la normalité et l'homogénéité des résidus.

```
arbres <-
  arbres |>
  mutate(
    residus = resid(m),
    D = cooks.distance(m)
  )
arbres |>
  ggplot(aes(x = residus)) +
  geom_histogram(bins = 10)
```



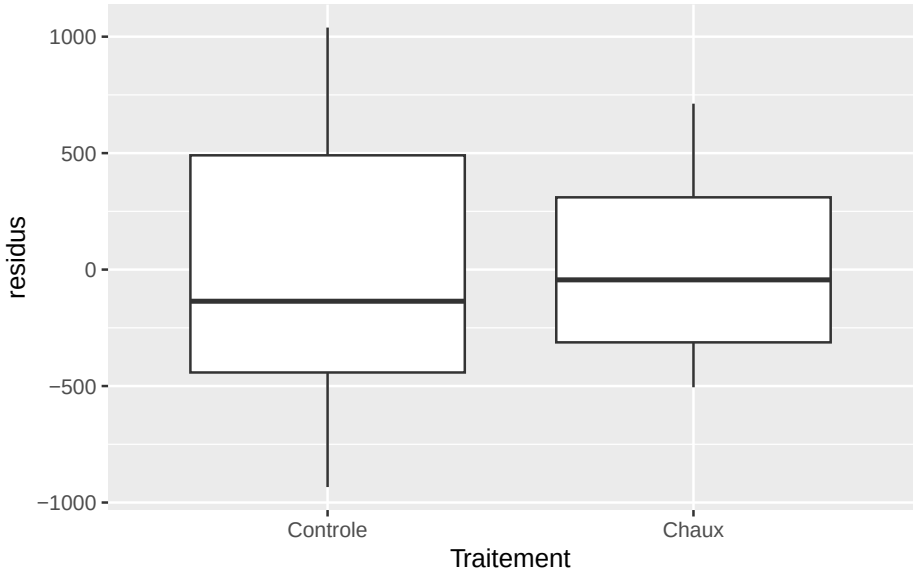
Les résidus sont répartis de façon relativement normale.

```
arbres |>
  ggplot(aes(x = Diametre_cm, y = residus)) +
  geom_point()
```



Et ils sont relativement homogènes à travers le gradient de diamètres. Mais maintenant que nous avons deux variables explicatives dans notre modèle, il faut valider l'homogénéité des résidus pour cette deuxième variable aussi. Comme il s'agit d'une variable qualitative, nous regarderons la répartition des résidus à l'aide d'un diagramme à moustache.

```
arbres |>
  ggplot(aes(x = Traitement, y = residus)) +
  geom_boxplot()
```



Les deux boîtes sont plutôt semblables, celle du traitement "Chaux" est légèrement plus étroite, mais pas de quoi s'alarmer compte tenu de la faible taille d'échantillon.

Il ne faut pas oublier de vérifier si certaines observations auraient plus influencer notre modèle de façon trop importante :

```
arbres |>
  filter(D > 1)
```

```
[1] Diametre_cm Traitement  Graines     residus
[5] D
<0 rows> (or 0-length row.names)
```

Il nous reste maintenant à valider l'assomption spécifique à l'ANOVA, soit l'homogénéité des pentes. La façon classique de tester cette assomption

est d'ajouter un terme au modèle, qui servira à quantifier la différence de pente entre le groupe de référence (Controle) et l'autre groupe (Chaux).

On ajustera un nouveau modèle avec ce paramètre supplémentaire, et si il s'avère significatif, la pente ne sera pas considérée homogène entre les deux groupes et l'ANCOVA ne pourra pas être interprétée. Le modèle pour tester l'homogénéité des pentes sera donc le suivant :

$$y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_1 x_2$$

Remarquez que ce nouveau paramètre (b_3) est associé à la multiplication des variables x_1 et x_2. On nomme ce genre de paramètre une interaction, et on l'ajoute dans une formule R avec le ":", comme ceci :

```
m_homogeneite <- lm(Graines~
  Diametre_cm+
  Traitement+
  Traitement:Diametre_cm,
  data = arbres)

summary(m_homogeneite)
```

```
Call:
lm(formula = Graines ~ Diametre_cm + Traitement +
Traitement:Diametre_cm,
    data = arbres)

Residuals:
   Min     1Q Median     3Q    Max
-921.1 -329.5 -112.8  337.7 1072.6

Coefficients:
```

```
                             Estimate Std. Error
(Intercept)                  1077.794    453.401
Diametre_cm                    96.289     12.613
TraitementChaux             -1073.732    613.127
Diametre_cm:TraitementChaux     5.358     17.980
                            t value Pr(>|t|)
(Intercept)                   2.377   0.0258 *
Diametre_cm                   7.634 7.15e-08 ***
TraitementChaux              -1.751   0.0927 .
Diametre_cm:TraitementChaux   0.298   0.7683
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 499.2 on 24 degrees of freedom
Multiple R-squared:  0.8734,    Adjusted R-squared:
0.8575
F-statistic: 55.18 on 3 and 24 DF,  p-value: 6.433e-11
```

Pour le moment, à l'étape de validation des assomptions, la seule partie qui nous intéresse est celle-ci :

```
Diametre_cm:TraitementChaux     5.358     17.980   0.298
0.7683
```

soit le paramètre d'interaction (b_3) discuté ci-haut. Comme ce dernier n'est pas significativement différent de zéro (p=0.768), on peut considérer nos pentes comme homogènes et interpréter notre modèle original.

On peut donc maintenant (enfin vous direz!) regarder les sorties de notre modèle d'ANCOVA :

```
summary(m)
```

```
Call:
lm(formula = Graines ~ Diametre_cm + Traitement, data =
arbres)

Residuals:
   Min     1Q Median     3Q    Max
-933.6 -350.9 -128.7  326.1 1039.0

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      987.537    331.199   2.982  0.00631
Diametre_cm       98.926      8.823  11.212 3.04e-11
TraitementChaux -900.202    188.417  -4.778 6.63e-05

(Intercept)     **
Diametre_cm     ***
TraitementChaux ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 490 on 25 degrees of freedom
Multiple R-squared:  0.8729,    Adjusted R-squared:
0.8627
F-statistic: 85.85 on 2 and 25 DF,  p-value: 6.331e-12
```

Ces sorties s'interprètent exactement comme celles de la régression linéaire. À la différence que maintenant, notre modèle contient 3 paramètres (i.e. 3 lignes à la section coefficients) plutôt que 2.

Ce sont dans l'ordre nos paramètres b_0 (Intercept), b_1 (Diametre_cm) et b_2 (TraitementChaux), soit respectivement l'ordonnée à l'origine, l'effet de la covariable et l'effet de notre variable qualitative. Comme discuté

ci-haut, le paramètre b0 a peu d'interprétation biologique. Le paramètre b1 nous informe que chaque cm de diamètre de l'arbre augmente en moyenne de 98,9 graines la production de l'arbre. Enfin, le traitement de chaux diminue de 900,2 le nombre de graines produites par rapport aux arbres sans ce traitement.

Étape 5 : Rejeter ou non l'hypothèse nulle

Comme notre hypothèse portait sur paramètre b_2, nous pouvons observer que sa valeur de t est de -4,778 et sa valeur de p est de 0,0000663. Comme cette valeur est plus petite que le seuil de signification de 0,05, on considère que l'effet du traitement de chaux est significativement différent de zéro.

On pourrait aussi, si on le désire, interpréter la valeur de p de notre paramètre Diametre_cm (b_1), qui est aussi significativement différent de zéro.

Enfin, on voit que le r^2 associé à notre modèle est de 0,87, ce qui est très élevé. Par contre, on ne sait pas si cette explication provient de l'effet de taille, du traitement de chaux ou d'une combinaison des deux. Tout ce que l'on sait, c'est que les deux ensemble expliquent 87 % de la variabilité du nombre de graines.

Étape 6 : Citer la taille de l'effet et son intervalle de confiance

Pour obtenir l'intervalle de confiance de nos paramètres, on utilise, comme pour la régression, la fonction confint, à laquelle on passe notre objet de résultat, comme ceci :

```
confint(m)
```

```
                    2.5 %    97.5 %
(Intercept)       305.42092 1669.6539
Diametre_cm        80.75366  117.0981
TraitementChaux -1288.25473 -512.1502
```

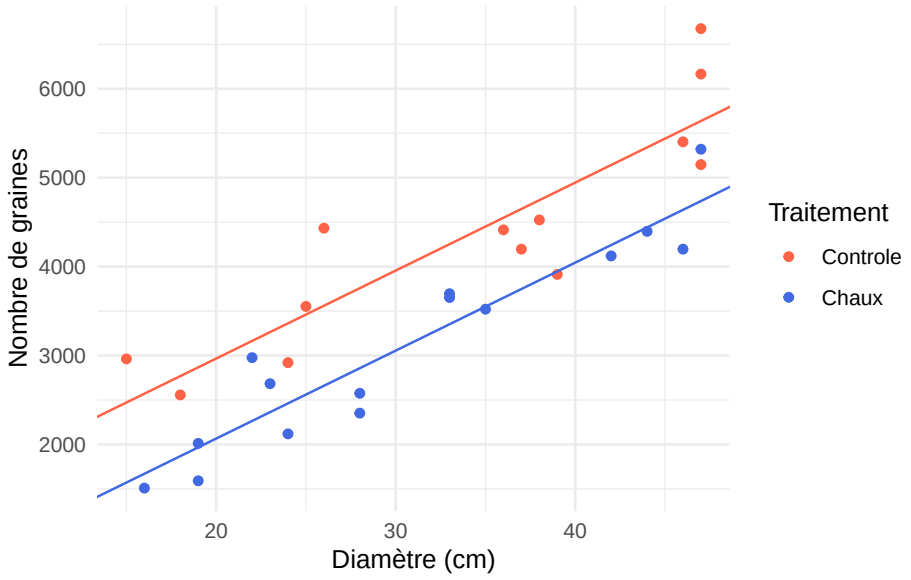
L'intervalle de confiance de notre paramètre d'intérêt (l'effet du traitement de chaux) va donc de -1288,25 à -512,15 graines.

On pourrait rapporter notre résultat comme ceci dans un rapport : «Nous avons effectué une analyse d'ANCOVA afin de d'évaluer l'effet du traitement de chaux sur la production de graines, tout en contrôlant pour le diamètre de l'arbre. Nous avons trouvé une diminution significative du nombre de graines, allant de 512,15 à 1288,25 graines de moins que le contrôle (IC 95%), ce qui était significativement différent de zéro (T_{25}=-4,78, p=0,0000663)».

Pour présenter ce résultat de façon visuelle dans un rapport, on peut utiliser la même technique que pour la régression linéaire, soit d'utiliser la couche geom_abline pour tracer la droite de régression. Par contre, comme ici nous avons deux pentes avec deux ordonnées à l'origine différentes, nous devons utiliser la couche deux fois, comme ceci :

```
arbres |>
  ggplot(aes(
    x = Diametre_cm,
    y = Graines,
    color = Traitement)
  ) +
  geom_point() +
  geom_abline(
    slope = 98.926,
    intercept = 987.537,
    color = "tomato"
  ) +
  geom_abline(
    slope = 98.926,
    intercept = 987.537-900.202,
    color = "royalblue"
  ) +
```

```
  scale_color_manual(values = c(
    "tomato", "royalblue"
  )) +
  theme_minimal() +
  labs(
    x = "Diamètre (cm)",
    y = "Nombre de graines"
  )
```



Notez que l'ordonnée à l'origine de la pente du traitement de chaux n'est pas -900. Elle à plutôt à 900 sous l'ordonnée à l'origine du niveau de référence. Il faut donc la placer à 987-900.

Vous remarquerez que j'ai fait bien attention ici de toujours utiliser le paramètre b_1 pour la covariable et le paramètre b_2 pour l'effet de la variable qualitative. Par contre, cet ordre est arbitraire. Vous auriez pu tout

aussi bien mettre b_1 pour la variable qualitative et b_2 pour la quantitative. Soyez attentif à cette subtilité si jamais vous lisez sur le sujet ailleurs que dans mes notes de cours.

20.5. Exercice : l'ANCOVA

Pour vous entraîner à appliquer l'ANCOVA, je vous suggère avec le tableau de données sur les manchots de Palmer de répondre à la mise en situation suivante :

Nous avons vu dans les chapitres précédents que la longueur des ailes de manchots influençait leur poids. Nous aimerions maintenant savoir si il existe un dimorphisme sexuel chez les manchots, par exemple à savoir si les mâles et les femelles ont un poids différent. Nous voudrions donc corriger notre test entre les mâles et les femelles pour s'assurer que les différences de longueurs d'ailes sont correctement prises en compte dans notre comparaison du poids des manchots.

21. Les tests non-paramétriques

21.1. Principe général

Nous avons vu dans les chapitres précédents une série de tests statistiques. Vous avez sans doute remarqué qu'une assomption importante revenait sans cesse : que les échantillons proviennent d'une distribution normale. Nous avons vu que pour la plupart des tests, cette assomption pouvait être étirée un peu. Nos données n'avaient jamais besoin d'être parfaitement normales. Mais que peut-on faire lorsqu'elles ne le sont clairement pas? Il existe deux stratégies possibles : soit utiliser des **transformations** pour normaliser nos données (Chapitre 9), soit d'utiliser des tests non-paramétriques. Nous verrons dans ce chapitre la deuxième stratégie, soit d'utiliser des tests différents, qui n'assument pas la normalité.

Notez bien que tous les tests non-paramétriques assument tout de même l'indépendance des observations. Ce ne sont pas les bons outils à utiliser si jamais c'est cette assomption en particulier que vos données ne respectent pas.

21.2. Perte de puissance

Vous vous demandez peut-être à ce point pourquoi on se casserait la tête avec des tests paramétriques quand les non-paramétriques sont dispo-

nibles? La raison est que les tests non-paramétriques sont, en général, moins puissants que leur équivalent paramétrique.

On peut dire de façon générale, que les tests non-paramétriques présentent 95 % de l'efficacité des tests paramétriques. Ce qui signifie qu'un test non-paramétrique avec un n de 100 serait aussi puissant (aurait tant de chances de détecter une différence significative) qu'un test paramétrique équivalent avec un n de 95. Autrement dit, faire un test non-paramétrique équivaut à gaspiller 5 % de vos échantillons.

La corrélation de Spearman a 91 % de la puissance de celle de Pearson. À l'extrême, le test de signe (*signed rank test*), que l'on ne verra pas ici, a 61 % seulement de la puissance de son équivalent paramétrique.

> Astuce
>
> Donc, si vous voulez valoriser vos données au maximum, utilisez les tests paramétriques chaque fois que c'est possible!

21.3. Test de Wilcoxon (remplacement du test de T)

Le premier test non-paramétrique que nous verrons est celui de Wilcoxon (*Wilcoxon Rank-Sum Test*). Il s'utilise en remplacement du test de T pour deux échantillons. L'application du test est d'ailleurs très semblable, à quelques changements près, que nous verrons ci-dessous.

Le premier changement est que, plutôt que de se baser sur les moyennes, le test compare la **médiane** des deux échantillons. L'hypothèse nulle étant que les médianes sont égales, et l'hypothèse alternative étant que les médianes sont différentes.

L'exploration visuelle des données est la même que le test de T, soit à l'aide d'un diagramme à moustaches. Remarquez qu'ici les diagrammes à moustaches sont encore plus appropriés que pour les tests de T,

puisqu'on y voit directement la médiane. Nous n'avons pas besoin de regarder l'histogramme de la distribution des fréquences, puisqu'aucune assomption n'y est reliée.

La statistique de test, elle, est complètement différente du test de T. Elle est basée sur le rang des observations plutôt que sur leur valeur comme tel. Il s'agit d'une stratégie commune des tests non-paramétriques, que l'on reverra dans le test de Kruskal-Wallis et dans la corrélation de Spearman.

Pour illustrer ce principe, imaginons que nous avons fait une mini-expérience, où nous avons mesuré le nombre d'alevins dans 2 aquariums où nous avons ajouté un mécanisme d'oxygénation et 3 aquariums où nous n'avions pas ce mécanisme. Nos données sont les suivantes :

Avec oxygénation : 43 et 32 alevins

Sans oxygénation : 41, 35 et 44 alevins

Si on trie nos observations, leurs rangs seront respectivement :

Avec oxygénation : 4, 1

Sans oxygénation: 3, 2, 5

Le calcul se base ensuite sur la somme des rangs de chacun des échantillons et la taille de l'échantillon pour calculer une statistique nommée W (plutôt que T ou F par exemple). Si jamais vous grattez un peu dans ces calculs, vous verrez qu'en fait, il existe deux calculs différents selon la taille de l'échantillon. R fera toujours pour vous la bonne version des calculs, vous n'aurez pas à vous inquiéter de ce détail.

Il faut ensuite, comme pour les autres tests, aller chercher la valeur de p associée à cette valeur de W et aux degrés de liberté et déterminer si une telle différence est rare ou non quand l'hypothèse nulle est vraie.

21.4. Labo : Le test de Wilcoxon

Nous reprenons pour ce test la même question que pour le test de T à deux échantillons, à savoir : est-ce que la longueur des ailes des machots Adélie varie entre l'île Torgersen et l'île Biscoe.

Étape 0 :

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(palmerpenguins)

adelie <-
  penguins |>
  filter(species == "Adelie") |>
  filter(island %in% c("Torgersen", "Biscoe")) |>
  drop_na(flipper_length_mm)

torgersen <- adelie |> filter(island == "Torgersen")
biscoe <- adelie |> filter(island == "Biscoe")
```

Étape 1 :

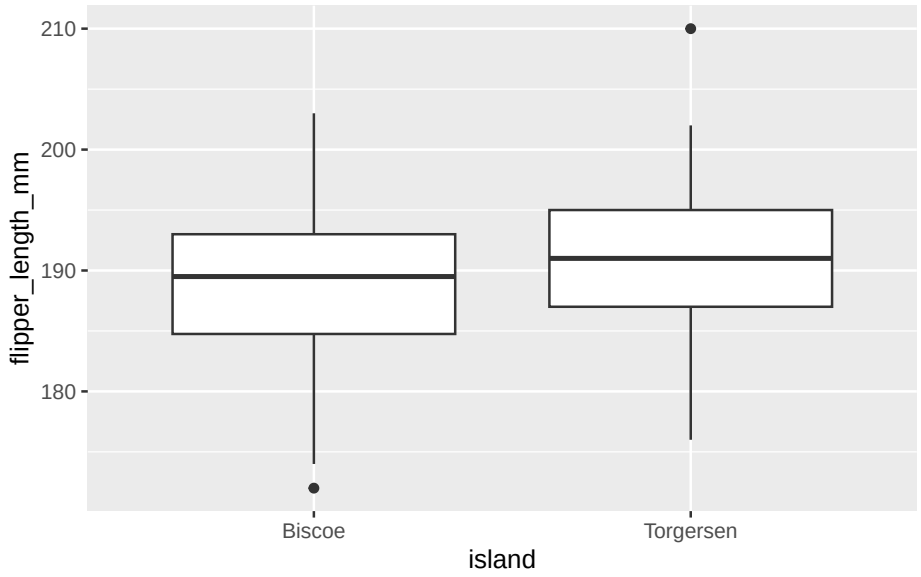
H_0 : La médiane de la longueur des ailes de l'île Torgersen est égale à celle de l'île Biscoe.

H_1 : La médiane de la longueur des ailes de l'île Torgersen est différente de celle de l'île Biscoe.

Étape 2 :

Bien que nous n'ayons pas d'assomption de normalité à vérifier, il est tout de même important de visualiser nos données, pour nous permettre de valider le résultat du test et de s'assurer que nos données sont chargées correctement, etc.

```
adelie |>
  ggplot(aes(x = island, y = flipper_length_mm)) +
  geom_boxplot()
```



À l'oeil, les manchots de l'île Biscoe ont des ailes légèrement plus courtes que sur l'île Torgersen, mais peut-être pas sufisamment différentes pour qu'on soit certains que cette différence n'est pas zéro.

Étapes 3 et 4 :

La fonction pour effectuer le test de Wilcoxon se nomme wilcox.test, et s'utilise comme celle du test de T :

```
wilcox.test(torgersen$flipper_length_mm,
↪  biscoe$flipper_length_mm)
```

```
    Wilcoxon rank sum test with continuity
    correction

data:  torgersen$flipper_length_mm and
biscoe$flipper_length_mm
W = 1341, p-value = 0.1023
alternative hypothesis: true location shift is not
equal to 0
```

Étape 5 :

Comme notre valeur de p est > 0,05, on ne peut pas rejeter l'hypothèse nulle d'aucune différence de médiane entre nos deux groupes.

Remarquez que (comme prévu), cette valeur de p est plus grande (moins proche du seuil de signification; 0,1023) que celle du test de T à deux échantillons (0.07444) pour les mêmes données, puisque le test de Wilcoxon a moins de puissance.

Étape 6 :

Le test de Wilcoxon par défaut dans R ne nous fournit pas d'intervalle de confiance associée à nos résultats. Nous devons activer manuellement cette option :

```
wilcox.test(torgersen$flipper_length_mm,
 ↪  biscoe$flipper_length_mm, conf.int = TRUE)
```

```
    Wilcoxon rank sum test with continuity
    correction

data:  torgersen$flipper_length_mm and
biscoe$flipper_length_mm
W = 1341, p-value = 0.1023
alternative hypothesis: true location shift is not
equal to 0
95 percent confidence interval:
 -1.973526e-05  4.999983e+00
sample estimates:
difference in location
              2.000049
```

Comme expliqué dans l'aide de la fonction wilcox.test, cet intervalle de confiance est un peu contre-intuitif. Alors que le test de Wilcoxon teste pour une différence de médiane, l'intervalle de confiance ne nous informe pas sur la différence entre les médianes, mais plutôt sur la médiane des différences. Une fois cela dit, ça ne change pas grand chose sur notre interprétation dans la vraie vie, mais c'est quand même bon à savoir.

Nous pourrions donc écrire nos résultats comme ceci : « Il n'y a pas de différence significative de longueur d'aile entre les îles de Torgersen et Biscoe (W = 1341, p = 0,102). L'intervalle de confiance à 95 % de la différence entre les deux groupes allait de -1,97x10^{-5} mm à 4,999 mm»

21.5. Kruskal-Wallis (remplacement de l'ANOVA)

Si jamais votre question vous demandait de tester simultanément plus de deux groupes et que vous ne respectiez clairement pas les assomptions de l'ANOVA, vous pouvez toujours vous rabattre sur le test de Kruskal-Wallis. Ce test, comme celui de Wilcoxon, se base sur le rang de vos observations et est en fait de test de comparaison de médianes plutôt que de moyennes.

Comme pour le test de Wilcoxon, le test de Kruskal-Wallis utilise la somme des rangs de chacun des groupes dans un calcul un peu compliqué pour trouver la statistique de test. Les statisticiens ont ensuite déterminé que, lorsque toutes les populations avaient la même médiane (l'hypothèse nulle...), cette statistique suivrait une distribution de khi-carré. C'est donc dans cette distribution que la fonction associée au test ira déterminer la valeur de p associée.

Nous n'irons pas plus dans les détails de ce test, car dans la vraie vie, vous aurez rarement à l'utiliser. Il sera souvent plus utile et productif de transformer vos données pour les conformer aux exigences de l'ANOVA (voir Chapitre 9).

21.6. Labo : Le test de Kruskal-Wallis

Nous reprendrons pour ce laboratoire la même question que celle abordée au Chapitre 15, soit de savoir si la longueur des ailes varie entre les 3 espèces de manchots de l'archipel Palmer.

Étape 0 :

```
library(tidyverse)
library(palmerpenguins)
```

```
pour_anova <- penguins |>
  drop_na(flipper_length_mm)
```

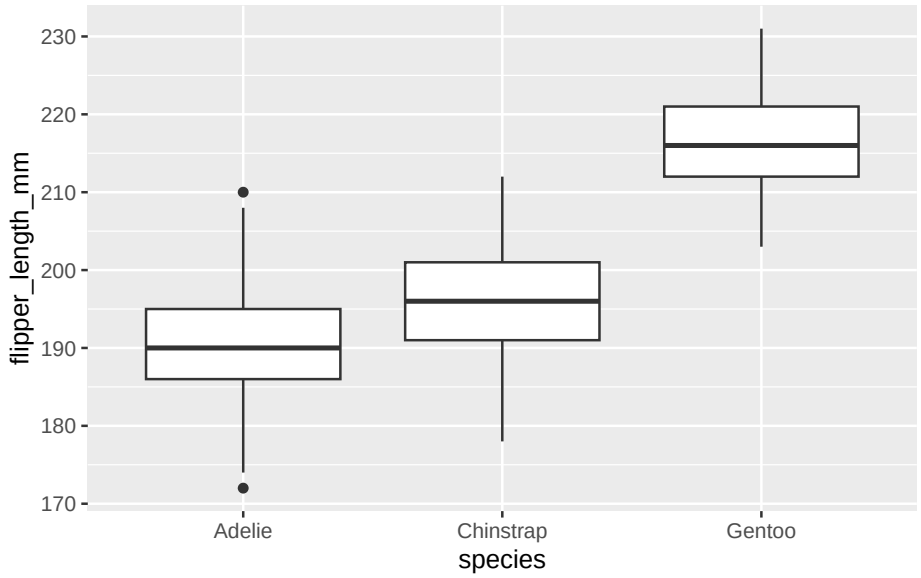
Étape 1 :

H_0 : Toutes les médianes de longueurs d'ailes sont égales entre les différentes espèces.

H_1 : Au moins une des médianes de longueur d'ailes est différente des autres.

Étape 2 :

```
pour_anova |>
  ggplot(aes(species, flipper_length_mm)) +
  geom_boxplot()
```



Étapes 3 et 4 :

```
kruskal.test(flipper_length_mm ~ species, data =
↪  pour_anova)
```

```
    Kruskal-Wallis rank sum test

data:  flipper_length_mm by species
Kruskal-Wallis chi-squared = 244.89, df = 2,
p-value < 2.2e-16
```

Étape 5 :

Comme la valeur de p est fortement sous 0,05, on peut rejeter l'hypothèse nulle d'aucune différence entre les médianes.

Étape 6 :

Ce test, comme l'ANOVA, ne fournit pas directement d'intervalle de confiance associé. On pourrait rapporter nos résultats comme ceci :

«Les espèces de manchots de l'archipel Palmer avaient des ailes de taille significativement différentes (χ^2_2=244,89, p<2,2x10^{-16}).»

Comme le test de Tukey HSD est basé lui aussi sur une distribution normale de nos données, il n'aurait pas pu être appliqué ici (si notre test de Kruskal-Wallis avait été significatif...). Par contre, nous aurions pu appliquer une série de tests de Wilcoxon en appliquant la correction de Bonferroni à l'interprétation des valeurs de p.

21.7. Corrélation de Spearman

Si jamais vous aviez eu besoin de calculer une corrélation entre deux variables, mais qu'une (ou les deux) ne respecte pas suffisamment les

assomptions de la corrélation de Pearson, vous pourriez appliquer à la place la corrélation de Spearman. Cette dernière se calcule exactement comme la corrélation de Pearson, à la nuance que (vous l'aurez peut-être deviné) on remplace chacune des valeurs par son rang avant le calcul.

Dans R, la corrélation de Spearman se calcule comme ceci :

```
propre <- penguins |>
  drop_na(body_mass_g, flipper_length_mm)

cor(
  propre$body_mass_g,
  propre$flipper_length_mm,
  method = "spearman"
)
```

```
[1] 0.8399741
```

La valeur, comme celle de la corrélation de Pearson, va de -1 à +1 et s'interprète exactement de la même façon. Elle mesure le lien (pas nécessairement linéaire cette fois) entre deux variables. Comme pour la corrélation de Pearson, elle n'implique pas nécessairement de lien de cause à effet.

21.8. Tableau synthèse

Voici, pour terminer la section, voici un résumé de toutes les techniques que nous avons vues, organisées par le type de variable étudié

	Qualitative	**Quantitative**
Quantitative	2 groupes seulement : • T à deux échantillons (variances égales) • Welch (variance inégales) • Wilcoxon (non-paramétrique) • T pairé (mesures répétées) 2 groupes ou plus : • ANOVA (variances égales) • Kruskal-Wallis (non-paramétrique) • ANCOVA (avec covariable) Tests post-hoc : Tukey HSD	Régression linéaire Corréations : • Pearson (paramétrique) • Spearman (non-paramétrique) Dépendant de la façon de formuler la question, la famille des tests de T pourrait aussi se retrouver ici
Qualitative	Khi-carré (attention aux valeurs attendues) Exact de Fisher	**Autres techniques** T à un échantillon Loi de Poisson Loi binomiale Comparaison de variance (Test de F)

21.9. Exercices : La boîte à outils

Outre le fait d'être capable d'appliquer chacun des tests statistiques vus dans ce livre, une autre compétence importante à acquérir en tant que biologiste est de savoir quel test appliquer dans quelle situation.

Pierre Magnan, à l'époque, parlait de bien connaître sa "boîte à outils". Je vais donc perpétuer cette tradition en vous proposant une série de mises en situations, pour lesquelles je vous demanderai non pas d'appliquer un test, mais uniquement de nommer quel test ou outil aurait été le plus approprié pour la situation. Bien que je n'ai jamais passé ces évaluations, à l'époque on nous disait que les examens de classement pour entrer aux différents ministères comportaient plusieurs questions de ce genre. Donc il vaut la peine d'être prêt!

À moins d'avis contraire, vous pouvez assumer la normalité des données et l'indépendance des observations. Si plusieurs tests pourraient faire l'affaire, choisissez celui offrant le plus de puissance statistique pour la situation. Si vous n'avez pas d'information concernant les variances, assumez qu'elles sont homogènes entre les groupes.

1. Vous avez mesuré la concentration de chlorophylle dans des feuilles d'érable rouge cueillies en sous-bois et en milieu ouvert. Vous voulez répondre à la question : est-ce que l'érable rouge augmente sa production de chlorophylle dans les milieux mieux éclairés.

2. Vous avez capturé et marqué les écureuils gris dans des milieux urbains et agricoles. Vous voulez savoir si la densité d'écureuil diffère de façon significative entre ces deux milieux. Une analyse préliminaire vous indique que la variance de la densité ne serait pas homogène entre les deux milieux.

3. Vous savez qu'en moyenne, on récolte 3,5 œufs par nid d'oiseaux. Sachant que ces données représentent des décomptes et que la variance associée à la récolte est aussi d'environ 3,5 œufs, quelle est la probabilité de ne récolter aucun œuf dans un nid?

4. Un agriculteur vous affirme qu'il n'a pas appliqué de pesticides sur sa bande riveraine. Vous savez qu'en moyenne, une bande riveraine contient 12 insectes nuisibles par mètre carré de terrain. Vous avez évalué le nombre d'insectes dans une série de parcelles sur sa bande

riveraine et êtes arrivés aux chiffres suivants : [8,5,15,12,14,9,10,6]. Cet agriculteur a-t-il appliqué des pesticides sur sa bande riveraine?

5. Vous avez été mandatée pour évaluer la survie des perchaudes à l'hiver sous la glace du lac Saint-Pierre. Votre première piste est d'évaluer si les perchaudes parviennent à maintenir leur poids pendant l'hiver. Vous avez donc capturé à l'automne 50 perchaudes, que vous avez pesées et marquées. Au printemps, à l'aide des puces installées, vous en retrouvez 5 que vous pesez à nouveau.

Voici le poids des perchaudes à l'automne : [1108, 1200, 1004, 801, 1500] grammes

Et celui à l'hiver, dans le même ordre [1102, 1230, 906, 765, 1200] grammes

Évaluez si ces perchaudes ont subi une perte significative de poids durant l'hiver.

6. Vous voulez tester une théorie selon laquelle les contraintes climatiques limitent le développement physiologique des insectes. Vous avez capturé une série de papillons en toundra et en milieu tropical et vous devez maintenant évaluer si ces mesures sont plus variables en milieu tropical qu'en toundra.

7. Vous savez, par les informations fournies par le fabricant, que 3 % de vos colliers émetteurs cesseront de fonctionner pendant une saison de terrain. Combien de colliers devrez vous installer si vous voulez être certain à 95 % de récolter l'information provenant de 15 colliers au terme de la saison?

8. Vous savez qu'en moyenne, une espèce de serpent produit 8,2 petits par portée. Sachant que ces données représentent des décomptes et que la variance associée au nombre de petits est aussi d'environ 8,2 petits, quelle est la probabilité qu'une portée produise 2 petits ou moins?

9. Vous savez qu'en moyenne seulement 30 % des oiseaux survivent à leur première migration. Si vous marquez 60 oiseaux à l'automne, quelle est la probabilité d'en retrouver 25 vivants ou plus l'année suivante?

10. Vous avez mesuré le nombre d'écureuils roux dans des parcelles de conifères et des parcelles de feuillus. Vous voulez répondre à la question : est-ce que l'écureuil roux est présent en plus grande densité dans les parcelles de conifères.

11. Vous avez mesuré les taux de mercure et le poids d'une série de poissons. Vous devez maintenant déterminer si une règle simple permettrait de savoir quel taux de mercure on peut s'attendre à retrouver en moyenne dans la chair d'un poisson d'un poids donné.

12. Vous avez été mandatée pour évaluer si les efforts de réhabilitation d'anciens stationnement portent fruit pour toute la biodiversité urbaine. Vous avez, pour chaque stationnement mesuré la richesse en espèces d'insectes avant la réhabilitation et avez ensuite revisité chacun des stationnements 4 ans plus tard pour prendre les mêmes mesures. Vous voulez maintenant savoir si la réhabilitation a effectivement favorisé la richesse en espèces d'insectes.

13. Vous avez mesuré les taux de mercure dans la chair d'une série de poissons. Vous avez noté pour chacun l'espèce à laquelle le poisson appartenait (perchaude, achigan, brochet, crapet), et vous voudriez maintenant déterminer si les taux de mercure varient en fonction de l'espèce.

14. Vous avez récolté des données permettant d'explorer la relation entre la richesse en espèce végétale et la productivité primaire. Vous voulez savoir si ces variables sont reliées, mais vous ne pouvez déterminer a priori d'hypothèses à savoir si la richesse augmente la productivité ou la productivité augmente la richesse.

15. Vous avez capturé et marqué les bruants chanteurs dans des milieux urbains et agricoles. Vous voulez savoir si la densité de bruants chanteurs diffère de façon significative entre ces deux milieux. Une analyse

préliminaire vous indique que la variance de la densité ne serait pas homogène entre les deux milieux.

16. Vous avez mesuré la concentration de chlorophylle dans des feuilles d'érable rouge cueillies en sous-bois et en milieu ouvert et en milieu humide. Vous voulez répondre à la question : est-ce que l'érable rouge modifie sa production de chlorophylle en fonction du milieu. Cependant, vous savez aussi que la production de chlorophylle peut-être influencée par la taille des feuilles. Vous avez donc pour chaque feuille, mesuré sa surface, sa teneur en chlorophylle et le type de milieu dans lequel elle a été trouvée.

17. Vous avez capturé et marqué les bruants chanteurs dans des milieux urbains, agricoles et forestiers. Vous voulez savoir si la densité de bruants chanteurs diffère de façon significative entre ces trois milieux. Une analyse préliminaire vous indique que vos données ne suivent clairement pas une distribution normale.

18. Vous avez mesuré le nombre de sittelles à poitrine rousse dans des parcelles de conifères et des parcelles de feuillus. Vous voulez répondre à la question : est-ce que cette espèce de sittelle est présente en plus grande densité dans les parcelles de conifères. Cependant, en explorant vos données, vous remarquez qu'elles contiennent beaucoup plus de valeurs "0" que l'on pourrait s'attendre pour une distribution normale.

19. Vous avez noté, pour une série de parcelles, le nombre de frênes d'amérique et de frênes rouges. Certaines de vos parcelles étaient inondées, d'autres non. Vous voulez déterminer si le type de frêne trouvé est associé au fait que la parcelle soit inondée ou non. Les parcelles étaient particulièrement grandes, et donc le nombre de frênes attendus dans chaque parcelle était plutôt élevé, jamais sous les 20 individus.

20. Vous avez récolté des données permettant d'explorer la relation entre la richesse en espèce végétale et la richesse en espèces d'insectes. Vous voulez savoir si ces variables sont reliées, mais vous ne pouvez déterminer a priori d'hypothèses à savoir si les insectes augmentent les plantes

(par pollinisation) ou l'inverse (par broutement). À première vue, vos données sont loin de suivre une distribution normale, puisque la richesse en espèces d'insectes était souvent de 0 ou 1.

21. Vous avez dénombré le nombre de plantes envahissantes et non-envahissantes dans des milieux urbains et naturels. Vous voulez maintenant déterminer si le type de milieu et la présence d'espèces envahissantes sont reliées. Comme il y avait très peu d'espèces envahissantes en milieu naturel, votre nombre d'individus attendus pour ce milieu est < 1.

partie IV.

L'exploration des données multivariées

22. Matrices et distances

22.1. Introduction

Un peu comme nous l'avions fait pour les tests statistiques en mettant en place plusieurs concepts (distributions, intervalle de confiance, etc.) avant d'attaquer le vif du sujet, nous devrons aussi mettre en place plusieurs concepts avant de s'attaquer au sujet des ordinations comme tel dans les prochains chapitres.

Ces nouvelles notions sont nécessaires parce que, contrairement aux tests statistiques qui étudiaient au maximum deux variables à la fois, les ordinations peuvent s'attaquer à des dizaines ou même des centaines ou des milliers de variables en même temps. À moins que vous vouliez regarder un à un des centaines de nuages de points pour comprendre vos données, vous aurez besoin d'outils appropriés à ce genre de problèmes.

Nous verrons d'abord quatre matrices, qui permettent de décrire, chacune à leur façon, nos données. Nous devrons aussi voir le concept de distance multivariée, et nous verrons quelques façons différentes de mesurer cette distance.

22.2. La matrice de données

Commençons doucement avec la matrice de données. Cette dernière n'a rien de sorcier. Il s'agit essentiellement de notre tableau de données, mais avec une nuance importante : une **matrice** ne peut contenir qu'un seul type de données. Soit du texte, ou soit des chiffres, mais jamais les deux mélangés.

Pour appliquer des ordinations, vous aurez donc besoin de nettoyer vos tableaux de données pour ne conserver que les variables quantitatives. Nous verrons au Chapitre 29 comment transformer des variables qualitatives en quantitatives, mais nous ne compliquerons pas les choses pour le moment.

Comme pour notre tableau de données, il sera important que votre matrice de données soit *dans le bon sens*, c'est-à-dire que chacune des colonnes soit une variable et que chacune des lignes soit une observation. Afin de faciliter la discussion, nous introduirons aussi une notation pour ces deux concepts. Le nombre d'observations dans une matrice (le nombre de lignes) sera noté par la lettre n, et le nombre de variables (de colonnes) par la lettre p.

Cette matrice contenant des observations sur une série de poissons capturés dans un lac pourrait être définie par n=5 et p=3 :

Longueur (cm)	Poids (kg)	Profondeur (m)
80	3	4
60	2	3
10	15	2
5	28	1
4	2	3

22.3. La matrice de la somme des carrés et des produits croisés

Cette deuxième matrice est sans doute celle dont les calculs sont les plus abstraits et complexes à effectuer parmi celles que nous verrons dans ce chapitre.

Elle contient deux informations distinctes. Sur sa diagonale, elle contient la variabilité de chacune des variables par rapport à sa moyenne respective, que l'on nomme la **somme des carrés**. Dans le reste de la matrice, on retrouvera une idée de comment deux variables varient ensemble ou non, que l'on nomme la **somme des produits croisés**.

Voici, par exemple, la matrice de la somme des carrés et des produits croisés pour la matrice de données de la section précédente :

	Longueur	Poids	Profondeur
Longueur	5084.8	-932	123.6
Poids	-932	526	-48
Profondeur	123.6	-48	5.2

Remarquez d'abord que nous passons d'une matrice de données de p x n à une matrice de p x p. Peu importe que notre matrice de données ait eu 3 ou 1000 observations, la matrice de la somme des carrés et des produits croisés aura toujours autant de lignes et de colonnes que notre matrice originale avait de variables (p).

Le calcul pour arriver à chacune des ces nouvelles valeurs peut être défini par l'équation suivante :

$$\sum_{i=1}^{n}(y_{im} - \bar{y}_m)(y_{il} - \bar{y}_l)$$

Pour les lignes i à n de notre matrice de données, où m et l sont les deux colonnes pour lesquelles ont veut faire le calcul.

Par exemple, pour arriver à la valeur -932, nous aurions donc procédé ainsi :

- Trouver la moyenne de la variable Poids : 10
- Trouver la moyenne de la variable Longueur : 31,8
- Pour la première ligne de la matrice : (3-10) * (80-31,8) = -337,4
- ...
- Pour la dernière ligne de la matrice : (2-10) * (4-31,8) = 222,4
- Faire la somme de toutes ces valeurs : -337,4 + ... + 222,4 = -932

Il n'est pas si important d'être capable de répliquer ce calcul manuellement. Par contre, il faut être capable de bien interpréter la matrice. Voici quelque exemples de questions pour vérifier votre compréhension :

A) Quelle variable varie le plus par rapport à sa moyenne?
B) Quelle variable varie le moins par rapport à sa moyenne?

(A: Longueur, B : Profondeur)

22.4. La matrice de variance-covariance

Vous avez sans doute remarqué que puisque la matrice précédente contient une série de sommes, les valeurs de chacune des cellules seront d'autant plus grandes que nous avons de lignes dans la matrice de données. Cela rend à peu près impossible une quelconque interprétation biologique des chiffres qui s'y retrouvent. C'est pourquoi avant l'interprétation, on divise chaque cellule par son degré de liberté (n - 1), pour en arriver à la matrice de variance/covariance :

	Longueur	Poids	Profondeur
Longueur	1271,2	-233,0	30,9
Poids	-233,0	131,5	-12
Profondeur	30,9	-12	1,3

Sur la diagonale de cette matrice, nous retrouvons donc la variance de chacune de nos variables, et dans les autres cellules, on retrouve ce que l'on appelle la **covariance** entre deux variables. Autrement dit, est-ce que deux variables varient fortement ensemble ou non, et si oui, dans quel sens.

Encore une fois, il faut être capable d'interpréter correctement cette matrice en répondant à des questions du genre :

A) Quelle variable présente la variance la plus faible?
B) Quelle variable présente la variance la plus élevée?
C) Quelles variables covarient le plus fortement?
D) La profondeur et la longueur sont-elles reliées positivement ou négativement?
E) Quelles variables semblent le moins reliées?

(A: Profondeur, B: Longueur, C: Poids et Longueur, D: Positivement, E: Profondeur et Poids)

Remarquez que, comme la matrice précédente, la matrice de variance-covariance est symétrique. La covariance entre Poids et Longueur est exactement la même qu'entre Longueur et Poids. Il arrivera fréquemment que seul le triangle inférieur ou supérieur de la matrice vous soit présenté pour économiser de l'espace.

22.5. La matrice de corrélation

Vous avez peut-être remarqué quelque chose qui clochait lors de vos interprétations de la matrice de variance-covariance? Les valeurs associées à la variable de Poids étaient toutes très élevées (en valeurs absolues) alors que celles associées à la profondeur étaient systématiquement très petites. Ce phénomène est dû au fait que la variance (et la covariance) se mesure à la même échelle que les données. Les valeurs associées à la profondeur sont petites parce que cette dernière a été mesurée en mètres. Si elle avait été mesurée en cm, elle aurait plutôt eu des valeurs très grandes, probablement les plus grandes de la matrice. Quelles sont donc vraiment les deux variables les plus associées, si on ne peut pas se fier aux chiffres de la matrice de variance/covariance?

Une façon d'y arriver est de transformer notre matrice de variance-covariance en matrice de corrélation. Vous vous rappelez sans doute que cette dernière sert justement à remettre toutes les relations à une même échelle, allant de -1 à +1, et ce, peu importe comment les données ont été mesurées (voir Chapitre 17 pour un rappel).

Nous avions énoncé le modèle conceptuel de la corrélation comme la covariance entre deux variables, divisé par le produit de leurs écart-types. Si nous appliquons cette transformation sur la matrice précédente, nous obtenons la matrice de corrélation, comme celle-ci pour l'exemple sur les poissons :

	Longueur	Poids	Profondeur
Longueur	1	-0,57	0,76
Poids	-0,57	1	-0,92
Profondeur	0,76	-0,92	1

Cette matrice contient donc, à chacune des intersections, la corrélation entre chacune des variables. La longueur et le poids sont par exemple

corrélés négativement, avec un r de -0,57. Cette matrice contient évidemment une série de 1 sur la diagonale, puisque chaque variable est parfaitement corrélée avec elle-même.

22.6. Le concept de distance multivariée

La covariance et la corrélation sont des façons plutôt intuitives de définir l'association entre deux variables, soit à l'échelle originale des données (variance-covariance) ou soit à une échelle standardisée allant de -1 à 1 (corrélation).

Par contre, si on tourne le problème de côté et qu'on se demande comment sont associées **deux observations** dans notre tableau de données (i.e. combien deux lignes se ressemblent), nous avons besoin d'outils différents : les distances multivariées.

Eh oui, étrangement, les statisticiens ont décidé que, lorsque l'on voulait savoir si deux observations se ressemblent ou non, ou n'allions pas mesurer si elles sont proches, mais bien si elles sont loins l'une de l'autre. Il s'agit là d'une des clés pour bien saisir les sections suivantes. Plus la distance entre deux observations est élevée, plus les observations sont différentes l'une de l'autre. Plus la distance est petite, plus les observations sont semblables. Comprenez bien ici que l'on discute d'une **distance conceptuelle**; on ne parle pas nécessairement de cm ou de km.

L'autre chose importante à savoir est qu'il existe, selon Numerical Ecology de Legendre & Legendre, au moins 26 (!) façons différentes de mesurer la distance entre deux observations. Toutes ces mesures existent pour tenir compte des particularités statistiques et biologiques des données que vous pourriez rencontrer. Mais rassurez-vous, dans ce cours, nous n'en verrons que 2 ½, qui devraient répondre à la majorité des besoins que vous rencontrerez.

22.7. La distance euclidienne

Si les observations dans votre matrice de données sont composées de variables continues (et non de présences/absences ou de décomptes d'individus), la façon la plus simple et la plus intuitive de décrire la distance entre elles est ce que l'on appelle la distance euclidienne.

Malgré son nom un peu ésotérique, vous avez souvent rencontré cette distance par le passé, puisque si vous n'avez que deux variables, elle est équivalente à l'hypoténuse d'un triangle rectangle, qui se définit comme ceci :

$$c^2 = a^2 + b^2$$

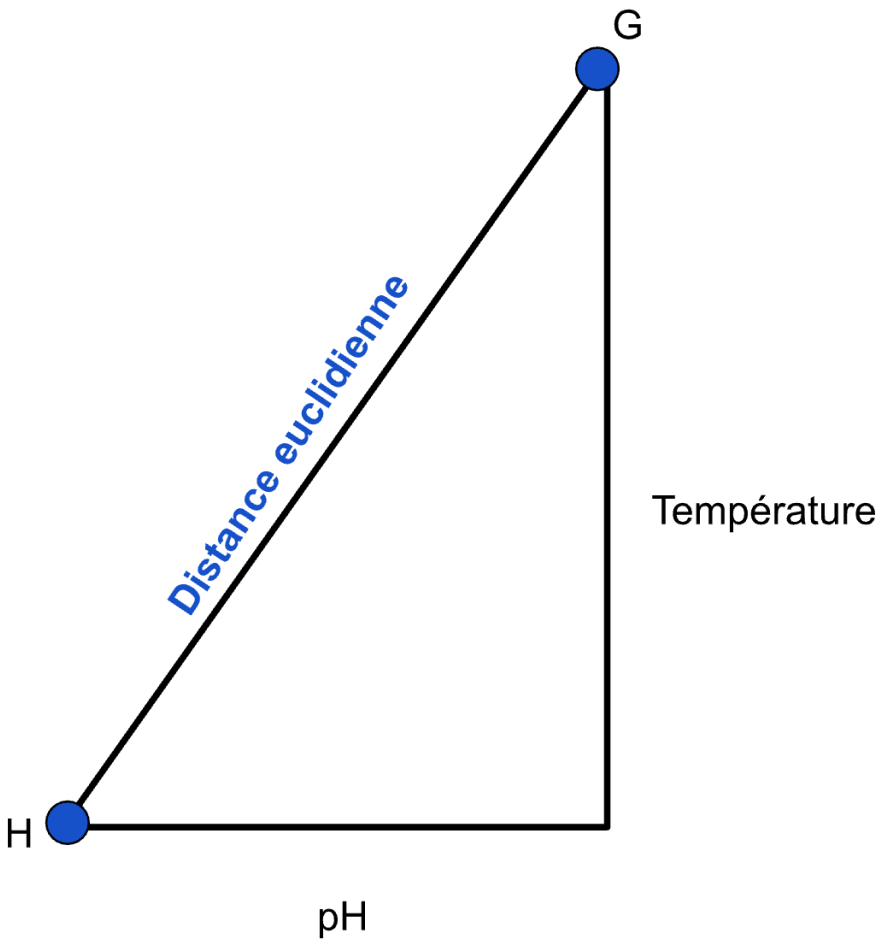
Où c est l'hypoténuse et a et b sont la longueur de chacun des côtés du triangle. L'hypoténuse se calcule donc comme ceci :

$$c = \sqrt{a^2 + b^2}$$

Si par exemple vous aviez un mini-tableau de données comme celui-ci :

Site	pH	Température
G	7	25
H	6.5	21

Si vous voulez connaître la distance euclidienne entre ces deux sites, vous devez d'abord visualiser un triangle où la température serait d'un côté et le pH de l'autre, un peu comme ceci :



La distance euclidienne entre ces deux observations se calculerait donc comme ceci :

$$\sqrt{(7-6.5)^2+(25-21)^2} = 4,03$$

Remarquez que lorsque l'on calcule des distances multivariées, le concept d'unités est un peu laissé de côté. Vous ne lirez jamais que ces deux observations sont à 4,03 pH-degrés l'une de l'autre.

Une fois cet exemple derrière nous, la définition formelle de la distance euclidienne est plutôt intuitive, puisqu'elle est l'équivalent d'une hypoténuse en p dimensions. Notre cerveau humain n'a aucun problème à imaginer une hypoténuse en 2 dimensions comme dans l'exemple précédent. Nous sommes aussi, avec un peu plus de travail, capables d'en imaginer une en 3 dimensions, mais l'ordinateur lui n'a aucun problème à la calculer en 4, 10, ou même 200 dimensions. Afin de rendre la notation plus compacte, la distance euclidienne est habituellement définie par la formule suivante :

$$\sqrt{\sum_{j=1}^{p}(y_{1j} - y_{2j})^2}$$

Évidemment, comme toutes les autres formules présentées dans ce cours, je vous la présente à titre de référence. Je ne vous demanderai jamais de me recracher cette formule par cœur. Si vous comprenez le principe que la distance euclidienne est comme une hypoténuse en plusieurs dimensions, vous en comprenez bien assez pour le moment. Remarquez cependant que, de par sa définition, la distance euclidienne la plus courte possible est zéro (pour des observations identiques) et qu'elle n'a pas de limite supérieure.

Comme vous vous en doutez peut-être, une distance seule, par elle-même, a peu d'utilité. On calcule habituellement une matrice de distances, à partir de toutes les observations d'une matrice de données. Si nous avions par exemple une série de mesures prises sur des espèces d'oiseaux comme ceci :

Espèce	Longueur (cm)	Envergure (cm)	Poids (g)
Durbec	23	37	56
Oriole	22	29	33
Pic	32	51	130
Canard	51	84	910
Busard	45,7	189,2	424,5

La matrice de distances euclidiennes correspondantes aurait l'air de ceci :

	Durbec	Oriole	Pic	Canard
Oriole	24,37			
Pic	75,84	99,96		
Canard	855,75	879,20	780,92	
Busard	376,19	400,33	300,51	486,18

Comme pour les matrices de variance-covariance et de corrélation, il est important de pouvoir bien interpréter une telle matrice, par exemple :

A. Quelles espèces sont les plus différentes les unes des autres?

B. Quelles espèces sont les plus semblables?

C. À quelle espèce le pic flamboyant ressemble-t-il le plus?

(A : Oriole-Canard; B : Oriole-Durbec; C: Durbec)

22.8. Centrer et réduire

La distance euclidienne présente une particularité importante à laquelle il faut être très attentifs : elle est très sensible aux différences d'échelles

entre nos variables. Si les unités d'une de nos variables sont beaucoup plus grandes que celles des autres, cette dernière contrôlerait à elle-seule ou presque le calcul de l'hypoténuse. Bien que l'on entrerait toutes les variables dans le calcul, le résultat représenterait essentiellement les différences dans notre variable ayant les grands chiffres.

La solution pour éviter ce genre de problème est de **centrer et réduire** (*to scale*) nos variables avant de les entrer dans le calcul. C'est-à-dire de soustraire la moyenne (centrer) et de diviser par l'écart-type (réduire) pour chacune des données.

Prenons ce mini-tableau de données comme exemple :

Variable A	Variable B
1	1000
2	2000

Vous constatez facilement que l'échelle de mesure de la variable B est beaucoup plus grande que celle de la variable A. Si on applique l'opération de centre et réduire chacune des variables, on obtient maintenant le tableau suivant :

Variable A	Variable B
-0,707	-0,707
0,707	0,707

Autrement dit, la première observation de la variable A est 0,707 écarts-types sous la moyenne, tout comme la première de la variable B. Les deux variables (A et B) auraient donc maintenant un poids équivalent dans le calcul de la distance euclidienne.

Dans la vraie vie, vous devrez dans presque tous les cas centrer et réduire vos variables avant d'appliquer le calcul de la distance euclidienne. Le

seul cas où ce n'est pas recommandé est si toutes vos variables sont déjà à la même échelle. Par exemple, si vos variables étaient toutes des températures, disons une pour le matin, une pour le midi, une pour le soir, etc.

22.9. La distance de Bray-Curtis

La distance euclidienne fonctionnera bien pour beaucoup de contextes, mais pour certains cas particuliers, elle pourrait vous causer de gros ennuis. Ce sera le cas si votre matrice de données contient des abondances d'espèces, en particulier si cette dernière contient beaucoup de zéros, par exemple comme ceci :

	Geai	Viréo	Oriole
Site A	0	1	1
Site B	1	0	0
Site C	0	4	4

Si vous calculez les distances euclidiennes sur cette matrice, le calcul vous informera que le site A ressemblerait plus au site B qu'au site C :

	Site A	Site B
Site B	1,73	
Site C	4,24	5,74

Ces distances sont clairement erronées d'un point de vue écologique. La composition en espèces de A est en fait identique à celle de C.

Pour cette raison, lorsque vous analysez des matrices d'abondances d'espèces, il est recommandé d'utiliser plutôt la distance de Bray-Curtis,

que l'on appelle aussi parfois le coefficient de Czekanowski ou le pourcentage de dissimilarité. Cette distance peut aller de zéro (des sites complètement identiques) à un (pour des sites complètement différents dans leur composition).

Nous verrons ensemble le calcul pour que vous ayez une idée d'où les chiffres viennent, mais vous n'aurez jamais à les calculer vous-même manuellement.

Voyons d'abord la formule :

$$1 - \frac{2\sum_{j=1}^{p} min(y_{1j}, y_{2j})}{\sum_{j=1}^{p}(y_{1j} + y_{2j})}$$

Yark, ça fait peur. Mais allons-y morceau par morceau, puisque le calcul se fait en 3 étapes principales. On regarde d'abord pour chaque espèce le minimum entre les deux sites, que l'on multiplie par 2, puis on en fait la somme. On divise ensuite ce total par l'abondance totale de toutes les espèces aux deux sites. Et finalement, on fait 1 moins cette valeur.

L'intuition à comprendre derrière le calcul est que si l'abondance de chaque espèce est égale aux deux sites, la partie d'en haut et d'en bas seront absolument égales et leur ratio sera 1.

Et voici à titre d'exemple le résultat de ce calcul sur la matrice de données précédente :

	Site A	Site B
Site B	1	
Site C	0,6	1

On récupère donc maintenant des distances qui ont du bons sens écologiquement. Remarquez que les sites A et C ne sont pas considérés comme

identiques, puisque bien que leur composition soit la même, les abondances sont différentes entre les deux sites.

22.10. L'indice de Jaccard

Le dernier scénario que nous verrons dans ce cours est celui où votre matrice de données, plutôt que de contenir des données continues ou des décomptes, contient en fait des données de présence-absence, c'est-à-dire des 1 ou des 0 uniquement. Dans ce genre de situation, un des calculs de distance souvent recommandé est l'indice de Jaccard.

La formule associée à l'indice de Jaccard est celle-ci :

$$1 - \frac{a}{a + b + c}$$

Où a est le nombre d'espèces communes aux deux sites, b est le nombre d'espèces uniques au site 1 et c et le nombre d'espèces uniques au site 2. L'indice de Jaccard sera donc de 0 si toutes les espèces sont les mêmes entre les sites, et de 1 si toutes les espèces sont différentes.

Par contre, plusieurs auteurs recommandent plutôt d'utiliser le coefficient de Sorensen pour calculer les distances dans des scénarios de présence-absence. La différence majeure entre le coefficient de Sorensen et celui de Jaccard étant que celui de Sorensen donne plus d'importance dans son calcul aux espèces communes qu'aux différences.

Ce qu'il est intéressant de savoir ici est que le calcul de Sorensen est exactement identique à celui de Bray-Curtis. Si on applique la formule sur une matrice d'abondance, on parle de Bray-Curtis, et sur une matrice de présence-absence, on parle alors de coefficient de Sorensen.

22.11. Labo : Les matrices et les distances

Nous verrons maintenant comment calculer avec R les matrices vues dans les sections précédentes. Remarquez que ces codes sont fournis plutôt à titre informatif, puisque la plupart du temps, vous n'aurez pas à faire ces calculs manuellement. Ce sont des étapes intermédiaires dans le calcul des ordinations. Il n'y aura donc pas d'exercices associés.

Nous allons comme c'est notre habitude, utiliser le tableau de données des manchots de l'archipel Palmer, sur lequel nous allons d'abord calculer les matrices de la somme des carrés et des produits croisés (`crossprod`), de variance-covariance puis de corrélation.

Avant de commencer, nous devrons cependant limiter notre tableau de données à 3 variables pour simplifier les sorties, et éliminer les lignes contenant des valeurs manquantes.

```
library(palmerpenguins)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
pour_matrices <-
  penguins |>
  select(body_mass_g, flipper_length_mm,
   ↪  bill_length_mm) |>
  drop_na()
```

```
pour_matrices |> as.matrix() |> crossprod()
```

```
                  body_mass_g flipper_length_mm
body_mass_g        6257228750         292065275
flipper_length_mm   292065275          13872913
bill_length_mm       64004320           3035186
                  bill_length_mm
body_mass_g           64004320.0
flipper_length_mm      3035185.7
bill_length_mm          669928.7
```

```
pour_matrices |> var()
```

```
                  body_mass_g flipper_length_mm
body_mass_g        643131.077        9824.41606
flipper_length_mm    9824.416         197.73179
bill_length_mm       2605.592          50.37577
                  bill_length_mm
body_mass_g           2605.59191
flipper_length_mm       50.37577
bill_length_mm          29.80705
```

```
pour_matrices |> cor()
```

```
                  body_mass_g flipper_length_mm
body_mass_g         1.0000000         0.8712018
flipper_length_mm   0.8712018         1.0000000
bill_length_mm      0.5951098         0.6561813
                  bill_length_mm
body_mass_g            0.5951098
flipper_length_mm      0.6561813
bill_length_mm         1.0000000
```

Remarquez que la fonction `crossprod` est un peu têtue et on doit absolument convertir notre tableau de données en objet matrice avant qu'elle puisse faire son calcul, alors que les deux autres le font elles-mêmes sans se plaindre.

Maintenant, pour les calculs de distances, vous aurez aussi besoin d'une librairie additionnelle qui se nomme `vegan`. Nous utiliserons cette librairie pour la grande majorité des calculs d'ordinations des prochains chapitres.

Nous nous créerons aussi un nouveau petit de tableau de données basé sur les manchots de Palmer. Il aurait été utile et légitime de calculer des distances entre chacun des individus, mais il sera beaucoup plus facile de comprendre si on se fait un mini-tableau avec une ligne par espèce :

```
mini_tableau <-
  penguins |>
  drop_na(body_mass_g, flipper_length_mm,
   ↪  bill_length_mm) |>
  group_by(species) |>
  summarize(
    body_mass_g = mean(body_mass_g),
    flipper_length_mm = mean(flipper_length_mm),
    bill_length_mm = mean(bill_length_mm)
```

```
  )
mini_tableau
```

```
# A tibble: 3 x 4
  species  body_mass_g flipper_length_mm bill_length_mm
  <fct>          <dbl>             <dbl>          <dbl>
1 Adelie         3701.              190.           38.8
2 Chinstr~       3733.              196.           48.8
3 Gentoo         5076.              217.           47.5
```

Nous utiliserons aussi, pour la première fois, la fonction `column_to_rownames` de la librairie `tibble` (inclue dans le `tidyverse`). Il s'agit en fait d'une technicalité, mais cette fonction permet d'éliminer une colonne de notre tableau, et de plutôt mettre l'information de cette colonne dans les méta-informations du tableau. C'est un peu de *gossage*, mais ça permet que les fonctions calculant les matrices de distances puissent nous afficher à quelle ligne était associée chaque distance, sans que le calcul bogue parce qu'on a une colonne qui n'est pas des chiffres.

```
mini_tableau <-
  mini_tableau |>
  column_to_rownames("species")
```

Remarquez que les noms de lignes doivent être uniques. Si vous n'avez pas une colonne permettant d'identifier de façon unique chacune des lignes, vous ne pourrez pas appliquer `column_to_rownames` pour retrouver chacune des observations dans les sorties de `vegan`. C'est tannant, mais pas dramatique.

Une fois cette librairie activée, les calculs de distances se feront avec la fonction `vegdist`, à laquelle il faudra passer comme argument le nom de la distance que l'on veut calculer :

```
library(vegan)
```

```
Loading required package: permute
```

```
Loading required package: lattice
```

```
This is vegan 2.6-8
```

```
mini_tableau |> vegdist(method = "euclidian")
```

```
             Adelie  Chinstrap
Chinstrap   34.44924
Gentoo    1375.65120 1343.09860
```

```
mini_tableau |> vegdist(method = "bray")
```

```
              Adelie   Chinstrap
Chinstrap 0.006113238
Gentoo    0.152241993 0.146550112
```

```
mini_tableau |> vegdist(method = "jaccard")
```

```
             Adelie  Chinstrap
Chinstrap 0.01215219
Gentoo    0.26425351 0.25563664
```

Remarquez ici que j'utilise la distance de Bray-Curtis et de Jaccard sur des données continues uniquement à titre d'illustration des arguments de la fonction. Il n'aura pas été pertinent de faire cela dans la vraie vie.

Enfin, si on veut centrer-réduire nos données avant de les envoyer au calcul de distance, R possède une fonction toute prête pour le faire, nommée `scale`. On pourrait par exemple l'utiliser comme ceci :

```
mini_tableau |>
  scale() |>
  vegdist(method = "euclidian")
```

```
           Adelie Chinstrap
Chinstrap 1.886421
Gentoo    3.038529  2.282269
```

Basé sur la distance euclidienne, les deux espèces les plus semblables sont les manchots Adélie et les manchots Chinstrap Les deux plus différentes sont les Adélie et les Gentoo.

22.12. Résumé

Si l'on résume ce que l'on vient de voir, il existe deux matrices pour mesurer l'association entre les variables dans un tableau de données :

- La matrice de variance-covariance, qui est affectée par l'échelle des données
- La matrice de corrélation, qui permet de remettre toutes les variables à la même échelle et éviter ces problèmes

Si l'on veut mesurer la distance entre les observations, il existe au moins trois calculs différents :

- La distance euclidienne pour les données continues (attention aux différences d'échelle)
- La distance de Bray-Curtis pour les décomptes
- La distance de Jaccard pour les présences-absences

On peut aussi utiliser la distance de Bray-Curtis (que l'on nomme alors Sorensen) pour les présences-absences, en se rappelant qu'elle donne plus de poids aux espèces communes qu'aux différences.

23. L'analyse en composantes principales

23.1. Le principe

Les méthodes d'ordination que nous verrons dans ce chapitre et les deux suivants (Chapitre 24 et Chapitre 25) fonctionnent toutes sur le même principe. Vous allez voir que cela est plutôt abstrait au début, mais au moment où ça clique, on réalise la puissance et la simplicité de cette approche. Ne stressez pas trop si au terme du cours le fonctionnement des ordinations est encore un peu flou dans vos têtes. Personnellement, je ne les ai vraiment comprises qu'une fois à la maîtrise. L'important sera surtout de savoir bien les appliquer et les interpréter.

Les ordinations partent de notre matrice de données originale, et la transforment en une nouvelle matrice contenant autant de variables que l'originale. Cependant, contrairement à notre matrice originale, les variables composant cette nouvelle matrice seront toujours entièrement **orthogonales** (i.e. non corrélées les unes avec les autres). Cela en soit peut être utile dans certaines situations, mais ce qui est surtout intéressant des ordinations est qu'elles rassemblent (ou résument) la variabilité (e.g. la variance) de nos données dans les premières variables de la nouvelle matrice.

Cela veut dire que si par exemple nous avions besoin de 30 variables pour décrire une observation dans notre matrice de données originale (dans la vraie vie, ça arrive plus souvent qu'on le voudrait!), avec une

ordination, on pourrait potentiellement réduire ce nombre à 3 ou 4. Cela simplifie grandement les interprétations et les analyses subséquentes. On pourrait constater que plusieurs variables étaient redondantes, etc. Les ordinations sont donc, avant tout, des méthodes de simplification des données. Il s'agit là de la clé importante pour bien saisir les ordinations.

> ⚠ Avertissement
>
> Ces méthodes ne cherchent PAS de lien cause à effet, elles ne trouvent pas d'explication, elle ne regardent pas de différence entre des groupes, etc. Elles ne font que réorganiser et simplifier les données.

Voici à titre d'illustration quelques bonnes questions associées à des ordinations :

- Est-il possible de résumer les caractéristiques du sommeil des mammifères en un nombre réduit de variables, ce qui faciliterait leur interprétation?
- Existe-t-il des patrons (ou des tendances etc.) dans les variables décrivant le climat des villes canadiennes?
- Comment sont reliés les différents descripteurs du milieu forestier?
- Comment s'organisent les communautés d'oiseaux du parc national de la Mauricie

Et au contraire, voici quelques exemples de mauvaises questions, qui ne peuvent PAS se répondre à l'aide d'une ordination :

- Quels sont les facteurs reliés à la fermeture de la canopée?
- Existe-il des différences significatives entre XXXXXXXX et YYYYY
- Quelles sont les causes d'un allongement des cycles de sommeil chez les mammifères?
- La proportion de villes plus froides augmente-elle au fil du temps?

23.2. L'aspect technique

Vous vous en doutez peut-être, mais il y a une certaine logique derrière la construction de la nouvelle matrice résultant d'une analyse en composantes principales (ACP). La nouvelle matrice n'est évidemment pas générée au hasard.

La première chose à savoir est qu'il existera une recette pour passer d'une matrice à l'autre. Cette recette se nomme les **eigenvector**. Elle nous permet de passer de la matrice originale à la nouvelle matrice, en nous disant comment construire les nouvelles variables à partir des originales. Ces nouvelles variables, dans l'ACP, se nomment **composantes principales** (d'où le nom de l'analyse...). Vous verrez que l'on utilise aussi souvent le terme axe ou axe principal.

Si notre matrice originale contenait trois variables et se nommait y, l'eigenvector de la première composante principale (z_i) pourrait ressembler à ceci :

$$z_1 = 2 \times y_{1,1} - 3 \times y_{1,2} + 4 \times y_{1,3}$$

Autrement dit, pour créer la première composante principale, on prend un peu de y1, beaucoup de y3 et on soustrait du y2. Comme notre matrice originale contenait trois variables, il existera deux autres recettes semblables pour définir les composantes principales 2 et 3.

L'autre aspect important du travail de l'ACP, comme nous en avons discuté plus haut, est de rassembler la variance dans les premières composantes principales. C'est ici qu'entrent en jeu les matrices de variance-covariance et de corrélation vues au Chapitre 22. Si nous reprenons l'exemple de ce chapitre avec les poissons dont nous avions noté la longueur, le poids et la profondeur de capture, rappelez-vous que la matrice de variance-covariance ressemblait à ceci :

	Longueur	Poids	Profondeur
Longueur	1271,2	-233,0	30,9
Poids	-233,0	131,5	-12
Profondeur	30,9	-12	1,3

Au terme du calcul de l'ACP, la nouvelle matrice de variance-covariance des composantes principales ressemblerait à ceci :

	Axe 1	Axe 2	Axe 3
Axe 1	1317,8	0	0
Axe 2	0	86,1	0
Axe 3	0	0	0,1

C'est ici qu'il faut apprendre (encore!) un nouveau mot de vocabulaire, soit les **eigenvalues**. C'est le terme technique qui désigne la variance de chacune des composantes principales au terme de l'analyse. Remarquez d'abord que toutes les covariances sont à zéro : les composantes principales sont orthogonales les unes par rapport aux autres. Remarquez ensuite que l'axe 1 est celui qui possède la plus grande eigenvalue, suivi de l'axe 2 et de l'axe 3. Dans le cas présent, comme nos variables étaient peu corrélées ensemble, il n'y a pas beaucoup de différences entre l'eigenvalue de l'axe 1 et la variance de notre variable originale ayant le plus de variance (la longueur), mais parfois ces différences pourraient être énormes.

Comme nous en avons discuté dans le Chapitre 22, interpréter la matrice de variance-covariance peut souvent être trompeur à cause des différences d'échelle entre les variables. C'est pourquoi, la majorité du temps, l'ACP sera calculée non pas sur la matrice de variance-covariance, mais bien sur la matrice de corrélation. En fait, à moins d'avoir d'excellentes raisons, il faudrait toujours calculer l'ACP sur la matrice de corrélation.

Soyez prudent cependant, la plupart des logiciels calculent l'ACP par défaut sur la matrice de variance-covariance et c'est à vous d'en choisir autrement.

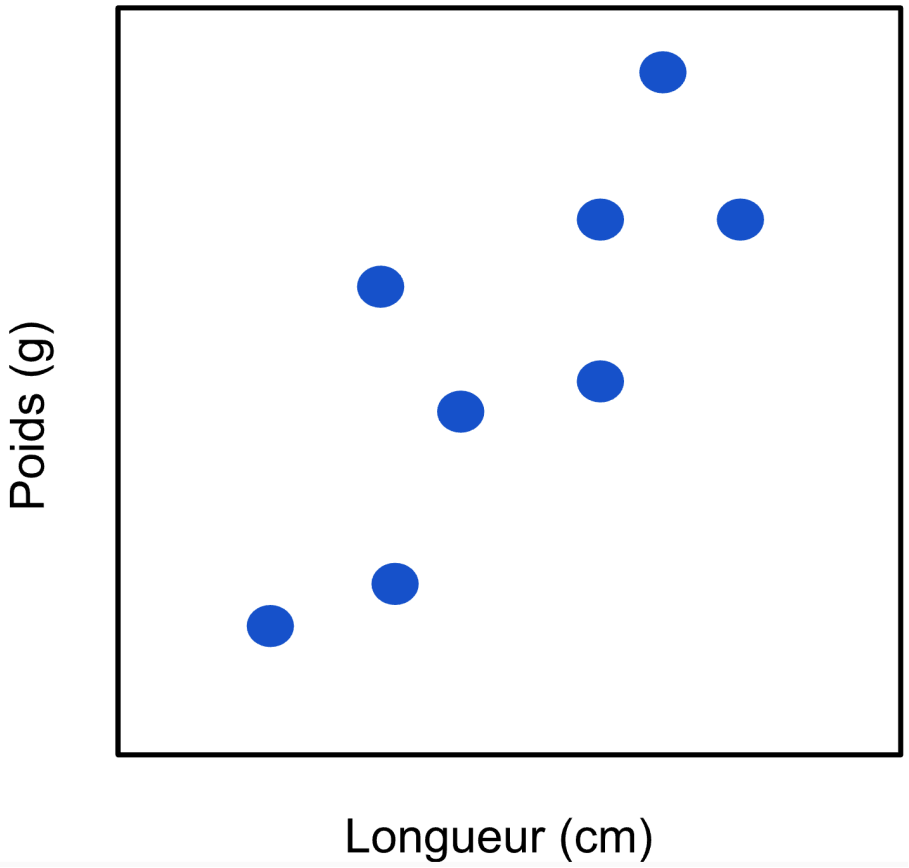
Enfin, pour terminer cette section sur la technicalités, sachez qu'il existe deux techniques mathématiques pour calculer l'ACP à partir d'une matrice de données. On peut soit effectuer une décomposition spectrale ou soit une décomposition en valeurs singulières (SVD).

23.3. Intuition visuelle

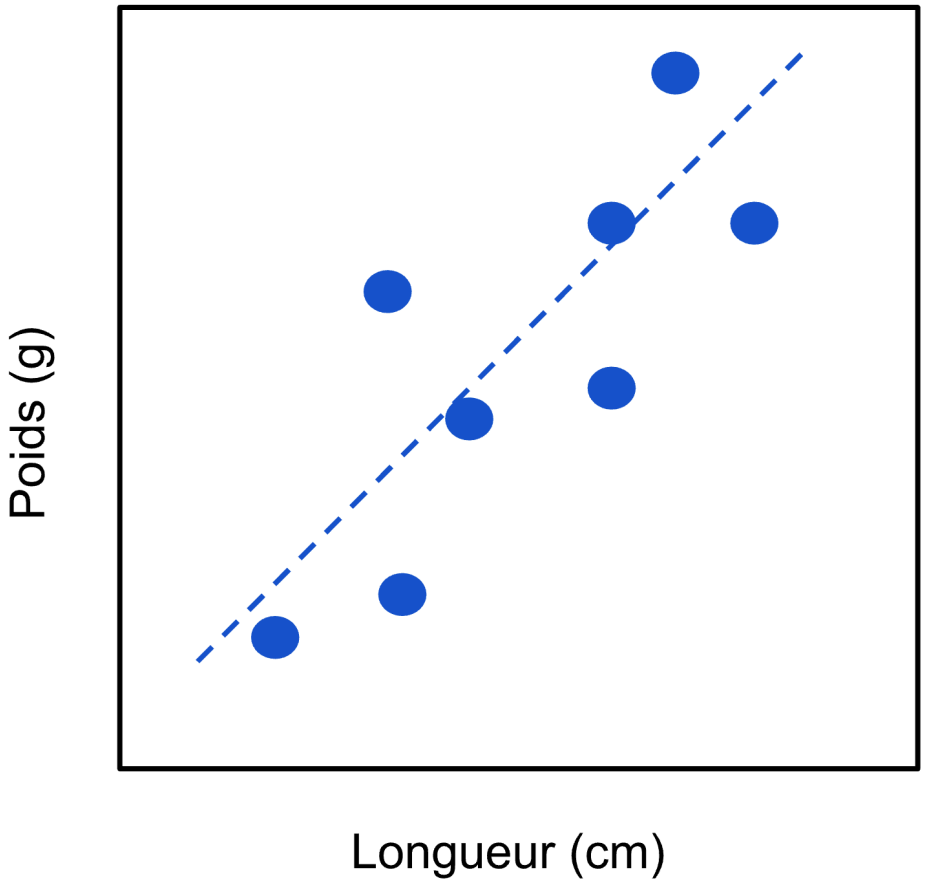
Pour les esprits plus mathématiques parmi vous, la section précédente a peut-être suffit à saisir l'ACP. Pour le commun des mortels par contre, l'ACP est habituellement plus facile à comprendre visuellement que mathématiquement.

Pour des fins d'illustration, je travaillerai comme si notre matrice de données originale ne contenait que deux variables. Remarquez que dans la vraie vie, si vous ne travaillez qu'avec deux variables, il n'est probablement pas pertinent de faire une ACP...

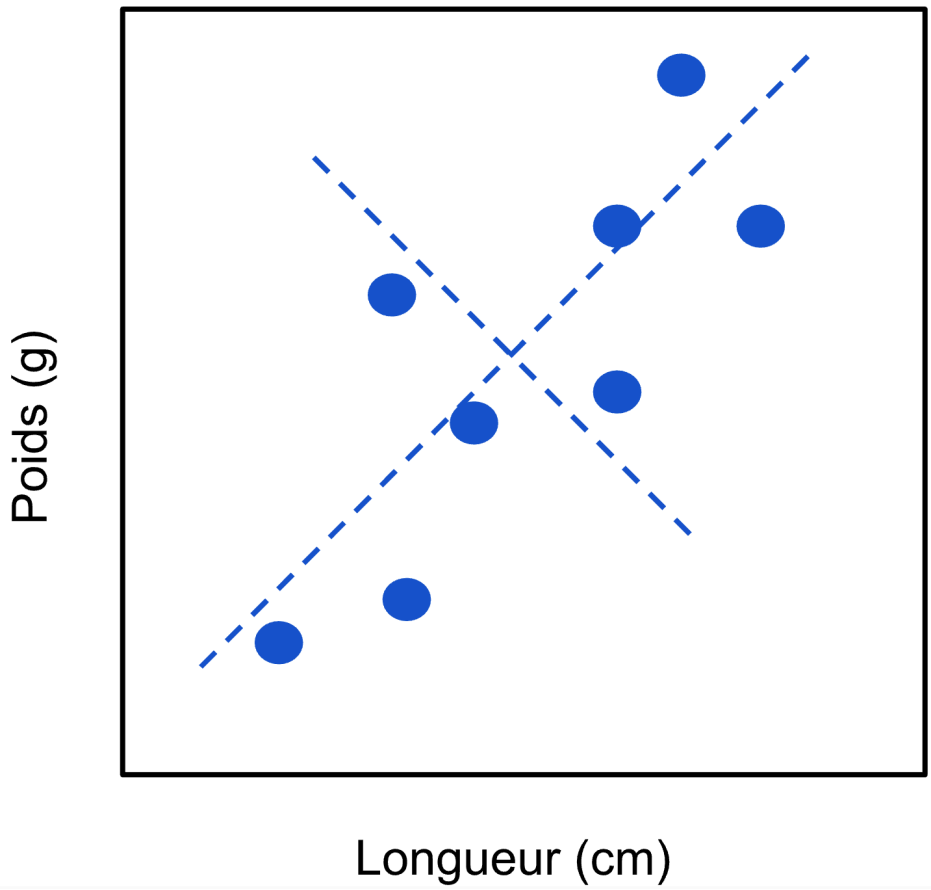
Donc voilà, disons que nous avons mesuré sur une dizaine de poissons le poids et la longueur, et que nous avons obtenu les résultats suivants :



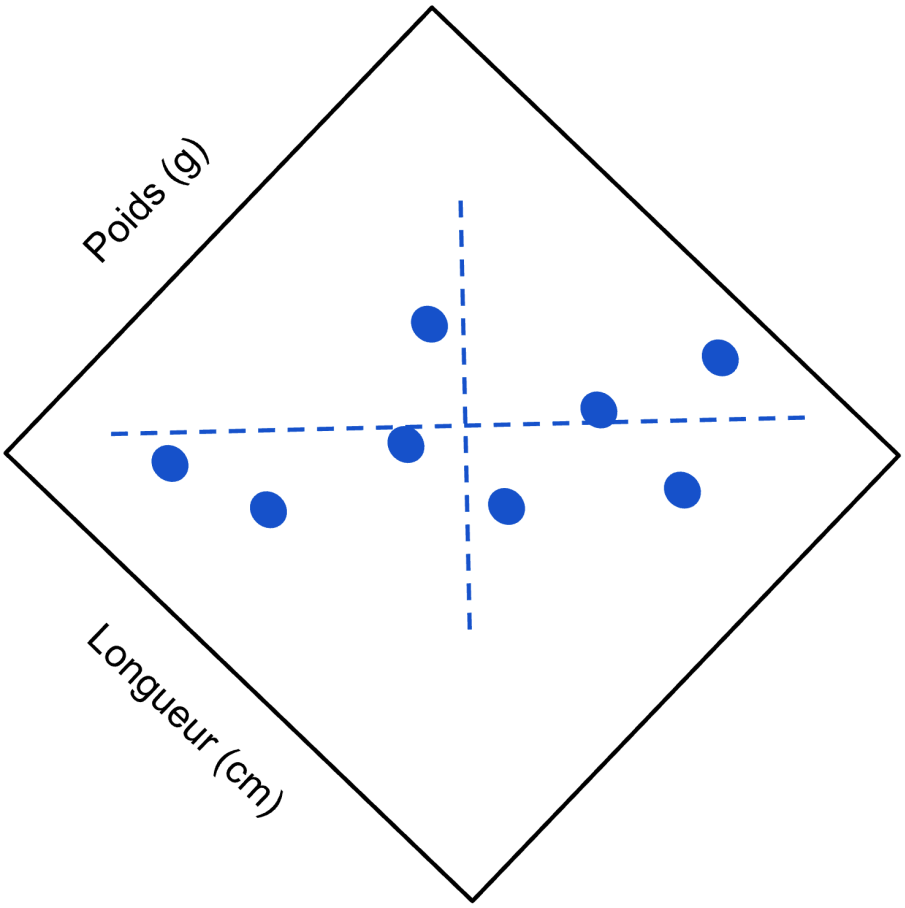
L'ACP tenterait d'abord de trouver dans ce nuage de points la façon de passer un nouvel axe, qui permettrait d'expliquer le plus de variabilité (i.e. de séparer les points le plus possible). Dans nos données, cet axe passerait probablement quelque part ici :

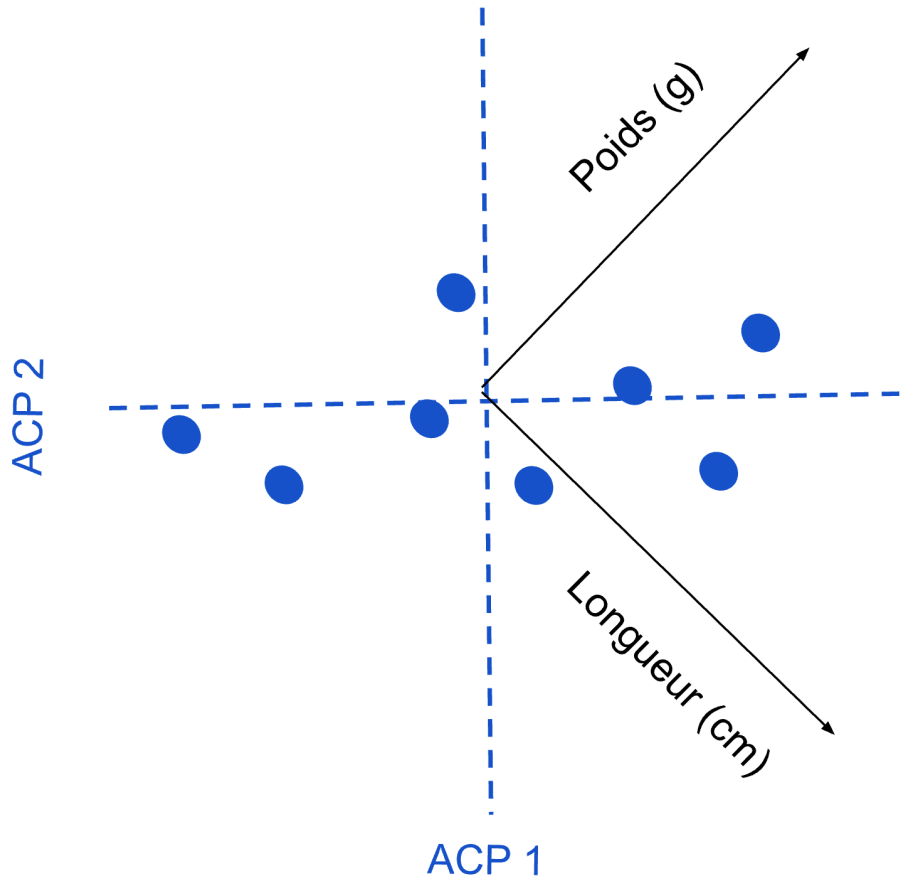


Une fois cet axe trouvé, l'ACP cherchait ensuite où passer un deuxième axe (puisque nos données originales contenaient deux variables), qui soit orthogonal au premier. Puisque nous sommes en deux dimensions, il suffit de placer cet axe à 90° du premier :



Ensuite, l'ACP fait pivoter notre nuage de points, de façon à ce que l'axe 1 de l'ACP soit bien horizontal et l'axe 2 bien vertical, comme ceci :

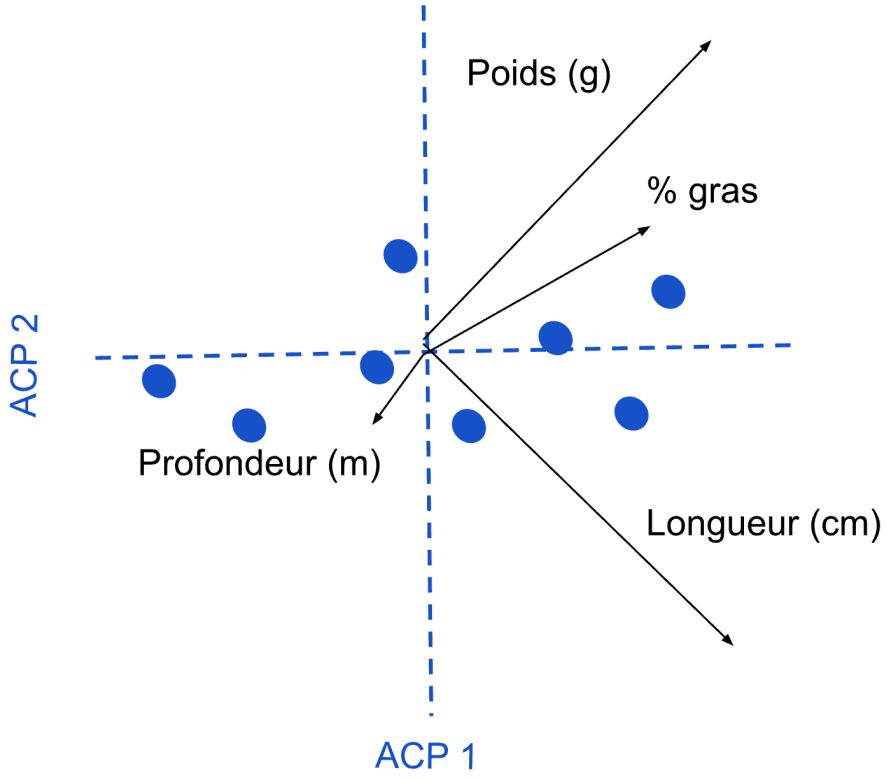

Poids (g)
Longueur (cm)



Dans ce nouveau système d'axes, défini par les composantes principales, on peut maintenant bien voir l'effet de simplification. Alors qu'au départ on avait besoin de deux variables pour définir la position d'une observation (longueur et poids), on peut voir que dans le graphique final, on pourrait essentiellement fournir uniquement la position dans l'axe 1 et que l'on perdrait très peu d'information. Remarquez que l'on peut visualiser les axes originaux (poids et longueur), pour mieux comprendre la composition des nouveaux axes. C'est la version visuelle des eigenvec-

tors, la recette pour composer les nouveaux axes.

À ce point, ce qu'il importe de bien saisir est que contrairement à moi sur la page, l'ACP peut travailler en autant de dimensions que nécessaire. Si nous avions trois variables, elle pivoterait le nuage de points aussi sur l'axe des Z, en trois dimensions. Si nous avions 10 variables, l'analyse n'aurait aucun problème à pivoter un nuage de points en 10 dimensions. Par contre, lorsque nous tentons de visualiser les résultats avec notre œil d'humain, nous ne pouvons regarder que 2 axes de l'ACP à la fois. Si on observe par exemple une ACP faite sur 4 variables comme ceci :



Il importe de bien comprendre que si la flèche de la variable profondeur semble courte, c'est qu'elle n'est pas très associée aux axes 1 et 2. Il faut s'imaginer qu'elle entre dans la feuille, et est probablement associée aux axes 3 ou 4 de l'analyse. De la même façon, si la flèche de la variable % gras semble plus courte, c'est qu'elle est moins associée à l'axe 1 que celles de longueur et de poids. Si on avait illustré les axes 3 et 4 de l'analyse, les résultats auraient été différents.

23.4. Mais où est la simplification?

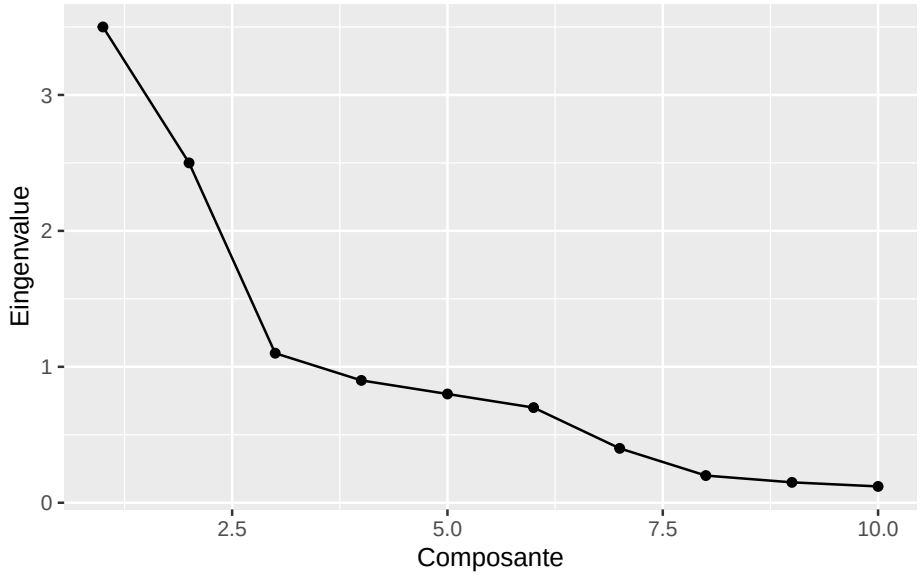
Si vous avez réussi à suivre jusqu'ici, vous vous demandez probablement où est l'aspect de simplification dans tout cela? Si la matrice des composantes principales contient autant de variables que la matrice originale, quel est l'intérêt?

C'est ici que l'aspect subjectif d'interprétation d'une ACP entre en ligne de compte. Comme nous avons discuté, l'ACP concentre la variabilité dans les premiers axes. Il n'en reste donc que très peu dans les derniers axes. Aussi peu que, souvent, on peut simplement ignorer les derniers axes au moment de l'interprétation.

Mais combien de composantes peut-on ignorer? Il n'existe pas une seule bonne réponse parfaite à cette question. La meilleure stratégie est d'utiliser votre jugement. Aucune technique statistique ne peut vous dire ce qui est important ou non. Cependant, il existe quelques stratégies pour guider votre choix.

La première stratégie, probablement la plus simple, consiste à ne conserver pour interprétation que les axes ayant un eigenvalue > 1. Évidemment, pour appliquer cette règle, l'ACP devra avoir été calculée sur la matrice de corrélation, sinon les valeurs d'eigenvalues sont complètement arbitraires.

La deuxième stratégie consiste à tracer et interpréter un diagramme en éboulis (scree plot). Ce dernier est relativement simple à tracer : vous mettez en X le numéro de chacune des composantes principales, et en Y, l'eigenvalue de chacun des axes :



Avec un peu de chance, vous réussirez à trouver dans ce graphique un coude, c'est-à-dire un changement de pente, où cette dernière passe de plus abrupte à plus douce. La stratégie consiste à conserver jusqu'au point d'inflexion, inclusivement. Dans le graphique précédent, on aurait par exemple conservé les trois premiers axes.

La dernière stratégie consiste à établir un seuil de % d'explication au-delà duquel on arrête d'interpréter les axes. Pour utiliser cette technique, il faut se préparer un petit tableau, dans lequel on calcule le % cumulatif d'explication donné par l'eigenvalue de chacun des axes. Par exemple, pour notre exemple sur les longueurs de poissons, il ressemblerait à ceci :

Axe	Eigenvalue	% expliqué	% cumulatif
ACP1	1317,8	93,86	93,86
ACP2	86,1	6,13	99,99
ACP3	0,1	0,01	100

Si on avait établi que l'on voulait garder tous les axes jusqu'à 90 % d'explication, on aurait conservé que l'axe 1 seulement, alors que si on avait établi notre seuil à 95 %, on aurait plutôt interprété les deux premiers axes.

23.5. Comment faire l'interprétation?

Il existe deux façons de présenter les résultats de l'ACP et d'en faire l'interprétation. On peut soit regarder le tableau des eigenvectors ou soit regarder le graphique d'ordination (**biplot**). Les deux stratégies nous apportent exactement la même information.

Nous analyserons les données présentées au Chapitre 22, où nous avions capturé 5 poissons, sur lesquels nous avions mesuré la longueur, le poids et noté la profondeur de la capture. Nous avons appliqué une ACP basée sur la matrice de corrélation, puisque nos mesures étaient à des échelles différentes. Observons d'abord le tableau des eigenvalues, afin de se donner une idée d'où arrêter notre interprétation, en se basant sur le fait que l'on veut interpréter tous les axes jusqu'à 95 % de la variance :

	Axe 1	Axe 2	Axe 3
Eigenvalue	2,51	0,45	0,05
% expliqué	84	15	1
% cumulatif	84	99	10

Dans ce cas-ci, nous interpréterons donc les deux premiers axes. Remarquez que ce tableau va de gauche à droite plutôt que de haut en bas comme le précédent, mais il contient exactement la même information.

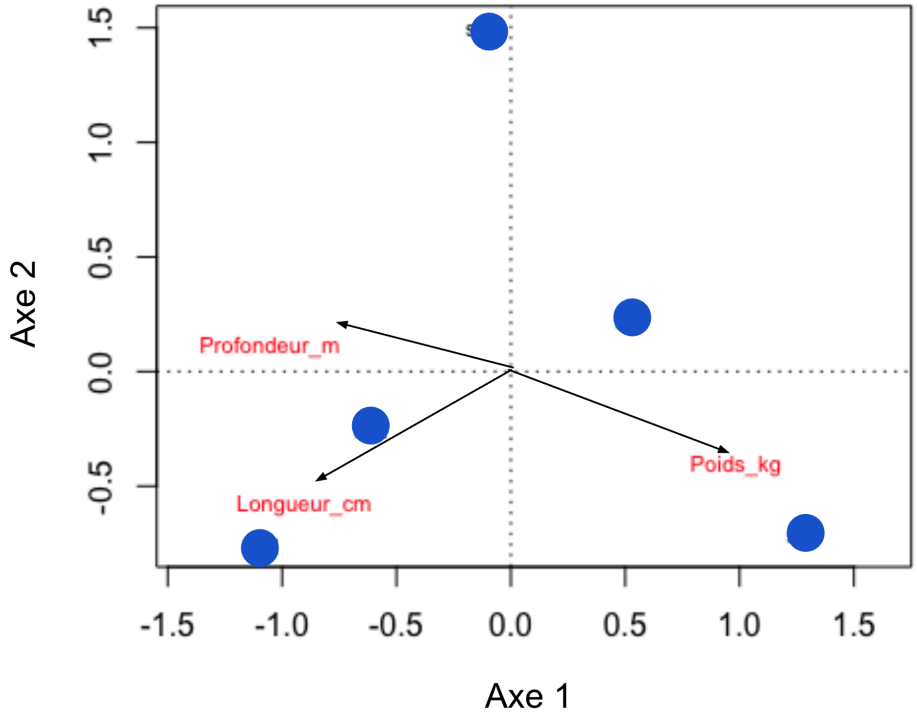
Maintenant, observons le tableau des eigenvectors :

	Axe 1	Axe 2	Axe 3
Longueur (cm)	-0,90	-0,58	0,06
Poids (kg)	0,98	-0,41	-0,13
Profondeur (m)	-1,05	0,11	-0,18

Première chose à remarquer : dans ce tableau, on ne parle PAS de fraction expliquée contrairement au tableau précédent. Il s'agit de coefficients, un peu comme les paramètres d'une régression, qui peuvent donc avoir des valeurs négatives. Chacun des nombres nous indique à quel point une des variables originale (par exemple longueur) est associée à une composante principale (par exemple l'axe 1). On voit, entre autres, que l'axe 1 est fortement associé aux trois variables (en valeur absolue). L'axe 2 quant à lui est plus associé (en valeur absolue) à la longueur et au poids, et moins à la profondeur.

Pour le sens du coefficient (positif ou négatif), il n'a par lui-même pas d'interprétation. On aurait pu inverser l'ensemble des signes (i.e. que chaque négatif soit un positif et chaque positif soit un négatif) et avoir exactement la même interprétation. Ce qu'il est surtout important d'observer est si des variables sont du même signe ou non sur un axe. Si elles ont le même signe, elles sont corrélées (ou associées) positivement. Si elles ont des signes différents, elles sont associés négativement (i.e. quand l'une augmente, l'autre diminue). Par exemple, pour l'axe 1, la profondeur et la longueur sont associées positivement (elles ont le même signe), alors que ces deux variables sont associées négativement au poids (qui a un signe différent).

Voyons maintenant comment ces mêmes résultats se traduisent de manière visuelle :



On y constate la même information, soit que la variable poids est seule de son côté de l'axe 1 (à droite) et les variables longueur et profondeur sont associées ensemble de l'autre côté (à gauche). On voit aussi que sur le deuxième axe, poids et longueur sont associées d'un côté (en bas) et profondeur de l'autre (en haut).

Notez que pour entrer à la fois les variables et les observations dans un même graphique, un cadrage (mise à l'échelle) doit être effectué. Cette opération peut être effectuée de plusieurs façons, entre autres :

- Type I : La distance relative entre les observations est respectée,

mais les angles entre les variables ne doivent PAS être interprétés
- Type II : La distance relative entre les observations ne peut PAS être interprétée, mais les angles entre les variables représentent leur corrélation.

Donc, si l'intérêt est de comparer les observations, il faut utiliser le cadrage de type I, si l'intérêt est les variables (flèches), utiliser le type II. Par défaut, la majorité des fonctions d'ACP utilisent le cadrage de type II, permettant d'interpréter les variables.

Et maintenant la partie la plus difficile, donner un sens aux axes de l'ACP. Notez d'abord qu'il n'y a aucun test statistique qui peut vous venir en aide ici. Seul votre jugement de biologiste et votre connaissance du système à l'étude pourrons vous aider.

Donc, que représentent les axes? L'axe 1 n'est pas simple à interpréter. Il semble nous indiquer que plus les poissons sont capturés en profondeur, plus ces derniers sont longs, mais minces. Si ce n'était pas des données fictives, nous aurions probablement pu conclure qu'il s'agit d'une sorte de compromis fonctionnel, où il est plus avantageux d'être mince que gros en profondeur. Remarquez cependant que l'ACP ne fournit pas de lien de cause à effet. On ne sait pas si la profondeur cause la longueur ou l'inverse. L'axe 2 est plus facilement interprétable. Puisque le poids et la longueur y sont associées positivement et que la profondeur y est moins associée, on peut probablement y voir un gradient de taille de poissons. D'un côté de l'axe (en bas), les poissons sont longs et lourds, de l'autre côté, petits et légers.

On peut donc dire que les deux gradients principaux dans notre jeu de données étaient premièrement un compromis fonctionnel selon la profondeur et un deuxièmement un gradient de taille. On pourrait aussi affirmer qu'à eux seuls, ces deux gradients expliquent 99 % de la variabilité de notre jeu de données.

Dans la vraie vie, il est rarement possible de donner un sens aux axes

après le 2e ou le 3e. Ils deviennent trop abstraits pour notre pauvre cerveau humain!

23.6. Les assomptions de l'ACP

Comme toutes les techniques statistiques, l'ACP comporte certaines assomptions. La première chose à savoir est que l'ACP cherche des relations linéaires entre les variables. Donc, elle sera beaucoup plus efficace pour résumer vos données si vous prenez le temps de linéariser les relations entre vos variables à l'aide des transformations appropriées (voir Chapitre 9). Aussi, comme pour toutes les analyses, la présence de données aberrantes peut influencer le résultat de vos analyses. Ces dernières doivent donc être inspectées et gérées adéquatement.

Pour une utilisation descriptive de l'ACP comme nous le faisons dans ce cours, l'ACP n'a pas d'assomption au niveau de la distribution des variables. Cependant, si un jour vous voulez appliquer des tests statistiques sur l'ACP afin de savoir quels axes sont significatifs, etc. sachez qu'à ce moment, chacune de vos variables doit aussi suivre une distribution normale.

23.7. Labo : L'analyse en composantes principales

Pour le laboratoire de ce chapitre, nous regarderons comment s'organisent les différentes mesures morphométriques des manchots de l'archipel de Palmer.

Sachez d'abord que pour appliquer l'ACP dans R, il existe une panoplie de fonctions différentes. En général, elles arrivent toutes exactement au même résultat. Si vous n'utilisez que R de base, la fonction `prcomp` est

tout à fait appropriée pour calculer des ACP. Cependant, comme les analyses des deux prochains chapitres nécessitent l'utilisation de la librairie **vegan**, je vous montrerai comment calculer vos ACP aussi à l'aide de cette librairie, afin que la façon de faire soit constante entre les différentes techniques.

La première chose à faire sera donc d'activer les librairies nécessaires :

```
library(vegan)
```

```
Loading required package: permute
```

```
Loading required package: lattice
```

```
This is vegan 2.6-8
```

```
library(tidyverse)
```

Ensuite, nous nous préparerons un tableau de données ne contenant que les variables quantitives, duquel nous retirerons les lignes contenant les valeurs manquantes.

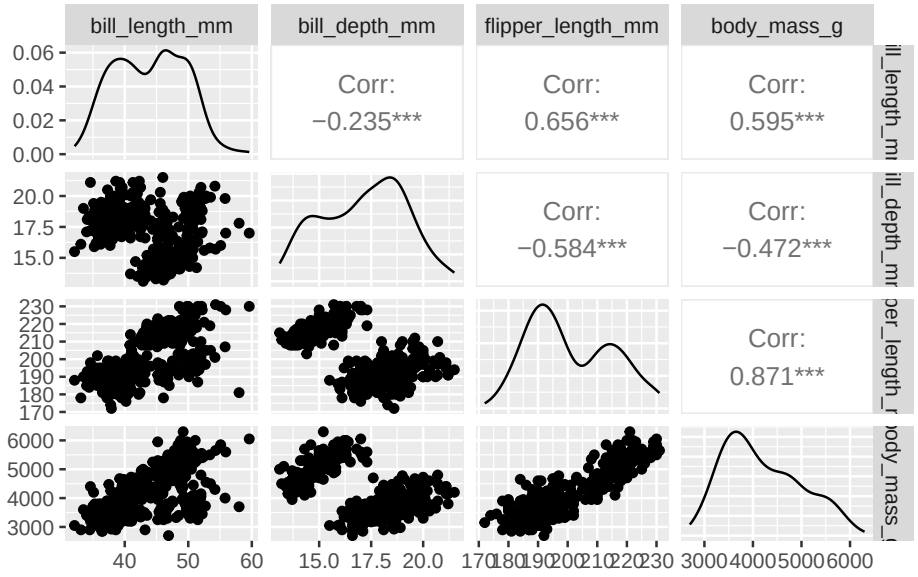
Nous conserverons dans un autre tableau le reste des informations pour pouvoir colorer nos points dans nos graphiques, par exemple selon l'espèce.

> ⚠ Avertissement
>
> Attention, il est très important d'extraire le reste des informations après avoir éliminé les valeurs manquantes, sinon les lignes ne correspondront pas entre nos deux tableaux.

```
library(palmerpenguins)

pour_acp <-
  penguins |>
  drop_na(bill_length_mm:body_mass_g)

infos_complementaires <-
  pour_acp |>
  select(species, island, sex, year)

pour_acp <-
  pour_acp |>
  select(bill_length_mm:body_mass_g)
```

Ensuite, une façon rapide d'explorer nos données avant de commencer une analyse avec plusieurs variables et d'utiliser la fonction `ggpairs` de la librairie `GGally`. Cette dernière nous permet de voir en une seule commande l'histogramme de chacune de nos variables et le nuage de points illustrant la forme de la relation entre chacune de nos variables.

```
library(GGally)
ggpairs(pour_acp)
```

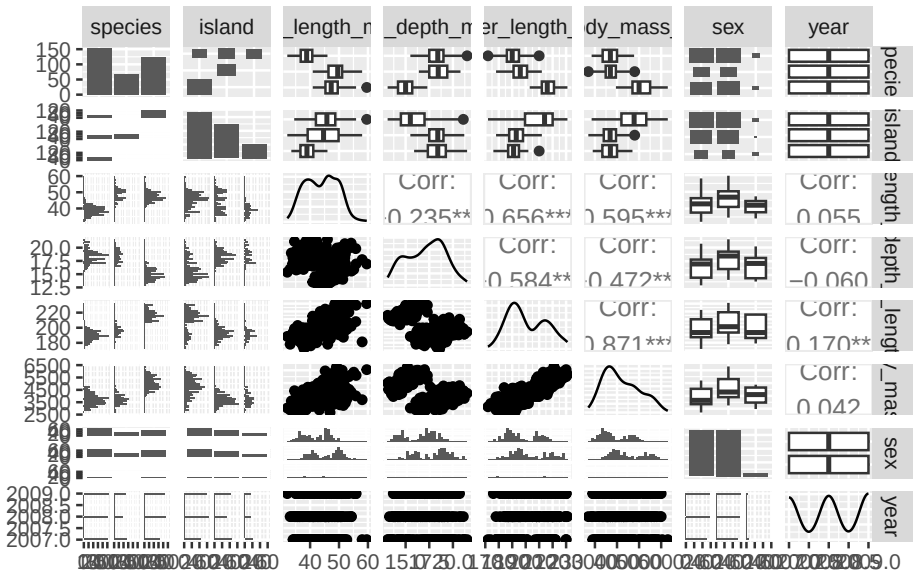


Remarquez que les histogrammes sur la diagonale sont lissés plutôt que d'afficher chacune des bandes, mais l'interprétation demeure la même.

En général, tout cela est tout à fait OK pour l'ACP. Les distributions sont relativement normales et les relations sont relativement linéaires. Ou au moins, on ne voit pas de relations clairement non-linéaires (p. ex. exponentielle).

Ce genre de graphiques présentant l'ensemble des distributions et des relations est extrêmement pratique. Je vous conseille, au début de chaque projet d'analyse, de lancer la commande `ggpairs` et d'imprimer le résultat pour conserver à portée de main ce graphique, pour pouvoir y référer et en discuter avec vos collègues. On aurait même pu préparer ce graphique pour notre tableau de données original, incluant les variables qualitatives :

```
ggpairs(penguins)
```



Pour calculer l'ACP à l'aide de la librairie vegan, vous devez utiliser une fonction nommée `pca` (*Principal Component Analysis*).

Enfin, comme nos données sont à des échelles différentes, il est important de spécifier à la fonction de travailler sur la matrice de corrélation, avec l'argument `scale = TRUE`. Vous remarquerez enfin que plutôt que d'afficher directement le résultat du calcul, je le conserve dans un objet que je nomme acp, puisque nous aurons plusieurs petites choses à faire avec :

```
acp <- pca(pour_acp, scale = TRUE)
```

La première chose que l'on peut regarder ensuite est les eigenvectors de votre ACP :

```
summary(acp)
```

```
Call:
pca(X = pour_acp, scale = TRUE)

Partitioning of correlations:
              Inertia Proportion
Total               4          1
Unconstrained       4          1

Eigenvalues, and their contribution to the correlations

Importance of components:
                          PC1    PC2     PC3     PC4
Eigenvalue             2.7538 0.7725 0.36524 0.10849
Proportion Explained   0.6884 0.1931 0.09131 0.02712
Cumulative Proportion  0.6884 0.8816 0.97288 1.00000
```

On peut y constater que le premier axe de l'ACP explique à lui seul 68% de la variation dans nos données (*Proportion explained*), et qu'avec trois axes, on dépasse déjà 95% d'explication (*Cumulative proportion*).

On peut ensuite aller observer les eigenvectors. Autrement dit, la contribution de chacune de nos variables originales à la construction des nouveaux axes de l'ACP.

Ici, on ajoute la mention `1:4` pour obtenir la construction de tous les axes (nous avions 4 varibales). Mais on aurait pu aussi en demander moins, par exemple avec `1:2`.

```
scores(acp, choices = 1:4)$species
```

```
                      PC1          PC2         PC3
bill_length_mm     2.295549 -1.594500048 -1.1831757
bill_depth_mm     -2.018643 -2.130607173  0.7683875
flipper_length_mm  2.904483 -0.006095108  0.4261922
body_mass_g        2.764994 -0.225309318  1.0955789
                       PC4
bill_length_mm     0.1456481
bill_depth_mm     -0.1681303
flipper_length_mm -0.7844720
body_mass_g        0.5803802
```

Remarquez qu'il est un peu étrange d'aller chercher l'item `species`. Ce qu'il faut comprendre à propos des noms étranges dans `vegan` est que la librairie a été conçue à l'origine pour analyser des données où chaque colonne était une espèce et chaque ligne était un site. Chaque fois qu'on lit les sorties de `vegan`, il faut transposer les termes pour les ramener à notre jeu de données actuel. C'est emmerdant, mais au-delà de ces problèmes de noms, les qualités techniques de `vegan` en ont fait le standard pour les ordinations en écologie, et les choses ne semblent pas près de changer.

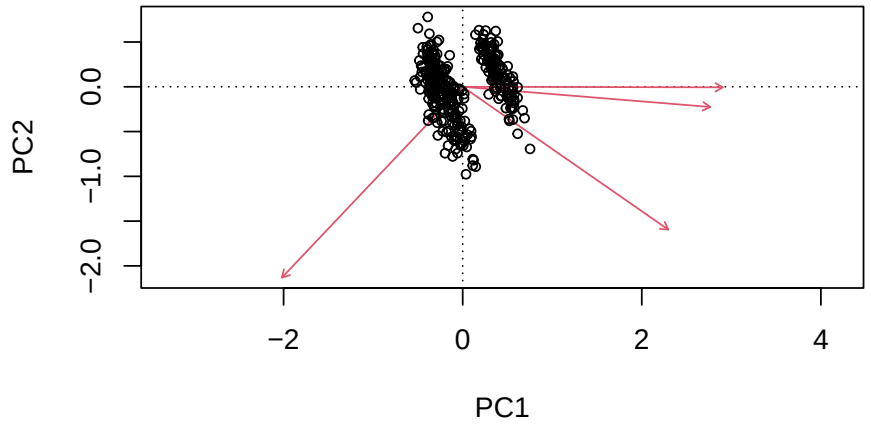
Donc, le tableau des eigenvectors (nommé *Species scores*) nous informe que les deux variables les plus associées à l'axe 1 (en valeur absolue) sont la longueur des ailes et le poids du corps, tous deux du même côté de l'axe. Les variables associées au bec sont aussi associées à cet axe, mais moins fortement.

L'axe 2 quant à lui est surtout formé des variables de longueur et épaisseur du bec, associées toutes les deux du même côté de l'axe.

On pourrait donc interpréter le premier axe comme un axe de taille du corps (des oiseaux lourds avec des grandes ailes d'un côté et petits de l'autre) et le deuxième comme un axe de taille de bec (les oiseaux avec un grand bec épais d'un côté, les petits becs de l'autre).

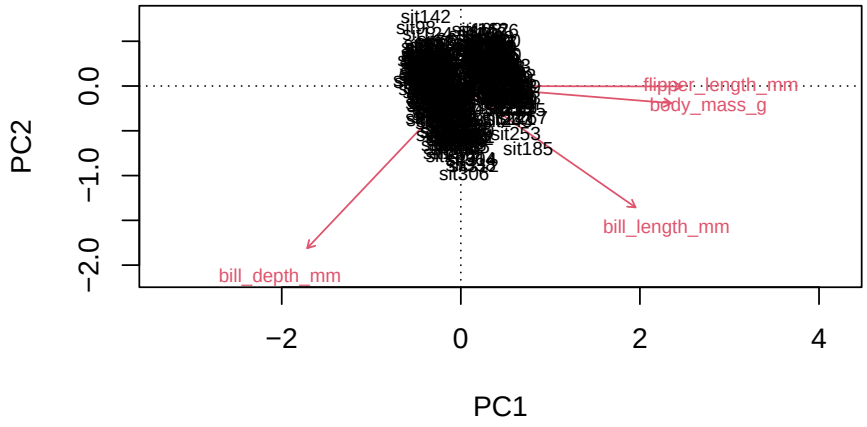
Si l'on veut baser (ou valider) nos interprétation sur les graphiques, on peut utiliser à cette fin la fonction `biplot`, à laquelle on passe notre objet de résultats, comme ceci :

```
biplot(acp)
```



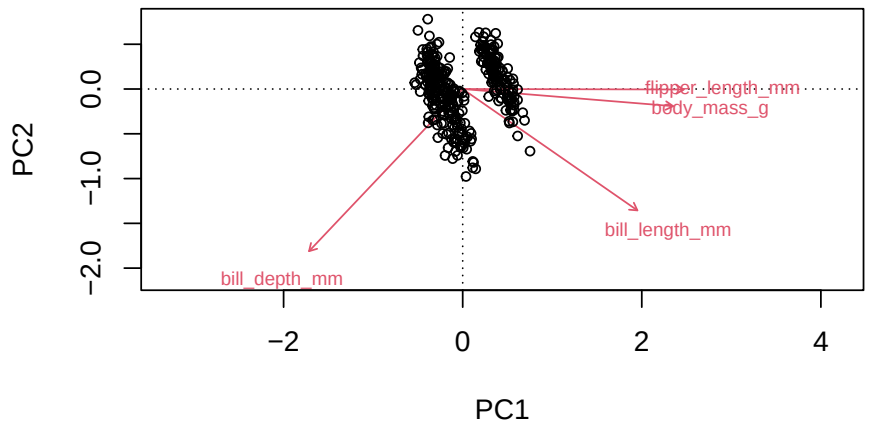
Malheureusement, ce premier graphique n'est pas très utile. Lorsque notre tableau de données contient beaucoup d'observations, la fonction `biplot` de vegan préfère cacher les étiquettes pour ne pas alourdir le graphique. Pour récupérer les étiquettes dans ces cas-là, il faut spécifier manuellement à la fonction que nous voulons les étiquettes :

```
biplot(acp, type = c("text","text"))
```



Comme nous n'avons pas associé de noms aux lignes du tableau de données, il pourrait être préférable d'afficher le nom seulement des variables :

```
biplot(acp, type = c("text","point"))
```



On constate donc dans ce graphique exactement la même chose :

- Un premier axe (gauche à droite) décrivant la grosseur des oiseaux.
- Un deuxième axe (de haut en bas) décrivant la grosseur des becs. Remarquez que sur cet axe, les grandes valeurs sont vers le bas.

La conclusion de notre analyse est donc que les 4 variables décrivants nos manchots peuvent être essentiellement remplacées par 2 variables seulement : une décrivant la grosseur de l'oiseaux et l'autre la grosseur du bec. Si on utilise ces deux axes plutôt que les 4 variables originales, on perd à peine 12% d'information (100%-88%).

Une façon de bien comprendre un graphique d'ACP est de l'analyser quadrant par quadrant :

- En haut à gauche, on a des petits oiseaux avec des petits becs
- En haut à droite, de grands oiseaux avec des petits becs
- En bas à gauche de petits oiseaux avec de grands becs et

- En bas à droite, de grands oiseaux avec de grands becs.

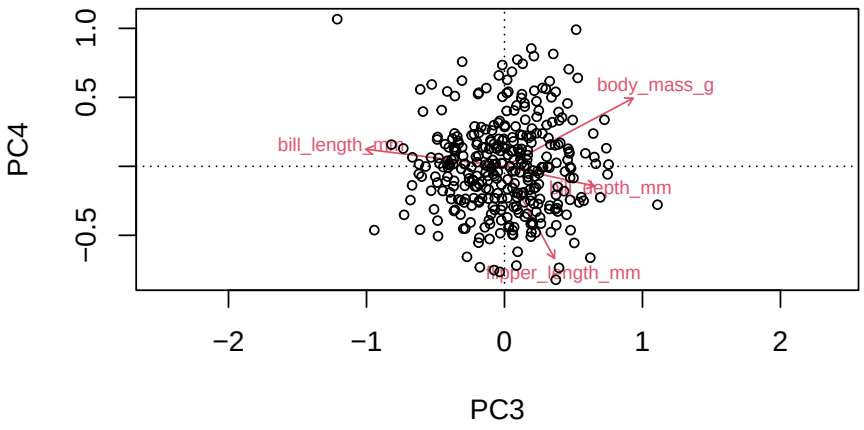
Remarquez que les variables associées au bec sont presque parfaitement en diagonale entre les deux premiers axes. Si on rapporte par exemple la position de la pointe de la flèche de bill_length_mm sur l'axe horizontal, elle arrive à peine derrière les deux autres variables. Il aurait donc aussi été légitime de décrire l'axe 1 comme :

Oiseaux lourds, avec de grandes ailes et de longs becs minces à droite et oiseaux légers, avec de petites ailes, et des becs courts et épais à gauche.

L'interprétation de l'ACP n'est pas tranchée au couteau. Il faut utiliser votre connaissances du système à l'étude et votre jugement pour faire le meilleur usage des résultats de l'analyse.

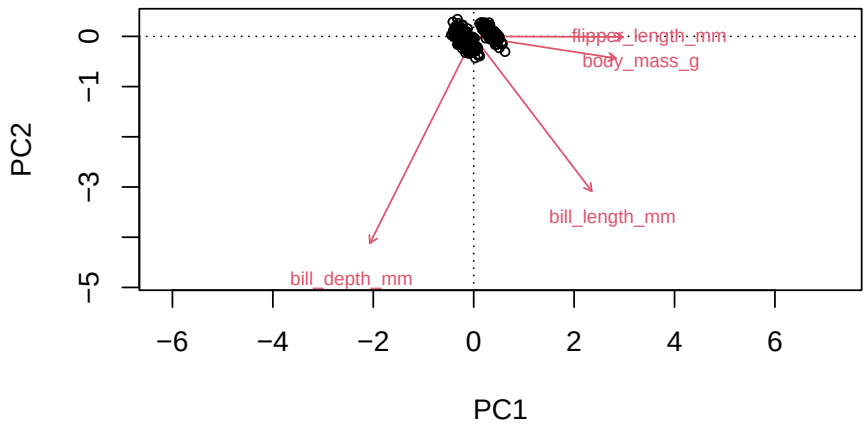
Remarquez qu'avec la fonction `biplot`, on peut aussi choisir de voir d'autres axes que les deux premiers. On pourrait par exemple regarder comme ceci le troisième avec le quatrième :

```
biplot(acp, choices = c(3,4), type = c("text","point"))
```



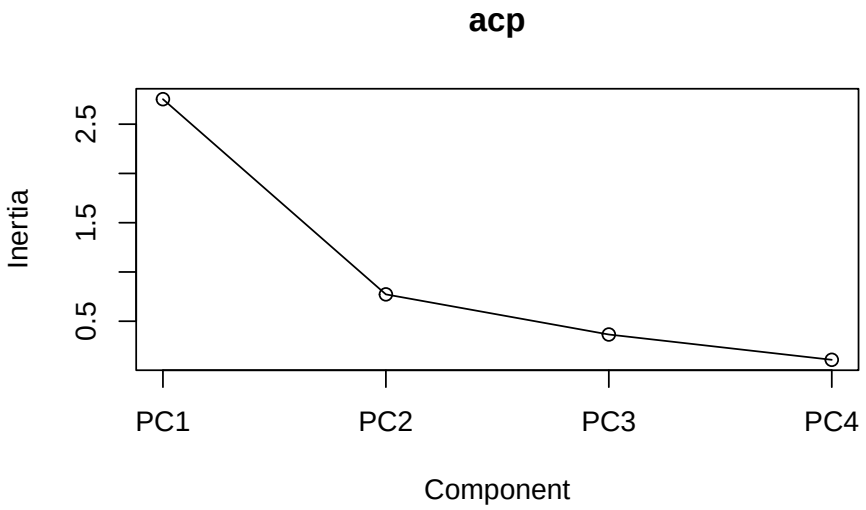
Par défaut, la fonction **biplot** utilise le cadrage de type II, permettant d'interpréter les variables. Pour utiliser le cadrage de type I, il faut ajouter l'argument **scaling = "sites"**, comme ceci :

```
biplot(acp,scaling = "sites", type = c("text","point"))
```



Nous avons vu dans les étapes précédentes comment retrouver chacune des données nécessaires pour utiliser les règles permettant de nous aider à savoir combien d'axes interpréter dans une ACP (*Proportion explained*, *Cumulative proportion*, etc). Il existe aussi une fonction nommée `screeplot`, permettant de visualiser facilement le diagramme en éboulis :

```
screeplot(acp, type = "lines")
```



Basé sur ce graphique, le point d'inflexion semble survenir au 2e axe. J'aurais donc probablement interprété deux axes.

23.8. Exercice : L'analyse en composantes principales

Comme exercice sur l'ACP, nous travaillerons sur une petite base de données que j'ai construite à partir de données d'environnement Canada concernant la météo moyenne de certaines villes du Canada et d'autres partout dans le monde[1].

Après avoir chargé le fichier, assurez-vous d'éliminer la colonne Neige.cm. Puisqu'elle contient beaucoup d'observations manquantes, nous ne l'utiliserons pas dans cet exercice.

[1] https://drive.google.com/file/d/1ZLeRkJl2MmJNFZjEgJljul9tUUIJqsyr/view?usp=sharing

Notre base de données contiendra donc pour chaque ville 6 variables :

- Precip.mm : La quantité de précipitations dans une année
- Jours.Precip : Le nombre de jours dans une année où il y a au moins une goutte de précipitations
- T.Max.Moy : La moyenne des températures quotidiennes
- T.Moy.Mois.Froid : La température moyenne du mois le plus froid
- T.Moy.Mois.Chaud : La température moyenne du mois le plus chaud

Pour cet exercice, assumez que j'ai déjà fait la vérification pour vous, et que les données sont suffisamment normales et les relations suffisamment linéaires pour donner de bons résultats avec l'ACP.

Donc, à partir des ces données :

1) Calculez une ACP à partir de la matrice de corrélation,
2) À partir des eigenvectors, déterminez quelles variables sont les plus associées à l'axe 1 et à l'axe 2,
3) Visualisez les deux premiers axes de cette ordination à l'aide d'un graphique,
4) Déterminez avec l'aide d'un diagramme en éboulis (screeplot) combien d'axes pourraient être interprétables dans cette analyse.
5) Comment ce résultat se compare-t-il avec le nombre d'axes qui auraient été retenus si on avait interprété tous les axes ayant un eigenvalue > 1.
6) Quelle interprétation donnez-vous aux axes 1 et 2 de l'ACP. Que représentent-ils, dans vos propres mots?

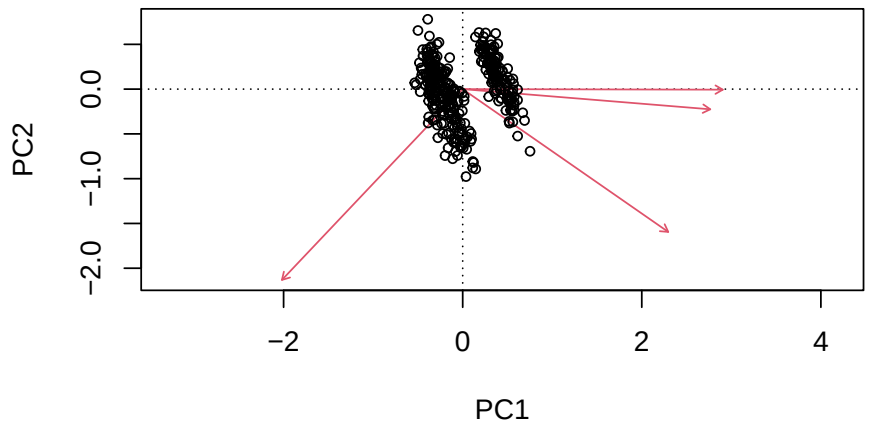
23.9. Contenu optionnel : Personnaliser un graphique d'ACP avec ggplot2

Il existe quelques options permettant de personnaliser les graphiques de la fonction `biplot` de `vegan`, mais les options sont relativement li-

mitées. Si jamais vous voulez personnaliser vos graphiques d'ACP entièrement à votre goût, il est aussi possible de le recréer avec `ggplot2`. Nous aurons besoin, pour ce faire, d'extraire les coordonnées de nos observations et de nos variables originales dans le nouveau système d'axes de l'ACP.

Ce travail se fait en 3 étapes. D'abord, on doit refaire la fonction `biplot`, mais récupérer le résultat dans un objet. Ici on l'appellera x pour la simplicité.

```
x <- biplot(acp)
```



On extrait ensuite de cet objet les nouvelles coordonnées des variables :

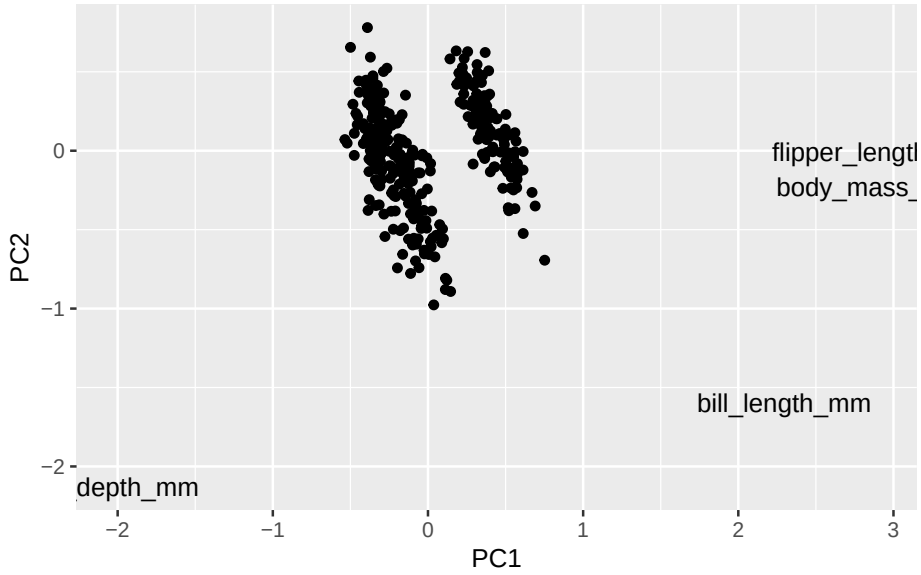
```
variables <- x$species |>
  as.data.frame() |>
  rownames_to_column("variable")
```

Et puis celles des observations :

```
observations <- x$sites |>
  as.data.frame()
```

Une fois ces étapes effectuées, ne reste plus qu'à les utiliser avec **ggplot2** pour construire notre graphique d'ACP à notre goût. La clé pour y arriver est de savoir que l'argument **data** est aussi disponible dans une couche graphique, pour qu'elle utilise des données provenant d'un second tableau. Voici le graphique le plus simple que l'on aurait pu faire :

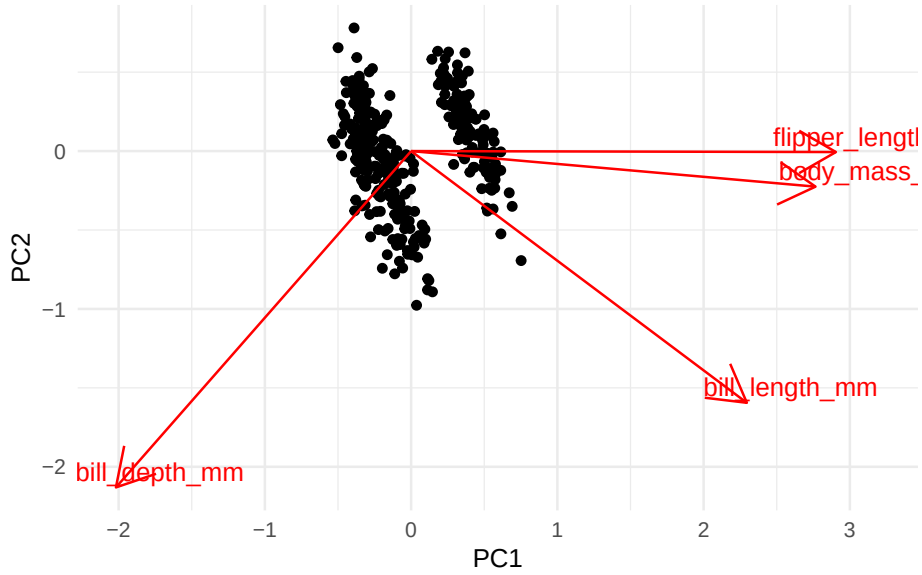
```
observations |>
  ggplot(aes(x = PC1, y = PC2)) +
  geom_point() +
  geom_text(
    data = variables,
    aes(label = variable)
  )
```



Avec légèrement plus de travail, il est aussi possible d'ajouter les flèches. Remarquez que, pour cet exemple, je décale aussi légèrement les étiquettes pour ne pas qu'elles se superposent avec les pointes des flèches :

```
observations |>
  ggplot(aes(x = PC1, y = PC2)) +
  geom_point() +
  geom_text(data = variables, aes(label = variable, x =
   ↪  PC1+0.3, y = PC2+0.1), color = "red") +
  geom_segment(
    data = variables, aes(xend = PC1, yend = PC2,
     ↪  x=0,y=0),
    arrow = arrow(),
    color = "red"
  ) +
```

```
  theme_minimal()
```

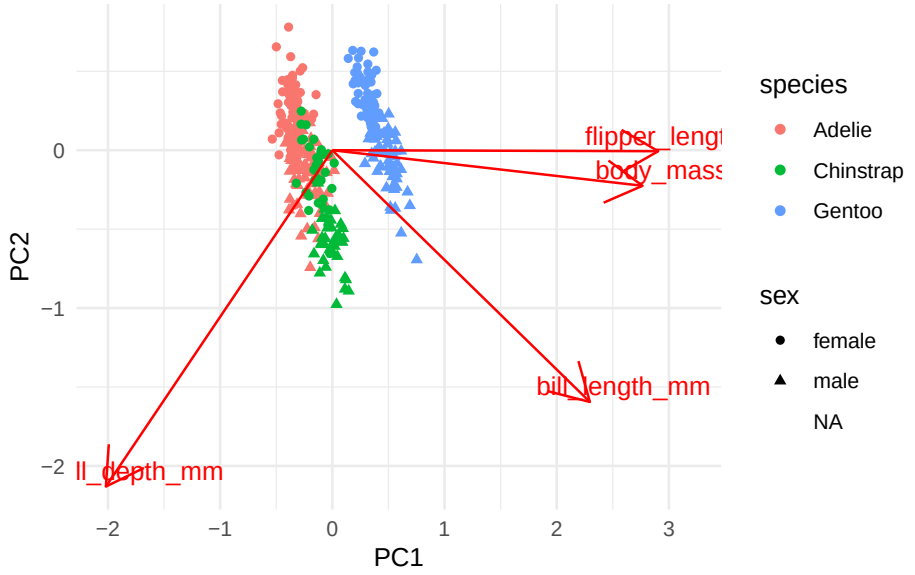


Enfin, comme discuté plus haut, il peut être intéressant d'ajouter de l'information supplémentaire au graphique, par exemple en colorant les points selon l'espèce et en changeant leur forme selon le sexe du manchot. Pour se faire, il faut connecter le tableau d'informations complémentaires au tableau d'observations avant de tracer le graphique :

```
observations |>
  bind_cols(infos_complementaires) |>
  ggplot(aes(x = PC1, y = PC2)) +
  geom_point(aes(color = species, shape = sex)) +
  geom_text(data = variables, aes(label = variable, x =
  ↪  PC1+0.3, y = PC2+0.1), color = "red") +
  geom_segment(
```

```
    data = variables, aes(xend = PC1, yend = PC2,
    ↪  x=0,y=0),
    arrow = arrow(),
    color = "red"
  ) +
  theme_minimal()
```

```
Warning: Removed 9 rows containing missing values or
values
outside the scale range (`geom_point()`).
```



Dans ce graphique, on peut donc constater qu'en général, les manchots Gentoo sont plus grands que les deux autres espèces (plus à droite sur le 1er axe), qui ne se distinguent pas beaucoup sur les deux axes illustrés. Leurs différences de niches écologiques doit se situer dans d'autres variables que nous n'avons pas mesurées.

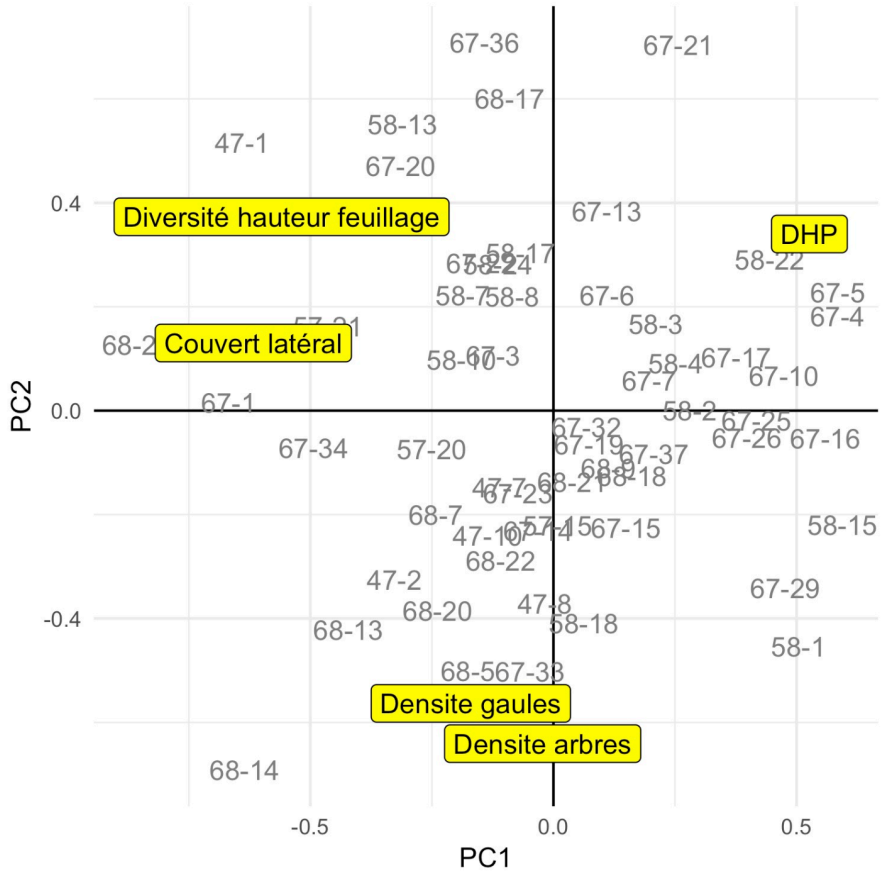
On peut aussi constater que les mâles ont en général des becs plus gros que les femelles (plus en bas sur l'axe 2), et ce, peu importe l'espèce.

Notez cependant que le but de l'ACP n'est pas de trouver des groupes comme tel. Le but de l'ACP est de résumer la variabilité dans un nombre réduit d'axes. Si on veut déterminer si il existe des groupes dans nos données ou non et comment les séparer, il faut consulter les techniques du Chapitre 26.

23.10. Un exemple concret

Voici, pour terminer ce chapitre, un exemple concret d'ACP que j'ai effectuée pour mon projet de maîtrise. Dans mes travaux terrain, nous avions visité une 50aine de peuplements forestiers du Parc National de la Mauricie. Pour chacun, nous avions noté (entre autres), la densité d'arbres, la densité de gaules, le couvert latéral, le diamètre des arbres (DHP) et une mesure de stratification de la forêt (diversité de hauteur de feuillage). Une fois la saison de terrain terminée, une des premières choses que j'ai fait était de faire une ACP pour voir à quoi ressemblaient mes données. Notez que comme ces variables sont à des échelles très différentes, j'ai dû calculer mon ACP sur la matrice de corrélation.

Voici le graphique de mes résultats :



Moi et mon directeur étions bien contents de ce graphique, puisque les axes fournis étaient facilement interprétables et nous permettaient de bien comprendre/décrire nos peuplements.

Le premier axe de l'ACP, formé d'un côté par le DHP et de l'autre le couvert latéral et la diversité de hauteur nous informait facilement sur l'ouverture du sous-bois. Les sites à droite sont formés de grands arbres, entre lesquels il n'y a pas grand chose. On y marche facilement. Les

sites à gauche sont formés de petits arbres, avec beaucoup de branches basses et d'arbustes en sous-bois. Ce sont des sites où il est difficile de circuler, mais facile de se cacher.

Le deuxième axe, quant à lui, nous informait clairement de la quantité de tiges que l'on pouvait trouver dans un site. Il était formé essentiellement des deux variables de densité d'arbres et de densités de gaules.

L'ACP nous informait que ces deux gradients étaient relativement indépendants, et que l'on pourrait donc en faire des analyses combinées sans problèmes.

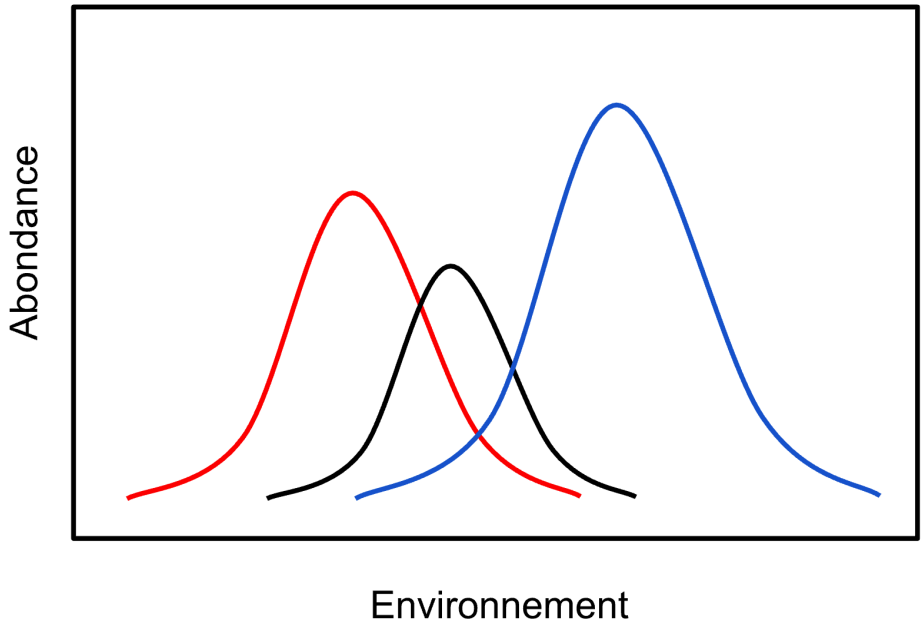
24. L'analyse factorielle des correspondances

24.1. Introduction

Après avoir bien saisi l'analyse en composantes principales (ACP) au Chapitre 23, il importe ici de réaliser maintenant une limitation qui lui est associée. Nous avons mentionné plusieurs fois qu'une des assomptions de l'ACP est qu'elle recherche des relations linéaires. Cela fonctionne bien pour beaucoup de phénomènes, particulièrement lorsque l'on parle de variables abiotiques.

Par contre, si nos variables sont des abondances d'espèces et que l'on réfléchit un peu à leur écologie, on réalise souvent que leur réponse à l'environnement est rarement linéaire. L'évolution a plutôt tendance à créer des réponses en forme de cloche face à un gradient environnemental. Chaque espèce ayant tendance à préférer certaines valeurs, et à être moins présente lorsque l'on s'éloigne de sa valeur idéale, soit d'un côté ou de l'autre. Un peu comme nous, on est bien quand il fait 21°C, mais à 40°C ou à 0°C, on est moins confortables.

Visuellement, on peut facilement s'imaginer trois espèces, dont les abondances varient comme ceci à travers un gradient environnemental (p. ex. pH, ouverture de la canopé, etc.) :



Si l'on voulait regarder l'association entre ces trois espèces dans une ordination, l'ACP ne serait pas l'analyse appropriée, puisqu'elle ne cherche que des relations linéaires. C'est pourquoi, pour ce genre de situations, il existe un autre type d'ordination, nommé en français l'analyse factorielle des correspondances (AFC) ou *correspondance analysis* (*CA*) en anglais.

Chaque fois que les variables dans votre matrice de données seront des abondances d'espèces, il sera approprié d'utiliser cette analyse.

Notez bien que pour que l'analyse fonctionne, il n'est pas nécessaire d'avoir étudié l'ensemble de la courbe d'une espèce. L'analyse fonctionnera tout aussi bien si vous n'avez attrapé que la partie ascendante ou descendante de la courbe.

24.2. Fonctionnement de l'AFC

L'AFC est en fait une sorte de croisement entre l'ACP et l'analyse de khi-carré (voir Chapitre 19). Plutôt que d'utiliser directement la matrice de données pour créer une matrice de variance-covariance, l'AFC travaille sur une matrice des différences entre les comptes observés et ceux attendus selon une hypothèse d'indépendance entre les variables et les sites (exactement comme dans le test de khi-carré). On applique ensuite à ces différence une double-transformation sur chaque valeur, en les divisant par le produit de la racine carrée du total de la colonne et de la ligne :

$$\frac{(o_{ij} - e_{ij})}{\sqrt{r_i}\sqrt{c_j}}$$

Comme dans les autres chapitres, il n'est pas si important de se rappeler de la formule exacte. Ce qu'il est surtout important de comprendre est que plus variables (i.e. les espèces) ou les observations (i.e. les sites) seront associés, plus les valeurs dans la matrice seront élevées. Au contraire, si nos espèces ou nos sites sont entièrement indépendants les uns des autres (i.e. si les espèces sont réparties aléatoirement dans les sites), les valeurs seront très près de zéro.

Une fois cette matrice de valeurs constituée, l'AFC applique une série d'opérations matricielles pour résumer la variation dans une série de nouvelles variables. Par contre, contrairement à l'ACP, on obtient au bout de l'opération deux séries d'eigenvectors, soit une pour les espèces et l'autre pour les sites. Tout comme dans l'ACP, le premier axe aura l'eigenvalue le plus élevé, les axes suivant se répartissant la variation restante. Au total, l'AFC produira $min(n, p) - 1$ axes différents. C'est-à-dire le plus petit nombre entre le nombre de colonnes moins 1 ou et le nombre de lignes moins 1 de la matrice de données.

Habituellement, on n'interprète qu'un ou deux axes de l'AFC, car ces axes sont censés correspondre à des gradients environnementaux et

l'interprétation de trois axes indépendants peut devenir rapidement très abstraite.

Comme nous en avons déjà discuté précédemment, la chose la plus difficile avec les ordinations est probablement de maîtriser le vocabulaire et les abréviations (qui sont en plus mentionnées en anglais dans les sorties de R). Pour l'AFC, vous devrez en particulier savoir que la somme des eigenvalues (ce qui correspond au khi-carré total de la matrice divisé par le nombre d'observations) s'appelle dans cette analyse l'**inertie**. Il s'agit d'une mesure d'association (*lack of independance*) entre les lignes et les colonnes. Dans les sorties d'une AFC, vous pouvez donc lire les valeurs d'inertie comme vous liriez les eigenvalues dans l'ACP.

24.3. Assomptions

L'AFC assume deux choses importantes à propos de vos données pour faire ses calculs. Contrairement à certaines autres analyses où on peut étirer les assomptions, ici elles doivent être clairement respectées. La première est que toutes les variables doivent être mesurées dans les mêmes dimensions physiques. Vous ne pourriez PAS avoir certaines colonnes avec des dénombrement d'individus et d'autres avec des mesures de pH par exemple. L'autre condition importante est que, comme pour l'analyse de khi-carré, toutes les valeurs doivent être des entiers positifs ou des zéros. On ne peut PAS calculer une AFC sur une matrice d'abondances relatives ou avec des pourcentages.

Vous lirez aussi parfois dans des rapports ou des articles que l'AFC est une analyse qui est sensible aux espèces rares (i.e. celles ayant de très faibles abondances et apparaissant rarement dans les données). Ce qu'il est important de comprendre est que, comme le rapportent Legendre et Legendre (Numerical Ecology 1998, p. 462), cette sensibilité n'est qu'apparente. Les espèces rares, de par leurs faibles abondances, ont très peu de poids dans le calcul comme tel. Elles peuvent néanmoins

apparaître à des endroits un peu extrêmes dans les graphiques. C'est pourquoi la recommandation de Legendre (et la mienne du même coup!) est de conserver toutes les espèces pour les calculs, et de simplement cacher les espèces rares au moment de la visualisation. Vous lirez néanmoins parfois dans des rapports ou des articles que les espèces rares ont été supprimées avant le calcul d'une AFC.

24.4. Labo : L'AFC

Pour essayer l'analyse factorielle des correspondances dans R, nous allons devoir charger une base de données externes, nommée oiseaux.xlsx[1]. Cette dernière contient l'abondance d'une dizaine d'espèces d'oiseaux notées à une dizaine de sites durant ma maîtrise au Parc national de la Mauricie.

Commençons donc par charger les librairies nécessaires et la base de données Excel. Comme pour l'ACP, nous devrons cacher la colonne contenant le nom des sites pour que l'AFC puisse s'effectuer correctement, mais garder le nom des sites dans les graphiques.

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
```

[1] https://drive.google.com/file/d/1Z7FwE6vhLtEHyuor1sKBr_L7MgJLV-C0/view?usp=sharing

```
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(readxl)
library(vegan)
```

```
Loading required package: permute
Loading required package: lattice
This is vegan 2.6-8
```

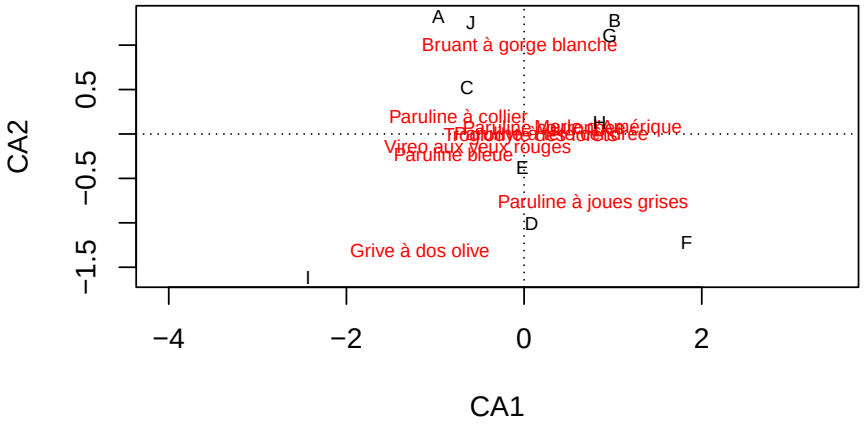
```
oiseaux <- read_excel("donnees/oiseaux.xlsx") |>
  column_to_rownames("Site")
```

Pour calculer l'AFC, il faudra utiliser la fonction de vegan nommée `ca` (*Correspondance analysis*).

```
afc <- ca(oiseaux)
```

Pour visualiser le résultats de l'AFC, la façon rapide est à l'aide de la fonction `plot`, appelée sur notre objet de résultats :

```
plot(afc)
```



Si vous voulez personnaliser ce graphique, toutes les techniques montrées avec ggplot2 au Chapitre 23 fonctionneront de façon identique, à part pour le nom des axes qui se nommeront CA1 et CA2 plutôt que PCA1 et PCA2, etc.

On peut obtenir l'ensemble des eigenvalues et eigenvectors de notre analyse à l'aide de la fonction summary, comme ceci :

```
summary(afc)
```

```
Call:
ca(X = oiseaux)

Partitioning of scaled Chi-square:
              Inertia Proportion
Total          0.9578          1
```

```
Unconstrained  0.9578          1

Eigenvalues, and their contribution to the scaled
Chi-square

Importance of components:
                          CA1    CA2    CA3     CA4
Eigenvalue            0.3319 0.2334 0.1543 0.09973
Proportion Explained  0.3466 0.2437 0.1611 0.10412
Cumulative Proportion 0.3466 0.5903 0.7514 0.85551
                          CA5     CA6     CA7      CA8
Eigenvalue            0.06380 0.05961 0.01284 0.002120
Proportion Explained  0.06661 0.06224 0.01341 0.002213
Cumulative Proportion 0.92212 0.98436 0.99777 0.999983
                           CA9
Eigenvalue            1.581e-05
Proportion Explained  1.651e-05
Cumulative Proportion 1.000e+00
```

En se basant sur ces sorties, on peut voir que les deux premiers axes de l'AFC expliquent ensemble 59% de la variabilité dans notre matrice d'abondances.

Dans le graphique, on peut constater que les deux espèces les plus extrêmes sur l'axe 1 sont le Merle d'Amérique d'un côté et la Grive à dos olive de l'autre. On peut donc en conclure que ces deux espèces sont rarement retrouvées ensemble. Si l'une est présente, l'autre sera presque toujours absente.

Remarquez qu'ici, l'AFC ne peut pas nous renseigner sur les causes de ce phénomène. Il pourrait être causé, entre autres, parce que ces deux espèces s'évitent parce qu'elles sont en compétition directe une avec l'autre, ou simplement parce qu'elles utilisent des milieux tellements différents qu'un site ne peut jamais correspondre en même temps aux besoins des deux espèces.

24.5. Exercice : L'AFC

À l'aide du tableau de données `dune` inclu avec la librairie `vegan` (utilisez la fonction `data(dune)` pour l'activer), calculez une AFC et répondez aux questions suivantes.

Pour votre information, le tableau de données est formé d'observations concernant 30 espèces de plantes observées dans 20 sites dans les dunes néerlandaises. Vous pouvez en savoir plus sur ces données avec la commande `?dune`

- Les données sont-elles appropriées pour appliquer une AFC?
- Quel pourcentage de la variance est expliqué par le premier axe?
- Et par le deuxième?
- Quelles espèces sont les plus représentatives du premier axe de l'AFC?
- Avancez une hypothèse sur les causes de ce gradient
- Et pour le deuxième axe, quelles sont les espèces les plus représentatives?

25. Le cadrage multidimensionnel non-métrique (NMDS)

25.1. Introduction

Alors, nous y voici, le chapitre des notes de cours avec le titre le plus long et compliqué : le cadrage multidimensionnel non-métrique! Vous me pardonnerez, je l'espère, de ne pas utiliser le nom complet de cette technique chaque fois que nous y ferons référence. Nous utiliserons plutôt son acronyme anglais NMDS (*nonmetric multidimensional scaling*), puisque c'est par ce nom que tous les biologistes le connaissent.

Le NMDS, contrairement à l'ACP et l'AFC ne s'intéresse pas à la ressemblance entre les variables (covariance, corrélation), mais bien aux distances entre les observations. Il existe d'autres techniques que le NMDS qui s'intéressent à ces distances (par exemple l'analyse en coordonnées principales; ACoP), mais le NMDS est de loin la plus connue et la plus robuste.

25.2. Terminologie

Il y a beaucoup de termes techniques dans le nom du NMDS, mais au final vous verrez que ce n'est pas si mal quand on prend le temps de les regarder un par un. Tout d'abord, qu'entend-on par non-métrique?

Ce terme signifie, dans ce contexte, que le NMDS ne conserve pas nécessairement les distances exactes entre les observations. Non seulement elle tourne le nuage de points (comme le ferait l'ACP et l'AFC), mais en plus, elle peut changer la position individuelle de certains points dans le nuage pour que les observations similaires soient le plus près possibles les unes des autres et que les différentes soient plus éloignées. Évidemment, elle ne déplace par les observations n'importe comment. Dans tous les cas, le rang original des distances sera conservé. C'est à dire que la paire d'observations la plus semblable demeurera toujours la plus semblable, la paire d'observations la plus différente demeurera toujours la plus différente, etc. Cela fonctionne en fait, exactement comme les tests non-paramétriques (Spearman, Wilcoxon, etc), mais ici on est en plusieurs dimensions (i.e. multi-dimensionnel).

25.3. Fonctionnement

Contrairement à toutes les techniques statistiques que nous avons vues jusqu'à présent, le NMDS ne peut pas être décrit par une simple opération d'algèbre matriciel. Le NMDS s'effectue plutôt en suivant un algorithme, qui en une succession d'itérations, s'approche progressivement de la configuration optimale du nuage de points.

Voici un aperçu du processus suivi lors d'une analyse par NMDS :

1. Il faut, comme pour les autres analyses, préparer notre matrice de données, en s'assurant de fournir uniquement des colonnes contenant des chiffres.
2. Il aussi choisir combien de dimensions (i.e. d'axes; k) nous voulons obtenir dans le résultat du NMDS. Il s'agit d'une différence importante comparé à l'ACP et l'AFC. Nous y reviendrons plus loin.
3. R calcule pour nous ensuite la matrice de dissimilarités, basées sur le distance que nous avons choisie (voir Chapitre 22).

4. R choisit alors pour chacune des nos observations, une coordonnée aléatoire dans l'espace en k dimensions.
5. Les observations sont ensuite déplacées itérativement (e.g. une à la fois) de façon à ce que la distance entre les observations dans l'espace en k dimensions s'approche le plus près possible des vraies distances entre les observations (celles mesurées à l'étape 3)
6. La position des observations est jugée finale lorsque le déplacement des objets n'améliore plus la correspondance entre les distances en k-dimensions et les distances réelles

Cette façon de faire par algorithme itératif apporte une différence importante par rapport aux deux ordinations précédentes. Puisque la configuration de départ est aléatoire, il peut arriver que le NMDS n'arrive pas toujours à la même solution. Lorsque nous lancerons la fonction R, vous remarquerez donc qu'elle calcule 20 NMDS sur notre matrice de données et nous fournit ensuite le résultat du meilleur NMDS trouvé parmi les 20 essais.

L'autre différence majeure, comme suggéré dans l'algorithme, est que contrairement à l'ACP et à l'AFC, on choisit d'avance combien on veut d'axes dans la solution finale (i.e. la valeur de k). Dans l'ACP et l'AFC, je vous le rappelle, le calcul nous fournissait autant d'axes qu'il y avait de variables dans nos données originales. C'était à nous de choisir lesquels interpréter ensuite. Ici, on choisit d'avance combien on veut en recevoir.

Mais comment choisir la valeur de k? Il n'y a pas de réponse facile à cette question. Moins on a d'axes, et plus notre résultat sera facile à interpréter pour notre petit cerveau humain. Rappelez-vous comment pour l'ACP, il devenait difficile d'interpréter les axes 3 et 4. Par contre, si le nombre d'axes que l'on choisit est trop petit, la solution que le NMDS trouvera pourrait ne pas être représentative de la structure de nos données originales (e.g. elle pourrait l'avoir trop simplifiée, la rendant inutilisable).

Comment savoir si on a choisi une valeur de k trop petite? C'est là qu'entre en ligne de compte un nouveau concept : le **stress**. Le stress dans une NMDS sert à mesurer combien loin est la configuration trouvée par le NMDS par rapport à la configuration originale de nos points. Il se calcule comme une régression, où chacun des points sera une paire d'observations dans notre matrice de données. On mettra en X la distance originale entre les deux observations et en Y la distance calculée avec les axes du NMDS (rassurez-vous, R s'occupe de tout ça pour nous). Comme le NMDS est un cadrage multi-dimensionnel non-métrique, cette régression sera calculée sur les rangs des observations plutôt que sur les valeurs elles-mêmes. Enfin, le stress comme tel se trouve à être la moyenne des résidus de cette relation. Si la relation est bonne, le stress sera très faible. Au contraire, si les distances trouvées par le NMDS sont très différentes des originales, le stress sera élevé.

Comment savoir à quel moment le stress est trop élevé? Il n'y a, comme d'habitude, pas de réponse absolue à cette question. Voici ce qu'en disent habituellement les manuels de statistiques :

- Une valeur de stress >= 0.3 nous indique que la configuration trouvée par le NMDS ne vaut pas grand chose. On aurait lancé les points au hasard et on aurait fait aussi bien que lui.
- On dit que si la valeur de stress est >= 0,2, on ne devrait pas interpréter les sorties. Certaines sources mettent ce seuil à 0,15 plutôt que 0,2
- Idéalement, le stress d'un NMDS devrait être < 0,1

Remarquez que ces guides d'interprétation assument que le calcul de stress utilisé est celui de Kruskal. Si jamais vous utilisez un autre logiciel que R, assurez-vous que le stress est bien calculé de la même façon avant d'utiliser ce tableau d'interprétation.

Une fois le concept de stress bien assimilé, le vrai compromis à comprendre lorsque l'on travaille avec un NMDS est qu'un modèle avec plus de dimensions (k plus grand) présentera un stress plus faible, mais sera

aussi plus difficile à interpréter. Au contraire, un modèle avec moins de dimensions sera (k plus faible) sera facile à interpréter, mais présentera un stress plus élevé, pouvant aller jusqu'au point où on ne devrait pas interpréter les sorties du modèle.

25.4. Labo : Le NMDS

Pour essayer le NMDS dans la vraie vie, nous allons l'appliquer à la base de données oiseaux comme au Chapitre 24, pour essayer de résumer les différences entre les sites (les lignes de notre base de données).

L'étape de préparation sera donc identique :

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ----------------- tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(readxl)
library(vegan)
```

```
Loading required package: permute
Loading required package: lattice
This is vegan 2.6-8
```

```
oiseaux <- read_excel("donnees/oiseaux.xlsx") |>
  column_to_rownames("Site")
```

La fonction que nous utiliserons pour calculer le NMDS se nomme **metaMDS** et provient de la librairie **vegan**. Elle attend au minimum 3 arguments, soit la matrice de données sur laquelle faire le calcul, le type de distance à utiliser et le nombre d'axes que devra comporter notre solution. Comme notre matrice contient des décomptes d'espèces, nous allons utiliser la distance de Bray-Curtis (voir Chapitre 22 pour la justification), et nous allons commencer par demander deux axes :

```
resultat <- metaMDS(oiseaux,distance = "bray",k=2)
```

Vous voyez que R lance à ce moment une série de calculs de NMDS, et pour chacun, il nous fournit la valeur de stress trouvée :

```
Run 0 stress 0.08878298
Run 1 stress 0.07541769
... New best solution
... Procrustes: rmse 0.1109761  max resid 0.2449838
Run 2 stress 0.08878298
Run 3 stress 0.1178968
Run 4 stress 0.08878298
Run 5 stress 0.07541769
... Procrustes: rmse 1.59902e-06  max resid 3.708063e-06
... Similar to previous best
Run 6 stress 0.08878298
Run 7 stress 0.08878298
Run 8 stress 0.1635454
```

```
Run 9 stress 0.07541769
... New best solution
... Procrustes: rmse 7.321764e-07  max resid
1.351767e-06
... Similar to previous best
Run 10 stress 0.1735171
Run 11 stress 0.1202882
Run 12 stress 0.08878298
Run 13 stress 0.1435191
Run 14 stress 0.1317843
Run 15 stress 0.1643384
Run 16 stress 0.1865188
Run 17 stress 0.08878298
Run 18 stress 0.1552375
Run 19 stress 0.127419
Run 20 stress 0.07541769
... New best solution
... Procrustes: rmse 5.964561e-07  max resid
1.261369e-06
... Similar to previous best
*** Best solution repeated 1 times
```

Les chiffres sur vos ordis seront probablement différents des miens, puisque, rappelez-vous, la configuration de départ de chaque calcul de NMDS est choisie aléatoirement. La dernière ligne de la sortie nous informe que l'algorithme a trouvé la solution optimale après les 20 essais. Nous verrons plus loin comment ajouter des essais si jamais la solution n'est pas trouvée.

Voyons un peu le résumé de notre calcul de NMDS :

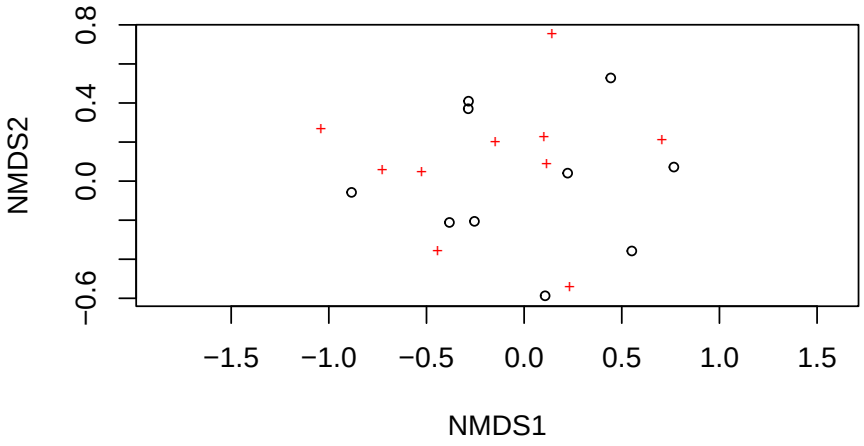
```
resultat
```

```
Call:
metaMDS(comm = oiseaux, distance = "bray", k = 2)

global Multidimensional Scaling using monoMDS

Data:     oiseaux
Distance: bray

Dimensions: 2
Stress:     0.07541769
Stress type 1, weak ties
Best solution was repeated 1 time in 20 tries
The best solution was from try 20 (random start)
Scaling: centring, PC rotation, halfchange scaling
Species: expanded scores based on 'oiseaux'
```

La chose importante à voir dans cette sortie est surtout la valeur de stress, qui est de 0,075. Comme elle est < 0,1, on peut interpréter sans crainte les sorties de notre modèle.

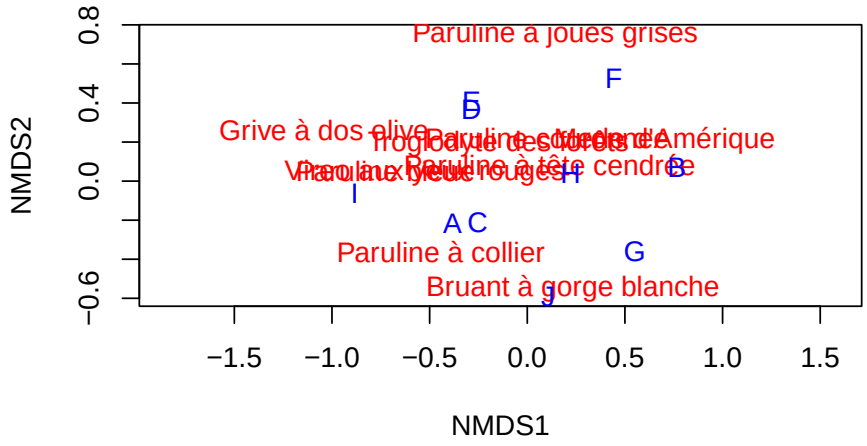
Si on essaie la fonction `plot` avec notre objet de résultats, vous verrez que les sorties sont plus ou moins intéressantes :

```
plot(resultat)
```



Je ne sais pas où ils avaient la tête quand ils ont préparé cette fonction, mais clairement, on ne peut rien en tirer d'intéressant. La librairie **vegan** nous fournit par contre quelques fonctions permettant d'afficher le nom des espèces et des sites dans le graphique pour le rendre plus utile :

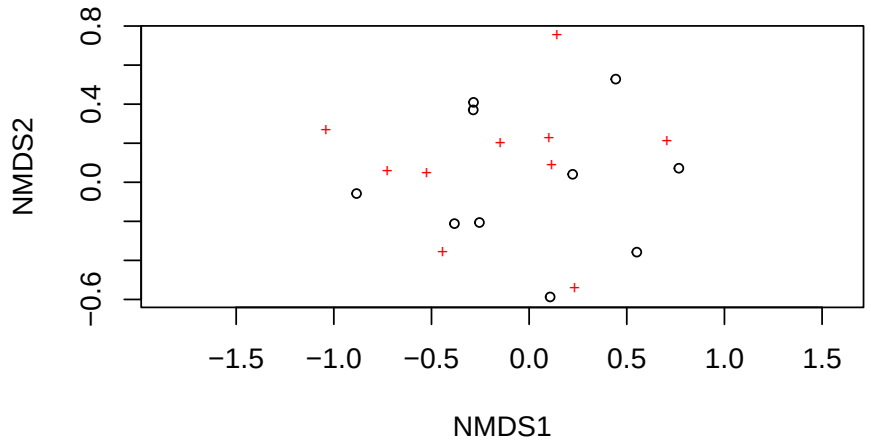
```
plot(resultat, type = "n")
text(resultat, display = "species", col = "red")
text(resultat, display = "sites", col = "blue")
```



Nous avons donc maintenant en bleu chacun des sites dans notre matrice de données, et on a affiché aussi les espèces, en rouge. Si jamais vous voudriez personnaliser davantage le graphique sortant du NMDS, sachez que la façon de faire avec `ggplot2` et la PCA montrée au Chapitre 23 fonctionnera aussi avec le NMDS, pour autant que vous utilisiez les axes nommés NMDS1, NMDS2 et la fonction `plot` plutôt que `biplot`.

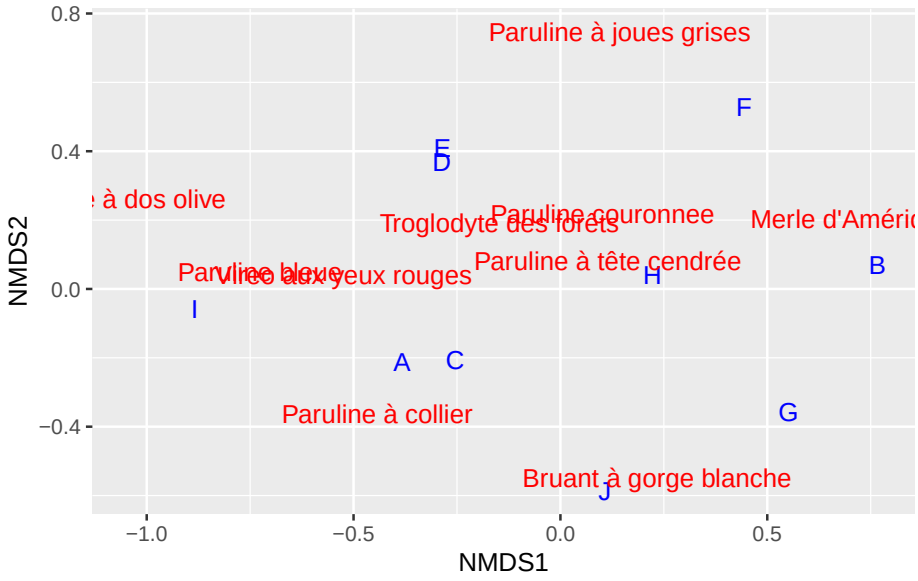
Par exemple :

```
x <- plot(resultat)
```




```
especes <- x$species |>
  as.data.frame() |>
  rownames_to_column("espece")

sites <- x$sites |>
  as.data.frame() |>
  rownames_to_column("site")

sites |>
  ggplot(aes(x = NMDS1, y = NMDS2)) +
  geom_text(color = "blue", aes(label = site)) +
  geom_text(
    data = especes,
```

```
    aes(label = espece),
    color = "red"
  )
```



Aussi il est possible que le graphique sur votre ordinateur soit inversé par rapport au mien, avec le merle à gauche et la grive à droite. C'est tout à fait normal. Comme pour l'ACP, le sens des axes n'a pas d'importance dans l'absolu.

Les deux sites les plus différents dans nos communautés sont donc le B d'un côté et le I de l'autre. Comme pour l'AFC, les différences principales semble être définies par une différence d'abondance avec beaucoup de Merles d'un côté, à des sites avec beaucoup des Grives de l'autre. Mais le gradient ici est encore plus clair.

Le deuxième axe quant à lui est défini par les sites F d'un côté et J de

l'autre, et est surtout caractérisé par une différence d'abondance entre la Paruline à joues grises et le Bruant à gorge blanche.

Voyons maintenant ce qui arriverait si on avait choisi de produire un NMDS à un seul axe plutôt que deux :

```
resultat_k1 <- metaMDS(oiseaux,distance = "bray",k=1)
```

```
Run 0 stress 0.2453025
Run 1 stress 0.2398848
... New best solution
... Procrustes: rmse 0.08534684  max resid 0.1777978
Run 2 stress 0.4614311
Run 3 stress 0.2515867
Run 4 stress 0.2456181
Run 5 stress 0.260419
Run 6 stress 0.5077229
Run 7 stress 0.2547873
Run 8 stress 0.2437508
Run 9 stress 0.4622032
Run 10 stress 0.3555717
Run 11 stress 0.466635
Run 12 stress 0.4307478
Run 13 stress 0.3844465
Run 14 stress 0.509052
Run 15 stress 0.4136504
Run 16 stress 0.4344361
Run 17 stress 0.3269235
Run 18 stress 0.3419624
Run 19 stress 0.4207077
Run 20 stress 0.3233229
*** Best solution was not repeated -- monoMDS stopping
criteria:
    20: scale factor of the gradient < sfgrmin
```

R nous répond à la fin de la fonction qu'il n'a pas réussi à trouver de bonne configuration, 20 essais n'étaient pas assez :

```
*** No convergence -- monoMDS stopping criteria:
```

La première chose à faire lorsque vous vous heurtez à ce message est de relancer la fonction en lui permettant de faire plus d'essais. Pour se faire, il faut ajouter un argument nommé `trymax`. Pour être vraiment sûrs de notre affaire, on lui demande ici de refaire notre NMDS de `k=1`, mais avec 1000 essais plutôt que 20 :

```
resultat_k1 <- metaMDS(oiseaux,distance = "bray",k=1,
↪  trymax = 1000)
```

Vous devriez maintenant obtenir ce message, après quelques centaines d'essais, chez moi j'en ai eu besoin de 273 :

```
*** Solution reached
```

On peut maintenant inspecter ce résultat pour voir la valeur de stress de la meilleure configuration trouvée :

```
resultat_k1
```

```
Call:
metaMDS(comm = oiseaux, distance = "bray", k = 1, trymax
= 1000)

global Multidimensional Scaling using monoMDS

Data:     oiseaux
Distance: bray

Dimensions: 1
```

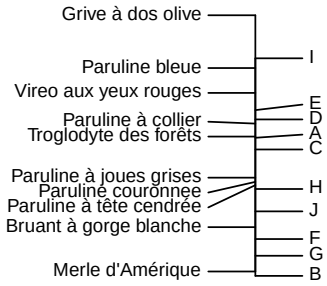
```
Stress:     0.2353499
Stress type 1, weak ties
Best solution was repeated 1 time in 110 tries
The best solution was from try 110 (random start)
Scaling: centring, PC rotation, halfchange scaling
Species: expanded scores based on 'oiseaux'
```

Cette fois, la valeur de stress trouvée est de 0,235.

Comme cette valeur est > 0,20, on ne doit pas interpréter ces résultats. Cet NMDS n'est pas suffisamment représentatif de nos données originales.

Pour des raisons pédagogiques, je vous suggère tout de même de faire le graphique de ce NMDS, pour bien comprendre à quoi ça peut ressembler un NDMS à un seul axe :

```
plot(resultat_k1)
```



Remarquez comment ce graphique est construit plus ou moins comme si on avait pris le graphique de `k=2`, et qu'on l'avait écrasé verticalement en une seule ligne (et qu'on l'avait ensuite tourné de 90° dans le sens anti-horaire).

Il y a aussi une dernière nuance importante à savoir à propos de la fonction `metaMDS`. Si jamais vous décidez d'utiliser la distance euclidienne, sachez que la fonction tentera de transformer vos données sans vous en parler! Rappelez-vous que la librairie `vegan` a d'abord été créée pour analyser des communautés, avec une colonne par espèce. Dans ces cas, il est souvent pratique d'appliquer une transformation wisconsin aux données. Ce genre d'opération excède de loin le cadre du cours, mais ce qu'il est important de savoir est que, si vous utilisez `metaMDS` et la distance euclidienne, il faut aussi modifier l'argument `autotransform`, pour le mettre à FALSE.

25.5. Exercice : Le NMDS

Comme au chapitre Chapitre 24, répondez aux questions suivantes en vous basant sur le tableau de données `dune`de la librairie `vegan` :

- Quelle distance devra-t-on utiliser pour calculer un NMDS sur ces données?
- Calculez un NMDS avec un seul axe
- Est-ce que ce NMDS peut-être interprété?
- Calculez un NMDS avec deux axes. Est-il plus interprétable?
- Comparez la formation des axes avec l'AFC. Est-ce plutôt semblable ou plutôt différent?
- Quels sont les deux sites les plus différents sur l'axe 1?
- À l'aide des données environnementales fournies dans un second tableau de données (`data(dune.env)`), colorez les sites en fonction du taux d'humidité (*Moisture*) et choisissez une forme en fonction du type de gestion (*Management*).

- Le gradient principal des espèces semble-t-il expliqué par une de ces variables?

26. Les analyses de regroupement

26.1. Introduction

Dans ce chapitre, nous verrons deux techniques qui nous permettent de déterminer si un jeu de données multivarié (qui contient plusieurs variables) contient des regroupements naturels de données.

On peut avoir différentes motivations pour chercher des regroupements dans les données. Les regroupements peuvent par exemple faciliter notre compréhension des données ou simplifier les processus de décisions. Ils agissent comme un préjugé : plutôt que de regarder tous les détails d'un échantillon, on peut le juger ou le comprendre uniquement par son appartenance à un groupe. Ce n'est pas très éthique d'agir ainsi lorsque l'on parle d'humains, mais le reste de la nature est moins soucieux de ce genre de problématique!

Nous verrons dans ce chapitre deux techniques de regroupements, soit l'analyse des K-means et la classification hiérarchique.

26.2. La technique des K-means

Pour présenter la technique des K-means, démarrons avec un scénario où nous nous avons observé une douzaine d'échantillons, sur lesquels nous avons mesuré deux variables, que nous présentons ici, une sur l'axe des X et l'autre sur l'axe des Y :

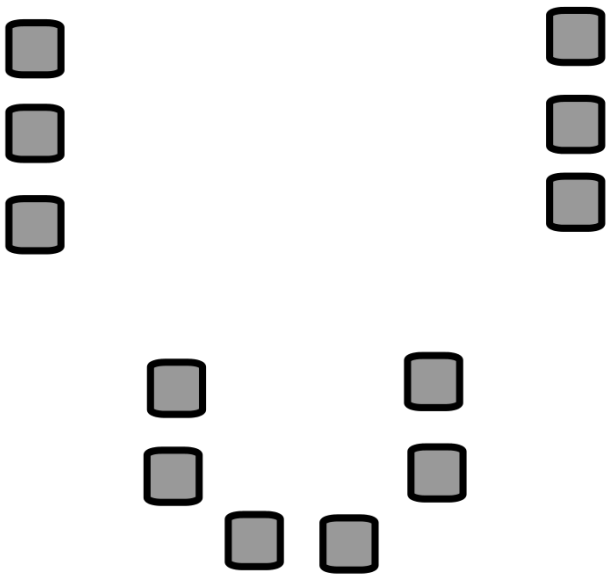


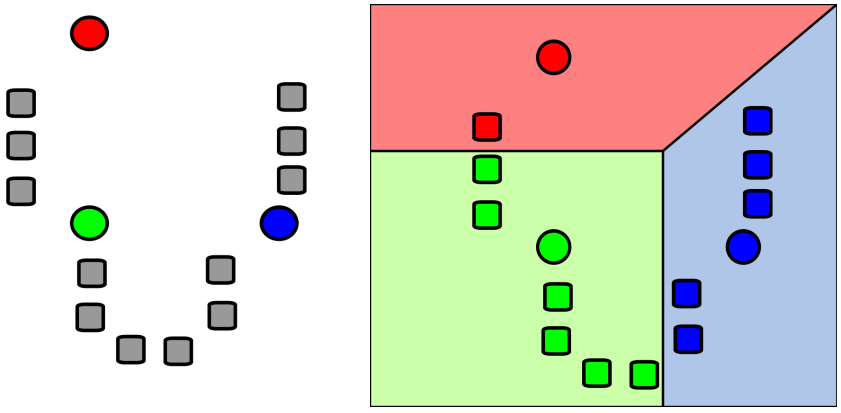
Figure 26.1.: Adapté de I, Weston.pace, CC BY-SA 3.0, via Wikimedia Commons

Si je vous demandais intuitivement combien de groupes contient ce jeu de données, vous répondrez probablement 3 groupes.

Mais où tracer précisément la ligne entre les groupes de façon objective?

26.2.1. Fonctionnement de l'algorithme

Comme illustré dans la figure suivante, la première étape du K-means consiste à placer dans l'espace des centroïdes au hasard. Ensuite, on associe chaque observation au centroïde le plus près d'elle :



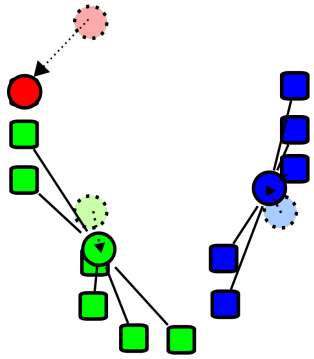
(a) I, Weston.pace, CC BY-SA 3.0, via Wikimedia Commons

(a) I, Weston.pace, CC BY-SA 3.0, via Wikimedia Commons

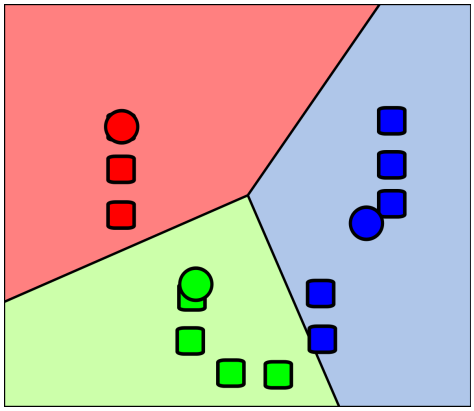
Par la suite, l'algorithme répètera les deux étapes suivantes, jusqu'au moment où les observations cesseront de changer de groupe :

- Calculer le centroïde (la moyenne en plusieurs dimensions) de chacun des groupes
- Assigner chaque observation au centroïde le plus proche (en se basant sur la distance euclidienne).

Cette façon de fonctionner a l'avantage qu'elle garantit que les groupes formés minimisent la variance intra-groupe. Autrement dit, que les groupes formés soient le plus homogènes possible.



(a) I, Weston.pace, CC BY-SA 3.0, via Wikimedia Commons



(a) I, Weston.pace, CC BY-SA 3.0, via Wikimedia Commons

26.2.2. À propos du nombre de groupes

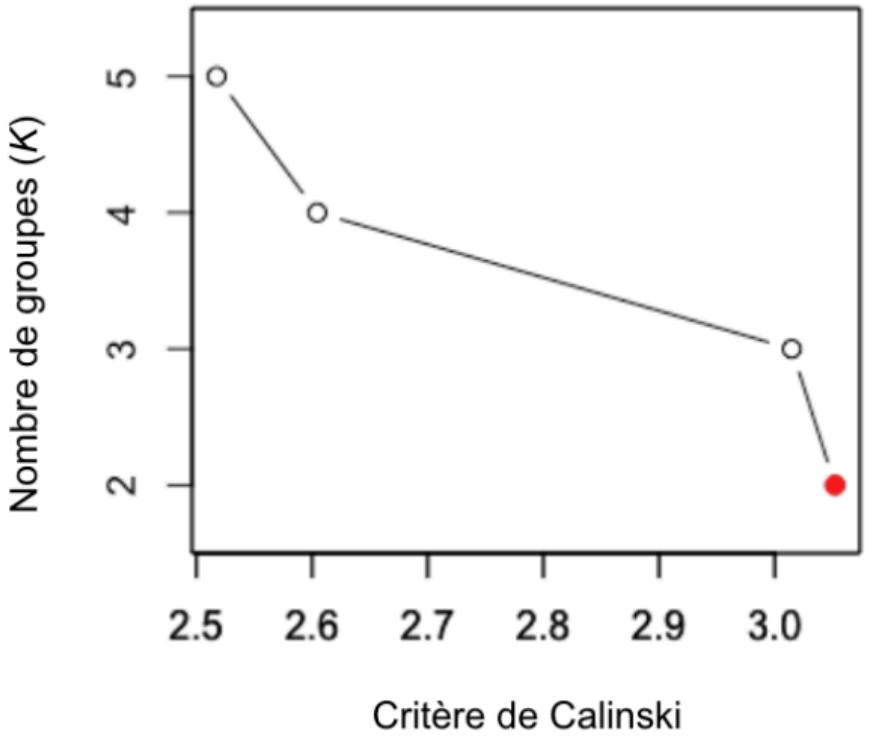
Vous avez peut-être remarqué dans l'exemple précédent que nulle part dans la procédure, l'algorithme du K-means change le nombre de groupes. C'est parce que, un peu comme pour le NMDS, nous devons spécifier au démarrage combien de groupes nous cherchons (qui se nomme ici k). L'algorithme fournira une solution différente selon le k choisit par l'utilisateur.

26.2.3. Le critère de Calinski

Donc, comment doit-on procéder si on ne sait pas à l'avance combien de groupes contient notre jeu de données? Comment peut-on savoir le plus objectivement possible combien de groupes il contient? L'outil pour y arriver se nomme le critère de Calinski.

Le **critère de Calinski**, un peu comme un ratio de F, se calcule en divisant la variance inter-groupe par la variance intra-groupe.

La stratégie pour déterminer le k optimal consistera donc à calculer un partitionnement avec différents k (p. ex. de k=2, k=3, etc. jusqu'à un maximum arbitraire, par exemple k=10) et d'observer l'évolution du critère de Calinski :



Le meilleur k sera celui avec la valeur de Calinski la plus élevée. Dans la figure précédente, le nombre de groupes optimal serait de 2.

Notez que ces groupes ne sont pas nécessairement les plus faciles à interpréter biologiquement ou écologiquement. Vous pouvez (fortement

recommandé!) utiliser votre jugement à cette étape, particulièrement si les valeurs de Calinski sont très rapprochées entre certaines valeurs de k.

Notez aussi que cet indice est une façon parmi d'autres pour déterminer le k optimal. Il en existe aussi d'autres si vous fouillez un peu. Le critère de Calinski est le plus utilisé. Il est néanmoins peu fiable lorsque la taille des groupes est très inégale. Il faut dans ces cas se fier à notre jugement plutôt que de se fier aveuglément au critère de Calinski.

26.2.4. La part du hasard

Comme pour la NMDS, l'algorithme du K-means démarre avec une configuration aléatoire. C'est donc dire que si vous relancez l'algorithme une seconde fois, il pourrait ne pas nécessairement trouver exactement le même résultat.

Il est généralement recommandé de lancer l'algorithme avec au moins 20 configurations de départ différentes (idéalement 50) et de conserver le meilleur résultat (i.e. la configuration avec le critère de Calinski le plus élevé). Ne vous en faites pas, la fonction R fera ces 50 répétitions automatiquement pour vous.

26.2.5. Labo : K-means en choisissant le k avant de débuter

Nous allons maintenant essayer d'appliquer la technique des K-means au jeu de données sur les manchots de Palmer, afin d'évaluer si l'algorithme réussit à séparer les espèces en se basant uniquement sur les mesures morphologiques.

Comme pour l'ACP, nous préparerons deux tableaux, soit un avec les variables quantitatives, et un autre avec les informations complémentaires que nous pourrons utiliser pour valider la classification.

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(palmerpenguins)

pour_regroupements <-
  penguins |>
  drop_na(bill_length_mm:body_mass_g)

infos_complementaires <-
  pour_regroupements |>
  select(species, island, sex, year)

pour_regroupements <-
  pour_regroupements |>
  select(bill_length_mm:body_mass_g)
```

Remarquez que dans une vraie application, il aurait été prudent d'explorer nos données avant de commencer, mais comme nous connaissons bien le jeu de données `penguins`, inutile de répéter ce travail ici.

Ensuite, ajustons un premier modèle de K-means, avec k=3 pour voir à quoi les résultats d'un K-means peuvent ressembler :

```
library(vegan)
```

```
Loading required package: permute
```

```
Loading required package: lattice
```

```
This is vegan 2.6-8
```

```
exemple <-
  pour_regroupements |>
  scale() |>
  kmeans(centers = 3, nstart = 50)
```

Il y a plusieurs choses importantes dans ce bout de code. D'abord, il faut centrer-réduire chacune de nos variables avant le calcul (la fonction `scale`) afin d'éviter les problèmes d'échelles, puisque le k-means travaille avec la distance euclidienne. Aussi, la fonction à utiliser pour faire le calcul se nomme `kmeans`. Elle attend 3 arguments, soit le tableau de données (implicite dans la chaîne), le nombre de groupes (`k`) et le nombre de configuration de départ à essayer (`nstart`).

Voyons maintenant ce que contient notre objet de résultats :

```
exemple
```

```
K-means clustering with 3 clusters of sizes 123, 87, 132

Cluster means:
  bill_length_mm bill_depth_mm flipper_length_mm
1      0.6562677    -1.0983711         1.1571696
```

```
2      0.6600059     0.8157307        -0.2857869
3     -1.0465260     0.4858415        -0.8899121
  body_mass_g
1   1.0901639
2  -0.3737654
3  -0.7694891

Clustering vector:
  [1] 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 2 3 2 3 3 3 3 3 3
 [26] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 2 3
 [51] 3 3 2 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 2 3 3 3 2 3 2
 [76] 3 3 3 2 3 2 3 3 3 3 3 3 3 3 3 2 3 3 3 2 3 3 3 2 3
[101] 2 3 3 3 3 3 3 3 2 3 2 3 2 3 2 3 3 3 3 3 3 3 3 2 3 3
[126] 3 3 3 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[151] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[176] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[201] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[226] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[251] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
[276] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 2 2 2 2
[301] 2 2 2 2 3 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[326] 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2

Within cluster sum of squares by cluster:
[1] 143.1502 112.9852 122.1477
 (between_SS / total_SS =  72.3 %)

Available components:

[1] "cluster"      "centers"      "totss"
[4] "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
```

On apprend d'abord dans ces sorties que la technique a produit 3

groupes (comme on avait demandé), qui contiennent 123, 87 et 132 observations chacun.

La section *Cluster means* nous informe de la moyenne de chacune des variables pour chacun des groupes. On voit par exemple que les manchots du premier groupe ont des becs plus longs (0,65) et moins épais (-1,09) que la moyenne.

Rappelez-vous qu'on ne parle plus ici de mm, puisque les données ont été centrées-réduites. On parle plutôt de 0,65 écart-type au-dessus de la moyenne.

La section *Clustering vector* nous informe ensuite du groupe auquel appartiennent chacune des observations après le classement.

Les dernières lignes nous informent enfin de certaines statistiques sur le partitionnement de la variance que nous n'utilisons pas ici.

Voyons maintenant comment on pourrait se faire un petit graphique pour voir comment se répartissent nos groupes. La clé pour y arriver est d'ajouter à notre tableau de données une colonne de groupe, que l'on extrait de notre objet de résultats. On peut ensuite utiliser cette information de groupe pour colorer les éléments dans notre graphique.

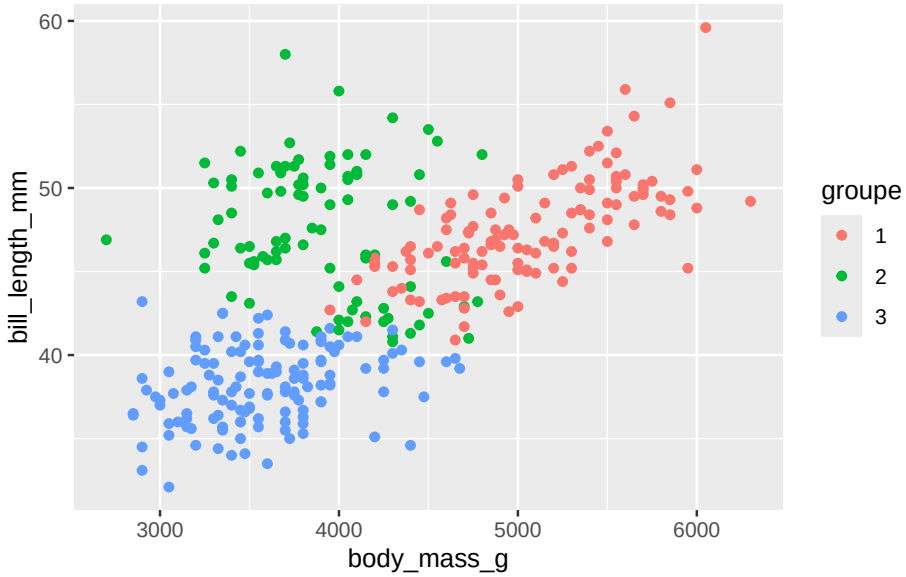
```
pour_regroupements |>
  mutate(groupe = as_factor(exemple$cluster)) |>
  ggplot(aes(x = body_mass_g, flipper_length_mm)) +
  geom_point(aes(color = groupe))
```

Remarquez que l'on doit utiliser la fonction `as_factor` pour emballer la colonne de groupe, car sinon, R l'aurait interprétée comme une variable quantitative puisque ce sont des chiffres.

Remarquez aussi que sur ces 2 variables, on voit bien la différence entre le groupe 1 et les groupes 2 et 3, mais que ces derniers sont très difficiles à distinguer.

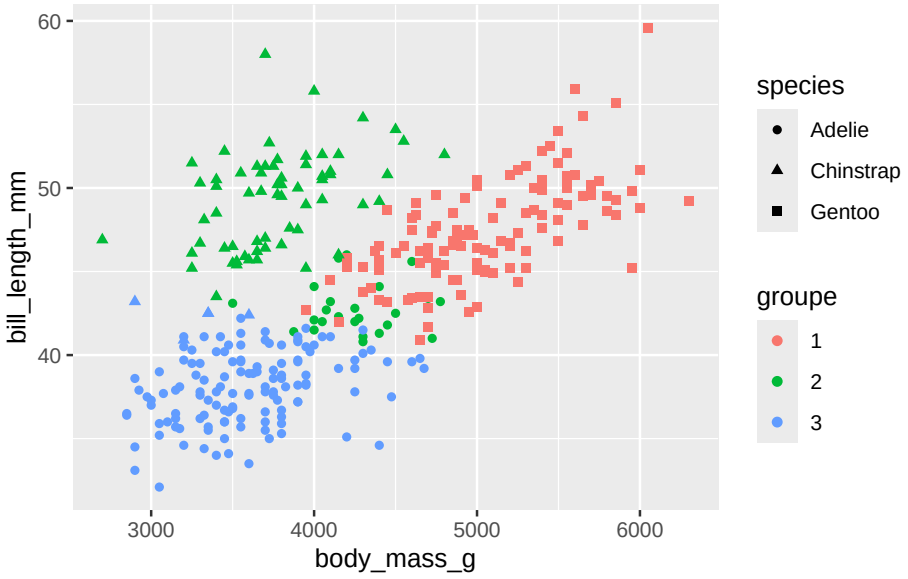
En se fiant sur les sorties numériques, on pourrait par exemple aller explorer la longueur du bec / poids du corps pour mieux illustrer les 3 groupes :

```
pour_regroupements |>
  mutate(groupe = as_factor(exemple$cluster)) |>
  ggplot(aes(x = body_mass_g, bill_length_mm)) +
  geom_point(aes(color = groupe))
```



Enfin, on pourrait aller voir à quel point le k-means a réussi à retrouver la séparation originale entre les 3 espèces, pour laquelle il n'avait pas l'information pour se valider.

```
pour_regroupements |>
  bind_cols(infos_complementaires) |>
  mutate(groupe = as_factor(exemple$cluster)) |>
  ggplot(aes(x = body_mass_g, bill_length_mm)) +
  geom_point(aes(color = groupe, shape = species))
```




```
table(as_factor(exemple$cluster),
 ↪  infos_complementaires$species)
```

```
   Adelie Chinstrap Gentoo
1       0         0    123
2      24        63      0
3     127         5      0
```

On voit que tous les Gentoo se sont retrouvés dans le groupe 1. La majorité des Adélie dans le groupe 3 et la majorité des manchots Chinstrap dans le groupe 2.

Évidemment ce n'est pas parfait, mais l'algorithme des k-means n'est pas fait pour reclasser parfaitement les groupes. Il est fait pour trouver les groupes les plus naturels dans nos données...

26.2.6. Labo : K-means pour sélectionner le meilleur nombre de groupes

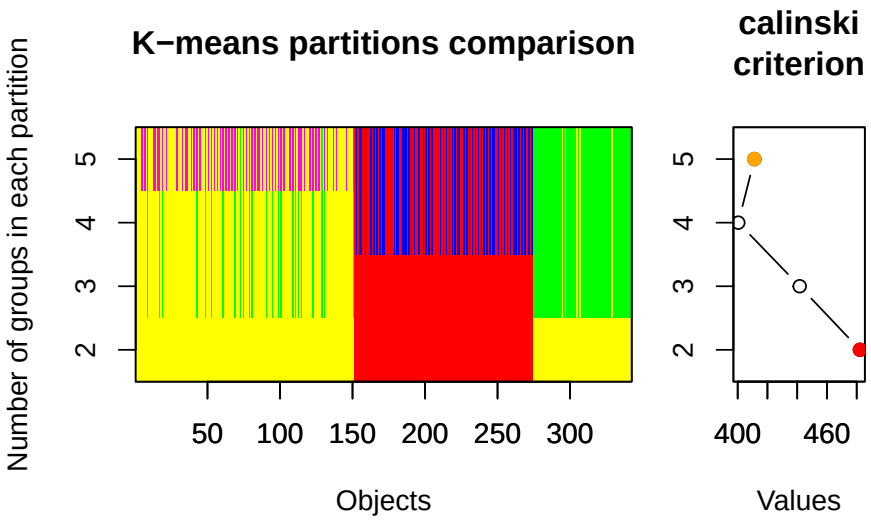
Comme expliqué plus haut, dans la vraie vie, on ne saura pas toujours combien de groupes contiennent nos données. Par exemple, si on applique un k-means sur les données physicochimiques d'une série de lacs, on ne saura pas d'avance en combien de groupes nos lacs devraient être séparés.

Pour déterminer le nombre de groupes idéal dans un jeu de données, la fonction pour y arriver provient aussi de la librairie **vegan** et se nomme **cascadeKM**. Elle fonctionne un peu comme **kmeans**, mais on lui fournit un **k** minimum et un **k** maximum, et elle essaie elle-même tous les k possibles dans cet intervalle. Si on voulait par exemple essayer entre 2 et 5 groupes, on l'utiliserait comme ceci :

```
plusieurs <-
  pour_regroupements |>
  scale() |>
  cascadeKM(inf.gr = 2, sup.gr = 5, iter = 50)
```

Par la suite, on peut explorer les valeurs du critère de Calinski pour ces résultats à l'aide de la fonction `plot` :

```
plot(plusieurs)
```



La partie de gauche ferait une excellente peinture abstraite (!), mais elle nous montre surtout à quel groupe (la couleur) appartient chacune des observations (en X) selon la valeur de k (en Y). La partie de droite nous montre quant à elle le critère de Calinski calculé (en X) pour chacune des valeurs de k (en Y). On y voit que selon le critère de Calinski, le nombre de groupes idéal serait de 2, puisque c'est la valeur la plus élevée.

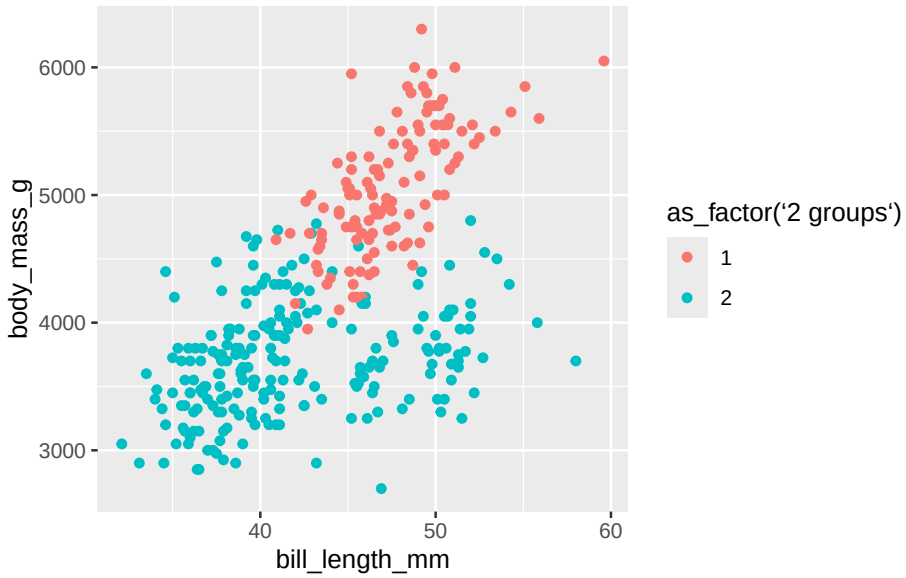
On peut aussi accéder à ces résultats en chiffres, en tapant le nom de notre objet de résultats dans la console :

```
plusieurs$results
```

```
          2 groups 3 groups 4 groups 5 groups
SSE       564.0535 378.2832 299.5212 231.9172
calinski 482.1915 441.6771 400.4100 411.2587
```

Si on voulait explorer visuellement à quoi ressemble la classification à 2 groupes, on pourrait ajouter à notre tableau de données le résultat de la classification, et l'utiliser dans notre graphique pour choisir la couleur comme dans le graphique précédent. Il faudrait par contre ici utiliser la fonction bind_cols, puisque l'on connectera plusieurs colonnes de résultats à la fois.

```
pour_regroupements |>
  bind_cols(
    as_tibble(plusieurs$partition)
  ) |>
  ggplot(aes(bill_length_mm, body_mass_g)) +
  geom_point(aes(col = as_factor(`2 groups`)))
```

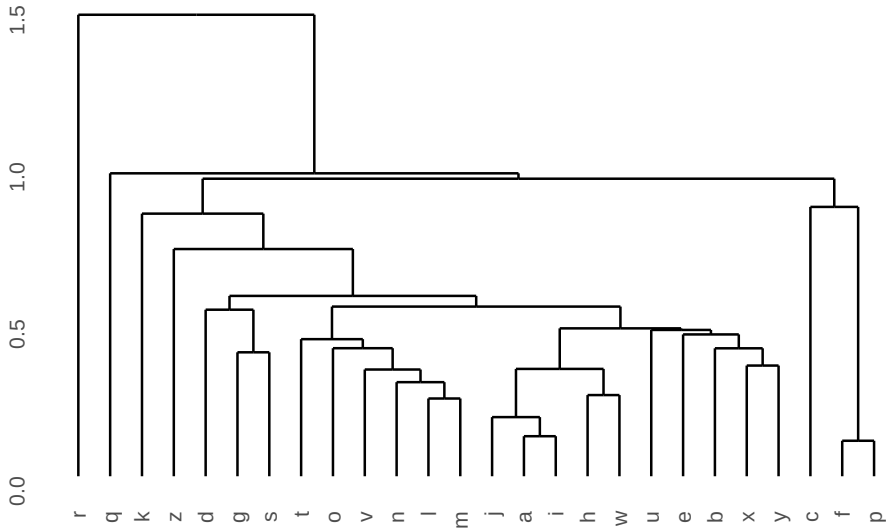


Remarquez dans le code précédent que nous avons dû utiliser les backticks (‘) pour accéder au nom de la colonne désirée, soit celle pour 2

groupes, puisque son nom contient des espaces.

26.3. La classification hiérarchique

Pour comprendre la classification hiérarchique, observons-en d'abord la sortie, que l'on nomme un dendrogramme :



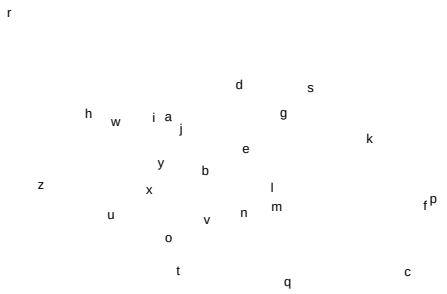
Dans cette figure, l'axe des X représente chacune des observations (dans un ordre quelconque; dans un cas réel, nous aurions affiché le nom de chaque observation au bout de la ligne...). L'axe des Y est un axe de distance (p. ex. euclidienne) entre les observations ou les groupes.

Plus la connexion entre deux observations est haute sur l'axe des Y, plus ces observations (ou ces groupes) sont différentes les unes des autres. On obtient donc une vue d'ensemble des ressemblances entre nos observations.

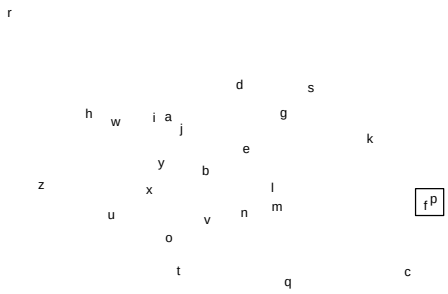
Notez que ultimement, toutes les observations sont connectées.

26.3.1. Fonctionnement de l'algorithme

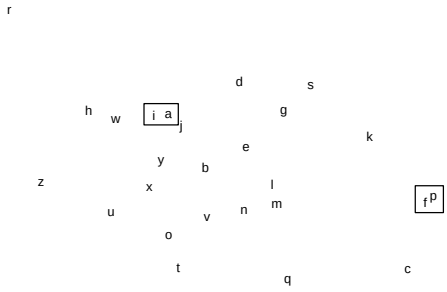
Au début de la procédure de partitionnement hiérarchique, l'algorithme considère chaque observation comme un groupe, contenant une seule observation.



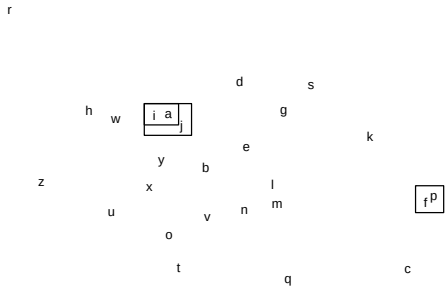
(a) Avant de débuter



(a) Connecter p et f



(a) Connecter i et a



(a) Connecter j à i+a

L'algorithme va tout d'abord calculer une matrice de distances entre ces groupes (exactement comme au Chapitre 22).

Il va ensuite trouver la paire de groupes la plus proche et les connecter.

Il recalcule ensuite une nouvelle matrice de distances entre les groupes, puis reconnecte les deux groupes les plus proches.

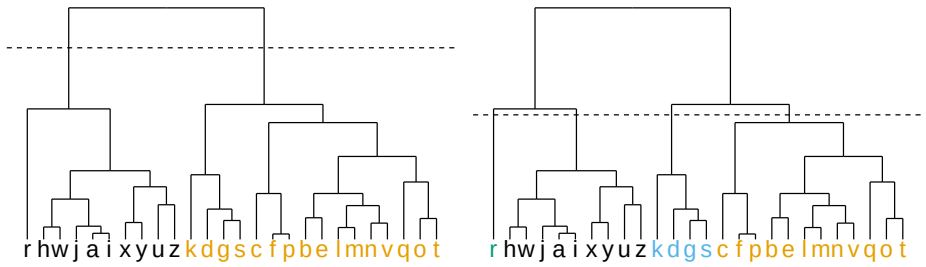
Il continue ainsi jusqu'au moment où toutes les observations sont connectées.

Autrement dit, le dendrogramme est construit par le bas, et on remonte vers le haut.

26.3.2. Combien de groupes

Comme pour le K-means, on peut ensuite se demander : oui mais, combien de groupes contient notre jeu de données finalement?

Encore une fois, il n'y a pas de réponse absolue à cette question.



La façon habituelle de faire est de choisir un seuil de distance à partir duquel on considère les groupes comme différents. Dans la partie de gauche de la figure précédente, en établissant notre seuil de distance à 4, on obtient 2 groupes, alors qu'en le plaçant à 2,6, on en obtient 4.

Plus on choisit un seuil de distance élevé, moins notre solution contiendra de groupes.

Il existe certaines règles du pouce (semblables au critère de Calinski) pour choisir le nombre final de groupes, mais cette tâche est souvent effectuée à l'œil, de façon arbitraire.

26.3.3. Le choix de la mesure d'attachement

Un point sur lequel nous avons glissé au moment de définir l'algorithme est de savoir comment on mesure la distance entre deux groupes. Il faut par contre bien faire la nuance entre deux concepts.

Le terme distance, dans ce contexte, est consacré à la mesure de distance, comme décrit au Chapitre 22 (euclidienne, Bray-Curtis, etc.)

Le terme attachement (*linkage*) définit par quels points on mesure la distance entre deux groupes. Est-ce qu'on calcule la distance par le point le plus loin du groupe, le plus proche, le milieu du paquet, etc?

Comme d'habitude, il n'existe pas de façon magique, meilleure que toutes les autres pour faire cette tâche. Il en existe des dizaines, mais nous en verrons 3 :

- La méthode complète (*complete*) se base sur la distance entre les deux points les plus éloignés
- La méthode simple (*single*) se base sur la distance entre les deux points les plus proches
- La méthode moyenne (*average*) se base sur la distance moyenne entre toutes les paires de points

En général, la méthode simple aura tendance à attacher les observations une à une, formant des groupes qui ne sont pas très compacts.

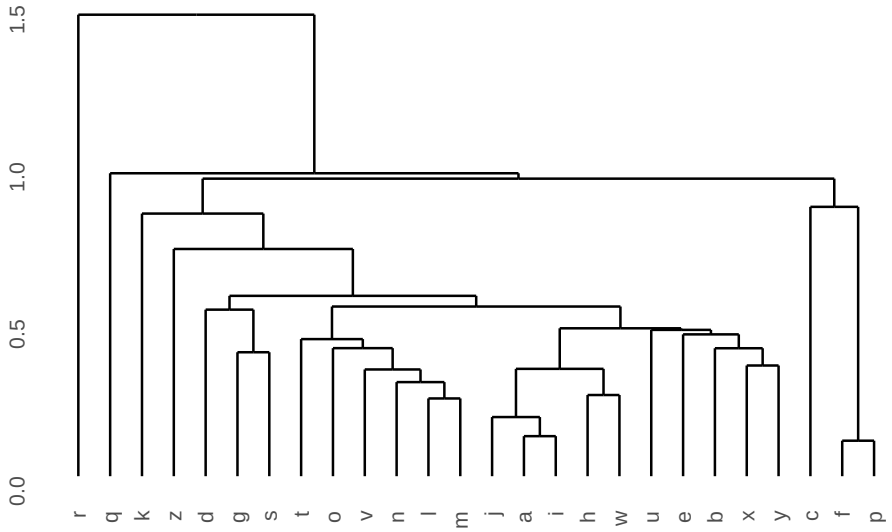


Figure 26.10.: La méthode simple (single)

La méthode complète créera des groupes compacts, mais peu séparés les uns des autres.

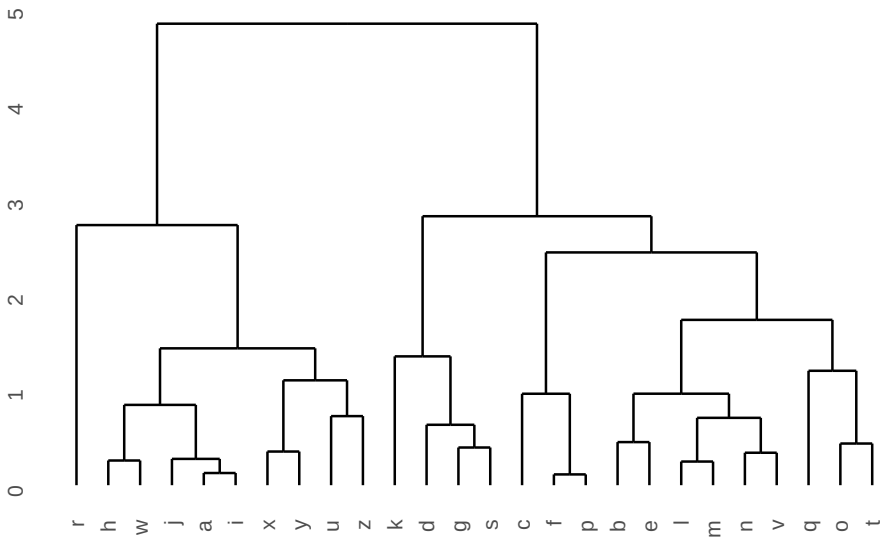


Figure 26.11.: La méthode complète (complete)

La méthode moyenne présente en général un bon compromis, formant des groupes à la fois relativement compacts et espacés.

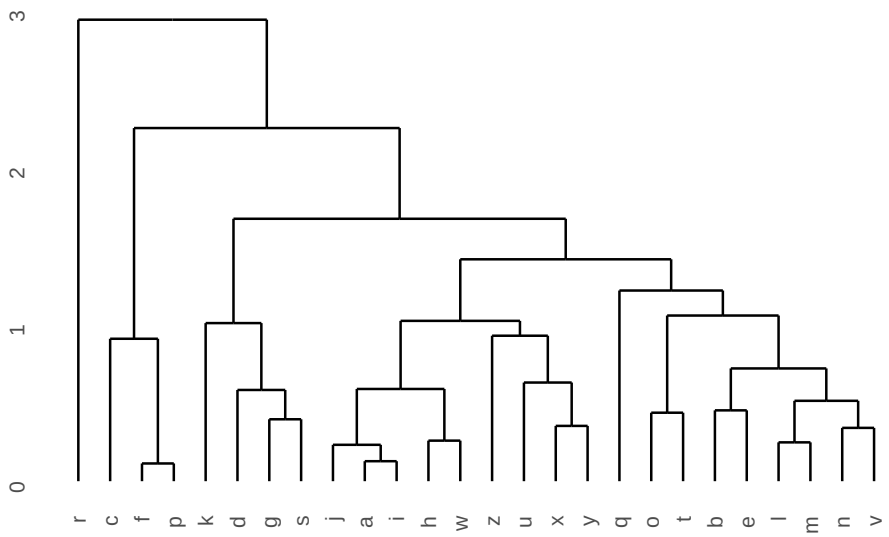
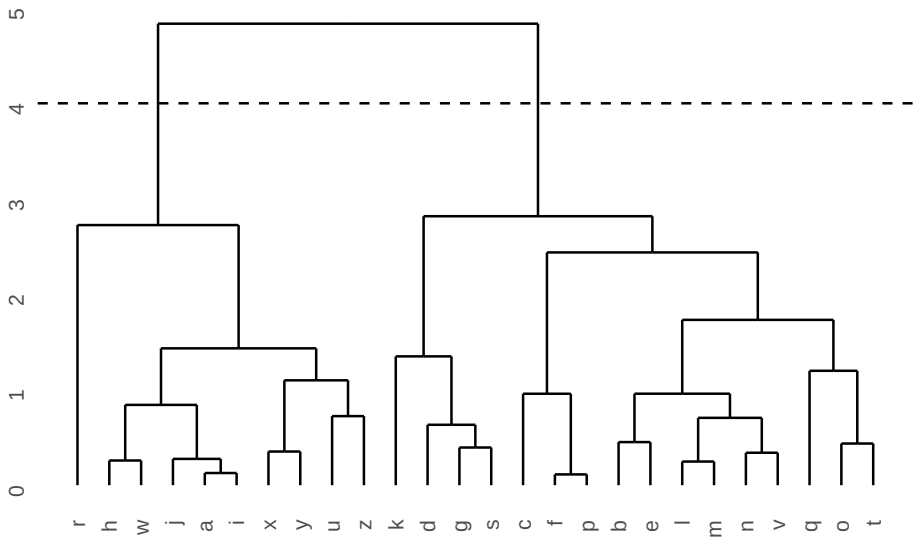


Figure 26.12.: La méthode moyenne (average)

Par contre, les méthodes simple et complète offrent une interprétation simple du point de coupure, que la méthode moyenne n'offre pas. Si l'on place par exemple notre point de coupure à une distance euclidienne de 4, comme ceci :



Dans la méthode simple, on peut affirmer que chaque point dans un groupe possède au moins un autre point à une distance de moins de 4.

Dans la méthode complète, on peut affirmer que tous les points d'un groupe sont à moins de 4 les uns des autres.

Dans la méthode moyenne, il n'y a pas d'interprétation de ce genre possible.

Aussi, contrairement aux deux autres méthodes, la méthode moyenne est sensible aux transformations. Si vous transformez vos données, elle pourrait vous donner un résultat différent.

26.3.4. Le choix de la mesure de distance

Contrairement au K-means qui fonctionne obligatoirement avec la distance euclidienne, la classification hiérarchique nous permet de choisir la mesure de distance désirée. Tout ce que nous avons vu au Chapitre 22 s'applique donc ici :

- si vos données sont continues, on utilise la distance euclidienne,
- si vos données sont des décomptes d'individus, on utilise la distance de Bray-Curtis et
- si vous avez des données de présence-absence, vous pouvez utiliser la distance de Jaccard ou celle de Bray-Curtis.

Il est aussi important de s'interroger sur les différences de variance et de standardiser les données si nécessaire.

26.3.5. Labo : La classification hiérarchique

Comme nous avons vu plus haut, le dendrogramme du partitionnement hiérarchique affiche chacune des obsevations au bas du graphique. Pour que notre graphique demeure lisible dans les pages de ce livre, nous allons piger un petit échantillon aléatoire dans le tableau de données des manchots et travailler uniquement avec ce dernier. Il faut aussi penser de réduire notre tableau d'informations supplémenatires pour qu'il contienne lui aussi uniquement les lignes choisies.

```
echantillon <-
  sample(1:nrow(pour_regroupements),size = 30)
echantillon
```

```
 [1]  50 323 147 114  23 322  42  99 334 341 111 328
[13] 204 320 315 121 140 267  88 216 159 325 106  16
[25] 101 102  76 227  59 284
```

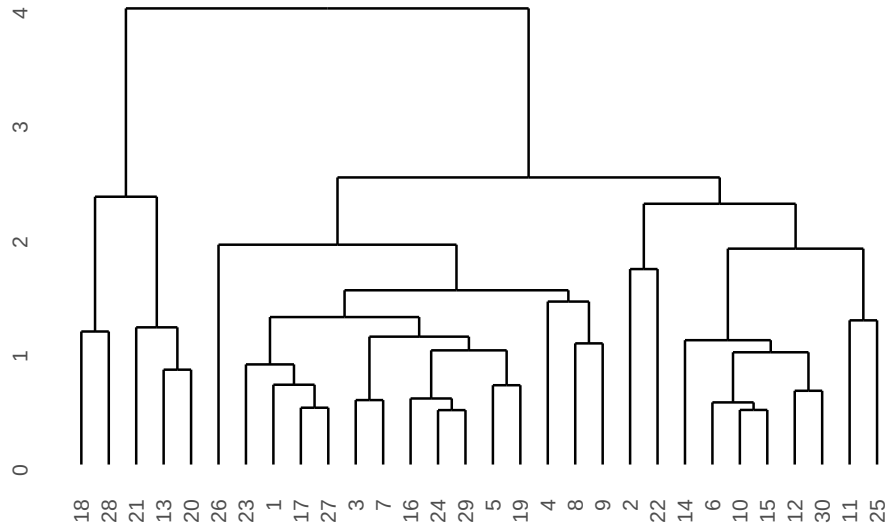
```
pour_hclust <-
  pour_regroupements |>
  slice(echantillon)

infos_pour_hclust <-
  infos_complementaires |>
  slice(echantillon)
```

La fonction pour construire un dendrogramme de classification hiérarchique se nomme `hclust` (*Hiearchical CLUSTering*), et provient aussi de la librairie `vegan`. Par contre, cette fonction ne s'attend pas à recevoir un tableau de données brutes, mais plutôt une matrice de distances entre les observations. C'est pourquoi nous allons construire une chaîne, dans laquelle nous allons aussi utiliser la fonction `vegdist`, pour calculer une matrice de distances. Comme nous utiliserons la distance euclidienne, il faut aussi centrer et réduire nos données avant de procéder au calcul :

```
dendro <-
  pour_hclust |>
  scale() |>
  vegdist("euclidian") |>
  hclust(method = "average")
```

On peut ensuite visualiser notre dendrogramme à l'aide de la fonction `ggdendrogram` de la librairie `ggdendro` :

```
library(ggdendro)
ggdendrogram(dendro)
```



Il existe aussi une fonction de base (`plot`) pour afficher le dendrogramme, mais cette dernière n'est pas basée sur `ggplot2`.

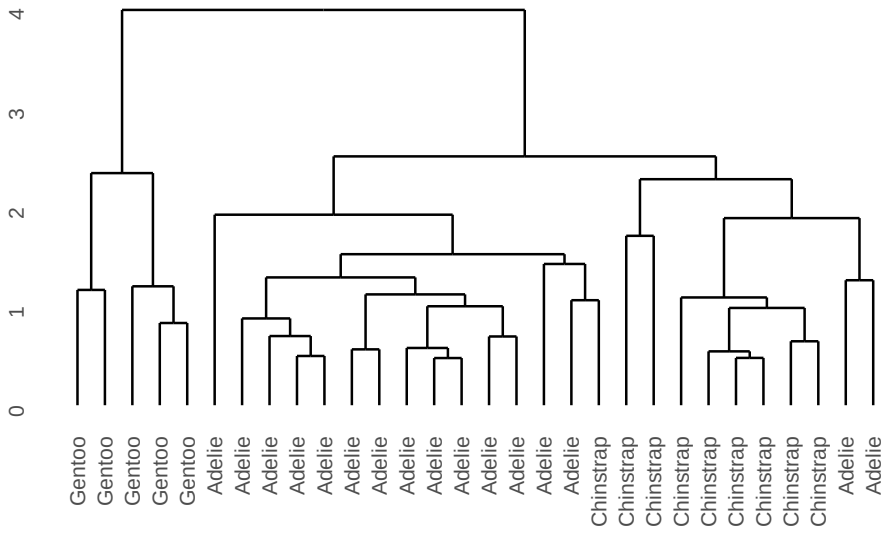
On pourrait aussi inscrire au bas du graphique une information plus intéressante que le numéro de la ligne de chaque obsevation, par exemple, l'espèce de chaque individu.

Pour se faire, il faut ajouter manuellement l'étiquette à chacune des lignes de notre dendrogramme. Comme l'objet `dendro` n'est pas un tableau de données, on ne peut PAS utiliser la fonction `mutate` :

```
dendro$labels = infos_pour_hclust$species
```

On peut ensuite afficher le dendrogramme :

```
ggdendrogram(dendro)
```

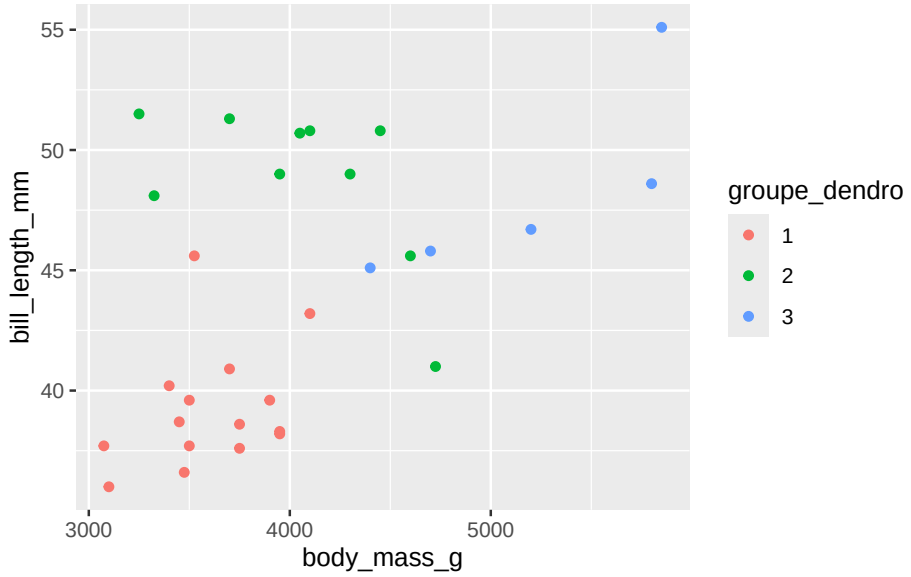


La première séparation, une distance d'environ 4, sépare 5 manchots Gentoo du reste du groupe. La deuxième séparation, à une hauteur d'environ 2,5 sépare les manchots Adélie des manchots Chinstrap et d'un groupe de deux manchots Adélie un peu plus différents à droite.

Comme discuté ci-haut, le partionnement hiérarchique ne fournit pas de nombre optimal de groupes. Ce serait à nous, à l'oeil, de déterminer le meilleur nombre de groupes.

Si on voulait explorer la classification donnée par 3 groupes dans ce dendrogramme, on peut utiliser la fonction `cutree`, et lui fournir le nombre de groupes désiré. La fonction va elle-même aller couper l'arbre et nous dire dans quel groupe tomberait chacune de nos observations.

```
trois <- cutree(dendro, k=3)
pour_hclust |>
  mutate(groupe_dendro = as_factor(trois)) |>
  ggplot(aes(body_mass_g, bill_length_mm)) +
  geom_point(aes(color = groupe_dendro))
```



26.4. Exercice : Les analyses de regroupements

À partir de la base de données de météo des villes utilisée au Chapitre 23 [1], nous allons maintenant appliquer les techniques ci-haut pour tenter

[1] https://drive.google.com/file/d/1ZLeRkJl2MmJNFZjEgJljul9tUUIJqsyr/view?usp=sharing

de voir si on trouve des regroupements naturels entre les villes, basé sur la météo.

Comme au Chapitre 23, éliminez la variable de neige avant de commencer les analyses puisque celle-ci contient beaucoup de valeurs manquantes.

Appliquez dans un premier temps la technique des K-means pour déterminer le meilleur nombre de groupes que pourrait contenir ce jeu de données en vous basant sur le critère de Calinski.

Ensuite, illustrez ce partitionnement à l'aide d'un graphique où on retrouvera en X la température maximum moyenne et en Y les précipitations en mm.

Sur quel critère semble s'être basé le K-means pour séparer nos groupes? Géographiquement, qu'ont en commun les villes dans le groupe de gauche du graphique?

Ensuite, appliquez la technique du partitionnement hiérarchique au même jeu de données et produisez le graphique du dendrogramme.

Quelles sont les villes appartenant au petit groupe se distinguant le plus du reste des villes?

En général, appréciez-vous les groupes formés par le dendrogramme? Vous semblent- ils naturels?

26.5. Conclusion

La question qui vous reste en tête à ce moment-ci est probablement : OK, parfait Charles, on a deux techniques pour explorer les regroupements, mais laquelle on utilise dans quelle situation?

En général, la technique des K-means sera utilisée lorsque vous connaissez à l'avance le nombre de groupes, mais que vous ne connaissez pas

les règles exactes qui permettraient d'automatiser le processus de classification. Cette technique est très utilisée par exemple pour séparer les différents types de terrain dans les applications de télédétection. Le K-means est plutôt une technique d'apprentissage automatique (*machine learning*) où on laisse l'ordinateur démêler les données pour nous.

Le partitionnement hiérarchique quant à lui est surtout utilisé dans des applications d'écologie des communautés, où on tente de comprendre quels sites se ressemblent par leur composition en espèces. C'est une technique plus exploratoire, mais qui permet aussi de meilleures interprétations biologiques des sorties que le K-means, particulièrement grâce à l'aspect hiérarchique des sorties.

Ici, pour des raisons pédagogiques, je vous ai montré les deux techniques sur un même problème, mais dans la vraie vie, on en choisit une ou l'autre, selon le travail à faire.

partie V.

Le modèle linéaire

27. La régression multiple

27.1. Introduction

Dans notre survol des tests statistiques, nous avons exploré une variété de méthodes pour tester le lien entre deux variables. Certaines de ces méthodes ne faisaient que tester pour une association (par exemple la corrélation et le khi-carré), mais d'autres assumaient un lien de cause à effet entre les deux variables (régression, ANOVA). Dans tous les cas, nous étions cependant limitées à étudier une seule cause à la fois. Pourtant, dans la vraie vie, particulièrement en écologie, il y a rarement une seule cause à un phénomène. Pensez simplement au poids d'un poisson. Ce dernier peut être influencé entre autres par son âge (croissance), la quantité de nourriture disponible, son bagage génétique, la saison, etc.

La régression multiple est l'outil par lequel nous pourrons étudier simultanément l'effet de plusieurs causes sur notre variable expliquée. La régression multiple telle que montrée ici nous permet de le faire avec plusieurs variables quantitatives, et nous verrons au Chapitre 29 comment on peut aussi y intégrer des variables qualitatives (oui oui!).

Comme pour le chapitre sur la régression linéaire, nous verrons d'abord la théorie et ensuite un exemple concret dans la partie laboratoire afin d'alléger la présentation de ce chapitre, qui pourrait facilement devenir lourde.

27.2. Modèle statistique et interprétation

Nous avons vu au Chapitre 18, que l'équation de la régression simple peut être écrite comme ceci :

$$y = b_0 + b_1 x$$

Où b_0 et b_1 sont respectivement les valeurs associées à nos paramètres d'ordonnée à l'origine et de pente.

La régression multiple fonctionne exactement de la même façon, mais peut compter autant de pentes que nous voulons étudier de variables dans notre modèle. La façon générale de décrire cela est donc comme ceci :

$$y = b_0 + b_1 x_1 + b_2 x_2 ... + b_p x_p$$

Si nous avions quatre variables explicatives dans notre modèle, l'équation serait par exemple comme ceci :

$$y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_4$$

L'interprétation de ces paramètres se fait un peu comme dans la régression linéaire simple, mais doit se faire de façon plus nuancée.

b_0 est encore notre ordonnée à l'origine. On se rappelle que dans la régression linéaire, on interprétait l'ordonnée à l'origine comme étant la hauteur où l'on était sur l'axe des Y quand X=0. Ici, puisque nous avons plusieurs variables explicatives, l'ordonnée à l'origine devient notre valeur de Y, lorsque toutes les variables explicatives (x_1, x_2, x_3, etc.) sont à zéro.

Chacun des autres paramètres (b_1, b_2, b_3, etc.) sont encore des pentes, mais on parle maintenant de **pentes partielles**. On se rappelle que dans

la régression linéaire simple, on interprétait une pente comme le changement en Y pour un changement de une unité en X. Ici, dans la régression multiple, b_1 est le changement en Y pour un changement d'une unité de X_1 en assumant que toutes autres variables étudiées (x_2, x_3, etc.) restent constantes.

Si jamais les variables explicatives que vous désirez étudier sont parfaitement indépendantes (i.e. non-corrélées), vous pouvez ignorer cette nuance dans l'interprétation des pentes partielles. Par contre, à moins de travailler uniquement en laboratoire où vous contrôlez précisément toutes vos variables, dans la vraie vie d'un biologiste, il faut habituellement être attentifs à ce genre de détail. Il est important de comprendre que la relation décrite par chacun des paramètres ne sera pas nécessairement observable directement sur le terrain, dépendant de la relation qui existe entre les variables expliquées.

Et enfin, la question qui vous reste peut-être en tête à ce point : jusqu'à combien de variables je peux mettre dans mon modèle? Cette limite changera en fait d'une analyse à l'autre, puisque le nombre de variables que vous pouvez entrer dans le modèle dépend du nombre d'observations que vous avez recueillies. Vous vous rappelez peut-être du concept de degrés de liberté (voir Chapitre 10)? Dans un modèle de régression multiple nous ajustons, comme suggéré dans les formules précédentes, autant de paramètres que nous avons de variables + un pour l'ordonnée à l'origine. Pour qu'un modèle statistique s'ajuste, il doit toujours lui rester au moins 1 degré de liberté. Donc, si vous avez 7 observations dans votre tableau de données, vous pouvez entrer jusqu'à 5 variables dans votre modèle. Si vous avez 1000 observations, vous pouvez ajuster jusqu'à 998 variables. Évidemment, ce genre de modèles sera très incertain, puisque l'on recommande en général d'avoir au moins 25, idéalement 40, observations par paramètre que l'on veut estimer.

27.3. Tests statistiques

Bien que le modèle de régression linéaire, avec ses paramètres et intervalles de confiance soit directement interprétable, il pourra arriver que l'on vous demande si votre modèle est statistiquement significatif, et si l'effet de certaines variables est significativement différent de zéro.

Pour tester si le modèle entier est significatif, il faut simplement appliquer un test de F avec au numérateur les carrés moyens associés au modèle, et au dénominateur les carrés moyens associés aux résidus :

$$F = MS_{rgression}/MS_{rsidus}$$

Ce test pourrait aussi s'utiliser avec la régression linéaire simple, et vous donnera exactement la même valeur de p que celle du test de t associé à la pente expliqué au Chapitre 18.

Ce qu'il faut bien comprendre avec ce test, c'est qu'il nous informe que notre modèle, au total, est significatif ou non. Par contre, il pourrait arriver que parmi les variables que nous avons incluses dans le modèle, que certaines aient un effet significatif, mais que d'autres non. Il existe deux stratégies équivalentes pour déterminer si une pente individuelle est significative ou non.

La première est ce que l'on appelle les **tests de F partiels**. L'idée derrière ces tests est de regarder l'apport d'une variable individuelle, par rapport au bruit dans notre modèle. Si l'apport de la variable est grand par rapport au bruit, on considère cette variable comme significative. Pour déterminer l'apport d'une variable, il faut commencer par ajuster un modèle complet, regarder la variance expliquée par ce modèle et y soustraire la variance expliquée par un modèle simplifié, où notre variable d'intérêt n'est pas présente. On fait ensuite le ratio entre cette

différence de variance et la variance des résidus pour trouver notre valeur de F, qui qui doit être comparée à la distribution théorique pour n-1 degrés de liberté.

La deuxième technique est celle des tests de t, telle que montrée au Chapitre 18. Autrement dit, on calcule une valeur de t en faisant le ratio entre la pente partielle et son erreur type, et on compare ensuite cette valeur trouvée à une distribution de t théorique pour les degrés de liberté de notre modèle (n moins le nombre de paramètres).

Les deux stratégies (test de F partiel et test de t) vous donneront toujours exactement la même valeur de p. Les tests de t ont l'avantage d'être plus simples à calculer et sont fournis directement dans les sorties de R. Par contre, l'approche des tests de F partiels est beaucoup plus flexible. Elle permet de tester des groupes entiers de variables d'un coup. On peut donc plus facilement tester des hypothèses, par exemple est-ce que les variables climatiques ont un effet significatif, est-ce que les variables physico-chimiques ont un effet significatif, etc.

27.4. Comparer la taille des effets

Comme nous en avons discuté dans le chapitre sur la régression linéaire, la mesure de la taille de l'effet d'une régression est la valeur du paramètre de pente. Plus ce chiffre est gros, plus la taille de l'effet est grande. On pourrait donc penser que pour savoir laquelle de nos variables a le plus grand effet dans une régression multiple, on peut simplement comparer les paramètres de pente entre eux... mais ce n'est malheureusement pas aussi simple.

La valeur du paramètre de pente dans une régression dépend grandement de l'échelle à laquelle nos données ont été mesurées. Si nous avons, par exemple, une régression dans laquelle nous tentons de prédire le nombre d'espèces d'oiseaux dans une parcelle, pour laquelle nous

avons mesuré le bruit ambiant et la surface de la parcelle, nous pourrions obtenir cette équation si la surface était mesurée en km^2 :

$$Oiseaux = 3 \times surface(km^2) - 5 \times bruit(db)$$

Et celle-ci si la surface était mesurée en m^2 :

$$Oiseaux = 30000 \times surface(m^2) - 5 \times bruit(db)$$

Avec la mesure en m^2, le bruit semble avoir un effet plus important que la surface. Avec les km^2, le bruit semble avoir un effet moins important que la surface. Vous voyez le genre de problèmes que ça peut occasionner?

Alors on ne peut jamais comparer l'ampleur des pentes Charles?

Évidemment il y a une solution! Nous l'avons même déjà évoqué au Chapitre 22 sur les matrices et les distances. Le truc est de ramener toutes nos variables à la même échelle pour les comparer. Il existe deux façons d'y arriver. Une que l'on peut appliquer après avoir ajusté notre modèle, et l'autre que l'on peut appliquer en amont, avant de lancer notre analyse.

La solution après-coup consiste à standardiser les valeurs de pentes que nous avons trouvées. Cette opération peut être décrite de façon formelle pour une variable j comme ceci :

$$b_j^* = b_j \frac{\sigma_{X_j}}{\sigma_Y}$$

Autrement dit, la pente standardisée est trouvée en multipliant la pente par le ratio entre les écart-types de la variable expliquée et de la variable explicative.

Si l'on retourne à notre exemple avec la surface en km^2, il faudrait donc aller calculer dans nos données l'écart-type de chacune des variables (par exemple $\sigma_{oiseaux} = 3$, $\sigma_{bruit} = 1$, $\sigma_{surface} = 2$).

La pente standardisée pour la surface est donc 3 x 2/3 = 2 et celle pour le bruit est de 5 x 1/3 = 1,67. On comprend maintenant que les deux variables ont un effet à peu près équivalent, mais que l'effet de la surface est un peu plus grand.

Remarquez que l'interprétation des pentes standardisée est un peu plus abstraite. Plutôt que dire que l'on augmente de 3 espèces d'oiseaux par kilomètre carré de surface, on doit maintenant dire que la richesse d'oiseaux augmente de 2 écart-types pour chaque changement de 1 écart-type de surface.

C'est pourquoi habituellement, on combine les deux valeurs dans notre interprétation. Celles à l'échelle originale pour bien saisir la taille des effets en termes concrets, et celles à l'échelle standardisée pour pouvoir bien comparer les pentes entre-elles.

Enfin, la deuxième façon de standardiser les pentes en sortie de notre modèle consiste à centrer et réduire chacune de nos variables avant de lancer l'analyse. Ainsi, tous les paramètres en sortie seront déjà standardisés. Il s'agit souvent de la façon la plus simple de procéder, puisque l'on peut souvent centrer et réduire l'ensemble de notre tableau de données avec une seule commande R.

27.5. Assomptions et validations

La régression multiple comporte les quatre mêmes assomptions que la régression linéaire : normalité des résidus, homogénéité de la variance des résidus, indépendance des observations et X est fixé par l'observateur. Elle en ajoute cependant une cinquième, qui consiste à dire que les variables explicatives ne doivent pas être corrélées entre

elles. C'est-à-dire qu'il doit y avoir absence de **colinéarité** entre les variables explicatives.

Comme pour la régression linéaire simple, les assomptions stipulant que les observations sont indépendantes et que X est fixé par l'observateur doivent être correctement réfléchies en amont de l'expérience. La validation de la normalité des résidus se fait aussi exactement de la même façon que dans la régression linéaire : on fait un histogramme des résidus.

Comme pour la régression linéaire, même si il ne s'agit pas d'une validation aussi formelle que celle effectuée sur les résidus après l'ajustement du modèle, c'est quand même toujours une bonne idée de regarder l'histogramme de chacune de nos variables et la linéarité des relations entre nos variables explicatives et la variable expliquée avant de se lancer dans la modélisation et d'appliquer les transformation nécessaires au besoin.

Là où les choses deviennent différentes/compliquées est pour la validation de l'homogénéité des résidus. Cette dernière doit maintenant être vérifiée non seulement en fonction des valeurs prédites, mais aussi en fonction de chacune des variables présentes dans notre modèle. On devra donc faire autant de nuages de points qu'il y a de variables dans notre modèle, en plus de celui des valeurs prédites. On veut s'assurer dans chacun de ces graphiques qu'il n'y a pas de patrons dans les résidus. Autrement dit, que notre modèle n'est pas meilleur ou pire à un extrême ou à l'autre des données. Si cela devait arriver, il faudra soit transformer nos données (ce qui peut souvent corriger des problèmes de variances inégales) ou se demander si il n'y a pas une variable importante qui a été négligée dans notre modèle.

27.6. La colinéarité

Et pour la colinéarité, comment peut-on valider son absence? D'abord, la première chose à savoir est que, particulièrement en écologique, vos variables explicatives seront toujours plus ou moins corrélées entre elles. Tester pour savoir s'il existe une corrélation entre vos variables est presque inutile parce qu'il en existera quasiment toujours une.

Avant d'aller plus loin, il importe de bien comprendre quels sont les effets de la colinéarité dans un modèle de régression. Il en existe deux majeurs. D'abord, les estimés de paramètre peuvent devenir instables. C'est-à-dire qu'enlever ou ajouter une seule observation au tableau de données pourrait complètement changer leur sens. On ne veut évidemment pas que ça nous arrive! L'autre problème qui survient en présence de colinéarité importante est que les intervalles de confiance vont être gonflés (i.e. plus larges). Comme nous en avons discuté plus haut, puisque nos tests statistiques pour les pentes sont basés sur notre mesure de certitude (si on parle du test de t), la colinéarité fera diminuer nos chances de trouver des pentes significatives. Cela peut même aller au point où le test de notre modèle entier serait significatif, mais qu'aucune pente individuelle ne le serait. Ce résultat plutôt surprenant est cependant tout à fait représentatif du problème : nos hypothèses sont bonnes, les deux variables ensemble ont une influence significative sur le phénomène, mais le fait qu'elles soient colinéaires ensemble nous empêche de distinguer leurs effets respectifs. On ne sait pas laquelle fait quoi.

Cela nous amène maintenant à parler de ce que la colinéarité n'affecte PAS. En effet, il est important de comprendre qu'avoir des variables colinéaires n'affecte en rien les prédictions du modèle, ni son r^2. La seule chose qui est affectée par la colinéarité est les estimés de pente et notre confiance en ces valeurs.

Pour savoir si la colinéarité est un problème dans notre modèle, on utilise une mesure nommée **VIF** (pour *Variance Inflation Factor*). Le calcul du VIF nous fournit un chiffre pour chacune de nos variables, nous informant de

combien cette variable ajoute de l'instabilité au modèle. Les statisticiens disent en général que l'on commence à avoir un problème de colinéarité quand le VIF d'une de nos variables dépasse 4. Vous trouverez parfois aussi la valeur de 10 ou même plus comme valeur seuil. Il s'agit en fait d'un aide à la décision (*rule of thumb*) et non d'un test statistique à proprement parler.

À l'aide du VIF, on peut cependant quantifier l'impact de la colinéarité sur l'incertitude de nos estimations de paramètres. Pour se faire, il faut savoir que la racine carrée de VIF nous donne un facteur par lequel l'erreur-type (une mesure d'incertitude) sera multipliée en comparaison d'un modèle sans colinéarité. Par exemple, si notre VIF est de 4, l'erreur-type est multipliée par 2, si le VIF est de 9, l'erreur-type sera multipliée par 3, etc. Cela nous donne un ordre de grandeur du problème.

Et on fait quoi si jamais on a une colinéarité importante dans notre modèle? Comme d'habitude, il n'y a pas de solution parfaite. La solution généralement recommandée est d'enlever du modèle une (ou plusieurs) variables avec un VIF élevé. Ainsi, les estimés de paramètres des variables restantes seront plus précis. Mais il faut garder en tête (et écrire dans notre rapport) que ces estimés de paramètres sont probablement aussi biaisés. La valeur élevée de VIF nous disait que le modèle était incapable de distinguer l'effet de ces variables corrélées. Il est donc possible que la variable que avons enlevée du modèle était celle qui en fait causait le phénomène. C'est pourquoi il est important, lorsque l'on supprime ainsi une variable de notre modèle à cause du VIF, de toujours conserver les variables pour lesquelles nos hypothèses étaient les plus solides et enlever celles pour lesquelles nos hypothèses étaient les moins fortes.

La deuxième solution possible est de remplacer les variables de notre analyse par les axes d'une ACP sur ces mêmes variables. Par exemple, si dans notre modèle on voulait mettre la longueur, la masse et l'envergure des oiseaux, on pourrait mettre ces variables dans une ACP et on pourrait utiliser le premier axe de l'ACP, sans doute représentatif de la taille globale de l'oiseau, comme variable dans notre modèle. Comme les axes de

l'ACP sont entièrement orthogonaux, nous n'aurons jamais de problème de colinéarité en les mettant dans notre modèle. Par contre, remarquez qu'ainsi, on complique aussi l'interprétation des pentes. Celle-ci ne serait plus ni en cm, ni en g, mais dans les unités conceptuelles de l'ACP.

La troisième façon de faire, que je privilégie souvent, mais qui n'est pas autant appliquée que les deux premières, est simplement de conserver les résultats ainsi dans le rapport, mais de bien discuter du problème de colinéarité. De dire directement que tel ou tel paramètre est probablement instable à cause de la colinéarité entre nos variables. Je trouve personnellement cette façon de faire plus utile pour la science, puisqu'elle met en relief les incertitudes de notre modèle, plutôt que de se créer de fausses certitudes en enlevant simplement une des variables du modèle. Cette façon de faire n'est pas particulièrement répandue, mais à mon avis, elle gagnerait à l'être.

27.7. Labo : la régression multiple

Nous reprendrons notre laboratoire du Chapitre 18 où nous avons étudié l'effet de la longueur des ailes sur le poids de manchots. Nous essayerons ici de comprendre plus globalement ce qui influence le poids des manchots, en ajoutant aussi la longueur et l'épaisseur du bec. Notre hypothèse étant que plus le bec et long et épais, plus le manchot pourra attraper de poissons, donc plus il sera lourd.

Avant de procéder à ce laboratoire, vous devrez installer les librairies `car` (comme dans voiture en anglais) et `visreg` (*VISualizing REGressions*), dans lesquelles nous pigerons des fonctions pour nous aider en cours de route.

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(palmerpenguins)
library(GGally)
```

```
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
```

```
library(car)
```

```
Loading required package: carData

Attaching package: 'car'

The following object is masked from 'package:dplyr':

    recode

The following object is masked from 'package:purrr':

    some
```

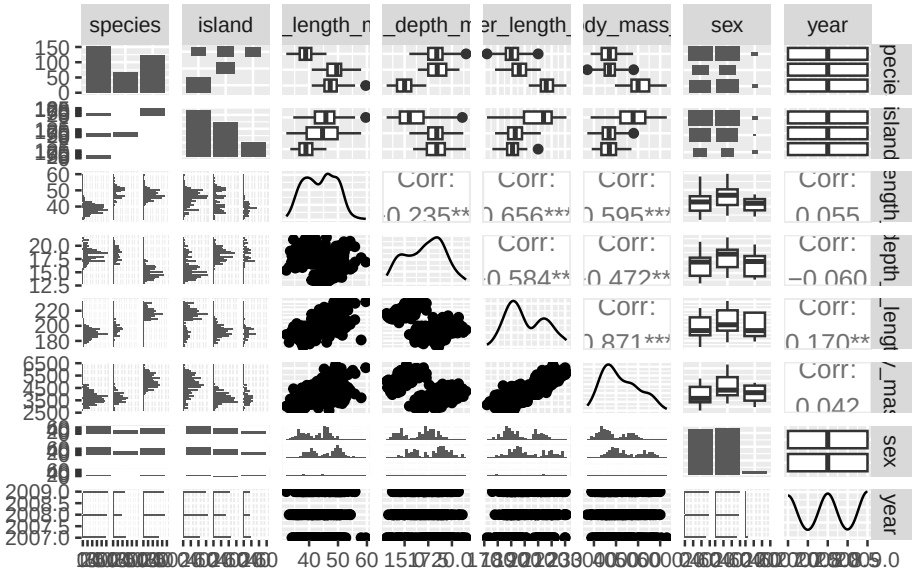
```
pour_regression_multiple <-
  penguins |>
  drop_na(body_mass_g, flipper_length_mm,
   ↪  bill_depth_mm, bill_length_mm)
```

On doit ensuite explorer visuellement nos données pour s'assurer que tout est OK :

```
ggpairs(pour_regression_multiple)
```

```
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```

```
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.

Warning: Removed 9 rows containing missing values or
values
outside the scale range (`stat_boxplot()`).

`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```



Donc, à première vue, nos relations sont relativement linéaires et nos variables relativement normales. Par contre, certaines de nos variables explicatives sont plus ou moins corrélées ensemble (entre autres, 0,65

entre le longueur des ailes et la longueur du bec). Il faudra vérifier que cela n'ajoute pas trop d'instabilité à notre modèle.

On peut maintenant ajuster notre modèle de régression multiple. Pour cela, on utilise la même fonction que pour la régression linéaire, soit `lm`:

```
modele <- lm(body_mass_g ~
               flipper_length_mm + bill_length_mm +
 ↪  bill_depth_mm,
             data = pour_regression_multiple)
```

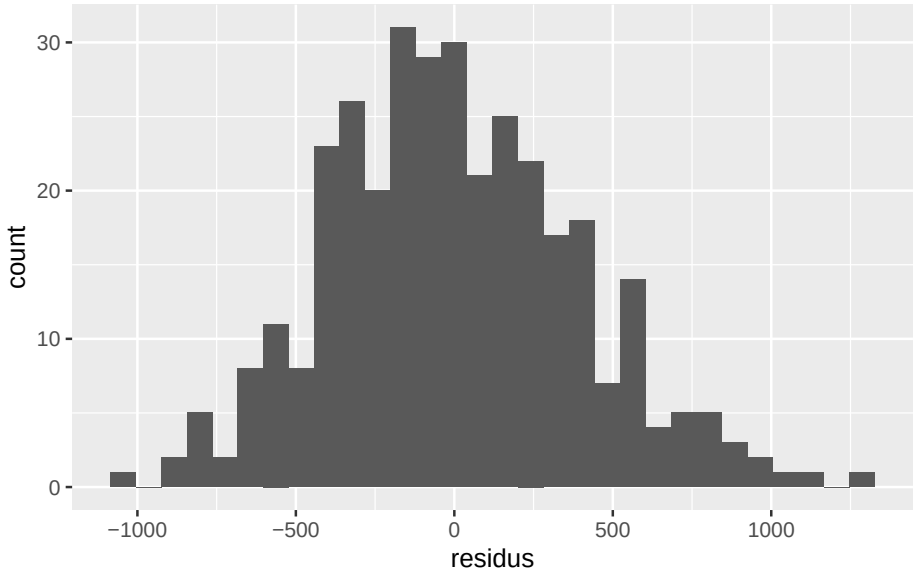
Avant d'aller voir quoi que ce soit dans nos résultats, il importe d'abord de bien valider notre modèle. Ajoutons donc d'abord pour se faire les chiffres qui nous seront nécessaires dans notre tableau de données, soit les résidus du modèle, les prédictions et les distances de Cook :

```
pour_regression_multiple <- pour_regression_multiple |>
  mutate(
   residus = resid(modele) ,
   predictions = predict(modele),
   D = cooks.distance(modele)
)
```

Donc, première chose à regarder, la normalité des résidus :

```
pour_regression_multiple |>
  ggplot(aes(x = residus)) +
  geom_histogram()
```

```
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```

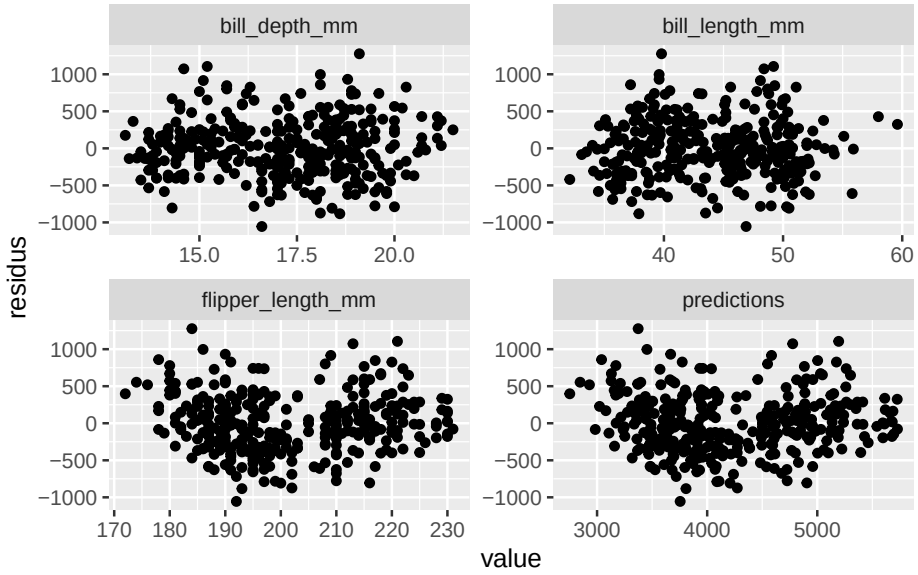


Les résidus sont relativement normaux. Rien d'inquiétant de ce côté.

Ensuite, pour vérifier l'homogénéité des résidus, il faut, je le rappelle, faire un graphique des résidus en fonction de (1) les valeurs prédites et (2) chacune des variables entrées dans le modèle.

On pourrait faire chacun de ces graphiques individuellement, mais avec un peu de magie noire, sachez qu'il est aussi possible de faire tous ces graphiques en une seule commande :

```
pour_regression_multiple |>
  pivot_longer(cols = c(flipper_length_mm,
   ↪  bill_length_mm, bill_depth_mm,predictions)) |>
  ggplot(aes(x = value, y = residus)) +
    geom_point() +
    facet_wrap(~name, scales = "free")
```



À première vue, les quatre nuages de points sont relativement homogènes. Le modèle ne semble pas être pire sur la droite ou sur la gauche de chacun de ces graphiques.

Comme pour la régression linéaire, il est important de vérifier que nous n'avons pas d'observations individuelles qui pourraient venir biaiser nos résultats. La distance de Cook s'utilise exactement de la même façon avec la régression multiple, soit qu'il faut s'assurer qu'aucune valeur n'est plus grande que 1 :

```
pour_regression_multiple |>
  filter(D > 1)
```

```
# A tibble: 0 x 11
# i 11 variables: species <fct>, island <fct>,
#   bill_length_mm <dbl>, bill_depth_mm <dbl>,
#   flipper_length_mm <int>, body_mass_g <int>,
```

```
#   sex <fct>, year <int>, residus <dbl>,
#   predictions <dbl>, D <dbl>
```

Ici, pas de problèmes, aucune valeur >1

Enfin, il faut aussi valider l'assomption supplémentaire de la régression multiple, soit la colinéarité avec le VIF. Pour se faire, nous utiliserons la fonction `vif` provenant de la librairie `car` :

```
vif(modele)
```

```
flipper_length_mm    bill_length_mm     bill_depth_mm
         2.673338          1.865090          1.611292
```

Toutes nos valeurs sont < 4, donc aucune inquiétude de ce côté.

Maintenant que tout cela est vérifié, on peut ENFIN regarder nos résultats :

```
summary(modele)
```

```
Call:
lm(formula = body_mass_g ~ flipper_length_mm +
bill_length_mm +
    bill_depth_mm, data = pour_regression_multiple)

Residuals:
     Min       1Q   Median       3Q      Max
-1054.94  -290.33   -21.91   239.04  1276.64

Coefficients:
                  Estimate Std. Error t value
(Intercept)       -6424.765    561.469 -11.443
flipper_length_mm    50.269      2.477  20.293
```

```
bill_length_mm        4.162      5.329   0.781
bill_depth_mm        20.050     13.694   1.464
                   Pr(>|t|)
(Intercept)         <2e-16 ***
flipper_length_mm   <2e-16 ***
bill_length_mm       0.435
bill_depth_mm        0.144
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 393.4 on 338 degrees of freedom
Multiple R-squared:  0.7615,    Adjusted R-squared:
0.7594
F-statistic: 359.7 on 3 and 338 DF,  p-value: < 2.2e-16
```

Remarquez que ces sorties sont très très semblables à celles que nous avons vues avec la régression simple. La seule différence est que le tableau Coefficients contient maintenant deux lignes supplémentaires. Notre modèle contient maintenant quatre paramètres, soit l'ordonnée à l'origine (Intercept) et les trois pentes pour les variables explicatives (longueur des ailes, longueur du bec et épaisseur du bec).

On peut donc constater qu'une seule pente est significativement différente de zéro (p<0,05). Le poids du corps augmente avec la longueur des ailes, mais les variables concernant le bec n'ont pas d'impact clair.

Comme une seule variable a un effet différente de zéro, nous n'avons pas vraiment besoin de standardiser les tailles d'effets pour voir laquelle a l'effet le plus fort. Pour les besoins de la cause, nous le ferons quand même pour que vous connaissiez la façon de le faire dans R.

Le plus simple est sans doute de refaire notre modèle en appliquant la fonction `scale` sur nos données. Comme notre tableau contient des variables qui ne sont pas numériques, on ne pas appliquer la fonction

`scale` sur toute les colonnes. On aurait pu évidemment simplifier notre tableau pour ne garder que ces dernières, mais on peut aussi utiliser un petit truc, pour appliquer la fonction `scale` uniquement sur les colonnes qui sont des chiffres (`is.numeric`).

```
modele_standardise <- lm(body_mass_g ~
          flipper_length_mm + bill_length_mm +
 ↪  bill_depth_mm,
        data = pour_regression_multiple |>
         ↪  mutate_if(is.numeric,scale)
)

summary(modele_standardise)
```

```
Call:
lm(formula = body_mass_g ~ flipper_length_mm +
bill_length_mm +
    bill_depth_mm, data =
    mutate_if(pour_regression_multiple,
    is.numeric, scale))

Residuals:
     Min       1Q   Median       3Q      Max
-1.31546 -0.36203 -0.02733  0.29807  1.59191

Coefficients:
                  Estimate Std. Error t value
(Intercept)       1.226e-15  2.653e-02   0.000
flipper_length_mm 8.814e-01  4.343e-02  20.293
bill_length_mm    2.833e-02  3.628e-02   0.781
bill_depth_mm     4.937e-02  3.372e-02   1.464
                  Pr(>|t|)
```

```
(Intercept)          1.000
flipper_length_mm   <2e-16 ***
bill_length_mm       0.435
bill_depth_mm        0.144
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4906 on 338 degrees of
freedom
Multiple R-squared:  0.7615,    Adjusted R-squared:
0.7594
F-statistic: 359.7 on 3 and 338 DF,  p-value: < 2.2e-16
```

Remarquez dans ces sorties que les valeurs de p et le r^2 sont absolument identiques à la sortie précédente. La seule chose importante qui a changé est l'échelle des estimés de paramètres (les colonnes Estimate et Std. Error).

On voit maintenant que pour chaque augmentation d'un écart-type de longueur d'aile, le poids du corps augment de 0.88 écart-type. Presque du 1:1. À titre de comparaison, augmenter la longueur du bec d'un écart-type augmente le poids du corps que de 0,028 écart-type.

27.8. Contenu optionnel : Visualiser les pentes partielles

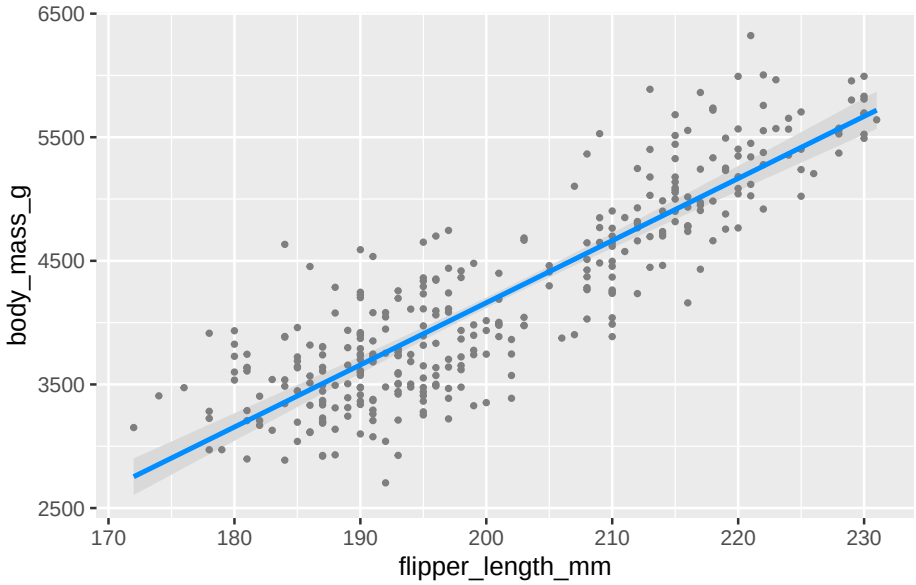
Nous avons discuté dans la section qui décrivait le modèle statistique que les pentes dans une régression multiple étaient en fait des pentes partielles. C'est-à-dire qu'elles décrivent un changement en Y pour un changement d'une de nos variables en X, en assumant que les autres variables ne bougent pas.

Il est possible de visualiser ces pentes partielles à l'aide de ce que l'on appelle un **graphique des résidus partiels** (*partial residual plot*). Dans ce dernier, on place comme à l'habitude notre variable explicative en X, mais en Y, plutôt que de mettre les valeurs de la variable expliquée, on place le résultat de l'addition de la pente de notre variable d'intérêt et des résidus du modèle. Cette passe-passe mathématique nous permet de recréer la pente partielle : ce que l'on met en Y sur le graphique est une recréation de nos données dans laquelle l'effet de toutes les autres variables a été éliminé.

Une des difficultés d'interprétation d'un graphique des résidus partiels vient du fait que l'échelle des Y est difficile à interpréter, puisque l'apport des autres variables est éliminé. C'est pourquoi, l'outil le plus commun pour visualiser les pentes partielles, soit la fonction `visreg` de la librairie du même nom, calcule plutôt un graphique conditionnel. Dans ce graphique, on a fait le calcul des résidus partiels, mais on intègre aussi au calcul l'apport des autres variables comme si elles étaient fixées à la médiane pour les quantitatives et au mode pour les qualitatives. Cette petite passe-passe permet de récupérer un axe des Y plus naturel à interpréter.

Voici par exemple le graphique des résidus partiels pour la longueur des ailes :

```
library(visreg)
visreg(modele, "flipper_length_mm", gg=TRUE)
```



Dans ce graphique, plutôt que d'avoir les vraies observations de poids en Y, on a une version simplifiée du poids, dans laquelle la variabilité attribuable aux autres variables a été enlevée. L'argument `gg=TRUE` est optionnel, mais informe la fonction que l'on veut obtenir un graphique compatible avec ggplot2, pour nous permettre d'ajouter des couches, de modifier les étiquettes, etc. comme on l'a fait ensemble depuis le début du cours.

27.9. Labo : Faire des prédictions

Pour faire des prédictions avec un modèle de régression multiple, on travaille exactement de la même façon qu'avec la régression linéaire, soit en appelant la fonction `predict` sur notre modèle et en lui fournissant un tableau avec les valeurs à prédire.

> ⚠ Avertissement
>
> Ce n'est pas le cas ici, mais remarquez que si vous aviez transformé vos variables (p. ex. log) pour ajuster le modèle, il faut aussi transformer les données avant de les utiliser pour faires des prédictions.

Si on voulait par exemple prédire le poids d'un manchot avec des ailes de 220 mm, un bec long de 39 mm et 18 mm d'épais, on ferait ceci :

```
predict(modele, data.frame(
  flipper_length_mm = 220,
  bill_length_mm = 39,
  bill_depth_mm = 18
))
```

```
       1
5157.667
```

Toutes les intervalles de confiance décrites au Chapitre 18 fonctionnent exactement de la même façon aussi.

27.10. Labo : La régression polynomiale

Vous avez peut-être remarqué que nous n'avons PAS parlé ici du fait que la linéarité est une assomption de la régression multiple. La raison est simple : la régression linéaire peut ajuster d'autres formes de relations que de simples droites!

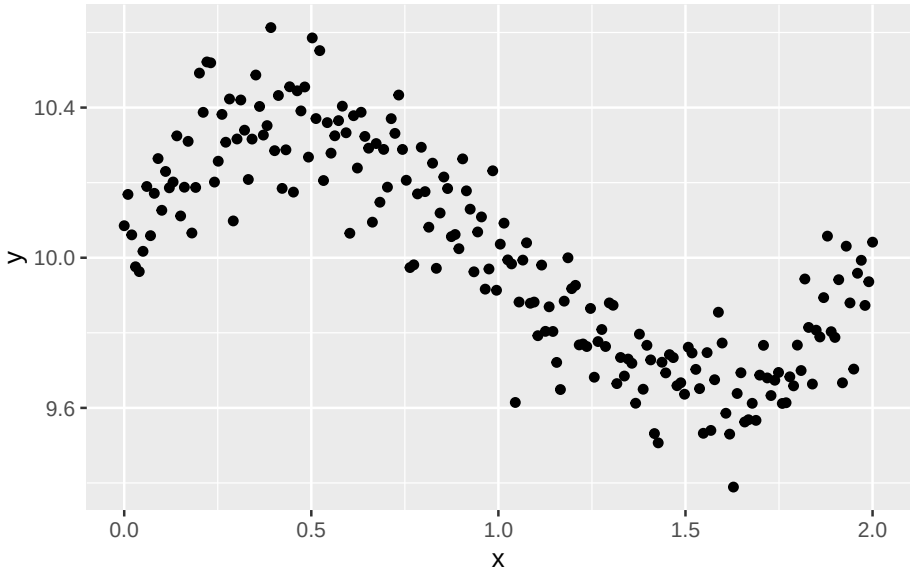
Nous avons tout d'abord vu dans le chapitre sur les transformations (Section 9.3) que l'ont peut, par exemple, linéariser une relation avec une transformation log-log. Dans les faits, la relation n'était pas linéaire, mais nous la transformons pour y arriver.

Une autre stratégie souvent utilisée est d'intégrer des termes polynomiaux à notre régression linéaire. Maintenant que nous savons que la régression peut contenir plusieurs termes, il est intéressant de savoir qu'elle peut aussi contenir le même terme, avec différents exposants. On pourrait par exemple tenter d'ajuster une polynomiale de degré trois comme ceci:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

Comme nous n'avons pas de données prêtes à tester ce genre de modèles, nous allons nous créer un tableau avec de fausses données pour essayer :

```
fausses <- tibble(
  x = seq(from=0,to=2,length.out = 200),
  y = rnorm(n=200,mean = 10 + 2*x + -3*x^2 + +x^3,sd =
   ↪  0.1)
)
fausses |>
ggplot(aes(x,y)) +
  geom_point()
```



Et maintenant, essayons de nous créer un modèle de régression multiple pour voir si nous réussissons à récupérer les valeurs de nos paramètres :

```
m <- lm(y~x+I(x^2)+I(x^3), data = fausses)
```

> **Avertissement**
>
> Remarquez que nos termes polynomiaux doivent être emballés dans la fonction `I()`. Si on ne le fait pas, R interprétera mal ce que l'on veut faire, mais ne fera PAS de message d'erreur pour nous avertir!

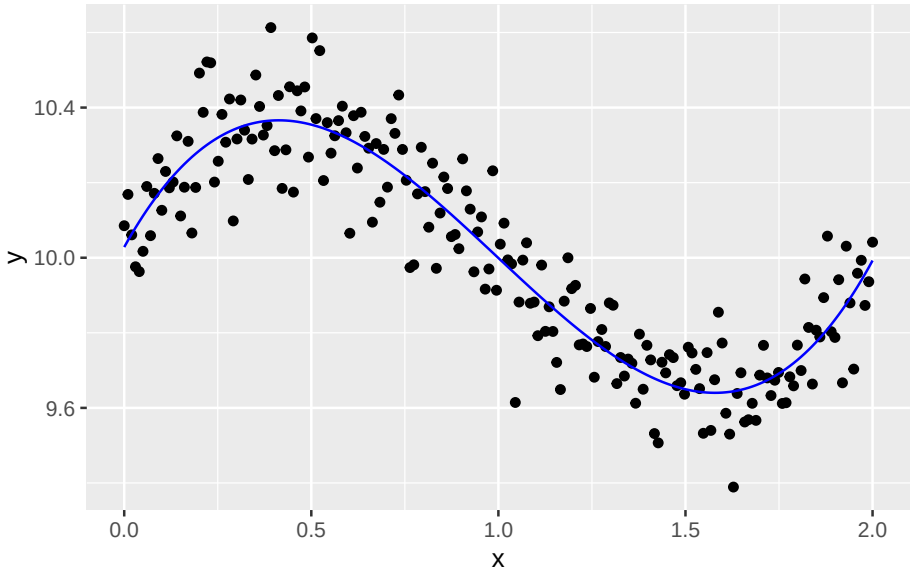
```
summary(m)
```

```
Call:
lm(formula = y ~ x + I(x^2) + I(x^3), data = fausses)

Residuals:
     Min       1Q   Median       3Q      Max
-0.34277 -0.06675  0.00574  0.07250  0.24798

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 10.02837    0.02969  337.81   <2e-16 ***
x            1.79183    0.12887   13.90   <2e-16 ***
I(x^2)      -2.73692    0.14994  -18.25   <2e-16 ***
I(x^3)       0.91603    0.04928   18.59   <2e-16 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1069 on 196 degrees of
freedom
Multiple R-squared:  0.8587,    Adjusted R-squared:
0.8565
F-statistic: 396.9 on 3 and 196 DF,  p-value: < 2.2e-16
```

Les coeffcients que nous avons récupéré sont très près des originaux, soit 9,97 vs 10, 2,18 vs 2, -3,22 vs -3 et 1,07 vs 1.

On peut constater le bel ajustement de ce modèle avec un graphique :

```
fausses |>
  mutate(predictions = predict(m)) |>
  ggplot(aes(x,y)) +
  geom_point() +
  geom_line(aes(y=predictions), color = "blue")
```



Remarquez qu'en général, tester un modèle avec de fausses données comme nous l'avons fait ci-haut est une excellente stratégie pour comprendre les limites de nos outils.

Avertissement

Attention, vos relations non-linéaires n'aurons pas nécessairement besoin d'une polynomiale de degré 3. Parfois, une degré 2 suffira (i.e. $y = \beta_0 + \beta_1 x + \beta_2 x^2$)

Note

Remarquez aussi que l'on peut combiner plusieurs variables dans un même modèle, avec ou sans des termes polynomiaux.

Par exemple, si on sait qu'un espèce a un optimal de température (et

donc, son abondance devrait suivre une courbe en cloche selon cette variable) et répond de façon linéaire à la quantité de proies (plus on a de proies, plus on a d'abonance), on pourrait ajuster le modèle suivant :

$$abondance = \beta_1 temprature + \beta_2 temprature^2 + \beta_3 temprature^3 + \beta_4 proies$$

27.11. Exercice : la régression multiple

Nous allons travailler pour cet exercice avec un jeu de données où l'auteur (Loyn, 1987) a tenté de modéliser l'abondance d'oiseaux dans des parcelles forestières, à partir de variables mesurées à l'échelle du paysage entourant la parcelle. Le jeu de données peut être récupéré ici en format CSV[1].

Pour cet exercice, nous tenterons de déterminer quels sont les facteurs affectant l'abondance d'oiseaux (ABUND), parmis les suivants :

- AREA : la surface de la parcelle en hectares
- YR.ISOL : l'année depuis laquelle la parcelle est isolée
- DIST : distance (km) à la parcelle la plus proche
- ALT : l'altitude de la parcelle (m au-dessus du niveau de la mer)

Vous verrez qu'à partir de maintenant, vous passerez beaucoup plus de temps à préparer et vérifier vos données et les assomptions des modèles plutôt qu'à faire des statistiques comme tel. Pour vous guider dans votre exercice, assurez-vous d'effectuer toutes les étapes suivantes dans votre modélisation :

- Chargement et vérification des données

[1] https://drive.google.com/file/d/19-dlzMx5xpXd7Bp_b0gMJrkgkU6ae43W/view?usp=sharing

- Préparation des variables (enlever les colonnes inutiles, les données manquantes, transformer les données)
- Ajustement du modèle
- Vérification du modèle
- Interprétation des résultats

Une fois votre modèle prêt, essayez de l'utiliser pour prédire l'abondance d'oiseaux que l'on pourrait trouver dans une parcelle de 100 ha, isolée depuis 1985, à 5 km de la parcelle la plus proche et à une altitude de 150 m au-dessus du niveau de la mer. Produisez aussi l'intervalle à 95 % de cette prédiction. Autrement dit, 95 % du temps, entre quelles valeurs devrait se situer l'abondance d'oiseaux dans une telle parcelle.

28. La sélection du meilleur modèle

28.1. Introduction

Au Chapitre 27, nous avons vu comment ajuster un modèle de régression multiple. Nous avons aussi vu qu'au terme de l'analyse, nous obtenions un tableau des estimés de paramètres, qui nous indiquait la pente partielle pour chaque variable, ainsi qu'une valeur de p et une erreur-type associée, nous indiquant notre certitude quant à l'estimation de chacune de ces pentes.

Par exemple :

```
                   Estimate Std. Error t value Pr(>|t|)
↪
(Intercept)       -6424.765    561.469 -11.443   <2e-16
↪   ***
flipper_length_mm    50.269      2.477  20.293   <2e-16
↪   ***
bill_length_mm        4.162      5.329   0.781    0.435
↪
bill_depth_mm        20.050     13.694   1.464    0.144
```

Basé sur la valeur de p,certains paramètres ont un effet clair (longueur des ailes), d'autres un effet plus mitigé (longueur et épaisseur du bec).

Ce qu'il est par contre important de savoir est que la valeur de p est une technique parmi d'autres permettant permettant d'évaluer quelles variables garder et quelles éliminer d'un modèle.

Ce processus, de passer d'un modèle complet contenant toutes les variables pour lesquelles nous avions des hypothèses à un modèle simplifié ne retenant que celles que nous jugeons importantes après le processus de modélisation se nomme la **sélection de modèle**. Ce sujet fera l'objet d'un chapitre entier, puisqu'il existe de multiples philosophies et techniques différentes, avec chacunes de bons et des mauvais côtés.

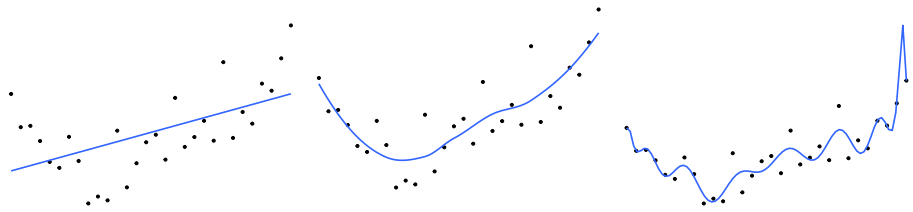
28.2. Comment juger de l'explication?

Comment savoir si un modèle statistique explique bien un phénomène naturel? Une façon classique de le déterminer est de se demander si toutes les variables importantes sont effectivement contenues dans le modèle. Et à l'inverse, si une variable n'est pas importante, est-elle effectivement laissée de côté?

28.3. Comment juger des prédictions?

Naturellement, on aurait tendance à penser que plus un modèle possède de petits résidus, plus il offre de bonnes prédictions, donc plus il est bon, non?

Dans la figure ci-dessous, on voit clairement par exemple que le modèle B est meilleur que le A, ses résidus seront plus petits, et il est mieux ajusté aux données. Mais qu'en est-il du modèle C? Sur les données affichées sur la figure, le modèle C a clairement les plus petits résidus, mais il a aussi un important défaut : il ne généralise pas bien.



(a) A. Modèle sous-ajusté (a) B. Modèle correctement ajusté (a) C. Modèle sur-ajusté

Si on va cueillir un nouvel échantillon sur le terrain et qu'on passe les données dans ces modèles, les résidus du modèle C seront inévitablement plus grands que ceux du B, parce que le modèle est non seulement ajusté au patron dans nos données, mais il s'est aussi ajusté au bruit.

On qualifie en général les modèles comme le C de sur-ajustés. Ils ont un pouvoir prédictif plus faible si on leur présente de nouvelles données.

Dans notre quête du meilleur modèle statistique pour décrire un phénomène, notre but sera donc de trouver un modèle qui à la fois (1) explique bien (2) prédit bien mais (3), qui n'est pas sur-ajusté.

28.4. Équivalence

En écologie, en plus des questionnements mentionnés ci-dessus, nous devons souvent gérer un problème supplémentaire. Puisque les variables prédictives sont souvent corrélées entre-elles, il peut être difficile d'en distinguer les effets, au point où il peut arriver que plusieurs modèles puissent faire un travail équivalent.

C'est quelque chose qu'il est important de garder en tête. Il ne sera pas toujours possible de trancher.

28.5. Alors, comment choisir le meilleur modèle?

La façon de choisir le meilleur modèle au terme d'une modélisation statistique est un sujet extrêmement complexe, où il y a des divergences d'opinions extrêmes. Dans certains cas, nous pouvons presque parler de guerres saintes tellement les arguments sont campés et basés sur de fortes croyances.

Puisque nous ne pouvons pas savoir à l'avance ce que vos supérieurs (directeurs de thèses, patrons etc.) exigerons de vous, nous n'avons d'autre choix que de voir chacune des façons de faire, avec leurs points forts et leurs points faibles.

La sélection du meilleur modèle peut être divisée en deux grands thèmes, soit la stratégie de sélection (quelles sont les étapes, comment on ajoute ou on enlève des variables) et l'évaluation du modèle (sur quoi se base-t-on pour dire si un modèle est meilleur qu'un autre).

28.6. Les 5 stratégies de sélection

Donc, voyons d'abord les différentes stratégies par lesquelles on peut partir d'un modèle statistique (tel que nous l'avions imaginé au début de l'expérience) au meilleur modèle, tel qu'il devra être communiqué à nos lecteurs et aux décideurs.

28.6.1. Tout garder

La première stratégie de sélection du meilleur modèle est aussi la plus simple. Elle consiste à simplement tout garder. C'est-à-dire, ne pas tenter de simplifier notre modèle en enlevant des variables.

Cette stratégie est peu employée. On la voit surtout dans les approches bayésiennes. Elle est particulièrement appropriée si votre modèle est simple (i.e. peu de variables) et que, surtout, vous avez choisi vos variables avec soin. Si vos variables ont toutes d'excellentes raisons biologiques de se retrouver dans le modèle et qu'il n'y a pas de long shot, de variables insérées "juste pour voir", cette stratégie peut alors être appropriée.

Les partisans (en général) de cette approche voient la nature comme un système où tout est relié. Leur but est alors non pas de tester des hypothèses, mais d'estimer le mieux possible chacun des paramètres. De quantifier chacune des forces, aussi faible soit-elle.

28.6.2. Exhaustive

Cette deuxième stratégie consiste à tester toutes les combinaisons de variables pour trouver la meilleure. Si notre jeu de données contient par exemple les variables explicatives A, B et C, nous devrons ajuster et comparer les modèles :

- Y ~ A + B + C
- Y ~ A + B
- Y ~ A + C
- Y ~ B + C
- Y ~ A
- Y ~ B
- Y ~ C
- Y ~ 1 (i.e. aucune variable).

En principe, il s'agit de la meilleure stratégie, parce qu'on est certain que la meilleure combinaison de variables sera trouvée.

Par contre, l'ajustement des modèles peut être particulièrement long. Si votre jeu de données contenait six variables, vous devrez tester 64 (2^6)

modèles différents. Cette approche est d'ailleurs parfois critiquée parce qu'elle va "à la pêche au modèle" en essayant beaucoup de combinaisons.

28.6.3. Par ajout (*forward selection*)

Le but de la méthode par ajout est de s'éviter de tester toutes les combinaisons possibles de variables, en utilisant une méthode itérative de construction qui nous permet de rapidement faire le tri.

Dans cette stratégie, on commence par ajuster un modèle vide, n'incluant aucune variable explicative. Ensuite, on ajoute progressivement des variables, en commençant par la meilleure (nous reviendrons sur comment le savoir), jusqu'à une valeur seuil où les variables non-ajoutées n'améliorent plus le modèle.

Ce seuil peut être un test d'hypothèse ou une statistique d'ajustement.

28.6.4. Par élimination (*backward selection*)

Cette stratégie est l'inverse de la précédente. On commence par un modèle complet contenant toutes les variables. On supprime ensuite progressivement des variables du modèle, jusqu'au seuil ou ne peut plus supprimer de variables sans dégrader le modèle de façon importante.

Encore une fois, ce seuil peut être un test d'hypothèse ou une statistique d'ajustement.

28.6.5. Comparaison

Contrairement à la méthode exhaustive, les méthodes par ajout et par élimination ne garantissent pas que le meilleur modèle sera trouvé.

La méthode par ajout aura tendance à bien fonctionner lorsque les variables sont indépendantes (non-corrélées) alors que la méthode par élimination aura tendance à donner un meilleur portrait de la situation lorsque les variables sont corrélées entre-elles.

28.6.6. Hybride (stepwise)

La méthode hybride combine les meilleurs côtés des approches par ajout et par élimination.

Elle démarre avec un modèle vide. Puis, à chaque étape, elle teste pour l'ajout de chaque variable présentement absente du modèle ET la suppression de chaque variable déjà incluse dans le modèle.

Par contre, cela ne garantit pas non plus que le meilleur modèle sera nécessairement trouvé... mais ça augmente nos chances.

28.6.7. Controverses

Les stratégies par ajout, par élimination et hybride sont particulièrement controversées.

D'abord, parce que changer la valeur seuil du test d'hypothèse changera souvent le modèle sélectionné. On voit souvent p. ex. un seuil d'ajout à $p < 0{,}1$ plutôt que $p < 0{,}05$, ce qui inévitablement inclura plus de variables dans le modèle final.

Ces méthodes augmentent aussi l'erreur de type I, puisque l'on applique souvent beaucoup de tests. Même si notre seuil est à p < 0,05, si on effectue 10 tests, notre taux d'erreur de type I effectif sera plutôt de 0,5 (10 x 0,05; i.e. une chance sur deux de trouver quelque chose quand il n'y avait rien).

28.7. 3 méthodes d'évaluation des modèles

Voyons maintenant trois façons différentes de comparer des modèles dans les stratégies de sélection vues ci-haut.

28.7.1. Tests d'hypothèses

Comme nous avons vu lors des cours précédents, on peut déterminer si une variable est statistiquement significative dans un modèle en comparant l'augmentation de variance expliquée fournie par une variable à la variance des résidus du modèle complet par un test de F partiel.

Cette méthode ne s'applique qu'aux 3 stratégies controversées (par ajout, par élimination et hybride). Aussi, elle ne s'applique qu'à des modèles imbriqués. C'est-à-dire qu'elle peut par exemple comparer

Y ~ A + B vs. Y ~ A

mais elle ne peut pas comparer

Y ~ A + B vs. Y ~ C + D

28.7.2. Labo : Les approches par tests d'hypothèses

Les tests d'hypothèses sont en général combinés soit à l'approche par ajout ou par élimination. Nous verrons donc ces deux façons de faire ici.

Pour toutes les approches, nous réutiliserons le modèle de régression multiple développé au Chapitre 27, sur lequel nous essaierons chacune des techniques.

Donc, commençons par charger nos librairies, et à préparer notre tableau de données.

```
library(tidyverse)
library(palmerpenguins)

pour_regression_multiple <-
  penguins |>
  drop_na(body_mass_g, flipper_length_mm,
   ↪  bill_depth_mm, bill_length_mm)
```

Pour débuter l'**approche par ajout**, il faut d'abord se créer un modèle de régression vide, dans lequel il n'y a aucune variable explicative. On met uniquement l'ordonnée à l'origine dans le modèle (oui, c'est bizarre la première fois qu'on entend ça!).

Habituellement, quand on spécifie la formule d'un modèle de régression dans R, on a pas besoin de parler de l'ordonnée à l'origine, R en met une pour nous automatiquement. Par contre, quand il n'y a aucune variable, il faut la spécifier nous-même, en mettant ~ 1 à droite de la formule :

```
modele1 <- lm(body_mass_g~1, data =
 ↪  pour_regression_multiple)
```

Ensuite, la façon la plus simple d'organiser notre méthode par ajouts est de se préparer une formule, contenant toutes les variables que l'on voudrait essayer de mettre dans notre modèle :

```
termes <- ~ flipper_length_mm + bill_length_mm +
 ↪  bill_depth_mm
```

Remarquez que l'on ne met rien à gauche de cette formule. C'est comme une formule partielle.

Ensuite, il existe dans R une fonction nommée `add1`, dans laquelle R essaie pour nous, chacun des termes d'une formule qui ne sont pas déjà dans le modèle :

```
add1(modele1, scope = termes, test = "F")
```

```
Single term additions

Model:
body_mass_g ~ 1
                  Df Sum of Sq       RSS    AIC
<none>                          219307697 4574.9
flipper_length_mm  1 166452902  52854796 4090.3
bill_length_mm     1  77669072 141638626 4427.4
bill_depth_mm      1  48840779 170466918 4490.8
                   F value    Pr(>F)
<none>
flipper_length_mm 1070.745 < 2.2e-16 ***
bill_length_mm     186.443 < 2.2e-16 ***
bill_depth_mm       97.414 < 2.2e-16 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La sortie de `add1` contient plusieurs lignes, une pour chaque variable que R a essayé d'ajouter. On a ensuite, pour chacun des modèles testés, les degrés de liberté (Df), la somme des carrés (Sum of Sq), la somme des carrés des résidus (RSS), l'AIC (nous en discuterons plus bas), la valeur de F et la valeur de p.

Quand on voit ce tableau, puisque l'on est en mode ajout de variable, ce qui nous intéresse est de savoir quelle variable était la plus significative. Pour se faire, on cherche la valeur de F la plus grande.

Dans notre cas, la valeur de F la plus élevée est celle de la longueur des ailes, à 1070,745. Il faut alors vérifier si cette variable avec la valeur de F la plus élevée est significative ou non. Ici, sa valeur de p est clairement < 0,05, et donc on décide d'ajouter cette variable officiellement dans notre meilleur modèle.

> **Note**
>
> Remarquez qu'en ajoutant une seule variable à la fois, chacune des variables aurait pu être significative dans le modèle, même l'épaisseur bec. Le but de la technique par ajout est justement de savoir si, une fois la meilleur variable dans le modèle, les autres ont encore un effet ou elles deviennent redondantes.

On se crée donc un deuxième modèle, qui contient cette nouvelle variable en plus de celles qui y étaient déjà (ici, aucune).

```
modele2 <- lm(body_mass_g ~ flipper_length_mm, data =
 ↪  pour_regression_multiple)
```

Une fois que c'est fait, on peut refaire la commande `add1`, pour que R teste tous les termes restants :

```r
add1(modele2, scope = termes, test = "F")
```

```
Single term additions

Model:
body_mass_g ~ flipper_length_mm
               Df Sum of Sq      RSS    AIC F value
<none>                      52854796 4090.3
bill_length_mm  1    211671 52643125 4090.9  1.3631
bill_depth_mm   1    449044 52405752 4089.4  2.9048
                Pr(>F)
<none>
bill_length_mm 0.24383
bill_depth_mm  0.08924 .
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La fonction `add1` a donc tenté d'ajouter, séparémment, les variables de longueur et d'épaisseur du bec au modèle contenant déjà la longueur des ailes. On répète donc encore la même procédure : quelle est la variable avec la valeur de F la plus élevée? Est-elle significative?

Dans ce cas-ci, la valeur de F la plus élevée est celle de l'épaisseur du bec, mais sa valeur de p (0,089) est > 0,05. On ne l'ajoute donc pas au meilleur modèle et la procédure s'arrête ici.

La sélection du meilleur modèle par ajout de variables s'arrête toujours ainsi, au moment où aucun des termes testés n'apporte un ajout significatif au modèle. Le meilleur modèle, tel que sélectionné par cette approche, est donc celui qui ne contenait que la variable de longueur des ailes.

```
modele_final <- modele2
```

Avec les tests d'hypothèses, on aurait aussi pu appliquer la méthode par élimination. Pour se faire, on doit plutôt commencer par un modèle complet :

```
modele1 <- lm(
  body_mass_g ~
    flipper_length_mm + bill_length_mm + bill_depth_mm,
  data = pour_regression_multiple)
```

Ensuite, on demande à R avec la fonction `drop1`, d'essayer d'enlever chacun des termes pour voir si ça fait une différence significative :

```
drop1(modele1, test = "F")
```

```
Single term deletions

Model:
body_mass_g ~ flipper_length_mm + bill_length_mm +
bill_depth_mm
                  Df Sum of Sq       RSS    AIC
<none>                          52311359 4090.8
flipper_length_mm  1  63735497 116046856 4361.3
bill_length_mm     1     94393  52405752 4089.4
bill_depth_mm      1    331766  52643125 4090.9
                   F value Pr(>F)
<none>
flipper_length_mm 411.8149 <2e-16 ***
bill_length_mm      0.6099 0.4354
bill_depth_mm       2.1436 0.1441
---
```

```
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Par contre, ici, il ne faut pas oublier que l'on est en mode élimination. On veut enlever les termes inutiles, ceux qui ne font PAS de différence significative. On cherche donc la valeur de F la plus faible.

Ici, c'est celle associée à la longueur du bec (F=0,61). Comme enlever cette variable ne fait pas de différence significative sur le modèle (p > 0,05), on peut l'éliminer :

```
modele2 <- lm(
  body_mass_g ~
    flipper_length_mm + bill_depth_mm,
  data = pour_regression_multiple)
```

Puis, on rappelle `drop1` sur ce nouveau modèle pour que R essaie d'enlever chacun des termes restants :

```
drop1(modele2, test = "F")
```

```
Single term deletions

Model:
body_mass_g ~ flipper_length_mm + bill_depth_mm
                  Df Sum of Sq       RSS    AIC
<none>                          52405752 4089.4
flipper_length_mm  1 118061166 170466918 4490.8
bill_depth_mm      1    449044  52854796 4090.3
                   F value  Pr(>F)
<none>
flipper_length_mm 763.7088 < 2e-16 ***
bill_depth_mm       2.9048 0.08924 .
```

```
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

On cherche à nouveau la valeur de F la plus faible. Ici, c'est l'épaisseur du bec (F = 2,90). Comme enlever cette variable n'aurait pas d'impact significatif (p > 0,05), on peut aussi l'enlever du modèle :

```
modele3 <- lm(
  body_mass_g ~
    flipper_length_mm,
  data = pour_regression_multiple)
```

Puis on essaie d'enlever les variables restantes. Ici il ne nous reste que la longueur des ailes, mais il important de tout de même tester.

```
drop1(modele3, test = "F")
```

```
Single term deletions

Model:
body_mass_g ~ flipper_length_mm
                  Df Sum of Sq       RSS    AIC
<none>                          52854796 4090.3
flipper_length_mm  1 166452902 219307697 4574.9
                  F value    Pr(>F)
<none>
flipper_length_mm  1070.7 < 2.2e-16 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

On répète à nouveau la procédure : chercher la valeur de F la plus faible, et vérifier sa valeur de p.

Ici, la valeur de F la plus faible (la seule) est celle de la longueur des ailes (F=1070,7). Comme sa valeur de p est < 0,05, on ne peut PAS l'enlever du modèle. L'éliminer *aurait* un impact significatif.

Notre meilleur modèle est donc le modèle 3, qui ne contenait que la longueur des ailes.

```
modele_final <- modele3
```

> **Note**
>
> Autant dans la méthode par ajout que dans la méthode par élimination, il pourrait arriver que votre meilleur modèle soit le modèle vide, avec uniquement l'ordonnée à l'origine. Si ça vous arrive, vos hypothèses étaient probablement erronées ou votre puissance statistique trop faible. Et ça arrive plus souvent qu'on voudrait se l'avouer...

28.7.3. Explication de la variance

Intuitivement, le R^2 peut sembler une bonne mesure de comparaison de modèles, puisqu'il est, entre autres, utilisable dans tous les types de stratégies. Mais comme discuté précédemment, le R^2 ne permet pas de comparer des modèles incluant un nombre différent de variables, puisque le R^2 ne peut qu'augmenter lorsque l'on ajoute des variables.

Il existe une version modifiée du R^2, le R^2-ajusté qui elle est adaptée à ce genre de situation. Ce dernier est pénalisé pour le nombre de paramètres du modèle. Il augmente donc en fonction de l'ajustement, mais il diminue en fonction du nombre de paramètres (nombre de variables).

Même dans sa version ajustée, plusieurs auteurs ont par contre montré que, bien que ce chiffre soit une excellente statistique descriptive, il favorise dans certaines situations des modèles sur-ajustés.

De plus, puisque les transformations affectent la variance, le R^2, même ajusté, ne peut pas être utilisé pour comparer des modèles avec différentes transformations de la variable en Y.

L'autre problème posé par le R^2-ajusté est que, lorsque nous sommes confrontés à plusieurs modèles équivalents (c'est souvent le cas en écologie), il ne permet pas de quantifier la probabilité que chacun de ces modèles soit individuellement le meilleur. Remarquez que la méthode basée sur les tests d'hypothèses ne permettait pas non plus de faire de telles affirmations probabilistes.

Néanmoins, comme pour les tests d'hypothèses, le R^2-ajusté demeure grandement utilisé dans la littérature.

28.7.4. Labo : L'approche basée sur le R^2-ajusté

La façon la plus commune d'utiliser le R^2-ajusté dans la sélection de modèle est en le combinant avec une approche exhaustive, où toutes les combinaisons de variables sont testées. On pourrait évidemment créer chacun de ces modèles nous-mêmes, mais il existe une fonction dans la librairie leaps nous permettant de le faire de façon automatisée :

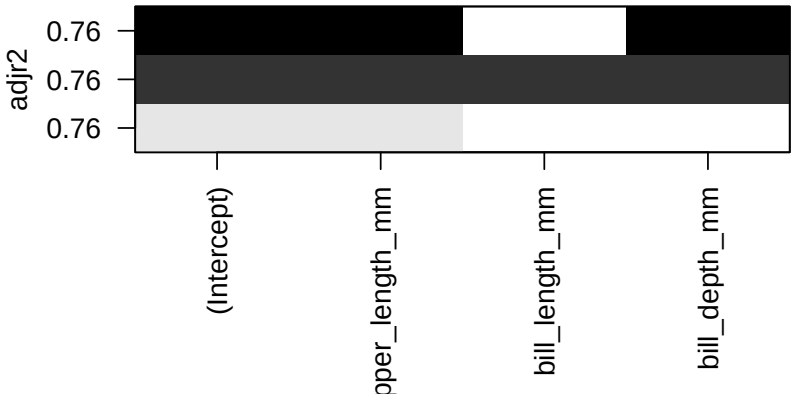
```
library(leaps)
```

La fonction pour tester toutes les combinaisons se nomme `regsubsets`. Elle s'utilise exactement comme la fonction `lm`, à laquelle on enverrait notre modèle complet :

```
resultat <- regsubsets(
  body_mass_g ~ flipper_length_mm + bill_length_mm +
↪   bill_depth_mm,
  data = pour_regression_multiple
)
```

Ensuite, on peut visualiser ces résultats, comme ceci :

```
plot(resultat, scale = "adjr2")
```



Il s'agit là, malheureusement, du graphique le plus mal conçu que vous aurez à interpréter dans ce livre. Sur l'axe des Y, vous avez des valeurs de R^2-ajusté et sur l'axe des X, vous avez les différentes variables de notre modèle complet. Ce qu'il faut comprendre est que chaque rangée dans le graphique représente un modèle (i.e. une combinaison de variables). Si la case est blanche vis-à-vis la variable, cette dernière était absente de ce modèle en particulier. Plus les cases sont sombres pour un modèle en particulier, meilleur était ce modèle. La couleur est donc redondante par rapport à l'axe des Y.

Ici, on voit donc que le meilleur modèle était celui contenant l'ordonnée à l'origine (intercept), la longueur des ailes et l'épaisseur du bec. Le modèle

modèle contenant les 3 variables était cependant très près derrière, au point où l'axe des Y affiche la même valeur pour tous les modèles (r^2-ajusté=0,76).

Remarquez que ce résultat est différent des deux précédents. Avec le R^2-ajusté, on aurait classé l'épaisseur du bec comme variable importante pour comprendre le poids des manchots.

À ce moment-ci, vous vous demandez peut-être pourquoi j'ai classé cette méthode comme exhaustive, alors que, entre autres, le modèle contenant uniquement la longuer du bec n'apparaît pas dans les résultats? En fait, `regsubsets` a effectivement testé toutes les combinaisons de variables, mais dans les résultats, il nous montre uniquement le meilleur modèle avec 1 variable, le meilleur avec 2 variables, le meilleur avec 3, etc. Comme le meilleur avec une seule variable était celui avec la longueur des ailes, celui avec la longueur du bec n'apparaît simplement pas dans le graphique.

28.7.5. Le Critère d'information d'Akaike (AIC)

Avant de présenter la troisième méthode d'évaluation, il faut d'abord mettre en place un nouveau concept. Le critère d'information d'Akaike (AIC) est basé sur un tout autre paradigme que ce que l'on a vu jusqu'à maintenant dans le cours. Plutôt que de se baser sur les principes des moindres carrés, il provient de la théorie de l'information[1].

L'AIC mesure l'information perdue, par rapport à la réalité, lorsque l'on utilise un modèle statistique pour la représenter. Tout comme le R^2-ajusté, il pénalise aussi pour la complexité (le nombre de paramètres) du modèle.

[1] https://fr.wikipedia.org/wiki/Th%C3%A9orie_de_l%27information

> **Mise en garde**
>
> Attention, puisque l'AIC mesure l'information perdue, les meilleurs modèles auront un AIC plus faible.

L'utilisation de l'AIC est en forte augmentation depuis 15 ans environ. Il permet de comparer des modèles non-imbriqués (contrairement aux tests d'hypothèses). Il permet aussi de comparer la probabilité que différents modèles soient les meilleurs. Contrairement au R^2, il est aussi défini pour des modèles plus complexes comme les GLM (Generalized Linear Models) et les modèles mixtes. Enfin, il a moins tendance à sélectionner des modèles sur-ajustés comparé aux approches précédentes.

Par contre, l'AIC ne renseigne pas directement sur l'ajustement du modèle. Il exprime uniquement la performance relative d'un modèle par rapport à un autre. La valeur d'AIC ne s'interprète que de façon relative, entre plusieurs modèles expliquant la même variable expliquée. Sa valeur n'est pas interprétable dans l'absolu. Il faut donc souvent combiner l'AIC avec une valeur de R^2 pour discuter de la qualité générale de notre meilleur modèle. L'approche par l'AIC ne fournit pas non plus de valeur de p, ce qui peut être embêtant si votre directeur ou votre éditeur vous en exige.

De plus, il est important de garder en tête que la pénalité pour le nombre de paramètres (k) est arbitraire dans le calcul de l'AIC. Plusieurs alternatives existent, pénalisant de différentes façons. Entre autres le BIC remplace la pénalité de $2 \times k$ de l'AIC par $k \times ln(N)$ où N est le nombre d'observations. Le BIC est donc une version plus conservatrice.

Comme l'AIC en absolu ne s'interprète pas, on interprète plutôt les différences d'AIC entre deux modèles, que l'on nomme delta AIC en anglais. En général, on interprète le delta AIC comme ceci :

- 0-2 Niveau de support substantiel pour ce modèle
- 4-7 Considérablement moins de support

- >10 Essentiellement aucun support pour ce modèle

28.7.6. L'inférence multi-modèle

En se basant sur les delta AIC, il est possible de calculer la probabilité que tel ou tel modèle soit le meilleur (parmi ceux testés). Cette statistique s'appelle le poids d'Akaike (*Akaike weight*). Voici un exemple de tableau contenant tous ces chiffres :

Modèle	AIC	Delta AIC	Akaike weight
A	21	0	0.989
B	30	9	0.011
C	80	59	0.000
D	118	97	0.000

Dans ce cas, le modèle A est clairement meilleur que les autres, puisque son poids d'Akaike est de 0,989. Autrement dit, parmis les modèles testés, il y a 98,9 % des chances que le A soit le meilleur.

Par contre, il peut arriver que le poids d'Akaike ne permette pas de trancher clairement entre deux modèles, par exemple dans ce jeu de données :

Modèle	AIC	Delta AIC	Akaike weight
X	1001	1	0.37
Y	1000	0	0.62
Z	1008	8	0.01

Le modèle X a plus du tier des chances d'être le meilleur modèle pour expliquer nos données. Il ne peut pas être mis facilement de côté. Dans un cas comme celui-là, il faut être clair lorsque l'on décrit nos résultats:

le meilleur modèle est Y, mais le X ne peut pas être exclu non plus. Il peut être utile de faire le ratio des deux poids d'Akaike, pour affirmer par exemple que le modèle Y est 1,68 fois plus probable que le X (0,62/0,37).

Puisque dans chacun de ces modèles raisonnables nous avions calculé un estimé de pente, lequel des estimés de pente est le plus représentatif de la réalité? En fait, aucun. L'approche par l'AIC nous permet de calculer une pente (et un intervalle de confiance) qui tient compte de cette incertitude entre les modèles.

Le calcul se fait à l'aide d'une moyenne pondérée, basée sur le support (poids d'Akaike) associé à chacun des estimés.

Si par exemple nous avions deux modèles, A et B, ayant un poids d'Akaike respectif de 0,6 et 0,4. Dans chaque modèle, nous avons estimé une pente pour la variable X, qui était de 2 dans un cas et 2., dans l'autre. La valeur pondérée, considérant notre incertitude quant au meilleur modèle serait donc de 0,6 * 2 + 0,4 * 2,5 = 2,2

De la même façon, puisque certaines variables seront absentes ou présentes dans certains modèles, on peut se demander quelle est notre certitude qu'une variable fasse partie du meilleur modèle. Prenons par exemple les données suivantes :

Modèle	AIC	delta AIC	Akaike weight
a	21	0	0.86
b	30	9	0.01
c	80	59	0.00
a + b	118	97	0.00
b + c	25	4	0.12
a + c	31	10	0.01
a + b + c	50	29	0.00

L'importance relative d'une variable (sa probabilité de faire partie du

meilleur modèle) se calcule comme la somme des poids d'Akaike de tous les modèles où elle est présente :

- Importance relative de a : 0,86 + 0,00 + 0,01 + 0,00 = 0,87
- Importance relative de b : 0,01 + 0,00 + 0,12 + 0,00 = 0,13
- Importance relative de c : 0,00 + 0,12 + 0,01 + 0,00 = 0,13

Nous sommes donc vraiment sûr que a fait partie du meilleur modèle, mais très peu d'évidences que b ou c en font partie.

28.7.7. L'AICc

L'AIC, dans sa définition originale, était défini pour de grands jeux de données. Lorsque notre échantillon est petit, il faut plutôt utiliser l'AICc, qui contient une correction pour les petits échantillons. L'AICc nous évite de sélectionner accidentellement des modèles sur-ajustés (trop complexes).

On recommande en général d'utiliser l'AICc lorsque n/k est < 40, où n est notre nombre d'observations et k est le nombre de paramètres dans le modèle (dans ce cas-ci, le nombre de variables + 1 paramètre pour l'ordonnée à l'origine).

Par contre, vous ne vous tromperez jamais en utilisant systématiquement l'AICc, puisque sa valeur converge vers l'AIC quand n est grand ou k est petit.

28.7.8. Labo : L'inférence multimodèle basée sur l'AICc.

Plusieurs librairies de R nous permettent de calculer des valeurs d'AIC, entre autres le libraire car (comme voiture en anglais) que nous utilisons pour calculer le VIF. Cependant, certaines fournissent aussi une série de fonctions nous permettant de mettre en application toute l'approche

d'inférence multi-modèle. Nous en verrons une ici, soit la librairie **MuMIn** (*MUlti Model INference*). Si vous n'avez pas déjà cette librairie, vous devrez l'installer avant de procéder pour la suite.

```
library(MuMIn)
```

Ensuite, il faut, comme pour l'approche par élimination, ajuster un modèle complet contenant toutes les variables qui pourraient nous intéresser. Il faut cependant ajouter une petite option supplémentaire à notre modèle, pour lui demander de *planter* si il trouve des **NA**, plutôt que de simplement enlever ces lignes sans nous en parler.

La raison est que la valeur d'AIC dépend du nombre d'observations dans notre tableau de données. Si on ne met pas cette option, il pourrait arriver que certaines lignes soient absentes de certains modèles à cause de valeurs manquantes, mais apparaissent dans d'autres où ces variables ne font pas partie du modèle, ce qui fausserait l'interprétation des valeurs d'AIC.

```
modele_complet <- lm(
  body_mass_g ~ flipper_length_mm + bill_length_mm +
↪  bill_depth_mm,
  data = pour_regression_multiple,
  na.action = na.fail
)
```

Ensuite, une fois ce modèle complet ajusté, on peut demander à **MuMIn** de calculer l'AICc pour toutes les combinaisons de variables possibles, avec la fonction **dredge** (littéralement draguer, comme dans ramasser tout ce qu'on trouve au fond de la piscine à modèles).

```
liste_modeles <- dredge(modele_complet)
```

```
Fixed term is "(Intercept)"
```

On peut ensuite explorer le tableau résultant :

```
liste_modeles
```

```
Global model call: lm(formula = body_mass_g ~
flipper_length_mm + bill_length_mm +
    bill_depth_mm, data = pour_regression_multiple,
    na.action = na.fail)
---
Model selection table
    (Int) bll_dpt_mm bll_lng_mm flp_lng_mm df
6 -6542.0      22.63                 51.54  4
5 -5781.0                            49.69  3
8 -6425.0      20.05      4.162      50.27  5
7 -5737.0                  6.047      48.14  4
4  3343.0    -142.70     75.280             4
3   362.3                87.420             3
2  7489.0    -191.60                       3
1  4202.0                                  2
     logLik   AICc  delta weight
6 -2526.968 5062.1   0.00  0.385
5 -2528.427 5062.9   0.87  0.249
8 -2526.660 5063.5   1.44  0.187
7 -2527.741 5063.6   1.55  0.178
4 -2662.910 5333.9 271.88  0.000
3 -2696.987 5400.0 337.99  0.000
2 -2728.667 5463.4 401.35  0.000
1 -2771.748 5547.5 485.48  0.000
Models ranked by AICc(x)
```

Ce tableau continent une ligne par modèle testé. Les premières colonnes [(Int),bll_dpt_mm, etc.] contiennent les estimés de pentes pour chacune des variables dans le modèle, incluant l'ordonnée à l'origine. Remarquez que la fonction a fait son possible pour raccourcir le nom de nos variables pour réussir à les entrer dans le tableau.

Les dernières colonnes (à partir de `df`) contiennent les statistiques d'ajustement. Celles qui nous intéressent sont les 3 dernières, soit l'AICc, le delta AICc et le poids d'Akaike.

Selon le poids d'Akaike (la colonne weight), le modèle le plus probable pour nos données serait celui contenant l'épaisseur du bec et la longueur des ailes. Ce modèle a seulement 38,5% des chances d'être le meilleur. Tout près derrière se trouve le modèle contenant uniquement la longueur des ailes (delta = 0,87), qui a lui 24,9% des chances d'être le meilleur. Le meilleur modèle est donc à peine une fois et demie plus probablement que le second modèle (`0.39/0.25=1.55`). Deux autres modèles sont aussi raisonnablement bons derrière, avec des poids d'Akaike de 0,187 et 0,178.

Comme c'est souvent le cas en écologie, on a donc plusieurs modèles presque équivalents, et il est difficile de trancher. C'est donc ici où l'approche d'inférence multimodèle prendre toute son importance pour bien gérer ces nuances.

> **Note**
>
> Remarquez que contrairement à l'approche basée sur les tests d'hypothèses, il n'y a pas de seuil pré-établi au delà duquel on peut considérer qu'un modèle se distingue clairement des autres. Est-ce qu'un poids d'Akaike de 0,80 est suffisant? 0,90? Vous devrez utiliser votre jugement...

Comme vous le remarquez dans le tableau fourni par `dredge`, les estimés de chacun des paramètres varient d'un modèle à l'autre. Entre autres,

l'estimé pour la pente partielle de la longueur des ailes est de 51,54 dans le premier modèle, 49,69 dans le second et ainsi de suite. Quelle est alors la valeur la plus probable pour nos estimés de paramètres?

On peut demander à `MuMIn` de nous calculer une valeur pondérée, qui tient compte du fait qu'il y a 38% des chances que la vraie valeur soit 51,54, 25 % des chances que ce soit 48,69, etc. La fonction pour le faire se nomme `model.avg` (pour *Model Averaging*) :

```
modele_moyen <- model.avg(liste_modeles)
summary(modele_moyen)
```

```
Call:
model.avg(object = liste_modeles)

Component model call:
lm(formula = body_mass_g ~ <8 unique rhs>, data
     = pour_regression_multiple, na.action = na.fail)

Component models:
       df   logLik    AICc  delta weight
13      4 -2526.97 5062.06   0.00   0.39
3       3 -2528.43 5062.93   0.87   0.25
123     5 -2526.66 5063.50   1.44   0.19
23      4 -2527.74 5063.60   1.55   0.18
12      4 -2662.91 5333.94 271.88   0.00
2       3 -2696.99 5400.05 337.99   0.00
1       3 -2728.67 5463.41 401.35   0.00
(Null)  2 -2771.75 5547.53 485.48   0.00

Term codes:
    bill_depth_mm    bill_length_mm flipper_length_mm
                1                 2                 3
```

```
Model-averaged coefficients:
(full average)
                   Estimate Std. Error Adjusted SE
(Intercept)       -6186.910    590.129     591.417
bill_depth_mm        12.478     14.836      14.861
flipper_length_mm    50.236      2.301       2.307
bill_length_mm        1.855      4.050       4.059
                  z value Pr(>|z|)
(Intercept)        10.461   <2e-16 ***
bill_depth_mm       0.840    0.401
flipper_length_mm  21.776   <2e-16 ***
bill_length_mm      0.457    0.648

(conditional average)
                   Estimate Std. Error Adjusted SE
(Intercept)       -6186.910    590.129     591.417
bill_depth_mm        21.789     13.472      13.520
flipper_length_mm    50.236      2.301       2.307
bill_length_mm        5.081      5.341       5.359
                  z value Pr(>|z|)
(Intercept)        10.461   <2e-16 ***
bill_depth_mm       1.612    0.107
flipper_length_mm  21.776   <2e-16 ***
bill_length_mm      0.948    0.343
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Il y a BEAUCOUP de matériel dans ce tableau, mais les premières sections sont essentiellement redondantes avec les chiffres que nous avons vu précédemment dans la sortie de `dredge`. La section qui nous intéresse est celle nommée Model-averaged coefficients. Vous remarquerez que cette section ressemble beaucoup à celle fournie par une régres-

sion linéaire multiple, mais qu'elle comporte deux parties (*full average* et *conditional average*).

La présence de deux séries de chiffres s'explique par une question : qu'est-ce qu'on fait avec notre moyenne pour les modèles où la variable n'était pas présente? Dans la section *full average*, MuMIn assume que pour ces fois, la valeur de la pente était de zéro. Au contraire, dans la section *conditional average*, le calcul ne fait la moyenne que des fois où le terme était présent dans le modèle. L'estimé dans full average est plus conservateur (proche de zéro), alors que celui de conditional average nous informe que, si la variable est importante, voici ce que serait sa valeur.

On peut aussi tenir compte de cette nuance lorsque l'on calcule les intervalles de confiance de nos paramètres :

```
confint(modele_moyen, full = TRUE)
```

```
                        2.5 %       97.5 %
(Intercept)       -7346.065814 -5027.754362
bill_depth_mm       -16.649717    41.604766
flipper_length_mm    45.714324    54.757556
bill_length_mm       -6.099579     9.810534
```

```
confint(modele_moyen)
```

```
                        2.5 %      97.5 %
(Intercept)       -7346.065814 -5027.75436
bill_depth_mm        -4.709035    48.28683
flipper_length_mm    45.714324    54.75756
bill_length_mm       -5.423470    15.58457
```

Remarquez que pour les variables qui font clairement partie du meilleur modèle (i.e. la longueur des ailes) les valeurs sont très semblables, alors que pour les autres, on observe une grande différence.

Enfin, si on veut calculer l'importance relative de chaque variable (sa probabilité de faire partie du meilleur modèle), on peut lancer la fonction nommée **sw** (*sum of weights*) sur la liste de modèles fournie par dredge :

```
sw(liste_modeles)
```

```
                    flipper_length_mm bill_depth_mm
Sum of weights:     1.00              0.57
N containing models:    4                 4
                    bill_length_mm
Sum of weights:     0.37
N containing models:    4
```

On voit dans ces résultats que la longueur des ailes fait clairement partie du meilleur modèle (100% des chances), l'épaisseur du bec à plus de la moitié des chances (57%) et la longueur du bec n'en fait probablement pas partie (seulement 37% des chances).

Ces informations sont nettement plus nuancées que celles provenant des tests d'hypothèses ou de l'approche par le R^2-ajusté, qui tranchaient de façon beaucoup plus drastique.

Comme mentionné dans la présentation de l'AIC, ce dernier ne nous informe pas si un modèle est bon ou non. Il ne fait que nous informer sur la performance relative de modèles les uns comparés aux autres. Il faut donc calculer un r^2 pour avoir une idée de la performance globale de notre modèle.

La question ici est de savoir quelles variables inclure dans ce modèle. Il serait légitime d'utiliser les variables présentes dans le modèle présentant le meilleur AIC, mais comme les 3 variables ont une chance raisonnable de faire partie du meilleur modèle, il serait tout aussi correct de le faire avec un modèle contenant les 3 variables.

Comme ces modèles sont très près les uns des autres, le R^2 sera très semblable entre ces modèles de toute façon.

```
modele_final <- lm(body_mass_g ~
                     flipper_length_mm + bill_depth_mm,
                   data = pour_regression_multiple)
summary (modele_final)
```

```
Call:
lm(formula = body_mass_g ~ flipper_length_mm +
bill_depth_mm,
    data = pour_regression_multiple)

Residuals:
     Min       1Q   Median       3Q      Max
-1029.78  -271.45   -23.58   245.15  1275.97

Coefficients:
                   Estimate Std. Error t value
(Intercept)       -6541.907    540.751 -12.098
flipper_length_mm    51.541      1.865  27.635
bill_depth_mm        22.634     13.280   1.704
                  Pr(>|t|)
(Intercept)         <2e-16 ***
flipper_length_mm   <2e-16 ***
bill_depth_mm       0.0892 .
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 393.2 on 339 degrees of freedom
Multiple R-squared:  0.761, Adjusted R-squared:  0.7596
F-statistic: 539.8 on 2 and 339 DF,  p-value: < 2.2e-16
```

Notre modèle explique donc 76 % de la variance dans le poids des manchots

28.8. Dans la vraie vie?

Ok, mais là Charles, ça fait un sacré paquet de méthodes pour faire essentiellement la même tâche (et encore, nous en avons laissé plusieurs combinaisons de côté et nous n'avons même pas parlé de l'approche bayésienne!). Laquelle on prend dans la vraie vie?

Impossible de savoir ce que vous aurez à affronter en recherche ou dans votre milieu de travail. Nous sommes dans une période de transition où les tests d'hypothèses sont encore grandement utilisés, le R^2-ajusté est utilisé dans beaucoup d'articles et l'AIC et l'inférence multi-modèle prennent de plus en plus de place.

Selon moi, l'inférence multi-modèle avec l'AIC est l'approche la plus utile en écologie, puisqu'elle intègre directement l'incertitude entourant l'approche observationnelle (la plus commune en écologie).

Les tests d'hypothèses ont toujours leur place dans des expériences contrôlées où les facteurs confondants sont correctement pris en compte (par exemple en laboratoire).

Enfin, l'explication de la variance demeurera un but important en écologie, i.e. est-ce qu'on comprend bien ou non un phénomène? Donc les R^2-ajustés sont assurément là pour rester aussi, particulièrement quand ce qui nous intéresse est la qualité des prédictions.

Une chose est claire par contre, pour chacun de vos projets, choisissez à l'avance la technique que vous allez utiliser, et tenez-vous y. Il est relativement mal vu dans un même projet d'essayer plusieurs méthodes différentes pour sélectionner le meilleur modèle, puisque vous augmentez de beaucoup votre risque de choisir la méthode dont les résultats

correspondent à vos attentes plutôt que celle décrivant le mieux la réalité...

28.9. Après avoir choisi le meilleur modèle?

Après avoir choisi le meilleur modèle avec l'aide d'une des méthodes ci-haut, il peut arriver que ce modèle soit différent du modèle complet que vous aviez validé initialement. Dans ce cas, il faut aussi valider ce modèle final (inspection des résidus, colinéarité, valeurs aberrantes, etc.). Les choses peuvent avoir drôlement changé en enlevant des variables.

28.10. Exercice sur la sélection de modèles

À partir du modèle construit dans le chapitre sur la régression multiple pour expliquer le l'abondance des oiseaux dans une paracelle (Section 27.11), appliquez les deux stratégies de sélection du meilleur modèle suivantes :

- Méthode par élimination, basée sur les tests d'hypothèses nulles,
- Inférence multi-modèle, basée sur l'AIC.

Comparez le résultat de vos deux modèles. Vos conclusions sont-elles différentes ou essentiellement identiques?

Laquelle des deux stratégies correspond le mieux à votre façon de voir la méthode scientifique?

29. La modélisation des variables qualitatives

29.1. Introduction

Dans les chapitres précédents (Chapitre 15,Chapitre 16), nous avons vu que pour modéliser la relation entre une variable quantitative et une variable qualitative, on devait utiliser une méthode statistique nommée ANOVA. Vous avez peut-être plutôt retenu que l'ANOVA est une méthode permettant de comparer 2+ variables quantitatives. Il s'agit en fait de deux façons différentes de décrire le même problème.

Nous avons aussi vu au Chapitre 18 que, lorsque l'on veut analyser la relation entre deux variables quantitatives, on doit plutôt utiliser la régression linéaire.

Aujourd'hui je vais vous révéler un grand secret : l'ANOVA et la régression linéaire sont en fait une seule et même procédure statistique. Eh oui! Si vous fouillez dans l'aide de la fonction `aov` dans R, vous y apprendrez qu'elle ne sert qu'à emballer (*to wrap*) la fonction `lm` et présenter les sorties comme on les attend dans une ANOVA.

Les deux méthodes (ANOVA et régression linéaire) ont été développées indépendamment l'une de l'autre et sont encore souvent enseignées de cette façon, car cela simplifie beaucoup leur compréhension, mais cette séparation n'est pas nécessaire. Cette révélation est d'ailleurs la clé pour comprendre comment combiner à la fois des variables quantitatives et

qualitatives dans un même modèle, comme nous allons le faire dans ce chapitre.

29.2. Le secret est dans l'encodage

Pour pouvoir intégrer des variables qualitatives à un modèle de régression, il faut les transformer en chiffres, puisque c'est tout ce que la régression comprend. Il existe plusieurs façons de le faire, mais nous verrons dans ce cours uniquement la façon la plus commune : le **dummy coding** (désolé, je n'ai pas trouvé d'équivalent français intéressant à ce terme).

Cet encodage consiste à transformer notre variable qualitative de k valeurs (que l'on appellera désormais niveaux) en k-1 nouvelles variables qui contiennent les valeurs 0 ou 1 plutôt que du texte. Vous allez voir, avec un exemple, ça s'éclaircit...

Imaginons un tableau de données où nous avons une variable qualitative qui nous dit à quel animal correspond chaque ligne et que nous avons observé des chats, des chiens et des poissons :

Nom	Animal
Garfield	chat
Fido	chien
Hello Kitty	chat
Azraël	chat
Némo	poisson

L'encodage *dummy* transformera notre variable Animal en 2 nouvelles variables, soit animalChien et animalPoisson. Les lignes où nous avions effectivement un chien, la variable animalChien aura la valeur 1 et animalPoisson 0 et vice versa. Les lignes contenant un chat contiendront un

0 dans les deux variables. Voici le même tableau de données, mais avec les variables provenant de l'encodage *dummy* :

Nom	animalChien	animalPoisson
Garfield	0	0
Fido	1	0
Hello Kitty	0	0
Azraël	0	0
Némo	0	1

Vous aurez rarement à effectuer cet encodage vous-même, la plupart des fonctions statistiques de R le feront pour vous, mais il est très important de comprendre le fonctionnement pour bien interpréter les sorties et les éventuelles erreurs provenant du modèle, en particulier au niveau du manque de degrés de libertés.

29.3. Que se passera-t-il dans le modèle?

Comme nous l'avons vu, un des niveaux de notre variable qualitative n'aura pas de variable dummy correspondante (les chats dans notre exemple).

> **!** Important
>
> La valeur de ce niveau manquant correspondra à l'ordonnée à l'origine de notre modèle.

On la nommera souvent **niveau de référence**. Le coefficient (syn. pente) associé à animalChien et animalPoisson nous informera respectivement de la différence entre un chien et un chat et de la différence entre un poisson et un chat.

Je le répète à nouveau car c'est la clé pour comprendre les variables qualitatives dans un modèle de régression multiple : tous les estimés de paramètres des variables dummy correspondent à des différences par rapport au niveau de référence (l'ordonnée à l'origine). Un estimé négatif pour animalChien ne signifie PAS une prédiction négative. Il signifie uniquement que la prédiction pour les chiens est plus petite que la prédiction pour les chats (notre référence).

29.4. Labo : Les variables qualitatives dans R

Nous avons discuté brièvement de ce sujet par le passé, mais dans R, il existe deux façons de stocker les variables qualitatives. Elles sont soit en format texte (`chr`), ou déjà encodées et prêtes à être entrées dans un modèle (`factor`).

Dans beaucoup de cas, cela ne fera pas vraiment de différence puisque beaucoup de fonctions R passent d'un format à l'autre sans vous en parler. Mais certaines fonctions comme la manipulation de texte s'effectuent mieux en format `chr`, alors que d'autres exigeront que vos variables soient en format `factor`, entre autres certaines fonctions de modélisation avancées.

Aussi, prenez note que par le passé, les fonctions qui créaient des tableaux de données convertissaient automatiquement les valeurs qualitatives en `factor`, mais qu'à partir de la version 4.0 de R, ce n'est plus le cas. Par contre, si vous utilisiez les fonctions fournies avec le `tidyverse`, alors les données sont toujours en format `chr`... Je sais, c'est le foutoir!

Ma recommandation est de travailler le plus possible vos données en format `chr`, et de les convertir au format `factor` à la toute dernière étape (ou presque...) avant de lancer votre modèle.

Comme nous avons discuté plus haut, l'encodage des variables qualitatives est fait de manière à ce qu'une des valeurs de notre variable soit

considérée comme le niveau de référence. Par défaut, si on ne dit rien à R, il prendra la première valeur qu'il a trouvée dans le tableau comme référence.

Dans le tableau de données `penguins`, les variables d'espèce et de sexe sont déjà correctement encodées, mais nous ferons comme si elles ne l'étaient pas, pour que vous puissiez voir comment faire :

```
library(palmerpenguins)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
qualitatives <-
  penguins |>
  drop_na(species, sex, body_mass_g, flipper_length_mm)
   ↪  |>
  mutate(
    species = as_factor(species),
    sex = as_factor(sex)
  )

levels(qualitatives$species)
```

```
[1] "Adelie"    "Chinstrap" "Gentoo"
```

```
levels(qualitatives$sex)
```

```
[1] "female" "male"
```

Pour connaître le niveau de référence d'un facteur dans R, on utilise la fonction `levels`. Le premier élément de chaque sortie est celui que R considère comme la référence pour cette variable. Le niveau de référence pour l'espèce sera donc "Adelie", et celui pour le sexe sera "female".

On peut changer le niveau de référence avec la fonction `relevel`, comme ceci :

```
qualitatives <-
  qualitatives |>
  mutate(
    species = relevel(species,"Gentoo"),
    sex = relevel(sex,"male")
  )

levels(qualitatives$species)
```

```
[1] "Gentoo"    "Adelie"    "Chinstrap"
```

```
levels(qualitatives$sex)
```

```
[1] "male"   "female"
```

Nos niveaux de référence sont maintenant "Gentoo" et "male".

Vous remarquerez qu'il n'est pas nécessairement simple de déterminer quel devrait être le niveau de référence pour notre analyse. Parfois c'est très logique, par exemple quand on a des parcelles contrôles et d'autres

avec des traitements, mais parfois c'est très arbitraire comme choix, comme ici avec nos espèces.

29.5. Attention aux degrés de liberté

À partir de maintenant, il faut se soucier plus attentivement des degrés de liberté de nos modèles lorsque nous ferons nos analyses.

> ⚠ Avertissement
>
> Par le dummy coding, la régression linéaire ajuste un paramètre pour chacun des niveaux de chacune de nos variables qualitatives, moins un pour le niveau de référence.

Donc, même si dans notre modèle on inscrit quelque chose qui semble tout simple comme :

```
croissance = taille + espece
```

Si notre variable espece contient 8 valeurs différentes, notre modèle devra ajuster 7 paramètres différents pour l'espèce, en plus de l'ordonnée à l'origine et de la pente pour la taille. On utilisera 9 degrés de liberté. Le nombre de degrés de liberté nécessaires peut augmenter très rapidement.

En passant, sachez que pour que notre modèle soit estimé correctement, on recommande d'avoir au minimum 3 observations par niveau de notre variable qualitative (p. ex. 3 individus pour chaque espèce dans l'exemple précédent).

29.6. Labo : Les variables qualitatives dans une régression multiple

Pour ce laboratoire, nous allons modéliser les facteurs affectant le poids des manchots. Comme nous l'avons vu dans d'autres chapitres, la longueur des ailes est un facteur important contrôlant le poids des manchots. Nous allons donc conserver cette variable, mais aussi ajouter le sexe et l'espèce de l'individu, afin d'avoir un portrait plus global de la situation.

Nous passerons outre les étapes préliminaires de vérification des données pour simplifier ce chapitre, mais dans un vrai travail, il serait important de bien vérifier vos données, la forme des relations et les histogrammes de distributions avant de commencer. Il est aussi important avant de lancer la modélisation de bien réfléchir à quels niveaux des variables qualitatives serviront de référence. Ici, nous utiliserons "male" et "Gentoo", comme nous l'avons déjà préparé ci-haut.

Voici donc comment nous pourrions ajuster un tel modèle dans R :

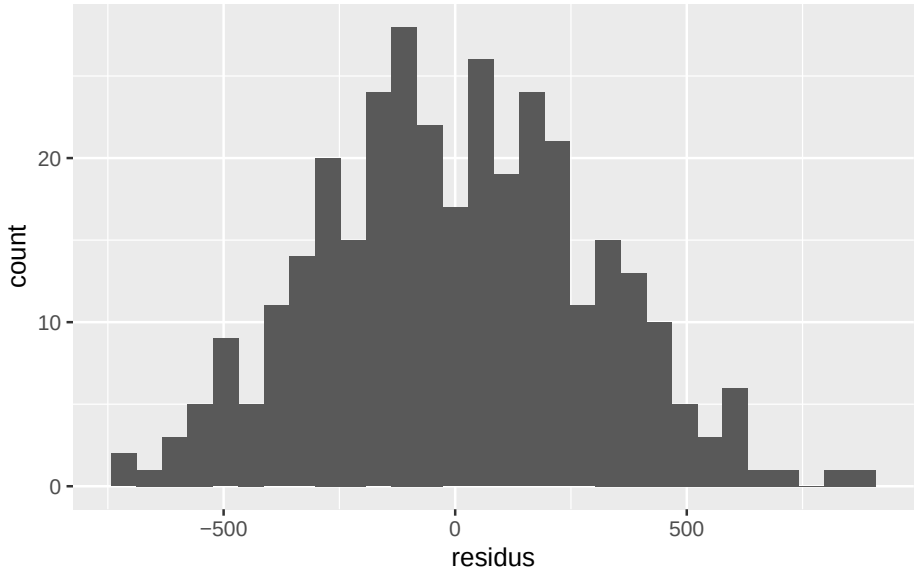
```
modele <- lm(body_mass_g ~
               flipper_length_mm + sex + species,
             data = qualitatives)
```

Ensuite, il faut, comme toujours, bien valider notre modèle avant de procéder à son interprétation :

```
qualitatives <- qualitatives |>
  mutate(
    residus = resid(modele) ,
    predictions = predict(modele),
    D = cooks.distance(modele)
  )
```

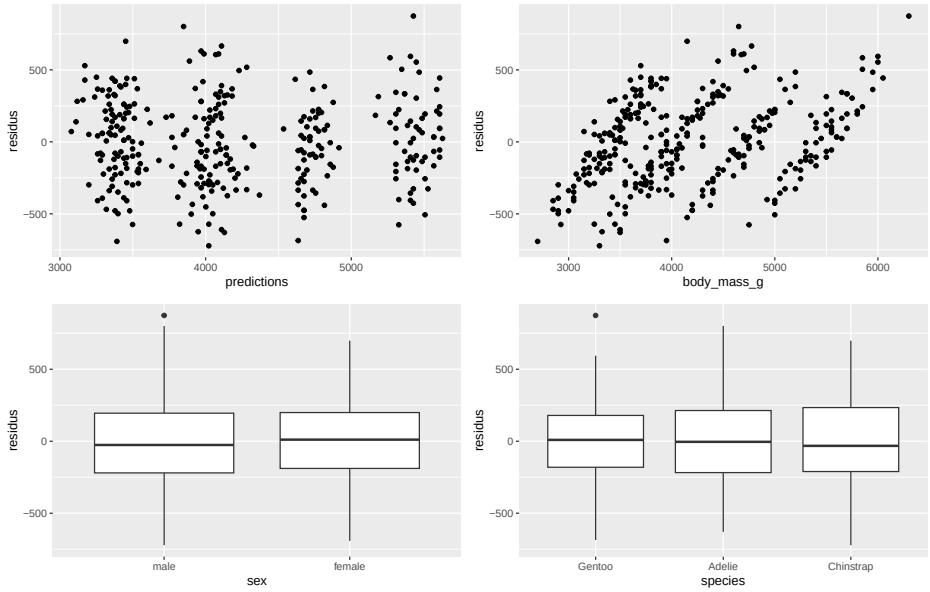
```
ggplot(qualitatives, aes(x = residus)) +
 ↪  geom_histogram()
```

```
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```



Tout a l'air normal ici

```
ggplot(qualitatives, aes(x=predictions,y=residus)) +
 ↪  geom_point()
ggplot(qualitatives, aes(x=body_mass_g,y=residus)) +
 ↪  geom_point()
ggplot(qualitatives, aes(x=sex,y=residus)) +
 ↪  geom_boxplot()
ggplot(qualitatives, aes(x=species,y=residus)) +
 ↪  geom_boxplot()
```



Remarquez bien que pour les variables qualitatives, l'inspection de l'homogénéité des résidus doit se faire avec des diagrammes à moustache plutôt que des nuages de points. On cherche dans ce cas à s'assurer que les boîtes ont à peu près la même taille, et sont à peu près toutes centrées sur zéro. On peut tolérer certaines différences, mais si c'est trop extrême, il y a probablement un problème avec notre modèle.

Il faut aussi s'assurer qu'aucune observation n'avait d'influence trop importante :

```
qualitatives |> filter(D>1)
```

```
# A tibble: 0 x 11
# i 11 variables: species <fct>, island <fct>,
```

```
#   bill_length_mm <dbl>, bill_depth_mm <dbl>,
#   flipper_length_mm <int>, body_mass_g <int>,
#   sex <fct>, year <int>, residus <dbl>,
#   predictions <dbl>, D <dbl>
```

Et vérifier que la colinéarité entre nos variables était raisonnable :

```
library(car)
```

```
Loading required package: carData


Attaching package: 'car'

The following object is masked from 'package:dplyr':

    recode

The following object is masked from 'package:purrr':

    some
```

```
vif(modele)
```

```
                     GVIF Df GVIF^(1/(2*Df))
flipper_length_mm 6.045957  1        2.458853
sex               1.362260  1        1.167159
species           5.653123  2        1.541956
```

Remarquez qu'ici, la fonction vif nous présente une statistique nommée **GVIF** plutôt que le VIF auquel nous étions habitués. Ce changement est nécessaire parce que l'encodage des variables qualitatives (le *dummy coding*) dans le modèle génère plusieurs variables, qui sont artificiellement

corrélées entre elles. Il faut donc utiliser un calcul alternatif nommé *Generalized Variance Inflation* Factor pour évaluer la colinéarité dans notre modèle[1].

Nous avons vu dans le Chapitre 27 sur la régression multiple que la racine carrée du VIF représentait un facteur multiplicatif de l'erreur-type de chacun des paramètres. Pour les variables qualitatives, les statisticiens recommandent de remplacer la racine carrée par la racine 2*d.d.l. pour obtenir un facteur multiplicatif représentatif.

Une fois tous ces détails racontés, l'important de savoir est que la valeur de GVIF s'interprète exactement comme la valeur de VIF normale!

Ici, on ne s'inquiète pas trop pour le sexe et l'espèce, mais l'erreur-type du paramètre pour la longueur des ailes et près de deux fois et demi (2,46) plus important que si on n'avait pas d'ennui de collinéarité. Il serait donc important de nuancer nos conclusions concernant ce paramètre dans la présentation des résultats.

On peut maintenant regarder les résultats de notre modèle :

```
summary(modele)
```

```
Call:
lm(formula = body_mass_g ~ flipper_length_mm + sex +
species,
    data = qualitatives)

Residuals:
   Min     1Q Median     3Q    Max
-721.8 -195.7   -5.9  198.6  873.7

Coefficients:
```

[1] https://www.jstor.org/stable/2290467

```
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       1000.824    628.485   1.592    0.112
flipper_length_mm   20.025      2.846   7.037 1.15e-11
sexfemale         -530.381     37.810 -14.027  < 2e-16
speciesAdelie     -836.260     85.185  -9.817  < 2e-16
speciesChinstrap  -923.894     75.511 -12.235  < 2e-16

(Intercept)
flipper_length_mm ***
sexfemale         ***
speciesAdelie     ***
speciesChinstrap  ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 295.6 on 328 degrees of freedom
Multiple R-squared:  0.8669,    Adjusted R-squared:
0.8653
F-statistic:   534 on 4 and 328 DF,  p-value: < 2.2e-16
```

Les sorties sont au même format que pour la régression multiple, mais on a maintenant beaucoup plus de paramètres dans notre modèle (donc beaucoup plus de lignes dans notre sortie!).

La première ligne, comme à l'habitude est celle de l'ordonnée à l'origine (Intercept). Ici par contre, elle correspond non seulement à la valeur en Y lorsque toutes les variables quantitatives sont à zéro, mais aussi lorsque toutes les variables qualitatives sont à leur niveau de référence (ici species="Gentoo" et sex="male").

On trouve d'abord la pente partielle associée à la longueur des ailes, qui nous indique que plus les ailes sont longues, plus le manchot sera lourd.

Dans la section suivante, on retrouve le paramètre décrivant la différence entre le sexe femelle et le sexe des référence (mâle). Puisque le coefficient est négatif, il nous indique que les femelles sont, en moyenne, 530 g plus légères que les mâles.

Enfin, dans la dernière section, on a les paramètres décrivant la différence de poids entre l'espèce Gentoo et les deux autres espèces. On constate que les manchots Adélie sont 836 g plus légers que les Gentoo, et que les manchots Chinstrap sont 923 g plus légers que les Gentoo.

Dans tous les cas, ces paramètres sont significativement différents de zéro, i.e. les différences et les pentes sont clairement différentes de zéro.

> **Note**
>
> Remarquez que le signe des paramètres dépend de la valeur de référence. Si le niveau de référence du sexe avait été femelle, on aurait eu un paramètre nommé `sexmale`, qui aurait eu la valeur +530.

Toutes les techniques de sélection de modèle vues dans le Chapitre 28 sont aussi valides ici.

> **Avertissement**
>
> Par contre, si vous prévoyez utiliser une approche basée sur l'explication de la variance et explorer la liste exhaustive de tous les modèles, il est important de savoir que la fonction `regsubsets` de la librairie `leaps` ne comprend pas l'encodage dummy.

Elle ne peut pas être utilisée si votre modèle comprend une variable qualitative. Par contre, avec une petite modification, on peut facilement exploiter la librairie `MuMIn` pour faire le même travail, par exemple :

```
library(MuMIn)
modele <- lm(body_mass_g ~
               flipper_length_mm + sex + species,
             data = qualitatives,
             na.action = "na.fail"
             )
liste <- dredge(modele, extra= "adjR^2")
```

```
Fixed term is "(Intercept)"
```

```
liste |> arrange(desc(`adjR^2`))
```

```
Global model call: lm(formula = body_mass_g ~
flipper_length_mm + sex + species,
    data = qualitatives, na.action = "na.fail")
---
Model selection table
     (Int) flp_lng_mm sex spc adjR^2 df    logLik
[1,]  1001      20.02   +   + 0.8669  6 -2364.387
[2,]  5418              +   + 0.8468  5 -2387.797
[3,] -5062      46.98   +     0.8058  4 -2427.242
[4,] -3729      40.61       + 0.7870  5 -2442.633
[5,] -5872      50.15         0.7621  3 -2461.073
[6,]  5092                  + 0.6745  4 -2513.273
[7,]  4546              +     0.1806  3 -2666.979
[8,]  4207                    0.0000  2 -2700.146
       AICc  delta weight
[1,] 4741.0   0.00      1
[2,] 4785.8  44.75      0
[3,] 4862.6 121.57      0
[4,] 4895.5 154.42      0
[5,] 4928.2 187.19      0
[6,] 5034.7 293.64      0
```

```
[7,] 5340.0 599.00      0
[8,] 5404.3 663.30      0
Models ranked by AICc(x)
```

Ici, le R^2-ajusté aurait sélectionné le modèle contenant les 3 variables, et la technique basée sur l'AIC serait arrivée au même résultat.

Remarquez que j'ai dû refaire mon objet modele avec la mention `na.action="na.fail"` puisque je n'avais pas fait précédemment.

29.7. Le concept d'effet aléatoire

Il est possible de classifier les variables qualitatives en deux grands groupes : les effets fixes et les effets aléatoires. La différence est principalement au niveau de l'interprétation que nous voulons faire de la variable. Dans le cas d'**effets fixes**, la valeur estimée pour chaque niveau de la variable qualitative nous intéresse. Pour les **effets aléatoires**, la valeur de chaque niveau ne nous intéresse pas vraiment. Ce qui nous intéresse est de savoir comment l'effet peut être variable entre les différents niveaux.

Voici quelques pistes pour mieux reconnaître un effet fixe et un effet aléatoire :

Effet fixe :

- Nous avons utilisé tous les niveaux disponibles de la variable
- Si on répétait l'expérience, on réutiliserait probablement les mêmes niveaux
- On ne veut pas extrapoler aux niveaux que nous n'avons pas observés
- Un exemple classique serait une variable TypeDeFeuille contenant les valeurs “conifère” et “feuillu”

Effet aléatoire :

- Nous n'avons utilisé qu'une partie (aléatoire) des niveaux disponibles
- Si on répétait l'expérience, on utiliserait probablement des niveaux différents
- Notre inférence s'applique à tous les niveaux disponibles, incluant ceux que nous n'avons pas observés.
- Un exemple classique serait une variable Site

Si notre variable est Sexe et contient "Mâle"/"Femelle"/"Autre", elle est clairement un effet fixe. Si notre variable se nomme Individu et contient le numéro du spécimen, elle est clairement un effet aléatoire.

Notez que la différence est uniquement conceptuelle. Une variable contenant le nom du lac étudié pourrait, pour un étudiant au doctorat, s'interpréter comme un effet aléatoire. Si il refaisait son expérience, il pourrait la faire ailleurs, etc. Par contre, si un biologiste d'un organisme de bassin versant faisait la même expérience, les valeurs individuelles de chaque lac pourraient l'intéresser. Il s'agirait alors d'un effet fixe.

Ce chapitre s'interessera uniquement aux variables qualitatives utilisées en effets fixes. Les effets aléatoires seront discutés dans le Chapitre 30 sur les modèles linéaires mixtes.

29.8. Les interactions

Le concept d'interaction en statistique possède une définition relativement simple, mais qui demande une certaine réflexion. On dit qu'il existe une **interaction** entre deux variables, lorsque l'effet d'une variable dépend de la valeur de l'autre. Notez que l'on parle ici d'interaction **entre les variables explicatives**. Il est implicite que toutes les variables explicatives interagissent avec la variable expliquée...

Dans le cas de variables qualitatives

Prenons l'exemple d'une expérience où l'on désire observer la productivité dans des parcelles de plantes et que l'on désire étudier l'effet de l'arrosage ("Avec"/"Sans") et l'effet d'un ajout d'engrais ("Avec"/"Sans"). Nous avons deux variables qualitatives, qui ont chacune deux niveaux.

Nous avons mesuré les moyennes suivantes :

	Sans engrais	Avec engrais
Sans arrosage	25 g m^{-2}	50 g m^{-2}
Avec arrosage	40 g m^{-2}	?

Si il n'existe PAS d'interaction entre l'ajout d'engrais et l'arrosage sur la productivité, on devrait s'attendre à mesurer 65 g m^{-2} dans les parcelles avec engrais ET arrosage (25 g m^{-2} + 15 g m^{-2} pour l'effet de l'arrosage + 25 g m^{-2} pour l'effet de l'engrais). Par contre, si on mesure autre chose, par exemple 110 g m^{-2}, alors il y a interaction entre les deux traitements. Les deux traitements ensemble n'ont pas le même effet que s'ils étaient appliqués indépendamment.

Notez que l'interaction pourrait aussi être négative, elle n'est pas obligée d'augmenter nécessairement l'effet.

Entre une variable qualitative et une variable quantitative

Si l'on reprend notre définition d'une interaction : l'effet d'une variable dépend de la valeur d'une autre, il est possible de comprendre ce qui survient lorsque l'on tente de modéliser l'interaction entre une variable qualitative et une variable quantitative. C'est comme poser la question : est-ce que la pente associée à notre variable quantitative sera la même pour chaque niveau de notre variable qualitative ou si elle changera selon la valeur que prend notre variable qualitative.

Nous avons déjà abordé ce genre de modèle sous l'appellation d'ANCOVA (voir Chapitre 20). L'ANCOVA n'est pas un modèle spécial en tant que tel. C'est uniquement l'étiquette donnée à un modèle de régression multiple qui contient une variable qualitative, une seule variable quantitative et leur interaction. Rien n'empêcherait dans un modèle d'avoir aussi d'autres variables. Mais on ne pourrait plus parler d'ANCOVA comme tel.

Dans le modèle statistique

Lorsque l'on désire modéliser une interaction dans une régression multiple, celle-ci sera ajoutée comme un nouveau terme au modèle, qui sera calculé par le logiciel comme la multiplication des termes dont on veut modéliser l'interaction.

Par exemple, si notre modèle contient les variables A et B :

$$Y = A + B$$

L'interaction entre ces deux termes sera ajoutée comme :

$$Y = A + B + A \times B$$

On peut donc tester, par le principe des tests de F partiels (voir Chapitre 27), l'effet de l'ajout de ce terme d'interaction à notre modèle. Dans ce modèle, on peut donc tester 3 choses : est-ce que A est significatif, est-ce que B est significatif et est-ce que l'interaction entre A et B est significative.

Si cette interaction est significative, elle nous informe que l'effet des deux variables (A et B) n'est pas indépendant. Si tel est le cas, il devient hasardeux d'interpréter les pentes individuelles, puisque l'on vient de conclure qu'elles sont changeantes. On les conserve dans le modèle, mais on ne les interprète plus.

> **!** Important
>
> Attention : il ne faut par contre jamais ajuster un modèle avec une interaction sans les deux variables correspondantes.

Quand mettre une interaction dans notre modèle?

Il n'y a pas de réponse simple à cette question. Elle dépend en premier lieu de votre question biologique. Parfois conceptuellement ça a du sens de tester une interaction, parfois non.

L'autre chose à se demander est : avons-nous suffisamment de degrés de liberté pour estimer cette interaction? Dans notre exemple précédent où nous avions ce modèle avec 8 espèces qui nécessitait 9 degrés de liberté :

$$croissance = taille + espece$$

Si nous ajoutons une interaction entre taille et espèce dans le modèle :

$$croissance = taille + espece + taille \times espece$$

Nous aurons besoin à ce moment de 16 (!) degrés de liberté :

- 1 pour l'ordonnée à l'origine
- 1 pour la pente de la taille
- 7 pour les espèces (8 - 1)
- 7 pour les interactions taille x espèce (8 - 1)

29.9. Labo : Les interactions

Nous allons maintenant voir comment intégrer des interactions à nos modèles dans R. Comme ce chapitre commence à s'étirer un peu, nous n'allons pas valider le modèle dans cet exemple, mais dans la vraie, vous devrez le faire.

Les interactions dans R peuvent s'inscrire de deux façons, soit avec le **:**, ou soit avec le *****.

La notation avec le **:** ajoute uniquement l'interaction au modèle. Nous devons manuellement ajouter aussi les termes simples, par exemple :

```
Y ~ X1 + X2 + X1:X2
```

La notation avec le ***** ajoute à la fois l'interaction et les deux termes simples au modèle. Le modèle précédent, peut par exemple s'écrire comme ceci :

```
Y ~ X1 * X2
```

Bien que la notation avec le ***** ait l'avantage d'être plus courte, je vous conseille d'appliquer celle avec le **:**, puisque sa façon plus longue d'écrire nous invite à penser à la quantité de paramètres que nous sommes en train d'ajouter au modèle.

Comme nous avons beaucoup de degrés de liberté, nous nous permettrons d'ajouter deux interactions à notre modèle. Tout d'abord, nous modéliserons une interaction qualitative-qualitative entre le sexe et l'espèce. Celle-ci nous permettra de savoir si la différence entre les mâles et les femelles varie entre les espèces. Nous en ajouterons aussi une deuxième, qualitative-quantitative cette fois, entre l'espèce et la longueur des ailes. Cette dernière nous permettra de savoir si la pente entre le poids et la longueur des ailes varie entre les espèces. Autrement dit, si avoir des ailes plus longues est plus payant chez certaines espèces que chez d'autres.

```r
modele_avec_interactions <- lm(
  body_mass_g ~
    flipper_length_mm + sex + species +
    sex:species +
    species:flipper_length_mm,
  data = qualitatives,
  na.action = "na.fail"
)

summary(modele_avec_interactions)
```

```
Call:
lm(formula = body_mass_g ~ flipper_length_mm + sex +
species +
    sex:species + species:flipper_length_mm, data =
    qualitatives,
    na.action = "na.fail")

Residuals:
    Min      1Q  Median      3Q     Max
-834.27 -192.62   -6.27  195.18  827.88

Coefficients:
                                  Estimate
(Intercept)                        278.810
flipper_length_mm                   23.499
sexfemale                         -597.501
speciesAdelie                      625.620
speciesChinstrap                 -1672.347
sexfemale:speciesAdelie             -1.842
sexfemale:speciesChinstrap         403.838
flipper_length_mm:speciesAdelie     -7.185
```

```
flipper_length_mm:speciesChinstrap     3.175
                                  Std. Error t value
(Intercept)                         1203.209   0.232
flipper_length_mm                      5.429   4.329
sexfemale                             71.220  -8.389
speciesAdelie                       1419.531   0.441
speciesChinstrap                    1703.088  -0.982
sexfemale:speciesAdelie               87.502  -0.021
sexfemale:speciesChinstrap           111.119   3.634
flipper_length_mm:speciesAdelie        6.691  -1.074
flipper_length_mm:speciesChinstrap     8.109   0.392
                                  Pr(>|t|)
(Intercept)                       0.816900
flipper_length_mm                 2.00e-05 ***
sexfemale                         1.53e-15 ***
speciesAdelie                     0.659707
speciesChinstrap                  0.326857
sexfemale:speciesAdelie           0.983216
sexfemale:speciesChinstrap        0.000324 ***
flipper_length_mm:speciesAdelie   0.283680
flipper_length_mm:speciesChinstrap 0.695652
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 287.1 on 324 degrees of freedom
Multiple R-squared:  0.8759,    Adjusted R-squared:
0.8729
F-statistic: 285.9 on 8 and 324 DF,  p-value: < 2.2e-16
```

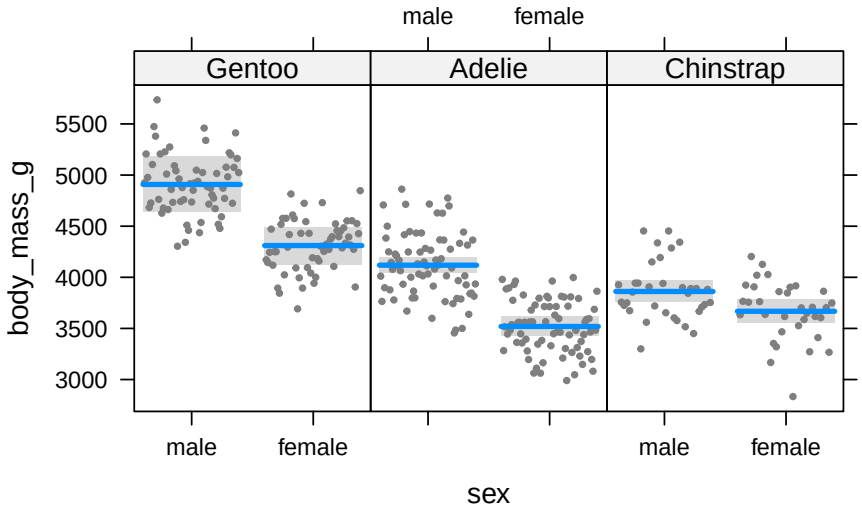
Pour interpréter ce tableau, il faut bien réaliser que maintenant, l'estimé de paramètre associé à sexfemale n'est plus un estimé global. Il correspond maintenant à la différence mâle-femelle pour notre niveau de référence de la variable species (i.e. les manchots Gentoo). La ligne `sexfe-`

`male:speciesChinstrap` nous informe, par exemple, que la différence mâle-femelle chez les manchots Chinstrap est 403 g plus élevée que celle des manchots Gentoo. Autrement dit, chez les Gentoo, les femelles sont 597 g plus légères que chez les mâles, mais chez les manchots Chinstrap, les femelles ne sont que (-597 g + 403 g) 194 g plus légères.

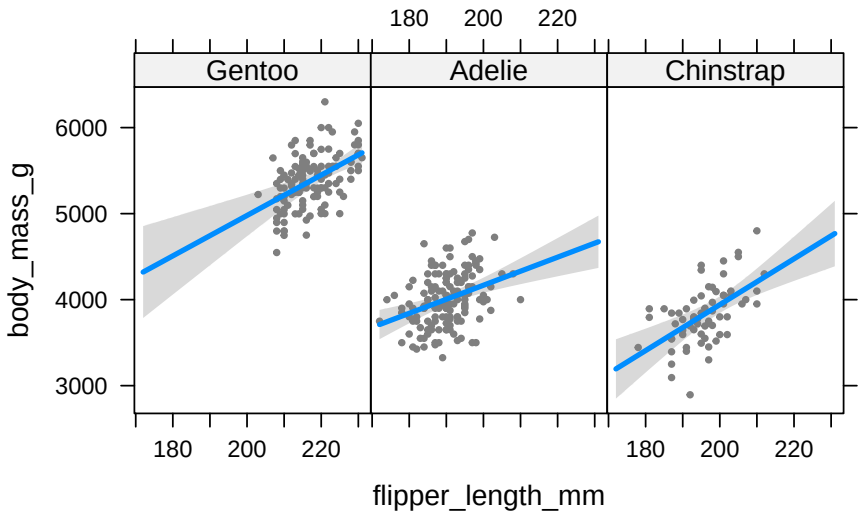
Les lignes `flipper_length_mm:speciesAdelie` et `flipper_length_mm:speciesChir` nous informent que la pente entre la taille du corps et la longueur des ailes est très sembable chez les 3 espèces. Celle pour les Gentoo est de 23.499 g/mm, celle pour les Adélie est (23.499-7.185) 16.313 g/mm et celle des manchots Chinstrap (23.499+3.175) es de 26.67 g/mm. Dans ce dernier cas, aucune des pentes de se distingue significativement de la pente de référence.

Comme les interactions sont un thème plutôt abstrait, il peut être utile de les visualiser pour mieux les comprendre. La fonction `visreg` est tout à fait appropriée pour ce genre d'exploration :

```
library(visreg)
#| layout-ncol: 2
visreg(modele_avec_interactions,"sex", by = "species")
```




```
visreg(modele_avec_interactions,"flipper_length_mm", by
 ↪  = "species")
```



On comprend mieux maintenant que les différences mâles-femelles sont presque inexisantes chez les manchots Chinstrap, mais très importantes dans les deux autres espèces. On voit aussi que les pentes poids-longueurs d'ailes sont très semblables entre les espèces. Les ordonnées à l'origine sont bien différentes, mais la pente comme tel ne se distingue pas vraiment.

Vous remarquerez peut-être que dans nos résultats, les différences entre les espèces (i.e. `speciesAdelie` et `speciesChinstrap`) ne sont plus considérées comme significativement différentes du niveau de référence. Cela arrive fréquemment, puisque maintenant, c'est l'interaction avec le sexe qui est significative. Il ne faut pas pour autant éliminer ce terme.

Prenez note aussi du fait que, bien que notre tableau de données contienne au départ 333 observations, il ne reste à notre modèle avec les interactions que 324 degrés de liberté, puisque nous devons ajuster 9 (!) paramètres pour répondre à notre question.

Enfin, on aurait pu aussi appliquer nos techniques de sélection de modèle sur ce modèles contenant les interaction :

```
liste_avec_interactions <-
 ↪  dredge(modele_avec_interactions, extra= "adjR^2")
```

```
Fixed term is "(Intercept)"
```

```
liste_avec_interactions |> arrange(desc(`adjR^2`))
```

```
Global model call: lm(formula = body_mass_g ~
flipper_length_mm + sex + species +
    sex:species + species:flipper_length_mm, data =
    qualitatives,
    na.action = "na.fail")
---
Model selection table
      (Int) flp_lng_mm sex spc flp_lng_mm:spc
 [1,]  278.8      23.50   +   +              +
 [2,]  946.3      20.49   +   +
 [3,] -604.9      27.40   +   +              +
 [4,] 1001.0      20.02   +   +
 [5,] 5485.0              +   +
 [6,] 5418.0              +   +
 [7,] -5062.0     46.98   +
 [8,] -6674.0     54.17       +              +
 [9,] -3729.0     40.61       +
[10,] -5872.0     50.15
[11,] 5092.0                  +
[12,] 4546.0              +
[13,] 4207.0
      sex:spc adjR^2 df    logLik   AICc  delta weight
 [1,]       + 0.8759 10 -2352.676 4726.0   1.68  0.302
```

```
 [2,]       + 0.8750  8 -2353.956 4724.4   0.00  0.698
 [3,]         0.8689  8 -2361.803 4740.1  15.69  0.000
 [4,]         0.8669  6 -2364.387 4741.0  16.67  0.000
 [5,]       + 0.8546  7 -2379.110 4772.6  48.21  0.000
 [6,]         0.8468  5 -2387.797 4785.8  61.42  0.000
 [7,]         0.8058  4 -2427.242 4862.6 138.25  0.000
 [8,]         0.7938  7 -2437.276 4888.9 164.54  0.000
 [9,]         0.7870  5 -2442.633 4895.5 171.09  0.000
[10,]         0.7621  3 -2461.073 4928.2 203.86  0.000
[11,]         0.6745  4 -2513.273 5034.7 310.31  0.000
[12,]         0.1806  3 -2666.979 5340.0 615.67  0.000
[13,]         0.0000  2 -2700.146 5404.3 679.97  0.000
Models ranked by AICc(x)
```

Selon le R^2-ajusté, le meilleur modèle serait celui contenant les deux interactions. Mais l'AIC, plus conservateur, aurait privilégié celui sans l'interaction entre la longueur des ailes et l'espèce. Ce dernier modèle serait, selon les poids d'Akaike, 2x plus probable que celui contenant les deux interactions (0.698/0.302=2.31).

29.10. Contenu optionnel : le modèle linéaire général

Nous avons vu dans ce chapitre que l'ANOVA et l'ANCOVA, peuvent être remplacées par une régression multiple dans laquelle on utilise le dummy coding et/ou les interactions. Ce qui est fascinant, c'est qu'en fait, l'ensemble des tests statistiques que nous avons vu ensemble jusqu'à maintenant (i.e. les chapitres Chapitre 12 à Chapitre 21) peuvent TOUS être remplacés par une régression multiple avec des variables qualitatives et/ou des interactions. C'est pourquoi on appelle cette approche le **modèle linéaire général**.

Par exemple, le test de T à deux échantillons peut facilement être remplacé par une régression linéaire, pour autant que l'on tourne les données au format long plutôt que large.

En réfléchissant en test de T, on aurait par exemple organisé nos données comme ceci :

Groupe A	Groupe B
12,5	13,2
11,8	14,5
...	...

On aurait pu ajuster dans R un tel modèle avec quelque chose comme :

```
t.test(GroupeA, GroupeB, var.equal = TRUE)
```

Mais on pourrait tout aussi bien organiser nos données comme ceci :

Groupe	Valeurs
A	12,5
A	11,8
B	13,2
B	14,5
...	...

Et coder notre modèle dans R comme cela :

```
lm(Valeurs ~ Groupe)
```

On obtiendrait intégralement la même valeur de p.

C'est fou hein?

Je me suis longtemps interrogé, à savoir si ça valait la peine d'enseigner toute cette panoplie de tests, quand au fond, les étudiants n'auraient besoin que du modèle linéaire général pour travailler.

Ma conclusion pour le moment est que, comme ces tests sont utilisés depuis plus d'une centaine d'années pour certains, il est important de les comprendre et de savoir les utiliser quand même. Ils sont aussi conceptuellement beaucoup plus faciles d'approche que le modèle linéaire général et font, en ce sens, une meilleure introduction aux biostatistiques.

Par contre, si le modèle linéaire général vous intrigue ou vous intéresse, je vous encourage fortement à consulter cet article, où l'auteur a décortiqué pour vous comment traduire des dizaines de tests statistiques connus au modèle linéaire général :

https://lindeloev.github.io/tests-as-linear/

Vous remarquerez que dans certains cas, par exemple pour le test de khi-carré, on doit pousser un peu plus loin et utiliser un GLM à cause de l'assomption d'une distribution Poisson (plutôt que normale) des erreurs, mais l'idée générale demeure la même.

29.11. Exercice : Modéliser une variable qualitative et une interaction

Pour cet exercice, nous allons nous éloigner un peu de l'écologie est traiter d'un problème plus près de la santé publique : est-ce que le taux de cholestérol varie entre les états américains. Comme nous savons que

le cholestérol a tendance à augmenter avec l'âge, nous aurons une sous-question à explorer, soit : est-ce que l'augmentation du cholestérol avec l'âge est la même entre les états américains.

Le fichier de données à analyser se trouve ici[2].

Remarquez que le fichier est au format .data. Il doit donc être ouvert avec la fonction `read_table2`, plutôt qu'avec `read_csv` ou `read_excel` comme c'est souvent notre habitude.

Je vous demande donc 6 choses :

1. Chargez et vérifiez et préparez ces données
2. Visualisez dans un graphique la différence de pente âge-cholestérol dans les deux états américains
3. Effectuez une modélisation permettant de répondre à notre question principale et à notre sous-question
4. Validez ce modèle
5. Choisissez une technique de sélection du meilleur modèle et appliquez la
6. Interprétez les résultats pour répondre à nos deux questions

29.12. En résumé

- L'ANOVA et la régression linéaire sont une seule et même procédure statistique (eh oui!).
- Les variables qualitatives peuvent être transformées en plusieurs variables contenant des valeurs 0 ou 1 pour les entrer dans une régression.
- Il est important de distinguer les effets aléatoires des effets fixes.

[2] https://drive.google.com/file/d/1LSo22QD_nL5XydoCIdSPOI4n-qvNEeJR/view?usp=sharing

- L'interaction (au sens statistique) peut être définie comme la non-indépendance des effets de deux variables.

30. Les modèles mixtes

30.1. Problématique

Nous avons souvent discuté qu'une assomption importante de l'ensemble des modèles statistiques vus jusqu'à maintenant était que les données avaient été récoltées de façon indépendante. Étonnamment, dans la vraie vie d'une biologiste, nos observations sont rarement parfaitement indépendantes. On mesure plusieurs arbres dans une même parcelle, on prend plusieurs profils de températures dans le même lac, etc.

Nous avons plusieurs fois discuté du fait que si l'on entre des observations non-indépendantes dans un modèle, cela viendra gonfler notre certitude sur nos inférences, fausser nos valeurs de p et possiblement biaiser nos résultats. Nous risquons plus souvent de commettre des erreurs de type I, puisque le n effectif (le nombre d'observations réellement indépendantes) de notre analyse est dans les faits plus petit que celui utilisé pour faire les calculs.

Pendant longtemps, la seule solution possible à cette problématique a été d'agréger nos données. Plutôt que d'utiliser 4 observations pour un même lac, on calculait la moyenne des 4 observations, et on utilisait cette valeur moyenne pour le lac entier. Cela règle le problème d'indépendance des observations, mais on gaspille aussi beaucoup de nos données, sans compter le fait qu'on perd de l'information sur la variabilité à l'intérieur de chaque lac.

Heureusement, depuis une vingtaine d'années, les modèles linéaires mixtes permettent de gérer ce problème de façon élégante et efficace. C'est de cette technique dont il sera question dans ce chapitre.

30.2. L'équation complète de la régression linéaire

Avant de s'attaquer aux modèles mixtes, nous devrons d'abord compléter l'équation que nous avons utilisée pour décrire la régression linéaire et la régression multiple. Jusqu'à maintenant, nous avions décrit la régression linéaire comme ceci :

$$y = b_0 + b_1 x_1 ... + b_p x_p$$

Hors, cette façon de décrire est une version simplifiée, que nous devrons compléter avant de pouvoir passer aux modèles mixtes.

La première chose que nous devrons faire est de clarifier que cette équation s'applique pour chacune de nos observations. L'ajout de l'indice i aux termes y, x_1 et x_p nous permettra d'accomplir cela :

$$y_i = b_0 + b_1 x_{1,i} + ... + b_p x_{p,i}$$

Il est maintenant plus clair que cette équation s'applique ligne par ligne (i=1 ... i=n) dans notre tableau de données.

Ensuite, si on prend quelques secondes pour réfléchir à cette formulation, on réalise que l'égalité n'est pas complète. Le modèle à droite n'est pas exactement égal au y à gauche. Notre modèle devrait aussi comprendre un résidu, que l'on peut écrire comme ceci :

$$y_i = b_0 + b_1 x_{1,i} + ... + b_p x_{p,i} + \epsilon_i$$

Autrement dit, la valeur observée de y est égale à l'ordonnée à l'origine, plus une pente partielle pour chaque variable, plus un terme d'erreur associé à chaque observation (le résidu).

Enfin, pour terminer, on pourrait ajouter une seconde équation à cette définition, soit la notation mathématique indiquant que les résidus sont distribués normalement, avec une moyenne de zéro et un écart-type de sigma, comme ceci :

$$\epsilon_i \sim N(0, \sigma_\epsilon^2)$$

30.3. Ordonnées à l'origine aléatoires

Pour bien comprendre les effets aléatoires, nous allons discuter d'un exemple où l'on aurait voulu mesurer, comme pour la régression linéaire, la relation entre la surface d'un parc et la richesse en espèces d'oiseaux qu'on y retrouve. L'équation de cette relation pourrait être écrite comme ceci :

$$Richesse_i = b_0 + b_1 Surface_i + \epsilon_i$$

Or, après avoir planifié l'expérience, on réalise que les parcs sont organisés par villes (4 parcs dans la ville A, 2 parcs dans la ville B, etc.). Comme on sait que le mode de gestion des parcs par chacune des villes peut avoir un impact sur le nombre d'espèces d'oiseaux, on réalise rapidement que nos données ne sont pas entièrement indépendantes. Elles sont organisées de façon hiérarchique.

Une première chose que l'on pourrait vouloir est de permettre à l'ordonnée à l'origine de notre modèle de varier en fonction de la ville où le parc de situait. Comme cela, si une ville a tendance à avoir plus

d'oiseaux qu'une autre, notre modèle pourra en tenir compte. On écrirait donc ce nouveau modèle comme ceci :

$$Richesse_{ij} = b_0 + b_1 Surface_{ij} + a_i Ville_i + \epsilon_{ij}$$

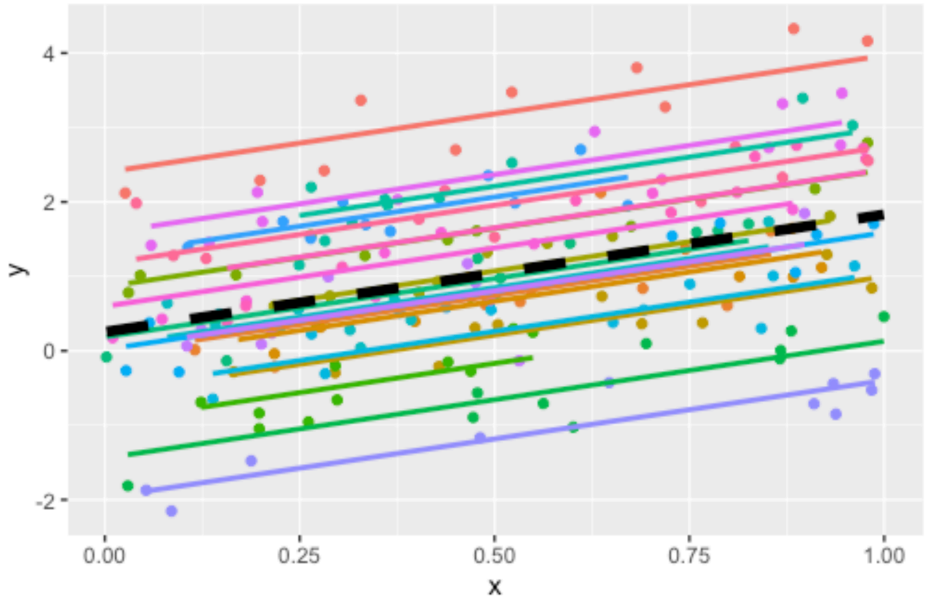
Le problème avec un tel modèle est que l'on ajuste autant de paramètres que le nombre de villes - 1, même si au fond, cet effet de la ville ne nous intéresse pas vraiment. On veut que notre modèle en tienne compte dans son calcul, mais on ne s'intéresse pas à savoir si la ville A est meilleure que la ville B, etc.

Le modèle avec ordonnée à l'origine aléatoire nous permet d'éviter d'ajuster un paramètre pour chacune des villes, en ajoutant la contrainte que les ordonnées à l'origine de chaque ville suivent une distribution normale définie comme ceci :

$$a_i \sim N(0, \sigma_a^2)$$

Notre modèle contient donc maintenant deux termes d'erreur normalement distribués, un pour les résidus et un pour les différences entre les villes.

Visuellement, notre modèle ajusterait quelque chose de semblable à ceci :



Où la ligne noire pointillée représente les paramètres de pente et d'ordonnée à l'origine, et chacune des lignes de couleur représente les ordonnées à l'origine individuelles de chacune des villes, qui suivent une distribution normale, entre elles. Beaucoup de pentes sont très près de la pente globale, et peu de pentes en sont fortement éloignées.

À ce point-ci, vous vous demandez peut-être quel est le gain en termes d'ajustement si notre modèle a tout de même à estimer une ordonnée à l'origine par groupe. La clé est qu'il n'a pas à le faire. Plutôt que de chercher la meilleure valeur pour chacune des ordonnées à l'origine, il cherche seulement à estimer la meilleure valeur pour leur variance (σ_a^2). Les valeurs des ordonnées à l'origine apparaissent dans le modèle, mais le modèle ne travaille pas spécifiquement pour les trouver.

30.4. Corrélations intra-classe

Puisque la prémisse des modèles mixtes est que les observations à l'intérieur d'un groupe se ressemblent entre elles, il peut être intéressant de savoir jusqu'à quel point elles le sont. De pouvoir mettre un chiffre sur cette ressemblance. C'est pourquoi les statisticiens ont inventé le coefficient de corrélation intra-classe (ICC; intraclass correlation coefficient). Dans un modèle simple, à un seul effet aléatoire comme celui ci-haut, ce coefficient peut être calculé comme ceci :

$$ICC = \frac{\sigma_a^2}{\sigma_a^2 + \sigma_\epsilon^2}$$

Autrement dit, l'ICC est la variance de l'effet aléatoire, divisée par la somme des variances de l'effet aléatoire et des résidus. Ce coefficient sera toujours entre 0 et 1.

Un ICC de 0 nous dirait que les observations à l'intérieur d'un même groupe sont aussi différentes que si elles provenaient de groupes différents. Un ICC de 1 nous dirait que les observations à l'intérieur d'un même groupe sont parfaitement identiques.

L'ICC peut s'interpréter comme un R^2, mais qui nous indique la proportion de variance expliquée par l'effet aléatoire de notre modèle.

Dans des modèles avec des structures d'effets aléatoires plus complexes comprenant par exemple plusieurs niveaux, la définition de l'ICC est loin d'être aussi simple et son calcul dépasse ce qu'il est réaliste de voir dans ce cours ...

30.5. Pentes aléatoires

Une fois l'ordonnée à l'origine aléatoire ajoutée au modèle, on pourrait se poser la question : est-ce uniquement l'ordonnée à l'origine qui varie entre les villes, ou la pente (i.e. la nature de la relation) pourrait aussi varier?

Dans la plupart des cas, la question sera légitime. Ici, par exemple, si les pratiques d'une ville peuvent ajouter ou soustraire des espèces d'oiseaux dans chacun des parcs, leurs pratiques peuvent probablement aussi influencer l'effet d'ajouter un hectare supplémentaire à un parc existant.

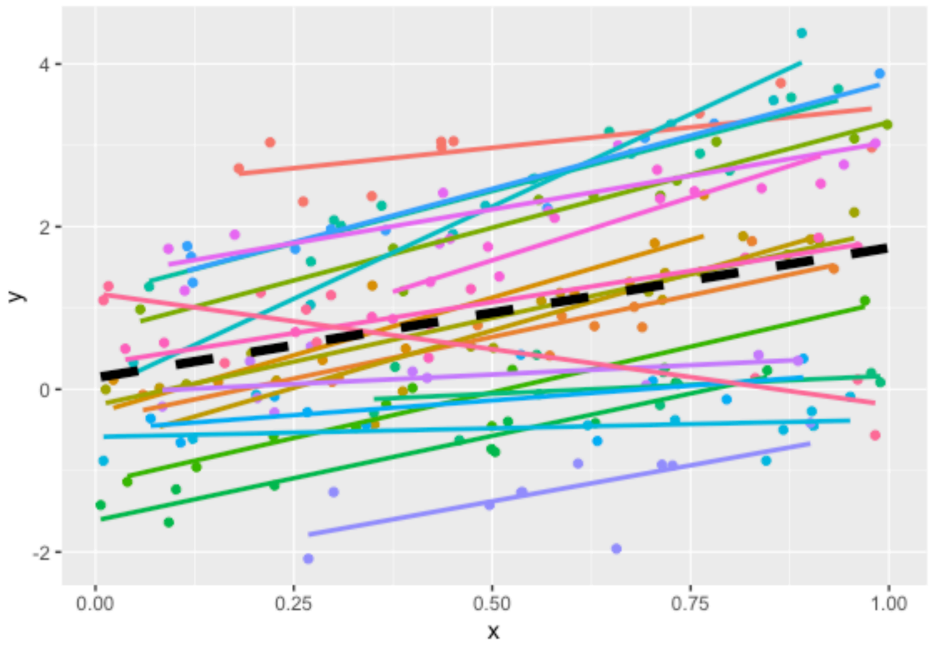
Pour tenir compte de cette possibilité, il convient d'ajouter un terme d'interaction (c_i) entre la ville et la surface dans notre modèle, comme ceci :

$$Richesse_{ij} = b_0 + b_1 Surface_{ij} + a_i Ville_i + c_i Surface_{ij} Ville_i + \epsilon_{ij}$$

Mais, tout comme la valeur d'ordonnée à l'origine associée à chaque ville ne nous intéresse pas directement, la valeur associée à la différence de pente pour chaque ville ne nous intéresse pas non plus. On peut donc placer cette interaction en effet aléatoire, en spécifiant que les termes d'interaction seront aussi distribués de façon aléatoire, suivant une distribution normale, comme ceci :

$$c_i \sim N(0, \sigma_c^2)$$

Visuellement, notre modèle ressemble maintenant à la figure ci-dessous. Il contient une pente et une ordonnée à l'origine globale, mais aussi, pour chacun des groupes, une pente et une ordonnée à l'origine, dont les valeurs suivent chacune une distribution normale.



30.6. Nuances sur les techniques d'ajustement

Contrairement à la régression linéaire multiple, un modèle linéaire mixte ne peut pas être ajusté à partir d'une opération d'algèbre matriciel. Il doit passer par un processus itératif qui se rapproche progressivement de la solution qui maximise la vraisemblance du modèle. Encore une fois, les détails ici ne sont pas si importants, mais la vraisemblance d'un modèle est la probabilité qu'on ait observé les données pour des valeurs de paramètres en particulier. Le processus d'ajustement essaie des valeurs pour chacun de nos paramètres, jusqu'à arriver à la solution la plus vraisemblable, i.e. la plus probable étant donné les données observées.

Il n'existe malheureusement pas de solution parfaite pour effectuer

ce processus. Les deux techniques existantes sont la **vraisemblance maximale** (*Maximum Likelihood*; ML) et la **vraisemblance maximale restreinte** (*Restricted Maximum Likelihood*; REML). Vos collègues, directrices, etc. utiliserons probablement les abréviations anglaises de ces termes (i.e. ML et REML) et nous ferons donc de même ici.

La différence importante à savoir entre les deux techniques est la suivante :

> **!** Important
>
> Les composantes de variance estimées par la technique ML seront biaisées, mais les estimés de paramètres seront fiables. Au contraire, la REML fournira des composantes de variances exactes, mais, parce qu'elle utilise une transformation des données pour y arriver, elle fournira des estimés de paramètre biaisés.

30.7. Processus de sélection de modèle

Donc, une fois que l'on sait tout cela, le processus typique de construction et sélection d'un modèle mixte se déroule habituellement comme ceci :

1. Construire le modèle complet, probablement surajusté, incluant toutes les variables et les effets aléatoires d'intérêt
2. Si désiré, épurer les termes aléatoires du modèle, en se basant sur l'ajustement REML.
3. Épurer les termes fixes du modèle en se basant sur l'ajustement ML.
4. Présenter le modèle final, ajusté avec la technique REML.

J'ai ajouté la mention "si désiré" à l'étape 2, car à mon avis et de l'avis de plusieurs de mes collègues, cette étape est rarement pertinente. Si

vos données ne sont pas indépendantes de par la structure de votre expérience (i.e. réplicats sur un même site, sur un même individu, etc.), il est justifié et pertinent de laisser l'effet aléatoire dans votre modèle, que ce dernier soit considéré important ou non en termes de variance expliquée.

Par souci d'économie d'espace, toutes nos comparaisons de modèles seront effectuées avec l'AIC, mais il existe aussi des façons de comparer les modèles mixtes qui fournissent des valeurs de p, par exemple les tests de ratios de vraisemblance (*likelihood ratio test*).

30.8. Validation du modèle

Un modèle mixte doit être validé selon les mêmes principes qu'un modèle à effet fixe (i.e. sans effets aléatoires). C'est-à-dire que l'on doit s'assurer que (1) la distribution des erreurs suit une loi normale et (2) que les résidus sont homogènes à travers le gradient de prédictions.

Comme notre modèle mixte contient, par définition, des observations corrélées entre-elles, les résidus bruts sont difficilement utilisables pour vérifier la normalité des erreurs. C'est pourquoi la plupart des auteurs recommandent d'utiliser une transformation, basée sur une décomposition de la matrice de variance-covariance. Le détail de cette transformation n'est absolument pas important ici. Ce qu'il faut retenir, c'est qu'après cette transformation, les résidus devraient maintenant suivre une distribution normale (si notre modèle est adéquat).

Une fois que l'on sait tout cela, on doit donc s'assurer que :

- Les résidus transformés forment effectivement une distribution normale
- Les résidus non-transformés ne présentent pas de patrons à travers le gradient de prédictions.

Une fois que vous savez tout cela, sachez par contre que récemment, des chercheurs ont montré par simulation que l'on peut torturer de beaucoup un modèle linéaire mixte, et qu'il continuera de donner des résultats fiables malgré les problèmes de non-normalité, hétéroscédasticité, etc.

30.9. Labo : Le poids des manchots, avec effets aléatoires

Depuis le début de nos travaux avec le tableau de données des manchots de Palmer, nous avons travaillé comme si les individus étaient entièrement indépendants les uns des autres. Cependant, il est clair dans les méthodes de cueillette de données que ce n'est pas exactement le cas. Les données ont été recueillies sur 3 îles. On pourrait donc s'attendre à ce que les individus provenant d'une même île se ressemblent plus entre eux, puisqu'ils vivent dans les mêmes conditions.

Les différences inter-îles sont l'exemple parfait d'une variable pouvant être traitée en effet aléatoire. Dans le cas de nos projets, les valeurs associées à chaque île en particulier ne nous intéressent pas vraiment. Si on refaisait l'expérience, elle pourrait tout à fait être faite sur une autre série d'îles. On veut seulement tenir compte correctement de la non-indépendance des données dans nos analyses.

Remarquez, encore une fois, que ce choix dépend du point du vue de l'analyste. Si nous étions un gestionnaire de la faune qui s'occupe de ces 3 espèces, les particularités de chacune des îles pourraient tout à fait nous intéresser. Il faudrait alors mettre l'identité de l'île en effet fixe dans nos modèles.

Nous allons donc ré-analyser notre modèle prédisant le poids des manchots en fonction de l'espèce, du sexe, de la longueur des ailes et de

l'interaction entre le sexe et l'espèce, mais en tenant compte cette fois de la non-indépendance de nos observations.

Tout d'abord, nous allons préparer nos données et charger les librairies nécessaires. Nous utiliserons pour nos exemples la librairie de modèles mixtes `nlme`, puisqu'elle possède la syntaxe la plus facile à comprendre, mais sachez qu'il en existe de nombreuses autres. Entre autres, la librairie `lme4` est un autre classique et `glmmTMB` gagne de plus en plus en popularité.

```
library(palmerpenguins)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(nlme)
```

```
Attaching package: 'nlme'

The following object is masked from 'package:dplyr':

    collapse
```

```
library(MuMIn)

pour_modele_mixte <-
  penguins |>
  drop_na(species, sex, body_mass_g, flipper_length_mm,
   ↪  island) |>
  mutate(
    species = relevel(species,"Gentoo"),
    sex = relevel(sex,"male")
  )
```

Étape 1 : Modèle surajusté

Donc, selon le protocole établi plus haut, notre première étape consistera à préparer un modèle complet, surajusté, contenant tous les effets fixes possible, afin de tester notre structure aléatoire. Comme expliqué plus haut, cette étape de sélection statistique de la présence ou non d'effets aléatoires est rarement justifiée. Néanmoins, elle nous permettra ici de voir comment inscrire dans R les différents types d'effets aléatoires.

Afin d'avoir un point de comparaison, nous allons d'abord nous créer un modèle sans effet aléatoire. Par contre, pour que nos modèles soient comparables avec la méthode REML, nous utiliserons la fonction `gls` pour ajuster ce premier modèle plutôt que la fonction `lm` :

```
modele_simple <- gls(
  body_mass_g ~
    flipper_length_mm + sex + species +
    sex:species +
    species:flipper_length_mm,
  data = pour_modele_mixte,
  na.action = "na.fail",
  method = "REML"
)
```

Pour votre information, la fonction `gls` permet de spécifier des structures de variance, permettant de corriger les problèmes comme les effets d'entonnoirs ou de variances inégales dans nos résidus. Cependant, si on ne spécifie pas de structure de variance, comme ici, le modèle ajuste une régression normale, mais par la méthode REML plutôt que par les moindres carrés.

Étape 2 : Déterminer la structure aléatoire optimale

Nous allons ensuite ajuster un modèle avec une ordonnée à l'origine aléatoire sur la variable d'île. Nous utiliserons pour cela la fonction `lme` de la librairie nlme :

```
ordonnee_aleatoire <- lme(
  body_mass_g ~
    flipper_length_mm + sex + species +
    sex:species +
    species:flipper_length_mm,
  random = ~1 | island,
  data = pour_modele_mixte,
  na.action = "na.fail",
  method = "REML"
)
```

Remarquez que la syntaxe est identique à une régression multiple normale, mais on ajoute dans l'argument random une deuxième formule, contenant uniquement la partie de droite (à partir du ~), qui décrit la structure des effets aléatoires dans notre modèle.

Remarquez la syntaxe étrange avec le `1 | island`. Le `1` dans les formules de R symbolise l'ordonnée à l'origine. Nous n'avons jamais eu à le spécifier parce que R l'ajoute toujours pour nous lorsque l'on entre une

formule. Par exemple, pour notre régression, on aurait pu tout aussi bien écrire

```
body_mass_g ~ 1 + flipper_length_mm
```

Si on avait voulu omettre l'ordonnée à l'origine dans un modèle, il aurait fallu le spécifier explicitement, en mettant un `0` plutôt qu'un `1`, par exemple :

```
body_mass_g ~ 0 + flipper_length_mm
```

Dans la syntaxe des effets aléatoire, il faut lire la barre verticale (`|`) comme étant "par". Par exemple, on demande ici une ordonnée à l'origine par valeur de la variable island. Implicitement, `lme` comprend que ces ordonnées à l'origine seront distribuées selon une loi normale (i.e. elles deviendront notre effet aléatoire).

Enfin, on aurait aussi pu vouloir tester la pertinence d'ajuster une pente différente par île plutôt qu'uniquement une ordonnée à l'origine. Autrement dit : ajuster un modèle dans lequel on a aussi une pente aléatoire. Dans R, la syntaxe pour ajuster un tel modèle aurait été la suivante :

```
pente_aleatoire <- lme(
  body_mass_g ~
    flipper_length_mm + sex + species +
    sex:species +
    species:flipper_length_mm,
  random = ~1 + flipper_length_mm |  island,
  data = pour_modele_mixte,
  na.action = "na.fail",
  method = "REML"
)
```

```
Error in lme.formula(body_mass_g ~ flipper_length_mm +
sex + species + : nlminb problem, convergence error code
= 1
  message = singular convergence (7)
```

On ajuste une ordonnée à l'origine (`1`) et une pente pour la variable de longueur des ailes par niveau de la variable d'île. Autrement dit, non seulement on pense que le poids moyen varie entre les îles, mais que l'effet de la longueur des ailes serait aussi différent entre les îles.

Malheureusement, notre jeu de données et sa structure de variance ne nous permettent pas d'évaluer ce modèle correctement.

Évidemment, on aurait pu compliquer les choses encore plus en se demandant si l'île change l'effet du sexe, de l'espèce, etc. Mais on en a déjà bien assez ici pour comprendre le principe. L'idée générale étant que notre modèle doit refléter la structure hiérarchique de nos données.

Donc, une fois tous ces modèles ajustés, avec le méthode REML, on peut les comparer afin de déterminer la meilleure structure aléatoire. Pour ce faire, une façon simple est d'utiliser la fonction `model.sel` de la librairie `MuMIn`, qui nous calculera l'AIC de chacun des modèles :

```
model.sel(modele_simple, ordonnee_aleatoire)
```

```
Model selection table
                   (Int) flp_lng_mm sex spc
modele_simple      278.8       23.5   +   +
ordonnee_aleatoire 278.8       23.5   +   +
                   flp_lng_mm:spc sex:spc class random
modele_simple                      +       +   gls
ordonnee_aleatoire                 +       +   lme      i
                   df   logLik   AICc delta weight
modele_simple      10 -2317.56 4655.8  0.00  0.745
ordonnee_aleatoire 11 -2317.56 4657.9  2.14  0.255
```

```
Models ranked by AICc(x)
Random terms:
 i: 1 | island
```

Ici, le meilleur modèle basé sur l'AIC est celui ne contenant pas une ordonnée à l'origine aléatoire pour chaque île. Ce modèle est 3x plus probablement que celui avec une ordonnée aléatoire (0.745 / 0.255 = 2.92).

Donc, si on avait voulu, il aurait été légime de continuer notre modélisation sans l'effet aléatoire. Cependant, je privilégie toujours de le laisser en place, puisque ce terme rend explicite la structure hiérarchique de nos données.

Étape 3 : Sélection de modèle pour la partie fixe

Une fois la structure aléatoire choisie, il faudrait maintenant effectuer la sélection de modèle sur la partie fixe du modèle. La procédure est exactememnt la même que celle suggéré au Chapitre 28, mais il faut penser d'ajuster les modèles avec la méthode "ML" si on ne veut pas biaiser nos résultats.

`MuMIn` vous avertira d'ailleurs si vous tentez de comparez les termes fixes d'un modèle avec la méthode REML.

```
avec_ML <- lme(
  body_mass_g ~
    flipper_length_mm + sex + species +
    sex:species +
    species:flipper_length_mm,
  random = ~1 | island,
  data = pour_modele_mixte,
  na.action = "na.fail",
  method = "ML"
)

dredge(avec_ML)
```

```
Fixed term is "(Intercept)"

Global model call: lme.formula(fixed = body_mass_g ~
flipper_length_mm + sex + species +
    sex:species + species:flipper_length_mm, data =
    pour_modele_mixte,
    random = ~1 | island, method = "ML", na.action =
    "na.fail")
---
Model selection table
     (Int) flp_lng_mm sex spc flp_lng_mm:spc sex:spc
24   946.3      20.49   +   +                      +
32   278.8      23.50   +   +              +       +
16  -604.9      27.40   +   +              +
8   1001.0      20.02   +   +
23  5485.0              +   +                      +
7   5418.0              +   +
4  -3623.0      39.74   +
14 -6674.0      54.17       +              +
6  -3729.0      40.61       +
2  -4981.0      45.60
5   5092.0                  +
3   4388.0              +
1   4052.0
   df    logLik   AICc  delta weight
24  9 -2353.956 4726.5   0.00  0.701
32 11 -2352.676 4728.2   1.70  0.299
16  9 -2361.803 4742.2  15.69  0.000
8   7 -2364.387 4743.1  16.65  0.000
23  8 -2379.110 4774.7  48.19  0.000
7   6 -2387.797 4787.9  61.38  0.000
4   5 -2407.385 4825.0  98.48  0.000
```

```
14   8 -2437.276 4891.0 164.53  0.000
6    6 -2442.633 4897.5 171.05  0.000
2    4 -2454.246 4916.6 190.14  0.000
5    5 -2513.273 5036.7 310.26  0.000
3    4 -2569.644 5147.4 420.94  0.000
1    3 -2625.644 5257.4 530.89  0.000
Models ranked by AICc(x)
Random terms (all models):
  1 | island
```

Nos résultats n'ont pas vraiment bougé en ajoutant l'effet aléatoire, ce qui était prévisible étant donné sa faible importance.

30.10. Labo : Validation du modèle avec effets aléatoires et interprétation

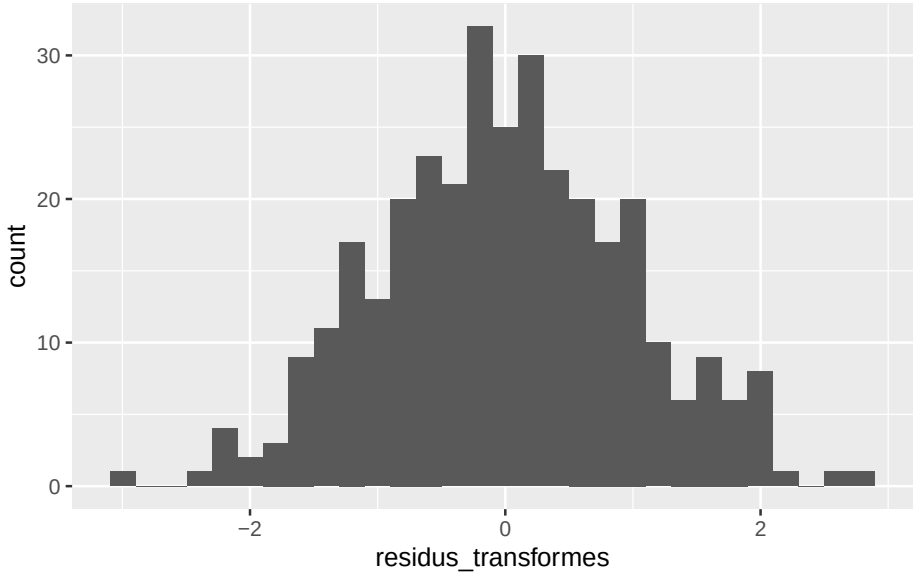
Comme discuté précédemment, nous devons maintenant extraire deux résidus différents de notre modèle, soit les résidus bruts, et les résidus transformés :

```
pour_modele_mixte <- pour_modele_mixte |>
  mutate(
    residus = residuals(ordonnee_aleatoire),
    residus_transformes = residuals(ordonnee_aleatoire,
    ↪  type = "normalized"),
    predictions = predict(ordonnee_aleatoire)
  )
```

Il faut ensuite valider la normalité des erreurs avec les résidus transformés :

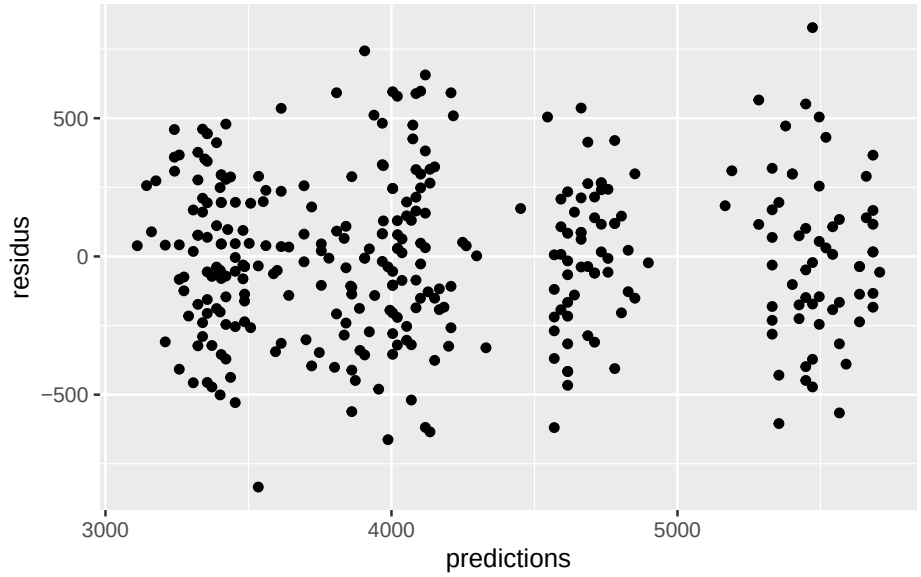
```
pour_modele_mixte |>
  ggplot(aes(x = residus_transformes)) +
   ↪  geom_histogram()
```

```
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```



Et vérifier l'homogénéité à l'aide des résidus bruts, comme on le fait habituellement :

```
pour_modele_mixte |>
  ggplot(aes(x = predictions, y = residus)) +
   ↪  geom_point()
```



Dans les deux cas, tout est fantastique.

Donc, au final, voilà les résultats de notre modèle :

```
summary(ordonnee_aleatoire)
```

```
Linear mixed-effects model fit by REML
  Data: pour_modele_mixte
       AIC      BIC   logLik
  4657.119 4698.707 -2317.56

Random effects:
 Formula: ~1 | island
        (Intercept) Residual
StdDev:  0.02246539 287.1067
```

```
Fixed effects:  body_mass_g ~ flipper_length_mm + sex +
species + sex:species +      species:flipper_length_mm
                                      Value
(Intercept)                         278.8098
flipper_length_mm                    23.4992
sexfemale                          -597.5011
speciesAdelie                       625.6201
speciesChinstrap                  -1672.3469
sexfemale:speciesAdelie              -1.8422
sexfemale:speciesChinstrap          403.8381
flipper_length_mm:speciesAdelie      -7.1848
flipper_length_mm:speciesChinstrap    3.1751
                                  Std.Error  DF
(Intercept)                      1203.2094 322
flipper_length_mm                   5.4286 322
sexfemale                          71.2203 322
speciesAdelie                    1419.5313 322
speciesChinstrap                 1703.0876 322
sexfemale:speciesAdelie            87.5023 322
sexfemale:speciesChinstrap        111.1189 322
flipper_length_mm:speciesAdelie     6.6906 322
flipper_length_mm:speciesChinstrap  8.1093 322
                                   t-value p-value
(Intercept)                       0.231722  0.8169
flipper_length_mm                 4.328804  0.0000
sexfemale                        -8.389472  0.0000
speciesAdelie                     0.440723  0.6597
speciesChinstrap                 -0.981950  0.3269
sexfemale:speciesAdelie          -0.021053  0.9832
sexfemale:speciesChinstrap        3.634287  0.0003
flipper_length_mm:speciesAdelie  -1.073870  0.2837
flipper_length_mm:speciesChinstrap  0.391545  0.6957
 Correlation:
                                  (Intr) flpp__
```

```
flipper_length_mm                  -1.000
sexfemale                          -0.689  0.673
speciesAdelie                      -0.848  0.847
speciesChinstrap                   -0.706  0.706
sexfemale:speciesAdelie             0.561 -0.548
sexfemale:speciesChinstrap          0.441 -0.432
flipper_length_mm:speciesAdelie     0.811 -0.811
flipper_length_mm:speciesChinstrap  0.669 -0.669
                                   sexfml spcsAd
flipper_length_mm
sexfemale
speciesAdelie                       0.584
speciesChinstrap                    0.487  0.599
sexfemale:speciesAdelie            -0.814 -0.594
sexfemale:speciesChinstrap         -0.641 -0.374
flipper_length_mm:speciesAdelie    -0.546 -0.997
flipper_length_mm:speciesChinstrap -0.451 -0.567
                                   spcsCh sxfm:A
flipper_length_mm
sexfemale
speciesAdelie
speciesChinstrap
sexfemale:speciesAdelie            -0.396
sexfemale:speciesChinstrap         -0.638  0.522
flipper_length_mm:speciesAdelie    -0.573  0.565
flipper_length_mm:speciesChinstrap -0.998  0.367
                                   sxfm:C fl__:A
flipper_length_mm
sexfemale
speciesAdelie
speciesChinstrap
sexfemale:speciesAdelie
sexfemale:speciesChinstrap
flipper_length_mm:speciesAdelie     0.350
```

```
flipper_length_mm:speciesChinstrap  0.618  0.543

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3
-2.90577208 -0.67090338 -0.02185134  0.67982865
        Max
 2.88351508

Number of Observations: 333
Number of Groups: 3
```

Donc, dans l'ordre, le modèle nous présente d'abord certaines statistiques d'ajustement (AIC, BIC , etc.).

Le deuxième bloc d'information nous présente la partie aléatoire de notre modèle. Il nous rappelle la structure de notre effet aléatoire, puis nous informe de l'écart-type associé à chacune des composantes de variance de notre modèle : 0,022 pour l'effet de l'île (Intercept) et 287,11 pour les résidus. Avec ces deux chiffres, on peut calculer manuellement l'ICC soit : $0{,}022^2$ / ($0{,}022^2$ + $287{,}11^2$) = $5{,}87 \times 10^{-9}$. Autrement dit, les manchots d'une même île sont aussi différents qu'entre les îles. Cette valeur d'ICC minuscule confirme aussi le diagnostic de l'AIC, qui nous suggérait que l'île n'avait pas vraiment d'impact.

Le troisième bloc d'information nous fournit les coefficients de la partie fixe du modèle, qui s'interprète exactement comme dans la régression multiple.

Le reste de l'information fournie ne sera pas utilisée dans ce cours.

30.11. Conclusion

Les modèles linéaires mixtes sont un sujet relativement complexe, mais qui vient résoudre un problème plutôt commun en écologie : le manque

d'indépendance entre les observations. Il y a énormément de cas particuliers qui pourraient survenir lorsque vous analyserez vos propres données avec des modèles mixtes, que nous n'avons pas traité ici. Je vous laisse donc en référence la Foire aux Questions (FAQ) préparée par Ben Bolker, l'auteur des librairies de modèles mixtes `lme4` et `glmmTMB` : https://bbolker.github.io/mixedmodels-misc/glmmFAQ.html. Elle vous sera probablement d'une grande utilité dans vos aventures statistiques!

31. Introduction aux GLM : la régression logistique

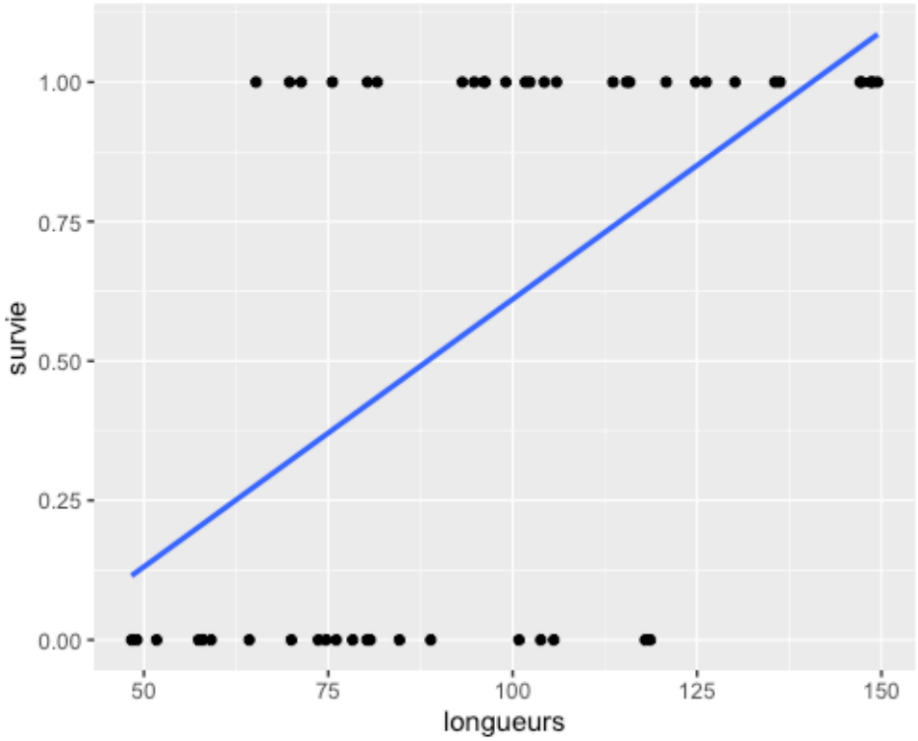
31.1. Contexte

Pour ce chapitre, nous nous attaquerons à un type de données précis. Celui où notre variable en Y (la variable à expliquer) ne peut avoir que deux valeurs : 0 et 1. Ce genre de données binaires peut survenir dans de nombreuses situations en écologie. Vous pourriez par exemple suivre une population de poissons, que vous marquez en début de saison, puis tentez de repêcher en fin de saison pour savoir si ils ont survécu, et ensuite analyser les facteurs favorisant leur survie. Chacune des observations de votre tableau de données sera donc un individu, avec une variable Survie (0 ou 1) que vous voudriez expliquer par une série d'autres facteurs (Age, Sexe, etc.). Ce genre de données surviendra aussi, par exemple, si vous tentez de déterminer quels sont les facteurs qui favorisent la présence d'une espèce. Vous codez alors 0 ou 1 sa présence et notez pour chaque site les facteurs que vous croyez importants.

31.2. Problématique

Au moment d'entrer ces données dans un modèle de régression multiple, on remarquera rapidement une série de problèmes.

Tout d'abord, avec des données binaires, il n'y a aucun moyen de transformer les données pour les forcer dans une distribution normale. C'est tout simplement impossible. Ensuite, si on décide d'outrepasser cette restriction, on se butte rapidement à d'autres problèmes. Si on voulait par exemple modéliser la survie en fonction de la taille d'un individu, la régression à ajuster pourrait visuellement ressembler à ceci :



On a PLUSIEURS problèmes! Tout d'abord, remarquez que notre modèle nous informe que, pour un poisson de 87 cm, on obtient 0,5 survie. Or, notre poisson a survécu ou pas, il ne peut pas avoir survécu à moitié.

Donc, premier constat, pour que notre modèle ait du sens, il devra modéliser non pas la survie directement, mais la probabilité de survie. Dans

ce contexte, cette probabilité est habituellement notée avec le symbole π (pi).

Par contre, cela ne règle pas tout. Pour un poisson de 150 cm, notre modèle prédit environ 1,15. Hors, par définition, une probabilité devrait toujours se situer entre 0 et 1.

31.3. Principe d'un GLM pour la régression logistique.

Les GLM (*Generalized Linear Models*) nous permettent de régler tous ces problèmes de façon extrêmement élégante et fonctionnelle.

Nous avons vu au chapitre précédent sur les modèles mixtes que la définition précise, mathématique, d'un modèle de régression comprend une partie systématique

$$y_i = b_0 + b_1 x_i + \epsilon_i$$

et une partie qui définit la distribution des résidus

$$\epsilon_i \sim N(0, \sigma_\epsilon^2)$$

Un GLM, quant à lui, comprendra trois parties. La partie systématique, la distribution des résidus mais aussi une fonction de lien.

La première chose à faire pour permettre l'analyse de données binaires sera de spécifier la bonne distribution d'erreurs, correspondant à nos données plutôt que de toujours utiliser la distribution normale.

Dans le cas de la régression logistique (le GLM approprié aux données binaires), les erreurs suivront une distribution binomiale avec un n de 1 et probabilité π. Si on se rappelle des premiers chapitres, la distribution

binomiale décrit les situations, comme un lancer de pièce de monnaie, où l'on connaît le nombre d'essais, et où chaque essai peut être un succès ou un échec.

Ainsi, les observations de notre modèle seront des observations 0 ou 1 (présence/absence, mort/survie, etc.), mais nous pourrons jouer avec des probabilités continues dans notre modèle (π).

Ensuite, nous devrons spécifier une fonction pour relier la partie systématique du modèle et la probabilité (π), de façon à contraindre les probabilités entre 0 et 1. Nous devrons choisir ce que l'on nomme dans un GLM la **fonction de lien**.

Dans un GLM pour données binaires, le lien utilisé est habituellement la transformation logit. Mathématiquement, la transformation logistique se définit comme ceci :

$$logit(\pi) = ln\left(\frac{\pi}{(1-\pi)}\right)$$

Nous prendrons maintenant quelques minutes pour comprendre pourquoi cette transformation fonctionne bien.

Le problème de nos probabilité est qu'elles sont coincées entre 0 et 1, alors que le modèle de régression, lui, n'est pas borné. C'est pourquoi la fonction logistique transforme d'abord la probabilité en cote (comme à Vegas) :

$$cote = \frac{\pi}{(1-\pi)}$$

Une probabilité de 0,75 sera traduite en une cote de 3 (à Vegas, on aurait dit 3:1 ou 3 contre 1). Donc, trois fois plus de chances qu'un événement se réalise que de chances qu'il ne se réalise pas.

Cette opération élimine le problème de la borne supérieure, qui peut maintenant dépasser 1 (voir Table 31.1).

Table 31.1.: Comparaison entre les probabilités, les cotes et les log-cotes (transformation logit).

Probabilité	0,9	0,5	0,1
Cote	9	1	0,111
Logit	2,20	0	-2,20

Par contre, les cotes ne peuvent pas être négatives, alors que le modèle de régression, lui, peut facilement nous prédire des valeurs négatives. Les statisticiens ont donc proposé d'utiliser le logarithme naturel de la cote (log-odds) pour contourner le problème. Notre probabilité de 0,75, transformée en cote de 3, deviendrait 1,099 en log-cotes (voir Table 31.1).

Si on met tous ces morceaux ensemble, on peut donc écrire le GLM de la régression logistique comme ceci :

$$\begin{aligned} logit(\pi) &= B_0 + B_1 x_1 \\ Y &\sim B(1, \pi) \end{aligned}$$

Un peu comme pour les modèles mixtes, les valeurs optimales des paramètres d'un GLM seront trouvées par un algorithme itératif, démarrant d'une solution aléatoire et s'approchant progressivement de la meilleure solution en maximisant la vraisemblance (*maximum likelihood*).

31.4. Comment interpréter un modèle de régression logistique

Si on ajuste un modèle de régression logistique à nos données de survie, on obtiendra les paramètres suivants :

$$B_0 = -4.77$$
$$B_1 = 0.055$$

Autrement dit, la survie augmente selon la longueur du poisson ($B_1 > 0$). Par contre, il est relativement complexe de se représenter l'ampleur de ce nombre puisque ce n'est pas directement la probabilité qui augmente de 0,055 par unité de longueur, mais bien le logit de la probabilité.

À cause de la transformation logit, l'effet d'un changement d'une unité de longueur n'aura pas le même effet selon l'intervalle de longueurs que l'on regarde. Pour constater cette différence, le plus simple est de se créer quelques prédictions. On pourrait par exemple prédire la probabilité de survie pour des poissons de 50 vs. 60 cm et 90 vs 100 cm à l'aide notre modèle logit(π) = -4,77+0,055*longueur :

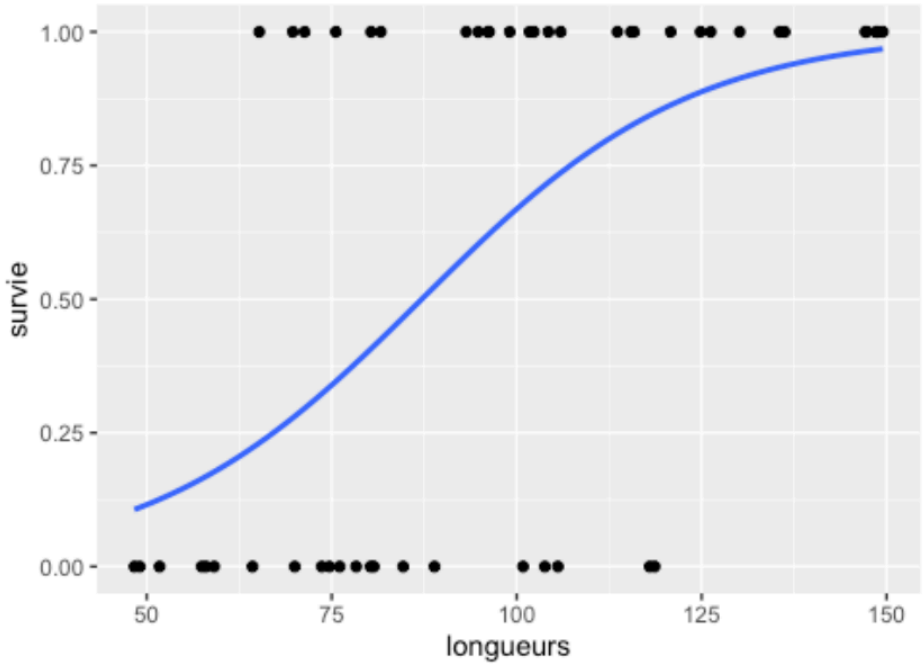
Longueur 1	Longueur 2	Logit 1	Logit 2	Prob. 1	Prob. 2	Diff.
50	60	-2,02	-1,47	0,12	0,18	0,06
90	100	0,18	0,73	0,54	0,67	0,13

Ensuite, on peut passer du logit à la probabilité en inversant la transformation :

$$\pi = \frac{exp(x)}{1 + exp(x)}$$

Donc, autrement dit, une différence de 10 cm aurait un effet deux fois plus grand sur la probabilité de survie si notre poisson mesure 90 cm plutôt que 50 cm.

La façon la plus simple pour se faire une tête sur ce sujet est de visualiser l'ensemble de la courbe de probabilités :



Autrement dit, la probabilité de survie augmente lentement avant 50 cm, elle connaît son maximum d'augmentation autour de 90 cm et ensuite elle ralentit, jusqu'à ne plus vraiment augmenter au-delà de 140 cm.

31.5. Évaluer la qualité des prédictions

La première question qui nous viendra à l'esprit est probablement de savoir : est-ce que notre modèle prédit bien ou non la survie des poissons. Ce qui est complexe pour évaluer cette performance est que notre modèle prédit une probabilité (entre 0 et 1), et que nos données terrain, elles, sont toujours 0 ou 1 (le poisson a survécu ou il n'a pas survécu).

Pour valider la qualité du modèle, une façon simple de contourner ce problème est de choisir un seuil de probabilité, par exemple 0,5 et d'assumer que notre modèle prédit 0 quand $\pi < 0,5$ et qu'il prédit 1 quand $\pi >= 0,5$. On peut ensuite calculer le nombre de cas pour lequel notre modèle a bien prédit et le nombre cas où il a mal prédit, i.e. sa **précision**. Dans notre exemple, le modèle fournirait 38 bonnes prédictions sur 50 individus. Il aurait donc une précision de 76%.

Dans les cas où notre jeu de données est relativement équilibré, calculer la précision du modèle sera suffisant. Par contre, dans les cas où une des classes sera plus rare que l'autre, calculer uniquement la précision ne serait pas très informatif.

Imaginez un scénario où 90% des poissons survivent à l'hiver. Un modèle stupide, qui fait simplement prédire que tout le monde survit aurait une précision de 90%, alors qu'en fait, il ne comprend strictement rien au système.

Il existe une série de mesures permettant de définir les qualités prédictives de notre modèle de façon plus nuancée : sensibilité, spécificité, etc, permettant d'aller très en détail sur les qualités d'un modèle (proportions de faux positifs, faux négatifs, etc.). Nous n'irons cependant pas dans ce genre de détail dans un cours censé introduire aux GLM en général.

Il existe heureusement une façon de mieux évaluer notre modèle avec une statistique nommée le **D de Tjur**, que l'on interprète habituellement

comme un coefficient de discrimination. L'idée générale de cette statistique est d'évaluer à quel point notre modèle sépare bien ou non les cas positifs et négatifs. Verbalement, le calcul va comme suit : on calcule la moyenne des probabilités prédites pour les cas positifs, à laquelle on soustrait la moyenne des probabilités prédites pour les cas négatifs. Dans notre exemple sur les poissons, la moyenne de nos prédictions pour nos poissons ayant vraiment survécu était de 0,711 et la moyenne des prédictions pour ceux n'ayant pas survécu était de 0,399. Le D de Tjur serait donc à ce moment de 0,312.

Dans un cas extrême comme notre modèle stupide qui prédit une survie à tout le monde, les deux moyennes seraient 1, et leur soustraction donnerait donc un D de zéro.

Comme le D de Tjur est borné entre 0 et 1 et qu'il nous renseigne sur la qualité d'ajustement d'un modèle, il est d'usage de l'interpréter comme une mesure de R^2 appropriée aux régressions logistiques.

31.6. La déviance dans un GLM

Dans les modèles linéaires, nous avons souvent utilisé les termes somme des carrés et somme des carrés des résidus (voir Chapitre 27 pour un rappel). Ces termes nous renseignaient respectivement sur la variabilité de Y et la qualité de l'ajustement de notre modèle. Plus le ratio somme des carrés des résidus sur somme des carrés totale est petit, meilleur est notre modèle.

Dans le cas des GLM, comme nos modèles sont ajustés en maximisant la vraisemblance, on s'intéressera plutôt à la déviance, qui se décline en plusieurs versions.

La première, la **déviance nulle** (*null deviance*) décrit la différence de vraisemblance entre un modèle qui ne contient qu'une ordonnée à l'origine et un modèle qui expliquerait parfaitement nos données. Elle servira

donc, un peu comme la somme des carrés, à décrire l'ensemble de ce qui pourrait être potentiellement expliqué par notre modèle.

La deuxième déviance que nous utiliserons est la **déviance des résidus**. Cette dernière mesure la différence de vraisemblance entre le modèle d'intérêt et un modèle parfait qui expliquerait tout.

On pourrait d'ailleurs calculer une mesure de pseudo-R2 nous informant de l'ajustement du modèle en calculant la **déviance expliquée**, soit :

$$\frac{\text{déviance nulle} - \text{déviance résiduelle}}{\text{déviance nulle}}$$

Ce nombre, comme le R^2 pourra varier entre 0 et 1.

31.7. Comment valider un GLM

Comme avec les modèles mixtes, il y a quelques particularités à comprendre à propos des résidus avant de pouvoir bien valider un GLM. La chose la plus importante à comprendre à propos des résidus est que, puisque notre réponse n'est plus linéaire et que la variance n'est plus nécessairement homogène à travers le gradient, un même résidu n'aura pas le même impact dépendamment pour quelle valeur de X on le retrouve.

Il existe toute une série de définitions et de corrections possibles pour contourner le problème (résidus de Pearson, résidus d'Ascombe, etc.), mais les plus utilisés sont les **résidus de déviance**.

Si on se rappelle plus haut, nous avons mentionné que la déviance résiduelle est l'équivalent dans un GLM de la somme des carrés des résidus. Autrement dit, de combien notre modèle dévie par rapport à la réalité. Une des propriétés intéressantes de la déviance est qu'elle peut

être compartimentée. La déviance totale d'un modèle est la somme de la déviance de chacune des observations.

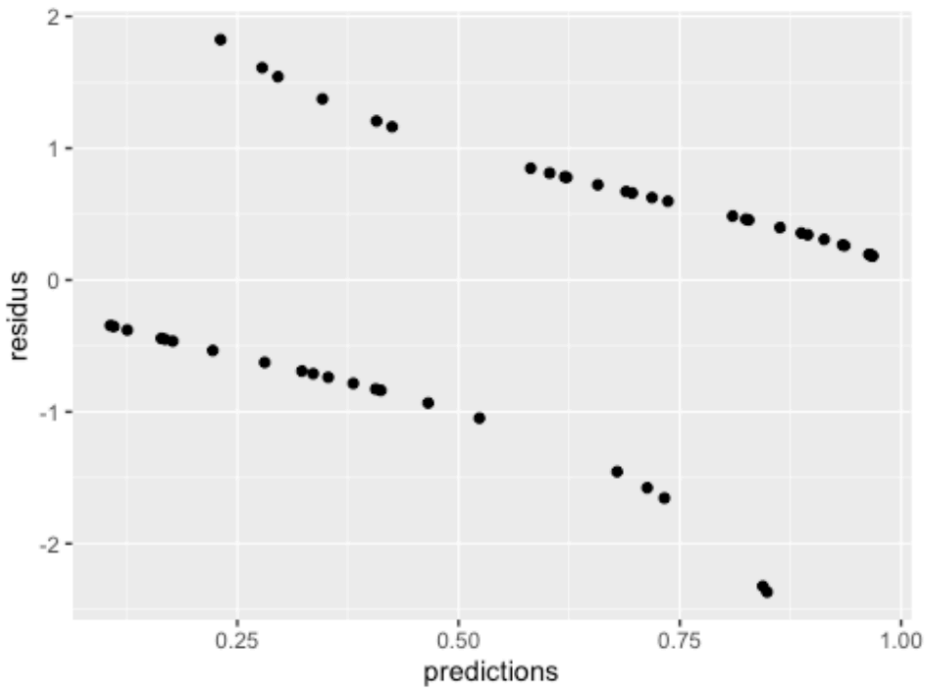
On peut donc définir le résidus de déviance de chaque observation comme :

$$\sigma_i^D = signe(y_i - \mu_i)\sqrt{(d_i)}$$

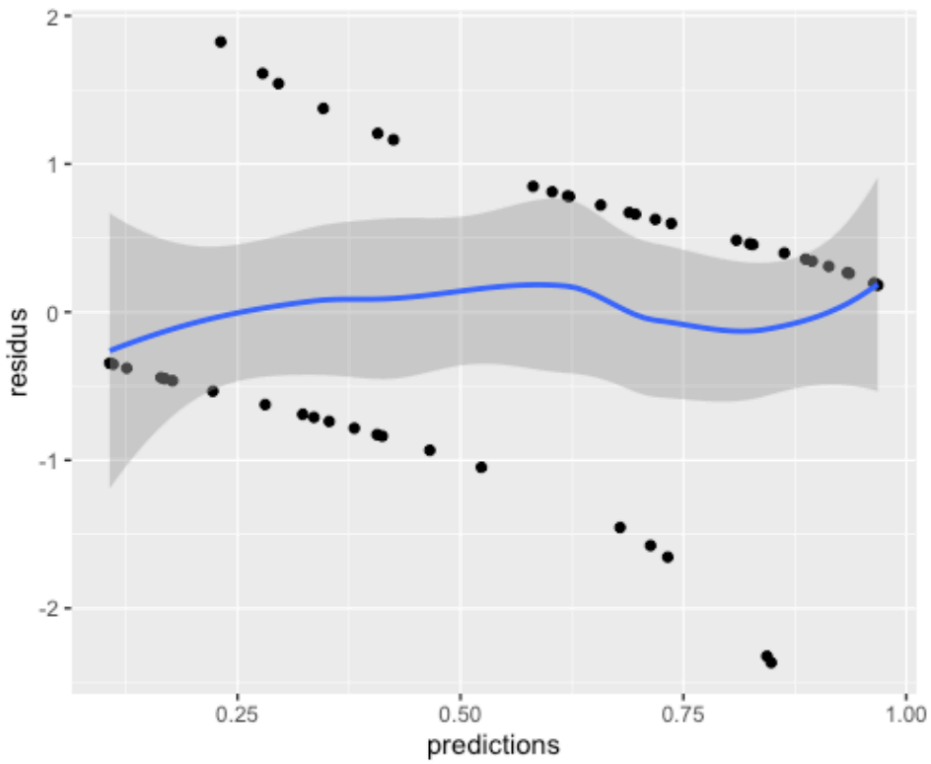
Autrement dit, le résidu de déviance d'une observation correspond à la racine carrée de la déviance de cette observation, à laquelle on colle le signe de la différence entre notre observation et la prédiction. Comme toujours, il ne faut pas s'enfarger dans les détails mathématiques. L'important est de comprendre que dans un GLM, on utilise les résidus de déviance, mais qu'ils s'interprètent comme des résidus ordinaires : un résidu négatif = sur-estimation, résidus positif = sous-estimation. Plus le résidu de déviance est grand (en absolu), moins bon est notre modèle pour cette observation.

À partir de ce moment, on peut effectuer toutes nos validations habituelles concernant l'absence de patrons dans les résidus de notre modèle, mais en effectuant nos vérifications à l'aide des résidus de déviance plutôt que des résidus ordinaires. Même chose pour les distances de Cook concernant l'influence de chacune des observations.

L'interprétation des graphiques est cependant beaucoup plus difficile. Voici par exemple le graphique des résidus de déviance de notre modèle prédisant la survie en fonction de la longueur du poisson :

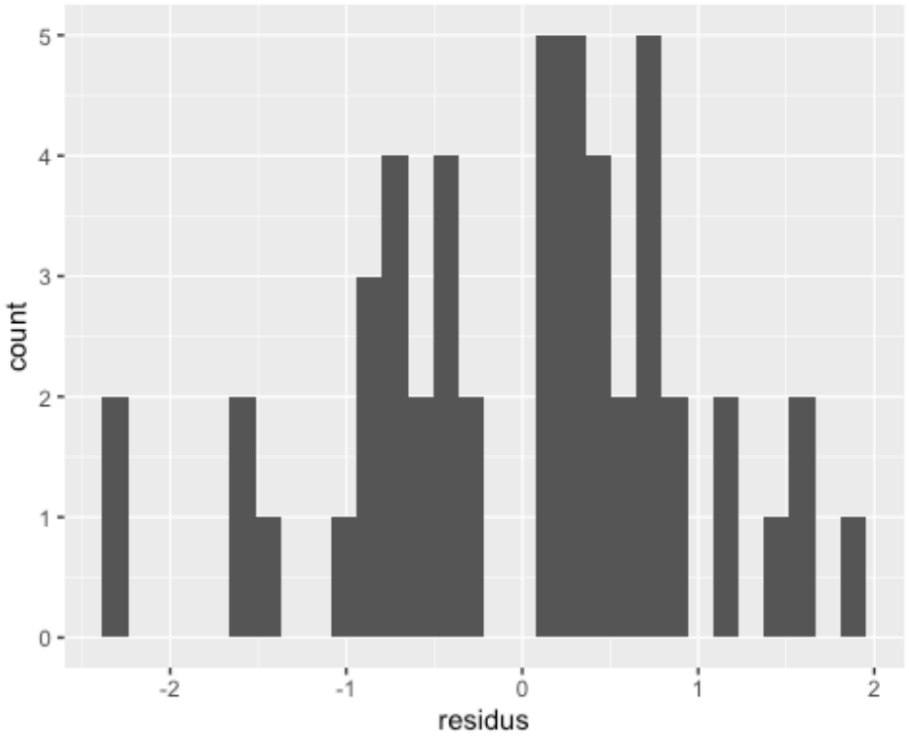


Comme nos observations ne contiennent que deux valeurs, les résidus sont inévitablement organisés en deux lignes. Ce n'est pas un problème en soi. Par contre, notre œil n'est pas particulièrement bon pour voir si les résidus ont tendance à être plus importants à un endroit ou à un autre du graphique. C'est pourquoi il est souvent pratique de passer une courbe de lissage dans le nuage de points pour nous aider à se faire une tête :



Avec la courbe, il est plus évident que les résidus sont relativement équivalents sur l'ensemble du spectre. Que notre modèle ne s'éloigne pas de zéro plus à un endroit qu'à un autre.

On peut aussi jeter un coup d'oeil à la distribution des résidus de déviance pour voir si elle forme une distribution normale :



Cependant, il est important de comprendre que les résidus de déviance vont habituellement tendre vers la normalité, mais il est difficile de prédire si pour un jeu de données en particulier elles vont réussir à l'atteindre. Ce n'est pas critique que la distribution de vos résidus ne semble pas normale, particulièrement si votre taille d'échantillon est relativement petite.

31.8. Labo : Survivre au naufrage du Titanic

Pour ce laboratoire, nous allons faire changement et, plutôt que nos manchots, nous utiliserons un jeu de données classique pour ce genre de modèle : la survie des passagers du Titanic, avec un tableau de données provenant de la librairie `car`.

Nous aurons besoin pour travailler des librairies suivantes :

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(car)
```

```
Loading required package: carData

Attaching package: 'car'

The following object is masked from 'package:dplyr':

    recode
```

```
The following object is masked from 'package:purrr':

    some
```

```
library(MuMIn)
library(visreg)
```

Le tableau de données TitanicSurvival contient 4 variables :

- survived : “yes” / ”no” si le passager a survécu ou non
- sex : “female” / “male”
- age
- passengerClass : “1st”, “2nd”, “3rd”, la première classe étant la plus luxueuse

Par contre, le tableau de données contient beaucoup de valeurs manquantes. De plus, pour passer à la fonction `glm`, nous aurons besoin que notre variable expliquée (survived) soit convertie en 0/1 plutôt que “yes”/”no”, donc :

```
donnees_glm <- TitanicSurvival |>
  mutate(survived = ifelse(survived == "yes",1,0)) |>
  drop_na()
```

Par la suite, comme avec n’importe quel modèle, la première chose est de bien explorer nos données.

Tout d’abord, allons voir globalement quel pourcentage des gens ont survécu au naufrage :

```
donnees_glm |>
  summarize(mean(survived))
```

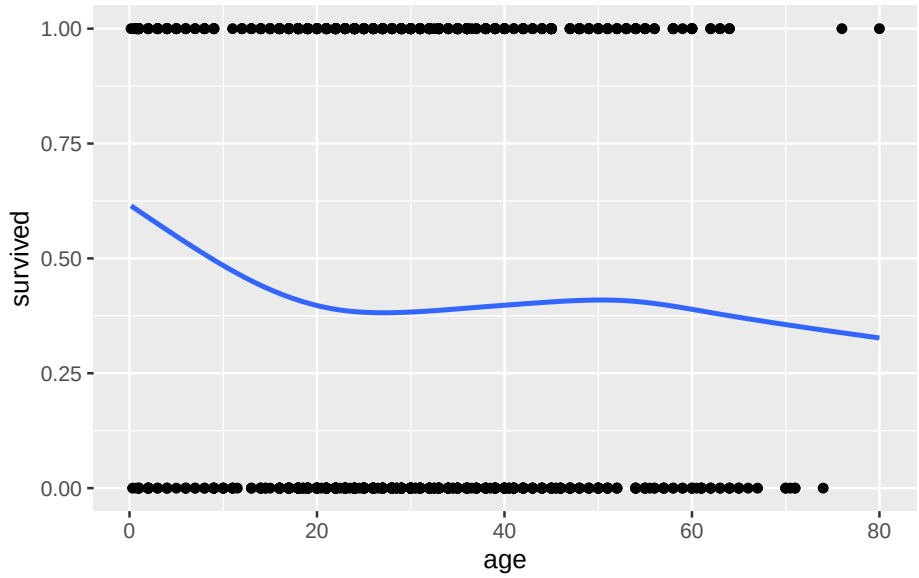
```
  mean(survived)
1      0.4082218
```

Donc, 41% des passagers ont survécu et 59% n'ont pas survécu. Un modèle idiot qui prédit que tous les passagers meurent aura raison 59% du temps. Il faudra garder ce fait en tête pour la suite.

Ensuite, allons explorer nos trois variables explicatives. D'abord les chances de survie en fonction de l'âge :

```
donnees_glm |>
  ggplot(aes(x = age, y = survived)) +
  geom_point() +
  geom_smooth(se = FALSE)
```

```
`geom_smooth()` using method = 'gam' and formula = 'y
~ s(x, bs = "cs")'
```



À première vue, le taux de survie semble diminuer avec l'âge, i.e. plus on est jeune, plus nos chances de survivre sont grandes. Remarquez dans ce graphique que les données observées sont soit des 0 ou des 1, mais que la courbe de lissage nous permet de visualiser la forme de la relation.

On peut aussi se faire une idée des taux de survie associés aux différents niveaux de nos variables catégoriques, par exemple comme ceci :

```
donnees_glm |>
  group_by(sex) |>
  summarize(sum(survived)/n())
```

```
# A tibble: 2 x 2
  sex    `sum(survived)/n()`
  <fct>                <dbl>
1 female               0.753
2 male                 0.205
```

```
donnees_glm |>
  group_by(passengerClass) |>
  summarize(sum(survived)/n())
```

```
# A tibble: 3 x 2
  passengerClass `sum(survived)/n()`
  <fct>                        <dbl>
1 1st                          0.637
2 2nd                          0.441
3 3rd                          0.261
```

Donc, à première vue, les chances de survie des femmes étaient beaucoup plus élevées. De même, les passagers plus riches avaient beaucoup plus de chances de s'en sortir que les passagers moins fortunés.

Maintenant que nous avons exploré nos données, il est enfin temps de lancer notre modèle. La fonction pour calculer un GLM dans R se nomme, vous l'aurez deviné, `glm` :

```
m <- glm(
  survived ~
    age + sex + passengerClass,
  data = donnees_glm,
  family = "binomial",
  na.action = "na.fail"
)
```

Comme pour nos autres modèles de régression on décrit par une formule que l'on veut expliquer la survie par l'âge, le sexe et la classe du passager. On fournit ensuite le tableau de données. L'argument supplémentaire à ajouter lorsque l'on calcule un GLM est qu'il faut spécifier la famille d'erreur de notre variable expliquée, ici, la famille binomiale.

Comme vous vous rappellez peut-être, dans un GLM, il faut aussi spécifier la fonction de lien, mais ici, puisque le lien logit est l'option par défaut de la famille binomiale, il n'est pas nécessaire de le spécifier. Enfin, j'ai aussi ajouté le `na.action = "na.fail"` puisque nous ferons, plus loin, une petite sélection de modèle avec la librairie `MuMIn`.

Ensuite, comme d'habitude, avant d'aller explorer nos résultats, il est important d'aller valider notre modèle. Pour se faire, il faut, comme toujours, aller ajouter les prédictions, les résidus et les distances de Cook à notre tableau de données.

```
donnees_glm <-
  donnees_glm |>
  mutate(
    predictions = predict(m, type = "response"),
    residus = resid(m),
```

```
    D = cooks.distance(m)
  )
```

Deux petites choses à remarquer ici. Tout d'abord, au moment d'extraire les prédictions, il est important de spécifier à R que l'on veut les prédictions de type réponse. C'est-à-dire que l'on veut obtenir les probabilités de survie pour chacun des passagers, et non le logit de la probabilité. Comme mentionné ci-haut, il est aussi important d'extraire les résidus de type déviance plutôt que les résidus bruts, mais ici, R comprend automatiquement que si on extrait les résidus d'un GLM, on veut les résidus de déviance.

Donc, la première chose que l'on peut valider à propos du modèle est de savoir si nos variables étaient trop colinéaires et pouvaient entraîner des instabilités dans notre modèle. On peut le vérifier avec le vif, comme dans un modèle linéaire ordinaire :

```
vif(m)
```

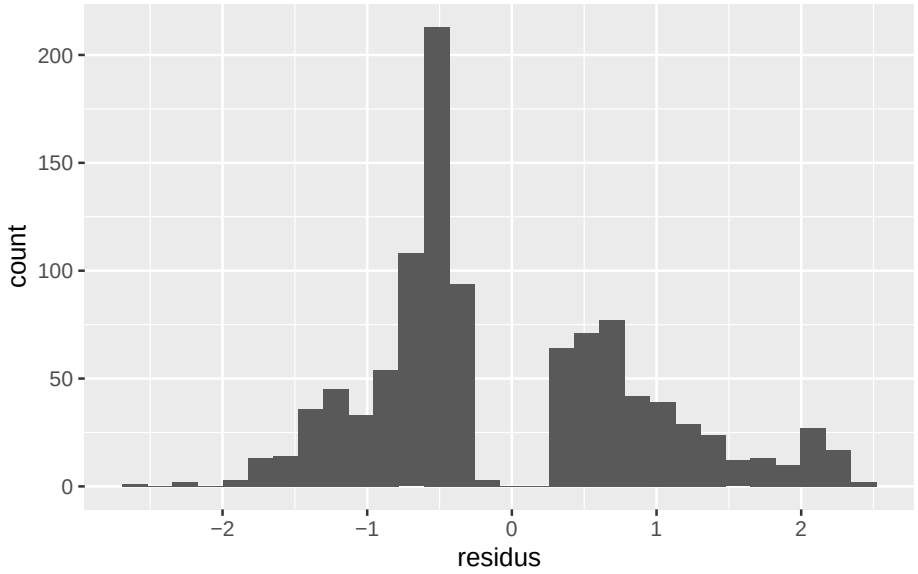
```
                   GVIF Df GVIF^(1/(2*Df))
age            1.354170  1        1.163688
sex            1.052059  1        1.025699
passengerClass 1.414640  2        1.090590
```

Ici, aucun problème, toutes les valeurs sont bien en-dessous de 4.

Deuxième chose que l'on peut regarder ensuite est la distribution des résidus. Comme discuté plus haut, des problèmes de normalité ici ne sont pas nécessairement indicatifs d'un mauvais modèle, mais il est tout de même intéressant de les regarder pour se donner une idée du comportement de notre modèle :

```
donnees_glm |>
  ggplot(aes(residus)) +
  geom_histogram()
```

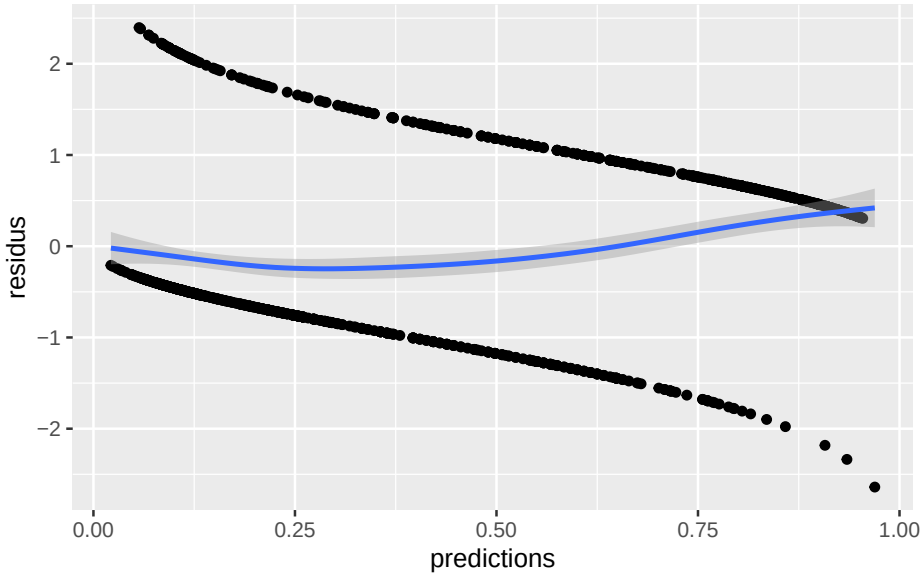
```
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```



Le graphique le plus important à vérifier est la distribution des résidus en fonction des valeurs prédites :

```
donnees_glm |>
  ggplot(aes(x = predictions, y = residus)) +
  geom_point() + geom_smooth()
```

```
`geom_smooth()` using method = 'gam' and formula = 'y
~ s(x, bs = "cs")'
```



Comme discuté plus haut, il est souvent nécessaire d'ajouter une courbe de lissage à ce graphique afin d'aider notre œil à trouver des patrons. En général, les résidus sont plutôt homogènes, à part la partie vers les grandes probabilités qui ont tendance à être légèrement sous-estimées. On comprend aussi avec ce graphique pourquoi notre histogramme avait une drôle d'allure. Il n'y a à peu près pas de résidus tout près de zéro. La densité des points est beaucoup plus forte dans les valeurs intermédiaires autour de 50% de chances de survie.

Il serait important ici d'aller explorer aussi les résidus en fonction de l'âge, du sexe et de la classe de passager, mais je laisse cet exercice au lecteur afin de ne pas allonger inutilement cette section.

Une fois notre modèle inspecté, on peut effectuer une petite sélection de modèle pour s'assurer que toutes les variables pour lesquelles nous avions des hypothèses sont effectivement importantes. Pour cela, on peut utiliser, entre autres, l'approche par AIC :

```
dredge(m)
```

```
Fixed term is "(Intercept)"

Global model call: glm(formula = survived ~ age + sex +
passengerClass, family = "binomial",
    data = donnees_glm, na.action = "na.fail")
---
Model selection table
  (Intrc)       age pssnC sex df   logLik   AICc
8  3.5220 -0.034390     +   +  5 -491.227  992.5
7  2.1600               +   +  4 -506.899 1021.8
5  1.1120                   +  2 -551.004 1106.0
6  1.2350 -0.004254         +  3 -550.669 1107.4
4  2.0670 -0.037410     +       4 -627.846 1263.7
3  0.5638               +       3 -652.942 1311.9
2 -0.1365 -0.007899              2 -705.691 1415.4
1 -0.3713                        1 -707.310 1416.6
   delta weight
8   0.00      1
7  29.32      0
5 113.51      0
6 114.85      0
4 271.22      0
3 319.40      0
2 422.88      0
1 424.11      0
Models ranked by AICc(x)
```

Autrement dit, nos 3 variables sont clairement importantes pour expliquer la survie des passagers.

On peut maintenant, enfin, regarder les résultats de notre modèle :

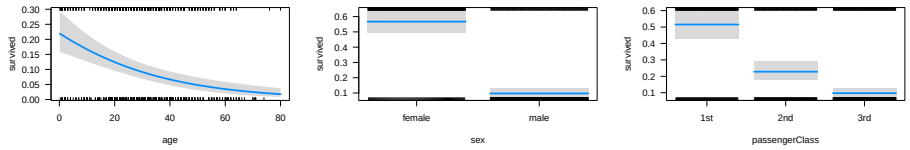
```r
summary(m)
```

```
Call:
glm(formula = survived ~ age + sex + passengerClass,
family = "binomial",
    data = donnees_glm, na.action = "na.fail")

Coefficients:
                   Estimate Std. Error z value
(Intercept)         3.522074   0.326702  10.781
age                -0.034393   0.006331  -5.433
sexmale            -2.497845   0.166037 -15.044
passengerClass2nd -1.280570   0.225538  -5.678
passengerClass3rd -2.289661   0.225802 -10.140
                  Pr(>|z|)
(Intercept)        < 2e-16 ***
age               5.56e-08 ***
sexmale            < 2e-16 ***
passengerClass2nd 1.36e-08 ***
passengerClass3rd  < 2e-16 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1414.62  on 1045  degrees of freedom
Residual deviance:  982.45  on 1041  degrees of freedom
AIC: 992.45

Number of Fisher Scoring iterations: 4
```

Donc, de ces résultats, on peut d'abord conclure que les chances de survie au naufrage diminuent effectivement avec l'âge (i.e. plus on est jeune, plus on survit). Les hommes ont moins de chance de survivre que les femmes. Enfin, plus un passager était riche (1ère vs. 3ème classe), plus il avait des chances de survivre. Ces résultats sont très rassurants, puisque l'on avait constaté la même chose dans notre exploration visuelle.

Par contre, comme discuté plus haut, ces chiffres sont difficiles à se mettre en tête. Par exemple, le paramètre associé à l'âge de -0,03 nous indique que le logit de la probabilité de survie diminue de 0,03 par année d'âge. C'est très peu parlant! Le mieux dans ce genre de situation est de visualiser chacun des paramètres avec un graphique :

```
visreg(m, "age", scale = "response")
visreg(m, "sex", scale = "response")
visreg(m, "passengerClass", scale = "response")
```



Remarquez que pour chaque appel à visreg, on ajoute l'option `scale="response"`, qui permet d'obtenir les résultats en termes de probabilités, plutôt qu'à l'échelle du lien logit.

En observant les pentes partielles associées aux paramètres, l'ampleur de l'importance de l'âge pour la survie est beaucoup plus claire. Pour un même sexe et une même classe, on passe d'environ 20% de chances de survie pour un enfant à 10% à 25 ans et pratiquement 0% passé la soixante. Si ces chiffres vous paraissent faibles, rappelez-vous que `visreg` calcule un graphique des effets marginaux. Dans ce cas-ci, la courbe

correspond aux hommes et aux passagers de 3e classe, puisque ce sont les groupes les plus nombreux dans la base de données.

On voit aussi qu'une fois la correction pour l'âge appliquée, un femme avait plus de 50% des chances de survire, alors qu'un homme avait moins de 10%, etc.

Par contre, bien qu'on ait une idée de la qualité du modèle grâce aux intervalles de confiance visuelles, nous n'avons pas de chiffre pour valider notre impression. Si on veut calculer la précision de notre modèle, on pourrait rapidement compter le nombre de fois où notre prédiction se révèle être la même que la valeur observée.

Pour se faire, on peut convertir toutes les prédictions >50% en 1 (survie) et celles <= 50% en 0 (décès). Ensuite, si l'on sait que R considère les valeur TRUE comme des 1 et les FALSE comme des zéros, on peut facilement calculer la précision de notre modèle, comme ceci :

```
donnees_glm |>
  mutate(
    predictions_01 = ifelse(predictions > 0.5,1,0)
  ) |>
  summarize(
    precision = sum(predictions_01 == survived)/n()
  )
```

```
  precision
1 0.7848948
```

Autrement dit, notre modèle a raison 78% du temps avec ses prédictions. Cependant, il ne faut pas oublier qu'un modèle idiot prédisant que tout le monde décède aura déjà raison 59% du temps.

Enfin, on peut calculer le D de Tjur pour se donner une idée de comment notre modèle réussit à distinguer les deux groupes, avec le calcul expliqué ci-haut :

```
donnees_glm |>
  group_by(survived) |>
  summarize(prediction_moyenne = mean(predictions))
```

```
# A tibble: 2 x 2
  survived prediction_moyenne
     <dbl>              <dbl>
1        0              0.255
2        1              0.631
```

```
0.631-0.255
```

```
[1] 0.376
```

Notre modèle explique donc 37% de la variance dans les taux de survie. Ce n'est pas si mal, mais loin d'être aussi fort que notre précision de 78% pouvait le laisser supposer.

31.9. Ouverture vers d'autres techniques

En terminant, je tenais à vous ouvrir la porte sur le fait qu'à partir d'ici de grandes possibilités de modélisation s'ouvrent à vous. Nous avons vu dans ce chapitre comment modéliser une variable en suivant la loi binomiale, mais on peut aussi le faire avec d'autres lois de probabilités comme celle de Poisson, Gamma, etc. Plusieurs distributions sont disponibles avec la fonction glm de base de R, mais il existe plusieurs librairies additionnelles fournissant un paquet d'autres lois (entre autres Beta, Binomiale-négative, etc.)

De même, bien que nous ayons vu les GLM et les modèles mixtes séparément, dans la réalité, ces deux techniques peuvent être combinées dans un même modèle, que l'on nomme alors de son acronyme anglais GLMM

(Generalized Linear Mixed Models). Les librairies `glmmTMB` et `lme4` sont, entre autres, conçues exactement pour ce genre de tâches.

partie VI.

Autres techniques

32. Les arbres de régression

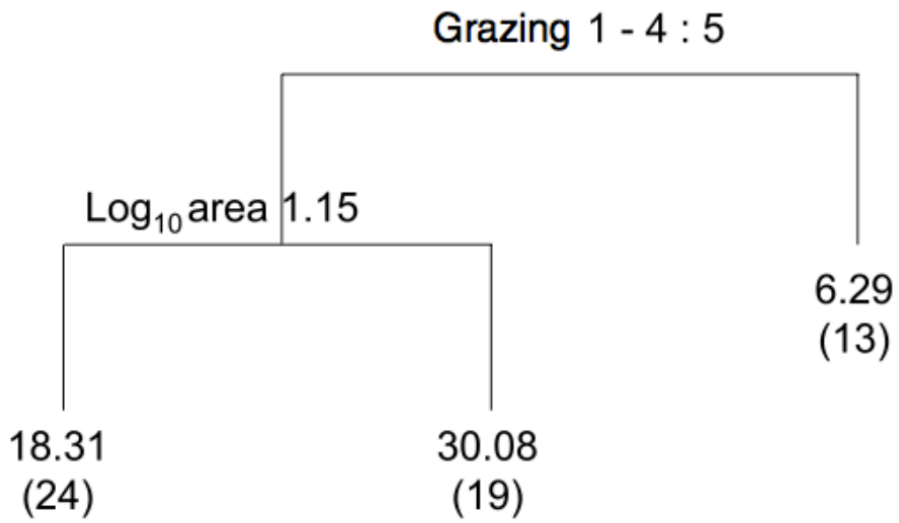
32.1. Introduction

La totalité des modèles rencontrés jusqu'ici dans le cours assument que la relation entre les variables étudiées est linéaire (ou du moins, qu'elle peut être linéarisée à l'aide de transformations). Évidemment, cette assomption ne tient pas toujours la route dans la vraie vie.

Dans ce chapitre, nous verrons une technique, les arbres de régression, qui assument plutôt que les réponses fonctionnent sous forme de seuils, de coupures, où les valeurs comportent des plateaux et des changements abrupts.

32.2. Exemple de sortie

Comme les arbres de régression sont conceptuellement très différents de ce que nous avons vu jusqu'à présent, commençons par voir à quoi peuvent ressembler les sorties de cette technique. Nous en verrons ensuite le fonctionnement et les particularités.



Cette figure est le résultat d'une modélisation de la richesse en espèces d'oiseaux à partir de variables décrivant le paysage (area, la surface de la parcelle et grazing le niveau de broutement, et d'autres, qui n'ont pas été retenues) à l'aide d'un arbre de régression.

La figure se lit de haut en bas, avec à chaque intersection un choix à faire, qui nous amène à l'une ou l'autre de sorties du modèle.

Cette figure s'interprète comme suit :

- Si le niveau de broutement est 5, le modèle prédit 6,29 espèces
- Si le broutement est entre 1 et 4 et que la surface est plus petite que que 1,15 (à l'échelle log), on prédit 18,31 espèces
- Si le broutement est entre 1 et 4 et que la surface est plus grande que 1,15 (à l'échelle log), on prédit 30,08 espèces.

Les chiffres entre parenthèses indiquent le nombre d'observations qui arrivent à ce point dans notre tableau de données, lorsque l'on parcourt

l'arbre avec chacune d'entre elles.

On nomme **noeuds** (*node*) les intersections dans l'arbre, qui contiennent les conditions et **feuilles** (*leaf*) les points où on prédit une valeur. L'arbre ci-dessus contient donc 2 nœuds et 3 feuilles.

Remarquez que l'arbre est à l'envers, avec les feuilles vers le bas. Ce n'est pas la chose la plus logique du monde, mais c'est comme ça que les inventeurs de la méthode ont proposé de faire les graphiques de sortie.

32.3. Avantages et inconvénients

Le principal avantage des arbres de régression est que les sorties sont faciles à interpréter, particulièrement lorsque vient le moment de discuter des résultats avec des praticiens ou des décideurs. Les points de coupure s'interprètent beaucoup plus facilement que l'ordonnée à l'origine et les pentes partielles d'une régression multiple.

L'autre avantage de cette méthode est que, outre l'assomption de normalité de la variable expliquée, elle n'a pas d'assomption quant à linéarité des relations avec les autres variables.

Conceptuellement, le principe des points de coupure est particulièrement approprié lorsque l'on modélise des processus décisionnels, p. ex. si on tente de déterminer si une espèce va occuper ou non une parcelle.

Cette méthode comporte par contre aussi certains inconvénients. Puisque les prédictions de l'arbre sont groupées, ces prédictions peuvent être parfois grossières lorsque l'on tente d'ajuster l'arbre de régression à des réponses linéaires (dans ces cas, la régression linéaire aurait été plus appropriée...). L'autre désavantage majeur de cette technique est que la structure de l'arbre peut parfois être instable, où

un petit changement dans les données modifiera la structure de l'arbre résultant[1].

32.4. Comment se construit l'arbre

Lorsque les anglophones décrivent le mécanisme de construction d'un arbre de régression, il utilisent les termes top down et greedy.

Top down, d'abord, parce que la construction de l'arbre se fait du haut vers le bas. On part du point où toutes les données sont dans un même groupe, on les sépare, on sépare ensuite les séparations, etc. Mais on ne remonte jamais pendant la construction. Un point de coupure n'est jamais modifié après qu'il ait été établi.

On dit aussi que l'algorithme est greedy (avide en français), parce qu'au moment de définir un point de coupure, il ne se soucie que de cette coupure en particulier, et pas comment elle va influencer la qualité totale du reste de l'arbre. Il ne regarde pas les conséquences plus loin dans le processus.

L'algorithme doit fonctionner de cette façon parce que, même en 2025, tester tous les points de coupure et toutes les configurations d'arbres possibles avec nos données serait juste impossiblement long.

À chaque fois que l'algorithme essaie de trouver une séparation, il choisira celle qui minimise la somme des carrés des résidus.

L'algorithme va couper et couper les données, jusqu'à atteindre une condition pré-déterminée. Cette condition peut être différentes choses, mais on se base souvent sur un nombre minimal d'observation par feuille (p. ex. 5).

[1] https://link-springer-com.biblioproxy.uqtr.ca/article/10.1007/s10021-005-0054-1

32.5. Labo : Les arbres de régression

Comme initiation aux arbres de régression dans le logiciel R, nous allons continuer de travailler avec les manchots de Palmer pour tenter de prédire leur poids. Nous utiliserons les mêmes variables qu'aux chapitres précédent, soit la longueur des ailes, le sexe et l'espèce.

Remarquez qu'ici, nous n'aurons pas besoin de spécifier de termes d'interaction. Ces derniers sont plutôt implicites dans l'enchaînement des noeuds. Une interaction entre le sexe et l'espèce sera représentée par un premier noeud avec le sexe, puis, dans chacune des branches (une pour les mâles et une pour les femelles), on en aurait une seconde pour le choix de l'espèce, etc.

Il est également inutile de se soucier des niveaux de références pour nos facteurs. Toutes les catégories sont directement présentes dans le modèle.

Pour ajuster notre arbre de régression, nous utiliserons la librairie nommée `tree`. Vous devrez donc l'installer avant de lancer le code. Sachez qu'il existe d'autres implémentations de l'algorithme d'arbre de régression dans R, entre autres dans la librairie `cart`.

Donc, activons d'abord nos librairies et préparons notre jeu de données :

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ----------------- tidyverse_conflicts() --
```

```
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(palmerpenguins)
library(tree)

pour_arbres <-
  penguins |>
  drop_na(species, sex, body_mass_g, flipper_length_mm,
   ↪  island)
```

La seule chose à vérifier avant de lancer notre modèle est que la distribution de notre variable expliquée (`body_mass_g`) suit une distribution normale. Pour les autres, l'algorithme trouvera sans problèmes les points de coupure sans besoin de transformations. Comme il s'agit du Xe chapitre dans lequel nous utilisons les manchots de Palmer, on sait que le poids des manchots est distribué normalement!

Enfin, nous avons discuté dans le Chapitre 29 que les variables catégoriques peuvent exister sous 2 formats (`character` et `factor`) et que la majorité des fonctions de R ne font pas la différence entre les deux. Hors, la fonction `tree`, elle, s'en soucie. Donc, si jamais vos variables qualitatives n'étaient pas encore en facteur avant d'arriver ici, il faudrait le faire explicitement. Mais dans notre cas, c'est déjà fait pour nous dans les données originales.

La fonction pour ajuster un arbre de régression se nomme `tree`, et s'utilise à l'aide d'une formule et d'un argument `data`, exactement comme la fonction `lm` :

```
m <- tree(
  body_mass_g ~
  flipper_length_mm + sex + species,
  data = pour_arbres
)
```

Par la suite, on peut inspecter notre objet de résultats :

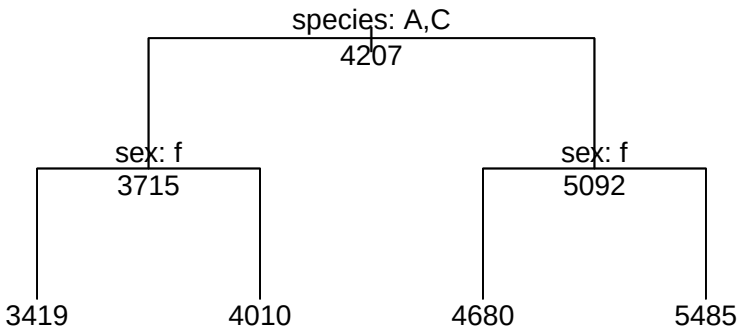
```
summary(m)
```

```
Regression tree:
tree(formula = body_mass_g ~ flipper_length_mm + sex +
species,
    data = pour_arbres)
Variables actually used in tree construction:
[1] "species" "sex"
Number of terminal nodes:  4
Residual mean deviance:  97680 = 32140000 / 329
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-760.30 -219.20   15.16    0.00  220.30  815.20
```

On voici ici, entre autres, que notre arbre contient 4 feuilles (*terminal nodes*). Il s'agit plus ou moins de la seule information intéressante pour nous dans cette sortie.

Ensuite, on peut faire afficher la structure de notre arbre, à l'aide d'une combinaison de la fonction `plot` et `text` :

```
plot(m, type = "uniform")
text(m,pretty = 1, all = TRUE)
```



Dans cette sortie, on peut donc constater que le premier point de coupure est basé sur l'espèce. À gauche Adélie (A) et Chinstrap (C), pour une moyenne de 3715 g et à droite le reste (i.e. le manchot Gentoo) pour une moyenne de 5092 g. Ensuite, pour la combinaison Adélie et Chinstrap, le deuxième point de coupure est basé sur le sexe, avec les femelles à gauche, pour 3419 g et les mâles à droite pour 4010 g. Le même phénomène se reproduit du côté des manchots Gentoo, avec les femelles à gauche (4680 g) et les mâles à droite (5485 g). Donc, contrairement au modèle linéaire, l'arbre de régression n'a pas utilisé la longueur des ailes pour prédire le poids des manchots.

Remarquez que cette fonction n'utilise pas `ggplot`. À l'heure actuelle, il n'existe malheureusement pas de façon simple de reproduire ce graphique avec `ggplot`. Vous devrez donc vous en tenir aux graphiques de base de R, ou sortir vos talents de Photoshop ;-)

Enfin, si on veut avoir une idée de la performance de notre modèle, on

peut en extraire les prédictions, et calculer soit une valeur de pseudo-r2, ou soit une mesure de l'erreur moyenne des prédictions (*mean absolute error*).

```
preds <- predict(m)
```

Pour calculer le pseudo-r^2, on calcule la corrélation entre les prédictions du modèle et les valeurs observées, et un met ce chiffre au carré. On parle de pseudo-r^2 plutôt que de r^2 pur parce qu'on ne parle pas ici de variance expliquée ou de variance résiduelle comme dans la définition originale, mais le chiffre s'interprète de la même manière.

```
cor(preds,pour_arbres$body_mass_g)^2
```

```
[1] 0.850702
```

Donc, c'est légèrement moins bon que notre modèle linéaire avec des variables qualitatives (Chapitre 29) qui atteignait 87%, mais aussi plus facile à interpréter pour un non-initié.

Pour calculer l'erreur de prédiction moyenne du modèle, nous allons calculer la différence, pour chaque observation, entre la valeur prédite et la valeur observée, prendre la valeur absolue de ces différences, et en faire la moyenne. Ça sonne compliqué, mais ça se calcule en une seule ligne de R :

```
mean(abs(preds-pour_arbres$body_mass_g))
```

```
[1] 249.5585
```

Donc, notre arbre de régression explique 85% de la variance du poids des manchots, et ses prédictions se trompent en moyenne de 249 g.

> **Note**
>
> Notez que l'erreur de prédiction moyenne peut aussi être calculée pour des modèles de régression. Il est simplement plus rare que les régression soit entraînées pour effectivement calculer des prédictions comme tel.

32.6. L'élagage

Il est reconnu dans la littérature qu'un arbre de régression construit à l'aide de la procédure décrite précédemment aura tendance à être trop complexe. Autrement dit, sur-ajusté aux données. Il ne généralisera pas très bien sur de nouvelles données qu'il ne connaît pas.

Il faut donc par la suite simplifier l'arbre, afin de régler ce problème. Dans le jargon des arbres de régression, on nomme cette opération élagage (*pruning*).

32.7. La validation croisée

Avant d'aller plus loin dans le processus d'élagage, il importe de définir un concept important, qui est la validation croisée. Depuis le début de nos analyses, lorsque nous voulions valider la performance d'un modèle, nous observions ses résidus.

Par contre, à chaque fois, nous utilisions les mêmes données pour évaluer la performance du modèle que celles utilisées pour l'ajuster. Cela nous laisse toujours dans le doute quant à savoir si le modèle fonctionnera aussi bien sur un autre jeu de données. On ne sait pas à quel point il généralise bien.

La validation croisée permet de régler ce problème en séparant nos données avant de commencer l'ajustement. Le principe est de prendre une partie de nos données pour ajuster le modèle, et l'autre pour en valider la performance réelle sur des données nouvelles.

> **Note**
>
> Notez que cette technique n'est pas restreinte aux arbres de régression. Elle pourrait s'appliquer à tout type de modèle prédictif pour lesquels on veut valider leur performance réelle, c'est-à-dire leur capacité à prédire de nouvelles données. Il existe entre autres une fonction nommée `cv.lm` dans la librairie **DAAG** qui permet d'appliquer la validation croisée sur une régression linéaire. Nous avions simplement évité le sujet jusqu'ici car la régression linéaire a beaucoup moins tendance à produire des modèles surajustés que les arbres de régression.

32.8. La validation croisée à k groupes

Une des façons classique de faire la validation croisée est d'utiliser la validation par k groupes (*K-fold cross-validation*). Cette technique consiste à :

- Séparer le jeu de données en k groupes aléatoires.
- Ajuster un modèle avec les données de k-1 groupes
- Tester avec les données restantes
- Recommencer ce processus par chacune des k fractions.

Par exemple, si j'ai 60 observations dans mon tableau de données et que je fais ma validation croisée avec k=3, j'obtiens trois groupes au hasard, contenant les données que l'on numérote respectivement 1-20, 21-40 et 41-60. Je devrais donc ajuster et tester mon modèle 3 fois :

- Ajuster avec 1-40 et tester avec 41-60
- Ajuster avec 21-60 et tester avec 1-20
- Ajuster avec 1-20+41-60 et tester avec 21-40.

On calcule ensuite l'erreur moyenne des k modèles pour obtenir une idée de la performance globale.

En général, on utilise un k de 4 ou 5, afin d'obtenir un bon compromis entre la quantité de données dans chacun des modèles, mais aussi que nos différents modèles ne soient pas trop corrélés ensemble.

Il existe d'autres façons de faire, par exemple le *Leave One Out cross-validation* (LOOCV) où k est égal au nombre d'observations dans la base de données. Cette stratégie est particulièrement utile lorsque notre jeu de données est très petit, ce qui cause beaucoup de variabilité entre les k modèles ajustés. Le désavantage du LOOCV est que les k modèles testés sont extrêmement corrélés entre eux, puisqu'ils ne diffèrent que par une seule observation.

32.9. Stratégie d'élagage

Même en utilisant le principe de la validation croisée, on ne règle pas magiquement notre problème de modèle sur-ajusté car nous n'avons toujours pas la possibilité d'explorer tous les arbres possibles et imaginables. Je le répète, ça serait beaucoup trop long, même pour un ordinateur moderne.

Pour contourner ce problème, il faut se munir d'une mesure d'ajustement complémentaire, qui pénalise pour les modèles trop complexes, un peu comme le R2-ajusté peut le faire pour la régression linéaire multiple.

Plusieurs algorithmes d'arbres de régression utilisent donc une mesure d'ajustement où on ajoute à la somme des carrés des résidus le nombre

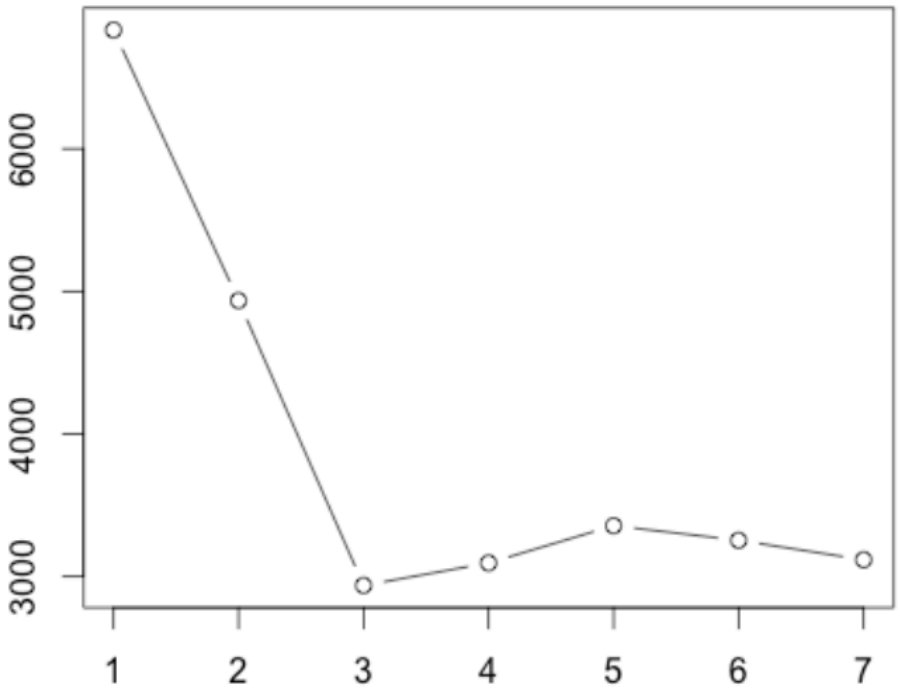
de feuilles multiplié par un paramètre nommé α (alpha), qui définit un *coût* associé à la complexité de l'arbre.

La stratégie à adopter par la suite est donc de faire augmenter progressivement la valeur de α à partir de zéro et pour chaque valeur de α que l'on veut tester, on cherche (plus précisément, l'ordinateur cherchera pour nous!) l'arbre qui minimise cette valeur.

L'avantage de cette méthode est que chacun des arbres trouvés pour des valeurs de α différentes sera un sous-ensemble simplifié de notre arbre original. Celui où α=0 correspondant à notre arbre complet, avant l'élagage, puisqu'il n'en *coûte* rien d'avoir beaucoup de feuilles.

On calcule alors l'erreur de prédiction de chacun de ces arbres à l'aide de la validation croisée et on conserve l'arbre le plus performant sur de nouvelles données.

On pourrait obtenir un graphique semblable à ceci, où on met en relation l'erreur moyenne de la validation croisée avec la taille de l'arbre :



Dans ce cas précis, nous conserverons pour interprétation l'arbre contenant 3 feuilles, puisqu'il possède l'erreur la plus faible.

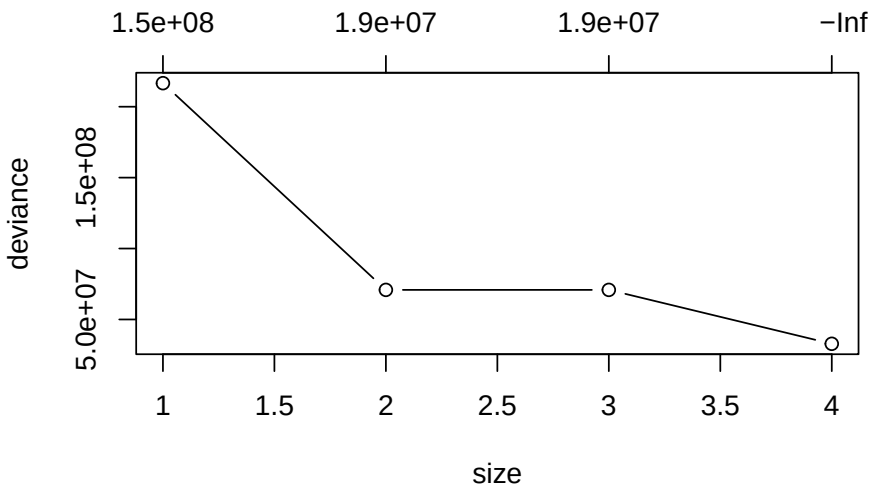
32.10. Labo : L'élagage d'un arbre de régression

Donc, voyons maintenant comment on peut essayer d'élaguer l'arbre ajusté précédemment, pour s'assurer qu'il n'est pas sur-ajusté à nos données et qu'il généralise bien sur de nouvelles données.

La fonction pour effectuer l'élagage de notre arbre en se basant sur la validation croisée et un paramètre de coût de complexité se nomme

`cv.tree` (*Cross-Validation TREE*). Comme nous avons eu jeu de données sufisamment grand, nous pourrons utiliser la validation croisée avec k=5 groupes.

```
validation_croisee <- cv.tree(m,K = 5)
plot(validation_croisee, type = "b")
```



Il se peut que votre graphique soit légèrement différent du mien, puisque les groupes sont générés de façon aléatoire, mais la conclusion qualitative devrait rester la même.

Dans ce graphique, on cherche la taille d'arbre (*size*) qui minimise notre mesure d'erreur (*deviance*). Dans notre cas, l'arbre idéal contiendrait encore 4 feuilles.

> **Note**
>
> Pour des raisons techniques, la librairie `tree` calcule l'erreur en terme de déviance plutôt qu'en somme des carrés des erreurs. Mais nos interprétations restent les mêmes, pluisque la déviance est toujours proportionnelle à la somme des carrés.

Si jamais la hauteur des points est difficile à interpréter parce qu'ils sont très près les uns des autres, on peut aussi aller voir directement les chiffres dans notre objet :

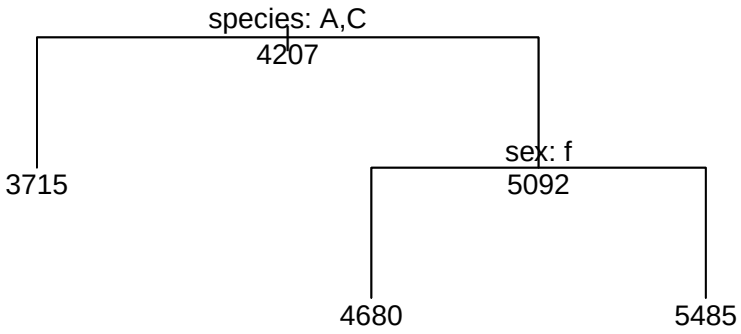
```
validation_croisee
```

```
$size
[1] 4 3 2 1

$dev
[1]  32833871  70853924  70853924 216591273

$k
[1]      -Inf  18694217  19271024 145156589

$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"
```

Donc, dans la vraie vie, on aurait pu arrêter ici. Notre arbre original n'a pas besoin d'être élagué.

Mais pour l'exercise, allons voir comment on aurait dû s'y prendre pour récupérer le meilleur arbres à 3 feuilles plutôt qu'à 4. Pour se faire, on doit utiliser la fonction `prune.tree` (égaluer.arbre).

```
arbre_a_3_feuilles <- prune.tree(m,best = 3)
plot(arbre_a_3_feuilles, type = "uniform")
text(arbre_a_3_feuilles,pretty = 1, all = TRUE)
```



Remarquez que dans cet arbre, on aurait pas séparé par sexe les individus de l'espèce Adélie ou Chinstrap.

On peut par la suite mesurer la performance de ce nouvel arbre, de la même façon dont nous avions mesuré la performance du premier :

```
preds <- predict(arbre_a_3_feuilles, newdata =
 ↪  pour_arbres)
cor(preds,pour_arbres$body_mass_g)^2
```

```
[1] 0.763857
```

```
mean(abs(preds-pour_arbres$body_mass_g))
```

```
[1] 309.2313
```

En enlevant la dernière feuille, on aurait perdu presque 10% d'explication et presque doublé notre erreur de prédiction.

> **Note**
>
> Notez qu'à cette étape, presque à tout coup, votre arbre élagué performera moins bien que votre arbre original. Comme discuté plus haut, les arbres de régression ont tendance à être surajusté. Votre arbre élagué sera cependant assurément le meilleur pour faire des prédictions sur de nouvelles données.

32.11. Un monde qui s'ouvre devant vous.

Il existe plusieurs autres techniques pour essayer d'ajuster des modèles statistiques à des réponses non linéaires, par exemple avec les modèles de type GAM (*Generalized Additive Models*) qui permettent d'ajuster une courbe de lissage plutôt qu'une réponse linéaire.

J'ai choisi de vous enseigner les arbres de régression car ils ouvrent la porte sur un monde de possibilités. Les arbres de régression peuvent, entre autres, être utilisés pour travailler avec des variables expliquées catégoriques, par exemple pour prédire la mort ou la survie d'un individu. On parlera alors d'arbres de classification.

Ils sont aussi à la base d'une technique nommée Random Forests (littéralement des forêts aléatoires) qui sont parmi les techniques d'apprentissage automatique (*machine learning*) les plus puissantes

utilisées à ce jour (à l'exception bien évidemment de l'apprentissage profond des réseaux neuronaux).

32.12. Exercice : Les arbres de régression

Pour cet exercice, je vais vous demande de réanalyser le jeu de données de Loyn vu au Chapitre 27, où l'on tentait de prédire l'abondance d'oiseaux dans des parcelles (ABUND), à partir du paysage environnant (AREA, YR.ISOL, DIST, ALT). Mais cette fois-ci, vous utiliserez les arbres de régressions plutôt que la régression multiple.

Je vous demande donc de :

a) Charger le jeu de données et les librairies nécessaires
b) Vérifier ce qu'il y a à vérifier avant de lancer un arbre de régression
c) Ajuster un modèle d'arbre de régression
d) Évaluer la performance de cet arbre (probablement surajusté)
e) Comparer la performance de cet arbre à celle de la régression multiple du Chapitre 27.
f) Élaguer cet arbre en vous basant sur la validation croisée
g) Préparer un arbre final, bien élagué et visualisez-le
h) Évaluer la performance de cet arbre élagué.
i) Selon cet arbre, quel est le pire scénario possible à avoir comme paysage si on veut avoir de fortes abondances d'oiseaux (en vos mots). Et au contraire, quelle est la meilleure façon d'obtenir de fortes abondances?

33. Références

Allaire, JJ, Yihui Xie, Christophe Dervieux, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, et al. 2024. *rmarkdown: Dynamic Documents for R*. https://github.com/rstudio/rmarkdown.

Arnold, Jeffrey B. 2024. *ggthemes: Extra Themes, Scales and Geoms for « ggplot2 »*. https://CRAN.R-project.org/package=ggthemes.

Bartoń, Kamil. 2023. *MuMIn: Multi-Model Inference*. https://CRAN.R-project.org/package=MuMIn.

Breheny, Patrick, et Woodrow Burchett. 2017. « Visualization of Regression Models Using visreg ». *The R Journal* 9 (2): 56?71.

de Vries, Andrie, et Brian D. Ripley. 2024. *ggdendro: Create Dendrograms and Tree Diagrams Using « ggplot2 »*. https://CRAN.R-project.org/package=ggdendro.

Fortran code by Alan Miller, Thomas Lumley based on. 2020. *leaps: Regression Subset Selection*. https://CRAN.R-project.org/package=leaps.

Fox, John, et Sanford Weisberg. 2019. *An R Companion to Applied Regression*. Third. Thousand Oaks CA: Sage. https://socialsciences.mcmaster.ca/jfox/Books/Companion/.

Francisco Rodriguez-Sanchez, et Connor P. Jackson. 2023. *grateful: Facilitate citation of R packages*. https://pakillo.github.io/grateful/.

Horst, Allison Marie, Alison Presmanes Hill, et Kristen B Gorman. 2020. *palmerpenguins: Palmer Archipelago (Antarctica) penguin data*. https://doi.org/10.5281/zenodo.3960218.

Oksanen, Jari, Gavin L. Simpson, F. Guillaume Blanchet, Roeland Kindt, Pierre Legendre, Peter R. Minchin, R. B. O'Hara, et al. 2024. *vegan: Community Ecology Package*. https://CRAN.R-project.org/package=vegan.

Pedersen, Thomas Lin. 2024. *ggforce: Accelerating « ggplot2 »*. https://CRAN.R-project.org/package=ggforce.

Pinheiro, José C., et Douglas M. Bates. 2000. *Mixed-Effects Models in S and S-PLUS*. New York: Springer. https://doi.org/10.1007/b98882.

Pinheiro, José, Douglas Bates, et R Core Team. 2023. *nlme: Linear and Nonlinear Mixed Effects Models*. https://CRAN.R-project.org/package=nlme.

R Core Team. 2024. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Ripley, Brian. 2023. *tree: Classification and Regression Trees*. https://CRAN.R-project.org/package=tree.

Schloerke, Barret, Di Cook, Joseph Larmarange, Francois Briatte, Moritz Marbach, Edwin Thoen, Amos Elberg, et Jason Crowley. 2024. *GGally: Extension to « ggplot2 »*. https://CRAN.R-project.org/package=GGally.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. « Welcome to the tidyverse ». *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Wood, S. N. 2017. *Generalized Additive Models: An Introduction with R*. 2? éd. Chapman; Hall/CRC.

Wood, S. N. 2003. « Thin-plate regression splines ». *Journal of the Royal Statistical Society (B)* 65 (1): 95?114.

———. 2004. « Stable and efficient multiple smoothing parameter estimation for generalized additive models ». *Journal of the American Statistical Association* 99 (467): 673?86.

———. 2011. « Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models ». *Journal of the Royal Statistical Society (B)* 73 (1): 3?36.

Wood, S. N., N., Pya, et B. S"afken. 2016. « Smoothing parameter and model selection for general smooth models (with discussion) ». *Journal of the American Statistical Association* 111: 1548?75.

Xie, Yihui. 2014. « knitr: A Comprehensive Tool for Reproducible Research

in R ». In *Implementing Reproducible Computational Research*, édité par Victoria Stodden, Friedrich Leisch, et Roger D. Peng. Chapman; Hall/CRC.

———. 2015. *Dynamic Documents with R and knitr*. 2nd éd. Boca Raton, Florida: Chapman; Hall/CRC. https://yihui.org/knitr/.

———. 2024. *knitr: A General-Purpose Package for Dynamic Report Generation in R*. https://yihui.org/knitr/.

Xie, Yihui, J. J. Allaire, et Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown.

Xie, Yihui, Christophe Dervieux, et Emily Riederer. 2020. *R Markdown Cookbook*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown-cookbook.

34. Librairies de code R utilisées dans ce livre

Ce livre n'aurait pas été possible sans la contribution immense de la communauté R et des extraordinaires librairies de code qu'elle produit.

Voici la liste de toutes les librairies utilisées dans ce livre, soit dans les laboratoires, ou pour la création du livre comme tel :

Package	Version	Citation
base	4.4.0	R Core Team (2024)
car	3.1.2	Fox et Weisberg (2019)
GGally	2.2.1	Schloerke et al. (2024)
ggdendro	0.2.0	de Vries et Ripley (2024)
ggforce	0.4.2	Pedersen (2024)
ggthemes	5.1.0	Arnold (2024)
grateful	0.2.4	Francisco Rodriguez-Sanchez et Connor P. Jackson (2023)
knitr	1.46	Xie (2014); Xie (2015); Xie (2024)
leaps	3.1	Fortran code by Alan Miller (2020)
mgcv	1.9.1	S. N. Wood (2003); S. N. Wood (2004); S. N. Wood (2011); S. N. Wood et al. (2016); S. N. Wood (2017)
MuMIn	1.47.5	Bartoń (2023)
nlme	3.1.164	J. C. Pinheiro et Bates (2000); J. Pinheiro, Bates, et R Core Team (2023)
palmerpenguins	0.1.1	Horst, Hill, et Gorman (2020)

Package	Version	Citation
rmarkdown	2.27	Xie, Allaire, et Grolemund (2018); Xie, Dervieux, et Riederer (2020); Allaire et al. (2024)
tidyverse	2.0.0	Wickham et al. (2019)
tree	1.0.43	Ripley (2023)
vegan	2.6.6.1	Oksanen et al. (2024)
visreg	2.7.0	Breheny et Burchett (2017)

Cette liste de librairies a été générée automatiquement par la librairie `grateful`.