TP de Biométrie Semestre 5

Benoît Simon-Bouhet

vendredi 4 octobre 2024

Table des matières

Introduction

Objectifs

Ce livre contient l'ensemble du matériel (contenus, exemples, exercices…) nécessaire à la réalisation des travaux pratiques de **Biométrie** de l'EC *Outils pour l'étude et la compréhension du vivant 4* du semestre 5 de la licence Sciences de la Vie de La Rochelle Université.

À la fin du semestre, vous devriez être capables de faire les choses suivantes dans le logiciel `RStudio` :

- Mettre en forme des tableaux de données : renommer des variables, en créer de nouvelles, modifier des variables existantes, filtrer des lignes, sélectionner des colonnes, etc. (vu au semestre 3)
- Explorer des jeux de données en produisant des graphiques appropriés selon la nature des variables disponibles, leur nombre et l'objectif recherché (vu au semestre 3)
- Explorer des jeux de données en produisant des résumés statistiques de variables de différentes nature (numériques continues ou catégorielles) et pour différentes modalités de facteurs d'intérêt (vu au semestre 4)
- Calculer des indices de position (moyennes, médianes…), de dispersion (écart-types, variances, IQR…) et d'incertitude (erreurs standard, intervalles de confiance…) pour plusieurs sous-groupes de vos jeux de données, et les représenter sur des graphiques adaptés (vu au semestre 4)
- Choisir et formuler des hypothèses adaptées à la question scientifique posée (hypothèses bilatérales ou unilatérales)
- Choisir les tests statistiques permettant de répondre à une question scientifique précise selon la nature de la question posée et la nature des variables à disposition

- Réaliser les tests usuels de comparaison de moyennes (t de Student à 1 ou 2 échantillons, appariés ou indépendants…)
- Vérifier les conditions d'application des tests (graphiquement et avec des tests appropriés tels que les test de Shapiro-Wilk et de Levene), et le cas échéant, réaliser des tests non paramétriques équivalents (test de Welch, de Wilcoxon…)
- Interpréter correctement les résultats des tests pour répondre aux questions scientifiques posées
- Identifier des cohortes dans une population et en étudier les caractéristiques et l'évolution temporelle
- Simuler le comportement de populations théoriques simples suivant des modèles démographiques précis (mortalité exponentielle, croissance exponentielle, croissance logistique, système prédateur-proies de Lotka et Volterra, et systèmes de compétition à 2 ou 3 espèces…)
- Simuler, par chaînes de Markov, les successions écologiques dans un écosystème théorique

Pré-requis

Pour atteindre les objectifs fixés ici, et compte tenu du volume horaire restreint qui est consacré aux TP et TEA de Biométrie au S5, je suppose que vous possédez un certain nombre de pré-requis. En particulier, vous devriez avoir à ce stade une bonne connaissance de l'interface du logiciel `RStudio`, et vous devriez être capables :

1. de créer un `Rproject` et un script d'analyse dans `RStudio`
2. d'importer des jeux de données issus de tableurs dans `RStudio`
3. d'effectuer des manipulations de données simples (sélectionner des variables, trier des colonnes, filtrer des lignes, créer de nouvelles variables, etc.)
4. de produire des graphiques de qualité, adaptés à la fois aux variables dont vous disposez et aux questions auxquelles vous souhaitez répondre
5. de réaliser une exploration statistique de vos données en calculant des indices de position, de dispersion et

d'incertitude pour plusieurs sous-groupes de vos données

> ⚠ Si ces pré-requis ne sont pas maîtrisés
>
> Mettez-vous à niveau de toute urgence en lisant attentivement :
>
> - le livre en ligne de Biométrie du semestre 3
> - le livre en ligne de Biométrie du semestre 4
>
> Vous aurez de toutes façons besoin de vous référer à ces 2 documents régulièrement au cours de ces TP. Donc gardez-les à portée de main et apprenez à naviguez rapidement dans leurs chapitres et sous-chapitres pour retrouver facilement les informations dont vous avez besoin.

Organisation

Volume de travail

Les travaux pratiques et TEA de biométrie auront lieu entre le 4 octobre et le 7 décembre 2024. Durant cette période, un créneau d'1h30 de TP (pour chaque groupe) et un créneau d'1h30 de TEA (pour la promotion entière) sont prévus chaque semaine. Attention, certains TP sont prévus jusqu'à 20 heures, et les TEA ont souvent lieu le samedi matin.

Les séances de TP ont lieu soit en salle MSI 217, soit dans les salles du Pôle Communication Multimédia Réseau (PCMR). Tous les TEA sont à distance.

Au total, chaque groupe aura donc 7 séances de TP et 7 séances de TEA, soit un total de 21 heures prévues dans vos emplois du temps. C'est peu pour atteindre les objectifs fixés et il y aura donc évidemment du travail personnel à fournir en dehors de ces séances. J'estime que vous devrez fournir à peu près une vingtaine d'heures de travail personnel en plus des séances prévues dans votre emploi du temps. Attention donc : pensez bien à prévoir du temps dans vos plannings car le travail personnel est essentiel pour progresser dans cette

matière. J'insiste sur l'importance de faire l'effort dès maintenant : vous allez en effet avoir des enseignements qui reposent sur l'utilisation de ces logiciels jusqu'à la fin du S6 (y compris pendant vos stage et, très vraisemblablement, dans vos futurs masters également). C'est donc maintenant qu'il faut acquérir des automatismes, cela vous fera gagner énormément de temps ensuite. Prévoyez donc de consacrer environ 6h00 par semaine à cette matière (3h prévues dans vos emplois du temps, et 3h de travail personnel).

Modalités d'enseignement

Pour suivre cet enseignement vous pourrez utiliser les ordinateurs de l'université, mais je ne peux que vous encourager à utiliser vos propres ordinateurs, sous Windows, Linux ou MacOS. Lors de vos futurs stages et pour rédiger vos comptes-rendus de TP, vous utiliserez le plus souvent vos propres ordinateurs, autant prendre dès maintenant de bonnes habitudes en installant les logiciels dont vous aurez besoin tout au long de votre licence. Si vous n'avez pas suivi l'EC immersion ou les cours de biométrie de L2 et que les logiciels `R` et `RStudio` ne sont pas encore installés sur vos ordinateurs, suivez la procédure décrite ici. Si vous ne possédez pas d'ordinateur, manifestez vous rapidement auprès de moi car des solutions existent (prêt par l'université, travail sur tablette via Posit cloud…).

> **!** Important
>
> L'essentiel du contenu de cet enseignement peut être abordé en autonomie, à distance, grâce à ce livre en ligne, aux ressources mises à disposition sur Moodle et à votre ordinateur personnel. Cela signifie que **la présence physique lors desTP n'est pas obligatoire pour toutes les séances**.

Plus que des séances de TP classiques, considérez plutôt qu'il s'agit de **permanences non-obligatoires** : si vous pensez avoir besoin d'aide, si vous avez des points de blocage ou des questions sur le contenu de ce document ou sur les exercices demandés, alors venez poser vos questions lors des séances de TP. Vous ne serez d'ailleurs pas tenus de rester pendant

1h30 : si vous obtenez une réponse en 10 minutes et que vous préférez travailler ailleurs, vous serez libres de repartir !

De même, si vous n'avez pas de difficulté de compréhension, que vous n'avez pas de problème avec les exercices de ce livre en ligne ni avec les quizz Moodle, votre présence n'est pas requise. Si vous souhaitez malgré tout venir en salle de TP, pas de problème, vous y serez toujours les bienvenus.

Ce fonctionnement très souple a de nombreux avantages :

- vous vous organisez comme vous le souhaitez
- vous ne venez que lorsque vous en avez vraiment besoin
- celles et ceux qui se déplacent reçoivent une aide personnalisée
- vous travaillez sur vos ordinateurs
- les effectifs étant réduits, c'est aussi plus confortable pour moi !

Toutefois, pour que cette organisation fonctionne, cela demande de la rigueur de votre part, en particulier sur la régularité du travail que vous devez fournir. Si la présence en salle de TP n'est pas requise, **le travail demandé est bel et bien obligatoire** ! Si vous venez en salle de TP sans avoir travaillé en amont, votre venue sera totalement inutile puisque vous n'aurez pas de question à poser et que vous passerez votre séance à lire et suivre ce livre en ligne, choses que vous pouvez très bien faire chez vous. Vous perdrez donc votre temps, celui de vos collègues, et le mien. De même, si vous attendez la 4e semaine pour vous y mettre, vous irez droit dans le mur. Je le répète, outre les heures de TP/TEA prévus dans vos emplois du temps, vous devez prévoir au moins 20 heures de travail personnel supplémentaire.

Je vous laisse donc une grande liberté d'organisation. À vous d'en tirer le maximum et de faire preuve du sérieux nécessaire. Le rythme auquel vous devriez avancer est présenté dans la partie suivante intitulée "Progression conseillée".

Utilisation de Slack

Outre les séances de permanence non-obligatoires, nous échangerons aussi sur l'application Slack, qui fonctionne un peu comme un "twitter privé". Slack facilite la communication des équipes et permet de travailler ensemble. Créez-vous

un compte en ligne et installez le logiciel sur votre ordinateur (il existe aussi des versions pour tablettes et smartphones). Lorsque vous aurez installé le logiciel, cliquez sur ce lien pour vous connecter à notre espace de travail commun intitulé `L3 SV 24-25 / EC outils` (ce lien expire régulièrement : faites moi signe s'il n'est plus valide).

Vous verrez que 3 "chaînes" sont disponibles :

- #général : c'est là que les questions liées à l'organisation générale du cours, des TP et TEA, des évaluations, etc. doivent être posées. Si vous ne savez pas si une séance de permanence a lieu, posez la question ici.
- #questions-rstudio : c'est ici que toutes les questions pratiques liées à l'utilisation de `R` et `RStudio` devront êtres posées. Problèmes de syntaxe, problèmes liés à l'interface, à l'installation des packages ou à l'utilisation des fonctions, à la création des graphiques, à la manipulation des tableaux… Tout ce qui concerne directement les logiciels sera traité ici. Vous êtes libres de poser des questions, de poster des captures d'écran, des morceaux de code, des messages d'erreur. Et **vous êtes bien entendus vivement encouragés à vous entraider et à répondre aux questions de vos collègues.** Je n'interviendrai ici que pour répondre aux questions laissées sans réponse ou si les réponses apportées sont inexactes. Le fonctionnement est celui d'un forum de discussion instantané. Vous en tirerez le plus grand bénéfice en participant et en n'ayant pas peur de poser des questions, même si elles vous paraissent idiotes. Rappelez-vous toujours que si vous vous posez une question, d'autres se la posent aussi probablement.
- #questions-stats : C'est ici que vous pourrez poser vos questions liées aux méthodes statistiques ou aux choix des modèles de dynamique des populations. Tout ce qui ne concerne pas directement l'utilisation du logiciel (comme par exemple le choix d'un test ou des hypothèses nulles et alternatives, la démarche d'analyse, la signification de tel paramètre ou estimateur, le principe de telle ou telle méthode…) peut être discuté ici. Comme pour le canal #questions-rstudio, **vous êtes encouragés à vous entraider et à répondre aux questions de vos collègues**.

Ainsi, quand vous travaillerez à vos TP ou TEA, que vous soyez installés chez vous ou en salle de TP, prenez l'habitude de garder Slack ouvert sur votre ordinateur. Même si vous n'avez pas de question à poser, votre participation active pour répondre à vos collègues est souhaitable et souhaitée. Je vous incite donc fortement à vous **entraider** : c'est très formateur pour celui qui explique, et celui qui rencontre une difficulté a plus de chances de comprendre si c'est quelqu'un d'autre qui lui explique plutôt que la personne qui a rédigé les instructions mal comprises.

Ce document est fait pour vous permettre d'avancer en autonomie et vous ne devriez normalement pas avoir beaucoup besoin de moi si votre lecture est attentive. L'expérience montre en effet que la plupart du temps, il suffit de lire correctement les paragraphes précédents et/ou suivants pour obtenir la réponse à ses questions. J'essaie néanmoins de rester disponible sur Slack pendant les séances de TP et de TEA de tous les groupes. Cela veut donc dire que **même si votre groupe n'est pas en TP, vos questions ont des chances d'être lues et de recevoir des réponses dès que d'autres groupes sont en TP ou TEA**. Vous êtes d'ailleurs encouragés à échanger sur Slack aussi pendant vos phases de travail personnel.

Progression conseillée

Si vous avez suivi le document de prise en main de `R` et `RStudio` lors de l'immersion ou en suivant l'EC de biométrie du semestre 3 de la licence SV de La Rochelle, vous savez que pour apprendre à utiliser ces logiciels, il faut faire les choses soi-même, ne pas avoir peur des messages d'erreurs (il faut d'ailleurs apprendre à les déchiffrer pour comprendre d'où viennent les problèmes), essayer maintes fois, se tromper beaucoup, recommencer, et surtout, ne pas se décourager. J'utilise ce logiciel presque quotidiennement depuis plus de 15 ans et à chaque session de travail, je rencontre des messages d'erreur. Avec suffisamment d'habitude, on apprend à les déchiffrer et on corrige les problèmes en quelques secondes. Ce livre est conçu pour vous faciliter la tâche, mais ne vous y trompez pas, vous rencontrerez des difficultés, et c'est normal. C'est le prix à payer pour profiter de la puissance du

meilleur logiciel permettant d'analyser des données, de produire des graphiques de qualité et de réaliser toutes les statistiques dont vous aurez besoin d'ici la fin de vos études et au-delà.

Pour que cet apprentissage soit le moins problématique possible, il convient de prendre les choses dans l'ordre. C'est la raison pour laquelle les chapitres de ce livre doivent être lus dans l'ordre, et les exercices d'application faits au fur et à mesure de la lecture.

Idéalement, voilà les étapes que vous devriez avoir franchi chaque semaine :

1. Les deux premières séances sont consacrées à la comparaison de la moyenne d'une population à une valeur théorique (Chapitre 1). Si vous êtes déjà capable d'importer et manipuler des données dans `R`, ainsi que de les explorer graphiquement et par le calcul d'indices de statistiques descriptives (indices de position, de dispersion et d'incertitude), les fonctions nouvelles seront peu nombreuses. Mais ce chapitre est l'occasion d'aborder des notions complexes (test paramétrique ou non, hypothèses nulles et alternatives, $p-$value, décision, erreurs, puissance, etc.) qui demanderont une lecture très attentive. C'est également la première fois que vous serez confronté à toutes les étapes d'une analyse de données : de l'importation des données dans `RStudio` jusqu'à la réalisation des tests statistiques et leur interprétation, en passant par le calcul des statistiques descriptives appropriées, la réalisation de graphiques exploratoires pertinents, et la vérification des conditions d'application du test. La maîtrise du logiciel n'est donc ici qu'une petite partie de ce qui est demandé : maîtriser les notions et concepts, comprendre la démarche d'analyse, être capable de l'expliquer et de la reproduire, est ici crucial. Évidemment, si vous ne disposez pas des bases essentielles à la compréhension des commandes de ce chapitre, vous devrez trouver le temps de (re)lire les livres en ligne du semestre 3 et du semestre 4. Avant votre troisième séance de TP, vous devriez ainsi avoir atteint la fin du chapitre 1, et vous devriez avoir terminé l'exercice d'application de la section Section 1.10.

2. Les deux séances suivantes sont consacrées à la comparaison de la moyenne de 2 populations, dans deux situations : lorsque les données des deux groupes sont appariées (Chapitre 2) et quand elles sont indépendantes (Chapitre 3). Ces chapitres seront également l'occasion d'aborder la notion de test unilatéral et bilatéral et d'apprendre comment coder des hypothèses alternatives spécifiques dans `RStudio`. Comme pour les semaines précédentes, vous aurez l'occasion de mettre en œuvre la totalité de la démarche d'analyse statistique, et dans chaque chapitre, un ou de exercices d'application vous sont également proposés.

Avant votre cinquième séance de TP, vous devriez donc être capable de choisir la procédure adaptée afin de comparer la moyenne d'une population à une valeur théorique, ou de comparer la moyenne de 2 populations, dans le cas où vous disposez d'échantillons appariés et dans le cas où les échantillons sont indépendants. Dans toutes ces situations, vous devrez être capable de vérifier les conditions d'application des tests paramétriques, et de choisir des tests non-paramétriques équivalents si les conditions d'application ne sont pas vérifiées. Vous devrez évidemment aussi être capables d'interpréter les résultats de ces tests. Et bien sûr, avant de faire chaque type de test, vous devrez aussi toujours penser au préalable à examiner les données graphiquement (à l'aide de graphiques adaptés), et par le biais des statistiques descriptives décrites au semestre 4 et rappelées plus haut (indices de position, de dispersion et d'incertitude).

Les 3 séances suivantes sont consacrées à la mise en pratique des notions vues dans le cours magistral de Population Dynamics (EC "Fonctionnement des Écosystèmes).

3. La cinquième séance est consacrée à l'analyse de cohortes (Chapitre 4). Avant votre sixième séance de TP, vous devriez être en mesure de réaliser l'analyse de cohorte d'une population étudiée pendant plusieurs années afin de produire les courbes de croissance, de survie et d'Allen d'une cohorte d'intérêt. Vous devrez en particulier importer et mettre en forme des données issues d'un suivi de terrain, produire les structures démographiques instantanées à chaque date d'échantillonnage, faire les décompositions polymodales afin de récupérer

les informations utiles au sujet de la cohorte dont on souhaite assurer le suivi et enfin, produire, pour cette cohorte, les courbes de croissance, de mortalité et d'Allen.

4. La sixième séance est consacrée à l'étude des systèmes dynamiques (Chapitre 5). Vous devrez coder différents modèles d'évolution d'une populations (mortalité exponentielle, croissance exponentielle, croissance logistique) ou de plusieurs populations ou espèces (modèle prédateurs-proies, modèle de compétition). Ces modèles généreront des données que vous devrez représenter graphiquement. Vous devrez enfin modifier la valeur de certains paramètres de ces modèles pour comprendre leur influence sur le comportement des systèmes dynamiques étudiés. Pour la première fois, vous utiliserez donc `RStudio` pour produire des données qui suivent un modèle que vous spécifierez, au lieu d'importer des données déjà disponibles. L'objectif est ici de vous montrer comment utiliser le logiciel pour étudier le comportement de modèles théoriques de dynamique des populations, et de vous permettre d'évaluer l'importance et la sensibilité des conditions initiales et des valeurs des paramètres des modèles.

5. La septième et dernière séance est consacrée à l'utilisation des chaînes de Markov dans un contexte d'écologie des communautés (Chapitre 6). Vous verrez comment utiliser `RStudio` pour coder une chaîne de Markov permettant d'étudier les successions écologiques et l'atteinte de l'état d'équilibre d'un écosystème abritant plusieurs communautés végétales. Comme pour la semaine précédente, vous devrez coder le système, représenter graphiquement les résultats de simulation, et examiner les conséquences de changements des conditions initiales ou des probabilités de transitions entre communautés. Comme toujours, l'objectif est donc double : vous permettre de découvrir de nouvelles possibilités d'analyses réalisables avec `R` et `RStudio`, et vous permettre de mieux comprendre les notions fondamentales abordées dans les cours magistraux.

Évaluation(s)

L'évaluation de la partie "Biométrie" de l'EC "Outils pour l'étude et la compréhension du vivant" aura lieu dans le cadre du travail de stratégie d'échantillonnage que vous mettez en œuvre avec Pierrick Bocher. Une évaluation **formative** servira à évaluer 3 choses :

- les grands principes de stratégie d'échantillonnage abordés par Pierrick Bocher
- la mise en œuvre de méthodes statistiques adaptées pour répondre aux questions scientifiques posées, telles que nous les traitons en Biométrie
- la maîtrise du logiciel `RStudio` pour réaliser les analyses de données pertinentes (de l'importation des données et leur mise en forme dans le logiciel, à la réalisation et l'interprétation correcte des tests statistiques appropriés, en passant par l'exploration des statistiques descriptives et la création de graphiques informatifs). Pour ce dernier volet, vous devrez rendre, en plus de votre compte-rendu, votre script d'analyse. C'est ce script qui me permettra d'évaluer votre niveau de compétence et de maîtrise de l'outil, tant sur la forme du script (lisibilité, structure, reproductibilité, etc.) que sur le fond (pertinence des analyses réalisées).

Pour vous aider à comprendre ce qui est attendu, je vous fournis ci-dessous la grille critériée dont je me servirai pour évaluer votre travail. C'est cette même grille qui sera utilisée au semestre 6 dans le cadre de l'évaluation sommative pour la biométrie. Je ne peux donc que vous encourager à lire attentivement les critères d'évaluation ci-dessous et à tenter de vous les approprier. Les séances de TP et de TEA qui viennent doivent vous permettre de vous entraîner à produire des scripts de qualité, et à y intégrer des analyses pertinentes et correctement interprétées.

Compétence visée : À l'issue de la licence, face à une question scientifique, vous devriez être capable de :
1. Identifier la ou les méthodes statistiques appropriées pour répondre à cette question
2. Mettre en œuvre ces méthodes dans RStudio
3. Interpréter correctement les résultats obtenus

Résultat d'apprentissage (RA) L'étudiant·e est capable de :	Très insuffisant	Insuffisant	Satisfaisant	Très satisfaisant	nº RA	Évalué par
Produire un script fonctionnel, qui fait ce qu'il est censé faire	Lorsqu'on exécute le script en une fois, de nombreux messages d'erreurs apparaissent, les résultats attendus ne sont pas produits ou sont faux	Lorsqu'on exécute le script en une fois, quelques messages d'erreurs apparaissent, tous les résultats attendus ne sont pas produits, ou certains résultats produits sont faux. La plupart des commandes utilisées pour manipuler ou explorer les données ne respectent pas les principes du tidyverse (par exemple, pas d'utilisation ou mauvaise utilisation des fonctions des packages dplyr, tidyr, ggplot2, etc.)	Lorsqu'on exécute le script en une fois, aucun message d'erreur n'apparait, mais certains résultats produits sont faux. La plupart des commandes utilisées pour manipuler ou explorer les données respectent les principes du tidyverse (bonne utilisation des fonctions des packages dplyr, tidyr, ggplot2, etc.)	Lorsqu'on exécute le script en une fois, aucun message d'erreur n'apparait, les résultats attendus sont correctement produits. Les commandes utilisées pour manipuler ou explorer les données respectent les principes du tidyverse et les conventions des livres en ligne de biométrie.	*1*	*BSB*
Fournir un script d'analyse correctement commenté	(Presque) pas de commentaires ou commentaires inadaptés	Peu de commentaires sont présents ou s'ils sont présents, la plupart ne permettent pas de comprendre clairement ce qui a été fait par (ou l'intention de) l'auteur du script	La plupart des commandes ou groupes de commandes sont commentés. Certains commentaires manquent parfois de clarté	Chaque commande ou groupe de commande est bien commenté. Les commentaires sont parlants et permettent de comprendre sans ambiguïté l'intention et les choix de l'auteur du script	*2*	*BSB*
Créer un script facile à lire et déchiffrer	Le script est difficile à déchiffrer en raison de plusieurs problèmes dans la liste suivante : noms d'objets peu parlants, espaces entre les éléments du code inconsistants ou absents, indentations inexistante, sauts de lignes manquants ou aux mauvais endroits, code non structuré, ordre des commandes inadéquat	Le script est globalement bien structuré, les commandes apparaissent dans un ordre logique, mais il pourrait être plus lisible car quelques problèmes subsistent dans la liste suivante : noms d'objets peu parlants, espaces entre les éléments du code inconsistants ou absents, indentations inexistante, sauts de lignes manquants ou aux mauvais endroits	Script bien structuré et lisible. La plupart des éléments de la liste suivante sont respectés : ordre des commandes adéquat, noms d'objets parlants, espaces, indentations et sauts de lignes consistants, bon équilibre entre commentaires et commandes R	Script parfaitement structuré et lisible. Tous les éléments de syntaxe sont respectés (espaces, ponctuation, indentation, sauts de lignes...), les noms d'objets sont courts et parlants, les commandes sont correctement ordonnées et un bon équilibre est respecté entre code et commentaires	*3*	*BSB*
Créer un script générique	Non. Si les données changent (ajout ou suppression de lignes dans le tableau de départ, changements de valeurs...), le script ne fonctionne plus (des messages d'erreurs apparaissent ou les résultats produits sont faux). Les commandes permettant d'importer les données ne sont pas fournies, ou si elles le sont, elles renseignent un chemin spécifique qui génère un message d'erreur sur un autre ordinateur que celui de l'auteur du script		Oui. Si les données changent (ajout ou suppression de lignes dans le tableau de départ, changements de valeurs...), le script fonctionne toujours : il ne renvoie pas de message d'erreur et les résultats fournis reflètent les modifications des données. Les commandes permettant d'importer les données sont fournies et permettent à tout utilisateur d'importer les données sans message d'erreur		*4*	*BSB*
Mettre en forme les données pour l'analyse (gestion des données manquantes, gestion des données aberrantes, modification d'unités, calcul de nouvelles variables composites, filtres, tris, etc.)	Les données sont dans un format qui ne permet pas les analyses statistiques correctes. Le tableau n'est pas "rangé", les formats tableaux longs et larges ne sont pas créés ou utilisés à mauvais escient	Les données sont dans un format qui permet les analyses statistiques (format long et/ou large selon les situations, tableau rangé), mais il contient encore plusieurs problèmes, par exemple : valeurs aberrantes non prises en compte, unités des variables incorrectes, types incorrect pour certaines variables (numériques, facteurs, etc.), pas de création de nouvelles variables pourtant indispensables, mauvaise gestion des données manquantes...	Les données sont dans un format qui permet les analyses statistiques (format long et/ou large selon les situations, tableau rangé), mais il contient encore un problème, par exemple : valeurs aberrantes non prises en compte, unités des variables incorrectes, types incorrect pour certaines variables (numériques, facteurs, etc.), pas de création de nouvelles variables pourtant indispensables, mauvaise gestion des données manquantes...	Les données sont dans un format qui permet les analyses statistiques (format long et/ou large selon les situations, tableau rangé), et tous les points suivants sont correctement traités : valeurs aberrantes prises en compte, unités des variables correctes, types des variables correct (numériques, facteurs, etc.), présence de toutes les nouvelles variables indispensables, bonne gestion des données manquantes...	*5*	*BSB*
D'explorer graphiquement les données avec des graphiques clairs et pertinents compte tenu de la nature des variables disponibles et de la question posée	Aucun graphique n'est produit (omission dans le rapport ou message d'erreur dans le script), ou les graphiques attendus ne sont pas produits. Si des graphiques sont fournis, leur type n'est pas adapté à la nature des variables disponibles ou à la question posée, ou la mise en forme inadaptée rend leur lecture et leur bonne interprétation impossible	Des graphiques sont produits (les commandes permettant de les produire fonctionnent normalement), mais ils ne sont pas idéaux : soit leur mise en forme n'est pas à la hauteur (mauvais choix de couleurs, de symboles, axes non ou incorrectement légendés, catégories impossibles à distinguer...), soit la nature des graphiques produits n'est pas idéale (diagrammes bâtons, histogrammes, nuages de points, boites à moustaches etc.)	Les graphiques produits sont adaptés à la nature des variables disponibles et aux questions posées et leur mise en forme est globalement bonne (p. ex. catégories faciles à distinguer, axes correctement légendés, unités appropriées...). Quelques oublis ou erreurs mineures peuvent subsister (fautes d'orthographe, symboles trop ressemblants, code peu clair), mais ils gênent peu la lecture et l'interprétation des graphiques produits	Les graphiques produits sont adaptés à la nature des variables disponibles et aux questions posées et leur mise en forme est parfaite. Les graphiques produits sont originaux, ils permettent une lecture et une interprétation aisée, en mettant en valeur, par exemple, les tendances principales, les comportements atypiques, les associations marquantes, etc.	*6*	*BSB et CL*
D'explorer les données à l'aide d'indices de statistiques descriptives pertinents compte tenu de la nature des variables disponibles et de la question posée	Aucun indice de statistiques descriptive n'est fourni, ou si des indices sont fournis, ils sont mal choisis ou calculés (p. ex. calcul d'indices tous groupes confondus alors qu'il aurait fallu calculer les indices pour chaque groupe). Les indices fournis ne permettent pas d'accéder à des informations pertinentes vis-à-vis des questions posées	Seuls quelques indices de statistiques descriptives sont fournis parmi la liste suivante : indices de position (p. ex. moyenne, médiane, quartiles), indices de dispersion (p.ex. variance, écart-type, intervalle inter-quartile) et indices d'incertitudes (p. ex. erreur standard, intervalle de confiance à 95%) et les plus pertinents sont manquants. Les indices fournis ne sont pas, ou pas suffisamment, décrits et commentés	Des indices de statistiques descriptives de position, dispersion et incertitude sont fournis, pour chaque variable utile, et à la bonne échelle d'observation (jeu de données global, groupes ou sous-groupes). La description de ces indices est claire et pertinente. Quelques rares indices peuvent être manquants, ou fournis à une échelle inadaptés, mais les plus importants sont bien présents	Tous les indices pertinents de statistiques descriptives de position, dispersion et incertitude sont fournis, à la bonne échelle d'observation (jeu de données global, groupes ou sous-groupes), et ils sont parfaitement présentés (dans des tables correctement formatées ou sur des graphiques adaptés), décrits et commentés	*7*	*BSB et CL*
De formuler clairement les hypothèses statistiques pertinentes dans le contexte de l'étude	Aucune hypothèse n'est posée, ou des hypothèses sont posées, mais ne sont pas en accord avec la problématique de l'étude ou sont contraires aux bases fondamentales des statistiques (par exemple, hypothèses nulles et alternatives inversées, ou hypothèses d'un test utilisées pour un autre tests inadapté)	Des hypothèses sont posées, mais manquent de clareté empêchant d'identifier clairement la variable considérée et/ou le facteur étudié	Les hypothèses sont clairement exposées, suivent les règles statistiques, mais ne sont pas toujours pertinentes (hypothèses unilatérales au lieu de bilatérales ou la contraire) et permettent de répondre partiellement à la question scientifique faisant l'objet de l'étude	Les hypothèses sont clairement exposées, elles sont pertinentes et suivent les règles statistiques. Elles permettent de répondre à la question scientifique faisant l'objet de l'étude	*8*	*BSB et CL*
De justifier le choix des méthodes statistiques utilisées pour répondre à la question posée (selon la nature des données, le respect ou non des conditions d'application, etc.)	Aucune explication n'est fournie, ou les justifications avancées ne sont pas fondées	Des éléments sont donnés pour justifier la logique de la démarche statistique, mais les méthodes employées (tests choisis notamment) ou l'ordre des étapes envisagées ne sont pas les bonnes	Les tests employés et l'ordre des étapes mises en œuvre sont adaptés, mais les justifications fournies ne sont pas suffisantes, ou pas suffisamment claires	Chaque test employé est clairement justifié et est adapté à la nature de la donnée. La démarche statistique employée est pertinente, ainsi que parfaitement décrite et justifiée	*9*	*BSB et CL*
De vérifier les conditions d'application des tests statistiques envisagés	Les conditions d'application ne sont pas prises en compte. Le principe fondamental des tests employés n'est pas compris. Si les conditions d'application sont énoncées, soit elles sont incorrectes, soit les tests employés pour les vérifier ne sont pas pertinents	Les conditions d'application sont énoncées et correctes, mais soit les tests employés ou les graphiques réalisés pour les vérifier sont inadaptés, soit ils sont adaptés mais pas correctement réalisés et/ou interprétés	Les conditions d'application sont énoncées et correctes, les tests employés ou les graphiques réalisés pour les vérifier sont adaptés et correctement réalisés, mais ils sont parfois mal interprétés.	Les conditions d'application sont énoncées et correctes, les tests employés ou les graphiques réalisés pour les vérifier sont adaptés et correctement réalisés et interprétés.	*10*	*BSB et CL*
De réaliser les tests statistiques choisis dans un ordre logique et cohérent	Les tests statistiques ne sont pas correctement réalisés, les valeurs calculées des paramètres (t, F…etc) sont erronées, ou si les valeurs calculées sont justes, l'ordre des étapes n'est pas le bon (p. ex. vérification des conditions d'application au mauvais moment, tests post-hoc réalisés avant l'ANOVA, tests non paramétriques réalisés à la place des tests paramétriques, ou le contraire...)	Les différents tests ne sont pas réalisés dans un ordre correct du point de vue statistique (par exemple, test de nomalité des données réalisé après un test de Student) ni du point de vue logique (p. ex. réalisation de tests post-hoc après une ANOVA même si l'hypothèse nulle de l'ANOVA n'a pas été rejetée, ou réalisation de tests non-paramétriques alors que les conditions d'application de leurs équivalents paramétriques sont pourtant vérifiées). La démarche / les différentes étapes ne sont pas clairement identifiables	Les différents tests sont réalisés dans un ordre correct du point de vue statistique (par exemple, test de nomalité des données réalisé avant un test de Student) mais pas forcément logique (p. ex. réalisation de tests post-hoc après une ANOVA même si l'hypothèse nulle de l'ANOVA n'a pas été rejetée, ou réalisation de tests non-paramétriques alors que les conditions d'application de leurs équivalents paramétriques sont pourtant vérifiées)	Les différents tests sont réalisés dans un ordre correct du point de vue statistique (par exemple, test de nomalité des données réalisé avant un test de Student) et logique (p. ex. réalisation de tests post-hoc après une ANOVA seulement si l'hypothèse nulle de l'ANOVA est rejetée, ou réalisation de tests non-paramétriques seulement lorsque les conditions d'application de leurs équivalents paramétriques ne sont pas vérifiées)	*11*	*BSB et CL*
D'interpréter correctement les résultats obtenus pour répondre à la question scientifique posée	Les p-values ou les valeurs critiques/seuils/théoriques sont incorrectes et les règles de décision ne sont pas correctement appliquées, engendrant systématiquement des conclusions erronées.	Les p-values ou les valeurs critiques/seuils/théoriques sont correctement déterminées, mais elles sont souvent mal utilisée. Il en découle que les décisions et conclusions sont souvent fausses. Les résultats de tests ne sont pas discutés au regard de la magnitude des effets détectés (p. ex. pas de calcul de moyennes, de différences de moyennes entre groupes, ni d'intervalles de confiance associés, etc.)	Les p-values ou les valeurs critiques/seuils/théoriques sont correctes, et bien utilisées la plupart du temps. Les règles de décision sont globalement comprises et correctement appliquées même s'il reste quelques rares erreurs. Il en découle que les décisions et les conclusions sont presque toujours correctes. La portée des résultats de tests est globalement bien discutée du point de vue de la magnitude des effets détectés, même si l'estimation des paramètres pertinents et de leurs incertitudes dans la population générale sont parfois manquantes (p. ex. moyennes ou différence de moyennes entre groupes et intervalles de confiance à 95% de ces différences, intensité et sens de la relation entre 2 variables numériques et incertitude associée, etc.)	Les p-values ou les valeurs critiques/seuils/théoriques sont correctement déterminée et utilisées. Les décision et les conclusions sont clairement énoncées et correctes. La portée des résultats de tests est discutée du point de vue de la magnitude des effets détectés, notamment grâce à l'estimation des paramètres pertinents et de leurs incertitudes dans la population générale (p. ex. moyennes ou différence de moyennes entre groupes et intervalles de confiance à 95% de ces différences, intensité et sens de la relation entre 2 variables numériques et incertitude associée, etc.)	*12*	*BSB et CL*

Licence

1 Comparaison de la moyenne d'une population à une valeur théorique

1.1 Pré-requis

Pour travailler dans de bonnes conditions, créez un nouveau dossier sur votre ordinateur, créez un `Rproject` et un script dans ce dossier, et travaillez systématiquement **dans votre script**, et surtout pas directement dans la console. Consultez le livre en ligne du semestre 3 si vous ne savez plus comment faire.

Dans ce chapitre, vous aurez besoin d'utiliser des packages spécifiques et d'importer des données depuis des fichiers externes téléchargeables directement depuis ce document. Les packages dont vous aurez besoin ici et que vous devez donc charger en mémoire, sont :

- le `tidyverse` (Wickham 2023), qui comprend notamment le package `readr` (Wickham, Hester, et Bryan 2024), pour importer facilement des fichiers `.csv` au format `tibble`, le package `dplyr` (Wickham et al. 2023), pour manipuler des tableaux, et le package `ggplot2` (Wickham et al. 2024) pour les représentations graphiques.
- `skimr` (Waring et al. 2022), qui permet de calculer des résumés de données très informatifs.

```
library(tidyverse)
library(skimr)
```

> **Important**
>
> Même si vous avez déjà installé le `tidyverse` ou `dplyr` au semestre précédent, ré-installez `dplyr` avec `install.packages("dplyr")`. Ce package a en effet été mis à jour récemment, et nous aurons d'une version récente (> v1.1.0). Chargez-le ensuite en mémoire avec `library(dplyr)`.

> **Attention**
>
> Pensez à installer tous les packages listés ci-dessous avant de les charger en mémoire si vous ne l'avez pas déjà fait. Si vous ne savez plus comment faire, consultez d'urgence la section dédiée aux packages dans le livre en ligne de Biométrie du semestre 3.

Vous aurez également besoin des jeux de données suivants :

- `Temperature.csv`
- `Temperature2.csv`

Enfin, je spécifie ici une fois pour toutes le thème que j'utiliserai pour tous les graphiques de ce chapitre. Libre à vous de choisir un thème différent ou de vous contenter du thème proposé par défaut :

```
theme_set(theme_bw())
```

1.2 Contexte

On s'intéresse ici à la température corporelle des adultes en bonne santé. On souhaite examiner la croyance populaire qui veut que cette température vaut en moyenne 37°C. Pour le vérifier, on dispose d'un échantillon de 25 adultes en bonne santé choisis au hasard parmi la population américaine et dont on a mesuré la température. Comme pour toute étude statistique, les étapes que nous allons devoir suivre sont les suivantes (dans l'ordre) :

1. Importer les données dans `RStudio`, les examiner et éventuellement les (re)mettre en forme si besoin.

2. Faire une première exploration des données, grâce au calcul d'indices de statistiques descriptives d'une part, et de représentations graphiques d'autre part.
3. Réaliser un test d'hypothèses en respectant la procédure adéquate (en particulier, la vérification des conditions d'application).

C'est donc ce que nous allons faire dans les sections suivantes.

> **!** À retenir !
>
> Avant de se lancer dans les tests d'hypothèses, il est **toujours indispensable** d'examiner les données dont on dispose à l'aide, d'une part de statistiques descriptives numériques, et d'autres part, de graphiques exploratoires.
> Nous avons vu au cours des semestres précédents quels indices statistiques il peut être utile de calculer (dans le livre en ligne du semestre 4) et quelles représentations graphiques il peut être utile de réaliser (dans le livre en ligne du semestre 3) afin de pouvoir se lancer dans des tests d'hypothèses sans risquer de grossières erreurs. N'hésitez pas à cliquer sur ces liens pour vous rafraîchir la mémoire !

1.3 Importation et mise en forme des données

Nous allons travailler ici sur les données contenues dans le fichier `Temperature.csv`. Téléchargez ces données dans votre répertoire de travail (attention : ne les ouvrez pas avec Excel !), puis importez les données dans `RStudio` grâce à l'assistant d'importation. Si vous ne savez plus comment faire, consultez la section dédiée à l'importation des données dans le livre en ligne de Biométrie du semestre 3

Vous stockerez les données dans un objet que vous nommerez `Temperature`. Après l'importation, tapez son nom dans la console de `RStudio` et vérifiez que vous obtenez bien exactement ce résultat :

```
Temperature
```

```
# A tibble: 25 x 2
   individual temperature
        <dbl>       <dbl>
 1          1        98.4
 2          2        98.6
 3          3        97.8
 4          4        98.8
 5          5        97.9
 6          6        99
 7          7        98.2
 8          8        98.8
 9          9        98.8
10         10        99
# i 15 more rows
```

La première chose à faire quand on travaille avec des données inconnues, c'est d'examiner les données brutes. Ici, les données sont importées au format `tibble`, donc seules les premières lignes sont visibles. Pour visualiser l'ensemble du tableau, utilisez la fonction `View()` (avec un `V` majuscule) ou, si vous avez mis en mémoire le `tidyverse`, la fonction `view()` (sans majuscule) :

```
View(Temperature)
```

Cette commande ouvre un nouvel onglet présentant les données dans un tableur simplifié, en lecture seule. On constate ici 2 choses que nous allons modifier :

1. la première colonne, intitulée `individual`, n'est pas véritablement une variable. Cette colonne ne contient qu'un identifiant sans intérêt pour notre étude et est en fait identique au numéro de ligne. Nous allons donc supprimer cette colonne.
2. les températures sont exprimées en degrés Fahrenheit, ce qui rend leur lecture difficile pour nous qui sommes habitués à utiliser le système métrique et les degrés Celsius. Nous allons donc convertir les températures en degrés Celsius grâce à la formule suivante :

$$C = \frac{F - 32}{1.8}$$

```
Temp_clean <- Temperature |>
  select(-individual) |>      # Suppression de la colonne `individual`
  mutate(                      # Transformation des températures en ºCelsius
    temperature = (temperature - 32) / 1.8
    )

Temp_clean
```

```
# A tibble: 25 x 1
   temperature
         <dbl>
 1        36.9
 2        37
 3        36.6
 4        37.1
 5        36.6
 6        37.2
 7        36.8
 8        37.1
 9        37.1
10        37.2
# i 15 more rows
```

Il nous est maintenant possible d'examiner à nouveau les données avec la fonction `View()`. Avec des valeurs de températures comprises entre 36.3ºC et 37.8ºC, il n'y a visiblement pas de données aberrantes.

Examiner les données brutes est donc la première chose que vous devriez prendre l'habitude de faire, et ce de façon systématique, car cela permet de repérer :

- La nature des variables présentes.
- Les variables inutiles qui pourront être supprimées ou négligées.
- Les unités des variables utiles, afin de pouvoir les convertir si nécessaire.
- Les valeurs manquantes, atypiques ou aberrantes qui demanderont toujours une attention particulière.

Maintenant que l'examen préliminaire des données est réalisé, on peut passer au calcul des statistiques descriptives.

1.4 Exploration statistique des données

1.4.1 Position et dispersion

On s'intéresse ici au calcul de grandeurs statistiques nous apportant des renseignements sur la **position** et la **dispersion** des valeurs de l'échantillon. Les questions auxquelles on tente de répondre à ce stade sont les suivantes :

- Quelle est la tendance centrale (moyenne ou médiane) ?
- Quelle est la dispersion des valeurs autour de la tendance centrale (écart-type, variance, intervalle interquartile…) ?

Pour répondre à ces questions, on peut faire appel à de multiples fonctions déjà présentées dans le livre en ligne du semestre 4. Par exemple la fonction `summarise()`, en conjonction avec les fonctions `mean()`, `median()`, `sd()`, `var()`, `min()`, `max()` ou `quantile()`, ou les fonctions `summary()` ou `skim()` (du package `skimr`).

Je prends ici un exemple simple, mais n'hésitez pas à expérimenter avec les méthodes décrites dans le livre en ligne du semestre 4.

```
summary(Temp_clean)
```

```
  temperature
 Min.   :36.33
 1st Qu.:36.67
 Median :37.00
 Mean   :36.96
 3rd Qu.:37.22
 Max.   :37.78
```

On constate ici que la moyenne et la médiane sont très proches. La distribution des températures doit donc être à peut près symétrique, avec à peu près autant de valeurs

au-dessus que de valeurs en dessous de la moyenne. Les premier et troisième quartiles sont à peu près aussi éloignés de la médiane l'un que l'autre, ce qui confirme l'apparente symétrie du jeu de données de part et d'autre de la tendance centrale.

La moyenne observée dans l'échantillon vaut 36.96°C, ce qui est très proche de la moyenne théorique de 37°C.

Une autre fonction utile est la fonction `IQR()`, qui renvoie l'étendue de l'intervalle interquartile (la valeur du troisième quartile moins la valeur de premier quartile) :

```
Temp_clean |>
  summarise(IQ_range = IQR(temperature))
```

```
# A tibble: 1 x 1
  IQ_range
     <dbl>
1    0.556
```

On constate ici que l'intervalle interquartile a une largeur de 0.56°C. Cela signifie que les 50% des températures les plus centrales de l'échantillon sont situées dans un intervalle d'environ un demi-degré Celsius autour de la médiane.

Enfin, pour obtenir des informations complémentaires, on peut utiliser la fonction `skim()` du package `skimr` :

```
skim(Temp_clean)
```

```
-- Data Summary ------------------------
                           Values
Name                       Temp_clean
Number of rows             25
Number of columns          1
-----------------------
Column type frequency:
  numeric                  1
-----------------------
Group variables            None

-- Variable type: numeric --------------------------------------------------------
```

```
  skim_variable n_missing complete_rate mean    sd   p0  p25 p50  p75 p100 hist
1 temperature           0             1 37.0 0.377 36.3 36.7  37 37.2 37.8
```

Tout comme `summary()`, la fonction `skim()` renvoie les valeurs minimales et maximales, les premiers et troisièmes quartiles ainsi que la moyenne et la médiane. Elle nous indique en outre la valeur de l'écart-type de l'échantillon, ainsi que le nombre d'observations et le nombre de données manquantes. Enfin, elle fournit un histogramme très simplifié et sans échelle. Cet histogramme nous permet de nous faire une première idée de la distribution des données et est particulièrement utile pour comparer rapidement un grand nombre de distributions quand il y a plusieurs catégories dans les données (ce qui n'est pas le cas ici).

Outre ces 3 fonctions (`summary()`, `IQR()`, et `skim()`), il est bien sûr possible de calculer toutes ces valeurs manuellement si besoin :

- `mean()` permet de calculer la moyenne.
- `median()` permet de calculer la médiane.
- `min()` et `max()` permettent de calculer les valeurs minimales et maximales respectivement.
- `quantile()` permet de calculer les quartiles.
- `sd()` permet de calculer l'écart-type.
- `var()` permet de calculer la variance.
- `n()` permet de compter le nombre d'observations.

Toutes ces fonctions prennent seulement un vecteur en guise d'argument. Il faut donc procéder comme avec `IQR()` pour les utiliser, en les intégrant à l'intérieur de la fonction `summarise()`. Par exemple, pour calculer la variance, on peut taper :

```
Temp_clean |>
  summarise(variance = var(temperature))
```

```
# A tibble: 1 x 1
  variance
     <dbl>
1    0.142
```

ou :

```
Temp_clean |>
  pull(temperature) |>
  var()
```

```
[1] 0.1417901
```

ou encore :

```
var(Temp_clean$temperature)
```

```
[1] 0.1417901
```

À vous d'utiliser la syntaxe qui vous semble la plus simple.

1.4.2 Incertitude

Outre les informations de position et de dispersion, nous avons vu au semestre 4 qu'il était également important d'avoir une idée de l'**incertitude** associée aux estimations de tendance centrale (erreur standard ou intervalle de confiance de la moyenne ou médiane). Ici, nous allons donc calculer l'intervalle de confiance à 95% de la moyenne. Si vous ne savez plus comment faire, ou que vous ne comprenez pas le code ci-dessous, consultez le livre en ligne du semestre 4 :

```
Temp_clean |>
  reframe(mean_cl_normal(temperature))
```

```
# A tibble: 1 x 3
      y  ymin  ymax
  <dbl> <dbl> <dbl>
1  37.0  36.8  37.1
```

On constate ici que les bornes inférieure (36.8°C) et supérieure (37.1°C) de l'intervalle de confiance à 95% de la moyenne sont proches de la valeur de moyenne de l'échantillon. Dans la population générale, la moyenne de la température corporelle chez les adultes en bonne santé a

de bonnes chances de se trouver quelque part entre 36.8ºC et 37.1ºC. Autrement dit, si la température corporelle des adultes en bonne santé n'est pas exactement de 37ºC, l'écart à cette valeur théorique ne doit pas être très important.

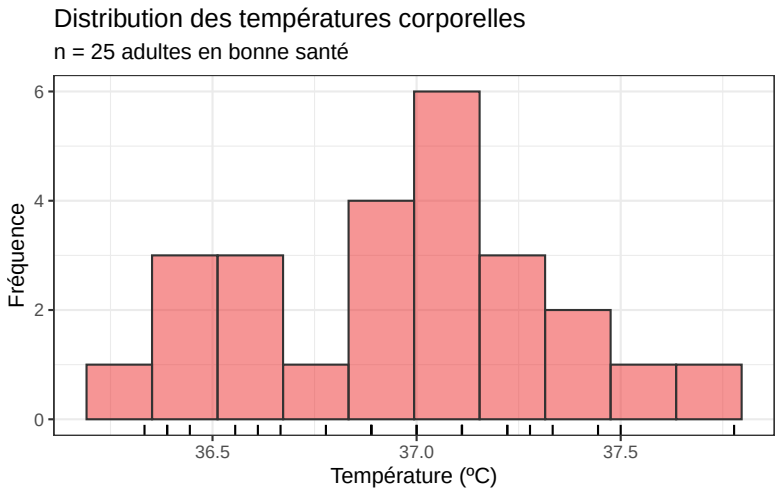
1.5 Exploration graphique des données

Ici, puisque nous ne disposons que d'une unique variable numérique et que nous n'avons donc qu'un unique groupe, les représentations graphiques que nous allons réaliser doivent nous permettre d'examiner la **distribution des données**. Pour cela, nous pouvons réaliser soit un histogramme, soit un diagramme de densité.

1.5.1 Histogramme

Voilà comment produire un histogramme de qualité pour les données de températures :

```
Temp_clean |>
  ggplot(aes(x = temperature)) +
  geom_histogram(bins = 10, fill = "firebrick2", color = "grey20",
                 alpha = 0.5) +
  geom_rug() +
  labs(x = "Température (ºC)",
       y = "Fréquence",
       title = "Distribution des températures corporelles",
       subtitle = "n = 25 adultes en bonne santé")
```



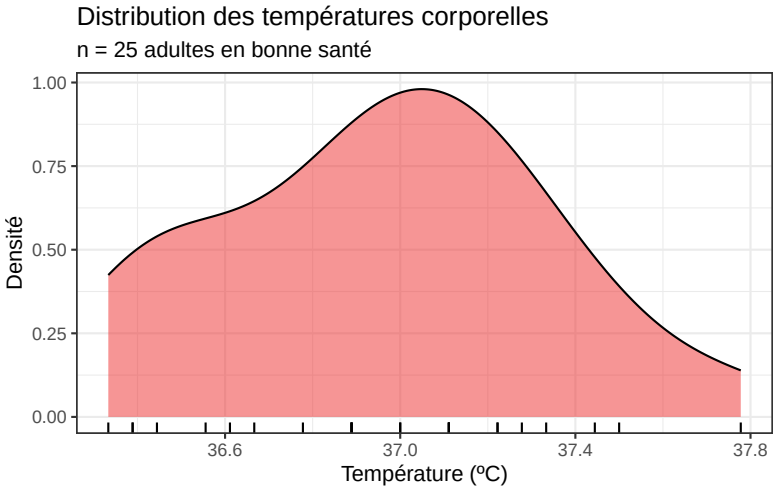
Si vous ne vous rappelez-plus ce qu'est un histogramme ou comment le faire, ou la signification de l'argument `bins`, relisez la section consacrée aux histogrammes du livre en ligne de Biométrie du semestre 3. Notez que j'ai ajouté une couleur de remplissage et de la transparence pour rendre le graphique plus facile à lire. J'ai également spécifié des titres pour les axes (en précisant l'unité de la variable numérique dont on représente la distribution) ainsi que le titre (et sous-titre) du graphique, qui précise ce qu'on a sous les yeux et la taille de l'échantillon. Il n'est pas toujours nécessaire de spécifier le titre (et le sous-titre) de cette façon : lorsque vous intégrez des graphiques dans un compte-rendu ou un rapport, le titre est en général précisé sous la figure, au début d'une légende qui la décrit. Enfin, j'ai ajouté `geom_rug()` pour faire apparaître sous le graphique, le long de l'axe des `x`, la position des données observées. Cela permet de visualiser les données brutes, et peut donc permettre de mieux comprendre pourquoi un histogramme présente telle ou telle forme.

Ici, la forme de ce l'histogramme est assez proche de celle présentée plus tôt par l'histogramme très simplifié produit par la fonction `skim()`. Cet histogramme nous apprend qu'en dehors d'un "trou" autour de la température 36.75ºC, la distribution des données est proche d'une courbe en cloche. Il y a fort à parier qu'un test de normalité conclurait à la normalité des données de cet échantillon. C'est ce que nous verrons dans la Section 1.6.1.

1.5.2 Diagramme de densité

Une autre façon de visualiser la distribution d'une variable numérique est de produire un graphique de densité. Il a l'avantage d'éviter à l'utilisateur d'avoir à choisir une valeur pour l'argument `bin` de la fonction `geom_histogram()`, mais il a l'inconvénient de présenter une échelle plus difficile à comprendre pour l'axe des ordonnées :

```
Temp_clean |>
  ggplot(aes(x = temperature)) +
  geom_density(fill = "firebrick2", alpha = 0.5) +
  geom_rug() +
  labs(x = "Température (ºC)",
       y = "Densité",
       title = "Distribution des températures corporelles",
       subtitle = "n = 25 adultes en bonne santé")
```



Les informations apportées par ce graphique sont cohérentes avec celle de l'histogramme :

- les températures les plus fréquemment observées dans notre échantillon de 25 adultes en bonne santé se situent légèrement au dessus de 37ºC. Il s'agit d'une information concernant la **position** des données (c'est-à-dire où se trouve le pic de la distribution sur l'axe des `x`)
- les températures observées ont une distribution qui ressemble à peu près à une courbe en cloche, avec des valeurs comprises entre 36.4ºC et 37.8ºC environ. La

symétrie de part et d'autre du pic n'est pas parfaite, mais elle reste bonne. Il s'agit d'informations concernant la forme de la distribution et la **dispersion** des données.

i Bilan des analyses préliminaires

Suite à l'exploration statistique et graphique des données de températures, voilà ce qu'on retient :

1. Il n'y a visiblement pas de données aberrantes.
2. La distribution des données semble suivre à peu près la loi Normale.
3. La médiane et la moyenne sont très proches de 37°C. Un test devrait donc arriver à la conclusion que la température corporelle des adultes n'est pas significativement différente de 37°C.
4. La largeur de l'intervalle de confiance à 95% semble faible, ce qui indique une incertitude relativement faible. Si la température réelle des adultes en bonne santé n'est pas exactement de 37°C, elle ne devrait pas en être très éloignée (quelques dixièmes de degrés Celsuis au plus).

1.6 Le test paramétrique

Le test permettant de comparer la moyenne μ d'une population à une valeur théorique, fixée par l'utilisateur, est le **test de Student à un échantillon**. Il permet de répondre à la question suivante :

> Les données observés dans l'échantillon dont je dispose sont-elles compatibles avec l'hypothèse que la moyenne μ de la population dont est issu mon échantillon vaut `XXX` ?

avec `XXX`, une valeur d'intérêt spécifiée par l'utilisateur. Il s'agit d'un test paramétrique très puissant. Comme tous les tests paramétriques, certaines conditions d'application doivent être vérifiées avant de pouvoir l'appliquer.

> **!** Important
>
> Comme pour tous les tests statistiques que nous allons réaliser lors de ces séances de TP et TEA, nous devrons commencer par **spécifier les hypothèses** nulles et alternatives de chaque test, ainsi que la **valeur du seuil** α que nous allons utiliser. À moins d'avoir une bonne raison de faire autrement, on utilise presque toujours le seuil $\alpha = 0.05$ dans le domaine des sciences du vivant. C'est donc ce seuil que nous utiliserons dans ce livre en ligne.

1.6.1 Conditions d'application

Les conditions d'application du test de Student à un échantillon sont les suivantes :

1. Les données de l'échantillon sont issues d'un **échantillonnage aléatoire** au sein de la population générale. Cette condition est partagée par toutes les méthodes que nous verrons dans ces TP. En l'absence d'informations sur la façon dont l'échantillonnage a été réalisé, on considère que cette condition est remplie. Il n'y a pas de moyen statistique de le vérifier, cela fait uniquement référence à la stratégie d'échantillonnage déployée et à la rigueur de la procédure mise en œuvre lors de l'acquisition des données.
2. La variable étudiée doit suivre une **distribution Normale** dans la population générale. Nous allons vérifier cette condition d'application avec un **test de normalité de Shapiro-Wilk**.

Pour un test de normalité, les hypothèses seront toujours les suivantes :

- H_0 : la variable étudiée suit une distribution Normale dans la population générale.
- H_1 : la variable étudiée ne suit pas une distribution Normale dans la population générale.

Le test de Shapiro-Wilk se réalise de la façon suivante :

```r
shapiro.test(Temp_clean$temperature)
```

ou

```r
Temp_clean |>
  pull(temperature) |>
  shapiro.test()
```

```
    Shapiro-Wilk normality test

data:  pull(Temp_clean, temperature)
W = 0.97216, p-value = 0.7001
```

la fonction `pull()` permet d'extraire une colonne (ici `temperature`) d'un tibble (ici `Temp_clean`) et de la transformer en vecteur.

`W` est la statistique du test. Elle permet à RStudio de calculer la p-value. Ici, $p > \alpha$. On ne peut donc pas rejeter l'hypothèse nulle de normalité : on ne peut pas exclure que dans la population générale, la température suive bel et bien une distribution Normale. Les conditions d'application du test de Student sont bien vérifiées.

> ⚠ Tests et décision : rappel de cours
>
> À l'issue d'un tests statistique, la décision finale est toujours prise par rapport à l'hypothèse nulle (H_0) :
>
> - Si la $p-$value du test est supérieure ou égale à α, on dit qu'on ne peut pas rejeter l'hypothèse nulle H_0. Attention, on ne dit jamais que "H_0 est vraie", car il est impossible de le vérifier avec une certitude absolue. Toutefois, les données observées (celles de notre échantillon), sont compatibles avec l'hypothèse nulle que nous avons formulée, jusqu'à preuve du contraire.
> - Si la $p-$value du test est inférieure à α, on dit qu'on rejette l'hypothèse nulle au seuil α. Autrement dit, les données observées ne sont pas compatibles avec l'hypothèse nulle. On accepte alors l'hypothèse alternative (H_A).

L'hypothèse nulle est toujours l'hypothèse la moins "intéressante", celle pour laquelle "il ne se passe rien de notable" (par exemple : "les données suivent la distribution Normale", ou "les moyennes sont égales").

1.6.2 Signification de la $p-$value

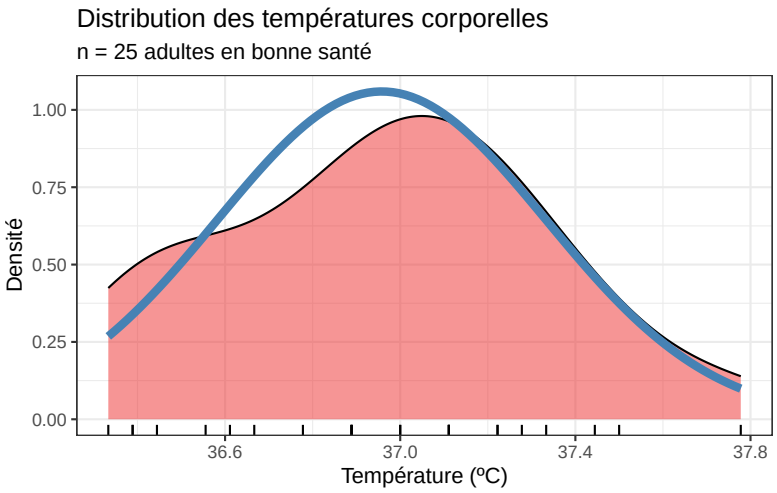
La $p-$value est une grandeur centrale en statistiques et elle est souvent mal comprise et donc mal interprétée. Je prends donc le temps ici d'expliquer ce qu'est la $p-$value et comment il faut la comprendre.

Définition : la $p-$value

La $p-$value d'un test statistique, c'est la probabilité, si H_0 est vraie, d'obtenir un effet au moins aussi extrême que celui qu'on a observé dans l'échantillon, sous le seul effet du hasard.

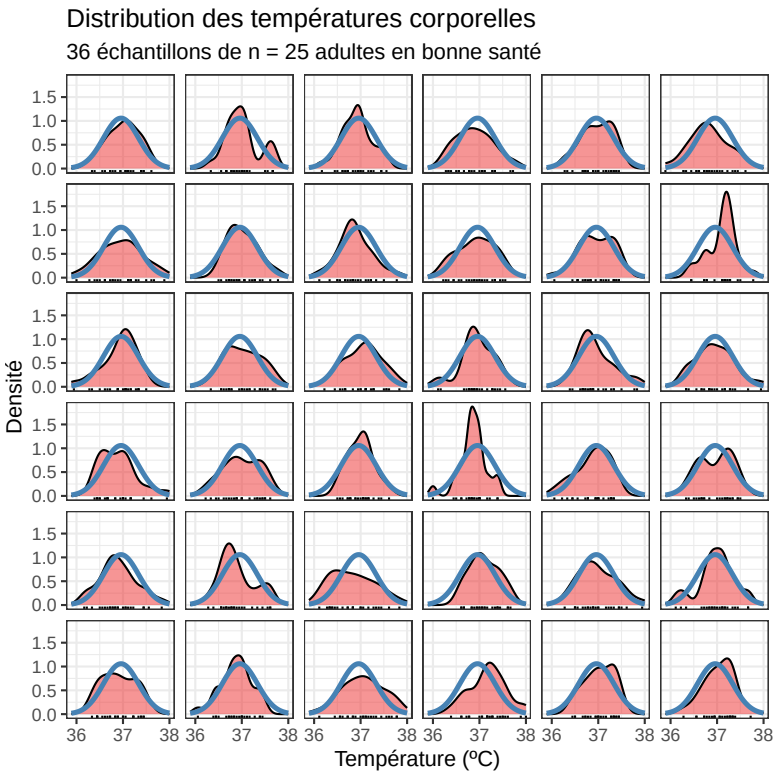
Ici, la $p-$value de notre test de Normalité de Shapiro-Wilk vaut 0.7101. Cela signifie que si les données suivent réellement la loi Normale dans la population générale (donc si H_0 est vraie), l'écart à la Normalité que nous avons observé (ou un écart encore plus important), peut être observé dans 70.1% des cas. Autrement dit, si on prélève un grand nombre d'échantillons de 25 adultes dans la population générale et qu'on regarde à quoi ressemble la distribution des températures dans chacun de ces échantillons, pour 70.1% d'entre eux, la distribution obtenue sera au moins aussi éloignée de la distribution Normale que celle que nous avons observée ici.

Dans notre cas, l'écart entre la loi Normale et les données de notre échantillon peut être visualisé de la façon suivante :

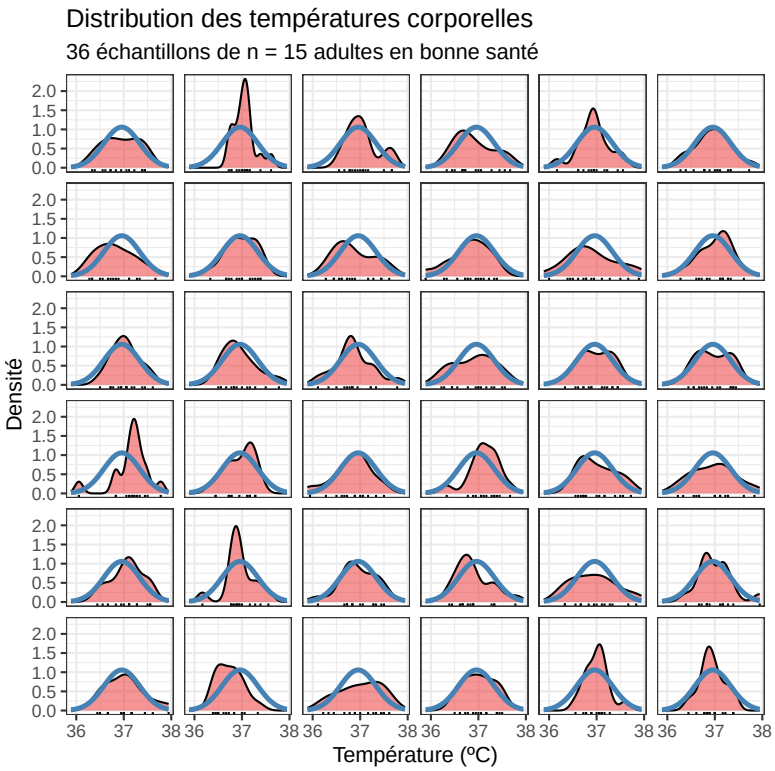


La courbe de densité des données observées est en rouge, et la distribution Normale théorique correspond à la courbe en bleu. Il y a donc un écart entre la courbe en cloche parfaite de la loi Normale et les données observées. La $p-$value du test de Shapiro-Wilk nous dit que si la température des adultes en bonne santé suit réellement la loi Normale dans la population générale, alors, l'écart que nous avons observé, ou un écart encore plus important, peut être observé simplement par hasard dans 70.1% des cas. Autrement dit, c'est très probable, et on peut donc considérer que l'écart à la loi Normale que nous avons observé est le fruit du hasard et que notre variable suit donc bien la Loi Normale.

Pour bien comprendre cette notion importante, je simule ci-dessous 36 échantillons de 25 adultes dont les températures suivent parfaitement la loi Normale dans la population générale. Je me place donc dans la situation ou je sais que H_0 est vraie, pour illustrer la notion de *fluctuation d'échantillonnage*. En raison du seul hasard de l'échantillonnage, et alors même que les échantillons que je génère sont issus d'une population qui suit parfaitement la Normale, la distribution dans chaque échantillon s'écarte parfois fortement de la courbe en cloche théorique :



On voit bien ici que certains échantillons s'écartent fortement de la distribution théorique alors même que tous les échantillons sont issus d'une population Normale. Et plus l'échantillon sera de taille réduite, plus les écarts à la courbe en cloche parfaite seront grands. La preuve ci-dessous avec des échantillons de n = 15 adultes au lieu de 25 :



Au final, la $p-$value de 0.701 de notre test de Shapiro-Wilk nous indique que l'hypothèse de la Normalité n'est pas incompatible avec les données que nous avons observées.

Imaginons qu'à l'inverse, nous ayons obtenu une $p-$value très faible, égale à 0.01 par exemple (donc inférieure à notre seuil α de 0.05). Nous aurions alors rejeté l'hypothèse nulle. En effet, obtenir une $p-$value de 0.01, signifie que si H_0 est vraie, obtenir un écart à la courbe en cloche théorique aussi important que celui que nous observons est très peu probable (une chance sur 100). Puisqu'il est très improbable d'observer un tel écart si H_0 est vraie, on en conclu que H_0 n'est pas vraie : les données sont incompatibles avec l'hypothèse nulle et on la rejette donc logiquement.

Cette logique sera valable pour tous les autres tests statistiques que nous aborderons dans cet ouvrage. Pour un test de Normalité, on regarde l'écart entre la distribution Normale et les données observées. Pour un test de comparaison de moyennes, on regarde l'écart entre la moyenne théorique et la moyenne observée, ou entre les 2 moyennes qu'on essaie de comparer. Mais la philosophie reste la même.

1.6.3 Réalisation du test de Student et interprétation

Puisque les conditions d'application du test de Student à un échantillon sont vérifiées, nous avons le droit de faire ce test, et nous devons donc maintenant spécifier les hypothèses nulles et alternatives que nous allons utiliser pour le réaliser :

- H_0 : dans la population générale, la température corporelle moyenne des adultes en bonne santé vaut 37°C ($\mu = 37$).
- H_1 : dans la population générale, la température corporelle moyenne des adultes en bonne santé est différente de 37°C ($\mu \neq 37$).

> ⚠ Hypothèses et paramètres
>
> Notez que les hypothèses des tests statistiques concernent toujours la valeur d'un paramètre de la population générale, et non la valeur des estimateurs calculés dans un échantillon.

On réalise ensuite le test de la façon suivante :

```
t.test(Temp_clean$temperature, mu = 37)
```

ou

```
t.test(temperature ~ 1, mu = 37, data = Temp_clean)
```

ou encore,

```
Temp_clean |>
  pull(temperature) |>
  t.test(mu = 37)
```

```
    One Sample t-test

data:  pull(Temp_clean, temperature)
t = -0.56065, df = 24, p-value = 0.5802
alternative hypothesis: true mean is not equal to 37
```

```
95 percent confidence interval:
 36.80235 37.11321
sample estimates:
mean of x
 36.95778
```

Les résultats fournis ont une forme particulière qui est utilisée par de nombreuses fonctions de tests statistiques dans `R`. Ils méritent donc qu'on s'y attarde un peu.
Sur la première ligne, `R` nous confirme que nous avons bien réalisé un test de Student à un échantillon. La première ligne de résultats fournit la valeur du t calculé (ici, -0.56), le nombre de degrés de libertés (ici, `df` $= 24$), et la $p-$value (ici, 0.58, soit une valeur supérieure à α). Cette première ligne contient donc tous les résultats du test qu'il conviendrait de rappeler dans un rapport. On devrait ainsi dire :

> Au seuil α de 5%, le test de Student ne permet pas rejeter l'hypothèse nulle $\mu = 37$ ($t = -0.56$, ddl $= 24$, $p = 0.58$). Les données observées sont donc compatibles avec l'hypothèse selon laquelle la température corporelle moyenne des adultes en bonne santé vaut 37ºC.

C'est de cette manière que vous devriez rapporter les résultats de ce test dans un compte-rendu ou un rapport à partir de maintenant.

Dans les résultats du test, la ligne suivante (`alternative hypothesis: ...`) **ne donne pas la conclusion du test**. Il s'agit simplement d'un rappel concernant l'hypothèse alternative qui a été utilisée pour réaliser le test. Ici, l'hypothèse alternative utilisée est une hypothèse bilatérale ($\mu \neq 37$). Nous verrons plus tard comment spécifier des hypothèses alternatives uni-latérales, même si la plupart du temps, mieux vaut s'abstenir de réaliser de tels tests (à moins bien sûr d'avoir une bonne raison de le faire).

Les résultats fournis ensuite concernent, non plus le test statistique à proprement parler, mais l'estimation. Ici, la moyenne de l'échantillon est fournie. Il s'agit de la meilleure estimation possible de la moyenne de la population : $\bar{x} = \hat{\mu} = 36.96$. Comme pour toutes les estimations, cette valeur est entachée d'incertitude liée à la fluctuation d'échantillonnage. L'intervalle de confiance à 95% de

cette estimation de moyenne est donc également fourni : [36.80; 37.11]. Vous notez qu'il s'agit des mêmes valeurs que celles que nous avions calculées dans la Section 1.4.2. Autrement dit, cet intervalle contient les valeurs les plus vraisemblables pour la véritable valeur de moyenne dans la population générale. Cela confirme bien que nous n'avons pas prouvé au sens strict que la moyenne de la population vaut 37°C. Nous avons en réalité montré que nous ne pouvions pas exclure que la moyenne de la population générale soit de 37°C. Puisque cette valeur est comprise dans l'intervalle de confiance, on ne peut donc pas l'exclure : nos données sont compatibles avec cette hypothèse. Mais beaucoup d'autres valeurs figurent aussi dans cet intervalle. Il est donc tout à fait possible que la moyenne soit en réalité différente de 37°C (par exemple, 36.9°C). Pour en être sûr, il faudrait probablement un échantillon de plus grande taille afin de limiter l'incertitude, d'augmenter la puissance statistique de notre test, et ainsi d'être en mesure de détecter des différences subtiles.

1.7 L'alternative non paramétrique

Si jamais les conditions d'application du test de Student à un échantillon n'étaient pas remplies, il faudrait alors réaliser son équivalent non paramétrique : le **test de Wilcoxon des rangs signés**. Ce test est moins puissant que son homologue paramétrique. On ne l'effectue donc que lorsque l'on n'a pas le choix :

```
wilcox.test(Temp_clean$temperature, mu = 37, conf.int = TRUE)
```

ou

```
wilcox.test(temperature ~ 1, mu = 37, conf.int = TRUE, data = Temp_clean)
```

ou encore

```
Temp_clean |>
  pull(temperature) |>
  wilcox.test(mu = 37, conf.int = TRUE)
```

```
Warning in wilcox.test.default(pull(Temp_clean, temperature), mu = 37, conf.int
= TRUE): impossible de calculer la p-value exacte avec des ex-aequos


Warning in wilcox.test.default(pull(Temp_clean, temperature), mu = 37, conf.int
= TRUE): impossible de calculer un intervalle de confiance exact avec des
ex-aequos



    Wilcoxon signed rank test with continuity correction

data:  pull(Temp_clean, temperature)
V = 143, p-value = 0.6077
alternative hypothesis: true location is not equal to 37
95 percent confidence interval:
 36.77780 37.11114
sample estimates:
(pseudo)median
      36.94446
```

La syntaxe est identique à celle du test de Student à un échantillon à une exception près : l'ajout de l'argument `conf.int = TRUE` qui permet d'afficher la (pseudo)médiane de l'échantillon et son intervalle de confiance à 95%.

Les hypothèses nulles et alternatives de ce test sont les mêmes que celles du test de Student à un échantillon. En toute rigueur, on compare la médiane à une valeur théorique, et non la moyenne. Mais dans la pratique, la grande majorité des utilisateurs de ce test font l'amalgame entre moyenne et médiane. Ici, la conclusion correcte devrait donc être :

> Au seuil α de 5%, on ne peut pas rejeter l'hypothèse nulle (test de Wilcoxon des rangs signés, $V = 143$, $p = 0.6077$). La médiane de la population ($\widehat{med} = 36.94$) n'est pas significativement différente de 37ºC (IC 95% : $[36.78; 37.11]$).

Si les données ne suivent pas la loi Normale, la médiane est bien la métrique la plus intéressante puisque c'est elle qui nous renseigne sur la tendance centrale des données.

Enfin, les tests de Wilcoxon renvoient souvent des messages d'avertissement. Il ne s'agit que de ça : des avertissements.

Tant que la p-value d'un test est éloignée de la valeur seuil α, cela n'a pas d'importance. Quand en revanche la p-value est très proche de α, les messages d'avertissement doivent vous alerter : il faut être très prudent face aux conclusions du test qui peuvent alors être assez "fragiles".

1.8 Les notions d'erreur et de puissance statistique

Pour avoir le droit de réaliser un test paramétrique, il faut au préalable vérifier qu'un certain nombre de conditions sont vérifiées. Si ce n'est pas le cas, on réalise un équivalent non paramétrique. On peut alors se demander pourquoi ne pas se contenter de faire des tests non paramétrique systématiquement, sans s'embêter à faire des tests supplémentaires ou des tests paramétriques.

La raison est simple et elle est liée aux notions d'erreur et de puissance statistique.

Définitions

- **Erreur de type I** : notée α, c'est la probabilité de rejeter à tort l'hypothèse nulle. C'est donc la probabilité de rejeter H_0 alors qu'elle est vraie.
- **Erreur de type II** : notée β, c'est la probabilité d'accepter à tort l'hypothèse nulle. C'est donc la probabilité d'accepter H_0 alors qu'elle est fausse.
- **Puissance statistique** : notée 1 - β), c'est la probabilité de rejeter l'hypothèse nulle à raison. C'est donc la probabilité de rejeter H_0 quand elle est réellement fausse.

À chaque fois que l'on réalise un test statistique, on commet nécessairement les 2 types d'erreurs α et β. On souhaite évidemment minimiser les erreurs, mais on ne peut malheureusement pas faire baisser les 2 en même temps. Faire baisser α (pour diminuer les faux positifs) conduit toujours à augmenter β (les faux négatifs). Faire baisser α revient en effet à accepter plus souvent l'hypothèse nulle quand elle est

vraie. Cela conduit inévitablement accepter aussi plus souvent l'hypothèse nulle quand elle est fausse (et donc, à augmenter les faux négatifs).

Pour bien comprendre l'enjeu associé à ces erreurs, prenons l'exemple de notre système judiciaire. Lorsqu'un accusé est jugé, il est présumé innocent jusqu'à preuve du contraire. Le procès est l'équivalent d'un test statistique, avec :

- H_0 : l'accusé est innocent
- H_1 : l'accusé est coupable

Commettre une erreur de type I revient à condamner à tort l'accusé (on rejette à tort H_0), donc on condamne un innocent. À l'inverse, commettre une erreur de type II revient à libérer un coupable (accepter à tort H_0). Un système de justice plus strict condamnera un plus grand nombre d'accusés, qu'ils soient coupables ou non. Un système plus strict fera donc augmenter l'erreur de type I et baisser l'erreur de type II. À vous de voir ce que vous préférez : libérer plus de coupables, ou condamner plus d'innocents ?

En statistiques, la question est tranchée puisqu'on préfère maintenir l'erreur de type I à un niveau assez faible (à 5% ou moins), quitte à laisser augmenter l'erreur de type II (qui est considérée comme acceptable jusqu'à 20% environ). Toutefois, seule l'erreur de type I est sous notre contrôle. En effet, c'est nous qui la choisissons lorsque l'on fixe le seuil α de nos tests statistiques.

> **!** À retenir
>
> C'est vous qui fixez l'erreur de type I lorsque vous faites un test statistique. L'erreur de type I est le seuil α du test, que l'on fixe en général à 0,05 (soit 5%) dans le domaine des sciences du vivant.

Une fois que le seuil α est fixé, l'erreur β l'est aussi dans une certaine mesure. Mais on ne peut la connaitre avec précision car elle dépend de beaucoup de choses, notamment la taille des échantillons dont on dispose, la variabilité des données, le type de test réalisé, etc. En général, **plus la taille de l'échantillon sera grande, plus l'erreur β sera faible, et donc plus la puissance sera élevée**. De même, par rapport aux tests non paramétriques, les tests paramétriques

permettent de minimiser l'erreur β et donc d'augmenter la puissance.

Puisque la puissance statistique vaut $1-\beta$, cela revient à dire que les tests paramétriques sont plus puissants que les tests non paramétriques (parfois, beaucoup plus). Au contraire des erreurs de type I et II, la puissance est une grandeur que l'on souhaite maximiser. On aimerait en effet être capables de systématiquement rejeter H_0 quand elle est fausse. Nous avons vu plus haut que c'est hélas impossible. Mais choisir le bon test et la bonne procédure statistique permettent néanmoins d'augmenter la puissance, jusqu'à un certain point. C'est la raison pour laquelle on réalisera toujours un test paramétrique si les données dont on dispose le permettent (donc si les conditions d'application des tests paramétriques sont respectées). Et ce n'est qu'en dernier recours qu'on se tournera vers les tests non paramétriques, toujours moins puissants.

> **!** Important
>
> Un test paramétrique est toujours plus puissant que ses homologues non paramétriques. Avec un test paramétrique, il est donc plus probable de rejeter H_0 à raison qu'avec un test non paramétrique.

1.9 Bilan

Nous avons vu dans ce chapitre quelle est la procédure à suivre pour réaliser un test de comparaison de la moyenne d'une population à une valeur théorique :

1. examen préliminaire des données
2. calcul de statistiques descriptives
3. création de graphiques exploratoires
4. vérification des conditions d'application du test paramétrique
5. réalisation du test paramétrique ou non paramétrique selon l'issue de l'étape 4

Mais nous avons aussi abordé des notions statistiques essentielles pour la suite :

1. Les ingrédients indispensables pour réaliser un test statistique (les hypothèses nulle et alternative, la statistique du test et le seuil α).
2. La $p-$value et la décision du test.
3. Les erreurs de type I (α) et II (β).
4. La puissance statistique (1 - β) qui n'a rien à voir avec la notion de précision.
5. La notion de test paramétrique ou non paramétrique.

Assurez-vous d'avoir les idées claires sur toutes ces notion car elles sont absolument centrales pour ne pas faire/dire de bêtises lorsque l'on analyse des données.

1.10 Exercice d'application

Le fichier `Temperature2.csv` contient les données brutes d'une seconde étude similaire, réalisée à plus grande échelle. Importez ces données et analysez-les afin de vérifier si la température corporelle moyenne des adultes en bonne santé vaut bien 37°C. Comme toujours, avant de vous lancer dans la réalisation des tests statistiques, prenez le temps d'examiner vos données comme nous l'avons décrit dans la Section 1.4 et la Section 1.5, afin de savoir où vous allez, et de repérer les éventuelles données manquantes ou aberrantes. Enfin, interprétez les résultats à la lumière des notions que nous avons abordées ici (en particulier la notion de puissance statistique).

2 Comparaison de moyennes : deux échantillons appariés

2.1 Pré-requis

Pour ce nouveau chapitre, je vous conseille de travailler dans un nouveau script que vous placerez dans votre répertoire de travail, et dans une nouvelle session de travail (Menu `Session > Restart R`). Inutile en revanche de créer un nouveau `Rproject` : vous pouvez tout à fait avoir plusieurs script dans le même répertoire de travail et pour un même `Rproject`. Comme toujours, consultez le livre en ligne du semestre 3 si vous ne savez plus comment faire.

Si vous êtes dans une nouvelle session de travail (ou que vous avez quitté puis relancé `RStudio`), vous devrez penser à recharger en mémoire les packages utiles. Dans ce chapitre, vous aurez besoin d'utiliser les mêmes packages que précédemment :

- le `tidyverse` (Wickham 2023), qui comprend notamment le package `readr` (Wickham, Hester, et Bryan 2024), pour importer facilement des fichiers `.csv` au format `tibble`, le package `dplyr` (Wickham et al. 2023), pour manipuler des tableaux, et le package `ggplot2` (Wickham et al. 2024) pour les représentations graphiques.
- `skimr` (Waring et al. 2022), qui permet de calculer des résumés de données très informatifs.

```
library(tidyverse)
library(skimr)
```

Vous aurez également besoin des jeux de données suivants que vous pouvez dès maintenant télécharger dans votre répertoire de travail :

- Autruches.csv
- Testosterone.csv

Enfin, je spécifie ici une fois pour toutes le thème que j'utiliserai pour tous les graphiques de ce chapitre. Libre à vous de choisir un thème différent ou de vous contenter du thème proposé par défaut :

```
theme_set(theme_bw())
```

2.2 Contexte

On s'intéresse ici à la comparaison de 2 séries de données dont les observations sont liées 2 à 2. C'est par exemple le cas lorsque l'on fait subir un traitement à différents sujets et que l'on souhaite comparer les mesures obtenues avant et après le traitement.

Autrement dit, dans les plans d'expériences appariés, **les deux traitements** ou modalités **sont appliqués à chaque unité d'échantillonnage** : chaque sujet ou unité d'échantillonnage fournit plusieurs valeurs. Ça n'était pas le cas du chapitre précédent (Chapitre 1) où chaque adulte n'avait fourni qu'une unique valeur de température.

Voici quelques exemples de situations qui devraient être traitées avec des tests sur données appariées :

- Comparaison de la masse de patients avant et après une hospitalisation.
- Comparaison de la diversité de peuplements de poissons dans des lacs avant et après contamination par des métaux lourds.
- Test des effets d'une crème solaire appliquée sur un bras de chaque volontaire alors que l'autre bras ne reçoit qu'un placébo.
- Test des effets du tabagisme dans un échantillon de fumeurs, dont chaque membre est comparé à un non fumeur choisi pour qu'il lui ressemble le plus possible en terme d'âge, de masse, d'origine ethnique et sociale, etc.
- Test des effets que les conditions socio-économiques ont sur les préférences alimentaires en comparant des vrais

jumeaux élevés dans des familles adoptives séparées qui diffèrent en termes de conditions socio-économiques.

Les 2 derniers exemples montrent que même des individus séparés peuvent constituer une “paire statistique” s’ils partagent un certain nombre de caractéristiques (physiques, environnementales, génétiques, comportementales, etc.) pertinentes pour l’étude.

Ici, nous allons nous intéresser au lien qui pourrait exister entre la production de testostérone et l’immunité chez une espèce d’oiseau vivant en Amérique du Nord, le carouge à épaulettes.

Figure 2.1: Le carouge à épaulettes

Chez de nombreuses espèces, les mâles ont plus de chances d’attirer des femelles s’ils produisent des niveaux de testostérone élevés. Est-ce que la forte production de testostérone de certains mâles a un coût, notamment en terme d’immunocompétence ? Autrement dit, est-ce que produire beaucoup de testostérone au moment de la reproduction (ce qui fournit un avantage sélectif) se traduit par une immunité plus faible par la suite, et donc une plus forte susceptibilité de contracter des maladies (ce qui constitue donc un désavantage sélectif) ? Ce type de question est central pour comprendre comment l’allocation des ressources affecte à la fois la survie et la fécondité des individus.

Pour étudier cette question, une équipe de chercheurs (Hasselquist et al. 1999) a mis en place le dispositif expérimental suivant. Les niveaux de testostérone de 13 carouges à épaulettes mâles ont été artificiellement augmentés par l'implantation chirurgicale d'un microtube perméable contenant de la testostérone. L'immunocompétence a été mesurée pour chaque oiseau avant et après l'opération chirurgicale. La variable mesurée est la production d'anticorps suite à l'exposition des oiseaux avec un antigène non pathogène mais censé déclencher une réponse immunitaire. Les taux de production d'anticorps sont exprimés en logarithmes de densité optique par minute $\left(\ln \frac{mOD}{min}\right)$. Si la production de testostérone influence l'immunocompétence, on s'attend à observer des différence de production d'anticorps avant et après l'intervention chirurgicale.

2.3 Importation et mise en forme des données

Les données se trouvent dans le fichier `Testosterone.csv`. Importez ces données dans un objet nommé `Testo` et affichez son contenu.

```
Testo
```

```
# A tibble: 13 x 5
   blackbird beforeImplant afterImplant logBeforeImplant logAfterImplant
       <dbl>         <dbl>        <dbl>            <dbl>           <dbl>
 1         1           105           85             4.65            4.44
 2         2            50           74             3.91            4.3
 3         3           136          145             4.91            4.98
 4         4            90           86             4.5             4.45
 5         5           122          148             4.8             5
 6         6           132          148             4.88            5
 7         7           131          150             4.88            5.01
 8         8           119          142             4.78            4.96
 9         9           145          151             4.98            5.02
10        10           130          113             4.87            4.73
11        11           116          118             4.75            4.77
12        12           110           99             4.7             4.6
13        13           138          150             4.93            5.01
```

Visiblement, il n'y a pas de données manquantes mais certaines variables sont inutiles. En effet, nous aurons besoin des variables transformées en logarithmes, mais pas des 2 colonnes `beforeImplant` et `afterImplant`. Nous allons donc les retirer avec la fonction `select()`. Par ailleurs, la variable `blackbird` est importante puisque chaque individu a fourni 2 valeurs de production d'anticorps : 1 avant et 1 après l'opération chirurgicale. Il sera donc important de conserver cet identifiant individuel. Toutefois, il apparaît ici sous la forme d'une variable numérique alors qu'il s'agit d'un identifiant, d'un code. Il faut donc le transformer en facteur car cela n'aurait pas de sens calculer une moyenne des identifiants par exemple. Pour cela, nous utiliserons la fonction `factor()` à l'intérieur de `mutate()`. Enfin, nous renommerons les colonnes avec `rename()` pour avoir des noms plus courts et plus faciles à utiliser. Si vous ne vous rappelez plus comment utilisez ces fonctions, consulter ces chapitres du livre en ligne de biométrie du semestre 3 : `select()` et `rename()`, `mutate()` et `factor()`. Enfin, nous donnerons le nom `Testo_large` au tableau modifié :

```
Testo_large <- Testo |>
  select(-beforeImplant, -afterImplant) |>  # Suppression des colonnes inutiles
  mutate(blackbird = factor(blackbird)) |>  # Transformation en facteur
  rename(ID = blackbird,                     # Changement des noms de variables
         Before = logBeforeImplant,
         After = logAfterImplant)

Testo_large
```

```
# A tibble: 13 x 3
   ID    Before After
   <fct>  <dbl> <dbl>
 1 1       4.65  4.44
 2 2       3.91  4.3
 3 3       4.91  4.98
 4 4       4.5   4.45
 5 5       4.8   5
 6 6       4.88  5
 7 7       4.88  5.01
 8 8       4.78  4.96
 9 9       4.98  5.02
10 10      4.87  4.73
```

```
11 11       4.75  4.77
12 12       4.7   4.6
13 13       4.93  5.01
```

Le tableau `Testo_large` dont nous disposons maintenant n'est pas dans un format qui nous permettra de réaliser toutes les opérations dont nous aurons besoin. En réalité, il ne s'agit pas d'un "tableau rangé" au sens du tidyverse. Un tableau rangé est un tableau dans lequel chaque ligne correspond à une unique observation et chaque colonne correspond à une unique variable. Ici, nous devrions avoir les 3 variables suivantes :

1. L'identifiant des individus. La colonne `ID` correspond à cette variable.
2. Le moment auquel chaque mesure a été effectuée, avant ou après l'opération chirurgicale. Cette information est pour l'instant stockée dans l'en-tête des colonnes 2 et 3 du tableau `Testo_large`
3. La mesure de réponse immunitaire (en logarithme de la densité optique par minute). Cette information est pour l'instant stockée sous forme de valeurs numériques dans les colonnes 2 et 3 du tableau `Testo_large`

Pour obtenir un tableau rangé, il nous faut donc réorganiser les colonnes 2 et 3 du tableau `Testo_large` :

- l'entête de ces 2 colonnes devrait constituer une nouvelle variable que nous nommerons `Moment`
- le contenu de ces 2 colonnes (les valeurs numériques) devrait constituer une nouvelle variable que nous nommerons `DO` (pour densité optique).

Pour effectuer cette transformation, nous utiliserons la fonction `pivot_longer()` du package `tidyr` (il est déjà chargé en mémoire si vous avez chargé le `tidyverse`). Comme son nom l'indique, cette fonction produira un tableau plus "long" (qui aura plus de lignes) que le tableau de départ. Nous l'appellerons donc `Testo_long` :

```
Testo_long <- Testo_large |>
  pivot_longer(cols = c(Before, After),   # Les colonnes qu'on veut réorganiser
               names_to = "Moment",   # Quel nom donner à la variable qui contiendra les
               values_to = "DO") |>      # Quel nom donner à la variable qui contiendra l
```

```
    mutate(Moment = factor(Moment, levels = c("Before", "After")))

  Testo_long
```

```
# A tibble: 26 x 3
   ID    Moment    DO
   <fct> <fct>  <dbl>
 1 1     Before  4.65
 2 1     After   4.44
 3 2     Before  3.91
 4 2     After   4.3
 5 3     Before  4.91
 6 3     After   4.98
 7 4     Before  4.5
 8 4     After   4.45
 9 5     Before  4.8
10 5     After   5
# i 16 more rows
```

Ce nouvel objet contient les mêmes données que précédemment, mais sous un format différent (il contient maintenant 26 lignes et non plus 13) : il s'agit d'un tableau rangé.

La plupart du temps, on a besoin de ces 2 formats de tableaux quand nous traitons des données. Le tableau au format long est à privilégier pour les représentations graphiques et les tests statistiques, et le format court sert souvent à présenter des résultats sous une forme synthétique. Mais parfois (et c'est justement le cas quand on dispose de données appariées comme pour notre exemple de lien entre testostérone et immunocompétence), le tableau au format large permettra de faire certains graphiques, certains tests ou certaines manipulations plus facilement que le tableau rangé au format long. Si on ne dispose que d'un tableau au format large, on peut passer au format long, comme nous venons de le faire, grâce à la fonction `pivot_longer()`. Et si on ne dispose que d'un tableau au format long, on peut passer au format large grâce à la fonction `pivot_wider()`. Nous avons déjà évoqué cette fonction dans le livre en ligne de biométrie du semestre 4 pour mettre en forme des résultats obtenus avec `summarise()` (par exemple ici) ou `reframe()` (ou là), et je vous encourage à y jeter un œil à nouveau pour vous remémorer la syntaxe. Car

il est important que vous maîtrisiez ces 2 fonctions dont vous aurez très souvent besoin.

Maintenant que nous disposons de ces 2 tableaux, `Testo_large` et `Testo_long`, nous pouvons commencer à décrire nos données.

2.4 Exploration statistique des données

Pour décrire simplement les données, nous nous en tiendront ici à l'utilisation des fonctions `summary()` et `skim()`.

Pour la fonction `summary()`, le plus simple est toujours d'utiliser le tableau au format large :

```
summary(Testo_large)
```

```
       ID        Before          After
 1      :1   Min.   :3.910   Min.   :4.30
 2      :1   1st Qu.:4.700   1st Qu.:4.60
 3      :1   Median :4.800   Median :4.96
 4      :1   Mean   :4.734   Mean   :4.79
 5      :1   3rd Qu.:4.880   3rd Qu.:5.00
 6      :1   Max.   :4.980   Max.   :5.02
 (Other):7
```

On constate ici que pour les 2 traitements, les valeurs des différents indices sont très proches entre les 2 séries de données, avec des valeurs de densité optiques (DO) légèrement supérieures après l'opération chirurgicale (sauf pour le premier quartile).

Pour la fonction `skim()` le plus simple est là aussi d'utiliser le tableau large :

```
skim(Testo_large)
```

```
-- Data Summary ------------------------
                           Values
Name                       Testo_large
Number of rows             13
```

```
Number of columns          3
-----------------------
Column type frequency:
  factor                   1
  numeric                  2
-------------------------
Group variables            None

-- Variable type: factor -------------------------------------------------------
  skim_variable n_missing complete_rate ordered n_unique top_counts
1 ID                    0             1 FALSE         13 1: 1, 2: 1, 3: 1, 4: 1

-- Variable type: numeric ------------------------------------------------------
  skim_variable n_missing complete_rate mean    sd   p0 p25  p50  p75 p100 hist
1 Before                0             1 4.73 0.280 3.91 4.7 4.8  4.88 4.98
2 After                 0             1 4.79 0.262 4.3  4.6 4.96 5    5.02
```

On arrive toutefois aux mêmes résultats avec le tableau long, à condition de grouper les données par traitement (variable `Traitement`) avec `group_by()` :

```
Testo_long |>
  group_by(Moment) |>
  skim(DO)
```

```
-- Data Summary ------------------------
                           Values
Name                       group_by(Testo_long, Mome...
Number of rows             26
Number of columns          3
-----------------------
Column type frequency:
  numeric                  1
-------------------------
Group variables            Moment

-- Variable type: numeric ------------------------------------------------------
  skim_variable Moment n_missing complete_rate mean    sd   p0 p25  p50  p75
1 DO            Before         0             1 4.73 0.280 3.91 4.7 4.8  4.88
2 DO            After          0             1 4.79 0.262 4.3  4.6 4.96 5
  p100 hist
1 4.98
2 5.02
```

Cela revient à demander à la fonction `skim()` de produire un résumé des données de densité optique (variable `DO`), pour chaque catégorie de la variable `Moment`, soit un résumé pour la catégorie `Before` (avant l'intervention chirurgicale), et un résumé pour la catégorie `After` (après l'intervention chirurgicale).

Par rapport aux résultats fournis par la fonction `summary()`, la fonction `skim()` nous permet de confirmer que les valeurs de DO sont très légèrement supérieures après l'opération (sauf pour le premier quartile). Elle nous permet également de constater que l'écart-type est du même ordre de grandeur pour les 2 catégories, bien qu'il soit légèrement plus faible après l'opération. Enfin, les petits histogrammes laissent entrevoir une distribution très asymétrique des données dans chacun des 2 groupes de mesures.

2.5 Exploration graphique des données

Ici, c'est le tableau rangé au format long qui sera le plus adapté. Lorsque nous avions une unique série de données, nous avons utilisé 2 types de représentations graphiques très similaires pour visualiser les données (les histogrammes et les graphiques de densités). Ici, nous allons utiliser ces mêmes types de graphiques mais "facettés". Les graphiques facettés ont été abordés dans le livre en ligne de biométrie du semestre 3. Ils permettent de faire des sous-graphiques pour chaque catégorie d'un facteur. Ici, le facteur `Moment` contient 2 catégories. Les facets nous permettrons donc de comparer les 2 distributions de densités optiques.

Outre ces graphiques, nous utiliserons aussi les stripcharts et les boites à moustaches pour comparer les 2 catégories. Ces 2 types de graphiques sont particulièrement adaptés pour ce genre de tâche, et seront aussi très utiles pour l'ANOVA lorsque nous aurons plus de 2 catégories à comparer.

D'une façon générale, nous disposons :

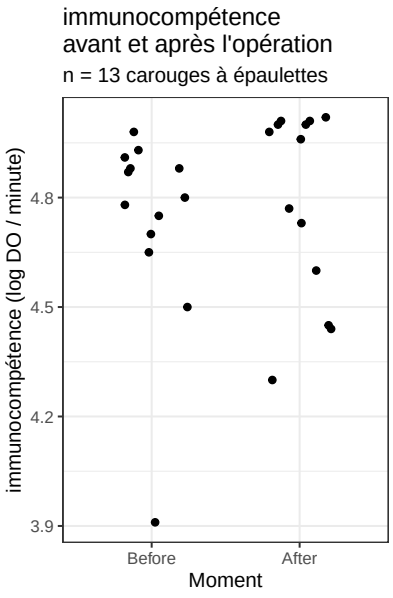
- d'une variable numérique, `DO` : la mesure de densité optique qui rend compte de l'immunocompétence des carouges à épaulettes

- d'une variable catégorielle, le facteur `Moment` : indique si les valeurs d'immunocompétences ont été mesurées avant ou après l'opération chirurgicale d'implantation de la capsule de testostérone.

Tous les graphiques présentés dans le chapitre consacré à cette situation précise dans le livre en ligne de biométrie du semestre 3, peuvent être réalisés. N'hésitez pas à le relire, en particulier la section expliquant comment faire apparaître et interpréter les encoches d'incertitudes sur des boiites à moustaches.

2.5.1 Avec un stripchart

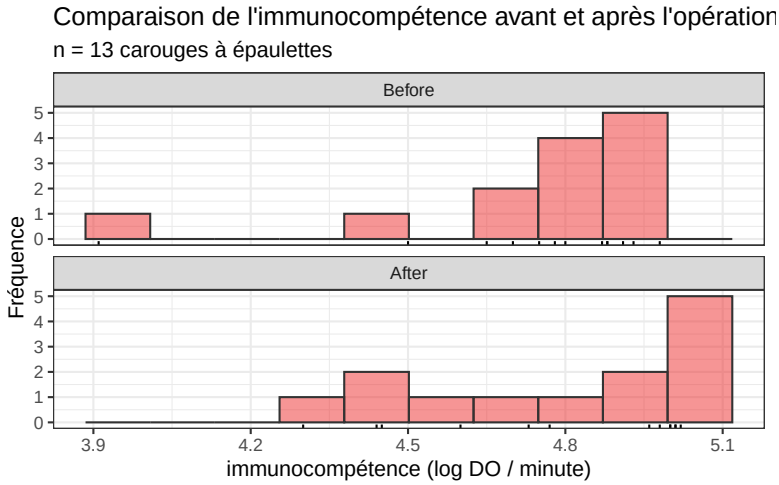
```
Testo_long |>
  ggplot(aes(x = Moment, y = DO)) +
  geom_jitter(height = 0, width = 0.25) +
  labs(y = "immunocompétence (log DO / minute)",
       title = "immunocompétence\navant et après l'opération",
       subtitle = "n = 13 carouges à épaulettes")
```



2.5.2 Avec des histogrammes facettés

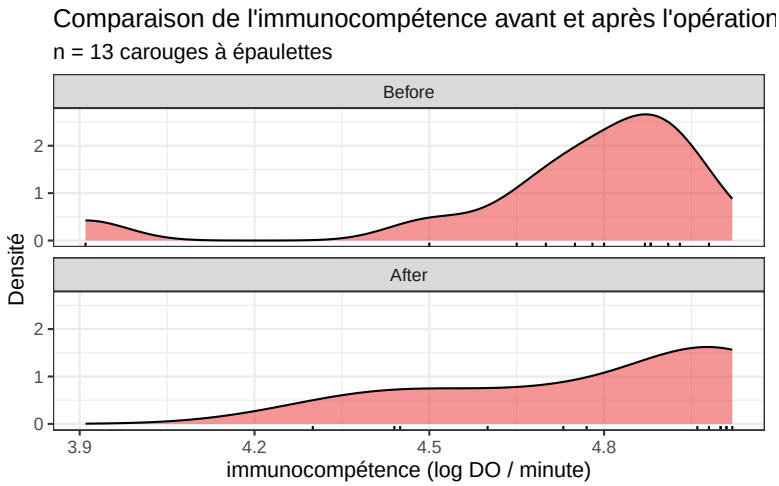
Nous allons faire un histogramme pour chaque série de données en utilisant des facettes :

```
Testo_long |>
  ggplot(aes(x = DO)) +
  geom_histogram(bins = 10, fill = "firebrick2", color = "grey20", alpha = 0.5)+
  geom_rug() +
  facet_wrap(~Moment, ncol = 1) +
  labs(x = "immunocompétence (log DO / minute)",
       y = "Fréquence",
       title = "Comparaison de l'immunocompétence avant et après l'opération chirurgicale
       subtitle = "n = 13 carouges à épaulettes")
```



2.5.3 Avec des diagrammes de densité facettés

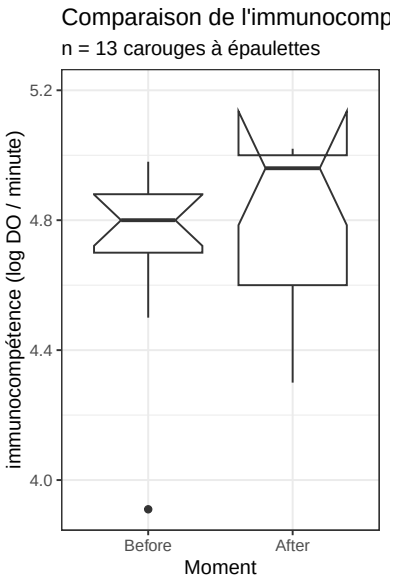
```
Testo_long |>
  ggplot(aes(x = DO)) +
  geom_density(fill = "firebrick2", alpha = 0.5) +
  geom_rug() +
  facet_wrap(~Moment, ncol = 1) +
  labs(x = "immunocompétence (log DO / minute)",
       y = "Densité",
       title = "Comparaison de l'immunocompétence avant et après l'opération chirurgicale
       subtitle = "n = 13 carouges à épaulettes")
```



2.5.4 Avec des boîtes à moustaches

```
Testo_long |>
  ggplot(aes(x = Moment, y = DO)) +
  geom_boxplot(notch = TRUE) +
  expand_limits(y = 5.2) +
  labs(y = "immunocompétence (log DO / minute)",
       title = "Comparaison de l'immunocompétence avant et après opération chirurgicale",
       subtitle = "n = 13 carouges à épaulettes")
```

```
Notch went outside hinges
i Do you want `notch = FALSE`?
```

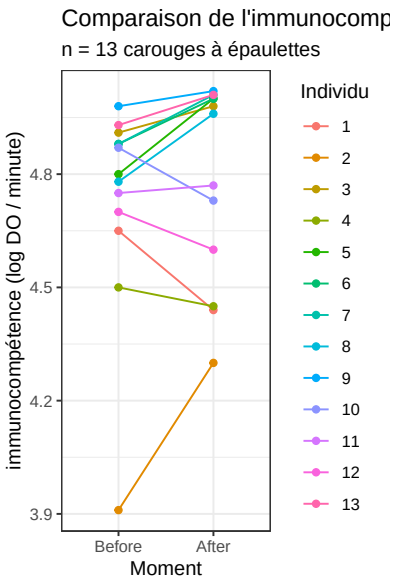


Du point de vue de la **position** des données, ces différents graphiques montrent tous que la seconde série de données (catégorie `After` : après l'opération chirurgicale) présente en moyenne des valeurs très légèrement plus élevées que la première (catégorie `Before` avant l'opération). En terme de **dispersion**, si l'on met de côté la valeur minimale de la série `Before` qui semble atypique (un individu outlier qui présente une immunocompétence très faible avant l'opération), la dispersion des données autour de la tendance centrale semble globalement plus importante pour la série `After`. Enfin, pour ce qui concerne l'**incertitude**, les intervalles de confiance à 95% des médianes (qui apparaissent sous la forme d'encoches sur les boîtes à moustaches) se chevauche assez largement, ce qui nous permet d'anticiper les résultats des tests que nous ferons ensuite : puisque les encoches se chevauchent, il y a fort à parier que le test de comparaison de moyenne ne montrera aucune différence significative. On note également que l'encoche de la série `After` est particulièrement large : la limite supérieure de l'intervalle de confiance à 95% de la médiane est supérieure à la valeur maximale observée dans l'échantillon. Cela traduit le fait que compte de la grande variabilité des données dans cette série, un échantillon de taille n = 13 n'est probablement pas suffisant pour avoir une estimation précise de la médiane.

2.5.5 Avec un nuage de points appariés

Toutes ces représentations graphiques sont certes utiles, mais elles masquent un élément crucial : ce sont les mêmes individus qui sont étudiés avant et après l'opération. Il s'agit de données appariées ! Les graphiques que nous avons faits jusque là ne permettent pas de visualiser ce lien entre les deux séries de données. Pour avoir une bonne vision de ce qui se passe, il nous faut faire apparaître ce lien entre les 2 séries de données :

```
Testo_long |>
  ggplot(aes(x = Moment, y = DO, group = ID, color = ID)) +
  geom_line() +
  geom_point() +
  labs(y = "immunocompétence (log DO / minute)",
       title = "Comparaison de l'immunocompétence avant et après opération chirurgicale",
       subtitle = "n = 13 carouges à épaulettes",
       color = "Individu")
```



Ce graphique nous donne une image très différente de la réalité des données. On constate ici que l'immunocompétence de certains individus augmente après l'opération (parfois fortement), alors que pour d'autres, elle diminue.

Une façon d'estimer si les changements d'immunocompétence sont majoritairement orientés dans un sens ou non est de calculer l'intervalle de confiance à 95% de la différence d'immunocompétence entre avant et après l'opération. Pour cela, on peut calculer, grâce au tableau large `Testo_large`, les différences d'immunocompétences (DO après opération moins DO avant opération), pour chacun des 13 individus. puis, grâce à la fonction `mean_cl_normal()` déjà utilisée à plusieurs reprises, on calcul l'intervalle de confiance à 95% de la moyenne de cette différence :

```
# Calcul de la différence de DO (After - Before)
Testo_large <- Testo_large |>
  mutate(Difference = After - Before)

# Affichage du tableau
Testo_large
```

```
# A tibble: 13 x 4
   ID    Before After Difference
   <fct>  <dbl> <dbl>      <dbl>
 1 1       4.65  4.44    -0.21
 2 2       3.91  4.3      0.390
 3 3       4.91  4.98     0.0700
 4 4       4.5   4.45    -0.0500
 5 5       4.8   5        0.200
 6 6       4.88  5        0.120
 7 7       4.88  5.01     0.130
 8 8       4.78  4.96     0.180
 9 9       4.98  5.02     0.0400
10 10      4.87  4.73    -0.140
11 11      4.75  4.77     0.0200
12 12      4.7   4.6     -0.100
13 13      4.93  5.01     0.0800
```

```
# Calcul de la moyenne des différences et de son IC95%
Testo_large |>
  reframe(mean_cl_normal(Difference))
```

```
# A tibble: 1 x 3
       y    ymin  ymax
```

```
   <dbl>   <dbl> <dbl>
1 0.0562 -0.0401 0.152
```

On constate ici que la moyenne des différences de densité optique vaut 0.06, soit une valeur positive, qui montre que l'immunocompétence augmente après l'opération (ce qui semble aller à l'opposé de l'hypothèse des chercheurs). Cette moyenne reste néanmoins très proche de 0. D'ailleurs, l'intervalle de confiance de cette moyenne comprend les valeurs situées entre -0.04 et +0.15. La valeur 0 est donc comprise dans cet intervalle. Le zéro fait donc partie des valeurs les plus probables pour la moyenne de ces différences dans la populations générale. C'est là encore un résultat qui nous permet d'anticiper sur les résultats du tests statistique que nous ferons ensuite.

2.6 Le test paramétrique

2.6.1 Procédure

Le test paramétrique permettant de comparer la moyenne sur des séries appariées est là encore un test de Student : le **test de Student sur données appariées** (étonnant non ?…). En réalité, ce test de Student n'est pas un test de comparaison de moyennes entre 2 séries de données. La procédure est la suivante :

1. Pour chaque individu, calculer la différence d'immunocompétence entre les deux temps de l'expérience (DO après - DO avant opération). C'est ce que nous avons fait plus haut en ajoutant la colonne `Difference` au tableau `Testo_large`.
2. Puisque nous avons 13 individus, nous aurons 13 valeurs de différences. La moyenne de cette différence sera comparée à la valeur théorique 0. Autrement dit, si cette moyenne vaut 0, l'immunocompétence sera la même avant et après l'opération. Si la moyenne des différence n'est pas égale 0, alors nous aurons prouvé qu'il existe une différence d'immunocompétence entre les 2 groupes, nous aurons prouvé que la procédure chirurgicale d'implantation de la capsule de testostérone a un

impact sur l'immunocompétence des carouges à épaulettes

> ⚠ Attention
>
> Dans un test sur données appariées, on s'intéresse à **la moyenne des différences** entre les données des 2 séries. Cette moyenne est alors comparée à la valeur théorique $\mu = 0$. Ce test est donc équivalent au test vu dans le Chapitre 1 sur la comparaison de la moyenne d'une population à une valeur théorique.
> Notez également que **la moyenne des différences** n'est pas équivalente à **la différence des moyennes**. La différence des moyennes est une grandeur qui nous sera utile dans le chapitre suivant (Chapitre 3) sur la comparaison de la moyenne de deux populations lorsque les données sont indépendantes.

2.6.2 Conditions d'application

Les conditions d'application de ce test paramétrique sont presque les mêmes que pour le test de Student à un échantillon :

1. Les individus sur lesquels portent la comparaison doivent être issus d'un échantillonnage aléatoire. Comme toujours, en l'absence d'indication contraire, on considère que cette condition est vérifiée.
2. Les différences par paires entre les 2 modalités du traitement doivent suivre une distribution Normale. Attention, ce n'est donc pas les données brutes de chaque série qui doivent suivre une loi Normale, mais bien la différence "après" - "avant" calculée pour chaque individu. Nous avons déjà calculé ces différences plus haut :

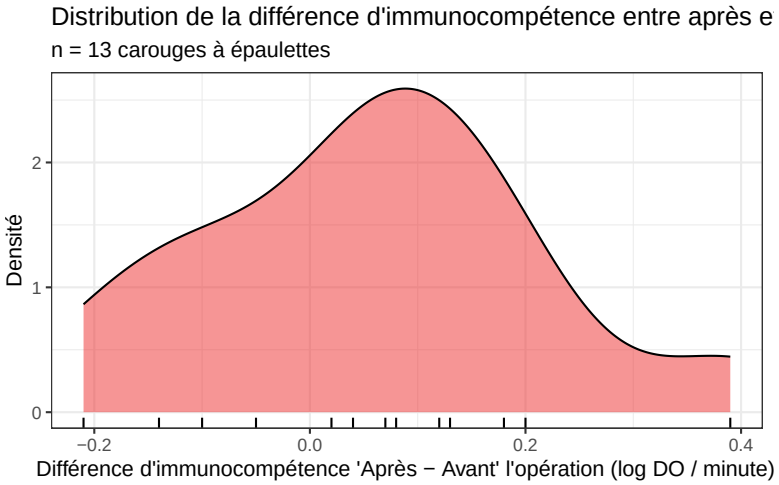
```
# On s'intéresse aux 13 différences calculées sur les 13 individus
Testo_large
```

```
# A tibble: 13 x 4
   ID    Before After Difference
   <fct>  <dbl> <dbl>      <dbl>
```

```
 1  1        4.65  4.44    -0.21
 2  2        3.91  4.3      0.390
 3  3        4.91  4.98     0.0700
 4  4        4.5   4.45    -0.0500
 5  5        4.8   5        0.200
 6  6        4.88  5        0.120
 7  7        4.88  5.01     0.130
 8  8        4.78  4.96     0.180
 9  9        4.98  5.02     0.0400
10  10       4.87  4.73    -0.140
11  11       4.75  4.77     0.0200
12  12       4.7   4.6     -0.100
13  13       4.93  5.01     0.0800
```

Il nous faut donc tester la Normalité de la nouvelle variable `Difference`. Commençons par en faire un graphique :

```
Testo_large |>
  ggplot(aes(x = Difference)) +
  geom_density(fill = "firebrick2", alpha = 0.5) +
  geom_rug() +
  labs(x = "Différence d'immunocompétence 'Après - Avant' l'opération (log DO / minute)",
       y = "Densité",
       title = "Distribution de la différence d'immunocompétence entre après et avant l'o
       subtitle = "n = 13 carouges à épaulettes")
```



Compte tenu du faible nombre d'individus (n = 13), la forme de cette courbe de densité n'est pas si éloignée que ça d'une courbe en cloche (notez que ce n'était pas du tout le cas pour

les données brutes de chaque série de départ qui ont toutes les deux des distributions très éloignées de la distribution Normale). On le vérifie avec un test de normalité de Shapiro-Wilk :

- H_0 : la différence d'immunocompétence des individus suit une distribution Normale.
- H_1 : la différence d'immunocompétence des individus ne suit pas une distribution Normale.

```
Testo_large |>
  pull(Difference) |>
  shapiro.test()
```

```
    Shapiro-Wilk normality test

data:  pull(Testo_large, Difference)
W = 0.97949, p-value = 0.977
```

> Au seuil $\alpha = 0.05$, on ne peut pas rejeter l'hypothèse nulle de normalité pour la différence d'immunocompétence entre après et avant l'intervention chirurgicale (test de Shapiro-Wilk, $W = 0.98$, $p = 0.977$).

Les conditions d'application du test paramétrique sont donc réunies.

> ⚠ Attention !
>
> Pour ce test, la Normalité doit bien être vérifiée sur la différence entre les 2 groupes de valeurs, et non sur chaque groupe de valeur pris séparément. C'est une source d'erreur fréquente. Ici, les données de départ (DO avant et DO après) ne suivaient pas du tout une distribution Normale. Pourtant, la différence de DO suit bel et bien la distribution Normale, nous permettant de faire le test paramétrique.

2.6.3 Réalisation du test et interprétation

Le test de Student sur données appariées peut se faire de 2 façons distinctes. Les 2 méthodes fournissent exactement les mêmes résultats, seule la syntaxe utilisée change. Quelle que soit la méthode utilisée, les hypothèses nulles et alternatives sont toujours les mêmes :

- H_0 : le changement moyen de production d'anticorps après la pose chirurgicale de l'implant de testostérone est nul ($\mu_{Diff} = 0$). La procédure chirurgicale n'a pas d'effet sur l'immunocompétence. Les variations observées ne sont que le fruit du hasard de l'échantillonnage.
- H_1 : le changement moyen de production d'anticorps après la pose chirurgicale de l'implant de testostérone n'est pas nul ($\mu_{Diff} \neq 0$). La procédure chirurgicale a effet significatif sur l'immunocompétence. Les variations observées ne sont pas uniquement dues à la fluctuation d'échantillonnage.

2.6.3.1 Première syntaxe

```
# Méthode n°2 : avec les 2 séries de données et le tableau au format large
t.test(Testo_large$Before, Testo_large$After, paired = TRUE)
```

```
    Paired t-test

data:  Testo_large$Before and Testo_large$After
t = -1.2714, df = 12, p-value = 0.2277
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -0.15238464  0.04007695
sample estimates:
mean difference
    -0.05615385
```

Avec cette première syntaxe, on indique le nom des 2 colonnes du tableau `Testo_large` qui contiennent les 2 séries de données. Il nous faut taper spécifiquement "`Testo_large$Before`" et "`Testo_large$After`", et non

pas simplement “`Before`” et “`After`” car la fonction `t.test` n’a aucun moyen de savoir que “`Before`” et “`After`” sont des colonnes du tableau `Testo-large`. Nous devons donc être explicites. Il est en outre indispensable d’indiquer “`paired = TRUE`” pour faire un test de Student sur données appariées. Omettre cet argument reviendrait à faire un test de Student sur données indépendantes, qui est moins puissant que le test sur données appariées. Ça n’est donc pas souhaitable avec les données dont nous disposons et ça pourrait même conduire à des conclusions fausses.

Ici, voilà la conclusion de ce test :

> Le test de Student sur données appariées ne permet pas de montrer de changement d’immunocompétence suite à l’intégration de l’implant chirurgical de testostérone. On ne peut pas rejeter l’hypothèse nulle au seuil $\alpha = 0.05$ ($t = -1.27$, $ddl = 12$, $p = 0.223$). La moyenne des différences de densités optiques observées entre avant et après l’intervention chirurgicale vaut -0.056 (intervalle de confiance à 95% de cette différence : [-0.152 ; 0.040])

Donc visiblement, une forte production de testostérone n’est pas significativement associée à une baisse de l’immunocompétence.

Notez bien que les résultats et leurs interprétation dépendent de l’ordre dans lequel les 2 séries de données sont indiquées dans la fonction `t.test()`. Pour vous en convaincre, regardez ce que donne cette commande :

```
t.test(Testo_large$After, Testo_large$Before, paired = TRUE)
```

Qu’est-ce qui change ? Et qu’est-ce qui reste inchangé ?

2.6.3.2 Deuxième syntaxe

```
# Méthode n°3 : avec la variable Diff, mu = 0, et le tableau au format large
t.test(Testo_large$Difference, mu = 0)
```

```
    One Sample t-test

data:  Testo_large$Difference
t = 1.2714, df = 12, p-value = 0.2277
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.04007695  0.15238464
sample estimates:
 mean of x
0.05615385
```

Comme expliqué plus haut, le test de Student sur données appariées est strictement équivalent à un test de Student à un échantillon pour lequel on compare la moyenne des différences individuelles à 0. Là encore, les résultats produits et leur interprétation sont identiques aux deux tests précédents. La seule différence concerne les signes puisque le premier test regardait la différence "Before - After" alors que ce test regarde la différence "After - Before" (que nous avons calculée manuellement).

À vous donc de choisir la syntaxe qui vous paraît la plus parlante ou celle que vous avez le plus de facilité à retenir et à interpréter.

2.7 L'alternative non paramétrique

Comme pour le test de Student à un échantillon, lorsque les conditions d'application du test de Student sur données appariées ne sont pas vérifiées (c'est à dire lorsque la différence entre les données appariées des deux séries ne suit pas une loi Normale), il faut utiliser un test non paramétrique équivalent.

Il s'agit là encore du **test de Wilcoxon des rangs signés** qui s'intéresse aux médianes. Les hypothèses nulles et alternatives sont les suivantes :

- H_0 : le changement **médian** de production d'anticorps après la pose chirurgicale de l'implant de testostérone est nul ($med_{Diff} = 0$).

- H_1 : le changement **médian** de production d'anticorps après la pose chirurgicale de l'implant de testostérone n'est pas nul ($med_{Diff} \neq 0$).

Comme pour le test de Student, 2 syntaxes sont possibles et strictement équivalentes. Il est important de ne pas oublier l'argument `paired = TRUE` afin de s'assurer que l'on réalise bien un test sur données appariées. Enfin, l'argument `conf.int = TRUE` doit être ajouté afin que la (pseudo-) médiane et son intervalle de confiance à 95% soient calculés et affichés.

```
wilcox.test(Testo_large$Before, Testo_large$After, paired = TRUE,
            conf.int = TRUE)
```

```
    Wilcoxon signed rank exact test

data:  Testo_large$Before and Testo_large$After
V = 30, p-value = 0.3054
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -0.145  0.040
sample estimates:
(pseudo)median
        -0.055
```

```
wilcox.test(Testo_large$Difference, mu = 0, conf.int = TRUE)
```

```
    Wilcoxon signed rank exact test

data:  Testo_large$Difference
V = 61, p-value = 0.3054
alternative hypothesis: true location is not equal to 0
95 percent confidence interval:
 -0.040  0.145
sample estimates:
(pseudo)median
         0.055
```

Ici, la conclusion de ce test est :

Le test de Wilcoxon des rangs signés n'a pas permis de montrer de changement d'immunocompétence suite à l'intégration de l'implant chirurgical de testostérone. On ne peut pas rejeter l'hypothèse nulle au seuil $\alpha = 0.05$ ($V = 61$, $p = 0.305$). La médiane des différences de densités optiques observées entre après et avant l'intervention chirurgicale vaut 0.055 (intervalle de confiance à 95% de cette différence : [-0.040 ; 0.145]).

2.8 Exercice d'application

Les autruches vivent dans des environnements chauds et elles sont donc fréquemment exposées au soleil durant de longues périodes. Dans des environnements similaires, les mammifères ont des mécanismes physiologiques leur permettant de réduire la température de leur cerveau par rapport à celle de leur corps. Une équipe de chercheurs (Fuller et al. 2003) a testé si les autruches pouvaient faire de même. La température du corps et du cerveau de 37 autruches a été enregistrée par une journée chaude typique. Les résultats, exprimés en degrés Celsius, figurent dans le fichier `Autruches.csv`.

Importez ces données et faites-en l'analyse pour savoir s'il existe une différence de température moyenne entre le corps et le cerveau des autruches. Vos observations chez les autruches sont-elles conformes à ce qui est observé chez les mammifères dans un environnement similaire ? Comme toujours, vous commencerez par faire une analyse descriptive des données, sous forme numérique et graphique, avant de vous lancer dans les tests d'hypothèses.

3 Comparaison de moyennes : deux échantillons indépendants

3.1 Pré-requis

Comme pour chaque nouveau chapitre, je vous conseille de travailler dans un nouveau script que vous placerez dans votre répertoire de travail, et dans une nouvelle session de travail (Menu `Session > Restart R`). Inutile en revanche de créer un nouveau `Rproject` : vos pouvez tout à fait avoir plusieurs script dans le même répertoire de travail et pour un même `Rproject`. Comme toujours, consultez le livre en ligne du semestre 3 si vous ne savez plus comment faire.

Si vous êtes dans une nouvelle session de travail (ou que vous avez quitté puis relancé `RStudio`), vous devrez penser à recharger en mémoire les packages utiles. Dans ce chapitre, vous aurez besoin d'utiliser :

- le `tidyverse` (Wickham 2023), qui comprend notamment le package `readr` (Wickham, Hester, et Bryan 2024), pour importer facilement des fichiers `.csv` au format `tibble`, le package `dplyr` (Wickham et al. 2023), pour manipuler des tableaux, et le package `ggplot2` (Wickham et al. 2024) pour les représentations graphiques.
- `readxl` (Wickham et Bryan 2023), pour importer facilement des fichiers Excel au format `tibble`.
- `skimr` (Waring et al. 2022), qui permet de calculer des résumés de données très informatifs.
- `car` (Fox, Weisberg, et Price 2023), qui permet d'effectuer le test de comparaison des variances de Levene.

- le package `palmerpenguins` (Horst, Hill, et Gorman 2022) pour accéder au jeu de données `penguins` que nous utiliserons pour les exercices d'application.

```
library(tidyverse)
library(readxl)
library(skimr)
library(car)
library(palmerpenguins)
```

Vous aurez également besoin des jeux de données suivants que vous pouvez dès maintenant télécharger dans votre répertoire de travail :

- HommesFemmes.xls
- HornedLizards.csv

Enfin, je spécifie ici une fois pour toutes le thème que j'utiliserai pour tous les graphiques de ce chapitre. Libre à vous de choisir un thème différent ou de vous contenter du thème proposé par défaut :

```
theme_set(theme_bw())
```

3.2 Contexte

On s'intéresse maintenant aux méthodes permettant de comparer la moyenne de deux groupes ou de deux traitements dans la cas d'échantillons indépendants. Au contraire de la situation décrite dans le Chapitre 2, dans ce type de design expérimentaux, les deux traitements sont appliqués à des échantillons indépendants issus de 2 groupes ou populations distincts. Chaque individu collecté, ou chaque unité expérimentale observée, ne fournit qu'une seule valeur, indépendante de toutes les autres.

Cette situation est extrêmement classique dans le domaine de l'écologie au sens large. Ainsi, par exemple, lorsque l'on souhaite comparer 2 sites, on réalise des prélèvements dans chacun des 2 sites. Chaque prélèvement ne fournit qu'une valeur pour l'un des 2 sites.

Ici, nous allons examiner une espèce intéressante, le lézard cornu *Phrynosoma mcallii*, qui possède une frange de piquants autour de la tête. Une équipe d'herpétologues (Young, Brodie, et Brodie 2004) a étudié la question suivante : des piquants plus longs autour de la tête protègent-ils le lézard cornu de son prédateur naturel, la pie grièche migratrice *Lanius ludovicianus* ? Ce prédateur a en effet une particularité : il accroche ses proies mortes à des barbelés ou des branches pour les consommer plus tard. Les chercheurs ont donc mesuré la longueur des cornes de 30 lézards retrouvés morts et accrochés dans des arbres par la pie grièche migratrice. Et en parallèle, ils ont mesuré les cornes de 154 individus vivants et en bonne santé choisis au hasard dans la population.

(a) Lézard cornu vivant

(b) Lézard cornu mort

(c) Pie grièche

Figure 3.1: Le lézard cornu et son prédateur

Nous disposons donc de 2 groupes indépendants : chaque lézard n'a fourni qu'une valeur de longueur de cornes, et chaque lézard n'appartient qu'à un groupe, vivant ou mort. Nous sommes donc dans la situation typique de la comparaison de moyennes de 2 populations avec des données indépendantes. Avant de procéder aux tests, et comme toujours, nous allons commencer par importer et mettre en forme les données (si besoin), puis nous devrons explorer les données, à l'aide d'une part d'indices statistiques de position, de dispersion et d'incertitude et d'autre part de représentations graphiques pertinentes. Enfin, nous vérifierons les conditions d'application du test paramétrique de Student avant de réaliser ce test si les conditions d'application sont remplies, ou son équivalent non paramétrique si elles ne le sont pas.

3.3 Importation et mise en forme des données

Les données de cette étude sont stockées dans le fichier `HornedLizards.csv`. Importez ces données dans un objet nommé `Lizard_raw` et examinez le tableau obtenu.

```
Lizard_raw
```

```
# A tibble: 185 x 2
   squamosalHornLength Survival
                 <dbl> <chr>
 1                25.2 living
 2                26.9 living
 3                26.6 living
 4                25.6 living
 5                25.7 living
 6                25.9 living
 7                27.3 living
 8                25.1 living
 9                30.3 living
10                25.6 living
# i 175 more rows
```

```
View(Lizard_raw)
```

On constate ici 3 choses :

1. la variable `Survival` devrait être un facteur.
2. le nom de la première colonne (`squamosalHornLength`), qui contient les mesures des longueurs de cornes, est bien trop long.
3. pour l'un des lézards vivants, la mesure de longueur des cornes est manquante. Nous allons donc retirer cet individu pour éviter les messages d'erreurs par la suite.

Nous pouvons facilement réaliser les 3 modifications d'un coup, et stokcer le résultat dans un nouveau tableau `Lizard` :

```
Lizard <- Lizard_raw |>
  mutate(Survival = factor(Survival)) |>
```

```
    rename(Horn_len = squamosalHornLength) |>
    filter(!is.na(Horn_len))

  Lizard
```

```
# A tibble: 184 x 2
   Horn_len Survival
      <dbl> <fct>
 1     25.2 living
 2     26.9 living
 3     26.6 living
 4     25.6 living
 5     25.7 living
 6     25.9 living
 7     27.3 living
 8     25.1 living
 9     30.3 living
10     25.6 living
# i 174 more rows
```

Ce tableau est bien un tableau rangé, au format long : chaque colonne contient une unique variable (`Horn_len` : longueur des cornes, `Survival` : groupe de l'individu mesuré, vivant ou mort), et chaque ligne contient les informations d'un unique individu.

Ici, il ne serait pas correct de présenter les données au format large. il nous faudrait en effet une colonne pour chaque groupe, lézard vivant et lézard mort, mais puisque les données de ces 2 groupes sont indépendantes, nous aurions 2 problèmes :

1. si le nombre d'individu n'est pas le même dans les 2 groupes, les deux colonnes n'auraient pas la même longueur. C'est impossible dans `RStudio`, et le logiciel remplierait donc la colonne la plus courte de NAs pour y remédier.
2. les lignes de cet hypothétique tableau large ne correspondraient plus à des observations uniques. Chaque ligne renseignerait en effet sur les mesures de 2 individus distincts, un vivant et un mort.

Lorsque vous disposez de données appartenant à des groupes indépendants, il faut donc toujours travailler avec un tableau rangé, nécessairement au format long.

3.4 Exploration statistique des données

Comme dans le Chapitre 2 sur les données appariées, les statistiques descriptives doivent ici être réalisées pour chaque groupe d'individus, et non tous groupes confondus. Ici, le plus simple est d'utiliser la fonction `skim()` sur les données groupées par niveau du facteur `Survival` (avec la fonction `group_by()`) :

```
Lizard |>
  group_by(Survival) |>
  skim()
```

```
-- Data Summary ------------------------
                           Values
Name                       group_by(Lizard, Survival...
Number of rows             184
Number of columns          2
------------------------
Column type frequency:
  numeric                  1
------------------------
Group variables            Survival

-- Variable type: numeric -------------------------------------------------------
  skim_variable Survival n_missing complete_rate mean   sd   p0  p25  p50  p75
1 Horn_len      killed           0             1 22.0 2.71 15.2 21.1 22.2 23.8
2 Horn_len      living           0             1 24.3 2.63 13.1 23   24.6 26
  p100 hist
1 26.7
2 30.3
```

On constate ici qu'il n'y a pas de données manquantes (`n_missing` = 0 dans les deux groupes). La moyenne des longueurs de cornes est plus grande chez les lézards vivants ($\bar{x}_{living}$ = 24.3 mm) que chez les lézards retrouvés morts ($\bar{x}_{killed}$ = 22.0), de plus de 2 millimètres. On retrouve cette

tendance pour les médianes, ainsi que pour les premiers et troisièmes quartiles. En revanche, les écarts-types des 2 groupes sont proches, et celui du groupe `living` est très légèrement plus faible (0.08 mm) que celui du groupe `killed`.

Enfin, les histogrammes très simplifiés fournis laissent penser que les données de chaque groupe ne s'écartent pas trop fortement d'une courbe en cloche.

Outre ces informations sur les ordres de grandeurs observés dans chaque groupe de lézards pour les indices de **position** (moyennes, médianes et quartiles), et de **dispersion** (écarts-types et histogrammes), la fonction `skim()` ne fournit pas les effectifs observés dans chaque groupe. On sait qu'il y a en tout 184 individus, mais on ne sait pas comment ils se répartissent dans les 2 groupes de lézards. Pour le déterminer, on peut utiliser une fonction décrite plus tôt, la fonction `count()` :

```
Lizard |>
  count(Survival)
```

```
# A tibble: 2 x 2
  Survival     n
  <fct>    <int>
1 killed      30
2 living     154
```

On peut obtenir la même information avec la fonction `summarise()` et son argument `.by`, et la fonction `n()` :

```
Lizard |>
  summarize(Effectif = n(), .by = Survival)
```

```
# A tibble: 2 x 2
  Survival Effectif
  <fct>       <int>
1 living        154
2 killed         30
```

On constate ici que les tailles d'échantillons sont très différentes. C'est normal compte tenu de la difficulté de repérer

des individus morts dans la nature, et ce n'est pas gênant pour nos analyses puisque la taille des deux échantillons reste élevée.

Enfin, on peut calculer des indices d'incertitude. C'est d'autant plus important qu'il est difficile de se faire une idée de la signification d'une différence moyenne de longueur de cornes de 2 millimètres. Est-ce important ou négligeable ? Est-ce que ces estimations sont précises ou non ? Comme dans les chapitres précédents, nous allons calculer les intervalles de confiance à 95% de la moyenne de chaque groupe :

```
Lizard |>
  reframe(mean_cl_normal(Horn_len), .by = Survival)
```

```
# A tibble: 2 x 4
  Survival     y  ymin  ymax
  <fct>    <dbl> <dbl> <dbl>
1 living    24.3  23.9  24.7
2 killed    22.0  21.0  23.0
```

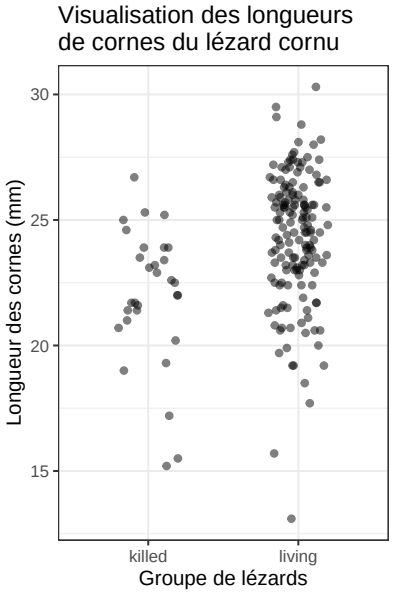
La colonne `y` nous présente à nouveau la moyenne de chaque groupe, la colonne `ymin` contient les bornes inférieures des intervalles de confiance à 95%, et la colonne `ymax` les bornes supérieures. On constate ici que les intervalles de confiance à 95% des longueurs de cornes des 2 groupes ne se chevauchent pas du tout : la borne inférieure du groupe `living` est au-dessus de la borne supérieure du groupe `killed`. Autrement dit, dans la population générale, la longueur moyenne des cornes chez les lézards vivants a de bonnes chances de se trouver dans l'intervalle [23.9 ; 24.7] millimètres, alors qu'elle a de bonnes chances de se trouver dans l'intervalle [21 ; 23] millimètres chez les lézards morts. La différence de moyennes entre ces 2 groupes vaut donc probablement entre 0.9 millimètres au moins, et 3.7 millimètres au plus. Le test statistique que nous ferons ensuite devrait donc confirmer que ces différences sont significatives, autrement dit, qu'elles ne sont pas liées au simple hasard de l'échantillonnage.

3.5 Exploration graphique des données

Comme toujours, nous pouvons réaliser plusieurs types de graphiques pour en apprendre plus sur **la distribution des données** dans les deux groupes. Si nous faisons un nuage de points, il est évidemment impossible ici de relier les points deux à deux. Non seulement cela n'aurait aucun sens puisque les échantillons sont indépendants, mais en outre, nous ne disposons pas du même nombre d'individus dans les 2 échantillons. Nous nous contenterons donc da faire un stripchart.

3.5.1 Avec un stripchart

```
Lizard |>
  ggplot(aes(x = Survival, y = Horn_len)) +
  geom_jitter(height = 0, width = 0.2, alpha = 0.5) +
  labs(x = "Groupe de lézards",
       y = "Longueur des cornes (mm)",
       title = "Visualisation des longueurs\nde cornes du lézard cornu")
```

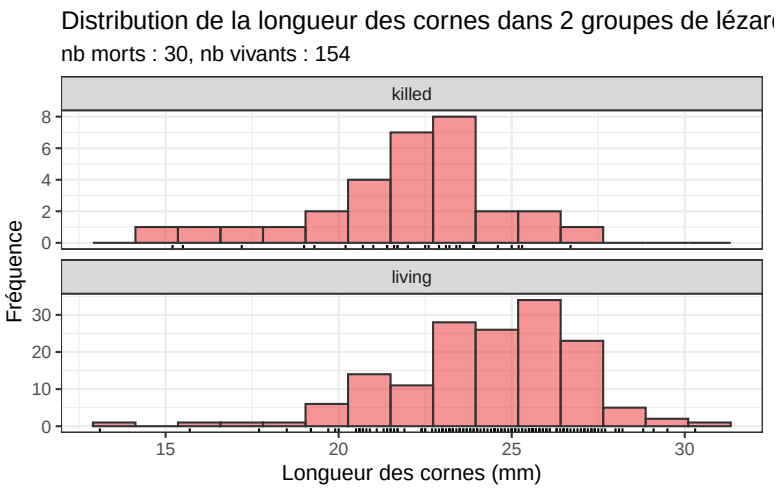


Ce premier graphique permet de visualiser très clairement les différences de tailles d'échantillons entre les deux groupes. Il permet également de voir que l'étendue des longueurs de

cornes est plus importante dans le groupe des individus vivants que dans celui des individus morts. En outre, le nuage de points des vivants semble être plus haut sur l'axe des y que celui des morts, confirmant les statistiques descriptives qui montraient des tailles de cornes en moyenne plus importantes dans le groupe des vivants.

3.5.2 Avec des histogrammes facettés

```
Lizard |>
  ggplot(aes(x = Horn_len)) +
  geom_histogram(bins = 15, fill = "firebrick2", color = "grey20", alpha = 0.5)+
  geom_rug() +
  facet_wrap(~Survival, ncol = 1, scales = "free_y") +
  labs(x = "Longueur des cornes (mm)",
       y = "Fréquence",
       title = "Distribution de la longueur des cornes dans 2 groupes de lézards cornus",
       subtitle = "nb morts : 30, nb vivants : 154")
```

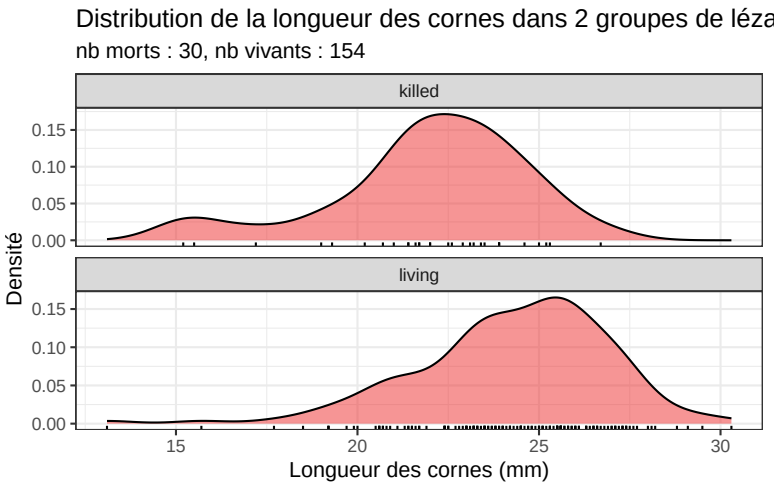


Notez ici l'utilisation de l'argument `scales = "free_y"` dans la fonction `facet_wrap()`. Cet argument permet de ne pas imposer la même échelle pour l'axe des ordonnées des 2 graphiques. Ce choix est ici pertinent puisque les effectifs des 2 groupes sont très différents. Faîtes un essai sans cet argument pour voir la différence. Il est en revanche important de conserver le même axe des `x` afin de faciliter la comparaison des 2 groupes.

Cette visualisation nous montre que les données doivent suivre à peu près une distribution Normale dans les 2 groupes, et que globalement la longueur des cornes semble légèrement plus élevée dans le groupe des vivants (avec un mode autour de 25-26 mm) que dans le groupes des morts (avec un mode autour de 23-24 mm). L'étendue des données semble légèrement plus grande dans le groupe des vivants, mais cela n'est peut-être dû qu'à la différence marquée des tailles d'échantillons.

3.5.3 Avec des diagrammes de densité facettés

```
Lizard |>
  ggplot(aes(x = Horn_len)) +
  geom_density(fill = "firebrick2", alpha = 0.5) +
  geom_rug() +
  facet_wrap(~Survival, ncol = 1, scales = "free_y") +
  labs(x = "Longueur des cornes (mm)",
       y = "Densité",
       title = "Distribution de la longueur des cornes dans 2 groupes de lézards cornus",
       subtitle = "nb morts : 30, nb vivants : 154")
```



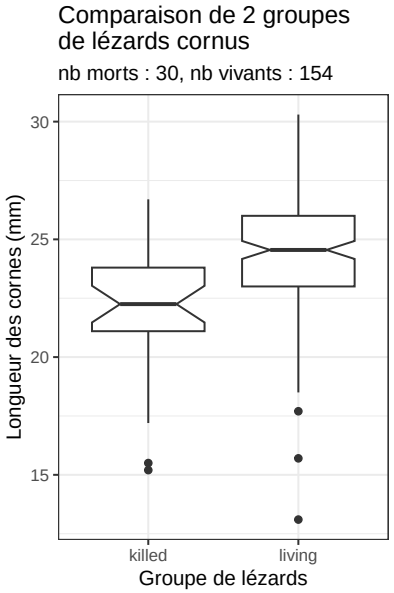
Les diagrammes de densité ressemblent ici beaucoup aux histogrammes. C'est normal car la taille des échantillons est importante (30 et 154 pour les groupes `killed` et `living` respectivement). C'était moins vrais dans les chapitres précédents car les tailles d'échantillons étaient plus faibles, et la forme des histogrammes dépendait alors beaucoup du nombre

de classes que l'on choisissait de représenter. Avec des échantillons de grande taille (n 30), c'est moins problématique.

En général, il est donc inutile de faire à la fois les histogrammes et les diagrammes de densité. Choisissez l'un ou l'autre selon vos préférences et la situation.

3.5.4 Avec des boites à moustaches

```
Lizard |>
  ggplot(aes(x = Survival, y = Horn_len)) +
  geom_boxplot(notch = TRUE) +
  labs(x = "Groupe de lézards",
       y = "Longueur des cornes (mm)",
       title = "Comparaison de 2 groupes\nde lézards cornus",
       subtitle = "nb morts : 30, nb vivants : 154")
```

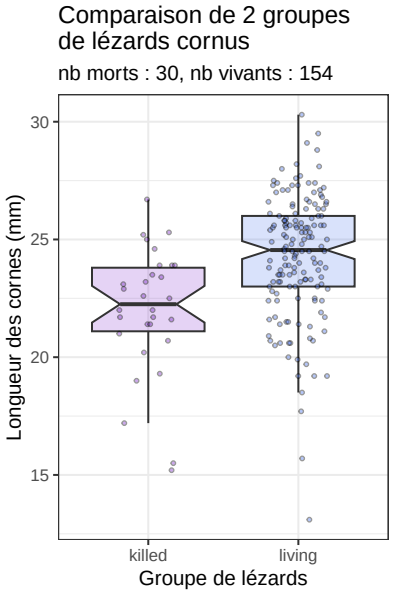


Nous visualisons ici encore plus clairement que sur les histogrammes le fait que les longueurs de cornes des individus vivants sont légèrement plus longues que celles des individus morts. D'ailleurs, puisque les intervalles de confiance à 95% des médianes des 2 groupes (les encoches) ne se chevauchent pas, un test de comparaison des moyennes devrait logiquement conclure à une différence significative en faveur des individus vivants. On peut également noter que la largeur de

l'encoche pour les individus morts est plus importante que celle des vivants. Cela traduit une incertitude plus grande autour de la médiane estimée dans le groupe des individus morts. C'est tout à fait logique compte tenu des effectifs plus faibles dans ce groupe.

Enfin, il est tout à fait possible de représenter sur le même graphique les boîtes à moustaches et les données brutes sous forme de stripchart. On a ainsi à la fois (i) une visualisation simplifiée de la position et de la dispersion des données avec les boîtes à moustache, et (ii) accès à l'ensemble des données brutes, ce qui permet parfois de voir des structures invisibles sur les boîtes à moustaches (regroupement de points par exemples). Afin de ne pas dupliquer les valeurs les plus extrêmes du jeu de données, nous indiquerons à `geom_boxplot()` de ne pas afficher les outliers sur le graphique : tous les points seront en effet déjà affichés par `geom_jitter()` :

```
Lizard |>
  ggplot(aes(x = Survival, y = Horn_len, fill = Survival)) +
  geom_boxplot(notch = TRUE, color = "grey20", alpha = 0.2,
               outlier.color = NA, show.legend = FALSE) +
  geom_jitter(height = 0, width = 0.2, alpha = 0.4, shape = 21,
              show.legend = FALSE, size = 0.8) +
  labs(x = "Groupe de lézards",
       y = "Longueur des cornes (mm)",
       title = "Comparaison de 2 groupes\nde lézards cornus",
       subtitle = "nb morts : 30, nb vivants : 154") +
  scale_fill_manual(values = c("purple3", "royalblue2"))
```



3.6 Le test paramétrique

Le test paramétrique le plus puissant que nous puissions faire pour comparer la moyenne de 2 populations est le test de Student. Ce test étant paramétrique, nous devons nous assurer que ses conditions d'application sont vérifiées avant de pouvoir le réaliser.

3.6.1 Conditions d'application

Les conditions d'application de ce test sont au nombre de 3 :

1. Chacun des deux échantillons est issu d'un échantillonnage aléatoire de la population générale. Comme toujours, en l'absence d'indication contraire, on considère que cette condition est toujours vérifiée.
2. La variable numérique étudiée est distribuée normalement dans les deux populations. Il nous faudra donc faire deux tests de Shapiro-Wilk, un pour chaque échantillon.
3. La variance de la variable numérique étudiée est la même dans les deux populations. C'est ce que l'on appelle l'homoscédasticité.

En réalité, le test du t de Student sur deux échantillons indépendants est assez robuste face au non respect de cette troisième condition d'application. Cela signifie que si cette troisième condition d'application n'est pas strictement vérifiée, les résultats du tests peuvent malgré tout rester valides. Lorsque les 2 échantillons comparés ont des tailles supérieures ou égales à 30, ce test fonctionne bien même si l'écart-type d'un groupe est jusqu'à 3 fois supérieur ou inférieur à l'écart-type du second groupe, à condition que la taille des 2 échantillons soit proche (ce qui n'est pas le cas ici !). En revanche, si les écart-types diffèrent de plus d'un facteur 3, ou si les tailles d'échantillons sont très différentes, le test du t de Student ne devrait pas être utilisé. De même, si la taille des échantillons est inférieure à 30 et que les variances ne sont pas homogènes, ce test ne devrait pas être réalisé. En conclusion, les résultats du test du t de Student à deux échantillons indépendants peuvent rester valides si la troisième condition d'homoscédasticité n'est pas respectée, mais dans certains cas seulement.

Le test du t de Student sur deux échantillons indépendants est également assez robuste face à des écarts mineurs à la distribution Normale, tant que la forme des deux distributions comparées reste similaire et unimodale. En outre, la robustesse de ce test augmente avec la taille des échantillons.

> **Robustesse**
>
> La robustesse d'un tests statistique est sa capacité à rester valide même lorsque certaines de ses conditions d'application ne sont pas parfaitement respectées. Plus un test est robuste, plus il est capable de supporter des "entorses" importantes à ses conditions d'application.}

Au final, avec un peu d'habitude, même lorsque les conditions d'application ne sont pas toutes vérifiées, on peut parfois passer outre. Mais à ce stade, on préfère s'en tenir à des choses plus simples et claires.

> **La procédure à suivre**
>
> 1. Faites un test de Normalité pour chacune des deux séries de données. Si elles suivent la loi Normale

toutes les deux, passez au point 2. Sinon, rendez-vous au point 4.
2. Faites un test d'homoscédasticité (homogénéité des variances). Si les variances sont homogènes, passez au point 3. Sinon,rendez-vous au point 5.
3. Faite un test de comparaison des moyennes paramétrique : le test de Student. Examinez la $p-$value pour conclure, et rendez-vous au point 6.
4. Faite un test de comparaison des moyennes non paramétrique : le test de Wilcoxon de la somme des rangs. Examinez la $p-$value pour conclure, et rendez-vous au point 6.
5. Faite un test de comparaison des moyennes non paramétrique : le test t de Welch. Examinez la $p-$value pour conclure, et rendez-vous au point 6.
6. Si la $p-$value est supérieure à α, on ne peut pas rejeter l'hypothèse nulle et on conclut alors à une absence de différence significative entre les 2 populations. À l'inverse, si la $p-$value est inférieure ou égale à α, on rejette l'hypothèse nulle et on valide l'hypothèse alternative. Les deux populations ont des moyennes significativement différentes, et pour savoir laquelle est supérieure ou inférieure à l'autre, on revient aux estimations des moyennes et des intervalles de confiances à 95%, calculés dans la partie consacrée aux statistiques descriptives.

3.6.1.1 Normalité des données

Nous commençons donc par tester la Normalité des 2 populations dont sont issus les échantillons, c'est le point 1 de la procédure détaillée ci-dessus. Pour les individus morts, les hypothèses sont les suivantes :

- H_0 : la taille des cornes suit une distribution Normale dans la population des lézards cornus morts.
- H_1 : la taille des cornes ne suit pas une distribution Normale dans la population des lézards cornus morts.

```
Lizard |>
  filter(Survival == "killed") |>
  pull(Horn_len) |>
  shapiro.test()
```

```
    Shapiro-Wilk normality test

data:  pull(filter(Lizard, Survival == "killed"), Horn_len)
W = 0.93452, p-value = 0.06482
```

La p-value est supérieure à $\alpha = 0.05$, donc on ne peut pas rejeter l'hypothèse nulle de normalité pour la taille des cornes de la population des lézards cornus morts (test de Shapiro-Wilk, $W = 0.93$, $p = 0.065$).

Pour les individus vivants :

- H_0 : la taille des cornes suit une distribution Normale dans la population des lézards cornus vivants.
- H_1 : la taille des cornes ne suit pas une distribution Normale dans la population des lézards cornus vivants.

```
Lizard |>
  filter(Survival == "living") |>
  pull(Horn_len) |>
  shapiro.test()
```

```
    Shapiro-Wilk normality test

data:  pull(filter(Lizard, Survival == "living"), Horn_len)
W = 0.96055, p-value = 0.0002234
```

La p-value est inférieure à $\alpha = 0.05$, donc on rejette l'hypothèse nulle de normalité pour la taille des cornes de la population des lézards cornus vivants (test de Shapiro-Wilk, $W = 0.96$, $p < 0.001$).

L'une des 2 séries de données ne suit pas la loi Normale, nous sommes donc censés passer directement au point 4 de la procédure.

Toutefois, si l'on examine les histogrammes (Section 3.5.2) ou les diagrammes de densité (Section 3.5.3) des 2 échantillons, on constate que la forme des distributions des 2 séries de données est très proche. Pour les 2 échantillons, la distribution est en effet uni-modale, avec une asymétrie gauche assez marquée (une longue queue de distribution du côté gauche). La forme des distributions étant similaire (on parle bien de la forme des histogrammes et non de la position du pic), et les histogrammes étant proches de la forme typique d'une courbe en cloche, le test de Student devrait rester valide car il est robuste dans cette situation. Ici, pour l'exemple, on va donc passer au point 2 de la procédure. Notez toutefois que passer directement au point 4 de la procédure serait tout à fait correct : on ne pourrait rien vous reprocher si vous passez directement au test non paramétrique de comparaison des moyennes lorsque vous constatez que l'une des 2 séries de données ne suit pas la distribution Normale.

3.6.1.2 Homogénéité des variances

Le test le plus simple pour comparer la variance de 2 populations est le test F :

- H_0 : la variance des 2 populations est égale, leur ratio vaut 1 $\left(\frac{\sigma^2_{killed}}{\sigma^2_{living}} = 1\right)$.
- H_1 : la variance des 2 populations est différente, leur ratio ne vaut pas 1 $\left(\frac{\sigma^2_{killed}}{\sigma^2_{living}} \neq 1\right)$.

```
var.test(Horn_len ~ Survival, data = Lizard)
```

```
    F test to compare two variances

data:  Horn_len by Survival
F = 1.0607, num df = 29, denom df = 153, p-value = 0.7859
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.6339331 1.9831398
```

```
sample estimates:
ratio of variances
          1.060711
```

Ici, le ratio des variances (la variance des individus morts divisée par la variance des individus vivants) est très proche de 1 ($F = 1.06$, IC 95% : [0.63 ; 1.98]). Le test F nous montre qu'il est impossible de rejeter H_0 : au seuil $\alpha = 0.05$, le ratio des variances n'est pas significativement différent de 1 (ddl = 29 et 153, $p = 0.79$), les variances sont homogènes.

Le test de Bartlett est un autre test qui permet de comparer la variance de plusieurs populations (2 ou plus). Lorsque le nombre de populations est égal à 2 (comme ici), ce test est absolument équivalent au test F ci-dessus.

- H_0 : toutes les populations ont même variance ($\sigma_A^2 = \sigma_B^2 = \sigma_C^2 = \cdots = \sigma_N^2$).
- H_1 : au moins une population a une variance différente des autres.

```
bartlett.test(Horn_len ~ Survival, data = Lizard)
```

```
    Bartlett test of homogeneity of variances

data:  Horn_len by Survival
Bartlett's K-squared = 0.042411, df = 1, p-value = 0.8368
```

Enfin, le test de Levene (attention, le package `car` doit être chargé) devrait être préféré la plupart du temps. Comme le test de Bartlett, il permet de comparer la variance de plusieurs populations, mais il est plus robuste vis à vis de la non-normalité des données.

- H_0 : toutes les populations ont même variance ($\sigma_A^2 = \sigma_B^2 = \sigma_C^2 = \cdots = \sigma_N^2$).
- H_1 : au moins une population a une variance différente des autres.

```r
# Le test de Levene fait partie du package car. Il doit être chargé en mémoire
# library(car)
leveneTest(Horn_len ~ Survival, data = Lizard)
```

```
Levene's Test for Homogeneity of Variance (center = median)
       Df F value Pr(>F)
group   1  0.0035  0.953
      182
```

Ici encore, les conclusions sont les mêmes :

> Il est impossible de rejeter l'hypothèse nulle d'homogénéité des variances au seuil $\alpha = 0.05$ (test de Levene, $F = 0.004$, ddl $= 1$, $p = 0.953$).

À vous de choisir lequel de ces 3 tests vous souhaitez réaliser : il est évident qu'on ne fait jamais les 3 !

Ici, puisque l'homoscédasticité est vérifiée, on passe au point 3 de la procédure.

3.6.2 Réalisation du test et interprétation

Puisque la taille des cornes du lézard cornu suit approximativement la même distribution "presque Normale" dans les 2 populations (lézards morts et vivants) et que ces 2 populations ont des variances homogènes, on peut réaliser le test du t de Student sur deux échantillons indépendants.

- H_0 : la moyenne des 2 populations est égale, leur différence vaut 0 ($\mu_{killed} - \mu_{living} = 0$).
- H_1 : la moyenne des 2 populations est différente, leur différence ne vaut pas 0 ($\mu_{killed} - \mu_{living} \neq 0$).

```r
t.test(Horn_len ~ Survival, data = Lizard, var.equal = TRUE)
```

```
    Two Sample t-test

data:  Horn_len by Survival
t = -4.3494, df = 182, p-value = 2.27e-05
```

```
alternative hypothesis: true difference in means between group killed and group living is no
95 percent confidence interval:
 -3.335402 -1.253602
sample estimates:
mean in group killed mean in group living
            21.98667             24.28117
```

Notez bien la syntaxe :

- Nous utilisons ici la syntaxe du type "formule" faisant appel au symbole "`~`" (Longueur des cornes en fonction de la Survie) et à l'argument "`data =`".
- L'argument "`paired = TRUE`" a disparu puisque nous avons ici deux échantillons indépendants
- L'argument "`var.equal = TRUE`" doit obligatoirement être spécifié : nous nous sommes assuré que l'homogénéité des variances était vérifiée. Il faut donc l'indiquer afin que le test de Student classique soit réalisé. Si on omet de le spécifier, c'est un autre test qui est réalisé (voir plus bas).

  Au seuil α de 5%, on rejette l'hypothèse nulle d'égalité des moyennes de la longueur des cornes entre lézards vivants et morts (test t de Student sur deux échantillons indépendant, $t = -4.35$, ddl = 182, $p < 0.001$). Les lézards morts ont en moyenne des cornes plus courtes ($\widehat{\mu}_{killed} = 21.99$ millimètres) que les lézards vivants ($\widehat{\mu}_{living} = 24.28$ millimètres). La gamme des valeurs les plus probables pour la différence de moyenne entre les deux populations est fournie par l'intervalle de confiance à 95% de la différence de moyennes : [-3.34 ; -1.25].

Ce test confirme donc bien l'impression des chercheurs : les lézards principalement pris pour cibles par les pies grièches migratrices ont des cornes en moyenne plus courtes (probablement entre 1.25 et 3.34 millimètres de moins) que les lézards de la population générale. Avoir des cornes plus longues semble donc protéger les lézards de la prédation, du moins dans une certaine mesure.

Notez bien que l'intervalle de confiance à 95% qui est fourni avec les résultats du test est l'intervalle de confiance à 95%

de la différence de moyenne entre les 2 groupes. Cet intervalle nous donne donc une idée de la **magnitude de l'effet**, de son ampleur. En effet, dire que les lézards morts ont des cornes en moyenne plus courtes est intéressant, mais cela n'aura pas la même portée si leurs cornes sont plus courtes de 0.02 millimètres ou si elles sont plus courtes de 5 millimètres. Un test statistique permet de rejeter ou non une hypothèse nulle, mais c'est bien l'estimation (la moyenne de chaque groupe et l'intervalle de confiance à 95% de la différence) qui nous dit ce qu'on doit penser des résultats, et de leur pertinence (écologique, biologique, physiologie, comportementale, etc.).

3.7 L'alternative non paramétrique

Si les conditions d'application du test de Student ne sont pas vérifiées (c'est bien le cas ici puisque la longueur des cornes ne suit pas une distribution Normale dans la population des lézards vivants), notre procédure nous conduit à l'étape 4 : nous devons utiliser un équivalent non paramétrique au test de Student. C'est le cas du **test de Wilcoxon sur la somme des rangs** (également appelé test de Mann-Whitney). Comme pour tous les tests de Wilcoxon, la comparaison porte alors non plus sur les moyennes mais sur les médianes.

- H_0 : la médiane des 2 populations est égale, leur différence vaut 0 ($med_{killed} - med_{living} = 0$).
- H_1 : la médiane des 2 populations est différente, leur différence ne vaut pas 0 ($med_{killed} - med_{living} \neq 0$).

```
wilcox.test(Horn_len ~ Survival, data = Lizard, conf.int = TRUE)
```

```
    Wilcoxon rank sum test with continuity correction

data:  Horn_len by Survival
W = 1181.5, p-value = 2.366e-05
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -3.200076 -1.300067
sample estimates:
```

```
difference in location
             -2.200031
```

L'argument `var.equal = TRUE` n'existe pas pour ce test, puisque c'est justement un test non paramétrique qui ne requiert pas l'homogénéité des variances. En revanche, comme pour tous les autres tests de Wilcoxon que nous avons réalisés jusqu'ici, l'argument `conf.int = TRUE` permet d'afficher les estimateurs pertinents, ici, la différence de médiane entre les 2 populations et l'intervalle de confiance à 95% de cette différence de médiane.

La conclusion est ici la même que pour le test de Student : puisque la p-value est très inférieure à α, on rejette l'hypothèse nulle : les médianes sont bel et bien différentes.

3.8 L'autre alternative non paramétrique

Enfin, dans le cas où la variable étudiée suit la loi Normale dans les deux populations mais qu'elle n'a pas la même variance dans les deux populations (donc si vous arrivez au point 5 de la procédure décrite plus haut), il est toujours possible de réaliser un test de Wilcoxon, il est préférable de réaliser un test de Student modifié : le **test approché du t de Welch**. Ce test est moins puissant que le test de Student classique, mais il reste plus puissant que le test de Wilcoxon, et surtout, il permet de comparer les moyennes et non les médianes.

- H_0 : la moyenne des 2 populations est égale, leur différence vaut 0 ($\mu_{killed} - \mu_{living} = 0$).
- H_1 : la moyenne des 2 populations est différente, leur différence ne vaut pas 0 ($\mu_{killed} - \mu_{living} \neq 0$).

```
t.test(Horn_len ~ Survival, data = Lizard)
```

```
    Welch Two Sample t-test

data:  Horn_len by Survival
```

```
t = -4.2634, df = 40.372, p-value = 0.0001178
alternative hypothesis: true difference in means between group killed and group living is no
95 percent confidence interval:
 -3.381912 -1.207092
sample estimates:
mean in group killed mean in group living
            21.98667             24.28117
```

La seule différence par rapport à la syntaxe du test t de Student paramétrique est la suppression de l'argument `var.equal = TRUE`. Attention donc, à bien utiliser la syntaxe correcte. Le test du t de Welch ne devrait être réalisé que lorsque la Normalité est vérifiée pour les 2 populations, mais pas l'homoscédasticité. Par rapport au test de Student classique, on constate que le nombre de degrés de libertés est très différent, et donc la p-value également. Les bornes de l'intervalle de confiance à 95% de la différence de moyenne sont différentes également puisque leur calcul a été fait en supposant que les 2 populations n'avaient pas même variance.

3.9 Exercices d'application

3.9.1 La taille des hommes et des femmes

On s'intéresse à la différence de taille supposée entre hommes et femmes. Le fichier `HommesFemmes.xls` contient les tailles en centimètres de 38 hommes et 43 femmes choisis au hasard parmi les étudiants de première année à La Rochelle Université. Importez, mettez en forme et analysez ces données. Vous prendrez soin de retirer les éventuelles valeurs manquantes, vous prendrez le temps d'examiner les données à l'aide de statistiques descriptives et de représentations graphiques adaptées, puis vous tenterez de répondre à la question suivante : les hommes et les femmes inscrits en première année à La Rochelle Université ont-il la même taille ? Si non, caractérisez cette différence de taille.

3.9.2 La longueur du bec des manchots Adélie

Dans le jeu de données `penguins` du package `palmerpenguins`, récupérez les lignes du tableau qui correspondent aux manchots Adélie, et comparez la longueur des becs entre mâles et femelles. Comme toujours, avant de vous lancer dans les tests, vous prendrez soin de retirer les éventuelles valeurs manquantes, et vous prendrez le temps d'examiner les données à l'aide de statistiques descriptives et de représentations graphiques adaptées. Faites l'effort d'expliquer votre démarche, de préciser les hypothèses nulles et alternatives de chaque test, et de rédiger l'interprétation que vous faites de chaque résultat.

3.10 Tests bilatéraux et unilatéraux

3.10.1 Principe

Jusqu'à maintenant, tous les tests que nous avons réalisés sont des **tests bilatéraux**. Pour chaque test, l'hypothèse nulle est imposée. En revanche, pour certains tests, l'hypothèse alternative est à choisir (et à spécifier) par l'utilisateur parmi 3 possibilités :

- Une hypothèse bilatérale. C'est celle qui est utilisée par défaut si l'utilisateur ne précise rien.
- Deux hypothèses unilatérales possibles, qui doivent être spécifiées explicitement par l'utilisateur.

Les tests unilatéraux peuvent concerner tous les tests pour lesquels les hypothèses sont de la forme suivante :

- H_0 : la valeur d'un paramètre de la population est égale à k (k peut être une valeur fixe, arbitraire, choisie par l'utilisateur, ou la valeur d'un paramètre d'une autre populations).
- H_1 : la valeur d'un paramètre de la population **n'est pas égale à** k.

En réalité, si nous remplaçons l'hypothèse H_1 par :

- H_1 : la valeur d'un paramètre de la population **est supérieure à** k.

ou par :

- H_1 : la valeur d'un paramètre de la population **est inférieure à** k.

nous réalisons un test unilatéral.

Dans `RStudio`, la syntaxe permettant de spécifier l'hypothèse alternative que nous souhaitons utiliser est toujours la même. Il faut préciser, au moment de faire le test l'argument suivant :

- `alternative = "two.sided"` : pour faire un test bilatéral. Si on ne le fait pas explicitement, c'est de toutes façons cette valeur qui est utilisée par défaut.
- `alternative = "greater"` : pour choisir l'hypothèse unilatérale "`>`".
- `alternative = "less"` : pour choisir l'hypothèse unilatérale "`<`".

Attention : le choix d'utiliser "greater" ou "less" dépend donc de l'ordre dans lequel les échantillons sont spécifiés. Cette syntaxe est valable pour tous les tests de Student vus jusqu'ici (un échantillon, deux échantillons appariés, deux échantillons indépendants) et pour leurs alternatives non paramétriques (test de Wilcoxon des rangs signés, test de Wilcoxon de la somme des rangs, test du t de Welch).

> **!** Attention
>
> L'utilisation de tests unilatéraux doit être réservée exclusivement aux situations pour lesquelles le choix de l'hypothèse unilatérale est possible à justifier par un mécanisme quelconque (biologique, physiologique, comportemental, écologique, génétique, évolutif, biochimique, etc.). Observer que l'un des échantillons a une moyenne plus grande ou plus faible qu'un autre lors de la phase des statistiques descriptives des données **n'est pas du tout une raison suffisante**. Il faut pouvoir justifier le choix de l'hypothèse alternative par une explication valable. D'ailleurs, si on veut être rigoureux, il faudrait toujours formuler les hypothèses que l'on souhaite tester **avant** de mettre en place le protocole expérimental et **avant** d'acquérir les données.

Pour s'embêter avec les tests unilatéraux puisqu'il est si rare qu'on ait le droit de les faire ? Tout simplement parce que toutes choses étant égales par ailleurs, un test unilatéral est toujours plus puissant (parfois, beaucoup plus puissant) qu'un test bilatéral. Or, la puissance est quelque chose qu'on cherche à maximiser (voir sec-puiss). Lorsqu'il est pertinent de réaliser un test unilatéral, on doit donc toujours le faire.

Reprenons l'un des exemples examinés précédemment pour mieux comprendre comment tout cela fonctionne.

3.10.2 Un exemple pas à pas

Reprenons l'exemple des lézards cornus. L'étude a été réalisée parce que les chercheurs supposaient que la longueur des cornes des lézards était susceptible de leur fournir une protection face à la prédation. Autrement dit, les chercheurs supposaient que des cornes *plus longues* devaient fournir une meilleure protection vis à vis de la prédation. Ainsi, les lézards morts devaient avoir des cornes moins longues en moyenne que les les lézards vivants, simplement parce que porter des cornes courtes expose plus fortement les individus à la prédation. Nous avons donc une bonne raison "écologique/évolutive" de considérer un test unilatéral (la susceptibilité face à la prédation qui a entraîné une pression de sélection sur la longueur des cornes des lézards), avant même de collecter les données.

Lorsque nous avons examiné cette question, nous avons fait le test du t de Student sur échantillons indépendants de la façon suivante :

```
t.test(Horn_len ~ Survival, data = Lizard, var.equal = TRUE)
```

```
    Two Sample t-test

data:  Horn_len by Survival
t = -4.3494, df = 182, p-value = 2.27e-05
alternative hypothesis: true difference in means between group killed and group living is no
95 percent confidence interval:
 -3.335402 -1.253602
sample estimates:
```

```
mean in group killed mean in group living
            21.98667              24.28117
```

Comme l'indiquent les résultats fournis, l'hypothèse alternative utilisée pour faire le test est : "La vraie différence de moyenne n'est pas égale à 0". Autrement dit, nous avons fait un test bilatéral avec les hypothèses suivantes :

- H_0 : la moyenne des 2 populations est égale, leur différence vaut 0 ($\mu_{killed} - \mu_{living} = 0$).
- H_1 : la moyenne des 2 populations est différente, leur différence ne vaut pas 0 ($\mu_{killed} - \mu_{living} \neq 0$).

Ce test est donc rigoureusement équivalent à celui-ci :

```
t.test(Horn_len ~ Survival, data = Lizard, var.equal = TRUE,
       alternative = "two.sided")
```

```
    Two Sample t-test

data:  Horn_len by Survival
t = -4.3494, df = 182, p-value = 2.27e-05
alternative hypothesis: true difference in means between group killed and group living is no
95 percent confidence interval:
 -3.335402 -1.253602
sample estimates:
mean in group killed mean in group living
            21.98667              24.28117
```

Ici, nous souhaitons en fait réaliser un **test unilatéral** avec les hypothèses suivantes :

- H_0 : la moyenne de longueur des cornes de la population des lézards morts est égale à celle des lézards vivants. Leur différence vaut 0 ($\mu_{killed} - \mu_{living} = 0$).
- H_1 : la moyenne de longueur des cornes de la population des lézards morts est **inférieure** à celle des lézards vivants. Leur différence est inférieure à 0 ($\mu_{killed} - \mu_{living} < 0$).

```
t.test(Horn_len ~ Survival, data = Lizard, var.equal = TRUE,
       alternative = "less")
```

```
    Two Sample t-test

data:  Horn_len by Survival
t = -4.3494, df = 182, p-value = 1.135e-05
alternative hypothesis: true difference in means between group killed and group living is le
95 percent confidence interval:
      -Inf -1.422321
sample estimates:
mean in group killed mean in group living
            21.98667             24.28117
```

Puisque la p-value de ce test est inférieure à $\alpha = 0.05$, on rejette l'hypothèse nulle de l'égalité des moyennes. On valide donc l'hypothèse alternative : les lézards cornus morts ont en moyenne des cornes plus courtes que les lézards vivants. Cette différence de longueur de cornes est en faveur des lézards vivants et vaut très probablement au moins 1.4 millimètres (c'est l'intervalle de confiance à 95% de la différence de moyennes qui nous le dit).

Dernière chose importante : il ne faut pas se tromper dans le choix de l'hypothèse alternative. En effet, nous aurions pu tenter de tester exactement la même chose en formulant les hypothèses suivantes :

- H_0 : la moyenne de longueur des cornes de la population des lézards **vivants** est égale à celle des lézards **morts**. Leur différence vaut 0 ($\mu_{living} - \mu_{killed} = 0$).
- H_1 : la moyenne de longueur des cornes de la population des lézards **vivants** est **supérieure** à celle des lézards **morts**. Leur différence est **supérieure** à 0 ($\mu_{living} - \mu_{killed} > 0$).

Ce test est normalement exactement le même que précédemment. Toutefois, si on essaie de le réaliser, on rencontre un problème :

```r
t.test(Horn_len ~ Survival, data = Lizard, var.equal = TRUE,
       alternative = "greater")
```

```
    Two Sample t-test
```

```
data:  Horn_len by Survival
t = -4.3494, df = 182, p-value = 1
alternative hypothesis: true difference in means between group killed and group living is gr
95 percent confidence interval:
 -3.166684       Inf
sample estimates:
mean in group killed mean in group living
            21.98667             24.28117
```

Ici, la p-value est très supérieure à α puisqu'elle vaut 1. Une p-value de 1 devrait toujours attirer votre attention. La conclusion devrait donc être que l'on ne peut pas rejeter H_0 : les lézards morts et vivants ont en moyenne des cornes de même longueur. Nous savons pourtant que c'est faux.

Le problème est ici liè à l'ordre des catégories "vivant" ou "mort" dans le facteur `Survival` du tableau `Lizard`. Les dernières lignes des tests que nous venons de faire indiquent la moyenne de chaque groupe, mais le groupe "killed" apparaît toujours avant le groupe "living". C'est l'ordre des niveaux dans le facteur `Survival` qui doit dicter la syntaxe appropriée :

```
Lizard$Survival
```

```
  [1] living living living living living living living living living living
 [11] living living living living living living living living living living
 [21] living living living living living living living living living living
 [31] living living living living living living living living living living
 [41] living living living living living living living living living living
 [51] living living living living living living living living living living
 [61] living living living living living living living living living living
 [71] living living living living living living living living living living
 [81] living living living living living living living living living living
 [91] living living living living living living living living living living
[101] living living living living living living living living living living
[111] living living living living living living living living living living
[121] living living living living living living living living living living
[131] living living living living living living living living living living
[141] living living living living living living living living living living
[151] living living living living killed killed killed killed killed killed
[161] killed killed killed killed killed killed killed killed killed killed
```

```
[171] killed killed killed killed killed killed killed killed killed killed
[181] killed killed killed killed
Levels: killed living
```

Par défaut, dans `RStudio`, les niveaux d'un facteur sont classés par ordre alphabétique sauf si on spécifie manuellement un ordre différent. Ici, le niveau "killed" est donc le premier niveau du facteur, et "living" le second. Attention, on parle bien ici des niveaux, ou modalités, et non des données elles-mêmes. Ici, le premier lézard mesuré appartient à la catégorie `living`. Ça n'est pas ça qui est important : c'est bien l'ordre des niveaux qui compte, et on peut le vour tout en bas, après `Levels: ...`. Lorsque l'on réalise un test de Student avec ces données (ou un test de Wilcoxon d'ailleurs), la différence de moyenne qui est examinée par le test est donc "moyenne des `killed` - moyenne des `living`". Lorsque nous avons tapé ceci :

```
t.test(Horn_len ~ Survival, data = Lizard, var.equal = TRUE,
       alternative = "greater")
```

nous avons donc en réalité posé les hypothèses suivantes :

- H_0 : la moyenne de longueur des cornes de la population des lézards **morts** est égale à celle des lézards **vivants**. Leur différence vaut 0 ($\mu_{killed} - \mu_{living} = 0$).
- H_1 : la moyenne de longueur des cornes de la population des lézards **morts** est **supérieure** à celle des lézards **vivants**. Leur différence est **supérieure** à 0 ($\mu_{killed} - \mu_{living} > 0$).

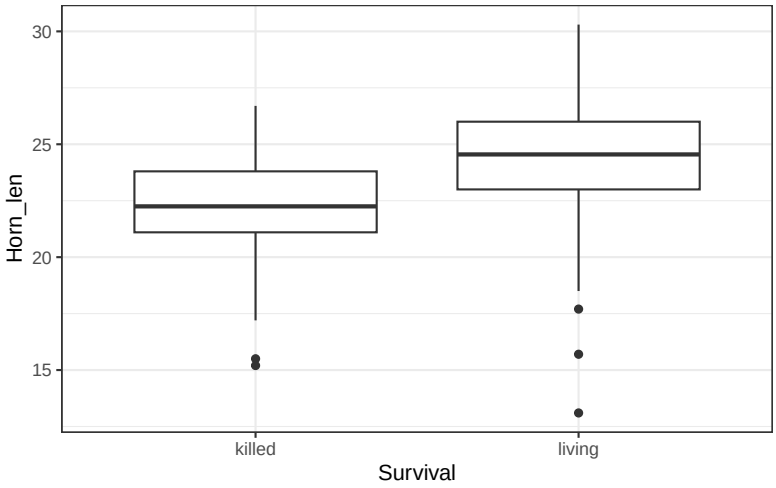
Ce test est donc erroné, ce qui explique qu'il nous renvoie un résultat faux et une p-value de 1. Ici, puisque l'ordre des catégories est "killed" d'abord et "living" ensuite, la seule façon correcte de faire un test unilatéral qui a du sens est donc celle que nous avons réalisée en premier :

```
t.test(Horn_len ~ Survival, data = Lizard, var.equal = TRUE,
       alternative = "less")
```

Faites donc toujours attention à l'ordre des catégories de vos facteurs pour ne pas vous tromper. Une façon simple de vérifier cet ordre et d'observer vos graphiques (par exemple, les

boîtes à moustaches). L'ordre dans lequel les catégories apparaissent sur l'axe des `x` reflète l'ordre des catégorie du facteur porté par cet axe :

```
Lizard |>
  ggplot(aes(x = Survival, y = Horn_len)) +
  geom_boxplot()
```



3.10.3 Exercice d'application

Reprenez chaque exemple et exercice traité depuis le premier chapitre et identifiez les situations où un test unilatéral aurait du sens. Si vous en trouvez, faites ce test et assurez-vous que les hypothèses choisies sont bien celles qui sont utilisées lors du test.

4 Analyse de cohortes

4.1 Objectifs

Cette section doit permettre d'illustrer la partie du cours de Population Dynamics de l'EC "Fonctionnement des écosystèmes" consacrée à l'analyse des cohortes. Vous utiliserez les tailles corporelles d'individus échantillonnés sur le terrain à plusieurs dates pour :

1. Produire la structure démographique instantanée de la population pour chaque date d'échantillonnage
2. Réaliser la décomposition polymodale pour identifier les cohortes à chaque date d'échantillonnage
3. Créer une courbe de croissance et une courbe de mortalité
4. Utiliser une relation allométrique pour passer de la taille des individus à leur masse
5. Produire la courbe d'Allen

Vous devrez en premier lieu importer les données fournies dans un fichier Excel et vous assurer qu'elles sont dans un format permettant les analyses et représentations graphiques.

4.2 Présentation de l'étude

Le suivi démographique des populations animales est extrêmement fréquent dans le domaine de l'écologie. Le suivi des populations naturelles est particulièrement pertinent dans un contexte de conservation, pour réaliser des études d'impact ou pour mesurer l'évolution de la biodiversité.

Ici, une population de gastéropodes *Nassarius reticulatus* (la nasse réticulée) a été suivie pendant 5 ans. Deux sessions d'échantillonnage ont été organisées chaque année depuis 2010 : la première a été réalisée en mars, juste après le recrutement des juvéniles, et la seconde a été réalisée 6

mois plus tard, en septembre. Tous les échantillons ont été collectés au même endroit, selon la même méthode (collecte systématique de tous les individus présents à l'intérieur de 10 quadrats positionnés aléatoirement dans la zone d'étude), lors d'une marée basse de fort coefficient (marées de printemps et d'automne). Les individus échantillonnés ont été mesurés sur place, de la base, à l'apex de la coquille (voir Figure 4.1) à l'aide d'un pied à coulisse (précision : centième de millimètre) et relâchés sur place.

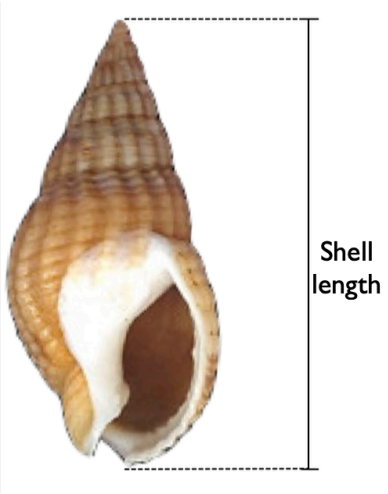


Figure 4.1: Coquille de nasse réticulée *Nassarius reticulatus* (Linnaeus, 1758)

L'espèce étudiée présente les caractéristiques suivantes :

- Les individus ont une durée de vie de 5 ans en moyenne.
- Les plus grands individus peuvent atteindre une taille de plus de 40 millimètres.
- Il n'y a qu'une seule période de reproduction chaque année. Il n'y a donc qu'un unique recrutement chaque année, au tout début du mois de mars.

Des travaux antérieurs, réalisés au laboratoire, ont montré que la taille et la masse des individus étaient liées par la relation allométrique suivante :

$$w = 0,0013 \cdot l^{2,3}$$

avec w, la masse en grammes et l la longueur des coquilles en millimètres.

> **!** Objectif principal
>
> L'objectif principal de cette étude est de produire la courbe d'Allen d'une cohorte de cette populations. Cette courbe sera utilisée pour déterminer des gains et des pertes de biomasses au sein de l'écosystème étudié. Les étapes nécessaires à la production de cette courbe sont présentées ci-dessous.

4.3 Avant de vous lancer...

Pour travailler dans de bonnes conditions, vous aurez absolument besoin de travailler dans un script et à l'intérieur d'un `Rproject`. Si vous ne savez plus comment faire, reportez-vous

aux chapitres correspondants du livre en ligne de Biométrie du semestre 3 : au sujet des scripts et au sujet des `Rprojects`. Vous devrez notamment (liste non exhaustive !) :

1. Créer un nouveau script (nommez-le `Nasses.R`)
2. Télécharger (si besoin) et charger quelques packages (voir plus bas)
3. Télécharger dans votre répertoire de travail le fichier Nassarius.csv
4. Importer dans `RStudio` les données du fichier `Nassarius.csv`

Pour cette section, vous aurez besoin des packages suivants :

- `tidyverse` : pour manipuler les données et faire des graphiques (Wickham et al. 2019)
- `mixdist` : pour effectuer les décompositions polymodales (Macdonald et Juan Du 2018)

Si vous ne savez plus comment installer et charger des packages en mémoire, reportez-vous au chapitre correspondant du livre en ligne de Biométrie du semestre 3

```
library(tidyverse)
library(mixdist)
```

Pour importer les données, utilisez l'assistant d'importation de `RStudio`. Notez que :

- Les colonnes sont séparées par des tabulations
- Le symbole utilisé pour les décimales dans le fichier `Nassarius.csv` est la virgule

Vous devrez donc spécifier correctement ces éléments pour pouvoir importer le fichier dans le logiciel. Si vous ne savez plus comment faire, reportez-vous au chapitre correspondant du livre en ligne de Biométrie du semestre 3.

Si l'importation s'est déroulée normalement, vous devriez maintenant disposer de l'objet nommé `Nassarius` suivant :

```
Nassarius
```

```
# A tibble: 3,710 x 2
    size date
   <dbl> <chr>
 1  4.8  march-10
 2  3    march-10
 3  5.56 march-10
 4  3.74 march-10
 5  4.1  march-10
 6  2.21 march-10
 7  2.75 march-10
 8  3.18 march-10
 9  2.56 march-10
10  4.36 march-10
# i 3,700 more rows
```

Et la commande suivante devrait produire exactement ces résultats :

```
Nassarius |>
  count(date)
```

```
# A tibble: 10 x 2
   date         n
   <chr>    <int>
 1 march-10   468
 2 march-11   481
 3 march-12   460
 4 march-13   468
 5 march-14   487
 6 sept-10    268
 7 sept-11    276
 8 sept-12    265
 9 sept-13    258
10 sept-14    279
```

4.4 Les étapes de l'analyse

Pour produire une courbe d'Allen, de nombreuses étapes sont nécessaires. Vous devrez :

`|>` : le pipe est un opérateur spécial qui prend l'objet situé à gauche et le transmets à la fonction placée à droite, en guise de premier argument. P. ex. : `Nassarius |> count(date)` est équivalent à `count(Nassarius, date)`
`count()` : compte le nombre d'occurrences de chaque valeur possible (ou niveau/modalité) d'une variable catégorielle (ou facteur).

1. Produire la *structure démographique instantanée* de la population à chaque date d'échantillonnage
2. Réaliser la *décomposition polymodale* pour identifier les cohortes présentes dans la population à chaque date d'échantillonnage.
3. Déterminer la taille moyenne de la coquille des individus de la cohorte d'intérêt, à chaque date d'échantillonnage, pour produire la *courbe de croissance*
4. Déterminer l'abondance des individus de la cohorte d'intérêt, à chaque date d'échantillonnage, pour produire la *courbe de survie*

Je vais présenter ci-dessous les étapes de cette procédure **pour une unique date d'échantillonnage** : march 2010. Vous devrez reproduire ces étapes pour les 9 autres dates d'échantillonnage.

4.5 Sélection des données d'une date spécifique

Notre jeu de données `Nassrisu` contient 2 colonnes : la première contient les dates d'échantillonnage et la seconde les tailles individuelles en millimètres. La première étape consiste à créer un nouvel objet (nous le nommerons `Nas_01`) qui contiendra uniquement les données collectées lors de la toute première session d'échantillonnage de mars 2010. Vous devriez déjà savoir comment utiliser la fonction `filter()` pour le faire :

```
Nas_01 <- Nassarius |>
  filter(date == "march-10")

Nas_01
```

```
# A tibble: 468 x 2
    size date
   <dbl> <chr>
 1  4.8  march-10
 2  3    march-10
 3  5.56 march-10
```

```
 4  3.74 march-10
 5  4.1  march-10
 6  2.21 march-10
 7  2.75 march-10
 8  3.18 march-10
 9  2.56 march-10
10  4.36 march-10
# i 458 more rows
```

```
dim(Nassarius)
```

```
[1] 3710    2
```

```
dim(Nas_01)
```

```
[1] 468   2
```

`filter()` : permet de filtrer les lignes d'un tableau (ici `Nassarius`) pour ne conserver que celles qui remplissent une condition spécifiée par l'utilisateur (ici `date == "march_10"`)

`dim()` : Affiche le nombre de lignes et de colonnes d'un tableau.

Comme nous pouvons le constater, on passe du tableau original `Nasarius` contenant 3710 lignes à un nouveau tableau `Nas_01` qui en contient seulement 468.

4.6 Structure démographique instantanée

Maintenant que nous disposons d'une table `Nas_01` qui contient uniquement la taille des individus collectés en mars 2010, il nous faut visualiser la structure démographique instantanée afin de déterminer combien de cohortes étaient présentes dans la population à cette date. La structure démographique instantanée est ici simplement un histogramme présentant la distribution des tailles individuelles. Nous allons donc placer la taille des individus sur l'axe des `x` en guise de descripteur individuel, et sur l'axe des `y`, `RStudio` placera automatiquement l'abondance pour chaque classe de taille en guise de descripteur populationnel.

```
Nas_01 |>
  ggplot(aes(x = size)) +
  geom_histogram()
```

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

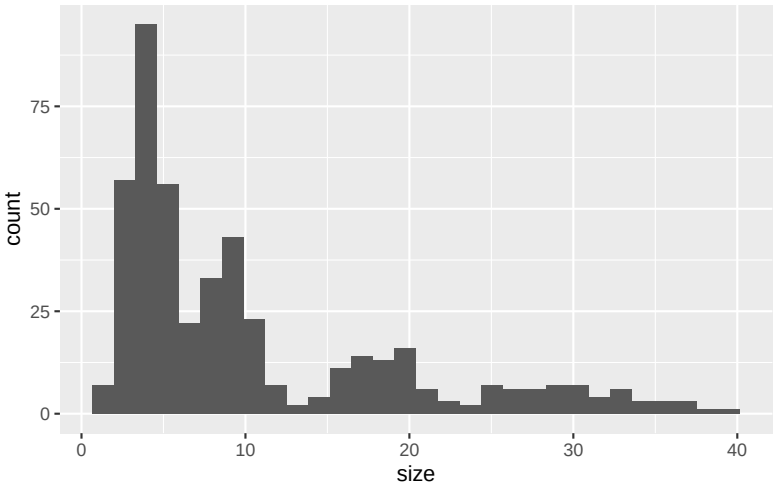


Figure 4.2: Structure démographique instantanée

Notez le message d'avertissement qui s'affiche quand vous produisez cet histogramme. Il indique que R a choisi pour nous le nombre de classes de tailles. Par défaut, il crée 30 classes, mais nous indique que ce n'est certainement pas le meilleur choix. Nous allons donc devoir créer nous même manuellement les classes de tailles pour (i) identifier les cohortes présentes dans la population et (ii) fixer des classes identiques que nous utiliserons pour toutes les autres dates d'échantillonnage et qui rendront les comparaisons plus aisées.

`ggplot()` : crée un graphique
`aes()` : associe une variable d'un jeu de données à une caractéristique esthétique d'un graphique (p. ex. la position le long de l'axe des `x` ou des `y`, la couleur, la forme, la taille, etc.)
`geom_histogram()` : ajoute un objet géométrique de type histogramme à une graphique produit par `ggplot()`

Puisque les individus de cette espèce peuvent atteindre une taille de 40 millimètres environ, nous allons définir des classes de tailles tous les millimètres. On peut faire ça simplement en créant un vecteur qui contient les limites des classes de tailles que l'on souhaite :

```
# Calcul de la taille maximale observée
taille_max <- max(Nassarius$size, na.rm = TRUE)
taille_max
```

```
[1] 41.33
```

`max()` : affiche la valeur maximale contenue dans un vecteur

```
# Définition des limites des classes de tailles
limites <- 0:(taille_max + 1)
```

```
limites
```

```
 [1]  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
[26] 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
```

L'objet `limites` contient les limites des classes de tailles que nous utiliserons pour produire les structures démographiques instantanées, pour chaque date d'échantillonnage, et pas seulement pour mars 2010. C'est la raison pour nous avons utilisé `max(Nassarius$size) +1` : cette syntaxe nous assure que nos classes de tailles recouvrent bien la taille de tous les individus échantillonnés au cours des 5 années d'études. Ainsi, nous pourrons utiliser les mêmes classes de tailles pour toutes les dates.

: : l'opérateur "deux points" permet de créer des suites d'entiers

Maintenant, on peut utiliser le vecteur `limites` comme argument de la fonction `geom_histogram()`. On modifie aussi les couleurs des barres de l'histogramme pour mieux visualiser les classes :

```
Nas_01 |>
  ggplot(aes(x = size)) +
  geom_histogram(breaks = limites, color = "black", fill = "steelblue")
```

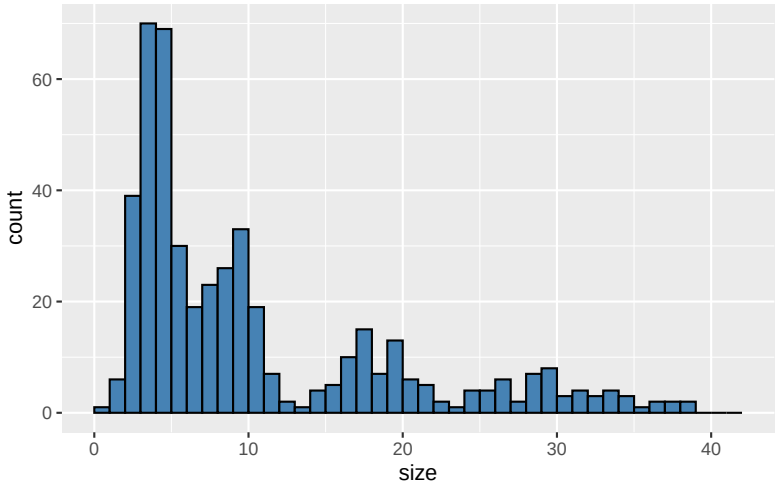


Figure 4.3: Structure démographique instantanée utilisable pour la décomposition polymodale

Cette structure démographique instantanée (Figure 4.3) fait apparaître 5 cohortes, bien que la plus âgée soit à peine visible. Les pics, ou modes, sont situés autour de 4, 9, 17, 30 et

34 millimètres. Ces valeurs approximatives sont importantes car nous les utiliserons pour réaliser la décomposition polymodale.

4.7 Décomposition polymodale

4.7.1 Principe et étapes

Effectuer une décomposition polymodale revient à tenter d'ajuster une distribution Normale à chaque cohorte de la population. Puisque nous avons identifié 5 cohortes dans la population en mars 2010, il nous faut identifier 5 distribution Normales distinctes dont la somme reproduira le plus fidèlement possible la structure démographique instantanée observée. Plusieurs étapes sont requises pour y parvenir.

Nous allons utiliser le package `mixdist`, que vous devez avoir déjà installé et chargé en mémoire. Ce package fournit plusieurs fonctions relativement simples d'utilisation et qui permettent d'ajuster des distributions Normales à une distribution observée (notre structure démographique instantanée).

La fonction du package `mixdist` qui nous permettra de réaliser la décomposition polymodale est la fonction `mix()`. Son fichier d'aide nous indique que pour fonctionner correctement, il faut lui fournir plusieurs choses :

```
?mix()
```

`?` : l'opérateur "point d'interrogation" suivi du nom d'une fonction, permet d'ouvrir le fichier d'aide de cette fonction.

Les deux premiers arguments de la fonction `mix()` ne possèdent pas de valeur par défaut. Nous devons donc spécifier nous même ces deux arguments :

1. Le premier s'appelle `mixdat`. Il doit obligatoirement s'agir d'un tableau de données (`data.frame` ou `tibble`) contenant 2 colonnes. La première colonne doit contenir les limites supérieures des classes de tailles de notre structure démographique instantanée. Le dernier élément de cette colonne doit être "ouvert" : il doit être

fixé à $+\infty$. La seconde colonne doit contenir les abondances pour chaque classe de taille de la structure démographique instantanée. Nous allons voir juste après comment créer cet objet `mixdat`.

2. Le second argument s'appelle `mixpar`. Là encore, il s'agit d'un tableau de données. Il doit contenir des valeurs approchées pour les paramètres des distributions Normales que nous souhaitons ajuster à chacune de nos 5 cohortes. Chaque distribution Normale possède 2 paramètres : la moyenne (qui correspond à la position du pic sur l'axe des `x` d'un histogramme) et l'écart-type (qui correspond à l'étalement de la courbe en cloche, c'est-à-dire à la dispersion des données de part et d'autre de la moyenne). Là encore, nous verrons plus bas comment créer cet objet.

4.7.2 Création du premier objet

Pour créer le premier tableau qui sera utilisé comme argument `mixdat` de la fonction `mix()`, nous allons nous servir de la fonction `mixgroup()` du package `mixdist`. Elle est très simple 'a utiliser puisqu'elle n'a besoin que de 2 arguments :

1. Un vecteur de données (ici, la taille de tous les individus échantillonnés en mars 2010)
2. La liste des valeurs correspondant aux limites des classes de tailles de la structure démographique instantanée. Nous disposons déjà de cet objet puisque nous l'avons créé plus tôt : c'est le vecteur `limites`

```
mix_01 <- mixgroup(Nas_01$size, breaks = limites)
class(mix_01)
```

```
[1] "mixdata"    "data.frame"
```

L'objet `mix_01` que nous venons de créer est donc un `data.frame`, mais c'est aussi un objet de classe `mixdata` que nous utiliserons en guise de premier argument de la fonction `mix()`. Pour afficher son contenu, il suffit comme toujours de taper son nom :

```
mix_01
```

	X	count
1	1	1
2	2	6
3	3	39
4	4	70
5	5	69
6	6	30
7	7	19
8	8	23
9	9	26
10	10	33
11	11	19
12	12	7
13	13	2
14	14	1
15	15	4
16	16	5
17	17	10
18	18	15
19	19	7
20	20	13
21	21	6
22	22	5
23	23	2
24	24	1
25	25	4
26	26	4
27	27	6
28	28	2
29	29	7
30	30	8
31	31	3
32	32	4
33	33	3
34	34	4
35	35	3
36	36	1
37	37	2
38	38	2
39	39	2
40	40	0
41	41	0
42	Inf	0

Notez que la dernière ligne de `mix_01` correspond à la catégorie [42 mm ; $+\infty$[. Pour vérifier que nous n'avons pas fait d'erreur, on peut faire une représentation graphique de cet objet particulier avec la fonction `plot()` :

```
plot(mix_01, xlab = "Size (mm)", ylab = "Probability")
```

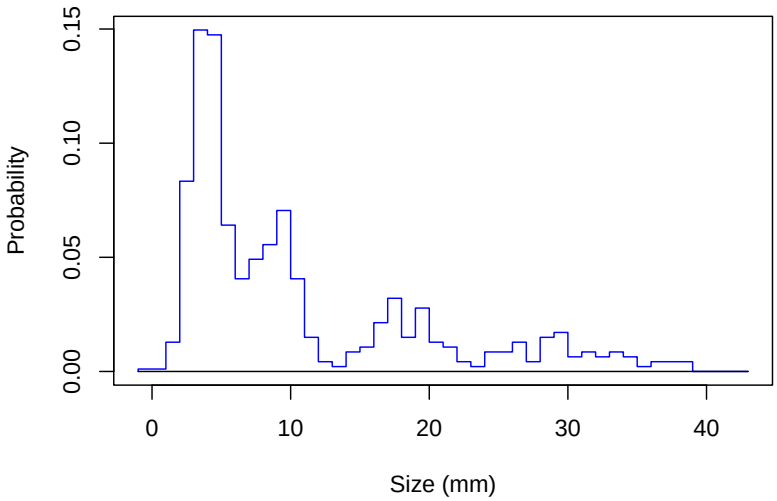


Figure 4.4: Le contenu de `mix_01` correspond exactement à la structure démographique instantanée de mars 2010. La ligne bleue est le contour de la structure démographique instantanée, qui devra être approchée par la superposition de 5 courbes Normales correspondant au 5 cohortes qui composent la population en mars 2010

Au final, l'objet `mix_01` est simplement la structure démographique instantanée de mars 2010, présentée dans un format qui sera compris par la fonction `mix()` que nous utiliserons pour effectuer la décomposition polymodale.

`plot()` : fonction générique permettant de produire des graphiques sans `ggplot2`. La forme du résultat dépendra de la classe de l'objet utilisé comme argument.

4.7.3 Création du deuxième objet

Pour créer le second `data.frame()` dont la fonction `mix()` aura besoin, nous allons utiliser une autre fonction du package `mixdist` : la fonction `mixparam()`. Cette fonction prend au minimum 2 arguments :

1. `mu` : un vecteur qui contient la position approximative des modes (ou pics) de chaque cohorte. Si nous avons 5 cohortes à identifier, il faudra fournir 5 valeurs pour `mu`. Pour mars 2010, nous utiliserons les valeurs présentées plus au, au niveau de la Figure 4.3 (4, 9, 17, 30 et 34 millimètres). Ces valeurs changeront pour chaque date d'échantillonnage et il faudra donc examiner attentivement les structures démographiques instantanées de chaque échantillonnage pour les déterminer. J'insiste

sur le fait que la position des pics peut être approximative, mais qu'elle ne doit malgré tout pas être trop éloignée des vraies valeurs.

2. `sigma` : un vecteur qui contient la valeur approchée de l'écart-type de chaque cohorte. L'écart-type d'une cohorte correspond à l'étalement des tailles de part et d'autres de la moyenne de la cohorte. C'est ce que nous avons appelé "polymorphisme" dans le cours de dynamique des populations, puisque cet étalement représente des performances de croissance variables pour des individus qui sont tous nés approximativement en même temps. Là encore, il faut fournir autant de valeurs que nous avons de cohortes (soit 5 pour mars 2010). Puisque nous n'avons pas d'idée précise d'ordre de grandeur pour ces écarts-types, nous utiliserons la valeur `1` pour chaque cohorte.

Ainsi, on obtient le second `data.frame` ainsi :

```
param_01 <- mixparam(mu = c(4, 9, 17, 30, 34),
                     sigma = c(1, 1, 1, 1, 1))
param_01
```

```
  pi mu sigma
1 0.2  4     1
2 0.2  9     1
3 0.2 17     1
4 0.2 30     1
5 0.2 34     1
```

La première colonne de ce nouveau `data.frame`, nommée `pi`, correspond à la proportion de l'effectif total échantillonné, contenu dans chaque cohorte. Puisque nous n'avons rien spécifié, `mixparam()` suppose que chaque cohorte contient une proportion identiques des individus de la population, et fixe donc la proportion à 0.2 (soit 20% de l'abondance totale dans chacune des 5 cohortes supposées). Nous savons pertinemment que ces proportions sont fausses puisqu'en réalité, l'abondance au sein des cohortes décroit avec l'âge des individus en raison de la mortalité. On sait donc que la cohorte la plus jeune sera la plus abondante, et que les cohortes plus âgées seront de moins en moins abondantes. Ces proportions seront estimées automatiquement par la fonction `mix()`

`mixparam()` : fonction qui crée un `data.frame` dans lequel chaque ligne contient les caractéristiques approchées d'une cohorte de la population échantillonnée
`c()` : fonction qui permet de créer un vecteur, donc une collection d'éléments qui sont tous du même type (ici, des valeurs numériques).

lorsque nous réaliserons la décomposition polymodale. Les autres colonnes de `param_01`, qui contiennent les valeurs que nous avons fournies manuellement, seront également ajustées lors de la décomposition polymodale

4.7.4 Ajustement des lois Normales aux données

Maintenant que nous disposons des deux objets nécessaires, nous pouvons réaliser la décomposition polymodale qui consiste, grâce à la fonction `mix()`, à ajuster une distribution Normale à chaque cohorte supposée de la population échantillonnés.

```
res_01 <- mix(mix_01, param_01)
```

```
Warning in mix(mix_01, param_01): The optimization process terminated because
iteration limit exceeded
```

```
res_01
```

```
Parameters:
       pi     mu sigma
1 0.46418  3.903 1.086
2 0.27062  8.795 1.555
3 0.14355 18.115 2.102
4 0.09100 28.231 2.861
5 0.03066 35.021 2.344

Distribution:
[1] "norm"

Constraints:
   conpi    conmu consigma
  "NONE"   "NONE"   "NONE"
```

La fonction `mix()` peut produire quelques avertissements. Ignorez-les à ce stade : seuls les messages d'erreurs sont problématiques et nous y reviendrons plus tard. L'objet `res_01` contient donc les résultats de la décomposition polymodale.

`mix()` : réalise la décomposition polymodale. La fonction identifie une combinaison de distributions Normales qui s'ajuste le mieux possible aux données observées compte tenu des données brutes et des caractéristiques approximatives de chaque distribution Normale fournie par l'utilisateur.

C'est une liste de 3 éléments dont seul le premier nous intéresse.^[Il s'appelle `parameters`, sans majuscule, même si `R` affiche son nom avec une majuscule. Vous pouvez le vérifier en tapant `str(res_01)`.

`str()` : affiche la **str**ucture interne d'objets complexes.] Il contient les valeurs ajustées pour les 3 paramètres des distributions Normales (`pi`, `mu` et `sigma`) pour chacune des 5 cohortes identifiées en mars 2010.

Ainsi, par exemple, pour l'échantillon de mars 2010, la première ligne du tableau `parameters` contenu dans `res_01` nous apprend les choses suivantes :

```
res_01$parameters
```

```
          pi        mu    sigma
1 0.46417551  3.903201 1.085989
2 0.27061611  8.795297 1.555004
3 0.14354832 18.114948 2.101807
4 0.09099874 28.230516 2.861214
5 0.03066132 35.020985 2.343520
```

- La première cohorte représente 46.42% de l'abondance totale de la population
- La taille moyenne des individus composant cette cohorte vaut 3.9 millimètres
- L'écart-type (ou polymorphisme de taille) de cette cohorte vaut 1.1 millimètres

Les distributions Normales qui ont été ajustées peuvent être visualisées grâce à la fonction `plot()` :

```
plot(res_01)
```

On observe sur la Figure 4.5 et dans l'objet `res_01$parameters` que la qualité de l'ajustement (et donc de la décomposition polymodale) est bonne :

- la courbe verte représente bien les données observées : elle est bien ajustée aux contours de l'histogramme.
- l'abondance des cohortes décroit avec l'âge de la cohorte : `pi` décroit de la cohorte la plus jeune à la cohorte la plus âgée.

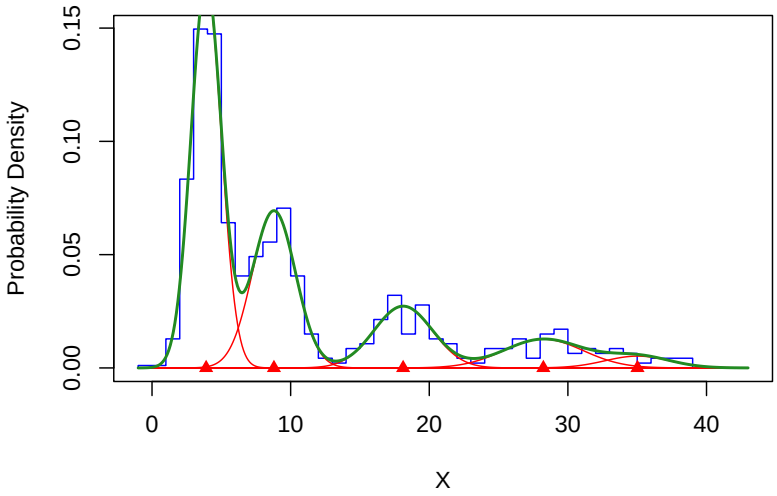


Figure 4.5: Distribution observée (histogramme bleu) et cohortes ajustées (courbes rouges). Les triangles indiquent la position du mode des cohortes et la courbe verte est la somme de toutes les distributions Normales. Idéallement, cette courbe devrait être étroitement ajustée aux données observées

- la taille moyenne des individus augmente avec l'âge de la cohorte : `mu` augmente de la cohorte la plus jeune à la cohorte la plus âgée.
- le polymorphisme (ou dispersion des tailles autour de la moyenne) augmente avec l'âge de la cohorte : `sigma` augmente de la cohorte la plus jeune à la cohorte la plus âgée.

! De l'importance du choix des valeurs initiales

Si nous avions choisi d'autres valeurs approchée lors de la création de l'objet `param_01` avec la fonction `mixparam()`, l'ajustement obtenu aurait pu être différent. La décomposition polymodale n'est pas une science exacte et il est souvent nécessaire de procéder par tâtonnements pour trouver des valeurs satisfaisantes. En particulier, les résultats que nous obtenons dépendent :

- du choix des valeurs pour la fonction `mixparam()`
- du choix des classes de tailles pour la structure démographique instantanée. Ici, nous avons des classes de tailles de un millimètre de large, mais nous aurions pu faire un autre choix (1,5 millimètre de large, ou 2) et nous aurions alors obtenu des résultats différents.

Il est donc important de retenir que les résultats obtenus ne sont pas "justes" ou "faux" en tant que tel et qu'il n'y a pas qu'une seule "bonne réponse". La qualité des ré-

sultats obtenus s'apprécie au regard de ce qu'on connait de l'espèce étudiée (traits d'histoire de vie, période de reproduction et de recrutement, nombre de cohorte supposées selon la date, etc.), et de ce qu'on connait du comportement "normal" des cohortes (baisse de l'abondance avec l'âge, augmentation de la taille moyenne avec l'âge, augmentation du polymorphisme avec l'âge).

4.7.5 Et en cas de message d'erreur ?

Lorsque vous utiliserez la fonction `mix()`, il se peut que des messages d'erreurs et/ou des messages d'avertissement apparaissent.

ℹ Les messages d'information et les avertissements commencent en général par le mot `Warning` et sont presque toujours sans conséquence. La commande a été comprise par `RStudio` et un résultat a été produit. Dans le cas de la fonction `mix()`, les avertissements indiquent que la solution trouvée (l'ajustement des loi Normales aux cohortes observées) n'est peut-être pas optimale, mais une solution a néanmoins été obtenue.

💣 Les messages d'erreurs commencent par `Erreur` ou `Error` et indiquent que la commande n'a pas abouti. Dans le cas de la fonction `mix()`, cela peut être lié à 2 choses :

1. soit les valeurs choisies pour la fonction `mixparam()` sont trop éloignées de la position des pics réels, et la fonction `mix()` ne parvient donc pas à trouver de solution satisfaisante
2. soit le nombre de valeurs choisies pour la fonction `mixparam()` n'est pas le bon. Par exemple, s'il y a 5 cohortes et qu'on ne fournit que 4 valeurs pour `mu` et `sigma`, un message d'erreur apparaîtra. De même, s'il y a 4 cohortes et qu'on fournit 5 valeurs pour `mu` et `sigma`, un message d'erreur apparaîtra. Enfin, si on ne fournit pas le même nombre de valeurs pour `mu` et pour `sigma`, un message d'erreur apparaîtra.

Dans les deux cas, vous devez

1. revenir à votre structure démographique instantanée et l'observer plus attentivement

2. déterminer des valeurs plus appropriées[1] pour la fonction `mixparam()`
3. ré-exécuter le code depuis la création de l'objet `param_01` et jusqu'à la décomposition polymodale avec la fonction `mix()`

[1] ajoutez ou retirez une cohorte si besoin, et choisissez des valeurs plus proches des pics observés

4.8 Taille moyenne et abondance des cohortes

À partir des résultats de la décomposition polymodale, nous devons maintenant calculer l'abondance (*i.e.* le nombre d'individus) de chaque cohorte. Nous devrons ensuite stocker dans un nouveau tableau :

1. la date d'échantillonnage (mars 2010)
2. la valeur d'abondance de la cohorte dont on souhaite réaliser le suivi (il s'agit de la cohorte la plus jeune en mars 2010)
3. la taille moyenne des individus de la cohorte dont on souhaite réaliser le suivi[2]

[2] on sait déjà que cette taille vaut 3.9 millimètres en mars 2010.

L'abondance d'une cohorte est obtenue en multipliant la valeur de `pi` de la cohorte d'intérêt par le nombre total d'individus échantillonnés en mars 2010. En effet, sur la Figure 4.5, la surface totale comprise entre l'axe des abscisses et la courbe verte vaut 1. Cette surface correspond au nombre total d'individus échantillonnés en mars 2010 :

```
nrow(Nas_01)
```

```
[1] 468
```

Puisque la première cohorte représente 46.42% de l'abondance totale[3], tout ce dont on a besoin pour connaitre l'abondance de la cohorte la plus jeune est :

```
res_01$parameters[1, 1] * nrow(Nas_01)
```

```
[1] 217.2341
```

En arrondissant à l'entier le plus proche, on obtient :

[3] c'est ce que nous dit la valeur de `pi` de la première cohorte dans le tableau `res_01$parameters` :

pi	mu	sigma
0.4642	3.9032	1.0860
0.2706	8.7953	1.5550
0.1435	18.1149	2.1018
0.0910	28.2305	2.8612
0.0307	35.0210	2.3435

`nrow()` : affiche le nombre de lignes d'une matrice ou d'un `data.frame`

```
round(res_01$parameters[1, 1] * nrow(Nas_01), 0)
```

```
[1] 217
```

Nous savons donc maintenant que la première cohorte, la plus jeune de la population échantillonnée en mars 2010, a donc une taille moyenne de 3.9 millimètre et une abondance de 217 individus. Nous allons maintenant utiliser ces valeurs pour créer un tableau et placer le premier point des courbes de croissance, de survie et d'Allen.

`round()` : arrondit des valeurs numériques. Le deuxième argument permet d'indiquer le nombre de décimales.

4.9 Tableau et mise en forme des données

Nous venons de décrire ci-dessus la méthode que vous devrez appliquer pour chaque date d'échantillonnage. En suivant les mêmes étapes, vous devriez être en mesure d'obtenir la taille et l'abondance moyennes de notre cohorte d'intérêt pour les 9 dates restantes. Lorsque vous effectuerez la décomposition polymodale pour ces 9 dates, n'oubliez jamais que vous voulez suivre *systématiquement la même cohorte dans le temps*. Cette cohorte va progressivement évoluer vers des tailles plus importantes puisque les individus grandissent chaque mois et chaque année. Notre cohorte d'intérêt va donc progressivement se décaler vers la droite des structures démographiques instantanées, et les informations de cette cohortes ne seront donc pas systématiquement situées sur la première ligne des résultats de la décomposition polymodale.

Pour chaque date, on s'attend donc à ce que la taille moyenne des individus soit plus grande que la taille obtenue pour la date d'échantillonnage précédente. De même, l'abondance devrait diminuer au fil du temps en raison de la mortalité naturelle qui affecte tous les individus de la population.

Pour produire les 3 courbes (croissance, survie et Allen) dont nous avons besoin, nous allons créer un `tibble` contenant 3 colonnes :

1. la date d'échantillonnage
2. la taille moyenne des individus de la cohorte en millimètres

3. l'abondance de la cohorte (nombre d'individus de la cohorte)

Nous pouvons utiliser la fonction `tribble()` pour le faire :

```
cohort <- tribble(
  ~date,         ~size,                    ~abundance,
  "2010-03-01", res_01$parameters[1,2], res_01$parameters[1,1] * nrow(Nas_01)
)
```

`tribble()` : permet de créer un `tibble` ligne par ligne. Le "r" de `tribble` est l'abréviation de "row".

Pour chaque nouvelle date d'échantillonnage, vous devrez compléter ce tableau en ajoutant une nouvelle ligne à l'intérieur de cette fonction `tribble()`.

> **Attention !**
>
> Une erreur fréquente est de saisir les données obtenues à chaque date d'échantillonnage dans un tableau différent à chaque fois. Ça n'est pas ce qu'il faut faire ! Il faut au contraire compléter le tableau `cohorte` en ajoutant une nouvelle ligne au tableau existant de la façon suivante :
>
> ```
> cohort <- tribble(
>   ~date,         ~size,                    ~abundance,
>   "2010-03-01", res_01$parameters[1,2], res_01$parameters[1,1] * nrow(Nas_01),
>   "2010-09-01", ...                     , ...
> )
> ```

Nous devons spécifier une dernière chose afin de produire les graphiques : à ce stade, la colonne `date` du nouveau tableau `cohort` est considéré comme une variable de type `character` (type `<chr>`) :

```
cohort
```

```
# A tibble: 1 x 3
  date        size abundance
  <chr>      <dbl>     <dbl>
1 2010-03-01  3.90      217.
```

Il nous faut donc la transformer pour que `R` la reconnaisse comme étant une variable temporelle afin que les données apparaissent dans l'ordre chronologique (et non alphabétique) sur l'axe des abscisses de nos graphiques. Voilà comment procéder :

```
# On charge le package lubridate pour travailler avec des dates
# Ce package fait partie du tidyverse
# Si vous l'avez installé, lubridate est disponible sur votre ordinateur
library(lubridate)
cohort <- cohort |>
  mutate(date = date(date))
```

On vérifie que la variable `date` possède maintenant le type `<date>` :

`mutate()` : crée de nouvelles variables dans un `tibble`, ou modifie des variables existantes. `date()` : transforme des variables de type `<chr>` (caractères) en variable de type `<date>` (dates).

```
cohort
```

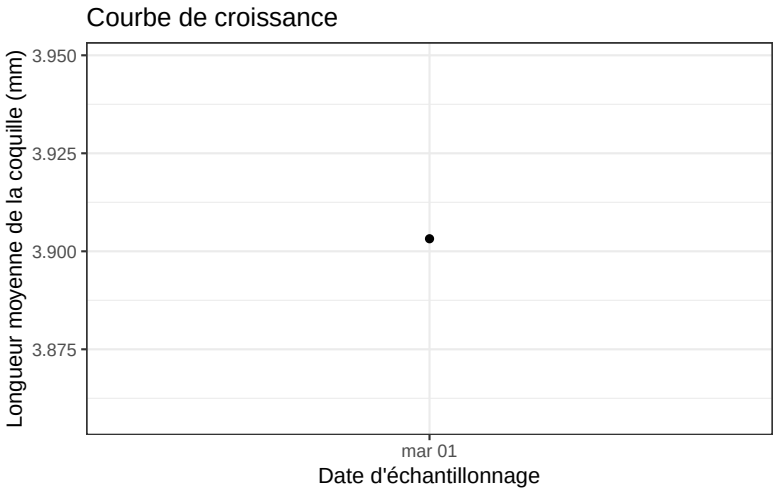
```
# A tibble: 1 x 3
  date        size abundance
  <date>     <dbl>     <dbl>
1 2010-03-01  3.90      217.
```

À ce stade, nous avons tout ce qu'il nous faut pour produire les courbes de croissance, de survie et d'Allen, à l'aide de `ggplot2`. Évidemment, chaque courbe ne contiendra ici qu'un seul point puisque nous avons examiné pour l'instant qu'une seule date d'échantillonnage. Elles ne seront complètes que lorsque que vous aurez répété ce travail pour l'ensemble des 10 dates d'échantillonnage.

4.10 Courbe de croissance

On place les dates d'échantillonnage sur l'axe des abscisses, et la taille moyennes des individus de la cohorte d'intérêt sur l'axe des ordonnées :

```
cohort |>
  ggplot(aes(x = date, y = size, group = 1)) +
  geom_point() +
  geom_line() +
  labs(x = "Date d'échantillonnage",
       y = "Longueur moyenne de la coquille (mm)",
       title = "Courbe de croissance") +
  theme_bw()
```



`geom_point()` et `geom_line()` : permettent d'ajouter des points et des lignes (respectivement) au graphique

`labs()` : permet de spécifier le titre du graphique et de ses axes

`theme_bw()` : change l'apparence générale du graphique

L'argument `group = 1` est utilisé pour indiquer que toutes les dates d'échantillonnage appartiennent à la même série temporelle, et qu'on souhaite donc relier les dates par une ligne. Le résultat sera nettement plus parlant quand vous aurez ajouté des données d'autres dates d'échantillonnage sur le graphique, afin de visualiser l'évolution de la taille moyenne des individus de la cohorte d'intérêt au fil du temps.

4.11 Courbe de survie

Pour produire la courbe de survie, on procède de la même façon, mais on place les dates d'échantillonnage sur l'axe des abscisses, et l'abondance de la cohorte d'intérêt sur l'axe des ordonnées :

```
cohort |>
  ggplot(aes(x = date, y = abundance, group = 1)) +
  geom_point() +
```

```
  geom_line() +
  labs(x = "Date d'échantillonnage",
       y = "Abondance de la cohorte",
       title = "Courbe de survie") +
  theme_bw()
```

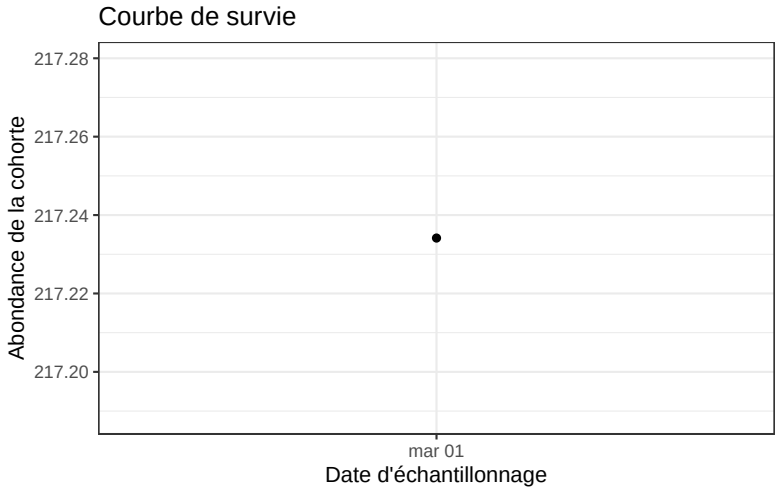


Figure 4.7: Courbe de survie d'une cohorte de la population de *Nassarius reticulatus*, suivie pendant 5 ans. À compléter avec les données des 9 autres dates d'échantillonnage

4.12 Courbe d'Allen

Pour produire la courbe de survie, on place la taille moyenne des individus de la cohorte sur l'axe des abscisses, et l'abondance de la cohorte d'intérêt sur l'axe des ordonnées :

```
cohort |>
  ggplot(aes(x = size, y = abundance, group = 1)) +
  geom_point() +
  geom_line() +
  labs(x = "Longueur moyenne de la coquille (mm)",
       y = "Abondance de la cohorte",
       title = "Courbe d'Allen") +
  theme_bw()
```

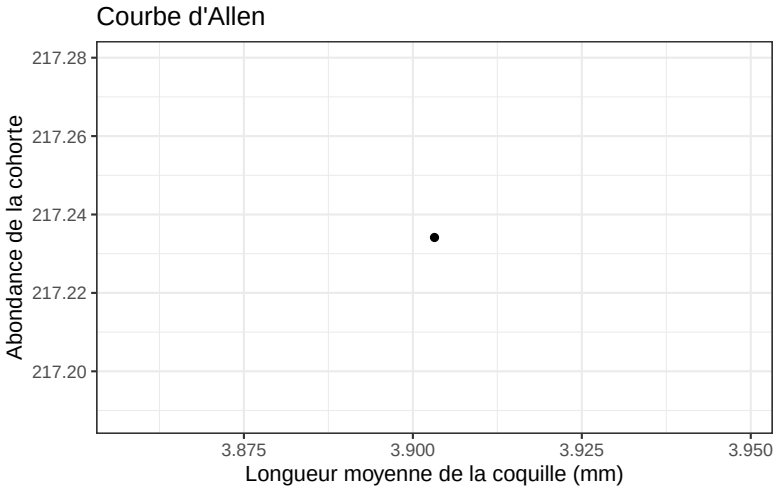


Figure 4.8: Courbe d'Allen d'une cohorte de la population de *Nassarius reticulatus*, suivie pendant 5 ans. À compléter avec les données des 9 autres dates d'échantillonnage

4.13 Relation allométrique

L'un des objectifs de ce travail était de produire une courbe d'Allen pour étudier les **variations de biomasses**. Pour y parvenir, nous devons faire une courbe d'Allen sur laquelle figure la masse moyenne des individus de la cohorte d'intérêt (en grammes), plutôt que la taille moyenne des individus de cette cohorte (en millimètres). Pour passer des tailles en millimètres aux masses en grammes, il nous suffit d'appliquer la relation allométrique fournie dans la Section 4.2, et d'ajouter une nouvelle colonne à notre tableau grâce à la fonction `mutate()` :

```
cohort_2 <- cohort |>
  mutate(weight = 0.0013 * size ^ 2.3)

cohort_2
```

```
# A tibble: 1 x 4
  date        size abundance weight
  <date>     <dbl>     <dbl>  <dbl>
1 2010-03-01  3.90      217. 0.0298
```

Il n'y a plus qu'à utiliser cette nouvelle variable sur l'axe des abscisses d'une courbe d'Allen :

```
cohort_2 |>
  ggplot(aes(x = weight, y = abundance, group = 1)) +
  geom_point() +
  geom_line() +
  labs(x = "Masse moyenne des individus (g)",
       y = "Abondance de la cohorte",
       title = "Courbe d'Allen") +
  theme_bw()
```

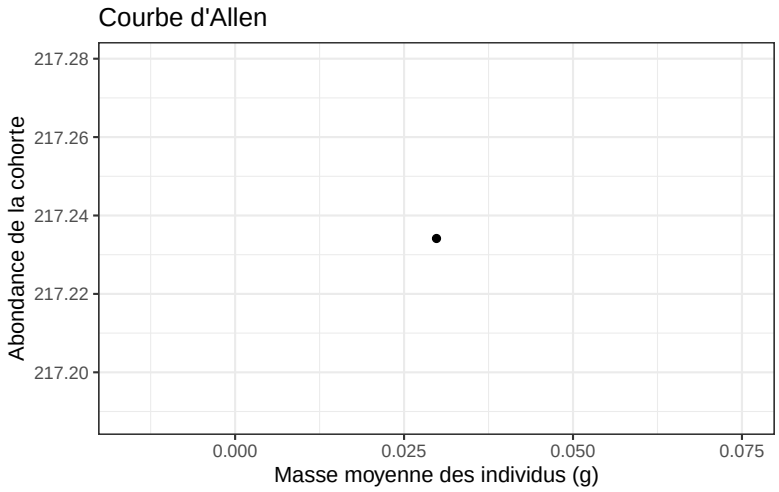
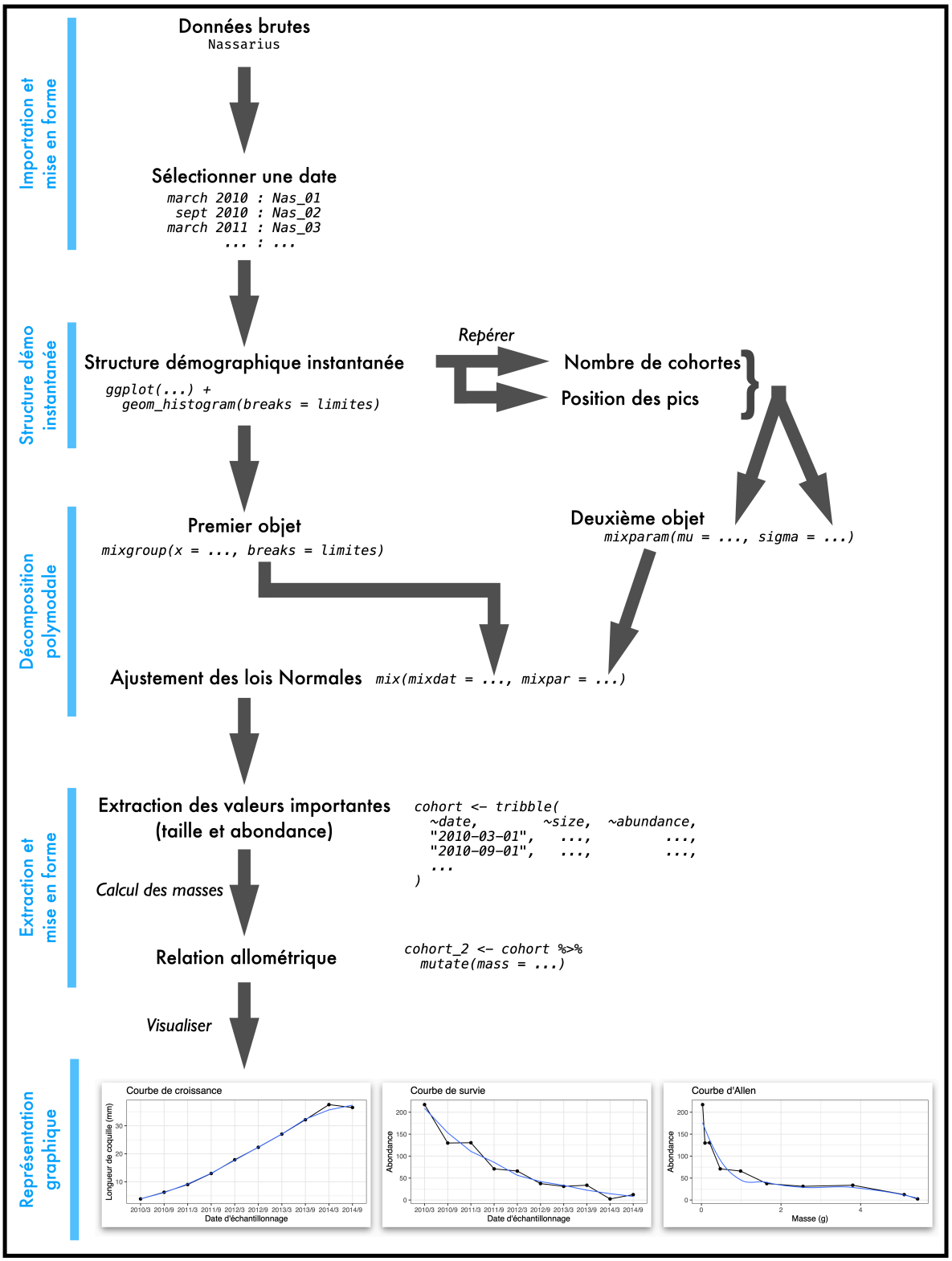


Figure 4.9: Courbe d'Allen d'une cohorte de la population de *Nassarius reticulatus*, suivie pendant 5 ans. À compléter avec les données des 9 autres dates d'échantillonnage

4.14 À vous de jouer !

Au final, voilà comment on peut résumer les différentes étapes de ce travail.

Assurez vous que vous avez bien compris chaque étape de la méthode décrite pour l'échantillonnage de mars 2010, et pourquoi on fait les choses présentées ici dans l'ordre où on les fait. À ce stade, vous devriez déjà avoir un script assez long qui devrait contenir la plupart des commandes évoquées plus haut. Vous devriez donc pouvoir reproduire ces étapes pour toutes les autres dates d'échantillonnage en faisant des copier-coller et en modifiant quelques éléments précis (dates, nom des objets, valeurs de tailles moyennes pour les cohortes, etc.). Attention aussi à adopter une structure de script la plus clair possible, afin de pouvoir corriger les éventuels bugs ou problèmes plus facilement.


Importation et mise en forme
Données brutes
Nassarius
Sélectionner une date
march 2010 : Nas_01
sept 2010 : Nas_02
march 2011 : Nas_03
... : ...
Structure démo instantanée
Structure démographique instantanée
ggplot(...) +
geom_histogram(breaks = limites)
Repérer
Nombre de cohortes
Position des pics
Décomposition polymodale
Premier objet
mixgroup(x = ..., breaks = limites)
Deuxième objet
mixparam(mu = ..., sigma = ...)
Ajustement des lois Normales
mix(mixdat = ..., mixpar = ...)
Extraction et mise en forme
Extraction des valeurs importantes
(taille et abondance)
cohort <- tribble(
~date, ~size, ~abundance,
"2010-03-01", ..., ...,
"2010-09-01", ..., ...,
...
)
Calcul des masses
Relation allométrique
cohort_2 <- cohort %>%
mutate(mass = ...)
Visualiser
Représentation graphique
Courbe de croissance
Longueur de coquille (mm)
Date d'échantillonnage
Courbe de survie
Abondance
Date d'échantillonnage
Courbe d'Allen
Abondance
Masse (g)

Il ne vous reste donc plus qu'à reproduire ce travail pour les 9 autres dates d'échantillonnage afin (i) de récupérer les valeurs d'abondance, de taille et de masse moyenne des individus de la cohorte d'intérêt (celle qui a été recrutée en mars 2010) et (ii) de compléter les 3 courbes que nous avons commencées plus haut.

Bon courage !

5 Systèmes dynamiques

5.1 Objectifs

Dans ce chapitre, vous allez étudier plusieurs modèles dynamiques vus en cours :

1. modèle de mortalité exponentielle
2. modèle de croissance exponentielle
3. modèle de croissance logistique
4. modèle de Lotka-Volterra
5. modèles de compétition

Pour cela, dans `RStudio`, vous devrez construire ces modèles en utilisant des boucles `for` (dont nous détaillerons la syntaxe à la Section 5.3) afin de décrire les variations d'abondance des populations étudiées au cours du temps. Quand ces modèles seront construits, vous devrez en faire varier les conditions initiales et les valeurs des paramètres, afin d'observer le comportement du modèles dans des conditions variées.

Pour chaque modèle cité plus haut, vous allez donc tout d'abord simuler des données dans des conditions précises, en faire la représentation graphique, puis modifier les paramètres du modèle pour comprendre comment les différents termes (de croissance, de mortalité, d'interaction…) affectent les populations.

5.2 Pré-requis

Comme pour chaque nouveau chapitre, je vous conseille de travailler dans un nouveau script que vous placerez dans votre répertoire de travail, et dans une nouvelle session de travail (Menu `Session > Restart R`). Inutile en revanche de créer un nouveau `Rproject` : vous pouvez tout à fait avoir plusieurs script dans le même répertoire de travail et pour un même

`Rproject`. Comme toujours, consultez le livre en ligne du semestre 3 si vous ne savez plus comment faire.

Si vous êtes dans une nouvelle session de travail (ou que vous avez quitté puis relancé `RStudio`), vous devrez penser à recharger en mémoire les packages utiles. Dans ce chapitre, vous aurez uniquement besoin des packages du `tidyverse` (Wickham 2023), en particulier les package `dplyr` (Wickham et al. 2023), pour manipuler des tableaux, et `ggplot2` (Wickham et al. 2024) pour les représentations graphiques.

```
library(tidyverse)
```

Nous n'aurons pas besoin ici d'importer de données depuis des fichiers externes : nous allons en effet générer directement des données à partir de différents modèles démographiques vus en cours.

Comme toujours, je spécifie une fois pour toutes le thème que j'utiliserai pour tous les graphiques de ce chapitre. Libre à vous de choisir un thème différent ou de vous contenter du thème proposé par défaut :

```
theme_set(theme_bw())
```

5.3 Les boucles `for`

Dans tous les langages de programmation, les boucles permettent de répéter automatiquement les mêmes actions un grand nombre de fois. Il existe plusieurs types de boucles, mais les boucles `for` sont les plus communes. Dans `R`, chaque boucle `for` possède 3 éléments :

1. **La sortie**. Avant d'exécuter la boucle, vous devez toujours créer de façon explicite un objet qui contiendra les résultats des calculs effectués à l'intérieur de la boucle. Une bonne façon de créer un vecteur vide pour stocker des valeurs numériques consiste à utiliser la fonction `numeric()`
2. **La séquence** pour laquelle vous souhaitez "faire tourner la boucle". Dans une boucle `for`, vous devez indi-

quer un indice qui spécifie le nombre d'itérations que la boucle doit effectuer. Par exemple :

```
for (i in 1:10)
```

indique que les commandes qui figurent à l'intérieur de la boucle devront être exécutées 10 fois, une fois pour chaque valeur de `i`, c'est-à-dire 1, 2, 3… jusqu'à 10. Pour chaque exécution des commandes de la boucle, l'indice `i` prendra donc une valeur différente comprise entre 1 et 10.

3. **Le corps de la boucle**. Il est constitué des commandes que l'on souhaite faire exécuter à la boucle, et il doit obligatoirement être encadré par des accolades `{ }`.

Voilà un exemple de boucle très simple :

```
# Création d'un vecteur numérique vide
res <- numeric()

# Pour chaque valeur de i comprise entre 1 et 5
for (i in 1:5) {
  res[i] <- 3 * i - 1 # Calcule 3 * i - 1
}

# Affichage des résultats
res
```

```
[1]  2  5  8 11 14
```

Les boucles peuvent faire des choses beaucoup plus complexes. Par exemple, supposez que vous avez une première valeur de 100, et que vous souhaitez diviser cette valeur par 2, puis que vous souhaitez diviser le résultat par 2, et ainsi de suite, 10 fois. Voilà comment faire ça avec une boucle `for` :

```
# Création d'un vecteur numérique vide
res <- numeric()

# Stockage de la valeur initialle dans cet objet
res[1] <- 100
```

```r
# Pour chaque valeur de i comprise entre 2 et 11
for (i in 2:11) {
  res[i] <- res[i - 1] / 2
}

# Affichage des résultats
res
```

```
 [1] 100.00000000  50.00000000  25.00000000  12.50000000   6.25000000
 [6]   3.12500000   1.56250000   0.78125000   0.39062500   0.19531250
[11]   0.09765625
```

Notez bien la position des accolades `{` et `}` dans le code ci-dessus. Leur position est importante, de même que les sauts de lignes. Notez aussi que dans cet exemple, `i` prend successivement les valeurs 2, 3, 4, … , 11, au lieu des valeurs 1, 2, 3, … , 10. En effet, on connait déjà la première valeur de `res`. On veut justement utiliser cette première valeur pour calculer la deuxième, la troisième et ainsi de suite. Si on souhaite répéter l'opération 10 fois de suite, il nous faut faire varier `i` de 2 à 11 inclus. Ici, nous avons en réalité calculé l'équation de récurrence suivante :

$$res_t = \frac{res_{t-1}}{2}$$

Vous savez maintenant comment fonctionnent les boucles `for` et vous devriez donc pouvoir construire votre premier modèle dynamique.

> **Attention**
>
> Dans `R`, le premier élément d'un objet occupe la position `[1]`. L'indice `[0]` n'existe pas. Il faut donc faire attention à ce que les indices choisis pour `i` dans l'expression `for (i in ...)` soient cohérents avec d'une part la position dans l'objet qui stockera les résultats, et d'autre part l'équation du système que l'on simule.

5.4 Modèle de mortalité exponentielle

Comme nous l'avons vu dans le cours, le modèle de mortalité exponentielle est défini par l'équation suivante :

$$\frac{\mathrm{d}N}{\mathrm{d}t} = -mN$$

Avec N, l'abondance de la population étudiée, t, le temps et m le taux de mortalité instantané. Pour coder ce modèle dans `RStudio`, vous devez vous rappeler de **la signification** de cette équation différentielle. Elle indique que dans un intervalle de temps très court $\mathrm{d}t$, **la variation de l'abondance de la population** $\mathrm{d}N$ vaut $-mN$. Autrement dit, entre 2 temps d'observation successifs (par exemple entre t et $t+1$, ou entre $t-1$ et t), l'abondance de la population passe de N à $N - mN$. Ce modèle peut donc être décrit par l'équation de récurrence suivante :

$$N_{t+1} = N_t - m \times N_t$$

ce qui est évidemment équivalent à

$$N_t = N_{t-1} - m \times N_{t-1}$$

Cette dernière équation peut se lire ainsi : l'abondance au temps t (N_t) est égale à l'abondance au temps précédent N_{t-1}, plus la variation d'abondance entre 2 pas de temps successifs ($-m \times N_{t-1}$). Vous devez utiliser cette équation pour construire le modèle de mortalité exponentielle à l'intérieur d'une boucle `for` comme dans le dernier exemple de la Section 5.3. Ici, on cherche donc à déterminer la nouvelle valeur de la variable (N au temps t) en utilisant la valeur de la même variable mais au temps précédent (N au temps $t-1$).

Plusieurs étapes sont donc nécessaires :

1. Créez un vecteur `N` qui sera utilisé pour stocker les résultats des calculs à chaque étape de simulation. Utilisez la commande suivante :

```
# Création d'un vecteur numérique qui contiendra les résultats
N <- numeric()
```

2. Fixez l'abondance initiale de la population à $N_0 = 10$

3. Fixez le taux de mortalité instantanée à $m = 0.05$ (à chaque pas de temps, 5% de la population disparaît)
4. Placez l'abondance initiale à la première position de `N`
5. Créez une boucle `for` pour calculer et placer dans `N` l'abondance de la population pour 100 pas de temps
6. Créez un tibble nommé `res` contenant 2 variables (`Generation` et `Abondance`) en utilisant cette commande :

```
# Création d'un tibble où les résultats seront stockés
res <- tibble(Generation = 0:100, N)
```

Si votre script est correct et qu'il s'exécute sans erreur, l'objet `res` devrait contenir 101 lignes, avec 101 valeurs d'abondance. Les premières lignes de `res` devraient contenir ces valeurs :

```
# A tibble: 101 x 2
   Generation     N
        <int> <dbl>
 1          0 10
 2          1  9.5
 3          2  9.02
 4          3  8.57
 5          4  8.15
 6          5  7.74
 7          6  7.35
 8          7  6.98
 9          8  6.63
10          9  6.30
# i 91 more rows
```

Visualisez ces données avec `ggplot2`. Votre graphique devrait ressembler à ceci :

> Pour comprendre ce modèle…
>
> Essayez d'utiliser plusieurs valeurs différentes pour N_0 et relancez tout le script. Que se passe-t-il quand N_0 est très grand ? Est-ce normal ?
> Enfin, essayez de changer la valeur de m et de relancer le script. Faites des essais avec plusieurs valeurs.

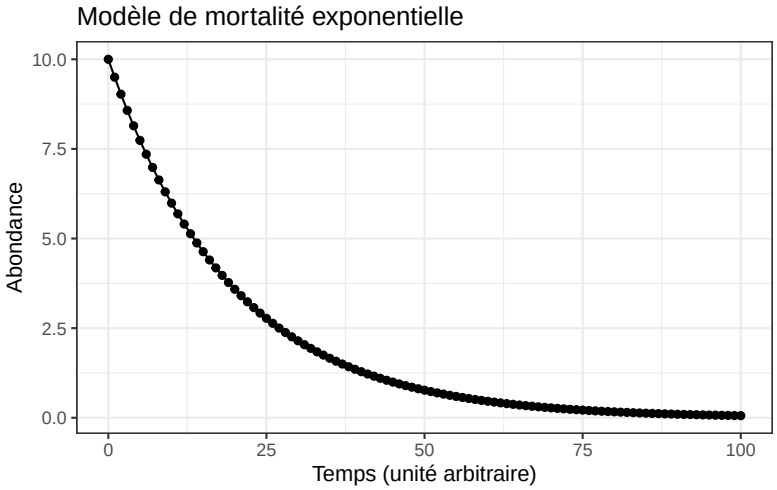


Figure 5.1: Modèle de mortalité exponentielle avec $N_0 = 10$ et $m = 0.05$.

Qu'observe-t-on quand la valeur de m se rapproche de 1 ? Est-ce normal ? Et que se passe-t-il si $m > 1$? Est-ce réaliste ? Pourquoi ? Mêmes questions avec $m = 2$ et $m = 3$?

5.5 Modèle de croissance exponentielle

Comme nous l'avons vu dans le cours, le modèle de croissance exponentielle est défini par l'équation suivante :

$$\frac{\mathrm{d}N}{\mathrm{d}t} = rN$$

Avec N, l'abondance de la population étudiée, t, le temps et r le taux de croissance instantané. Pour coder ce modèle dans `RStudio`, vous devez vous rappeler de **la signification** de cette équation différentielle. Elle indique que dans un intervalle de temps très court $\mathrm{d}t$, **la variation de l'abondance de la population** $\mathrm{d}N$ vaut $+rN$. Autrement dit, entre 2 temps d'observation successifs (par exemple entre t et $t+1$, ou entre $t-1$ et t), l'abondance de la population passe de N à $N + rN$. Ce modèle peut donc être décrit par l'équation de récurrence suivante :

$$N_{t+1} = N_t + r \times N_t$$

ce qui est évidemment équivalent à

$$N_t = N_{t-1} + r \times N_{t-1}$$

Cette dernière équation peut se lire ainsi : l'abondance au temps t (N_t) est égale à l'abondance au temps précédent N_{t-1}, plus la variation d'abondance entre 2 pas de temps successifs ($+r \times N_{t-1}$). Vous devez utiliser cette équation pour construire le modèle de mortalité exponentielle à l'intérieur d'une boucle `for` comme dans le dernier exemple de la Section 5.3. Ici, on cherche donc à déterminer la nouvelle valeur de la variable (N au temps t) en utilisant la valeur de la même variable mais au temps précédent (N au temps $t-1$).

Vous fixerez l'abondance initiale à $N_0 = 5$ et le taux de croissance instantané à $r = 0.03$. Vous calculerez l'abondance de la population pour 100 pas de temps. Vous devriez ainsi obtenir les résultats suivants :

```
# A tibble: 101 x 2
   Generation     N
        <int> <dbl>
 1          0  5
 2          1  5.15
 3          2  5.30
 4          3  5.46
 5          4  5.63
 6          5  5.80
 7          6  5.97
 8          7  6.15
 9          8  6.33
10          9  6.52
# i 91 more rows
```

Pour comprendre ce modèle…

Comme pour le modèle de mortalité exponentielle, essayez plusieurs valeurs pour la condition initiale N_0 et pour le paramètre du modèle r. Que se passe-t-il quand $r = 1$? Examinez les première valeurs d'abondance dans le tableau `res` pour vous faire une meilleure idée. Est-ce réaliste ? Que se passe-t-il quand $r > 1$? Est-ce possible dans la nature ?

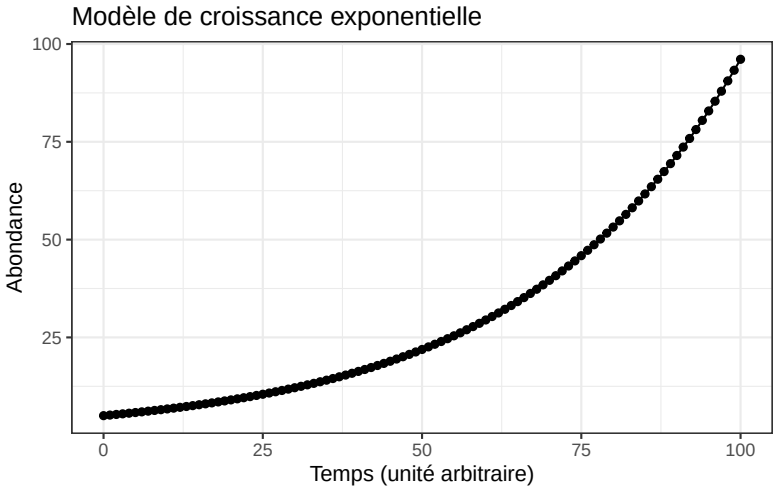


Figure 5.2: Modèle de croissance exponentielle avec $N_0 = 5$ et $r = 0.03$.

5.6 Modèle de croissance logistique

À partir du modèle de croissance exponentielle construit à la Section 5.5, vous devez maintenant construire un modèle de croissance logistique qui est défini par l'équation suivante :

$$\frac{\mathrm{d}N}{\mathrm{d}t} = rN - \frac{r}{K} \times N^2$$

Avec N, l'abondance de la population étudiée, t, le temps, r le taux de croissance instantané, et K la capacité biotique (ou capacité de charge) du milieu. Comme toujours, commencez par écrire cette équation sous forme d'une relation de récurrence :

$$N_t = N_{t-1} + \cdots$$

Rappelez-vous que la nouvelle abondance (au temps t) est égale à l'ancienne (au temps $t-1$), plus la variation d'abondance entre 2 pas de temps successifs, la formule de la variation étant fournie par la formule $\frac{\mathrm{d}N}{\mathrm{d}t} = \cdots$

Vous devez donc modifier le script de la Section 5.5 pour ajouter le terme d'auto-limitation lié à l'environnement $\frac{r}{K} \times N^2$. Vous utiliserez les valeurs suivantes pour les conditions initiales et les paramètres du modèle :

- $N_0 = 1$
- $r = 0.05$
- $K = 80$

- durée de la simulation : 200 pas de temps

Vous devriez ainsi obtenir les résultats suivants :

```
# A tibble: 201 x 2
   Generation     N
        <int> <dbl>
 1          0  1
 2          1  1.05
 3          2  1.10
 4          3  1.16
 5          4  1.21
 6          5  1.27
 7          6  1.33
 8          7  1.40
 9          8  1.47
10          9  1.54
# i 191 more rows
```

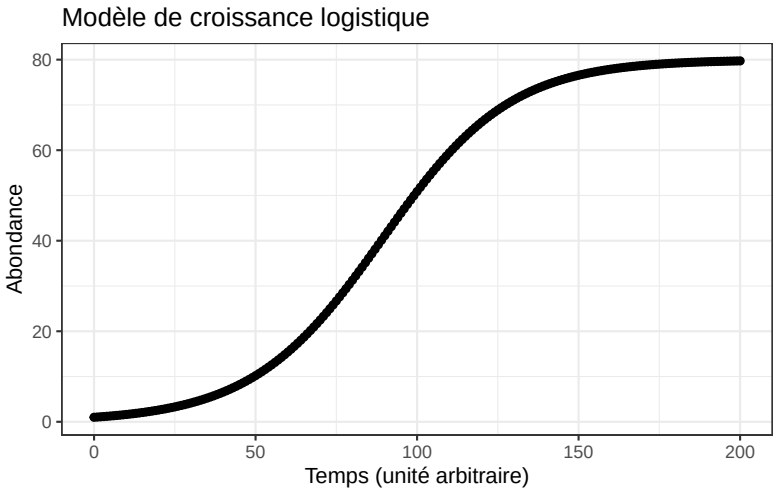


Figure 5.3: Modèle de croissance logistique avec $N_0 = 1$, $r = 0.05$ et $K = 80$.

Pour comprendre ce modèle…

Modifiez les valeurs de N_0, r et K l'une après l'autre, pour comprendre comment chaque paramètre affecte la cinétique observée. Que se passe-t-il quand N_0 augmente ? Que se passe-t-il quand $N_0 > K$? Est-ce logique ? Pouvez-vous imaginer des situations réalistes où $N_0 > K$? Quelle est l'unité de K ?
Enfin, que se passe-t-il si le terme d'auto-limitation du

milieu est ajouté à la croissance exponentielle au lieu d'être soustrait ? Expliquez.

5.7 Modèle prédateur-proie de Lotka-Volterra

Vous allez maintenant étudier le modèle prédateur-proie tel que formulé par Lotka et Volterra et dont les éuqations vous ont été décrites dans le cours magistral de "fonctionnement des écosystèmes" :

$$\begin{cases} \frac{dX}{dt} &= rX - \frac{r}{K}X^2 - c_1XY \\ \\ \frac{dY}{dt} &= -mY + c_2XY \end{cases}$$

avec X, l'abondance des proies, Y, l'abondance des prédateurs, r le taux de croissance instantanée des proies, K la capacité biotique du milieu, m le taux de mortalité instantanée des prédateurs, c_1 l'effet négatif de la prédation sur les proies (*i.e.* la mortalité des proies causée par les rencontres avec les prédateurs) et c_2 l'effet positif de la prédation sur les prédateurs (*i.e.* la croissance de la population de prédateurs causée par les rencontres avec les proies).

En utilisant la même démarche que précédemment, construisez un modèle de Lotka-Volterra avec les valeurs suivantes :

- Abondance initiale des proies : $X_0 = 1$
- Abondance initiale des prédateurs : $Y_0 = 10$
- Taux de croissance instantanée des proies : $r = 0.05$
- Taux de mortalité instantanée des prédateurs : $m = 0.05$
- Capacité biotique du milieu : $K = 50$
- Effet de la prédation sur les proies : $c_1 = 0.01$
- Effet de la prédation sur les prédateurs : $c_2 = 0.01$
- Durée de la simulation : 1000 pas de temps

Avec ce modèle, il vous faut calculer en parallèle l'abondance de 2 populations : celle des proies et celle des prédateurs.

! Calcul simultané

Attention : il est impossible de calculer d'abord toutes les abondances des proies, puis ensuite toutes les abondances des prédateurs, dans 2 boucles `for` distinctes. À chaque nouveau pas de temps, au sein d'une unique boucle `for`, il faut :

1. calculer la nouvelle abondance des proies à partir de l'abondance des proies et des prédateurs au temps précédent
2. calculer la nouvelle abondance des prédateurs à partir de l'abondance des proies et des prédateurs au temps précédent

Au final, l'objet `res` qui contiendra les résultats de vos simulation devra donc posséder 3 colonnes :

1. Une colonne `Generation` contenant le numéro des étapes de simulation (les pas de temps)
2. Une colonne `X` contenant les abondances de la population de proies à chaque étape de simulation
3. Une colonne `Y` contenant les abondances de la population de prédateurs à chaque étape de simulation

Avec ce tableau `res`, vous devriez pouvoir produire le diagramme des phases suivants :

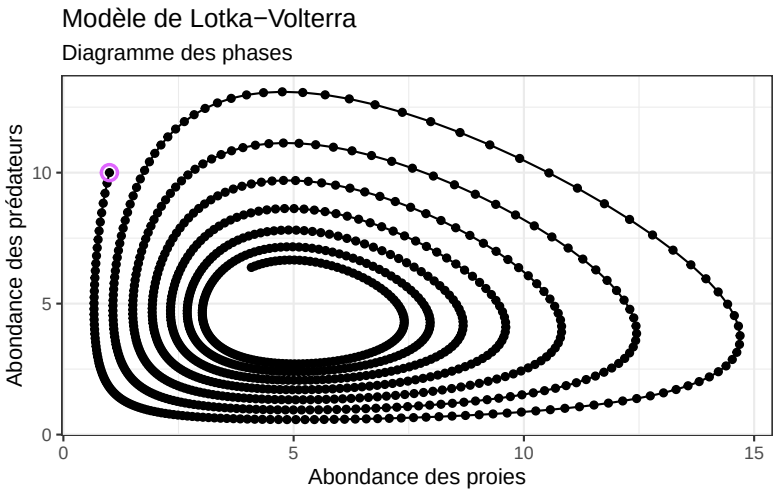
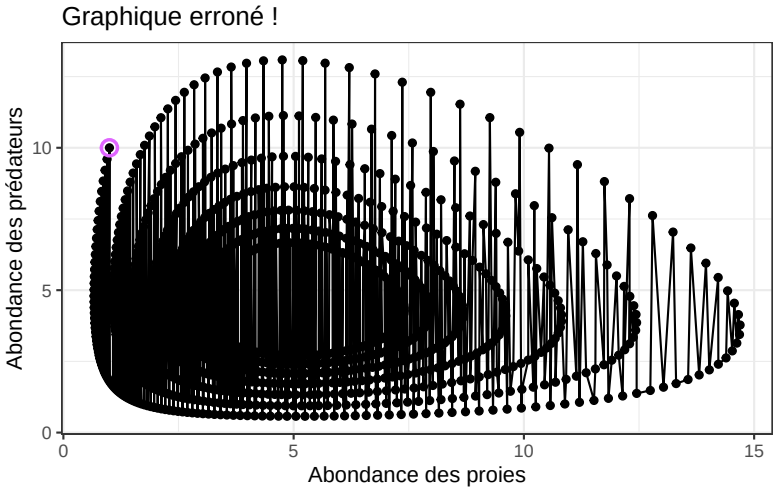


Figure 5.4: Modèle de Lotka-Volterra avec $X_0 = 1$, $Y_0 = 10$, $r = 0.05$, $m = 0.05$, $K = 80$ et $c_1 = c_2 = 0.01$. Le cercle violet indique les conditions initiales.

Pour faire ce graphique, vous aurez besoin d'utiliser la fonction `geom_path()` au lieu de `geom_line()`. En effet,

`geom_line()` relie les points d'un graphique par ordre d'abscisse croissant :



Ça n'est pas ce que nous voulons obtenir ! On souhaite relier les points par ordre chronologique, donc dans l'ordre où ils apparaissent dans le tableau `res`.

Enfin, pour mieux visualiser la dynamique de ces deux populations au cours du temps, produisez le graphique suivant :

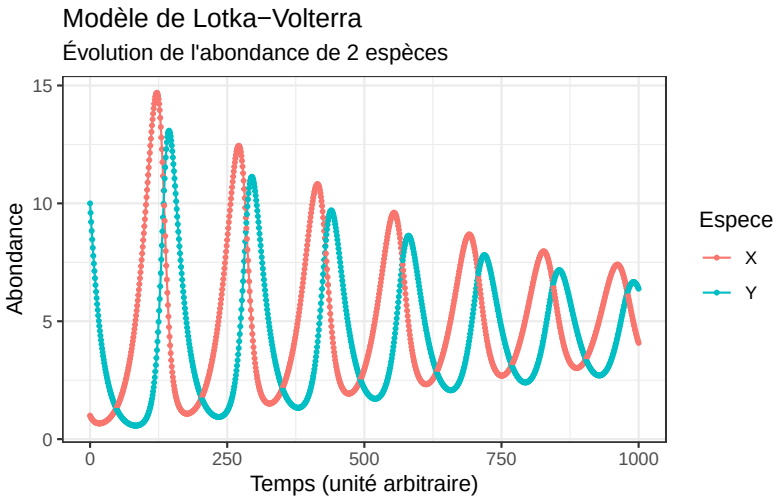


Figure 5.5: Modèle de Lotka-Volterra avec $X_0 = 1$, $Y_0 = 10$, $r = 0.05$, $m = 0.05$, $K = 80$ et $c_1 = c_2 = 0.01$.

Pour produire ce graphique, vous aurez besoin de transformer le tableau `res` en format long. Une colonne `Espece` devra contenir soit `X`, soit `Y`, et une colonne `Abondance` devra contenir toutes les valeurs d'abondance des 2 espèces. Si vous ne savez plus comment faire, relisez le chapitre 1.3 du livre

en ligne du semestre 4. Voilà à quoi devraient ressembler les premières lignes de votre tableau :

```
res_long
```

```
# A tibble: 2,002 x 3
   Generation Espece Abondance
        <int> <chr>      <dbl>
 1          0 X          1
 2          0 Y         10
 3          1 X          0.949
 4          1 Y          9.6
 5          2 X          0.904
 6          2 Y          9.21
 7          3 X          0.866
 8          3 Y          8.83
 9          4 X          0.832
10          4 Y          8.47
# i 1,992 more rows
```

Pour comprendre ce modèle…

Expérimentez et tentez de répondre aux questions suivantes :

- Quelle sera l'issue de ce modèle prédateur-proie ? Augmentez la durée de simulation pour le prouver.
- Quelles seront les abondances des proies et des prédateurs quand l'équilibre sera atteint ?
- Modifiez la capacité biotique de la proie. QUe se passe-t-il quand vous l'augmentez ? Que se passe-t-il quand vous la diminuez ? Est-ce normal ? Est-ce ce à quoi vous vous attendiez ?
- Que se passe-t-il quand vous modifiez les valeurs des autres paramètres du modèle (r, m, c_1 et c_2) ?
- Trouvez une combinaison de valeurs qui produisent un système à l'équilibre dynamique (*i.e.* un système d'oscillations harmoniques avec un diagramme des phases présentant un attracteur cyclique ou une courbe fermée)
- Est-ce que certaines combinaisons de paramètres produisent des résultats irréalistes ?

5.8 Modèle de compétition interspécifique

Dans sa forme la plus simple, on considère un système de compétition entre 2 espèces présentant un modèle de croissance logistique. La compétition ajoutera un terme de mortalité pour chaque espèce :

$$\begin{cases} \frac{dX}{dt} &= r_1 X - \frac{r_1}{K_1} X^2 - c_1 XY \\ \frac{dY}{dt} &= r_2 Y - \frac{r_2}{K_2} Y^2 - c_2 XY \end{cases}$$

avec X et Y l'abondance des deux espèces, r_1 et r_2 les taux de croissance instantanée des deux espèces, K_1 et K_2 les capacités biotiques pour les 2 espèces, et c_1 et c_2 les effets négatifs de la compétition sur chacune des 2 espèces (*i.e.* la mortalité causée par les rencontres entre les deux espèces en compétition).

En modifiant le système prédateur-proie précédent, codez ce système de compétition et essayez de trouver une combinaison de paramètres et de valeurs initiales permettant de reproduire les 4 situations décrites en cours :

1. Deux espèces très compétitives (avec différentes abondances initiales). Retrouvez-vous la ligne de catastrophe ?

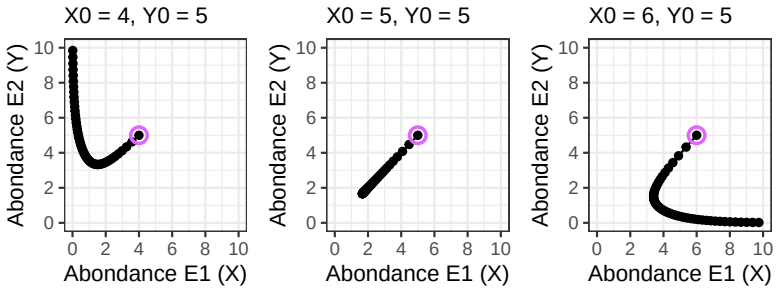


Figure 5.6: Diagramme des phases de 2 espèces en compétition : 2 espèces très compétitives avec des conditions initiales différentes.

2. Deux espèces très peu compétitives
3. Première espèce plus compétitive que la seconde
4. Seconde espèce plus compétitive que la première

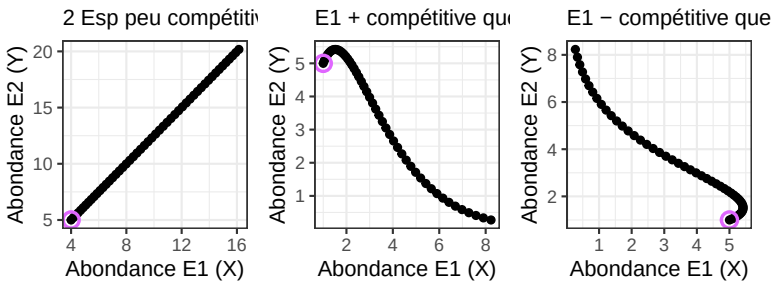


Figure 5.7: Diagramme des phases de 2 espèces en compétition : 3 situations distinctes.

Pour comprendre ce modèle…

Attention à l'échelle des axes de vos graphiques. Pour bien comprendre ce qui se passe et retrouver les figures schématiques vues en cours, vous devez tenter de replacer les graphique produits ici dans un plan plus large. Notez aussi que contrairement au modèle prédateur-proie, les modèles de compétition ne font pas apparaître de cycles sur les diagrammes des phases. L'évolution des abondances au cours du temps ne fait donc jamais apparaître d'oscillations.

5.9 Modèles à 3 espèces

Sur la base des chapitres précédents, vous devriez maintenant pouvoir construire des modèles plus complexes, par exemple à 3 espèces en compétition, ou à 3 espèces dont 2 proies en compétition et une espèce de prédateur. La clé est ici de ne pas oublier de termes lorsque l'on construit le modèle : pour chaque espèce supplémentaire dans le modèle, la complexité augmente singulièrement. En effet, une espèce signifie une équation de plus dans le système, mais également des termes d'interaction supplémentaires dans les équations déjà existantes.

Par exemple, si on considère un système dans lequel 3 espèces sont présentes, deux espèces de proies en compétition et un prédateur, voilà à quoi ressemblent le système d'équations :

$$\begin{cases} \frac{dX}{dt} &= r_1 X - \frac{r_1}{K_1} X^2 - c_{12} XY - c_{13} XZ \\ \frac{dY}{dt} &= r_2 Y - \frac{r_2}{K_2} Y^2 - c_{21} XY - c_{23} YZ \\ \frac{dZ}{dt} &= -mZ + c_{31} XZ + c_{32} YZ \end{cases}$$

Ce système de 3 équations comprend maintenant 11 paramètres : r_1, r_2, K_1, K_2, m, c_{12}, c_{21}, c_{13}, c_{23}, c_{31} et c_{32}.

Essayez de coder ce système et de produire une figure telle que la Figure 5.8.

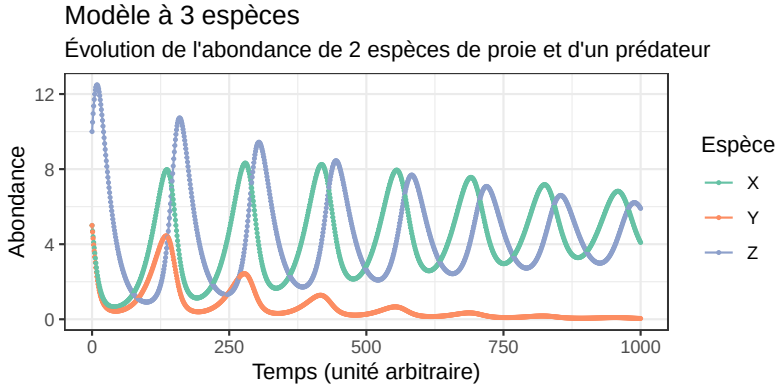


Figure 5.8: Évolution de l'abondance de 2 espèces de proies en compétition et d'une espèce prédatrice. Ici, les 2 espèces de proies ont les mêmes caractéristiques démographiques et sont peu compétitives. Le prédateur a un impact plus important sur l'espèce de proie 2 (abondance Y) et conduit donc à son extinction.

Pour comprendre ce modèle...

Expérimentez et tentez de répondre aux questions suivantes :

- Pouvez-vous trouver une combinaison de paramètres conduisant à la coexistence des 3 espèces ?
- Pouvez-vous trouver une combinaison de paramètres conduisant à la coexistence de 2 des 3 espèces ?
- Que se passe-t-il si on augmente légèrement la capacité biotique de la première espèce ? À quelle espèce cela devrait profiter le plus ? Est-ce ce que l'on observe ? Cela vous paraît-il logique ?
- Dans le système d'équation ci-dessus, êtes-vous capables d'identifier à quoi correspondent chacun des termes ? Quels sont les termes qui rendent

comptent de la compétition ? Quels sont les termes qui rendent compte de la prédation ?

6 Les Chaînes de Markov

6.1 Pré-requis

Comme pour chaque nouveau chapitre, je vous conseille de travailler dans un nouveau script que vous placerez dans votre répertoire de travail, et dans une nouvelle session de travail (Menu `Session > Restart R`). Inutile en revanche de créer un nouveau `Rproject` : vous pouvez tout à fait avoir plusieurs script dans le même répertoire de travail et pour un même `Rproject`. Comme toujours, consultez le livre en ligne du semestre 3 si vous ne savez plus comment faire.

Si vous êtes dans une nouvelle session de travail (ou que vous avez quitté puis relancé `RStudio`), vous devrez penser à recharger en mémoire les packages utiles. Dans ce chapitre, vous aurez uniquement besoin des packages du `tidyverse` (Wickham 2023), en particulier les package `dplyr` (Wickham et al. 2023), pour manipuler des tableaux, et `ggplot2` (Wickham et al. 2024) pour les représentations graphiques. Vous pourrez aussi avoir besoin du package `scales` (Wickham, Pedersen, et Seidel 2023) pour améliorer la mise en forme des axes des graphiques produits

```
library(tidyverse)
library(scales)
```

Nous n'aurons pas besoin ici d'importer de données depuis des fichiers externes : nous allons en effet générer directement des données à partir d'une chaîne de Markov.

Comme toujours, je spécifie une fois pour toutes le thème que j'utiliserai pour tous les graphiques de ce chapitre. Libre à vous de choisir un thème différent ou de vous contenter du thème proposé par défaut :

```
theme_set(theme_bw())
```

6.2 Les matrices dans R

Lors de votre découverte du logiciel R, de son fonctionnement et de sa syntaxe, vous avez appris qu'il existe plusieurs types d'objets que l'on peut créer et manipuler : des vecteurs, des facteurs, des data.frames ou tibbles, des listes, etc. L'un des types d'objets que nous n'avons pas encore décrit est la matrice.

Dans R, une matrice est un tableau dans lequel chaque cellule contient des éléments du même type. Contrairement aux data.frames et aux tibbles, une matrice ne peut donc pas avoir des chiffres dans une colonne, et des chaînes de caractères ou des vrais-faux dans une autre. De ce point de vue, une matrice est donc similaire à un vecteur qui ne peut contenir que des éléments qui sont tous du même type. Mais comme un data.frame ou un tibble, et contrairement à un vecteur, une matrice possède des dimensions : un nombre de lignes et un nombre de colonnes.

Les matrices sont donc des objets qui possèdent des points communs et des différences, avec les vecteurs et avec les data.frames ou tibbles.

6.2.1 Création

La façon la plus simple de créer une matrice est d'utiliser la fonction `matrix()`. Cette fonction possède de nombreux arguments et vous devrez au moins renseigner les suivants :

- `data` : quelles données souhaitez vous placer dans la matrice. Il peut s'agir d'une unique valeur si vous souhaitez que toutes les cellules de la matrice contiennent la même chose. En général, il s'agit d'un vecteur contenant les éléments que l'on souhaite placer dans la matrice.
- `nrow` : combien de lignes la matrice que vous souhaitez créer devra-t-elle contenir ?
- `ncol` : combien de colonnes la matrice que vous souhaitez créer devra-t-elle contenir ?
- `byrow` : souhaitez-vous que les données fournies dans `data` soient remplies en lignes (`byrow = TRUE`) ou en colonnes (`byrow = FALSE`, c'est le choix par défaut) dans la matrice ?

En général, on renseigne donc systématiquement `data`, et soit `nrow`, soit `ncol`. Il n'est pas nécessaire de renseigner à la fois `nrow` et `ncol` puisque `R` est capable de déterminer l'un des deux automatiquement selon la taille du vecteur que vous avez fourni à `data`. Par exemple, si vous fournissez un vecteur de 15 éléments à data, et que vous indiquez `nrow = 3`, `R` produira automatiquement une matrice de 3 lignes et 5 colonnes. Cela suppose évidemment que la longueur du vecteur fourni soit compatible avec les dimensions de la matrice que vous envisagez.

Vous pouvez évidemment spécifier explicitement le nombre de lignes et le nombre de colonnes souhaitées, mais alors la longueur du vecteur fourni devra être parfaitement compatible avec les dimensions de la matrice envisagée. Par exemple, si vous souhaitez une matrice de 4 lignes et 4 colonnes (soit 16 cellules), vous devrez fournir exactement 16 valeurs dans le vecteur transmis à l'argument `data`. Si vous en fournissez plus ou moins, des choses inattendues peuvent se produire :

- si vous fournissez trop de valeurs, les valeurs en trop ne seront pas intégrées à la matrice, votre vecteur sera donc tronqué après la seizième valeur
- si vous fournissez trop peu de valeurs, le recyclage sera utilisé pour combler la matrice qui ne peut pas avoir de cellule vide : les première valeurs du vecteur seront réutilisées jusqu'à ce que la matrice soit remplie, et un message d'avertissement vous sera renvoyé.

Voici quelques exemples :

```
# Création d'un vecteur de 8 éléments
a <- c(1, 5, 6, 9, 10, 5, 4, 2)  # 8 éléments

# Création d'une matrice de 2 lignes et 4 colonnes
matrix(data = a, nrow = 2)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    6   10    4
[2,]    5    9    5    2
```

```
# Une autre façon de créer la même matrice
matrix(data = a, ncol = 4)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    6   10    4
[2,]    5    9    5    2
```

Si la longueur du vecteur contenant les données n'est pas compatible avec les dimensions voulues pour la matrice, il faut spécifier les nombres de lignes et de colonnes de façon explicite, et un message d'avertissement est renvoyé.

```
# Pour créer une matrice de 3 lignes et 4 colonnes, il faut spécifier
# explicitement nrow et ncol
matrix(data = a, nrow = 3, ncol = 4)
```

```
Warning in matrix(data = a, nrow = 3, ncol = 4): data length [8] is not a
sub-multiple or multiple of the number of rows [3]
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    9    4    5
[2,]    5   10    2    6
[3,]    6    5    1    9
```

Ici, les premières valeurs du vecteur `a` ont été recyclées (1, 5, 6 et 9) pour remplir toutes les cellules de la matrice.

Dans tous les exemples ci-dessus, les chiffres contenus dans le vecteur `a` ont été remplis en colonnes dans la matrice. Pour remplir la matrice en lignes, on indique `byrow = TRUE` :

```
matrix(data = a, nrow = 2, byrow = TRUE)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    5    6    9
[2,]   10    5    4    2
```

```
matrix(data = a, ncol = 4, byrow = TRUE)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    5    6    9
[2,]   10    5    4    2
```

```r
matrix(data = a, nrow = 3, ncol = 4, byrow = TRUE)
```

```
Warning in matrix(data = a, nrow = 3, ncol = 4, byrow = TRUE): data length [8]
is not a sub-multiple or multiple of the number of rows [3]
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    5    6    9
[2,]   10    5    4    2
[3,]    1    5    6    9
```

Et bien sûr, comme vous le savez maintenant, si vous souhaitez pouvoir réutiliser les matrices créées avec la fonction `matrix()`, vous devez leur donner un nom avec l'oérateur `<-` :

```r
mat <- matrix(data = a, nrow = 3, ncol = 4, byrow = TRUE)
```

```
Warning in matrix(data = a, nrow = 3, ncol = 4, byrow = TRUE): data length [8]
is not a sub-multiple or multiple of the number of rows [3]
```

```r
mat
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    5    6    9
[2,]   10    5    4    2
[3,]    1    5    6    9
```

6.2.2 Produit matriciel

Le produit matriciel est l'opération qui permet de multiplier 2 matrices entre elles, ou de multiplier une matrice par un vecteur. Comme nous l'avons vu en cours, les dimension des objets que l'on multiplie sont importantes.

> **!** Produit matriciel
>
> Si A et B sont deux matrices, le produit matriciel $A \times B$ n'est possible que si le nombre de colonnes de A est égal au nombre de lignes de B.

Par ailleurs, sauf cas particulier, $A \times B \neq B \times A$

Dans R, l'opérateur qui permet d'effectuer un produit matriciel est %*%.

```
# Création d'une matrice A
A <- matrix(1:15, nrow = 5)
A
```

```
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
```

```
dim(A)
```

```
[1] 5 3
```

```
# Création d'une matrice B
B <- matrix(1:15, ncol = 5)
B
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    2    5    8   11   14
[3,]    3    6    9   12   15
```

```
dim(B)
```

```
[1] 3 5
```

```
# Produit matriciel A * B
A %*% B
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]   46  100  154  208  262
[2,]   52  115  178  241  304
[3,]   58  130  202  274  346
[4,]   64  145  226  307  388
[5,]   70  160  250  340  430
```

```r
# Produit matriciel B * A
B %*% A
```

```
     [,1] [,2] [,3]
[1,]  135  310  485
[2,]  150  350  550
[3,]  165  390  615
```

Vous constatez ici que les dimensions de la matrice obtenue sont variables et dépendent du sens du produit matriciel et des dimensions des matrices multipliées. Multiplier 2 matrices dont les dimensions sont incompatibles fait apparaître un message d'erreur :

```r
# Création d'une matrice C (4 lignes et 3 colonnes)
C <- matrix(1:12, ncol = 3)
C
```

```
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

```r
# Affichage de la matrice A (5 lignes et 3 colonnes)
A
```

```
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
```

```r
# Calcul des produits matriciels : impossibles !
A %*% C
```

```
Error in A %*% C: arguments inadéquats
```

```r
C %*% A
```

```
Error in C %*% A: arguments inadéquats
```

Enfin, il est tout à fait possible d'effectuer un produit matriciel entre une vecteur et une matrice. Dans `R`, les vecteurs n'ont pas de dimension, mais ils ont une longueur (*i.e.* un nombre d'éléments), que l'on peut afficher avec la fonction `length()`. Lorsque l'on effectue un produit matriciel impliquant un vecteur, le vecteur est automatiquement transformé soit en vecteur ligne (une matrice d'une seule ligne) soit en vecteur colonne (une matrice d'une seule colonne, voir le cours de "Fonctionnement des écosystèmes") :

- Si les vecteur est à gauche (`vec %*% mat`), il est transformé en vecteur ligne. Pour que le produit matriciel puisse être calculé, il faut alors que le nombre d'éléments de `vec` soit égal au nombre de lignes de `mat`.
- Si les vecteur est à droite (`mat %*% vec`), il est transformé en vecteur colonne. Pour que le produit matriciel puisse être calculé, il faut alors que le nombre d'éléments de `vec` soit égal au nombre de colonnes de `mat`.

```r
# Création d'un vecteur
vec <- c(2, 4, 3)
vec
```

```
[1] 2 4 3
```

```r
# Création d'une matrice D : 3 lignes et 2 colonnes
D <- matrix(c(5, 6, 8, 4, 2, 3), nrow = 3)
D
```

```
     [,1] [,2]
[1,]    5    4
[2,]    6    2
[3,]    8    3
```

```
# Création d'une matrice E : 2 lignes et 3 colonnes
E <- matrix(c(1, 2, 3, 9, 8, 7), nrow = 2)
E
```

```
     [,1] [,2] [,3]
[1,]    1    3    8
[2,]    2    9    7
```

```
# Calcul de différents produits matriciels
vec %*% D
```

```
     [,1] [,2]
[1,]   58   25
```

```
D %*% vec
```

```
Error in D %*% vec: arguments inadéquats
```

```
E %*% vec
```

```
     [,1]
[1,]   38
[2,]   61
```

```
vec %*% E
```

```
Error in vec %*% E: arguments inadéquats
```

Là encore, les dimensions de la matrice obtenue à l'issue du produit matriciel dépendent des dimensions des objets multipliés et de l'ordre de la multiplication.

6.3 Simulation d'écosystème

Nous avons vu dans le cours magistral de "Fonctionnement des écosystème" qu'il était possible de simuler l'évolution de plusieurs types de communautés végétales en utilisant une chaîne de Markov très simple. Il nous faut pour cela 2 objets :

1. une matrice de transition P : il s'agit d'une matrice carrée qui se lit en lignes (la somme de chaque ligne vaut 1) et qui contient les probabilités de transition d'un type de communauté végétale à un autre entre 2 observations successives de l'écosystème étudié (tous ses coefficients sont donc soit nuls, soit strictement positifs et inférieurs à 1).
2. un vecteur π_0 qui contient les proportions de chaque type d'écosystème au temps zéro (il s'agit donc des conditions initiales)

Pour connaître la situation de l'écosystème au temps 1 ($\pi 1$), il suffit d'effectuer un produit matriciel :

$$\pi_1 = \pi_0 \times P$$

Pour connaître la situation de l'écosystème au temps 2 (π_2), il suffit d'effectuer un produit matriciel :

$$\pi_2 = \pi_1 \times P = \pi_0 \times P \times P$$

Pour connaitre la situation de l'écosystème au temps t (π_t), il suffit d'effectuer un produit matriciel :

$$\pi_t = \pi_{t-1} \times P = \pi_0 \times P^t$$

En utilisant ce que vous venez d'apprendre sur les matrices d'une part (création et produits matriciels) et ce que vous avez appris sur les boucles `for` dans la Section 5.3, simulez l'évolution de l'écosystème décrit en cours. Vous prendrez soin de respecter l'ordre des étapes suivant :

1. Création de la matrice de transition `P`
2. Création du vecteur des conditions initiales `pi0`
3. Réglage de la durée de simulation à 50 pas de temps

4. Création d'une matrice remplie de 0 qui accueillera les résultats des calculs pour chaque étape de simulation
5. Placer `pi0` sur la première ligne de la matrice
6. Calcul des 50 produits matriciels à l'aide d'une boucle `for`
7. Transformation de la matrice de résultat en tibble grâce à la fonction `as_tibble()`. Attention, n'oubliez pas d'ajouter une colonne `Temps` pour indiquer le numéro des étapes de simulation (de 0 à 50)
8. Transformation du tibble au format long avec `pivot_longer()`
9. Représentation graphique des résultats obtenus avec `ggplot()`.

Pour mémoire, voilà les données utilisées dans le cours pour simuler l'écosystème méditerranéen d'intérêt et ses 5 communautés végétales.

- Conditions initiales `pi0` :

```
pi0
```

```
  Chêne   Vigne Pelouse Garigue  Pinède
   0.20    0.35    0.10    0.15    0.20
```

- Matrice de transition `P` :

```
P
```

```
         Chêne Vigne Pelouse Garigue Pinède
Chêne      0.8   0.2    0.00     0.0   0.00
Vigne      0.0   0.7    0.30     0.0   0.00
Pelouse    0.0   0.0    0.40     0.6   0.00
Garigue    0.0   0.0    0.00     0.2   0.80
Pinède     0.1   0.0    0.25     0.0   0.65
```

Le graphique que vous devriez être en mesure de produire est le suivant :

Pour comprendre ce système...

- Essayez de modifier les conditions initiales en choisissant d'autres valeurs pour `pi0`. Attention, tous les coefficients représentent des proportions. Ils

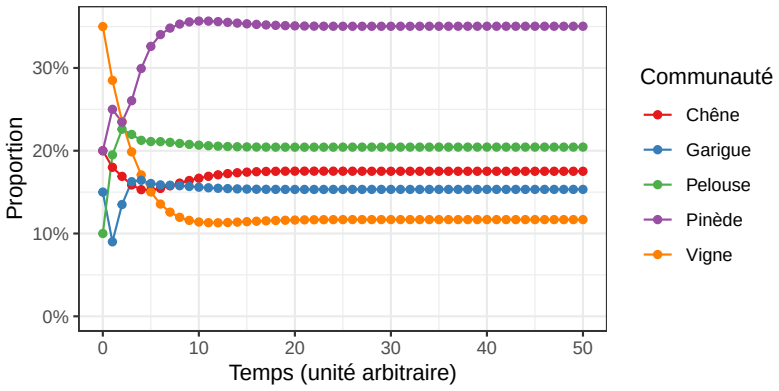


Figure 6.1: Évolution des proportions de 5 types de communautés végétales au sein d'un écosystème méditerranéen (50 pas de temps)

doivent donc obligatoirement être compris entre 0 et 1 et leur somme doit faire 1.

- Que se passe-t-il si les forêts de chênes occupent 100% de l'écosystème au temps 0 ? Est-ce que la situation d'équilibre est modifiée ? Cela vous semble-t-il logique ? Essayez de l'expliquer en revenant au cours et en examinant les conditions requises pour qu'un équilibre stable apparaisse. Est-ce que cela dépend de π_0 ?
- Reprenez les valeurs de départ pour `pi0`, mais changez maintenant les probabilités de transition de la matrice `P`. Attention, là encore, tous les coefficients doivent être compris entre 0 et 1 et la somme de chaque ligne doit faire exactement 1 (P est une matrice stochastique).
- Est-ce que l'état d'équilibre est modifié ? Est-ce logique ?
- En tant que gestionnaire de l'environnement, votre objectif est maintenant d'arriver à une situation où les forêts de chênes sont plus abondantes que les pinèdes dans cet écosystème. Quels sont vos leviers d'action ? Pouvez vous trouver des coefficients pour `P` qui reflètent ces changements ? À l'équilibre, cela se traduit-il par une dominance des forêts de chênes ?

References

Fox, John, Sanford Weisberg, et Brad Price. 2023. *car: Companion to Applied Regression*. https://r-forge.r-project.org/projects/car/.

Fuller, Andrea, Peter R. Kamerman, Shane K. Maloney, Graham Mitchell, et Duncan Mitchell. 2003. « Variability in brain and arterial blood temperatures in free-ranging ostriches in their natural habitat ». *Journal of Experimental Biology* 206 (7): 1171-81. https://doi.org/10.1242/jeb.00230.

Hasselquist, Dennis, James A. Marsh, Paul W. Sherman, et John C. Wingfield. 1999. « Is avian humoral immunocompetence suppressed by testosterone? » *Behavioral Ecology and Sociobiology* 45 (3): 167-75. https://doi.org/10.1007/s002650050550.

Horst, Allison, Alison Hill, et Kristen Gorman. 2022. *palmerpenguins: Palmer Archipelago (Antarctica) Penguin Data*. https://allisonhorst.github.io/palmerpenguins/.

Macdonald, Peter, et with contributions from Juan Du. 2018. *mixdist: Finite Mixture Distribution Models*. https://www.r-project.org/.

Waring, Elin, Michael Quinn, Amelia McNamara, Eduardo Arino de la Rubia, Hao Zhu, et Shannon Ellis. 2022. *skimr: Compact and Flexible Summaries of Data*. https://docs.ropensci.org/skimr/.

Wickham, Hadley. 2023. *tidyverse: Easily Install and Load the Tidyverse*. https://tidyverse.tidyverse.org.

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.

Wickham, Hadley, et Jennifer Bryan. 2023. *readxl: Read Excel Files*. https://readxl.tidyverse.org.

Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke,

Kara Woo, Hiroaki Yutani, Dewey Dunnington, et Teun van den Brand. 2024. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. https://ggplot2.tidyverse.org.

Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, et Davis Vaughan. 2023. *dplyr: A Grammar of Data Manipulation*. https://dplyr.tidyverse.org.

Wickham, Hadley, Jim Hester, et Jennifer Bryan. 2024. *readr: Read Rectangular Text Data*. https://readr.tidyverse.org.

Wickham, Hadley, Thomas Lin Pedersen, et Dana Seidel. 2023. *scales: Scale Functions for Visualization*. https://scales.r-lib.org.

Young, Kevin V., Edmund D. Brodie, et Edmund D. Brodie. 2004. « How the Horned Lizard Got Its Horns ». *Science* 304 (5667): 65-65. https://doi.org/10.1126/science.1094790.