Le traitement de données comportementales – un tour d'horizon avec des exemples traités par **R**

©Christian Jost - jost@cict.fr - 2008/2009
document en cours de développement,
merci de me signaler les erreurs

15 avril 2010

Avant-propos

Ce document est un petit tour d'horizon des situations expérimentales qu'on peut rencontrer dans l'étude du comportement et des traitements statistiques à appliquer dans ces cas. Le but ne sera pas de vous expliquer les théories sous-jacentes à ces méthodes statistiques (il y a des livres qui font cela beaucoup mieux que je pourrais le faire, voir Scherrer (1984) ou Zar (1999)) mais de donner quelques notions clés nécessaires pour leur application, l'interprétation des résultats et pour les communiquer dans un travail écrit. Chaque méthode sera suivi d'un petit exemple concret et de quelques références pour aller plus loin.

Le contenu est regroupé en quatres chapitres. Le premier rappelle le type de données qu'on peut rencontrer et comment les représenter graphiquement. Le second concerne les statistiques descriptives et l'estimation des intervalles de confiance. Le troisième chapitre fait le tour d'horizon des tests d'hypothèses les plus répandus. Enfin, le quatrième chapitre fait une petite introduction à la modélisation statistique, en particulier les régressions linéaires et non-linéaires. D'un point de vue longueur le chapitre trois prend clairement le plus de place, mais je pense que le chapitre deux est bien plus important parce que plus fondamental. Donc, n'hesitez pas de revenir dans ce chapitre si des notions tel que précision, erreur standard ou intervalle de confiance vous font douter.

Ce document devrait vous servir comme un point de départ qui vous permette de trouver le traitement statistique adapté à votre question biologique et à votre dispositif expérimental, ou au moins de vous aiguiller dans la bonne direction. Toute suggestion pour l'améliorer par rapport à ces buts sera la bienvenue [1].

Le traitement statistique ce fait de nos jours à l'aide d'ordinateurs, aussi bien pour la préparation que pour l'analyse des données. Les exemples seront donc faits avec un logiciel libre spécifiquement développé pour les analyses statistiques, **R** [2]. Un fichier avec les données des exemples est disponible sur internet [3] (voir chapitre A.2 pour les détails de l'installation). Vous verrez rapidement que **R** n'est pas un logiciel aussi simple à utiliser que certains logiciels statistiques que vous allez rencontrer dans les laboratoires où vous allez peut-être faire un stage (par exemple Systat, SPSS, S-Plus, ...). Donc, pourquoi **R** et non pas un logiciel plus simple ? Il y a trois raisons : a) les logiciels "simples" sont payants, vous ne pouvez donc pas les installer sur vos ordinateurs sans payer une licence, b) si vous avec compris l'organisation des données et l'interprétation des analyses dans **R** vous allez comprendre intuitivement aussi l'utilisation de tous les logiciels mentionnés ci-dessus, et c) **R** est aussi un environnement de programmation qui vous permet de faire des petits modèles et des expériences virtuelles, chose très utile par exemple pour estimer la taille de l'échantillon nécessaire pour mettre en évidence un effet si vous disposez des données d'une pré-expérience. Si vous n'avez pas (encore) cette ambition c) contentez-vous de recopier les commandes **R** et concentrez-vous sur l'interprétation des sorties, cela suffira largement par rapport aux buts de ce document. Sinon, l'annexe A vous donnera une rapide introduction dans l'utilisation de **R** pour organiser et analyser vos propres données.

1. jost@cict.fr
2. `http://www.r-project.org/`
3. `http://cognition.ups-tlse.fr/vas-y.php?id=chj`

Sommaire

Chapitre 1

Le type de données et leur représentation graphique

On distingue plusiers types de données : nominales, ordinales, continues et discrètes. Pour chaque type il y a des visualisations graphiques et des traitements statistiques appropriés. Dans ce chapitre on va voir le graphisme.

1.1 Les données nominales

Il s'agit d'un trait qualitatif tel que le sexe d'un cerf ou l'appartenance d'une guêpe à une caste. Ce qui nous intéresse sont les fréquences absolues (par exemple 10 mâles et 15 femelles dans un groupe de marmottes) ou les fréquences relatives (on a une proportion de $10/25 = 0.40$ [1] mâles dans notre groupe). On peut rapporter ces données sous forme d'un tableau (fréquences absolues ou fréquences relatives avec le nombre d'observations) ou d'un diagramme en bâton (Fig 1.1). Les diagrammes en forme de "camembert" ne sont pas recommandés parce qu'ils sont plus difficiles à interpréter (par exemple déterminer la catégorie la plus fréquente).

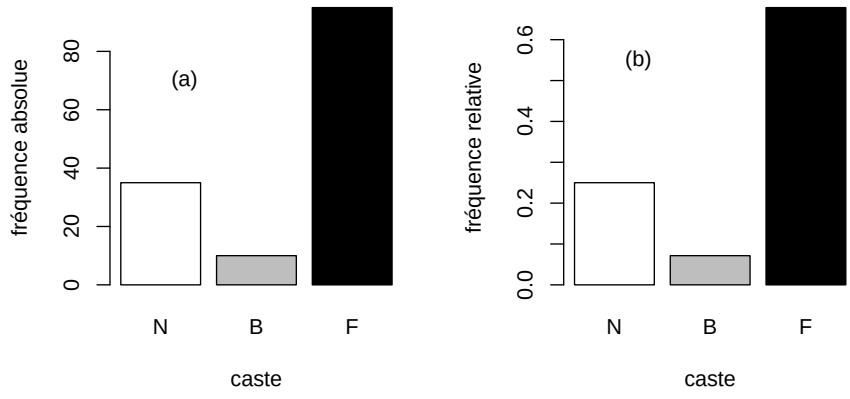

FIGURE 1.1 – Représentation de données nominales : les fréquences absolues (a) et relatives (b) d'appartenance à 3 castes dans une colonie de 140 guêpes (N = nourrices, B = bâtisseuses, F = fourrageuses).

Exemple Prenons l'exemple de la Fig 1.1. On crée d'abord un vecteur contenant les fréquences, ensuite on utilise la commande `barplot` pour créer le diagramme en bâtons :

1. Dans ce document, je vais utiliser le point décimal au lieu de la virgule parce que **R** suit les conventions internationales habituelles des logiciels libres, en particulier il utilise la syntaxe anglophone des décimaux.

```
R> castes = c(35,10,95);
R> barplot(castes/sum(castes),names.arg = c("N","B","F"),xlab="caste")
```
Maintenant on va ajouter les fréquences relatives d'une plus vieille colonie :
```
R> castesV = c(100,95,230);
R> lesCastes = rbind(castes/sum(castes),castesV/sum(castesV));
R> lesCastes
R> barplot(lesCastes,beside=T, names.arg=c("N","B","F"),
+  col = c("red","blue"))     # ne pas taper le '+', il indique la nouvelle ligne
R> legend(2,0.5,legend=c("jeune","vieille"),fill=c("red","blue"))
```
Remarque concernant **R** : les deux premiers chiffres de la commande `legend` indiquent les coordonnées dans le graphique où vous vouler mettre cette légende. Vous pouvez trouver ces coordonnées avec la commande `locator(1)` et en cliquant sur le bon endroit.

1.2 Les données ordinales

Comme pour les données nominales, c'est un trait qualitatif, mais les catégories se caractérisent par un ordre naturel. Par exemple, on peut distinguer des classes d'âge comme 'larve' et 'adulte', ou des degrées d'intensité d'un comportement comme 'calme', 'agité' et 'paniqué'. On rapporte ces données sous forme d'un tableau ou d'un diagramme en bâtons (comme dans la Fig 1.1), mais en respectant cet ordre naturel.

Exemple On observe les interactions entre les lézards sur une île pendant deux saisons, automne et printemps, en distinguant les catégories "neutre" (1), "agréssive" (2) et "très agressive" (3). Voici les observations :
```
R> automne = c(2,2,1,2,1,1,2,1,1,3)                  # comportement de 10 lézards
R> hiver = c(1,3,2,2,3,1,3,3,2,2)                    # comportement de 10 lézards
R> au = table(automne);
R> hi = table(hiver);
R> tab = rbind(au/sum(au),hi/sum(hi));
R> tab                                                  # afficher le résultat
R> barplot(tab,beside=T,legend.text=c("automne","hiver"),
+  names.arg=c("Neutre","Aversif","Très agressif"))                          #
```

1.3 Les données continues et discrètes

Finalement, on a souvent des données chiffrées telles que le poids d'un canari ou le temps de réaction d'un capucin. Ce sont des données continues qu'on peut représenter par un histogramme ou par une "boîte à moustache" (voir Fig 1.2). Si on dispose de beaucoup de données l'histogramme donne le plus d'informations (forme de la distribution, unimodale ou bimodale, symétrique ou asymétrique, ...), mais si on a peu de valeurs ou si on veut représenter plusieurs jeux de données dans le même graphique les boîtes à moustaches sont plus adaptées. Pour clairement indiquer que l'histogramme représente la distribution empirique d'une variable continue il n'y a jamais d'espace entre ces 'bâtons', contrairement au diagramme en bâtons pour les données nominales ou ordinales.

Les données discrètes tel que le nombre de neurones dans un cerveau ont en commun avec les données continues qu'on peut les comparer (4000 neurones est le double de 2000 neurones, avec des données ordinales cela ne serait pas possible : dans l'exemple ci-dessus, un lézard "très agressif" n'est pas trois fois plus agressif qu'un lézard "neutre"). Si les nombres sont grands on peut les traiter comme des données continues. Si les nombres sont plus petits on peut les représenter comme les données ordinales.

Avec les données continues il y a souvent trop de valeurs pour les rapporter tous : on se contente de donner un **graphique des données** et des mesures de **tendance centrale** et de **dispersion** (précisant le nombre de données utilisées). Ces mesures sont des paramètres statistiques. En général, on appelle toute grandeur calculée à partir d'un jeu de données un **paramètre statistique** ou simplement une **statistique** ('statistic' en anglais). Dans les boîtes à moustaches la tendance centrale est la médiane (autant de valeurs au-dessous qu'au-dessus), la boîte est formée par le premier quartile (un quart des valeurs au-dessous, trois quarts au-dessus) et le troisième quartile,

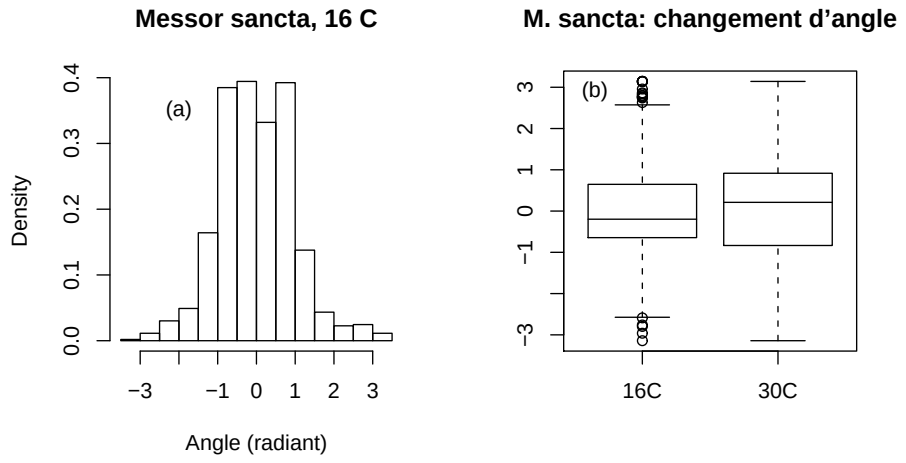

FIGURE 1.2 – Distribution des angles lorsqu'une fourmi *M. sancta*, en déplacement dans un milieu homogène, change de direction. (a) histogramme des angles à 16°C, on observe que l'histogramme est symétrique et que les fourmis ont une tendance à continuer quasiment dans la même direction. (b) la comparaison entre 16°C et 30°C montre que la dispersion des angles est plus grande à 30°C.

et les "moustaches" sont formées par le minimum et le maximum (les quelques points isolés sont considérés être des mesures extrêmes, c'est-à-dire les valeurs qui sont plus éloignées du 1^{er} ou $3^{ème}$ quartile que 1.5 fois la hauteur de la boîte). Ce sont des mesures **non-paramétriques**.

Les mesures du type **paramétrique** sont la moyenne pour la tendance centrale et l'écart-type pour la dispersion. On les appelle 'paramétrique' parce qu'elles sont souvent associées à des distributions bien précises telle que la distribution normale ou log-normale.

Pour des données continues avec un zéro absolu (tel que le nombre de neurones dans le cerveau de différentes espèces, poids d'une fourmi, ..., mais pas la température en °C) on observe souvent que l'écart-type augmente de façon proportionnelle à la moyenne. Dans ce cas le **coefficient de variation** CV=écart-type/moyenne donne une bonne description de la forme de la distribution.

Exemple Prenons les données de la Fig 1.2. Assurez-vous que le répertoire de travail dans **R** est bien celui dans lequel se trouvent les fichiers que vous avez téléchargés pour ce document (voir chapitre A.2).

```
R> getwd(); list.files();          # vérification que R est dans le bon répertoire
R> load("angleMSancta.rda")                                  # charger les données
R> ls()                                  # vous devriez voir 'angle16' et 'angle30'
R> summary(angle16)              # les quartiles et la moyenne, comparer à angle30
R> mean(angle16); sd(angle16)                                # moyenne et écart-type
R> sd(abs(angle16))/mean(abs(angle16))                  # le CV des angles absolus
R> hist(angle16)
R> boxplot(list("16C"=angle16,"30C"=angle30))
```

Pour aller plus loin Le premier chapitre dans Zar (1999) donne plus d'exemples et de détails sur les type de données.

Un cas spécial : les durées d'évènements

Reprenons les trajectoires des fourmis *Messor sancta*. Dans la Fig 1.2 on a visualisé la distribution des angles quand la fourmi change de direction. Un autre aspect de ces trajectoires et la distribution des longueurs de trajets rectilignes avant changement de direction (on appelle cela les « libres parcours »). Quelle est la "meilleure" façon de visualiser ces libres parcours ? Pour ces données continues on pense d'abord à une boîte à moustache (mais qui, dans ce cas, ne permet que de voir l'asymétrie de la distribution) ou à un histogramme (qui nous donne surtout l'information

qu'il y a beaucoup de petites valeurs, voir Fig 1.3). Mais ni l'un ni l'autre ne nous donne vraiment une idée de la "vraie" distribution de ces données.

Une autre façon de visualiser ces données nous est fournis par l'analyse de survie [2]. On "fait commencer" tous les évènements au temps $t = 0$ et on trace ensuite le nombre (ou la proportion) d'évènements qui durent encore en fonction du temps écoulé, voir Fig 1.3 (comme nos fourmis ont une vitesse quasiment constante la durée d'un déplacement rectiligne est proportionnelle à la longueur du libre parcours).

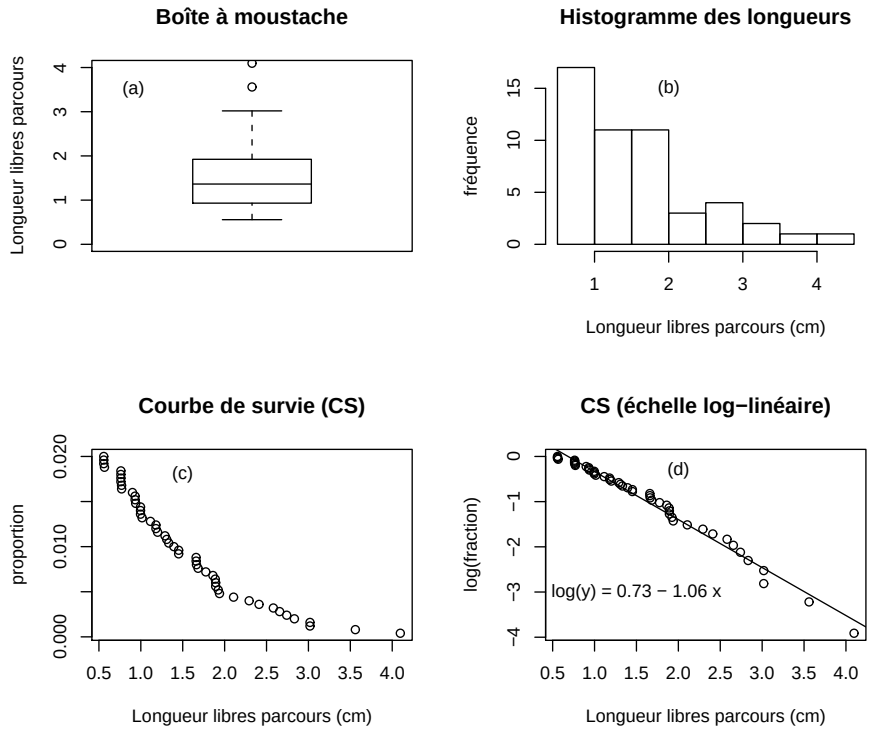

FIGURE 1.3 – Visualisation de durées d'évènements à l'exemple du déplacement droit (en cm) d'une fourmi avant de changer de direction (à 16 °C). (a) boîte à moustache des 50 mesures, (b) histogramme des mesures, (c) courbe de survie en échelle linéaire et (d) courbe de survie en échelle log-linéaire (avec la droite de régression linéaire).

Notons en particulier la représentation en échelle log-linéaire de la courbe de survie (Fig 1.3d). Dans notre exemple elle a l'allure d'une ligne droite. Ceci est important, parce que cela indique que la probabilité par unité de temps (ou unité de distance dans notre cas) de changer de direction ne change pas au cours du temps : c'est un **processus sans mémoire**, et les durées des déplacements rectilignes sont distribuées selon une loi exponentielle.

Exemple

```
R> load("lpMessorSancta.rda")                              # charger les données
R> source("fonctionsCompStat.R")           # charger la fonction 'survivalCurve'
R> libParc.16.30                                  # afficher les données brutes
R> boxplot(libParc.16.30)                  # représentation en boîte à moustache
R> lp16 = libParc.16.30$lp16cm
R> hist(lp16)                                    # représentation en histogramme
R> survivalCurve(lp16,xlab="libre parcours (cm)",              # courbe de survie
+ main = "Courbe de survie",fitLine=T)                                      #
```

Pour aller plus loin Le livre de Haccou & Meelis (1992) donne une introduction extensive dans

2. Ce type d'analyse a été initialement développé pour les études médicales de l'évolution d'une maladie, la durée d'un évènement étant le temps avant récupération ou la mort de l'individu (d'où le nom "analyse de survie").

l'analyse de durées de comportements.

Chapitre 2

Estimation : des moyennes et des erreurs standards

2.1 La population statistique et l'échantillon

Les notions de **population statistique** et **échantillon** sont au cœur de la raison d'être des traitements statistiques. Par exemple, vous voulez décrire la distribution des temps de réaction des étudiants au niveau M1 (il y a des logiciels qui permettent de mesurer ce temps de réaction). On dispose d'un jeu de données $(tr_1, tr_2, \ldots, tr_n)$ obtenu avec les n étudiants du M1 BioSanté à Toulouse. On calcule la moyenne empirique $\bar{tr}$ et l'écart type s_{tr}, donc deux paramètres statistiques du type paramétrique. Mais $\bar{tr}$ et s_{tr} ne représentent qu'une partie de ce qui nous intéresse réellement : la moyenne μ et écart-type σ du temps de réaction de tous les étudiants M1 dans le monde.

C'est la situation typique de tous les jours, on veut savoir quelques chose sur toute une population statistique (les étudiants M1 dans le monde), mais on ne dispose que d'un échantillon (les mesures sur la promotion M1 en BioSanté 2004). Notre moyenne empirique $\bar{tr}$ n'est qu'une **estimation** de la vraie moyenne μ, et s_{tr} une **estimation** du vrai écart-type σ. [1]

Estimation sur un échantillon de données continues

Une première conséquence de cette information partielle (échantillon au lieu de la population entière) est le calcul d'un intervalle de confiance dans lequel se trouve 95% des mesures de temps de réaction (IC_{tr}). Si on connaissait μ et σ ce calcul serait

$$IC_{tr} = (\mu - 1.96\,\sigma, \mu + 1.96\,\sigma)$$

mais le fait d'avoir seulement $\bar{tr}$ et s_{tr}, qui ne sont qu'une estimation de μ et σ, ajoute plus d'incertitude et il faut remplacer le facteur 1.96 par un facteur plus grand qui dépendra du nombre n de données desquelles s_{tr} a été calculé. Ce nouveau facteur suit une distribution de Student avec degrés de liberté $df = n-1$: pour l'exemple des temps de réaction $n = 24$ et ce facteur est de 2.07. L'intervalle de confiance des temps de réaction IC_{tr} est donc $(\bar{tr} - 2.07\,s_{tr}, \bar{tr} + 2.07\,s_{tr})$ indiqué en rouge au dessus de l'histogramme (Fig 2.1).

Une seconde conséquence est qu'il faut aussi spécifier quelle est l'incertitude de $\bar{tr}$, c'est-à-dire s'il est proche de μ ou pas. Pour cela on construit à partir de $\bar{tr}$, s_{tr} et n un intervalle de confiance dans lequel se trouve le vrai μ avec 95% de chance, $IC_{\bar{tr}}$,

$$\begin{aligned} IC_{\bar{tr}} &= (\bar{tr} - 2.07\,s_{\bar{tr}}, \bar{tr} + 2.07\,s_{\bar{tr}}) \\ s_{\bar{tr}} &= \frac{s_{tr}}{\sqrt{n}} \end{aligned}$$

On appelle $s_{\bar{tr}}$ l'**erreur standard** de $\bar{tr}$. L'intervalle de confiance $IC_{\bar{tr}}$ est dessiné dans la Fig 2.1 en bleu. Evidemment, plus l'échantillon et grand et plus l'erreur standard sera petite, c'est-à-dire meilleure est la précision de $\bar{tr}$).

1. Pour bien distinguer les paramètres au niveau population de ceux au niveau échantillon on utilise d'habitude des lettres grecques pour les premiers (par exemple μ et σ) et des lettres latines pour les seconds ($\bar{t}$ et s).

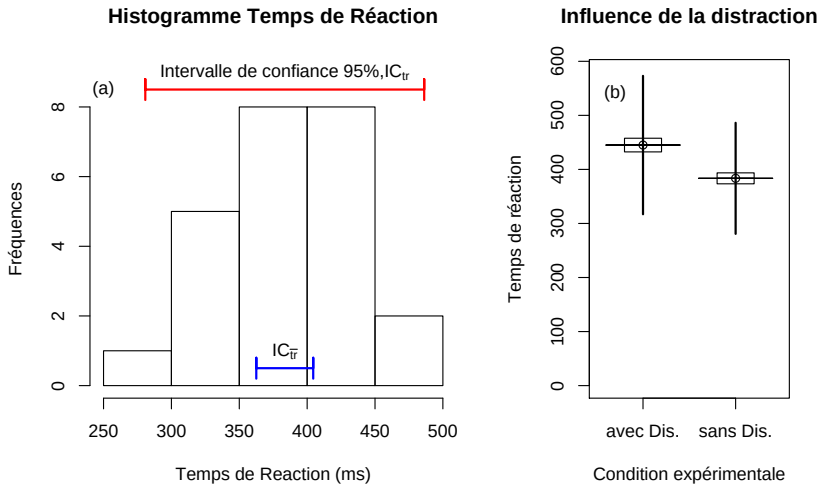

FIGURE 2.1 – (a) L'histogramme des temps de réaction, l'intervalle de confiance à 95% de la population (rouge en haut) et de l'estimation de la moyenne (bleu en bas). (b) Les temps de réaction (en ms) avec et sans distraction : le rectangle indique la moyenne±erreur standard, la ligne verticale représente l'intervalle de confiance à 95% de la population. Veuillez noter que cette représentation a seulement un but pédagogique, d'habitude on ne tracerait qu'une ligne verticale indiquant l'erreur standard.

Pour rapporter ces valeurs on note soit **moyenne±écart-type** (si on s'interesse à la dispersion de la population), soit **moyenne±erreur standard** (si on s'intéresse à la précision de la moyenne estimée). Il faut toujours ajouter la taille de l'échantillon n.

Remarque importante : la construction de IC_{tr} suppose que les données dans la population sont distribuées de façon normale, ne l'utilisez donc que dans le cas ou l'histogramme de l'échantillon est compatible avec une distribution normale (on verra dans le prochain chapitre comment le vérifier). Par contre, le calcul de $IC_{\bar{tr}}$ est quasiment toujours juste si la taille de l'échantillon est suffisament grand. C'est un corollaire d'un théorème fondamentale en statistique, le théorère de la limite centrale.

Exemple Prenons l'exemple des temps de réaction :

```
R> tempsReaction = read.table("tempsReaction2004.txt",h=T)
R> names(tempsReaction);tr = tempsReaction$simple.sans.dist
R> moy = mean(tr); et = sd(tr)
R> t95 = qt(0.975,df=(length(tr)-1))          # calcul du facteur t_{0.95,df} pour l'IC
```

Vous vous posez surement la question pourquoi l'argument de `qt` est 0.975 et ne pas 0.95. En fait, pour obtenir un intervalle de confiance de 95% on coupe les « queues » à gauches et à droites de la distribution, enlevant 2.5% des valeurs à gauche et 2.5% des valeurs à droite. `qt` est l'inverse de la distribution cumulée (les quantiles), `qt(0.975, ...)` indique donc où commence cette queue de droite et `-qt(0.975,...)=qt(0.025,...)` où commence celle de gauche.

```
R> c(moy - t95 * et, moy + t95 * et)                          # calcul de IC_tr
R> es = et/sqrt(length(tr))                          # calcul de l'erreur standard
R> c(moy - t95 * es, moy + t95 * es)                          # calcul de IC_trbar
```

Rapporter les résultats Le temps de réaction des étudiant au niveau M1 est de 383.5 ± 10.1 ms (moyenne ± erreur standard, n=24).

Pour aller plus loin Le chapitre 7 de Zar (1999) donne tous les détails sur les intervalles de confiance dans le contexte des tests d'hypothèse sur un seul échantillon (voir ci-dessous). Par rapport à la question si n est suffisament grand pour que le calcul de $IC_{\bar{tr}}$ soit juste même si l'échantillon n'est pas du tout distribué de façon normale (les matheux évoquent dans ce contexte le théorème de la limite centrale) je vous recommande Boos & Hughes-Oliver (2000), un article

récent et très pédagogique.

Estimation sur un échantillon de données nominales ou ordinales

Le problème échantillon ↔ population se pose aussi quand on regarde le nombre d'évènements, ne. Mais pour les fréquences relatives on dispose d'une information supplémentaire : la taille de l'échantillon, n. Ainsi, si on a une fréquence relative $p = ne/n$ tel que $n\,p \geq 5$, on peut estimer l'écart-type associé par une approximation normale,

$$s_p = \sqrt{\frac{p(1-p)}{n-1}}$$

et l'intervalle de confiance à 95% devient

$$IC_p = (p - t(0.95, n-1)\, s_p, p + t(0.95, n-1)\, s_p)$$

où $t(0.95, n-1)$ est le facteur mulitipicatif distribué selon une distribution de Student avec $df = n-1$ degrés de liberté.

On peut également calculer l'intervalle de confiance pour les fréquences absolues $ne = p\,n$:

$$\begin{aligned} s_{ne} &= \sqrt{n\,p\,(1-p)} \\ IC_{ne} &= (ne - t(0.95, n-1)\, s_{ne}, ne + t(0.95, n-1)\, s_{ne}) \end{aligned}$$

Exemple Reprenons l'exemple du nombre de guêpes dans une caste (Fig 1.1). On avait

```
R> castes=c(35,10,95)
R> n = sum(castes);                                    # la taille de l'échantillon
R> pN = castes[1]/n                                    # la proportion de nourrices
R> sdp = sqrt(pN*(1-pN)/n);                                          # écart-type
R> t95=qt(0.975,n-1);                              # facteur multiplicatif pour IC_p
R> c(pN-t95*sdp,pN+t95*sdp)                       # l'intervalle de confiance de pN
R> barplot(castes/sum(castes),names.arg = c("N","B","F"),xlab="caste",
+  ylab="fréquence absolue")                                  # diagramme en bâton
R> lines(c(0.7,0.7),c(pN-sdp,pN+sdp))                       # ajout de l'écart-type
```

Remarque spécifique pour **R** : pour trouver la coordonnée $x = 0.7$ où dessiner l'écart-type j'ai utilisé la commande

```
R> locator(1)
```

qui, en cliquant sur l'endroit du graphique où je veux centrer ma ligne, me donne les coordonnées (x, y) de cet endroit. Essayez.

Pour aller plus loin Si on a $np < 5$ l'approximation normale ne tient plus et il faut faire un calcul plus sophistiqué (voir exemple 24.5 dans Zar 1999) qui donnera un intervalle de confiance asymétrique.

2.2 Le bootstrap : une méthode universelle pour estimer l'erreur standard

La formule ci-dessus pour l'erreur standard de la moyenne (et l'$IC_{\bar{tr}}$) ou d'une proportion est un résultat statistique non-trivial. Pour beaucoup d'autres paramètres statistiques qu'on peut calculer à partir d'un échantillon une telle formule n'existe pas pour calculer la précision de ce paramètre. Un exemple classique est le coefficient de variation CV. Comment s'en sortir ? C'est grâce aux ordinateurs et leur puissance de calcul qu'il existe aujourd'hui une méthode universelle pour calculer l'erreur standard d'un paramètre statistique quelconque : le **bootstrap**. L'idée est simple :

1. créer un jeu de données bootstrap en tirant au hasard (avec remise) n valeurs dans l'échantillon original (de taille n),
2. calculer le paramètre statistique sur ce jeu de données bootstrap,

3. répéter les deux premières étapes au moins 300 fois (cela suffit d'habitude pour une estimation de l'erreur standard) et stocker les paramètres statistiques du bootstrap dans un vecteur,
4. le résultat fondamentale de la théorie du bootstrap nous dit maintenant que l'écart-type de ce vecteur est une estimation fiable de l'erreur standard du paramètre statistique.

Vous voyez pourquoi on a besoin d'un ordinateur : répéter le même calcul 300 fois à la main prendrait du temps. La première partie dans l'algorithme ci-dessus est à l'origine du nom 'bootstrap' : comme le Baron de Münchhausen, étant coincé dans un marécage, se sauvait en tirant sur ses propres lacets sans appui extérieur, la méthode du bootstrap estime la précision d'un paramètre statistique calculé sur un échantillon on s'appuyant sur le même échantillon.

Exemple Reprenons les donnés du temps de réaction et calculons l'erreur standard du coefficient de variation :

```
R> tempsReaction = read.table("tempsReaction2004.txt",h=T)
R> tr = tempsReaction$simple.sans.dist
R> cv = sd(tr)/mean(tr)                    # calcul du coefficient de variation cv
R> cvBS={}            # initaliser la variable qui contiendra les CV du bootstrap
R> for (k in 1:300) {
+  trBS = sample(tr,length(tr),replace=T);    # tirage du graphique de données BS
+  cvBS[k] = sd(trBS)/mean(trBS);                        # calcul du CV bootstrap
+ }                                                                          #
R> se=sd(cvBS);se                          # calcul de l'erreur standard de cv
R> c(mean(cvBS)-qt(0.975,length(tr)-1)*se, mean(cvBS)+qt(0.975,length(tr)-1)*se)
R> hist(cvBS);                     # calcul de l'IC et affichage de l'histogramme
```

Rapporter les résultats Le temps de réaction des étudiants en M1 à un coéfficient de variation $CV = 0.130 \pm 0.020$ (erreur standard, $n = 24$).

Pour aller plus loin Une introduction très lisible aux méthodes de bootstrap et ses applications est sans doute Efron & Tibshirani (1993). Pour ceux avec une fibre plus statistique on peut aussi recommander Davison & Hinkley (1997).

Exemple 2 pour le bootstrap : estimation de la durée moyenne d'un évènement

Reprenons l'exemple de longueurs des libres parcours d'une fourmi *Messor sancta* du chapitre 1.3. Rappelons que les mesures représentent les longueurs des bouts de trajectoires d'une fourmi avant changement de direction. En divisant ces longueurs par la vitesse moyenne d'une fourmi à cette température (0.8 cm/s) on obtient les durées qu'une fourmi va tout droit.

Nous voulons maintenant calculer la durée moyenne et son erreur standard. D'après ce que nous avons dit ci-dessus il suffit donc de calculer la moyenne arithmétique et l'erreur standard de notre échantillon. Ceci serait vrai si on pouvait estimer la durée d'un trajet rectiligne avec une précision parfaite, en particulier les durées courtes. Cependant, la nature du mouvement des fourmis (sinueux à cause des six pattes) et les problèmes liés à la trajectométrie (résolutions d'une caméra numérique, détermination des coordonnées d'une fourmi) fait qu'on ne peut pas mesurer des durées au-dessous d'un certain seuil (dans notre exemple ce seuil est une demi-seconde). Ajoutons à cela que les durées sont distribuées de façon exponentielle (voir chapitre 1.3) : cela veut dire que de nombreux libres parcours sont très courts et manqueront dans notre jeu de données. Il est donc évident que la moyenne arithmétique de l'échantillon surestime la vraie moyenne, c'est un estimateur **biaisé**.

Heureusement, dans le cas d'un processus sans mémoire (avec la distribution exponentielle des durées d'évènements) il existe un estimateur sans biais. Dans la Fig 1.3d on a vu que la courbe de survie d'un tel processus est une droite en échelle log-linéaire. La valeur absolue de la pente de cette courbe représente le taux de fin d'évènement, et son inverse est directement la durée moyenne estimée sans biais.

Exemple On utilise les mêmes données comme dans l'exemple du chapitre 1.3, les libres parcours à 16°C :

```
R> load("lpMessorSancta.rda")                                  # charger les données
```

```
R> source("fonctionsCompStat.R")            # charger la fonction 'survivalCurve'
R> lp16sec = libParc.16.30$lp16cm/0.8                  # calcul des durées (en s)
R> mean(lp16sec)                                          # moyenne arithmétique
R> res=survivalCurve(lp16sec,fitLine=T)                     # calcul de la pente
R> taux=abs(res$coefficients[[2]]);taux # le taux de changer de direction, en s^-1
R> moySansBiais=1/taux;              # estimation non-biaisée de la durée moyenne
R> moySansBiais
```

On voit que cette durée moyenne de 1.18 s est bien inférieur à la moyenne arithmétique de 1.96 s. L'importance du biais devient évident quand on calcule l'intervalle de confiance de la moyenne par un bootstrap :

```
R> nbBS = 100;                              # nombre de répétitions boostrap à faire
R> durMoy = rep(0,nbBS);     # préparer le vecteur qui contiendra les estimations
R> for (k in 1:nbBS) {
+  lp16BS = sample(lp16sec,length(lp16sec),replace=T);  # graphique de données bs
+  res = survivalCurve(lp16BS,fitLine=T,plotCurve=F);              # calcul pente
+  durMoy[k] = 1/abs(res$coefficients[[2]]);     # durée moyenne jeu de données k
+ }                                                                            #
R> es = sd(durMoy);es                              # estimation de l'erreur standard
```

On peut maintenant calculer l'intervalle de confiance à 95% :

```
R> t95 = qt(0.975,length(lp16sec)-1)             # calcul du facteur multiplicatif
R> c(moySansBiais - t95*es,moySansBiais + t95*es)                          # IC_0.95
```

La moyenne arithmétique de 1.96 s est largement en dehors de cette intervalle de confiance, l'absence de libres parcours courts a significativement biaisé l'estimation par moyenne arithmétique.

Chapitre 3

Les tests d'hypothèses

Dans les deux premiers chapitres on n'a fait que des statistiques descriptives. Le présent chapitre parlera des statistiques inférentielles, c'est-à dire que l'on se met à tester des hypothèses. Tout commence alors avec la formulation d'une hypothèse, par exemple que les Toulousains ne sont pas de même taille que les Nantais.

La philosophie des tests d'hypothèse s'inspire maintenant de celle de K. Popper [1] qui postulait, de façon simplifiée, qu'il est impossible de prouver que quelques chose est vrai, mais seulement que quelques chose est faux. On devrait donc s'approcher de la vérité en éliminant tout ce qui est faux, une hypothèse qui résiste aux tests expérimentaux n'est donc pas « confirmée » (inductionisme) mais « corroborée ». Dans l'exemple de la taille des Toulousains et Nantais on essaye donc de réfuter le contraire de l'hypothèse d'origine, ce qu'on appelle l'**hypothèse nulle** ou H_0 : "Les Toulousains ont la même taille que les Nantais". Ou, si on note par $\mu_{taille,T}$ la taille moyenne des Toulousains et par $\mu_{taille,N}$ celle des Nantais, on a

$$H_0 : \mu_{taille,T} = \mu_{taille,N}$$

Il suffit maintenant de mesurer tous les Toulousains et tous les Nantais pour regarder si leur taille moyenne est la même, sinon on rejette H_0 et on *corrobore* l'hypothèse du début.

Cela semble simple, sauf qu'il est impossible de mesurer la taille de **tous** les Toulousains et de **tous** les Nantais : on ne disposera que d'un échantillon de chaque population dont les moyennes ne sont que des approximations de $\mu_{taille,T}$ et de $\mu_{taille,N}$ (voir le chapitre 2). Avec cette information partielle on ne pourra jamais dire que H_0 est vraie ou fausse, on se débrouille avec un raisonnement probabiliste : on calcule d'abord la "différence" entre les deux échantillons, et ensuite, en supposant que H_0 est vraie, on calcule la probabilité p de trouver une telle différence ou plus grande. Si cette probabilité p est plus petite qu'une probabilité seuil α ($p < \alpha$) on rejette H_0, sinon on ne la rejette pas. Si on la rejette, on encourt un risque α de la rejeter à tort (c'est-à-dire que H_0 est vraie) : on appelle cela une erreur du type I. Si on ne la rejette pas on encourt le risque contraire, c'est-à-dire que H_0 est en réalité fausse. On appelle cela une erreur de type II et le risque de commettre une telle erreur est nommé β. Ces faits sont résumés dans le Tab 3.1.

TABLE 3.1 – Les deux types d'erreurs qu'on peut commettre en testant une hypothèse. La colonne à gauche représente le résultat du test et la première ligne la réalité.

	H_0 est vraie	H_0 est fausse
H_0 est rejeté	erreur type I, α	pas d'erreur
H_0 n'est pas rejeté	pas d'erreur	erreur type II, β

Le risque de ne pas commettre une erreur de type II est $1 - \beta$, on l'appelle la **puissance**. Il y a évidemment un compromis à faire : pour un échantillon donné, plus α est petit plus β est grand (et la puissance petite) et vice versa. En biologie α est d'habitude fixé à 0.05 (5%). β est assez difficile à calculer analytiquement, mais avec un ordinateur on peut toujours le trouver par simulation (par exemple avec **R**, voir 3.11 pour un exemple). Quand on définit son protocole expérimental on devrait s'assurer de se donner les moyens pour pouvoir rejeter H_0, c'est-à-dire d'avoir suffisamment de données pour que $\beta < 0.2$ (équivalent à une puissance $> 80\%$). Mais pour cela on a besoin des

1. Sir Karl Raimund Popper, 1902-1994. Professeur en logique et théorie de la science, à Londres.

résultats d'une pré-expérience (calculer la puissance à posteriori ne sert pas à l'interprétation, voir Nakagawa & Foster (2004)).

3.1 Les choix binaires et les tests associés

On observe un animal qui choisit entre deux actions (les abeilles qui volent à gauche ou à droite dans un tunnel en Y, les blattes qui s'abritent dans l'abri 1 ou 2, ...). Notons par f_1 le nombre d'animaux qui ont fait le choix 1 et par f_2 ceux qui ont fait le choix 2. Par exemple, dans une expérience on a laissé accéder 50 blattes une après l'autre à une arène contenant deux abris et on observe à la fin $f_1 = 33$ et $f_2 = 17$. On peut se demander si le choix est aléatoire ou si les animaux ont une préférence (par exemple une affinité sociale, c'est-à-dire choisir de préférence l'abri avec plus de blattes déjà présent). On a donc l'hypothèse nulle

$$H_0 : \quad f_1 = f_2$$

Plusieurs tests existent pour cette situation. Notons dans la suite $n = f_1 + f_2$ et par $\hat{f}_i$ les fréquences attendues. Pour l'exemple des blattes on a $n = 50$, $\hat{f}_1 = \hat{f}_2 = 25$. Notons aussi la différence entre fréquence absolue ($f_1 = 33$) et fréquence relative ($f_1 = 33/50 = 0.66$), cette différence est cruciale.

Le χ^2-test pour les choix binaires

Un premier test est le χ^2, un test générique pour comparer entre fréquences observées et attendues. On utilise les fréquence absolues pour calculer

$$\chi^2 = \sum_{i=1}^{k} \frac{(f_i - \hat{f}_i)^2}{\hat{f}_i} \tag{3.1}$$

(dans notre cas $k = 2$). Plus χ^2 est grand, plus H_0 devient improbable, mais les valeurs seuils pour rejeter H_0 ne dépendent pas seulement de α mais aussi du nombre de fréquences à comparer (k). χ^2 est distribué selon une Loi du même nom et $df = k - 1$ degrés de liberté, la valeur seuil pour $\alpha = 0.05$ et $k = 2$ est, par exemple, 3.841. Donc, si $\chi^2 > 3.841$ on rejette H_0. Attention, on recommande de n'appliquer le test de χ^2 que si la plus petite fréquence attendue ($\hat{f}_i$) n'est pas au-dessous de 5, sinon le calcul du p associé devient imprécis. Mais si toutes les fréquences attendues sont égales on peut baisser cette limite jusqu'à 1 ($\alpha = 0.05$) ou 2 ($\alpha = 0.01$), voir chap. 22.5 de Zar (1999).

Exemple

```
R> fobs = c(33,17)
R> chisq.test(fobs,p=c(0.5,0.5))            # p contient les proportions attendues
```

Le résultat nous donne $\chi^2 = 5.12$ qui est plus grand que la valeur seuil de 3.841, le logiciel calcule $p = 0.024$ qui est effectivement plus petit que α, on peut donc rejeter H_0. D'habitude on ne connaît pas par cœur la valeur seuil, dans ce cas on regarde uniquement si $p < \alpha$ pour rejeter H_0 ou non.

Rapporter les résultats On a observé les fréquences 33 et 17 et on a rejeté l'hypothèse d'un choix aléatoire pour un seuil de significativité de $\alpha = 0.05$ par un test de χ^2 ($\chi^2 = 5.12, df = 1, p = 0.024 < \alpha$).

Remarque : si vous avez des fréquences < 5 et des fréquences théoriques hétérogènes **R** permet quand-même de tester H_0 en calculant le p de façon numérique (Monte Carlo) sans s'appuyer sur la distribution théorique du χ^2 :

```
R> chisq.test(fobs,p=c(0.5,0.5),simulate.p.value=T) # calcul de p par Monte Carlo
```

ce qui nous donne un p d'environ 0.029, très proche du $p = 0.024$ ci-dessus.

D'habitude on ne dispose pas directement des fréquences f_i mais des données brutes qui, dans le cas des blattes, sont

```
R> choixObs = c(2,2,2,2,2,2,1,1,1,1,2,1,1,1,1,1,1,1,2,1,2,2,1,1,1,1,2,2,
+ 1,1,2,1,1,1,2,1,1,2,2,1,2,1,1,1,1,1,1,1)                                  #
R> table(choixObs)                                    # calculer les fréquences
```

et dans ce cas la commande `table` permet de calculer les fréquences, 33 fois la valeur 1 et 17 fois la valeur 2. On peut même intégrér cette commande directement dans l'appel du test statistique

```
R> chisq.test(table(choixObs))
```

Pour aller plus loin Voir le chapitre 24 dans Zar (1999) et le chapitre 22.5 pour les limites d'application du test de χ^2. En particulier, on n'a discuté que des tests avec des H_0 'bilatérales' (égalité des deux fréquences), dans Zar (1999) vous trouverez comment faire des tests unilatérals (fréquence 1 plus petite que fréquence 2 ou vice versa). Il arrive aussi qu'on répète une expérience plusieurs fois, est-ce qu'on a le droit de cumuler les résultats pour avoir plus de puissance dans le test de χ^2 ? On peut répondre à cette question par un test d'hétérogénéité (voir 22.6 dans Zar 1999).

Le test binomial

Si la probabilité de succès d'un évènement est p la distribution binomiale nous dit que la probabilité d'avoir exactement m succès dans n tirages est

$$\binom{n}{m} p^m (1-p)^{n-m} = \frac{n!}{m!\,(n-m)!} p^m (1-p)^{n-m}.$$

Sur cette base et une approximation de l'intervalle de confiance d'une proportion par la distribution de Fisher (qu'on peut relier à la distribution binomiale) il existe un test binomial pour voir si la fréquence observée est égale à une fréquence attendue, il suffit de préciser ces deux valeurs.

Exemple On reprend l'exemple du choix des blattes, avec 33 à gauche et 17 à droite.

```
R> binom.test(33,50,p=0.5)
R> binom.test(c(33,17),p=0.5)          # le test calcule n à partir des fréquences
```

Le résultat donne un p légèrement différent de celui du test de χ^2 (c'est normal, les deux tests sont basés sur différents fondements théoriques), mais il reste significatif à $\alpha = 0.05$.

Dans **R** il est d'habitude simple de faire les tests unilatérals, par exemple H_0 : $33/50 < 0.5$ pour corroborer l'hypothèse que l'abri 1 est choisi de façon préférentielle.

```
R> help(binom.test)                 # trouver la syntaxe pour indiquer 'unilatéral'
R> binom.test(33,50,p=0.5,alternative='greater')    # avec H0 : 33/50 < 0.5 unilatérale
```

Rapporter les résultats On a observé les fréquences 33 et 17 et on a rejeté l'hypothèse d'un choix aléatoire ($\alpha = 0.05$) par un test binomial bilatéral ($p = 0.033 < \alpha$).

Remarque : **R** propose dans ce contexte aussi un autre test, `prop.test`, qui donne quasiment le même résultat

```
R> prop.test(33,50,p=0.5)
```

`prop.test` a l'avantage qu'on peut aussi l'utiliser quand on veut comparer plusieurs proportions à des proportions attendues, pas seulement une seule comme dans le cas du `binom.test`.

Extension aux courbes d'apprentissage succès-échec

Dans les expériences d'apprentissage à choix binaire il faut entraîner un individu plusieurs fois avant qu'il apprenne à effectuer le bon choix. On regarde donc si après un certain nombre d'expériences la proportion de succès a augmenté de façon significative. Comme ce sont les mêmes individus qui sont conditionnés à chaque essai on parle de mesures binaires répétées. Le test Q de Cochran teste H_0 que les fréquences de succès ne changent pas au cours de l'entraînement, c'est donc un bon test pour voir si l'entraînement a marché. Attention : les choix binaires doivent être codés en 0 et 1 pour que la fonction marche.

Exemple Dans le jeu de données `abeillesCond` (extrait de Giurfa 2004) 15 abeilles ont été entraînées pendant 15 séances : les résultats sont organisés dans un tableau ou chaque ligne contient les essais pour un même individu, codé en `1` pour succès et `0` pour échec.

```
R> load("giurfa004abs.rda")                              # charger abeillesCond
R> abeillesCond[1,]                        # les 15 essais de la première abeille
R> plot(apply(abeillesCond,2,sum)/15,xlab="Essai",    # graphique d'apprentissage
+ ylab="Proportion appris",type='b',ylim=c(0.5,1))                            #
R> source("fonctionsCompStat.R");           # charger la fonction Cochran.Q.test
```

```
R> Cochran.Q.test(abeillesCond)          # tester si la fréquence de succès change
```
On obtient $p = 0.0015$ et on peut donc rejeter H_0

Rapporter les résultats Pour voir si nos 15 abeilles ont bien appris le choix binaire durant les 15 essais nous avons effectué un test Q de Cochran ($\alpha = 0.05$). Les résultats ($Q = 34.96$, $df = 14$, $p = 0.0015$) indiquent que c'était bien le cas.

3.2 Les tests sur les fréquences d'évènements

Souvent on n'a pas seulement deux choix (binaire) mais plusieurs ($k > 2$ dans l'équation 3.1) et on compte la fréquence de chaque choix (par exemple : appartenance d'un individu à une caste, fréquences des phénotypes dans une population). On compare ensuite ces fréquences à des fréquences théoriques qu'on calcule sur la base de H_0. Sous H_0 l'équation (3.1) est distribuée selon une distribution de χ^2 avec $df = n - 1$ degrés de liberté. Dans l'exemple des castes, on peut tester si chaque caste a la même taille (ce qui indiquerait que l'appartenance à une caste est peut-être aléatoire).

Exemple Reprenons l'exemple de trois castes (nourrices, bâtisseuses, fourrageuses) du chapitre 1.1 :

```
R> castes=c(35,10,95)
R> chisq.test(castes,p=c(1/3,1/3,1/3))            # H0: tailles des castes égaux
R> chisq.test(castes)               # c'est en fait l'hypothèse par défaut dans R
R> F2 = c(4,18,12,55);                    # fréquence de quatres phénotypes en F2
R> freqAtt = c(1,3,3,9)/16                   # les fréquence relatives attendues
R> chisq.test(F2, p=freqAtt)              # la différence n'est pas significative
```

Rapporter les résultats On a observé les phénotypes en fréquence (4,18,12,55), ce qui n'est pas significativement différent ($\alpha = 0.05$) de la fréquence attendue (1,3,3,9), avec $\chi^2 = 2.35$, $df = 3$, $p = 0.50$.

Prenons un dernier exemple tiré de Roces & Kleineidam (2000). Ces chercheurs voulaient savoir si les fourmis *Atta sexdens* ont une zone d'humidité préférée pour stocker le champignon qu'ils cultivent pour nourrir le couvain et la reine. A cette fin ils ont préparé quatre boîtes interconnectées, chacune avec une humidité différente : 33%, 75%, 84% et 98%, y ont laissé les fourmis déplacer librement le champignon et à la fin ils ont regardé où se trouvait le champignon. Ils ont trouvé les fréquences associées de 6, 2, 12 et 60.

```
R> champ = c(6,2,12,60)
R> chisq.test(champ,p=c(1/4,1/4,1/4,1/4))
```
et ce résultat est hautement significative, ce qui veut dire deux choses : a) les fourmis savent détecter l'humidité et b) elles préfèrent une forte humidité pour stocker le champignon.

Pour aller plus loin Si on détecte une différence significative on peut se demander laquelle des fréquences est la plus différente de sa valeur théorique. On peut répondre à cette question par une sous-division du χ^2 (voir 22.3 dans Zar 1999). Il arrive aussi que l'on répète une expérience plusieurs fois, est-ce qu'on a le droit de cumuler les résultats pour avoir plus de puissance dans le test de χ^2 ? On peut répondre à cette question par un test d'hétérogénéité (voir 22.6 dans Zar 1999 et le paragraphe suivant).

χ^2 d'hétérogénéité

Revenons au choix des blattes, mais cette fois-ci elles on le choix entre un abri marqué (on y a déjà laissé séjourner des blattes) et un abri tout neuf. Après chaque essai on enlève la blatte de l'àrène pour éviter un effet du facteur socialité. Dans une première expérience on laisse choisir 20 blattes une par une, et on compte combien ont choisi l'abri marqué. Nous obtenons 14 dans l'abri marqué, 6 dans l'autre. Ce résultat n'est pas significatif. On répète l'expérience deux autres fois, (13,7) et (12,7), chaque fois ce n'est pas significatif. Est-ce qu'on a le droit de cumuler ces trois échantillons ? Pour cela on regarde si la différence entre la somme des trois χ^2 moins le χ^2 de la somme des trois expériences est significative par rapport à une distribution du χ^2 à 2=3-1 degrés de liberté.

Exemple

```
R> exp1=c(14,6);chi1=chisq.test(exp1)$statistic; chi1  # < 3.841, pas signficatif
R> exp2=c(13,7);chi2=chisq.test(exp2)$statistic; chi2           # pas signficatif
R> exp3=c(12,7);chi3=chisq.test(exp3)$statistic; chi3           # pas signficatif
R> chiCum = chisqu.test(exp1+exp2+exp3)$statistic; chiCum
R> chiDiff = chi1+chi2+chi3-chiCum; chiDiff
R> qchisq(0.95,2);    # calcul de la valeur seuil à df = 2, plus grand que chiDiff
R> pchisq(chiDiff,2)                        # calcul de p associé à chiDiff, < 0.05
```

Le χ^2 d'hétérogénéité `chiDiff` est plus petit que la valeur seuil, on a donc le droit de cumuler les expériences

```
R> chisq.test(exp1+exp2+exp3)$p.value    # plus petit que 0.05, donc significatif
```

On peut donc conclure que les blattes préfèrent des abris marqués par leur espèce.

Analyse des tableaux de contingences

Il arrive qu'on ne veuille pas comparer des fréquences observées à des fréquences théoriques mais comparer les fréquences observées entre deux conditions expérimentales. Par exemple, reprenons l'exemple du chapitre 1.1, les fréquences des trois castes dans un jeune nid et un vieux nid. Le tableau de fréquences est

	nourrices	bâtisseuses	fourrageuses
jeune nid	35	10	95
vieux nid	100	95	230

On appelle cela un tableau de contingences et H_0 est dans ce cas que les fréquences des castes sont indépendantes de l'état du nid (jeune ou vieux).

Exemple

```
R> tabCon = rbind(c(35,10,95),c(100,95,230));tabCon     # construction du tableau
R> chisq.test(tabCon)           # chisq.test reconnais un tableau de contingences
```

Le degrés de liberté rapporté par **R** est $(nombreColonnes - 1) * (nombreLignes - 1)$.

Rapporter les résultats On a observé (35,10,95) nourrices, bâtisseuses et fourrageuses dans le jeune nid et (100,95,230) dans le vieux nid. L'effet 'âge du nid' est significatif à $\alpha = 0.05$ ($\chi^2 = 16.7, df = 2, p < 0.001$).

Remarque : dans cet exemple on aurait aussi pu appliquer le `prop.test` mentionné ci-dessus,

```
R> prop.test(tabCon[1,],tabCon[1,]+tabCon[2,])
```

Pour aller plus loin Tout le chapitre 23 de Zar (1999) est consacré aux tableaux de contingences. Il y discute aussi le cumul de plusieurs expériences (χ^2 d'hétérogénéité) et la sous-division d'une analyse pour détecter les fréquences qui sont les plus différentes entre les deux conditions.

Le test de McNemar

Dans les tests comportementaux la variabilité est souvent très élevée d'un individu à l'autre : pour tester si deux différents traitements ont un effet signficativement différent l'un de l'autre on a donc besoins de grands échantillons. Si ce test donne un résultat nominale (oui/non, succès/échec, ...) il existe un protocole alternative où on teste la réaction de chaque individu pour les deux traitements (un cas particulier de mesures appariées), le test de McNemar.

Exemple Prenons par exemple deux traitements anxiolytiques, le diazepam (Valium) et le lorazepam (Temesta), et on veut tester s'ils ont le même effet sur le choix des souris de 3 mois dans le test d'anxiété clair/obscur (on fait les deux essais pour chaque individu décalé d'une journée et on tire au hasard les individus qui passent d'abord avec le diazepam (randomisation de l'ordre de passage). Sur 50 souris, $f_{11} = 15$ on choisi la chambre 'clair' dans les deux cas, $f_{22} = 10$ ont choisi la chambre 'obscure' dans les deux cas, $f_{12} = 5$ on choisi 'clair' avec le diazepam et 'obscur' avec le lorazepam, et $f_{21} = 25$ ont fait le contraire. Le test de McNemar teste maintenant $H_0 : \psi = f_{12}/f_{21} = 1$.

```
R> resultat = cbind(c(15,5),c(20,10));resultat            # organiser les données
R> mcnemar.test(resultat)                                    # effectuer le test
```

Rapporter les résultats Un test de McNemar montre que l'effect anxiolytique est significativement ($\alpha = 0.05$) différent entre le diazepam et le lorazepam ($\chi^2 = 7.84$,dl $= 1$, $p = 0.005$).

Pour aller plus loin Attention, le test de McNemar n'a rien à voir avec les tableaux de contingences malgré une organisation similaire des données. Voir le chapitre 9.7 dans Zar (1999) pour plus d'information.

3.3 Le test de normalité, paramétrique ↔ non-paramétrique

Jusqu'à présent on n'a analysé que des données nominales ou ordinales. Sur ces données on applique directement le test associé à son H_0 (χ^2-test, test binomial).

Quand on analyse un échantillon de données continues il y a une vérification à faire avant le test proprement dit : voir si les données sont distribuées selon une distribution normale. Ceci est nécessaire parce que le calcul de p (la probabilité de commettre une erreur de type I) dans des tests '**paramétriques**' tels que le test de Student ou les ANOVAs s'appuie sur cette propriété. Si un échantillon n'est pas distribué selon une Loi normale il faut utiliser des tests du type '**non-paramétrique**' tel que le test de Mann-Whitney ou Kruskal-Wallis. Faire cette distinction (plutôt que d'utiliser systématiquement des tests non-paramétriques) et due à la puissance du test (le risque de commettre une erreur de type II) : si l'échantillon **est** distribué selon une Normale un test paramétrique est **plus puissant** que le test non-paramétrique correspondant. En d'autres termes, si H_0 est fausse, avec un test paramétrique on a besoin de moins de données pour pouvoir la rejeter en comparaison avec un test non-paramétrique.

On a donc besoin d'un test qui nous dit si les données d'un échantillon sont distribuées de façon Normale ou non. L'hypothèse nulle est

$$H_0 : \text{Les données sont distribuées selon une Loi normale}$$

et le test calcule p : si $p < \alpha$ on rejette H_0 et on utilise un test non-paramétrique, sinon on ne la rejette pas et on peut utiliser un test paramétrique.

Le premier test pour faire cela et le test de 'Kolmogorov-Smirnov' (K-S), un test générique pour comparer entre la distribution empirique d'un échantillon et une distribution théorique (ou aussi entre les distributions empiriques de deux échantillons, mais cela ne nous intéresse pas ici). Dans ce test on spécifie qu'on veut comparer à une Normale et on fournit en même temps la moyenne et l'écart-type de cette Normale (qu'on estime à partir de l'échantillon).

Le deuxième test est spécifique pour tester la normalité : le test de Shapiro-Wilks. Il ne regarde pas la distribution empirique entière mais seulement si elle est symétrique (la "skewness") et si elle est plus pointue ou moins pointue qu'une Normale (la 'kurtosis'). Ce test est moins strict que le K-S, mais il suffit largement pour pouvoir décider entre paramétrique et non-paramétrique. Il est donc recommandé.

Exemple Regardons le temps que les moutons consacrent au brout. Le jeu de données 'MoutBroutSexe' contient le pourcentage de temps passé à brouter chez les mâles et femelles moutons "Mérino d'Arles", et on veut tester si ces deux graphiques de données sont distribuées de façon normale

```
R> load("moutons.rda")                                      # charger les données
R> moutBroutSexe
R> boxplot(moutBroutSexe)                                 # visualiser les données
R> f = moutBroutSexe$fem                  # f contient le % de brout des femelles
R> m = moutBroutSexe$mal
```

On appelle maintenant le test K-S en fournissant l'échantillon, la distribution et les estimations de moyenne et écart-type :

```
R> ks.test(f,"pnorm",mean=mean(f),sd=sd(f))    # le test de K-S pour les femelles
R> ks.test(m,"pnorm",mean=mean(m),sd=sd(m))                    # pour les mâles
R> shapiro.test(f)                    # le test de Shapiro-Wilks pour les femelles
```

```
R> shapiro.test(m)                                          # pour les mâles
```

Rapporter les résultats L'échantillon des moutons femelles est distribué selon une Loi normale (Kolmogorov-Smirnov, $D = 0.167$, $p = 0.638$).
L'échantillon des moutons mâles est distribué selon une Loi normale (Shapiro-Wilks test, $W = 0.937$, $p = 0.209$).

Pour aller plus loin Pour une illustration de la 'kurtosis' voir le chapitre 6, p. 67, de Zar (1999). Ce chapitre donne une introduction générale à la distribution Normale.

3.4 Comparaison entre deux échantillons

On compare deux échantillons entre eux et on veut savoir si ils sont significativement différentes (c'est-à-dire si les moyennes des deux populations qu'ils représentent sont différentes) :

$$H_0 : \mu_{\text{échantillon 1}} = \mu_{\text{échantillons 2}}$$

Si les deux échantillons suivent une loi Normale on passe à un test de Student (paramétrique), sinon on fait un test de Mann-Whitney.

Le cas paramétrique

Prenons nos moutons mâles et femelles dont on sait déjà qu'ils suivent une loi Normale. On applique maintenant le test de Student sur ces deux échantillons

```
R> load("moutons.rda")                                      # charger les données
R> f = moutBroutSexe$fem                  # f contient le % de brout des femelles
R> m = moutBroutSexe$mal
R> t.test(f,m)                                               # le test de Student
```

Rapporter les résultats Un test de Student (après vérification de la Normalité) n'a donné aucune différence significative ($\alpha = 0.05$) entre le temps de brout des moutons (Mérino d'Arles) femelles et mâles ($|t| = 1.56$, $df = 37.3$, $p = 0.128$).

Remarque : vous avez peut-être vu qu'on a un degré de liberté non-entier, $df = 37.3$. Comment est-ce possible ? En fait, le test de Student qui est implémenté dans **R** est une généralisation de celui qu'on trouve dans la majorité des livres. Le test classique exige, au-delà de la normalité des échantillons, que ces échantillons aient la même variance (homoscédasticité). Mais en cas d'hétéroscédasticité on peut corriger le biais que cela entraîne en calculant un degré de liberté qui la prend en compte. Voir Zar (1999) ch. 8.1 pour plus de détails sur ce « Behrens-Fisher problem ».

Le cas paramétrique apparié

Il se peut qu'on teste le même individu sous deux conditions (avec ou sans traitement, avant et après traitement, etc.). Dans ce cas les données des deux échantillons ne sont pas indépendantes, on ne peut donc pas faire le test de Student ci-dessus. Mais on peut regarder si la différence entre les deux mesures sur le même individu est significativement différente de 0. Le test de Student peut s'adapter à ce cas,

$$H_0 : \mu_{\text{différences}} = 0$$

Exemple Reprenons nos données de temps de réactions. On a fait des mesures dans deux conditions, sans et avec distraction (dans ce cas, la distraction consiste à discuter avec son collègue pendant l'expérience) :

```
R> tempsReaction = read.table("tempsReaction2004.txt",h=T)     # lire les données
R> trS = tempsReaction$simple.sans.dist           # trS: données sans distraction
R> trD = tempsReaction$simple.avec.dist
R> trDiff = trS-trD                                      # calcul de la différence
```

```
R> shapiro.test(trDiff)                                    # test de Normalité
R> t.test(trDiff,mu=0)            # appliquer le test de Student à un échantillon
R> t.test(trS,trD,paired=T)        # c'est la même chose avec l'option "paired=T"
```

Rapporter les résultats On a testé si le temps de réaction est différent ($\alpha = 0.05$) selon qu'on est distrait ou pas. Après avoir vérifié la Normalité (Shapiro-Wilks test, $W = 0.971$, $p = 0.709$) on a trouvé une différence significative (test de Student à un échantillon, $|t| = 6.0441$, $df = 23$, $p < 0.001$).

Le cas non-paramétrique

Si l'un des deux échantillons n'est pas normal le p calculé par le test de Student n'est pas correct (on dit qu'il est biaisé). Dans ce cas on applique un test non-paramétrique qui ne fait pas l'hypothèse de cette Normalité mais qui est moins puissant (d'un côté pratique cela veut dire que les échantillons doivent être plus grands pour ne pas commettre une erreur du type II).

Exemple On regarde les trajectoires des fourmis *Messor sancta* à deux températures, 16°C et 30°C, en particulier les longueurs des déplacements rectilignes avant de tourner. Ces bouts de trajectoire rectiligne s'appellent les 'libres parcours' (pour la visualisation de ces données voir aussi le chapitre 1.3).

```
R> load("lpMessorSancta.rda")                                  # lire les donées
R> libParc.16.30                                             # afficher les données
R> boxplot(libParc.16.30)                                   # visualiser les données
R> shapiro.test(libParc.16.30$lp16cm)                          # test de Normalité
R> hist(libParc.16.30$lp16cm)          # en fait, la distribution est asymétrique
R> shapiro.test(libParc.16.30$lp30cm)
R> wilcox.test(libParc.16.30$lp16cm,libParc.16.30$lp30cm); # test de Mann-Whitney
```

Rapporter les résultats Les libres parcours de *Messor sancta* sont significativement différents ($\alpha = 0.05$) entre 16°C et 30°C (Mann-Whitney test, $W = 457.5$, $n_1 = 50, n_2 = 50$, $p < 0.001$).

Remarque On peut aussi analyser des données appariées avec l'option "paired=T". Attention, dans ce cas on teste si la **médiane** des différences est significativement différente de 0 (et pas la moyenne). Si vous devez comparer la moyenne à une constante il faut avoir un échantillon suffisamment grand et appliquer le test de Student (Boos & Hughes-Oliver, 2000).

Pour aller plus loin Les chapitres 7 et 8 dans Zar (1999) sont consacrés à tous les tests qu'on peut effectuer sur un ou deux échantillons.

3.5 Comparaison entre plusieurs échantillons

Dans le cas où on fait des mesures (on parle aussi souvent d'une variable dépendante) dans plus de deux conditions expérimentales (la variable indépendante) on a une nouvelle H_0 à tester :

$$H_0 : \mu_{condition\,1} = \mu_{condition\,2} = \ldots = \mu_{condition\,k}$$

Le test de Student ne suffit plus dans ce cas parce qu'il ne compare qu'entre deux échantillons. Pour savoir si la variable dépendante est influencée par les différentes conditions/modalités de la variable indépendante, il faut faire appel à des tests du type ANOVA (paramétrique) ou Kruskal-Wallis (non-paramétrique). Pour pouvoir utiliser un test paramétrique on a non seulement besoin que tous les échantillons aient une distribution Normale, mais aussi que leurs variances soient égales (ce qu'on appelle l'**homoscédasticité**, par opposition à l'**hétéroscédasticité** = inégalité des variances). L'hypothèse H_0 à tester est :

$$H_0 : \sigma^2_{condition\,1} = \sigma^2_{condition\,2} = \ldots = \sigma^2_{condition\,k}.$$

Un test qui fait cela est le test de Bartlett. Tandis qu'une ANOVA est assez robuste vis-à-vis d'une légère non-normalité des échantillons, une hétéroscédasticité peut être fatale dans le sens que le p calculé ne veut rien dire.

Exemple Prenons un exemple où on a caractérisé le comportement des moutons pendant le brout par deux mesures : le nombre de bouchées par minute (mesure de la consommation) et la durée où la tête est dressé (mesure de la vigilance). Ces mesures ont été faites dans quatre conditions, des femelles et des mâles dans des groupes unisexes, et les femelles et mâles dans des groupes mixtes.

```
R> load("plusieursEchantillons.rda")                          # charger les données
R> boxplot(tpsDroitRegard ~ type, data = moutonsBrout)            # visualisation
R> bartlett.test(tpsDroitRegard ~ type, data=moutonsBrout)     # test de Bartlett
R> parType = split(moutonsBrout$tpsDroitRegard, moutonsBrout$type)
R> lapply(parType,shapiro.test)                               # test de la Normalité
```

On voit qu'on a homoscédasticité et que tous les échantillons sont normaux, on peut donc utiliser un test paramétrique.

Dans un autre cas on a mesuré la croissance des conifères en fonction de l'orientation du versant (S, O, E) et on se demande si cette orientation influence la croissance.

```
R> boxplot(croiss ~ versant, data=mesConifs)                       # visualisation
R> bartlett.test(split(mesConifs$croiss, mesConifs$versant))   # test de Bartlett
R> lapply(split(croissances, typeVers),shapiro.test)          # test de Normalité
```

Cette fois-ci on remarque une nette hétéroscédasticité, un test non-paramétrique serait donc plus prudent à utiliser.

Remarque : le format des données ci-dessus,

```
R> mesConifs
```

est typique pour les analyses statistiques dans tous les logiciels statistiques : une première colonne qui donne toutes les mesures et une seconde colonne qui donne les conditions expérimentales. On peut ajouter d'autres colonnes pour donner plus d'informations sur les conditions, par exemple la colonne 'taille' dans

```
R> moutonsBrout
```

indique la taille du groupe testé.

Veuillez aussi noter l'écriture du **modèle statistique** : `croiss ~ versant`. Le signe `~` veut dire "en fonction de", on fait donc un graphique du type `boxplot` de la mesure `croiss` en fonction du facteur `versant`. Cette écriture est très répandue dans les logiciels statistiques et des textes expliquant l'utilisations de ces tests.

Pour aller plus loin Les tests sur plusieurs échantillons sont largement discutés dans les chapitres 10 et 11 de Zar (1999).

Dans le cas de données hétéroscédastiques on peut parfois faire une transformation des données (prendre la racine ou le logarithme de toutes les mesures) pour les rendre homoscédastiques (voir ch. 13 dans Zar 1999).

Le test de Bartlett a une mauvaise performance si les échantillons ne suivent pas une loi Normale. Zar (1999) recommande donc la non-utilisation de ce test dans ce cas et évoque la robustesse de l'ANOVA pour quand-même en faire une. Mais voir l'exemple ci-dessous qui suggère qu'un test non-paramétrique peut être plus adapté.

La meilleure explication concernant l'écriture des modèles statistiques dans **R** se trouve dans le chapitre 7 de Crawley (2005).

Le cas paramétrique

Dans le cas d'échantillons normaux et d'homoscédasticité on fait une ANOVA ("analysis of variance") qui teste

$$H_0 : \mu_{condition\,1} = \mu_{condition\,2} = \ldots = \mu_{condition\,k}$$

Si le p qui en résulte est plus petit que α on rejette H_0, il y a donc un effet global, mais on ne sait pas encore où est la différence.

Exemple Prenons les temps passés par les moutons la tête dressée :

```
R> boxplot(tpsDroitRegard ~ type, data = moutonsBrout)
R> resultat = aov(tpsDroitRegard ~ type, data=moutonsBrout)
R> resultat               # afficher les résultats bruts, difficile à interpréter
R> summary(resultat)                          # 'summary' calcule le p de ce test.
```

Rapporter les résultats Dans une ANOVA on calcule une statistique qui s'appelle le F de Fisher [2]. La valeur seuil de ce F (pour décider si H_0 peut être rejeté ou non) dépend de deux degrés de liberté : de $k-1$ (nombre d'échantillons moins un) et de $n-k$ (nombre total de mesures moins le nombre d'échantillons). Il faut rapporter le F calculé, ces deux degrés de liberté et le p associé. Pour les moutons : le temps passé la tête dressée dépend significativement ($\alpha = 0.05$) du type de groupe d'animaux ($F = 4.98$, $df = (76, 3)$, $p = 0.0033$).

Un autre chiffre qui est util de rapporter est le coefficient de détermination R^2 qui chiffre la fraction de la variance globale qui est expliquée par le modèle stastistique. Dans notre cas ce modèle statistique est de décrire les données associées aux quatre modalités "f", "m", "f.mixte" et "m.mixte" par quatre moyennes. On obtient le coefficient de détermination par la commande

```
R> summary.lm(resultat)
```

Il a la valeur $R^2 = 0.1644$, c'est-à-dire 16.4% de la variance totale est expliquée par l'ANOVA (voir aussi le chapitre 3.8). Ceci peut paraître assez faible, mais Møller & Jennions (2002) ont fait une méta-analyse sur les articles publiés en écologie et évolution et ils ont trouvé qu'en moyenne seulement 2.51–5.42% de la variance sont expliqués par les facteurs testés dans ces études.

Le test post-hoc dans le cas paramétrique

Une ANOVA ne nous dit pas encore quel échantillon est différent duquel. Ceci peut être détecté après une ANOVA par un test 'post-hoc'. Il y a toute une gamme de test post-hoc dont les plus connus sont : Tukey honestly significant test, Newman-Keuls test, Dunnett test (pour comparer la moyenne d'un contrôle à toutes les autres), Scheffé's multiple contrasts. La Fig 3.1 résume les résultats du Tukey HSD, en identifiant chaque population statistique par une lettre 'a' ou 'b'. Le fait que le type `f` (femelles seules) a les deux lettres veut dire qu'on ne peut pas décider s'il appartient à la population statistique 'a' (`f.mixte`) ou 'b' `m` et `m.mixte`, c'est-à-dire que cet échantillon n'est ni significativement différent de l'échantillon `f.mixte`, ni des échantillons `m` et `m.mixte`.

Exemple

```
R> resultat = aov(tpsDroitRegard ~ type, data=moutonsBrout)
R> TukeyHSD(resultat)            # application du Tukey HSD au résultat de l'ANOVA
```

Rapporter les résultats D'habitude on ne rapporte pas les détails d'une analyse post-hoc, on identifie seulement les populations statistiques sur un graphique résumant les données (voir Fig 3.1).

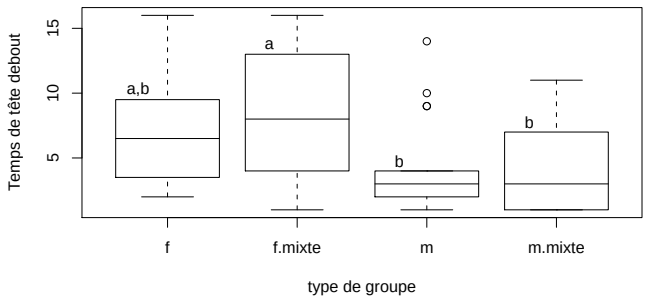

FIGURE 3.1 – Le temps d'avoir la tête dressée (indicateur de vigilance) de femelles et de mâles en groupe unisexe (f, m) ou en groupe mixte (f.mixte, m.mixte). 'a' et 'b' indiquent les populations statistiques détectées par un Kruskal-Wallis test ($\alpha = 0.05$). Remarque : les lettres ont été ajouté dans **R** avec les commandes `locator(...)` et `text(...)`.

2. Sir Ronald Aylmer Fisher (1890-1962). Statisticien anglais qui a développé les bases de l'ANOVA (entre outre).

Les mesures appariées/répétées

Il y a une généralisation du test de Student pour mesures appariées au cas de plusieurs échantillons : l'ANOVA à mesures répétées. Comme dans l'ANOVA classique il faut d'abord visualiser les données et vérifier ensuite la normalité et l'homoscédasticité. Les données sont organisées commme dans l'ANOVA : une colonne de mesures, une colonne du facteur et, de plus, une colonne indiquant quelles mesures sont appariées ou du même sujet. Toutes les mesures sur un même sujet sont aussi souvent appelées un **block**, concept qui comprend aussi toute série de mesures qui ont quelques chose de particulier en commun (du même sujet dans le cas-ci).

Il y a une grande différence entre l'ANOVA classique et l'ANOVA à mesures répétées : il ne faut pas tester l'homoscédasticité mais ce qu'on appelle la « sphéricité » (Zar, 1999, p. 298). Cette sphéricité veut dire que l'ordre entre les sujets (si on les mets en rang par rapport à la variable mesurée) ne varie pas trop d'une séance à l'autre. Cette sphéricité peut être testé par le test de Mauchly.

Exemple On a soumis un échantillon de 7 souris à quatre séances d'exploration d'une arène et à chaque fois on mesure le temps d'exploration du même objet. On se demande si ces temps d'exploration changent au cours des séances.

```
R> load("aovMesRepete.rda")                                    # charger les données
R> sourisApp                # 4 colonnes: (traitement NaCl, séance, temps, sujet)
R> boxplot(sourisApp$explore ~ sourisApp$seance)                  # visualisation
R> lapply(split(sourisApp$explore,sourisApp$seance),shapiro.test)     # normalité
```

Testons maintenant la sphéricité. Pour cela nous devons réorganiser les données en une matrice ou chaque ligne contient les temps d'exploration pour une souris et ensuite appliquer le Mauchly test :

```
R> sourisMat = matrix(sourisApp$explore,nrow=7,byrow=F)          # réorganisation
R> mauchly.test(lm(sourisMat ~ 1))                                          # test
```

On voit que les données ne sont pas tout à fait sphérique ($p = 0.043$) et que la normalité n'est pas parfaite non plus, mais faisons pour le moment appel à la robustesse de l'ANOVA et continuons notre analyse (on verra une alternative dans le chapitre 3.3).

```
R> summary(aov(explore ~ seance + Error(sujet),data=sourisApp))
```

Cette dernière ligne fait l'ANOVA à mesures répétées avec le modèle statistique modifié pour ce cas, temps d'exploration `explore` en fonction de la séance, mais en prenant en compte le facteur `sujet` (chaque sujet a été mesuré plusieurs fois). On pourrais ajouter un test de Dunnett (qui n'existe pas encore dans **R**) pour déterminer à partir de quelle séance le temps est signficativement différent du temps initial.

Cette technique s'applique aussi pour mettre en évidence que le trafic des fourmis sur un pont reliant le nid à une nouvelle source de nourriture augmente au cours du temps :

```
R> dat=recrLinepHumile      # recrLinepHumile se trouvait dans "aovMesRepete.rda"
R> boxplot(dat$nbFourmis ~ dat$temps)                             # visualisation
R> fourmisMat = matrix(dat$nbFourmis,ncol=30,byrow=T)            # réorganisation
R> mauchly.test(lm(fourmisMat ~ 1))                           # parfaite sphéricité
R> summary(aov(dat$nbFourmis ~ dat$temps + Error(dat$experience)))
```

Rapporter les résultats Une ANOVA à mesures répétées démontre que le traffic sur le pont change significativement ($\alpha = 0.05$) au cours du temps ($F = 6.579$, $df = (29, 116)$, $p < 0.001$).

Pour aller plus loin Voir le chapitre 14.4 dans Zar (1999) pour plus de détails. Veuillez remarquer qu'on appelle souvent l'ensemble des mesures effectuées sur le même sujet (ou faisant partie de la même expérience dans le cas du recrutement de fourmis) un "block", c'est donc un dispositif expérimental 'bloqué' sur les sujets.

Les post-hoc dans le cas des mesures appariées

L'exemple précédent sur le temps d'exploration des souris en fonction des séances nous a permis de voir que ce temps diminue significativement d'une façon globale. Mais entre lesquelles des séances y a-t-il une différence significative ? Le test post-hoc de Tukey de la page 23 ne nous aidera pas parce qu'il n'est pas adapté au cas des mesures répétées. Mais **R** propose une méthode qui compare

les échantillons paire par paire avec un test de Student et qui corrige ensuite les valeurs des p tout en prenant en compte que les données sont appariées et qu'on fait plusieurs comparaisons deux par deux (voir aussi le paragraphe 3.10). La méthode qu'il nous faut est due à Benjamini et Yekutieli, indiqué par l'abbréviation `BY` dans le code ci-dessous.

```
R> load("aovMesRepete.rda")                                    # charger les données
R> boxplot(sourisApp$explore   sourisApp$seance)                    # visualisation
R> summary(aov(explore   seance + Error(sujet), data=sourisApp))          # ANOVA
R> pairwise.t.test(sourisApp$explore,sourisApp$seance,p.adj="BY")      # post-hoc
```

Rapporter les résultats Une ANOVA à mesures répétées nous indique que le temps d'exploration des souris diminue au cours des séances ($F = 7.85$, $dl = (3, 18)$, $p = 0.0015$). Un post-hoc d'après la méthode de Benjamini & Yekutieli montre que la différence est significative entre les séances 1 et 3 ($p = 0.032$) et 1 et 4 ($p = 0.012$).

Le cas non-paramétrique

On est toujours dans le cas de plusieurs échantillons, chacun représentant les mesures dans une condition spécifique, et on pose l'H_0 du chapitre 3.5. Si un des échantillons n'est pas distribué de façon normale, ou s'il y a hétéroscédasticité, la théorie statistique dit qu'on n'a pas le droit de faire une ANOVA. Cependant, des études de simulation ont montré que l'ANOVA est très 'robuste' vis à vis d'une légère non-normalité des échantillons. Ce terme **robuste** veut dire dans ce contexte que le p calculé dans le test est légèrement incorrect, ce qui a comme résultat que le risque de commettre une erreur de type I n'est pas exactement α mais un peu plus grand ou plus petit. Dans le cas d'une non-normalité ce 'un peu' est souvent moins de 1 %, dans le cas d'un seuil $\alpha = 0.05$ le vrai risque est donc entre 0.06 et 0.04, une incertitude qui est négligeable si on a des p largement inférieure à 5 %. Par contre, des échantillons hétéroscédastiques peuvent facilement augmenter cette différence entre α et le "vrai" risque de commettre une erreur de type I à des niveaux beaucoup plus élevés. Dans ces cas des p proches du α choisi deviennent difficiles à interpréter et un test non-paramétrique devient recommendable. Le test de Kruskal-Wallis est un tel test non-paramétrique pour tester

$$H_0 : \mu_{echantillon1} = \mu_{echantillon2} = \ldots = \mu_{echantillon\,k}$$

Exemple Reprenons l'exemple de la croissance des conifères en fonction de l'orientation du versant du début du chapitre :

```
R> load("plusieursEchantillons.rda")                          # charger les données
R> bartlett.test(split(mesConifs$croiss, mesConifs$versant))   # homoscédasticité
```

Ce test nous rend $p = 0.002$, il y a donc une forte hétéroscédasticité. Un test de Kruskal-Wallis nous donne $p = 0.044$, ce qui est signficatif à $\alpha = 0.05$, on peut rejeter H_0.

```
R> kruskal.test(croiss ~ versant, data=mesConifs);
```

Rapporter les résultats On a testé si l'orientation du versant influence la croissance des conifères. Comme les échantillons sont hétéroscédastiques on a fait un test de Kruskal-Wallis. Le résultat est signficatif à $\alpha = 0.05$ ($\chi^2 = 6.24$, $df = 2$, $p = 0.044$).

Quel serait le résultat d'une ANOVA ? Essayons,

```
R> summary(aov(croiss ~ versant, data=mesConifs));
```

nous donne $p = 0.279$, pas du tout significatif ! C'est un cas assez inhabituel, mais l'énigme se résoud en regardant les données :

```
R> boxplot(croiss ~ versant, data=mesConifs)                          # voir Fig 3.2
```

Vous voyez dans l'échantillon 'S' une mesure qui est beaucoup plus petite que toutes les autres mesures (Fig 3.2), une vraie 'valeur aberrante'. Dans l'ANOVA cette valeur aberrante influence la moyenne totale à tel point que l'ANOVA n'est plus significative. Cependant, le test non-paramétrique démontre sans ambiguité qu'il y a un effet de l'orientation du versant sur la croissance des conifères. La petite morale de cet exemple est qu'il faut toujours visualiser les données et vérifier si la tendance dans ce graphique est "compatible" avec le résultat statistique.

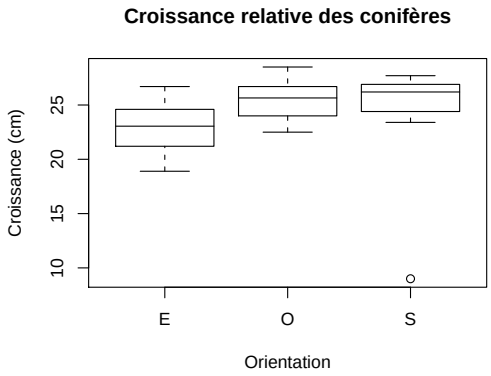

FIGURE 3.2 – Croissance relative des conifères en fonction de l'orientation (Est, Ouest, Sud). Notez la valeur extrême dans l'échantillon S qui fausse complètement le résultat d'une ANOVA.

Un test post-hoc non-paramétrique

Comme dans l'ANOVA il faut faire un test post-hoc pour savoir lequel des échantillons est différent de l'autre. Dans le cas non-paramétrique un test disponible est celui de Nemenyi (Zar, 1999).

Exemple Le test de Nemenyi n'existe pas encore dans **R**, j'ai donc fait ma propre petite fonction (un peu maladroitement, je ne suis pas un programmeur). On reprend les données de la croissance des conifères :

```
R> load("plusieursEchantillons.rda")                          # charger les données
R> source("fonctionsCompStat.R")                              # charger la fonction
R> mesConifsList = split(mesConifs$croiss,mesConifs$versant)          # conversion
R> Nemenyi.test(mesConifsList,alpha=0.1)                          # appel du test
```

Pour interpréter il faut regarder les valeurs non-nuls de la matrice `Q` et les comparer à la valeur `Qseuil`. Par exemple, `Q[3,1]=2.32 > 2.13`, la médiane du 3ème échantillon est donc significativement différente de celle du 1er échantillon (à la valeur $\alpha = 0.10$). Par contre, `Q[2,1]=2.09 < 2.13`, il n'y a donc pas de différence significative entre le 2ème et le 1er échantillon. Veuillez noter que j'ai défini $\alpha = 0.1$: pour $\alpha = 0.05$ il n'y a aucune différence significative malgré l'effet significatif détecté par le test de Kruskal-Wallis, cela peux arriver parce que les deux tests sont indépendants l'un de l'autre. Mais le test de Nemenyi est aussi connu pour sa faible puissance (García & Hererra, 2008), une alternative serait donc de faire des tests deux par deux par Mann-Whitney (3.4) avec une correction de Bonferroni ou Dunn-Sivak (3.10).

Le cas non-paramétrique des mesures répétées

Si on teste dans chaque condition expérimentale les mêmes sujets on est dans le cas des mesures répétées. D'une condition à l'autre on n'a donc pas de vraies réplications, les mesures sur un même individu représentent un block. Si on a hétéroscédasticité ou si un des échantillons n'est pas normal il est plus prudent d'utiliser un test non-paramétrique, par exemple le test de Friedman. Les données ont le même format comme dans tous les autres tests à plusieurs échantillons : une colonne des mesures, une colonne des conditions expérimentales, et une colonne indiquant le sujet testé.

Exemple Reprenons l'exemple des 7 souris soumis à 4 séances d'exploration d'un objet et on mesure chaque fois le temps d'exploration.

```
R> load("aovMesRepete.rda")                                   # charger les données
R> boxplot(sourisApp$explore ~ sourisApp$seance)                    # visualisation
```

On s'aperçoit qu'un des échantillons n'est pas normal,

```
R> lapply(split(sourisApp$explore,sourisApp$seance),shapiro.test)
```

Par prudence on procède donc au test non-paramétrique de Friedman, le modèle statistique étant le temps d'exploration en fonction des séances, sachant que chaque sujet a été testé à chaque séance :

```
R> friedman.test(explore ~ seance | sujet, data=sourisApp)
```

Rapporter les résultats Due à la non-normalité de l'échantillon à la quatrième séance on passe à un test de Friedman pour voir si les souris changent leur temps d'exploration au cours des séances. L'effet est significatif à $\alpha = 0.05$ ($\chi^2_r = 14.49$, $df = 3$, $p = 0.0023$).

Pour aller plus loin Voir chapitre 12.8 dans Zar (1999).

3.6 Les ANOVA à 2 facteurs

Une variable dépendante tel que le temps d'exploration d'un objet ne dépend pas uniquement du seul facteur apprentissage mais de nombreux autres facteurs ou variables indépendantes : le sexe de l'individu, la lignée génétique, le traitement médical, ... Le concept de l'analyse de variance permet de tester simultanément l'effet de plusieurs facteurs et (ce qui est encore plus important) de détecter des interactions entre ces facteurs. Prenons un exemple concret, l'ingestion des moutons en fonction du sexe (1^{er} facteur) et du type de groupe, unisexe ou mixte (2^{nd} facteur). Une ANOVA à 2 facteurs teste simultanément trois H_0 :

H_0 : le sexe n'influence pas l'ingestion des moutons
H_0 : le type de groupe n'influence pas l'ingestion
H_0 : il n'y a pas d'interaction entre sexe et type de groupe

Que veut dire interaction ? Il y a interaction si, par exemple, les femelles mangent autant que les mâles dans un groupe mixte, mais qu'elles mangent plus quand elles sont toutes seules (unisexe). Cette interaction représente donc une information biologique qu'on ne peut pas mettre en évidence en faisant sur l'ensemble des mesures deux ANOVAs à un facteur (une ANOVA par rapport au sexe et une autre ANOVA par rapport au type de groupe).

Exemple

```
R> load("plusieursEchantillons.rda");ls()     # chargement des données, affichage
R> boxplot(nbBouchParMin ~ sexe*condExp,data=moutonsBrout2)       # visualisation
```

La variable `moutonsBrout2` contient en première colonne le nombre de bouchées par minute, une deuxième colonne représentant le facteur 'sexe' et une troisième colonne représentant le facteur 'type de groupe' (groupe mixte ou unisexe). On a déjà vu que ces données sont homoscédastiques.

```
R> summary(aov(nbBouchParMin ~ sexe*condExp,data=moutonsBrout2))
```

Notez la forme du modèle statistique, `nbBouchParMin ~ sexe*condExp`, ou le `*` indique qu'on ne veut pas seulement l'effet des facteurs `sexe` et `condExp`, mais aussi l'interaction entre ces deux facteurs. Le résultat nous donne trois valeurs de p, une pour chaque H_0. Seul celui du facteur 'type de groupe' est significatif, il n'y a pas d'effet 'sexe' ou d'interaction.

Rapporter les résultats Une ANOVA à deux facteurs montre que le nombre de bouchées par minute des moutons dépend significativement du facteur 'type de groupe' ($F = 4.04$, $dl = (74, 1)$, $p = 0.048$), mais pas du facteur `sexe` ($F = 0.057$, $dl = (74, 1)$, $p = 0.812$) et il n'y a pas d'interaction entre ces deux facteurs ($F = 0.1575$, $dl = (74, 1)$, $p = 0.693$).

Les mesures répétées dans une ANOVA à deux facteurs

Un autre exemple d'application d'une ANOVA à deux facteurs est le cas ou on teste l'effet simultané de l'apprentissage des souris (temps d'exploration d'un objet, chapitre 3.5) et l'effet d'un traitement par AP5 (un neuro-inhibiteur).

```
R> load("aovMesRepete.rda");ls()                              # charger les données
R> boxplot(explore ~ trmt*seance,data=sourisAP5)                    # visualisation
```

Dans ce cas on a deux échantillons, un à 7 sujets contrôle (NaCl) et un autre à 9 sujets (AP5). Chaque sujet passe par 4 séances d'exploration. On utilise de nouveau une ANOVA à mesures répétées et on teste l'effet du facteur 'traitement' et du facteur 'séance', en indiquant que les 'sujets' sont testés 4 fois.

```
R> summary(aov(explore ~ trmt*seance + Error(sujet),data=sourisAP5))
```

Cette ANOVA détecte un effet séance et une interaction (qui se voit sur le graphique : mêmes

temps d'exploration avec AP5, mais temps d'exploration diminuant avec NaCl), mais pas d'effet traitement. On peut visualiser les résultats pour chaque facteur par les graphiques suivants :

```
R> boxplot(explore ~ trmt,data=sourisAP5)                    # pas d'effet traitement
R> boxplot(explore ~ seance,data=sourisAP5)                # effet global entrainement
```

Rapporter les résultats Une ANOVA à deux facteurs (dont un à mesures répétées) a révélé un effet significatif pour l'entraînement ($F = 5.26$, $df = (3, 42)$, $p = 0.004$), pas d'effet traitement ($F = 1.66$, $df = (1, 14)$, $p = 0.22$) et une interaction significative entre les deux ($F = 7.01$, $df = (3, 42)$, $p < 0.001$).

Remarque. Il n'y a pas de test non-paramétrique pour détecter des interactions. Dans l'exemple ci-dessus on a utilisé l'ANOVA paramétrique malgré une légère hétéroscédasticité dans les données,

```
R> bartlett.test(split(sourisAP5$explore,list(sourisAP5$trmt,sourisAP5$seance)))
```

en évoquant la robustesse de l'ANOVA.

Application aux courbes d'apprentissage succès-échec

Il est aussi possible de faire des ANOVA à deux facteurs si l'un des facteurs est répété et binaire (oui/non ou succès/échec) et l'autre un facteur traitement ou autre. Bien sur, un échantillon avec des valeurs 0 et 1 ne peut pas être distribué selon une distribution normale. Mais, et voici une autre application du théorème de la limite centrale, la moyenne des valeurs 0 et 1 le sera si l'échantillon est suffisamment grand (Lunney, 1970). Ce raisonnement justifie de faire une ANOVA même si le facteur prend des valeurs binaires.

Exemple Reprenons les données de Giurfa (2004) qu'on a déjà vues partiellement sur la page 16. Il s'agit des abeilles qui apprennent dans un tunnel à choix binaire que la présence de nourriture est indiquée par une couleur précise (dans ce cas c'est le violet, 15 séances d'entraînement). Le deuxième facteur concerne la façon dont se présente le choix non-rémunéré : soit il n'y a aucune couleur (apprentissage absolu), soit il y a une couleur légèrement différente de la couleur rémunérée (par rapport à la perception de l'abeille cette autre couleur est le bleu, apprentissage différentiel). Est-ce que les abeilles apprennent de la même façon dans ces deux conditions ?

```
R> load("giurfa004.rda")                                          # charger les donnees
```

Le data.frame 'abeilles' contient 4 colonnes : 'result' est le choix de l'abeille (1=correct, 0=faux), 'expCond' la condition expérimentale (abs et diff), 'trial' la séance et 'individual' l'identité de l'individu (indiqué par un chiffre de 1 à 30).

```
R> summary(aov(result ~ expCond*trial + Error(individual),data=abeilles))
```

Rapporter les résultats Une ANOVA à deux facteurs (dont un répété) démontre qu'il y a un effet global d'apprentissage ($F = 3.68$, $df = (14, 392)$, $p < 0.001$), un effet de la condition expérimentale ($F = 29.75$, $df = (1, 28)$, $p < 0.001$), mais il n'y a pas d'interaction entre apprentissage et la condition expérimentale ($F = 1.09$, $df = (14, 392)$, $p = 0.36$). On peut donc conclure que l'acquisition est signifcativement différente entre apprentissage absolue et apprentissage différentiel.

Les post-hoc et les ANOVA à 2 facteurs

S'il n'y a pas interaction on peut détecter les différences entre les échantillons d'un facteur particulier par un test post-hoc habituel (Tukey HSD ou autre). Dans le cas d'une interaction significative il n'y a pas de test post-hoc.

3.7 MANOVA et l'ANOVA à mesures répétées

Revenons sur les courbes d'apprentissages des souris sous l'effet du neuro-inhibiteur AP5 (chapitre 3.6). L'ANOVA à mesures répétées avais montré qu'il n'y a pas d'effet traitement, mais un effet séance et une interaction entre les deux. Cette interaction nous indique en fait que le traitement a néanmoins un effet dans le sense que les souris AP5 apprennent différemment au cours des séances comparées au souris du contrôle.

Une analyse alternative pour tester directement l'effet traitement est la MANOVA (multivariate analysis of variance). Elle interprète la performance à chaque séance comme une variable

dépendante, le fait qu'on la mesure sur le même individu n'a du coup plus d'importance, au lieu d'avoir des mesures répétées on augmente le nombre de variables dépendantes. La MANOVA regarde ensuite le vecteur des performances moyennes pour chaque séance $(\bar{s}_1, \bar{s}_2, \bar{s}_3, \bar{s}_4)$ et teste si l'orientation de ce vecteurs change de façon significative entre traitement AP5 et contrôle. Comme il est difficile de visualiser un espace de 4 dimensions la Fig 3.3 le fait pour deux variables dépendantes à la fois, comparant les séances 1 et 2, et ensuite séances 1 et 4 (où on voit que l'effet traitement devient significatif, voir l'exemple ci-dessous).

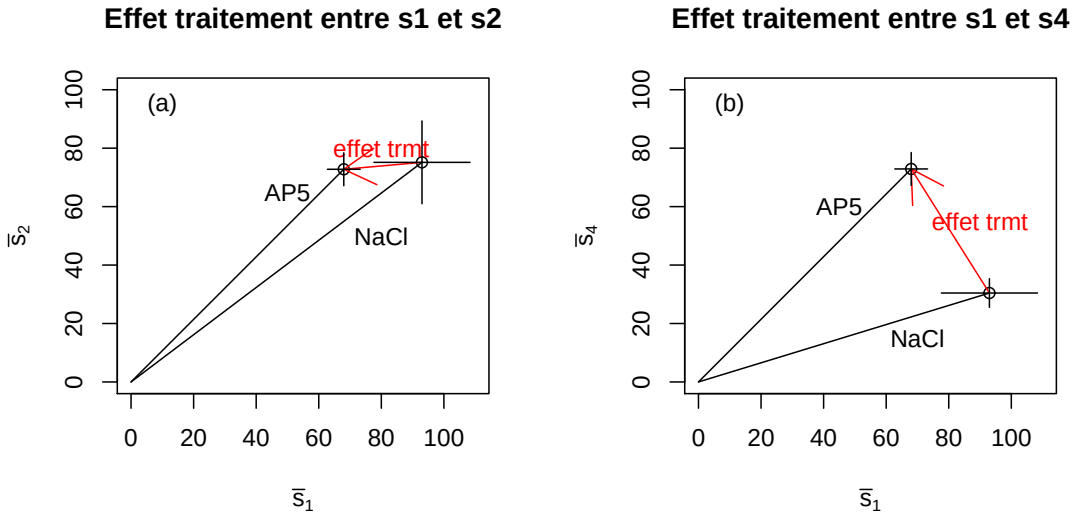

FIGURE 3.3 – Effet du traitement (NaCl contre AP5) sur le temps d'exploration d'un objet nouveau par les souris, séance 1 contre séance 2 (a) et séance 1 contre séance 4 (b).

Exemple Refaisons d'abord l'ANOVA à mesures répétées pour comparer à la suite

```
R> load("sourisPourManova.rda")                          # charger les données
R> boxplot(explore ~ seance + trmt,data=sourisA)
R> summary(aov(explore ~ trmt*seance+Error(sujet),data=sourisA))
```

La sphéricité de ce jeu de données est correcte

```
R> sourA = as.matrix(sourisAM[,2:5])              # extraire la matrice des données
R> mauchly.test(lm(sourA ~ 1))
```

mais les données ne sont par parfaitement normale

```
R> lapply(split(sourisA$explore,sourisA$seance), shapiro.test)
```

Une des conditions d'application d'une MANOVA est que les données sont issues d'une distribution normale multivariée (voir Zar 1999, ch. 16), ce qui est le cas pour nos données

```
R> library(energy)                                  # voir ch. A.2 en cas d'erreur
R> mvnorm.etest(sourA)                            # tester la normalité multivariée
```

Mais, comme pour l'ANOVA, la MANOVA est assez robuste vis à vis des déviations de normalité ou d'homoscédasticité, c'est plutôt la puissance qui souffre dans ces cas.

Pour faire une MANOVA il faut maintenant spécifier dans le modèle statistique qu'il y a plusieurs variables dépendantes

```
R> souris.manova = manova(cbind(s1,s2,s3,s4) ~ trmt, data=sourisAM)
R> summary(souris.manova)
```

et on voit qu'il y a un fort effet traitement ($F = 6.5040, df = (4, 11), p = 0.0061$). En fait, il faut être un peu plus prudent parce que le F calculé par la MANOVA n'est qu'une approximation (les statisticiens n'ont pas encore trouvé une formule exacte) et il y a plusieurs façons d'approximer cet F. Le mieux est de vérifier en essayant aussi les autres approximations

```
R> summary(manova(cbind(s1,s2,s3,s4) ~ trmt, data=sourisAM),test="Pillai")
R> summary(manova(cbind(s1,s2,s3,s4) ~ trmt, data=sourisAM),
+ test="Hotelling-Lawley")                                                    #
```

```
R> summary(manova(cbind(s1,s2,s3,s4) ~ trmt, data=sourisAM),test="Roy")
```
On obtient avec toutes les méthodes un F significatif, c'est rassurant.

On peux maintenant se poser la question à partir de quelle séance l'effet traitement devient significatif. En faisant les trois comparaisons il faut, comme dans le chapitre 3.10, corriger le α afin de conserver le même α global : un résultat n'est donc significatif que si $p < 0.05/3 = 0.017$:
```
R> summary(manova(cbind(s1,s2) ~ trmt, data=sourisAM))
R> summary(manova(cbind(s1,s3) ~ trmt, data=sourisAM))
R> summary(manova(cbind(s1,s4) ~ trmt, data=sourisAM))
```
On voit (comparer à Fig 3.3) qu'entre séance 1 et séance 2 l'effet n'est pas encore significatif ($F = 1.56$, $dl = (2, 13)$, $p = 0.247$) mais le devient en comparant la séance 1 à la séance 4 ($F = 14.05$, $dl = (2, 13)$, $p = 0.00056$).

Pour aller plus loin La MANOVA est disctué dans le chapitre 16 de Zar (1999), en particulier les conditions d'application (normalité et homoscédasticité). Voir aussi Potvin et al. (1990) qui compare entre ANOVA à mesures répétées et MANOVA.

3.8 Les ANCOVA : des facteurs à modalité fixe ou continue

Dans les ANOVA ci-dessus les facteurs contiennent chacun au moins deux modalités, mais ces modalités restent fixes (sexe, nombre d'entraînement, présence/absence d'un traitement, ...). Il arrive qu'il y ait un autre facteur qu'on ne peut pas contraindre à une valeur fixe et qui varie de façon continue tout au long de l'expérience. Ceci ajoute de la variabilité à notre variable dépendante qui peut masquer un effet de la variable indépendante à modalités fixes.

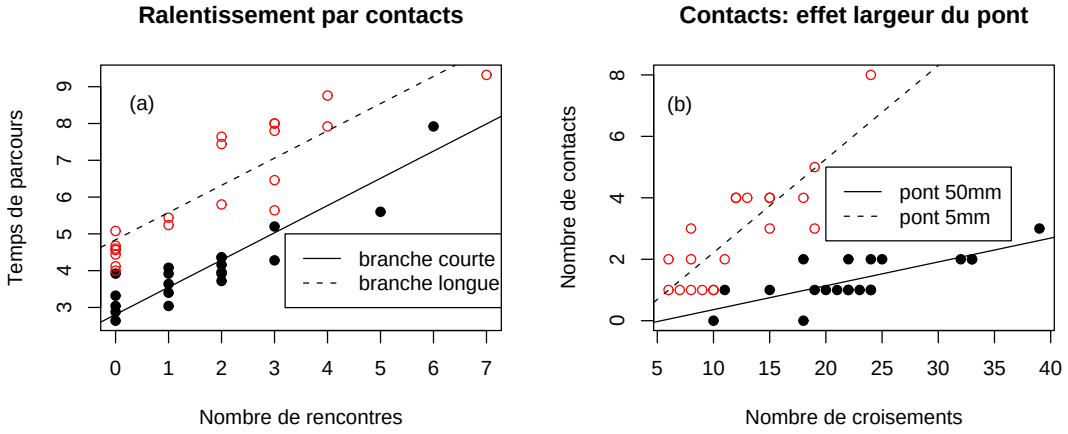

FIGURE 3.4 – Analyse de la fluidité du trafic des fourmis et des facteurs qui l'influencent. (a) dissociation de l'effet de la longueur d'un pont et du nombre de rencontres avec d'autres fourmis sur la durée de passage sur le pont (ANCOVA). (b) effet du nombre de rencontres et de la largeur du pont sur le nombre de contacts physiques entre fourmis : on voit la présence d'une interaction entre les deux facteurs.

Prenons comme exemple la durée de trajet d'une fourmi sur un pont entre le nid et la source de nourriture (Fig 3.4a). On se demande si cette durée augmente avec la longueur du pont. Pour répondre à cette question on pourrait directement mesurer les durées de passages sur deux ponts de longueur différente et comparer les deux échantillons avec un test de Student. Mais, sur un pont long une fourmi croise en moyenne beaucoup plus d'autres fourmis que sur un pont court, et à chaque contact la fourmi va s'arrêter un moment pour faire quelques contacts antennaires. Une augmentation de la durée du passage pourrait donc être due à l'augmentation du nombre de contacts plutot que due à la longueur du pont. Il faut donc dissocier l'effet du nombre de rencontres (variable indépendante à modalité continue, qu'on appelle la covariable) et l'effet de la longueur du pont (variable indépendante à deux modalités, long et court). Une ANCOVA (analyse de covariance) permet de faire cette dissociation. En fait, elle combine les techniques d'ANOVA avec celles de la régression linéaire (voir chapitre 4), mais elle teste les mêmes H_0 comme l'ANOVA

à deux facteurs,

H_0 : la longueur du pont n'influence pas la durée du trajet
H_0 : le nombre de contacts n'influence pas la durée du trajet
H_0 : il n'y a pas d'interaction entre longueur et nombre de contacts

```
R> load("traffic.rda")                                    # charger les données
R> trafficTempsParcours; attach(trafficTempsParcours)
R> is.factor(typeBranche)                    # c'est bien un facteur à effet fixe
R> is.factor(rencontres);is.numeric(rencontres)                  # effet continu
R> plot(tpsParcours ~ rencontres)           # relation temps de passage et nombre
rencontres
R> plot(tpsParcours ~ typeBranche)    # relation temps de passage et type branche
```

On voit que le temps de passage augmente aussi bien avec la longueur du pont comme avec le nombre de rencontres. Faisons d'abord une analyse globale, le temps de passage en fonction linéaire de la longueur du pont, du nombre de rencontres et de l'interaction entre les deux :

```
R> model1 = lm(tpsParcours ~ typeBranche*rencontres)            # modèle linéaire
R> summary.aov(model1)                                                        #
```

On voit que l'interaction entre la longueur du pont et le nombre de rencontres n'est pas significative, ce qui signifie que les deux droites de régression ont la même pente. C'est très bien, on peut continuer avec notre ANCOVA

```
R> model2 = lm(tpsParcours ~ typeBranche+rencontres)            # même chose sans
interaction
R> summary.aov(model2)                                    # quelles sont les effets?
```

La dernière commande nous indique qu'en fait les deux facteurs ont un effet significatif. Ceci veut dire que la pente de la relation entre nombre de contacts et rencontres est significativement différente de 0 et que pour un nombre de rencontres donné le temps de passage est plus long sur les branches longues. Pour tracer les deux droites ajustées (temps de passage en fonction du nombre de rencontres) il faut regarder les paramètres ajustés :

```
R> summary.lm(model2)              # cette sortie demande plus d'explications ...


Call:
lm(formula = tpsParcours \~{} typeBranche + rencontres)
# rappel du modèle statistique ajusté
Residuals:
    Min      1Q  Median      3Q     Max
-1.4218 -0.4271 -0.1353  0.4057  1.3189
# quelques statistiques sur les résidues
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)     2.8088     0.1788  15.711  < 2e-16 ***
# l'ordonnée à x=0 pour la droite de régression de la branche courte bc,
# significativement différente de 0
typeBranchebl   2.0309     0.2075   9.788 8.21e-12 ***
# différence avec l'ordonnée à x=0 de la droite par la branche longue
# (qui vient en second parce que bc passe avant bl dans l'alphabet)
rencontres      0.7407     0.0603  12.283 1.27e-14 ***
# la pente de la droite, sans interaction c'est la même pour bc et bl

Residual standard error: 0.655 on 37 degrees of freedom
Multiple R-Squared: 0.8761,Adjusted R-squared: 0.8694
F-statistic: 130.8 on 2 and 37 DF,  p-value: < 2.2e-16
```

Les trois dernières lignes méritent un commentaire spécifique : j'ai souvent appelé l'objet `y ~ x+z` un modèle statistique. En fait, on est effectivement en train de faire de la modélisation en nous laissant guider par les données. C'est une modélisation purement empirique et à distinguer des modèles mécanistes que vous trouvez d'habitude sous la dénomination 'modélisation', mais c'est

une modélisation néanmoins. Le chiffre $R^2 = 0.8761$ (multiple R-squared) veut dire que 87.61% de la variation totale dans les données peut être expliqué par notre modèle ou, en d'autres mots, les variables indépendantes `typeBranche` et `rencontres` peuvent expliquer 87.61% de la variabilité dans la variable dépendante `contacts` [3]. Pour toutes les modélisations statistiques le R^2 est donc un indicateur de la qualité du modèle.

Toute cette information nous permet enfin de tracer le graphique tel qu'il est dans la Fig 3.4a :

```
R> plot(rencontres, tpsParcours, xlab="Nombre de rencontres",         # graphique
+ ylab = "Temps de parcours", type='n',     # type = 'n', ne pas tracer les points
+ main="Ralentissement par contacts")                          # titre du graphique
R> tc = split(tpsParcours,typeBranche)     # séparer temps de passage par branche
R> tp = split(rencontres,typeBranche)            # même chose pour les rencontres
R> points(tp[[1]], tc[[1]],pch=16)   # tracer les points de la branche courte ...
R> points(tp[[2]], tc[[2]],col='red')                # ...et de la branche longue
R> abline(2.8088, 0.74)             # régression linéaire pour branche courte ...
R> abline(2.8088+2.0309, 0.74, lty=2)                      #  ...et branche longue
R> legend(3.5,5,lty=c(1,2),legend=c("branche courte","branche longue"))
```

ANCOVA : et s'il y a interaction ?

Regardons maintenant un autre exemple (Fig 3.4b). Il s'agit d'une étude du nombre de contacts physiques entre fourmis champignonnistes *Atta collombica* qui circulent sur un pont de largeur 5mm ou 50mm. La covariable est le nombre de croisements de fourmis sur le pont (le nombre de contacts est évidemment lié au nombre de rencontres). La première étape de l'analyse est équivalente au dernier exemple,

```
R> load("traffic.rda")                                  # si ce n'est pas déjà fait
R> names(trafficContactRencontres); attach(trafficContactRencontres)
R> model = lm(contact ~ pont*rencontre)                       # modèle statistique
R> summary.aov(model)                                      # résultat de l'anlyse
```

```
               Df Sum Sq Mean Sq F value    Pr(>F)
pont            1 24.025  24.025  37.764 4.447e-07 ***
rencontre       1 33.011  33.011  51.889 1.779e-08 ***
pont:rencontre  1 16.436  16.436  25.835 1.166e-05 ***
Residuals      36 22.903   0.636
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Ce résultat indique que les deux facteurs (largeur du pont et nombre de rencontres) ont un effet hautement significatif, mais qu'il y a aussi une interaction significative (ce qui veut dire que la pente de la régression est différente pour les deux largeurs de pont, voir Fig 3.4b). Cette interaction veut aussi dire que l'influence qu'a la largeur du pont sur le nombre de contacts dépend du nombre de croisements : pour des faibles nombres la largeur ne joue aucun rôle, mais pour des grands nombres de croisements le nombre de contacts est plus élevé sur le pont de 5mm (Fig 3.4a).

Enfin, la commande

```
R> summary.lm(model)
```

nous donne les paramètres estimés qui sont à interpréter comme dans la section précédente, la ligne supplémentaire contenant la différence entre les deux pentes,

```
pont5mm:rencontre  0.22704    0.04467   5.083 1.17e-05 ***
```

qui est hautement significative. On obtient aussi un $R^2 = 0.7624$, le modèle statistique explique donc 76.24% de la variabilité dans nos mesures.

Pour aller plus loin L'ANCOVA fait partie du grand champs des régressions multiples qui permettent d'étudier l'influence de plusieurs variables indépendantes (et des interactions entre elles) sur une variable dépendante et de sélectionner les plus pertinentes, voir Aiken & West (1991) et Engqvist (2005) pour un commentaire récent par rapport à l'application en comportement animal.

3. Dans la sortie de **R** on trouve également le 'adjusted R-squared' qui prend en compte la taille de l'échantillon, N, et le nombre de VI, k, par la formule $R^2_{adj} = 1 - (1 - R^2)\frac{N-1}{N-k-1}$.

3.9 Agrégation et ségrégation entre individus

La distribution des animaux dans l'espace donne des indications sur leur comportement social, par exemple s'ils ont un comportement d'agrégation plutôt que d'être distribués de façon aléatoire (grégarité). Regardons par exemple la distribution de 30 moutons dans une enceinte de 35 sur 30 m (Fig 3.5a) : est-ce que ces moutons sont regroupés ensemble ou sont-ils aléatoirement distribués ? Un paramètre statistique classique pur quantifier la dispersion dans un groupe est la distance moyenne au voisin le plus proche (VPP). Cette distance peut être comparée à la distance attendue sous une distribution aléatoire (par exemple la distribution Poissonienne où x et y sont tiré au hasard de façon uniforme dans l'enceinte où se trouvent les moutons). Le plus simple est de calculer cette distribution attendue directement par simulation Monte Carlo : a) tirer au hasard autant de coordonnées x et y que de moutons, b) calculer la distance moyenne au VPP, et c) répéter a) et b) de nombreuses fois pour calculer la proportion de simulations pour lesquelles cette distance est plus petite que la distance avec les coordonnées mesurées. Cette proportion est une estimation du p pour

H_0 **:** la distance moyenne au VPP est plus grande ou égale à cette distance attendue sous une distribution Poissonienne

et si $p < \alpha$ on rejette H_0, c'est-à-dire il y a agrégation.

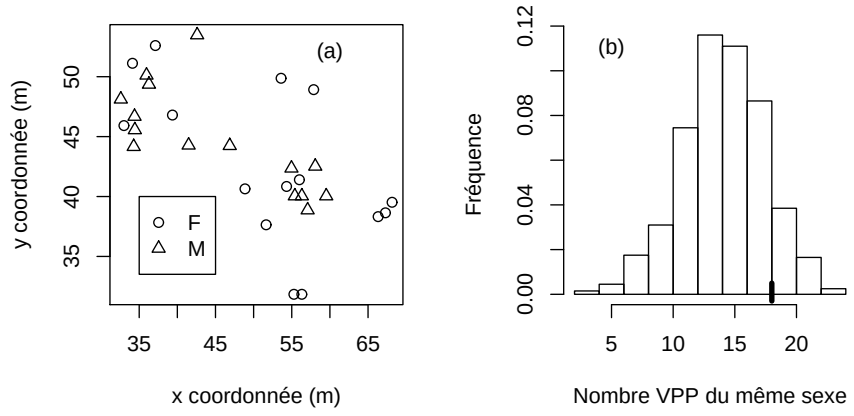

FIGURE 3.5 – Analyse de ségrégation sexuelle : (a) les positions des mâles et femelles dans l'arène, (b) distribution du nombre où le voisin le plus proche est du même sexe sous permutation aléatoire, la ligne épaisse indiquant la valeur des données originales.

Exemple

```
R> source("fonctionsCompStat.R")          # charger la fonction 'distMoyenneVPP'
R> load(file="moutonsScans.rda")                          # charger les données
R> x = mout3$AvantX                       # coordonnées x des têtes des moutons
R> y = mout3$AvantY                                          # coordonnées y
R> plot(x,y, type='p', xlab="x (m)", ylab="y (m)")     # graphique des positions
R> distOrig = distMoyenneVPP(x, y); distOrig            # calcul des la distance
```

Pour tester s'il y a agrégation on va simuler maintenant la dispersion de 1000 troupeaux de moutons virtuelles (Monte Carlo) si les individus se placent aléatoirement à l'intérieur de leur champ (délimité dans la Fig 3.5 par $x \in (30, 70)$ et $y \in (30, 55)$) :

```
R> nombreMC = 1000;
R> nombreBetes = length(x);                 # il y a 30 moutons dans le troupeau
R> lesDistMC = rep(0,nombreMC);            # vecteur pour les distances simulées
R> for (k in 1:nombreMC) {
+  xMC = 30 + runif(nombreBetes) * 40                  # positions x aléatoires
+  yMC = 30 + runif(nombreBetes) * 25;                 # positions y aléatoires
+  lesDistMC[k] = distMoyenneVPP(xMC, yMC);             # calcul de la distance
```

```
+ }                                                                    #
R> hist(lesDistMC)                          # distribution des distances sous H0
R> tmp = which(lesDistMC < distOrig)       # détection des distances inférieurs à
distOrig
R> length(tmp)/nombreMC                                    # estimation de p
```
On voit que $p > 0.05$, ce troupeau de moutons n'est donc pas agrégé.

Détection de la ségrégation sexuelle par permutation

Une technique similaire permet de tester s'il y a ségrégation sexuelle dans un troupeau. L'idée est de calculer le nombre de fois qu'un VPP est du même sexe et de regarder si ce nombre est significativement plus grand que ce qu'on attendrait sous

H_0 **:** le sexe du VPP est aléatoire ou du sexe opposé

Pour cela on fait des simulations Monte Carlo où on distribue le sexe aléatoirement sur tous les individus. La proportion de simulations ou le nombre de VPP du même sexe est plus grand que celui avec le sexe original est une estimation du p associé à H_0.

Exemple

```
R> mout = mout3                        # on travaille avec le jeu de données mout3
R> x = mout$AvantX                          # coordonnées x des têtes des moutons
R> y = mout$AvantY                                          # coordonnées y
R> sexe = mout$Sexe                                  # le sexe des moutons
R> plot(x,y, type='p',xlab="x (m)", ylab="y (m)", pch=as.numeric(sexe))
R> sexeVPP = nombreVPPmemeSexe(x,y,sexe); sexeVPP          # calcul VPP même sexe
```
Le VPP est du même sexe dans 21 sur 30 cas. Est-ce que c'est significativement plus grand que 15 sur 30 ? Pour répondre on va maintenant faire 1000 simulations avec le sexe permuté pour estimer la distribution du nombre de VPP du même sexe sous H_0 :

```
R> nombrePerm = 1000                              # nombre de permutations à faire
R> memeSexePerm = rep(0,nombrePerm) # vecteur pour les résultats des permutations
R> for (k in 1:nombrePerm) {
+  sexePerm = sample(sexe,replace=F)              # attribution aléatoire du sexe
+  res = nombreVPPmemeSexe(x,y,sexePerm)              # calul de VPP du même sexe
+  memeSexePerm[k] = res$meme               # garder cette valeur dans le vecteur
+ }                                                                    #
R> hist(memeSexePerm)                   # histogramme des VPP du même sexe sous H0
R> comp = which(memeSexePerm > sexeVPP$meme)     # détection des simulations avec
nombre VPP du même sexe supérieur à 21
R> length(comp)/nombrePerm                                 # estimation de p
```
On obtient environ $p = 0.024 < 0.05$, on peut donc rejeter H_0, il y a ségrégation sexuelle dans ce troupeau.

Pour aller plus loin La littérature sur les indices d'agrégation ou de ségrégation est assez vaste, mais Dixon (1994) et Conradt (1998) donnent une bonne introduction au sujet, sachant que ce domaine d'analyse est en pleine évolution. Une approche plus technique mais potentiellement intéressante sont les fonctions de Ripley (Ripley, 1988), mais leur utilité pour l'étude comportementale reste encore à démontrer.

3.10 Du 'data-mining' et de la détection d'effets significatifs

Dans une étude d'exploration générale (ou 'data-mining' en anglais) on ne sait pas d'avance quelles composantes d'un système sont affectées par la variation d'un facteur : on les mesure donc tous et on fait un test d'hypothèse séparément pour chaque composante. Cette approche comprend un très grand risque, le fait d'utiliser le même α pour chaque test est de croire que cet α représente le risque de rejeter H_0 à tort pour chacune des composantes testées.

Comment expliquer le problème ? Jetez un dé trois fois. A chaque essai vous avez une probabilité d'$\frac{1}{6} = 0.167$ d'avoir le chiffre 6. Quelle est maintenant la probabilité d'avoir au moins une fois un

6 ? Elle est de un moins la probabilité de ne jamais avoir de 6,

$$1 - \left(\frac{5}{6}\right)^3 = 0.421,$$

cette probabilité est donc nettement supérieure à 0.167. Dans le 'data-mining' c'est exactement le même problème. Supposons que vous testiez dix composantes (donc dix H_0 que le facteur n'influence pas cette composante), et supposons en plus que toutes les H_0 soient vraies. Avec un $\alpha = 0.05$, quel est le risque de commettre au moins une fois une erreur de type I (c'est-à-dire qu'au moins un des dix tests soit significatif) ?

$$1 - \left(\frac{95}{100}\right)^{10} = 0.401$$

C'est beaucoup plus grand que le α initial. [4]

Comment remédier au problème ? La litérature propose plusieurs solutions qui sont toutes basées sur une correction d'α afin de maintenir un risque global d'erreur type I à la valeur de l'α initiale (correction de Bonferroni ou de Dunn-Sivak, correction séquentielle de Bonferroni, ...). Toutes ces corrections diminuent la puissance globale de votre série de tests, il est donc toujours utile de ne pas mesurer tous et toutes mais de se limiter aux composantes qui vous intéressent vraiment ou qui sont susceptibles d'être influencées par le facteur.

Exemple Prenons une étude de neuro-anatomie fonctionnelle sur les effets du sommeil sur l'activité neuronale dans 12 régions du cerveau. Cette activité est mesurée par un marqueur d'activation (FOS), Le facteur comprend trois modalités (PSC = contrôle, PSD = privation de sommeil et PSR = récupération après privation de sommeil) et pour chaque région on a fait une ANOVA à un facteur. Le Tab 3.2 résumé les résultats. Pour détecter les ANOVAs significatives **R** ne corrige pas les α mais nous rend des valeurs de p corrigées qu'on compare ensuite à l'α initial. Il faut indiquer la correction à utiliser, 'holm' choisi ci-dessous correspond à une correction séquentielle de Bonferroni :

```
R> load("neuroAnat.rda")                          # charger les valeurs de p
R> p.neuro.anat                                      # afficher les valeurs
R> table(p.neuro.anat < 0.05)                 # dix p au-dessous de 0.05, mais ...
R> p.adjust(p=p.neuro.anat,method='holm')                          # correction
```

Avec le choix $\alpha = 0.05$ on voit que seulement la $3^{ème}$ région, vlPAG, est influencée de façon significative par le sommeil (et non pas 10 régions comme on le pourrait conclure naïvement).

Pour aller plus loin L'article de Rice (1989) donne une bonne introduction dans la problématique.

3.11 La puissance statistique

Dans le chapitre 3 on a discuté les différents types d'erreurs qu'on peut commettre dans le cadre de tests d'hypothèse (Tab 3.1) : rejeter une H_0 qui, en réalité, est vraie (erreur type I, associé à α) et ne pas rejeter une H_0 qui, en réalité, est fausse (erreur type II, associé à β).

Dans tout ce qui précède on a seulement considéré les erreurs de type I en calculant un p (probabilité de commettre une telle erreur) et en décidant le rejet ou non-rejet de H_0 selon le α choisi au préalable (d'habitude $\alpha = 0.05$). Mais le risque de commettre une erreur de type II (β), c'est-à-dire de ne pas rejeter une H_0 qui, en réalité, est fausse, est aussi important et à prendre en compte (par exemple, si ce risque est proche de 1 c'est une perte de temps et d'argent de faire une expérience pour tester H_0). Ce risque est complémentaire à la **puissance**, c'est-à-dire la probabilité de rejeter H_0 si elle est effectivement fausse (puissance $= 1 - \beta$).

4. Un exemple récent de cette erreur est donné par une campagne publicitaire d'un leader mondiale de produits de tabacs (appelons-le x) qui prétend que fumer représente un risque bien inférieur d'attraper un cancer du poumon en comparaison avec d'autres facteurs complètement anodins tel que le fait d'aller 3 fois au cinéma par semaine. En fait, les employés de x avaient tout simplement pris une énorme base de données comprenant des centaines d'habitudes de personnes, et pour chacune des habitudes il ont fait un test de corrélation avec la présence ou absence d'un cancer du poumon. Bien naturellement, en piochant dans ces centaines d'habitudes ils en ont trouvé quelques unes qui étaient nettement mieux corrélées avec le cancer du poumon que le fait de fumer. Il reste à élucider s'il s'agit d'une ignorance parfaite du côté des statistitiens de x ou d'une campagne d'intox volontaire, les deux cas étant graves.

TABLE 3.2 – Activité neuronale (FOS) dans 12 régions du cerveau (rat) en fonction de trois régimes de sommeil (voir texte). L'activation (FOS) est indiquée par moyenne±erreur standard, et la dernière colonne indique le p calculé par l'ANOVA.

région	PSC	PSD	PSR	p
dPAG	8.0±5.4	29.8±5.6	144.5±56.2	0.0343
lPAG	9.3±4.8	50.0±9.6	47.8±11.5	0.0189
vlPAG	15.5±5.7	68.5±3.9	85.3±16.0	0.0021
CGPn	4.3±2.3	21.8±14.7	35.5±13.3	0.2132
DRN	4.0±2.1	31.0±7.9	7.5±4.7	0.0135
MnR	0.3±0.3	3.3±1.3	15.3±4.2	0.0057
PPTg	3.0±2.4	20.0±3.9	24.3±9.1	0.0690
PPTgM	0±0	0±0	11.8±3.4	0.0066
PPTg+PPTgM	3.3±2.3	23.3±4.1	37.0±12.3	0.0348
LDTg	13.5±7.0	22.5±2.6	63.0±15.6	0.0148
LDTgV	1.8±0.5	11.0±5.4	23.3±6.0	0.0293
LDTg+LDTgV	15.3±7.1	33.5±6.0	86.3±21.0	0.0112

TABLE 3.3 – Les composantes de la puissance statistique : ce tableau résume le lien entre la puissance et les trois autres paramètres statistiques, α, n et d (d'après Nakagawa & Foster 2004).

Paramètre statistique	Pour augmenter la puissance ...
Critère de significativité α	augmenter α
Taille de l'échantillon n	augmenter n
Taille d'effet d	augmenter d

Quand on planifie une expérience pour tester une hypothèse on veut naturellement une grande puissance (proche de 1). Mais cette puissance dépend directement de trois autres paramètres statistiques : a) la taille n de l'échantillon, b) l'α choisi et c) la taille de l'effet attendu dans l'expérience (voir le Tab 3.3 pour un résumé). Ce dernier terme, taille d'effet, demande plus de précision : il dépend en fait du type de test statistique utilisé et je vais l'illustrer ici uniquement pour le cas de la comparaison paramétrique entre deux populations. Supposons que ces populations aient les moyennes μ_1 et μ_2 et la même variance σ^2. Dans ce cas, l'effet d de la comparaison est

$$d = \frac{|\mu_1 - \mu_2|}{\sigma}$$

c'est-à-dire la différence entre les moyennes pondérées par l'écart type. Si on connaît trois de ces quatre composantes on peut calculer la quatrième. Le cas le plus fréquent est de connaître la puissance désirée, α et d et de calculer le n associé (voir ci-dessous).

Comme il y a une valeur standard pour α, il y en a aussi une pour $1-\beta$: la puissance devrait être ≥ 0.8 (Cohen, 1988). L'analyse de puissance a donc souvent pour but d'estimer la taille de l'échantillon nécessaire pour avoir cette puissance minimale. Comme on l'a vu ci-dessus, pour cela il faut connaître la taille de l'effet (d) : soit on connaît cette valeur approximativement à partir d'une pré-expérience, soit la littérature nous la donne. Ensuite il existe des calculs statistique pour calculer le n.

Exemple On compare deux échantillons dont on sait par une pré-expérience que la différence des moyennes est $\Delta = 3.5 \approx |\mu_1 - \mu_2|$, la variance est de 5^2 et on a choisi $\alpha = 0.05$ et une puissance $1 - \beta = 0.8$. H_0 est qu'il n'y a pas de différence entre les deux échantillons.

```
R> power.t.test(delta=3.5, sd=5,sig.level=0.05,power=0.8)
```

nous donne comme résultat que chaque échantillon devrait comprendre au moins 33 mesures pour rejeter H_0 avec le α et β choisi.

Inversement, on peut se mettre dans la situation d'une pré-expérience où on a deux échantillons de taille $n = 20$, une différence $\Delta = 3.5$, un écart type total de 5.0 et un $\alpha = 0.05$ (le test de Student donnait un $p > \alpha$, donc non-significatif),

```
R> power.t.test(n=20,delta=3.5, sd=5,sig.level=0.05)
```

nous dit que, si Δ et σ sont proches des valeurs aux niveaux des populations, notre test n'a qu'une puissance de 0.58. C'est un peu faible pour être sûr qu'il n'y a pas d'effet, il faudrait faire une

seconde expérience où on a augmenté la taille des échantillons à 40 (par exemple).

Dans le cas d'une ANOVA à un facteur l'effet d se calcule à partir de la variance entre les groupes (c'est-à-dire la variance des moyennes de chaque groupe) et de la variance intra-groupe (c'est-à-dire la moyenne des variances de chaque groupe). Reprenons l'exemple du temps que les moutons passent avec la tête dressée (chapitre 3.5).

```
R> load("plusieursEchantillons.rda")                          # charger les données
R> resultat = aov(tpsDroitRegard ~ type, data=moutonsBrout)
R> summary(resultat)                    # l'ANOVA nous donne un effet significative
```

On veut maintenant estimer la puissance de cette analyse :

```
R> parEchantillon = split(moutonsBrout$tpsDroitRegard, moutonsBrout$type)
R> EntreGrp = var(as.numeric(lapply(parEchantillon,mean)));      # variance entre
groupes
R> IntraGrp = mean(as.numeric(lapply(parEchantillon,var)))              # variance
intra-groupe
R> power.anova.test(groups=4,n=20, between.var=EntreGrp, within.var=IntraGrp)
```

On a donc une puissance de 0.90. Inversement, si on veut refaire l'expérience pour être sûr, tout en se contentant d'une puissance de 0.8, on voit

```
R> power.anova.test(groups=4,between.var=EntreGrp,within.var=IntraGrp,power=0.8)
```

que 16 moutons par groupes sera suffisant.

Pour aller plus loin Le commentaire de Nakagawa & Foster (2004) donne une très bonne introduction dans l'analyse de puissance et les problèmes associés. Zar (1999) donne les détails et les méthodes pour les tests statistiques qu'on a vus jusqu'à présent. On peut aussi se débrouiller avec une petite simulation *in silico* de l'expérience prévue (techniques du Monte Carlo pour l'analyse de puissance).

3.12 Au-delà des tests d'hypothèses

Chapitre 4

Ajustement de modèle : estimation et tests associés

Il arrive souvent qu'on mesure une variable dépendante y en fonction d'une variable indépendante x et qu'on a une idée claire de la façon dont y dépend de x, par exemple

- $y = ax + b$ est un modèle linéaire,
- $y = ax^{-b}$ est un modèle non-linéaire et
- $y = \frac{ax}{1+ahx}$ est un autre modèle non-linéaire.

Ce qui nous intéresse dans ce cas ce sont les valeurs des paramètres utilisés (a, b, h) et leur précision, c'est-à-dire leurs erreurs standards (voir chapitre 2). Mais on se demande aussi si le modèle proposé est une bonne description de la relation entre y et x (il est peut-être trop complexe, trop simple ou tout simplement faux). Pour tout cela il faut d'abord « ajuster » le modèle aux données : qu'est-ce que ça veut dire ? Regardez par exemple la Fig 4.1 : x représente le nombre d'orignaux sur l'Ile Royale (Amérique du Nord) et y est la réponse fonctionnelle des loups qui habitent aussi sur cette île (la réponse fonctionnelle est la quantité d'orignaux mangée par loup et par an). Ajuster une droite à ces données consiste à trouver les valeurs a et b du modèle linéaire ci-dessus tel que la distance verticale (au carré) entre valeurs d'y mesurée et valeur d'y prédite par le modèle pour un x devient minimal. On appelle ce type d'ajustement aussi la méthode des moindres carrés (least squares).

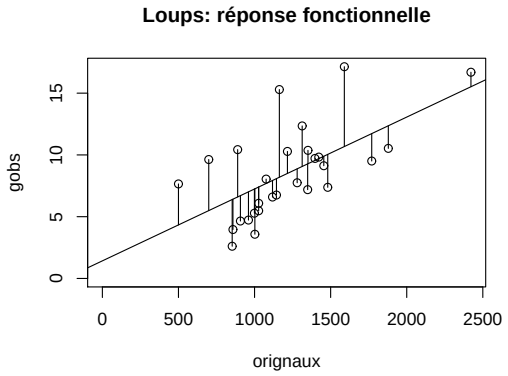

FIGURE 4.1 – Prédation sur l'Ile Royale : réponse fonctionelle des loups en fonction du nombre d'orignaux disponibles. La droite représente le meilleur ajustement par le modèle linéaire $y = ax+b$ et les traits verticaux indiquent les résidus (différence entre l'y mesuré et l'y prédit par le modèle). La somme de ces résidus au carré est minimal.

4.1 La régression linéaire

Si y est une fonction polynomiale de x (par exemple la droite $y = ax + b$ ou l'équation quadratique $y = ax^2 + bx + c$) on parle d'une régression linéaire. Trouver le meilleur ajustement du modèle est dans ce cas toujours possible et peu coûteux du côté temps de calcul.

Exemple Etudions directement la prédation des loups sur les orignaux.

```
R> load("predation.rda"); predation                    # charger le jeu de données
```

Veuillez noter que ce jeu de données contient trois colonnes : une mesure (`gobs`) de la réponse fonctionnelle (faite chaque année durant un mois d'observation, ensuite transformée pour avoir l'échelle temporelle 'proies par prédateur et par an'), une estimation du nombre d'orignaux sur l'île (`proie`) à ce moment-là, et une estimation du nombre de loups (`pred`) présents sur l'île.

```
R> attach(predation)                # rendre les colonnes accessible par leur nom
R> plot(proie, gobs, main="Loups: réponse fonctionnelle",
+ xlab="orignaux", ylab="gobs", ylim = c(0,max(gobs)), xlim=c(0,max(proie)))   #
```

La fonction dans **R** pour ajuster un modèle linéaire est `lm(...)` :

```
R> res = lm(gobs~proie); res        # la sortie 'res' est difficile à interpréter
R> abline(res)                                          # trace la droite ajustée
R> segments(proie,predict(res),proie,gobs)                    # trace les residus
```

Pour interpéter cet ajustement il faut faire

```
R> summary(res)
```

C'est la partie des coefficients qui nous intéresse le plus

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.423441   1.874309   0.759 0.454415
proie       0.005820   0.001466   3.969 0.000507 ***
```

La première ligne nous dit que l'ordonnée à l'origine (b) est estimée à 1.4234 ± 1.8743. Le $p = 0.454$ à la fin de la ligne teste $H_0 : b = 0$: nous voyons que b n'est pas significativement différent de 0 [1]. La deuxième ligne donne l'estimation de la pente, $a = 0.00582 \pm 0.00147$ et le $p = 0.0005$ indique que cette pente est significativement différente de 0. Cela veut dire que la quantité de proies disponibles change la réponse fonctionnelle de façon significative [2]. Finalement, le chiffre $R^2 = 0.3773$ (multiple R-squared) s'appelle le coefficient de détermination. Il chiffre la proportion de la variation totale qui est expliquée par le modèle ajusté, c'est donc une autre mesure de la qualité de l'ajustement qu'il faut mentionner dans un rapport (voir aussi page 31).

Ce n'est pas encore fini : pour s'assurer que l'ajustement est bon il faut que les résidus soient distribués de façon normale (voir 3.3) et qu'il n'y ait pas de corrélation entre les résidus (c'est-à-dire qu'il n'y ait pas trop de résidus voisins situés du même côté de la droite, ce qui indiquerait que le nuage de points ne représente pas une droite).

```
R> resid = gobs - predict(res);                              # calcul des residus
R> hist(resid)                                             # Histogramme des residus
R> shapiro.test(resid)                  # les résidus ne sont pas normales, p=0.008
```

Pour tester l'auto-corrélation entre résidus il faut d'abord les mettre dans le même ordre que les valeurs de l'abscisse :

```
R> sortedx = sort(proie,index.return=T)        # calculer l'ordre des valeurs de x
R> residOrd = resid[sortedx$ix]                # mettre les résidus dans cet ordre
R> acf(residOrd)             # On ne détecte aucune auto-corrélation significative
```

Enfin, faire un graphique des résidus en fonction de la variable dépendante est aussi un très bon test visuel si le modèle statistique est raisonnable, il faudrait voir un nuage de points parfaitement horizontal qui a partout une hauteur similaire,

```
R> plot(res$residuals ~ gobs)                          # résidus en fonction de la VD
```

On voit qu'il y a une tendance positive et une variablilité qui augmente de gauche à droite, indiquant que le modèle linéaire n'est pas très bien, on reviendra là-dessus.

1. C'est rassurant dans le sens que le modèle le plus fréquent pour modéliser la réponse fonctionnelle est celui de Lotka-Volterra, $g = a\,proie$, où a est le taux d'attaque.

2. La ressemblance avec une ANOVA n'est pas fortuite : les calculs dans une ANOVA et dans une régression linéaire sont tellement proches que si vous faites une ANOVA dont la variable indépendante est continue au lieu d'être des catégories, la commande `aov` fait *de facto* une régression linéaire (voir l'ANCOVA).

Rapporter les résultats Une analyse de régression linéaire ($R^2 = 0.38$) nous indique que la prédation des loups sur l'Ile Royale est une fonction linéaire du nombre d'orignaux avec pente $a = 0.00582 \pm 0.00147$ et ordonnée à l'orgine $b = 1.4234 \pm 1.8743$ (pas significativement différent de 0, $p = 0.008$). Cependant, les résidus ne sont pas distribués de façon normale (Shapiro-Wilk, $W = 0.8936$, $p = 0.008$).

Pour aller plus loin Le chapitre 17 dans Zar (1999) décrit la régression linéaire dans tous les détails.

4.2 La corrélation simple

Dans une régression linéaire on suppose implicitement que toute erreur de mesure se trouve dans la variable dépendante et que la variable indépendante est connue parfaitement (les résidus dans la Fig 4.1 sont verticaux). C'est pour cette raison qu'une régression linéaire $y \sim x$ ne donne pas le même résultat qu'une régression $x \sim y$. Dans une analyse de régression il faut donc bien choisir sa variable dépendante et indépendante.

Si les deux variables sont incertaines et qu'il n'y a aucune raison particulière de traîter l'une ou l'autre en variable dépendante on peut faire une analyse de corrélation simple. Pour cela on calcule un coefficient de corrélation r (à ne pas confondre avec le coefficient de détermination R^2) qui est compris dans l'intervalle $(-1, 1)$. Une valeur $r = 0$ indique que les deux variables ne sont pas corrélées, r proche de 1 qu'il y a une corrélation positive et r proche de -1 que cette corrélation est négative.

Exemple Un exemple légendaire est la corrélation entre l'abondance des cigognes et les nouveaux nés. Dans les années 1965, 1970, 1975 et 1980 on a compté 1900, 1400, 1050 et 900 couples de cigognes et 1.1, 0.88, 0.65 et 0.65 millions nouveaux nés en allemagne occidentale. Le coefficient de corrélation est

```
R> cig = c(1900, 1400, 1050, 900)                # saisir les couples de cigognes
R> nn = c(1.1, 0.88, 0.65, 0.65)                          # nouveaux nées en 10^6
R> cor(cig, nn)                             # calcul du coefficient de détermination
R> cor(nn, cig)                                           # donne la même valeur
```

On observe $r = 0.989$ (qui est très proche de 1) et on peut tester si ce r est significativement différent de 0

```
R> cor.test(cig, nn)
```

qui montre que r est significativement différent de 0 ($t = 9.4731$, $dl = 2$, $p = 0.01$). Je vous laisse faire l'interprétation biologique.

4.3 La régression non-linéaire

Continuons directement avec l'exemple ci-dessus. En traçant y en fonction de x on voit souvent que l'allure du nuage de points n'est pas du tout une droite mais plutôt une courbe (voir par exemple les points dans la Fig 4.2). Si cette courbe ne ressemble pas à un polynôme il faut ajuster le modèle adéquat par des techniques de régression non-linéaire. La différence majeure avec la régression linéaire est qu'il faut donner au logiciel des valeurs initiales des paramètres qui ne soient pas trop éloignées de leurs valeurs optimales (cela vient du fait que l'algorithme qui cherche les paramètres optimaux s'en approche en diminuant la somme des carrés des résidus de façon itérative jusqu'à trouver un minimum (moindres carrés), mais ce minimum peut être un minimum local, ou l'algorithme s'arrête pour d'autres raisons avant d'atteindre le minimum). Il faut donc d'abord trouver ces valeurs initiales, ensuite appliquer la régression non-linéaire et vérifier visuellement que l'ajustement du modèle au nuage de points semble raisonnable, pour finalement vérifier que les résidus sont normales et sans corrélation.

Reprenons comme exemple la prédation des loups. On a vu que la réponse fonctionnelle est assez bien décrits comme dépendant linéairement du nombre de proies disponibles. Mais il y a une théorie qui propose que la réponse fonctionnelle devrait être une fonction croissante et bornée du rapport proie par prédateur (appelé modèle ratio-dépendant, voir Jost et al. (2005) pour plus de détails). Notre jeu de données contient aussi le nombre de prédateurs sur l'Ile Royale, on peut donc tracer ce nuage de points :

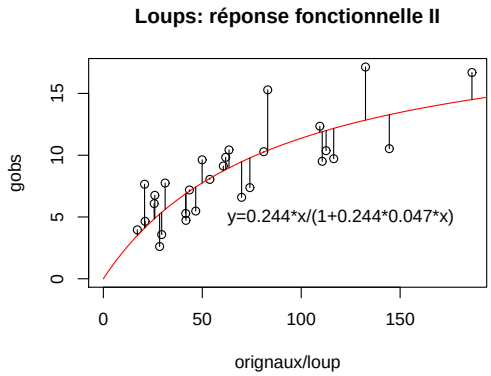

FIGURE 4.2 – La réponse fonctionnelle des loups sur l'Ile Royale en fonction du rapport orignaux par loup. La courbe représente le modèle $y = ax/(1 + ahx)$ ajusté à ces données.

```
R> load("predation.rda"); ls()                    # charger les données predation
R> names(predation); attach(predation)   # afficher les noms des collonnes et les
attacher
R> ratio = proie/pred                    # calculer le rapport proie par prédateur
R> plot(ratio,gobs,main="Loups: réponse fonctionnelle II",   # faire le graphique
+ xlab="orignaux/loup",ylab="gobs",ylim = c(0,max(gobs)), xlim=c(0,max(ratio))) #
```

On voit (Fig 4.2) que le nuage devient effectivement plus dense que dans la Fig 4.1 et qu'il prend la forme d'une courbe croissante et bornée, le modèle ratio-dépendant semble donc être une bonne description de la prédation des orginaux par les loups. Un modèle classique pour modéliser ce type de courbe est

$$gobs = \frac{a\,ratio}{1 + a\,h\,ratio}$$

où a est une sorte de taux de disparition d'orignaux suite a la prédation et h et le temps que prend un loup pour consommer un orignal. Trouvons d'abord les valeurs initiales pour a et h. On note que si `ratio` est petit a est simplement la pente de la relation entre `gobs` et `ratio`, une pente qu'on estime facilement sur le graphique et qui est proche de $a_{initial} = 10/50 = 0.2$. Pour estimer h veuillez notez que le modèle converge/plafonne vers $1/h$ pour des `ratio` très grands. Ce plafond est sur le graphique de l'ordre de 10, on estime donc $h_{init} = 1/10 = 0.1$.

Maintenant on peut appeler la fonction qui ajuste le modèle de façon itérative en minimisant la somme des carrés des résidus, `nls` :

```
R> fitnl = nls(gobs ~ a*ratio/(1+a*h*ratio),     # on donne le modèle non-linéaire
+ start = list(a=0.2, h=0.1))                  # et on donne les valeurs initiales
R> summary(fitnl)                               # donner un résumé de l'ajustement
```

Comme précédemment on s'intéresse surtout aux valeurs estimées,

```
Parameters:
  Estimate Std. Error t value Pr(>|t|)
a 0.244165   0.046566   5.243 1.77e-05 ***
h 0.046960   0.009218   5.095 2.62e-05 ***
```

On a $a = 0.244 \pm 0.047$ et $h = 0.047 \pm 0.009$, tous les deux significativement différents de 0 (le p à la fin des lignes). On peut donc dire qu'un loup prend en moyenne $365 * 0.047 = 17.2$ jours pour manger un orignal (ce qui est compatible avec les observations sur le terrain) et que, s'il n'y avait pas de plafond dans la réponse fonctionnelle, un orignal vit environ $1/0.244 = 4.1$ ans avant de se faire manger (mais le plafond fait que cette durée est en réalité bien plus longue).

Avant de continuer cette analyse il est important de faire le graphique de cette courbe ajustée pour vérifier qu'elle est bien compatible avec le nuage de points,

```
R> plot(ratio,gobs,main="Loups: réponse fonctionnelle II", # refaire le graphique
+ xlab="orignaux/loup",ylab="gobs",ylim = c(0,max(gobs)), xlim=c(0,max(ratio))) #
```

```
R> valRatio=seq(0,200)          # valeurs de x pour lesquelles on trace la courbe
R> lines(valRatio,predict(fitnl,list(ratio=valProie)),col='red')         # tracer
```
Le résultat est la courbe dans la Fig. 4.2, à vue d'oeuil cette courbe semble bien modéliser le nuage de points.

Une remarque importante concerne les erreurs standards : dans une régression non-linéaire ces erreurs standards ne sont que des approximations et il est assez difficile de savoir si l'approximation est bonne. Le mieux est de les vérifier en faisant un petit bootstrap (voir chapitre 2.2). La création des jeux de données bootstrap est un peu plus compliquée parce qu'il faut garder ensemble les couples (x, y), mais à titre d'exemple voici le code **R** pour le faire :

```
R> nombreBS = 300; nombreDonnees = length(gobs);
R> aBS = rep(0,nombreBS); hBS = rep(0,nombreBS);
R> for (k in 1:nombreBS) {
+  valBS = sample(1:nombreDonnees,replace=T);    # les couples (x,y) tiré au sort
+  gobsBS = gobs[valBS];                         # réponse fonctionnelle boostrap
+  ratioBS = ratio[valBS];                                  # ratio boostrap
+  fitnlBS = nls(gobsBS ~ a*ratioBS/(1+a*h*ratioBS),             # ajustement
+  start = list(a=0.2, h=0.1));                                  # bootstrap
+  aBS[k] = coefficients(fitnlBS)[[1]];                  # valeur bootstrap de a
+  hBS[k] = coefficients(fitnlBS)[[2]];                  # valeur bootstrap de h
R> }
R> sd(aBS); sd(hBS);                                  # calcul des erreur standards
```
La vraie erreur standard de a est donc d'environ 0.055 et celle de h d'environ 0.011, l'approximation a seulement légèrement sous-estimé les erreurs standards.

On n'a pas encore fini : il faut vérifier la normalité des résidus et étudier leur auto-corrélation.

```
R> residus = predict(fitnl) - gobs                          # calcul des residus
R> hist(residus); shapiro.test(residus)                    # tester la normalité
R> sortedx = sort(ratio,index.return=T)       # détecter l'ordre des valeurs de x
R> residOrd = residus[sortedx$ix]; acf(residOrd)      # tester l'auto-corrélation
```
On conclut que les résidus sont bien distribués de façon normale ($p = 0.07$) et qu'il n'y a aucune auto-corrélation, l'ajustement n'est donc pas trop mauvais.

Si `nls` produit des erreurs, quoi faire ?

Des fois la fonction `nls` produit des messages d'erreurs et refuse de rendre un résultat. Pour comprendre la raison de ce « refus » il faut regarder la régression linéaire un peu de plus près. La figure 4.3(a) trace la somme des carrés des écarts,

$$sce = \sum_i \left(gobs_i - \frac{a\, ratio_i}{1 + a\, h\, ratio_i} \right)^2 \tag{4.1}$$

en fonction des valeurs des deux paramètres a et h. Si on commence la recherche par exemple à partir du point 1 ($a = 0.6$, $h = 0.045$), comment l'algorithme peut-il trouver les valeurs optimaux ? Comme mentionné au début de cette section, l'algorithme cherche itérativement, essayant à chaque pas de diminuer *sce* (dans la figure 4.3(a), passant du point 1 au point 2 diminue *sce* de 739.1 à 193.0). Comme un algorithme numérique ne peut pas tourner un temps infini il y a toujours un nombre maximal d'itérations pré-programmé. Dans la fonction `nls` ce nombre est de 50 (voir l'aide de `nls`, option `control`), donc si au bout de 50 pas l'algorithme n'a pas encore trouvé l'optimum il s'arrête tout simplement et rend un message d'avertissement qu'il n'a pas encore convergé vers l'optimum. Une première possiblité pour résoudre le problème est donc de permettre plus d'itérations

```
R> fitnl = nls(gobs ~ (a*ratio/(1+a*h*ratio)),      # définir le modèle à ajuster
+  start = list(a=0.6, h=0.045),                  # définir les valeurs initiales
+  control=nls.control(maxiter=100))     # changer le nombre maximal d'itérations
R> fitnl$control                     # afficher les paramètres de control utilisés
```

La progression des pas itératives peut s'observer avec l'option `trace` :

```
R> fitnl = nls(gobs ~ (a*ratio/(1+a*h*ratio)),      # définir le modèle à ajuster
+  start = list(a=0.6, h=0.045),                  # définir les valeurs initiales
```

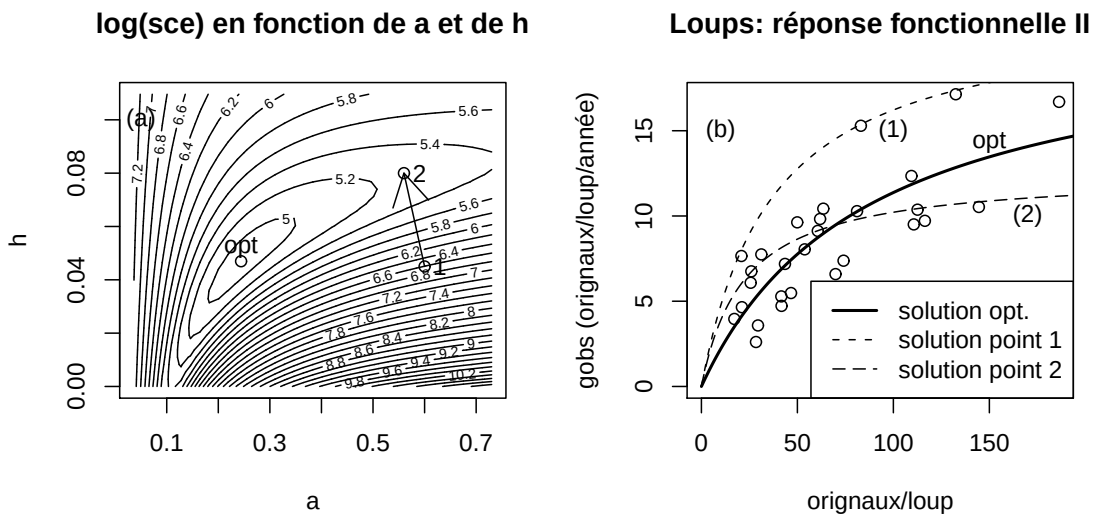

FIGURE 4.3 – Le paysage de la fonction d'erreur (*sce*, en échelle log) et les réponses fonctionnelles pour différentes valeurs des paramètres a et h. Pour le point 1 on a $sce = 739.1$ ($\log(sce) = 6.6$), au point 2 on a $sce = 193.0$ ($\log(sce) = 5.3$) et au point optimale on a $sce = 134.2$ ($\log(sce) = 4.9$).

```
+  trace = T)                                  # activer l'affichage de chaque itération
```

qui affiche à chaque pas la valeur de *sce*, de a et de h. Dans l'exemple il a fallu 6 itérations pour converger, informations que vous trouvez aussi par la commande

```
R> summary(ftinl)
```

Une autre raison pour le « refus » de `nls` peut être due à la méthode comment l'algorithme choisi des nouvelles valeurs de a et de h. La figure 4.3(a) montre, comme une carte topographiques des pyrénées, les isolignes sur lesqeulles *sce* ne change pas de valeur. Le **gradient** de *sce* est une fonction qui donne le vecteur perpendiculaire à ces isolignes et qui pointe en direction de la pointe la plus raide (si vous suivez dans les pyrénées le gradient vous aller monter au sommet le plus proche par le chemin le plus raide). Pour diminuer *sce* il suffirait donc de se déplacer en direction de **-gradient** (la flêche dans la figure 4.3(b)). Cette idée est implémentée dans l'algorithme de Gauss-Newton qu'utilise la fonction *nls*. Le gradient est calculé de façon numérique. Le problème est que ce calcul peut faire une division par 0, donnant un gradient infini qui bloque l'algorithme.

Une alternative est l'algorithme de Nelder-Mead (aussi appelé algorithme de simplex) qui n'a pas besoin du gradient (mais qui nécessite plus d'itérations pour converger, il est donc moins efficace). Cet algorithme est implémenté dans la fonction `optim` qui a besoin de deux arguments : un vecteur avec les valeurs initiales des paramètres, et le nom d'une fonction qui calcule *sce*. Programmons d'abord cette fonction

```
R> calculSCE = function(x) {          # x est le vecteur des valeurs des paramètres
R>  a = x[1];                           # définir que le premier élément de x est a
R>  h = x[2];                            # définir que le second argument de x et h
R>  resid = (gobs-a*ratio/(1+a*h*ratio));                  # calculer les résidus
R>  sce = sum(resid^2);                                          # calculer sce
R>  return(sce)                                        # rendre la valeur de sce
R> }                                                                            #
```

et on vérifie que cette fonction calcule la même valeur de *sce* pour les paramètres optimaux trouvé ci-dessus

```
R> calculSCE(c(0.244167,0.046961))                  # doit donner la valeur 134.2
```

Maintenant nous pouvons utiliser `optim`

```
R> fitSimplex = optim(c(0.6,0.045),calculSCE)                    # appel d'optim
R> fitSimplex                                                 # afficher le résultat
R> fitSimplex$par[1]                                 # extraire la valeur de a ajusté
```

```
R> fitSimplex$par[2]                                    # extraire la valeur de h ajusté
```
L'option `trace` permet également de suivre la progression de cet algorithme itérative
```
R> fitSimplex = optim(c(0.6,0.045),calculSCE,control=list(trace=1))
```
qui, dans ce cas, a eu besoin de 83 itérations (ou, plus précisément, de 83 appels à la fonction `calculSCE`). Cet algorithme est donc effectivement plus lent que `nls`, mais il est plus robuste face à des problèmes purement numérique.

Un inconvénient de la méthode de Nelder-Mead est que sans gradient on ne peut pas calculer les erreurs standards par approximation linéaire (comme pour `nls`). Pour estimer les erreur standards de a et de h il faut donc passer par la méthode du bootstrap (que je vous laisse implémenter vous-même, il suffit de vous inspirer ci-dessus et de ne pas oublier que `calculSCE` utilise les variables globales `gobs` et `ratio` qu'il faut donc changer pour chaque jeu de données bootstrap).

Pour aller plus loin La régression non-linéaire est un vaste champ et peut s'avérer rapidement assez compliquée. Une bonne introduction est donnée dans Press et al. (1992).

4.4 Analyse de survie

A l'origine, l'analyse de survie (ou « survival analysis » en anglais) a été développée dans les sciences médicales pour analyser les durées de survie (ou avant guérison) des personnes affectées par une maladie particulière en fonction des traitements possibles qu'on peut appliquer. Mais ces méthodes peuvent s'appliquer à l'analyse de n'importe quelles données qui sont issues de durées d'évènements, que ce soit la durée de vie d'une protéine avant sa destruction, la durée d'activité d'une substance injectée dans un organisme ou la durée d'un comportement effectué par un insecte. Pour illustrer ces méthodes je me servirai d'un jeu de données issu de cette dernière catégorie, notemment la durée de suivi de bord d'une fourmis *Lasius niger* avant de quitter ce bord. Mais il faut insister que ces méthodes s'appliquent vraiment à toutes les données qui décrivent la durée d'un évènement quelconque.

Disons quelques mots suplémentaires sur ce jeu de données. Les fourmis ont une tendance générale de suivre les hétérogénités dans l'environnement, et des bords de tous types font partie de ces hétérogénites. Mais le fait de quitter ces bords de temps à autre permet d'explorer d'autres parties de l'environnement. On peut ensuite se poser la question des lois comportementales sous-jacentes à ce phénomène, et l'analyse de survie nous permettra d'émettre certaines hypothèses sur ces lois. Nous étudierons en particulier l'effet de la température ambiante et de la courbure du bord sur le temps de suivre ce bord.

Qu'est-ce qu'une courbe de survie

Regardons d'abord les durées qu'une fourmi suit un bord droit avant de quitter ce bord, et ceci à une température de 23 °C.
```
R> dat = read.table("thigmotaxis.txt",h=T,dec=",")              # lire les données
R> courb0t23 = subset(dat,courb==0 & temp==23)          # bords droits et temp 23 C
R> hist(courb0t23$duree)                                  # histogramme des durées
```
Cet histogramme nous montre qu'ils y a beacoups de durées très courtes et peu de durées longues. La distribution des durées est donc très différentes des distributions normales si chères aux statistiques paramétriques. Mais l'analyse de survie est justement adaptée à ce type de distribution. Regardons de plus près ces données en forme de courbe de survie (que nous avions déjà rencontrée sur la page 6). Supposons que toutes les durées commencent au même moment - la courbe de survie représente la fraction d'évènements qui durent encore en fonction du temps passé. Dans **R** il y a une bibliothèque spéciale pour tracer ce types de courbes avec son intervalle de confiance.
```
R> library(survival)                              # charger la bibliothèque 'survival'
R> cs1 = survfit(Surv(duree) ~ 1, data = courb0t23)          # calcul de la courbe
R> plot(cs1, main="Courbe de survie des suivis de bord",xlab="Temps (s)")
```
Dans ce graphique la courbe solide reprèsente la courbe de survie et les lignes en trait l'intervalle de confiance autour de cette courbe. La courbe de survie commence à une fraction de 1 (aucun évènement ne s'est encore arrêté) et diminue de façon monotone vers une fraction de 0 (tous les évènements sont terminés). La forme de cette courbe rappelle une exponentielle à coéfficient négative, ce qui est confirmé quand on la trace avec l'ordonnée en échelle logarithmique :

```
R> plot(cs1, main="Courbe de survie des suivis de bord",
+ xlab="Temps (s)", log="y")                    # axe y en échelle logarithmique
```
En échelle log-linéaire la courbe devient quasiment une droite (et l'intervalle de confiance indique qu'une droite est parfaitement compatible avec cette courbe si on prend en compte l'incertitude sur son estimation). Ceci indique que le taux de quitter le bord (c'est-à-dire la probabilité par unité de temps) est constant au cours du temps : a chaque instant, la fourmis a la même probabilité de quitter le bord durant le pas de temps suivant, et ceci indépendemment de la durée total du suivi de bord déjà effectué (on parle d'un processus sans mémoire).

Tester l'effet d'un facteur externe sur les durées d'évènements

L'analyse de survie permet de regarder si un paramètre externe (tel que la température) influence la durée d'un évènement. Regardons cela de plus près pour les durées de suivi de bord à trois températures : 16°C, 23°C et 30°C. Traçons d'abord les trois courbes de survie :

```
R> courb0 = subset(dat,courb==0)                # tous les suivis de bord droit
R> cs2 = survfit(Surv(duree) ~ temp,data=courb0)          # calcul des 3 courbes
R> plot(cs2,lty=c("solid","dashed","dotted"), log="y",xlab="Temps (s)")
R> legend('topright',lty=c("solid","dashed","dotted"),
+ legend = c("16 C","23 C","30 C"))                        # ajout d'une légende
```
Pour faciliter la lisibilité **R** n'ajoute pas les trois intervalles de confiances, mais on identifie rapidement la tendance que plus la température est basse, plus les suivis de bord durent longtemps. Est-ce que c'est un effet significative ?

```
R> survdiff(Surv(duree) ~ temp, data=courb0)               # effet température?
```
R teste cet effet par un χ^2, et le résultat est nette : $\chi^2 = 28.2$, $dl = 2$ et $p < 0.0017$. La température a donc un effet hautement significative sur la durée de suivi de bord.

Un peu de biologie des insectes

Cependant, connaissant un peu la biologie des insectes, est-ce qu'on a mesuré la bonne chose ? La température influence surtout la vitesse des fourmis (elle augmente avec la température) et on sait d'ailleurs que les fourmis mesurent des distances par le nombre de pas qu'elles doivent effectuer. Au lieu de regarder les durées de suivi de bord on devrait peut-être plutôt regarder la distance de suivi de bord en multipliant les durées par les vitesses propres à chaque température.

```
R> v23 = 1.4;v16 = 0.80;v30 = 1.50;                          # vitesses en cm/s
R> l16 = courb0$duree[courb0$temp==16] * v16;      # tranformer les durées à 16 C
R> l23 = courb0$duree[courb0$temp==23] * v23;
R> l30 = courb0$duree[courb0$temp==30] * v30;
R> l = c(l16,l23,l30);                           # les regrouper dans un vecteur
R> survdiff(Surv(l) ~ courb0$temp)                  # tester l'effet température
```
Le résultat est nette, la température n'a aucune influence sur la distance du suivi de bord ($\chi^2 = 0.3$, $dl = 2$, $p = 0.857$). Ceci suggère que la probabilité par pas de fourmi de quitter le bord est la même pour toutes les températures testées.

Le modèle de Cox : une méthode très puissante

Indépendemment de la distribution particulière de la courbe de survie (exponentielle ou autre), le modèle de Cox permet de tester les effets de divers facteurs sur cette courbe (similaires aux méthodes d'ANOVA à plusieurs facteurs). A titre d'exemple on va regarder l'effet combiné de température, courbure (inverse du radius d'un cercle décrivant la courbe) et colonie de fourmis sur la durée de suivi de bord.

```
R> dat = read.table("thigmotaxis.txt",h=T,dec=",")          # (re)lire les données
R> modCox = coxph(Surv(duree) ~ courb + temp + col, data=dat)
R> summary(modCox)                                    # appliquer le modèle de Cox
```
Le résultat brut est

```
Call: coxph(formula = Surv(duree) ~ courb + temp + col, data = dat)
  n= 1017
           coef exp(coef) se(coef)     z     p
courb     1.9738     7.198  0.09781 20.18 0.000
```

```
temp      0.0704     1.073  0.00574 12.28 0.000
colverte -0.1258     0.882  0.06385 -1.97 0.049
```

On a donc un effet significative de la courbure ($p < 0.001$), de la température ($p < 0.001$) et même un effet tout juste signficative de la colonie ($p = 0.049$). Ce dernier effet pourrait venir du fait que les deux colonies n'avaient pas le même age, mais cela vaudrait le coup de faire des expériences supplémentaires pour clarifier l'étendu de cet effet colonie.

Enfin, rappelons-nous que les courbes de survie peuvent être décrites par une exponentielle, $cs(t) = \exp(-\lambda t)$. L'analyse de survie nous permet aussi de modéliser la façon comment λ dépend de la température T et de la courbure c, $\lambda(T, c)$, par la régression suivante :

```
R> survreg(Surv(duree) ~ courb + temp, data=dat,dist="exponential")            #
```

On obtient ainsi la paramétrisation

$$\lambda(T, c) = -3.46 - 1.21c - 0.045T$$

qui pourrait être utilisé dans un modèle individu centré de déplacement de fourmis dans un milieu hétérogène (à préciser dans ce cas que la température est mesurée en °C et la courbure en cm^{-1}).

Annexe A

Quelques aides pour se servir de R

A.1 Quelques mots sur R

R (Team, 2003) est un graticiel du type « open source », c'est-à-dire il est entièrement gratuit et tout le monde peut télécharger les sources du code à partir duquel il a été créé. Il y a une communauté dans le monde entier qui l'utilise et qui continue à développer de nouvelles fonctionnalités (gratuitement). Pour honorer ce travail le mieux est de mentionner dans vos rapports que vous avez fait vos calculs avec **R**, voir

```
R> citation()
```

pour trouver la bonne façon de le citer. Pour télécharger une version pour votre ordinateur, rendez-vous sur

`http://www.r-project.org`

R est en fait basé sur un langage nommé S qui a été défini par J. Chambers dans les laboratoires de Bell. Ce langage a d'abord été implémenté dans le logiciel commercial S-plus qui fonctionne avec exactement les mêmes commandes que **R** mais offre en plus une interface graphique (par exemple un tableur pour saisir les données). Donc, tout ce que vous apprenez pour **R** sera utilisable dans S-plus. Il faut aussi ajouter que la façon de structurer les données dans **R** est un standard international pour tous les logiciels statistiques, c'est-à-dire la philosophie que vous acquérez en utilisant **R** vous servira aussi pour comprendre comment faire des analyses (et l'interprétation) dans un logiciel comme Systat ou SPSS.

Pour plus d'informations sur les idées du « open source », voir

`http://www.gnu.org/`

A.2 Les bibliothèques et les fichiers accompagnant ce document

Sur le site web de l'auteur,

`http://cognition.ups-tlse.fr/vas-y.php?id=chj`

vous pouvez trouver un fichier compressé qui contient tous les jeux de données utilisés dans ce poly (format `tgz` pour linux, format `sit` pour Mac, format `zip` pour PC). Téléchargez ce fichier sur votre ordinateur et décompactez-le. Il faut ensuite indiquer à **R** où se trouvent les fichiers décompactés (menu "Change working directory" ou quelques chose comme cela).

Il y a trois types de fichiers

- `*.R` Ce sont des fichiers avec des commandes **R** que vous pouvez ouvrir dans l'éditeur de script et copier/coller dans **R** ou directement éxécuter avec la commande `source(...)`.
- `*.txt` Ce sont des fichiers de données en format text que vous pouvez également ouvrir dans l'éditeur de script pour regarder ce qu'il y a dedans mais qu'il faut lire avec la commande `read.table(...)`.
- `*.rda` ou `*.RData` Ce sont des fichiers sous format binaire qui contiennent des jeux de données et que vous ne pouvez lire et charger qu'avec la commande `load(...)`.

Il existe aussi pleines de fonctions supplémentaires pour **R** qui se trouvent dans ce q'on appelle des « bibliothèques ». Ces bibliothèques se chargent avec la commande `library`, mais si, par exemple, la commande

```
R> library(energy)
```
vous donne un message d'erreur il faut encore l'installer via internet. Le plus simple est de passer par le menu `Packages-Install packages` qui vous laisse d'abord choisir un serveur sur internet et qui vous donne ensuite la liste complète de bibliothèques que vous pouvez installer. Choisissez `energy` et tout devrait s'installer de façon automatique.

A.3 Organiser, saisir, enregistrer et (re)lire vos données dans R

Comment préparer vos données pour les analyser dans **R** ? Si **R** ne comprend pas ce que vous lui demandez c'est d'habitude parce que les données n'ont pas le bon format. La documentation n'est souvent pas très 'helpful' non plus, c'est un peu le problème général d'un logiciel libre. Il faut donc être un peu joueur et tâtonner, ou demander à un collègue qui s'y connaît mieux. Mais ci-dessous quelques conseils qui vous donneront un peu plus d'autonomie.

Organiser ses données

R reconnaît quatre structures de données, les **vecteurs**, les **matrices**, les **data.frame** et les **list**. Regardons-les une par une.

Un **vecteur** est simplement une liste d'éléments (des nombres, des caractères, des mots). On le crée par la commande `c(...)` :

```
R> x=c(1.3, 1.5, 3.3,4.5)
R> noms = c("Maud","Christian","Alex")
```

et on peut se servir des éléments avec des crochets

```
R> x[3]                                          # le troisième élément de x
R> noms[2]                                     # le deuxième nom dans ma liste
```

Beaucoup de fonctions demandent comme argument simplement un vecteur, par exemple

```
R> mean(x)                                  # calcule la moyennes des valeurs dans x
```

Attention : certains caractères ont une signification bien précise, par exemple si vous mettez le sexe des individus dans le vecteur 'sexe'

```
R> sexe = c(F, F, F, F)
```

ce n'est pas un vecteur de caractères mais de variables logiques (F=FALSE). Pour avoir un vecteur avec les lettres F il faut taper

```
R> sexe = c("F","F","F","F")
```

Pour connaître la taille d'un vecteur il y a la commande

```
R> length(sexe)
```

Une **matrice** est un tableau d'éléments du même type (nombre, caractères, etc.). Pour désigner la taille de ce tableau on donne toujours d'abord le nombre de lignes et ensuite le nombre de colonnes. On peut les créer en collant ensemble plusieurs vecteurs de même longeur

```
R> mat4fois2 = cbind(x,c(4.3,4.4,3.1,2.9))                # cbind = 'column bind'
R> mat2fois4 = rbind(x,c(4.3,4.4,3.1,2.9))                   # rbind = 'row bind'
```

Et on s'adresse aux éléments en indiquant la ligne (row) et la colonne (column)

```
R> mat4fois2[4,2]                      # élément de la 4ème ligne et 2ème colonne
R> mat2fois4[2,3]                      # élément de la 2ème ligne et 3ème colonne
```

Attention, il ne faut pas confondre lignes et colonnes,

```
R> mat4fois2[2,4]                                                  # n'existe pas
```

Si on veut créer une matrice remplie de la même valeur on peut utiliser la commande

```
R> zeros2fois3 = matrix(0,nrow=2,ncol=3)            # une matrice contenant des 0
```

Pour connaître les dimensions (nombre de lignes et de colonnes) d'une matrice il y a la commande

```
R> dim(mat4fois2)      # donne un vecteur avec le nombre de lignes et de colonnes
```

Un **data.frame** est une matrice un peu plus sophistiquée dont chaque colonne peut être d'un type différent (nombres, caractères, etc.). En plus, on peut donner des noms à chaque colonne et à chaque ligne. L'idée générale est que chaque ligne contient les informations concernant un élément d'étude, par exemple la taille du nez, le sexe et le QI d'une personne

```
R> monJeuDeDon = data.frame(tailleNez=x, sexe=sexe, QI=c(91,101,150,95))
```

```
R> names(monJeuDeDon)                                    # les noms des colonnes
R> monJeuDeDon$sexe                                          # colonne du sexe
R> monJeuDeDon$QI[2]; monJeuDeDon[[3]][2];  # un data.frame n'est pas une matrice
R> plot(monJeuDeDon$QI ~ monJeuDeDon$tailleNez)  # relation entre QI et tailleNez
R> dim(monJeuDeDon)                             # nombre de lignes et de colonnes
```

Finalement, le « fourre-tout » dans **R** est la **list** qui peut regrouper des éléments de n'importe quels type et taille.

```
R> monFourreTout = list(nom="Jean",monVecteur=c(35,88,97), maMatrice=mat4fois2)
R> names(monFourreTout)                                  # les noms des éléments
R> monFourreTout$monVecteur                                # extraire le vecteur
R> monFourreTout$maMatrice[3,1]                          # un élément de la matrice
```

Souvent les résultats d'une analyse statistique sont rendus sous forme d'une liste dont on peut extraire et utiliser les différents éléments :

```
R> resultat = hist(c(1.3,1.5,3.3,4.5,5.5,1.3,2.2,2.3,3.1))
R> names(resultat)                                 # les éléments rendus par 'hist'
R> resultat$breaks                          # les limites des catégories de valeurs
R> resultat$counts                   # le nombre de valeurs dans chaque catégorie
```

On peut toujours interroger **R** pour apprendre le type de structure d'un objet

```
R> is.data.frame(monFourreTout)                               # test si data.frame
R> is.list(monFourreTout)                                        # test si list
R> is.vector(sexe)                                            # test si vecteur
R> is.matrix(monJeuDeDon)                                     # test si matrice
```

et on peut aussi transformer un objet en un autre (si c'est logiquement possible) :

```
R> mdf=as.data.frame(mat4fois2)            # tranformer une matrice en data.frame
R> names(mdf) = c("x","y"); mdf                    # changer les noms des colonnes
```

Enregistrer vos données dans un fichier

Supposons que vous êtes en train de travailler sur un rapport et que votre collegue doit continuer l'analyse de vos données : comment les lui faire parvenir ? La solution la plus simple est de sauver les variables dont vous avez besoin dans un fichier propre à **R**,

```
R> save(monFourreTout,monJeuDeDon,sexe,file="mesdonnees.rda")
```

que vous envoyez par email à votre collègue ou que vous recopiez comme n'importe quel fichier sur une disquette ou une clé USB (que faire si vous ne savez pas ou **R** a enregistré ce fichier ? La commande `getwd()` (= get working directory) vous le dira). Votre collègue peut maintenant recharger ces données avec la commande

```
R> load("mesdonnees.rda")
```

après avoir indiqué à **R** dans quel répertoire il se trouve. Veuillez noter que la commande `save` crée un fichier codé en binaire que seulement **R** peut décoder avec la commande `load`. Si vous ouvrez ce fichier dans un éditeur de texte tel que BlocNotes vous n'y comprendrez rien de ce qui s'affichera.

Les données du type `vector`, `matrix` ou `data.frame` peuvent aussi être enregistrées dans un fichier du type text que vous pouvez ensuite ouvrir dans d'autres logiciels tel qu'Excel ou BlocNotes. La commande est

```
R> write.table(monJeuDeDon, file="monFichier.txt", quote=F, row.names=F)
```

Voir

```
R> help(write.table)
```

pour plus d'informations sur les options (par exemple, `dec=","` fera des virgules décimales au lieu des points décimales, ce qui est probablement le format des chiffres sur votre PC).

Préparation des données pour une ANOVA à 1 facteur

Prenons l'exemple de l'engrais (4 types d'engrais, on mesure le rendement sur plusieurs répliques). Vos données sont probablement saisies dans un tableur (genre Excel) avec une colonne contenant les rendements pour un type d'engrais, donc 4 colonnes :

```
eng1    eng2    eng3    eng4
45      31      32      39
42,5    34      33      38
39      40      38      41
41              33      44
48
```

et **R** en a besoin sous la forme de deux vecteurs, le premier (rend) contenant tous les rendement enchaînés,
`R> rend=c(45, 42.5, 39, 41, 48, 31, 34, 40, 32, 33, 38, 33, 39, 38, 41, 44)`
(veuillez noter que la virgule décimale a été saisi en point décimal comme c'est habituel dans des logiciels internationals) et le second (facteur) contenant les facteurs associés,
`R> facteur = c(1, 1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4)`
pour faire une ANOVA avec la commande
`R> summary(aov(rend ~ as.factor(facteur)))`
Attention, `as.factor` est nécessaire à cause de l'encodage de `facteur` en chiffres qui peuvent inciter **R** à faire une régression linéaire plutôt qu'une ANOVA, voir
`R> summary(aov(rend ~ facteur))`
(on s'aperçoit que c'est le résultat d'une régression linéaire à cause du degrée de liberté de `facteur` qui est 1 au lieu de 3 ci-dessus ;, le $p = 0.2904$ affiché teste l'hypothèse nulle que la pente est égale à 0). Pour éviter ce type de confusion il est toujours prudent de coder les différentes modalités d'un facteur en commençant avec des lettres et en les déclarant explicitement comme étant du type `factor`,

```
R> facteur = factor(c('e1','e1','e1','e1','e1','e2','e2','e2',
+ 'e3','e3','e3','e3','e4','e4','e4','e4'))                    #
R> summary(aov(rend ~ facteur))
```

Dans l'exemple ci-dessus il n'y a avait que 16 mesures à saisir, le plus vite est donc de les saisir directement dans **R** comme on vient de le faire. Mais si vous avez des centaines, voir des milliers de mesures il est évident que les (re-)saisir à la main est inefficace. Une alternative est de préparer les données sous forme d'un tableau dans un tableur. Dans ce tableau on commence pour chaque mesure une ligne qu'on complète on y ajoutant les valeurs du ou des facteurs associés à cette mesure. Par exemple, la première ligne serait
`45 e1`
pour indiquer que la mesure 45 correspond au rendement d'un champ avec l'engrais du type 1. Ce tableau peut ensuite être enregistré sous format text et lu dans **R**.

Préparer les données dans un tableur

1. En copiant/collant dans votre tableur créer le tableau suivant :

   ```
   rend fact
   45   e1
   42,5 e1
   39   e1
   41   e1
   48   e1
   31   e2
   34   e2
   40   e2
   32   e3
   33   e3
   38   e3
   33   e3
   39   e4
   38   e4
   41   e4
   44   e4
   ```

   que vous enregistrez ensuite en format 'text' en choisissant comme séparateur entre chiffres le tabulateur. Disons que votre fichier s'appelle `engraisR.txt`.

2. dans **R** lisez ces données avec la commande
   ```
   R> don = read.table("engraisR.txt",header=T, dec=",")
   R> don
   R> names(don)              # vérifier que les noms des colonnes sont bien lus
   ```
   l'option `header=T` indique que la première ligne contient les noms des colonnes (header = en-tête) et l'option `dec=","` vous permet d'indiquer que les décimales sont séparées par une virgule (ce qui est le cas si vous avez une version française de votre tabulateur),
3. analysez avec la commande
   ```
   R> summary(aov(rend ~ fact, data = don))
   ```

Pour une ANOVA à 2 facteurs il suffit d'ajouter dans votre tableur une colonne au nom de `fact2` contenant les modalités du deuxième facteur associés à chaque mesure. Vous faites ensuite l'analyse comme décrit dans les chapitres précédents. Veuillez noter que la commande `read.table` ne fonctionne que si chaque ligne contient le même nombre d'éléments (c'est-à-dire que vos données sont organisé dans un tableau complet). S'il y a des valeurs manquantes il faut remplir ces trous par 'NA' (= not available), **R** saura ensuite quoi faire. Vueillez également noter que le tableau rendu par `read.table` est du type `data.frame`
```
R> is.data.frame(don)
```

Préparer les données pour une ANOVA à mesures répétées

Supposons que vous ayez saisi les données (de la page 27) dans votre tableur sous la forme suivante

traitement	s1	s2	s3	s4
NaCl	65	30	19	28
NaCl	143	68	53	21
NaCl	61	69	59	50
NaCl	139	79	32	22
NaCl	41	56	30	48
NaCl	119	152	79	20
NaCl	83	72	20	24
AP5	75	86	75	100
AP5	82	65	49	75
AP5	45	58	58	65
AP5	59	45	89	98
AP5	78	100	73	65
AP5	65	89	76	67
AP5	89	75	45	78
AP5	45	72	26	56
AP5	74	65	98	52

où chaque ligne correspond aux mesures prises sur une souris (temps d'exploration) lors des séances d'entrainement `s1` à `s4`. Mais **R** en a besoin dans la forme classique d'une colonne de mesures et de trois colonnes pour les facteurs traitement, séance et sujet

temps	traitement	seance	individu
65	NaCl	s1	i1
30	NaCl	s2	i1
...	...	...	...
75	AP5	s1	i8
86	AP5	s2	i8
...	...	...	...
52	AP5	s4	i16

Vous avez toujours la possiblité de créer ce format directement dans votre tableur et d'enregistrer le tableau ensuite avec la commande `read.table`. Mais vous avez aussi la possibilité de préparer ce tableau directement dans **R** à partir du premier tableau. Regardons cette possibilité plus en détail.

1. Enregistrez votre tableau original sous format 'text' en choisissant comme séparateur entre chiffres le tabulateur. Disons que votre fichier s'appelle `sourisA.txt`.
2. Lire ce fichier dans **R** avec la commande
```
R> don = read.table("sourisA.txt",header=T)
R> don
R> names(don)              # vérifier que les noms des colonnes sont bien lus
```
3. créer les vecteurs `trmt`, `sujet`, `seance` et `explore` avec les commandes
```
R> trmt = as.factor(rep(c(rep("NaCl",7),rep("AP5",9)),4)); trmt
R> sujet = as.factor(rep(seq(1,16,1),4))
R> seance = as.factor(c(rep("s1",16),rep("s2",16),rep("s3",16),
+ (rep("s4",16))));                                          # rep = répéter
R> explore = c(don$s1, don$s2, don$s3, don$s4); explore
```

Attention, si vous utilisez pour les facteurs des noms plus « français », ne mettez **pas** d'accents : **R** ne les interprète pas correctement et vous affichera des erreurs.

Et, bien sûr, vous pouvez les organiser dans un `data.frame` par la commande suivante :
`R> sourisA=data.frame(trmt=trmt,sujet=sujet,seance=seance,explore=explore)`
(remarque : dans `trmt=trmt`, le premier indique le nom de la colonne dans `sourisA` et le second le nom du vecteur contenant les données. Si vous voulez changer les noms des colonnes dans `sourisA` il faut changer le premier, `trait = trmt`.

A.4 Enregistrer et utiliser les graphiques de R

Pour utiliser un graphique de **R** dans vos rapports vous pouvez tout simplement faire un copier dans la fenêtre graphique de **R** et un coller dans le traitement de texte de votre choix (par exemple OpenOffice ou, si vous insistez sur l'utilisation de produits Micro$oft, Word). Les versions PC ou Mac de **R** permettent aussi de sélectionner la fenêtre avec le graphique et de l'enregistrer via les menus sous différents formats, le meilleur étant d'habitude 'png' qui comprime sans perte d'information.

Si vous ne pouvez pas le faire par les menus la ligne de commande offre plusieurs formats graphiques. Le plus classique et de les enregistrer sous forme vectorielle dans un fichier postscript avec la commande
`R> dev.copy2eps(filename="monFichier.eps")                                    #`
mais postscript est un peu difficile à utiliser sur un PC (c'est le langage universel des imprimantes et de linux). Des formats plus simples sont `jpeg` et `png` (le dernier très utile pour des graphismes statistiques parce qu'il comprime toute l'information sans perte). La création de ces fichiers est un peu plus compliquée parce que vous devez d'abord ouvrir le fichier destinataire, ensuite refaire toutes les commandes graphiques, et finalement fermer le fichier graphique
```
R> jpeg(filename="monFichier.jpg")                    # commence l'enregistrement
R> barplot(c(35,10,95),names.arg = c("N","B","F"),xlab="caste",
+ ylab="fréquence absolue",col=c("white","grey","black"))                      #
R> dev.off()                                        # fermer le fichier graphique
```
(en fait, `dev` vient du mot anglais "device" qui désigne dans ce contexte le destinataire des commandes graphiques : une fenêtre sur l'écran, un fichier, l'imprimante, ...). Pour faire un `png` c'est la commande `png(filename=...)`.

Pour imprimer directement une fenêtre graphique vous pouvez utiliser la commande
`R> dev.print()`

Références bibliographiques

Aiken, L. S., & S. G. West. 1991. Multiple regression : testing and interpreting interactions. Sage Publications, London.

Boos, D. D., & J. M. Hughes-Oliver. 2000. How large does n have to be for Z and t intervals ? *The American Statistician* 54 :121–128.

Cohen, J. 1988. Statistical power analysis for the behavioral sciences. Erlbaum, Hillsdale, 2nd edn.

Conradt, L. 1998. Measuring the degree of sexual segregation in group-living animals. *Journal of Animal Ecology* 67 :217–226.

Crawley, M. J. 2005. Statistics : an introduction using R. John Wiley & Sons, New York.

Davison, A. C., & D. Hinkley, V. 1997. Bootstrap methods and their applications. Cambridge University Press, London.

Dixon, P. M. 1994. Testing spatial segregation using a nearest-neighbor contingency table. *Ecology* 75 :1940–1948.

Efron, B., & R. J. Tibshirani. 1993. An Introduction to the Bootstrap. Chapman and Hall, London, New York.

Engqvist, L. 2005. The mistreatment of covariate interaction terms in linear model analyses of behavioural and evolutionary ecology studies. *Animal Behaviour* 70 :967–871.

García, S., & F. Hererra. 2008. An extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all pairwise comparisons. *Journal of Machine Learning Research* 9 :2677–2694.

Giurfa, M. 2004. Conditioning procedure and color discrimination in the honeybee *Apis mellifera*. *Naturwissenschaften* 91 :228–231.

Haccou, P., & E. Meelis. 1992. Statistical analysis of behavioural data : an approach based on time structured models. Oxford University Press.

Jost, C., G. Devulder, J. A. Vucetich, R. O. Peterson, & R. Arditi. 2005. The wolves of Isle Royale display scale-invariant satiation and ratio-dependent predation on moose. *Journal of Animal Ecology* 74 :809–816.

Lunney, G. H. 1970. Using analysis of variance with a dichotomous dependent variable : an empirical study. *Journal of educational measurement* 7 :263–269.

Møller, A. P., & M. D. Jennions. 2002. How much variance can be explained by ecologists and evolutionary biologists. *Oecologia* 132 :492–500.

Nakagawa, S., & T. M. Foster. 2004. The case against retrospective statistical power analyses with an introduction to power analysis. *Acta ethologica* 7 :103–108.

Potvin, C., M. J. Lechowicz, & S. Tardif. 1990. The statistical anlaysis of ecophysiological response curves obtained from experiments involving repeated measures. *Ecology* 71 :1389–1400.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, & B. P. Flannery. 1992. Numerical Recipes in C : The Art of Scientific Computing. Cambridge University Press, London, 2nd edn.

Rice, W. R. 1989. Analyzing tables of statistical tests. *Evolution* 43 :223–225.

Ripley, B. D. 1988. Statistical inference for spatial processes. Cambridge University Press, London.

Roces, F., & C. Kleineidam. 2000. Humidity preference for fungus culturing by workers of the leaf-cutting ant *Atta sexdens rubropilosa*. *Insectes Sociaux* 47 :348–350.

Scherrer, B. 1984. Biostatistique. Gaëtan Morin éditeur.

Team, R. D. C. 2003. R : A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-00-3.

Zar, J. H. 1999. Biostatistical analysis. Prentice Hall, New Jersey, 4th edn.

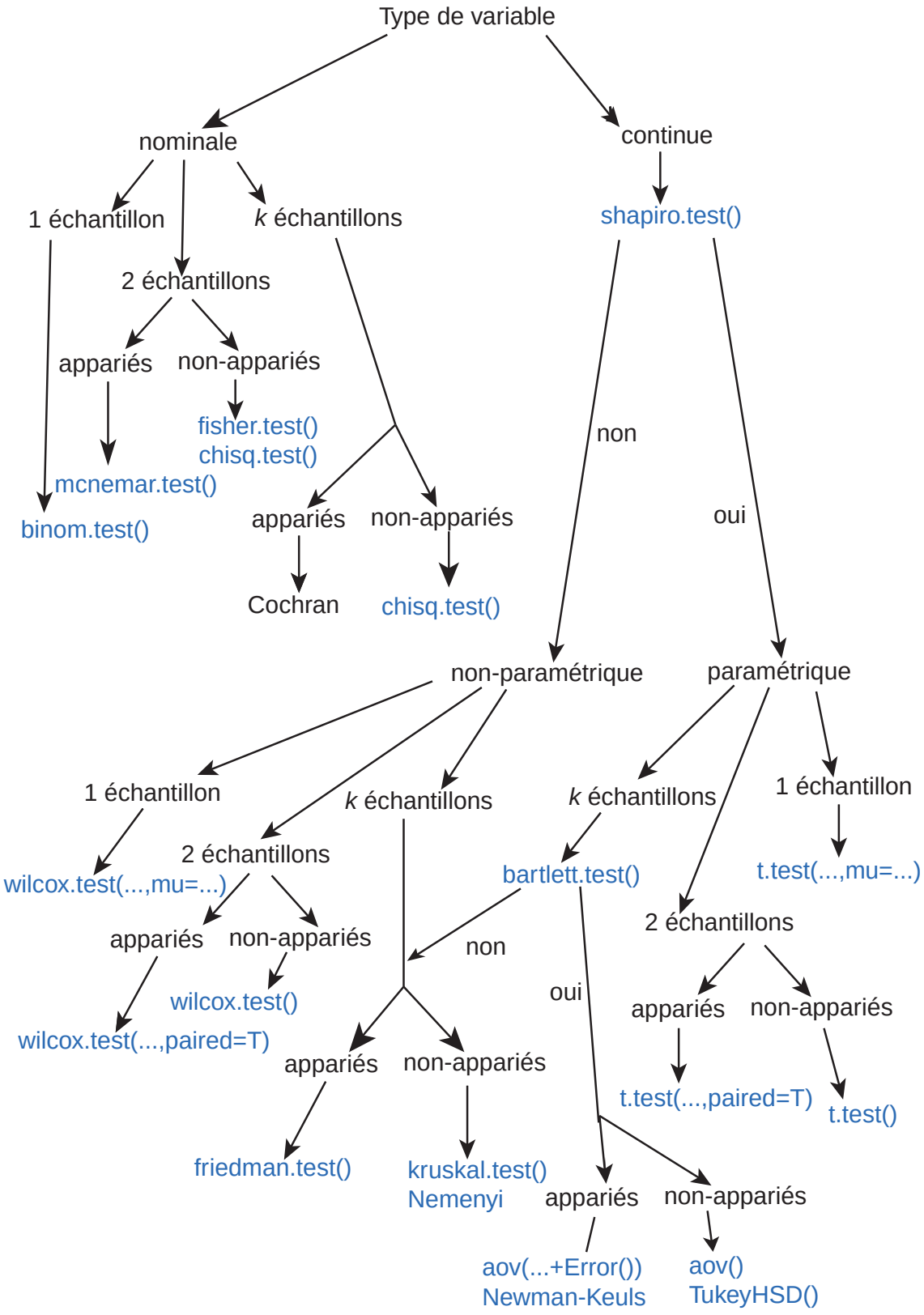

FIGURE A.1 – Résumé des tests d'hypothèses selon le type de données et leurs propriétés.

Index