ATTENTION, version « 0.0 » brute de décoffrage

donc particulièrement susceptible de comporter des erreurs.

Si vous en trouvez, soyez sympa et prévenez l'auteur :

denis.poinsot@univ-rennes1.fr

R

pour les statophobes

Utilisation du logiciel statistique R pour réaliser les analyses statistiques de base, à l'attention des étudiants allergiques aux statistiques en général et aux logiciels en particulier

Denis Poinsot

Avant propos

Lorsque j'ai rédigé avec fièvre la première version de *Statistiques pour Statophobes* en 1998, il s'agissait de fournir un support autonome de « cours » complétant des séances de travaux dirigés effectuées dans des salles banalisées (sans ordinateurs). Tous les calculs devaient donc (et peuvent toujours) se faire au moyen d'une simple calculatrice. Il existait cependant de ce fait un décalage regrettable entre les exercices présentés et la manière dont les étudiant traitent leurs données réelles lors de la rédaction d'un rapport de recherche. Dans ce dernier cas, en effet, un logiciel était systématiquement utilisé. Je dis « un » mais je devrais dire « des tas », et c'était justement un problème décourageant : fallait-il leur apprendre à se servir de Minitab™ ? SAS™ Statistica™ ? Statview™ ? Et que feraient ils en stage si la structure d'accueil n'avait pas le logiciel hors de prix qu'ils avaient (péniblement) appris à utiliser ?

C'est alors que des collègues bien plus forts que moi en stats[1] m'ont fait découvrir d'une part l'existence et d'autre part la puissance du logiciel gratuit R, disponible en ligne et utilisé par un nombre croissant de chercheurs dans le monde. Passé le premier moment de recul (R a la réputation — non méritée — d'être difficile d'accès) j'y ai trempé un orteil, puis l'autre, puis j'ai été tellement convaincu que j'ai définitivement effacé tous les autres logiciels de stats qui encombraient mon ordinateur.

Tenant enfin la perle rare (puissant, gratuit, téléchargeable partout, flexible), il ne restait plus qu'à convaincre les étudiants encore-plus-réfractaires-que-moi à s'y mettre, d'où le présent document. Il complète (mais ne remplace pas) le précédent. Là ou *Statistiques pour Statophobes* explique longuement (peut être même trop !) et pas à pas les notions statistiques, les précautions à prendre, les erreurs à ne pas commettre, pour que vous *compreniez* ce que vous faites, le présent document se contente de "montrer comment on fait des tests avec R ", Utilisés *ensemble*, ces deux ouvrages devraient donc j'espère vous permettre non seulement de "faire des tests" mais de *comprendre ce que vous faites* quand vous utilisez les méthodes statistiques de base.

D.P.

6 novembre 2005

[1] Carlos Bernstein, Jean-Sébastien Pierre et Eric Wajnberg, en particulier

Sommaire

1.. R, céquoidonc ?

R est un logiciel d'analyse statistique extrêmement puissant mais pourtant **gratuit, régulièrement mis à jour**, et dont la version en cours peut être obtenue sur le web en libre accès (`http://cran.cict.fr/`). Il est à SAS et autres Statistica ce que Linux est à Windows : une solution de rechange non seulement crédible mais professionnelle et dont la disponibilité est garantie à long terme. En effet, R est largement utilisé dans le monde de la recherche scientifique au plus haut niveau et repose sur un très solide réseau mondial d'informaticiens et de statisticiens bénévoles regroupés dans le *R core team*. Ces bienfaiteurs de l'humanité statistique contribuent chaque jour à l'ajout de nouvelles fonctions (chaque utilisateur de R est par ailleurs invité à contribuer à l'oeuvre commune s'il a mis au point une nouvelle procédure, qui ira enrichir la vaste bibliothèque existante). R permet ainsi d'effectuer des analyses statistiques d'une complexité ébouriffante sur les sujets les plus divers, et ses possibilités graphiques sont virtuellement infinies.

Etant un utilisateur novice (je m'y suis plongé, avec circonspection, en 2004, sur les conseils de collègues très expérimentés), je me contenterai ici de vous décrire par des exemples concrets les quelques fonctions statistiques de base que je sais utiliser et qui permettent d'effectuer les tests les plus courants. Ces fonctions font partie du module nommé justement "base" et qui comme son nom l'indique constitue le module le plus basique de R. Le module "base" est chargé en mémoire automatiquement chaque fois que vous ouvrez R, donc vous n'avez à vous préoccuper de rien au delà du simple double-clic sur l'icône R lorsque vous aurez installé le logiciel après l'avoir téléchargé (c'est très facile, la preuve étant que j'y suis arrivé). Après avoir double-cliqué pour ouvrir R, vous vous retrouvez devant une demi page de baratin qui se termine par le *prompt* (le signe *« supérieur à ») :*

```
>
```

Eh oui, pas le moindre menu déroulant à se mettre sous la souris (en tout cas pas question de lancer une analyse statistique en utilisant ceux que vous verrez dans la barre des tâches), il va vous falloir entrer vos instructions "à l'ancienne", ligne par ligne, respectant ainsi la fière devise (un brin macho) des informaticien purs et durs, utilisateurs d'Unix :

« *Real Men Don't Click* ».

Mais en réalité c'est beaucoup moins compliqué qu'on veut bien le dire. Et maintenant passons aux choses sérieuses.

Conventions d'écriture dans ce document :

Times New Roman 12pts, noir	*Texte explicatif*
Arial 10pts, bleu	*Commandes à saisir dans R après le prompt* >
`Courrier New 10pts, rouge`	*Réponses de R à ces commandes*

2. FONCTIONS DE BASE

2.1 manipulation des objets

R utilise un langage informatique dit *orienté objet*. Ne partez pas ! *Vous n'avez rien à programmer*, mais vous pourrez vous délecter des aspects pratiques qui en découlent. Pour faire simple, un objet est une sorte de fichier. Il peut contenir par exemple des données, ou bien le résultat de tests que vous avez demandés. La très grande supériorité d'un objet sur un fichier est la suivante. Les seules opérations que vous puissiez effectuer sur un fichier sont : ouvrir/fermer et copier/déplacer/effacer. Pour agir sur le *contenu* d'un fichier, il faut d'abord l'ouvrir. Combiner le contenu de fichiers différents par exemple demande des opérations fastidieuses à base de couper/coller. Pas dans R. En effet, tout objet possède une structure interne formée d'objets plus petits sur lesquels vous pouvez agir séparément, mais aussi globalement. Pour l'instant, ça ne vous dit rien mais vous verrez que c'est d'une efficacité redoutable. Voyons simplement comment on entre des données dans un objet.

La manière la plus primaire est de saisir les données directement dans R. Rassurez-vous, la manière de récupérer les données de vos tableaux Excel (ou autre) vous est présentée aussi, en **Annexe 1**. Supposons pour l'instant que nous voulions étudier la taille moyenne d'étudiants de sexe masculin et féminins en vous basant sur deux échantillons aléatoires de 10 et 8 personnes respectivement. La saisie des données est alors :

```
> men = c(172.5, 175, 176, 177, 177, 178.5, 179, 179, 179.5, 180)
> women = c(167, 168, 168.5, 170, 171, 175, 175, 176)
```

Notez bien les points suivants :

1) on peu nommer les objets librement, donc autant le faire de manière explicite. Le choix de l'anglais évite d'utiliser les lettres accentuées (é, à, ç...) que R n'aimera pas trop si vous utilisez une version anglaise du logiciel, origine anglo-saxonne oblige. Un conseil : dans R, nommez donc vos variables avec des noms anglais, ça vous évitera d'ajouter inconsciemment des accents partout.
2) pour former une liste de données qui sera ensuite manipulée éventuellement d'un bloc, il suffit de lui donner un nom, de faire suivre ce nom du signe "=" puis de lister les valeurs séparées par des virgules, le tout entre parenthèses ( ) et précédé de la lettre c (qui est l'abréviation de *concatenate* c'est à dire "faire une chaîne avec les éléments")
3) du fait que la virgule sert de séparateur entre les valeurs, le séparateur décimal dans R est le **point décimal** anglo-saxon et non la virgule.

R sait maintenant que les noms men et women désignent deux objets bien précis, et il est capable de les manipuler globalement comme d'effectuer des opérations sur eux. Il peut évidemment lister leur contenu, il suffit de les appeler par leur nom (peut on imaginer plus simple ?) :

```
> men
```

Réponse :

```
[1] 172.5
[2] 175
[3] 176
[4] 177
[5] 177
[6] 178.5
[7] 179
[8] 179
[9] 179.5
[10] 180
```

Vous remarquerez que le listing est numéroté, et vous indique le rang de chaque valeur dans le tableau initial. Pour connaître respectivement la moyenne, la variance, l'écart-type ou tout simplement pour vérifier l'effectif de votre échantillon, les instructions sont les suivantes (en `rouge` les réponses de R):

```
> mean(men)
[1] 177.35
```
=Donner la moyenne de l'objet men

```
> var(men)
[1] 5.502778
```
Donner la variance de l'objet men

```
> sd(men)
[1] 2.3458
```
Donner l'écart-type (sd=*standard deviation*) de l'objet men

```
> length(men)
[1] 10
```
Donner le nombre d'éléments (length= longueur) de l'objet men

Petite remarque : il n'existe pas de fonction "donner l'erreur standard", tout simplement parce que vous pouvez le faire tout seul au moyen de la formule classique "erreur standard de la moyenne=racine(variance/effectif)", donc si vous la voulez, écrivez:

```
> sqrt(var(men)/length(men))
[1] 0.7418071
```
Calculer la racine carrée (sqrt=*square root*) du ratio entre la variance de men et le nombre d'éléments (length) de men.

Supposons maintenant que nous voulions faire un **test t de Student** entre ces deux moyennes. Avec un logiciel classique, il nous faudrait cliquer sur un menu déroulant "*tests paramétriques*" puis "*test de comparaison de moyennes*" puis , puis "*test t de Student*", etc et une dizaine de clics plus tard vous auriez enfin eu ce que vous vouliez. Maintenant comparez avec ceci :

```
>t.test(men,women)
```

Alors, c'est compliqué R ? Nous découvrirons la réponse de R à cette demande de test de Studentdans la section **4.1** consacrée au test de Student. Voyons pour l'instant quelques autres fonctions de base *très utiles*.

2.2. La fonction stripchart( ) : examinez vos données individuelles

Cette fonction établit (par défaut, horizontalement, comme sur une corde à linge) un graphe montrant *chaque point individuel de données*. Si vous avez plusieurs échantillons, chacun aura droit à sa corde à linge. Si vous préférez une orientation verticale, demandez l'option vertical=TRUE. Par ailleurs, pour éviter que les points de valeur identique soient cachés les uns sous les autres, choisissez l'option method="jitter", qui les décale.

Exemple : dans mon répertoire de travail, le fichier texte `sizes.txt` (mis en forme comme expliqué en **Annexe 1**) contient trois colonnes contenant respectivement la taille d'étudiants MBPE de sexe mâle (student), celles de leurs pères (father) et mère (mother).

Chargement du tableau dans un objet nommé par exemple mydata

> mydata=read.table("sizes.txt",header=T)

Placer dans l'objet mydata un tableau de données construit à partir des informations contenues dans le fichier sizes.txt se trouvant dans le répertoire de travail actif. Ne pas tenir compte de la première ligne du fichier car elle contient les noms des colonnes (header=T)

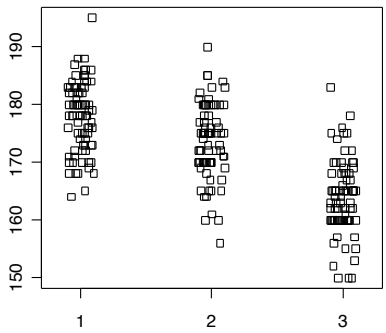
>attach(mydata)

Pour indiquer à R que les noms de colonnes mentionnés dans la suite des événements sont ceux de mydata.

Jetons un coup d'oeil à la répartition des trois nuages de données student, father et mother :

>stripchart(list(student,father,mother), method="jitter", vertical=T)

Voici le résultat, incrusté sous forme d'un fichier métafile à partir de la fenêtre graphique ouverte automatiquement par R quand on réclame une figure :



Par défaut, les numéros de l'axe des X désignent les traitements dans l'ordre de l'instruction list( ). Il faudrait un test pour en être certain, mais on repère déjà ici que les fils (1) sont plutôt plus grands que les pères (2) (qui sont eux-mêmes — sans surprise — plus grands que les mères (3)).

Ce type de graphe est dit *exploratoire*. Il permet en particulier de déceler les erreurs de saisie dans les fichiers (les points aberrants du type "virgule oubliée" sautent aux yeux). Surtout, ce type de figure a le grand mérite de vous obliger à *regarder vos données individuelles* et donc à ne jamais oublier leur *variabilité*. Vous évitez ainsi de foncer un peu trop vite sur les moyennes, dont le chiffre unique "bien propre", cache parfois un vaste marécage brumeux. Un truc à connaître : quand il y a beaucoup de données comme ici, il vaut mieux utiliser des symboles "ouverts". Leur empilement noircit alors le fond graduellement, en vous donnant une bonne idée de la *densité* des individus selon la zone du graphe, autrement dit vous accédez visuellement à la notion statistique de *densité de probabilité*.

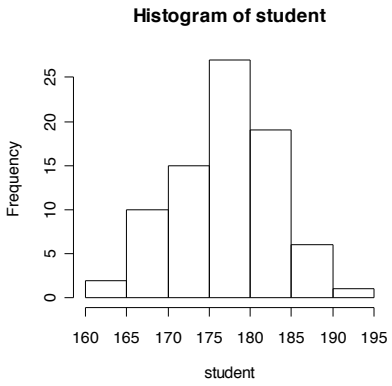
2.2. La fonction hist( ) : réalisez instantanément un histogramme de la distribution

Dans Excel, réaliser l'histogramme d'une distribution à partir d'une liste de données brutes est pénible car rien n'est prévu pour. Il vous faut d'abord définir vos intervalles de classe, puis trier les données, puis compter vous-mêmes combien il y a d'individus dans chaque classe, puis saisir ces effectifs (et le nom des intervalles) dans un nouveau coin de la feuille de calcul, et enfin cliquer encore une demi douzaine de fois avant de pouvoir contempler votre oeuvre. Et si vous n'êtes pas content de votre histogramme et que vous voulez changer vos intervalles de classe... **tout est à recommencer**. R est bien plus malin que ça, il fait tout le sale boulot à votre place.

Et maintenant, voici comment obtenir l'histogramme des tailles des étudiants par exemple :

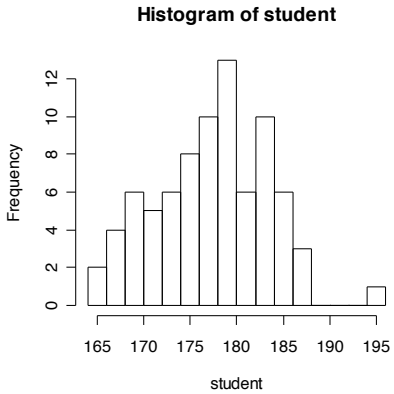
```
> hist(student)
```

Est il humainement possible de concevoir une instruction plus simple ?



R a choisi automatiquement les intervalles de classe. Cependant, modifier ce paramètre selon vos besoins précis est un jeu d'enfant, il suffit de spécifier combien de coupures (breaks) vous voulez sur l'axe des X lorsque vous demandez l'histogramme. Supposons que vous vouliez une vingtaine de classes :

```
>hist(student,breaks=20)
```



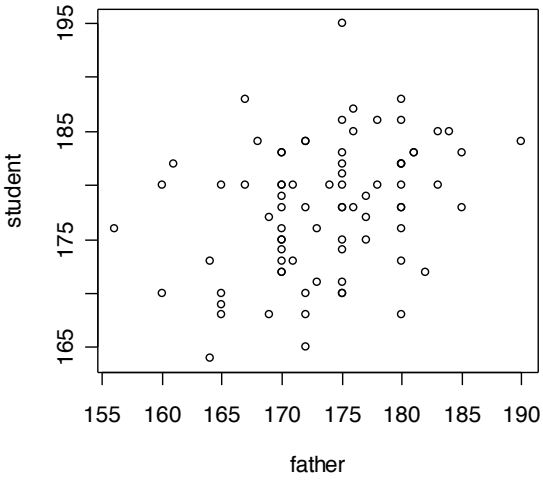
Et hop là ! C'est tout simplement magique. Explorer une distribution polymodale à la recherche de cohortes est nettement facilité avec ce genre d'outils.

2.3. La fonction plot( ) : examinez et personnalisez rapidement vos nuages de points

Deux syntaxes possibles pour obtenir un nuage "Y = f(X)":

>plot(X,Y) ou bien >plot(Y~X) (le signe "tilde" signifiant "en fonction de")

Exemple : taille de l'étudiant en fonction de la taille de son père
>plot(student~father)



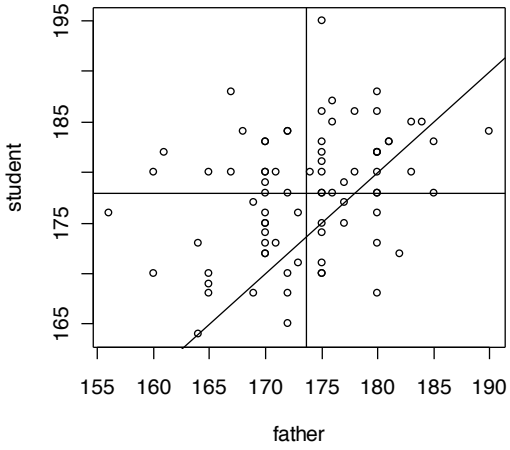
Il est maintenant possible de modifier ce graphe, en examinant par exemple si les points sont répartis harmonieusement de part et d'autre de la droite de pente 1:1 partant de zéro (ça n'est pas le cas, il y en a plus au dessus car les fils sont plus grands que les pères en moyenne), et on peut également facilement créer une "mire" centrée sur le centre de gravité du nuage en traçant deux droites passant par la moyenne de student et celle de father. Admirez la simplicité des commandes :

>abline(0,1) Tracer une droite d'ordonnée à l'origine 0 et de pente 1

>abline(h=mean(student)) Tracer une horizontale d'ordonnée égale à la moyenne de student

>abline(v=mean(father)) Tracer une verticale d'abscisse égale à la moyenne des données father

Résultat de toutes ces commandes (qui s'appliquent sur le graphe actif)



Le décalage du centre de gravité du nuage par rapport à la droite de rapport 1:1 (fils et pères de même taille) saute aux yeux ici. la suite des opérations impliquerait probablement un test statistique, mais on sait déjà à peu près où on va.

3. INTERVALLES DE CONFIANCE

3.1. Intervalle de confiance d'une moyenne, loi normale

Cette condition est remplie dans l'un des deux cas suivants : (1) si la variable étudiée suit une loi normale, (2) si l'échantillon est de grande taille (N>30). Dans ce second cas, *quelle que soit* la loi suivie par la variable étudiée, la moyenne calculée sur notre échantillon de grande taille suit au moins approximativement une loi normale.

Reprenons l'exemple de la taille des hommes (données contenues dans l'objet men). L'échantillon est petit, mais la taille chez l'homme suit une loi normale (nous sommes donc dans le cas 1). la fonction qui nous permet d'obtenir l'intervalle de confiance est en fait un sous-module du test t de Student de *conformité avec une valeur théorique*. Si nous voulions savoir par exemple si la taille moyenne de la population dont men est issu est significativement différente de 175cm. nous utiliserions ce type de test t de Student.

Or, au cours de ce test, R calcule par routine l'intervalle de confiance de notre moyenne, et place ce résultat dans un objet nommé `conf.int`. C'est cet objet que nous allons pêcher, et lui seul au moyen de la commande $ (dollar). En effet, le reste des résultats produits par le test ne nous intéresse pas ici. Il nous suffit de préciser $conf.int à la suite de l'instruction demandant le test t de Student, ainsi :

```
>t.test(men)$conf.int
```

```
[1] 175.6719 179.0281
attr(,"conf.level")
[1] 0.95
```

Effectuer un test t de Student de conformité sur men et afficher uniquement l'objet conf.int

l'$IC_{95\%}$ de la taille des hommes de cette population est donc [175,7 — 179,0cm]

3.2 Intervalle de confiance d'une moyenne, loi non normale

Supposons maintenant que notre échantillon soit petit (n <<30) et que la variable soit distribuée de manière inconnue, voire connue pour être éloignée de la loi normale (ex. binomiale négative). On peut tout de même calculer facilement non pas l'intervalle de confiance de la moyenne mais celle de la *médiane*. Ceci s'effectue en utilisant le test de Wilcoxon de conformité à une médiane théorique, exactement de la même manière que nous venons d'utiliser le test t de Student (cf **3.1**). La seule différence est que le test de Wilcoxon ne calcule pas l'IC par défaut, il faut le demander au moment du test en précisant l'option conf.int=TRUE:

```
> wilcox.test(men,conf.int=TRUE)$conf.int
```

Effectuer un test de Wilcoxon sur l'objet men, incluant le calcul d'un intervalle de confiance de la médiane (conf.int=TRUE), et afficher uniquement le résultat contenu dans l'objet conf.int.
IC95% = [175,5 — 179,0cm]

```
[1] 175.5000 179.0001
attr(,"conf.level")
[1] 0.95
```

Un intervalle de confiance a bien été calculé, et il est remarquablement similaire à celui obtenu par la méthode paramétrique du t de Student (voir section 3.1).

Rappel : l'intervalle de confiance calculé ici n'est pas vraiment celui de la moyenne mais celui de la médiane. Une autre méthode, plus compliquée, permet de calculer les intervalles de confiance de moyennes ou de n'importe quoi d'autre dans n'importe quelle situation (le bootstrap). Elle nécessite d'écrire quelques lignes d'instructions et vous la découvrirez bientôt dans la section 3.4, qui traite du **bootstrap**.

3.3 Intervalle de confiance d'un pourcentage

Un pourcentage repose sur *N* individus répartis en deux catégories mutuellement exclusives. Lorsque la catégorie dont l'effectif est le plus petit comporte quand même nettement plus de 5 individus, vous n'avez pas besoin de logiciel, votre calculatrice fera très bien l'affaire. L'intervalle de confiance d'un pourcentage *p* se calcule alors *approximativement* grâce à la loi normale, et vaut environ :

$$IC_{95} = p \pm 1{,}96 \times \sqrt{\ [p \times q/(N-1)]}$$

Exemple : sur 20 individus, il y a 5 mâles (soit $p = 0{,}25$). Quel est l'intervalle de confiance de ce pourcentage ?

$$IC_{95} = 0{,}25 \pm 1{,}96 \times \sqrt{\ (0{,}25 \times 0{,}75/19)} = [\ 0{,}055 \text{ — } 0{,}445\ ] = [\ 5{,}5\ \% \text{ — } 44{,}5\%\ ]$$

J'ai gardé les décimales pour rester fidèle au calcul mais vous aurez compris qu'un intervalle de confiance aussi vaste mérite simplement d'être écrit [5% — 46%].

Rappel : ce calcul est *approximatif* et d'autant plus grossier que l'effectif le plus faible est petit. Avec cinq individus mâles seulement, on risque ici une *grosse* approximation. Voyons plutôt la méthode infaillible, que vous pouvez utiliser quel que soit votre effectif. Il s'agit « tout simplement » du calcul de la probabilité exacte basé sur la loi binomiale. Dans R, le calcul de cet intervalle de confiance est intégré dans le **test Binomial exact**, qu'on va utiliser ici pour lire seulement la partie qui nous intéresse (l'intervalle de confiance), comme nous l'avons fait dans les deux sections précédentes. La syntaxe de ce test est la suivante :

```
>binom.test(x,n,p=)
```

x le nombre d'individus de la catégorie qui vous intéresse
n l'effectif total
p la proportion théorique de l'hypothèse Ho (utile seulement lorsqu'on fait un test. par défaut, elle est réglée sur *p*=0,5)

Donc ici (5 mâles sur 20 individus), on demande (sans préciser de proportion théorique puisqu'on s'en moque):

```
> binom.test(5,20)$conf.int
```

Effectuer un test binomial exact pour savoir si la proportion observée "5 sur 20" s'éloigne significativement de la proportion théorique (par défaut, le test utilisera p=50%) mais afficher seulement l'intervalle de confiance de la proportion observée, contenu dans l'objet conf.int

```
[1] 0.08657147 0.49104587
attr(,"conf.level")
[1] 0.95
```

D'ou il ressort que le véritable IC95% n'était pas *exactement* celui de notre calcul approximatif précédent, il vaut en réalité : [9% — 49%].

3.4 Intervalle de confiance de *n'importe quoi* : le bootstrap

Bien qu'il puisse être appliqué à n'importe quel paramètre calculé à partir des données et quelle que soit la distribution de la variable aléatoire, je vais illustrer le principe ici avec le calcul par bootstrap de l'intervalle de confiance d'une *moyenne*, la variable suivant par surcroît une loi *normale*. Ainsi, nous pourrons comparer la performance de cette méthode avec celle de la méthode traditionnelle qui nécessite une distribution normale. Le décor : un petit échantillon aléatoire tiré dans une loi normale bien connue : celle qui régit le Quotient Intellectuel dans l'espèce humaine, qui est sur tous les continents et à toutes les altitudes de moyenne $\mu = 100$ et d'écart-type $\sigma = 15$. On découvrira au passage l'utilisation des instructions permettant de *générer des résultats aléatoires selon une loi normale*.

```
> QI=rnorm(10,mean=100,sd=15)
```

Tirer aléatoirement 10 individus dans une loi normale (rnorm=*random normal*) de moyenne 100 et d'écart type 15 (sd=*standard deviation*)

```
> QI
 [1]  79.78505 104.69105  79.75360  81.86443  92.40690  85.90238  82.56055
 [8]  83.13703  96.42429 104.02850
```

Un peu difficile à lire. J'arrondis grâce à la commande round en demandant 0 décimales (c'est juste un prétexte pour vous présenter aussi cette commande):

```
> QI=round(QI,0)
```

Arrondir les valeurs de QI sans aucune décimale et placer le résultat à nouveau dans QI

```
> QI
 [1]  80 105  80  82  92  86  83  83  96 104
```

Vous remarquerez que les arrondis sont effectués de manière correcte. Que vaut la moyenne ?

```
>mean(QI)
[1] 89.1
```

Notre groupe semble intellectuellement plus faible que la moyenne théorique de 100. Naturellement ça n'est ici qu'un effet purement aléatoire (on a seulement 10 individus, les fluctuations d'échantillonnage sont donc importantes). Notre mission: calculer un intervalle de confiance autour de la moyenne observée (89,1) pour savoir dans quelle zone se cache la véritable moyenne μ de la population (dont on sait **déjà** ici qu'elle vaut 100 mais dans la réalité on ne le saurait évidemment **pas,** sinon pourquoi échantillonner ?)

Méthode paramétrique habituelle : l'IC95% de la moyenne via le test t (cf section 3.1)

```
> t.test(QI)$conf.int

 [1] 82.22653 95.97347
attr(,"conf.level")
[1] 0.95
```

Donner l'intervalle de confiance de QI calculé par la méthode paramétrique du t de student

$IC_{95\%}$=[82 — 96 points de QI]

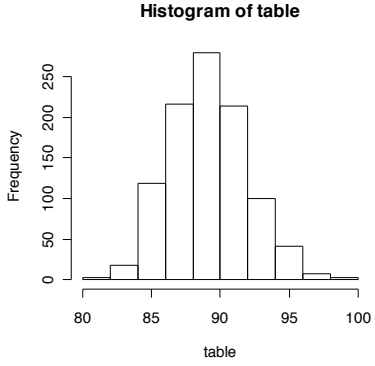
Au passage, nous avons droit ici à une très belle[2] démonstration du fait qu'un intervalle de confiance **à 95%** manque parfois sa cible (pour être précis, il la manque évidement dans... 5% des cas, soit quand même une fois sur vingt): ici il n'a pas réussi à capturer la véritable moyenne de la population échantillonnée (qui vaut 100), même si elle lui échappe de justesse.

Méthode 2, le bootstrap (qui implique d'utiliser quelques brèves lignes d'instructions, mais on va tout expliquer au fur et à mesure)

`>table=numeric(1000)`	Créer un vecteur (tableau à une seule rangée) nommé table devant accueillir 1000 valeurs *numériques*. Il contiendra les 1000 valeurs de moyennes obtenues par tirage avec remise au sein même du jeu de données. Initialement, il ne contient que des zéros.
`>for(i in 1:1000)`	Annonce d'une boucle utilisant un compteur i et qui va tourner 1000 fois
`+{`	Accolade annonçant une *suite d'instructions*. Ici, début de la boucle
`+table[i]=mean(sample(QI,10,replace=T))`	Placer dans la case de table de rang [i] la moyenne (mean) obtenue en tirant 10 individus *avec remise* (replace=T pour *TRUE*) dans QI
`+}`	Accolade de fin de boucle. Vous remarquez que tant que l'accolade de fin de boucle n'apparaît pas, le prompt > est remplacé par un signe + si vous passez à la ligne. R s'attend à une *suite* d'instructions, il attend patiemment l'accolade de fin.

La méthode du bootstrap vous paraît naturellement plus compliquée que l'autre, mais elle a l'énorme avantage de pouvoir être appliquée à *absolument n'importe quoi* et pas seulement aux moyennes. C'est tout son intérêt. Vous pourriez par exemple ici calculer facilement l'intervalle de confiance de la *variance* en remplaçant "mean" par "var" sur la ligne 4 de ce "programme".

En attendant, vous pouvez visualiser la distribution obtenue par le bootstrap en demandant hist(table) cf ci-contre.



[2] et involontaire, je vous assure que je n'ai pas triché.

Ne reste plus qu'à déterminer les bornes de l'intervalle de confiance à 95%. Il nous faut pour cela exclure de chaque côté 2,5% des valeurs, soit (sur 1000 valeurs) les 25 valeurs les plus basses (la borne est au rang 25) et les 25 valeurs les plus haute (la borne est au rang 975, à condition de ranger les valeurs par ordre croissant, évidemment) :

```
>table.sorted=sort(table)
```
Trier table par ordre croissant de ses valeurs et placer le résultat dans table.sorted

```
> c(table.sorted[25], table.sorted[975])
```
Afficher les éléments de table.sorted de rang [25] et [975]

```
[1] 84.2 95.0
```
Bornes de l'$IC_{95\%}$ obtenues: [84,2 — 95,0]

L'intervalle de confiance obtenu par bootstrap est très similaire au précédent (et lui aussi passe à côté de la véritable valeur 100).

4. COMPARAISONS DE MOYENNES

4.1 Comparaison de deux moyennes, test t de Student

L'utilisation de ce test suppose soit que la variable étudiée suive au moins approximativement une loi normale soit que vous ayez plus de 30 individus par échantillon. Nous retrouvons nos deux échantillons de tailles men et women (cf **1.2**). Comme la taille dans l'espèce humaine est notoirement distribuée selon une loi normale, le fait que nous n'ayons que 10 et 8 individus respectivement par échantillon n'est vraiment pas un problème, le test t de Student s'applique.

Sa syntaxe est:

```
> t.test(men,women)
```

Comparer (les moyennes de) men et women par un test t

On peut difficilement faire plus sobre La réponse de R est assez riche, on va l'examiner ligne par ligne.

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

C'est un test t un peu spécial (voir plus bas).

```
data:  men and women
```

Rappel de la source des données

```
t = 4.1368, df = 11.628, p-value = 0.001472
```

Valeur du t, nombre de degrés de liberté (`df`=*degree of freedom*), valeur de la probabilité *P* associée au test (`p-value`)

```
alternative hypothesis: true difference in means is not equal to 0
```

L'hypothèse "alternative" (H1) est que les moyenne sont *différentes* (sous entendu : l'hypothèse nulle Ho est "*les moyennes sont égales*")

```
95 percent confidence interval:
2.846265 9.228735
```

Intervalle de confiance à 95% **de la différence entre les deux moyennes** (c'est à ce genre de détails qu'on voit que R a été conçu par des gens qui comprennent ce qui est *vraiment important* en sciences)

```
sample estimates:
mean of x mean of y
177.3500  171.3125
```

Rappel des moyennes comparées

Le nombre de degrés de liberté *décimal* (`df = 11.628` ?!?) peut en ébouriffer certains (en tout cas il m'a sacrément interpellé au niveau du vécu la première fois) mais résulte d'une approximation, due à Welsh, qui tient compte du fait que les variances ne sont pas supposées être égales dans l'option « par défaut » du test t utilisé par R, ce qui représente l'approche *la plus prudente possible*. Le test t de Welsh n'est donc pas le test t « classique ». Il est cependant aisé d'utiliser le test t « canal historique » (autrement dit le t de Student), il suffit de préciser à R que les variances sont supposées être égales :

```
> t.test(men,women, var.equal=TRUE)
```

Attention à bien utiliser les majuscules pour TRUE. Par contre, inutile d'écrire TRUE en entier, vous pouvez utiliser seulement l'initiale, comme ceci :

```
> t.test(men,women, var.equal=T)
```

Quoi qu'il en soit, voici la réponse :

```
Two Sample t-test
data:  men and women
t = 4.3341, df = 16, p-value = 0.0005129
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 3.084405 8.990595
sample estimates:
mean of x mean of y
177.3500  171.3125
```

Vous aurez remarqué que le nombre de ddl est maintenant plus élevé (`16` au lieu de `11,6`) et *entier*, et que le test est encore plus affirmatif que tout à l'heure (P = 0,00051229 vs P = 0.001472, avec un intervalle de confiance de la différence entre les deux moyennes qui est plus précis. C'est normal, on a fait une hypothèse supplémentaire (l'égalité des variances) qui permet de gagner de la puissance (à condition qu'elle soit vérifiée !).

4.2 Comparaison de deux moyennes, test t de Student pour séries appariées

Grands échantillons appariés (ou petits échantillon appariés si la variable étudiée suit une loi normale)

Ce test a son intérêt lorsque les données sont organisées de manière logique par *paires*. C'est le cas particulièrement si les mêmes individus ont été mesurés deux fois (avant et après un traitement, quel que soit la nature de ce que recouvre le mot "traitement") et qu'on se demande si le "traitement" a modifié la moyenne. Pour des raisons qui serait trop longues à expliquer ici, il est peu performant de simplement comparer la moyenne par un test de Student classique. Vous comprendrez peut être d'ailleurs sans démonstration mathématique que procéder ainsi vous ferait perdre automatiquement la structure appariée des données. La bonne méthode est donc d'utiliser un test qui respecte cette structure.

Evidemment, un test apparié exige deux séries de données *de même longueur* (sinon R générera un message d'erreur, et il aura raison). Supposons que le "traitement" soit l'entraînement à réaliser une tâche et que la variable mesurée soit le score obtenu lors d'un test (on supposera, ce qui resterait à démontrer, que la variable est distribuée de manière normale).

Saisie des données :

```
> before = c(125, 130, 132, 135, 136, 138, 140, 145)
> after =c(127, 132, 133, 136, 139, 141, 145, 148)
```

La simple comparaison des *moyennes* ne révèle rien de bien spectaculaire, et on peut la calculer directement ainsi :

```
> mean(after)-mean(before)
[1] 2.5
```

A peine plus de deux points d'écart en moyenne sur plus d'une centaine, pas de quoi fouetter un chat. Si on reste focalisé sur les moyennes, on se dit que cela pourrait facilement s'expliquer par les fluctuations aléatoires de score d'une fois sur l'autre (le même individu n'obtiendra jamais exactement le même score deux fois de suite). Cependant, la comparaison des données *appariées* suggère fortement que le traitement a eu un *effet positif*: tous les scores individuels ont bougé *dans le même sens*, progressant de 1 à 5 points, c'est tout de même louche ! C'est la situation dans laquelle un test apparié démontre sa supériorité sur un test non apparié. Jugez plutôt. Voilà ce que donnerait le test de Student non apparié :

```
> t.test(before,after,var.equal=TRUE)

Two Sample t-test
data:  before and after
t = -0.7558, df = 14, p-value = 0.4623
```

```
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -9.594663  4.594663
sample estimates:
mean of x mean of y
  135.125   137.625
```

la valeur de *P* est 0,46 donc le test est largement *non significatif*, et vous constatez que l'intervalle de confiance de la différence entre les moyennes est large (la différence avant–après pourrait tout aussi bien être nettement négative que nettement positive). Le test non apparié n'a rien vu. Cependant, le test apparié ne va pas se laisser berner si facilement (pour rendre le test t apparié il suffit d'ajouter l'instruction « paired=TRUE »:

```
> t.test(avant,après,var.equal=TRUE,paired=TRUE)

Paired t-test
data:  before and after
t = -5.4006, df = 7, p-value = 0.001008
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.594608 -1.405392
sample estimates:
mean of x mean of y
  135.125   137.625
```

Cette fois, fini de rire, on détecte bien une différence hautement significative (*P* = 0,001) dont le sens est ***négatif*** (à savoir : les valeurs « avant » sont en moyenne significativement *inférieures* aux valeurs « après »). On voit ici l'importance d'utiliser un test adapté à la situation traitée. Nous serions passé complètement à côté de l'effet en utilisant un test non apparié.

4.3. Comparaison de deux moyennes, test W de Wilcoxon

Petits échantillons et variable ne suivant pas la loi normale

Rappel: même si votre variable ne suit pas une loi normle, ça n'est pas un souci si vos échantilons sont grands (n>30). En effet les moyennes que vous comparez sont des variables aléatoires qui sont, elles, approximativement distribuées selon la loi normale. Donc, effectuez un test t de Student comme si de rien était (cf 4.1)

Si vos échantillons sont petits, et en particulier si votre variable est vraiment distribuée de manière bizaroïde (beaucoup de valeurs 0 ou de valeurs élevées) il ne faut pas utiliser un test t de Student. Vous ferez alors appel à un test non paramétrique tel le *test U de Mann et Whitney* ou bien le *test W de Wilcoxon* (c'est la même chose sous deux habillages différents, et on les appelle de plus en plus le *test de Mann-Whitney-Wilcoxon*). Ces deux tests raisonnent non pas sur les valeurs mais sur leurs rangs relatifs. Vous pouvez donc les utiliser chaque fois qu'on peut hiérarchiser les individus. Strictement parlant, ces tests ne comparent pas les moyennes mais les médianes (la médiane est la valeur qui sépare votre échantillon en deux parties égales, avec autant d'individus au dessus que en dessous).

Je reprend l'exemple men *vs* women utilisé pour le test t :

La syntaxe est :

```
>wilcox.test(men,women)
```

Comparer (les médianes de) men et women par un test W de Wilcoxon

```
Wilcoxon rank sum test with continuity correction
data:  men and women
W = 74.5, p-value = 0.00243
alternative hypothesis: true mu is not equal to 0
Warning message:
Cannot compute exact p-value with ties in: wilcox.test.default(men, women)
```

Ne vous inquiétez pas de ce dernier message d'avertissement, il indique simplement qu'il y a des ex-aequo qui empêchent de calculer la valeur de *P* **exacte** dans sa plus parfaite pureté mathématique (ça n'est pas grave)

Ici encore, on décèle un écart nettement significatif ($P = 0{,}002$).

4.4. Comparaison de deux moyennes, test W de Wilcoxon pour séries appariées

Echantillons appariés et variable ne suivant pas la loi normale

Tout ce qui été dit plus haut sur l'intérêt d'utiliser un test **apparié** quand on mesure successivement les mêmes individus reste valable. Je reprend l'exemple de l'entraînement et des scores avant/après.

```
> before = c(125, 130, 132, 135, 136, 138, 140, 145)
> after =c(127, 132, 133, 136, 139, 141, 145, 148)
```

Si on teste sans appariement :

```
> wilcox.test(before,after)
```

```
Wilcoxon rank sum test with continuity correction
data:  avant and après
W = 24.5, p-value = 0.4613
alternative hypothesis: true mu is not equal to 0
```

On ne décèle pas d'écart significatif. Par contre, si on tient compte de l'appariement :

```
> wilcox.test(before,after,paired=T)
```

```
Wilcoxon signed rank test with continuity correction
data:  before and after
V = 0, p-value = 0.01356
alternative hypothesis: true mu is not equal to 0
```

Abracadabra ! La différence est maintenant très bien détectée.

4.5 Comparaison simultanée de plus de deux moyennes, ANOVA suivie du test HSD de Tukey

Grands échantillons (N>30) ou petits échantillons si la variable étudiée suit une loi normale :

ICI, prochainement, la réponse à cette passionnante question (si vous êtes impatients, allez dans l'aide de R et faites >help(aov) si vous l'osez). Précisons toutefois que

1) l'ANOVA vous est inutile pour le rapport marée
2) je pense qu'il en est de même pour les rapports éthologie
3) dans tous les cas ou une ANOVA à un facteur est applicable, le test H de Kruskal-Wallis (décrit en **4.6**) fonctionne et donnera des résultats similaires, au prix d'une très modeste perte de puissance).

4.6 Comparaison simultanée de plus de deux moyennes, test H de Kruskal-Wallis

Petits échantillons (N<<30) et variable suivant une loi inconnue ou non normale

Ce test effectue l'équivalent d'une ANOVA à un facteur (une seule variable étudiée), à ceci près qu'il est applicable même si la variable étudiée ne suit pas la loi normale et qu'il y a peu d'individus dans un ou plusieurs échantillons. C'est la généralisation du test de Wilcoxon (=test de Mann-Whitney) à plusieurs échantillons. Strictement parlant, il ne compare pas les *moyennes* des échantillons mais leurs *médianes* (la valeur qui sépare l'échantillon en deux parties contenant le même nombre d'individus)

Il y a deux syntaxes possibles pour ce test :

```
>Kruskal.test(list(liste des échantillons à traiter))
```

```
>Kruskal.test(<objet qui contient les données>,g=<objet qui contient les numéros d'échantillon>)
```

Si vous avez peu de valeurs et peu d'échantillons, le moyen le plus rapide est de les saisir directement dans autant d'objets qu'il y a d'échantillons. Exemple de trois échantillons x, y et z de 5, 5 et 4 individus :

```
>x=c(1,2,5,8,12)
> y=c(5,8,12,25,32)
> z=c(25,33,39,42)
```

Pour indiquer sa cible à R, il suffit de former une liste contenant x, y et z par l'instruction list.

Le test est alors :

```
> kruskal.test(list(x,y,z))
```

Effectuer un test H de Kruskal-Wallis comparant les échantillons de la liste formée par x, y et z

```
Kruskal-Wallis rank sum test
data:  list(x, y, z)
Kruskal-Wallis chi-squared = 8.6712,
df = 2, p-value = 0.01309
```

Test significatif, au moins une des trois médianes est différente des deux autres (ou bien elles sont toutes différentes les unes des autres).

L'autre syntaxe nécessite d'avoir les données listées à la queue-leu-leu dans un premier objet, et la liste (dans l'ordre !) des numéros de groupes (traitements) auxquels ils appartiennent dans un autre objet. Dans la pratique, cela permet de désigner comme objet deux colonnes de tableaux, l'une contenant les données et l'autre les numéros de groupes.

5. COMPARAISON DE POURCENTAGES

5.1 Comparaison d'un pourcentage observé avec un pourcentage théorique, le test binomial exact.

Taille de l'échantillon quelconque

Ce test fonctionne en utilisant la formule de la loi binomiale et en calculant la probabilité exacte d'obtenir sous l'effet du hasard un pourcentage *encore* plus éloigné du pourcentage théorique p que le pourcentage p_{obs} que nous observons. Contrairement au test du chi2 de conformité, le test binomial exact fonctionne même avec des effectifs minuscules (mais n'attendez pas de miracles : il ne pourra alors déceler que des écarts *importants* par rapport à la théorie)

Sa syntaxe est :

```
>binom.test(x, n, p = <à indiquer>)
```

x le nombre d'individus de la catégorie qui vous intéresse
n l'effectif total
p la proportion théorique de l'hypothèse Ho (exprimée entre 0 et 1, réglée automatiquement sur 0.5 si vous n'indiquez rien)

Exemple : En théorie, une population humaine compte 10% de gauchers. Or, ayant échantillonné aléatoirement 10 étudiants de master scientifique, vous trouvez 3 gauchers. Les étudiants de ce type de master sont plus souvent gauchers que la population générale ?

Le test demandé est alors :

```
>binom.test(3, 10, p=0.1)
```

```
Exact binomial test
data:  3 and 10
number of successes = 3, number of trials = 10, p-value = 0.07019
alternative hypothesis: true probability of success is not equal to 0.1
95 percent confidence interval:
 0.06673951 0.65245285
sample estimates:
probability of success
                   0.3
```

Tester par un test binomial exact la proportion 3 succès sur 10 essais, la probabilité théorique de succès à chaque tirage étant 0.1
Test non significatif (P = 0,070), mais le calcul de l'intervalle de confiance du pourcentage de gauchers nous prévient très clairement de la *grande imprécision* de l'information disponible :
$IC_{95\%}$ = [6,7% — — 65,2%]

En clair nous pouvons simplement affirmer que le pourcentage réel de gauchers dans cette population étudiante est vraisemblablement... compris **entre 6,7% et 65,2%** ! Il serait donc sage d'accumuler davantage de données si nous voulons vraiment répondre à notre question...

5.2 Comparaison d'une distribution observée avec une distribution théorique : le test du chi2 de conformité

> **ATTENTION**, dans ce cas particulier, R exige bizarrement que les valeurs ***théoriques*** soient données sous forme des ***proportions*** attendues (entre zéro et 1) alors que les valeurs ***observées*** doivent être données sous forme ***d'effectifs*** (entre zéro et l'infini).

Dans le cas ou il y a plus de deux catégories, le test binomial exact vu plus haut déclare forfait. Il faut utiliser un test du chi2, qui a le défaut de vous obliger à avoir des effectifs théoriques minimaux de $n = 5$ dans chaque case. Pour en savoir la raison, et savoir que faire si vous êtes dans cette situation désagréable, ouvrez un livre de stats ou lisez le chapitre sur le chi2 de *Statistiques pour statophobes*.

La syntaxe dans R est :

```
>chisq.test(liste des effectifs,p=liste des proportions théoriques)
```

Exemple : parmi les 89 individus obtenus lors d'un croisement, la répartition des homozygotes et hétérozygotes est 25 **AA**, 59 **Aa** mais 5 **aa** seulement, alors que les proportions Mendéliennes attendues sont respectivement 0,25 : 0,5 : 0,25. On observe ici un déficit en hétérozygotes **aa** tout à fait suspect. Une vérification rapide nous confirme qu'on peut bien utiliser le chi2 puisque le plus petit de nos trois effectifs théoriques est :

$$0{,}25 \times 89 = 22{,}25 >> 5$$

Rappel, dans R il faut écrire pour les valeurs observées, des **effectifs** :

```
> observed=c(25, 59, 5)
```

Mais pour les proportions théoriques, des **proportions** (entre zéro et 1):

```
>theo=c(0.25,0.5,0.25)
```

Avec ces noms d'objets, le test chi2 de conformité est alors :

```
> chisq.test(observed,p=theo)
```

Réponse :

```
Chi-squared test for given probabilities
data:  observed
X-squared = 18.4382, df = 2, p-value =
9.913e-05
```

La mention « *for given probabilities* » vous indique bien que c'est un test de conformité à une distribution théorique de probabilités que vous avez fournies.

Nous avions raison de nous inquiéter, il se passe quelque chose de tout à fait anormal. La probabilité d'observer une telle distribution si vraiment on est dans une population suivant les lois de Mendel est $P < 0{,}0001$.

5.3 Comparaison entre elles de plusieurs distributions observées, le test du chi2 d'homogénéité (ou d'indépendance)

Condition : pas plus de 20% des effectifs *théoriques* (obtenus par calcul) inférieurs à 5 (règle de Cochran)

Dans cette seconde situation, pas de distribution théorique fournie à l'avance, rien que des observations. Vous avez deux (ou plus) échantillons avec à chaque fois (les mêmes) k catégories mutuellement exclusives. Pour savoir si vos échantillons sont homogènes (ou encore, pour déterminer si les lignes sont indépendantes des colonnes), le test de choix est habituellement le chi2 d'homogénéité. Sa syntaxe est alors:

>chisq.test(nom du tableau à analyser)

La méthode la plus fruste (donc ma préférée) pour fabriquer ce tableau est de le saisir ligne à ligne puis de lier les lignes par la commance rbind (*row bind*), ce qui forme un tableau. Voyez vous mêmes :

```
> males=c(0,12,25)
> females=c(12,16,45)
> juveniles=c(15,43,1)
> table=rbind(males,females,juveniles)
```

Un petit coup d'oeil à la table ainsi créée :

```
> table
           [,1] [,2] [,3]
males         0   12   25
females      12   16   45
Juveniles    15   43    1
```

Les noms `[,1] [,2] [,3]`sont les noms que R donne par défaut aux colonnes. Ils pourraient être (par exemple) le haut, le milieu et le bas d'un estran. Le test est alors :

```
> chisq.test(table)
Pearson's Chi-squared test
data:  table
X-squared = 65.7, df = 4, p-value = 1.832e-13
```

La probabilité d'observer un tableau aussi déséquilibré simplement sous l'effet du hasard est microscopiquement faible ($P < 1{,}8 \times 10^{-13}$). On s'en doutait un peu, soit dit en passant...

Rappel . Ce test est soumis aux limitations habituelles des chi2 : on peut tolérer seulement 20% des effectifs *théoriques* inférieurs à 5. Le calcul des effectifs théoriques en question est expliqué en détail dans tous les bouquins, y compris dans le chapitre de *Statistiques pour statophobes* qui porte sur le chi2. Si vous êtes coincés car vos effectifs *théoriques* (pas ceux de votre tableau de données !) sont trop petits, utilisez le test exact de Fisher (section suivante).

5.4 Comparaison entre elles de plusieurs distributions observées quels que soient les effectifs, le test exact de Fisher

Aucune condition sur la taille des échantillons.

Ce test s'utilise dans les mêmes situations que le chi2 d'homogénéité (cf 5.3), **mais** sans aucune contrainte de taille d'échantillon. Son seul inconvénient — qui explique l'hégémonie historique du chi2 — est de nécessiter un ordinateur alors que le chi2 peut facilement s'effectuer avec une simple calculette. Le principe du test exact de Fisher consiste en effet à générer les dizaines, centaines, ou centaines de milliers de tableaux possibles avec les totaux de vos lignes et de vos colonnes, puis de calculer la proportion exacte (d'ou son nom) de ceux qui sont *encore plus* éloignés de l'hypothèse nulle "le tableau est équilibré" que vos propres résultats. Cette probabilité est celle d'obtenir sous le seul effet du hasard des résultats encore plus éloignés que les vôtres de l'hypothèse nulle. Si cette probabilité est faible, on conclut que le hasard n'a pas pu faire ça tout seul, et qu'il y a donc un effet réel.

Sa syntaxe est simplement >fisher.test(nom du tableau de données)

Comme vu pour le chi2, le tableau peut être facilement rentré ligne par ligne, en liant ensuite les lignes par la commande rbind (pour *row bind*) suivi du nom des lignes à lier. Exemple.

```
> males=c(2,1,1)
> females=c(1,2,2)
> juveniles=c(0,0,3)
> tableau=rbind(males,females,juveniles)
```

```
> tableau
           [,1] [,2] [,3]
males         2    1    1
females       1    2    2
juveniles     0    0    3
```

On demande le test :

```
> fisher.test(tableau)
Fisher's Exact Test for Count Data
data:  tableau
p-value = 0.5909
alternative hypothesis: two.sided
```

Le test est *largement* non significatif ici : $P = 0{,}59$ (près de 60% de chances d'obtenir un tableau encore plus déséquilibré que le tableau de données simplement sous l'effet du hasard). A titre de comparaison, le chi2 (utilisé ici hors de son domaine de validité) serait trop optimiste puisqu'il obtient une probabilité $P = 0{,}28$; avec toutefois la même conclusion qualitative : test non significatif (voir ci-dessous, et notez l'avertissement signalant que le chi2 donne un résultat probablement *très approximatif* puisqu'il est utilisé hors de son domaine normal) :

```
> chisq.test(tableau)
Pearson's Chi-squared test
data:  tableau
X-squared = 5.1, df = 4,
p-value = 0.2772
Warning message:
Chi-squared approximation may be incorrect in: chisq.test(tableau)
```

6 MESURE DE LA LIAISON ENTRE VARIABLES QUANTITATIVES

6.1 Calcul de la force de la liaison entre deux variables quantitatives distribuées normalement l'une par rapport à l'autre : le coefficient de corrélation "R" de Pearson.

La condition impressionnante citée dans le titre (nommée "binormalité") signifie que pour un X d'une valeur donnée, les valeurs Y observées doivent être distribuées à peu près normalement, et vice versa. Comme d'habitude, elle est d'autant plus importante que l'effectif est faible. Si vous avez beaucoup de données, ça n'est pas un souci. Le coefficient de corrélation R que vous allez obtenir variera de 1 (liaison parfaite, Y augmente linéairement avec X) à –1 (idem mais Y diminue linéairement avec X). Une valeur de zéro signifierait une totale indépendance entre X et Y. le problème est bien sûr que dans la réalité, les R parfaitement nuls n'existent pas à cause des fluctuations d'échantillonnage, d'où la nécessité d'un test statistique pour vérifier si le R calculé est significativement différent de zéro.

Exemple : y a t-il un lien statistique entre la longueur et la largeur des feuilles de gougnafier à fleurs bleues ? Dans cet exemple simpliste on saisit les données directement dans R (dans la réalité il y en aura probablement beaucoup plus, voir en **Annexe 1** comment importer les données à partir d'un fichier)

```
> length=c(1,5,10,15,20)
> width=c(3,7,25,20,50)
```

L'obtention du coefficient de corrélation est très simple :

```
> cor(length,width)
[1] 0.9167439
```

Cette valeur de 0,917 est très proche de 1, mais est-ce **significativement** différent de zéro ? La question peut sembler absurde mais méfiance : on a ici *très peu de données*. Il vaut mieux vérifier en faisant un test qui permettra de déterminer l'intervalle de confiance de notre R.

```
> cor.test(length,width)
Pearson's product-moment correlation
data:  length and width
t = 3.9748, df = 3, p-value = 0.02847
alternative hypothesis: true
correlation is not equal to 0
95 percent confidence interval:
 0.1803332 0.9945810
sample estimates:
      cor
0.9167439
```

Nous voilà rassurés car le test est significatif (P = 0,028). Cependant, l'intervalle de confiance *extrêmement large* de ce coefficient de corrélation nous appelle à la prudence puisqu'il vaut [0,18 — 0,99]. la force de la liaison n'est peut être pas si grande que ça...:

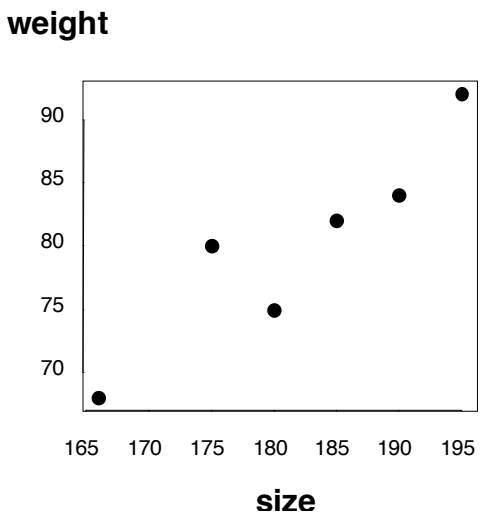
Pour concrétiser la force de cette liaison, petit rappel : il faut élever ces R au carré. la borne inférieure donne un pitoyable $(0{,}18)^2 = 0{,}0324$, soit 3%. Avec cet extrême, fixer X permettrait seulement de réduire la variabilité de Y de 3%...

6.2 Calcul de la force de la liaison entre deux variables quantitatives non distribuées normalement l'une par rapport à l'autre : le coefficient de corrélation "R" de Spearman.

Ce coefficient doit être utilisé à la place du R de Pearson lorsque la condition de binormalité des variables étudiées est fausse, ce qui pourrait causer des distorsions nuisibles en particulier si l'effectif est faible.

Exemple :

```
>table=data.frame(size=c(166,175,180,185,190,195),weight=c(68,80,75,82,84,92))
> table
  size weight
1  166     68
2  175     80
3  180     75
4  185     82
5  190     84
6  195     92
> attach(table)
> plot(size,weight)
```




```
>cor.test(size,weight,method="spearman")

Spearman's rank correlation rho
data:  size and weight
S = 2, p-value = 0.01667
alternative hypothesis: true rho is
not equal to 0
sample estimates:
      rho
0.9428571
```

Effectuer un test du coefficient de corrélation de Spearman sur la liaison entre size et weight

test significatif ($P = 0{,}017$)

6.3 Détermination de la pente reliant une variable causale X à une variable expliquée Y, avec Y distribuée normalement pour un X donné : régression linéaire de Y en X.

Cas typique: on s'intéresse à une relation dose/effet (peu importe les unités, résultats fictifs). J'en profite pour illustrer la méthode permettant d'obtenir en une seule étape un tableau de données pouvant être traité par une analyse par régression linéaire[3] :

```
>table=data.frame(dose=c(1,2,5,10,20),effect=c(2,3,6,15,30))
```

Stocker dans table un tableau de données (data.frame) comportant une colonne nommée dose contenant les valeurs 1,2,5,10,20et une colonne nommée effet avec les valeurs 2,3,6,15,30.

Vérification du contenu :

```
> table
   dose    effect
1     1         2
2     2         3
3     5         6
4    10        15
5    20        30
```

On demande ensuite la régression ainsi :

```
> myresult=lm(effect~dose,data=table)
```

Stocker dans myresult le résultat d'une régression linéaire (lm = *linear model*) étudiant effect en fonction de (symbole ~) dose, les données à analyser (data) se trouvant dans table

Ne reste plus qu'à faire parler myresult:

```
> summary(myresult)
Call:
lm(formula = effect ~ dose, data = table)
Residuals:
      1       2       3       4       5
 0.7164  0.2139 -1.2935  0.1940  0.1692

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.21891    0.58027  -0.377 0.731093
dose         1.50249    0.05636  26.659 0.000116 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 0.8753 on 3 degrees of freedom
Multiple R-Squared: 0.9958,     Adjusted R-squared: 0.9944
F-statistic: 710.7 on 1 and 3 DF,  p-value: 0.0001158
```

Un minimum de décodage n'est peut-être pas inutile.

[3] voir en annexe comment utiliser un volumineux fichier de données sans avoir à le saisir à nouveau dans R.

Les `Residuals` (résidus) sont les écarts résiduels subsistant entre chaque donné et la droite de régression. Si les résidus sont systématiquement positifs d'abord puis négatifs par la suite (ou l'inverse), cela met en doute la linéarité de vos données (en clair, vos données dessinent une *courbe*, et non une droite), et une régression linéaire n'est pas la bonne méthode. Aucun souci dans notre cas (mais inutile de mettre des drapeaux aux fenêtres, avec si peu de données seul un écart monstrueux à la linéarité aurait pu être détecté).

La suite est un tableau d'analyse proche dans sa structure de celui d'une ANOVA. Deux tests t sont effectués (avant dernière colonne) avec les probabilités associées rangées dans la dernière colonne.

L'effet `intercept` (ordonnée à l'origine de la droite, valeur `-0.21891)` est *non significatif* ($P = 0{,}73$) donc l'ordonnée à l'origine n'est pas significativement différente de zéro. Dans notre cas, cela signifie que pour une dose nulle, l'effet pourrait être nul. Ca n'a rien de systématique : si la dose d'engrais artificiel apporté dans un champs est nulle, le rendement obtenu ne sera pas égal à zéro.

L'effet `dose` (pente de `1.50249`) est *hautement significatif* ($P < 0{,}001$), ce qui (dans une situation expérimentale ou X est le seul facteur variable) démontrerait bien une liaison de cause à effet entre X et Y.

Notre droite de régression linéaire peut donc s'écrire :

effect = **1.50249** dose **-0.21891**

Vous noterez que le tableau vous fournit aussi le coefficient de corrélation de Pearson qui vaut R=0,9944.

Annexe 1 : travailler à partir de fichier de données volumineux saisies initialement dans Excel

Les manoeuvres suivantes n'ont d'intérêt que si vous avez déjà saisi d'impressionnantes colonnes de chiffres. Si c'est juste pour vous éviter de saisir à nouveau une quinzaine de valeurs, sachez que vous allez perdre largement plus de temps qu'en saisissant à nouveau vos valeurs dans des objets R directement à l'écran.

Nous allons commencer par spécifier dans quel répertoire R ira puiser vos données si nécessaire. Pour éviter la pagaille dans vos fichiers, il vous est vivement conseillé de créer un dossier qui contiendra les fichiers de données de votre projet au lieu de les enfouir dans le répertoire par défaut lorsque R s'ouvre. En supposant que le chemin d'accès de votre dossier de travail soit **D:/mondossier**, la manoeuvre à effectuer est simplement de saisir ceci après le prompt :

```
> setwd(« d:/mondossier »)
```

Ce qui est l'abréviation de "*set working directory as* d:/mondossier", c'est à dire "utiliser d:/mondossier comme répertoire de travail". Respectez les guillemets. Si vous les oubliez, R fera la source oreille. En matière de respect des usages, R est encore plus psychorigide que moi (comme quoi il ne faut jamais désespérer, on trouve toujours pire).

Pendant qu'on en parle, souvenez-vous bien des choses suivantes, sinon je vous prédis bien des messages d'erreur:

1) le séparateur décimal de R est le point décimal, pas la virgule.

2) il faudra toujours préciser si la première ligne du tableau contient les titres des colonnes (par défaut, R essaiera de les lire comme des données)

3) NE FAITES PAS comme dans Excel, des tableaux complètement hétéroclites avec des colonnes à moitié vides et des effectifs différents et même plusieurs tableaux sans aucun lien dans le même fichier (et des figures au milieu par dessus le marché). R ne peut avaler que des jeux de données bien propres et bien carrés (**pas de cases vides**).

Supposons que votre tableau Excel ressemble à ceci :

Données de notre super sortie à Penvins, qu'est ce qu'on a rigolé !

	Supra	médiosup	médioinf	infra
	20,5	20,2	20,8	0
	12,6	15	15,2	0,25
	3	0	14,3	0
	15	0	7	
	15,256	0	5	
moyenne	**13,2712**	**7,04**	**12,46**	**0**

Ne pas oublier : dentiste à 17h30

Naturellement votre véritable tableau serait beaucoup plus grand mais inutile de gâcher du papier pour rien. Il va maintenant falloir faire le ménage. On commence par **vérifier visuellement la vraisemblance des données saisies**, ce qui permet de constater que la moyenne de la colonne infra est manifestement fausse. Une fois cette vérification effectuée, on découpe à la souris la seule partie à utiliser pour l'analyse (encadrée en pointillés bleu ci-dessus), qu'on colle par exemple dans le presse-papier ou n'importe quel éditeur de texte de base :

```
Supra médiosup    médioinf    infra
20,5  20,2  20,8  0
12,6  15    15,2  0,25
3     0     14,3  0
15    0     7
15,256      0     5
```

Le nombre de décimales gagnerait à être harmonisé, et le nom de nos colonnes est trop long, De plus, il contient des lettres accentuées (déconseillé) et des mélanges de majuscules et de minuscules (vous avez 99 chances sur 100 de ne plus vous souvenir que c'est à ***S**upra* seulement que vous avez mis la majuscule, ce qui vous garantit des messages exaspérants car incompréhensibles quand vous voudrez utiliser le nom de vos colonnes). Enfin, mais c'est le plus important) il faut impérativement remplacer ces virgules par des **points**, et placer la mention **`NA`** (*not available*, donnée manquante) dans les cases vides. R veut des tableaux « carrés ». La fonction édition/remplacer du bloc-notes le fera sans problèmes, et finalement vous obtiendrez ceci :

```
supr  msup  minf  infr
20.5  20.2  20.8  0
12.6  15    15.2  0.25
3     0     14.3  0
15    0     7     NA
15.3  0     5     NA
```

Vous sauvegardez alors ce fichier (le bloc notes le fera automatiquement au format **`.txt`** que R sait lire sans problèmes) dans votre répertoire de travail, sous un nom si possible *explicite*. En effet, votre répertoire de travail sera bientôt encombré de dizaines de fichiers portant *tous* sur votre manip. Donc, évitez les titres flous du style « `moules.txt` »

Ne reste plus qu'à charger les données dans R, ce qui se fait semble t-il de *tas* de manières différentes, dont la seule que je comprenne pour l'instant est l'instruction read.table ( ).

Syntaxe

```
>densityperzone=read.table(« estran.txt », header=TRUE)
```

Mettre dans l'objet densityperzone un tableau de données (*table*) construit à partir du fichier estran.txt (contenu dans le répertoire actif actuellement), sans tenir compte de la première ligne du fichier car elle contient des titres (*headers*) et non des données.

Vous êtes maintenant prêts à travailler. Pour effectuer par exemple un test de Wilcoxon sur les deux premières colonnes, vous demanderez :

```
>wilcox.test(densityperzone$supr,densityperzone$msup)
```

Plutôt compliqué à écrire, j'avoue . L'autre option, plus raffinée, consiste à **attacher** votre objet densityperzone d'abord (pour signaler à R que c'est à propos de lui que vous posez la question) ce qui permet de simplifier la syntaxe du test en étant plus obligé de rappeler le nom de l'objet suivi du signe $ à chaque variable :

```
>attach(densityperzone)
```

L'objet étant attaché, c'est comme si votre radar de tir avait accroché la cible, tout devient facile :

```
>wilcox.test(supr, msup)
```

Effectuera le test sans se tromper de données (et surtout en sachant où les chercher).

Pour travailler ensuite sur un autre fichier sans vous mélanger les variables, il faut **détacher** le précédent :

```
>detach(densityperzone)
```

Vous voilà libres.

Et si (enfer !) vous n'aviez pas donné de nom à vos colonnes dans votre tableau de départ ? Pas de panique, par défaut elles se nomment `[,1]`, `[,2]` etc., donc :

```
>wilcox.test(densityperzone[,1],densityperzone[,2])
```

fera un test de wilcoxon comparant les médianes des deux premières colonnes.

Annexe 2; Démonstration de l'impuissance totale du test W de Wilcoxon face à deux échantillons vraiment trop petits

Comparaison de deux échantillons de 3 individus (ou 3 valeurs)

```
> A=c(0,1,2)
> B=c(100,150,5000)
```

La différence A *vs* B est clairement colossale. Et pourtant, voyez :

```
> wilcox.test(A,B)

Wilcoxon rank sum test
data:  A and B
W = 0, p-value = 0.1
alternative hypothesis: true mu is not equal to 0
```

Le test n'est même pas significatif à 5% ! (ici, $P = 0{,}1$).

On augmente la taille de B (4 individus)

```
> B=c(100,150,5000,6000)
> wilcox.test(A,B)

Wilcoxon rank sum test
data:  a and b
W = 0, p-value = 0.05714
alternative hypothesis: true mu is not equal to 0
```

Toujours pas significatif (bien que de justesse) si on veut respecter $\alpha = 0{,}05$

On augmente A (4 vs 4)

```
> A=c(0,1,2,3,4)
> wilcox.test(A,B)

Wilcoxon rank sum test
data:  A and B
W = 0, p-value = 0.02857
alternative hypothesis: true mu is not equal to 0
```

Enfin une proba < 0,05 !!! .

Retenez donc qu'un test de Wilcoxon est **complètement aveugle** tant que vous n'avez pas **au moins 4 individus** dans chacun de vos deux échantillons. Et encore faudra t-il pour atteindre la significativité dans ces conditions que les 4 individus d'un des échantillons aient tous une valeur inférieure à celle du plus faible de l'échantillon opposé.

Aide-mémoire

Intervalle de confiance d'une moyenne (grand échantillon sample, ou variable suivant une loi proche de la loi normale)
>t.test(sample)$conf.int

Intervalle de confiance d'une médiane (petit échantillon sample, variable suivant une loi inconnue ou connue pour être éloignée de la loi normale)
>wilcox.test(sample,conf.int=T)$conf.int

Intervalle de confiance d'un pourcentage (n individus parmi N)
>binom.test(n,N)$conf.int

Comparaison de deux moyennes (grands échantillons sampleA, sampleB ou petits échantillons mais variable suivant une loi normale) : **Test t de Student**
>t.test(sampleA, sampleB, var.equal=T)

Comparaison de deux médianes (petits échantillons sampleA, sampleB variable suivant une loi inconnue ou connue pour être éloignée de la loi normale) : **Test W de Wilcoxon** (= U de Mann et Whitney)
>wilcox.test(sampleA, sampleB)

Comparaison d'un nombre quelconque de médianes (exemple : trois petits échantillons sampleA, sampleB, sampleC suivant une loi inconnue ou connue pour être éloignée de la loi normale) : **Test H de Kruskal-Wallis**.
>kruskal.test(list(sampleA, sampleB, sampleC))

Comparaison d'un pourcentages observé (n individus parmi N) avec une fréquence théorique F (exprimée entre 0 et 1). **Test binomial exact**.
>binom.test(n,N,p=F)

Comparaison d'une distribution observée (effectifs a,b,c,d) avec une distribution théorique. (proportions attendues F1, F2, F3, F4 exprimées entre 0 et 1). Grand échantillon. **Test du Chi2 de conformité**.
>table=data.frame(obs=c(a,b,c,d), p=c(F1,F2,F3,F4))
>chisq.test(table)

<pas d'équivalent si petit échantillon. Regrouper des classes est la seule solution>

Comparaison entre elles de plusieurs distributions observées (ex. D1, D2, D3, contenant des suites **d'effectifs** a, b, c, d...). Grands échantillons. **Test du Chi2 d'homogénéité** (=d'indépendance).
>table=data.frame(D1=c(a,b,c,d), D2=c(e,f,g,h),D3=c(i,j,k,l))
>chisq.test(table)

Comparaison entre elles de plusieurs distributions observées (ex. D1, D2, D3, contenant des suites **d'effectifs** a, b, c, d...). Echantillons de taille quelconque. **Test exact de Fisher.**
>table=data.frame(D1=c(a,b,c,d), D2=c(e,f,g,h),D3=c(i,j,k,l))
>fisher.test(table)

Corrélation entre variables quantitatives X et Y distribuées normalement l'une par rapport à l'autre. Test du R de Pearson.
>cor.test(X,Y)

Corrélation entre variables X et Y quelconques mais pouvant donner lieu à des rangs. Test du R de Spearman..
>cor.test(X,Y, method="spearman")

Régression linéaire de Y (variable supposée expliquée) en X (variable supposée causale).
>myresult=(lm(X~Y))
>summary(myresult)