Aide à l'utilisation de R pour l'enseignement de biostatistique de base

Marie Laure Delignette-Muller

12 novembre 2024

Table des matières

1 Introduction

1.1 Qu'est-ce que R ?

R est à la fois un langage de programmation dédié à l'analyse statistique de données et un logiciel très performant, avec un domaine d'application très large, mais nécessitant un apprentissage. Ce document a pour but de vous permettre de l'utiliser de façon la plus simple possible (sans trop d'investissement de votre part) pour analyser vos données dans les cas les plus basiques.

Pour un apprentissage de la programmation avec R, se référer à l'introduction sous forme d'exercice réalisée au TD 1. Et n'oubliez pas, vous avez toujours accès à l'aide de chaque fonction R en tapant `?nomdelafonction`.

Pour ceux qui voudraient mieux maîtriser le langage R, un manuel introductif est fourni par les auteurs du logiciels à l'adresse http://www.r-project.org/ dans la rubrique " Documentation " à la sous rubrique " Manuals " (https://cran.r-project.org/manuals.html), et divers manuels sont aussi proposés par d'autres auteurs dont certains en français dans la même rubrique, au lien proposé nommé " Contributed documentation" (https://cran.r-project.org/other-docs.html).

Lorsque l'on utilise cet outil pour des analyses statistiques que l'on souhaite publier dans une revue scientifique, la référence à citer est celle que l'on peut obtenir dans Rstudio en tapant `citation()` dans la console (fenêtre par défaut en bas à gauche) et qui dépendra de la version que vous avez téléchargée :

```
citation()

## To cite R in publications use:
##
##   R Core Team (2024). _R: A Language and Environment for Statistical
##   Computing_. R Foundation for Statistical Computing, Vienna, Austria.
##   <https://www.R-project.org/>.
##
## Une entrée BibTeX pour les utilisateurs LaTeX est
##
##   @Manual{,
##     title = {R: A Language and Environment for Statistical Computing},
##     author = {{R Core Team}},
##     organization = {R Foundation for Statistical Computing},
##     address = {Vienna, Austria},
##     year = {2024},
##     url = {https://www.R-project.org/},
##   }
##
## We have invested a lot of time and effort in creating R, please cite it
## when using it for data analysis. See also 'citation("pkgname")' for
## citing R packages.
```

1.2 Interface Rstudio

En TD nous utiliserons R via l'application Rstudio qui fournit une interface conviviale pour programmer en R : elle permet une gestion agréable de l'ensemble des fenêtres utiles lors de la programmation avec R et la coloration syntaxique du script qui aide à repérer d'éventuelles erreurs de frappe notamment. Elle est disponible pour MAC, PC et linux, et rend l'utilisation de R plus uniforme d'un système d'exploitation à l'autre. Elle permet aussi de créer des rapports d'analyse de données.

R et Rstudio sont installables à partir de l'url suivante : https://posit.co/download/rstudio-desktop/. Il convient d'installer R puis Rstudio. Rstudio peut aussi être utilisé en ligne sur les tablettes ou les ordinateurs sur lesquels l'application ne peut être installée (cas rares). Il convient poru cela de créer un compte sur le cloud de Rstudio (https://posit.cloud/, cf. explications détaillées dans la section 1.9).

Il est fortement déconseillé de taper du code R directement dans la console de Rstudio (fenêtre de commande, située par défaut en bas à gauche de l'interface de Rstudio), excepté pour des codes que l'on ne veut pas garder, comme les appels à l'aide d'une fonction donnée : `? nomdelafonction`). Il est possible d'écrire son code dans un script (sauvé en fichier .R afin d'en garder une trace et de pouvoir lancer d'un coup toutes les lignes de code) ou dans un **rapport d'analyse (fichier rmarkdown, avec l'extension .Rmd)** comprenant le code et des commentaires formatés. La compilation d'un tel fichier permettra de générer un document contenant en plus les sorties de R associées, y compris les sorties graphiques. **En TD nous privilégierons l'écriture et la sauvegarde du code R et des commentaires dans un tel fichier rmarkdown**. Les

parties comprenant du code R dans un fichier .Rmd sont appelés des **chunks**, et sont reconnaissables par les lignes qui les encadrent (cf. ci-dessous). Dans un chunk on peut aussi mettre des commentaires comme ci-dessous (lignes commençant par un dièse (ou hashtag) qui ne seront pas compilées).

```
```{r}
# commentaire 1
code R

# commentaire 2
code R
```
```

1.3 A FAIRE en début, en cours et en fin de session Rstudio

- En début de session on **ouvre un fichier Rmarkdown (.Rmd) existant ou nouveau** (ou un script, fichier .R), et si c'est un nouveau **on le sauve d'emblée avec l'extension .Rmd (resp. .R) dans le même dossier que celui où sont les fichiers de données** (pas obligatoire mais plus simple).
- On **change le répertoire de travail** (Onglet Session, Set Working Directory) et on le définit au répertoire où se trouve le script et les données (option **To Source File location**).
- On **sauve le script très régulièrement** ! ! !
- En fin de session **Rstudio vous demande si vous voulez sauver l'espace de travail** (workspace). Il est fortement conseillé de **toujours refuser** pour éviter des problèmes (sauf cas très exceptionnels qui ne devraient pas vous concerner de suite).

1.4 A SAVOIR avant de démarrer

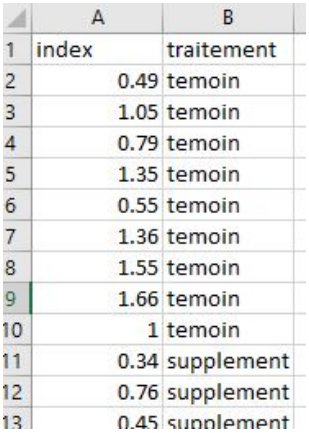
- R est sensible à la casse (majuscule / minuscule)
- Dans un code R , tout ce qui se trouve à droite du signe # est considéré comme un commentaire, donc non interprété comme du code (mais bien utile pour comprendre son code).
- Le séparateur de décimale dans R est le point et non la virgule.
- Le symbole <- est utilisé pour affecter une valeur à un objet. Mieux vaut ne pas le remplacer par le symbole = même si souvent ça fonctionne (mais pas toujours !). Le symbole = est à réserver pour la définition des valeurs des arguments dans l'appel à une fonction et le symbole == pour tester une égalité dans une condition logique.
- Nommez les objets R avec des noms parlants (pas trop courts donc) mais pas trop longs non plus.
- Evitez de donner des noms qui sont susceptibles de correspondre à des noms de fonctions R (ex. : `median`)
- Ne commencez pas le nom d'un objet par un chiffre.
- Evitez les accents, les espaces et les caractères spéciaux dans les noms des objets R ainsi que dans les fichiers de données que vous allez importer dans R (donc y penser bien en amont dès la saisie des données dans un tableur par exemple). Les seuls caractères spéciaux conseillés dans les noms d'objets, tant qu'ils ne sont pas en début de nom, sont `_` et `.` (ex. : `animal_prefere`).
- Dans les données, codez les variables quantitatives par des nombres (donc pas `J1`, `J3`, `J10` par ex.) et les variables qualitatives par des chaînes de caractères (donc pas 1 pour les mâles et 0 pour les femelles par ex.). Cela vous évitera une étape de recodage dans R.
- Les données manquantes doivent être codées `NA` (pas de cellule vide dans un tableau de données que l'on souhaite importer dans R).

1.5 Comment importer des données pour les analyser avec R

Dans R, on travaille généralement avec des jeux de données codés sous la forme d'objet de type `data.frame`, c'est-à-dire une liste de vecteurs correspondant aux différentes variables étudiées, chacun étant formé soit d'un ensemble de nombres (vecteur numérique) soit d'un ensemble de mots (vecteur de chaînes de caractères). L'argument `stringsAsFactors` que nous verrons plus tard permet de faire en sorte que les vecteurs numériques soient considérés comme des variables quantitatives et les vecteurs de chaînes de caractères comme des variables qualitatives ou facteurs.

Un des moyens les plus commodes pour importer des données dans R, consiste à **créer le jeu de données dans Microsoft Excel** puis à le **sauver au format texte** (dans Microsoft Excel aller dans "Fichier", "Enregistrer sous" et **choisir dans "Type de Fichier" le format "Texte(séparateur :tabulation)(*.txt)"** et donner un nom du type "nomfichier.txt" au fichier).

Le jeu de données importé doit donc avoir le format classique en statistique : **une colonne par variable, et une ligne par individu, animal ou plus largement unité d'observation, avec en première ligne du fichier de données l'intitulé de chaque vecteur colonne, c'est-à-dire le nom de chaque variable (comme dans l'exemple ci-dessous)**.



	A	B
1	index	traitement
2	0.49	temoin
3	1.05	temoin
4	0.79	temoin
5	1.35	temoin
6	0.55	temoin
7	1.36	temoin
8	1.55	temoin
9	1.66	temoin
10	1	temoin
11	0.34	supplement
12	0.76	supplement
13	0.45	supplement

Il est fortement conseillé de ne pas mettre de caractères spéciaux dans ces intitulés, ni d'espace, ni d'accents.

Lorsqu'il y a des trous dans le fichier de données (données manquantes) il est impératif de les coder par les deux caractères `NA`.

Avant chaque analyse, il est nécessaire de charger les données dans R à partir du fichier "nomfichier.txt" pour les stocker dans un objet de type `data.frame` que l'on nommera comme on le souhaite (`dtartre` dans le code suivant chargeant les données du fichier `tartre.txt`) :

```
dtartre <- read.table("tartre.txt", header=TRUE, dec=".", stringsAsFactors = TRUE)
```

Si R vous dit qu'il ne trouve pas le fichier, il se peut que vous ayez oublié de spécifier le nom du dossier de travail, cf. chapitre 1.3.

Si vous partez d'un tableau dans lequel certaines cellules contenaient plusieurs mots séparés par des espaces vous pouvez ajouter l'argument `sep` à l'appel de `read.table()` comme ci-dessous pour spécifier que le seul séparateur de colonne reconnu est la tabulation.

```
dtartre <- read.table("tartre.txt", header=TRUE, dec=".", stringsAsFactors = TRUE, sep = "\t")
```

Décrivons les différents arguments utilisés dans les appels à la fonction `read.table()` :

- L'argument `header` est fixé à `TRUE` si les intitulés (noms des colonnes qui seront utilisés comme noms pour les variables) sont présents dans le fichier de données (en première ligne) et à `FALSE` s'ils sont absents.
- L'argument `dec` correspond au séparateur des décimales. Dans la configuration française standard de nos ordinateurs, ce séparateur est une virgule, mais dans la configuration anglo-saxonne standard ce séparateur est un point. Si l'on oublie de spécifier cet argument dans la fonction `read.table()` il sera fixé par défaut au point et cela posera problème pour lire des fichiers obtenus à partir d'Excel (sauf si l'ordinateur a été configuré avec des options régionales anglo-saxonnes où si les virgules ont été changées en points dans les fichiers issus d'Excel). Dans tous les exemples suivants, pour plus de simpicité, nous ne spécifierons pas cet argument, en supposant que les fichiers de données qui vous sont fournis comportent des points en séparateur de décimale (ce sera le cas en TD).
- L'argument `stringsAsFactors` est fixé à `TRUE`, si l'on veut qu'au moment de l'import des données, toute colonne non numérique (donc comportant des caractères ou chaîne de caractères), soit considérée comme une variable qualitative, de classe `factor`. Dans les anciennes versions de R (précédant la version 4) cet argument était fixé par défaut à `TRUE` et il n'était pas nécessaire de le spécifier, mais c'est dorénavant nécessaire.
- L'argument `sep` permet de spécifier un séparateur de colonnes, en particulier comme montré précédemment pour n'admettre que les tabulations comme séparateur de colonnes.

Si vous ne disposez pas de Microsoft Excel sur votre ordinateur, d'autres solutions sont possibles, mais le format de fichier importé sera différent et la fonction d'import aussi. Voici quelques exemples :

- Avec les tableurs des suites **Open Office ou Libre Office**, il est possible d'exporter un table de données au format CSV ("nomdefichier.csv"). Au moment de l'export, il est alors **TRES IMPORTANT de choisir le séparateur de champs** (je vous conseille d'indiquer le point virgule) et d'utiliser ce même même séparateur de champ lors de l'import du fichier dans R à l'aide de la fonction `read.csv()` comme ci-dessous :

  ```
  nomdelobjetcree <- read.csv("nomfichier.csv", header = TRUE, stringsAsFactors = TRUE, sep = ";")
  ```

- avec le **logiciel Numbers des Mac**, il est possible d'exporter aussi le fichier sous format .csv et le séparateur de colonne utilisé est le point virgule, donc on utilise le code précédent pour l'importer ensuite. Il faut juste penser à effacer le titre que met Numbers par défaut en haut du tableau avant de l'exporter pour que ça fonctionne.
- avec **Googlesheet**, vous pouvez télécharger votre fichier en format .csv avec comme séparateur la virgule. Il faudra donc l'importer en spécifiant ce séparateur comme ci-dessous :

  ```
  nomdelobjetcree <- read.csv("nomfichier.csv", header = TRUE, stringsAsFactors = TRUE, sep = ",")
  ```

1.6 Comment accéder dans R au jeu de données importé

Pour visualiser complètement le jeu de donnée chargé il suffit de taper le nom de l'objet créé par les fonctions `read.table` ou `read.csv`, par exemple ici :

```
dtartre
```

```
##    index traitement
## 1   0.49     temoin
## 2   1.05     temoin
## 3   0.79     temoin
## 4   1.35     temoin
## 5   0.55     temoin
## 6   1.36     temoin
## 7   1.55     temoin
## 8   1.66     temoin
## 9   1.00     temoin
## 10  0.34 supplement
## 11  0.76 supplement
## 12  0.45 supplement
## 13  0.69 supplement
## 14  0.87 supplement
## 15  0.94 supplement
## 16  0.22 supplement
## 17  1.07 supplement
## 18  1.38 supplement
```

Si le fichier de données est de grande taille on peut n'afficher que les premières lignes, on peut utiliser la fonction `head()` :

```
head(dtartre)
```

```
##   index traitement
## 1  0.49     temoin
## 2  1.05     temoin
## 3  0.79     temoin
## 4  1.35     temoin
## 5  0.55     temoin
## 6  1.36     temoin
```

Enfin je vous conseille fortement d'utiliser après chaque import de jeu de données la fonction `str()`, notamment sur les gros jeux de données (avec de nombreuses variables), pour obtenir une vision globale de la structure du jeu de données, décrivantsur chaque ligne de sa sortie chacune des variables du jeu de données (variables numériques avec les premières valeurs, variables qualitatives ou facteurs avec leurs modalités).

```
str(dtartre)
```

```
## 'data.frame': 18 obs. of  2 variables:
##  $ index     : num  0.49 1.05 0.79 1.35 0.55 1.36 1.55 1.66 1 0.34 ...
##  $ traitement: Factor w/ 2 levels "supplement","temoin": 2 2 2 2 2 2 2 2 2 1 ...
```

La fonction `summary()` peut aussi être utile pour obtenir un résumé statistique de chaque variable.

```
summary(dtartre)
```

```
##      index            traitement
##  Min.   :0.220   supplement:9
##  1st Qu.:0.585   temoin    :9
##  Median :0.905
##  Mean   :0.918
##  3rd Qu.:1.280
##  Max.   :1.660
```

Pour accéder séparément à chaque variable du jeu de données, on peut spécifier le nom du jeu de données suivi du signe $ puis du nom de la colonne correspondante. Par exemple pour accéder à la variable "index" on tape :

```
dtartre$index

##  [1] 0.49 1.05 0.79 1.35 0.55 1.36 1.55 1.66 1.00 0.34 0.76 0.45 0.69 0.87 0.94
## [16] 0.22 1.07 1.38
```

1.7 Manipulation et transformation de données

1.7.1 Changement de l'ordre ou des noms des modalités d'un facteur

Par défaut R classe les modalités d'un facteur par ordre alphabétique. Si l'on veut **changer cet ordre**, on peut utiliser une ligne de code de ce type :

```
dtartre$traitement

##  [1] temoin     temoin     temoin     temoin     temoin     temoin
##  [7] temoin     temoin     temoin     supplement supplement supplement
## [13] supplement supplement supplement supplement supplement supplement
## Levels: supplement temoin

dtartre$traitement <- factor(dtartre$traitement, levels = c("temoin", "supplement"))
dtartre$traitement

##  [1] temoin     temoin     temoin     temoin     temoin     temoin
##  [7] temoin     temoin     temoin     supplement supplement supplement
## [13] supplement supplement supplement supplement supplement supplement
## Levels: temoin supplement
```

On peut aussi **changer les noms des modalités** de la façon suivante :

```
levels(dtartre$traitement) <- c("contrôle", "supplément")
dtartre$traitement

##  [1] contrôle   contrôle   contrôle   contrôle   contrôle   contrôle
##  [7] contrôle   contrôle   contrôle   supplément supplément supplément
## [13] supplément supplément supplément supplément supplément supplément
## Levels: contrôle supplément
```

1.7.2 Sélection de lignes dans un jeu de données et opérateurs logiques

Si l'on veut créer un sous-jeu de données ne comprenant qu'une partie du jeu de données initial, on peut utiliser la fonction `subset()` en indiquant en premier argument le jeu de données complet, et en deuxième argument la variable logique utilisée pour sélectionner les lignes que l'on veut garder : ainsi l'exemple suivant permet de créer un nouveau jeu de données qui garde uniquement les lignes correspondant au groupe "supplement".

```
dtartre.suppl <- subset(dtartre, traitement == "supplement")
dtartre.suppl

## [1] index      traitement
## <0 lignes> (ou 'row.names' de longueur nulle)
```

Les 4 principaux opérateurs logiques dans R sont == pour "égal à", != pour "différent de", | pour le "ou" logique et & pour le "et" logique.

Lorsque l'on définit un sous-jeu de données et que cela réduit le nombre de modalités d'un facteur, R garde néanmoins en mémoire toutes les modalités du facteur présentes dans le jeu de données initiales. Pour réduire son nombre de modalités à celles présentes dans le sous-jeu de données il convient de refaire le facteur avec la fonction `factor()` comme ci-dessous.

```
d <- read.table("animfamibrut.txt", header = TRUE, stringsAsFactors = TRUE)
str(d) # pour voir ce qu'il y a dans le jeu de données

## 'data.frame': 1100 obs. of  2 variables:
##  $ espece: Factor w/ 2 levels "chat","chien": 2 2 2 2 2 2 2 2 2 2 ...
##  $ lien  : Factor w/ 4 levels "ami","enfant",..: 2 2 2 2 2 2 2 2 2 2 ...
```

```r
levels(d$lien) # pour voir les niveaux du facteur

## [1] "ami"     "enfant"  "famille" "invite"

# création d'un sous-jeu de données dans les animaux considérés juste comme "invite"
ds <- subset(d, lien != "invite")
levels(ds$lien) # on voit que le facteur garde nénanmoins le niveau invite

## [1] "ami"     "enfant"  "famille" "invite"

ds$lien <- factor(ds$lien) # on refait le facteur à partir des sous-données
levels(ds$lien) # il enlève alors bien le niveau invite

## [1] "ami"     "enfant"  "famille"
```

1.7.3 Création d'une nouvelle variable au sein du jeu de données

Pour créer une nouvelle variable intégrée au jeu de données il suffit de définir cette nouvelle variable en mettant devant son nom l'affectation au jeu de données sous la forme classique comme ci-dessous pour intégrer l'index en log directement dans le jeu de données `dtartre` :

```r
dtartre$indexenlog <- log(dtartre$index)
# on peut voir le résultat en regardant à nouveau la structure de dtartre
str(dtartre)

## 'data.frame': 18 obs. of  3 variables:
##  $ index     : num  0.49 1.05 0.79 1.35 0.55 1.36 1.55 1.66 1 0.34 ...
##  $ traitement: Factor w/ 2 levels "contrôle","supplément": 1 1 1 1 1 1 1 1 1 2 ...
##  $ indexenlog: num  -0.7133 0.0488 -0.2357 0.3001 -0.5978 ...
```

Attention, dans R le logarithme népérien est donné par la fonction `log()` et le logarithme décimal par la fonction `log10()`. Il est utile aussi de connaître la fonction qui calcule la racine carrée : `sqrt()`.

1.7.4 Création d'une variable qualitative à partir d'une variable quantitative

Une variable qualitative qui aurait été codée numériquement lors de la saisie des données n'est pas reconnue comme qualitative mais comme quantitative. Elle doit donc être transformée en facteur, à l'aide de la fonction `factor()`, si l'on veut que R la considère bien comme qualitative.

```r
varqualmalcodee <- c(1,0,0,1,1,1,0)
str(varqualmalcodee)

##  num [1:7] 1 0 0 1 1 1 0

varqual <- factor(varqualmalcodee)
str(varqual)

##  Factor w/ 2 levels "0","1": 2 1 1 2 2 2 1
```

On peut aussi facilement créer des variables qualitatives binaires, à partir de vecteurs logiques (donc à deux modalités : `TRUE` ou `FALSE`) en utilisant des opérateurs logiques vus au paragraphe précédent et le calcul vectoriel automatique de R. Par exemple la ligne de code suivante crée un vecteur logique dont la valeur sera égale à `TRUE` uniquement si l'index de tartre se situe entre 0.5 et 0.8. Ce genre de manipulation peut être utile par exemple lorsqu'on l'on veut définir une variable qualitative (dépassement ou non d'un seuil par exemple) à partir d'une variable quantitative.

```r
dtartre$index_entre_0.5ET0.8 <- (dtartre$index > 0.5) & (dtartre$index < 0.8)
head(dtartre)

##   index traitement indexenlog index_entre_0.5ET0.8
## 1  0.49   contrôle    -0.7133                FALSE
## 2  1.05   contrôle     0.0488                FALSE
## 3  0.79   contrôle    -0.2357                 TRUE
## 4  1.35   contrôle     0.3001                FALSE
## 5  0.55   contrôle    -0.5978                 TRUE
## 6  1.36   contrôle     0.3075                FALSE
```

Plus généralement, on peut utiliser la fonction `cut()` qui permet de coder l'appartenance d'une variable quantitative avec des intervalles définis par son argument `breaks`. Cette fonction renvoie un facteur (variable qualitative).

```
dtartre$index_qual <- cut(dtartre$index, breaks = c(0,0.5,0.8,2))
head(dtartre)
```

```
##   index traitement indexenlog index_entre_0.5ET0.8 index_qual
## 1  0.49   contrôle    -0.7133                FALSE    (0,0.5]
## 2  1.05   contrôle     0.0488                FALSE    (0.8,2]
## 3  0.79   contrôle    -0.2357                 TRUE  (0.5,0.8]
## 4  1.35   contrôle     0.3001                FALSE    (0.8,2]
## 5  0.55   contrôle    -0.5978                 TRUE  (0.5,0.8]
## 6  1.36   contrôle     0.3075                FALSE    (0.8,2]
```

1.8 Les graphes avec le package (ou la librairie) ggplot2

ggplot2 est un package qui a été créé en 2007 par Hadley Wickham sur une idée très originale de grammaire des graphes (gg pour "grammar of graphics"), qui permet de définir un graphe comme un objet R.

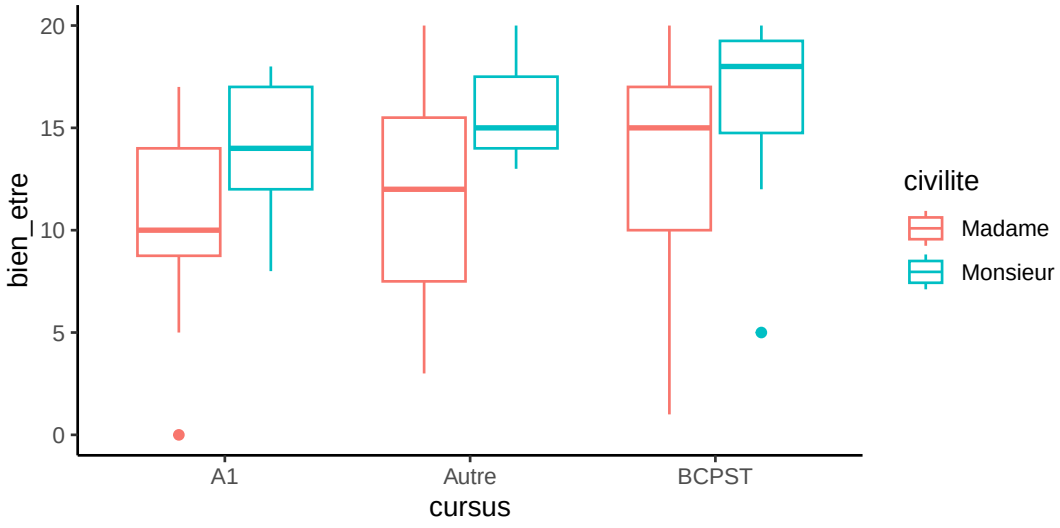
Un graphe créé à partir de la fonction `ggplot()` est un objet R défini tout d'abord par un jeu de données (argument `data`) et une esthétique (argument `mapping` définie avec la fonction `aes()`). Cette esthétique permet par exemple de définir la variable représentée sur l'axe des x, la variable représentée sur l'axe des y, la variable qui code pour la couleur, celle qui code pour la forme des points, celle qui permet de séparer des groupes... Un graphe est défini par superposition de couches (`layers`) : la fonction `ggplot()` crée un objet graphique en définissant le jeu de données et l'esthétique et il faut lui ajouter une ou des fonctions (cf. liste non exhaustive ci-dessous) avec le symbole + pour compléter sa définition.

- Pour spécifier le type de graphe on utilise les fonctions de géométrie (fonctions `geom_point()`, `geom_line()`, `geom_boxplot()`,...).
- Pour découper le graphe par groupe on utilise la fonction `facet_wrap()` pour un découpage basé sur une variable et `facet_grid()` pour un découpage basé sur deux variables.
- Pour modifier le thème de base (fond, grille) on utilise une fonction de type `theme_..()`, par ex. `theme_bw()` pour avoir un fond blanc.
- Pour ajouter / modifier titre / sous-titres, labels des axes, ..., on peut utiliser la fonction `labs()` (ex. `labs(title = "Montitre", x = "Mon label des x")`).
- Pour changer l'échelle des axes, on peut utiliser les fonctions `scale_x_log10()` et `scale_y_log10()`.

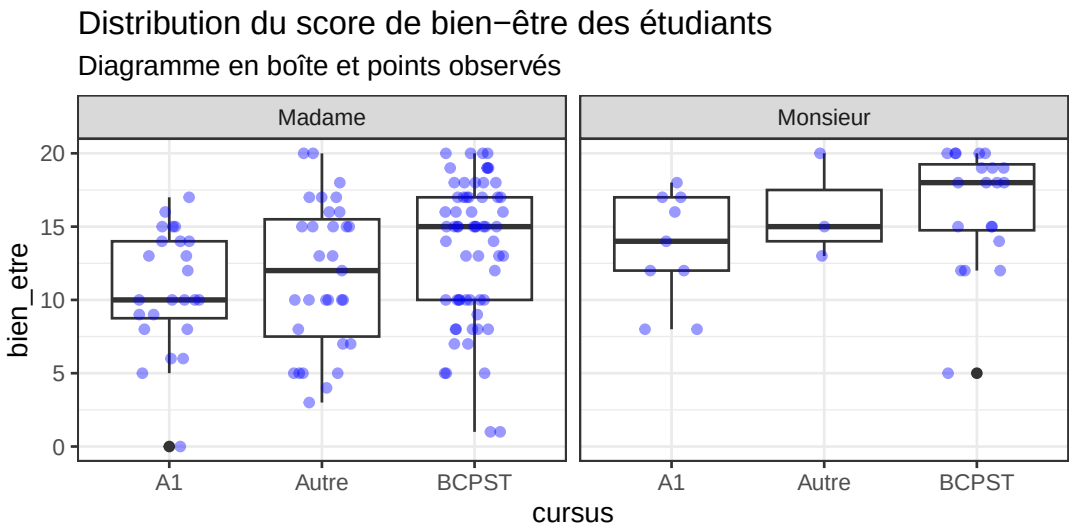
Pour un apprentissage de l'utilisation de `ggplot2`, se référer à l'introduction sous forme d'exercice réalisée au TD 2.

Voici deux exemples de graphes réalisés avec le package ggplot2 qu'il convient d'installer au préalable (dans Rstudio onglet Tools / Install packages - à faire une seule fois sur son ordinateur) puis de le charger (code `require(ggplot2)` - à faire au début du rapport d'analyse ou en début de session R) :

```
dENQ <- read.table("ENQ2223.txt", header = TRUE, stringsAsFactors = TRUE)
ggplot(data = dENQ,
       mapping = aes(x = cursus, y = bien_etre, col = civilite)) +
       geom_boxplot() + theme_classic()
```



```r
ggplot(data = dENQ, mapping = aes(x = cursus, y = bien_etre)) +
        facet_wrap(~ civilite) + theme_bw() + geom_boxplot() +
        geom_jitter(alpha = 0.4, width = 0.2, height = 0, col = "blue") +
        labs(title = "Distribution du score de bien-être des étudiants",
             subtitle = "Diagramme en boîte et points observés")
```



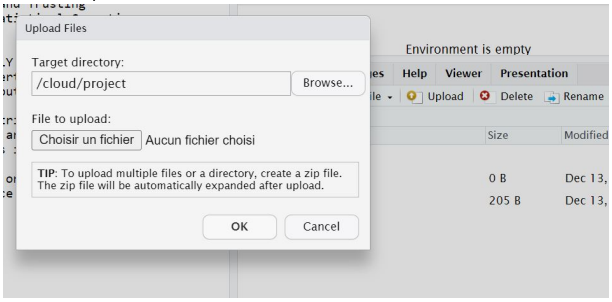
Lorsque l'on travaille avec Rstudio, les fenêtres graphiques sont générées à chaque fois qu'on fait un graphe, stockées en empilement (la plus récente devant), et accessibles avec les flèches horizontales au-dessus du bloc de gestion des graphes (en bas à droite).

Une fois un graphe réalisé, il peut être sauvé sous différents formats (jpeg, pdf, png, ...) et à différentes tailles soit à l'aide de la fonction ggsave() (voir l'aide de la fonction : ?ggsave) soit à partir de l'onglet de la fenêtre graphique "Export" de Rstudio. Les graphes sont empilés dans Rstudio et l'on peut donc remonter aux graphes précédents à l'aide des onglets "flèche" de la fenêtre graphique. On peut aussi tous les effacer à l'aide de l'onglet "balai".

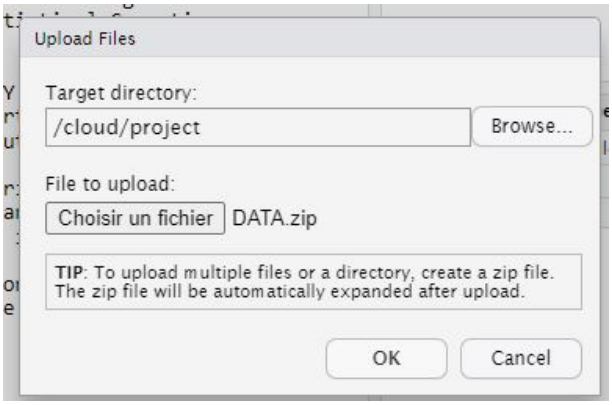
1.9 Et si l'on ne peut pas installer R sur son ordinateur ou si l'on ne dispose que d'une tablette ?

Pour ceux qui ne pourraient pas installer R sur leur ordinateur (par exemple parce qu'ils ne disposent que d'une tablette), il y a moyen de l'utiliser R en ligne en utilisant le cloud de Rstudio. Rstudio (société renommée récemment Posit) propose un cloud d'utilisation gratuite pour un usage limité (qui devrait vous suffire pour les besoins de l'enseignement de biostatistique). Il faut pour cela se créer un compte sur https://posit.cloud/ et une fois que votre compte est créé, faire un nouveau projet (bouton **New project**) en demandant bien de créer un nouveau projet Rstudio.

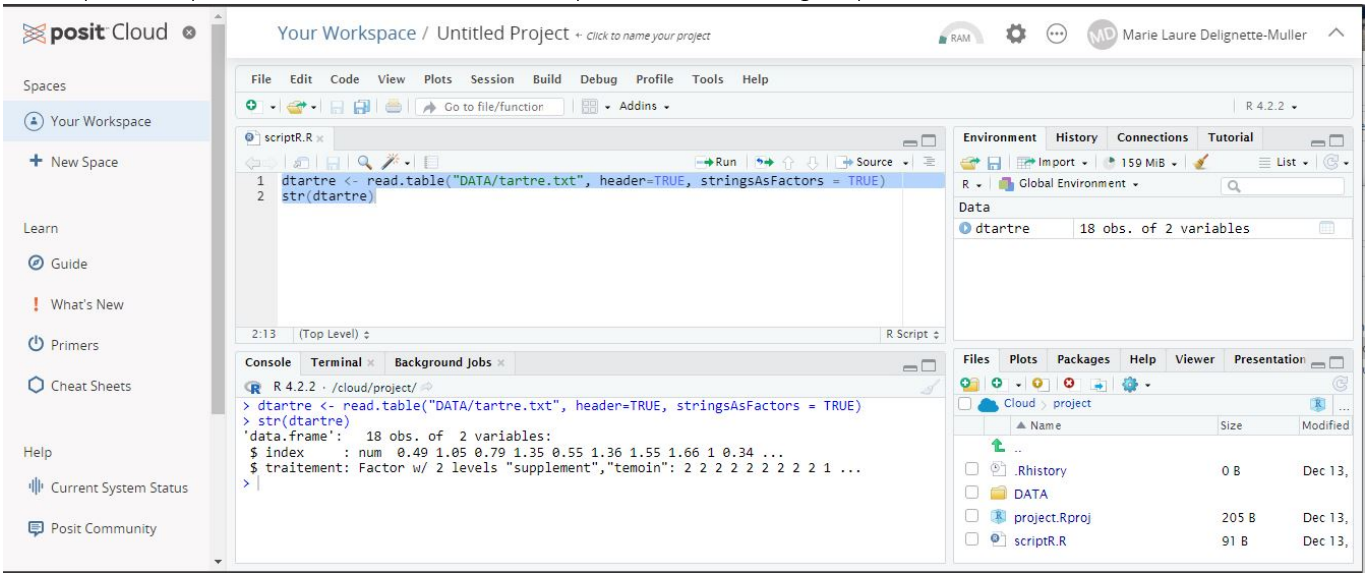
Après un petit temps d'attente il va vous ouvrir Rstudio à distance et vous pourrez travailler comme si vous aviez Rstudio et R sur votre ordinateur. Il faudra juste faire une petite manipulation pour charger dans votre projet les jeux de données (par exemple depuis VetAgrotice pour les séances de TD). Au préalable il faudra charger quelque part sur votre ordinateur ou tablette le .zip contenant le fichier zip contenant ces jeux de données (ou le jeu de données si vous n'en ajouter qu'un à votre projet). Puis dans la fenêtre de Rstudio en bas à droite, cliquez sur **Upload**. Vous obtenez le cadre ci-dessous.



Il suffit alors de spécifier le nom du fichier .zip que vous souhaitez charger en cliquant sur parcourir.



Vous verrez alors apparaître ce dossier correspondant dans la fenêtre en bas à droite et vous pourrez, comme ci-dessous, importer des fichiers à partir de ce dossier en indiquant son nom complet (dans cet exemple DATA/tartre.txt) ou en spécifiant ce dossier (ici DATA) comme votre répertoire courant (cf. partie 1.3 de ce guide).



1.10 Présentation du document

Dans la suite de ce document, vous trouverez quelques exemples d'utilisation des méthodes classiques utilisées dans les cas les plus simples. Ils sont ordonnées en fonction du type de jeu de données à analyser.

Vous pouvez copier / coller puis modifier les bouts de code de ce guide pour vous simplifier la tâche lors des TD. Tout ce qui figure après le signe # constitue des commentaires explicatifs, non interprétés par R, et donc n'est pas à copier.

Le code peut avoir l'air un peu compliqué au début, mais vous verrez que pour faire des analyses répétitives, c'est beaucoup plus performant que d'utiliser des menus avec boutons. Dans ce guide, par souci de simplicité, une seule façon de procéder est généralement décrite pour réaliser une tâche donnée, alors que plusieurs sont souvent possibles. Si vous voulez en savoir plus sur une fonction, utilisez l'aide en ligne : elle est accessible pour chaque fonction de R en tapant `help(nomdelafonction)` ou de façon équivalente `?nomdelafonction`.

ATTENTION, dans ce guide, pour un jeu de données et une question posée, divers tests sont souvent présentés (paramétrique et non paramétrique notamment) sans que le choix du bon test (le plus puissant parmi ceux dont les conditions d'utilisation sont respectées par les données) ne soit effectué. Le propos de ce guide n'est pas d'expliquer comment faire le choix du test le plus approprié (voir le polycopié de cours pour cela : https://biostatistique.vetagro-sup.fr/polyM1.pdf), mais de vous donner accès à tous les codes potentiellement utiles pour faire ce choix et réaliser les tests. CE MANUEL PRESENTE DONC DES ANALYSES ADAPTEES AUX JEUX DE DONNEES PRIS EN EXEMPLE ET D'AUTRES NON ADAPTEES !

2 Avec une série d'observations

2.1 Variable quantitative

Il convient tout d'abord d'importer les données. Il s'agit ici d'une seule variable observée nommée `temps`.

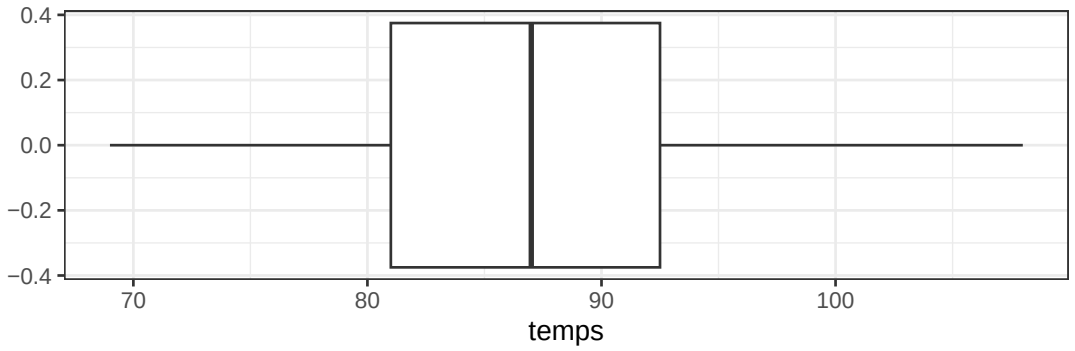
```
d <- read.table("reflexe.txt", header=TRUE, stringsAsFactors = TRUE)
d

##    temps
```

```
## 1    108
## 2    100
## 3     78
## 4     81
## 5     69
## 6     87
## 7     88
## 8     81
## 9     87
## 10    94
```

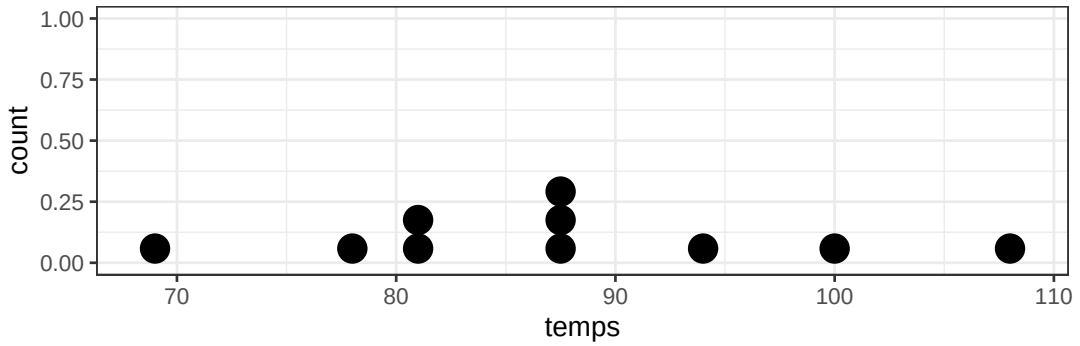
2.1.1 Examen de la distribution

On représente classiquement ce type de données par un diagramme en boîte. Lorsque l'effectif est très petit il est préférable de représenter directement tous les points (cf. deuxième graphe ci-dessous).

```
ggplot(d, aes(x = temps)) + geom_boxplot()
```

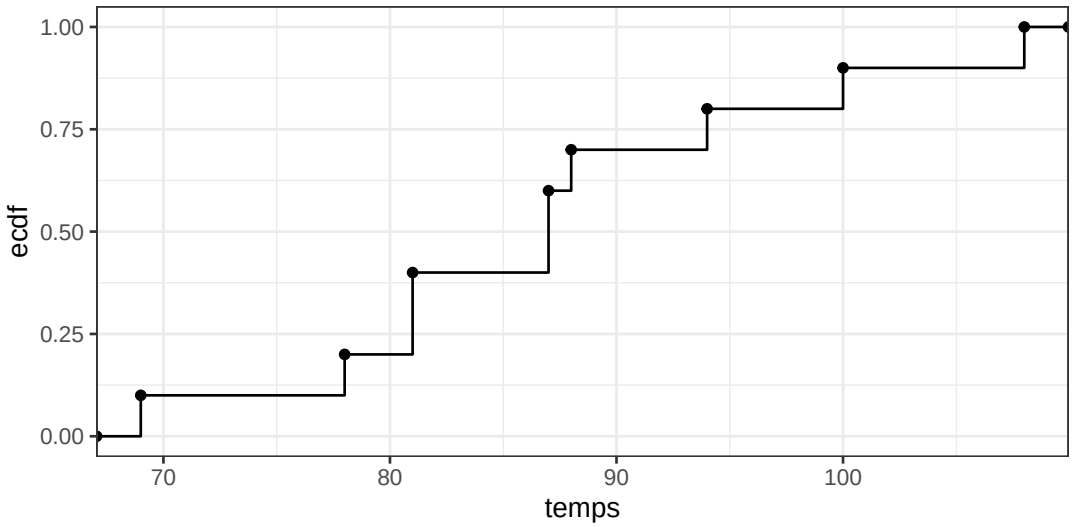



```
ggplot(d, aes(x = temps)) + geom_dotplot()
```



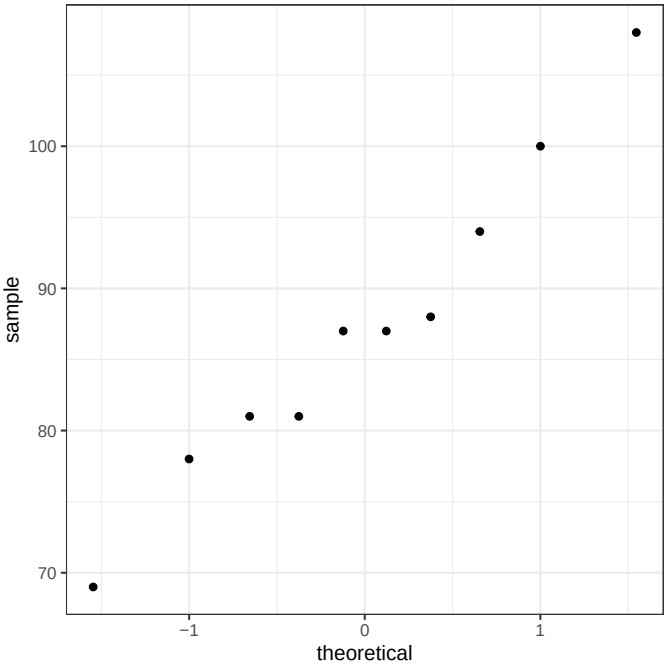
Lorsque l'effectif est très grand il est possible de représenter la distribution par un histogramme à l'aide de la fonction `geom_histogram()`. Ici l'effectif est trop petit mais on peut par contre obtenir facilement la courbe de fréquences cumulées à l'aide de la commande suivante :

```
ggplot(d, aes(x = temps)) + stat_ecdf(geom = "step") + stat_ecdf(geom = "point")
```



Pour juger de la normalité de la distribution, il est souvent utile de représenter les quantiles observés en fonction des quantiles de la loi normale :

```
ggplot(d, aes(sample = temps)) + geom_qq()
```



Pour arriver à bien juger de la normalité, donc de l'alignement (à peu près) des points, mieux vaut rearder ce diagramme quantile quantile sur un format carré. Un test de normalité peut être utilisé en complément, **tout en gardant à l'esprit qu'un tel test manque de puissance pour les petits échantillons et qu'en aucun cas ce test ne permet pas de montrer la normalité d'une distribution**.

```
shapiro.test(d$temps)

##
##  Shapiro-Wilk normality test
##
## data:  d$temps
## W = 1, p-value = 0.9
```

2.1.2 Description de la distribution (moyenne, écart type, quantiles)

On peut bien entendu facilement calculer les paramètres décrivant la distribution observée :

— la moyenne à l'aide de la fonction `mean()`,

```
mean(d$temps)
## [1] 87.3
```

— la variance estimée à l'aide de la fonction `var()` et l'écart type estimé à l'aide de la fonction `sd()`,

```
var(d$temps)
## [1] 126
sd(d$temps)
## [1] 11.2
```

— la médiane à l'aide de la fonction `median()` et plus généralement n'importe quel quantile à l'aide de la fonction `quantile()`, en indiquant en argument les probabilités auxquelles on veut calculer ces quantiles dans le vecteur `probs`.

```
median(d$temps)
## [1] 87
quantile(d$temps, probs = c(1/4,1/2,3/4))
##  25%  50%  75%
## 81.0 87.0 92.5
```

2.1.3 Test de conformité : comparaison de la moyenne observée à une moyenne théorique

Le test T de conformité à une moyenne théorique est réalisé de la façon suivante, en affectant l'argument `mu` de la fonction `t.test()` à la moyenne théorique :

```
t.test(d$temps, mu = 83.2)

##
##  One Sample t-test
##
## data:  d$temps
## t = 1, df = 9, p-value = 0.3
## alternative hypothesis: true mean is not equal to 83.2
## 95 percent confidence interval:
##  79.3 95.3
## sample estimates:
## mean of x
##      87.3
```

2.1.4 Intervalle de confiance autour d'une moyenne

Pour calculer un intervalle de confiance autour d'une moyenne observée sans réaliser de test, on utilise la même fonction que pour réaliser le test de conformité de Student, sans spécifier l'argument `mu` et on récupère uniquement la sortie de la fonction qui correspond à l'intervalle de confiance (`$conf.int` après le nom de la fonction). On peut modifier le seuil de confiance à l'aide de l'argument `conf.level` par défaut fixé à 0.95.

```
t.test(d$temps, conf.level = 0.95)$conf.int

## [1] 79.3 95.3
## attr(,"conf.level")
## [1] 0.95
```

Cette fonction permet occasionnellement de calculer des intervalles de confiance unilatéraux. L'argument `alternative` doit être fixé à

— `"less"` pour obtenir un intervalle de confiance du type $[-\infty,a]$
— `"greater"` pour obtenir un intervalle de confiance du type $[a,+\infty]$
— `"two.sided"` pour obtenir un intervalle de confiance du type [a,b] (option par défaut)

2.2 Variable qualitative

2.2.1 Calcul des effectifs observés dans chaque classe et examen de la distribution

Suivant que vous disposez (1) ou non (2) des données brutes, la construction de la table des effectifs sera réalisée différemment :

1. Lorsque les **données sont disponibles sous forme brute** (par exemple observations de la variable qualitative pour chaque individu), il est nécessaire au préalable d'utiliser la fonction `table()` sur la variable qualitative afin d'obtenir de façon automatique les effectifs observés comme ci-dessous (exemple reportant le type de lien entre les animaux et leur maître)) :

```r
d <- read.table("animfamibrut.txt", header = TRUE, stringsAsFactors = TRUE)
str(d) # pour voir ce qu'il y a dans le jeu de données
```

```
## 'data.frame': 1100 obs. of  2 variables:
##  $ espece: Factor w/ 2 levels "chat","chien": 2 2 2 2 2 2 2 2 2 2 ...
##  $ lien  : Factor w/ 4 levels "ami","enfant",..: 2 2 2 2 2 2 2 2 2 2 ...
```

```r
t <- table(d$lien)
t
```

```
##
##     ami  enfant famille  invite
##     204     208     638      50
```

2. Si l'on **travaille directement à partir des effectifs**, par exemple tels que récupérés dans un article, on peut créer cette table manuellement :

```r
t <- as.table(c(204, 208, 638, 50))
row.names(t) <- c("ami", "enfant", "famille", "invite")
t
```

```
##     ami  enfant famille  invite
##     204     208     638      50
```

Une fois la table des effectifs `t` obtenue d'une façon ou d'une autre, on peut ensuite, si besoin, calculer les fréquences observées dans chaque classe à partir de celle-ci :

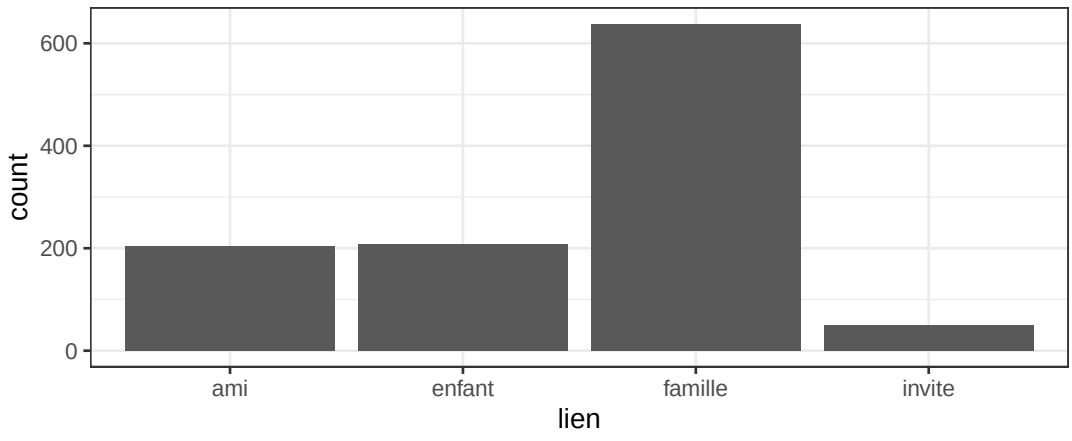
```r
tf <- prop.table(t)
tf
```

```
##     ami  enfant famille  invite
##  0.1855  0.1891  0.5800  0.0455
```

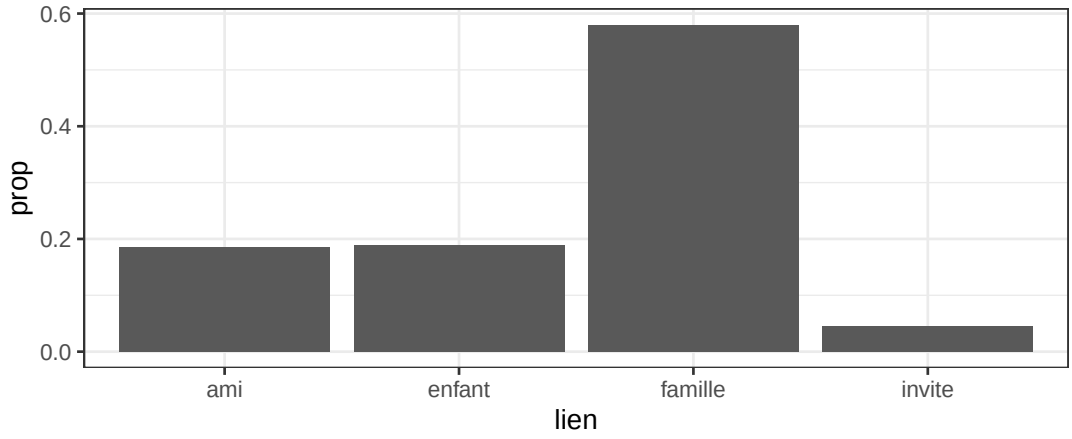
2.2.2 Visualisation de la distribution

On peut représenter la distribution avec `ggplot2` à partir des données brutes si on en dispose (cas 1 seulement), soit à l'aide de la fonction `barplot()` si on a que les tables `t` ou `tf` (cas 2).

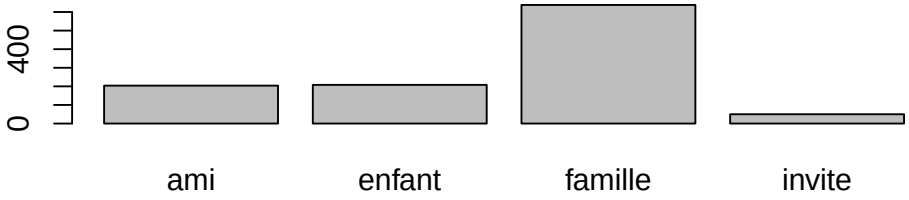
```r
## Cas 1
# Diagramme en bâtons en effectifs à partir des données brutes
ggplot(d, aes(x = lien)) + geom_bar()
```



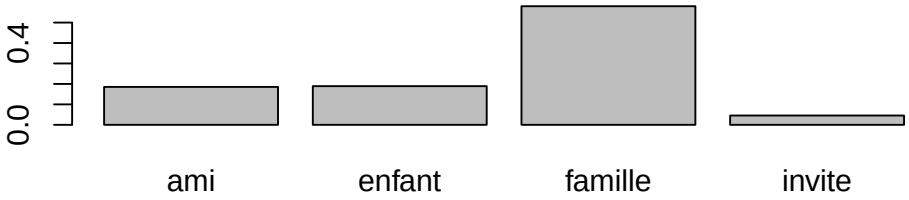

```r
# Diagramme en bâtons en proportions à partir des données brutes
ggplot(d, aes(x = lien)) + geom_bar(aes(y = after_stat(prop), group = 1))
```




```
# Cas 2
# Diagramme en bâtons à partir de la table des effectifs
barplot(t)
```




```
# Diagramme en bâtons à partir de la tables des proportions
barplot(tf)
```



2.2.3 Test du χ^2 d'ajustement

Pour réaliser le test du χ^2 d'ajustement, on doit coder les effectifs observés dans un vecteur ou une table que l'on nommera `nobs` comme ci-dessous (cf. section 2.2.1 pour les explications relatives à la création de tables) et les proportions théoriques dans chaque classe dans un vecteur que l'on nommera `ptheo` dans cet exemple) :

```
## Définition des effectifs observés selon 3 façons possibles
# 1/ nobs défini comme une table créée à partir des données brutes
d1 <- read.table("insemination.txt", header = TRUE, stringsAsFactors = TRUE)
nobs <- table(d1$resultat)
# 2/ ou nobs défini comme une table créée manuellement
```

```r
nobs <- as.table(c(78, 275))
row.names(nobs) <- c("echec", "succes")
# 3/ ou nobs défini comme un simple vecteur
nobs <- c(78, 275)

## Définition des proportions théoriques
ptheo <- c(0.25, 0.75)
```

Le test du χ^2 d'ajustement est ensuite réalisé en mettant comme premier argument de la fonction `chisq.test()` les effectifs observés, et en affectant l'argument p aux proportions théoriques :

```r
khi2 <- chisq.test(nobs, p = ptheo)
khi2

##
##  Chi-squared test for given probabilities
##
## data:  nobs
## X-squared = 2, df = 1, p-value = 0.2
```

Les effectifs théoriques calculés automatiquement à partir des proportions théoriques peuvent être affichés afin de vérifier les conditions d'utilisation du test du χ^2 (effectifs théoriques tous supérieurs à 5) :

```r
print(khi2$expected)

## [1]  88.2 264.8
```

2.2.4 Test exact (utilisant la loi binomiale) de comparaison d'une fréquence observée à une fréquence théorique

La fonction `binom.test()` permet de réaliser ce test. Elle doit être paramétrée avec
- comme 1er argument le nombre de réalisation de l'évènement étudié,
- comme 2ème argument le nombre total de tirages,
- comme 3ème argument p, la fréquence théorique à laquelle on veut comparer l'observée,
- et comme 4ème argument le type d'hypothèse alternative testée (par défaut fixé à `"two.sided"`, c'est-à-dire bilatéral).

Voici un exemple sur lequel on peut utiliser cette fonction : un traitement préventif d'une maladie a été testée sur une population animale dont on sait que sans traitement environ 4% des animaux sont touchés par cette maladie. Sur 100 animaux traités, 2 sont atteints par la maladie. Ce pourcentage observé est-il significativement différent du 4% observé habituellement sans traitement ? Pour résoudre cet exemple on utilisera le code suivant :

```r
binom.test(2, 100, p = 0.04, alternative = "two.sided",conf.level = 0.95)

##
##  Exact binomial test
##
## data:  2 and 100
## number of successes = 2, number of trials = 100, p-value = 0.4
## alternative hypothesis: true probability of success is not equal to 0.04
## 95 percent confidence interval:
##  0.00243 0.07038
## sample estimates:
## probability of success
##                   0.02
```

2.2.5 Intervalle de confiance autour d'une fréquence

Pour calculer un intervalle de confiance autour d'une fréquence, on utilise la même fonction `binom.test()` que précédemment (cf. section 2.2.4 pour la définition des arguments), sans spécifier l'argument p si on n'a pas de valeur théorique à laquelle on veut comparer la fréquence observée. On récupère alors uniquement la sortie de la fonction qui correspond à l'intervalle de confiance (sortie (`$conf.int` après le nom de la fonction). On peut modifier le seuil de confiance à l'aide de l'argument `conf.level` par défaut fixé à 0.95.

Cette fonction permet si nécessaire de calculer des intervalles de confiance unilatéraux. L'argument `alternative` doit être fixé à

— "less" pour obtenir un intervalle de confiance du type [0,a]
— "greater" pour obtenir un intervalle de confiance du type [a,1]
— "two.sided" pour obtenir un intervalle de confiance du type [a,b] (option par défaut)

Voici un exemple de calcul d'intervalle de confiance du type [0, a] : en comparant deux méthodes diagnostiques on observe 3 discordances sur 100. On veut calculer le seuil au dessous duquel on est sûr à 95% que se trouve la proportion théorique de discordances entre les deux méthodes (intervalle de confiance unilatéral [0 ; seuil]). Voici comment on répond à cette question avec R :

```
binom.test(3, 100, alternative = "less", conf.level = 0.95)$conf.int

## [1] 0.0000 0.0757
## attr(,"conf.level")
## [1] 0.95
```

Si l'on souhaite simplement obtenir l'intervalle de confiance bilatéral sur cette proportionde discordances on tapera :

```
binom.test(3, 100, alternative = "two.sided",conf.level = 0.95)$conf.int

## [1] 0.00623 0.08518
## attr(,"conf.level")
## [1] 0.95
```

Voici un exemple de calcul d'intervalle de confiance du type [a, 1] : On évalue la sensibilité d'un test diagnostique : sur 100 animaux malades, le test en détecte 96. On veut calculer le seuil au dessus duquel on est sûr à 95% que se trouve la sensibilité théorique du test diagnostique (intervalle de confiance unilatéral [seuil ; 1]). Voici comment on répond à cette question avec R :

```
binom.test(96, 100, alternative = "greater",conf.level = 0.95)$conf.int

## [1] 0.911 1.000
## attr(,"conf.level")
## [1] 0.95
```

3 Avec deux séries indépendantes d'observations

3.1 Variable quantitative

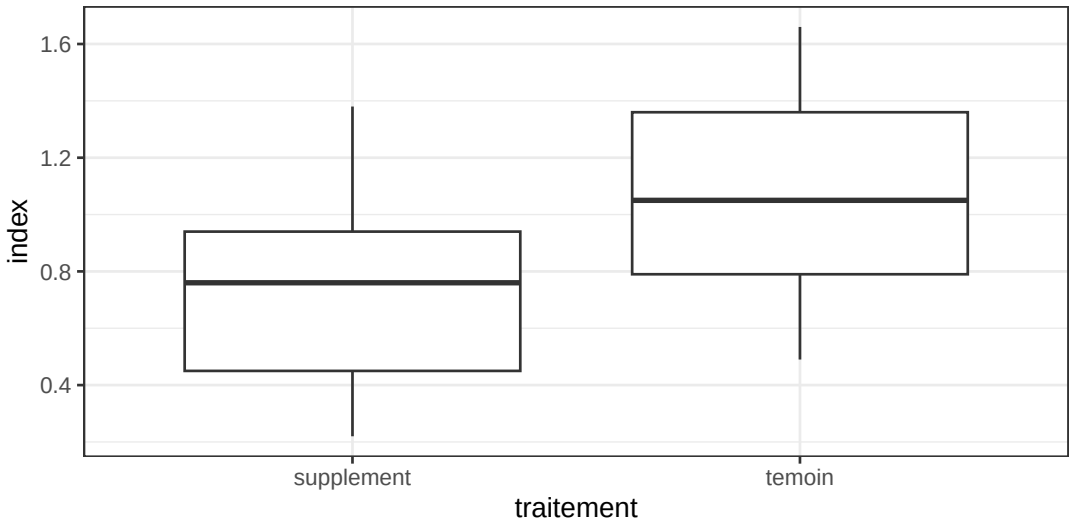
Les données peuvent être importées sous la forme d'un tableau contenant deux colonnes, une colonne avec les valeurs observées de la variable quantitative (nommée index dans cet exemple) et une colonne avec le facteur codant pour le groupe d'appartenance de chaque observation (nommé traitement dans cet exemple). L'ordre de présentation des colonnes et des lignes n'a pas d'importance

```
d2 <- read.table("tartre.txt", header = TRUE, stringsAsFactors = TRUE)
str(d2)

## 'data.frame': 18 obs. of  2 variables:
##  $ index     : num  0.49 1.05 0.79 1.35 0.55 1.36 1.55 1.66 1 0.34 ...
##  $ traitement: Factor w/ 2 levels "supplement","temoin": 2 2 2 2 2 2 2 2 2 1 ...
```

3.1.1 Examen des distributions

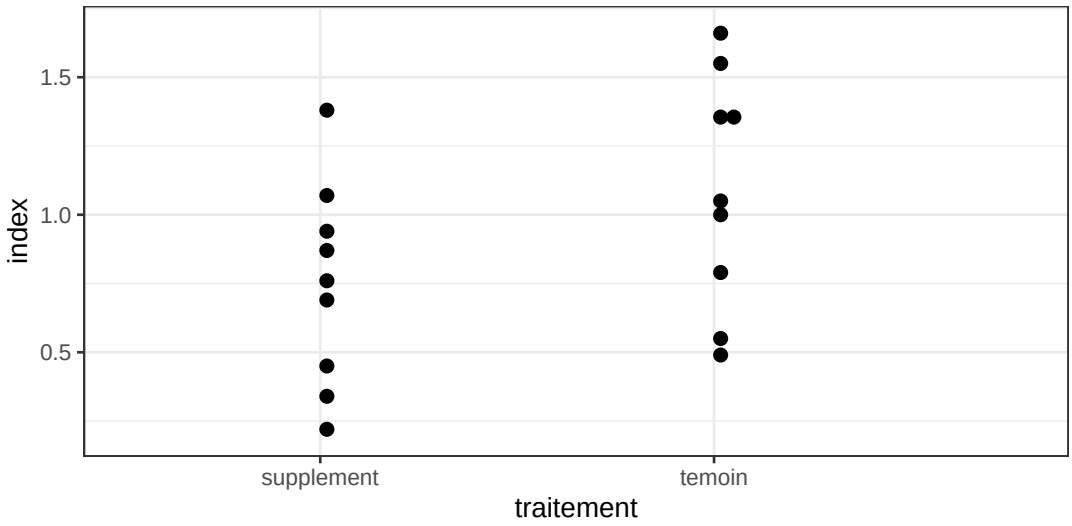
Les distributions sont le plus souvent représentées sous forme de deux diagrammes en boîte.

```
ggplot(d2, aes(x = traitement, y = index)) + geom_boxplot()
```



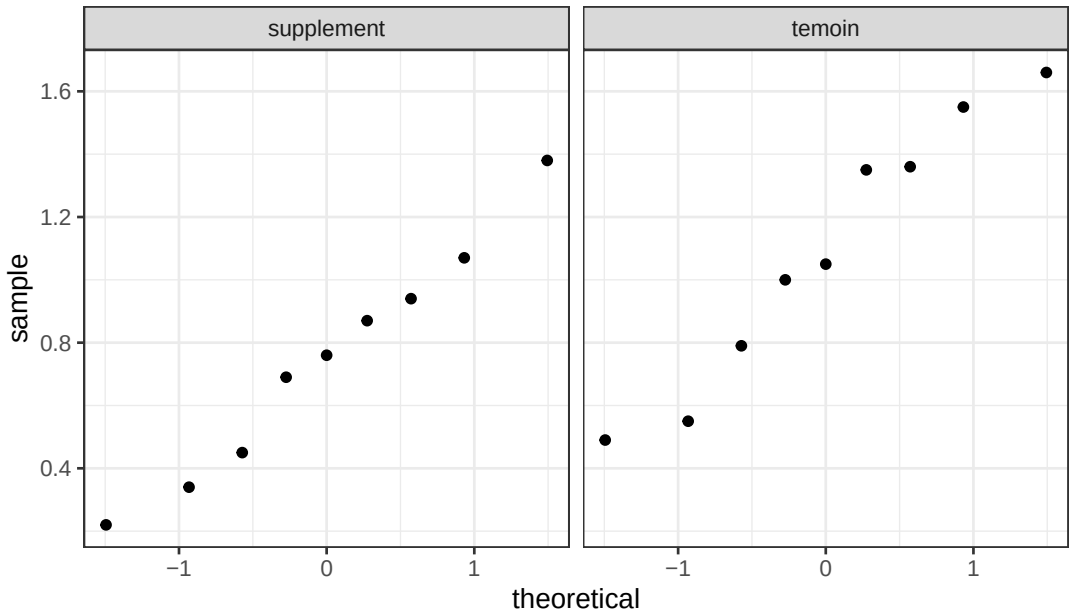
Lorsque les effectifs sont très petits, et/ou que la variable est discrète (donc avec une probabilité élevée qu'il y ait des ex-aequos) il est préférable de représenter directement tous les points observés dans chaque groupe par exemple comme ci-dessous.

```
ggplot(d2, aes(x = traitement, y = index)) + geom_dotplot(binaxis = "y")
```



Il est possible de tracer les graphes des quantiles associés aux quantiles de la loi normale pour chacun des deux groupes par exemple comme ci-dessous.

```
ggplot(d2, aes(sample = index)) + geom_qq() + facet_wrap(~ traitement)
```



3.1.2 Tests de comparaison de 2 moyennes (ou tendances centrales)

Suivant la forme et la dispersion des distributions et la taille des effectifs, on pourra réaliser un des trois tests suivants de comparaison de deux moyennes ou tendances centrales :

— soit le **test T avec variances égales (Student)**

```
t.test(d2$index ~ d2$traitement, var.equal = TRUE)
##
##  Two Sample t-test
##
## data:  d2$index by d2$traitement
## t = -2, df = 16, p-value = 0.09
## alternative hypothesis: true difference in means between group supplement and group temoin is not
## 95 percent confidence interval:
##  -0.7389  0.0544
## sample estimates:
## mean in group supplement     mean in group temoin
##                    0.747                    1.089
```

— soit le **test T avec variances inégales (test de Welch dit aussi Student avec variances inégales)**

```
t.test(d2$index ~ d2$traitement, var.equal = FALSE)
##
##  Welch Two Sample t-test
##
## data:  d2$index by d2$traitement
## t = -2, df = 16, p-value = 0.09
## alternative hypothesis: true difference in means between group supplement and group temoin is not
## 95 percent confidence interval:
##  -0.739  0.055
## sample estimates:
## mean in group supplement     mean in group temoin
##                    0.747                    1.089
```

— soit le **test non paramétrique de la somme des rangs (Mann-Whitney-Wilcoxon)**

```
wilcox.test(d2$index ~ d2$traitement)
##
##  Wilcoxon rank sum exact test
##
## data:  d2$index by d2$traitement
## W = 22, p-value = 0.1
## alternative hypothesis: true location shift is not equal to 0
# comme on ne peut plus parler strictement de comparaison de moyennes
# dans ce cas il peut être intéressant de calculer les médianes par groupe
tapply(d2$index, d2$traitement, median)
```

```
## supplement     temoin
##        0.76       1.05
```

3.1.3 Intervalle de confiance autour de la différence entre deux moyennes

Lorsqu'un test paramétrique de comparaison de moyennes est réalisé avec la fonction `t.test()` (cf. paragraphe précédent), il est accompagné du calcul de l'intervalle de confiance autour de la différence entre les deux moyennes comparées (sortie `conf.int`) affiché en dernier dans la sortie de la fonction.

Dans certains cas (par ex. la réalisation d'un test d'équivalence), on s'intéressera directement et uniquement à ce calcul d'intervalle de confiance, que l'on pourra d'ailleurs extraire du résultat de la fonction `t.test()` de la façon suivante dans l'exemple d'une statistique de Student supposant les variances égales :

```
t.test(d2$index ~ d2$traitement, var.equal = TRUE)$conf.int
```

```
## [1] -0.7389  0.0544
## attr(,"conf.level")
## [1] 0.95
```

3.1.4 Tests de comparaison de 2 variances

Si les distributions peuvent être supposées normales, et que les variances semblent différentes, un test de comparaison de variances peut être réalisé de la façon suivante :

```
var.test(d2$index ~ d2$traitement)
```

```
##
##  F test to compare two variances
##
## data:  d2$index by d2$traitement
## F = 0.8, num df = 8, denom df = 8, p-value = 0.7
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.173 3.391
## sample estimates:
## ratio of variances
##              0.765
```

Rappelons que ce test ne présente pas beaucoup d'intérêt pour juger s'il est raisonnable de supposer les variances égales dans un test de comparaison de moyennes, notamment du fait de son manque de puissance pour de faibles effectifs. Dans ce but on se basera plus souvent sur une représentation graphique des distributions. **Ce test est par contre utile pour démontrer l'existence d'un différence entre variances** lorsque celle-ci est d'intérêt biologique.

3.2 Variable qualitative

3.2.1 Résumé des données sous la forme d'une table de contingence

Suivant que l'on dispose (1) ou non (2) des données brutes, la construction de la table de contingence sera réalisée différemment :

1. **Lorsque l'on dispose des données brutes** (jeu de données comprenant les deux colonnes correspondant aux deux variables qualitatives observées) il est facile de créer la table de contingence associée en utilisant la fonction `table()`. Voici un exemple sur un jeu de données correspondant à l'étude de la nature du lien entre les animaux et leur maître :

```
d4 <- read.table("animfamibrut.txt", header = TRUE, stringsAsFactors = TRUE)
str(d4)
```

```
## 'data.frame': 1100 obs. of  2 variables:
##  $ espece: Factor w/ 2 levels "chat","chien": 2 2 2 2 2 2 2 2 2 2 ...
##  $ lien  : Factor w/ 4 levels "ami","enfant",..: 2 2 2 2 2 2 2 2 2 2 ...
```

```
# Etape préliminaire de construction de la table de contingence
tlien <- table(d4$espece, d4$lien)
tlien
```

```
##
##         ami enfant famille invite
##   chat  128     86     342     44
##   chien  76    122     296      6
```

Il est parfois opportun dans un tel cas de réordonner les modalités des deux facteurs (variables qualitatives), dans l'ordre qui nous convient (ici en suivant un ordre décroissant pour l'attachement à l'animal) comme ci-dessous :

```
d4$lien <- factor(d4$lien, levels = c("enfant", "famille", "ami", "invite"))
tlien <- table(d4$espece, d4$lien)
tlien
```

```
##
##         enfant famille ami invite
##   chat      86     342 128     44
##   chien    122     296  76      6
```

2. **Si les données dont on dispose sont déjà sous forme d'une table de contingence**, il convient de saisir cette table de contingence dans R avec ses intitulés en utilisant les fonctions `matrix()` puis `as.table()`. Voici un exemple sur un jeu de données correspondant à l'étude de la nature du lien entre les animaux et leur maître :

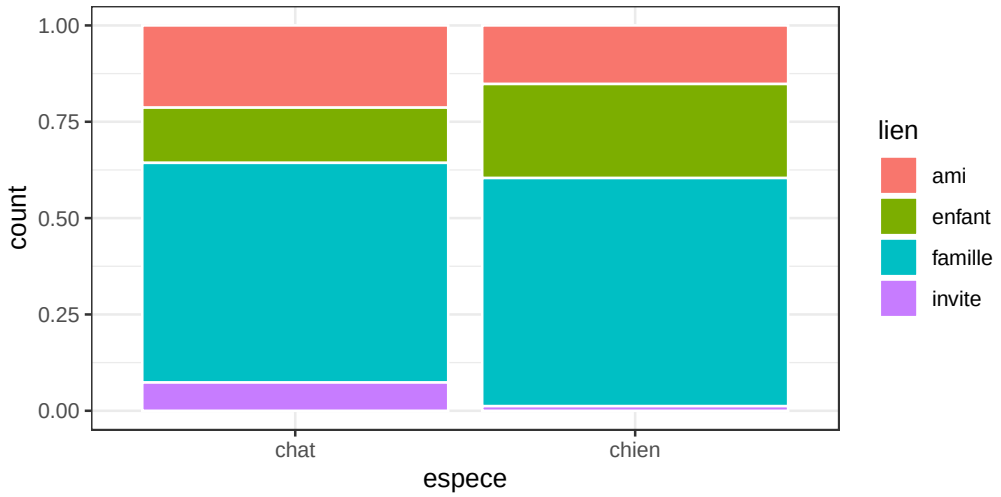
```
m <- matrix(c(86, 342, 128, 44, 122, 296, 76, 6), nrow = 2, byrow = TRUE)
rownames(m) <- c("chat", "chien")
colnames(m) <- c("enfant","famille","ami","invite")
tlien <- as.table(m)
tlien
```

```
##       enfant famille ami invite
## chat      86     342 128     44
## chien    122     296  76      6
```

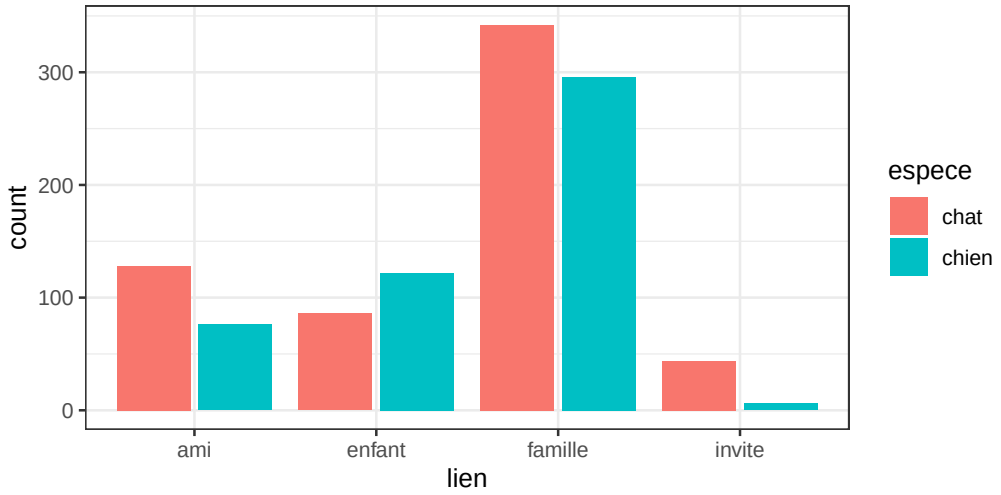
3.2.2 Représentation graphique de données

1. **Si l'on dispose des données brutes**, on peut utiliser les fonctions de `ggplot2` pour faire des diagrammes en bâtons accolés ou un diagramme en barres, comme ci-dessous.

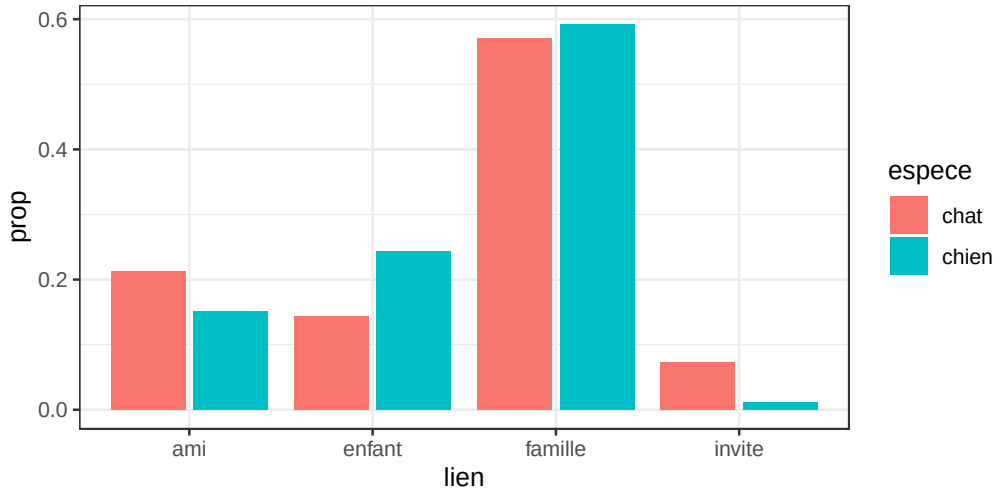
```
# Diagramme en barres
ggplot(d, aes(x = espece, fill = lien)) + geom_bar(position = "fill", col = "white")
```




```
# Diagrammes en bâtons accolés en effectifs
ggplot(d, aes(x = lien, fill = espece)) + geom_bar(position = "dodge2")
```

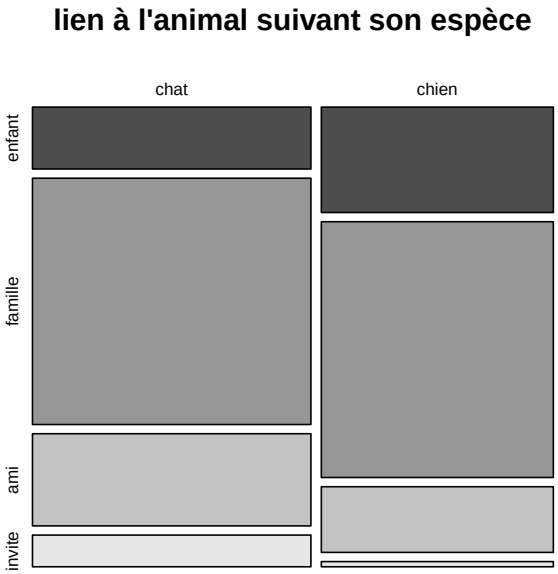



```
# Diagrammes en bâtons accolés en proportions
ggplot(d, aes(x = lien, fill = espece)) +
  geom_bar(aes(y = after_stat(prop), group = espece), position = "dodge2")
```



2. **A partir de la table de contingence**, et notamment si on ne dispose que de celle-ci, on peut utiliser la fonction générique `plot()` de R qui propose un diagramme en barre (cf. ci-dessous et voir section 3.2.1 pour le codage au préalable de la table de contingence).

```
plot(tlien, color = TRUE, main = "lien à l'animal suivant son espèce")
```



3.2.3 Réalisation du test du χ^2

1. **Si l'on dispose des données brutes** on peut réaliser le test du χ^2 d'indépendance directement à partir de celles-ci comme dans le code suivant.

```
khi2 <- chisq.test(d4$espece, d4$lien)
khi2

##
##  Pearson's Chi-squared test
##
## data:  d4$espece and d4$lien
## X-squared = 43, df = 3, p-value = 3e-09
```

2. On peut aussi réaliser le test du χ^2 d'indépendance **à partir de la table de contingence** créée préalablement (voir section 3.2.1 pour le codage au préalable de la table de contingence), comme ci-dessous.

```
khi2 <- chisq.test(tlien)
khi2

##
##  Pearson's Chi-squared test
##
## data:  tlien
## X-squared = 43, df = 3, p-value = 3e-09
```

Le stockage du résultat de la fonction dans l'objet nommé ici `khi2` permet de récupérer ensuite quelques résultats non affichés par défaut, comme les effectifs théoriques (pour vérifier qu'ils sont tous supérieurs à 5) :

```
print(khi2$expected)

##       enfant famille   ami invite
## chat   113.5     348 111.3   27.3
## chien   94.5     290  92.7   22.7
```

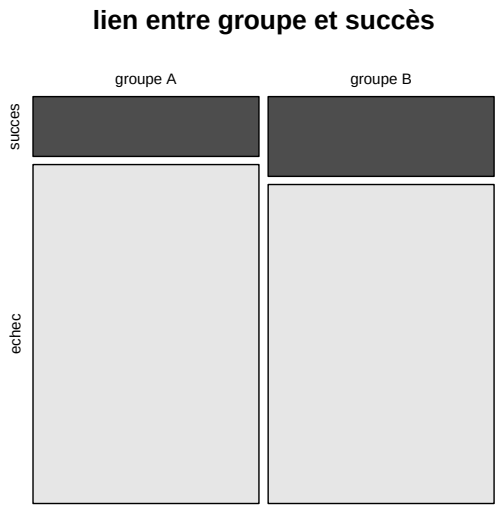
3.2.4 Cas particulier du test de comparaison de deux fréquences observées et intervalle de confiance autour de la différence entre deux fréquences

Dans le cas particulier de la comparaison de deux fréquences observées, on peut bien sûr utiliser le test du χ^2 comme décrit précédemment, mais on dispose aussi d'une fonction spécifique `prop.test()` qui réalise le test du χ^2 d'indépendance

et calcule l'intervalle de confiance autour de la différence entre les deux fréquences. Cette fonction peut être appelée avec comme argument la **table de contingence créée préalablement soit à partir de données brutes soit directement à partir des données résumées** (voir section 3.2.1 pour le codage au préalable de la table de contingence).

Voici un exemple de codage correspondant à la comparaison de 2 fréquences de 15% et 20% observées chacune sur un échantillon de 200 animaux : Dans cet exemple une nouvelle table de contingence a été codée à partir des données résumées, mais une table construite à partir des données brutes avec la fonction `table()`, créée comme sur l'exemple précédent (section 3.2.1), est bien sûr utilisable.

```
t <- as.table(matrix(c(30, 170, 40, 160), nrow = 2, byrow = TRUE))
rownames(t) <- c("groupe A", "groupe B")
colnames(t) <- c("succes","echec")
t

##          succes echec
## groupe A     30   170
## groupe B     40   160

plot(t, col = TRUE, main = "lien entre groupe et succès")
```




```
prop.test(t)

##
##  2-sample test for equality of proportions with continuity correction
##
## data:  t
## X-squared = 1, df = 1, p-value = 0.2
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.1293  0.0293
## sample estimates:
## prop 1 prop 2
##   0.15   0.20
```

Attention, avant utilisation de l'intervalle de confiance sur la différence entre les 2 fréquences il est prudent de vérifier que les deux fréquences estimées, notées `prop 1` **et** `prop 2` **dans la sortie de la fonction, correspondent bien aux deux fréquences dont on souhaite faire la différence.** En effet, de façon peu intuitive, les **fréquences sont calculées sur les lignes de la table donnée en argument** de la fonction et non sur ses colonnes.

Enfin, toujours dans le cas de la comparaison de deux fréquences, un calcul exact de la p-value peut être réalisé à l'aide du test de Fisher, en appelant la fonction `fisher.test()` avec comme argument la table de contingence. Ce test sera en particulier à utiliser lorsque les effectifs sont faibles. Voici son application sur les données précédentes :

```
fisher.test(t)

##
##  Fisher's Exact Test for Count Data
##
## data:  t
## p-value = 0.2
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.404 1.225
## sample estimates:
## odds ratio
##      0.707
```

4 Avec deux séries dépendantes (ou appariées) d'observations

4.1 Variable quantitative

Lorsque l'on souhaite comparer deux moyennes ou tendances centrales sur deux séries appariées, on peut importer les données sous la forme d'un tableau avec deux colonnes correspondant aux observations de la variable quantitative sur chacune des deux séries (colonnes nommées gestation1 et gestation2 dans cet exemple) afin de conserver l'information sur l'appariement des séries. On code ainsi les données de la même façon que s'il s'agissait de deux variables différentes (colonnes) mesurées sur les mêmes unités d'observation (lignes) :

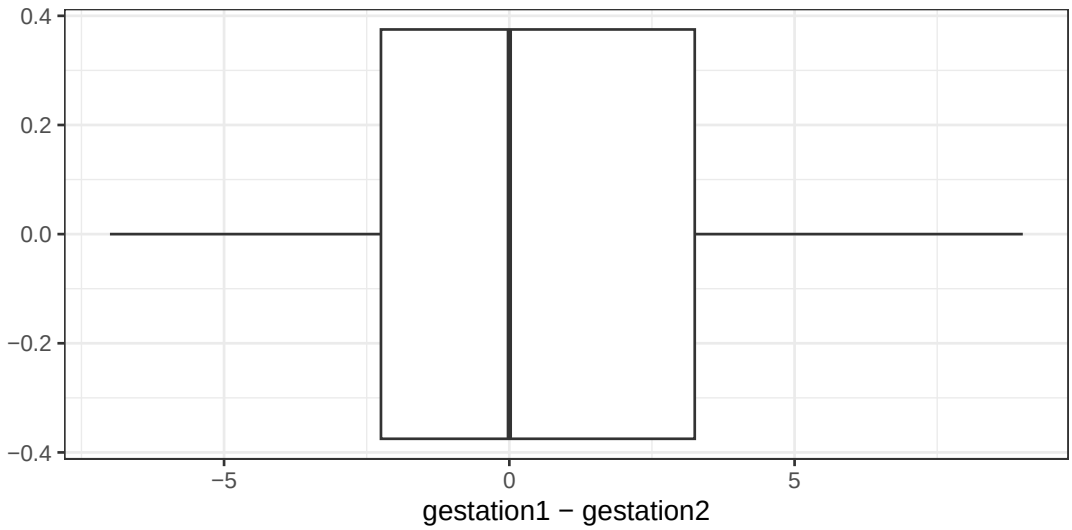
```
d5 <- read.table("gestation.txt",  header = TRUE, stringsAsFactors = TRUE)
str(d5)

## 'data.frame': 12 obs. of  2 variables:
##  $ gestation1: int  289 282 292 290 292 282 296 287 297 286 ...
##  $ gestation2: int  286 288 290 297 292 284 287 287 289 286 ...
```

ATTENTION, il est important de noter que le codage des données dans ce cas est différent de celui utilisé pour des séries indépendantes (avec des séries indépendantes dans un exemple de ce type, on aurait dans le jeu de données une colonne correspondant au numéro de gestation (gestation1 ou gestation2) et une colonne correspondant à la durée de gestation), codage qui ne permet pas de coder un appariement.

4.1.1 Visualisation des données appariées et examen de la distribution des différences

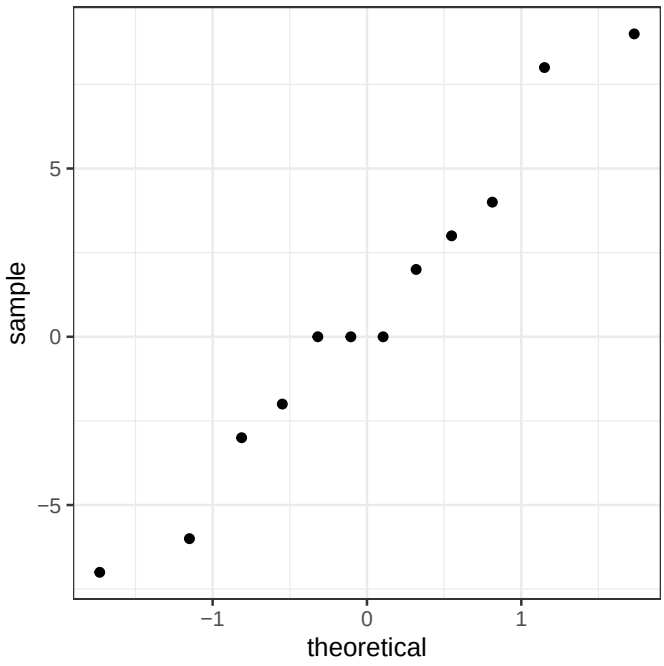
La distribution des différences entre les deux séries peut être représentée soit sous forme de diagramme en boîte comme ci-dessous, soit à l'aide des fonctions `geo_histogram()` dans le cas de grands effectifs ou de `geom_doplot()` dans le cas de très petits effectifs (cf. section 2.1.1).

```
ggplot(d5, aes(x = gestation1 - gestation2)) + geom_boxplot()
```



Comme décrit précédemment dans le cas d'une série d'observations, un diagramme quantile-quantile peut aussi être réalisé pour aider à juger de la normalité de la distribution des différences :

```r
ggplot(d5, aes(sample = gestation1 - gestation2)) + geom_qq()
```



4.1.2 Test de comparaison de moyennes ou tendances centrales

Suivant que l'on pourra ou non supposer la distribution normale et suivant l'effectif, l'un des deux tests suivants pourra être réalisé :

— soit le **test T de comparaison de 2 moyennes sur séries appariées (Student des séries appariées)**

```r
t.test(d5$gestation1, d5$gestation2, paired = TRUE)
##
##  Paired t-test
##
## data:  d5$gestation1 and d5$gestation2
## t = 0.5, df = 11, p-value = 0.6
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -2.46  3.80
## sample estimates:
## mean difference
##           0.667
```

— soit le **test non paramétrique de Wilcoxon des rangs signés**

```r
wilcox.test(d5$gestation1, d5$gestation2, paired = TRUE)
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  d5$gestation1 and d5$gestation2
## V = 27, p-value = 0.6
## alternative hypothesis: true location shift is not equal to 0
```

4.1.3 Calcul de l'intervalle de confiance autour de la différence entre deux moyennes

Lorsque le test T de Student peut être utilisé, l'intervalle de confiance autour de la différence entre les deux moyennes est donné par la fonction `t.test()` (sortie `conf.int`) et affiché en dernier dans la sortie de la fonction. Dans certains (ex. test d'équivalence) seul le calcul de l'intervalle de confiance intéressera l'utilisateur, et il aura peut-être intérêt, pour ne pas être parasité par le résultat du test de signification (p-value) à ne regarder que cet intervalle de confiance à l'aide da ligne de code suivante :

```r
t.test(d5$gestation1, d5$gestation2, paired = TRUE)$conf.int

## [1] -2.46  3.80
## attr(,"conf.level")
## [1] 0.95
```

4.2 Variable qualitative à 2 modalités

4.2.1 Création de la table de concordance

Suivant que l'on dispose (1) ou non (2) des données brutes, la construction de la table de concordance sera réalisée différemment :

1. **Si l'on dispose des données brutes**, la **table de concordance** peut être obtenue automatiquement à l'aide de la fonction `table()` à partir des vecteurs correspondant aux deux séries appariées de la variable qualitative comme ci-dessous :

```
d8 <- read.table("testsouris.txt", header=TRUE, stringsAsFactors = TRUE)
str(d8)
```

```
## 'data.frame': 100 obs. of  2 variables:
##  $ test1: Factor w/ 2 levels "echec","succes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ test2: Factor w/ 2 levels "echec","succes": 2 2 2 2 2 2 2 2 2 2 ...
```

```
tconcordance <- table(d8$test1, d8$test2)
tconcordance
```

```
##
##          echec succes
##   echec      6     18
##   succes     6     70
```

ATTENTION, il est important de noter que le codage des données dans ce cas est différent de celui utilisé pour des séries indépendantes : avec des séries indépendantes dans un exemple de ce type, on aurait dans le jeu de données une colonne correspondant au type de test (test1 ou test2) et une colonne correspondant au résultat du test, codage qui ne permet pas de coder un appariement.

2. **Si l'on dispose directement de la table de concordance**, on peut la coder comme une table sous la forme suivante avec e, f, g, h (cf. illustration ci-dessous) les nombres de chaque combinaison possible de valeurs des deux séries (valeurs notées + ou - par exemple pour succès ou échec lors de la comparaison de deux tests diagnostiques), g+f correspondant au nombre total de discordances entre les deux séries :

	+	-
+	e	f
-	g	h

```
tconcordancebis <- as.table(matrix(c(70,6,18,6),nrow=2,byrow=TRUE))
rownames(tconcordancebis) <- c("TA+", "TA-")
colnames(tconcordancebis) <- c("TB+", "TB-")
tconcordancebis
```

```
##     TB+ TB-
## TA+  70   6
## TA-  18   6
```

4.2.2 Réalisation du test du McNemar à partir de la table de concordance

Une fois la table de concordance obtenue, on réalise le test de McNemar de la façon suivante :

```
mcnemar.test(tconcordance)
```

```
##
##  McNemar's Chi-squared test with continuity correction
##
## data:  tconcordance
## McNemar's chi-squared = 5, df = 1, p-value = 0.02
```

5 Avec plusieurs séries indépendantes d'observations

5.1 Variable quantitative

5.1.1 Test de comparaison globale de plusieurs moyennes ou tendances centrales

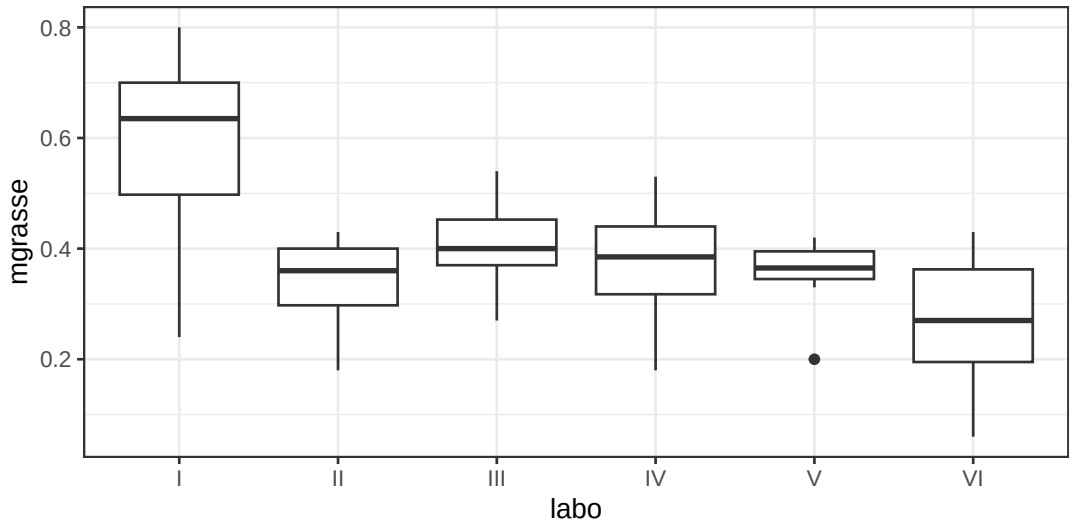
Comme dans le cas de la comparaison de deux moyennes observées sur des séries indépendantes, les données peuvent être importées sous la forme d'un tableau contenant deux colonnes, une colonne avec les valeurs observées de la variable

quantitative (nommée "mgrasse" dans cet exemple) et une colonne avec l'indication du groupe d'appartenance de chaque observation (nommé "labo" dans cet exemple). L'ordre de présentation des colonnes et des lignes n'a pas d'importance.

```
d9 <- read.table("egg.txt", header = TRUE, stringsAsFactors = TRUE)
head(d9)

##   mgrasse labo
## 1    0.62    I
## 2    0.55    I
## 3    0.34    I
## 4    0.24    I
## 5    0.80    I
## 6    0.68    I
```

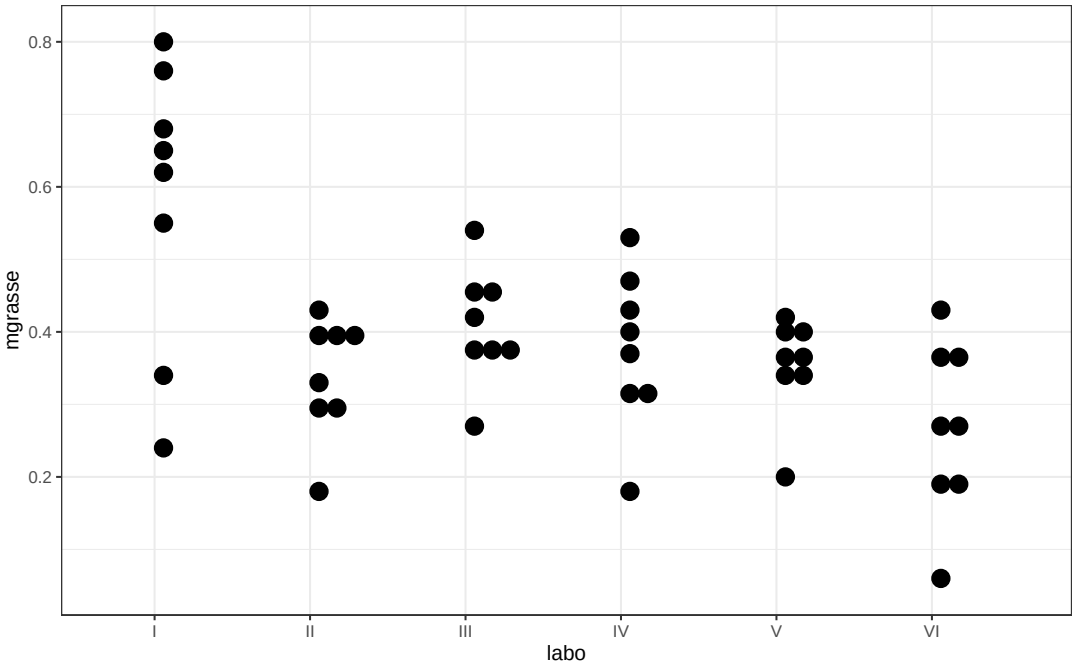
Les diagrammes en boîte des distributions dans chaque groupe peuvent être facilement tracés :

```
ggplot(d9, aes(x = labo, y = mgrasse)) + geom_boxplot()
```



La fonction geom_dotplot() permettant de représenter tous les points observés par groupe est plus adaptée dans le cas de très faibles effectifs.

```
ggplot(d9, aes(x = labo, y = mgrasse)) + geom_dotplot(binaxis = "y")
```



En fonction de l'exploration des distributions (normalité, effectifs, égalité des variances), l'un des trois tests suivants peut être réalisé :

— soit le **test de l'analyse de variance classique avec variances égales**

```
oneway.test(d9$mgrasse ~ d9$labo, var.equal = TRUE)
##
##  One-way analysis of means
##
## data:  d9$mgrasse and d9$labo
## F = 6, num df = 5, denom df = 42, p-value = 2e-04
# calcul complémentaire des moyennes par groupe pour la description des données
tapply(d9$mgrasse, d9$labo, mean)
##     I    II   III    IV     V    VI
## 0.580 0.340 0.408 0.376 0.354 0.268
```

— soit le **test de l'analyse de variance avec variances différentes (extension du test de Welch)** :

```
oneway.test(d9$mgrasse ~ d9$labo,var.equal = FALSE)
##
##  One-way analysis of means (not assuming equal variances)
##
## data:  d9$mgrasse and d9$labo
## F = 3, num df = 5, denom df = 19, p-value = 0.03
# calcul complémentaire des moyennes par groupe pour la description des données
tapply(d9$mgrasse, d9$labo, mean)
##     I    II   III    IV     V    VI
## 0.580 0.340 0.408 0.376 0.354 0.268
```

— soit le **test non paramétrique de comparaison de plusieurs tendances centrales de Kruskal-Wallis** :

```
kruskal.test(d9$mgrasse ~ d9$labo)
##
##  Kruskal-Wallis rank sum test
##
## data:  d9$mgrasse by d9$labo
## Kruskal-Wallis chi-squared = 14, df = 5, p-value = 0.02
# calcul complémentaire des médianes par groupe pour la description des données
tapply(d9$mgrasse, d9$labo, median)
##     I    II   III    IV     V    VI
## 0.635 0.360 0.400 0.385 0.365 0.270
```

5.1.2 Test de comparaison deux à deux de plusieurs moyennes ou tendances centrales

Dans le cas de la mise en évidence d'une différence globale significative entre les moyennes, il est possible, si besoin, de comparer les moyennes deux à deux en corrigeant les valeurs de p-value par la méthode historique de Bonferroni ou par sa version améliorée dite de Bonferroni-Holm (choix par défaut dans R si on indique pas la méthode), ou par une autre méthode spécifiée dans l'argument `p.adjust.method` des fonctions ci-dessous, en cohérence avec le choix du test global réalisé auparavant (paramétrique ou non, avec variances égales ou non),

— ex. d'un **test de comparaison 2 à 2 paramétrique avec estimation d'une variance commune et la correction de Bonferroni** :

```
pairwise.t.test(d9$mgrasse, d9$labo, pool.sd = TRUE, p.adjust.method = "bonferroni")
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  d9$mgrasse and d9$labo
##
##     I     II    III   IV    V
## II  0.003 -     -     -     -
## III 0.081 1.000 -     -     -
## IV  0.018 1.000 1.000 -     -
## V   0.006 1.000 1.000 1.000 -
## VI  6e-05 1.000 0.327 1.000 1.000
##
## P value adjustment method: bonferroni
```

— ex. d'un **test de comparaison 2 à 2 paramétrique avec estimation d'une variance commune et la correction de Bonferroni-Holm** :

```
pairwise.t.test(d9$mgrasse, d9$labo, pool.sd = TRUE, p.adjust.method = "holm")
##
```

```
##  Pairwise comparisons using t tests with pooled SD
##
## data:  d9$mgrasse and d9$labo
##
##     I     II    III   IV    V
## II  0.003 -     -     -     -
## III 0.059 1.000 -     -     -
## IV  0.015 1.000 1.000 -     -
## V   0.005 1.000 1.000 1.000 -
## VI  6e-05 1.000 0.218 0.641 1.000
##
## P value adjustment method: holm
```

— ex. d'un **test de comparaison 2 à 2 paramétrique avec estimation d'une variance par groupe et la correction de Bonferroni-Holm** :

```
pairwise.t.test(d9$mgrasse, d9$labo, pool.sd = FALSE, p.adjust.method = "holm")
##
##  Pairwise comparisons using t tests with non-pooled SD
##
## data:  d9$mgrasse and d9$labo
##
##     I    II   III  IV   V
## II  0.15 -    -    -    -
## III 0.47 0.84 -    -    -
## IV  0.29 1.00 1.00 -    -
## V   0.18 1.00 1.00 1.00 -
## VI  0.04 1.00 0.21 0.70 0.84
##
## P value adjustment method: holm
```

— ex. d'un **test de comparaison 2 à 2 non paramétrique et la correction de Bonferroni-Holm** :

```
pairwise.wilcox.test(d9$mgrasse, d9$labo, p.adjust.method = "holm")
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  d9$mgrasse and d9$labo
##
##     I   II  III IV  V
## II  0.4 -   -   -   -
## III 0.8 1.0 -   -   -
## IV  0.5 1.0 1.0 -   -
## V   0.5 1.0 1.0 1.0 -
## VI  0.2 1.0 0.2 0.8 1.0
##
## P value adjustment method: holm
```

5.1.3 Test de comparaison globale de plusieurs variances

Dans le cas de distributions proches de lois normales, le test de Bartlett peut être réalisé à l'aide de la fonction `bartlett.test()`, pour mettre en évidence une différence globale entre plusieurs variances. **Rappelons qu'il ne permet pas, par contre, de montrer l'égalité des variances** :

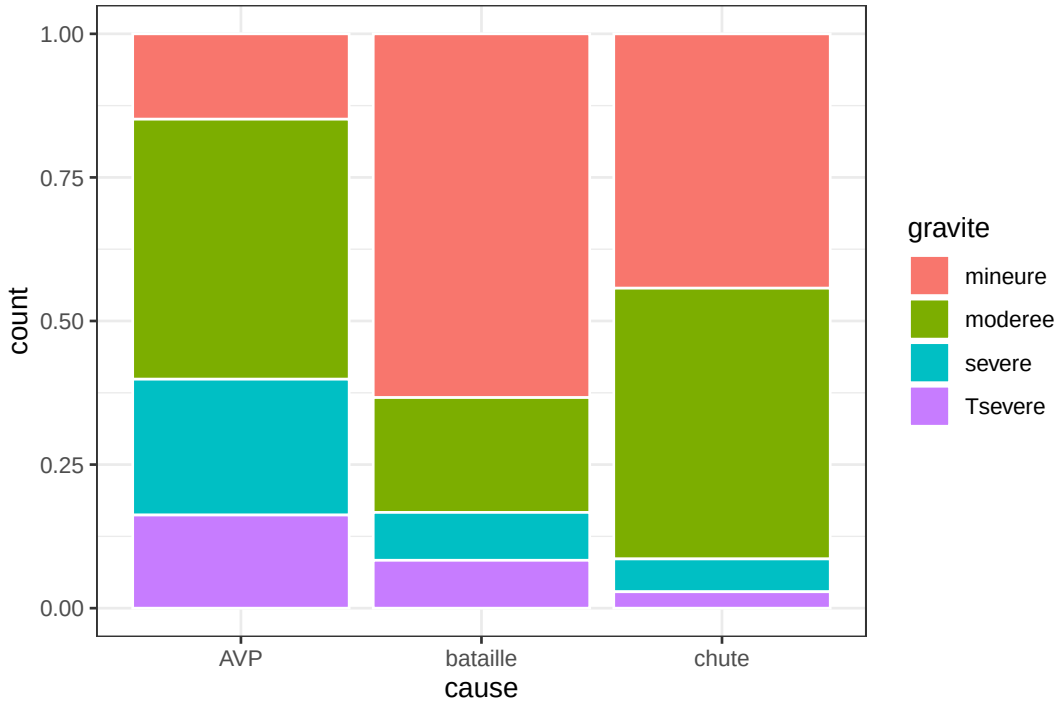
```
bartlett.test(d9$mgrasse ~ d9$labo)

##
##  Bartlett test of homogeneity of variances
##
## data:  d9$mgrasse by d9$labo
## Bartlett's K-squared = 11, df = 5, p-value = 0.06
```

5.2 Variable qualitative

Le cas de la comparaison de plusieurs séries indépendantes pour une variable qualitative équivaut à la comparaison de plusieurs distributions ou plusieurs fréquences sur séries indépendantes et se traite par un test du χ^2 d'indépendance comme

expliqué au chapitre 3.2 pour la comparaison de deux distributions ou fréquences. Voici un exemple sur un jeu de données correspondant à l'étude de la cause et de la gravité des traumatismes d'animaux admis au service de chirurgie sur le campus vétérinaire de Lyon.

```
d10 <- read.table('chirurgmodifie.txt', header = TRUE, stringsAsFactors = TRUE)
str(d10)

## 'data.frame': 278 obs. of  2 variables:
##  $ gravite: Factor w/ 4 levels "mineure","moderee",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ cause  : Factor w/ 3 levels "AVP","bataille",..: 1 1 1 1 1 1 1 1 1 1 ...

# Construction de la table de contingence
tchir <- table(d10$cause, d10$gravite)
tchir

##
##             mineure moderee severe Tsevere
##   AVP            22      67     35      24
##   bataille       38      12      5       5
##   chute          31      33      4       2

# Visualisation des données
ggplot(d10, aes(x = cause, fill = gravite)) + geom_bar(position = "fill", col = "white")
```




```
# test du khi2 d'indépendance
khi2 <- chisq.test(tchir)

# ou de façon équivalente ici
khi2 <- chisq.test(d10$cause, d10$gravite)
khi2

##
##  Pearson's Chi-squared test
##
## data:  d10$cause and d10$gravite
## X-squared = 63, df = 6, p-value = 1e-11

# Vérification des conditions d'utilisation
khi2$expected

##           d10$gravite
```

```
## d10$cause  mineure moderee severe Tsevere
##   AVP         48.4    59.6   23.4   16.50
##   bataille    19.6    24.2    9.5    6.69
##   chute       22.9    28.2   11.1    7.81
```

6 Avec plusieurs séries dépendantes d'observations

6.1 Variable quantitative

ATTENTION, ce cas n'est pas traité dans le cours de base, car il nécessite l'abord de l'analyse de variance à plusieurs facteurs et des modèles mixtes qui est un sujet vaste et relativement délicat (abordé en optionnel de A6 - lien vers programme : https://biostatistique.vetagro-sup.fr/formcont.html).

6.2 Variable qualitative à 2 modalités

Les données doivent être importées sous la forme d'un tableau contenant trois colonnes, l'une codant le facteur étudié (celui qui différencie les séries, nommé site dans cet exemple), une autre codant la variable qualitative à deux modalités observée (nommé detection dans cet exemple), et enfin une dernière codant le facteur d'appariement des séries (facteur bloc, nommé carcasse dans cet exemple)

```
d11 <- read.table("carcasses.txt", header = TRUE, stringsAsFactors = TRUE)
head(d11)

##   site carcasse detection
## 1   BV        1         0
## 2   EC        1         0
## 3   GC        1         1
## 4   JC        1         0
## 5   JV        1         0
## 6   LC        1         0
```

Si l'on souhaite calculer au préalable les fréquences à comparer, d'une des deux modalités de la variable étudiée en fonction du facteur étudié (indépendamment du facteur bloc), on peut simplement calculer les moyennes de la variable qualitative à deux modalités (codée 0 ou 1) pour chaque série de la fonction suivante :

```
tapply(d11$detection, d11$site, mean)

##     BV     EC     GC     JC     JV     LC     PC     PV
## 0.0303 0.1515 0.1515 0.0606 0.0000 0.2424 0.2424 0.1212
```

Le test de Cochran-Mantel-Haenszel peut ensuite être réalisé à l'aide de la fonction `mantelhaen.test()` en indiquant en premier argument la variable qualitative à deux modalités, en deuxième argument la variable codant la série d'appartenance, et en troisième argument le facteur d'appariement des séries :

```
mantelhaen.test(as.factor(d11$detection), d11$site,as.factor(d11$carcasse))

##
##  Cochran-Mantel-Haenszel test
##
## data:  as.factor(d11$detection) and d11$site and as.factor(d11$carcasse)
## Cochran-Mantel-Haenszel M^2 = 19, df = 7, p-value = 0.007

# rappelons que la fonction as.factor permet de transformer un variable
# codée numériquement (ici avec des 0 et des 1) en variable qualitative
```

7 Lorsque deux variables quantitatives sont observées sur les mêmes unités d'observation (corrélation ou régression linéaire)

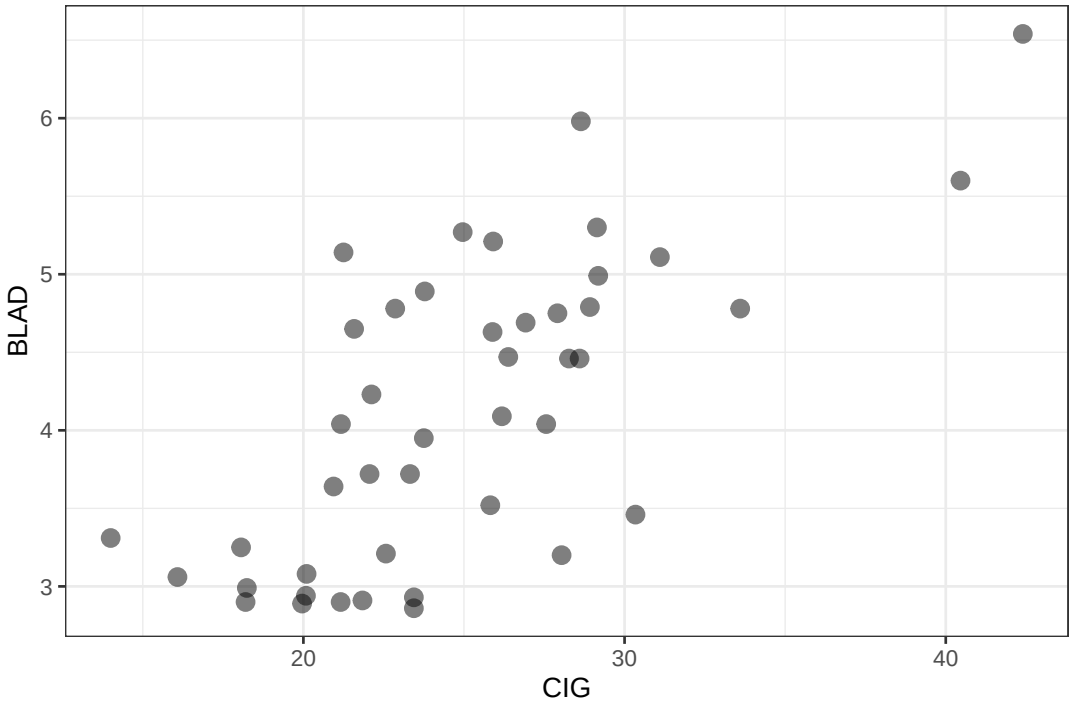
7.0.1 Corrélation linéaire entre deux variables observées

Les données seront importées sous la forme d'un tableau contenant les deux colonnes correspondant aux deux variables observées à corréler, dans cet exemple la consommation de cigarettes nommée "CIG" et la mortalité par cancer de la vessie nommée "BLAD".

```r
d6 <- read.table("smoking.txt", header = TRUE, stringsAsFactors = TRUE)
str(d6)

## 'data.frame': 44 obs. of  6 variables:
##  $ STATE: Factor w/ 44 levels "AK","AL","AR",..: 2 4 3 5 6 8 7 9 10 11 ...
##  $ CIG  : num  18.2 25.8 18.2 28.6 31.1 ...
##  $ BLAD : num  2.9 3.52 2.99 4.46 5.11 4.78 5.6 4.46 3.08 4.75 ...
##  $ LUNG : num  17.1 19.8 16 22.1 22.8 ...
##  $ KID  : num  1.59 2.75 2.02 2.66 3.35 3.36 3.13 2.41 2.46 2.95 ...
##  $ LEUK : num  6.15 6.61 6.94 7.06 7.2 6.45 7.08 6.07 6.62 7.27 ...
```

Les données peuvent être représentées sous la forme d'un diagramme de dispersion (ou nuage de points) tout simplement avec la fonction `plot()`.

```r
ggplot(d6, aes(x = CIG, y = BLAD)) + geom_point(alpha = 0.5, size = 3)
```




```r
# pour rappel l'argument alpha permet de gérer la transparence
# et size ici la taille de tous les points
```

Suivant la forme plus ou moins elliptique du nuage de points, on pourra effectuer l'un des deux tests de corrélations suivants :

— soit le **test paramétrique du coefficient de corrélation linéaire de Pearson**

```r
cor.test(d6$CIG, d6$BLAD, method = "pearson")
##
##  Pearson's product-moment correlation
##
## data:  d6$CIG and d6$BLAD
## t = 6, df = 42, p-value = 1e-07
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.514 0.828
## sample estimates:
##   cor
## 0.704
```

— soit le **test non paramétrique du coefficient de corrélation de rangs de Spearman**

```r
cor.test(d6$CIG, d6$BLAD, method = "spearman")
##
##  Spearman's rank correlation rho
##
```

```
## data:  d6$CIG and d6$BLAD
## S = 4688, p-value = 7e-07
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##  rho
## 0.67
```

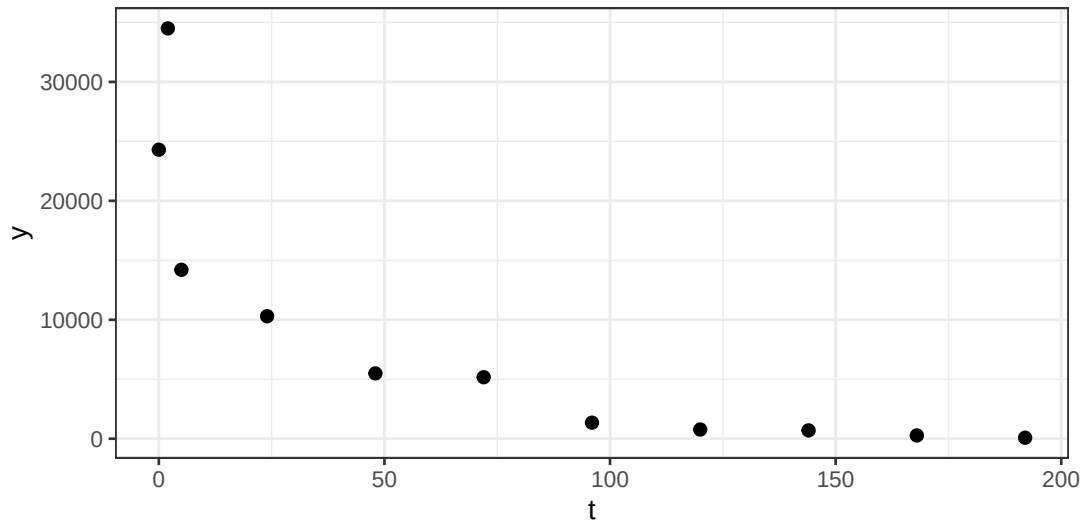
7.0.2 Régression linéaire entre une variable observée et une variable contrôlée

Les données seront importées sous la forme d'un tableau contenant les deux colonnes correspondant à la variable de contrôle ("t" dans cet exemple), et à la variable observée ("y" dans cet exemple).

```
d7<- read.table("survyaourtbrut.txt", header = TRUE, stringsAsFactors = TRUE)
str(d7)

## 'data.frame': 11 obs. of  2 variables:
##  $ t: int  0 2 5 24 48 72 96 120 144 168 ...
##  $ y: num  24300 34500 14200 10300 5490 5170 1350 773 699 273 ...
```

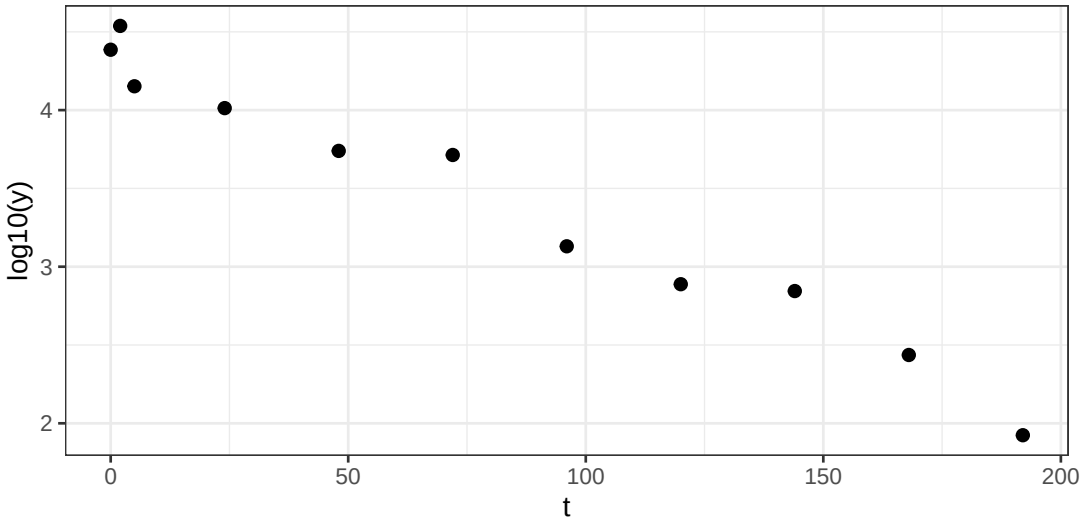
Les données pourront être représentées sous forme de diagramme de dispersion (ou nuage de points) avec la fonction `plot()` en indiquant bien la variable contrôlée comme premier argument :

```
ggplot(d7, aes(x = t, y = y)) + geom_point(size = 2)
```

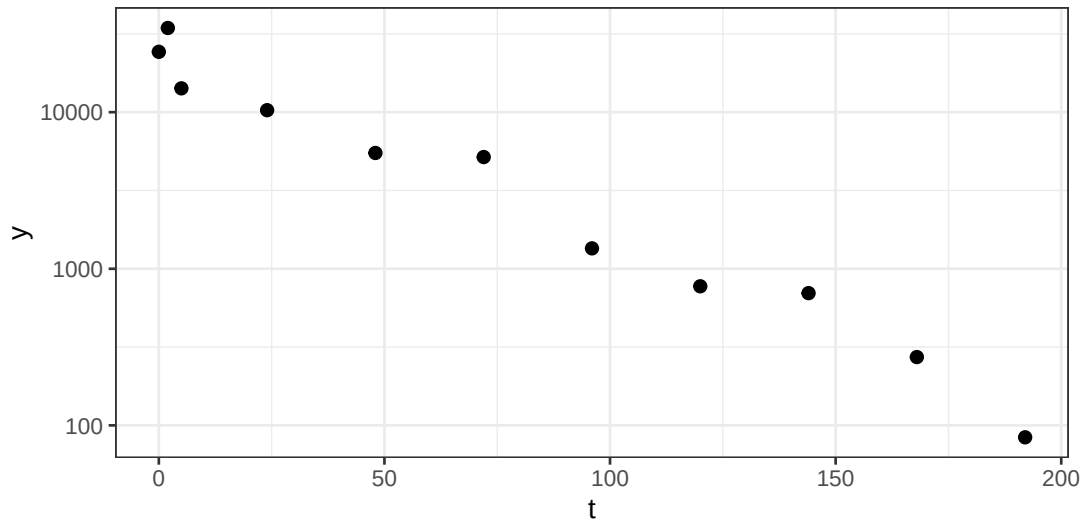


Dans cet exemple il conviendra de tracer les données avec la variable observée transformée en logarithme décimal :

```
# Soit en traçant la variable transformée
ggplot(d7, aes(x = t, y = log10(y))) + geom_point(size = 2)
```



```r
# Soit en changeant l'échelle des y
ggplot(d7, aes(x = t, y = y)) + geom_point(size = 2) + scale_y_log10()
```



L'ajustement d'un modèle linéaire aux données par régression peut être réalisé à l'aide de la fonction `lm()` (pour "linear model"). Il convient de lui mettre en argument la formule du modèle, composée du nom de la variable observée, suivi du signe $\sim$ et du nom de la variable de contrôle :

```r
(modele <- lm(log10(y) ~ t, data = d7))

##
## Call:
## lm(formula = log10(y) ~ t, data = d7)
##
## Coefficients:
## (Intercept)            t
##       4.382       -0.012
```

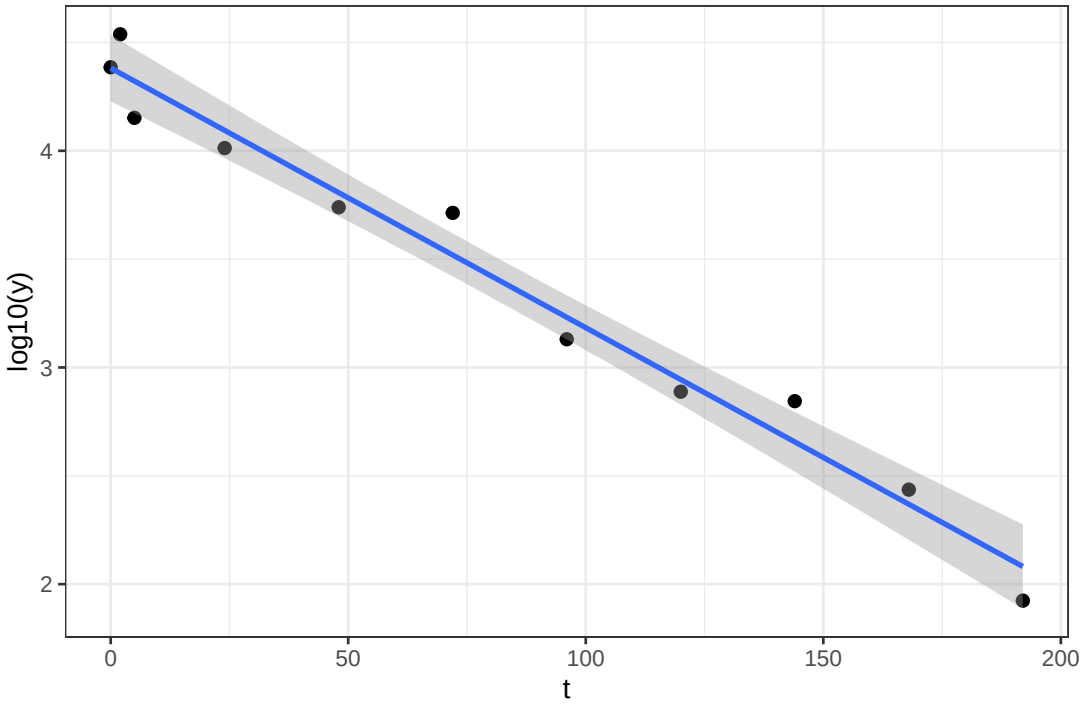
On peut récupérer un résumé assez complet de l'ajustement avec la fonction `summary()`, les seuls coefficients (pour utilisation dans un calcul par exemple) avec la fonction `coef()` et leur intervalle de confiance à l'aide de la fonction `confint()`.

```r
# Résumé de l'ajustement
summary(modele)

##
## Call:
## lm(formula = log10(y) ~ t, data = d7)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.1697 -0.0914 -0.0559  0.1236  0.1943
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.381893   0.067820    64.6  2.6e-13
## t           -0.011982   0.000656   -18.3  2.0e-08
##
## Residual standard error: 0.145 on 9 degrees of freedom
## Multiple R-squared:  0.974, Adjusted R-squared:  0.971
## F-statistic:  334 on 1 and 9 DF,  p-value: 2.02e-08

# Coefficients de la droite
coef(modele)

## (Intercept)           t
##       4.382      -0.012

# Intervalle de confiance à 95% sur chacun des ces coefficients
confint(modele)
```

```
##                2.5 %  97.5 %
## (Intercept)  4.2285  4.5353
## t           -0.0135 -0.0105
```

La droite ajustée peut être ajoutée au diagramme de dispersion réalisé préalablement à l'aide de la fonction geom_smooth() comme ci-dessous. Par défaut geom_smooth() ajoute à la droite sa bande de confiance à 95% qui donne pour chaque valeur de x, l'intervalle de confiance sur la valeur moyenne de y prédite par le modèle. ()

```
ggplot(d7, aes(x = t, y = log10(y))) + geom_point(size = 2) +
  geom_smooth(method = "lm")
```

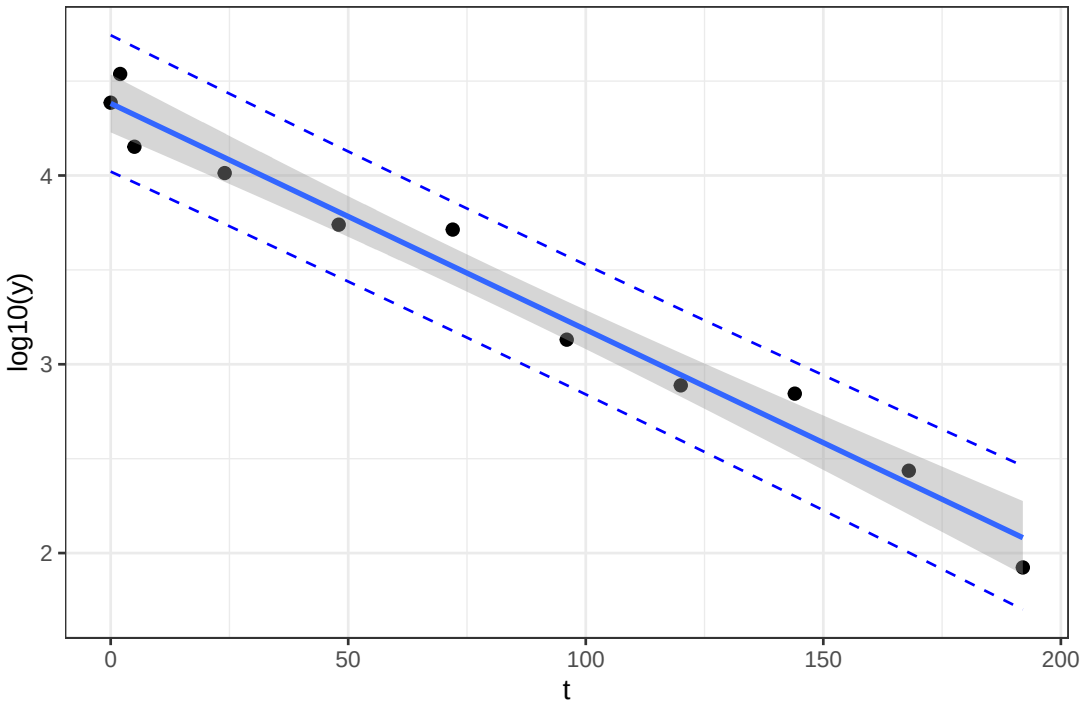


On peut obtenir les valeurs prédites pour de nouvelles valeurs de la variable explicative ainsi que les intervalles de confiance sur la valeur prédite (interval = "prediction") ou de confiance sur la moyenne prédite (interval = "confidence") ainsi :

```
# Prédiction de log10y pour t = 100 et 150
predict(modele, data.frame(t = c(100, 150)), interval = "prediction")

##    fit  lwr  upr
## 1 3.18 2.84 3.53
## 2 2.58 2.23 2.94
```

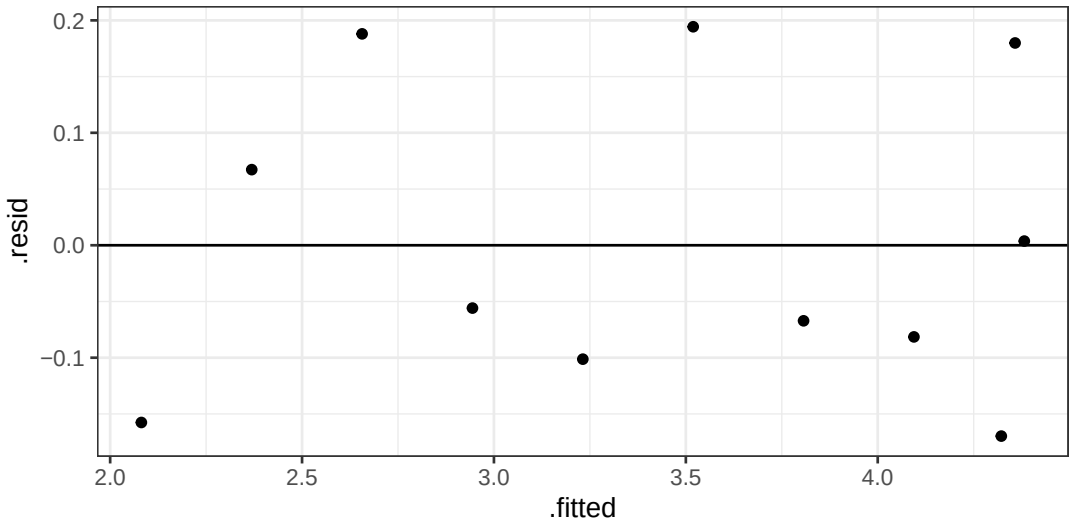
On peut ajouter au graphe d'ajustement la bande de prédiction donnant les intervalles de prédiction à 95% de y pour chaque valeur de x en utilisant par exemple le code suivant.

```
# calcul des intervalles de prédiction pour les points du jeu de données
ipred <- predict(modele, interval = "prediction")
# collage des colonnes lwr (lower bound) et upr (upper bound)
# au jeu de données
d7ETpred <- cbind(d7, ipred)
# Tracé de la courbe ajusté à partir du jeu de données complétés
# et ajout des bornes des intervalles de prédiction
ggplot(d7ETpred, aes(x = t, y = log10(y))) + geom_point(size = 2) +
  geom_smooth(method = "lm") +
  geom_line(aes(y = upr), color = "blue", linetype = "dashed") +
  geom_line(aes(y = lwr), color = "blue", linetype = "dashed")
```

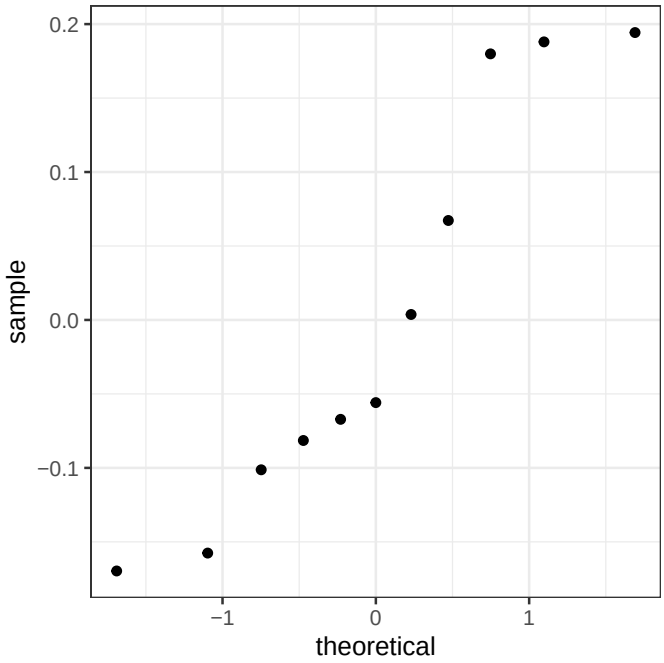


Enfin pour examiner les résidus on peut facilement faire un graphe des résidus et un diagramme quantile-quantile des résidus :

```
# Graphe des résidus
ggplot(modele, aes(x = .fitted, y = .resid)) +
  geom_point() + geom_hline(yintercept = 0)
```




```
# Diagramme quantile-quantile des résidus
ggplot(modele, aes(sample = .resid)) + geom_qq()
```



8 Calcul de puissance ou détermination *a priori* du nombre d'observations nécessaires

8.1 Tests sur deux séries indépendantes d'observations

8.1.1 Variable quantitative

Prenons un exemple : on veut comparer la production laitière de vaches de race Holstein-Frisonne élevées dans deux régions différentes de l'Angleterre. La production moyenne attendue pour ces vaches en Angleterre est d'environ 6500 kg par lactation (période de 305 jours en moyenne). On veut savoir combien d'animaux il faudrait observer par échantillon (pris dans chaque région) pour avoir une probabilité de 85% de détecter avec un risque de 5% une différence moyenne de production de 250 kg. Des informations publiées précédemment indiquent un écart type de la production laitière de 1425 kg pour cette race bovine.

La fonction `power.t.test()` permet de répondre à cette question en supposant qu'un test T avec variances égales sera applicable. Il faut affecter l'argument `type` de la fonction à `"two.sample"`, spécifier le type d'hypothèse alternative dans l'argument `alternative` (par défaut "two.sided" pour un spécifier un test bilatéral), et affecter les arguments numériques nécessaires, pour le calcul de l'argument affecté à `NULL`. `n` représente l'effectif de chaque groupe, `delta` la différence qu'on veut pouvoir détecter, `sd` l'écart type supposé commun dans les 2 groupes, `sig.level` le risque de première espèce et `power` la puissance du test.

Ainsi pour répondre à la question posée dans l'exemple il faudra saisir le code suivant :

```
power.t.test(n = NULL, delta = 250, sd = 1425, sig.level = 0.05, power = 0.85,
             type = "two.sample", alternative = "two.sided")

##
##      Two-sample t test power calculation
##
##               n = 584
##           delta = 250
##              sd = 1425
##       sig.level = 0.05
##           power = 0.85
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

Si maintenant nous voulions connaître la puissance du test réalisé avec un effectif de 100 vaches par groupe, il faudrait saisir le code suivant :

```
power.t.test(n = 100, delta = 250, sd=1425, sig.level = 0.05, power = NULL,
             type = "two.sample", alternative = "two.sided")
```

```
##
##      Two-sample t test power calculation
##
##               n = 100
##           delta = 250
##              sd = 1425
##       sig.level = 0.05
##           power = 0.234
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

Nous pourrions aussi calculer la différence que le test aurait une probabilité de 85% de détecter toujours avec un effectif de 100 vaches par groupe en saisissant :

```
power.t.test(n = 100, delta = NULL, sd = 1425, sig.level = 0.05, power = 0.85,
             type = "two.sample", alternative = "two.sided")
```

```
##
##      Two-sample t test power calculation
##
##               n = 100
##           delta = 607
##              sd = 1425
##       sig.level = 0.05
##           power = 0.85
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

8.1.2 Variable qualitative

Prenons un exemple : *Toxocara canis* est un parasite commun du chien qui peut provoquer une cécité chez les enfants contaminés. On veut savoir si la prévalence d'infestation par ce parasite est réellement plus importante chez les chiots (âgés de moins de 6 mois) que chez les chiens adultes (âgés de plus d'un an) en zone urbaine. On s'attend à une prévalence d'environ 30% chez les chiots et 5% chez les adultes. En se fixant un risque de première espèce de 5% et une puissance de 90%, combien faut-il que l'on observe d'animaux de chaque groupe ?

La fonction `power.prop.test()` permet de répondre à cette question en supposant qu'un test de comparaison de deux fréquences (équivalent au test du χ^2 d'indépendance) sera applicable (à vérifier *a posteriori*). Il faut spécifier le type d'hypothèse alternative dans l'argument `alternative` (par défaut "two.sided" pour un spécifier un test bilatéral), et affecter les arguments numériques nécessaires, pour le calcul de l'argument affecté à `NULL`. `n` représente l'effectif de chaque groupe, `p1` la fréquence attendue dans le premier groupe, `p2` la fréquence attendue dans le deuxième groupe, `sig.level` le risque de première espèce et `power` la puissance du test.

Ainsi pour répondre à la question posée dans l'exemple il faudra saisir le code suivant :

```
power.prop.test(n = NULL, p1 = 0.3, p2 = 0.05, sig.level = 0.05,
                power = 0.9, alternative = "two.sided")
```

```
##
##      Two-sample comparison of proportions power calculation
##
##               n = 46.4
##              p1 = 0.3
##              p2 = 0.05
##       sig.level = 0.05
##           power = 0.9
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

8.2 Tests sur deux séries dépendantes (ou appariées) d'observations

8.2.1 Variable quantitative

Prenons un exemple : on veut comparer l'effet de deux exercices d'entraînement de chevaux sur tapis roulant sur la concentration plasmatique en lactate. Pour ceci on conduit un essai de type " cross-over " où chaque sujet est pris comme son propre témoin, c'est-à-dire que chaque cheval subit successivement et dans un ordre tiré au sort les deux exercices à comparer. Une différence de 1 mmol/l est considérée comme intéressante d'un point de vue biologique. Une étude pilote à permis d'estimer l'écart type des différences entre les concentrations plasmatiques en lactate entre les deux exercices à 1.7 mmol/l. Combien faudrait-il de chevaux pour avoir une probabilité de 90% de détecter avec un risque de 1% une différence de 1 mmol/l entre les deux exercices d'entraînement ?

La fonction `power.t.test()` permet de répondre à ces questions en supposant qu'un test T de comparaison de séries appariées sera applicable. Il faut affecter l'argument `type` de la fonction à `"paired"`, spécifier le type d'hypothèse alternative dans l'argument `alternative` (par défaut "two.sided" pour un spécifier un test bilatéral), et affecter les arguments numériques nécessaires, pour le calcul de l'argument affecté à `NULL`. `n` représente le nombre de paires d'observations, `delta` la différence qu'on veut pouvoir détecter, `sd` l'écart type des différences, `sig.level` le risque de première espèce et `power` la puissance du test.

Pour répondre à cette question, il faut saisir le code suivant :

```
power.t.test(n = NULL, delta = 1, sd = 1.7, sig.level = 0.01, power = 0.90,
             type = "paired", alternative = "two.sided")

##
##      Paired t test power calculation
##
##               n = 46.4
##           delta = 1
##              sd = 1.7
##       sig.level = 0.01
##           power = 0.9
##     alternative = two.sided
##
## NOTE: n is number of *pairs*, sd is std.dev. of *differences* within pairs
```

9 Liens vers des aides en ligne utiles

Pour l'utilisation de R

— la "cheat sheet" de la base de R : https://iqss.github.io/dss-workshops/R/Rintro/base-r-cheat-sheet.pdf

— la carte de référence de R en français : https://www.apmep.fr/IMG/pdf/R_RefCard.pdf

— la carte de référence de R (version 2 plus moderne) en anglais : https://cran.r-project.org/doc/contrib/Baggott-refcard-v2.pdf

— Le "cookbook" de R : http://www.cookbook-r.com/

Pour l'utilisation de rmarkdown pour inclure le code R (et les sorties) dans un rapport d'analyse

— le "cookbook" de rmarkdown : https://bookdown.org/yihui/rmarkdown-cookbook/

— la "cheat sheet" de rmardown : https://rstudio.github.io/cheatsheets/rmarkdown.pdf

Pour l'utilisation la réalisation des graphes, notamment avec ggplot2

— le "cookbook" des graphes : https://r-graphics.org/

— la "cheat sheet" de ggplot2 : https://rstudio.github.io/cheatsheets/data-visualization.pdf

10 Récapitulatif des principales fonctions pour les tests et la régression linéaire

Voici la liste des fonctions utilisées pour réaliser les différents tests au programme et ajuster un modèle linéaire, dans leur ordre d'apparition dans le guide, avec la spécification de leurs arguments, et leur nom en anglais donné dans la sortie de R.

Test de normalité de Shapiro-Wilk (Shapiro-Wilk normality test)

```
shapiro.test(variable_quanti)
```

Test de Student de conformité à une moyenne théorique (One Sample t-test)

```
t.test(variable_quanti, mu = moyenne_theorique)
```

Test du χ^2 d'ajustement (Chi-squared test for given probabilities)

```
chisq.test(vecteur_effectifs_obs, p = vecteur_proportions_theo)
```

Test exact (utilisant la loi binomiale) de comparaison d'une fréquence observée à une fréquence théorique (Exact binomial test)

```
binom.test(nb_realisations_evenement, nb_tirages_total, p = proportion_theo)
```

Test de Student de comparaison de 2 moyennes sur séries indépendantes avec variances égales (Two Sample t-test)

```
t.test(variable_quanti ~ groupe_variable_quali, var.equal = TRUE)
```

Test de Welch de comparaison de 2 moyennes sur séries indépendantes avec variances inégales (Welch Two Sample t-test)

```
t.test(variable_quanti ~ groupe_variable_quali, var.equal = FALSE)
```

Test non paramétrique de la somme des rangs de Mann-Whitney-Wilcoxon de comparaison de séries indépendantes (Wilcoxon rank sum exact test)

```
wilcox.test(variable_quanti ~ groupe_variable_quali)
```

Test de Fisher de comparaison de 2 variances (F test to compare two variances)

```
var.test(variable_quanti ~ groupe_variable_quali)
```

Test du χ^2 d'indépendance réalisé à partir de la table de contingence (Pearson's Chi-squared test)

```
# A partir des données brutes
chisq.test(variable_quali1, variable_quali2)
# A partir de la table de contingence
chisq.test(table_contingence)
# Fonction aussi utilisable pour la comparaison de 2 fréquences sur séries
# indépendantes qui donne en plus l'intervalle de confiance sur la différence
# entre les 2 fréquences comparées - ATTENTION de bien vérifier
# que les fréquences affichées en sorties (prop 1 et prop 2)
# sont bien celles sur lesquelles vous voulez calculer cette différence
prop.test(table_contingence)
```

Test exact de comparaison de 2 fréquences sur séries indépendantes (Fisher's Exact Test for Count Data)

```
fisher.test(table_contingence)
```

Test de Student de comparaison de 2 moyennes sur séries appariées (Paired t-test)

```r
t.test(variable_quanti_1, variable_quanti_2, paired = TRUE)
```

Test non paramétrique de Wilcoxon des rangs signés de comparaison de séries appariées (Wilcoxon signed rank test)

```r
wilcox.test(variable_quanti_1, variable_quanti_2, paired = TRUE)
```

Test de Mc Nemar de comparaison de 2 fréquences sur séries appariées, à partir de la table de concordance (McNemar's Chi-squared test)

```r
mcnemar.test(table_concordance)
```

Test de comparaison de plusieurs moyennes sur séries indépendantes en supposant les variances égales - analyse de variance classique avec variances égales (One-way analysis of means)

```r
oneway.test(variable_quanti ~ groupe_variable_quali, var.equal = TRUE)
```

Test de comparaison de plusieurs moyennes sur séries indépendantes sans supposer les variances égales extension du test de Welch - analyse de variance classique avec variances inégales (One-way analysis of means (not assuming equal variances))

```r
oneway.test(variable_quanti ~ groupe_variable_quali, var.equal = FALSE)
```

Test non paramétrique de la somme des rangs de Kruskal-Wallis de comparaison de séries indépendantes (Kruskal-Wallis rank sum exact test)

```r
kruskal.test(variable_quanti ~ groupe_variable_quali)
```

Test de comparaison de plusieurs variances (Bartlett test of homogeneity of variances)

```r
bartlett.test.test(variable_quanti ~ groupe_variable_quali)
```

Test de Cochran-Mantel-Haenszel de comparaison de plusieurs fréquences sur séries dépendantes (Cochran-Mantel-Haenszel test)

```r
mantelhaen.test(variable_quali_A, variable_quali_B, identifiant)
```

Test de corrélation linéaire de Pearson (Pearson's product-moment correlation)

```r
cor.test(variable_quanti_A, variable_quanti_B, method = "pearson")
```

Test non paramétrique de corrélation de rangs de Spearman (Spearman's rank correlation rho)

```r
cor.test(variable_quanti_A, variable_quanti_B, method = "spearman")
```

Régression linéaire simple (simple linear model)

```r
# Spécification du modèle
modele <- lm(variable_quanti_Y_a_expliquer ~ variable_quanti_X_explicative, data = jeu_de_donnes)
# Paramètres estimés et diverses statistiques résumées
summary(modele)
coef(modele)
# Intervalle de confiance sur les paramètres estimés
confint(modele)
# Intervalles de confiance sur les moyennes prédites pour des valeurs de X données
predict(modele, data.frame(variable_quanti_A_expliquer = vecteur_valeurs_X),
        interval = "confidence")
# Intervalles de confiance sur des prédictions individuelles pour des valeurs de X données
predict(modele, data.frame(variable_quanti_A_expliquer = vecteur_valeurs_X),
        interval = "prediction")
```