Introduction au logiciel R

Février 2015

Table des matières

Cours 1. Éléments du langage

Sommaire

Plus qu'un simple logiciel pour les statistiques, R est avant tout un langage de manipulation de données statistiques (observations, échantillon, modèle statistique, etc.). Cela explique en partie sa difficulté de prise en main pour les utilisateurs habitués à suivre des menus déroulants tels que ceux proposés par SPSS (même si SPSS offre également un langage basique de macro). Dans ce premier chapitre, on aborde les notions de base du langage R, les objets élémentaires que l'on peut manipuler et qui nous permettront de stocker les valeurs prises par une ou plusieurs variable (au sens statistique du terme), pour terminer avec la gestion de données pour des projets d'analyse un peu plus avancés : construction d'un tableau de données avec des variables numériques et catégorielles, gestion des données manquantes, sauvegarde dans un fichier externe.

Commandes essentielles

`c` : stocker une série de valeurs
`head` : aperçu des premières observations
`factor` : convertir une variable numérique en facteur
`is.na` : valeurs manquantes
`scan` : lire une série de valeurs
`write.table` : exporter des données au format texte
`data.frame` : créer un tableau de données
`summary` : résumé numérique d'une ou plusieurs variables
`ls` : afficher toutes les variables enregistrées par R
`length` : nombre d'observations
`levels` : niveaux ou modalités d'un facteur
`complete.cases` : valeurs non manquantes
`read.table` : importer un fichier de données texte
`save` : sauvegarder des données au format R
`names` : nommer les colonnes d'un tableau
`table` : tableau d'effectif pour une variable qualitative

1.1 Représentation des données sous R

1.1.1 Gestion de variables numériques

Supposons que l'on dispose du poids de 10 nouveaux-nés (x, en grammes) et de celui de leur mère (y, en kilogrammes) :

x	2523	2551	2557	2594	2600	2622	2637	2637	2663	2665
y	82.7	70.5	47.7	49.1	48.6	56.4	53.6	46.8	55.9	51.4

Dans l'exemple suivant, on créé une variable appelée `x` dans laquelle on stocke les observations de poids des nouveaux-nés reportées ci-dessus.

```
> x <- c(2523, 2551, 2557, 2594, 2600, 2622, 2637, 2637, 2663, 2665)
```

On notera l'usage du symbole `<-` (à préférer au symbole `=`) pour associer une série de mesures à une variable nommée. On peut à présent afficher ces valeurs sous R en utilisant la commande `print(x)` ou en tapant simplement le nom de la variable :

```
> x
 [1] 2523 2551 2557 2594 2600 2622 2637 2637 2663 2665
```

On procèderait naturellement de la même manière avec y

```
> y <- c(82.7, 70.5, 47.7, 49.1, 48.6, 56.4, 53.6, 46.8, 55.9, 51.4)
```

et l'on peut vérifier que R a gardé en mémoire les deux variables dans ce que l'on appelle l'espace de travail :

```
> ls()
[1] "x" "y"
```

Le nombre d'observations pour la variable x correspond à la longueur de x (nombre d'éléments), lorsqu'il n'y a pas de valeurs manquantes.

```
> length(x)
[1] 10
```

La 5e observation s'obtient de la manière suivante :

```
> x[5]
[1] 2600
```

et la première et la dernière sont ainsi :

```
> x[1]
[1] 2523
> x[10]
[1] 2665
```

Les valeurs d'une variable, ou en termes plus statistiques les valeurs prises par une variable pour un échantillon de taille n, peuvent être sélectionnées en précisant l'index ou n° de l'observation entre crochets. On verra plus en détails dans la partie suivante comment on peut sélectionner plus d'une observation à la fois, et comment sélectionner des observations sur la base d'un critère externe, et un peu plus tard encore comment on peut généraliser cette approcahe à la sélection d'un ensemble d'observations recueillies pour une même unité statistique.

1.1.2 Opérations sur une variable numérique

On aura remarqué que les poids des bébés sont exprimés en grammes, alors que ceux de la mère sont en kg. Le poids du 5e bébé en kg s'obtient à l'aide d'une simple opération arithmétique de division.

```
> x[5] / 1000
[1] 2.6
```

Mais, cette même division peut être appliquée à l'ensemble des observations, sans avoir à la réaliser pour chaque éléments de x pris individuellement, comme le montre l'exemple suivant :

```
> x / 1000
 [1] 2.52 2.55 2.56 2.59 2.60 2.62 2.64 2.64 2.66 2.67
```

L'opération précédente n'a pas réellement changé les valeurs de la varible x d'origine, mais il est très facile de créer une nouvelle variable dans laquelle on stockera les poids en kg.

```
> x2 <- x / 1000
> x
 [1] 2523 2551 2557 2594 2600 2622 2637 2637 2663 2665
```

```
> x2
 [1] 2.52 2.55 2.56 2.59 2.60 2.62 2.64 2.64 2.66 2.67
```

Évidemment, on peut également vouloir directement remplacer les anciennes valeurs de `x` par les nouvelles

```
> x <- x / 1000
```

mais on gardera à l'esprit que dans ce cas, il ne sera pas possible de revenir aux anciennes valeurs.

Outre les opérateurs arithmétiques de base (addition, soustraction, multiplication et division), R dispose de la plupart des fonctions que l'on trouve sur un calculateur : logarithme, exponentielle, valeur absolue, etc. Ainsi, `log(x)` renverrait les valeurs de `x` après transformation par le logarithme népérien, alors pour le logarithme en base 10 on utiliserait `log10(x)`.

Enfin, R offre une large gamme de commandes à plus spécifiquement statistiques, en particulier pour résumer la distribution des valeurs observées (tendance centrale, dispersion, étendue, etc.). La moyenne arithmétique des valeurs de `x` s'obtient simplement avec la commande `mean` :

```
> mean(x)
[1] 2605
```

D'autres commandes, sur lesquelles nous reviendrons dans le deuxième chapitre, permettent d'obtenir les informations essentielles concernant la forme de la distribution d'une variable continue.

```
> range(x)
[1] 2523 2665
> c(min(x), max(x))
[1] 2523 2665
> var(x)
[1] 2377
```

On notera qu'il est tout à fait possible de combiner deux commandes qui renvoient le même type de résultat, comme on le voit avec l'expression `c(min(x), max(x))`.

Enfin, les illustrations suivantes reposent sur la même idée que l'opération de division que l'on a évoquée plus haut : chaque opération (mise au carré et soustraction de la moyenne) est appliquée élément par élément.

```
> sum(x)
[1] 26049
> sum(x^2)
[1] 67876431
> sum(x^2 - mean(x))
[1] 67850382
```

Dans l'exemple ci-dessus, `mean(x)` est une constante (qui dépend des données mais qui ne varie pas dans ce cas de figure) que l'on soustrait de chaque x_i^2 (i dénotant le n° d'observation ou index dans la variable `x`), ce qui revient à manipuler 11 valeurs différentes au total. Dans l'expression suivante, en revanche,

```
> x - x^2
 [1] -6363006 -6505050 -6535692 -6726242 -6757400 -6872262 -6951132 -6951132 -7088906 -7099560
```

on soustrait de chaque élément de `x` sa propre valeur au carré (10 opérations au total, avec chaque fois 10 paires de nombres différents).

1.1.3 Gestion de variables catégorielles

Les variables catgéorielles ou qualitatives possèdent un statut à part dans la plupart des logiciels statistiques : leurs modalités sont représentés par des nombres (1, 2, 3, etc.) mais on leur associe généralement des étiquettes, encore appelées `labels`,

Dans l'exemple précédent on disposait de données sur le poids des bébés à leur naissance (x) et le poids de leur mère (y). Supposons que l'on sache également si lamère fumait pendant le premier trimestre de grossesse et appelons cette variable z. Lorsque la mère ne fumait pas pendant cette période, la variable vaut 1 ; lorsque la mère fumait, la variable vaut 2. Voici ci-dessous le tableau de données précédent modifié pour prendre en compte cette information.

x	2523	2551	2557	2594	2600	2622	2637	2637	2663	2665
y	82.7	70.5	47.7	49.1	48.6	56.4	53.6	46.8	55.9	51.4
z	1	1	2	2	2	1	1	1	2	2

On saisira les données sous R comme on l'a fait pour les variables x et y.

```
> z <- c(1, 1, 2, 2, 2, 1, 1, 1, 2, 2)
> z
```

```
 [1] 1 1 2 2 2 1 1 1 2 2
```

```
> factor(z, labels=c("NF","F"))
```

```
 [1] NF NF F  F  F  NF NF NF F  F
Levels: NF F
```

On notera que lorsque R affiche le contenu de la variable z, c'est-à-dire les valeurs observées pour le volume expiratoire de pointe, il est précisé qu'il s'agit d'une variable de type `factor`, avec trois niveaux, dans ce cas non ordonnés. Certaines opérations, comme le calcul d'une moyenne arithmétique, n'ont généralement aucun sens sur ce type de variable et c'est ce qui se passe avec R également. En revanche, les opérations de tri à plat ou croisé restent tout à fait valides.

Autre aspect important : les étiquettes associées à une variable qualitative peuvent être retrouvées à l'aide de la commande `levels`, tandis que `nlevels` renvoit le nombre total de modalités.

```
> z <- factor(z, labels=c("NF","F"))
> levels(z)
```

```
[1] "NF" "F"
```

```
> nlevels(z)
```

```
[1] 2
```

1.1.4 Manipulation de variables catégorielles

Il arrive fréquemment que l'on ait besoin de recoder une variable qualitative à k classes en $j < k$ classes, ou que l'on souhaite associer des étiquettes (ou *labels*) aux modalités numériques d'une variable qualitative. Par exemple, supposons que l'on dispose des données de 10 patients sur le volume expiratoire maximal seconde (variable `vems`) sous forme d'une variable à trois niveaux ordonnés : critique (1), bas (2), normal (3).

```
> vems <- sample(1:3, 10, replace=TRUE)
> vems
```

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

On souhaite remplacer les valeurs numériques par les étiquettes décrites ci-dessus. Il y a plusieurs solutions, mais le plus simple est d'utiliser la commande `factor` et d'assigner des étiquettes à chaque valeur de `vems`, encore appelée niveau (`levels`).

```
> vems <- factor(vems, levels=c(1,2,3), labels=c("critique","bas","normal"))
> vems
```

```
 [1] bas      critique bas      normal   critique critique normal   bas      critique normal
Levels: critique bas normal
```

On peut très rapidement vérifier la distribution des effectifs par classe à l'aide de la commande `table` qui permet de réaliser des tris à plat ou des tris croisés (que l'on verra au chapitre suivant).

```
> table(vems)
```

```
vems
critique      bas   normal
       4        3        3
```

On peut tout à fait modifier les niveaux d'un facteur : si l'on souhaite, par exemple, recoder la variable `vems` à 3 classes en une variable à 2 classes en aggrégeant les deux premières modalités ou niveaux, on peut procéder de la manière suivante :

```
> levels(vems)
```

```
[1] "critique" "bas"      "normal"
```

```
> levels(vems)[1:2] <- "critique ou bas"
> levels(vems)
```

```
[1] "critique ou bas" "normal"
```

```
> table(vems)
```

```
vems
critique ou bas          normal
              7               3
```

On remarquera que les modalités de la variable `vems` ont bien été modifiées et que les effectifs des deux premiers niveaux d'origine se retrouvent à présent dans le premier niveau intitulé `"critique ou bas"`. La variable `vems` a été définitivement modifiée.

1.2 Sélection d'observations

1.2.1 Sélection indexée d'observations

Avec les données de poids discutées plus haut, on pourrait vouloir sélectionner plus d'une observation, par exemple la 3e et la 5e :

```
> x[c(3,5)]
```

```
[1] 2557 2600
```

ou de la 3e et la 5e :

```
> x[3:5]
```

```
[1] 2557 2594 2600
```

La syntaxe un peu particulière `3:5` signifie en fait la séquence des nombres entiers débutant à 3 et se terminant à 5. ceci est donc équivalent à `c(3,4,5)`. Le même principe s'applique aux variables qualitatives :

```
> vems[2]
```

```
[1] critique ou bas
Levels: critique ou bas normal
```

Pour accéder à des valeurs particulières d'une variable, on se contente donc de fournir une liste de no d'observation (ou index).

1.2.2 Sélection critériée d'observations

Supposons que l'on veuille à présent obtenir le poids des bébés dont la mère pèse moins de 50 kg.

```
> x[y < 50]
```

```
[1] 2557 2594 2600 2637
```

La commande ci-dessus se lit : sélectionner les valeurs de `x` pour lesquelles la condition (logique) `y < 50` est vérifiée. Cette condition peut se visualiser en tapant directement `y < 50` sous R ce qui produit le résultat suivant :

```
 [1] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE
```

Ce principe de sélection sur la base d'un critère externe reste valable lorsque le critère est une variable catégorielle. Si l'on souhaite obtenir la liste des poids des bébés dont la mère est non-fumeur, on tapera simplement la commande suivante :

```
> x[z == "NF"]
```

```
[1] 2523 2551 2622 2637 2637
```

On notera l'usage du double signe égal (==) pour signifier la condition logique « `z` valant `NF` ». Enfin, on peut combiner des critères portant sur des variables du même type ou de différents types à l'aide d'opérateurs logiques, `&` (et) et `|` (ou), principalement. La commande suivante retourne les valeurs de `x` telles que `z` vaut `NF` *et* `y`$<$ 55 (poids des bébés dont la mère de fume pas et pesant 55 kg ou moins).

```
> x[z == "NF" & y <= 55]
```

```
[1] 2637 2637
```

1.3 Représentation et traitement des valeurs manquantes

Dans la réalité, il est rare que l'on dispose de données sans valeurs manquantes. Quelle que soit la manière dont décide de les traiter sur le plan statistique, il est important de s'assurer qu'elles sont bien représentées comme telles par le logiciel statistique.

Reprenons l'exemple précédent. Supposons que la 3^{e} observation pour le poids des bébés est en fait était manquante, ou que l'on souhaite la traiter comme telle. Cette donnée est représentée par un point (•) dans le tableau ci-dessous.

x	2523	2551	•	2594	2600	2622	2637	2637	2663	2665
y	82.7	70.5	47.7	49.1	48.6	56.4	53.6	46.8	55.9	51.4

On aurait très bien représenter cette valeur manquante par une cellule vide, ou n'importe quel autre symbole. En utilisant la même approche que lorsque l'on a défini la variable `x` pour la première fois, une façon de saisir les poids des bébés consisterait à remplacer la valeur manquante par le symbole `NA`, qui est le symbole réservé par R pour coder les valeurs manquantes.

```
> c(2523, 2551, NA, 2594, 2600, 2622, 2637, 2637, 2663, 2665)
```

```
 [1] 2523 2551   NA 2594 2600 2622 2637 2637 2663 2665
```

Puisque la variable `x` a déjà été saisie, on peut simplement la mettre à jour, en l'occurence remplacer le 3^{e} élément par `NA`.

```
> x[3] <- NA
> x
```

```
 [1] 2523 2551   NA 2594 2600 2622 2637 2637 2663 2665
```

On prendra garde au fait que la commande `length` renvoit toujours 10 : il y a bien 10 observations, mais l'une d'elle est manquante. La comamnde `is.na` permet de vérifier quelles sont les valeurs manquantes présentes dans une variable :

```
> is.na(x)
 [1] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> which(is.na(x))
[1] 3
```

Comme on l'a déjà vu dans le cas de la sélection d'observation, les valeurs renvoyées par la commande `is.na` sont des booléens valant vrai lorsque la valeur de `x` est manquante (`NA`), faux sinon. Cette commande renverra donc autant de valeurs que la variable `x` en contient. Pour faciliter l'identification des n° d'observations manquantes et au fond limiter l'information aux seules informations utiles, on peut préferer combiner `is.na` avec la commande `which`. On notera que les commandes se combinent de manière intuitive : la commande `which` utilise le résultat renvoyé par `is.na`, sans que l'on ait besoin de créer une variable auxiliaire pour stocker ce résultat.

Une alternative consiste à dénombrer les cas dits complets avec la commande `complete.cases`

```
> complete.cases(x)
 [1]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

En utilisant le principe de sélection d'observations vu précédemment, la commande suivante renverrra donc les valeurs de `y` pour lesquelles `x` a une valeur non manquante :

```
> y[complete.cases(x)]
[1] 82.7 70.5 49.1 48.6 56.4 53.6 46.8 55.9 51.4
```

On retiendra également que, en présence de valeurs manquantes, il faut explicitement indiquer à R comment calculer les indicateurs statistiques usuels tels que somme, moyenne ou variance :

1.4 Importation et sauvegarde de données

1.4.1 Données univariées

Les trois séries de 10 mesures discutées dans les parties précédentes étaient de taille suffisamment modérée pour permettre leur saisie directement dans R. Cependant, dans la plupart des cas, les données ont déjà été saisies dans un fichier externe et la première étape consiste généralement à les importer dans R.

Voici le contenu du fichier `poids.dat`, qui est un simple fichier texte que l'on peut visualiser avec n'importe quel éditeur de texte :

```
2523 2551 2557 2594 2600 2622 2637 2637 2663 2665
```

Dans un cas simple où l'on désire importe une seule série de mesures, la commande `scan` est suffisante. Il suffit d'indiquer l'endroit où se trouve le fichier de données. Par « endroit » on entend la localisation précise du fichier dans l'arborescence des fichiers de l'ordinateur.

```
> x <- scan("data/poids.dat")
> head(x, n=6)
[1] 2523 2551 2557 2594 2600 2622
```

1.4.2 Données multivariées

Dans les cas un plus complexes, mais plus proches de la réalité, les données sont multivariées avec, en général, les variables organisées en colonnes et les observations en ligne : chaque ligne du fichier représente donc une unité statistique sur laquelle on a collecté plusieurs mesures ou informations (variables ou champs), ces dernières étant séparées l'une de l'autre soit par une virgule, un point-virgule, des espaces ou des tabulations. Dans tous les cas, la commande à utiliser est `read.table` (le séparateur de champ est une espace ou une

tabulation), ou l'un de ses raccourcis `read.csv` (séparateur de type virgule) et `read.csv2` (séparateur de type oint-virgule).

Considérons le fichier `birthwt.dat`, dont les trois premières lignes sont reproduites ci-dessous :

```
0 19 182 2 0 0 0 1 0 2523
0 33 155 3 0 0 0 0 3 2551
0 20 105 1 1 0 0 0 1 2557
```

Comme on le voit, il y a 10 colonnes (soit 10 variables) et chaque valeur est séparée de la suivante par une espace. Le nom des variables ne figure nulle part, mais l'on sait qu'il s'agit des variables suivantes : statut pondéral du bébé à la naissance `low` (= 1 si poids < 2.5 kg, 0 sinon), `age` de la mère (années), `lwt` poids de la mère (en livres), `race` ethnicité de la mère (codée en trois classes, 1 = white, 2 = black, 3 = other), `smoke` (= 1 si consommation de tabac durant la grossesse, 0 sinon), `ptl` (nombre d'accouchements pré-terme antérieurs), `ht` (= 1 si antécédent d'hypertension, 0 sinon), `ui` (= 1 si manifestation d'irritabilité utérine, 0 sinon), `ftv` (nombre de consultations chez le gynécologue durant le premier trimestre de grossesse), `bwt` pour le poids des bébés à la naissance (en grammes). Voici comment ces données peuvent être importées sous R :

```
> bt <- read.table("data/birthwt.dat", header=FALSE)
> varnames <- c("low", "age", "lwt", "race", "smoke", "ptl", "ht", "ui", "ftv", "bwt")
> names(bt) <- varnames
> head(bt)
```

```
  low age lwt race smoke ptl ht ui ftv  bwt
1   0  19 182    2     0   0  0  1   0 2523
2   0  33 155    3     0   0  0  0   3 2551
3   0  20 105    1     1   0  0  0   1 2557
4   0  21 108    1     1   0  0  1   2 2594
5   0  18 107    1     1   0  0  1   0 2600
6   0  21 124    3     0   0  0  0   0 2622
```

Si le fichier de données avait eu la forme suivante,

```
0,19,182,2,0,0,0,1,0,2523
0,33,155,3,0,0,0,0,3,2551
0,20,105,1,1,0,0,0,1,2557
```

qui est typiquement le format d'exportation offert par un tableur comme Excel, alors on aurait simplement remplacé la commande `read.table` par `read.csv` (ou modifié l'option `sep=` dans `read.table`).

1.4.3 Sauvegarde de données dans un fichier externe

Dans le cas univarié comme dans le cas multivarié, on pourra exporter les données traitées sous R à l'aide des commande `write.table` ou `write.csv` qui produisent des fichiers texte identiques à ceux discutés plus haut. Par exemple,

```
> write.csv(bt, file="bt.csv")
```

permettra de sauvegarder le fichier importé avec la commande `read.table` sous la forme d'un fichier où les valeurs des variables sont séparées par des virgule, et le fichier `bt.csv` pourra être visualisé à l'aide d'un tableur.

Si l'on souhaite sauvegarder les données au format R directement, on utilisera plutôt `save(bt, file="bt.RData")`, `RData` étant l'extension réservée au fichier de données R.

1.5 Gestion de données multidimensionnelles

1.5.1 Construction d'un tableau ou stucture de données

Jusqu'à présent, nous avons crée et manipulé différentes variables (`x`, `y` et `z`, principalement)

Une manière plus naturelle de rendre compte de cette association entre les observations est de regrouper ces variables dans une même structure de données, ou tableau, que l'on appelle `data.frame` sous R. Nos 3 variables seront stockées dans 3 colonnes distinctes, les lignes du tableau permettant de bien identifier les différentes unités statistiques pour lesquelles nous disposons de mesures sur ces 3 variables. Voici comment l'on peut procéder sous R :

```
> d <- data.frame(x, y, z)
> names(d) <- c("poids.bébé", "poids.mère", "ethnicité.mère")
> head(d, n=3)
```

```
  poids.bébé poids.mère ethnicité.mère
1       2523       82.7             NF
2       2551       70.5             NF
3       2557       47.7              F
```

On remarquera que l'on en a profité pour renommer le nom des variables. Le 1er individu pèse donc 2523 g et sa mère qui pèse 82,7 kg ne fume pas. Pour afficher ces seules informations, on indexera le tableau nommé `d` par le numéro de ligne correspondante.

```
> d[1,]
```

```
  poids.bébé poids.mère ethnicité.mère
1       2523       82.7             NF
```

Le fait de ne rien préciser pour les numéros de colonne signifie que l'on désire toutes les sélectionner. Au contraire, pour afficher la valeur de la 2e variable pour les deux premières observations, on utilisera

```
> d[c(1,2), 2]
```

```
[1] 82.7 70.5
```

1.5.2 Application

Les données complètes sur le poids des bébés discutées sont disponibles dans les exemples de base fournis avec R, et on peut les importer comme suit avec la commande `data`.

```
> data(birthwt, package="MASS")
> c(nrow(birthwt), ncol(birthwt))
```

```
[1] 189  10
```

```
> names(birthwt)
```

```
 [1] "low"   "age"   "lwt"   "race"  "smoke" "ptl"   "ht"    "ui"    "ftv"   "bwt"
```

```
> head(birthwt, n=2)
```

```
   low age lwt race smoke ptl ht ui ftv  bwt
85   0  19 182    2     0   0  0  1   0 2523
86   0  33 155    3     0   0  0  0   3 2551
```

La variable `birthwt` est un tableau (`data.frame`) regroupant 10 variables, chacune ayant 189 observations. On dispose donc d'informations sur 189 unités statistiques dans cette étude prospective. Pour accéder aux valeurs d'une variable particulière, on utilisera la notation suivante : nom du tableau (`birthwt`) suivi du nom de la variable préfixée du signe dollar (`$`). Par exemple, les 5 premières valeurs pour le poids des bébés (`bwt`) sont ainsi :

```
> birthwt$bwt[1:5]
```

```
[1] 2523 2551 2557 2594 2600
```

La signification de chaque variable a été donnée plus haut. On sait que certaines variables sont strictement binaires (0/1), comme `low`, `smoke`, `ht` et `ui`, alors que les autres variables sont numériques, soit à valeurs discrètes comme `ftv` soit à valeurs supposées continues comme `lwt` ou `bwt`. On sait que dans le cas des variables binaires, une valeur de 1 signifie que le signe est présent (la mère fume, la mère a des antécédents d'hypertension, etc.). Cela ne coûte rien d'ajouter des étiquettes plus informatives à ces variables. Le cas du facteur ethnicité

(race) est plus particulier car il s'agit d'une variable qualitative mais qui est pour l'instant traitée par R comme une variable numérique.

```
> summary(birthwt$race)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    1.00    1.00    1.85    3.00    3.00
```

La commande summary permet de résumer la distribution des variables numériques et catégorielles, et s'applique aussi bien à une variable seule qu'à un tableau de données (appelé data.frame) tel que celui présenté plus haut (birthwt). Voici une manière de recoder les modalités des variables qualitatives pour le tableau birthwt :

```
> ouinon <- c("No", "Yes")
> birthwt$smoke <- factor(birthwt$smoke, labels=ouinon)
> birthwt$race <- factor(birthwt$race, levels=c(1, 2, 3), labels=c("White", "Black", "Other"))
```

On utilisera la même démarche pour low, ht, ui (utiliser les étiquettes ouinon) et on vérifiera que les données finales sont bien dans le format attendu à l'aide de summary qui permet de repérer rapidement les variables considérées comme des variables numériques (résumé numérique avec min, max, moyenne) ou qualitatives (tableau d'effectifs) :

```
> summary(birthwt)
```

```
  low          age            lwt         race    smoke          ptl          ht        ui
 No :130   Min.   :14.0   Min.   : 80   White:96   No :115   Min.   :0.000   No :177   No :161
 Yes: 59   1st Qu.:19.0   1st Qu.:110   Black:26   Yes: 74   1st Qu.:0.000   Yes: 12   Yes: 28
           Median :23.0   Median :121   Other:67             Median :0.000
           Mean   :23.2   Mean   :130                        Mean   :0.196
           3rd Qu.:26.0   3rd Qu.:140                        3rd Qu.:0.000
           Max.   :45.0   Max.   :250                        Max.   :3.000
      ftv             bwt
 Min.   :0.00   Min.   : 709
 1st Qu.:0.00   1st Qu.:2414
 Median :0.00   Median :2977
 Mean   :0.79   Mean   :2945
 3rd Qu.:1.00   3rd Qu.:3487
 Max.   :6.00   Max.   :4990
```

On peut à présent facilement répondre aux questions suivantes :

- quel est le poids moyen des femmes qui fumait durant leur grossesse ?
- combien dénombre-t-on d'antécédents d'hypertension chez les femmes pesant plus de 60 kg (sachant que les mesures du fichier sont exprimées en livres) ?
- quel est le poids minimal des bébés chez les mères n'ayant pas manifesté une irritabilité utérine ?

```
> mean(birthwt$lwt[birthwt$smoke == "Yes"]) ## poids en livres
> table(birthwt$ht[birthwt$lwt/2.2 > 60])   ## poids en kg
> min(birthwt$bwt[birthwt$ui == "No"])
```

Par la suite, plutôt que de répéter systématiquement le nom du data.frame suivi de $ pour accéder à une ou plusieurs variables, comme dans l'exemple ci-dessus, on préférera utiliser la commande with qui consiste à indiquer dans quel data.frame on souhaite chercher des variables.

```
> with(birthwt, lwt[1:5])
> with(birthwt, mean(lwt[smoke == "Yes"]))
```

Ce qu'il faut retenir

- Une variable sera représentée sous la forme d'une liste de nombres ou de caractères (simples ou multiples).
- Les variables seront généralement arrangées dans un tableau rectangulaire (data.frame).
- Les opérations arithmétiques usuelles opèrent sur chaque élément (observation) d'une variable.
- Les valeurs manquantes doivent être traitées de manière explicite, la plupart du temps à travers les options des commandes R.

Cours 2. Statistiques descriptives et estimation

Sommaire

À partir de ce chapitre et jusqu'à celui traitant de la régression logistique, on travaillera avec le même jeu de données sur les poids à la naissance. L'objectif de ce cours est de présenter les principaux outils disponibles sous R pour la description univariée d'une variable numérique ou catégorielle. Le résumé d'une variable peut se faire de manière numérique : indicateurs de tendance centrale et de dispersion pour une variable numérique, tableau d'effectifs ou de fréquences pour une variable catégorielle. La distribution des valeurs observées pour une variable peut également être visualisée sous forme d'un graphique, soit en utilisant les données individuelles (diagramme de dispersion univariée) soit sous en utilisant une forme de résumé de l'information (histogramme, boîte à moustaches, diagrammes en barres ou en points).

Commandes essentielles

`mean` : moyenne arithmétique
`quantile` : fractiles de la distribution
`var` : variance
`xtabs` : tableau d'effectifs
`stripplot` : diagramme de dispersion univarié
`densityplot` : courbe de densité non-paramétrique
`barchart` : diagramme en barres
`pnorm` : valeur de probabilité pour la loi normale
`median` : médiane
`sum` : somme des valeurs prises par une variable
`sd` : écart-type
`prop.table` : tableau de fréquences
`histogram` : histogramme
`bwplot` : boîte à moustaches
`dotplot` : diagramme en points
`qnorm` : fractile de la loi normale

2.1 Résumer une variable numérique

2.1.1 Tendance centrale et forme de la distribution

Considérons la variable `bwt` qui représente les poids à la naissance des nouveaux-nés. Les poids sont exprimés en grammes et il n'y a pas de valeurs manquantes. On peut le vérifier rapidement en comptant le nombre de valeurs pour lesquelles la commande `is.na` renvoit vrai (`TRUE`) :

```
> sum(is.na(birthwt$bwt))
```

```
[1] 0
```

Dans l'expression ci-dessus, on combine deux commandes : la commande `is.na` renvoit la valeur `TRUE` si un des éléments de `bwt` est considéré comme une valeur manquante, `FALSE` sinon. Le résultat de cette commande contient donc autant d'éléments que la variable `bwt` et on dénombre le nombre de valeurs valant `TRUE`, celles-ci étant représentées de manière interne comme des nombres valant 1 (d'où l'usage de la commande `sum`).

Les principaux indicateurs de tendance centrale, moyenne et médiane, peuvent être obtenus avec les commandes `mean` et `median`. Elles supposent qu'il n'y a pas de valeur manquante, le cas échéant il est nécessaire de rajouter l'option `na.rm=TRUE`. Par exemple,

```
> mean(birthwt$bwt)
[1] 2945
> median(birthwt$bwt)
[1] 2977
```

Comme on l'a vu dans le chapitre précédent, on peut également utiliser directement la commande `summary` qui a l'avantage d'afficher l'étendue des valeurs observées (minimum et maximum, voir les commandes `min`, `max` et `range`) et les quartiles (la médiane constituant le 2e quartile) :

```
> summary(birthwt$bwt)
```

Les quartiles peuvent être retrouvés de la manière suivante :

```
> quantile(birthwt$bwt)
  0%  25%  50%  75% 100%
 709 2414 2977 3487 4990
```

ou

```
> quantile(birthwt$bwt, probs=c(0.25,0.50,0.75))
```

On procéderait de même pour les déciles

```
> quantile(birthwt$bwt, probs=seq(0.1, 1, by=0.1))
```

à ceci près que plutôt que d'écrire chaque décile (10 %, 20 %, etc.) on a préféré laisser R générer lui-même la séquence de déciles à l'aide de la commande `seq`.

En ce qui concerne l'estimation de l'asymétrie ou de l'aplatissement de la distribution, il n'existe pas de commandes de base sous R, mais elles sont disponibles dans le package `e1071` (`skewness` et `kurtosis`).

2.1.2 Indicateurs de dispersion

Pour les mesures de dispersion, on dispose déjà des quartiles qui permettent d'estimer l'intervalle inter-quartile (50 % des observations). On pourrait donc très bien calculer cet intervalle comme suit

```
> quantile(birthwt$bwt, probs=c(0.75)) - quantile(birthwt$bwt, probs=c(0.25))
 75%
1073
```

ou plus simplement

```
> diff(quantile(birthwt$bwt, probs=c(0.25, 0.75)))
 75%
1073
```

L'affichage `75%` au-dessus de la valeur de l'intervalle inter-quartile est dû au fait que R« nomme » explicitement les quantiles comme on l'a vu en utilisant `quantile(birthwt$bwt`. Une alternative consisterait à utiliser directement la commande `IQR`.

R fournit également un estimateur de l'écart-type et de la variance théorique (avec un dénominateur à $n-1$) avec les commandes `sd` et `var`.

```
> var(birthwt$bwt)
[1] 531753
> sd(birthwt$bwt)
[1] 729
> sd(birthwt$bwt)^2
```

```
[1] 531753
```

Les mêmes précautions d'usage concernant les valeurs manquantes s'appliquent (`na.rm=TRUE`).

2.2 Résumer une variable catégorielle

Concernant les variables qualitatives, que les modalités de la variable soient ordonnées ou non, on peut toujours fournir un tableau d'effectifs ou de fréquences. La commande `table` qui a été utilisée au chapitre précédent permet de fournir un tableau d'effectifs. Comme les commandes `mean` ou `median`, il est nécessaire de préciser que faire en présence de valeurs manquantes. Par défaut, R ne les affichera pas comme une catégorie à part ; sinon, il faut ajouter l'option `useNA="always"`.

```
> table(birthwt$race)
```

```
White Black Other
   96    26    67
```

La commande `summary` fournit essentiellement les mêmes informations, mais la commande `table` sera plus souple d'utilisation par la suite. Pour obtenir les fréquences relatives, plutôt que de diviser les effectifs de la table ci-dessus par l'effectif total (`table(birthwt$race)/nrow(birthwt$race)`, ce qui suppose l'absence de données manquantes), on préférera la commande `prop.table` en combinaison de `table`. Le tableau de fréquences relatives est donc

```
> prop.table(table(birthwt$race))
```

```
White Black Other
0.508 0.138 0.354
```

On pourrait ajouter `*100` à l'expression ci-dessus pour exprimer les résultats en pourcentages plutôt qu'en fréquences relatives. Notons que l'on peut sauvegarder les résultats intermédiaires, y compris le résultat de commandes telles que `table`, dans des variables R. Cela facilite leur réutilisation et améliore la lisibilité des commandes utilisant ces mêmes résultats.

```
> tab <- table(birthwt$race)
> tab
> prop.table(tab)
```

À partir d'un tableau d'effectif, on peut retrouver le mode en recherchant l'effectif le plus élevé, comme illustré ci-après.

```
> which.max(tab)
```

```
White
    1
```

Ici, `tab` représente bien le tableau construit avec la commande `table(birthwt$race)`. Le résultat renvoyé par `which.max` nous dit que la classe modale est en fait la première classe de la variable `race` (`White`, en 1re position dans la liste des modalités de la variable).

Il existe une autre commande R qui permet de réaliser le même type d'opération (tris à plat et tris croisés, ces derniers étant présentés au chapitre suivant) : `xtabs`.

```
> xtabs(~ birthwt$race)
```

Cette commande fait intervenir une nouvelle notation, l'usage du symbole `~` qui est utilisé pour décrire des relations entre variables, par exemple pour exprimer une relation du type : une variable numérique, `x`, décrite par une variable catégorielle, `z` (`x ~ z`). Dans le cas univarié, la notation `~ birthwt$race` indique à R de décrire les variations de la variable `race` du tableau `birthwt`, ce qui pour une variable qualitative équivaut à la distribution des effectifs par classe ou modalité de la variable. On appelle cette notation une « notation par formule ».

Dans certaines situations, on peut vouloir résumer une variable numérique à l'aide d'un tableau d'effectifs en considérant différents intervalles de classe (borne inférieure et supérieure entre lesquelles les valeurs de la variable numérique se situent) : on se retrouve alors avec une variable catégorielle avec laquelle on peut utiliser `table` ou `xtabs`. Considérons l'âge des mères (`age`, qui varie entre 14 et 45 ans), et les intervalles de classe suivants : $[15,19]$, $[20,24]$, $[25,29]$, $[30,45]$. Tels qu'ils sont définis, ces intervalles incluent les valeurs indiquées pour leurs bornes inférieures et supérieures. Voici une manière de construire le tableau d'effectifs correspondant à cette répartition des individus dans des classes d'âge mutuellement exclusives :

```
> table(cut(birthwt$age, breaks=c(15,19,24,29,45), include.lowest=TRUE))
```

```
[15,19] (19,24] (24,29] (29,45]
     48      69      42      27
```

On remarquera que l'usage de la commande `cut` nécessite d'indiquer à R comment découper les valeurs de la variable numérique, et en particulier si les bornes inférieures ou supérieures doivent être inclusives. Dans l'exemple ci-dessus, on indique que les intervalles doivent inclure la borne inférieure du 1er intervalle (`include.lowest=TRUE`).

2.3 Représenter graphiquement la distribution d'une variable

Toutes les commandes graphiques utilisées dans le cours sont fournies par le package `lattice` qui est disponible avec l'installation de base du logiciel R. Pour pouvoir utiliser ces commandes, il est nécessaire de taper

```
> library(lattice)
```

en début de session R, ou en tout cas avant d'utiliser une des commandes discutées dans les paragraphes suivants. On ne tapera cette commande qu'une seule fois pour chaque session R.

2.3.1 Cas des variables numériques

Pour représenter graphiquement l'ensemble des observations, on peut utiliser un simple diagramme de dispersion à une dimension à l'aide de la commande `stripplot` dont un exemple d'utilisation est fourni ci-après avec une variable supposée être appelée `poids`. Lorsque le nombre d'observations n'est pas trop important, cela permet de visualiser rapidement les données plus ou moins extrêmes et la tendance centrale.

```
> stripplot(~ poids)
```

On retrouve ici la notation par formule présentée plus haut : ici, dans un contexte univarié, on indique à R de décrire les variations de la variable `poids`, ce qui pour une variable numérique équivaut à la distribution de l'ensemble des observations, sans considération d'un quelconque facteur de classification ou stratification. Il est également possible d'utiliser ce type de représentation graphique avec des données de taille importante, mais dans ce cas il y a un risque que les points se chevauchent rendant l'interprétation de la distribution univariée plus délicate. Une alternative consiste à ajouter un léger décalage, horizontal ou vertical selon l'axe choisi pour représenter la variable mesurée, aux points sur le graphique. Voici une illustration avec les poids des bébés :

```
> stripplot(~ birthwt$bwt, jitter.data=TRUE)
```

Le décalage vertical des points est ici contrôlé par l'option `jitter.data=` (on ajoute en fait un aléa gaussien pour chaque point sur l'axe des ordonnées), et on peut augmenter l'amplitude du décalage avec l'option `amount=`.

Une représentation bien connue pour les variables continues est un histogramme des effectifs ou des fréquences en définissant une série d'intervalles de classe pour les valeurs prises par la variable numérique. R offre différentes options pour le choix du calcul des intervalles de classe, et généralement ils sont satisfaisants. On peut toujours spécifier manuellement le nombre d'intervalles (option `nint=`) ou les classes elles-mêmes (option `breaks=`). Par exemple, pour découper les effectifs sur des intervalles de taille 500, on utiliserait une

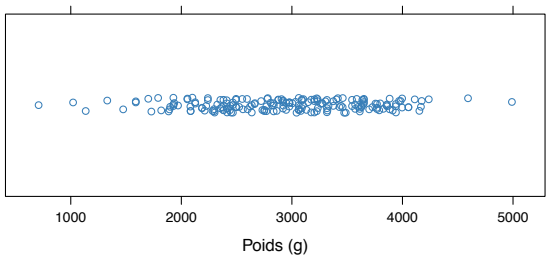
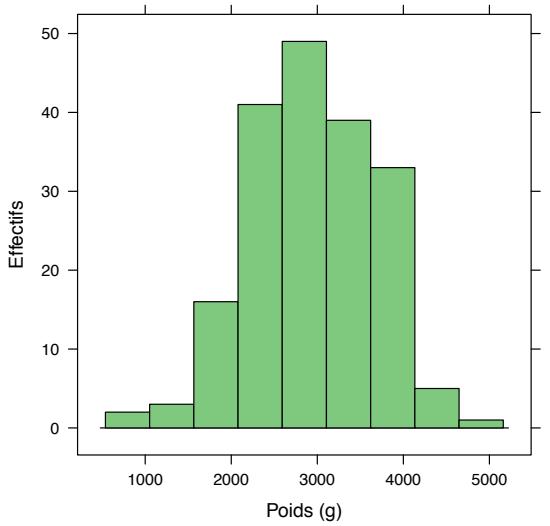


Diagramme en points pour la distribution des poids des bébés.

```
stripplot(~ bwt, data=birthwt, jitter.data=TRUE,
amount=0.05, xlab='Poids (g)')
```

option telle que `breaks=seq(700, 5000, by=500)`, les bornes 700 et 5000 étant légèrement en dehors de l'intervalle des valeurs observées pour la variable `bwt`. Par défaut, R affiche un histogramme de proportions, ce que l'on peut changer en ajoutant l'option `type="count"` pour obtenir un histogramme des effectifs. Voici le résultat pour la variable `bwt` :

```
> histogram(~ birthwt$bwt, type="count")
```



Histogramme pour la distribution des poids des bébés.

```
histogram(~ bwt, data=birthwt, type='count',
xlab='Poids (g)', ylab='Effectifs')
```

La commande précédente peut être également formulée de la manière suivante :

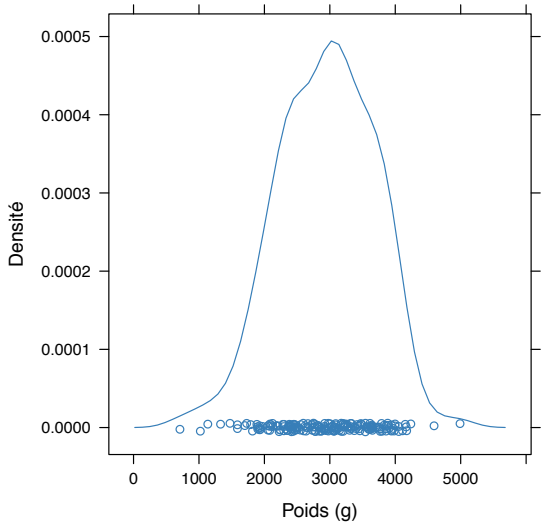
```
> histogram(~ bwt, data=birthwt, type="count")
```

Ici, on introduit un concept important dans R : de nombreuses commandes offrent la possibilité de désigner le tableau (`data.frame`) dans lequel se trouvent les variables à l'aide de l'option `data=`, et d'utiliser directement le nom des variables plutôt que de les préfixer systématiquement avec le nom du tableau suivi du caractère `$`. Avec la notation par formule, ces deux concepts constituent la base de la description des données statistiques sous R. Un autre intérêt de cette approche est la possibilité de ne considérer qu'un sous-ensemble des observations du tableau de données spécifié par l'option `data=`. Pour cela, il suffit d'ajouter l'option `subset=` et d'indiquer une condition logique permettant de filtrer les unités statistiques, ce filtre pouvant porter sur la variable d'intérêt, par exemple `subset = bwt > 3000` (seulement les poids des bébés pesant plus de 3 kg), ou sur n'importe quelle variable (ou combinaison de variables) disponibles dans le `data.frame`, par exemple `smoke == "Yes"` (seulement les poids des bébés dont la mère fume).

```
> histogram(~ bwt, data=birthwt, subset=smoke == "Yes", type="count")
```

Une alternative à l'histogramme est la courbe de densité non-paramétrique qui permet de s'affranchir du problème du choix des intervalles de classe. En l'occurence, plutôt que de regrouper les observations dans des classes définies à l'avance (manuellement ou à l'aide d'un algorithme spécifique), on peut tout à fait utiliser des méthodes plus sophistiquées pour estimer la courbe de densité, en contrôlant le degré de « lissage » de cette approximation.

```
> densityplot(~ bwt, data=birthwt)
```



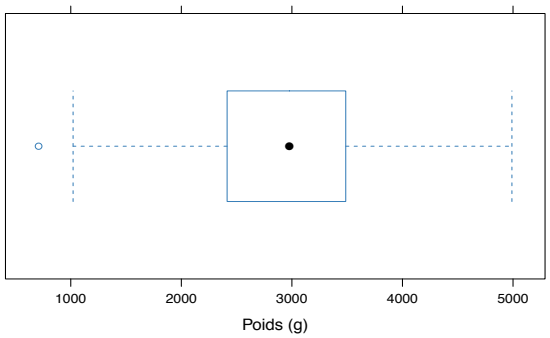
Courbe de densité estimée pour la distribution des poids des bébés.

```
densityplot(~ bwt, data=birthwt, xlab='Poids (g)',
ylab='Effectifs')
```

La taille de la fenêtre de lissage de la courbe se contrôle avec l'option `bw=` ; plus elle est élevée, plus les variations locales sont atténuées (par exemple, avec `bw=1000` la courbe apparaît beaucoup plus proche d'une courbe gaussienne pour les poids des bébés).

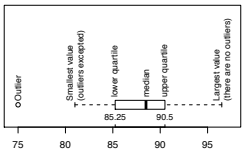
Enfin, une dernière façon de représenter graphiquement la distribution d'une variable numérique est un diagramme de type boîte à moustaches. Dans ce type de représentation, la « boîte » contient 50 % des observations de sorte que ces deux bords représentent le 1er et le 3e quartile. La médiane est représentée par un point à l'intérieur de cette boîte. Les « moustaches » représentent les valeurs minimale et maximale de la distribution, sauf dans le cas où certaines observations sont situées à plus de 1.5 × IQR (IQR = intervalle inter-quartile) du 1er ou du 3e quartile auquel cas ces dernières sont indiquées explicitement sur le graphique et les « moustaches » représentent alors les bornes des valeurs limites (1.5 × IQR du 1er ou 3e quartile) considérées comme limites pour qualifier une observation d'extrême (diagramme de Tukey).

```
> bwplot(~ bwt, data=birthwt)
```



Représentation de type boîte à moustache pour la distribution des poids des bébés.

```
bwplot(~ bwt, data=birthwt, xlab='Poids (g)')
```



2.3.2 Cas des variables catégorielles

Dans le cas des variables catégorielles on utilisera soit des diagrammes en barres soit des diagrammes en point (diagramme de Cleveland), en évitant dans la mesure du possible les représentations circulaire de type « boîte à camembert » qui ne fournissent de toute façon pas plus d'informations que les diagrammes précédents.

Dans les deux cas, il est nécessaire de travailler avec les données tabulées, c'est-à-dire un tableau d'effectifs construit à l'aide de `table` ou `xtabs`. On peut également associer ces commandes à `prop.table` pour obtenir

un tableau de fréquences relatives. Comme on l'a vu plus haut, la commande `xtabs` permet une notation par formule ce qui est parfois plus expressif.

```
> barchart(xtabs(~ bwt, data=birthwt))
```

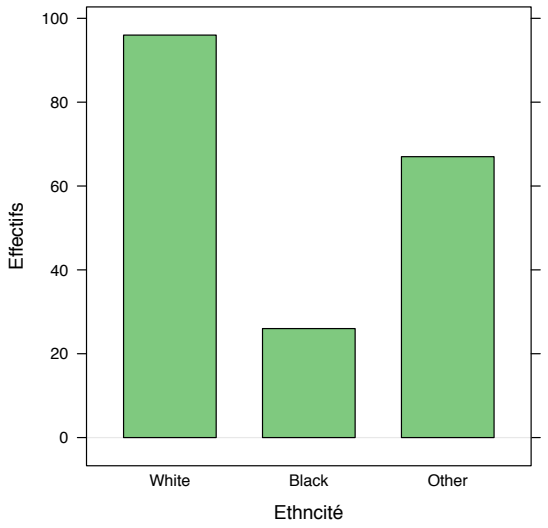


Diagramme en barres pour la distribution des effectifs selon l'ethnicité de la mère.
`barchart(xtabs(~ race, data=birthwt), xlab='Ethnicité', ylab='Effectifs', horizontal=FALSE)`

Il est possible de changer l'orientation des barres (par défaut horizontales) grâce à l'option `horizontal=FALSE` (barres verticales). Les diagrammes en points sont construits exactement de la même manière, avec la commande `dotplot` qui partage une grande partie des options de base de la commande `barchart` (en particulier l'option `horizontal=FALSE` pour un diagramme orienté verticalement).

```
> dotplot(xtabs(~ bwt, data=birthwt))
```

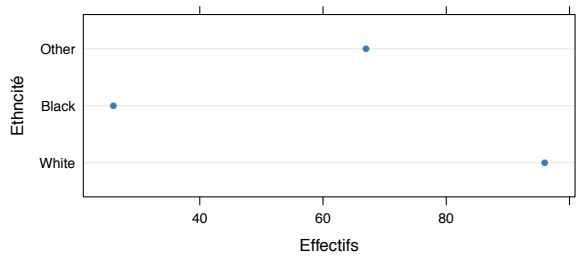


Diagramme en barres pour la distribution des effectifs selon l'ethnicité de la mère.
`dotplot(xtabs(~ race, data=birthwt), xlab='Effectifs', ylab='Ethnicité')`

2.4 Estimation par intervalles pour une moyenne ou une proportion

2.4.1 Intervalles de confiance pour une moyenne

Comme on l'a vu, les commandes `mean` et `summary` fournissent toutes les deux une estimation de la moyenne (arithmétique) pour une variable numérique. Pour obtenir des intervalles de confiance à $100(1-\alpha)$ %, où α désigne le risque de première espèce (habituellement 5 %), et en se référant à une distribution gaussienne (loi « normale »), il faut une estimation de l'erreur standard et du quantile correspondant. Pour ce dernier, plutôt que d'utiliser l'approximation 1.96 (pour $\alpha = 5$ %), on peut obtenir directement sa valeur sous R en utilisant la commande `qnorm` qui fournit les fractiles d'une loi normale standardisée $\mathcal{N}(0;1)$.

```
> qnorm(0.975)
```

```
[1] 1.96
```

On pourra vérifier que la commande `1-pnorm(1.96)` renvoit l'aire sous la courbe de densité pour la queue de distribution à droite de la loi normale.

Quant à l'erreur standard, définie comme le rapport entre l'écart-type estimé sur les données et la racine carrée de la taille de l'échantillon, on l'obtiendra simplement comme

```
> sd(birthwt$bwt) / sqrt(length(birthwt$bwt))
```

```
[1] 53
```

Comme on l'a vu au chapitre précédent, plutôt que de répéter systématiquement le nom du `data.frame` suivi du symbole $ pour travailler avec une variable donnée, on peut tout à fait remplacer l'expression ci-dessus par

```
> with(birthwt, sd(bwt) / sqrt(length(bwt)))
```

ce qui indique à R de travailler avec la variable `bwt` disponible dans le tableau `birthwt`. On rappelle qu'il n'y a pas de valeurs manquantes dans ce jeu de données et donc la commande `length` renvoit bien la taille de l'échantillon. En combinant les deux résultats précédents, on obtient assez facilement les bornes inférieures et supérieures d'un intervalle de confiance à 95 % pour la moyenne des poids des bébés dans cet échantillon.

```
> mean(birthwt$bwt) - qnorm(0.975) * sd(birthwt$bwt)/ sqrt(length(birthwt$bwt))
> mean(birthwt$bwt) + qnorm(0.975) * sd(birthwt$bwt)/ sqrt(length(birthwt$bwt))
```

On peut stocker ces deux résultats dans des variables auxiliaires et les combiner ensemble pour faciliter la lecture :

```
> bwt.lci <- mean(birthwt$bwt) - qnorm(0.975) * sd(birthwt$bwt)/ sqrt(length(birthwt$bwt))
> bwt.uci <- mean(birthwt$bwt) + qnorm(0.975) * sd(birthwt$bwt)/ sqrt(length(birthwt$bwt))
> c(bwt.lci, bwt.uci)
```

```
[1] 2841 3049
```

Notons que l'on aurait très bien pu ramener le calcul des deux bornes de l'intervalle de confiance à une seule commande :

```
> mean(birthwt$bwt) + c(-1,1) * qnorm(0.975) * sd(birthwt$bwt)/ sqrt(length(birthwt$bwt))
```

```
[1] 2841 3049
```

Le package `epiR` dispose d'une commande spécifique pour ce type de calcul : `epi.conf`.

Dans le cas des petits échantillons, on peut préférer recourir à une distribution de Student pour construire l'intervalle de confiance associé à une moyenne. Les commandes `pt` et `qt` peuvent être utilisées en lieu et place de `pnorm` et `qnorm` pour obtenir les valeurs de probabilité ou de fractile d'une loi t pour ν degrés de liberté. Les degrés de liberté pour la distribution de Student sont contrôlés par l'option `df=`. Le fractile associé à une probabilité de 0.025 à droite d'une distribution à 18 degrés de liberté est ainsi :

```
> qt(0.975, 18)
```

```
[1] 2.1
```

Comme on le verra dans le chapitre suivant, une autre alternative pour obtenir ce type d'intervalles de confiance est d'utiliser directement une procédure de test, `t.test`, qui renvoit le résultat d'un test d'hypothèse et fournit l'intervalle de confiance associé à la statistique de test. Dans le cas présent, on utiliserait un test t pour un échantillon ($H_0 : \mu = 0$).

2.4.2 Intervalles de confiance pour une proportion

En utilisant une approximation normale pour l'estimation de l'intervalle de confiance d'une proportion, on procéderait exactement de la même manière que celle décrite ci-dessus, en remplaçant les estimations de la moyenne par celle de la fréquence p (obtenue à partir d'une commande `table`, par exemple) et de l'erreur standard par $\sqrt{p(1-p)/n}$. Il s'agit, comme dans le cas précédent, d'une approximation qui est valable essentiellement pour les grands échantillons.

Avec les données sur le statut fumeur des mères (variable `smoke`), un intervalle de confiance à 95 % pour la proportion de mères fumant durant leur grossesse peut donc être formulé ainsi :

```
> tab <- prop.table(table(birthwt$smoke))
> p <- tab[2]
> n <- nrow(birthwt)
> p + c(-1,1) * qnorm(0.975) * sqrt((p*(1-p))/n)
```

```
[1] 0.322 0.461
```

La séquence de commandes ci-dessus permet d'illustrer quelques-unes des notions évoquées depuis le 1er cours : on peut tout à fait stocker le résultat d'une commande `table` ou `prop.table` dans une variable auxiliaire. Ici, il s'agit d'un tri simple de la variable `smoke` et donc la variable `p` contient deux valeurs :

```
> tab
```

```
   No   Yes
0.608 0.392
```

la deuxième correspondant à la proportion de mères fumeuses, et `tab[2]` permet de récupèrer cette valeur. Une alternative consisterait à utiliser la syntaxe `tab["Yes"]` (nom de la colonne du tableau au lieu du numéro de colonne). Il n'y a pas de données manquantes, donc `length(birthwt$bwt)` et `nrow(birthwt)` renverront essentiellement la même information : la taille de l'échantillon. On en profite pour associer cette quantité à une variable appelée `n`. Le reste correspond à l'expression de l'intervalle de confiance $p \pm 1.96\sqrt{p(1-p)/n}$ en langage R.

La commande `epi.conf(cbind(74,115), ctype="prop.single")` en utilisant le package `epiR` fournirait essentiellement le même résultat, de même que la commande `prop.test` qui sera présentée au chapitre suivant, mais qui peut d'ores et déjà être testée sur les données ci-dessus en consultant l'aide en ligne.

Ce qu'il faut retenir

- La commande `summary` fournit l'essentiel des informations concernant la forme de la distribution d'une variable numérique et un tableau d'effectif pour les variables catégorielle. La plupart de ces informations peut être obtenue individuellement à l'aide de comamndes spécifiques telles que `mean`, `sd`, `quantile` ou `table`.
- Une notation par formule est utilisée pour décrire les variations d'une variable, et celle-ci sert de point d'entrée aux commandes graphiques de base pour représenter la distribution d'une variable numérique ou catégorielle.
- Pour construire des intervalles de confiance asymptotiques pour une moyenne ou une proportion estimée à partir des données observées, on peut traduire directement leur formulation mathématique en langage R à partir de la commande `qnorm` et du calcul de l'erreur standard associée à ces quantités.

Cours 3. Mesures et tests d'association entre deux variables

Sommaire

Dans ce cours, on s'intéressera plus particulièrement à quantifier (direction et magnitude) et tester l'association entre deux variables jouant un rôle symétrique ou non. Dans un premier temps, on discutera de la comparaison de moyennes pour deux échantillons, indépendants ou non (une variable joue le rôle de variable réponse, l'autre de variable explicative), à l'aide du test de Student, et son alternative non-paramétrique (test de Wilcoxon-Mann-Whitney pour échantillons indépendants ou non). Dans un second temps, on s'intéressera aux tableaux de contingence à deux entrées (échantillons indépendants) et aux mesures d'association classique (chi-deux) ou plus spécifique de l'épidémiologie (odds-ratio, risque relatif), incluant le cas des échantillons non-indépendants (test de McNemar). Dans ces situations, les variables peuvent jouer un rôle symétrique ou non (cas du risque relatif). Avant de décrire ces procédures de test, on discutera les commandes R qui peuvent être utilisées pour résumer une structure de données composée de deux variables.

Commandes essentielles

`t.test` : test de Student
`prop.test` : test pour deux proportions
`binom.test` : test binomial
`fisher.test` : test de Fisher
`melt` : transformation d'un tableau de données
`wilcox.test` : test de Wilcoxon-Mann-Whitney
`chisq.test` : test du χ^2 de Pearson
`oddsratio` et `epi.2by2` : odds-ratio et risque relatif
`var.test` : test d'égalité de deux variances
`margin.table` : tableau d'effectifs marginaux
`tapply` : applique une commande pour chaque niveau d'un facteur

3.1 Statistiques descriptives bivariées

3.1.1 Décrire une variable numérique par d'une variable qualitative

Considérons les variables âge de la mère (`age`) et poids des bébés (`low`). On souhaite estimer l'âge moyen des mères selon le statut pondéral du nouveau-né ($<$ 2500 grammes ou dans les normes). On peut très bien calculer les moyennes de groupe en sélectionnant les observations appartenant à chacun des groupes, ou niveaux (`levels`) du facteur d'étude. Par exemple, dans l'étude sur les poids à la naissance l'âge moyen des mères de bébés nés avec un poids inférieur à 2,5 kg vaut

```
> with(birthwt, mean(age[bwt < 2500]))
```

```
[1] 22.3
```

ou de manière équivalente

```
> with(birthwt, mean(age[low == "Yes"]))
```

On procèderait de la même manière pour calculer l'âge moyen des mères ayant eu un enfant dans les normes du point de vue du poids de naissance (`mean(age[low == "No"])`).

Cependant, il est souvent plus commode de grouper ce genre d'opérations (calcul d'une statistique quelconque) à l'aide de la commande `tapply` en indiquant le facteur sur lequel on souhaite opérer.

```
> with(birthwt, tapply(age, low, mean))
```

```
  No  Yes
23.7 22.3
```

Dans l'expression ci-dessus, on utilise la commande `with` pour indiquer à R dans quel `data.frame` chercher les variables. La commande `tapply` fonctionne sur le principe suivant : on indique le nom de la variable réponse sur laquelle on souhaite faire des calculs, puis le nom de la variable de classification (obligatoirement de type `factor`) et enfin la commande à utiliser pour faire les calculs. Le troisième argument de la commande `tapply` peut être une commande R, par exemple `mean`, `sum`, `summary`, ou n'importe quelle combinaison de commandes. On pourrait très bien définir une « fonction » qui calcule moyenne et écart-type, comme ceci

```
> f <- function(x) c(m=mean(x), s=sd(x))
```

et l'utiliser en lieu et place de la commande `mean` :

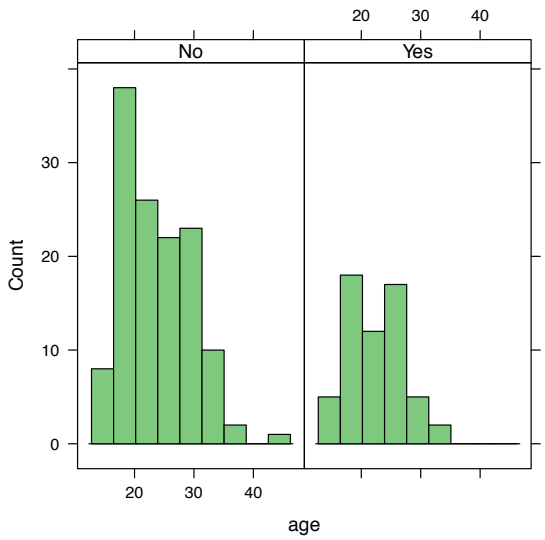
```
> with(birthwt, tapply(age, low, f))
```

```
$No
    m     s
23.66  5.58

$Yes
    m     s
22.31  4.51
```

On peut examiner la distribution des âges selon le statut pondéral à l'aide d'un histogramme, en indiquant la variable de stratification (`low`) à l'aide de l'opérateur `|`. Il s'agit encore une fois d'une notation par formule, déjà évoquée au chapitre 2 : ici, on cherche à décrire les variations de la variable age selon les niveaux de `low`. Le conditionnement s'exprime à l'aide du symbole `|`. L'avantage de cette approche est que R utilisera le même système d'unités pour l'axe des ordonnées, ce qui facilite souvent la comparaison entre les distributions conditionnelles.

```
> histogram(~ age | low, data=birthwt, type="count")
```

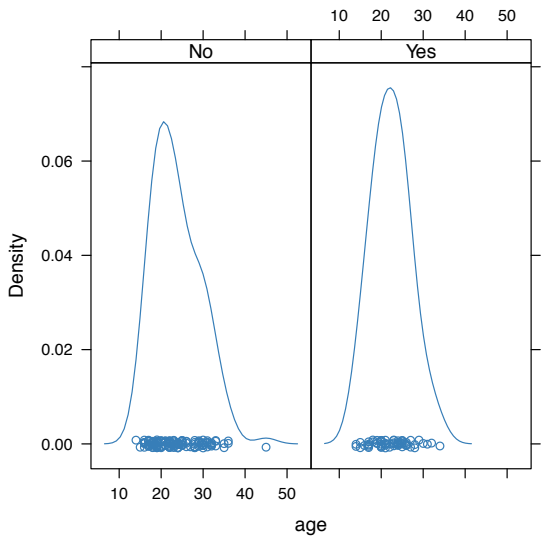


Histogramme pour la distribution de l'âge des mères conditionnellement au statut pondéral à la naissance.

Les niveaux du facteur de classification sont automatiquement indiqués dans la partie supérieure de chaque graphique.

On pourrait tout aussi bien utiliser des courbes de densité au lieu d'histogrammes. La commande à utiliser est alors `densityplot` et les options restent strictement identiques à celles utilisées avec `histogram`, à l'exception de l'option `type="count"`. Dans l'illustration suivante, on a précisé une option `bw=2.5` pour indiquer à R de considérer une fenêtre de lissage assez grande, donc moins sensible aux variations locales des âges.

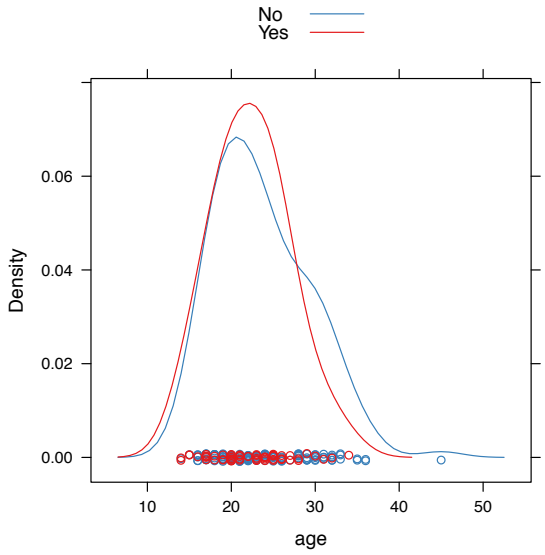
```
> densityplot(~ age | low, data=birthwt, bw=2.5)
```



Courbes de densité pour la distribution de l'âge des mères conditionnellement au statut pondéral à la naissance.

Dans ce cas précis, une formulation alternative pourrait être :

```
> densityplot(~ age, data=birthwt, groups=low, bw=2.5, auto.key=TRUE)
```



Représentation graphique des courbes de densité superposées (`group=low` au lieu de `| low`) pour la distribution de l'âge des mères selon le statut pondéral à la naissance.

Dans ce cas, il est nécessaire d'ajouter une légende pour identifier les deux sous-échantillons, et l'option `auto.key=TRUE` fournit une légende adapatée avec un placement automatique (par défaut, en haut au-dessus du graphique).

La variable `low` reste le facteur de classification mais au lieu de tracer la courbe de densité de chacun des deux groupes dans deux graphiques séparés, l'option `groups=` permet de superposer les courbes sur le même graphique. Il est alors souvent nécessaire d'ajouter une légende pour bien identifier les deux groupes d'individus, ce que l'on peut faire en ajoutant l'option `auto.key=TRUE` : R génère alors automatiquement une légende correspondant aux éléments graphiques utilisés, ici des points et des lignes de couleur différente.

3.1.2 Décrire deux variables qualitatives

Les commandes `table` et `xtabs`, utilisées dans les chapitres précédents pour résumer une variable qualitative sous forme d'un tableau d'effectifs, peuvent être utilisées pour construire un tableau croisant les modalités de deux variables qualitatives dans un tableau de contingence. Si l'on s'intéresse à la répartition des effectifs par modalités des variables `low` et `smoke` (statut pondéral et mères fumeuses), on utilisera :

```
> with(birthwt, table(low, smoke))
```

```
     smoke
low   No Yes
  No  86  44
  Yes 29  30
```

Avec la commande `xtabs`, on écrirait

```
> xtabs(~ low + smoke, data=birthwt)
```

Les effectifs marginaux s'obtiennent avec la commande `margin.table` en précisant l'option `margin=` (1, totaux lignes ; 2, totaux colonnes).

```
> margin.table(xtabs(~ low + smoke, data=birthwt), margin=1)
```

```
low
 No Yes
130  59
```

On retrouve donc facilement la répartition des bébés dans les deux catégories de poids (`low`) avec $n = 29 + 30 = 59$ bébés ayant un poids inférieur à la norme dans cet échantillon.

On peut toujours utiliser la commande `prop.table` pour convertir le tableau d'effectifs en tableau de fréquences : par défaut, R remplacera chaque cellule du tableau de contingence par la valeur de l'effectif rapporté à l'effectif total (il s'agit donc de fréquences conditionnelles).

```
> prop.table(xtabs(~ low + smoke, data=birthwt))
```

```
     smoke
low      No   Yes
  No  0.455 0.233
  Yes 0.153 0.159
```

Mais, il est également possible de calculer des fréquences marginales grâce à l'option `margin=`, comme dans l'illustration suivante :

```
> prop.table(xtabs(~ low + smoke, data=birthwt), margin=1)
```

```
     smoke
low      No   Yes
  No  0.662 0.338
  Yes 0.492 0.508
```

Cette commande permet donc de connaître la fréquence de mères qui fumaient pendant la grossesse parmi l'ensemble des bébés avec un poids inférieur à la norme ($n = 29 + 30 = 59$).

Comme dans le cas univarié, il est tout à fait possible de représenter graphiquement la distribution des effectifs pour deux variables qualitatives en utilisant soit un diagramme en barres (`barchart`, soit un diagramme en points (diagramme de Cleveland, `dotplot`). Dans les deux cas, ces commandes attendent un tableau d'effectifs et non une liste de variables reliées par une notation par formule. Voici un exemple avec les variables `low` et `smoke` pour un diagramme en barres (graphique page suivante).

```
> barchart(xtabs(~ low + smoke, data=birthwt), stack=FALSE, auto.key=TRUE, ylab="low")
```

3.2 Comparaisons de deux moyennes de groupe

Les procédures de test « simples » sont généralement accessibles à partir de commandes R incluant le mot `test` : `t.test` (test de Student), `chisq.test` (test du χ^2 de Fisher), etc. Dans certains cas, elles reposent sur une notation par formule R selon laquelle on désigne une variable réponse « décrite » par une variable explicative ; par exemple, `y ~ x` signifiera généralement la réponse `y` décrite ou fonction de la variable (`factor`) `x`.

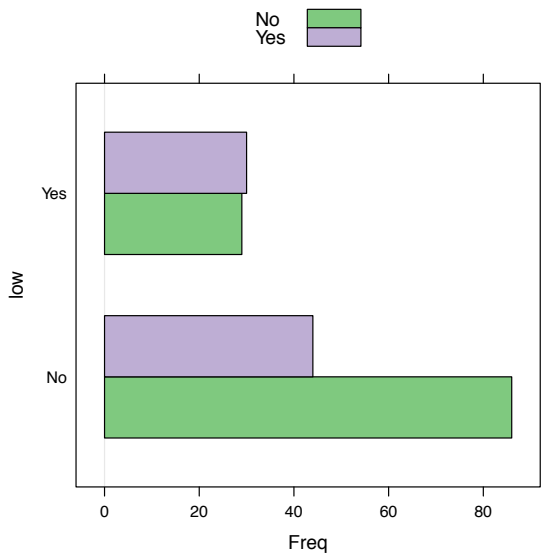


Diagramme en barres pour un tableau de contingence.

L'option `stack=FALSE` permet d'afficher des barres séparées pour les modalités de la variable `low`, rappelées en légende à l'aide de `auto.key=TRUE`. L'orientation des barres se contrôle avec l'option `horizontal=` (par défaut, elle vaut `TRUE`).

3.2.1 Échantillons indépendants

Dans le cas de deux échantillons indépendants, le test de Student permettant de tester l'égalité des moyennes de groupe s'obtient avec la commande `t.test`. Si l'on souhaite comparer le poids moyen des mères ayant présenté un problème d'irritabilité de l'utérus (variable `ui`) et celui des mères n'ayant aucun problème, on procèdera de la manière suivante :

```
> t.test(lwt ~ ui, data=birthwt, var.equal=TRUE)


Two Sample t-test

data:  lwt by ui
t = 2.11, df = 187, p-value = 0.03586
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  0.875 25.354
sample estimates:
 mean in group No mean in group Yes
              132               119
```

La notation par formule indique à R que l'on s'intéresse à la relation entre les deux variables `lwt` et `ui`, où les variations de `lwt` sont « décrites » par `ui`. En d'autres termes, `lwt` est considérée comme la variable réponse que l'on cherche à expliquer par la variable de classification `ui` (de type `factor`). L'option `var.equal=TRUE` (par défaut, elle vaut `FALSE`) indique à R de ne pas appliquer une correction de type Satterwaithe (test dit de Welch) lors du calcul.

R affiche le résultat du test de l'hypothèse nulle en donnant la valeur de la statistique de test (`t`), le nombre de degrés de liberté (`df`) et le degré de signification associé (`p-value`). Il fournit également un intervalle de confiance pour le paramètre, ici la différence de moyennes théoriques.

On peut naturellement vérifier si les conditions d'application de ce test, en particulier l'égalité des variances parentes, sont vérifiées en comparant les valeurs des variances estimées à partir de l'échantillon :

```
> with(birthwt, tapply(lwt, ui, var))

 No Yes
941 784
```

Une alternative consiste à vérifier visuellement la distribution de `lwt` dans les deux groupes définis par la variable `ui`, par exemple en utilisant un diagramme de type boîte à moustaches. Un test plus formel pour la comparaison des deux variances est disponible via la commande `var.test`.

On peut également comparer les résultats précédents avec ce que l'on obtiendrait en corrigeant les degrés de liberté de la distribution t de référence pour tenir compte de l'hétérogénéité des variances (les degrés de liberté ne sont alors généralement plus des entiers) :

```
> t.test(lwt ~ ui, data=birthwt)


Welch Two Sample t-test

data:  lwt by ui
t = 2.25, df = 39.2, p-value = 0.02982
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  1.35 24.88
sample estimates:
 mean in group No mean in group Yes
              132               119
```

Les tests réalisés ci-dessus considèrent une hypothèse alternative non orientée (différence des deux moyennes) et sont donc bilatéraux. Pour spécifier a priori le sens de la différence attendue sous l'hypothèse alternative (p. ex., $H_1 : \mu_1 > \mu_2$), il faut renseigner l'option `alternative=` dans la commande `t.test`.

3.2.2 Échantillons non indépendants

Dans le cas où les échantillons ne sont pas indépendants, la commande R à utiliser reste `t.test` mais il faut ajouter l'option `paired=TRUE` pour indiquer à R que les deux échantillons ne sont pas indépendants (cas où les séries de mesure ont été recueillies sur les mêmes unités statistiques à deux reprises, par exemple).

On profitera de cette section pour discuter l'organisation possible des données dans un tableau lorsque l'on travaille avec deux séries de mesures appariées. Considérons deux séries de scores sur une échelle de dépression (échelle de Hamilton) obtenues sur 9 patients avant (`x1`) et après traitement (`x2`) par tranquilisants :

```
> x1 <- c(1.83,  0.50,  1.62,  2.48, 1.68, 1.88, 1.55, 3.06, 1.30)
> x2 <- c(0.878, 0.647, 0.598, 2.05, 1.06, 1.29, 1.06, 3.14, 1.29)
```

Une façon naturelle d'organiser ces données consisterait à arranger ces deux séries de mesure dans un tableau à 9 lignes et 2 colonnes, les colonnes représentant les deux variables ci-dessus.

```
> d <- data.frame(x1, x2)
> dim(d)

[1] 9 2

> summary(d)

       x1              x2
 Min.   :0.50   Min.   :0.598
 1st Qu.:1.55   1st Qu.:0.878
 Median :1.68   Median :1.060
 Mean   :1.77   Mean   :1.335
 3rd Qu.:1.88   3rd Qu.:1.290
 Max.   :3.06   Max.   :3.140
```

Un test t pour comparer les moyennes entre les deux périodes se formulerait ainsi :

```
> with(d, t.test(x1, x2, paired=TRUE))


Paired t-test

data:  x1 and x2
t = 3.04, df = 8, p-value = 0.01618
alternative hypothesis: true difference in means is not equal to 0
```

```
95 percent confidence interval:
 0.104 0.760
sample estimates:
mean of the differences
                 0.432
```

On pourrait tout aussi bien adopter une autre organisation des données : dans une colonne on a l'ensemble des scores de dépression, et dans l'autre la période de mesure (avant ou après traitement). Pour transformer le tableau `d` (2 colonnes, `x1` et `x2`) en un tableau contenant ces deux nouvelles variables, on peut utiliser la commande `melt` du package `reshape2`. Ce package n'étant pas fourni avec l'installation de base de R, il sera nécessaire de l'installer en tapant `install.packages("reshape2")` dans R. On utilise ensuite la commande `library` pour « charger » le package dans la session R et pouvoir accéder aux commandes qu'il fournit.

```
> library(reshape2)
> dm <- melt(d)
> dim(dm)
```

```
[1] 18  2
```

```
> summary(dm)
```

```
 variable     value
 x1:9     Min.   :0.50
 x2:9     1st Qu.:1.06
          Median :1.43
          Mean   :1.55
          3rd Qu.:1.87
          Max.   :3.14
```

On peut vérifier que l'on retrouve bien les mêmes moyennes dans chaque groupe qu'avec la commande `summary(d)` utilisée plus haut.

```
> with(dm, tapply(value, variable, mean))
```

```
  x1   x2
1.77 1.33
```

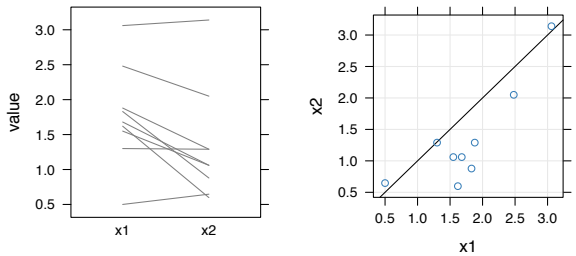
Quant au test *t*, il s'écrirait à présent :

```
> t.test(value ~ variable, data=dm, paired=TRUE)
```

On pourrait également rendre plus explicite le fait que ce sont les mêmes sujets qui sont mesurés dans les deux conditions expérimentales (`x1` puis `x2`) en ajoutant une variable codant pour l'identifiant du patient (n°).

```
> dm$id <- factor(rep(1:9, 2))
> head(dm, n=3)
```

```
  variable value id
1       x1  1.83  1
2       x1  0.50  2
3       x1  1.62  3
```



(Gauche) Profil d'évolution des scores individuels. (Droite) Diagramme de dispersion des scores : les points situés en dessous de la diagonale, représentée par la droite de pente unité, suggèrent qu'en général les patients ont un score plus bas après traitement (`x2`).

3.3 Comparaisons de proportions

3.3.1 Cas de deux proportions

Supposons que l'on s'intéresse à la probabilité d'observer des enfants en sous-poids à la naissance (`low`) selon que leur mère fumait ou non durant leur grossesse (`smoke`). L'hypothèse nulle est que ces deux probabilités sont égales. La commande `prop.test` permet de tester cette hypothèse en utilisant comme distribution asymptotique la loi normale.

```
> tab <- with(birthwt, table(smoke, low))
> tab
```

```
     low
smoke No Yes
  No  86  29
  Yes 44  30
```

```
> prop.table(tab, margin=1)
```

```
     low
smoke    No   Yes
  No  0.748 0.252
  Yes 0.595 0.405
```

```
> prop.test(tab[,c(2,1)], correct=FALSE)
```

```

2-sample test for equality of proportions without continuity correction

data:  tab[, c(2, 1)]
X-squared = 4.92, df = 1, p-value = 0.02649
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.2904 -0.0161
sample estimates:
prop 1 prop 2
 0.252  0.405
```

On prendra garde au fait que le tableau doit être légèrement réorganisé pour que l'événement d'intérêt (`low='Yes'`) figure bien dans la première colonne, d'où la permutation des deux colonnes du tableau avec `tab[,c(2,1)]` et les proportions recherchées 30/74 (fumeur) et 29/115 (non fumeur). Cela ne change rien au résultat du test d'hypothèse, mais cela évidemment les proportions estimées et l'intervalle de confiance associé à la différence entre ces deux proportions.

Une formulation alternative consiste à indiquer le nombre de « succès » (ici, le fait que le bébé soit en sous-poids) et l'effectif total, pour chaque classe définie par la variable `smoke`.

```
> prop.test(c(29,30), c(86+29, 44+30), correct=FALSE)
```

```

2-sample test for equality of proportions without continuity correction

data:  c(29, 30) out of c(86 + 29, 44 + 30)
X-squared = 4.92, df = 1, p-value = 0.02649
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.2904 -0.0161
sample estimates:
prop 1 prop 2
 0.252  0.405
```

On remarquera que la commande `prop.test` inclut une option `correct=` qui permet de décider si une correction de continuité (correction de Yates) doit être appliquée ou non.

3.3.2 Test du chi-deux

Un autre test fréquemment utilisé lorsque l'on travaille avec des tableaux de contingence est le test du χ^2 de Pearson qui vise à tester l'association entre deux variables qualitatives, l'écart à l'indépendance étant quantifié par la différence entre les effectifs théoriques attendus sous l'hypothèse d'indépendance et les effectifs observés à partir de l'échantillon.

Voici une application en considérant les mêmes variables `low` et `smoke`.

```
> chisq.test(xtabs(~ low + smoke, data=birthwt))


Pearson's Chi-squared test with Yates' continuity correction

data:  xtabs(~low + smoke, data = birthwt)
X-squared = 4.24, df = 1, p-value = 0.03958
```

Notons que l'on peut très bien remplacer l'instruction ci-dessus par `with(birthwt, chisq.test(table(low, smoke)))` ou `summary(xtabs(~ low + smoke, data=birthwt))`. Comme dans le cas de `prop.test`, l'option `correct=` peut être utilisée dans le cas d'un tableau 2×2 pour réaliser une correction de continuité.

En stockant le résultat précédent dans une variable auxiliaire, il est même possible de vérifier si les conditions d'application de ce test sont vérifiées, c'est-à-dire des effectifs théoriques suffisamment larges (généralement > 5) :

```
> res <- chisq.test(xtabs(~ low + smoke, data=birthwt))
> res$expected
     smoke
low     No  Yes
  No  79.1 50.9
  Yes 35.9 23.1
```

3.3.3 Cas de deux échantillons non indépendants

Dans le cas où le tableau de contingence est formé à partir du croisement de deux variables observées sur le même échantillon (par exemple, sévérité de la symptomatologie à deux classes en pré- et post-traitement), on peut recourir au test de McNemar à l'aide de la commande `mcnemar.test`. Cette commande accepte également les données présentées sous la forme d'un tableau généré avec `table` ou `xtabs`. L'aide en ligne fournit un exemple d'utilisation de cette commande, voir `help(mcnemar.test)`.

3.4 Mesures de risque et odds-ratio

Les mesures de risque usuellement calculées à partir d'un tableau de contingence croisant un facteur d'exposition (ou n'importe quelle variable binaire d'intérêt) et une variable réponse binaire (malade/pas malade, échec/réussite) sont le risque relatif et l'odds-ratio (rapport de cotes). Ce dernier peut être utilisé plus généralement pour quantifier le degré d'association entre deux variables binaires.

Les données sont généralement arrangées dans un tableau de contingence 2×2 où les effectifs associés aux « événements positifs » figurent sur la première ligne et la première colonne. Les variables `smoke` et `low` utilisées dans les illustrations précédentes ont deux niveaux, 0 et 1, et par défaut ce sont les niveaux de référence 0 (codant pour les événements négatifs, non-fumeur ou poids normal) qui apparaîtront sur la première ligne et la première colonne lorsque l'on utilise `table` ou `xtabs`. En pratique cela ne pose pas de problème pour le calcul de l'odds-ratio, mais pour le risque relatif il faut faire attention. On ré-arrangera donc le tableau de contingence en permutant les deux lignes et les deux colonnes.

```
> tab <- xtabs(~ smoke + low, data=birthwt)
> tab <- tab[c(2,1),c(2,1)]
> tab
```

```
     low
smoke Yes No
  Yes  30 44
  No   29 86
```

On voit donc que la première ligne du tableau `tab` contient la répartition des enfants ayant un poids inférieur ou supérieur à 2,5 kg pour les mères fumeuses, et que la première colonne contient bien la répartition des mères fumeuses ou non-fumeuses pour les enfants en sous-poids à la naissance. L'odds-ratio est simplement le rapport des odds (cotes) 30/44 et 29/86, soit $\frac{30/44}{29/86} = \frac{30\times86}{44\times29} = 2.02$. Quant au risque relatif, il représente le rapport des risques de chaque groupe d'exposition. Ici, on considère que le fait que la mère aît fumé durant sa grossesse représente le facteur d'exposition, d'où un risque de $30/(30+44)$ chez les fumeuses comparé à $29/(29+86)$ chez les non-fumeuses, soit un risque relatif de $\frac{30/(30+44)}{29/(29+86)} = 1.61$.

Il existe de nombreuses commandes permettant de retrouver ces calculs manuels et de fournir des intervalles de confiance pour les valeurs estimées de l'odds-ratio ou du risque relatif. Pour le calcul de l'odds-ratio, on utilisera un package externe, `vcd`, qu'il sera nécessaire d'installer au préalable. Pour cela, on tapera simplement `install.packages("vcd")` dans R, et après avoir vérifié que l'installation s'est correctement déroulée, on chargera le package à l'aide de la commande `library`, comme on l'a déjà vu pour le package `lattice` ou `reshape2`.

```
> library(vcd)
> oddsratio(tab, log=FALSE)
```

```
[1] 2.02
```

L'option `log=FALSE` est indispensable si l'on souhaite obtenir la valeur de l'odds-ratio et non le logarithme de sa valeur. La commande `confint` permet ensuite d'obtenir l'intervalle de confiance associé, par défaut à 95 % : `confint(oddsratio(tab, log=FALSE))`.

Une autre possibilité est d'utiliser le package `epiR`, également à installer et charger avant de pouvoir utiliser les commandes fournies dans ce package. L'avantage de ce package et de la commande `epi.2by2` est qu'il fournit une estimation du risque relatif et de l'odds-ratio, en considérant par défaut une étude de cohorte pour laquelle ces deux indicateurs ont un sens.

```
> library(epiR)
> epi.2by2(tab)
```

```
             Outcome +    Outcome -      Total        Inc risk *        Odds
Exposed +           30           44         74              40.5       0.682
Exposed -           29           86        115              25.2       0.337
Total               59          130        189              31.2       0.454

Point estimates and 95 % CIs:
-------------------------------------------------------
Inc risk ratio                                 1.61 (1.06, 2.44)
Odds ratio                                     2.01 (1.03, 3.96)
Attrib risk *                                  15.32 (1.61, 29.04)
Attrib risk in population *                    6.00 (-4.33, 16.33)
Attrib fraction in exposed (%)                 37.80 (5.47, 59.07)
Attrib fraction in population (%)              19.22 (-0.21, 34.88)
-------------------------------------------------------
 * Cases per 100 population units
```

3.5 Approches non-paramétriques et tests exacts

Pour la comparaison de mesures numériques recueillies sur deux échantillons indépendants, une approche non-paramétrique reposant uniquement sur les rangs des observations consiste à utiliser le test de Wilcoxon-Mann-Whitney, disponible dans la commande `wilcox.test`. Comme dans le cas du test de Student, lorsque

les échantillons ne sont pas indépendants, on ajoutera l'option `paired=TRUE`. Avec les deux exemples discutés précédemment (§ 3.2), on procèderait de la manière suivante :

```
> wilcox.test(lwt ~ ui, data=birthwt)


Wilcoxon rank sum test with continuity correction

data:  lwt by ui
W = 2896, p-value = 0.01626
alternative hypothesis: true location shift is not equal to 0
```

```
> with(d, wilcox.test(x1, x2, paired=TRUE))


Wilcoxon signed rank test

data:  x1 and x2
V = 40, p-value = 0.03906
alternative hypothesis: true location shift is not equal to 0
```

Le second test pour le cas des échantillons appariés est appelé test signé des rangs. Avec cet exemple utilisant les variables `x1` et `x2` dans le `data.frame d`, une formulation équivalente en utilisant le tableau faisant apparaître distinctement variable explicative et variable réponse (`dm`) serait alors :

```
> wilcox.test(value ~ variable, data=dm, paired=TRUE)
```

En ce qui concerne les tableaux de contingence, une alternative au test du χ^2 est le test de Fisher disponible via la commande `fisher.test`.

```
> fisher.test(xtabs(~ low + smoke, data=birthwt))


Fisher's Exact Test for Count Data

data:  xtabs(~low + smoke, data = birthwt)
p-value = 0.03618
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 1.03 3.96
sample estimates:
odds ratio
      2.01
```

On notera que cette commande renvoit également l'odds-ratio et son intervalle de confiance à 95 %.

En dernier lieu, R dispose d'une commande pour réaliser un test binomial, `binom.test`, qui permet de tester une probabilité de succès dans le cas d'une variable binaire en considérant une distribution binomiale et non une approximation normale comme dans le cas de `prop.test`.

Ce qu'il faut retenir

- Avant de réaliser un test, on calcule toujours quelques statistiques descriptives pour chaque niveau d'une variable explicative (`tapply`) ou en croisant les modalités de deux variables qualitatives (`table` ou `xtabs`).
- Les représentations graphiques pour deux variables ne diffèrent pas vraiment de celles vues dans le cas univarié, à ceci près que l'on utilise une notation par formule pour indiquer les relations entre variables. Dans le cas des variables qualitatives, on fournit généralement un tableau d'effectifs (`barchart` ou `dotplot`).
- Les procédures de test renvoit généralement le résultat du test de l'hypothèse nulle associée au test et un intervalle de confiance pour le paramètre estimé.
- Dans le cas des mesures de risque (odds-ratio, mais surtout risque relatif), on prendra garde à l'arrangement des effectifs (en lignes et en colonnes) dans un tableau de contingence.

Cours 4. Analyse de variance et plans d'expérience

Sommaire

Dans ce chapitre, on aborde l'analyse de la variance et quelques notions élémentaires sur les plans d'expérience. Seuls les cas de l'ANOVA à un et deux facteurs de classification (effets fixes, modèle avec interaction) sont abordés. Pour l'ANOVA à un facteur, le test de tendance linéaire est approché par deux techniques : la régression linéaire et la méthode des contrastes. Le cas des comparaisons multiples entre traitements est également abordé, en se limitant à la méthode de Bonferroni pour protéger les tests de l'inflation du risque de première espèce. Il est important de procéder dans tous les cas à une description approfondie des données avant de réaliser le modèle d'ANOVA (techniques de résumé numérique, méthodes graphiques, etc.).

Commandes essentielles

`sapply` : applique une commande aux colonnes d'un tableau
`aov` : construction d'un modèle d'ANOVA
`pairwise.t.test` : test t simultanés corrigés
`aggregate` : identique à `tapply`
`summary` (`summary.aov`) : tableau d'ANOVA
`bartlett.test` : test d'homogénéité des variances
`interaction` : construit les traitements associés au croisement de plusieurs facteurs

4.1 Représentation des données et statistiques descriptives

4.1.1 Format de représentation des données

Si l'on considère une variable numérique servant de variable réponse et une variable qualitative à k niveaux (encore appelés modalités, ou traitements) utilisée comme facteur de classification, on peut imaginer au moins deux manières de représenter les données sous R : un tableau où les k séries d'observations sont représentées dans différentes colonnes (soit un tableau à k colonnes et n lignes dans le cas d'un plan équilibré où on a le même nombre d'observations par traitement) ou bien un tableau où toutes les mesures figurent dans une première colonne et les traitements associés dans une seconde colonne, comme discuté en § 3.3.3 (p. 26).

On prendra toujours bien garde au fait que le facteur de classification doit bien être traité comme un `factor` et pas une variable numérique, au risque de se retrouver à faire de la régression linéaire sans le savoir.

4.1.2 Statistiques descriptives

Dans l'exemple ci-dessous, on génère une structure de données composée de 20 observations réparties de manière équilibrée sur les 4 niveaux (traitements) d'une variable qualitative. Les données sont disposées en 4 colonnes, chaque colonne correspondant à un traitement spécifique, et on a donc au final un tableau à 5 lignes et 4 colonnes. Les mesures sont générées artificiellement à partir de tirages indépendants (`replicate`) dans une loi normale de moyenne 12 et d'écart-type 1,2.

```
> d <- data.frame(replicate(4, rnorm(5, mean=12, sd=1.2)))
> d
```

```
    X1   X2   X3   X4
1 11.8 11.7 11.2 11.7
2 12.1 10.3 12.1 13.1
3 10.7 12.4 13.5 14.3
4 11.4 13.3 13.8 12.3
5 12.3 12.0 13.1 12.6
```

```
> summary(d)
```

```
       X1              X2              X3              X4
 Min.   :10.7   Min.   :10.3   Min.   :11.2   Min.   :11.7
 1st Qu.:11.4   1st Qu.:11.7   1st Qu.:12.1   1st Qu.:12.3
 Median :11.8   Median :12.0   Median :13.1   Median :12.6
 Mean   :11.7   Mean   :11.9   Mean   :12.7   Mean   :12.8
 3rd Qu.:12.1   3rd Qu.:12.4   3rd Qu.:13.5   3rd Qu.:13.1
 Max.   :12.3   Max.   :13.3   Max.   :13.8   Max.   :14.3
```

On souhaite calculer la moyenne des mesures pour chaque colonne. Dans le cas où le tableau de données est un `data.frame`, le plus simple pour appliquer la même commande à chaque colonne du tableau consiste à utiliser la commande `sapply` comme ceci :

```
> sapply(d, mean)
```

```
  X1   X2   X3   X4
11.6 11.9 12.7 12.8
```

Supposons à présent que ces mêmes données soient arrangées dans un tableau à deux colonnes, la première colonne indiquant le traitement et la seconde colonne la mesure recueillie. Par souci de simplicité, on ne fera pas apparaître le n° d'observation dans une autre colonne. Le package `reshape2` ou `reshape` (à charger au préalable) fournit la commande `melt` qui permet de transformer un tableau où les niveaux d'un facteur sont arrangés en colonnes en un tableau où ceux-ci sont arrangés dans la même colonne.

```
> dm <- melt(d, value.name="y", variable.name="trt")
> head(dm)
```

```
  trt    y
1  X1 11.8
2  X1 12.1
3  X1 10.7
4  X1 11.4
5  X1 12.3
6  X2 11.7
```

```
> summary(dm)
```

```
 trt          y
 X1:5   Min.   :10.3
 X2:5   1st Qu.:11.7
 X3:5   Median :12.2
 X4:5   Mean   :12.3
        3rd Qu.:13.1
        Max.   :14.3
```

```
> with(dm, tapply(y, trt, mean))
```

```
  X1   X2   X3   X4
11.6 11.9 12.7 12.8
```

On voit qu'avec ce type de structure, on se retrouve dans les situations évoquées aux chapitres 2 et 3 et que l'on peut utiliser sans problème la commande `tapply`. On cherchera en règle générale à se ramener à ce type d'organisation des données pour les modèles statistiques sous R. Une commande similaire à `tapply` mais qui permet d'utiliser une notation par formule est la commande `aggregate` dont l'usage est relativement simple : on indique le type de relation à laquelle on s'intéresse entre deux (ou plus) variables, le `data.frame` où trouver les variables et la commande à appliquer, ici `mean`.

```
> aggregate(y ~ trt, data=dm, mean)

  trt    y
1  X1 11.6
2  X2 11.9
3  X3 12.7
4  X4 12.8
```

Cette commande est strictement équivalente à `with(dm, tapply(y, trt, mean))`, seul le format du résultat renvoyé est différent : la commande `aggregate` renvoit ses résultats sous forme de `data.frame` ce qui parfois peut être intéressant pour des analyses secondaires ou des représentations graphiques.

4.2 ANOVA à un facteur

4.2.1 Le modèle d'ANOVA à un facteur

Prenons l'exemple des variations entre les poids des bébés (`bwt`) selon l'ethnicité de la mère (`race`).

Le modèle d'ANOVA est construit à l'aide de la commande `aov` sous R, en indiquant à l'aide d'une formule R la relation reliant la variable réponse (à gauche du symbole `~`) et la variable explicative (à droite du symbole `~`).

```
> aov(bwt ~ race, data=birthwt)

Call:
   aov(formula = bwt ~ race, data = birthwt)

Terms:
                    race Residuals
Sum of Squares   5015725  94953931
Deg. of Freedom        2       186

Residual standard error: 714
Estimated effects may be unbalanced
```

On voit que le résultat renvoyé par R ne ressemble pas vraiment à un tableau d'analyse de variance et ne contient que des sommes de carrés (avec leurs degrés de liberté) et la variance résiduelle (`sqrt(96179472/187)`). Pour afficher le tableau d'analyse de variance correspondant au modèle `bwt ~ race`, il faut utiliser la commande `summary`. Il s'agit de la « même commande » que celle utilisée pour résumer un tableau de données (`data.frame`) ou une variable mais elle se comporte différemment dans le cas où l'objet R en question est un modèle d'ANOVA construit avec la commande `aov` (plus précisément, la commande est en fait `summary.aov`).

```
> summary(aov(bwt ~ race, data=birthwt))

             Df   Sum Sq Mean Sq F value Pr(>F)
race          2  5015725 2507863    4.91 0.0083
Residuals   186 94953931  510505
```

Notons qu'il sera souvent plus commode de stocker le résultat du modèle ANOVA dans une variable pour le réutiliser ensuite avec d'autres commandes. Les deux étapes précédentes peuvent donc être reformulées ainsi :

```
> res <- aov(bwt ~ race, data=birthwt)
> summary(res)
```

Une représentation graphique naturelle pour ce type de données (une variable numérique décrite par une variable qualitative) est un diagramme de type boîte à moustaches, pour avoir une idée des fluctuations à l'intérieur des groupes ou traitements, et un diagramme en barres ou en points pour avoir une idée de la variation des moyennes par traitement.

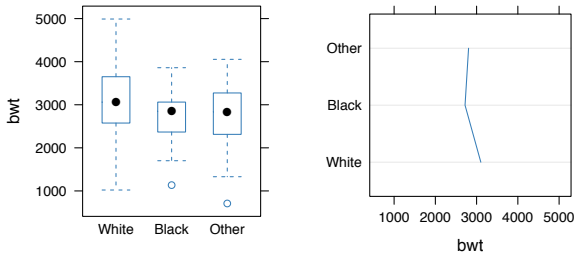
```
> bwplot(bwt ~ race, data=birthwt)
```

On remarquera que la notation par formule est identique à celle utilisée pour construire le modèle d'ANOVA. En ce qui concerne le diagramme en points, on doit introduire une petite variation :

```
> dotplot(race ~ bwt, data=birthwt, type="a")
```

L'option `type="a"` permet de calculer automatiquement les moyennes (*a*verage) de `bwt` par niveau de la variable `race`. Cette dernière figure ici à gauche du symbole ~ dans la formule R ce qui permet de la représenter sur l'axe des ordonnées plutôt que l'axe des abscisses. Une manière équivalente d'arriver à ce résultat consiste à utiliser directement un tableau des moyennes de groupe, à l'aide de la commande `aggregate` présentée dans la section précédente. Voici une illustration en remplaçant le diagramme en points (`dotplot`) par un diagramme en barres (`barchart`) :

```
> bwtmeans <- aggregate(bwt ~ race, data=birthwt, mean)
> bwtmeans
> barchart(bwt ~ race, data=bwtmeans, horizontal=FALSE)
```



(Gauche) Distribution des poids des bébés selon l'ethnicité de la mère à l'aide de boîtes à moustaches. Les points à l'intérieur des boîtes représentent les médianes dans chaque groupe. (Droite). Poids moyens des bébés en fonction de l'ethnicité. Les moyennes de groupe sont reliées par des segments afin de faciliter la lecture.

Pour vérifier les conditions d'application de l'ANOVA (normalité des résidus et homogénéité des variances), on peut utiliser des représentations graphiques ou des tests formels. Le test de Shapiro-Wilk (normalité d'une distribution, cas des grands échantillons) est accessible via la commande `shapiro.test`, mais on lui préfèrera généralement les méthodes graphiques (QQ plot ou boîtes à moustaches). Pour l'égalité des variances, on dispose des tests de Levene (commande `leveneTest` dans le package externe `car`) ou de Bartlett (`bartlett.test`). Ces commandes utilisent la même notation par formule :

```
> bartlett.test(bwt ~ race, data=birthwt)


Bartlett test of homogeneity of variances

data:  bwt by race
Bartlett's K-squared = 0.659, df = 2, p-value = 0.7191
```

Les commandes `fitted` et `resid` permettent d'obtenir les valeurs prédites par le modèle et les résidus (écarts entre valeurs observées et valeurs prédites), respectivement. On vérifiera sans mal que dans ce cas les valeurs prédites correspondent bien aux moyennes des traitements.

```
> unique(predict(res))
[1] 2720 2805 3103
```

4.2.2 Comparaisons par paires de traitement

Un résultat significatif à l'ANOVA (F de Fisher-Snedecor) permet de conclure qu'au moins une paire de moyennes peut être considérée comme statistiquement différente au seuil fixé (5 % par défaut). Plusieurs techniques dites de « comparaisons multiples » ont été proposées dans la littérature. On se contentera de présentera la méthode de Bonferroni pour la comparaison de l'ensemble des paires de moyennes à l'aide de tests t.

Une approche assez conservatrice est d'appliquer une correction de type Bonferroni pour chacun des tests

réalisés : cela revient à multiplier chaque degré de signification par le nombre total de tests réalisés. La commande `pairwise.t.test` permet de tester toutes les paires de moyennes à l'aide de tests de Student, et elle renvoit les degrés de signification corrigés pour chaque paire de traitement.

```
> with(birthwt, pairwise.t.test(bwt, race, p.adjust.method="bonferroni"))


Pairwise comparisons using t tests with pooled SD

data:  bwt and race

      White Black
Black 0.049 -
Other 0.029 1.000

P value adjustment method: bonferroni
```

On peut vérifier que l'on obtient bien les mêmes résultats à l'aide de simples tests de Student, en utilisant `t.test` et en excluant systématiquement l'un des niveaux de la variable `race`. Voici un exemple avec les comparaisons `White/Black` et `White/Other` :

```
> t1 <- t.test(bwt ~ race, data=birthwt, subset=race != "Other")
> t2 <- t.test(bwt ~ race, data=birthwt, subset=race != "Black")
> c(t1$p.value, t2$p.value) * 3
[1] 0.0351 0.0328
> with(birthwt, pairwise.t.test(bwt, race, p.adj="bonf", pool.sd=FALSE))


Pairwise comparisons using t tests with non-pooled SD

data:  bwt and race

      White Black
Black 0.035 -
Other 0.033 1.000

P value adjustment method: bonferroni
```

Pour que la comparaison entre les deux approches soit valide, il est nécessaire de demander à R de ne pas utiliser une estimation de la variance commune à l'ensemble des traitements (comme dans l'ANOVA) mais bien celle des seuls traitements pris en compte dans la comparaison, d'où l'option `pool.sd=FALSE`. Enfin, on retiendra que, lorsque les options ne prêtent à aucune confusion, elles peuvent être abbrégées ; dans le cas ci-dessus, on peut très bien écrire `p.adj="bonf"` au lieu de `p.adjust.method="bonferroni"`.

4.2.3 Test de tendance linéaire

Dans les cas où cela fait sens de considérer que les niveaux du facteur de classification sont ordonnés, il peut être intéressant de formuler l'ANOVA sous la forme d'un test de tendance linéaire. On supposera généralement que les niveaux du facteur sont équi-espacés.

Considérons par exemple la variable `ftv`, qui représente le nombre de visites chez le gynécologue durant le premier trimestre de grossesse. Cette variable prend des valeurs entre 0 et 6, les valeurs supérieures à 2 étant assez rarement observées (12 cas au total, `which(birthwt$ftv > 3)`).

```
> table(birthwt$ftv)


  0   1   2   3   4   6
100  47  30   7   4   1
```

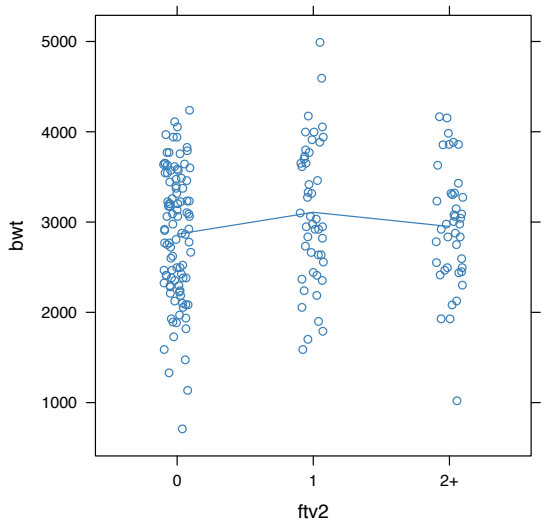
On peut décider de recoder cette variable de comptage en variable qualitative à 3 niveaux, `ftv2`, après avoir regroupé les 3 dernières valeurs (3, 4 et 6) avec la valeur 2 dans une modalité appelée 2+.

```
> birthwt$ftv2 <- birthwt$ftv
> birthwt$ftv2[birthwt$ftv2 > 2] <- 2
> birthwt$ftv2 <- factor(birthwt$ftv2, labels=c("0","1","2+"))
> table(birthwt$ftv2)


  0   1  2+
100  47  42
```

On peut représenter ces données sous forme d'un diagramme de dispersion décrivant les variations de `bwt` selon les niveaux de `ftv2`.

```
> xyplot(bwt ~ ftv2, data=birthwt, type=c("p","a"), jitter.x=TRUE)
```



Distribution du poids des bébés en fonction de la fréquence des visites gynécologiques durant le 1er trimestre de grossesse.

L'option `type=c('p','a')` permet de combiner la représentation des observations individuelles sous forme de points avec les moyennes pour chaque niveau du facteur `ftv2`. L'option `jitter.x=TRUE` ajoute un léger décalage horizontal des points pour faciliter la lecture en cas de chevauchement des points (cela n'affecte en rien les valeurs lues sur l'axe des ordonnées puisque le décalage est horizontal).

Une ANOVA classique donnerait les résultats suivants :

```
> summary(aov(bwt ~ ftv2, data=birthwt))
             Df   Sum Sq Mean Sq F value Pr(>F)
ftv2          2  1887925  943963    1.79   0.17
Residuals   186 98081730  527321
```

Pour tester un effet de tendance linéaire entre ces deux variables, on peut utiliser deux approches : l'approche basée sur la régression linéaire (la variable `ftv2` est alors considérée comme une variable numérique et non un facteur), qui sera vue plus en détails au chapitre suivant, ou celle reposant sur l'usage de contrastes associés aux niveaux de `ftv2`. Dans l'approche par régression linéaire, on se contente de considérer que la variable explicative est en fait une variable numérique et cela revient à effectuer une régression en considérant les variables `bwt` et `ftv2` sans convertir cette dernière en facteur comme on l'a fait plus haut. On peut tout de même indiquer à R de traiter la variable qualitative `ftv2` comme une variable numérique en utilisant la commande `as.numeric`. Le résultat de la régression linéaire, réalisée à l'aide de la commande `lm`, est résumé dans un tableau d'analyse de variance ci-dessous. Il indique que les données ne sont pas compatibles avec l'existence d'une relation linéaire entre les variables `bwt` et `ftv2`.

```
> anova(lm(bwt ~ as.numeric(ftv2), birthwt))

Analysis of Variance Table

Response: bwt
                  Df   Sum Sq Mean Sq F value Pr(>F)
as.numeric(ftv2)   1   542578  542578    1.02   0.31
```

```
Residuals        187 99427078  531696
```

Notons que l'on peut aisément vérifier comment R a transformé les niveaux de `ftv2` en valeurs numériques :

```
> levels(birthwt$ftv2)
[1] "0"  "1"  "2+"
> sort(unique(as.numeric(birthwt$ftv2)))
[1] 1 2 3
```

Il y a bien un décalage d'une unité, mais cela ne change foncièrement rien à la pente de la droite de régression.

Concernant la méthode des contrastes, R fournira automatiquement ce type de test de tendance lorsque la variable qualitative est spécifiée comme un `factor` à modalités ou niveaux ordonnés à l'aide de la commande `ordered` ou `as.ordered`. Dans le cas d'une variable à k modalités, R construira automatiquement $k-1$ contrastes orthogonaux permettant de tester l'existence d'une relation linéaire, quadratique, cubique, etc. Pour notre exemple, la variable `ftv2` possède 3 niveaux et R fournira deux contrastes : le premier pour la tendance linéaire et le second pour la tendance quadratique (qui revient à considérer les valeurs au carré de `ftv2` dans un modèle de régression).

```
> birthwt$ftv3 <- ordered(birthwt$ftv2)
> res <- aov(bwt ~ ftv3, data=birthwt)
> summary(res, split=list(ftv3=c(Linear=1, Quadratic=2)))
                  Df   Sum Sq Mean Sq F value Pr(>F)
ftv3               2  1887925  943963    1.79   0.17
  ftv3: Linear     1   542578  542578    1.03   0.31
  ftv3: Quadratic  1  1345348 1345348    2.55   0.11
Residuals        186 98081730  527321
```

Remarque : Le même type de résultat peut être obtenu en utilisant la commande `summary.lm` (cette fois, on est obligés de spécifier explicitement le suffixe `.lm`) pour obtenir un résumé des coefficients du modèle linéaire.

```
> summary.lm(res)

Call:
aov(formula = bwt ~ ftv3, data = birthwt)

Residuals:
    Min      1Q  Median      3Q     Max
-2156.1  -484.9    26.1   578.9  1882.0

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2974.7       56.8   52.36   <2e-16
ftv3.L          60.6       94.4    0.64     0.52
ftv3.Q        -163.3      102.2   -1.60     0.11

Residual standard error: 726 on 186 degrees of freedom
Multiple R-squared:  0.0189,Adjusted R-squared:  0.00834
F-statistic: 1.79 on 2 and 186 DF,  p-value: 0.17
```

4.3 Approche non-paramétrique de l'ANOVA à un facteur

L'alternative non-paramétrique à l'ANOVA à un facteur est l'ANOVA de Kruskal-Wallis qui consiste en une ANOVA réalisée sur les rangs des observations plutôt que les valeurs d'origine. La commande R permettant de réaliser ce type d'analyse est `kruskal.test`, et celle-ci s'utilise de la même manière que `aov` à l'aide d'une formule dans laquelle on indique la variable réponse et la variable explicative. Contrairement à `aov`, il n'est pas nécessaire de combiner la commande `kruskal.test` avec `summary`.

Voici un exemple d'utilisation avec les données de poids à la naissance selon l'éthnicité de la mère.

```
> kruskal.test(bwt ~ race, data=birthwt)


Kruskal-Wallis rank sum test

data:  bwt by race
Kruskal-Wallis chi-squared = 8.52, df = 2, p-value = 0.01412
```

Des comparaisons par paires de traitement peuvent être réalisées à l'aide de tests de Wilcoxon, selon le même principe que celui présenté en § 4.2.2.

4.4 ANOVA à deux facteurs

En présence de deux facteurs, la formulation du modèle d'ANOVA reste sensiblement équivalente à celle présentée plus haut. On exprime la relation entre les variables à l'aide d'une formule R, les facteurs de classification se trouvant à droite du symbole `~`. Par exemple, `y ~ x1 + x2` représente un modèle à deux facteurs `x1` et `x2`. Pour représenter le terme d'interaction, on utilise le symbole `:` pour lier les deux variables en interaction. Avec les notations précédentes, on utiliserait `y ~ x1 + x2 + x1:x2` pour représenter le modèle d'ANOVA complet (deux effets principaux et un effet d'interaction). Cette dernière notation peut se simplifier sous la forme `y ~ x1 * x2` (on remplace le `+` par un `*`). Dans ce contexte, on appelera traitement le croisement de deux modalités des facteurs.

4.4.1 Construction du tableau d'ANOVA

Si l'on s'intéresse à l'effet de l'ethnicité et des antécédents d'hypertension chez la mère sur le poids des nouveaux-nés, on peut formuler le modèle d'ANOVA, incluant l'interaction entre ces deux facteurs, de la manière suivante :

```
> m <- aov(bwt ~ race + ht + race:ht, data=birthwt)
```

Comme on l'a dit, l'instruction `aov(bwt ~ race * ht, data=birthwt)` serait strictement équivalente. Les résultats de ce modèle sont indiqués ci-dessous.

```
> summary(m)

             Df   Sum Sq Mean Sq F value Pr(>F)
race          2  5015725 2507863    4.97 0.0079
ht            1  1776713 1776713    3.52 0.0621
race:ht       2   889649  444825    0.88 0.4157
Residuals   183 92287568  504304
```

Le tableau d'ANOVA indique pour chaque source de variabilité la somme de carrés associée, les degrés de liberté, le carré moyen et le test F. On voit que dans ce cas, seul le facteur `race` est significatif en considérant un risque d'erreur de 5 %. L'interaction n'est pas significative, ce qui indique que l'effet de ce facteur ne dépend pas du niveau de `ht`.

Le modèle sans interaction s'obtient sans difficulté en omettant le terme d'interaction, soit

```
> summary(aov(bwt ~ race + ht, data=birthwt))
```

```
             Df   Sum Sq Mean Sq F value Pr(>F)
race          2  5015725 2507863    4.98 0.0078
ht            1  1776713 1776713    3.53 0.0619
Residuals   185 93177217  503661
```

Outre un résumé numérique des moyennes des traitements, un diagramme d'interaction permet souvent de mieux se représenter les résultats. On peut le construire à l'aide des commandes suivantes :

```
> xyplot(bwt ~ race, data=birthwt, groups=ht, type=c("a","g"),
+        auto.key=list(corner=c(0,1), title="ht", cex.title=1, points=FALSE, lines=TRUE))
```

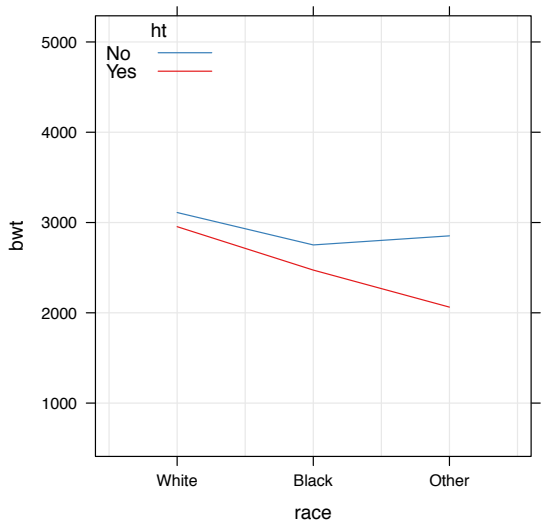


Diagramme d'interaction pour le modèle d'ANOVA à deux facteurs.

Les options de légende utilisées permettent de préciser la variable utilisée pour le groupement et la position de la légende (coin supérieur droit), dans laquelle on affiche des segments de ligne (`lines=TRUE`) plutôt que des points. L'option `type=c('a','g')` permet de n'afficher que les moyennes des traitements (croisement des niveaux des deux facteurs), ainsi qu'un quadrillage du graphique ce qui facilite le repérage des valeurs affichées.

Pour obtenir les moyennes des traitements, on peut utiliser soit la commande `tapply`, soit la commande `aggregate`. Avec cette dernière, on écrirait ainsi :

```
> aggregate(bwt ~ race + ht, data=birthwt, mean)
```

```
   race  ht  bwt
1 White  No 3111
2 Black  No 2752
3 Other  No 2852
4 White Yes 2954
5 Black Yes 2473
6 Other Yes 2063
```

Notons également que R dispose de la commande `model.tables` qui calcule automatiquement les effets moyens (déviations entre les moyennes conditionnelles et la moyenne générale) ainsi que les erreurs-types associées.

```
> model.tables(m)
```

```
Tables of effects

 race
    White  Black  Other
    158.1 -224.9 -139.3
rep  96.0   26.0   67.0

 ht
        No  Yes
     25.15 -371
rep 177.00   12
```

```
 race:ht
       ht
race    No     Yes
  White  -12.6  229.6
  rep     91.0    5.0
  Black  -13.9  106.7
  rep     23.0    3.0
  Other   23.3 -367.0
  rep     63.0    4.0
```

4.4.2 Diagnostic du modèle

Les hypothèses du modèle d'ANOVA à deux facteurs restent identiques à celles présentées pour le modèle à un facteur (indépendance des observations, normalité et égalité des variances), à ceci près qu'elles portent sur les résidus du modèle, et non les valeurs brutes observées. Pour tester l'hypothèse d'homoscédasticité (égalité des variances), par exemple, on prendra donc garde au fait que le test de Bartlett doit porter sur les 6 traitements formés à partir du terme d'interaction, et non sur les niveaux de chaque facteur considérés en isolation. Avec le modèle précédent, `bwt ~ race * ht`, on formulerait le test de la manière suivante :

```
> htrace <- with(birthwt, interaction(race, ht))
> bartlett.test(bwt ~ htrace, data=birthwt)


Bartlett test of homogeneity of variances

data:  bwt by htrace
Bartlett's K-squared = 5.12, df = 5, p-value = 0.4011
```

Le même principe peut être utilisé pour visualiser la distribution des réponses, des valeurs prédites ou des résidus en fonction des traitements.

Ce qu'il faut retenir

— Plutôt que de considérer des séries d'observations arrangées en colonne (cas d'un plan équilibré), on préférera travailler avec un tableau de données faisant explicitement apparaître une variable réponse et un ou plusieurs facteurs de classification.
— La commande `aggregate` peut avantageusement remplacer la commande `tapply` grâce à l'usage de formules R.
— La commande `aov` permet de construire un modèle d'ANOVA, le tableau de décomposition de la variance étant directement fourni par la commande `summary`.

Cours 5. Corrélation et régression linéaire

Sommaire

Dans ce chapitre, on s'intéressera aux mesures d'association linéaire ou monotone entre deux variables numériques (estimation ponctuelle, par intervalle, et test d'hypothèse) et aux représentations graphiques exploratoires associées (diagramme de dispersion et courbe loess). Le modèle de régression linéaire est ensuite développé dans le cas d'une variable explicative (régression linéaire simple), en particulier l'estimation des coefficients de la droite de régression et la construction des tests de significativité associés, le tableau d'analyse de variance associé au modèle de régression, la prédiction ponctuelle et par intervalle, ainsi que la vérification de ses conditions d'application et le diagnostic du modèle par l'analyse des résidus.

Commandes essentielles

`cor` : calcul du coefficient de corrélation
`lm` : construction d'un modèle de régression
`anova` : tableau d'analyse de variance pour la régression
`resid` : résidus du modèle
`cor.test` : test de significativité pour une corrélation
`summary` (`summary.lm`) : tableau des coefficients
`fitted` : valeurs ajustées
`predict` : prédiction ponctuelle et par intervalle

5.1 Corrélation

5.1.1 Statistiques descriptives

Les résumés descriptifs pour deux variables numériques s'obtiennent aisément avec `summary` ou une commande opérant par variable (`sapply` ou `apply`), dans le cas où les variables sont arrangées en colonnes. Si l'on s'intéresse aux mesures de poids des mères (`lwt`) et des nouveaux-nés (`bwt`), les commandes suivantes fourniront essentiellement les mêmes informations :

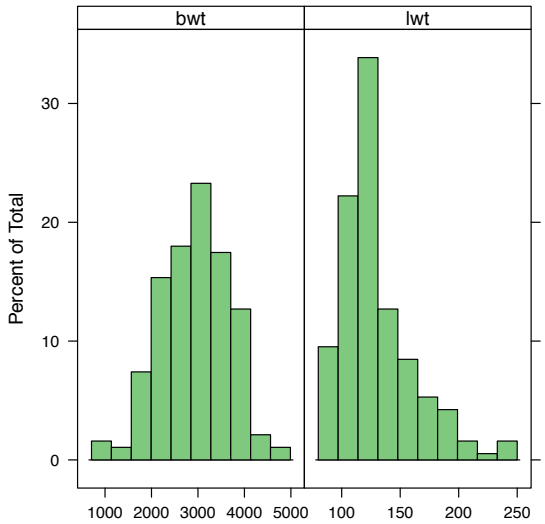
```
> summary(birthwt[,c("bwt","lwt")])
> summary(birthwt[,c(10,3)])
> summary(subset(birthwt, select=c(bwt, lwt)))
> sapply(birthwt[,c("bwt","lwt")], summary)
> apply(birthwt[,c(10,3)], 2, summary)
```

Quelques remarques : il est toujours plus facile de relire les scripts lorsque les variables sont explicitement nommées (`c("bwt","lwt")`) plutôt qu'adressées par leur position dans le tableau (`c(10,3)`) ; la commande `apply` fonctionne avec les objets R de type `matrix` ; le nom des variables dans la commande `subset` ne doit pas être entouré de quotes (").

Il est possible de visualiser directement les histogrammes de ces deux variables d'intérêt à l'aide de la commande `histogram`. La seule subtilité est que généralement les commandes du package `lattice` cherchent à unifier les unités de mesure pour les axes des ordonnées et des abscisses. Dans le cas présent, cela signifie que `histogram` construira un axe des abscisses représentant les plus grandes variations (poids des bébés exprimés en grammes, avec une étendue 700 à 5000) ce qui rendra illisible l'histogramme du poids des mères. Pour

remédier à ce problème, il est nécessaire d'imposer à R de constuire deux systèmes de coordonnées différents, comme cei :

```
> histogram(~ bwt + lwt, data=birthwt, scales=list(x=list(relation="free")),
+           breaks=NULL, xlab="")
```



Histogrammes des poids des mères (`lwt`, en kg) et des nouveaux-nés (`bwt`, en g).

En l'absence de variable de classification (facteur) permettant de séparer deux séries de mesures, il suffit d'indiquer la liste des variables dont on souhaite afficher la distribution sous forme d'histogramme séparées par un +.

On peut éventuellement rajouter `y=list(relation="free")` à l'option `scales=` pour que R utilise des métriques différentes pour les deux axes verticaux.

5.1.2 Diagramme de dispersion et courbe loess

Avant de calculer n'importe quelle statistique sur une série bivariée d'observations, il est utile de visualiser graphiquement la relation entre les deux variables, le plus souvent sous forme d'un diagramme de dispersion. La commande `xyplot` dispose d'un certain nombre d'options via l'option `type=` qui facilite l'identification d'une association plus ou moins linéaire entre les deux variables, en particulier l'option `type="smooth"`. Celle-ci permet de superposer sur le diagramme de dispersion une courbe de type « loess » qui est obtenue à partir d'une sorte de régression locale dont on peut contrôler le degré de lissage, voir `help(loess)`.

Convertissons dans un premier temps les poids des mères en kilogrammes,

```
> birthwt$lwt <- birthwt$lwt/2.2
```

puis affichons le diagramme de dispersion et la courbe loess :

```
> xyplot(bwt ~ lwt, data=birthwt, type=c("p","g","smooth"))
```

Le degré de lissage de la courbe loess peut se contrôler avec l'option `span=` ; par exemple, `span=1` fournira une courbe moins sensible aux variations locales et donc d'apparence beaucoup plus lisse.

5.1.3 Mesures d'association paramétrique et non-paramétrique

Le coefficient de corrélation de Bravais-Pearson est obtenu à partir de la commande `cor`, en indiquant les deux variables d'intérêt :

```
> with(birthwt, cor(bwt, lwt))
```

```
[1] 0.186
```

Lorsqu'il y a des données manquantes, il est nécessaire d'indiquer à R comment calculer le coefficient de corrélation dans ce cas. Une approche classique (dans le cas bivarié, ou pour construire des matrices de corrélation sur un plus grand nombre de variables) consiste à ne considérer que les paires d'observations

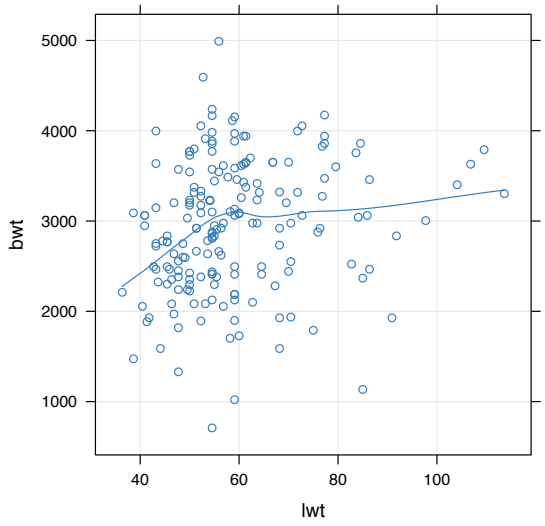


Diagramme de dispersion représentant les variations du poids des nouveaux-nés (`bwt`) en fonction du poids des mères (`lwt`).

Dans le cas des grands échantillons, en cas de superposition des points il est peu conseillé d'ajouter un décalage horizontal (ce qui risque de biaiser légèrement la forme du nuage de points) et on préférera jouer sur transparence des points (`alpha=0.5`) ou la taille des symboles (`cex=0.6`).

complètes. Si on a deux variables `x` et `y`, R excluera les unités statistiques pour lesquelles `is.na(x)` ou `is.na(y)` vaut `TRUE` (ceci inclut évidemment la conjonction des deux conditions). L'option à utiliser dans la commande `cor` est `use="pairwise"`.

Pour calculer le coefficient de corrélation de Spearman (basé sur les rangs des observations), on ajoutera l'option `method="spearman"`.

```
> with(birthwt, cor(bwt, lwt, method="spearman"))
```

```
[1] 0.249
```

Ces deux commandes fonctionnent également avec plus de deux variables : il suffit dans ce cas de fournir à `cor` un tableau avec les variables d'intérêt, par exemple `cor(birthwt[,c("age","lwt","bwt")])`.

5.1.4 Estimation par intervalles et test d'inférence

La commande `cor.test` fournit à la fois un test de l'hypothèse $H_0 : \rho = 0$ (nullité du paramètre, c'est-à-dire corrélation théorique égale à zéro) via un test de Student, ainsi qu'un intervalle de confiance calculé par transformation inverse de Fisher. Dans le cas non-paramétrique, R propose de calculer un degré de signication exact pour les échantillons de taille $n < 50$ (sans ex-æquo), autrement une approximation normale est utilisée.

```
> with(birthwt, cor.test(bwt, lwt))
```

```
Pearson's product-moment correlation

data:  bwt and lwt
t = 2.58, df = 187, p-value = 0.0105
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.0442 0.3200
sample estimates:
  cor
0.186
```

Notons que l'on peut utiliser une notation par formule avec cette commande, de manière similaire à celle utilisée pour `xtabs` lorsque les deux variables jouent un rôle symétrique. Il est également possible de restreindre les observations utilisées pour l'estimation du paramètre et le test de sa nullité à l'aide d'une option `subset=`. On utilisera, par exemple,

```
> cor.test(~ bwt + lwt, data=birthwt, subset=bwt > 2500)
```

pour restreindre les calculs aux seules observations pour lesquelles le poids des nouveaux-nés est supérieur à 2,5 kg.

On ajoutera l'option `method="spearman"` pour réaliser un test d'inférence équivalent portant sur le coefficient de corrélation basé sur les rangs des observations.

```
> cor.test(~ bwt + lwt, data=birthwt, method="spearman")


Spearman's rank correlation rho

data:  bwt and lwt
S = 845136, p-value = 0.0005535
alternative hypothesis: true rho is not equal to 0
sample estimates:
  rho
0.249
```

Pour tester si une corrélation diffère d'une valeur autre que zéro, il est nécessaire de recourir à la commande `r.test` du package `psych` (à installer et charger au préalable).

5.2 Régression linéaire simple

5.2.1 Droite de régression

Pour réaliser une régression linéaire, et plus généralement n'importe quel modèle linéaire (régression linéaire multiple, analyse de variance, analyse de covariance), R offre la commande `lm`. Son usage est globalement identique à celui de `aov` vu au chapitre précédent : on décrit à l'aide d'une formule R la relation entre les variables (variable réponse et variable explicative), le `data.frame` dans lequel se trouvent les variables, et éventuellement le sous-ensemble d'observations sur lequel on souhaite travailler avec l'option `subset=`. Cette dernière option permet par exemple de travailler sur un groupe d'observations défini par une variable catégorielle ou n'importe quel critère de filtrage des observations. Pour modéliser la relation entre le poids des nouveaux-nés (variable réponse) et le poids des mères (variable explicative ou prédicteur), on utilisera la formulation suivante :

```
> lm(bwt ~ lwt, data=birthwt)


Call:
lm(formula = bwt ~ lwt, data = birthwt)

Coefficients:
(Intercept)          lwt
    2369.62         9.74
```

R renvoit l'ordonnée à l'origine (« Intercept ») et la pente de la droite de régression, celle-ci reflétant les variations de `bwt` lorsque `lwt` varie d'une unité.

Comme on l'a vu dans le cas de l'ANOVA avec `aov` (§ 4.2.1), l'information retournée par la commande `lm` est minimale. Pour obtenir le tableau des coefficients de régression, leur erreur standard et leur degré de signification obtenu par des tests de Student, on utilisera encore une fois la commande `summary`.

```
> r <- lm(bwt ~ lwt, data=birthwt)
> summary(r)


Call:
lm(formula = bwt ~ lwt, data = birthwt)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-2192.1  -498.0    -3.8   508.3  2075.6

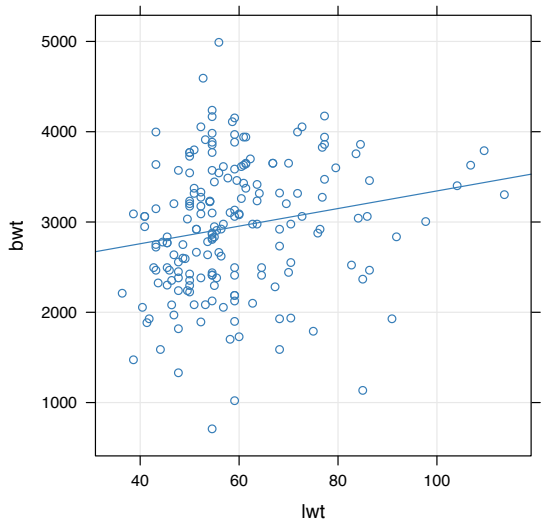
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2369.62     228.49   10.37   <2e-16
lwt             9.74       3.77    2.58    0.011

Residual standard error: 718 on 187 degrees of freedom
Multiple R-squared:  0.0345,Adjusted R-squared:  0.0293
F-statistic: 6.68 on 1 and 187 DF,  p-value: 0.0105
```

En plus des tests de significativité sur les coefficients de régression, R inclut des informations supplémentaires comme la résiduelle et le coefficient de détermination R^2 qui reflète la part de variance expliquée par la régression (voir § 5.2.2).

Il est possible de faire figurer la droite de régression sur le diagramme de dispersion, comme on l'a fait avec la courbe loess, à l'aide de l'option `type="r"` (au lieu de `type="smooth"`), bien que les deux options puissent être combinées.

```
> xyplot(bwt ~ lwt, data=birthwt, type=c("p","g","r"))
```



Relation entre poids des bébés et poids des mères et droite de régression associée.

Le type de symbole utilisé par défaut (cercle) convient dans le cas des observations en nombre élevé, mais on peut soit en régler la taille (`cex=0.8`, par exemple), soit utiliser des symboles transparents (`pch=19, alpha=0.5`).

5.2.2 Estimation par intervalles et tableau d'analyse de variance

La commande `summary` appliquée à un modèle de régression ne fournit pas d'intervalles de confiance pour les coefficients de régression. Pour cela, on utilisera la commande `confint`, éventuellement en précisant la largeur de l'intervalle de confiance via l'option `level=` (par défaut, elle vaut 0.95 pour un intervalle à 95 %).

```
> confint(r)
```

```
                2.5 % 97.5 %
(Intercept) 1918.87 2820.4
lwt            2.31   17.2
```

On pourrait associer ces intervalles aux coefficients estimés à l'étape précédente en construisant un tableau à 3 colonnes :

```
> res <- cbind(coef(r), confint(r))
> colnames(res)[1] <- "Coef"
```

```
> round(res, 3)
```

```
               Coef   2.5 % 97.5 %
(Intercept) 2369.62 1918.87 2820.4
lwt            9.74    2.31   17.2
```

On utilise ici le fait que la commande `coef` renvoit les coefficients de régression estimés dans le modèle que l'on a appelé `r`, tandis que `cbind` permet de concaténer des listes de nombres en colonnes (sous réserve que chaque liste possède le même nombre d'éléments).

Le tableau d'ANOVA associé au modèle de régression, qui décompose la variabilité totale en une part de variance expliquée par le modèle et la résiduelle, s'obtient à l'aide de la commande `anova`.

```
> anova(r)
```

```
Analysis of Variance Table

Response: bwt
           Df   Sum Sq Mean Sq F value Pr(>F)
lwt         1  3448639 3448639    6.68  0.011
Residuals 187 96521017  516155
```

5.2.3 Prédictions à partir du modèle de régression

Comme dans le cas de la commande `aov` pour l'ANOVA, les commandes `fitted` et `predict` sont utilisées pour obtenir les valeurs ajustées et construire de nouvelles prédictions, respectivement. La commande `fitted(r)` renverra les valeurs prédites pour les poids des bébés observés.

```
> head(fitted(r))
```

```
  85   86   87   88   89   91
3176 3056 2835 2848 2844 2919
```

On peut facilement comparer les valeurs observées aux valeurs prédites par le modèle (situées sur la droite de régression) en juxtaposant les deux séries de valeurs de la manière suivante :

```
> head(cbind(birthwt$bwt, fitted(r)))
```

```
   [,1] [,2]
85 2523 3176
86 2551 3056
87 2557 2835
88 2594 2848
89 2600 2844
91 2622 2919
```

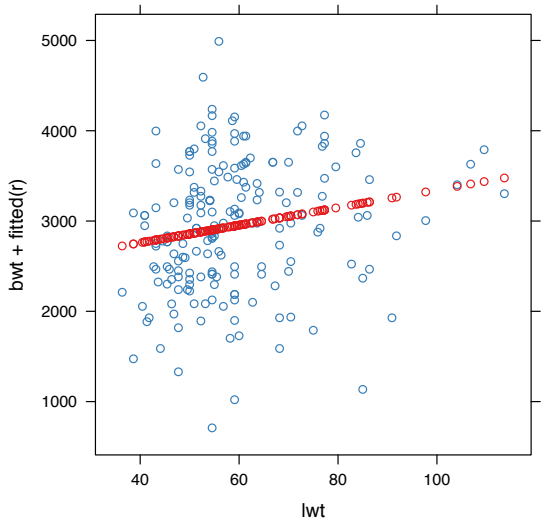
Il est également possible de rajouter au diagramme de dispersion précédent les valeurs prédites en modifiant légèrement la formule R utilisée : dans ce cas, on fera apparaître deux variables à gauche de l'opérateur de liaison ~ pour indiquer à R qu'il y a deux séries de mesure à représenter en fonction de la même variable reportée sur l'axe des abscisses (ici, `lwt`).

```
> xyplot(bwt + fitted(r) ~ lwt, data=birthwt)
```

La commande `predict` s'utilise pour obtenir les valeurs prédites sur un ensemble d'observations autre que celui observé dans les données de travail. Il est nécessaire d'indiquer à R les valeurs prises par la variable explicative, ici `lwt`, et de les stocker dans un `data.frame`. Voici un exemple où l'on estime les valeurs prédites par le modèle de régression pour des poids maternels allant de 35 à 120 kg, par pas de 5 kg :

```
> dp <- data.frame(lwt=seq(35, 120, by=5))
> bwtpred <- predict(r, newdata=dp)
> dp$bwt <- bwtpred
> head(dp)
```

```
  lwt  bwt
```



Données observées (bleu) et prédites (rouge) par le modèle de régression linéaire, `lm(bwt  lwt, data=birthwt)`.

```
1  35 2711
2  40 2759
3  45 2808
4  50 2857
5  55 2906
6  60 2954
```

Il est possible d'ajouter l'option `se=TRUE` pour obtenir l'erreur standard associée à chaque valeur prédite, et `interval="prediction"` pour un intervalle de confiance à 95 % pour la prédiction. En ce qui concerne les valeurs utilisées pour construire le modèle de régression, on peut se contenter de `interval="confidence"` pour construire des intervalles de confiance à 95 % pour les valeurs prédites (ajustées) sur les données actuelles.

5.2.4 Diagnostic du modèle et analyse des résidus

Le coefficient de détermination est un bon indicateur de la qualité du modèle, mais il ne renseigne pas sur les éventuelles observations qui pourraient avoir influé sur les paramètres du modèle (coefficients de régression). Pour cela, R offre toute une batterie de mesures d'influence des observations sur le modèle via la commande `influence.measures`. Ceci inclut les valeurs DFBETAS pour les deux coefficients de régression, les valeurs DFFIT, ou les distances de Cook. Il est également possible de calculer les résidus standardisés ou studentisés d'un modèle de régression à l'aide des commandes `rstandard` (ou `stdres` dans le package `MASS`) et `rstudent` (ou `studres` dans le package `MASS`).

À l'image de `fitted`, la commande `resid` permet d'obtenir la valeur des résidus (écarts entre les valeurs observées et les prédictions) du modèle de régression.

```
> head(resid(r))
  85   86   87   88   89   91
-653 -505 -278 -254 -244 -297
> head(birthwt$bwt - fitted(r))
  85   86   87   88   89   91
-653 -505 -278 -254 -244 -297
```

Une représentation graphique utile pour vérifier la constance de la variance consiste à afficher la valeurs des résidus en fonction des valeurs observées ou prédites par le modèle. Par exemple,

```
> xyplot(resid(r) ~ fitted(r), type=c("p","g"))
```

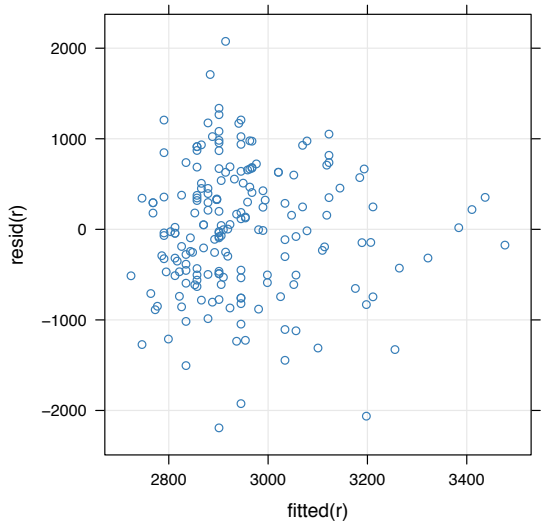


Diagramme de fluctuation des résidus en fonction des valeurs ajustées.

Pour tracer une droite horizontale coupant l'axe vertical en 0, il est possible d'ajouter l'option `abline=list(h=0, lty=2)`, par exemple.

Les autres conditions d'application du modèle de régression linéaire (linéarité et normalité des erreurs) peuvent se vérifier à l'aide d'un simple diagramme de dispersion (avec courbe loess, par exemple) et d'un histogramme des résidus.

5.2.5 Lien avec l'ANOVA

La commande `aov` repose essentiellement sur `lm` mais permet en plus de construire des termes d'erreur spécifiques pour les tests inférentiels. En revanche, on peut tout à fait étudier la relation entre une variable réponse numérique et un prédicteur catégoriel par une approche de régression : les résultats s'interpréteront sensiblement de la même manière. Sans entrer dans les détails du codage des contrastes et de leur manipulation sous R, on peut regarder les résultats d'une régression linéaire considérant les variables `bwt` et `race`, qui ont été étudiées au chapitre précédent (§ 4.2.1).

```
> lm(bwt ~ race, data=birthwt)


Call:
lm(formula = bwt ~ race, data = birthwt)

Coefficients:
(Intercept)    raceBlack    raceOther
       3103         -383         -297
```

La commande ci-dessus renvoit trois coefficients : un terme pour l'ordonnée à l'origine, et deux coefficients de régression correspondant aux deux dernières modalités du facteur `race`. R utilisant par défaut des contrastes de type traitement, ces résultats sont à interpréter de la manière suivante : l'ordonnée à l'origine représente le poids moyen des bébés pour les mères dont l'ethnicité est `White`, tandis que les deux autres termes représentent les différences entre les poids moyens pour les groupes `Black` et `Other` et le poids moyens des bébés du groupe `White`. Il est assez simple de le vérifier numériquement :

```
> bwtmeans <- with(birthwt, tapply(bwt, race, mean))
> bwtmeans

White Black Other
 3103  2720  2805

> bwtmeans[2:3] - bwtmeans[1]

Black Other
 -383  -297
```

On notera que l'on pourrait très bien supprimer le terme d'ordonnée à l'origine pour retrouver les trois moyennes de groupe. Pour cela, il est nécessaire de modifier légèrement la formule R :

```
> lm(bwt ~ race - 1, data=birthwt)


Call:
lm(formula = bwt ~ race - 1, data = birthwt)

Coefficients:
raceWhite  raceBlack  raceOther
     3103       2720       2805
```

Indépendemment de l'interprétation des coefficients de régression, on peut également vérifier que le tableau d'analyse de variance pour la régression fournit bien les mêmes informations que celui de l'ANOVA à un facteur.

```
> anova(lm(bwt ~ race, data=birthwt))
Analysis of Variance Table

Response: bwt
           Df   Sum Sq Mean Sq F value Pr(>F)
race        2  5015725 2507863    4.91 0.0083
Residuals 186 94953931  510505
```

(À comparer au tableau p. 33.)

5.3 Régression linéaire multiple

En ce qui concerne la régression multiple, on se contentera d'indiquer que l'usage de `lm` reste le même : on fait figurer à gauche de l'opérateur `~` la variable réponse, et à sa droite les variables explicatives que l'on souhaite inclure dans le modèle. Pour un modèle additif, on aura par exemple une formulation du type `lm(y ~ x1 + x2)` pour signifier à R de calculer les coefficients de régression (partiels) associés à `x1` et `x2`. La notation `x1:x2` pour désigner une interaction reste valable, et généralement l'une des deux variables sera catégorielle (on retrouve ce type de formulation dans l'analyse de covariance, par exemple).

Il est parfois utile de centrer la variable numérique, en particulier dans le cas où elle entre dans un terme d'interaction avec une variable qualitative ; ceci peut être réalisé à l'aide de la commande `scale`. Par exemple, la commande

```
> lm(bwt ~ scale(lwt, scale=FALSE) + ftv, data=birthwt)
```

permet d'estimer l'effet du poids de la mère (`lwt`, centré sur sa moyenne) et de la fréqeunce des visites chez le gynécologue durant le premier trimestre de grossesse.

Enfin, on peut tout à fait transformer les variables (réponse ou explicatives) lorsque l'on construit la formule R liant les variables. Par exemple, `y ~ log10(x1) + x2` permettrait de modéliser le logarithme (base 10) de `x1` plutôt que `x1` directement. On prendra garde au fait que l'inclusion de termes quadratiques, cubiques ou autres doit se faire de la manière suivante : `y ~ x1 + I(x1^2) + x2`, afin d'indiquer à R que la transformation `^2` s'applique aux valeurs de `x1` et ne doit pas être interprétée comme un terme de formule (en l'occurrence, la notation `^2` permet de construire automatique des termes d'interaction de second-ordre).

Ce qu'il faut retenir

- Avant d'estimer un coefficient de corrélation ou de régression supposant une relation *linéaire* entre deux variables, on représentera graphiquement la distribution conjointe des deux variables à l'aide d'un histogramme (`histogram` avec une option `type="smooth"`).
- Les commandes `cor` et `cor.test` permettent de calculer le coefficient de corrélation de Bravais-Pearson ou de Spearman (`method="spearman"`), et de tester l'hypothèse de nullité du paramètre.
- La commande `lm` permet de construire un modèle de régression linéaire ; elle s'utilise avec `summary` pour obtenir le tableau des coefficients de régression et `anova` pour le tableau de variance associé.

Cours 6. Mesures d'association en épidémiologie et régression logistique

Sommaire

Dans ce chapitre on revisite les mesures d'association classiquement retrouvées dans les études d'épidémiologie ou de recherche clinique, en particulier les mesures de risque telles que l'odds-ratio dans les études prospectives ou les mesures de sensibilité/spécificité d'un test diagnostique. Le modèle de régression logistique simple où l'on considère une seule variable explicative, numérique ou qualitative, est présenté en détails : estimation des paramètres du modèle, dans le cas des données individuelles ou groupées, prédiction ponctuelle et par intervalle. Enfin, la construction d'une courbe ROC complète ce chapitre portant sur le cas où la variable réponse est une variable binaire.

Commandes essentielles

`mantelhaen.test` : test de Mantel-Haenszel
`prediction` : prédiction pour un modèle diagnostique
`glm` : modèle de régression logistique
`anova` : tableau d'analyse de déviance pour la régression
`resid` : résidus du modèle
`epi.tests` : qualités diagnostiques d'un nouveau test
`woolf_test` : test de Woolf
`performance` : courbe ROC
`summary` (`summary.glm`) : tableau des coefficients
`fitted` : valeurs ajustées
`predict` : prédiction ponctuelle

6.1 Tableaux de contingence et mesures d'association

Remarque : la plupart des notions présentées dans cette partie ont déjà été discutées au chapitre 3. Il s'agit ici de simples rappels.

6.1.1 Tableaux de contingence

Les commandes `table` et `xtabs` permettent de construire des tableaux à deux dimensions, voire plus (c'est le cas de `xtabs`, mais également `ftable`). On rappelle le principe de la construction d'un tableau à partir d'une matrice de décompte des effectifs, ici avec les variables `low` et `race` dont les effectifs sont saisis manuellement :

```
> tab <- matrix(c(73, 23, 15, 11, 42, 25), nrow=2)
> rownames(tab) <- c(">= 2.5 kg", "< 2.5 kg")
> colnames(tab) <- c("White", "Black", "Other")
> tab
```

```
          White Black Other
>= 2.5 kg    73    15    42
< 2.5 kg     23    11    25
```

Lorsque l'on saisit les effectifs par colonnes, comme dans l'exemple ci-dessus, il suffit d'indiquer à R comment arranger le tableau en termes de nombre de lignes ou de colonnes. Si les effectifs sont saisis par ligne, il faut ajouter l'option `by.row=TRUE`. Bien évidemment, on a les mêmes résultats en utilisant directement le jeu de données :

```
> with(birthwt, table(low, race))
```

```
     race
low   White Black Other
  No     73    15    42
  Yes    23    11    25
```

qui est équivalent à `xtabs(~ low + race, data=birthwt)`.

6.1.2 Mesures et tests d'association

Les mesures et tests d'association pour deux variables qualitatives ont été présentés en § 3.3 (p. 27 et § 3.4 (p. 28). Pour l'odds-ratio et le risque relatif, on prendra garde à l'orientation du tableau (lignes et colonnes) et à l'ordre des modalités des variables binaires sur chaque dimension (généralement, événement positif en 1re ligne et 1re colonne).

6.1.3 Odds-ratio et stratification

Lorsque l'on s'intéresse au croisement de deux variables binaires selon les k niveaux ou modalités d'une troisième variable qualitative, considérée comme facteur de stratification, il est possible de calculer k odds-ratio, et s'ils s'avèrent qu'ils sont homogènes on peut combiner les estimations par strates en un odds-ratio commun. Le test permettant de vérifier l'homogénéité entre strates est le test de Mantel-Haenszel et s'obtient avec la commande `mantelhaen.test`. Les données doivent être arrangées sous la forme d'un tableau à 3 dimensions, la dernière dimension représentant les niveaux du facteur de stratification. Il y a deux approches possibles pour construire un tel tableau : à partir de matrices à deux dimensions (`matrix`) assemblées ensemble pour former un `array`, ou à partir de la commande `xtabs`, plus simple d'utilisation en règle générale.

Voici comment procéder avec une série de matrices résumant le tableau d'effectifs obtenu en croisant les modalités des variables `ui` (douleurs intra-utérines) et `low` (statut pondéral à la naissance), pour chaque modalité de la variable `race` (ethnicité).

```
> t1 <- with(subset(birthwt, race == "White"), table(ui, low))
> t2 <- with(subset(birthwt, race == "Black"), table(ui, low))
> t3 <- with(subset(birthwt, race == "Other"), table(ui, low))
> tab <- array(c(t1, t2, t3), dim=c(2,2,3),
+              dimnames=list(c("ui=0","ui=1"), c(">= 2.5 kg", "< 2.5 kg"),
+                levels(birthwt$race)))
> tab
```

```
, , White

     >= 2.5 kg < 2.5 kg
ui=0        65       18
ui=1         8        5

, , Black

     >= 2.5 kg < 2.5 kg
ui=0        14        9
ui=1         1        2

, , Other

     >= 2.5 kg < 2.5 kg
```

```
ui=0        37       18
ui=1         5        7
```

Notons qu'il est possible de nommer les lignes et colonnes du tableaux en utilisant

```
> dimnames(tab) <- list(c("ui=0","ui=1"), c(">= 2.5 kg", "< 2.5 kg"), levels(birthwt$race))
```

ou, comme cela a été fait ci-dessus, en utilisant directement l'option `dimnames=` de la commande `array`.

Voici la solution reposant sur la commande `xtabs`, pour laquelle on indique grâce à une formule la liste des variables à considérer, avec la variable de stratification en dernière position. On pourra vérifier que la commande suivante fournit bien le même tableau que l'illustration précédente.

```
> xtabs(~ ui + low + race, data=birthwt)
```

Enfin, quel que soit le tableau que l'on utilise, le résultat du test de Mantel-Haenszel indique que l'association entre le statut pondéral des bébés et la présence de douleurs intra-utérines n'est pas indépendante de l'ethnicité des mères, suggérant qu'il n'est pas possible de dériver une estimation de l'odds-ratio commun.

```
> mantelhaen.test(tab)


Mantel-Haenszel chi-squared test with continuity correction

data:  tab
Mantel-Haenszel X-squared = 4.23, df = 1, p-value = 0.03962
alternative hypothesis: true common odds ratio is not equal to 1
95 percent confidence interval:
 1.13 6.02
sample estimates:
common odds ratio
             2.61
```

R propose également le test de Woolf pour tester l'hétérogénité entre strates, voir le la commande `woolf_test` dans le package `vcd`.

6.2 Études diagnostiques

Dans les situations où l'on s'intéresse à la capacité prédictive ou discriminante d'un test de screening (malade/pas malade), connaissant le diagnostic des patients (via un jugement d'experts ou un test déjà existant servant de *gold standard*), la plupart des mesures d'intérêt (sensibilité, spécificité, valeurs prédictives positive et négative, odds-ratio pré ou post-test, etc.) peuvent se calculer à partir d'un tableau de contingence.

6.2.1 Sensibilité et spécificité d'un test diagnostique

Comme cela a été discuté en § 3.4 (p. 28), afin de calculer et d'interpréter correctement la plupart des mesures d'association sur un tableau 2×2, il convient de prendre garde à l'orientation du tableau (lignes et colonnes) et aux modalités reportées sur les lignes et les colonnes. Considérons les données suivantes :

```
> dat <- as.table(matrix(c(670,202,74,640), nrow = 2, byrow = TRUE))
> colnames(dat) <- c("Dis+","Dis-")
> rownames(dat) <- c("Test+","Test-")
> dat
      Dis+ Dis-
Test+  670  202
Test-   74  640
```

Il s'agit des résultats d'une étude de validation d'un nouveau test diagnostique réalisée chez 1586 patients. Parmi les 744 patients malades, 670 ont été identifiés comme tels par ce nouveau test. La commande `as.table`

n'est pas indispensable pour les calculs qui suivent. On notera simplement que les lignes et colonnes du tableau sont nommées ce qui permet d'adresser ses différentes cellules à l'aide des modalités des deux variables, plutôt que par les numéros de ligne ou de colonnes. La sensibilité et la spécificité peuvent se calculer par simple indexation du tableau, de la manière suivante :

```
> sens <- dat["Test+","Dis+"]/sum(dat[,"Dis+"])
> spec <- dat["Test-","Dis-"]/sum(dat[,"Dis-"])
> round(c(sens=sens, spec=spec), 3)
```

```
 sens  spec
0.901 0.760
```

On peut calculer des intervalles de confiance approximatifs à partir de la loi normale, en utilisant la formule $p \pm 1.96\sqrt{p(1-p)/n}$, où p est la proportion étudiée. Les bornes de l'intervalle de confiance serait ainsi estimées comme suit :

```
> sens + c(-1, 1) * 1.96 * sqrt(sens * (1-sens)/sum(dat))
```

```
[1] 0.886 0.915
```

6.2.2 Valeurs prédictives positive et négative

Le calcul de la valuer prédictive positive (VPP) et négative (VPN) peut s'effectuer sur le même principe.

```
> vpp <- dat["Test+","Dis+"]/sum(dat["Test+",])
> vpn <- dat["Test-","Dis-"]/sum(dat["Test-",])
> round(c(VPP=vpp, VPN=vpn), 3)
```

```
  VPP   VPN
0.768 0.896
```

Comme il s'agit de simple proportions, on peut tout à fait tester l'hypothèse nulle $H_0 : \ p = \frac{1}{2}$ à l'aide d'un test binomial. Pour la VPP, par exemple, on a les résultats suivants :

```
> binom.test(dat["Test+","Dis+"], sum(dat["Test+",]))
```

```
Exact binomial test

data:  dat["Test+", "Dis+"] and sum(dat["Test+", ])
number of successes = 670, number of trials = 872, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.739 0.796
sample estimates:
probability of success
                 0.768
```

On peut également utiliser ce type de distribution binomiale pour calculer des intervalles de confiance, comme illustré dans les paragraphes suivants.

6.2.3 Tableau de synthèse

La commande `epi.tests` du package `epiR` fournit l'ensemble des indicateurs précédents dans un seul et même tableau, avec des intervalles de confiance reposant sur la distribution binomiale.

```
> epi.tests(dat, conf.level = 0.95)
```

```
          Disease +    Disease -      Total
Test +          670          202        872
Test -           74          640        714
Total           744          842       1586
```

```
Point estimates and 95 % CIs:
-------------------------------------------------------------
Apparent prevalence                   0.55 (0.52, 0.57)
True prevalence                       0.47 (0.44, 0.49)
Sensitivity                           0.90 (0.88, 0.92)
Specificity                           0.76 (0.73, 0.79)
Positive predictive value             0.77 (0.74, 0.80)
Negative predictive value             0.90 (0.87, 0.92)
Positive likelihood ratio             3.75 (3.32, 4.24)
Negative likelihood ratio             0.13 (0.11, 0.16)
-------------------------------------------------------------
```

En fait, cette commande calcule d'autres quantités d'intérêt qui ne sont pas affichées mais que l'on peut obtenir assez simplement lorsque l'on stocke le résultat de l'appel à la commande `epi.tests` dans une variable auxiliaire et que l'on définit l'option `verbose=` à `TRUE`. Par exemple, supposons que l'on assigne le résultat produit par l'instruction précédente à une variable appelée `diagstats`.

```
> diagstats <- epi.tests(dat, conf.level = 0.95)
```

À partir de maintenant, on peut accéder aux autres statistiques calculées à partir du tableau `dat` en suffixant notre variable auxiliaire par le nom des statistiques décrites dans l'aide en ligne (`help(epi.tests)`. Les deux résultats précédents correspondent à l'OR diagnostique et aux nombre de patients à tester pour obtenir un résultat positif correct.

```
> diagstats$rval$diag.or

   est lower upper
1 28.7  21.5  38.2

> diagstats$rval$nnd

   est lower upper
1 1.51  1.41  1.65
```

6.3 Régression logistique

6.3.1 Estimation des paramètres du modèle

Dans le chapitre précédent, on s'est intéressés à la relation entre le poids des bébés à la naissance, traité comme une variable numérique, et le poids des mères. On peut s'intéresser au même type de relation, mais en considérant cette fois le statut pondéral à la naissance traité comme variable binaire (`low`).

Pour construire un modèle de régression logistique, on utilisera la commande `glm` qui fonctionne exactement de la même manière que la commande `lm` pour la régression linéaire, à ceci près qu'il faut indiquer à R quelle type de relation lie la variable réponse binaire à la combinaison linéaire du ou des prédicteurs d'intérêt. Pour la régression logistique, on modélisera le logit de la variable réponse.

```
> glm(low ~ lwt, data=birthwt, family=binomial("logit"))


Call:  glm(formula = low ~ lwt, family = binomial("logit"), data = birthwt)

Coefficients:
(Intercept)          lwt
     0.9983      -0.0309

Degrees of Freedom: 188 Total (i.e. Null);  187 Residual
Null Deviance:     235
Residual Deviance: 229  AIC: 233
```

La fonction de lien par défaut étant le logit, on peut simplifier l'option `family=` en indiquant simplement `binomial`. Comme dans le cas de l'ANOVA ou de la régression, la commande `glm` en elle-même ne fournit qu'une partie des informations et il est nécessaire de la coupler avec `summary` pour obtenir le tableau des coefficients de régression :

```
> m <- glm(low ~ lwt, data=birthwt, family=binomial("logit"))
> summary(m)


Call:
glm(formula = low ~ lwt, family = binomial("logit"), data = birthwt)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.095  -0.902  -0.802   1.361   1.982

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.9983     0.7853    1.27    0.204
lwt          -0.0309     0.0136   -2.28    0.023

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 234.67  on 188  degrees of freedom
Residual deviance: 228.69  on 187  degrees of freedom
AIC: 232.7

Number of Fisher Scoring iterations: 4
```

On procèdera de la même manière lorsque la variable explicative est une variable qualitative. Par exemple, si l'on souhaite modéliser la relation entre statut pondéral à la naissance et les antécédents de douleurs intra-utérines, on aura le modèle `low ~ ui`.

```
> table(birthwt$ui)


 No Yes
161  28
> levels(birthwt$ui)
[1] "No"  "Yes"
```

On prendra garde au fait que le premier niveau de la variable `ui` est la réponse `"No"` (pas de douleurs), mais si ce n'était pas le cas on pourrait interchanger les deux niveaux pour que l'interprétation des coefficients de régression se fasse par rapport à cette modalité. Pour cela, il suffirait de modifier l'ordre des niveaux à l'aide de la commande `relevel`, qui permet de changer le niveau de référence d'une variable qualitative :

```
> birthwt$ui <- relevel(birthwt$ui, ref="No")
```

Les résultats de ce deuxième modèle sont indiqués ci-après.

```
> m2 <- glm(low ~ ui, data=birthwt, family=binomial)
> summary(m2)


Call:
glm(formula = low ~ ui, family = binomial, data = birthwt)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
 -1.18   -0.81   -0.81    1.18    1.60

Coefficients:
```

```
            Estimate Std. Error z value   Pr(>|z|)
(Intercept)   -0.947      0.176   -5.39 0.00000007
uiYes          0.947      0.417    2.27      0.023

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 234.67  on 188  degrees of freedom
Residual deviance: 229.60  on 187  degrees of freedom
AIC: 233.6

Number of Fisher Scoring iterations: 4
```

L'odds-ratio s'obtient simplement en prenant l'exponentielle du coefficient de régression (pente), mais par souci de simplicité on peut appliquer cette transformation à l'ensemble des coefficients du modèle (on prendra garde que dans le cas de l'ordonnée à l'origine, ou *intercept*, on aura alors un odds simple).

```
> exp(coef(m2))
```

```
(Intercept)       uiYes
      0.388       2.578
```

(Pour sélectionner uniquement la pente, il suffit de remplacer l'expression précédente par `exp(coef(m2)["uiYes"])`.

Avec la commande `oddsratio` du package `vcd`, on peut vérifier que l'on retrouve bien la même valeur pour l'odds-ratio :

```
> oddsratio(xtabs(~ ui + low, data=birthwt), log=FALSE)
```

```
[1] 2.58
```

On peut toujours utiliser la commande `confint` sur les coefficients du modèle pour obtenir leurs intervalles de confiance à 95 %, par exemple :

```
> confint(m2)
```

```
             2.5 % 97.5 %
(Intercept) -1.301 -0.611
uiYes        0.125  1.772
```

À l'image du tableau de variance dérivé à partir d'une analyse de régression linéaire, il est possible de fournir une table de déviance pour le modèle de régression logistique. Cela se fait toujours avec la commande `anova`, mais en spécifiant le type de test qui doit être réalisé à l'aide de l'option `test=`, dans ce cas un test du χ^2 plutôt qu'un test F.

```
> anova(m2, test="Chisq")
```

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: low

Terms added sequentially (first to last)


     Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                   188        235
ui    1     5.08       187        230    0.024
```

6.3.2 Prédiction par intervalles

On utilisera toujours la commande `predict` pour estimer les valeurs prédites (ou ajustées) par le modèle pour les observations actuelles ou pour des observations futures. Dans ce dernier cas, il ne faut pas oublier de fournir

un `data.frame` avec les valeurs de la variable explicative pour lesquelles on souhaite calculer les prédictions.

Toutefois, à la différence de la régression linéaire, il est indispensable d'indiquer quel type de prédictions on souhaite calculer : des valeurs exprimées sur l'échelle du log-odds (par défaut, `type="link"`) ou en termes de probabilités de réponse (`type="response"`).

```
> head(predict(m2, type="link"))
```

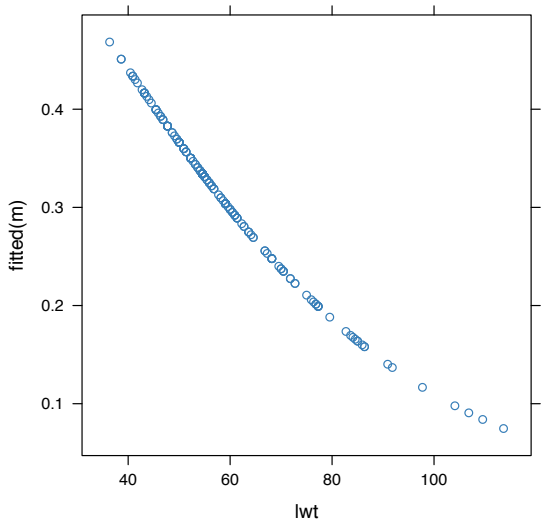
```
       85        86        87        88        89        91
 2.22e-16 -9.47e-01 -9.47e-01  2.22e-16  2.22e-16 -9.47e-01
```

```
> head(predict(m2, type="response"))
```

```
  85   86   87   88   89   91
0.50 0.28 0.28 0.50 0.50 0.28
```

On peut également utiliser la commande `fitted` pour obtenir les valeurs ajustées calculées sur l'échantillon. Dans le cas du premier modèle avec le poids des mères, on utiliserait ainsi :

```
> xyplot(fitted(m) ~ lwt, data=birthwt)
```



Valeurs ajustées pour le modèle `low ~ lwt`. Chaque point représente une prédiction individuelle.

Pour la prédiction par intervalles, on utilisera `predict`. Cependant, à la différence du cas de la régression linéaire, il n'y a pas d'option `interval=` mais on peut calculer des intervalles asymptotiques très facilement à partir du moment où l'on dispose de l'erreur-type d'estimation, obtenue grâce à l'option `se.fit=`. Soit

```
> pp <- predict(m2, type="link", se.fit=TRUE)
> lwr <- pp$fit - qnorm(0.975) * pp$se.fit
> upr <- pp$fit + qnorm(0.975) * pp$se.fit
> ppd <- data.frame(y=pp$fit, lwr=lwr, upr=upr)
> head(ppd)
```

```
           y    lwr    upr
85  2.22e-16 -0.741  0.741
86 -9.47e-01 -1.291 -0.603
87 -9.47e-01 -1.291 -0.603
88  2.22e-16 -0.741  0.741
89  2.22e-16 -0.741  0.741
91 -9.47e-01 -1.291 -0.603
```

Les intervalles ci-dessus sont pour les prédictions exprimées en unités logit. Pour repasser aux probabilités, on peut utiliser la commande `plogis` et l'appliquer à chacune des colonnes du tableau précédent.

```
> head(sapply(ppd, plogis))
```

```
        y   lwr   upr
[1,] 0.50 0.323 0.677
[2,] 0.28 0.216 0.354
[3,] 0.28 0.216 0.354
[4,] 0.50 0.323 0.677
[5,] 0.50 0.323 0.677
[6,] 0.28 0.216 0.354
```

On pourra vérifier que l'on obtient bien les mêmes résultats en faisant le calcul manuellement, ci-dessous dans le cas de la borne supérieure de l'IC :

```
> head(exp(pp$fit+1.96*pp$se.fit)/(1+exp(pp$fit+1.96*pp$se.fit)))
```

```
   85    86    87    88    89    91
0.677 0.354 0.354 0.677 0.677 0.354
```

6.3.3 Cas des données groupées

Dans l'exemple précédent, on disposait des données individuelles, c'est-à-dire des valeurs de `low` et `lwt` pour chaque unité statistique. C'est souvent le cas lorsque l'une des variables explicatives est de type numérique. Mais dans le cas de variables purement qualitatives, il arrive souvent que les données disponibles soient présentées sous la forme d'un tableau de contingence à deux dimensions, dans le cas le plus simple un tableau à deux dimensiosn croisant les deux modalités de la variable réponse (malade/pas malade) et celles de la variable explicative, typiquement un tableau produit par `xtabs` :

```
> tab <- xtabs(~ ui + low, data=birthwt)
> tab
```

```
     low
ui     No Yes
  No  116  45
  Yes  14  14
```

Une représentation alternative pour ces données consisterait à indiquer pour chaque modalité de la variable explicative le nombre d'événements positifs (ici, `low=1`) et le nombre total d'événements (ou d'observations), soit

```
> tab2 <- data.frame(ui=c("No","Yes"), n=c(45,14), tot=c(45+116, 14+14))
> tab2
```

```
   ui  n tot
1  No 45 161
2 Yes 14  28
```

Les trois formulations suivantes sont alors équivalentes :

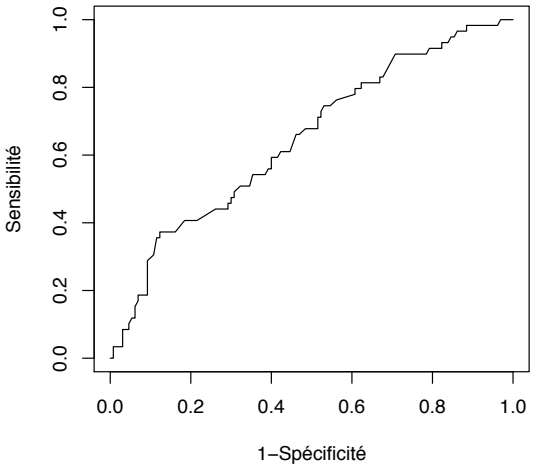
```
> glm(n/tot ~ ui, data=tab2, family=binomial, weights=tot)
> glm(cbind(n, tot-n) ~ ui, data=tab2, family=binomial)
> glm(tab ~ c(0,1), family=binomial)
```

6.4 Courbe ROC

Il existe de nombreux packages R permettant de construire une courbe ROC. Voici une illustration avec le package `ROCR` qui a l'avantage de ne pas avoir beaucoup de contraintes de formatage des données : à partir d'un modèle de régression, dans l'exemple ci-dessous le statut pondéral en fonction du poids et de l'éthnicité des mères, il suffit d'utiliser les deux commandes `prediction` et `performance` pour formater les données et calculer les statistiques nécessaires pour l'analyse ROC. Une commande `plot` se charge d'afficher les résultats de manière graphique, en fonction des statistiques choisies. Dans le cas le plus classique, on s'intéresse au

compromis entre la sensibilité (vrais positifs, ou `tpr`) et le complémentaire de la spécificité (faux positifs, ou `fpr`) ce qui se traduit par les instructions suivantes :

```
> library(ROCR)
> m3 <- glm(low ~ lwt + race, data=birthwt, family=binomial)
> pred <- prediction(predict(m3, type="response"), birthwt$low)
> perf <- performance(pred, "tpr","fpr")
> plot(perf, ylab="Sensibilité", xlab="1-Spécificité")
```



Courbe ROC pour le modèle `low lwt + race`. Une courbe s'écartant de la première bissectrice dans le quadrant supérieur gauche indique que les prédictions observées s'écartent de prédictions réalisées au hasard.

cmdprediction Il est donc nécessaire de fournir à la fois les prédictions du modèle (de type `"response"`) et le diagnostique réel (ici, `low`).

La commande `roc.from.table` du package `epicalc` permet de travailler directement à partir de données tabulées (cas du tableau croisant les résultats d'un test diagnostique avec un *gold standard*, par exemple). Dans le cas d'une régression logistique, il suffit de donner la variable contenant le résultat du modèle, dans notre cas `m3`. Voici deux exemples d'utilisation pour données groupées ou individuelles :

```
> roc.from.table(dat)  # § 6.2
> lroc(m2)
```

Ce qu'il faut retenir

— En plus de la commande `oddsratio`, les commandes `mantelhaen.test` et `woolf_test` sont utiles pour évaluer l'indépendance conditionnelle des associations entre deux variables et l'homégénéité des OR selon les niveaux définis par une troisième variable qualitative.

— La commande `epi.tests` du package `epiR` fournit l'essentiel des mesures permettant d'évaluer la qualité d'un test diagnostique.

— La commande `glm` permet de construire un modèle de régression logistique ; elle s'utilise avec `summary` pour obtenir le tableau des coefficients de régression et `anova` pour le tableau de variance associé.

Cours 7. Analyse des données de survie

Sommaire

Dans ce dernier chapitre, on s'intéresse à la modélisation des données de survie. Une attention particulière sera consacrée à la représentation des données censurées sous R et aux principales commandes permettant de manipuler et résumer ce type de données. L'estimation de la fonction de survie par la méthode de Kaplan-Meier et le modèle de régression de Cox sont ensuite discutés plus en détails.

Remarque : Dans ce chapitre, les illustrations sont réalisées à partir d'un jeu de données différent de celui utilisé jusqu'à présent. Les données sont disponibles dans le package `survival` et peuvent être chargées de la manière suivante :

```
> library(survival)
> data(lung)
```

Il s'agit de données sur la survie de patients atteints d'un cancer du poumon. Les variables d'intérêt sont les suivantes : `time` temps de survie en jours, `status` état à la date de point (1=donnée censurée, 2=patient décédé), `age` âge du patient et `sex` sexe du patient (1=homme, 2=femme). Au total, il y a 228 patients.

Commandes essentielles

`Surv` : création des variables de survie	`survfit` : fonction de survie
`plot` : affichage de la courbe de survie	`survdiff` : test du log-rank
`coxph` : modèle de Cox	`summary` (`summary.coxph`) : tableau des coefficients

7.1 Représentation des données et statistiques descriptives

7.1.1 Format de représentation des données

Les données censurées représentent un type à part, de même que les dates et les séries chronologiques. Pour l'essentiel, les commandes de gestion des données censurées sont fournies dans le package `survival`, qui est un des packages de base de R. Si l'on dispose d'une série d'événements codés en 0/1 et de données temporelles associées (durée, dates, etc.), la commande à utiliser est `Surv` où l'on spécifie les mesures temorelles (`time=`) et l'événement (`event=`). Il est également possible de spécifier un intervalle de temps lorsque les données s'y prêtent (voir l'option `time2=`). Dans tous les cas, le status de l'observation au moment de la date de point, ou de fin d'étude, doit permettre de distinguer les censures des événements survenus durant la date de suivi. Habituellement, les données sont codées en 0/1 ou 1/2 (1 ou 2 = patient décédé, ou survenue de l'événement d'intérêt). Le status peut également être codé en `TRUE/FALSE`, où `event=TRUE` signifie la survenue de l'événement (décès ou autre) et `event=FALSE` indique une donnée censurée (à droite).

```
> st <- with(lung, Surv(time=time, event=status))
> head(st)
```

```
[1]  306   455  1010+  210   883  1022+
```

Les données censurées sont signalées par la présence du suffixe +. C'est le cas de la 3e observation dans l'exemple ci-dessus : cet individu n'était pas décédé au bout de 1010 jours de suivi, mais on n'a plus d'information ensuite. On peut le vérifier à partir des données brutes en affichant les 6 premières valeurs pour le status vital des patients :

```
> head(subset(lung, select=c(time, status)))
```

```
  time status
1  306      2
2  455      2
3 1010      1
4  210      2
5  883      2
6 1022      1
```

7.1.2 Statistiques descriptives

On dispose des commandes habituelles (`mean`, `median`, `summary`, etc.) pour résumer les données numériques. Elles sont toutefois de peu d'intérêt pour l'estimation par intervalles et on va voir dans les paragraphes suivants que ces opérations peuvent être réalisées à partir de commandes prenant en compte les caractéristiques propres aux données de survie, en particulier la notion de censure des observations (plus aucune information à partir d'une certaine date).

7.2 Fonction de survie et courbe de Kaplan-Meier

7.2.1 Table de mortalité/survie

Il est possible de construire un tableau de survie et d'obtenir les probabilités de survie, éventuellement après stratification selon les niveaux d'un facteur de groupe, à l'aide de la commande `survfit`. Par défaut, `survfit` affiche la médiane de survie et son intervalle de confiance à 95 %

```
> s <- survfit(st ~ 1, data=lung)
> s
```

```
Call: survfit(formula = st ~ 1, data = lung)

records   n.max n.start  events  median 0.95LCL 0.95UCL
    228     228     228     165     310     285     363
```

L'option `conf.type=` permet éventuellement de modifier le type d'intervalles de confiance calculés pour l'estimateur de Kaplan-Meier. Par défaut, une transformation logarithmique est utilisée. Dans tous les cas, les IC sont estimés pour chaque point de la fonction de survie. Des informations complémentaires sont disponibles dans l'aide en ligne, voir `help(survfit.formula)`.

La commande `summary` permet de générer le tableau de survie et l'estimation non-paramétrique de la fonction de survie.

```
> summary(s)
```

Par défaut, R affiche les valeurs de la fonction de survie $S(t)$ pour l'ensemble des durées observées. On peut toutefois restreindre le calcul et l'affichage des résultat à une certaine fenêtre temporelle en utilisant l'option `times=`, par exemple

```
> summary(s, times=seq(1, 200, by=20))
```

```
Call: survfit(formula = st ~ 1, data = lung)

 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    1    228       0    1.000  0.0000        1.000        1.000
   21    220       8    0.965  0.0122        0.941        0.989
```

```
  41    217       3    0.952  0.0142        0.924        0.980
  61    211       7    0.921  0.0179        0.887        0.957
  81    205       7    0.890  0.0207        0.851        0.932
 101    196       6    0.864  0.0227        0.821        0.910
 121    189       6    0.837  0.0245        0.791        0.887
 141    184       5    0.815  0.0257        0.766        0.867
 161    176       8    0.780  0.0275        0.728        0.836
 181    159      15    0.713  0.0301        0.656        0.774
```

La commande `summary` fournit directement une estimation de la survie d'un individu à un instant donné. Par exemple, la probabilité qu'un patient soit encore vivant au bout de 300 jours s'obtient ainsi :

```
> summary(s, times=300)

Call: survfit(formula = st ~ 1, data = lung)

 time n.risk n.event survival std.err lower 95% CI upper 95% CI
  300     92     101    0.531  0.0346        0.467        0.603
```

ou plus simplement,

```
> summary(s, times=300)$surv

[1] 0.531
```

Si l'on souhaite comparer deux groupes, il sufffit d'ajouter la variable de groupement (de type `factor`) à droite de la formule, comme ceci :

```
> lung$sex <- factor(lung$sex, labels=c("Male", "Female"))
> s2 <- survfit(st ~ sex, data=lung)
> s2

Call: survfit(formula = st ~ sex, data = lung)

           records n.max n.start events median 0.95LCL 0.95UCL
sex=Male       138   138     138    112    270     212     310
sex=Female      90    90      90     53    426     348     550
```

La commande `summary` fonctionne ensuite sur le même principe et fournit une estimation de $S(t)$ pour chacun des groupes définis par les niveaux de la variable `sex`.

7.2.2 Courbe de Kaplan-Meier

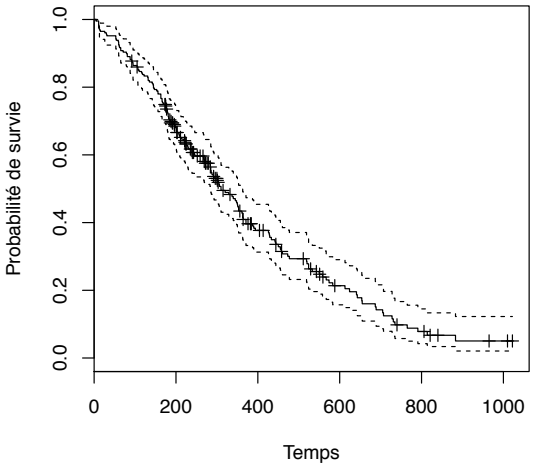
Pour représenter graphiquement la fonction de survie sous la forme d'une courbe de Kaplan-Meier, la commande `plot` doit être utilisée en combinaison avec `survfit`. Il est souvent plus pratique de sauvegarder dans une variable auxiliaire le résultat de la commande `survfit`, comme cela a été fait au paragraphe précédent (variables `s` et `s2`).

```
> plot(s, xlab="Temps", ylab="Probabilité de survie")
```

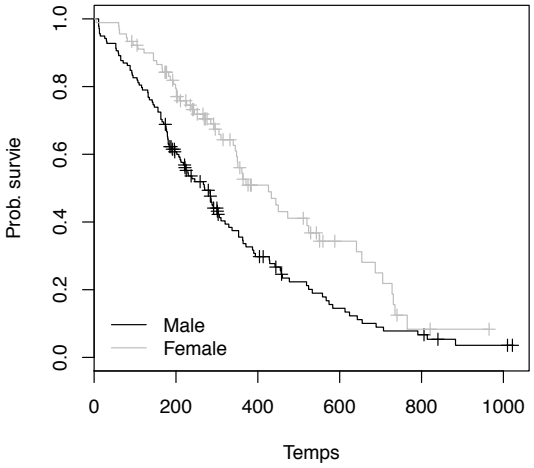
On remarquera que les commandes graphiques utilisées dans ce cas appartiennent aux commandes des graphiques de base de R et ne font pas partie du package `lattice`. Mais on peut toujours retrouver les informations qui nous intéressent, ici le temps et la probabilité de survie estimée par la méthode de Kaplan-Meier, et travailler directement avec la commande `xyplot`.

Par défaut, R affiche des intervalles de confiance à 95 % pour la courbe de survie. Pour les supprimer, il faut ajouter l'option `conf.int="none"` (anciennement `conf.int=0`). À l'inverse, lorsque l'on souhaite afficher plusieurs courbes de survie, par défaut R ne présente pas les intervalles de confiance. On peut les ajouter en ajoutant l'option `conf.int="both"` (anciennement `conf.int=1`). On peut ajouter une légende grâce à la commande `legend`.

```
> plot(s2, conf.int="none", col=c("black", "gray"), xlab="Temps", ylab="Prob. survie")
> legend("bottomleft", legend=levels(lung$sex), lty=1, col=c("black", "gray"), bty="n")
```



Fonction de survie calculée à partir de l'estimateur de Kaplan-Meier, avec intervalle de confiance à 95 %. Les censures apparaissent sous la forme de petits segments le long de la courbe de survie.



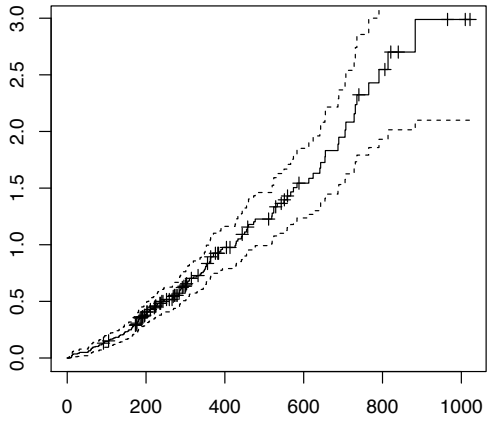
Fonction de survie pour chacun des niveaux de la variable `sex`. La commande `legend` permet de positionner une légende en bas à droite du graphique pour identifier les deux groupes.

L'option `bty="n"` permet de supprimer l'affichage d'un cadre autour de la légende. On remarquera que même si le rendu des lignes représentant les courbes de survie ne varie pas (trait plein, pointillés, etc.), il est tout de même nécessaire d'indiquer une option `lty=1` dans la commande `legend` pour afficher des lignes dans l'encadré de légende.

7.2.3 Fonction de risque cumulé

Il est également possible de travailler avec la fonction de risque $H(t)$ (*hazard rate*) qui est reliée à la fonction de survie par la relation $S(t) = \exp(-H(t))$. Une façon d'estimer cette courbe consiste à utiliser une transformation de type $\hat{H}(t) = -\log \hat{S}(t)$, ce que propose la commande `plot` pour les objets créés avec `survfit` en ajoutant l'option `fun="cumhaz"`.

```
> plot(s, fun="cumhaz")
```



7.2.4 Test d'égalité des fonctions de survie

Pour tester l'hypothèse nulle d'égalité de 2 ou plusieurs fonctions de survie, on peut utiliser le test du log-rank qui est disponible dans la commande `survdiff`.

```
> survdiff(st ~ sex, data=lung)

Call:
survdiff(formula = st ~ sex, data = lung)

             N Observed Expected (O-E)^2/E (O-E)^2/V
sex=Male   138      112     91.6      4.55      10.3
sex=Female  90       53     73.4      5.68      10.3

 Chisq= 10.3  on 1 degrees of freedom, p= 0.00131
```

Si l'on souhaite réaliser un test de Gehan-Wilcoxon à la place, il est nécessaire de préciser l'option `rho=1`.

```
> survdiff(st ~ sex, data=lung, rho=1)

Call:
survdiff(formula = st ~ sex, data = lung, rho = 1)

             N Observed Expected (O-E)^2/E (O-E)^2/V
sex=Male   138     70.4     55.6      3.95      12.7
sex=Female  90     28.7     43.5      5.04      12.7

 Chisq= 12.7  on 1 degrees of freedom, p= 0.000363
```

Lorsque l'on souhaite donner plus de poids à la première partie de la fonction de survie, on donne généralement des poids (`rho`) supérieurs à 0, alors qu'au contraire si l'on souhaite donner plus d'importance aux événements plus éloignés dans le temps on utilisera des valeurs négatives pour `rho`.

7.3 Régression de Cox

La commande permettant de réaliser une régression de Cox est `coxph`. Comme dans le cas de la régression linéaire (chapitre 5) ou logistique (chapitre 6), on associera cette commande qui permet de construire le modèle

de régression à la commande `summary` qui fournit les coefficients de régression et les tests de significativité associés.

```
> m <- coxph(st ~ sex, data=lung)
> m
```

```
Call:
coxph(formula = st ~ sex, data = lung)


            coef exp(coef) se(coef)     z      p
sexFemale -0.531     0.588    0.167 -3.18 0.0015

Likelihood ratio test=10.6  on 1 df, p=0.00111  n= 228, number of events= 165
```

La variable `st` est bien la représentation adéquate pour les données de survie et correspond à `Surv(time, status)`, où `time` dénote les durées et `status` code pour l'événement (0=donnée censurée, 1=l'événement s'est produit).

Il est possible de considérer différentes méthodes pour gérer le cas des ex-æquos grâce à l'option `method=`. Par défaut, R utilise la méthode de Efron, mais on peut ajouter `method="breslow"` pour utiliser la méthode de Breslow. On peut également faire intervenir des facteurs de stratification en utilisant l'instruction `strata` directement dans le membre droit de la formule R spécifiant le modèle. Par exemple, le modèle suivant est un modèle stratifié sur le sexe et considérant l'âge comme principale variable explicative.

```
> coxph(Surv(time, status) ~ age + strata(sex), data=lung)
```

Le tableau des coefficients de régression est donné ci-dessous :

```
> summary(m)
```

```
Call:
coxph(formula = st ~ sex, data = lung)

  n= 228, number of events= 165

            coef exp(coef) se(coef)     z Pr(>|z|)
sexFemale -0.531     0.588    0.167 -3.18   0.0015

          exp(coef) exp(-coef) lower .95 upper .95
sexFemale     0.588       1.7     0.424     0.816

Concordance= 0.579  (se = 0.022 )
Rsquare= 0.046   (max possible= 0.999 )
Likelihood ratio test= 10.6  on 1 df,   p=0.00111
Wald test            = 10.1  on 1 df,   p=0.00149
Score (logrank) test = 10.3  on 1 df,   p=0.00131
```

et, comme dans le cas des autres modèles de régression discutés aux chapitres précédents, il est possible d'obtenir directement les coefficients en utilisant la commande `coef`.

```
> coef(m)
```

```
sexFemale
   -0.531
```

On peut retrouver l'estimateur de Nelson-Aalen à partir du modèle de Cox de la manière suivante :

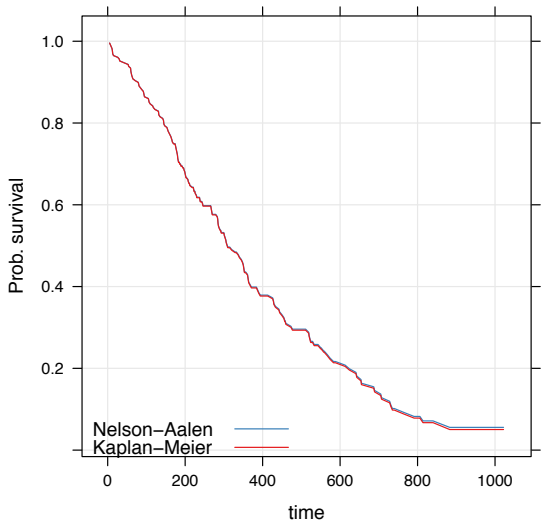
```
> sna <- survfit(coxph(st ~ 1), type="aalen")
> summary(sna)
```

et comparer avec les probabilités de survie estimées par la méthode de Kaplan-Meier (§ 7.2.3).

```
> h <- -log(sna$surv)
> hest <- data.frame(time=sna$time, d=sna$n.event, n=sna$n.risk, aalen=h, s=sna$surv)
> kmest <- data.frame(time=s$time, s=s$surv)
> psurv <- merge(hest, kmest, by="time")
```

Les commandes ci-dessus construisent un tableau de survie avec pour chaque valeur de temps, le nombre d'événements observés (`d`), le nombre de personnes à risque (`n`), l'estimateur de Nelson-Aalen (`aalen`) calculé comme l'anti-log de la probabilité de survie estimée via le modèle de Cox. La commande `merge` est utilisée pour assembler les deux tableaux de données, le premier contenant les données calculées à partir du modèle de Cox (`hest`) et le second les données calculées par la méthode actuariale en début de chapitre (`kmest`). L'opération de jointure des deux tableaux se fait à partir de la variable `time`. Après fusion des deux tableaux, la variable `s.x` désigne l'estimateur de Nelson-Aalen et `s.y` l'estimateur de Kaplan-Meier pour la fonction de survie. On peut les renommer pour plus de clarté et afficher les deux courbes de survie.

```
> xyplot(s.x + s.y ~ time, data=psurv, type=c("l", "g"), ylab="Prob. survival",
+        auto.key=list(corner=c(0,0), text=c("Nelson-Aalen", "Kaplan-Meier"),
+          lines=TRUE, points=FALSE))
```



Comparaison des fonctions de survie en utilisant l'estimateur de Kaplan-Meier ou de Nelson-Aalen.

Ce qu'il faut retenir

— Les données de survie sont représentées sous R à l'aide de la commande `Surv` dans laquelle on indique les informations concernant le temps et les événements (décès ou autre, et censure).
— L'estimateur de Kaplan-Meier s'obtient avec la commande `survfit` et on peut comparer différentes fonctions de survie grâce à la commande `survdiff` (test du log-rank ou de Wilcoxon).
— Le modèle de Cox fonctionne sur le même principe que les autres modèles de régression, et on utilise une notation par formule pour représenter la relation entre une variable réponse de type survie et des variables explicatives. La commande `summary` permet d'afficher les coefficients du modèle.

Annexes

Liste thématique de packages

Voici une liste des principaux packages qui complètent la batterie des commandes de base de R et se révèleront souvent utiles dans lors de l'analyse de données biomédicales. Il y a également pléthore de commandes plus ou moins redondantes sur le site CRAN (`http://cran.r-project.org`) et il est conseillé de se limiter à un minimum de packages additionnels afin de bien maîtriser les différentes commandes qu'ils fournissent.

Pour les méthodes d'analyse multivariées de type analyse en composantes principales ou analyse des correspondances (simple et multiple), on peut utiliser les packages `FactoMineR` ou `ade4`, ce dernier proposant des fonctionnalités propres à la modélisation des données écologiques. Le package `fpc` complète le package `cluster` pour les analyses de classification, et on trouvera d'autres packages pour les modèles de mélange tels que `mclust`.

La plupart des méthodes d'analyses factorielles sont disponibles dans le package `psych`, qui propose plus généralement de nombreuses commandes pour l'analyse psychométrique des tests. Le package `psy` fournit également quelques outils intéressant pour la psychométrie, et le package `irr` fournit l'essentiel des mesures d'accord inter-juges.

Les packages `vcd`, `gnm` et `vcdExtra` sont très utiles pour l'analyse des données catégorielles, et incluent des extensions pour les modèles log-linéaires. Les packages `epitools` et `Epi` complètent les fonctionnalités offertes par `epiR` pour l'analyse de données épidémiologiques.

Les packages `car` et `rms` sont essentiels pour la modélisation par régression (régression linéaire et logistique), et le package `effects` permet d'estimer des effets marginaux et les représenter graphiquement.

Le package `multcomp` fournit l'essentiel des commandes pour les comparaisons multiples, avec contrôle du risque expriméntal (FWER) ou du taux de fausses découvertes (FDR). Il fonctionne de concert avec la plupart des modèles de régression présentés dans ce cours.

Le package `lme4` étend les fonctionnalités du package `nlme` figurant parmi les packages installés avec R qui reste limité aux modèles mixtes pour une variable réponse continue. Les sorties graphiques reposent sur `lattice`.

Les graphiques avec le package lattice

Pourquoi le package lattice

Il existe trois principaux systèmes graphiques sous R : les commandes graphiques de base, dans le package `graphics`, inclut la plupart des outils nécessaires pour la représentation uni -ou multivariée des données numériques et catégorielles. Les commandes typiques sont `plot`, `hist`, `barplot`, `dotchart`, `qqplot`, etc. Voir pour des illustrations :

```
> demo(graphics)
```

Le package `lattice` (`http://lmdvr.r-forge.r-project.org/`) fournit des commandes équivalentes aux commandes de base, avec une interface de saisie unifiée, reposant sur l'usage de formules R, des options par défaut plus uniformes et, dans la plupart des cas, largement satisfaisantes pour des graphiques prêts à imprimer, ainsi que la construction automatique de graphique dits « en trellis ». Voir `demo(lattice)` pour des illustrations.

Le package `ggplot2` (`http://ggplot2.org`), qui offre une implémentation de la grammaire des graphiques développée par Leland Wilkinson, se démarque des deux autres systèmes graphiques par l'usage d'une syntaxe et de commandes différentes, tout en offrant un degré de souplesse incomparable pour la création de graphiques statistiques. Les deux derniers systèmes graphiques reposent sur un même package, `grid`.

Dans ce cours, nous avons privilégié le package lattice pour plusieurs raisons. Les relations entre variables sont spécifiées à l'aide de formules R (et d'une option `data=`), formules qui sont également utilisées dans les modèles statistiques (ANOVA et régression, entre autres) et certaines commandes de résumé numérique (`aggregate`, et `summary.formula` ou `summarize` du package Hmisc, `http://biostat.mc.vanderbilt.edu/wiki/Main/Hmisc`, cf. chapitre suivant). Les options par défaut (annotation des axes, taille de police et de symboles graphiques, etc.) sont généralement satisfaisantes. Pour une visualisation rapide, il existe des options intéressantes qui évitent d'avoir à combiner plusieurs commandes graphiques. Par exemple,

```
> xyplot(y ~ x, data=d, type=c("p", "g", "r"))
```

permet d'afficher un diagramme de dispersion (`y` en fonction de `x`) avec une droite de régression alors qu'en utilisant les graphiques de base on utiliserait une série de trois commandes :

```
> with(d, plot(x, y))  # ou plot(y ~ x, data=d)
> abline(lm(y ~ x, data=d))
> grid()
```

Les paragraphes qui suivent offrent un panorama des principales options. On se limite dans la plupart des cas à mentionner les principaux mots-clés afin que le lecteur puisse s'orienter dans l'aide en ligne (assez complexe pour ce package).

Principaux types de graphique

Dans ce qui suit, on présente les principales commandes graphiques pour les distributions uni- et bi-variées, avec éventuellement un conditionnement des résultats graphiques sur les niveaux d'une ou plusieurs variables catégorielles. Il est également possible de conditionner la relation entre deux variables par une troisième variable numérique mais ce cas ne sera pas traité (voir la notion de `shingle`). Dans chacun des cas, on indique entre parenthèses la commande de base équivalente. Par souci de simplicité, on omettra systématiquement le

nom du `data.frame` et on ne considèrera aucun facteur de conditionnement ou de stratification explicitement. On utilisera comme variables illustratives `x`, `y` (variables numériques) et `z` (variable catégorielle).

histogram `histogram(~ x)` (`hist(x)`) affiche un histogramme de fréquence, d'effectif (`type="count"`) ou de de densité (`type="density"`). Possibilité de conditionnement avec l'opérateur de formule `| z`.

densityplot `densityplot(~ x)` (`plot(density(x))`) affiche une courbe de densité non-paramétrique. Possibilité de conditionnement avec l'opérateur de formule `| z` et `groups=z`.

stripplot `stripplot(~ x)` ou `stripplot(z ~ x)` (`plot(z ~ x)`) affiche un diagramme de dispersion univarié, éventuellement conditionné sur une variable apparaissant à gauche de l'opérateur `~` dans la formule. Possibilité de conditionnement avec l'opérateur de formule `| z` et `groups=z`.

bwplot `bwplot(~ x)` (`boxplot(x)`) affiche un diagramme en forme de boîtes à moustaches. Le conditionnement se fait dans la formule directement, par exemple `bwplot(z ~ x)` (barres orientées horizontalement) ou `bwplot(x ~ z)` (barres orientées verticalement). On peut également utiliser l'opérateur de formule `| z`.

xyplot `xyplot(y ~ x)` (`plot(y ~ x)` ou `plot(x, y)`) affiche un diagramme de dispersion. Possibilité de conditionnement avec l'opérateur de formule `| z` et `groups=z`.

barchart `barchart(xtabs(~ z))` (`barplot(table(x))`) affiche un diagramme en barres. Possibilité de conditionnement avec l'opérateur de formule `| z`, ou en fournissant directement un tableau d'effectifs ou de proportions à deux entrées, par exemple `barchart(xtabs(~ z1 + z2), stack=FALSE)`.

dotplot `dotplot(xtabs(~ z))` (`dotchart(table(x))`, ou `dotchart(x)` si `x` est déjà une série de données aggrégées, par exemple des moyennes ou des effectifs). Possibilité de conditionnement avec l'opérateur de formule `| z` et `groups=z`. Remarque : `dotplot(z ~ x)` est essentiellement équivalent à `stripplot(z ~ x)`.

splom `splom(data.frame(x, y))` ou `splom(d)`, avec `d` un `data.frame` contenant des variables numériques. Possibilité de conditionnement avec l'opérateur de formule `| z` et `groups=z`.

qqmath `qqmath(~ x)` affiche un diagramme de type quantile-quantile. Possibilité de conditionnement avec l'opérateur de formule `| z` et `groups=z`.

De manière générale, lorsqu'un facteur de classification supplémentaire est ajouté, le conditionnement peut se faire de deux manières. Si la variable est placée dans la formule R avec un opérateur `| z`, alors R produira autant de graphiques pour la relation que de niveaux pour la variable `z`. Voici une illustration :

```
> xyplot(bwt ~ lwt | ui, data=birthwt)
```

Si, au contraire, on utilise `groups=z` R affiche toutes les données dans le même graphique mais utilise des symboles différents (points ou lignes) pour chaque niveau du facteur de conditionnement. Voici une illustration avec les mêmes données.

```
> xyplot(bwt ~ lwt, data=birthwt, groups=ui)
```

Dans certaines situations, il est intéressant de transformer le tableau de données à l'aide de la commande `melt` du package `reshape2` (ou `reshape`). C'est le cas en particulier lorsque l'on souhaite afficher dans le même graphique des boîtes à moustaches pour différentes variables numériques d'un `data.frame` mais sans considérer de facteur de classificaiton. Avec les données sur les poids à la naissance, on pourrait vouloir afficher la distribution des variables `bwt`, `lwt` et `age`. Une commande telle que

```
> bwplot(~ bwt + lwt + age, data=birthwt)
```

ne fonctionnera pas car dans ce cas R va afficher la distribution de la somme des trois variables (pour chaque observation). Au contraire, on utilisera (moyennant le problème de la standardisation des variables qui peut être réglé au préalable avec la commande `scale`) :

```
> library(reshape2)
> d <- melt(birthwt[,c("bwt","lwt","age")])
> bwplot(variable ~ value, data=d)
```

Cette notation permet également d'afficher la distribution de chaque variable dans des histogrammes séparés :

```
> histogram(~ value | variable, data=d)
```

On voit donc l'importance de transformer le tableau de données initial pour ré-exprimer les relations entre variables.

Voici quelques liens fournissant d'autres illustrations de chacune de ces commandes graphiques :

- Lattice graphics in R, `http://www.his.sunderland.ac.uk/~cs0her/Statistics/UsingLatticeGraphicsInR.htm`
- Creating graphs using the lattice package in R, `http://polisci.msu.edu/jacoby/icpsr/graphics/`
- Trellis Graphics : the Lattice Package (chapitre 4 du libre de Paul Murrell, *R Graphics*), `http://www.stat.auckland.ac.nz/~paul/RGraphics/chapter4.pdf` (PDF)
- Lattice and Other Graphics in R, `http://maths-people.anu.edu.au/~johnm/r-book/2edn/xtras/rgraphics.pdf` (PDF)

Pour une comparaison des packages `lattice` et `ggplot2`, voir le site `http://bit.ly/Xv7P2G`.

Options générales pour la personnalisation des graphiques

Type d'éléments graphiques et annotation

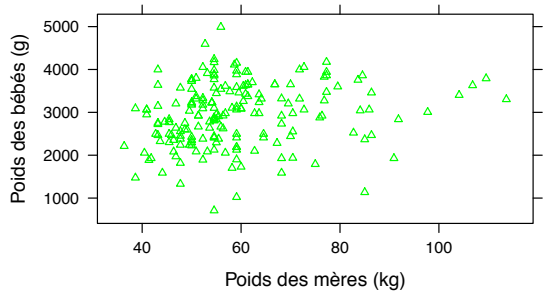
Concernant l'annotation des axes ou la personnalisation des éléments graphiques (points, lignes, etc.) , il est possible d'utiliser les mêmes options que pour les graphiques de base, à savoir

- `main=`, `xlab=` et `ylab=` pour modifier le titre, et le nom des axes x et y, respectivement. L'option est une chaîne de caractères, par exemple `"Mon titre"` ;
- `pch=` et `cex=`, pour modifier le type (1 = cercle, 2 = carré, etc.) et la taille relative des symboles (les valeurs ¡ 1 signifiant des symboles de taille réduite) ;
- `lty=` et `lwd=`, pour modifier le type (1 = trait plein, 2 = tirets, etc.) et la taille relative des lignes ;
- `col=` pour changer la couleur des points ou des lignes.

On peut se référer à l'aide en ligne pour plus d'informations : `help(par)`.

Par exemple, la commande affichera un diagramme de dipersion où les poids des bébés sont représentés en ordonnées (axe vertical, libellé « Poids des bébés ») en fonction du poids des mères, représentés en abscisses (axe horizontal, libéllé « Poids des mères »). Les observations seront représentées par des carrés (`pch=2`), de taille réduite (`cex=.6`, soit 60 % de la taille par défaut) et de couleur verte.

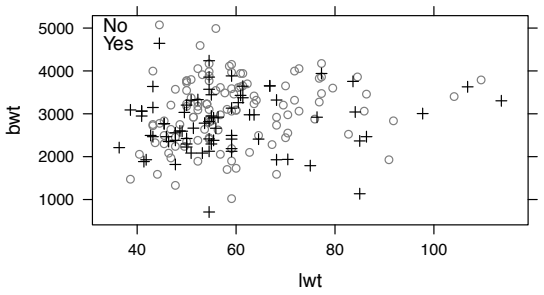
```
> xyplot(bwt ~ lwt, data=birthwt, pch=2, cex=.6, col="green",
+        xlab="Poids des mères (kg)", ylab="Poids des bébés (g)")
```



Gestion automatique des éléments graphiques à l'aide de thème

Les options décrites ci-dessus sont pratiques pour les graphiques simples, mais dans le cas où il existe un facteur de classification, il est plus commode d'utiliser un « thème » qui permet de gérer automatiquement les symboles associés à chaque niveau du facteur de classification et d'indiquer correctement les options graphiques appliquées dans la légende. Pour cela, on utilise la commande `simpleTheme` de la manière suivante :

```
> xyplot(bwt ~ lwt, data=birthwt, groups=smoke,
+        par.settings=simpleTheme(pch=c(1,3), col=c("gray50", "black")),
+        auto.key=list(corner=c(0,1)))
```



Le thème de couleur global peut également être modifié en utilisant les commandes `trellis.par.set`. Voici par exemple celui retenu pour ce document :

```
> library(latticeExtra)
> trellis.par.set(custom.theme.2())
> trellis.par.set("strip.background"=list(col="transparent"))
```

Pour obtenir un thème noir et blanc ou niveaux de gris, on peut utiliser les instructions suivantes :

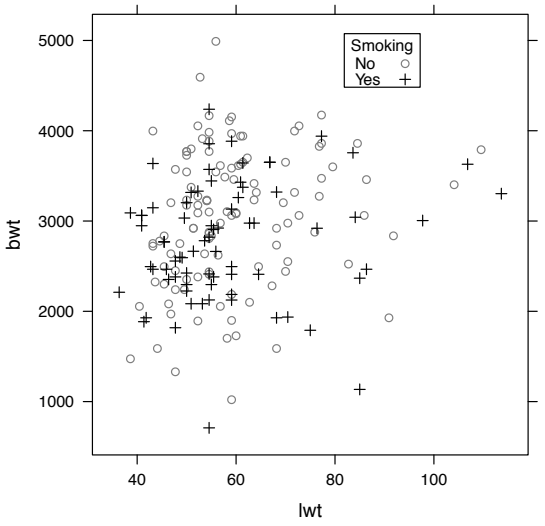
```
> ltheme <- canonical.theme(color=FALSE)
> ltheme$strip.background$col <- "transparent"
> lattice.options(default.theme=ltheme)
```

Dans le premier cas, on utilise un thème de couleurs fourni par le package `latticeExtra` et on supprime la couleur des bandes supérieures affichées dans le cas où il y a plusieurs panneaux graphiques (c'est-à-dire en présence d'un facteur de conditionnement). Dans le second cas, on utilise directement une commande du package `lattice` pour modifier le thème par défaut. On notera que ces modifications prennent effet sur le même « device » graphique : lorsque l'on ferme la fenêtre graphique ouverte par défaut par R, elles ne s'appliqueront plus.

Légende graphique

L'option `auto.key=` permet de générer une légende correspondant aux éléments graphiques présent dans la figure. Lorsque l'on utilise `simpleTheme`, on n'a pas à se soucier de la couleur des points ou du type de ligne, il suffit de gérer le placement de la légende. Il est possible d'indiquer dans quelle marge afficher la légende (par défaut, à droite), ou d'indiquer des coordonnées relatives (entre 0 et 1, pour chaque axe). Par exemple, `auto.key=list(space="top")` indique à R de placer la légende dans la marge supérieure du graphique. On peut également modifier l'arrangement des différents symboles de légende en ajoutant `columns=2` : les éléments seront ainsi disposés horizontalement sur 2 emplacements (s'il y a plus de deux niveaux pour le facteur de conditionnement, les éléments de légende seront empilés sur les deux mêmes emplacements). Concernant le placement en coordonnées relatives, on remplacera `space=` par `corner=` en indiquant une liste pour les coordonnées (x, y) de la partie supérieure gauche du cadre de légende. Il existe plusieurs autres options pour l'option de légende. On retiendra qu'il est possible d'ajouter un titre de légende avec une instruction du type `title=`. En reprenant un des exemples précédents, on utiliserait ainsi :

```
> xyplot(bwt ~ lwt, data=birthwt, groups=smoke,
+        par.settings=simpleTheme(pch=c(1,3), col=c("gray50", "black")),
+        auto.key=list(corner=c(0.7,0.95), title="Smoking", cex=.8, cex.title=0.8, border=TRUE))
```



Combinaison d'éléments graphiques

Il existe également une option `type=` qui permet de combiner différents éléments graphiques :

p afficher des points ;

l afficher des lignes ;

b afficher des points et des lignes ;

smooth afficher une courbe loess dans le cas d'un diagramme de dispersion, éventuellement pour chaque niveau d'un facteur de classification indiqué dans une option `groups=` ;

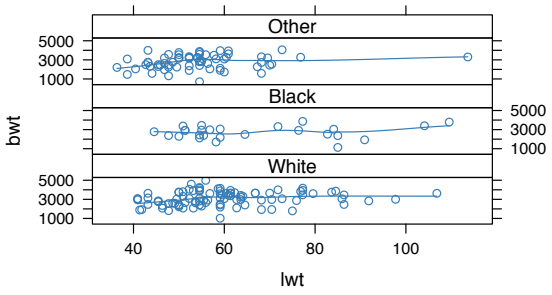
r afficher la droite de régression (éventuellement plusieurs droites de régression selon l'option de conditionnement `groups=`) ;

g afficher une grille alignée sur les intervalles définis pour les deux axes.

Organisation spatiale de plusieurs graphiques

Enfin, l'option `layout=` permet de spécifier l'arrangement des graphiques multiples en présence d'une variable de conditionnement.

```
> xyplot(bwt ~ lwt | race, data=birthwt, type=c("p", "smooth"),
+        layout=c(1,3))
```



Options spécifiques

Certaines commandes graphiques possèdent des options qui leur sont propres. Par exemple, les commandes `stripplot` et `xyplot` offrent toutes les deux la possibilité d'ajouter un léger décalage (horizontal et/ou vertical) des points afin d'éviter les problèmes de superposition dans le cas des grands échantillons. Avec `stripplot`, on utilise par exemple

```
> stripplot(~ bwt, data=birthwt, jitter.data=TRUE)
```

alors qu'avec `xyplot` on peut spécifier un décalage aléatoire dans les deux directions (horizontale et verticale). Voici un exemple d'usage :

```
> xyplot(bwt ~ race, data=birthwt, jitter.x=TRUE)
```

La plupart des options spécifiques des principales commandes graphiques peuvent se retrouver en demandant l'aide pour la commande `panel.*` où `*` désigne généralement le nom de la commande graphique principale. Par exemple, `help(panel.xyplot)` fournit la liste des options possibles pour une représentation de la relation entre deux variables sous forme de diagramme de dispersion.

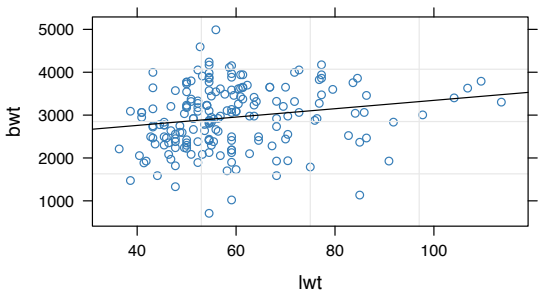
Composition graphique

La commande

```
> xyplot(bwt ~ lwt, data=birthwt, type=c("p", "g", "r"))
```

peut en fait se ré-écrire

```
> xyplot(bwt ~ lwt, data=birthwt,
+        panel=function(x, y, ...) {
+          panel.xyplot(x, y, ...)
+          panel.grid(...)
+          panel.lmline(x, y, ...)
+        })
```



L'instruction `panel=function(x, y, ...) { ... }` se charge de l'affichage de différents éléments graphiques, à partir des données `x` et `y` correspondant aux variables `lwt` et `btw`, respectivement. Ensuite, on ajoute les éléments graphiques suivants : des points dont les coordonnées sont définies par `x` et `y` (donc les valeurs prises par `lwt` et `btw`) avec `panel.xyplot`, une grille alignée sur les coordonnées affichées des axes avec `panel.grid`, et enfin une droite de régression avec la commande `panel.lmline`.

Il existe des cas plus complexes que l'exemple ci-dessus, notamment lorsque la commande graphique inclut une variable de conditionnement produisant différents graphiques (opérateur `|`) ou différents types de points ou lignes (options `groups=`). Ces cas se gèrent en ajoutant une option `subscripts` ou `groups` dans la déclaration de la fonction `panel`. Voir aussi les commandes `packet.number` et `panel.number` dans l'aide en ligne.

Pour aller plus loin

La référence définitive sur le package `lattice` est

> Sarkar, D. (2008). *Lattice : Multivariate Data Visualization with R* (Use R !). Springer.

On trouvera également des informations utiles sur le site companion, en particulier les commandes R permettant de générer différents types de graphiques : `http://lmdvr.r-forge.r-project.org/figures/figures.html`.

La package `latticeExtra` (`http://latticeextra.r-forge.r-project.org`) offre également des extensions aux commandes `lattice`, en particulier pour les graphiques multivariés et la personnalisation (thème) des graphiques.

Les packages Hmisc et rms

Les packages `Hmisc` et `rms` offrent un ensemble de fonctionnalités additionnelles pour la gestion des données, la production de tableaux et de graphiques de résumé descriptifs de variables numériques et qualitatives et la modélisation (régression linéaire, logistique et analyse de données de survie).

Les commandes essentielles de Hmisc

Gestion de données

Pour la gestion de données, `Hmisc` offre plusieurs fonctionnalités intéressantes. Ce package fournit des commandes permettant l'importation de fichiers CSV (`csv.get`), SAS (`sas.get`), SPSS (`spss.get`) ou Stata (`stata.get`). Ces commandes se chargent de convertir les noms de variable en minuscules, de recoder les dates si besoin. Les deux dernières commandes reposent en fait sur les commandes du package `foreign`. Avant d'utiliser ce package, il est nécessaire de le charger au préalable avec la commande `library(Hmisc)`.

La commande `contents` fournit un résumé du format de représentation des variables et de la présence éventuelle de valeurs manquantes :

```
> library(rms)
> birthwt$age[5] <- NA
> birthwt$ftv[sample(1:nrow(birthwt), 5)] <- NA
> contents(birthwt)

Data frame:birthwt 189 observations and 12 variables    Maximum # NAs:5


      Levels   Class Storage NAs
low        2         integer   0
age                  integer   1
lwt                   double   0
race       3         integer   0
smoke      2         integer   0
ptl                  integer   0
ht         2         integer   0
ui         2         integer   0
ftv                  integer   5
bwt                  integer   0
ftv2       3         integer   0
ftv3       3 ordered integer   0

+--------+----------------+
|Variable|Levels          |
+--------+----------------+
|  low   |No,Yes          |
|  smoke |                |
|  ht    |                |
|  ui    |                |
```

```
+--------+-----------------+
|  race  |White,Black,Other|
+--------+-----------------+
|  ftv2  |0,1,2+           |
|  ftv3  |                 |
+--------+-----------------+
```

On a vu qu'il était possible d'associer des étiquettes aux modalités d'une variable qualitative avec l'option `levels=` de la commande `factor`, ou directement avec la commande `levels`. Il est également possible d'associer des étiquettes aux variables, et des unités de mesure, lorsque cela s'applique, grâce aux commandes `label` (sans « s ») et `units`.

```
> label(birthwt$age) <- "Age de la mère"
> units(birthwt$age) <- "années"
> label(birthwt$bwt) <- "Poids du bébé à la naissance"
> units(birthwt$bwt) <- "grammes"
```

La commande `label` peut également servir à fournir des informations sur un `data.frame`, par exemple

```
> label(birthwt, self=TRUE) <- "Données sur les poids à la naissance de Hosmer et Lemeshow."
```

Ces étiquettes peuvent ensuite être associées automatiquement dans les tableaux, graphiques ou résumés produits par `Hmisc`. La commande `describe` fournit une synthèse descriptive de l'ensemble des variables d'un `data.frame` ou d'une liste de variables, à l'image d'un « codebook ».

```
> describe(subset(birthwt, select=c(age, race, bwt, low)))
```

```
subset(birthwt, select = c(age, race, bwt, low))

 4  Variables      189  Observations
---------------------------------------------------------------------------------------------------------
age : Age de la mère [années]
      n missing  unique    Info    Mean     .05     .10     .25     .50     .75     .90     .95
    188       1      24       1   23.27      16      17      19      23      26      31      32

lowest : 14 15 16 17 18, highest: 33 34 35 36 45
---------------------------------------------------------------------------------------------------------
race
      n missing  unique
    189       0       3

White (96, 51%), Black (26, 14%), Other (67, 35%)
---------------------------------------------------------------------------------------------------------
bwt : Poids du bébé à la naissance [grammes]
      n missing  unique    Info    Mean     .05     .10     .25     .50     .75     .90     .95
    189       0     131       1    2945    1801    2038    2414    2977    3487    3865    3997

lowest :  709 1021 1135 1330 1474, highest: 4167 4174 4238 4593 4990
---------------------------------------------------------------------------------------------------------
low
      n missing  unique
    189       0       2

No (130, 69%), Yes (59, 31%)
---------------------------------------------------------------------------------------------------------
```

La commande `cut2` remplace avantageusement la commande `cut` que l'on trouve souvent combinée à `quantile` pour convertir des variables numériques en variables qualitatives avec des classes équilibrées (et elle permet d'éviter d'avoir à se rappeler l'option `include.lowest=TRUE`).

```
> table(cut2(birthwt$lwt, g=4))

[36.4, 50.9) [50.9, 55.5) [55.5, 64.1) [64.1,113.6]
          53           43           46           47
> table(cut2(birthwt$age, g=3, levels.mean=TRUE))

18.074 23.091 30.019
    68     66     54
```

On peut spécfiier le nombre de groupes désirés (`g=`) ou le nombre minimal d'observations par groupe (`m=`). L'option `levels.mean=` permet de renvoyer comme niveau de facteur le centre de classe plutôt que les bornes de l'intervalle.

Pour les données de cohorte ou de survie, la commande `event.chart` fournit des représentations graphiques intéressantes pour visualiser les durées de suivi, les censures, etc. Voir l'aide en ligne et les exemples fournis :

```
> example(event.chart)
```

Il existe également un certain nombre de commandes permettant d'effectuer de l'imputation simple ou multiple (voir `impute` et `aregImpute`). Un exemple d'imputation par la médiane pour les données supprimées dans la variable `ftv` plus haut est donné ci-dessous :

```
> ftvi <- impute(birthwt$ftv)
> summary(ftvi)

 5 values imputed to 0
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    0.00    0.00    0.78    1.00    6.00
```

Les observations imputées sont signalées d'une astérisque lorsqu'on affiche l'ensemble des valeurs prises par la variable.

Statistiques descriptives

La commande `summarize` fonctionne sur le même principe que `aggregate` et permet de fournir pour une variable numérique des résumés stratifiés selon une ou plusieurs variables qualitatives. À la différence de `aggregate`, il est possible de calculer plusieurs statistiques et de les stocker dans des colonnes différents d'un même tableau de résultats (`aggregate` stocke en fait tous ses résultats dans une seule et même colonne, ce qui n'est pas gênant pour l'affichage mais ne facilite pas l'exploitation des résultats).

```
> f <- function(x) c(mean=mean(x), sd=sd(x))
> with(birthwt, summarize(bwt, race, f))

   race  bwt  sd
3 White 3103 728
1 Black 2720 639
2 Other 2805 722
```

La commande `summary` (en fait, `summary.formula`) peut s'utiliser avec une notation par formule décrivant les relations entre plusieurs variables. On distingue trois principales méthodes (`method=`) : `"response"`, où l'on résume une variable numérique selon les niveaux d'une ou plusieurs variables (dans le cas de variables numériques, `Hmisc` recodera automatiquement la variable en 4 classes équilibrées), `"cross"` pour calculer moyennes conditionnelles et marginales d'une ou plusieurs variables réponse en fonction de variables qualitatives ou numériques (3 au maximum), et `"reverse"` pour résumer de manière univariée un ensemble de variables numériques (trois quartiles) ou qualitatives (effectif et proportion) selon les niveaux d'un facteur. La position des variables par rapport à l'opérateur de liaison `~` de la formule joue un rôle particulièrement important pour déterminer le type de résumé à réaliser selon la méthode sélectionnée. Pour les méthodes `"response"` et `"reverse"`, on peut coupler la commande `plot` directement avec le résultat renvoyé par `summary` pour avoir

une représentation graphique des résultats. On peut également fournir une formule et indiquer à R quelle commande appliquer pour résumer la structure de données. Par exemple, dans le cas `low ~ race + ht`, l'option `fun=table` permettra de construire automatiquement les deux tableaux de contingence correspondant au croisement des variables `low` avec `race` et `low` avec `ht`.

Voici quelques illustrations :

```
> library(Hmisc)
> summary(bwt ~ race + ht + lwt, data=birthwt)
```

```
Poids du bébé à la naissance    N=189

+-------+------------+---+----+
|       |            |N  |bwt |
+-------+------------+---+----+
|race   |White       | 96|3103|
|       |Black       | 26|2720|
|       |Other       | 67|2805|
+-------+------------+---+----+
|ht     |No          |177|2972|
|       |Yes         | 12|2537|
+-------+------------+---+----+
|lwt    |[36.4, 50.9)| 53|2656|
|       |[50.9, 55.5)| 43|3059|
|       |[55.5, 64.1)| 46|3075|
|       |[64.1,113.6]| 47|3038|
+-------+------------+---+----+
|Overall|            |189|2945|
+-------+------------+---+----+
```

```
> summary(cbind(lwt, age) ~ race + bwt, data=birthwt, method="cross")
```

```
 mean by race, bwt

+-------+
|N      |
|Missing|
|lwt    |
|age    |
+-------+
+-----+----------+----------+----------+----------+----+
| race|[ 709,2424)|[2424,3005)|[3005,3544)|[3544,4990]| ALL|
+-----+----------+----------+----------+----------+----+
|White|     19    |     23    |     20    |     33    | 95 |
|     |    0      |    1      |    0      |    0      |1   |
|     |    55.6   |    57.7   |    62.2   |    63.3   |60.1|
|     |    22.7   |    24.8   |    24.5   |    24.9   |24.4|
+-----+----------+----------+----------+----------+----+
|Black|      9    |      9    |      6    |      2    | 26 |
|     |    0      |    0      |    0      |    0      |0   |
|     |    65.1   |    59.7   |    70.8   |    93.4   |66.7|
|     |    23.4   |    20.9   |    20.0   |    20.5   |21.5|
+-----+----------+----------+----------+----------+----+
|Other|     20    |     16    |     19    |     12    | 67 |
|     |    0      |    0      |    0      |    0      |0   |
|     |    51.2   |    53.0   |    58.9   |    55.3   |54.6|
|     |    22.2   |    22.7   |    22.3   |    22.5   |22.4|
+-----+----------+----------+----------+----------+----+
|ALL  |     48    |     48    |     45    |     47    |188 |
|     |    0      |    1      |    0      |    0      |1   |
```

```
|     |    55.5   |    56.5   |    62.0   |    62.5   |59.1|
|     |    22.6   |    23.4   |    23.0   |    24.1   |23.3|
+-----+-----------+-----------+-----------+-----------+----+
```

```
> summary(low ~ race + age + ui, data=birthwt, method="reverse", overall=TRUE)
```

```
Descriptive Statistics by low
```

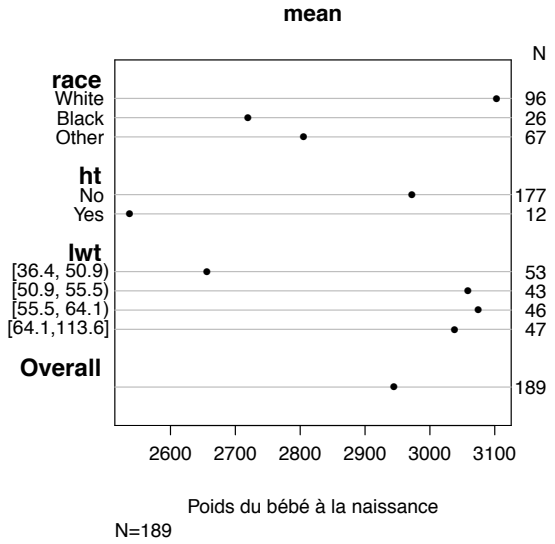
```
+----------------------+---+--------------+--------------+--------------+
|                      |N  |No            |Yes           |Combined      |
|                      |   |(N=130)       |(N=59)        |(N=189)       |
+----------------------+---+--------------+--------------+--------------+
|race : White          |189|   56% (73)   |   39% (23)   |   51% (96)   |
+----------------------+---+--------------+--------------+--------------+
|    Black             |   |   12% (15)   |   19% (11)   |   14% (26)   |
+----------------------+---+--------------+--------------+--------------+
|    Other             |   |   32% (42)   |   42% (25)   |   35% (67)   |
+----------------------+---+--------------+--------------+--------------+
|Age de la mère [années]|188|19.0/23.0/28.0|19.5/22.0/25.0|19.0/23.0/26.0|
+----------------------+---+--------------+--------------+--------------+
|ui : Yes              |189|   11% ( 14)  |   24% ( 14)  |   15% ( 28)  |
+----------------------+---+--------------+--------------+--------------+
```

```
> summary(low ~ race + ht, data=birthwt, fun=table)
```

```
low    N=189

+-------+-----+---+---+---+
|       |     |N  |No |Yes|
+-------+-----+---+---+---+
|race   |White| 96| 73|23 |
|       |Black| 26| 15|11 |
|       |Other| 67| 42|25 |
+-------+-----+---+---+---+
|ht     |No   |177|125|52 |
|       |Yes  | 12|  5| 7 |
+-------+-----+---+---+---+
|Overall|     |189|130|59 |
+-------+-----+---+---+---+
```

```
> plot(summary(bwt ~ race + ht + lwt, data=birthwt))
```



Lorsque l'on utilise `method="reverse"`, il est possible d'ajouter l'option `test=TRUE` pour obtenir automatiquement un test de comparaison de moyenne ou de fréquence entre les groupes définis par la variable de stratification. Il est également possible d'exporter automatiquement les tableaux produits au format LaTeXou PDF.

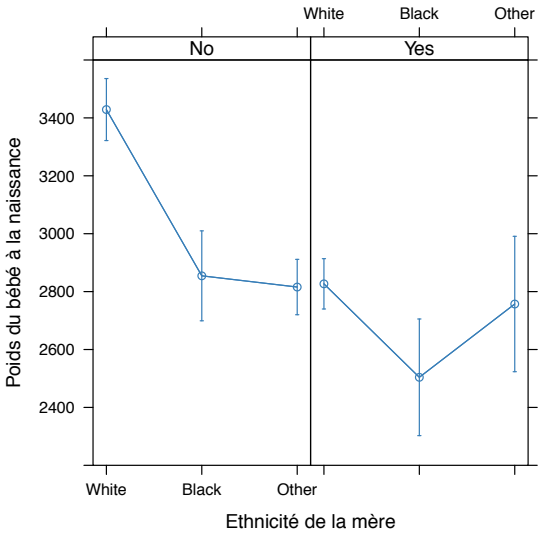
Commandes graphiques

Le package `Hmisc` repose pour l'essentiel sur les graphiques `lattice`, mais il fournit des commandes « plus intégrées ». Parmi les commandes graphiques fournies par `Hmisc`, on distingue les diagrammes en points (`Dotplot` et `dotchart2`), les diagrammes en boîtes à moustaches (`panel.bpplot` à utiliser avec la commande usuelle `bwplot`) et les diagrammes de dispersion (`xyplot`). On se contentera de présenter un exemple pour ce dernier cas.

```
> f <- function(x) c(mean(x), mean(x) + c(-1, 1) * sd(x)/sqrt(length(x)))
> bwtmeans <- with(birthwt, summarize(bwt, llist(race, smoke), f))
> names(bwtmeans)[4:5] <- c("lwr", "upr")
> bwtmeans
```

```
  race smoke  bwt  lwr  upr
5 White    No 3429 3322 3536
6 White   Yes 2827 2740 2914
1 Black    No 2854 2699 3010
2 Black   Yes 2504 2303 2705
3 Other    No 2816 2720 2911
4 Other   Yes 2757 2523 2991
```

```
> xYplot(Cbind(bwt, lwr, upr) ~ numericScale(race, label="Ethnicité de la mère") | smoke,
+        data=bwtmeans, type="b", ylim=c(2200, 3600),
+        scales=list(x=list(at=1:3, labels=levels(birthwt$race))))
```



On notera que `Cbind` prend un « c » majuscule, pour différencier cette commande de la commande de base, et que si elle existe l'étiquette de la variable remplace automatiquement le nom de la variable (c'est le cas ici pour la variable `bwt`). Il est également possible de libeller automatiquement les diféfrentes courbes, segments ou points définis par une variable de conditionnement à l'aide de l'option `keys=`, comme dans l'exemple suivant.

```
> xYplot(Cbind(bwt, lwr, upr) ~ numericScale(race), groups= smoke,
+        data=bwtmeans, type="l", keys="lines")
```

Les commandes essentielles de rms

Pour utiliser les commandes du package `rms`, on chargera au préalable le package à l'aide de la commande `library`. Notons que ce package charge automatiquement le package `Hmisc`.

```
> library(rms)
```

Pour la régression linéaire, il s'agit de la commande `ols`. Le principe d'utilisation reste le même que pour la commande `lm` : on spécifie à l'aide d'une formule le rôle de chaque variable (variable réponse ~ variables explicatives) et le `data.frame` dans lequel chercher les données. Dans son usage le plus simple, on se contente d'afficher les résultats renvoyés par `ols` avec la commande `print` ou en tapant simplement le nom de la variable auxiliaire dans laquelle on a stocké les résultats du modèle. Pour obtenir des informations additionnelles, en particulier concernant les effets marginaux, les résidus du modèle, et profiter de fonctionnalités graphiques étendues, il est nécessaire d'utiliser la commande `datadist` pour stocker les prédicteurs du modèle dans une structure à part. On disposera alors de commandes telles que `summary` ou `plot`.

Exemple. Modélisation du poids des bébés en fonction de différents indicateurs de risque chez la mère.

```
> m <- ols(bwt ~ age + lwt + race + ftv, data=birthwt, x=TRUE)
> m

Linear Regression Model

ols(formula = bwt ~ age + lwt + race + ftv, data = birthwt, x = TRUE)

Frequencies of Missing Values Due to Each Variable
 bwt  age  lwt race  ftv
   0    1    0    0    5
```

```
                 Model Likelihood     Discrimination
                    Ratio Test           Indexes
Obs        184    LR chi2     16.46    R2       0.086
sigma 704.9359    d.f.            5    R2 adj   0.060
d.f.       178    Pr(> chi2) 0.0056    g      241.224

Residuals

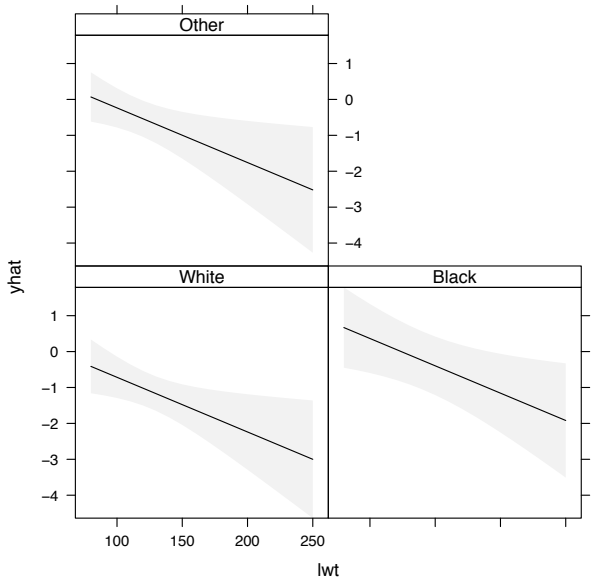
Poids du bébé à la naissance [grammes]
     Min       1Q   Median       3Q      Max
-2119.06  -462.28    39.96   478.27  1876.45

            Coef      S.E.     t     Pr(>|t|)
Intercept  2417.7530 319.2928  7.57 <0.0001
age           2.7483  10.3747  0.26 0.7914
lwt          10.1881   3.9790  2.56 0.0113
race=Black -439.1451 162.0372 -2.71 0.0074
race=Other -222.3614 117.6767 -1.89 0.0604
ftv           2.5163  50.2617  0.05 0.9601
```

Pour la régression logistique (simple, multiple ou ordinale), on utilise la commande `lrm`. Cette commande choisit automatiquement le modèle selon le nombre de niveaus de la variable réponse : dans le cas où il y en a plus de deux, un modèle à odds proportionnels est utilisé.

Exemple. Modélisation du poids des bébés (0/1) en fonction des mêmes variables explicatives.

```
> ddist <- datadist(birthwt)
> options(datadist='ddist')
> m <- lrm(low ~ age + lwt + race + ftv, data=birthwt)
> print(m)
> summary(m)
> m2 <- update(m, . ~ . - age - ftv)
> lrtest(m2, m)
> anova(m2)
> pm2 <- Predict(m2, lwt=seq(80, 250, by=10), race)
> print(xYplot(Cbind(yhat,lower,upper) ~ lwt | race, data=pm2,
+              method="filled bands", type="l", col.fill=gray(.95)))
```



Enfin, pour la régression de Cox, il s'agit de la commande `cph`. La représentation des données de survie passe toujours par une étape préalable de conversion avec la commande `Surv`.

Pour aller plus loin

Il existe un très bon tutoriel sur `Hmisc` (anciennement `Design`), *An Introduction to S and the Hmisc and Design Libraries ; CF Alzola and FE Harrell* (PDF, 310 pages), disponible à l'adresse suivante : `http://biostat.mc.vanderbilt.edu/Hmisc`. On y trouvera également d'autres resources documentaires.

La référence bibliographique concernant le package `rms` est :

> Harrell, F.E., Jr (2001). *Regression Modeling Strategies, With Applications to Linear Models, Logistic Regression, and Survival Analysis.* Springer. (600 pages)

Le cours en ligne suivant repose cet ouvrage et fournit l'essentiel des idées dans un document PDF : `http://biostat.mc.vanderbilt.edu/wiki/Main/CourseBios330`. Le livre *Clinical Prediction Models* de E.W. Steyerberg repose en partie sur le package `rms` et constitue un bon complément à l'ouvrage ci-dessus. Le site companion du livre est : `http://www.clinicalpredictionmodels.org`.

R/Markdown et reporting automatisé

Si la sauvegarde automatique des commandes R dans un fichier script R permet de « rejouer » les mêmes analyses plusieurs mois après avoir terminé un projet statistique, l'usage de commentaires dans le fichier de commandes permet de comprendre les choix de recodage des variables ou de modélisation et facilite le transfert des analyses sur un autre logiciel statistique. Il est également possible de produire directement des documents combinant des commandes R et une description textuelle des actions réalisées. L'idée est décrire l'analyse pas à pas, et d'y associer les commandes et sorties R, le tout dans un seul et même document. Les sorties R, incluant graphiques et tableaux, sont générées automatiquement lorsque le document final (format HTML ou PDF) est produit, ce qui évite d'avoir à copier-coller les résultats de la console dans le document de travail. De plus, il est possible d'exporter l'ensemble des commandes R du document, sans le texte d'accompagnement.

Ceci est possible en utilisant le langage Markdown et le package `knitr`, successeur du package `Sweave` qui implémente la même idée d'associer description et commandes R mais repose sur l'usage de LaTeXpour la production du document final. L'édition de document R/Markdown est intégrée au logiciel RStudio (`http://http://www.rstudio.com`).

Un exemple de document R/Markdown est fourni ci-après. Le fichier a été sauvegardé sous le nom `demo.rmd`. Les commandes R sont incluses dans des balises identifiées par les symboles ```` ```{r} ```` et ```` ``` ````. Le reste du document est une description textuelle, reposant sur Markdown pour le balisage des titres de niveau 1 (`\#`) ou 2 (`\#`). Un descriptif de l'ensemble des commandes de formatage est disponible à l'adresse suivante : `http://daringfireball.net/projects/markdown/syntax`.

````
# Un exemple d'analyse

On se propose de générer des données artificielles et d'estimer les
coefficients de la droite de régression pour un ensemble de 30
observations.

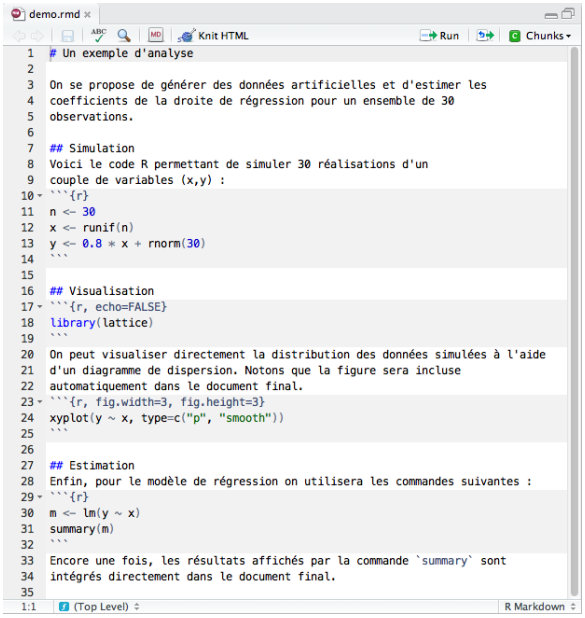
## Simulation
Voici le code R permettant de simuler 30 réalisations d'un
couple de variables (x,y) :
```{r}
n <- 30
x <- runif(n)
y <- 0.8 * x + rnorm(30)
```

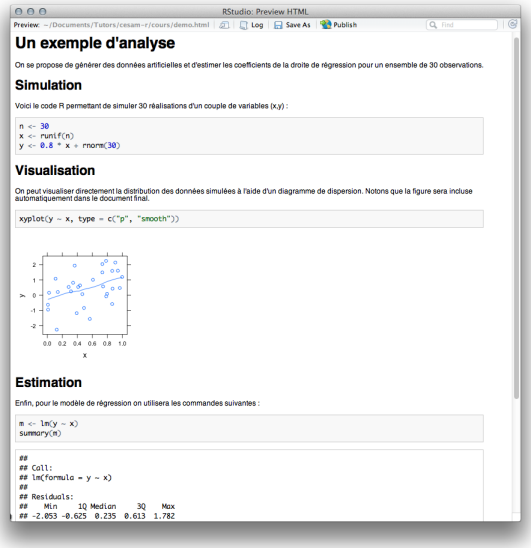
## Visualisation
```{r, echo=FALSE}
library(lattice)
```
On peut visualiser directement la distribution des données simulées à l'aide
d'un diagramme de dispersion. Notons que la figure sera incluse
automatiquement dans le document final.
```{r, fig.width=3, fig.height=3}
xyplot(y ~ x, type=c("p", "smooth"))
```

## Estimation
Enfin, pour le modèle de régression on utilisera les commandes suivantes :
````

```
```{r}
m <- lm(y ~ x)
summary(m)
```
```
Encore une fois, les résultats affichés par la commande `summary` sont
intégrés directement dans le document final.
```

Un aperçu du script ci-dessus sous RStudio est fourni ci-après. Pour créer un tel fichier, il suffit d'ouvrir le menu `File > New > R Markdown` et de recopier le texte. Ensuite, il suffit simplement de cliquer sur le bouton `Knit HTML` pour générer un document HTML reprenant l'ensemble du texte et des commandes saisies dans le fichier et les sorties produites par R. Voici un aperçu du document HTML généré par RStudio et affiché directement avec le visualisateur HTML intégré à RStudio.





On trouvera toute l'aide nécessaire sur le site de RStudio à l'adresse suivante : `http://www.rstudio.com/ide/docs/r_markdown`.
```