

# S03: Statistiques fondamentales

Analyse de données quantitatives avec R

Samuel Coavoux

- 1 Calcul vectorisé
- 2 Statistique univariée
- 3 Statistique bivariée et test d'hypothèse

# Calcul vectorisé

## Principe de la vectorisation

Une opération **vectorisée** est une opération qui est répétée sur chacun des éléments d'un vecteur sans qu'il y a besoin d'appeler explicitement une **boucle** (de type `for` ou `apply` – que nous verrons plus tard).

Dans R, la plupart des fonctions de base sont vectorisées.

Cela vaut pour quasiment toutes les fonctions **qui renvoient un vecteur de même taille que l'input**.

## Exemple: + vectorisée

Si l'on ajoute deux objets de même taille, les éléments sont ajoutés un à un ; si l'on ajoute à un objet un autre objet de taille 1 ou de taille diviseur de celle du premier, le second est recyclé (`rep()`), puis les éléments sont ajoutés un à un.

```
x <- c(1, 2, 3)
y <- c(6, 5, 4)
z <- 1
x + y
```

```
## [1] 7 7 7
```

```
x + z
```

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

## Comparer à + non vectorisé

Dans d'autres langages, on écrirait cela (qui fonctionne en R, mais qui ne doit **surtout pas** être utilisé)

```
x <- c(1, 2, 3)
y <- c(6, 5, 4)
```

```
for(i in 1:3)
  x[i] + y[i]
```

```
for(i in 1:3)
  x[i] + z
```

# Calculs

Pour calculer des scores, des moyennes, des pondérations, etc.

```
# Calcul de l'écart-type
x <- 1:10
# Carré des écarts à la moyenne
y <- (x - mean(x))^2
# Racine de la moyenne des carrés
sqrt(sum(y)/length(x))

## [1] 2.872281
```

## Simuler des données

R dispose d'un ensemble de fonctions qui génèrent la densité (d), la distribution cumulée (p), les quantiles (q), et des valeurs aléatoires (r) pour les principales distributions statistiques. Elles suivent toutes les mêmes conventions de nommage: la lettre de la fonction recherchée (d, p, q ou r) suivie du nom abrégé de la distribution. Par exemple pour la distribution normale: `dnorm()`, `pnorm()`, `qnorm()`, `rnorm()`.

Cf. `help("Distributions")` pour la liste de ces fonctions. On utilisera souvent `rnorm(x)` pour générer rapidement une variable aléatoire de taille x, de moyenne 0 et d'écart-type 1. Cf. également `runif()` (distribution uniforme), `rt()` (t de Student), `rchisq()` (loi du  $\chi^2$ ).

# Statistique univariée

## Décrire une variable quantitative

# Somme

```
x <- runif(50, min = 0, max = 10)
x

## [1] 1.2629102 5.8674708 2.9242263 4.4185836
## [5] 9.3675576 1.2073350 9.4890959 7.0973041
## [9] 4.3338250 5.6590816 6.8061206 2.4939643
## [13] 3.6600472 3.8388153 3.2522614 9.5231317
## [17] 2.6160545 0.8133658 1.6561570 7.4396236
## [21] 1.8306877 5.8065367 7.8853348 9.2452837
## [25] 2.0142680 6.0931831 9.6398638 7.0627860
## [29] 3.9085860 2.3717247 0.6284908 6.3783622
## [33] 2.5087155 4.8191233 4.1357403 8.8953838
## [37] 5.1478489 2.2834781 7.6941402 6.2207018
## [41] 9.4719685 0.6947180 2.5541115 2.7427234
## [45] 0.9459276 5.2464958 1.2805352 1.5421348
## [49] 7.1177100 2.1035000 4.5500000 5.0000000
## [53] 2.0000000 5.0000000 1.0000000 1.0000000
## [57] 1.0000000 1.0000000 1.0000000 1.0000000
## [61] 1.0000000 1.0000000 1.0000000 1.0000000
## [65] 1.0000000 1.0000000 1.0000000 1.0000000
## [69] 1.0000000 1.0000000 1.0000000 1.0000000
## [73] 1.0000000 1.0000000 1.0000000 1.0000000
## [77] 1.0000000 1.0000000 1.0000000 1.0000000
## [81] 1.0000000 1.0000000 1.0000000 1.0000000
## [85] 1.0000000 1.0000000 1.0000000 1.0000000
## [89] 1.0000000 1.0000000 1.0000000 1.0000000
## [93] 1.0000000 1.0000000 1.0000000 1.0000000
```

# Tendance centrale

# Moyenne

```
mean(x)
```

```
## [1] 4.753584
```

# enlever les 5% de valeurs les plus hautes

# et les 5% de valeurs les plus basses

```
mean(x, trim=.05)
```

```
## [1] 4.721587
```

# Médiane

```
median(x)
```

```
## [1] 4.376204
```

# Extrêmes

```
min(x)
```

```
## [1] 0.6284908
```

```
max(x)
```

```
## [1] 9.639864
```

```
# Nombre de valeurs supérieures à
# une valeur fixée
sum(x > max(x)/2)
```

```
## [1] 23
```

# Dispersion

```
var(x)
```

```
## [1] 8.087279
```

```
sd(x)
```

```
## [1] 2.843814
```

## Dispersion: quantiles

```
# Quartiles, min et max  
quantile(x)
```

```
##          0%        25%        50%        75%       100%  
## 0.6284908 2.4022846 4.3762043 7.0886746 9.6398638
```

```
# 1er et 9e déciles  
quantile(x, c(.1, .9))
```

```
##          10%        90%  
## 1.257353 9.257511
```

# Résumé

```
summary(x)
```

```
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
## 0.6285  2.4020  4.3760  4.7540  7.0890  9.6400
```

# Arrondis

```
round(x, 2)
```

```
## [1] 1.26 5.87 2.92 4.42 9.37 1.21 9.49 7.10 4.33
## [10] 5.66 6.81 2.49 3.66 3.84 3.25 9.52 2.62 0.81
## [19] 1.66 7.44 1.83 5.81 7.89 9.25 2.01 6.09 9.64
## [28] 7.06 3.91 2.37 0.63 6.38 2.51 4.82 4.14 8.90
## [37] 5.15 2.28 7.69 6.22 9.47 0.69 2.55 2.74 0.95
## [46] 5.25 1.28 1.54 7.11 7.80
```

```
round(mean(x), 2)
```

```
## [1] 4.75
```

# Intervalle de confiance autour d'une moyenne

```
tt <- t.test(x)
print(tt)
```

```
##
##  One Sample t-test
##
## data:  x
## t = 11.82, df = 49, p-value = 5.89e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  3.945381 5.561788
## sample estimates:
## mean of x
## 4.753584
```

# Intervalle de confiance autour d'une moyenne

```
str(tt)
```

```
## List of 9
## $ statistic : Named num 11.8
##   ..- attr(*, "names")= chr "t"
## $ parameter : Named num 49
##   ..- attr(*, "names")= chr "df"
## $ p.value    : num 5.89e-16
## $ conf.int   : atomic [1:2] 3.95 5.56
##   ..- attr(*, "conf.level")= num 0.95
## $ estimate   : Named num 4.75
##   ..- attr(*, "names")= chr "mean of x"
## $ null.value : Named num 0
##   ..- attr(*, "names")= chr "mean"
## $ alternative: chr "two.sided"
```

# Intervalle de confiance autour d'une moyenne

```
## Pour accéder au seul intervalle de confiance  
tt$conf.int
```

```
## [1] 3.945381 5.561788  
## attr(,"conf.level")  
## [1] 0.95
```

```
## Ou directement  
t.test(x)$conf.int
```

```
## [1] 3.945381 5.561788  
## attr(,"conf.level")  
## [1] 0.95
```

## Intervalle de confiance autour d'une moyenne

Il est possible d'ajuster le niveau de confiance souhaité. Par défaut, on fixe  $\alpha$  à 0.05 avec l'argument `conf.level = 0.95` ; il est possible de réduire  $\alpha$  en augmentant `conf.level`.

```
t.test(x, conf.level = .99)$conf.int
```

```
## [1] 3.675772 5.831397
## attr(,"conf.level")
## [1] 0.99
```

# Représentation graphique

Il existe plusieurs systèmes de représentations graphiques dans R.

- R-base : package `graphics`, chargé dans toute nouvelle session;
- R-base : package `lattice`, qui n'est pas chargé par défaut, plus complexe et plus puissant;
- À installer: package `ggplot2` (*Grammar of graphics*).

Pour le moment, nous faisons des graphiques **exploratoires**: avec `graphics` (plus simple), sans ajouter de titres, labels, couleurs, etc. Nous verrons plus tard comment faire des graphiques pour la publication.

# Une fonction générique, `plot()`

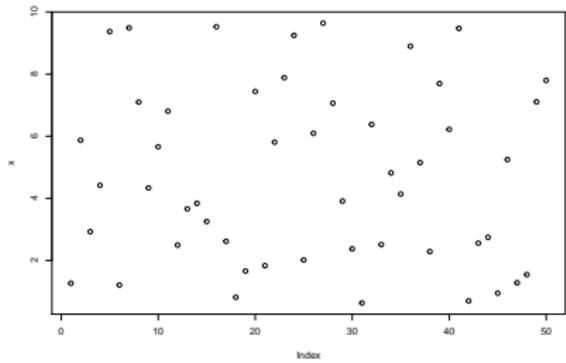
`plot()` est une fonction générique, un “wrapper”: elle évalue les arguments qui lui sont donnés, et appelle d’autres fonctions, selon la valeur de ces arguments. Elle permet d’aller vite, sans avoir besoin de connaître toutes les fonctions graphiques, pour faire de l’exploration de données.

`plot`

```
## function (x, y, ...)
## UseMethod("plot")
## <bytecode: 0xe731f8>
## <environment: namespace:graphics>
```

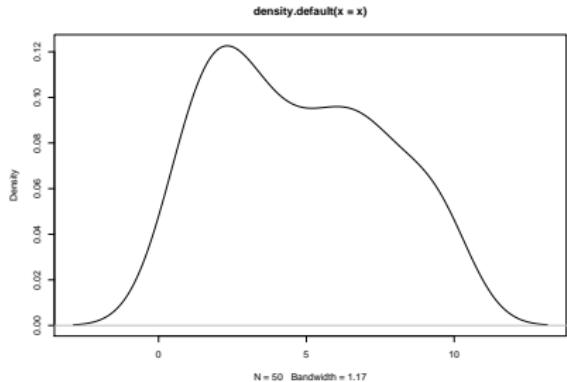
# Plot

`plot(x)`



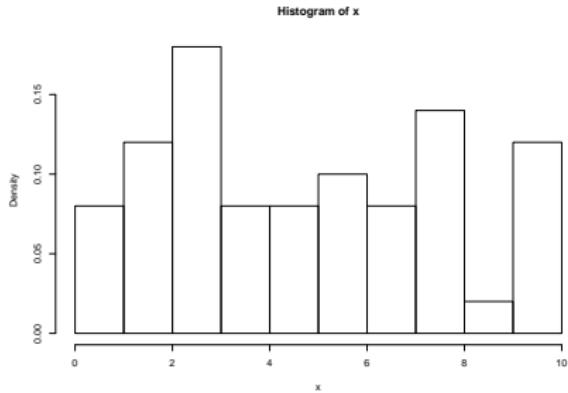
# Densité

```
plot(density(x))
```



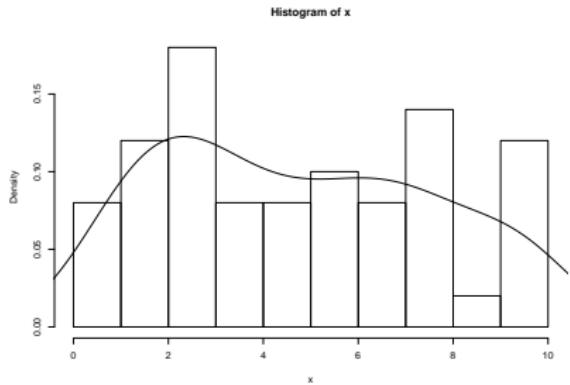
# Histogramme

```
hist(x, freq = FALSE)
```



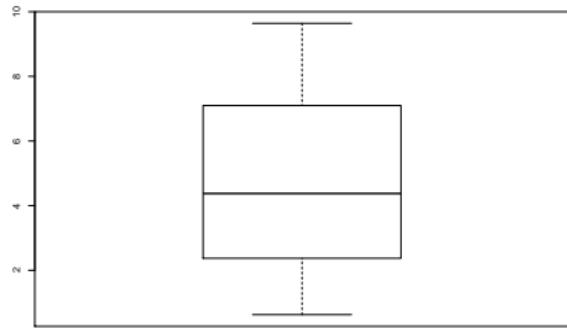
# Histogramme + densité

```
hist(x, freq = FALSE)  
lines(density(x))
```



# Boxplot

```
boxplot(x)
```



## Gestion des valeurs manquantes

La grande majorité de ces fonctions renvoient NA s'il y a une valeur manquante dans le vecteur. On peut alors préciser `na.rm = TRUE` pour que les valeurs manquantes soient supprimées avant le calcul.

## Décrire une variable catégorielle

# Données: Artist Community Survey

```
load("./data/ACS_artists.Rdata")
```

# Tri à plat

```
table(dt$sexe)  
  
##  
## Female    Male  
## 61185    62838
```

# Tri à plat

```
table(dt$sexe)/nrow(dt)
```

```
##  
##      Female      Male  
## 0.4933359 0.5066641
```

```
prop.table(table(dt$sexe))
```

```
##  
##      Female      Male  
## 0.4933359 0.5066641
```

# Tri à plat

```
library(questionr)
freq(dt$sexe)
```

```
##           n   % val%
## Female  61185 49.3 49.3
## Male    62838 50.7 50.7
## NA      0   0.0   NA
```

## Arguments de freq()

- digits: nombre de chiffres après la virgule (1 par défaut)
- cum: calculer les pourcentages cumulés (FALSE par défaut)
- sort: trier les modalités ("inc": dans l'ordre croissant, "dec" : dans l'ordre décroissant ; défaut: ordre normal des modalités)

```
freq(dt$eng, cum = TRUE)
```

		n	%	val%	%cum	val%cum
##	(1) Very well	12857	10.4	71.3	10.4	71.3
##	(2) Well	3401	2.7	18.9	13.1	90.2
##	(3) Not well	1453	1.2	8.1	14.3	98.2
##	(4) Not at all	319	0.3	1.8	14.5	100.0
##	NA	105993	85.5	NA	100.0	NA

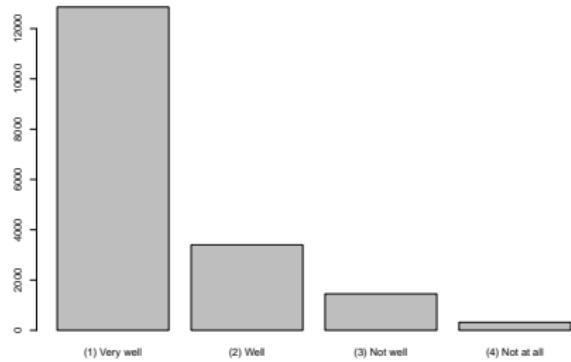
# Test de proportion

```
pt <- prop.test(x = sum(dt$sex == "Female") ,  
                 n = nrow(dt) ,  
                 conf.level = .99)  
  
pt$conf.int
```

```
## [1] 0.4896756 0.4969970  
## attr(,"conf.level")  
## [1] 0.99
```

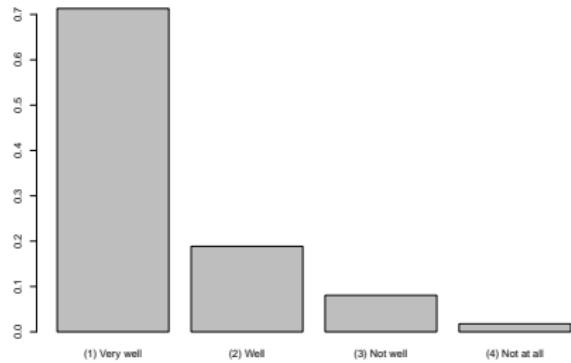
# Diagramme en barre – effectif

```
barplot(table(dt$eng))
```



# Diagramme en barre – fréquence

```
barplot(prop.table(  
  table(dt$eng)))
```

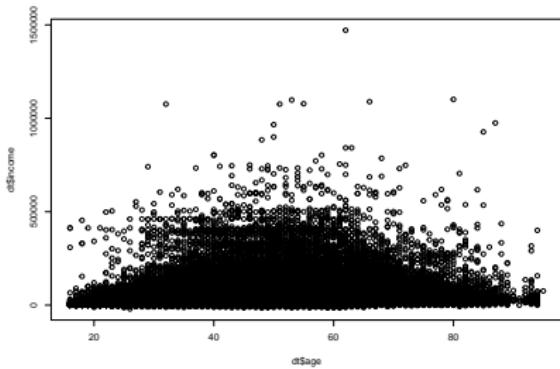


## Statistique bivariée et test d'hypothèse

## Deux variables quantitatives

# Scatterplot

```
plot(dt$age, dt$income)
```



# Analyse de corrélation

```
var(dt$age)
```

```
## [1] 229.5101
```

```
var(dt$income)
```

```
## [1] 3326444615
```

```
cov(dt$age, dt$income)
```

```
## [1] 128716.8
```

```
cor(dt$age, dt$income)
```

```
## [1] 0.1473141
```

## Deux variables qualitatives

## Tableau croisé

```
tb <- table(dt$dipl_c, dt$sex) tb
```

```

##                                     Female   Male
##    Aucun                      2132  2244
##    HS degree                  18624 20223
##    College                    31177 30324
##    Graduate ed.              9252 10047

```

## Tableau croisé: structure

```
str(tb)
```

```
##  'table' int [1:4, 1:2] 2132 18624 31177 9252 2244 20223
##  - attr(*, "dimnames")=List of 2
##    ..$ : chr [1:4] "Aucun" "HS degree" "College" "Graduat
##    ..$ : chr [1:2] "Female" "Male"
```

## Tableau croisé: marges

```
addmargins(tb)
```

```
##  
##  
##          Female   Male    Sum  
## Aucun        2132    2244   4376  
## HS degree    18624   20223  38847  
## College      31177   30324  61501  
## Graduate ed. 9252    10047  19299  
## Sum          61185   62838 124023
```

## Tableau croisé: fréquences

```
# Fréquences du total;
# Ajouter margin = 1 pour % ligne
# Ajouter margin = 2 pour % colonne
prop.table(tb)
```

```

##                                     Female      Male
## Aucun          0.01719036 0.01809342
## HS degree     0.15016570 0.16305846
## College       0.25138079 0.24450304
## Graduate ed. 0.07459907 0.08100917

```

## Tableau croisé: fréquences ligne

```
library(questionr)
lprop(tb)
```

```
##
##                                     Female  Male   Total
##    Aucun                  48.7  51.3 100.0
##    HS degree              47.9  52.1 100.0
##    College                50.7  49.3 100.0
##    Graduate ed.           47.9  52.1 100.0
##    Ensemble               49.3  50.7 100.0
```

## Tableau croisé: fréquences colonne

cprop(tb)

```

##                                     Female  Male Ensemble
##    Aucun                  3.5     3.6   3.5
##    HS degree              30.4    32.2  31.3
##    College                51.0    48.3  49.6
##    Graduate ed.           15.1    16.0  15.6
##    Total                  100.0   100.0 100.0

```

# Test du $\chi^2$

```
ct <- chisq.test(tb)
ct

##
##  Pearson's Chi-squared test
##
## data: tb
## X-squared = 91.248, df = 3, p-value <
## 2.2e-16
```

## Test du $\chi^2$ : structure

```
str(ct)
```

```
## List of 9
## $ statistic: Named num 91.2
##   ..- attr(*, "names")= chr "X-squared"
## $ parameter: Named int 3
##   ..- attr(*, "names")= chr "df"
## $ p.value : num 1.18e-19
## $ method  : chr "Pearson's Chi-squared test"
## $ data.name: chr "tb"
## $ observed : 'table' int [1:4, 1:2] 2132 18624 31177 92
##   ..- attr(*, "dimnames")=List of 2
##     ..$ : chr [1:4] "Aucun" "HS degree" "College" "Grad"
##     ..$ : chr [1:2] "Female" "Male"
## $ expected : num [1:4, 1:2] 2159 19165 30341 9521 2217
```

# Test du $\chi^2$ : effectifs théoriques

```
ct$expected
```

```
##                                     Female      Male
## Aucun           2158.838  2217.162
## HS degree     19164.620 19682.380
## College        30340.652 31160.348
## Graduate ed.  9520.890  9778.110
```

# Test du $\chi^2$ : p value

```
ct$p.value
```

```
## [1] 1.181377e-19
```

## Test de Fisher

```
# On produit un tableau croisé imaginaire avec
# de très petits effectifs
tb <- as.table(matrix(c(10, 9, 4, 1), nrow = 2))
ft <- fisher.test(tb)
ft

##
## Fisher's Exact Test for Count Data
##
## data: tb
## p-value = 0.3577
## alternative hypothesis: true odds ratio is not equal to
## 95 percent confidence interval:
## 0.005091836 3.718776118
## sample estimates:
##      . . .
```

## Test de Fisher

```
str(ft)
```

```
## List of 7
## $ p.value      : num 0.358
## $ conf.int     : atomic [1:2] 0.00509 3.71878
##   ..- attr(*, "conf.level")= num 0.95
## $ estimate     : Named num 0.292
##   ..- attr(*, "names")= chr "odds ratio"
## $ null.value   : Named num 1
##   ..- attr(*, "names")= chr "odds ratio"
## $ alternative: chr "two.sided"
## $ method       : chr "Fisher's Exact Test for Count Data"
## $ data.name    : chr "tb"
## - attr(*, "class")= chr "htest"
```

```
ft$p.value
```

## Une variable quantitative, une variable qualitative

## tapply()

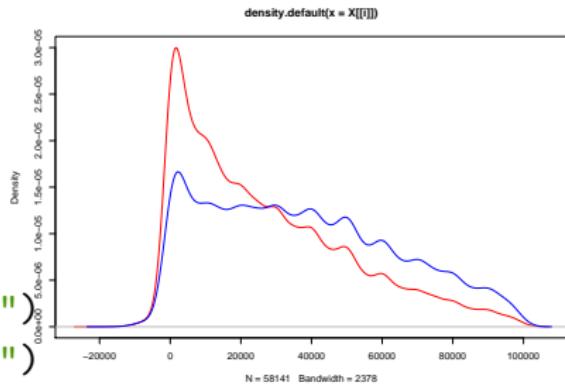
La fonction `tapply()` permet d'appliquer une fonction à un vecteur en le découplant selon les modalités d'un deuxième vecteur. Par exemple, si l'on souhaite appliquer la fonction `mean()` au revenus des artistes en les différenciant par niveau d'éducation, on peut écrire:

```
# Les arguments sont dans le bon ordre
# Vous pouvez omettre les noms d'arguments
# qui sont rappelés parce que leur ordre
# n'est pas intuitif
tapply(X = dt$income, INDEX = dt$dipl_c, FUN = mean)
```

```
##          Aucun      HS degree       College
##    19307.13    31941.02    49261.19
## Graduate ed.
##    62959.98
```

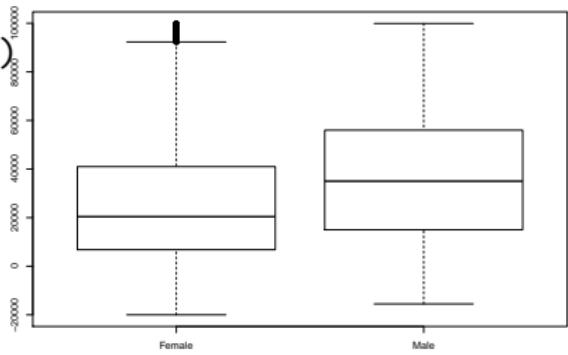
# Représenter deux densités

```
dt <- dt[dt$income <  
         100000, ]  
ids <- tapply(dt$income,  
              dt$sexe,  
              density)  
plot(ids$Female, col = "red")  
lines(ids$Male, col = "blue")
```



# Boxplot multiple

```
boxplot(dt$income ~ dt$sexe)
```



## Test de student

```
tt <- t.test(dt$income ~ dt$sex)
tt

##
##  Welch Two Sample t-test
##
## data:  dt$income by dt$sex
## t = -69.87, df = 109450, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not
## 95 percent confidence interval:
## -10745.96 -10159.52
## sample estimates:
## mean in group Female    mean in group Male
##                      26677.38                      37130.12
```

## Test de student

On remarque que, par défaut, `t.test()` fait un test de Welch. On peut forcer un test de Student en ajoutant `var.equal = TRUE` (à condition évidemment que les variances soient bien égales).

```
tapply(dt$income, dt$sex, var)
```

```
##      Female      Male
## 561779863 692542839
```

```
# Elles ne le sont pas...
# t.test(dt$income ~ dt$sex, var.equal=TRUE)
```

## Test de student

Par défaut, le test est bilatéral. On peut réaliser un test unilatéral avec alternative ("less" pour un test unilatéral à gauche, "greater" pour un test unilatéral à droite)

```
t.test(dt$income ~ dt$sex, alternative = "less")
```

```
##  
## Welch Two Sample t-test  
##  
## data: dt$income by dt$sex  
## t = -69.87, df = 109450, p-value < 2.2e-16  
## alternative hypothesis: true difference in means is less than 0  
## 95 percent confidence interval:  
##       -Inf -10206.67  
## sample estimates:  
## mean in group Female   mean in group Male
```

# Test de student

```
tt$p.value
```

```
## [1] 0
```

## Test de student

On peut enfin comparer une moyenne à une moyenne théorique  $\mu$ .

```
t.test(dt$income, mu=45000)
```

```
##  
##  One Sample t-test  
##  
## data: dt$income  
## t = -174.31, df = 112590, p-value < 2.2e-16  
## alternative hypothesis: true mean is not equal to 45000  
## 95 percent confidence interval:  
## 31583.33 31881.70  
## sample estimates:  
## mean of x  
## 31732.52
```