Atelier R - 24 juillet 2020

Document préparé par Miceline Mésidor

PLAN

- Introduction
- Interface de R-Studio
- Fonctionnement de R-Studio
- Sommaire / Visualisation
- Tests statistiques
- Régression linéaire et corrélation
- Régression logistique

Introduction

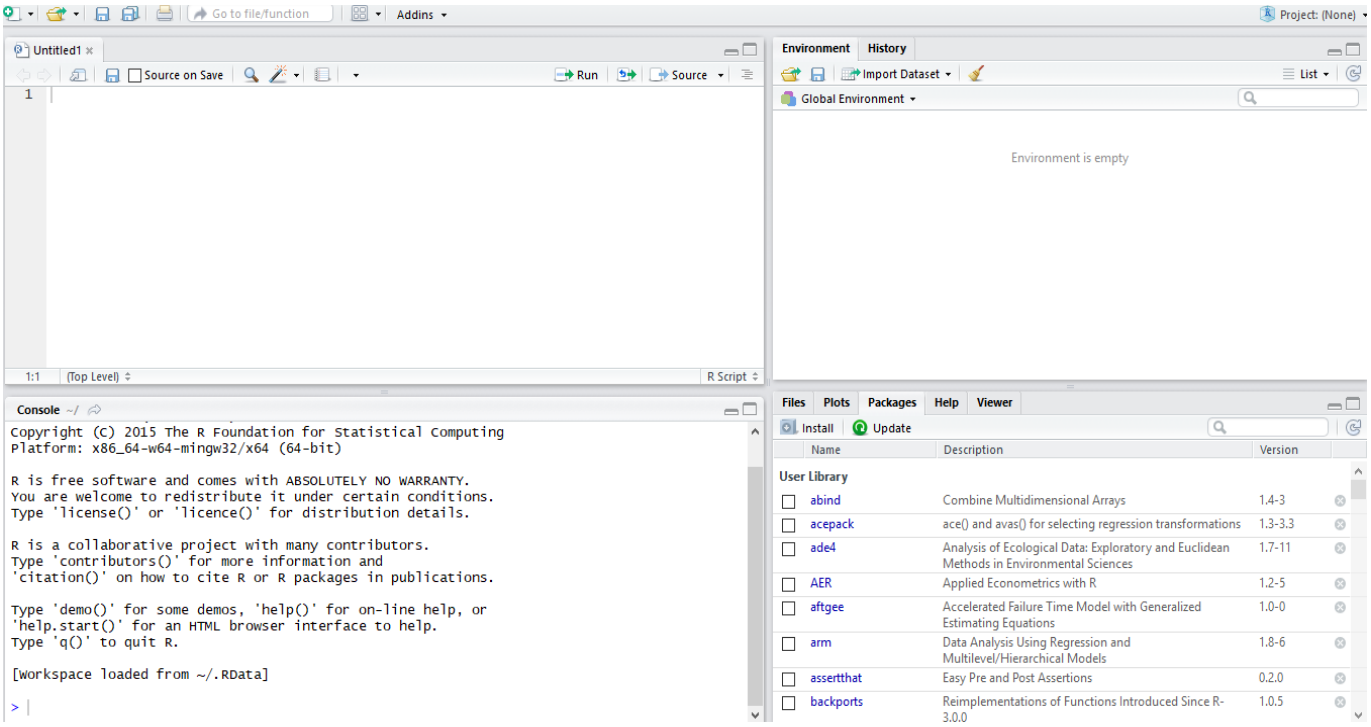
R : C'est quoi?

- Logiciel de statistique libre → codes le constituant accessibles et réutilisables
- Langage de programme complet
- Gratuit comparativement à la plupart des autres logiciels statistiques (SAS, Stata) qui sont payants
- R-Studio : environnement intégré de développement, sert d'interface entre R et l'utilisateur

1- INTERFACE DE R-STUDIO

Utilisation de R-Studio

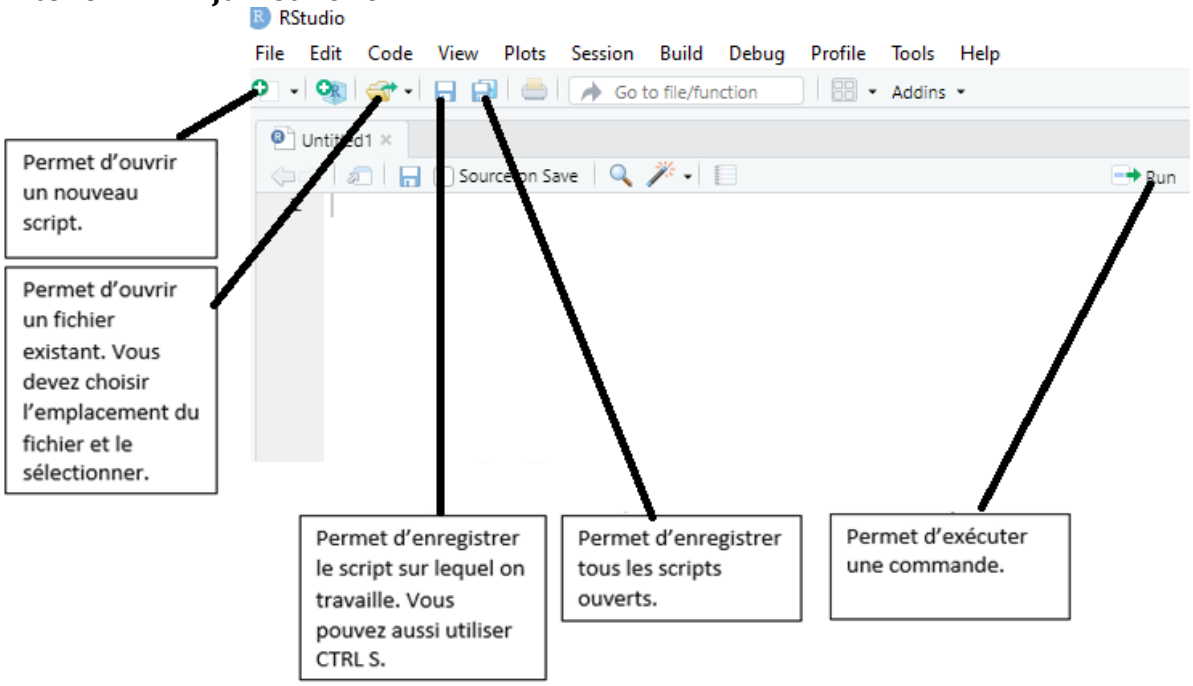
Pour utiliser R-Studio, on peut utiliser l'icône sur le bureau ou le menu démarrer.
L'interface de R-Studio se présente de la façon suivante :



Vous pouvez remarquer que l'interface de R-Studio est divisée en quatre parties.

Éditeur

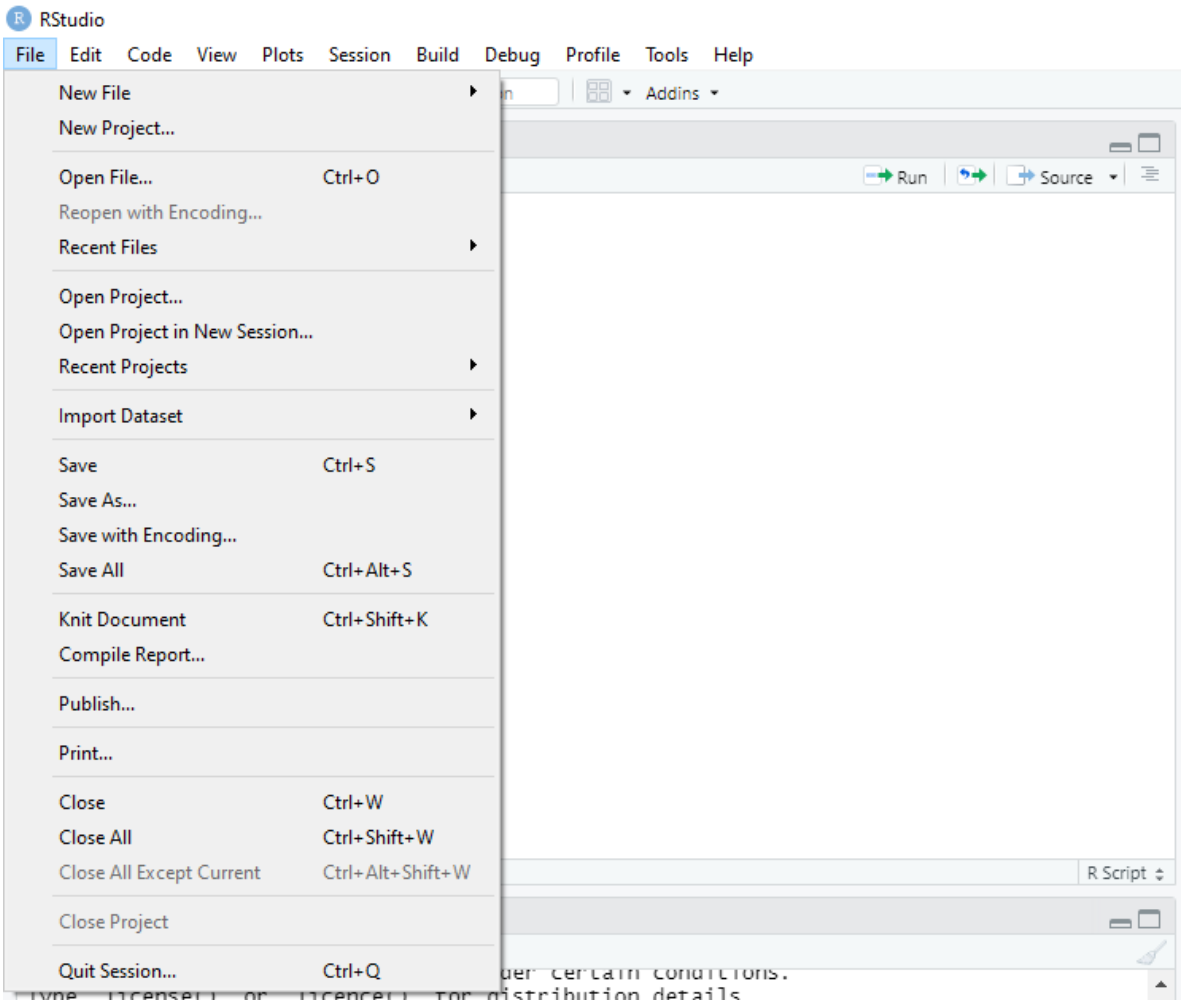
La première fenêtre dans la page d'accueil de R-Studio s'appelle l'éditeur et il sert à écrire les différentes commandes.



Les fichiers script de l'éditeur ont comme extension '.R'. Par exemple, Atelier.R.

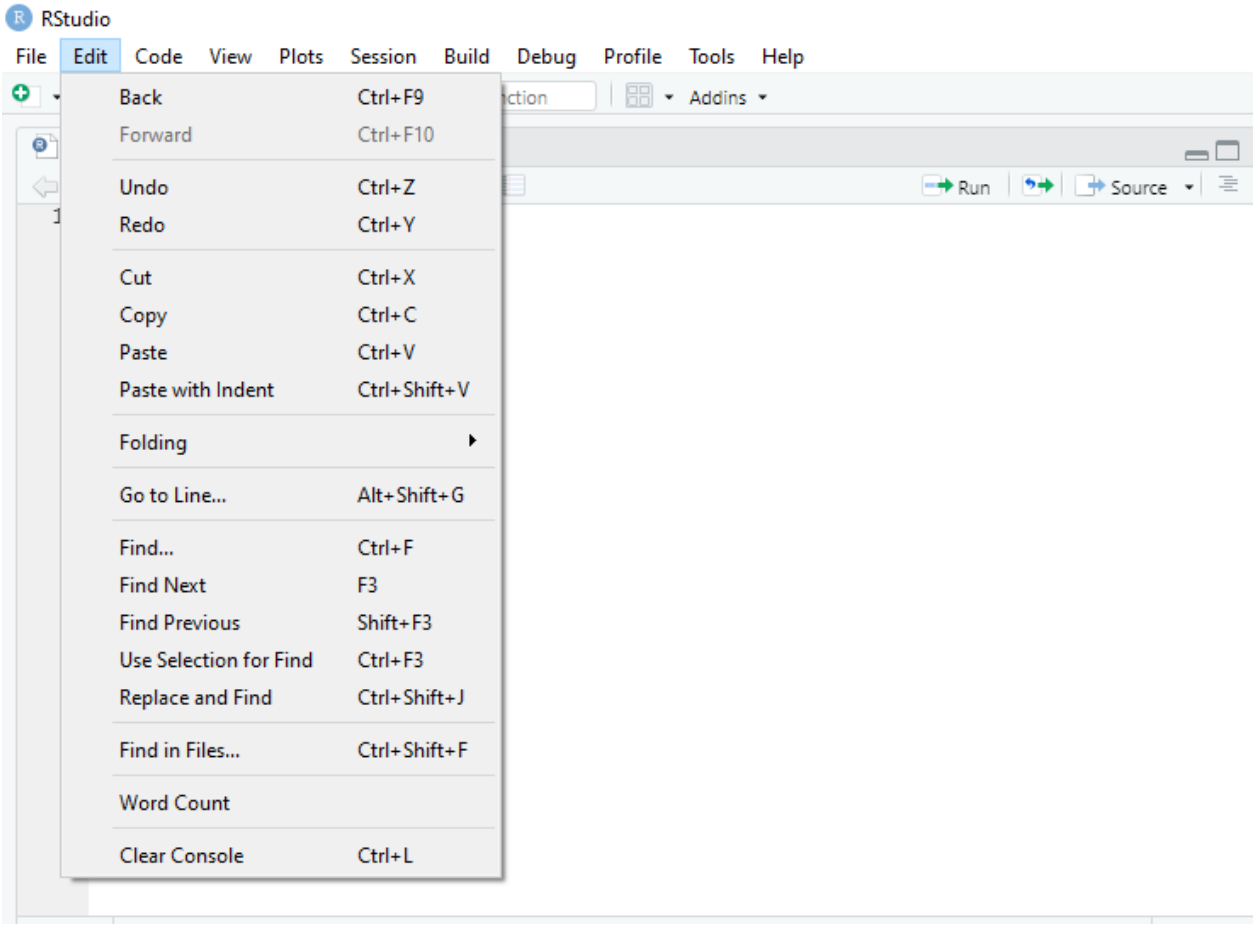
Passons rapidement en revue quelques éléments de l'éditeur.

File : Contient plusieurs commandes permettant d'ouvrir un fichier (notamment un script), de voir les fichiers les plus récents.

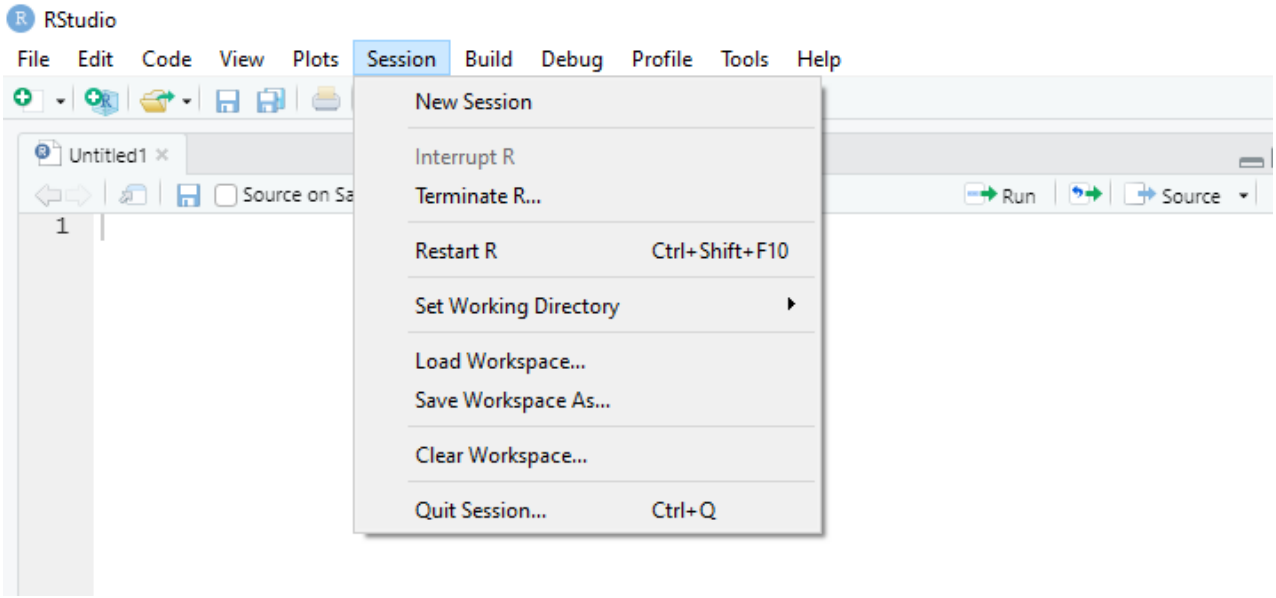


Il convient de souligner que R-studio sauvegarde automatiquement le script créé et celui-ci s'ouvre à chaque fois qu'on relance R-Studio.

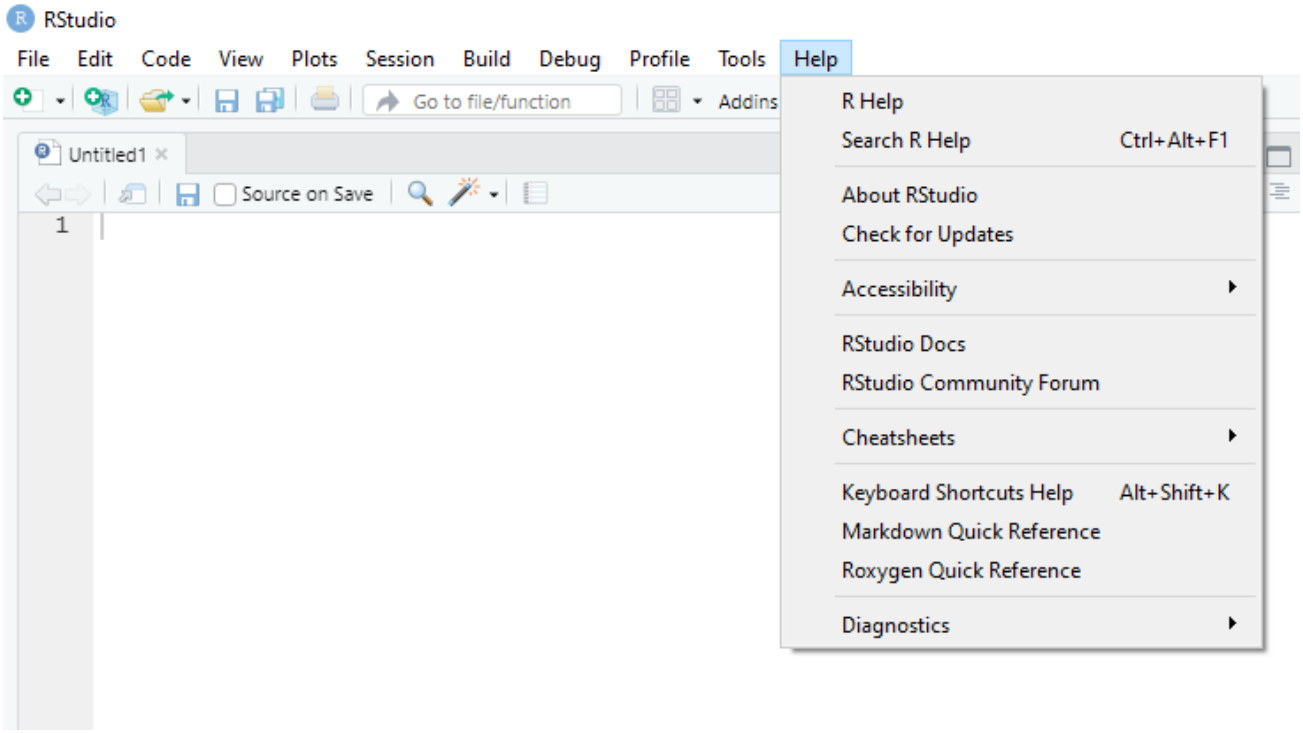
Edit : permet par exemple d'annuler, de couper, de coller des commandes.



Session : Permet d'ouvrir une nouvelle session R ou de terminer la session sur laquelle on travaille.

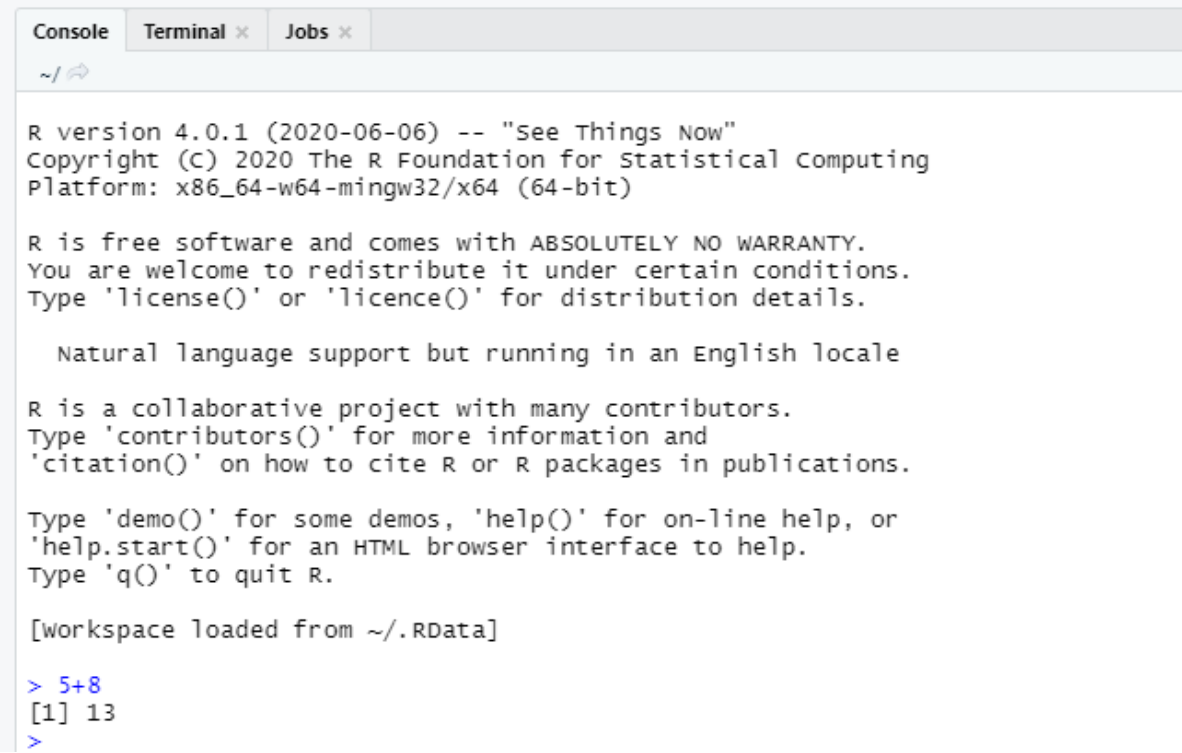


Help : Permet d'avoir de la documentation sur R-Studio et sur certaines commandes.



Console

La fenêtre, sous l'éditeur, s'appelle la console et permet d'afficher les résultats des commandes. On peut aussi écrire les commandes dans la console.



Pour passer de l'éditeur à la console, il suffit de sélectionner les commandes voulues dans l'éditeur et de les copier (Edit-Copy) et de les coller dans la console (Edit-Paste).

Opérations sur R-Studio

Avant d'être un logiciel de programmation, R-Studio permet de faire des calculs. Ainsi vous pouvez effectuer des opérations de toute sorte sur R-Studio, en les écrivant soit dans l'éditeur soit dans la console.

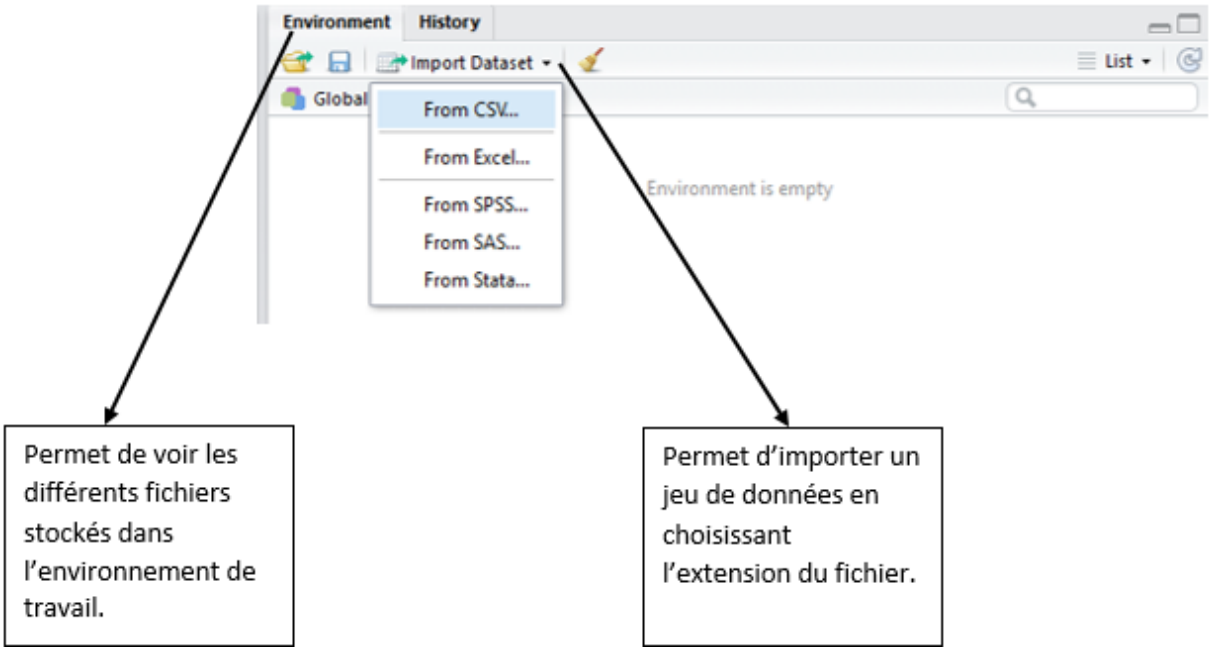
```
5+8

## [1] 13

8-2

## [1] 6

6/3

## [1] 2
```

Vous pouvez aussi faire des calculs en utilisant des fonctions déjà enregistrées sur R-Studio. Ces dernières sont des portions de code qui sont exécutées lorsqu'on les appelle.

Dans le cadre de cet exemple, la fonction `round` a été utilisée pour arrondir à deux décimales près. La fonction `sqrt` permet d'obtenir la racine carrée et la fonction `sum` permet de calculer la somme.

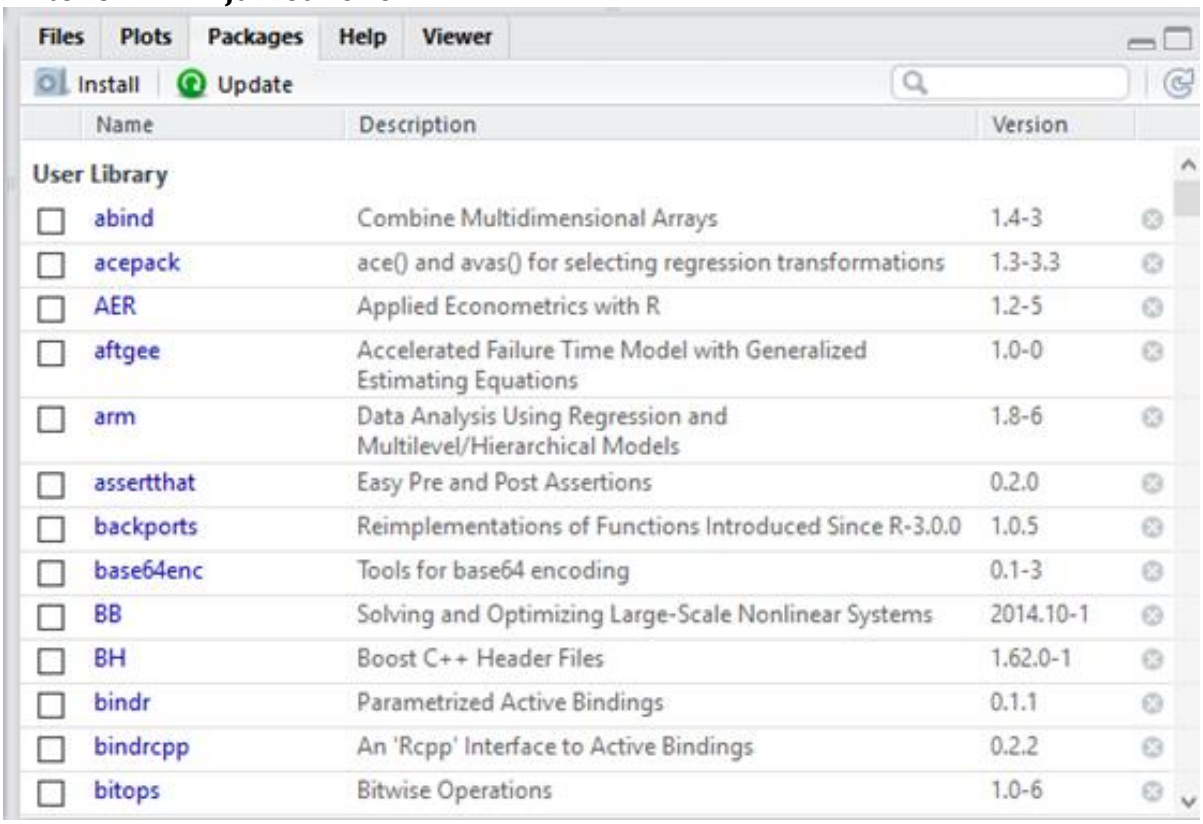
```
round(6.255553,2)

## [1] 6.26

sqrt(16)

## [1] 4

sum(3+7+8)

## [1] 18
```

Environment / History



Bibliothèques

Certaines commandes de R-Studio, pour être exécutés demandent l'utilisation d'outils supplémentaires appelés « bibliothèques (français) ; packages (anglais) ». Généralement, on installe le nom de la bibliothèque qu'on veut utiliser, en cliquant sur le bouton « Install » ; et ensuite on la lance, en cliquant dans la case tout près du nom de la bibliothèque.

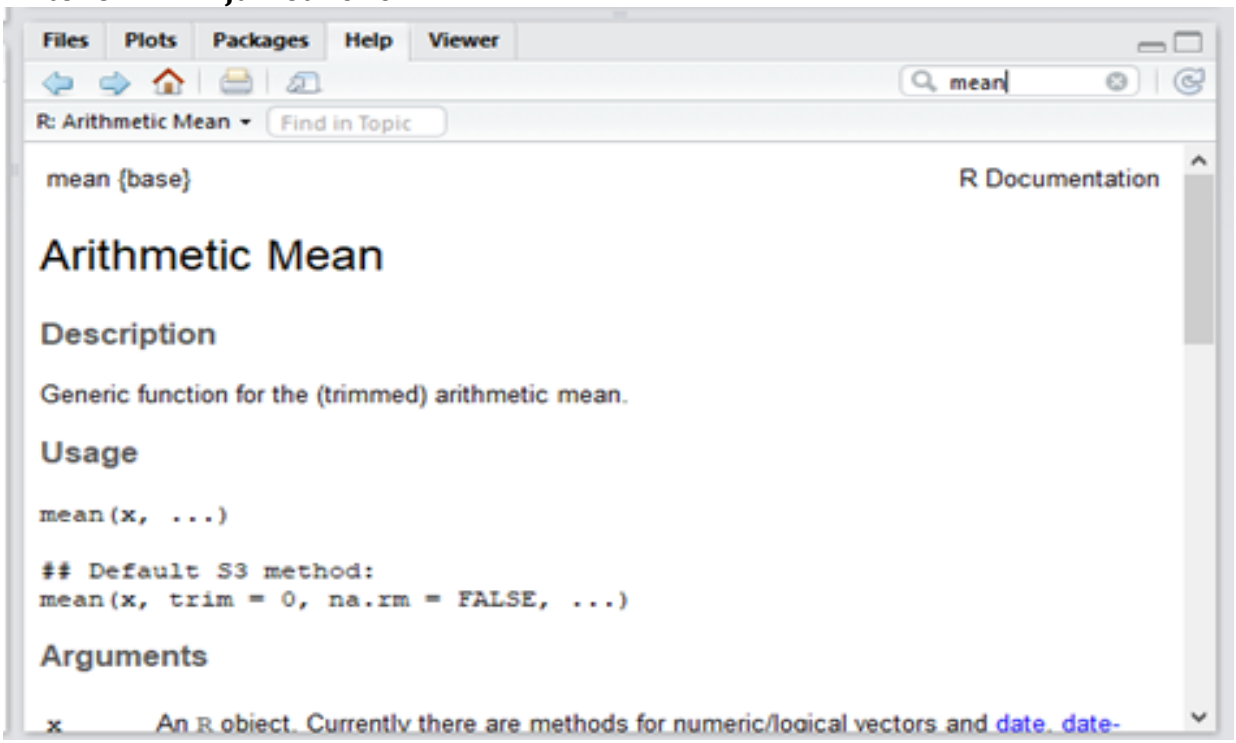


Aide

Chaque fonction dans R-Studio est décrite dans un fichier d'aide. Pour le faire apparaître, tapez dans la console un point d'interrogation, suivi de la fonction. Par exemple, la fonction `mean` calcule la moyenne. Pour obtenir de l'aide à son sujet :

```
?mean

## starting httpd help server ... done
```

Une fenêtre d'aide apparaitra. Vous pouvez aussi utiliser la commande `Help` dans la fenêtre à droite de la console. Ici, R-Studio donne des informations sur la commande « mean ».



Notez que la structure du fichier d'aide est toujours la même, avec des exemples à la fin.

Pour quitter R-studio, on peut soit taper q() ou cliquer sur le X en haut à droite comme pour n'importe quel programme.

Conseils

- R est sensible à la casse; donc il faut faire attention à l'orthographe des noms utilisés et des fonctions. Par exemple, `mean` et `Mean` ne sont pas équivalents.
- Pour écrire des commentaires dans le script, il faut que le caractère # précède le texte. Ainsi le texte ne sera pas interprété par R.

```
# Atelier R

Atelier R

Atelier R

Error: <text>:2:7: unexpected symbol
## 1:
## 2: Atelier R
##
```

- Même s'ils sont équivalents, privilégiez l'utilisation des guillemets doubles (") aux guillemets simples (') pour ne pas que R-Studio confonde avec l'utilisation de l'apostrophe en français.

2- FONCTIONNEMENT DE R-STUDIO

Lire et sauvegarder des données

Fichier csv

R-Studio peut lire des données provenant de plusieurs sources. Pour importer des données, on peut utiliser le bouton `Import dataset` qui se trouve dans la fenêtre `Environment` et on choisit le format du jeu de données. On peut aussi utiliser la commande `read.table [read.csv ou read.csv2 (si l'ordinateur est en français)]`. Il est plus facile et prudent d'importer et d'exporter les données sous format csv. Pour importer un fichier « csv », on écrit la commande suivante :

```
donnees <- read.table("C:/Users/Mice/OneDrive/DOCTORAT/2017-2018/TRIMESTRE 6/MSO6002/donnees.csv", header=T, sep=",")
```

Notez l'utilisation de :

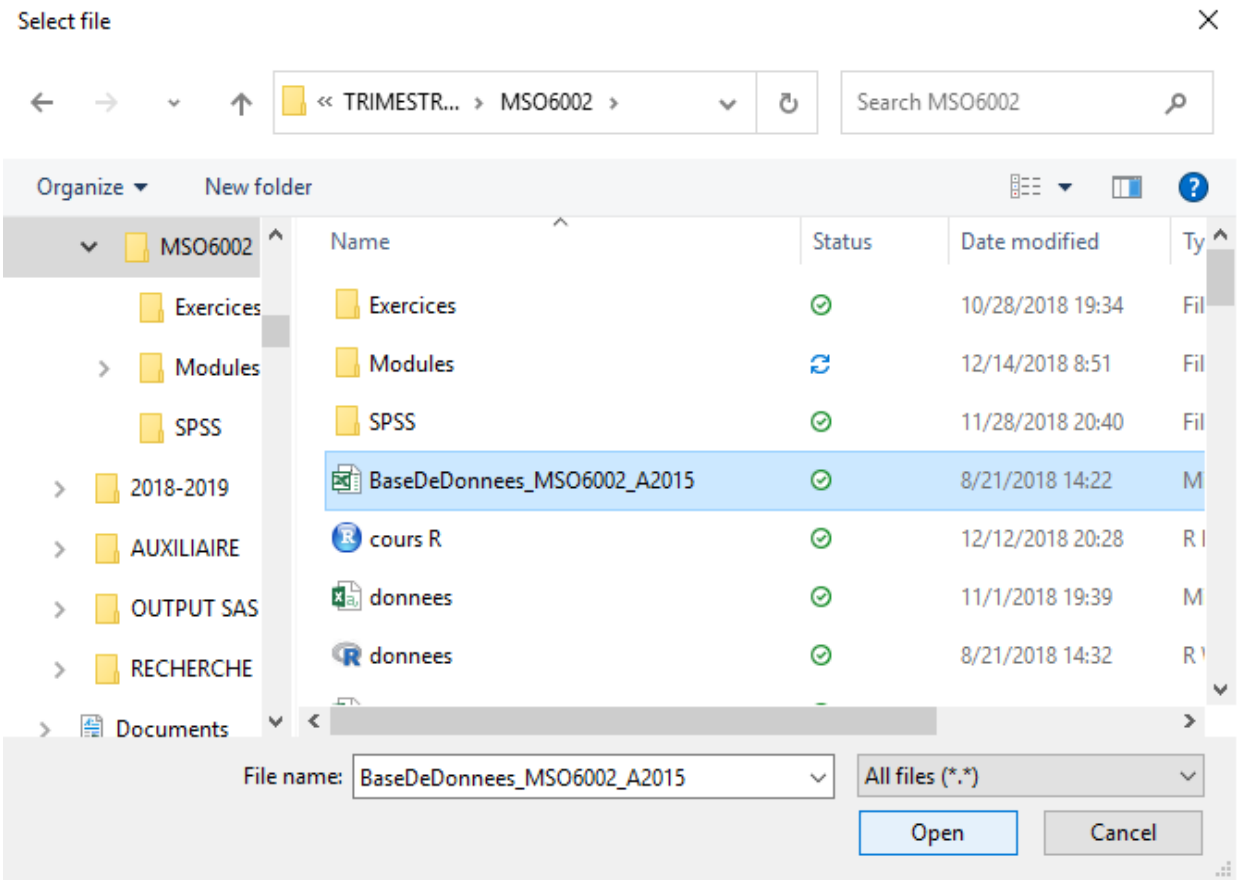
- <- pour définir un nouvel objet portant le nom de « donnees ».
- header = T, pour dire à R que la première ligne contient le nom des variables.
- sep = « , », précise que le séparateur de caractères est la virgule.

Pour exporter un jeu de données en fichier « csv », on procède ainsi :

```
write.table(donnees, file="donnees2.csv", sep=",")
```

Vous pouvez aussi choisir manuellement le chemin du fichier en utilisant la commande `file.choose`. La fenêtre ci-après apparait, vous devez choisir le fichier que vous désirez ouvrir puis cliquez sur Open.

```
file.choose()
```



Vous verrez que le chemin apparaitra dans la console, ensuite il vous suffit de copier le chemin dans le script. Cette astuce permet d'éviter les erreurs de chemin du fichier.
Attention : Il faut penser à modifier le sens des flèches par la suite.

```
> file.choose()
[1] "C:\\Users\\Choue\\OneDrive\\DOCTORAT\\2017-2018\\TRIMESTRE 6\\MSO6002\\
BaseDeDonnees_MSO6002_A2015.xls"
```

Changer le répertoire par défaut

R-Studio utilise un répertoire par défaut défini lors de l'installation, plus précisément tous les codes et toutes les bases de données seront sauvegardés et téléchargés à cet endroit. Pour savoir le répertoire par défaut, il faut écrire :

```
getwd()

## [1] "C:/Users/Mice/OneDrive/DOCTORAT/2017-2018/TRIMESTRE 6/MSO6002/Modules
"
```

Du coup, si on ne fournit pas de chemin lors de l'utilisation de load ou save, R-Studio prendra pour acquis que le fichier se trouve dans le répertoire par défaut. Pour modifier le répertoire, utiliser la fonction setwd.

```
setwd("C:/Users/Mice/OneDrive/DOCTORAT/2017-2018/TRIMESTRE 6/MSO6002")
```

Ainsi, écrire read.table("C:/Users/Choue/OneDrive/DOCTORAT/2017-2018/TRIMESTRE 6/MSO6002/donnees.csv", header=T, sep=",") ou read.table("donnees.csv", header=T, sep=",") seront maintenant équivalents.

ATTENTION! Cette modification n'a cours que lors de la session active. Dès qu'on ferme R-Studio et que l'ouvre à nouveau, on change de session.

Présentation rapide de l'objet

Tout ce qu'on manipule sur R est un objet. Parmi ceux-ci, on compte les vecteurs, les matrices, les data.frames, les listes etc.

Par exemple, créons un vecteur et un data.frame :

```
a <- c(1,2,3)
b <- data.frame(ID=c(1,2,3), AGE=c(20,30,40))
a

## [1] 1 2 3

b

##   ID AGE
## 1  1  20
## 2  2  30
## 3  3  40
```

On peut ensuite interroger R quant au type d'objet que 'a' ou 'b' est :

```
class(a)

## [1] "numeric"

class(b)

## [1] "data.frame"
```

Chaque objet a le nom que l'utilisateur lui donne, comme 'a' ou 'b'. Par exemple, « somme » est un objet auquel est assigné le résultat de la commande « sum(7+9+8) ».

```
somme <- sum(7+9+8)
somme

## [1] 24
```

Quelques commandes de gestion de data.frame

Nous avons vu précédemment les processus permettant de lire un objet et d'importer un jeu de données. Maintenant, nous allons voir de façon détaillée ce que contient un `data.frame`.

Les premières valeurs sont affichées à côté de chaque variable. Vous pouvez aussi utiliser la commande `head` pour voir les premières valeurs des variables:

```
head(donnees)
```

```
##   Id sex tx affection FexEl1  FexN1 FexEl2  FExN2 VAS1 VAS2 Grossesse
## 1  1   1  1         3  6.524  5.714  6.379  6.043   23   62         0
## 2  2   1  1         3  7.935  8.288  7.996  9.024   38   13         0
## 3  3   2  1         3 17.168 16.454 16.249 19.701    8    8         0
## 4  4   2  1         1 19.033 16.941 24.241 18.576   48   19         0
## 5  5   2  1         1 19.428 18.913 20.756 17.669   40   26         0
## 6  6   2  1         1 17.811 17.910 20.517 19.933   18    8         0
```

De même, vous pouvez aussi utiliser la commande `dim` pour savoir le nombre de lignes et de colonnes.

```
dim(donnees)
```

```
## [1] 60 11
```

Pour connaitre la structure de l'objet « donnees », il faut utiliser la commande `str`:

```
str(donnees)
```

```
## 'data.frame':    60 obs. of  11 variables:
##  $ Id       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ sex      : int  1 1 2 2 2 2 2 2 1 1 ...
##  $ tx       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ affection: int  3 3 3 1 1 1 1 1 1 2 ...
##  $ FexEl1   : num  6.52 7.93 17.17 19.03 19.43 ...
##  $ FexN1    : num  5.71 8.29 16.45 16.94 18.91 ...
##  $ FexEl2   : num  6.38 8 16.25 24.24 20.76 ...
##  $ FExN2    : num  6.04 9.02 19.7 18.58 17.67 ...
##  $ VAS1     : int  23 38 8 48 40 18 3 39 95 81 ...
##  $ VAS2     : num  62 13 8 19 26 8 3.5 11 29 38 ...
##  $ Grossesse: int  0 0 0 0 0 0 0 0 0 0 ...
```

3- SOMMAIRE/VISUALISATION

Sommaire des variables

La commande « summary » produit un résumé statistique des différentes variables.

```
summary(donnees)
```

```
##        Id           sex           tx          affection       FexEl1      
##  Min.   : 1.00   Femme:32   Min.   :1.000   Min.   :1.0   Min.   : 4.229  
##  1st Qu.:15.75   Homme:28   1st Qu.:1.000   1st Qu.:1.0   1st Qu.: 7.949  
##  Median :30.50              Median :1.000   Median :1.0   Median :12.290  
##  Mean   :30.50              Mean   :1.467   Mean   :1.6   Mean   :12.342  
##  3rd Qu.:45.25              3rd Qu.:2.000   3rd Qu.:2.0   3rd Qu.:17.228  
##  Max.   :60.00              Max.   :2.000   Max.   :3.0   Max.   :19.428  
##      FexN1            FexEl2           FExN2              VAS1       
##  Min.   : 4.735   Min.   : 2.722   Min.   :  4.624   Min.   : 3.00  
##  1st Qu.: 8.202   1st Qu.: 8.624   1st Qu.:  9.024   1st Qu.:23.00  
##  Median :11.833   Median :11.393   Median : 11.661   Median :39.50  
##  Mean   :12.311   Mean   :12.997   Mean   : 29.615   Mean   :41.72  
##  3rd Qu.:16.941   3rd Qu.:16.330   3rd Qu.: 18.857   3rd Qu.:64.00  
##  Max.   :22.116   Max.   :24.241   Max.   :995.000   Max.   :95.00  
##       VAS2       
##  Min.   : 0.00  
##  1st Qu.: 8.00  
##  Median :18.00  
##  Mean   :19.12  
##  3rd Qu.:26.00  
##  Max.   :62.00
```

Pour présenter un tableau croisé, on procède ainsi :

```
table(donnees$tx)
```

```
## 
##  1  2 
## 32 28
```

```
donnees$tx <- factor(donnees$tx, 1:2, c("Expérimental", "Témoin"))
sex_tx <- table(donnees$sex, donnees$tx)
sex_tx
```

```
##         
##          Expérimental Témoin
##   Femme            16     16
##   Homme            16     12
```

Pour afficher le pourcentage en ligne, on fait :

```
prop.table(sex_tx,1)
```

```
##
##          Expérimental    Témoin
##    Femme    0.5000000 0.5000000
##    Homme    0.5714286 0.4285714
```

50% des femmes sont dans le groupe témoin.

Pour afficher le pourcentage en colonne, on fait :

```
prop.table(sex_tx,2)
```

```
##
##          Expérimental    Témoin
##    Femme    0.5000000 0.5714286
##    Homme    0.5000000 0.4285714
```

Pour afficher les pourcentages, on fait :

```
round(100* prop.table(sex_tx,1),1)
```

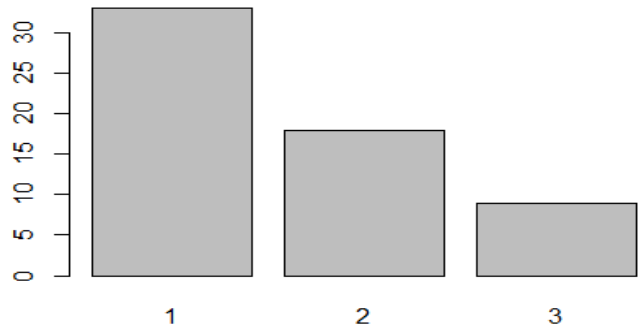
```
##
##          Expérimental Témoin
##    Femme         50.0   50.0
##    Homme         57.1   42.9
```

Visualisation

R donne la possibilité de faire plusieurs types de graphique. Le choix se fait en fonction du type des variables et de vos besoins.

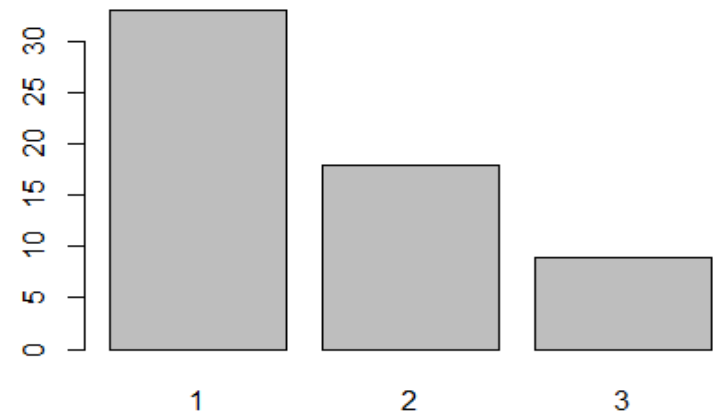
- Graphiques pour les variables catégorielles

```
donnees$affection <- factor(donnees$affection)
plot(donnees$affection)
```



On pourrait aussi utiliser la commande « barplot »; cependant, il faut d'abord mettre la variable sous forme de tableau.

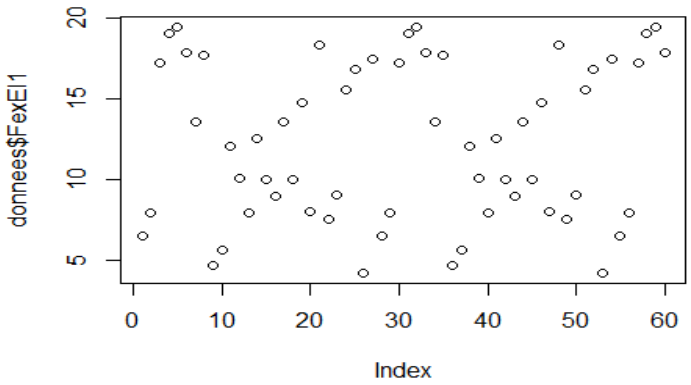
```
affec <- table(donnees$affection)
barplot(affec)
```



- Graphiques pour les variables continues

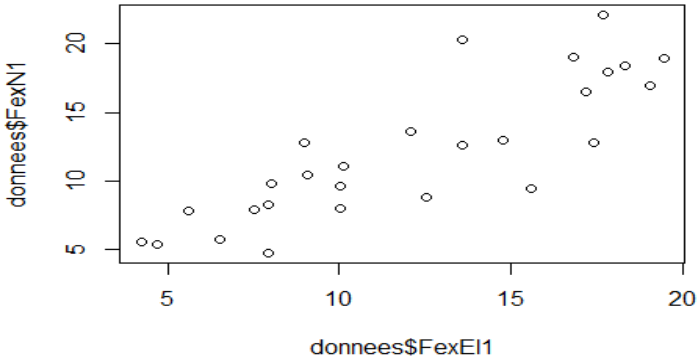
La commande « plot » peut aussi être utilisée pour une variable continue. C'est la commande la plus utilisée pour faire des graphiques.

```
plot(donnees$FexEl1)
```



Si on veut faire un graphique avec deux variables, on n'a qu'à ajouter le nom de la deuxième variable.

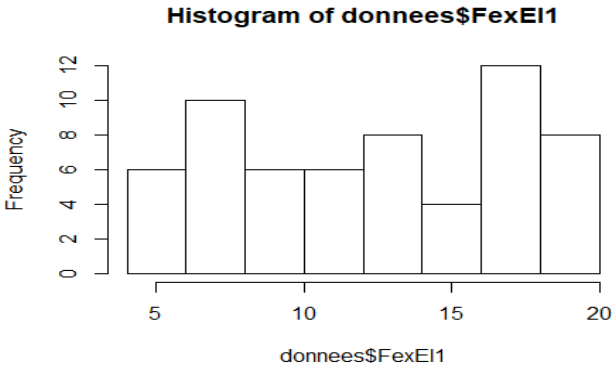
```
plot(donnees$FexEl1, donnees$FexN1)
```



Notez la différence entre les graphiques produits pour une variable de type « Facteur» et de type « numérique » en utilisant la commande `plot`.

Pour faire un histogramme, on procède ainsi:

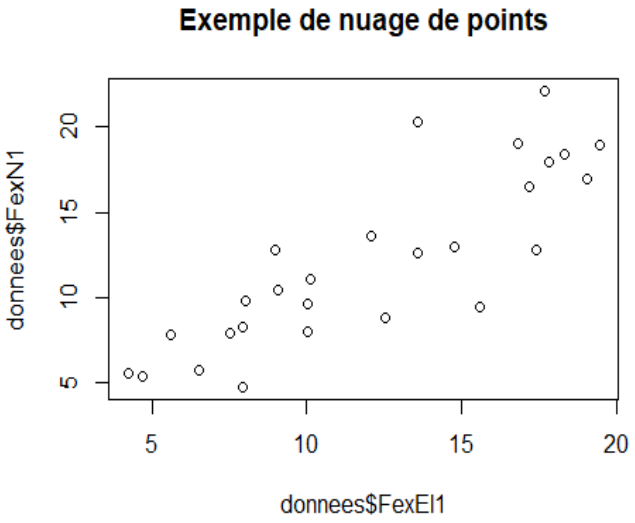
```
hist(donnees$FexEl1)
```



Ce sont les graphiques de base produits par R; néanmoins nous pouvons rendre ces graphiques plus complets en ajoutant par exemple des titres, des lignes ou des légendes. Pour la vaste majorité des graphiques les paramètres sont les mêmes tels que main, xlab, xlim.

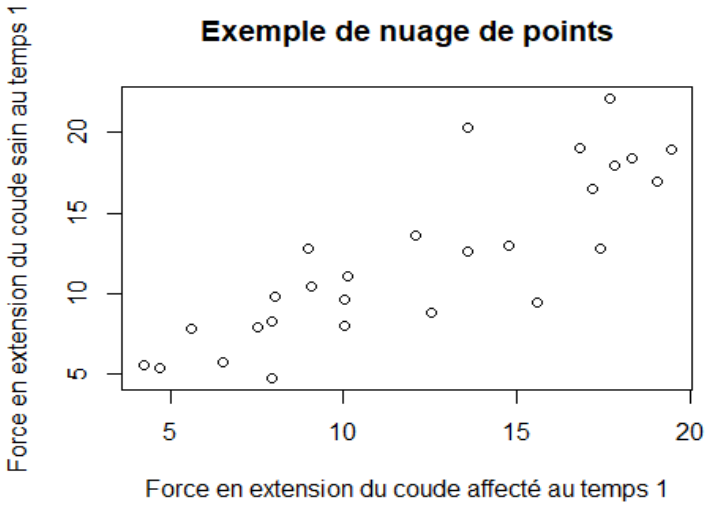
- Ajout de titre

```
plot(donnees$FexEl1, donnees$FexN1, main="Exemple de nuage de points")
```



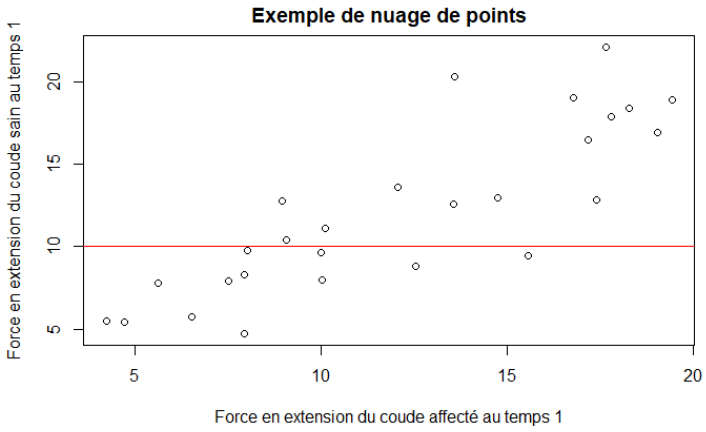
- Ajout du nom des axes

```
plot(donnees$FexEl1, donnees$FexN1, main="Exemple de nuage de points", xlab =
"Force en extension du coude affecté au temps 1", ylab = "Force en extension
du coude sain au temps 1")
```



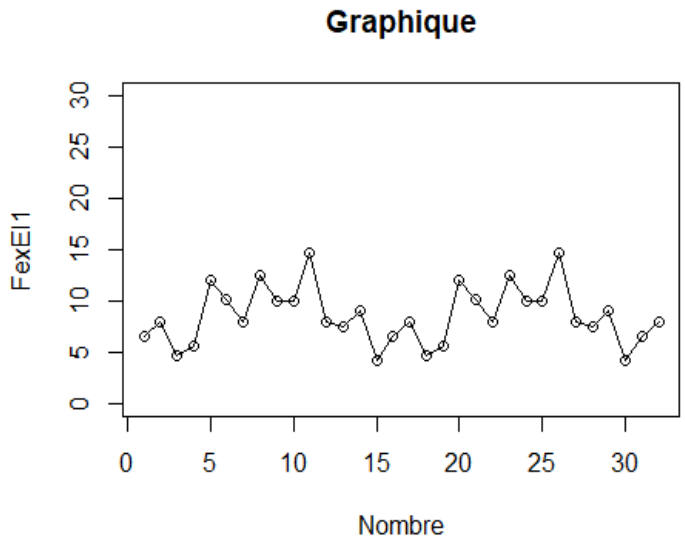
- Ajout d'une droite

```
plot(donnees$FexEl1, donnees$FexN1, main="Exemple de nuage de points", xlab =
"Force en extension du coude affecté au temps 1", ylab = "Force en extension
du coude sain au temps 1")
abline(a = 10, col = "red")
```



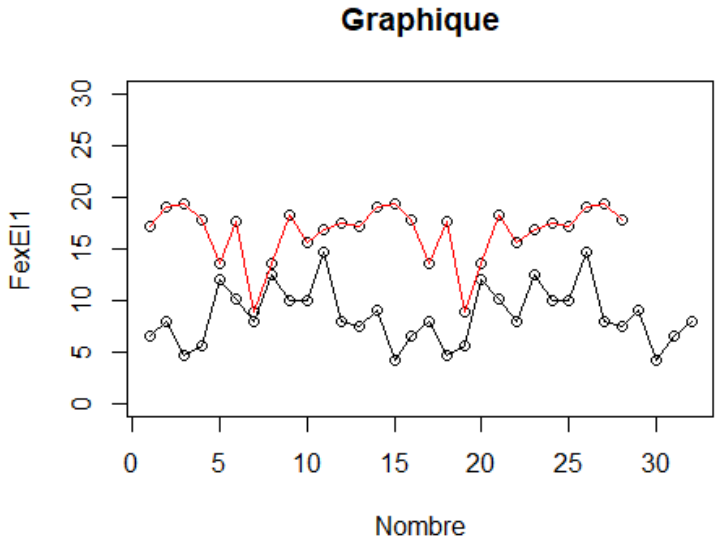
- Ajout d'une ligne

```
plot(donnees$FexEl1[donnees$sex=="Femme"], main="Graphique", ylab="FexEl1",
xlab="Nombre", ylim=c(0,30))
lines(donnees$FexEl1[donnees$sex=="Femme"])
```



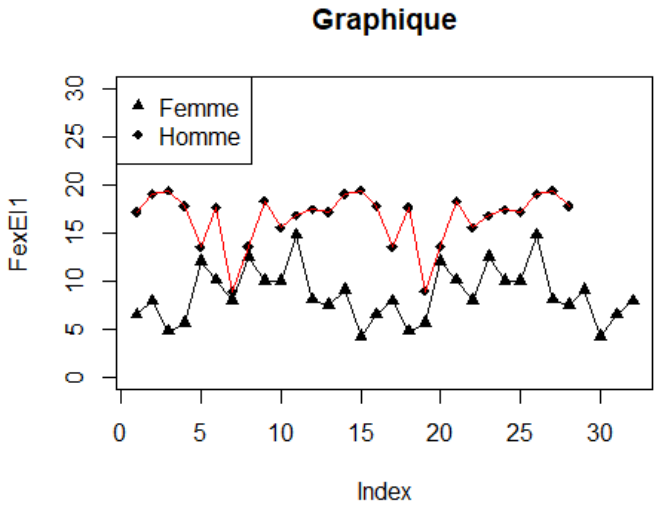
- Ajout de points

```
plot(donnees$FexEl1[donnees$sex=="Femme"], main="Graphique", ylab="FexEl1", x
lab="Nombre", ylim=c(0,30))
lines(donnees$FexEl1[donnees$sex=="Femme"])

points(donnees$FexEl1[donnees$sex=="Homme"])
lines(donnees$FexEl1[donnees$sex=="Homme"], col="red")
```



- Ajout de légendes

```
plot(donnees$FexEl1[donnees$sex=="Femme"], main="Graphique", ylim=c(0,30), yl
ab="FexEl1", pch=17)
lines(donnees$FexEl1[donnees$sex=="Femme"])

points(donnees$FexEl1[donnees$sex=="Homme"], pch=18)
lines(donnees$FexEl1[donnees$sex=="Homme"], col="red")
legend('topleft', pch=c(17,18), legend = paste(c("Femme", "Homme")))
```



Il existe le package ggplot qui permet d'aller plus loin dans les graphiques [http://docs.ggplot2.org/current/].

4- TESTS STATISTIQUES

Comparaison de moyennes

Test t de Student pour moyennes de groupes indépendants

Supposons qu'on ait une variable quantitative « FexEl1» et une variable catégorielle « sexe». On aimerait comparer si les moyennes de « FexEl1» sont différentes selon le sexe.

```
homme <- subset(donnees, sex=="Homme")
femme <- subset(donnees, sex=="Femme")
t.test(x=homme$FexEl1, y=femme$FexEl1)
```

```
##
##  Welch Two Sample t-test
##
## data:  homme$FexEl1 and femme$FexEl1
## t = 10.896, df = 57.064, p-value = 1.454e-15
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  6.470200 9.383809
## sample estimates:
## mean of x mean of y
## 16.569786  8.642781
```

Notez que R affiche la valeur p associée au test ainsi que les intervalles de confiance pour la différence entre les moyennes.

ou

```
t.test(donnees$FexEl1 ~ donnees$sex)
```

```
##
##  Welch Two Sample t-test
##
## data:  donnees$FexEl1 by donnees$sex
## t = -10.896, df = 57.064, p-value = 1.454e-15
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -9.383809 -6.470200
## sample estimates:
## mean in group Femme mean in group Homme
##           8.642781           16.569786
```

Test d'ANOVA pour plus de deux moyennes

Supposons maintenant qu'on aimerait comparer toujours la force de l'extension du coude « FexEl1» mais selon le type d'affection.

```
summary(donnees$affection)
```

```
##  1  2  3
## 33 18  9
```

Notez que la variable « affection» a plus de deux modalités. Par conséquent, il faut faire le test d'analyse de variance communément appelé ANOVA.

```
mod <- aov(donnees$FexEl1~donnees$affection)
summary(mod)
```

```
##                    Df Sum Sq Mean Sq F value Pr(>F)
## donnees$affection   2  117.7   58.83   2.621 0.0815 .
## Residuals          57 1279.6   22.45
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Test d'ANOVA pour mesures répétées

Supposons que les variables sont maintenant répétées par sujet. On ajoute maintenant la commande « Error(ID) » pour préciser que les données sont répétées avec comme identifiant « ID ».

```
mod <- aov(Poids~Cigarette+Error(ID), data=data)
mod
```

```
##
## Call:
## aov(formula = Poids ~ Cigarette + Error(ID), data = data)
##
## Grand Mean: 63.77778
##
## Stratum 1: ID
##
## Terms:
##                 Cigarette
## Sum of Squares       1176
## Deg. of Freedom         1
##
## Estimated effects are balanced
##
## Stratum 2: Within
```

```
##
## Terms:
##                 Cigarette Residuals
## Sum of Squares   173.5736  717.9820
## Deg. of Freedom         1         6
##
## Residual standard error: 10.93909
## Estimated effects are balanced
```

```
summary(mod)
```

```
##
## Error: ID
##           Df Sum Sq Mean Sq
## Cigarette  1   1176    1176
##
## Error: Within
##           Df Sum Sq Mean Sq F value Pr(>F)
## Cigarette  1  173.6   173.6   1.451  0.274
## Residuals  6  718.0   119.7
```

Test de Wilcoxon / Mann-Whitney

Les deux tests que nous venons de voir (t de Student et ANOVA) sont utilisés si la variable dépendante suit une loi normale. Dans le cas où cette hypothèse n'est pas respectée, on peut utiliser des tests non-paramétriques comme celui de Wilcoxon. Supposons que la variable « FexEl1» ne suit pas une distribution normale, nous allons donc utiliser le test de Wilcoxon pour comparer les moyennes de « FexEl1» selon le sexe.

```
wilcox.test(FexEl1 ~ sex, data=donnees)
```

```
## Warning in wilcox.test.default(x = c(6.524, 7.935, 4.723, 5.617, 12.049, :
## cannot compute exact p-value with ties
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  FexEl1 by sex
## W = 36, p-value = 1.053e-09
## alternative hypothesis: true location shift is not equal to 0
```

Notez que le test de Wilcoxon, tout comme celui de t de Student, est utilisé que si la variable catégorielle est binaire.

Comparaison de deux proportions

Les tests que nous avons vus sont utilisés lorsque nous avons une variable quantitative et une variable catégorielle. Si on a deux variables catégorielles, on fait le test de chi-deux. Le test de chi-deux est effectué sur un tableau de contingence. On peut donc écrire

```
table(donnees$sex, donnees$tx)
```

```
##
##          1  2
##   Femme 16 16
##   Homme 16 12
```

```
chisq.test(table(donnees$sex, donnees$tx))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table(donnees$sex, donnees$tx)
## X-squared = 0.086396, df = 1, p-value = 0.7688
```

5- REGRESSION ET CORRELATION

Régression linéaire

Dans le cas où on est intéressés à regarder l'association entre une variable dépendante continue et une autre variable continue/catégorielle, il faut utiliser la régression linéaire.

L'objectif de l'atelier étant de donner une introduction au logiciel R, les hypothèses d'utilisation de cette méthode ne seront pas vues ici.

Supposons qu'on veut regarder l'association entre «FexEl1» et « sexe» :

```
mod <- lm(donnees$FexEl1~donnees$sex)
summary(mod)

##
## Call:
## lm(formula = donnees$FexEl1 ~ donnees$sex)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.6068 -1.3673  0.5062  1.4752  6.0922
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        8.6428     0.4973   17.38  < 2e-16 ***
## donnees$sexHomme   7.9270     0.7279   10.89  1.2e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.813 on 58 degrees of freedom
## Multiple R-squared:  0.6716, Adjusted R-squared:  0.6659
## F-statistic: 118.6 on 1 and 58 DF,  p-value: 1.198e-15
```

ou

```
mod <- lm(FexEl1~sex, data=donnees)
summary(mod)

##
## Call:
## lm(formula = FexEl1 ~ sex, data = donnees)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.6068 -1.3673  0.5062  1.4752  6.0922
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)   8.6428     0.4973   17.38  < 2e-16 ***
## sexHomme      7.9270     0.7279   10.89  1.2e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.813 on 58 degrees of freedom
## Multiple R-squared:  0.6716, Adjusted R-squared:  0.6659
## F-statistic: 118.6 on 1 and 58 DF,  p-value: 1.198e-15
```

Notez l'utilisation de la commande `lm` pour linear models. R affiche les coefficients estimés et leurs erreurs-types. De même, vous trouverez les valeurs de la statistique du test t (t value) ainsi que les valeurs p associées.

Remarquez que R affiche les résultats pour la modalité «homme». Les étiquettes «homme» et «femme» ont été ajoutées pour faciliter l'interprétation. Ici, comme il s'agit d'une variable catégorielle, l'interprétation se fait par rapport à une catégorie de référence. Dans notre exemple, ce sont les femmes qui sont la référence.

Notez aussi que cette commande vous permet d'obtenir le coefficient de détermination (Multiple R-squared: 0.6716).

N.B : Comme la régression linéaire a comme fonction de lien « identité », on peut interpréter les coefficients directement.

Si on veut voir les intervalles de confiance, on procède ainsi:

```
confint(mod)
```

```
##                2.5 %   97.5 %
## (Intercept) 7.647411 9.638151
## sexHomme    6.469932 9.384077
```

Interprétation : Les personnes de sexe masculin ont en moyenne une force d'extension du coude de 7.93 comparativement aux femmes. L'association est statistiquement significative parce que l'intervalle de confiance ne contient pas la valeur 0.

Supposons maintenant que nous voulons regarder l'association entre «FexEl1» et «affection». Pour rappel, la variable «affection» est catégorielle et à 3 modalités. Nous reprenons la commande utilisée précédemment.

```
mod2 <- lm(donnees$FexEl1~donnees$affection)
summary(mod2)
```

```
##
## Call:
## lm(formula = donnees$FexEl1 ~ donnees$affection)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.6902 -3.4092 -0.3697  4.5183  7.3252
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        15.1983     1.4611  10.402 7.05e-15 ***
## donnees$affection  -1.7852     0.8298  -2.151   0.0356 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.723 on 58 degrees of freedom
## Multiple R-squared:  0.0739, Adjusted R-squared:  0.05793
## F-statistic: 4.628 on 1 and 58 DF,  p-value: 0.03563
```

Notez que R traite la variable «affection» comme si elle était continue. Pour que R puisse la considérer comme une variable catégorielle, il faut recoder la variable de façon à n'avoir que des variables binaires autrement appelées variables indicatrices. Comme «affection» a trois modalités, nous devons définir 2 variables binaires, l'autre étant la catégorie de référence.

De façon générale, si une variable a k modalités, il faut définir k-1 variables indicatrices.

Création des variables indicatrices

```
donnees$affection2 <- NA
donnees$affection2[donnees$affection==2] <-1
donnees$affection2[donnees$affection==1| donnees$affection == 3 ] <-0
donnees$affection3 <- NA
donnees$affection3[donnees$affection == 3] <- 1
donnees$affection3[donnees$affection == 1 | donnees$affection == 2]  <- 0

table(donnees$affection)

##
##  1  2  3
## 33 18  9

table(donnees$affection2)

##
##  0  1
## 42 18

table(donnees$affection3)

##
##  0  1
## 51  9
```

Dans le cas de cet exemple, la modalité 1 a été choisie comme référence.

Maintenant, relançons notre régression.

```
mod3 <- lm(donnees$FexEl1~donnees$affection2 + donnees$affection3)
summary(mod3)
```

```
##
## Call:
## lm(formula = donnees$FexEl1 ~ donnees$affection2 + donnees$affection3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8809 -3.7045 -0.4287  4.6467  6.6257
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          13.6039     0.8248  16.494   <2e-16 ***
## donnees$affection2   -2.6755     1.3883  -1.927   0.0590 .
## donnees$affection3   -3.0616     1.7818  -1.718   0.0912 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.738 on 57 degrees of freedom
## Multiple R-squared:  0.08421,    Adjusted R-squared:  0.05208
## F-statistic: 2.621 on 2 and 57 DF,  p-value: 0.0815
```

On peut donc constater qu'on a un coefficient pour l'affection de type 2 et un autre pour celle de type 3 et ces coefficients s'interprètent par rapport au type 1.

Nous venons de créer les variables indicatrices manuellement, il est plus facile de le faire automatiquement sur R en utilisant la commande `factor`.

```
mod4 <- lm(FexEl1~factor(affection), data=donnees)
summary(mod4)
```

```
##
## Call:
## lm(formula = FexEl1 ~ factor(affection), data = donnees)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8809 -3.7045 -0.4287  4.6467  6.6257
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          13.6039     0.8248  16.494   <2e-16 ***
## factor(affection)2   -2.6755     1.3883  -1.927   0.0590 .
## factor(affection)3   -3.0616     1.7818  -1.718   0.0912 .
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.738 on 57 degrees of freedom
## Multiple R-squared:  0.08421,    Adjusted R-squared:  0.05208
## F-statistic: 2.621 on 2 and 57 DF,  p-value: 0.0815
```

Remarquez que nous obtenons des résultats similaires avec les deux commandes.

Il est également possible de considérer plusieurs variables indépendantes dans le modèle de régression. Il suffit d'utiliser le signe «+» comme outil de liaison des variables indépendantes.

```
mod5 <- lm(FexEl1~sex + VAS1, data=donnees)
summary(mod5)

##
## Call:
## lm(formula = FexEl1 ~ sex + VAS1, data = donnees)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.3700 -1.4170  0.3405  1.8518  6.0185
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.45503    0.83186  11.366  2.8e-16 ***
## sexHomme     7.79665    0.73283  10.639  3.7e-15 ***
## VAS1        -0.01801    0.01482  -1.215    0.229
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.801 on 57 degrees of freedom
## Multiple R-squared:  0.6799, Adjusted R-squared:  0.6686
## F-statistic: 60.52 on 2 and 57 DF,  p-value: 7.986e-15
```

Corrélation

On peut aussi vouloir tester la corrélation entre deux ou plusieurs variables.

Supposons qu'on aimerait savoir si les deux mesures de la force en extension sont corrélées:

```
cor(donnees$FexEl1, donnees$FexEl2)

## [1] 0.9350385
```

Notez qu'il existe des arguments optionnels à cette commande, que vous utiliseriez selon vos besoins. Toutefois, il est important d'utiliser le fichier `Aide` car R gère les valeurs manquantes de manière particulière.

Supposons qu'on ajoute des valeurs manquantes pour ces variables:

```
cor(donnees$FexEl1, donnees$FexEl2)

## [1] NA
```

Vous pouvez donc remarquer que la corrélation ne s'affiche pas. En fait, il faut préciser que nous voulons utiliser seulement les observations complètes.

```
cor(donnees$FexEl1, donnees$FexEl2,use="c")

## [1] 0.9350385
```

Maintenant, nous obtenons la valeur trouvée quand il n'y avait pas de données manquantes. Notez l'utilisation de use="c" (complete.obs) pour dire que nous voulons utiliser les observations complètes. Pour plus de détails sur cette commande, vous pouvez l'Aide sur R.

Régression logistique

Dans le cas où on est intéressés à regarder l'association entre une variable dépendante binaire et une autre variable continue/catégorielle, il faut utiliser la régression logistique.

Supposons qu'on veut regarder l'association entre «tx» et « FexEl1» :

```
mod <- glm(tx~FexEl1, data=donnees, family=binomial(link = "logit"))
summary(mod)

##
## Call:
## glm(formula = tx ~ FexEl1, family = binomial(link = "logit"),
##     data = donnees)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.234  -1.125  -1.020   1.200   1.326
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.29585    0.71167   0.416    0.678
## FexEl1      -0.03487    0.05392  -0.647    0.518
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 82.911  on 59  degrees of freedom
## Residual deviance: 82.490  on 58  degrees of freedom
## AIC: 86.49
```

```
##
## Number of Fisher Scoring iterations: 4
```

Notez l'utilisation de la commande `glm` pour generalized linear models et de la commande `family = binomial(link = "logit")` pour préciser à R qu'il s'agit d'une distribution binomiale avec comme fonction de lien le logit.

Comme dans la régression linéaire, R affiche les coefficients estimés et leurs erreurs-types. Mais à la différence de la régression linéaire, les coefficients estimés ne sont pas directement interprétables. Il faut les considérer sur l'échelle de rapport de côtes.

```
exp(mod$coefficients)

## (Intercept)      FexEl1
##   1.3442708   0.9657347

exp(confint(mod))

## Waiting for profiling to be done...

##                 2.5 %   97.5 %
## (Intercept) 0.3312756 5.548629
## FexEl1      0.8670565 1.073081
```

La commande précédente permet d'obtenir le rapport de cotes ainsi que l'intervalle de confiance associé.

Interprétation : Le rapport de cotes estimé est de 0.96. L'augmentation d'une unité de la force en extension du code est associée avec une diminution de la cote d'être traité de (100-96) = 4%. L'association n'est pas statistiquement significative car l'intervalle de confiance contient la valeur 1.

On peut ajouter d'autres variables en procédant de la même manière que pour la régression linéaire.

En résumé

- R est un outil de programmation et d'analyse des données
- Importation des données (format csv): read.table, read.csv
- Description des données :
  - ✓ Statistiques descriptives : summary, mean, sd, table
  - ✓ Visualisation : plot, barplot, hist, boxplot
- Tests statistiques
  - ✓ 1 variable continue et 1 variable binaire : t.test
  - ✓ 1 variable continue et 1 variable catégorielle : ANOVA
  - ✓ 2 variables binaires : chi-deux
- Régression
  - ✓ Linéaire : lm
  - ✓ Logistique : glm
- Modèle des risques instantanés de Cox : coxph (voir bibliothèque « survival »)

- Liens utiles :
  - ✓ https://stats.idre.ucla.edu/other/dae/
  - ✓ http://www.sthda.com/english/
  - ✓ https://cran.r-project.org/web/packages/index.html
  - ✓ https://cran.r-project.org/doc/contrib/Seefeld_StatsRBio.pdf