



Initiation à R

Pour Master 2 MBFA-Analyse des risques de marchés

Sonia Tieo

Ingénieur d'étude Bioinformatique/Biostatistique (IMAG)

Pour avoir le lien du cours et des exercices de TP:

https://github.com/soniamaitieo/CoursM2_MBFA

Contact: sonia.tieo@umontpellier.fr

1. Introduction
2. Les objets R
3. Séance TP1: Se familiariser avec R
4. Séance TP2: Statistiques descriptives avec R

Intro

R est un logiciel de Statistique distribué librement par le CRAN, créé dans les années 90 par R. Ihaka et R. Gentleman:

<http://cran.r-project.org/>

Il est dédié à l'analyse statistique et à la visualisation de données.

Pour installer R :

<http://cran.univ-paris1.fr/>

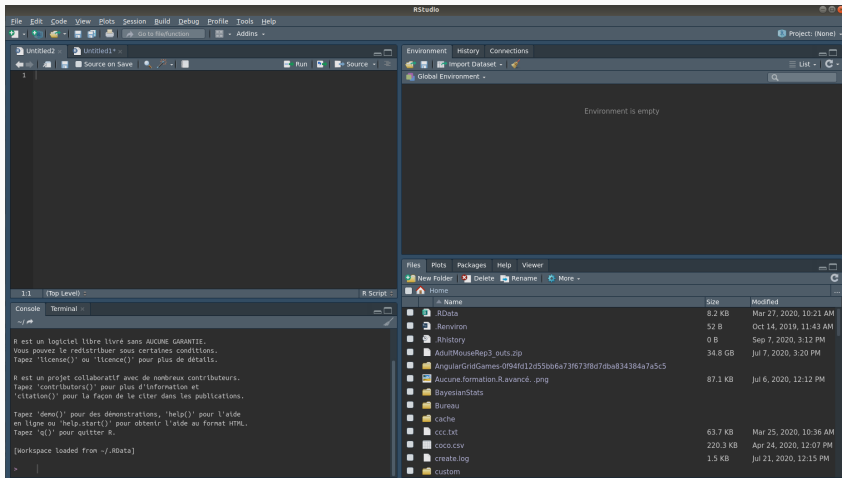
Choisissez votre plateforme....

R est disponible sous de nombreux systèmes d'exploitation.
(Windows, Mac OS X ou Linux)

R est simple à installer, il suffit de suivre les instructions.

L'utilisation de R est facilitée par l'utilisation d'un environnement de développement intégré (IDE). Ici nous utiliserons RStudio:

<https://rstudio.com/products/rstudio/download/>



Rstudio est divisé en 4 sous-fenêtres :

1. Editeur de texte, de codes... (haut-gauche);
2. Console (bas-gauche);
3. Environnement, historique, importation (haut-droit)
4. Fichiers, graphiques, package, aide (bas-droit)

Ouverture de session

```
R version 3.6.2 (2019-12-12) -- "Dark and Stormy Night"  
Copyright (C) 2019 The R Foundation for Statistical Computing  
Platform: x86_64-pc-linux-gnu (64-bit)
```

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

R attend une instruction : ceci est indiqué par > en début de ligne. Cette instruction doit être validée par Entrée pour être exécutée.

- instruction correcte, R exécute et redonne la main >
- instruction incomplète R retourne +, il faut alors compléter l'instruction ou sortir avec **Echap** ou **Ctrl+C** en mode console.

Conseil

Pour chaque projet: utiliser un répertoire de travail pour sauvegarder tout ce qui est lié au projet: jeu(x) de données, script(s), graphe(s), environnement...

Au démarrage d'une session R, le repertoire de travail peut-être indiqué avec la commande suivante:

```
> getwd() #get working directory
```

Pour modifier le répertoire de travail:

```
> setwd("C:\\Users") #notation typée windows  
> setwd("C:/Users") #notation typée linux
```

SAUVEGARDE D'OBJETS: R conserve les objets stockée dans la fenêtre environnement (haut-droite)

SAUVEGARDE DE LA SESSION: Ces objets peuvent être sauvegardé sans une image de la session appelée **.RData** grâce à la commande:

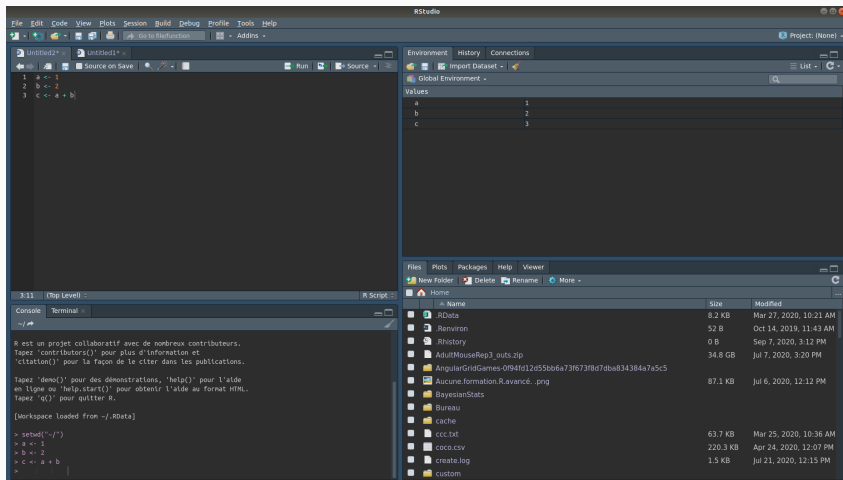
```
> save.image()
```

ou en quittant avec

```
> q() # en redémarrant la session, les objets seront chargés à nouveau
```

SAUVEGARDE SCRIPT: A sauvegarder dans le répertoire de travail en cliquant sur la disquette (ou **File** puis **Save**) pour garder une trace en format **.r**

Sauvegarde



Disponibles sur le CRAN :

<https://cran.r-project.org/>

L'installation se fait avec l'onglet **Packages** dans **RStudio** ou avec la fonction **install.packages**

```
install.packages("ibr")  
# Une fois installé, il faut charger le package  
library(ibr)
```

La session R comme une calculatrice

```
> 1+1  
[1] 2  
> pi  
[1] 3.141593  
> sin(pi/2)  
[1] 1
```


Premières commandes R

R est composée de fonctions et/ou opérateurs qui agissent sur des objets.

CRÉATION:

```
> a <- 1.2 # a est un objet que l'on crée en lui donnant la
           valeur 1.2
> x <- a # x reçoit la valeur a (donc 1.2)
> x = a # x reçoit la valeur a (donc 1.2)
> a -> x # x reçoit la valeur a
```

Règles d'écriture

- le nom des variables commence par une lettre
- le nom des variables peuvent contenir des caractères spéciaux
- respecter la casse ("a" différent "A")
- si l'objet existe déjà, sa valeur précédente est effacée.
- le nom de la variable ne doit pas être appelé par des voms existants dans le langage R

Premières commandes R

AFFICHAGE:

```
> print(x)  
> x # alternative
```

LISTER CONTENU MÉMOIRE:

```
> ls() # Affiche tous les objets situés dans la mémoire de l'  
      environnement R (par exemple: a et x)
```

SUPPRIMER:

```
> rm(a) # Avec a qui représente le nom de l'objet à supprimer
```

AIDE:

```
> help(ls)  
> ?ls
```

Les objets R

Les types de données

LE MODE D'UN OBJET:

- booléen (**logical**) : TRUE, FALSE
- **numeric** (integer) : 1 ou (double) : 3.14
- caractères (**character**) : 'bonjour', "hello"
- vide (**null**) : NULL
- complexe (**complex**) : 2+0i, 2i

Pour connaître le mode d'un objet:

```
> mode(a) # qui retourne numeric
```

On peut aussi tester l'appartenance d'un objet à un mode:

```
> is.character(a) # retourne FALSE  
> is.numeric(a) #retourne TRUE
```

STRUCTURATION DES OBJETS:

- monotype (ou atomique), tous les éléments sont de même type (vecteur, matrice, tableaux)
- de types différents : liste, data.frame

Un vecteur est composé de données de même mode.
Chaque élément est séparé par une virgule.

CRÉATION D'UN VECTEUR:

Création avec la fonction "collecteur" `c`:

```
> monVEC <- c(3.2,-2.1,100,0.5) # vecteur numérique de 4 données
> monVEC2 <-c(monVEC , 1, c(2,3)) #poupée russe
> monVEC2
[1] 3.2 -2.1 100.0 0.5 1.0 2.0 3.0
```

Création avec l'opérateur séquence `"1:5"` :

```
> 1:5
> [1] 1 2 3 4 5
```

Les vecteurs

Création avec la fonction `seq`:

```
> seq(1,5)
[1] 1 2 3 4 5
> seq(1,6, by=0.5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0
> seq(1,6,length=5)
[1] 1.00 2.25 3.50 4.75 6.00
```

Création avec la fonction `rep`:

```
> rep("cool",4)
[1] "cool" "cool" "cool" "cool"
> rep(c("not","cool"), each=3)
[1] "not" "not" "not" "cool" "cool" "cool"
> rep(c("not","cool"), times=3)
[1] "not" "cool" "not" "cool" "not" "cool"
```



Manipulation vecteurs

TAILLE D'UN VECTEUR:

```
> length(monVEC)
[1] 4
> length(seq(1,6, by=0.5))
[1] 11
```

CONCATÉNER DES CARACTÈRES:

```
> paste("X",1:5,sep="-")
[1] "X-1" "X-2" "X-3" "X-4" "X-5"
> paste(c("X","Y"),1:5,"txt",sep=".")
[1] "X.1.txt" "Y.2.txt" "X.3.txt" "Y.4.txt" "X.5.txt"
> paste("X",1:5,sep=".",collapse="+")
[1] "X.1+X.2+X.3+X.4+X.5"
```


Manipulation vecteurs

EXTRACTION DE CARACTÈRES PAR POSITION:

Exemple: Extraction de caractères du mot "livre" entre les rangs 2 et 5

```
> substr("livre",2,5)
[1] "ivre"
```

EXTRACTION DE CARACTÈRES PAR MOTIFS:

grep rechercher le motif dans le mot et donne la position du mot avec le motif et **gsub** remplace le motif

```
> txtvec <- c("truc.jpg" , "truc2.gif" , "truc3.jpg" , "machin.
  txt" , "machine.jpg")
> grep("jpg",txtvec)
[1] 1 3 5
> gsub("truc", txtvec, replacement = "TRUC")
[1] "TRUC.jpg"      "TRUC2.gif"     "TRUC3.jpg"     "machin.txt"    "
  machine.jpg"
```

Manipulation vecteurs: sélection

Pour sélectionner une partie du vecteur, on utilise l'opérateur de sélection `[]` ainsi qu'un **vecteur de sélection**:

```
> monvec[vecteurdeselection]
```

Ce vecteur de sélection peut-être un vecteur d'entiers positifs, négatifs ou de logiques Quelques exemples:

```
> v <- 1:10  
> v[6] #donne le 6è élément de v  
[1] 6  
> v[6:8] #donne le 6è, 7è et 8è élément de v  
[1] 6 7 8  
> v[c(6,6,1:2)] #donne les 6è, 6è, 1er et 2è éléments de v  
[1] 6 6 1 2  
> v[5:1] #donne les 5è, 4è, 3è, 2è puis 1er élément de la liste  
[1] 5 4 3 2 1
```

Manipulation vecteurs: sélection

On peut aussi enlever des éléments du vecteur:

```
> v[-(1:5)] #v sans les 5 premiers éléments
[1] 6 7 8 9 10
> v[-c(1,5)] # v sans le 1er et le 5ème élément
[1] 2 3 4 6 7 8 9 10
```

Exemple de sélection par des vecteurs logiques:

```
> v <- 1:15
> (v<5) #ce vecteur est un vecteur logique car il est composé de
TRUE et FALSE
[1] TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
     FALSE FALSE FALSE FALSE FALSE
> v[(v<5)]
[1] 1 2 3 4
```

Notes sur les vecteurs logiques

Les vecteurs logiques sont en partie générés par des opérateurs logiques ">", "<", ">=", "<=", "==", "!="

Manipulation vecteurs: sélection

Exemple de sélection par des vecteurs logiques (suite):

```
> v <- 1:15
> v[(v>5) & (v<10)]
[1] 6 7 8 9
> v[(v<5) & (v>10)]
integer(0)
> v[(v<5) | (v>10)] #signifie "ou"
[1] 1 2 3 4 11 12 13 14 15
```

Exemple de sélection sur un vecteur character:

```
> txtvec
[1] "truc.jpg"      "truc2.gif"      "truc3.jpg"      "machin.txt"      "
    machine.jpg"
> txtvec[grep("jpg",txtvec)]
[1] "truc.jpg"      "truc3.jpg"      "machine.jpg"
```

Manipulation vecteurs: sélection

Autre cas de sélection: Trouver les valeurs extrêmes

```
> b <- c(1,5,3,9,23,53,9,2)
> which.max(b) #donne la position où se trouve le plus grand
élément du vecteur b
[1] 6
> b[which.max(b)]
[1] 53
> #Equivalent à:
> which(b==max(b))
[1] 6
> b[which(b==max(b))]
[1] 53
```

Nommer les éléments d'un vecteur

Jusque là, ce sont des objets à part intégrale que nous avons nommés. On les a assignés des noms pour les garder dans notre environnement de travail. Maintenant, nous allons donner un nom aux éléments de vecteur. Dressons l'analogie suivante. Notre environnement dans R est comme une rue. Dans celle-ci, nous avons des concessions dont les portes sont toutes numérotées: ce sont les noms des objets. A l'intérieur des concessions, nous avons des individus: ce sont les éléments à l'intérieur de nos objets. Tout comme ces individus portent des prénoms, nous pouvons donner des appellations aux éléments contenus dans nos objets.

Nommer les éléments d'un vecteur

```
> v <- 1:5
> v
[1] 1 2 3 4 5
> vnames <- c("un" , "deux", "trois" , "quatre" , "cinq")
> vnames
[1] "un"      "deux"    "trois"   "quatre"  "cinq"
> names(v) <- vnames
> v
      un      deux      trois quatre      cinq
      1        2        3        4        5
```

La matrice c'est un ensemble de vecteurs.

- Elle hérite d'une propriété fondamentale du vecteur: ne peuvent former une matrice que des éléments de même nature.
- Diffère par sa bi-dimensionnalité

CRÉATION D'UNE MATRICE:

Création avec la fonction `matrix`

```
> X <- matrix(c("R", "T", "G", "Y"), ncol=2, nrow=2)
> X
      [,1] [,2]
[1,]  "R"  "G"
[2,]  "T"  "Y"
> x <- matrix(1:6, nrow=2, ncol=3, byrow=TRUE)
> x
      [,1] [,2] [,3]
[1,]     1     2     3
[2,]     4     5     6
> x2 <- matrix(1:6, nrow=2, ncol=3, byrow=FALSE)
> x2
      [,1] [,2] [,3]
[1,]     1     3     5
[2,]     2     4     6
```

CRÉATION D'UNE MATRICE:

Transformer un vecteur en matrice `as.matrix`

#On peut créer un objet matrice à 1 dimension

```
> y <- matrix(1:2, ncol=1)
```

```
> y
```

```
      [,1]
```

```
[1,]    1
```

```
[2,]    2
```

#On transforme un vecteur en matrice

```
> v <- 1:5
```

```
> vmatrix <- as.matrix(v)
```

```
> vmatrix
```

```
      [,1]
```

```
[1,]    1
```

```
[2,]    2
```

```
[3,]    3
```

```
[4,]    4
```

```
[5,]    5
```

CRÉATION D'UNE MATRICE PAR CONCATÉNATION DES VECTEURS:

Coller horizontalement des vecteurs `rbind`

```
> prenom = c("Jeff", "Britta", "Abed", "Shirley", "Annie", "Troy", "Pierce")
> nom = c("Winger", "Perry", "Nadir", "Bennett", "Edison", "Barnes", "Hawthorne")
> prenom_nom_hmatrix <- rbind(prenom, nom)
> prenom_nom_hmatrix
      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[ ,7]
prenom "Jeff"    "Britta" "Abed"  "Shirley" "Annie"  "Troy"
      "Pierce"
nom     "Winger" "Perry"  "Nadir" "Bennett" "Edison" "Barnes"
      "Hawthorne"
```

CRÉATION D'UNE MATRICE PAR CONCATÉNATION DES VECTEURS:

Coller verticalement des vecteurs `rbind`

```
> prenom_nom_vmatrix <- cbind(prenoms, noms)
> prenom_nom_vmatrix
      prenoms  noms
[1,] "Jeff"    "Winger"
[2,] "Britta"  "Perry"
[3,] "Abed"    "Nadir"
[4,] "Shirley" "Bennett"
[5,] "Annie"   "Edison"
[6,] "Troy"    "Barnes"
[7,] "Pierce"  "Hawthorne"
```

Manipulation matrices

TAILLE D'UNE MATRICE:

Pour connaître le nombre total d'éléments dans la matrice **length**

```
> length(prenom_nom_vmatrix)
[1] 14
```

DIMENSION D'UNE MATRICE:

Pour connaître le nombre de lignes et colonnes **dim** , **nrow** et **ncol**

```
> dim(prenom_nom_vmatrix)
[1] 7 2
> nrow(prenom_nom_vmatrix)
[1] 7
> ncol(prenom_nom_vmatrix)
[1] 2
```

Manipulation matrices: sélection

SÉLECTION D'ÉLÉMENTS OU D'UNE PARTIE D'UNE MATRICE:

La position d'un élément est donnée par le numéro de sa ligne i et de sa colonne j . Pour sélectionner l'élément i,j de la matrice m :

```
> m[i,j]
```

Exemple:

```
# L'élément de la 3ème ligne et le 2ème colonne
```

```
> prenom_nom_vmatrix[3,2]
```

```
      noms
```

```
"Nadir"
```

```
#3ème ligne toute entière
```

```
> prenom_nom_vmatrix[3,]
```

```
pre noms
```

```
"Abed" "Nadir"
```

```
#la 1ère colonne toute entière
```

```
> prenom_nom_vmatrix[,1]
```

```
[1] "Jeff"    "Britta"  "Abed"    "Shirley" "Annie"   "Troy"
     "Pierce"
```

Manipulation matrices: sélection

SÉLECTION D'ÉLÉMENTS PAR NOM:

Si les rangées sont nommées, alors il est aussi possible de passer par ces noms pour les sélectionner. Pour connaître le nom des lignes et des colonnes, on peut utiliser les fonctions **rownames** ou **colnames** implémentées:

```
> colnames(prenom_nom_vmatrix)
[1] "prenoms" "noms"
> rownames(prenom_nom_vmatrix) #ici les lignes n'ont pas de noms
NULL
```

Exemple de sélection:

```
> prenom_nom_vmatrix[,colnames(prenom_nom_vmatrix) == "prenoms"]
[1] "Jeff"      "Britta"    "Abed"      "Shirley"   "Annie"     "Troy"
     "Pierce"
```

Les facteurs sont des vecteurs qui permettent de manipuler des données qualitatives.

- Tout comme un vecteur elle possède une longueur
- C'est un vecteur particulier uniquement constitué d'éléments de type **levels**
- Ils peuvent être non ordonnés ou ordonnés
- On peut utiliser la fonction **plots** dessus

Les facteurs

CRÉATIONS DES FACTEURS:

Avec la fonction **factor** :

```
> sexe <- factor(c("M", "F", "M", "F", "F", "M", "M")) # on peut aussi
  utiliser la fonction as.factor()
> sexe
[1] M F M F F M M
Levels: F M
#on peut également nommer chaque niveau
> factor(c(2,1,2,1,1,2,2), labels = c("femme" , "homme"))
[1] homme femme homme femme femme homme homme
Levels: femme homme
```

Avec la fonction **ordered** :

```
> niveau <- ordered(c("débutant", "débutant", "champion", "moyen", "
  moyen", "moyen", "champion"), levels=c("débutant", "moyen", "
  champion"))
> niveau
[1] débutant débutant champion moyen      moyen      moyen
     champion
Levels: débutant < moyen < champion
```

ATTRIBUTS DES FACTEURS:

```
> levels(sexe)
[1] "F" "M"
> nlevels(sexe)
[1] 2
> table(sexe)
sexe
 F  M
 3  4
> plot(sexe)
```

L'objet **liste** peut contenir des objets qui peuvent avoir des modes différents et des tailles différentes. Ces objets appelés composants de la liste peuvent avoir un nom.

CRÉATION DE LISTES:

On utilise les éléments créés dans les exemples précédents:

```
> num_etu <- 1:7
> prenomes <- c("Jeff", "Britta", "Abed", "Shirley", "Annie", "Troy", "
  Pierce")
> noms <- c("Winger", "Perry", "Nadir", "Bennett", "Edison", "
  Barnes", "Hawthorne")
> sexe <- factor(c("M", "F", "M", "F", "F", "M", "M"))
> niveau_espagnol <- ordered(c("débutant", "débutant", "champion", "
  moyen", "moyen", "moyen", "champion"), levels=c("débutant", "
  moyen", "champion"))
> greendale_list <- list(num_etu, prenomes, noms, sexe, niveau_
  espagnol)
```

Les listes

```
> greendale_list
[[1]]
[1] 1 2 3 4 5 6 7

[[2]]
[1] "Jeff"      "Britta"    "Abed"      "Shirley"   "Annie"     "Troy"
     "Pierce"

[[3]]
[1] "Winger"    "Perry"     "Nadir"     "Bennett"   "Edison"
     "Barnes"   "Hawthorne"

[[4]]
[1] M F M F F M M
Levels: F M

[[5]]
[1] débutant débutant champion moyen      moyen      moyen
     champion
Levels: débutant < moyen < champion
```

Attention: les composantes de la liste créée n'ont pas de noms. On peut les nommer:

```
> names(greendale_list) = c("num_etu" , "prenom" , "nom" , "
  sexe" , "espagnol")
> names(greendale_list)
[1] "num_etu" "prenom" "nom" "sexe" "espagnol"
```

Les listes: sélection

SÉLECTION D'ÉLÉMENTS PAR SA POSITION:

```
> greendale_list[[2]] #2eme composant de la liste
[1] "Jeff"      "Britta"   "Abed"     "Shirley"  "Annie"    "Troy"
     "Pierce"

> greendale_list[[2]][1] #1er element du 2eme composant de la
    liste
[1] "Jeff"

> greendale_list[c(2,5)] #2eme et 5eme composant de la liste
$prenom
[1] "Jeff"      "Britta"   "Abed"     "Shirley"  "Annie"    "Troy"
     "Pierce"

$espagnol
[1] débutant débutant champion moyen      moyen      moyen
     champion

Levels: débutant < moyen < champion
```

Les listes: sélection

SÉLECTION D'ÉLÉMENTS PAR NOM:

Les 2 sont équivalents

```
> greendale_list$prenom  
[1] "Jeff"      "Britta"   "Abed"     "Shirley"  "Annie"    "Troy"  
     "Pierce"  
> greendale_list[["prenom"]]  
[1] "Jeff"      "Britta"   "Abed"     "Shirley"  "Annie"    "Troy"  
     "Pierce"
```

QUELQUES FONCTIONS SUR LES LISTES:

#Ajout d'un composant à la liste

```
> greendale_list2 = greendale_list  
> greendale_list2$metier = c("avocat")
```

#concaténer des listes

```
> c(greendale_list,greendale_list2)
```

#créer un vecteur contenant tous les éléments de la liste (ne marche que si tous les éléments sont de même mode)

```
> unlist(maliste)
```


Les dataframes

Liste spéciale : dataframes

Tableau de données : variables quantitatives + qualitatives

CRÉATION DE DATAFRAMES: C

```
> df_greendale <- data.frame(num_etu, prenoms, noms, sexe, niveau_espagnol)
> df_greendale
```

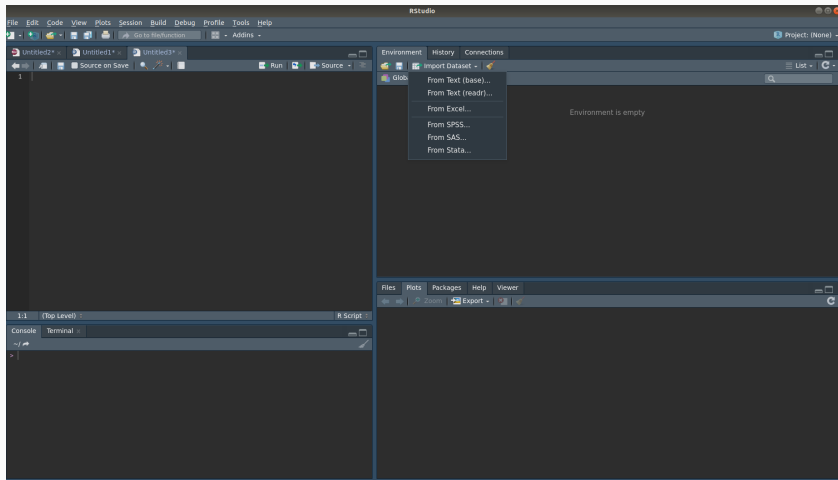
| | num_etu | prenoms | noms | sexe | niveau_espagnol |
|---|---------|---------|-----------|------|-----------------|
| 1 | 1 | Jeff | Winger | M | débutant |
| 2 | 2 | Britta | Perry | F | débutant |
| 3 | 3 | Abed | Nadir | M | champion |
| 4 | 4 | Shirley | Bennett | F | moyen |
| 5 | 5 | Annie | Edison | F | moyen |
| 6 | 6 | Troy | Barnes | M | moyen |
| 7 | 7 | Pierce | Hawthorne | M | champion |

Il est possible d'importer tous les formats de fichiers car il existera un package pour vous aider.

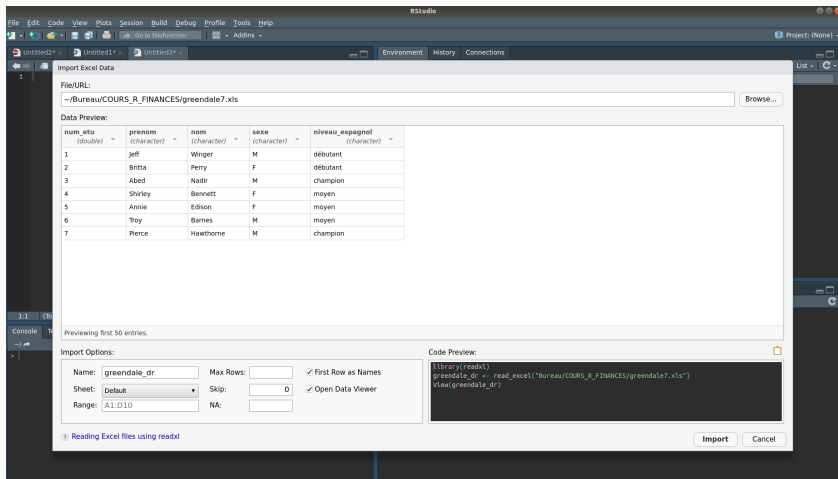
Il est possible d'importer directement des fichiers avec Rstudio et ce dernier vous propose directement dans import Dataset différents formats (Excel, SPSS, SAS, Stat) mais aussi à partir de Text avec 2 possibilités base ou readr.

Comme son nom l'indique, base propose d'utiliser la version de base de R que nous présentons dans un petit exemple. L'objet obtenu est un data.frame et toutes les variables character sont considérées comme des facteurs.

Importations Excel



Importations Excel

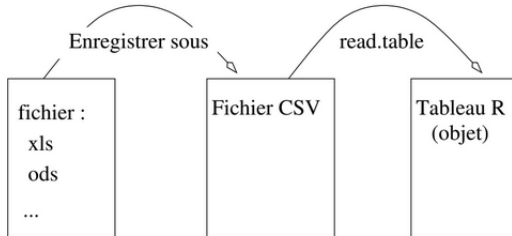


En ligne de commande:

```
greendale7 <- read_excel("Bureau/COURS_R_FINANCES/greendale7.xls")
```

Importations CSV

Sur R, il est recommandé d'importer des tableaux en formats CSV et de convertir les fichiers en csv.



Importations CSV

Import Dataset

Name

Encoding Automatic ▼

Heading ☒ Yes ☐ No

Row names Automatic ▼

Separator Comma ▼

Decimal Period ▼

Quote Double quote (") ▼

Comment None ▼

na.strings

☒ Strings as factors

Input File

```
num_etu,prenom,nom,sexe,niveau_espagnol
1,Jeff,Winger,M,débutant
2,Britta,Perry,F,débutant
3,Abed,Nadir,M,champion
4,Shirley,Bennett,F,moyen
5,Annie,Edison,F,moyen
6,Troy,Barnes,M,moyen
7,Pierce,Hawthorne,M,champion
```

Data Frame

| num_etu | prenom | nom | sexe | niveau_espagnol |
|---------|---------|-----------|------|-----------------|
| 1 | Jeff | Winger | M | débutant |
| 2 | Britta | Perry | F | débutant |
| 3 | Abed | Nadir | M | champion |
| 4 | Shirley | Bennett | F | moyen |
| 5 | Annie | Edison | F | moyen |
| 6 | Troy | Barnes | M | moyen |
| 7 | Pierce | Hawthorne | M | champion |

Import

Cancel

Importations CSV

Pour importer en ligne de commande:

```
greendale7 <- read.csv("~/Bureau/COURS_R_FINANCES/greendale7.csv")
```

Pour exporter en ligne de commande:

```
write.table(greendale7, "C:/temp/mydf_greendale.txt",  
            quote=TRUE, sep=",", "row.names=FALSE, col.names=TRUE)
```

SÉLECTION D'ÉLÉMENTS PAR SA POSITION:

Avec les numéros des coordonnées (cf. matrices)

SÉLECTION D'ÉLÉMENTS PAR DES NOMS:

Comme pour les matrices:

```
> greendale7[,c("prenom" , "nom")]  
  prenom      nom  
1   Jeff   Winger  
2 Britta   Perry  
3   Abed   Nadir  
4 Shirley Bennett  
5   Annie  Edison  
6   Troy   Barnes  
7 Pierce Hawthorne
```

SÉLECTION D'ÉLÉMENTS DES LOGIQUES (CF. MATRICES)

Séance TP1: Se familiariser avec R

Séance TP2: Statistiques descriptives avec R

Les données

DONNÉES:

On illustrera les fonctions de base en utilisant le jeu de données **mtcars** disponible dans R. Pour une description de ces données, consulter l'aide `?mtcars`.

```
# charger les données  
data(mtcars)
```

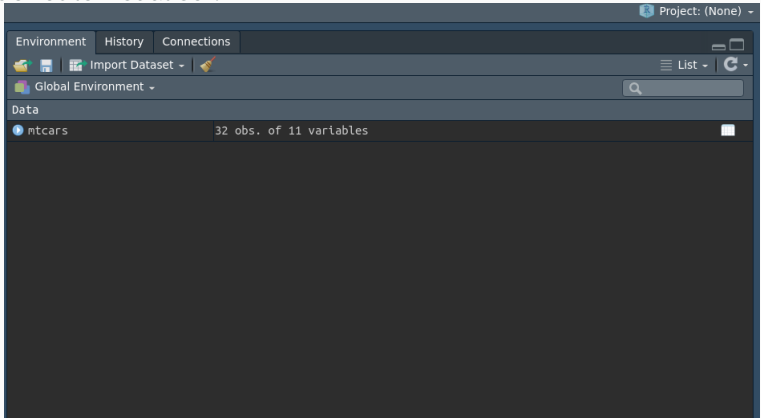
On peut avoir une idée de la classe et du contenu avec **str** et observer le début du jeu de données avec **head**

```
# charger les données  
str(mtcars)  
#aperçu du début du dataframe  
head(mtcars)
```

Les "datas" est les données sont organisées en observations(lignes) décrites en fonction de variables.

Les données

Le data frame mtcars est chargé dans l'environnement, et on peut y accéder et le visualiser:



VARIABLE QUANTITATIVE:

Pour une variable quantitative, on peut calculer les statistiques de bases suivantes:

| Fonction | Opération |
|-------------|----------------------------------------|
| sum(x) | somme |
| mean(x) | moyenne |
| var(x) | variance |
| sd(x) | écart-type |
| min(x) | minimum |
| max(x) | maximum |
| median(x) | médiane |
| quantile(x) | quantiles à 0, 25%,50%,75% et 100% |
| length(x) | Nombre d'observations pour la variable |

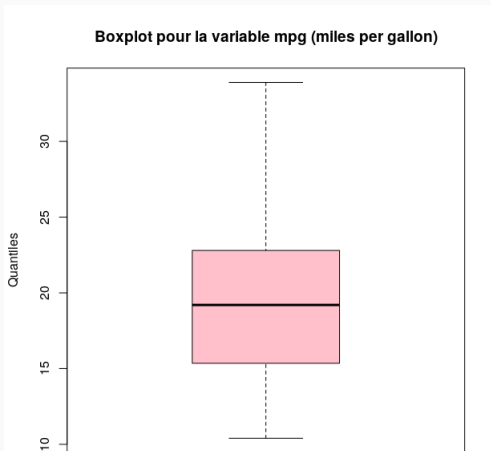
La fonction **summary** donne les statistiques de base pour chaque variable :

Pour les variables quantitatives : Minimum, maximum, moyenne, médiane, 1er quartile, 3ème quartile.

Pour les variables qualitatives : Nombre d'observations par classe.

VARIABLE QUANTITATIVE-REPRÉSENTATION GRAPHIQUE:

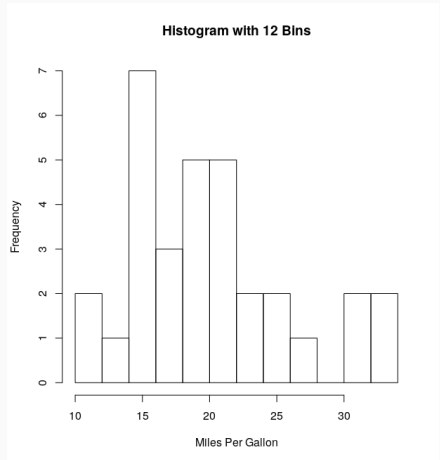
Le **boxplot** est souvent utilisé pour résumer graphiquement des statistiques descriptives sur chaque variable.



Analyse univari  e

VARIABLE QUANTITATIVE-REPR  SENTATION GRAPHIQUE:

L'histogramme (**hist**) permet de regarder la distribution d'une variable quantitative.



VARIABLE QUALITATIVE:

Pour un résumé pertinent d'une variable qualitative, on trace un tableau des effectifs de chaque classe. La fonction **table** appliquée à un facteur (ou à un vecteur de caractères) comptent les occurrences des différents niveaux.

VARIABLE QUALITATIVE-REPRÉSENTATION GRAPHIQUE:

Barplot , camembert ... (voir TP)

L'objectif est de faire apparaître d'éventuelles liaisons entre deux variables X et Y décrivant une même population (mesurées sur un même individu).

2 VARIABLES QUANTITATIVES:

- Covariance de deux variable: écarts conjoints par rapport à leurs moyennes respectives.

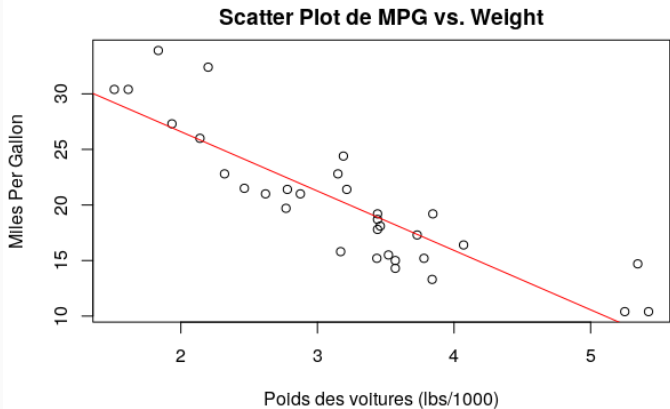
- Coefficient de corrélation linéaire de Pearson (alternative: Spearman sur les rangs):

intensité de dépendance linéaire entre deux variables. Valeur entre $[-1,1]$: $+1(-1)$ si dépendance linéaire positive(négative) et 0 si indépendance linéaire.

Analyse bivariee

2 VARIABLES QUANTITATIVES - REPRÉSENTATION GRAPHIQUE:

Pour représenter graphiquement deux variables quantitatives: on utilisera un nuage de point(scatter plot). Figure à réaliser en TP



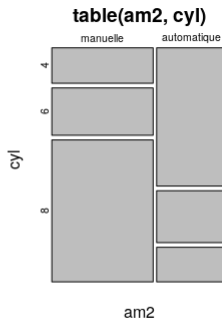
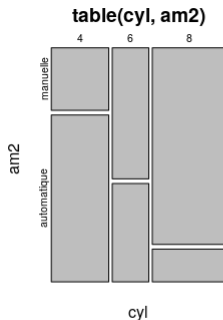
Analyse bivariable

2 VARIABLES QUALITATIVES:

On peut créer une table de contingence avec la fonction
`mosaicplot(table(varqual1,varqual2))`

2 VARIABLES QUALITATIVES- REPRÉSENTATION GRAPHIQUE:

Figure à réaliser en TP: `mosaicplot(table(varqual1,varqual2))`



VARIABLE QUANTITATIVE VS VARIABLE QUALITATIVE:

Pour calculer une fonction f sur les valeurs d'une variable y pour chaque niveau d'un facteur x :

```
> tapply(X=y, INDEX=x, FUN=f)
#Exemple: moyenne de "miles per gallon" en fonction des 3
           catégories de volumes de cylindres
>tapply(mtcars$mpg, mtcars$cyl, mean)
```

Analyse bivariee

VARIABLE QUANTITATIVE VS VARIABLE QUALITATIVE - REPRÉSENTATION GRAPHIQUE

