Initation à R

Pour Master 2 MBFA-Analyse des risques de marchés

Sonia Tieo

Doctorante au CEFE-CRNS Bioinformatique et Ecologie Comportementale

Contact et Lien du cours/TP

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

https://github.com/soniamaitieo/CoursM2_MBFA

Contact: sonia.tieo@cefe.cnrs.fr

Sommaire

Intro

R: un logiciel et un langage

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.

Installation

Pour installer R :

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

Choisissez votre plateforme....

RStudio

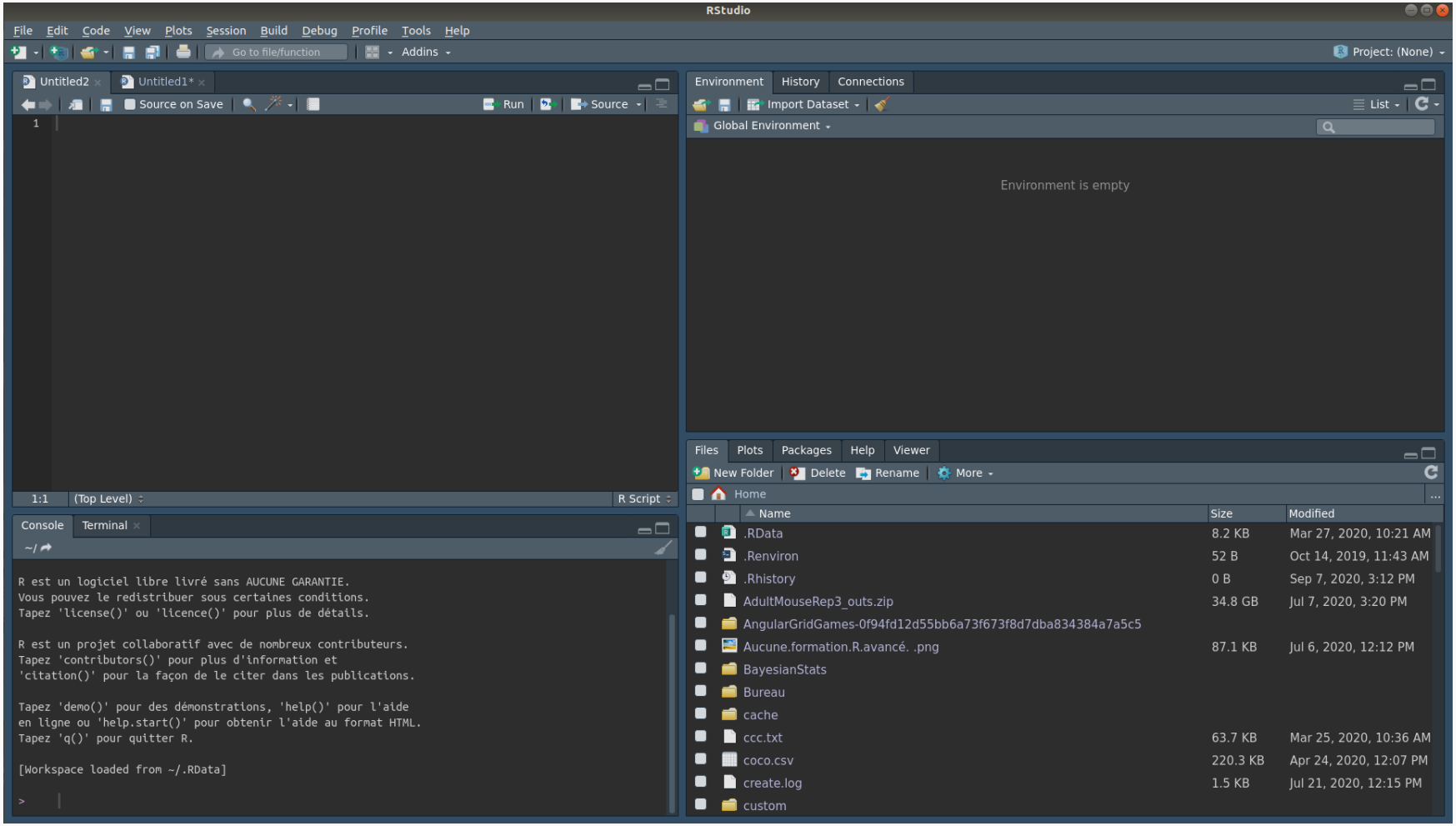
R est disponible sous de nombreux systèmes d'exploitation.
(Windows, Mac OS X ou Linux)

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


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled2
Untitled1*
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Console
Terminal
~/
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.
[Workspace loaded from ~/.RData]
>
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
Name
Size
Modified
.RData 8.2 KB Mar 27, 2020, 10:21 AM
.Renviron 52 B Oct 14, 2019, 11:43 AM
.Rhistory 0 B Sep 7, 2020, 3:12 PM
AdultMouseRep3_outs.zip 34.8 GB Jul 7, 2020, 3:20 PM
AngularGridGames-0f94fd12d55bb6a73f673f8d7dba834384a7a5c5
Aucune.formation.R.avancé. .png 87.1 KB Jul 6, 2020, 12:12 PM
BayesianStats
Bureau
cache
ccc.txt 63.7 KB Mar 25, 2020, 10:36 AM
coco.csv 220.3 KB Apr 24, 2020, 12:07 PM
create.log 1.5 KB Jul 21, 2020, 12:15 PM
custom

RStudio

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.
```

Ouverture de session

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

Environnement de travail

Conseil

Pour chaque projet: utiliser un répertoire de travail pour sauvegarder tout ce qui est lié au projet: jeu(x) de donnéees, 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

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:

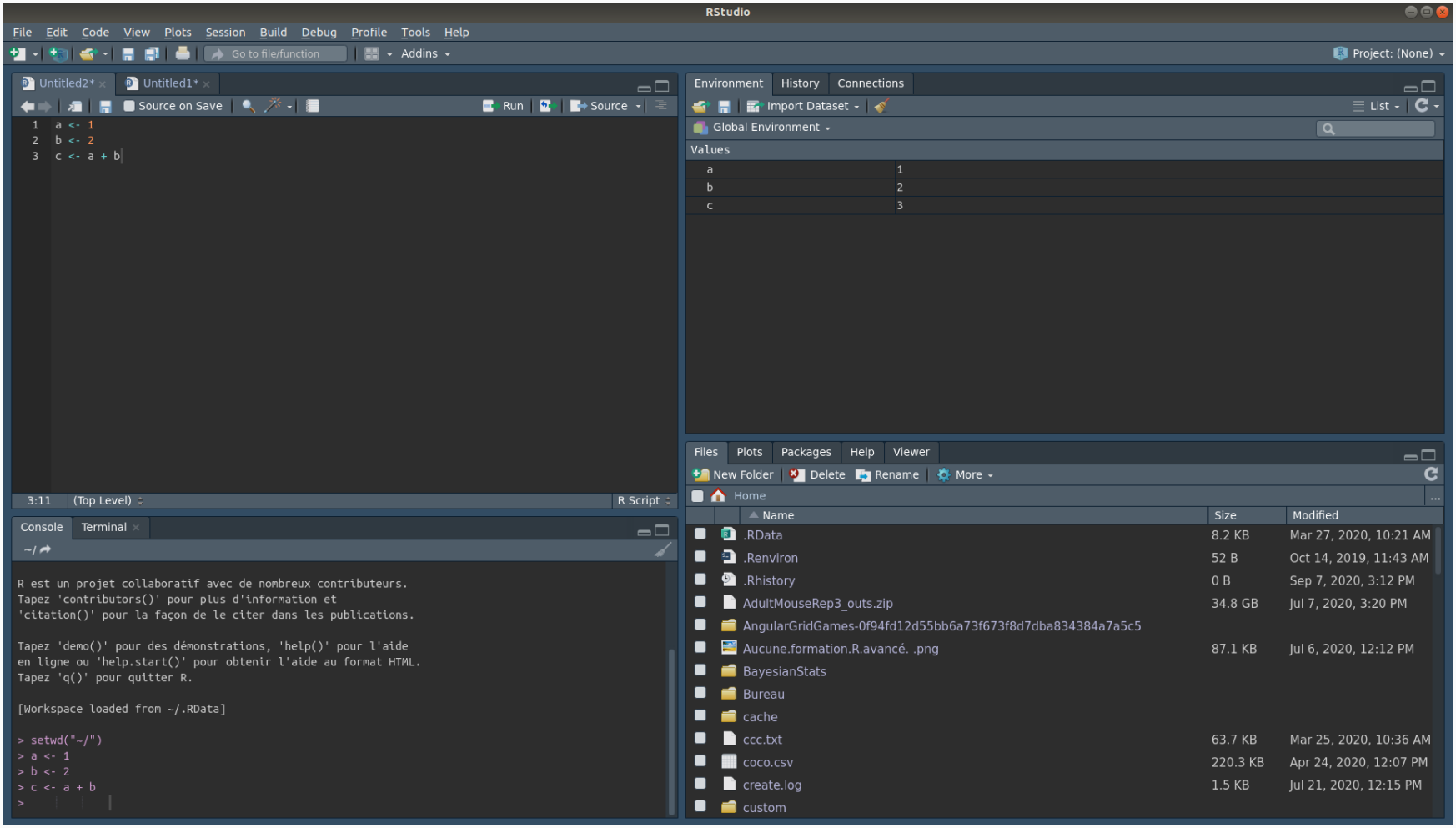
```
> save.image()
```

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


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled2*
Untitled1*
Source on Save
Run
Source
1 a <- 1
2 b <- 2
3 c <- a + b
3:11
(Top Level)
R Script
Console
Terminal
~/
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.
[Workspace loaded from ~/.RData]
> setwd("~/")
> a <- 1
> b <- 2
> c <- a + b
>
Environment
History
Connections
Import Dataset
List
Global Environment
Values
a 1
b 2
c 3
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
Name
Size
Modified
.RData 8.2 KB Mar 27, 2020, 10:21 AM
.Renviron 52 B Oct 14, 2019, 11:43 AM
.Rhistory 0 B Sep 7, 2020, 3:12 PM
AdultMouseRep3_outs.zip 34.8 GB Jul 7, 2020, 3:20 PM
AngularGridGames-0f94fd12d55bb6a73f673f8d7dba834384a7a5c5
Aucune.formation.R.avancé. .png 87.1 KB Jul 6, 2020, 12:12 PM
BayesianStats
Bureau
cache
ccc.txt 63.7 KB Mar 25, 2020, 10:36 AM
coco.csv 220.3 KB Apr 24, 2020, 12:07 PM
create.log 1.5 KB Jul 21, 2020, 12:15 PM
custom

Packages

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 créer 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 caratères 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 appelés par des vnoms 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 données

Structuration des objets:

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

Les vecteurs

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] 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: séléction

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 1err élément de la liste
[1] 5 4 3 2 1
```

Manipulation vecteurs: séléction

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: séléction

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 séléction 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: séléction

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 appélations 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
```

Les matrices

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

Les matrices

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
```

Les matrices

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 transformer un vecteur en matrice
> v <- 1:5
> vmatrix <- as.matrix(v)
> vmatrix
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
[5,]    5
```

Les matrices

Création d'une matrice par concaténation des vecteurs:

Coller horizontalement des vecteurs **rbind**

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

Les matrices

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 connaitre le nombre total d'éléments dans la matrice **length**

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

Dimension d'une matrice:

Pour connaitre 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,]
prenoms    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 nommés, alors il est aussi possible de passer par ces noms pour les sélectionner. Pour connaitre le nom des lignes et des colonnes, on peut utiliser les fontions **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

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 consitué 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
```

Les facteurs

Attributs des facteurs:

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

Les listes

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.

Les listes

Création de listes:

On utilise les éléments crées dans les exemples précédents:

```
> num_etu <-1:7
> prenoms <- 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, prenoms, 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
```

Les listes

Attention: les composantes de la liste crée 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

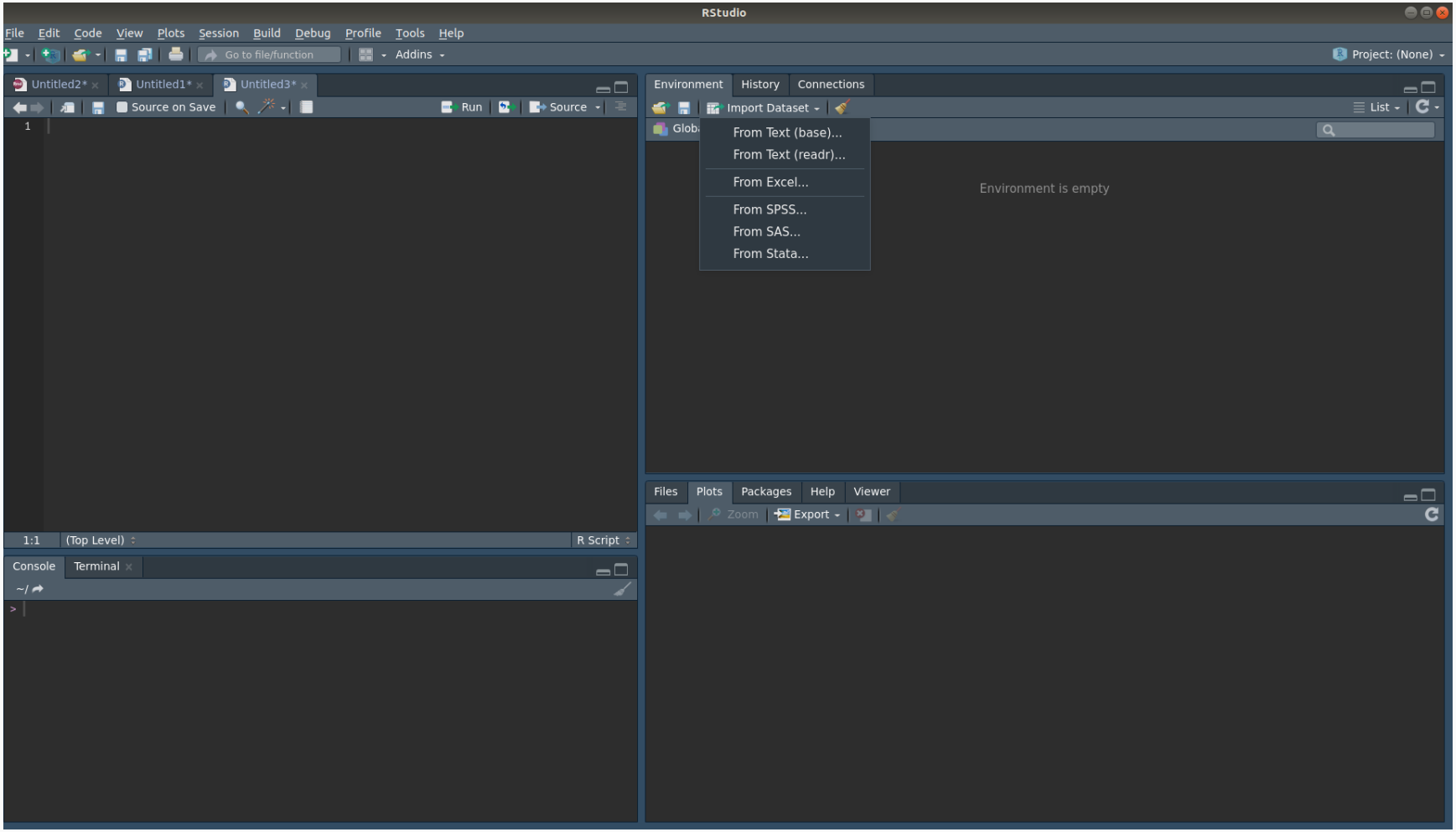
Création de dataframes: c

```
> 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
```

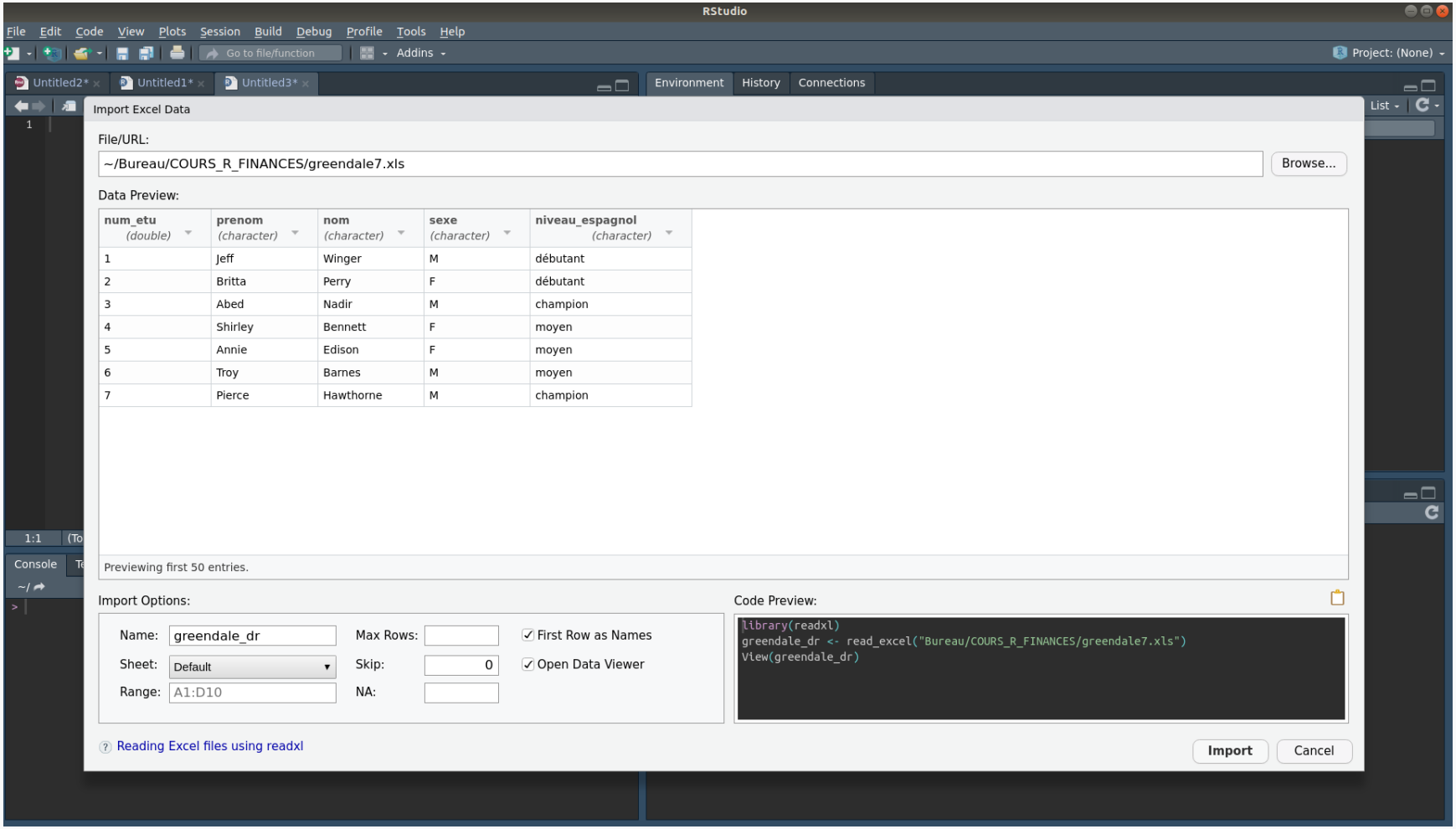
Importations

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 possibités 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

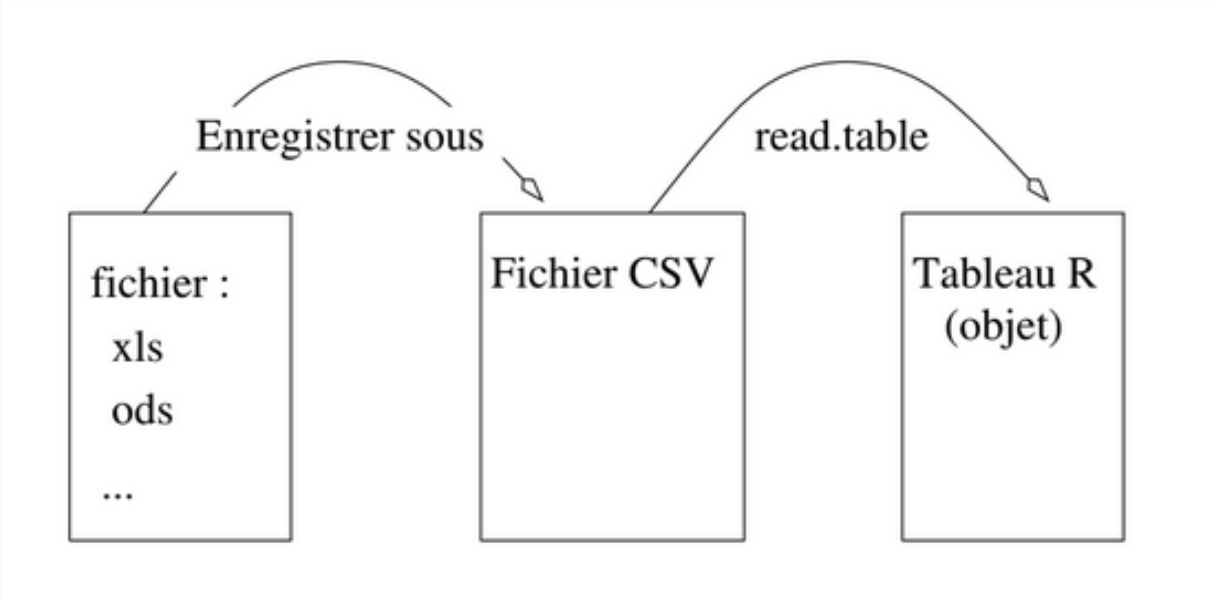


En ligne de commande:

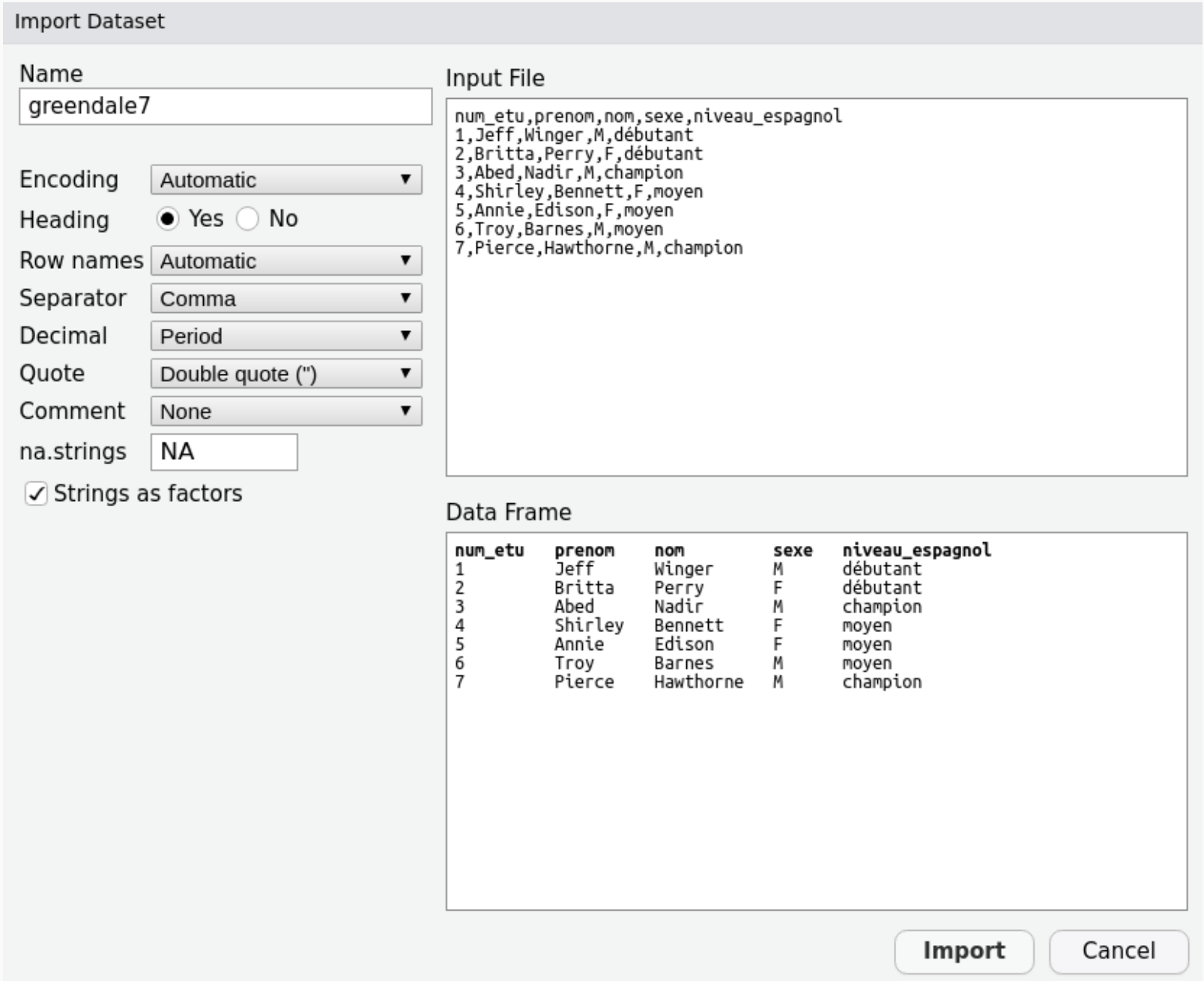
```
greendale7 <- read_excel("Bureau/COURS_R_FINANCES/greendale7.
    xls")
```

Importations CSV

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



Importations CSV



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)
```

Manipulation de Data Frame

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)
```

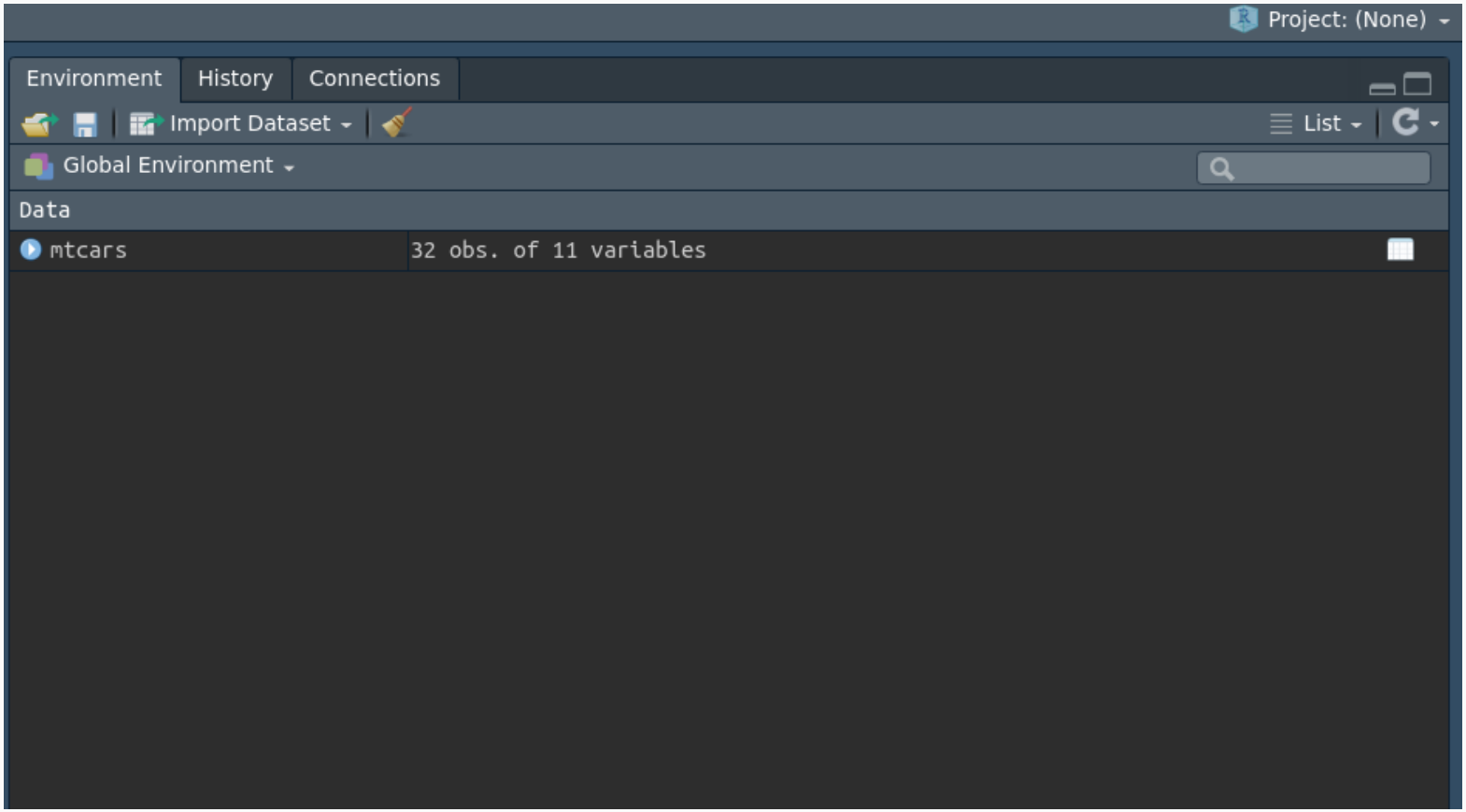
On peut avoir une idée de la classe et du contenu avec **str** et oberver le début du jeu de données avec **head**

```
# charger les données
str(mtcars)
#apercu 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:



Analyse univariée

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

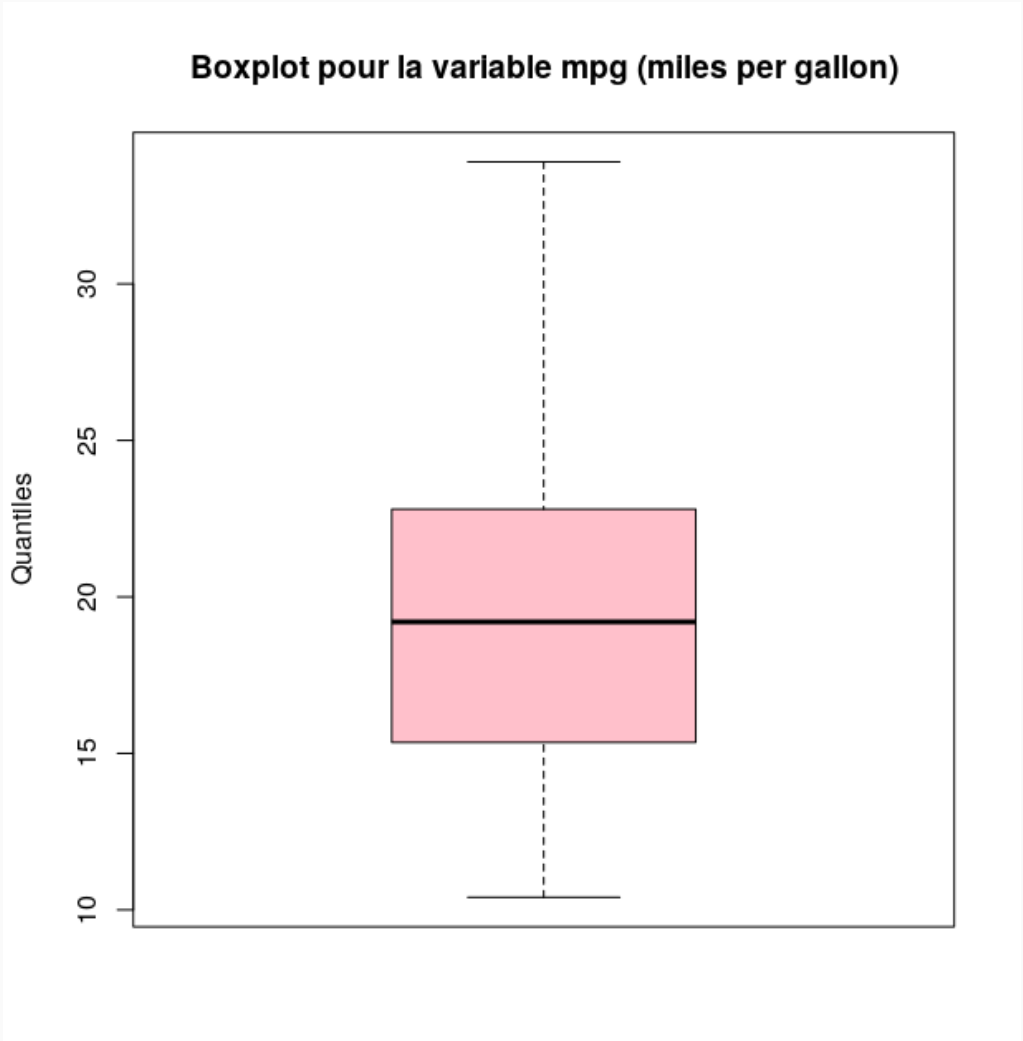
La fonction **summary** donne les statistiques de base pour chaque variable :

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

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

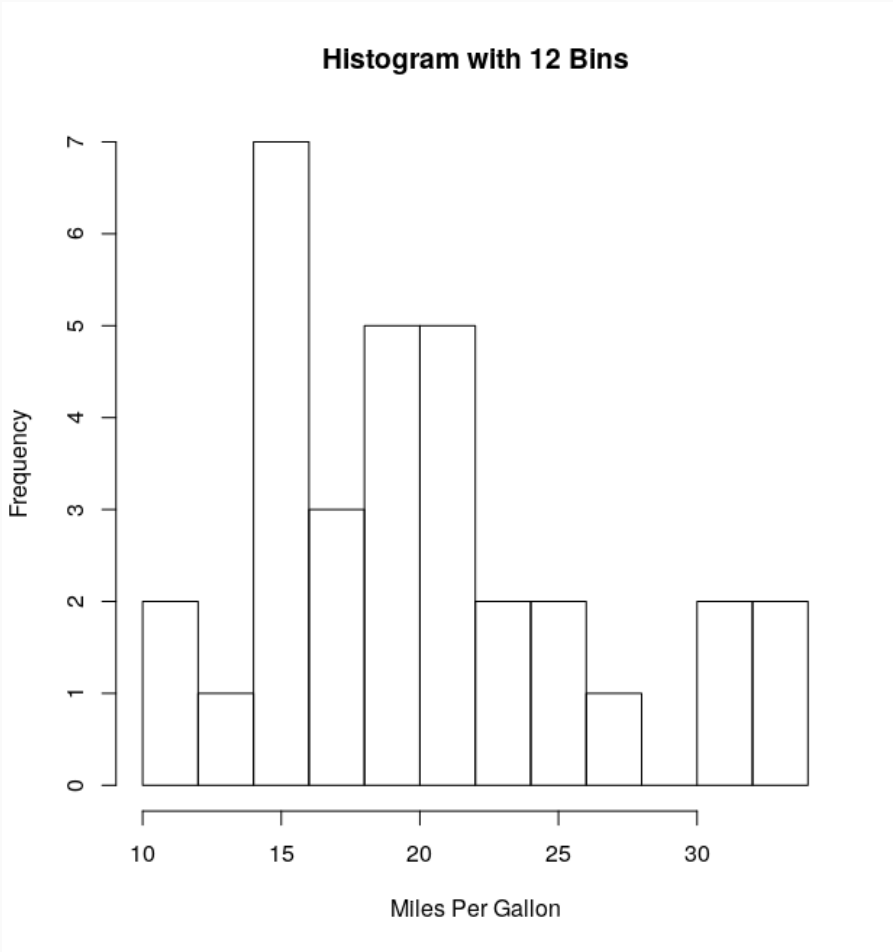
Variable quantitative-Représentation graphique:

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



Variable quantitative-Représentation graphique:

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



Analyse univariée

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)

Analyse bivariée

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

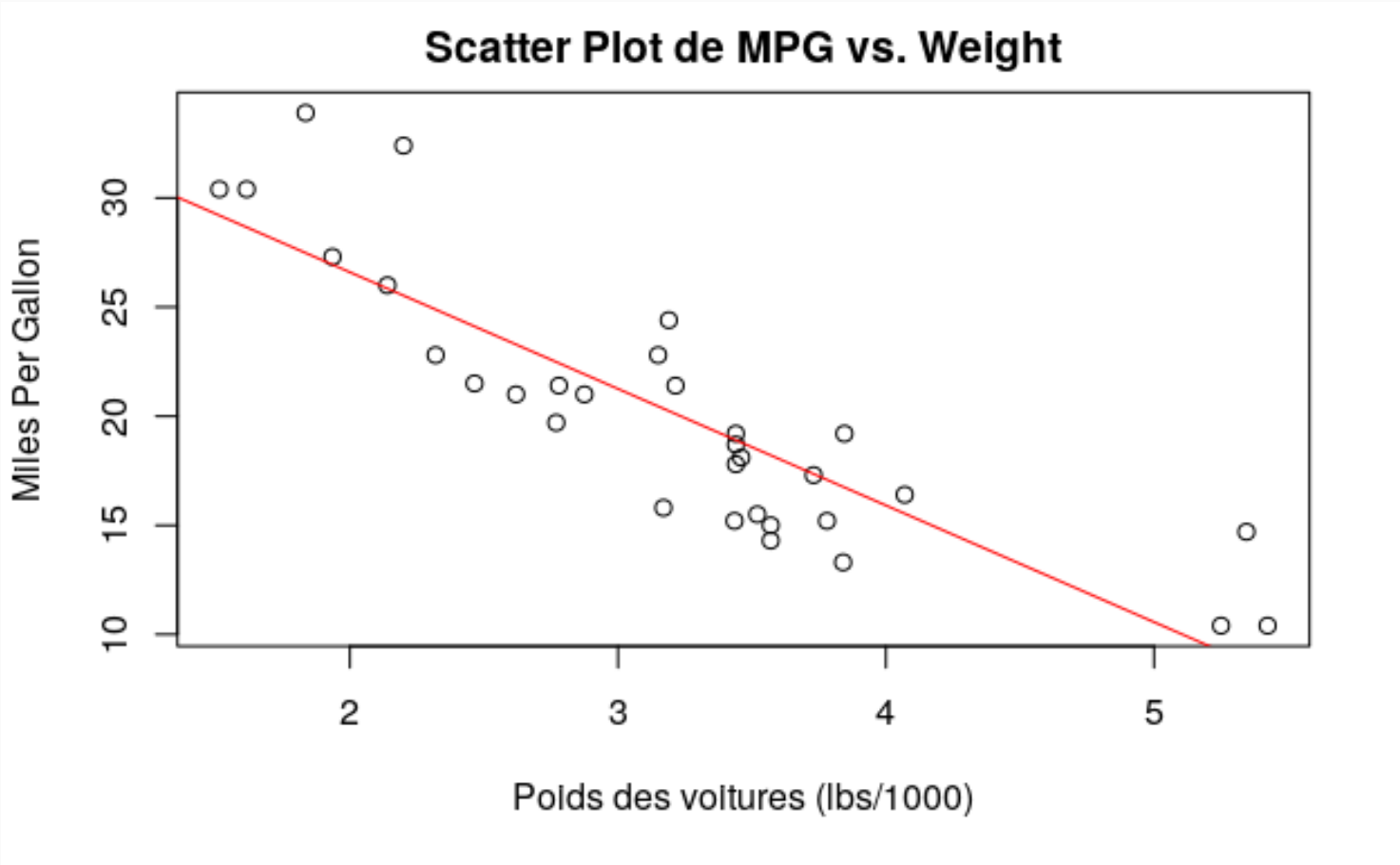
2 Variables quantitatives:

- 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.

2 Variables quantitatives - Représentation graphique:

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



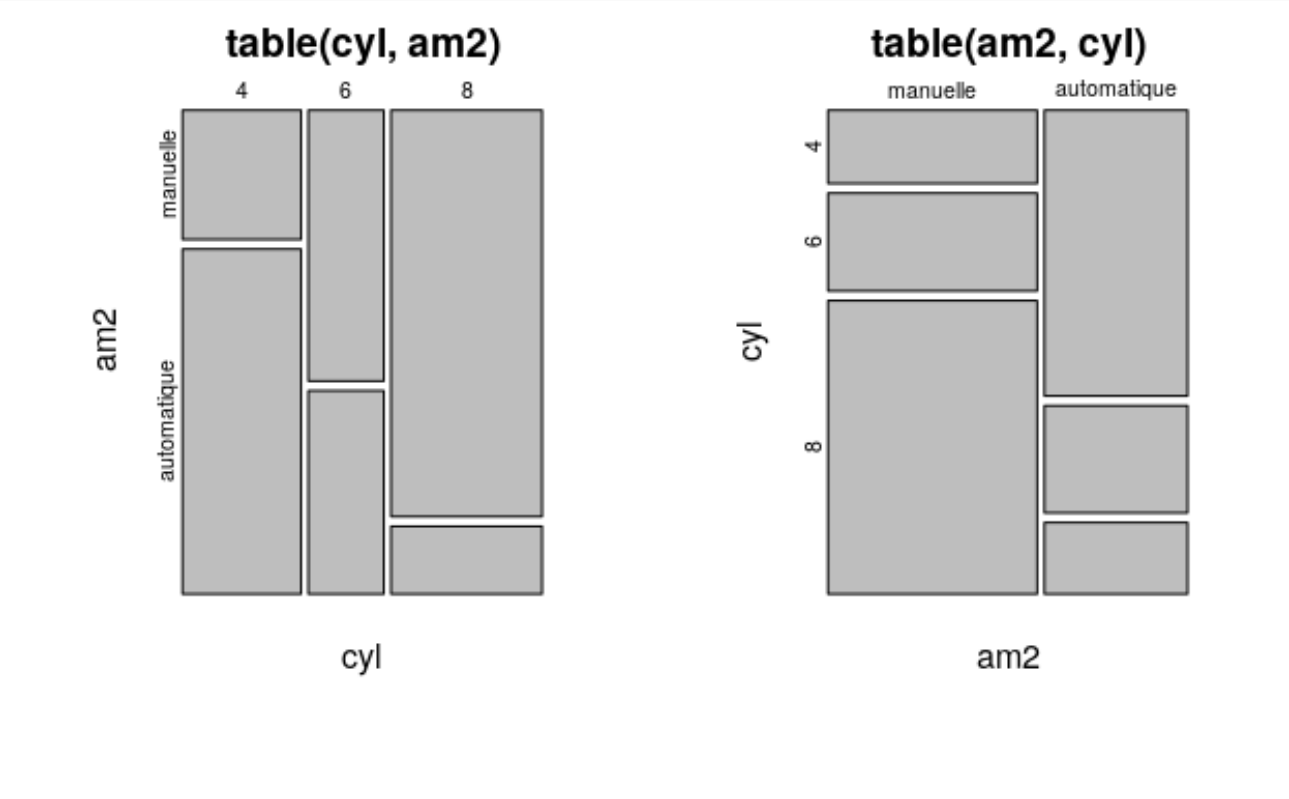
Analyse bivariée

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))**

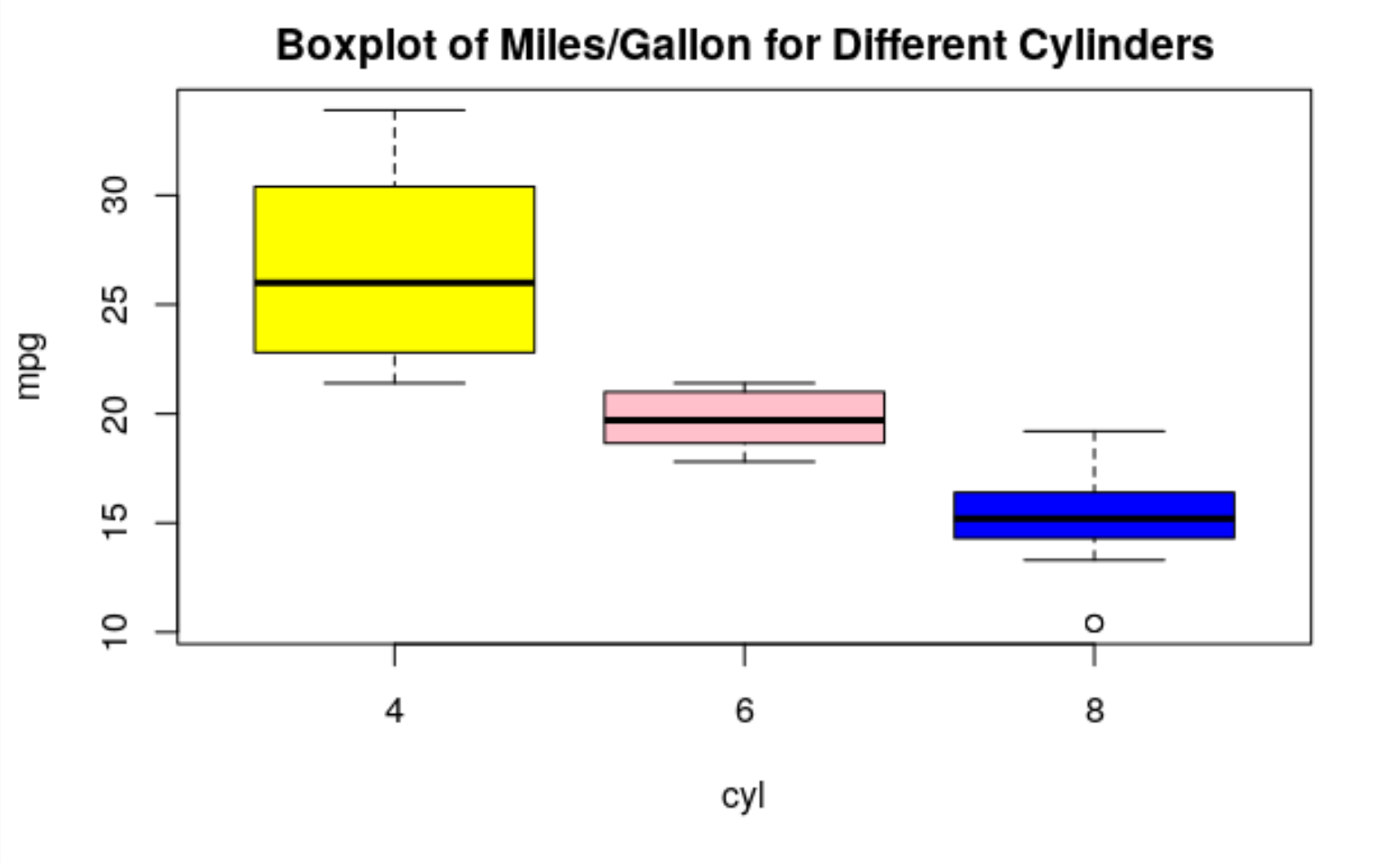


Variable quantitative vs variable qualitative:

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)
```

Variable quantitative vs variable qualitative - Représentation graphique



Séance TP3: Analyse Financière avec R

Séance TP5: Machine Learning en finance

Machine learning en bref

Le Machine Learning ou apprentissage automatique a connu une explosion dans son utilisation et de son application à des problèmes d'automatisation dans divers domaines.

C'est une sous-catégorie de l'intelligence artificielle.

Principe: On laisse les algorithmes découvrir des " patterns ", à savoir des motifs récurrents, dans les ensembles de données. Ces données peuvent être des chiffres, des mots, des images, des statistiques ... Les algorithmes apprennent et améliorent leurs performances dans l'exécution d'une tâche spécifique.

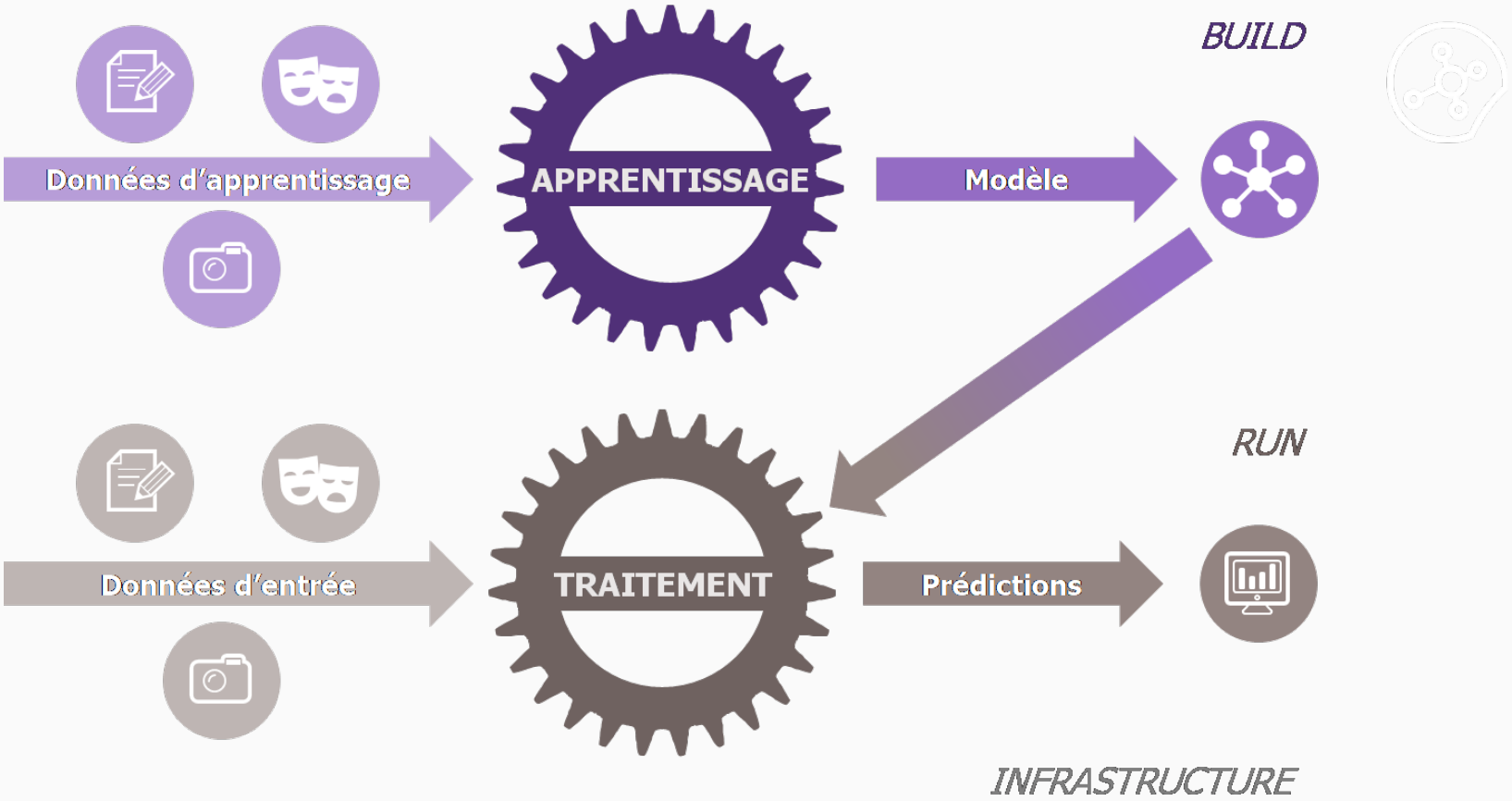
Pour résumer, les algorithmes de Machine Learning apprennent de manière autonome à effectuer une tâche ou à réaliser des prédictions à partir de données et améliorent leurs performances au fil du temps. Une fois entraîné, l'algorithme pourra retrouver les patterns dans de nouvelles données.

Pourquoi le Machine learning en finance ?

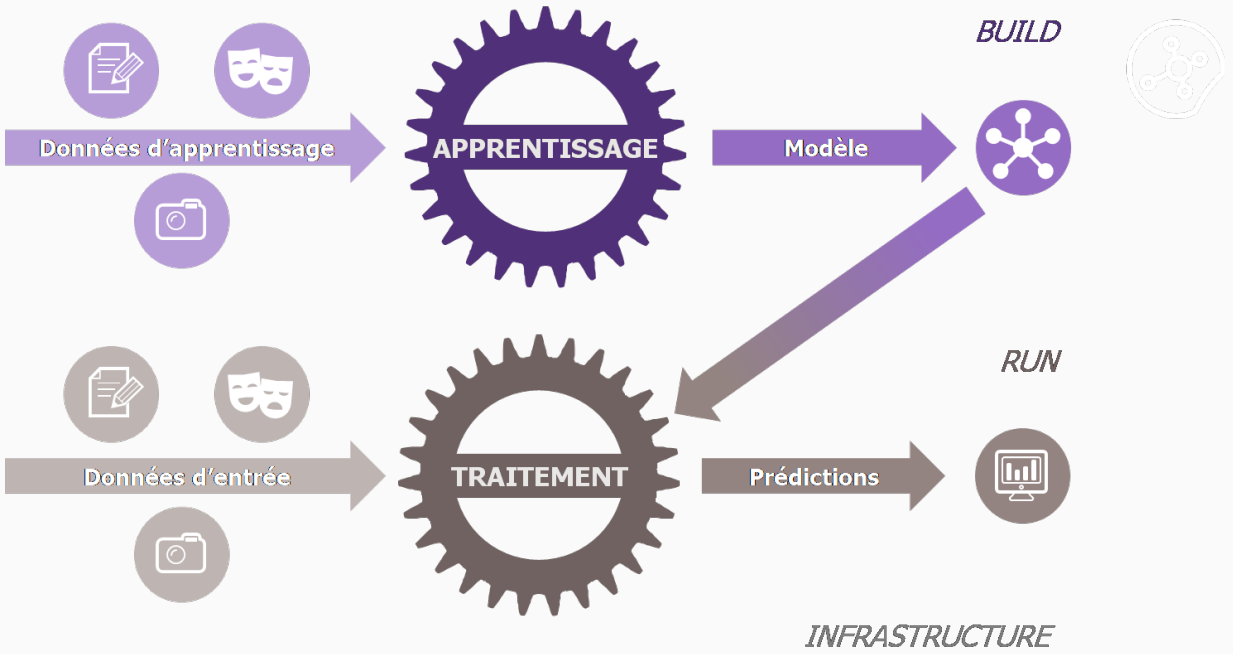
”Normalement, un bon trader prédit le prix des actions et achète une action si son prix va augmenter dans un avenir proche, et revend une action si son prix s’apprête à diminuer. Cependant, la volatilité expliquée ci-dessus rend la tâche des traders bien difficile, car il est impossible de prédire avec 100 % d’exactitude l’évolution du prix d’une action. C’est dans ce cadre-là que les investisseurs et gestionnaires de porte-feuille ont commencé à se pencher vers le Machine Learning. Bien qu’il soit difficile de remplacer les compétences acquises par un trader expérimenté, un algorithme de prédiction précis peut également engendrer des bénéfices pour des entreprises d’investissements, mettant en évidence une relation entre la précision de l’algorithme et les profits de son utilisation (Shah, 2007 and DePaepe; 2021).”

Comment fonctionne le Machine Learning ?

Le développement d'un modèle de Machine Learning repose sur quatre étapes principales.



Etape 1: Sélectionner et préparer un ensemble de données d'apprentissage



Ces données seront utilisées pour nourrir le modèle de ML pour lui apprendre à résoudre le problème pour lequel il est conçu.

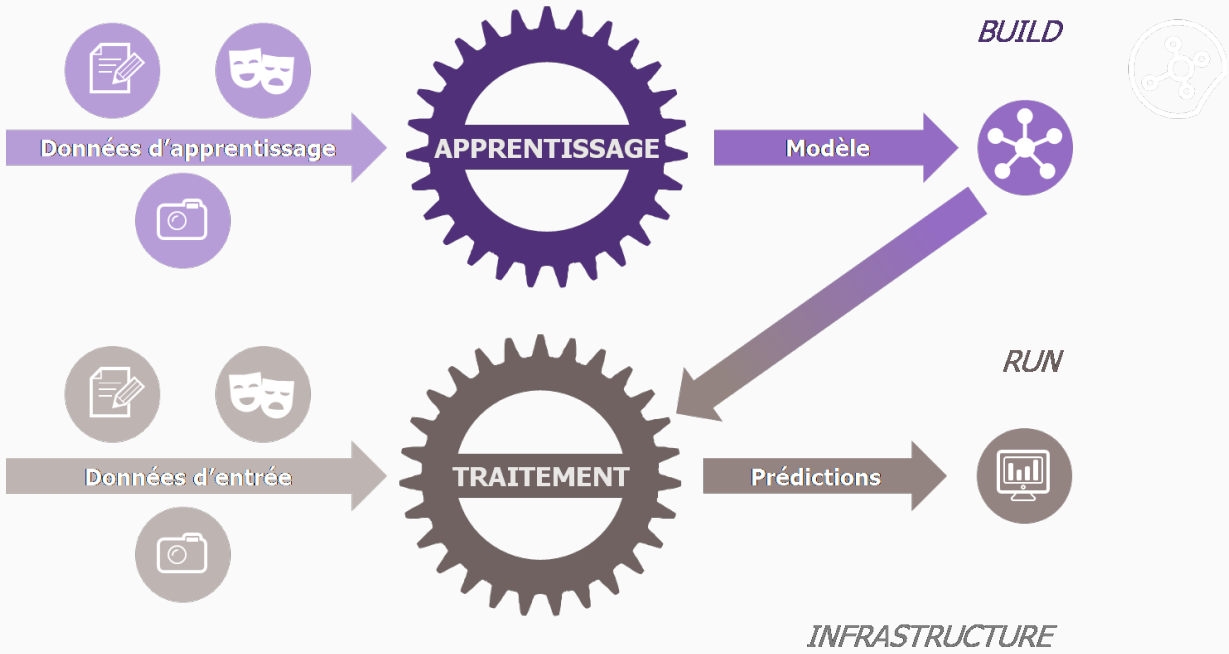
Etape 1: Sélectionner et préparer un ensemble de données d'apprentissage

Les données peuvent être étiquetées, afin d'indiquer au modèle les caractéristiques qu'il devra identifier. Elles peuvent aussi être non étiquetées, et le modèle devra repérer et extraire les caractéristiques récurrentes de lui-même.

Dans les deux cas, les données doivent être soigneusement préparées, organisées et nettoyées. Dans le cas contraire, l'entraînement du modèle de Machine Learning risque d'être biaisé. Les résultats de ses futures prédictions seront directement impactés.

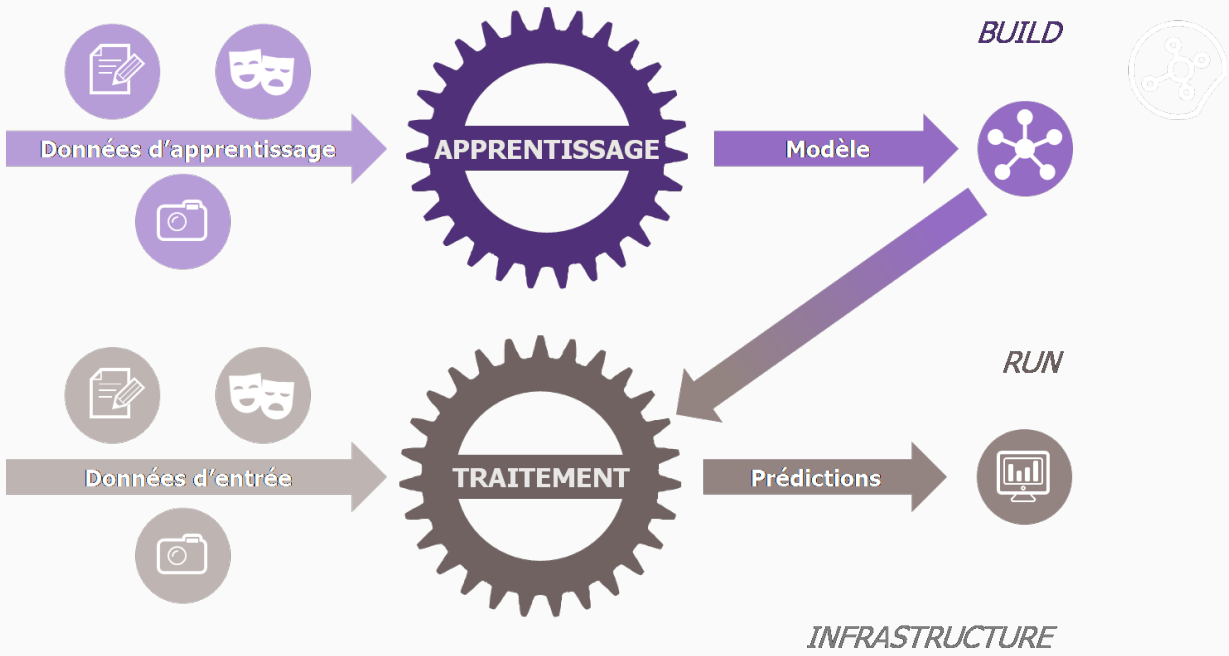
Exemple: Modèle qui doit apprendre à classifier les chats et les chiens (étiquetés) OU le modèle doit identifier la variabilité entre des images de chiens ou de chats(sur des photos non étiquetés).

Etape 2: Sélectionner un algortithme



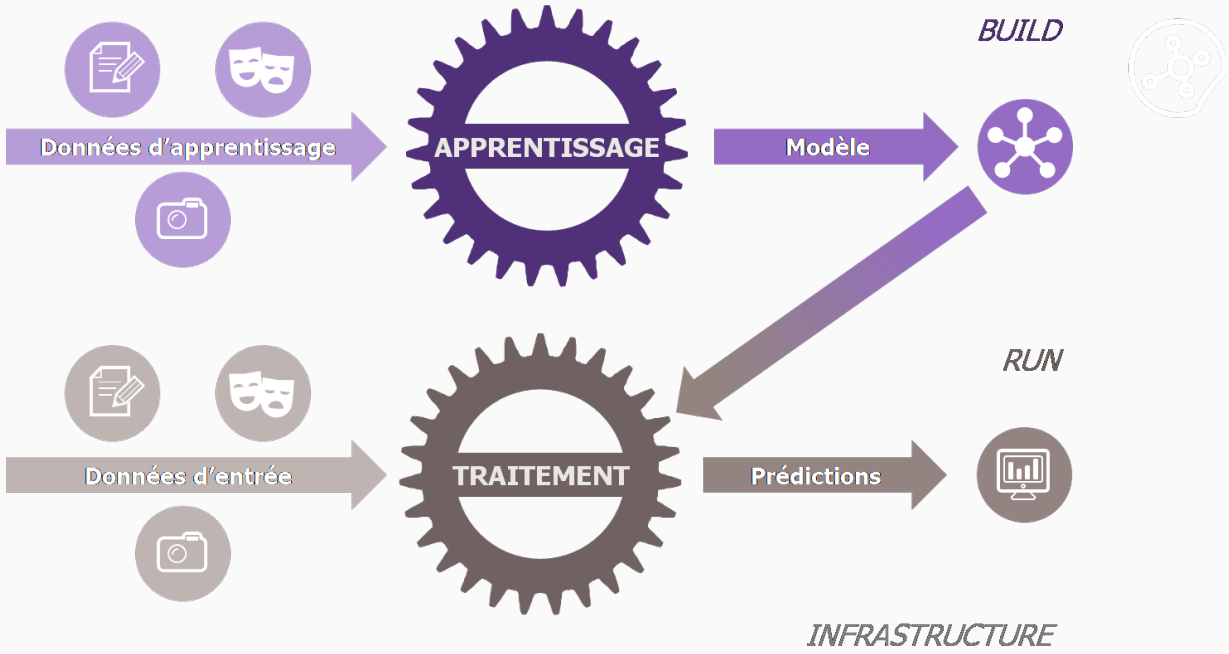
Le type d'algorithme à utiliser dépend du type et du volume de données d'entraînement et du type de problème à résoudre.
entraînement de l'algorithme

Etape 3: Entraînement de l'algorithme



De manière itérative, des variables sont exécutées à travers l'algorithme, et les résultats sont comparés avec ceux qu'il aurait du produire. Les " poids " et le biais peuvent ensuite être ajustés pour accroître la précision du résultat. On exécute ensuite de nouveau les variables jusqu'à ce que l'algorithme produise le résultat correct la plupart du temps. L'algorithme, ainsi entraîné, est le modèle de Machine Learning.

Etape 4: Utilisation et l'amélioration du modèle



On utilise le modèle sur de nouvelles données, dont la provenance dépend du problème à résoudre.Par exemple, un modèle de Machine Learning conçu pour détecter les spams sera utilisé sur des emails.

Apprentissage supervisé vs. apprentissage non-supervisé

Apprentissage supervisé: La majorité des apprentissages automatiques utilisent un apprentissage supervisé où on a des variables d'entrée (x) et une variable de sortie (Y). On utilise un algorithme pour apprendre la fonction de mapping de l'entrée à la sortie.

$$Y = f(X)$$

Le but est d'avoir une fonction qui peut prédire les variables de sortie (Y) pour ces données pour des nouvelles données d'entrée (x). Cette méthode nécessite moins de données d'entraînement et facilite le processus d'entraînement puisque les résultats du modèle peuvent être comparés avec les données déjà étiquetées. Mais, l'étiquetage des données peut être coûteux. Un modèle peut aussi être biaisé à cause des données d'entraînement, ce qui impactera ses performances par la suite lors du traitement de nouvelles données.

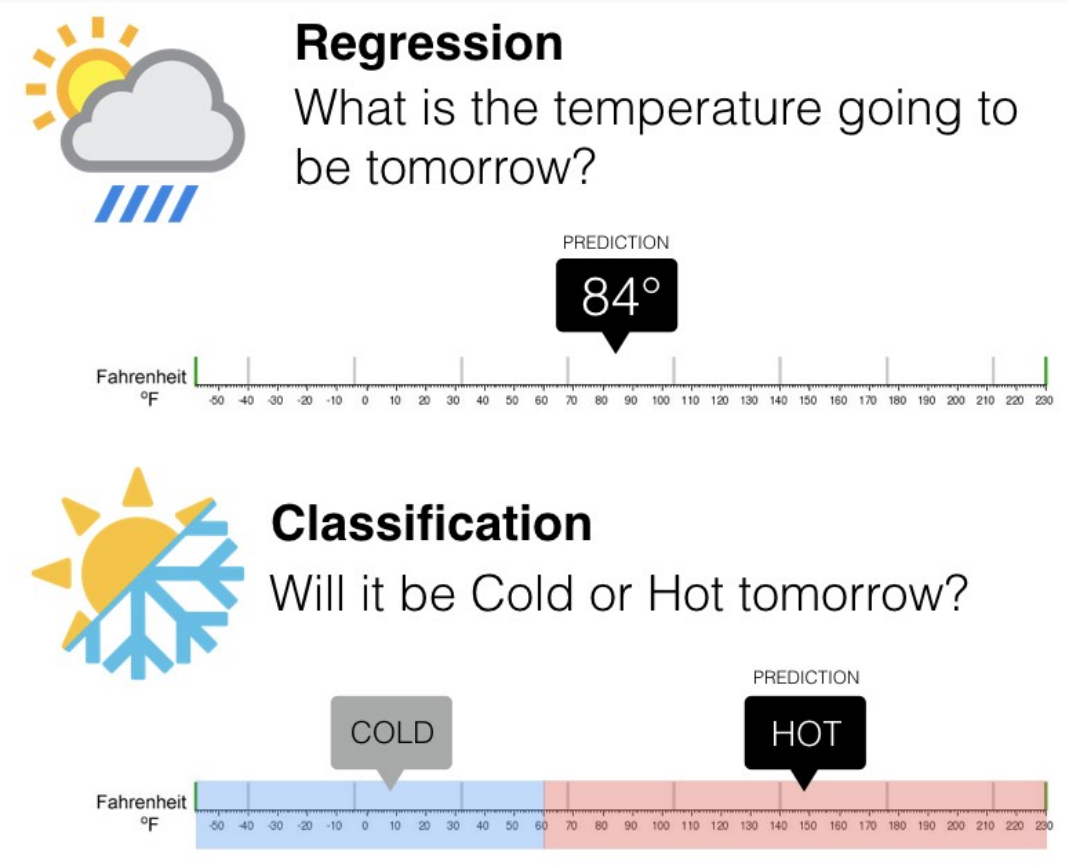
Apprentissage supervisé vs. apprentissage non-supervisé

Apprentissage non-supervisé:

Les données n'ont pas d'étiquettes. La machine se contente d'explorer les données à la recherche d'éventuelles patterns. Elle ingère de vastes quantités de données, et utilise des algorithmes pour en extraire des caractéristiques pertinentes requises pour étiqueter, trier et classifier les données en temps réel sans intervention humaine.

Plutôt que d'automatiser les décisions et les prédictions, cette approche permet d'identifier les patterns et les relations que les humains risquent de ne pas identifier dans les données. Cette technique n'est pas très populaire, car moins simple à appliquer.

Algorithme de classification vs algorithme de régression



Classification: Avoir une fonction de relation entre les données d'entrées et la variable de sortie discrète.

Régression: Les variables de sorties seront continues.

Algorithme de classification vs algorithme de régression

”Prenons l’exemple où il faut classer les futurs mouvements du prix d’une action. L’algorithme de classification pourrait avoir deux variables de sorties discrètes, 1 si le prix de l’action augmente et -1 si celui-ci diminue. Alors que l’algorithme de régression va proposer en sortie des évolutions du prix de l’action en pourcentages continus (Rasekhschaffe and Jones, 2019)”

Évaluez un algorithme de classification qui retourne des valeurs binaires

Sur des données étiquetées, on veut prédire à quelle classe un objet appartient. dans un premier temps, on s'intéresse à une classification binaire. Exemple: Dire si une image représente un chien. Si oui, on dit que cette image est positive ; sinon, qu'elle est négative.

Attention

On ne peut pas utiliser le nombre d'erreurs comme mesure de performance car ce n'est pas le seul critère et toutes les erreurs ne se valent pas.

Évaluez un algorithme de classification qui retourne des valeurs binaires

Matrice de Confusion: Prenons un algorithme qui prédit s'il y a un incendie à un endroit donné. Déclencher une alerte incendie quand il n'y a pas le feu est moins grave que de ne pas déclencher d'alerte quand l'appartement est en flamme.

Prédire un incendie quand il n'y en a pas, ne pas prédire un incendie quand il y en a un, ne pas en prédire quand il n'y en a pas... On s'y perd vite ! Pour y voir plus clair, on utilise une matrice de confusion.

		Classe réelle	
		-	+
Classe prédite	-	True Negatives *(vrais négatifs)*	False Negatives *(faux négatifs)*
	+	False Positives *(faux positifs)*	True Positives *(vrais positifs)*

Évaluez un algorithme de classification qui retourne des valeurs binaires

		Classe réelle	
		-	+
Classe prédite	-	True Negatives *(vrais négatifs)*	False Negatives *(faux négatifs)*
	+	False Positives *(faux positifs)*	True Positives *(vrais positifs)*

Appelons ”positive” la classe correspondant à un incendie et ”négative” l’autre. Si on prédit un incendie quand il y en a bien un, on fait une prédiction ”positif” qui est correcte, c’est un vrai positif. Si par contre cette prédiction est incorrecte, il s’agit d’un faux positif. Et ainsi de suite. On appelle aussi parfois ”erreur de type I” les faux positifs, et ”erreur de type II” les faux négatifs.

```
MC = table(Vecteur_donnees, Vecteur_donnees_predites)
```

Évaluez un algorithme de classification qui retourne des valeurs binaires

		Classe réelle	
		-	+
Classe prédite	-	**T**rue **N**egatives *(vrais négatifs)*	**F**alse **N**egatives *(faux négatifs)*
	+	**F**alse **P**ositives *(faux positifs)*	**T**rue **P**ositives *(vrais positifs)*

À partir de la matrice de confusion on peut dériver tout un tas de critères de performance.

RECALL: Le rappel (”recall”), ou sensibilité (”sensitivity”), est le taux de vrais positifs = la proportion de positifs correctement identifiés. C’est la capacité de notre modèle à détecter tous les incendies.

$Rappel = TP/(TP + FN)$

Évaluez un algorithme de classification qui retourne des valeurs binaires

		Classe réelle	
		-	+
Classe prédite	-	**T**rue **N**egatives *(vrais négatifs)*	**F**alse **N**egatives *(faux négatifs)*
	+	**F**alse **P**ositives *(faux positifs)*	**T**rue **P**ositives *(vrais positifs)*

Precision: La précision, c'est la proportion de prédictions correctes parmi les points que l'on a prédits positifs. C'est la capacité de notre modèle à ne déclencher d'alarme que pour un vrai incendie.

$Precision = TP/(TP + FP)$

Évaluez un algorithme de classification qui retourne des valeurs binaires

		Classe réelle	
		-	+
Classe prédite	-	True Negatives *(vrais négatifs)*	False Negatives *(faux négatifs)*
	+	False Positives *(faux positifs)*	True Positives *(vrais positifs)*

F-MESURE: Pour évaluer un compromis entre rappel et précision, on peut calculer la "F-mesure", qui est leur moyenne harmonique.

$Fmesure = (2PrecisionRappel)/(Precision + Rappel) = 2TP/(2TP + FP + FN)$

Évaluez un algorithme de classification qui retourne des valeurs binaires

		Classe réelle	
		-	+
Classe prédite	-	**T**rue **N**egatives *(vrais négatifs)*	**F**alse **N**egatives *(faux négatifs)*
	+	**F**alse **P**ositives *(faux positifs)*	**T**rue **P**ositives *(vrais positifs)*

Precision: La précision, c'est la proportion de prédictions correctes parmi les points que l'on a prédits positifs. C'est la capacité de notre modèle à ne déclencher d'alarme que pour un vrai incendie.

$Precision = TP/(TP + FP)$

Évaluez un algorithme de classification qui retourne des valeurs binaires

		Classe réelle	
		-	+
Classe prédite	-	**T**rue **N**egatives *(vrais négatifs)*	**F**alse **N**egatives *(faux négatifs)*
	+	**F**alse **P**ositives *(faux positifs)*	**T**rue **P**ositives *(vrais positifs)*

SPECIFICITY: Spécificité est le taux de vrais négatifs, autrement dit la capacité à détecter toutes les situations où il n'y a pas d'incendie. C'est une mesure complémentaire de la sensibilité.

$$Specificity = TN/(FP + TN)$$

Évaluez un algorithme de classification qui retourne des valeurs binaires

Toutes ces mesures de performance sont disponibles sur R:

https://rdrr.io/cran/caret/man/recall.html

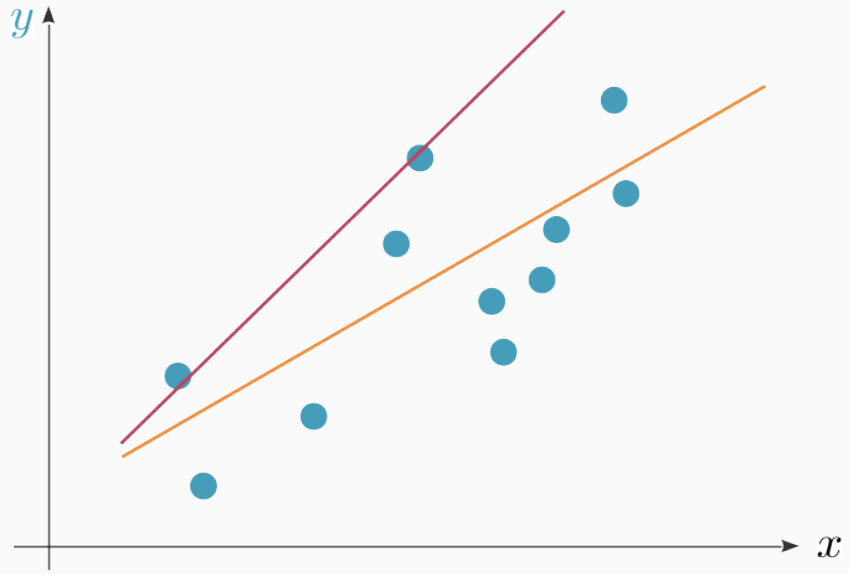
Évaluer la performance des algorithmes de régression

Il s'agit ici d'utiliser des données étiquetées pour associer une valeur numérique à un objet, comme par exemple de prédire a température de demain.

Pour la classification, on compte le nombre d'erreurs de prédiction mais pour la régression, ce n'est pas adapté: à quelle précision numérique peut-on dire qu'une prédiction est correcte ou non ?

Il existe également plusieurs méthodes d'évaluation. Ici seront présentés: le Root Mean Square Error (RMSE), le Mean Absolute Error (MAE) et le R-squared (R2).

Évaluer la performance des algorithmes de régression



RMSE: Le Root Mean Square Error (RMSE) donne l'information sur le niveau d'erreur que le système émettra dans ses prédictions. Un poids plus élevé est accordé aux larges erreurs.

$$RMSE = \sqrt{\frac{1}{m}\sum(f(x_i) - y_i)^2}$$

Avec m désigne le nombre d'individus dans la base de données, $f(x_i)$ est la valeur prédite de la variable dépendante pour l'individu i et y_i est la valeur observée, attendue pour l'élément i.

Évaluer la performance des algorithmes de régression

RMSE:

$$RMSE = \sqrt{\frac{1}{m}\sum(f(x_i) - y_i)^2}$$

Une valeur de notre RMSE égale à 0 indiquerait que le modèle de régression s'adapte parfaitement aux données, et donc une petite valeur est préférable. Il existe une autre mesure semblable, mais plus adaptée aux situations contenant de nombreux cas particuliers. On préférera dans ce cas appliquer le Mean Absolute Error (MAE) (Géron, 2019).

MAE:

$$MAE = \sqrt{\frac{1}{m}\sum|(f(x_i) - y_i)|}$$

Évaluer la performance des algorithmes de régression

R^2: Proportion de la variance expliquée par les variables de prédiction dans l'échantillon (R-squared) et un estimateur de la population (adjusted R-squarred).

Plus R^2 sera proche de 1, mieux notre modèle sera en mesure de prédire notre variable dépendante, même si dans certaines situations un faible R2 n'indique pas forcément une mauvaise prédiction.

Toutes ces mesures de performance sont disponibles sur R:

https://rdrr.io/cran/caret/man/recall.html

References i