

Premiers pas dans R

Galharret Jean-Michel
Département MSC

Ressources disponibles à l'adresse : https://galharret-github.io/cours_ONIRIS.html

Présentation rapide de R

Bref historique : - R est à la fois un logiciel et un langage. Il est gratuit et open source.

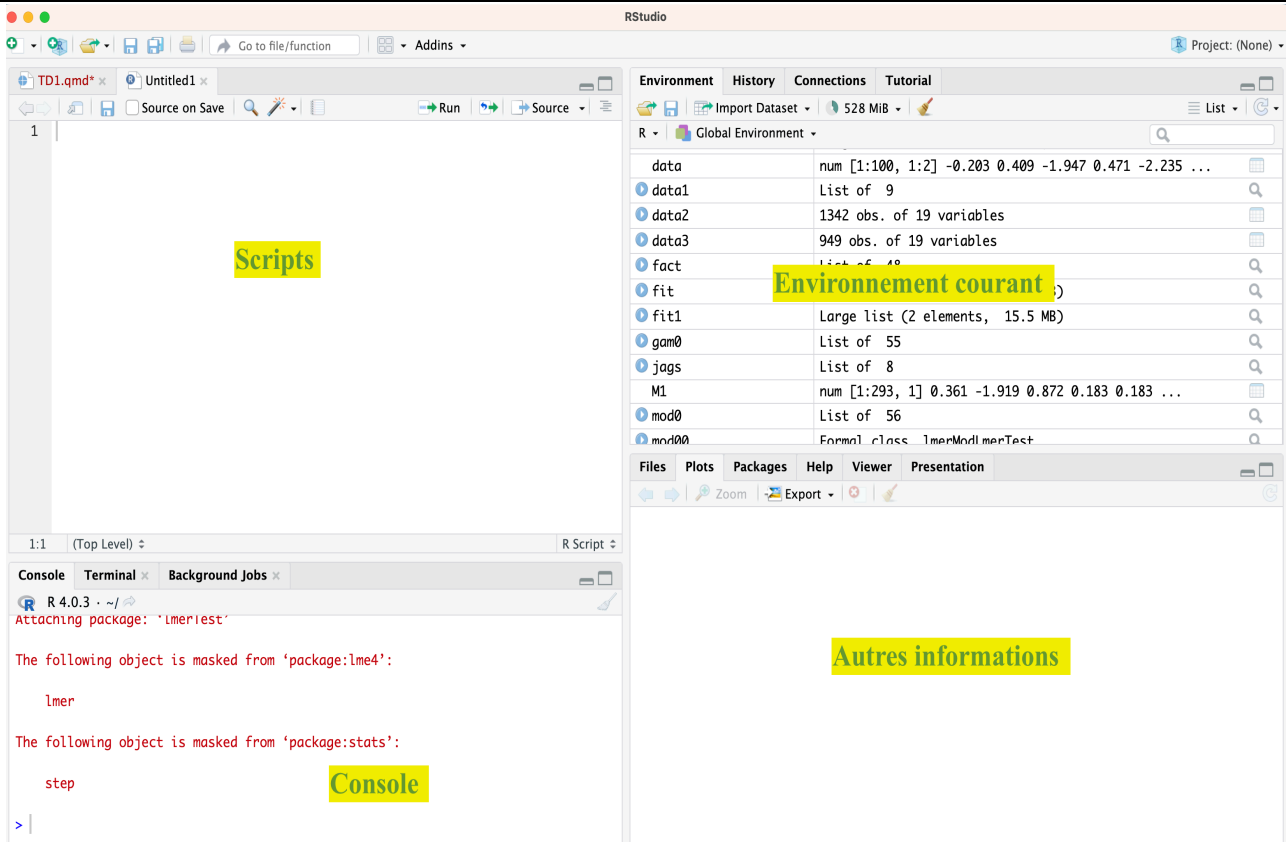
- Version libre et gratuite du langage S développée chez Bell Laboratories par John Chambers (1980).
- Robert Gentleman & Ross Ihaka (Université d'Auckland) proposent une première version de R en 1993.
- R Core Team créée en 1997 assure la maintenance et l'évolution de R.
- CRAN (Comprehensive R Archive Network) regroupe et met à disposition l'ensemble des éléments de R.

Fichiers gérés par R

- Les scripts (fichiers .R) : ils vont contenir les codes R ainsi que des commentaires sur ces codes (très importants par la suite).
- Les environnements (fichiers .RData) qui seront des ensembles d'objets.

Interfaces de R :

- La basique : Rgui (déconseillée) composée d'une fenêtre principale appelée la Console à partir de laquelle on exécute les fonctions.
- RStudio : interface graphique utilisée.



RStudio

Premières commandes

Dans la console exécuter les lignes suivantes :

```
1 + 1
[1] 2
pi
[1] 3.141593
sin(0)
[1] 0
sin(3*pi/2)
[1] -1
```

A partir de maintenant toutes les commandes seront enregistrées dans un fichier script que vous nommerez TD1.R Pour ce faire : File -> New File -> R Script

Les vecteurs

- Un des objets de base de R
- toutes les valeurs de même type (nombre, chaîne de caractères, booléen)
- création via la fonction `c()` [combine].

```
a1=c(1.5,2,3.2,-1.5,0,-1)
a2=c("fille","garçon","fille","fille","fille","fille")
a3=c(TRUE,FALSE,TRUE,TRUE)
```

On peut déterminer la nature du vecteur en utilisant `class()` :

```
class(a1)
[1] "numeric"
class(a2)
[1] "character"
class(a3)
[1] "logical"
```

Exercice :

- 1) Combiner les vecteurs `a1` et `a2` dans un nouveau vecteur `a`.
- 2) Quelle est la classe du résultat ?
 - **rep** permet d'écrire `n` fois la valeur `a` : `rep(a,n)`

```
rep(5,10)
[1] 5 5 5 5 5 5 5 5 5 5
rep("fille",10)
[1] "fille" "fille" "fille" "fille" "fille" "fille" "fille" "fille" "fille"
[10] "fille"
```

Exercice : créer un vecteur avec 5 fois fille et 10 fois garçon.

- **seq** définit une séquence de nombre entre MIN et MAX avec un pas de L : `seq(MIN,MAX,by=L)`

```
seq(0,1,by=0.1)
[1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
seq(10,20,by=5)
[1] 10 15 20
```

Indexation et longueur d'un vecteur

La fonction `length()` permet de calculer le nombre de valeurs d'un vecteur (quelle que soit la nature du vecteur considéré).

```
length(a1)
```

```
[1] 6
```

```
length(a2)
```

```
[1] 6
```

Chaque élément d'un vecteur est repéré par un indice entre [].

```
a1[1]
```

```
[1] 1.5
```

```
a1[3]
```

```
[1] 3.2
```

```
# Opérations sur les éléments d'un vecteur
```

```
a1[1]+a1[3]
```

```
[1] 4.7
```

```
a1[1]/a1[3]
```

```
[1] 0.46875
```

On peut changer la valeur de a1[3] en lui affectant une nouvelle valeur :

```
a1[3]= -3
```

```
a1
```

```
[1] 1.5 2.0 -3.0 -1.5 0.0 -1.0
```

On peut aussi ajouter de nouvelles valeurs aux vecteurs a1

```
a1[12]=10
```

```
a1[13]=11
```

Les calculs de base avec un vecteur

On peut ajouter, soustraire, multiplier des vecteurs entre eux à condition qu'ils soient de même longueur et qu'ils soient numériques.

```
a1=seq(10,20,by=1)
```

```
a2=seq(20,30,by=1)
```

```
a1+a2
```

```
[1] 30 32 34 36 38 40 42 44 46 48 50
```

```
a2-a1
```

```
[1] 10 10 10 10 10 10 10 10 10 10 10
```

```
a1*a2
```

```
[1] 200 231 264 299 336 375 416 459 504 551 600
```

```
a2/a1
```

```
[1] 2.000000 1.909091 1.833333 1.769231 1.714286 1.666667 1.625000 1.588235
[9] 1.555556 1.526316 1.500000
```

```
a1^2*log(a2)
```

```
[1] 299.5732 368.3872 445.1101 529.8985 622.8986 724.2471 834.0727
[8] 952.4969 1079.6343 1215.5938 1360.4790
```

Ouvrir l'aide

Toutes les fonctions prédéfinies dans R possèdent une vignette (description des arguments de la fonction, des sorties et un exemple utilisant la fonction). Pour accéder à cette aide on met un point d'interrogation devant le nom de la fonction.

Exemple : Décrire la fonction *sample*

```
?sample
```

Une fonction comporte plusieurs arguments. Par exemple ***sample(x, size, replace = FALSE, prob = NULL)***

Parmi les arguments, certains sont obligatoires

- *x* est un vecteur dans lequel on va choisir des valeurs au hasard.
- *size* est la taille du résultat

d'autres sont facultatifs (ils ont des valeurs par défaut) ici *replace* et *prob*.

```
sample(-10:10,1)
```

```
[1] -10
```

```
sample(-10:10,3)
```

```
[1] -8 -6 7
```

Un peu de logique

On définit un vecteur

```
set.seed("44")
```

```
x=sample(-20:20,40,T)
```

On veut savoir combien de valeurs de *x* sont positives.

```
sum(x>0)
```

```
[1] 15
```

Quels sont les positions de ces valeurs ?

```
which(x>0)
```

```
[1] 2 7 8 16 18 21 22 24 26 27 30 32 33 35 39
```

Quels sont les indices des valeurs supérieures à 5 ou inférieures à 0 ?

```
which(x>5 | x<0)
```

```
[1] 1 2 3 4 5 6 7 8 9 10 11 12 14 15 16 17 18 19 22 23 24 25 26 27 28
[26] 29 30 31 32 33 34 35 36 37 38 39 40
```

Quels sont les indices des valeurs dans [5,10] ?

```
which(x>=5 & x<=10)
```

```
[1] 8 32 39
```

Exercices

Exo 1 :

1. Créer le vecteur `x=c("lannister", "targaryen", "baratheon", "starck", "greyjoy")`
2. Afficher le premier élément de `x`
3. Afficher tous les éléments de `x` sauf le premier
4. Afficher les trois premiers éléments de `x`.
5. Afficher le deuxième et le quatrième élément de `x`.
6. Classer les éléments de `x` dans l'ordre alphabétique puis anti-alphabétique grâce aux fonctions `sort` et `rev`.

Exo 2 :

1. Créer un vecteur `y` contenant les entiers pairs de 0 à 100 grâce à la fonction `seq`.
2. Ajouter les entiers les entiers impairs de 151 à 175 à `y`. Vérifier que l'ajout s'est fait correctement.
3. A l'aide de la fonction `length` déterminer la taille du vecteur `y` final.
4. A l'aide de la fonction `rep` créer un vecteur sonnette contenant 4 fois « ding » puis 4 fois « dong ».
5. A l'aide de la fonction `sample` créer un vecteur `melodie` contenant une succession aléatoire de « ding » et de « dong » de taille 100.
6. Grâce à la fonction `table`, déterminer combien il y a de « ding » et combien il y a de « dong » dans `melodie`.

Les matrices

Les matrices sont comme en mathématiques des tableaux de données. Attention comme pour les vecteurs les matrices ne contiendront que des éléments du même type (numérique, texte, booléen).

Définition et fonctions de base

```
A=matrix(c(1,2,3,
           2,1,-1,
           -3,-1,1),nrow=3)
```

Quelques fonctions utiles la dimension de la matrice

```
dim(A)
```

```
[1] 3 3
```

Le résultat est donc un vecteur ayant deux éléments le nombre de ligne et le nombre de colonnes

On peut la transposer (ie échanger lignes et colonnes).

```
t(A)
```

```
      [,1] [,2] [,3]
[1,]     1     2     3
[2,]     2     1    -1
[3,]    -3    -1     1
```

Calculer son déterminant

```
det(A)
```

```
[1] 5
```

Calculer son inverse

```
solve(A)
```

```
      [,1] [,2] [,3]
[1,] 7.401487e-17 0.2 0.2
[2,] -1.000000e+00 2.0 -1.0
[3,] -1.000000e+00 1.4 -0.6
```

Attention pour le produit matriciel on utilise %*%

Un vecteur est considéré comme une matrice 3 lignes 1 colonne ou comme une matrice 1 ligne 3 colonnes.

```
B<-c(1,2,3)
```

Multiplie chaque colonne de A par la colonne B

```
A*B
```

```
      [,1] [,2] [,3]
[1,]     1     2    -3
[2,]     4     2    -2
[3,]     9    -3     3
```

%*% fait le "vrai" produit matriciel

```
A%%B
```

```
      [,1]
[1,]    -4
```

```
[2,] 1
[3,] 4

B%%A

      [,1] [,2] [,3]
[1,] 14    1   -2
```

Indexation des éléments d'une matrice

Les éléments d'une matrice sont identifiés par un numéro de ligne et de colonne entre crochets $A[i,j]$

```
A[1,2]

[1] 2

# première ligne de A
A[1,]

[1] 1 2 -3

# première colonne de A
A[,1]

[1] 1 2 3
```

Les data frames

C'est l'objet le plus important et le plus utilisé dans R il s'agit d'un tableau de données mais contrairement aux matrices on peut avoir différents types de données dans un dataframe.

Création d'un data frame

On crée une variable x contenant $n = 100$ nombres distribués selon une loi uniforme continue sur $[0,20]$ (fonction `runif`) et un variable `gr` (expliquer le résultat du code correspondant)

```
x<-runif(100,0,20)
gr<-sample(c("A","B"),size=100,replace=T,prob=c(0.6,0.4))
df<-data.frame(x=x,gr=gr)
```

Importation d'un data frame inclus dans un package

Pour accéder aux data frames de R :

```
data()
```

ensuite on peut grâce à la fonction `help` obtenir de l'aide sur l'un des data frame :

```
help("mtcars")
```

Ensuite on charge le jeu de données via

```
data("mtcars")
```


Une fois le data frame importé on constate qu'il est présent dans la fenêtre environnement

Environment

History

Connections

Tutorial

Import Dataset

637 MiB

List

R

Global Environment

Data

mtcars

32 obs. of 11 variables

\$ mpg : num

21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...

\$ cyl : num

6 6 4 6 8 6 8 4 4 6 ...

\$ disp: num

160 160 108 258 360 ...

\$ hp : num

110 110 93 110 175 105 245 62 95 123 ...

\$ drat: num

3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...

\$ wt : num

2.62 2.88 2.32 3.21 3.44 ...

\$ qsec: num

16.5 17 18.6 19.4 17 ...

\$ vs : num

0 0 1 1 0 1 0 1 1 1 ...

\$ am : num

1 1 1 0 0 0 0 0 0 0 ...

\$ gear: num

4 4 4 3 3 3 3 4 4 4 ...

\$ carb: num

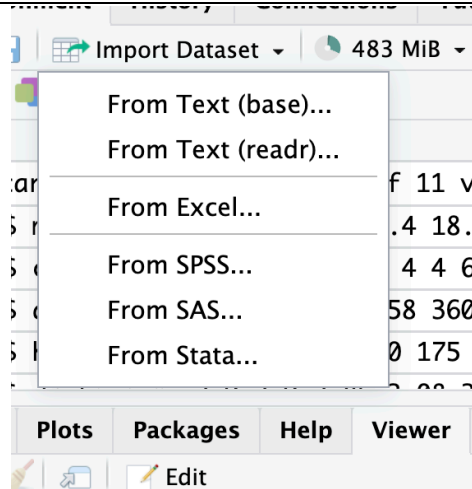
4 4 1 1 2 1 4 2 2 4 ...

Fenêtre environnement

Importation d'une base de données externe

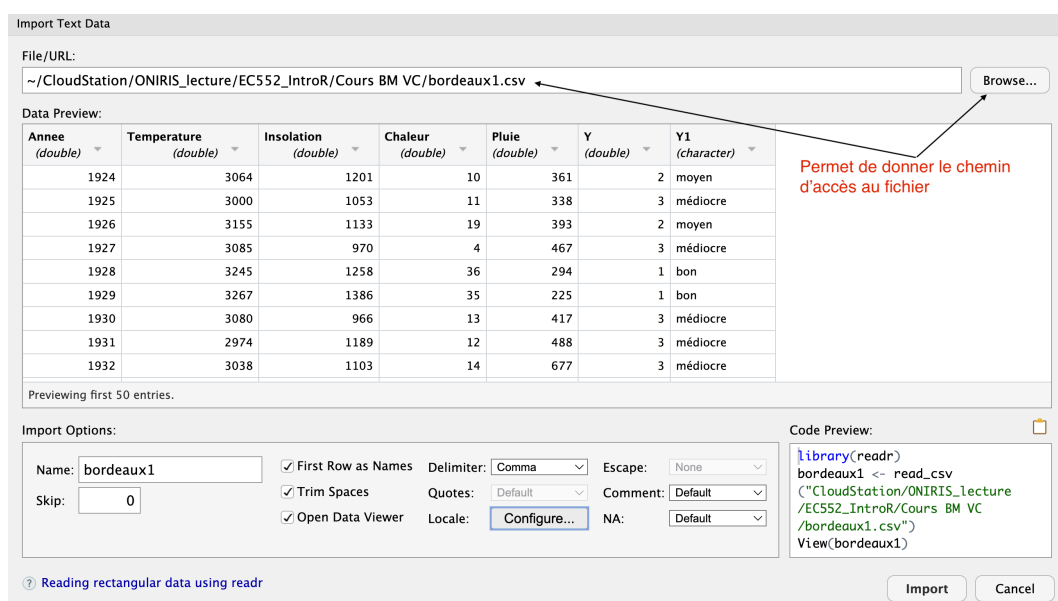
On peut utiliser une interface graphique pour ouvrir les fichiers contenant le data frame. Les plus courants sont :

- *.csv : (Comma Separated Values) ce sont des fichiers de type tableaux sans mise en forme
- *.xlsx : fichiers produits à partir du logiciel excel
- *.ods : fichiers produits à partir du logiciel Calc de LibreOffice.



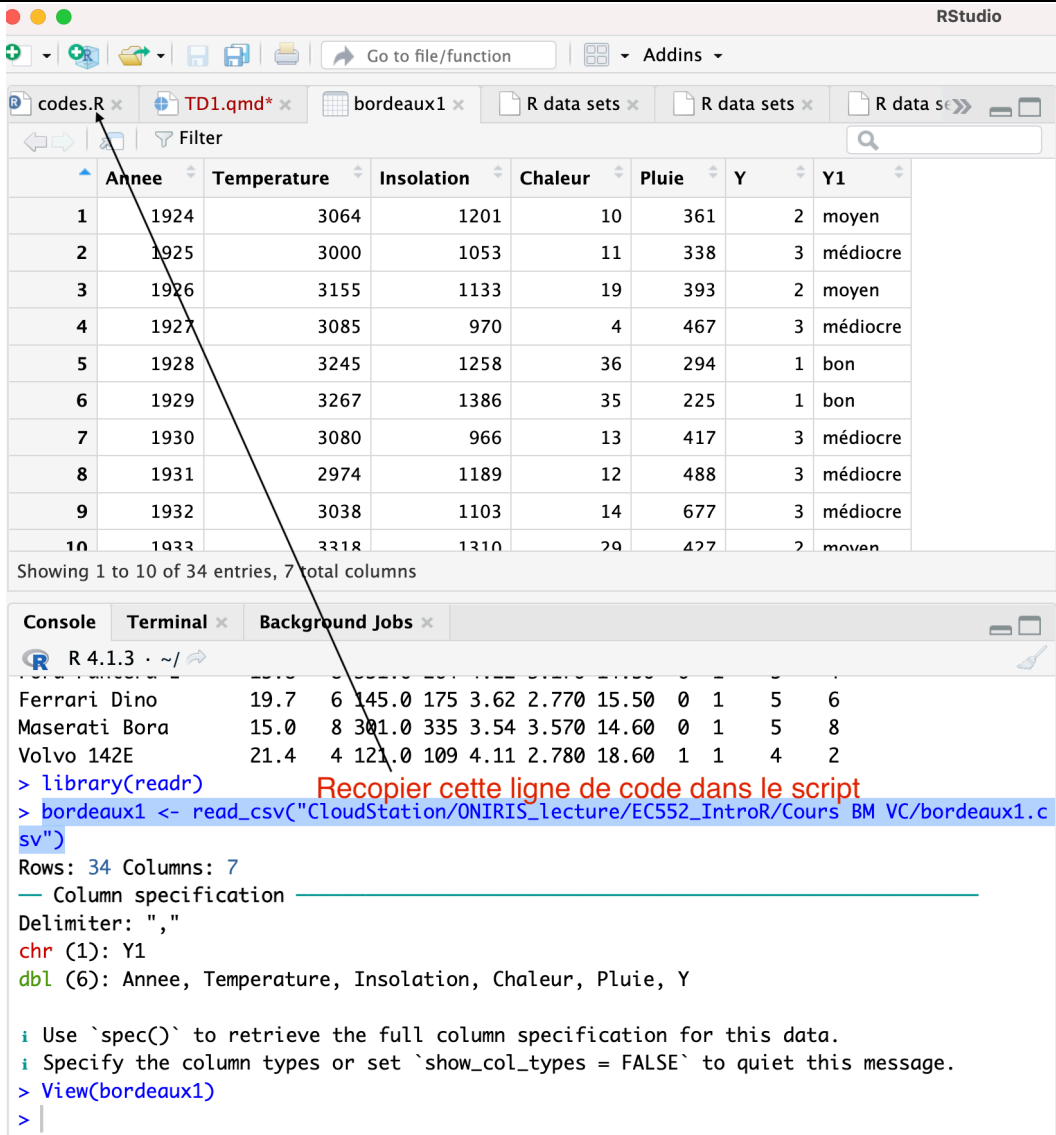
Menu Importer

On va importer le data frame contenu dans le fichier bordeaux.csv disponible sur Connect (vous devez le télécharger).



Importation du fichier .csv

Ensuite il est recommandé de copier/coller la ligne de commande dans le script R afin de pas à avoir à reproduire la manipulation à chaque session de travail sur le fichier bordeaux.csv



Showing 1 to 10 of 34 entries, 7 total columns

	Annee	Temperature	Insolation	Chaleur	Pluie	Y	Y1
1	1924	3064	1201	10	361	2	moyen
2	1925	3000	1053	11	338	3	médiocre
3	1926	3155	1133	19	393	2	moyen
4	1927	3085	970	4	467	3	médiocre
5	1928	3245	1258	36	294	1	bon
6	1929	3267	1386	35	225	1	bon
7	1930	3080	966	13	417	3	médiocre
8	1931	2974	1189	12	488	3	médiocre
9	1932	3038	1103	14	677	3	médiocre
10	1933	3318	1310	29	427	2	moyen

```

R 4.1.3 ~ /
Ferrari Dino      19.7  6 145.0 175 3.62 2.770 15.50 0 1 5 6
Maserati Bora     15.0  8 301.0 335 3.54 3.570 14.60 0 1 5 8
Volvo 142E        21.4  4 121.0 109 4.11 2.780 18.60 1 1 4 2

> library(readr)
> bordeaux1 <- read_csv("CloudStation/ONIRIS_lecture/EC552_IntroR/Cours BM VC/bordeaux1.csv")
Rows: 34 Columns: 7
— Column specification —————
Delimiter: ","
chr (1): Y1
dbl (6): Annee, Temperature, Insolation, Chaleur, Pluie, Y

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
> View(bordeaux1)
>

```

La fonction `read_csv()`

La fonction `read_csv()` est une fonction du package `readr` donc il faut également indiquer dans le script qu'il faudra charger ce package :

```
library(readr)
```

Exercice sur les data frame

1. Charger le fichier de données iris. Lire l'aide de iris pour comprendre le jeu de données.
2. Quel est le type de iris ? Quelles sont les dimensions de iris ?
3. Appliquer la fonction `str` à iris. A quoi correspondent les informations renvoyées ?
4. Appliquer la fonction `summary` à iris. A quoi correspondent les informations renvoyées ?
5. Utiliser les fonctions `colnames()` et `rownames()`. Quelles sont les informations renvoyées ?
6. A l'aide la fonction de texte paste remplacer le nom des lignes par fleur 1, ..., fleur 150.

7. Pour un data frame on peut appeler une colonne par son nom en utilisant \$: iris\$Sepal.Length. En utilisant cette information donner la classe de la colonne Species.
8. Quels sont les niveaux du facteur "Species" (fonction levels) ?
9. Créer une nouvelle colonne nommée groupe (on pensera à \$) dans le data frame iris identique à la colonne Species. Quelle est la classe de cette nouvelle colonne ?
10. Renommer les niveaux de la colonne groupe en A, B, C (A pour setosa, B pour versicolor, C pour virginica). Afficher les indices des lignes de iris correspondant au groupe B.
11. Créer l'ensemble des numéros de lignes correspondantes aux fleurs du groupe A. A l'aide de I afficher les lignes de iris correspondant au groupe A. Proposer une solution alternative.
12. En adoptant la même logique que la question précédente, afficher uniquement les lignes de iris où « Sepal.Length » est inférieur à 5.
13. Combien y a-t-il d'individus ayant la longueur des sépales inférieure à 5 ?

Pour aller plus loin

1. Charger le data frame mtcars. Combien de véhicules et de caractéristiques sur les véhicules sont contenus dans le data frame ?
2. Deux caractéristiques sont mal identifiées dans le data frame car ce sont des variables qualitatives (facteurs), rectifier en utilisant la fonction as.factor.
3. En utilisant l'aide du data frame mtcars renommer les niveaux des deux facteurs précédents.
4. Etablir la table de contingence de ces deux facteurs (fonction table)
5. Calculer la moyenne des 7 premières variables (on utilisera les fonctions mean et apply).
6. Reprendre la question pour les voitures avec un moteur à plat. De même pour les voitures ayant un moteur en V et qui sont automatiques.
7. Calculer le nombre de voitures ayant un moteur en V qui ont au moins 3 carburateurs.