PrivateTeacher

Cours Privés de Science

Maîtriser R tel un Expert

Introduction

R est un ensemple de programmes
qui permettent de faire des calculs,
de manipuler des données ou d'en faire
des représentations graphiques.

L'ensemble des programmes qui constituent R
fonctionnent main dans la main sans que
l'utilisateur ne s'en rende compte.
on parle de suite logicielle intégrée.

R est également libre et gratuit.

"libre" cela signifie que chacun
a le droit de contribuer à améliorer
la suite logicielle librement !

"gratuit" cela signifie que tu peux
t'en servir sans payer

R est utilisé par des débutants
aussi bien que par des professionnels

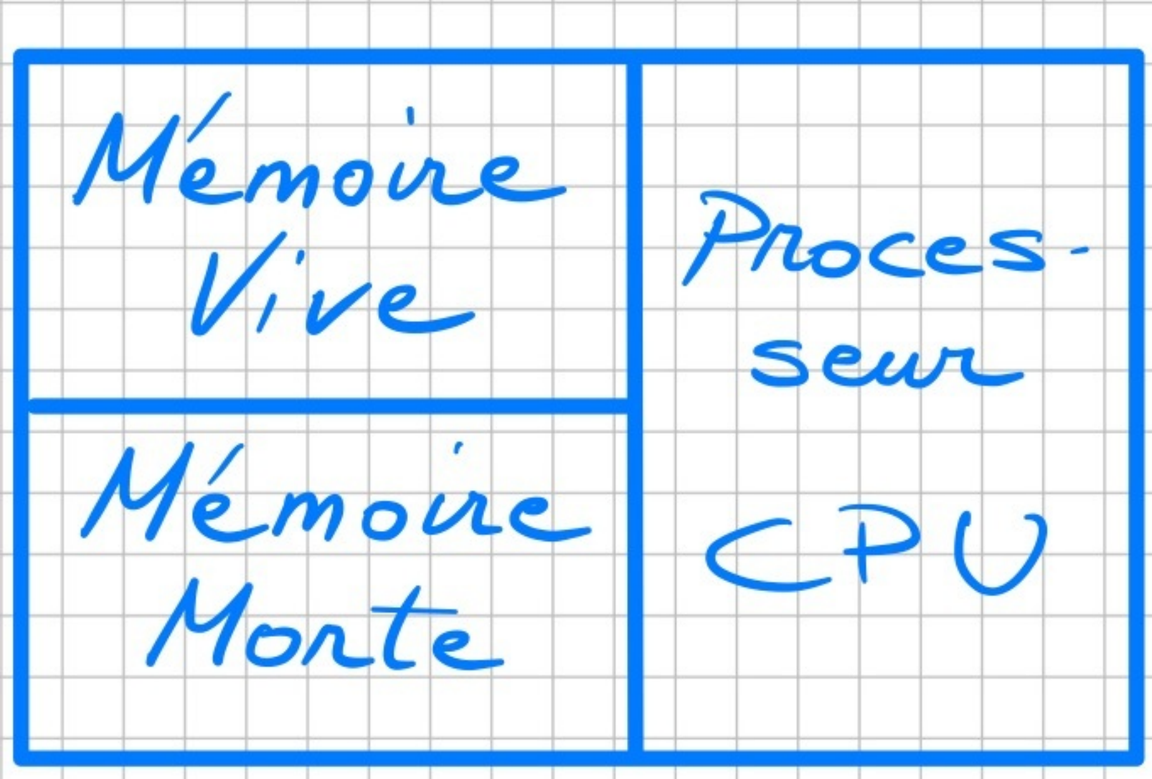
Il s'agit donc d'un investissement
intellectuel intelligent dont tu profiteras
longtemps.

Fonctionnement des ordinateurs.

Pour bien comprendre comment utiliser un logiciel tel que R, il est utile de comprendre comment fonctionne un ordinateur.

Un ordinateur est construit à l'aide de trois constituants essentiels :

1) La mémoire vive ou RAM
2) La mémoire morte ou disque dure
3) Le processeur ou CPU



1) La RAM est une <u>mémoire temporaire</u>
Ce type de mémoire ne fonctionne que si l'ordinateur est allumé. Elle à besoin d'électricité pour fonctionner. Pour cette raison, on l'appel <u>mémoire vive</u>.

2) La <u>mémoire morte</u> au contraire, est une mémoire <u>permanente</u>. Elle n'a pas besoin d'électricité pour fonctionner. Lorsque l'ordinateur est éteint, l'information qu'elle contient est préservée

3) Le processeur enfin est le constituant qui effectue des <u>modifications de la la mémoire</u>. C'est ainsi qu'il effectue toutes les opérations qui permettent à l'ordinateur de fonctionner de manière <u>dynamique</u>.

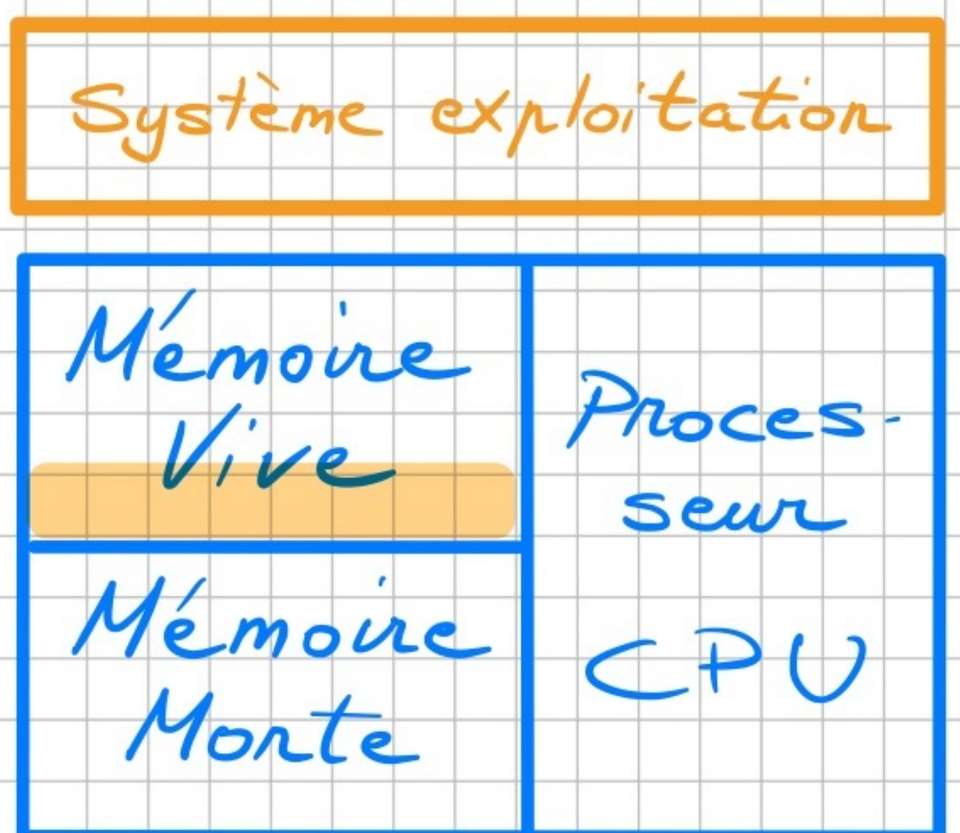
Utilisation de la mémoire

Voici ce qui se passe à présent lorsqu'on allume sa machine :

Le courant électrique circule, la mémoir vive est accessible.

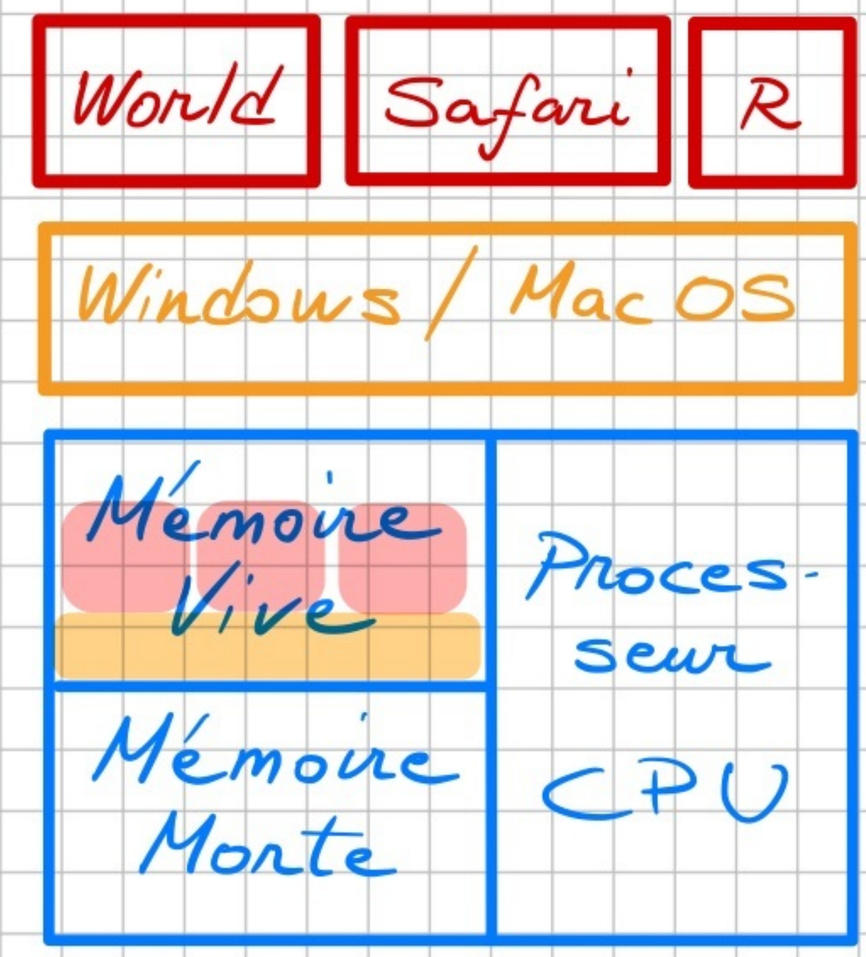
Le processeur commence par charger Windows ou MacOS (le système d'exploitation) du disque dure ou celui-ci est enregistré à la mémoire vive où il pourra être utilisé.

Une fois le système d'exploitation chargé en mémoire, l'utilisateur peut prendre la main et utiliser les resources matérielles pour ses propres activités.



Les activités propres à l'utilisateur sont p. exple le traitement de texte, la navigation internet ou encore les jeux vidéos.

Chacune de ces activités nécessite un logiciel spécifique mais toutes repose sur le même principe : l'exploitation des resources matérielles misent à disposition par le système d'exploitation.



Parmi ces logiciels, il en existe un qui permet de faire des calculs sur un grand nombre de données.

Il s'agit du <u>logiciel R</u> que nous allons voir à présent.

Ligne de commande

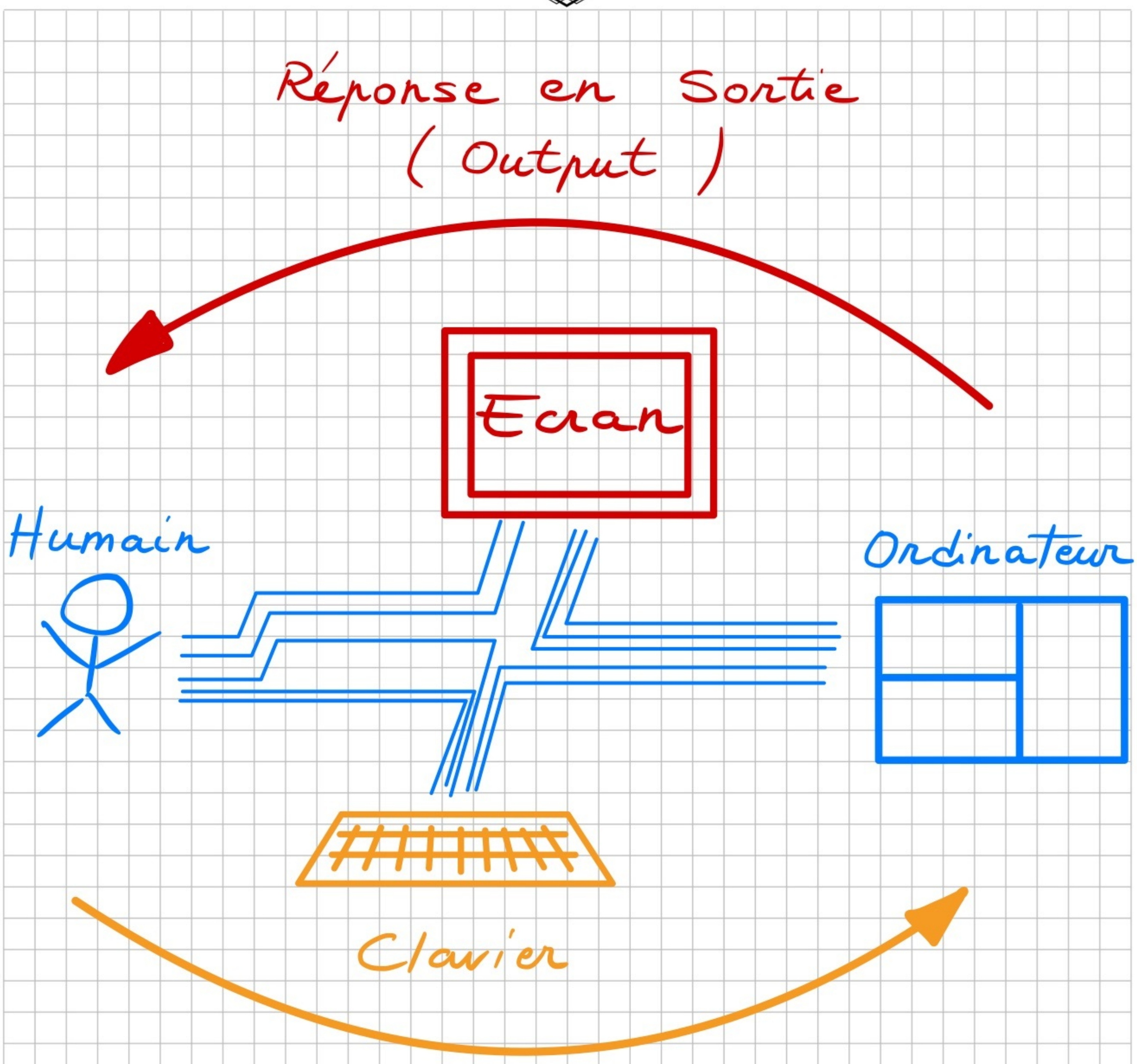
Il existe plusieurs facons de donner ses instructions à un logiciel. La manière la plus courante est certainement la méthode graphique qui consiste à utiliser la souris pour accéder à des menus.

Il existe une autre manière cependant beaucoup plus ancienne et pourtant encore largement utilisée. Il s'agit de la ligne de commande.

Selon cette méthode, l'utilisateur communique ses instructions sous forme de text

Cette méthode est encore largement utilisée car elle est compact et rapide

R est l'un de ces logicil qui prends ses instructions sous forme de text écrit par l'utilisateur.



Le clavier et l'écran constituent ce que l'on appel des "interfaces"

Une interface est ce qui permet à l'être humain de "communiquer" avec l'ordinateur.

Notion de Variable

Une variable est un emplacement de la mémoire vive qui contient une information.

Cela peut être un chiffre, une phrase une matrice ou un vecteur. Il existe un certain nombre de structure de donnée chacun donnant lieu à un type de variable différent.

Il est utile de distinguer le type de variable avec lequel on travail car car tous n'ont pas la même utilité.

On peut additionner par exemple, deux variable de type "chiffe" mais on ne peut pas additionner deux variables de type "text"...

```
x = 4
y = 6
z = "bonjour"
```

Deux variables de type 'numeric' et une variable de type 'caracter'

Le type est donné par la commande "class":

```
class(x)
```

Nom de Variable

Pour pouvoir accéder facilement à l'emplacement de la mémoir qui contient l'information souhaitée, on lui donne un nom. Il s'agit du nom de la variable.

La commande suivante crée un emplacement dans la mémoir qui contient la valeur 46 et lui donne le nom X

```
x = 46
```

Type de fichier

Tout comme une variable stock de l'information dans la mémoir vive et permet de la retrouver à l'aide du nom de la variable, un fichier stock de l'information dans la mémoire morte et permet de la retrouver grâce au nom de fichier.

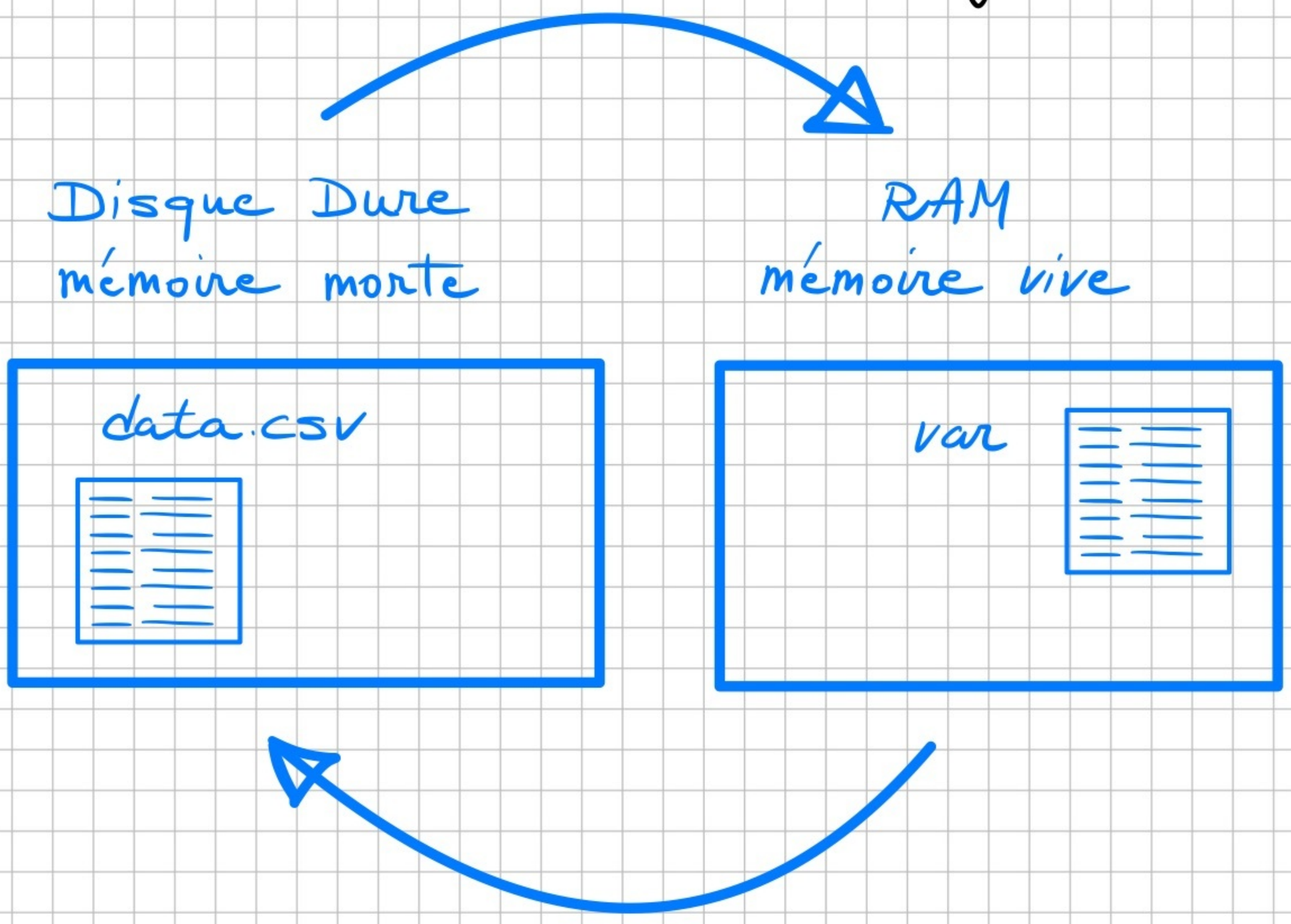
Comme nous l'avons vu, l'ordinateur ne peut travailler avec des données que si celle-ci sont présentent dans la mémoir vive.

Il arrive donc souvent que l'on doive charger le contenu d'un fichier (les données) dans du disque dure dans la RAM

De même, une fois les opérations effectuée, il est utile de sauvegarder le contenu de la mémoir vive sur le disque dure.

Ces deux opérations sont donc réciproques. On peut les représenter ainsi :

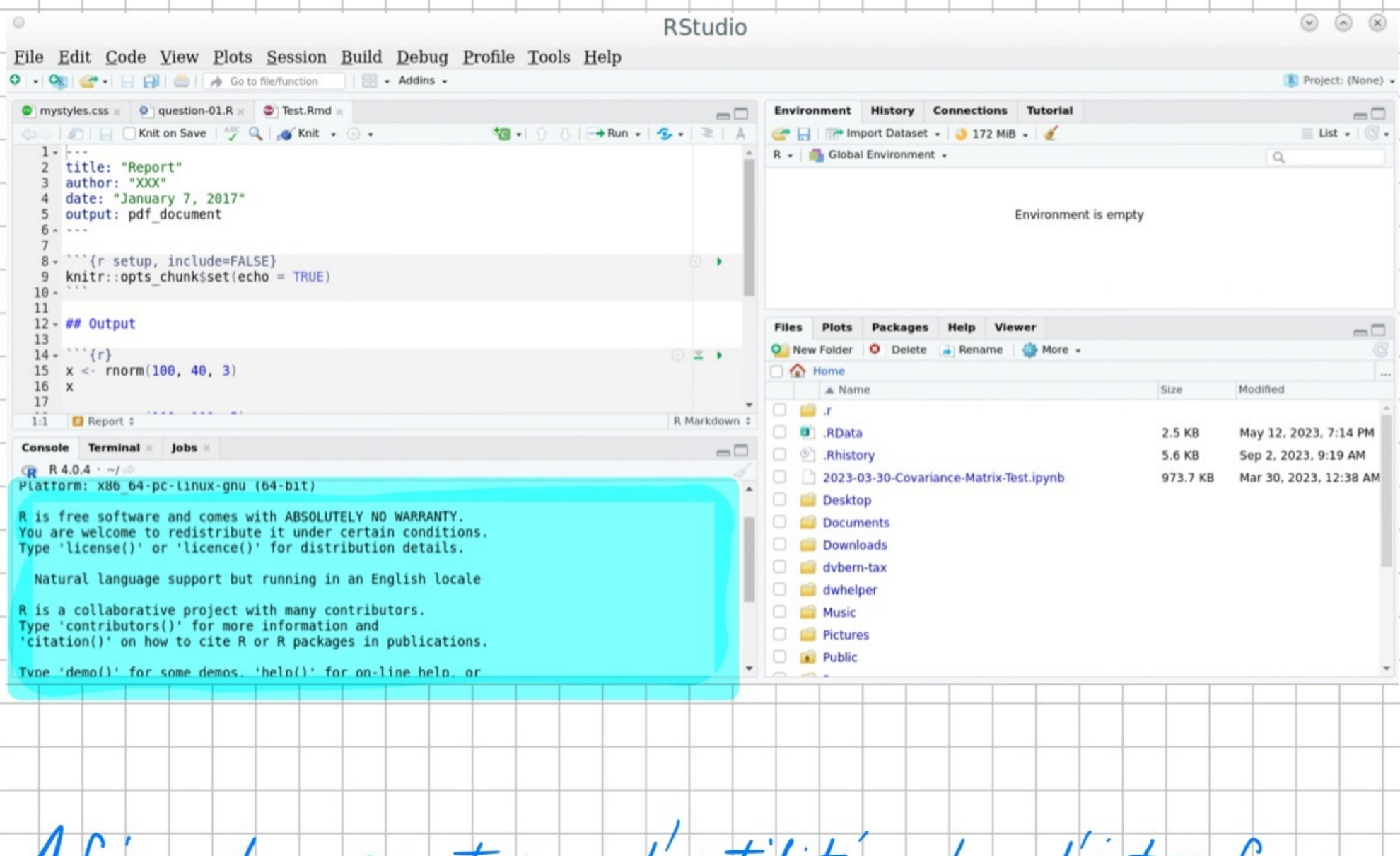
```
nom_variable = read.csv("nom_fichier.csv")
```




```
save.csv(nom_variable, "nom_fichier.csv")
```

RStudio

RStudio est un logiciel très bien construit capable de simplifier beaucoup l'utilisation de R en permettant de bénéficier à la fois des avantage / rapidité de la ligne de commande et la facilité / confort de l'interface graphique.



Afin de montrer l'utilité de l'interface graphique et la puissance de la ligne de commande commencons par importer un fichier CSV :

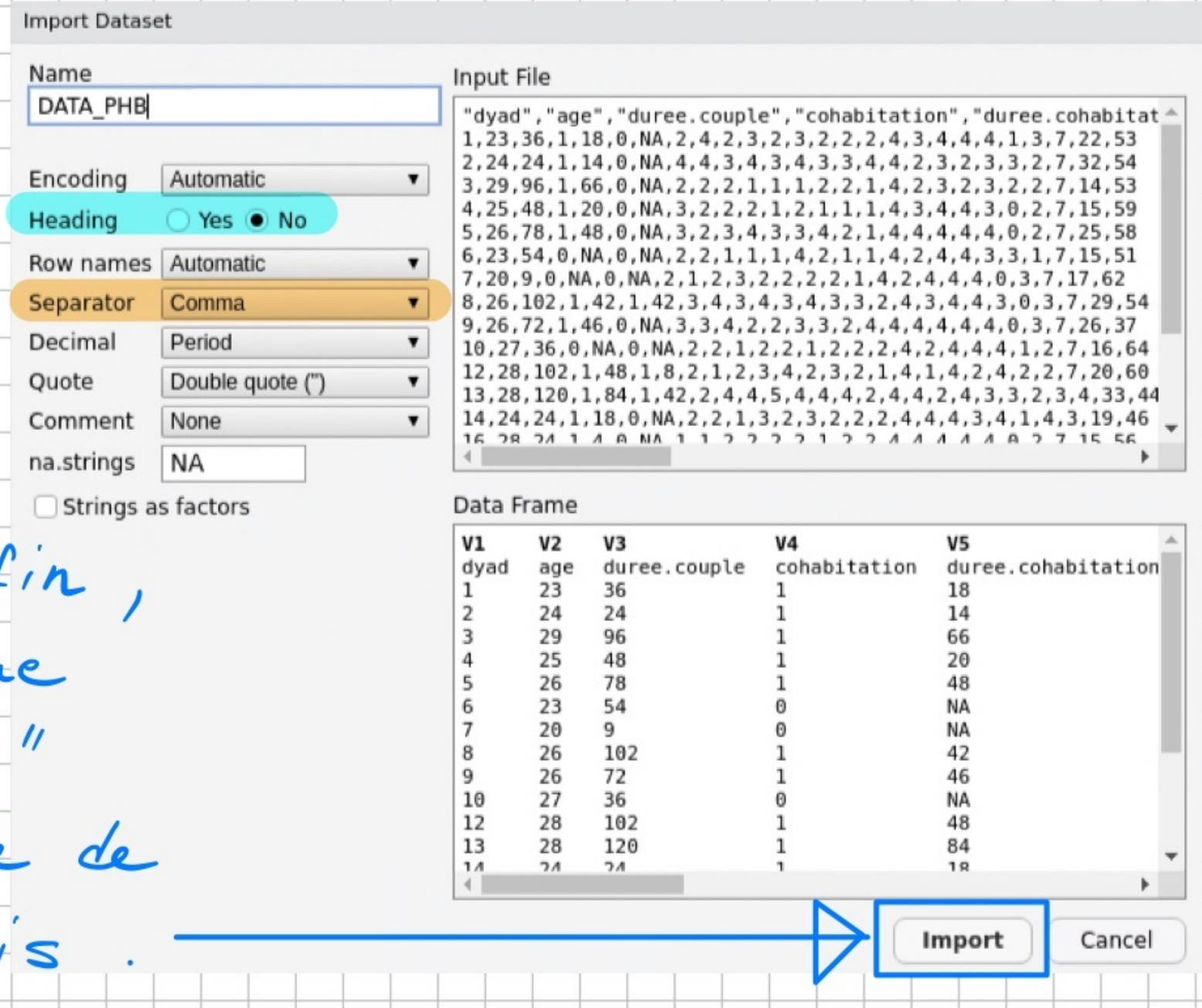
Importer un fichier CSV

Dans RStudio, on clique :

file → import dataset → from text (base)

On selectionne ensuite les options qui nous permettent de lire le fichier correctement.

En particulier les options heading et separator sont importantes comme nous allons le voir :



Puis, enfin, on clique "import" à l'aide de la souris.

L'option **heading** permet de indiquer au logiciel R que la première ligne du fichier que l'on souhaite importer contient le nom des colonnes de donnée

L'option **separator** nous permet de préciser quel est le caractère utilisé pour séparer les données. (ici il s'agit de la virgule "," (comma))

Le résultat nous est montré dans la fenêtre du dessous qui nous montre un apperçu des données telles qu'elles seraient importées avec ces options.

Très utile si l'on ne connait pas le contenu du fichier CSV.

Si l'on souhaite réaliser les mêmes opérations à l'aide de la ligne de commande, on écrit l'instruction suivante :

read.csv("/chemin/vers/fichier.csv", header=True, sep=",")

On indique donc en une seule ligne :

1) Ou trouver le fichier
2) Comment lire les en-tête
3) Quel est le séperateur

Par cet exemple on peut voir que la procédure via l'interface graphique est simple et prend du temps alors que via l'interface graphique, elle plus compliquée mais rapide et efficace

Les deux opérations cependant sont <u>absolument équivalentes</u>.

Il existe un processus cependant, qui ne peut pas être reproduit par l'interface graphique et des cliques de souris :

Il s'agit de reproduire des tâches répétitives.

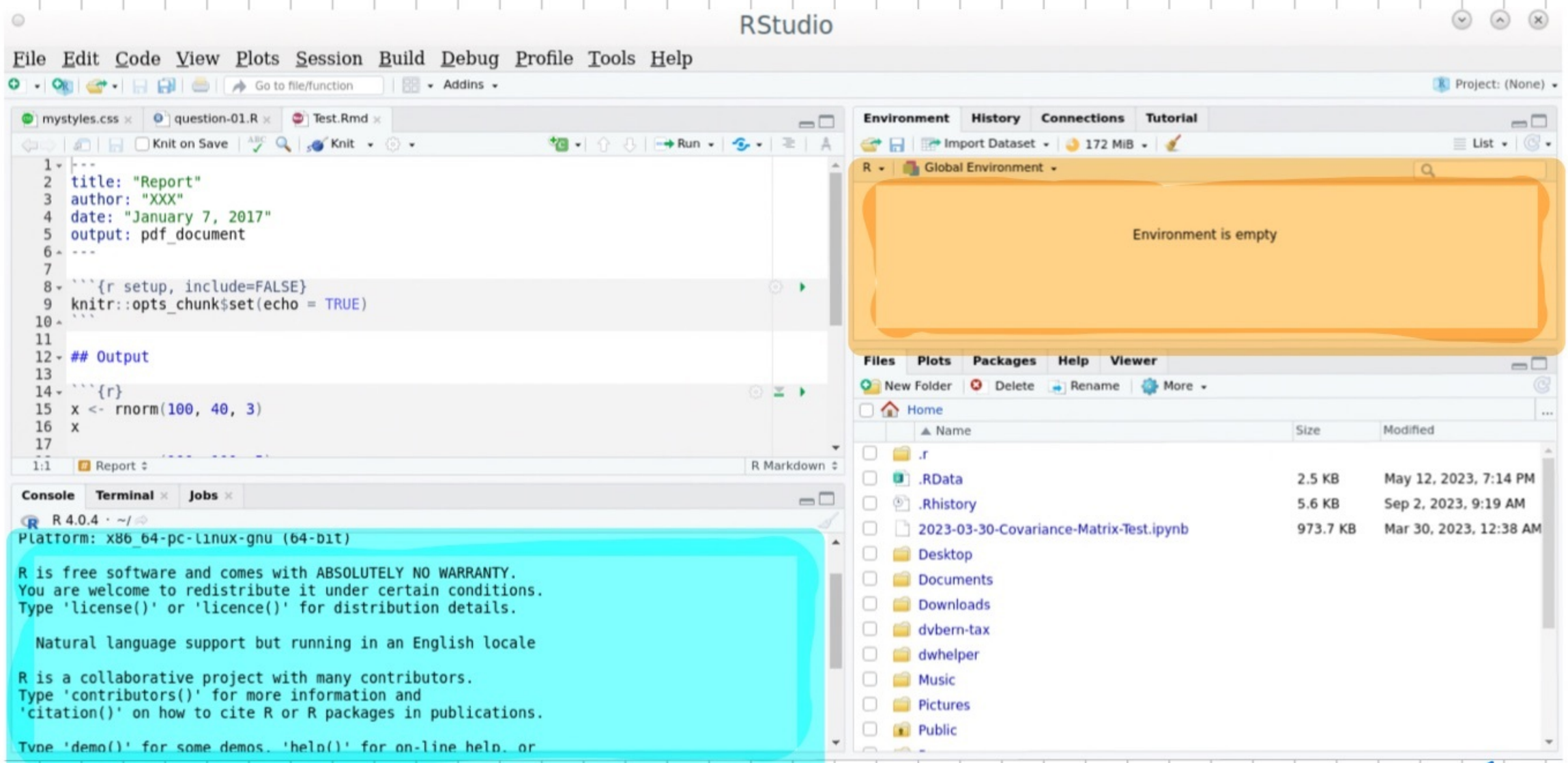
Avec l'interface graphique, il faut refaire la même démarche à chaque fois.

L'interface ligne de commande quant à elle offre la possibilité d'inscrire les instructions dans un fichier text et de le faire lire à R plutôt que de reproduire les mêmes opérations à chaque fois.

On appel cela un script R. Il s'agit simplement d'un fichier text contenant une suite d'instructions.

Environnement de travail

Contenu de la mémoire vive :



ligne de commande

Aide en ligne (tab)

Historique (↓↑)

Set working directory

Fonction

Une fonction est une suite d'instructions à laquelle on a donné un nom : le nom de la fonction.

À la différence d'un scripte, qui contient lui aussi une suite d'instructions, la fonction est présente dans la mémoir vive et on peut donc l'appeler à tout moment au sein de R

d'une console R

de l'environnement de travail

Nous allons à présent créer notre propre fonction. La syntaxe est la suivante :

```
nom_de_la_fonction = function ( ) {
    Instruction 1
    Instruction 2
    ...
}
```

Ouvrir un nouveau script et créer une fonction nommée : fonction_007

```
fonction_007 = function(){
print("Je suis ici dans une fonction")
}
```

Une fois ce code exécuté, la fonction "fonction_007" se trouve en mémoir.

On peut dès lors l'appeler en tapant son nom :

```
fonction_007()
```

Le résultat est facile à anticiper, les instructions présentes dans la fonction (ici une seule) sont exécutée, et la phrase :

"Je suis ici dans une fonction"

s'affiche à l'écran.

Chaque fois que l'on utilise un nom suivit de parenthèses, on fait appel à une fonction.

Argument

Les arguments d'une fonction sont de l'information que l'on transmet à la fonction pour que celle-ci la traîte en interne.

Si l'on souhaite utiliser des arguments au sein d'une fonction, il faut le préciser en l'inscrivant entre les parathèses

Ecrivons à présent une fonction qui accepte des arguments :

```
fonction_008 = function(arg1){

    print(arg1)

}
```

Plutôt que d'écrire toujours la même chose, cette fonction affichera ce qu'on lui donnera en argument :

```
fonction_008("Hello !")

fonction_008(123)
```

Packages

Un package est un ensemble de fonctions (on dit aussi une librairie de fonctions) qui permet d'étendre les possibilités de R.

```
fct_001 = function( ){
        Instruction 1
        Instruction 2
}
```

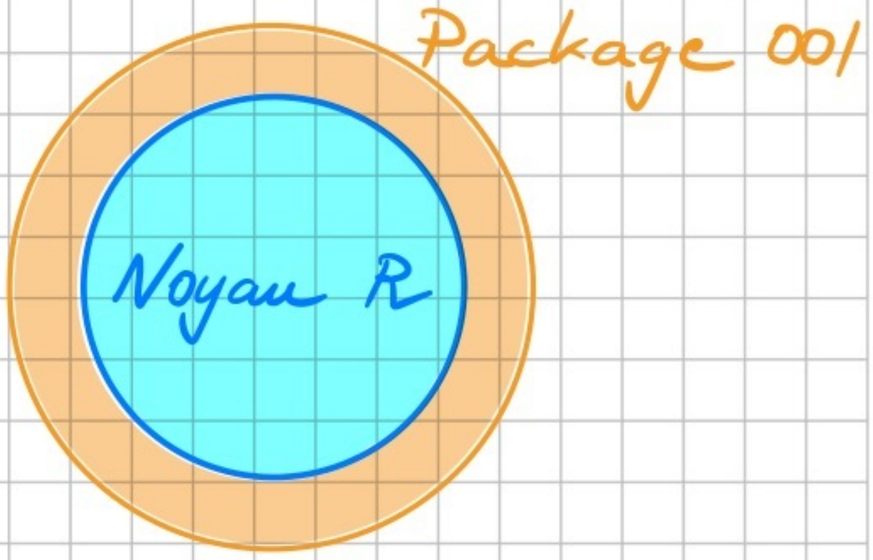
Package

fct_001
fct_002
...
fct_010

```
c(1,2,3,4)    mean(x)
```

Sont des fonctions qui constituent le coeur de R, le noyau de toutes les fonctionnalités de base permettant à R de fonctionner normalement.

On peut augmenter le nombre de fonctions disponibles à l'aide de packages.



Pour pouvoir utiliser les fonctions présentes au sein d'un package on commence par télécharger le fichier depuis internet sur son disque dure (mémoir morte)

```
package.installe("nom_du_package")
```

Puis on charge l'ensemble des fonctions du package dans la mémoir vive :

```
library("nom_du_package")
```

On peut dès lors utiliser l'ensemble de toutes les nouvelles fonctions au sein de R comme s'il s'agissait des fonctions du noyau.

Dans les faits, nous avons ainsi augmenté les capacités de base du noyau

Le strict minimum

Il est essentiel de réaliser que tout ce dont R a besoin pour travailler, ce sont des données chiffrées.

Que celles-ci proviennent d'un fichier CSV, qu'elle soit entrée à la main ou générée artificiellement, R peut tout faire à partire du moment ou on lui donne une série de chiffres.

On peut par exemple générer une suite de nombres aléatoires à l'aide de la commande suivante

rnorm(n=30, mean=1, sd=5)

Cette commande permet de générer 30 nombres aléatoir (random : r) pris au sein d'ure distribution normal (norm) de moyenne 1 et d'écart-type 5.

Si les arguments sont donnés dans cet ordre, on peut très bien aussi écrire :

rnorm(30, 1, 5)

Je t'invite maintenant à essayer
cette commande tu verras le résultat.

Pour utiliser facilement cet suite de
nombre, on la met dans une variable X

```
x = rnorm(30, 1, 5)
```

On peut dès lors demander à R
de faire toute sorte d'opérations
sur ces donnée :

1) trouver le max

```
min(x)
```

2) trouver le min

```
max(x)
```

3) calculer la moyenne

```
mean(x)
```

4) calculer la variance

```
var(x)
```

5) calculer les quantiles Q1 Q2 Q3

```
quantile(x, prob=c(0.25, 0.5, 0.75), type=2)
```

6) faire tout cela en une fois

```
summary(x)
```

7) tracer un histogramme

```
hist(x)
```

Il existe un autre moyen d'entrer des données :

X	Freq(x)
4	5
8	7
12	3
16	2

Admettons que les observations soient les suivantes :

Il est possible d'indiquer à R les observations dans un premier vecteur :

obs = c(4, 8, 12, 16)

Les fréquences dans un second

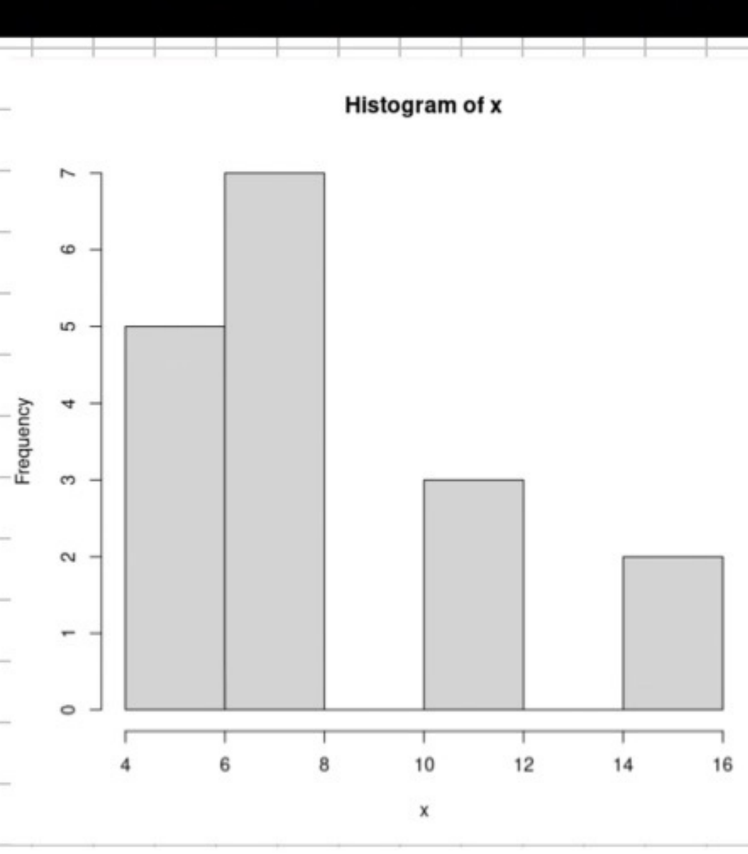
freq = c(5, 7, 3, 2)

puis lui demander de répéter chaque observation selon l'ordre des fréquences

x = rep(obs, freq)

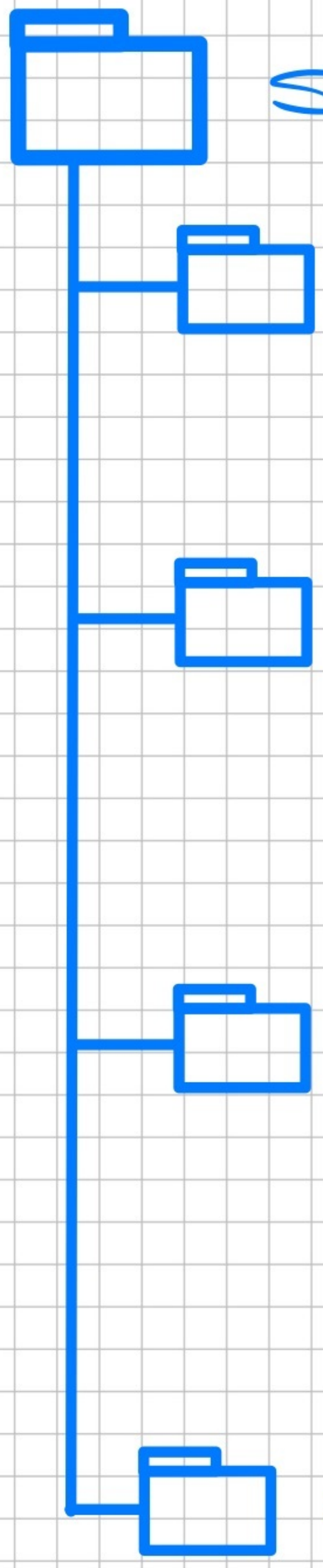
```
> obs = c(4,8,12,16)
> freq = c(5,7,3,2)
> rep(obs, freq)
 [1]  4  4  4  4  4  8  8  8  8  8  8  8 12 12 12 16 16
> x = rep(obs, freq)
```

On reconstitue ainsi les données brutes qu'il devient alors possible de traiter à l'aide des fonctions usuelles **hist(x)**



Organiser ses fichiers

Voici un modèle dont tu peux te servire pour organiser tes propres fichiers



Statistique_1

Cours PDF

Cours_001.PDF

Cours_002.PDF

Exces

S01 - Enoncé.PDF

S01 - Corrigé.PDF

S01 - Script.R

Data

Data_Films.CSV

Data_QI.xls

Data_Inventaire.RData

Script - R - Perso

Importer_CSV.R

Créer_DF.R

Renommer_Colonne.R

Bonus

Tu peux ajouter à ta librairie de document le manuel fichier PDF de R que tu pourra ainsi consulter directement.

Full Reference Index