

# R pour archéologues

## Travaux pratiques

# L'environnement de travail « R »

## Qu'est-ce que R ?

- R est un **langage** de programmation
- > il permet de communiquer avec la machine pour lui demander de réaliser des tâches

> `nrow(table)`



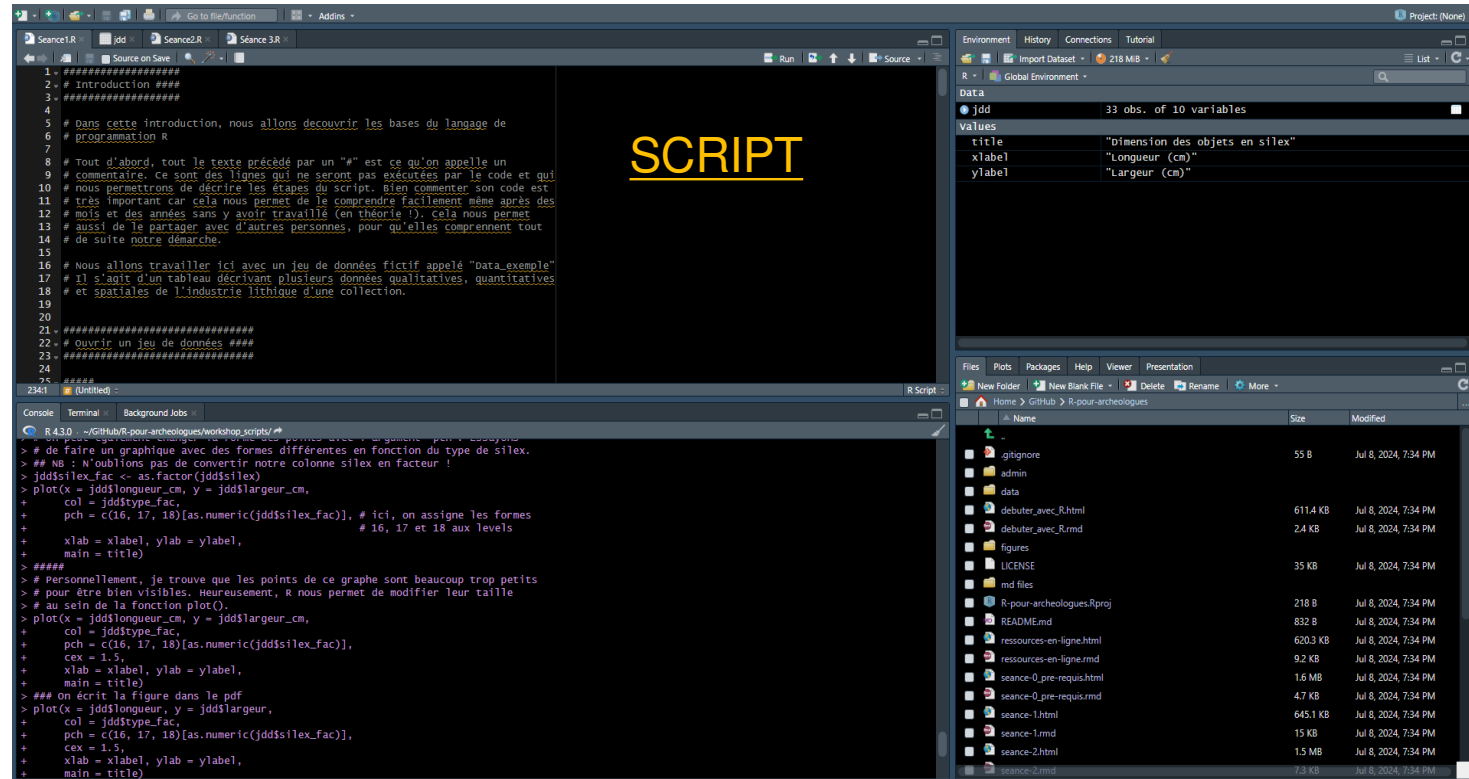
**Affiche-moi le nombre de lignes dans le tableau « table »**



# L'environnement de travail « R »

## Qu'est-ce que R ?

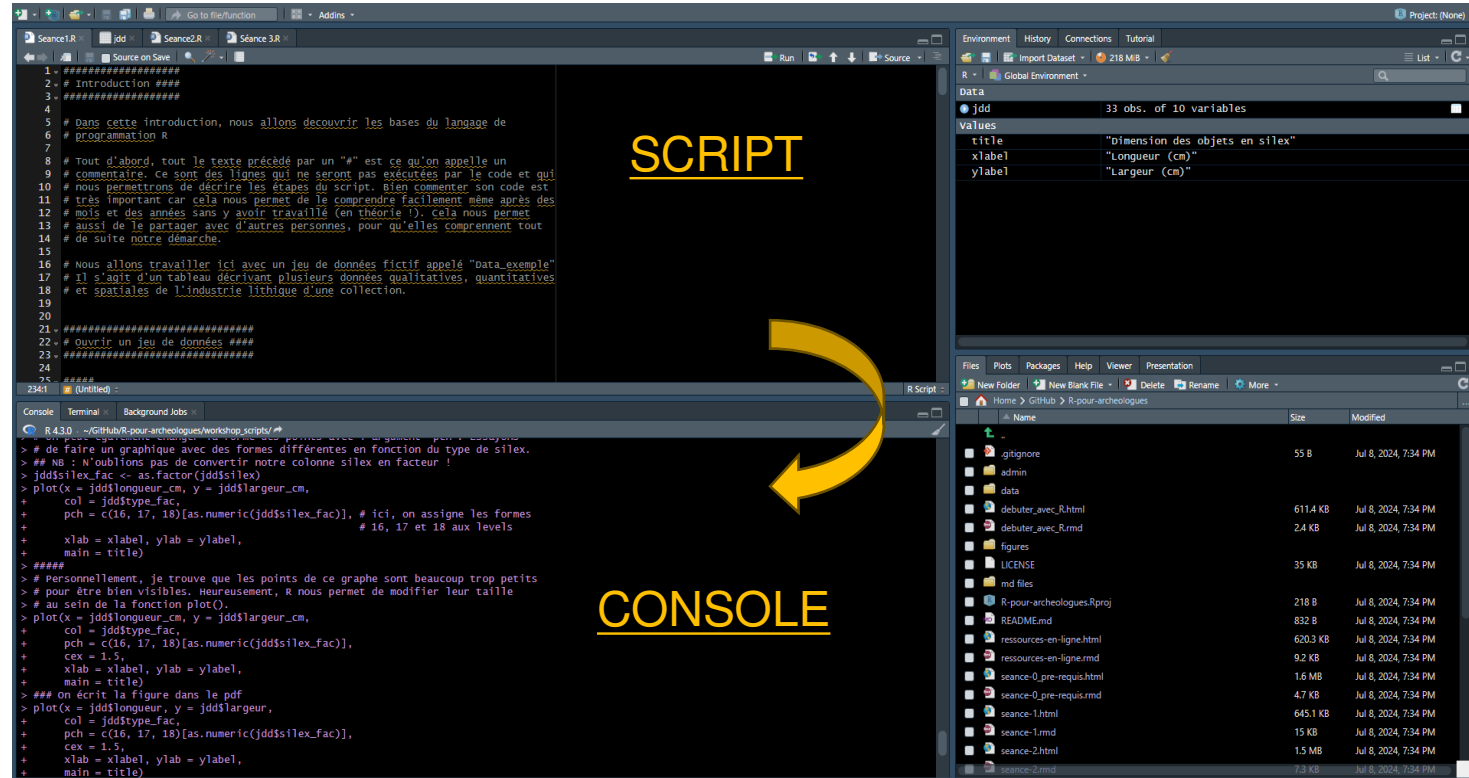
- Rstudio est une **interface** facilitant la programmation avec R



# L'environnement de travail « R »

## Qu'est-ce que R ?

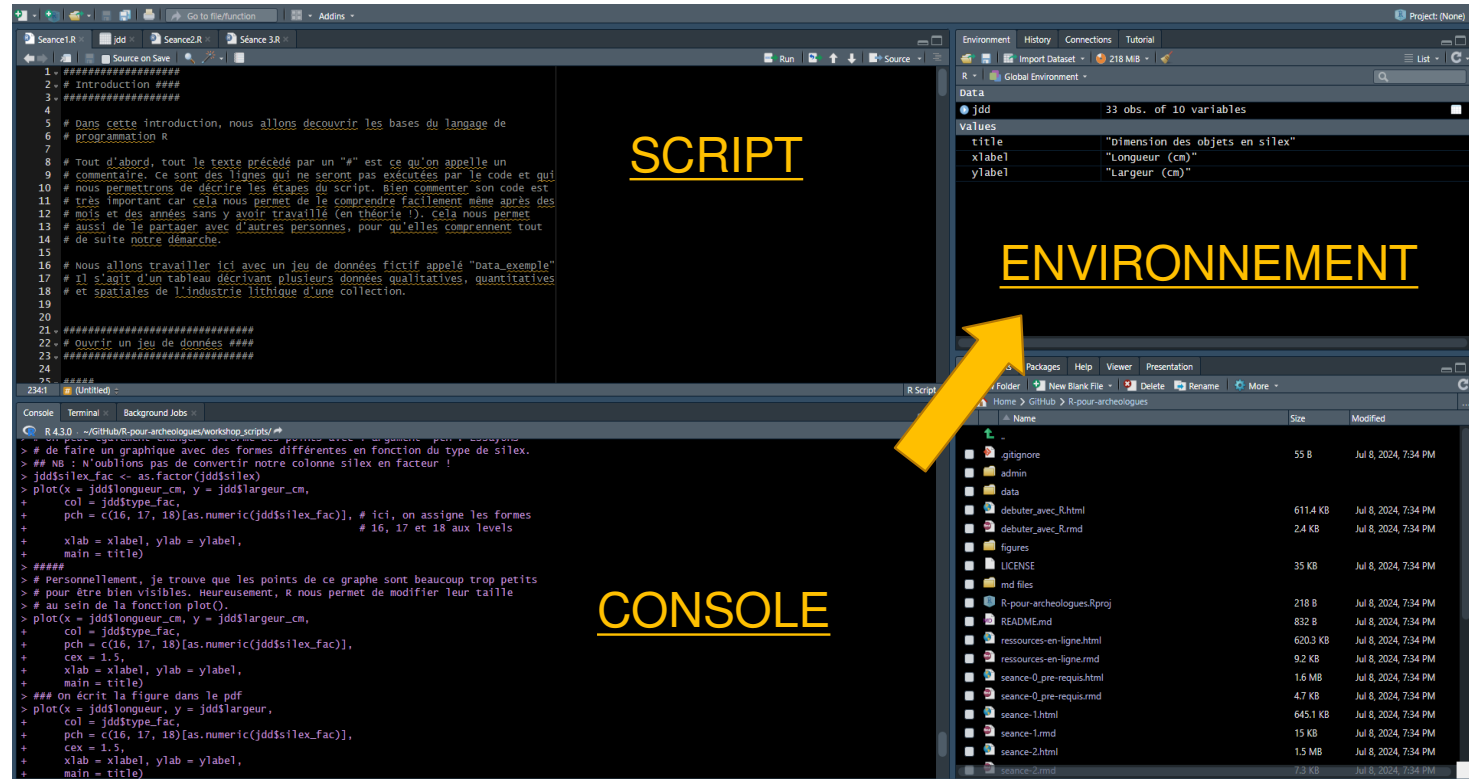
- Rstudio est une **interface** facilitant la programmation avec R



# L'environnement de travail « R »

## Qu'est-ce que R ?

- Rstudio est une **interface** facilitant la programmation avec R



# L'environnement de travail « R »

## Qu'est-ce que R ?

- Rstudio est une **interface** facilitant la programmation avec R

The screenshot displays the RStudio IDE interface. The main window is divided into three panes:



- SCRIPT**: The top-left pane shows an R script with comments in French and code for data manipulation and plotting. The code includes comments about the script's purpose, data loading, and plotting functions like `plot()` and `pdf()`.
- ENVIRONNEMENT**: The top-right pane shows the environment with 33 observations and 10 variables. The variables are `title`, `xlabel`, and `ylabel`, all of type `character`.
- Explorateur de fichiers, Prévisualisation des figures**: The bottom-right pane shows a file explorer with a list of files and folders, including `.gitignore`, `admin`, `data`, `debuter_avec_R.html`, `debuter_avec_R.rmd`, `figures`, `md files`, `ressources-en-ligne.html`, `seance-0_pre-requis.html`, `seance-1.html`, `seance-1.rmd`, and `seance-2.html`.

The bottom-left pane shows the console with the output of the R script, including the execution of the `plot()` function and the `pdf()` command.

# L'environnement de travail « R »

## Quelques bases du langage R

- Le commentaire sert à **documenter** le code



```
159 ### Suppression d'enregistrements sans infos suffisantes
160 Flag.data.clean <- subset(Flag.data.clean, Flag.data.clean$Z != "" | # suppression des pièces sans Z
161                           Flag.data.clean$Multi == "") # suppression des doublons de marquage
162
163 ### Conversion des coordonnées en numérique
164 Flag.data.clean$xabs <- as.numeric(as.character(Flag.data.clean$X))
165 Flag.data.clean$yabs <- as.numeric(as.character(Flag.data.clean$Y))
166 Flag.data.clean$zabs <- as.numeric(as.character(Flag.data.clean$Z))
167
168 write.csv(Flag.data.clean, "Output/Flag_data_clean.csv")
169
```

#



# L'environnement de travail « R »

## Quelques bases du langage R

- Un objet sert à **stocker** une donnée. Plusieurs classes d'objets existent en fonction du type de données que l'on veut manipuler

```
24 # L'objet de base dans R est le vecteur, aussi appelé une variable: une liste de valeurs
25 # Créons une variable nommée 'var1' correspondant a une liste de 1 a 5
26 var1 <- c(1,2,3,4,5)
27 # Créons une variable nommée 'var2' correspondant a une liste continue de A a E
28 var2 <- c("A","B","C","D","E") # notons que le contenu des variables de caractères
29                               # doivent être notés avec des ""
```

Objet

Donnée  
(ici une liste de caractères)





# L'environnement de travail « R »

## Quelques bases du langage R

- Une fonction est une **opération** que l'on va mener sur un/des objet.s

```
121 # A présent, essayons de visualiser un graphique de la longueur vs la largeur
122 # des pièces. Dans le R de base, on utilise la fonction plot()
123 plot(x = jdd$longueur_cm, y = jdd$largeur_cm)
124
125 ## ici, à l'intérieur de la fonction, on a précisé quels arguments étaient
126 ## concernés. Un argument est un objet que l'on passera à la fonction
127 ## pour qu'elle s'exécute. Les fonctions comme class(), levels() n'ont qu'un
128 ## argument, c'est pourquoi nous n'avons pas eu besoin de le préciser. De façon
129 ## générale, les arguments sont rentrés dans l'ordre chronologique dans lequel
130 ## ils sont programmés dans la fonction. Voyons voir cela de plus près.
131 ?plot
```

fonction

Afficher la fiche descriptive de la fonction



# L'environnement de travail « R »

## Quelques bases du langage R

- Une fonction est une **opération** que l'on va mener sur un/des objet.s

```
121 # A présent, essayons de visualiser un graphique de la longueur vs la largeur
122 # des pièces. Dans le R de base, on utilise la fonction plot()
123 plot(x = jdd$longueur_cm, y = jdd$largeur_cm)
124
125 ## ici, à l'intérieur de la fonction, on a précisé quels arguments étaient
126 ## concernés. Un argument est un objet que l'on passera à la fonction
127 ## pour qu'elle s'exécute. Les fonctions comme class(), levels() n'ont qu'un
128 ## argument, c'est pourquoi nous n'avons pas eu besoin de le préciser. De façon
129 ## générale, les arguments sont rentrés dans l'ordre chronologique dans lequel
130 ## ils sont programmés dans la fonction. Voyons voir cela de plus près.
131 ?plot
```

( )

arguments

,



# L'environnement de travail « R »

## Quelques bases du langage R

- Une bibliothèque est un **ensemble de fonctions et/ou d'objets** déjà codées

### Package 'terra'

May 22, 2024

Type Package  
Title Spatial Data Analysis  
Version 1.7-78  
Date 2024-05-22  
Depends R (>= 3.5.0)  
Suggests parallel, tinytest, ncdf4, sf (>= 0.9-8), deldir, XML,  
leaflet (>= 2.2.1), htmlwidgets  
LinkingTo Rcpp  
Imports methods, Rcpp (>= 1.0-10)  
SystemRequirements C++17, GDAL (>= 2.2.3), GEOS (>= 3.4.0), PROJ (>= 4.9.3), sqlite3  
Encoding UTF-8  
Language en-US  
Maintainer Robert J. Hijmans <r.hijmans@gmail.com>  
Description

Methods for spatial data analysis with vector (points, lines, polygons) and raster (grid) data. Methods for vector data include geometric operations such as intersect and buffer. Raster methods include local, focal, global, zonal and geometric operations. The predict and interpolate methods facilitate the use of regression type (interpolation, machine learning) models for spatial prediction, including with satellite remote sensing data. Processing of very large files is sup-

### R topics documented:

terra-package	7
activeCat	20
add	21
add_box	22
add_grid	23
add_legend	24
adjacent	24
aggregate	26
align	28
all.equal	29
animate	30
app	31
approximate	33
Arith-methods	34
as.character	35
as.data.frame	36
as.lines	37
as.list	38
as.points	39
as.polygons	40
as.raster	42
atan2	42
autocorrelation	43
barplot	45
bestMatch	46
boundaries	47
boxplot	48
buffer	49
c	50



# L'environnement de travail « R »

## Importer, afficher et explorer des données

```
> jdd <- read.csv("data/data_exemple.csv", sep = ",", header = T)
> head(jdd)
```

	numero	type	x_cm	y_cm	longueur_cm	largeur_cm	silex	raccord
1	1	gravette	10	60	5.0	1.5	type A	NA
2	2	gravette	30	50	5.9	1.8	type C	NA
3	3	picardie	50	80	3.2	1.2	type B	1
4	4	percoir	260	100	4.7	2.1	type B	NA
5	5	burin	240	120	6.0	2.4	type B	2
6	6	burin diedre	280	150	8.0	3.5	type A	NA

Un fichier de données **CSV**

**Afficher** les 6 premières lignes du tableau



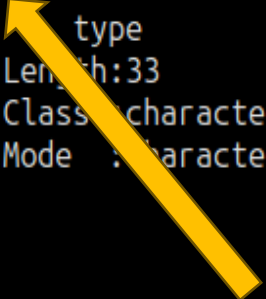
# L'environnement de travail « R »

## Importer, afficher et explorer des données

- Quelques fonctions pour explorer les données

```
> summary(jdd)
```

numero	type	x_cm	y_cm	longueur_cm	largeur_cm	silex
Min. : 1	Length:33	Min. : 10.0	Min. : 50.0	Min. : 1.4	Min. : 0.500	Length:33
1st Qu.: 9	Class :character	1st Qu.: 50.0	1st Qu.: 90.0	1st Qu.: 4.5	1st Qu.: 1.900	Class :character
Median : 17	Mode :character	Median : 210.0	Median : 280.0	Median : 5.9	Median : 2.800	Mode :character
Mean : 17		Mean : 174.8	Mean : 217.6	Mean : 5.8	Mean : 2.976	
3rd Qu.: 25		3rd Qu.: 260.0	3rd Qu.: 340.0	3rd Qu.: 7.4	3rd Qu.: 3.700	
Max. : 33		Max. : 290.0	Max. : 390.0	Max. : 10.2	Max. : 7.100	



Statistiques de base

```
> nrow(jdd)
[1] 33
> ncol(jdd)
[1] 8
> class(jdd$type)
[1] "character"
> class(jdd$longueur)
[1] "numeric"
```

} Nombre de lignes/colonnes

} Type de données ("classes")

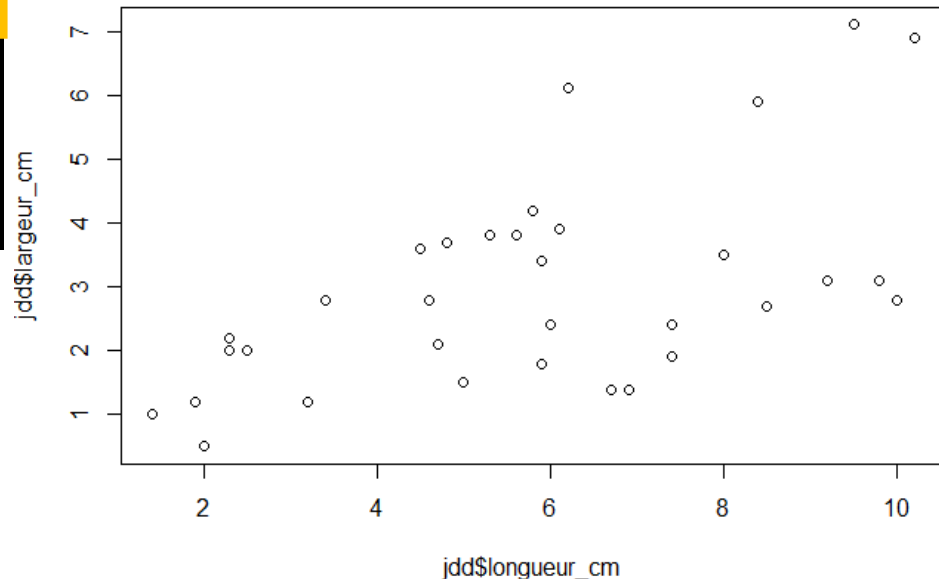
# L'environnement de travail « R »

## Importer, afficher et explorer des données

- Créer une figure simple

```
> plot(x = jdd$longueur_cm, y = jdd$largeur_cm,  
+      col = jdd$type_fac,  
+      pch = c(16, 17, 18)[as.numeric(jdd$silex_fac)],  
+      cex = 1.5,  
+      xlab = "Longueur (cm)", ylab = "Largeur (cm)",  
+      main = "Dimension des objets en silex")
```

> Afficher les points



# L'environnement de travail « R »

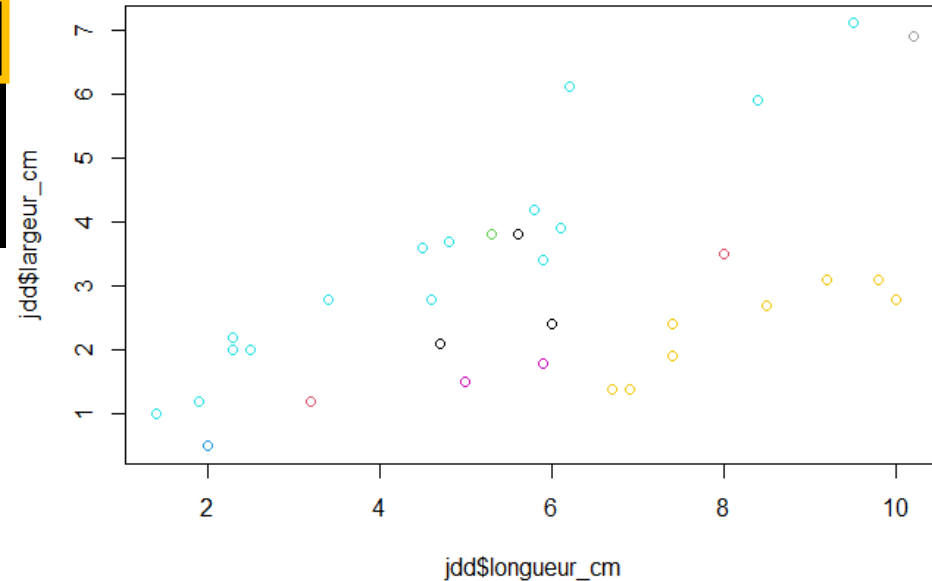
## Importer, afficher et explorer des données

- Créer une figure simple

```
> plot(x = jdd$longueur_cm, y = jdd$largeur_cm,  
+      col = jdd$type_fac,  
+      pch = c(16, 17, 18)[as.numeric(jdd$silex_fac)],  
+      cex = 1.5,  
+      xlab = "Longueur (cm)", ylab = "Largeur (cm)",  
+      main = "Dimension des objets en silex")
```

> Afficher les points

+ **Colorer les points en fonction du type d'objet**



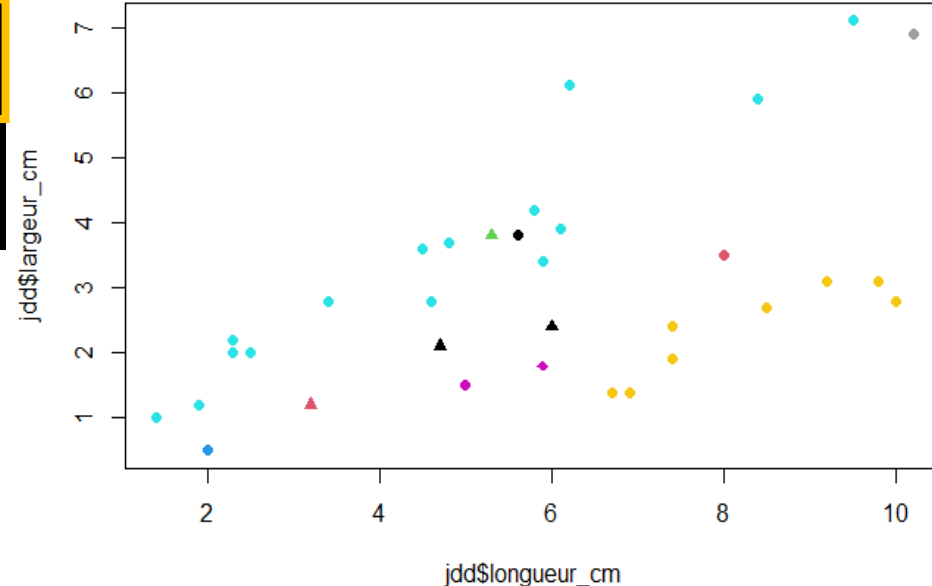
# L'environnement de travail « R »

## Importer, afficher et explorer des données

- Créer une figure simple

```
> plot(x = jdd$longueur_cm, y = jdd$largeur_cm,  
+      col = jdd$type_fac,  
+      pch = c(16, 17, 18)[as.numeric(jdd$silex_fac)],  
+      cex = 1.5,  
+      xlab = "Longueur (cm)", ylab = "Largeur (cm)",  
+      main = "Dimension des objets en silex")
```

- > Afficher les points
- + Colorer les points en fonction du type d'objet
- + **Modifier la forme des points en fonction du type de silex**





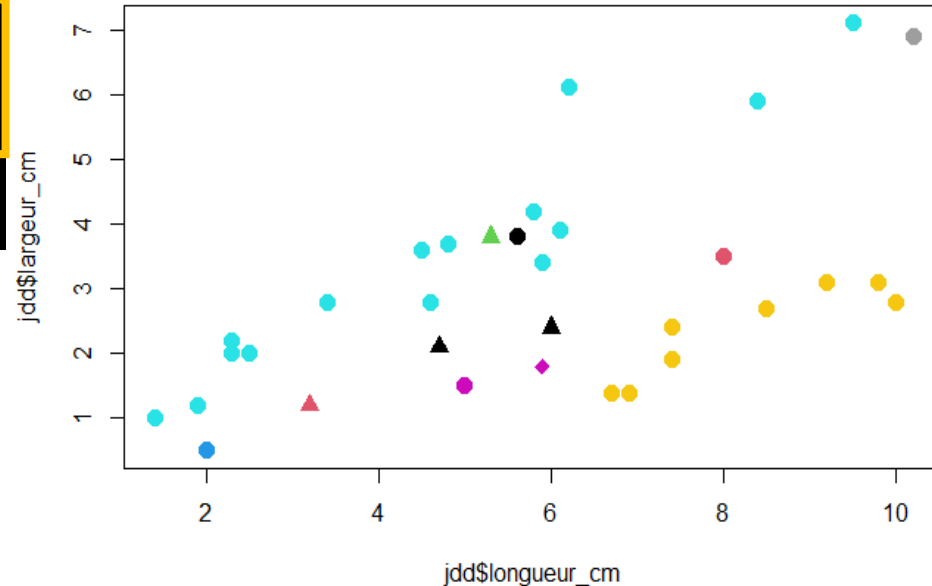
# L'environnement de travail « R »

## Importer, afficher et explorer des données

- Créer une figure simple

```
> plot(x = jdd$longueur_cm, y = jdd$largeur_cm,  
+      col = jdd$type_fac,  
+      pch = c(16, 17, 18)[as.numeric(jdd$silex_fac)],  
+      cex = 1.5,  
+      xlab = "Longueur (cm)", ylab = "Largeur (cm)",  
+      main = "Dimension des objets en silex")
```

- > Afficher les points
- + Colorer les points en fonction du type d'objet
- + Modifier la forme des points en fonction du type de silex
- + **Augmenter la taille des points**



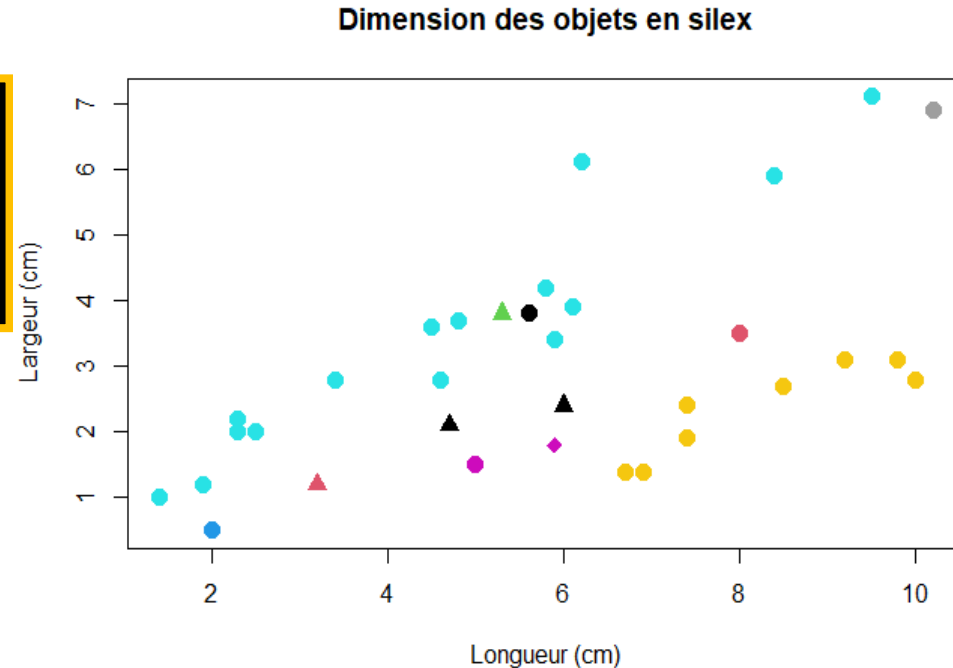
# L'environnement de travail « R »

## Importer, afficher et explorer des données

- Créer une figure simple

```
> plot(x = jdd$longueur_cm, y = jdd$largeur_cm,  
+       col = jdd$type_fac,  
+       pch = c(16, 17, 18)[as.numeric(jdd$silex_fac)],  
+       cex = 1.5,  
+       xlab = "Longueur (cm)", ylab = "Largeur (cm)",  
+       main = "Dimension des objets en silex")
```

- > Afficher les points
- + Colorer les points en fonction du type d'objet
- + Modifier la forme des points en fonction du type de silex
- + Augmenter la taille des points
- + **Ajouter des titres d'axes et de graphique**




# L'environnement de travail « R »

## Automatisation : le concept de « boucle » en programmation

Objectif : **répéter** la même opération un nombre  $i$  de fois (boucle "for") ou tant qu'une condition est/n'est pas remplie (boucle "while")

```
322  ## créons un objet x contenant la valeur "0".
323  x <- 0
324
325  ## Nous allons y ajouter des valeurs à l'aide d'une boucle :
326
327  i <- 1 # tout d'abord, on initialise la valeur i
328
329  for(i in 1:5) # conditions de la boucle : tant que i se trouve dans ("in")
330                # l'intervalle 1 à 5 (défini par ":"), répéter le code suivant.
331  { # on ouvre la boucle
332    x <- c(x, i) # on place la valeur i dans l'objet x.
333
334    i <- i + 1 # on incrémente i de 1
335  }
336
337  ## Affichons le résultat :
338  x
```



```
[1] 0 1 2 3 4 5
```

# L'environnement de travail « R »

## Automatisation : le concept de « boucle » en programmation

Ce qu'il se passe dans la machine...

```
322 ## créons un objet x
323 x <- 0
324
325 ## Nous allons y ajout
326
327 i <- 1 # tout d'abord
328
329 for(i in 1:5) # condi
330             # l'int
331 { # on ouvre la bou
332   x <- c(x, i) # on p
333
334   i <- i + 1 # on inc
335 }
336
337 ## Affichons le résul
338 x
```

```
0 > x = 0
      i = 1
```



# L'environnement de travail « R »

## Automatisation : le concept de « boucle » en programmation

Ce qu'il se passe dans la machine...

```
322 ## créons un objet x
323 x <- 0
324
325 ## Nous allons y ajout
326
327 i <- 1 # tout d'abord
328
329 for(i in 1:5) # condi
330             # l'int
331 { # on ouvre la bou
332   x <- c(x, i) # on p
333
334   i <- i + 1 # on inc
335 }
336
337 ## Affichons le résul
338 x
```

```
0 > x = 0
      i = 1
```

```
1 > x = c(x, i) = 0 1
      i = i + 1 = 2
```



# L'environnement de travail « R »

## Automatisation : le concept de « boucle » en programmation

Ce qu'il se passe dans la machine...

```
322 ## créons un objet x
323 x <- 0
324
325 ## Nous allons y ajout
326
327 i <- 1 # tout d'abord
328
329 for(i in 1:5) # condi
330               # l'int
331 { # on ouvre la bou
332   x <- c(x, i) # on p
333
334   i <- i + 1 # on inc
335 }
336
337 ## Affichons le résul
338 x
```

0 > x = 0  
i = 1

1 > x = c(x, i) = 0 1  
i = i + 1 = 2

2 > x = c(x, i) = 0 1 2  
i = i + 1 = 3



# L'environnement de travail « R »

## Automatisation : le concept de « boucle » en programmation

Ce qu'il se passe dans la machine...

```
322 ## créons un objet x
323 x <- 0
324
325 ## Nous allons y ajout
326
327 i <- 1 # tout d'abord
328
329 for(i in 1:5) # condi
330               # l'int
331 { # on ouvre la bou
332   x <- c(x, i) # on p
333
334   i <- i + 1 # on inc
335 }
336
337 ## Affichons le résul
338 x
```

0 > x = 0  
i = 1

1 > x = c(x, i) = 0 1  
i = i + 1 = 2

2 > x = c(x, i) = 0 1 2  
i = i + 1 = 3

3 > x = c(x, i) = 0 1 2 3  
i = i + 1 = 4



# L'environnement de travail « R »

## Automatisation : le concept de « boucle » en programmation

Ce qu'il se passe dans la machine...

```
322 ## créons un objet x
323 x <- 0
324
325 ## Nous allons y ajout
326
327 i <- 1 # tout d'abord
328
329 for(i in 1:5) # condi
330               # l'int
331 { # on ouvre la bou
332   x <- c(x, i) # on p
333
334   i <- i + 1 # on inc
335 }
336
337 ## Affichons le résul
338 x
```

0 > x = 0  
i = 1

1 > x = c(x, i) = 0 1  
i = i + 1 = 2

2 > x = c(x, i) = 0 1 2  
i = i + 1 = 3

3 > x = c(x, i) = 0 1 2 3  
i = i + 1 = 4

4 > x = c(x, i) = 0 1 2 3 4  
i = i + 1 = 5





# L'environnement de travail « R »

## Automatisation : le concept de « boucle » en programmation

Ce qu'il se passe dans la machine...

```
322 ## créons un objet x
323 x <- 0
324
325 ## Nous allons y ajout
326
327 i <- 1 # tout d'abord
328
329 for(i in 1:5) # condi
330               # l'int
331 { # on ouvre la bou
332   x <- c(x, i) # on p
333
334   i <- i + 1 # on inc
335 }
336
337 ## Affichons le résul
338 x
```

0 > x = 0  
i = 1

1 > x = c(x, i) = 0 1  
i = i + 1 = 2

2 > x = c(x, i) = 0 1 2  
i = i + 1 = 3

3 > x = c(x, i) = 0 1 2 3  
i = i + 1 = 4

4 > x = c(x, i) = 0 1 2 3 4  
i = i + 1 = 5

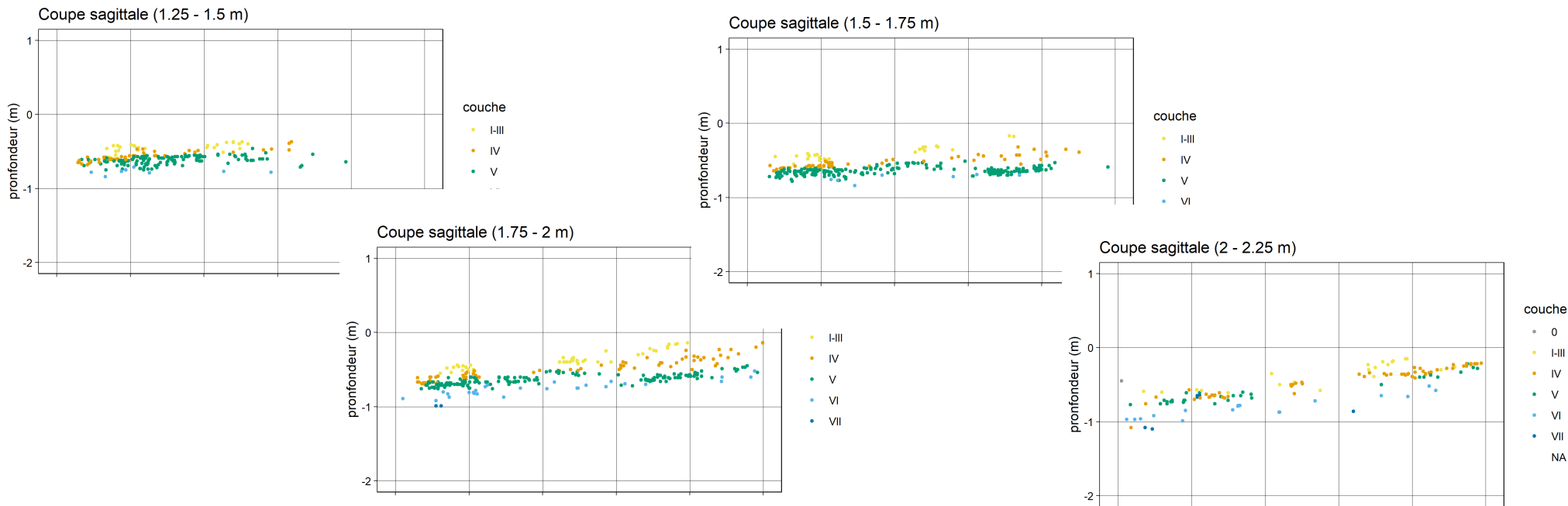
5 > x = c(x, i) = 0 1 2 3 4 5  
i = i + 1 = 6



# L'environnement de travail « R »

## Automatisation : le concept de « boucle » en programmation

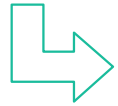
Utile pour répéter la même operation sur plusieurs données similaires !



# Place à la pratique !

## Le TP « R pour archéologues » de l'ED4

<https://github.com/ALVignoles/R-pour-archeologues>



Dossier « ED4-Bruxelles »



TP-ED4.html

data\_exemple\_ext.csv

