Session 1 : R base

en explorant des données omiques

Teachers: Anne Badel, Claire Vandiedonck
Helpers: Antoine Bridier-Nahmias, Clémence Réda, Bruno Toupance, Jacques van Helden

Plan de la session 1: R base

1. Start-R: introduction au module 3

2. Vérification et consolidation des pré-requis:
   a. Session R
   b. Vecteurs
   c. Matrices

Objectifs d'apprentissage de la session 3

3. Dataframes
   a. Créer un dataframe
   b. Extraire des données d'un dataframe
   c. Manipuler un dataframe: filtrer, créer des sous-dataframes, fusionner des dataframes

=> Travail personnel ce vendredi:
   a. consolidation des bases de R:
      - vecteurs, matrices, dataframes
      - statistiques descriptives et figures de base selon le type de données
   b. facteurs: découverte par un tutorial
   c. listes: diapo et tutorial

1. Start-R

introduction au module 3 « R – stats »

Equipe pédagogique

Responsables :

Claire Vandiedonck
(MCF UP)

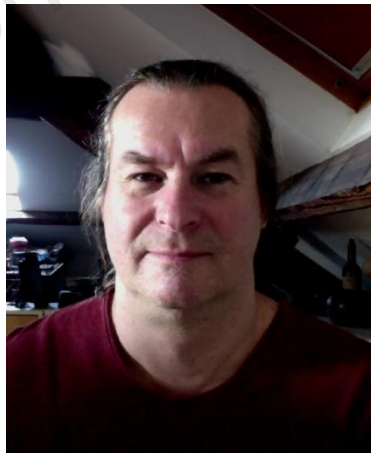
Jacques van Helden
(Pr AMU, directeur IFB)

Instructeurs:

Anne Badel
(MCF UP),

Antoine Bridier-Nahmias
(MCF UP)

Magali Berland
(IR INRA MétaGénoPolis),



Olivier Sand
(IR CNRS, IFB-core)

Helpers:

Bruno Toupance
(MCF UP),

Clémence Réda
(DCEM UP),

Yves Clément
(MCF UP),

Olivier Taboureau
(Pr UP)

Planning du module 3 R-Stats

Jour	Horaire	Description	Instructeurs	Helpers
3 mars	9h30 - 12h30	**R base** - Manipuler les différents types d'objets dans R - Statistiques descriptives - Figures de base	*Anne Badel* *Claire Vandiedonck*	*Antoine-Bridier-Nahmias* *Bruno Toupance* *Clémence Réda* *Jacques van Helden*
4 mars	13h30 - 16h30	**Renforcement de R:** - Contrôles de flux: exécutions conditionnelles, boucles - Paquets, écrire ses propres fonctions - Customiser ses figures avec R base - Introduction à dplyr/tidyverse/ggplot	*Magali Berland* *Claire Vandiedonck*	*Antoine-Bridier-Nahmias* *Yves Clément* *Bruno Toupance* *Jacques van Helden*
9 mars	14h30 - 17h30	**Statistiques pour les données à haut débit** **RStudio et R markdown**	*Antoine Bridier-Nahmias* *Claire Vandiedonck*	*Anne Badel* *Clémence Réda* *Olivier Sand* *Jacques van Helden*
11 mars	9h00 - 12h00	**Régression linéaire** **Corrélation** **Exploration de données multidimensionnels (ACP/MDS)**	*Magali Berland* *Jacques van Helden*	*Anne Badel* *Clémence Reda* *Olivier Taboureau* *Claire Vandiedonck*
29 mars	10h00 - 13h00	**Classification supervisée et apprentissage**	*Olivier Sand* *Jacques van Helden*	*Anne Badel* *Olivier Taboureau* *Bruno Toupance* *Claire Vandiedonck*
29 mars	14h30 - 17h30	**Classification non supervisée (clustering)** **Analyse d'enrichissment**	*Anne Badel* *Olivier Sand* *Jacques van Helden*	*Yves Clément* *Olivier Taboureau* *Bruno Toupance* *Claire Vandiedonck*

Site web du cours

https://du-bii.github.io/module-3-Stat-R/

module-3-Stat-R

Analyse statistique avec R

View the Project on GitHub
DU-Bii/module-3-Stat-R

This project is maintained by DU-Bii

Hosted on GitHub Pages — Theme by orderedlist

DUBii - module 3 - Analyse statistique avec R

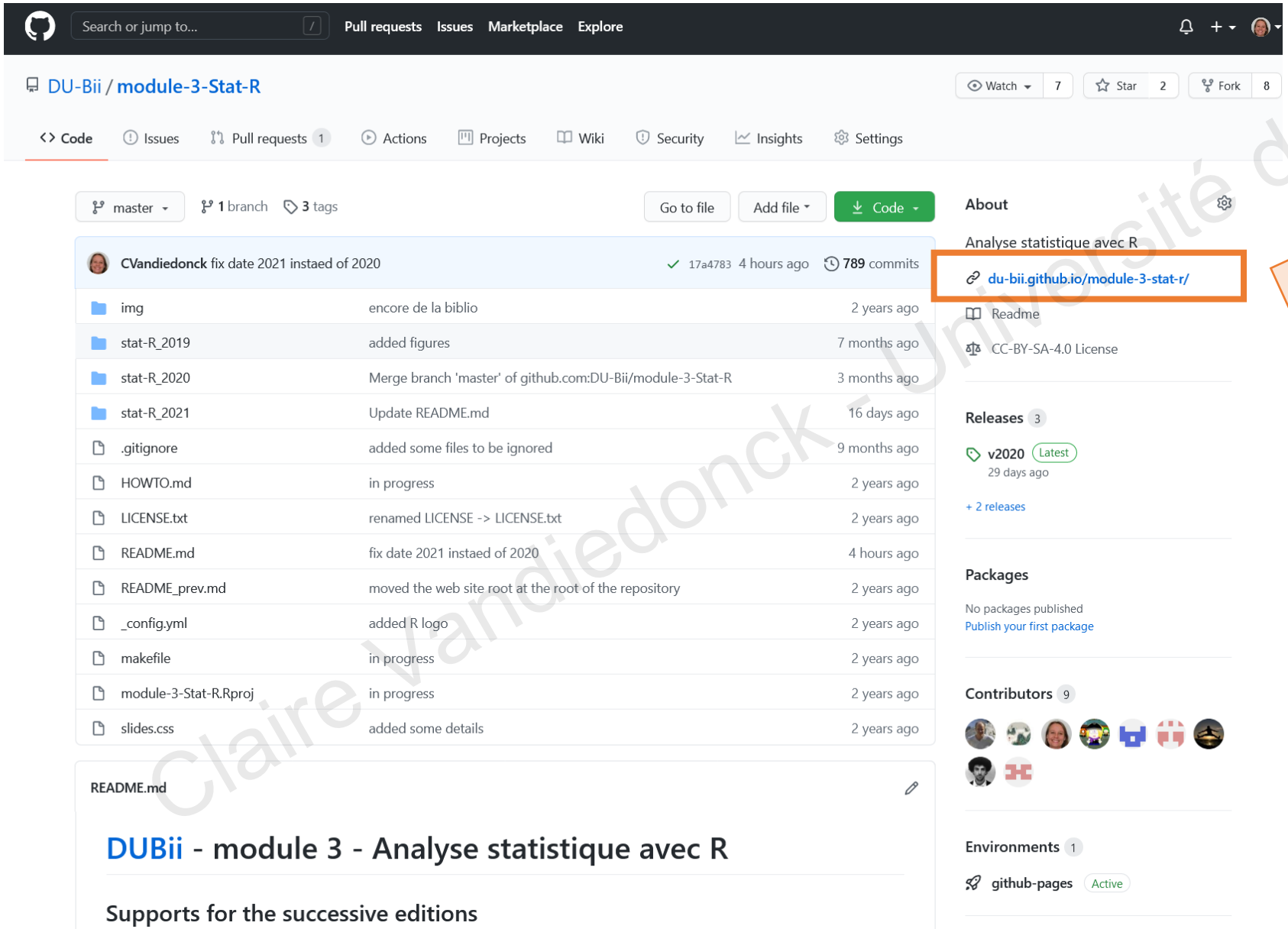
Supports for the successive editions

Edition	Site
2021	stat-R_2021/
2020	stat-R_2020/
2019	stat-R_2019/

Links

Doc	Description	URL
Git pages	**Web site of the course** (to see the supports)	https://du-bii.github.io/module-3-Stat-R/
Git repo	Repository enabling to download or clone the teaching material on your computer	https://github.com/DU-Bii/module-3-Stat-R

Retour vers le site web depuis le dépôt Github du DU



Pas de panique si vous avez atterri sur cette page via google, cliquez juste ici et vous reviendrez sur la page web correspondante!

Modalités de contrôle des connaissance

Travail personnel sous forme d'atelier (15%)

1. rendu individuel (7.5%)pour le **19/03** -

2. évaluation par les pairs (7.5%) pour le **02/04** -

Assiduité (25%)

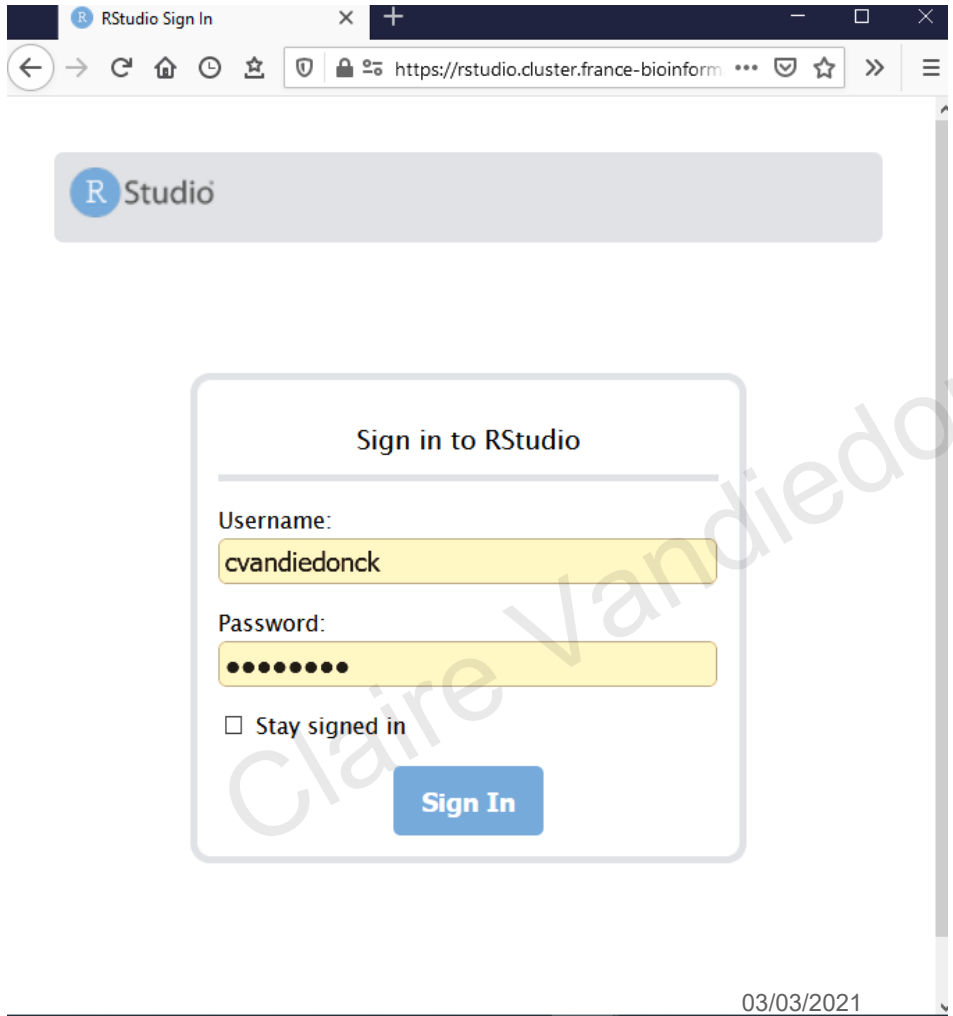
- **29/03** matin QCM sur moodle (10%)

Mini-projet (50%) pour le **10/05** -

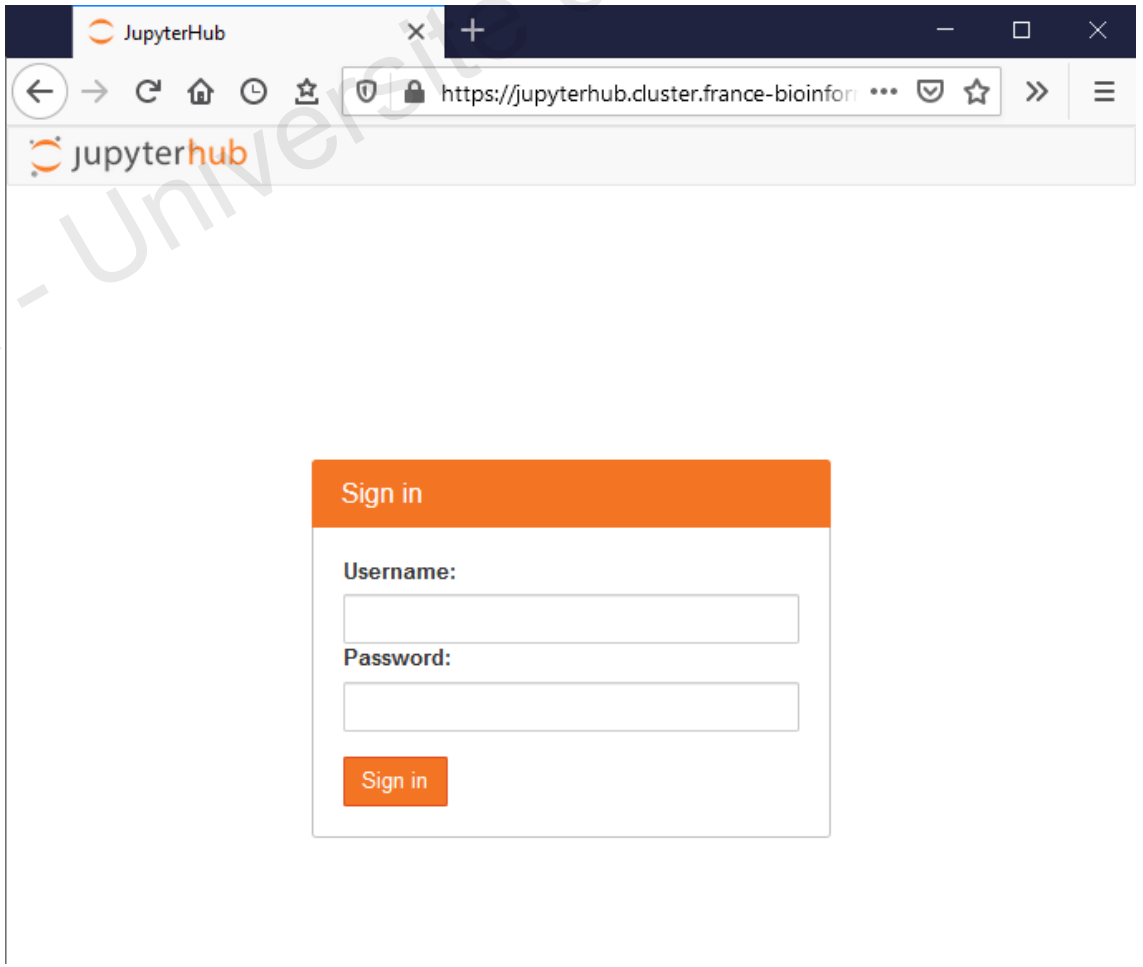
Ressources de l'IFB pour R:

Deux plateformes avec vos mêmes identifiants!

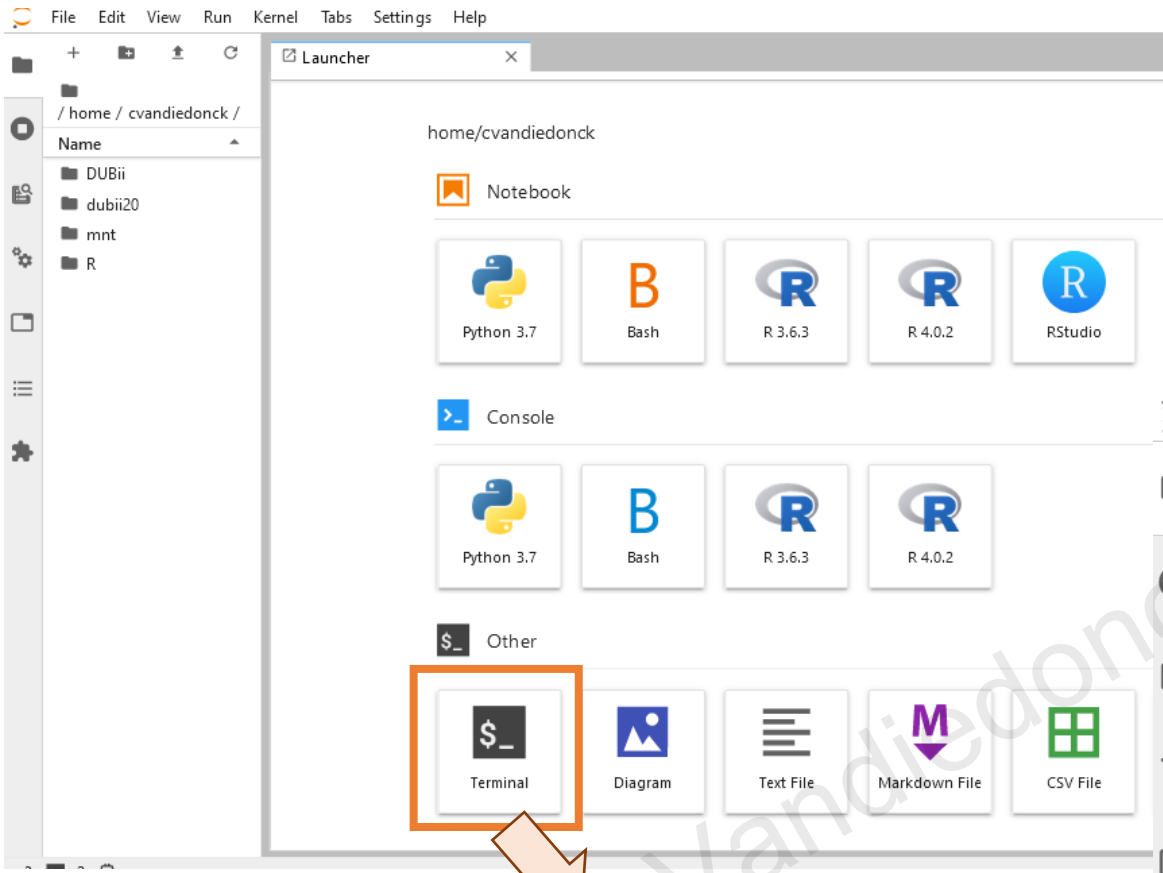
https://rstudio.cluster.france-bioinformatique.fr/



https://jupyterhub.cluster.france-bioinformatique.fr/



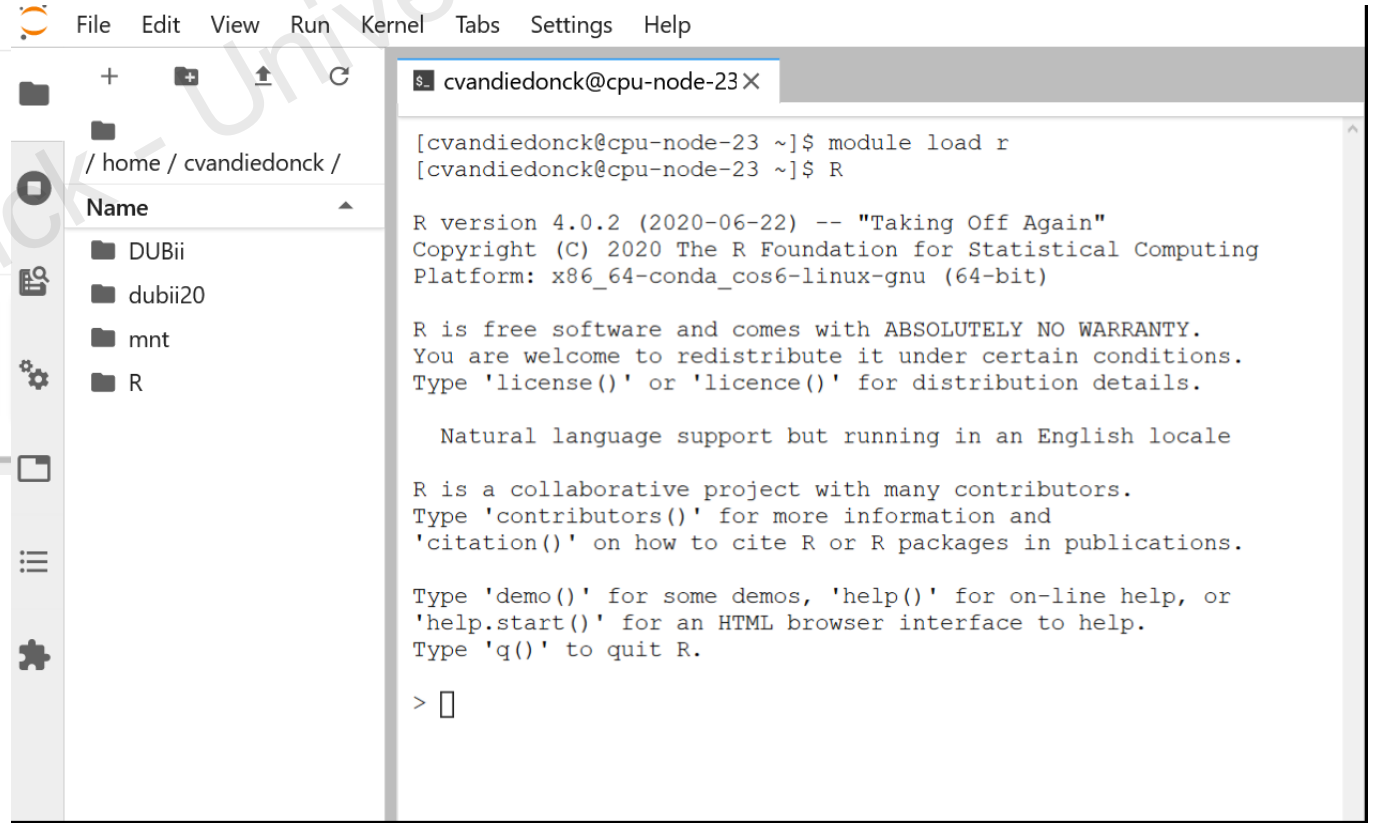
4 façons de faire du R sur le Jupyter lab



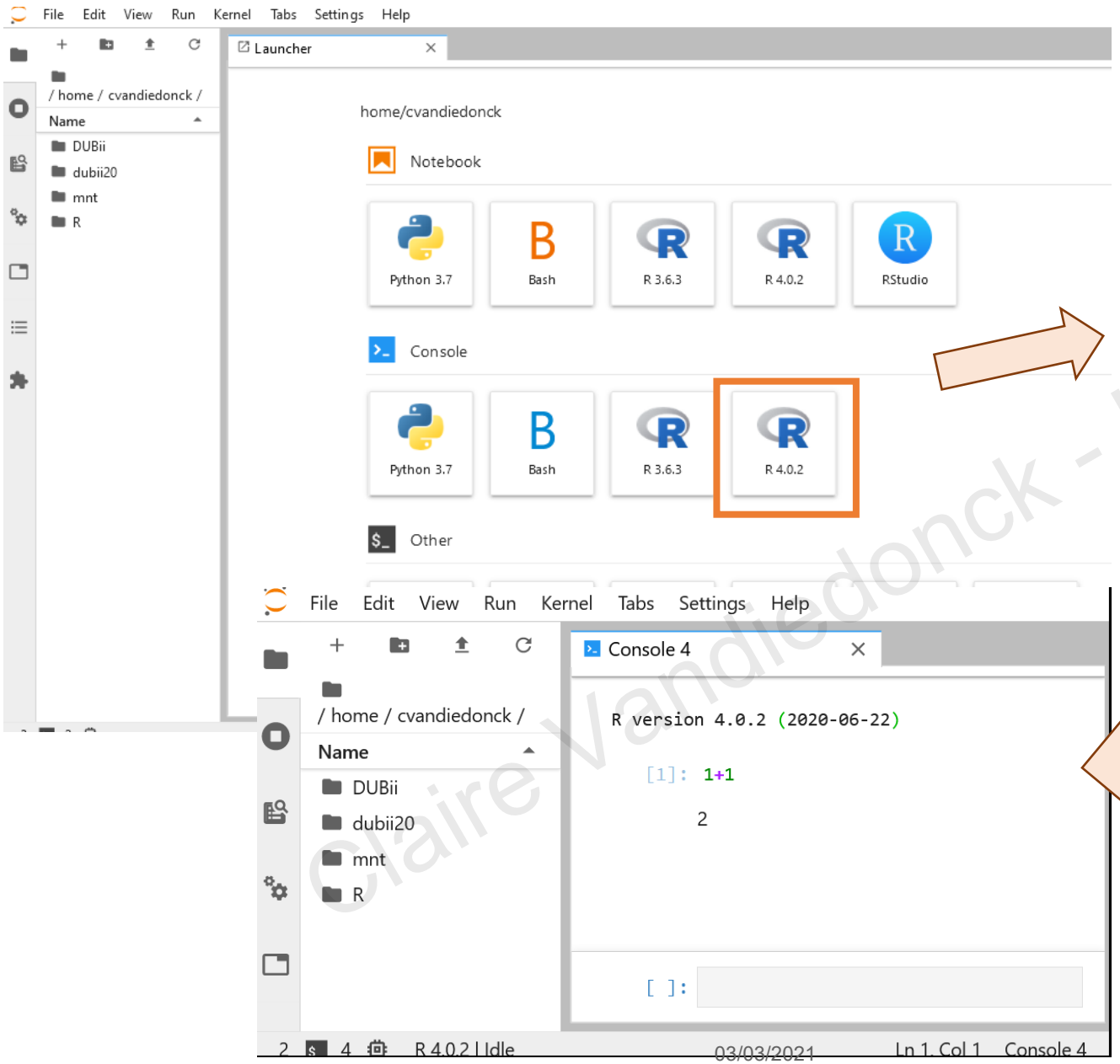
1. Avec le terminal:

$ module load r

$ R



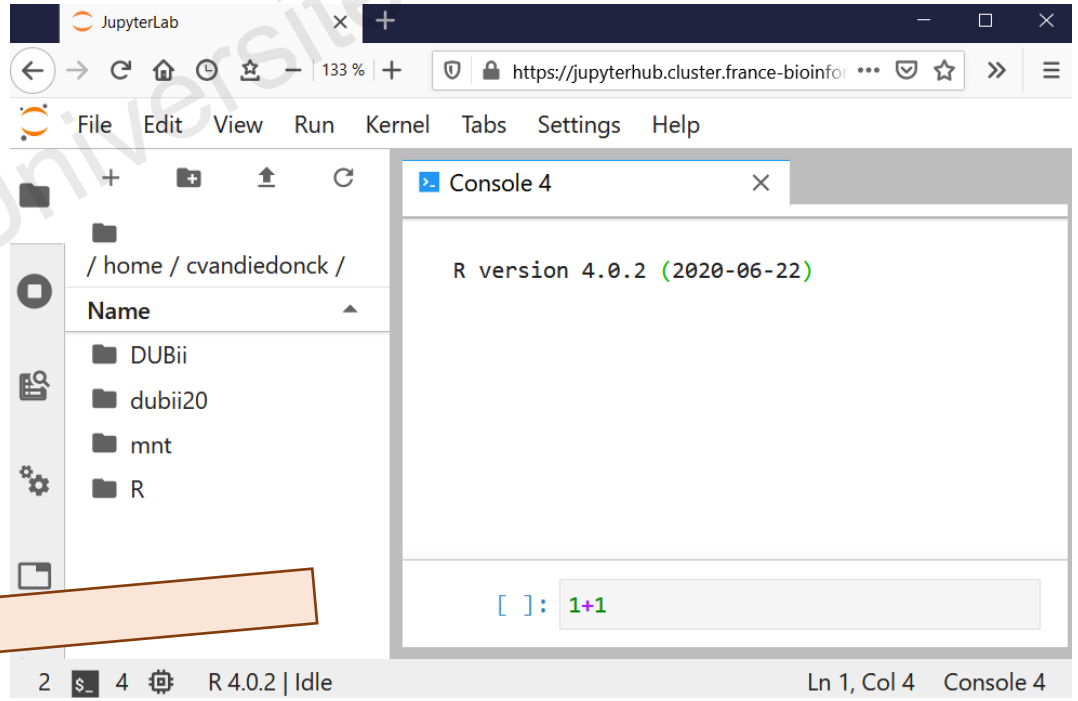
4 façons de faire du R sur le Jupyter lab



2. Avec une console

on entre les commandes dans la cellule en bas

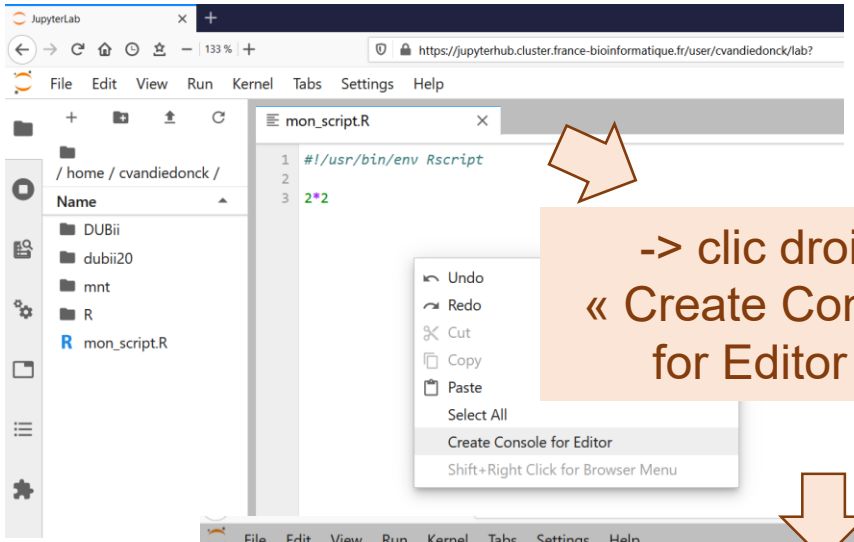
puis shift + Enter pour exécuter la commande



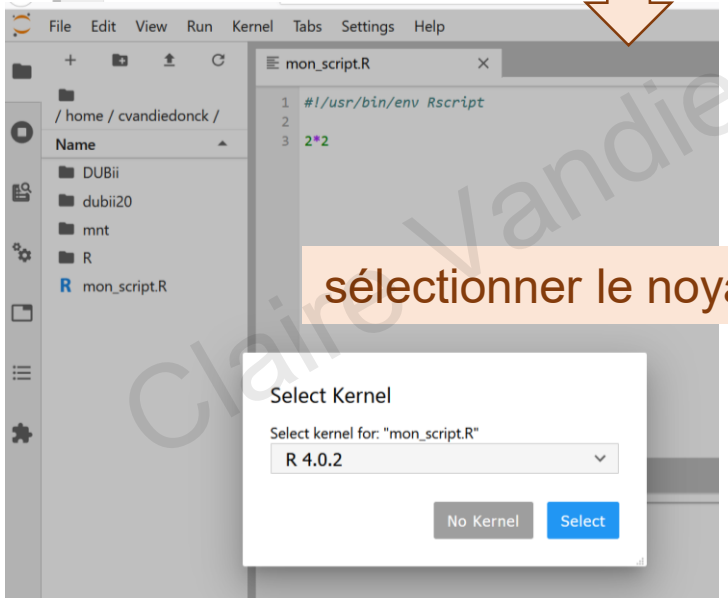
le résultat apparait dans la fenêtre supérieure

4 façons de faire du R sur le Jupyter lab

Ouvrez un script

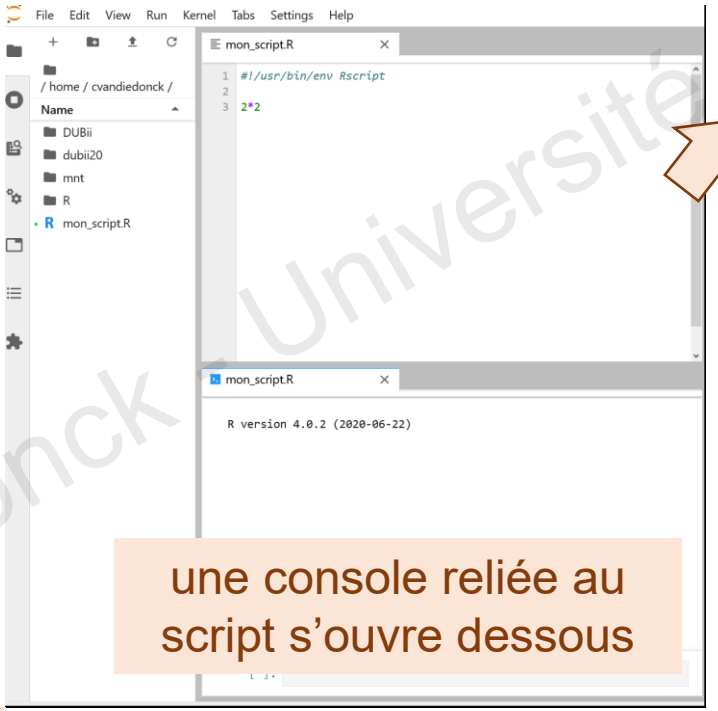


-> clic droit : « Create Console for Editor »



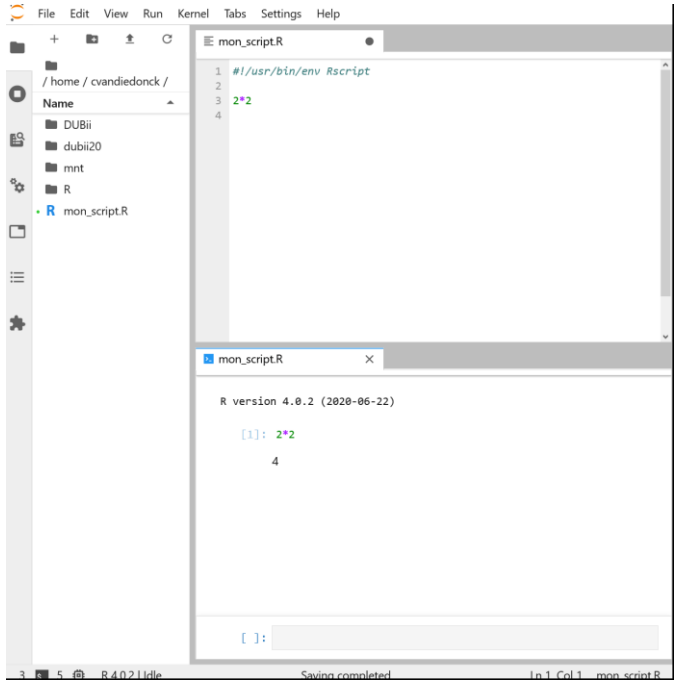
sélectionner le noyau R

2. Avec une console et un script!

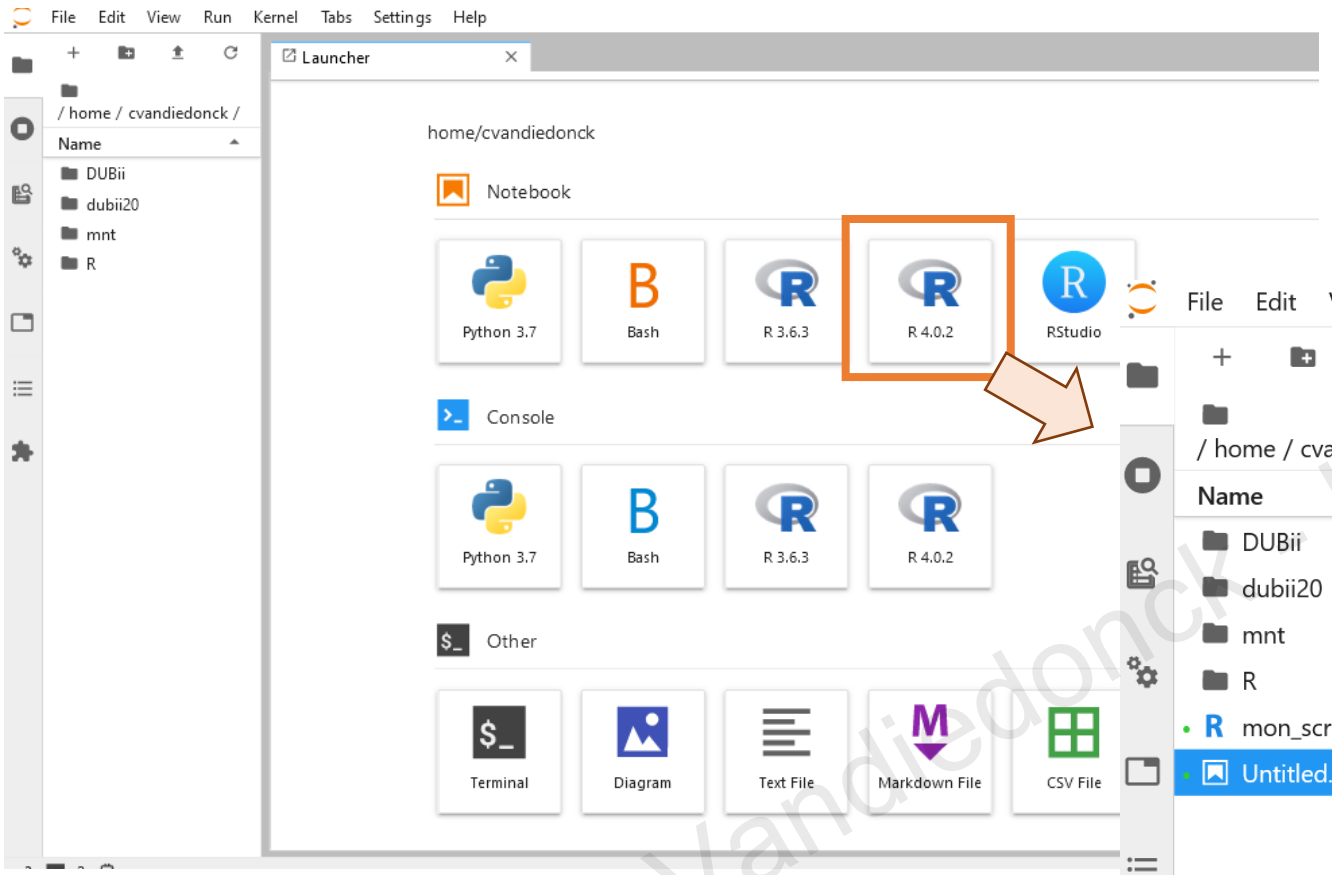


une console reliée au script s'ouvre dessous

exécuter ensuite les commandes du script dans la console avec shift+Entrée

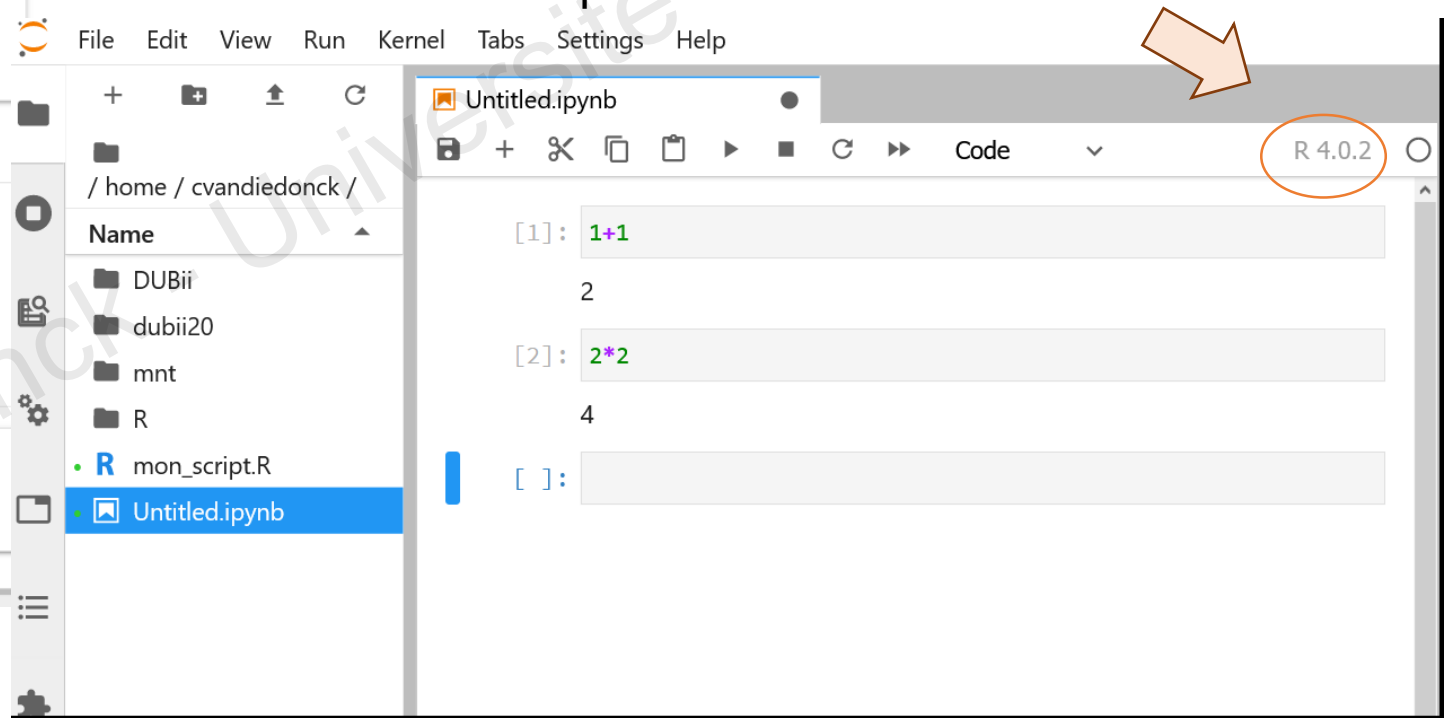


4 façons de faire du R sur le Jupyter lab



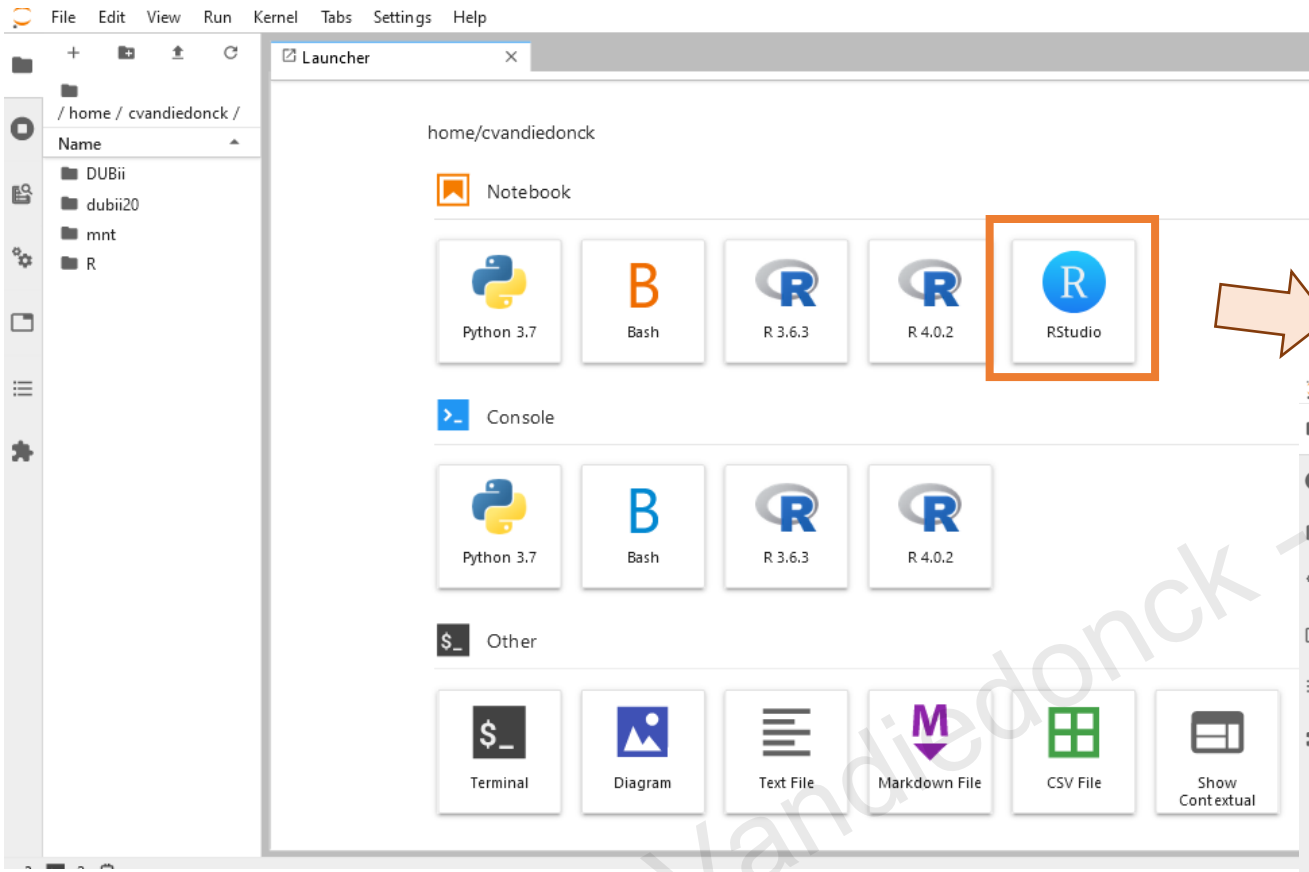
3. Avec un notebook R:

le noyau utilisé est indiqué dans le coin supérieur droit

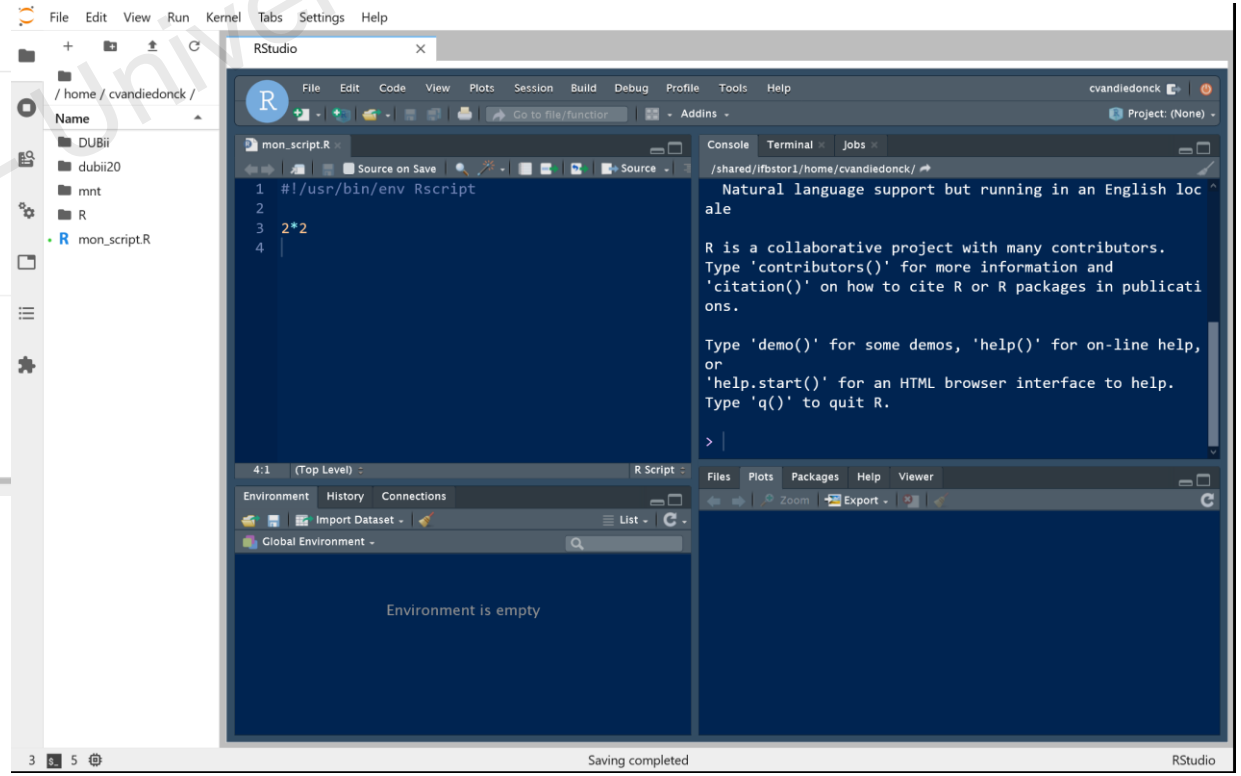


exécutez et affichez le résultat de vos cellules une par une directement dans le notebook!

4 façons de faire du R sur le Jupyter lab



4. Avec une tabulation Rstudio:



Sondage wooclap -> R et vous?

AIFGCO

2. Vérification et consolidation des pré-requis:

Session R
Vecteurs
Matrices

Rappel sur une Session R

sessionInfo()	R version, core packages , version of additional loaded packages
getwd()	get the working directory of the R session on my computer
setwd()	set the working directory of the R session on my computer
ls()	list objects and functions present in my R session
list.files() or dir()	list files and folders in the given directory (by default the working directory) on my computer
save()	to save some objects/functions of my R session on the computer
save.image()	to save all objects/functions of my R session on the computer
data/"data"	no quotes if the object is in the R Session, quotes if calling a file which is not in the R Session

Caution when naming R objects/variables: no accent, no special characters (like "-"), cannot start with a number…
cf. https://google.github.io/styleguide/Rguide.xml

Rappel sur les types d'objets

Main data structures

object	Heterogeneous = several types may coexist
vector	no
matrix	no
dataframe	yes
list	yes

Summary on vectors

Format	one-dimension
Datatype	homogeneous: only one type of character, numeric, logical, factor… -> ceorcion if heterogeneous - check with class() or mode() - checking type with is.num() , is.character() , … - conversion with as.num() , as.character() , …
Creation	c() , : , seq() , rep() , sample() , rnorm() , …
Adding new items	c(), append()
Size	length(), nchar()
Slicing	my_vector[i]
Filling	my_vector[i] <- "toto"
Naming	names()

Let's play together!

Exercice sur les vecteurs

Pour chaque question posée, entrez vos solutions de code (ou likez celle de vos collègues):

1. Créer un vecteur avec: 5 4 3 2 1 et 0. J'attends au moins 3 façons possibles.

2. Créer un vecteur avec: 1_impair, 2_pair, 3_impair, 4_pair, 5_impair, 6_pair

3. Ecrire dans R « Il y a 26 lettres dans l'alphabet » en codant la valeur 26

4. Comptez le nombre de caractères de la phrase en Q3

5. Ecrire la même phrase en lettres capitales.

6. Créez un **vecteur x** contenant 100 entiers tirées aléatoirement avec remise de manière équiprobable à partir des valeurs -10 à 10 et de 10 valeurs manquantes.

7. Quelle commande vous permet de compter le nombre de données manquantes dans x?

8. Calculez la moyenne sur les 80 1ères valeurs

Summary on matrices

Format	two-dimensions
Datatype	class() to check it is a matrix homogeneous: only one type of character, numeric, logical, factor -> ceorcion if heterogeneous -> check with mode()
Creation	matrix() , cbind() , rbind()
Adding new items	cbind() , rbind()
Size	length() -> nb of items
Dim	dim(), str()
Slicing	my_matrix [i,j]
Filling	my_matrix[i,j] <- "toto"
Naming	colnames() , rownames()

Let's play together!

Exercice sur les matrices

entrez vos solutions de code ou réponses (ou likez celle de vos collègues):

1. Créer un vecteur **myVector** avec 12 entiers de 1 à 12
2. Regarder la taille puis la dimension de **myVector**
3. Imposer des dim 3 et 4 au vecteur **myVector**
4. Quelle est à présent la classe de **myVector**
5. Ajouter une colonne avec un vecteur avec les chaînes de caractères: « one », « two » et « three »
6. Qu'est-il advenu aux valeurs numériques?
7. Que faire? Un dataframe
8. Renommer les colonnes de A à F avec une constante

3. dataframes:

Support pédagogique

Des diapos ci après pour mémo/archivage

Un tutoriel avec un jupyter notebook reprenant les exemples des diapos

« **Rsession1_tuto_dataframes.ipynb** » sur github

ou dans /shared/projects/dubii2021/trainers/module3/tutorials

Un practical dans un jupyter notebook avec des données omiques

« **Rsession1_practicals_dataframes.ipynb** » sur github

ou dans /shared/projects/dubii2021/trainers/module3/practicals

Dataframe

Dataframe = two-dimensional object that can be heterogeneous,

↳Create a dataframe with function data.frame()

```
data.frame(..., row.names = NULL, check.rows = FALSE,
           check.names = TRUE, fix.empty.names = TRUE,
           stringsAsFactors = default.stringsAsFactors())
```

Dataframe created with existing vectors

↳ Create a dataframe with function data.frame()

```
> myDataf <- data.frame(weight, size, bmi)

> myDataf          # it looks pretty much like the matrix myData2
                weight size bmi
Fabien          60 1.75 19.59184
Pierre          72 1.80 22.22222
Sandrine        57 1.65 20.93664
Claire          90 1.90 24.93075
Bruno           95 1.74 31.37799
Delphine        72 1.91 19.73630
> class(myDataf)# but this is well a dataframe and not a matrix
[1] "data.frame"
> str(myDataf)      # this one is a homogeneous df with numeric vectors
'data.frame':  6 obs. of  3 variables:
 $ weight: num  60 72 57 90 95 72
$ size  : num  1.75 1.8 1.65 1.9 1.74 1.91
 $ bmi   : num  19.6 22.2 20.9 24.9 31.4 ...
> dim(myDataf)
[1] 6 3
```

Important:
If vectors are character chains, use stringsAsFactors= FALSE to avoid their conversion into factors

in R>4, the default is now FALSE !!!

Creating an empty dataframe

creating an empty dataframe?

```
> d <- data.frame()
> d
data frame with 0 columns and 0 rows
> dim(d)
[1] 0 0
```

BUT USELESS : impossible to fill!

Better way: converting a matrix in a dataframe with function as.data.frame()

```
> class(myData2)
[1] "matrix"
> class(as.data.frame(myData2))
[1] "data.frame"
```

```
> d <- as.data.frame(matrix(NA,2,3))
> d
  V1 V2 V3   # by default, col names are V1, V2, etc…
1 NA NA NA # while if you are using the function
2 NA NA NA # data.frame() and not as.dataframe(),
                #col names are called X1, X2, etc…
> dim(d)
[1] 2 3
> str(d)
'data.frame':   2 obs. of  3 variables:
 $ V1: logi  NA NA
 $ V2: logi  NA NA
 $ V3: logi  NA NA
```

You may also use as.data.frame on a matrix generated by binding rows or columns

```
> d2 <- as.data.frame(cbind(1:2, 10:11))
> str(d2)
'data.frame':   2 obs. of  2 variables:
 $ V1: int  1 2
 $ V2: int  10 11
```

Row/Column names of dataframes

↳ Either use same fonctions as for matrices
rownames() and colnames()

↳ Or better use the ones dedicated to dataframes: row.names() and names()

```
> row.names(d)
[1] "1" "2"
> names(d)
[1] "V1" "V2" "V3"
```

Important:
each row name
must be unique!

Note: data.frames are a special case of a list of variables of the same number of rows with unique row names

Extracting vectors from dataframes

Getting the vector corresponding to a column from a dataframe:

- either by specifying its index

```
> myDataf[,2]
[1] 1.75 1.80 1.65 1.90 1.74 1.91
```

- Or by giving its name within the " " inside the squared brackets

```
> myDataf[,"size"]
[1] 1.75 1.80 1.65 1.90 1.74 1.91
```

- Or by giving its name after the character « $ »

```
> myDataf$size
[1] 1.75 1.80 1.65 1.90 1.74 1.91
```

Extracting rows from dataframes

Getting a « dataframe » corresponding to a row from a dataframe:

- either by specifying its index

```
> myDataf
           weight size      bmi   sex
Fabien         60 1.75 19.59184   Man
Pierre         72 1.80 22.22222   Man
Sandrine       57 1.65 20.93664 Woman
Claire         90 1.90 24.93075 Woman
Bruno          95 1.74 31.37799   Man
Delphine       72 1.91 19.73630 Woman

> myDataf[2,]
           weight size bmi   sex
Pierre      72  1.8 22.22222 Man
```

- Or by giving its name within the " " inside the squared brackets

```
> myDataf["Pierre",]
           weight size bmi sex
Pierre      72  1.8 22.22222  Man
> class(myDataf["Pierre",])
[1] "data.frame"
```

- If you extract only one row, you do not get a vector but a dataframe.

=> To convert into a vector, use unlist()

```
> temp <- unlist(myDataf["Pierre",])
> class(temp)
[1] "character"
```

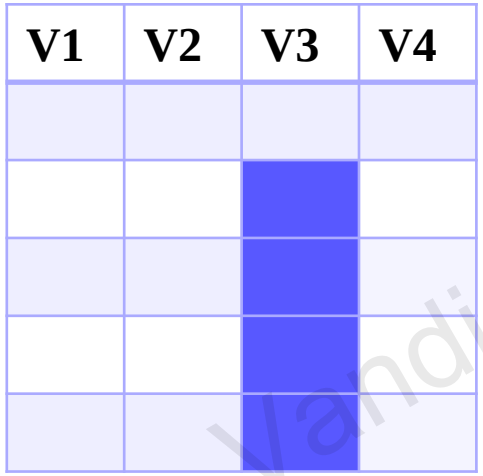
Let's summarize and give it a try

How do we create a dataframe?

Which are the three methods to slice datrames?

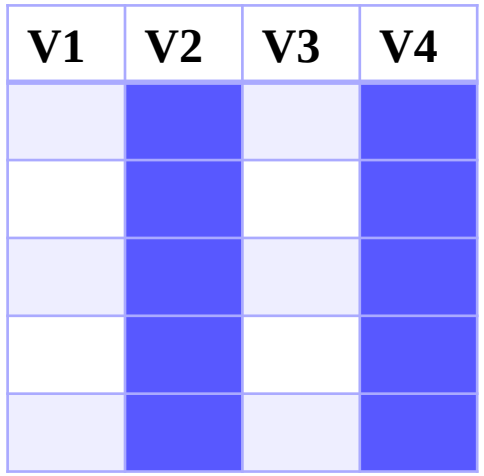
Which command should I use to extract the blue cells of the 3 dataframes below?

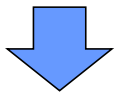
dataframe1



dataframe2

dataframe3





Now, how to extract the even columns if I have 500 000 columns?

Adding new vectors to a dataframe

Either enter **one vector at a time** as a new variable

```
my_dataframe$new_variable <- my_variable
```

```
> d2$new <- 1:2
V1  V2 new
1   1 10   1
2   2 11   2
```

Or **several vectors or subsets of dataframes** at once

- Using data.frame()

```
mynew_dataframe <- data.frame(data.frame1, data.frame2)
```

This method will keep the data types of each data.frame

```
> d3 <-  data.frame(d, d2)
  V1  V2  V3 V1.1 V2.1 new
1 NA NA NA    1   10   1
2 NA NA NA    2   11   2
```

- Using cbind() BUT to avoid ->prefer using data.frame()

```
mynew_dataframe <- cbind(data.frame1, data.frame2)
```

💣 BE CAREFULL : this method will keep the data types only if the data.frames 1 and 2 have several variables.
If they have only one, these variable will be converted as a vector and cbind() will convert character strings as factors.

```
> d4 <-  cbind(d, d2)
   V1  V2  V3 V1  V2 new
1 NA NA NA   1 10   1
2 NA NA NA   2 11   2
```

Reading a text file into R creates a dataframe

Read a text file using read.table():

```
> temperatures <- read.table("temperatures.txt", sep="\t", header=T, stringsAsFactors=F)
>temperatures
 Month Mean_Temp
1    January      2.0
2    February     2.6
3    March        7.9
4    April       11.2
5    May         15.3
6    June        22.2
7    July        22.9
8    August      22.5
9    September   17.3
10   October     11.7
11   November     5.2
12   December     2.8
> str(temperatures)              # the R object is a dataframe !!!!
'data.frame':   12 obs. of  2 variables:
$ Month    : chr  "January" "February" "March" "April" ...
$ Mean_Temp: num  2 2.6 7.9 11.2 15.3 22.2 22.9 22.5 17.3 11.7 ...
```

specify the field separator of the text file

TRUE if header in the text file

FALSE to avoid factorisation of character vectors

Reading a text file into R: to factorize or not to factorize?

💣Warning: use stringsAsFactors=F (now by default in R>4 but TRUE in previous versions) otherwise vectors of character values converted into factors -> see below, the Months were factorized!!!!

```
> temperatures <- read.table("Temperatures.txt", sep="\t", header=T, stringsAsFactors=T)
> str(temperatures)
'data.frame':   12 obs. of  2 variables:
 $ Month    : Factor w/ 12 levels "April","August",..: 5 4 8 1 9 7 6 2 12 11 ...
 $ Mean_Temp: num  2 2.6 7.9 11.2 15.3 22.2 22.9 22.5 17.3 11.7 ...

> levels(temperatures$Month) # the levels of the factor are in alphabetic order
 [1] "April"    "August"   "December" "February" "January"  "July"
 [7] "June"     "March"    "May"      "November" "October"  "September"
```

TRUE is by default in R<4

Factors in R

See tutorial_Factors_in_R.html (on GitHub)

Much care on:

- levels order
- coercion

=> ∈ travail personnel de vendredi

Reading a text file into R

Caution if:

- fewer names than columns in the header
- fewer columns than names in the header -> add argument fill=T to overcome the issue
- some rows with fewer columns -> add argument fill = T to overcome the issue
- using row.names=1 -> this cannot be used when several rows have the same name

Check the data.frame is as expected using:

str()

head() : displays the first 6 rows

tail() : displays the last 6 rows

and by displaying some rows in the middle of the file using their index

-> a general habit with any programmation language

Other functions: read.csv(), scan()

or *read.xlsx() or read_excel()* *to read worksheet from an exel file with library « xlsx » or « readxl »*

Saving a dataframe as a text file in the working directory

Saving a dataframe into a text file using write.table()

```
> write.table(myDataf, file="bmi_data.txt", sep="\t" , quote=F, col.names=T)
```

specify the name of the saved text file

Specify the column separator in the text file "\t" is for tabulation

FALSE to avoid quotes flanking strings of characters in the saved text file

to write the column names in a header

Filtering dataframes on criteria with which()

It generates a new dataframe

↳ use which() that returns the index of what is TRUE in the condition

```
> which ( myDataf$sex == "Woman")
[1] 3 4 6
> myDataf [ which ( myDataf$sex == "Woman") , ]
          weight size      bmi   sex
Sandrine      57 1.65 20.93664 Woman
Claire        90 1.90 24.93075 Woman
Delphine      72 1.91 19.73630 Woman
> str(myDataf [ which ( myDataf$sex == "Woman") , ])
'data.frame':   3 obs. of  4 variables:
 $ weight: num  57 90 72
 $ size  : num  1.65 1.9 1.91
 $ bmi   : num  20.9 24.9 19.7
 $ sex   : chr  "Woman" "Woman" "Woman"
```

💣**Important:**

you may enter this without including « which »
BUT this would not deal with NA values
=> **safer to use which()**

Or what does not match using "!=" for "different" or "!" for "not" before the test

```
> which ( myDataf$sex != "Man")
[1] 3 4 6
> which ( ! myDataf$sex == "Man")
[1] 3 4 6
```

What happens if I am not using which() ?

CAUTION! If you have NA values, not using which() will also return them!

↳ Let's make a copy of our dataframe and replace the gender of Claire by a missing value

```
> myDataf2 <- myDataf
> myDataf2["Claire", "sex"] <- NA

> myDataf [ myDataf$sex == "Woman", ]
           weight size bmi   sex
Sandrine       57 1.65 20.93664 Woman
Claire         90 1.90 24.93075 NA
Delphine       72 1.91 19.73630 Woman
> myDataf2 [ myDataf2$sex == "Woman", ]
            weight size       bmi   sex
Sandrine       57 1.65 20.93664 Woman
NA             NA   NA       NA  <NA>
Delphine       72 1.91 19.73630 Woman
 > myDataf2 [ which(myDataf2$sex == "Woman"), ]
            weight size bmi   sex
Sandrine       57 1.65 20.93664 Woman
Delphine       72 1.91 19.73630 Woman
```

Filtering dataframes on patterns with grep()

- use grep() that returns the index of what matches (even partially)

```
> grep("Wom", myDataf$sex)
[1] 3 4 6

> grep("Woman", myDataf$sex)
[1] 3 4 6

> myDataf [grep("Woman", myDataf$sex), ]
           weight size   bmi   sex
Sandrine       57 1.65 20.93664 Woman
Claire         90 1.90 24.93075 Woman
Delphine       72 1.91 19.73630 Woman

 > grep("a", row.names(myDataf)) # returns indexes of rows with an “a” in its name
[1] 1 3 4
> myDataf [grep("a", row.names(myDataf)),]
           weight size      bmi   sex
Fabien         60 1.75 19.59184   Man
Sandrine       57 1.65 20.93664 Woman
Claire         90 1.90 24.93075 Woman
```

Filtering/Subsetting dataframes on criteria with subset()

Subsetting the rows on the columns:

↳ use subset() : the easiest and most efficient way!

```
> WomenDataf <- subset(myDataf, sex== "Woman")
> WomenDataf
          weight size      bmi sex
Sandrine      57 1.65 20.93664 Woman
Claire        90 1.90 24.93075 Woman
Delphine      72 1.91 19.73630 Woman
```

Filtering dataframes on several criteria

logical: & = and, | = or, ! = not
comparison: ==, != (different), >, < , >=, <=
is an element of a vector using : %in%

Either with which()

```
> filteredData <- myDataf [ which ( myDataf$sex == "Woman" & myDataf$weight < 80 & myDataf$bmi > 20), ]

> filteredData                                    # only one woman with these criteria!
          weight size  bmi sex
Sandrine    57 1.65 20.93664  Woman
```

Or more easily with subset()

```
> subset( myDataf, sex == "Woman" & weight < 80 & bmi > 20)
          weight size       bmi    sex
Sandrine     57 1.65 20.93664 Woman
```

Merging dataframes

Merge two dataframes **with a key**

in this example I create a new column for the key
but we can also use an existing variable

```
> myDataf$index <- 1:6
> myDataf
          weight size    bmi   sex   index
Fabien        60 1.75 19.59184   Man     1
Pierre        72 1.80 22.22222   Man     2
Sandrine      57 1.65 20.93664 Woman     3
Claire        90 1.90 24.93075 Woman     4
Bruno         95 1.74 31.37799   Man     5
Delphine      72 1.91 19.73630 Woman     6

> OtherData <- data.frame(c(1:5, 7),rep(c("right-handed","left-handed"),3))
> names(OtherData) <- c("ID","handedness")
> OtherData
      ID   handedness
1  1 right-handed
2  2  left-handed
3  3 right-handed
4  4  left-handed
5  5 right-handed
6  7  left-handed
```

Merging dataframes

Merge two dataframes with a key

```
> myDataf$index <- 1:6
> myDataf
```

	weight	size	bmi	sex	index
Fabien	60	1.75	19.59184	Man	1
Pierre	72	1.80	22.22222	Man	2
Sandrine	57	1.65	20.93664	Woman	3
Claire	90	1.90	24.93075	Woman	4
Bruno	95	1.74	31.37799	Man	5
Delphine	72	1.91	19.73630	Woman	6

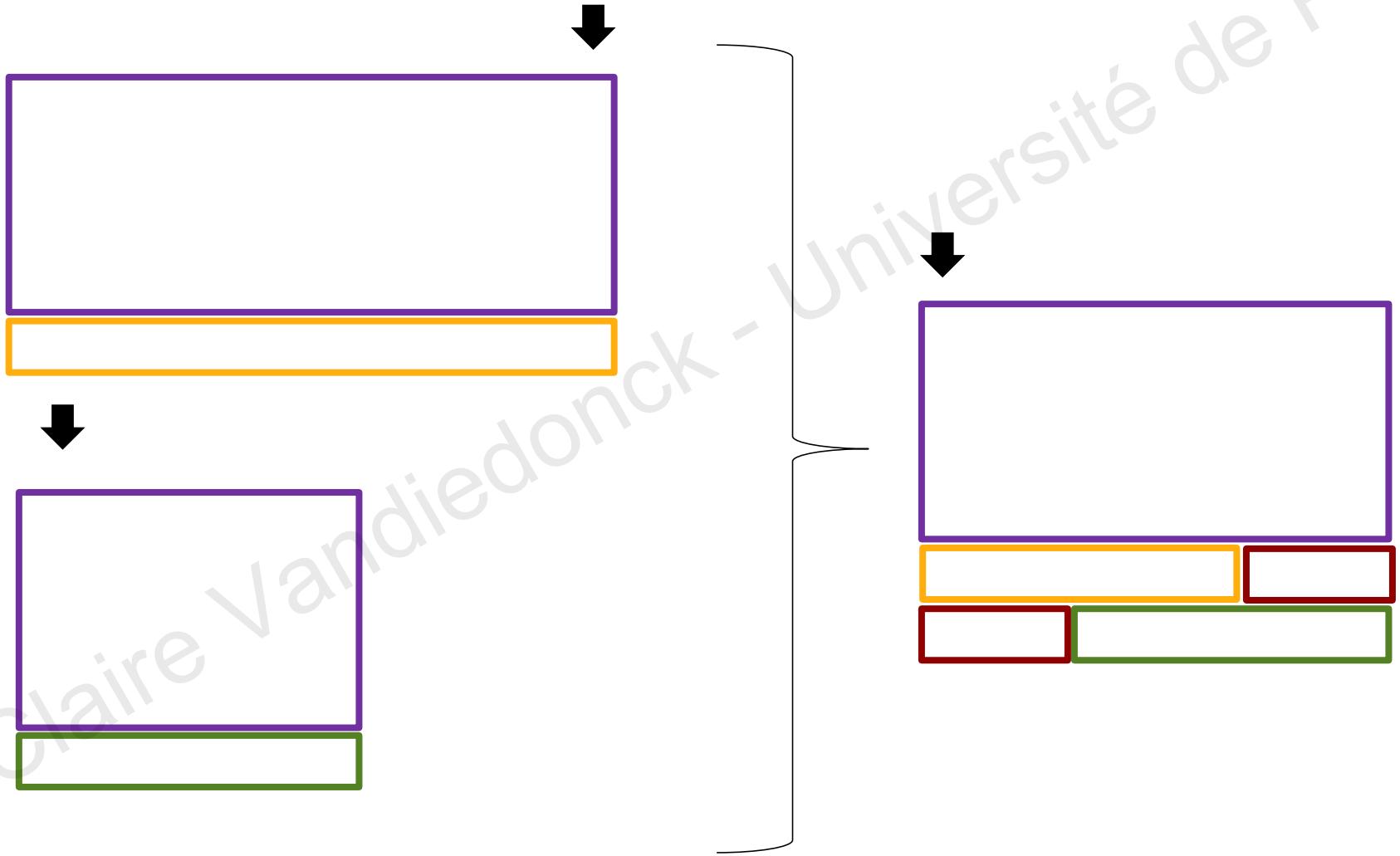
```
> OtherData <- data.frame(c(1:5, 7),rep(c("right-handed","left-handed"),3))
> names(OtherData) <- c("ID","handedness")
> OtherData
```

	ID	handedness
1	1	right-handed
2	2	left-handed
3	3	right-handed
4	4	left-handed
5	5	right-handed
6	7	left-handed

Merging dataframes

Merge two dataframes with a key

with merge(dataframe1,dataframe2,by="key" , all=T, sort=F)



Merged dataframe

Merge two dataframes with a key

```
> myMergedDataf <- merge(myDataf, OtherData, by.x="index", by.y="ID", all.x=T,
all.y=T, sort=F)
> myMergedDataf
```

	index	weight	size	bmi	sex	handedness
1	1	60	1.75	19.59184	Man	right-handed
2	2	72	1.80	22.22222	Man	left-handed
3	3	57	1.65	20.93664	Woman	right-handed
4	4	90	1.90	24.93075	Woman	left-handed
5	5	95	1.74	31.37799	Man	right-handed
6	6	72	1.91	19.73630	Woman	<NA>
7	7	NA	NA	NA	<NA>	left-handed

- ✓ unless the merge is done on the row.names(), the row.names of initial data.frames are lost -> the new data.frame has its own row names
- ✓ if two columns had the same name, a « .x » or a « .y » is added to the first/second

Summary on dataframes

Format	two-dimensions
Datatype	class() to check it is a dataframe hteterogeneous: type of columns/variable can differ
Creation	data.frame() with existing vectors, as.data.frame() when converting a matrix
Adding new items	my_dataf$new_var to add a new column, rbind() to add a new row AVOID using cbind() to add a columns -> use data.frame() instead
Dim	dim(), str(), nrows(), ncol()
Slicing	my_dataf[i,j] by indexes, my_dataf["my_row","my_var"] by name, my_dataf$my_var for a given columns
Filling	my_dataf[i,j] <- "toto" , etc…using slicing methods
Naming	names() , row.names()