Application sous R

Nadia Bessoltane - INRAe

École de bioinformatique IFB-INSERM-INRAe 2025

Rappel : les objets sous R

```
# variable : stocker des valeurs numériques ou des chaînes de caractères
> string <- “hello world”
> value  <- 10
```

```
# vecteur : stocker une liste de valeurs numériques ou de chaînes de caractères
> vect  <- c(1,2,11,12)
```

1	2	11	12

```
# matrice : stocker un tableau 2D de valeurs numériques ou de chaîne de caractères
> mat <- matrix(c(1,2, 11,12), nrow = 2, ncol = 2, byrow = TRUE)
```

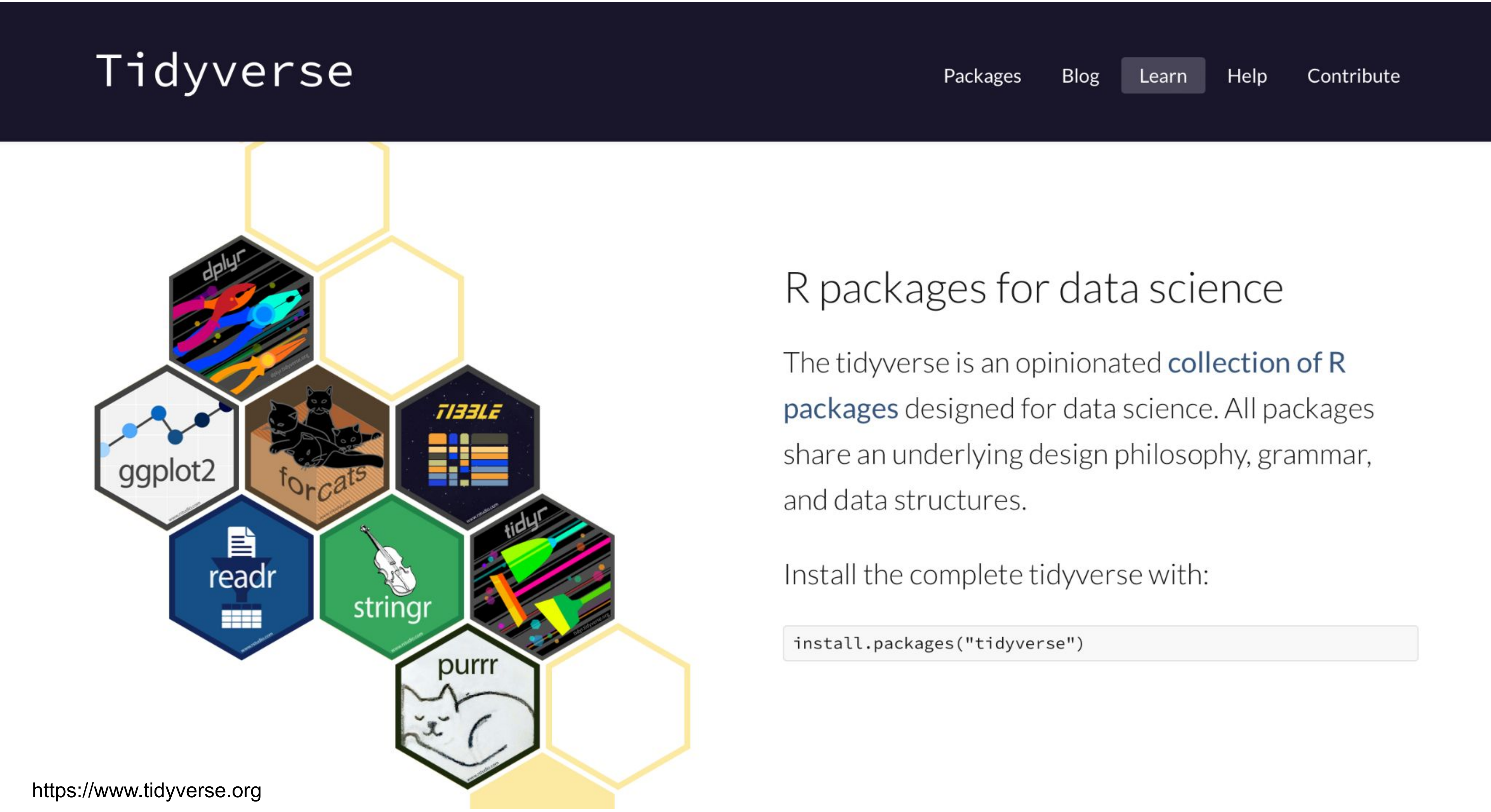
1	2
11	12

```
# data.frame : stocker des valeurs numériques et de chaînes de caractères
> df  <- data.frame(c(1,11), c("x", "y"))
```

1	x
11	y

```
# liste : stocker des objets de nature différente
> list <- list(vector = vect, matrix = mat, dataframe = df)
> list
```

```
# objet S3/S4 (POO) : stocker des données structurées
```



R packages for data science

The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

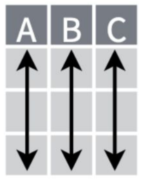
Install the complete tidyverse with:

```
install.packages("tidyverse")
```

https://www.tidyverse.org

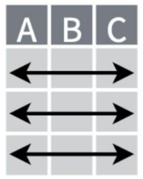
tidy data

is a way to organize tabular data in a consistent data structure across packages.

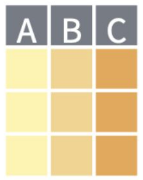


Each **variable** is in its own **column**

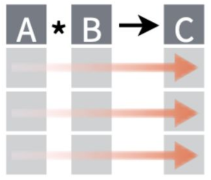
&



Each **observation**, or **case**, is in its own row



Access **variables** as **vectors**

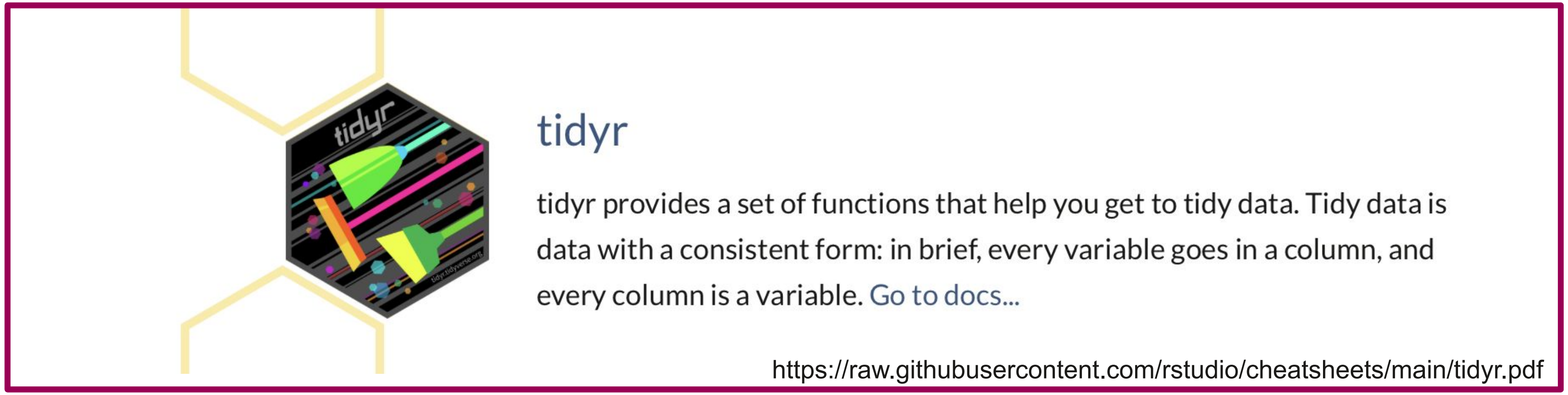


Preserve **cases** in vectorized operations

Tibbles are a table format provided by the tibble package. They inherit the data frame class, but have improved behaviors

```
> install.packages(“tibble”)
> library(tibble)
```

Data tidying



tidyr

tidyr provides a set of functions that help you get to tidy data. Tidy data is data with a consistent form: in brief, every variable goes in a column, and every column is a variable. Go to docs...

https://raw.githubusercontent.com/rstudio/cheatsheets/main/tidyr.pdf

```
> install.packages(“tidyr”)
> library(tidyr)
```

Data tidying with tidyr

unite / separate

	name	mounth	year
1	dupont_lepetit	09	1945
2	jean_legrand	11	2000
3	toto_tutu	04	1820

	name	birth
1	dupont_lepetit	09/1945
2	jean_legrand	11/2000
3	toto_tutu	04/1820

```
unite(tb, month, year, col="birth", sep="/", remove=TRUE)
```

Data tidying with tidyr
unite / separate

	name	month	year
1	dupont_lepetit	09	1945
2	jean_legrand	11	2000
3	toto_tutu	04	1820

→

	Fname	Lname	month	year
1	dupont	lepetit	09	1945
2	jean	legrand	11	2000
3	toto	tutu	04	1820

```
separate(tb, col = “name”, into = c(“Fname”, “Lname”), sep=”_”, remove=TRUE)
```

Data tidying with tidyr
unite / separate

	name	month	year
1	dupont_lepetit	09	1945
2	jean_legrand	11	2000
3	toto_tutu	04	1820

→

	Fname	Lname	birth
1	dupont	lepetit	09/1945
2	jean	legrand	11/2000
3	toto	tutu	04/1820

```
tb.u <- unite(tb, month, year, col="birth", sep="/")
separate(tb.u, col = "name", into = c("Fname", "Lname"), sep="_")
```

=

```
unite(tb, month, year, col="birth", sep="/") %>%
      separate(col = "name", into = c("Fname", "Lname"), sep="_")
```

Data tidying with tidyr
spread / gather

	name	month	year
1	dupont_lepetit	09	1945
2	jean_legrand	11	2000
3	toto_tutu	04	1820

```
gather(tb, key, value, -name)
```

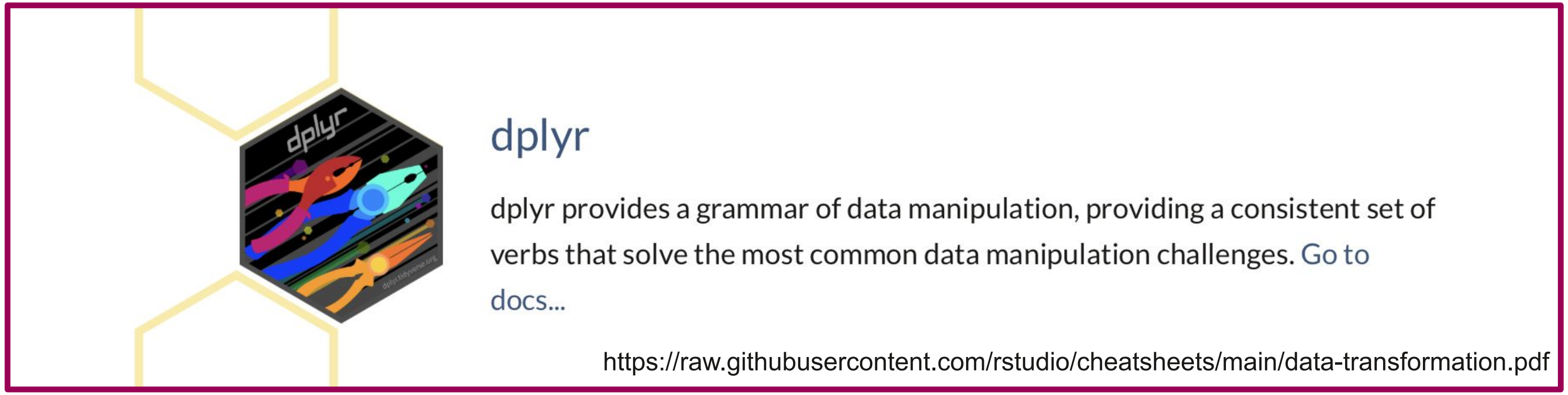
$\longrightarrow$

$\longleftarrow$

```
spread(tb, key, value)
```

	name	key	value
1	dupont_lepetit	month	09
2	dupont_lepetit	year	1945
3	jean_legrand	month	11
4	jean_legrand	year	2000
5	toto_tutu	month	04
6	toto_tutu	year	1820

Data transformation



dplyr

dplyr provides a grammar of data manipulation, providing a consistent set of verbs that solve the most common data manipulation challenges. Go to docs...

https://raw.githubusercontent.com/rstudio/cheatsheets/main/data-transformation.pdf

```
> install.packages("dplyr")
> library(dplyr)
```

Data transformation with dplyr

filter : *picks cases based on their values*

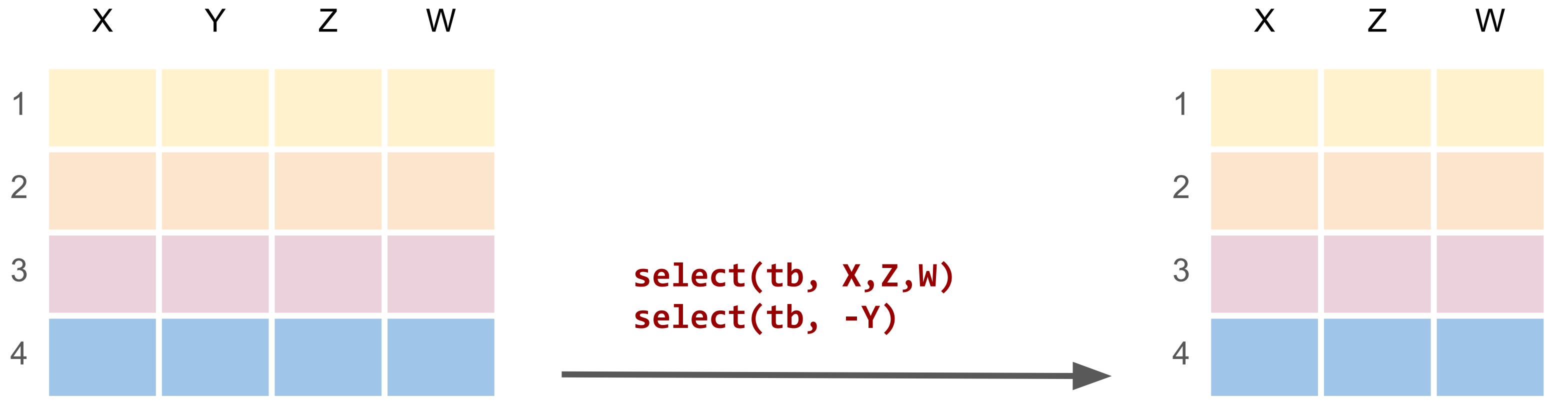
	X	Y	Z	W
1	0	no		
2	4	yes		
3	10	yes		
4	1	yes		

```
filter(tb, X > 2, Y == “yes”)
```

	X	Y	Z	W
1	4	yes		
2	10	yes		

Data transformation with dplyr

select : picks variables based on their names



```
select(tb, X,Z,W)
select(tb, -Y)
```

Data transformation with dplyr

mutate : adds new variables that are functions of existing variables

	X	Y	Z
1	a	b	
2	c	d	
3	e	f	
4	g	h	

```
mutate(tb, W = paste(X, Y, sep = "_"))
```

	X	Y	Z	W
1	a	b		a_b
2	c	d		c_d
3	e	f		e_f
4	g	h		g_h

```
> paste("hello", "world", sep = " ")
[1] "hello world"
```

Data transformation with dplyr

mutate / if_else

	X	Y	Z
1	NA	b	
2	c	d	
3	e	f	
4	g	h	

```
mutate(tb, W = if_else(is.na(X), Y,
                paste(X, Y sep = “_”)))
```

	X	Y	Z	W
1	NA	b		b
2	c	d		c_d
3	e	f		e_f
4	g	h		g_h

```
> x <- c(NA, 0, NA, 3)
> dplyr::if_else(is.na(x), 0, x)
[1] 0 0 0 3
```

Data transformation with dplyr

summarie / group_by

	X	Y	Z
1	1	a	
2	2	a	
3	3	a	
4	2	b	

```
summarise(tb, avgX = mean(X))
```

	avgX
1	2

```
tb.g <- group_by(tb, Y)
summarise(tb.g , avgX = mean(X), countY = n())
```

	Y	avgX	countY
1	a	2	3
2	b	2	1

Data transformation with dplyr

full_join / left_join / right_join

	X	Y	Z
1	1	a	
2	2	a	
3	3	a	
4	2	b	

+

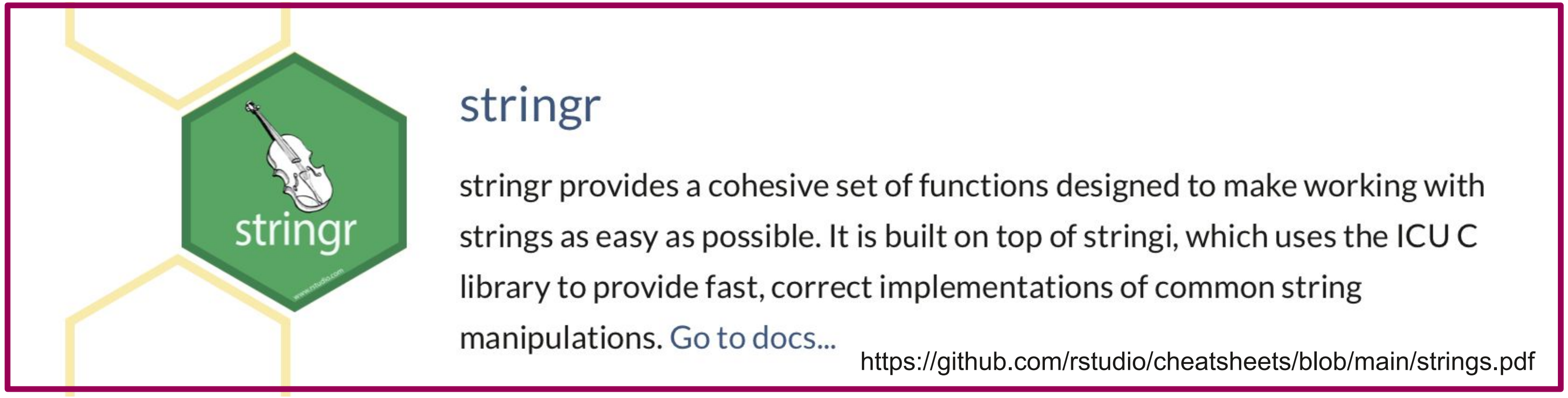
	Y	avgX	countY
1	a	2	3
2	b	2	1

→

	X	Y	Z	avgX	countY
1	1	a		2	3
2	2	a		2	3
3	3	a		2	3
4	2	b		2	1

```
full_join(tb1 , tb2, by = “Y”)
```

Data transformation



stringr

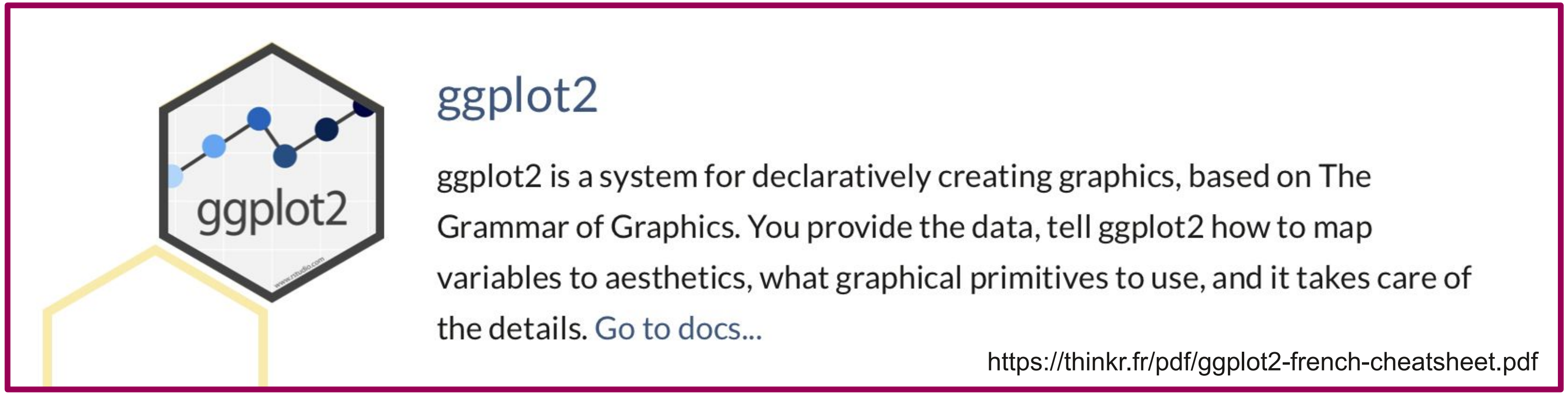
stringr provides a cohesive set of functions designed to make working with strings as easy as possible. It is built on top of stringi, which uses the ICU C library to provide fast, correct implementations of common string manipulations. Go to docs...

https://github.com/rstudio/cheatsheets/blob/main/strings.pdf

```
> install.packages("ggplot2")
> library(ggplot2)
```

https://stringr.tidyverse.org

Graphique



ggplot2

ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details. Go to docs...

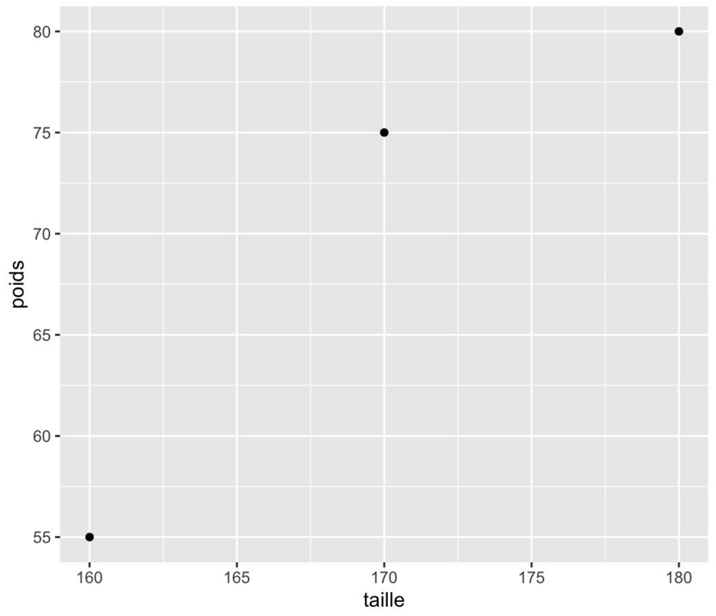
https://thinkr.fr/pdf/ggplot2-french-cheatsheet.pdf

```
> install.packages("ggplot2")
> library(ggplot2)
```

Les Graphiques avec ggplot2
geom_point

```
ggplot(data = tb) + geom_point(aes(x = taille, y = poids))
```

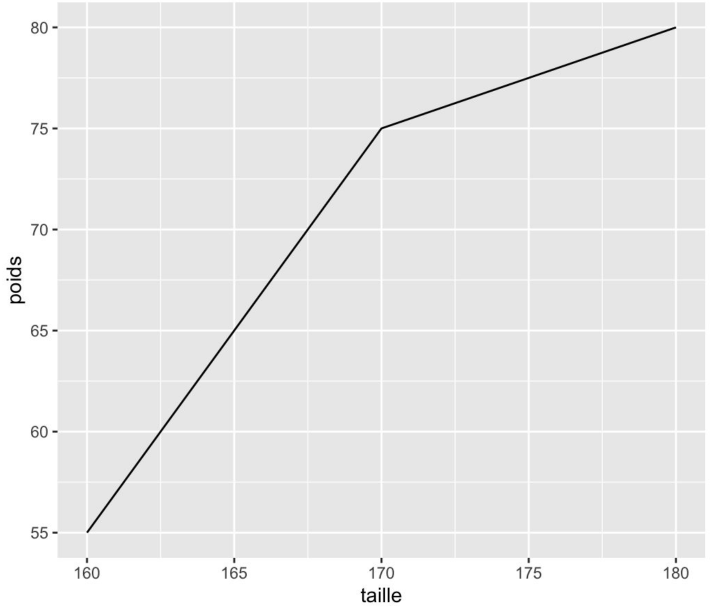
	name	taille	poids
1	dupont_lepetit	180	80
2	jean_legrand	170	75
3	toto_tutu	160	55



Les Graphiques avec ggplot2
geom_line

```
ggplot(data = tb) + geom_line(aes(x = taille, y = poids))
```

	name	taille	poids
1	dupont_lepetit	180	80
2	jean_legrand	170	75
3	toto_tutu	160	55

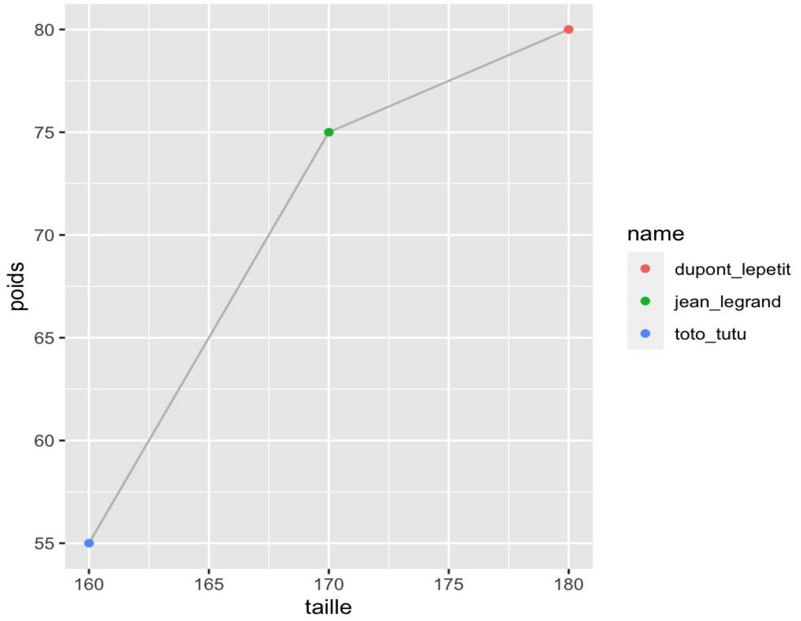


Les Graphiques avec ggplot2

geom_line

```
ggplot(data = tb) +
geom_line(aes(x = taille, y = poids) , color="grey") +
geom_point(aes(x = taille, y = poids, color=name))
```

	name	taille	poids
1	dupont_lepetit	180	80
2	jean_legrand	170	75
3	toto_tutu	160	55


poids
taille
name
dupont_lepetit
jean_legrand
toto_tutu