Atelier : R pour les projets d'étude/recherche

J. Montornès

21 mars 2023

Plan

1. Les premiers pas (interface, aide)
2. Les packages
3. La syntaxe
4. L'acquisition de données (séries temporelles, données individuelles)
5. Le traitement de données
6. L'estimation
7. Les sorties (graphiques, tableaux)
8. Les dates, les chaînes de caractère
9. Application 1 : l'étalonnage-calage des anticipations quanti/quali
10. Application 2 : l'équation d'euler de la consommation

1. L'interface

- Editeur : R script ~ do.file, pas de log.file en R (File → New File → R Script)
- Console : résultats ou erreurs
- Environement : objets ou fonctions crées
- Output : packages utilisés, graphiques crées

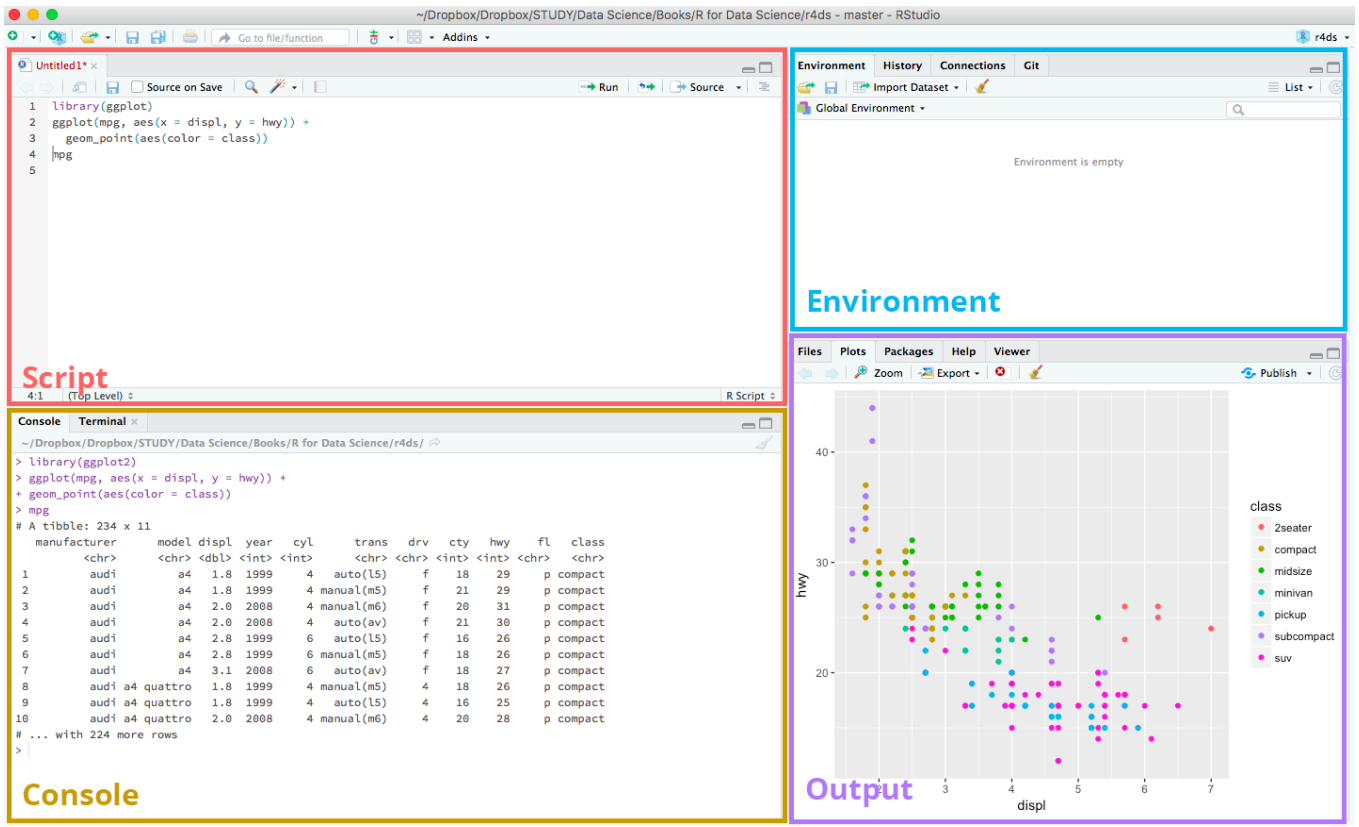


Figure 1: L'interface R

L'aide

- Fonction help()
- Cheatsheet : accessibles depuis le menu Help
- Pour aller plus loin : UtilitR, Stata2r et la Mixtape

2. Packages

- 15 000 packages disponibles aujourd'hui sur le CRAN (r-project.org)
- Installer la première fois puis commenter

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

- Relancer la commande `library()` à chaque session

```
library(tidyverse)
```

- Un package est composé de fonctions complémentaires. Package et fonction peuvent avoir le même nom (stargazer) ou pas (pkg ggplot2 et ggplot())

2. Packages

- Packages concurents dans la gestion des données : tidyverse vs. data.table
- Cette présentation privilégie le tidyverse mais le data.table est plus adapté pour les grosses bases (centaine de millions d'observation)
- Dans un data.table, les crochets dt[...] permettent de faire beaucoup plus de choses (quasiment tout, en pratique). En fait, les instructions à l'intérieur des crochets peuvent être envisagées comme des requêtes SQL mises en forme différemment

3. La syntaxe : ce qui diffère de Stata

- <- l'assignatation est préférée à '='

```
prix_baguette <- 0.95
print(prix_baguette)
```

```
## [1] 0.95
```

- l'opérateur '%in%' (identifie si un élement appartient à un vecteur)

```
3 %in% c(1,2,3)
```

```
## [1] TRUE
```

- la fonction c() crée des vecteurs
- l'opérateur 'pipe' %>% : on en reparle après

4. Chargement des données

Stata

Stata ne peut charger qu'une seule base de données à la fois

```
use example_data.dta
browse
```

R

R peut charger plusieurs bases à la fois, il faut donc assigner une base à un objet.

```
example_data <- c(2,3,5,7,11,13)
view(example_data)
example_data_2 <- c(0,1,1,2,3,8)
print(example_data_2)
```

```
## [1] 0 1 1 2 3 8
```

Importation et exportation de fichiers

- La fonction read_xlsx() dans le package readxl

```
chem<-"Cconsumer_inflation_quantitative_estimates.xlsx"
q61_dataset<- read_xlsx(chem,  sheet = "EA_Q61")
```

- Des fonctions de type existent pour tous les formats de données : read_csv(), read_csv2(), read_dta, read_sas dans le package haven
- Exportation : write.xlsx (openxlsx)

```
data(mtcars)
write.xlsx(mtcars,'//intra/partages/AU_AMIC/PROJET_GGSM/Olc
```

Les objets

- Le data frame

```
## New names:
## * `` -> `...1`
```

- La série temporelle

```
q61_ts <- ts(q61   , start = c(2004, 1), frequency = 4)
```

- Vecteur

```
services <- c("samic", "seps", "seec")
```

- Matrice

```
A<-matrix(1:9,nrow=3,ncol=3)
```

- Liste, bouleen, etc.

Import d'une série macroéconomique avec rdbnomics

https://db.nomics.world/

Ma méthode : 1) mots clés "national accounts insee" 2) selection de la série avec vos critères 3) récupérer le code de la série

```
# Options obligatoires a la BdF
options(rdbnomics.use_readLines = TRUE)

df_ct <- rdb(ids = c("INSEE/CNT-2014-CSI/T.CNT-OPERATIONS_S
```

```
## [1] 669 694 697
```

Pour aller plus loin : les posts de blog de Thomas Brand (https://www.r-bloggers.com/2020/10/access-the-free-economic-database-dbnomics-with-r-4/)

5. Le traitement des données avec dplyr

- **dplyr**

1. mutate() création de variables
2. select() selection de colones
3. filter() filtre des lignes
4. summarise() agrégation
5. arrange() tri

- Les fonction de **dplyr** ne modifie pas les données. Il faut assigner les données dans un nouveau dataframe

5. Le traitement des données avec dplyr

- l'opérateur "pipe" '%>%' permet d'enchaîner les instructions à la suite les unes des autres avec le package dplyr

```
data("mtcars")
mtcars %>%
  summarise(mean_mpg = mean(mpg))
```

```
##   mean_mpg
## 1 20.09062
```

Filtrer des lignes, sélectionner de colonnes

Stata

```
keep if mpg>15
keep mpg cyl gear
```

R

```
data("mtcars")
mtcars %>%
   filter(mpg>15) %>%
   select(mpg,cyl,gear)
```

```
##                    mpg cyl gear
## Mazda RX4         21.0   6    4
## Mazda RX4 Wag     21.0   6    4
## Datsun 710        22.8   4    4
## Hornet 4 Drive    21.4   6    3
## Hornet Sportabout 18.7   8    3
## Valiant           18.1   6    3
## Merc 240D         24.4   4    4
```

Créer une variable : mutate

Créer une variable

Stata

```
gen kml=mpg*0.4
```

R dplyr

```
mtcars %>%
  mutate(kml=mpg*0.4)  %>%
  select(kml,mpg) %>%
  arrange()
```

```
##                      kml  mpg
## Mazda RX4           8.40 21.0
## Mazda RX4 Wag       8.40 21.0
## Datsun 710          9.12 22.8
## Hornet 4 Drive      8.56 21.4
## Hornet Sportabout   7.48 18.7
## Valiant             7.24 18.1
```

Agréger les données (moyenne par groupe)

Stata

```
collapse (mean) mean_mpg = mpg, by(cyl)
```

R dplyr

```
data("mtcars")
mtcars %>%
  group_by(cyl) %>%
  summarise(mean_mpg = mean(mpg))
```

```
## # A tibble: 3 x 2
##     cyl mean_mpg
##   <dbl>    <dbl>
## 1     4     26.7
## 2     6     19.7
## 3     8     15.1
```

Fusionner deux tables

Stata

```
merge 1:1 id name using stat.dta
```

R : Methode 1

```
df1 <- data.frame(id=c(1, 2, 3, 4, 5),
                  revenue=c(34, 36, 40, 49, 43))
df2 <- data.frame(id=c(1, 2, 5, 6, 7),
                  expenses=c(22, 26, 31, 40, 20))
df <- merge( df1,  df2, by="id")
print(df)
```

```
##   id revenue expenses
## 1  1      34       22
## 2  2      36       26
## 3  5      43       31
```

Methode 2 : inner_join(),left_join(), right_join(), full_join()

6. L'estimation

Stata

```
reg y x1 x2
```

R base

```
eps <- rnorm(100)
x1 <- rnorm(100)
x2 <- rnorm(100)
y <-  2*x1 + x2 + eps
secteur<-round(runif(100, min=1, max=6))
df<-data.frame(y,x1,x2,secteur)
lm(y ~ x1 + x2, df)
```

Fixest

Estimation très rapide et conviviale de modèles à effets fixes pour un vaste ensemble de modèles linéaire ou non-linéaire : feglm, femlm, fenegbin, fepois

fixest > reghdfe (Stata) ; fixest > FixedEffestModels (Julia)

```
feols(y ~ x1 + x2, data=df)
```

R fixest

```
library(fixest)
data(trade)
# OLS estimation
gravity = feols (log(Euros) ~ log(dist_km) |
  Destination + Origin + Product + Year, trade)
summary(gravity)
```

```
## OLS estimation, Dep. Var.: log(Euros)
## Observations: 38,325
## Fixed-effects: Destination: 15,  Origin: 15,  Product: 2
## Standard-errors: Clustered (Destination)
##                Estimate Std. Error  t value   Pr(>|t|)
## log(dist_km) -2.16988   0.138436 -15.6742 2.8408e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
## RMSE: 1.74337     Adj. R2: 0.705139
##                 Within R2: 0.219322
```

Dummies dans un modèle

Stata

Dans Stata, on ajoute `i.` :

```
reg y i.secteur
```

R

Dans R, on tranforme la variable en factor()

```
feols(y ~ x1 + x2 + factor(secteur), data=df)
```

Si la variable est assignée dans le dataframe comme facteur il n'est pas besoin de la transformer

Ecart-types robustes

Stata

```
* ", robust" utilise hc1 par defaut
regress y x1 x2, robust
regress y x1 x2, vce(hc3)
```

R

```
# sandwich's vcovHC utilise HC3 par defaut
feols(y ~ x1 + x2, df, vcov = sandwich::vcovHC)
```

7. Les sorties : Tableau de regression

Stata

```
reg y x1 x2
eststo est1
esttab est1b

esttab est1 est1b
```

R

```
mod1<-lm(y ~ x1 + x2, df)
mod2<-lm(y ~ x1 + x2 + factor(secteur), df)
stargazer(mod1, mod2,type="text")
```

7. Les sorties : Tableau de regression

	Dependent variable:	
	y	
	(1)	(2)
x1	1.991***	1.981***
	(0.116)	(0.119)
x2	1.009***	1.028***
	(0.115)	(0.119)
factor(secteur)2		-0.363
		(0.443)
factor(secteur)3		-0.067
		(0.427)
factor(secteur)4		0.026
		(0.458)
factor(secteur)5		-0.032
		(0.436)
factor(secteur)6		-0.599
		(0.525)
Constant	0.090	0.233
	(0.112)	(0.360)
Observations	100	100
R2	0.790	0.796
Adjusted R2	0.785	0.781
Residual Std. Error	1.120 (df = 97)	1.131 (df = 92)
F Statistic	182.258*** (df = 2; 97)	51.414*** (df = 7; 92)
Note:		*p<0.1; **p<0.05; ***p<0.01

Figure 2: Modèle (1) vs (2)

Histograme simple

Stata

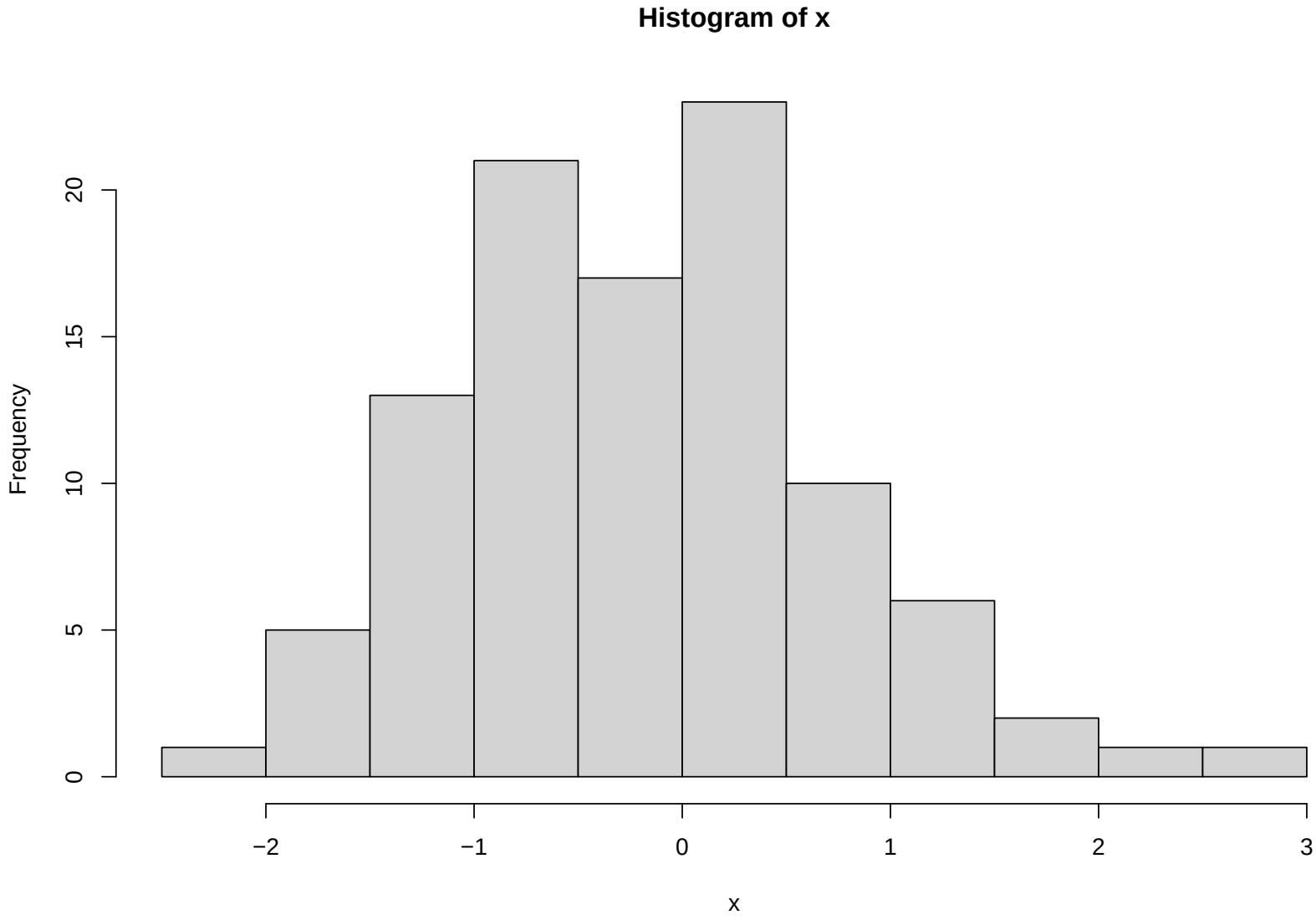
```
set obs 100
gen x = rnormal()
histogram x
```

R

```
x <- rnorm(100)
hist(x)
```

Histograme simple

R



Nuage de points

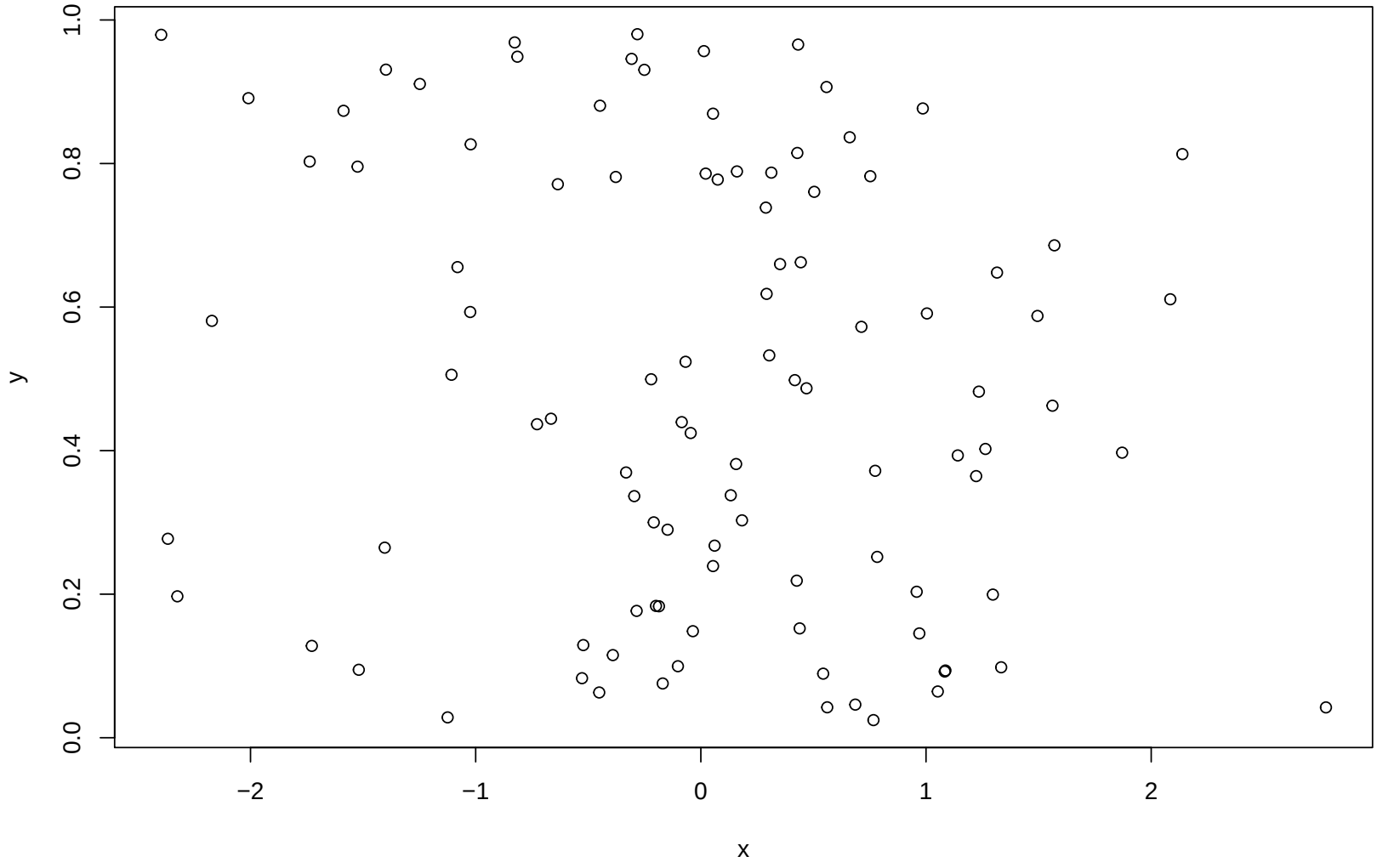
Stata

```
twoway scatter x y
```

R

```
plot(x, y)
```

Nuage de points : Méthode 1

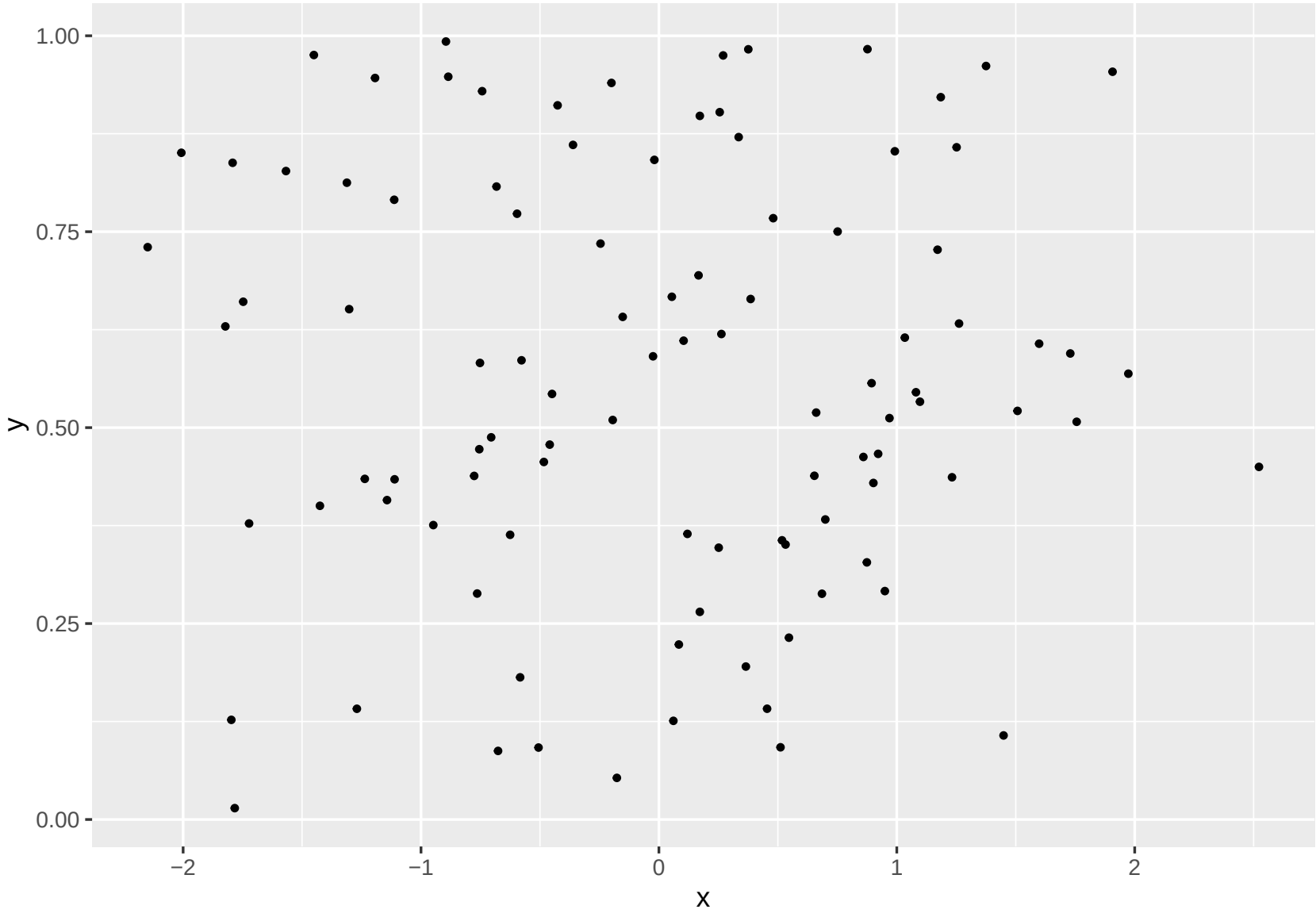


Nuage de points : Méthode ggplot2

```
x <- rnorm(100)
y <- runif(100)
df<-data.frame(x,y)

ggplot(df) + geom_point(aes(x, y))
```

Nuage de points : Méthode ggplot2



Les graphiques avancés avec ggplot2

- On dispose d'un data frame de 3 variables (**df1**) : period, value, var
- Choix d'un type de graphique : ici c'est **geom_line**
- **aes** : Lier les données aux élements visuels "period" $\rightarrow$ x, "value" $\rightarrow$ y, "var" $\rightarrow$ shape, color, etc.
- Ensuite, les options du graphiques ("theme", "scale", etc.) s'ajoutent les unes aux autres avec "+"

```
ggplot(df1) +
  geom_line(size = 1.2, aes(x = period, y = value, color =
  theme_minimal()+ xlab("") + ylab("") +
  scale_x_date(breaks='2 year',expand=c(0.01,0.01),date_lab
  scale_y_continuous(breaks=seq(-6, 8, by = 2)) +
  theme(legend.position ="bottom") +
  theme(legend.title = element_blank())
```

R ggplot

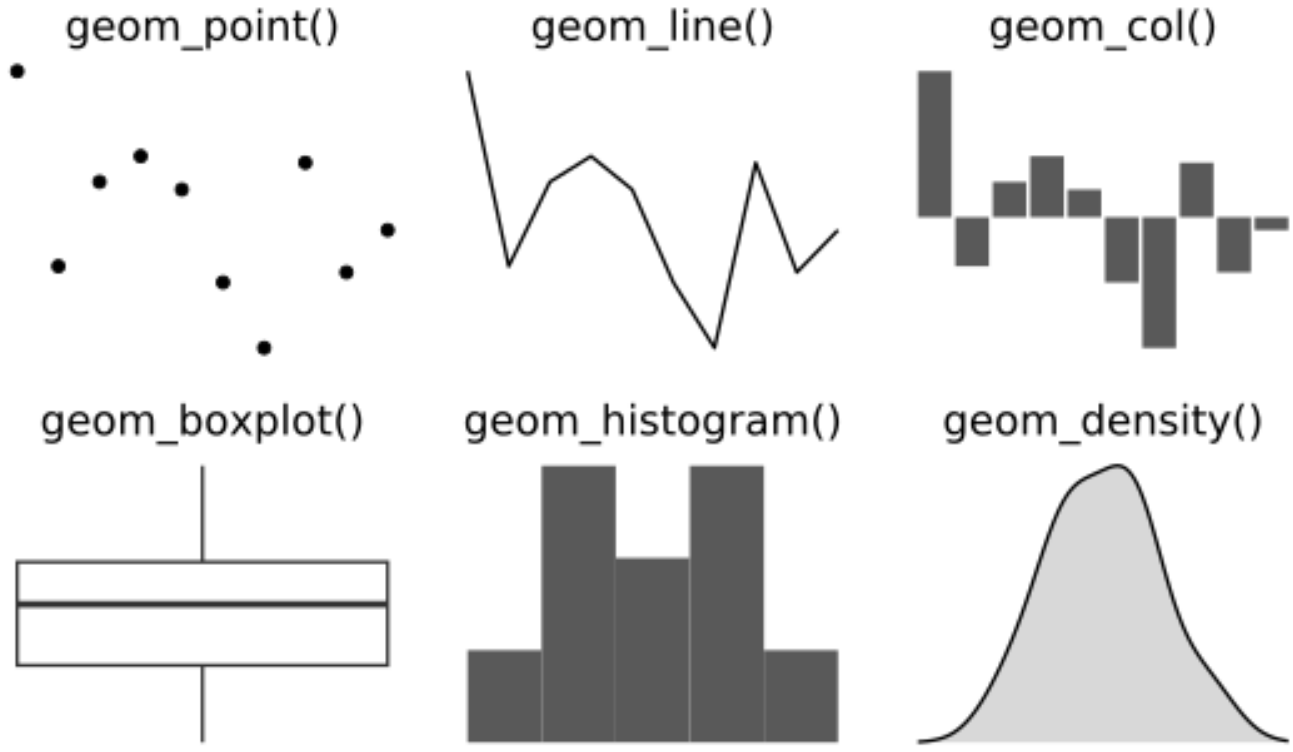


Figure 3: Commandes des principaux types de graphique ggplot

R ggplot

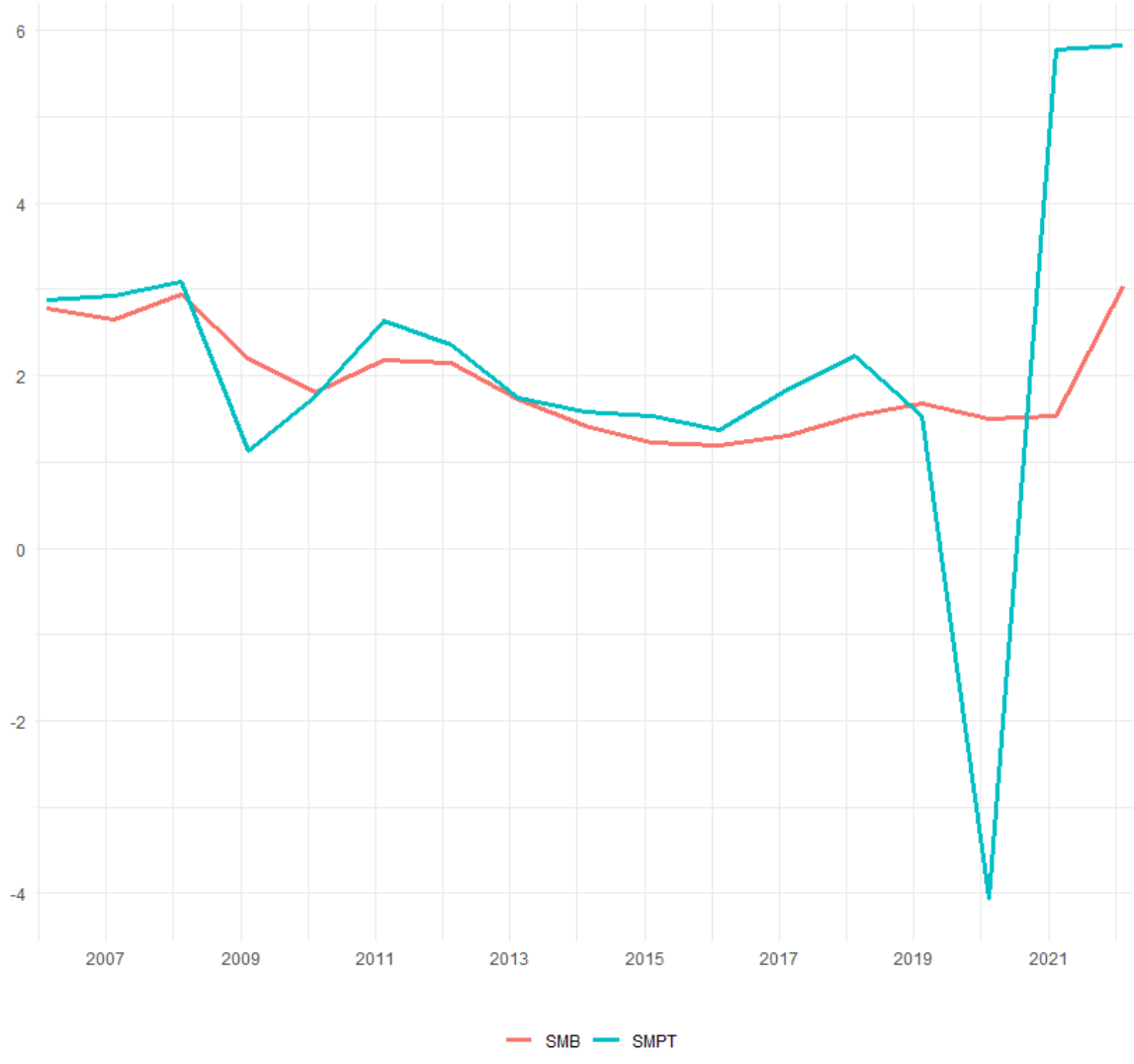


Figure 4: Evolution des salaires en France, moyenne annuelle

8. Manipuler les dates avec Lubridate

Stata

- J'ai beaucoup de problèmes avec les dates Stata

R

- En R, il existe un vaste ensemble de fonctions qui rendent la vie plus simple
- Un exemple avec une fonction qui convertit une "chaîne de caractères" FR en date

```
jourJ <- dmy("11 février 2023")
class(jourJ)
```

```
## [1] "Date"
```

```
print(jourJ)
```

```
## [1] "2023-02-11"
```

8. Manipuler les dates avec Lubridate

Conversion (suite) : ymd(), ymd_hms, dmy_hms, mdy(), ...

```
jourJ <-ymd(20230211)
print(jourJ)
```

```
## [1] "2023-02-11"
```

Extraire un composant d'une date : year(), month(), mday(), hour(), minute() and second()

```
annee <-year(jourJ)
print(annee)
```

```
## [1] 2023
```

8. Les chaînes de caractères avec Stringr

Stata

```
subinstr("Hello world", "world", "universe", .)
substr("Hello world", 1, 4)
regexm("Hello world", "Hello")
```

R

```
str_replace_all("Hello world", "world", "universe")
```

```
## [1] "Hello universe"
```

```
str_sub("Hello world", 1, 4)
```

```
## [1] "Hell"
```

```
str_detect("Hello world", "Hello")
```

```
## [1] TRUE
```

8. Les chaînes de caractères : conversion

Stata

```
detring id, replace
```

R

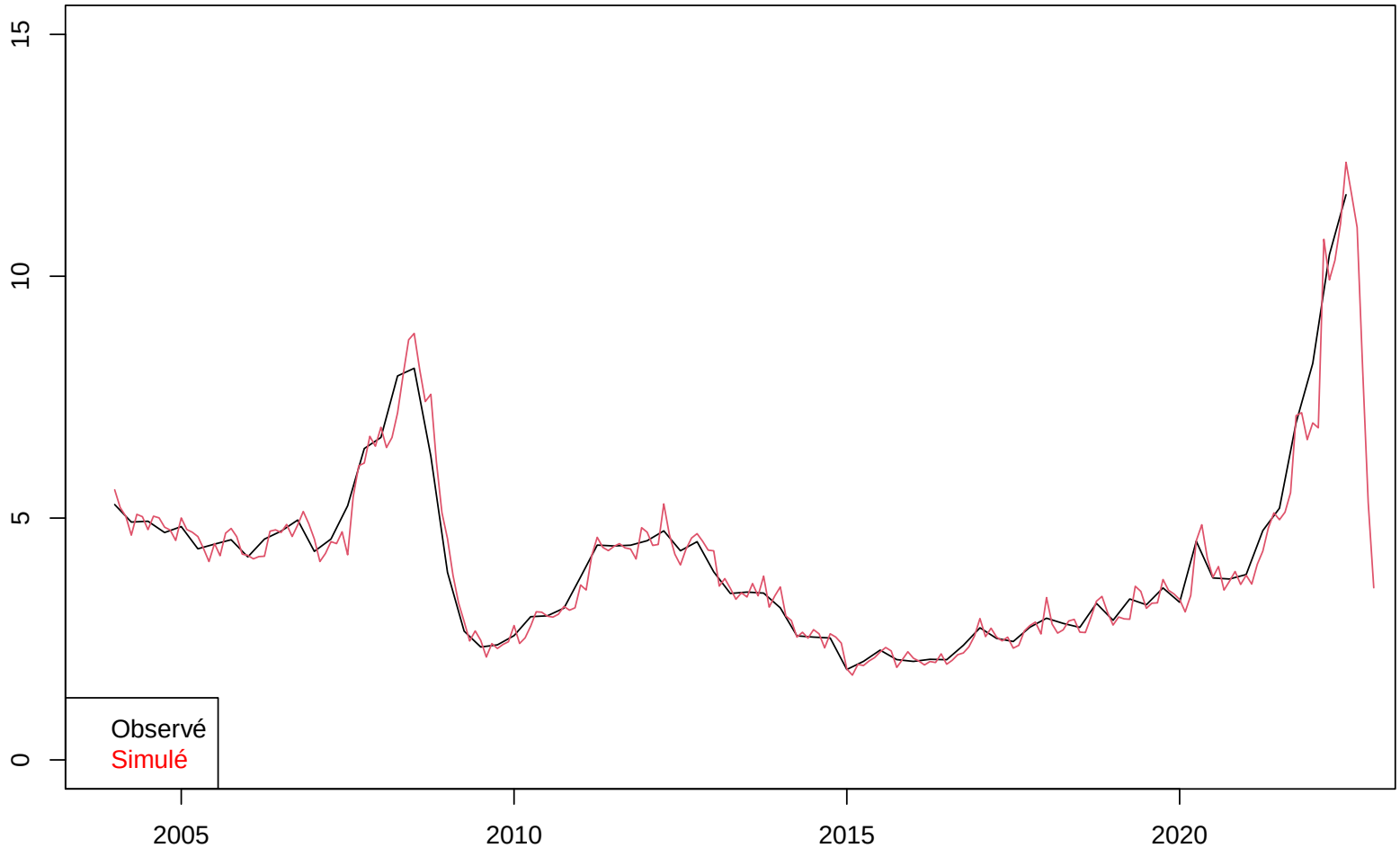
```
id <- c("0999")
id_num <-as.numeric(id)
print(id_num)
```

```
## [1] 999
```

Application 1 : Etallonage-calage

```
## New names:
## * `` -> `...2`
## * `` -> `...10`
## * `` -> `...18`
## * `` -> `...26`
## * `` -> `...34`
## * `` -> `...42`
## * `` -> `...50`
## * `` -> `...58`
## * `` -> `...66`
## * `` -> `...74`
## * `` -> `...82`
## * `` -> `...90`
## * `` -> `...98`
## * `` -> `...106`
## * `` -> `...114`
## * `` -> `...122`
## * `` -> `...130`
```

Application 1 : Etallonage-calage



Application 2 : l'équation d'euler

```
# paramètres
r<-1
rho<-0.9
theta<-0.5
g<-0.9
coninital <- c(con = 1)
times <- seq(from = 0, to = 100, by = 0.2)

# définition d'une fonction
cdot <- function(t, con, parms){
  list((((r-rho)/theta))*con)}

# résoltion numérique de l'équation différentielle
out <- ode(y = coninital, times = times, func = cdot,
           parms = NULL)
```

Application : l'équation d'euler