Introduction to R

Yao Thibaut Kpegli

Master 1 in Economics, ENS Paris-Saclay
2023–2024

November 2023

Introduction

Purpose

- A quick and intensive reminder about R (4.5 hours)
  - to make you operational during your econometrics classes over the year

R: definition

R is a free software to conduct statistical and econometrics analysis, optimization, simulation, numerical computation, etc...

Installation

Installation

Download two distinct elements :

- download R from "comprehensive R archive network" (CRAN) `https://cran.r-project.org/`
  - Execute the installation program
  - Keep default parameters
- Download RStudio, for example from `https://posit.co/download/rstudio-desktop/`
  - Execute the installation program
  - Keep default parameters

Assign operator

Assign operator

create a scalar a that takes value **2**

- a = 2
- a <- 2
- 2 -> a

Common operations

Common operations

- addition : 2+2
- subtraction 2-2
- product : 2*2
- division: 2/2

Common math functions

Common math functions

- square/exponent : 2^2
- square root: sqrt(2) or 2**0.5
- exponential : exp(4)
- logarithm : log(4)
- min: min(2,3)
- max: max(2,3)
- absolute val.: abs(-4)
- sign: sign(-10)
- round: round(4.56789, 2)
- ceiling: ceiling(4.56789)
- integer: as.integer(4.56789)

Vectors

Specific vectors

- repetition: rep(1, 10)
- sequence: seq(from = 0, to = 20, by =1)
  - even sequence: seq(from = 0, to = 20, by =2)
  - odd sequence: seq(from = 1, to = 21, by =2)
- trend: 1980:2023

Vectors

- c(1,3,2,0,4,5)

$$\begin{pmatrix} 1 & 3 & 4 & 0 & 4 & 5 \end{pmatrix}$$

  - second element of the vector: c(1,3,2,0,4,5)[2]
  - stabdard-deviation : sd(c(1,3,2,0,4,5))
  - sum : sum(c(1,3,2,0,4,5))
  - median: median(c(1,3,2,0,4,5))
  - sort: sort(c(1,3,2,0,4,5))
  - rank: rank(c(1,3,2,0,4,5))
- vector of characters: c("abcd","a", "ab", "abc")

Matrix operations

Matrix operations

- matrix(1:6, nrow = 2) or matrix(1:6, ncol = 3)

$$\begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$

  - by default, R arranges elements by column “byrow = F"... Try matrix(1:6, nrow = 2,byrow = T)
- nb of row : nrow(matrix(1:6, nrow = 2))
- nb of col: ncol(matrix(1:6, nrow = 2))
- element at the line i=1 and column j=3: matrix(1:6, nrow = 2)[1,3]
- extract first column: matrix(1:6, nrow = 2)[,1]
- extract the third line: matrix(1:6, nrow = 2)[2,]
- extract the first and third columns: matrix(1:6, nrow = 2)[1:2, c(1,3)] or matrix(1:6, nrow = 2)[1:2, c(1,3)]

Matrix operations

Matrix operations

- transpose: t(matrix(1:6, nrow = 2))
- dimension: dim(matrix(1:6, nrow = 2))
- matrix product: matrix(1:6, nrow = 2)%*%matrix(1:6, nrow = 3)
- determinant: det(matrix(1:6, nrow = 2)%*%matrix(1:6, nrow = 3))
- eigenvalues : eigen(matrix(1:6, nrow = 2)%*%matrix(1:6, nrow = 3))
- inverse: solve(matrix(1:6, nrow = 2)%*%matrix(1:6, nrow = 3))

cross and Kronecker products

cross product

$$A = \begin{pmatrix} 0.5 \\ 2 \end{pmatrix} \qquad \text{and} \qquad B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

- cross product: crossprod(A,B) = t(A)%*%B
    - result: (2.5 , 2)
- Kronecker product: kronecker(A, B)
    - result:

$$\begin{pmatrix} 0.5 & 0 \\ 0.5 & 0.5 \\ 2 & 0 \\ 2 & 2 \end{pmatrix}$$

Matrix operations

Matrix operations

- c(1,3,2,0,4,5)

$$\begin{pmatrix} 1 & 3 & 4 & 0 & 4 & 5 \end{pmatrix}$$

- matrix(c(1,3,2,0,4,5), nrow = 2)
- matrix(c(1,3,2,0,4,5), nrow = 2, byrow=F)
- matrix(c(1,3,2,0,4,5), nrow = 2,byrow=T)

identy matrix

diag(3)

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Matrix operations

rbind

- rbind with (1,3,2) and (0,4,5)
  - c(1,3,2) and c(0,4,5)
  - rbind(c(1,3,2),c(0,4,5))

$$\begin{pmatrix} 1 & 3 & 2 \\ 0 & 4 & 5 \end{pmatrix}$$

cbind

- cbind with (1,3,2) and (0,4,5)
  - c(1,3,2) and c(0,4,5)
  - cbind(c(1,3,2),c(0,4,5))

$$\begin{pmatrix} 1 & 0 \\ 3 & 4 \\ 2 & 5 \end{pmatrix}$$

Exercise

- Create a matrix X that takes the values

$$\begin{pmatrix} 1 & 1 \\ 1 & 4 \\ 1 & 2 \\ 1 & 5 \end{pmatrix}$$

- Compute $B = (X'X)^{-1}$
  - determine the dimension
  - extract the elements on the diagonal
- Create a matrix Y that takes value

$$\begin{pmatrix} 3 \\ 6 \\ 4 \\ 7 \end{pmatrix}$$

- Compute $(X'X)^{-1}X'Y$

Result

- X <- cbind(1,c(1,4,2,5))
- B <- solve(crossprod(X,X))
- dim(B)
- diag(B)
- Y <- matrix(c(3,6,4,7),nrow=4)
- OLS <- B%*%crossprod(X,Y)
- OLS

Working Directory

Working Directory

- Useful to import/export data and results
  1. set the work directory: setwd("D:/ENS Paris Saclay/R2023_2024")
  2. check what is work directory: getwd()

Definition

Definition

- dataframe = matrix of data
  - set of vectors with the same length
  - placed next to each other vertically
- Column = Variable
  - possible of different types: quantitative, numerical but also qualitative, characters, dates.
- Line = Observation

creation

Creation

- example <- data.frame(one = 1:10, two = 11 :20, three = 21:30)
- or equivalently
  - example <- as.data.frame(matrix(1:30, ncol=3))
  - names(example) <- c("one", "two", "three")
- display data : View(example)

basic data manipulation

Useful symbols

- Strict inequality : >, <
- equal or inequal : =<, >=
- equal: ==
- different equal: !=
- and : &
- or : |

basic data manipulation

- mean: mean(example$one)
- stat des: summary(example$one)
- rename var.: names(example)[names(example) == "ratio"] <- "new"
- replace obs: example$one[example$one==3] <- 5

attach, detach, and with

attach, detach, and with

- "attach" allows to avoid referring to the dataframe at each line of code
  - e.g: attach(example), then directly write
    - mean(one)
    - min(two)
    - sd(two)
- "detach" is used to clear the "attach" command
- "attach" for a specific command: with(example, mean(two))

new variable

new variable

- new var.: example$ratio <- (example$one)/(example$two)
- dummy (binary) var.: example$dumy_ch <- ifelse(example$one <= 5, "Low", "High")
- dummy (binary) var.: example$dumy_num <- ifelse(example$one <= 5, 1, 0)

Subset of dataframe

Subset of dataframe

- keep lines with two<=16: set_obs<- subset(example, two<=16)
- delete column two: set_var<- subset(example, select = -two)
- keep columns one and three : set_varA<- subset(example, select = c("one","three"))
- combination of conditions: set_obs_var<- subset(example, two<=16, select = -two)

Exportation

exportation in text file

write.table(example, file="example.txt", col.names=TRUE)

exportation in csv file

write.csv(example, file="example.csv")

exportation in excel file

1. install package: install.packages("writexl")
2. call library: library("writexl")
3. export: write_xlsx(example, "example.xlsx")

Importation

Importation of text file

text_file <- read.table("example.txt", header=TRUE)

exportation of csv file

read.csv(example, file="example.csv")

Importation of Excel file

1. install package: install.packages("xlsx")
2. library(xlsx)
3. Excel_file <- read.xlsx("example.xlsx")

Exercise

- Use data "Journals" within the package AER of R: data("Journals", package = "AER")
- Create a new variable citeprice = price/citations
- Attach the dataframe Journal
- Use the codes from this class to compute:
  - OLS estimator of the regression $log(subs) = a \times log(price/citations) + b + \epsilon$:

$$(X'X)^{-1}X'Y$$

  - matrix of variance-covariance:

$$\hat{\sigma}_{\epsilon}^{2}(X'X)^{-1}$$

  - t-stat
  - compare the results with those obtained from "lm(.)" command

Exercise: result

- install.packages("AER")
- data("Journals", package = "AER")
- Journals$citeprice <- Journals$price/Journals$citations
- names(Journals)
- View(Journals)
- attach(Journals)
- X<-cbind(log(citeprice),1)
- Y<-log(subs)
- beta<-solve(crossprod(X,X))%*%t(X)%*%Y ; beta
- e<-Y-X%*%beta ; e
- N<-nrow(Journals) ; N
- vcm<- (kronecker(crossprod(e,e)/(N-2),solve(crossprod(X,X)))) ; vcm
- sd<-sqrt(diag(vcm))
- sd
- tstat<-beta/sd
- tstat
- lms= lm(logsubs logpc, data = Journals)
- summary(lms)

Vertical merge of data

Vertical merge of data

- v1=data.frame(Num =c(1,-1,0,3,0,2), Str=c("L1","L2","L3","M1","M2","D1"))
- v2=data.frame(Num=c(10,20,30,40), Str=c("L1","L2","M1","D3"))
- v <- rbind(d1,d2)

Horizontal merge

Horizontal merge

- h1=data.frame(id =c(1,2,3,4,5,6), Num =c(1,-1,0,3,0,2), Str=c("L1","L2","L3","M1","M2","D1"))
- h2=data.frame(id =c(1,2,3,4,5,7), Name=c("Rac","Elo","Fra","Hon","Hor","Ben"))
- inner join: merge_inner <- merge(x=h1,y=h2,by="id")
- left join: merge_left <- merge(x=h1,y=h2,by="id", all.x=T)
- right join: merge_right <- merge(x=h1,y=h2,by="id", all.y=T)
- outer join: merge_outer <- merge(x=h1,y=h2,by="id", all=T)

Scatter plot

ggplot2

- install package: install.packages("ggplot2")
- scatter plot:
  ggplot(Journals, aes(log(citeprice),log(subs))) +
  geom_point()
- add colors (as a third dimension): ggplot(Journals, aes(log(citeprice),log(subs))) + geom_point(aes(color = pages))
- add linear fit:
  ggplot(Journals, aes(log(citeprice),log(subs))) +
  geom_point(aes(color = pages)) + geom_smooth(method = "lm", se=F)
- add linear fit+confidence interval:
  ggplot(Journals, aes(log(citeprice),log(subs))) +
  geom_point(aes(color = pages)) + geom_smooth(method = "lm", se=T)

common plots

bar plot

```
ggplot(Journals) + geom_bar(aes(society))
```

histogram

```
ggplot(Journals) + geom_histogram(aes(log(subs)))
```

empirical cumulative density function

```
ggplot(Journals) + stat_ecdf(aes(log(subs)))
```

empirical density

```
ggplot(Journals) + geom_density(aes(log(subs)))
```

box plot

```
ggplot(Journals) + geom_boxplot(aes(subs))
```

Function

- create function $f(\alpha) = \alpha^2$

  ```
  f <- function(alpha){
  y <- alpha**2
  return(y)
  }
  ```

- computation:
  - at a specific value: f(0)
  - on a range: f(-10:10)
  - on a range: f(seq(from=-2,to=2, by=0.1))

Function

- create function $f(x) = x^2 + y^2$
  f <- function(x,y){
  z <- x**2+y**2
  return(z)
  }
- computation:
  - at a specific value: f(1,1)
  - on a range: f(0:2,-1:1)
  - on a range: f(seq(from=0,to=2, by=0.1),seq(from=-,to=1, by=0.1))

Function

- create function $f(x) = x^2 + y^2$
  ```
  f <- function(x){
  z <- x[1]**2 + x[2]**2
  return(z)
  }
  ```
- computation:
  - at a specific value: f(c(1,1))

grid

grid

1. Two vectors: v1 and v2
2. To get all possible combinations of elements from v1 and v2
    - .... use “expand.grid(v1,v2)”
    - valid also for more than two vectors: expand.grid(v1,v2,...,vn)

grid

- vector: v1<- c(-1:1)
- vector 2: v2 <- c(0:2)
- grid: expand.grid(v1,v2)

Exercise

- create a vector x taking values from -1 to 1, by a step of 0.01
- create a vector y taking values from -1 to 1, by a step of 0.01
- create a grid of values of x and y
- compute $f(x, y)$ over the grid
- plot the function over the grid
  - define the color option of ggplot on the log(f)
  - use the 3D plot

Result

- vector x : x<- seq(from=-1, to=1,by=0.01)
- vector y : y<- seq(from=-1, to=1,by=0.01)
- grid : data <- data.frame(expand.grid(x,y))
- names(data) <- c("x","y")
- data$z <- (data$x)**2 + (data$y)**2
- plot:
  - ggplot ggplot(data, aes(x,y)) + geom_point(aes(color = log(z)))+ scale_color_gradientn(colours = rainbow(4))
  - 3D plot
    - library("plotly")
    - plot_ly(x=data$x, y=data$y, z=data$z, type="scatter3d", color=data$z, mode="markers")

Optimization

$$\min_x f(x) = x^2$$

- create function $f(x) = x^2$
  ```
  f <- function(x){
  y= x**2
  return(y)
  }
  ```
- initial values:
  ```
  initial_value <- 5
  ```
- use "optim(...)" function that minimizes functions:
  ```
  optim(initial_value, f, method="BFGS")
  ```

Optimization

$$\max_x f(x) = x - 0.5x^2$$

- create function $f(x) = x - 0.5x^2$
  f <- function(x){
  y= x - 0.5*x**2
  return(-y)
  }
- initial values:
  initial_value <- 5
- use "optim(...)" function that minimizes functions:
  optim(initial_value, f, method="BFGS")

Optimization

$$\min_{x,y} f(x,y) = x^2 + y^2$$

- create function $f(x) = x^2 + y^2$
  f <- function(x){
  y = x[1]**2+x[2]**2
  return(y)
  }
- initial values:
  initial_value <- c(10, 10)
- use "optim(...)" function:
  optim(initial_value, f, method="BFGS")

Otimization over dataframe (e.g. OLS)

```
Y <- log(Journals$subs)
X <- log(Journals$price/Journals$citations)
ols <- function(z){
sse <- sum((Y - z[1]*X-z[2])^2)
return(sse)
}
initial_value <- c(1, 1)
optim(initial_value, ols, method="BFGS")
```

for

for

```
n <- nrow(Journals)
count <- 0
for (i in 1:n) {
if(Journals$society[i] == "yes"){count = count+1}
}
print(count)
```

while

while

```
i <- 1
while (Journals$society[i] == "no") {
i = i+1
}
print(i)
```

Random values and re-allocation

useful random values

- normal: rnorm(n, mu, sigma)
  - vector of 10 random values of $N(0, 1)$: rnorm(10,0,1)
- uniform: runif(n, a, b)
  - vector of 10 random values of $U[0, 1]$: runif(10,0,1)

random re-allocation

- sample(vector)
  - v <- 1:10
  - sample(v)

Exercise

1. First replication
   - simulates a vector ϵ of $n = 1000$ random values of $N(0, 2)$
   - simulates a vector x of $n = 1000$ random values of $U(-5, 5)$
   - generate a vector $y = 2 \times x + 1 + \epsilon$
   - compute OLS estimator of the regression of y on x
   - save the OLS estimates
2. 1000 replications
   - Repeat the first question 1000 times
3. Compute the mean of the OLS estimates over the 1000 replications
4. Compare with the true values of parameters: 2 and 1

Result

- nb of replication: rep <- 10000
- sample size: n <- 1000
- true coef: coef <-c(2,1)
- stock vector: stock <- matrix(rep(NA,2*rep), ncol=2)
- loop “for” to make replications:
  for (i in 1:rep){
  sim <- data.frame(x=runif(n,-5,5),er=rnorm(n,0,2))
  sim$y = coef[1]*sim$x + coef[2] + sim$er
  lms= lm(y x, data = sim)
  stock[i,] <- lms$coefficients
  }
- compute the mean: Mean <- c(mean(stock[,1]),mean(stock[,2]))
- show the result: Mean

Exercise

- provides an approximated value for $F(x)$

$$F(x) = \int_{-10}^{10} f(x)dx$$

with $f(.)$ the density function of the standard normal distribution, and $\pi \simeq 3.14$

  - hint: $F(x) = (10 - (-10)) \times E[f(x)]$, with $E[.]$ the expectation computed with $U[-10, 10]$
  - this value is almost 1

1. simulates a vector x of $n = 100000000$ random values of $U(-10, 10)$
2. compute f(.) over the simulated values of x
3. Compute the mean of the $\overline{f}$ over the n simulated values
   - rmk (central limit theorem): $\overline{f}$ converges in proba towards $E[f(x)]$
4. compute the approximated values as $20 \times \overline{f}$

Result

- uniform : x <- runif(100000000,-10,10)
- compute f: f <- (1/(sqrt(2*3.14)))*exp(-0.5*x**2)
- compute mean of f: f_bar <-mean(f)
- compute approximated value: approx <- f_bar*20
- show the approximated value: approx

Exercise

Provide an approximation for the quantity π

- hint: leverage on (i) the area of a circle $x^2 + y^2 \leq 1$, (ii) the area of a square centered at (0,0) and whose side is 2, and (ii) random draws of x and y from $U[-1, 1]$.

Result

- simulate random values on the square whose side is 2: pid <- data.frame(x=runif(100000000,-1,1),y=runif(100000000,-1,1))
- identify random values that belong to the area of a circle $x^2 + y^2 \leq 1$
  - pid$area <- (pid$x)**2+(pid$y)**2
  - pid$dum <- ifelse(pid$area <= 1, 1, 0)
- compute the approximation: pi_approx <- 4*mean(pid$dum)
  - note that "4" corresponds to the area of the square whose side is 2
- pi_approx

Next: STATA !