TP d'introduction à R

Evan Voyles

02 Avril, 2022

Exercice 1

```
r <- c(4, 5, 6)
A <- matrix(rep(r, 3), nrow = 3, ncol = 3, byrow = TRUE)
print(A)
```

```
##      [,1] [,2] [,3]
## [1,]    4    5    6
## [2,]    4    5    6
## [3,]    4    5    6
```

```
B <- rbind(A, r)
dimnames(B) <- NULL # enlever les noms des dimensions
print(B)
```

```
##      [,1] [,2] [,3]
## [1,]    4    5    6
## [2,]    4    5    6
## [3,]    4    5    6
## [4,]    4    5    6
```

```
dim(B)
```

```
## [1] 4 3
```

B est une matrice avec 4 lignes et 3 colonnes.

Exercice 2

```
set.seed(10)
x <- runif(5)
x
```

```
## [1] 0.50747820 0.30676851 0.42690767 0.69310208 0.08513597
```

```
x[2] # Deuxième élément de x
```

```
## [1] 0.3067685
```

```
x[c(2, 4)] # deuxième et quatrième élément de x
```

```
## [1] 0.3067685 0.6931021
```

```
x[-1] # Tous les éléments de x sauf le premier
```

```
## [1] 0.30676851 0.42690767 0.69310208 0.08513597
```

```
M <- matrix(runif(16), 4, 4)
M
```

```
##           [,1]      [,2]       [,3]      [,4]
## [1,] 0.2254366 0.4296715 0.59592531 0.2641777
## [2,] 0.2745305 0.6516557 0.35804998 0.3987907
## [3,] 0.2723051 0.5677378 0.42880942 0.8361341
## [4,] 0.6158293 0.1135090 0.05190332 0.8647212
```

```
M[1, 3] # Elément dans la position 1, 3
```

```
## [1] 0.5959253
```

```
M[,1]   # première colonne de M
```

```
## [1] 0.2254366 0.2745305 0.2723051 0.6158293
```

```
M[2,]   # deuxième ligne de M
```

```
## [1] 0.2745305 0.6516557 0.3580500 0.3987907
```

```
L = list(x, M, "Hello")

L[[2]] # Deuxième élément de la liste
```

```
##           [,1]      [,2]       [,3]      [,4]
## [1,] 0.2254366 0.4296715 0.59592531 0.2641777
## [2,] 0.2745305 0.6516557 0.35804998 0.3987907
## [3,] 0.2723051 0.5677378 0.42880942 0.8361341
## [4,] 0.6158293 0.1135090 0.05190332 0.8647212
```

Exercice 3

```
x <- -5:5 # pour réduire la taille des tableaux affichés dans le pdf
x
```

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

```
x^2   # mettre aux carré tous les éléments de x
```

```
##  [1] 25 16  9  4  1  0  1  4  9 16 25
```

```
x + 3 # rajoute 3 aux tous les éléments de x
```

```
##  [1] -2 -1  0  1  2  3  4  5  6  7  8
```

```
2 * x # multiplie tous les éléments de x par 2
```

```
##  [1] -10  -8  -6  -4  -2   0   2   4   6   8  10
```

Exercice 4

```
      # Renvoie un tableau logical dont vrai veut
x == 0 # dire que l'élément correspondant est égal à 0
```

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

```
      # Renvoie un tableau logical dont vrai indique
x > 0 # les positions ou x_i est supérieur à 0
```

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

```
x[x > 0]  # Extraire les éléments de x qui sont plus grands que 0
```

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

```
x[x >= 0] # Extraire les éléments de x qui sont plus grans où égal à 0
```

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

Exercice 5

```
B %*% A
```

```
##      [,1] [,2] [,3]
## [1,]   60   75   90
## [2,]   60   75   90
## [3,]   60   75   90
## [4,]   60   75   90
```

```
# A %*% B # Non-conformable arguments, on peut pas multiplier une matrice 3 x 3 par une matrice 4 x 3..
A %*% t(B)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   77   77   77   77
## [2,]   77   77   77   77
## [3,]   77   77   77   77
```

Exercice 6

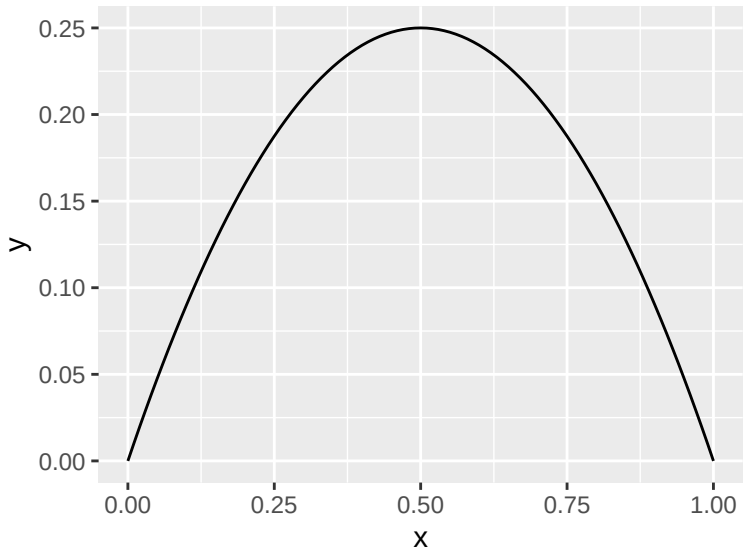
```
# 1.
d_x0 <- dnorm(0)   # 0.3989423
p_xlt0 <- pnorm(0) # P(X <= 0) = 0.5
```

Exercice 7

```
library(tidyverse)

x <- seq(0, 1, 0.01)
y <- x * (1 - x)

df <- tibble(x, y)
df |> ggplot(aes(x, y)) + geom_line()
```

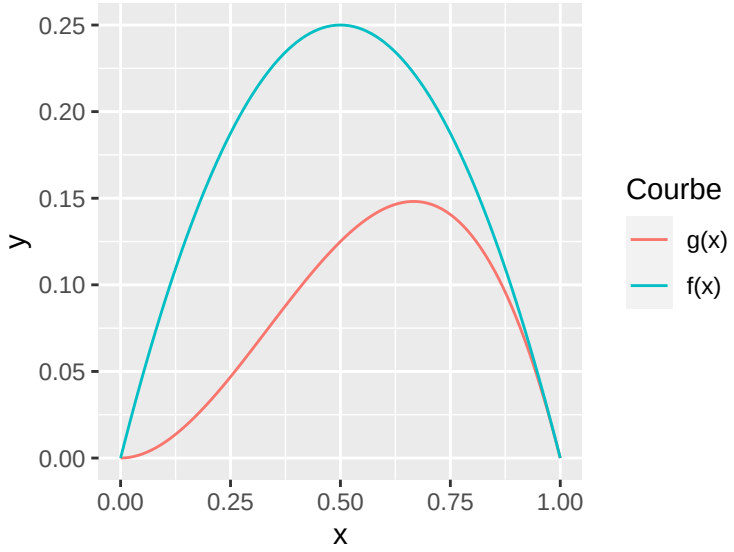



```r
imax <- which.max(y)

x_maximizer <- x[imax] # x* = 0.5
y_max <- y[imax]       # f(x*) = 0.25

g_x <- x^2 * (1 - x)

df |>
    mutate(g = g_x) |>
    pivot_longer(
        c(y, g),
        names_to = "curve",
        values_to = "y"
    ) |>
    ggplot(aes(x, y)) +
    scale_color_hue(labels = c("g(x)", "f(x)")) +
    geom_line(aes(color = curve)) +
    labs(col = "Courbe")
```



Exercice 8

```
x <- -10:10

sum(x == 0) # 1, cela compte le nombre d'éléments de x qui sont égaux à 0
```

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

```
sum(x > 0) # pareil, cette fois-ci les éléments qui sont plus grand  que zero
```

```
## [1] 10
```

```
sort(x^2) # renvoyer un tableau des éléments de x en ordre croissant
```

```
##  [1]   0   1   1   4   4   9   9  16  16  25  25  36  36  49  49  64  64  81  81 100 100
```

```
table(x^2) # renvoyer une "table" dont la première ligne est les éléments uniques
```

```
##
##   0   1   4   9  16  25  36  49  64  81 100
##   1   2   2   2   2   2   2   2   2   2   2
```

```
          # de x^2 et dont la deuxième ligne est le nombre de fois que la valeur du
          # même colonne aparraissent
unique(x^2) # Renvoi la première ligne de table(x^2), dans l'ordre décroissant
```

```
##  [1] 100  81  64  49  36  25  16   9   4   1   0
```

Exercice 9

```
ma_somme <- function(x) {

    acc <- 0
    n <- length(x)

    for (i in seq_len(n)) {
        acc <- acc + x[i]
    }

    acc
}

ma_variance <- function(x) {

    n <- length(x)
    my_sum <- ma_somme(x)

    x_bar <- (1 / n) * my_sum
    dev_sq <- (x - x_bar)^2

    ma_somme(dev_sq) / (n - 1)
}

x <- runif(1000)

v  <- var(x)
```

```r
mv <- ma_variance(x)

abs(v - mv) < 1E-16 # v == mv renvoie FALSE
```

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

Exercice 10

```r
compte <- function(char_seq, lettre) {
    ma_somme(char_seq == lettre)
}

composition <- function(char_seq) {

    comp <- vector("numeric", 4)
    lettres <- c("a", "c", "g", "t")
    i <- 1

    for (let in lettres) {
        comp[i] <- compte(char_seq, let)
        i <- i + 1
    }

    comp
}

bio_sequence <- c("a", "a", "t", "g", "a", "g", "c", "t", "a", "g", "c", "t", "g")

comp <- composition(bio_sequence)
comp
```

```
## [1] 4 2 4 3
```