code_R_1 : Caluls triviaux et vectors

Henri Laude

03/10/2020

Installation d'un package, vérification de la version de R

```
install.packages("matrixcalc")  # pour manipuler des matrices
library(matrixcalc)             # on lance le package

version                         # version de R
```

```
               _
platform       x86_64-pc-linux-gnu
arch           x86_64
os             linux-gnu
system         x86_64, linux-gnu
status
major          4
minor          0.2
year           2020
month          06
day            22
svn rev        78730
language       R
version.string R version 4.0.2 (2020-06-22)
nickname       Taking Off Again
```

R votre calculette

```
#--------#--------#--------#--------#--------#--------#--------#--------#--------
rm(list =ls())                # magie pour nettoyer l'environnement
```

```
10*(1+1+1.5)  # = 35         # calculs
```

```
[1] 35
```

```
10**2         # = 100        # carre
```

```
[1] 100
```

```
10^2          # idem
```

```
[1] 100
```

```
100**(1/2)    # = 10         # puissance
```

```
[1] 10
```

```r
100^(1/2)      # idem
```

```
[1] 10
```

```r
sqrt(100)      # = 10          # racine
```

```
[1] 10
```

```r
pi             # = 3.1416      # pi
```

```
[1] 3.141593
```

```r
cos(pi)        # = -1          # cos
```

```
[1] -1
```

```r
sin(pi/2)      # = 1           # sin
```

```
[1] 1
```

```r
exp(1)         # = 2.71882     # exponentielle
```

```
[1] 2.718282
```

```r
log(1)         # = 0           # log neperien
```

```
[1] 0
```

```r
round(2.566)      # arrondi a un entier
```

```
[1] 3
```

```r
round(pi,0)       # idem
```

```
[1] 3
```

```r
round(pi,2)       # arrondi 2 chiffres apres ,
```

```
[1] 3.14
```

```r
a <- 100
a
```

```
[1] 100
```

```r
print(a)
```

```
[1] 100
```

```r
v <- c(10,20,30)  # un vector
v
```

```
[1] 10 20 30
```

```r
length(v)          # longueur du vector
```

```
[1] 3
```

```r
length(a)
```

```
[1] 1
```

```r
is.vector(a)
```

```
[1] TRUE
```

```r
2*v+1              # sur chaque composante du vector
```

```
[1] 21 41 61
```

```r
v**2               # carre de chaque composante
```

```
[1] 100 400 900
```

```r
log(v)             # log de chaque composante
```

```
[1] 2.302585 2.995732 3.401197
```

```r
w <- c(1,2,3)      # un autre vector
v-w                # soustraction membre a membre
```

```
[1]  9 18 27
```

```r
v*w                # multiplication membre a membre
```

```
[1] 10 40 90
```

```r
v/w                # division membre a membre
```

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

```r
v%*%w              # produit scalaire
```

```
     [,1]
[1,]  140
```

```r
sum(v)          # = 60  somme
```

```
[1] 60
```

```r
mean(v)         # = 20  moyenne
```

```
[1] 20
```

```r
min(v)          # = 10  minimum
```

```
[1] 10
```

```r
max(v)          # = 30  maximum
```

```
[1] 30
```

```r
sd(v)           # = 10  ecart type
```

```
[1] 10
```

```r
median(v)       # = 20  medianne
```

```
[1] 20
```

```r
u <- c(1,2,3,4,5,6,7,8) # un autre vector
u[2]                    # deuxieme composante
```

```
[1] 2
```

```r
u[3:5]                  # nouveau vector
```

```
[1] 3 4 5
```

```r
                        # issu des composantes 3 a 5
```

```
u[8] <- 80              # affectation une composante
u
```

```
[1]  1  2  3  4  5  6  7 80
```

```
u[1:5] <- 1             # affectation 5 composantes
u
```

```
[1]  1  1  1  1  1  6  7 80
```

Jeux avec les vectors

```
v <- c(10,20,30,30,60,50)      # un vector
w <- c(20,10,31,31,61,51)      # un autre vector
u <- c(5 ,5 ,5 ,32,62,49)      # un autre vector

str(v)                    # jeter un oeil sur les data
```

```
 num [1:6] 10 20 30 30 60 50
```

```
sum(is.na(v))             # nb de valeurs manquantes
```

```
[1] 0
```

```
v_ <- c(NA,v,NA,NA)       # un vecteur avec 3 valeur manquantes
v_
```

```
[1] NA 10 20 30 30 60 50 NA NA
```

```
sum(is.na(v_))            # nb valeurs manquantes
```

```
[1] 3
```

```
range(v)                  # min et max du vector
```

```
[1] 10 60
```

```
range(v_)                 # min et max du vector ECHEC !
```

```
[1] NA NA
```

```
range(v_ , na.rm = TRUE)  # sans tenir compte des NA
```

```
[1] 10 60
```

```
quantile(v)                        # quartiles de v
```

```
  0%  25%  50%  75% 100%
10.0 22.5 30.0 45.0 60.0
```

```
quantile(v, probs =c(0,0.1,0.9,1)) # 80/20
```

```
  0%  10%  90% 100%
  10   15   55   60
```

```
summary(v)                    # resume
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.00   22.50   30.00   33.33   45.00   60.00
```

```r
sd(v, na.rm = TRUE)             # ecart type
```

```
[1] 18.61899
```

```r
cor(v,w)                        # coeff correlation entre vectors
```

```
[1] 0.9433573
```

```r
sort(v)                         # vector tri ordre croissant
```

```
[1] 10 20 30 30 50 60
```

```r
sort(v, decreasing = TRUE)      # vector tri ordre decroissant
```

```
[1] 60 50 30 30 20 10
```

```r
order(w)                        # donne pointer tri sur elements
```

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

```r
rank(w, ties.method="min")      # vecteur des rangs
```

```
[1] 2 1 3 3 6 5
```

```r
                                # de valeurs base plus petit=1
                                # "min" : style compet. sport

rank(w, ties.method="max")      # vecteur des rangs
```

```
[1] 2 1 4 4 6 5
```

```r
                                # de valeurs base plus petit=1
                                # "min" : style data-sciences



pmax(v,w,u)                     # valeurs max membre a membre
```

```
[1] 20 20 31 32 62 51
```

```r
pmin(v,w,u)                     # valeurs min membre a membre
```

```
[1]  5  5  5 30 60 49
```

```r
cumsum(v)                       # sommes cumulees
```

```
[1]  10  30  60  90 150 200
```

```r
cumprod(v)                      # produits successifs
```

```
[1] 1.00e+01 2.00e+02 6.00e+03 1.80e+05 1.08e+07 5.40e+08
```

```r
cummax(w)                       # maximum entre membre
```

```
[1] 20 20 31 31 61 61
```

```r
                                # considere et membres precedents
cummin(w)                       # idem avec min
```

```
[1] 20 10 10 10 10 10
```

Logique Booléenne

```
a <- 1
b <- 2

(a == 1)                        # TRUE
```

```
[1] TRUE
```

```
(a == b)                        # FALSE
```

```
[1] FALSE
```

```
(a <= b)                        # TRUE
```

```
[1] TRUE
```

```
A <- c(TRUE,TRUE,FALSE,FALSE)
B <- c(TRUE,FALSE,TRUE,FALSE)
A & B                           # table de verite de "et"
```

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

```
A | B                           # table de verite de "ou"
```

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

```
! A                             # non-A
```

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

```
xor(A,B)                        # table verite ou exclusif
```

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

```
!A|B                            # table de l'implication A==>B
```

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

```
str(A)                          # vector compose de logical
```

```
 logi [1:4] TRUE TRUE FALSE FALSE
```

```
c <- (a > b)                    # stocker le resultat d'un test
c
```

```
[1] FALSE
```

```
v <- c(10,20,30,30,60,50)       # un vector
t <- (v > 30)                   # vecteur resultant du test
t                               # membre a membre
```

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

```
w <- v[(v>30)]                  # on ne garde que les membres
                                # avec expression TRUE
w
```

```
[1] 60 50
```

```
which(v == 30)                  # trouve les indices ou
```

```
[1] 3 4
```

```
                                # membre egal a 30

which(v == max(v))              # trouve les indices ou
```

```
[1] 5
```

```
                                # membre egal val max membre

which(v == min(v))              # idem mais recherche min
```

```
[1] 1
```

```
s <- 1*t                        # transformation en vecteur 1,0
s
```

```
[1] 0 0 0 0 1 1
```

```
v <- c(10,20,70,30,60,50)       # un vector
all(v > 5)                      # ?toutes les val sont sup a 5
```

```
[1] TRUE
```

```
any(v < 5)                      # ?une valeur inf a 5
```

```
[1] FALSE
```