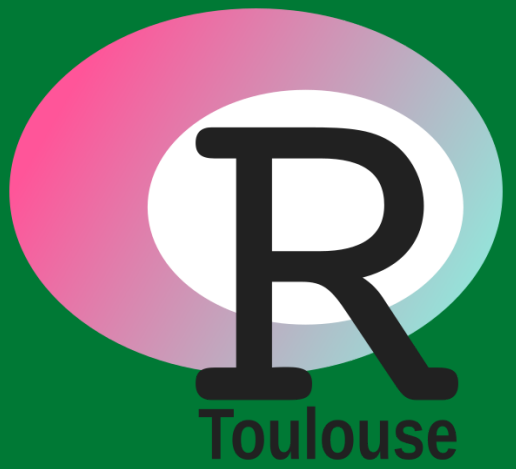
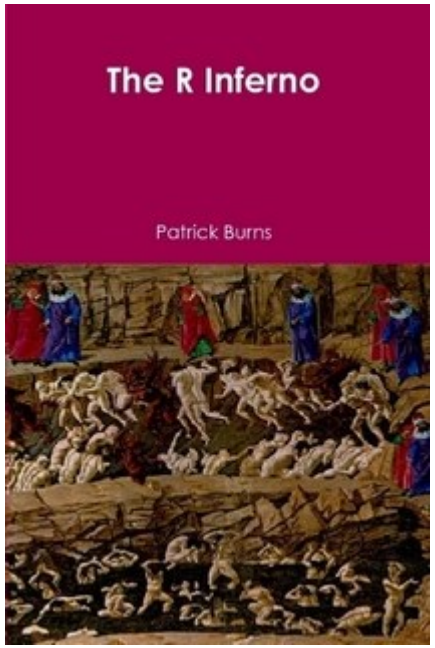
(ノ°□°)ノ ⌒ ┻━┻

C'est l'enfeR : Petits pièges du langage R

Guillaume Devailly
2018/10/25



Fortement inspiré de **The R Inferno** par Patrick Burns



eBook gratuit : https://www.burns-stat.com/documents/books/the-r-inferno/

= vs <-

Pleins d'espaces

```
x<-1
x< -1
x=1
x=-1
```

Pleins d'espaces

```r
x<-1
x< -1
x=1
x=-1
```

```r
x <- 1
x < -1
x = 1
x = -1
```

"<-" != "="

```
data.frame(
    a1 <- 1:3
)
##   a1....1.3
## 1         1
## 2         2
## 3         3

a1
## [1] 1 2 3

data.frame(
    a2 = 1:3
)
##   a2
## 1  1
## 2  2
## 3  3

a2
## Error in eval(expr, envir, enclos): objet 'a2' introuvable
```

"<-" != "="

```
system.time(
    x <- 1:5
)
##    user  system elapsed
##       0       0       0

x
## [1] 1 2 3 4 5

system.time(
    y = 1:5
)
## Error in system.time(y = 1:5): argument inutilisé (y = 1:5)

y
## Error in eval(expr, envir, enclos): objet 'y' introuvable
```

Comparaisons

```r
x <- 1:5
x
```

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

```r
x == 2|3
```

```r
x <- 1:5
x
```

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

```r
x == 2|3
```

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

```r
x <- 1:5
x
```

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

```r
x == 2|3
```

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

```r
x == (2|3)
```

```r
x <- 1:5
x
```

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

```r
x == 2|3
```

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

```r
x == (2|3)
```

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

```r
x <- 1:5
x
```

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

```r
x == 2|3
```

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

```r
x == (2|3)
```

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

```r
x == 2 | x == 3
```

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

```r
x %in% c(2, 3)
```

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

```
x
```

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

```
1 < x < 4
```

```
## Error: <text>:1:7: '<' inattendu(e)
## 1: 1 < x <
##           ^
```

```
1 < x & x < 4
```

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

```
0.1/1
```

```
## [1] 0.1
```

```
0.1/1
```

```
## [1] 0.1
```

```
0.1/1 == 0.1
```

```
0.1/1
```

```
## [1] 0.1
```

```
0.1/1 == 0.1
```

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

```
0.1/1
```

```
## [1] 0.1
```

```
0.1/1 == 0.1
```

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

Tout va vien.

```
0.1/1
```

```
## [1] 0.1
```

```
0.1/1 == 0.1
```

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

Tout va vien.

```
0.3/3
```

```
0.1/1
```

```
## [1] 0.1
```

```
0.1/1 == 0.1
```

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

Tout va vien.

```
0.3/3
```

```
## [1] 0.1
```

```
0.1/1
```

```
## [1] 0.1
```

```
0.1/1 == 0.1
```

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

Tout va vien.

```
0.3/3
```

```
## [1] 0.1
```

```
0.3/3 == 0.1
```

```
0.1/1
```

```
## [1] 0.1
```

```
0.1/1 == 0.1
```

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

Tout va vien.

```
0.3/3
```

```
## [1] 0.1
```

```
0.3/3 == 0.1
```

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

(ー_ー゛)

Résumons :

```
mon_vecteur <- c(0.1/1, 0.2/2, 0.3/3, 0.4/4, 0.5/5, 0.6/6, 0.7/7, 0.8/8)
mon_vecteur
```

```
## [1] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
```

Résumons :

```
mon_vecteur <- c(0.1/1, 0.2/2, 0.3/3, 0.4/4, 0.5/5, 0.6/6, 0.7/7, 0.8/8)
mon_vecteur
```

```
## [1] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
```

```
mon_vecteur == 0.1
```

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

Résumons :

```
mon_vecteur <- c(0.1/1, 0.2/2, 0.3/3, 0.4/4, 0.5/5, 0.6/6, 0.7/7, 0.8/8)
mon_vecteur
```

```
## [1] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
```

```
mon_vecteur == 0.1
```

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

5/8, pas terrible. :-/

Les floats sonts des approximations

```r
print(0.1   , digits = 18)
## [1] 0.100000000000000006

print(0.3/3, digits = 18)
## [1] 0.099999999999999917
```

Les floats sonts des approximations

```
print(0.1  , digits = 18)
## [1] 0.100000000000000006

print(0.3/3, digits = 18)
## [1] 0.0999999999999999917
```

```
pryr::bytes(0.1  )
## [1] "3F B9 99 99 99 99 99 9A"

pryr::bytes(0.3/3)
## [1] "3F B9 99 99 99 99 99 99"

pryr::bits( 0.1  )
## [1] "00111111 10111001 10011001 10011001 10011001 10011001 10011001 10011010"

pryr::bits( 0.3/3)
## [1] "00111111 10111001 10011001 10011001 10011001 10011001 10011001 10011001"
```

Une solution ?

```
epsilon <- 10^-10

mon_vecteur
## [1] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1

mon_vecteur == 0.1
## [1]  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE

abs(mon_vecteur - 0.1) < epsilon
## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

Redéfinissons '=='

```
0.3/3 == 0.1
## [1] FALSE

`==` <- function(x, y, epsilon = 10^-10) {
    abs(x - y) < epsilon
}

0.3/3 == 0.1
## [1] TRUE
```

Redéfinissons '=='

```
0.3/3 == 0.1
## [1] FALSE

`==` <- function(x, y, epsilon = 10^-10) {
    abs(x - y) < epsilon
}

0.3/3 == 0.1
## [1] TRUE
```

```
"a" == "a"
## Error in x - y: argument non numérique pour un opérateur binaire
```

Redéfinissons '=='

```
0.3/3 == 0.1
## [1] FALSE

`==` <- function(x, y, epsilon = 10^-10) {
    abs(x - y) < epsilon
}

0.3/3 == 0.1
## [1] TRUE
```

```
"a" == "a"
## Error in x - y: argument non numérique pour un opérateur binaire
```

Pas bien !

```
rm(`==`)
"a" == "a"
## [1] TRUE
```

Une meilleure solution :

```
`%~=%` <- function(x, y, epsilon = 10^-10) {
    abs(x - y) < epsilon
}

0.3/3 == 0.1
## [1] FALSE

0.3/3 %~=% 0.1
## [1] Inf
```

Une meilleure solution :

```
`%~=%` <- function(x, y, epsilon = 10^-10) {
    abs(x - y) < epsilon
}

0.3/3 == 0.1
## [1] FALSE

0.3/3 %~=% 0.1
## [1] Inf
```

```
(0.3/3) %~=% 0.1
## [1] TRUE
```

Un problème assez fréquent

```
my_seq <- seq(0, 0.4, len = 5)
my_seq
```

```
## [1] 0.0 0.1 0.2 0.3 0.4
```

```
my_seq[4]
```

```
## [1] 0.3
```

```
my_seq[4] == 0.3
```

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

Quizz : NA, NaN, NULL

```
3L == NA

3L == NaN

3L == NULL
```

Quizz : NA, NaN, NULL

```
3L == NA
```

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

```
3L == NaN
```

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

```
3L == NULL
```

```
## logical(0)
```

Quizz : NA, NaN, NULL

```
NA == NA

NaN == NaN

NULL == NULL
```

Quizz : NA, NaN, NULL

```
NA == NA

NaN == NaN

NULL == NULL
```

```
## [1] NA

## [1] NA

## logical(0)
```

Quizz : NA, NaN, NULL

```
NA == NaN

NA == NULL

NaN == NULL
```

Quizz : NA, NaN, NULL

```
NA == NaN

NA == NULL

NaN == NULL
```

```
## [1] NA

## logical(0)

## logical(0)
```

Quizz : NA, NaN, NULL

```
NA == NaN
```

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

```
NA == NULL
```

```
## logical(0)
```

```
NaN == NULL
```

```
## logical(0)
```

Utilisez `is.na()`, `is.nan()` et `is.null()` pour tester contre `NA`, `NaN` et `NULL`.

```
is.null(logical(0))
```

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

Inf == Inf

```
1/0

1/-0

0/0

exp(Inf)

exp(Inf) == log(Inf)
```

Inf == Inf

```
1/0

1/-0

0/0

exp(Inf)

exp(Inf) == log(Inf)
```

```
## [1] Inf

## [1] -Inf

## [1] NaN

## [1] Inf

## [1] TRUE
```

Inf == Inf

```
1/0
```

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

```
1/-0
```

```
## [1] -Inf
```

```
0/0
```

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

```
exp(Inf)
```

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

```
exp(Inf) == log(Inf)
```

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

L'aide de R :

NaN means 'Not a Number'

Inf == Inf

```
1/0

1/-0

0/0

exp(Inf)

exp(Inf) == log(Inf)
```

```
## [1] Inf

## [1] -Inf

## [1] NaN

## [1] Inf

## [1] TRUE
```

L'aide de R :

NaN means 'Not a Number'

```
is.numeric(NaN)
## [1] TRUE
```

Une dispute sur les arguments

```
min(4, 5, 1, 2, 3)
```

```r
min(4, 5, 1, 2, 3)
```

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

```r
min(4, 5, 1, 2, 3)
```

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

```r
max(4, 5, 1, 2, 3)
```

```r
min(4, 5, 1, 2, 3)
```

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

```r
max(4, 5, 1, 2, 3)
```

```
## [1] 5
```

```r
min(4, 5, 1, 2, 3)
```

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

```r
max(4, 5, 1, 2, 3)
```

```
## [1] 5
```

Tout va bien.

```
min(4, 5, 1, 2, 3)
```

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

```
max(4, 5, 1, 2, 3)
```

```
## [1] 5
```

Tout va bien.

```
mean(4, 5, 1, 2, 3)
```

```
min(4, 5, 1, 2, 3)
```

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

```
max(4, 5, 1, 2, 3)
```

```
## [1] 5
```

Tout va bien.

```
mean(4, 5, 1, 2, 3)
```

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

```
min(4, 5, 1, 2, 3)
```

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

```
max(4, 5, 1, 2, 3)
```

```
## [1] 5
```

Tout va bien.

```
mean(4, 5, 1, 2, 3)
```

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

Pas d'erreure, pas de *warnings*, un résultat de type attendu. (ー_ゝ)

```r
min(4, 5, 1, 2, 3)
```

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

```r
max(4, 5, 1, 2, 3)
```

```
## [1] 5
```

Tout va bien.

```r
mean(4, 5, 1, 2, 3)
```

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

Pas d'erreure, pas de *warnings*, un résultat de type attendu. (ー_ゝ)

```r
median(4, 5, 1, 2, 3)
```

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

Pas vraiement un problème en pratique :

```
x <- c(4, 5, 1, 2, 3)
min(x)
```

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

```
max(x)
```

```
## [1] 5
```

```
mean(x)
```

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

```
median(x)
```

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

"L'échantillonage aléatoire" aléatoire

```
x <- c(3.14, 1.41, 42)

sample(x, size = 7, replace = TRUE)
```

```
## [1]  3.14 42.00  1.41  3.14  1.41  3.14  1.41
```

"L'échantillonage aléatoire" aléatoire

```
x <- c(3.14, 1.41, 42)

sample(x, size = 7, replace = TRUE)
```

```
## [1]  3.14 42.00  1.41  3.14  1.41  3.14  1.41
```

```
x <- c(3.14)

sample(x, size = 7, replace = TRUE)
```

"L'échantillonage aléatoire" aléatoire

```
x <- c(3.14, 1.41, 42)

sample(x, size = 7, replace = TRUE)
```

```
## [1]  3.14 42.00  1.41  3.14  1.41  3.14  1.41
```

```
x <- c(3.14)

sample(x, size = 7, replace = TRUE)
```

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

"L'échantillonage aléatoire" aléatoire

```
x <- c(3.14, 1.41, 42)

sample(x, size = 7, replace = TRUE)
```

```
## [1]  3.14 42.00  1.41  3.14  1.41  3.14  1.41
```

```
x <- c(3.14)

sample(x, size = 7, replace = TRUE)
```

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

Vérifier la taille de `x` avant de le passer a `sample()`

Les facteurs accidentels

stringsAsFactors = FALSE

```r
x <- factor(c(105:100, 105, 104))
x
## [1] 105 104 103 102 101 100 105 104
## Levels: 100 101 102 103 104 105
x >= 103
## Warning in Ops.factor(x, 103): '>=' not meaningful for factors
## [1] NA NA NA NA NA NA NA NA
```

```r
x <- factor(c(105:100, 105, 104))
x
## [1] 105 104 103 102 101 100 105 104
## Levels: 100 101 102 103 104 105
x >= 103
## Warning in Ops.factor(x, 103): '>=' not meaningful for factors
## [1] NA NA NA NA NA NA NA NA
```

```r
as.numeric(x) >= 103
## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```r
x <- factor(c(105:100, 105, 104))
x
## [1] 105 104 103 102 101 100 105 104
## Levels: 100 101 102 103 104 105
x >= 103
## Warning in Ops.factor(x, 103): '>=' not meaningful for factors
## [1] NA NA NA NA NA NA NA NA
```

```r
as.numeric(x) >= 103
## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```r
as.numeric(x)
## [1] 6 5 4 3 2 1 6 5

as.numeric(as.character(x))
## [1] 105 104 103 102 101 100 105 104

as.numeric(as.character(x)) >= 103
## [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE

as.numeric(levels(x))[x]
## [1] 105 104 103 102 101 100 105 104
```

fusions de factors

```
x <- factor(c(5, 6))
x
```

```
## [1] 5 6
## Levels: 5 6
```

```
y <- factor(c(10, 11))
y
```

```
## [1] 10 11
## Levels: 10 11
```

```
c(x, y)
```

fusions de factors

```
x <- factor(c(5, 6))
x
```

```
## [1] 5 6
## Levels: 5 6
```

```
y <- factor(c(10, 11))
y
```

```
## [1] 10 11
## Levels: 10 11
```

```
c(x, y)
```

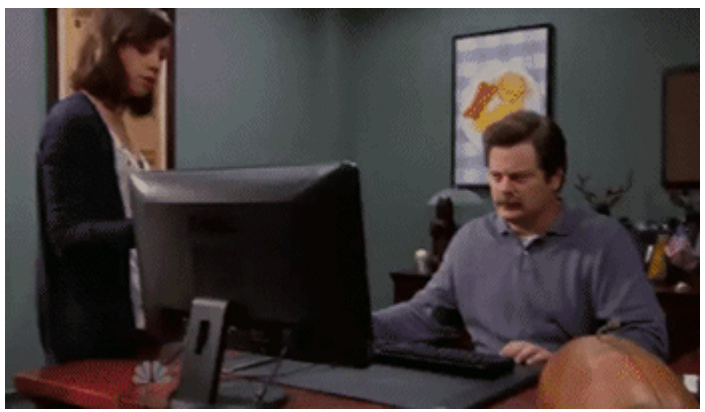
```
## [1] 1 2 1 2
```

fusions de factors

```
x
## [1] 5 6
## Levels: 5 6

y
## [1] 10 11
## Levels: 10 11

factor(c(as.numeric(as.character(x)), as.numeric(as.character(y))))
## [1] 5  6  10 11
## Levels: 5 6 10 11

unlist(list(x, y))
## [1] 5  6  10 11
## Levels: 5 6 10 11
```

Factors, encore :

```
df <- data.frame(a = 2:3, b = c("x", "y"))
df
##   a b
## 1 2 x
## 2 3 y

df[1, ]
##   a b
## 1 2 x

as.character(df[1, ])
## [1] "2" "1"
```



Factors, encore :

```
df[1, , drop = TRUE]
## $a
## [1] 2
##
## $b
## [1] x
## Levels: x y

df <- data.frame(a = 2:3, b = c("x", "y"), stringsAsFactors = FALSE)
df[1, ]
##   a b
## 1 2 x
```

Extractions

```r
tb1 <- data.frame(x1 = 1:5)
tb1
##   x1
## 1  1
## 2  2
## 3  3
## 4  4
## 5  5

tb2 <- tb1[c(2, 4, 5), ]
```

```r
tb1 <- data.frame(x1 = 1:5)
tb1
##   x1
## 1  1
## 2  2
## 3  3
## 4  4
## 5  5

tb2 <- tb1[c(2, 4, 5), ]
```

```r
tb2
## [1] 2 4 5
```

```r
tb1 <- data.frame(x1 = 1:5)
tb1
##   x1
## 1  1
## 2  2
## 3  3
## 4  4
## 5  5

tb2 <- tb1[c(2, 4, 5), ]
```

```r
tb2
## [1] 2 4 5
```

```r
tb2 <- tb1[c(2, 4, 5), , drop = FALSE]
tb2
##   x1
## 2  2
## 4  4
## 5  5
```

Une petite énigme

```
i
```

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

```
x <- 1:5
x
```

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

```
x[i]
```

Une petite énigme

```
i
```

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

```
x <- 1:5
x
```

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

```
x[i]
```

```
## [1] 2
```

Une petite énigme

```
i
```

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

```
x <- 1:5
x
```

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

```
x[i]
```

```
## [1] 2
```

```
as.integer(i)
```

```
## [1] 2
```

Une petite énigme

```
i
```

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

```
x <- 1:5
x
```

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

```
x[i]
```

```
## [1] 2
```

```
as.integer(i)
```

```
## [1] 2
```

```
i <- 3 - 10^-15
```

```
myMat <- matrix(1:6, ncol = 2)
myMat
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

```
df1 <- data.frame(X = 101:103,
                  Y = myMat)
df1
##     X Y.1 Y.2
## 1 101   1   4
## 2 102   2   5
## 3 103   3   6
```

```
df2 <- data.frame(X = 101:103)
df2$Y <- myMat
df2
##     X Y.1 Y.2
## 1 101   1   4
## 2 102   2   5
## 3 103   3   6
```

```
myMat <- matrix(1:6, ncol = 2)
myMat
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

```
df1 <- data.frame(X = 101:103,
                  Y = myMat)
df1
##     X Y.1 Y.2
## 1 101   1   4
## 2 102   2   5
## 3 103   3   6
dim(df1)
## [1] 3 3
df1$Y
## NULL
df1$Y.1
## [1] 1 2 3
```

```
df2 <- data.frame(X = 101:103)
df2$Y <- myMat
df2
##     X Y.1 Y.2
## 1 101   1   4
## 2 102   2   5
## 3 103   3   6
dim(df2)
## [1] 3 2
df2$Y
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
df2$Y.1
## NULL
```

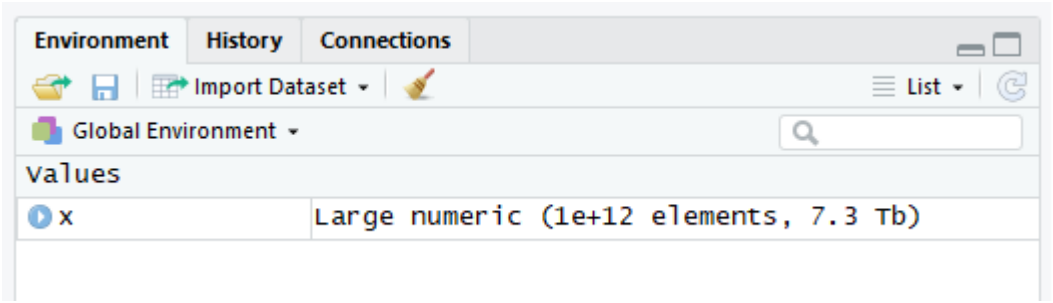
Error: cannot allocate vector of size...

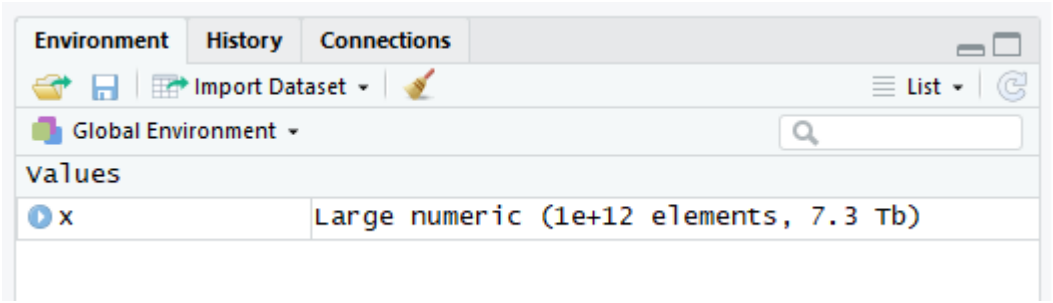
Nouvelle édition

```r
x <- 1:(10^12)
head(x)
## [1] 1 2 3 4 5 6

tail(x)
## [1] 1e+12 1e+12 1e+12 1e+12 1e+12 1e+12

x[1]
## [1] 1

x[5]
## [1] 5

x[12345]
## [1] 12345
```

```r
x <- 1:(10^12)
head(x)
## [1] 1 2 3 4 5 6

tail(x)
## [1] 1e+12 1e+12 1e+12 1e+12 1e+12 1e+12

x[1]
## [1] 1

x[5]
## [1] 5

x[12345]
## [1] 12345
```

```r
x[5] <- 2
## Error: impossible d'allouer un vecteur de taille 7450.6 Go

x[5] <- 2L
## Error: impossible d'allouer un vecteur de taille 7450.6 Go

x
## Error: impossible d'allouer un vecteur de taille 7450.6 Go
```


Environment
History
Connections
Import Dataset
List
Global Environment
Values
x
Large numeric (1e+12 elements, 7.3 Tb)



La magie d'ALTREP

```
x[2] <- 1
```

```
## Error: impossible d'allouer un vecteur de taille 7450.6 Go
```

```
sum(x)
```

```
## [1] 5e+23
```

ALTREP

Plus d'infos :

- ftp://stat.ethz.ch/Teaching/maechler/R/eRum_2018_ProgR-ALTREP.html#20
- http://homepage.stat.uiowa.edu/~luke/talks/nzsa-2017.pdf
- https://svn.r-project.org/R/branches/ALTREP/ALTREP.html

Pour aller plus loin

- The R inferno (Patrick Burns) : www.burns-stat.com/documents/books/the-r-inferno/
- C'est l'enfeR : bioinfo-fr.net/cest-lenfer
- Petite collection d'exemples : github.com/EdinbR/edinbr-talks/blob/master/2015-11-18/Rinferno.R

Bonus

```
T
## [1] TRUE

T <- FALSE

T
## [1] FALSE
```

```
c(5, 10, 12)
## [1]  5 10 12

c <- function(...) list(...)

c(5, 10, 12)
## [[1]]
## [1] 5
##
## [[2]]
## [1] 10
##
## [[3]]
## [1] 12
```

Bonus

```
plus_deux <- function(x) {
    return(x + 2)
}

plus_deux(5)
## [1] 7

return <- function(x) x + 10

plus_deux(5)
## [1] 17
```

```
rm(T, c, return)
```

Bonus

```r
mon_test <- c(TRUE, FALSE, FALSE)

if(mon_test) {
    message("Succés !")
} else {
    message("Echec !")
}
```

```
## Warning in if (mon_test) {: la condition a une longueur > 1 et seul le
## premier élément est utilisé

## Succés !
```