Programacion R grupo GJJAB

Baptiste Caillé[a], Gastón Barmat[a], Juan Ignacio Gutiérrez Glielmi[a], Jacinta Calle Monzo[a], Agustina Salvarredi[a]

[a]*Facultad de ingenieria Universidad Nacional de Cuyo - Centro Universitario Edifico 6 M5502KAF*

Abstract

Este Informe repite el trabajo que hicimos en el módulo 2 en R

Keywords: RStudio, RCran, programacion

1. Consigas del trabajo

El trabajo a presentar implica la revisión de los algoritmos que se presentan a continuación. Primero debe ejecutarlos en la línea de comandos de la consola. Luego, debe elegir un método para medir el rendimiento. y comparar los resultados con otros colegas usando otros métodos para medir el rendimiento. Luego todo tiene que ser entregado como informe en formato pdf RStudio, archivos RMarkdow.

1.0.1. Algoritmos a revisar y Metodos para medir la performance

Los algoritmos a revisar son:

- Generacion de un *Vector Secuencia*,
- Implementacion de una *Serie de Fibonacci*,
- Ordenacion de un vector por *Metodo burbuja*.

Los metodos para ver la performance son:

- Sys.time,
- Biblioteca tictoc,
- Biblioteca rbenchmark,
- Biblioteca Microbenchamrk.

2. Generar un vector secuencia

2.1. for

```
A <- vector()
for(i in 0:50000){
  A[i+1] <- i*2
}
tail(A)
```

*Corresponding author

Email addresses: baptiste.caille.move@gmail.com (Baptiste Caillé), gastonbarmat@gmail.com (Gastón Barmat), juangglielmi@hotmail.com (Juan Ignacio Gutiérrez Glielmi), jacicalle@hotmail.com (Jacinta Calle Monzo), agusalvarredi@gmail.com (Agustina Salvarredi)

```
## [1]  99990  99992  99994  99996  99998 100000
```

```
head(A)
```

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

2.2. seq

```
B <- vector()
B<- seq(0,100000,2)
tail(B)
```

```
## [1]  99990  99992  99994  99996  99998 100000
```

```
head(B)
```

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

2.3. Performance

uso del algoritmo *Biblioteca tictoc "In general, calls to tic and toc start the timer when the tic call is made and stop the timer when the toc call is made, recording the elapsed time between the calls from proc.time."*

```
library(tictoc)
A <- vector()
B <- vector()

tic.clearlog()

tic
```

```
## function (msg = NULL, quiet = TRUE, func.tic = NULL, ...)
## {
##     stim <- get(".tictoc", envir = asNamespace("tictoc"))
##     smsg <- get(".ticmsg", envir = asNamespace("tictoc"))
##     tic <- proc.time()["elapsed"]
##     if (!is.null(func.tic)) {
##         outmsg <- func.tic(tic, msg, ...)
##         if (!quiet)
##             writeLines(outmsg)
##     }
##     push(stim, tic)
##     push(smsg, msg)
##     invisible(tic)
## }
## <bytecode: 0x000002231115c0e0>
## <environment: namespace:tictoc>
```

```
for(C in 0:50000){
  A[C+1] <- C*2
}
toc(log = TRUE, quiet = TRUE)

tic
```

```
## function (msg = NULL, quiet = TRUE, func.tic = NULL, ...)
## {
##     stim <- get(".tictoc", envir = asNamespace("tictoc"))
##     smsg <- get(".ticmsg", envir = asNamespace("tictoc"))
##     tic <- proc.time()["elapsed"]
##     if (!is.null(func.tic)) {
##         outmsg <- func.tic(tic, msg, ...)
##         if (!quiet)
##             writeLines(outmsg)
##     }
##     push(stim, tic)
##     push(smsg, msg)
##     invisible(tic)
## }
## <bytecode: 0x000002231115c0e0>
## <environment: namespace:tictoc>
```

```
B <- seq(0,100000,2)
toc(log = TRUE, quiet = TRUE)

unlist(tic.log())
```

```
## NULL
```

3. Serie de Fibonacci

```
w <- vector()
w[1]<- 0
w[2]<- 1
for(var in 0:15)
{
  w[var+3] <- w[var+2]+w[var+1]
}
head(w)
```

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

3.1. Performance

uso del algoritmo *Biblioteca rbenchmark.*

"The library consists of just one function, benchmark, which is a simple wrapper around system.time. Given a specification of the benchmarking process (counts of replications, evaluation environment) and an arbitrary number of expressions, benchmark evaluates each of the expressions in the specified environment, replicating the evaluation as many times as specified, and returning the results conveniently wrapped into a data frame"

```
library(rbenchmark)

k <- vector()
w <- vector()
```

```
w = function(it){
  k[1] <- 0
  k[2] <- 1
  for(i in 0:it){
    k[i+3] <- (k[i+2]+k[i+1])
  }
  return(k)
}


it=197
benchmark(w(it), replications = 1000)
```

```
##    test replications elapsed relative user.self sys.self user.child sys.child
## 1 w(it)         1000    0.12        1      0.13        0         NA        NA
```

- *elapsed*: tiempo acumulado
- *relative*: razon con la prueba mas rapida.
- *user.self*: CPU time spent by the current process
- *sys.self*: CPU time spent by the kernel (the operating system) on behalf of the current process. # Algoritmo para la pesadilla de Gauss

```
for(i in 0:5)
{ a<-i
b <-i+1
c <- a+b

print(c)
}
```

```
## [1] 1
## [1] 3
## [1] 5
## [1] 7
## [1] 9
## [1] 11
```

4. Ordenacion de un vector por Metodo Burbuja

A continuacion estan las lineas de codigo para ordenar, de mayor a menor, por metodo bubble un vector de 150 numeros elegidos aleatoriamente entre el 1 y el 1000000:

```
example <- sample(1:1000000,150)

head(example)
```

```
## [1] 318037 501066 725722 417345 482529 614518
```

```
tail(example)
```

```
## [1]  93242 726546 666106 418298 587862 429746
```

```r
bubble = function(example){
  n <- length(example)
  for(i in 1:(n-1)){
    for(j in 1:(n-i)){
      if (example[j] < example[j+1]){
        temporal <- example[j]
        example[j] <- example[j+1]
        example[j+1] <- temporal
      }
    }
  }
  return(example)
}

example_ordenada <- bubble(example)

head(example_ordenada)
```

```
## [1] 989126 983177 979108 971968 960103 953081
```

```r
tail(example_ordenada)
```

```
## [1] 22999 21761 20510 15286 13704  6279
```

4.1. Performance

uso del algoritmo *Biblioteca Microbenchmark.*

"Microbenchmark serves as a more accurate replacement of the often seen system.time(replicate(1000, expr)) expression. It tries hard to accurately measure only the time it takes to evaluate expr. To achieved this, the sub-millisecond (supposedly nanosecond) accurate timing functions most modern operating systems provide are used."

A continuacion vemos las lineas de codigo para calcular el tiempo de ejecucion de el codigo para ordenar por metodo bubble. Tomaremos una muestra de 2000 numeros entre el 1 y el 1000000 Lo compararemos con el comando *sort* de R:

```r
library(microbenchmark)

example <- sample(1:1000000,2000)

bubble = function(example){
  n <- length(example)
  for(i in 1:(n-1)){
    for(j in 1:(n-i)){
      if (example[j] < example[j+1]){
        temporal <- example[j]
        example[j] <- example[j+1]
        example[j+1] <- temporal
      }
    }
  }
  return(example)
```

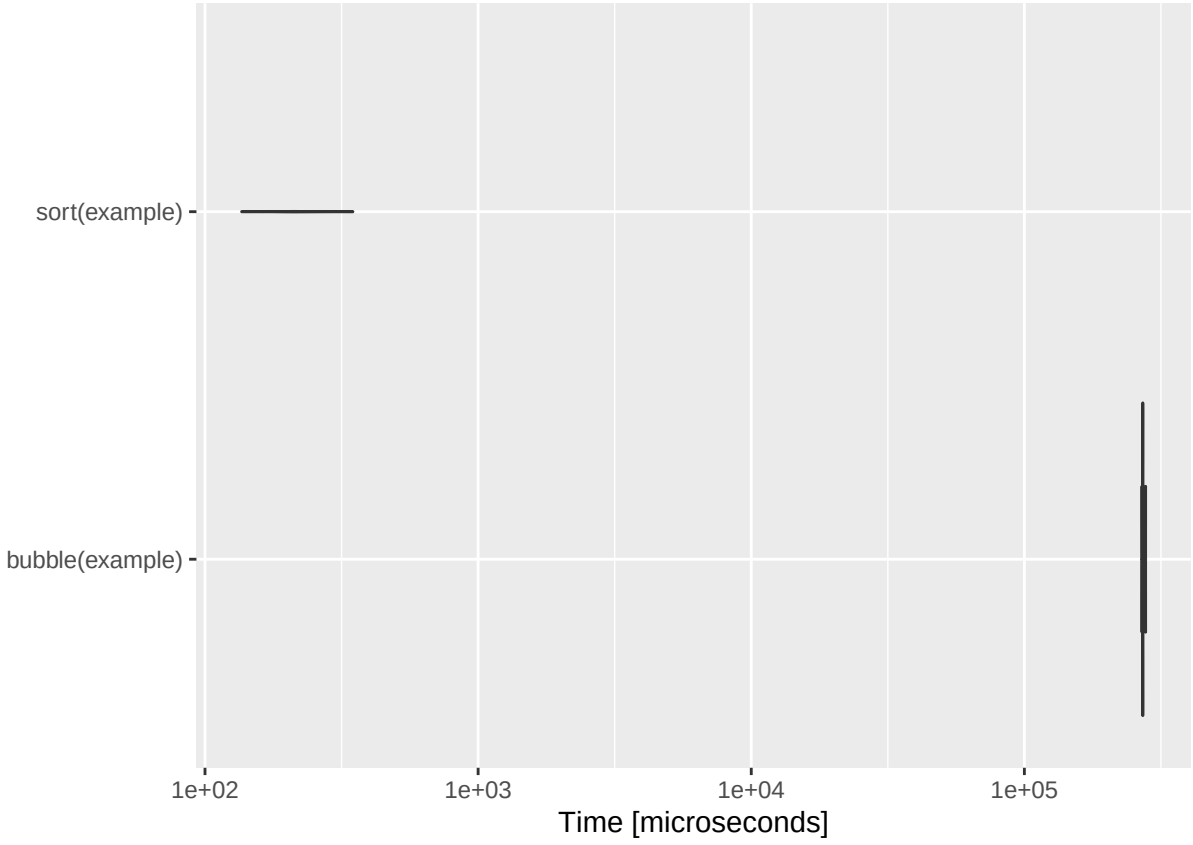
```
}


microbenchmark(bubble(example),sort(example), times=5)
```

```
## Unit: microseconds
##              expr      min       lq      mean   median       uq      max neval
##  bubble(example) 268720.4 269347.5 291359.28 284772.6 300506.0 333449.9     5
##    sort(example)    138.2    144.4    224.64    207.4    216.7    416.5     5
```

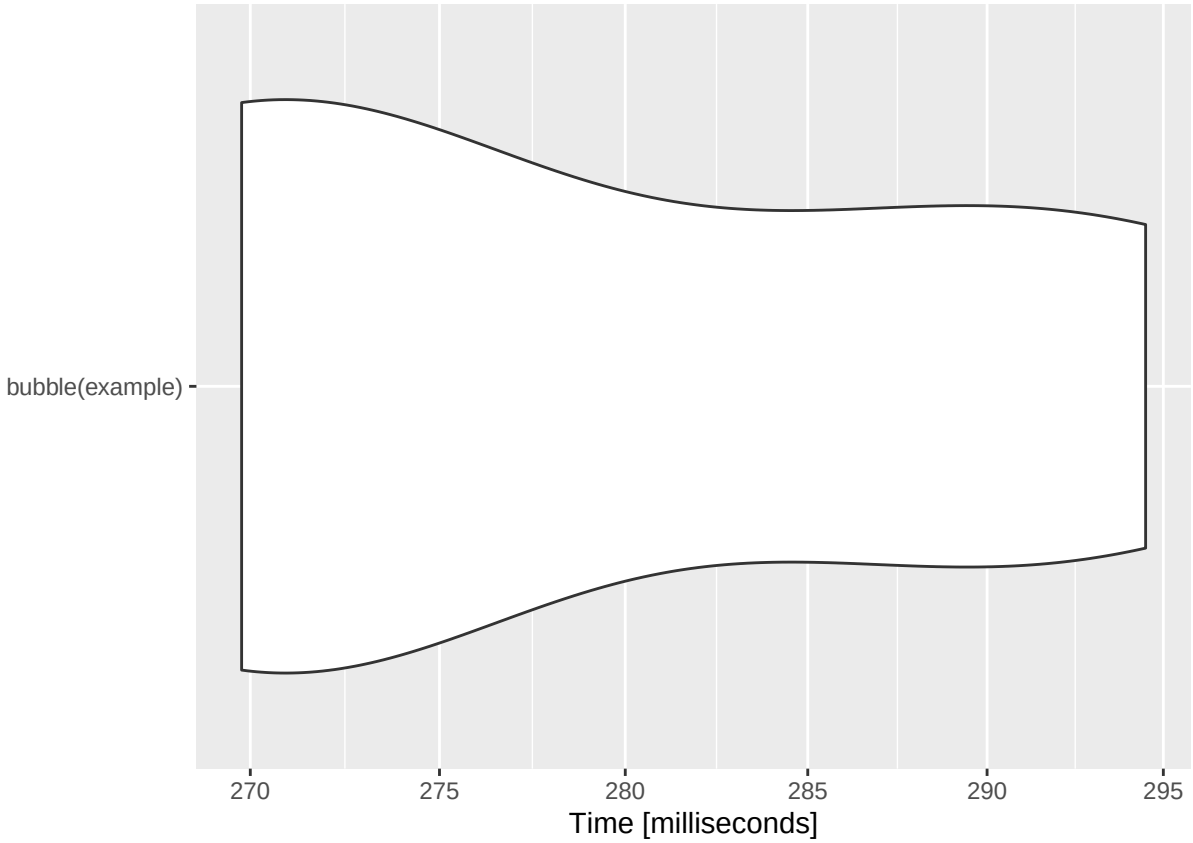
```
library(ggplot2)
autoplot(microbenchmark(bubble(example),sort(example), times=5))
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing or
```



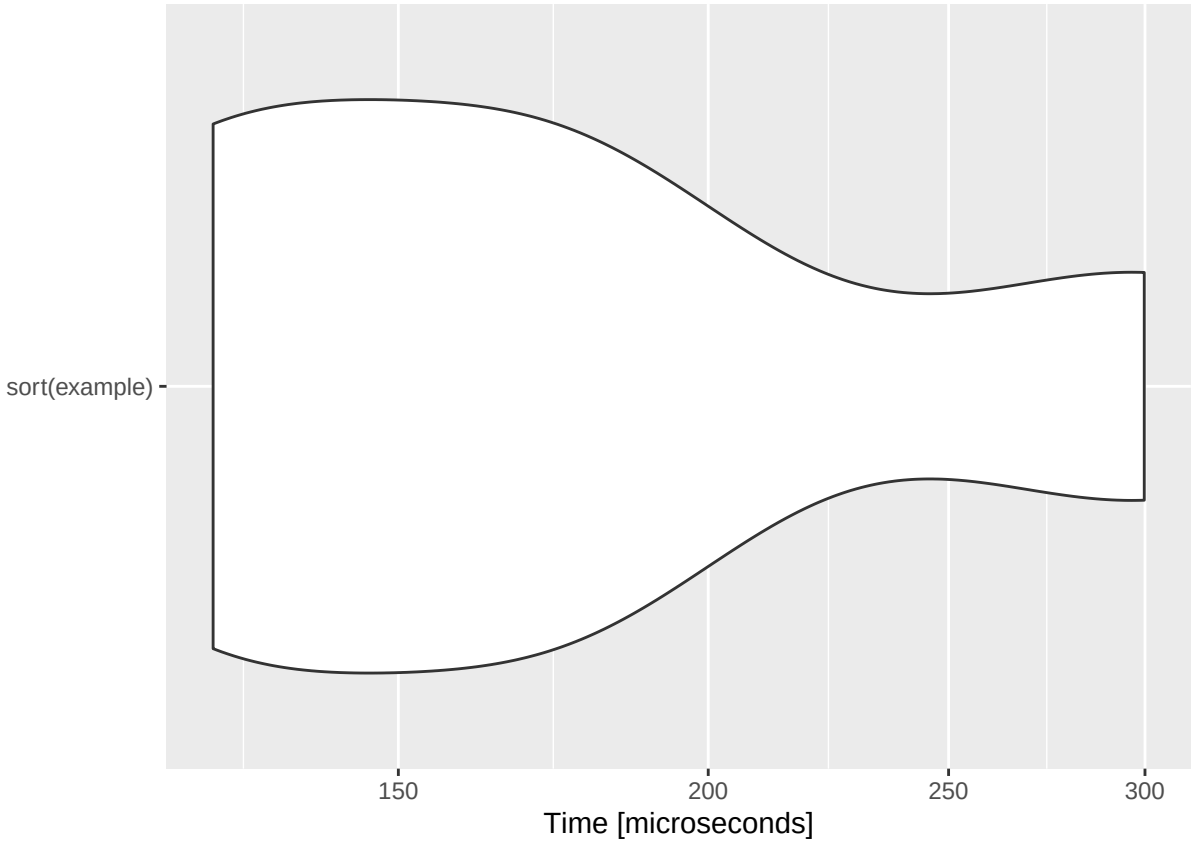

```
autoplot(microbenchmark(bubble(example),times=5))
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing or
```




```
autoplot(microbenchmark(sort(example),times=5))
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing or
```



Progresión geométrica del COVid-19 ## Modelado matemático de una epidema

```
library(readr)
location <- getwd()
setwd(location)
casos_A <- read_delim("casos.csv", ";", escape_double = FALSE, trim_ws = TRUE, skip = 1)
```

```
## Rows: 34 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr (1): Fecha
## dbl (1): Casos
## lgl (1): E_P+1
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#Estadística de casos
summary(casos_A$Casos)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   36.75  245.50  514.71  995.50 1715.00
```

```r
m <- length(casos_A$Casos)
F <- (casos_A$Casos[2:m])/(casos_A$Casos[1:m-1])
#Estadísticos de F
mean(F,na.rm = TRUE)
```

```
## [1] 1.350739
```

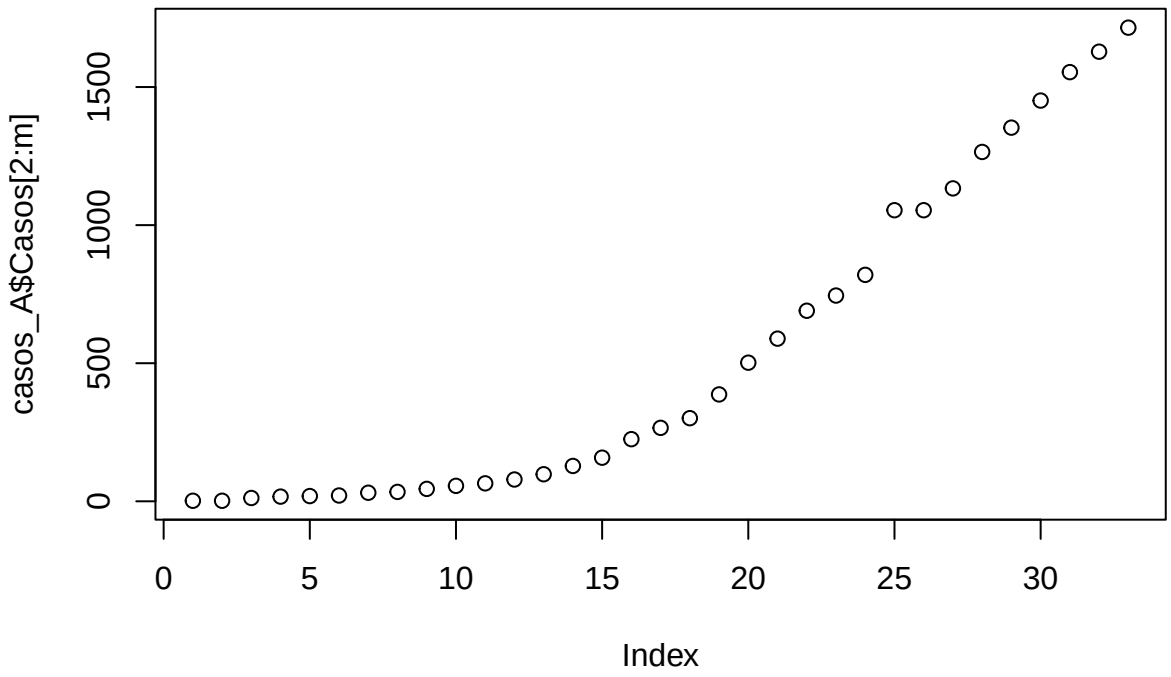
```r
sd(F,na.rm = TRUE)
```

```
## [1] 0.8554107
```

```r
var(F,na.rm = TRUE)
```

```
## [1] 0.7317275
```

```r
#Grafico de casos
plot(casos_A$Casos[2:m]);(casos_A$Casos[1:m-1])
```




```
##  [1]    1    2    2   12   17   19   21   31   34   45   56   65   79   98  128
## [16]  158  225  266  301  387  502  589  690  745  820 1054 1054 1133 1265 1353
## [31] 1451 1554 1628
```

4.2. Accediendo a los datos actualizados del Covid-19

```r
library(readr)
location <- getwd()
setwd(location)
casos_B <- read.csv("time_series_covid19_confirmed_global.csv",header = TRUE, sep = ",", dec=".")
#en nustro caso la base de dato esta muy grande, como una matriz y no un vector,
#asi no se puede utilizar la funcion which.max
# definimos aca los dos variables que necessitamos por la determincaion del modelo
epidemia=4520
fecha <- as.Date(c("01/01/22"),  format = "%m/%d/%y")
head(fecha)
```

```
## [1] "2022-01-01"
```

```r
#determinacion en que fecha se contagiaran 40 millones de personas
library(lubridate)
```

```
##
## Attachement du package : 'lubridate'
```

```
## Les objets suivants sont masqués depuis 'package:base':
##
##     date, intersect, setdiff, union
```

```r
F=1.62
while (epidemia<40000000) {
fecha=fecha+ days(1)
epidemia=epidemia*F
}
"fecha se contagiaran 40 millones de personas:"
```

```
## [1] "fecha se contagiaran 40 millones de personas:"
```

```r
head(fecha)
```

```
## [1] "2022-01-20"
```