

TyHM - Programación en R-CRan Grupo Boca Samurai

Campora Juan^{1,1,*}, Franco Bertani¹, Rubio Andres^{1,1}, Ortiz Matias^{1,1}

^a *Mendoza Argentina*

Abstract

1. Introducción

Tenemos como tarea utilizar el lenguaje de programación R para resolver problemas de la vida cotidiana a partir de métodos matemáticos, luego se revisarán y compararán distintos algoritmos para encontrar el que tenga mejor costo computacional, es decir, mejor tiempo de ejecución.

2. RESOLUCIÓN DE EJERCICIOS

2.1. Ejercicio 1: Generar un vector secuencia

2.1.1. Creación de vectores

Escribimos `c` para concatenar. Acabamos de crear un vector de 5 componentes

```
v1 <- c(1,2,3,4,5)
```

Creación de un vector de 9 componentes

```
v2 <- c(1,2,3,4,5,6,7,8,9)
```

2.1.2. Creación de una matriz de 3×3

Escribimos `ncol` para cantidad de columnas. Siempre es `byrow = TRUE` o `FALSE` para que lo acomode en orden de filas o de columnas

```
n1 <- matrix(v2, ncol = 3, byrow = TRUE)
n2 <- matrix(v2, ncol = 3, byrow = FALSE)
```

En la consola escribo “?” antes de la palabra que tenga duda

Averiguar que clase de objeto hemos creado: para saber de qué clase es un objeto se utiliza el comando “`class` (nombre del objeto)” vemos que nos dice que el vector es de tipo numérico y la matriz de tipo array

```
class(v1)
```

```
## [1] "numeric"
```

*Corresponding author

Email addresses: juaniicampora@gmail.com (Campora Juan), francobertanim@gmail.com (Franco Bertani), andresrubio@gmail.com (Rubio Andres), matiasortiz@gmail.com (Ortiz Matias)

```
class(n1)
```

```
## [1] "matrix" "array"
```

2.1.3. Creación de un vector de palabras

```
v3 <- c("a","b","c")  
class(v3)
```

```
## [1] "character"
```

```
v3
```

```
## [1] "a" "b" "c"
```

Para conocer el valor que se encuentra en la fila 1 columna 3 se escribe en la consola: `n1[1,3]`

Para decir que son todas las filas o columnas pongo un espacio entre los corchetes

Importar datos de excel o de la red

, es comma ; es semicolon

Lo que nos dice al cargar un dato de excel es que enumera todos los cambios que tuvimos que hacer en el archivo original para dejarlo ordenado y acomodado

Dataset es un conjunto de datos de una tabla www.kaggle.com

```
A<-0  
start_time<-Sys.time()  
for (i in 1:50000) { A[i] <- (i*2)}  
end_time<-Sys.time()  
end_time-start_time
```

```
## Time difference of 0.21878 secs
```

3. Ejercicio: Implementación de una serie Fibonacci

La serie Fibonacci comienza con los números 0 y 1, a partir de estos cada uno de los siguientes términos es la suma de los dos anteriores, a continuación puede verse el código para implementar la serie:

```
A <- 0  
B <- 1  
F[1] <- A  
F[2] <- B  
for (i in 3:100)  
{F[i] <- (F[i-1]+F[i-2])}  
head(F)
```

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

Posteriormente se quiere saber la cantidad de iteraciones necesarias para generar un número de la serie mayor que 1000000. Para esto vamos a eliminar la F con el fin de poder comenzar desde cero con la implementación de la serie. A continuación se puede observar el código correspondiente a la obtención de las iteraciones:

```

remove(F)
A<-0
B <- 1
C <- 0
it <- 30
F[1]<-A
F[2]<-B
for(i in 3:(2+it)) {F[i]<-(F[i-1]+F[i-2])}
C <- length(F)
message("Para ",it," iteraciones, el penúltimo valor es: \n",F[C-1],
        "\n","y el último es: \n"
        ,F[C])

```

```

## Para 30 iteraciones, el penúltimo valor es:
## 832040
## y el último es:
## 1346269

```

En conclusión se observa que se necesitan 30 iteraciones para superar el número 1000000

3.1. Ejercicio: Ordenación de un vector por método burbuja

En este apartado vamos a ordenar un vector con el método burbuja, que funciona revisando cada elemento de la lista y comparándolo con el siguiente, luego de compararlos los intercambia de posición (si están en orden equivocado), posteriormente se ordenará con el comando “SORT” de R.

```

library(tictoc)
t1<-Sys.time()
#Tomo una muestra de 10 números entre 1 y 100
x<-sample(1:100,10)
#Creo una funcion para ordenar
burbuja<-function(x){
  n<-length(x)
  for(j in 1:(n-1)){
    for(i in 1:(n-j)){
      if(x[i]>x[i+1]){
        temp<-x[i]
        x[i]<-x[i+1]
        x[i+1]<-temp
      }
    }
  }
  return(x)
}
t2 <- Sys.time()
res<-burbuja(x)
#muestra ordenada
res

```

```

## [1] 5 13 31 35 44 51 52 62 75 94

```

```
#La diferencia de tiempo es t2-t1
t2-t1
```

```
## Time difference of 0.02822018 secs
```

```
toc()
```

3.2. Ejercicio: Progresión geométrica del COVID-19

Primero descargamos la librería de “readr”, luego ingresamos los datos .csv a partir del archivo descargado de la página de la cátedra de la siguiente manera: File->Import dataset->from text(readr), luego se siguen las consignas de la guía y se copian los pasos realizados en lenguaje R para pegarlos en el siguiente código:

```
library(readr)
casos <- read_delim("D:/Users/Lenovo/Desktop/FING/Tecnicas/casos.csv",
  delim = ";", escape_double = FALSE, col_types = cols(`Covid Argentina` = col_date(format = "%m/%d/%Y"),
  Casos = col_integer()), trim_ws = TRUE,
  skip = 1)
```

```
## Warning: The following named parsers don't match the column names: Covid
## Argentina
```

```
#Estadística de casos
```

```
summary(casos$Casos)
```

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

```
m <- length(casos$Casos)
F <- (casos$Casos[2:m])/(casos$Casos[1:m-1])
F[m-1]
```

```
## [1] 1.05344
```

Podemos calcular el factor de contagios dividiendo los infectados de hoy sobre los de ayer, en el siguiente código se muestra cómo hacerlo:

```
m <- length(casos$Casos)
F <- (casos$Casos[2:m])/(casos$Casos[1:m-1])
F[m-1]
```

```
## [1] 1.05344
```

```
Este es el factor de contagios
#Estadísticos de F (factor de contagios)
```

```
mean(F, na.rm = TRUE)
```

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

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

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

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

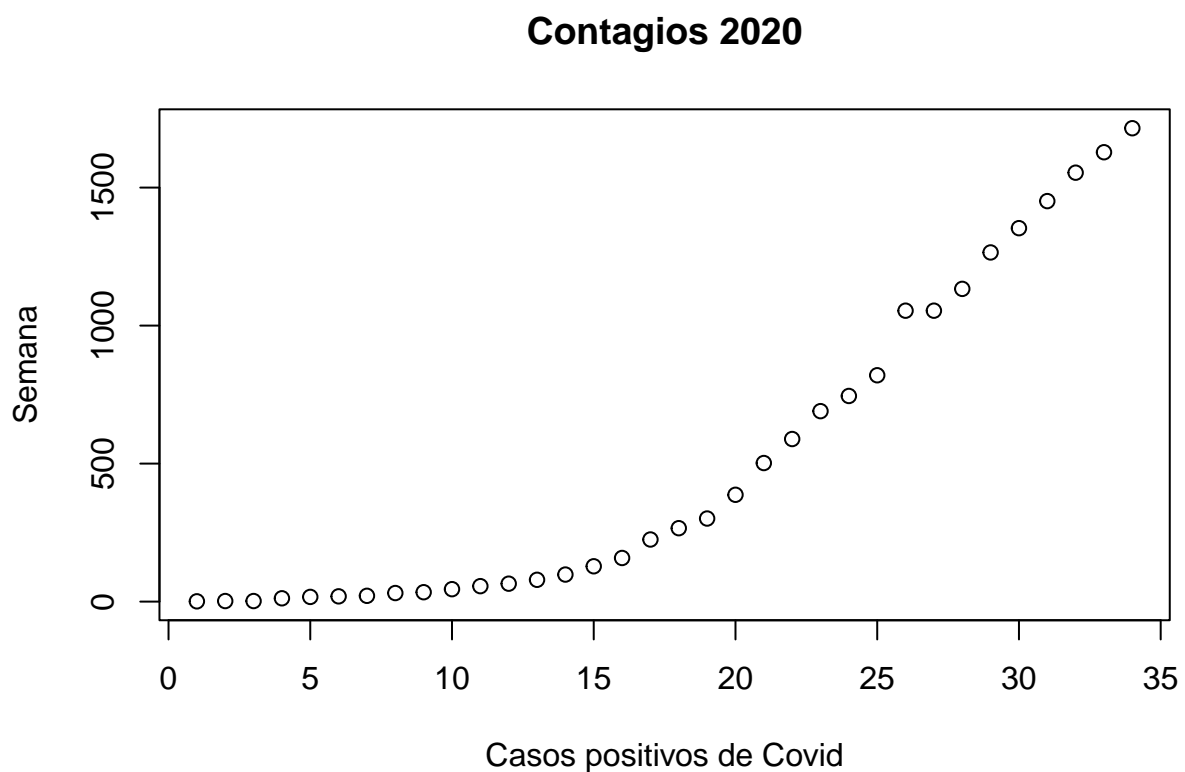
```
## [1] 0.7317275
```

#Ploteo de datos Con los datos importados anteriormente procedimos a realizar un ploteo para poder visualizarlos. A continuación se muestran los códigos implementados para los distintos gráficos.

```
casos$Casos
```

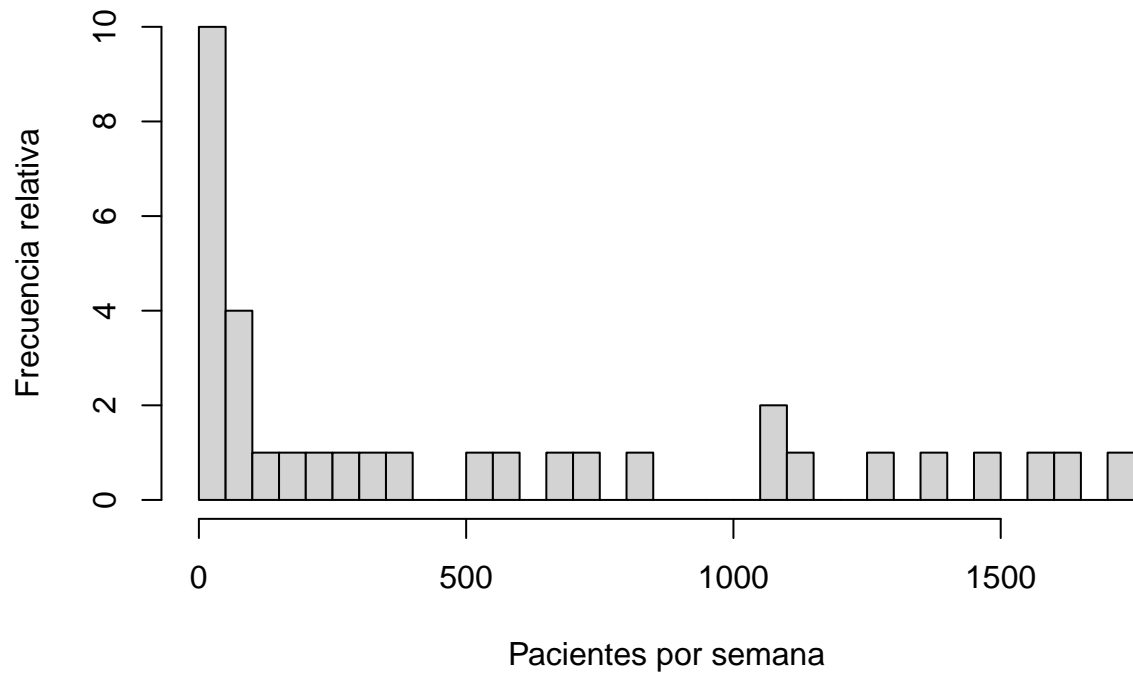
```
## [1] 1 2 2 12 17 19 21 31 34 45 56 65 79 98 128  
## [16] 158 225 266 301 387 502 589 690 745 820 1054 1054 1133 1265 1353  
## [31] 1451 1554 1628 1715
```

```
plot(casos$Casos ,main="Contagios 2020",ylab="Semana",  
      xlab="Casos positivos de Covid")
```



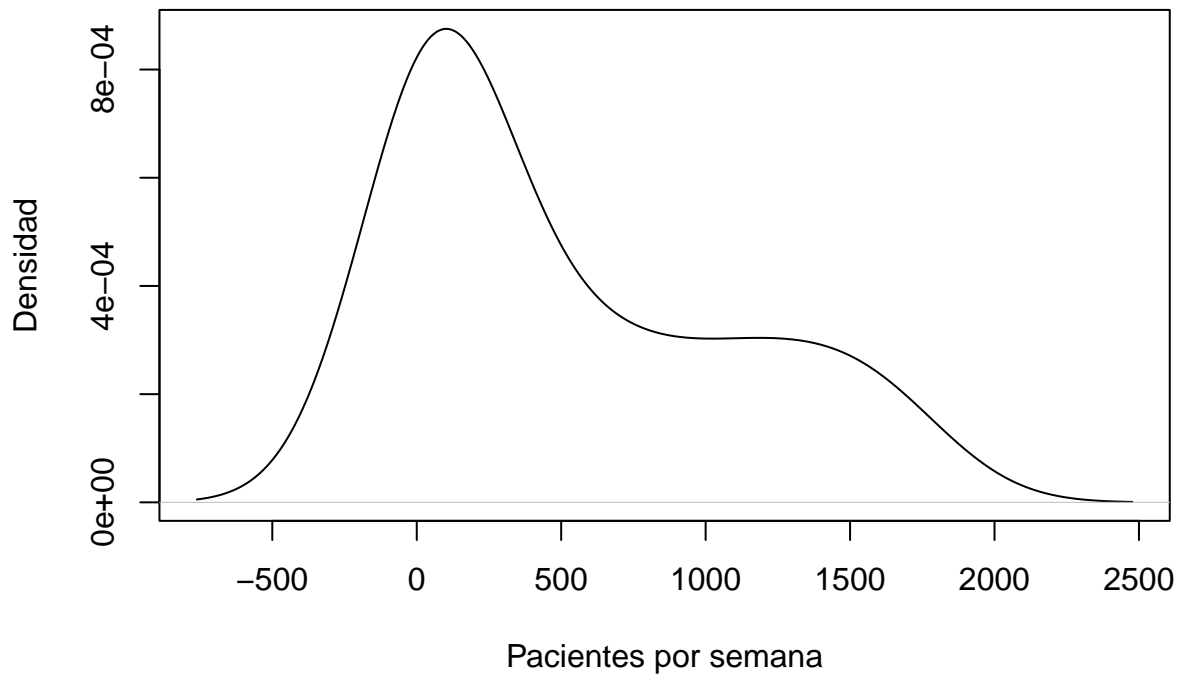
```
hist(casos$Casos,breaks=50,main = "Contagios en la Argentina",  
      xlab = "Pacientes por semana",ylab = "Frecuencia relativa")
```

Contagios en la Argentina



```
plot(density(na.omit(casos$Casos)),main="Densidad de contagios en la Argentina",  
      ylab = "Densidad", xlab = "Pacientes por semana")
```

Densidad de contagios en la Argentina



4. Conclusión

- 4.1. *R* es un entorno de software libre (licencia GNU GLP) y lenguaje de programación interpretado, es decir, ejecuta las instrucciones directamente, sin una previa compilación del programa a instrucciones en lenguaje máquina.