Elementos de programación en R-Cran

OJEDA,YESICA R. - Facultad de Ingeniería
<yesimza@gmail.com>

Universidad Nacional de Cuyo

7 de junio de 2020

Elementos básicos de programación

- Conocer y profundizar la comprensión de las funciones básicas del software RStudio.
- Utilizar algoritmos bien conocidos.
- Comparar las soluciones obtenidas utilizando algoritmos conocidos con las soluciones que brinda RStudio el cual posee sus propios algoritmos.
- Comparar el costo computacional, medido como tiempo de ejecución. Esto permitirá entender la calidad del algoritmo que se implementa.

Consignas del Trabajo

- Revisar los algoritmos que se presentan en el apartado de actividades. Primero se deberá ejecutarlos en la l<ed>nea de comando de la consola y luego hacer scripts de cada uno
- Generar un vector secuencia y tomar el tiempo para resolverlo.
- Implementación de una serie Fibonacci.
- Ordenar un vector por mótodo burbuja y sort, comparación de uso de recursos y rapidez de ambos comandos.
- Progresión geom<e9>trica del COVid-19 y generación de mapa
- Utilización de base de datos propia y base de datos externa actualizable.
- Compilar todo y presentarlo en un infome formato pdf construido con RStudio, archivo rswave y Latex

Herramientas para medir el tiempo de ejecución

La actividad propuesta requiere de medir el correspondiente tiempo de ejecución de los algoritmos ejecutados. Para llevar a cabo esta tarea R-Studio cuenta con al menos tres paquetes R que permiten comparar la perfomance de los códigos en cuesti<f3>n (rbenchmark, microbenchmark y tictoc). Estos adem<e1>s de medir el tiempo nos indican porcentaje de memoria y microprocesador utilizados.

Además, el lenguaje R proporciona al menos dos mótodos para medir el tiempo de ejecución del código R (Sys.time y system.time), que es una aproximación bastante ótil para un curso como el que desarrollamos. Todos los comandos anteriormente mencionados pueden ser consultados en R-Studio colocando en la consola ¿nombre del comando "por ejemplo ?tictoc.

Antes de comenzar se recomienda hacer una limpieza del entorno con los comandos

```
> rm(list = ls())
```

Ahora se ejecuta primero el comando tic-toc, donde "tic .[es] como dar la largada en un cronometro y "tocçuando se finaliza.

0.1. Usando tic-toc

```
> library(tictoc)
> A<-20
> tic("sleeping")
> print("dormir siesta")

[1] "dormir siesta"

> Sys.sleep(2)
> print("....suena el despertador")

[1] "....suena el despertador"

> toc()

sleeping: 2.01 sec elapsed
```

0.2. Usando sleeping

```
> #C<f3>digo para sys.sleep Verificad
> sleep_for_a_minute <- function() { Sys.sleep(14) }
> start_time <- Sys.time()
> sleep_for_a_minute()
> end_time <- Sys.time()
> end_time - start_time

Time difference of 14.02251 secs

> 'Resultado que me dio es...'

[1] "Resultado que me dio es..."
```

```
> print(end_time- start_time)

Time difference of 14.02251 secs
```

Este comando tiene como desventaja que Si se usa dentro de un documento en R-Studio se demoraró mucho tiempo cuando se compile un PDF o una presentación.

Generar un vector secuencia y medir el tiempo

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

[1]  2  4  6  8 10 12

> tail(A)

[1]  99990  99992  99994  99996  99998 100000

> end_time <- Sys.time()
> end_time - start_time

Time difference of 0.07900405 secs

> start_time <- Sys.time()
> B <- seq(1,1000000, 2)
> head (B)

[1]  1  3  5  7  9 11

> tail (B)

[1] 999989 999991 999993 999995 999997 999999

> end_time <- Sys.time()
> end_time - start_time

Time difference of 0.02300191 secs

> print(end_time- start_time)

Time difference of 0.02300191 secs
```

Ordenar de un vector por meótodo burbuja y comparación con Sort usando benchmark

```
> x<-sample(1:20000,10)
> #plot(x)
> 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)
+ }
> #Muestra obtenida
> resBurbuja<-burbuja(x)
> #Muestra Ordenada
> x<-sample(1:20000,10)
> sort(x)

 [1]  1502  3166  3822  5214  6403 10222 14967 15200 15807 16408

> respSort<-sort(x)
```

Ahora se busca comparar el uso de recursos entre burbuja y sort

```
> library(printr)
> library(ggrepel)
> library(ggplot2)
> library(microbenchmark)
> mbm1<-microbenchmark(
+   resBurbuja,respSort)
> mbm1

Unit: nanoseconds
       expr min lq  mean median uq  max neval
 resBurbuja   0  0 56.78      0  0 5673   100
   respSort   0  0  4.14      0  0  406   100

> #Mostrar resultado de mbm
> autoplot(mbm1)
```

Se debe se<f1>alar que el resultado varía todas las veces que se realiza, ya que depende de si hay otras ventanas abiertas y de la capacidad del procesador y la memoria que se posea la máquina.

Si se compara sort respecto de burbuja, se obtiene:

```
> mbm2<-microbenchmark(
+   respSort,
+   resBurbuja)
> print(mbm2)

Unit: nanoseconds
       expr min lq  mean median uq  max neval
   respSort   0  0 40.63      0  0 3647   100
 resBurbuja   0  0  0.03      0  0    1   100
```

```
> #Mostrar resultado de mbm
> autoplot(mbm2)
```

Progresión geomótrica, Fibonacci

Primero se calcula Fibonacci

```
> for(i in 0:500000)
+ { a<-i
+ b <-i+1
+ c <- a+b
+ # comentar esta l<ed>nea para conocer el n<fa>umero m<e1>s grande hallado
+ #print(c)
+ }
> print(c)

[1] 1000001
```

Luego, usando una base de datos en formato csv sobre el COVID-19 al 9 de marzo de 2020, que est<e1>n est<e1>ticas en la computadora.

Para modelar el problema se usa una progresión geomótrica

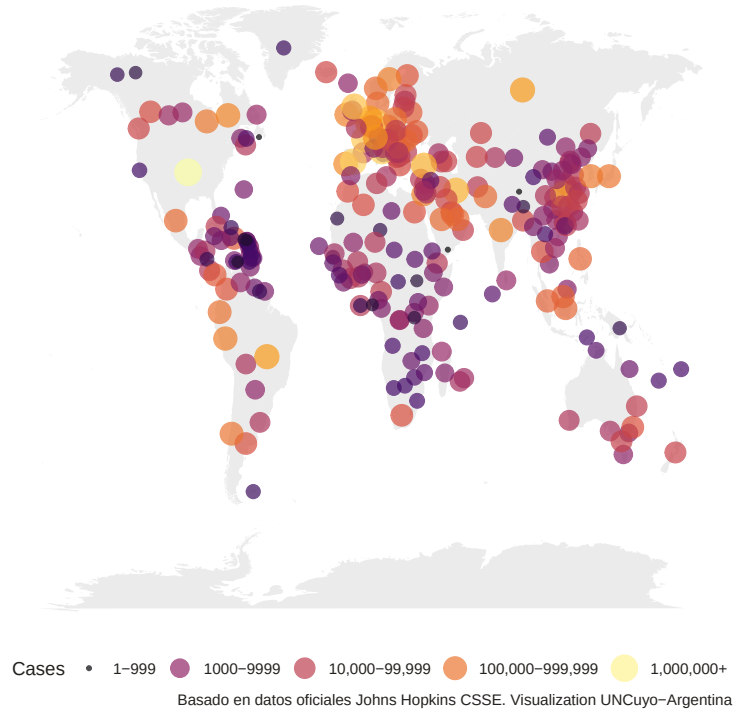
```
> library(readr)
> casos_A <- read_delim("C:/Users/YES/Desktop/TP COVID/BD COVID/casos.csv",
+                       ";", escape_double = FALSE, trim_ws = TRUE,
+                       skip = 1)
> #view(casos_A)
> summary(casos_A)

    Fecha               Casos          E_P+1
 Length:16          Min.   :  1.00   Mode:logical
 Class :character   1st Qu.: 14.50   NA's:16
 Mode  :character   Median : 31.00
                    Mean   : 40.67
                    3rd Qu.: 60.50
                    Max.   :128.00
                    NA's   :1

> m <- length(casos_A$Casos)
> F <- (casos_A$Casos[2:m])/(casos_A$Casos[1:m-1])
> mean(F,na.rm = TRUE)

[1] 1.62166

> sd(F,na.rm = TRUE)

[1] 1.283397

> var(F,na.rm = TRUE)

[1] 1.647107
```

Uso de base externa y generación de mapa con localización de casos

Se utilizá en el caso anterior y en este datos proporicionados por el hospital J. Hopkings de USA, los cuales son actualizados y de libre uso en el repositorio Github

```
> #Generación de mapa con infectados
> library(readr)
> datacovid_Jun7_global <- read_csv("https://raw.githubusercontent.com/CSSEGISandData/COVI
> #View(datacovid_Jun7_global)
> #Para saber en cu<e1>ntas regiones se subidividen los pa<ed>ses
> #paises <- summary(factor(datacovid_Jun7_global$`Country/Region`))
> #head(paises)
>
> #Calculo para el 25 de marzo
> 'La cantidad de infectados al 25 de marzo de 2020 es..'

[1] "La cantidad de infectados al 25 de marzo de 2020 es.."

> datacovid_Jun7_global[datacovid_Jun7_global$`Country/Region`=="Argentina",68]

# A tibble: 1 x 1
  `3/25/20`
      <dbl>
1       387

> library(maps)
> library(ggrepel)
> library(ggplot2)
> library(rlang)
> library(tidyverse)
> library(viridis)
> library(dplyr)
> library(maps)
> library(ggrepel)
> planisferio <- map_data("world")
> total_infec <-length(datacovid_Jun7_global)
> datas_map <- datacovid_Jun7_global[ , c(1,2,3,4,total_infec)]
> ## get the COVID-19 data
>
> # cutoffs based on the number of cases
> mybreaks <- c(1, 200, 1000, 10000, 500000)
> total_infec <-length(datacovid_Jun7_global)
> datas_map <- datacovid_Jun7_global[ , c(1,2,3,4,total_infec)]
> ggplot() +
+   geom_polygon(data = planisferio, aes(x=long, y = lat, group = group), fill="grey", alp
+   geom_point(data=datacovid_Jun7_global, aes(x=Long, y=Lat, size=`4/17/20`, color=`4/17/
+   scale_size_continuous(name="Cases", trans="log", range=c(1,7),breaks=mybreaks, labels
+
+   scale_color_viridis_c(option="inferno",name="Cases", trans="log",breaks=mybreaks, labe
```

```
+   theme_void() +
+   guides( colour = guide_legend()) +
+   labs(caption = "Basado en datos oficiales Johns Hopkins CSSE. Visualization UNCuyo-Arg
+   theme(
+     legend.position = "bottom",
+     text = element_text(color = "#22211d"),
+     plot.background = element_rect(fill = "#ffffff", color = NA),
+     panel.background = element_rect(fill = "#ffffff", color = NA),
+     legend.background = element_rect(fill = "#ffffff", color = NA)
+   )
```



Conclusión

Realizar este praáctico me permitió introducirme y profundizar el conocimiento en la utilización de RStudio y comprender más del lenguaje de programación R y pudiendo dar distintos modos de visibilidad a los resultados.

Además, puede ser un buen punto de inicio si se quiere trabajar en áreas relacionadas a Data analist, Data Scientist, entre otras.

Entre los conocimientos que adquirí se pueden nombrar:

- Generar un vector secuencia.
- Implementación de una serie Fibonacci.
- Orden de un vector por m<e9>todo burbuja y sort.
- Creación de un mapograma

- Uso de secciones y recuadros
- Uso de Rsweave para generar PDFs
- Instalar Bibliotecas
- Trabajar con archivos .csv y Datasets

Lamentablemente, no logr<e9>incorporar los gráficos en el texto principal. Si se generaron en un documento aparte y los pude visualizar en R-Studio-