Reporte en R-Markdown

Juan Diego Cubides 79998

Tomas Felipe Mendez Acosta 98417

Larry Alfredo Castro Ramos 81022

Jersson Leandro Ávila Méndez 85550

Link de GIT HUB: https://github.com/DiegoCubides

1. Números primos:

Para la solucion de este programa decidimos anidar dos for, el primero nos permite tener el conteo del 1 hasta el 100 permitiendonos recorrer todos los numeros a evaluar y el segundo for inicia desde 2 hasta x siendo x el valor en el que va el primer for. Cuando se quiere evaluar que numero es primo o no se tienen dos condiciones esenciales:

- el numero X no puede se evaluado por b=x/2, es decir que si b es mayor de la mitad de x se sabe que nos va a dar valores con decimales, para esto igualamos b a x y dejamos de evaluar el restante de numero que faltaban por recorrer el for.
- si b es diferente de x y el modulo de la division de estos dos es 0 se sabe que ya no es un numero primo.

```
library(knitr)
for (x in 1:100){
  primo <- TRUE
  for(b in 2:x){
    mitad=x/2
    if(b>mitad){
      b=x
    }
    if(b!=x & x%%b==0){
      primo <- FALSE
    }

  }
  if(primo==TRUE){
  print(x)
}
}
```

2.Uso básico del paquete Tidyverse

- 5.2.4 Exercises: items 1, and 2

```
        library(nycflights13)
        library(tidyverse)
        library(dplyr)
         nycflights13::flights
        #5.2.4 Exercises
```

```
        #1
        arrival_delays <- filter(flights,arr_delay>=120)
        #2
        vuelos_houston <- filter(flights,dest %in% c("IAH","HOU"))
```

- 5.3.1 Exercises: all items

```
#1 -´

valores_NA_al_principio <-arrange(flights,desc(is.na(dep_delay)))
#2
vuelos_retrasados <- arrange(flights,desc(dep_delay))
 vuelos_salieron_antes <-  filter(vuelos_retrasados,dep_delay<=-1)
#3
 vuelos_mas_rapidos <- arrange(flights,hour,desc(distance))
view(flights)

#4
vuelos_con_mas_distancia <- arrange(flights,desc(distance))
#vuelo con menor distancia  -> 1632
```

- 5.4.1 Exercises: items 2, 3, and 4

```
#1
select(flights,dep_time,dep_delay,arr_time,arr_delay)
filtro <- select(flights,dep_time,dep_delay,arr_time,arr_delay,everything())
select(filtro,dep_time:arr_delay)
select(flights,starts_with("de"),starts_with("arr"))
vars2 <- c("dep_time","dep_delay","arr_time","arr_delay")
select(flights,any_of(vars2))
#2
select(flights,dep_time,dep_time)  #solo la muestra 1 vez
#3
vars <- c("year", "month", "day", "dep_delay", "arr_delay")
#Este vector nos ayuda a escoger las variables que componen el vector
select(flights,any_of("year"))
# el any of escoge las varibales que se le indiquien de la tabla seleccionada
#4
select(flights, contains("DEP"))
# De esta manera se puede filtrar las variables que tengan esta palabra,
#por el tema del nombre de la funcion se logra esperar un resultado.
```

- 5.5.2 Exercises: items 1, and 2

```
#1
transmute(flights,
          dep_time,
          hour = dep_time %/% 100,
          minute = dep_time %% 100,
          sched_dep_time,
          hour_sched_dep_time = sched_dep_time %/%100,
          minute_sched_dep_time =sched_dep_time%%100
)
#2
view(transmute(flights,air_time,x=arr_time-dep_time))
#Que esperas ver? esperaba ver el mismo tiempo ya que el tiempo de vuelo
```

```
#corresponde al tiempo que pasa desde que despega hasta que aterriza
#¿Que ves? que los tiempos no coinciden y algunos estan muy desfazados
#¿Qué necesitas hacer para arreglarlo?crear una nueva variable que me compare
#la diferencia del resultado y eso restarselo a a x para que  de el mismo
#tiempo de vuelo
```

- 5.6.7 Exercises: item 1

```
not_cancelled <- flights %>%
  #se filtran los datos para que no tengan datos invalidos
  filter(!is.na(dep_delay), !is.na(arr_delay))
#15 minutos de retraso
Numberfly <- not_cancelled  %>%
  group_by(flight) %>%# se selecciona por la comluna de numero de vuuelo
  summarise(
    pos = median(arr_delay[arr_delay ==15]),
    #se utilza la funcion median para obtener el 50% y se filtra por los
    #resultados igual a 15
    neg = median(arr_delay[arr_delay ==-15])
    #se utilza la funcion median para obtener el 50%
    #y se filtra por los resultados igual a -15
  )
#Siempre llega 10 minutos
Ten <- not_cancelled %>% # se crea un nuevo grupo de datos
  group_by(flight)%>% #se agrupa po el numero de vuelo
    filter(arr_delay==-10) #se f iltra el delay por los que sean igual a 10
#30 minutos de retraso
trit <- not_cancelled  %>%
  group_by(flight) %>%
  summarise(
    pos = median(arr_delay[arr_delay ==30]),
    #se utilza la funcion median y se filtra por los resultados igual a 30
    neg = median(arr_delay[arr_delay ==-30])
    #se utilza la funcion median y se filtra por los resultados igual a -30
)
```

3.Uso básico del paquete Tidyverse

- Funcion SelelecT()

  Para este caso vamos a usar la libreria **Lahman**, primeramente vamos a cargar la libreria junto con la **tidyverse, luego vamos a dejar echo=TRUE para ver las librerias y como traemos la tabla batting. Como es una tabla tan grande se deja eval=FALSE para que no muestre el resultado del codigo.

  ```
  library(tidyverse)
  library(dplyr)
  Lahman::Batting
  ```

  la funcion select() nos permite organizar las variables de las tablas de una mejor manera, para este caso vamos a tomar solo 7 filas y 7 columnas que nos permitiran ver la tabla. dejamos eval :TRUE para uqe nos muestre que hace el codigo

  ```
  ## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
  ## v ggplot2 3.4.0     v purrr   1.0.1
  ## v tibble  3.1.8     v dplyr   1.1.0
  ## v tidyr   1.3.0     v stringr 1.5.0
  ```

```
## v readr   2.1.3     v forcats 1.0.0
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
##    playerID yearID stint teamID lgID  G  AB
## 1 abercda01   1871     1    TRO   NA  1   4
## 2  addybo01   1871     1    RC1   NA 25 118
## 3 allisar01   1871     1    CL1   NA 29 137
## 4 allisdo01   1871     1    WS3   NA 27 133
## 5 ansonca01   1871     1    RC1   NA 25 120
## 6 armstbo01   1871     1    FW1   NA 12  49
## 7 barkeal01   1871     1    RC1   NA  1   4
```

```
##    playerID yearID stint teamID
## 1 abercda01   1871     1    TRO
## 2  addybo01   1871     1    RC1
## 3 allisar01   1871     1    CL1
## 4 allisdo01   1871     1    WS3
## 5 ansonca01   1871     1    RC1
## 6 armstbo01   1871     1    FW1
## 7 barkeal01   1871     1    RC1
```

solo tomamos los datos desde playerID hasta teamID, tambien podriamos reorganizr esta tabla de la siguiente manera

```
##   CS SB RBI  playerID yearID stint teamID lgID  G  AB  R  H X2B X3B HR BB SO
## 1  0  0   0 abercda01   1871     1    TRO   NA  1   4  0  0   0   0  0  0  0
## 2  1  8  13  addybo01   1871     1    RC1   NA 25 118 30 32   6   0  0  4  0
## 3  1  3  19 allisar01   1871     1    CL1   NA 29 137 28 40   4   5  0  2  5
## 4  1  1  27 allisdo01   1871     1    WS3   NA 27 133 28 44  10   2  2  0  2
## 5  2  6  16 ansonca01   1871     1    RC1   NA 25 120 29 39  11   3  0  2  1
## 6  1  0   5 armstbo01   1871     1    FW1   NA 12  49  9 11   2   1  0  0  1
## 7  0  0   2 barkeal01   1871     1    RC1   NA  1   4  0  1   0   0  0  1  0
##   IBB HBP SH SF GIDP
## 1  NA  NA NA NA    0
## 2  NA  NA NA NA    0
## 3  NA  NA NA NA    1
## 4  NA  NA NA NA    0
## 5  NA  NA NA NA    0
## 6  NA  NA NA NA    0
## 7  NA  NA NA NA    0
```

en este caso pusimos las variables CS,SB y RBI al principio de nuestra tabla, con la funcion everything() ponemos el resto de datos faltantes

Función Arrange (). Esta función es utilizada para cambiar el orden de las filas y poder ordenarlas de acuerdo a la necesidad de datos que tengamos. Si colocamos en el arreglo varios nombres de columnas esta función ordenará la primera y luego seguirá con la segunda ordenando los datos sin desordenar la anterior y así sucesivamente con todas.

Para este ejemplo utilizaremos las librerías library(nycflights13) , library(tidyverse) y library(dplyr);

```
library(nycflights13)
  library(tidyverse)
  library(dplyr)

  arr1 <- tibble(x = c(1,3,NA,6,5,7,4,NA,2,1,4,7,7,3,2), y = c(1,1,NA,2,2,3,3,4,4,5,5,6,6,7,7)
```

```
              ,z= c("F","E","A","C","D",NA,"B","G",NA,"M","I","L","H","J","K")) %>%
  arrange(x,y,z)
```

Cabe aclarar que los valores NA siempre quedan al final, a no ser que usemos la herramienta is.na()

```
arr1 <- tibble(x = c(1,3,NA,6,5,7,4,NA,2,1,4,7,7,3,2), y = c(1,1,NA,2,2,3,3,4,4,5,5,6,6,7,7)
               ,z= c("F","E","A","C","D",NA,"B","G",NA,"M","I","L","H","J","K")) %>%
  arrange(arr1,desc(y))
```

4 Git Hub

Para el ultimo punto hicimos una funcion que recibe una variable "a", dentro de la variable tenemos varios else if anidados los cuales nos permiten evaluar todos los casos y cuando se cumpla la condicion dejara de evaluar el resto.

```
retrieve_answer <- function(a){
  if(a=="1"){
    library(nycflights13)
    library(tidyverse)
    library(dplyr)
    nycflights13::flights
    #5.2.4 Exercises
    #1
    arrival_delays <- filter(flights,arr_delay>=120)
    #2
    vuelos_houston <- filter(flights,dest %in% c("IAH","HOU"))
    #4
    salida_jun_agost_sep <- filter(flights,month %in% c(7,8,9))

  }else if(a=="2"){
    library(nycflights13)
    library(tidyverse)
    library(dplyr)
    nycflights13::flights
    #5.3.1 Ejercicios
    #1 -

    valores_NA_al_principio <-arrange(flights,desc(is.na(dep_delay)))
    #2
    vuelos_retrasados <- arrange(flights,desc(dep_delay))
    vuelos_salieron_antes <-  filter(vuelos_retrasados,dep_delay<=-1)
    #3
    vuelos_mas_rapidos <- arrange(flights,hour,desc(distance))
    #4
    vuelos_con_mas_distancia <- arrange(flights,desc(distance))
    #vuelo con menor distancia  -> 1632


  }
  else if(a=="3"){
    library(nycflights13)
    library(tidyverse)
    library(dplyr)
    nycflights13::flights
```

```
  #5.4.1
  #1
  select(flights,dep_time,dep_delay,arr_time,arr_delay)
  filtro <- select(flights,dep_time,dep_delay,arr_time,arr_delay,everything())
  select(filtro,dep_time:arr_delay)
  select(flights,starts_with("de"),starts_with("arr"))
  vars2 <- c("dep_time","dep_delay","arr_time","arr_delay")
  select(flights,any_of(vars2))
  #2
  select(flights,dep_time,dep_time)  #solo la muestra 1 vez
  #3
  vars <- c("year", "month", "day", "dep_delay", "arr_delay") #para este vector nos ayuda a escoger l
  select(flights,any_of("year")) # el any of escoge las varibales que se le indiquien de la tabla sel
  #4
  select(flights, contains("DEP"))# De esta manera se puede filtrar las variables que tengan esta pal

}
else if(a=="4"){

  library(nycflights13)
  library(tidyverse)
  library(dplyr)
  nycflights13::flights
  #5.5.2
  #1
  transmute(flights,
            dep_time,
            hour = dep_time %/% 100,
            minute = dep_time %% 100,
            sched_dep_time,
            hour_sched_dep_time = sched_dep_time %/%100,
            minute_sched_dep_time =sched_dep_time%%100
  )
  #2
  view(transmute(flights,air_time,x=arr_time-dep_time))
  #Que esperas ver? esperaba ver el mismo tiempo ya que el tiempo de vuelo corresponde al tiempo que
  #¿Que ves? que los tiempos no coinciden y algunos estan muy desfazados
  #¿Qué necesitas hacer para arreglarlo?crear una nueva variable que me compare la diferencia del res
}
else if(a=="5"){
  library(nycflights13)
  library(tidyverse)
  library(dplyr)
  nycflights13::flights
  #5.6.7
  #1
  not_cancelled <- flights %>%
    filter(!is.na(dep_delay), !is.na(arr_delay))
  #15 minutos de retraso
  Numberfly <- not_cancelled  %>%
    group_by(flight) %>%
    summarise(
      pos = median(arr_delay[arr_delay ==15]),
```

```
        neg = median(arr_delay[arr_delay ==-15])
      )
    #Siempre llega 10 minutos
    Ten <- not_cancelled %>%
      group_by(flight)%>%
      filter(arr_delay==-10)
    #30 minutos de retraso
    trit <- not_cancelled  %>%
      group_by(flight) %>%
      summarise(
        pos = median(arr_delay[arr_delay ==30]),
        neg = median(arr_delay[arr_delay ==-30])
      )


  }
  else if(a=="6"){
    print("falta ")

  }
}
print("El siguiente codigo tiene la solucion de los ejercicios propuesto para la primera entrega")
```

```
## [1] "El siguiente codigo tiene la solucion de los ejercicios propuesto para la primera entrega"
```

```
print("1-> 5.2.4 : items 1, and 2")
```

```
## [1] "1-> 5.2.4 : items 1, and 2"
```

```
print("2-> 5.3.1 : all items")
```

```
## [1] "2-> 5.3.1 : all items"
```

```
print("3-> 5.4.1 : items 2, 3, and 4")
```

```
## [1] "3-> 5.4.1 : items 2, 3, and 4"
```

```
print("4-> 5.5.2 Exercises: items 1, and 2")
```

```
## [1] "4-> 5.5.2 Exercises: items 1, and 2"
```

```
print("5-> 5.6.7 Exercises: item 1")
```

```
## [1] "5-> 5.6.7 Exercises: item 1"
```

```
print("5-> 7.1 Exercises: item 2 esta ")
```

```
## [1] "5-> 7.1 Exercises: item 2 esta "
```

```
x <- readline(prompt="Ingrese una respuesta")
```

```
## Ingrese una respuesta
```

```
retrieve_answer(x)
```