Informe Programacion en Rstudio

Santiago Avila, Andres Hernandez, Andres Montenegro, Brayan Zambrano, Nikolas Riapira

2023-02-23

INFORME - ELECTIVA AREA ELECTRONICA

1. Código básico en R

Números primos del 1 al 100

A través del siguiente código se muestran los números primos del 1 al 100

```
for (i in 2:100) {
  x <- 0
  for (j in 1:i) {
    if (i %% j == 0) {
      x <- x + 1
    } else{

    }
  }
  if (x <= 2) {
    print(i)
  }
}
```

Obteniendo como resultado lo siguiente

```
## [1] 2
## [1] 3
## [1] 5
## [1] 7
## [1] 11
## [1] 13
## [1] 17
## [1] 19
## [1] 23
## [1] 29
## [1] 31
## [1] 37
## [1] 41
## [1] 43
## [1] 47
## [1] 53
## [1] 59
## [1] 61
## [1] 67
## [1] 71
## [1] 73
```

```
## [1] 79
## [1] 83
## [1] 89
## [1] 97
```

Notese que el siguiente mensaje es debido a que el chunk de activacion de la libreria de tidyverse tiene como TRUE el valor para message, debido a esto se muestra el mensaje de carga de libreria en el documento

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.1     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   2.1.4     v forcats 1.0.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

2. Uso básico de Tidyverse

5.2.4.1 Ejercicios (filter())

A continuación se presentan los ejercicios básicos para la utilización de la función filter()

Item 1: Tuvieron un retraso de llegada de dos o mas horas

```
Arrival_Delay <- flights %>%
  filter(arr_delay >= 120)
View(Arrival_Delay)
```

Lo que genera la tabla flights unicamente con los vuelos con un retraso de llegada mayor a 120 minutos.

Item 2: Volaron a Houston

```
Houston_destiny <- flights %>%
  filter(dest == "HOU" | dest == "IAH")
view(Houston_destiny)
```

Este código muestra los vuelos con destino a Houston determinado por “HOU” o “IAH”

Item 3: Fueron operados por United, American o Delta

```
Operated_by <- flights %>%
  filter(carrier %in% c("UA", "AA", "DL"))
view(Operated_by)
```

Se filtran y muestran los vuelos correspondientes a las operacione de las 3 aerolineas mencionadas

Item 4: Vuelos que salieron en verano

```
Summer_dep <- flights %>%
  filter(month >= 7 & month <= 9)
view(Summer_dep)
```

Aca se filtran los vuelos que fueron operados en los 3 meses correspondientes a verano los cuales son Julio, Agosto y Septiembre

Item 5: Llegaron mas de dos horas tarde pero no partieron con retraso

```
LateArr_timeLeft <- flights %>%
  filter(arr_delay > 120 & dep_delay <= 0)
view(LateArr_timeLeft)
```

Se filtran dos variables diferentes siendo el retraso en la llegada superior a 2 horas y el retraso en la salida inferior a 0, es decir que salieron a tiempo o mucho antes del horario establecido

Item 6: Fueron retrasados por lo menos por una hora, pero recuperaron mas de 30min en vuelo

```
Delayed1H <- flights %>%
  filter(dep_delay >= 60 &  dep_delay - arr_delay > 30)
view(Delayed1H)
```

Se muestran los vuelos que tuvieron un retraso en la salida de al menos una hora, pero que recuperaron al menos 30min de retraso durante el vuelo, por ello se hace la resta entre ambos retrasos y deberia dar superior a los 30

Item 7: Salieron entre la medianoche y las 6 de la mañana

```
Midnight_to_6am <- flights %>%
  filter(dep_time >=0 & dep_time <= 600)
view(Midnight_to_6am)
```

Igualmente se filtran los vuelos con salidas entre la medianoche (0) y las 6am (600)

5.2.4.2 Ejercicios (filter() + between())

A continuación se presentan los ejercicios anteriores utilizando la funcion between()

Item 1: Tuvieron un retraso de llegada de dos o mas horas

```
Arrival_Delay2 <- flights %>%
  filter(between(arr_delay, 120, Inf))
View(Arrival_Delay2)
```

Item 2: Volaron a Houston

```
Houston_destiny2 <- flights %>%
  filter(between(dest, "HOU","IAH"))
view(Houston_destiny2)
```

Item 3: Fueron operados por United, American o Delta (No es posible utilizar between ya que se analizan 3 valores de la variable carrier)

```
Operated_by <- flights %>%
  filter(carrier %in% c("UA", "AA", "DL"))
view(Operated_by)
```

Item 4: Vuelos que salieron en verano

```
Summer_dep2 <- flights %>%
  filter(between(month, 7, 9))
view(Summer_dep2)
```

Item 5: Llegaron mas de dos horas tarde pero no partieron con retraso (No es posible utilizar between ya que se analizan 2 variables distintas)

```
LateArr_timeLeft2 <- flights %>%
  filter(arr_delay > 120 & dep_delay <= 0)
view(LateArr_timeLeft2)
```

Item 6: Fueron retrasados por lo menos por una hora, pero recuperaron mas de 30min en vuelo (No es posible utilizar between ya que se analizan 2 variables distintas)

```
Delayed1H2 <- flights %>%
  filter(dep_delay >= 60 & dep_delay - arr_delay > 30)
view(Delayed1H2)
```

Item 7: Salieron entre la medianoche y las 6 de la mañana

```
Midnight_to_6am2 <- flights %>%
  filter(between(dep_time, 0, 600))
view(Midnight_to_6am2)
```

5.4.1 Ejercicios (select() ; any_of())

Item 2: ¿Qué sucede si incluye el nombre de una variable varias veces en una funcion select()?

```
library(tidyverse)
library(nycflights13)
x <- flights %>%
  select(distance, distance, distance, distance, distance)
view(x)
```

Respuesta: Al ejecutar el anterior codigo se puede observar que el resultado es un data-frame donde la variable distancia aparece con todos sus valores como una unica columna, no importa cuantas veces se escriba la variable "distancia" siempre se va a obtener la columna antes mencionada sin niguna alteracion.

Item 3: ¿Que hace la funcion "any_of()"?

```
#missing variables
w <- flights %>%
  select(any_of("distance"))
view(w)

#hide the variable
w <- flights %>%
  select(-any_of("distance"))
view(w)

#3.1) Why might it be helpful in conjunction with this vector?


vars <- c("year", "month", "day", "dep_delay", "arr_delay", "carrier")

#missing variables
  z <- flights %>%
    select(any_of(vars))
  view(z)

#Hide the variables

  z <- flights %>%
    select(-any_of(vars))
  view(z)
```

Respuesta: La funcion "any_of()" en el anterior item se usa para catalogar las variables tanto las que esten dentro de ella como variables no faltantes (linea 39 de codigo), como las que esten dentro de ellas como variables faltantes con el cambio de signo en la funcion (linea 44); al correr el anterior codigo si la funcion no tiene el signo negativo las variables que esten dentro del parentesis seran las unicas que apareceran en el

data-frame resultante, de otra forma si el signo negativo es colocado las variables dentro del parentesis seran quitadas del data-frame final y apareceran el resto de variables.

3.1) ¿Por qué podría ser útil la sentecia "any_of()" junto con el siguiente vector?

```
vars <- c("year", "month", "day", "dep_delay", "arr_delay")
```

Respuesta: La sentencia que toma el valor de la variable "vars" funciona o complementa la funcion "any_of()" al hacer que esta ultima tome mas valores para hacerlos variables faltantes o para omitir en el data frame las variables que no esten en la funcion; esto conlleva que al correr el codigo del item 3 se tengan muchas mas columnas de varios tipos de variables (int, caracter etc).

Item 4: *¿Que puede concluir al ejecutar el siguiente código?*

```
#4Does the result of running the following code surprise you?
#4.1How do the select helpers deal with case by default?

 select(flights, contains("TIME"))
```

```
## # A tibble: 336,776 x 6
##    dep_time sched_dep_time arr_time sched_arr_time air_time time_hour
##       <int>          <int>    <int>          <int>    <dbl> <dttm>
##  1      517            515      830            819      227 2013-01-01 05:00:00
##  2      533            529      850            830      227 2013-01-01 05:00:00
##  3      542            540      923            850      160 2013-01-01 05:00:00
##  4      544            545     1004           1022      183 2013-01-01 05:00:00
##  5      554            600      812            837      116 2013-01-01 06:00:00
##  6      554            558      740            728      150 2013-01-01 05:00:00
##  7      555            600      913            854      158 2013-01-01 06:00:00
##  8      557            600      709            723       53 2013-01-01 06:00:00
##  9      557            600      838            846      140 2013-01-01 06:00:00
## 10      558            600      753            745      138 2013-01-01 06:00:00
## # ... with 336,766 more rows
```

Respuesta: A simple vista puede parecer que la linea de codigo necesita mas especificaciones para funcionar pero se concluye que de por si la funcion "contains()" es bastante util ya que abarca bastantes variables que tengan una relacion entre ellas y como se ve posteriormente en el item 4.2 la forma de obtener el mismo resultado que ocurre con el anterior codigo es mucho menos efectiva.

4.1)¿Cómo tratan los "select helpers" el caso de forma predeterminada?

Respuesta: En esta forma predeterminada la sentencia "contains("TIME")" es muy util ya que afecta el funcionamiento al hacer que se muestren solo las columnas cuyas variables tengan en el nombre o cadena de caracteres "time", como curiosidad parece que no afecta si se usan las mayusculas y de paso esta sentencia funcionaria como un filtro mas efectivo de las variables.

4.2) ¿Cómo puede cambiar ese valor predeterminado?

```
#4.2 How can you change that default?

  #solution with any_of

  library(tidyverse)
  library(nycflights13)
  vars <- c("dep_time", "sched_dep_time", "arr_time", "sched_arr_time", "air_time","time_hour")
  a <- flights %>%
    select(any_of(vars))
  view(a)
```

Respuesta: Se interpreto que usando la sentencia de "any_of()" podria darse un resultado igual al que se obtiene con "contains()" aunque si bien cumple con el objetivo el codigo de la parte superior, como se dijo anteriormente este no seria muy eficaz si se tiene un dataset con muchas columnas que necesiten ser filtradas; situacion que por otro lado con la sentencia "contains()" solo tomaria una linea de codigo.

```
  library(readr)
  library(dplyr)
  library(tidyverse)

  myData <- nycflights13::flights
```

5.6.7 Ejercicios

1) Haga una lluvia de ideas sobre al menos 5 formas diferentes de evaluar las características típicas de retraso de un grupo de vuelos. Considerelos siguientes escenarios.

Item 1: Un vuelo llega 15 minutos antes el 50% del tiempo y 15 minutos tarde el 50% del tiempo.

-Vuelos con llegada 15 minutos antes

```
library(knitr)
ANTES <- myData%>%
  filter (arr_delay== (-15))
#kable(head(ANTES))
```

-Vuelos con llegada 15 minutos despues

```
DESPUES <- myData%>%
  filter (arr_delay== 15)
```

-tiempo de vuelo de 30 minutos

```
MITAD_TIEMPO <- myData%>%
  filter (air_time<=30)
```

-combinacion 15 minutos antes de llegada con 30 minutos de vuelo

```
ANTES_MITAD_TIEMPO<- myData%>%
  select( carrier,flight, tailnum, origin , dest,arr_delay,air_time)%>%
  filter (arr_delay== (-15),air_time==30)
kable(head(ANTES_MITAD_TIEMPO))
```

carrier	flight	tailnum	origin	dest	arr_delay	air_time
EV	4309	N12922	EWR	ALB	-15	30
US	2126	N957UW	LGA	BOS	-15	30
9E	2946	N8672A	JFK	PHL	-15	30
EV	3807	N14568	EWR	PVD	-15	30
US	1911	N955UW	LGA	PHL	-15	30

-combinacion 15 minutos despues de llegada con 30 minutos de vuelo

```
DESPUES_MITAD_TIEMPO<- myData%>%
  select( carrier,flight, tailnum, origin , dest,arr_delay,air_time)%>%
  filter (arr_delay== (15),air_time==30)
kable(head(DESPUES_MITAD_TIEMPO))
```

carrier	flight	tailnum	origin	dest	arr_delay	air_time
EV	4404	N17196	EWR	PVD	15	30
EV	4404	N14905	EWR	PVD	15	30
EV	4125	N14204	EWR	ALB	15	30
US	2055	N713UW	LGA	PHL	15	30

Item 2: Un vuelo siempre llega 10 minutos tarde.

```
SIEMPRE_TARDE <- myData%>%
  filter (arr_delay== 10)
```

Item 3: Un vuelo llega 30 minutos antes el 50% del tiempo y 30 minutos tarde el 50% del tiempo.

-Vuelos con llegada 30 minutos antes

```
ANTES_30 <- myData%>%
  filter (arr_delay== (-30))
```

-Vuelos con llegada 30 minutos despues

```
DESPUES_30 <- myData%>%
  filter (arr_delay== 30)
```

-tiempo de vuelo de 60 minutos

```
MITAD_TIEMPO_60 <- myData%>%
  filter (air_time<=60)
```

-combinacion 30 minutos antes de llegada con 60 minutos de vuelo

```
ANTES_MITAD_30_TIEMPO_60<- myData%>%
  select( carrier,flight, tailnum, origin , dest,arr_delay,air_time)%>%
  filter (arr_delay== (-30),air_time==60)
  kable(head(ANTES_MITAD_30_TIEMPO_60))
```

carrier	flight	tailnum	origin	dest	arr_delay	air_time
EV	5378	N716EV	LGA	PIT	-30	60
9E	3320	N935XJ	JFK	BUF	-30	60
DL	1235	N315NB	LGA	PIT	-30	60
DL	1129	N301NB	LGA	PIT	-30	60
9E	3455	N919XJ	JFK	PIT	-30	60
B6	6	N304JB	JFK	BUF	-30	60

-combinacion 30 minutos despues de llegada con 60 minutos de vuelo

```
DESPUES_MITAD_30_TIEMPO_60<- myData%>%
  select( carrier,flight, tailnum, origin , dest,arr_delay,air_time)%>%
  filter (arr_delay== (30),air_time==60)
  kable(head(DESPUES_MITAD_30_TIEMPO_60))
```

carrier	flight	tailnum	origin	dest	arr_delay	air_time
9E	3846	N8894A	JFK	ORF	30	60
9E	2906	N604LR	JFK	BUF	30	60
MQ	4418	N852MQ	JFK	DCA	30	60

carrier	flight	tailnum	origin	dest	arr_delay	air_time
UA	1638	N36247	EWR	CLE	30	60

Item 4: El 99% de las veces un vuelo es puntual. El 1% de las veces llega 2 horas tarde.

```
-vuelos puntuales aproximadamente el 99% del total menos a 120 min
```

```
PUNTUAL <- myData %>%
  filter(arr_delay< 120)
```

-vuelos que llegan 2 horas o mas tarde aproximadamente El 1% ddel total

```
DOS_HORAS_TARDE <- myData %>%
  filter(arr_delay>= 120)
```

¿Qué es más importante: el retraso en la llegada o el retraso en la salida?

Consideramos que si sabemos la hora de salida del vuelo y hay un retraso de llegada según la hora programada puede generar incertidumbre ya que pudo haber pasado algo en el vuelo o en la logística de trasporte aéreo. Si embargo un retraso en la salida genera molestias para el usuario, pero no hay inseguridad para él, por tal motivo es más importante el retraso en la llegada que en la salida en termino de seguridad para quien usan el servicio y sus servidores.

5.7.1 Ejercicio

2) ¿Qué avión (tailnum) tiene el peor récord de puntualidad?

```
RECORD_INPUNTUAL <- myData %>%
select( carrier,flight, tailnum, origin , dest,arr_delay)%>%
  filter(arr_delay>1200)
kable(head(RECORD_INPUNTUAL))
```

carrier	flight	tailnum	origin	dest	arr_delay
HA	51	N384HA	JFK	HNL	1272

Reporting with Rmarkdown

Function: Mutate ()

La función Mutate permite crear, modificar y eliminar variables existentes en un conjunto de datos, un ejemplo básico de su uso es el siguiente:

ConjuntoN <- **mutate**(**ConjuntoDD**, **Nvariable** = **Calculo**)

Donde **"ConjuntoN"** es el conjunto al cual llegaran la modificación, **"ConjuntoDD"** es el conjunto de datos de donde se toman originalmente, **"Nvariable"** es el nombre asignado a la variable creada y por último **"Cálculo"** las expresiones que van a definir como se obtiene la variable a crear.

A continuación se muestra el código al crear un dataframe básico con tres columnas, los primeros 5 números pares, los primeros 5 números impares y los primeros 5 números primos.

```
Numeros_pares <-  c(2, 4, 6, 8, 10)
Numeros_impares <- c(1, 3, 5, 7, 9)
Numeros_primos <- c(2, 3, 5, 7, 11)

Numeros <- data.frame(Numeros_pares,
```

```
                      Numeros_impares,
                      Numeros_primos)
```

Lo que genera el siguiente dataframe

```
##   Numeros_pares Numeros_impares Numeros_primos
## 1             2               1              2
## 2             4               3              3
## 3             6               5              5
## 4             8               7              7
## 5            10               9             11
```

Para modificar a través de mutate se realiza la suma de el valor 2 a cada uno de los números pares, para crear a través de mutate se se genera una nueva columna donde se hace la multiplicación de las filas entre ambas columnas del dataframe Números y adicional a ello, para eliminar a través de mutate, se remueve la columna de los números primos.

```
Numeros <- Numeros %>%
  mutate(
    Numeros_pares = Numeros_pares + 2, #Modificación
    MultParesImpares = Numeros_pares * Numeros_impares, #Creación
    Numeros_primos = NULL #Eliminación
  )
```

Donde como resultado se obtiene los nuevos numeros pares en la misma columna, una nueva columna en el dataframe Números con el resultado de la multiplicación y se observa que se eliminó la columna de los números primos

```
##   Numeros_pares Numeros_impares MultParesImpares
## 1             4               1                4
## 2             6               3               18
## 3             8               5               40
## 4            10               7               70
## 5            12               9              108
```

Function: Select ()

La funcion select() como su nombre lo indica funciona para distinguir una variable por su nombre o reasignarla a otra previamente declarada para, en este caso usarla en mas operaciones como lo son el any_of() o simplemente para mostrar esa variable independientemente del resto que se encuentra en el dataset correspondiente.

Ejemplo:

```
example <- starwars %>%
  select(name, height, mass, skin_color)
head(example)
```

```
## # A tibble: 6 x 4
##   name           height  mass skin_color
##   <chr>           <int> <dbl> <chr>
## 1 Luke Skywalker    172    77 fair
## 2 C-3PO             167    75 gold
## 3 R2-D2              96    32 white, blue
## 4 Darth Vader       202   136 white
## 5 Leia Organa       150    49 light
## 6 Owen Lars         178   120 light
```

Segun el anterior ejemplo se tiene el dataset de nombre “starwars” el cual se le asigna la variable de nombre

“example” para posteriormente visualizar los datos de nombre, altura, masa y color de piel; en este caso solo se busca separa variables o distinguir estas del resto y visualizar las cuatro columnas en una grafica diferente.

Respecto al chunk “error”, puede decirse que al haber un error debajo de el este se visualiza en la parte inferior del codigo en una linea en caso de estar en true y del chunk “echo” si este esta en true lo que este en ese espacio (codigo) no se vera en el posterior archivo previsualizado.