Exploration and Application of R for Data Science

Alejandro Pachón, Santiago Meza, Alexander Morgan

2023-02-25

GitHub

Puedes visitar nuestro repositorio en internet, para más información: **Nuestro Repositorio**

Numeros primos

En este codigo, la linea for (x in 1:100){, identifica con la variable "x", desde el valor 1 al 100. Luego, se declara "Nprimo" como TRUE, para que en cuyo caso de que el numero a valorar sea primo, el codigo posteriormente lo imprima sin necesidad de volver a preguntar el numero almacenado, despues se abre un nuevo 'for' en donde se encuentran dos "if", para que luego de realizar la operación, identifique si es o no un numero primo. Conociendo que si la división es menor al valor que este en el "for N", tomara la variable N el valor de x para salir del "for" y continuar a imprimir el numero. De lo contrario, si el modulo es diferente de "0", Nprimo pasara a ser falso y no imprimira el numero en la consola. Y asi consecutivamente hasta llegar al número 100.

```
#numeros primos
for (x in 1:100){
  Nprimo <- TRUE
  for(N in 2:x){
    mid=x/2
    if(N>mid){
      N=x
    }
    if(N!=x & x%%N==0){
      Nprimo <- FALSE
    }
  }
  if(Nprimo==TRUE & x!=1){
    print(x)
  }
}
```

[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 ## 2. Uso basico de la libreria Tidyverse

los ejercicios propuestos en el documento se tomaron del apartado **Data transformation**, de la pagina *R for Data Science*.

5.2.4: 1. Encuentra todos los vuelos que:

- **Item 1:** Tuvieron un retraso de llegada de dos o más horas.

Primero identificamos la variable de retraso de llegada, en la base de datos es `arr_delay`, tambien identificamos que los datos que nececitamos son un conjunto de filas, por esto se toma la funcion `filter()` y se agrega la

condicion que nececitamos para tomar los vuelos que tuvieron un retraso de llegada de dos horas o mas. De esta forma tendriamos que:

```
        library(nycflights13)
        library(dplyr)
        filter(flights,arr_delay>119)
```

```
## # A tibble: 10,200 x 19
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      811        630     101    1047     830     137 MQ
##  2  2013     1     1      848       1835     853    1001    1950     851 MQ
##  3  2013     1     1      957        733     144    1056     853     123 UA
##  4  2013     1     1     1114        900     134    1447    1222     145 UA
##  5  2013     1     1     1505       1310     115    1638    1431     127 EV
##  6  2013     1     1     1525       1340     105    1831    1626     125 B6
##  7  2013     1     1     1549       1445      64    1912    1656     136 EV
##  8  2013     1     1     1558       1359     119    1718    1515     123 EV
##  9  2013     1     1     1732       1630      62    2028    1825     123 EV
## 10  2013     1     1     1803       1620     103    2008    1750     138 MQ
## # ... with 10,190 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

- **item 2:** Voló a Houston (IAH o HOU).

Se identifica la variable que indique el destino de los vuelos y esta se le da la condicion a la cual queremos filtrar en este caso se define con 2 nombres diferentes al destino del vuelo, por lo tanto en la condicion de `filter()` debe ser consideraro para que tome ambas nomenclaturas, para esto se agrega la operacion logica `|`. De esta forma tendriamos que:

```
        library(nycflights13)
        library(dplyr)
        filter(flights, dest=="IAH"|dest=="HOU" )
```

```
## # A tibble: 9,313 x 19
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      533        529       4     850     830      20 UA
##  3  2013     1     1      623        627      -4     933     932       1 UA
##  4  2013     1     1      728        732      -4    1041    1038       3 UA
##  5  2013     1     1      739        739       0    1104    1038      26 UA
##  6  2013     1     1      908        908       0    1228    1219       9 UA
##  7  2013     1     1     1028       1026       2    1350    1339      11 UA
##  8  2013     1     1     1044       1045      -1    1352    1351       1 UA
##  9  2013     1     1     1114        900     134    1447    1222     145 UA
## 10  2013     1     1     1205       1200       5    1503    1505      -2 UA
## # ... with 9,303 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

- **Item 3:** Fueron operados por United, American o Delta.

Lo primero es identificar las siglas de las aereolineas que nos dan en este caso UA, AA y DL respectivamente, despues tomamos la variable `carrier` y asignamos las siglas medinte una funcion `filter()`, de esta manera tendremos los vuelos operados por dichas aereolineas, y como en el caso anterior se agrega la operacion logica `|` Para cumplir la condicion de que se filtren cualquiera de las 3 aereolineas. De esta forma tendriamos que:

```
       library(nycflights13)
       library(dplyr)
 filter(flights, carrier=="AA"|carrier=="DL"|carrier=="UA")
```

```
## # A tibble: 139,504 x 19
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      533        529       4     850     830      20 UA
##  3  2013     1     1      542        540       2     923     850      33 AA
##  4  2013     1     1      554        600      -6     812     837     -25 DL
##  5  2013     1     1      554        558      -4     740     728      12 UA
##  6  2013     1     1      558        600      -2     753     745       8 AA
##  7  2013     1     1      558        600      -2     924     917       7 UA
##  8  2013     1     1      558        600      -2     923     937     -14 UA
##  9  2013     1     1      559        600      -1     941     910      31 AA
## 10  2013     1     1      559        600      -1     854     902      -8 UA
## # ... with 139,494 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

- **Item 4**: Volo en verano (julio, agosto y septiembre).

Para la imprementacion este item se identifica que los meses estan definidos en la variable `month` y que los meses son almacenados por el orden numerico de estos, por esto se identofican los meses de verano como 7,8 y 9 correspondientes a julio, agosto y septiembre respectivamente, para la sintaxis de la condicion se usa la funcion `filter()` y la funcion `%in%`, de esta manera mediante `%in%` se utiliza para verificar si los elementos de un cojunto de datos estan presentes. De esta forma tendriamos que:

```
library(nycflights13)
library(dplyr)
filter(flights,month%in%c(7,8,9))
```

```
## # A tibble: 86,326 x 19
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     7     1        1       2029     212     236    2359     157 B6
##  2  2013     7     1        2       2359       3     344     344       0 B6
##  3  2013     7     1       29       2245     104     151       1     110 B6
##  4  2013     7     1       43       2130     193     322      14     188 B6
##  5  2013     7     1       44       2150     174     300     100     120 AA
##  6  2013     7     1       46       2051     235     304    2358     186 B6
##  7  2013     7     1       48       2001     287     308    2305     243 VX
##  8  2013     7     1       58       2155     183     335      43     172 B6
##  9  2013     7     1      100       2146     194     327      30     177 B6
## 10  2013     7     1      100       2245     135     337     135     122 B6
## # ... with 86,316 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
```

```
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

- **Item 5**: LLego mas de dos horas tarde, pero no se retraso.

Para este punto se toma la condicion del Item 1 ya que se mencionan nuevamente los buelos con mas de 2 horas de retraso en llegar, y para la segunda condicion se identifica la variable `dep_delay` como la que toma los datos de los vuelos retrsados, y se agrega el operador logico `&` para combinar las condiciones y filtar los vuelos que cumplan con las condiciones. De esta forma tendriamos que:

```
library(nycflights13)
library(dplyr)
filter(flights, arr_delay>119&dep_delay==0)
```

```
## # A tibble: 3 x 19
##    year month   day dep_time sched_dep~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##   <int> <int> <int>    <int>       <int>   <dbl>   <int>   <int>   <dbl> <chr>
## 1  2013    10     7     1350        1350       0    1736    1526     130 EV
## 2  2013     5    23     1810        1810       0    2208    2000     128 MQ
## 3  2013     7     1      905         905       0    1443    1223     140 DL
## # ... with 9 more variables: flight <int>, tailnum <chr>, origin <chr>,
## #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>, and abbreviated variable names 1: sched_dep_time,
## #   2: dep_delay, 3: arr_time, 4: sched_arr_time, 5: arr_delay
```

- **Item 6**: Se retrasaron al menos una hora, pero estuvieron más de 30 minutos en vuelo.

Con la logica planteada en el Item 5, cambiamos las variables para cumplir con las condiciones, en este caso solo se nececitan los vuelos con un retraso de 1 hora, pero que tambien volaron por mas de 30 minutos, para esto usamos la variable `air_time` y se asigna un tiempo menor o igual 30 minutos. De esta forma tendriamos que:

```
library(nycflights13)
library(dplyr)
  filter(flights, arr_delay<=60 & air_time<=30)
```

```
## # A tibble: 1,194 x 19
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1     1318       1322      -4    1358    1416     -18 EV
##  2  2013     1     1     2000       2000       0    2054    2110     -16 9E
##  3  2013     1     1     2116       2110       6    2202    2212     -10 EV
##  4  2013     1     1     2302       2200      62    2342    2253      49 EV
##  5  2013     1     2      602        600       2     646     659     -13 US
##  6  2013     1     2      743        745      -2     858     857       1 9E
##  7  2013     1     2     1335       1322      13    1414    1416      -2 EV
##  8  2013     1     2     1606       1610      -4    1730    1729       1 9E
##  9  2013     1     2     2003       2015     -12    2102    2125     -23 9E
## 10  2013     1     2     2125       2110      15    2221    2212       9 EV
## # ... with 1,184 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

- **Item 7**: Salio entre la medianoche y las 6 am.

Con la funcion `filter()`se definen un rango de valores entre las 0 (media noche) y 6 (6 de la mañana) e

incluyendo los limites del rango, aplicamos esta logica a la variable `hour`, asi tendriamos los vuelos entre media noche y las 6 am.

```
library(nycflights13)
library(dplyr)
  filter(flights, hour>=0 & hour<=6)
```

```
## # A tibble: 27,905 x 19
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      533        529       4     850     830      20 UA
##  3  2013     1     1      542        540       2     923     850      33 AA
##  4  2013     1     1      544        545      -1    1004    1022     -18 B6
##  5  2013     1     1      554        600      -6     812     837     -25 DL
##  6  2013     1     1      554        558      -4     740     728      12 UA
##  7  2013     1     1      555        600      -5     913     854      19 B6
##  8  2013     1     1      557        600      -3     709     723     -14 EV
##  9  2013     1     1      557        600      -3     838     846      -8 B6
## 10  2013     1     1      558        600      -2     753     745       8 AA
## # ... with 27,895 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

5.2.4: 2.Otro ayudante de filtrado de dplyr útil es between(). ¿Qué hace? ¿Puede usarse para simplificar los códigos del anterior punto? la funcion `between()` realiza un rango de valores partiendo de una variable, de esta forma puede cambiar la estructura de varios ejercicios vistos anteriormente, un ejemplo utilizando el Item 7 del punto anterior seria:

```
library(nycflights13)
library(dplyr)

filter(flights, between(hour, 0,6))
```

```
## # A tibble: 27,905 x 19
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      533        529       4     850     830      20 UA
##  3  2013     1     1      542        540       2     923     850      33 AA
##  4  2013     1     1      544        545      -1    1004    1022     -18 B6
##  5  2013     1     1      554        600      -6     812     837     -25 DL
##  6  2013     1     1      554        558      -4     740     728      12 UA
##  7  2013     1     1      555        600      -5     913     854      19 B6
##  8  2013     1     1      557        600      -3     709     723     -14 EV
##  9  2013     1     1      557        600      -3     838     846      -8 B6
## 10  2013     1     1      558        600      -2     753     745       8 AA
## # ... with 27,895 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

donde como se ve