

Taller primer corte

Andrés Hernández código: 9563, Andrés Leonardo Mendoza Chunza código: 86204,
David Eduardo Martinez Agudelo código: 81639, Jaime Alberto Gonzalez Vargas
código: 33236, Juan Sebastian Rodriguez Padilla código: 92699

2023-02-27

NÚMEROS PRIMOS

El siguiente código identifica y muestra los números primos del 1 al 100.

```
modulo<-0
impresion<-F
controlImpresion<-T
for(números in 3:100){
  for(denominador in 2:50){
    modulo<-números%%denominador
    if(modulo!=0){
      impresion<-T
    }
    else{
      impresion<-F
      break
    }
    if(denominador==(números-1)){
      break
    }
  }

  if(controlImpresion==T){
    print(1)
    print(2)
    controlImpresion<-F
  }
  if(impresion==T){
    print(números)
  }
}

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

Primero creamos las variables

Modulo: Es tipo numérica, la utilizamos para identificar si el número solo es divisible entre el mismo y el número 1.

Impresión: Es tipo boolean, permite imprimir el número solo si es primo.

controlImpresion: Es tipo boolean, permite imprimir solo una vez los números 1 y 2.

```
modulo<-0
impresion<-F
controlImpresion<-T
```

Tenemos dos ciclos FOR anidados el principal incrementamos la variable números desde el número 3 hasta el número 100 y el FOR anidado se encarga de recorrer la variable denominador desde el número 2 hasta número el 50.

Dentro del ciclo For anidado realizamos la operación modulo entre las variables numeros y denominador esta operación nos permite saber si el número es divisible por un número diferente de el mismo y uno, si el número es divisible por un número diferente nos da como resultado 0 y si el número es divisible por el mismo y por uno nos entrega un 1.

Luego tenemos la estructura de control IF ELSE con la variable modulo si modulo es 1 la variable impresión es T y permite imprimir el número que estamos analizando pero si en alguno de los ciclos de la variable denominador nos da que el modulo es 0 la variable impresión ahora es F y con la instrucción break rompemos el ciclo anidado para que no siga contando y así disminuimos los tiempos de ejecución.

También tenemos una estructura IF que permite sacar al ciclo FOR anidado cuando la variable denominador alcanza el valor de la variable numero menos uno.

```

for(números in 3:100){
  for(denominador in 2:50){
    modulo<-números%%denominador
    if(modulo!=0){
      impresion<-T
    }
    else{
      impresion<-F
      break
    }
    if(denominador==(números-1)){
      break
    }
  }
}

```

Por ultimo tenemos la parte de la impresión esta ubicada en el FOR principal en el primer IF preguntamos si la variable controlImpresion es T si se cumple esta condición imprime los números 1 y 2 luego iguala la variable controlImpresion a F lo que causa que se bloquee y no vuelva a entrar en esa condición mientras se ejecuta el programa, luego tenemos el IF que permite imprimir los números primos si la variable impresión es T y se ejecuta con cada iteración del ciclo FOR principal.

```

if(controlImpresion==T){
  print(1)
  print(2)
  controlImpresion<-F
}

## [1] 1
## [1] 2

if(impresion==T){
  print(numeros)
}

```

EJERCICIOS DE TRANSFORMACION

Punto 5.2.4

Instalamos las librerías nycflights13 que nos permite acceder a la información de los vuelos y tidyverse que permite analizar la información.

```

library(nycflights13)
library(tidyverse)

```

1. Encuentra los vuelos que cumplan las condiciones.

Creamos una variable llamada P1 y le asignamos los valores del dataset flights y realizamos los filtros correspondientes.

```

P1<-flights
P1<-flights%>%
  filter(arr_delay>=2                                #Pto 1, ud. hora
        ,dest=="HOU"|dest=="IAH"                    #Pto 2,
        ,carrier %in% c("UA","DL","AA")             #Pto 3,*
        ,month>=7&month<=9                           #Pto 4,
        ,dep_delay<=0,arr_time>=120                 #Pto 5, ud. minutos
        ,dep_time>=60,air_time>=30                  #Pto 6, ud. minutos *
        ,hour>=0, hour<=6)                           #Pto 7,

```

Y obtenemos como resultado 17 objetos

```

## # A tibble: 17 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     3     531        536        -5      810      806
4 UA
## 2  2013     7     3     542        545        -3      823      813
10 UA
## 3  2013     7    21     626        630        -4      928      915
13 UA
## 4  2013     7    22     545        545         0      826      813
13 UA
## 5  2013     7    22     636        638        -2      921      911
10 UA
## 6  2013     8     1     538        545        -7      832      813
19 UA
## 7  2013     8     5     542        545        -3      836      813
23 UA
## 8  2013     8    10     630        635        -5      921      908
13 UA
## 9  2013     8    13     627        631        -4      913      905
8 UA
## 10 2013     8    14     542        545        -3      821      816
5 UA
## 11 2013     8    16     540        545        -5      817      813
4 UA
## 12 2013     8    16     627        631        -4      907      905
2 UA
## 13 2013     8    17     544        545        -1      819      815
4 UA
## 14 2013     8    24     627        631        -4      919      904
15 UA
## 15 2013     9    20     526        530        -4      819      810
9 UA
## 16 2013     9    25     509        517        -8      803      757
6 UA
## 17 2013     9    25     535        545       -10      829      827

```

2 UA

```
## # ... with 9 more variables: flight <int>, tailnum <chr>, origin <chr>,
## #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute
## #   <dbl>,
## #   time_hour <dtm>, and abbreviated variable names 1sched_dep_time,
## #   2dep_delay, 3arr_time, 4sched_arr_time, 5arr_delay
```

#	year	month	day	dep_time	sched_dep_time	dep_delay	air_time	sched_arr_time	arr_delay	carrier	flight	tailnum	origin	dest	air_time	distance	hour	minute	time_hour
1	2013	7	6	013	010	-3	010	010	0	UA	113	N25208	ORD	IAH	011	1400	8	00	2013-07-06 08:00
2	2013	7	6	042	040	-2	020	013	-10	UA	479	N578UA	LGA	IAH	004	1416	5	45	2013-07-06 05:00
3	2013	7	21	010	010	-4	020	010	-11	UA	1101	N12123	ORD	IAH	008	1400	6	30	2013-07-21 06:00
4	2013	7	22	040	040	-2	020	010	-11	UA	078	N21108	LGA	IAH	002	1416	8	40	2013-07-22 08:00
5	2013	7	22	036	038	-2	021	011	-10	UA	250	N507UA	ORD	IAH	000	1400	6	30	2013-07-22 06:00
6	2013	8	1	010	005	-5	022	013	-19	UA	088	N528UA	LGA	IAH	000	1416	8	00	2013-08-01 08:00
7	2013	8	5	042	040	-2	020	011	-23	UA	479	N578UA	LGA	IAH	000	1416	5	45	2013-08-05 05:00
8	2013	8	10	010	010	-5	021	010	-11	UA	300	N505UA	ORD	IAH	001	1400	6	30	2013-08-10 06:00
9	2013	8	13	017	015	-2	019	009	-8	UA	107	N418UA	LGA	IAH	000	1416	6	31	2013-08-13 06:00
10	2013	8	14	042	040	-2	021	010	-8	UA	300	N505UA	LGA	IAH	000	1416	5	45	2013-08-14 05:00
11	2013	8	18	040	040	-8	017	010	-8	UA	080	N518UA	LGA	IAH	000	1416	8	40	2013-08-18 08:00
12	2013	8	18	017	015	-2	017	000	-17	UA	224	N405UA	LGA	IAH	001	1416	6	31	2013-08-18 06:00
13	2013	8	17	044	040	-4	010	010	-8	UA	377	N547UA	LGA	IAH	001	1416	5	45	2013-08-17 05:00
14	2013	8	24	017	015	-2	010	004	-13	UA	210	N408UA	LGA	IAH	001	1416	6	31	2013-08-24 06:00
15	2013	8	20	010	010	-8	010	010	-8	UA	080	N513UA	ORD	IAH	000	1400	5	30	2013-08-20 05:00
16	2013	8	20	009	017	-8	005	011	-8	UA	262	N578UA	ORD	IAH	001	1400	5	17	2013-08-20 05:00
17	2013	8	20	010	010	-10	010	017	-7	UA	1714	N27108	LGA	IAH	001	1416	8	40	2013-08-20 08:00

2. ¿Para que sirve between()?

Es una función que permite filtrar valores en un rango específico la estructura que maneja es la siguiente `x >= left & x <= right` se usa `between(x, left, right)` utilizando la estructura podemos identificar que el límite inferior lo ubicamos en la left y el límite superior en right.

Information tomada de la documentation de R con el menu de ayuda.

¿Puedes usarlo para simplificar el código necesario para responder a los desafíos anteriores?

Si. podemos usar `between()` en los puntos 4 y 7 donde tenemos variables del tipo numérica que debemos filtrar en un rango de valores.

```
library(nycflights13)
library(tidyverse)
P2<-flights
P2<-flights%>%
  filter(arr_delay>=2, #Pto 1, ud. hora
         ,dest=="HOU"|dest=="IAH" #Pto 2
         ,carrier %in% c("UA","DL","AA") #Pto 3 **
         ,between(month,7,9) #Pto 4
         ,dep_delay<=0,arr_time>=120 #Pto 5 ud. minutos
         ,dep_time>=60,air_time>=30 #Pto 6 ud. minutos
         ,between(hour,0,6)) #Pto 7,
```

PUNTO 5.3.1

1. ¿Cómo podrías usar `arrange()` para ordenar todos los valores faltantes desde el principio?

```
library(nycflights13)
library(tidyverse)
P4.1<-arrange(flights,desc(is.na(dep_time)),dep_time)
```

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier
1	2013	1	1	N/A	1630	N/A	N/A	1815	N/A	EV
2	2013	1	1	N/A	1935	N/A	N/A	2240	N/A	AA
3	2013	1	1	N/A	1500	N/A	N/A	1825	N/A	AA
4	2013	1	1	N/A	600	N/A	N/A	901	N/A	B6
5	2013	1	2	N/A	1540	N/A	N/A	1747	N/A	EV
6	2013	1	2	N/A	1620	N/A	N/A	1746	N/A	EV
7	2013	1	2	N/A	1355	N/A	N/A	1459	N/A	EV
8	2013	1	2	N/A	1420	N/A	N/A	1644	N/A	EV
9	2013	1	2	N/A	1321	N/A	N/A	1536	N/A	EV

2. Ordenar flights para encontrar los vuelos más retrasados y los vuelos que salieron antes.

Vuelos más retrasados

```
library(nycflights13)
library(tidyverse)
P4.2<-arrange(flights,desc(dep_delay))
```

Relacion de los vuelos más retrasados

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier
1	2013	1	9	641	900	1301	1242	1530	1272	HA
2	2013	6	15	1432	1935	1137	1607	2120	1127	MQ
3	2013	1	10	1121	1635	1126	1239	1810	1109	MQ
4	2013	9	20	1139	1845	1014	1457	2210	1007	AA
5	2013	7	22	845	1600	1005	1044	1815	989	MQ
6	2013	4	10	1100	1900	960	1342	2211	931	DL
7	2013	3	17	2321	810	911	135	1020	915	DL
8	2013	6	27	959	1900	899	1236	2226	850	DL
9	2013	7	22	2257	759	898	121	1026	895	DL

Vuelos que salieron antes

```
library(nycflights13)
library(tidyverse)
P4.3<-arrange(flights,dep_delay)
```

Relacion de los vuelos que salieron antes.

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier
1	2013	12	7	2040	2123	-43	40	2352	48	B6
2	2013	2	3	2022	2055	-33	2240	2338	-58	DL
3	2013	11	10	1406	1440	-32	1549	1559	-10	EV
4	2013	1	11	1900	1930	-30	2233	2243	-10	DL
5	2013	1	29	1703	1730	-27	1947	1957	-10	F9
6	2013	8	9	729	755	-26	1002	955	7	MQ
7	2013	10	23	1907	1932	-25	2143	2143	0	EV
8	2013	3	30	2030	2055	-25	2213	2250	-37	MQ
9	2013	3	2	1431	1455	-24	1601	1631	-30	9E

3. Ordenar flights para encontrar los vuelos más rápidos (velocidad más alta).

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

```
P4.4<-head(arrange(flights,desc(distance/air_time)))
```

Vuelos más rápidos

	arr_delay	carrier	flight	tailnum	origin	dest	air_time	distance	hour	minute	time_hour
37	-14	DL	1499	N566DN	LGA	ATL	65	762	17	0	2013-05-25 17:00:00
19	26	EV	4667	N17196	EWB	MSP	93	1008	15	13	2013-07-02 15:00:00
26	-1	EV	4292	N1456B	EWB	GSP	55	594	20	25	2013-05-13 20:00:00
13	2	EV	3805	N12567	EWB	BNA	70	748	19	10	2013-03-23 19:00:00
17	-28	DL	1902	N956DL	LGA	PBI	105	1035	16	0	2013-01-12 16:00:00
30	-51	DL	315	N3768	JFK	SJU	170	1598	6	55	2013-11-17 06:00:00

4. ¿Qué vuelos viajaron más lejos?

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

```
P4.5<-arrange(flights, desc(distance))
```

Vuelos que viajaron más lejos

	arr_delay	carrier	flight	tailnum	origin	dest	air_time	distance	hour	minute	time_hour
	-14	HA	51	N380HA	JFK	HNL	659	4983	9	0	2013-01-01 09:00:00
	-5	HA	51	N380HA	JFK	HNL	638	4983	9	0	2013-01-02 09:00:00
	-26	HA	51	N380HA	JFK	HNL	616	4983	9	0	2013-01-03 09:00:00
	-14	HA	51	N384HA	JFK	HNL	639	4983	9	0	2013-01-04 09:00:00
	-11	HA	51	N381HA	JFK	HNL	635	4983	9	0	2013-01-05 09:00:00
	28	HA	51	N385HA	JFK	HNL	611	4983	9	0	2013-01-06 09:00:00
	50	HA	51	N385HA	JFK	HNL	612	4983	9	0	2013-01-07 09:00:00
	-26	HA	51	N389HA	JFK	HNL	645	4983	9	0	2013-01-08 09:00:00
	1272	HA	51	N384HA	JFK	HNL	640	4983	9	0	2013-01-09 09:00:00

¿Cuál viajó menos?

```
library(nycflights13)
library(tidyverse)
P4.6<-arrange(flights,distance)
```

Vuelos que viajaron menos

	arr_delay	carrier	flight	tailnum	origin	dest	air_time	distance	hour	minute	time_hour
	NA	US	1632	NA	EWB	LGA	NA	17	1	6	2013-07-27 01:00:00
	-2	EV	3833	N13989	EWB	PHL	30	80	21	29	2013-01-03 21:00:00
	27	EV	4193	N14972	EWB	PHL	30	80	12	0	2013-01-04 12:00:00
	136	EV	4502	N15983	EWB	PHL	28	80	16	15	2013-01-04 16:00:00
	-6	EV	4645	N27962	EWB	PHL	32	80	21	29	2013-01-04 21:00:00
	-25	EV	4193	N14902	EWB	PHL	29	80	12	0	2013-01-05 12:00:00
	0	EV	4619	N22909	EWB	PHL	22	80	21	29	2013-01-06 21:00:00
	-12	EV	4619	N33182	EWB	PHL	25	80	21	29	2013-01-07 21:00:00
	39	EV	4619	N11194	EWB	PHL	30	80	21	30	2013-01-08 21:00:00

PUNTO 5.4.1

2.¿Qué sucede si incluye el nombre de una variable varias veces en una llamada select()?

Si se incluye el nombre de la variable varias veces en select solo selecciona una vez la variable como lo podemos observar en la tabla. Information tomada de la documentation de R con el menu de ayuda.

```
library(knitr)
P3<-flights%>%
  select(day,day,day)
kable(head(P3))
```

day

1

1

1

1

1

1

1

3. ¿Que función tiene any_of()?

any_of se utiliza para eliminar variables de un vector.

4. ¿El resultado de ejecutar el código te sorprende?

Lo sorprendente de la función “contains” es que tiene la posibilidad de interpretar los datos ingresados independientemente de si son con letras mayusculas o minusculas, en este caso particular selecciona todas las columnas que contengan la palabra “time” asi se ingrese con letras mayusculas.

¿cómo tratan los ayudantes seletcotos el caso de forma predeterminada?

Programando de manera que todas las variables se nombren con letras minusculas y evitando combinaciones.

¿Cómo se puede cambiar ese valor predeterminado?

Para poder corregir este valor predeterminado se puede agregar una condicion a la sentencia ingresada ignore.case=FALSE. de esta manera solo mostrara los valores que cumplan las dos condiciones.

```
select(flights,contains("TIME,ignore.case=FALSE"))

## # A tibble: 336,776 × 0
```

PUNTO 5.5.2

1. Actualmente dep_time y sched_dep_time son convenientes a la vista, pero difíciles de calcular porque en realidad no son números continuos. Conviértalos a una representación más conveniente de la cantidad de minutos desde la medianoche.

```
library(nycflights13)
library(tidyverse)
flights_times <- mutate(flights, dep_time_mins = (
  dep_time %% 100*60 + dep_time %% 100) %% 1440, sched_dep_time_mins = (
  sched_dep_time %% 100 * 60 + sched_dep_time %% 100) %% 1440)

select(flights_times, dep_time, dep_time_mins, sched_dep_time,
  sched_dep_time_mins)

## # A tibble: 336,776 × 4
##   dep_time dep_time_mins sched_dep_time sched_dep_time_mins
##   <int>      <dbl>         <int>         <dbl>
## 1      517          317           515           915
## 2      533          333           529           329
## 3      542          342           540          1000
## 4      544          344           545          1305
## 5      554          354           600            0
## 6      554          354           558           658
## 7      555          355           600            0
## 8      557          357           600            0
## 9      557          357           600            0
## 10     558          358           600            0
## # ... with 336,766 more rows
```

2. Comparar air_time con arr_time - dep_time.

Comparacion

```
library(nycflights13)
library(tidyverse)
#> Comparacion
flights_airtime <- mutate(flights,
  dep_time = (dep_time %% 100 * 60 + dep_time %% 100)
  %% 1440,
  arr_time = (arr_time %% 100 * 60 + arr_time %% 100) %%
  1440,
  air_time_diff = air_time - arr_time +
  dep_time)
nrow(filter(flights_airtime, air_time_diff != 0))

## [1] 327346
```

¿Que esperas ver?

```
{r}{r echo=TRUE,message=FALSE, warning=FALSE} library(nycflights13)
library(tidyverse) ggplot(flights_airtime, aes(x = air_time_diff)) +
  geom_histogram(binwidth = 1)
```

¿Que ves?

¿Qué necesitas hacer para arreglarlo?

```
```r}{r echo=TRUE,message=FALSE, warning=FALSE} library(nycflights13)
library(tidyverse) ggplot(filter(flights_airtime, dest == "LAX"), aes(x = air_time_diff)) +
geom_histogram(binwidth = 1)
```

**\*\*PUNTO 5.6.7\*\***

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

1. Analicis de los tiempos de mantenimiento de los aviones para garantizar la disponibilidad.
2. Sincronizar los tiempos de carrier para la salida a plataforma.
3. Garantizar que el taxeo se sincronice con el rodaje a pista para tener disponibilidad de rampa.
4. Es muy comun la demora o retraso de un vuelo cuando se encuentran FOD en pista (son materiales o piezas desprendidas) que podrian ocasionar un accidente.
5. Incluir un tiempo de limpieza de pista y almacenar los tiempos para tener una trazabilidad de estos tiempos.

¿Que es mas importante el retraso en la llegada o en la salida?

Se debe considerar desde que posicion se contempla, para la aerolinea es mas beneficioso un retraso en la llegada ya que da tiempo a trabajo en pista para planeacion de despegue de otras rutas evitando la congestion de flota, pero para un pasajero puede representar mayores problemas y costos una llegada tarde esto porque se puede alterar cronogramas de viaje y posibles sobre costos.

**\*\*PUNTO 5.7.1\*\***

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

```
```r
library(knitr)
avion<-flights %>%
  filter(!is.na(arr_delay)) %>%
  group_by(tailnum)%>%
  summarise(arr_delay = mean(arr_delay), n = n()) %>%
```

tailnum	arr_delay	n
N203FR	59.12195	41

tailnum	arr_delay	n
N203FR	59.12195	41

[illegible]

y_e_a_r	m_o_n_d_h_y	d_e_p_t_m_e	sche_d_de_p_ti_me	de_p_del_ay	ar_r_t_im_e	sche_d_ar_r_ti_me	ar_r_de_la_y	c_ar_r_i_er	fl_i_g_h_t	tai_l_n_u_m	o_r_i_g_i_n	d_e_s_t	ai_r_ti_m_e	di_st_a_n_c_e	h_o_u_r	m_i_n_u_t_e	ti_me_h_ou_r
																	0:00
2013	1	1	N A	1935	NA	N A	2240	N A	791	N 3E H A	L G A	D F W	N A	1389	19	35	2013-01-19 0:00
2013	1	1	N A	1500	NA	N A	1825	N A	1925	N 3E V A	L G A	M I A	N A	1096	15	0	2013-01-15 0:00
2013	1	1	N A	600	NA	N A	901	N B 6	125	N 618J B	J F K	F L L	N A	1069	6	0	2013-01-06 0:00
2013	1	2	N A	1540	NA	N A	1747	N E V	4352	N 10575	E W R	C V G	N A	569	15	40	2013-02-15

m			de					ar			fl			ai			ti	
y	o	d	p_	sche	de	ar	sche	r_	c	i	tai	o	d	r_	di	h	m	me
e	n	a	ti	d_de	p_	r_t	d_ar	de	ar	g	ln	ri	e	ti	st	o	in	_h
a	t	a	m	p_ti	del	im	r_ti	la	ri	h	u	gi	s	m	an	u	ut	ou
r	h	y	e	me	ay	e	me	y	er	t	m	n	t	e	ce	r	e	r
																		:0
																		0:
																		00
2	1	2	N	162	NA	N	174	N	E	4	N	E	P	N	31	1	2	20
0			A	0		A	6	A	V	4	13	W	I	A	9	6	0	13
1										0	94	R	T					-
3										6	9							01
																		-
																		02
																		16
																		:0
																		0:
																		00

Ejercicio tomado del punto 5.7.1 PUNTO 2. ¿Qué avión (tailnum) tiene el peor récord de puntualidad? vamos a generar el documento con el comando echo=FALSE.

```
{r echo=FALSE,message=FALSE, warning=FALSE} library(nycflights13)
library(tidyverse) library(knitr) avion<-flights %>% filter(!is.na(arr_delay)) %>%
group_by(tailnum)%>% summarise(arr_delay = mean(arr_delay), n = n()) %>%
filter(n >= 20) %>% filter(min_rank(desc(arr_delay)) == 1) kable(head(avion))
```

Como podemos observar solo nos muestra el resultado del código.

tailnum	arr_delay	n
N203FR	59.12195	41

Ahora cambiamos echo=TRUE y al generar el documento final podemos visualizar el código y el resultado del programa.

```
library(nycflights13)
library(tidyverse)
library(knitr)
avion<-flights %>%
  filter(!is.na(arr_delay)) %>%
  group_by(tailnum)%>%
  summarise(arr_delay = mean(arr_delay), n = n()) %>%
  filter(n >= 20) %>%
  filter(min_rank(desc(arr_delay)) == 1)
kable(head(avion))
```

tailnum	arr_delay	n
---------	-----------	---

tailnum	arr_delay	n
N203FR	59.12195	41
...		

GitHub, - R packages

Enlace de GitHub <https://github.com/AFH-15/PrimerCorte/tree/main/R>