Ejercicio Electiva

Raul Pinilla

2023-08-28

Analizar base de datos flights

En este taller utilizamos la libreria tidyverse

```
library(tidyverse)
```

5.2.4 item 1 Mostraremos los vuelos que llegaron con dos o mas horas de retrazo

- Para esto usamos la base de datos nycflights13 el cual lo asignamos a la variable "vuelos"
- Mediante la funcion filter, le pediremos al sistema que nos muestre unicamente los vuelos que llegaron con dos horas o mas de retraso (>=), mediante el encabezado "arr_delay"

```
vuelos <- nycflights13 ::flights
vuelos_con_retraso <- filter (vuelos, arr_delay >= "2",
                              na.rm=TRUE)
```

5.2.4 item 2 Mostraremos los vuelos que llegaron a Houston en el aeropuerto HOU y en IAH

- Ahora nuevamente con "filter" pero en el encabezado de "dest" le pedimos que nos muestre unicamente los que llegaron a los aeropuertos HOU y IAH y aca sera util el == para que unicamente muestre las casillas con esos valores
- na.rm=TRUE nos permite quitar las casillas con NA (sin informacion)

```
vue_dest_hou <- filter (vuelos, dest == "HOU" | dest == "IAH"
                        ,na.rm=TRUE)
```

5.3.1 item 1 Organizar los vuelos de manera descendente iniciando con los que no tienen informacion (NA)

Con la funcion arrange permite orgnanizar, ahora organizamos los vuelos de manera descentende con "desc" y agregamos "is.na" para incluir estos valores

```
organizar <- vuelos%>%
  arrange (desc(is.na(dep_time)))
```

5.3.1 item 2 Organizar los vuelos con mayor retraso

- Creamos una variable "vue_conmayorretraso" y traemos la informacion de la base vuelos
- Recuerda usar "desc" para mostrar desde el que mas tiempo de retraso tiene

```
vue_conmayorretraso <- vuelos%>%
  arrange (desc(dep_delay))
```

5.3.1 item 2 Organizar vuelos adelantados o con menor retraso

- Creamos una variable "vue_adel" y traemos la informacion de la base vuelos
- Cuando solo colamos "arrange" el automaticamente organizará en orden ascendente

```
vue_adel <- vuelos%>%
  arrange (dep_delay)
```

5.3.1 item 3 Organizar los vuelos rapidos, gastaron menos tiempo

- Ahora mediante los encabezados "hour" y "minute"
- Para escoger mas de un encabezado solo basta con agregar una coma y colocar el siguiente encabezado

```
vue_rapid <- vuelos%>%
  arrange (hour, minute)
```

5.3.1 item 4 Organizar los vuelos con un trayecto mayor

- Creamos una variable "vue_lejos" y traemos la informacion de la base vuelos
- Para organizar (arrange)los vuelos con mayor recorrido usamos el encabezado "distance" y agregamos "desc" para que nos organice de manera descendente

```
vue_lejos <- vuelos%>%
  arrange (desc(distance))
```

5.3.1 item 4 vuelos mas cercanos

- Creamos una variable "vue_cerca" y traemos la informacion de la base vuelos
- Para organizar (arrange)los vuelos con mayor recorrido usamos el encabezado "distance" y omitimos "desc" para que nos organice de manera ascendente

```
vue_cerca <- vuelos%>%
  arrange (distance)
```

5.4.1 item 2 llamar una variable varias veces la misma variable

- Si llamo varias veces la misma variable no cambia en nada, muestra unicamente esa variable, una sola vez.
- Con la funcion "select" nos ayuda a escoger una variable en especifico que queremos ver

```
call_var <-  vuelos%>%
  select(origin, origin, origin)
```

5.4.1 item 3 Funcion any_of

- Le indicamos al motor de expresiones que asocie solo unas variables.
- Coloque las variables que desea mostrar, en este caso "year", "month", "day", "dep_delay", "arr_delay"

```
rp <-  vuelos%>%
select(any_of(c("year", "month", "day", "dep_delay", "arr_delay")))
```

5.4.1 item 4 ¿Te sorprende el resultado de ejecutar el siguiente código? ¿Cómo manejan el caso los ayudantes seleccionados de forma predeterminada? ¿Cómo se puede cambiar ese valor predeterminado?

- Los datos mostrados son de variables que incluyen la palabra TIME gracias a la funcion de "contains"
- Si cambio por otra palabra en comun me mostrara unicamente las variables que tengan esta palabra
- Select nos permite escoger una variable en especifico, en este caso unas que tengan la palabra "TIME"

```
rp1 <-  vuelos%>%
select(vuelos, contains("TIME"))
```

5.5.2 item 1 Mostrar datos de dep_time y sched_dep_time mas parecido a una hora

- Creamos una variable "hora_vuelos" y traemos la infomacion de "vuelos"
- Con select, escogemos los encabezados "dep_time" y "sched_dep_time"
- Creamos una nueva variable "minutos_deptime" y le damos el valor de "dep_time" dividido entre 100 y luego, multiplicado por 60 para tener los numeros desde las 00:00

```
hora_vuelos <- vuelos %>%
  select(dep_time, sched_dep_time)%>%
  mutate(minutos_deptime = (dep_time/100*60), minutos_sched = (sched_dep_time/100*60))
```

#5.5.2 item 2 Comparar air_timecon arr_time - dep_time. Que esperas ver? ¿Que ves? ¿Qué necesitas hacer para solucionarlo?

- Asignamos una variable "comparar" y traemos la infomacion de "vuelos"
- Creamos una nueva variable ("air_t") con "mutate"
- Seleccionamos (select) las variables "air_t", "air_time", "everything"
- Creamos una nueva variable ("arr_t") con "mutate"
- Seleccionamos (select) las variables "arr_t" y "everything"

```
comparar <- vuelos %>%
  mutate(air_t = air_time /100) %>%
  select(air_t, air_time, everything()) %>%
  mutate(arr_t = arr_time/100) %>%
  select(arr_t, everything())
```

5.6.7 Un vuelo llega 15 minutos antes el 50% del tiempo y 15 minutos tarde el 50% del tiempo

- Asignamos una variable "no_cancelados" y traemos la infomacion de "vuelos"
- Mediante "filter" y con ayuda de "!is.na" le pedimos a la base de datos que nos muestre todos los valores menos "air_time"

```
no_cancelados <- vuelos %>%
  filter(!is.na(air_time))
```

Definimos la llegada como la mas importante

- Agrupamos con la funcion "group_by" la variable "tailnum"
- Creamos variables nueva con "mutate" "median_arr_delay" y "median_dep_delay"
- Las variables las igualamos a la mediana de los encabezados "arr_delay" y "dep_delay" respectivamente
- Filtramos con "filter" valos mayor a 30

- “Arrange” para organizar las variables “median_arr_delay”, “median_dep_delay”

```
no_cancelados %>%
  group_by(tailnum) %>%
  mutate(
    count = n(),
    median_arr_delay = median(arr_delay),
    median_dep_delay = median(dep_delay)
  ) %>%
  filter(count > 30) %>%
  arrange(median_arr_delay, median_dep_delay)
```

De la tabla observamos que N479AA no el avión llega antes de los 20 minutos el 50% del tiempo, sale antes de los 3 minutos el 50% del tiempo.

```
no_cancelados %>%
  group_by(tailnum) %>%
  summarise(
    count = n(),
    p_15_early_arr = mean(arr_delay < -15),
    p_15_dep_arr = mean(dep_delay < -15)
  ) %>%
  filter(p_15_early_arr > 0.5 | p_15_dep_arr > 0.5) %>%
  filter(count > 30) %>%
  arrange(desc(p_15_early_arr), desc(p_15_dep_arr))
```

Estos vuelos realizaron al menos 30 vuelos en un año y llegaron o salieron antes de los 15 minutos durante más del 50% del tiempo. La llegada es más importante.

Vuelos con mas de 10 minutos de retrazo

- Agrupamos con “group_by” las variables tailnum, origin, dest
- Con “mean” nos muestra la media aritmetica y con “sum” se realiza la suma de los argumentos

```
no_cancelados %>%
  group_by(tailnum, origin, dest) %>%
  summarise(
    count = n(),
    arr_delay_10_c = sum(arr_delay > 10),
    arr_delay_10_p = mean(arr_delay > 10),
    dep_delay_10_c = sum(dep_delay > 10),
    dep_delay_10_p = mean(dep_delay > 10)
  ) %>%
  filter(count > 20) %>%
  arrange(desc(arr_delay_10_p))
```

Vuelos con exactamente 10 minutos de probabilidad de tardanza

```r
no_cancelados %>%
  group_by(tailnum) %>%
  summarise(
    count = n(),
    exact_10 = mean(arr_delay == 10)
  ) %>%
  filter(count > 10) %>%
  arrange(desc(exact_10))
```

Seleccionamos Seattle para un enfoque de 30 minutos, mejores vuelos con menos de 30 minutos

```r
no_cancelados %>%
  group_by(tailnum) %>%
  mutate(
    count = n(),
    arr_30_early = mean(arr_delay < -30),
    dep_30_early = mean(dep_delay < -30),
    arr_30_late  = mean(arr_delay > 30),
    dep_30_late = mean(dep_delay > 30)
  ) %>%
  filter(count > 20) %>%
  arrange(desc(arr_30_early), desc(dep_30_early), arr_30_late, dep_30_late) %>%
  select(dest)
```

5.7.1 Calcular el retraso máximo de llegada y la proporción de puntualidad.

```r
no_cancelados  %>%
  group_by(tailnum) %>%
  summarise(
    count = n(),
    max_arr_delay = max(arr_delay),
    is_on_time_freq = mean(arr_delay <= 0, na.rm = TRUE)
  ) %>%
  filter(count > 30) %>%
  arrange(desc(is_on_time_freq))
```