Activity

Maicol Vargas

2023-08-25

Start

In practice R language was used, detailing usage functions for each code.

Library used

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

Flight data loaded

Flights set **flights_data**, using the package function **nycflights13**

```
f_d <- nycflights13::flights
```

Exercises

5.2.4 Exercises: Items 1 and 2

By solving this point 1, using the **filter()** function of tidyverse, the rows **flights_data** are selected and the value of the column **arr_delay** (arrival delay) is greater than or equal to 2. where it is stored in **filtered_de flights**.

```
myDf1 <- filter(f_d, arr_delay >= 2)
```

The **Knitr* function is now used to mark the table with the aircraft tail number and delay time:

```
library(knitr)
kable(f_d[1:10,c(12,9)],caption = "ARRIVE DELAY", align = "c")
```

Table 1: ARRIVE DELAY

tailnum	arr_delay
N14228	11
N24211	20
N619AA	33
N804JB	-18
N668DN	-25
N39463	12
N516JB	19
N829AS	-14
N593JB	-8
N3ALAA	8

Solving item 2 of the exercise. We are using **filter()** to select rows where the value in the dest column equals "HOU". The result is stored in **filtered_hou_flights**.

```
myDf2 <- filter(f_d,dest =="HOU")
```

```
library(knitr)
kable(myDf2[1:10,c(13,14)],caption = "HOUSTON DESTINY", align = "c")
```

Table 2: HOUSTON DESTINY

origin	dest
JFK	HOU
EWR	HOU
EWR	HOU
JFK	HOU
EWR	HOU
JFK	HOU
EWR	HOU
EWR	HOU
JFK	HOU
EWR	HOU

5.3.1 Exercises: All Items

```
sorted_flights_missing_first <- flights %>%
  arrange(desc(is.na(dep_time)))
```

For this code, we perform the following:

1. We take flights from the dataset and cross them through the operator % >%, allowing for sequence operations.Then the function **arrange()** sorts the descending rows according to the column in **s. na(dep_time)**. This column is logical and returns TRUE if **dep_time** (output time) is absent and FALSE if it is not and those with missing values in **dep_time** appear first.

```
library(knitr)
kable(sorted_flights_missing_first[1:10,c(7,12)],caption = "MISSING DATA FIRST", align = "c")
```

Table 3: MISSING DATA FIRST

arr_time	tailnum
NA	N18120
NA	N3EHAA
NA	N3EVAA
NA	N618JB
NA	N10575
NA	N13949
NA	N10575
NA	N759EV
NA	N13550
NA	NA

```
most_delayed_flights <- flights %>%
  arrange(desc(arr_delay))
```

2. We take flights **dataset** and pass them through the operator *% >%. With **arrange()** to sort the rows according to the column **arr_delay**. which means that flights with longer delays come first.

```
library(knitr)
kable(most_delayed_flights[1:10,c(6,9,12)],caption = "MOST DELAYED FLIGHTS", align = "c")
```

Table 4: MOST DELAYED FLIGHTS

dep_delay	arr_delay	tailnum
1301	1272	N384HA
1137	1127	N504MQ
1126	1109	N517MQ
1014	1007	N338AA
1005	989	N665MQ
960	931	N959DL
911	915	N927DA
898	895	N6716C
896	878	N5DMAA
878	875	N523MQ

```
fastest_flights_desc <- flights %>%
  mutate(speed = distance / air_time) %>%
  arrange(desc(speed))
```

3. The data set `flights` is passed through the operator `%>%`. The function `mutate()` creates a new column `speed`. Calculating the speed by dividing the column `distance` * between the column `air_time`. To know its speed of each flight. Creating the column `speed`, and we use the function `arrange()` to sort the rows according to the column `speed`. Where flights with the highest speed are the first.

```
library(knitr)
kable(fastest_flights_desc[1:10,c(12,20)],caption = "FASTEST FLIGHTS", align = "c")
```

Table 5: FASTEST FLIGHTS

tailnum	speed
N666DN	11.723077
N17196	10.838710
N14568	10.800000
N12567	10.685714
N956DL	9.857143
N3768	9.400000
N779JB	9.290698
N5FFAA	9.274286
N3773D	9.236994
N571JB	9.236994

```
farthest_flights <- flights %>%
  arrange(desc(distance))
```

4. Data `flights` * is passed by `%>%`. and the `arrange()` function to sort the rows according to the `distance`. To see flights with long distances first.

```r
library(knitr)
kable(farthest_flights[1:10,c(12,15)],caption = "FARTHEST FLIGHTS", align = "c")
```

Table 6: FARTHEST FLIGHTS

tailnum	air_time
N380HA	659
N380HA	638
N380HA	616
N384HA	639
N381HA	635
N385HA	611
N385HA	612
N389HA	645
N384HA	640
N388HA	633

```r
closest_flights <- flights %>%
  arrange(distance)
```

5. Data **flights** is passed by **%>%** and so the **arrange()** function is used to sort rows according to the **distance**. Where flights with short distances are seen first.

```r
library(knitr)
kable(closest_flights[1:10,c(12,15)],caption = "CLOSEST FLIGHTS", align = "c")
```

Table 7: CLOSEST FLIGHTS

tailnum	air_time
NA	NA
N13989	30
N14972	30
N15983	28
N27962	32
N14902	29
N22909	22
N33182	25
N11194	30
N17560	27

5.4.1 Exercises: Items 2, 3, and 4

R's **select()** function, if you include the name of a variable multiple times, it will appear multiple times in the resulting output. For example:

```r
select(flights, dep_time, dep_time)
```

```
## # A tibble: 336,776 x 1
##    dep_time
##       <int>
##  1      517
##  2      533
##  3      542
```

```
## 4       544
## 5       554
## 6       554
## 7       555
## 8       557
## 9       557
## 10      558
## # i 336,766 more rows
```

The column **`dep_time`** is included twice in the resulting output. Where the function **`any_of()`** selects columns from a dataframe of the column names. Used when a column name vector wants to select columns that are similar to any of the names in the vector.

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

```
## # A tibble: 336,776 x 5
##     year month   day dep_delay arr_delay
##    <int> <int> <int>     <dbl>     <dbl>
##  1  2013     1     1         2        11
##  2  2013     1     1         4        20
##  3  2013     1     1         2        33
##  4  2013     1     1        -1       -18
##  5  2013     1     1        -6       -25
##  6  2013     1     1        -4        12
##  7  2013     1     1        -5        19
##  8  2013     1     1        -3       -14
##  9  2013     1     1        -3        -8
## 10  2013     1     1        -2         8
## # i 336,766 more rows
```

```
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
## # i 336,766 more rows
```

The code selects the names containing the string "TIME," such as "dep_time" and "arr_time."

5.5.2 Exercises: Items 1 and 2

```
flights_modified <- flights %>%
  mutate(
    dep_time_mins = (dep_time %/% 100) * 60 + dep_time %% 100,
    sched_dep_time_mins = (sched_dep_time %/% 100) * 60 + sched_dep_time %% 100)
```

In this code:

We take the data **flights** * and pass it through **%>%**. using **mutate()** by adding new columns to the data. Creating two new columns: **dep_time_mins** and **sched_dep_time_mins**.

Each column is performing calculations to convert the values (stored in HHMM format) in minutes from midnight. It uses **%/%** to get the hours and **%** to get the minutes.

The code is executed and then the dataset **flights_modified** will have the original columns together with the new columns **dep_time_mins** and **sched_dep_time_mins**, representing departure hours (programmed and real) in minutes from midnight.

```
library(knitr)
kable(flights_modified[1:10,c(12,4,5,20,21)],caption = "SCHEDULED DEPARTURE TIME", align = "c")
```

Table 8: SCHEDULED DEPARTURE TIME

tailnum	dep_time	sched_dep_time	dep_time_mins	sched_dep_time_mins
N14228	517	515	317	315
N24211	533	529	333	329
N619AA	542	540	342	340
N804JB	544	545	344	345
N668DN	554	600	354	360
N39463	554	558	354	358
N516JB	555	600	355	360
N829AS	557	600	357	360
N593JB	557	600	357	360
N3ALAA	558	600	358	360

```
comparison_result <- flights_modified %>%
  mutate(arr_dep_time_diff = arr_time - dep_time_mins) %>%
  filter(!is.na(air_time) & !is.na(arr_dep_time_diff)) %>%
  select(air_time, arr_dep_time_diff)
```

In this second code:

We take the data **flights_modified** (the above result) and pass it through **%>%**. the function **mutate()** is used by creating a new column called **arr_dep_time_diff**. Calculating the difference between arrival time **arr_time** and departure time in minutes from midnight **dep_time_mins**. Then **filter()** is used by removing rows where values are missing in the columns **air_time** or **arr_dep_time_diff**. and finally, use **select()** * to choose the columns **air_time** and **arr_dep_time_diff**.

```
library(knitr)
kable(comparison_result[1:10,c(1,2)],caption = "COMPARISION OF ARRIVES AND DEPARTURES", align = "c")
```

Table 9: COMPARISION OF ARRIVES AND DEPARTURES

air_time	arr_dep_time_diff
227	513
227	517
160	581
183	660
116	458
150	386
158	558

air_time	arr_dep_time_diff
53	352
140	481
138	395

5.6.7 Exercises: item 1

1. Calculate the average arrival delay for the flight group. It provides a central value that represents the typical delay experienced upon arrival.
2. Percentage of flights arriving 15 minutes earlier or 15 minutes late, 30 minutes earlier or 30 minutes late or 2 hours late. Helping to understand the distribution of delay scenarios within the group.
3. Calculated departure mean delay for the flight group. Giving an idea of the average flight delay before take-off.
4. Calculate the percentage of flights that are punctual and compare it with the percentage of flights that are extremely delayed. This helps an understanding of how often flights are on time against delays.
5. Create a chart showing the distribution of arrival delays on all flights. Visual representation helps identify common delay ranges and abnormal values.

Question: What's More Important - Arrival Delay or Departure Delay?

He wondered whether delays in arrival or departure of flights had a greater impact on the travel experience. This depends on several factors, as delays in arrival affect passenger schedules and connections, while delays in departure can cause scheduling issues. Both are important, but their importance varies according to passengers' priorities and plans.

5.7.1 Exercises: item 2

```
## # A tibble: 4,044 x 4
##    tailnum total_flights punctual_flights punctuality_percentage
##    <chr>           <int>            <int>                  <dbl>
##  1 N121DE              2                0                      0
##  2 N136DL              1                0                      0
##  3 N143DA              1                0                      0
##  4 N17627              2                0                      0
##  5 N240AT              5                0                      0
##  6 N26906              1                0                      0
##  7 N295AT              4                0                      0
##  8 N302AS              1                0                      0
##  9 N303AS              1                0                      0
## 10 N32626              1                0                      0
## # i 4,034 more rows
```

In this code:

We start with `flights` and use the operator `%>%` to chain operations. then we collect the data by aircraft tail number `tailnum`. So the calculations will be done separately for each aircraft.

The `summary()` function for calculating statistics for each group (aircraft). Within the `summary()`:

After `summary()`, use `arrange()` to sort groups (aircraft) based on the punctuality percentages in ascending order. Where planes with the lowest punctuality percentages will appear first. Then `filter()` is used to remove rows where `punctuality_percentage` (`NA`) is not available. Where the resulting data set is assigned to `worst_punctuality`, containing the tail numbers of aircraft, the total number of flights, the number of punctual flights and the corresponding punctuality percentages.

Finally, the data set **worst_punctuality** is displayed to see aircraft tail numbers with the lowest punctuality percentages.

```r
library(knitr)
kable(worst_punctuality[1:10,c(1,2,3,4)],caption = "WORST PUNCTUALITY TOP", align = "c")
```

Table 10: WORST PUNCTUALITY TOP

tailnum	total_flights	punctual_flights	punctuality_percentage
N121DE	2	0	0
N136DL	1	0	0
N143DA	1	0	0
N17627	2	0	0
N240AT	5	0	0
N26906	1	0	0
N295AT	4	0	0
N302AS	1	0	0
N303AS	1	0	0
N32626	1	0	0