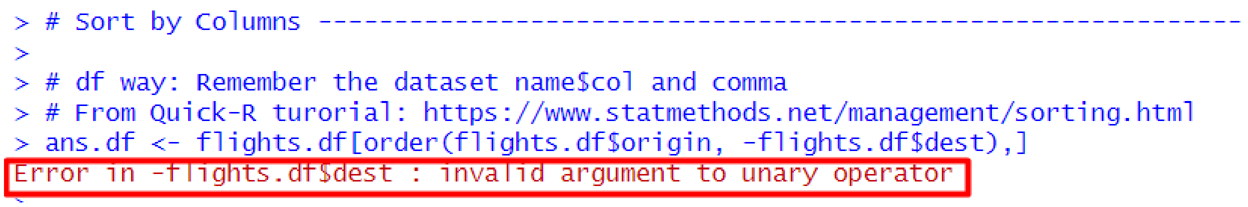
Group 18 | Pepito, Alyssa Mae M. & Soleño, Keziah Antonette C BM2 | CS174

M1-FA2 Data Exploration and Summaries

Previously in M1-FA1.2, we used R studio to create a simple multiple regression model using R programming. We were tasked to download a specific dataset and use it for our regression model to analyze. For this assessment, we are tasked with running an R script with a dataset involving flights and then observe a few specified details such as arrival delay, departure delay, or number of flights.

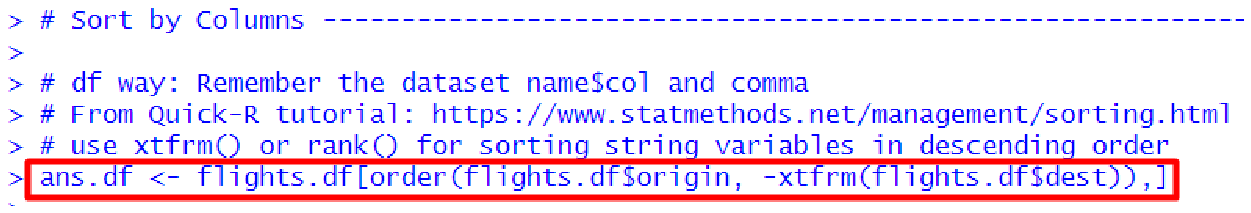
I. Running the R Script

While running the R script, we encountered an error regarding sorting the data frame by columns. The problem appears to be that `dest` is not a numeric column but rather a string column, so `-dest` does not work. So, to fix the problem we used the *xtfrm()* function to convert the column into one.



```
> # Sort by Columns -------------------------------------------------------------
>
> # df way: Remember the dataset name$col and comma
> # From Quick-R turorial: https://www.statmethods.net/management/sorting.html
> ans.df <- flights.df[order(flights.df$origin, -flights.df$dest),]
Error in -flights.df$dest : invalid argument to unary operator
```

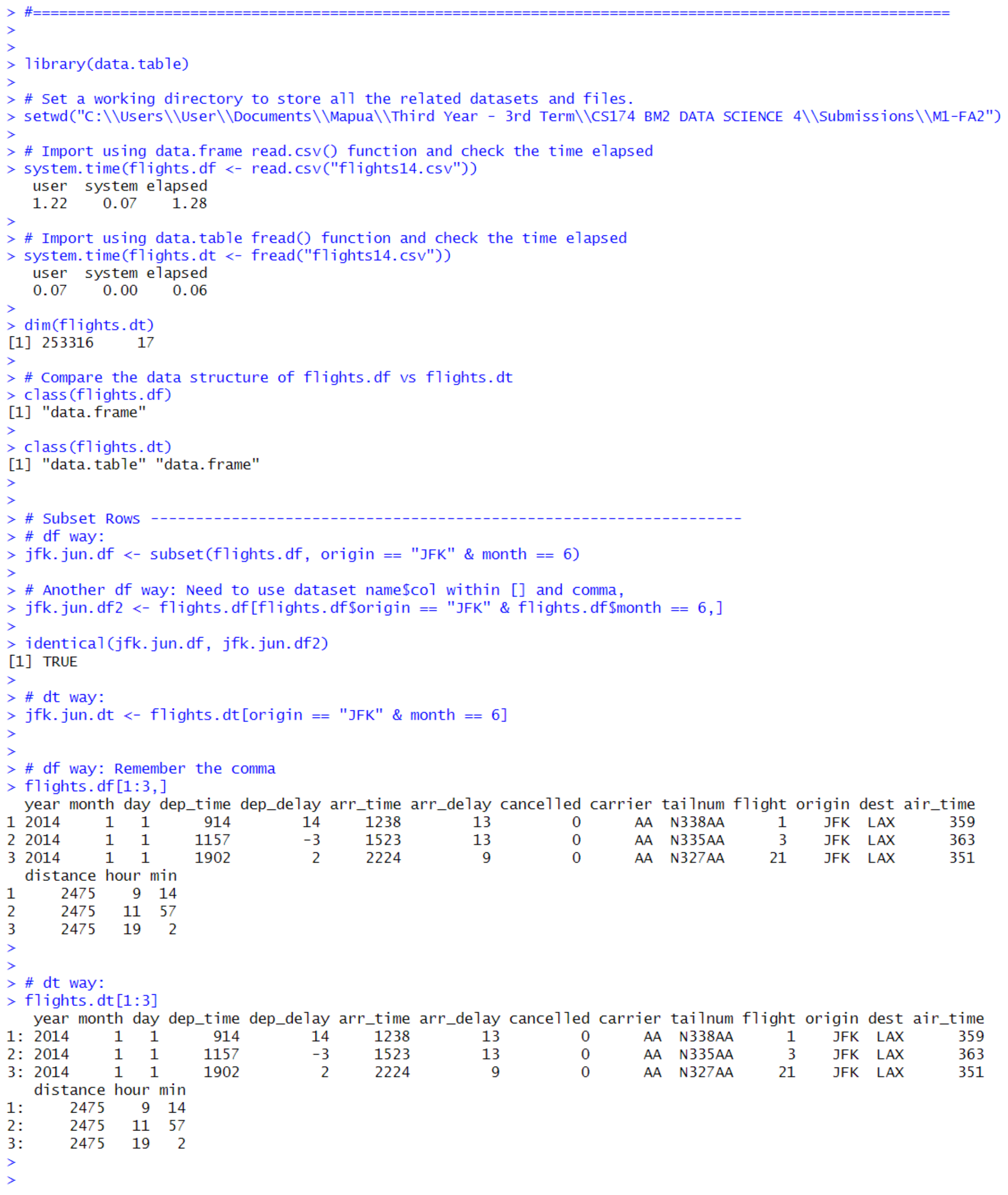
The error encountered in -flights.df$dest



```
> # Sort by Columns -------------------------------------------------------------
>
> # df way: Remember the dataset name$col and comma
> # From Quick-R tutorial: https://www.statmethods.net/management/sorting.html
> # use xtfrm() or rank() for sorting string variables in descending order
> ans.df <- flights.df[order(flights.df$origin, -xtfrm(flights.df$dest)),]
```

Using xtfrm function to fix the error

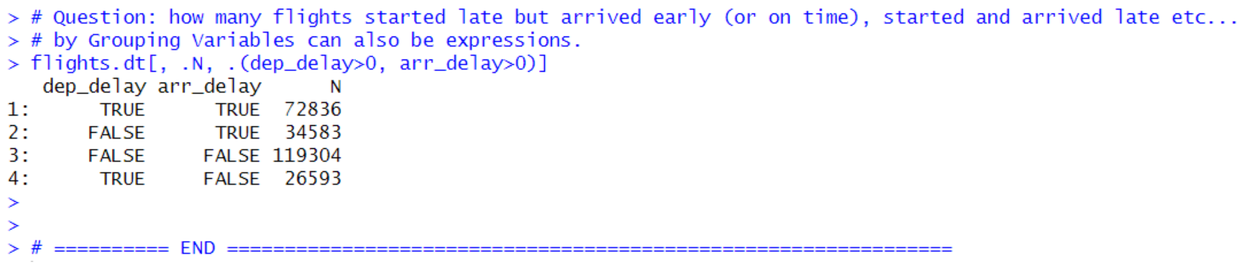
After the problem has been fixed, the rest of the code was run without errors, as seen in the code snippet below.



```
> #==========================================================================================================
>
>
> library(data.table)
>
> # Set a working directory to store all the related datasets and files.
> setwd("C:\\Users\\User\\Documents\\Mapua\\Third Year - 3rd Term\\CS174 BM2 DATA SCIENCE 4\\Submissions\\M1-FA2")
>
> # Import using data.frame read.csv() function and check the time elapsed
> system.time(flights.df <- read.csv("flights14.csv"))
   user  system elapsed 
   1.22    0.07    1.28 
>
> # Import using data.table fread() function and check the time elapsed
> system.time(flights.dt <- fread("flights14.csv"))
   user  system elapsed 
   0.07    0.00    0.06 
>
> dim(flights.dt)
[1] 253316     17
>
> # Compare the data structure of flights.df vs flights.dt
> class(flights.df)
[1] "data.frame"
>
> class(flights.dt)
[1] "data.table" "data.frame"
>
>
> # Subset Rows -------------------------------------------------------------------
> # df way:
> jfk.jun.df <- subset(flights.df, origin == "JFK" & month == 6)
>
> # Another df way: Need to use dataset name$col within [] and comma,
> jfk.jun.df2 <- flights.df[flights.df$origin == "JFK" & flights.df$month == 6,]
>
> identical(jfk.jun.df, jfk.jun.df2)
[1] TRUE
>
> # dt way:
> jfk.jun.dt <- flights.dt[origin == "JFK" & month == 6]
>
>
> # df way: Remember the comma
> flights.df[1:3,]
  year month day dep_time dep_delay arr_time arr_delay cancelled carrier tailnum flight origin dest air_time
1 2014     1   1      914        14     1238        13         0      AA  N338AA      1    JFK  LAX      359
2 2014     1   1     1157        -3     1523        13         0      AA  N335AA      3    JFK  LAX      363
3 2014     1   1     1902         2     2224         9         0      AA  N327AA     21    JFK  LAX      351
  distance hour min
1     2475    9  14
2     2475   11  57
3     2475   19   2
>
>
> # dt way:
> flights.dt[1:3]
   year month day dep_time dep_delay arr_time arr_delay cancelled carrier tailnum flight origin dest air_time
1: 2014     1   1      914        14     1238        13         0      AA  N338AA      1    JFK  LAX      359
2: 2014     1   1     1157        -3     1523        13         0      AA  N335AA      3    JFK  LAX      363
3: 2014     1   1     1902         2     2224         9         0      AA  N327AA     21    JFK  LAX      351
   distance hour min
1:     2475    9  14
2:     2475   11  57
3:     2475   19   2
>
>
```

```
> # Sort by Columns -------------------------------------------------------------
>
> # df way: Remember the dataset name$col and comma
> # From Quick-R tutorial: https://www.statmethods.net/management/sorting.html
> # use xtfrm() or rank() for sorting string variables in descending order
> ans.df <- flights.df[order(flights.df$origin, -xtfrm(flights.df$dest)),]
>
> # dt way:
> ans.dt <- flights.dt[order(origin, -dest)]
>
>
> # Select and rename columns ---------------------------------------------------
> # df way: remember the quotation marks
> ans.df <- flights.df[, c("arr_delay", "dep_delay")]
> names(ans.df) <- c("delay_arr", "delay_dep")
>
> # dt way:
> ans.dt <- flights.dt[, .(delay_arr = arr_delay, delay_dep = dep_delay)]
>
>
> # Question: How many trips have had total delay < 0? ---------------------------
> # df way: Use two functions nrow() and subset()
> nrow(subset(flights.df, (arr_delay + dep_delay) < 0))
[1] 141814
>
> # dt way: j can take expressions.
> flights.dt[, sum((arr_delay + dep_delay) < 0)]
[1] 141814
>
>
> # Question: What is the average arrival and departure delay for all flights with "JFK" as the origin airport in t
he month of June?
> # df way: subset and sapply
> jfk.jun.delay.df <- subset(flights.df, origin == "JFK" & month == 6,
+                            select = c(arr_delay, dep_delay))
> sapply(jfk.jun.delay.df, mean)
arr_delay dep_delay 
 5.839349  9.807884 
>
> # dt way:
> flights.dt[origin == "JFK" & month == 6, .(avg_arr_delay = mean(arr_delay),
+                                             avg_dep_delay = mean(dep_delay))]
   avg_arr_delay avg_dep_delay
1:      5.839349      9.807884
>
>
> # Question: How many trips have been made in 2014 from JFK airport in the month of June?
> # df way:
> nrow(subset(flights.df, origin == "JFK" & month == 6))
[1] 8422
>
> # dt way: .N is a special in-built variable that holds the number of obs in the group
> flights.dt[origin == "JFK" & month == 6, .N]
[1] 8422
>
>
> # Question: Number of trips corresponding to each origin airport?
> # df way:
> summary(flights.df$origin)
   Length     Class      Mode 
   253316 character character 
>
> # dt way:
> flights.dt[, .N, by = origin]
   origin     N
1:    JFK 81483
2:    LGA 84433
3:    EWR 87400
>
>
```

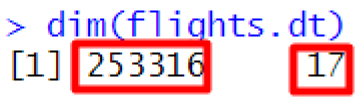
```
> # Question: total number of trips for each origin, dest pair for carrier code AA?
> ans.df <- subset(flights.df, carrier == "AA", select = c(origin,dest))
> table(ans.df$origin, ans.df$dest)

       AUS  BOS  DCA  DFW  EGE  IAH  LAS  LAX  MCO  MIA  ORD  PBI  PHX  SAN  SEA  SFO  SJU  STT
  EWR    0    0    0 1618    0    0    0   62    0  848    0    0  121    0    0    0    0    0
  JFK  297 1173  172  474   85    7  595 3387  597 1876  432    0    0  299  298 1312  690  229
  LGA    0    0    0 3785    0    0    0    0    0 3334 4366  245    0    0    0    0    0    0
> ## Many zeros. Not a good output. Use data.table.
>
> # dt way:
> flights.dt[carrier == "AA", .N, by = .(origin,dest)]
    origin dest    N
 1:    JFK  LAX 3387
 2:    LGA  PBI  245
 3:    EWR  LAX   62
 4:    JFK  MIA 1876
 5:    JFK  SEA  298
 6:    EWR  MIA  848
 7:    JFK  SFO 1312
 8:    JFK  BOS 1173
 9:    JFK  ORD  432
10:    JFK  IAH    7
11:    JFK  AUS  297
12:    EWR  DFW 1618
13:    LGA  ORD 4366
14:    JFK  STT  229
15:    JFK  SJU  690
16:    LGA  MIA 3334
17:    LGA  DFW 3785
18:    JFK  LAS  595
19:    JFK  MCO  597
20:    JFK  EGE   85
21:    JFK  DFW  474
22:    JFK  SAN  299
23:    JFK  DCA  172
24:    EWR  PHX  121
    origin dest    N
>
>
> # Question: average arrival, departure delay and number of flights for each orig, dest pair for each month for ca
rrier code AA?
> ans.dt <- flights.dt[carrier == "AA", .(avg.arr.delay = mean(arr_delay), avg.dep.delay = mean(dep_delay), .N), by
= .(origin, dest, month)]
>
> # Same as above and in addition, to sort results by the 3 grouping variables via keyby.
> ans.dt <- flights.dt[carrier == "AA", .(avg.arr.delay = mean(arr_delay), avg.dep.delay = mean(dep_delay), .N), ke
yby = .(origin, dest, month)]
>
>
> # Question: total number of trips for each origin, dest pair for carrier AA, and sort origin by ascending order a
nd then dest by descending order
> # Use data table chaining to avoid creating intermediate data strructures to hold temporary results.
> flights.dt[carrier == "AA", .N, by = .(origin, dest)][order(origin, -dest)]
    origin dest    N
 1:    EWR  PHX  121
 2:    EWR  MIA  848
 3:    EWR  LAX   62
 4:    EWR  DFW 1618
 5:    JFK  STT  229
 6:    JFK  SJU  690
 7:    JFK  SFO 1312
 8:    JFK  SEA  298
 9:    JFK  SAN  299
10:    JFK  ORD  432
11:    JFK  MIA 1876
12:    JFK  MCO  597
13:    JFK  LAX 3387
14:    JFK  LAS  595
15:    JFK  IAH    7
16:    JFK  EGE   85
17:    JFK  DFW  474
18:    JFK  DCA  172
19:    JFK  BOS 1173
20:    JFK  AUS  297
21:    LGA  PBI  245
22:    LGA  ORD 4366
23:    LGA  MIA 3334
24:    LGA  DFW 3785
    origin dest    N
>
```



```
> # Question: how many flights started late but arrived early (or on time), started and arrived late etc...
> # by Grouping Variables can also be expressions.
> flights.dt[, .N, .(dep_delay>0, arr_delay>0)]
   dep_delay arr_delay      N
1:      TRUE      TRUE  72836
2:     FALSE      TRUE  34583
3:     FALSE     FALSE 119304
4:      TRUE     FALSE  26593
>
>
> # ========== END ==============================================================
```

II. Determining dim(flights.dt)

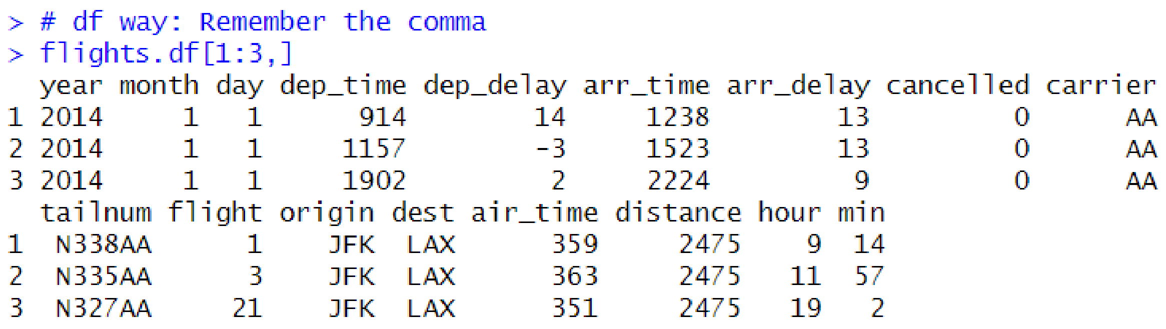
The *dim(flights.dt)* uses the *dim()* function to get the dimension of the specified data frame. In this case, *flights.dt* is the data frame and the *dim(flights.dt)* results reveal **two hundred fifty-three thousand three hundred sixteen observable data/entries** and **seventeen attributes/variables**, as shown below.



```
> dim(flights.dt)
[1] 253316     17
```

III. Determining the output of flights.df[1:3,]

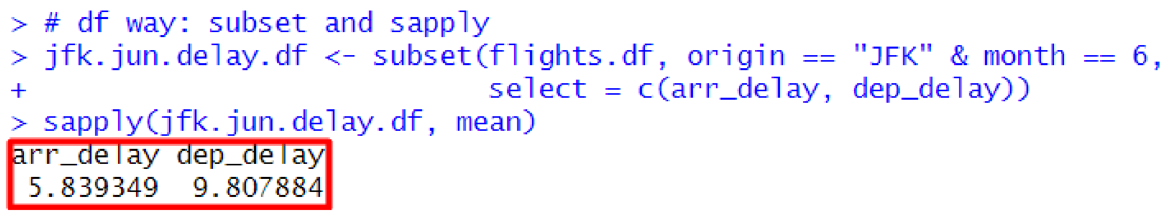
In R programming, we can subset the first N rows of a data frame by simply using square brackets together with the data frame name as observed in the following image. In this R script, we subset the first three rows of the *flights.df* data frame.



```
> # df way: Remember the comma
> flights.df[1:3,]
  year month day dep_time dep_delay arr_time arr_delay cancelled carrier
1 2014     1   1      914        14     1238        13         0      AA
2 2014     1   1     1157        -3     1523        13         0      AA
3 2014     1   1     1902         2     2224         9         0      AA
  tailnum flight origin dest air_time distance hour min
1  N338AA      1    JFK  LAX      359     2475    9  14
2  N335AA      3    JFK  LAX      363     2475   11  57
3  N327AA     21    JFK  LAX      351     2475   19   2
```

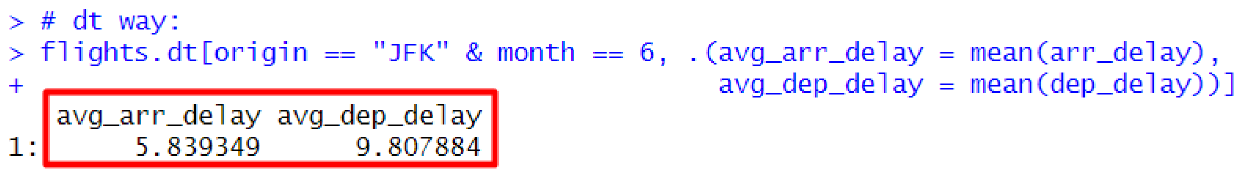
IV. Determining the average arrival and departure delay

In the R script, we determined the average arrival and departure delay for all flights with "JFK" as the origin airport in June using the data frame and data table way. In the data frame way, we use the *subset()* and *sapply()* functions. The *subset()* is used to select variables and as seen in the code snippet below, we created a new data frame variable and selected all rows from the `flights.df` data frame with a value of origin to be "JFK" and month to "6". We keep the `arr_delay` and `dep_delay` columns. Next, the *sapply()* function takes a data frame as input and gives output in vector or matrix. Here, we used `jfk.jun.delay.df` as the input and the mean as the output.



```
> # df way: subset and sapply
> jfk.jun.delay.df <- subset(flights.df, origin == "JFK" & month == 6,
+                            select = c(arr_delay, dep_delay))
> sapply(jfk.jun.delay.df, mean)
arr_delay dep_delay
 5.839349  9.807884
```

In the data table way, we can immediately subset rows in the `flights.dt` data table without using the *subset()* function. We then recreate the datatable with new columns based on the summarized values of rows. The summary function that we are using is the mean() function to get the average of the `arr_delay` and `dep_delay` columns.

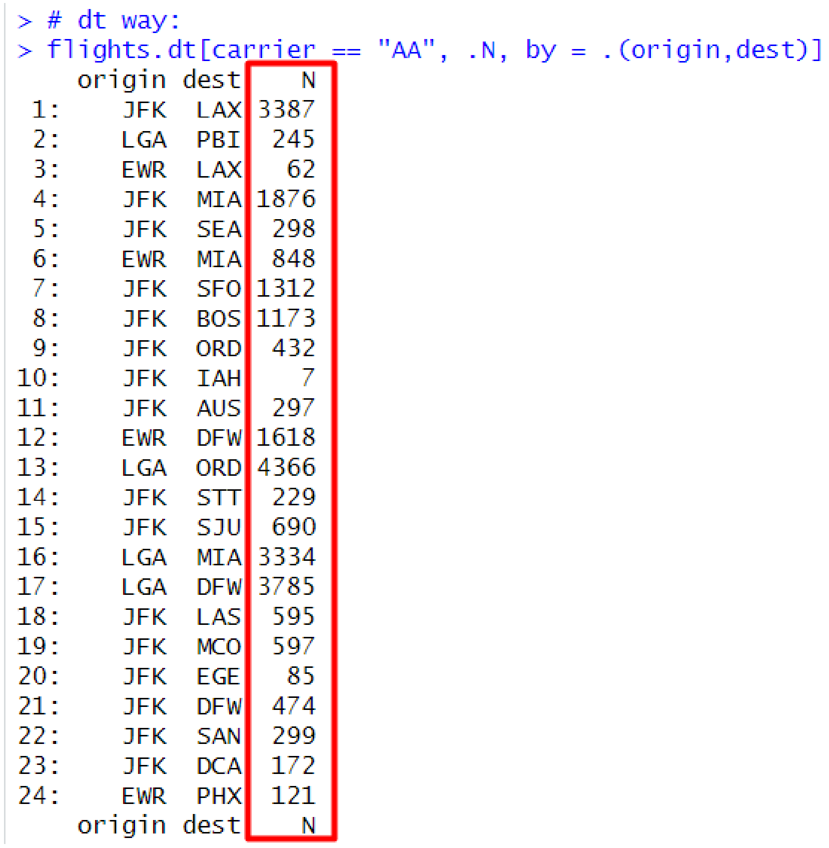


```
> # dt way:
> flights.dt[origin == "JFK" & month == 6, .(avg_arr_delay = mean(arr_delay),
+                                             avg_dep_delay = mean(dep_delay))]
   avg_arr_delay avg_dep_delay
1:      5.839349      9.807884
```

From both ways, we were able to determine that the **average arrival delay is 5.839349** and **average departure delay is 9.807884**.

V. Determining flights.dt[carrier == "AA", .N, by = .(origin,dest)]

In the R script, we determined the total number of trips for each origin, dest pair for carrier code AA. Instead of using a data frame (which causes many zeroes and is not a good output), the data table way is shorter and gives a better result. We first subset the data table based on the value of carrier as “AA”. We use `.N` to get each group's total number of observations as determined by “`by = .(origin,dest)`” meaning it groups rows by values in origin and dest columns. We highlighted the total number of trips for each origin, dest pair for carrier code AA below.

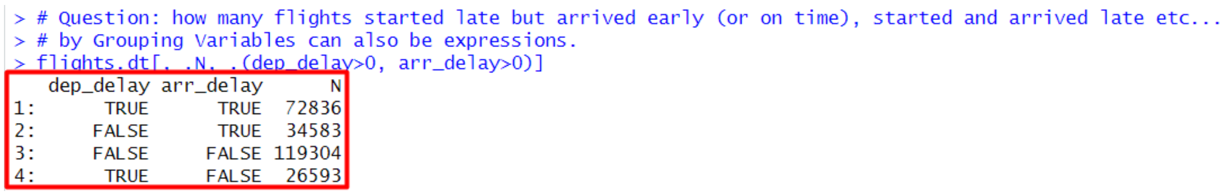


```
> # dt way:
> flights.dt[carrier == "AA", .N, by = .(origin,dest)]
    origin dest    N
 1:    JFK  LAX 3387
 2:    LGA  PBI  245
 3:    EWR  LAX   62
 4:    JFK  MIA 1876
 5:    JFK  SEA  298
 6:    EWR  MIA  848
 7:    JFK  SFO 1312
 8:    JFK  BOS 1173
 9:    JFK  ORD  432
10:    JFK  IAH    7
11:    JFK  AUS  297
12:    EWR  DFW 1618
13:    LGA  ORD 4366
14:    JFK  STT  229
15:    JFK  SJU  690
16:    LGA  MIA 3334
17:    LGA  DFW 3785
18:    JFK  LAS  595
19:    JFK  MCO  597
20:    JFK  EGE   85
21:    JFK  DFW  474
22:    JFK  SAN  299
23:    JFK  DCA  172
24:    EWR  PHX  121
    origin dest    N
```

VI. Determining the number of flights that started late but arrived early (or on time), started and arrived late, etc.

Lastly, we were able to determine the number of flights that started and arrived late, started early but arrived late, started and arrived early (or on time), and started late but arrive early (or on time). Using the logical operator >, we subset rows based on the value of `dep_delay` to be more than 0 and `arr_delay` to be more than 0. We observed the following:

- The first row corresponds to `dep_delay > 0 = TRUE` and `arr_delay > 0 = TRUE`. We can see that **72,836** flights started and arrived late.
- The second row corresponds to `dep_delay > 0 = FALSE` and `arr_delay > 0 = TRUE`. We can see that **34,583** flights started early but arrived late.
- The third row corresponds to `dep_delay > 0 = FALSE` and `arr_delay > 0 = FALSE`. We can see that **119,304** flights started and arrived early (or on time).
- The last row corresponds to `dep_delay > 0 = TRUE` and `arr_delay > 0 = FALSE`. We can see that **26,593** flights started late but arrived early (or on time).



```
> # Question: how many flights started late but arrived early (or on time), started and arrived late etc...
> # by Grouping Variables can also be expressions.
> flights.dt[, .N, .(dep_delay>0, arr_delay>0)]
   dep_delay arr_delay      N
1:      TRUE      TRUE  72836
2:     FALSE      TRUE  34583
3:     FALSE     FALSE 119304
4:      TRUE     FALSE  26593
```

References

"Extract First N Rows of Data Frame in R (3 Examples) | Select and Subset." *Statistics Globe*, 17 Mar. 2022, statisticsglobe.com/extract-first-n-rows-of-data-frame-in-r.

GeeksforGeeks. "Get or Set Dimensions of a Matrix in R Programming Dim Function." *GeeksforGeeks*, 4 June 2020, www.geeksforgeeks.org/get-or-set-dimensions-of-a-matrix-in-r-programming-dim-function.

Johnson, Daniel. "Apply(), Lapply(), Sapply(), Tapply() Function in R With Examples." *Guru99*, 21 Jan. 2023, www.guru99.com/r-apply-sapply-tapply.html.

Naveen. "R Sort DataFrame Rows by Column Value." *Spark by {Examples}*, 25 Aug. 2022, sparkbyexamples.com/r-programming/sort-data-frame-in-r.

"Problem Sorting R Data Frame in Descending Order." *Stack Overflow*, 11 Dec. 2019, stackoverflow.com/questions/59285124/problem-sorting-r-data-frame-in-descending-order.

Quick-R: Sorting. www.statmethods.net/management/sorting.html.

Quick-R: Subsetting Data. www.statmethods.net/management/subset.html.

Rdatatable. "Getting Started." *GitHub*, github.com/Rdatatable/data.table/wiki/Getting-started.