Week 6 Data Transformation

Pooja Chopra

2019-06-07

In real word, Data which we get is not exactly the way we need it , In this we need to transform data like Creating new variable , Reorder or rename variable to make data as per ur requirment these changes are called data transformation. In this we are going to use DPLYR Package and NY Flight Dataset nycflights13.

In this we will be covering five key dplyr functions 1. filter 2. arrange 3. select 4. mutate 5. summarise

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

Flights dataset Contains 336776 obs. and 19 variables

```
str(flights)
```

```
# 1. Find all flights that
# Q1 Had an arrival delay of two or more hours
filter(flights, arr_delay>=120)

# Q2 Flew to Houston (IAH or HOU)
filter(flights, dest == 'IAH' | dest == 'HOU')
filter(flights, dest %in% c('IAH', 'HOU'))

# Q3 Were operated by United, American, or Delta
filter(flights, carrier == 'UA' | carrier == 'AA' | carrier == 'DL')
filter(flights, carrier %in% c('UA', 'AA', 'DL'))

# Q4 Departed in summer (July, August, and September)
filter(flights, month >= 7 & month <= 9)
filter(flights, month %in% c(7, 8, 9))

# Q5 Arrived more than two hours late, but didn't leave late
filter(flights, arr_delay > 120, dep_delay <= 0)

# Q6 Were delayed by at least an hour, but made up over 30 minutes in flight
filter(flights, dep_delay >= 60, dep_delay-arr_delay > 30)

# Q7 Departed between midnight and 6am (inclusive)
filter(flights, dep_time <=600 | dep_time == 2400)

#2 Another useful dplyr filtering helper is between(). What does it do? Can you use it to simplify the

#Ans: Between is a shorter, faster way of testing two inequalities at once: it tests if its first argum

filter(flights, between(flights$dep_time, 600, 700))

# 3. How many flights have a missing dep_time? What other variables are missing? What might these rows
summary(flights)
```

```r
# 8255 flights have a missing `dep_time`, 8255 have a missing `dep_delay`, 8713 have a missing `arr_tim

#These could be lost data about perfectly normal flights or .
```

```r
#arrange fuction changes the order of rows, we can use desc in the code to reorder by a column in desce

#Q1: How could you use arrange() to sort all missing values to the start? (Hint: use is.na()).
df <- tibble(x = c(3, 7, NA))

arrange(df, desc(is.na(x)))
arrange(df, -(is.na(x)))

#Q2: Sort flights to find the most delayed flights. Find the flights that left earliest.

arrange(flights, desc(dep_delay))
arrange(flights, dep_delay)

#Q3. Sort flights to find the fastest flights.
arrange(flights, air_time)

# Q4. Which flights travelled the longest? Which travelled the shortest?
# Longest
flights %>%
  arrange(-air_time) %>%
  select(carrier, flight, air_time)

# Shortest
flights %>%
  arrange(air_time) %>%
  select(carrier, flight, air_time)
```

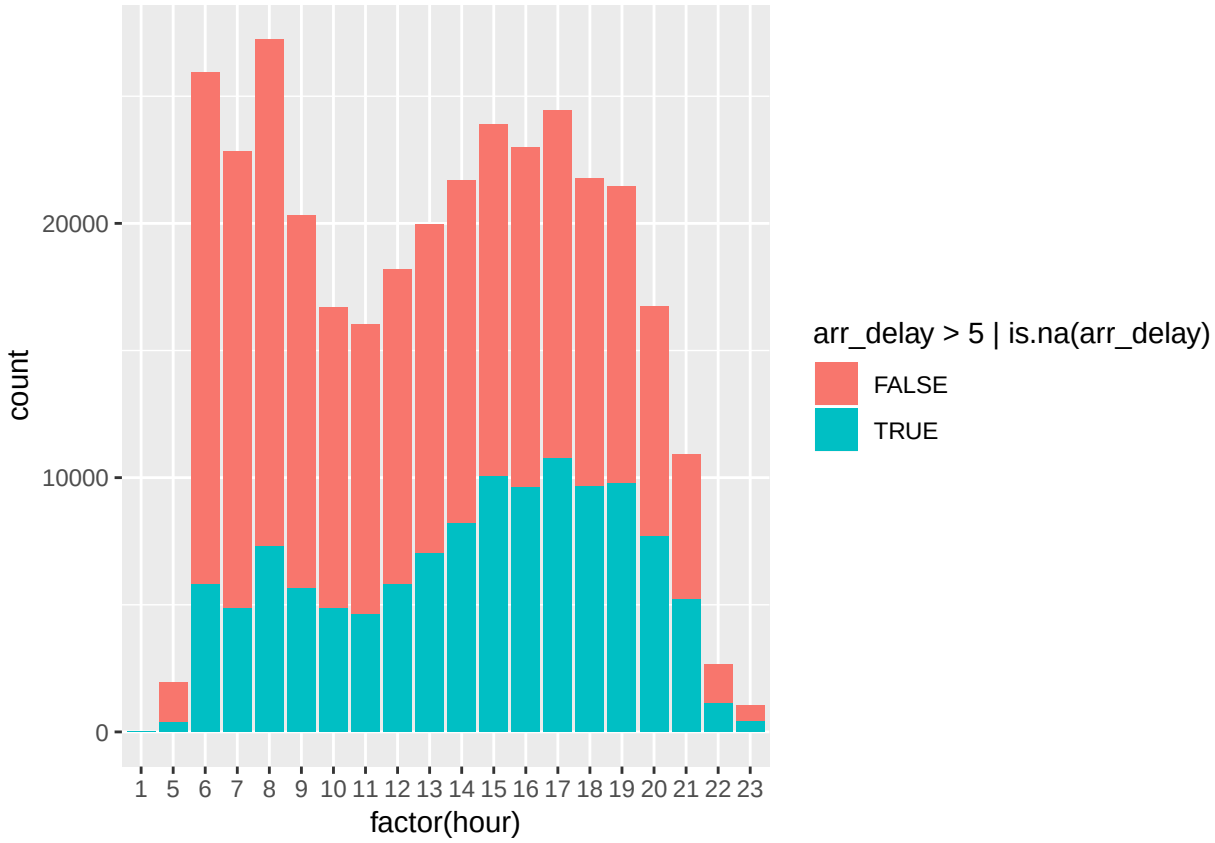
```r
#Q1 Brainstorm as many ways as possible to select dep_time, dep_delay, arr_time, and arr_delay from fli
select(flights, dep_time, dep_delay, arr_time, arr_delay)
select(flights, starts_with("dep"), starts_with("arr"))
select_(flights, "dep_time", "dep_delay", "arr_time", "arr_delay")
select(flights, matches("dep"), matches("arr"), -matches("sched"), -carrier)
select(flights, matches("time$|delay$"), -contains("sched"), -contains("air"))

#Q2. What happens if you include the name of a variable multiple times in a select() call?
# it will show depeated column name once
select(flights, dep_time, dep_time , arr_delay)

# Q3. What does the one_of() function do? Why might it be helpful in conjunction with this vector?

# oneof() function let us select variable from a vector instead of putting variable name in select call

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

# 4. Does the result of running the following code surprise you? How do the select helpers deal with ca
select(flights, contains("TIME"))
select(flights, contains("TIME", ignore.case = F))
```

```r
# Mutate add new columns at the end of the dataset after doing calcuations based on existig column.

# Q1. Currently dep_time and sched_dep_time are convenient to look at, but hard to compute with because

mutate(flights,
       dep_time = (dep_time %/% 100) * 60 + (dep_time %% 100),
       sched_dep_time = (sched_dep_time %/% 100) * 60 + (sched_dep_time %% 100))

str(flights)

#Q2. Compare air_time with arr_time - dep_time. What do you expect to see? What do you see? What do you

flights %>%
  mutate(dep_time = (dep_time %/% 100) * 60 + (dep_time %% 100),
         sched_dep_time = (sched_dep_time %/% 100) * 60 + (sched_dep_time %% 100),
         arr_time = (arr_time %/% 100) * 60 + (arr_time %% 100),
         sched_arr_time = (sched_arr_time %/% 100) * 60 + (sched_arr_time %% 100)) %>%
  transmute((arr_time - dep_time) %% (60*24) - air_time)

# Q3. Compare dep_time, sched_dep_time, and dep_delay. How would you expect those three numbers to be r

hours2mins <- function(x) {
  x %/% 100 * 60 + x %% 100
}
select(flights, contains("dep")) %>%
  mutate(dep_time_two = hours2mins(dep_time) - hours2mins(sched_dep_time))
# these two numbers don't match
# where the departure time is the next day from the scheduled departure time.
select(flights, contains("dep")) %>%
  mutate(dep_time_two = hours2mins(dep_time) - hours2mins(sched_dep_time)) %>%
  filter(dep_delay != dep_time_two) %>%
  mutate(dep_time_two = hours2mins(dep_time) - hours2mins(sched_dep_time - 2400))
```

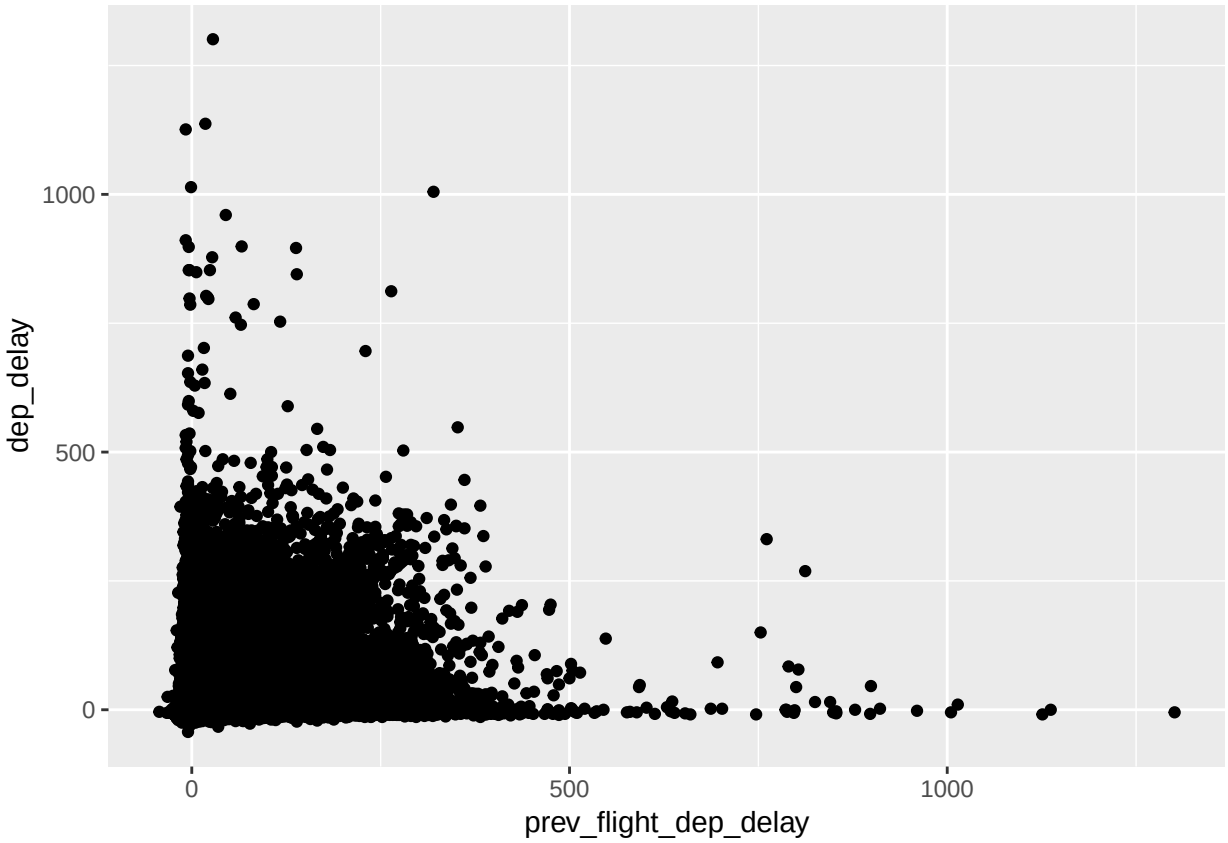
```r
# Q1. Brainstorm at least 5 different ways to assess the typical delay characteristics of a group of fl

delay_char <-
  flights %>%
  group_by(flight) %>%
  summarise(n = n(),
            fifteen_early = mean(arr_delay == -15, na.rm = T),
            fifteen_late = mean(arr_delay == 15, na.rm = T),
            ten_always = mean(arr_delay == 10, na.rm = T),
            thirty_early = mean(arr_delay == -30, na.rm = T),
            thirty_late = mean(arr_delay == 30, na.rm = T),
            percentage_on_time = mean(arr_delay == 0, na.rm = T),
            twohours = mean(arr_delay > 120, na.rm = T)) %>%
  map_if(is_double, round, 2) %>%
  as_tibble()

# 1.1. A flight is 15 minutes early 50% of the time, and 15 minutes late 50% of the time.
delay_char  %>%
  filter(fifteen_early == 0.5, fifteen_late == 0.5)
```

```r
# 1.2. A flight is always 10 minutes late
delay_char %>%
  filter(ten_always == 1)

# 1.3. 99% of the time a flight is on time. 1% of the time it's 2 hours late
delay_char %>%
  filter(percentage_on_time == 0.99 & twohours == 0.01)


# Q2. Come up with another approach that will give you the same output as not_cancelled %>% count(dest)

not_cancelled <- filter(flights, !is.na(dep_delay), !is.na(arr_delay))
not_cancelled %>%
  group_by(dest) %>%
  tally()
not_cancelled %>%
  group_by(tailnum) %>%
  summarise(n = sum(distance))

# Q3. Our definition of cancelled flights (is.na(dep_delay) | is.na(arr_delay) ) is slightly suboptimal

#In this case we can just use `!is.na(dep_delay)`

flights %>%
    group_by(departed = !is.na(dep_delay), arrived = !is.na(arr_delay)) %>%
    summarise(n=n())

# Q4. What does the sort argument to count() do. When might you use it?
# The sort argument to `count()` sorts by descending order of `n`. This is useful because often the mos
```

```r
#Q1 Which plane (tailnum) has the worst on-time record?

flights %>%
  group_by(tailnum) %>%
  summarise(prop_on_time = sum(arr_delay <= 30 & !is.na(arr_delay))/n(),
            mean_arr_delay = mean(arr_delay, na.rm=TRUE),
            flights = n()) %>%
  arrange(prop_on_time, desc(mean_arr_delay))
flights %>%
  group_by(tailnum) %>%
  filter(all(is.na(arr_delay))) %>%
    tally(sort=TRUE)

# Many of the planes  never reached on time, and 7 went missing . Missing data which we found is becaus

# Q2. What time of day should you fly if you want to avoid delays as much as possible?
flights %>%
    ggplot(aes(x=factor(hour), fill=arr_delay>5 | is.na(arr_delay))) + geom_bar()
```




```
# Q3. Delays are typically temporally correlated: even once the problem that caused the initial delay h
flights %>%
  mutate(new_sched_dep_time = lubridate::make_datetime(year, month, day, hour, minute)) %>%
  group_by(origin) %>%
  arrange(new_sched_dep_time) %>%
  mutate(prev_flight_dep_delay = lag(dep_delay)) %>%
  ggplot(aes(x=prev_flight_dep_delay, y= dep_delay)) + geom_point()
```




```
# Q4. Find all destinations that are flown by at least two carriers. Use that information to rank the c
flights %>%
  group_by(dest) %>%
  filter(n_distinct(carrier)>=2) %>%
  group_by(carrier) %>%
  summarise(possible_transfers = n_distinct(dest)) %>%
  arrange(desc(possible_transfers))
```