Week 3 : Transformation

**Install.packages(“nycflights13”)** [Also tidyverse, if you don’t have already]

Data frame to use: **flights**

Assignment Submission

1. Filter()
   1. Find all flights that
   2. Had an arrival delay of two or more hours
   3. Flew to Houston (IAH or HOU)
   4. Were operated by United, American, or Delta
   5. Departed in summer (July, August, and September)
   6. Arrived more than two hours late, but didn’t leave late
   7. Were delayed by at least an hour, but made up over 30 minutes in flight
   8. Departed between midnight and 6am (inclusive)
   9. How many flights have a missing dep\_time? What other variables are missing? What might these rows represent?
2. Arrange()
   1. Sort flights to find the most delayed flights. Find the the flights that left earliest.
   2. Sort flights to find the fastest flights.
   3. Which flights travelled the longest? Which travelled the shortest?
3. Select()
4. Mutate ()
   1. Currently dep\_time and sched\_dep\_time are convenient to look at, but hard to compute with because they’re not really continuous numbers. Convert them to a more convenient representation of number of minutes since midnight.
   2. Compare air\_time with arr\_time - dep\_time . What do you expect to see? What do you see? What do you need to do to fix it?
   3. Compare dep\_time , sched\_dep\_time , and dep\_delay . How would you expect those three numbers to be related?
   4. Find the 10 most delayed flights using a ranking function. How do you want to handle ties? Carefully read the documentation for min\_rank() .
   5. What does 1:3 + 1:10 return? Why?
   6. What trigonometric functions does R provide?
5. Summarize (Group\_by)
   1. Explore the relationship between the distance and average delay for each location. Try the general way as well as the pipe way.
   2. Which carrier has the worst delays? Challenge: can you disentangle the effects of bad airports vs. bad carriers? Why/why not? (Hint: think about flights %>% group\_by(carrier, dest) %>% summarise(n()) )
   3. Look at the number of cancelled flights per day. Is there a pattern? Is the proportion of cancelled flights related to the average delay?

Misc

1. Refer back to the lists of useful mutate and filtering functions. Describe how each operation changes when you combine it with grouping.
2. Which plane ( tailnum ) has the worst on-time record?
3. What time of day should you fly if you want to avoid delays as much as possible?
4. For each destination, compute the total minutes of delay. For each flight, compute the proportion of the total delay for its destination.
5. Delays are typically temporally correlated: even once the problem that caused the initial delay has been resolved, later flights are delayed to allow earlier flights to leave. Using lag() , explore how the delay of a flight is related to the delay of the immediately preceding flight.
6. Look at each destination. Can you find flights that are suspiciously fast? (i.e. flights that represent a potential data entry error). Compute the air time a flight relative to the shortest flight to that destination. Which flights were most delayed in the air?
7. Find all destinations that are flown by at least two carriers. Use that information to rank the carriers.
8. For each plane, count the number of flights before the first delay of greater than 1 hour.

### **ANSWERS**

**1. Filter()**

Find all flights that

(a) Had an arrival delay of two hours.

library(nycflights13)

library(tidyverse)

#glimpse(flights)

filter(flights, arr\_delay > 119 & arr\_delay < 121)

(b) flew to Houston

filter(flights, dest == 'HOU' | dest == 'IAH')

(c) Were operated by American, United or Delta airlines.

filter(flights, carrier == "UA" | carrier == 'AA' | carrier == 'DA')

(d) Departed in the summeer (July, August and September)

filter(flights, month == 7 | month == 8 | month == 9)

(e) Arrived more than two hours late, but didn’t leave late

filter(flights,arr\_delay > 120 & dep\_delay <= 0)

(f) Were delayed by at least an hour, but made up over 30 minutes in flight

filter(flights, dep\_delay > 60 & air\_time > 30)

(g) Departed between midnight and 6am (inclusive)

filter(flights, hour >= 0 & hour <= 6)

(h) How many flights have a missing dep\_time? What other variables are missing? What might these rows represent?

filter(flights, is.na(dep\_time) )

**2. Arrange()**

(a) Sort flights to find the most delayed flights. Find the flights that left earliest

arrange(flights, desc(dep\_delay))

arrange(flights, (dep\_delay))

(b) Sort flights to find the fastest flights.

arrange(flights, air\_time)

(c) Which flights travelled the longest? Which travelled the shortest?

arrange(flights,desc(distance))

arrange(flights,distance)

**3. Select()**

**4. Mutate ()**

(a) Currently dep\_time and sched\_dep\_time are convenient to look at, but hard to compute with because they’re not really continuous numbers. Convert them to a more convenient representation of number of minutes since midnight.

transmute(flights,

dep\_time\_since\_midnight = (dep\_time %% 100) + ((dep\_time %/% 100) \* 60),

sched\_dep\_time\_since\_midnight = (sched\_dep\_time %% 100) + ((sched\_dep\_time %/% 100) \* 60)

)

(b) Compare air\_time with arr\_time - dep\_time. What do you expect to see? What do you see? What do you need to do to fix it?

I expect to see air\_time to equal arr\_time - dep\_time

transmute(flights,

air\_time,

tmp = arr\_time - dep\_time

)

transmute(flights,

air\_time,

arr\_minutes = (arr\_time %% 100) + ((arr\_time %/% 100) \* 60),

dep\_minutes = (dep\_time %% 100) + ((dep\_time %/% 100) \* 60),

arr\_dep\_minutes\_diff = arr\_minutes - dep\_minutes

)

(c) Compare dep\_time, sched\_dep\_time, and dep\_delay. How would you expect those three numbers to be related?

I would expect dep\_delay = dep\_time - sched\_dep\_time

transmute(flights,

dep\_delay,

theoretical\_dep\_time = dep\_time - sched\_dep\_time)

(d) Find the 10 most delayed flights using a ranking function. How do you want to handle ties? Carefully read the documentation for min\_rank().

mutate(flights,

dep\_delay\_rank = min\_rank(-dep\_delay)) %>%

arrange(dep\_delay\_rank)

(e) What does 1:3 + 1:10 return? Why?

1:3

1:10

1:3 + 1:10

Because 1:3 is short than 1:10, 1:3 is recycled and it goes 1+1,2+2,3+3,1+4,2+5,3+6, etc.

(f) What trigonometric functions does R provide?

R provides:

cos(x) sin(x) tan(x)

acos(x) asin(x) atan(x) atan2(y, x)

cospi(x) sinpi(x) tanpi(x)

From ? Trig, they respectively compute the cosine, sine, tangent, arc-cosine, arc-sine, arc-tangent, and the two-argument arc-tangent.

**5. Summarize (Group\_by)**

(b) Which carrier has the worst delays? Challenge: can you disentangle the effects of bad airports vs. bad carriers? Why/why not? (Hint: think about flights %>% group\_by(carrier, dest) %>% summarise(n())) So we want to group by carrier. And since we want to take into account the airport (dest), we can group by dest and carrier.

not\_cancelled <- flights %>%

filter(!is.na(dep\_delay), !is.na(arr\_delay))

tmp <- not\_cancelled %>% group\_by(carrier) %>% summarise( "mean\_delay" = mean(arr\_delay))

ggplot( data = tmp , mapping = aes(x = carrier, y = mean\_delay) ) +

geom\_bar(stat="identity")

(c) Look at the number of cancelled flights per day. Is there a pattern? Is the proportion of cancelled flights related to the average delay?

cancelled\_delayed <-

flights %>%

mutate(cancelled = (is.na(arr\_delay) | is.na(dep\_delay))) %>%

group\_by(year, month, day) %>%

summarise(prop\_cancelled = mean(cancelled),

avg\_dep\_delay = mean(dep\_delay, na.rm = TRUE))

ggplot(cancelled\_delayed, aes(x = avg\_dep\_delay, prop\_cancelled)) +

geom\_point() +

geom\_smooth()

**6. Misc**

1. Refer back to the lists of useful mutate and filtering functions. Describe how each operation changes when you combine it with grouping.

When you combine the mutate and filtering functions with groupings, you operate on the grouped data as opposed to the entire data frame.

2. Which plane (tailnum) has the worst on-time record?

not\_cancelled %>%

group\_by(tailnum) %>%

summarise(arr\_delay = mean(arr\_delay, na.rm = TRUE)) %>%

ungroup() %>%

filter(rank(desc(arr\_delay)) <= 1)

3. What time of day should you fly if you want to avoid delays as much as possible?

not\_cancelled %>%

group\_by(hour) %>%

summarise( mean\_arr\_delay = mean(arr\_delay, na.rm=T) ) %>%

ungroup() %>%

arrange(mean\_arr\_delay)

4. For each destination, compute the total minutes of delay. For each, flight, compute the proportion of the total delay for its destination.

not\_cancelled %>%

group\_by(dest) %>%

summarise( sum\_arr\_delay = sum(arr\_delay) )

not\_cancelled %>%

group\_by(flight) %>%

transmute( prop\_arr\_delay = arr\_delay / sum(arr\_delay) )

5. Delays are typically temporally correlated: even once the problem that caused the initial delay has been resolved, later flights are delayed to allow earlier flights to leave. Using lag() explore how the delay of a flight is related to the delay of the immediately preceding flight.

flights %>%

group\_by(year, month, day) %>%

filter(!is.na(dep\_delay)) %>%

mutate(lag\_delay = lag(dep\_delay)) %>%

filter(!is.na(lag\_delay)) %>%

ggplot(aes(x = dep\_delay, y = lag\_delay)) +

geom\_point() +

geom\_smooth()

6. Look at each destination. Can you find flights that are suspiciously fast? (i.e. flights that represent a potential data entry error). Compute the air time a flight relative to the shortest flight to that destination. Which flights were most delayed in the air?

I also computed an observed vs expected ratio using the median air time. I’m sure there’s better ways to find outliers.

flights %>%

filter(!is.na(air\_time)) %>%

group\_by(dest) %>%

mutate( med\_air\_time = median(air\_time),

o\_vs\_e = (air\_time - med\_air\_time) / med\_air\_time,

air\_time\_diff = air\_time - min(air\_time) ) %>%

arrange(desc(air\_time\_diff)) %>%

select(air\_time, o\_vs\_e, air\_time\_diff, dep\_time, sched\_dep\_time, arr\_time, sched\_arr\_time) %>%

head(15)

7. Find all destinations that are flown by at least two carriers. Use that information to rank the carriers.

not\_cancelled %>%

group\_by(dest, carrier) %>%

count(carrier) %>%

filter(n >= 2) %>%

group\_by(carrier) %>%

count(sort = TRUE)

8. For each plane, count the number of flights before the first delay of greater than 1 hour.

not\_cancelled %>%

group\_by(tailnum) %>%

mutate(delay\_gt1hr = dep\_delay > 60) %>%

mutate(before\_delay = cumsum(delay\_gt1hr)) %>%

filter(before\_delay < 1) %>%

count(sort = TRUE)