Math 300 NTI Lesson 5

filter and summarize

Professor Bradley Warner

June, 2022

Contents

Objectives

1. Use the `filter()` function and logical operators to subset a data frame.
2. Use the `summarize()` function with appropriate `R` functions to summarize variables in a data frame.
3. Explain the possible impacts of simply ignoring missing values.

Reading

Chapter 3 - 3.3

Lesson

Remember that you will be running this more like a lab than a lecture. You want them using `R` and answering questions. Have them open the notes rmd and work through it together.

Work through the learning checks LC3.1 - LC3.4.

- As you have probably heard: about 80% of your work analyzing data will be data acquisition and wrangling.
- The order of logical operations in the `filter()` function work from left to right. Parentheses, however, are executed first. Be careful and when in doubt use parentheses.
- Pay attention to functions that default an `NA` for variables with a missing value. Functions such as `mean()` and `sd()` are examples.
- This pdf summarizes the pipe operator and the main data wrangling functions. There should be time to discuss it in this lesson.

Setup

```
library(nycflights13)
library(ggplot2)
library(dplyr)
```

Notice that order of logical operators doesn't matter in this example but the parentheses help the reading of the code.

```
flights %>%
  filter(origin == "JFK", (dest == "BTV" | dest == "SEA"), month >= 10) %>%
  glimpse()
```

```
## Rows: 815
## Columns: 19
## $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2~
## $ month          <int> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,~
## $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ dep_time       <int> 729, 853, 916, 1216, 1452, 1459, 1754, 1825, 1925, 2238~
## $ sched_dep_time <int> 735, 900, 925, 1221, 1459, 1500, 1800, 1830, 1930, 2245~
## $ dep_delay      <dbl> -6, -7, -9, -5, -7, -1, -6, -5, -5, -7, 0, -2, -7, -4, ~
## $ arr_time       <int> 1049, 1217, 1016, 1326, 1602, 1817, 2102, 2159, 2227, 2~
## $ sched_arr_time <int> 1040, 1157, 1033, 1328, 1622, 1829, 2103, 2150, 2250, 2~
## $ arr_delay      <dbl> 9, 20, -17, -2, -20, -12, -1, 9, -23, -5, -14, 15, -7, ~
## $ carrier        <chr> "DL", "B6", "B6", "B6", "B6", "DL", "B6", "DL", "AA", "~
## $ flight         <int> 183, 63, 1634, 34, 1734, 161, 263, 442, 235, 234, 183, ~
## $ tailnum        <chr> "N721TW", "N807JB", "N192JB", "N318JB", "N258JB", "N169~
## $ origin         <chr> "JFK", "JFK", "JFK", "JFK", "JFK", "JFK", "JFK", "JFK",~
## $ dest           <chr> "SEA", "SEA", "BTV", "BTV", "BTV", "SEA", "SEA", "SEA",~
## $ air_time       <dbl> 352, 362, 48, 49, 46, 348, 338, 366, 332, 48, 330, 344,~
## $ distance       <dbl> 2422, 2422, 266, 266, 266, 2422, 2422, 2422, 2422, 266,~
## $ hour           <dbl> 7, 9, 9, 12, 14, 15, 18, 18, 19, 22, 7, 9, 9, 12, 14, 1~
## $ minute         <dbl> 35, 0, 25, 21, 59, 0, 0, 30, 30, 45, 35, 0, 25, 21, 59,~
## $ time_hour      <dttm> 2013-10-01 07:00:00, 2013-10-01 09:00:00, 2013-10-01 0~
```

```
flights %>%
  filter(origin == "JFK", dest == "BTV" | dest == "SEA", month >= 10) %>%
  glimpse()
```

```
## Rows: 815
## Columns: 19
## $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2~
## $ month          <int> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,~
## $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ dep_time       <int> 729, 853, 916, 1216, 1452, 1459, 1754, 1825, 1925, 2238~
## $ sched_dep_time <int> 735, 900, 925, 1221, 1459, 1500, 1800, 1830, 1930, 2245~
## $ dep_delay      <dbl> -6, -7, -9, -5, -7, -1, -6, -5, -5, -7, 0, -2, -7, -4, ~
## $ arr_time       <int> 1049, 1217, 1016, 1326, 1602, 1817, 2102, 2159, 2227, 2~
## $ sched_arr_time <int> 1040, 1157, 1033, 1328, 1622, 1829, 2103, 2150, 2250, 2~
## $ arr_delay      <dbl> 9, 20, -17, -2, -20, -12, -1, 9, -23, -5, -14, 15, -7, ~
## $ carrier        <chr> "DL", "B6", "B6", "B6", "B6", "DL", "B6", "DL", "AA", "~
## $ flight         <int> 183, 63, 1634, 34, 1734, 161, 263, 442, 235, 234, 183, ~
## $ tailnum        <chr> "N721TW", "N807JB", "N192JB", "N318JB", "N258JB", "N169~
## $ origin         <chr> "JFK", "JFK", "JFK", "JFK", "JFK", "JFK", "JFK", "JFK",~
## $ dest           <chr> "SEA", "SEA", "BTV", "BTV", "BTV", "SEA", "SEA", "SEA",~
## $ air_time       <dbl> 352, 362, 48, 49, 46, 348, 338, 366, 332, 48, 330, 344,~
## $ distance       <dbl> 2422, 2422, 266, 266, 266, 2422, 2422, 2422, 2422, 266,~
## $ hour           <dbl> 7, 9, 9, 12, 14, 15, 18, 18, 19, 22, 7, 9, 9, 12, 14, 1~
## $ minute         <dbl> 35, 0, 25, 21, 59, 0, 0, 30, 30, 45, 35, 0, 25, 21, 59,~
## $ time_hour      <dttm> 2013-10-01 07:00:00, 2013-10-01 09:00:00, 2013-10-01 0~
```

LC 3.1 (Objective 1)

(LC3.1) What's another way using the "not" operator `!` to filter only the rows that are not going to Burlington, VT nor Seattle, WA in the `flights` data frame? Test this out using the code above.

Solution:

```
# Original in book
not_BTV_SEA <- flights %>%
  filter(!(dest == "BTV" | dest == "SEA"))

# Alternative way
not_BTV_SEA <- flights %>%
  filter(!dest == "BTV" & !dest == "SEA")

# Yet another way
not_BTV_SEA <- flights %>%
  filter(dest != "BTV" & dest != "SEA")
```

LC 3.2 (Objective 3)

(LC3.2) Say a doctor is studying the effect of smoking on lung cancer for a large number of patients who have records measured at five year intervals. She notices that a large number of patients have missing data points because the patient has died, so she chooses to ignore these patients in her analysis. What is wrong with this doctor's approach?

Solution: The missing patients may have died of lung cancer! So to ignore them might seriously **bias** your results! It is very important to think of what the consequences on your analysis are of ignoring missing data! Ask yourself:

- There is a systematic reasons why certain values are missing? If so, you might be biasing your results!
- If there isn't, then it might be ok to "sweep missing values under the rug."

LC 3.3 (Objective 2)

```
summary_temp <- weather %>%
  summarize(mean = mean(temp, na.rm = TRUE),
            std_dev = sd(temp, na.rm = TRUE))
```

```
summary_temp
```

```
## # A tibble: 1 x 2
##    mean std_dev
##   <dbl>   <dbl>
## 1  55.3    17.8
```

(LC3.3) Modify the above `summarize` function to create `summary_temp` to also use the `n()` summary function: `summarize(count = n())`. What does the returned value correspond to?

Solution: It corresponds to a count of the number of observations/rows:

```
weather %>%
  summarize(count = n())
```

```
## # A tibble: 1 x 1
##   count
##   <int>
## 1 26115
```

LC 3.4 (Objective 2)

(LC3.4) Why doesn't the following code work? Run the code line by line instead of all at once, and then look at the data. In other words, run `summary_temp <- weather %>% summarize(mean = mean(temp, na.rm = TRUE))` first.

```
summary_temp <- weather %>%
  summarize(mean = mean(temp, na.rm = TRUE)) %>%
  summarize(std_dev = sd(temp, na.rm = TRUE))
```

Solution: Consider the output of only running the first two lines:

```
weather %>%
  summarize(mean = mean(temp, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##    mean
##   <dbl>
## 1  55.3
```

Because after the first `summarize()`, the variable `temp` disappears as it has been collapsed to the value `mean`. So when we try to run the second `summarize()`, it can't find the variable `temp` to compute the standard deviation of.

Documenting software

- File creation date: 2022-06-04
- R version 4.1.3 (2022-03-10)
- `ggplot2` package version: 3.3.6
- `dplyr` package version: 1.0.9
- `nycflights13` package version: 1.0.2