Math 300 NTI Lesson 7

join, select, rename, and top_n

Professor Bradley Warner

June, 2022

Contents

Objectives

1. Use the `inner_join()` function to combine data frames to use to explore, explain, and visualize.
2. Explain how to join data frame to include the use of keys and the advantages and the disadvantages of normal forms.
3. Use `rename()` and `select()` to reorganize data frames to use to explore, explain, and visualize. This includes all the ways to select columns including helper functions such as `everthing()`, `contains()`, etc.
4. Use `top_n()` to select a subset of the data frame.

Reading

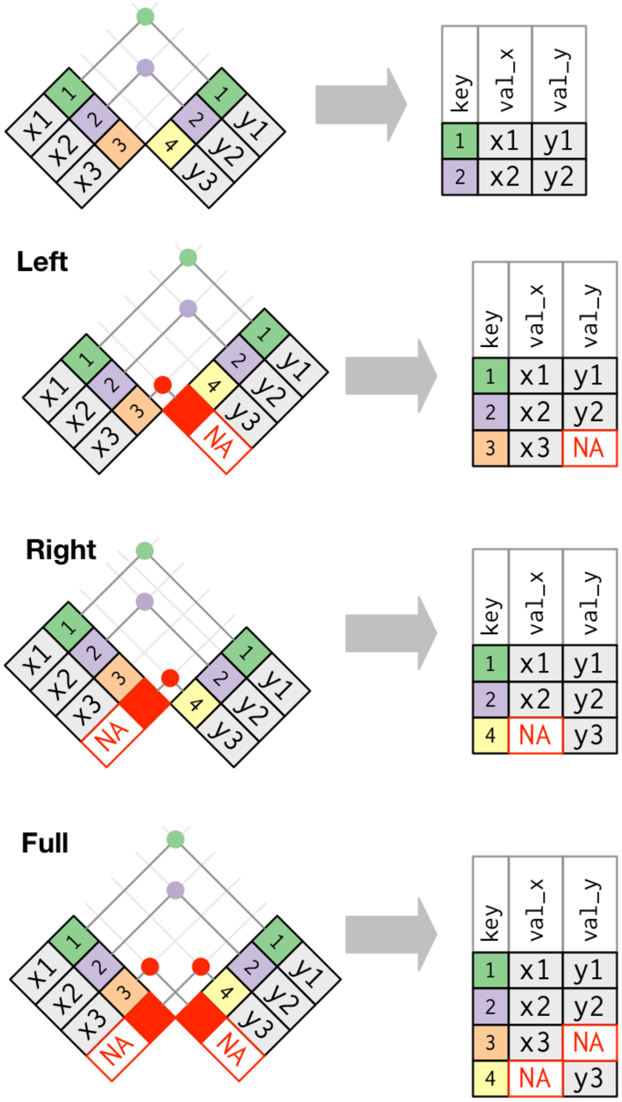
Chapter 3.7 - 3.9

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 LC3.13 - LC3.20.

- We are using an inner join. The important ideas are the **key** variable to join the data frames and the type of join. Figure 3.8 shows an inner join. It only keeps observations that are in each data frame. See R for Data Science for more information to include a discussion of outer joins. The following figures can help.



- The *keys* don't have to be called the same name and as such will require the use of the `by` option.
- The use of `everthing()`, `contains()`, `starts_with()`, and `ends_with()` makes the use of `select()` easier.
- `rename()` creates the new variables and deletes the old. If we used `mutate()` it would keep both variables and we would then need to use `select()`.
- Likewise, `top_n()` could be achieved with `arrange()` and `head()`.
- LC 3.20 is difficult.

Setup

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

LC 3.13 (Objective 2)

(LC3.13) Looking at Figure 3.7, when joining `flights` and `weather` (or, in other words, matching the hourly weather values with each flight), why do we need to join by all of `year`, `month`, `day`, `hour`, and `origin`, and not just `hour`?

Solution: Because `hour` is simply a value between 0 and 23; to identify a *specific* hour, we need to know which year, month, day and at which airport.

LC 3.14 (Objective 1, 3)

(LC3.14) Recreate the data object `named_dests` from the reading. What surprises you about the top 10 destinations from NYC in 2013?

Solution:

We need to recreate the data object

```
named_dests <- flights %>%
  group_by(dest) %>%
  summarize(num_flights = n()) %>%
  arrange(desc(num_flights)) %>%
  inner_join(airports, by = c("dest" = "faa")) %>%
  rename(airport_name = name)
```

```
head(named_dests,n=10)
```

```
## # A tibble: 10 x 9
##    dest  num_flights airport_name              lat    lon   alt    tz dst   tzone
##    <chr>       <int> <chr>                   <dbl>  <dbl> <dbl> <dbl> <chr> <chr>
##  1 ORD         17283 Chicago Ohare Intl       42.0  -87.9   668    -6 A     Amer~
##  2 ATL         17215 Hartsfield Jackson At~   33.6  -84.4  1026    -5 A     Amer~
##  3 LAX         16174 Los Angeles Intl         33.9 -118.    126    -8 A     Amer~
##  4 BOS         15508 General Edward Lawren~   42.4  -71.0    19    -5 A     Amer~
##  5 MCO         14082 Orlando Intl             28.4  -81.3    96    -5 A     Amer~
##  6 CLT         14064 Charlotte Douglas Intl   35.2  -80.9   748    -5 A     Amer~
##  7 SFO         13331 San Francisco Intl       37.6 -122.     13    -8 A     Amer~
##  8 FLL         12055 Fort Lauderdale Holly~   26.1  -80.2     9    -5 A     Amer~
##  9 MIA         11728 Miami Intl               25.8  -80.3     8    -5 A     Amer~
## 10 DCA          9705 Ronald Reagan Washing~   38.9  -77.0    15    -5 A     Amer~
```

This question is subjective! What surprises us is that very few flights, with the exception of Chicago, go to the middle of the country. Also, there are a high number of flights to Boston. New York and Boston are close to each other.

LC 3.15 (Objective 2)

(LC3.15) What are some advantages of data in normal forms? What are some disadvantages?

Solution: When datasets are in normal form, we can easily `_join` them with other datasets! For example, we can join the `flights` data with the `planes` data. Using normal forms keeps the size of files down. For example airlines is only 16 rows. The size of flights is 336,776 and we repeated the airline names over and over in this file, its size could increase.

LC 3.16 (Objective 3)

(LC3.16) What are some ways to select all three of the `dest`, `air_time`, and `distance` variables from `flights`? Give the code showing how to do this in at least three different ways.

Solution:

```
# The regular way:
flights %>%
  select(dest, air_time, distance) %>%
  head()
```

```
## # A tibble: 6 x 3
##   dest  air_time distance
##   <chr>    <dbl>    <dbl>
## 1 IAH        227     1400
## 2 IAH        227     1416
## 3 MIA        160     1089
## 4 BQN        183     1576
## 5 ATL        116      762
## 6 ORD        150      719
```

```
# Since they are sequential columns in the dataset
flights %>%
  select(dest:distance) %>%
  head()
```

```
## # A tibble: 6 x 3
##   dest  air_time distance
##   <chr>    <dbl>    <dbl>
## 1 IAH        227     1400
## 2 IAH        227     1416
## 3 MIA        160     1089
## 4 BQN        183     1576
## 5 ATL        116      762
## 6 ORD        150      719
```

```
# Not as effective, by removing everything else
flights %>%
  select(
    -year, -month, -day, -dep_time, -sched_dep_time, -dep_delay, -arr_time,
    -sched_arr_time, -arr_delay, -carrier, -flight, -tailnum, -origin,
    -hour, -minute, -time_hour
  ) %>%
  head()
```

```
## # A tibble: 6 x 3
##   dest  air_time distance
##   <chr>    <dbl>    <dbl>
## 1 IAH        227     1400
## 2 IAH        227     1416
## 3 MIA        160     1089
## 4 BQN        183     1576
## 5 ATL        116      762
## 6 ORD        150      719
```

LC 3.17 (Objective 3)

(LC3.17) How could one use `starts_with`, `ends_with`, and `contains` to select columns from the `flights` data frame? Provide three different examples in total: one for `starts_with`, one for `ends_with`, and one for `contains`.

Solution:

```r
# Anything that starts with "d"
flights %>%
  select(starts_with("d")) %>%
  head()
```

```
## # A tibble: 6 x 5
##     day dep_time dep_delay dest  distance
##   <int>    <int>     <dbl> <chr>    <dbl>
## 1     1      517         2 IAH       1400
## 2     1      533         4 IAH       1416
## 3     1      542         2 MIA       1089
## 4     1      544        -1 BQN       1576
## 5     1      554        -6 ATL        762
## 6     1      554        -4 ORD        719
```

```r
# Anything related to delays:
flights %>%
  select(ends_with("delay")) %>%
  head()
```

```
## # A tibble: 6 x 2
##   dep_delay arr_delay
##       <dbl>     <dbl>
## 1         2        11
## 2         4        20
## 3         2        33
## 4        -1       -18
## 5        -6       -25
## 6        -4        12
```

```r
# Anything related to departures:
flights %>%
  select(contains("dep")) %>%
  head()
```

```
## # A tibble: 6 x 3
##   dep_time sched_dep_time dep_delay
##      <int>          <int>     <dbl>
## 1      517            515         2
## 2      533            529         4
## 3      542            540         2
## 4      544            545        -1
## 5      554            600        -6
## 6      554            558        -4
```

LC 3.18 (Objective 3)

(LC3.18) Why might we want to use the `select()` function on a data frame?

Solution: To narrow down the data frame, to make it easier to view or print such as using `View()` for example.

LC 3.19 (Objective 4)

(LC3.19) Create a new data frame that shows the top 5 airports with the largest average arrival delays from NYC in 2013.

Solution:

```
top_five <- flights %>%
  group_by(dest) %>%
  summarize(avg_delay = mean(arr_delay, na.rm = TRUE)) %>%
  arrange(desc(avg_delay)) %>%
  top_n(n = 5)
```

```
## Selecting by avg_delay
```

```
top_five
```

```
## # A tibble: 5 x 2
##   dest  avg_delay
##   <chr>     <dbl>
## 1 CAE        41.8
## 2 TUL        33.7
## 3 OKC        30.6
## 4 JAC        28.1
## 5 TYS        24.1
```

LC 3.20 (Many objectives of Chapter 3)

(LC3.20) Using the datasets included in the `nycflights13` package, compute the available seat miles for each airline sorted in descending order. After completing all the necessary data wrangling steps, the resulting data frame should have 16 rows (one for each airline) and 2 columns (airline name and available seat miles). Here are some hints:

- **Crucial**: Unless you are very confident in what you are doing, it is worthwhile to not starting coding right away, but rather first sketch out on paper all the necessary data wrangling steps not using exact code, but rather high-level *pseudocode* that is informal yet detailed enough to articulate what you are doing. This way you won't confuse *what* you are trying to do (the algorithm) with *how* you are going to do it (writing `dplyr` code).
- Take a close look at all the datasets using the `View()` function: `flights`, `weather`, `planes`, `airports`, and `airlines` to identify which variables are necessary to compute available seat miles.
- Figure 3.7 above showing how the various datasets can be joined will also be useful.
- Consider the data wrangling verbs in Table 3.2 as your toolbox!

Solution:

We need the number of seats, number of flights, miles, and airline. The number of seats in a plane is in the `planes` data frame and we can use `tailnum` as a key. The data frame `flights` the `carrier` and distance flown `distance`. It also has the `tailnum` for a key back to plane. The data frame `airline` has the carrier code and full airline name.

Psuedo code

- Subset the flights data frame to include `distance`, `tailnum`, and `carrier`,

- Join the result `planes` with only `tailnum` and `seats` which requires a select within the `inner_join()` function,
- Join with `airlines`,
- Create available seat miles variable,
- Next group by carrier,
- Sort in descending

Let's piece it together.

```
flights %>%
  select(carrier,distance,tailnum) %>% head()
```

```
## # A tibble: 6 x 3
##   carrier distance tailnum
##   <chr>      <dbl> <chr>
## 1 UA          1400 N14228
## 2 UA          1416 N24211
## 3 AA          1089 N619AA
## 4 B6          1576 N804JB
## 5 DL           762 N668DN
## 6 UA           719 N39463
```

```
flights %>%
  select(carrier,distance,tailnum) %>%
  inner_join(select(planes,seats,tailnum)) %>% head()
```

```
## Joining, by = "tailnum"
```

```
## # A tibble: 6 x 4
##   carrier distance tailnum seats
##   <chr>      <dbl> <chr>   <int>
## 1 UA          1400 N14228    149
## 2 UA          1416 N24211    149
## 3 AA          1089 N619AA    178
## 4 B6          1576 N804JB    200
## 5 DL           762 N668DN    178
## 6 UA           719 N39463    191
```

```
flights %>%
  select(carrier,distance,tailnum) %>%
  inner_join(select(planes,seats,tailnum)) %>%
```

```
  inner_join(airlines) %>%
  mutate(asm=distance*seats) %>%
  select(name,carrier,asm,distance,seats) %>%
  head()
```

```
## Joining, by = "tailnum"
## Joining, by = "carrier"
```

```
## # A tibble: 6 x 5
##   name                   carrier    asm distance seats
##   <chr>                  <chr>    <dbl>    <dbl> <int>
## 1 United Air Lines Inc.  UA      208600     1400   149
## 2 United Air Lines Inc.  UA      210984     1416   149
## 3 American Airlines Inc. AA      193842     1089   178
## 4 JetBlue Airways        B6      315200     1576   200
## 5 Delta Air Lines Inc.   DL      135636      762   178
## 6 United Air Lines Inc.  UA      137329      719   191
```

```
flights %>%
  select(carrier,distance,tailnum) %>%
  inner_join(select(planes,seats,tailnum)) %>%
  inner_join(airlines) %>%
  mutate(asm=distance*seats) %>%
  select(name,carrier,asm,distance,seats) %>%
  group_by(name) %>%
  summarize(total_asm=sum(asm)) %>%
  arrange(desc(total_asm))
```

```
## Joining, by = "tailnum"
## Joining, by = "carrier"
```

```
## # A tibble: 16 x 2
##    name                           total_asm
##    <chr>                              <dbl>
##  1 United Air Lines Inc.        15516377526
##  2 Delta Air Lines Inc.         10532885801
##  3 JetBlue Airways               9618222135
##  4 American Airlines Inc.        3677292231
##  5 US Airways Inc.               2533505829
##  6 Virgin America                2296680778
##  7 ExpressJet Airlines Inc.      1817236275
##  8 Southwest Airlines Co.        1718116857
##  9 Endeavor Air Inc.              776970310
## 10 Hawaiian Airlines Inc.         642478122
## 11 Alaska Airlines Inc.           314104736
## 12 AirTran Airways Corporation    219628520
## 13 Frontier Airlines Inc.         184832280
## 14 Mesa Airlines Inc.              20163632
## 15 Envoy Air                        7162420
## 16 SkyWest Airlines Inc.            1299835
```

Here is the author's code.

```
flights %>%
  inner_join(planes, by = "tailnum") %>%
  select(carrier, seats, distance) %>%
  mutate(ASM = seats * distance) %>%
  group_by(carrier) %>%
  summarize(ASM = sum(ASM, na.rm = TRUE)) %>%
  arrange(desc(ASM))
```

```
## # A tibble: 16 x 2
##    carrier         ASM
##    <chr>         <dbl>
##  1 UA      15516377526
##  2 DL      10532885801
##  3 B6       9618222135
##  4 AA       3677292231
##  5 US       2533505829
##  6 VX       2296680778
##  7 EV       1817236275
##  8 WN       1718116857
##  9 9E        776970310
## 10 HA        642478122
## 11 AS        314104736
## 12 FL        219628520
## 13 F9        184832280
## 14 YV         20163632
## 15 MQ          7162420
## 16 OO          1299835
```

Let's now break this down step-by-step. To compute the available seat miles for a given flight, we need the `distance` variable from the `flights` data frame and the `seats` variable from the `planes` data frame, necessitating a join by the key variable `tailnum`. To keep the resulting data frame easy to view, we'll `select()` only these two variables and `carrier` and use `head()` to keep only the first 6 rows:

```
flights %>%
  inner_join(planes, by = "tailnum") %>%
  select(carrier, seats, distance) %>%
  head()
```

```
## # A tibble: 6 x 3
##   carrier seats distance
##   <chr>   <int>    <dbl>
## 1 UA        149     1400
## 2 UA        149     1416
## 3 AA        178     1089
## 4 B6        200     1576
## 5 DL        178      762
## 6 UA        191      719
```

Now for each flight we can compute the available seat miles `ASM` by multiplying the number of seats by the distance via a `mutate()`:

```r
flights %>%
  inner_join(planes, by = "tailnum") %>%
  select(carrier, seats, distance) %>%
  # Added:
  mutate(ASM = seats * distance) %>%
  head()
```

```
## # A tibble: 6 x 4
##   carrier seats distance    ASM
##   <chr>   <int>    <dbl>  <dbl>
## 1 UA        149     1400 208600
## 2 UA        149     1416 210984
## 3 AA        178     1089 193842
## 4 B6        200     1576 315200
## 5 DL        178      762 135636
## 6 UA        191      719 137329
```

Next we want to sum the `ASM` for each carrier. We achieve this by first grouping by `carrier` and then summarizing using the `sum()` function:

```r
flights %>%
  inner_join(planes, by = "tailnum") %>%
  select(carrier, seats, distance) %>%
  mutate(ASM = seats * distance) %>%
  # Added:
  group_by(carrier) %>%
  summarize(ASM = sum(ASM)) %>%
  head()
```

```
## # A tibble: 6 x 2
##   carrier         ASM
##   <chr>         <dbl>
## 1 9E        776970310
## 2 AA       3677292231
## 3 AS        314104736
## 4 B6       9618222135
## 5 DL      10532885801
## 6 EV       1817236275
```

However, because for certain carriers certain flights have missing `NA` values, the resulting table also returns `NA`'s. We can eliminate these by adding a `na.rm = TRUE` argument to `sum()`, telling R that we want to remove the `NA`'s in the sum.

```r
flights %>%
  inner_join(planes, by = "tailnum") %>%
  select(carrier, seats, distance) %>%
  mutate(ASM = seats * distance) %>%
  group_by(carrier) %>%
  # Modified:
  summarize(ASM = sum(ASM, na.rm = TRUE))
```

```
## # A tibble: 16 x 2
```

```
##    carrier         ASM
##    <chr>         <dbl>
##  1 9E        776970310
##  2 AA       3677292231
##  3 AS        314104736
##  4 B6       9618222135
##  5 DL      10532885801
##  6 EV       1817236275
##  7 F9        184832280
##  8 FL        219628520
##  9 HA        642478122
## 10 MQ          7162420
## 11 OO          1299835
## 12 UA      15516377526
## 13 US       2533505829
## 14 VX       2296680778
## 15 WN       1718116857
## 16 YV         20163632
```

Finally, we `arrange()` the data in `desc()`ending order of `ASM`.

```
flights %>%
  inner_join(planes, by = "tailnum") %>%
  select(carrier, seats, distance) %>%
  mutate(ASM = seats * distance) %>%
  group_by(carrier) %>%
  summarize(ASM = sum(ASM, na.rm = TRUE)) %>%
  # Added:
  arrange(desc(ASM))
```

```
## # A tibble: 16 x 2
##    carrier         ASM
##    <chr>         <dbl>
##  1 UA      15516377526
##  2 DL      10532885801
##  3 B6       9618222135
##  4 AA       3677292231
##  5 US       2533505829
##  6 VX       2296680778
##  7 EV       1817236275
##  8 WN       1718116857
##  9 9E        776970310
## 10 HA        642478122
## 11 AS        314104736
## 12 FL        219628520
## 13 F9        184832280
## 14 YV         20163632
## 15 MQ          7162420
## 16 OO          1299835
```

While the above data frame is correct, the IATA `carrier` code is not always useful. For example, what carrier is `WN`? We can address this by joining with the `airlines` dataset using `carrier` is the key variable. While this step is not absolutely required, it goes a long way to making the table easier to make sense of. It is important to be empathetic with the ultimate consumers of your presented data!

```
flights %>%
  inner_join(planes, by = "tailnum") %>%
  select(carrier, seats, distance) %>%
  mutate(ASM = seats * distance) %>%
  group_by(carrier) %>%
  summarize(ASM = sum(ASM, na.rm = TRUE)) %>%
  arrange(desc(ASM)) %>%
  # Added:
  inner_join(airlines, by = "carrier")
```

```
## # A tibble: 16 x 3
##    carrier         ASM name
##    <chr>         <dbl> <chr>
##  1 UA      15516377526 United Air Lines Inc.
##  2 DL      10532885801 Delta Air Lines Inc.
##  3 B6       9618222135 JetBlue Airways
##  4 AA       3677292231 American Airlines Inc.
##  5 US       2533505829 US Airways Inc.
##  6 VX       2296680778 Virgin America
##  7 EV       1817236275 ExpressJet Airlines Inc.
##  8 WN       1718116857 Southwest Airlines Co.
##  9 9E        776970310 Endeavor Air Inc.
## 10 HA        642478122 Hawaiian Airlines Inc.
## 11 AS        314104736 Alaska Airlines Inc.
## 12 FL        219628520 AirTran Airways Corporation
## 13 F9        184832280 Frontier Airlines Inc.
## 14 YV         20163632 Mesa Airlines Inc.
## 15 MQ          7162420 Envoy Air
## 16 OO          1299835 SkyWest Airlines Inc.
```

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