Data Wrangling II

Ayush Jain

1-25-2022

Main Ideas

- To answer questions with data, we often need to use related data from many different datasets.
- We can combine data from different sources using a well-chosen join function.

Coming Up

- Homework #01 due Thursday.
- Lab #03 due Friday

Lecture Notes and Exercises

```
library(tidyverse)
```

Instead of working with a single dataset, usually you will have to work with many different related datasets. To answer research questions using relateddatasets, we need to develop tools to join datasets together.

There are many possible types of joins. All have the format `something_join(x, y)`.

- `inner_join()`: join all rows from `x` where there are matching values in `y`. Return all combinations in case of multiple matches
- `left_join()`: include all rows from `x`
- `right_join()`: include all rows from `y`
- `full_join()`: include all rows in `x` or `y`
- `semi_join()`: return all rows from `x` with match in `y`
- `anti_join()`: return all rows from `x` without a match in `y`

```
x <- tibble(value = c(1, 2, 3),
            xcol = c("x1", "x2", "x3"))
y <- tibble(value = c(1, 2, 4),
            ycol = c("y1", "y2", "y4"))
x
```

```
## # A tibble: 3 x 2
##   value xcol
##   <dbl> <chr>
```

```
## 1     1 x1
## 2     2 x2
## 3     3 x3
```

```r
y
```

```
## # A tibble: 3 x 2
##   value ycol
##   <dbl> <chr>
## 1     1 y1
## 2     2 y2
## 3     4 y4
```

We will demonstrate each of the joins on these small, toy datasets.

```r
x
```

```
## # A tibble: 3 x 2
##   value xcol
##   <dbl> <chr>
## 1     1 x1
## 2     2 x2
## 3     3 x3
```

```r
y
```

```
## # A tibble: 3 x 2
##   value ycol
##   <dbl> <chr>
## 1     1 y1
## 2     2 y2
## 3     4 y4
```

```r
inner_join(x, y)
```

```
## Joining, by = "value"
```

```
## # A tibble: 2 x 3
##   value xcol  ycol
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
```

```r
x
```

```
## # A tibble: 3 x 2
##   value xcol
##   <dbl> <chr>
## 1     1 x1
## 2     2 x2
## 3     3 x3
```

```r
y
```

```
## # A tibble: 3 x 2
##   value ycol
##   <dbl> <chr>
## 1     1 y1
## 2     2 y2
## 3     4 y4
```

```r
left_join(x, y)
```

```
## Joining, by = "value"
```

```
## # A tibble: 3 x 3
##   value xcol  ycol
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
## 3     3 x3    <NA>
```

```r
x
```

```
## # A tibble: 3 x 2
##   value xcol
##   <dbl> <chr>
## 1     1 x1
## 2     2 x2
## 3     3 x3
```

```r
y
```

```
## # A tibble: 3 x 2
##   value ycol
##   <dbl> <chr>
## 1     1 y1
## 2     2 y2
## 3     4 y4
```

```r
right_join(x, y)
```

```
## Joining, by = "value"
```

```
## # A tibble: 3 x 3
##   value xcol  ycol
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
## 3     4 <NA>  y4
```

```
x
```

```
## # A tibble: 3 x 2
##   value xcol
##   <dbl> <chr>
## 1     1 x1
## 2     2 x2
## 3     3 x3
```

```
y
```

```
## # A tibble: 3 x 2
##   value ycol
##   <dbl> <chr>
## 1     1 y1
## 2     2 y2
## 3     4 y4
```

```
full_join(x, y)
```

```
## Joining, by = "value"
```

```
## # A tibble: 4 x 3
##   value xcol  ycol
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
## 3     3 x3    <NA>
## 4     4 <NA>  y4
```

```
x
```

```
## # A tibble: 3 x 2
##   value xcol
##   <dbl> <chr>
## 1     1 x1
## 2     2 x2
## 3     3 x3
```

```
y
```

```
## # A tibble: 3 x 2
##   value ycol
##   <dbl> <chr>
## 1     1 y1
## 2     2 y2
## 3     4 y4
```

```
semi_join(x, y)
```

```
## Joining, by = "value"

## # A tibble: 2 x 2
##   value xcol
##   <dbl> <chr>
## 1     1 x1
## 2     2 x2
```

```
x
```

```
## # A tibble: 3 x 2
##   value xcol
##   <dbl> <chr>
## 1     1 x1
## 2     2 x2
## 3     3 x3
```

```
y
```

```
## # A tibble: 3 x 2
##   value ycol
##   <dbl> <chr>
## 1     1 y1
## 2     2 y2
## 3     4 y4
```

```
anti_join(x, y)
```

```
## Joining, by = "value"

## # A tibble: 1 x 2
##   value xcol
##   <dbl> <chr>
## 1     3 x3
```

How do the join functions above know to join `x` and `y` by `value`? Examine the names to find out.

```
names(x)
```

```
## [1] "value" "xcol"
```

```
names(y)
```

```
## [1] "value" "ycol"
```

We will again work with data from the `nycflights13` package.

```
library(nycflights13)
```

Examine the documentation for the datasets `airports`, `flights`, and `planes`.

Question: How are these datasets related? Suppose you wanted to make a map of the route of every flight. What variables would you need from which datasets? Answer: From airports, you would need name, lat, lon. From flights, you would need origin, dest, and distance.

Join flights to airports. Note these two datasets have no variables in common so we will have to specify the variable to join by using `by =`. Check out the documentation for more information.

```
flights %>%
  left_join(airports, by = c("dest" = "faa"))
```

```
## # A tibble: 336,776 x 26
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # ... with 336,766 more rows, and 18 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>,
## #   name <chr>, lat <dbl>, lon <dbl>, alt <dbl>, tz <dbl>, dst <chr>,
## #   tzone <chr>
```

Practice

(1) Create a new dataset `dest_delays` with the median arrival delay for each destination. Note this question does not require you to use joins.

```
dest_delays <-
  flights %>%
  group_by(dest) %>%
  summarise(median_arr_delay = median(arr_delay, na.rm= TRUE))
```

(2) Create a new dataset by joining `dest_delays` and `airports`. Only include observations that have both delay and airport information. Note `dest_delays` and `flights` have no variables in common so you will need to specify the variables to join using `by` as in the example above.

```
dest_delays %>%
  inner_join(airports, by = c("dest" = "faa"))
```

```
## # A tibble: 101 x 9
##    dest  median_arr_delay name             lat    lon   alt    tz dst   tzone
##    <chr>            <dbl> <chr>          <dbl>  <dbl> <dbl> <dbl> <chr> <chr>
##  1 ABQ               -5.5 Albuquerque I~  35.0 -107.   5355    -7 A     America~
```

```
##   2 ACK                 -3    Nantucket Mem   41.3  -70.1     48     -5 A      America~
##   3 ALB                 -4    Albany Intl     42.7  -73.8    285     -5 A      America~
##   4 ANC                  1.5  Ted Stevens A~  61.2 -150.     152     -9 A      America~
##   5 ATL                 -1    Hartsfield Ja~  33.6  -84.4   1026     -5 A      America~
##   6 AUS                 -5    Austin Bergst~  30.2  -97.7    542     -6 A      America~
##   7 AVL                 -1    Asheville Reg~  35.4  -82.5   2165     -5 A      America~
##   8 BDL                -10    Bradley Intl    41.9  -72.7    173     -5 A      America~
##   9 BGR                 -9    Bangor Intl     44.8  -68.8    192     -5 A      America~
##  10 BHM                 -2    Birmingham In~  33.6  -86.8    644     -6 A      America~
## # ... with 91 more rows
```

Question: Are all of the variables in `dest_delays` included in the new dataset you created by joining `dest_delays` and `airports`? Use an appropriate join function to investigate this issue and determine what is going on here. Answer: No, there are 4 rows that are in dest_delays but not in airports

Use an `anti_join` to help diagnose this issue. Recall `anti_join` returns all rows from `x` without a match in `y`, so it will return all rows in `dest_delays` that don't have a match in `airports`.

```
dest_delays %>%
  anti_join(airports, by = c("dest" = "faa"))
```

```
## # A tibble: 4 x 2
##   dest  median_arr_delay
##   <chr>            <dbl>
## 1 BQN                 -1
## 2 PSE                  0
## 3 SJU                 -6
## 4 STT                 -9
```

(3) Is there a relationship between the age of a plane and its delays? The plane tail number is given in the `tailnum` variable in the `flights` dataset. The year the plane was manufactured is given in the `year` variable in the `planes` dataset.

- Step #1: Start by finding the average arrival delay for each plane and store the resulting dataset in `plane_delays`.

```
plane_delays <- flights %>%
  group_by(tailnum) %>%
  summarise(mean_arr_delay = mean(arr_delay, na.rm = TRUE))
```

- Step #2: Join `plane_delays` to the `planes` data using an appropriate join and then use mutate to create an age variable. Note this data is from 2013.

```
plane_delays %>%
  left_join(planes, by = "tailnum") %>%
  mutate(age = 2013 - year)
```

```
## # A tibble: 4,044 x 11
##    tailnum mean_arr_delay  year type     manufacturer  model engines seats speed
##    <chr>            <dbl> <int> <chr>    <chr>         <chr>   <int> <int> <int>
##  1 D942DN            31.5    NA <NA>     <NA>          <NA>       NA    NA    NA
```

```
## 2 N0EGMQ          9.98     NA <NA>     <NA>           <NA>        NA    NA    NA
## 3 N10156         12.7    2004 Fixed w~ EMBRAER        EMB-~~       2    55    NA
## 4 N102UW          2.94   1998 Fixed w~ AIRBUS INDUS~ A320-~        2   182    NA
## 5 N103US         -6.93   1999 Fixed w~ AIRBUS INDUS~ A320-~        2   182    NA
## 6 N104UW          1.80   1999 Fixed w~ AIRBUS INDUS~ A320-~        2   182    NA
## 7 N10575         20.7    2002 Fixed w~ EMBRAER        EMB-~~       2    55    NA
## 8 N105UW         -0.267  1999 Fixed w~ AIRBUS INDUS~ A320-~        2   182    NA
## 9 N107US         -5.73   1999 Fixed w~ AIRBUS INDUS~ A320-~        2   182    NA
## 10 N108UW        -1.25   1999 Fixed w~ AIRBUS INDUS~ A320-~        2   182    NA
## # ... with 4,034 more rows, and 2 more variables: engine <chr>, age <dbl>
```

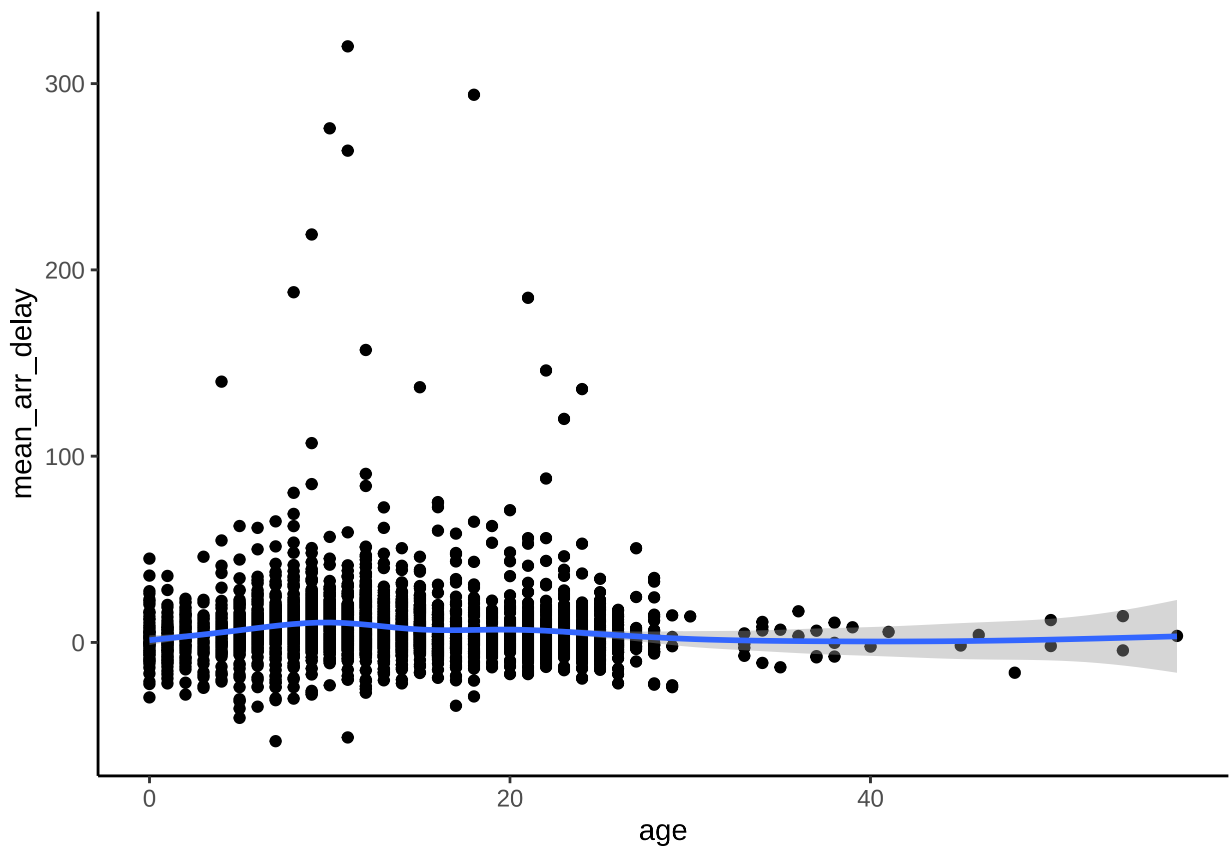
- Step #3: Finally, create an effective visualization of the data.

```
plane_delays %>%
  left_join(planes, by = "tailnum") %>%
  mutate(age = 2013 - year) %>%
  ggplot(aes(x = age, y = mean_arr_delay)) +
  geom_point() +
  geom_smooth() +
  theme_classic()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

## Warning: Removed 798 rows containing non-finite values (stat_smooth).

## Warning: Removed 798 rows containing missing values (geom_point).
```



Additional Resources

- https://rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf