Relational Data with dplyr (joins)

J. Wall

March 20, 2020

Learning Outcomes

- mutating joins - add new variables to one tibble by matching observations
- filtering joins - filter observations from one tibble based on whether or not they match an observation in another tibble
- set operations - treat observations as if they were set elements

Resources:

- r4ds book chapter
- dplyr cheatsheet
- youtube video

How to find the datasets and function available in a package?

```
library(nycflights13)
d <- data(package = "nycflights13")
d$results[,c("Item","Title")]
```

```
##      Item       Title
## [1,] "airlines" "Airline names."
## [2,] "airports" "Airport metadata"
## [3,] "flights"  "Flights data"
## [4,] "planes"   "Plane metadata."
## [5,] "weather"  "Hourly weather data"
```

```
dplyr_functions <- lsf.str("package:dplyr", all = TRUE)

airlines
```

```
## # A tibble: 16 x 2
##    carrier name
##    <chr>   <chr>
##  1 9E      Endeavor Air Inc.
##  2 AA      American Airlines Inc.
##  3 AS      Alaska Airlines Inc.
##  4 B6      JetBlue Airways
##  5 DL      Delta Air Lines Inc.
##  6 EV      ExpressJet Airlines Inc.
##  7 F9      Frontier Airlines Inc.
##  8 FL      AirTran Airways Corporation
##  9 HA      Hawaiian Airlines Inc.
## 10 MQ      Envoy Air
## 11 OO      SkyWest Airlines Inc.
## 12 UA      United Air Lines Inc.
```

```
## 13 US      US Airways Inc.
## 14 VX      Virgin America
## 15 WN      Southwest Airlines Co.
## 16 YV      Mesa Airlines Inc.
```

```
flights
```

```
## # A tibble: 336,776 x 19
##     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 11 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>
```

```
planes
```

```
## # A tibble: 3,322 x 9
##    tailnum  year type          manufacturer   model  engines seats speed engine
##    <chr>   <int> <chr>         <chr>          <chr>    <int> <int> <int> <chr>
##  1 N10156   2004 Fixed wing m~ EMBRAER        EMB-1~       2    55    NA Turbo-~
##  2 N102UW   1998 Fixed wing m~ AIRBUS INDUST~ A320-~       2   182    NA Turbo-~
##  3 N103US   1999 Fixed wing m~ AIRBUS INDUST~ A320-~       2   182    NA Turbo-~
##  4 N104UW   1999 Fixed wing m~ AIRBUS INDUST~ A320-~       2   182    NA Turbo-~
##  5 N10575   2002 Fixed wing m~ EMBRAER        EMB-1~       2    55    NA Turbo-~
##  6 N105UW   1999 Fixed wing m~ AIRBUS INDUST~ A320-~       2   182    NA Turbo-~
##  7 N107US   1999 Fixed wing m~ AIRBUS INDUST~ A320-~       2   182    NA Turbo-~
##  8 N108UW   1999 Fixed wing m~ AIRBUS INDUST~ A320-~       2   182    NA Turbo-~
##  9 N109UW   1999 Fixed wing m~ AIRBUS INDUST~ A320-~       2   182    NA Turbo-~
## 10 N110UW   1999 Fixed wing m~ AIRBUS INDUST~ A320-~       2   182    NA Turbo-~
## # ... with 3,312 more rows
```

```
weather
```

```
## # A tibble: 26,115 x 15
##    origin  year month   day  hour  temp  dewp humid wind_dir wind_speed
##    <chr>  <int> <int> <int> <int> <dbl> <dbl> <dbl>    <dbl>      <dbl>
##  1 EWR     2013     1     1     1  39.0  26.1  59.4      270      10.4
##  2 EWR     2013     1     1     2  39.0  27.0  61.6      250       8.06
##  3 EWR     2013     1     1     3  39.0  28.0  64.4      240      11.5
##  4 EWR     2013     1     1     4  39.9  28.0  62.2      250      12.7
##  5 EWR     2013     1     1     5  39.0  28.0  64.4      260      12.7
##  6 EWR     2013     1     1     6  37.9  28.0  67.2      240      11.5
##  7 EWR     2013     1     1     7  39.0  28.0  64.4      240      15.0
##  8 EWR     2013     1     1     8  39.9  28.0  62.2      250      10.4
##  9 EWR     2013     1     1     9  39.9  28.0  62.2      260      15.0
## 10 EWR     2013     1     1    10  41    28.0  59.6      260      13.8
## # ... with 26,105 more rows, and 5 more variables: wind_gust <dbl>,
```

```
## #   precip <dbl>, pressure <dbl>, visib <dbl>, time_hour <dttm>
```

Keys

Joins use Keys or variable names that are the same between datasets and that combine to uniquely identify an observation. Keys:

- a primary key uniquely identifies an observation its own table - can be one variable or the combination of several variables
- a foreign key uniquely identifies an observation in another table

To be sure the identification is unique, count the primary keys and see if any are greater than 1

```
planes %>%
  count(tailnum) %>%
  filter(n>1)
```

```
## # A tibble: 0 x 2
## # ... with 2 variables: tailnum <chr>, n <int>
```

What is the primary key in the flights tibble?

```
flights %>%
  count(year, month, day, flight) %>%
  filter(n>1)
```

```
## # A tibble: 29,768 x 5
##     year month   day flight     n
##    <int> <int> <int>  <int> <int>
##  1  2013     1     1      1     2
##  2  2013     1     1      3     2
##  3  2013     1     1      4     2
##  4  2013     1     1     11     3
##  5  2013     1     1     15     2
##  6  2013     1     1     21     2
##  7  2013     1     1     27     4
##  8  2013     1     1     31     2
##  9  2013     1     1     32     2
## 10  2013     1     1     35     2
## # ... with 29,758 more rows
```

```
flights%>%
  count(year, month, day, tailnum) %>%
  filter(n>1)
```

```
## # A tibble: 64,928 x 5
##     year month   day tailnum     n
##    <int> <int> <int> <chr>   <int>
##  1  2013     1     1 NOEGMQ      2
##  2  2013     1     1 N11189      2
##  3  2013     1     1 N11536      2
##  4  2013     1     1 N11544      3
##  5  2013     1     1 N11551      2
##  6  2013     1     1 N12540      2
##  7  2013     1     1 N12567      2
##  8  2013     1     1 N13123      2
##  9  2013     1     1 N13538      3
```

```
## 10  2013     1     1 N13566      3
## # ... with 64,918 more rows
```

Add a surrogate key to flights using mutate and row_number()

```
flightskey <- flights %>%
  mutate(rownum = row_number()) %>%
  select(rownum, everything())
```

- **Exercise 1:** Identify the primary keys in the following datasets (1.5 points: 1/2 pt. for each dataset). Be sure to show that you have the primary key by showing there are no duplicate entries.
  - Lahman::Batting
  - babynames::babynames
  - nasaweather::atmos

```
## # A tibble: 6 x 5
##    year sex   name          n   prop
##   <dbl> <chr> <chr>     <int>  <dbl>
## 1  1880 F     Mary       7065 0.0724
## 2  1880 F     Anna       2604 0.0267
## 3  1880 F     Emma       2003 0.0205
## 4  1880 F     Elizabeth  1939 0.0199
## 5  1880 F     Minnie     1746 0.0179
## 6  1880 F     Margaret   1578 0.0162
```

```
## # A tibble: 0 x 4
## # ... with 4 variables: name <chr>, sex <chr>, year <dbl>, nn <int>
```

```
## # A tibble: 6 x 11
##     lat  long  year month surftemp  temp pressure ozone cloudlow cloudmid
##   <dbl> <dbl> <int> <int>    <dbl> <dbl>    <dbl> <dbl>    <dbl>    <dbl>
## 1  36.2 -114.  1995     1     273.  272.      835   304      7.5     34.5
## 2  33.7 -114.  1995     1     280.  282.      940   304     11.5     32.5
## 3  31.2 -114.  1995     1     285.  285.      960   298     16.5     26
## 4  28.7 -114.  1995     1     289.  291.      990   276     20.5     14.5
## 5  26.2 -114.  1995     1     292.  293.     1000   274     26       10.5
## 6  23.7 -114.  1995     1     294.  294.     1000   264     30        9.5
## # ... with 1 more variable: cloudhigh <dbl>
```

```
## # A tibble: 0 x 5
## # ... with 5 variables: lat <dbl>, long <dbl>, year <int>, month <int>, n <int>
```

- **Exercise2:** (1/2 pt) What is the relationship between the Batting, Master, and Salaries tables in the Lahman package? What are the keys for each dataset and how do they relate to each other?

Mutating joins

A join connects each row in x to 0, 1, or more rows in y

Inner join - connects pairs of observations when their keys are equal. Unmatched rows not included in the result

```
x <- tribble(
  ~key, ~val_x,
```

```
  #-----/------
      1,  "x1",
      2,  "x2",
      3,  "x3"
  )
y <- tribble(
  ~key, ~val_y,
  #-----/------
      1,  "y1",
      2,  "y2",
      4,  "y3"
  )
x %>%
  inner_join(y, by = "key")
```

```
## # A tibble: 2 x 3
##     key val_x val_y
## * <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
```

```
x %>%
  inner_join(y)
```

```
## # A tibble: 2 x 3
##     key val_x val_y
## * <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
```

Outer joins

Keep observations that appear in at least one of the tables

- `left_join(x, y)` keeps all observations in x
- `right_join(x, y)` keeps all observations in y
- `full_join(x, y)` keeps all observations

```
x %>%
  left_join(y)
```

```
## # A tibble: 3 x 3
##     key val_x val_y
## * <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
## 3     3 x3    <NA>
```

```
x %>%
  right_join(y)
```

```
## # A tibble: 3 x 3
##     key val_x val_y
## * <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
## 3     4 <NA>  y3
```

```
x %>%
  full_join(y)
```

```
## # A tibble: 4 x 3
##     key val_x val_y
## * <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
## 3     3 x3    <NA>
## 4     4 <NA>  y3
```

duplicate keys

If one table has duplicate keys, adds in additional info from the other table. If both tables have duplicate keys, will get every possible combination of the entries.

```
x <- tribble(
  ~key, ~val_x,
  #-----/------
      1,  "x1",
      2,  "x2",
      2,  "x3",
      3,  "x4"
  )
y <- tribble(
  ~key, ~val_y,
  #-----/------
      1,  "y1",
      2,  "y2"
  )
x %>% left_join(y, by="key")
```

```
## # A tibble: 4 x 3
##     key val_x val_y
## * <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
## 3     2 x3    y2
## 4     3 x4    <NA>
```

```
y <- tribble(
  ~key, ~val_y,
  #-----/------
      1,  "y1",
      2,  "y2",
      2,  "y3",
      3,  "y4"
  )
x %>% left_join(y)
```

```
## # A tibble: 6 x 3
##     key val_x val_y
## * <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
```

```
## 3     2 x2    y3
## 4     2 x3    y2
## 5     2 x3    y3
## 6     3 x4    y4
```

Specifying keys for the joins

- natural join is the default and uses all variables with the same name in both tables
- by = "variable name" joins by specified variables only. other variables with same name are included with a suffix
- by = c("vara" = "varb") to set two variable names equal

Exercises

- **Exercise 3:** (2 points) Use an appropriate join to add a column containing the airline name to the flights dataset. Be sure to put the carrier code and name in the first two columns of the result so we can see them. Save the result as flights2.
- **Exercise 4:** (2 points) Use an appropriate join to add the airport name to the flights2 dataset you got in exercise 3. The codes and names of the airports are in the airports dataset of the nycflights13 package. Put the carrier and carrier name first followed by the destination and destination name, then everything else.

```
## # A tibble: 336,776 x 27
##    carrier carrier_name dest  dest_name  year month   day dep_time
##    <chr>   <chr>        <chr> <chr>     <int> <int> <int>    <int>
##  1 UA      United Air ~ IAH   George B~  2013     1     1      517
##  2 UA      United Air ~ IAH   George B~  2013     1     1      533
##  3 AA      American Ai~ MIA   Miami In~  2013     1     1      542
##  4 B6      JetBlue Air~ BQN   <NA>       2013     1     1      544
##  5 DL      Delta Air L~ ATL   Hartsfie~  2013     1     1      554
##  6 UA      United Air ~ ORD   Chicago ~  2013     1     1      554
##  7 B6      JetBlue Air~ FLL   Fort Lau~  2013     1     1      555
##  8 EV      ExpressJet ~ IAD   Washingt~  2013     1     1      557
##  9 B6      JetBlue Air~ MCO   Orlando ~  2013     1     1      557
## 10 AA      American Ai~ ORD   Chicago ~  2013     1     1      558
## # ... with 336,766 more rows, and 19 more variables: sched_dep_time <int>,
## #   dep_delay <dbl>, arr_time <int>, sched_arr_time <int>, arr_delay <dbl>,
## #   flight <int>, tailnum <chr>, origin <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>, lat <dbl>, lon <dbl>,
## #   alt <dbl>, tz <dbl>, dst <chr>, tzone <chr>
```

Filtering joins

filter observations from one tibble based on whether or not they match an observation in another tibble

`semi_join(x,y)`

- keeps all observations in x that have a match in y
- useful for matching filtered summary back to original rows

```
top_dest <- flights %>%
  count(dest, sort=TRUE) %>%
```

```
  head(10)
top_dest
```

```
## # A tibble: 10 x 2
##    dest      n
##    <chr> <int>
##  1 ORD   17283
##  2 ATL   17215
##  3 LAX   16174
##  4 BOS   15508
##  5 MCO   14082
##  6 CLT   14064
##  7 SFO   13331
##  8 FLL   12055
##  9 MIA   11728
## 10 DCA    9705
```

```
# keep only flights from the top destinations
flights %>%
  semi_join(top_dest)
```

```
## # A tibble: 141,145 x 19
##     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      542            540         2      923            850
##  2  2013     1     1      554            600        -6      812            837
##  3  2013     1     1      554            558        -4      740            728
##  4  2013     1     1      555            600        -5      913            854
##  5  2013     1     1      557            600        -3      838            846
##  6  2013     1     1      558            600        -2      753            745
##  7  2013     1     1      558            600        -2      924            917
##  8  2013     1     1      558            600        -2      923            937
##  9  2013     1     1      559            559         0      702            706
## 10  2013     1     1      600            600         0      851            858
## # ... with 141,135 more rows, and 11 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>
```

anti_join(x,y)

- drops all observations in x that have a match in y
- useful for diagnosing join mismatches

```
fl <- flights %>%
  anti_join(planes, by="tailnum") %>%
  count(tailnum, sort=TRUE)
```

Interesting. Look at the ones with tailnum = NA.

```
flights %>% filter(is.na(tailnum))
```

```
## # A tibble: 2,512 x 19
##     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     2       NA           1545        NA       NA           1910
##  2  2013     1     2       NA           1601        NA       NA           1735
```

```
##  3  2013     1     3       NA            857        NA       NA           1209
##  4  2013     1     3       NA            645        NA       NA            952
##  5  2013     1     4       NA            845        NA       NA           1015
##  6  2013     1     4       NA           1830        NA       NA           2044
##  7  2013     1     5       NA            840        NA       NA           1001
##  8  2013     1     7       NA            820        NA       NA            958
##  9  2013     1     8       NA           1645        NA       NA           1838
## 10  2013     1     9       NA            755        NA       NA           1012
## # ... with 2,502 more rows, and 11 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>
```

```
# Let's see if these are all due to cancelled flights
flights %>% filter(is.na(tailnum)) %>% filter(!is.na(dep_time))
```

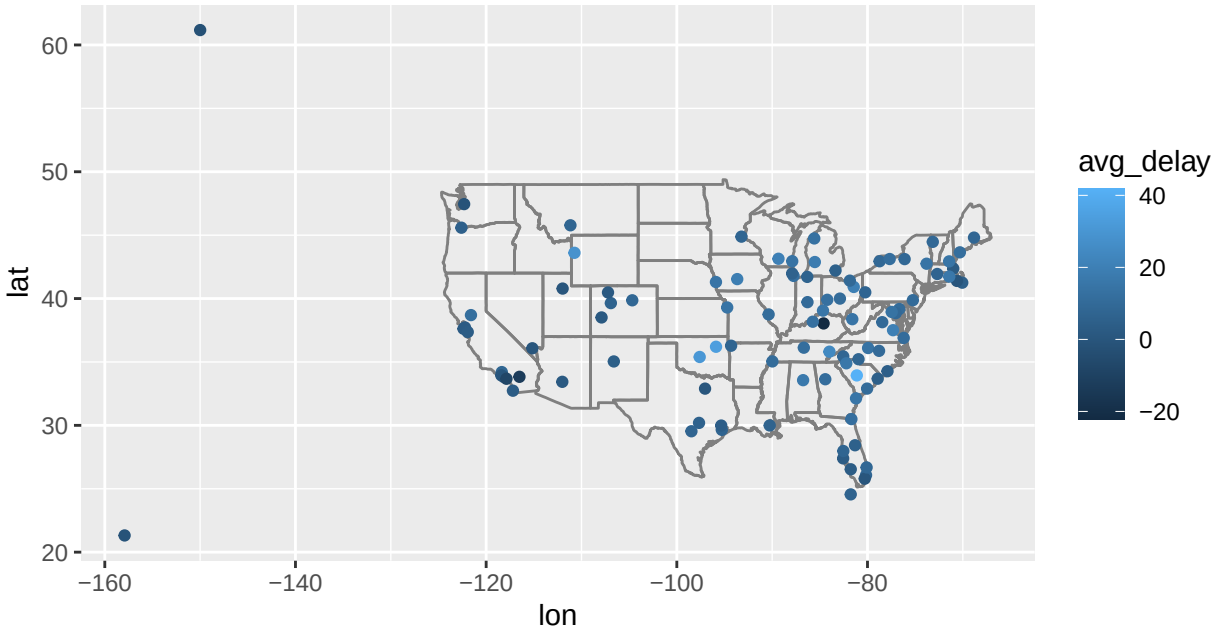
```
## # A tibble: 0 x 19
## # ... with 19 variables: year <int>, month <int>, day <int>, dep_time <int>,
## #   sched_dep_time <int>, dep_delay <dbl>, arr_time <int>,
## #   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Here is a way to see all the flight destinations from NYC on a map.

```
airports %>%
  semi_join(flights, c("faa" = "dest")) %>%
  ggplot(aes(lon, lat)) +
    borders("state") +
    geom_point() +
    coord_quickmap()
```

- **Exercise 5:** (2 points) Compute the average delay by destination, then join on the airports data frame so you can show the spatial distribution of delays.
- Use the size or colour of the points to display the average delay for each airport.
- Add the location of the origin and destination (i.e. the lat and lon) to flights.
- Compute the average delay by destination.

```
## Warning: Removed 4 rows containing missing values (geom_point).
```



Set operations

- intersect(x,y)
- union(x,y)
- setdiff(x,y) - those in x but not in y

```
x <- tribble(
  ~var1, ~var2,
    "a", 1,
    "b", 2,
    "c", 3
)
y <- tribble(
  ~var1, ~var2,
     "a", 1,
     "c", 3,
     "d", 4
)
intersect(x,y)
```

```
## # A tibble: 2 x 2
##   var1   var2
##   <chr> <dbl>
## 1 a         1
```

```
## 2 c         3
```

```
union(x,y)
```

```
## # A tibble: 4 x 2
##   var1   var2
##   <chr> <dbl>
## 1 a         1
## 2 b         2
## 3 c         3
## 4 d         4
```

```
setdiff(x,y)
```

```
## # A tibble: 1 x 2
##   var1   var2
##   <chr> <dbl>
## 1 b         2
```

```
setdiff(y,x)
```

```
## # A tibble: 1 x 2
##   var1   var2
##   <chr> <dbl>
## 1 d         4
```

Exercise 6: (2 points) Use a set operation function to find which airport codes from flights are not in the airports dataset.