Lecture 16: Relational Data

STAT 450, Fall 2021

Reading: Sections 13.1 - 13.4 from *R for Data Science*

In this lecture we discuss how to combine multiple tables of data together.

Multiple tables of data that are related are called **relational data**. The variable(s) used to connect two tables is called a **key**. A key should uniquely identify the observations (rows) in one of the tables that are being joined.

Example: `nycflights13`

```
library(tidyverse)
library(nycflights13)
```

Recall the `flights` data that contains information on all flights departing from NYC in 2013.

```
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>
```

`airlines` is a related data set that gives the abbreviation and full name for each carrier.

```
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.
```

The `flights` data frame has many columns, so let's select a few to make it easier to see.

```
flights2 <- flights %>%
  select(year:day, hour, origin, dest, tailnum, carrier)
flights2
```

```
## # A tibble: 336,776 x 8
##     year month   day  hour origin dest  tailnum carrier
##    <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>
##  1  2013     1     1     5 EWR    IAH   N14228  UA
##  2  2013     1     1     5 LGA    IAH   N24211  UA
##  3  2013     1     1     5 JFK    MIA   N619AA  AA
##  4  2013     1     1     5 JFK    BQN   N804JB  B6
##  5  2013     1     1     6 LGA    ATL   N668DN  DL
##  6  2013     1     1     5 EWR    ORD   N39463  UA
##  7  2013     1     1     6 EWR    FLL   N516JB  B6
##  8  2013     1     1     6 LGA    IAD   N829AS  EV
##  9  2013     1     1     6 JFK    MCO   N593JB  B6
## 10  2013     1     1     6 LGA    ORD   N3ALAA  AA
## # ... with 336,766 more rows
```

Suppose we want to add the names of the different carriers to `flights2`. To do this we would need to combine the `airlines` and `flights2` data frames. The variable `carrier` is the key that relates the two tables. We can use the function `inner_join()` to combine the tables:

```
flights2 %>% inner_join(airlines, by = "carrier")
```

```
## # A tibble: 336,776 x 9
##     year month   day  hour origin dest  tailnum carrier name
##    <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>   <chr>
##  1  2013     1     1     5 EWR    IAH   N14228  UA      United Air Lines Inc.
##  2  2013     1     1     5 LGA    IAH   N24211  UA      United Air Lines Inc.
##  3  2013     1     1     5 JFK    MIA   N619AA  AA      American Airlines Inc.
##  4  2013     1     1     5 JFK    BQN   N804JB  B6      JetBlue Airways
##  5  2013     1     1     6 LGA    ATL   N668DN  DL      Delta Air Lines Inc.
##  6  2013     1     1     5 EWR    ORD   N39463  UA      United Air Lines Inc.
##  7  2013     1     1     6 EWR    FLL   N516JB  B6      JetBlue Airways
##  8  2013     1     1     6 LGA    IAD   N829AS  EV      ExpressJet Airlines Inc.
##  9  2013     1     1     6 JFK    MCO   N593JB  B6      JetBlue Airways
## 10  2013     1     1     6 LGA    ORD   N3ALAA  AA      American Airlines Inc.
## # ... with 336,766 more rows
```

Note that the following command gives the same result:

```
inner_join(flights2, airlines, by = "carrier")
```

We see that the combined data set has 336,776 rows, which is the same as the original data set `flights2`. This is because each row of `flights2` matched with exactly one corresponding entry of `airlines`. It's good practice to always check the dimensions of your data after doing a join operation.

Inner join

An inner join matches pairs of observations whenever their keys are equal. Consider the following simple example:

```
x <- tibble(
  key = c(1, 2, 3),
  val_x = c("x1", "x2", "x3")
)
y <- tibble(
  key = c(1, 2, 4),
  val_y = c("y1", "y2", "y3")
)
```

```
x
```

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

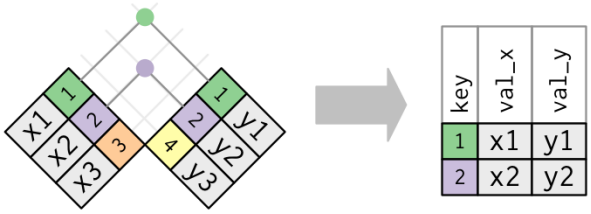
```
y
```

```
## # A tibble: 3 x 2
##     key val_y
##   <dbl> <chr>
## 1     1 y1
## 2     2 y2
## 3     4 y3
```

```
inner_join(x, y, by = "key")
```

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

The result of an inner join only contains observations (rows) that match according to the key. In the above example only the rows with key values 1 and 2 matched in both tables. Here's an illustration:



Exercise: Consider the following data frames:

```
band_members
```

```
## # A tibble: 3 x 2
##   name  band
##   <chr> <chr>
## 1 Mick  Stones
## 2 John  Beatles
## 3 Paul  Beatles
```

```
band_instruments
```

```
## # A tibble: 3 x 2
##   name  plays
##   <chr> <chr>
## 1 John  guitar
## 2 Paul  bass
## 3 Keith guitar
```

a) What is the *key* that relates the two data frames?

b) Try to predict the output of the following code. Then run the code in R to check if your prediction was correct.

```
inner_join(band_members, band_instruments, by = "name")
```

Exercise: Use `group_by()` and `summarise()` to compute the average arrival delay for each carrier. Then use `inner_join()` to include a column with the full carrier name. This is what the resulting data frame should look like:

```
## # A tibble: 16 x 4
##    carrier count arr_delay_mean name
##    <chr>   <int>          <dbl> <chr>
##  1 9E      18460          7.38  Endeavor Air Inc.
##  2 AA      32729          0.364 American Airlines Inc.
##  3 AS        714         -9.93  Alaska Airlines Inc.
##  4 B6      54635          9.46  JetBlue Airways
##  5 DL      48110          1.64  Delta Air Lines Inc.
##  6 EV      54173         15.8   ExpressJet Airlines Inc.
##  7 F9        685         21.9   Frontier Airlines Inc.
##  8 FL       3260         20.1   AirTran Airways Corporation
##  9 HA        342         -6.92  Hawaiian Airlines Inc.
## 10 MQ      26397         10.8   Envoy Air
## 11 OO         32         11.9   SkyWest Airlines Inc.
## 12 UA      58665          3.56  United Air Lines Inc.
## 13 US      20536          2.13  US Airways Inc.
## 14 VX       5162          1.76  Virgin America
## 15 WN      12275          9.65  Southwest Airlines Co.
## 16 YV        601         15.6   Mesa Airlines Inc.
```

Outer joins

An inner join only keeps the rows that match in both tables. An outer join, on the other hand, keeps all rows in at least one of the tables, regardless of whether they match. There are three types of outer joins:

- A **left join** keeps all rows in the first table `x`
- A **right join** keeps all rows in the second table `y`
- A **full join** keeps all rows in both tables `x` and `y`

```
x
```

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

```
y
```

```
## # A tibble: 3 x 2
##     key val_y
##   <dbl> <chr>
## 1     1 y1
## 2     2 y2
## 3     4 y3
```

```
left_join(x, y, by = "key")
```

```
## # 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>
```

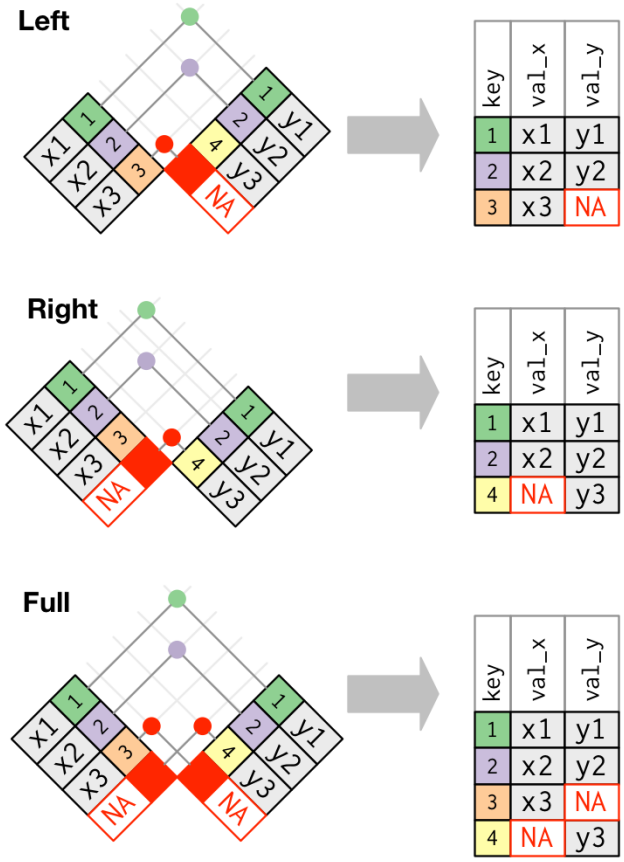
```
right_join(x, y, by = "key")
```

```
## # 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
```

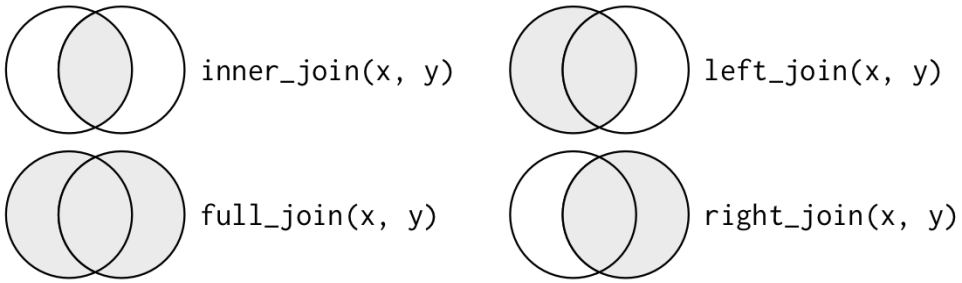
```
full_join(x, y, by = "key")
```

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

An illustration:



We can also use Venn diagrams to depict the different types of joins:



Inner and left joins are the most common types of joins. Right and full joins are used less often in practice.

Exercise: Again, consider the following data frames:

```
band_members
```

```
## # A tibble: 3 x 2
##   name  band
##   <chr> <chr>
## 1 Mick  Stones
## 2 John  Beatles
## 3 Paul  Beatles
```

```
band_instruments
```

```
## # A tibble: 3 x 2
##   name  plays
##   <chr> <chr>
## 1 John  guitar
## 2 Paul  bass
## 3 Keith guitar
```

Try to predict the output of the following code. Then run the code in R to check if your prediction was correct.

```
left_join(band_members, band_instruments, by = "name")
full_join(band_members, band_instruments, by = "name")
```

Defining the key columns

So far, the pairs of tables have always been joined by a single variable, and that variable has the same name in both tables. The syntax `by = "key"` was used to perform the join.

Example: joining two tables using several variables as the key

By default, all variables that are common to both tables are used to perform the join.

For example, consider the data frame `weather` which gives the weather at each NYC airport on each hour of each day of the year.

```
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>
```

The flights and weather tables match on their common variables: `year`, `month`, `day`, `hour`, and `origin`

```
flights2 %>% left_join(weather)
```

```
## Joining, by = c("year", "month", "day", "hour", "origin")
```

```
## # A tibble: 336,776 x 18
##     year month   day  hour origin dest  tailnum carrier  temp  dewp humid
##    <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>   <dbl> <dbl> <dbl>
##  1  2013     1     1     5 EWR    IAH   N14228  UA       39.0  28.0  64.4
##  2  2013     1     1     5 LGA    IAH   N24211  UA       39.9  25.0  54.8
##  3  2013     1     1     5 JFK    MIA   N619AA  AA       39.0  27.0  61.6
##  4  2013     1     1     5 JFK    BQN   N804JB  B6       39.0  27.0  61.6
##  5  2013     1     1     6 LGA    ATL   N668DN  DL       39.9  25.0  54.8
##  6  2013     1     1     5 EWR    ORD   N39463  UA       39.0  28.0  64.4
##  7  2013     1     1     6 EWR    FLL   N516JB  B6       37.9  28.0  67.2
##  8  2013     1     1     6 LGA    IAD   N829AS  EV       39.9  25.0  54.8
##  9  2013     1     1     6 JFK    MCO   N593JB  B6       37.9  27.0  64.3
## 10  2013     1     1     6 LGA    ORD   N3ALAA  AA       39.9  25.0  54.8
## # ... with 336,766 more rows, and 7 more variables: wind_dir <dbl>,
## #   wind_speed <dbl>, wind_gust <dbl>, precip <dbl>, pressure <dbl>,
## #   visib <dbl>, time_hour <dttm>
```

Example: the key has different names

If the key variable has different names in the two tables use `by = c("a" = "b")`

This will match the variable named `a` in table `x` with the variable named `b` in table `y`

For example, the data frame `airports` gives information about each airport, identified by the `faa` airport code.

```
airports
```

```
## # A tibble: 1,458 x 8
##    faa   name                            lat    lon   alt    tz dst   tzone
##    <chr> <chr>                         <dbl>  <dbl> <dbl> <dbl> <chr> <chr>
##  1 04G   Lansdowne Airport              41.1  -80.6  1044    -5 A     America/~
##  2 06A   Moton Field Municipal Airport  32.5  -85.7   264    -6 A     America/~
##  3 06C   Schaumburg Regional            42.0  -88.1   801    -6 A     America/~
##  4 06N   Randall Airport                41.4  -74.4   523    -5 A     America/~
##  5 09J   Jekyll Island Airport          31.1  -81.4    11    -5 A     America/~
##  6 0A9   Elizabethton Municipal Airport 36.4  -82.2  1593    -5 A     America/~
##  7 0G6   Williams County Airport        41.5  -84.5   730    -5 A     America/~
##  8 0G7   Finger Lakes Regional Airport  42.9  -76.8   492    -5 A     America/~
##  9 0P2   Shoestring Aviation Airfield   39.8  -76.6  1000    -5 U     America/~
## 10 0S9   Jefferson County Intl          48.1 -123.    108    -8 A     America/~
## # ... with 1,448 more rows
```

Suppose we want to join `flights2` and `airports` so that each row of `flights2` contains additional information about the destination airport. Then we would want to match the `dest` column of `flights2` with the `faa` column of `airports`:

```
flights2 %>%
  left_join(airports, by = c("dest" = "faa")) %>%
  glimpse()
```

```
## Rows: 336,776
## Columns: 15
## $ year    <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 20~
## $ month   <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ day     <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ hour    <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6, 6,~
## $ origin  <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LGA", "JFK",~
## $ dest    <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IAD", "MCO",~
## $ tailnum <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N39463", "N~
## $ carrier <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6", "AA", "B~
## $ name    <chr> "George Bush Intercontinental", "George Bush Intercontinental"~
## $ lat     <dbl> 29.98443, 29.98443, 25.79325, NA, 33.63672, 41.97860, 26.07258~
## $ lon     <dbl> -95.34144, -95.34144, -80.29056, NA, -84.42807, -87.90484, -80~
## $ alt     <dbl> 97, 97, 8, NA, 1026, 668, 9, 313, 96, 668, 19, 26, 126, 13, 60~
## $ tz      <dbl> -6, -6, -5, NA, -5, -6, -5, -5, -5, -6, -5, -5, -8, -8, -6, -5~
## $ dst     <chr> "A", "A", "A", NA, "A", "A", "A", "A", "A", "A", "A", "A", "A"~
## $ tzone   <chr> "America/Chicago", "America/Chicago", "America/New_York", NA, ~
```

Last, it's good practice to check that the key is a unique identifier for the rows in one of the two tables that you are trying to join. One way to do this is with `count()`:

```
airports %>% count(faa)
```

```
## # A tibble: 1,458 x 2
##    faa       n
##    <chr> <int>
##  1 04G       1
##  2 06A       1
##  3 06C       1
##  4 06N       1
##  5 09J       1
##  6 0A9       1
##  7 0G6       1
##  8 0G7       1
##  9 0P2       1
## 10 0S9       1
## # ... with 1,448 more rows
```

```
airports %>%
  count(faa) %>%
  filter(n > 1)
```

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

```
weather %>%
  count(year, month, day, hour, origin) %>%
  filter(n > 1)
```

```
## # A tibble: 3 x 6
##    year month   day  hour origin     n
##   <int> <int> <int> <int> <chr>  <int>
## 1  2013    11     3     1 EWR        2
## 2  2013    11     3     1 JFK        2
## 3  2013    11     3     1 LGA        2
```

So there are multiple entries in `weather` on 2013-11-03 at 1am. Let's check the entries for that day:

```
weather %>%
  filter(year == 2013, month == 11, day == 3, hour == 1)
```

```
## # A tibble: 6 x 15
##   origin  year month   day  hour  temp  dewp humid wind_dir wind_speed wind_gust
##   <chr>  <int> <int> <int> <int> <dbl> <dbl> <dbl>    <dbl>      <dbl>     <dbl>
## 1 EWR     2013    11     3     1  52.0  39.0  61.2      310       6.90        NA
## 2 EWR     2013    11     3     1  50    39.0  65.8      290       5.75        NA
## 3 JFK     2013    11     3     1  54.0  37.9  54.5      320       9.21        NA
## 4 JFK     2013    11     3     1  52.0  37.9  58.6      310       6.90        NA
## 5 LGA     2013    11     3     1  55.0  39.0  54.7      330       9.21        NA
## 6 LGA     2013    11     3     1  54.0  39.9  58.9      310       8.06        NA
## # ... with 4 more variables: precip <dbl>, pressure <dbl>, visib <dbl>,
## #   time_hour <dttm>
```

Daylight savings ended on 2013-11-03, this is probably why there are duplicate entries:

https://www.timeanddate.com/time/change/usa?year=2013