Stat 260, Lecture 7, Relational Data

Brad McNeney

Load packages and datasets

```
library(tidyverse)
library(nycflights13)
# Example from http://www.itl.nist.gov/div897/ctg/dm/sql_examples.htm
station <- read_csv("station.csv",col_types=cols(ID=col_integer()))
stats <- read_csv("stats.csv",col_types=cols(ID=col_integer()))
```

Reading

- Relational Data: Chapter 10 of printed text, Chapter 13 of online text.
  - In the sections on "Joins", we will focus on the left-join (in the **Mutating Joins** section) and the semi-join (in the **Filtering Joins** section).
- Data transformation (dplyr) cheatsheet at [https://github.com/rstudio/cheatsheets/raw/master/data-transformation.pdf]

Multiple Tables

- Modern data comes in multiple tables, called relational data.
- Such structure is motivated by relational database management systems (RDBMS) that revolutionized database management.
- **Example** `station` and `stats` tables for weather data:

```
station
```

```
## # A tibble: 3 x 5
##      ID City    State Lat_N Long_W
##   <int> <chr>   <chr> <dbl>  <dbl>
## 1    13 Phoenix AZ       33    112
## 2    44 Denver  CO       40    105
## 3    66 Caribou ME       47     68
```

```
stats
```

```
## # A tibble: 6 x 4
##      ID Month Temp_F Rain_I
##   <int> <dbl>  <dbl>  <dbl>
## 1    13     1   57.4   0.31
## 2    13     7   91.7   5.15
## 3    44     1   27.3   0.18
## 4    44     7   74.8   2.11
## 5    66     1    6.7   2.1
## 6    66     7   65.8   4.52
```

Relations in the Weather Data

- The relation, or connection between `station` and `stats` is the `ID` variable present in both.
- Think of the `ID`s as short-hand for the info in the `station` table.
  - For example, `ID` 13 from `station` is short-hand for: Phoenix AZ at latitude 33N and longitude 112W.
- The `stats` table is much more concise for not repeating the info on each station.

Joining tables

- However, in some cases we may wish to include information, such as station name, in the `stats` table.
  - The text calls this a "mutating join".
- Or, we may wish to filter weather measurements in `stats` based on the characteristics of the stations, such as latititude `Lat_N >= 40`.
  - The text calls this a "filtering join".

The nycflights13 Relational Data

- We used the `flights` data from `nycflights13`. There are several other tables in this package:

```
print(airlines,n=3)
```

```
## # A tibble: 16 x 2
##   carrier name
##   <chr>   <chr>
## 1 9E      Endeavor Air Inc.
## 2 AA      American Airlines Inc.
## 3 AS      Alaska Airlines Inc.
## # ... with 13 more rows
```

```
print(airports,n=3)
```

```
## # A tibble: 1,458 x 8
##   faa   name                      lat   lon   alt    tz dst   tzone
##   <chr> <chr>                   <dbl> <dbl> <int> <dbl> <chr> <chr>
## 1 04G   Lansdowne Airport        41.1 -80.6  1044    -5 A     America/New_~
## 2 06A   Moton Field Municipal ~  32.5 -85.7   264    -6 A     America/Chic~
## 3 06C   Schaumburg Regional      42.0 -88.1   801    -6 A     America/Chic~
## # ... with 1,455 more rows
```

```r
print(planes,n=3)
```

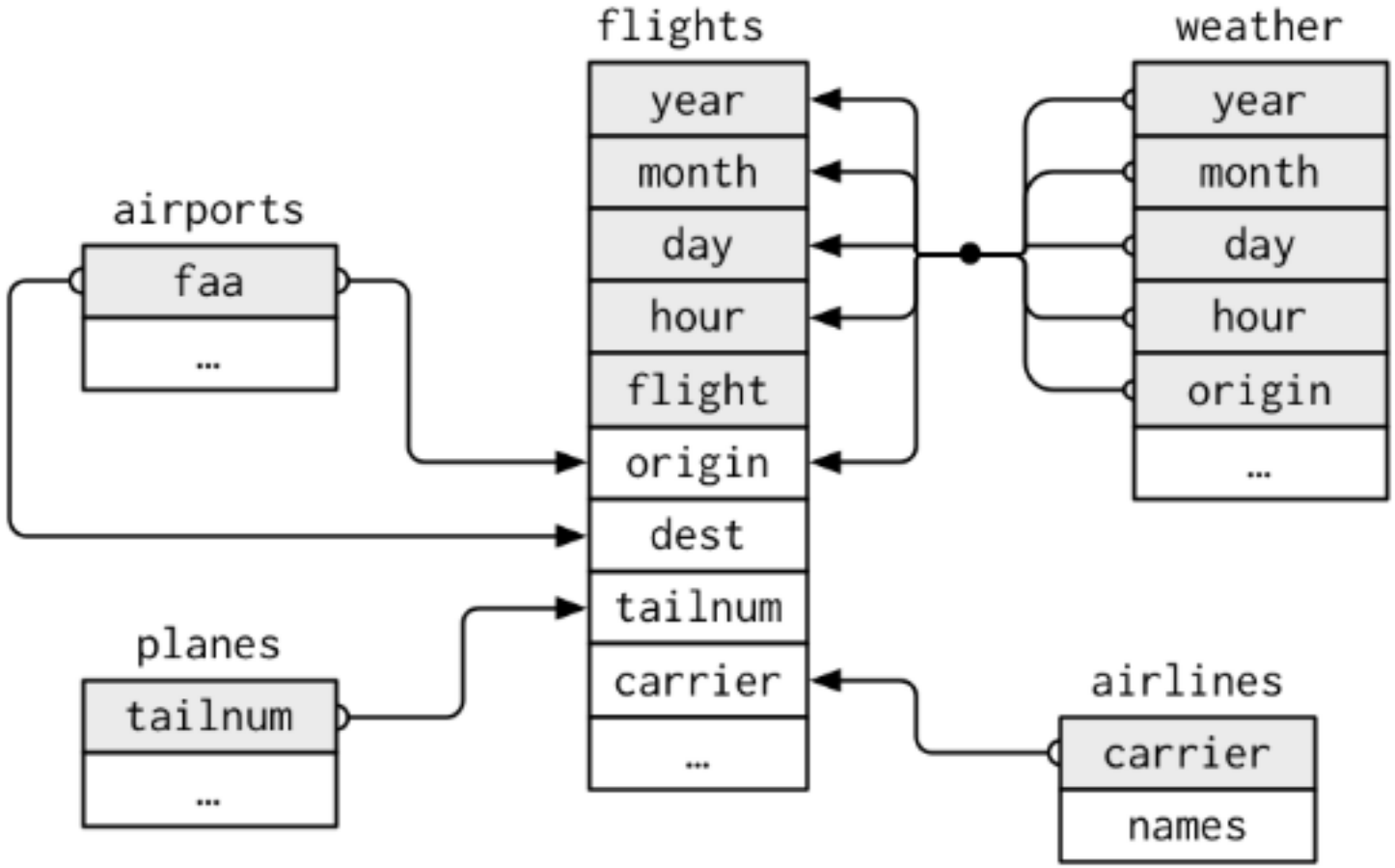
```
## # A tibble: 3,322 x 9
##   tailnum  year type       manufacturer   model  engines seats speed engine
##   <chr>   <int> <chr>      <chr>          <chr>    <int> <int> <int> <chr>
## 1 N10156   2004 Fixed win~ EMBRAER        EMB-1~       2    55    NA Turbo~
## 2 N102UW   1998 Fixed win~ AIRBUS INDUST~ A320-~       2   182    NA Turbo~
## 3 N103US   1999 Fixed win~ AIRBUS INDUST~ A320-~       2   182    NA Turbo~
## # ... with 3,319 more rows
```

```r
print(weather,n=3)
```

```
## # A tibble: 26,115 x 15
##   origin  year month   day  hour  temp  dewp humid wind_dir wind_speed
##   <chr>  <dbl> <dbl> <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
## # ... with 2.611e+04 more rows, and 5 more variables: wind_gust <dbl>,
## #   precip <dbl>, pressure <dbl>, visib <dbl>, time_hour <dttm>
```

Relations in the `nycflights` Tables

- Figure from Chapter 10 of R for Data Science:



- **Exercise:** The relationship between the `weather` and `airports` tables is not shown on the diagram. What is it?

Keys

- The `ID` variable in the `station` table is its "primary key".
- It uniquely identifies observations (rows) in the table.
- The `ID` column in the `stats` table is a "foreign key" that links to the `station` table.
- Primary/secondary keys are the "relations" in relational data.

```
print(station,n=3)
```

```
## # A tibble: 3 x 5
##      ID City    State Lat_N Long_W
##   <int> <chr>   <chr> <dbl>  <dbl>
## 1    13 Phoenix AZ       33    112
## 2    44 Denver  CO       40    105
## 3    66 Caribou ME       47     68
```

```
print(stats,n=3)
```

```
## # A tibble: 6 x 4
##      ID Month Temp_F Rain_I
##   <int> <dbl>  <dbl>  <dbl>
## 1    13     1   57.4   0.31
## 2    13     7   91.7   5.15
## 3    44     1   27.3   0.18
## # ... with 3 more rows
```

Multiple Keys

- It may take multiple variables to uniquely identify an observation in a table.
- For example, in the `weather` table from `nycflights13`, we need `year-month-day-hour` plus `origin`.

```
print(weather,n=3)
```

```
## # A tibble: 26,115 x 15
##   origin  year month   day  hour  temp  dewp humid wind_dir wind_speed
##   <chr>  <dbl> <dbl> <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
## # ... with 2.611e+04 more rows, and 5 more variables: wind_gust <dbl>,
## #   precip <dbl>, pressure <dbl>, visib <dbl>, time_hour <dttm>
```

Tables with no Primary Key

- Some tables lack the variables needed to uniquely identify its observations.
- For example, in the flights table `year-month-day`, `flight` and `tailnum` do **not** identify the flight.

```
flights %>%
  count(year,month,day,flight,tailnum) %>%
  filter(n > 1) %>% print(n=3)
```

```
## # A tibble: 11 x 6
##    year month   day flight tailnum     n
##   <int> <int> <int>  <int> <chr>   <int>
## 1  2013     2     9    303 <NA>        2
## 2  2013     2     9    655 <NA>        2
## 3  2013     2     9   1623 <NA>        2
## # ... with 8 more rows
```

Surrogate Key

- You can add a "surrogate key" to a table with no primary key using `mutate()` and `row_number()`.
- **Exercise** Select year, month, day, flight and tailnum from `flights` and add a surrogate key to this 5-column table.

Joining Tables

- To start, add the data in our `stations` table to the weather statistics in the `stats` table.
  - This is what the text calls a mutating joing, because it adds columns to `stats` like a call to `mutate()` would.

```
stats %>% left_join(station)
```

```
## # A tibble: 6 x 8
##      ID Month Temp_F Rain_I City    State Lat_N Long_W
##   <int> <dbl>  <dbl>  <dbl> <chr>   <chr> <dbl>  <dbl>
## 1    13     1   57.4   0.31 Phoenix AZ       33    112
## 2    13     7   91.7   5.15 Phoenix AZ       33    112
## 3    44     1   27.3   0.18 Denver  CO       40    105
## 4    44     7   74.8   2.11 Denver  CO       40    105
## 5    66     1    6.7   2.1  Caribou ME       47     68
## 6    66     7   65.8   4.52 Caribou ME       47     68
```

A Note on Joins

- The term "join" is from SQL (Structured Query Language), which is the standard language used to construct RDBMS queries.
- There are **many** types of joins. In this class we will focus on the two that I think are most useful in data analysis, the left-join (a mutating join) and the semi-join (a filtering join).
- However, for a bit of context, we will discuss inner- and outer-joins.

Inner Joins

- To illustrate inner- and outer-joins we remove one of the rows of `station`.

```
station <- station[-3,]
```

- Now repeat the `left_join()`:

```
stats %>% left_join(station)
```

```
## # A tibble: 6 x 8
##      ID Month Temp_F Rain_I City    State Lat_N Long_W
##   <int> <dbl>  <dbl>  <dbl> <chr>   <chr> <dbl>  <dbl>
## 1    13     1   57.4   0.31 Phoenix AZ       33    112
## 2    13     7   91.7   5.15 Phoenix AZ       33    112
## 3    44     1   27.3   0.18 Denver  CO       40    105
## 4    44     7   74.8   2.11 Denver  CO       40    105
## 5    66     1    6.7   2.1  <NA>    <NA>     NA     NA
## 6    66     7   65.8   4.52 <NA>    <NA>     NA     NA
```

The Left-Join as an Outer Join

- In the output of `left_join()`, the data from `stats` is on the left.
- The `left_join()` is an "outer join", because it keeps observations in one or more of the tables, in this case the left table.
- Similarly, a right-join keeps all observations in the right table, and a full-join keeps all observations in both tables.

The Inner-Join

- The inner-join keeps observations that appear in **both** tables.

```
stats %>% inner_join(station)
```

```
## # A tibble: 4 x 8
##      ID Month Temp_F Rain_I City    State Lat_N Long_W
##   <int> <dbl>  <dbl>  <dbl> <chr>   <chr> <dbl>  <dbl>
## 1    13     1   57.4   0.31 Phoenix AZ       33    112
## 2    13     7   91.7   5.15 Phoenix AZ       33    112
## 3    44     1   27.3   0.18 Denver  CO       40    105
## 4    44     7   74.8   2.11 Denver  CO       40    105
```

- This might be good if we only wanted data from stations in `station`, but there is a tendency to accidentally loose data.
  - Better to use a filtering join (next topic).

Defining the Key Columns

- The `station` and `stats` tables are very easy to join because the primary key in `station` has the same name as the primary key in `stats`.
- The `by` argument to `left_join()` lets you specify the keys to match on.
- The default is `by = NULL`.
  - Uses all variables that appear in both tables; `ID` for `station` and `stats`.
  - This is called a "natural" join.

Left-Joins with nycflights13

- To illustrate left-joins with the nycflights13 data we use the reduced flights table defined in the text.

```
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
```

Natural Join of weather

- flights2 and weather share the variables year, month, day, hour and origin.

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

```
## # A tibble: 336,776 x 18
##     year month   day  hour origin dest  tailnum carrier  temp  dewp humid
##    <dbl> <dbl> <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>
```

by = x

- Use by=x to join on a specific column.
- For example, year means something different in flights2 and planes. Use only tailnum.

```
flights2 %>% left_join(planes, by="tailnum")
```

```
## # A tibble: 336,776 x 16
##    year.x month   day  hour origin dest  tailnum carrier year.y type
##     <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>    <int> <chr>
##  1   2013     1     1     5 EWR    IAH   N14228  UA        1999 Fixe~
##  2   2013     1     1     5 LGA    IAH   N24211  UA        1998 Fixe~
##  3   2013     1     1     5 JFK    MIA   N619AA  AA        1990 Fixe~
##  4   2013     1     1     5 JFK    BQN   N804JB  B6        2012 Fixe~
##  5   2013     1     1     6 LGA    ATL   N668DN  DL        1991 Fixe~
##  6   2013     1     1     5 EWR    ORD   N39463  UA        2012 Fixe~
##  7   2013     1     1     6 EWR    FLL   N516JB  B6        2000 Fixe~
##  8   2013     1     1     6 LGA    IAD   N829AS  EV        1998 Fixe~
##  9   2013     1     1     6 JFK    MCO   N593JB  B6        2004 Fixe~
## 10   2013     1     1     6 LGA    ORD   N3ALAA  AA          NA <NA>
## # ... with 336,766 more rows, and 6 more variables: manufacturer <chr>,
## #   model <chr>, engines <int>, seats <int>, speed <int>, engine <chr>
```

Matching Keys with Different Names

- The airport codes are in either `origin` or `dest` in the `flights2` table, and in `faa` in the `airports` table.
- Use, e.g., `by= c("dest" = "faa")`

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

```
## # A tibble: 336,776 x 15
##     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
```

Exercise

- Change the name of the `ID` column in `stats` to `Station`.
- With these modified tables, do a left-join of the `stats` and `station` tables.

Filtering Joins

- In past lectures we have use %in% for filtering a table according to a character string.
  - E.G., `gapminder %>% filter(country %in% c("Canada","United States"))`
- Filtering joins are an extension to filter a table according to another table.

```
top_dest <- flights2 %>% count(dest,sort=TRUE) %>% head(n=10)
print(top_dest,n=4)
```

```
## # A tibble: 10 x 2
##   dest      n
##   <chr> <int>
## 1 ORD   17283
## 2 ATL   17215
## 3 LAX   16174
## 4 BOS   15508
## # ... with 6 more rows
```

- The "old" way and the semi-join way:

```
flights2%>% filter(dest %in% top_dest$dest) %>% print(n=4)
```

```
## # A tibble: 141,145 x 8
##    year month   day  hour origin dest  tailnum carrier
##   <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>
## 1  2013     1     1     5 JFK    MIA   N619AA  AA
## 2  2013     1     1     6 LGA    ATL   N668DN  DL
## 3  2013     1     1     5 EWR    ORD   N39463  UA
## 4  2013     1     1     6 EWR    FLL   N516JB  B6
## # ... with 1.411e+05 more rows
```

```
flights2 %>% semi_join(top_dest) %>% print(n=4)
```

```
## # A tibble: 141,145 x 8
##    year month   day  hour origin dest  tailnum carrier
##   <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>
## 1  2013     1     1     5 JFK    MIA   N619AA  AA
## 2  2013     1     1     6 LGA    ATL   N668DN  DL
## 3  2013     1     1     5 EWR    ORD   N39463  UA
## 4  2013     1     1     6 EWR    FLL   N516JB  B6
## # ... with 1.411e+05 more rows
```

Notes

- The `n` column of the `top_dest` table used for filtering does not appear in the output.
- This ensures the same behaviour as the "old" way.
- The power of `semi_join()` is in matching to multiple columns, as in the following exercise.

Exercise

- From the original `flights` table, create a table called `top_dep_delay` comprised of the year-month-days with the 3 largest total delays, defined as the sum of the `dep_delay` variable.
  - Hints: use `group_by()` to group `flights` by year-month-day; use `summarize()` to compute total delays (watch out for missing values); use `arrange()` to sort on your total delays variable (you want to sort in descending order)
- Do a semi-join to filter `flights` to these days.

Set Operations

- An add-on to this chapter is the set operations, `intersect()`, `union()`, and `setdiff()`, which can act on pairs of tables.
- Illustrate with `intersect()`.

```
v1 <- c("apple","pen","pineapple"); v2 <- c("apple","orange","grape")
intersect(v1,v2)
```

```
## [1] "apple"
```

```
df1 <- tibble(x=c(1,2),y=c(1,1)); df2 <- tibble(x=c(1,1),y=c(1,2))
intersect(df1,df2)
```

```
## # A tibble: 1 x 2
##       x     y
##   <dbl> <dbl>
## 1     1     1
```