Relational Data with dplyr

Data Analysis and Visualization (Fall 2019)

Instructor: Debopriya Ghosh

Relations are defined between a pair of tables.

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.0     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(nycflights13)
```

Airlines help look up the full carrier name from abbreviated code.

```
data("airlines")
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.
```

Airports gives information about each airport, identified by faa airport code.

```
data("airports")
airports
```

```
## # 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 Municipa~  32.5  -85.7   264    -6 A     America/Chic~
##  3 06C   Schaumburg Regional    42.0  -88.1   801    -6 A     America/Chic~
##  4 06N   Randall Airport        41.4  -74.4   523    -5 A     America/New_~
##  5 09J   Jekyll Island Airport  31.1  -81.4    11    -5 A     America/New_~
##  6 0A9   Elizabethton Municip~  36.4  -82.2  1593    -5 A     America/New_~
##  7 0G6   Williams County Airp~  41.5  -84.5   730    -5 A     America/New_~
##  8 0G7   Finger Lakes Regiona~  42.9  -76.8   492    -5 A     America/New_~
##  9 0P2   Shoestring Aviation ~  39.8  -76.6  1000    -5 U     America/New_~
## 10 0S9   Jefferson County Intl  48.1 -123.    108    -8 A     America/Los_~
## # ... with 1,448 more rows
```

Planes gives information about each plane, identified by the tailnum.

```
data(planes)
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 win~ EMBRAER       EMB-1~       2    55    NA Turbo~
##  2 N102UW   1998 Fixed win~ AIRBUS INDUS~ A320-~       2   182    NA Turbo~
##  3 N103US   1999 Fixed win~ AIRBUS INDUS~ A320-~       2   182    NA Turbo~
##  4 N104UW   1999 Fixed win~ AIRBUS INDUS~ A320-~       2   182    NA Turbo~
##  5 N10575   2002 Fixed win~ EMBRAER       EMB-1~       2    55    NA Turbo~
##  6 N105UW   1999 Fixed win~ AIRBUS INDUS~ A320-~       2   182    NA Turbo~
##  7 N107US   1999 Fixed win~ AIRBUS INDUS~ A320-~       2   182    NA Turbo~
##  8 N108UW   1999 Fixed win~ AIRBUS INDUS~ A320-~       2   182    NA Turbo~
##  9 N109UW   1999 Fixed win~ AIRBUS INDUS~ A320-~       2   182    NA Turbo~
## 10 N110UW   1999 Fixed win~ AIRBUS INDUS~ A320-~       2   182    NA Turbo~
## # ... with 3,312 more rows
```

Weather gives the weather at each NYC airport for each hour.

```
data("weather")
weather
```

```
## # 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
##  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

The variables used to connect each pair of tables are called keys. A key is a variable that uniquely identifies an observation.

- Primary Key: Uniquely identifies an observation in its own table. For example, planes$tailnum is a primary key because it uniquely identifies each plane in the planes table.
- Foreign Key: Uniquely identifies an observation in another table. For example, flights$tailnum is a foreign key because it appears in the flights table where it matches each flight to a unique plane.

A variable can be both primary key and a foreign key. For example, origin is part of the weather primary key, and is also a forign key for the airport table.

Once you identify the primary keys, you can verify whether they uniquely identify each observation or not. One way to do so is to count() the primary keys and look for entries where n is greater than 1.

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

```
## # A tibble: 3,322 x 2
##    tailnum     n
##    <chr>   <int>
##  1 N10156      1
##  2 N102UW      1
##  3 N103US      1
##  4 N104UW      1
##  5 N10575      1
##  6 N105UW      1
##  7 N107US      1
##  8 N108UW      1
##  9 N109UW      1
## 10 N110UW      1
## # ... with 3,312 more rows
```

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

```
## # A tibble: 3 x 6
##    year month   day  hour origin     n
##   <dbl> <dbl> <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
```

```
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
```

Sometimes a table doesnot have an explicit primary key: each row is an observation, but no combination of variables identifies it. If a table lacks a primary key, it is useful to add one with mutate() and row_number(). This makes it easier to match observations if you want to perform filtering. This is called surrogate key.

A primary key and the corresponding foreign key in another table form a relation. Relations are typically one-to-many.

Mutating Joins

Allows to combine variables from two tables. It first matches observations by their keys, then copies across variables from one table to the other.

```
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 you want to add the full airline name to the flights2 data. You can combine airlines and flights2 data frames with left_join()

```
flights2 %>%
  select(-origin, -dest) %>%
  left_join(airlines, by = "carrier")
```

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

Alternative approach:

```
flights2 %>%
  select(-origin, -dest) %>%
  mutate(name = airlines$name[match(carrier,airlines$carrier)])
```

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

Inner Join

Simplest type of join. Matches a pair of observations whenever their keys are equal.

```
x = tribble(
  ~key, ~val_x,
  1,"x1",
  2,"x2",
  3, "x3"
)

x
```

```
## # A tibble: 3 x 2
##     key val_x
##   <dbl> <chr>
```

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

```
y = tribble(
  ~key, ~val_y,
  1, "y1",
  2, "y2",
  3, "y3"
)

y
```

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

```
x %>%
  inner_join(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    y3
```

Outer Joins

Inner join keeps observations that appear in both tables. An outer join keeps observations that appear at least one of the tables. There are four types of outer joins:

- left join keeps all observations in x
- right join keeps all observations in y
- full join keeps all observations in x and y

```
a = tribble(
  ~key, ~val_x,
  1,"a1",
  2,"a2",
  3, "a3"
)

a
```

```
## # A tibble: 3 x 2
##     key val_x
##   <dbl> <chr>
## 1     1 a1
## 2     2 a2
## 3     3 a3
```

```
b = tribble(
  ~key, ~val_x,
```

```
  1,"b1",
  2,"b2",
  4, "b3"
)
b
```

```
## # A tibble: 3 x 2
##     key val_x
##   <dbl> <chr>
## 1     1 b1
## 2     2 b2
## 3     4 b3
```

```
a %>%
  left_join(b,by = "key")
```

```
## # A tibble: 3 x 3
##     key val_x.x val_x.y
##   <dbl> <chr>   <chr>
## 1     1 a1      b1
## 2     2 a2      b2
## 3     3 a3      <NA>
```

```
a %>%
  right_join(b,by = "key")
```

```
## # A tibble: 3 x 3
##     key val_x.x val_x.y
##   <dbl> <chr>   <chr>
## 1     1 a1      b1
## 2     2 a2      b2
## 3     4 <NA>    b3
```

```
a %>%
  full_join(b,by = "key")
```

```
## # A tibble: 4 x 3
##     key val_x.x val_x.y
##   <dbl> <chr>   <chr>
## 1     1 a1      b1
## 2     2 a2      b2
## 3     3 a3      <NA>
## 4     4 <NA>    b3
```

Duplicate Keys

```
x = tribble(
  ~key, ~val_x,
  1, "x1",
  2, "x2",
  2, "x3",
  1,"x4"
)
x
```

```
## # A tibble: 4 x 2
```

```
##     key val_x
##   <dbl> <chr>
## 1     1 x1
## 2     2 x2
## 3     2 x3
## 4     1 x4
```

```
y = tribble(
  ~key, ~val_y,
  1, "y1",
  2,"y2"
)
y
```

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

```
left_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     2 x3    y2
## 4     1 x4    y1
```

When we join duplicated keys, we get all possible combinations or Cartesian Product.

Defining the Key Columns

So, far the pair of tables have been joined by a single variable, and the variable has the same name in both tables. That constraint was specfied using by = "key".

- The dafult by = NULL. It uses all variables that appear in both tables and is called the natural join. For example, 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
##    <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". For example flights and planes have year variables, but they mean different things, we only want to join by 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>
```

- by = c("a" = "b"). This will match variable a in table x to variable b in table y.Suppose we want to combine flights data with the airport data, which contains the location of each airport. Each flight has an origin and destination airport, so we need to specify which one we want to join to.

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

```
## # A tibble: 336,776 x 15
##     year month   day  hour origin dest  tailnum carrier name    lat   lon
##    <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>   <chr> <dbl> <dbl>
##  1  2013     1     1     5 EWR    IAH   N14228  UA      Geor~  30.0 -95.3
##  2  2013     1     1     5 LGA    IAH   N24211  UA      Geor~  30.0 -95.3
##  3  2013     1     1     5 JFK    MIA   N619AA  AA      Miam~  25.8 -80.3
##  4  2013     1     1     5 JFK    BQN   N804JB  B6      <NA>   NA    NA
##  5  2013     1     1     6 LGA    ATL   N668DN  DL      Hart~  33.6 -84.4
##  6  2013     1     1     5 EWR    ORD   N39463  UA      Chic~  42.0 -87.9
##  7  2013     1     1     6 EWR    FLL   N516JB  B6      Fort~  26.1 -80.2
##  8  2013     1     1     6 LGA    IAD   N829AS  EV      Wash~  38.9 -77.5
##  9  2013     1     1     6 JFK    MCO   N593JB  B6      Orla~  28.4 -81.3
## 10  2013     1     1     6 LGA    ORD   N3ALAA  AA      Chic~  42.0 -87.9
## # ... with 336,766 more rows, and 4 more variables: alt <int>, tz <dbl>,
## #   dst <chr>, tzone <chr>
```

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

```
## # A tibble: 336,776 x 15
##     year month   day  hour origin dest  tailnum carrier name    lat   lon
##    <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>   <chr> <dbl> <dbl>
```

```
##  1  2013     1     1     5 EWR    IAH   N14228  UA      Newa~  40.7 -74.2
##  2  2013     1     1     5 LGA    IAH   N24211  UA      La G~  40.8 -73.9
##  3  2013     1     1     5 JFK    MIA   N619AA  AA      John~  40.6 -73.8
##  4  2013     1     1     5 JFK    BQN   N804JB  B6      John~  40.6 -73.8
##  5  2013     1     1     6 LGA    ATL   N668DN  DL      La G~  40.8 -73.9
##  6  2013     1     1     5 EWR    ORD   N39463  UA      Newa~  40.7 -74.2
##  7  2013     1     1     6 EWR    FLL   N516JB  B6      Newa~  40.7 -74.2
##  8  2013     1     1     6 LGA    IAD   N829AS  EV      La G~  40.8 -73.9
##  9  2013     1     1     6 JFK    MCO   N593JB  B6      John~  40.6 -73.8
## 10  2013     1     1     6 LGA    ORD   N3ALAA  AA      La G~  40.8 -73.9
## # ... with 336,766 more rows, and 4 more variables: alt <int>, tz <dbl>,
## #   dst <chr>, tzone <chr>
```

Filtering Joins

semi_join(x,y) keeps all observations in x that have a match in y anti_join(x,y) drops all observations in x that have a match in y

```
# find top 10 most popular destinations
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
```

```
# find each flight that went to one of those destinations
flights %>%
  filter(dest %in% top_dest$dest)
```

```
## # A tibble: 141,145 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      542            540         2      923
##  2  2013     1     1      554            600        -6      812
##  3  2013     1     1      554            558        -4      740
##  4  2013     1     1      555            600        -5      913
##  5  2013     1     1      557            600        -3      838
##  6  2013     1     1      558            600        -2      753
##  7  2013     1     1      558            600        -2      924
##  8  2013     1     1      558            600        -2      923
##  9  2013     1     1      559            559         0      702
```

```
## 10  2013     1     1      600            600         0      851
## # ... with 141,135 more rows, and 12 more variables: 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>
```

Alternative:

```
flights %>%
  semi_join(top_dest)
```

```
## Joining, by = "dest"
```

```
## # A tibble: 141,145 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      542            540         2      923
##  2  2013     1     1      554            600        -6      812
##  3  2013     1     1      554            558        -4      740
##  4  2013     1     1      555            600        -5      913
##  5  2013     1     1      557            600        -3      838
##  6  2013     1     1      558            600        -2      753
##  7  2013     1     1      558            600        -2      924
##  8  2013     1     1      558            600        -2      923
##  9  2013     1     1      559            559         0      702
## 10  2013     1     1      600            600         0      851
## # ... with 141,135 more rows, and 12 more variables: 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>
```

Inverse of semi_join is anti_join. It keeps rows that does not match. For example, when connecting flights and planes, we are interested to know that there are many flights that do not have a match in planes.

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

```
## # A tibble: 722 x 2
##    tailnum     n
##    <chr>   <int>
##  1 <NA>     2512
##  2 N725MQ    575
##  3 N722MQ    513
##  4 N723MQ    507
##  5 N713MQ    483
##  6 N735MQ    396
##  7 NOEGMQ    371
##  8 N534MQ    364
##  9 N542MQ    363
## 10 N531MQ    349
## # ... with 712 more rows
```

Set Operations

```
df1 = tribble(
  ~x, ~y,
  1, 1,
  2, 1
)
df2 = tribble(
  ~x, ~y,
  1, 1,
  1, 2
)
```

```
intersect(df1,df2)
```

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

```
union(df1,df2)
```

```
## # A tibble: 3 x 2
##       x     y
##   <dbl> <dbl>
## 1     1     1
## 2     2     1
## 3     1     2
```

```
setdiff(df1,df2)
```

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

```
setdiff(df2,df1)
```

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