Joins

Daniel Anderson

To get started today, run the following code. We won't use it a lot, but we will use it some.

```
library(tidyverse)
install.packages("nycflights13")
library(nycflights13)
```

keys

- Variable(s) in common between two datasets to be joined.
- **Primary keys:** Uniquely identify observations in their dataset
- **Foreign keys:** Uniquely identify observations in other datasets.

A key can be more than one variable.

What's the primary key here?

Are there foreign keys?

```
library(haven)
ecls <- read_sav("./data/ecls-k_samp.sav") %>%
    mutate_if(is.labelled, as_factor)
ecls
```

```
## # A tibble: 984 × 33
##    child_id teacher_id school_id   k_type school_type    sex
##       <chr>      <chr>     <chr>   <fctr>      <fctr> <fctr>
## 1  0842021C    0842T02      0842 full-day      public   male
## 2  0905002C    0905T01      0905 full-day     private   male
## 3  0150012C    0150T01      0150 full-day     private female
## 4  0556009C    0556T01      0556 full-day     private female
## 5  0089013C    0089T04      0089 full-day      public   male
## 6  1217001C    1217T13      1217 half-day      public female
## 7  1092008C    1092T01      1092 half-day      public female
## 8  0083007C    0083T16      0083 full-day      public   male
## 9  1091005C    1091T02      1091 half-day     private   male
## 10 2006006C    2006T01      2006 full-day     private   male
## # ... with 974 more rows, and 27 more variables: ethnic <fctr>,
## #   famtype <fctr>, numsibs <dbl>, SES_cont <dbl>, SES_cat <fctr>,
```

Double-checking

```
ecls %>%
    count(child_id) %>%
    filter(n > 1)
```

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

What about here?

```
income_ineq <- read_csv("./data/incomeInequality_tidy.csv")
income_ineq
```

```
## # A tibble: 726 × 6
##     Year Number.thousands realGDPperCap PopulationK percentile    income
##    <int>            <int>         <dbl>       <int>      <dbl>     <dbl>
## 1   1947            37237      14117.32      144126       20.0  14243.00
## 2   1947            37237      14117.32      144126       40.0  22984.00
## 3   1947            37237      14117.32      144126       60.0  31166.00
## 4   1947            37237      14117.32      144126       80.0  44223.00
## 5   1947            37237      14117.32      144126       50.0  26764.14
## 6   1947            37237      14117.32      144126       90.0  41477.00
## 7   1947            37237      14117.32      144126       95.0  54172.00
## 8   1947            37237      14117.32      144126       99.0 134415.00
## 9   1947            37237      14117.32      144126       99.5 203001.00
## 10  1947            37237      14117.32      144126       99.9 479022.00
## # ... with 716 more rows
```

```
income_ineq %>%
    count(Year, percentile) %>%
    filter(n > 1)
```

```
## Source: local data frame [0 x 3]
## Groups: Year [0]
##
## # ... with 3 variables: Year <int>, percentile <dbl>, n <int>
```

Sometimes there is no key

- Generally in these cases there's an implicit id - the row numbers. For example:

```
flights
```

```
## # A tibble: 336,776 × 20
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1   2013     1     1      517            515         2      830
## 2   2013     1     1      533            529         4      850
## 3   2013     1     1      542            540         2      923
## 4   2013     1     1      544            545        -1     1004
## 5   2013     1     1      554            600        -6      812
## 6   2013     1     1      554            558        -4      740
## 7   2013     1     1      555            600        -5      913
## 8   2013     1     1      557            600        -3      709
## 9   2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 336,766 more rows, and 13 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>, new_id <int>
```

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

```
## Source: local data frame [29,768 x 5]
## Groups: year, month, day [365]
##
##     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
```

Create a key

- If there is no key, it's often helpful to add one. These are called *surrogate* keys.

```
flights <- flights %>%
    mutate(new_id = row_number())
flights[ ,c(1:3, ncol(flights))]
```

```
## # A tibble: 336,776 × 4
##     year month   day new_id
##    <int> <int> <int>  <int>
## 1   2013     1     1      1
## 2   2013     1     1      2
## 3   2013     1     1      3
## 4   2013     1     1      4
## 5   2013     1     1      5
## 6   2013     1     1      6
## 7   2013     1     1      7
## 8   2013     1     1      8
## 9   2013     1     1      9
## 10  2013     1     1     10
## # ... with 336,766 more rows
```

Mutating joins

- In *tidyverse*, we use `mutate()` to create new variables within a dataset.
- A mutating join works similarly, in that we're adding to variables to the existing dataset through a join.
- Two tables of data joined by a common key

Four types of joins

- `left_join`: Keep all the data in the left dataset, drop any non-matching cases from the right dataset.
- `right_join`: Keep all the data in the right dataset, drop any non-matching cases from the left dataset.
- `inner_join`: Keep only data that matches in both datasets
- `full_join`: Keep all the data in both datasets. This is also sometimes referred to as an *outer* join.

Illustrating the joins

- Consider the following hypothetical datasets to be merged

```
set.seed(100)
left_dataset <- tibble(key = 1:3, male = rbinom(3, 1, .5))
right_dataset <- tibble(key = c(1, 2, 4), sped = rbinom(3, 1, .5))
```

```
left_dataset
```

```
## # A tibble: 3 × 2
##     key  male
##   <int> <int>
## 1     1     0
## 2     2     0
## 3     3     1
```

```
right_dataset
```

```
## # A tibble: 3 × 2
##     key  sped
##   <dbl> <int>
## 1     1     0
## 2     2     0
## 3     4     0
```

What will happen with a left join?

```
left_join(left_dataset, right_dataset)
```

```
## Joining, by = "key"
```

```
## # A tibble: 3 × 3
##     key  male  sped
##   <dbl> <int> <int>
## 1     1     0     0
## 2     2     0     0
## 3     3     1    NA
```

What about a right join?

```
right_join(left_dataset, right_dataset)
```

```
## Joining, by = "key"
```

```
## # A tibble: 3 × 3
##     key  male  sped
##   <dbl> <int> <int>
## 1     1     0     0
## 2     2     0     0
## 3     4    NA     0
```

inner join?

```
inner_join(left_dataset, right_dataset)
```

```
## Joining, by = "key"
```

```
## # A tibble: 2 × 3
##     key  male  sped
##   <dbl> <int> <int>
## 1     1     0     0
## 2     2     0     0
```

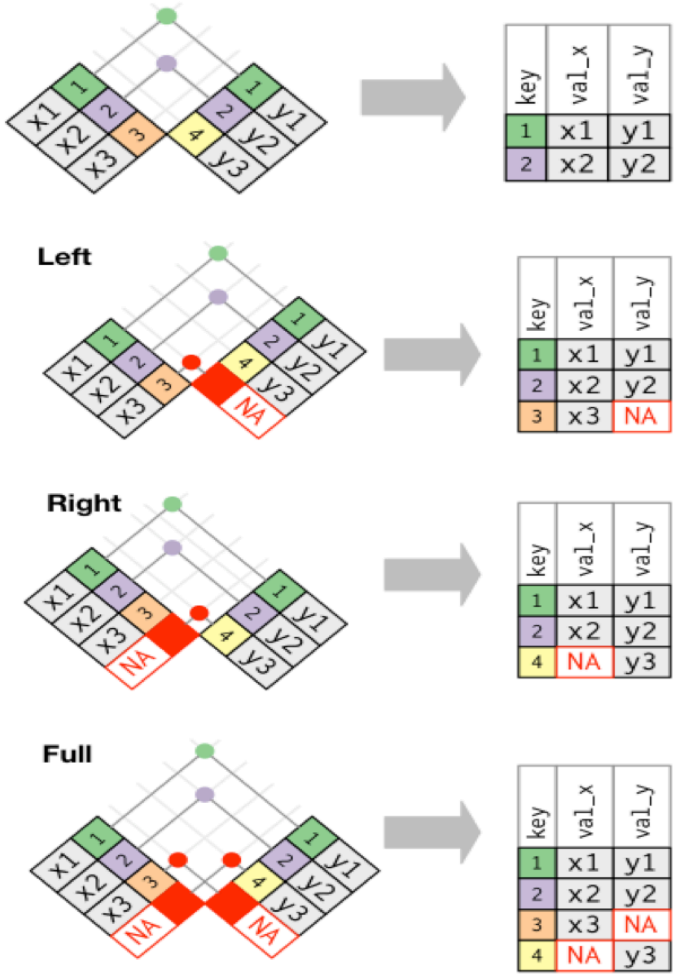
full join?

```
full_join(left_dataset, right_dataset)
```

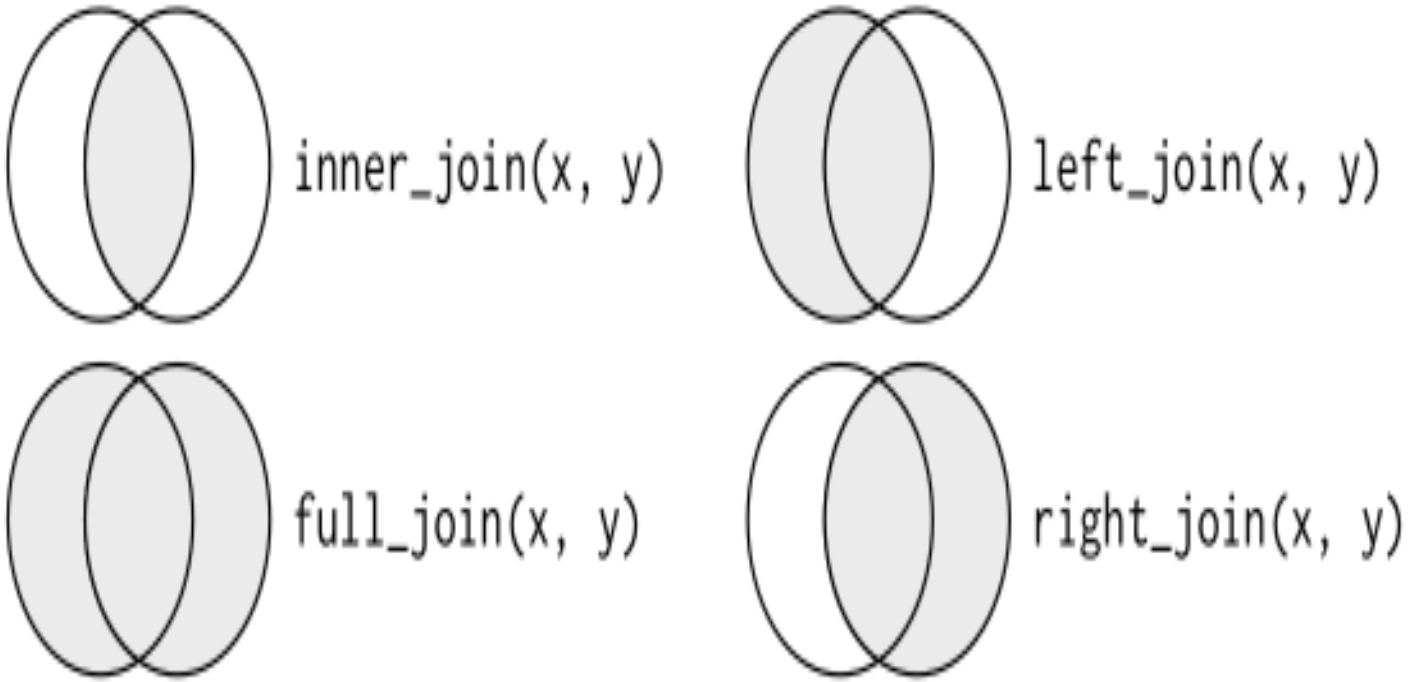
```
## Joining, by = "key"
```

```
## # A tibble: 4 × 3
##     key  male  sped
##   <dbl> <int> <int>
## 1     1     0     0
## 2     2     0     0
## 3     3     1    NA
## 4     4    NA     0
```

Joins graphically



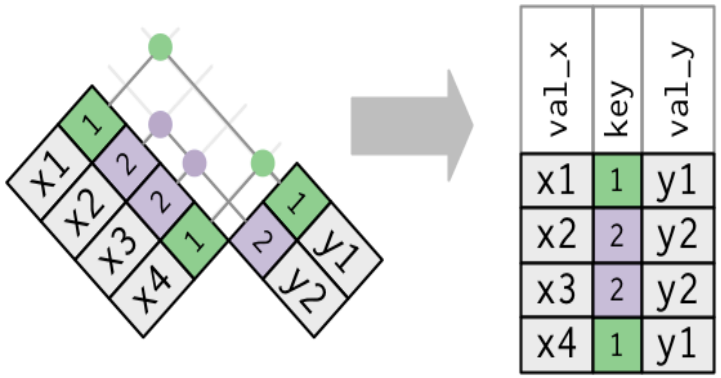
Alternative conceptualization



What if the key is not unique?

- As long as they are unique on one of the tables, there's no problem.

1. One table has duplicate keys. This is useful when you want to add in additional information as there is typically a one-to-many relationship.



val_x	key	val_y
x1	1	y1
x2	2	y2
x3	2	y2
x4	1	y1

Example

```
stu <- tibble(sid = rep(1:3, each = 3),
              score = c(10, 12, 15,
                         8,  9, 11,
                        12, 15, 17))
stu
```

```
## # A tibble: 9 × 2
##     sid score
##   <int> <dbl>
## 1     1    10
## 2     1    12
## 3     1    15
## 4     2     8
## 5     2     9
## 6     2    11
## 7     3    12
## 8     3    15
## 9     3    17
```

```
means <- stu %>%
    group_by(sid) %>%
    summarize(mean_score = mean(score))
means
```

```
## # A tibble: 3 × 2
##     sid mean_score
##   <int>      <dbl>
## 1     1  12.333333
## 2     2   9.333333
## 3     3  14.666667
```

```
left_join(stu, means)
```

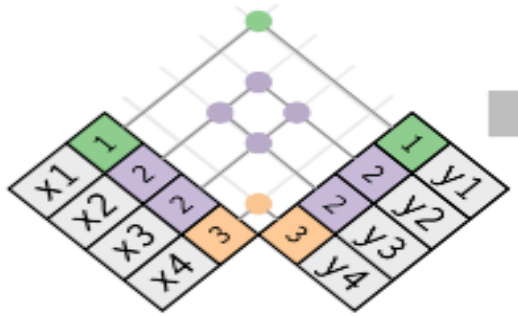
```
## Joining, by = "sid"
```

```
## # A tibble: 9 × 3
##     sid score mean_score
##   <int> <dbl>      <dbl>
## 1     1    10  12.333333
## 2     1    12  12.333333
## 3     1    15  12.333333
## 4     2     8   9.333333
## 5     2     9   9.333333
## 6     2    11   9.333333
## 7     3    12  14.666667
## 8     3    15  14.666667
## 9     3    17  14.666667
```

What if key is not unique to either table?

- Generally this is an error
- Result is probably not going to be what you want.

2. Both tables have duplicate keys. This is usually an error because in neither table do the keys uniquely identify an observation. When you join duplicated keys, you get all possible combinations, the Cartesian product:



key	val_x	val_y
1	x1	y1
2	x2	y2
2	x2	y3
2	x3	y2
2	x3	y3
3	x4	y4

Example

```
dems <- tibble(sid = rep(1:3, each = 3),
               sped = c(rep("no", 6), rep("yes", 3)))
dems
```

```
## # A tibble: 9 × 2
##     sid  sped
##   <int> <chr>
## 1     1    no
## 2     1    no
## 3     1    no
## 4     2    no
## 5     2    no
## 6     2    no
## 7     3   yes
## 8     3   yes
## 9     3   yes
```

```
left_join(stu, dems)
```

```
## Joining, by = "sid"
```

```
## # A tibble: 27 × 3
##      sid score  sped
##    <int> <dbl> <chr>
## 1      1    10    no
## 2      1    10    no
## 3      1    10    no
## 4      1    12    no
## 5      1    12    no
## 6      1    12    no
## 7      1    15    no
## 8      1    15    no
## 9      1    15    no
## 10     2     8    no
## # ... with 17 more rows
```

How do we fix this?

In this case it's pretty simple - select for distinct cases in the demo file. In others it's not so straight forward. But the important thing to remember is that you need to work toward making sure at least one of the keys is unique.

```
dems <- dems %>%
    distinct(sid, .keep_all = TRUE)
dems
```

```
## # A tibble: 3 × 2
##     sid  sped
##   <int> <chr>
## 1     1    no
## 2     2    no
## 3     3   yes
```

```
left_join(stu, dems)
```

```
## Joining, by = "sid"
```

```
## # A tibble: 9 × 3
##     sid score  sped
##   <int> <dbl> <chr>
## 1     1    10    no
## 2     1    12    no
## 3     1    15    no
## 4     2     8    no
## 5     2     9    no
## 6     2    11    no
## 7     3    12   yes
## 8     3    15   yes
## 9     3    17   yes
```

Another example

- Often you want to add summary info to your dataset.
- You can do this easily with by piping arguments

```
ecls <- ecls %>%
    group_by(school_id) %>%
    summarize(sch_pre_math = mean(T1MSCALE)) %>%
    left_join(ecls)
```

```
## Joining, by = "school_id"
```

```
ecls
```

```
## # A tibble: 984 × 34
##    school_id sch_pre_math child_id teacher_id   k_type school_type    sex
##        <chr>        <dbl>    <chr>      <chr>   <fctr>      <fctr> <fctr>
## 1       0001      20.4580 0001010C    0001T01 full-day      public   male
## 2       0002      14.9770 0002010C    0002T01 half-day      public female
## 3       0009      18.8200 0009026C    0009T01 half-day      public   male
## 4       0009      18.8200 0009014C    0009T02 half-day      public   male
## 5       0009      18.8200 0009005C    0009T01 half-day      public   male
## 6       0013      42.3210 0013003C    0013T01 full-day     private female
## 7       0016      17.5510 0016004C    0016T01 half-day      public   male
## 8       0016      17.5510 0016009C    0016T01 half-day      public female
## 9       0022      17.8465 0022005C    0022T01 half-day      public   male
## 10      0022      17.8465 0022014C    0022T03 half-day      public female
## # ... with 974 more rows, and 27 more variables: ethnic <fctr>,
## #   famtype <fctr>, numsibs <dbl>, SES_cont <dbl>, SES_cat <fctr>,
## #   age <dbl>, T1RSCALE <dbl>, T1MSCALE <dbl>, T1GSCALE <dbl>,
## #   T2RSCALE <dbl>, T2MSCALE <dbl>, T2GSCALE <dbl>, IRTreadgain <dbl>,
## #   IRTmathgain <dbl>, IRTgkgain <dbl>, T1ARSLIT <dbl>, T1ARSMAT <dbl>,
## #   T1ARSGEN <dbl>, T2ARSLIT <dbl>, T2ARSMAT <dbl>, T2ARSGEN <dbl>,
## #   ARSlitgain <dbl>, ARSmathgain <dbl>, ARSgkgain <dbl>,
## #   testdate1 <date>, testdate2 <date>, elapse <dbl>
```

Default behavior & changing it

- By default, the `*_join` functions will use all columns with common names as keys.

```
flights2 <- flights %>%
  select(year:day, hour, origin, dest, tailnum, carrier)
flights2[1:2, ]
```

```
## # A tibble: 2 × 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
```

```
weather[1:2, ]
```

```
## # A tibble: 2 × 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     0 37.04 21.92 53.97      230   10.35702
## 2    EWR  2013     1     1     1 37.04 21.92 53.97      230   13.80936
## # ... with 5 more variables: wind_gust <dbl>, precip <dbl>,
## #   pressure <dbl>, visib <dbl>, time_hour <dttm>
```

```
left_join(flights2, weather)
```

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

```
## # A tibble: 336,776 × 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    NA    NA    NA
## 2   2013     1     1     5    LGA   IAH  N24211      UA    NA    NA    NA
## 3   2013     1     1     5    JFK   MIA  N619AA      AA    NA    NA    NA
## 4   2013     1     1     5    JFK   BQN  N804JB      B6    NA    NA    NA
## 5   2013     1     1     6    LGA   ATL  N668DN      DL 39.92 26.06 57.33
## 6   2013     1     1     5    EWR   ORD  N39463      UA    NA    NA    NA
## 7   2013     1     1     6    EWR   FLL  N516JB      B6 39.02 26.06 59.37
## 8   2013     1     1     6    LGA   IAD  N829AS      EV 39.92 26.06 57.33
## 9   2013     1     1     6    JFK   MCO  N593JB      B6 39.02 26.06 59.37
## 10  2013     1     1     6    LGA   ORD  N3ALAA      AA 39.92 26.06 57.33
## # ... 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>
```

Use only some vars?

- If we were joining *flights2* and *planes*, we would not want to use the year variable in the join, because it means different things in each dataset.

```
planes
```

```
## # A tibble: 3,322 × 9
##    tailnum  year                type            manufacturer     model
##      <chr> <int>                <chr>                   <chr>     <chr>
## 1   N10156  2004 Fixed wing multi engine              EMBRAER EMB-145XR
## 2   N102UW  1998 Fixed wing multi engine AIRBUS INDUSTRIE  A320-214
## 3   N103US  1999 Fixed wing multi engine AIRBUS INDUSTRIE  A320-214
## 4   N104UW  1999 Fixed wing multi engine AIRBUS INDUSTRIE  A320-214
## 5   N10575  2002 Fixed wing multi engine              EMBRAER EMB-145LR
## 6   N105UW  1999 Fixed wing multi engine AIRBUS INDUSTRIE  A320-214
## 7   N107US  1999 Fixed wing multi engine AIRBUS INDUSTRIE  A320-214
## 8   N108UW  1999 Fixed wing multi engine AIRBUS INDUSTRIE  A320-214
## 9   N109UW  1999 Fixed wing multi engine AIRBUS INDUSTRIE  A320-214
## 10  N110UW  1999 Fixed wing multi engine AIRBUS INDUSTRIE  A320-214
## # ... with 3,312 more rows, and 4 more variables: engines <int>,
## #   seats <int>, speed <int>, engine <chr>
```

- How? Specify the variables with `by`

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

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

Mismatched names?

- What if you had data to merge like this?

```
stu
```

```
## # A tibble: 9 × 2
##     sid score
##   <int> <dbl>
## 1     1    10
## 2     1    12
## 3     1    15
## 4     2     8
## 5     2     9
## 6     2    11
## 7     3    12
## 8     3    15
## 9     3    17
```

```
names(dems)[1] <- "stu_id"
dems
```

```
## # A tibble: 3 × 2
##   stu_id  sped
##    <int> <chr>
## 1      1    no
## 2      2    no
## 3      3   yes
```

Join w/mismatched names

```
left_join(stu, dems, by = c("sid" = "stu_id"))
```

```
## # A tibble: 9 × 3
##     sid score  sped
##   <int> <dbl> <chr>
## 1     1    10    no
## 2     1    12    no
## 3     1    15    no
## 4     2     8    no
## 5     2     9    no
## 6     2    11    no
## 7     3    12   yes
## 8     3    15   yes
## 9     3    17   yes
```

Relation to `base::merge()`

dplyr	merge
`inner_join(x, y)`	`merge(x, y)`
`left_join(x, y)`	`merge(x, y, all.x = TRUE)`
`right_join(x, y)`	`merge(x, y, all.y = TRUE)` ,
`full_join(x, y)`	`merge(x, y, all.x = TRUE, all.y = TRUE)`

Benefits of `*_join`

Hadley Wickham

☆ **Other recipients:** roger.j...@gmail.com

---

join is a plyr function. dplyr has inner_join, left_join, semi_join and anti_join. The advantages of the dplyr versions over merge are:

* rows are kept in existing order
* much faster
* tells you what keys you're merging by (if you don't supply)
* also work with database tables.

Hadley

Filtering joins

- `semi_join()` works just like `left_join` or `inner_join` but you don't actually add the variables.
- Let's filter classrooms with extremely high math pretest average scores.
    - First, calculate averages

```
av_pre_mth <- ecls %>%
    group_by(teacher_id, k_type) %>%
    summarize(av_pre_mth = mean(T1MSCALE))
av_pre_mth
```

```
## Source: local data frame [707 x 3]
## Groups: teacher_id [?]
##
##    teacher_id   k_type av_pre_mth
##         <chr>   <fctr>      <dbl>
## 1     0001T01 full-day    20.4580
## 2     0002T01 half-day    14.9770
## 3     0009T01 half-day    17.6475
## 4     0009T02 half-day    21.1650
## 5     0013T01 full-day    42.3210
```

- Next, filter for means 3 standard deviations above the mean.

```
extr_high <- av_pre_mth %>%
    ungroup() %>%
    filter(av_pre_mth > (mean(av_pre_mth) + 3*sd(av_pre_mth)))
extr_high
```

```
## # A tibble: 8 × 3
##   teacher_id   k_type av_pre_mth
##        <chr>   <fctr>      <dbl>
## 1    0013T01 full-day    42.3210
## 2    0078T04 half-day    45.7500
## 3    0162T02 half-day    42.3180
## 4    0360T01 full-day    41.4220
## 5    0384T03 full-day    41.2900
## 6    0663T01 full-day    42.8455
## 7    0944T03 half-day    45.3710
## 8    1045T02 full-day    40.7340
```

- Finally, use `semi_join` to filter.

```
extr_high_ecls <- semi_join(ecls, extr_high)
```

```
## Joining, by = c("teacher_id", "k_type")
```

```
extr_high_ecls
```

```
## # A tibble: 10 × 34
##    school_id sch_pre_math child_id teacher_id   k_type school_type    sex
##        <chr>        <dbl>    <chr>      <chr>   <fctr>      <fctr> <fctr>
## 1       0013     42.32100 0013003C    0013T01 full-day     private female
## 2       0078     25.64000 0078020C    0078T04 half-day      public female
## 3       0162     30.52425 0162009C    0162T02 half-day      public female
## 4       0360     41.42200 0360014C    0360T01 full-day      public female
## 5       0384     30.40000 0384014C    0384T03 full-day      public female
## 6       0663     42.84550 0663006C    0663T01 full-day     private   male
## 7       0663     42.84550 0663012C    0663T01 full-day     private female
## 8       0944     45.37100 0944017C    0944T03 half-day     private female
## 9       1045     35.45325 1045015C    1045T02 full-day     private   male
## 10      1045     35.45325 1045020C    1045T02 full-day     private female
## # ... with 27 more variables: ethnic <fctr>, famtype <fctr>,
## #   numsibs <dbl>, SES_cont <dbl>, SES_cat <fctr>, age <dbl>,
```

Filtering joins

- `anti_join()` does the opposite, keeping any rows that do **not** match.

```
extr_low_ecls <- anti_join(ecls, extr_high)
```

```
## Joining, by = c("teacher_id", "k_type")
```

```
extr_low_ecls
```

```
## # A tibble: 974 × 34
##    school_id sch_pre_math child_id teacher_id   k_type school_type    sex
##        <chr>        <dbl>    <chr>      <chr>   <fctr>      <fctr> <fctr>
## 1       3116     12.35600 3116019C    3116T03 full-day      public   male
## 2       3110     17.47400 3110009C    3110T02 full-day      public   male
## 3       3107     19.77000 3107011C    3107T03 full-day      public female
## 4       3103     18.38450 3103013C    3103T04 full-day      public female
## 5       3094     28.62975 3094018C    3094T01 half-day      public   male
## 6       3094     28.62975 3094008C    3094T01 half-day      public female
## 7       3094     28.62975 3094019C    3094T01 half-day      public female
## 8       3084     19.79700 3084024C    3084T03 half-day      public female
## 9       3084     19.79700 3084011C    3084T02 half-day      public   male
## 10      3084     19.79700 3084017C    3084T02 half-day      public female
```

Check: Expected?

- Always a good idea to make sure the number of rows you end up with is what you expected.

```
nrow(ecls)
```

```
## [1] 984
```

```
nrow(extr_high)
```

```
## [1] 8
```

```
nrow(extr_high_ecls) == nrow(extr_high)
```

```
## [1] FALSE
```

- Why is the above false? Let's investigate

```
nrow(extr_high)
```

```
## [1] 8
```

```
extr_high_ecls
```

```
## # A tibble: 10 × 34
##    school_id sch_pre_math child_id teacher_id   k_type school_type    sex
##        <chr>        <dbl>    <chr>      <chr>   <fctr>      <fctr> <fctr>
## 1       0013     42.32100 0013003C    0013T01 full-day     private female
## 2       0078     25.64000 0078020C    0078T04 half-day      public female
## 3       0162     30.52425 0162009C    0162T02 half-day      public female
## 4       0360     41.42200 0360014C    0360T01 full-day      public female
## 5       0384     30.40000 0384014C    0384T03 full-day      public female
## 6       0663     42.84550 0663006C    0663T01 full-day     private   male
## 7       0663     42.84550 0663012C    0663T01 full-day     private female
## 8       0944     45.37100 0944017C    0944T03 half-day     private female
## 9       1045     35.45325 1045015C    1045T02 full-day     private   male
## 10      1045     35.45325 1045020C    1045T02 full-day     private female
## # ... with 27 more variables: ethnic <fctr>, famtype <fctr>,
## #   numsibs <dbl>, SES_cont <dbl>, SES_cat <fctr>, age <dbl>,
## #   T1RSCALE <dbl>, T1MSCALE <dbl>, T1GSCALE <dbl>, T2RSCALE <dbl>,
## #   T2MSCALE <dbl>, T2GSCALE <dbl>, IRTreadgain <dbl>, IRTmathgain <dbl>,
```

- Did these teachers really only teach one students?
    - Check first teacher

```
ecls %>%
  group_by(teacher_id) %>%
  count() %>%
  filter(teacher_id == "0013T01")
```

```
## # A tibble: 1 × 2
##   teacher_id     n
##        <chr> <int>
## 1    0013T01     1
```

Answer: YES!

- 6/8 teachers with a μ_c > 3 sd above μ taught only 1 student.
- Remaining two teachers taught only two students

```
extr_high_ecls %>%
  count(teacher_id)
```

```
## # A tibble: 8 × 2
##   teacher_id     n
##        <chr> <int>
## 1    0013T01     1
## 2    0078T04     1
## 3    0162T02     1
## 4    0360T01     1
## 5    0384T03     1
## 6    0663T01     2
## 7    0944T03     1
## 8    1045T02     2
```

- Using what we've learned, how could we subset the data to teachers with only one student?
- Try it out!

One method

```
tch_one <- ecls %>%
  group_by(teacher_id) %>%
  count() %>%
  filter(n == 1) %>%
  semi_join(x = ecls, y = .)
```

```
## Joining, by = "teacher_id"
```

Did it work?

```
tch_one %>%
  count(teacher_id) %>%
  count(n)
```

```
## # A tibble: 1 × 2
##       n    nn
##   <int> <int>
## 1     1   514
```

Another filter join example

- This is an example I particularly like, which I came up with before knowing that `semi_join()` or `anti_join()` existed.
- Trying to select one cohort of students from the seda data

```
seda <- read_csv("./data/district means national-referenced by year grade subject (long file).
seda
```

```
## # A tibble: 326,246 × 10
##     leaid                leaname  fips stateabb  year grade mean_link_ela
##     <int>                  <chr> <int>    <chr> <int> <int>         <dbl>
## 1  100002 ALABAMA YOUTH SERVICES     1       AL  2009     8      210.5474
## 2  100002 ALABAMA YOUTH SERVICES     1       AL  2011     8      231.6601
## 3  100002 ALABAMA YOUTH SERVICES     1       AL  2012     8      226.1813
## 4  100005       ALBERTVILLE CITY     1       AL  2009     3      204.4659
## 5  100005       ALBERTVILLE CITY     1       AL  2009     4      207.4045
## 6  100005       ALBERTVILLE CITY     1       AL  2009     5      216.8594
## 7  100005       ALBERTVILLE CITY     1       AL  2009     6      229.0331
## 8  100005       ALBERTVILLE CITY     1       AL  2009     7      242.0856
## 9  100005       ALBERTVILLE CITY     1       AL  2009     8      250.2386
## 10 100005       ALBERTVILLE CITY     1       AL  2010     3      205.7993
## # ... with 326,236 more rows, and 3 more variables: se_link_ela <dbl>,
```

- First, create a data frame with the criteria for the filter

```
criteria <- tibble(year = 2009:2013, grade = 3:7)
criteria
```

```
## # A tibble: 5 × 2
##    year grade
##   <int> <int>
## 1  2009     3
## 2  2010     4
## 3  2011     5
## 4  2012     6
## 5  2013     7
```

- Then use a `right_join` with *seda* and *criteria*

```
seda <- right_join(seda, criteria)
```

```
## Joining, by = c("year", "grade")
```

```
seda %>% arrange(leaid)
```

```
## # A tibble: 54,311 × 10
##     leaid          leaname  fips stateabb  year grade mean_link_ela
##     <int>            <chr> <int>    <chr> <int> <int>         <dbl>
## 1  100005 ALBERTVILLE CITY     1       AL  2009     3      204.4659
## 2  100005 ALBERTVILLE CITY     1       AL  2010     4      219.0224
## 3  100005 ALBERTVILLE CITY     1       AL  2011     5      220.5850
## 4  100005 ALBERTVILLE CITY     1       AL  2012     6      235.1851
## 5  100005 ALBERTVILLE CITY     1       AL  2013     7      243.9144
## 6  100006  MARSHALL COUNTY     1       AL  2009     3      205.9022
## 7  100006  MARSHALL COUNTY     1       AL  2010     4      217.7184
## 8  100006  MARSHALL COUNTY     1       AL  2011     5      227.1329
## 9  100006  MARSHALL COUNTY     1       AL  2012     6      237.4675
## 10 100006  MARSHALL COUNTY     1       AL  2013     7      254.0359
## # ... with 54,301 more rows, and 3 more variables: se_link_ela <dbl>,
## #   mean_link_math <dbl>, se_link_math <dbl>
```

Why is this so beneficial?

- What would it look like if we tried to select for only this one cohort using `filter()`?

Let's do some practicing

Go here and copy the data:

- http://stat545.com/bit001_dplyr-cheatsheet.html#why-the-cheatsheet

```
superheroes <- "
    name, alignment, gender,         publisher
 Magneto,       bad,   male,            Marvel
   Storm,      good, female,            Marvel
Mystique,       bad, female,            Marvel
  Batman,      good,   male,                DC
   Joker,       bad,   male,                DC
Catwoman,       bad, female,                DC
 Hellboy,      good,   male, Dark Horse Comics
"
superheroes <- read_csv(superheroes, trim_ws = TRUE, skip = 1)

publishers <- "
  publisher, yr_founded
         DC,        1934
     Marvel,        1939
      Image,        1992
"
publishers <- read_csv(publishers, trim_ws = TRUE, skip = 1)
```

The *superhero* data

- What's the key here?

```
superheroes
```

```
## # A tibble: 7 × 4
##       name alignment gender         publisher
##      <chr>     <chr>  <chr>             <chr>
## 1  Magneto       bad   male            Marvel
## 2    Storm      good female            Marvel
## 3 Mystique       bad female            Marvel
## 4   Batman      good   male                DC
## 5    Joker       bad   male                DC
## 6 Catwoman       bad female                DC
## 7  Hellboy      good   male Dark Horse Comics
```

The *publishers* data

- What's the key here?

```
publishers
```

```
## # A tibble: 3 × 2
##   publisher yr_founded
##       <chr>      <int>
## 1        DC       1934
## 2    Marvel       1939
## 3     Image       1992
```

Look at the data side-by-side

```
superheroes
```

```
## # A tibble: 7 × 4
##       name alignment gender         publish
##      <chr>     <chr>  <chr>           <ch
## 1  Magneto       bad   male          Marv
## 2    Storm      good female          Marv
## 3 Mystique       bad female          Marv
## 4   Batman      good   male             DC
## 5    Joker       bad   male             DC
## 6 Catwoman       bad female             DC
## 7  Hellboy      good   male Dark Horse Comics
```

```
publishers
```

```
## # A tibble: 3 × 2
##   publisher yr_founded
##       <chr>      <int>
## 1        DC       1934
## 2    Marvel       1939
## 3     Image       1992
```

Let's do some predicting

- For each of the following, make a prediction as to what will result, then run it to see if you were right.
  - `inner_join(superheroes, publishers)`
  - `inner_join(publishers, superheroes)`
  - `left_join(superheroes, publishers)`
  - `left_join(publishers, superheroes)`
  - `right_join(superheroes, publishers)`
  - `right_join(publishers, superheroes)`
  - `full_join(superheroes, publishers)`
  - `full_join(publishers, superheroes)`