Joins and logic

Week 4

AEM 2850 / 5850 : R for Business Analytics
Cornell Dyson
Spring 2023

Acknowledgements: Grant McDermott, Jenny Bryan, R4DS (2e)

Announcements

Reminders:

- Submit assignments via canvas
    - Lab-03 was due yesterday (Monday) at 11:59pm

Questions before we get started?

Plan for this week

Prologue

Joins (Tuesday)

- example-04-1

Logic (Thursday)

- example-04-2

Prologue

What sports do we watch?

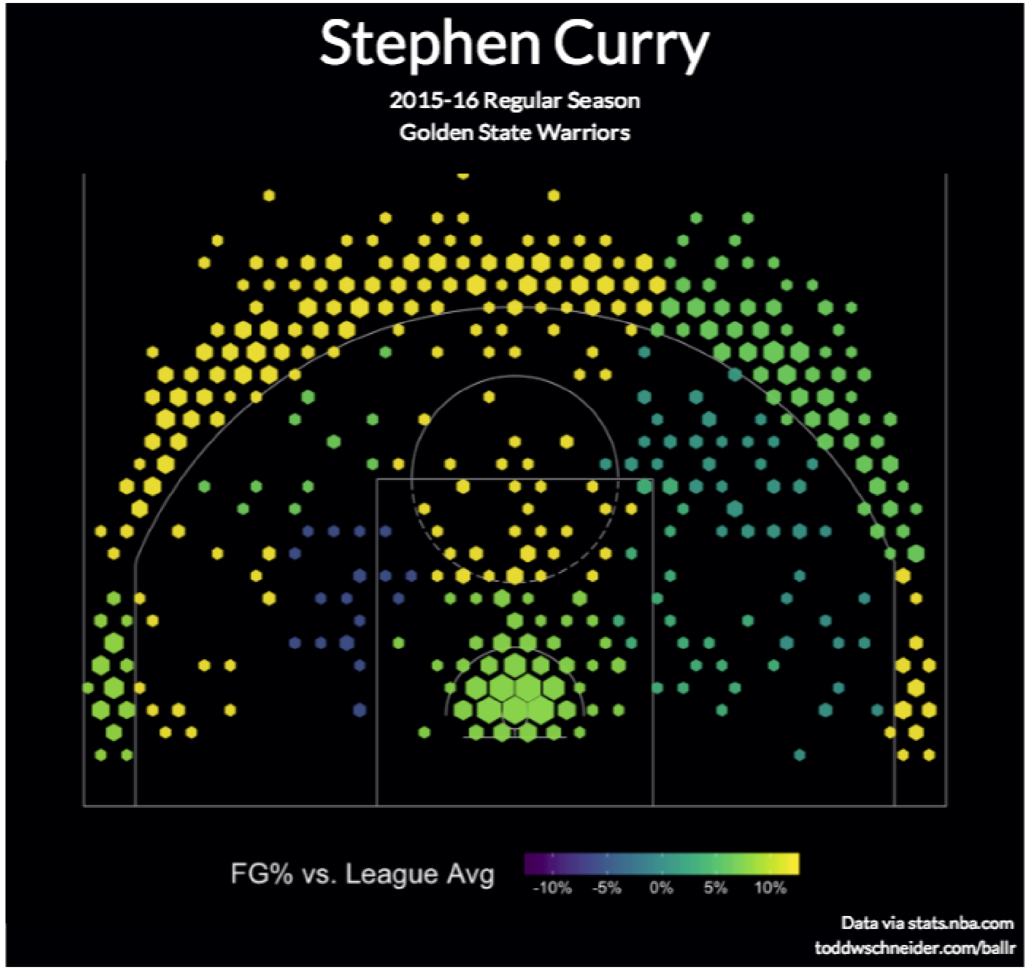
Take a guess: what's the most popular spectator sport among classmates?

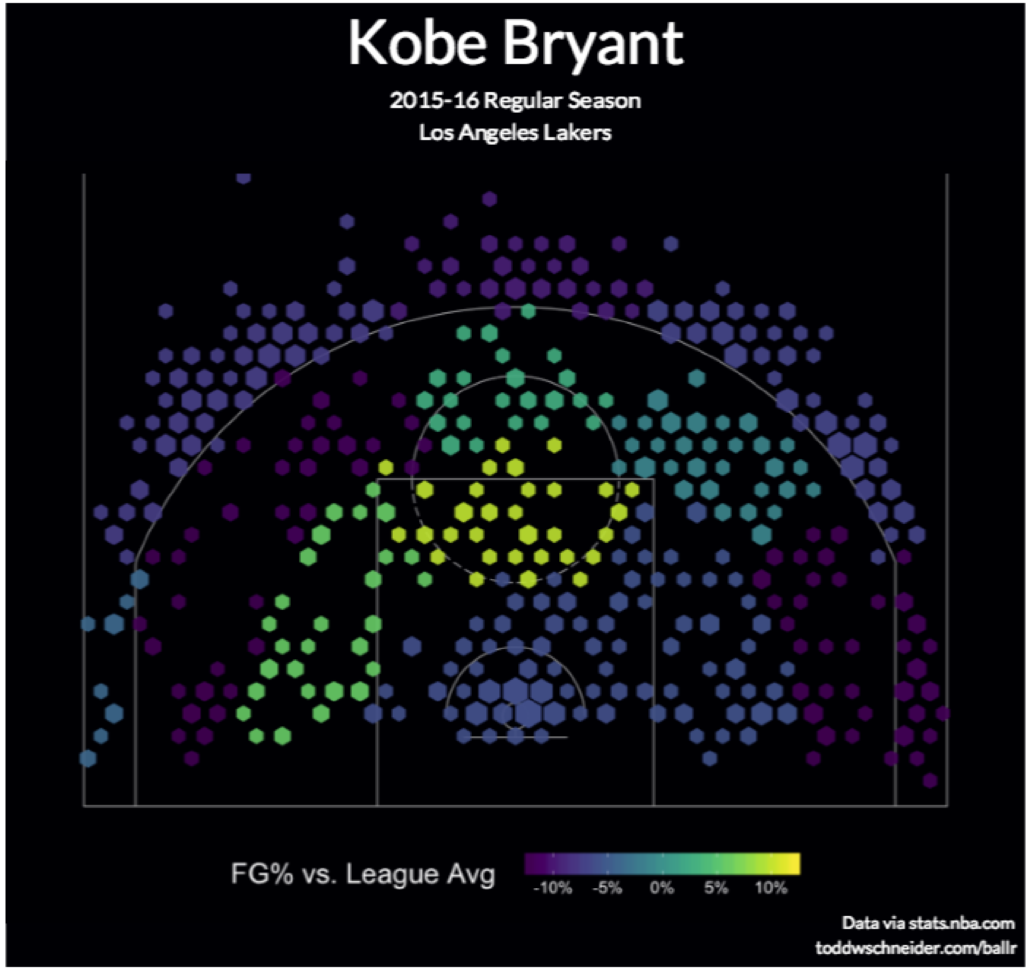
```
##  [1] "football"   "basketball" "volleyball" "soccer"     NA          
##  [6] "baseball"   "football"   "basketball" "basketball" "football"  
## [11] "gymnastics" "gymnastics" "squash"     NA           "basketball"
## [16] "gymnastics" "tennis"     "baseball"   "basketball" "basketball"
## [21] NA           "soccer"     "basketball" "baseball"   "tennis"    
## [26] "polo"       "hockey"     "basketball" "gymnastics" "soccer"    
## [31] "tennis"     NA           "soccer"     "cricket"    "tennis"    
## [36] "tennis"     "badminton"  "football"   "football"
```

Let's `count` and `arrange` to get the top 3:

```
## # A tibble: 3 × 2
##   sport          n
##   <chr>      <int>
## 1 basketball     8
## 2 football       5
## 3 tennis         5
```

R can be used for sports analytics, too!





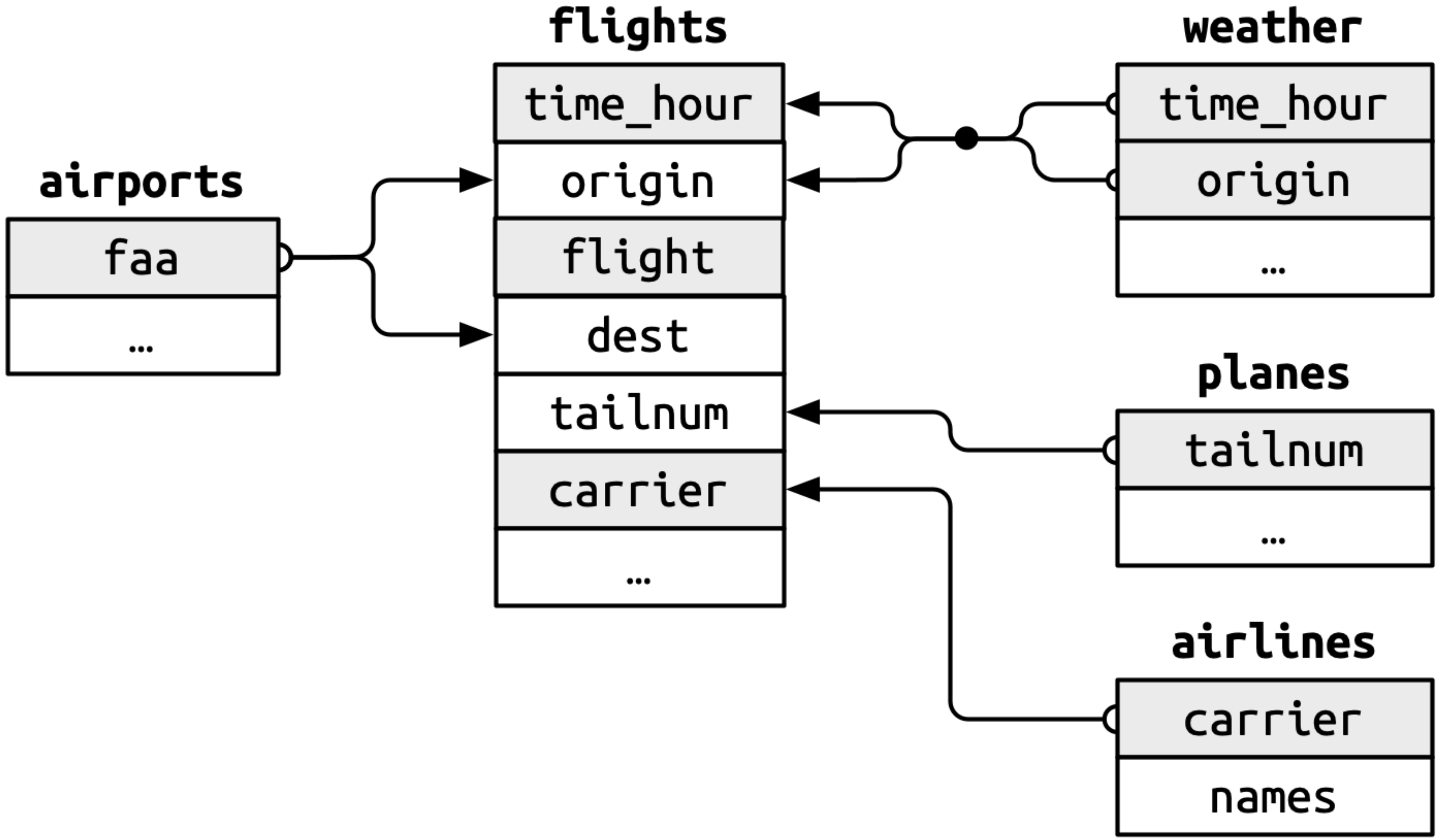
Joins

What are relational data?

Multiple tables of data with pairwise relations

Relations across >2 data tables are determined by the relations between each pair

Relational data example: **`nycflights13`**



Relational data verbs from dplyr

1. **Mutating joins**: add new variables
    - `left_join()`
    - `right_join()`
    - `inner_join()`
    - `full_join()`
2. **Filtering joins**: filter observations
    - `semi_join()`
    - `anti_join()`

Joins

Let's learn these join commands using two small data frames

```
superheroes
```

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

```
publishers
```

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

1) dplyr::left_join(x, y)

```
left_join(superheroes, publishers)
```

```
## Joining with `by = join_by(publisher)`
```

```
## # A tibble: 7 × 4
##   name     alignment publisher         year_founded
##   <chr>    <chr>     <chr>                    <int>
## 1 Magneto  bad       Marvel                    1939
## 2 Storm    good      Marvel                    1939
## 3 Mystique bad       Marvel                    1939
## 4 Batman   good      DC                        1934
## 5 Joker    bad       DC                        1934
## 6 Catwoman bad       DC                        1934
## 7 Hellboy  good      Dark Horse Comics           NA
```

`left_join` is a **mutating join**: it adds variables to `x`

`left_join` returns all rows from `x`

2) dplyr::right_join(x, y)

```
right_join(superheroes, publishers)
```

```
## Joining with `by = join_by(publisher)`
```

```
## # A tibble: 7 × 4
##   name     alignment publisher year_founded
##   <chr>    <chr>     <chr>            <int>
## 1 Magneto  bad       Marvel            1939
## 2 Storm    good      Marvel            1939
## 3 Mystique bad       Marvel            1939
## 4 Batman   good      DC                1934
## 5 Joker    bad       DC                1934
## 6 Catwoman bad       DC                1934
## 7 <NA>     <NA>      Image             1992
```

`right_join` is a **mutating join**: it adds variables to `y`

`right_join` returns all rows from `y`

3) dplyr::inner_join(x, y)

```
inner_join(superheroes, publishers)
```

```
## Joining with `by = join_by(publisher)`
```

```
## # A tibble: 6 × 4
##   name     alignment publisher year_founded
##   <chr>    <chr>     <chr>            <int>
## 1 Magneto  bad       Marvel            1939
## 2 Storm    good      Marvel            1939
## 3 Mystique bad       Marvel            1939
## 4 Batman   good      DC                1934
## 5 Joker    bad       DC                1934
## 6 Catwoman bad       DC                1934
```

How is `inner_join` different from `left_join` and `right_join`?

`inner_join` returns all rows in `x` **AND** `y`

4) dplyr::full_join(x, y)

```
full_join(superheroes, publishers) # how many rows do you think this will produce?
```

```
## Joining with `by = join_by(publisher)`

## # A tibble: 8 × 4
##   name      alignment publisher         year_founded
##   <chr>     <chr>     <chr>                    <int>
## 1 Magneto   bad       Marvel                    1939
## 2 Storm     good      Marvel                    1939
## 3 Mystique  bad       Marvel                    1939
## 4 Batman    good      DC                        1934
## 5 Joker     bad       DC                        1934
## 6 Catwoman  bad       DC                        1934
## 7 Hellboy   good      Dark Horse Comics           NA
## 8 <NA>      <NA>      Image                     1992
```

`full_join` returns all rows in `x` **OR** `y`

5) dplyr::semi_join(x, y)

```
superheroes
```

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

```
semi_join(superheroes, publishers)
```

```
## Joining with `by = join_by(publisher)`

## # A tibble: 6 × 3
##   name      alignment publisher
##   <chr>     <chr>     <chr>
## 1 Magneto   bad       Marvel
## 2 Storm     good      Marvel
## 3 Mystique  bad       Marvel
## 4 Batman    good      DC
## 5 Joker     bad       DC
## 6 Catwoman  bad       DC
```

`semi_join` is a **filtering join**: it keeps observations in `x` that have a match in `y`

Note that the variables do not change

6) dplyr::anti_join(x, y)

```
superheroes
```

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

```
anti_join(superheroes, publishers)
```

```
## Joining with `by = join_by(publisher)`

## # A tibble: 1 × 3
##   name    alignment publisher
##   <chr>   <chr>     <chr>
## 1 Hellboy good      Dark Horse Comics
```

`anti_join` is a **filtering join**: it keeps obs. in `x` that **DO NOT** have a match in `y`

Note that the variables do not change

Key variables

How do `dplyr` join commands know what variables to use as **keys**?

By default, `*_join()` uses all variables that are common across `x` and `y`

```
intersect(names(superheroes), names(publishers)) # variable used for matching before
```

```
## [1] "publisher"
```

Or, we can specify what to join by: `*_join(..., by = join_by(publisher))`

Note: before `dplyr` 1.1.0, the syntax was: `*_join(..., by = "publisher")`

Exploring keys

```r
library(nycflights13) # let's explore keys using the nycflights13 data
flights |> print(n = 8) # print the first 8 rows of flights
```

```
## # A tibble: 336,776 × 19
##    year month   day dep_time sched_dep…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵ carrier
##   <int> <int> <int>    <int>       <int>   <dbl>   <int>   <int>   <dbl> <chr>
## 1  2013     1     1      517         515       2     830     819      11 UA
## 2  2013     1     1      533         529       4     850     830      20 UA
## 3  2013     1     1      542         540       2     923     850      33 AA
## 4  2013     1     1      544         545      -1    1004    1022     -18 B6
## 5  2013     1     1      554         600      -6     812     837     -25 DL
## 6  2013     1     1      554         558      -4     740     728      12 UA
## 7  2013     1     1      555         600      -5     913     854      19 B6
## 8  2013     1     1      557         600      -3     709     723     -14 EV
## # … with 336,768 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   ¹sched_dep_time, ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
```

Exploring keys

```r
planes # print the first 10 rows of planes
```

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

Let's perform a left join on flights and planes

```
left_join(flights, planes) |>
  select(year:dep_time, arr_time, carrier:tailnum, type, model) |> # keep text to one slide
  print(n = 5) # just to save vertical space on the slide
```

```
## Joining with `by = join_by(year, tailnum)`
```

```
## # A tibble: 336,776 × 10
##    year month   day dep_time arr_time carrier flight tailnum type  model
##   <int> <int> <int>    <int>    <int> <chr>    <int> <chr>   <chr> <chr>
## 1  2013     1     1      517      830 UA        1545 N14228  <NA>  <NA>
## 2  2013     1     1      533      850 UA        1714 N24211  <NA>  <NA>
## 3  2013     1     1      542      923 AA        1141 N619AA  <NA>  <NA>
## 4  2013     1     1      544     1004 B6         725 N804JB  <NA>  <NA>
## 5  2013     1     1      554      812 DL         461 N668DN  <NA>  <NA>
## # … with 336,771 more rows
```

Uh-oh! What's up with `type` and `model`?

Uh-oh!

As before, `dplyr` guessed which columns to join on

It uses columns with the same name:

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

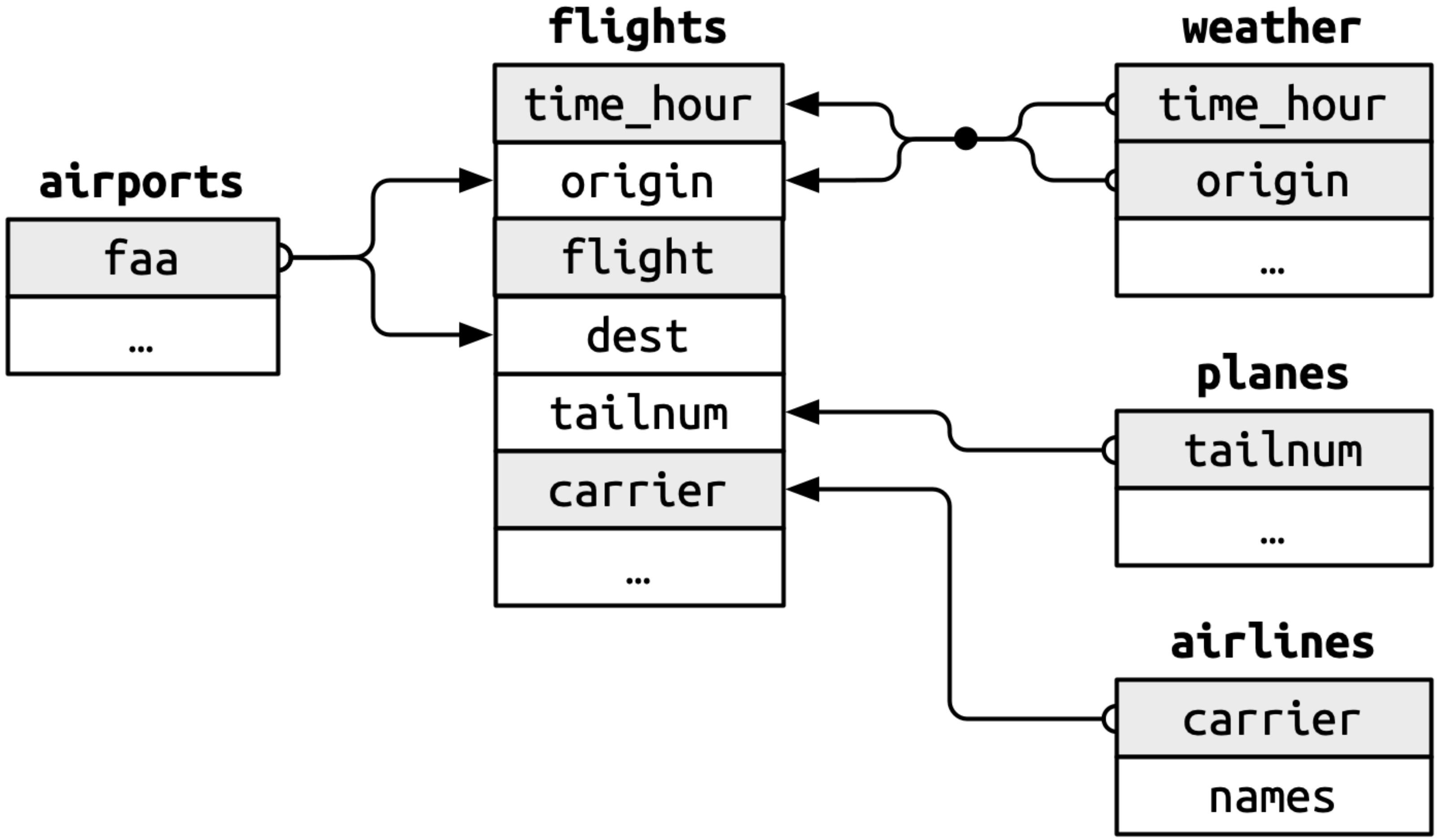
Does anyone see a potential problem here?

The variable `year` does not have a consistent meaning across the datasets

In `flights` it refers to the *year of flight*, in `planes` it refers to *year of construction*

Luckily we can avoid this by using the argument `by = join_by(...)`

What should we join flights and planes by?



Specifying join keys

We just need to be explicit in the join call by using the by argument

```
left_join(flights,
          planes |> rename(year_built = year), # not necessary w/ below line, but helpful
          by = join_by(tailnum) # be specific about the joining column
          ) |>
  select(year, month:dep_time, carrier, flight, tailnum, year_built, type, model) |>
  print(n = 5) # just to save vertical space on the slide
```

```
## # A tibble: 336,776 × 10
##    year month   day dep_time carrier flight tailnum year_built type        model
##   <int> <int> <int>    <int> <chr>    <int> <chr>        <int> <chr>       <chr>
## 1  2013     1     1      517 UA        1545 N14228        1999 Fixed wing… 737-…
## 2  2013     1     1      533 UA        1714 N24211        1998 Fixed wing… 737-…
## 3  2013     1     1      542 AA        1141 N619AA        1990 Fixed wing… 757-…
## 4  2013     1     1      544 B6         725 N804JB        2012 Fixed wing… A320…
## 5  2013     1     1      554 DL         461 N668DN        1991 Fixed wing… 757-…
## # … with 336,771 more rows
```

Specifying join keys

What happens if we don't rename `year` before this join?

```r
left_join(flights,
          planes, # not renaming "year" to "year_built" this time
          by = join_by(tailnum)
          ) |>
  select(contains("year"), month:dep_time, arr_time, carrier, flight, tailnum, type, model) |>
  print(n = 4) # just to save vertical space on the slide
```

```
## # A tibble: 336,776 × 11
##   year.x year.y month   day dep_time arr_time carrier flight tailnum type  model
##    <int>  <int> <int> <int>    <int>    <int> <chr>    <int> <chr>   <chr> <chr>
## 1   2013   1999     1     1      517      830 UA        1545 N14228  Fixe… 737-…
## 2   2013   1998     1     1      533      850 UA        1714 N24211  Fixe… 737-…
## 3   2013   1990     1     1      542      923 AA        1141 N619AA  Fixe… 757-…
## 4   2013   2012     1     1      544     1004 B6         725 N804JB  Fixe… A320…
## # … with 336,772 more rows
```

What is `year.x`? What is `year.y`?

Summary of key verbs so far

Key verbs

Import

readr

1. `read_csv`
2. `write_csv`

readxl

1. `read_excel`

Tidy

tidyr

1. `pivot_longer`
2. `pivot_wider`
3. `separate`
4. `unite`

Join

dplyr

1. `left_join`
2. `right_join`
3. `inner_join`
4. `full_join`
5. `semi_join`
6. `anti_join`

Transform

dplyr

1. `filter`
2. `arrange`
3. `select`
4. `mutate`
5. `summarize`

link to example-04-1

Logic

Logic slides will be added for Thursday

See https://aem2850.toddgerarden.com/content/04-content/ for the updated version of these slides

link to example-04-2