Filtering Joins

Max Turgeon

SCI 2000–Introduction to Data Science

Lecture Objectives

- Understand the difference between a mutating join and a filtering join
- Be able to recognize when to use each type
- Be able to transform datasets using set operations

Motivation

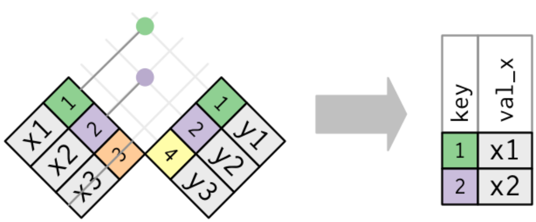
- In the previous lecture, we talked about **mutating joins**.
    - Create new dataset by combining two datasets with a common variable
- Today we will talk about **filtering joins**.
    - Filter a dataset based on its relationship with another dataset
- For completeness, we will also talk about set operations that can be used with relational data.

Filtering joins in general

- The starting point is still the same:
    - We have two `data.frame`s `x` and `y`
    - They have a variable in common that allows us to match rows across
- In filtering joins, we want to filter the rows of `x` based on their relationship with the rows of `y`.
    - In particular, the output of a filtering join is a *subset* of `x`.

Semijoin

- In a **semijoin**, we only keep the rows of x with a corresponding match in y



Example i

```r
library(tidyverse)

df_beers <- read_csv("beers.csv")
df_breweries <- read_csv("breweries.csv")
```

```r
# Top 5 states for # breweries
state_top5 <- df_breweries %>%
  count(state) %>%
  top_n(5)
```

Example ii

```
state_top5
```

```
## # A tibble: 5 x 2
##   state     n
##   <chr> <int>
## 1 CA       39
## 2 CO       47
## 3 MI       32
## 4 OR       29
## 5 TX       28
```

Example iii

```r
breweries_top5 <- semi_join(df_breweries,
                            state_top5)
```

```r
breweries_top5
```

```
## # A tibble: 175 x 4
## brewery_id name city state
## <dbl> <chr> <chr> <chr>
## 1 3 Mike Hess Brewing Company San Diego CA
## 2 4 Fort Point Beer Company San Francisco CA
## 3 6 Great Divide Brewing Company Denver CO
```

Example iv

```
## 4 7 Tapistry Brewing Bridgman MI
## 5 8 Big Lake Brewing Holland MI
## 6 9 The Mitten Brewing Company Grand Rapids MI
## 7 10 Brewery Vivant Grand Rapids MI
## 8 11 Petoskey Brewing Petoskey MI
## 9 12 Blackrocks Brewery Marquette MI
## 10 13 Perrin Brewing Company Comstock Park MI
## # ... with 165 more rows
```

Example v

```
# Only keep beers from these states
semi_join(df_beers,
          breweries_top5,
          by = "brewery_id") %>%
  count(style, sort = TRUE)
```

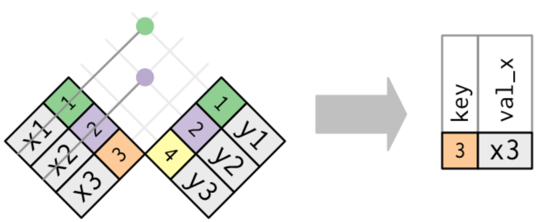
```
## # A tibble: 86 x 2
##    style                              n
##    <chr>                          <int>
##  1 American IPA                     141
##  2 American Pale Ale (APA)           90
```

Example vi

```
## 3 American Amber / Red Ale            57
## 4 American Double / Imperial IPA      43
## 5 American Blonde Ale                 38
## 6 American Pale Wheat Ale             38
## 7 Saison / Farmhouse Ale              24
## 8 American Brown Ale                  21
## 9 Cider                               21
## 10 American Stout                     20
## # ... with 76 more rows
```

Antijoin

- In an **antijoin**, we only keep the rows of x *without* a corresponding match in y



Example i

```r
# Let's look at the other states
breweries_nottop5 <- anti_join(df_breweries,
                               state_top5)
```

```r
breweries_nottop5
```

```
## # A tibble: 383 x 4
## brewery_id name city state
## <dbl> <chr> <chr> <chr>
## 1 0 NorthGate Brewing Minneapolis MN
## 2 1 Against the Grain Brewery Louisville KY
```

Example ii

```
## 3 2 Jack's Abby Craft Lagers Framingham MA
## 4 5 COAST Brewing Company Charleston SC
## 5 16 Flat 12 Bierwerks Indianapolis IN
## 6 17 Tin Man Brewing Company Evansville IN
## 7 18 Black Acre Brewing Co.  Indianapolis IN
## 8 19 Brew Link Brewing Plainfield IN
## 9 20 Bare Hands Brewery Granger IN
## 10 21 Three Pints Brewing Martinsville IN
## # ... with 373 more rows
```

Example iii

```r
# Only keep beers from these states
semi_join(df_beers,
          breweries_nottop5,
          by = "brewery_id") %>%
  count(style, sort = TRUE)
```

```
## # A tibble: 92 x 2
##    style                          n
##    <chr>                      <int>
##  1 American IPA                 283
##  2 American Pale Ale (APA)      155
```

Example iv

```
##  3 American Amber / Red Ale          76
##  4 American Blonde Ale               70
##  5 American Double / Imperial IPA    62
##  6 American Pale Wheat Ale           59
##  7 American Brown Ale                49
##  8 American Porter                   49
##  9 Fruit / Vegetable Beer            36
## 10 Witbier                           31
## # ... with 82 more rows
```

Exercise

Filter the dataset `flights` from the `nycflights13` package to only show flights with planes that have flown at least 100 flights.

Solution i

```r
library(nycflights13)

planes100 <- flights %>%
  count(tailnum) %>%
  filter(n >= 100)
```

```r
flights100 <- semi_join(flights,
                        planes100)
```

Solution ii

```r
# Do we get flights with missing
# tail number?
flights100 %>%
  filter(is.na(tailnum)) %>%
  nrow
```

```
## [1] 2512
```

Solution iii

```
# We can remove these NAs from planes100
planes100 <- filter(planes100,
                    !is.na(tailnum))
# Or we can remove them from flights100
flights100 <- filter(flights100,
                     !is.na(tailnum))
```

Some tips about joins

- You can join using more than one variable:

```
inner_join(x, y, by = c("var1", "var2"))
```

- You can join even when the same variable is named differently:

```
inner_join(x, y, by = c("name1" = "name2"))
```

Set operations i

- Here, the setup is slightly different.
    - We still have two `data.frame`s `x` and `y`.
    - But we assume they have **exactly** the same variables.
- We want to create a new dataset `z` that will also have the same variables as `x` and `y`.
- There are three different set operations:
    - **Union**: `z` has the unique observations from `x` and `y`.
    - **Intersection**: `z` has the observations common between `x` and `y`.
    - **Set difference**: `z` has the observations from `x` that are not in `y`.

Set operations ii

```
library(tidyverse)
df1 <- tibble(
  x = c(1, 2),
  y = c(1, 1)
)
df2 <- tibble(
  x = c(1, 1),
  y = c(1, 2)
)
```

Set operations iii

```
# Note that we get 3 rows, not 4
# because of duplicates
union(df1, df2)
```

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

Set operations iv

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

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

Set operations v

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

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

Set operations vi

```r
# The order is important!
setdiff(df2, df1)
```

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

Exercise

Find the states with at least 30 breweries. Create a dataset that contains information about beers from these states. Using linear regression, investigate whether there is a significant difference between the average ABV for beers from these states.

Solution i

- There are several ways of doing this, but a key observation is that we need the variable `state` to appear in the final dataset, otherwise we can't use it as a covariate.
- This suggests that the final dataset should be created using a *mutating* join.
- Given that we only want beers from some states, we also want to choose an **inner join**.
- Finally, the inner join should be between `df_beers` and the subset of `df_breweries` corresponding to these top states.

Solution ii

```
# One solution: group by state
# and use n() inside filter
breweries_30 <- df_breweries %>%
  group_by(state) %>%
  filter(n() >= 30) # n() counts per group

dataset <- inner_join(df_beers,
                      breweries_30,
                      by = "brewery_id")
```

Solution iii

```
count(dataset, state, sort = TRUE)
```

```
## # A tibble: 3 x 2
##   state     n
##   <chr> <int>
## 1 CO      265
## 2 CA      183
## 3 MI      162
```

Solution iv

```
fit <- lm(abv ~ state, data = dataset)
fit
```

```
##
## Call:
## lm(formula = abv ~ state, data = dataset)
##
## Coefficients:
## (Intercept)      stateCO      stateMI
##    0.061082     0.002290     0.002295
```

Solution v

```r
confint(fit)
```

```
##                     2.5 %      97.5 %
## (Intercept)  0.0589595040 0.063205331
## stateCO     -0.0005010600 0.005080225
## stateMI     -0.0008575131 0.005447645
```

Solution vi

```
# Alternatively, we can use a semijoin
# to create breweries_30
breweries_top <- df_breweries %>%
  count(state) %>%
  filter(n >= 30)

breweries_30 <- semi_join(df_breweries,
                          breweries_top)
```