Lab 10 - Merging Data

Ignacio Cabezudo

November 2, 2017

Using your own dataset (which may include more than one table) carry out the following data cleaning steps. Knit together the PDF document and commit both the Lab 10 RMD file and the PDF document to Git. Push the changes to GitHub so both documents are visible in your public GitHub repository.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readr)


#Setup a filter for later
gtd_filter <- c("iyear","country_txt", "latitude","longitude", "nkill", "gname" )


###### Load in the data to be defined by the filter
gtd_full <- read_csv("GTD FULL DB.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   eventid = col_double(),
##   iyear = col_integer(),
##   imonth = col_integer(),
##   iday = col_integer(),
##   extended = col_integer(),
##   country = col_integer(),
##   region = col_integer(),
##   latitude = col_double(),
##   longitude = col_double(),
##   specificity = col_integer(),
##   vicinity = col_integer(),
##   crit1 = col_integer(),
##   crit2 = col_integer(),
##   crit3 = col_integer(),
##   doubtterr = col_integer(),
##   alternative = col_integer(),
##   multiple = col_integer(),
##   success = col_integer(),
##   suicide = col_integer(),
```

```
##   attacktype1 = col_integer()
##   # ... with 44 more columns
## )
```

```
## See spec(...) for full column specifications.
```

```
## Warning in rbind(names(probs), probs_f): number of columns of result is not
## a multiple of vector length (arg 1)
```

```
## Warning: 246 parsing failures.
## row # A tibble: 5 x 5 col     row       col               expected actual              file expected
## ... ................. ... ..................................................................... ........
## See problems(...) for more details.
```

```
dat <- gtd_full %>%
       select(gtd_filter)
colnames(dat) <- c("year", "country", "latitude", "longitude", "nkill", "gname")
```

1. For your poster project, do you have multiple tables you'd like to join together to create your complete dataset? If so, describe what each table represents.

I don't think that I want to join them together. One table is a list of several events that each have corresponding years and several datapoints describing the event. The other is a small 33x5 index of a rate for 4 countries. I guess I could add a column on to my table of events called GDPR, then note the GDPR during each event.

2. What is/are your primary key(s)? If you have more than one table in your data, what is/are your foreign key(s)? Do your primary key(s) and foreign key(s) have the same name? If not, what does this mean for the way you need to specify potential data merges?

For this exercise I'll make two new df's called dat and mutant_dat. dat will be my pre-filtered df with all events in the GTD, but only 6 catagories. mutant_dat will be an object I create that has at least 1 death and happened after 1995. It will consist of several summary stats of dat and the keys will be country.

3. If you do not need to merge tables to create your final dataset, create a new dataset from your original dataset with a `grouped_by()` summary of your choice. You will use this separate dataset to complete the following exercises.

```
#Group the data by country, then filter out all events with no deaths, and give
#a summary of the median & IQR death toll for the countries


mutant_dat <- dat %>%
  group_by(country) %>%
  filter(nkill > 0 & year > 1995) %>%
  summarise(IQR(nkill), mean(nkill), n_distinct(gname), max(nkill))

#set column names for new data

colnames(mutant_dat) <- c("country", "IQR_deaths", "mean_deaths", "groups", "nkill")
```

If you are merging separate tables as part of your data manipulation process, are your keys of the same data type? If not, what are the differences? Figure out the appropriate coercion process(es) and carry out the steps below.

The keys are part of the same type

4. Perform each version of the mutating joins (don't forget to specify the `by` argument) and print the results to the console. Describe what each join did to your datasets and what the resulting data table

looks like. For those joining two separate datasets, did any of these joins result in your desired final dataset? Why or why not?

```
mutant_right <- right_join(mutant_dat, dat, by = "country")
head(mutant_right)
```

```
## # A tibble: 6 x 10
##              country IQR_deaths mean_deaths groups nkill.x  year latitude
##                <chr>      <dbl>       <dbl>  <int>   <dbl> <int>    <dbl>
## 1 Dominican Republic        0.5    1.500000      2       3  1970 18.45679
## 2             Mexico        3.0    3.354167     14      45  1970 19.43261
## 3        Philippines        1.0    2.259924     34     116  1970 15.47860
## 4             Greece        0.0    7.000000      8      66  1970 37.98377
## 5              Japan        0.0   19.000000      1      19  1970 33.58041
## 6      United States        2.0   46.042857     18    1383  1970 37.00511
## # ... with 3 more variables: longitude <dbl>, nkill.y <int>, gname <chr>
```

The right join adds the 170,350 rows from dat to mutant_right as expected. The data for nkill shows up as nkill.x and nkill.y The problem here is that the nkill variable in mutant_dat is an aggregate of all deaths in a country, where as the nkill.y is individual event death numbers. I can solve this by renaming the mutant nkill to nkill_total or something like this. The years that I excluded when I created the mutant_dat object are now included.

```
#Left join
mutant_left <- left_join(mutant_dat, dat, by = "country")
head(mutant_left)
```

```
## # A tibble: 6 x 10
##       country IQR_deaths mean_deaths groups nkill.x  year latitude
##         <chr>      <dbl>       <dbl>  <int>   <dbl> <int>    <dbl>
## 1 Afghanistan          4    4.293063     20     240  1973 34.53306
## 2 Afghanistan          4    4.293063     20     240  1979 34.53306
## 3 Afghanistan          4    4.293063     20     240  1979 33.55000
## 4 Afghanistan          4    4.293063     20     240  1979 34.34194
## 5 Afghanistan          4    4.293063     20     240  1987       NA
## 6 Afghanistan          4    4.293063     20     240  1988 34.53306
## # ... with 3 more variables: longitude <dbl>, nkill.y <int>, gname <chr>
```

The left join drops several entries. I'm honestly not exactly sure why. Oddly enough, the order in which the list shows up is different, as if the left join organized by country alphabetically. The years that I excluded when I created the mutant_dat object are now included.

```
mutant_inner <- inner_join(mutant_dat, dat, by = "country")
head(mutant_inner)
```

```
## # A tibble: 6 x 10
##       country IQR_deaths mean_deaths groups nkill.x  year latitude
##         <chr>      <dbl>       <dbl>  <int>   <dbl> <int>    <dbl>
## 1 Afghanistan          4    4.293063     20     240  1973 34.53306
## 2 Afghanistan          4    4.293063     20     240  1979 34.53306
## 3 Afghanistan          4    4.293063     20     240  1979 33.55000
## 4 Afghanistan          4    4.293063     20     240  1979 34.34194
## 5 Afghanistan          4    4.293063     20     240  1987       NA
## 6 Afghanistan          4    4.293063     20     240  1988 34.53306
## # ... with 3 more variables: longitude <dbl>, nkill.y <int>, gname <chr>
```

The inner join honestly seems to have done the exact same thing as the left join. The years that I excluded when I created the mutant_dat object are now included.

```
mutant_full <- full_join(mutant_dat, dat, by = "country")
head(mutant_full)
```

```
## # A tibble: 6 x 10
##       country IQR_deaths mean_deaths groups nkill.x  year latitude
##         <chr>      <dbl>       <dbl>  <int>   <dbl> <int>    <dbl>
## 1 Afghanistan          4    4.293063     20     240  1973 34.53306
## 2 Afghanistan          4    4.293063     20     240  1979 34.53306
## 3 Afghanistan          4    4.293063     20     240  1979 33.55000
## 4 Afghanistan          4    4.293063     20     240  1979 34.34194
## 5 Afghanistan          4    4.293063     20     240  1987       NA
## 6 Afghanistan          4    4.293063     20     240  1988 34.53306
## # ... with 3 more variables: longitude <dbl>, nkill.y <int>, gname <chr>
```

The full join seems to mimic the right join, in the same way the inner join mimiced the left join. The years that I excluded when I created the mutant_dat object are now included.

5. Do the same thing with the filtering joins. What was the result? Give an example of a case in which a `semi_join()` or an `anti_join()` might be used with your primary dataset

```
mutant_semi <- semi_join(mutant_dat, dat, by = "country")
head(mutant_semi)
```

```
## # A tibble: 6 x 5
##       country IQR_deaths mean_deaths groups nkill
##         <chr>      <dbl>       <dbl>  <int> <dbl>
## 1 Afghanistan          4    4.293063     20   240
## 2     Albania          2    2.222222      1     6
## 3     Algeria          6    6.675439     30   256
## 4      Angola         11   15.733945      7   259
## 5   Argentina          0    1.000000      2     1
## 6     Armenia          5    3.600000      3     8
```

This basically leaves mutant_dat alone and does nothing, as there are no matches.

```
mutant_anti <- anti_join(dat, mutant_dat, by = "country")
head(mutant_anti)
```

```
## # A tibble: 6 x 6
##    year            country latitude longitude nkill
##   <int>              <chr>    <dbl>     <dbl> <int>
## 1  1970 East Germany (GDR) 52.51667 13.400000    NA
## 2  1970 East Germany (GDR) 52.51667 13.400000    NA
## 3  1970 West Germany (FRG) 48.13913 11.580186     1
## 4  1970 West Germany (FRG) 48.13913 11.580186     7
## 5  1970 West Germany (FRG) 50.11145  8.680615     0
## 6  1970 West Germany (FRG) 48.13913 11.580186     0
## # ... with 1 more variables: gname <chr>
```

When I anti join these two with mutant_dat set to a and dat set to b, I get nothing, on a hunch I decided I should choose the larger database to be a, and I was correct, and found a list of 1300 events that don't occur in my mutant set. They do not appear because they either have an nkill = 0, or they happened before 1995.

6. What happens when you apply the set operations joins to your tables? Are these functions useful for you for this project? Explain why or why not. If not, give an example in which one of them might be usefully applied to your data.

```
#intersect(dat, mutant_dat)
#union(mutant_dat, dat)
#setdiff(mutant_dat,dat)
```

Errors, all errors. I won't be using this function as I'm comparing two vastly different data types.

7. If you have any reason to compare tables, apply `setequal()` below. What were the results?

I'm not going to be comparing them in the way that setequal will help with.

8. What is the purpose of binding data and why might you need to take extra precaution when carrying out this specific form of data merging? If your data requires any binding, carry out the steps below and describe what was accomplished by your merge.

Binding data is use to appened to an existing dataframe. This could very helpful when constructing a continues database. Adding rows doesn't seem to be dangerous at all, however adding columns presents difficulties because of the need to match them to rows.

9. Do you need to merge multiple tables together using the same type of merge? If so, utilize the `reduce()` function from the `purrr` package to carry out the appropriate merge below.

Nope

10. Are there any other steps you need to carry out to further clean, transform, or merge your data into one, final, tidy dataset? If so, describe what they are and carry them out below.

I'm very happy with my data as it is, then again, 2 or 3 labs ago I feel like I said the same thing. I honestly wish I had data that made better use of the mutates as I feel given the size of my database I most likely missed some of the benefits of these tools.