Lab 10 - Merging Data

Johnathan Hsu

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.

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 do! They are data_95_96 (1995-1996)...all the way up to data_2015. They are individual sentences from each year.

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?

They all have the sae name. I do not have to specify anything. The keys have the same name. The data has been merged and you can see the work in "scripts/merge_data.r".

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.

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.

My keys are the same datatype after I swapped them out. If you look in merge_data.r I do a lot of the coersion and other necessary steps. Data is saved in num form so I could refer to the codebook as needed.

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?

I really don't need to join my datasets, I will practice though.

```
# tidyverse and readr for reading csv and the functions.
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------------ tidyv
```

```
## ✓ ggplot2 2.2.1     ✓ purrr   0.2.4
## ✓ tibble  1.3.4     ✓ dplyr   0.7.4
## ✓ tidyr   0.7.2     ✓ stringr 1.2.0
## ✓ readr   1.1.1     ✓ forcats 0.2.0
```

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

```
library(readr)

# read in data
setwd("~/monitoring-federal-criminal-sentences")
data <- read_csv("clean_data/collapsed_data/95-96.csv")
```

```
## Parsed with column specification:
## cols(
##   USSCIDN = col_integer(),
##   MONRACE = col_integer(),
##   YEAR = col_integer(),
##   MONSEX = col_integer(),
##   CITIZEN = col_integer(),
##   EDUCATN = col_integer(),
##   AGE = col_integer(),
##   STATMIN = col_integer(),
##   STATMAX = col_integer(),
##   DISPOSIT = col_integer(),
##   TOTPRISN = col_integer(),
##   XFOLSOR = col_integer(),
##   XCRHISSR = col_integer(),
##   DISTRICT = col_integer(),
##   MONCIRC = col_integer()
## )
```

```
data <- as_tibble(data)
# Spliting data into to 2 parts
tibble_data <- as_tibble(data)
data_1 <- tibble_data[, 1:5]
data_2 <- tibble_data[, 6:10]
names <- data$USSCIDN
data_1 <- cbind(names, data_1)
data_2 <- cbind(names, data_2)

# left_join
head(left_join(data_1, data_2, by = "names"))
```

```
##    names USSCIDN MONRACE YEAR MONSEX CITIZEN EDUCATN AGE STATMIN STATMAX
## 1 246845  246845       2 1996      1       1       1  20     120     540
## 2 248876  248876       2 1996      0       1       1  54       0     252
## 3 248986  248986       2 1996      0       1       2  21       0     240
## 4 252632  252632       1 1996      0       1       1  25      60     480
## 5 252645  252645       1 1996      0       3       1  35       0      36
## 6 252646  252646       1 1996      1       1       1  21       0      60
##   DISPOSIT
## 1        0
## 2        0
## 3        0
## 4        0
## 5        0
## 6        0
```

```
# right_join
head(right_join(data_1, data_2, by = "names"))
```

```
##    names USSCIDN MONRACE YEAR MONSEX CITIZEN EDUCATN AGE STATMIN STATMAX
## 1 246845  246845       2 1996      1       1       1  20     120     540
## 2 248876  248876       2 1996      0       1       1  54       0     252
## 3 248986  248986       2 1996      0       1       2  21       0     240
## 4 252632  252632       1 1996      0       1       1  25      60     480
## 5 252645  252645       1 1996      0       3       1  35       0      36
## 6 252646  252646       1 1996      1       1       1  21       0      60
```

```
##   DISPOSIT
## 1        0
## 2        0
## 3        0
## 4        0
## 5        0
## 6        0
```

```
# inner_join (simulated by taking out case number 246845)
data_3 <- data_1 %>%
  filter(names != "246845")

head(inner_join(data_3, data_2))
```

```
## Joining, by = "names"
##    names USSCIDN MONRACE YEAR MONSEX CITIZEN EDUCATN AGE STATMIN STATMAX
## 1 248876  248876       2 1996      0       1       1  54       0     252
## 2 248986  248986       2 1996      0       1       2  21       0     240
## 3 252632  252632       1 1996      0       1       1  25      60     480
## 4 252645  252645       1 1996      0       3       1  35       0      36
## 5 252646  252646       1 1996      1       1       1  21       0      60
## 6 252647  252647       1 1996      1       3       1  47       0     240
##   DISPOSIT
## 1        0
## 2        0
## 3        0
## 4        0
## 5        0
## 6        0
```

```
# full_join - we see that the missing variables for 246845 is last.
head(full_join(data_3, data_2))
```

```
## Joining, by = "names"
##    names USSCIDN MONRACE YEAR MONSEX CITIZEN EDUCATN AGE STATMIN STATMAX
## 1 248876  248876       2 1996      0       1       1  54       0     252
## 2 248986  248986       2 1996      0       1       2  21       0     240
## 3 252632  252632       1 1996      0       1       1  25      60     480
## 4 252645  252645       1 1996      0       3       1  35       0      36
## 5 252646  252646       1 1996      1       1       1  21       0      60
## 6 252647  252647       1 1996      1       3       1  47       0     240
##   DISPOSIT
## 1        0
## 2        0
## 3        0
## 4        0
## 5        0
## 6        0
```

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.

```
# we see that semi_join excludes 246845
head(semi_join(data_3, data_2, "names"))
```

```
##    names USSCIDN MONRACE YEAR MONSEX CITIZEN
```

```
## 1 248876  248876       2 1996      0       1
## 2 248986  248986       2 1996      0       1
## 3 252632  252632       1 1996      0       1
## 4 252645  252645       1 1996      0       3
## 5 252646  252646       1 1996      1       1
## 6 252647  252647       1 1996      1       3
```

```
# we see that 246845 is the exception
head(anti_join(data_2, data_3, "names"))
```

```
##    names EDUCATN AGE STATMIN STATMAX DISPOSIT
## 1 246845       1  20     120     540        0
```

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.

```
data_frankenstein <- bind_cols(data_1, data_2)
```

Binding rows will be useful for my project, but since I'm working with data, I will be binding them by columns.

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

Not applicable.

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.

I binded the data so we can match everything by year

```
# importing clean data
setwd("/Users/hsujohnathan/monitoring-federal-criminal-sentences")
data_95_96 <- read.csv("clean_data/collapsed_data/95-96.csv")
data_96_97 <- read.csv("clean_data/collapsed_data/96-97.csv")
data_97_98 <- read.csv("clean_data/collapsed_data/97-98.csv")
data_1999 <- read.csv("clean_data/collapsed_data/1999.csv")
data_2000 <- read.csv("clean_data/collapsed_data/2000.csv")
data_2001 <- read.csv("clean_data/collapsed_data/2001.csv")
data_2002 <- read.csv("clean_data/collapsed_data/2002.csv")
data_2003 <- read.csv("clean_data/collapsed_data/2003.csv")
data_2004 <- read.csv("clean_data/collapsed_data/2004.csv")
data_2005 <- read.csv("clean_data/collapsed_data/2005.csv")
data_2006 <- read.csv("clean_data/collapsed_data/2006.csv")
data_2007 <- read.csv("clean_data/collapsed_data/2007.csv")
data_2008 <- read.csv("clean_data/collapsed_data/2008.csv")
data_2009 <- read.csv("clean_data/collapsed_data/2009.csv")
data_2010 <- read.csv("clean_data/collapsed_data/2010.csv")
data_2011 <- read.csv("clean_data/collapsed_data/2011.csv")
data_2012 <- read.csv("clean_data/collapsed_data/2012.csv")
data_2013 <- read.csv("clean_data/collapsed_data/2013.csv")
data_2014 <- read.csv("clean_data/collapsed_data/2014.csv")
data_2015 <- read.csv("clean_data/collapsed_data/2015.csv")

data_bind <- bind_rows(data_95_96, data_96_97, data_97_98, data_1999,
                       data_2000, data_2001, data_2002, data_2003,
                       data_2004, data_2005, data_2006, data_2007,
                       data_2008, data_2009, data_2010, data_2011,
```

```
                        data_2012, data_2013, data_2014, data_2015)
```

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.

Not applicable

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.

No.

```
tbl_df(head(data_bind))
```

```
## # A tibble: 6 x 16
##   USSCIDN MONRACE  YEAR MONSEX CITIZEN EDUCATN   AGE STATMIN STATMAX
## *   <dbl>   <int> <dbl>  <int>   <int>   <int> <int>   <int>   <dbl>
## 1  246845       2  1996      1       1       1    20     120     540
## 2  248876       2  1996      0       1       1    54       0     252
## 3  248986       2  1996      0       1       2    21       0     240
## 4  252632       1  1996      0       1       1    25      60     480
## 5  252645       1  1996      0       3       1    35       0      36
## 6  252646       1  1996      1       1       1    21       0      60
## # ... with 7 more variables: DISPOSIT <int>, TOTPRISN <int>,
## #   XFOLSOR <int>, XCRHISSR <int>, DISTRICT <int>, MONCIRC <int>,
## #   TYPE <int>
```