Lab 10 - Merging Data

William Bernard

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.

N/A

One table, Americans' changing lives represents the changing lives of americans.

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?

the variables in my data are labeled uniformyly, V then number. Keys become less important since I will only be using one very large data set.

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.

```
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(tidyr)
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: readr
## Loading tidyverse: purrr
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## filter(): dplyr, stats
## lag():    dplyr, stats
```

```
setwd("C:\\Users\\William Bernard\\Desktop\\William Bernard Poster Project v.2\\william_bernard_poster_

# Read in your data with the appropriate function
changing_lives <- load("C:\\Users\\William Bernard\\Desktop\\americans's_changing_lives_data_set\\ICPSR_
```

```
data_subset <- da04690.0001 %>%
  select(V6, V7, V103, V104, V220, V221, V222, V223, V224, V225, V301, V302, V303, V304, V305, V306, V3

group_by(data_subset, V301)
```

```
## # A tibble: 3,617 x 59
## # Groups:   V301 [6]
##              V6           V7       V103  V104        V220        V221
##          <fctr>       <fctr>     <fctr> <dbl>      <fctr>      <fctr>
##  1  (1) CORRECT  (1) CORRECT (2) FEMALE    69   (6) NEVER   (2) 1X/WK
##  2  (1) CORRECT  (1) CORRECT   (1) MALE    44  (2) 1X/DAY   (2) 1X/WK
##  3  (1) CORRECT  (1) CORRECT   (1) MALE    75  (5) <1X/WK  (1) >1X/WK
##  4  (1) CORRECT  (1) CORRECT   (1) MALE    25 (3) 2-3X/WK   (2) 1X/WK
##  5 (5) INCORRCT  (1) CORRECT (2) FEMALE    30 (3) 2-3X/WK  (1) >1X/WK
##  6  (1) CORRECT  (1) CORRECT   (1) MALE    57 (3) 2-3X/WK   (2) 1X/WK
##  7 (5) INCORRCT (5) INCORRCT (2) FEMALE    56   (6) NEVER   (2) 1X/WK
##  8 (5) INCORRCT  (1) CORRECT (2) FEMALE    37 (3) 2-3X/WK (3) 2-3X/MO
##  9  (1) CORRECT  (1) CORRECT (2) FEMALE    27 (1) >1X/DAY  (1) >1X/WK
## 10 (5) INCORRCT  (1) CORRECT   (1) MALE    73 (3) 2-3X/WK   (4) 1X/MO
## # ... with 3,607 more rows, and 53 more variables: V222 <fctr>,
## #   V223 <fctr>, V224 <fctr>, V225 <fctr>, V301 <fctr>, V302 <fctr>,
## #   V303 <fctr>, V304 <fctr>, V305 <fctr>, V306 <fctr>, V307 <fctr>,
## #   V308 <fctr>, V309 <fctr>, V310 <fctr>, V311 <fctr>, V314 <fctr>,
## #   V315 <fctr>, V316 <fctr>, V317 <fctr>, V322 <fctr>, V323 <fctr>,
## #   V325 <fctr>, V326 <fctr>, V328 <fctr>, V329 <fctr>, V330 <fctr>,
## #   V331 <fctr>, V332 <fctr>, V333 <fctr>, V334 <fctr>, V335 <fctr>,
## #   V401 <fctr>, V402 <fctr>, V405 <fctr>, V406 <fctr>, V407 <fctr>,
## #   V408 <fctr>, V410 <dbl>, V416 <fctr>, V419 <fctr>, V420 <dbl>,
## #   V425 <dbl>, V430 <fctr>, V431 <fctr>, V432 <fctr>, V433 <fctr>,
## #   V434 <fctr>, V437 <fctr>, V438 <fctr>, V440 <fctr>, V441 <dbl>,
## #   V445 <fctr>, V446 <fctr>
```

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 the same.

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?

```
data_table_1 <- data_subset %>%
  select(V301, V305, V311, V432, V408)

data_table_2 <- data_subset %>%
  select(V301, V420, V335, V309, V437)

as_tibble(left_join(data_table_1, data_table_2, by = "V301"))
```

```
## # A tibble: 3,877,815 x 9
##           V301   V305   V311   V432   V408  V420         V335   V309
##         <fctr> <fctr> <fctr> <fctr> <fctr> <dbl>       <fctr> <fctr>
##  1 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>     2  (2) AG SOME   <NA>
##  2 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA  (2) AG SOME   <NA>
```

```
##  3 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>     0  (4) STR DIS   <NA>
##  4 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>     0  (4) STR DIS   <NA>
##  5 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA (3) DIS SOME   <NA>
##  6 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA  (2) AG SOME   <NA>
##  7 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA  (4) STR DIS   <NA>
##  8 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA  (4) STR DIS   <NA>
##  9 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>     1 (3) DIS SOME   <NA>
## 10 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA  (4) STR DIS   <NA>
## # ... with 3,877,805 more rows, and 1 more variables: V437 <fctr>
```

```
as_tibble(right_join(data_table_1, data_table_2, by = "V301"))
```

```
## # A tibble: 3,877,815 x 9
##           V301   V305   V311         V432         V408  V420        V335
##         <fctr> <fctr> <fctr>       <fctr>       <fctr> <dbl>      <fctr>
##  1 (1) COMPSAT   <NA>   <NA>         <NA>         <NA>     2 (2) AG SOME
##  2 (1) COMPSAT   <NA>   <NA>  (4) ALITTLE         <NA>     2 (2) AG SOME
##  3 (1) COMPSAT   <NA>   <NA> (5) NOTATALL (5) NOTATALL     2 (2) AG SOME
##  4 (1) COMPSAT   <NA>   <NA>  (4) ALITTLE  (4) ALITTLE     2 (2) AG SOME
##  5 (1) COMPSAT   <NA>   <NA> (5) NOTATALL         <NA>     2 (2) AG SOME
##  6 (1) COMPSAT   <NA>   <NA>         <NA>         <NA>     2 (2) AG SOME
##  7 (1) COMPSAT   <NA>   <NA> (5) NOTATALL         <NA>     2 (2) AG SOME
##  8 (1) COMPSAT   <NA>   <NA>         <NA>         <NA>     2 (2) AG SOME
##  9 (1) COMPSAT   <NA>   <NA> (5) NOTATALL (5) NOTATALL     2 (2) AG SOME
## 10 (1) COMPSAT   <NA>   <NA> (5) NOTATALL     (3) SOME     2 (2) AG SOME
## # ... with 3,877,805 more rows, and 2 more variables: V309 <fctr>,
## #   V437 <fctr>
```

```
as_tibble(inner_join(data_table_1, data_table_2, by = "V301"))
```

```
## # A tibble: 3,877,815 x 9
##           V301   V305   V311   V432   V408  V420         V335   V309
##         <fctr> <fctr> <fctr> <fctr> <fctr> <dbl>       <fctr> <fctr>
##  1 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>     2  (2) AG SOME   <NA>
##  2 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA  (2) AG SOME   <NA>
##  3 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>     0  (4) STR DIS   <NA>
##  4 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>     0  (4) STR DIS   <NA>
##  5 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA (3) DIS SOME   <NA>
##  6 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA  (2) AG SOME   <NA>
##  7 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA  (4) STR DIS   <NA>
##  8 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA  (4) STR DIS   <NA>
##  9 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>     1 (3) DIS SOME   <NA>
## 10 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA  (4) STR DIS   <NA>
## # ... with 3,877,805 more rows, and 1 more variables: V437 <fctr>
```

```
as_tibble(full_join(data_table_1, data_table_2))
```

```
## Joining, by = "V301"
```

```
## # A tibble: 3,877,815 x 9
##           V301   V305   V311   V432   V408  V420         V335   V309
##         <fctr> <fctr> <fctr> <fctr> <fctr> <dbl>       <fctr> <fctr>
##  1 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>     2  (2) AG SOME   <NA>
##  2 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA  (2) AG SOME   <NA>
##  3 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>     0  (4) STR DIS   <NA>
##  4 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>     0  (4) STR DIS   <NA>
##  5 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA (3) DIS SOME   <NA>
```

```
##  6 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA  (2) AG SOME   <NA>
##  7 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA  (4) STR DIS   <NA>
##  8 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA  (4) STR DIS   <NA>
##  9 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>     1 (3) DIS SOME   <NA>
## 10 (1) COMPSAT   <NA>   <NA>   <NA>   <NA>    NA  (4) STR DIS   <NA>
## # ... with 3,877,805 more rows, and 1 more variables: V437 <fctr>
```

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

```
as_tibble(semi_join(data_table_1, data_table_2, by = "V301"))
```

```
## # A tibble: 3,617 x 5
##           V301   V305   V311         V432         V408
##         <fctr> <fctr> <fctr>       <fctr>       <fctr>
##  1 (1) COMPSAT   <NA>   <NA>         <NA>         <NA>
##  2 (2) VERYSAT   <NA>   <NA>     (3) SOME (2) QUITEBIT
##  3 (2) VERYSAT   <NA>   <NA>  (4) ALITTLE         <NA>
##  4 (2) VERYSAT   <NA>   <NA>         <NA>         <NA>
##  5 (2) VERYSAT   <NA>   <NA>         <NA>         <NA>
##  6 (2) VERYSAT   <NA>   <NA> (5) NOTATALL         <NA>
##  7 (3) SOMESAT   <NA>   <NA> (2) QUITEBIT         <NA>
##  8 (1) COMPSAT   <NA>   <NA>  (4) ALITTLE         <NA>
##  9 (3) SOMESAT   <NA>   <NA>         <NA>         <NA>
## 10 (2) VERYSAT   <NA>   <NA> (5) NOTATALL (5) NOTATALL
## # ... with 3,607 more rows
```

I would most likely not use this function, but If I was, it would be to check if there are similar rows in another data frame.

```
as_tibble(anti_join(data_table_1, data_table_2, by = "V301"))
```

```
## # A tibble: 0 x 5
## # ... with 5 variables: V301 <fctr>, V305 <fctr>, V311 <fctr>,
## #   V432 <fctr>, V408 <fctr>
```

I would use this to see how useful a potential data set would be to me.

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.

```
as_tibble(intersect(data_table_1, data_table_1))
```

```
## # A tibble: 424 x 5
##           V301   V305   V311         V432         V408
##         <fctr> <fctr> <fctr>       <fctr>       <fctr>
##  1 (1) COMPSAT   <NA>   <NA>         <NA>         <NA>
##  2 (2) VERYSAT   <NA>   <NA>     (3) SOME (2) QUITEBIT
##  3 (2) VERYSAT   <NA>   <NA>  (4) ALITTLE         <NA>
##  4 (2) VERYSAT   <NA>   <NA>         <NA>         <NA>
##  5 (2) VERYSAT   <NA>   <NA> (5) NOTATALL         <NA>
##  6 (3) SOMESAT   <NA>   <NA> (2) QUITEBIT         <NA>
##  7 (1) COMPSAT   <NA>   <NA>  (4) ALITTLE         <NA>
##  8 (3) SOMESAT   <NA>   <NA>         <NA>         <NA>
##  9 (2) VERYSAT   <NA>   <NA> (5) NOTATALL (5) NOTATALL
## 10 (2) VERYSAT   <NA>   <NA>  (4) ALITTLE (5) NOTATALL
## # ... with 414 more rows
```

```
as_tibble(union(data_table_1, data_table_1))
```

```
## # A tibble: 424 x 5
##            V301
##          <fctr>
##  1 (5) NOTAASAT
##  2 (5) NOTAASAT
##  3  (3) SOMESAT
##  4 (5) NOTAASAT
##  5  (3) SOMESAT
##  6 (5) NOTAASAT
##  7  (1) COMPSAT
##  8  (3) SOMESAT
##  9  (2) VERYSAT
## 10  (1) COMPSAT
## # ... with 414 more rows, and 4 more variables: V305 <fctr>, V311 <fctr>,
## #   V432 <fctr>, V408 <fctr>
```

```
as_tibble(setdiff(data_table_1, data_table_1))
```

```
## # A tibble: 0 x 5
## # ... with 5 variables: V301 <fctr>, V305 <fctr>, V311 <fctr>,
## #   V432 <fctr>, V408 <fctr>
```

none of these functions are particularly useful for this project, I have one data set. If I got my hands on other data sets, union could be useful when looking at the variables of other data sets. Quickly finding the overlap with this function would be very helpful.

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

N/A

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 basically pastes data sets on to one another. you could mess with the total observations of some of the variables if the data doesnt line up right, which would severly hinder your work.

N/A

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.

N/A

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.

N/A