Lab 10 - Merging Data

Snow Christensen

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.

No, I will just be using one.

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?

My primary keys are RACE, NOASSER, PLEASE, AUTHSI and PRESSSI.

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.

```
#load package to read .por files
library("haven")

#load package tidyverse for the rest of the lab
install.packages("tidyverse")
```

```
## Installing package into 'C:/Users/snowbc/Documents/Projects/sexual_violence_college/packrat/lib/x86_
## (as 'lib' is unspecified)

##
##   There is a binary version available (and will be installed) but
##   the source version is later:
##           binary source
## tidyverse  1.1.1  1.2.0

## also installing the dependencies 'openssl', 'cellranger', 'selectr', 'broom', 'httr', 'lubridate', 'm

## package 'openssl' successfully unpacked and MD5 sums checked
## package 'cellranger' successfully unpacked and MD5 sums checked
## package 'selectr' successfully unpacked and MD5 sums checked
## package 'broom' successfully unpacked and MD5 sums checked
## package 'httr' successfully unpacked and MD5 sums checked
## package 'lubridate' successfully unpacked and MD5 sums checked
## package 'modelr' successfully unpacked and MD5 sums checked
## package 'readxl' successfully unpacked and MD5 sums checked
## package 'rvest' successfully unpacked and MD5 sums checked
## package 'xml2' successfully unpacked and MD5 sums checked
## package 'tidyverse' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Temp\29\Rtmp00aFc1\downloaded_packages
```

```
library("tidyverse")
```

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

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

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

```
#read in full dataset
data <- read_por("~/Projects/sexual_violence_college/ICPSR_03212/DS0001/03212-0001-Data.por")

#convert original dataset to a tbl
as_tibble(data)
```

```
## # A tibble: 1,580 x 2,075
##    CODENUM      EDUC       MOB     DAYOB      B_YR      RACE   MARSTAT
##      <chr> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
##  1 90-0002        12         3        NA      1972         1         1
##  2 90-0003        12         6        NA      1972         2         1
##  3 90-0004        12         6        NA      1973         3         1
##  4 90-0005        12        10        NA      1972         1         1
##  5 90-0006        12         6        NA      1972         1         1
##  6 90-0007        12         1        NA      1972         3         1
##  7 90-0011        12         1        NA      1972         1         1
##  8 90-0014        12         4        NA      1972         1         1
##  9 90-0015        12         8        NA      1970         1         1
## 10 90-0016        12         1        NA      1972         1         1
## # ... with 1,570 more rows, and 2068 more variables: RELINFL <dbl+lbl>,
## #   RELATT <dbl+lbl>, RELIG <dbl+lbl>, RELINFL1 <dbl+lbl>,
## #   RELATT1 <dbl+lbl>, GRAD <dbl+lbl>, CHARGE <dbl+lbl>, WHINY <dbl+lbl>,
## #   TOUGH <dbl+lbl>, GREATEST <dbl+lbl>, EMOTE <dbl+lbl>,
## #   GIVEIN <dbl+lbl>, BRAG <dbl+lbl>, GETMAD <dbl+lbl>, BUSY <dbl+lbl>,
## #   CENTER <dbl+lbl>, ALTRUE <dbl+lbl>, WHIMPY <dbl+lbl>, ROUGH <dbl+lbl>,
## #   COMPLAIN <dbl+lbl>, HELPFUL <dbl+lbl>, CONTESTS <dbl+lbl>,
## #   NOASSERT <dbl+lbl>, HOMEY <dbl+lbl>, GREED <dbl+lbl>, MEAN <dbl+lbl>,
## #   APPROV <dbl+lbl>, BOSSY <dbl+lbl>, NOHURT <dbl+lbl>, NAG <dbl+lbl>,
## #   NOEMPATH <dbl+lbl>, INDECIS <dbl+lbl>, FUSSY <dbl+lbl>,
## #   GIVEUP <dbl+lbl>, NOTRUST <dbl+lbl>, NOCRY <dbl+lbl>,
## #   CONFIDEN <dbl+lbl>, NUMONE <dbl+lbl>, BETTER <dbl+lbl>,
## #   REVENGE <dbl+lbl>, EMPATH <dbl+lbl>, FRIENDLY <dbl+lbl>,
## #   PLEASE <dbl+lbl>, NORISK <dbl+lbl>, TRUSTFUL <dbl+lbl>,
## #   FLUSTER <dbl+lbl>, NERVOUS <dbl+lbl>, BADNERV <dbl+lbl>,
## #   TENSE <dbl+lbl>, ANXIOUS <dbl+lbl>, NOCALM <dbl+lbl>, JUMPY <dbl+lbl>,
## #   RESTLESS <dbl+lbl>, RATTLED <dbl+lbl>, SHAKEH <dbl+lbl>,
## #   RELAX <dbl+lbl>, MOODY <dbl+lbl>, LOSPIRIT <dbl+lbl>, BLUE <dbl+lbl>,
## #   DEPRESS <dbl+lbl>, STRAIN <dbl+lbl>, CONTROL <dbl+lbl>,
## #   LOSEMIND <dbl+lbl>, STABLE <dbl+lbl>, NOSUCCES <dbl+lbl>,
## #   CRYING <dbl+lbl>, DEAD <dbl+lbl>, DUMPS <dbl+lbl>, SUICIDE <dbl+lbl>,
## #   NOFORWRD <dbl+lbl>, HAPPY <dbl+lbl>, SATISFID <dbl+lbl>,
```

```
## #   INTEREST <dbl+lbl>, CALM <dbl+lbl>, CHEERFUL <dbl+lbl>,
## #   ENJOY <dbl+lbl>, NOTENSE <dbl+lbl>, ADVENTUR <dbl+lbl>,
## #   XPECTDAY <dbl+lbl>, WAKEUP <dbl+lbl>, FUTRHOPE <dbl+lbl>,
## #   LOVED <dbl+lbl>, LUVRELAT <dbl+lbl>, LONELY <dbl+lbl>,
## #   IDISCUSS <dbl+lbl>, HDISCUSS <dbl+lbl>, IDISCUSD <dbl+lbl>,
## #   HDISCUSD <dbl+lbl>, IGOTINFO <dbl+lbl>, HGOTINFO <dbl+lbl>,
## #   IGOTHELP <dbl+lbl>, HGOTHELP <dbl+lbl>, IARGUED <dbl+lbl>,
## #   HARGUED <dbl+lbl>, IYELLED <dbl+lbl>, HYELLED <dbl+lbl>,
## #   ISULKED <dbl+lbl>, HSULKED <dbl+lbl>, ISTOMPED <dbl+lbl>,
## #   HSTOMPED <dbl+lbl>, ...
```

```r
#read in data_condensed from last lab
data_condensed <- data %>%
  select(RACE, NOASSERT, PLEASE, AUTHSI, PRESSSI)

# create first new data subset
practice_data1 <- head(data_condensed, n=15)

# create second new data subset
practice_data2 <- tail(data_condensed, n=15)

#group practice_data1 by SEXACTS variable and summarise the min and max of that variable
practice_data1 %>%
  group_by(NOASSERT) %>%
    summary(min = min(SEXACTS),
            max = max(SEXACTS))
```

```
##       RACE          NOASSERT         PLEASE          AUTHSI
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1
##  Median :1.000   Median :1.000   Median :2.000   Median :1
##  Mean   :1.467   Mean   :1.733   Mean   :1.733   Mean   :1
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:1
##  Max.   :3.000   Max.   :4.000   Max.   :3.000   Max.   :1
##     PRESSSI
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :1.000
##  Mean   :1.267
##  3rd Qu.:1.500
##  Max.   :2.000
```

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.

I am not merging separate tables for my real data, but for this practice they are of the same data 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?

The full_join() function did because it added all of the rows from the corresponding columns in the secondary data subset to the first. It allowed me to create a more complete final dataset than the other joins did, which is what I wanted.

```r
#perform left join with practice_data1 as pirmary dataset
left_join(practice_data1, practice_data2, by = c("RACE", "NOASSERT", "PLEASE", "AUTHSI", "PRESSSI"))
```

```
## # A tibble: 21 x 5
##         RACE  NOASSERT    PLEASE    AUTHSI   PRESSSI
##    <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
##  1         1         1         2         1         1
##  2         2         2         1         1         2
##  3         3         4         1         1         1
##  4         1         1         2         1         1
##  5         1         1         2         1         1
##  6         3         3         3         1         1
##  7         1         4         2         1         1
##  8         1         1         1         1         2
##  9         1         2         3         1         1
## 10         1         1         2         1         1
## # ... with 11 more rows
```

```r
#perform right join
right_join(practice_data1, practice_data2, by = c("RACE", "NOASSERT", "PLEASE", "AUTHSI", "PRESSSI"))
```

```
## # A tibble: 19 x 5
##         RACE  NOASSERT    PLEASE    AUTHSI   PRESSSI
##    <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
##  1         1         2         2         1         2
##  2         1         2         2         1         2
##  3         1         3         2         1         1
##  4         1         1         1         1         1
##  5         1         1         1         1         1
##  6         1         1         1         1         1
##  7         1         1         1         1         1
##  8         1         3         3         1         1
##  9         1         1         1         1         1
## 10         1         1         1         1         1
## 11         2         2         2         1         2
## 12         1         2         1         1         1
## 13         2         1         1         1         1
## 14         2         1         1         1         1
## 15         1         2         2         1         1
## 16         1         3         1         1         2
## 17         1         2         4         1         1
## 18         1         1         1         1         1
## 19         1         1         1         1         1
```

```r
#perform inner join
inner_join(practice_data1, practice_data2, by = c("RACE", "NOASSERT", "PLEASE", "AUTHSI", "PRESSSI"))
```

```
## # A tibble: 8 x 5
##        RACE  NOASSERT    PLEASE    AUTHSI   PRESSSI
##   <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
## 1         1         1         1         1         1
## 2         1         1         1         1         1
## 3         1         1         1         1         1
## 4         1         1         1         1         1
## 5         1         1         1         1         1
```

```
## 6         1         1         1         1         1
## 7         1         1         1         1         1
## 8         1         1         1         1         1
```

```
#perform full join
full_join(practice_data1, practice_data2, by = c("RACE", "NOASSERT", "PLEASE", "AUTHSI", "PRESSSI"))
```

```
## # A tibble: 32 x 5
##         RACE  NOASSERT    PLEASE    AUTHSI   PRESSSI
##    <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
##  1         1         1         2         1         1
##  2         2         2         1         1         2
##  3         3         4         1         1         1
##  4         1         1         2         1         1
##  5         1         1         2         1         1
##  6         3         3         3         1         1
##  7         1         4         2         1         1
##  8         1         1         1         1         2
##  9         1         2         3         1         1
## 10         1         1         2         1         1
## # ... with 22 more rows
```

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

semi_join() could be used to find similarities between the two separate datasets and anti_join() could be used to find the differences.

```
#perform semi_join()
semi_join(practice_data1, practice_data2, by = c("RACE", "NOASSERT", "PLEASE", "AUTHSI", "PRESSSI"))
```

```
## # A tibble: 2 x 5
##        RACE  NOASSERT    PLEASE    AUTHSI   PRESSSI
##   <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
## 1         1         1         1         1         1
## 2         1         1         1         1         1
```

```
#perform anti_join()
anti_join(practice_data1, practice_data2, by = c("RACE", "NOASSERT", "PLEASE", "AUTHSI", "PRESSSI"))
```

```
## # A tibble: 13 x 5
##         RACE  NOASSERT    PLEASE    AUTHSI   PRESSSI
##    <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
##  1         1         1         2         1         1
##  2         2         2         1         1         2
##  3         3         4         1         1         1
##  4         1         1         2         1         1
##  5         1         1         2         1         1
##  6         3         3         3         1         1
##  7         1         4         2         1         1
##  8         1         1         1         1         2
##  9         1         2         3         1         1
## 10         1         1         2         1         1
## 11         1         1         3         1         1
## 12         2         1         1         1         2
## 13         2         2         1         1         2
```

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.

```
#use union() function to return every row in both datasets
union(practice_data1, practice_data2, by = c("RACE", "NOASSERT", "PLEASE", "AUTHSI", "PRESSSI"))
```

```
## # A tibble: 19 x 5
##         RACE  NOASSERT    PLEASE    AUTHSI   PRESSSI
##    <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
##  1         2         1         1         1         2
##  2         1         1         3         1         1
##  3         1         3         3         1         1
##  4         1         1         1         1         2
##  5         3         4         1         1         1
##  6         1         2         2         1         2
##  7         1         2         1         1         1
##  8         1         1         2         1         1
##  9         1         4         2         1         1
## 10         2         1         1         1         1
## 11         2         2         1         1         2
## 12         2         2         2         1         2
## 13         3         3         3         1         1
## 14         1         3         1         1         2
## 15         1         1         1         1         1
## 16         1         3         2         1         1
## 17         1         2         2         1         1
## 18         1         2         3         1         1
## 19         1         2         4         1         1
```

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

```
setequal(practice_data1, practice_data2, by = c("RACE", "NOASSERT", "PLEASE", "AUTHSI", "PRESSSI"))
```

```
## FALSE: Rows in x but not y: 14, 11, 8, 3, 1, 7, 2, 6, 9. Rows in y but not x: 6, 1, 9, 10, 8, 13, 3,
```

The results tell me if the tables are equal and if they are not, how they compare. For mine they are not equal and it lists all of the rows in x but not y and vice versa.

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 useful to add one dataset to the bottom of another if the variables are exactly the same. You have to be careful though because r doesn't double check that the variables are the same and if they are off by a little it could mess up your entire dataset.

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.

I don't need to merge multiple tables.

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 don't think so, my data is already very tidy.