Lab 10 - Merging Data

George Rhodes

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 don't think. All my data is in the same table. However, I will be comparing different variables from it.

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 Primary keys are labeled as Identity, numbers assinged to ndividual observations.

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.

```r
#install.packages("dplyr")
library("dplyr")
```

```
## Warning: package 'dplyr' was built under R version 3.4.2

##
## 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
```

```r
#install.packages("readr")
library("readr")

#load National Survey on Drug Abuse and Mental Health and create object nsduh2015.
load("/Users/george/Documents/School/UW/SOC321/honors_thesis/honors_thesis/NSDUH-2015-survey-data.rda")
names(PUF2015_102016) <- tolower(names(PUF2015_102016))

#subset to relevant variables.
subset_nsduh2015 <- PUF2015_102016 %>%
  select(questid2, filedate, txevrrcvd, alclottm, catag6, irsex, newrace2, irmaritstat, eduhighcat,
         al30est, alcbng30d, irpinc3, irfamin3, poverty3, coutyp2)
#rename columns
colnames(subset_nsduh2015) <- c("ident", "date", "treatment", "alcohol", "age", "sex", "race", "marital
                                "drink30", "binge30", "income", "famincome", "poverty", "countytype")

#recovered_respondants set to those that have recieved treatment AND not drank in last 12 months.
recovered_respondants <- subset_nsduh2015 %>%
  filter(treatment == 1 & alcohol == 93)
  # 925 observations.


# by_famincome <-group_by(recovered_respondants, famincome, add = TRUE)
#Struggling with this code.

#creating new groups of variables.
alc_variables <- recovered_respondants %>%
  select(1,4, 10, 11)
alc_variables[1:10,]
```

```
##       ident alcohol drink30 binge30
## 1  55015443      93      99       0
## 2  19892643      93      93      93
## 3  43267743      93      93      93
## 4  69210843      93      93      93
## 5  11374843      93      93      93
## 6  43230943      93      99       2
## 7  49340943      93      99       0
## 8  57111943      93      93      93
## 9  88912943      93      93      93
## 10 29323043      93      93      93
```

```
income_variables <- recovered_respondants %>%
  select(1,12:13)

demog_variables <- recovered_respondants %>%
  select(1,5:8, 15)
```

I will use other groups I have created and rejoin them.

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.

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?

```
joined_left <- left_join(alc_variables, income_variables, by = "ident")
head(joined_left, n = 10)
```

```
##       ident alcohol drink30 binge30 income famincome
## 1  55015443      93      99       0      6         6
## 2  19892643      93      93      93      1         2
## 3  43267743      93      93      93      2         3
## 4  69210843      93      93      93      5         7
## 5  11374843      93      93      93      2         6
## 6  43230943      93      99       2      4         5
## 7  49340943      93      99       0      2         2
## 8  57111943      93      93      93      1         2
## 9  88912943      93      93      93      1         2
## 10 29323043      93      93      93      3         3
```

```r
joined_right <- right_join(alc_variables, income_variables, by = "ident")
head(joined_right, n = 10)
```

```
##       ident alcohol drink30 binge30 income famincome
## 1  55015443      93      99       0      6         6
## 2  19892643      93      93      93      1         2
## 3  43267743      93      93      93      2         3
## 4  69210843      93      93      93      5         7
## 5  11374843      93      93      93      2         6
## 6  43230943      93      99       2      4         5
## 7  49340943      93      99       0      2         2
## 8  57111943      93      93      93      1         2
## 9  88912943      93      93      93      1         2
## 10 29323043      93      93      93      3         3
```

```r
joined_inner <- inner_join(alc_variables, income_variables, by = "ident")
head(joined_inner, n = 10)
```

```
##       ident alcohol drink30 binge30 income famincome
## 1  55015443      93      99       0      6         6
## 2  19892643      93      93      93      1         2
## 3  43267743      93      93      93      2         3
## 4  69210843      93      93      93      5         7
## 5  11374843      93      93      93      2         6
## 6  43230943      93      99       2      4         5
## 7  49340943      93      99       0      2         2
## 8  57111943      93      93      93      1         2
## 9  88912943      93      93      93      1         2
## 10 29323043      93      93      93      3         3
```

```r
joined_full <- full_join(alc_variables, income_variables, by = "ident")
head(joined_full, n = 10)
```

```
##       ident alcohol drink30 binge30 income famincome
## 1  55015443      93      99       0      6         6
## 2  19892643      93      93      93      1         2
## 3  43267743      93      93      93      2         3
## 4  69210843      93      93      93      5         7
## 5  11374843      93      93      93      2         6
## 6  43230943      93      99       2      4         5
## 7  49340943      93      99       0      2         2
## 8  57111943      93      93      93      1         2
## 9  88912943      93      93      93      1         2
## 10 29323043      93      93      93      3         3
```

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

```r
joined_semi <- semi_join(alc_variables, income_variables)
```

```
## Joining, by = "ident"
```

```r
head(joined_semi, n = 10)
```

```
##       ident alcohol drink30 binge30
## 1  55015443      93      99       0
## 2  19892643      93      93      93
## 3  43267743      93      93      93
## 4  69210843      93      93      93
## 5  11374843      93      93      93
## 6  43230943      93      99       2
## 7  49340943      93      99       0
## 8  57111943      93      93      93
## 9  88912943      93      93      93
## 10 29323043      93      93      93
```

```
joined_anti <- anti_join(alc_variables, income_variables)
```

```
## Joining, by = "ident"
```

```
head(joined_anti, n = 10)
```

```
## [1] ident   alcohol drink30 binge30
## <0 rows> (or 0-length row.names)
```

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.

In these cases, they all have the same rows and number of rows, it is not useful. If I was combining this survey data with follow up interviews of some of the respondants, then the set joins would be useful.

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

I have no reason at this point.

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 just pastes together tables if they have the same dimensions. This is dangerous becasue if they are not in the same order, they will paste together anyway and no longer provide accurate data. I could use this if I want to slice certain variables away from recovered_respondants and then paste the tables back to gether.

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.

Again. I don't think so. However, the reduce() didn't make sense to me, so I'd have to relearn it.

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.