Analysis 2: Joining Cholesterol with Crimes and Web Scraping Wikipedia

Sunghwu Song

March 18, 2021

Instructions

Overview: For each question, show your R code that you used to answer each question in the provided chunks. When a written response is required, be sure to answer the entire question in complete sentences outside the code chunks. When figures are required, be sure to follow all requirements to receive full credit. Point values are assigned for every part of this analysis.

Helpful: Make sure you knit the document as you go through the assignment. Check all your results in the created PDF file.

Submission: Submit via an electronic document on Gradescope. Must be submitted as an PDF file generated in RStudio.

Introduction

Does high cholesterol lead to high crime rates? Probably not, but web scraping will definitely lead to lower crime rates. This data analysis assignment is separated into three parts which cover material from the lectures on tidy data, joins, and webscraping. In Part 1, you will demonstrate the basic concept of joins by connecting relational data involving a cholesterol study. For this segment, `pivot_longer` and `pivot_wider` will be utilized to create a single tidy dataset ready for analysis. In Part 2, we will join all 5 datasets from the lecture series on web scraping. Part 3 will require an understanding of web scraping to import a table found on Wikipedia directly into R. The following R code reads in all datasets required for this assignment.

```
# Data for Part 1
CHOL1=read_csv("Cholesterol.csv")
CHOL2=read_csv("Cholesterol2.csv")

# Data for Part 2
VIOLENT=read_csv("FINAL_VIOLENT.csv")
ZIP=read_csv("FINAL_ZIP.csv")
STATE_ABBREV=read_csv("FINAL_STATE_ABBREV.csv")
CENSUS=read_csv("FINAL_CENSUS.csv")
S_VS_D=read_csv("FINAL_SAFE_VS_DANGEROUS.CSV")
```

Assignment

Part 1: Cholesterol Experiment

The data frame `CHOL1` contains experimental results from randomly assigning *18* people to one of two competing margarine brands "A" and "B". The cholesterol of these patients was measured once before using the margarine brand, once after *4* weeks with the margarine brand, and then again after *8* weeks with the margarine brand. Researchers want to see if there is benefit of these brands of margarine on reducing an individual's cholesterol and want to determine if there is a statistically significant difference between the two competing brands.

Q1 *(3 Points)*

Start by examing the tables `CHOL1` and `CHOL2` and answering the following questions with *Yes* or *No* responses.

Is the variable `ID` in `CHOL1` a primary key?

Answer *(1 Point)*: Yes

Is the variable, `Margarine` in `CHOL1` a primary key?

Answer *(1 Point)*: No

Is the variable, `Brand` in `CHOL2` a primary key?

Answer *(1 Point)*: No

Q2 *(2 Points)*

In a new data frame called `CHOL1a` based on `CHOL1`, rename the variables `After4weeks` and `After8weeks` to nonsynctactic variable names `4` and `8`, respectively. Use `names(CHOL1a)` to display this modification.

```
CHOL1a = rename(CHOL1, '4' = After4weeks, '8' = After8weeks)
names(CHOL1a)
```

```
## [1] "ID"        "Before"    "4"         "8"         "Margarine"
```

Q3 *(4 Points)*

Use the `pivot_longer()` function or `gather()` function on `CHOL1a` to create a new numeric variable called `Week` that contains numeric values *4* or *8* and a new numeric variable called `Response` that contains the Cholesterol after the corresponding number of weeks. Create a new data frame called `CHOL1b` with these modifications and use `str(Chol1b)` to show that both variables have been created correctly and are indeed numeric (an integer variable is a specific type of numeric variable).

```
CHOL1b <- CHOL1a %>%
  gather("4","8", key = "Response", value = "Chloesterol", convert = TRUE)
str(CHOL1b)
```

```
## tibble [36 x 5] (S3: tbl_df/tbl/data.frame)
##  $ ID         : num [1:36] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Before     : num [1:36] 6.42 6.76 6.56 4.8 8.43 7.49 8.05 5.05 5.77 3.91 ...
##  $ Margarine  : chr [1:36] "B" "A" "B" "A" ...
##  $ Response   : int [1:36] 4 4 4 4 4 4 4 4 4 4 ...
##  $ Chloesterol: num [1:36] 5.83 6.2 5.83 4.27 7.71 7.12 7.25 4.63 5.31 3.7 ...
```

Q4 *(4 Points)*

Now working with `CHOL2`, we want to spread the variable `Statistic` across multiple columns. Do this in a new data frame called `CHOL2a` and use `print(CHOL2a)` to display the modified complete table.

```
CHOL2a <- CHOL2 %>%
  spread(key = Statistic, value = Value)

print(CHOL2a)
```

```
## # A tibble: 2 x 6
##   Brand Calories   Fat SatFat Serving Sodium
##   <chr>    <dbl> <dbl>  <dbl>   <dbl>  <dbl>
## 1 A           70     7    2.5      14    130
## 2 B           50     6    1.5      14     NA
```

Q5 *(3 Points)*

Start by examing the tables `CHOL1b` and `CHOL2a` and answering the following questions with *Yes* or *No* responses.

Is the variable `ID` in `CHOL1b` a primary key?

Answer *(1 Point)*: No

Is the variable, `Margarine` in `CHOL1b` a primary key?

Answer *(1 Point)*: No

Is the variable, `Brand` in `CHOL2a` a primary key?

Answer *(1 Point)*: Yes

Q6 *(4 Points)*

Get the nutritional facts of the different margarine brands in `CHOL2a` into the experimental results found in `CHOL1b` using a join. Create a new data frame named `CHOL.COMBINED` and display the table using `head(CHOL.COMBINED)`. This final data frame should contain *36* observations and *10* variables.

```
CHOL.COMBINED <- CHOL1b %>%
  inner_join(CHOL2a, by = c("Margarine" = "Brand"))

head(CHOL.COMBINED)
```

```
## # A tibble: 6 x 10
##      ID Before Margarine Response Chloesterol Calories   Fat SatFat Serving
##   <dbl>  <dbl> <chr>        <int>       <dbl>    <dbl> <dbl>  <dbl>   <dbl>
## 1     1   6.42 B                4        5.83       50     6    1.5      14
## 2     2   6.76 A                4        6.2        70     7    2.5      14
## 3     3   6.56 B                4        5.83       50     6    1.5      14
## 4     4   4.8  A                4        4.27       70     7    2.5      14
## 5     5   8.43 B                4        7.71       50     6    1.5      14
## 6     6   7.49 A                4        7.12       70     7    2.5      14
## # ... with 1 more variable: Sodium <dbl>
```

Part 2: Linking Important Information to 2017 Violent Crimes Data

In the zipped folder, there are *5* CSV files. In this section, we are going to merge all of that data into one object called `FINAL.VIOLENT`.

Q1 *(2 Points)*

The dataset `S_VS_D` contains a variable `CLASS` where "S=Safe" and "D=Dangerous" according to the article *These Are the 2018 Safest and Most Dangerous States in the U.S* by Steve Karantzoulidis. We seek to compare the violent crime statistics for states not in this list. Use a filtering join to create a new data frame called `VIOLENT2` that only contains violent crime statistics from the states not represented in the data frame `S_VS_D`. Use `str(VIOLENT2)` to display the variables and the dimensions of `VIOLENT2`.

```
VIOLENT2 = anti_join(VIOLENT, S_VS_D, by = c("State" = "STATE"))

str(VIOLENT2)
```

```
## tibble [68 x 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ State     : chr [1:68] "Arizona" "Arizona" "Arizona" "Arizona" ...
##  $ City      : chr [1:68] "Chandler" "Gilbert" "Glendale" "Mesa" ...
##  $ Population: num [1:68] 249355 242090 249273 492268 1644177 ...
##  $ Total     : num [1:68] 259.5 85.5 488.2 415.8 760.9 ...
##  $ Murder    : num [1:68] 2.01 2.07 4.81 4.67 9.55 ...
##  $ Rape      : num [1:68] 52.1 16.1 38.9 51.2 69.5 ...
##  $ Robbery   : num [1:68] 57 21.1 193 92.2 200.3 ...
##  $ Assault   : num [1:68] 148.4 46.3 251.5 267.7 481.6 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   State = col_character(),
##   ..   City = col_character(),
##   ..   Population = col_double(),
##   ..   Total = col_double(),
##   ..   Murder = col_double(),
##   ..   Rape = col_double(),
##   ..   Robbery = col_double(),
##   ..   Assault = col_double()
##   .. )
```

Q2 *(4 Points)*

Start by creating a new data set called `VIOLENT3` based on `VIOLENT2` that fixes some problems in the variable `City`. Specifically, we would like to change "Louisville Metro" to "Louisville".

Next, create a new data frame named `VIOLENT4` that connects the population change and density measures from 2019 contained in `CENSUS` to the cities and states in `VIOLENT3`. Use `head(VIOLENT4)` to give a preview of the new merged dataset.

Finally, in a complete sentence, identify any location(s) (Cities and States) missing violent crime information.

Code and Output *(2 Points)*:

```
VIOLENT3 <- VIOLENT2 %>%
  mutate(City=ifelse(City=="Louisville Metro", "Louisville", City))
```

```
VIOLENT4 <- VIOLENT3 %>%
  left_join(CENSUS, by = c("City" = "Name", "State" = "State"))
head(VIOLENT4)
```

```
## # A tibble: 6 x 10
##   State   City      Population Total Murder  Rape Robbery Assault Change Density
##   <chr>   <chr>          <dbl> <dbl>  <dbl> <dbl>   <dbl>   <dbl>  <dbl>   <dbl>
## 1 Arizona Chandler      249355 259.    2.01  52.1    57.0   148.    5.55    1554
## 2 Arizona Gilbert       242090  85.5   2.07  16.1    21.1    46.3   7.21    1443
## 3 Arizona Glendale      249273 488.    4.81  38.9   193.    252.    2.58    1648
## 4 Arizona Mesa          492268 416.    4.67  51.2    92.2   268.    6.87    1450
## 5 Arizona Phoenix      1644177 761.    9.55  69.5   200.    482.    4.49    1254
## 6 Arizona Scottsda~     251840 157.    1.99  40.9    39.7    74.6   4.6      542
```

```
which(is.na(VIOLENT4$Total))
```

```
## [1] 52 53 54 58
```

```
VIOLENT4
```

```
## # A tibble: 68 x 10
##    State    City    Population Total Murder  Rape Robbery Assault Change Density
##    <chr>    <chr>        <dbl> <dbl>  <dbl> <dbl>   <dbl>   <dbl>  <dbl>   <dbl>
##  1 Arizona  Chandl~     249355 259.    2.01  52.1    57.0   148.    5.55    1554
##  2 Arizona  Gilbert     242090  85.5   2.07  16.1    21.1    46.3   7.21    1443
##  3 Arizona  Glenda~     249273 488.    4.81  38.9   193.    252.    2.58    1648
##  4 Arizona  Mesa        492268 416.    4.67  51.2    92.2   268.    6.87    1450
##  5 Arizona  Phoenix    1644177 761.    9.55  69.5   200.    482.    4.49    1254
##  6 Arizona  Scotts~     251840 157.    1.99  40.9    39.7    74.6   4.6      542
##  7 Arizona  Tucson      532323 802.    8.64  93.6   269.    431.    3.25     917
##  8 Califor~ Anaheim     353400 355.    2.83  32.5   136.    183.   -0.2     2706
##  9 Califor~ Bakers~     381154 479.   10.8   24.1   198.    247.    2.13     997
## 10 Califor~ Chula ~     271109 298.    0.74  22.9   112.    162.    2.7     2136
## # ... with 58 more rows
```

Answer *(2 Points)*: The cities missing violent crime information are those that match rows 52, 53, 54, and 58, which are Charlotte, Durham, Greensboro, and Toledo.

Q3 *(6 Points)*

Either ambitiously using one step or less-ambitiously using multiple steps add the longitude and latitude information provided in `ZIP` to the cities and states in `VIOLENT4`. You will need to use `STATE_ABBREV` data to link these two data frames. Your final data frame named `FINAL.VIOLENT` should contain all of the information in `VIOLENT4` along with the variables `lat` and `lon` from `ZIP`. There should be **no** state abbreviations in `FINAL.VIOLENT` since this information is redundant. Use `str(FINAL.VIOLENT)` to demonstrate that everything worked as planned.

In `FINAL.VIOLENT` identify what cities are missing latitude and longitude. Closely, inspect both the `ZIP` and `VIOLENT4` data frames. Report the location(s) missing geographical information and explain in complete sentences why this happened.

Finally, challenge yourself and attempt to fix this problem in a new data frame called `FINAL.VIOLENT.FIX`. Use a combination of `str()` and `filter()` to only display the data in `FINAL.VIOLENT.FIX` for the location(s) that `FINAL.VIOLENT` was missing latitude and longitude. Do this in the second code chunk below.

Code and Output *(4 Points)*:

```
x = left_join(VIOLENT4, STATE_ABBREV, by = "State")
FINAL.VIOLENT = left_join(x, ZIP, by = c("City" = "city", "state" = "state")) %>%
  select(-state)

str(FINAL.VIOLENT)
```

```
## tibble [68 x 12] (S3: tbl_df/tbl/data.frame)
##  $ State     : chr [1:68] "Arizona" "Arizona" "Arizona" "Arizona" ...
##  $ City      : chr [1:68] "Chandler" "Gilbert" "Glendale" "Mesa" ...
##  $ Population: num [1:68] 249355 242090 249273 492268 1644177 ...
##  $ Total     : num [1:68] 259.5 85.5 488.2 415.8 760.9 ...
##  $ Murder    : num [1:68] 2.01 2.07 4.81 4.67 9.55 ...
##  $ Rape      : num [1:68] 52.1 16.1 38.9 51.2 69.5 ...
##  $ Robbery   : num [1:68] 57 21.1 193 92.2 200.3 ...
##  $ Assault   : num [1:68] 148.4 46.3 251.5 267.7 481.6 ...
##  $ Change    : num [1:68] 5.55 7.21 2.58 6.87 4.49 4.6 3.25 -0.2 2.13 2.7 ...
##  $ Density   : num [1:68] 1554 1443 1648 1450 1254 ...
##  $ lat       : num [1:68] 33.3 33.3 33.5 33.4 33.4 ...
##  $ lon       : num [1:68] -112 -112 -112 -112 -112 ...
```

```
FINAL.VIOLENT
```

```
## # A tibble: 68 x 12
##    State City  Population Total Murder  Rape Robbery Assault Change Density
##    <chr> <chr>      <dbl> <dbl>  <dbl> <dbl>   <dbl>   <dbl>  <dbl>   <dbl>
##  1 Ariz~ Chan~     249355 259.    2.01  52.1    57.0   148.    5.55    1554
##  2 Ariz~ Gilb~     242090  85.5   2.07  16.1    21.1    46.3   7.21    1443
##  3 Ariz~ Glen~     249273 488.    4.81  38.9   193.    252.    2.58    1648
##  4 Ariz~ Mesa      492268 416.    4.67  51.2    92.2   268.    6.87    1450
##  5 Ariz~ Phoe~    1644177 761.    9.55  69.5   200.    482.    4.49    1254
##  6 Ariz~ Scot~     251840 157.    1.99  40.9    39.7    74.6   4.6      542
##  7 Ariz~ Tucs~     532323 802.    8.64  93.6   269.    431.    3.25     917
##  8 Cali~ Anah~     353400 355.    2.83  32.5   136.    183.   -0.2     2706
##  9 Cali~ Bake~     381154 479.   10.8   24.1   198.    247.    2.13     997
## 10 Cali~ Chul~     271109 298.    0.74  22.9   112.    162.    2.7     2136
## # ... with 58 more rows, and 2 more variables: lat <dbl>, lon <dbl>
```

Answer *(1 Points)*: Washington D.C. lacks the values for latitude and longitude; I think it is because its state is the District of Columbia but D.C. part in the name created an error when using the ZIP data.

Code and Output *(1 Point)*:

```
y = x %>%
  mutate(City = ifelse(City == "Washington DC", "Washington", City)) %>%
  mutate(state = ifelse(is.na(state), "DC", state))

z = left_join(y, ZIP, by = c("City" = "city", "state"))
```

```
FINAL.VIOLENT.FIX = filter(z, City %in% "Washington")

str(FINAL.VIOLENT.FIX)
```

```
## tibble [1 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ State     : chr "District of Columbia"
##  $ City      : chr "Washington"
##  $ Population: num 693972
##  $ Total     : num 949
##  $ Murder    : num 16.7
##  $ Rape      : num 63.8
##  $ Robbery   : num 339
##  $ Assault   : num 529
##  $ Change    : num 3.65
##  $ Density   : num 4461
##  $ state     : chr "DC"
##  $ lat       : num 38.9
##  $ lon       : num -77
##  - attr(*, "spec")=
##   .. cols(
##   ..   State = col_character(),
##   ..   City = col_character(),
##   ..   Population = col_double(),
##   ..   Total = col_double(),
##   ..   Murder = col_double(),
##   ..   Rape = col_double(),
##   ..   Robbery = col_double(),
##   ..   Assault = col_double()
##   .. )
```

Part 3: Web Scraping a Table From Wikipedia

Wikipedia contains a rough estimate of a billion tables. Search through Wikipedia pages and identify an article, completely unrelated to crimes data, that contains an HTML table that has at least *5* rows and *3* columns. You will be required to web scrape the table into a data frame or tibble into R. This portion will require a minor knowledge of the `rvest` package. Utilize information from the web scraping lectures and tutorials to assist you with this.

Q1 *(4 Points)*

What is the URL of the Wikipedia page you plan on webscraping (Knit the Document and Check the Hyperlink)?

Answer *(2 Points)*: https://en.wikipedia.org/wiki/List_of_Crayola_crayon_colors

In *2* to *5* sentences, Identify and describe the specific table you plan on web scraping. State the variables in *1* of the sentences.

Answer *(2 Points)*: The table I plan to web scrape is the standard list of colors for crayola crayon colors. It shows a table with the following columns: photo showing the color, name of the color, its hexdecimal code, years in production, any notes, and whether they are in the 16-box, 24-box, and/or the 64-box.

Q2 ***(4 Points)***

Utilize the functions `read_html()` and `html_table()` to web scrape the specific table you described above. Internet access will be required for these functions to work. Create an R data frame named `DATA` which contains the information from the Wikipedia table. All code should be contained in the R code chunk below. Finally, use the `print()` function to display the table to demonstrate that everything worked as planned. The variable names and the content should match the table on the Wikipedia page you chose exactly. You are not required to perform any cleaning of this data. As long as the content of the table you describe matches `DATA`, then you are good. Don't worry if the table bleeds over multiple pages.

```
URL.DATA = "https://en.wikipedia.org/wiki/List_of_Crayola_crayon_colors"
DATA = URL.DATA %>%
  read_html() %>%
  html_table(fill=T) %>%
  .[[2]]

print(DATA)
```

```
##     Color                     Name Hexadecimal in their website depiction[b]
## 1      NA                      Red                                  #ED0A3F[1]
## 2      NA                   Maroon                                  #C32148[1]
## 3      NA                  Scarlet                                  #FD0E35[1]
## 4      NA                Brick Red                                  #C62D42[1]
## 5      NA         English Vermilion
## 6      NA              Madder Lake
## 7      NA   Permanent Geranium Lake
## 8      NA              Maximum Red
## 9      NA                  Chestnut                                 #B94E48[1]
## 10     NA               Orange-Red                                  #FF5349[1]
## 11     NA             Sunset Orange                                 #FE4C40[1]
## 12     NA               Bittersweet                                 #FE6F5E[1]
## 13     NA         Dark Venetian Red
## 14     NA              Venetian Red
## 15     NA        Light Venetian Red
## 16     NA           Vivid Tangerine                                 #FF9980[1]
## 17     NA                Middle Red
## 18     NA              Burnt Orange                                 #FF7034[1]
## 19     NA                Red-Orange                                 #FF681F[1]
## 20     NA                    Orange                                 #FF8833[1]
## 21     NA       Macaroni and Cheese                                 #FFB97B[1]
## 22     NA         Middle Yellow Red
## 23     NA              Mango Tango                                  #E77200[1]
## 24     NA            Yellow-Orange                                  #FFAE42[1]
## 25     NA        Maximum Yellow Red
## 26     NA              Banana Mania                                 #FBE7B2[1]
## 27     NA                     Maize
## 28     NA            Orange-Yellow                                  #F8D568[1]
## 29     NA                Goldenrod                                  #FCD667[1]
## 30     NA                Dandelion                                  #FED85D[1]
## 31     NA                    Yellow                                 #FBE870[1]
## 32     NA             Green-Yellow                                  #F1E788[1]
## 33     NA            Middle Yellow
## 34     NA              Olive Green                                  #B5B35C[1]
## 35     NA             Spring Green                                  #ECEBBD[1]
```