Lab 8

Mara Kage

October 27, 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 8 RMD file and the PDF document to Git. Push the changes to GitHub so both documents are visible in your public GitHub repository.

Before you begin: as many of you have large datasets, you're going to want to select only the variables you're interested in utilizing for this project (ideally no more than twenty columns but perhaps much smaller) so you don't have R Studio's memory working on the entire dataset. The example code provided below can be modified to allow you to subset your data to only the variables you wish to use. First, read in your complete dataset and save it as `data`. Then, add the names of the variables you wish to use for your poster project to the `select` function, separated by commas. Run the two lines of code to save this new, smaller version of your data to `data_subset`. Use this smaller dataset to complete the rest of the lab

```
## 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
```

```
## Parsed with column specification:
## cols(
##   POPGROUP.id = col_character(),
##   `POPGROUP.display-label` = col_character(),
##   GEO.id = col_character(),
##   GEO.id2 = col_character(),
##   `GEO.display-label` = col_character(),
##   D001 = col_character()
## )
```

```
## Parsed with column specification:
## cols(
##   GEO.id = col_character(),
##   GEO.id2 = col_character(),
##   `GEO.display-label` = col_character(),
##   HD01_VD01 = col_character(),
##   HD02_VD01 = col_character()
## )
```

```
## [1] 389
```

```
## # A tibble: 1 x 2
##   `sum(TotalCount)`     n
##               <dbl> <int>
## 1           1879189   398
```

```
## As of version 0.5.1, tigris does not cache downloaded data by default. To enable caching of data, se
##
## Attaching package: 'tigris'
## The following object is masked from 'package:graphics':
##
##     plot
```

1. To get a feel for its structure, look at the class, dimensions, column names, structure, and basic summary statistics of your data.
2. Preview the first and last 15 rows of your data. Is you dataset tidy? If not, what principles of tidy data does it seem to be violating?

```
head(data_subset_chinese, 15)
```

```
## # A tibble: 15 x 5
##    `POPGROUP.display-label`               GEO.id     GEO.id2
##                       <chr>                <chr>       <chr>
##  1  Chinese alone (410-419) 1400000US53033000100 53033000100
##  2  Chinese alone (410-419) 1400000US53033000200 53033000200
##  3  Chinese alone (410-419) 1400000US53033000300 53033000300
##  4  Chinese alone (410-419) 1400000US53033000401 53033000401
##  5  Chinese alone (410-419) 1400000US53033000600 53033000600
##  6  Chinese alone (410-419) 1400000US53033000700 53033000700
##  7  Chinese alone (410-419) 1400000US53033000800 53033000800
##  8  Chinese alone (410-419) 1400000US53033001000 53033001000
##  9  Chinese alone (410-419) 1400000US53033001100 53033001100
## 10  Chinese alone (410-419) 1400000US53033001200 53033001200
## 11  Chinese alone (410-419) 1400000US53033001300 53033001300
## 12  Chinese alone (410-419) 1400000US53033001701 53033001701
## 13  Chinese alone (410-419) 1400000US53033001702 53033001702
## 14  Chinese alone (410-419) 1400000US53033001800 53033001800
## 15  Chinese alone (410-419) 1400000US53033001900 53033001900
## # ... with 2 more variables: `GEO.display-label` <chr>, Count <dbl>
```

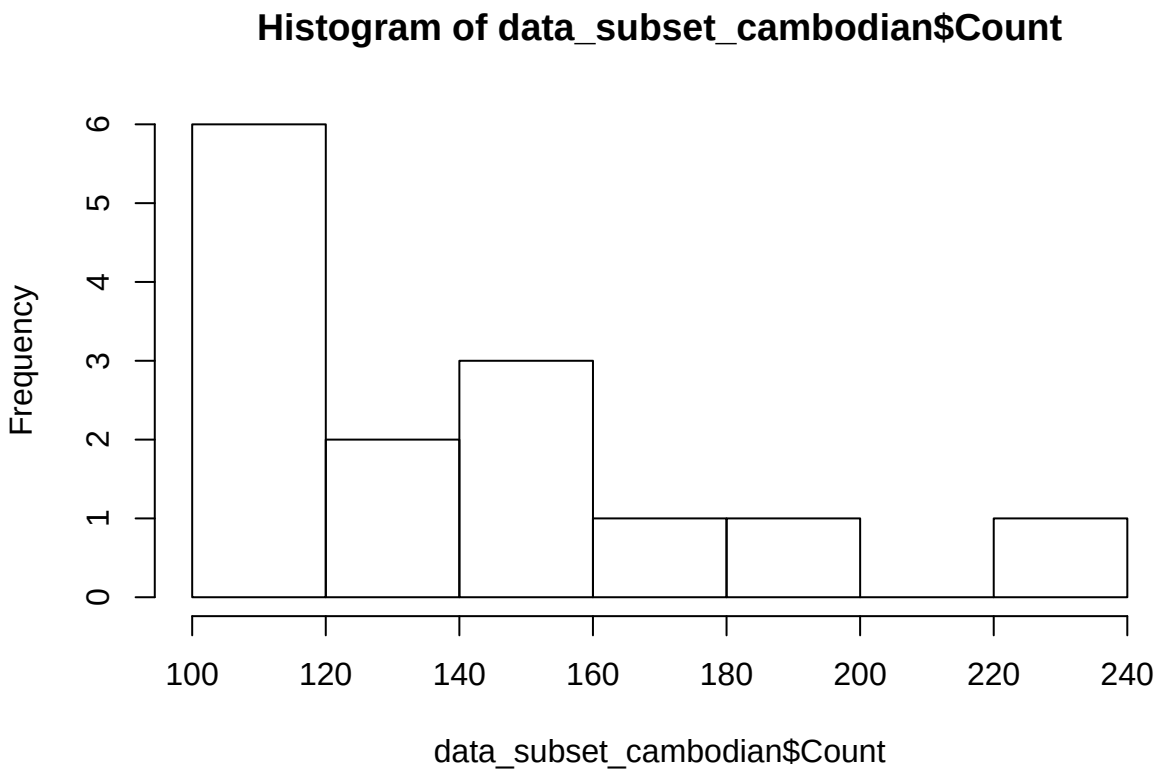
```
tail(data_subset_chinese, 15)
```

```
## # A tibble: 15 x 5
##    `POPGROUP.display-label`               GEO.id     GEO.id2
##                       <chr>                <chr>       <chr>
##  1  Chinese alone (410-419) 1400000US53033032213 53033032213
##  2  Chinese alone (410-419) 1400000US53033032214 53033032214
##  3  Chinese alone (410-419) 1400000US53033032215 53033032215
##  4  Chinese alone (410-419) 1400000US53033032309 53033032309
##  5  Chinese alone (410-419) 1400000US53033032313 53033032313
##  6  Chinese alone (410-419) 1400000US53033032316 53033032316
##  7  Chinese alone (410-419) 1400000US53033032317 53033032317
##  8  Chinese alone (410-419) 1400000US53033032318 53033032318
##  9  Chinese alone (410-419) 1400000US53033032321 53033032321
## 10  Chinese alone (410-419) 1400000US53033032323 53033032323
## 11  Chinese alone (410-419) 1400000US53033032324 53033032324
## 12  Chinese alone (410-419) 1400000US53033032325 53033032325
## 13  Chinese alone (410-419) 1400000US53033032327 53033032327
## 14  Chinese alone (410-419) 1400000US53033032329 53033032329
## 15  Chinese alone (410-419) 1400000US53033032602 53033032602
```

```
## # ... with 2 more variables: `GEO.display-label` <chr>, Count <dbl>
```

3. Create a histogram for at least two variables you plan to focus on for your study. Describe what these plots show you about these variables.

```
AsianAlone <- AsianAlone[-1,]
prop_data <- AsianAlone %>%
  mutate(Count = as.numeric(as.character(D001)),
         POPGROUP.id = as.numeric(as.character(POPGROUP.id))) %>% # Make new count variable which is nu

  group_by(POPGROUP.id) %>% # Group by ethnicity/subgroup
  summarise(Total_pop = sum(Count)) %>% #Sum within subgroup
  filter(POPGROUP.id != 031) %>% #Remove asian and other
  mutate(Total_Asian = sum(Total_pop), #Create total asian pop variable
         proportion = Total_pop/Total_Asian) # Creating proportion of subgroup of total Asians

hist(data_subset_cambodian$Count)
```

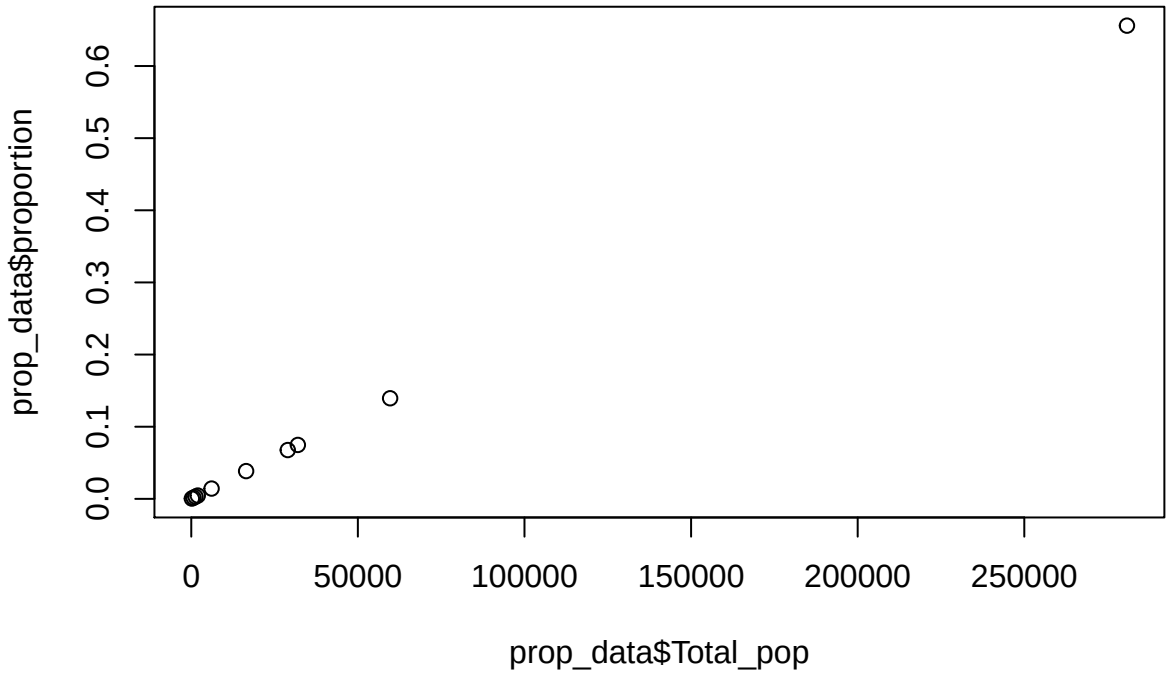



```
#Changing factors to numeric on D001 column/population

pop.chinese <- sum(as.numeric(data_subset_chinese$Count), na.rm=FALSE)
pop.cambodian <- sum(as.numeric(data_subset_cambodian$Count), na.rm=FALSE)
pop.japanese <- sum(as.numeric(data_subset_japanese$Count), na.rm=FALSE)
pop.korean <- sum(as.numeric(data_subset_korean$Count), na.rm=FALSE)
pop.laotian <- sum(as.numeric(data_subset_laotian$Count), na.rm=FALSE)
pop.hmong <- sum(as.numeric(data_subset_hmong$Count), na.rm=FALSE)
pop.taiwanese <- sum(as.numeric(data_subset_taiwanese$Count), na.rm=FALSE)
pop.vietnames <- sum(as.numeric(data_subset_vietnamese$Count), na.rm=FALSE)
pop.filipino <- sum(as.numeric(data_subset_filipino$Count), na.rm=FALSE)
```

```
pop <- c(pop.chinese, pop.cambodian, pop.japanese, pop.korean, pop.laotian,
                 pop.hmong, pop.taiwanese, pop.vietnames, pop.filipino) %>% as.data.frame() %>% t()

colnames(pop) <- c("Chinese", "Cambodian", "Japanese", "Korean", "Laotian",
                   "Hmong", "Taiwanese", "Vietnames", "Filipino")
```

4. Create at least one bivariate plot showing the relationship between two variables of interest. What does/do the(se) plot(s) tell you about the association between these two variables?

```
plot(prop_data$Total_pop, prop_data$proportion)
```




```
# The plot on x-axis is showing the total population by Asian subgroup, and on y-axis is showing the pr
```

5. Load the `tidyr` package. Do all of your columns correspond to variables? Do any columns represent multiple variables? If your answer is yes to either question, carry out the appropriate `tidyr` function (`gather()` or `spread()` respectively) to tidy your data.

```
library(tidyr)
# Dataset prop_data is already tidy with columns corresponding group id(ethnic group) count, population
```

6. Do any columns need to be separated into two or more? Do any columns need to be combined into one? If so, carry out the appropriate the appropriate `tidyr` function (`separate()` or `unite()` respectively) to tidy your data.

At this stage each row in your data should represent one observation, each column should be a variable, and each table should be observational unit.

7. What is the class 'les in your analysis? Are these classes appropriate for the type of measurement they purport to capture? Explain your reasoning.

Variables are appriate as they capture the intended variables of Asian subgroups and proportionality of each subgroup in relationship to the Total asian population.

8. Do any of your variables need to be coerced into a different data type? If so, carry out the appropriate coercion methods below. (This includes transformation of any date objects using the `lubridate` package)

```
prop_data <- AsianAlone %>%
  mutate(Count = as.numeric(as.character(D001)),
         POPGROUP.id = as.numeric(as.character(POPGROUP.id))) # Make new count variable which is numeri

#  Data frame variables of proportin and Total population count in prop-data table have been formated f
```

9. Are there any strings you need to manipulate for your analysis? If so, use the appropriate function from the `stringr` package.

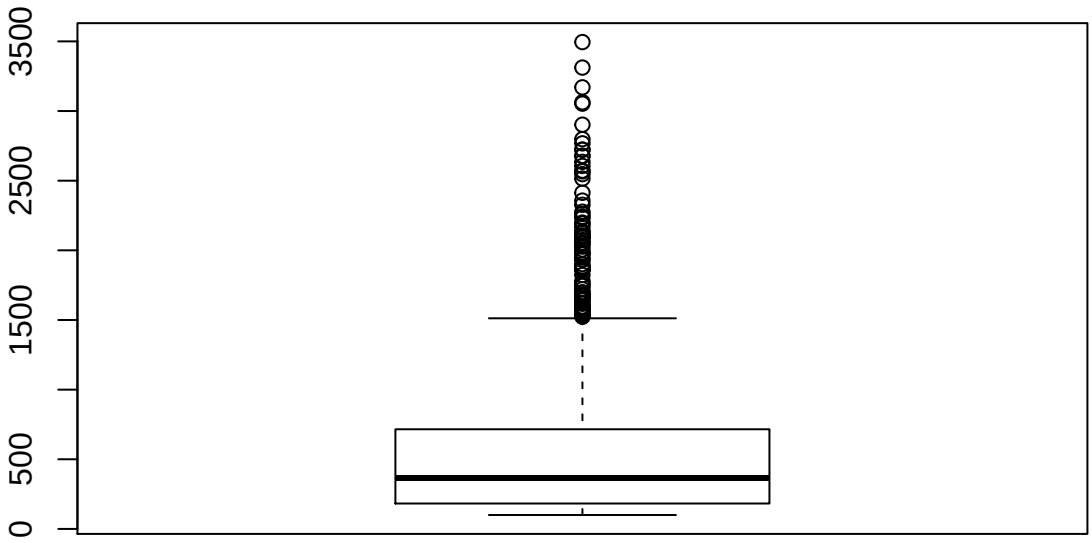
10. Do you have any missing values in your dataset? How many and how are they coded? **Be sure to look out for specific codebook values for missing values (i.e. `-1` for `NA`) as well as empty strings or other software-specific values for `NA`.** Don't worry about removing NAs yet - we'll tackle this question later once discern whether they're random or systematically distributed.

There in no missing values in my data

11. Are there any special values in your dataset? If so, what are they and how do you think they got there? *The presence of special values is less likely if you haven't performed any data manipulation yet so you should remember to return to this step each time you carry out a mathematical transformation of any values in your dataset.*

12. Create a boxplot of your data (you can create an individual boxplot for each variable if there are too many variables in your dataset to meaningfully visualize them all in one plot). Are there any outliers? If so, what are they and to which variable do they correspond? Do any of these outliers seem like obvious errors? If so, why?

```
Asian_plot <- as.numeric(as.character(AsianAlone$D001))
boxplot(Asian_plot, data=001)
```




```
#The boxplot shows a great numbers of outliers. The median number or individuals in each tract is less

save.image("lab8.RData")
```

13. For any outliers and/or obvious errors, what do you think is the best way to handle them (i.e. remove them entirely, run analyses including and excluding them and compare the results, manually change them to an appropriate measure of center, or something else?).

In my analysis, it is actually important to consider the outliers as I am trying to represent minority groups within the Asian American community in King County.