Problem Set 2

YOUR NAME HERE

DATE

- Due date: !
- Submission process: Please submit your assignment directly to Gradescope. You can do this by knitting your file and downloading the PDF to your computer. Then navigate to Gradescope.com or via the link on BCourses to submit your assignment.

Helpful hints:

- Knit your file early and often to minimize knitting errors! If you copy and paste code from the slides, you are bound to get an error that is hard to diagnose. Typing out the code is the way to smooth knitting. We recommend knitting your file each time after you write a few sentences/add a new code chunk, so yu can detect the source of the knitting error more easily. This will save you and the teaching team time!
- Please make sure that your code does not run off the page of the knitted PDF. If it does, we can't see your work. To avoid this, have a look at your knitted PDF and ensure all the code fits in the file. When it doesn't, go back to your .Rmd file and add spaces (new lines) using hte return or enter key so that the code runs onto the next line.

Throughout this problem set please use the reprex function where appropriate.

```
library(reprex)
```

Question

Create a data frame and a tibble that matches the image below:

```
# by the way, you can load images into rmarkdown! Cool, right?!
# here we use the knitr library (though there are multiple ways to load images)
library(knitr)

# notice that we specify the path to look within the current directory
# by using the period: .
# followed by a slash: / to pull the image file
knitr::include_graphics('./table_replicate.png')
```

data_id	gender	temperature
101	female	98
102	male	97.3
103	non-binary	101.1
104	male	97.5
105	NA	99.6

Hint: You may need to load a library for tibbles!

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.1
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
(df <- data.frame(
  id = 101:105,
  gender = c("female", "male", "non-binary", "male", NA),
  temperature = c(98, 97.3, 101.1, 97.5, 99.6))
)
```

```
##    id     gender temperature
## 1 101     female        98.0
## 2 102       male        97.3
## 3 103 non-binary       101.1
## 4 104       male        97.5
## 5 105       <NA>        99.6
```

```r
(tib <- tibble(
  id = 101:105,
  gender = c("female", "male", "non-binary", "male", NA),
  temperature = c(98, 97.3, 101.1, 97.5, 99.6))
)
```

```
## # A tibble: 5 x 3
##      id gender     temperature
##   <int> <chr>            <dbl>
## 1   101 female            98
## 2   102 male              97.3
## 3   103 non-binary       101.
## 4   104 male              97.5
## 5   105 <NA>              99.6
```

What are the key differences between data frames and tibbles?

Why are tibbles preferable?

Question

We just found out results for COVID testing and want to add it to our data. Using the tibble you just created, add the following test results to a new column called "results".

- 101 = NEGATIVE
- 102 = POSITIVE
- 103 = NEGATIVE
- 104 = NEGATIVE
- 105 = NEGATIVE

```
tib$results <- c("NEGATIVE", "POSITIVE", "NEGATIVE", "NEGATIVE", "NEGATIVE")
tib
```

```
## # A tibble: 5 x 4
##      id gender     temperature results
##   <int> <chr>            <dbl> <chr>
## 1   101 female            98   NEGATIVE
## 2   102 male              97.3 POSITIVE
## 3   103 non-binary       101.  NEGATIVE
## 4   104 male              97.5 NEGATIVE
## 5   105 <NA>              99.6 NEGATIVE
```

Question

You find out there was an error in data collection and subject 102's temperature is actually 98.3. Correct the value in your data frame.

```
tib[2, 3] <- 98.3
```

Question

Load the "stds-by-disease-county-year-sex.csv" data set.

You can find more information about this data set from the California Open Data Portal:

https://data.ca.gov/dataset/stds-in-california-by-disease-county-year-and-sex

```
library(readr)
(stds <- read_csv("data/stds-by-disease-county-year-sex.csv",
                  skip = 3)
)
```

```
## Parsed with column specification:
## cols(
##   Disease = col_character(),
##   County = col_character(),
##   Year = col_double(),
##   Sex = col_character(),
##   Cases = col_double(),
##   Population = col_double()
## )
```

```
## # A tibble: 9,558 x 6
##    Disease   County      Year Sex     Cases Population
##    <chr>     <chr>      <dbl> <chr>   <dbl>      <dbl>
##  1 Chlamydia California  2001 Female  75941   17339700
##  2 Chlamydia California  2001 Male    24885   17173042
##  3 Chlamydia California  2001 Total  101590   34512742
##  4 Chlamydia California  2002 Female  81583   17554666
##  5 Chlamydia California  2002 Male    28521   17383624
##  6 Chlamydia California  2002 Total  110759   34938290
##  7 Chlamydia California  2003 Female  85153   17782868
##  8 Chlamydia California  2003 Male    31007   17606060
##  9 Chlamydia California  2003 Total  116385   35388928
## 10 Chlamydia California  2004 Female  89438   17968347
## # ... with 9,548 more rows
```

You may have noticed that there are empty cells in the first three rows. Modify your code above (if you haven't already) to remove these rows.

Question

Let's explore this data set. Insert R chunks as needed. Find the following values:

```
str(stds)
```

```
## tibble [9,558 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Disease   : chr [1:9558] "Chlamydia" "Chlamydia" "Chlamydia" "Chlamydia" ...
##  $ County    : chr [1:9558] "California" "California" "California" "California" ...
##  $ Year      : num [1:9558] 2001 2001 2001 2002 2002 ...
##  $ Sex       : chr [1:9558] "Female" "Male" "Total" "Female" ...
##  $ Cases     : num [1:9558] 75941 24885 101590 81583 28521 ...
##  $ Population: num [1:9558] 17339700 17173042 34512742 17554666 17383624 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Disease = col_character(),
##   ..   County = col_character(),
##   ..   Year = col_double(),
##   ..   Sex = col_character(),
##   ..   Cases = col_double(),
##   ..   Population = col_double()
##   .. )
```

How many rows? 9558

How many columns? 6

What are the column names? Disease, County, Year, Sex, Cases, Population

What are the column types? chr, chr, num, chr, num, num

Question

You want to dig deeper into the data and focus on the years 2015 - 2018. Use the which() function to index which rows fit this year range and assign the results to a new data frame. To check whether this was done correctly you should expect the following dimensions: 2124 rows x 6 columns

```
(stds_2008_2015 <- stds[which(stds$Year > 2014), ])
```

```
## # A tibble: 2,124 x 6
##    Disease   County      Year Sex     Cases Population
##    <chr>     <chr>      <dbl> <chr>   <dbl>      <dbl>
##  1 Chlamydia California  2015 Female 121749   19634752
##  2 Chlamydia California  2015 Male    67694   19441376
##  3 Chlamydia California  2015 Total  189747   39076128
##  4 Chlamydia California  2016 Female 123924   19758238
##  5 Chlamydia California  2016 Male    73708   19570099
##  6 Chlamydia California  2016 Total  198245   39328337
##  7 Chlamydia California  2017 Female 134847   19891334
##  8 Chlamydia California  2017 Male    83203   19719222
##  9 Chlamydia California  2017 Total  218519   39610556
## 10 Chlamydia California  2018 Female 142397   19989903
## # ... with 2,114 more rows
```

Question

Your colleague is interested in this data set as well but hasn't setup their git repository. They ask you to help them out by exporting this new data set as a .csv file. Place your output in the /data folder.

As a test, you can try to read in the .csv you created to make sure everything looks correct.

```
write_csv(stds_2008_2015, "data/stds_2008_2015.csv")
```

Challenge

Look up how to use the unique() function and run it on the County column. You should see a total of 59 counties.

```
unique(stds$County)
```

```
##  [1] "California"      "Alameda"         "Alpine"          "Amador"
##  [5] "Butte"           "Calaveras"       "Colusa"          "Contra Costa"
##  [9] "Del Norte"       "El Dorado"       "Fresno"          "Glenn"
## [13] "Humboldt"        "Imperial"        "Inyo"            "Kern"
## [17] "Kings"           "Lake"            "Lassen"          "Los Angeles"
## [21] "Madera"          "Marin"           "Mariposa"        "Mendocino"
## [25] "Merced"          "Modoc"           "Mono"            "Monterey"
## [29] "Napa"            "Nevada"          "Orange"          "Placer"
## [33] "Plumas"          "Riverside"       "Sacramento"      "San Benito"
## [37] "San Bernardino"  "San Diego"       "San Francisco"   "San Joaquin"
## [41] "San Luis Obispo" "San Mateo"       "Santa Barbara"   "Santa Clara"
## [45] "Santa Cruz"      "Shasta"          "Sierra"          "Siskiyou"
## [49] "Solano"          "Sonoma"          "Stanislaus"      "Sutter"
## [53] "Tehama"          "Trinity"         "Tulare"          "Tuolumne"
## [57] "Ventura"         "Yolo"            "Yuba"
```

You decide to focus on one county. Subset your data for one of your choice.

```
stds_subset <- stds[which(stds$County == "Alameda"), ]
```

You're very interested in finding the rate of cases per 100,000 population. Create a new column called "rate" with the calculated values.

Hint: R allows you to use manipulate variables within a data frame to calculate new values so long as the rows and data types match up. For example: df$var3 <- df$var1 + df$var2

```
stds_subset$Rate <- (stds_subset$Cases / stds_subset$Population) * 100000
```

You're done! Please knit to pdf and upload to gradescope.