PHW251 Problem Set 3

Clara Voong

Submission process: Please submit your assignment directly to Gradescope. You can do this by rendering 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:

- Render your file early and often to minimize rendering 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 rendering. We recommend rendering your file each time after you write a few sentences/add a new code chunk, so you can detect the source of the rendering error more easily. You can render by clicking the blue "Render" arrow in the top menu bar in R studio. This will save you and the teaching team time!
- To render as PDF, you will need to have LaTeX installed. If you are using the datahub **this should already be there and you don't need to install or update**. If you are not using datahub, there is an R package to handle this! If you don't have LaTeX, you can un-comment and run the following code (note: you only need to do this once, and don't need to load this like other packages).

```
# If you don't have LaTeX installed at all,
# uncomment and run the line below in the console
# install.packages("tinytex")
# Be sure to comment it again before rendering, it will cause problems

# If you need to update LaTeX or run into rendering issues,
# uncomment and run the line below in the console
#tinytex::reinstall_tinytex()
# Be sure to comment it again before rendering, it will cause problems
```

- Please make sure that your code does not run off the page of the rendered PDF. If it does, we can't see your work. To avoid this, have a look at your rendered PDF and ensure all the code fits in the file. When it doesn't, go back to your .qmd file and add spaces (new lines) using the return or enter key so that the code runs onto the next line.

Question 1

In this question you will create a data frame. Below is code for how to do this:

```
# data frame with three columns and three rows
# notice how we start with the column name and then the row values
# each column, in this example, has three values
df_example <- data.frame("column_1" = 1:3,
                         "column_2" = c("string_1", "string_2", "2"),
                         # here we add an NA value due to missing data
                         "column_3" = c(NA, "string_3", 50))
df_example
```

```
  column_1 column_2 column_3
1        1 string_1     <NA>
2        2 string_2 string_3
3        3        2       50
```

Now you try! Create a data frame with three columns and the following values:

- Column 1: ID
  - 1, 2, 3
- Column 2: NAME
  - "Pam", "Jim", "Dwight"
- Column 3: AGE
  - 40, NA, 48

```
df1 <- data.frame(
  "ID" = 1:3,
  "NAME" = c("Pam", "Jim", "Dwight"),
  "AGE" = c(40, NA, 48)
)

df1
```

```
  ID   NAME AGE
1  1    Pam  40
2  2    Jim  NA
3  3 Dwight  48
```

Question 2

With your new data frame created in the previous question, find the following values:

- length
- typeof
- class

```
length(df1)
```

```
[1] 3
```

```
typeof(df1)
```

```
[1] "list"
```

```
class(df1)
```

```
[1] "data.frame"
```

Question 3

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 core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
tib1 <- tibble(
  data_id = 101:105,
  gender = c("female", "male", "non-binary", "male", NA),
  temperature = c(98, 97.3, 101.1, 97.5, 99.6)
)
```

```
tib1
```

```
# A tibble: 5 x 3
  data_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
```

Question 4

What are the key differences between data frames and tibbles?

Data frames and tibbles are similar, but tibbles often lead to cleaner codes because it lets you know if there are problems earlier.

Why are tibbles preferable?

Tibbles are easier to use and don't require as much code to perform tasks. i.e. to create a data frame, we must put the column names as strings, but to make a tibble we can directly refer to the column name. Tibble column names can also be those that are typically seen as invalid in base R.

Question 5

We just found out results for COVID testing and want to add it to our data. Using the tibble you created in Question 3, add the following test results to a new column called "results".

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

```
tib1["results"] <- c("NEGATIVE", "POSITIVE", "NEGATIVE", "NEGATIVE", "NEGATIVE")
tib1
```

```
# A tibble: 5 x 4
  data_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 6

You find out there was an error in data collection and subject 102's temperature is actually 98.3, not 97.3. Correct the value in your data frame.

```
# your code here
tib1[2, 3] = 98.3
tib1
```

```
# A tibble: 5 x 4
  data_id gender     temperature results
    <int> <chr>            <dbl> <chr>
1     101 female            98   NEGATIVE
2     102 male              98.3 POSITIVE
3     103 non-binary       101.  NEGATIVE
4     104 male              97.5 NEGATIVE
5     105 <NA>              99.6 NEGATIVE
```

Question 7

Load the "stds-by-disease-county-year-sex.csv" data set, which is in the data folder.

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

```
std_data <- read.csv("~/phw251_fall24/week 4/stds-by-disease-county-year-sex.csv", skip=3)
```

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 8

Let's explore this STD data set. **Use code** to find the values requested below. Insert R chunks as needed.

How many rows?

```
nrow(std_data)
```

```
[1] 9558
```

How many columns?

```
ncol(std_data)
```

```
[1] 6
```

What are the column names?

```
colnames(std_data)
```

```
[1] "Disease"    "County"     "Year"       "Sex"        "Cases"
[6] "Population"
```

What are the column types?

```
str(std_data)
```

```
'data.frame':   9558 obs. of  6 variables:
 $ Disease   : chr  "Chlamydia" "Chlamydia" "Chlamydia" "Chlamydia" ...
 $ County    : chr  "California" "California" "California" "California" ...
 $ Year      : int  2001 2001 2001 2002 2002 2002 2003 2003 2003 2004 ...
 $ Sex       : chr  "Female" "Male" "Total" "Female" ...
 $ Cases     : int  75941 24885 101590 81583 28521 110759 85153 31007 116385 89438 ...
 $ Population: int  17339700 17173042 34512742 17554666 17383624 34938290 17782868 17606060 3
```

Question 9

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

```
std_data_2015_18 <- std_data[which(std_data$Year %in% c("2015", "2016", "2017", "2018")),]
dim(std_data_2015_18)
```

```
[1] 2124    6
```

Question 10

Your colleague is interested in this data set 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(std_data_2015_18, "~/PHW251_2024/problem_sets/data/std_data_2015_18.csv")
#read.csv("~/PHW251_2024/problem_sets/data/std_data_2015_18.csv")
```

Question 11

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

```
unique(std_data$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 county of your choice.

```
alameda_std_data <- std_data %>% filter(County == "Alameda")
```

Question 12

You're very interested in finding the rate of cases per 100,000 population. In your subset data frame (from the previous question), create a new column called "rate" with the calculated values.

Rate = (Cases / Population) * 100,000

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`

```
alameda_std_data <- alameda_std_data %>% mutate(
  Rate = (as.numeric(Cases)/as.numeric(Population)) * 100000
)
```

Question 13

You see a classmate post the comment below in an Ed Discussion.

I can't get my file to import right:

rm(list = ls())

step 1 find file

step 2 get file into r folder

step 3 check with classmate

co2 <- read_csv(file = "co2_mm_mlo.csv", skip = 51, header = TRUE, row.names = NULL)

Write a response to the poster and include at least three specific suggestions for them to include to make their response fit guidelines for repeatable examples.

-Use public shareable data in lieu of the csv file, or if your file is shareable, share a subset of the data.

-Include all necessary libraries and the R version you are using.

-Identify which part of the code you are having trouble with. Run each line separately to do so.

-List all the solutions you've tried, including what happened and what you expected to happen.

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