Assignment 1: Manipulation of birthweight data

Your name and student ID

Today's date

Instructions

- Due date: Thursday, July 9 at 10:00pm.
- Remember: autograder is meant as sanity check ONLY. It will not tell you if you have the correct answer. It will tell you if you are in the ball park of the answer so *CHECK YOUR WORK*.
- Submission process: Follow the submission instructions on the final page. Make sure you do not remove any `\newpage` tags or rename this file, as this will break the submission.

Helpful hints:

- Every function you need to use was taught during lecture! So you may need to revisit the lecture code to help you along by opening the relevant files on Datahub. Alternatively, you may wish to view the code in the condensed PDFs posted on the course website. Good luck!
- Knit your file early and often to minimize knitting errors! If you copy and paste code for 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 you can detect the source of knitting errors more easily. This will save you and the GSIs from frustration! **You must knit correctly before submitting.** Remember, knitting does two things: it saves you work and generate a PDF. The submission process turns in your last **saved** work so YOU MUST KNIT BEFORE SUBMITTING.
- If your code runs off the page of the knitted PDF then you will LOSE POINTS! To avoid this, have a look at your knitted PDF and ensure all the code fits in the file (you can easily view it on Gradescope via the provided link after submitting). If it doesn't look right, go back to your .Rmd file and add spaces (new lines) using the return or enter key so that the code runs onto the next line.

Begin by knitting this document by pushing the "Knit" button above. As you fill in code and text in the document, you can re-knit (push the button again) and see how the document changes. It is important to re-knit often, because if there is any error in your code, the file will not generate a PDF, so our advice is to knit early and often!

Using `dplyr` to investigate birthweight

The data file `birthweight.csv` contains the birthweights for babies at several different hospitals. Hit the green arrow icon in the line below to execute the two lines of code in the code chunk, or execute them line by line by placing your cursor on the first line and hitting cmd + enter on Mac or ctrl + enter on PC.

```
library(dplyr)
library(readr)
birthwt <- read_csv("birthweight.csv")
```

```
## Parsed with column specification:
## cols(
##   low = col_double(),
##   age = col_double(),
##   wt_mom = col_double(),
##   smoke = col_double(),
##   premature = col_double(),
##   ht = col_double(),
##   doc_visits = col_double(),
##   wt_birth = col_double(),
##   hospitals = col_character()
## )
```

- The `library` command loads the library `dplyr` into memory.
- The `readr` library contains functions to read in the dataset.
- The `dplyr` library contains functions we will use to manipulate data.

Notice that an object called `birthwt` appeared in the Environment tab under "Data".

1. [2 points] Use four useful functions discussed in lecture to examine the birthweight data set:

```
# Text inside a code chunk that begins with "#" is called a comment.
# We sometimes use comments to explain code to you in plain English.
# Write your four functions below these comments, replacing the placeholder
# text "<<<<YOUR CODE HERE>>>>". Remember, code does *not* begin with a "#"

"<<<<YOUR CODE HERE>>>>"
```

```
## [1] "<<<<YOUR CODE HERE>>>>"
```

```
"<<<<YOUR CODE HERE>>>>"
```

```
## [1] "<<<<YOUR CODE HERE>>>>"
```

```
"<<<<YOUR CODE HERE>>>>"
```

```
## [1] "<<<<YOUR CODE HERE>>>>"
```

```
"<<<<YOUR CODE HERE>>>>"
```

```
## [1] "<<<<YOUR CODE HERE>>>>"
```

```
# Then, assign p1 to a vector of your function names, in alphabetical order.
# For example, assigning p0 to a vector of fruits looks like this:
# p0 <- c("apple", "banana", "orange")

p1 <- "<<<<YOUR CODE HERE>>>>"

check_problem1()
```

```
## [1] "Checkpoint 1 Error: You should list 4 functions just like in the fruits example above"
## [1] "Checkpoint 2 Error: Incorrect. Check your spelling and review lecture notes!"
##
## Problem 1
## Checkpoints Passed: 0
## Checkpoints Errored: 2
## 0% passed
## --------
## Test: FAILED
```

Description of the variables found in the birthwt dataset:

Column name	Description
low	Indicator of low birthweight
age	Mother's age
wt_mom	Mother's weight at last period
smoke	Indicator of mother's smoking status
premature	Number of premature labors
ht	Indicator of hypetension in pregnancy
doc_visits	Number of dr's visits in 1st trimester
wt_birth	Infant's birth weight
hospitals	Hospital that birth took place

2. [2 points] Write code to select a set of columns. Specifically select the `age`, `wt_mom`, and `smoke` columns. Assign this smaller dataset to a data frame called `birthwt_small`

```
birthwt_small <- "<<<<YOUR CODE HERE>>>>"

check_problem2()
```

```
## [1] "Checkpoint 1 Error: Make sure your final answer is a data frame."
## [1] "Checkpoint 2 Error: Did you subset your columns correctly?"
##
## Problem 2
## Checkpoints Passed: 0
## Checkpoints Errored: 2
## 0% passed
## --------
## Test: FAILED
```

3. [1 point] To select a range of columns by name, use the ":" (colon) operator. Redo the selection for question 1, but use the colon operator. Assign this to `birthwt_small_colon`. Note that this returns the same dataframe as the previous problem, but is not recommended in practice because it depends on the ordering of the columns and isn't explicit in the columns that are selected, whereas selection by name offers much higher readability for someone else looking at your code later on.

```
birthwt_small_colon <- "<<<<YOUR CODE HERE>>>>"

check_problem3()
```

```
## [1] "Checkpoint 1 Error: Make sure your final answer is a data frame."
## [1] "Checkpoint 2 Error: Did you subset your columns correctly?"
##
## Problem 3
## Checkpoints Passed: 0
## Checkpoints Errored: 2
## 0% passed
## --------
## Test: FAILED
```

4. [1 point] Select all the columns except for the `ht` variable. Assign this to `birthwt_no_ht`.

```
birthwt_no_ht <- "<<<<YOUR CODE HERE>>>>"

check_problem4()
```

```
## [1] "Checkpoint 1 Error: Make sure your final answer is a data frame."
## [1] "Checkpoint 2 Error: Did you subset your columns correctly?"
##
## Problem 4
## Checkpoints Passed: 0
## Checkpoints Errored: 2
## 0% passed
## --------
## Test: FAILED
```

5. [1 point] Run the following chunk of code.

```
select(birthwt, starts_with("wt"))
```

```
## # A tibble: 189 x 2
##    wt_mom wt_birth
##     <dbl>    <dbl>
##  1    182     2523
##  2    155     2551
##  3    105     2557
##  4    108     2594
##  5    107     2600
##  6    124     2622
##  7    118     2637
##  8    103     2637
##  9    123     2663
## 10    113     2665
## # ... with 179 more rows
```

What does it return? Uncomment one of the possible choices.

```
# p5 <- "returns the number of columns that start with wt"
# p5 <- "returns all columns that start with wt"
# p5 <- "returns all rows that start with wt"

check_problem5()
```

```
## [1] "Checkpoint 1 Error: Incorrect. Try again!"
##
## Problem 5
## Checkpoints Passed: 0
## Checkpoints Errored: 1
## 0% passed
## --------
## Test: FAILED
```

6. [1 point] Rewrite the previous chunk of code using the pipe operator. Assign this to `birthwt_wt`.

```
birthwt_wt <- "<<<<YOUR CODE HERE>>>>"

check_problem6()
```

```
## [1] "Checkpoint 1 Error: Make sure your final answer is a data frame."
## [1] "Checkpoint 2 Error: Did you subset your columns correctly?"
##
## Problem 6
## Checkpoints Passed: 0
## Checkpoints Errored: 2
## 0% passed
## --------
## Test: FAILED
```

7. [1 point] Filter the rows for infants that weigh less than 2500 grams from your full dataset `birthwt`. Assign this to `birthwt_low`. Note: infants that weigh less than 2500 grams are considered low birthweight

```
birthwt_low <- "<<<<YOUR CODE HERE>>>>"

check_problem7()
```

```
## [1] "Checkpoint 1 Error: Make sure your final answer is a data frame."
## [1] "Checkpoint 2 Error: Did you subset from the right data frame?"
## [1] "Checkpoint 3 Error: Did you subset your rows correctly?"
##
## Problem 7
## Checkpoints Passed: 0
## Checkpoints Errored: 3
## 0% passed
## --------
## Test: FAILED
```

8. [2 points] Filter the rows for infants that weigh less than 2500 grams and whose mothers experience at least one other premature labor. Assign this to `birthwt_prem`.

```
birthwt_prem <- "<<<<YOUR CODE HERE>>>>"

check_problem8()
```

```
## [1] "Checkpoint 1 Error: Make sure your final answer is a data frame."
## [1] "Checkpoint 2 Error: Did you subset from the right data frame?"
## [1] "Checkpoint 3 Error: Did you subset your rows correctly?"
##
## Problem 8
## Checkpoints Passed: 0
## Checkpoints Errored: 3
## 0% passed
## --------
## Test: FAILED
```

9. [1 point] Go back to the full dataset, `birthwt`. Suppose you are specifically interested in infants who were born at Seattle Grace, Sacred Heart, and Princeton Plainsboro. Assign `birth_hosp` to a data frame for births from these hospitals only.

```
#make sure to pay attention to capitalization!
birth_hosp <- "<<<<YOUR CODE HERE>>>>"

check_problem9()
```

```
## [1] "Checkpoint 1 Error: Make sure your final answer is a data frame."
## [1] "Checkpoint 2 Error: Did you subset from the right data frame?"
## [1] "Checkpoint 3 Error: Did you subset your rows correctly?"
##
## Problem 9
## Checkpoints Passed: 0
## Checkpoints Errored: 3
## 0% passed
## --------
## Test: FAILED
```

10. [1 point] Order the dataset `birth_hosp` by birthweight from lowest birthweight to highest birthweight. Assign this to `birth_order`.

```
birth_order <- "<<<<YOUR CODE HERE>>>>"

check_problem10()
```

```
## [1] "Checkpoint 1 Error: Make sure your final answer is a data frame."
## [1] "Checkpoint 2 Error: Did you arrange the right data frame?"
## [1] "Checkpoint 3 Error: You should have the same number of rows as birth_hosp."
##
## Problem 10
## Checkpoints Passed: 0
## Checkpoints Errored: 3
## 0% passed
## --------
## Test: FAILED
```

11. [1 point] Now reverse the order and assign it to `birth_rev`.

```
birth_rev <- "<<<<YOUR CODE HERE>>>>"

check_problem11()
```

```
## [1] "Checkpoint 1 Error: Make sure your final answer is a data frame."
## [1] "Checkpoint 2 Error: Did you arrange the right data frame?"
## [1] "Checkpoint 3 Error: You should keep all observations."
##
## Problem 11
## Checkpoints Passed: 0
## Checkpoints Errored: 3
## 0% passed
## --------
## Test: FAILED
```

12. [2 points] Suppose you are interested in the order of birthweight, but according to the smoking status of the mother. Returning to the full dataset `birthwt`, order the dataset first by the smoking status of the mother, then the birthweight from lowest to highest. Assign this to `birthwt_smoke`. Note: smoke == 1 means the woman is a smoker.

```
birthwt_smoke <- "<<<<YOUR CODE HERE>>>>"

check_problem12()
```

```
## [1] "Checkpoint 1 Error: Make sure your final answer is a data frame."
## [1] "Checkpoint 2 Error: Did you arrange the right data frame?"
## [1] "Checkpoint 3 Error: You should keep all observations."
##
## Problem 12
## Checkpoints Passed: 0
## Checkpoints Errored: 3
## 0% passed
## --------
## Test: FAILED
```

13. [1 point] Create a new column called `wt_birth_lbs` which is the birthweight of the infants in pounds. Add this to a new dataset `birthwt_lbs`. Note: 1 gram = .0022 pounds

```
birthwt_lbs <- "<<<<YOUR CODE HERE>>>>"

check_problem13()
```

```
## [1] "Checkpoint 1 Error: Make sure your final answer is a data frame."
## [1] "Checkpoint 2 Error: Do you have the correct number of columns?"
## [1] "Checkpoint 3 Error: You should keep all observations."
##
## Problem 13
## Checkpoints Passed: 0
## Checkpoints Errored: 3
## 0% passed
## --------
## Test: FAILED
```

14. [1 point] Calculate the average birthweight across all the infants in the dataset using a `dplyr` function and assign it to the variable `avg_birthwt`. Your answer should be a data frame of 1 observation and 1 variable called `bw_avg`

```
avg_birthwt <- "<<<<YOUR CODE HERE>>>>"

check_problem14()
```

```
## [1] "Checkpoint 1 Error: Make sure your final answer is a data frame."
## [1] "Checkpoint 2 Error: Did you use the correct dplyr function?"
## [1] "Checkpoint 3 Error: Did you calculate the mean?"
##
## Problem 14
## Checkpoints Passed: 0
## Checkpoints Errored: 3
## 0% passed
## --------
## Test: FAILED
```

15. [2 points] Calculate the average birthweight according to whether the mother is a smoker or a non-smoker. Hint: you'll need to use two `dplyr` functions! The column names should be `smoke` and `bw_avg`.

```
avg_by_smoker <- "<<<<YOUR CODE HERE>>>>"

check_problem15()
```

```
## [1] "Checkpoint 1 Error: Make sure your final answer is a data frame."
## [1] "Checkpoint 2 Error: Did you use the group_by function?"
## [1] "Checkpoint 3 Error: Did you use the summarize function?"
##
## Problem 15
## Checkpoints Passed: 0
## Checkpoints Errored: 3
## 0% passed
## --------
## Test: FAILED
```

16. **BONUS** [+4 points] At the end of every homework, you will have the opportunity to obtain up to 4 extra credit points on that assignment.

+1 point for developing a multiple choice question on the material in the homework

+1 point for developing a true/false question on the material

+2 points for developing a free response question.

To get full credit, you need to write the question, write an answer, provide a rubric for grading the answer, and say something about why/how you feel this question addresses a concept in this homework. You can only write **one question per category**.

[YOUR ANSWER HERE]

Check your score

```
# Just run this chunk.
total_score()
```

```
##                      Test Points_Possible          Type
## Problem 1          FAILED               2    autograded
## Problem 2          FAILED               2    autograded
## Problem 3          FAILED               1    autograded
## Problem 4          FAILED               1    autograded
## Problem 5          FAILED               1    autograded
## Problem 6          FAILED               1    autograded
## Problem 7          FAILED               1    autograded
## Problem 8          FAILED               2    autograded
## Problem 9          FAILED               1    autograded
## Problem 10         FAILED               1    autograded
## Problem 11         FAILED               1    autograded
## Problem 12         FAILED               2    autograded
## Problem 13         FAILED               1    autograded
## Problem 14         FAILED               1    autograded
## Problem 15         FAILED               2    autograded
## Problem 16 NOT YET GRADED               0 free-response
```

Submission

For assignments in this class, you'll be submitting using the **Terminal** tab in the pane below. In order for the submission to work properly, make sure that:

1. Any image files you add that are needed to knit the file are in the `src` folder and file paths are specified accordingly.
2. You **have not changed the file name** of the assignment.
3. The file is saved (the file name in the tab should be **black**, not red with an asterisk).
4. The file knits properly.

Once you have checked these items, you can proceed to submit your assignment.

1. Click on the **Terminal** tab in the pane below.
2. Copy-paste the following line of code into the terminal and press enter.

cd; cd ph142-su20/hw/hw01; python3 turn_in.py

3. Follow the prompts to enter your Gradescope username and password. When entering your password, you won't see anything come up on the screen–don't worry! This is just for security purposes–just keep typing and hit enter.
4. If the submission is successful, you should see "Submission successful!" appear as output.
5. If the submission fails, try to diagnose the issue using the error messages–if you have problems, post on Piazza.

The late policy will be strictly enforced, **no matter the reason**, including submission issues, so be sure to submit early enough to have time to diagnose issues if problems arise.