**Study Inclusion / Selection & Graphics with ggplot2**

Once again, the files you need to submit are your **R script and this document in word or PDF by 11:59pm on the due date**. Paste in your code and output into this document where it is specifically requested. If directions are given but nothing is asked directly, an “OK” response will suffice. Name your files following this convention (where X is the number of the HW) and email them to Mike:

epid799b\_hwX\_lastname.pdf (Homework document with answers filled in)

epid799b\_hwX\_lastname.r (R script).

**Readability.** A reminder, please, for your sake and the instructor’s, make efforts to comment, create code blocks, and write legible code. However, on answering questions in this document, please write as little as you can – short answers and sentence fragments are encouraged.

**Recoding**. Homework 2 extends on the work of homework 1, meaning these scripts should be built on top of the recoded variables from homework 1. Please update your HW 1 looking at the released HW 1

Complete the follow steps in R/RStudio:

# Part 1: Subset for our study inclusion criteria

1. **Create useful calculated variables**
   1. **Week number**: Create the weeknum variable in the births dataset using the week() function in lubridate and the date of birth (dob) variable in already in.
2. **Create simple exclusion variables**We will now exclude births from the study to simplify and hone our research question. Specifically, we’ll look at births that have data on our variables of interest, had all their weeks of gestation at risk of preterm birth within our study year, are single births, and have no congenital anomalies. In each case we’ll create a variable starting with “incl\_” in the dataset so we can keep track of various criteria for study inclusion, with the variable = 1 meaning it passes that inclusion test, and 0 meaning it does not pass. We’ll use \*apply() functions where helpful to help reduce our code. The as.numeric() function may be helpful to cast Booleans to 1s and 0s, though T and F are interpreted intelligently as 1 and 0 as well – these instructions are based on 1s and 0s, but you may use T and F if that makes more sense to you.
   1. **Has Gestation**: Create incl\_hasgest in the births dataset to be a 1 if the birth has wksgest data, 0 if missing.
   2. **Enough gest:** Create incl\_enoughgest in the births dataset to be a 1 if the birth has >= 20 weeks of gestation, 0 if otherwise.
   3. **Late enough**: Create incl\_lateenough to be 1 when wksgest-weeknum <= 19, 0 otherwise.
   4. **Early enough**: Create incl\_earlyenough to be 1 when weeknum - wksgest <= 7, 0 otherwise.
   5. **Singletons**: Create incl\_singleton to be 1 when plur is 1, 0 otherwise.
3. **Create congenital anomaly variable using apply()**There is no single “has any congenital anomaly” variable, so we’ll need to create one. Some hints: Use the MARGIN parameter of apply() to decide between rows and columns (we’ll want rows). The dataframe to pass to apply() will need to be subsetted to just the congen\_anom variables. It may be easiest to send that dataframe into apply() as all Booleans (e.g. df[subsetting-goes-here] ==”N”).  
   1. **Create name vector**: Create a variable congen\_anom (outside births) to be equal to a vector of the names of congenital anomaly variables. You would normally refer to the meta data, but to save you typing, here are those variables in an array:  
      c("anen","mnsb","cchd", "cdh", "omph","gast", "limb", "cl","cp","dowt","cdit", "hypo")
   2. **Check all anomalies at** once: Use apply() with the all() function on the congenital anomaly columns of births to return a 1 if all congenital anomaly variables are “N”, 0 otherwise and assign it to variable incl\_noanomalies.
4. **Finish the inclusion criteria**
   1. **Save** your dataset, before we subset it, as old\_births. If you need to in the future, you can birth=old\_births to reset your work, since this dataset is relatively small (sometimes I’ll save these sorts of save points in a comment... ).
   2. **Explore the grepl function**. To take advantage of our incl\_ naming convention, we’ll use a text search function, grepl. As an example, try (and note the misspelling of banana) : grepl("a", c("banana", "peach", "ornge"))
   3. **(Optional Challenge)**: In one line of R, report the number that failed each inclusion test using apply() and sum over the column margins instead of rows. The grepl function may be useful to subset births to the inclusion criteria of interest.
   4. **Create overall inclusion variable**: Create a variable in our dataset include\_allpass that represents a 1 if all inclusion criteria are passed. Using the grepl() function to find just the variables with incl\_ in their name, subset births to be just those that have a 1 in every incl\_ variable. This is similar to what’s done in 3c, above, but looks at rows instead of columns.
   5. **Subset** to create a new births dataset on the outcome of that apply function, just including births that pass the inclusion criteria.
   6. **Summarize exclusion:** Report the number of births before and after the inclusion criteria using nrow() or dim().
5. **Optional Challenge:** Create an “any exclusion” variable in oldbirths (before we subset it). Use tableone::CreateTableOne grouped by that variable to describe the difference in the exclusion and inclusion cohorts.