**Data Management with R**

The files you need to submit are your **R script and this document in PDF**. 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. Please name your files following this convention:

epid799b\_hw3\_lastname.pdf (Homework document with answers filled in)

epid799b\_hw3\_lastname.r (R script).

**Readability.** Both your graders, and in “real” life, your colleagues and future-you will appreciate greatly your well-written, structured and documented code. Please continue to explore readability habits as you code. You may need certain packages that are not pre-installed. If needed, install the packages first, and load the packages by using the command **library(*functionname*)**. Complete the follow steps in R/RStudio:

1. **Read & subset the file**As usual, let’s read in births.csv, similar to how we’ve done in the past (a data.frame called ‘births”, stringsAsFactors=F, and lowercase names. If you wish to subset the variables, for this assignment we’re looking at: "MAGE", "SEX", "MDIF", "VISITS", "WKSGEST", "MRACE", "CIGDUR", "DOB", "DOBMONTH".
2. **Factors: sex**Let’s take a moment to explore factors.   
   *NB Social Epi note: we are largely ignoring discussions of the sex/gender context here, but in this case, working with birth data, we’ll use a biological sex binary.*
   1. Create a character vector named sex with three values: (“F”, “F”, “F”)
   2. Convert that character vector to a factor called sex\_f using no parameters.
   3. Call table() on that character vector. What is returned?
   4. Reconvert g to factor sex\_f, but use the factor() parameters to specific the possible values M and F (binary gender in this case).
   5. Call table() again. What is different?
   6. Create a factor variable births$sex\_f in the births dataset with two levels M and F, and in the same call, assign missing values to NA. Run table() on it, using the right parameter to show missing values.
   7. Do the same for smoke\_f, with levels Yes, No, and Missing using the levels and labels arguments.
3. **Dates: birth date**Let’s look at dates. This year’s birth file already breaks out year, month and day in separate variables. Let’s verify it’s done right.  
   *NOTE: If you work with dates often, you might check out the “lubridate” package. For now we’ll stick to base R.*
   1. Use str() or head() to look at the births$dob variable. What format is it in?
   2. Use as.POSIXct() with the appropriate format to convert the birth from its format to an R date format.
   3. Now let’s extract the month element: Use as.integer() and strftime() with the right format to pull the month element into a dob\_monthcheck variable in births.
   4. Create a table of the old and new birth months to verify we did it right.
4. **Recoding**: Creating factors with cut()  
   On the subject of factors, sometimes there are a lot of levels, and it’s easier to categorize them all at once in groups. Let’s use the cut() function to create a factor.
   1. Create births$pnc5\_f as a factor using the cut functions. Assign the missing value first, then set the breaks to 1,4,9 and 88, with labels “Early PNC (1-4)”, “Late PNC (5 to 9)”, and “No PNC”. Note the set ( [ marks, or the “include.lowest” parameter. Careful about these “fencepost” problems!
5. **Subsetting** w/ Base:
   1. We’ve been subsetting data.frames using base R for some time. Explain how to subset a data.frame’s rows or columns using an array of (1) booleans, (2) integers, and (3) character strings.
6. **Manipulating with dplyr**   
   Now we’ll use dplyr to answer some tabular questions about our birth dataset. Remember: there’s a dplyr cheat sheet built into R…
   1. Explain what each of these dplyr “verbs” do:   
      filter, select, summarize, group\_by, mutate
   2. What is the “pipe” operator, and how does it work?
   3. We’ll be working with the county data, but that’s in another table. Read in county\_table.csv into county\_table.
   4. The “base” way to merge data is with the (you guessed it) merge() function. Create a births\_merged file merging births and county\_table (hint: you’ll need to identify the by.x and by.y variables…). This can also be done with dplyr’s join functions (e.g. left\_join).
   5. Using births\_merged, chain filter(), select(), group\_by() and summarize() together to answer this question: excluding births with congenital anomalies, how many births are ascribed to each county (by county of residence of mother, “cores”) and what was their average gestation (wksgest)? Note that you’ll need to deal with the missing values (currently =99!)
7. **Project: Data cleaning**Respond to the following prompts
   1. How “clean” is your data (in terms of recoding, missing values, etc.), and how do you know? What was the “data generation process” (is this from an experiment? Following over time? Claims? How close is it to “reality?”)
   2. Describe the (planned, or previously completed) process of cleaning and recoding your data – what variables are you using, how will they need to be cleaned or recoded, etc.
   3. If possible, actually clean and recode your data. If your data is already clean (congratulations you lucky duck!), describe who, when and how your data was cleaned.