**Cleaning Up Survival Data.**

What are all the things that you could image that could go wrong with your data collection and how are you going to find them?

* Load the “MessedUpBeyondAllReason.csv” into a data frame called “mubar”. NOTE, this is a “.csv” file and you will want to treat it appropriately in order to get it to load properly. Show the full data set here because it is only 8 rows. This data set should have the data for 2 different DGRP lines (LINE 1 and LINE 2). There should be a single replicate vial for Low Sugar (low) and High Sugar (High) for each line and each replicate vial should have a row for MALE and FEMALE flies.
* How many rows and columns are there?
* This data should look very similar to the format that will be present in your google sheets data set. Look, by eye, at the data and tell me all the errors you find**. EVERY ROW HAS SOME ISSUE, SOME ARE HARD TO SPOT.** DON’T READ ON UNTIL YOU FIND ALL THE PROBLEMS. “R” is very particular about stuff. A is not the same as a, for example.

How can you get find and remove all the wrong data? This is tricky but if you think about it carefully, it can be done AND it needs to be done if you want to have high quality data.

Several of the issues (MaLE vs MALE, c instead of C, wrong line numbers…) can be identified just by counting the number of entries in each of the different columns (you know the function to do this)

BEFORE YOU DO ANYTHING, I WOULD MAKE A NEW DATA FRAME CALLED “mubar.b” which stands for mubar backup. You can go back to this if you think there are any problems and you don’t have to start over from scratch if you make a mistake that really messes things up. As you progress, I will “encourage” you to continue to make these backups.

* Make a new data frame called “mubar.b” and store the “mubar” data in it. If you make a mistake, you can take the data that is in “mubar.b” and put it back into “mubar”.
* How many different DGRP LINES are there and how many different entries are there for each LINE?
* Is this what the data is supposed to look like? Do you think there are 3 LINES? Based on what the data is supposed to look like is it fair to change the value of “26” to “2”?
* Change the LINE “26” to read “2” and print the data to the screen to verify the change.

Now work through similar logic for mubar$TREAT and mubar$SEX and mubar$REP and find the errors and fix them if you are **absolutely certain** that you can determine things that are just typos. If you are not absolutely certain about typos, delete the entire entry.

* Identify and fix any problems with mubar$TREAT. Use the approach that you already learned that DOES NOT require the which() function.
* Verify that the error has been corrected using the table() function.
* Identify and fix any problems with mubar$SEX. Again use the approach that DOES NOT require the which() function.
* Verify that the change has been made using the table() function.
* Identify any problems with mubar$REP. Are there any?

Now we can start investigating the values in the DAY columns but before we do that let’s save “mubar” in “mubar.b”.

* Verify that “mubar” is a data frame and if it is, save “mubar” as “mubar.b” so you have a backup that is correct up to this point.

Is there any missing data and if so, where in the data set? We found out how to do this in previous assignments, but there is a cool argument that can be changed for the which() function. Check out the “arr.ind” argument. The default setting is “FALSE” but it might be helpful to change it to “TRUE”.

* Use the which() function to determine where the missing data is. Use the default setting for “arr.ind” and also change it to “TRUE”
* How many rows of data have missing data and will need to be removed?
  + You don’t need to do any additional R commands here, base your answer on the above commands? I am asking this question because it is possible that a single row of data has more than 1 value of missing data so it would appear that you would need to remove multiple rows but in reality you only need to remove a single row of data. Note: it is not always required to remove replicates with missing data from the data set prior to analysis because most R functions have ways to deal with it. BUT, for simplicity and consistency and ease of grading for the TAs, we are are going to do it in this class.
* Remove any row that has missing data and save the resulting data frame back into the current “mubar” data frame using the the na.omit() function.

As is the cases with many things in R, there are many ways to remove missing data. Another useful function is the complete.cases() function. This command says replace the data frame “mubar” with only those rows of “mubar” that have complete data (i.e. no NA). Here is the script for that.

mubar<-mubar[complete.cases(mubar),]

OR, take the information from the which function (with arr.ind = TRUE) and delete the row that has missing data by hand (note, it is rare that you would want to do it this way, but it is good to know how). Here is the script for that noting that you previous identified row 2 and the row with the missing data.

mubar<-mubar[-c(2),]

* Verify that data set is of the correct size after removing you have removed the row with the missing value in it.
* And just for good measure, verify that there is no missing data.
* If you have done everything correctly, I would resave this data frame as a new “backup” called “mubar.b2”. Now you have the original back and another backup that is a bit farther along (just in case you realize you messed up).

Another type of data that we are going to remove are the cases where there were no flies of a given sex in the original, DAY\_1 vial. The subset() function is a great way to do this. Note, there are different ways to form the correct logical operator to do this. For example, you can include specific data or exclude specific data and some ways are better than other ways. It is usually always better to only include the data that you want because it is often easier to know all the things you what you rather than knowing all the things you don’t want

* Use the subset() function to select only those rows where DAY\_1 has at least 1 fly. Save this back into the mubar data frame.

In theory, you could select the cases where the DAY\_1 values are not equal to 0. As shown below:

mubar<-subset(mubar,mubar$DAY\_1!=0).

Note, the way you did it above is a approach is better because it will also remove any cases where the DAY\_1 values are negative.

* Verify that your “mubar” looks like this.

> mubar

LINE TREAT SEX REP DAY\_1 DAY\_3 DAY\_5 DAY\_8

1 1 Low MALE 1 3 4 2 2

3 1 High MALE 1 4 4 2 2

4 1 High FEMALE 1 122 1 1 1

5 2 LOW MALE 2 3 2 2 1

6 2 Low FEMALE 2 4 1 1 0

7 2 High MALE 2 3 3 2 -1

* If you have done everything correctly, I would resave this data frame as a new “backup” mubar.b3

Now you should have a data frame that has 6 rows and 8 columns. We have cleaned up the naming of LINES, SEX and TREAT and removed rows with missing data or no flies at the start. Now let’s tackle the problem of entries that clearly have the wrong number of flies entered either because the entries are negative (you can’t have a negative number of flies) or because there are too many flies (you were supposed to have 10 flies in there.) The latter criterion is a bit subjective but let’s set the cutoff to be “15”.

* Before you do this, copy your “mubar” object into a new object called “temp”. We will use this in a bit later.
* Include only those row of data that have fly numbers that are less than “15” AND greater than or equal to “0”. One way to do this is to do a series of subset() functions for each of the different days (DAY\_1, DAY\_3…). Note, you can use “&” to combine logical operations. You will have to do this 4 times, once for each of the data columns
* Print “mubar” to the screen and verify that it looks like this.
* If you have done everything correctly, I would resave this data frame as a new “backup” mubar.b4

Presumably, some of you will have noticed that with your actual data set, you will need to go through a very large number of columns. This will not be fun. There are a couple of ways to get around this. This first is to write a for loop. Read about them here. <https://www.datamentor.io/r-programming/for-loop/>

**BONUS POINT.** Write a loop to solve this problem and apply it to “temp”.

Finally, we need to remove any of the entries where flies mysterious “reanimate” themselves (i.e. when there are more flies alive at a later time point compared to an earlier time point). The most intuitive way to do this (at least for me) is to make new columns in “mubar” that are the difference in the number of flies between the subsequent time points and then subset the data based on a logical operation to find those cases where flies “reanimate”.

* Make 3 new variables (columns) in the data frame “mubar” called D\_3\_1, D\_5\_3, and D\_8\_5 that are respectively, the difference in the number of flies between DAY\_3 and DAY\_1 (DAY\_3 minus DAY\_1), the difference in the number of flies between DAY\_5 and DAY\_3, and the difference in the number of flies between DAY\_8 and DAY\_5. Subtract the earlier day from the later day.
* Print “mubar” to the screen.
* How do you identify those values for D\_7\_0, D\_14\_7, and D\_21\_7 that indicate flies have “reanimated”?
* Save “mubar” as “mubar.b4”
* Subset “mubar” to exclude the rows that have “reanimated” flies and print “mubar” to the screen. Remember, you can use “&” to combine logical operations such that you can check all the difference columns in a single line of code because you will exclude the whole row in ANY of the values are positive.
* If you have done everything correctly, I would resave this data frame as a new “backup” mubar.b5
* To fully clean up the data set, remove the difference columns that you created. Again, you can do this in a single step using the bracket [] indexing if you know the column numbers you are trying to get rid of. Print the final “mubar” data frame to the screen.

Your data should look like this

LINE TREAT SEX REP DAY\_0 DAY\_7 DAY\_14 DAY\_21

3 1 M MALE 1 4 4 2 2

5 2 C MALE 2 3 2 2 1

6 2 C FEMALE 2 4 1 1 0

* Turn your cleaned data to a new data frame called “nolongermubar” and use the write.table() function to save the data to a text file called “nolongermubar.txt”. In this case, have it saved it with column names but without quotes and without row names.

Now you have created a text file with your cleaned up data that you have saved to your working directory. You will be able to open the text file later to further analyses.