**Tidyverse “verbs” cheatsheet**

In tidyverse, we use a series of “verbs” to wrangle data.

Your verbs are:

1. ***filter()*** - Subsets (AKA “filters”) data by row. To subset, we use *logical operators*. Logical operators set conditions by which to filter your data. Your logical operators are:
   1. **==** Which means “Is *exactly* equal to”
   2. **!=** Which means “Is NOT equal to”
   3. **>** Which means “Is greater than”
   4. **<** Which means “Is less than”
   5. **>=** Which means “Is greater than OR equal to”
   6. **<=** Which means “Is less than OR equal to”
   7. **is.na()** Which means “Contains an NA value”.
   8. **!is.na()** Which means “Does NOT contain an NA value”
   9. **%%** - is called a “modulo”, which means “Has a remainder of” (It’s not used as much, but is actually very useful)
   10. Here’s the basic format: filter(*dataset*, variable == condition)
       1. “==” can be whatever logical operator works best for your purposes.
       2. Or, if you are working with the “is.na()” logical operators, the variable is contained in the parentheses. filter(*dataset*, is.na(variable))
   11. You can also combine logical operators using “&” and “|”
       1. For example, if you use a “&”, then you can say things like “filter(dataset, var > 3 & var < 7)” which will give you values between 3 AND 7. Or the “|” is equivalent to “or” which can be used like “filter(*dataset*, var == 9 | var == 15) will give you rows that the values are 9 OR 15.
2. ***select() -*** subsets (AKA“selects”) variables that you are interested in from a dataset.
   1. Write the name of the variables you would like to keep, separated by a comma within parentheses. You can use a “-” in front of the variable name to NOT include that variable.
   2. Here’s the basic: “select(*dataset*, all, of, the, variables, you, want, -or, -do, -NOT, -want, separated, by, commas)”
3. ***mutate() -*** manipulates (AKA“mutates”) variables to get you something interesting.
   1. You can use this to rename variables, or do simple math calculations, etc. etc.
   2. When mutating, you will always be creating a new variable.The “=” sign is telling he mutate function you want to create a new variable.
   3. Here’s the basic format for mutate(): mutate(dataframe, NewVariable = OldVariable mutation instructions)
4. ***arrange() -*** sorts (AKA“arranges”) your data. You can sort on as many variables as you want.
   1. Simply put the variable(s) you would like to arrange in ascending order within the parentheses.
   2. Here’s the general format: “arrange(*dataset*, variables, to, sort, by)”
      1. ***NOTE:*** There is a hierarchy, first it will arrange by the first variable, then the second, then 3rd and so on. . . So make sure you know what order works best!
5. **summarize() - “**summarizes” variables for you. We can get means, standard deviation, median, minimum, maximum and much much more. It will collapse cases for easier descriptive interpretations. There are a lot of summarizing functions, but here are the most essential:
   1. **Mean** = mean(variable, na.rm = TRUE)
   2. **Standard deviation** = sd(variable, na.rm = TRUE)
   3. **Median** = median(variable. na.rm = TRUE)
   4. **Minimum** = min(variable, na.rm = TRUE)
   5. **Max** = max(variable, na.rm = TRUE)
   6. **Interquartile range** = IQR(variable, na.rm = TRUE)
   7. **N of dataset**: n() <- because you specified you dataset in the verb already, this just stays blank.
   8. Just like with mutate() verb, you will actually be creating a new variable every time you use the summarize() verb. Therefore, you need to incorporate the “=” sign is telling he mutate function you want to create a new variable.
   9. Here’s the general format: “summarize(***dataframe****,* new\_name = summarizefunction())”