Problem Set #2 Cheat Sheet

Victoria, Menissah, Sarayu

Key Terms:

Function: Pre-written pieces of code that accomplish some task; they generally follow three steps: Input, Process, Return

Data Frame: An object where the type is a list; datasets are data frames

Length (in context of data frame): The number of variables in a data frame

Element Name (in context of data frame): Variable name

NA: Missing value, different from character value "NA"

Filter: Selecting observations based on values of variables

Tips and Tricks:

- **If your code isn't running, check punctuation and formatting**
  - Backtick (`) vs. apostrophe (')
  - Spaces between characters and values
  - Indentation
  - Etc.

- **Make sure your code aligns with the variable values in your data frame**
  - Decimals vs. Whole Numbers, etc.

- **Use shortcuts, when applicable**
  - https://support.rstudio.com/hc/en-us/articles/200711853-Keyboard-Shortcuts
  - 'ALT'+ '-' gets you the assignment operator '<-' only works in r code not in text

- **Explore alternate r packages if something isn't working**
  - Tinytex, smaller and simpler to install then MikeTex

```
install.packages('tinytex')
tinytex::install_tinytex
```

Cannot have another latex program installed (Miktex for examples)

- **Remember that there is often more than one way to accomplish a function**

Example with filters (PS2, Section 3, Question 3):

VERSION 1:

```
uni_alabama <- filter(df_school_all, visits_by_100751 >= 1, school_type == "public", state_code == "AL")

count(filter(uni_alabama,pct_black >= 50.0000 | pct_hispanic>= 50.0000))
```

VERSION 2:

```
count(filter(df_school_all, state_code == "AL", school_type == "public", visits_by_100751 >= 1, pct_black >= 50|pct_hispanic >= 50))
```

Problem Set #3 Cheat Sheet

Anne, Esthela, Laura, Lauren

Key Terms:

Arrange: Sort the observations/rows in a data frame in ascending or descending (using `desc()` cording to one or more variables.

Pipe (%>%): The pipe operator helps you perform multiple steps sequentially (left to right) in one line of code. It is only available in the tidyverse suite.

Input variable: The (possibly raw) data that is used to create new variables for analysis (rather than replacing the original data). It is a good idea to investigate an input variable before you create new variables. Try using the summary() function to find out mean, median, min/max, using range() to find min/mad, and/or looking into missing data.

Analysis variable: The variables used in your analyses. Many of these will be created using `mutate()`

Assign: By using the assignment operator (<-), you can save an object to a data frame (new or existing). Assignment occurs at the beginning of a line of code, even when using pipes.

Tips and Tricks

- **To limit the number of rows displayed (i.e., first 15)**
  - head () function
  - Examples:
    - Without pipes

```
head(select(arrange(df_school_all, desc(total_visits)), name, state_code,
city, school_type, total_visits, med_inc, pct_white, pct_black,
pct_hispanic, pct_asian, pct_amerindian), n=15)
```

    - With pipes

```
df_school_all %>%
   arrange(desc(total_visits)) %>%
      select(name, state_code, city, school_type, total_visits, med_inc,
      pct_white, pct_black, pct_hispanic, pct_asian, pct_amerindian) %>%
         head(n=15)
```

- **Assignment problems**
  - If you encounter problems with assignment, try creating a new data frame with each test of your code, so that you don't have to keep reloading the data to work with it. You can keep track of what happens with each change, taking notes along the way. Example:

```
df_school_all2 <- df_school_all %>% select(num_took_math) %>%
mutate(miss_took_math = ifelse(is.na(num_took_math),1,0)) %>%
count(miss_took_math)
   #This creates a data frame with 2 obs and 2 variables

df_school_all3 <- df_school_all %>% select(num_took_math) %>%
mutate(miss_took_math = ifelse(is.na(num_took_math),1,0))
   #This creates a data frame with 21301 obs and 2 variables
```

- **Issues with knitting (specific to tidyverse)**
  - When loading tidyverse, the output displays check marks that indicate which specific tidyverse packages were installed (e.g., purrr, dplyr, etc).
  - These check marks may not be read by the LaTeX set up because they are special symbols that requires "xelatex".

- In order to knit to PDF, you may have to suppress the output in the chunk by wrapping the library(tidyverse) with another function called suppressMessages (note that there are quotes around tidyverse):

  supressMessages(library("tidyverse"))
- Here is a link where Hadley Wickam comments that the message has to be suppressed in order to knit to PDF:

https://community.rstudio.com/t/tidyverse-1-2-1-knitting-to-pdf-issue/2880/7

- **When using case_when(), the order of your variables matters**
  - The following code works (but creates a value with string name "NA" instead of a missing value):

```
wwlist <- wwlist %>%
 select(for_country, state) %>%
 mutate(residency = case_when(for_country =="No Response" ~ "NA", state == "WA" ~ "in_state", state != "WA" ~ "out_state_us", !is.na(for_country) ~ "not_in_us"))
```

  - The following code does not create any NA values, because `for_country` is not *missing* for the "No Response" students (it is a string: "No Response"), so the students with "No Response" get coded as "not_in_us" during the third case_when statement:

```
wwlist <- wwlist %>%
 select(for_country, state) %>%
 mutate(residency = case_when(state == "WA" ~ "in_state", state != "WA" ~ "out_state_us", !is.na(for_country) ~ "not_in_us", for_country =="No Response" ~ "NA"))
```

  - Here is a better option, which creates three value labels and leaves the 17 students with "No Response" as true missing (<NA>)

```
wwlist <- wwlist %>%
  select(for_country, state) %>%
  mutate(residency = case_when(
    state == "WA" ~ "in_state",
    state != "WA" ~ "out_state_us",
    is.na(state) & for_country !="No Response" ~ "not_in_us"))
```