Week 3 Day 2: Lab 3 Warm-up

# Using the Native Pipe (|>)

In RStudio: Tools > Global Options... > Code > check Use native pipe operator box

Keyboard shortcut: ctrl/cmd + shift + m

## filter()

Do you want to filter a set of values of a variable?

colleges\_clean |>   
 filter(STABBR %in% c("CO", "MT", "ID", "UT", "WY"))

## select()

Do you want to select a range of columns?

colleges |>   
 select(ADM\_RATE:UGDS)

Do you want to select columns that share a common word or symbol?

cereal |>   
 select(name, starts\_with("TUITION"))

Do you want to select columns that share a common data type?

colleges\_clean |>   
 select(INSTNM, where(is.numeric))

## rename()

Change (bad) column names!

1. In the rename() function, what goes on the right hand side of the = sign? What goes on the left hand side?

**Bonus:** If you want to rename variables **and then** only select some variables, use select()!

cereal |>   
 select(name,  
 temp = type,  
 cons\_rating = rating)

# count()

Obtain frequency tables of the number of observations in each level of a categorical variable

1. You could have obtained frequencies of observations by using group\_by() + summarize(). Why is count() a better choice?

## group\_by() + slice\_max()

1. What happens when you use group\_by() before slice\_max()? What output will you get?

## group\_by() + summarize()

1. What groups are dropped after summarize() calculates the statistics?
2. What groups **are not** dropped after summarize() calculates the statistics?
3. How can you make sure all the groups have been dropped after summarize() calculates the statistics?

## if\_else() & case\_when()

if\_else() is designed for **binary** options:

colleges\_clean |>   
 filter(CONTROL != 3) |>   
 mutate(CONTROL = if\_else(CONTROL == 1,   
 "Public",   
 "Private, Non-Profit")  
 )

case\_when() is designed for **three or more** options:

colleges\_clean |>   
 mutate(TYPE = case\_when(CONTROL == 1 ~ "Public",   
 CONTROL == 2 ~ "Private, Non-Profit",   
 CONTROL == 3 ~ "Private, For Profit")  
 )

## across()

Performing the same function across() multiple columns

colleges\_clean |>   
 group\_by(REGION) |>   
 summarize(  
 across(  
 .cols = c(ADM\_RATE, SAT\_AVG, TUITIONFEE\_IN, TUITIONFEE\_OUT, UGDS),   
 .fns = ~ mean(.x)  
 )  
 )

1. How do columns need to be input into the across() function?
2. How do functions need to be input into the across() function?
3. What does the .x inside the function represent?

# across() friends

## if\_any() & if\_all()

Remember, you got warnings in PA3 when converting some columns to numeric? If you look at the original data, you can see this is because missing values were indicated with the string "NULL".

We could drop these rows before converting the columns to numeric if desired, using if\_any():

colleges\_clean <- colleges\_clean |>   
 filter(  
 !if\_any(.cols = ADM\_RATE:TUITIONFEE\_OUT,   
 .fns = ~ .x == "NULL")  
 )

1. How would you describe what ~ .x == "NULL" is doing?
2. What does the ! in front of the if\_any() do?

## Saving Objects

Sometimes you just want to show a table and there is no need to save an object:

cereal |>   
 filter(type == "cold") |>   
 mutate(potass\_per\_cup = potass / cups) |>   
 group\_by(manuf) |>   
 summarise(mean\_potass\_per\_cup = mean(potass\_per\_cup))

Other times, you will want to save your wrangled data for later:

cereal\_summary <- cereal |>   
 filter(type == "cold") |>   
 mutate(potass\_per\_cup = potass / cups) |>   
 group\_by(manuf) |>   
 summarise(mean\_potass\_per\_cup = mean(potass\_per\_cup))

## kable()

library(knitr)

colleges\_clean <- colleges\_clean |>   
 mutate(TUITION\_DIFF = TUITIONFEE\_OUT - TUITIONFEE\_IN)  
   
cereal\_clean |>   
 group\_by(REGION) |>   
 slice\_max(order\_by = TUITION\_DIFF, n = 5) |>   
 kable()