Problem Set #5

NAME HERE

9/23/2020

Due date: Monday, October 5th

At this point in the course we have introduced a fair amount of code, which can be a lot to hold in our memory at once! Thankfully we we have search engines and these helpful cheatsheets. You may find the Base R and Data Transformation Cheatsheet helpful.

Question 1

Below is a function that ought to return TRUE if x is an even number. However, we're having trouble receiving a value despite x == 4, which we know is an even number. Debug and explain why this error is occurring.

```
# note: %% is the "moduluo" operator, which returns the remainder of an operation
# try: 2 %% 2 (should equal 0 as 2/2 has no remainder)
# try: 5 %% 2 (should equal 1 as 5/2 has a remainder of 1)

return_even <- function(x){
  if (x %% 2 == 0) {
    return("I'm an even number!")
  }
}

x <- 4
return_even()
```

```
# answer
return_even <- function(x){
  if (x %% 2 == 0) {
    return("I'm an even number!")
  }
}

x <- 4
return_even(x)
```

```
## [1] "I'm an even number!"
```

R functions are not able to access global variables unless we provide them as inputs.

Below is a function that determines if a number is odd and adds 1 to that number. The function ought to return that value, but we can't seem to access the value. Debug the code and explain why this error is occurring. Does it make sense to try and call odd_add_1 after running the function?

```r
return_odd <- function(y){
  if (y %% 2 != 0) {
    odd_add_1 = y + 1
  }

}

return_odd(3)
odd_add_1
```

```r
# answer
return_odd <- function(y){
  if (y %% 2 != 0) {
    odd_add_1 = y + 1
  }
  return(odd_add_1)
}

return_odd(3)
```

```
## [1] 4
```

```r
#odd_add_1
```

Although we created a variable within our function, these new variables do not transfer to the global environment. Calling odd_add_1 after running the function does not make sense. If we wanted to utilize the odd_add_1 we would need to assign it to another variable or use it in a calculation.

Question

Below is a function that calculates BMI and assumes the weight and height inputs are from the imperial system.

Can you write a function that calculates BMI based on the metric system? You can test your function with the Taiwan data set. Please look at only the first row for the Taiwan data.

```
df_colorado <- read_csv("data/colorado_data.csv")
```

```
## Parsed with column specification:
## cols(
##   location = col_character(),
##   gender = col_character(),
##   height = col_double(),
##   weight = col_double(),
##   date = col_double()
## )
```

```
bmi_imperial <- function(height, weight){
  bmi = (703 * weight)/(height * 12)^2
  return(bmi)
}

# calculate bmi for the first observation
bmi_imperial(df_colorado$height[1], df_colorado$weight[1])
```

```
## [1] 42.62802
```

```
# your code here
df_taiwan <- readxl::read_xlsx("data/taiwan_date.xlsx")

bmi_metric <- function(height, weight){
  bmi = weight/(height/100)^2
  return(bmi)
}

bmi_metric(df_taiwan$height[1], df_taiwan$weight[1])
```

```
## [1] 21.45357
```

Question

Can you write a function that combines both of the BMI functions? You will need to determine a way to evaluate which calculation to perform. Use the first row for the Colorado and Taiwan data to test.

```
calculate_bmi <- function(height, weight){
  # the tallest person was 8 ft 11.1 in tall, which we can use as a crude cutoff
  if_else(height > 9, # conditional statement
          (weight/(height/100)^2), # if true, metric
          (703 * weight)/(height * 12)^2) # if false, imperial
}

calculate_bmi(df_colorado$height[1], df_colorado$weight[1])
```

```
## [1] 42.62802
```

```
calculate_bmi(df_taiwan$height[1], df_taiwan$weight[1])
```

```
## [1] 21.45357
```

Question

It's possible to nest functions within another function (we've actually been doing this many times already in the course). Let's try this idea by first writing a function called feet_to_centimeter() that converts height from feet to centimeters. We created another function for you called lbs_to_kg() that converts weight from pounds to kg.

Once you've created lbs_to_kg(), modify our calculate_bmi() to utilize the the two new functions. We can use the first rows of Colorado and Taiwan to confirm results.

1 lbs == 0.453 kilogram 1 foot == 30.48 centimeter

```
# created for you
lbs_to_kg <- function(lbs){
  return(lbs/2.205)
}

lbs_to_kg(1) # test
```

```
## [1] 0.4535147
```

```
# create feet_to_centimeter
feet_to_centimeter <- function(feet){
  return(feet * 30.48)
}

feet_to_centimeter(1) # test
```

```
## [1] 30.48
```

```
# modify calculate_bmi()
calculate_bmi <- function(height, weight){
  # the tallest person was 8 ft 11.1 in tall, which we can use as a crude cutoff
  if_else(height > 9, # conditional statement
          (weight/(height/100)^2), # if true, metric
          (lbs_to_kg(weight))/(feet_to_centimeter(height)/100)^2) # if false, imperial
}

# values should match above
calculate_bmi(df_colorado$height[1], df_colorado$weight[1])
```

```
## [1] 42.62494
```

```
calculate_bmi(df_taiwan$height[1], df_taiwan$weight[1])
```

```
## [1] 21.45357
```

Question

Question

Just how we use other functions, we can also use our new calculate_bmi() function during a dplyr action. That is, we can *pipe* information into our function. Create a new column called **bmi** with a dplyr call and our new function. Perform this action on both the Colorado data and Taiwan data.

```
df_colorado <- df_colorado %>%
  mutate(bmi = calculate_bmi(height, weight))

df_taiwan <- df_taiwan %>%
  mutate(bmi = calculate_bmi(height, weight))
```

Challenge

1. Create a github account.

2.

Please also submit your pdf to gradescope.

```
# functions within functions:
suppressWarnings(writeLines(readLines("data/big_little_dipper.txt")))
```