PHW251 Problem Set 4

AUTHOR
Val Stacey

PUBLISHED
September 28, 2025

```r
library(tibble)
library(dplyr)
library(tidyr)
library(here)
library(janitor)
```

For this problem set you will tidy up a dataset of 500 individuals. We also want to calculate each individual’s BMI and appropriately categorize them.

Load your data (“../data/500_Person_Gender_Height_Weight.csv”):

Question 1

Clean the column headers to be all lower case, have no spaces, and rename “Location information” to location.

```
df <- df_raw %>%
  rename("location" = "Location.information") %>%
  janitor::clean_names()
```

Question 2

Create a new variable that calculates BMI for each individual.

You will need to navigate the different system of measurements (metric vs imperial). Only the United States is using imperial.

- BMI calculation and conversions:
  - metric: $BMI = weight(kg)/[height(m)]^2$
  - imperial: $BMI = 703 * weight(lbs)/[height(in)]^2$
  - 1 foot = 12 inches
  - 1 cm = 0.01 meter

Although there's many ways you can accomplish this task, we want you to use an if_else() to calculate BMI with the appropriate formula based on each person's location.

```
df_BMI <- df %>%
  mutate(
    msr_sys = if_else(location %in% c("United Kingdom", "Taiwan"), "metric", "imperial"),
     height = if_else(msr_sys == "metric", height/100, height*12),
        BMI = round(if_else(
                msr_sys == "imperial", 703*weight/(height^2),
                weight/(height^2)),1)
  )
```

Question 3

Create a new variable that categorizes BMI with case_when():

- Underweight: BMI below 18.5
- Normal: 18.5-24.9
- Overweight: 25.0-29.9
- Obese: 30.0 and Above

```r
df_BMI <- df_BMI %>%
  mutate(
    BMI_cat = case_when(
      BMI < 18.5 ~ "Underweight",
      between(BMI, 18.5, 24.9) ~ "Normal",
      between(BMI, 25.0, 29.9) ~ "Overweight",
      BMI >= 30.0 ~ "Obese",
      TRUE ~ NA_character_
    )
  )
```

Could we have used if_else()?

Yes, we could have used a series of nested `if_else` statements, and it could have gave the same exact result. However, numerous `if_else` statements can become difficult to read and follow, and can be more prone to coding or syntax errors. When there a number of conditions such as creating categories based on numeric cut-off points, `case_when` is the way to go!

Question 4

Arrange your data first by location and then by descending order of BMI.

```r
df_BMI <- df_BMI %>%
  arrange(location, desc(BMI))

head(df_BMI, 5)
```

```
  location gender height weight  msr_sys  BMI BMI_cat
1 Colorado Female  55.92 350.60 imperial 78.8   Obese
2 Colorado Female  55.08 321.93 imperial 74.6   Obese
3 Colorado   Male  56.64 319.73 imperial 70.1   Obese
4 Colorado Female  59.40 348.39 imperial 69.4   Obese
5 Colorado Female  55.92 302.09 imperial 67.9   Obese
```

Question 5

Use a dplyr method to remove the height, weight, and BMI columns from your data.

```r
df_clean <- df_BMI %>%
  select(-c(height, weight, BMI))

head(df_clean, 5)
```

```
  location gender  msr_sys BMI_cat
1 Colorado Female imperial   Obese
2 Colorado Female imperial   Obese
3 Colorado   Male imperial   Obese
4 Colorado Female imperial   Obese
5 Colorado Female imperial   Obese
```

Optional Challenge

Perform all the actions in this problem set with one dpylr call.

```
df_single <- df_raw %>%
  rename("location" = "Location.information") %>%
  janitor::clean_names() %>%

  mutate(
    msr_sys = if_else(location %in% c("United Kingdom", "Taiwan"), "metric", "imperial"),
    height  = if_else(msr_sys == "metric", height/100, height*12),
    BMI     = round(if_else(msr_sys == "imperial", 703*weight/(height^2), weight/(height^2)),1),

    BMI_cat = case_when(
                    BMI < 18.5 ~ "Underweight",
      between(BMI, 18.5, 24.9) ~ "Normal",
      between(BMI, 25.0, 29.9) ~ "Overweight",
                   BMI >= 30.0 ~ "Obese",
                          TRUE ~ NA_character_
    )
  ) %>%

  arrange(location, desc(BMI)) %>%
  select(-c(height, weight, BMI))


##################################################
# test - did we get the same result in the end? #
##################################################
identical(df_single, df_clean)
```

```
[1] TRUE
```