PHW251 Problem Set 6

Teaching Team

2023

Part 1

For this part we will work with fictional data comparing the efficacy of two interventions. The interventions took place across several states and cities, with slight variations in dates. The outcome is a continuous variable.

Question 1

There's missing data in this data set. Can you identify them? In the next question you will re-code these values to NA.

```
# your code here
head(df) # I see some strange values in gender already
```

```
## # A tibble: 6 x 7
##   date       city        state intervention gender orientation       outcome
##   <chr>      <chr>       <chr>        <dbl> <chr>  <chr>               <dbl>
## 1 25/05/2018 atlanta     GA               1 -999   heterosexual           10
## 2 25/05/2018 Atlanta     gA               1 -999   heterosexual            6
## 3 25/05/2018 atlAnTa     TX               2 female lesbian/gay woman       3
## 4 25/02/2019 San Antonio TX               1 -1     heterosexual            9
## 5 25/02/2019 austin      tX               2 male   heterosexual            1
## 6 25/03/2018 oakland     ca               2 male   heterosexual            2
```

```
# first see how many NAs in the dataset
sum(is.na(df)) # 25 NAs
```

```
## [1] 32
```

```
# take a look at the gender unique values
unique(df$gender) # -999, -1 are most likely NA values
```

```
## [1] "-999"   "female" "-1"     "male"   NA
```

```
# take a look at orientation unique values
unique(df$orientation) # -999, -1 are most likely NA values
```

```
## [1] "heterosexual"      "lesbian/gay woman" "gay"
## [4] "-999"              "-1"                NA
## [7] "other"
```

```r
# we haven't learned this, but you can also use the following code
# to find columns with missing values:
# sapply taks a list, vector, or data frame and outputs a vector or matrix
# we supply the data frame and use the function anyNA()
# all of this occurs when accessing names(df)
# note that this code won't help us find -999, -1
names(df)[sapply(df, anyNA)]
```

```
## [1] "state"        "intervention" "gender"       "orientation"  "outcome"
```

How many NAs did you find?

YOUR ANSWER HERE

- 32 NAs

Are there other values you think may count as NA?

YOUR ANSWER HERE

- -999, -1

Question 2

For the other values you believe may also be NAs, re-code them as NA.

```
# your code here
df <- df %>%
  # notice how we can manipulate multiple variables in one mutate call
  mutate(gender = if_else(gender %in% c("female", "male"), # if female/male
                          gender, # keep same
                          NA_character_), # otherwise, NA
         orientation = if_else(orientation %in% # check for:
                                 c("heterosexual", "lesbian/gay woman", "gay", "other"),
                               orientation, # keep same
                               NA_character_)) # otherwise NA
head(df)
```

```
## # A tibble: 6 x 7
##   date       city        state intervention gender orientation       outcome
##   <chr>      <chr>       <chr>        <dbl> <chr>  <chr>               <dbl>
## 1 25/05/2018 atlanta     GA               1 <NA>   heterosexual           10
## 2 25/05/2018 Atlanta     gA               1 <NA>   heterosexual            6
## 3 25/05/2018 atlAnTa     TX               2 female lesbian/gay woman       3
## 4 25/02/2019 San Antonio TX               1 <NA>   heterosexual            9
## 5 25/02/2019 austin      tX               2 male   heterosexual            1
## 6 25/03/2018 oakland     ca               2 male   heterosexual            2
```

Question 3

Now that we've fixed our NA values, let's address the errors we see with city and state names. Let's fix these entries to have uniform naming where cities are properly capitalized and state abbreviations are in all capital letters. For example, we want to see "San Antonio" and "TX" rather than "san Antonio" and "tx".

Use `distinct()` and `pull()` to see all the variations you need to account for. Then, use `case_when()` to fix the values. We have provided the code to fix the variation for Georgia and Texas using `case_when()`. Expand this code to fix the state abbreviations for California and all the city names.

```
# pull/look at unique city names
df %>%
  select(city) %>%
  distinct() %>%
  pull()
```

```
##  [1] "atlanta"     "Atlanta"     "atlAnTa"     "San Antonio" "austin"
##  [6] "oakland"     "Hayward"     "hayward"     "san Antonio" "iakland"
## [11] "Haywarf"
```

```
# this would accomplish the same thing without dplyr
unique(df$city)
```

```
##  [1] "atlanta"     "Atlanta"     "atlAnTa"     "San Antonio" "austin"
##  [6] "oakland"     "Hayward"     "hayward"     "san Antonio" "iakland"
## [11] "Haywarf"
```

```
# pull/look at unique states
df %>%
  select(state) %>%
  distinct() %>%
  pull()
```

```
##  [1] "GA"  "gA"  "TX"  "tX"  "ca"  "CA"  NA    "ga"  "tx"  "C A" "G A" "CA_"
```

```
# this would accomplish the same thing without dplyr
unique(df$state)
```

```
##  [1] "GA"  "gA"  "TX"  "tX"  "ca"  "CA"  NA    "ga"  "tx"  "C A" "G A" "CA_"
```

```
# fix city and state using case_when()
df <- df %>%
  mutate(
    city = case_when(
      city %in% c("Atlanta", "atlanta", "atlAnTa") ~ "Atlanta",
      city %in% c("Austin", "austin") ~ "Austin",
      city %in% c("San Antonio", "san Antonio") ~ "San Antonio",
      city %in% c("Oakland", "oakland", "iakland") ~ "Oakland",
      city %in% c("Hayward", "hayward", "Haywarf") ~ "Hayward"),
    state = case_when(
      state %in% c("GA", "gA", "ga", "G A") ~ "GA",
```

```
      state %in% c("TX", "tX", "tx") ~ "TX",
      state %in% c("CA", "ca", "C A", "CA_") ~ "CA"))

# the above solution was somewhat laborious; we could have changed all of the
# city names to the same case to reduce some of the coding using
# str_to_upper or str_to_title function https://stringr.tidyverse.org/reference/case.html
library(stringr)
df$city <- str_to_title(df$city) # Only first letter to upper case
unique(df$city)
```

```
## [1] "Atlanta"     "San Antonio" "Austin"      "Oakland"     "Hayward"
```

```
df$state <- str_to_upper(df$state) # All letters to upper case
unique(df$state)
```

```
## [1] "GA" "TX" "CA" NA
```

```
df <- df %>%
  mutate(
    city = case_when(
      city == "Iakland" ~ "Oakland",
      city == "Haywarf" ~ "Hayward",
      TRUE ~ city),
    state = case_when(
      state == "G A" ~ "GA",
      state %in% c("C A", "CA_") ~ "CA",
      TRUE ~ state))
```

Question 4

Format the date column into a date format using a lubridate function. Ominously, these interventions all occurred on the 25th day of the month.

```
# your code here
df$date <- dmy(df$date)

# alternative using mutate()
#df <- df %>% mutate(date = dmy(date))
```

Question 5

You may have noticed that some of the cities don't match their state. We can't, at least from our data, distinguish which value is correct (the city or the state). Let's drop those rows with this inconsistency. The correct city and state pairings are:

- Atlanta, GA
- Austin, TX
- San Antonio, TX
- Hayward, CA
- Oakland, CA

One suggestion is to create a variable indicating whether to drop the row. If you performed this step correctly you should have 33 rows.

```
# your code here
df <- df %>%
  # create drop variable to indicate which rows to drop
  mutate(drop = case_when(
    state == "CA" & city %in% c("Oakland", "Hayward") ~ "keep",
    state == "GA" & city == "Atlanta" ~ "keep",
    state == "TX" & city %in% c("San Antonio", "Austin") ~ "keep",
    TRUE ~ NA_character_)) %>%
  drop_na(drop) # the non-keep cells (NAs) will be dropped
```

Question 6

Another issue: our interventions column has missing data. We have two interventions that occurred in these locations:

- Intervention 1: Hayward, Atlanta, San Antonio
- Intervention 2: Oakland, Atlanta, Austin

For all of the cities except Atlanta it's clear what intervention took place. In these clear instances, replace NAs with the appropriate intervention. As for Atlanta, we are forced to throw out the observations with missing intervention data since we cannot determine which intervention occurred.

```
# your code here
df <- df %>%
  mutate(intervention = case_when(
    city %in% c("Hayward", "San Antonio") ~ 1,
    city %in% c("Oakland", "Austin") ~ 2,
    TRUE ~ intervention)) %>%
  drop_na(intervention)
```

How many observations did you drop?

YOUR ANSWER HERE 2

Question 7

We have a few NAs in the outcome column. Our on-site researchers informed us that when a score of "0" was provided, the data collection team left the cell blank. Re-code the NAs to 0.

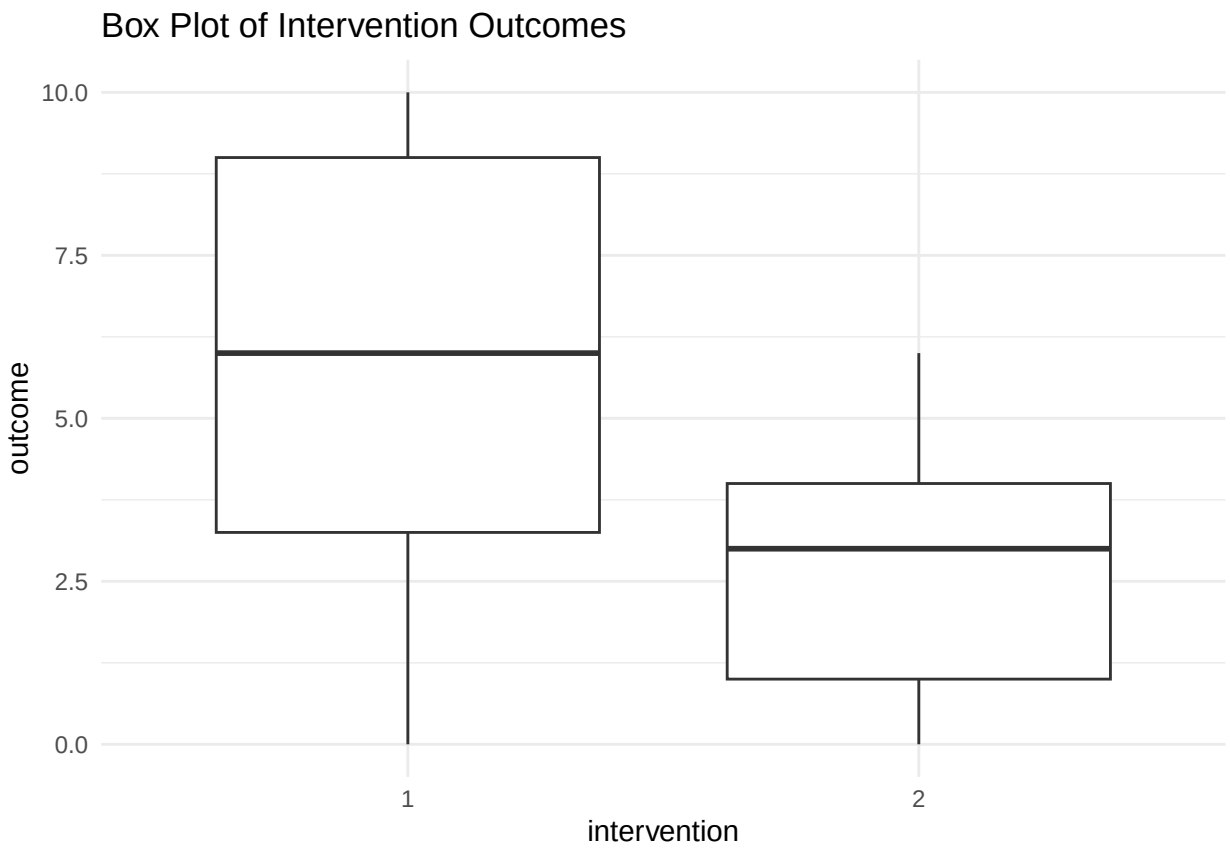
```
# your code here
df <- df %>%
  mutate(outcome = if_else(
    is.na(outcome), # if outcome is NA
    0, # re-code to 0
    outcome # otherwise keep the same value
  ))
```

Question 8

Use ggplot to create a box plot comparing the two interventions and their outcome. The outcome is a continuous variable from 0 to 10. You may need to factor one of your variables. Look at the visualization cheatsheet if you don't know the "geom" for creating a boxplot.

```
# make intervention a factor
df <- df %>% mutate(intervention = as.factor(intervention))

ggplot(df, aes(x=intervention, y=outcome)) +
  geom_boxplot() +
  labs(title = "Box Plot of Intervention Outcomes") +
  theme_minimal()
```



Part 2

For this part we will use *fictional* data inspired by research on non-deceptive or open-label placebos. Non-deceptive placebos are placebos but without the deception. Some studies have found suggestions that, despite not being tricked, participants are reporting similar benefits to what they would have with placebos! You can read more here:

NPR: Is A Placebo A Sham If You Know It's A Fake And It Still Works?

Nature Communications: Placebos without deception reduce self-report and neural measures of emotional distress

In this fictional data we conducted an experiment across two university sites to investigate whether non-deceptive placebos decreased self-report pain ratings. There were three groups: control, placebo, and non-deceptive placebo. Each participant completed a pre- and post- pain induction task and provided a pain rating. All participants completed the same procedures during the pre-test. Only during the post-test did participants in the intervention arms (placebo, non-deceptive) receive additional instructions prior to the pain induction task (i.e., placebo or non-deceptive placebo ratings).

Data coding:

- ID: Contains participant ID number, a letter to indicate group, and pre or post tags.
  C = Control P = Placebo N = Non-deceptive
- LOCATION: Research Site
- PAIN RATING: Self report of pain based on a 0-10 scale
- DATE: Date of observation

Question 9

Read in the data! To make it slightly more challenging we have changed the format from a .csv to .xlsx and "hidden" the data one level deeper in the /data folder. Take a look at the data to get oriented. Please use "placebo_df" as the name of your data frame.

```
# your code here
# read in data with readr
placebo_df <- readxl::read_xlsx("data/one_level_deeper/non_deceptive_placebo.xlsx")
```

Question 10

It's a bit difficult to tell what group (control, placebo, or non-deceptive placebo) each participant is in with their IDs combined with their grouping. Create a new column called "GROUP" based on the letter assignment for IDs. The stringr function 'str_detect()' will be useful here!

```
# your code here
# Method 1
# grab index for each group
index_c <- str_detect(placebo_df$ID, "^C")
index_p <- str_detect(placebo_df$ID, "^P")
index_n <- str_detect(placebo_df$ID, "^N")

# use index for each group to assign group name
placebo_df$GROUP[index_c] <- "Control"
```

```
## Warning: Unknown or uninitialised column: `GROUP`.
```

```
placebo_df$GROUP[index_p] <- "Placebo"
placebo_df$GROUP[index_n] <- "Non-deceptive"

# Method 2
# case_when, and you can also try if_else
placebo_df$GROUP <- case_when(index_c == TRUE ~ "Control",
                              index_p == TRUE ~ "Placebo",
                              index_n == TRUE ~ "Non-deceptive")
# can also add GROUP column with mutate function
placebo_df <- placebo_df %>%
  mutate(GROUP = case_when(index_c == TRUE ~ "Control",
                           index_p == TRUE ~ "Placebo",
                           index_n == TRUE ~ "Non-deceptive"))

head(placebo_df)
```

```
## # A tibble: 6 x 5
##   ID       LOCATION DATE                PAIN_RATE GROUP
##   <chr>    <chr>    <chr>                   <dbl> <chr>
## 1 C101_pre UCLA     January 31st, 2018          8 Control
## 2 P102_pre UCLA     February 25th, 2018         7 Placebo
## 3 N103_pre UCLA     January 17th, 2018          7 Non-deceptive
## 4 C104_pre UCLA     January 31st, 2018          8 Control
## 5 P105_pre UCLA     February 25th, 2018         6 Placebo
## 6 N106_pre UCLA     January 17th, 2018          8 Non-deceptive
```

Question 11

We have a similar issue telling apart the pre- and post- observations. Create a new column called "TEST" that distinguishes whether the observation is a pre- or post-test.

Unfortunately, the two research sites were not consistent in their naming convention. You will need to consider the different cases.

```
# your code here
# change all of ID case to uppercase to standardize
placebo_df$ID <- str_to_upper(placebo_df$ID)

# Method 1
# grab index
index_pre <- str_detect(placebo_df$ID, "PRE$")
index_post <- str_detect(placebo_df$ID, "POST$")

# use index to place correct test instance
placebo_df$TEST[index_pre] <- "Pre"
```

```
## Warning: Unknown or uninitialised column: `TEST`.
```

```
placebo_df$TEST[index_post] <- "Post"

# Method 2
# case_when, and you can also try if_else
placebo_df <- placebo_df %>%
  mutate(TEST = case_when(index_pre == TRUE ~ "Pre",
                          index_post == TRUE ~ "Post"))

head(placebo_df)
```

```
## # A tibble: 6 x 6
##   ID       LOCATION DATE                PAIN_RATE GROUP         TEST
##   <chr>    <chr>    <chr>                   <dbl> <chr>         <chr>
## 1 C101_PRE UCLA     January 31st, 2018          8 Control       Pre
## 2 P102_PRE UCLA     February 25th, 2018         7 Placebo       Pre
## 3 N103_PRE UCLA     January 17th, 2018          7 Non-deceptive Pre
## 4 C104_PRE UCLA     January 31st, 2018          8 Control       Pre
## 5 P105_PRE UCLA     February 25th, 2018         6 Placebo       Pre
## 6 N106_PRE UCLA     January 17th, 2018          8 Non-deceptive Pre
```

Question 12

There were differences in the formatting for dates across the two research sites. Create a new column called "DATE_FIX" that grabs only the date. Make sure this new date column takes the following format: yyyy-mm-dd

Hint: Check out ?parse_date_time

```
# your code here
placebo_df$DATE_FIX <- parse_date_time(placebo_df$DATE, c("mdy", "dmy"))

# may also use mutate to add a column
placebo_df <- placebo_df %>%
  mutate(DATE_FIX = parse_date_time(DATE, c("mdy", "dmy")))

head(placebo_df)
```

```
## # A tibble: 6 x 7
##   ID       LOCATION DATE               PAIN_RATE GROUP TEST  DATE_FIX
##   <chr>    <chr>    <chr>                  <dbl> <chr> <chr> <dttm>
## 1 C101_PRE UCLA     January 31st, 2018         8 Cont~ Pre   2018-01-31 00:00:00
## 2 P102_PRE UCLA     February 25th, 20~         7 Plac~ Pre   2018-02-25 00:00:00
## 3 N103_PRE UCLA     January 17th, 2018         7 Non-~ Pre   2018-01-17 00:00:00
## 4 C104_PRE UCLA     January 31st, 2018         8 Cont~ Pre   2018-01-31 00:00:00
## 5 P105_PRE UCLA     February 25th, 20~         6 Plac~ Pre   2018-02-25 00:00:00
## 6 N106_PRE UCLA     January 17th, 2018         8 Non-~ Pre   2018-01-17 00:00:00
```

Question 13

You realize there was a strange error in your excel file that, for every date, pushed the date forward by 1 year. Rather than editing your excel sheet and potentially making an incorrect permanent change to your raw data you decide to fix the error in R. Create a new column called "DATE_FIX_2" that fixes the date.

```
# your code here
placebo_df$DATE_FIX2 <- placebo_df$DATE_FIX %m-% years(1)
```

Question 14

Let's clean up our data frame by removing DATE and DATE_FIX. Afterwards, rename DATE_FIX2 to DATE.

```
# your code here
# base R method
placebo_df <- subset(placebo_df, select = -c(DATE, DATE_FIX))
names(placebo_df)[6] <- "DATE"

# tidyverse method
#placebo_df <- placebo_df %>%
#  select(-DATE, -DATE_FIX) %>%
#  rename(DATE = DATE_FIX2)

head(placebo_df)
```

```
## # A tibble: 6 x 6
##   ID       LOCATION PAIN_RATE GROUP         TEST  DATE
##   <chr>    <chr>        <dbl> <chr>         <chr> <dttm>
## 1 C101_PRE UCLA             8 Control       Pre   2017-01-31 00:00:00
## 2 P102_PRE UCLA             7 Placebo       Pre   2017-02-25 00:00:00
## 3 N103_PRE UCLA             7 Non-deceptive Pre   2017-01-17 00:00:00
## 4 C104_PRE UCLA             8 Control       Pre   2017-01-31 00:00:00
## 5 P105_PRE UCLA             6 Placebo       Pre   2017-02-25 00:00:00
## 6 N106_PRE UCLA             8 Non-deceptive Pre   2017-01-17 00:00:00
```

Question 15

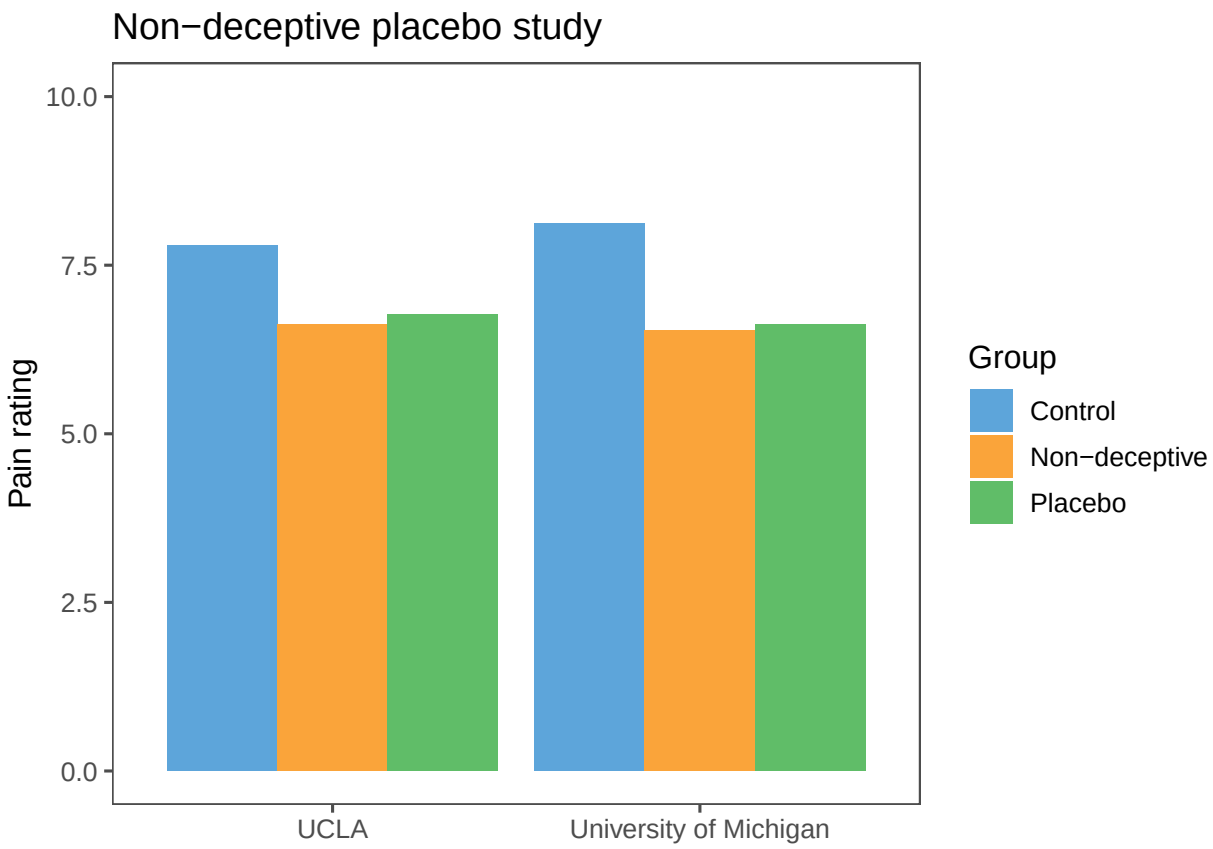
We're interested in plotting our data to begin digging into the results. Below is dplyr and ggplot code to do this. Uncomment and run the following code as-is (visualization is not the focus of this problem set). You may need to install ggthemes.

```
# install.packages("ggthemes")
library(ggthemes)

df_plot <- placebo_df %>%
  group_by(GROUP, LOCATION) %>%
  summarize(MEAN_PAIN = mean(PAIN_RATE))
```

```
## `summarise()` has grouped output by 'GROUP'. You can override using the
## `.groups` argument.
```

```
ggplot(df_plot, aes(x = LOCATION, y = MEAN_PAIN, fill = GROUP)) +
  geom_col(position = "dodge") +
  ylim(0, 10) +
  theme_few() +
  scale_fill_few("Medium") +
  theme(axis.title = element_blank(),
        axis.title.y = element_text()) +
  labs(fill = "Group",
       title = "Non-deceptive placebo study",
       y = "Pain rating")
```



For a quick first pass we think this visualization isn't so bad. However, logically, we think that the order of the groups should be: Control, Placebo, Non-deceptive. Make GROUP into a factor that reflects this order. If done correctly, when you re-run the above chunk, the plot should show the bars in that order

```
# your code here
placebo_df$GROUP <- factor(placebo_df$GROUP,
                           levels = c("Control", "Placebo", "Non-deceptive"),
                           ordered = TRUE)
```

You're done! Please knit to pdf and upload to gradescope.