PHW251 Problem Set 6

AUTHOR
Val Stacey

PUBLISHED
November 3, 2025

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.

```
na_sum <- df %>%
  summarise(across(everything(), ~sum(is.na(.)))) %>%
  pivot_longer(
    everything(),
    names_to = "variable",
    values_to = "na_count"
  ) %>%
  janitor::adorn_totals()
```

1A. How many NAs did you find?

- **32** total "NAs"

1B. Are there other values you think may count as NA?

- Yes. I believe that anything coded as **"-999"** is very likely a code for "Unknown" or "NA." Some are also coded as **"-1"** , which likely also indicates an unknown or NA field.

Column Name	NA Count
date	0
city	0
state	6
intervention	11
gender	4
orientation	4
outcome	7
Total	**32**

Question 2

2A. For the other values you believe may also be NAs, re-code them as NA.

```
df <- df %>%
  mutate(
    across(
      where(is.character),
      ~ replace(., . %in% c("-999", "-1"), NA_character_)
      )
    )
```

2B. Print the head() of the dataframe

```
head(df, 5)
```

```
# A tibble: 5 × 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
```

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".

3A. Use `distinct()` and `pull()` to see all the variations you need to account for.

```
states <- df %>%
  distinct(state) %>%
  arrange(state) %>%
  pull(state)

cities <- df %>%
  distinct(city) %>%
  arrange(city) %>%
  pull(city)
```

States:

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

Cities:

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

3B. 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.

```
df <- df %>%
mutate(
  state = case_when(
      state %in% c("TX", "tX", "tx")         ~ "TX",
      state %in% c("GA", "gA", "ga", "G A")  ~ "GA",
      state %in% c("C A", "CA", "CA_", "ca") ~ "CA",
      TRUE ~ NA_character_
    ),
  city = case_when(
      city %in% c("austin")                       ~ "Austin",
      city %in% c("iakland", "oakland")           ~ "Oakland",
      city %in% c("San Antonio", "san Antonio")   ~ "San Antonio",
      city %in% c("Atlanta", "atlAnTa", "atlanta") ~ "Atlanta",
      city %in% c("Hayward", "Haywarf", "hayward") ~ "Hayward",
      TRUE ~ NA_character_
    )
  )
```

Question 4

4A. Format the date column into a date format using a lubridate function.

Ominously, these interventions all occurred on the 25th day of the month.

```
df <- df %>% mutate(date = as.Date(date, format = "%d/%m/%Y"))
df %>% select(state, city, date) %>% slice_head(n=5)
```

```
# A tibble: 5 × 3
  state city        date
  <chr> <chr>       <date>
1 GA    Atlanta     2018-05-25
2 GA    Atlanta     2018-05-25
3 TX    Atlanta     2018-05-25
4 TX    San Antonio 2019-02-25
5 TX    Austin      2019-02-25
```

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). The correct city and state pairings are:

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

5A. Drop the rows with this city/state inconsistency.

```
df <- df %>%
  mutate(
    drop_row = case_when(
      state == "GA" &  city == "Atlanta" ~ "keep",
      state == "TX" &  city %in% c("Austin", "San Antonio") ~ "keep",
      state == "CA" &  city %in% c("Hayward", "Oakland") ~ "keep",
      TRUE ~ "drop"
    )
  ) %>%
  filter(drop_row == "keep") %>%
  select(-drop_row)

nrow(df)
```

```
[1] 33
```

5B. Print the unique combinations of city and state that are now in the data frame

Use the code below and modify if needed.

```
unique(df[,c("city", "state")])
```

```
# A tibble: 5 × 2
  city        state
  <chr>       <chr>
1 Atlanta     GA
2 San Antonio TX
3 Austin      TX
4 Oakland     CA
5 Hayward     CA
```

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.

6A. In these clear instances, replace NAs with the appropriate intervention.

6B. For Atlanta, drop the observations with missing intervention data since we cannot determine which intervention occurred.

```
df <- df %>%
#--6B.
filter(!(city == "Atlanta" & is.na(intervention))) %>%
#--6A.
mutate(
  intervention =
      as.integer(
        case_when(
          city %in% c("Hayward", "San Antonio") ~ 1,
          city %in% c("Oakland", "Austin") ~ 2,
          TRUE ~ intervention
        )
      )
    )
```

6C. How many observations did you drop?

- **2** observations were dropped

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.

7A. Re-code the NAs to 0.

```
df <- df %>% mutate(outcome = if_else(is.na(outcome), 0, outcome))
```

7B. Use code to confirm that there are no longer any NAs in the outcome column.

```
sum(is.na(df$outcome))
```

```
[1] 0
```

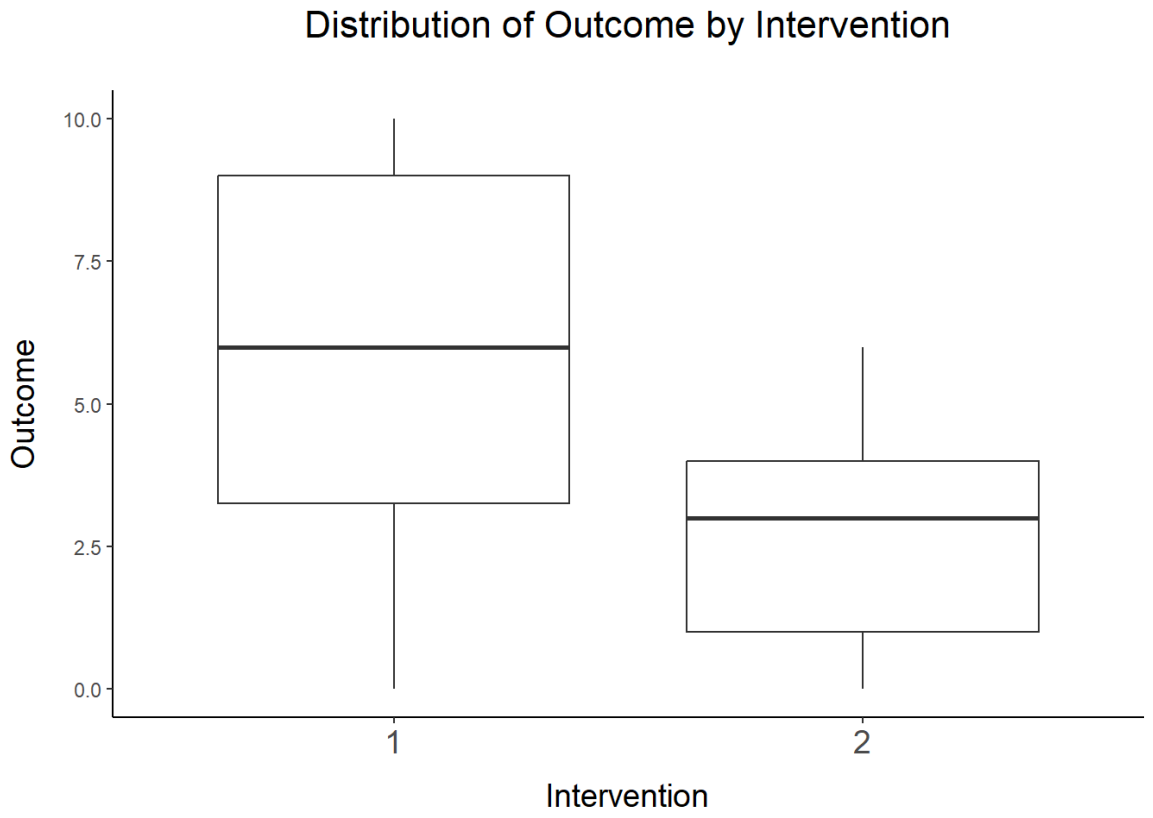
Question 8

8A 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.

```
df <- df %>%
  mutate(intervention = factor(intervention, levels = c(1,2), ordered = TRUE))
```

```
ggplot(df) +
  geom_boxplot(aes(x = intervention, y = outcome)) +
  ylab("Outcome") +
  xlab("Intervention")+
  ggtitle("Distribution of Outcome by Intervention") +
  theme_classic() +
  theme(
    legend.position = "none",
    plot.title      = element_text(size = 16, hjust = 0.5, margin =
         margin(b=20)),
    axis.title.y    = element_text(size = 14, margin = margin(r = 10)),
    axis.title.x    = element_text(size = 14, margin = margin(t = 10)),
    axis.text.x     =  element_text(size = 14)
  )
```


Distribution of Outcome by Intervention
Outcome
10.0
7.5
5.0
2.5
0.0
1
2
Intervention

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

9A 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.

```
placebo_df <- 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.

10A. Create a new column called "GROUP" based on the letter assignment for IDs.

The stringr function 'str_detect()' will be useful here!

```
placebo_df <- placebo_df %>%
  mutate(
    GROUP =
        case_when(
          str_starts(ID, regex("c", ignore_case = TRUE)) ~ "Control",
          str_starts(ID, regex("p", ignore_case = TRUE)) ~ "Placebo",
          str_starts(ID, regex("n", ignore_case = TRUE)) ~ "Non-Deceptive",
          TRUE ~ NA_character_
        )
      )
```

10B. Print the head() of the dataframe

```
placebo_df %>% group_by(GROUP) %>% slice_head(n=2)
```

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

Question 11

We have a similar issue telling apart the pre- and post- observations.

11A. 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.

```
placebo_df <- placebo_df %>%
  mutate(
    TEST =
      case_when(
        str_detect(ID, regex("pre", ignore_case = TRUE)) ~ "Pre",
        str_detect(ID, regex("post", ignore_case = TRUE)) ~ "Post",
        TRUE ~ NA_character_
      )
    )
```

11B. Print the head() of the dataframe

```
placebo_df %>% group_by(GROUP, TEST) %>% slice_head(n=1)
```

```
# A tibble: 6 × 6
# Groups:   GROUP, TEST [6]
  ID        LOCATION DATE                PAIN_RATE GROUP         TEST
  <chr>     <chr>    <chr>                   <dbl> <chr>         <chr>
1 C101_post UCLA     July 1st, 2018              9 Control       Post
2 C101_pre  UCLA     January 31st, 2018          8 Control       Pre
3 N103_post UCLA     June 16th, 2018             8 Non-Deceptive Post
4 N103_pre  UCLA     January 17th, 2018          7 Non-Deceptive Pre
5 P102_post UCLA     July 2nd, 2018              5 Placebo       Post
6 P102_pre  UCLA     February 25th, 2018         7 Placebo       Pre
```

Question 12

There were differences in the formatting for dates across the two research sites.

12A. 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

```
placebo_df <- placebo_df %>%
  mutate(
    DATE_FIX =
      as.Date(parse_date_time(DATE, c("%b%d%Y", "%d%m%y")), format = "%Y-%m-%d")
    )
```

12B. Print the head() of the dataframe

```
placebo_df %>% group_by(LOCATION) %>% slice_head(n=3)
```

```
# A tibble: 6 × 7
# Groups:   LOCATION [2]
  ID       LOCATION               DATE          PAIN_RATE GROUP TEST  DATE_FIX
  <chr>    <chr>                  <chr>             <dbl> <chr> <chr> <date>
1 C101_pre UCLA                   January 31st…         8 Cont… Pre   2018-01-31
2 P102_pre UCLA                   February 25t…         7 Plac… Pre   2018-02-25
3 N103_pre UCLA                   January 17th…         7 Non-… Pre   2018-01-17
4 N118_PRE University of Michigan 09-Jan-18             6 Non-… Pre   2018-01-09
5 C119_PRE University of Michigan 21-Jan-18             8 Cont… Pre   2018-01-21
6 P120_PRE University of Michigan 22-Mar-18             7 Plac… Pre   2018-03-22
```

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.

13A. Create a new column called "DATE_FIX_2" that fixes the date.

```
placebo_df <- placebo_df %>%
  mutate(
    DATE_FIX_2 = as.Date(DATE_FIX - years(x=1)),
    wrng_yr = year(DATE_FIX)
    )

placebo_df %>%
  select(LOCATION, DATE, wrng_yr, DATE_FIX, DATE_FIX_2) %>%
  group_by(LOCATION, wrng_yr) %>%
  slice_head(n=1)
```

```
# A tibble: 4 × 5
# Groups:   LOCATION, wrng_yr [4]
  LOCATION               DATE               wrng_yr DATE_FIX   DATE_FIX_2
  <chr>                  <chr>                <dbl> <date>     <date>
1 UCLA                   January 31st, 2018    2018 2018-01-31 2017-01-31
2 UCLA                   January 15th, 2019    2019 2019-01-15 2018-01-15
3 University of Michigan 09-Jan-18             2018 2018-01-09 2017-01-09
4 University of Michigan 25-Mar-19             2019 2019-03-25 2018-03-25
```

Question 14

14A. Clean up the data frame by removing DATE and DATE_FIX.

14B. Afterwards, rename DATE_FIX2 to DATE

```
placebo_df <- placebo_df %>%
  #--14A.
  select(-c(DATE, DATE_FIX, wrng_yr)) %>%
  #--14B.
  rename("DATE" = "DATE_FIX_2") %>%
  relocate(DATE, .before = PAIN_RATE)
```

14C. Print the head() of the dataframe

```
head(placebo_df, 5)
```

```
# A tibble: 5 × 6
  ID       LOCATION DATE       PAIN_RATE GROUP         TEST
  <chr>    <chr>    <date>         <dbl> <chr>         <chr>
1 C101_pre UCLA     2017-01-31         8 Control       Pre
2 P102_pre UCLA     2017-02-25         7 Placebo       Pre
3 N103_pre UCLA     2017-01-17         7 Non-Deceptive Pre
4 C104_pre UCLA     2017-01-31         8 Control       Pre
5 P105_pre UCLA     2017-02-25         6 Placebo       Pre
```

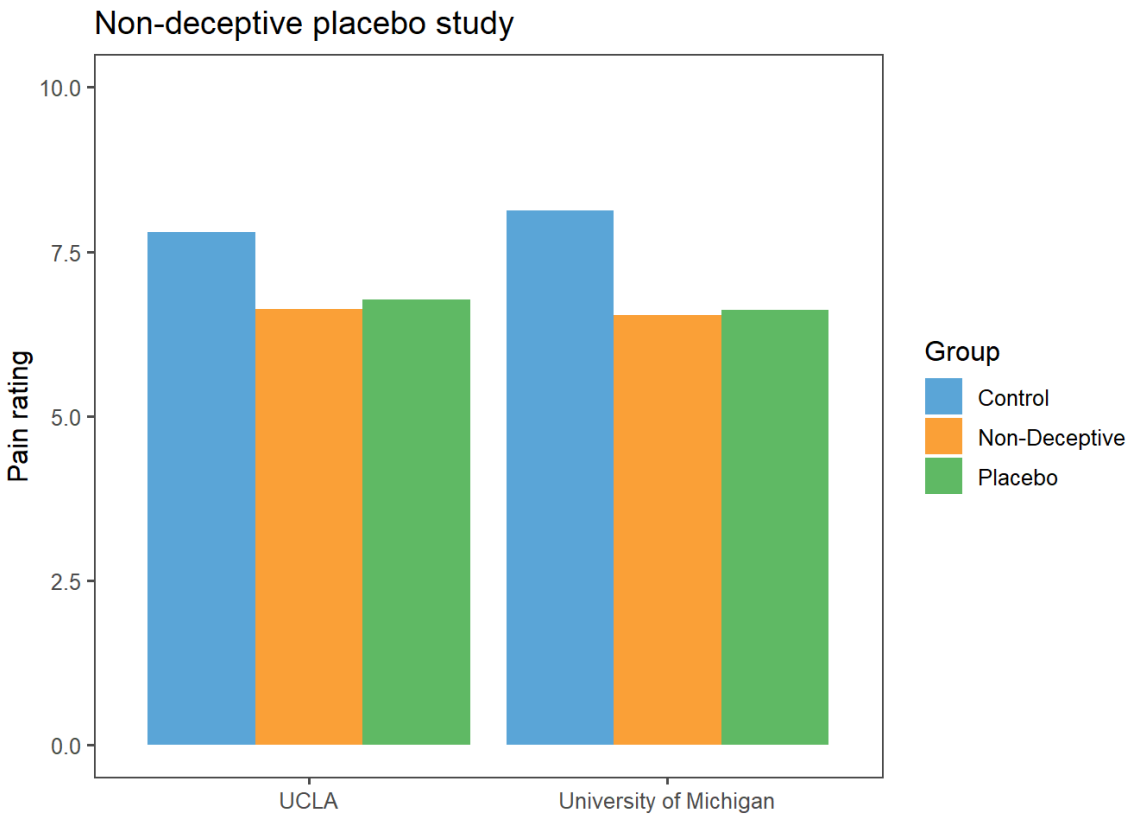
Question 15

We're interested in plotting our data to begin digging into the results. Below is dplyr and ggplot code to do this.

15A. Uncomment and run the following code as-is (visualization is not the focus of this problem set).

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

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.

15B. 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

```
placebo_df <- placebo_df %>%
  mutate(
    GROUP =
        factor(
          GROUP,
          levels = c("Control", "Placebo", "Non-Deceptive"),
          ordered = TRUE
          )
        )
```