Data Challenge 01

Jake Moore

Objective

Practice the data science workflow (question -> acquire -> tidy -> explore/visualize -> communicate) with small, beginner-friendly tasks in R. Knit to PDF in an RStudio Project, use base R only (no tidyverse), style with snake_case, spaces, and <-. Un-comment and fill in/complete code as needed to answer each question or prompt.

Questions

1. Workflow paragraph - 5 pts

Write **4-6 sentences** about a real situation you care about (school, work, hobby). For each of the five steps, say what you might do: **question** (what you want to find out), **acquire** (where the data comes from), **tidy** (how you clean/organize it), **explore/visualize** (quick checks or simple plots to see patterns), and **communicate** (how you would share the result). Write your sentences as normal text below.

> Your Answer: My question is to see how much my caffeine intake affects my sleep time and next day energy levels. To do this I will create a caffeine intake log, where I can acquire mg of caffeine consumed, when it was consumed, sleep duration, and energy score for the next morning. I will tidy this data by creating different catagories like sleep duration, mg consumed, mg consumed after 2pm, etc. I will then explore this data by visualizing it through scatterplots, looking for correlation between the variables. Then I'll communicate my results through a PDF after collecting the appropriate amount of data to find a true correlation. I will also run hypothesis tests and correlation analysis.

2. Markdown & formatting practice - 10 pts

Practice basic Markdown (the plain-text formatting used in R Markdown), run one code expression, and answer some questions.

For each of the following phrases, do as the phrase instructs:

make me italic

make me bold

make me Heading 1

make me Heading 2

make me Heading 3

Run the code chunk below and answer the following questions.

- What does the `**` operator do?

  Your Answer: ** Raises the number to a power (exponents)

** Explain what setting `echo = FALSE` in the chunk header does.

  Your Answer: When setting echo = FALSE, it makes it so the R code is not shown in the output or the knitted document. In this example, it ensures that the R code doesn't show up in the PDF. However, values can be shown in plots, and tables.

```
## [1] 6
```

3. Vectors & Data Frame - 25 pts

Make a tiny table from three **vectors** (a vector is a one-dimensional list of values of the same type). Use **NA** for a missing value, and a **logical** vector (TRUE/FALSE).

- Create `name_vector` (character), `score_vector` (numeric; include at least one NA, one <85, and one >85 score), and `passed_vector` (logical; TRUE/FALSE). Treat NA scores as **FALSE** when deciding `passed`, otherwise >= 60 is passing. (16 pts)
- Combine into `df <- data.frame(..., stringsAsFactors = FALSE)` and preview with `head(df)` or `print(df)`. (5 pts)
- Explain what `name`, `score`, and `passed` does in the data.frame() function. (4 pts)

  Your Answer:

```
# YOUR CODE HERE:
name_vector <- c("Jake", "Pepsi", "Laura", "James", "Ron", "Spencer")
score_vector <- c(NA, 70, 95, 90, 86, 77)
passed_vector <- c(FALSE, FALSE, TRUE, TRUE, TRUE, FALSE)
df <- data.frame(name = name_vector,
                 score = score_vector,
                 passed = passed_vector,
                 stringsAsFactors = FALSE)

head(df)
```

```
##      name score passed
## 1    Jake    NA  FALSE
## 2   Pepsi    70  FALSE
## 3   Laura    95   TRUE
## 4   James    90   TRUE
## 5     Ron    86   TRUE
## 6 Spencer    77  FALSE
```

4. Slicing (subsetting rows/columns) - 20 pts

To **slice/subset** means "pick certain rows and/or columns." Use base R indexing.

- Part A (10 pts): Select rows 2 to 5 and only the columns `name` and `score`.
- Part B (10 pts): Keep only rows where `score >= 85`. Ignore NA (drop rows where score is NA). Show just `name` and `score`.

```
# YOUR CODE HERE:
part_a <- df[2:5, c("name", "score")]
print(part_a)
```

```
##    name score
## 2 Pepsi    70
## 3 Laura    95
## 4 James    90
## 5   Ron    86
```

```
score_85_condition <- df$score >= 85
part_b <- df[score_85_condition, c("name","score")]
print(part_b)
```

```
##     name score
## NA  <NA>    NA
## 3  Laura    95
## 4  James    90
## 5    Ron    86
```

5. New Variables - 25 pts

Add two new columns to `df`.

- z_score (15 pts): A **z-score** tells how many standard deviations a value is from the mean. Formula: `z_score = (score - mean(score, na.rm = TRUE)) / sd(score, na.rm = TRUE)`
- grade_band (10 pts): Keep NA if score is NA. Use these cutoffs:

  A if `score >= 90`; B if `80 <= score && score <= 89`; C if `70 <= score && score <= 79`; D if `60 <= score && score <= 69`; F if `score < 60`.

```
# YOUR CODE HERE:
df$z_score <- (df$score-mean(df$score, na.rm = TRUE)) / sd(df$score, na.rm = TRUE)

df$grade_band <- ifelse(
   is.na(df$score), NA,
   ifelse(df$score >= 90, "A",
   ifelse(df$score >= 80 & df$score <= 89, "B",
   ifelse(df$score >= 70 & df$score <= 79, "C",
   ifelse(df$score >= 60 & df$score <= 69, "D", "F")))))
head(df)
```

```
##      name score passed    z_score grade_band
## 1    Jake    NA  FALSE         NA       <NA>
## 2   Pepsi    70  FALSE -1.3512453          C
## 3   Laura    95   TRUE  1.1326615          A
## 4   James    90   TRUE  0.6358801          A
## 5     Ron    86   TRUE  0.2384550          B
## 6 Spencer    77  FALSE -0.6557514          C
```

6. Descriptive Stats - 15 pts

Identify and fill in the functions to compute the following simple **descriptive statistics** of `df$score` (summary numbers that describe a variable): the mean, standard deviation, minimum, and maximum. Use `na.rm = TRUE` so NAs are ignored when computing these values. Print each result clearly.

```
## Mean: 83.6
```

```
## SD: 10.06479
```

```
## Min: 70
```

```
## Max: 95
```