Challenge-1

Merkayla Wong

2023-08-16

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

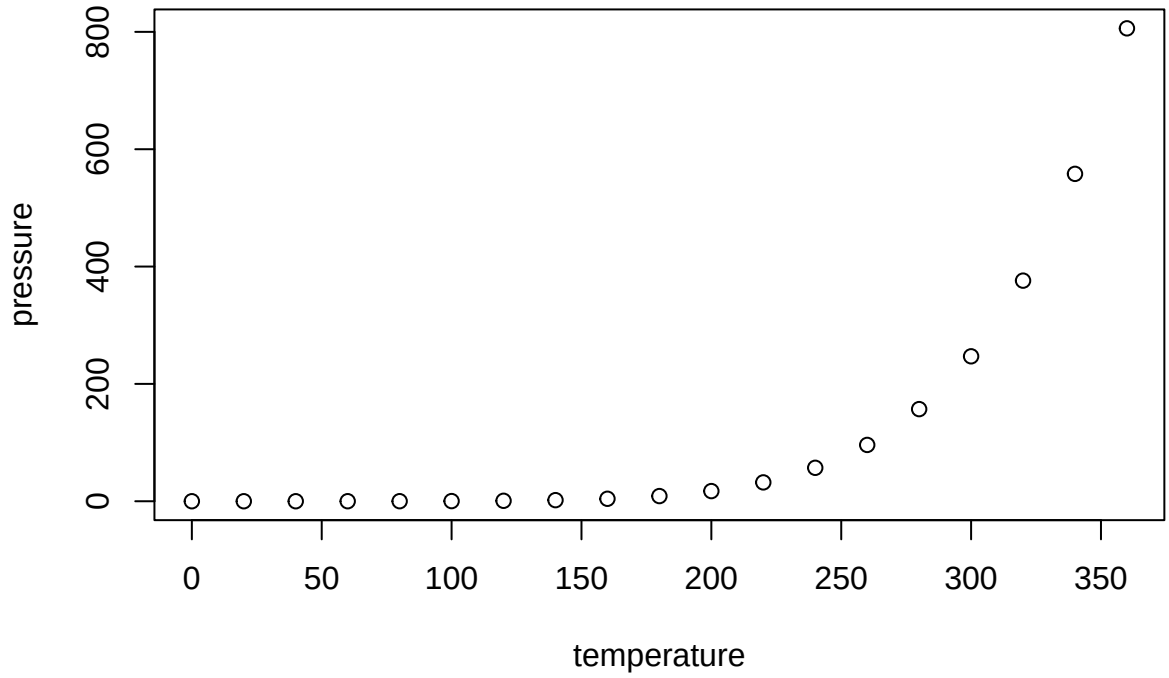
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Hello!I am Merkayla. A year 4 student from Industrial design who hopes to minor in Interactive Media Development.:)

Figure 1: This is my photo

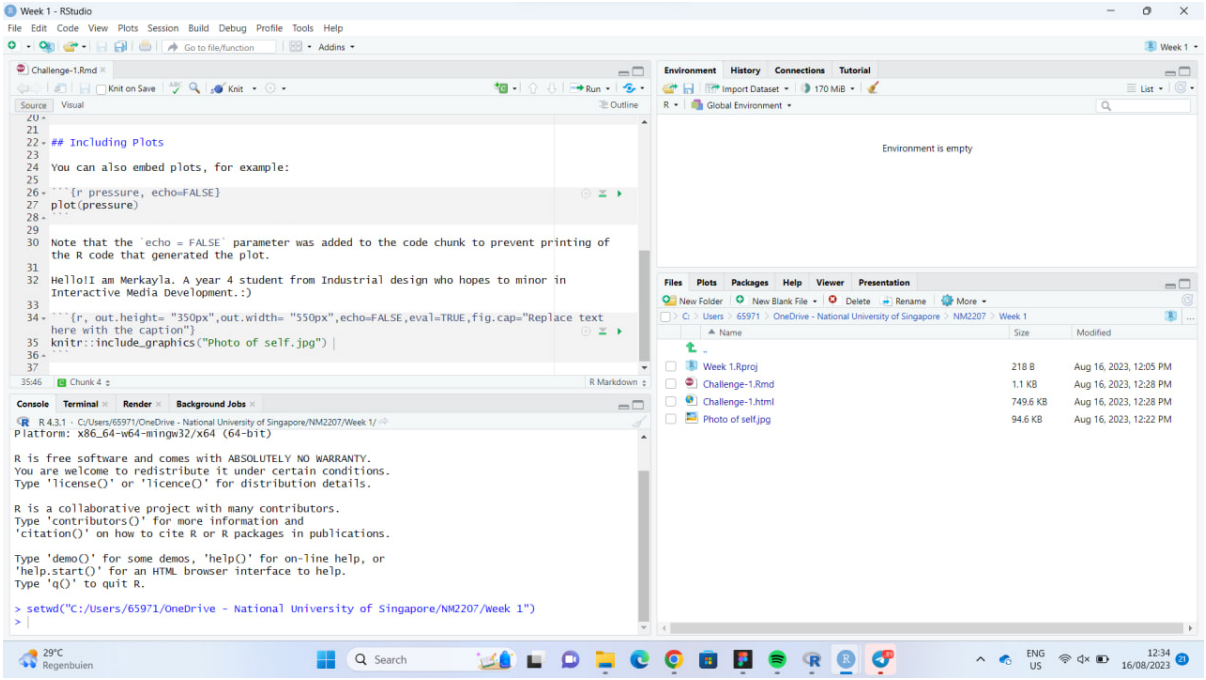


Figure 2: This is my screenshot

Challenge-2

Merkayla

23-08-2023

Welcome! Hope you have watched the lecture videos and followed the instructions in code-along. Go through the steps described below, *carefully*. It is totally fine to get stuck - **ASK FOR HELP**; reach out to your friends, TAs, or the discussion forum on Canvas.

Here is what you have to do,

1. **Pair** with a neighbor and work
2. **Download** the `Challenge-2.Rmd` and `playlist_data.csv` files from Canvas
3. **Move** the downloaded files to the folder, “Week-2”
4. **Set** it as the working directory
5. **Edit** content wherever indicated
6. **Remember** to set `eval=TRUE` after completing the code to generate the output
7. **Ensure** that `echo=TRUE` so that the code is rendered in the final document
8. **Inform** the tutor/instructor upon completion
9. **Submit** the document on Canvas after they approve
10. **Attendance** will be marked only after submission
11. Once again, **do not hesitate** to reach out to the tutors/instructor, if you are stuck

I. Exploring music preferences

A. Background

Imagine that you have been hired as a data analyst by a radio station to analyze music preferences of their DJs. They have provided you with a dataset, `playlist_data.csv` , containing information about DJs, their preferred music genres, song titles, and ratings.

Using the data-set you are required to complete some tasks that are listed subsequently. All these tasks are based on the concepts taught in the video lectures. The questions may not be entirely covered in the lectures; To complete them, you are encouraged to use Google and the resources therein.

B.Tasks

Task-1

In the lecture, we used two data-sets, `starwars` and `anscombe's quartet` that were readily available with the packages, `tidyverse` and `Tmisc` , respectively. When we have to use custom-made data-sets or the ones like we downloaded from Canvas, we have to import it using the R commands before using them. All the questions below are related to this task.

Question 1.1: What does the term “CSV” in `playlist_data.csv` stand for, and why is it a popular format for storing tabular data?

Solution: CSV refers to Comma-Separated Values.It is popular for storing tabular data as it is known for its compatibility, for example being able to be opened by most spreadsheet or data analysis softwares and various programming languages.It is also compact, which makes them efficient for storing and transferring large amounts of data. Additionally, it is a simple and lightweight format tgat makes it easy to read and understand.

Question 1.2: load the `tidyverse` package to work with `.csv` files in R.
>> **Solution:**

```
# Load the necessary package to work with CSV files in R.
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.2     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.3     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to be
come errors
```

Question 1.3: Import the data-set, `playlist_data.csv`

Solution:

```
# Import the "playlist_data.csv" dataset into R

read_csv("playlist_data.csv")
```

```
## Rows: 26 Columns: 7
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (4): DJ_Name, Music_Genre, Experience, Location
## dbl (3): Rating, Age, Plays_Per_Week
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## # A tibble: 26 × 7
##    DJ_Name Music_Genre Rating Experience     Age Location Plays_Per_Week
##    <chr>   <chr>        <dbl> <chr>        <dbl> <chr>              <dbl>
##  1 DJ A    Pop            4.2 Advanced        28 City X                80
##  2 DJ B    Rock           3.8 Intermediate    24 City Y                60
##  3 DJ C    Electronic     4.5 Advanced        30 City Z               100
##  4 DJ D    Pop            4   Intermediate    22 City X                70
##  5 DJ E    Electronic     4.8 Advanced        27 City Y                90
##  6 DJ F    Rock           3.6 Intermediate    25 City Z                55
##  7 DJ G    Pop            4.3 Advanced        29 City X                85
##  8 DJ H    Electronic     4.1 Intermediate    23 City Y                75
##  9 DJ I    Rock           3.9 Advanced        31 City Z                70
## 10 DJ J    Pop            4.4 Intermediate    26 City X                95
## # ℹ 16 more rows
```

Question 1.4: Assign the data-set to a variable, `playlist_data`

Solution:

```
# Assign the variable to a dataset

playlist_data <- read_csv("playlist_data.csv")
```

```
## Rows: 26 Columns: 7
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (4): DJ_Name, Music_Genre, Experience, Location
## dbl (3): Rating, Age, Plays_Per_Week
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

From now on, you can use the name of the variable to view the contents of the data-set

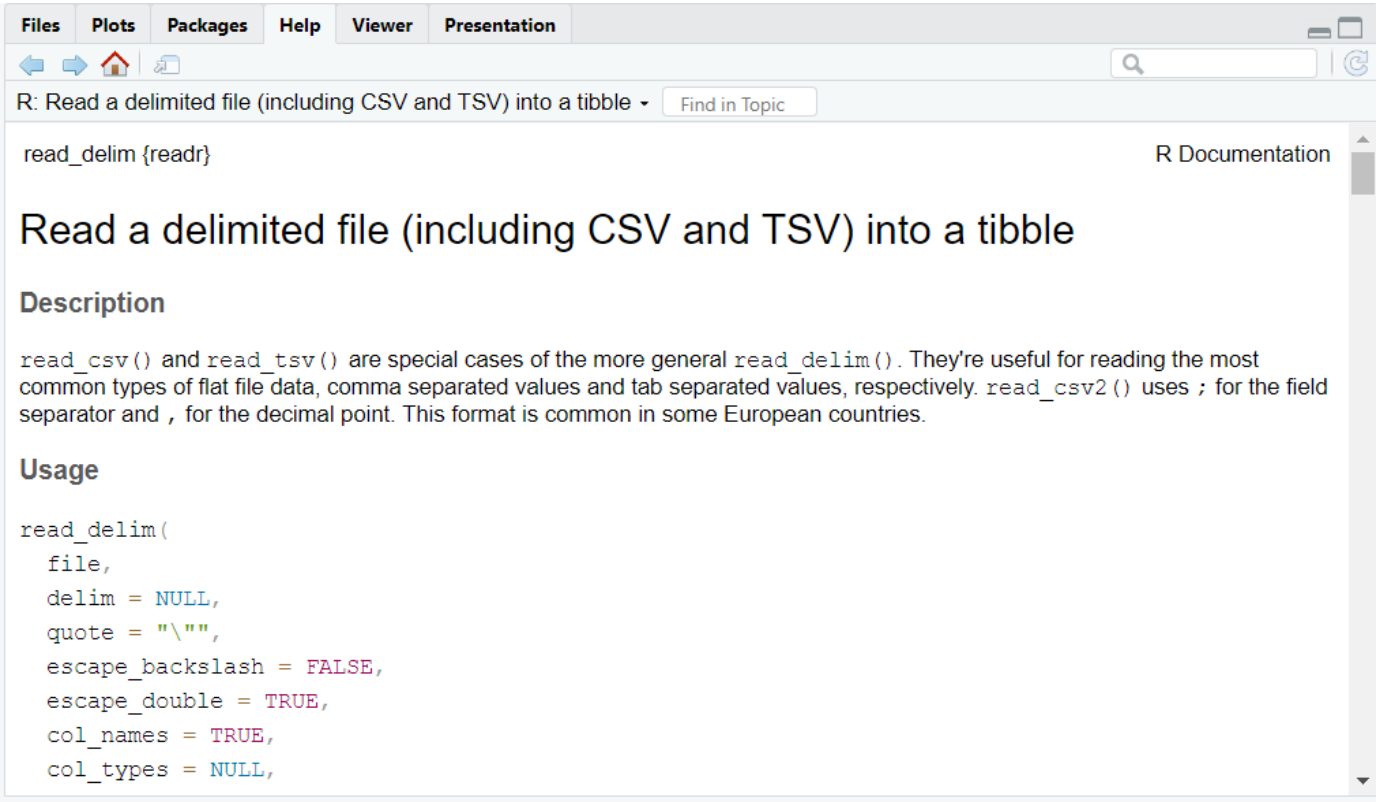
Question 1.5: Get more information about `read_csv()` command and provide a screenshot of the information displayed in the “Help” tab of the “Files” pane

Solution:

```
# More information about the R command, complete the code

?read_csv()
```

```
## starting httpd help server ... done
```

```
knitr::include_graphics("screenshot of output.png")
```



SCREENSHOT

Question 1.6: What does the `skip` argument in the read_csv() function do?

Solution: The skip argument allows me to skip the specified amount of rows. The header argument lets me specify whether the first row should be used as a header row.

Question 1.7: Display the contents of the data-set

Solution:

```
# Type the name of the variable, to see what it contains
playlist_data
```

```
## # A tibble: 26 × 7
##    DJ_Name Music_Genre Rating Experience     Age Location Plays_Per_Week
##    <chr>   <chr>        <dbl> <chr>        <dbl> <chr>             <dbl>
##  1 DJ A    Pop            4.2 Advanced        28 City X               80
##  2 DJ B    Rock           3.8 Intermediate    24 City Y               60
##  3 DJ C    Electronic     4.5 Advanced        30 City Z              100
##  4 DJ D    Pop            4   Intermediate    22 City X               70
##  5 DJ E    Electronic     4.8 Advanced        27 City Y               90
##  6 DJ F    Rock           3.6 Intermediate    25 City Z               55
##  7 DJ G    Pop            4.3 Advanced        29 City X               85
##  8 DJ H    Electronic     4.1 Intermediate    23 City Y               75
##  9 DJ I    Rock           3.9 Advanced        31 City Z               70
## 10 DJ J    Pop            4.4 Intermediate    26 City X               95
## # ℹ 16 more rows
```

Question 1.8: Assume you have a CSV file named `sales_data.csv` containing information about sales transactions. How would you use the `read_csv()` function to import this file into R and store it in a variable named `sales_data` ?

Solution:

```
# No output is required for this code
# Only the list of commands that execute the task mentioned in the question are required
library(tidyverse)
read_csv("sales_data.csv")
Sales_data <- read_csv("sales_data.csv")
```

Task-2

After learning to import a data-set, let us explore the contents of the data-set through the following questions

Question 2.1: Display the first few rows of the data-set to get an overview of its structure

Solution:

```
# Type the name of the variable we assigned the data-set to
head(playlist_data)
```

```
## # A tibble: 6 × 7
##   DJ_Name Music_Genre Rating Experience     Age Location Plays_Per_Week
##   <chr>   <chr>        <dbl> <chr>        <dbl> <chr>              <dbl>
## 1 DJ A    Pop            4.2 Advanced        28 City X                80
## 2 DJ B    Rock           3.8 Intermediate    24 City Y                60
## 3 DJ C    Electronic     4.5 Advanced        30 City Z               100
## 4 DJ D    Pop            4   Intermediate    22 City X                70
## 5 DJ E    Electronic     4.8 Advanced        27 City Y                90
## 6 DJ F    Rock           3.6 Intermediate    25 City Z                55
```

Question 2.2: Display all the columns of the variable stacked one below another

Solution:

```
# Stack columns of playlist_data
glimpse(playlist_data)
```

```
## Rows: 26
## Columns: 7
## $ DJ_Name        <chr> "DJ A", "DJ B", "DJ C", "DJ D", "DJ E", "DJ F", "DJ G",…
## $ Music_Genre    <chr> "Pop", "Rock", "Electronic", "Pop", "Electronic", "Rock…
## $ Rating         <dbl> 4.2, 3.8, 4.5, 4.0, 4.8, 3.6, 4.3, 4.1, 3.9, 4.4, 4.6, …
## $ Experience     <chr> "Advanced", "Intermediate", "Advanced", "Intermediate",…
## $ Age            <dbl> 28, 24, 30, 22, 27, 25, 29, 23, 31, 26, 32, 28, 29, 25,…
## $ Location       <chr> "City X", "City Y", "City Z", "City X", "City Y", "City…
## $ Plays_Per_Week <dbl> 80, 60, 100, 70, 90, 55, 85, 75, 70, 95, 110, 75, 60, 8…
```

Question 2.3: How many columns are there in the dataset?

Solution:

```
# Number of columns
ncol(playlist_data)
```

```
## [1] 7
```

Question 2.4: What is the total count of DJs?

Solution:

```
# Number of DJs
nrow(playlist_data)
```

```
## [1] 26
```

Question 2.5: Display all the location of all the DJs

Solution:

```
# Location of DJs
playlist_data$Location
```

```
##  [1] "City X" "City Y" "City Z" "City X" "City Y" "City Z" "City X" "City Y"
##  [9] "City Z" "City X" "City Y" "City Z" "City X" "City Y" "City Z" "City X"
## [17] "City Y" "City Z" "City X" "City Y" "City Z" "City X" "City Y" "City Z"
## [25] "City X" "City Y"
```

Question 2.6: Display the age of the DJs

Solution:

```
# Age of DJs
playlist_data$Age
```

```
##  [1] 28 24 30 22 27 25 29 23 31 26 32 28 29 25 31 26 27 24 29 23 28 24 30 22 27
## [26] 25
```

Task-3

Let us plot the data to get more insights about the DJs.

Question 3.1: Create a plot to visualize the relationship between DJs' ages and their ratings.

Solution:

```
# complete the code to generate the plot

ggplot(playlist_data)
```



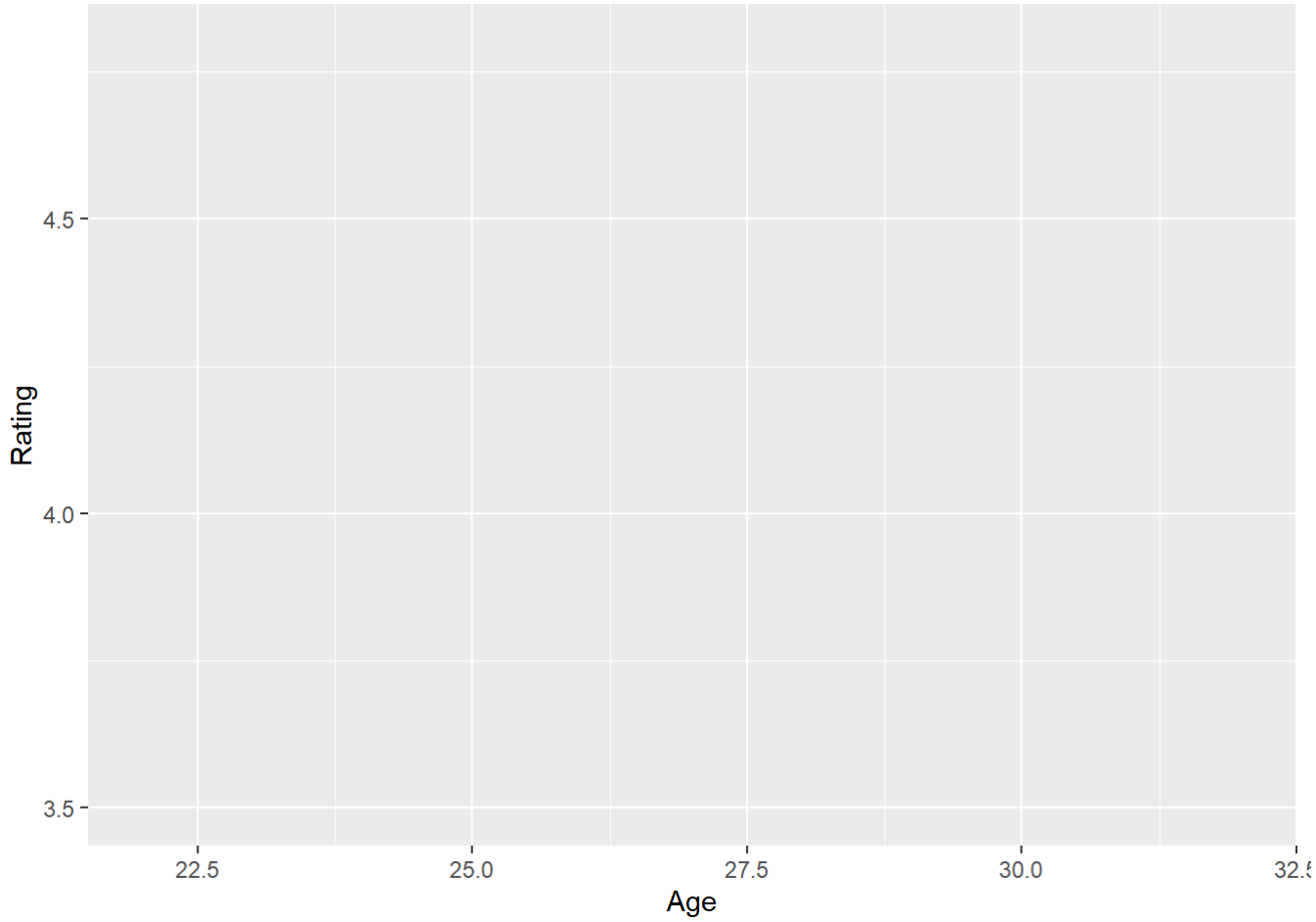
```
aes(x=column_name,y=column_name)
```

```
## Aesthetic mapping:
## * `x` -> `column_name`
## * `y` -> `column_name`
```

Question 3.2: Label the x-axis as "Age" and the y-axis as "Rating."

Solution:

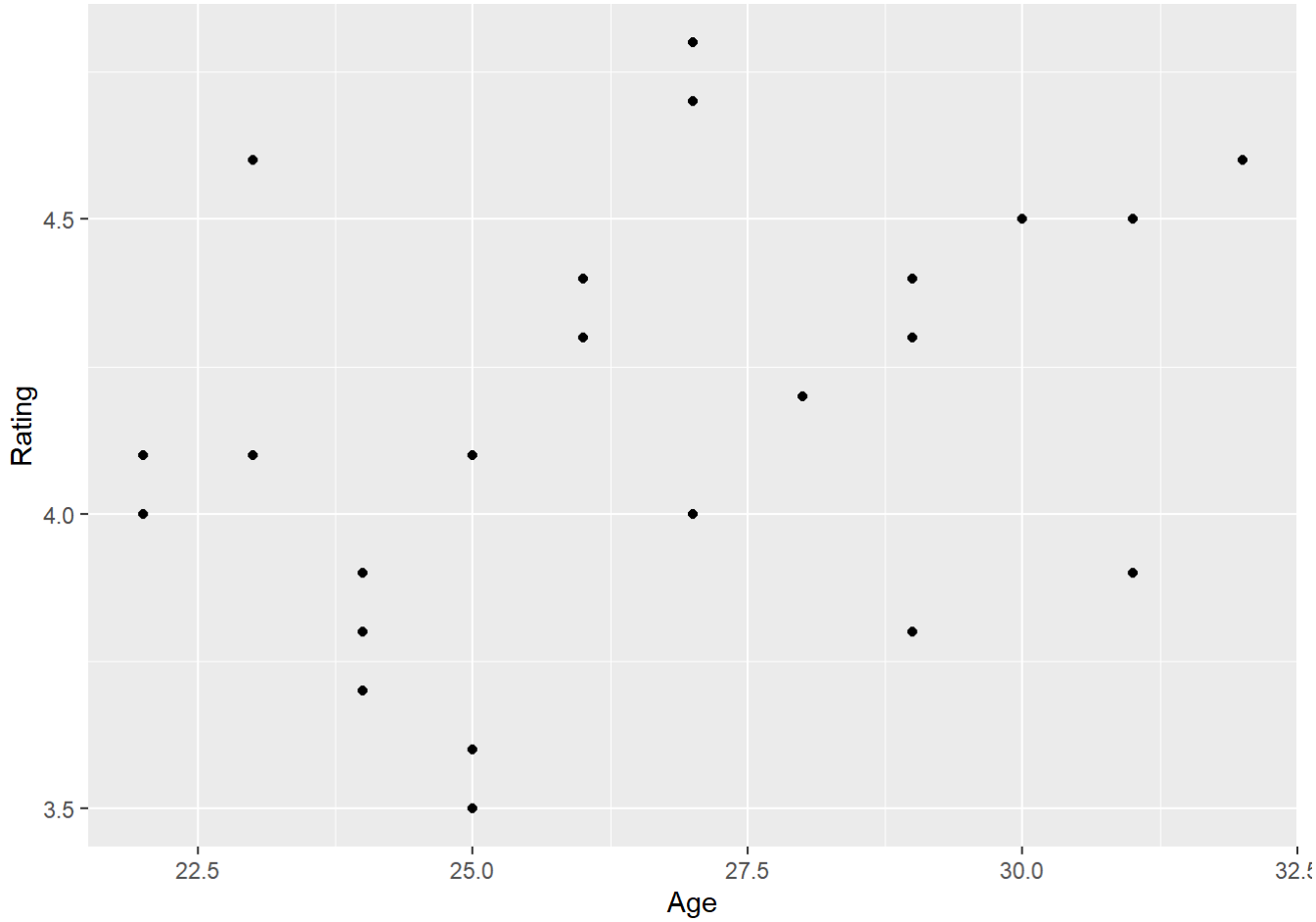
```
# complete the code to generate the plot

ggplot(data=playlist_data,mapping=aes(x=Age,y=Rating))
```



Question 3.3: Represent data using points

Solution:

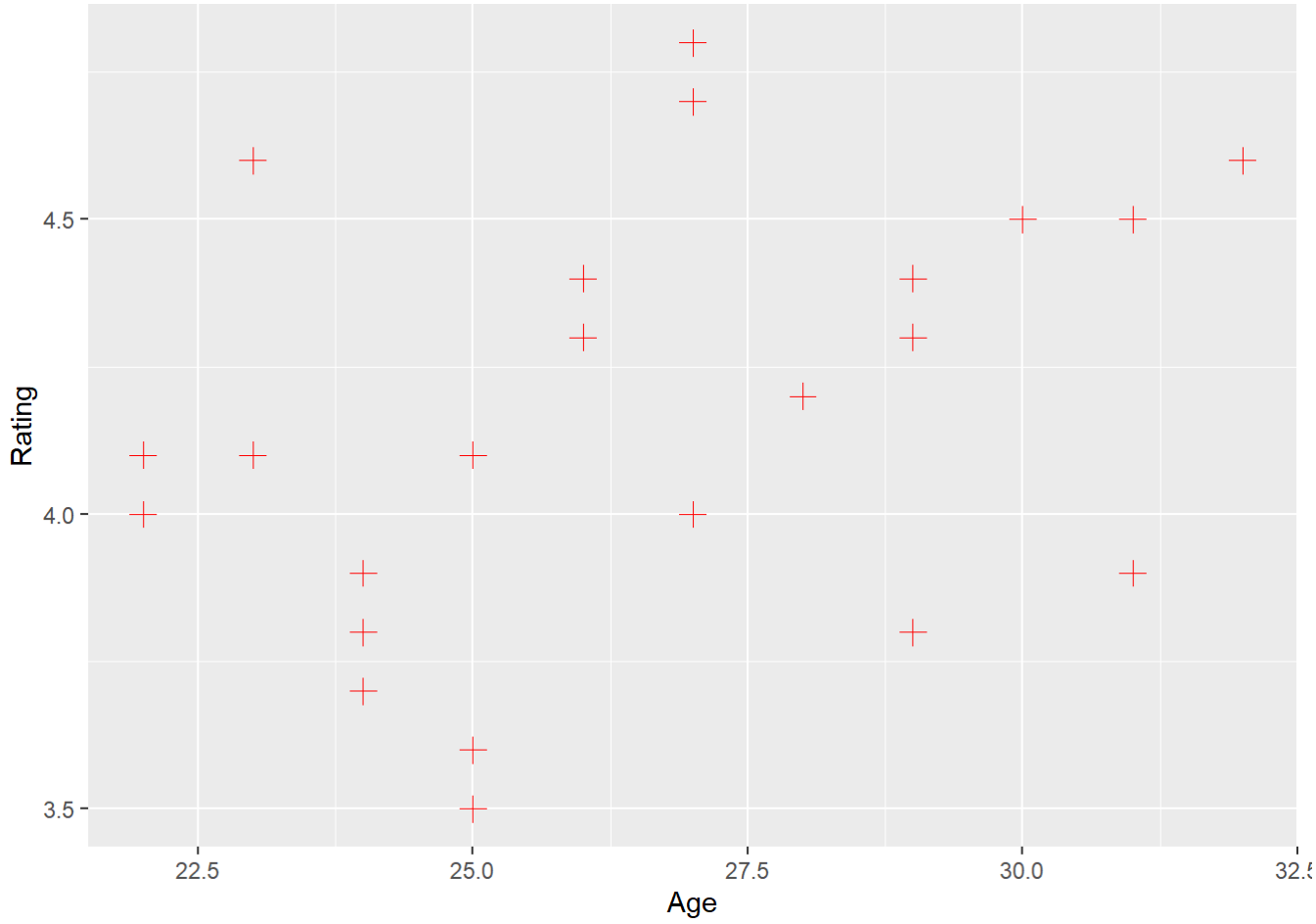
```
# complete the code to generate the plot

ggplot(data=playlist_data,mapping=aes(x=Age,y=Rating)) +
  geom_point()
```



Question 3.4: Can you change the points represented by dots/small circles to any other shape of your liking?

Solution:

```
# complete the code to generate the plot
ggplot(data=playlist_data,mapping=aes(x=Age,y=Rating)) +
  geom_point(colour= "red",size=3,shape=3)
```

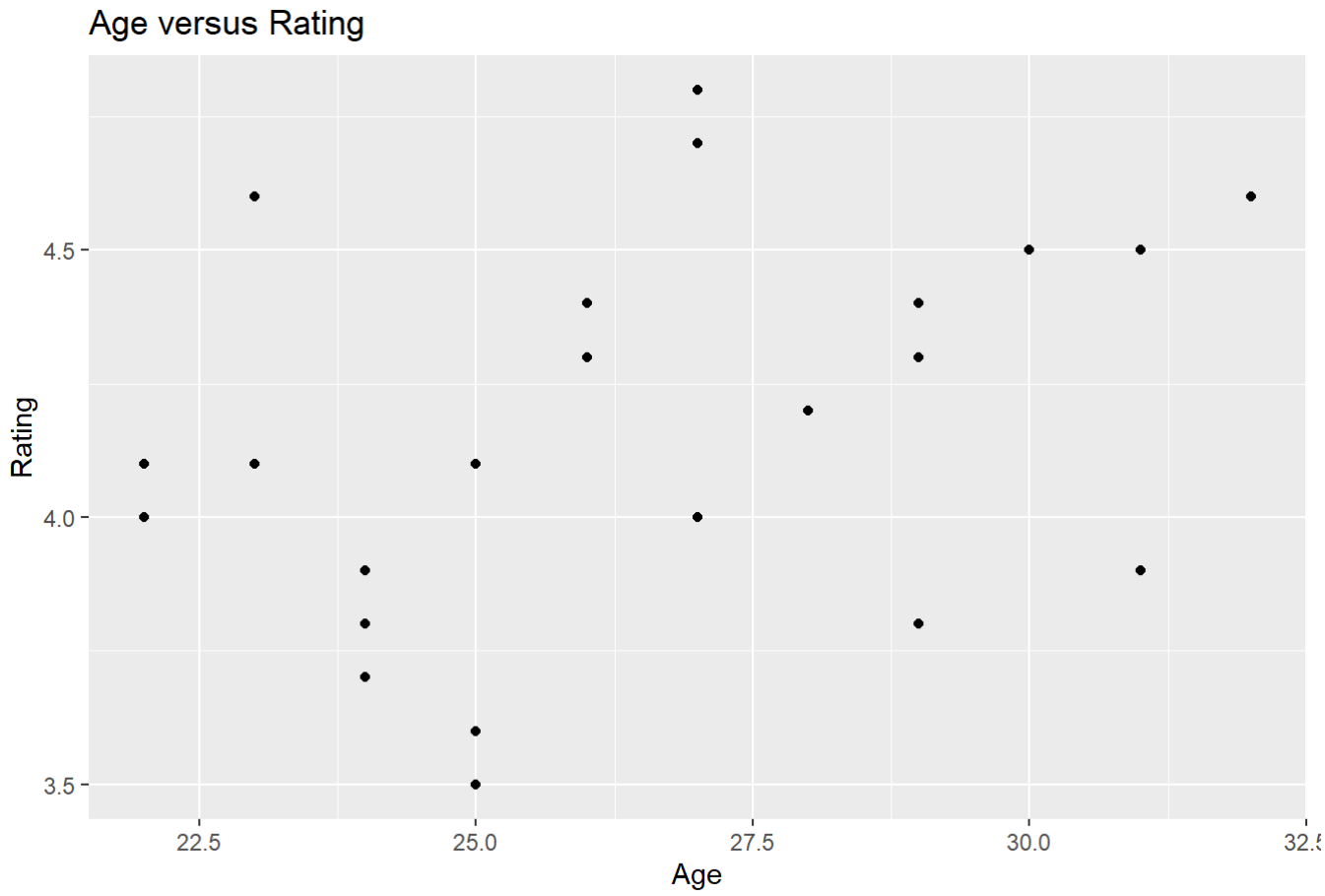



```
# <-- Hint: Use ? to learn more about geom_point and use appropriate values for shape
```

Question 3.5: Insert a suitable title and briefly provide your insights in the caption

Solution:

```
# complete the code to generate the plot

ggplot(data=playlist_data,mapping=aes(x=Age,y=Rating)) +
  geom_point() +
  labs(x="Age",y="Rating",
title="Age versus Rating",
caption="Source: tidyverse/ playlist dataset")
```


Age versus Rating
Rating
4.5
4.0
3.5
22.5
25.0
27.5
30.0
Age
Source: tidyverse/ playlist dataset

Challenge-3

Merkayla

`30-08-2023

I. Questions

Question 1: Emoji Expressions

Imagine you're analyzing social media posts for sentiment analysis. If you were to create a variable named "postSentiment" to store the sentiment of a post using emojis (😄 for positive, 😐 for neutral, 😢 for negative), what data type would you assign to this variable? Why? (*narrative type question, no code required*)

Solution: *I would use a string (character) as it is categoric where I am able to assign meaningful names that are related to the data and ordinal.*

Question 2: Hashtag Havoc

In a study on trending hashtags, you want to store the list of hashtags associated with a post. What data type would you choose for the variable "postHashtags"? How might this data type help you analyze and categorize the hashtags later? (*narrative type question, no code required*)

Solution: *character. It can be used to retrieve messages later or allow us to count the entries"*

Question 3: Time Traveler's Log

You're examining the timing of user interactions on a website. Would you use a numeric or non-numeric data type to represent the timestamp of each interaction? Explain your choice (*narrative type question, no code required*)

Solution: *I would use characters a numeric data type, as the timestamp include only discrete numbers.*

Question 4: Event Elegance

You're managing an event database that includes the date and time of each session. What data type(s) would you use to represent the session date and time? (*narrative type question, no code required*)

Solution: *I would use a characters, non numeric data type as there are not only numerical values in date and time*

Question 5: Nominee Nominations

You're analyzing nominations for an online award. Each participant can nominate multiple candidates. What data type would be suitable for storing the list of nominated candidates for each participant? (*narrative type question, no code required*)

Solution: *I would use the character data type.*

Question 6: Communication Channels

In a survey about preferred communication channels, respondents choose from options like "email," "phone," or "social media." What data type would you assign to the variable "preferredChannel"? (*narrative type question, no code required*)

Solution: *character*

Question 7: Colorful Commentary

In a design feedback survey, participants are asked to describe their feelings about a website using color names (e.g., “warm red,” “cool blue”). What data type would you choose for the variable “feedbackColor”? (*narrative type question, no code required*)

Solution: *character*

Question 8: Variable Exploration

Imagine you’re conducting a study on social media usage. Identify three variables related to this study, and specify their data types in R. Classify each variable as either numeric or non-numeric.

Solution: *the PreferredPlatform is non-numeric, the AmountOfTimeOnPlatform is numeric, and the mostUsedHashtag is non-numeric.*

Question 9: Vector Variety

Create a numeric vector named “ages” containing the ages of five people: 25, 30, 22, 28, and 33. Print the vector.

Solution:

```
# Enter code here
age<-c(25L,30L,22L,28L,33L)
print(age)
```

```
## [1] 25 30 22 28 33
```

Question 10: List Logic

Construct a list named “student_info” that contains the following elements:

- A character vector of student names: “Alice,” “Bob,” “Catherine”
- A numeric vector of their respective scores: 85, 92, 78
- A logical vector indicating if they passed the exam: TRUE, TRUE, FALSE

Print the list.

Solution:

```
# Enter code here
student_info = list(name=c("Alice","Bob","Catherine"), score=c(85,92,78), passedexams=c(TRUE,
TRUE,FALSE))
print(student_info)
```

```
## $name
## [1] "Alice"     "Bob"       "Catherine"
##
## $score
## [1] 85 92 78
##
## $passedexams
## [1]  TRUE  TRUE FALSE
```

Question 11: Type Tracking

You have a vector “data” containing the values 10, 15.5, “20”, and TRUE. Determine the data types of each element using the typeof() function.

Solution:

```
# Enter code here
a<-c(10)
typeof(a)
```

```
## [1] "double"
```

```
b<-c(15.5)
typeof(b)
```

```
## [1] "double"
```

```
c<-c("20")
typeof(c)
```

```
## [1] "character"
```

```
d<-c(TRUE)
typeof(d)
```

```
## [1] "logical"
```

Question 12: Coercion Chronicles

You have a numeric vector “prices” with values 20.5, 15, and “25”. Use explicit coercion to convert the last element to a numeric data type. Print the updated vector.

Solution:

```
# Enter code here
x<-as.numeric("25")
typeof(x)
```

```
## [1] "double"
```

```
prices<-c(20.5,15,x)
print(prices)
```

```
## [1] 20.5 15.0 25.0
```

Question 13: Implicit Intuition

Combine the numeric vector c(5, 10, 15) with the character vector c(“apple”, “banana”, “cherry”). What happens to the data types of the combined vector? Explain the concept of implicit coercion.

Solution: *They become a vector of character.Implicit coercion is the automatic conversion of the data type when an operation involving different data types is performed.*

```
# Enter code here
x<-c(5,10,15)
y<-c("apple","banana","cherry",x)
typeof(y)
```

```
## [1] "character"
```

Question 14: Coercion Challenges

You have a vector “numbers” with values 7, 12.5, and “15.7”. Calculate the sum of these numbers. Will R automatically handle the data type conversion? If not, how would you handle it?

Solution: no R will not automatically handle the data type conversion, it will return NA.

```
# Enter code here
x<-c(7,12.5)
y<-as.double("15.7")
numbers<-c(x,y)
sum(numbers)
```

```
## [1] 35.2
```

Question 15: Coercion Consequences

Suppose you want to calculate the average of a vector “grades” with values 85, 90.5, and “75.2”. If you directly calculate the mean using the mean() function, what result do you expect? How might you ensure accurate calculation?

Solution: It would return NA because R cannot perform arithmetic function on non numeric functions.I can ensure an accurate calculation by explicitly converting the vector to a numeric format and handling potential NA values.

```
# Enter code here
y<-c(85,90.5)
x<-as.numeric("75.2")
newvector<-c(y,x)
mean(newvector)
```

```
## [1] 83.56667
```

Question 16: Data Diversity in Lists

Create a list named “mixed_data” with the following components:

- A numeric vector: 10, 20, 30
- A character vector: “red”, “green”, “blue”
- A logical vector: TRUE, FALSE, TRUE

Calculate the mean of the numeric vector within the list.

Solution:

```
# Enter code here
mixed_data=list(numb=c(10,20,30),char=c("red","green", "blue"),logi=c(TRUE,FALSE,TRUE))
mixed_data$numb
```

```
## [1] 10 20 30
```

```
mean(mixed_data$numb)
```

```
## [1] 20
```

Question 17: List Logic Follow-up

Using the “student_info” list from Question 10, extract and print the score of the student named “Bob.”

Solution:

```
# Enter code here
student_info$score[student_info$name=="Bob"]
```

```
## [1] 92
```

Question 18: Dynamic Access

Create a numeric vector values with random values. Write R code to dynamically access and print the last element of the vector, regardless of its length.

Solution:

```
# Enter code here
x<-c(3,4,5)
x[length(x)]
```

```
## [1] 5
```

Question 19: Multiple Matches

You have a character vector words <- c(“apple”, “banana”, “cherry”, “apple”). Write R code to find and print the indices of all occurrences of the word “apple.”

Solution:

```
# Enter code here
words<-c("apple", "banana", "cherry", "apple")
words=="apple"
```

```
## [1]  TRUE FALSE FALSE  TRUE
```

```
which(words=="apple")
```

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

Question 20: Conditional Capture

Assume you have a vector ages containing the ages of individuals. Write R code to extract and print the ages of individuals who are older than 30.

Solution:

```
# Enter code here
x<-c(30,45,21,46)
x[x>30]
```

```
## [1] 45 46
```

Question 21: Extract Every Nth

Given a numeric vector sequence <- 1:20, write R code to extract and print every third element of the vector.

Solution:

```
# Enter code here
x<-c(1:20)
print(x[seq(1, 20, 3)])
```

```
## [1]  1  4  7 10 13 16 19
```

Question 22: Range Retrieval

Create a numeric vector numbers with values from 1 to 10. Write R code to extract and print the values between the fourth and eighth elements.

Solution:

```
# Enter code here
x<-c(1:10)
print(x[seq(4, 8)])
```

```
## [1] 4 5 6 7 8
```

Question 23: Missing Matters

Suppose you have a numeric vector data <- c(10, NA, 15, 20). Write R code to check if the second element of the vector is missing (NA).

Solution:

```
# Enter code here
x<- c(10, NA, 15, 20)
print(x[2])
```

```
## [1] NA
```

Question 24: Temperature Extremes

Assume you have a numeric vector temperatures with daily temperatures. Create a logical vector hot_days that flags days with temperatures above 90 degrees Fahrenheit. Print the total number of hot days.

Solution:

```
# Enter code here
temp<-c(90,100,81,102)
x<-temp>90
sum(x)
```

```
## [1] 2
```

Question 25: String Selection

Given a character vector fruits containing fruit names, create a logical vector long_names that identifies fruits with names longer than 6 characters. Print the long fruit names.

Solution:

```
# Enter code here
x<-c("banana","pineapple","apple")
long_names<-x[nchar(x)>6]
print(long_names)
```

```
## [1] "pineapple"
```

Question 26: Data Divisibility

Given a numeric vector numbers, create a logical vector divisible_by_5 to indicate numbers that are divisible by 5. Print the numbers that satisfy this condition.

Solution:

```
# Enter code here
y<-c(5,6,7,10)
y[y%%5==0]
```

```
## [1]  5 10
```

Question 27: Bigger or Smaller?

You have two numeric vectors vector1 and vector2. Create a logical vector comparison to indicate whether each element in vector1 is greater than the corresponding element in vector2. Print the comparison results.

Solution:

```
# Enter code here
x<-c(12,14,15,17)
y<-c(14,15,14,18)
x>y
```

```
## [1] FALSE FALSE  TRUE FALSE
```

Challenge-4

Merkayla Wong

6-9-2023

Questions

Load the “CommQuest2023.csv” dataset using the `read_csv()` command and assign it to a variable named “comm_data.”

```
# Enter code here
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.2     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.3     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to be
come errors
```

```
# Read data from the csv file
comm_data. <- read_csv("CommQuest2023_Larger.csv")
```

```
## Rows: 1000 Columns: 5
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (3): channel, sender, message
## dbl  (1): sentiment
## date (1): date
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Question-1: Communication Chronicles

Using the select command, create a new dataframe containing only the “date,” “channel,” and “message” columns from the “comm_data” dataset.

Solution:

```
# Enter code here
select(comm_data., date,channel,message)
```

```
## # A tibble: 1,000 × 3
##    date       channel message
##    <date>     <chr>   <chr>
##  1 2023-08-11 Twitter Fun weekend!
##  2 2023-08-11 Email   Hello everyone!
##  3 2023-08-11 Slack   Hello everyone!
##  4 2023-08-18 Email   Fun weekend!
##  5 2023-08-14 Slack   Need assistance
##  6 2023-08-04 Email   Need assistance
##  7 2023-08-10 Twitter Hello everyone!
##  8 2023-08-04 Slack   Hello everyone!
##  9 2023-08-20 Email   Team meeting
## 10 2023-08-09 Slack   Hello everyone!
## # ℹ 990 more rows
```

Question-2: Channel Selection

Use the filter command to create a new dataframe that includes messages sent through the “Twitter” channel on August 2nd.

Solution:

```
# Enter code here
comm_data. %>%
  filter(channel == "Twitter", date == "2023-08-02")
```

```
## # A tibble: 15 × 5
##    date       channel sender        message         sentiment
##    <date>     <chr>   <chr>         <chr>               <dbl>
##  1 2023-08-02 Twitter alice@example Team meeting        0.210
##  2 2023-08-02 Twitter @erin_tweets  Exciting news!      0.750
##  3 2023-08-02 Twitter dave@example  Exciting news!      0.817
##  4 2023-08-02 Twitter @erin_tweets  Exciting news!      0.582
##  5 2023-08-02 Twitter @erin_tweets  Exciting news!     -0.525
##  6 2023-08-02 Twitter alice@example Team meeting        0.965
##  7 2023-08-02 Twitter dave@example  Great work!         0.516
##  8 2023-08-02 Twitter carol_slack   Hello everyone!     0.451
##  9 2023-08-02 Twitter carol_slack   Hello everyone!     0.174
## 10 2023-08-02 Twitter carol_slack   Need assistance     0.216
## 11 2023-08-02 Twitter @frank_chat   Need assistance    -0.115
## 12 2023-08-02 Twitter alice@example Need assistance     0.158
## 13 2023-08-02 Twitter carol_slack   Exciting news!     -0.693
## 14 2023-08-02 Twitter @bob_tweets   Need assistance    -0.282
## 15 2023-08-02 Twitter @erin_tweets  Need assistance     0.821
```

Question-3: Chronological Order

Utilizing the arrange command, arrange the “comm_data” dataframe in ascending order based on the “date” column.

Solution:

```
# Enter code here
arrange(comm_data. ,date)
```

```
## # A tibble: 1,000 × 5
##    date       channel sender        message         sentiment
##    <date>     <chr>   <chr>         <chr>               <dbl>
##  1 2023-08-01 Twitter alice@example Need assistance     0.677
##  2 2023-08-01 Twitter @bob_tweets   Need assistance     0.148
##  3 2023-08-01 Twitter @frank_chat   Need assistance     0.599
##  4 2023-08-01 Twitter @frank_chat   Exciting news!     -0.823
##  5 2023-08-01 Slack   @frank_chat   Team meeting       -0.202
##  6 2023-08-01 Slack   @bob_tweets   Exciting news!      0.146
##  7 2023-08-01 Slack   @erin_tweets  Great work!         0.244
##  8 2023-08-01 Twitter @frank_chat   Team meeting       -0.526
##  9 2023-08-01 Twitter @frank_chat   Exciting news!     -0.399
## 10 2023-08-01 Slack   @frank_chat   Need assistance     0.602
## # ℹ 990 more rows
```

Question-4: Distinct Discovery

Apply the distinct command to find the unique senders in the “comm_data” dataframe.

Solution:

```
# Enter code here
comm_data. %>% distinct(sender)
```

```
## # A tibble: 6 × 1
##   sender
##   <chr>
## 1 dave@example
## 2 @bob_tweets
## 3 @frank_chat
## 4 @erin_tweets
## 5 alice@example
## 6 carol_slack
```

Question-5: Sender Stats

Employ the count and group_by commands to generate a summary table that shows the count of messages sent by each sender in the “comm_data” dataframe.

Solution:

```
# Enter code here
comm_data. %>%
  group_by(sender) %>%
    summarise(count = n())
```

```
## # A tibble: 6 × 2
##   sender        count
##   <chr>         <int>
## 1 @bob_tweets     179
## 2 @erin_tweets    171
## 3 @frank_chat     174
## 4 alice@example   180
## 5 carol_slack     141
## 6 dave@example    155
```

Question-6: Channel Chatter Insights

Using the group_by and count commands, create a summary table that displays the count of messages sent through each communication channel in the “comm_data” dataframe.

Solution:

```
# Enter code here
comm_data. %>%
  group_by(channel) %>%
    summarise(count = n())
```

```
## # A tibble: 3 × 2
##   channel count
##   <chr>   <int>
## 1 Email     331
## 2 Slack     320
## 3 Twitter   349
```

Question-7: Positive Pioneers

Utilize the filter, select, and arrange commands to identify the top three senders with the highest average positive sentiment scores. Display their usernames and corresponding sentiment averages.

comm_data.%>% filter(sentiment>0) %>% group_by(sender)%>% summarise(sender, mean_sentiment = mean(sentiment))%>% distinct(sender, mean_sentiment) %>% slice_max((mean_sentiment), n = 3) **Solution:**

```
# Enter code here
comm_data.%>%
  filter(sentiment>0) %>%
    group_by(sender)%>%
      summarise(sender, mean_sentiment = mean(sentiment))%>%
          distinct(sender, mean_sentiment) %>%
            arrange(desc(mean_sentiment)) %>%
  ungroup() %>%
  slice(1:3)
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## ℹ Please use `reframe()` instead.
## ℹ When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

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

```
## # A tibble: 3 × 2
##   sender        mean_sentiment
##   <chr>                  <dbl>
## 1 dave@example           0.541
## 2 @frank_chat            0.528
## 3 alice@example          0.493
```

Question-8: Message Mood Over Time

With the group_by, summarise, and arrange commands, calculate the average sentiment score for each day in the “comm_data” dataframe.

Solution:

```
# Enter code here
comm_data. %>%
    group_by(date)%>%
      summarise(date, mean_sentiment = mean(sentiment))%>%
        distinct(date, mean_sentiment)
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## ℹ Please use `reframe()` instead.
## ℹ When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

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

```
## # A tibble: 20 × 2
## # Groups:   date [20]
##    date       mean_sentiment
##    <date>              <dbl>
##  1 2023-08-01        -0.0616
##  2 2023-08-02         0.136
##  3 2023-08-03         0.107
##  4 2023-08-04        -0.0510
##  5 2023-08-05         0.193
##  6 2023-08-06        -0.0144
##  7 2023-08-07         0.0364
##  8 2023-08-08         0.0666
##  9 2023-08-09         0.0997
## 10 2023-08-10        -0.0254
## 11 2023-08-11        -0.0340
## 12 2023-08-12         0.0668
## 13 2023-08-13        -0.0604
## 14 2023-08-14        -0.0692
## 15 2023-08-15         0.0617
## 16 2023-08-16        -0.0220
## 17 2023-08-17        -0.0191
## 18 2023-08-18        -0.0760
## 19 2023-08-19         0.0551
## 20 2023-08-20         0.0608
```

Question-9: Selective Sentiments

Use the filter and select commands to extract messages with a negative sentiment score (less than 0) and create a new dataframe.

Solution:

```
# Enter code here
comm_data. %>%
  filter(sentiment<0)%>%
    select(message,sentiment)
```

```
## # A tibble: 487 × 2
##    message         sentiment
##    <chr>               <dbl>
##  1 Hello everyone!    -0.143
##  2 Need assistance    -0.108
##  3 Hello everyone!    -0.741
##  4 Hello everyone!    -0.188
##  5 Hello everyone!    -0.933
##  6 Need assistance    -0.879
##  7 Great work!        -0.752
##  8 Team meeting       -0.787
##  9 Fun weekend!       -0.539
## 10 Exciting news!     -0.142
## # ℹ 477 more rows
```

Question-10: Enhancing Engagement

Apply the mutate command to add a new column to the “comm_data” dataframe, representing a sentiment label: “Positive,” “Neutral,” or “Negative,” based on the sentiment score.

Solution:

```
# Enter code here
comm_data.%>%
  mutate(sentiment_label = ifelse(sentiment>0, "Positive",
                                  ifelse(sentiment<0, "Negative", "Neutral"))) %>%
  select(sentiment, sentiment_label)
```

```
## # A tibble: 1,000 × 2
##    sentiment sentiment_label
##        <dbl> <chr>
##  1     0.824 Positive
##  2     0.662 Positive
##  3    -0.143 Negative
##  4     0.380 Positive
##  5     0.188 Positive
##  6    -0.108 Negative
##  7    -0.741 Negative
##  8    -0.188 Negative
##  9     0.618 Positive
## 10    -0.933 Negative
## # ℹ 990 more rows
```

Question-11: Message Impact

Create a new dataframe using the mutate and arrange commands that calculates the product of the sentiment score and the length of each message. Arrange the results in descending order.

Solution:

```
# Enter code here
comm_data.%>%
  mutate(new = sentiment*nchar(message)) %>%
    arrange(desc(new))
```

```
## # A tibble: 1,000 × 6
##    date       channel sender       message         sentiment   new
##    <date>     <chr>   <chr>        <chr>               <dbl> <dbl>
##  1 2023-08-16 Email   @frank_chat  Hello everyone!     0.998  15.0
##  2 2023-08-14 Slack   @erin_tweets Hello everyone!     0.988  14.8
##  3 2023-08-18 Email   dave@example Hello everyone!     0.978  14.7
##  4 2023-08-17 Email   dave@example Hello everyone!     0.977  14.7
##  5 2023-08-07 Slack   carol_slack  Hello everyone!     0.973  14.6
##  6 2023-08-06 Slack   dave@example Hello everyone!     0.968  14.5
##  7 2023-08-08 Slack   @frank_chat  Need assistance     0.964  14.5
##  8 2023-08-09 Email   @erin_tweets Need assistance     0.953  14.3
##  9 2023-08-17 Twitter @frank_chat  Hello everyone!     0.952  14.3
## 10 2023-08-12 Email   carol_slack  Need assistance     0.938  14.1
## # ℹ 990 more rows
```

Question-12: Daily Message Challenge

Use the group_by, summarise, and arrange commands to find the day with the highest total number of characters sent across all messages in the “comm_data” dataframe.

Solution:

```
# Enter code here
comm_data. %>%
    group_by(date)%>%
      summarise(date, all_char = sum(nchar(message)))%>%
      distinct(date, all_char)%>%
  arrange(desc(all_char))%>%
  ungroup()%>%
  slice(1)
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## ℹ Please use `reframe()` instead.
## ℹ When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

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

```
## # A tibble: 1 × 2
##   date       all_char
##   <date>        <int>
## 1 2023-08-10      875
```

Question-13: Untidy data

Can you list at least two reasons why the dataset illustrated in slide 10 is non-tidy? How can it be made Tidy?

Solution: *There are multiple variables in the columns 2,3,4, and there are also observations in the columns which makes it more difficult to handle(filter etc.)In order to make it tidy i should make sure each variable must have its own column, each observation must have its own row and each value must have its own cell.*

Challenge-5

Merkayla Wong

13-09-2023

Questions

Question-1: Local Variable Shadowing

Create an R function that defines a global variable called `x` with a value of 5. Inside the function, declare a local variable also named `x` with a value of 10. Print the value of `x` both inside and outside the function to demonstrate shadowing.

Solutions:

```
# Enter code here
x<- 5
sprintf("The value assigned to z outside the function is %d",x)
```

```
## [1] "The value assigned to z outside the function is 5"
```

```
foo<- function(x = 2) {
x <- 10
return(x)
}
foo()
```

```
## [1] 10
```

```
sprintf("The final value of z after reassigning it to a different value inside the function i
s %d",x)
```

```
## [1] "The final value of z after reassigning it to a different value inside the function is
5"
```

Question-2: Modify Global Variable

Create an R function that takes an argument and adds it to a global variable called `total`. Call the function multiple times with different arguments to accumulate the values in `total`.

Solutions:

```
# Enter code here
total<-0
addition_to_total<-function(n){
  total<<-total+n
}

addition_to_total(5)
addition_to_total(7)
addition_to_total(9)
print(total)
```

```
## [1] 21
```

Question-3: Global and Local Interaction

Write an R program that includes a global variable `total` with an initial value of 100. Create a function that takes an argument, adds it to `total`, and returns the updated `total`. Demonstrate how this function interacts with the global variable.

Solutions:

```
# Enter code here
total<-100
added_to_total<-function(n){
  total<<-total+n
  return(total)
}
added_to_total(35)
```

```
## [1] 135
```

```
sprintf("The final value of total after reassigning it to a different value inside the functi
on is %d",total)
```

```
## [1] "The final value of total after reassigning it to a different value inside the functio
n is 135"
```

Question-4: Nested Functions

Define a function `outer_function` that declares a local variable `x` with a value of 5. Inside `outer_function`, define another function `inner_function` that prints the value of `x`. Call both functions to show how the inner function accesses the variable from the outer function's scope.

Solutions:

```
# Enter code here
outer_function<- function(){
  x<-5
  inner_function<- function(){
    print(x)
  }
  inner_function()
}
outer_function()
```

```
## [1] 5
```

Question-5: Meme Generator Function

Create a function that takes a text input and generates a humorous meme with the text overlaid on an image of your choice. You can use the `magick` package for image manipulation. You can find more details about the commands offered by the package, with some examples of annotating images here: https://cran.r-project.org/web/packages/magick/vignettes/intro.html (https://cran.r-project.org/web/packages/magick/vignettes/intro.html)

Solutions:

```
# Enter code here
library(magick)
```

```
## Linking to ImageMagick 6.9.12.93
## Enabled features: cairo, freetype, fftw, ghostscript, heic, lcms, pango, raw, rsvg, webp
## Disabled features: fontconfig, x11
```

```
create_meme <- function(image_path, text) {

  img <- image_read(image_path)

  text_color <- "white"
  text_size <- 40
  text_font <- "comic sans"

  image_annotate(
    img,
    text,
    location = "+30+150",
    color = text_color,
    size = text_size,
    font = text_font
  )
}
image_path <-"help meme.png"
text <- "me running to my TA everytime
my code cant knit"
create_meme(image_path, text)
```

Question-6: Text Analysis Game

Develop a text analysis game in which the user inputs a sentence, and the R function provides statistics like the number of words, characters, and average word length. Reward the user with a “communication skill level” based on their input.

Solutions:

```
# Enter code here
communication_skill_level<-function(ave_word_length){
  if (ave_word_length >= 12){
    return("pro")
  }else if(ave_word_length >=7){
    return("average")
  }else{
    return("poor")
  }
}
input_sentence<-function(){
  sentence<-readline("put sentence here ")
  num_of_chars<-nchar(sentence)
  words<-strsplit(sentence, split= " ")
  num_of_words<-lengths(words)
  ave_word_length<-sum(nchar(words))/num_of_words
  print(num_of_chars)
  print(num_of_words)
  print(ave_word_length)
  print(communication_skill_level(ave_word_length))
}
```