# Challenge-2

Janelle Tan

2023-08-21

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. Edit content wherever indicated
5. Remember to set `eval=TRUE` after completing the code to generate the output
6. Ensure that `echo=TRUE` so that the code is rendered in the final document
7. Inform the tutor/instructor upon completion
8. Submit the document on Canvas after they approve
9. Attendance will be marked only after submission
10. 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 rstudio 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 = comma separated values. CSV is popular for several reasons:

Simplicity: CSV files are easy to create and manipulate using a basic text editor or spreadsheet software. The format is human-readable and doesn't require complex encoding or binary data structures.

Compatibility: CSV is a plain text format, so it can be opened and read by a wide range of software applications, making it a universal choice for data exchange between different programs.

Lightweight: CSV files have a small file size compared to some other file formats like Excel spreadsheets, making them suitable for storing and sharing data without using excessive storage space.

Tabular Data Representation: CSV is well-suited for representing structured tabular data, which is a common way to organize data in rows and columns. This makes it ideal for datasets, lists, and other structured information.

Platform Independence: Since CSV is a plain text format, it is platform-independent. It can be used on various operating systems without compatibility issues.

Integration with Programming Languages: Many programming languages have built-in or third-party libraries to read and write CSV files. This makes CSV a popular choice for data import/export operations in software development.

Data Interchange: CSV files are commonly used to exchange data between different systems or software applications because of their simplicity and compatibility.

However, it's important to note that CSV has limitations. It lacks support for more complex data types, metadata, and formatting options that some other formats (like Excel or JSON) offer. Additionally, handling special cases such as text fields that contain commas or newline characters requires careful handling or quoting within the CSV data.

In summary, the term "CSV" stands for "Comma-Separated Values," and its popularity as a format for storing tabular data is attributed to its simplicity, compatibility, lightweight nature, and ease of integration with various software tools and programming languages.

**Question 1.2:** load the `readr` package to work with `.csv` files in rstudio.

**Solution:**

```
# Load the necessary package to work with CSV files in R.

library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
```

```
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

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

**Solution:**

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

read_csv("NM2207 W2 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 x 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

music <- read_csv("NM2207 W2 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:**

?read_csv()

```
knitr::include_graphics("C:\\Users\\janel\\OneDrive\\Documents\\Y2S1 NM2207\\Week 2\\W2Screenshot.png")
```

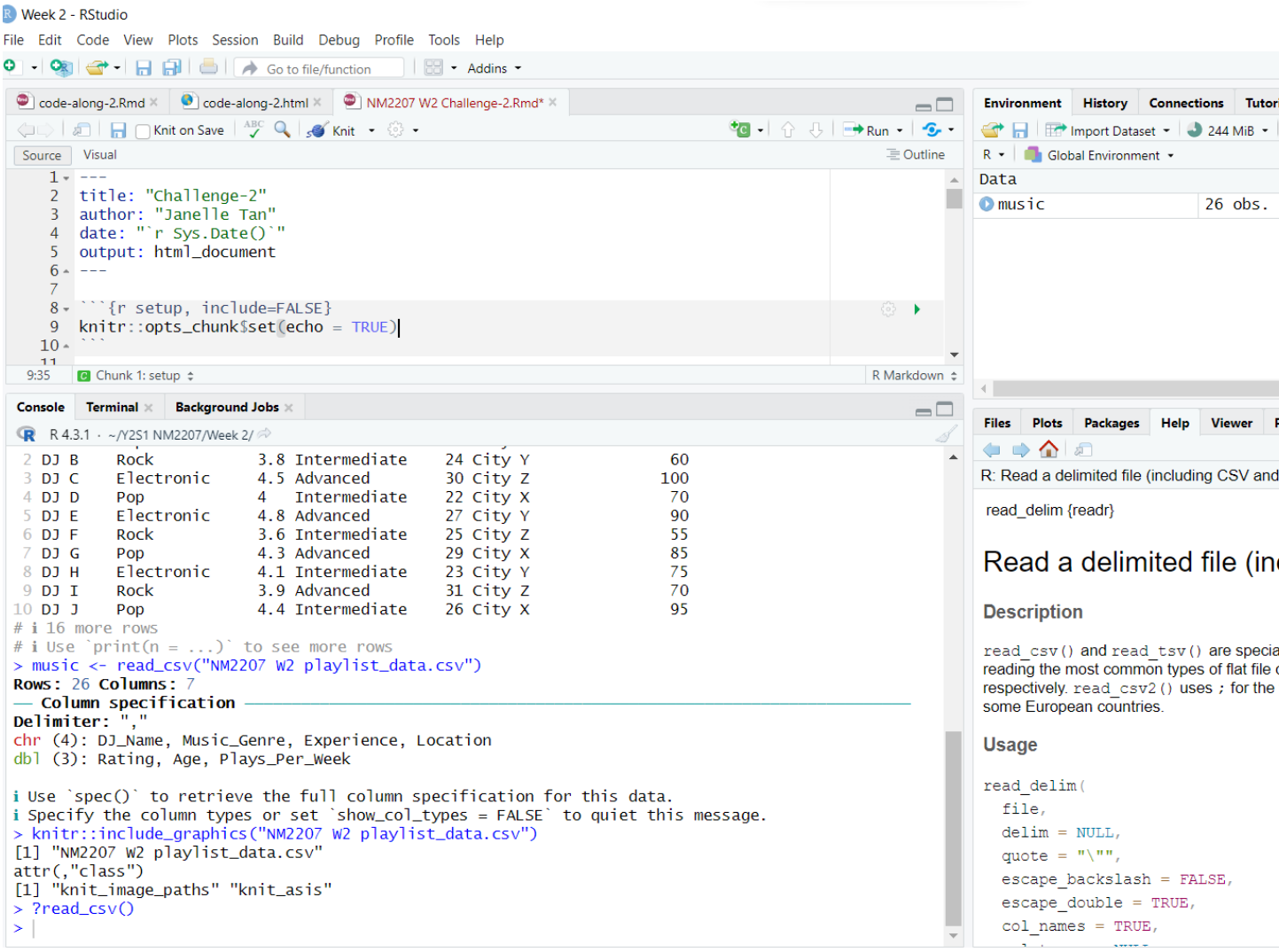



Figure 1: Screenshot

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

**Solution:** The skip argument in the read_csv() function is used to specify the number of lines at the beginning of the CSV file that should be skipped during the reading process. This argument is helpful when you have header rows or other introductory information in your CSV file that you want to exclude from the imported data.

**Question 1.7:** Display the contents of the data-set

**Solution:**

```
# Type the name of the variable, to see what it contains
glimpse(music)
```

```
## 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 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 rstudio 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

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(music)
```

```
## # A tibble: 6 x 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(music)
```

```
## 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:**

```
ncol(music)
```

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

**Question 2.4:** What is the total count of DJs?

**Solution:**

```
nrow(music)
```

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

**Question 2.5:** Display all the location of all the DJs

**Solution:**

```
music$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:**

```
music$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
```

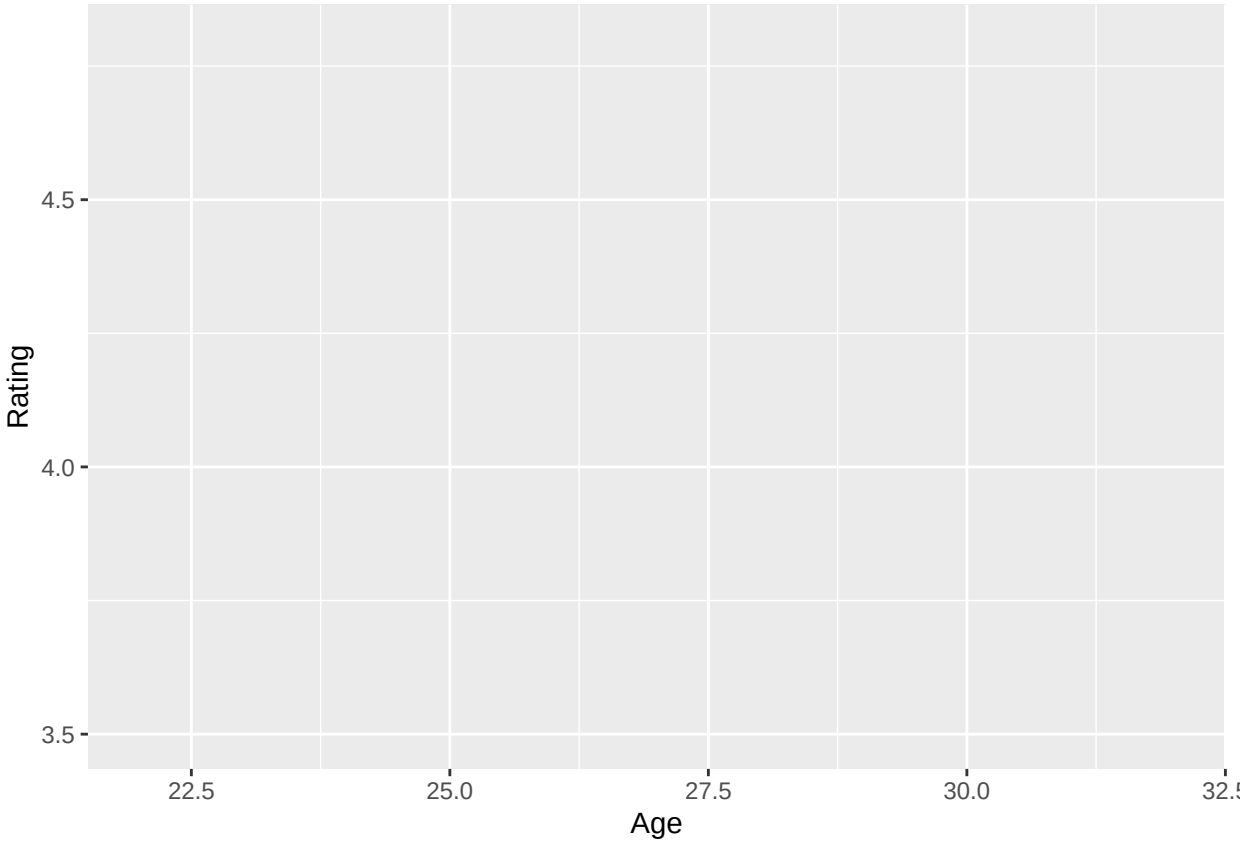
**Task-3** Let us plot the data to get more insights about the DJs.

**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(music, aes(x=Age,y=Rating))
```

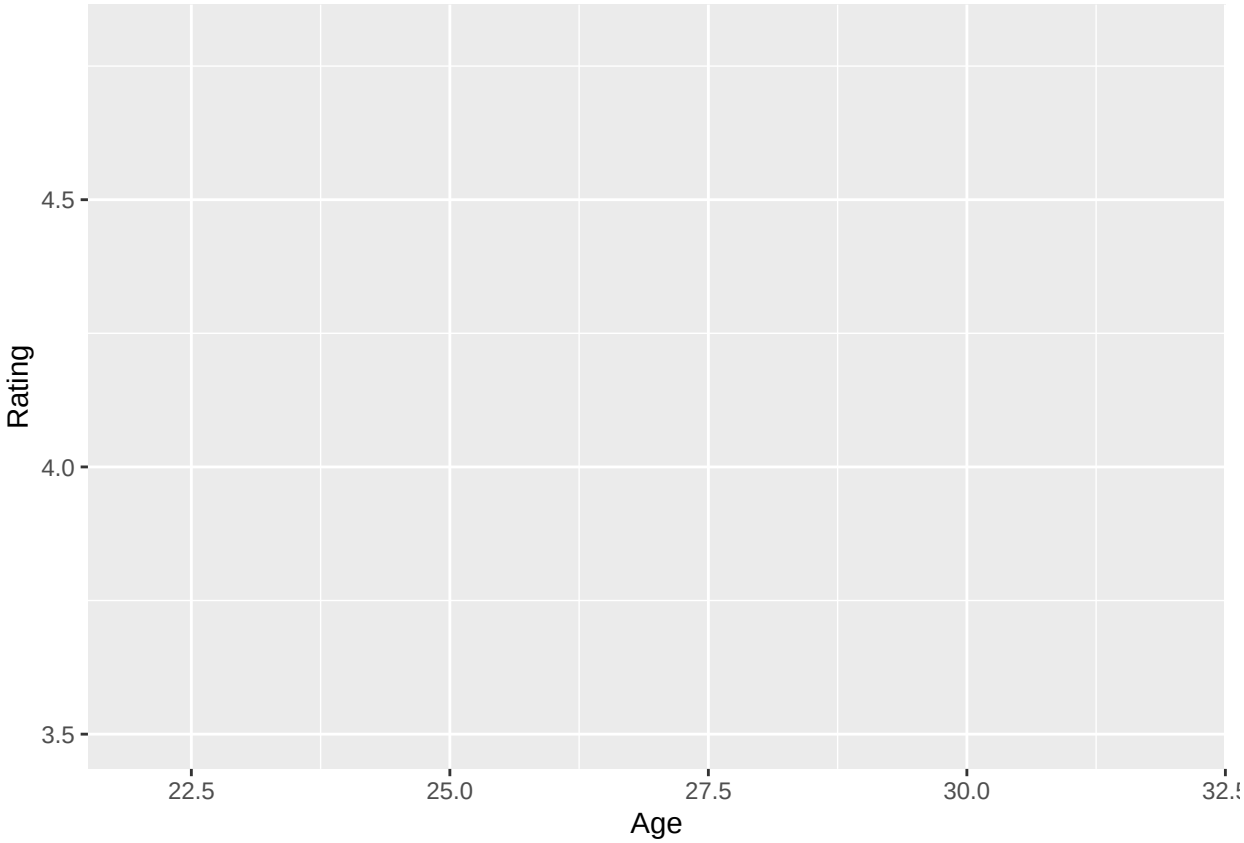



**Question 3.2:** Label the x-axis as "Age" and the y-axis as "Rating."

**Solution:**

```
# complete the code to generate the plot

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

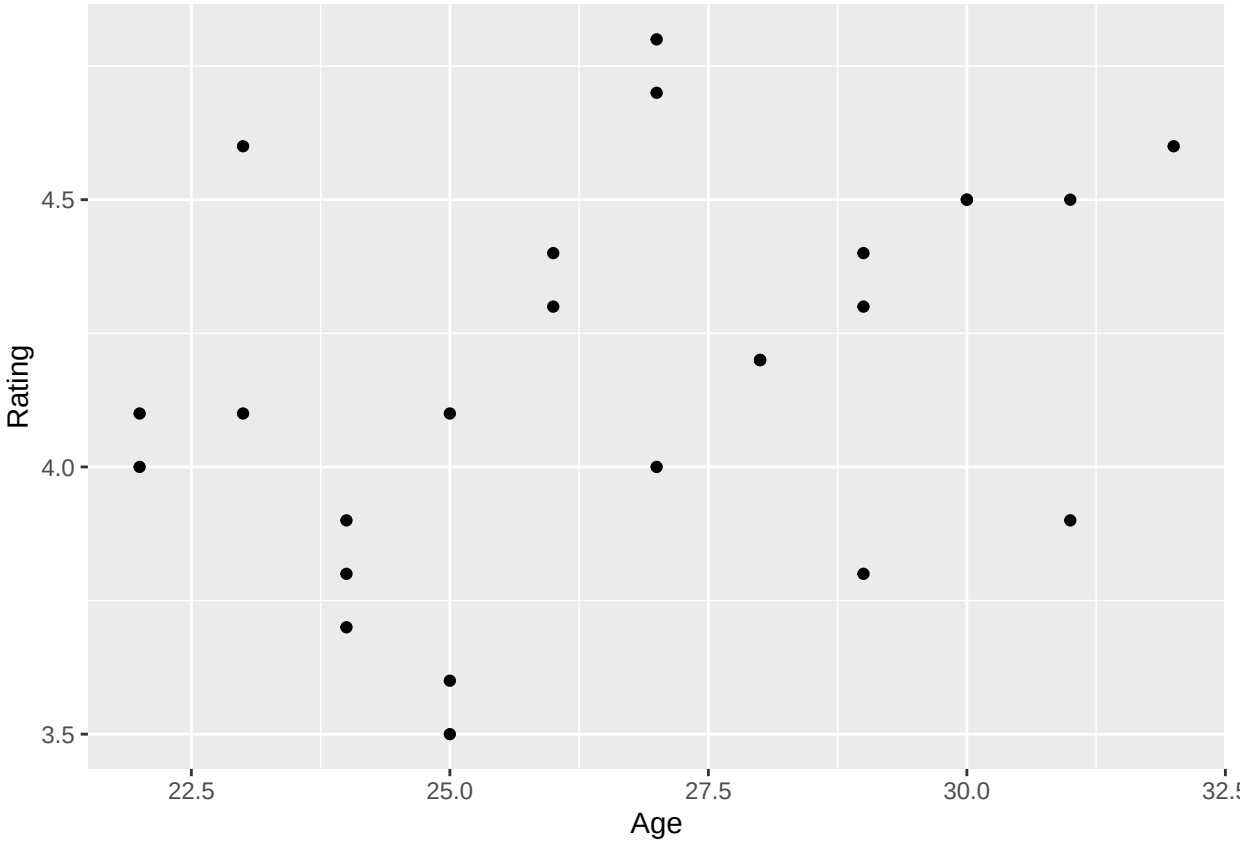



**Question 3.3:** Represent data using points

**Solution:**

```
# complete the code to generate the plot

ggplot(data=music,mapping=aes(x=Age,y=Rating)) + geom_point() +
labs(x="Age",y="Rating")
```

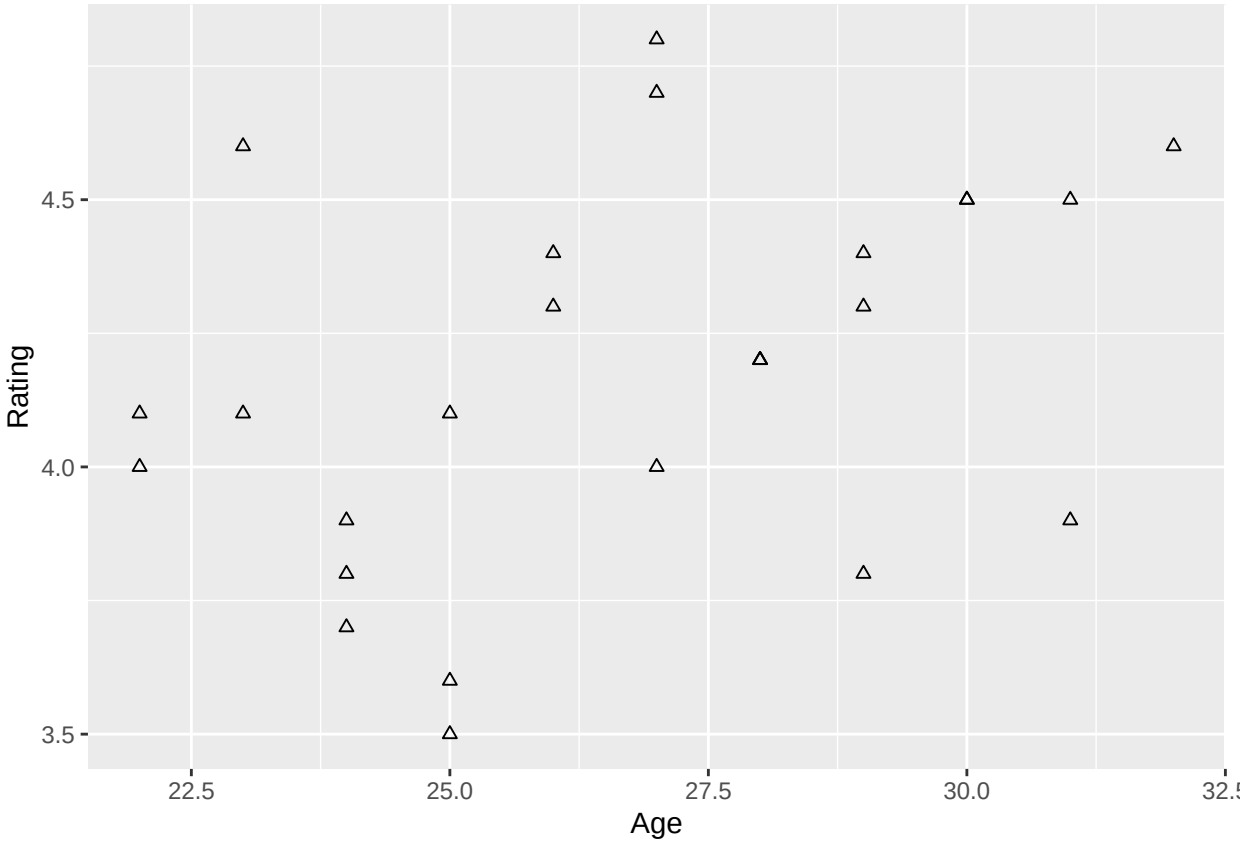



**Question 3.4:** Can you change the points represented by dots/small circles to any other shape of your liking?

**Solution:**

```
# complete the code to generate the plot

ggplot(data=music,mapping=aes(x=Age,y=Rating)) + geom_point(shape=2) +
labs(x="Age",y="Rating")
```

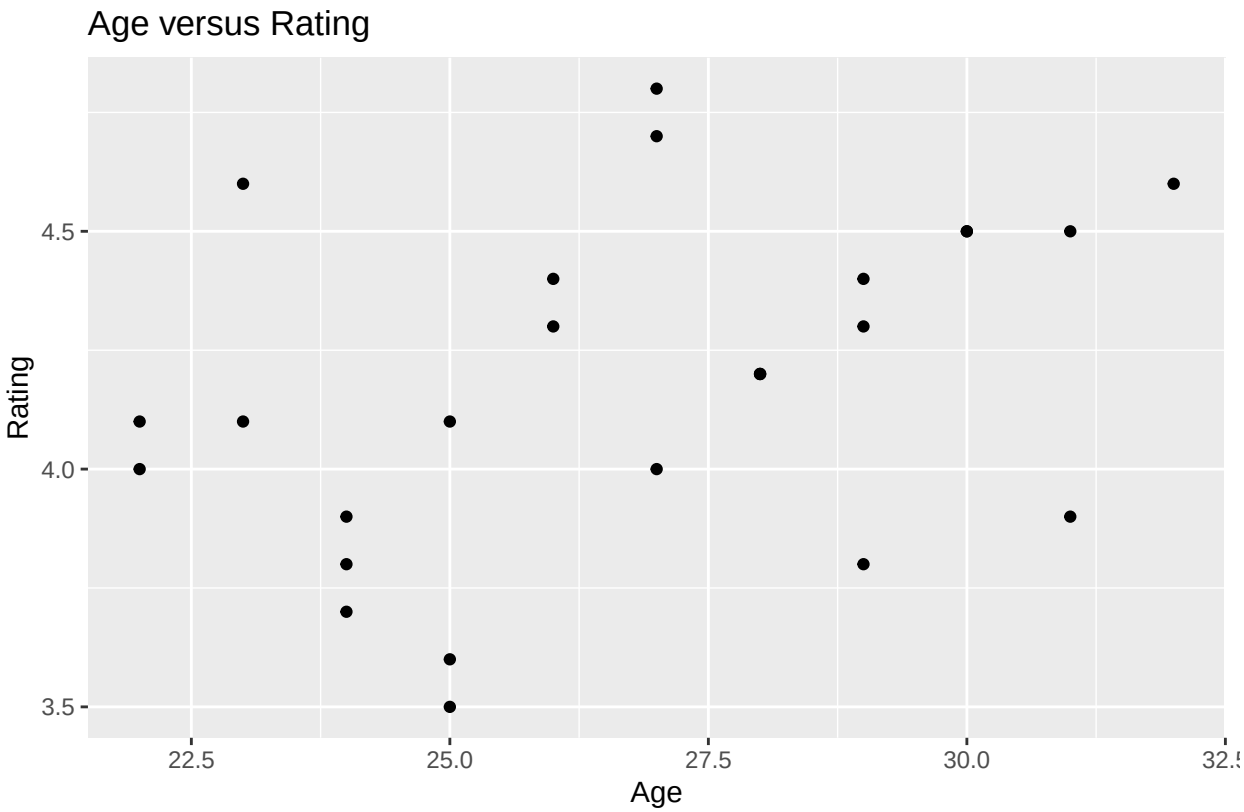



**Question 3.5:** Insert a suitable title and briefly provide your insights in the caption

**Solution:**

```
# complete the code to generate the plot

ggplot(data=music,mapping=aes(x=Age,y=Rating)) +
 geom_point() +
 labs(x="Age",y="Rating",
title="Age versus Rating",
caption="There does not seem to be a very strong correlation between Age and ratings of Djs")
```




There does not seem to be a very strong correlation between Age and ratings of Djs