Challenge-2

Ho Wei Ni

2023-08-21

I. Exploring music preferences

Task-1 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" stand for comma-separated values and indicates that `playlist_data.csv` is a comma-separated values file. It is a popular format for storing tabular data due to its ability to be used across nearly every platform, allowing for ease of data exchange between different systems.

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 --
## 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("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

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()
```

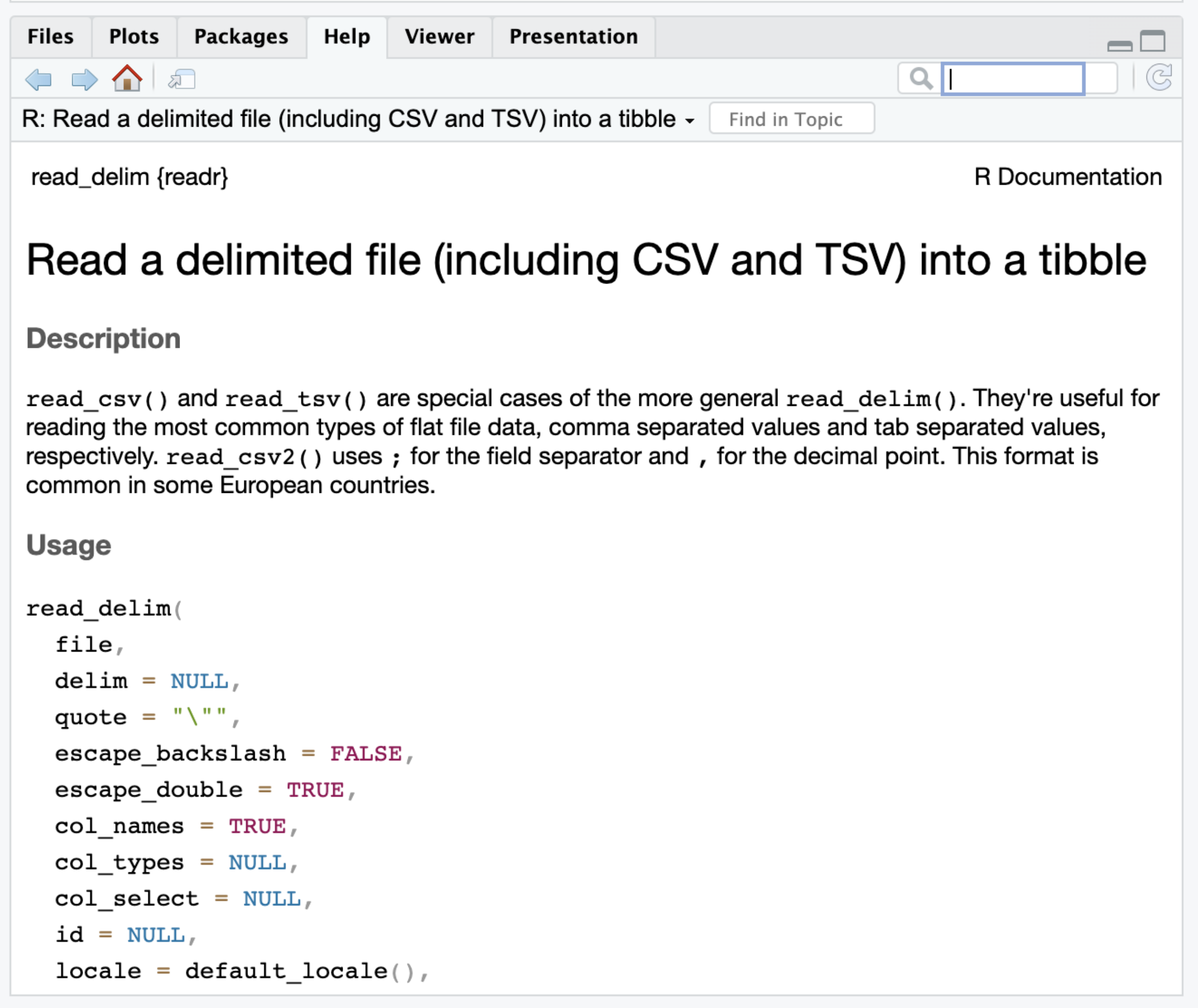
```
knitr::include_graphics("Screenshot 2023-08-21 at 3.02.46 PM.png")
```

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

Solution: It shows the number of lines to skip before reading data. If comment is supplied, any commented lines are ignored after skipping.

Question 1.7: Display the contents of the data-set

Solution:


Files
Plots
Packages
Help
Viewer
Presentation
R: Read a delimited file (including CSV and TSV) into a tibble
Find in Topic

read_delim {readr} R Documentation

Read a delimited file (including CSV and TSV) into a tibble

Description

`read_csv()` and `read_tsv()` are special cases of the more general `read_delim()`. They're useful for reading the most common types of flat file data, comma separated values and tab separated values, respectively. `read_csv2()` uses `;` for the field separator and `,` for the decimal point. This format is common in some European countries.

Usage

```
read_delim(
  file,
  delim = NULL,
  quote = "\"",
  escape_backslash = FALSE,
  escape_double = TRUE,
  col_names = TRUE,
  col_types = NULL,
  col_select = NULL,
  id = NULL,
  locale = default_locale(),
```

Figure 1: Screenshot of information displayed in Help tab

```
playlist_data
```

```
## # 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 1.8: 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:

```
# 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 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(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
```

There are 7 columns in the dataset.

Question 2.4: What is the total count of DJs?

Solution:

```
# Number of DJs
playlist_data$DJ_Name
```

```
##  [1] "DJ A" "DJ B" "DJ C" "DJ D" "DJ E" "DJ F" "DJ G" "DJ H" "DJ I" "DJ J"
## [11] "DJ K" "DJ L" "DJ M" "DJ N" "DJ O" "DJ P" "DJ Q" "DJ R" "DJ S" "DJ T"
## [21] "DJ U" "DJ V" "DJ W" "DJ X" "DJ Y" "DJ Z"
```

There is a total of 26 DJs.

Question 2.5: Display all the location of all the DJs

Solution:

```
# Location of DJs
playlist_data %>% select(DJ_Name,Location)
```

```
## # A tibble: 26 x 2
##    DJ_Name Location
##    <chr>   <chr>
##  1 DJ A    City X
##  2 DJ B    City Y
##  3 DJ C    City Z
##  4 DJ D    City X
##  5 DJ E    City Y
##  6 DJ F    City Z
##  7 DJ G    City X
##  8 DJ H    City Y
##  9 DJ I    City Z
## 10 DJ J    City X
## # i 16 more rows
```

Question 2.6: Display the age of the DJs

Solution:

```
# Age of DJs
playlist_data %>% select(DJ_Name,Age)
```

```
## # A tibble: 26 x 2
##    DJ_Name   Age
##    <chr>   <dbl>
##  1 DJ A       28
##  2 DJ B       24
##  3 DJ C       30
##  4 DJ D       22
##  5 DJ E       27
##  6 DJ F       25
##  7 DJ G       29
##  8 DJ H       23
##  9 DJ I       31
## 10 DJ J       26
## # i 16 more rows
```

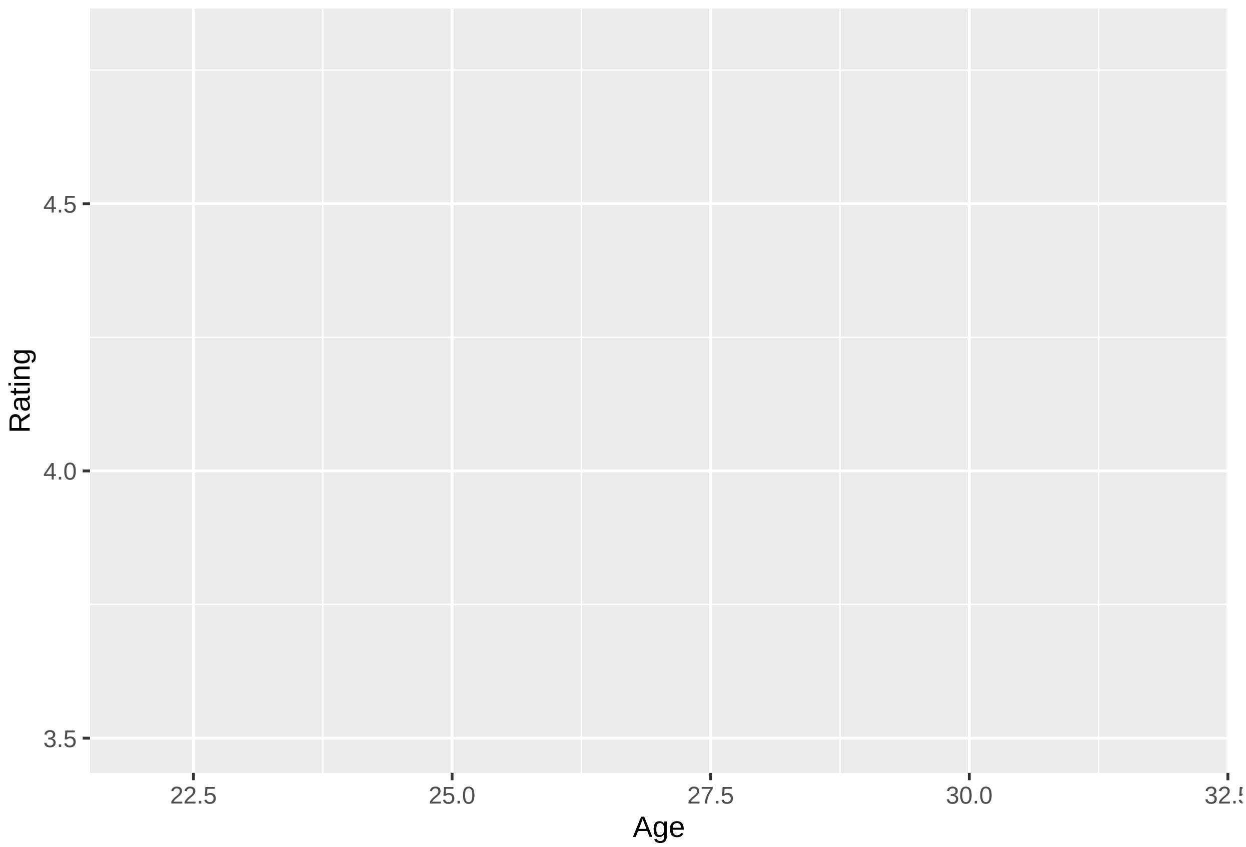
Task-3 Let us plot the data to get more insights about the DJs.

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

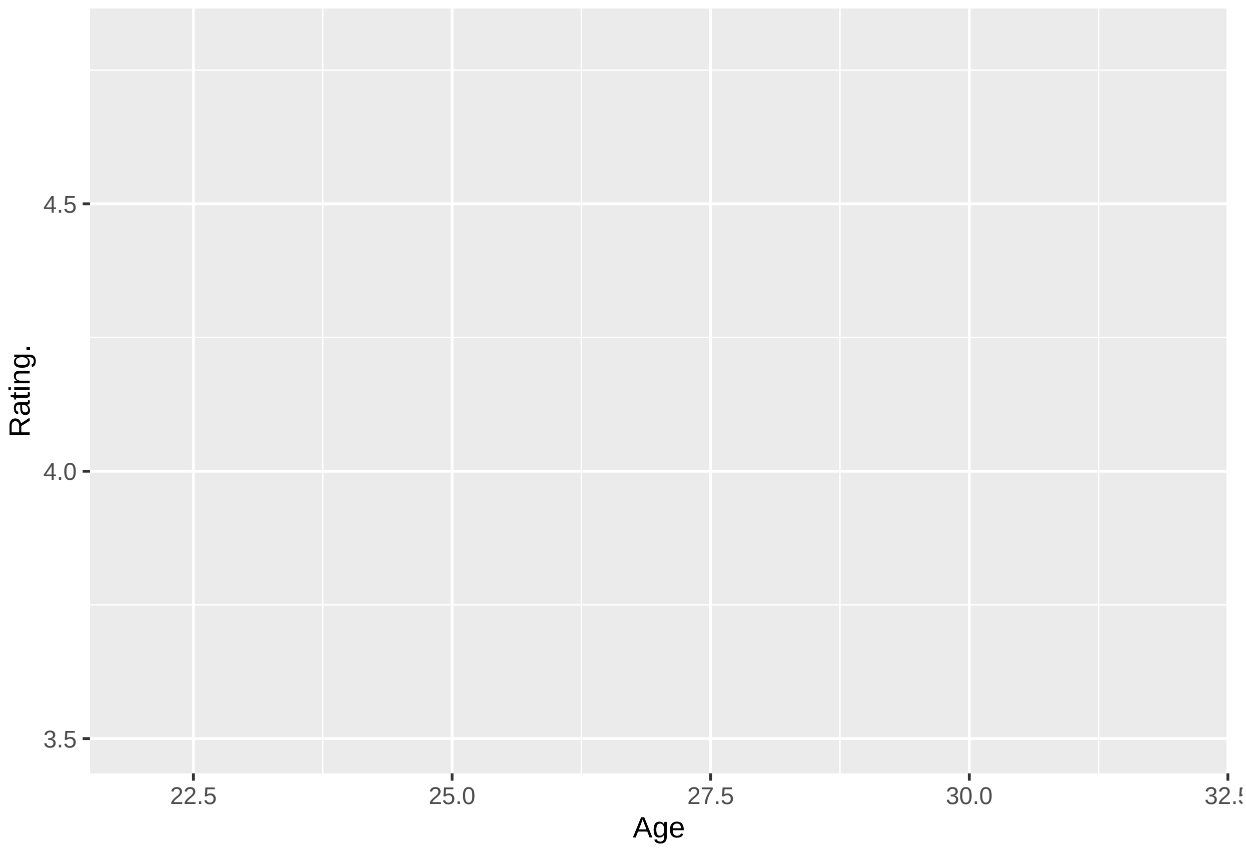


Question 3.2: Label the x-axis as “Age” and the y-axis as “Rating.”

Solution:

```
# complete the code to generate the plot

ggplot(data = playlist_data)  +
  aes(x=Age,y=Rating) +
    labs(x="Age",y="Rating.")
```

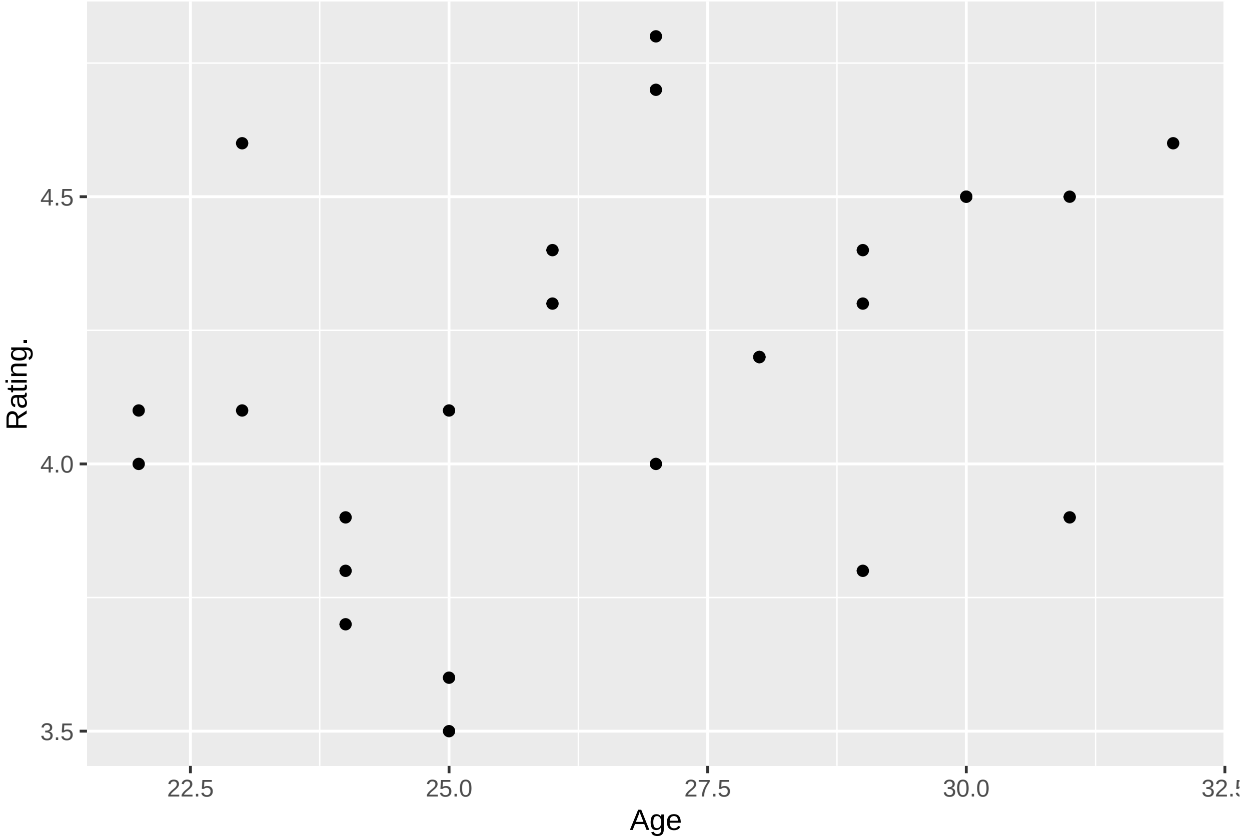


Question 3.3: Represent data using points

Solution:

```
# complete the code to generate the plot

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

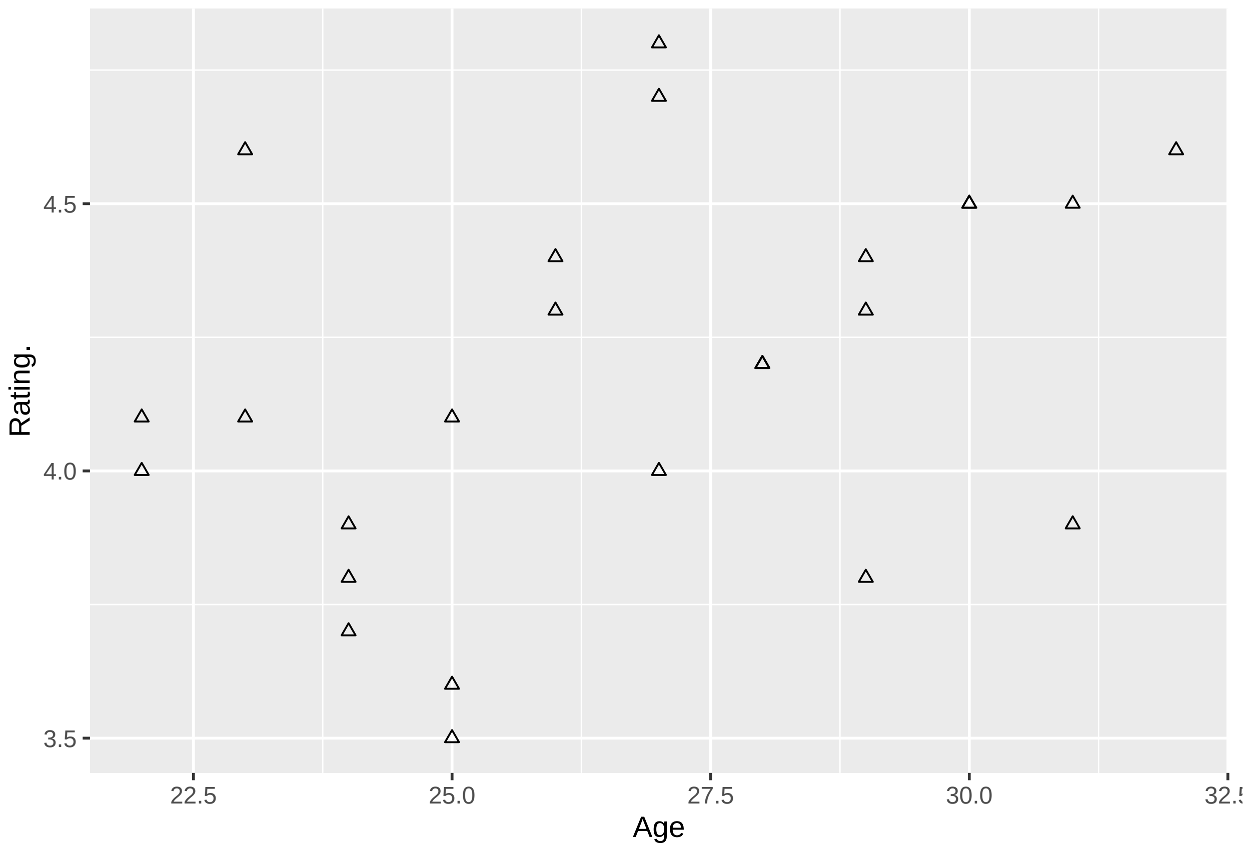


Question 3.4: Can you change the points represented by dots/small circles to any other shape of your liking?

Solution:

```
# complete the code to generate the plot

ggplot(data = playlist_data)  +
  aes(x=Age,y=Rating) +
    geom_point(shape = 24) +
    labs(x="Age",y="Rating.")
```

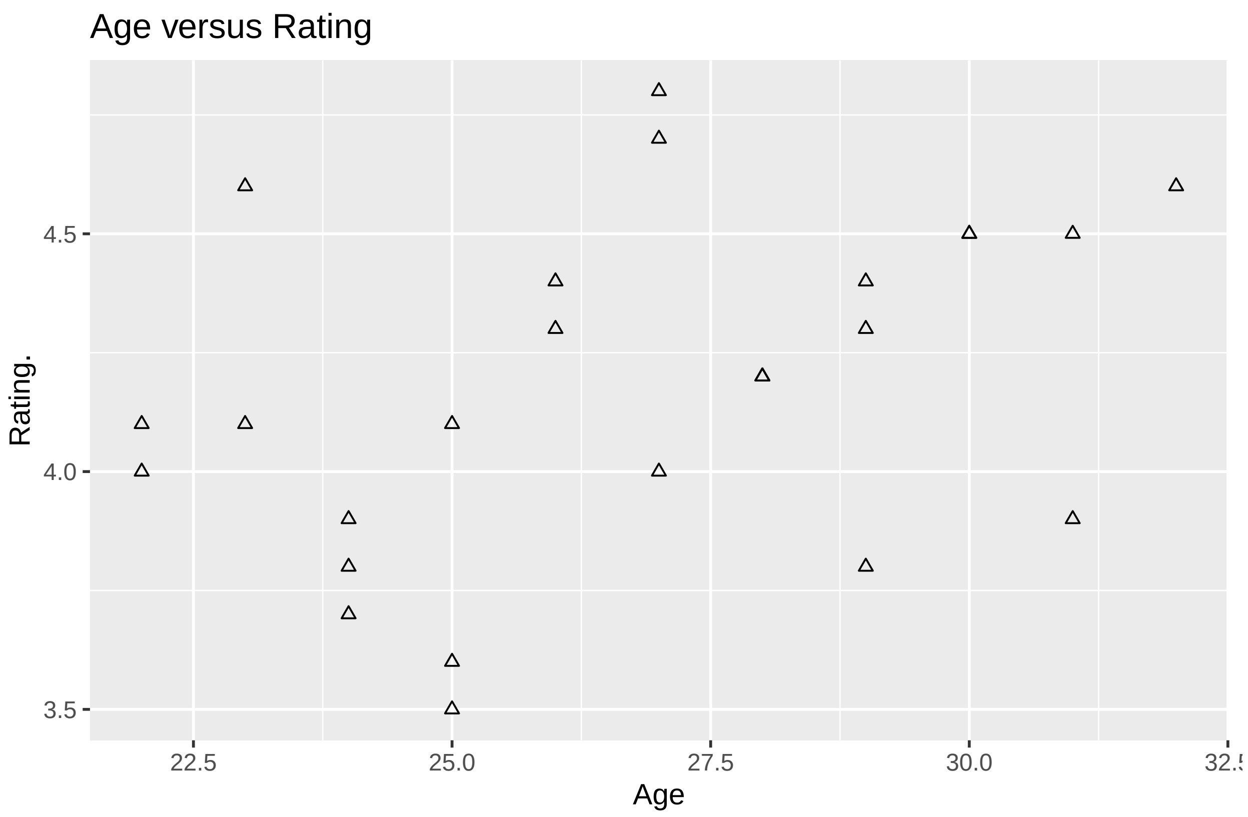


Question 3.5: Insert a suitable title and briefly provide your insights in the caption

Solution:

```
# complete the code to generate the plot

ggplot(data = playlist_data)  +
  aes(x=Age,y=Rating) +
    geom_point(shape = 24) +
    labs(x="Age",y="Rating.",
          title="Age versus Rating",
            caption="Generally, age and rating have a weak positive association.")
```


Age versus Rating

Generally, age and rating have a weak positive association.