Challenge-2: Solutions

Narayani

2023-09-09

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: A CSV (Comma-Separated Values) file is a plain-text file format used for storing tabular data, such as spreadsheets or database tables. In a CSV file, each line represents a single row of data, and the values within each row are separated by commas (or other specified delimiters).

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   3.4.2     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.1
## ── 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 conflic
ts 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 × 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

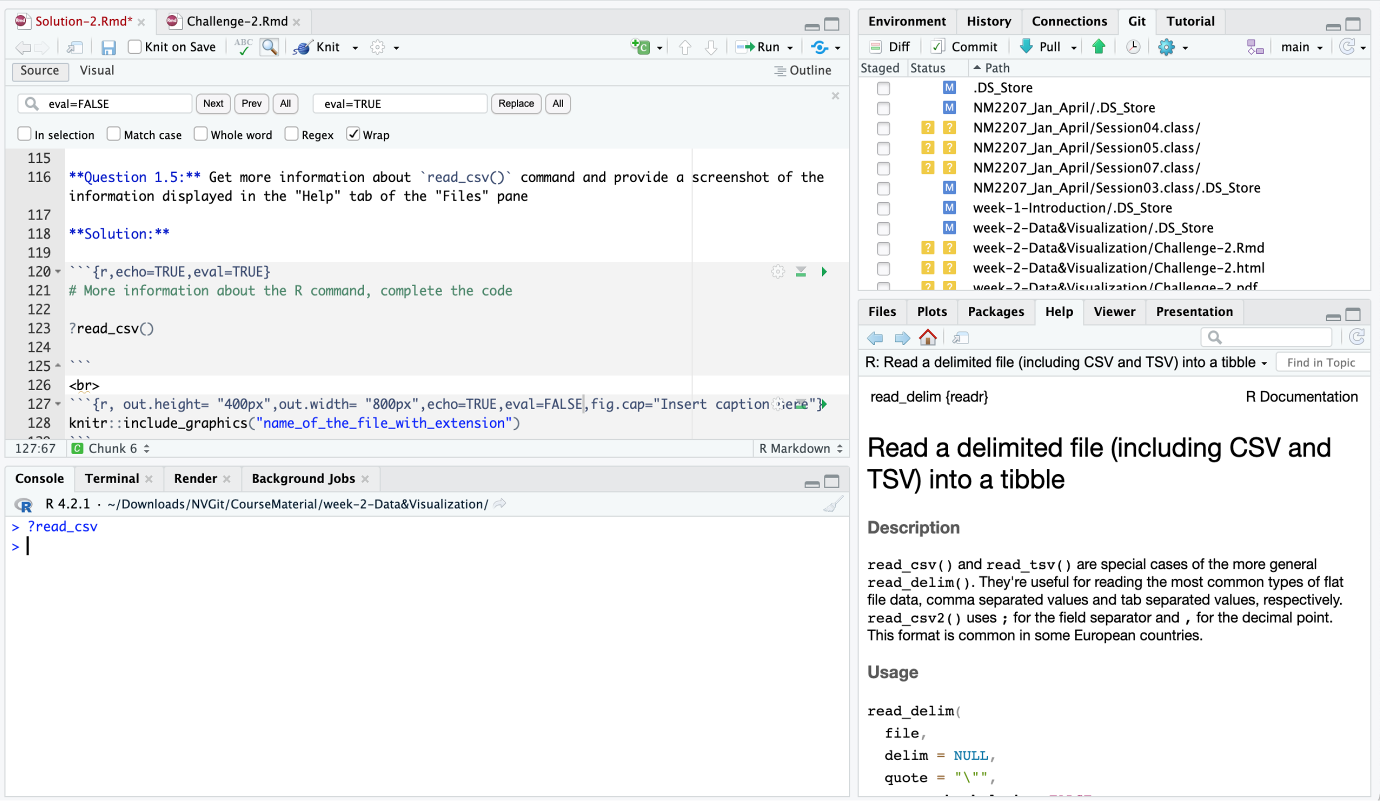
Question 1.5: Get more information about `read_csv()` command and provide a screenshot of the information displayed in the "Help" tab of the "Files" pane

Solution:

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

?read_csv()
```

```
knitr::include_graphics("solution.png")
```



eval=FALSE
eval=TRUE

~~~
**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:**

```{r,echo=TRUE,eval=TRUE}
# More information about the R command, complete the code

?read_csv()

~~~


```{r, out.height= "400px",out.width= "800px",echo=TRUE,eval=FALSE,fig.cap="Insert caption here"}
knitr::include_graphics("name_of_the_file_with_extension")
```

.DS_Store
NM2207_Jan_April/.DS_Store
NM2207_Jan_April/Session04.class/
NM2207_Jan_April/Session05.class/
NM2207_Jan_April/Session07.class/
NM2207_Jan_April/Session03.class/.DS_Store
week-1-Introduction/.DS_Store
week-2-Data&Visualization/.DS_Store
week-2-Data&Visualization/Challenge-2.Rmd
week-2-Data&Visualization/Challenge-2.html
week-2-Data&Visualization/Challenge-2.pdf

```
R 4.2.1 · ~/Downloads/NVGit/CourseMaterial/week-2-Data&Visualization/
> ?read_csv
>
```

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

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 = "\"",
```

Insert caption here

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

Solution: It skips the number of specified (integer) lines of the data file before beginning to read data.

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 requi
red
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:

```
ncol(playlist_data)
```

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

Question 2.4: What is the total count of DJs?

Solution:

```
nrow(playlist_data)
```

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

Question 2.5: Display all the location of all the DJs

Solution:

```
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:

```
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

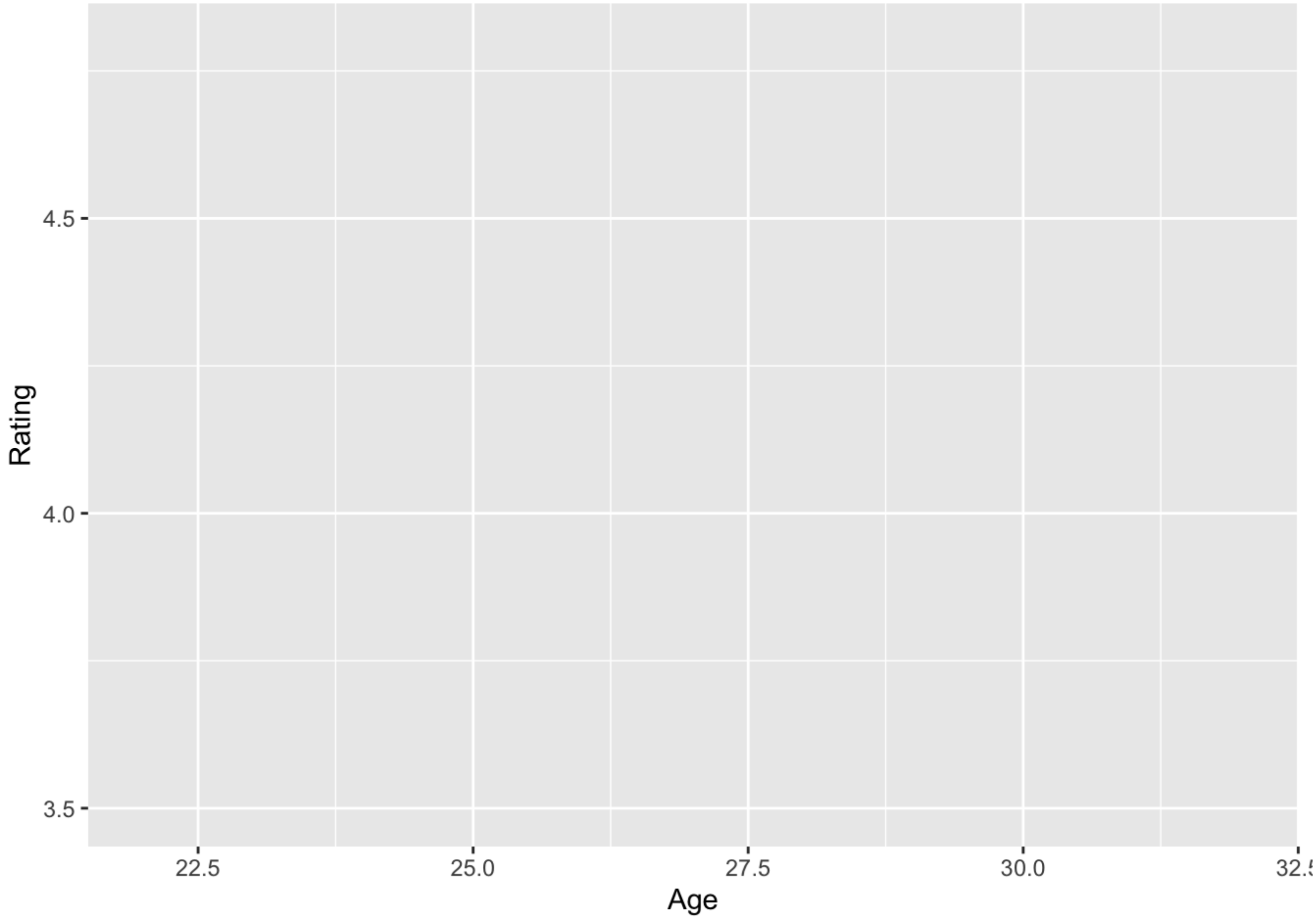
Let us plot the data to get more insights about the DJs.

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=Age,y=Rating)
```

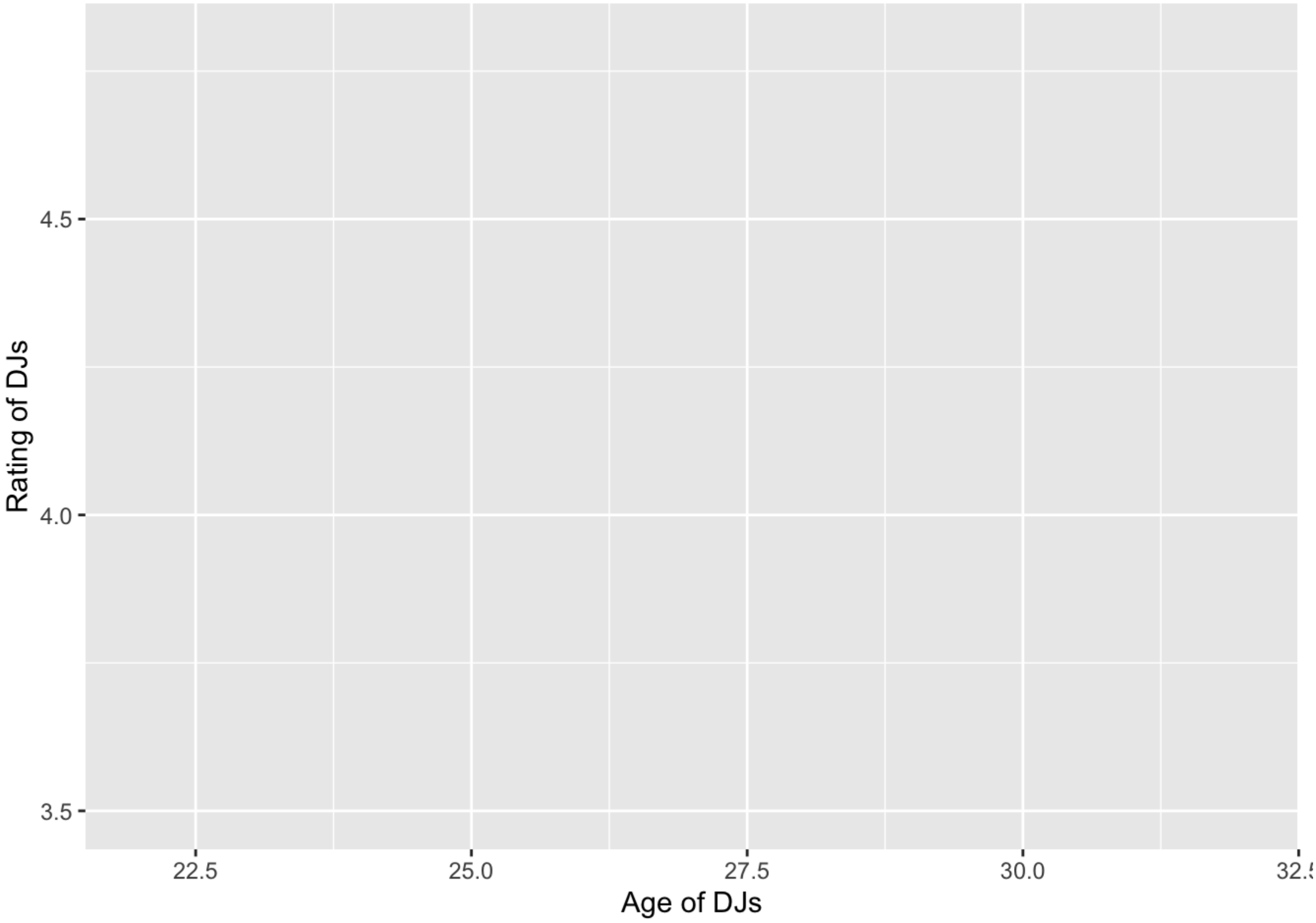


Question 3.2: Label the x-axis as “Age” and the y-axis as “Rating.”

Solution:

```
# complete the code to generate the plot

ggplot(playlist_data) +
aes(x=Age,y=Rating) +
  labs(x="Age of DJs", y="Rating of DJs")
```

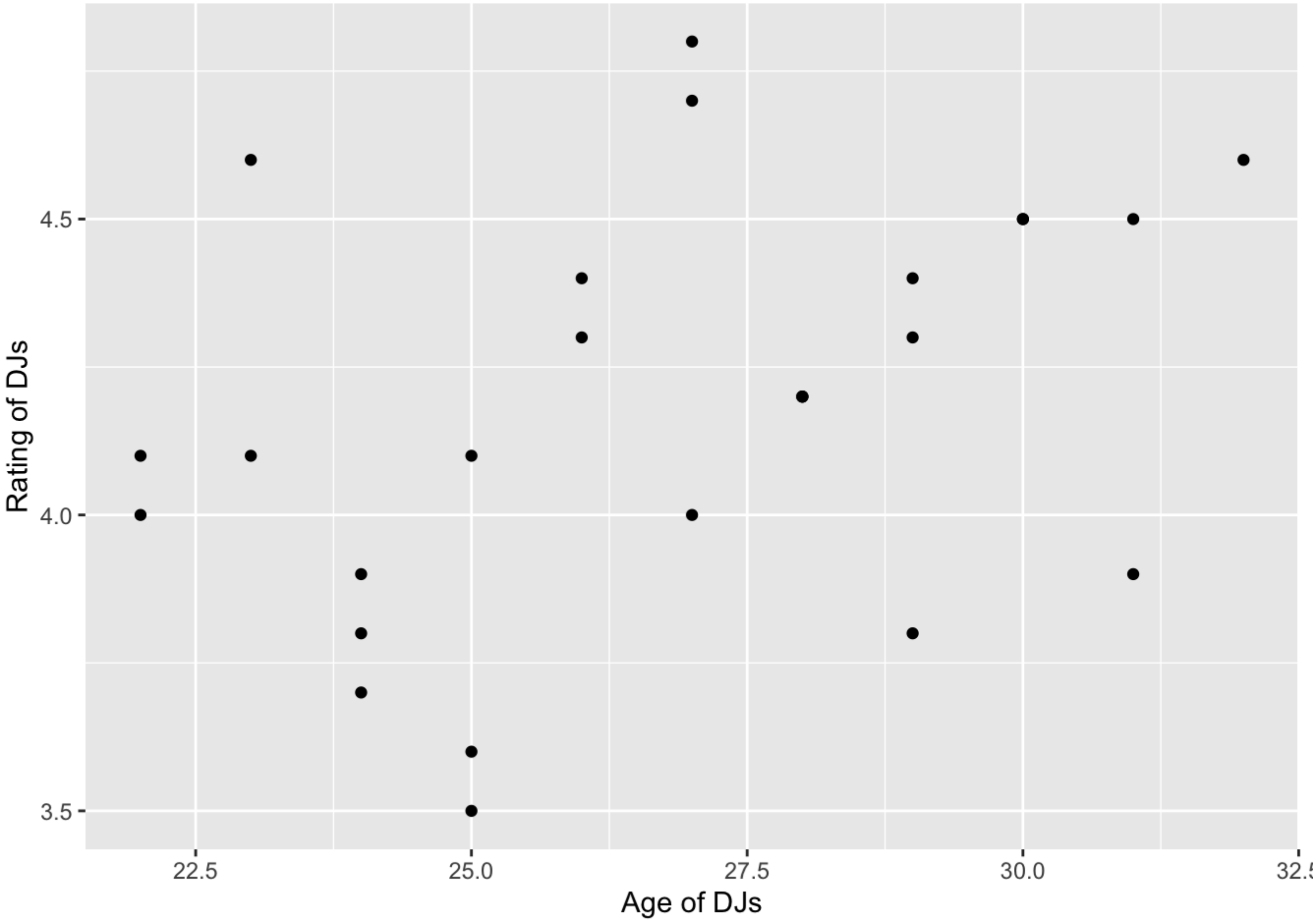


Question 3.3: Represent data using points

Solution:

```
# complete the code to generate the plot

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

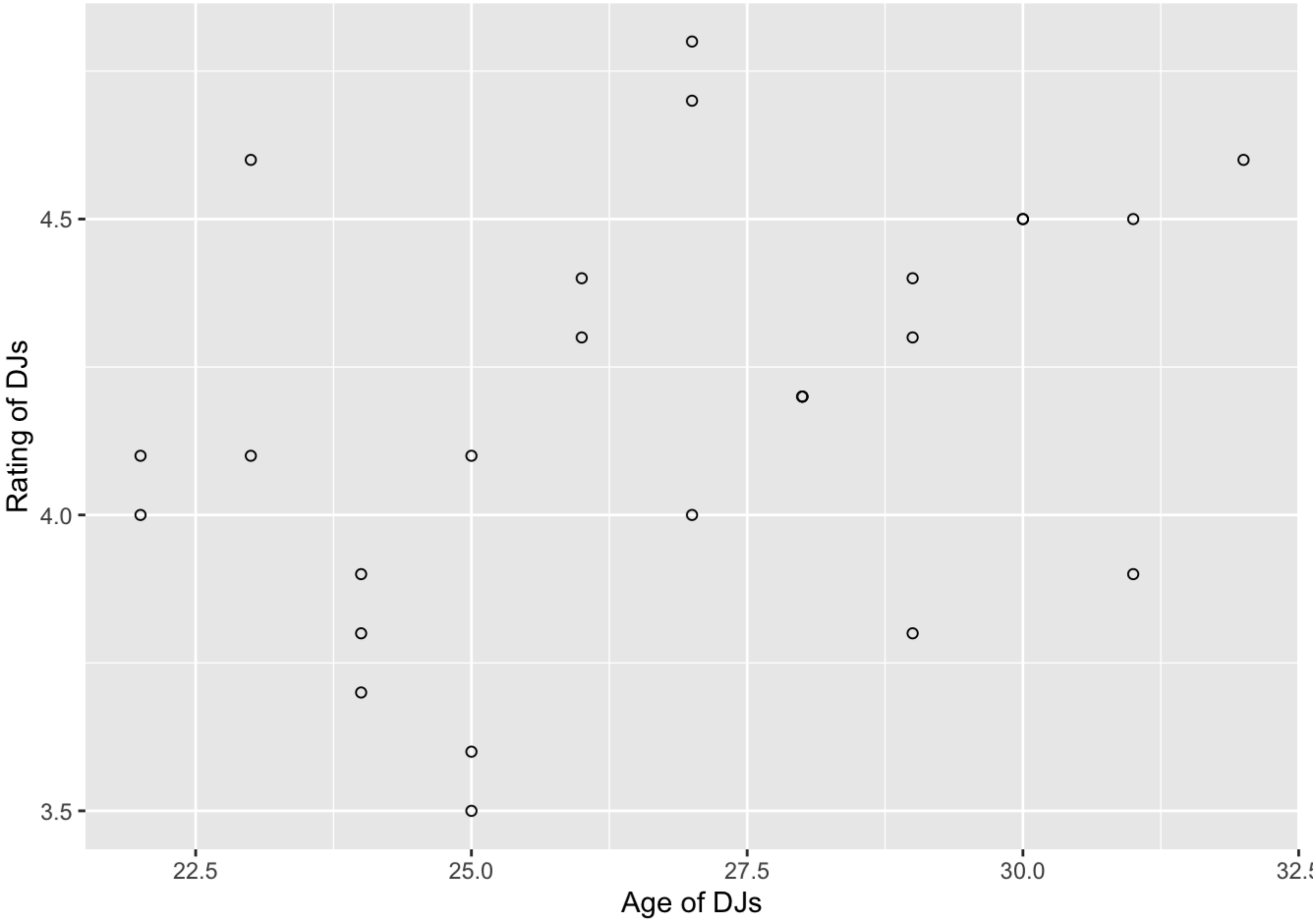


Question 3.4: Can you change the points represented by dots/small circles to any other shape of your liking?

Solution:

```
# complete the code to generate the plot

ggplot(playlist_data) +
aes(x=Age,y=Rating) +
  labs(x="Age of DJs", y="Rating of DJs")+
  geom_point(shape=21) #<-- Could be any of the allowed shapes, use ?geom_point to se
e more information
```

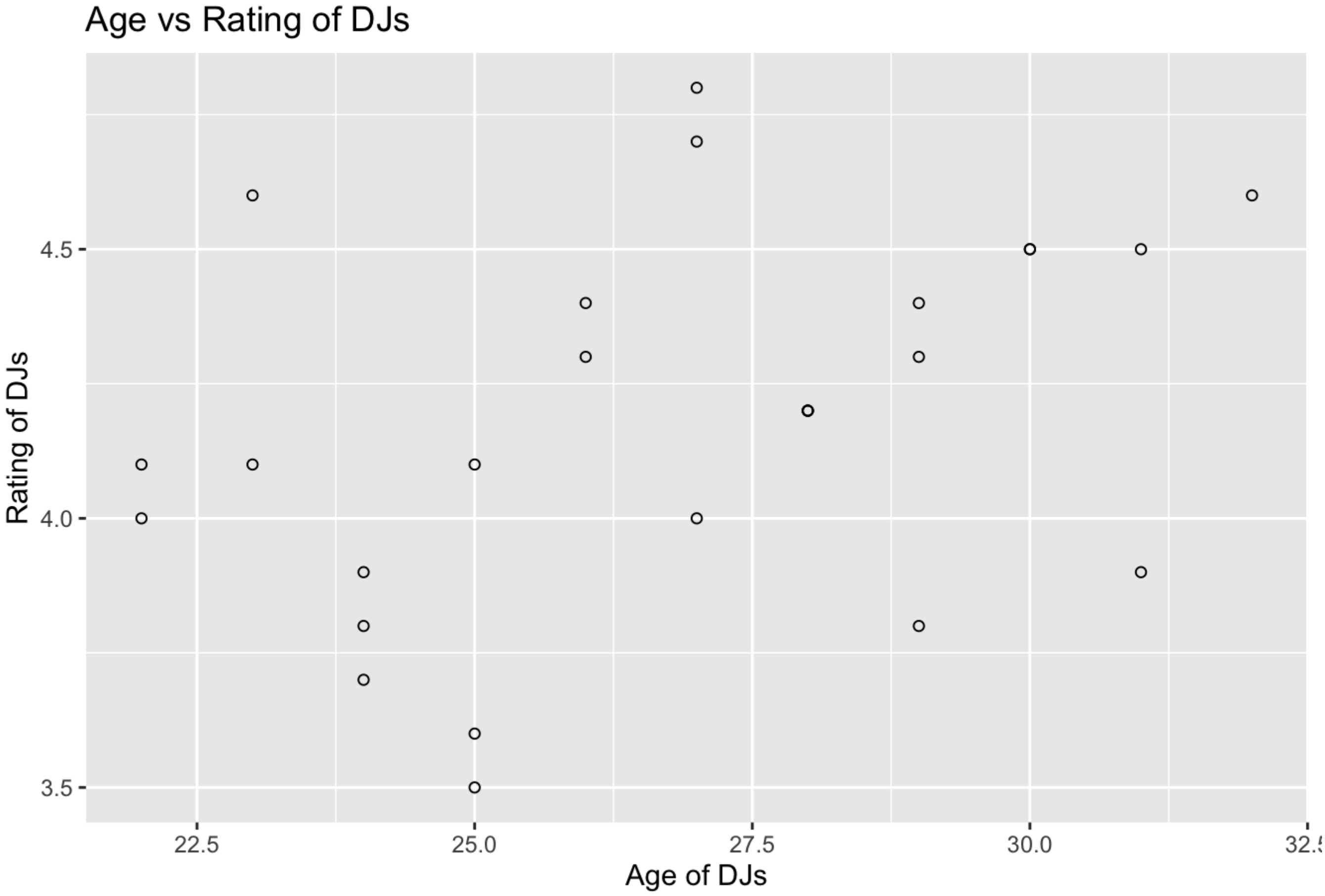


Question 3.5: Insert a suitable title and briefly provide your insights in the caption

Solution:

```
# complete the code to generate the plot

ggplot(playlist_data) +
aes(x=Age,y=Rating) +
  labs(x="Age of DJs", y="Rating of DJs",title="Age vs Rating of DJs",caption="No par
ticular pattern could be established")+ #<--Caption can be their interpretation of th
e plot
  geom_point(shape=21)
```



No particular pattern could be established