AFRAAZ HUSSAIN
20BDS0374
DATA VISUALIZATION AND PRESENTATION
LAB ASSESSMENT 4

QUESTION:

Write an R program to perform sentiment analysis on the 'Shakespear.rda' dataset.

CODE:

```
# Name: Afraaz Hussain
# Admission number: 20BDS0374
# Date: 06.03.2023


library(dplyr)


# Load the dataset
load('shakespeare.rda')

# Pipe the shakespeare dataframe into the next line
# Use count to find out how many titles/types there are
shakespeare %>% count(title, type)

# Load tidyverse and tidytext
library(tidyverse)
library(tidytext)

# #Create an object tidy_shakespeare
# Group by the titles of the plays
# Define a new column line number
# Transform the non-tidy text data to tidy text data
tidy_shakespeare <- shakespeare %>%
  group_by(title) %>%
  mutate(line_number = row_number()) %>%
  unnest_tokens(word, text)

# Pipe the tidy Shakespeare data frame to the next line
# Use count to find out how many times each word is used
tidy_shakespeare %>% count(word, sort = TRUE)

# Sentiment analysis of tidy_shakespeare assign to object
shakespeare_sentiment
# Implement sentiment analysis with the "bing" lexicon
bing <- get_sentiments("bing")
shakespeare_sentiment <- tidy_shakespeare %>%
  inner_join(get_sentiments("bing"), by = "word")

# shakespeare_sentiment
```

AFRAAZ HUSSAIN
20BDS0374
DATA VISUALIZATION AND PRESENTATION
LAB ASSESSMENT 4

```
# Find how many positive/negative words each play has
# shakespeare_sentiment %>% spread(sentiment, n, fill = 0)
shakespeare_sentiment %>%
  group_by(title, sentiment) %>%
  summarise(count = n())

# Tragedy or comedy from tidy_shakespeare  assign to sentiment_counts
# Implement sentiment analysis using the "bing" lexicon
# Count the number of words by title, type, and sentiment
sentiment_counts <- tidy_shakespeare %>%
  inner_join(get_sentiments("bing"), by = "word") %>%
  count(title, type, sentiment, sort = TRUE)

# from sentiment_counts
# Group by the titles of the plays
# Find the total number of words in each play
# Calculate the number of words divided by the total
# Filter the results for only negative sentiment then arrange percentages in
ASC order
sentiment_counts %>%
  group_by(title, type, index) %>%
  summarise(total = sum(n)) %>%
  ungroup() %>%
  mutate(prop = n/total) %>%
  filter(sentiment == "negative") %>%
  arrange(prop)

# Most common positive and negative words and assign to word_could
# Implement sentiment analysis using the "bing" lexicon
# Count by word and sentiment
bing <- get_sentiments("bing")
word_counts <- tidy_shakespeare %>%
  inner_join(bing) %>%
  count(word, sentiment, sort = TRUE)

# extract the top 10 words from word_counts and assign to top_words
# Group by sentiment
# Take the top 10 for each sentiment and ungroup it
# Make word a factor in order of n
top_words <- word_counts %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = factor(word, levels = rev(unique(word))))

# Use aes() to put words on the x-axis and n on the y-axis
```

AFRAAZ HUSSAIN
20BDS0374
DATA VISUALIZATION AND PRESENTATION
LAB ASSESSMENT 4

```
# Make a bar chart with geom_col()
# facet_wrap for sentiments and apply scales  as free
# Move x to y and y to x
#Move x to y and y to x
ggplot(top_words, aes(x = n, y = word, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free") +
  scale_y_discrete(position = "right")
```

OUTPUT:

- Use count to find out how many titles/types there are:

```
# A tibble: 6 x 3
  title                          type        n
  <chr>                          <chr>   <int>
1 A Midsummer Night's Dream      Comedy   3459
2 Hamlet, Prince of Denmark      Tragedy  6776
3 Much Ado about Nothing         Comedy   3799
4 The Merchant of Venice         Comedy   4225
5 The Tragedy of Macbeth         Tragedy  3188
6 The Tragedy of Romeo and Juliet Tragedy  4441
```

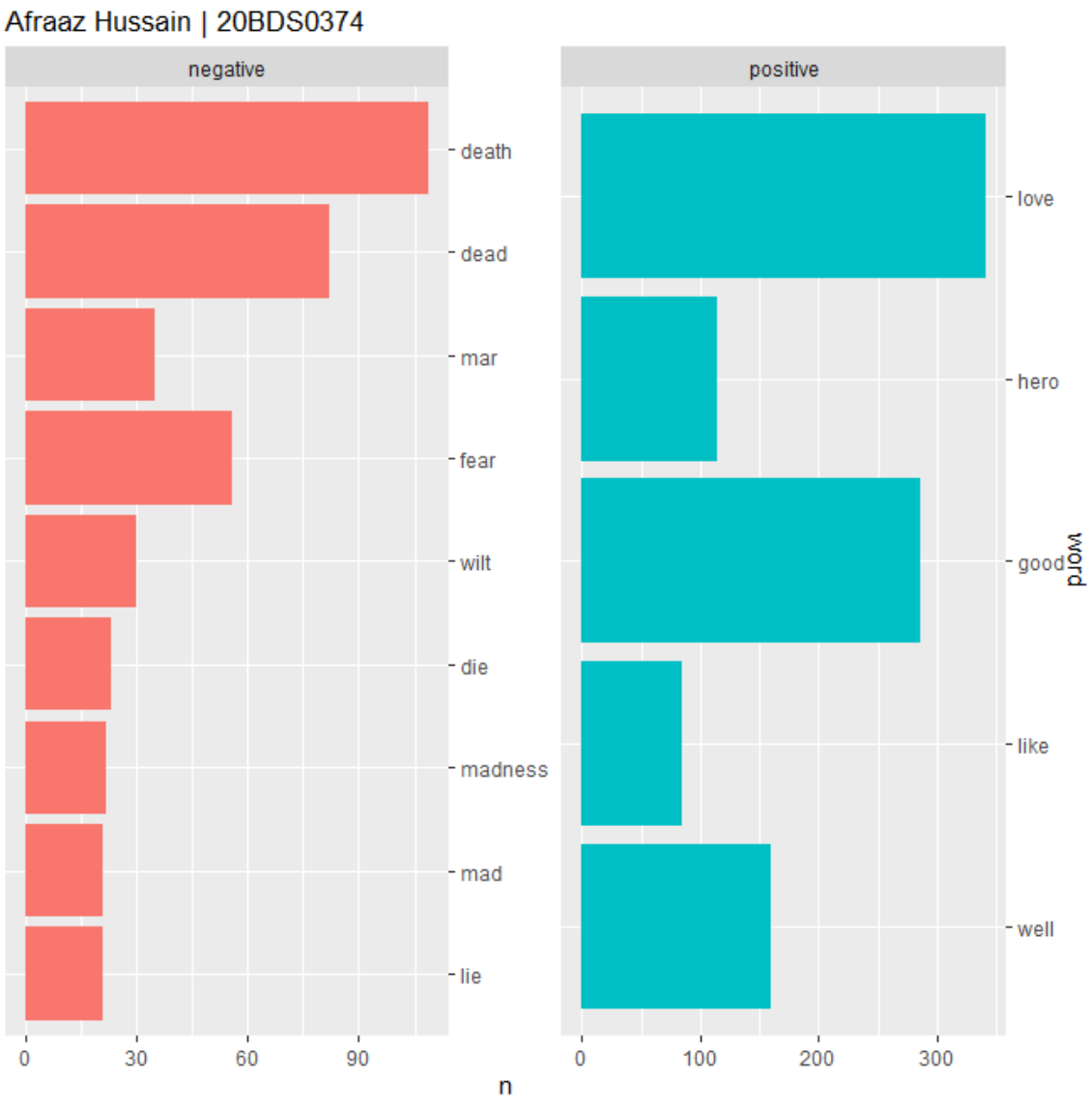
- Use count to find out how many times each word is used:

```
# A tibble: 21,650 x 3
# Groups:   title [6]
   title                          word      n
   <chr>                          <chr> <int>
 1 Hamlet, Prince of Denmark      the    1151
 2 Hamlet, Prince of Denmark      and     983
 3 The Merchant of Venice         the     829
 4 The Tragedy of Macbeth         the     802
 5 Hamlet, Prince of Denmark      to      764
 6 The Tragedy of Romeo and Juliet and     724
 7 Much Ado about Nothing         i       695
 8 The Tragedy of Romeo and Juliet the     687
 9 Hamlet, Prince of Denmark      of      675
10 The Merchant of Venice         i       656
# ... with 21,640 more rows
# i Use `print(n = ...)` to see more rows
```

- Find how many positive/negative words each play has. Also find the total number of words in each play:

```
# Groups:   title [6]
   title                           sentiment count
   <chr>                           <chr>     <int>
 1 A Midsummer Night's Dream       negative    681
 2 A Midsummer Night's Dream       positive    773
 3 Hamlet, Prince of Denmark       negative   1323
 4 Hamlet, Prince of Denmark       positive   1223
 5 Much Ado about Nothing          negative    767
 6 Much Ado about Nothing          positive   1127
 7 The Merchant of Venice          negative    740
 8 The Merchant of Venice          positive    962
 9 The Tragedy of Macbeth          negative    914
10 The Tragedy of Macbeth          positive    749
11 The Tragedy of Romeo and Juliet negative   1235
12 The Tragedy of Romeo and Juliet positive   1090
```

- Use aes() to put words on the x-axis and n on the y-axis:



AFRAAZ HUSSAIN
20BDS0374
DATA VISUALIZATION AND PRESENTATION
LAB ASSESSMENT 4

QUESTION:

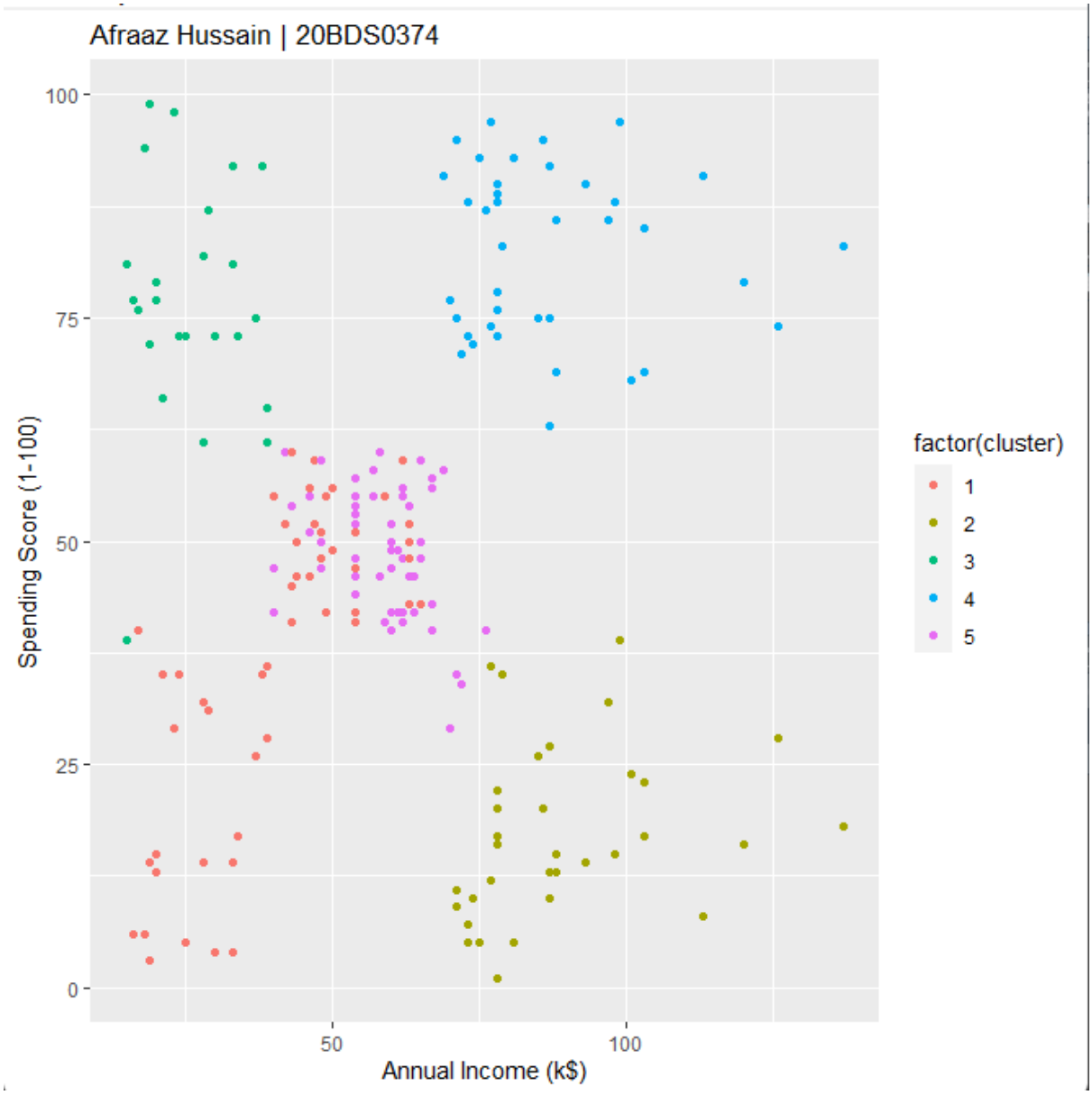
Write an R program to perform K-means clustering on the 'Mall_Customers.csv' dataset.

CODE:

```
# Load the required packages
library(tidyverse)

# Read the data
dataset <- read.csv("Mall_Customers.csv")
# Convert the 'Genre' column from character to factor
dataset$gender <- as.factor(dataset$gender)

# Perform K-means clustering with 5 clusters
set.seed(123)
kmeans_clusters <- kmeans(dataset[, c("age", "annualIncome",
"spendingScore")], centers = 5)

# Add cluster assignments to the original data
dataset$cluster <- kmeans_clusters$cluster

# Plot the clusters
ggplot(dataset, aes(x = `annualIncome`, y = `spendingScore`, color =
factor(cluster))) +
  geom_point() +
  xlab("Annual Income (k$)") +
  ylab("Spending Score (1-100)") +
  ggtitle("Afraaz Hussain | 20BDS0374")
```

AFRAAZ HUSSAIN
20BDS0374
DATA VISUALIZATION AND PRESENTATION
LAB ASSESSMENT 4

OUTPUT:



AFRAAZ HUSSAIN
20BDS0374
DATA VISUALIZATION AND PRESENTATION
LAB ASSESSMENT 4

QUESTION:

Write an R program to Market basket analysis on the given dataset.

CODE:

```
# Load the required packages
library(arules)

# Read the data as a transaction dataset
dataset <- read.transactions("Market_Basket_Optimisation.csv", sep = ",")

# Perform Market Basket Analysis with Apriori algorithm
rules <- apriori(dataset, parameter = list(supp = 0.005, conf = 0.5))

# Inspect the top 10 rules
inspect(head(rules, n = 10))
```

OUTPUT:

```
> # Perform Market Basket Analysis with Apriori algorithm
> rules <- apriori(mydata, parameter = list(supp = 0.005, conf = 0.5))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.5    0.1    1 none FALSE            TRUE       5   0.005      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 37

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[119 item(s), 7501 transaction(s)] done [0.00s].
sorting and recoding items ... [101 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [20 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
> # Inspect the top 10 rules
> inspect(head(rules, n = 10))
     lhs                          rhs             support     confidence coverage    lift     count
[1]  {salmon, spaghetti}       => {mineral water} 0.006799093 0.5049505  0.013464871 2.118363 51
[2]  {olive oil, soup}         => {mineral water} 0.005199307 0.5820896  0.008932142 2.441976 39
[3]  {frozen vegetables, soup} => {mineral water} 0.005065991 0.6333333  0.007998933 2.656954 38
[4]  {ground beef, soup}       => {mineral water} 0.005065991 0.5205479  0.009732036 2.183798 38
[5]  {milk, soup}              => {mineral water} 0.008532196 0.5614035  0.015197974 2.355194 64
[6]  {chocolate, soup}         => {mineral water} 0.005599253 0.5526316  0.010131982 2.318395 42
[7]  {soup, spaghetti}         => {mineral water} 0.007465671 0.5233645  0.014264765 2.195614 56
[8]  {cooking oil, eggs}       => {mineral water} 0.006399147 0.5454545  0.011731769 2.288286 48
[9]  {milk, turkey}            => {mineral water} 0.006132516 0.5411765  0.011331822 2.270338 46
[10] {chicken, chocolate}      => {mineral water} 0.007598987 0.5181818  0.014664711 2.173871 57
```

AFRAAZ HUSSAIN
20BDS0374
DATA VISUALIZATION AND PRESENTATION
LAB ASSESSMENT 4

QUESTION:

Create a dashboard on the 'gapminder' dataset using shiny.

CODE:

```
# Load the required packages
library(shiny)
library(dplyr)
library(ggplot2)
library(gapminder)

# Define the user interface
ui <- fluidPage(
  # Add a title to the dashboard
  titlePanel("Gapminder Dashboard"),

  # Add a sidebar panel with input options
  sidebarLayout(
    sidebarPanel(
      # Add a dropdown menu for selecting the continent
      selectInput("continent", "Select a Continent",
                  choices = c("Asia", "Europe", "Africa", "Americas",
"Oceania")),

      # Add a slider for selecting the year range
      sliderInput("year_range", "Select a Year Range",
                  min = min(gapminder$year), max = max(gapminder$year),
                  value = c(min(gapminder$year), max(gapminder$year)), step =
5),

      # Add a checkbox for showing/hiding the life expectancy trendline
      checkboxInput("trendline", "Show Life Expectancy Trendline", value =
TRUE),

      # Add a text output for displaying your name and admission number
      textOutput("info")
    ),

    # Add the main panel for displaying the plots
    mainPanel(
      # Add a plot for displaying the population vs GDP per capita for the
selected continent
      plotOutput("pop_gdp_plot")
    )
  )
```

```
)

# Define the server
server <- function(input, output) {
  # Create a reactive dataset for the selected continent and year range
  continent_data <- reactive({
    gapminder %>% filter(continent == input$continent & year >=
input$year_range[1] & year <= input$year_range[2])
  })

  # Create a plot for displaying the population vs GDP per capita for the
selected continent and year range
  output$pop_gdp_plot <- renderPlot({
    ggplot(continent_data(), aes(x = gdpPercap, y = pop, size = lifeExp, color
= year)) +
      geom_point(alpha = 0.7) +
      scale_x_log10() +
      scale_size(range = c(2, 20)) +
      labs(x = "GDP per capita", y = "Population", size = "Life Expectancy",
color = "Year") +
      theme_classic() +
      # Add a trendline for the life expectancy
      if(input$trendline) geom_smooth(method = "lm", se = FALSE)
  })

  output$info <- renderText({
    paste("Afraaz Hussain | 20BDS0374")
  })
}

# Run the application
shinyApp(ui = ui, server = server)
```

OUTPUT: