**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

# 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

# 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)) +

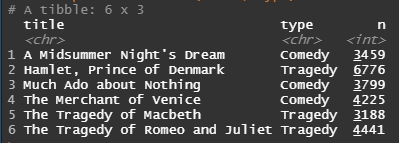
  geom\_col(show.legend = FALSE) +

  facet\_wrap(~sentiment, scales = "free") +

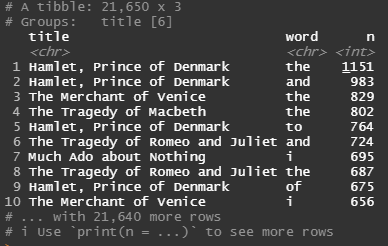
  scale\_y\_discrete(position = "right")

**OUTPUT:**

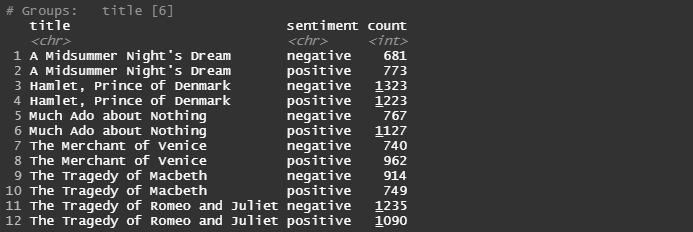
* Use count to find out how many titles/types there are:



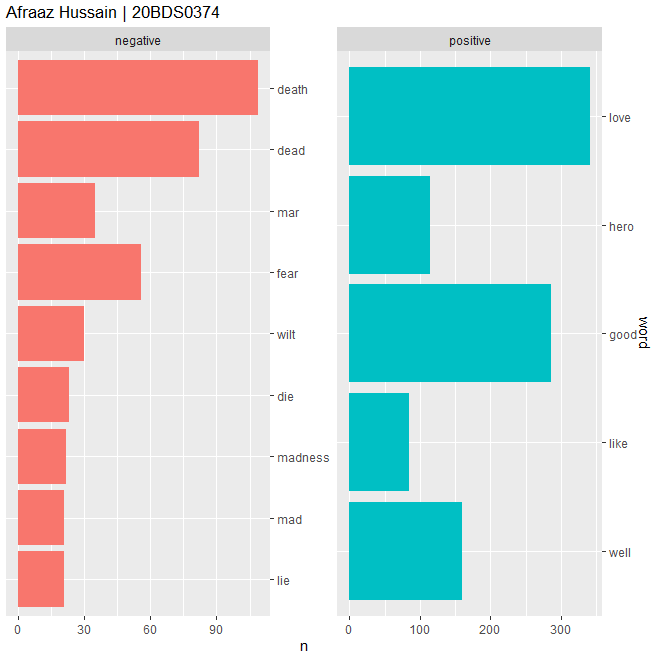
* Use count to find out how many times each word is used:



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



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



**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))) +

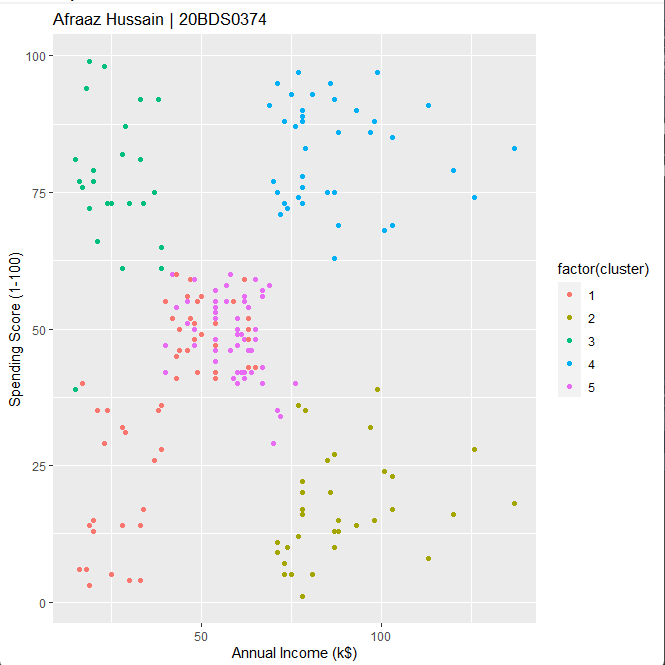
  geom\_point() +

  xlab("Annual Income (k$)") +

  ylab("Spending Score (1-100)") +

  ggtitle("Afraaz Hussain | 20BDS0374")

**OUTPUT:**

****

**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 = ",")

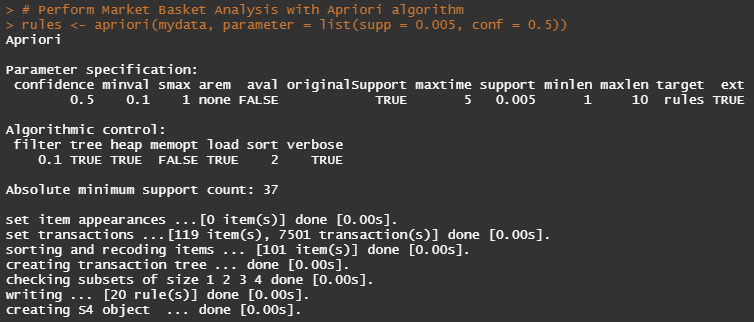
# Perform Market Basket Analysis with Apriori algorithm

rules <- apriori(dataset, parameter = list(supp = 0.005, conf = 0.5))

# Inspect the top 10 rules

inspect(head(rules, n = 10))

**OUTPUT:**



Text

Description automatically generated

**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:**

A screenshot of a computer

Description automatically generated with medium confidence