MONASH University

FIT5145
Foundations of Data Science
Assignment-2

Bhavna Balakrishnan
33954437

Below is the entire RMD Code that will explained as we go on:

```
---
title: "<center>FIT5145: Foundations of Data Science</center>
Name: Bhavna Balakrishnan <br> SID:33954427"
output: html_document
date: "2024-04-26"
---
```

We begin by first loading in necessary libraries for the code to work.Also mentioned in comments is code on how to install a new library:

```
```{r}
#Load all Necessary Libraries:
#Syntax to install any library is for example;
install.packages("plotly")
library(ggplot2)
library(plotly)
library(dplyr)
library(lubridate)
library(tm)
library(tidyr)
library(wordcloud)

```
```

- Ggplot2 is a library that we use to plot graphic visualisations with R
- Plotly is also a library used to make visulaisationsand charts but it helps in making them more interactive.
- Dplr helps with manipulating and transforming data.
- Lubridate is a library package that works with dates and times in R.
- Tm is a text mining package that provides tools for preprocessing of textual data as well as aids in the analysis of text.
- Tidyr as the name suggests,helps with tidying up data.It also helps with the transformation and reshaping of data that prepares data for analysis and visualistaion.
- Wordcloud Librray is essentially used for the creatin of worcloud visualisations.

Next we must be able to actually read the CSV File that was provided to us on Moodle.Prior to running the code,it must be ensured that the file has been set as the working directory and the same location as the rmd file:

```
```{r}
# Read the CSV file into a data frame
ireland_news <- read.csv("ireland_news.csv")
```
```

Once the initialisation of the libraries required and the ability to read the data from the csv file has been done, we can move on to answering the questions per the assignment.

```
## Q1. Sorting Articles by time

```{r}
# Convert publish_date column to standard Date format
ireland_news <- ireland_news %>%
  mutate(publish_date = as.Date(publish_date, format = "%A,
%dth of %B, %Y"))

# Exclude rows with NA values in publish_date column
ireland_news <- ireland_news %>%
  filter(!is.na(publish_date))

# Sort the data by publish_date in ascending order
ireland_news_sorted <- ireland_news %>%
  arrange(publish_date)

# Display 5 most recent records
head(ireland_news_sorted, 5)

# Display the last 5 records of the sorted data
tail(ireland_news_sorted, 5)

```
```

Output:

	publish_date <date>	headline_category <chr>
1611391	2021-06-30	business.commercial-property
1611392	2021-06-30	opinion.letters
1611393	2021-06-30	news.politics.oireachtas
1611394	2021-06-30	business.markets
1611395	2021-06-30	news.world.us

5 Most Recent Records

	publish_date <date>	headline_category <chr>
1	1996-01-02	sport
2	1996-01-02	business
3	1996-01-02	sport
4	1996-01-02	sport
5	1996-01-02	sport

Last 5 Records

From above, we can see both the most recent and the oldest records. The most recent articles are all from 2021 and the oldest are all from 1996.The most recent record happens to be from the category business.commercial-property and the oldest from sport.

Explanations:

The above code makes use of the dplyr library for data manipulation.
First off the line :
ireland_news <- ireland_news %>% mutate(publish_date = as.Date(publish_date, format = "%A, %dth of %B, %Y")) converts the column publish_date from the data set to a standard data format since it is currently something like Wednesday,25th of March,2015.The strptime() also helps with conversion of characters into date format:

- **%A**: Full weekday name (e.g., "Wednesday").
- **%d**: Day of the month as decimal number (01-31).
- **%B**: Full month name (e.g., "March").
- **%Y**: Year with century (e.g., "2015").

After the dates have been formatted appropriately we have to filter out the NA values using the filter() function to ensure we get accurate results in the output.
Once we have been able to decipher the date then we will order the data by using the order() function. To return the first five records we will us ethe head() function and the last five, tail() function and specify that we want only 5 records.

```
## Q2. Unique Headline Categories
```{r}
# Convert headline_category column to lowercase to handle
variations
ireland_news$headline_category <-
tolower(ireland_news$headline_category)

# Count the number of unique headline_category values
num_unique_categories <-
length(unique(ireland_news$headline_category))

# Print the result
cat("Number of unique headline category values:",
num_unique_categories, "\n")


#2 a

# Convert headline_text column to character type
ireland_news$headline_text <-
as.character(ireland_news$headline_text)

# Define the keywords and year range
keywords <- c("ireland", "irish", "us", "usa")
year_range <- 2000:2024

# Construct the regular expression pattern
pattern <- paste0("(\\b(?:", paste(keywords, collapse = "|"),
")\\b.*\\b(?:", paste(year_range, collapse = "|"), ")\\b)")
```
```

```
# Count the number of matches
num_matches <- sum(grepl(pattern,
tolower(ireland_news$headline_text)))

# Print the result
cat("Number of news category articles containing the keywords
'Ireland', 'Irish', 'US', or 'USA' along with year digits from
2000 to 2024 in headline_text:", num_matches, "\n")
```

``

Output

```
Number of unique headline category values: 110
Number of news category articles containing the keywords 'Ireland', 'Irish',
'US', or 'USA' along with year digits from 2000 to 2024 in headline_text: 1506
```

The conclusion we got was that there are 110 unique headline categories and there are 1506 categories with articles containing the words US,USA and Irish in them.

Explanation:

We begin by converting the headline category into lowercase in order to standardise and variation among the capitalizations of words. After that the use of the unique() function to determine the unique values in the dataset. We then define a vector named keywords with the words “Ireland”, “US” and “USA” to be looked up and matched in the headlines text column after ensuring that all text is treaded as character data by using the as.character() function.. We will also define a range in the date from 2000-2004.Finally we use the paste() function and define all our required criteria to return desired results. The cat() function will print out the output.

Q.3 Top 10 Monday Headlines

```{r}
# Convert publish_date column to Date format
ireland_news$publish_date <-
as.Date(ireland_news$publish_date, format = "%A, %dth of %B,
%Y")

# Filter the dataset to include only articles published on
Mondays
monday_articles <-
ireland_news[format(ireland_news$publish_date, "%A") ==
"Monday", ]

# Count the number of articles for each headline category
category_counts <- table(monday_articles$headline_category)

# Identify the top 10 headline categories with the largest
number of articles
```

```
top_10_categories <- head(sort(category_counts, decreasing =
TRUE), 10)

# Draw a chart showing the total number of articles for the
top 10 headline categories for each year

# Filter the dataset to include only the top 10 headline
categories
top_categories_data <-
monday_articles[monday_articles$headline_category %in%
names(top_10_categories), ]

# Create a data frame with the counts of articles for each
headline category and year
category_year_counts <- with(top_categories_data,
table(headline_category, format(publish_date, "%Y")))

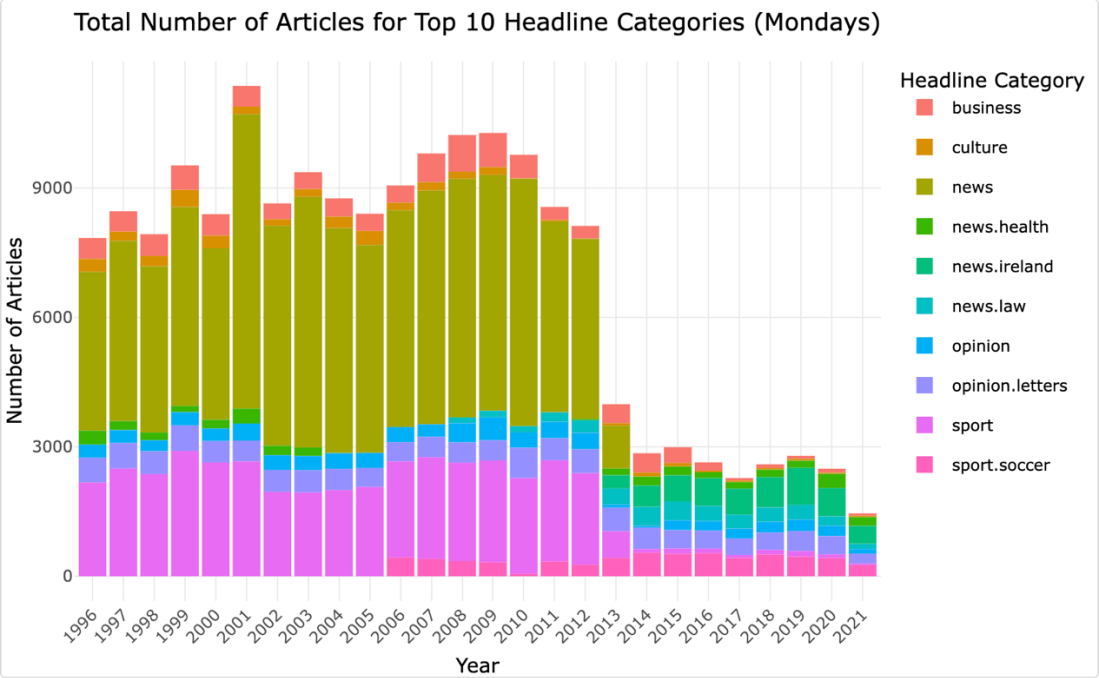
# Convert the data frame to long format for plotting
category_year_counts_long <-
as.data.frame(category_year_counts)
colnames(category_year_counts_long) <- c("headline_category",
"year", "count")

# Convert the data to a plotly object
p <- ggplot(category_year_counts_long, aes(x = year, y =
count, fill = headline_category)) +
  geom_bar(stat = "identity", position = "stack") +
  labs(title = "Total Number of Articles for Top 10 Headline
Categories (Mondays)",
       x = "Year", y = "Number of Articles") +
  theme_minimal() +
  theme(legend.position = "top", axis.text.x =
element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5)) +
  scale_fill_discrete(name = "Headline Category")


# Convert to a plotly object and add tooltips
p <- ggplotly(p, tooltip = c("year", "count"))

# Display the plot
p
```

Output:



From above graph, we can conclude that from years 1996-2013 the biggest chunk of headline categories was news and the least has been either business or culture.After 2013 sport.soccer has more articles and news.ireland starts dominating the headline categories.

Explanation

As we have done previously, the first step in this code is to standardise the date in the publish_date column by specifying the current format and using the as.Date() function.Once we finish off with this initial step we will start filtering out the Monday articles using the format() function to extract the day of the week from the publish_date column.Next we use the table() function to count the number of articles and used the head() function to identify the top 10 categories.We then create another table category_year_counts that utilises the with() function to determine the yearly counts of articles within a given headline category.We then convert this table into a dataframe and assign relevant column names to plot out the information in a stacked bar graph using ggpot library. We also use plotly package to make the chart more interactive.

```
## Q.4 Headline Information and Statistics

```{r}
# Compute the total number of articles for each headline
category and news provider
article_counts <- ireland_news %>%
  group_by(headline_category, news_provider) %>%
  summarise(total_articles = n())

# Print the computed total number of articles for each
headline category and news provider
print(article_counts)

# Compute and display the statistical information (Min, Max,
and Mean) for each provider in a single command
article_counts %>%
```
```

```
  group_by(news_provider) %>%
  summarise(across(total_articles, list(Min = min, Max = max,
Mean = mean)))
```

Output

headline_category <chr>	news_provider <chr>	total_articles <int>
NEWS	Irish Examiner	1
OPINION.LETTERS	RTE News	1
Opinion.Letters	Irish Independent	1
business	Irish Examiner	27902
business	Irish Independent	5516
business	Irish Times	38942
business	RTE News	22170
business	TheJournal.ie	16905
business.MARKETS	Irish Examiner	1
business.agribusiness-and-food	Irish Examiner	971

1-10 of 536 rows Previous 1 2 3 4 5 6 ... 54 Next

Snippet of Article Counts

news_provider <chr>	total_articles_Min <int>	total_articles_Max <int>	total_articles_Mean <dbl>
Irish Examiner	1	144569	3797.8962
Irish Independent	1	29134	774.6442
Irish Times	1	203532	5378.6000
RTE News	1	116154	2979.8056
TheJournal.ie	19	86776	2346.3107
NA	1	2	1.3000

Statistics by Provider

Thejournal.ie has most minimum and Irish Independent has the biggest mean number of articles.

Explanation

We use the dplyr library to help with data manipulation and use the group_by() function to to group the calculated total number of articles for every combination of headline_category and news_provider and then we summarise the results. The single command to display summary statistics is the summarise() function.

```
## Q.5 Article Breakdown and averages

```{r}

# Convert publish_date to a Date object
ireland_news$publish_date <-
as.Date(ireland_news$publish_date, format = "%A, %dth of %B,
%Y")

# Filter out rows with NA values in publish_date
ireland_news <- na.omit(ireland_news, cols = "publish_date")
```
```

```
# Compute the total number of articles for each headline
category, news provider, and day of the week
article_counts <- ireland_news %>%
  group_by(headline_category, news_provider, day_of_week =
format(publish_date, "%A")) %>%
  summarise(total_articles = n())

# Compute the average number of articles for each news
provider and the day of the week
average_articles <- article_counts %>%
  group_by(news_provider, day_of_week) %>%
  summarise(average_articles = mean(total_articles, na.rm =
TRUE))

# Find the day of the week with the highest average number of
articles for each provider
top_day <- average_articles %>%
  group_by(news_provider) %>%
  top_n(1, average_articles)

# Print the results in the specified table format
print(top_day)
```

Output

news_provider <chr>	day_of_week <chr>	average_articles <dbl>
Irish Examiner	Saturday	696.4316
Irish Independent	Friday	146.6146
Irish Times	Saturday	962.4021
RTE News	Friday	549.7228
TheJournal.ie	Friday	418.8400

We observe that Irish Times has the maximum average articles and that Irish independent has the least.

Explanation

As previously done, the first step is to first standardise the date format next to remove any NA values we use na.omit() function to remove NA values in the publish_date column. Once this is done, the data gets grouped by category,provider and day using group_by() function.Then the data is summarised using summarise() to compute the total number of articles in each group. We calculate the average by using mean() and ensure we don't include NA values by using na.rm= TRUE.Finally we again group by provider and use top() to give us the highest average by provider.The result is displayed using Print().We mainly us ethe dplyr library here.

```
## Q.6 Assigning Periods
```{r}

# Convert publish_date to a Date object
```
```

```
ireland_news$publish_date <-
as.Date(ireland_news$publish_date, format = "%A, %dth of %B,
%Y")

# Filter the data for the years 2019 and 2020
filtered_data <- ireland_news %>%
  filter(year(publish_date) %in% c(2019, 2020))

# Add a new column named "Period" based on the publish_date
value
filtered_data <- filtered_data %>%
  mutate(Period = case_when(
    publish_date >= as.Date("2019-01-01") & publish_date <=
as.Date("2019-03-31") ~ "Period 1",
    publish_date >= as.Date("2019-04-01") & publish_date <=
as.Date("2019-06-30") ~ "Period 2",
    publish_date >= as.Date("2019-07-01") & publish_date <=
as.Date("2019-09-30") ~ "Period 3",
    publish_date >= as.Date("2019-10-01") & publish_date <=
as.Date("2019-12-31") ~ "Period 4",
    publish_date >= as.Date("2020-01-01") & publish_date <=
as.Date("2020-03-31") ~ "Period 5",
    publish_date >= as.Date("2020-04-01") & publish_date <=
as.Date("2020-06-30") ~ "Period 6",
    publish_date >= as.Date("2020-07-01") & publish_date <=
as.Date("2020-09-30") ~ "Period 7",
    publish_date >= as.Date("2020-10-01") & publish_date <=
as.Date("2020-12-31") ~ "Period 8",
    TRUE ~ "Other"
  ))

# Convert publish_date to a Date object
ireland_news$publish_date <-
as.Date(ireland_news$publish_date, format = "%A, %dth of %B,
%Y")

# Filter the data for the years 2019 and 2020
filtered_data <- ireland_news %>%
  filter(year(publish_date) %in% c(2019, 2020))

# Add a new column named "Period" based on the publish_date
value
filtered_data <- filtered_data %>%
  mutate(Period = case_when(
    publish_date >= as.Date("2019-01-01") & publish_date <=
as.Date("2019-03-31") ~ "Period 1",
    publish_date >= as.Date("2019-04-01") & publish_date <=
as.Date("2019-06-30") ~ "Period 2",
    publish_date >= as.Date("2019-07-01") & publish_date <=
as.Date("2019-09-30") ~ "Period 3",
```

```
    publish_date >= as.Date("2019-10-01") & publish_date <=
as.Date("2019-12-31") ~ "Period 4",
    publish_date >= as.Date("2020-01-01") & publish_date <=
as.Date("2020-03-31") ~ "Period 5",
    publish_date >= as.Date("2020-04-01") & publish_date <=
as.Date("2020-06-30") ~ "Period 6",
    publish_date >= as.Date("2020-07-01") & publish_date <=
as.Date("2020-09-30") ~ "Period 7",
    publish_date >= as.Date("2020-10-01") & publish_date <=
as.Date("2020-12-31") ~ "Period 8",
    TRUE ~ "Other"
  ))

# View the resulting dataset with the new "Period" column
#print(filtered_data)

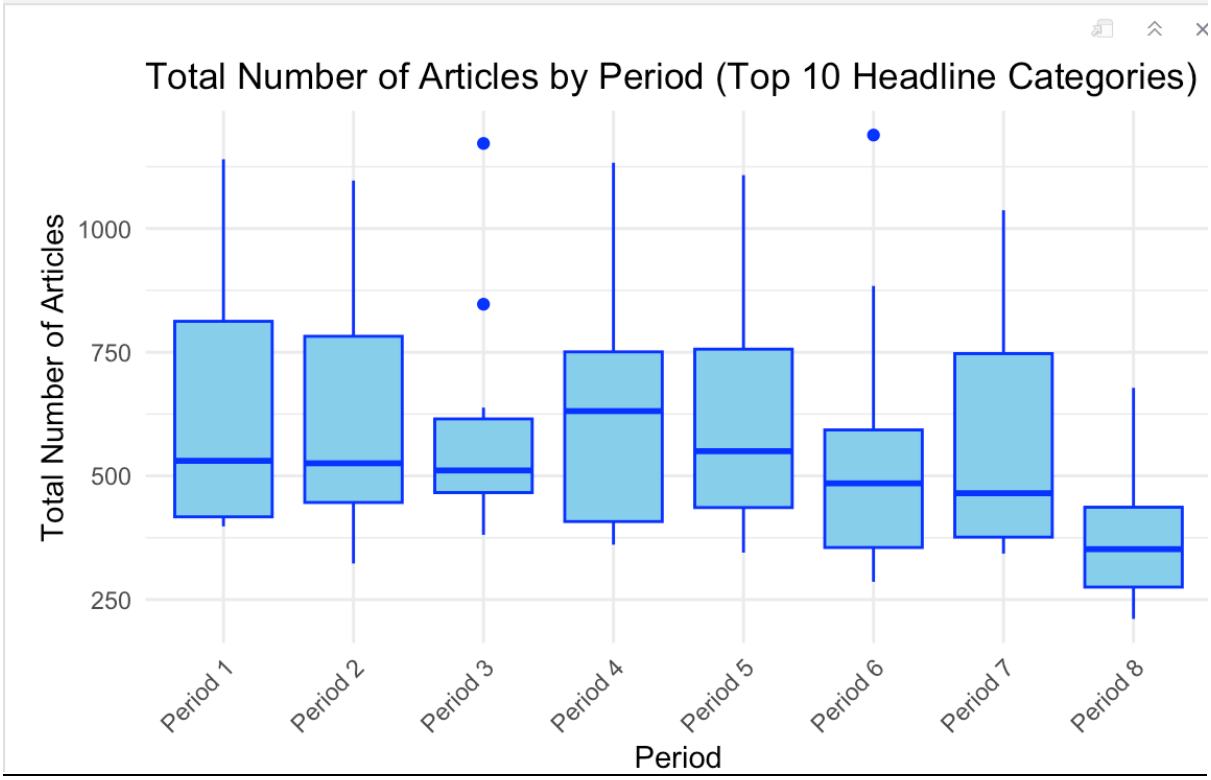

# Step 1: Compute the total number of articles for each
headline category and period
article_counts <- filtered_data %>%
  group_by(headline_category, Period) %>%
  summarise(total_articles = n())

# Step 2: Identify the top ten headline categories
top_categories <- article_counts %>%
  group_by(headline_category) %>%
  summarise(total_articles = sum(total_articles)) %>%
  top_n(10, total_articles) %>%
  select(headline_category)

# Step 3: Filter the data to include only the top ten headline
categories
filtered_article_counts <- article_counts %>%
  filter(headline_category %in%
top_categories$headline_category)

# Step 4: Create a boxplot
ggplot(filtered_article_counts, aes(x = Period, y =
total_articles)) +
  geom_boxplot(fill = "skyblue", color = "blue") +
  labs(title = "Total Number of Articles by Period (Top 10
Headline Categories)",
       x = "Period", y = "Total Number of Articles") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

Output



From above, we can conclude that period 4 has the maximum median of articles and that period 8 has the least.

Explanation

First couple of steps are same which is standardising the date format and then removing NA values in the publish_date column. To assign a period as specified in the assignment requirements we use the mutate() function and filter the data per assigned period.

```
## Q.7 Sample Data and Charts

```{r}
#To ensure reproducibility
set.seed(100)

# Step 1: Sample 1% of the data
sampled_data <- ireland_news %>% sample_frac(0.01)

# Step 2: Perform text pre-processing
corpus <- Corpus(VectorSource(sampled_data$headline_text))
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, removeWords, stopwords("en"))
corpus <- tm_map(corpus, stripWhitespace)

# Step 3: Create a document-term matrix
dtm <- DocumentTermMatrix(corpus)
#print(dtm)
```
```

```
# Step 4: Generate a plot showing the top 10 most frequent
words
freq_words <- colSums(as.matrix(dtm))
top_words <- sort(freq_words, decreasing = TRUE)[1:10]

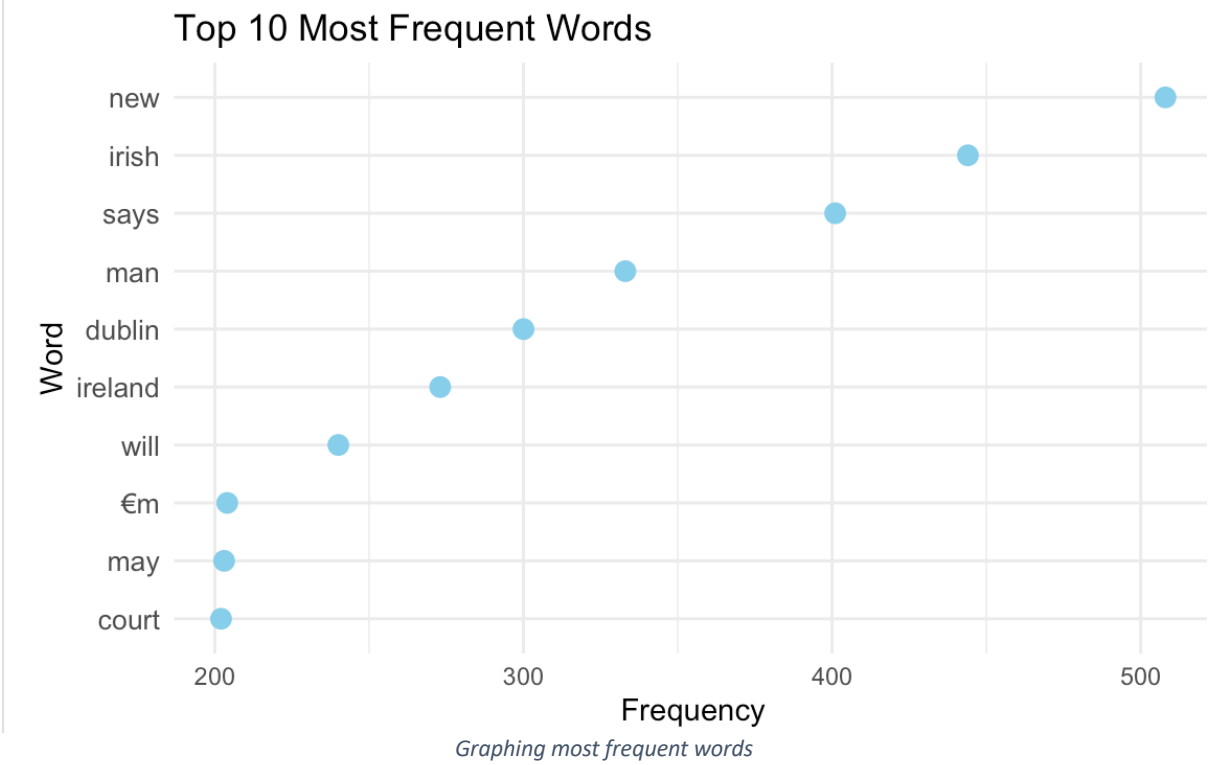
# Create a data frame for plotting
word_freq_df <- data.frame(word = names(top_words), frequency
= top_words)

# Plot
ggplot(word_freq_df, aes(x = frequency, y = reorder(word,
frequency))) +
  geom_point(size = 3, color = "skyblue") +
  labs(title = "Top 10 Most Frequent Words",
       x = "Frequency", y = "Word") +
  theme_minimal() +
  theme(axis.text.y = element_text(size = 10))

#install.packages("wordcloud")

# Create a word cloud
wordcloud(words = word_freq_df$word, freq =
word_freq_df$frequency,
          min.freq = 1, max.words = 10, random.order = FALSE,
          colors = brewer.pal(8, "Dark2"))
```

Output



Graphing most frequent words

Word Cloud

The most frequent word in this sample is “New” and least is “court”.

Explanation

The first thing in this code is that we set a seed to ensure that the results can be reproduced correctly. Next we set a sample of 1% of the data using the sample_frac() function. We then have to start preprocessing the data using the tm package and using tm_map() function to transform text and remove numbers,punctuations,stopwords etc.After prepocessing is done, a document term matrix that I named dtm is made out of the preprocessed data named corpus to represent the frequency of the words in each headline.Using colSum() and sort() functions we identify the most frequently occurring words and plot them using ggplot. We use the wordclous library to make a wordcloud and input parameters like minimum frequency,maximum words etc.

```
## Q.8 Analysing News Categories and trends over the Years

```{r}

# Define the grouping rules
ireland_news <- ireland_news %>%
  mutate(grouped_category = case_when(
    grepl("sport", headline_category, ignore.case = TRUE) ~
"Sports",
    grepl("business|business_economy", headline_category,
ignore.case = TRUE) ~ "Business",
    grepl("culture|culture_books", headline_category,
ignore.case = TRUE) ~ "Culture",
    grepl("lifestyle|lifestyle_fashion", headline_category,
ignore.case = TRUE) ~ "Fashion.News",
```
```

```
    grepl("news|news_ireland|newses|NEWS", headline_category,
ignore.case = TRUE) ~ "News",
    grepl("opinion", headline_category, ignore.case = TRUE) ~
"Opinion",
    TRUE ~ headline_category  # Default to original category
if no match
  ))

# Count the frequency of each grouped category
category_counts <- table(ireland_news$grouped_category)

# Plot the pie chart
pie_chart <- ggplot(as.data.frame(category_counts), aes(x =
"", y = Freq, fill = Var1)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y", start = 0) +
  labs(title = "Pie Chart of Grouped Headline Categories",fill
="Category") +
  theme_void() +
  theme(legend.position = "right")

# Display the pie chart
print(pie_chart)

#now making a line chart

# Define custom color palette
custom_colors <- c("Sports" = "#1f77b4",    # Blue
                   "Business" = "#ff7f0e",  # Orange
                   "Culture" = "#2ca02c",   # Green
                   "Fashion.News" = "#d62728",  # Red
                   "News" = "#9467bd",      # Purple
                   "Opinion" = "#8c564b")   # Brown

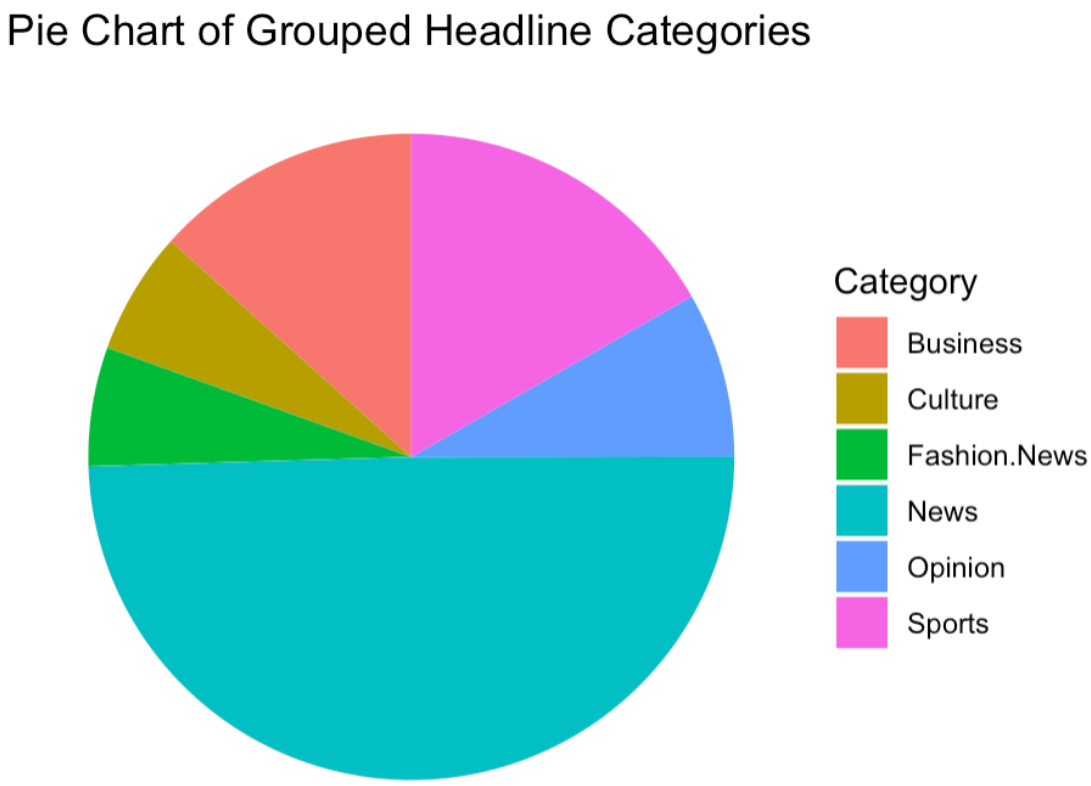
# Filter and group the data by category and year
category_year_counts <- ireland_news %>%
  mutate(year = as.numeric(format(as.Date(publish_date),
"%Y"))) %>%
  filter(grouped_category %in% c("Sports", "Business",
"Culture", "Fashion.News", "News", "Opinion")) %>%
  group_by(grouped_category, year) %>%
  summarise(article_count = n())

# Plot the line graph with custom colors
line_plot <- ggplot(category_year_counts, aes(x = year, y =
article_count, color = grouped_category)) +
  geom_line() +
  scale_color_manual(values = custom_colors) +  # Set custom
colors
  labs(title = "Number of Articles in Each Category Over the
Years",
```

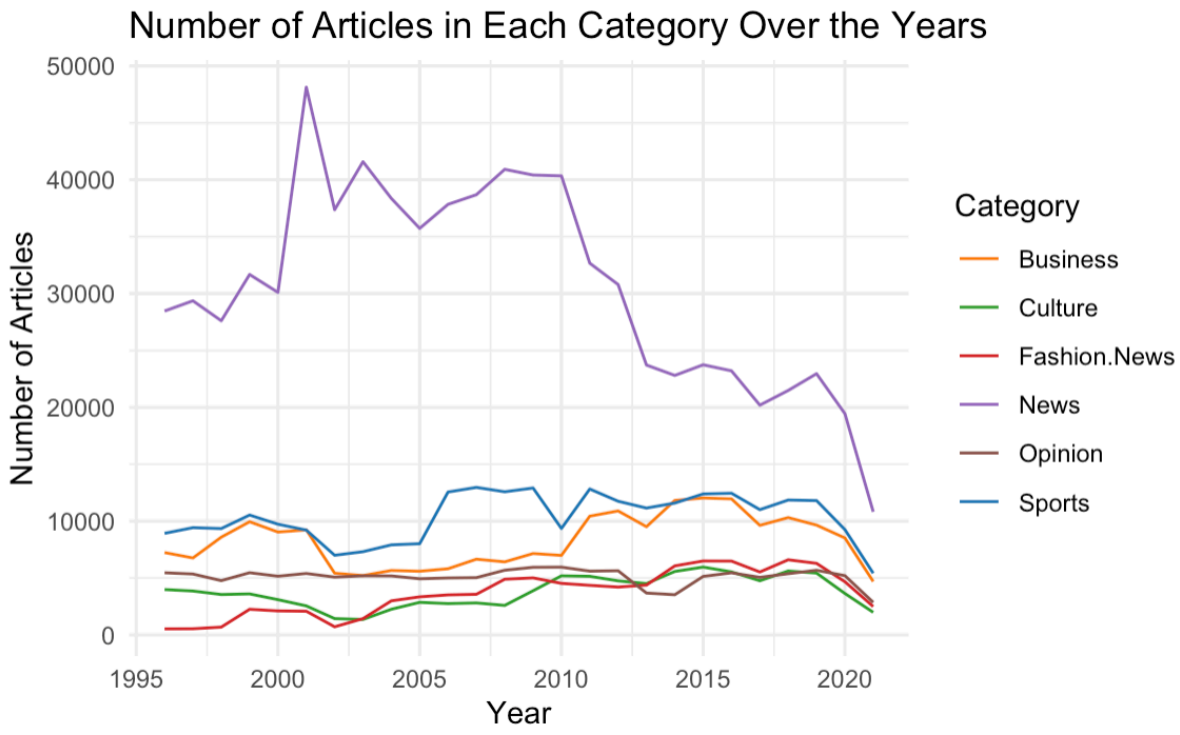
```
        x = "Year", y = "Number of Articles",
        color = "Category") +
  theme_minimal()

# Display the line graph
print(line_plot)
```

Output



Pie Chart depicting various categories



Trends of articles over the years by category

News takes up the most among the categories and from the trends we see an overall decline across all categories in 2020.The top two here happen to be news and sports.

Explanation

This is a combination of Q3 and Q7 where we are trying to show the breakup of the bulk of articles by categories and then observing the trends of the categories over the years. To do this we start by defining grouping rules to make sure all similar categories are grouped together. We ensure that the titles are together by using mutate() function. We then create a frequency table to compute the frequencies and create two charts one Pie Chart to show the category break up and a line chart to see the trends by using ggplot package.