Challenge-4

Tang Ching Xian

04 SEPT 2023

```
knitr::opts_chunk$set(echo = TRUE)
```

Questions

Load the “CommQuest2023.csv” dataset using the `read_csv()` command and assign it to a variable named “comm_data.”

```
# Load Tidyverse
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   3.4.3     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.2
## ── 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 conflicts to
become errors
```

```
# Read the CSV file and assign it to the variable "comm_data"
comm_data <- read.csv("CommQuest2023_Larger.csv")
comm_data_org <- read.csv("CommQuest2023_Larger.csv")
```

Question-1: Communication Chronicles

Using the select command, create a new dataframe containing only the “date,” “channel,” and “message” columns from the “comm_data” dataset.

Solution:

```
# Create a new dataframe with selected columns
new_df1<- comm_data%>% select(date,channel,message)
# Overview of new dataframe
glimpse(new_df1)
```

```
## Rows: 1,000
## Columns: 3
## $ date    <chr> "2023-08-11", "2023-08-11", "2023-08-11", "2023-08-18", "2023-…
## $ channel <chr> "Twitter", "Email", "Slack", "Email", "Slack", "Email", "Twitt…
## $ message <chr> "Fun weekend!", "Hello everyone!", "Hello everyone!", "Fun wee…
```

Question-2: Channel Selection

Use the filter command to create a new dataframe that includes messages sent through the "Twitter" channel on August 2nd.

Solution:

```
# Create new dataframe with filtered data
new_df2 <- comm_data %>%filter(channel == "Twitter", date == "2023-08-02")%>%select(channe
l,date,message)
glimpse(new_df2)
```

Question-3: Chronological Order

Utilizing the arrange command, arrange the "comm_data" dataframe in ascending order based on the "date" column.

Solution:

```
#Arrange the dataframe in ascending order based on the "date" column
arranged_data <- comm_data %>%arrange(date)
print(arranged_data)
```

Question-4: Distinct Discovery

Apply the distinct command to find the unique senders in the "comm_data" dataframe.

Solution:

```
#Use distinct() to find unique senders
unique_senders <- comm_data %>%distinct(sender)%>%select(sender)
print(unique_senders)
```

```
##          sender
## 1  dave@example
## 2   @bob_tweets
## 3   @frank_chat
## 4  @erin_tweets
## 5 alice@example
## 6   carol_slack
```

Question-5: Sender Stats

Employ the count and group_by commands to generate a summary table that shows the count of messages sent by each sender in the “comm_data” dataframe.

Solution:

```
# Group by sender and count messages
summary_table <- comm_data %>%group_by(sender) %>%summarise(count = n())
print(summary_table)
```

```
## # A tibble: 6 × 2
##   sender         count
##   <chr>          <int>
## 1 @bob_tweets      179
## 2 @erin_tweets     171
## 3 @frank_chat      174
## 4 alice@example    180
## 5 carol_slack      141
## 6 dave@example     155
```

```
#Another method just using count()
##summary_table2 <- comm_data %>%group_by(sender)%>%count(sender)
#Break down of each message and count
##summary_table2 <- comm_data %>%group_by(sender)%>%*count(message)*
```

Question-6: Channel Chatter Insights

Using the group_by and count commands, create a summary table that displays the count of messages sent through each communication channel in the “comm_data” dataframe.

Solution:

```
# Group by channel and count messages
summary_table2 <- comm_data %>%group_by(channel) %>%summarise(count = n())
print(summary_table2)
```

```
## # A tibble: 3 × 2
##   channel count
##   <chr>   <int>
## 1 Email     331
## 2 Slack     320
## 3 Twitter   349
```

```
#Another method
##summary_table2 <- comm_data %>%group_by(channel) %>%count(channel)
```

Question-7: Positive Pioneers

Utilize the filter, select, and arrange commands to identify the top three senders with the highest average positive sentiment scores. Display their usernames and corresponding sentiment averages.

Solution:

```
# Filter for positive sentiment scores
positive_data <- comm_data %>%
  filter(sentiment> 0)

# Group by sender and calculate the average sentiment score
sender_avg_sentiment <- positive_data %>%
  group_by(sender) %>%
  summarise(avg_sentiment=mean(sentiment))

# Arrange senders in descending order of average sentiment score
top_senders <- sender_avg_sentiment %>%
  arrange(desc(avg_sentiment))

# Select the top three senders
top_three_senders <- top_senders %>%
  slice(1:3)

# Display the top three senders and their corresponding sentiment averages
print(top_three_senders)
```

```
## # A tibble: 3 × 2
##   sender        avg_sentiment
##   <chr>                 <dbl>
## 1 dave@example          0.541
## 2 @frank_chat           0.528
## 3 alice@example         0.493
```

Question-8: Message Mood Over Time

With the group_by, summarise, and arrange commands, calculate the average sentiment score for each day in the “comm_data” dataframe.

Solution:

```
# Group by date and calculate the average sentiment score for each day
daily_avg_sentiment <- comm_data %>%
  group_by(date) %>%
  summarise(avg_sentiment = mean(sentiment))

# Arrange the results in ascending order of date
daily_avg_sentiment <- daily_avg_sentiment %>%
  arrange(date)

# Display the dataframe with daily average sentiment scores
print(daily_avg_sentiment)
```

```
## # A tibble: 20 × 2
##    date       avg_sentiment
##    <chr>              <dbl>
##  1 2023-08-01       -0.0616
##  2 2023-08-02        0.136
##  3 2023-08-03        0.107
##  4 2023-08-04       -0.0510
##  5 2023-08-05        0.193
##  6 2023-08-06       -0.0144
##  7 2023-08-07        0.0364
##  8 2023-08-08        0.0666
##  9 2023-08-09        0.0997
## 10 2023-08-10       -0.0254
## 11 2023-08-11       -0.0340
## 12 2023-08-12        0.0668
## 13 2023-08-13       -0.0604
## 14 2023-08-14       -0.0692
## 15 2023-08-15        0.0617
## 16 2023-08-16       -0.0220
## 17 2023-08-17       -0.0191
## 18 2023-08-18       -0.0760
## 19 2023-08-19        0.0551
## 20 2023-08-20        0.0608
```

```
#another method
##daily<-comm_data%>%group_by(date)%>%summarise(mean(sentiment))%>%arrange(date)
```

Question-9: Selective Sentiments

Use the filter and select commands to extract messages with a negative sentiment score (less than 0) and create a new dataframe.

Solution:

```
# Filter for messages with negative sentiment scores
negative_sentiment_data <- comm_data %>%
  filter(sentiment< 0)

new_df3<-negative_sentiment_data
print.Date(new_df3)

#another method
##negative<-comm_data%>%select(message,sentiment)%>%filter(sentiment<0)
```

Question-10: Enhancing Engagement

Apply the mutate command to add a new column to the "comm_data" dataframe, representing a sentiment label: "Positive," "Neutral," or "Negative," based on the sentiment score.

Solution:

```
# Mutate to add a new column "sentiment_label" based on sentiment scores
comm_data<-comm_data %>% select(sentiment) %>% mutate(positive = as.logical(sentiment >
0), neutral = as.logical(sentiment == 0), negative = as.logical(sentiment < 0))
print(comm_data)


#comm_data %>% select(sentiment) %>% mutate(sentiment_label = ifelse(as.logical(sentiment
> 0), "Positive", ifelse(as.logical(sentiment < 0), "Negative", "Neutral")))
#print(comm_data)
```

Question-11: Message Impact

Create a new dataframe using the mutate and arrange commands that calculates the product of the sentiment score and the length of each message. Arrange the results in descending order.

Solution:

```
#SHINY ERROR, INTRA ERROR, UNABLE TO RESOLVE

comm_data %>% mutate(a = sentiment*nchar(comm_data_org$message))%>%arrange(desc(a))
print(comm_data)
```

Question-12: Daily Message Challenge

Use the group_by, summarise, and arrange commands to find the day with the highest total number of characters sent across all messages in the “comm_data” dataframe.

Solution:

```
# Group by date and calculate the total number of characters sent on each day
daily_char_count <- comm_data %>%
  group_by(date) %>%
  summarise(total_characters = sum(nchar(message)))

#note:sum(nchar(message) can be replaced by length(message)

# Arrange the results in descending order of total_characters
daily_char_count <- daily_char_count %>%
  arrange(desc(total_characters))

# Get the day with the highest total_characters
day_with_highest_chars <- daily_char_count$day[1]

# Optionally, you can print the day with the highest total_characters
print(day_with_highest_chars)
```

Question-13: Untidy data

Can you list at least two reasons why the dataset illustrated in slide 10 is non-tidy? How can it be made Tidy?

Solution: 1)X is not a data, can be left as NA to be recognised 2)empty rows/observations 3)+/- is not recognised 4) too many variables in a column