Challenge-4

Alicia Tan

6 Sept 2023

Questions

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

```
# Enter code here
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 conflic
ts to become errors
```

```
comm_data <- read_csv("CommQuest2023_Larger.csv")
```

```
## Rows: 1000 Columns: 5
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (3): channel, sender, message
## dbl  (1): sentiment
## date (1): date
##
## 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.
```

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:

```
# Enter code here
select(comm_data, date, channel, message)
```

```
## # A tibble: 1,000 × 3
##    date       channel message
##    <date>     <chr>   <chr>
##  1 2023-08-11 Twitter Fun weekend!
##  2 2023-08-11 Email   Hello everyone!
##  3 2023-08-11 Slack   Hello everyone!
##  4 2023-08-18 Email   Fun weekend!
##  5 2023-08-14 Slack   Need assistance
##  6 2023-08-04 Email   Need assistance
##  7 2023-08-10 Twitter Hello everyone!
##  8 2023-08-04 Slack   Hello everyone!
##  9 2023-08-20 Email   Team meeting
## 10 2023-08-09 Slack   Hello everyone!
## # ℹ 990 more rows
```

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:

```
# Enter code here
comm_data %>%
  filter(date == '2023-08-02',
         channel == 'Twitter') %>%
select(date, channel, message)
```

```
## # A tibble: 15 × 3
##    date       channel message
##    <date>     <chr>   <chr>
##  1 2023-08-02 Twitter Team meeting
##  2 2023-08-02 Twitter Exciting news!
##  3 2023-08-02 Twitter Exciting news!
##  4 2023-08-02 Twitter Exciting news!
##  5 2023-08-02 Twitter Exciting news!
##  6 2023-08-02 Twitter Team meeting
##  7 2023-08-02 Twitter Great work!
##  8 2023-08-02 Twitter Hello everyone!
##  9 2023-08-02 Twitter Hello everyone!
## 10 2023-08-02 Twitter Need assistance
## 11 2023-08-02 Twitter Need assistance
## 12 2023-08-02 Twitter Need assistance
## 13 2023-08-02 Twitter Exciting news!
## 14 2023-08-02 Twitter Need assistance
## 15 2023-08-02 Twitter Need assistance
```

Question-3: Chronological Order

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

Solution:

```
# Enter code here
arrange(
  select(comm_data, date)
 )
```

```
## # A tibble: 1,000 × 1
##    date
##    <date>
##  1 2023-08-11
##  2 2023-08-11
##  3 2023-08-11
##  4 2023-08-18
##  5 2023-08-14
##  6 2023-08-04
##  7 2023-08-10
##  8 2023-08-04
##  9 2023-08-20
## 10 2023-08-09
## # ℹ 990 more rows
```

Question-4: Distinct Discovery

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

Solution:

```
# Enter code here
comm_data %>% distinct(sender)
```

```
## # A tibble: 6 × 1
##   sender
##   <chr>
## 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:

```
# Enter code here
comm_data %>%
group_by(sender) %>%
summarise(count = n())
```

```
## # 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
```

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:

```
# Enter code here
comm_data %>%
group_by(message) %>%
summarise(count = n())
```

```
## # A tibble: 6 × 2
##   message         count
##   <chr>           <int>
## 1 Exciting news!    164
## 2 Fun weekend!      165
## 3 Great work!       168
## 4 Hello everyone!   159
## 5 Need assistance   183
## 6 Team meeting      161
```

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:

```
# Enter code here
comm_data %>%
  group_by(sender) %>%
  filter(sentiment > 0) %>%
  summarise(mean_sentiment = mean(sentiment)) %>%
  slice(6,3,4)
```

```
## # A tibble: 3 × 2
##   sender        mean_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:

```
# Enter code here
comm_data %>%
 group_by(date) %>%
 summarise(average_sentiment = mean(sentiment))
```

```
## # A tibble: 20 × 2
##    date       average_sentiment
##    <date>                 <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
```

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:

```
# Enter code here
comm_data %>%
filter(sentiment < 0) %>%
select(message, sentiment)
```

```
## # A tibble: 487 × 2
##    message         sentiment
##    <chr>               <dbl>
##  1 Hello everyone!    -0.143
##  2 Need assistance    -0.108
##  3 Hello everyone!    -0.741
##  4 Hello everyone!    -0.188
##  5 Hello everyone!    -0.933
##  6 Need assistance    -0.879
##  7 Great work!        -0.752
##  8 Team meeting       -0.787
##  9 Fun weekend!       -0.539
## 10 Exciting news!     -0.142
## # ℹ 477 more rows
```

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:

```
# Enter code here
comm_data %>%
  mutate(sentiment_label = ifelse(comm_data$sentiment > 0, "positive", ifelse(comm_da
ta$sentiment  == 0, "neutral", "negative")))
```

```
## # A tibble: 1,000 × 6
##    date       channel sender        message         sentiment sentiment_label
##    <date>     <chr>   <chr>         <chr>               <dbl> <chr>
##  1 2023-08-11 Twitter dave@example  Fun weekend!        0.824 positive
##  2 2023-08-11 Email   @bob_tweets   Hello everyone!     0.662 positive
##  3 2023-08-11 Slack   @frank_chat   Hello everyone!    -0.143 negative
##  4 2023-08-18 Email   @frank_chat   Fun weekend!        0.380 positive
##  5 2023-08-14 Slack   @frank_chat   Need assistance     0.188 positive
##  6 2023-08-04 Email   @erin_tweets  Need assistance    -0.108 negative
##  7 2023-08-10 Twitter @frank_chat   Hello everyone!    -0.741 negative
##  8 2023-08-04 Slack   alice@example Hello everyone!    -0.188 negative
##  9 2023-08-20 Email   dave@example  Team meeting        0.618 positive
## 10 2023-08-09 Slack   @erin_tweets  Hello everyone!    -0.933 negative
## # ℹ 990 more rows
```

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:

```
# Enter code here
product_dataframe <- comm_data %>%
  mutate(product_dataframe = sentiment * nchar(message)) %>%
  arrange(desc(product_dataframe))

product_dataframe
```

```
## # A tibble: 1,000 × 6
##    date       channel sender       message         sentiment product_dataframe
##    <date>     <chr>   <chr>        <chr>               <dbl>              <dbl>
##  1 2023-08-16 Email   @frank_chat  Hello everyone!     0.998               15.0
##  2 2023-08-14 Slack   @erin_tweets Hello everyone!     0.988               14.8
##  3 2023-08-18 Email   dave@example Hello everyone!     0.978               14.7
##  4 2023-08-17 Email   dave@example Hello everyone!     0.977               14.7
##  5 2023-08-07 Slack   carol_slack  Hello everyone!     0.973               14.6
##  6 2023-08-06 Slack   dave@example Hello everyone!     0.968               14.5
##  7 2023-08-08 Slack   @frank_chat  Need assistance     0.964               14.5
##  8 2023-08-09 Email   @erin_tweets Need assistance     0.953               14.3
##  9 2023-08-17 Twitter @frank_chat  Hello everyone!     0.952               14.3
## 10 2023-08-12 Email   carol_slack  Need assistance     0.938               14.1
## # ℹ 990 more rows
```

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:

```
# Enter code here
comm_data %>%
  group_by(date) %>%
  mutate(number_of_character = nchar(message)) %>%
  summarise(count = n())
```

```
## # A tibble: 20 × 2
##    date       count
##    <date>     <int>
##  1 2023-08-01    45
##  2 2023-08-02    32
##  3 2023-08-03    44
##  4 2023-08-04    44
##  5 2023-08-05    44
##  6 2023-08-06    48
##  7 2023-08-07    61
##  8 2023-08-08    51
##  9 2023-08-09    43
## 10 2023-08-10    66
## 11 2023-08-11    49
## 12 2023-08-12    58
## 13 2023-08-13    51
## 14 2023-08-14    64
## 15 2023-08-15    52
## 16 2023-08-16    50
## 17 2023-08-17    43
## 18 2023-08-18    56
## 19 2023-08-19    46
## 20 2023-08-20    53
```

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: There are multiple variables in the column United States. It can be made more tidy if they is only one variable in a column. Under the percentage column, there are values with both % and without, making it hard to compute during code.It can be made more tidy if the values are presented in % form too.