Challenge-4

Wu Xingyi (Kyra)

2023-09-04

Questions

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

```
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.
```

```
comm_data
```

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

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:

```r
new_data <- comm_data %>%
  select(date, channel, message)
print(new_data)
```

```
## # A tibble: 1,000 x 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:

```r
twitter_secondaug <- comm_data %>%
  filter(channel == "Twitter", date == "2023-08-02")

print(twitter_secondaug)
```

```
## # A tibble: 15 x 5
##    date       channel sender        message         sentiment
##    <date>     <chr>   <chr>         <chr>               <dbl>
##  1 2023-08-02 Twitter alice@example Team meeting        0.210
##  2 2023-08-02 Twitter @erin_tweets  Exciting news!      0.750
##  3 2023-08-02 Twitter dave@example  Exciting news!      0.817
##  4 2023-08-02 Twitter @erin_tweets  Exciting news!      0.582
##  5 2023-08-02 Twitter @erin_tweets  Exciting news!     -0.525
##  6 2023-08-02 Twitter alice@example Team meeting        0.965
##  7 2023-08-02 Twitter dave@example  Great work!         0.516
##  8 2023-08-02 Twitter carol_slack   Hello everyone!     0.451
##  9 2023-08-02 Twitter carol_slack   Hello everyone!     0.174
## 10 2023-08-02 Twitter carol_slack   Need assistance     0.216
## 11 2023-08-02 Twitter @frank_chat   Need assistance    -0.115
## 12 2023-08-02 Twitter alice@example Need assistance     0.158
## 13 2023-08-02 Twitter carol_slack   Exciting news!     -0.693
## 14 2023-08-02 Twitter @bob_tweets   Need assistance    -0.282
## 15 2023-08-02 Twitter @erin_tweets  Need assistance     0.821
```

Question-3: Chronological Order Utilizing the arrange command, arrange the "comm_data" dataframe in ascending order based on the "date" column.

Solution:

```
comm_data %>%
  arrange(date)
```

```
## # A tibble: 1,000 x 5
##    date       channel sender        message         sentiment
##    <date>     <chr>   <chr>         <chr>               <dbl>
##  1 2023-08-01 Twitter alice@example Need assistance     0.677
##  2 2023-08-01 Twitter @bob_tweets   Need assistance     0.148
##  3 2023-08-01 Twitter @frank_chat   Need assistance     0.599
##  4 2023-08-01 Twitter @frank_chat   Exciting news!     -0.823
##  5 2023-08-01 Slack   @frank_chat   Team meeting       -0.202
##  6 2023-08-01 Slack   @bob_tweets   Exciting news!      0.146
##  7 2023-08-01 Slack   @erin_tweets  Great work!         0.244
##  8 2023-08-01 Twitter @frank_chat   Team meeting       -0.526
##  9 2023-08-01 Twitter @frank_chat   Exciting news!     -0.399
## 10 2023-08-01 Slack   @frank_chat   Need assistance     0.602
## # i 990 more rows
```

Question-4: Distinct Discovery Apply the distinct command to find the unique senders in the "comm_data" dataframe.

Solution:

```
comm_data %>%
  distinct(sender)
```

```
## # A tibble: 6 x 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:

```
comm_data %>%
  group_by(sender) %>%
  count()
```

```
## # A tibble: 6 x 2
## # Groups:   sender [6]
##   sender            n
##   <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:

```
comm_data %>%
  group_by(channel) %>%
  count()
```

```
## # A tibble: 3 x 2
## # Groups:   channel [3]
##   channel     n
##   <chr>   <int>
## 1 Email     331
## 2 Slack     320
## 3 Twitter   349
```

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:

```
top_three_senders <- comm_data %>%
  group_by(sender) %>%
  summarize(average_sentiment = mean(sentiment)) %>%
  arrange(desc(average_sentiment)) %>%
  head(3)

print(top_three_senders)
```

```
## # A tibble: 3 x 2
##   sender        average_sentiment
##   <chr>                     <dbl>
## 1 carol_slack             0.118
## 2 alice@example           0.0570
## 3 dave@example            0.00687
```

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:

```
comm_data %>%
  group_by(date) %>%
  summarize(average_sentiment = mean(sentiment))
```

```
## # A tibble: 20 x 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:

```
negative_sentiment <- comm_data %>%
  filter(sentiment < 0) %>%
  select (message, sentiment)

print (negative_sentiment)
```

```
## # A tibble: 487 x 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:

```
comm_data <- comm_data %>%
  mutate(sentiment_value = case_when(
    sentiment > 0 ~ "Positive",
    sentiment == 0 ~ "Neutral",
    sentiment < 0 ~ "Negative"))

print(comm_data)
```

```
## # A tibble: 1,000 x 6
##    date       channel sender        message         sentiment sentiment_value
##    <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:

```
product_scores <- comm_data %>%
  mutate(no_of_character = nchar(message)) %>%
  mutate(sentiment_length = sentiment * no_of_character) %>%
  arrange(desc(sentiment_length))

print(product_scores)
```

```
## # A tibble: 1,000 x 8
##    date       channel sender   message sentiment sentiment_value no_of_character
##    <date>     <chr>   <chr>    <chr>       <dbl> <chr>                      <int>
##  1 2023-08-16 Email   @frank_~ Hello ~     0.998 Positive                      15
##  2 2023-08-14 Slack   @erin_t~ Hello ~     0.988 Positive                      15
##  3 2023-08-18 Email   dave@ex~ Hello ~     0.978 Positive                      15
##  4 2023-08-17 Email   dave@ex~ Hello ~     0.977 Positive                      15
##  5 2023-08-07 Slack   carol_s~ Hello ~     0.973 Positive                      15
##  6 2023-08-06 Slack   dave@ex~ Hello ~     0.968 Positive                      15
##  7 2023-08-08 Slack   @frank_~ Need a~     0.964 Positive                      15
##  8 2023-08-09 Email   @erin_t~ Need a~     0.953 Positive                      15
##  9 2023-08-17 Twitter @frank_~ Hello ~     0.952 Positive                      15
## 10 2023-08-12 Email   carol_s~ Need a~     0.938 Positive                      15
## # i 990 more rows
## # i 1 more variable: sentiment_length <dbl>
```

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:

```
comm_data %>%
  group_by(date) %>%
  summarise(total_characters = sum(nchar(message))) %>%
  arrange(desc(total_characters)) %>%
  head(1)
```

```
## # A tibble: 1 x 2
##   date       total_characters
##   <date>                <int>
## 1 2023-08-10              875
```

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: *The variable "United States" shares a column with multiple other variables like "Estimate", "Margin of Error", "Percent", "Percent Margin of Error". This makes the data untidy as every column should only contain one variable. To make it Tidy, we could input a new column with "Country" and include "United States" as an observation. Furthermore, there are a lot of subsets - the ages categories for 16 and over, females 16 and over, own children of householder under 6 and own children of householder from 6 to 17. Instead there could be column of "age" and the data can be compared across the entire population.*