Challenge-4

Janelle Tan

2023-09-03

Questions

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

```
# Enter code here
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readr)
comm_data <- read_csv("NM2207 W4 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:

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

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

```
## # A tibble: 15 x 1
##    message
##    <chr>
##  1 Team meeting
##  2 Exciting news!
##  3 Exciting news!
##  4 Exciting news!
##  5 Exciting news!
##  6 Team meeting
##  7 Great work!
##  8 Hello everyone!
##  9 Hello everyone!
## 10 Need assistance
## 11 Need assistance
## 12 Need assistance
## 13 Exciting news!
## 14 Need assistance
## 15 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
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:

```
# Enter code here
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:

```
# Enter code here
comm_data %>%
  group_by(sender) %>%
  count(message)
```

```
## # A tibble: 36 x 3
## # Groups:   sender [6]
##    sender       message             n
##    <chr>        <chr>           <int>
```

```
##  1 @bob_tweets  Exciting news!     30
##  2 @bob_tweets  Fun weekend!       29
##  3 @bob_tweets  Great work!        29
##  4 @bob_tweets  Hello everyone!    29
##  5 @bob_tweets  Need assistance    30
##  6 @bob_tweets  Team meeting       32
##  7 @erin_tweets Exciting news!     27
##  8 @erin_tweets Fun weekend!       27
##  9 @erin_tweets Great work!        29
## 10 @erin_tweets Hello everyone!    27
## # i 26 more rows
```

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(channel) %>%
  count(message)
```

```
## # A tibble: 18 x 3
## # Groups:   channel [3]
##    channel message             n
##    <chr>   <chr>           <int>
##  1 Email   Exciting news!     51
##  2 Email   Fun weekend!       61
##  3 Email   Great work!        53
##  4 Email   Hello everyone!    47
##  5 Email   Need assistance    61
##  6 Email   Team meeting       58
##  7 Slack   Exciting news!     52
##  8 Slack   Fun weekend!       47
##  9 Slack   Great work!        50
## 10 Slack   Hello everyone!    58
## 11 Slack   Need assistance    66
## 12 Slack   Team meeting       47
## 13 Twitter Exciting news!     61
## 14 Twitter Fun weekend!       57
## 15 Twitter Great work!        65
## 16 Twitter Hello everyone!    54
## 17 Twitter Need assistance    56
## 18 Twitter Team meeting       56
```

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:

```r
# Enter code here
comm_data %>%
  group_by(sender) %>%
  summarise(avg_sentiment = mean(sentiment)) %>%
  arrange(desc(avg_sentiment)) %>% # Arrange in descending order of average sentiment
  slice_head(n = 3) %>% # Select the top three senders
  select(sender, avg_sentiment)
```

```
## # A tibble: 3 x 2
##   sender        avg_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:

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

```
## # A tibble: 20 x 2
##    date       avg_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 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:

```
# Enter code here
comm_data %>%
  mutate(sentiment_label = ifelse(sentiment > 0, "Positive",
                           ifelse(sentiment == 0, "Neutral", "Negative")))
```

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

```r
# Enter code here
comm_data %>%
  mutate(product = sentiment*nchar(message)) %>%
  arrange(desc(product))
```

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

```r
# Enter code here
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:

1."United States" could have been under its own column of "Country".

2. The subjects on the left could have been made columns of their own.