Challenge-4

Jing Ying

2023-09-04

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 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
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.
```

```
## # 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
```

```
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

newframeeee <- select(comm_data, date,channel,message)

newframeeee
```

```
## # 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:

```
# Enter code here
```

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

secnewframe
```

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

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

Solution:

```r
# Enter code here

arrange(comm_data, 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 %>% count(sender)
```

```
## # A tibble: 6 x 2
##   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:

```
# Enter code here

grouped_channel <- comm_data %>% group_by(channel)

count(grouped_channel)
```

```
## # 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:

```
# Enter code here

comm_data %>%
  select(sender,sentiment)%>%
  group_by(sender)%>%
  summarise(mean_senti = mean(sentiment))%>%
  arrange(desc(mean_senti)) %>%
  slice(1:3)
```

```
## # A tibble: 3 x 2
##   sender        mean_senti
##   <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:

```
# Enter code here

comm_data %>%
  select(date,sentiment) %>%
  group_by(date)%>%
  summarise(mean_senti = mean(sentiment))
```

```
## # A tibble: 20 x 2
##    date       mean_senti
##    <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

negative_scorers <- comm_data %>%
  select(date,channel,sender,sentiment,message) %>%
  filter(sentiment <=0)

negative_scorers
```

```
## # A tibble: 487 x 5
##    date       channel sender        sentiment message
##    <date>     <chr>   <chr>             <dbl> <chr>
##  1 2023-08-11 Slack   @frank_chat      -0.143 Hello everyone!
##  2 2023-08-04 Email   @erin_tweets     -0.108 Need assistance
##  3 2023-08-10 Twitter @frank_chat      -0.741 Hello everyone!
##  4 2023-08-04 Slack   alice@example    -0.188 Hello everyone!
##  5 2023-08-09 Slack   @erin_tweets     -0.933 Hello everyone!
##  6 2023-08-08 Slack   @erin_tweets     -0.879 Need assistance
##  7 2023-08-11 Twitter @bob_tweets      -0.752 Great work!
##  8 2023-08-12 Twitter dave@example     -0.787 Team meeting
##  9 2023-08-04 Email   @bob_tweets      -0.539 Fun weekend!
## 10 2023-08-16 Twitter @bob_tweets      -0.142 Exciting news!
## # 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 = case_when(
    sentiment > 0 ~ "Positive",
    sentiment == 0 ~ "Neutral",
    sentiment < 0 ~ "Negative",
    TRUE ~ "Unknown"))
```

```
## # 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:

```
# Enter code here

sentilength <- comm_data %>%
  mutate(yippee=sentiment*nchar(message)) %>%
  arrange(comm_data,desc(yippee))

sentilength
```

```
## # A tibble: 1,000 x 6
##    date       channel sender       message         sentiment yippee
##    <date>     <chr>   <chr>        <chr>               <dbl>  <dbl>
##  1 2023-08-01 Email   @bob_tweets  Hello everyone!     0.903  13.5
##  2 2023-08-01 Email   @bob_tweets  Team meeting        0.784   9.40
##  3 2023-08-01 Email   @erin_tweets Fun weekend!        0.204   2.44
##  4 2023-08-01 Email   @frank_chat  Fun weekend!       -0.797  -9.57
##  5 2023-08-01 Email   @frank_chat  Team meeting       -0.670  -8.05
##  6 2023-08-01 Email   @frank_chat  Team meeting       -0.308  -3.69
##  7 2023-08-01 Email   carol_slack  Fun weekend!        0.757   9.09
##  8 2023-08-01 Email   carol_slack  Great work!        -0.390  -4.29
##  9 2023-08-01 Email   carol_slack  Great work!         0.606   6.66
## 10 2023-08-01 Email   carol_slack  Need assistance    -0.131  -1.96
## # 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 %>%
  mutate(msglength = nchar(message))%>%
  group_by(date)%>%
  summarise(totalmsglength = sum(msglength))%>%
  arrange(desc(totalmsglength))%>%
  slice(1)
```

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

day with highest total number is August 10, 2023.

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:

It is non-tidy as the column subject contains multiple variables within it under a single column. For example, the variable "Employment status" under the "Subject" column is further split into more sub-categories. Hence, the variables under "Subject" appear to be arranged in an ad hoc way that makes it difficult to immediately discern at first glance. It can be made tidy by splitting the "Subject" column into more columns for every variable such as "Employment Status" and "Has children".

Furthermore, another reason why it is non-tidy is because the percent column does not only display percentages. The whole number from Estimate is included under the Percent column. This has likely interfered with the calculations within the percent margin of error column, hence yielding (X). This results in possible added complications for us in the future if we intend to use any calculations involving strict percentages as the integers (total count for relevant sub-categories) will interfere with commands we execute. We can make it tidy by replacing the integers in the Percentage with 100%.