

Challenge-4

Yang Juan Hun

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)
```

```
## Warning: package 'tidyverse' was built under R version 4.2.2
```

```
## -- Attaching packages ----- tidyverse 1.3.2 --  
## v ggplot2 3.4.0      v purrr   1.0.1  
## v tibble  3.1.8      v dplyr  1.1.0  
## v tidyr   1.2.1      v stringr 1.5.0  
## v readr   2.1.3      v forcats 0.5.2
```

```
## Warning: package 'ggplot2' was built under R version 4.2.2
```

```
## Warning: package 'tibble' was built under R version 4.2.1
```

```
## Warning: package 'tidyr' was built under R version 4.2.1
```

```
## Warning: package 'readr' was built under R version 4.2.2
```

```
## Warning: package 'purrr' was built under R version 4.2.2
```

```
## Warning: package 'dplyr' was built under R version 4.2.2
```

```
## Warning: package 'stringr' was built under R version 4.2.2
```

```
## Warning: package 'forcats' was built under R version 4.2.2
```

```
## -- Conflicts ----- tidyverse_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag()    masks stats::lag()
```

```
comm_data <- 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:

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

```
##           date channel      message
## 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
```

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

```
##           date channel      sender      message sentiment
## 1 2023-08-02 Twitter alice@example Team meeting  0.2100900
## 2 2023-08-02 Twitter @erin_tweets Exciting news!  0.7504925
## 3 2023-08-02 Twitter dave@example Exciting news!  0.8171056
## 4 2023-08-02 Twitter @erin_tweets Exciting news!  0.5815569
## 5 2023-08-02 Twitter @erin_tweets Exciting news! -0.5251436
## 6 2023-08-02 Twitter alice@example Team meeting  0.9649580
```

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(comm_data, date) %>%
  head()
```

```
##           date channel      sender      message sentiment
## 1 2023-08-01 Twitter alice@example Need assistance  0.6767770
## 2 2023-08-01 Twitter @bob_tweets Need assistance  0.1483952
## 3 2023-08-01 Twitter @frank_chat Need assistance  0.5990454
## 4 2023-08-01 Twitter @frank_chat Exciting news! -0.8227803
## 5 2023-08-01   Slack @frank_chat Team meeting -0.2020947
## 6 2023-08-01   Slack @bob_tweets Exciting news!  0.1463969
```

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

Solution:

```
# Enter code here
distinct(comm_data, sender)
```

```
##           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:

```
# Enter code here
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:

```
# Enter code here
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:

```
# Enter code here
comm_data %>%
  group_by(sender) %>%
  mutate(sentiment_avg = mean(sentiment)) %>%
  filter(sentiment_avg > 0) %>%
  arrange(desc(sentiment_avg)) %>%
  select(sender, sentiment_avg) %>%
  unique() %>%
  slice(1:3)
```

```
## # A tibble: 3 x 2
## # Groups:   sender [3]
##   sender      sentiment_avg
##   <chr>          <dbl>
## 1 alice@example    0.0570
## 2 carol_slack      0.118
## 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 %>%
  group_by(date) %>%
  summarise(avg_sentiment_score = mean(sentiment))
```

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

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) %>%
  head()
```

```
##           message
## 1 Hello everyone!
## 2 Need assistance
## 3 Hello everyone!
## 4 Hello everyone!
## 5 Hello everyone!
## 6 Need assistance
```

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 ~ 'Negative',
    sentiment > 0 ~ 'Positive',
    TRUE ~ 'Neutral')) %>%
  head()
```

```
##           date channel      sender      message  sentiment sentiment_label
## 1 2023-08-11 Twitter dave@example  Fun weekend!  0.8240997      Positive
## 2 2023-08-11 Email  @bob_tweets Hello everyone!  0.6624869      Positive
## 3 2023-08-11 Slack @frank_chat Hello everyone! -0.1434508      Negative
## 4 2023-08-18 Email  @frank_chat  Fun weekend!  0.3801966      Positive
## 5 2023-08-14 Slack @frank_chat Need assistance  0.1879540      Positive
## 6 2023-08-04 Email  @erin_tweets Need assistance -0.1083762      Negative
```

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
comm_data %>%
  mutate(product = sentiment * nchar(message)) %>%
  arrange(desc(product)) %>%
  head()
```

```
##           date channel      sender      message sentiment  product
## 1 2023-08-16   Email @frank_chat Hello everyone! 0.9976019 14.96403
## 2 2023-08-14   Slack @erin_tweets Hello everyone! 0.9878323 14.81748
## 3 2023-08-18   Email dave@example Hello everyone! 0.9782200 14.67330
## 4 2023-08-17   Email dave@example Hello everyone! 0.9768948 14.65342
## 5 2023-08-07   Slack carol_slack Hello everyone! 0.9734297 14.60145
## 6 2023-08-06   Slack dave@example Hello everyone! 0.9680817 14.52123
```

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) %>%
  summarise(total_char = sum(nchar(message))) %>%
  arrange(desc(total_char)) %>%
  slice(1)
```

```
## # A tibble: 1 x 2
##   date      total_char
##   <chr>         <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: *There are variables in both rows and columns, and some rows do not contain data at all. First, I would remove all empty rows, then reformat the table from the current “long” format to a “wide” format using `pivot_wider()`.*