TASK2

Rachit Biswas

2023-06-12

```
library(dplyr)
```

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

```
##
## 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(readxl)
```

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

```
purchase_data=read.csv("E:\\PROJECTS\\FORAGE R\\QVI_purchase_behaviour
(1).csv")
print("Purchase Data")
```

```
## [1] "Purchase Data"
```

```
head(purchase_data)
```

```
##   LYLTY_CARD_NBR               LIFESTAGE PREMIUM_CUSTOMER
## 1           1000  YOUNG SINGLES/COUPLES          Premium
## 2           1002  YOUNG SINGLES/COUPLES       Mainstream
## 3           1003         YOUNG FAMILIES           Budget
## 4           1004  OLDER SINGLES/COUPLES       Mainstream
## 5           1005 MIDAGE SINGLES/COUPLES       Mainstream
## 6           1007  YOUNG SINGLES/COUPLES           Budget
```

```
transcation_data=read_xlsx("E:\\PROJECTS\\FORAGE R\\QVI_transaction_data
(1).xlsx")
print("Transaction Data")
```

```
## [1] "Transaction Data"
```

```
head(transcation_data)
```

```
## # A tibble: 6 × 8
##    DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME       PROD_…¹
```

```
TOT_S…²
##   <dbl>     <dbl>          <dbl>  <dbl>    <dbl> <chr>              <dbl>
<dbl>
## 1 43390         1           1000      1        5 Natural Chip  …       2
6
## 2 43599         1           1307    348       66 CCs Nacho Chee…       3
6.3
## 3 43605         1           1343    383       61 Smiths Crinkle…       2
2.9
## 4 43329         2           2373    974       69 Smiths Chip Th…       5
15
## 5 43330         2           2426   1038      108 Kettle Tortill…       3
13.8
## 6 43604         4           4074   2982       57 Old El Paso Sa…       1
5.1
## # … with abbreviated variable names ¹PROD_QTY, ²TOT_SALES
```

```
print("NA value Check")
```

```
## [1] "NA value Check"
```

```
sum(is.na(transcation_data))
```

```
## [1] 0
```

```
sum(is.na(purchase_data))
```

```
## [1] 0
```

```
summary_stats <- purchase_data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(count = n())
```

```
## `summarise()` has grouped output by 'LIFESTAGE'. You can override using
the
## `.groups` argument.
```

```
# Print the summary statistics
print(summary_stats)
```

```
## # A tibble: 21 × 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER count
##    <chr>                  <chr>            <int>
##  1 MIDAGE SINGLES/COUPLES Budget            1504
##  2 MIDAGE SINGLES/COUPLES Mainstream        3340
##  3 MIDAGE SINGLES/COUPLES Premium           2431
##  4 NEW FAMILIES           Budget            1112
##  5 NEW FAMILIES           Mainstream         849
##  6 NEW FAMILIES           Premium            588
##  7 OLDER FAMILIES         Budget            4675
##  8 OLDER FAMILIES         Mainstream        2831
```

```
##  9 OLDER FAMILIES         Premium           2274
## 10 OLDER SINGLES/COUPLES  Budget            4929
## # … with 11 more rows
```

```
#transaction Datat
print("Transaction Data")
```

```
## [1] "Transaction Data"
```

```
str(transcation_data)
```

```
## tibble [264,836 × 8] (S3: tbl_df/tbl/data.frame)
##  $ DATE          : num [1:264836] 43390 43599 43605 43329 43330 ...
##  $ STORE_NBR     : num [1:264836] 1 1 1 2 2 4 4 4 5 7 ...
##  $ LYLTY_CARD_NBR: num [1:264836] 1000 1307 1343 2373 2426 ...
##  $ TXN_ID        : num [1:264836] 1 348 383 974 1038 ...
##  $ PROD_NBR      : num [1:264836] 5 66 61 69 108 57 16 24 42 52 ...
##  $ PROD_NAME     : chr [1:264836] "Natural Chip        Compny SeaSalt175g"
"CCs Nacho Cheese    175g" "Smiths Crinkle Cut  Chips Chicken 170g" "Smiths
Chip Thinly  S/Cream&Onion 175g" ...
##  $ PROD_QTY      : num [1:264836] 2 3 2 5 3 1 1 1 1 2 ...
##  $ TOT_SALES     : num [1:264836] 6 6.3 2.9 15 13.8 5.1 5.7 3.6 3.9 7.2
...
```

```
summary(transcation_data)
```

```
##       DATE         STORE_NBR     LYLTY_CARD_NBR        TXN_ID
##  Min.   :43282   Min.   :  1.0   Min.   :   1000   Min.   :      1
##  1st Qu.:43373   1st Qu.: 70.0   1st Qu.:  70021   1st Qu.:  67602
##  Median :43464   Median :130.0   Median : 130358   Median : 135138
##  Mean   :43464   Mean   :135.1   Mean   : 135550   Mean   : 135158
##  3rd Qu.:43555   3rd Qu.:203.0   3rd Qu.: 203094   3rd Qu.: 202701
##  Max.   :43646   Max.   :272.0   Max.   :2373711   Max.   :2415841
##     PROD_NBR       PROD_NAME            PROD_QTY         TOT_SALES
##  Min.   :  1.00   Length:264836      Min.   :  1.000   Min.   :  1.500
##  1st Qu.: 28.00   Class :character   1st Qu.:  2.000   1st Qu.:  5.400
##  Median : 56.00   Mode  :character   Median :  2.000   Median :  7.400
##  Mean   : 56.58                      Mean   :  1.907   Mean   :  7.304
##  3rd Qu.: 85.00                      3rd Qu.:  2.000   3rd Qu.:  9.200
##  Max.   :114.00                      Max.   :200.000   Max.   :650.000
```

```
summary_stats2 <- transcation_data %>%
  summarise(
    total_sales = sum(TOT_SALES),
    total_quantity = sum(PROD_QTY),
    num_transactions = n(),
    avg_sales_per_transaction = mean(TOT_SALES),
    top_selling_products = paste(unique(PROD_NAME), collapse = ", ")
  )
```

```r
# Print the summary statistics
print(summary_stats2)
```

```
## # A tibble: 1 × 5
##   total_sales total_quantity num_transactions avg_sales_per_transaction
top_se…¹
##         <dbl>          <dbl>            <int>                      <dbl>
<chr>
## 1     1934415         505124           264836                       7.30
Natural…
## # … with abbreviated variable name ¹top_selling_products
```

```r
print("outlier detection")
```

```
## [1] "outlier detection"
```

```r
# Identify outliers using the IQR method
outliers <- transcation_data %>%
  filter(TOT_SALES > quantile(TOT_SALES, 0.75) + 1.5 * IQR(TOT_SALES) |
           TOT_SALES < quantile(TOT_SALES, 0.25) - 1.5 * IQR(TOT_SALES))

# Print the outliers
print(outliers)
```

```
## # A tibble: 578 × 8
##     DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME      PROD_…¹
TOT_S…²
##    <dbl>     <dbl>          <dbl>  <dbl>    <dbl> <chr>            <dbl>
<dbl>
##  1 43329         2           2373    974       69 Smiths Chip T…       5
15
##  2 43332         8           8294   8221      114 Kettle Sensat…       5
23
##  3 43601        74          74336  73182       84 GrnWves Plus …       5
15.5
##  4 43331        96          96203  96025        7 Smiths Crinkl…       5
28.5
##  5 43605       130         130108 134125        2 Cobs Popd Sou…       5
19
##  6 43600       133         133250 137666       30 Doritos Corn …       4
17.6
##  7 43602       168         168219 170719       33 Cobs Popd Swt…       4
15.2
##  8 43602       222         222209 222693       40 Thins Chips S…       5
16.5
##  9 43329       257         257258 257308      114 Kettle Sensat…       4
18.4
## 10 43331       262         262126 262025      108 Kettle Tortil…       4
18.4
## # … with 568 more rows, and abbreviated variable names ¹PROD_QTY, ²
TOT_SALES
```

```r
library(tinytex)
```

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

```r
print("removing")
```

```
## [1] "removing"
```

```r
q1 <- quantile(transcation_data$TOT_SALES, 0.25)
q3 <- quantile(transcation_data$TOT_SALES, 0.75)
iqr <- q3 - q1
lower_threshold <- q1 - 1.5 * iqr
upper_threshold <- q3 + 1.5 * iqr

# Remove outliers
transaction_data <- transcation_data %>%
  filter(TOT_SALES >= lower_threshold, TOT_SALES <= upper_threshold)

View(transaction_data)

#MUTATE

transaction_data <- transaction_data %>%
  mutate(
    PACK_SIZE = as.numeric(gsub("[^0-9]", "", PROD_NAME)),
    BRAND_NAME = gsub("[0-9]", "", PROD_NAME)
  )

# Print the updated dataset
head(transaction_data)
```

```
## # A tibble: 6 × 10
##    DATE STORE_NBR LYLTY…¹ TXN_ID PROD_…² PROD_…³ PROD_…⁴ TOT_S…⁵ PACK_…⁶
BRAND…⁷
##   <dbl>     <dbl>   <dbl>  <dbl>   <dbl> <chr>     <dbl>   <dbl>   <dbl>
<chr>
## 1 43390         1    1000      1       5 Natura…       2     6       175
Natura…
## 2 43599         1    1307    348      66 CCs Na…       3     6.3     175
CCs Na…
## 3 43605         1    1343    383      61 Smiths…       2     2.9     170
Smiths…
## 4 43330         2    2426   1038     108 Kettle…       3    13.8     150
Kettle…
## 5 43604         4    4074   2982      57 Old El…       1     5.1     300
Old El…
## 6 43601         4    4149   3333      16 Smiths…       1     5.7     330
Smiths…
## # … with abbreviated variable names ¹LYLTY_CARD_NBR, ²PROD_NBR, ³
```

```
PROD_NAME,
## #   ⁴PROD_QTY, ⁵TOT_SALES, ⁶PACK_SIZE, ⁷BRAND_NAME

transaction_data <- merge(transaction_data, purchase_data, by =
"LYLTY_CARD_NBR")
head(transaction_data)

##   LYLTY_CARD_NBR  DATE STORE_NBR TXN_ID PROD_NBR
## 1           1000 43390         1      1        5
## 2           1002 43359         1      2       58
## 3           1003 43532         1      4      106
## 4           1003 43531         1      3       52
## 5           1004 43406         1      5       96
## 6           1005 43462         1      6       86
##                               PROD_NAME PROD_QTY TOT_SALES PACK_SIZE
## 1 Natural Chip        Compny SeaSalt175g        2       6.0       175
## 2  Red Rock Deli Chikn&Garlic Aioli 150g        1       2.7       150
## 3 Natural ChipCo      Hony Soy Chckn175g        1       3.0       175
## 4  Grain Waves Sour    Cream&Chives 210G        1       3.6       210
## 5         WW Original Stacked Chips 160g        1       1.9       160
## 6                     Cheetos Puffs 165g        1       2.8       165
##                          BRAND_NAME              LIFESTAGE
PREMIUM_CUSTOMER
## 1 Natural Chip        Compny SeaSaltg  YOUNG SINGLES/COUPLES
Premium
## 2  Red Rock Deli Chikn&Garlic Aioli g  YOUNG SINGLES/COUPLES
Mainstream
## 3 Natural ChipCo      Hony Soy Chckng         YOUNG FAMILIES
Budget
## 4  Grain Waves Sour    Cream&Chives G         YOUNG FAMILIES
Budget
## 5         WW Original Stacked Chips g  OLDER SINGLES/COUPLES
Mainstream
## 6                     Cheetos Puffs g MIDAGE SINGLES/COUPLES
Mainstream

metrics <- transaction_data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(
    total_spending = sum(TOT_SALES),
    average_spending = mean(TOT_SALES),
    total_quantity = sum(PROD_QTY),
    average_price_per_chip = sum(TOT_SALES) / sum(PROD_QTY),
    purchase_frequency = n(),
    top_brand = names(which.max(table(BRAND_NAME))),
    top_pack_size = names(which.max(table(PACK_SIZE)))
  )

## `summarise()` has grouped output by 'LIFESTAGE'. You can override using
the
## `.groups` argument.
```

```r
# Print the metrics
print(metrics)
```

```
## # A tibble: 21 × 9
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE     PREMI…¹ total…² avera…³ total…⁴ avera…⁵ purch…⁶ top_b…⁷
top_p…⁸
##    <chr>         <chr>     <dbl>   <dbl>   <dbl>   <dbl>   <int> <chr>
<chr>
##  1 MIDAGE SINGL… Budget   35309.    7.05    9445    3.74    5009 Infzns…
175
##  2 MIDAGE SINGL… Mainst…  90178.    7.61   22561    4.00   11843 Smiths…
175
##  3 MIDAGE SINGL… Premium  58096.    7.09   15449    3.76    8199 Pringl…
175
##  4 NEW FAMILIES  Budget   21862.    7.28    5558    3.93    3002 Kettle…
175
##  5 NEW FAMILIES  Mainst…  16940.    7.30    4301    3.94    2321 Kettle…
175
##  6 NEW FAMILIES  Premium  11450.    7.22    2948    3.88    1587 Grain …
175
##  7 OLDER FAMILI… Budget  167214.    7.24   44816    3.73   23104 Smiths…
175
##  8 OLDER FAMILI… Mainst… 102669.    7.23   27576    3.72   14204 Old El…
175
##  9 OLDER FAMILI… Premium  80062.    7.18   21626    3.70   11158 Infuzi…
175
## 10 OLDER SINGLE… Budget  135859.    7.40   35022    3.88   18361 Cobs P…
175
## # … with 11 more rows, and abbreviated variable names ¹PREMIUM_CUSTOMER,
## #   ²total_spending, ³average_spending, ⁴total_quantity,
## #   ⁵average_price_per_chip, ⁶purchase_frequency, ⁷top_brand, ⁸
top_pack_size
```

```r
library(ggplot2)
```

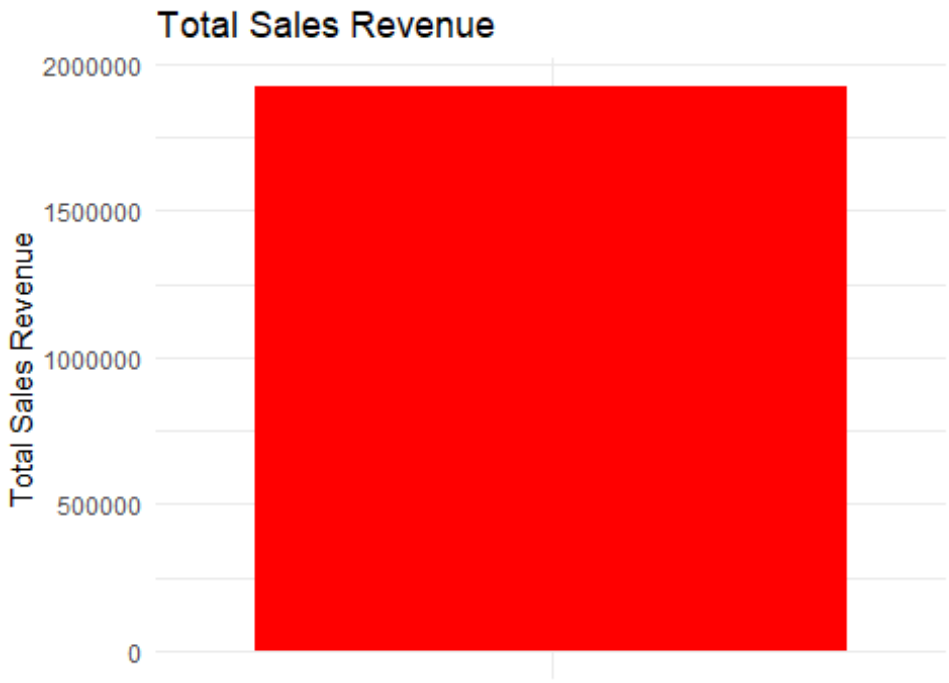
```
## Warning: package 'ggplot2' was built under R version 4.2.2
```

```r
total_revenue <- sum(transaction_data$TOT_SALES)

# Print the total sales revenue
print(total_revenue)
```

```
## [1] 1921757
```

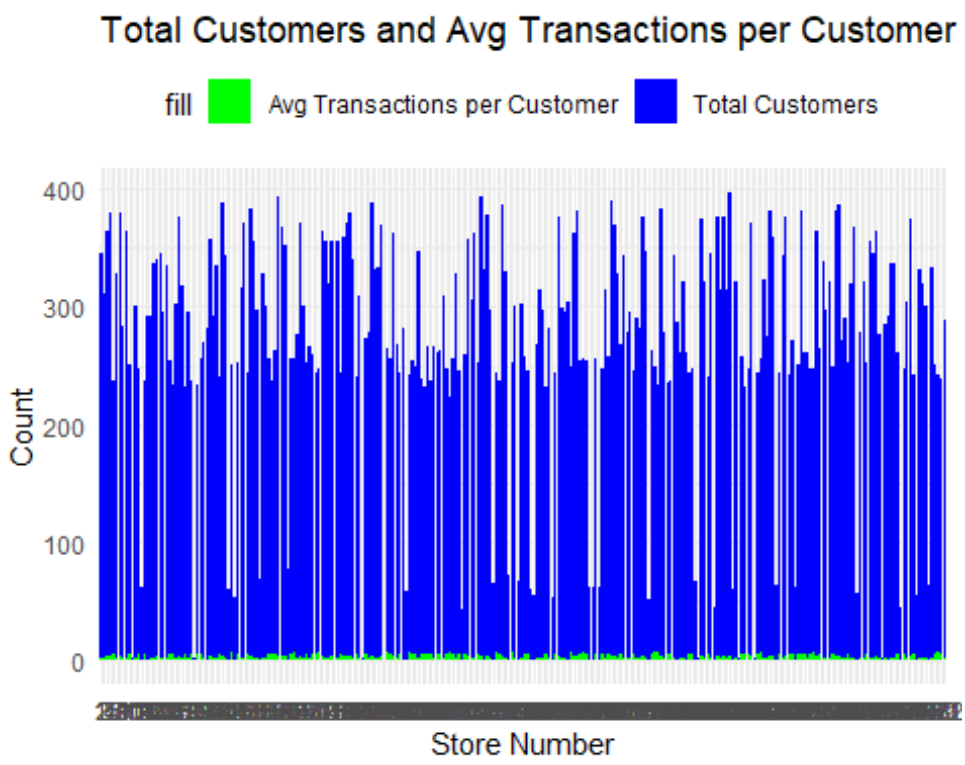
```r
plot <- ggplot() +
  geom_bar(data = data.frame(total_revenue), aes(x = "", y = total_revenue),
stat = "identity", fill = "red") +
  labs(x = "", y = "Total Sales Revenue") +
  ggtitle("Total Sales Revenue") +
  theme_minimal()
```

```r
# Print the bar plot
print(plot)
```




```r
# Calculate total number of customers
total_customers <- transaction_data %>%
  group_by(STORE_NBR) %>%
  summarise(total_customers = n_distinct(LYLTY_CARD_NBR))

# Calculate average number of transactions per customer
avg_transactions_per_customer <- transaction_data %>%
  group_by(STORE_NBR) %>%
  summarise(avg_transactions_per_customer = n()/n_distinct(LYLTY_CARD_NBR))

# Merge the metrics into a single dataset
metrics <- merge(total_customers, avg_transactions_per_customer, by =
"STORE_NBR")

# Create a grouped bar plot
plot <- ggplot(metrics, aes(x = factor(STORE_NBR))) +
  geom_bar(aes(y = total_customers, fill = "Total Customers"), stat =
"identity", position = "dodge") +
  geom_bar(aes(y = avg_transactions_per_customer, fill = "Avg Transactions
per Customer"), stat = "identity", position = "dodge") +
  labs(x = "Store Number", y = "Count") +
  ggtitle("Total Customers and Avg Transactions per Customer") +
```

```
  scale_fill_manual(values = c("Total Customers" = "blue", "Avg Transactions
per Customer" = "green")) +
  theme_minimal() +
  theme(legend.position = "top")

# Print the bar plot
print(plot)
```




```
print(total_customers)

## # A tibble: 272 × 2
##    STORE_NBR total_customers
##        <dbl>           <int>
##  1         1             345
##  2         2             312
##  3         3             364
##  4         4             379
##  5         5             238
##  6         6             328
##  7         7             379
##  8         8             283
##  9         9             364
## 10        10             251
## # … with 262 more rows

print(avg_transactions_per_customer)
```

```
## # A tibble: 272 × 2
##    STORE_NBR avg_transactions_per_customer
##        <dbl>                         <dbl>
##  1         1                          1.66
##  2         2                          1.62
##  3         3                          4.13
##  4         4                          4.42
##  5         5                          5.70
##  6         6                          1.62
##  7         7                          3.91
##  8         8                          1.83
##  9         9                          1.80
## 10        10                          5.84
## # … with 262 more rows

# Function to calculate the magnitude distance measure
calculate_magnitude_distance <- function(trial_store, control_stores, data) {
  # Subset the data for the trial store and control stores
  trial_data <- subset(data, STORE_NBR == trial_store)
  control_data <- subset(data, STORE_NBR %in% control_stores)

  # Calculate the total sales for the trial store and control stores
  trial_sales <- sum(trial_data$TOT_SALES)
  control_sales <- aggregate(data$TOT_SALES, by = list(data$STORE_NBR),
sum)$x

  # Calculate the observed distance for the trial store
  observed_distance <- abs(trial_sales - control_sales)

  # Calculate the minimum and maximum distances among the control stores
  min_distance <- min(abs(control_sales - trial_sales))
  max_distance <- max(abs(control_sales - trial_sales))

  # Calculate the magnitude distance measure
  magnitude_distance <- 1 - (observed_distance - min_distance) /
(max_distance - min_distance)

  # Return the magnitude distance measure
  return(magnitude_distance)
}

# Example usage of the function
trial_store <- 77
control_stores <- c(86, 77, 88)
data <- transaction_data

magnitude_distance <- calculate_magnitude_distance(trial_store,
control_stores, data)
print(magnitude_distance)
```

```
##   [1] 0.95809798 0.93023248 0.32182302 0.19539749 0.55512049 0.97824997
##   [7] 0.34597704 0.89904636 0.93232170 0.49706506 0.79297344 0.86376482
##  [13] 0.46934484 0.82092196 0.55124990 0.90434551 0.89080016 0.98129386
##  [19] 0.59090705 0.96566622 0.85354701 0.95329692 0.49945175 0.57345306
##  [25] 0.84441877 0.21075188 0.85761476 0.56159224 0.88729276 0.49196653
##  [31] 0.79353379 0.51787425 0.43769522 0.91398913 0.90259527 0.38027969
##  [37] 0.88270962 0.98954698 0.50678478 0.13627323 0.96873779 0.81034441
##  [43] 0.35576248 0.81372729 0.55195899 0.99927016 0.90017745 0.50397609
##  [49] 0.29917019 0.99950883 0.92965829 0.82469916 0.98033573 0.83885674
##  [55] 0.46972878 0.63079249 0.48203927 0.15782610 0.31332086 0.32028377
##  [61] 0.83145107 0.48044814 0.38075357 0.87168587 0.31047066 0.85616199
##  [67] 0.49558462 0.93755167 0.42811040 0.55382684 0.41329561 0.22838573
##  [73] 0.93739947 0.94464948 0.30695980 0.79292501 1.00000000 0.56231862
##  [79] 0.39699692 0.39702459 0.22321456 0.92802565 0.52220489 0.83722756
##  [85] 0.79347153 0.47404558 0.93420338 0.08519455 0.95785585 0.99124188
##  [91] 0.45317067 0.79314639 0.32566594 0.42440929 0.25570817 0.90758311
##  [97] 0.48833807 0.90117709 0.80786087 0.39416748 0.51334991 0.48794720
## [103] 0.97001761 0.50222585 0.48419076 0.46659149 0.56542824 0.95320699
## [109] 0.48936538 0.54003245 0.95535847 0.42163174 0.48180406 0.43902346
## [115] 0.93372258 0.49379979 0.80370318 0.52462963 0.42893017 0.91773520
## [121] 0.96335562 0.51119843 0.28487079 0.93905978 0.28689429 0.87901198
## [127] 0.81858369 0.39302601 0.53153029 0.22461199 0.96060228 0.82519033
## [133] 0.41114067 0.84560174 0.81880506 0.93241163 0.45051072 0.46129926
## [139] 0.81605172 0.80945199 0.99947424 0.83475094 0.89543519 0.61030844
## [145] 0.94920841 0.81154122 0.52581606 0.60294428 0.87425243 0.86366797
## [151] 0.92002158 0.39190531 0.30676956 0.33507089 0.47509019 0.37776502
## [157] 0.43053514 0.81678502 0.81595487 0.45365147 0.81569890 0.53896708
## [163] 0.97484634 0.47014732 0.11433295 0.38079162 0.94771414 0.39634663
## [169] 0.96296130 0.91508909 0.88404825 0.48179023 0.81424959 0.89207307
## [175] 0.39625670 0.98136304 0.80712065 0.42677177 0.39111666 0.46128542
## [181] 0.24256407 0.90005638 0.51828241 0.45242699 0.98946396 0.92441448
## [187] 0.99127647 0.99503983 0.88704372 0.53287237 0.49191811 0.81750449
## [193] 0.79341619 0.31069896 0.97028395 0.48745603 0.94128736 0.80999159
## [199] 0.18728619 0.82678146 0.22842032 0.79954203 0.20610993 0.82405579
## [205] 0.99652719 0.79303570 0.49611038 0.56994912 0.54553913 0.26318994
## [211] 0.79286967 0.59814668 0.35546847 0.97911471 0.90765921 0.31289540
## [217] 0.24559759 0.82149614 0.53277552 0.99835353 0.50262017 0.53711999
## [223] 0.43354099 0.81969056 0.48179714 0.00000000 0.47534270 0.91442496
## [229] 0.48790569 0.36125879 0.31452112 0.48340211 0.98489120 0.88265774
## [235] 0.81234370 0.47814447 0.13675057 0.27036385 0.92476384 0.88683618
## [241] 0.55942346 0.80964570 0.88763174 0.81546023 0.91079304 0.79859081
## [247] 0.47442953 0.91789777 0.97309609 0.31905930 0.88700913 0.79302186
## [253] 0.86884951 0.90467065 0.98865456 0.96745105 0.43760528 0.81574733
## [259] 0.40909641 0.83296265 0.25756910 0.57195878 0.81329837 0.96554861
## [265] 0.95963376 0.91306904 0.81156197 0.96849912 0.44001965 0.43048325
## [271] 0.53494083 0.88702642

analyze_trial_and_control <- function(trial_store, control_store, data) {
  # Subset the data for the trial store and control store during the trial
period
```

```r
  trial_data <- subset(data, STORE_NBR == trial_store)
  control_data <- subset(data, STORE_NBR == control_store)

  # Calculate the total sales for the trial store and control store during
the trial period
  trial_total_sales <- sum(trial_data$TOT_SALES)
  control_total_sales <- sum(control_data$TOT_SALES)

  # Perform a t-test to compare total sales between trial store and control
store
  t_test_result <- t.test(trial_data$TOT_SALES, control_data$TOT_SALES)

  # Calculate the metrics for trial and control stores during the trial
period
  trial_num_customers <- n_distinct(trial_data$LYLTY_CARD_NBR)
  control_num_customers <- n_distinct(control_data$LYLTY_CARD_NBR)

  trial_avg_transactions_per_customer <- nrow(trial_data) /
trial_num_customers
  control_avg_transactions_per_customer <- nrow(control_data) /
control_num_customers

  # Create a summary dataframe
  summary_df <- data.frame(
    Store = c("Trial", "Control"),
    Total_Sales = c(trial_total_sales, control_total_sales),
    Num_Customers = c(trial_num_customers, control_num_customers),
    Avg_Transactions_per_Customer = c(trial_avg_transactions_per_customer,
control_avg_transactions_per_customer)
  )

  # Print the t-test result and the summary dataframe
  print(paste("T-Test Result for Trial Store", trial_store, "and Control
Store", control_store))
  print(t_test_result)
  plot_sales <- ggplot(data = summary_df, aes(x = Store, y = Total_Sales,
fill = Store)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(x = "Store", y = "Total Sales", title = "Total Sales Comparison") +
    theme_minimal()

  # Plotting number of customers comparison
  plot_customers <- ggplot(data = summary_df, aes(x = Store, y =
Num_Customers, fill = Store)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(x = "Store", y = "Number of Customers", title = "Number of Customers
Comparison") +
    theme_minimal()
```

```
  # Plotting average transactions per customer comparison
  plot_avg_transactions <- ggplot(data = summary_df, aes(x = Store, y =
Avg_Transactions_per_Customer, fill = Store)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(x = "Store", y = "Average Transactions per Customer", title =
"Average Transactions per Customer Comparison") +
    theme_minimal()


  print("Summary:")
  print(summary_df)
}



# Example usage of the function
trial_store <- 77
control_store <- 86
data <- transaction_data  # Replace with your actual data file

analyze_trial_and_control(trial_store, control_store, data)
```

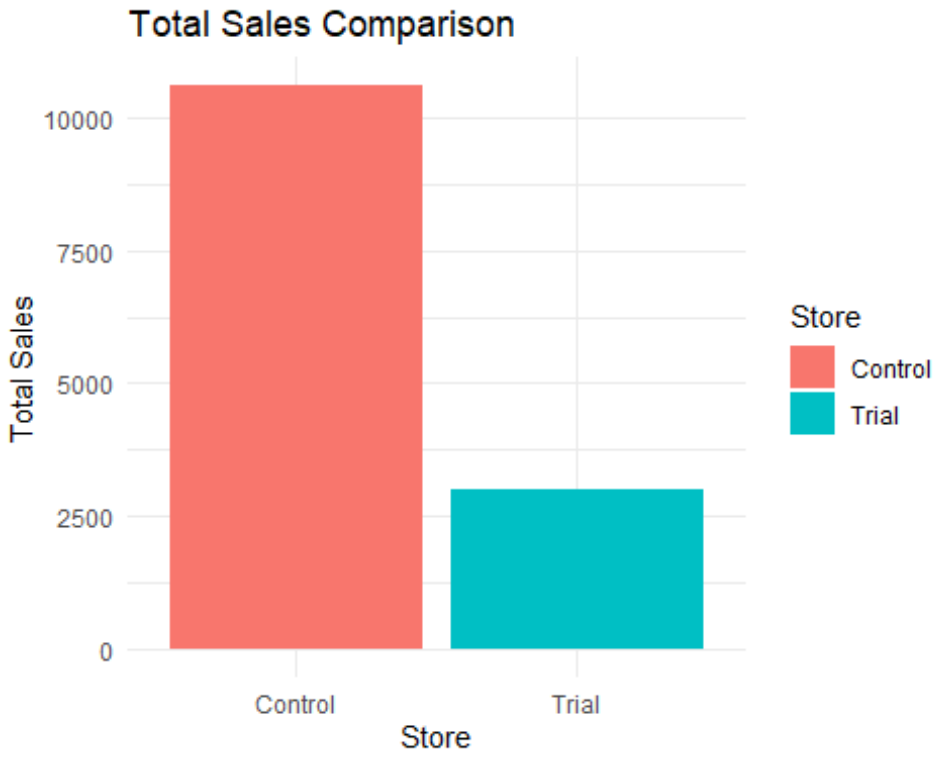
```
## [1] "T-Test Result for Trial Store 77 and Control Store 86"
##
##  Welch Two Sample t-test
##
## data:  trial_data$TOT_SALES and control_data$TOT_SALES
## t = -13.122, df = 936.07, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.788743 -1.323318
## sample estimates:
## mean of x mean of y
##  5.346346  6.902376
##
## [1] "Summary:"
##     Store Total_Sales Num_Customers Avg_Transactions_per_Customer
## 1   Trial     2999.30           355                       1.580282
## 2 Control    10602.05           273                       5.626374
```

```
analyze_trial_and_control <- function(trial_store, control_store, data) {
  # Subset the data for the trial store and control store during the trial
period
  trial_data <- subset(data, STORE_NBR == trial_store)
  control_data <- subset(data, STORE_NBR == control_store)

  # Calculate the total sales for the trial store and control store during
the trial period
  trial_total_sales <- sum(trial_data$TOT_SALES)
  control_total_sales <- sum(control_data$TOT_SALES)
```

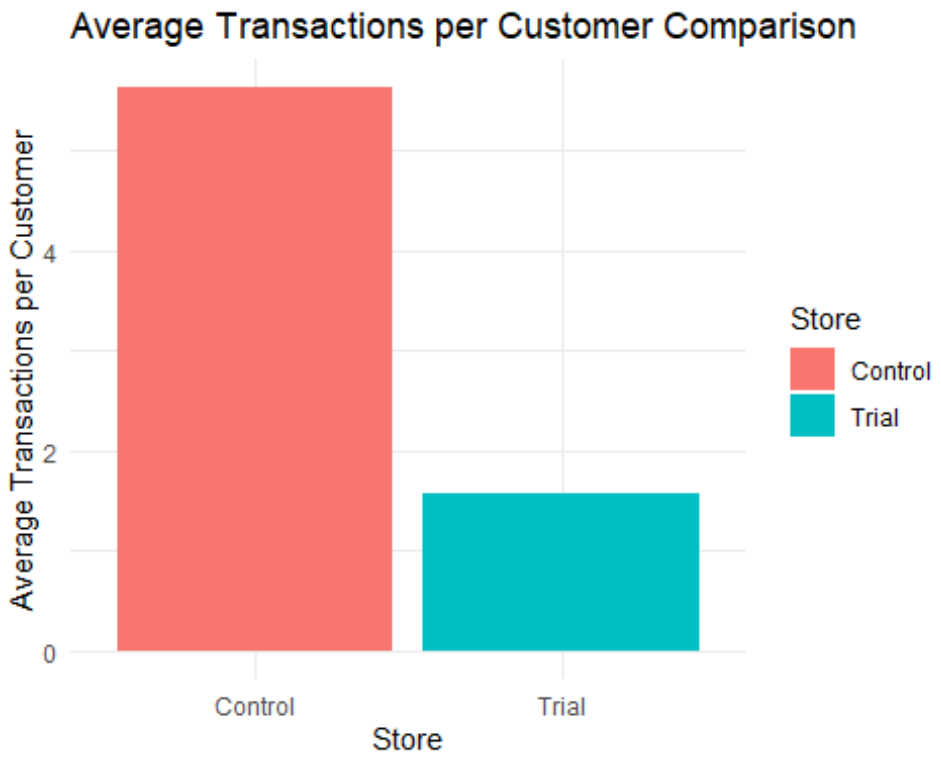
```r
  # Perform a t-test to compare total sales between trial store and control
store
  t_test_result <- t.test(trial_data$TOT_SALES, control_data$TOT_SALES)

  # Calculate the metrics for trial and control stores during the trial
period
  trial_num_customers <- n_distinct(trial_data$LYLTY_CARD_NBR)
  control_num_customers <- n_distinct(control_data$LYLTY_CARD_NBR)

  trial_avg_transactions_per_customer <- nrow(trial_data) /
trial_num_customers
  control_avg_transactions_per_customer <- nrow(control_data) /
control_num_customers

  # Create a summary dataframe
  summary_df <- data.frame(
    Store = c("Trial", "Control"),
    Total_Sales = c(trial_total_sales, control_total_sales),
    Num_Customers = c(trial_num_customers, control_num_customers),
    Avg_Transactions_per_Customer = c(trial_avg_transactions_per_customer,
control_avg_transactions_per_customer)
  )

  # Print the t-test result and the summary dataframe
  print(paste("T-Test Result for Trial Store", trial_store, "and Control
Store", control_store))
  print(t_test_result)

  print("Summary:")
  print(summary_df)

  # Plotting total sales comparison
  plot_sales <- ggplot(data = summary_df, aes(x = Store, y = Total_Sales,
fill = Store)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(x = "Store", y = "Total Sales", title = "Total Sales Comparison") +
    theme_minimal()

  # Plotting number of customers comparison
  plot_customers <- ggplot(data = summary_df, aes(x = Store, y =
Num_Customers, fill = Store)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(x = "Store", y = "Number of Customers", title = "Number of Customers
Comparison") +
    theme_minimal()

  # Plotting average transactions per customer comparison
  plot_avg_transactions <- ggplot(data = summary_df, aes(x = Store, y =
```

```
Avg_Transactions_per_Customer, fill = Store)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(x = "Store", y = "Average Transactions per Customer", title =
"Average Transactions per Customer Comparison") +
    theme_minimal()

  # Print the plots
  print(plot_sales)
  print(plot_customers)
  print(plot_avg_transactions)
}

# Example usage of the function
trial_store <- 77
control_store <- 86
data <- transaction_data  # Replace with your actual data file

analyze_trial_and_control(trial_store, control_store, data)
```

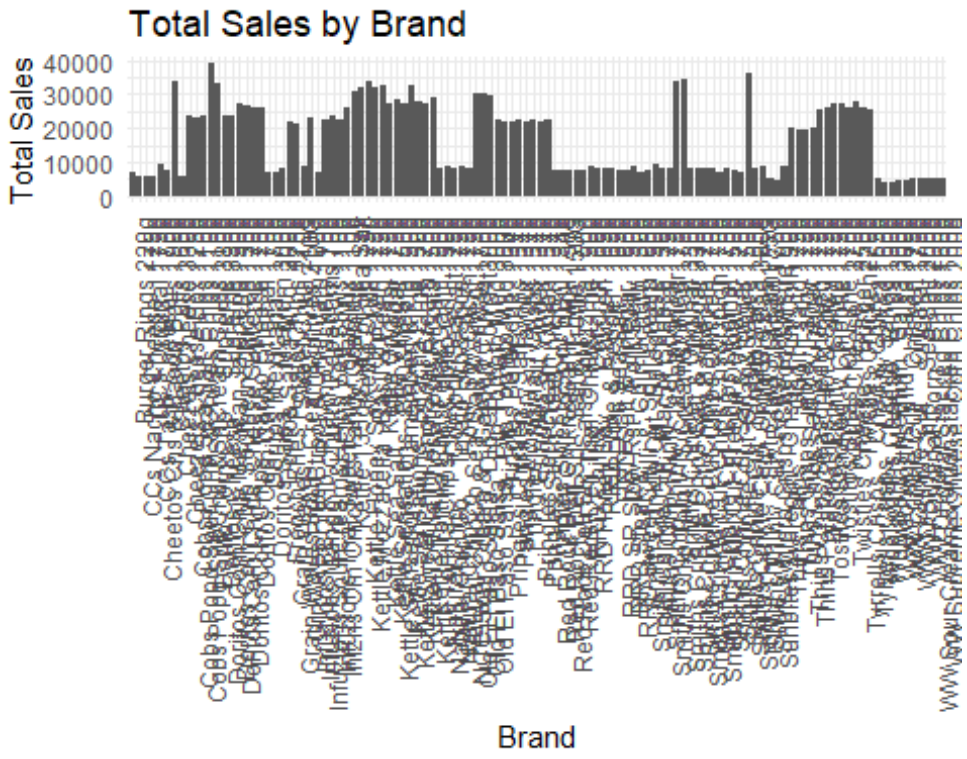
```
## [1] "T-Test Result for Trial Store 77 and Control Store 86"
##
##  Welch Two Sample t-test
##
## data:  trial_data$TOT_SALES and control_data$TOT_SALES
## t = -13.122, df = 936.07, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.788743 -1.323318
## sample estimates:
## mean of x mean of y
##  5.346346  6.902376
##
## [1] "Summary:"
##     Store Total_Sales Num_Customers Avg_Transactions_per_Customer
## 1   Trial     2999.30           355                       1.580282
## 2 Control    10602.05           273                       5.626374
```


Total Sales Comparison
10000
7500
5000
2500
0
Total Sales
Control
Trial
Store
Store
Control
Trial

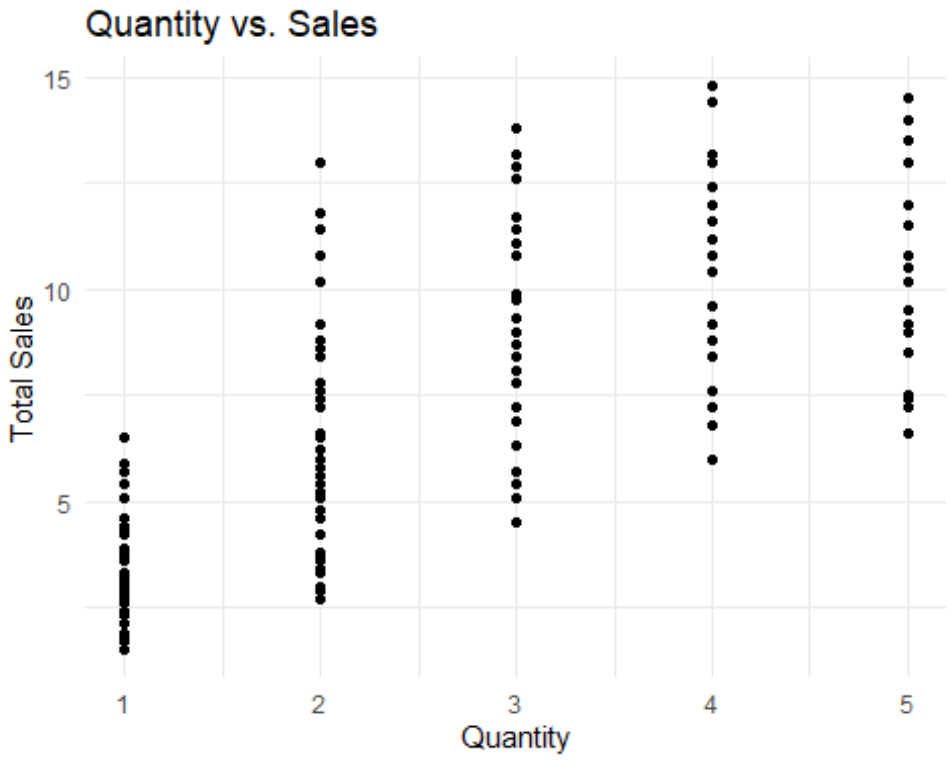

Number of Customers Comparison
300
200
100
0
Number of Customers
Control
Trial
Store
Store
Control
Trial




```r
library(ggplot2)


# Calculate total sales by brand
total_sales_by_brand <- aggregate(transaction_data$TOT_SALES, by = list(Brand
= transaction_data$PROD_NAME), FUN = sum)

# Plot total sales by brand
plot_total_sales_by_brand <- ggplot(data = total_sales_by_brand, aes(x =
Brand, y = x)) +
  geom_bar(stat = "identity") +
  labs(x = "Brand", y = "Total Sales", title = "Total Sales by Brand") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

# Print the plot
print(plot_total_sales_by_brand)
```




```r
# Plot the relationship between quantity and sales
plot_quantity_vs_sales <- ggplot(data = transaction_data, aes(x = PROD_QTY, y
= TOT_SALES)) +
  geom_point() +
  labs(x = "Quantity", y = "Total Sales", title = "Quantity vs. Sales") +
  theme_minimal()

# Print the plot
print(plot_quantity_vs_sales)
```




```r
library(ggplot2)

# Calculate total sales by store
total_sales_by_store <- aggregate(transaction_data$TOT_SALES, by = list(Store
= transaction_data$STORE_NBR), FUN = sum)

# Plot total sales by store
plot_total_sales_by_store <- ggplot(data = total_sales_by_store, aes(x =
Store, y = x)) +
  geom_bar(stat = "identity") +
  labs(x = "Store", y = "Total Sales", title = "Total Sales by Store") +
  theme_minimal()

# Print the plot
print(plot_total_sales_by_store)
```


Total Sales by Store
Total Sales
15000
10000
5000
0
0
100
200
Store