Quantium Virtual Internship - Retail Strategy and Analytics - Task 1

Kharchenko R.

2023-07-04

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

Load required libraries and datasets

```
install.packages("data.table", repos = "http://cran.us.r-project.org")

install.packages("tidyverse", repos = "http://cran.us.r-project.org")

install.packages("ggmosaic", repos = "http://cran.us.r-project.org")

install.packages("ggpubr", repos = "http://cran.us.r-project.org")

# Load required libraries
library(data.table)

library(ggplot2)

library(ggmosaic)

library(readr)

library(dplyr)

library(tidyr)

library(stringr)

library(scales)

library(ggpubr)

# Load working files
customerData <- fread("QVI_purchase_behaviour.csv")
transactionData <- fread("QVI_transaction_data.csv")
```

Exploratory data analysis (EDA)

Examine transaction data

```
# Display the first few rows of the transaction data
head(transactionData)
```

```
##     DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
## 1: 43390         1           1000      1        5
## 2: 43599         1           1307    348       66
## 3: 43605         1           1343    383       61
## 4: 43329         2           2373    974       69
## 5: 43330         2           2426   1038      108
## 6: 43604         4           4074   2982       57
##                                    PROD_NAME PROD_QTY TOT_SALES
## 1:   Natural Chip        Compny SeaSalt175g        2       6.0
## 2:                 CCs Nacho Cheese    175g        3       6.3
## 3:   Smiths Crinkle Cut  Chips Chicken 170g        2       2.9
## 4:   Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0
## 5: Kettle Tortilla ChpsHny&Jlpno Chili 150g        3      13.8
## 6: Old El Paso Salsa   Dip Tomato Mild 300g        1       5.1
```

```
# Get the dimensions (number of rows and columns) of the transaction data
dim(transactionData)
```

```
## [1] 264836      8
```

```
# Display the structure and summary of the transaction data
str(transactionData)
```

```
## Classes 'data.table' and 'data.frame':   264836 obs. of  8 variables:
##  $ DATE          : int  43390 43599 43605 43329 43330 43604 43601 43601 43332 43330
...
##  $ STORE_NBR     : int  1 1 1 2 2 4 4 4 5 7 ...
##  $ LYLTY_CARD_NBR: int  1000 1307 1343 2373 2426 4074 4149 4196 5026 7150 ...
##  $ TXN_ID        : int  1 348 383 974 1038 2982 3333 3539 4525 6900 ...
##  $ PROD_NBR      : int  5 66 61 69 108 57 16 24 42 52 ...
##  $ PROD_NAME     : chr  "Natural Chip        Compny SeaSalt175g" "CCs Nacho Cheese
175g" "Smiths Crinkle Cut  Chips Chicken 170g" "Smiths Chip Thinly  S/Cream&Onion 175g"
...
##  $ PROD_QTY      : int  2 3 2 5 3 1 1 1 1 2 ...
##  $ TOT_SALES     : num  6 6.3 2.9 15 13.8 5.1 5.7 3.6 3.9 7.2 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Observations

- The transaction data contains a total of 264,836 observations (rows) and 8 variables (columns).
- The “DATE” variable is represented as an integer (it might be a numeric representation of the date), and we will need to transform it into a more usable format.
- The “STORE_NBR”, “LYLTY_CARD_NBR”, “TXN_ID”, and “PROD_NBR” variables are represented as integers, indicating unique identifiers for the store, loyalty card number, transaction ID, and the product number, respectively.
- The “PROD_NAME” variable is represented as a character (string) and contains the names of the purchased products.
- The “PROD_QTY” variable is represented as an integer and indicates the quantity of the purchased product in each transaction.
- The “TOT_SALES” variable is represented as a numeric (floating-point) value and represents the total sales amount for each transaction.

Transform DATE column into date format

```
# Since the "QVI_transaction_data.csv" dataset uses the 1900 date system
transactionData[, DATE := as.Date(DATE, origin = "1899-12-30")]
head(transactionData)
```

```
##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
## 1: 2018-10-17         1           1000      1        5
## 2: 2019-05-14         1           1307    348       66
## 3: 2019-05-20         1           1343    383       61
## 4: 2018-08-17         2           2373    974       69
## 5: 2018-08-18         2           2426   1038      108
## 6: 2019-05-19         4           4074   2982       57
##                                    PROD_NAME PROD_QTY TOT_SALES
## 1:   Natural Chip        Compny SeaSalt175g        2       6.0
## 2:                 CCs Nacho Cheese    175g        3       6.3
## 3:   Smiths Crinkle Cut  Chips Chicken 170g        2       2.9
## 4:   Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0
## 5: Kettle Tortilla ChpsHny&Jlpno Chili 150g        3      13.8
## 6: Old El Paso Salsa   Dip Tomato Mild 300g        1       5.1
```

Examine PROD_NAME column

```
# Generate a summary of the `PROD_NAME` column
product_summary <- table(transactionData$PROD_NAME)
head(product_summary)

## 
##              Burger Rings 220g       CCs Nacho Cheese    175g 
##                           1564                           1498 
##              CCs Original 175g       CCs Tasty Cheese    175g 
##                           1514                           1539 
## Cheetos Chs & Bacon Balls 190g             Cheetos Puffs 165g 
##                           1479                           1448

# Get the number of unique values in `PROD_NAME`
num_unique_products <- length(unique(transactionData$PROD_NAME))
print(num_unique_products)

## [1] 114
```

- Since we are only interested in data related to potato chips, let's check the column to see if its values correspond to chips or if there are other products too.

```
# Further examine `PROD_NAME`
productWords <- data.table(unlist(strsplit(unique(transactionData[, PROD_NAME]), " ")))
setnames(productWords, 'words')
```

- Now, as we are only interested in words that will tell us if the product is Chips or not, let's remove all words with digits and special characters such as '&' from our set of productWords

```
# Remove digits from `productWords`
productWords$words <- gsub("\\d", " ", productWords$words)

# Remove special characters from `productWords`
productWords$words <- gsub("[[:punct:]]", " ", productWords$words)

# Remove redundant last letters "g" or "G" from productWords remained after the removal
of the pack size
productWords$words <- gsub("[gG]$", "", productWords$words)

# Create a data frame with words and their frequencies
combinedString <- paste(productWords$words, collapse = " ") # Merge into a single string
variable

# Remove leading/trailing white spaces and split the combined string into words
words <- unlist(strsplit(trimws(combinedString), "\\s+"))
wordFrequency <- table(words) # Count the frequency of each word
```

```
# Create a data frame
wordCounts <- data.frame(
  word = names(wordFrequency),
  count = as.integer(wordFrequency),
  stringsAsFactors = FALSE
)

wordCounts <- wordCounts[order(-wordCounts$count), ] # Sort the data frame by count in descending order
threshold <- 5 # Set the threshold for the minimum count
filteredWordCounts <- wordCounts[wordCounts$count >= threshold, ] # Filter the data frame based on the count condition
print(filteredWordCounts) # Print only words with the count of 5 or more

##          word count
## 32      Chips    21
## 152    Smiths    16
## 50    Crinkle    14
## 57        Cut    14
## 87     Kettle    13
## 21     Cheese    12
## 140      Salt    12
## 114 Original    10
## 29       Chip     9
## 63    Doritos     9
## 139     Salsa     9
## 24    Chicken     8
## 47       Corn     8
## 49      Cream     8
## 128 Pringles     8
## 136       RRD     8
## 27     Chilli     7
## 198        WW     7
## 144       Sea     6
## 156      Sour     6
## 52     Crisps     5
## 180    Thinly     5
## 181     Thins     5
## 191   Vinegar     5
```

Remove redundant products

- Since we are only interested in the chips category, let's remove salsa products from our transaction data

```
# Remove salsa products
transactionData[, SALSA := grepl("salsa", tolower(PROD_NAME))]
transactionData <- transactionData[SALSA == FALSE, ][, SALSA := NULL]
```

Check for missing values in the data

```
# Check for missing values in each column
missing_values <- sapply(transactionData, function(x) sum(is.na(x)))
print(missing_values) # Print the column names and their corresponding missing value counts

##           DATE      STORE_NBR LYLTY_CARD_NBR         TXN_ID       PROD_NBR
##              0              0              0              0              0
##      PROD_NAME       PROD_QTY      TOT_SALES
##              0              0              0
```

Observations

- It can be seen we have no missing values in any column.
- The DATE column requires a separate analysis and we will do that later.
- Columns such as STORE_NBR, LYLTY_CARD_NBR, TXN_ID, and PROD_NBRare numeric but they are also identifiers, so their values of the min, max or mean will not bring really meaningful information for our analysis. But in order to have an idea of the distribution of data in these columns, we will build some visuals.

Check for unique values in the columns-identifiers

```
storeCount <- length(unique(transactionData$STORE_NBR)) # Count unique values in
STORE_NBR
loyaltyCount <- length(unique(transactionData$LYLTY_CARD_NBR)) # Count unique values in
LYLTY_CARD_NBR
txnCount <- length(unique(transactionData$TXN_ID)) # Count unique values in TXN_ID
prodCount <- length(unique(transactionData$PROD_NBR)) # Count unique values in PROD_NBR

# Create a data frame with the counts
uniqueCounts <- data.frame(Column = c("STORE_NBR", "LYLTY_CARD_NBR", "TXN_ID",
"PROD_NBR"),
                           Count = c(storeCount, loyaltyCount, txnCount, prodCount))

print(uniqueCounts)

##           Column  Count
## 1      STORE_NBR    271
## 2 LYLTY_CARD_NBR  71288
## 3         TXN_ID 245257
## 4       PROD_NBR    105
```

Important note

- It was revealed that the number of records in the transaction data is 264836, while the number of unique transaction IDs in the TXN_ID is 245257 and this fact most likely indicates the presence of duplicate rows in the TXN_ID column. Let’s check this.

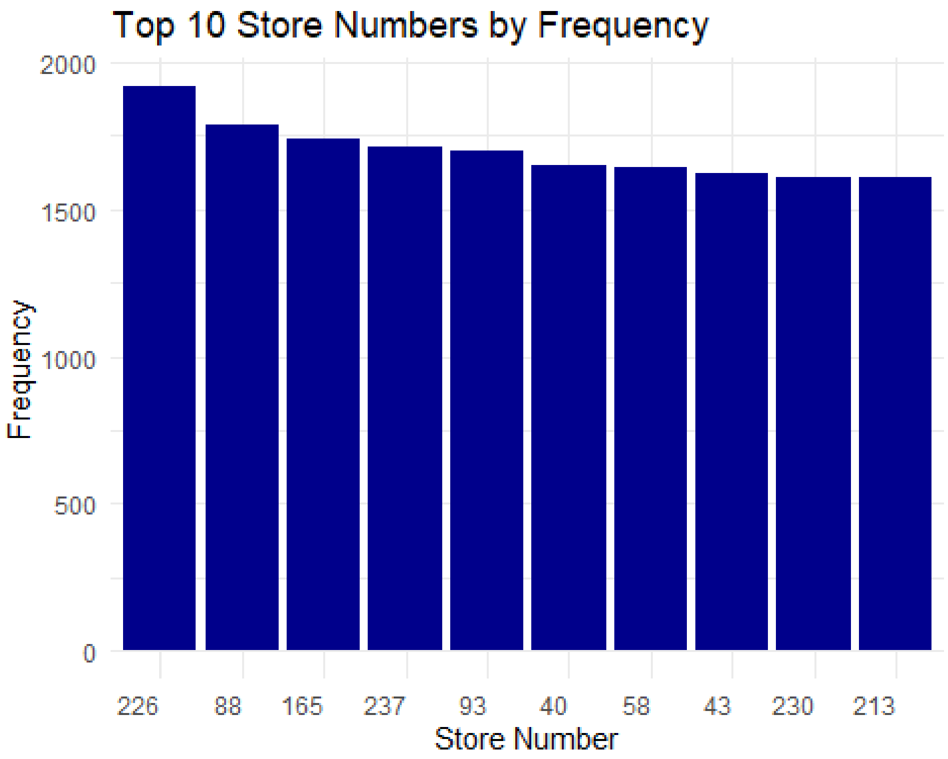
```
# Check for duplicate rows based on TXN_ID
duplicated_rows <- transactionData[duplicated(TXN_ID), ]
num_duplicate_rows <- nrow(duplicated_rows) # Count the number of duplicate rows
print(num_duplicate_rows)

## [1] 1485
```

Observations

- The presence of duplicate rows in the TXN_ID column indicates that there are repeated transaction IDs in the transaction data.This could be due to various reasons, such as:
  - data entry errors or multiple entries for the same transaction
  - multiple products were purchased within the same transaction, and each product was recorded as a separate row; in such cases, it is common practice to have one transaction ID associated with multiple product records.

Build a visual for STORE_NBR column

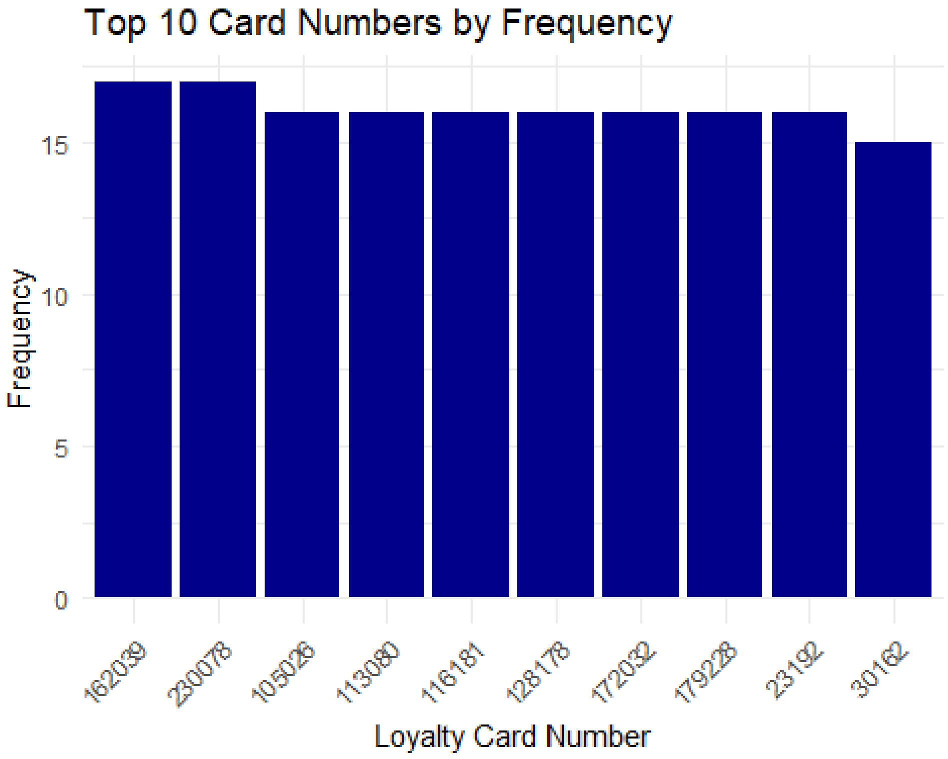
```
store_counts <- table(transactionData$STORE_NBR) # Count the frequency of each store
number

# Create a data frame with store numbers and their frequencies
store_data <- data.frame(
  STORE_NBR = as.integer(names(store_counts)),
  Frequency = as.integer(store_counts)
)

store_data <- store_data[order(-store_data$Frequency), ] # Sort the data frame by
frequency in descending order
top_10_stores <- store_data[1:10, ] # Select the top 10 store numbers

# Create a vertical column chart for top 10 store numbers
ggplot(top_10_stores, aes(x = reorder(as.character(STORE_NBR), -Frequency), y =
Frequency)) +
  geom_col(fill = "darkblue") +
  labs(x = "Store Number", y = "Frequency") +
  ggtitle("Top 10 Store Numbers by Frequency") +
  theme_minimal() +
  theme(axis.text.x = element_text(hjust = 1))
```



Build a visual for LYLTY_CARD_NBR column

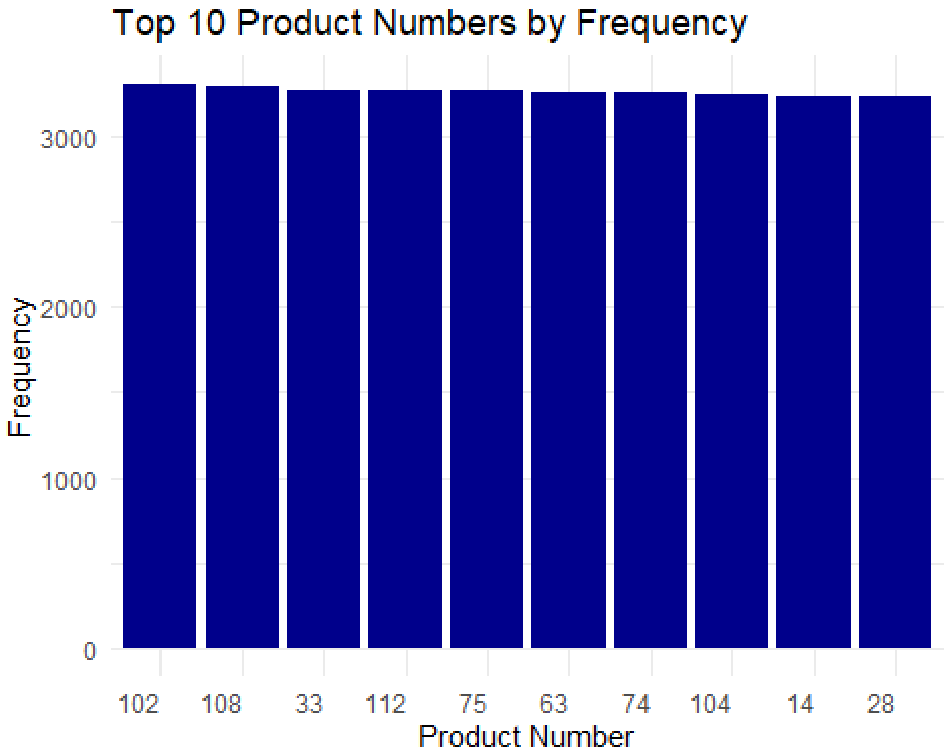
```
card_counts <- table(transactionData$LYLTY_CARD_NBR) # Count the frequency of each
loyalty card number

# Create a data frame with card numbers and their frequencies
card_data <- data.frame(
  LYLTY_CARD_NBR = as.integer(names(card_counts)),
  Frequency = as.integer(card_counts)
)
```

```
card_data <- card_data[order(-card_data$Frequency), ] # Sort the data frame by frequency
in descending order
top_10_cards <- card_data[1:10, ] # Select the top 10 card numbers

# Create a vertical column chart for top 10 card numbers
ggplot(top_10_cards, aes(x = reorder(as.character(LYLTY_CARD_NBR), -Frequency), y =
Frequency)) +
  geom_col(fill = "darkblue") +
  labs(x = "Loyalty Card Number", y = "Frequency") +
  ggtitle("Top 10 Card Numbers by Frequency") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Build a visual for PROD_NBR column

```
product_counts <- table(transactionData$PROD_NBR) # Count the frequency of each store
number

# Create a data frame with store numbers and their frequencies
product_data <- data.frame(
  PROD_NBR = as.integer(names(product_counts)),
  Frequency = as.integer(product_counts)
)

product_data <- product_data[order(-product_data$Frequency), ] # Sort the data frame by
frequency in descending order
top_10_products <- product_data[1:10, ] # Select the top 10 store numbers

# Create a vertical column chart for top 10 store numbers
ggplot(top_10_products, aes(x = reorder(as.character(PROD_NBR), -Frequency), y =
Frequency)) +
  geom_col(fill = "darkblue") +
  labs(x = "Product Number", y = "Frequency") +
  ggtitle("Top 10 Product Numbers by Frequency") +
  theme_minimal() +
  theme(axis.text.x = element_text(hjust = 1))
```



Get summary statistics and visuals for PROD_QTY and TOT_SALES columns

```
# Display summary statistics of the transaction data for those columns for which it makes
sense
summary(transactionData[, c("PROD_QTY", "TOT_SALES")])

##     PROD_QTY          TOT_SALES
##  Min.   :  1.000   Min.   :  1.700
##  1st Qu.:  2.000   1st Qu.:  5.800
##  Median :  2.000   Median :  7.400
##  Mean   :  1.908   Mean   :  7.321
##  3rd Qu.:  2.000   3rd Qu.:  8.800
##  Max.   :200.000   Max.   :650.000
```

Observations

- The total sales amounts vary widely, with a maximum value of 650.0. This indicates the presence of some transactions with high sales values.
- The PROD_QTY and TOT_SALES columns show a significant gap between the values of the third quartile and the maximum value, which indicates the presence of outliers.

Handle with outliers

```
# Filter the dataset to find the outliers and corresponding records
outliers <- transactionData[transactionData$PROD_QTY == max(transactionData$PROD_QTY), ]
print(outliers)

##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
## 1: 2018-08-19       226         226000 226201        4
## 2: 2019-05-20       226         226000 226210        4
##                           PROD_NAME PROD_QTY TOT_SALES
## 1: Dorito Corn Chp     Supreme 380g      200       650
## 2: Dorito Corn Chp     Supreme 380g      200       650
```

Observations

- As we can see there are two transactions where 200 packets of chips are bought in one transaction, and both of these transactions were made by the same customer 9 months apart.

Check if the identified customer has had other transactions

```
customerTransactions <- transactionData[LYLTY_CARD_NBR == 226000, ]
print(customerTransactions)
```

```
##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
## 1: 2018-08-19       226         226000 226201        4
## 2: 2019-05-20       226         226000 226210        4
##                           PROD_NAME PROD_QTY TOT_SALES
## 1: Dorito Corn Chp     Supreme 380g      200       650
## 2: Dorito Corn Chp     Supreme 380g      200       650
```

Observations

- The output indicates that the customer made two unusually large purchases of 200 packets of “Dorito Corn Chp Supreme 380g” in both transactions in the same store 226 for some reason.
- But for the sake of our analysis, we remove this loyalty card number from further consideration.

Filter out the customer based on the loyalty card number

```
transactionData <- transactionData[LYLTY_CARD_NBR != 226000, ]
```

Display summary statistics of the transaction data

```
# In terms of statistics and common sense, only two columns are of interest
summary(transactionData[, c("PROD_QTY", "TOT_SALES")])
```

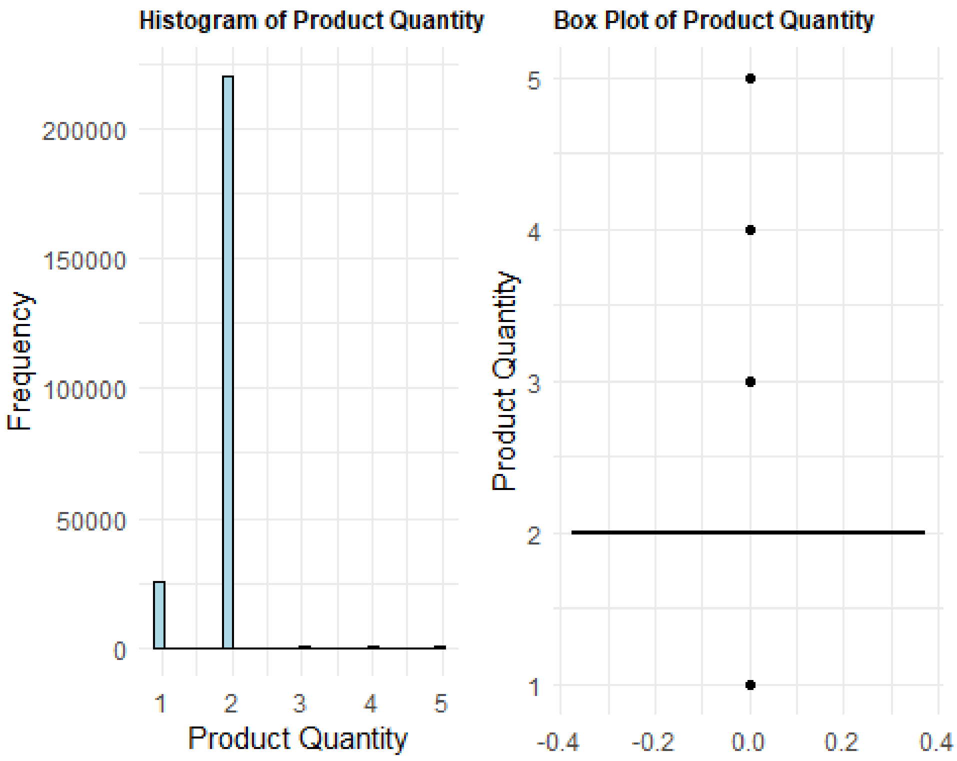
```
##     PROD_QTY       TOT_SALES
##  Min.   :1.000   Min.   : 1.700
##  1st Qu.:2.000   1st Qu.: 5.800
##  Median :2.000   Median : 7.400
##  Mean   :1.906   Mean   : 7.316
##  3rd Qu.:2.000   3rd Qu.: 8.800
##  Max.   :5.000   Max.   :29.500
```

```
# Histogram
hist_plot_QTY <- ggplot(transactionData, aes(x = PROD_QTY)) +
  geom_histogram(fill = "lightblue", color = "black") +
  labs(x = "Product Quantity", y = "Frequency", title = "Histogram of Product Quantity")
+
  theme_minimal() +
  theme(plot.title = element_text(size = 9, face = "bold"))

# Box Plot
bar_plot_QTY <- ggplot(transactionData, aes(y = PROD_QTY)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(y = "Product Quantity", title = "Box Plot of Product Quantity") +
  theme_minimal() +
  theme(plot.title = element_text(size = 9, face = "bold"))
```

```
plot_grid = ggarrange(hist_plot_QTY, bar_plot_QTY, ncol = 2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

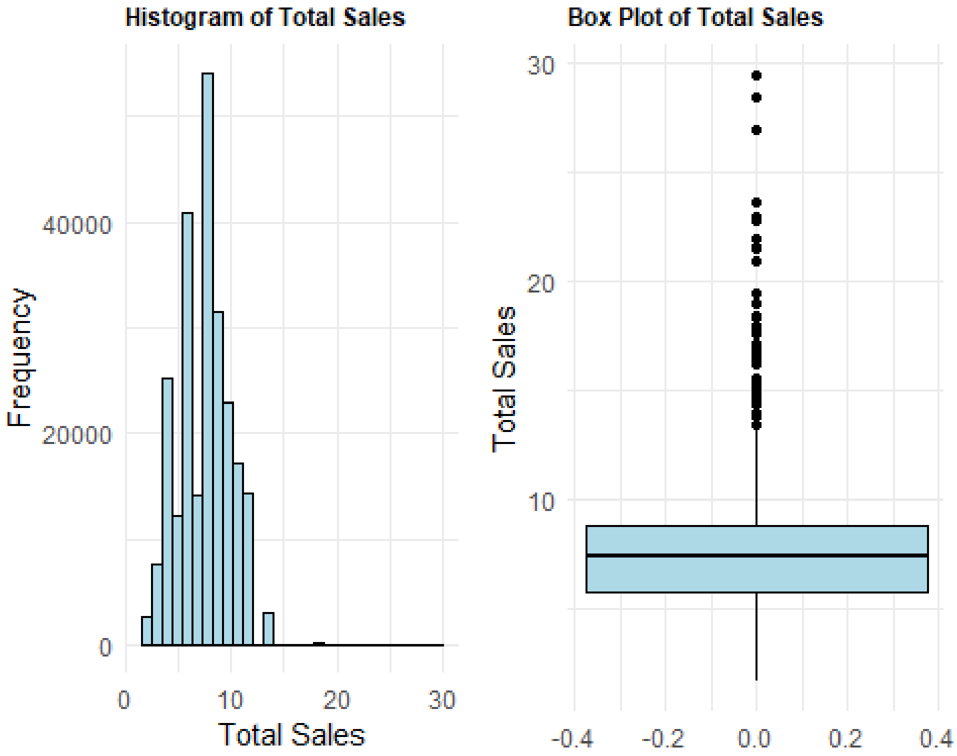
```
plot_grid
```



```
# Histogram
hist_plot_SALES <- ggplot(transactionData, aes(x = TOT_SALES)) +
  geom_histogram(fill = "lightblue", color = "black") +
  labs(x = "Total Sales", y = "Frequency", title = "Histogram of Total Sales") +
  theme_minimal() +
  theme(plot.title = element_text(size = 9, face = "bold"))

# Box Plot
bar_plot_SALES <- ggplot(transactionData, aes(y = TOT_SALES)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(y = "Total Sales", title = "Box Plot of Total Sales") +
  theme_minimal() +
  theme(plot.title = element_text(size = 9, face = "bold"))

plot_grid = ggarrange(hist_plot_SALES, bar_plot_SALES, ncol = 2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
plot_grid
```



Observations

- The average product quantity per transaction is less than 2, suggesting that customers tend to purchase small quantities of products.
- The distribution of product quantities is skewed towards smaller values, as indicated by the median, quartiles and max value.
- The total sales amounts range from 1.7 to 29.5
- The average total sales is approximately 7.316, which suggests a moderate level of sales.

Get the number of transactions over time

```r
# Count the number of transactions by date
transactionCount <- transactionData[, .N, by = DATE]
summary(transactionCount) # Create a summary of transaction count by date
```

```
##       DATE                  N        
##  Min.   :2018-07-01   Min.   :607.0  
##  1st Qu.:2018-09-29   1st Qu.:658.0  
##  Median :2018-12-30   Median :674.0  
##  Mean   :2018-12-30   Mean   :677.9  
##  3rd Qu.:2019-03-31   3rd Qu.:694.2  
##  Max.   :2019-06-30   Max.   :865.0
```

Observations

- The dataset contains transactions spanning from July 1, 2018, to June 30, 2019.
- The number of transactions per date ranges from a minimum of 607 to a maximum of 865.
- The median number of transactions per date is 674, indicating that approximately half of the dates have transaction counts below this value. The mean number of transactions is slightly higher at 677.9.
- The interquartile range (IQR) of the transaction count is relatively narrow, ranging from 658 to 694.2. This suggests that the majority of dates have a relatively consistent number of transactions.

Handle with missing dates

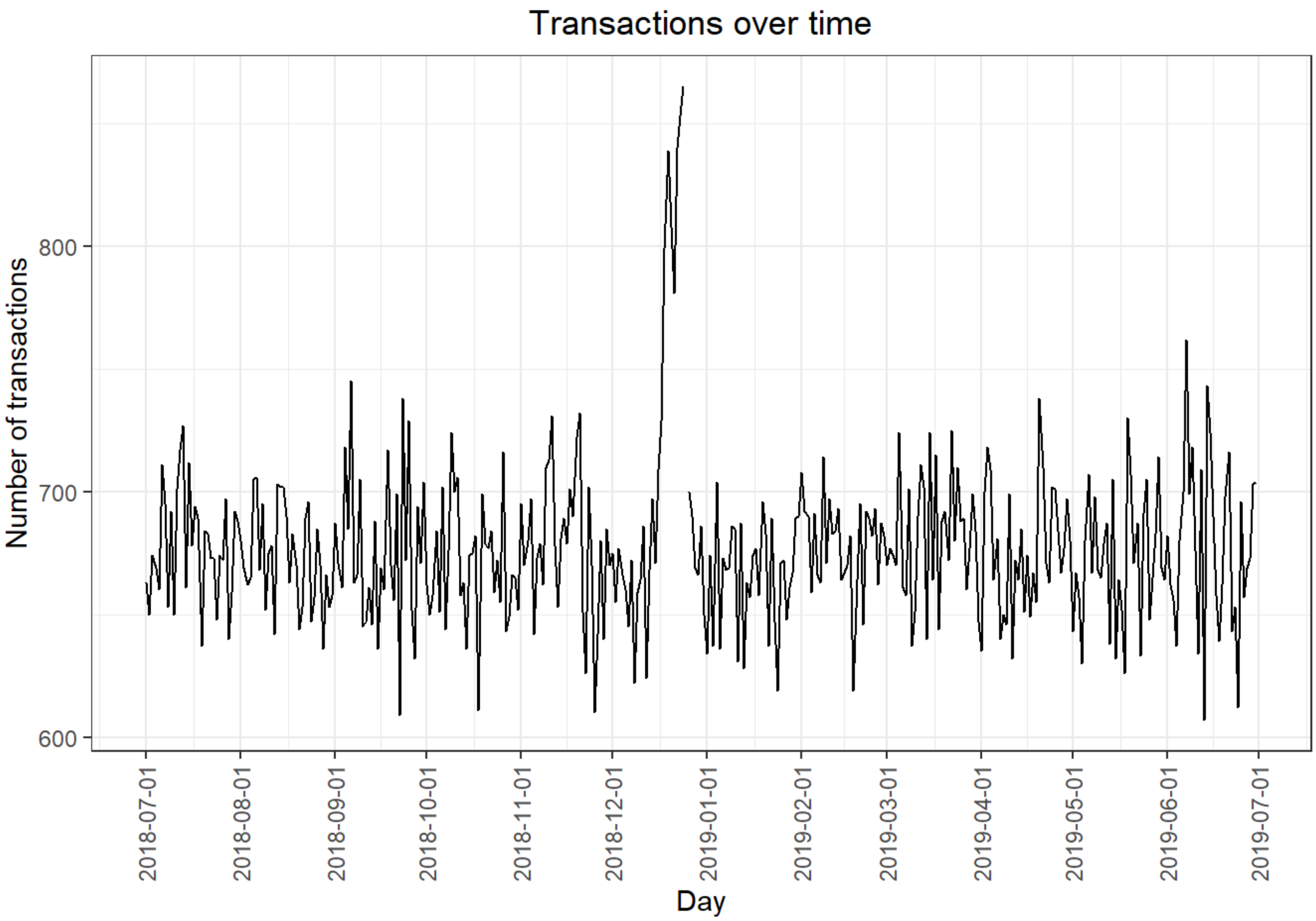
```
# Define the date range
start_date <- as.Date("2018-07-01")
end_date <- as.Date("2019-06-30")
date_range <- seq(start_date, end_date, by = "day")
unique_dates <- unique(transactionCount$DATE) # Get the unique dates in the dataset
contains_all_dates <- all(date_range %in% unique_dates) # Check if the date range
contains all dates in a row

# Print the result
if (contains_all_dates) {
  cat("The date range contains all dates in a row.\n")
} else {
  cat("The date range does not contain all dates in a row.\n")
}

## The date range does not contain all dates in a row.

# Determine the number of missing dates
total_length <- length(date_range) # Calculate the total length of the date range
unique_dates_count <- length(unique_dates) # Calculate the number of unique dates in the
dataset
missing_dates_count <- total_length - unique_dates_count # Calculate the number of
missing dates

# Print the results
cat("Total length of date range:", total_length, "\n")

## Total length of date range: 365

cat("Number of missing dates:", missing_dates_count, "\n")

## Number of missing dates: 1
```

Fill in the missing day

```
# Create a calendar table for the date range
calendar <- data.frame(DATE = seq(as.Date("2018-07-01"), as.Date("2019-06-30"), by =
"day"))

# Perform left join and add rows for missing dates
transactions_by_day <- complete(transactionCount, calendar)
```

Create a chart of number of transactions over time

```
# Set plot themes to format graphs
theme_set(theme_bw())
theme_update(plot.title = element_text(hjust = 0.5))

# Plot transactions over time
ggplot(transactions_by_day, aes(x = DATE, y = N)) +
geom_line() +
labs(x = "Day", y = "Number of transactions", title = "Transactions over time") +
scale_x_date(breaks = "1 month") +
theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```

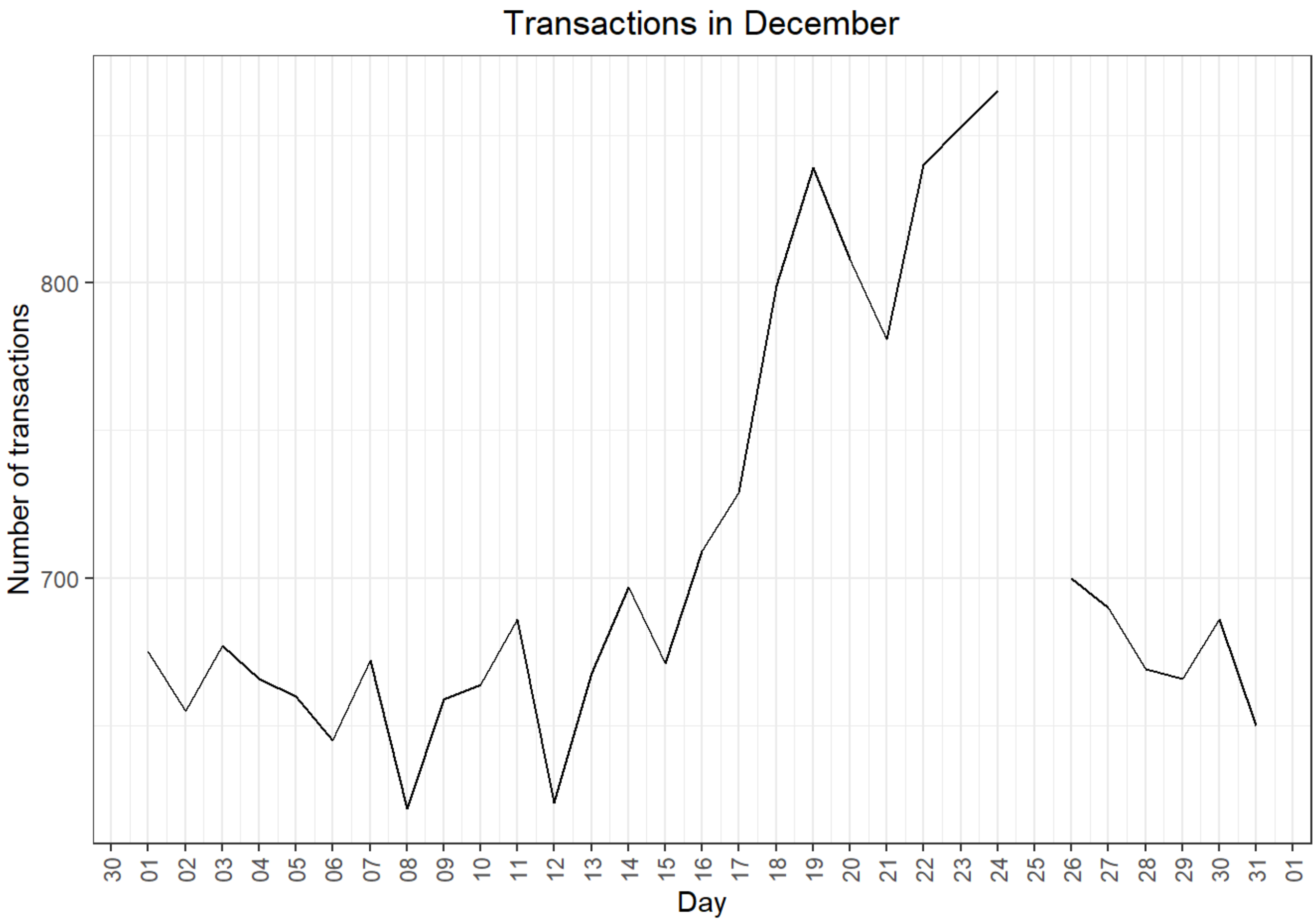


Observations

- We can observe that throughout the year the data shows more or less uniform sales, however, there is an increase in purchases in December and a break in late December. Let’s zoom in on this.

Filter data for December

```
december_transactions <- transactions_by_day %>%
  filter(format(DATE, "%m") == "12")

# Create a zoomed-in chart for the relevant dates
ggplot(december_transactions, aes(x = DATE, y = N)) +
  geom_line() +
  labs(x = "Day", y = "Number of transactions", title = "Transactions in December") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_text(angle = 90, vjust = 0.5)) +
  scale_x_date(date_breaks = "1 day", date_labels = "%d")
```



Observations

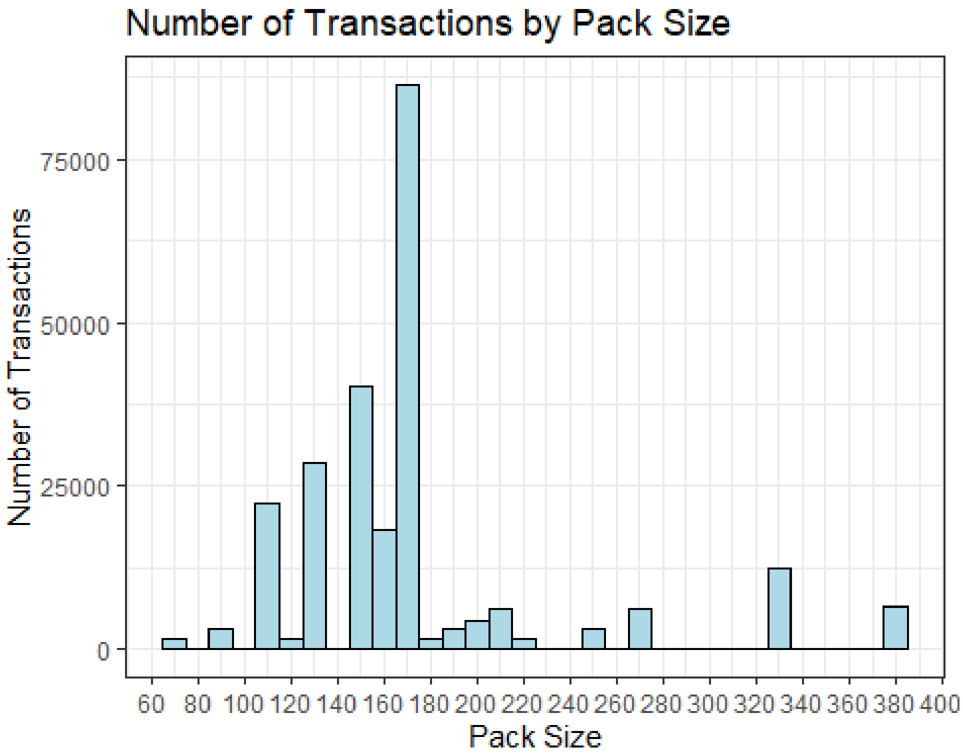
- The chart reveals a noticeable increase in sales during the period from December 15 to December 24, which corresponds to the lead-up to Christmas.
- This suggests that customers tend to make more purchases in preparation for the holiday season.
- On Christmas day itself (December 25), there is a gap in transactions due to the fact that most shops are closed on Christmas day.

Create a pack size feature from PROD_NAME

```
# Handle this task by taking the digits that are in `PROD_NAME` column
transactionData[, PACK_SIZE := parse_number(PROD_NAME)]

# Check if the pack sizes look sensible
transactionData[, .N, PACK_SIZE][order(PACK_SIZE)]

##     PACK_SIZE     N
##  1:        70  1507
##  2:        90  3008
##  3:       110 22387
##  4:       125  1454
##  5:       134 25102
##  6:       135  3257
##  7:       150 40203
##  8:       160  2970
##  9:       165 15297
## 10:       170 19983
## 11:       175 66390
## 12:       180  1468
## 13:       190  2995
## 14:       200  4473
## 15:       210  6272
## 16:       220  1564
## 17:       250  3169
## 18:       270  6285
## 19:       330 12540
## 20:       380  6416
```

- Let’s plot a histogram of PACK_SIZE since we know that this variable combines the properties of a categorical and a numerical variable

```
# Plot histogram of transactions by pack size
ggplot(transactionData, aes(x = PACK_SIZE)) +
  geom_histogram(binwidth = 10, fill = "lightblue", color = "black") +
  scale_x_continuous(breaks = seq(40, 400, by = 20)) +
  labs(x = "Pack Size", y = "Number of Transactions", title = "Number of Transactions by
Pack Size") +
  theme_bw()
```



Observations

- The “PACK_SIZE” feature represents different pack sizes of products.
- The pack sizes range from 70 to 380.
- The most common pack sizes are 175, 150, 134, and 170, as they have the highest counts.
- Pack sizes 70, 125, 180, and 220 have the lowest counts of occurrences (<1600).

```
head(transactionData)
```

```
##           DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
## 1: 2018-10-17         1           1000      1        5
## 2: 2019-05-14         1           1307    348       66
## 3: 2019-05-20         1           1343    383       61
## 4: 2018-08-17         2           2373    974       69
## 5: 2018-08-18         2           2426   1038      108
## 6: 2019-05-16         4           4149   3333       16
##                                   PROD_NAME PROD_QTY TOT_SALES PACK_SIZE
## 1:   Natural Chip        Compny SeaSalt175g        2       6.0       175
## 2:                 CCs Nacho Cheese    175g        3       6.3       175
## 3:   Smiths Crinkle Cut  Chips Chicken 170g        2       2.9       170
## 4:   Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0       175
## 5: Kettle Tortilla ChpsHny&Jlpno Chili 150g        3      13.8       150
## 6: Smiths Crinkle Chips Salt & Vinegar 330g        1       5.7       330
```

Create a brand name feature from PROD_NAME

```
# Extract the brand name from the product name
transactionData[, BRAND := toupper(substr(PROD_NAME, 1, regexpr(pattern = ' ', PROD_NAME)
-1))]

transactionData[, .N, BRAND][order(-N)] # Check if the brand names look sensible
```

```
##           BRAND     N
##  1:      KETTLE 41288
##  2:      SMITHS 27390
##  3:    PRINGLES 25102
##  4:     DORITOS 22041
##  5:       THINS 14075
##  6:         RRD 11894
##  7:   INFUZIONS 11057
##  8:          WW 10320
##  9:        COBS  9693
## 10:    TOSTITOS  9471
## 11:    TWISTIES  9454
## 12:    TYRRELLS  6442
## 13:       GRAIN  6272
## 14:     NATURAL  6050
## 15:    CHEEZELS  4603
## 16:         CCS  4551
## 17:         RED  4427
## 18:      DORITO  3183
## 19:      INFZNS  3144
## 20:       SMITH  2963
## 21:     CHEETOS  2927
## 22:       SNBTS  1576
## 23:      BURGER  1564
## 24:  WOOLWORTHS  1516
## 25:     GRNWVES  1468
## 26:    SUNBITES  1432
## 27:         NCC  1419
## 28:      FRENCH  1418
##           BRAND     N
```

Observations

- The “BRAND” feature and its values look reasonable.
- At the same time, some brands may be spelled differently, either with full words, with an abbreviation, or just cropped, but they are the same brands. So we need to make some adjustments for BRAND:

- concatenate values of DORITOS and DORITO, and delete DORITO
- concatenate values of GRAIN and GRNWVES, and delete GRNWVES
- concatenate values of INFUZIONS and INFZNS, and delete INFZNS
- concatenate values of NATURAL and NCC, and delete NCC
- concatenate values of RRD and RED, and delete RED
- concatenate values of SMITHS and SMITH, and delete SMITH
- concatenate values of SUNBITES and SNBTS, and delete SNBTS
- concatenate values of WOOLWORTHS and WW, and delete WW

```r
# Make adjustments to brand names
transactionData[BRAND == "DORITO", BRAND := "DORITOS"]
transactionData[BRAND == "GRAIN", BRAND := "GRNWVES"]
transactionData[BRAND == "INFZNS", BRAND := "INFUZIONS"]
transactionData[BRAND == "NCC", BRAND := "NATURAL"]
transactionData[BRAND == "RED", BRAND := "RRD"]
transactionData[BRAND == "SMITH", BRAND := "SMITHS"]
transactionData[BRAND == "SNBTS", BRAND := "SUNBITES"]
transactionData[BRAND == "WW", BRAND := "WOOLWORTHS"]

# Recalculate the number of transactions by the adjusted brand names
brandCounts <- transactionData[, .N, by = BRAND][order(-N)]
print(brandCounts)
```

```
##          BRAND     N
##  1:     KETTLE 41288
##  2:     SMITHS 30353
##  3:    DORITOS 25224
##  4:   PRINGLES 25102
##  5:        RRD 16321
##  6:  INFUZIONS 14201
##  7:      THINS 14075
##  8: WOOLWORTHS 11836
##  9:       COBS  9693
## 10:   TOSTITOS  9471
## 11:   TWISTIES  9454
## 12:    GRNWVES  7740
## 13:    NATURAL  7469
## 14:   TYRRELLS  6442
## 15:   CHEEZELS  4603
## 16:        CCS  4551
## 17:   SUNBITES  3008
## 18:    CHEETOS  2927
## 19:     BURGER  1564
## 20:     FRENCH  1418
```

Examine customer data

```r
# Display the first few rows of the customer data
head(customerData)
```

```
##    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
```

```r
# Get the dimensions (number of rows and columns) of the customer data
dim(customerData)
```

```
## [1] 72637     3
```

```r
# Display the structure and summary of the customer data
str(customerData)
```

```
## Classes 'data.table' and 'data.frame':   72637 obs. of  3 variables:
##  $ LYLTY_CARD_NBR  : int  1000 1002 1003 1004 1005 1007 1009 1010 1011 1012 ...
##  $ LIFESTAGE       : chr  "YOUNG SINGLES/COUPLES" "YOUNG SINGLES/COUPLES" "YOUNG
FAMILIES" "OLDER SINGLES/COUPLES" ...
##  $ PREMIUM_CUSTOMER: chr  "Premium" "Mainstream" "Budget" "Mainstream" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Observations

- The customer data contains 72,637 records and 3 columns: LYLTY_CARD_NBR, LIFESTAGE, and PREMIUM_CUSTOMER.
- Each record represents a customer and includes their loyalty card number, lifestage, and premium customer segment.
- The “LYLTY_CARD_NBR” values are represented as integers, indicating a unique identifier for the loyalty card number.
- The “LIFESTAGE” variable is represented as a character (string); this attribute identifies the customer’s life stage, indicating whether they have a family and at what point in life they are.
- The “PREMIUM_CUSTOMER” variable is represented as a character (string); this attribute is used for customer segmentation based on the price point and types of products purchased.

Check for missing values in the data

```r
# Check for missing values in each column
missing_values <- sapply(customerData, function(x) sum(is.na(x)))
print(missing_values) # Print the column names and their corresponding missing value
counts
```

```
##   LYLTY_CARD_NBR        LIFESTAGE PREMIUM_CUSTOMER
##                0                0                0
```

Observations

- It can be seen we have no missing values in any column.

Check for unique values in the data

```r
# Count unique values in LYLTY_CARD_NBR
loyaltyCount <- length(unique(customerData$LYLTY_CARD_NBR))

# Count unique values in LIFESTAGE
lfsCount <- length(unique(customerData$LIFESTAGE))

# Count unique values in PREMIUM_CUSTOMER
premCount <- length(unique(customerData$PREMIUM_CUSTOMER))

# Create a data frame with the counts
uniqueCounts <- data.frame(Column = c("LYLTY_CARD_NBR", "LIFESTAGE", "PREMIUM_CUSTOMER"),
                           Count = c(loyaltyCount, lfsCount, premCount))

print(uniqueCounts)

##              Column Count
## 1   LYLTY_CARD_NBR 72637
## 2        LIFESTAGE     7
## 3 PREMIUM_CUSTOMER     3
```

Important note

- It was found that the number of unique loyalty cards in the transaction data (71287) and the number of unique loyalty cards in the customer data (72637) do not match.This could occur due to various reasons, such as:

1. Some customers may have obtained loyalty cards but have not made any purchases yet.
2. Certain customers may have made purchases in the past but did not make any transactions during the specific period covered by the transaction data.
3. There could be delays or gaps in recording transaction data, resulting in some transactions not being captured for certain customers.

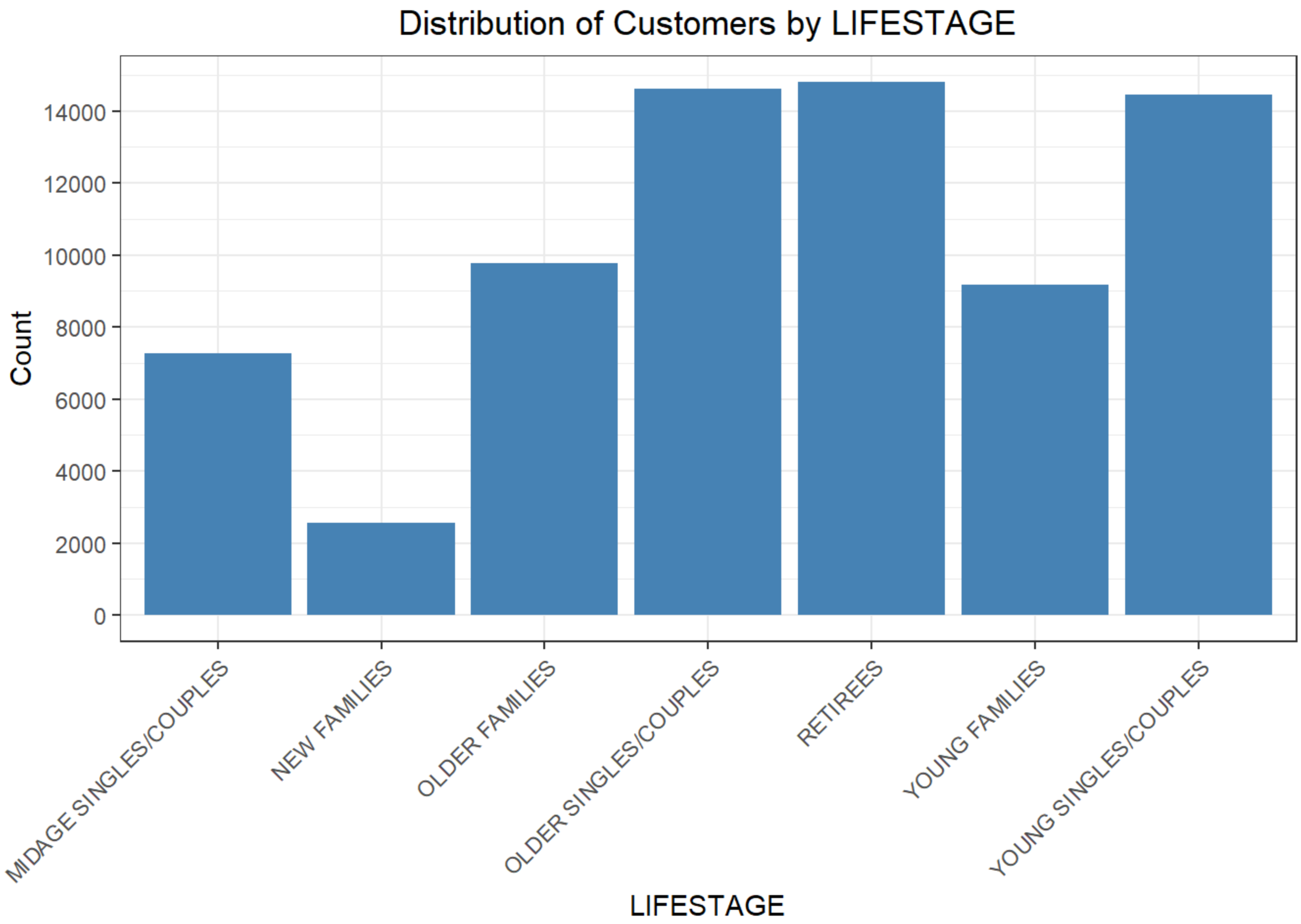
- So, it’s important to consider these possibilities when analyzing the data and drawing conclusions about customer behavior.

List the unique values in the LIFESTAGE

```
customerData[, .N, by = LIFESTAGE][order(-N)]

##                  LIFESTAGE     N
## 1:                RETIREES 14805
## 2:   OLDER SINGLES/COUPLES 14609
## 3:   YOUNG SINGLES/COUPLES 14441
## 4:          OLDER FAMILIES  9780
## 5:          YOUNG FAMILIES  9178
## 6:  MIDAGE SINGLES/COUPLES  7275
## 7:            NEW FAMILIES  2549
```

Create visual for the LIFESTAGE

```
ggplot(customerData, aes(x = LIFESTAGE)) +
  geom_bar(fill = "steelblue") +
  labs(x = "LIFESTAGE", y = "Count", title = "Distribution of Customers by LIFESTAGE") +
  scale_y_continuous(breaks = seq(0, 16000, 2000)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Observations

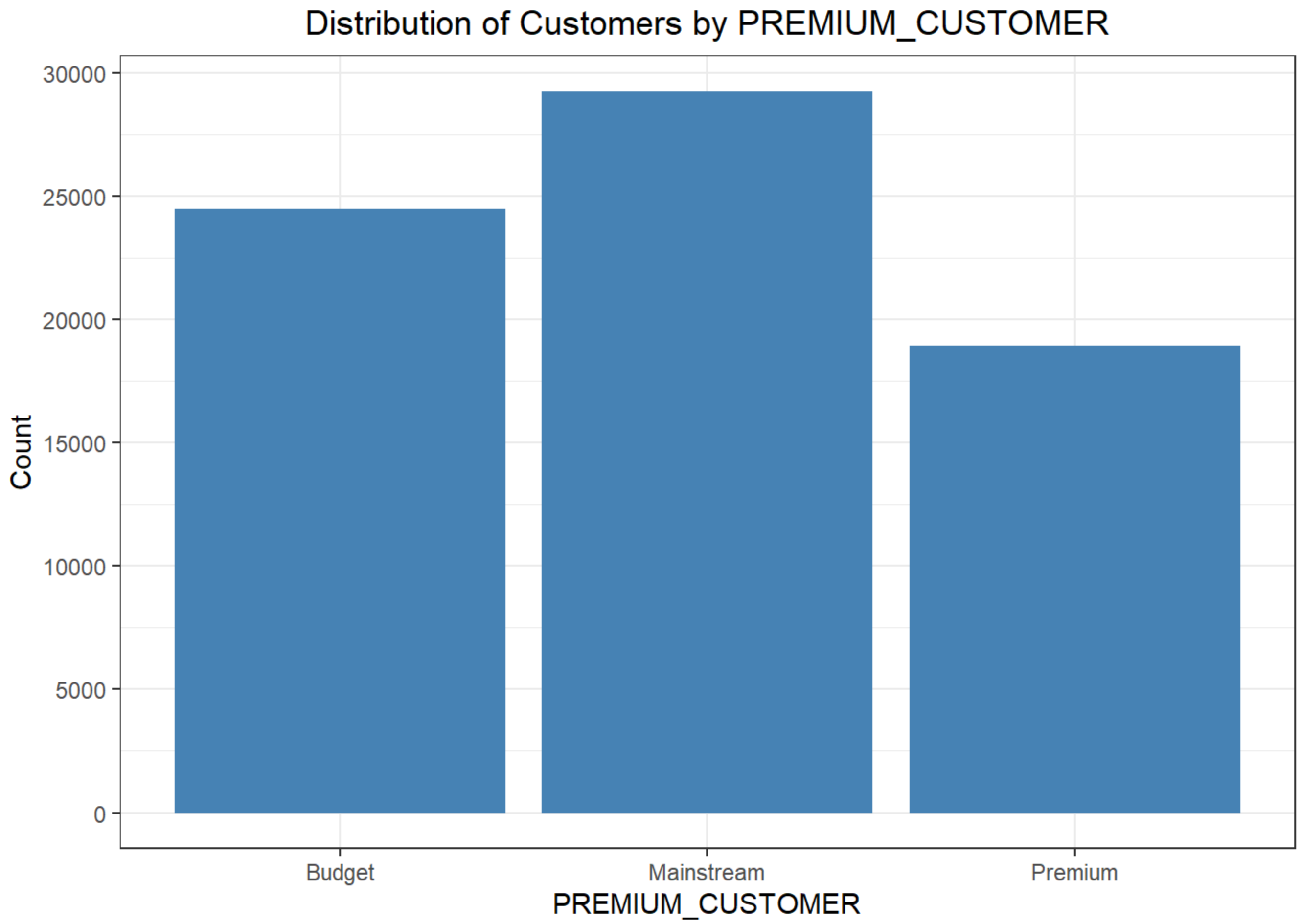
- The most common lifestage among the customers is "RETIREES" with 14,805 occurrences.
- The next two most common lifestages are "OLDER SINGLES/COUPLES" and "YOUNG SINGLES/COUPLES" with 14,609 and 14,441 occurrences, respectively.
- "NEW FAMILIES" has the lowest count of lifestage with 2,549 occurrences.

List the unique values in the PREMIUM_CUSTOMER

```
customerData[, .N, by = PREMIUM_CUSTOMER][order(-N)]

##    PREMIUM_CUSTOMER     N
## 1:       Mainstream 29245
## 2:           Budget 24470
## 3:          Premium 18922
```

Create visual for the PREMIUM_CUSTOMER

```
ggplot(customerData, aes(x = PREMIUM_CUSTOMER)) +
  geom_bar(fill = "steelblue") +
  labs(x = "PREMIUM_CUSTOMER", y = "Count", title = "Distribution of Customers by
PREMIUM_CUSTOMER") +
  scale_y_continuous(breaks = seq(0, 30000, 5000))
```



Observations

- The distribution of customers based on their premium categories indicates that the majority of customers are in the “Mainstream” category (29,245 occurrences), followed by the “Budget” category (24,470 occurrences), while the “Premium” category has the smallest customer count (18,922 occurrences).

Merge transaction data to customer data

```
mergedData <- merge(transactionData, customerData, all.x = TRUE)
head(mergedData)
```

```
##    LYLTY_CARD_NBR       DATE STORE_NBR TXN_ID PROD_NBR
## 1:           1000 2018-10-17         1      1        5
## 2:           1002 2018-09-16         1      2       58
## 3:           1003 2019-03-07         1      3       52
## 4:           1003 2019-03-08         1      4      106
## 5:           1004 2018-11-02         1      5       96
## 6:           1005 2018-12-28         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:  Grain Waves Sour    Cream&Chives 210G        1       3.6       210
## 4: Natural ChipCo      Hony Soy Chckn175g        1       3.0       175
## 5:         WW Original Stacked Chips 160g        1       1.9       160
## 6:                    Cheetos Puffs 165g        1       2.8       165
##         BRAND              LIFESTAGE PREMIUM_CUSTOMER
## 1:    NATURAL  YOUNG SINGLES/COUPLES          Premium
## 2:        RRD  YOUNG SINGLES/COUPLES       Mainstream
## 3:    GRNWVES         YOUNG FAMILIES           Budget
## 4:    NATURAL         YOUNG FAMILIES           Budget
## 5: WOOLWORTHS  OLDER SINGLES/COUPLES       Mainstream
## 6:    CHEETOS MIDAGE SINGLES/COUPLES       Mainstream
```

```
dim(mergedData)
```

```
## [1] 246740     12
```

Observations

- As a result, we merged the transaction data and customer data. The new dataset consists of 246740 records and 12 fields and contains data regarding chip sales.
- The 'all.x = TRUE' argument ensures that all rows from the transaction data are included in the merged data, even if there is no corresponding customer data.

Let's check if some customers were not matched on by checking for nulls.

```
# Check for missing values in all columns
missingValues <- colSums(is.na(mergedData))
print(missingValues) # Print the missing values count for each column
```

```
##   LYLTY_CARD_NBR             DATE        STORE_NBR           TXN_ID
##                0                0                0                0
##         PROD_NBR        PROD_NAME         PROD_QTY        TOT_SALES
##                0                0                0                0
##        PACK_SIZE            BRAND        LIFESTAGE PREMIUM_CUSTOMER
##                0                0                0                0
```

- The next code creates a new variable `missingCustomers` that contains the rows from the merged data where either the "LIFESTAGE" or "PREMIUM_CUSTOMER" columns have missing values (NA). These rows represent transactions that did not have a matched customer.

```
# Check for missing customer details
missingCustomers <- mergedData[is.na(LIFESTAGE) | is.na(PREMIUM_CUSTOMER)]
print(missingCustomers)
```

```
## Empty data.table (0 rows and 12 cols):
LYLTY_CARD_NBR,DATE,STORE_NBR,TXN_ID,PROD_NBR,PROD_NAME...
```

Observations

- We can see that all the transactions in the `transactionData` dataset were successfully matched with customer details from the `customerData` dataset, and there are no missing values in the merged dataset `mergedData`. This ensures that we have complete information for all the transactions.

```
# Save dataset as a csv in the working directory
fwrite(mergedData, file = "QVI_mergedData.csv")
```

Data analysis on customer segments

- Now that the data is ready for analysis, we can define some metrics of interest to the client (our current tasks):
  - Who spends the most on chips (total sales), describing customers by "LIFESTAGE" and how "Premium" their general purchasing behavior is?
  - How many customers are in each segment?
  - How many chips are bought per customer by segment?

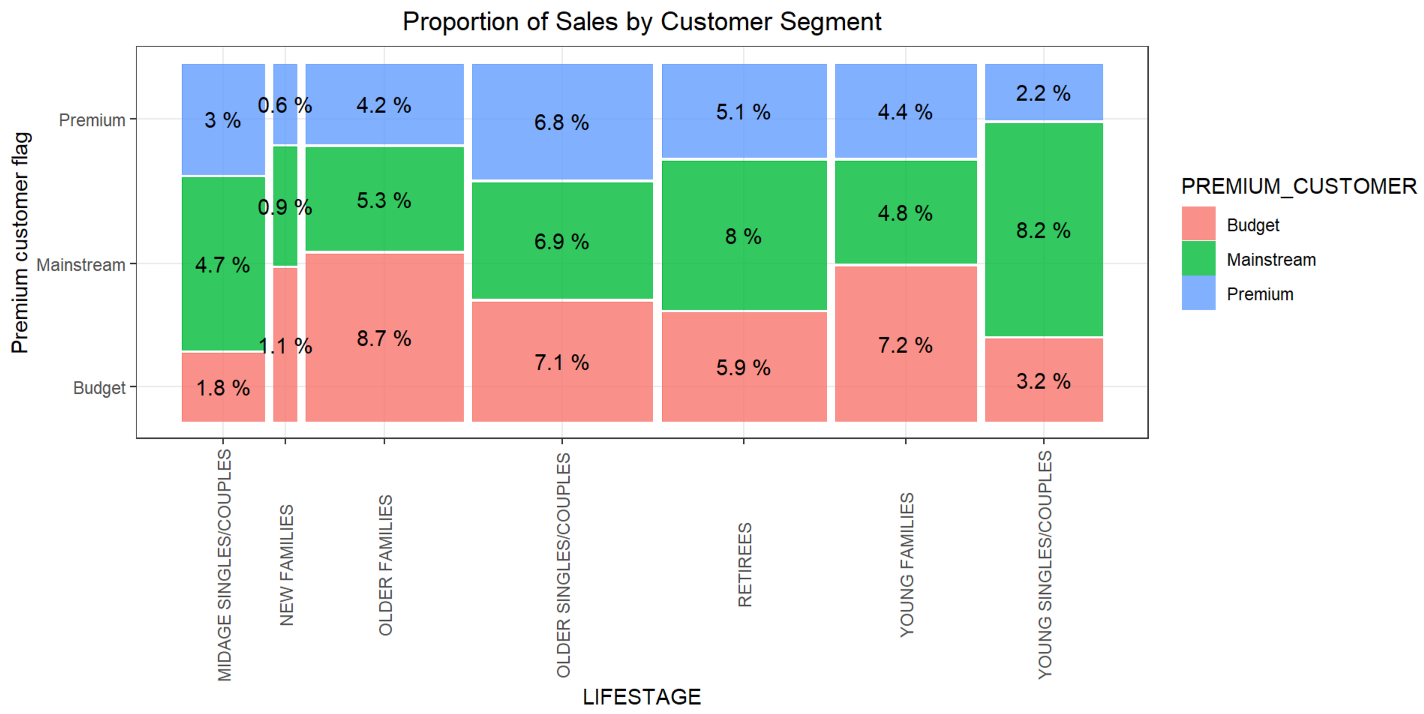
- What’s the average chip price by customer segment?

Calculate total sales by LIFESTAGE and PREMIUM_CUSTOMER

```
data <- mergedData

# Total sales by LIFESTAGE and PREMIUM_CUSTOMER
total_sales <- data[, .(Total_Sales = sum(TOT_SALES)), .(LIFESTAGE,
PREMIUM_CUSTOMER)][order(-Total_Sales)]

# Calculate the proportion of total sales and format as percentage
total_sales[, Share_of_Total_Sales := round(100 * Total_Sales / sum(Total_Sales),2)]
total_sales

##                  LIFESTAGE PREMIUM_CUSTOMER Total_Sales Share_of_Total_Sales
##  1:         OLDER FAMILIES           Budget   156863.75                 8.69
##  2:  YOUNG SINGLES/COUPLES       Mainstream   147582.20                 8.18
##  3:               RETIREES       Mainstream   145168.95                 8.04
##  4:         YOUNG FAMILIES           Budget   129717.95                 7.19
##  5:  OLDER SINGLES/COUPLES           Budget   127833.60                 7.08
##  6:  OLDER SINGLES/COUPLES       Mainstream   124648.50                 6.91
##  7:  OLDER SINGLES/COUPLES          Premium   123537.55                 6.84
##  8:               RETIREES           Budget   105916.30                 5.87
##  9:         OLDER FAMILIES       Mainstream    96413.55                 5.34
## 10:               RETIREES          Premium    91296.65                 5.06
## 11:         YOUNG FAMILIES       Mainstream    86338.25                 4.78
## 12: MIDAGE SINGLES/COUPLES       Mainstream    84734.25                 4.69
## 13:         YOUNG FAMILIES          Premium    78571.70                 4.35
## 14:         OLDER FAMILIES          Premium    75242.60                 4.17
## 15:  YOUNG SINGLES/COUPLES           Budget    57122.10                 3.16
## 16: MIDAGE SINGLES/COUPLES          Premium    54443.85                 3.02
## 17:  YOUNG SINGLES/COUPLES          Premium    39052.30                 2.16
## 18: MIDAGE SINGLES/COUPLES           Budget    33345.70                 1.85
## 19:           NEW FAMILIES           Budget    20607.45                 1.14
## 20:           NEW FAMILIES       Mainstream    15979.70                 0.89
## 21:           NEW FAMILIES          Premium    10760.80                 0.60
##                  LIFESTAGE PREMIUM_CUSTOMER Total_Sales Share_of_Total_Sales
```

Plot the split by these segments to describe which customer segment contributes most to chip sales

```
p <- ggplot(data = total_sales) +
  geom_mosaic(aes(weight = Total_Sales, x = product(PREMIUM_CUSTOMER, LIFESTAGE), fill =
PREMIUM_CUSTOMER)) +
  labs(x = "LIFESTAGE", y = "Premium customer flag", title = "Proportion of Sales by
Customer Segment") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))

# Plot and label with proportion of sales
p + geom_text(data = ggplot_build(p)$data[[1]], aes(x = (xmin + xmax)/2 , y = (ymin +
ymax)/2,
                                                    label =
as.character(paste(round(.wt/sum(.wt),3)*100,'%'))))
```



Observations

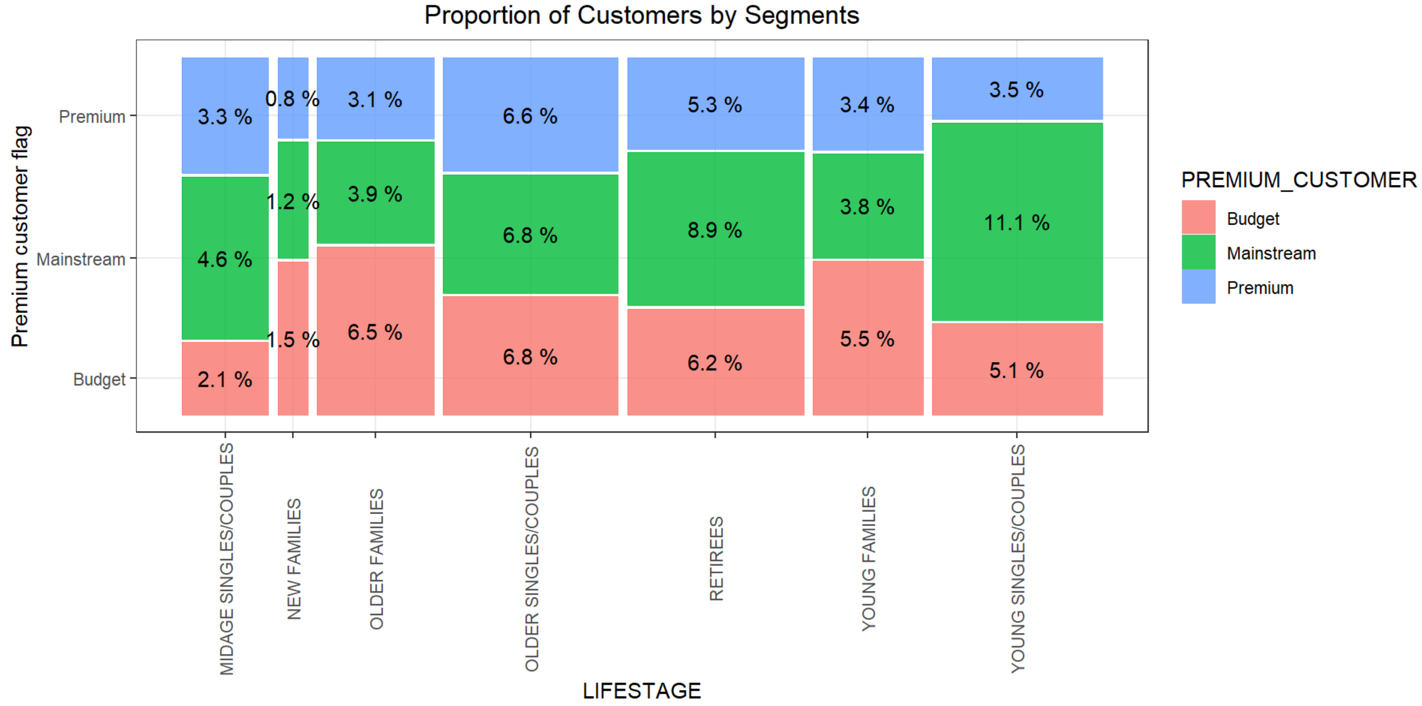
- The “OLDER FAMILIES” segment in the “Budget” category has the highest total sales of almost 157K (8.69%), making them the largest contributors to chip sales.
- The “YOUNG SINGLES/COUPLES” segment in the “Mainstream” category is the second-highest contributor with total sales of almost 148K (8.18%).
- The “RETIREES” segment in the “Mainstream” category follows closely with total sales of just over 145K (8.04%).

Check if the higher sales are due to there being more customers who buy chips

```
# Calculate the number of customers in each segment
customer_count <- data[, .(CUSTOMERS = uniqueN(LYLTY_CARD_NBR)), .(LIFESTAGE,
PREMIUM_CUSTOMER)][order(-CUSTOMERS)]

# Calculate the proportion of customer counts and format as percentage
customer_count[, Share_of_Total_Customers := round(100 * CUSTOMERS / sum(CUSTOMERS), 2)]
customer_count

##                  LIFESTAGE PREMIUM_CUSTOMER CUSTOMERS Share_of_Total_Customers
##  1:  YOUNG SINGLES/COUPLES       Mainstream      7917                    11.11
##  2:               RETIREES       Mainstream      6358                     8.92
##  3:  OLDER SINGLES/COUPLES       Mainstream      4858                     6.81
##  4:  OLDER SINGLES/COUPLES           Budget      4849                     6.80
##  5:  OLDER SINGLES/COUPLES          Premium      4682                     6.57
##  6:         OLDER FAMILIES           Budget      4611                     6.47
##  7:               RETIREES           Budget      4385                     6.15
##  8:         YOUNG FAMILIES           Budget      3953                     5.55
##  9:               RETIREES          Premium      3812                     5.35
## 10:  YOUNG SINGLES/COUPLES           Budget      3647                     5.12
## 11: MIDAGE SINGLES/COUPLES       Mainstream      3298                     4.63
## 12:         OLDER FAMILIES       Mainstream      2788                     3.91
## 13:         YOUNG FAMILIES       Mainstream      2685                     3.77
## 14:  YOUNG SINGLES/COUPLES          Premium      2480                     3.48
## 15:         YOUNG FAMILIES          Premium      2398                     3.36
## 16: MIDAGE SINGLES/COUPLES          Premium      2369                     3.32
## 17:         OLDER FAMILIES          Premium      2231                     3.13
## 18: MIDAGE SINGLES/COUPLES           Budget      1474                     2.07
## 19:           NEW FAMILIES           Budget      1087                     1.52
## 20:           NEW FAMILIES       Mainstream       830                     1.16
## 21:           NEW FAMILIES          Premium       575                     0.81
##                  LIFESTAGE PREMIUM_CUSTOMER CUSTOMERS Share_of_Total_Customers
```

Plot a visual to see the distribution of the number of customers by segments

```
p <- ggplot(data = customer_count) +
  geom_mosaic(aes(weight = CUSTOMERS, x = product(PREMIUM_CUSTOMER,LIFESTAGE), fill =
PREMIUM_CUSTOMER)) +
  labs(x = "LIFESTAGE", y = "Premium customer flag", title = "Proportion of Customers by
Segments") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))

# Plot and label with proportion of customers
p + geom_text(data = ggplot_build(p)$data[[1]], aes(x = (xmin + xmax)/2 , y = (ymin +
ymax)/2,
                                                          label =
as.character(paste(round(.wt/sum(.wt),3)*100,'%'))))
```



Observations

- The segment with the highest number of customers is “YOUNG SINGLES/COUPLES” in the “Mainstream” category with 7917 customers (11.11%).
- The segment with the next highest number of customers are “RETIREES” in the “Mainstream” category with 6358 customers (8.92%).
- “OLDER FAMILIES” in the “Budget” category with 4611 customers (6.47%) occupies only the sixth position.

- So, we can see that the first two segments with the highest sales at the same time have the highest customer count. This indicates that the higher sales at least in these segments are indeed connected to a larger customer base.
- However, this is not the case for the segment “OLDER FAMILIES” in the “Budget” category, and the number of customers is not the main driver for sales.

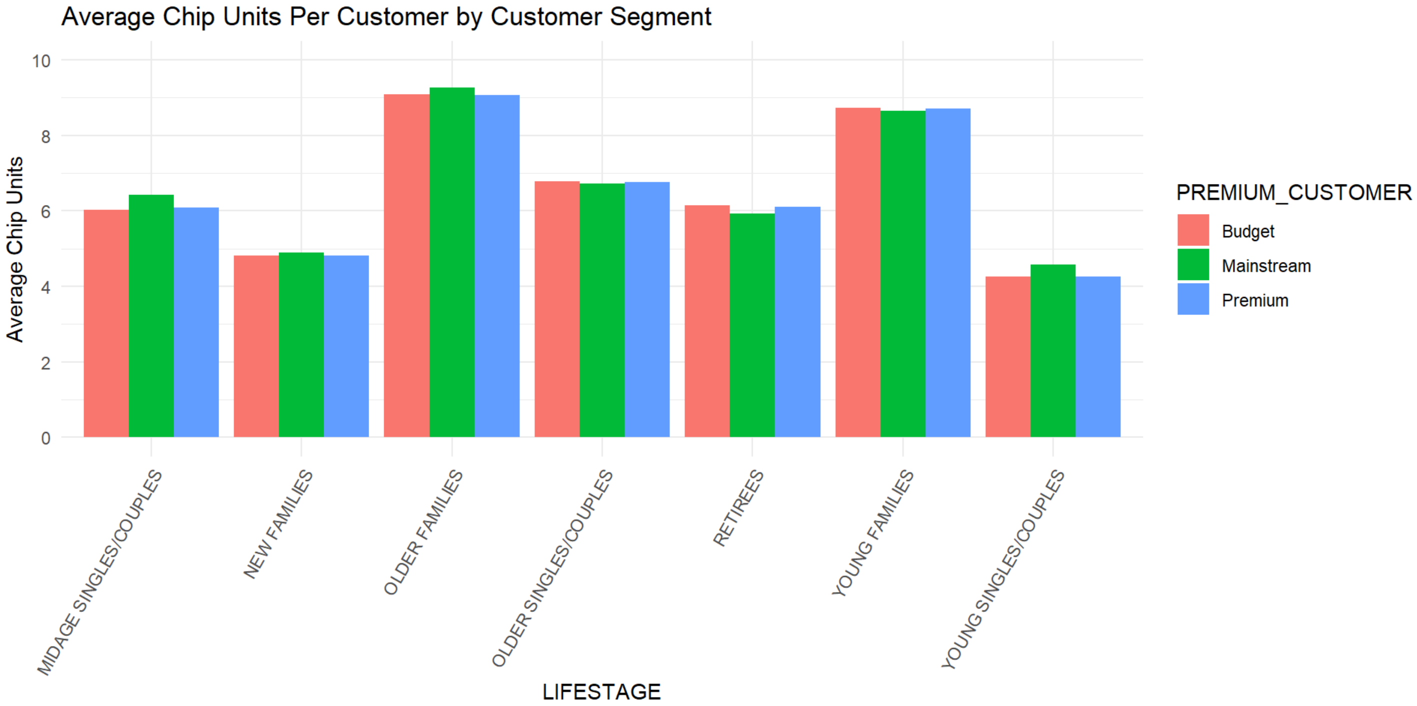
Calculate the average number of chips bought per customer by segment

- Let’s look at it from another side. Higher sales may also be driven by more units of chips being bought per customer.

```
chips_per_customer <- data[, .(AVG_num_of_units = sum(PROD_QTY)/uniqueN(LYLTY_CARD_NBR)),
                           .(LIFESTAGE, PREMIUM_CUSTOMER)][order(-AVG_num_of_units)]
chips_per_customer

##                  LIFESTAGE PREMIUM_CUSTOMER AVG_num_of_units
##  1:         OLDER FAMILIES       Mainstream         9.255380
##  2:         OLDER FAMILIES           Budget         9.076773
##  3:         OLDER FAMILIES          Premium         9.071717
##  4:         YOUNG FAMILIES           Budget         8.722995
##  5:         YOUNG FAMILIES          Premium         8.716013
##  6:         YOUNG FAMILIES       Mainstream         8.638361
##  7:  OLDER SINGLES/COUPLES           Budget         6.781398
##  8:  OLDER SINGLES/COUPLES          Premium         6.769543
##  9:  OLDER SINGLES/COUPLES       Mainstream         6.712021
## 10: MIDAGE SINGLES/COUPLES       Mainstream         6.432080
## 11:               RETIREES           Budget         6.141847
## 12:               RETIREES          Premium         6.103358
## 13: MIDAGE SINGLES/COUPLES          Premium         6.078514
## 14: MIDAGE SINGLES/COUPLES           Budget         6.026459
## 15:               RETIREES       Mainstream         5.925920
## 16:           NEW FAMILIES       Mainstream         4.891566
## 17:           NEW FAMILIES           Budget         4.821527
## 18:           NEW FAMILIES          Premium         4.815652
## 19:  YOUNG SINGLES/COUPLES       Mainstream         4.575597
## 20:  YOUNG SINGLES/COUPLES          Premium         4.264113
## 21:  YOUNG SINGLES/COUPLES           Budget         4.250069
##                  LIFESTAGE PREMIUM_CUSTOMER AVG_num_of_units
```

Plot a visual to see the distribution of the average number of chips by segments

```
# Create the bar plot
ggplot(chips_per_customer, aes(x = LIFESTAGE, y = AVG_num_of_units, fill =
PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "LIFESTAGE", y = "Average Chip Units", fill = "PREMIUM_CUSTOMER") +
  ggtitle("Average Chip Units Per Customer by Customer Segment") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  scale_y_continuous(limits = c(0, 10.0), breaks = seq(0, 10.0, 2.0))
```



Observations

- The “OLDER FAMILIES” segment in each category has the highest average number of chip units per customer with values in the range of 9.26-9.07.
- The next segment is “YOUNG FAMILIES” in each category which also has high average of chip units per customer with values ranging from 8.72 to 8.64.
- In summary, these insights provide an alternative perspective on the factors influencing higher sales. They indicate that certain segments, such as “OLDER FAMILIES” and “YOUNG FAMILIES,” contribute to higher sales including thanks to a higher average quantity of chips purchased by each customer in these segments.

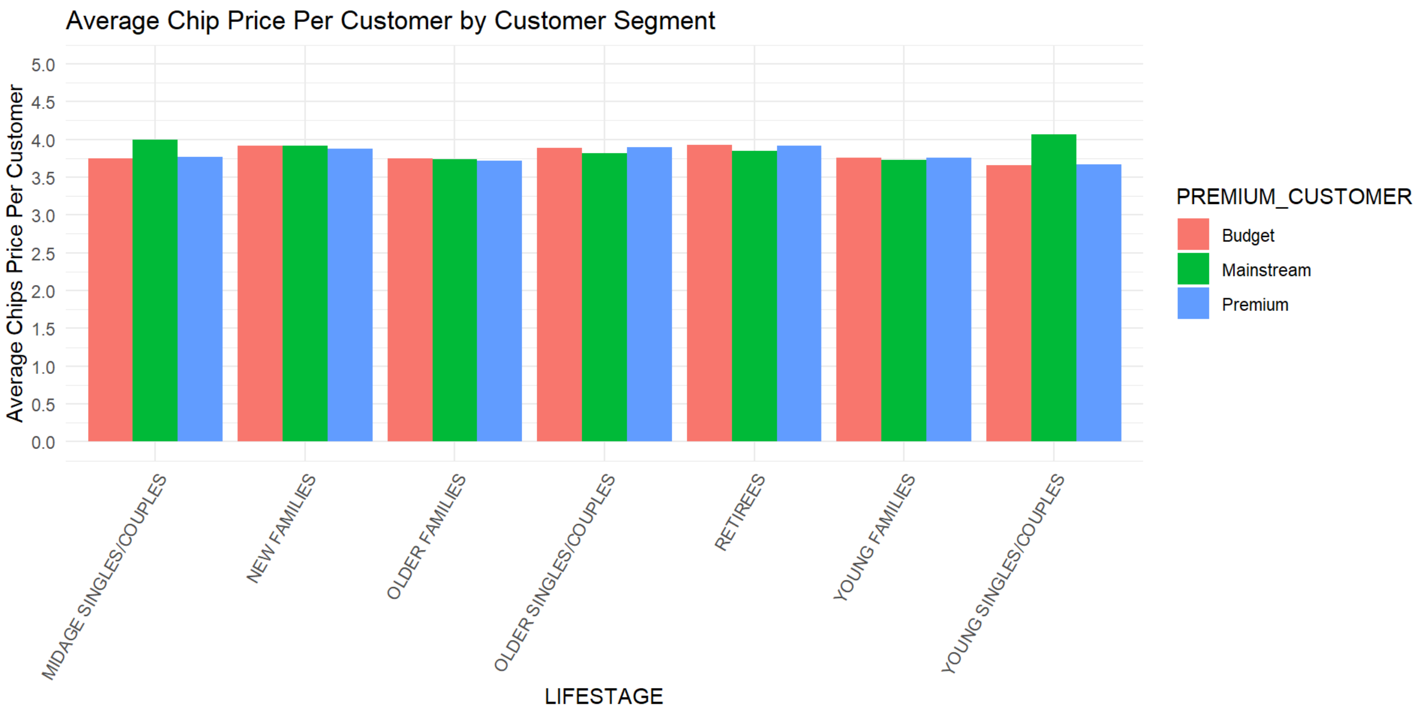
Calculate the average chip price per customer segment

- Let’s investigate the average price per unit of chips bought for each customer segment and whether it is also a driver of total sales.

```
avg_chip_price <- data[, .(Avg_Chip_Price = mean(TOT_SALES / PROD_QTY)), .(LIFESTAGE,
PREMIUM_CUSTOMER)][order(-Avg_Chip_Price)]
avg_chip_price

##                  LIFESTAGE PREMIUM_CUSTOMER Avg_Chip_Price
##  1:  YOUNG SINGLES/COUPLES       Mainstream       4.065642
##  2: MIDAGE SINGLES/COUPLES       Mainstream       3.994241
##  3:               RETIREES           Budget       3.924404
##  4:               RETIREES          Premium       3.920942
##  5:           NEW FAMILIES           Budget       3.917688
##  6:           NEW FAMILIES       Mainstream       3.916133
##  7:  OLDER SINGLES/COUPLES          Premium       3.893182
##  8:  OLDER SINGLES/COUPLES           Budget       3.882096
##  9:           NEW FAMILIES          Premium       3.872110
## 10:               RETIREES       Mainstream       3.844294
## 11:  OLDER SINGLES/COUPLES       Mainstream       3.814665
## 12: MIDAGE SINGLES/COUPLES          Premium       3.770698
## 13:         YOUNG FAMILIES          Premium       3.762150
## 14:         YOUNG FAMILIES           Budget       3.760737
## 15:          OLDER FAMILIES           Budget       3.745340
## 16: MIDAGE SINGLES/COUPLES           Budget       3.743328
## 17:          OLDER FAMILIES       Mainstream       3.737077
## 18:          YOUNG FAMILIES       Mainstream       3.724533
## 19:          OLDER FAMILIES          Premium       3.717000
## 20:  YOUNG SINGLES/COUPLES          Premium       3.665414
## 21:  YOUNG SINGLES/COUPLES           Budget       3.657366
##                   LIFESTAGE PREMIUM_CUSTOMER Avg_Chip_Price
```

Plot a visual to see the distribution of the average chip price per customer by segments

```
# Create the bar plot
ggplot(avg_chip_price, aes(x = LIFESTAGE, y = Avg_Chip_Price, fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "LIFESTAGE", y = "Average Chips Price Per Customer", fill =
"PREMIUM_CUSTOMER") +
  ggtitle("Average Chip Price Per Customer by Customer Segment") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  scale_y_continuous(limits = c(0, 5.0), breaks = seq(0, 5.0, 0.5))
```



Observations

- We can observe that "YOUNG SINGLES/COUPLES" in the "Mainstream" category (with the value of 4.066) and "MIDAGE SINGLES/COUPLES" in the "Mainstream" category (with the value of 3.994) are more willing to pay more per packet of chips compared to their "Budget" and "Premium" counterparts.
- The difference in average price per unit isn't large, and ranges from 3.657 to 4.066

Perform an independent t-test to see if the difference is significant

- As the difference in average price per unit isn't large, we can check if this difference is statistically different. Let's perform an independent t-test between "Mainstream" category vs "Premium" and "Budget" categories of "MIDAGE SINGLES/COUPLES" and "YOUNG SINGLES/COUPLES".

```
pricePerUnit <- data[, price := TOT_SALES/PROD_QTY]
t.test(data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES")
            & PREMIUM_CUSTOMER == "Mainstream", price],
       data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES")
            & PREMIUM_CUSTOMER != "Mainstream", price],
       alternative = "greater")

##
##  Welch Two Sample t-test
##
## data:  data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") &
PREMIUM_CUSTOMER == "Mainstream", price] and data[LIFESTAGE %in% c("YOUNG
SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & PREMIUM_CUSTOMER != "Mainstream", price]
## t = 37.624, df = 54791, p-value < 2.2e-16
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.3187234       Inf
## sample estimates:
## mean of x mean of y
##  4.039786  3.706491
```

Observations

- The t-test results in a p-value < 2.2e-16, i.e. the mean price per unit for "Mainstream" customers in the "YOUNG SINGLES/COUPLES" and "MIDAGE SINGLES/COUPLES" is significantly higher than than the same value for "Budget" or "Premium" customers in the "YOUNG SINGLES/COUPLES" and "MIDAGE SINGLES/COUPLES".

Deep dive into specific customer segments for insights

```
# Refresh the picture of our work data
head(data)

##    LYLTY_CARD_NBR       DATE STORE_NBR TXN_ID PROD_NBR
## 1:           1000 2018-10-17         1      1        5
## 2:           1002 2018-09-16         1      2       58
## 3:           1003 2019-03-07         1      3       52
## 4:           1003 2019-03-08         1      4      106
## 5:           1004 2018-11-02         1      5       96
## 6:           1005 2018-12-28         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:  Grain Waves Sour    Cream&Chives 210G        1       3.6       210
## 4: Natural ChipCo      Hony Soy Chckn175g        1       3.0       175
## 5:         WW Original Stacked Chips 160g        1       1.9       160
## 6:                   Cheetos Puffs 165g        1       2.8       165
##         BRAND             LIFESTAGE PREMIUM_CUSTOMER price
## 1:    NATURAL  YOUNG SINGLES/COUPLES          Premium   3.0
## 2:        RRD  YOUNG SINGLES/COUPLES       Mainstream   2.7
## 3:    GRNWVES         YOUNG FAMILIES           Budget   3.6
## 4:    NATURAL         YOUNG FAMILIES           Budget   3.0
## 5: WOOLWORTHS  OLDER SINGLES/COUPLES       Mainstream   1.9
## 6:    CHEETOS MIDAGE SINGLES/COUPLES       Mainstream   2.8
```

Observations

Based on the analysis conducted, the following customer segments can be considered as potential targets for deeper exploration:

1. “OLDER FAMILIES” in the “Budget” category:
   - This segment has the highest total sales and ranks sixth in terms of the customer count.
   - Also, it has a second-highest average number of chips bought per customer.
2. “YOUNG SINGLES/COUPLES” in the “Mainstream” category:
   - This segment ranks second in terms of total sales.
   - It has the highest customer count.
   - It exhibits the highest average price per unit of chips, indicating a willingness to pay more.
3. “RETIREES” in the “Mainstream” category:
   - This segment demonstrates a significant volume of total sales and ranked third.
   - It has a second-highest customer count.
   - It has a comparable average price per unit of chips.

By focusing on these target customer segments, we can gain valuable insights into their specific needs, preferences, and behaviors. This information can help tailor marketing strategies, product offerings, and promotions to better engage and satisfy these target segments.

Deep dive into “YOUNG SINGLES/COUPLES” in the “Mainstream” category

- Let’s find out if they tend to buy a particular brand of chips

```
segment1 <- data[LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER ==
"Mainstream",]
other <- data[!(LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER ==
"Mainstream"),]

# Brand affinity compared to the rest of the population
quantity_segment1 <- segment1[, sum(PROD_QTY)]
quantity_other <- other[, sum(PROD_QTY)]
quantity_segment1_by_brand <- segment1[, .(targetSegment =
sum(PROD_QTY)/quantity_segment1),
                                       by = BRAND]
quantity_other_by_brand <- other[, .(other = sum(PROD_QTY)/quantity_other),
                                 by = BRAND]
brand_proportions <- merge(quantity_segment1_by_brand,
                           quantity_other_by_brand)[, affinityToBrand :=
targetSegment/other]
brand_proportions[order(-affinityToBrand)]

##          BRAND targetSegment       other affinityToBrand
##  1:   TYRRELLS   0.031552795 0.025692464       1.2280953
##  2:   TWISTIES   0.046183575 0.037876520       1.2193194
##  3:    DORITOS   0.122760524 0.101074684       1.2145526
##  4:     KETTLE   0.197984817 0.165553442       1.1958967
##  5:   TOSTITOS   0.045410628 0.037977861       1.1957131
##  6:   PRINGLES   0.119420290 0.100634769       1.1866703
##  7:       COBS   0.044637681 0.039048861       1.1431238
##  8:  INFUZIONS   0.064679089 0.057064679       1.1334347
##  9:      THINS   0.060372671 0.056986370       1.0594230
## 10:    GRNWVES   0.032712215 0.031187957       1.0488733
## 11:   CHEEZELS   0.017971014 0.018646902       0.9637534
## 12:     SMITHS   0.096369910 0.124583692       0.7735355
## 13:     FRENCH   0.003947550 0.005758060       0.6855694
## 14:    CHEETOS   0.008033126 0.012066591       0.6657329
## 15:        RRD   0.043809524 0.067493678       0.6490908
## 16:    NATURAL   0.019599724 0.030853989       0.6352412
## 17:        CCS   0.011180124 0.018895650       0.5916771
## 18:   SUNBITES   0.006349206 0.012580210       0.5046980
## 19: WOOLWORTHS   0.024099379 0.049427188       0.4875733
## 20:     BURGER   0.002926156 0.006596434       0.4435967
```

Observations

- The `affinityToBrand` column represents the brand affinity ratio, indicating how much more likely the “YOUNG SINGLES/COUPLES” in the “Mainstream” category are to purchase a specific brand compared to the rest of the population. A value greater than 1 suggests a higher affinity for the brand, while a value less than 1 suggests a lower affinity.
- Such brands as Tyrrells, Twisties, Doritos, Kettle, and Tostitos show the highest brand affinity values.
- Customers in this segment are 23% more likely to purchase Tyrrells chips compared to the rest of the population.
- At the same time, it is important to take into account that the share of sold units of Tyrrells chips is only 3.16% of the total quantity sold in the segment while the same value for the Kettle brand is 19.8%
- Let’s also find out if our target segment tends to buy larger packs of chips

```
# Preferred pack size compared to the rest of the population
quantity_segment1_by_pack <- segment1[, .(targetSegment =
sum(PROD_QTY)/quantity_segment1),
                                      by = PACK_SIZE]
quantity_other_by_pack <- other[, .(other = sum(PROD_QTY)/quantity_other),
                                by = PACK_SIZE]
pack_proportions <- merge(quantity_segment1_by_pack,
                          quantity_other_by_pack)[, affinityToPack :=
targetSegment/other]
pack_proportions[order(-affinityToPack)]

##     PACK_SIZE targetSegment       other affinityToPack
##  1:       270   0.031828847 0.025095929      1.2682873
##  2:       380   0.032160110 0.025584213      1.2570295
##  3:       330   0.061283644 0.050161917      1.2217166
##  4:       134   0.119420290 0.100634769      1.1866703
##  5:       110   0.106280193 0.089791190      1.1836372
##  6:       210   0.029123533 0.025121265      1.1593180
##  7:       135   0.014768806 0.013075403      1.1295106
##  8:       250   0.014354727 0.012780590      1.1231662
##  9:       170   0.080772947 0.080985964      0.9973697
## 10:       150   0.157598344 0.163420656      0.9643722
## 11:       175   0.254989648 0.270006956      0.9443818
## 12:       165   0.055652174 0.062267662      0.8937572
## 13:       190   0.007481021 0.012442016      0.6012708
## 14:       180   0.003588682 0.006066692      0.5915385
## 15:       160   0.006404417 0.012372920      0.5176157
## 16:        90   0.006349206 0.012580210      0.5046980
## 17:       125   0.003008972 0.006036750      0.4984423
## 18:       200   0.008971705 0.018656115      0.4808989
## 19:        70   0.003036577 0.006322350      0.4802924
## 20:       220   0.002926156 0.006596434      0.4435967
```

Observations

- The `affinityToPack` column represents the pack size affinity ratio, indicating how much more likely the “YOUNG SINGLES/COUPLES” in the “Mainstream” category are to purchase a specific pack size compared to the rest of the population. A value greater than 1 suggests a higher affinity for the pack size, while a value less than 1 suggests a lower affinity.
- The pack sizes with the highest affinity among the “YOUNG SINGLES/COUPLES” in the “Mainstream” category are 270g, 380g, and 330g. These pack sizes have a higher proportion of purchase within the segment compared to the rest of the population.
- Customers in this segment are 27% more likely to purchase a 270g pack of chips compared to the rest of the population.
- At the same time, it is important to take into account that the share of sold units of 270g packs of chips is only 3.18% of the total quantity sold in the segment while the same value for the 175g packs is 25.5%

```
data[PACK_SIZE == 270, unique(PROD_NAME)]

## [1] "Twisties Cheese     270g" "Twisties Chicken270g"

data[PACK_SIZE == 175, unique(PROD_NAME)]

##  [1] "Natural Chip        Compny SeaSalt175g"
##  [2] "Natural ChipCo      Hony Soy Chckn175g"
##  [3] "CCs Tasty Cheese    175g"
##  [4] "Tostitos Splash Of  Lime 175g"
##  [5] "Smiths Chip Thinly  S/Cream&Onion 175g"
##  [6] "Natural ChipCo Sea  Salt & Vinegr 175g"
##  [7] "Natural Chip Co     Tmato Hrb&Spce 175g"
##  [8] "Smiths Thinly       Swt Chli&S/Cream175G"
##  [9] "Thins Chips Seasonedchicken 175g"
## [10] "Smiths Thinly Cut   Roast Chicken 175g"
## [11] "Smiths Chip Thinly  Cut Original 175g"
## [12] "Thins Chips Light&  Tangy 175g"
## [13] "Kettle Chilli 175g"
## [14] "WW Crinkle Cut      Chicken 175g"
## [15] "Smiths Chip Thinly  CutSalt/Vinegr175g"
## [16] "Tostitos Smoked     Chipotle 175g"
## [17] "CCs Nacho Cheese    175g"
## [18] "French Fries Potato Chips 175g"
## [19] "Kettle Mozzarella   Basil & Pesto 175g"
## [20] "CCs Original 175g"
## [21] "Tostitos Lightly    Salted 175g"
## [22] "Kettle Original 175g"
## [23] "Kettle Sweet Chilli And Sour Cream 175g"
## [24] "Kettle Sea Salt     And Vinegar 175g"
## [25] "WW Crinkle Cut      Original 175g"
## [26] "Thins Chips Salt &  Vinegar 175g"
## [27] "Thins Chips         Originl saltd 175g"
## [28] "Kettle Honey Soy    Chicken 175g"
## [29] "NCC Sour Cream &    Garden Chives 175g"
## [30] "Thins Potato Chips  Hot & Spicy 175g"
```

Observations

- Twisties are the only brand offering 270g packs and so, this may instead be reflecting a higher likelihood of purchasing Twisties.
- At the same time, it is obvious that the 175g pack size is extremely popular among customers and many brands use it for their products.

Create certain generic functions to optimize the further process

Define the top 5 values based on the quantity of the purchased product

```
top_values_qty <- function(data, column_name) {
  top_values <- data %>%
    group_by({{ column_name }}) %>%
    summarise(Sold_Units = sum(PROD_QTY)) %>%
    arrange(desc(Sold_Units)) %>%
    top_n(5, Sold_Units)

  return(top_values)
}
```

Define the top 5 values based on the total sales

```
top_values_sales <- function(data, column_name) {
  top_values <- data %>%
    group_by({{ column_name }}) %>%
    summarise(Total_Sales = sum(TOT_SALES)) %>%
    arrange(desc(Total_Sales)) %>%
    top_n(5, Total_Sales)

  return(top_values)
}
```

Examine "OLDER FAMILIES" in the "Budget" category

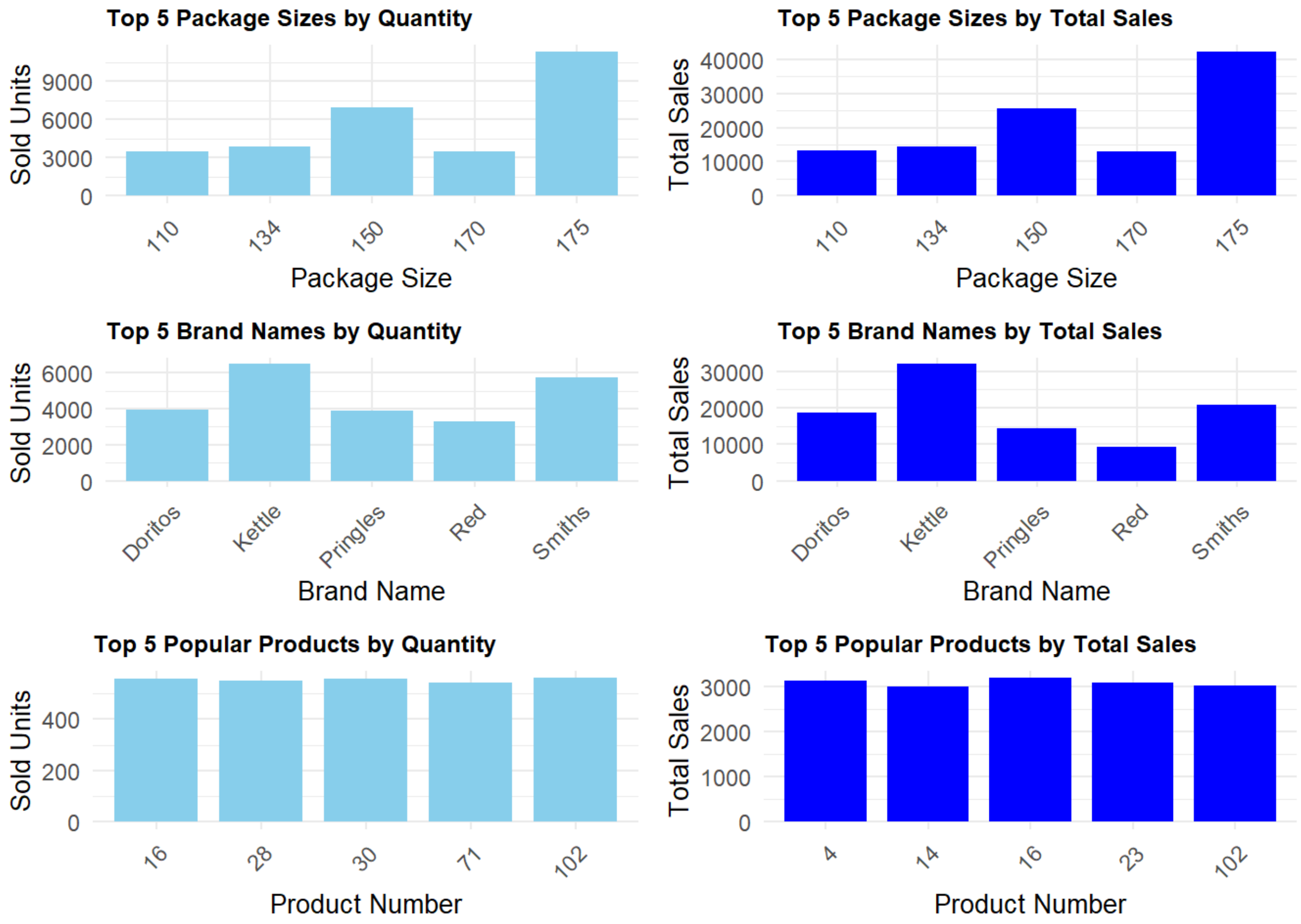
```
# Subset the data for the current target customer segment
target_OldFmlBdg <- subset(data, LIFESTAGE == "OLDER FAMILIES" & PREMIUM_CUSTOMER ==
"Budget")
head(target_OldFmlBdg)

##    LYLTY_CARD_NBR       DATE STORE_NBR TXN_ID PROD_NBR
## 1:           1022 2018-10-24         1     29        3
## 2:           1090 2019-04-27         1    103        6
## 3:           1102 2018-07-20         1    117       54
## 4:           1103 2019-03-06         1    118       73
## 5:           1136 2019-02-02         1    157       66
## 6:           1190 2019-05-22         1    225      114
##                                    PROD_NAME PROD_QTY TOT_SALES PACK_SIZE  BRAND
## 1: Kettle Sensations   Camembert & Fig 150g        1       4.6       150 KETTLE
## 2:                 RRD Lime & Pepper   165g        1       3.0       165    RRD
## 3:                        CCs Original 175g        1       2.1       175    CCS
## 4:  Smiths Crinkle Cut  Salt & Vinegar 170g        1       2.9       170 SMITHS
## 5:                 CCs Nacho Cheese    175g        1       2.1       175    CCS
## 6:    Kettle Sensations   Siracha Lime 150g        1       4.6       150 KETTLE
##          LIFESTAGE PREMIUM_CUSTOMER price
## 1: OLDER FAMILIES           Budget   4.6
## 2: OLDER FAMILIES           Budget   3.0
## 3: OLDER FAMILIES           Budget   2.1
## 4: OLDER FAMILIES           Budget   2.9
## 5: OLDER FAMILIES           Budget   2.1
## 6: OLDER FAMILIES           Budget   4.6
```

```
# Get top 5 pack sizes based on quantity
pack_size_qty <- top_values_qty(target_OldFmlBdg, PACK_SIZE)

# Get top 5 pack sizes based on sales
pack_size_sales <- top_values_sales(target_OldFmlBdg, PACK_SIZE)

# Get top 5 brand names based on quantity
brand_qty <- top_values_qty(target_OldFmlBdg, BRAND)

# Get top 5 brand names based on sales
brand_sales <- top_values_sales(target_OldFmlBdg, BRAND)

# Get top 5 popular products based on quantity
product_qty <- top_values_qty(target_OldFmlBdg, PROD_NBR)

# Get top 5 popular products based on sales
product_sales <- top_values_sales(target_OldFmlBdg, PROD_NBR)
```

Visuals for the top 5 values

```
# Define the custom theme
custom_theme <- theme_minimal() +
  theme(
    plot.title = element_text(size = 9, face = "bold"),
    axis.text.x = element_text(angle = 45, hjust = 1),
    axis.title = element_text(size = 10)
  )

# Set the custom theme as the default theme for all plots
theme_set(custom_theme)

# Create the bar plot for `PACK_SIZE` based on the quantity
bar_plot_qty1 <- ggplot(pack_size_qty, aes(x = as.factor(PACK_SIZE), y = Sold_Units)) +
  geom_bar(stat = "identity", fill = "skyblue", width = 0.8) +
  labs(x = "Package Size", y = "Sold Units",
       title = "Top 5 Package Sizes by Quantity")

# Create the bar plot for `PACK_SIZE` based on total sales
bar_plot_sales1 <- ggplot(pack_size_sales, aes(x = as.factor(PACK_SIZE), y =
Total_Sales)) +
  geom_bar(stat = "identity", fill = "blue", width = 0.8) +
  labs(x = "Package Size", y = "Total Sales",
       title = "Top 5 Package Sizes by Total Sales")

# Create the bar plot for `BRAND` based on the quantity
bar_plot_qty2 <- ggplot(brand_qty, aes(x = as.factor(BRAND), y = Sold_Units)) +
  geom_bar(stat = "identity", fill = "skyblue", width = 0.8) +
  labs(x = "Brand Name", y = "Sold Units",
       title = "Top 5 Brand Names by Quantity")
```

```
# Create the bar plot for `BRAND` based on total sales
bar_plot_sales2 <- ggplot(brand_sales, aes(x = as.factor(BRAND), y = Total_Sales)) +
  geom_bar(stat = "identity", fill = "blue", width = 0.8) +
  labs(x = "Brand Name", y = "Total Sales",
       title = "Top 5 Brand Names by Total Sales")

# Create the bar plot for `PROD_NBR` based on the quantity
bar_plot_qty3 <- ggplot(product_qty, aes(x = as.factor(PROD_NBR), y = Sold_Units)) +
  geom_bar(stat = "identity", fill = "skyblue", width = 0.8) +
  labs(x = "Product Number", y = "Sold Units",
       title = "Top 5 Popular Products by Quantity")

# Create the bar plot for `PROD_NBR` based on total sales
bar_plot_sales3 <- ggplot(product_sales, aes(x = as.factor(PROD_NBR), y = Total_Sales)) +
  geom_bar(stat = "identity", fill = "blue", width = 0.8) +
  labs(x = "Product Number", y = "Total Sales",
       title = "Top 5 Popular Products by Total Sales")

plot_grid = ggarrange(bar_plot_qty1, bar_plot_sales1,
                      bar_plot_qty2, bar_plot_sales2,
                      bar_plot_qty3, bar_plot_sales3,
                      ncol = 2, nrow = 3)
plot_grid
```



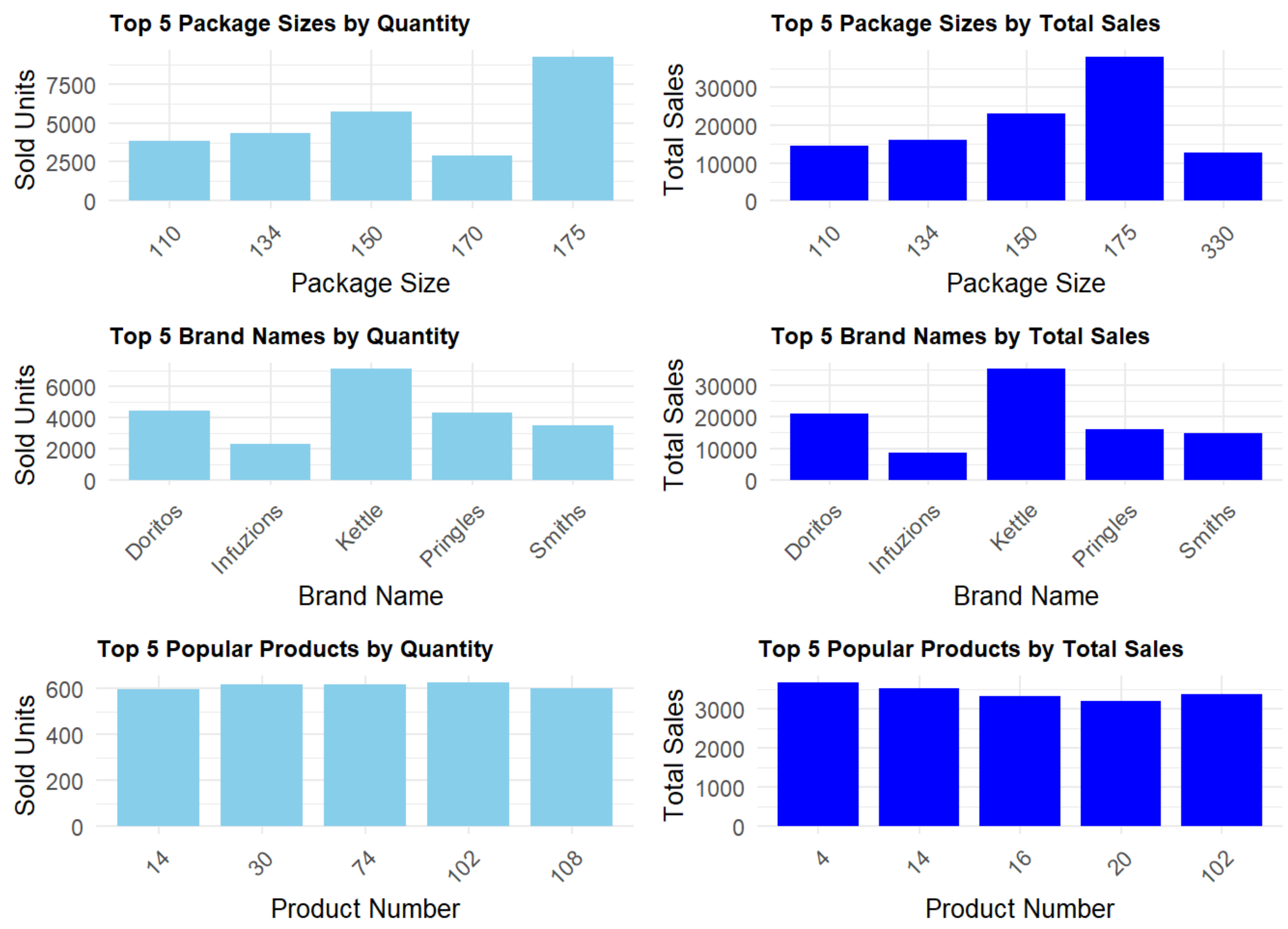
Examine “YOUNG SINGLES/COUPLES” in the “Mainstream” category

```
# Subset the data for the current target customer segment
target_YngSCMns <- subset(data, LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER
== "Mainstream")
head(target_YngSCMns)

##    LYLTY_CARD_NBR       DATE STORE_NBR TXN_ID PROD_NBR
## 1:           1002 2018-09-16         1      2       58
## 2:           1010 2018-09-09         1     10       51
## 3:           1018 2018-09-03         1     22        3
## 4:           1018 2018-11-28         1     23       97
## 5:           1018 2019-06-20         1     24       38
## 6:           1020 2018-08-16         1     26       19
##                                    PROD_NAME PROD_QTY TOT_SALES PACK_SIZE
## 1:    Red Rock Deli Chikn&Garlic Aioli 150g        1       2.7       150
## 2:                 Doritos Mexicana    170g        2       8.8       170
## 3: Kettle Sensations   Camembert & Fig 150g        1       4.6       150
## 4:                 RRD Salt & Vinegar  165g        1       3.0       165
## 5:  Infuzions Mango     Chutny Papadums 70g        1       2.4        70
## 6:      Smiths Crinkle Cut  Snag&Sauce 150g        1       2.6       150
##        BRAND             LIFESTAGE PREMIUM_CUSTOMER price
## 1:       RRD YOUNG SINGLES/COUPLES       Mainstream   2.7
## 2:   DORITOS YOUNG SINGLES/COUPLES       Mainstream   4.4
## 3:    KETTLE YOUNG SINGLES/COUPLES       Mainstream   4.6
## 4:       RRD YOUNG SINGLES/COUPLES       Mainstream   3.0
## 5: INFUZIONS YOUNG SINGLES/COUPLES       Mainstream   2.4
## 6:    SMITHS YOUNG SINGLES/COUPLES       Mainstream   2.6

# Get top 5 pack sizes based on quantity
pack_size_qty <- top_values_qty(target_YngSCMns, PACK_SIZE)

# Get top 5 pack sizes based on sales
pack_size_sales <- top_values_sales(target_YngSCMns, PACK_SIZE)

# Get top 5 brand names based on quantity
brand_qty <- top_values_qty(target_YngSCMns, BRAND)

# Get top 5 brand names based on sales
brand_sales <- top_values_sales(target_YngSCMns, BRAND)

# Get top 5 popular products based on quantity
product_qty <- top_values_qty(target_YngSCMns, PROD_NBR)

# Get top 5 popular products based on sales
product_sales <- top_values_sales(target_YngSCMns, PROD_NBR)
```

Visuals for the top 5 values

```
# Create the bar plot for `PACK_SIZE` based on the quantity
bar_plot_qty1 <- ggplot(pack_size_qty, aes(x = as.factor(PACK_SIZE), y = Sold_Units)) +
  geom_bar(stat = "identity", fill = "skyblue", width = 0.8) +
  labs(x = "Package Size", y = "Sold Units",
       title = "Top 5 Package Sizes by Quantity")

# Create the bar plot for `PACK_SIZE` based on total sales
bar_plot_sales1 <- ggplot(pack_size_sales, aes(x = as.factor(PACK_SIZE), y =
Total_Sales)) +
```

```r
  geom_bar(stat = "identity", fill = "blue", width = 0.8) +
  labs(x = "Package Size", y = "Total Sales",
       title = "Top 5 Package Sizes by Total Sales")

# Create the bar plot for `BRAND` based on the quantity
bar_plot_qty2 <- ggplot(brand_qty, aes(x = as.factor(BRAND), y = Sold_Units)) +
  geom_bar(stat = "identity", fill = "skyblue", width = 0.8) +
  labs(x = "Brand Name", y = "Sold Units",
       title = "Top 5 Brand Names by Quantity")

# Create the bar plot for `BRAND` based on total sales
bar_plot_sales2 <- ggplot(brand_sales, aes(x = as.factor(BRAND), y = Total_Sales)) +
  geom_bar(stat = "identity", fill = "blue", width = 0.8) +
  labs(x = "Brand Name", y = "Total Sales",
       title = "Top 5 Brand Names by Total Sales")

# Create the bar plot for `PROD_NBR` based on the quantity
bar_plot_qty3 <- ggplot(product_qty, aes(x = as.factor(PROD_NBR), y = Sold_Units)) +
  geom_bar(stat = "identity", fill = "skyblue", width = 0.8) +
  labs(x = "Product Number", y = "Sold Units",
       title = "Top 5 Popular Products by Quantity")

# Create the bar plot for `PROD_NBR` based on total sales
bar_plot_sales3 <- ggplot(product_sales, aes(x = as.factor(PROD_NBR), y = Total_Sales)) +
  geom_bar(stat = "identity", fill = "blue", width = 0.8) +
  labs(x = "Product Number", y = "Total Sales",
       title = "Top 5 Popular Products by Total Sales")

plot_grid = ggarrange(bar_plot_qty1, bar_plot_sales1,
                      bar_plot_qty2, bar_plot_sales2,
                      bar_plot_qty3, bar_plot_sales3,
                      ncol = 2, nrow = 3)
plot_grid
```



Examine “RETIREES” in the “Mainstream” category

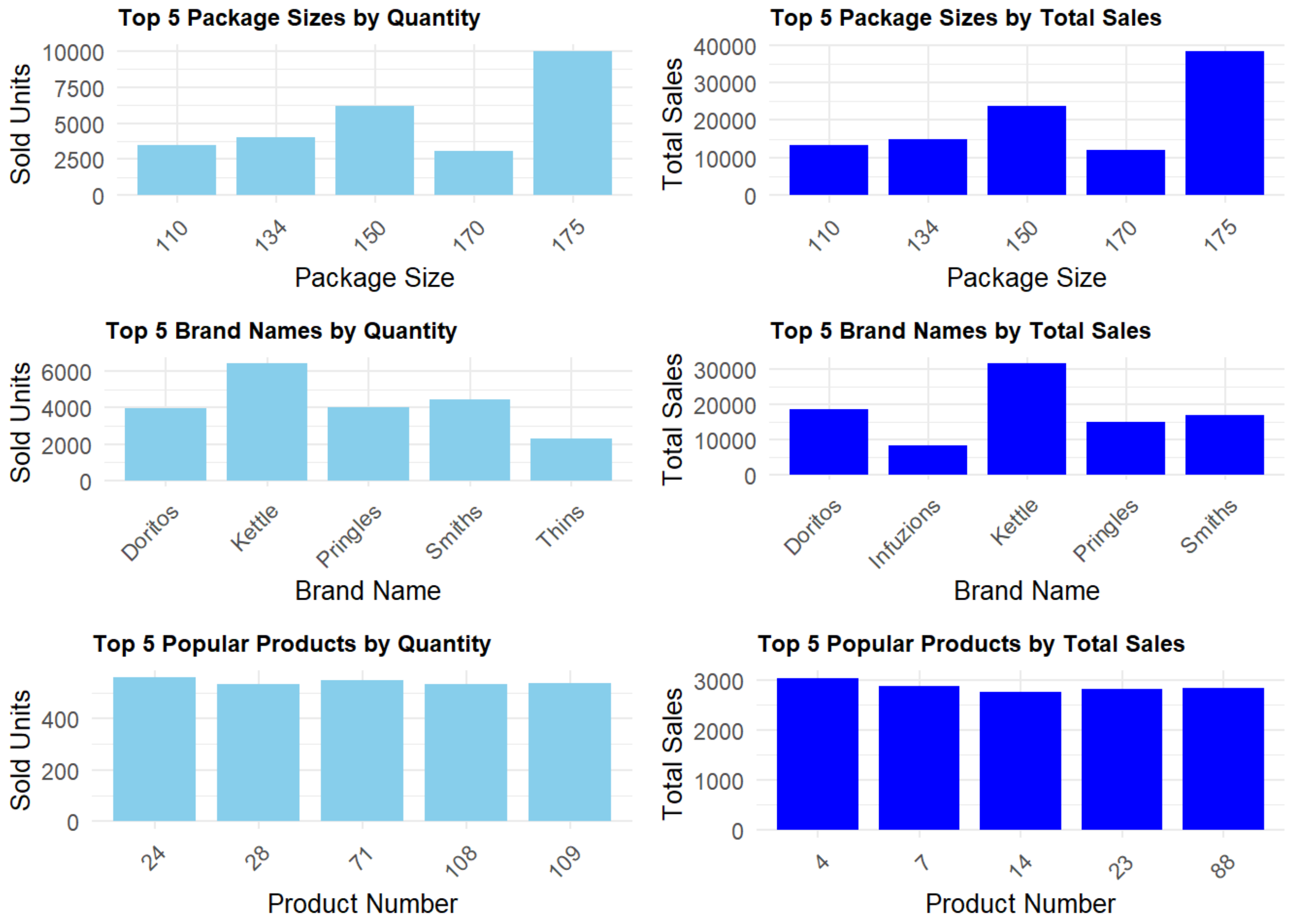
```
# Subset the data for the current target customer segment
target_RtrMns <- subset(data, LIFESTAGE == "RETIREES" & PREMIUM_CUSTOMER == "Mainstream")
head(target_RtrMns)

##    LYLTY_CARD_NBR       DATE STORE_NBR TXN_ID PROD_NBR
## 1:           1030 2019-06-23         1     37       10
## 2:           1055 2018-07-08         1     62       43
## 3:           1089 2019-03-23         1    102       97
## 4:           1095 2019-01-10         1    109       79
## 5:           1101 2018-07-17         1    116        6
## 6:           1104 2018-09-11         1    119       50
##                                  PROD_NAME PROD_QTY TOT_SALES PACK_SIZE    BRAND
## 1:    RRD SR Slow Rst     Pork Belly 150g        2       5.4       150      RRD
## 2:     Smith Crinkle Cut   Bolognese 150g        1       2.6       150   SMITHS
## 3:               RRD Salt & Vinegar  165g        1       3.0       165      RRD
## 4: Smiths Chip Thinly  CutSalt/Vinegr175g        1       3.0       175   SMITHS
## 5:               RRD Lime & Pepper   165g        1       3.0       165      RRD
## 6:        Tostitos Lightly    Salted 175g        2       8.8       175 TOSTITOS
##    LIFESTAGE PREMIUM_CUSTOMER price
## 1:  RETIREES       Mainstream   2.7
## 2:  RETIREES       Mainstream   2.6
## 3:  RETIREES       Mainstream   3.0
## 4:  RETIREES       Mainstream   3.0
## 5:  RETIREES       Mainstream   3.0
## 6:  RETIREES       Mainstream   4.4
```

```r
# Get top 5 pack sizes based on quantity
pack_size_qty <- top_values_qty(target_RtrMns, PACK_SIZE)

# Get top 5 pack sizes based on sales
pack_size_sales <- top_values_sales(target_RtrMns, PACK_SIZE)

# Get top 5 brand names based on quantity
brand_qty <- top_values_qty(target_RtrMns, BRAND)

# Get top 5 brand names based on sales
brand_sales <- top_values_sales(target_RtrMns, BRAND)

# Get top 5 popular products based on quantity
product_qty <- top_values_qty(target_RtrMns, PROD_NBR)

# Get top 5 popular products based on sales
product_sales <- top_values_sales(target_RtrMns, PROD_NBR)
```

Visuals for the top 5 values

```r
# Create the bar plot for `PACK_SIZE` based on the quantity
bar_plot_qty1 <- ggplot(pack_size_qty, aes(x = as.factor(PACK_SIZE), y = Sold_Units)) +
  geom_bar(stat = "identity", fill = "skyblue", width = 0.8) +
  labs(x = "Package Size", y = "Sold Units",
       title = "Top 5 Package Sizes by Quantity")

# Create the bar plot for `PACK_SIZE` based on total sales
bar_plot_sales1 <- ggplot(pack_size_sales, aes(x = as.factor(PACK_SIZE), y =
Total_Sales)) +
  geom_bar(stat = "identity", fill = "blue", width = 0.8) +
  labs(x = "Package Size", y = "Total Sales",
       title = "Top 5 Package Sizes by Total Sales")

# Create the bar plot for `BRAND` based on the quantity
bar_plot_qty2 <- ggplot(brand_qty, aes(x = as.factor(BRAND), y = Sold_Units)) +
  geom_bar(stat = "identity", fill = "skyblue", width = 0.8) +
  labs(x = "Brand Name", y = "Sold Units",
       title = "Top 5 Brand Names by Quantity")

# Create the bar plot for `BRAND` based on total sales
bar_plot_sales2 <- ggplot(brand_sales, aes(x = as.factor(BRAND), y = Total_Sales)) +
  geom_bar(stat = "identity", fill = "blue", width = 0.8) +
  labs(x = "Brand Name", y = "Total Sales",
       title = "Top 5 Brand Names by Total Sales")

# Create the bar plot for `PROD_NBR` based on the quantity
bar_plot_qty3 <- ggplot(product_qty, aes(x = as.factor(PROD_NBR), y = Sold_Units)) +
  geom_bar(stat = "identity", fill = "skyblue", width = 0.8) +
  labs(x = "Product Number", y = "Sold Units",
       title = "Top 5 Popular Products by Quantity")

# Create the bar plot for `PROD_NBR` based on total sales
bar_plot_sales3 <- ggplot(product_sales, aes(x = as.factor(PROD_NBR), y = Total_Sales)) +
  geom_bar(stat = "identity", fill = "blue", width = 0.8) +
  labs(x = "Product Number", y = "Total Sales",
       title = "Top 5 Popular Products by Total Sales")
```

```
plot_grid = ggarrange(bar_plot_qty1, bar_plot_sales1,
                      bar_plot_qty2, bar_plot_sales2,
                      bar_plot_qty3, bar_plot_sales3,
                      ncol = 2, nrow = 3)
plot_grid
```



Observations

From the above analysis, we can highlight the following characteristics, purchasing trends, and behaviors of customers in the most significant customer segments in terms of total sales.

1. “OLDER FAMILIES” in the “Budget” category:
- total sales: the highest (about 157K or 8.69% of all sales)
- customer count: the sixth place (just over 4.6K)
- the average number of chips bought per customer: the second-highest (9.077)
- the average price per unit of chips: relatively low (3.745 in the general range of 4.066-3.657)
- preferred pack sizes (g) in descending order: 175, 150, 134, 110, 170
- preferred Brand names in descending order: Kettle, Smiths, Doritos, Pringles, RRD
- the most popular products in descending order: 102, 16, 30, 28, 71 (by quantity sold) and 16, 4, 23, 102, 14 (on total sales)
2. “YOUNG SINGLES/COUPLES” in the “Mainstream” category:
- total sales: the second-highest (about 148K or 8.18% of all sales)
- customer count: the highest (just over 7.9K)

- the average number of chips bought per customer: one of the lowest (4.576)
- the average price per unit of chips: the highest (4.066 in the general range of 4.066-3.657)
- preferred pack sizes (g) in descending order: 175, 150, 134, 110, 170 (by quantity sold) and 330 (on total sales)
- preferred Brand names in descending order: Kettle, Doritos, Pringles, Smiths, Infuzions

- the most popular products in descending order: 102, 30, 74, 108, 14 (by quantity sold) and 4, 14, 102, 16, 20 (on total sales)

3. "RETIREES" in the "Mainstream" category:

- total sales: the third-highest (just over 145K or 8.04% of all sales)
- customer count: the second-highest (about 6.4K)
- the average number of chips bought per customer: relatively low (5.926)
- the average price per unit of chips: slightly above the middle value (3.845 in the general range of 4.066-3.657)
- preferred pack sizes (g) in descending order: 175, 150, 134, 110, 170
- preferred Brand names in descending order: Kettle (by quantity sold and on total sales), Doritos (on total sales), Smiths (on total sales), Pringles (on total sales), Infuzions (on total sales)
- the most popular products in descending order: 24, 71, 109, 108, 28 (by quantity sold) and 4, 7, 88, 23, 14 (on total sales)

4. Among the top values, especially noticeable are the products of the "Kettle" brand and the package size of 175g as they demonstrate a significant separation from other values. As for the top 5 product names, they all show fairly similar values both in terms of quantity sold and in terms of total sales. However, the following are the most common: 102, 14, 16, and 4.

Conclusions

- Sales have mainly been due to "OLDER FAMILIES" in the "Budget" category, "YOUNG SINGLES/COUPLES", and "RETIREES" both in the "Mainstream" category.

- The high spend on chips for "YOUNG SINGLES/COUPLES", and "RETIREES" both in the "Mainstream" category is due to there being more of them than other customers.

- "YOUNG SINGLES/COUPLES", and "MIDAGE SINGLES/COUPLES" both in the "Mainstream" category are also more likely to pay more per packet of chips. This is indicative of impulse buying behavior.

- "YOUNG SINGLES/COUPLES" in the "Mainstream" category are 23% more likely to purchase Tyrrells chips compared to the rest of the population. The Category Manager may want to increase the category's performance by off-locating some Tyrrells and smaller packs of chips in discretionary space near segments where young singles and couples frequent more often to increase visibility and impulse behavior.

- Quantium can help the Category Manager with recommendations of where these segments are and further help them with measuring the impact of the changed placement.