Quantium Virtual Internship - Retail Strategy and Analytics - Task 1

Solution template for Task 1

This file is a solution template for the Task 1 of the Quantium Virtual Internship. It will walk you through the analysis, providing the scaffolding for your solution with gaps left for you to fill in yourself.

Look for comments that say "over to you" for places where you need to add your own code! Often, there will be hints about what to do or what function to use in the text leading up to a code block - if you need a bit of extra help on how to use a function, the internet has many excellent resources on R coding, which you can find using your favourite search engine. ## Load required libraries and datasets Note that you will need to install these libraries if you have never used these before.

```
#### Example code to install packages
install.packages("data.table")
#### Load required libraries
library(data.table)
library(ggplot2)
library(ggmosaic)
library(readr)
#### Point the filePath to where you have downloaded the datasets to and
#### assign the data files to data.tables
# machine you will need to use forward slashes (/) instead of backshashes (\)
filePath <- "~/Downloads/"
transactionData <- fread(paste0(filePath,"QVI_transaction_data.csv"))
customerData <- fread(paste0(filePath,"QVI_purchase_behaviour.csv"))
```

Exploratory data analysis

The first step in any analysis is to first understand the data. Let's take a look at each of the datasets provided. ### Examining transaction data We can use str() to look at the format of each column and see a sample of the data. As we have read in the dataset as a data.table object, we can also run transactionData in the console to see a sample of the data or use head(transactionData) to look at the first 10 rows. Let's check if columns we would expect to be numeric are in numeric form and date columns are in date format.

```
#### Examine transaction data
# Look at the structure of the 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>```

```r
# See the first few rows of the data
head(transactionData)

## DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR ## <int> <int> <int> <int>
<int> ## 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 ## <char> <int> <num> ## 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```

```r
# Check the class of the 'DATE' column to confirm it's in date format
class(transactionData$DATE)

## [1] "integer"```
We can see that the date column is in an integer format. Let's change this to a
date format.

```r
#### Convert DATE column to a date format
#### A quick search online tells us that CSV and Excel integer dates begin on 30 Dec 1899
transactionData$DATE <- as.Date(transactionData$DATE, origin = "1899-12-30")
```

We should check that we are looking at the right products by examining PROD_NAME.

```r
#### Examine PROD_NAME
summary(transactionData$PROD_NAME)

## Length Class Mode ## 264836 character character```
Looks like we are definitely looking at potato chips but how can we check that
these are all chips? We can do some basic text analysis by summarising the
individual words in the product name.

```r
#### Examine the words in PROD_NAME to see if there are any incorrect entries
#### such as products that are not chips

# Split PROD_NAME into individual words and store them in a data table
productWords <- data.table(unlist(strsplit(unique(transactionData$PROD_NAME), " ")))

# Set the name for the column in the data table
setnames(productWords, 'V1', 'words')
```
```

```
# Remove digits and special characters, then sort by frequency of occurrence
productWords <- productWords[!grepl(pattern = "[0-9]|&", productWords$words)]
productCounts <- productWords[, .N, by = words][order(-N)]
productCounts

## words N ## <char> <int> ## 1: 234 ## 2: Chips 21 ## 3: Smiths 16 ## 4:
Crinkle 14 ## 5: Cut 14 ## --- ## 168: Rst 1 ## 169: Pork 1 ## 170: Belly 1 ##
171: Pc 1 ## 172: Bolognese 1```
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 product words. We can do this using `grepl()`.

```r
# Over to you! Remove digits, and special characters, and then sort the distinct words by frequency of occurrence.
#### Removing digits

# Load the data.table library
library(data.table)

# Read in the product data - replace filePath with the actual path to your file
transactionData <- fread(paste0(filePath,"QVI_transaction_data.csv"))

# Split the PROD_NAME column into individual words
productWordsList <- strsplit(transactionData$PROD_NAME, " ")

# Unlist into a vector of words
productWordsVector <- unlist(productWordsList)

# Create a data table from this vector
productWords <- data.table(words = productWordsVector)

# Remove digits and special characters
productWords <- productWords[!grepl(pattern = "\\d+|\\&|*|\\(|\\)|\\-|\\/", words)]

# Count the frequency of each word and sort by frequency
wordFrequency <- productWords[, .N, by = words][order(-N)]

# Display the most common words
head(wordFrequency)

## words N ## <char> <int> ## 1: 504838 ## 2: Chips 49770 ## 3: Kettle 41288 ##
4: Smiths 28860 ## 5: Salt 27976 ## 6: Cheese 27890```

```r
# Remove digits and special characters from the words
cleanProductWords <- productWords[!grepl(pattern = "\\d+|\\&|\\*|\\(|\\)|\\-|\\/", words)]

# Count the frequency of each word and sort by frequency
wordFrequency <- cleanProductWords[, .N, by = words][order(-N)]

# Show the most common words, which should describe the chip products
head(wordFrequency)

## words N ## <char> <int> ## 1: 504838 ## 2: Chips 49770 ## 3: Kettle 41288 ##
```

4: Smiths 28860 ## 5: Salt 27976 ## 6: Cheese 27890```
Removing special characters
Let's look at the most common words by counting the number of times a word appears and
sorting them by this frequency in order of highest to lowest frequency
There are salsa products in the dataset but we are only interested in the chips
category, so lets remove these.

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

Next, we can use summary() to check summary statistics such as mean, min and max values for each feature to see if there are any obvious outliers in the data and if there are any nulls in any of the columns (NA's : number of nulls will appear in the output if there are any nulls).

```r
#### Summarise the data to check for nulls and possible outliers

# Summary statistics for each column to check for nulls and outliers
summary(transactionData)
```

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.: 70015 1st Qu.: 67569 ##
Median :43464 Median :130.0 Median : 130367 Median : 135183 ## Mean :43464 Mean
:135.1 Mean : 135531 Mean : 135131 ## 3rd Qu.:43555 3rd Qu.:203.0 3rd Qu.:
203084 3rd Qu.: 202654 ## Max. :43646 Max. :272.0 Max. :2373711 Max.
:2415841 ## PROD_NBR PROD_NAME PROD_QTY TOT_SALES ## Min. : 1.00 Length:246742
Min. : 1.000 Min. : 1.700 ## 1st Qu.: 26.00 Class :character 1st Qu.: 2.000
1st Qu.: 5.800 ## Median : 53.00 Mode :character Median : 2.000 Median : 7.400
Mean : 56.35 Mean : 1.908 Mean : 7.321 ## 3rd Qu.: 87.00 3rd Qu.: 2.000 3rd
Qu.: 8.800 ## Max. :114.00 Max. :200.000 Max. :650.000```

```r
# Additionally, check for the presence of any null values in the dataset
sapply(transactionData, function(x) sum(is.na(x)))
```

DATE STORE_NBR LYLTY_CARD_NBR TXN_ID ## 0 0 0 0 ## PROD_NBR PROD_NAME
PROD_QTY TOT_SALES ## 0 0 0 0```
There are no nulls in the columns but product quantity appears to have an outlier which we should investigate further. Lets investigate further the ca

```r
#### Filter the dataset to find the outlier
# Filter for transactions where 200 packets were purchased
outlierTransactions <- transactionData[PROD_QTY == 200, ]
outlierTransactions
```

DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR ## <int> <int> <int> <int>
<int> ## 1: 43331 226 226000 226201 4 ## 2: 43605 226 226000 226210 4 ##
PROD_NAME PROD_QTY TOT_SALES ## <char> <int> <num> ## 1: Dorito Corn Chp
Supreme 380g 200 650 ## 2: Dorito Corn Chp Supreme 380g 200 650```
There are two transactions where 200 packets of chips are bought in one transaction and both of these transactions were by the same customer.

```r
# Remove transactions by the outlier customer from the dataset
transactionData <- transactionData[!LYLTY_CARD_NBR %in% outlierTransactions$LYLTY_CARD_NBR, ]
```

It looks like this customer has only had the two transactions over the year and is not an ordinary retail customer. The customer might be buying chips for commercial purposes instead. Well remove this loyalty card number from further analysis.

```r
outlierCardNumber <- unique(outlierTransactions$LYLTY_CARD_NBR)
transactionData <- transactionData[!LYLTY_CARD_NBR %in% outlierCardNumber]
```

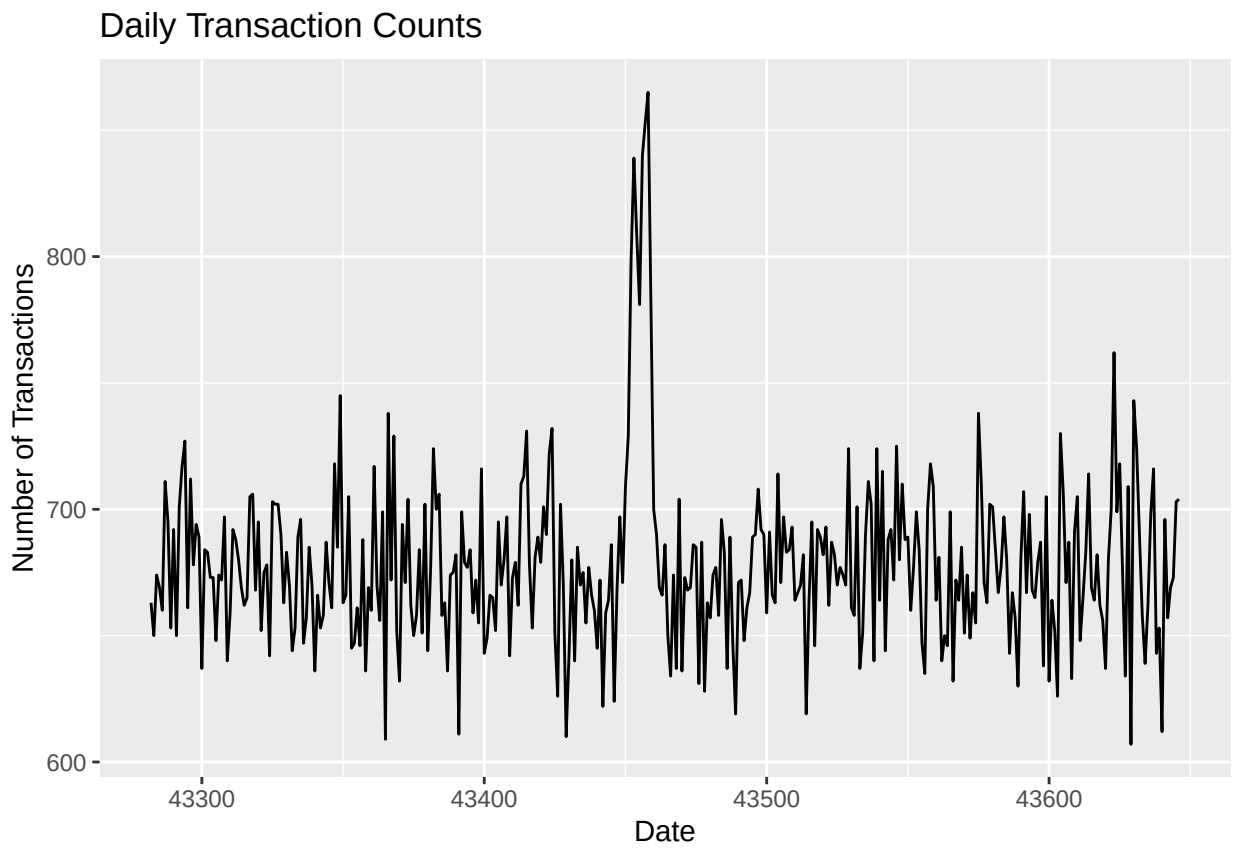
Thats better. Now, lets look at the number of transaction lines over time to see if there are any obvious data issues such as missing data.

```r
transactionsByDate <- transactionData[, .N, by = DATE]
transactionsByDate <- transactionsByDate[order(DATE)]

print(transactionsByDate)
```

```
## DATE N ## <int> <int> ## 1: 43282 663 ## 2: 43283 650 ## 3: 43284 674 ## 4:
43285 669 ## 5: 43286 660 ## --- ## 360: 43642 657 ## 361: 43643 669 ## 362:
43644 673 ## 363: 43645 703 ## 364: 43646 704```
```

```r
library(ggplot2)
ggplot(transactionsByDate, aes(x = DATE, y = N)) +
  geom_line() +
  labs(title = "Daily Transaction Counts", x = "Date", y = "Number of Transactions")
```



Theres only 364 rows, meaning only 364 dates which indicates a missing date. Lets create a sequence of dates from 1 Jul 2018 to 30 Jun 2019 and use this to create a chart of number of transactions over time to find the missing date.

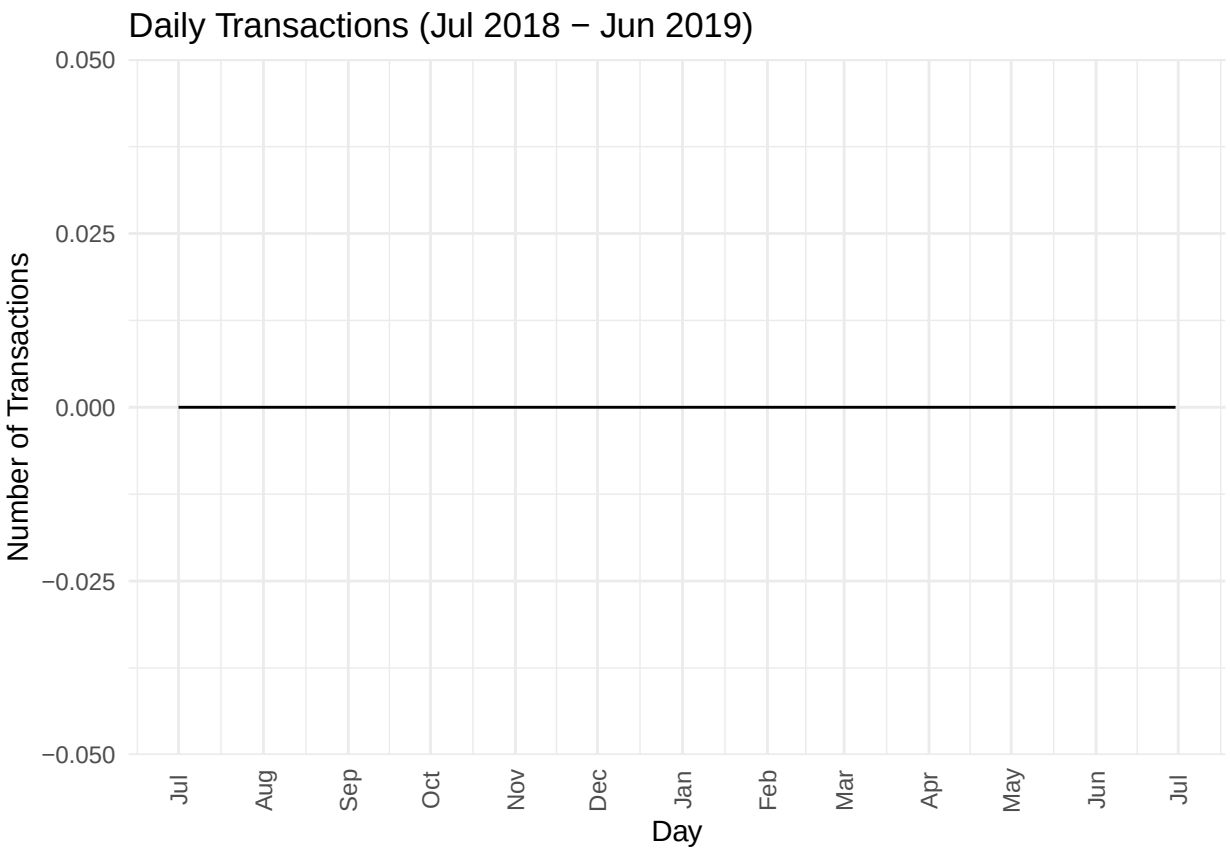
```
# Creating a sequence of dates from July 1, 2018, to June 30, 2019
# Ensure transactionData is a data.table
setDT(transactionData)

# Count transactions by DATE
transactionsBy_day <- transactionData[, .(N = .N), by = DATE]

# Creating a sequence of dates from July 1, 2018, to June 30, 2019
fullDateSequence <- seq(as.Date("2018-07-01"), as.Date("2019-06-30"), by = "day")
fullDates <- data.frame(DATE = fullDateSequence)

# Merge the full date sequence with the transaction count by date
transactionsByDateFull <- merge(fullDates, transactionsBy_day, by = "DATE", all.x = TRUE)
transactionsByDateFull$N[is.na(transactionsByDateFull$N)] <- 0

# Plot the number of transactions over time
ggplot(transactionsByDateFull, aes(x = DATE, y = N)) +
  geom_line() +
  labs(x = "Day", y = "Number of Transactions", title = "Daily Transactions (Jul 2018 - Jun 2019)") +
  scale_x_date(date_breaks = "1 month", date_labels = "%b") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```

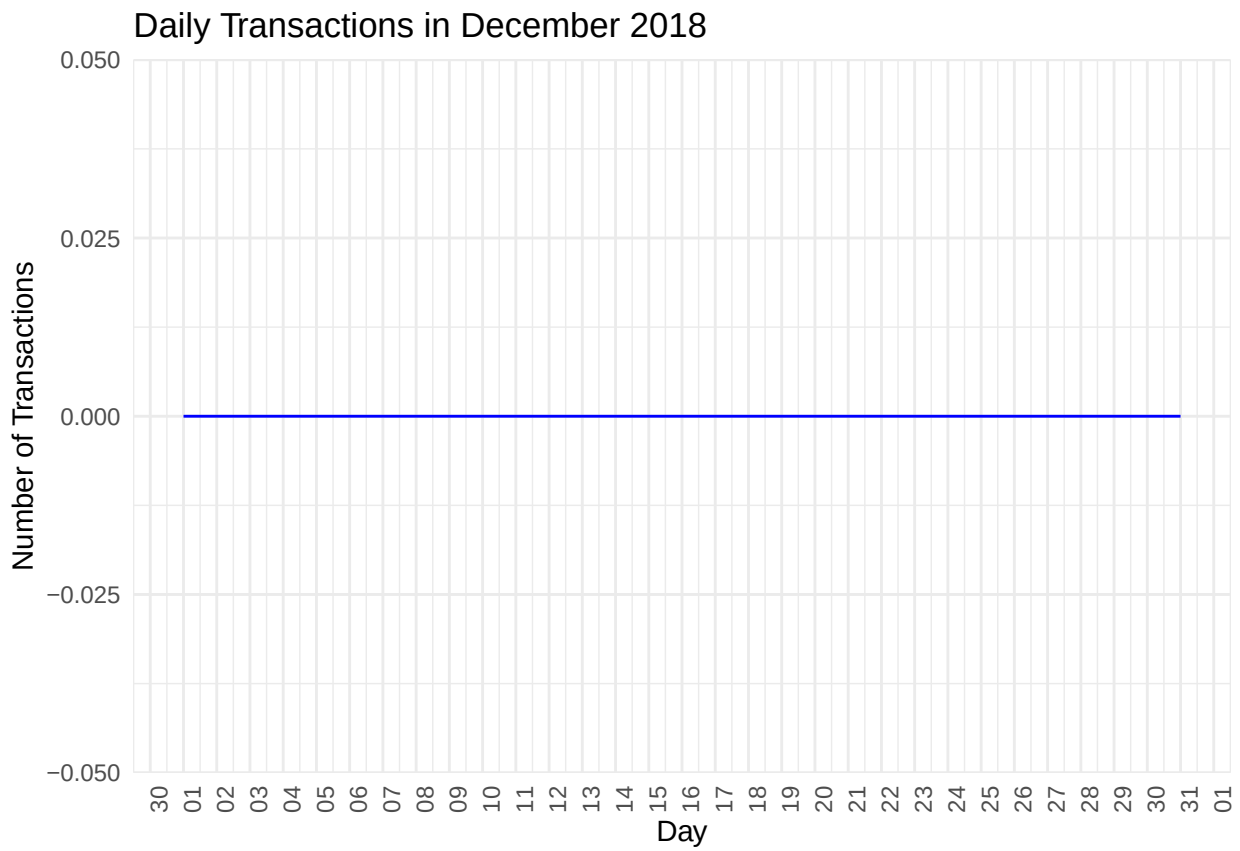


We can see that there is an increase in purchases in December and a break in late December. Lets zoom in on this.

```
# Filter the dataset to include only December 2018
transactionsByDateFull$DATE <- as.Date(transactionsByDateFull$DATE)

# Filter the dataset to include only December 2018
decemberTransactions <- transactionsByDateFull[transactionsByDateFull$DATE >= as.Date("2018-12-01") &
                                                 transactionsByDateFull$DATE <= as.Date("2018-12-31"), ]

# Plot the transactions for December 2018
ggplot(decemberTransactions, aes(x = DATE, y = N)) +
  geom_line(color = "blue") +
  labs(x = "Day", y = "Number of Transactions", title = "Daily Transactions in December 2018") +
  scale_x_date(date_breaks = "1 day", date_labels = "%d") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



We can see that the increase in sales occurs in the lead-up to Christmas and that there are zero sales on Christmas day itself. This is due to shops being closed on Christmas day. Now that we are satisfied that the data no longer has outliers, we can move on to creating other features such as brand of chips or pack size from PROD_NAME. We will start with pack size.

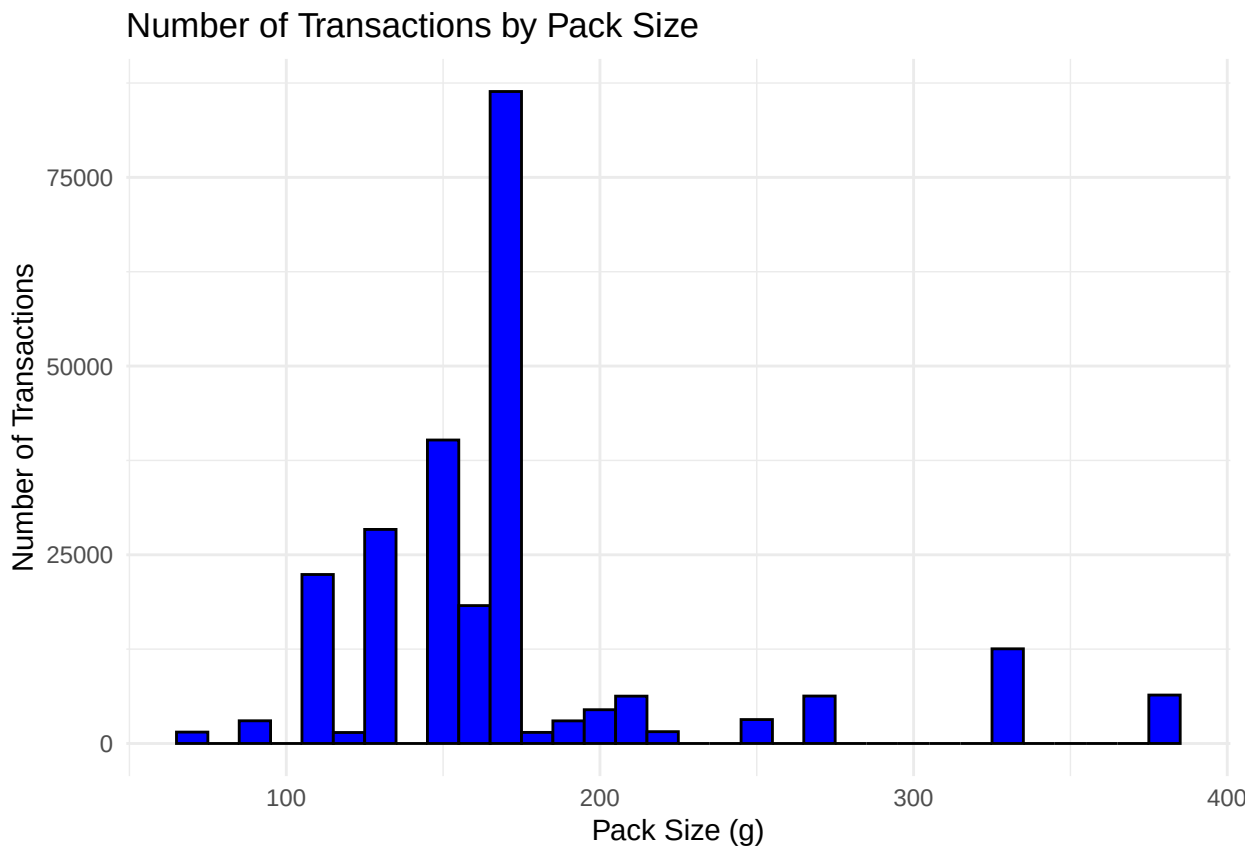
```
#### Pack size
#### We can work this out by taking the digits that are in PROD_NAME
transactionData[, PACK_SIZE := parse_number(PROD_NAME)]
#### Always check your output
#### Let's check if the pack sizes look sensible
transactionData[, .N, PACK_SIZE][order(PACK_SIZE)]
```

PACK_SIZE N ## <num> <int> ## 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 ## PACK_SIZE N```
The largest size is 380g and the smallest size is 70g - seems sensible!

```r
#### Let's plot a histogram of PACK_SIZE since we know that it is a categorical variable
#### and not a continuous variable even though it is numeric.
library(ggplot2)

# Plot a histogram of PACK_SIZE
ggplot(transactionData, aes(x = PACK_SIZE)) +
```

```
geom_histogram(binwidth = 10, fill = "blue", color = "black") +
labs(title = "Number of Transactions by Pack Size", x = "Pack Size (g)", y = "Number of Transactions") +
theme_minimal()
```



Pack sizes created look reasonable. Now to create brands, we can use the first word in PROD_NAME to work out the brand name…

```
# Extract the first word from PROD_NAME as the brand name
transactionData[, BRAND := sapply(strsplit(PROD_NAME, " "), `[`, 1)]

# View the unique brands to check the extraction
unique(transactionData$BRAND)
```

```
## [1] "Natural" "CCs" "Smiths" "Kettle" "Grain" ## [6] "Doritos" "Twisties"
"WW" "Thins" "Burger" ## [11] "NCC" "Cheezels" "Infzns" "Red" "Pringles" ##
[16] "Dorito" "Infuzions" "Smith" "GrnWves" "Tyrrells" ## [21] "Cobs" "French"
"RRD" "Tostitos" "Cheetos" ## [26] "Woolworths" "Snbts" "Sunbites"```

```r
# Check the unique brands extracted
unique(transactionData$BRAND)
```

```
## [1] "Natural" "CCs" "Smiths" "Kettle" "Grain" ## [6] "Doritos" "Twisties"
"WW" "Thins" "Burger" ## [11] "NCC" "Cheezels" "Infzns" "Red" "Pringles" ##
[16] "Dorito" "Infuzions" "Smith" "GrnWves" "Tyrrells" ## [21] "Cobs" "French"
```
```

"RRD" "Tostitos" "Cheetos" ## [26] "Woolworths" "Snbts" "Sunbites"```
Some of the brand names look like they are of the same brands - such as RED and RRD, which are both Red Rock Deli chips. Lets combine these f

```r
# Adjusting for the specific example provided
transactionData[BRAND == "RED" | BRAND == "RRD", BRAND := "Red Rock Deli"]
# Checking the brands again
unique(transactionData$BRAND)
```

```
## [1] "Natural" "CCs" "Smiths" "Kettle" ## [5] "Grain" "Doritos" "Twisties"
"WW" ## [9] "Thins" "Burger" "NCC" "Cheezels" ## [13] "Infzns" "Red" "Pringles"
"Dorito" ## [17] "Infuzions" "Smith" "GrnWves" "Tyrrells" ## [21] "Cobs"
"French" "Red Rock Deli" "Tostitos" ## [25] "Cheetos" "Woolworths" "Snbts"
"Sunbites"```
```

Examining customer data
Now that we are happy with the transaction dataset, lets have a look at the customer dataset.

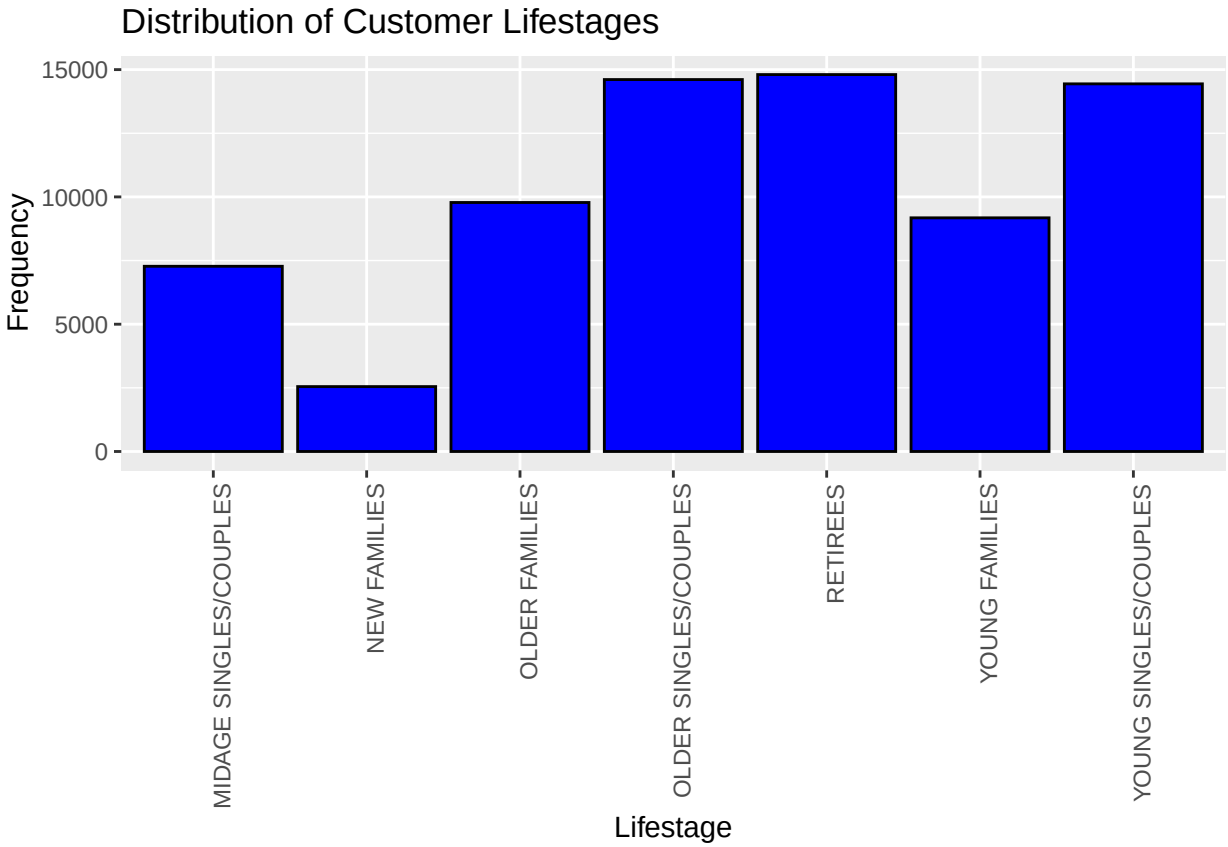
```r
# Summary statistics of the customer dataset
summary(customerData)
```

```
## LYLTY_CARD_NBR LIFESTAGE PREMIUM_CUSTOMER ## Min.  : 1000 Length:72637
Length:72637 ## 1st Qu.: 66202 Class :character Class :character ## Median :
134040 Mode :character Mode :character ## Mean : 136186 ## 3rd Qu.: 203375 ##
Max.  :2373711```
```

```r
# Checking for null values in each column
sapply(customerData, function(x) sum(is.na(x)))
```

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

```r
# Optional: Visualize distributions of key columns
# For example, if 'AGE' is a column
library(ggplot2)
library(ggplot2)
ggplot(customerData, aes(x = LIFESTAGE)) +
  geom_bar(fill = "blue", color = "black") +
  labs(title = "Distribution of Customer Lifestages", x = "Lifestage", y = "Frequency") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
#### Merge transaction data to customer data
data <- merge(transactionData, customerData, by = "LYLTY_CARD_NBR", all.x = TRUE)
```

As the number of rows in data is the same as that of transactionData, we can be sure that no duplicates were created. This is because we created data by setting all.x = TRUE (in other words, a left join) which means take all the rows in transactionData and find rows with matching values in shared columns and then joining the details in these rows to the x or the first mentioned table.
Lets also check if some customers were not matched on by checking for nulls.

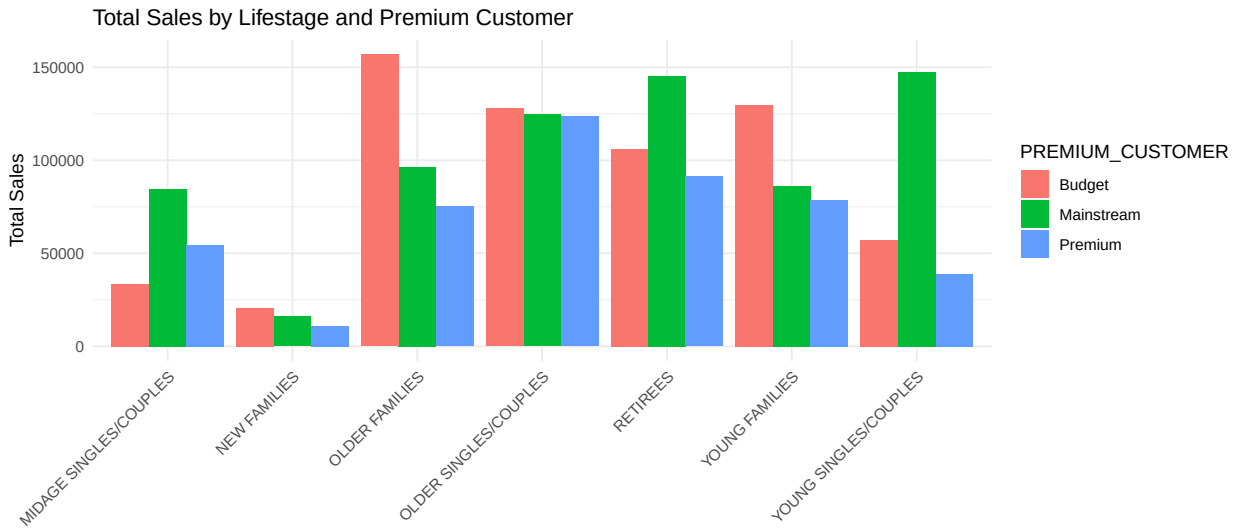
```
numUnmatchedTransactions <- sum(is.na(data$CustomerSegment))
```

Great, there are no nulls! So all our customers in the transaction data has been accounted for in the customer dataset. Note that if you are continuing with Task 2, you may want to retain this dataset which you can write out as a csv

```
fwrite(data, paste0(filePath,"QVI_data.csv"))
```

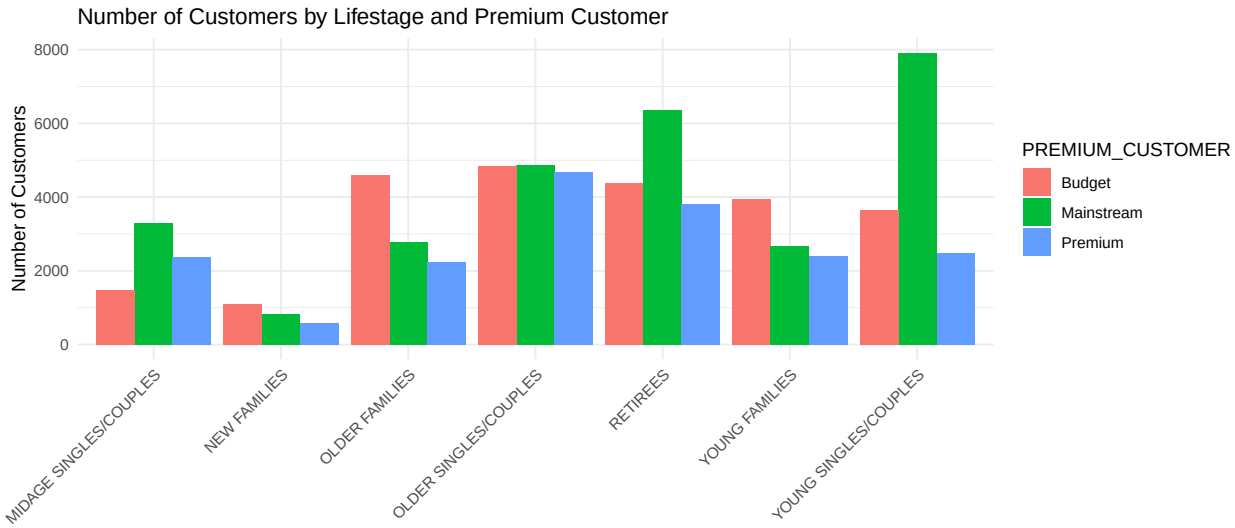
Data exploration is now complete! ## Data analysis on customer segments Now that the data is ready for analysis, we can define some metrics of interest to the client: - Who spends the most on chips (total sales), describing customers by lifestage and how premium their general purchasing behaviour is - How many customers are in each segment - How many chips are bought per customer by segment - Whats the average chip price by customer segment We could also ask our data team for more information. Examples are: - The customers total spend over the period and total spend for each transaction to understand what proportion of their grocery spend is on chips - Proportion of customers in each customer segment overall to compare against the mix of customers who purchase chips Lets start with calculating total sales by LIFESTAGE and PREMIUM_CUSTOMER and plotting the split by these segments to describe which customer segment contribute most to chip sales.

```
#### Total sales by LIFESTAGE and PREMIUM_CUSTOMER
totalSales <- data[, .(TotalSales = sum(TOT_SALES)), by = .(LIFESTAGE, PREMIUM_CUSTOMER)]
ggplot(totalSales, aes(x = LIFESTAGE, y = TotalSales, fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(title = "Total Sales by Lifestage and Premium Customer", x = "", y = "Total Sales") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



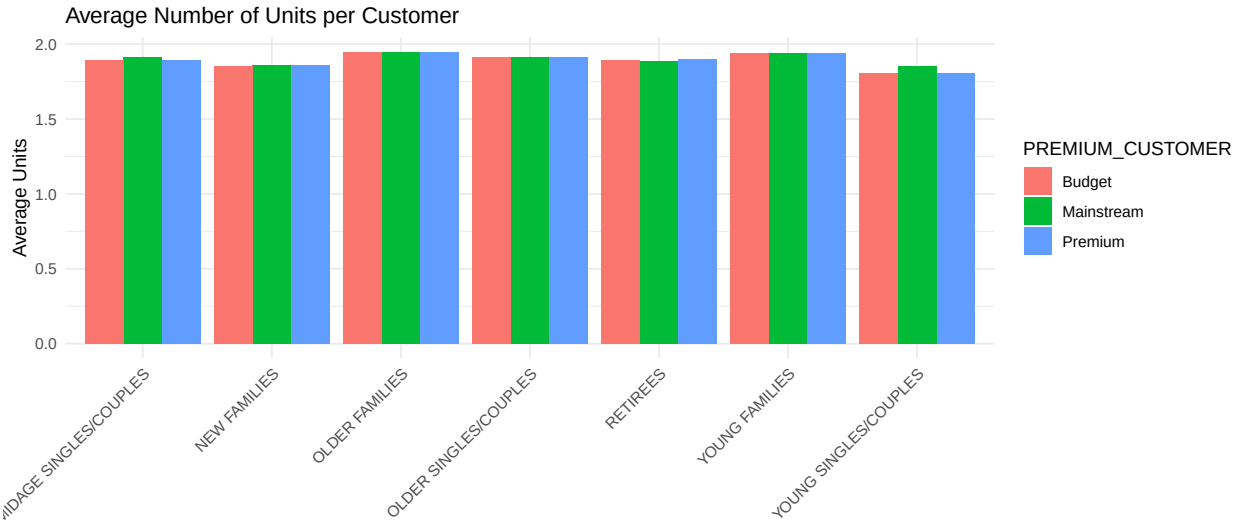
Sales are coming mainly from Budget - older families, Mainstream - young singles/couples, and Mainstream - retirees Lets see if the higher sales are due to there being more customers who buy chips.

```
#### Number of customers by LIFESTAGE and PREMIUM_CUSTOMER
customerCounts <- data[, .(CustomerCount = uniqueN(LYLTY_CARD_NBR)), by = .(LIFESTAGE,
↪  PREMIUM_CUSTOMER)]
ggplot(customerCounts, aes(x = LIFESTAGE, y = CustomerCount, fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(title = "Number of Customers by Lifestage and Premium Customer", x = "", y = "Number of Customers") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



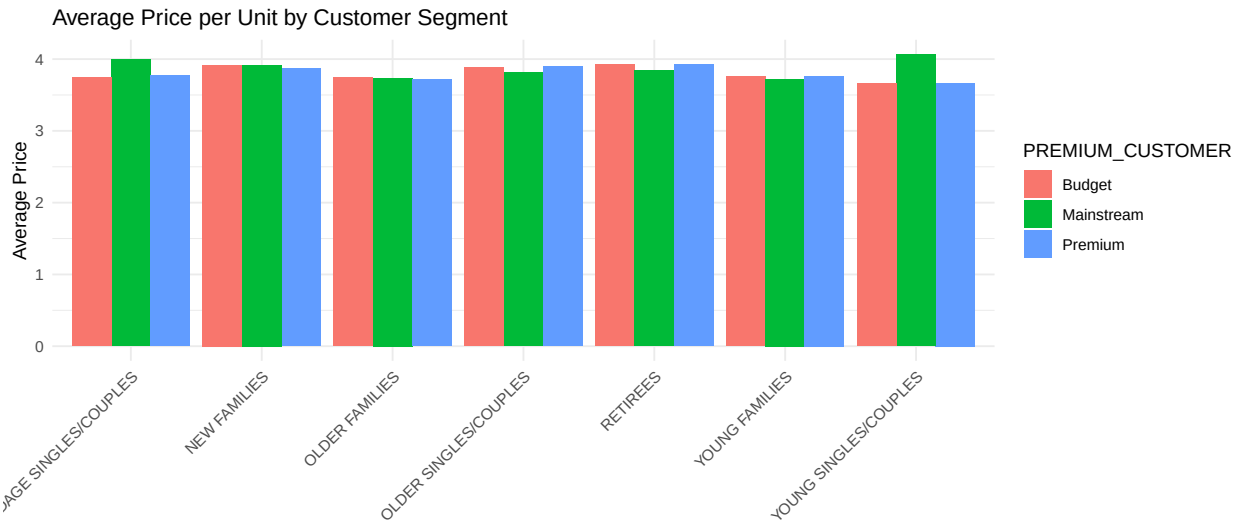
There are more Mainstream - young singles/couples and Mainstream - retirees who buy chips. This contributes to there being more sales to these customer segments but this is not a major driver for the Budget - Older families segment. Higher sales may also be driven by more units of chips being bought per customer. Lets have a look at this next.

```
#### Average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER
averageUnits <- data[, .(AverageUnits = mean(PROD_QTY)), by = .(LIFESTAGE, PREMIUM_CUSTOMER)]
ggplot(averageUnits, aes(x = LIFESTAGE, y = AverageUnits, fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(title = "Average Number of Units per Customer", x = "", y = "Average Units") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Older families and young families in general buy more chips per customer Lets also investigate the average price per unit chips bought for each customer segment as this is also a driver of total sales.

```
#### Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER
data[, UnitPrice := TOT_SALES / PROD_QTY]
averagePrice <- data[, .(AveragePrice = mean(UnitPrice)), by = .(LIFESTAGE, PREMIUM_CUSTOMER)]
ggplot(averagePrice, aes(x = LIFESTAGE, y = AveragePrice, fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(title = "Average Price per Unit by Customer Segment", x = "", y = "Average Price") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Mainstream midage and young singles and couples are more willing to pay more per packet of chips compared to their budget and premium counterparts. This may be due to premium shoppers being more likely to buy healthy snacks and when they buy chips, this is mainly for entertainment purposes rather than their own consumption. This is also supported by there being fewer premium midage and young singles and couples buying chips compared to their mainstream counterparts. As the difference in average price per unit isnt large, we can check if this difference is statistically different.

```
#### Perform an independent t-test between mainstream vs premium and budget midage and
#### young singles and couples
transactionData$UnitPrice <- transactionData$TOT_SALES / transactionData$PROD_QTY
mainstream_ymc <- subset(data, LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES")
 ↪  & PREMIUM_CUSTOMER == "Mainstream")
budget_premium_ymc <- subset(data, LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE
 ↪  SINGLES/COUPLES") & PREMIUM_CUSTOMER %in% c("Budget", "Premium"))
t_test_result <- t.test(mainstream_ymc$UnitPrice, budget_premium_ymc$UnitPrice)
print(t_test_result)
```

Welch Two Sample t-test ## ## data: mainstream_ymc$UnitPrice and budget_premium_ymc$UnitPrice ## t = 37.624, df = 54791, p-value < 2.2e-16 ## alternative hypothesis: true difference in means is not equal to 0 ## 95 percent confidence interval: ## 0.3159319 0.3506572 ## sample estimates: ## mean of x mean of y ## 4.039786 3.706491```

The t-test results in a p-value of XXXXXXX, i.e. the unit price for mainstream, young and mid-age singles and couples [ARE / ARE NOT] significant

Deep dive into specific customer segments for insights
We have found quite a few interesting insights that we can dive deeper into.
We might want to target customer segments that contribute the most to sales to retain them or further increase sales. Lets look at Mainstream - young singles/couples. For instance, lets find out if they tend to buy a particular brand of chips.

```r
#### Deep dive into Mainstream, young singles/couples
brand_preference <- table(mainstream_ymc$Brand)
brand_preference_sorted <- sort(brand_preference, decreasing = TRUE)
top_brands <- head(brand_preference_sorted, 5)
print(top_brands)


## integer(0)```
We can see that :
This code will help identify the top brands preferred by mainstream young singles/couples.

```r
#### Preferred pack size compared to the rest of the population
library(stringr)
transactionData$PackSize <- as.numeric(str_extract(transactionData$PROD_NAME, "\\d+"))
# Analysing the preferred pack size for Mainstream Young Singles/Couples
pack_size_preference <- table(mainstream_ymc$PackSize)
pack_size_preference_sorted <- sort(pack_size_preference, decreasing = TRUE)
top_pack_sizes <- head(pack_size_preference_sorted, 5)
print(top_pack_sizes)


## integer(0)
```
```