Quantium Virtual Internship - Retail Strategy and Analytics - Task 1

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.

Load required libraries and datasets.

```
# Example code to install packages
# Load required libraries
library(knitr)
library(data.table)
library(ggplot2)
library(ggmosaic)
library(readr)
library(readxl)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Point the file Path to where you have downloaded the data sets to and assign the data files to
↪  data.tables

setwd("C:/Users/saisr/OneDrive/Documents/Job Projects/QuanitumR/")
transactionData <- read_excel("QVI_transaction_data.xlsx")
transactionData <- as.data.table(transactionData)
customerData <- fread("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

Let's check if columns we would expect to be numeric are in numeric form and date columns are in date format.

```
# Examine the data using one or more of the methods described above.
str(transactionData)
```

```
## Classes 'data.table' and 'data.frame': 264836 obs. of 8 variables:
## $ DATE : num 43390 43599 43605 43329 43330 ...
## $ STORE_NBR : num 1 1 1 2 2 4 4 4 5 7 ...
## $ LYLTY_CARD_NBR: num 1000 1307 1343 2373 2426 ...
## $ TXN_ID : num 1 348 383 974 1038 ...
## $ PROD_NBR : num 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 : num 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>
```

We can see that the date column is in an integer format. Let's change this to a date format.

```
#### 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.

```
#### Examine PROD_NAME
#Generate a summary of the PROD_NAME column.
transactionData %>%
  mutate(Product_Name = PROD_NAME) %>%
  group_by(Product_Name) %>%
  count(sort = TRUE)
```

```
## # A tibble: 114 x 2
## # Groups:   Product_Name [114]
##    Product_Name                                 n
##    <chr>                                    <int>
##  1 Kettle Mozzarella   Basil & Pesto 175g    3304
##  2 Kettle Tortilla ChpsHny&Jlpno Chili 150g  3296
##  3 Cobs Popd Swt/Chlli &Sr/Cream Chips 110g  3269
##  4 Tyrrells Crisps     Ched & Chives 165g    3268
##  5 Cobs Popd Sea Salt  Chips 110g            3265
##  6 Kettle 135g Swt Pot Sea Salt              3257
```

```
##   7 Tostitos Splash Of  Lime 175g             3252
##   8 Infuzions Thai SweetChili PotatoMix 110g  3242
##   9 Smiths Crnkle Chip  Orgnl Big Bag 380g    3233
##  10 Thins Potato Chips  Hot & Spicy 175g      3229
## # i 104 more rows
```

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.

```
#### Examine the words in PROD_NAME to see if there are any incorrect entries such as products that
↪  are not chips
productWords <- data.table(unlist(strsplit(unique(transactionData$PROD_NAME), " ")))
setnames(productWords, 'words')
```

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.

```
# Remove digits, and special characters, and then sort the distinct words by frequency of
↪  occurrence.

#### Removing digits
# productWords[, isDigit := grepl(pattern = "\\d", x = productWords$words)]
# productWords <- productWords[isDigit == FALSE, ][, isDigit := NULL]

productWords <- productWords[grepl("\\d", words) == FALSE, ]

#### Removing special characters
# productWords[, isSpecial := grepl(pattern = "[[:punct:]_]", x = productWords$words)]
# productWords <- productWords[isSpecial == FALSE, ][, isSpecial := NULL]

productWords <- productWords[grepl("[:alpha:]", words), ]

#### 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
productWords %>%
  mutate(Words = words) %>%
  group_by(Words) %>%
  count(sort = TRUE)
```

```
## # A tibble: 131 x 2
## # Groups:   Words [131]
##    Words        n
##    <chr>    <int>
##  1 Chips       21
##  2 Smiths      16
##  3 Crinkle     14
##  4 Kettle      13
##  5 Cheese      12
##  6 Salt        12
##  7 Original    10
##  8 Chip         9
##  9 Salsa        9
```

```
## 10 Pringles     8
## # i 121 more rows
```

There are salsa products in the dataset but we are only interested in the chips category, so let's 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(transactionData)
```

```
##       DATE              STORE_NBR     LYLTY_CARD_NBR        TXN_ID
##  Min.   :2018-07-01   Min.   :  1.0   Min.   :   1000   Min.   :      1
##  1st Qu.:2018-09-30   1st Qu.: 70.0   1st Qu.:  70015   1st Qu.:  67569
##  Median :2018-12-30   Median :130.0   Median : 130367   Median : 135183
##  Mean   :2018-12-30   Mean   :135.1   Mean   : 135531   Mean   : 135131
##  3rd Qu.:2019-03-31   3rd Qu.:203.0   3rd Qu.: 203084   3rd Qu.: 202654
##  Max.   :2019-06-30   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
```

There are no nulls in the columns but product quantity appears to have an outlier which we should investigate further. Let's investigate further the case where 200 packets of chips are bought in one transaction.

```r
#### Filter the dataset to find the outlier
transactionData[PROD_QTY == 200, ]
```

```
##          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
```

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
#### Let's see if the customer has had other transactions
transactionData[LYLTY_CARD_NBR == 226000, ]
```

```
##          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
```

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. We'll remove this loyalty card number from further analysis.

```r
#### Filter out the customer based on the loyalty card number
transactionData <- transactionData[LYLTY_CARD_NBR != 226000,]

#### Re-examine transaction data
summary(transactionData)
```

```
##       DATE              STORE_NBR     LYLTY_CARD_NBR        TXN_ID
##  Min.   :2018-07-01   Min.   :  1.0   Min.   :   1000   Min.   :      1
##  1st Qu.:2018-09-30   1st Qu.: 70.0   1st Qu.:  70015   1st Qu.:  67569
##  Median :2018-12-30   Median :130.0   Median : 130367   Median : 135182
##  Mean   :2018-12-30   Mean   :135.1   Mean   : 135530   Mean   : 135130
##  3rd Qu.:2019-03-31   3rd Qu.:203.0   3rd Qu.: 203083   3rd Qu.: 202652
##  Max.   :2019-06-30   Max.   :272.0   Max.   :2373711   Max.   :2415841
##     PROD_NBR       PROD_NAME            PROD_QTY        TOT_SALES
##  Min.   :  1.00   Length:246740      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   :1.906   Mean   : 7.316
##  3rd Qu.: 87.00                      3rd Qu.:2.000   3rd Qu.: 8.800
##  Max.   :114.00                      Max.   :5.000   Max.   :29.500
```

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

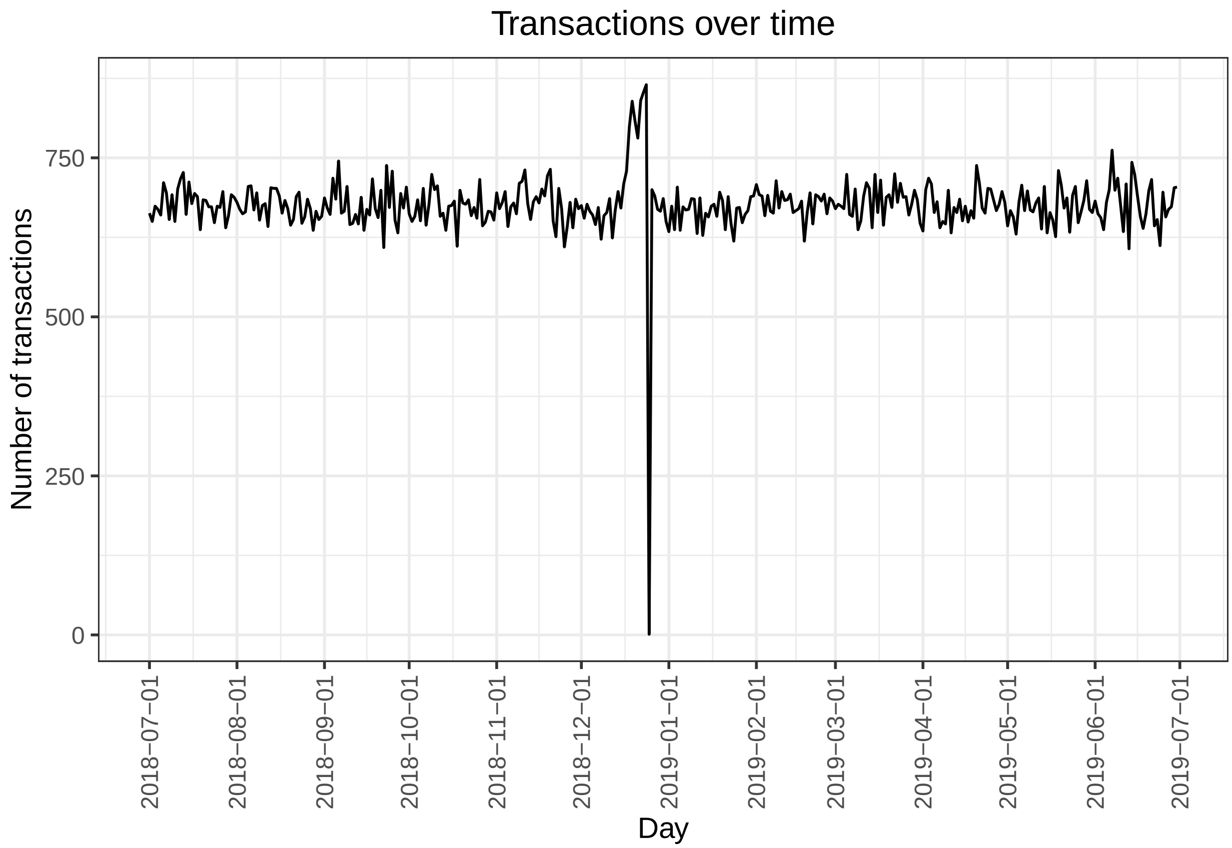
```r
#### Count the number of transactions by date
transactionData %>%
    mutate(Date = DATE) %>%
    group_by(Date) %>%
    summarise(Total_num = n())%>%
    arrange(desc(Total_num))
```

```
## # A tibble: 364 x 2
##    Date       Total_num
##    <date>         <int>
##  1 2018-12-24       865
##  2 2018-12-23       853
##  3 2018-12-22       840
##  4 2018-12-19       839
##  5 2018-12-20       808
##  6 2018-12-18       799
##  7 2018-12-21       781
```

```
##   8 2019-06-07       762
##   9 2018-09-06       745
## 10 2019-06-14       743
## # i 354 more rows
```

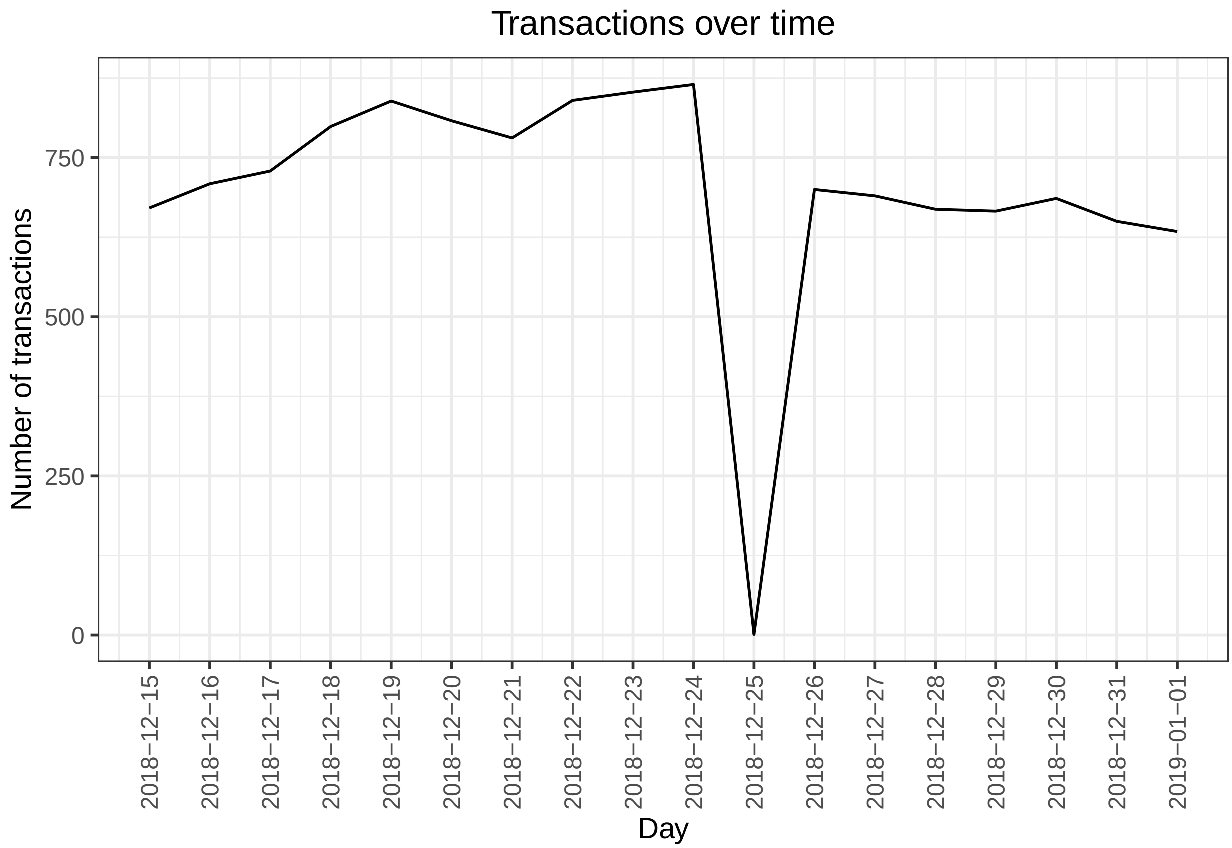
There's only 364 rows, meaning only 364 dates which indicates a missing date. Let's create a sequence of dates from 1 Jul 2018 to 30 Jun 2019 and use this to create a chart of number of transactions over time to find the missing date.

```
#### Create a sequence of dates and join this the count of transactions by date
# Create a column of dates that includes every day from 1 Jul 2018 to 30 Jun 2019, and join it
↪  onto the data to fill in the missing day.

all_dates <- data.table(DATE = seq(as.Date("2018-07-01"), as.Date("2019-06-30"), by="days"))
transactionData <- all_dates %>% left_join(transactionData, by= c("DATE" = "DATE"))

transactions_by_day <- transactionData %>%
    mutate(Date = DATE) %>%
    group_by(Date) %>%
    summarise(N = n())

#### Setting 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))
```



We can see that there is an increase in purchases in December and a big dip in late December. Let's zoom in on this.

```
#### Filter to December and look at individual days
# Recreate the chart above zoomed in to the relevant dates.

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 day", limits = as.Date(c("2018-12-15","2019-01-01")) ) +
 theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



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
##  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
```

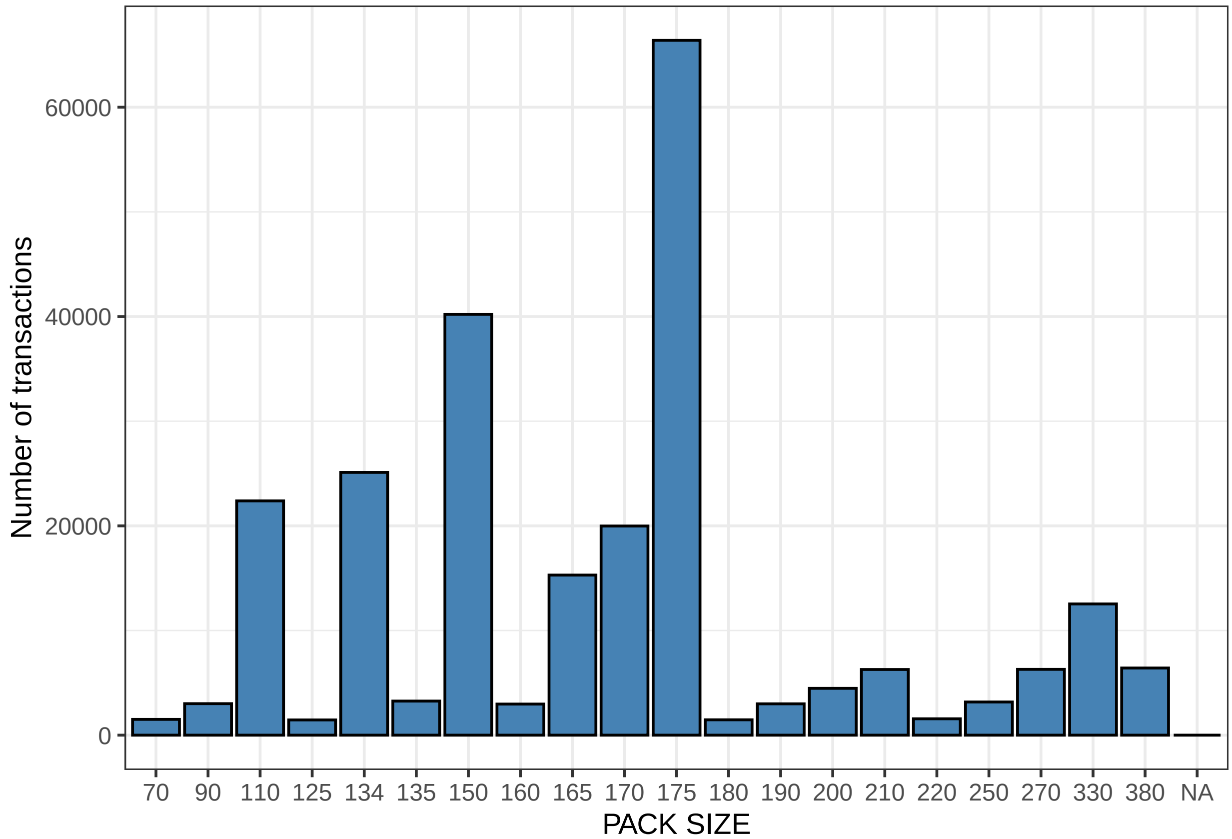
```
## 12:        180  1468
## 13:        190  2995
## 14:        200  4473
## 15:        210  6272
## 16:        220  1564
## 17:        250  3169
## 18:        270  6285
## 19:        330 12540
## 20:        380  6416
## 21:         NA     1
##     PACK_SIZE     N
```

The largest size is 380g and the smallest size is 70g - seems sensible!

```
#### 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.
head(transactionData)
```

```
##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
## 1: 2018-07-01        47          47142  42540       14
## 2: 2018-07-01        55          55073  48884       99
## 3: 2018-07-01        55          55073  48884       91
## 4: 2018-07-01        58          58351  54374      102
## 5: 2018-07-01        68          68193  65598       44
## 6: 2018-07-01        69          69207  67156       49
##                                  PROD_NAME PROD_QTY TOT_SALES PACK_SIZE
## 1:   Smiths Crnkle Chip  Orgnl Big Bag 380g        2      11.8       380
## 2:         Pringles Sthrn FriedChicken 134g        2       7.4       134
## 3:                 CCs Tasty Cheese    175g        2       4.2       175
## 4:   Kettle Mozzarella   Basil & Pesto 175g        2      10.8       175
## 5:           Thins Chips Light&  Tangy 175g        2       6.6       175
## 6: Infuzions SourCream&Herbs Veg Strws 110g        2       7.6       110
```

```
# Plot a histogram showing the number of transactions by pack size.

ggplot(transactionData, aes(x=factor(PACK_SIZE)))+
  geom_histogram(stat = "Count", color="black", fill="steelblue") +
  xlab("PACK SIZE") + ylab("Number of transactions")
```



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

```r
#### Brands
# Create a column which contains the brand of the product, by extracting it from the product name.

library("stringr")
transactionData[, BRAND := word(transactionData$PROD_NAME, 1)]

#### Checking brands
# Check the results look reasonable.
transactionData[, .N, by = BRAND][order(-N)]
```

```
##          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
## 29:       <NA>     1
##          BRAND     N
```

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. Let's combine these together.

```
#### Clean brand names
transactionData[BRAND == "RED", BRAND := "RRD"]
transactionData[BRAND == "Red", BRAND := "RRD"]
transactionData[BRAND == "Dorito", BRAND := "Doritos"]
transactionData[BRAND == "Smith", BRAND := "Smiths"]
transactionData[BRAND == "Infzns", BRAND := "Infuzions"]
transactionData[BRAND == "Snbts", BRAND := "Sunbites"]
transactionData[BRAND == "WW", BRAND := "Woolworths"]
transactionData[BRAND == "Grain", BRAND := "GrnWves"]
transactionData[BRAND == "NCC", BRAND := "Natural"]

#### Check again
transactionData[, .N, BRAND][order(BRAND)]
```

```
##          BRAND     N
##  1:     Burger  1564
##  2:        CCs  4551
##  3:    Cheetos  2927
##  4:   Cheezels  4603
##  5:       Cobs  9693
##  6:    Doritos 25224
##  7:     French  1418
##  8:    GrnWves  7740
##  9:  Infuzions 14201
## 10:     Kettle 41288
## 11:    Natural  7469
## 12:   Pringles 25102
## 13:        RRD 16321
## 14:     Smiths 30353
## 15:   Sunbites  3008
## 16:      Thins 14075
```

```
## 17:    Tostitos  9471
## 18:    Twisties  9454
## 19:    Tyrrells  6442
## 20: Woolworths 11836
## 21:        <NA>     1
##          BRAND     N
```

```
# Check the results look reasonable.
transactionData <- na.omit(transactionData)
```

Examining customer data

Now that we are happy with the transaction dataset, let's have a look at the customer dataset.

```
#### Examining customer data
# Do some basic summaries of the dataset, including distributions of any key columns.

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>
```

```
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
```

```
# Frequency Tables

customerData %>%
        count(LIFESTAGE, sort= TRUE)
```

```
##                  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
```

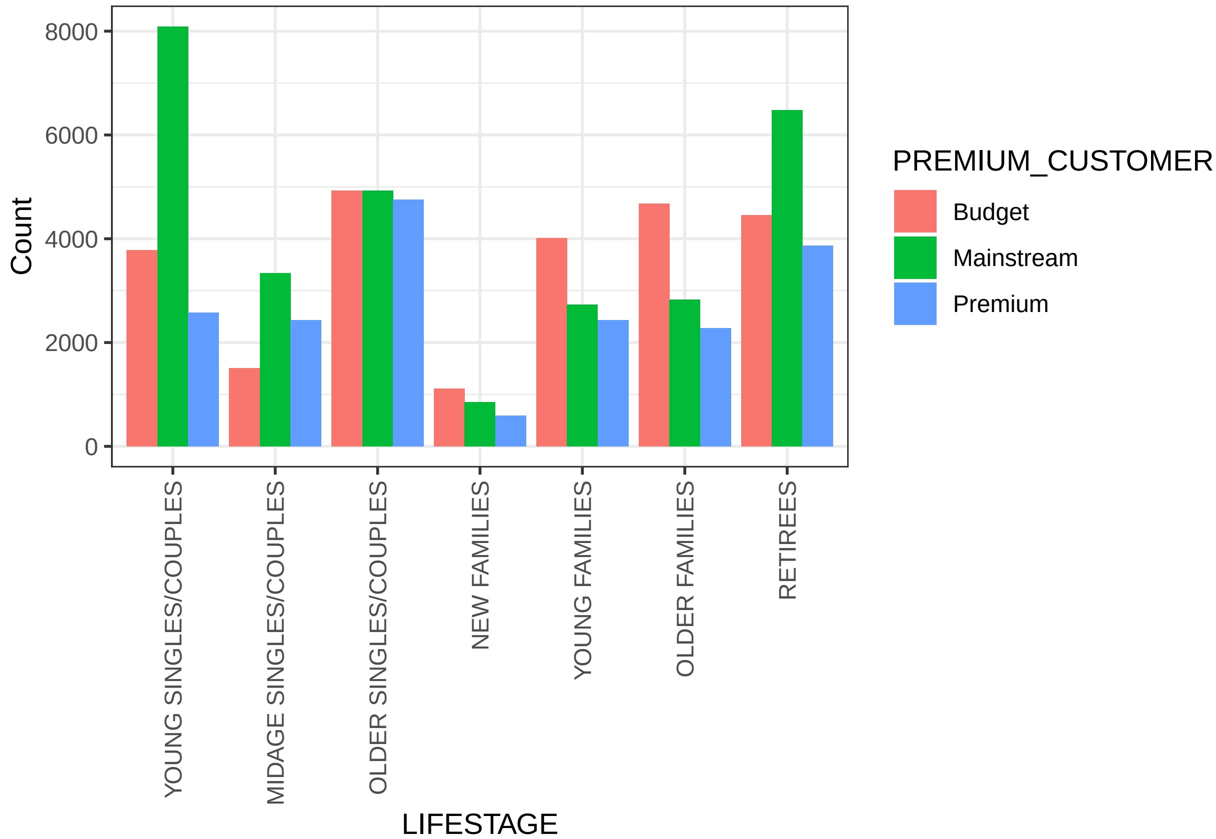
```r
customerData %>%
        count(PREMIUM_CUSTOMER, sort= TRUE)
```

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

```r
table(customerData$PREMIUM_CUSTOMER, customerData$LIFESTAGE)
```

```
##
##               MIDAGE SINGLES/COUPLES NEW FAMILIES OLDER FAMILIES
##   Budget                        1504         1112           4675
##   Mainstream                    3340          849           2831
##   Premium                       2431          588           2274
##
##               OLDER SINGLES/COUPLES RETIREES YOUNG FAMILIES
##   Budget                       4929     4454           4017
##   Mainstream                   4930     6479           2728
##   Premium                      4750     3872           2433
##
##               YOUNG SINGLES/COUPLES
##   Budget                       3779
##   Mainstream                   8088
##   Premium                      2574
```

```r
# Distributions

level_order = c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES", "OLDER SINGLES/COUPLES", "NEW
↪  FAMILIES", "YOUNG FAMILIES", "OLDER FAMILIES", "RETIREES")

customerData %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarize(Count = n()) %>%
  ggplot(aes(x=LIFESTAGE, y=Count, fill=PREMIUM_CUSTOMER)) +
  geom_bar(stat='identity', position= "dodge") +
  scale_x_discrete(limits = level_order) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```




```
# No null values
sum(is.na(customerData))
```

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

As there do not seem to be any issues with the customer data, we can now go ahead and join the transaction and customer data sets together

```
#### Merge transaction data to customer data
data <- merge(transactionData, customerData, 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.

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

```
# See if any transactions did not have a matched customer.
sum(is.na(data))
```

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

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("QVI_data.csv"))
```

Data exploration is now complete!

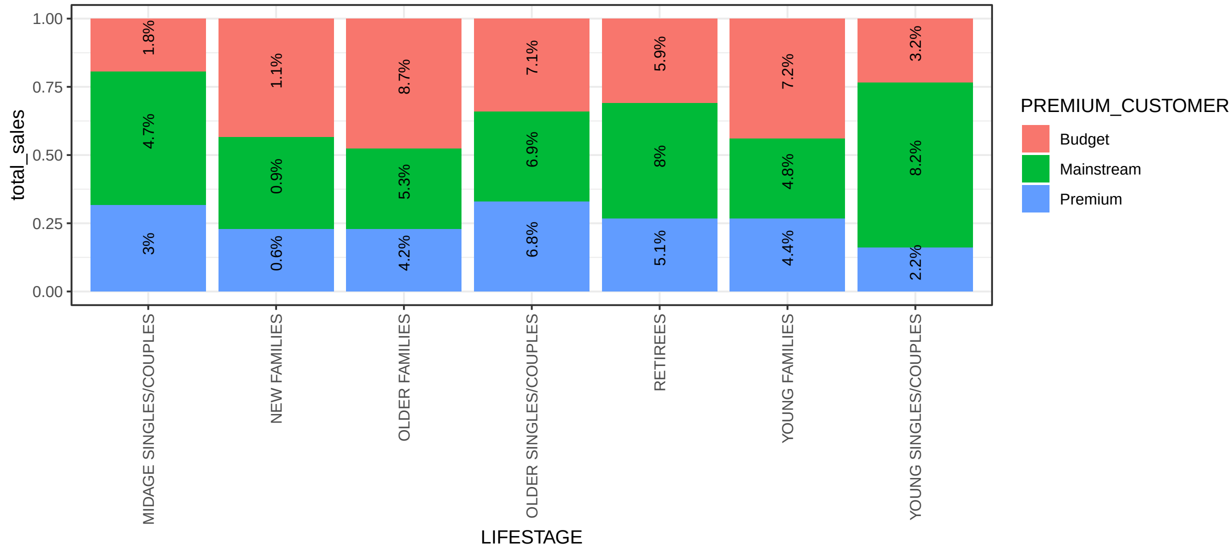
Data analysis on customer segments

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 - What's the average chip price by customer segment

We could also ask our data team for more information. Examples are: - The customer's 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

Let's 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
# Calculate the summary of sales by those dimensions and create a plot.
sales <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarize(total_sales = sum(TOT_SALES)) %>%
  arrange(desc(total_sales))

total_sales_all <- sum(sales$total_sales)

sales <- sales %>%
  mutate(percent = total_sales / total_sales_all * 100)

p <- ggplot(sales, aes(fill=PREMIUM_CUSTOMER, y=total_sales, x=LIFESTAGE)) +
    geom_bar(position="fill", stat="identity") +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
    geom_text(aes(label = paste0(round(percent, 1), "%"),
                  y = 0.5 * total_sales),
              position = position_fill(vjust = 0.5),
              size = 3, color = "black",
              angle = 90,
              hjust = 0.3)+
  coord_cartesian(clip = "off")

p
```

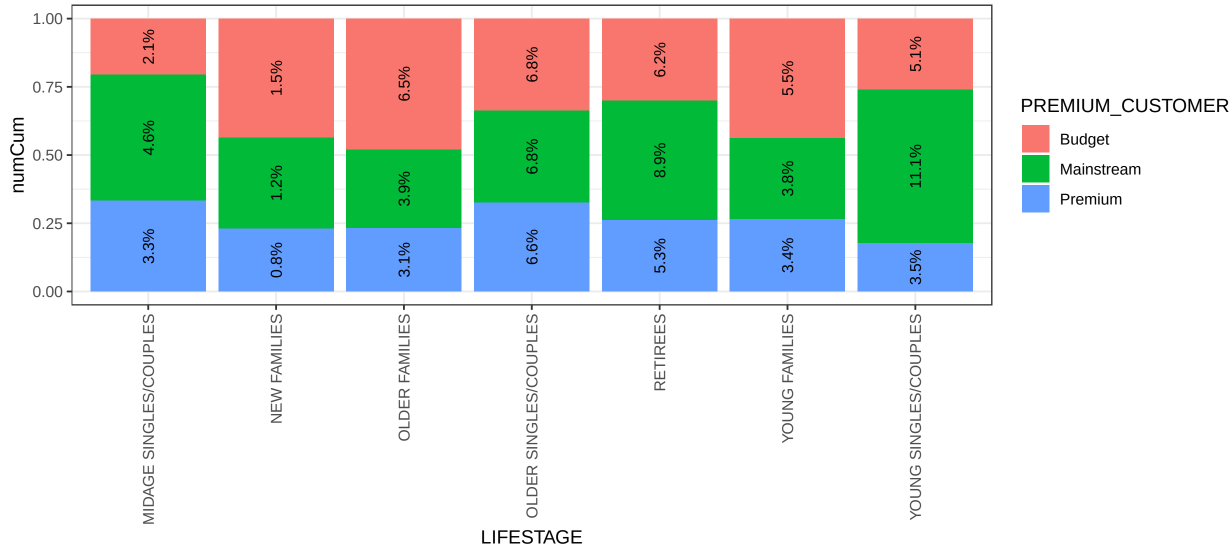


Sales are coming mainly from Budget - older families, Mainstream - young singles/couples, and Mainstream - retirees Let's see if the higher sales are due to there being more customers who buy chips.

```
#### Number of customers by LIFESTAGE and PREMIUM_CUSTOMER
# Calculate the summary of number of customers by those dimensions and create a plot.
customers <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarize(numCum = n_distinct(LYLTY_CARD_NBR))
```

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

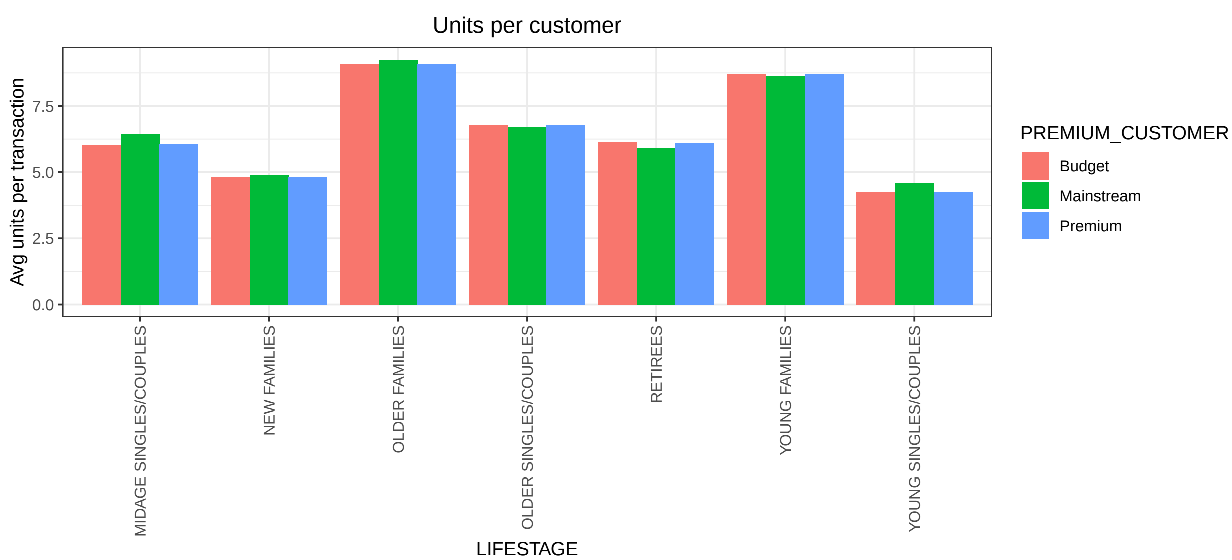
```
customers <- customers %>%
  mutate(percent = numCum / sum(customers$numCum) * 100)

p1 <- ggplot(customers, aes(fill=PREMIUM_CUSTOMER, y=numCum, x=LIFESTAGE)) +
    geom_bar(position="fill", stat="identity") +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
    geom_text(aes(label = paste0(round(percent, 1), "%"),
                  y = 0.5 * numCum),
              position = position_fill(vjust = 0.5),
              size = 3, color = "black",
              angle = 90,
              hjust = 0.5)

p1
```



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. Let's have a look at this next.

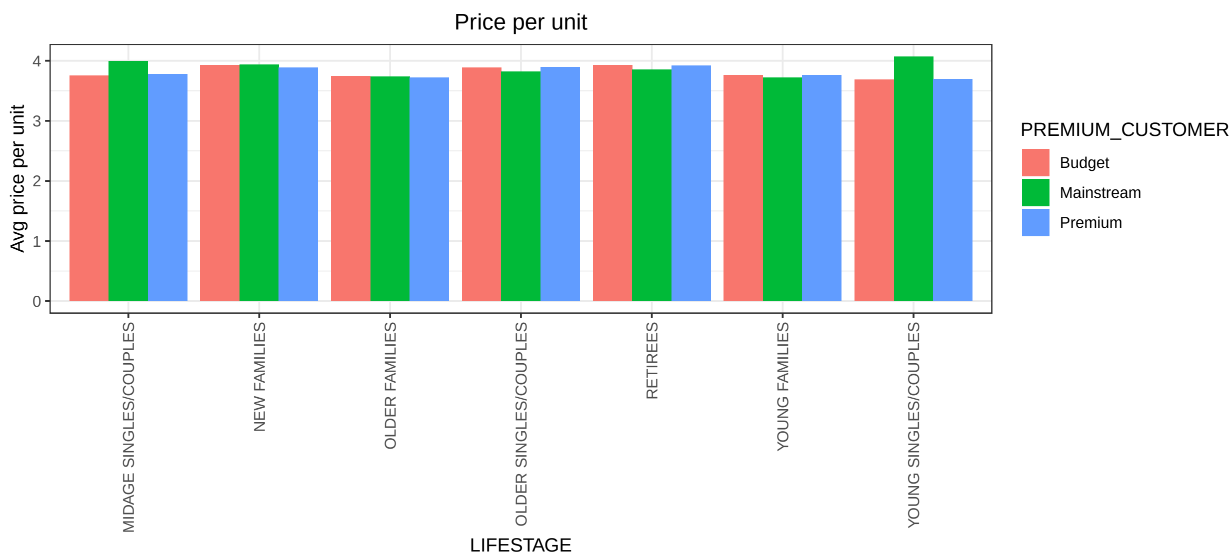
```
#### Average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER
# Calculate and plot the average number of units per customer by those two dimensions.
data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarize(avg_units = sum(PROD_QTY)/uniqueN(LYLTY_CARD_NBR)) %>%
  ggplot(aes(x=LIFESTAGE, y=avg_units, fill=PREMIUM_CUSTOMER)) +
  geom_bar(stat='identity', position= "dodge") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  labs(y = "Avg units per transaction", title = "Units per customer")
```



Older families and young families in general buy more chips per customer

Let's 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
# Calculate and plot the average price per unit sold (average sale price) by those two customer
↪  dimensions.
data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarize(avg_price = sum(TOT_SALES)/sum(PROD_QTY)) %>%
  ggplot(aes(x=LIFESTAGE, y=avg_price, fill=PREMIUM_CUSTOMER)) +
  geom_bar(stat='identity', position= "dodge") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  labs(y = "Avg price per unit", title = "Price per unit")
```



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 isn't 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

# Perform a t-test to see if the difference is significant.
data$avg_price_unit = data$TOT_SALES/data$PROD_QTY

# Mainstream midage singles/couples vs premium midage singles/couples
MSC = data %>%
  filter(
    PREMIUM_CUSTOMER == 'Mainstream',
```

```
    LIFESTAGE == c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES")
) %>%
  select(avg_price_unit)

PSC = data %>%
  filter(
    PREMIUM_CUSTOMER != 'Mainstream',
    LIFESTAGE == c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES")
) %>%
  select(avg_price_unit)

t.test(MSC, PSC, alternative = "greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  MSC and PSC
## t = 27.447, df = 27658, p-value < 2.2e-16
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.3221282       Inf
## sample estimates:
## mean of x mean of y
##  4.044382  3.701718
```

The t-test results in a p-value less than 0.5, i.e. the unit price for mainstream, young and mid-age singles and couples ARE significantly higher than that of budget or premium, young and midage singles and couples.

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. Let's look at Mainstream - young singles/couples. For instance, let's find out if they tend to buy a particular brand of chips.

```
#### Deep dive into Mainstream, young singles/couples
# Work out of there are brands that these two customer segments prefer more than others. You could
↪  use a technique called affinity analysis or a-prior analysis (or any other method if you
↪  prefer)

# data %>%
#   filter(PREMIUM_CUSTOMER == 'Mainstream') %>%
#   group_by(BRAND) %>%
#   summarise(count = n()) %>%
#   arrange(-count) %>%
#   ggplot(aes(x= reorder(BRAND, count), y=count)) +
#   xlab("Brand Name") + geom_col() + coord_flip()
#
# data %>%
#   filter(LIFESTAGE == 'YOUNG SINGLES/COUPLES') %>%
#   group_by(BRAND) %>%
```

```r
#   summarise(count = n()) %>%
#   arrange(-count) %>%
#   ggplot(aes(x= reorder(BRAND, count), y=count)) +
#   xlab("Brand Name") + geom_col() + coord_flip()
#
# data %>%
#   filter(PREMIUM_CUSTOMER == 'Mainstream', LIFESTAGE == 'YOUNG SINGLES/COUPLES' ) %>%
#   group_by(BRAND) %>%
#   summarise(count = n()) %>%
#   arrange(-count) %>%
#   ggplot(aes(x= reorder(BRAND, count), y=count)) +
#   xlab("Brand Name") + geom_col() + coord_flip()

#### Deep dive into Mainstream, young singles/couples
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
```

We can see that : • Mainstream young singles/couples are 23% more likely to purchase Tyrrells chips compared to the rest of the population • Mainstream young singles/couples are 56% less likely to purchase Burger Rings compared to the rest of the population

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
# Do the same for pack size.
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
```

It looks like Mainstream young singles/couples are 27% more likely to purchase a 270g pack of chips compared to the rest of the population but let's dive into what brands sell this pack size.

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

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

Twisties are the only brand offering 270g packs and so this may instead be reflecting a higher likelihood of purchasing Twisties.

Conclusion

Let's recap what we've found!

Sales have mainly been due to Budget - older families, Mainstream - young singles/couples, and Mainstream - retirees shoppers. We found that the high spend in chips for mainstream young singles/couples and retirees is due to there being more of them than other buyers. Mainstream, midage and young singles and couples are also more likely to pay more per packet of chips. This is indicative of impulse buying behaviour. We've also found that Mainstream young singles and couples 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 visibilty and impulse behaviour.

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.