# Quantium Virtual Internship - Retail Strategy and Analytics - Task 1

Sarita

2024-05-13

## Load required libraries

```
library(data.table)
library(ggplot2)
library(ggmosaic)
library(readr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v stringr   1.5.1
## v forcats   1.0.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()     masks data.table::between()
## x dplyr::filter()      masks stats::filter()
## x dplyr::first()       masks data.table::first()
## x lubridate::hour()    masks data.table::hour()
## x lubridate::isoweek() masks data.table::isoweek()
## x dplyr::lag()         masks stats::lag()
## x dplyr::last()        masks data.table::last()
## x lubridate::mday()    masks data.table::mday()
## x lubridate::minute()  masks data.table::minute()
## x lubridate::month()   masks data.table::month()
## x lubridate::quarter() masks data.table::quarter()
## x lubridate::second()  masks data.table::second()
## x purrr::transpose()   masks data.table::transpose()
## x lubridate::wday()    masks data.table::wday()
## x lubridate::week()    masks data.table::week()
## x lubridate::yday()    masks data.table::yday()
## x lubridate::year()    masks data.table::year()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

## import csv

```
filePath <- ""
transactionData <- fread(paste0(filePath,"QVI_transaction_data.csv"))
customerData <- fread(paste0(filePath,"QVI_purchase_behaviour.csv"))
```

```r
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
head(customerData)
```

```
##    LYLTY_CARD_NBR              LIFESTAGE PREMIUM_CUSTOMER
##             <int>                 <char>           <char>
## 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
```

# Exploratory data analysis

## TransactionData

```r
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
transactionData %>%
  summarise_all(class) %>%
  gather(variale, class)
```

```
##          variale     class
## 1           DATE   integer
## 2      STORE_NBR   integer
## 3 LYLTY_CARD_NBR   integer
## 4         TXN_ID   integer
## 5       PROD_NBR   integer
## 6      PROD_NAME character
## 7       PROD_QTY   integer
## 8      TOT_SALES   numeric
```

### CustomerData

```
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>
```

```
customerData %>%
  summarise_all(class) %>%
  gather(variable, class)
```

```
##           variable     class
## 1   LYLTY_CARD_NBR   integer
## 2        LIFESTAGE character
## 3 PREMIUM_CUSTOMER character
```

## Examining TransactionData data

From exploring the dataset I found that the date column is in an integer format, so the first step is changing the date format.

### Converting Date Format

```
transactionData$DATE <- as.Date(transactionData$DATE, origin = "1899-12-30")
head(transactionData)
```

```
##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##        <Date>     <int>          <int>  <int>    <int>
## 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
##                                    <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
```

### Examining PROD_NAME

```
transactionData %>% count(PROD_NAME)
```

```
##                                     PROD_NAME     n
##                                        <char> <int>
##   1:                        Burger Rings 220g  1564
##   2:                   CCs Nacho Cheese    175g  1498
##   3:                        CCs Original 175g  1514
##   4:                   CCs Tasty Cheese    175g  1539
##   5:             Cheetos Chs & Bacon Balls 190g  1479
##  ---
## 110: WW Sour Cream &OnionStacked Chips 160g  1483
## 111:    WW Supreme Cheese   Corn Chips 200g  1509
## 112:         Woolworths Cheese   Rings 190g  1516
## 113:         Woolworths Medium   Salsa 300g  1430
## 114:         Woolworths Mild     Salsa 300g  1491
```

There are 114 types of product but we are only interested in the potato chips, so we would like to keep only the data of potato ships and discard other by summarising the individual words in the product name.

```
productWords <- data.table(unlist(strsplit(unique(transactionData$PROD_NAME), " ")))
setnames(productWords, "words")
```

### Removing digits

```
productWords <- productWords[grepl("[[:digit:]]", words) == FALSE, ]
```

### Removing special characters

```
productWords <- productWords[grepl("[[:punct:]]", words) == FALSE, ]
```

### Sorting by frequency

```
productWords[, .N, words][order(N, decreasing = TRUE)]
```

```
##          words     N
##         <char> <int>
##   1:             234
##   2:     Chips    21
##   3:    Smiths    16
##   4:   Crinkle    14
##   5:       Cut    14
##  ---
## 165:       Rst     1
## 166:      Pork     1
## 167:     Belly     1
## 168:        Pc     1
```

```
## 169: Bolognese     1
```

## Remove the salsa product

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

## Summarizing the 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 : 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 case where 200 packets of chips are bought in one transaction.

## Filter the dataset to find the outlier

```
transactionData[PROD_QTY == 200, ]
```

```
##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##        <Date>     <int>          <int>  <int>    <int>
## 1: 2018-08-19       226         226000 226201        4
## 2: 2019-05-20       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.

### Examining other transactions that customer made

```
transactionData[LYLTY_CARD_NBR == 226000, ]
```

```
##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##        <Date>     <int>          <int>  <int>    <int>
## 1: 2018-08-19       226         226000 226201        4
## 2: 2019-05-20       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
```

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.

#### Filter out the outlier

```
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
```

Now, let's examine the number of transaction lines over time to find obvious data issues such as missing data.

### Finding obvious data issue

#### Counting the number of transactions by date

```
transactionData[, .N, by = DATE]
```

```
##            DATE     N
##          <Date> <int>
##   1: 2018-10-17   682
##   2: 2019-05-14   705
##   3: 2019-05-20   707
##   4: 2018-08-17   663
##   5: 2018-08-18   683
##  ---
## 360: 2018-12-08   622
## 361: 2019-01-30   689
## 362: 2019-02-09   671
## 363: 2018-08-31   658
## 364: 2019-02-12   684
```

There's only 364 rows, meaning only 364 dates which indicates a missing date. Next, I will 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.

**Filling the missing day**

Creating a sequence of dates and join this the count of transactions by date to fill in the missing day

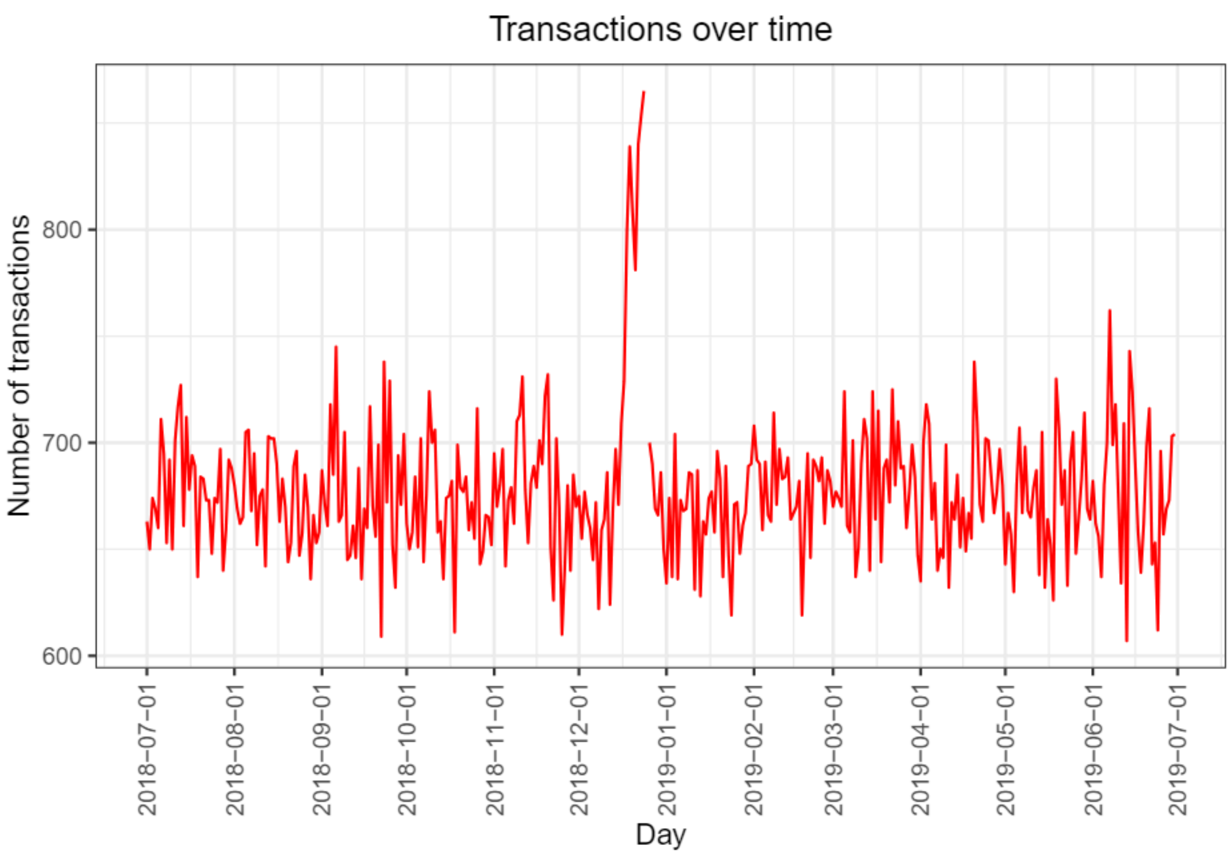
```
allDates <- data.table(seq(as.Date("2018/07/01"), as.Date("2019/06/30"), by = "day"))
setnames(allDates, "DATE")
transactions_by_day <- merge(allDates, transactionData[, .N, by = DATE], all.x = TRUE)
```

**Setting plot themes to format graphs**

```
theme_set(theme_bw())
theme_update(plot.title = element_text(hjust = 0.5))
```

**Ploting transactions over time**

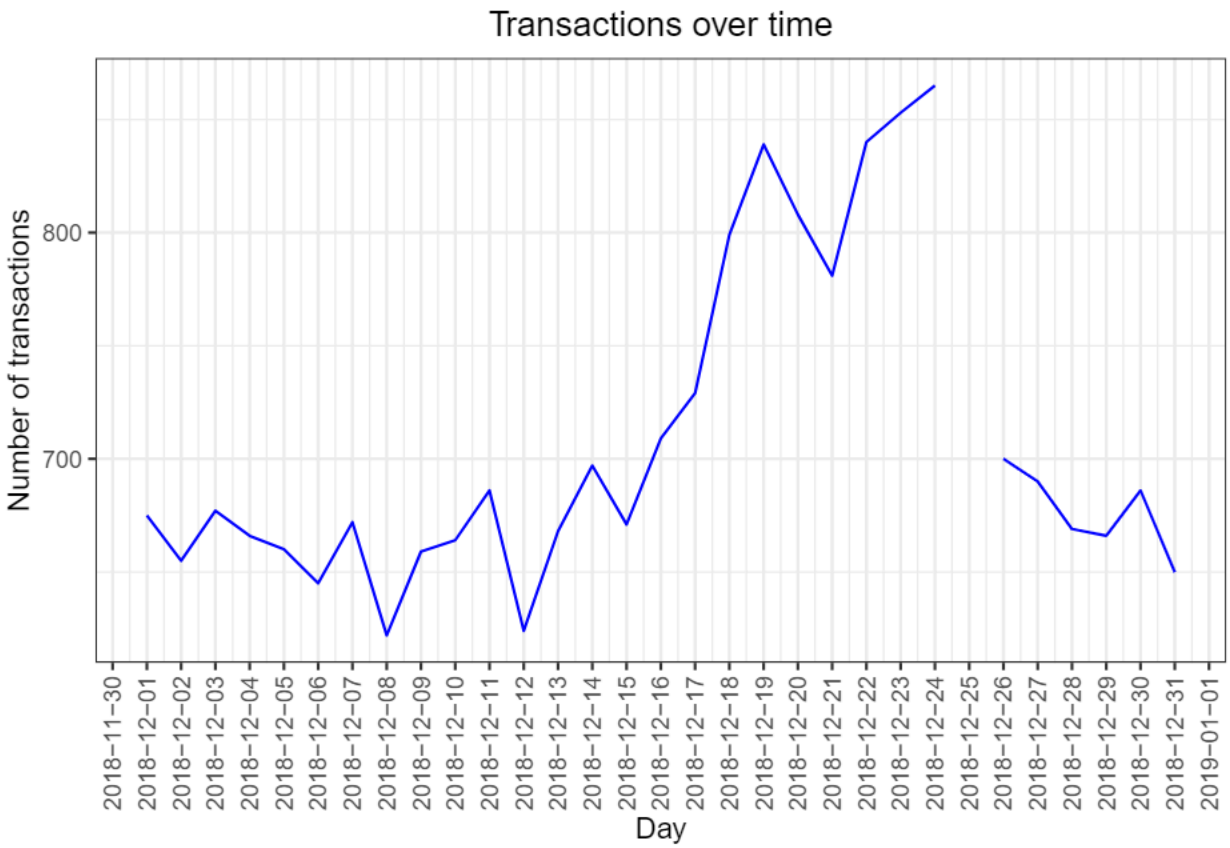
```
ggplot(transactions_by_day, aes(x = DATE, y = N)) +
 geom_line(col = "red") +
 labs(x = "Day", y = "Number of transactions", title = "Transactions over time") +
 scale_x_date(breaks = "1 month") +
 theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```




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

**Filtering to December and look at individual days**

```
ggplot(transactions_by_day[month(DATE)==12, ], aes(x = DATE, y = N)) +
 geom_line(col = "blue") +
 labs(x = "Day", y = "Number of transactions", title = "Transactions over time") +
 scale_x_date(breaks = "1 day") +
 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, the data no longer has outliers and any other missing value. So we can move on to creating other features. I will start with pack size from PROD_NAME.

## Examining pack size

```
transactionData[, PACK_SIZE := parse_number(PROD_NAME)]
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
```

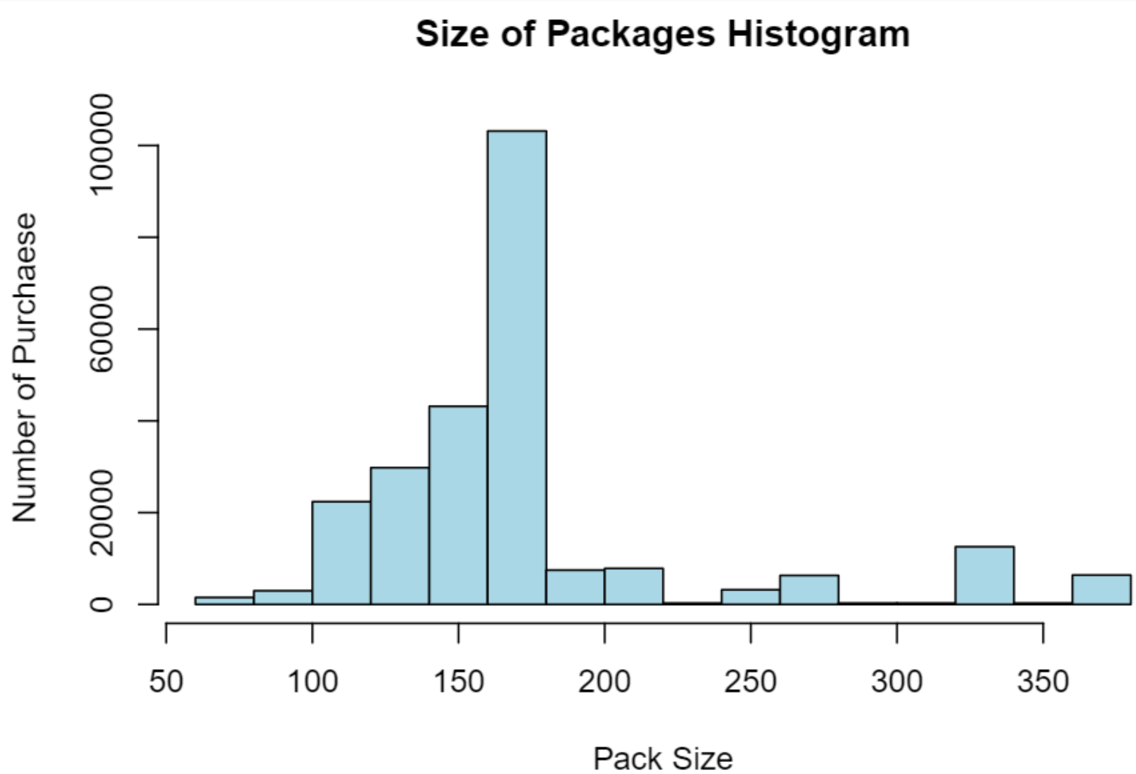
```
##  7:       150 40203
##  8:       160  2970
##  9:       165 15297
## 10:       170 19983
## 11:       175 66390
## 12:       180  1468
## 13:       190  2995
## 14:       200  4473
## 15:       210  6272
## 16:       220  1564
## 17:       250  3169
## 18:       270  6285
## 19:       330 12540
## 20:       380  6416
##     PACK_SIZE     N
```

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

Next, we will plot a histogram of PACK_SIZE since we know that it's a categorical variable and not a continuous variable even though it's numeric.

### Ploting Histogram of PACK_SIZE

```r
options(scipen = 999)
hist(transactionData$PACK_SIZE,
     col = "lightblue",
     main = "Size of Packages Histogram",
     xlab = "Pack Size",
     ylab = "Number of Purchaese")
```



The histogram looks reasonable. We found that the packs of size 150-200 was purchased the most.

## Brand

**To creating BRAND, we will use the first word in PROD_NAME to work out the brand name.**

```
transactionData[, BRAND := word(transactionData[, PROD_NAME], 1)]
```

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.

### Checking brands

```
transactionData[, .N, by = BRAND][order(-N)]
```

```
##          BRAND     N
##         <char> <int>
##  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
```

### Cleaning BRAND

```
transactionData[toupper(BRAND) == "RED", BRAND := "RRD"]
transactionData[BRAND == "Dorito", BRAND := "Doritos"]
transactionData[BRAND == "Smith", BRAND := "Smiths"]
```

```r
transactionData[BRAND == "WW", BRAND := "Woolworths"]
transactionData[BRAND == "Grain", BRAND := "GrnWves"]
transactionData[BRAND == "Snbts", BRAND := "Sunbites"]
transactionData[BRAND == "Infzns", BRAND := "Infuzions"]
transactionData[BRAND == "NCC", BRAND := "Natural"]
```

Rechecking brands

```r
transactionData[, .N, by = BRAND][order(-N)]
```

```
##          BRAND     N
##         <char> <int>
##  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
##          BRAND     N
```

Finally, we have 20 different brands since 8 of our rows that had similar brands have been merged.

# Examining customer data

```r
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
```

We can see that the loyalty card number is a numeric vector while lifestage and premium_customer are character vectors. Next, I will join the transaction and customer data sets together.

### Merge data sets

```
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.

### checking for nulls

```
colSums(is.na(data))
```

```
##    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
```

There are no nulls. So all our customers in transaction data has been accounted for in the customer dataset.

### Writing data as a csv

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

## Data analysis on customer segment

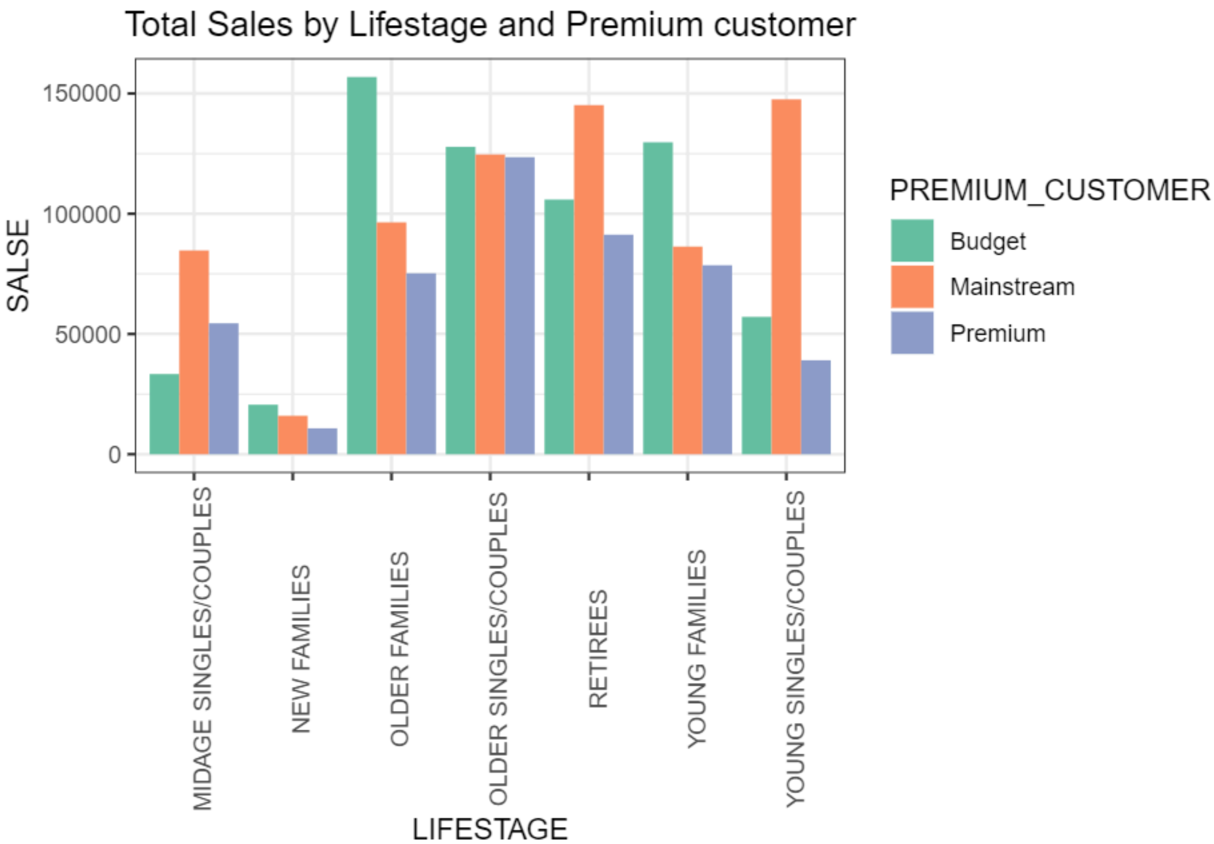
### 1.Total Sales by LIFESTAGE and PREMIUM_CUSTOMER

```
total_sales <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(SALSE = sum(TOT_SALES), .groups = "keep")
total_sales
```

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE, PREMIUM_CUSTOMER [21]
##    LIFESTAGE              PREMIUM_CUSTOMER   SALSE
##    <chr>                  <chr>              <dbl>
##  1 MIDAGE SINGLES/COUPLES Budget            33346.
##  2 MIDAGE SINGLES/COUPLES Mainstream        84734.
##  3 MIDAGE SINGLES/COUPLES Premium           54444.
##  4 NEW FAMILIES           Budget            20607.
##  5 NEW FAMILIES           Mainstream        15980.
##  6 NEW FAMILIES           Premium           10761.
##  7 OLDER FAMILIES         Budget           156864.
##  8 OLDER FAMILIES         Mainstream        96414.
##  9 OLDER FAMILIES         Premium           75243.
## 10 OLDER SINGLES/COUPLES  Budget           127834.
## # i 11 more rows
```

Plot for salse

```r
ggplot(total_sales, aes(LIFESTAGE, SALSE, fill=PREMIUM_CUSTOMER)) +
  geom_bar(stat="identity", position=position_dodge()) +
    theme(axis.text.x = element_text(angle = 90)) +
  labs(title="Total Sales by Lifestage and Premium customer") +
    scale_fill_brewer(palette = "Set2")
```




Sales are coming mainly from Budget - older families, Mainstream - young singles/couples, and Mainstream - retirees

## 2.Number of customers by LIFESTAGE and PREMIUM_CUSTOMER

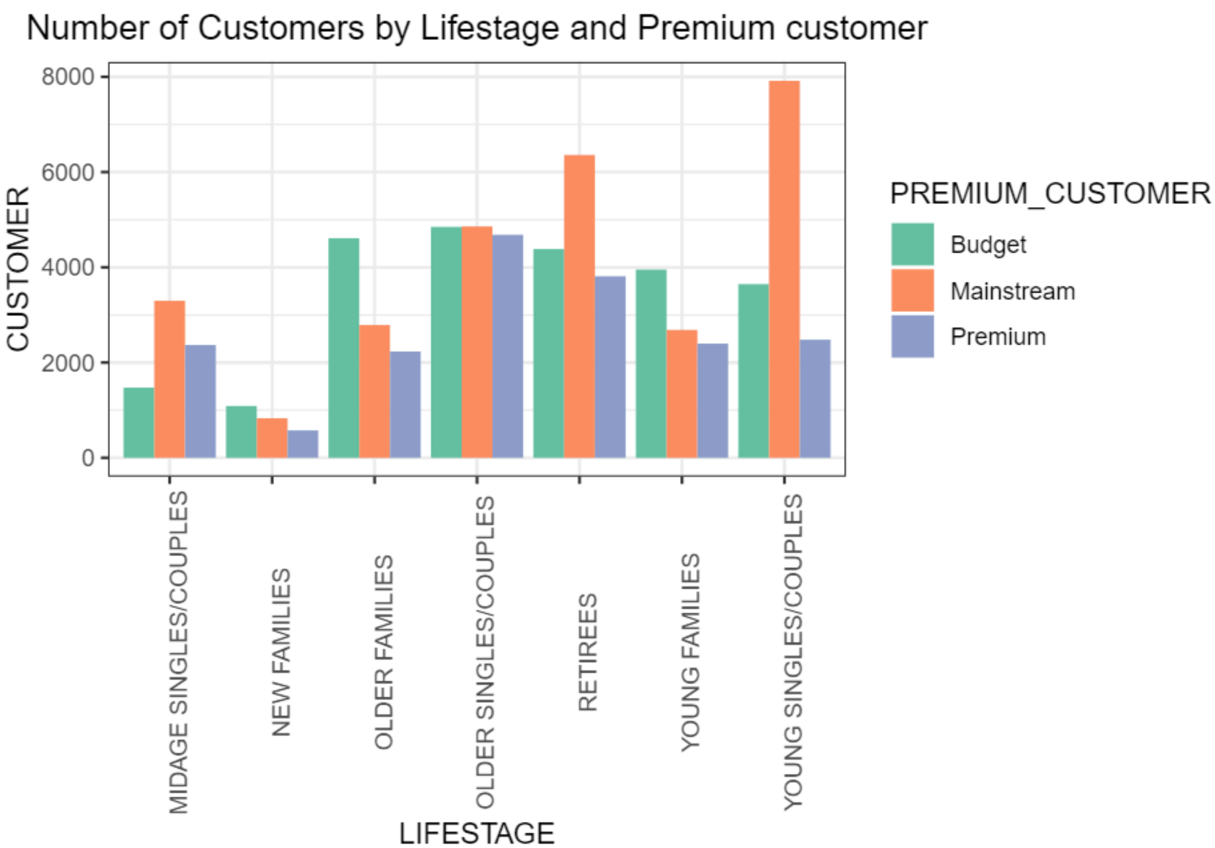
```r
customer <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(CUSTOMER = uniqueN(LYLTY_CARD_NBR), .groups = "keep")
customer
```

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE, PREMIUM_CUSTOMER [21]
##    LIFESTAGE              PREMIUM_CUSTOMER CUSTOMER
##    <chr>                  <chr>               <int>
##  1 MIDAGE SINGLES/COUPLES Budget               1474
##  2 MIDAGE SINGLES/COUPLES Mainstream           3298
##  3 MIDAGE SINGLES/COUPLES Premium              2369
##  4 NEW FAMILIES           Budget               1087
##  5 NEW FAMILIES           Mainstream            830
```

```
##   6 NEW FAMILIES          Premium             575
##   7 OLDER FAMILIES        Budget             4611
##   8 OLDER FAMILIES        Mainstream         2788
##   9 OLDER FAMILIES        Premium            2231
##  10 OLDER SINGLES/COUPLES Budget             4849
## # i 11 more rows
```

### Plot for number of customers

```r
ggplot(customer, aes(LIFESTAGE, CUSTOMER, fill=PREMIUM_CUSTOMER)) +
  geom_bar(stat="identity", position=position_dodge()) +
    theme(axis.text.x = element_text(angle = 90)) +
  labs(title="Number of Customers by Lifestage and Premium customer") +
    scale_fill_brewer(palette = "Set2")
```




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 average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER.

### 3.Average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER

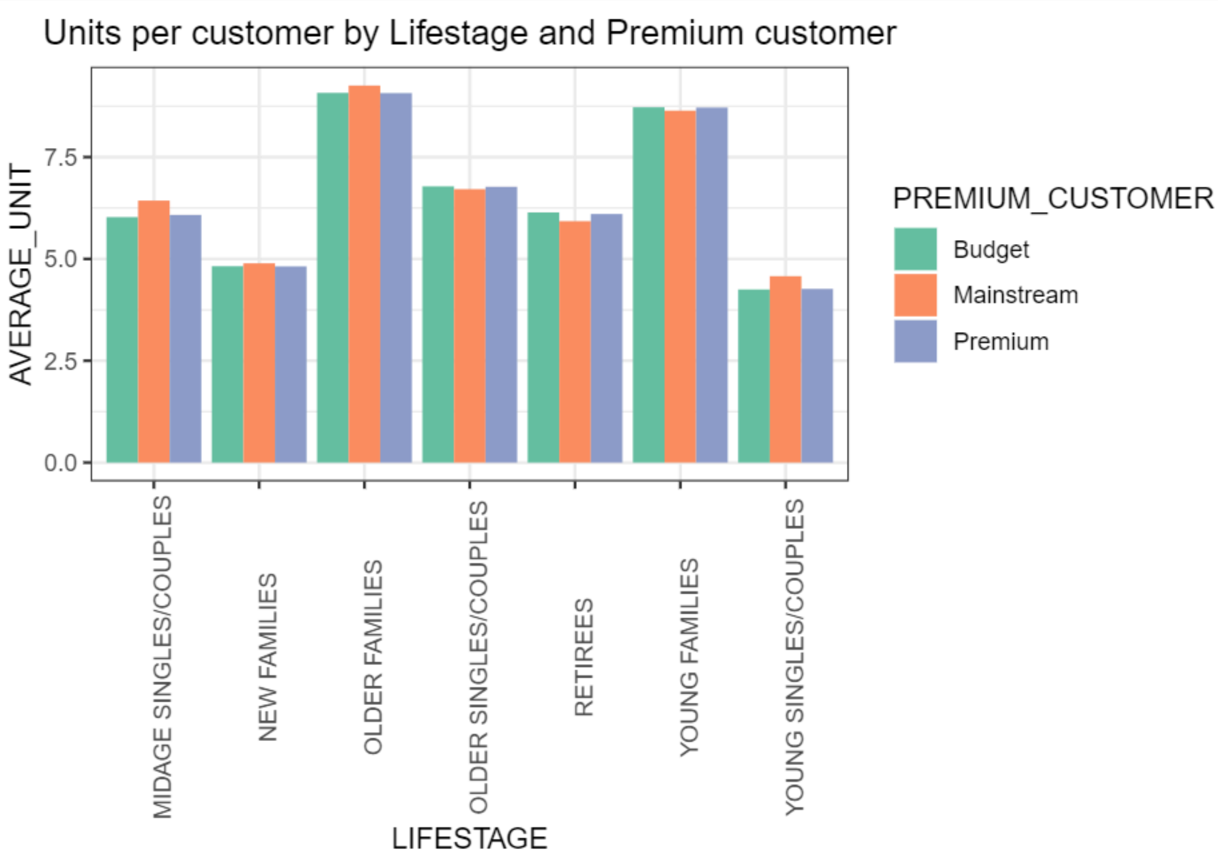
```r
avg_unit <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
```

```
  summarise(AVERAGE_UNIT = sum(PROD_QTY)/uniqueN(LYLTY_CARD_NBR), .groups = "keep")
avg_unit
```

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE, PREMIUM_CUSTOMER [21]
##    LIFESTAGE              PREMIUM_CUSTOMER AVERAGE_UNIT
##    <chr>                  <chr>                   <dbl>
##  1 MIDAGE SINGLES/COUPLES Budget                   6.03
##  2 MIDAGE SINGLES/COUPLES Mainstream               6.43
##  3 MIDAGE SINGLES/COUPLES Premium                  6.08
##  4 NEW FAMILIES           Budget                   4.82
##  5 NEW FAMILIES           Mainstream               4.89
##  6 NEW FAMILIES           Premium                  4.82
##  7 OLDER FAMILIES         Budget                   9.08
##  8 OLDER FAMILIES         Mainstream               9.26
##  9 OLDER FAMILIES         Premium                  9.07
## 10 OLDER SINGLES/COUPLES  Budget                   6.78
## # i 11 more rows
```

Plot for average number of units per customer

```
ggplot(avg_unit, aes(LIFESTAGE, AVERAGE_UNIT, fill=PREMIUM_CUSTOMER)) +
  geom_bar(stat="identity", position=position_dodge()) +
    theme(axis.text.x = element_text(angle = 90)) +
  labs(title="Units per customer by Lifestage and Premium customer") +
    scale_fill_brewer(palette = "Set2")
```



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.

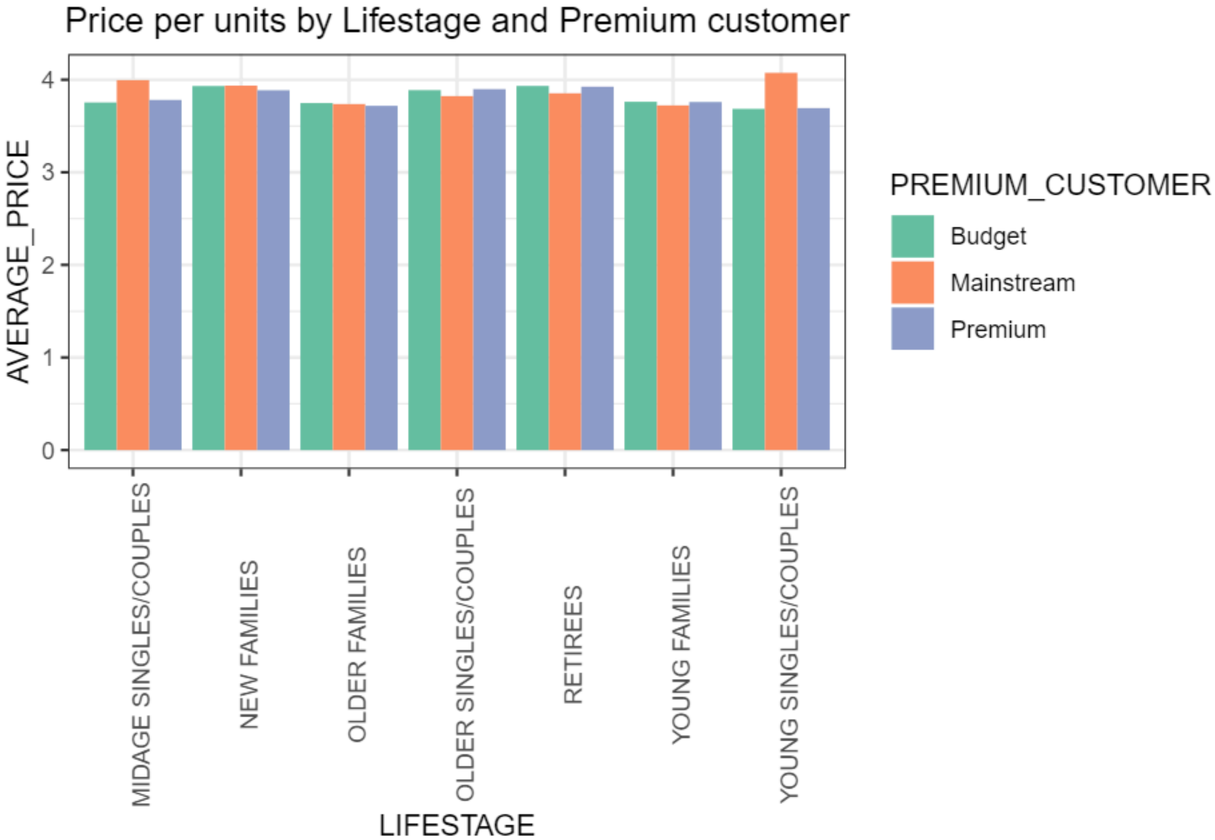
## 4.Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER

```
avg_price <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(AVERAGE_PRICE = sum(TOT_SALES)/sum(PROD_QTY), .groups = "keep")
avg_price
```

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE, PREMIUM_CUSTOMER [21]
##    LIFESTAGE              PREMIUM_CUSTOMER AVERAGE_PRICE
##    <chr>                  <chr>                    <dbl>
##  1 MIDAGE SINGLES/COUPLES Budget                    3.75
##  2 MIDAGE SINGLES/COUPLES Mainstream                3.99
##  3 MIDAGE SINGLES/COUPLES Premium                   3.78
##  4 NEW FAMILIES           Budget                    3.93
##  5 NEW FAMILIES           Mainstream                3.94
##  6 NEW FAMILIES           Premium                   3.89
##  7 OLDER FAMILIES         Budget                    3.75
##  8 OLDER FAMILIES         Mainstream                3.74
##  9 OLDER FAMILIES         Premium                   3.72
## 10 OLDER SINGLES/COUPLES  Budget                    3.89
## # i 11 more rows
```

### Plot for average price per unit

```
ggplot(avg_price, aes(LIFESTAGE, AVERAGE_PRICE, fill=PREMIUM_CUSTOMER)) +
  geom_bar(stat="identity", position=position_dodge()) +
    theme(axis.text.x = element_text(angle = 90)) +
  labs(title="Price per units by Lifestage and Premium customer") +
    scale_fill_brewer(palette = "Set2")
```




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.

## Conclusion

The data reveals valuable insights into customer segments, preferred brands, pack sizes, and spending trends.

- The purchasers of chips are coming mainly from Budget - older families, Mainstream - young singles/couples, and Mainstream - retirees shoppers. Among these groups, Mainstream - young singles/couples and retirees shoppers stand out for their high chip purchases due to their highest population.
- Kettle chips suggest an opportunity to capitalize on this by enhancing product visibility to attract more customers from this segment.
- The consistent preference for the 175-gram chip size followed by the 150-gram size across all customer segments indicates a strong market demand for these particular sizes.
- Sales peak just before Christmas, indicating a significant opportunity for increased revenue during this period. It's crucial to ensure sufficient stock levels to meet the heightened demand before Christmas, optimizing sales potential during this critical period.