Quantium Virtual Internship - Retail Strategy & Analytics - Task 1

Rounak Saha

2024-11-26

Load required libraries and datasets

```r
options(repos = c(CRAN = "https://cloud.r-project.org"))
install.packages("tidyverse")
install.packages("readr")
install.packages("readxl")
install.packages("ggplot2")
install.packages("tidyr")
install.packages("ggmosaic")
install.packages("data.table")
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.5.1     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::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(readxl)
library(readr)
library(ggplot2)
library(tidyr)
library(ggmosaic)
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
# Load the datasets
transactionData <- read_xlsx("QVI_transaction_data.xlsx")
customerData <- fread("QVI_purchase_behaviour.csv")
transactionData <- as.data.table(transactionData)
```

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. We can also run in the console to see a sample of the data or use `head(transactionData)` to look at the first 10 rows.

```r
head(transactionData,10)
```

```
##      DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##     <num>     <num>          <num>  <num>    <num>
##  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
##  7: 43601         4           4149   3333       16
##  8: 43601         4           4196   3539       24
##  9: 43332         5           5026   4525       42
## 10: 43330         7           7150   6900       52
##                                     PROD_NAME PROD_QTY TOT_SALES
##                                        <char>    <num>     <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
##  7: Smiths Crinkle Chips Salt & Vinegar 330g        1       5.7
##  8:    Grain Waves         Sweet Chilli 210g        1       3.6
##  9:  Doritos Corn Chip Mexican Jalapeno 150g        1       3.9
## 10:    Grain Waves Sour    Cream&Chives 210G        2       7.2
```

```r
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 C
##  $ 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 numeric 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")
```

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
prodWords <- data.table(unlist(strsplit(unique(transactionData[, PROD_NAME]), " ")))
setnames(prodWords, 'names')
```

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
prodWords <- prodWords[grepl("\\d", names) ==FALSE, ]
prodWords <- prodWords[grepl("[:alpha:]", names), ]
```

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

```r
prodWords[, .N, names][order(-N)]
```

```
##             names     N
##            <char> <int>
##   1:        Chips    21
##   2:       Smiths    16
##   3:      Crinkle    14
##   4:       Kettle    13
##   5:       Cheese    12
##  ---
## 127: Chikn&Garlic     1
## 128:        Aioli     1
## 129:         Slow     1
## 130:        Belly     1
## 131:    Bolognese     1
```

There are salsa and dip products in the dataset but we are only interested in the chips category, so let's remove these.

```
transactionData <- transactionData[!grepl("salsa|dip", tolower(PROD_NAME)), ]
```

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).

```
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.:  70014   1st Qu.:  67559  
##  Median :2018-12-30   Median :130.0   Median : 130368   Median : 135186  
##  Mean   :2018-12-30   Mean   :135.1   Mean   : 135535   Mean   : 135134  
##  3rd Qu.:2019-03-31   3rd Qu.:203.0   3rd Qu.: 203086   3rd Qu.: 202666  
##  Max.   :2019-06-30   Max.   :272.0   Max.   :2373711   Max.   :2415841  
##     PROD_NBR       PROD_NAME            PROD_QTY         TOT_SALES      
##  Min.   :  1.00   Length:245304      Min.   :  1.000   Min.   :  1.700  
##  1st Qu.: 26.00   Class :character   1st Qu.:  2.000   1st Qu.:  5.800  
##  Median : 52.00   Mode  :character   Median :  2.000   Median :  7.400  
##  Mean   : 56.05                      Mean   :  1.908   Mean   :  7.335  
##  3rd Qu.: 86.00                      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.

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

```
##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##        <Date>     <num>          <num>  <num>    <num>
## 1: 2018-08-19       226         226000 226201        4
## 2: 2019-05-20       226         226000 226210        4
##                           PROD_NAME PROD_QTY TOT_SALES
##                              <char>    <num>     <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. 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
##        <Date>     <num>          <num>  <num>    <num>
## 1: 2018-08-19       226         226000 226201        4
## 2: 2019-05-20       226         226000 226210        4
##                           PROD_NAME PROD_QTY TOT_SALES
##                              <char>    <num>     <num>
## 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.

```
transactionData <- transactionData[!PROD_QTY== 200, ]
```

```
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.:  70014   1st Qu.:  67558
##  Median :2018-12-30   Median :130.0   Median : 130367   Median : 135186
##  Mean   :2018-12-30   Mean   :135.1   Mean   : 135534   Mean   : 135133
##  3rd Qu.:2019-03-31   3rd Qu.:203.0   3rd Qu.: 203086   3rd Qu.: 202665
##  Max.   :2019-06-30   Max.   :272.0   Max.   :2373711   Max.   :2415841
##     PROD_NBR       PROD_NAME            PROD_QTY       TOT_SALES
##  Min.   :  1.00   Length:245302      Min.   :1.000   Min.   : 1.70
##  1st Qu.: 26.00   Class :character   1st Qu.:2.000   1st Qu.: 5.80
##  Median : 52.00   Mode  :character   Median :2.000   Median : 7.40
##  Mean   : 56.06                      Mean   :1.907   Mean   : 7.33
##  3rd Qu.: 86.00                      3rd Qu.:2.000   3rd Qu.: 8.80
##  Max.   :114.00                      Max.   :5.000   Max.   :29.50
```

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

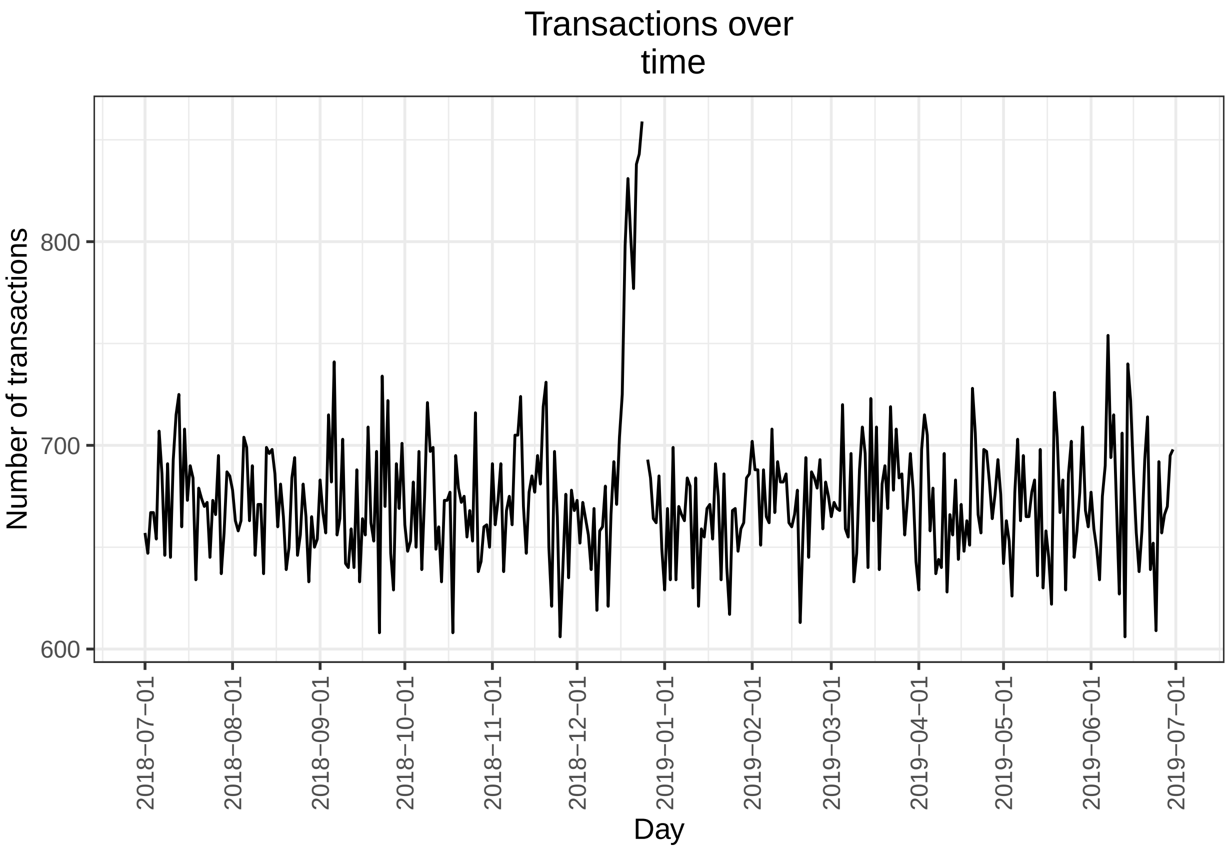
```
transactionData[, .N, by= DATE ][order(DATE)]
```

```
##            DATE     N
##          <Date> <int>
##   1: 2018-07-01   657
##   2: 2018-07-02   647
##   3: 2018-07-03   667
##   4: 2018-07-04   667
##   5: 2018-07-05   654
##  ---
## 360: 2019-06-26   657
## 361: 2019-06-27   666
## 362: 2019-06-28   670
## 363: 2019-06-29   695
## 364: 2019-06-30   698
```

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.

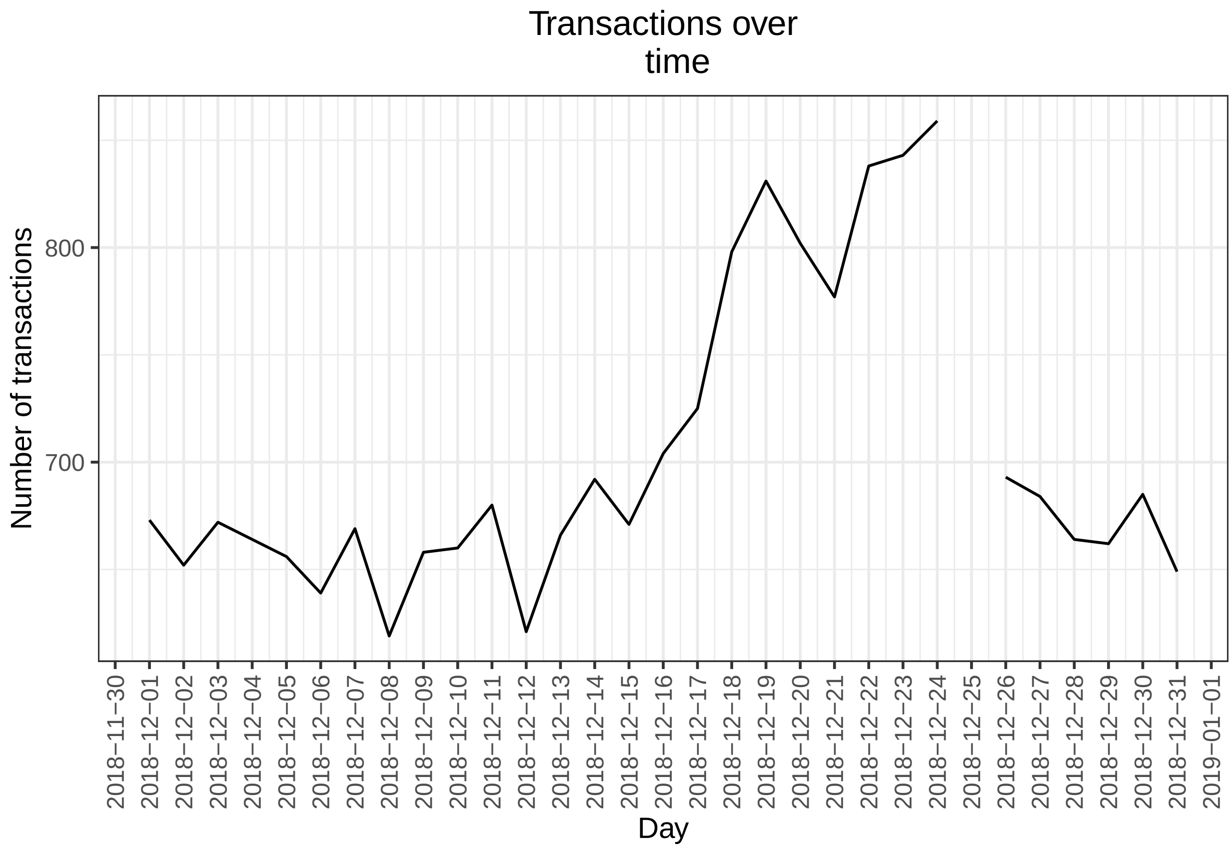
```
allDays <- as.data.table(seq(min(transactionData$DATE), max(transactionData$DATE), by = 'day'))
setnames(allDays, 'DATE')
allDayTransaction <- merge(allDays, transactionData[, .N, by = DATE], all.x = TRUE)

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

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



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

```r
ggplot(allDayTransaction[month(DATE) == 12, ], aes(x = DATE, y = N)) + geom_line() +
  labs(x = "Day", y = "Number of transactions", title = "Transactions over
   time") + theme(axis.text.x = element_text(angle = 90, vjust = 0.5)) + scale_x_date(breaks = "day")
```



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.

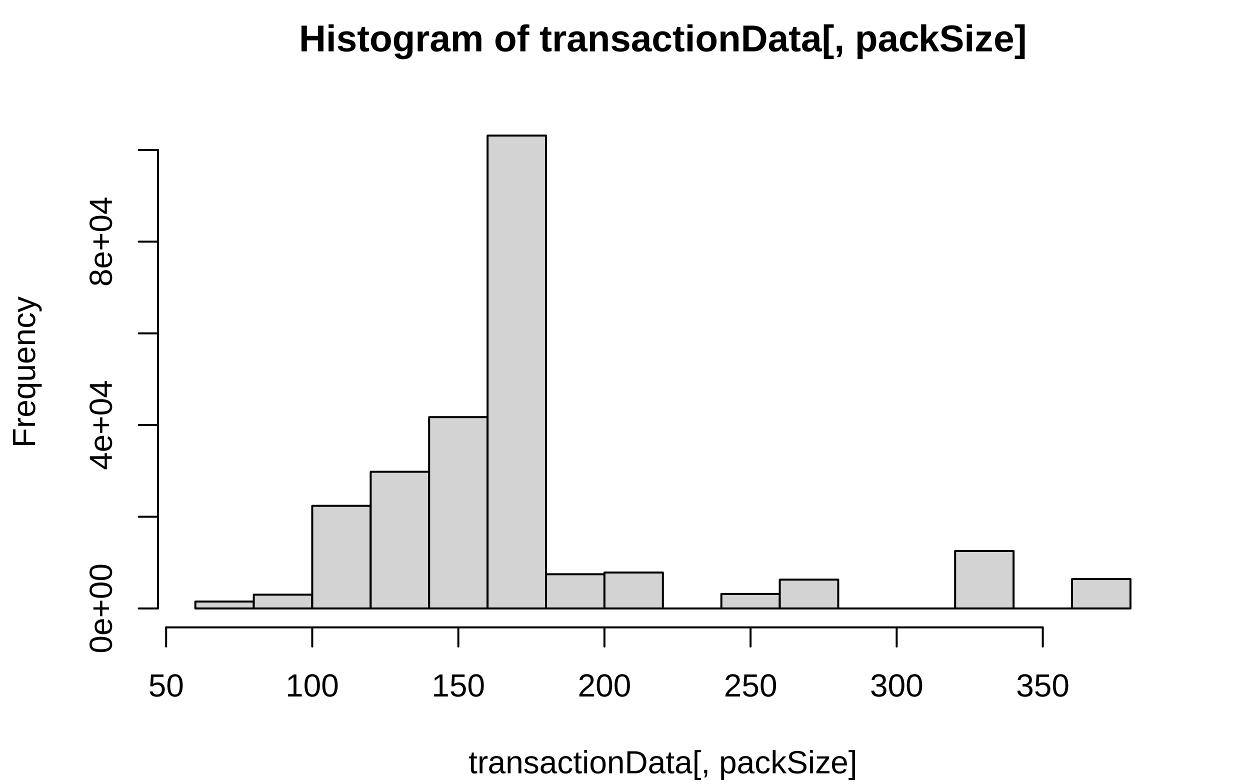
```
#Create Pack Size
transactionData[, packSize := parse_number(PROD_NAME)]

#Let's check if the pack sizes look sensible
transactionData[, .N, by= packSize][order(-packSize)]
```

```
##     packSize     N
##        <num> <int>
##  1:      380  6416
##  2:      330 12540
##  3:      270  6285
##  4:      250  3169
##  5:      220  1564
##  6:      210  6272
##  7:      200  4473
##  8:      190  2995
##  9:      180  1468
## 10:      175 66390
## 11:      170 19983
## 12:      165 15297
```

```
## 13:       160  2970
## 14:       150 38765
## 15:       135  3257
## 16:       134 25102
## 17:       125  1454
## 18:       110 22387
## 19:        90  3008
## 20:        70  1507
##      packSize     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.

```r
hist(transactionData[, packSize])
```



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

```r
# Extract the first word using tstrsplit
transactionData[, Brand := tstrsplit(PROD_NAME, " ")[[1]]]

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

```
##           Brand     N
##          <char> <int>
```

```
##  1:     Kettle 41288
##  2:     Smiths 25952
##  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
```

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, Smiths and Smith, Doritos and Dorito,Infuzions and Infzns, etc. Let's combine these together.

```
transactionData[, Brand := toupper(Brand)]
#Clean brand names
transactionData[Brand == "RED", Brand := "RRD"]
transactionData[Brand == "SNBTS", Brand := "SUNBITES"]
transactionData[Brand == "INFZNS", Brand := "INFUZIONS"]
transactionData[Brand == "WW", Brand := "WOOLWORTHS"]
transactionData[Brand == "SMITH", Brand := "SMITHS"]
transactionData[Brand == "NCC", Brand := "NATURAL"]
transactionData[Brand == "DORITO", Brand := "DORITOS"]
transactionData[Brand == "GRAIN", Brand := "GRNWVES"]
```

Check again!

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

```
##           Brand     N
##          <char> <int>
##  1:     KETTLE 41288
##  2:     SMITHS 28915
```

```
##  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
```

Examining Customer data

Now that we are happy with the transaction dataset, let's have a look at the customer dataset.We will do some basic summaries of the dataset, including distributions of any key columns.

```
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
```

```
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
```

Let's have a closer look at the LIFESTAGE and PREMIUM_CUSTOMER columns.

```
customerData[, .N, by= LIFESTAGE]
```

```
##                  LIFESTAGE     N
```

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

```
customerData[, .N, by= PREMIUM_CUSTOMER]
```

```
##    PREMIUM_CUSTOMER     N
##              <char> <int>
## 1:          Premium 18922
## 2:       Mainstream 29245
## 3:           Budget 24470
```

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

```
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.

```
sum(is.na(data$LIFESTAGE))
```

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

```
sum(is.na(data$PREMIUM_CUSTOMER))
```

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

Great, there are no nulls! So all our customers in the transaction data has been accounted for in the customer dataset.

Data analysis on customer segments

Now that the data is ready for analysis, we can define some metrics of interest to the client:

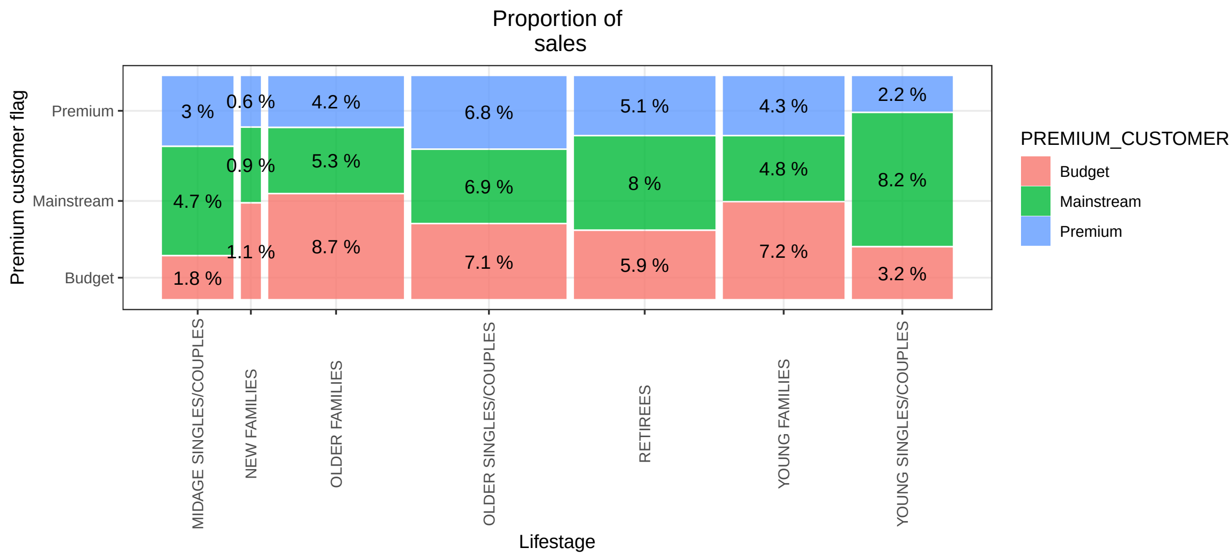
*Who spends the most on chips (total sales), describing customers by lifestage and how premium their general purchasing behaviour is

- 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

```r
#customers by lifestage and how premium their general purchasing behaviour
x <- data[, .(SALES = sum(TOT_SALES)),
 by = .(PREMIUM_CUSTOMER, LIFESTAGE)][order(-SALES)][,
 PROP_PERCENT := paste(round(SALES / sum(SALES) * 100, 2), "%")]

# Checking the top 10 purchasing behaviour
head(x, 10)
```

```
##     PREMIUM_CUSTOMER              LIFESTAGE     SALES PROP_PERCENT
##               <char>                 <char>     <num>       <char>
##  1:           Budget         OLDER FAMILIES 156096.75       8.68 %
##  2:       Mainstream YOUNG SINGLES/COUPLES 147244.20       8.19 %
##  3:       Mainstream               RETIREES 144677.55       8.05 %
##  4:           Budget         YOUNG FAMILIES 129151.15       7.18 %
##  5:           Budget OLDER SINGLES/COUPLES 127279.80       7.08 %
##  6:       Mainstream OLDER SINGLES/COUPLES 124089.50        6.9 %
##  7:          Premium OLDER SINGLES/COUPLES 123147.55       6.85 %
##  8:           Budget               RETIREES 105586.10       5.87 %
##  9:       Mainstream         OLDER FAMILIES  96059.95       5.34 %
## 10:          Premium               RETIREES  91013.25       5.06 %
```

```r
library(ggmosaic)
# Create plot
p <- ggplot(data = x) +
  geom_mosaic(aes(
    weight = SALES,
    x = product(PREMIUM_CUSTOMER, LIFESTAGE),
    fill = PREMIUM_CUSTOMER
  )) +
  labs(x = "Lifestage", y = "Premium customer flag", title = "Proportion of
 sales") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))

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



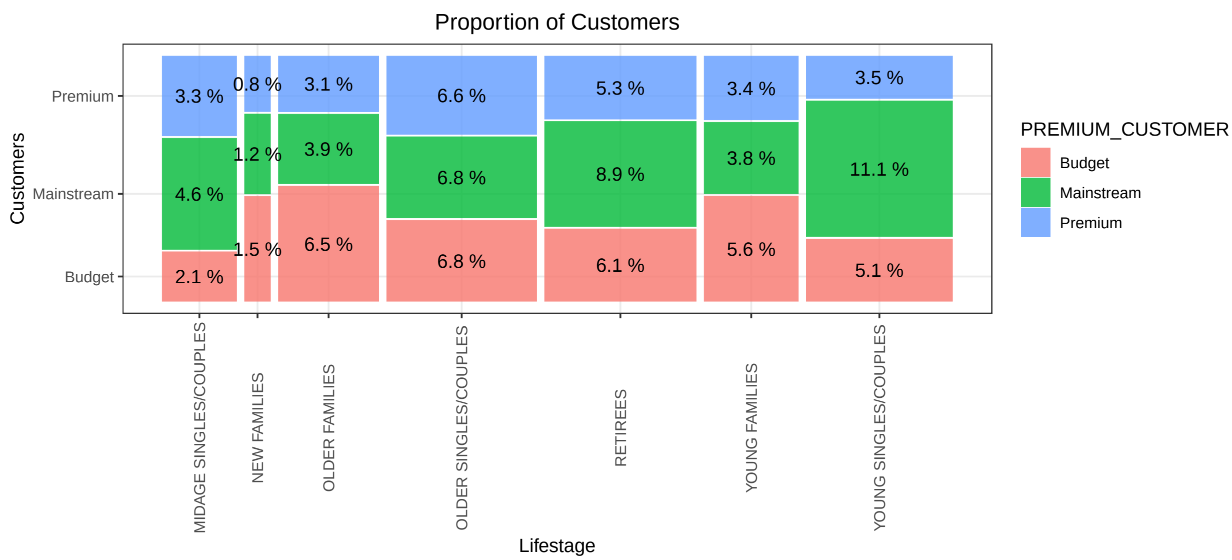
Sales are coming mainly from Budget - older families, Mainstream - young singles/couples, and Mainstream - retirees Let's see if the higher sales are due to there being more customers who buy chips.

```
x1 <- data[, .(CUSTOMER = uniqueN(LYLTY_CARD_NBR)),
 by = .(PREMIUM_CUSTOMER,
 LIFESTAGE)][order(-CUSTOMER)][,
 CUST_PERCENT := paste(round(CUSTOMER / sum(CUSTOMER) * 100, 2), "%")]

head(x1,12)
```

```
##     PREMIUM_CUSTOMER              LIFESTAGE CUSTOMER CUST_PERCENT
##               <char>                 <char>    <int>       <char>
##  1:       Mainstream  YOUNG SINGLES/COUPLES     7908      11.11 %
##  2:       Mainstream               RETIREES     6345       8.91 %
##  3:       Mainstream  OLDER SINGLES/COUPLES     4854       6.82 %
##  4:           Budget  OLDER SINGLES/COUPLES     4839        6.8 %
##  5:          Premium  OLDER SINGLES/COUPLES     4679       6.57 %
##  6:           Budget         OLDER FAMILIES     4606       6.47 %
##  7:           Budget               RETIREES     4376       6.15 %
##  8:           Budget         YOUNG FAMILIES     3951       5.55 %
##  9:          Premium               RETIREES     3808       5.35 %
## 10:           Budget  YOUNG SINGLES/COUPLES     3632        5.1 %
## 11:       Mainstream MIDAGE SINGLES/COUPLES     3296       4.63 %
## 12:       Mainstream         OLDER FAMILIES     2782       3.91 %
```

```
# Create plot
p1 <- ggplot(data = x1) +
  geom_mosaic(aes(
    weight = CUSTOMER,
    x = product(PREMIUM_CUSTOMER, LIFESTAGE),
    fill = PREMIUM_CUSTOMER
  )) +
  labs(x = 'Lifestage', y = 'Customers', title = 'Proportion of Customers') +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```

```
# Plot and label with proportion of sales
p1 + geom_text(data = ggplot_build(p1)$data[[1]], aes(
  x = (xmin + xmax) / 2 ,
  y =
    (ymin + ymax) / 2,
  label = as.character(paste(round(.wt / sum(
    .wt
  ), 3) * 100, '%'))
))
```



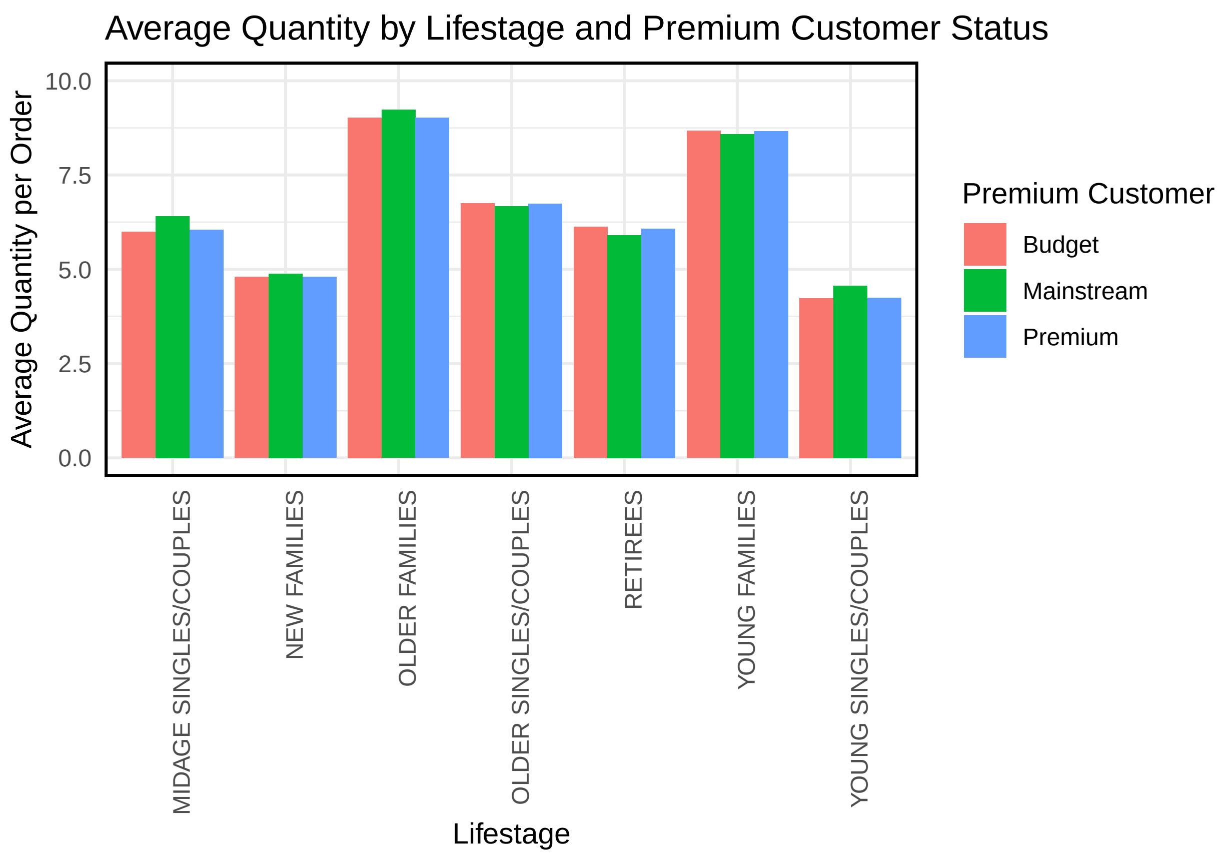
There are more Mainstream - young singles/couples and Mainstream - retirees who buy chips. This contributes to there being more sales to these customer segments but this is not a major driver for the Budget - Older families segment.

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 ordered per customer by LIFESTAGE and PREMIUM_CUSTOMER
x2 <- data[, .(AVG_QTY_ORDER = sum(PROD_QTY) / uniqueN(LYLTY_CARD_NBR)),
  by = .(PREMIUM_CUSTOMER,
         LIFESTAGE)][order(-AVG_QTY_ORDER)][,
 ORDER_PERCENT := paste(round(
   AVG_QTY_ORDER / sum(AVG_QTY_ORDER) * 100, 2), "%")]

head(x2,10)

ggplot(data = x2) + geom_col(aes(x = LIFESTAGE, y = AVG_QTY_ORDER,
                                 fill = PREMIUM_CUSTOMER),
                             position = position_dodge()) +
  labs(x = "Lifestage",
       y = "Average Quantity per Order",
       fill = "Premium Customer",
       title = "Average Quantity by Lifestage and Premium Customer Status") +
  theme_minimal() + ylim(0, 10) +
```

```
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  theme(panel.border = element_rect(
    color = "black",
    fill = NA,
    size = 1
  ))
```

```
## Warning: The `size` argument of `element_rect()` is deprecated as of ggplot2 3.4.0.
## i Please use the `linewidth` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



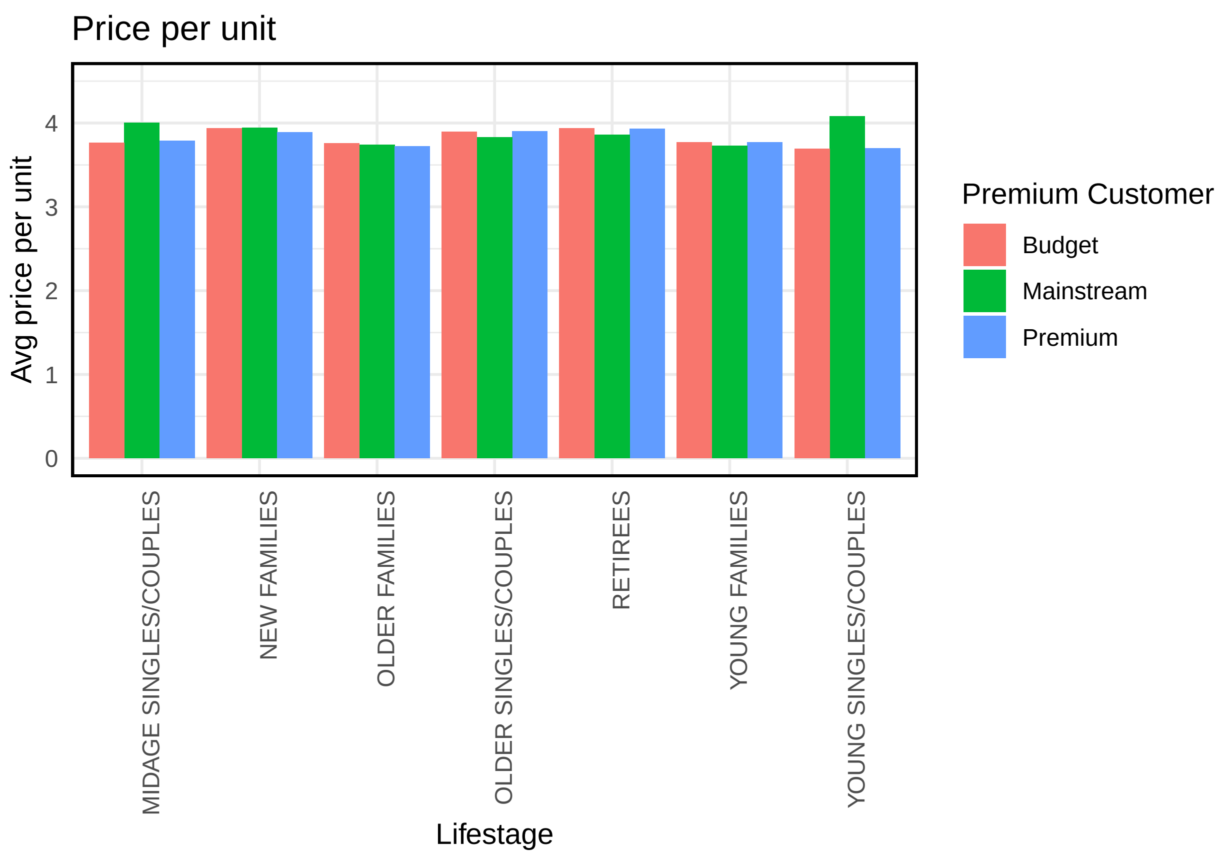
Older families and young families in general buy more chips per customer. Let's also investigate the average price per unit chips bought for each customer segment as this is also a driver of total sales.

```
data[, .(AVG_PRICE_PER_UNIT= sum(TOT_SALES)/sum(PROD_QTY)),
     by= .(PREMIUM_CUSTOMER,LIFESTAGE)][order(-AVG_PRICE_PER_UNIT)]
```

```
##     PREMIUM_CUSTOMER              LIFESTAGE AVG_PRICE_PER_UNIT
##               <char>                 <char>              <num>
##  1:       Mainstream  YOUNG SINGLES/COUPLES           4.079352
##  2:       Mainstream MIDAGE SINGLES/COUPLES           4.000391
##  3:       Mainstream           NEW FAMILIES           3.941504
```

```
##  4:            Budget           NEW FAMILIES           3.939123
##  5:            Budget               RETIREES           3.939045
##  6:           Premium               RETIREES           3.930269
##  7:           Premium  OLDER SINGLES/COUPLES           3.903869
##  8:            Budget  OLDER SINGLES/COUPLES           3.895923
##  9:           Premium           NEW FAMILIES           3.891298
## 10:        Mainstream               RETIREES           3.859303
## 11:        Mainstream  OLDER SINGLES/COUPLES           3.830869
## 12:           Premium MIDAGE SINGLES/COUPLES           3.786840
## 13:            Budget         YOUNG FAMILIES           3.769296
## 14:           Premium         YOUNG FAMILIES           3.767724
## 15:            Budget MIDAGE SINGLES/COUPLES           3.763703
## 16:            Budget         OLDER FAMILIES           3.756118
## 17:        Mainstream         OLDER FAMILIES           3.742401
## 18:        Mainstream         YOUNG FAMILIES           3.730285
## 19:           Premium         OLDER FAMILIES           3.725041
## 20:           Premium  YOUNG SINGLES/COUPLES           3.699962
## 21:            Budget  YOUNG SINGLES/COUPLES           3.693836
##     PREMIUM_CUSTOMER              LIFESTAGE AVG_PRICE_PER_UNIT
```

```
p3 <- ggplot(data = data[, .(AVG_PRICE_PER_UNIT = sum(TOT_SALES) / sum(PROD_QTY)),
  by = .(PREMIUM_CUSTOMER, LIFESTAGE)][order(-AVG_PRICE_PER_UNIT)]) +
  geom_col(aes(x = LIFESTAGE, y = AVG_PRICE_PER_UNIT, fill = PREMIUM_CUSTOMER),
           position = position_dodge()) +
  labs(x = "Lifestage",
       y = "Avg price per unit",
       fill = "Premium Customer",
       title = "Price per unit") +
  theme_minimal() + ylim(0, 4.5) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

p3 + theme(
  panel.border = element_rect(color = "black", fill = NA, size = 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 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.

```
# Two sample t-test
data[, PRICE_PER_UNIT := TOT_SALES / PROD_QTY]
#First-sample
sample1 <- data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") &
                  PREMIUM_CUSTOMER == "Mainstream", PRICE_PER_UNIT]
#Second-sample
sample2 <- data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") &
                  PREMIUM_CUSTOMER != "Mainstream", PRICE_PER_UNIT]

t.test(sample1, sample2, alternative = "greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  sample1 and sample2
## t = 37.301, df = 54383, p-value < 2.2e-16
```

```
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.3164708       Inf
## sample estimates:
## mean of x mean of y
##  4.045400  3.714329
```

The t-test results in a p-value < 2.2e-16, 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 mid-age singles and couples.

Deep dive into specific customer segments for insights

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
segment1 <- data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") &
                   PREMIUM_CUSTOMER == "Mainstream", ]

other <- data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") &
                PREMIUM_CUSTOMER != "Mainstream", ]
# Brand affinity compared to the rest of the population
segment1_qty <- segment1[, sum(PROD_QTY)]
other_qty <- other[, sum(PROD_QTY)]

seg1_qty_prop_byBrand <- segment1[, .(targetSegment = sum(PROD_QTY) / segment1_qty), by = Brand]
other_qty_prop_byBrand <- other[, .(other = sum(PROD_QTY) / other_qty), by = Brand]

affinity <- merge(seg1_qty_prop_byBrand, other_qty_prop_byBrand)[, affinityToBrand :=
                                                                   targetSegment /

affinity
```

```
##          Brand targetSegment       other affinityToBrand
##         <char>         <num>       <num>           <num>
##  1:     KETTLE   0.196738788 0.152812060       1.2874559
##  2:   TOSTITOS   0.044898459 0.035352196       1.2700331
##  3:   TWISTIES   0.045527631 0.035862182       1.2695165
##  4:    DORITOS   0.118092209 0.095489688       1.2367012
##  5:   TYRRELLS   0.029955608 0.024662900       1.2146020
##  6:       COBS   0.044880982 0.038024520       1.1803168
##  7:   PRINGLES   0.114229788 0.099202383       1.1514823
##  8:  INFUZIONS   0.063843546 0.056710389       1.1257822
##  9:    GRNWVES   0.032559684 0.030578732       1.0647820
## 10:      THINS   0.059421860 0.056220803       1.0569372
## 11:   CHEEZELS   0.018735363 0.018543073       1.0103699
## 12:     SMITHS   0.099863679 0.122784113       0.8133274
## 13:    CHEETOS   0.008843371 0.012606842       0.7014739
## 14:        RRD   0.047852075 0.072458742       0.6604044
## 15:    NATURAL   0.021549163 0.034944208       0.6166734
## 16:     FRENCH   0.003914852 0.006793007       0.5763062
```

```
## 17:        CCS    0.012426160 0.023683727       0.5246708
## 18: WOOLWORTHS    0.026879653 0.060117093       0.4471216
## 19:   SUNBITES    0.006326680 0.014952775       0.4231108
## 20:     BURGER    0.003460450 0.008200567       0.4219769
##           Brand targetSegment       other affinityToBrand
```

We can see that :

- Mainstream young singles/couples are 28% more likely to purchase KETTLE chips compared to the rest of the population
- Mainstream young singles/couples are 58% 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.

```
# Deep dive into Mainstream, young singles/couples
segment1 <- data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") &
                   PREMIUM_CUSTOMER == "Mainstream", ]

other <- data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") &
                PREMIUM_CUSTOMER != "Mainstream", ]
# Brand affinity compared to the rest of the population
segment1_packSize <- segment1[, sum(PROD_QTY)]
other_packSize <- other[, sum(PROD_QTY)]

seg1_qty_prop_byPackSize <- segment1[, .(targetSegment = sum(PROD_QTY) /
                                         segment1_packSize), by = packSize]
other_qty_prop_byPackSize <- other[, .(other = sum(PROD_QTY) / other_packSize),
                                   by = packSize]

packSizeProportion <- merge(seg1_qty_prop_byPackSize,
                            other_qty_prop_byPackSize)[, packSizeProp :=
                                                         targetSegment /other][order(-packSizeProp)]

packSizeProportion
```

```
##     packSize targetSegment       other packSizeProp
##        <num>         <num>       <num>        <num>
##  1:      330   0.060942361 0.044613533    1.3660062
##  2:      270   0.031546017 0.024173313    1.3049935
##  3:      380   0.030899367 0.024581302    1.2570273
##  4:      135   0.014733126 0.012117256    1.2158797
##  5:      110   0.105124262 0.087003529    1.2082758
##  6:      250   0.013981614 0.011688868    1.1961478
##  7:      210   0.028714740 0.024540503    1.1700958
##  8:      134   0.114229788 0.099202383    1.1514823
##  9:      170   0.080569052 0.080149324    1.0052368
## 10:      150   0.155405642 0.155586381    0.9988383
## 11:      175   0.261001783 0.274555803    0.9506329
## 12:      165   0.056398336 0.065400543    0.8623527
## 13:      190   0.008563739 0.013443218    0.6370304
## 14:      180   0.003844944 0.006038229    0.6367669
```

```
## 15:       70   0.003600266 0.007731380    0.4656692
## 16:      200   0.010154147 0.022378164    0.4537525
## 17:      160   0.007410256 0.016523531    0.4484668
## 18:      125   0.003093432 0.007119398    0.4345076
## 19:       90   0.006326680 0.014952775    0.4231108
## 20:      220   0.003460450 0.008200567    0.4219769
##     packSize targetSegment       other packSizeProp
```

It looks like Mainstream young singles/couples are 36% more likely to purchase a 330g pack of chips compared to the rest of the population but let's dive into what brands sell this pack size.

```
data[packSize==330, unique(PROD_NAME)]
```

```
## [1] "Doritos Cheese      Supreme 330g"
## [2] "Smiths Crinkle      Original 330g"
## [3] "Smiths Crinkle Chips Salt & Vinegar 330g"
## [4] "Cheezels Cheese 330g"
```

Doritos and Smiths are the only brand offering 330g packs and so this may instead be reflecting a higher likelihood of purchasing these.

Conclusion

- 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 28% more likely to purchase KETTLE chips compared to the rest of the population. The Category Manager may want to increase the category's performance by off-locating some KETTLE and smaller packs of chips in discretionary space near segments where young singles and couples frequent more often to increase visibilty and impulse behaviour.