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

Shams Reza Sajid

31-10-2022

## Task 1

### Loading the Libraries

```r
library(data.table)

library(knitr)

library(ggplot2)

library(ggmosaic)

library(readr)

library(readxl)

library (scales)
```

```
##
## Attaching package: 'scales'

## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
library (lessR)
```

```
##
## lessR 4.2.3                         feedback: gerbing@pdx.edu
## --------------------------------------------------------------
## > d <- Read("")   Read text, Excel, SPSS, SAS, or R data file
##   d is default data frame, data= in analysis routines optional
##
## Learn about reading, writing, and manipulating data, graphics,
## testing means and proportions, regression, factor analysis,
## customization, and descriptive statistics from pivot tables.
##   Enter:  browseVignettes("lessR")
##
## View changes in this and recent versions of lessR.
##   Enter: news(package="lessR")
##
## **New Feature**: Interactive analysis of your data
##   Enter: interact()
```

```
##
## Attaching package: 'lessR'

## The following object is masked from 'package:scales':
##
##     rescale

## The following object is masked from 'package:data.table':
##
##     set
```

```
library (tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --

## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v purrr   0.3.5      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()     masks data.table::between()
## x scales::col_factor() masks readr::col_factor()
## x purrr::discard()     masks scales::discard()
## x dplyr::filter()      masks stats::filter()
## x dplyr::first()       masks data.table::first()
## x dplyr::lag()         masks stats::lag()
## x dplyr::last()        masks data.table::last()
## x dplyr::recode()      masks lessR::recode()
## x dplyr::rename()      masks lessR::rename()
## x purrr::transpose()   masks data.table::transpose()
```

```
library(tinytex)


filePath <- ""

transactionData <- data.table(read_excel("QVI_transaction_data.xlsx", sheet = "in"))

customerData <- data.table(fread (paste0 (filePath, "QVI_purchase_behaviour.csv")))

str (transactionData)
```

## Ensuring Data is in Correct Format

```
### Examine Transaction Data

str(transactionData)
```

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 the CSV and Excel Integar dates begins on 30 Dec 1899
transactionData$DATE = as.Date(transactionData$DATE, origin = "1899-12-30")

#### Verifying The Transformed Date Column
str(transactionData)
```

```
## Classes 'data.table' and 'data.frame': 264836 obs. of 8 variables:
## $ DATE : Date, format: "2018-10-17" "2019-05-14" ...
```

```
## $ 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>
```

Checking to see if we are looking at the right product

```
#### Examining PROD_NAME
head(transactionData$PROD_NAME)
```

```
## [1] "Natural Chip        Compny SeaSalt175g"
## [2] "CCs Nacho Cheese    175g"
## [3] "Smiths Crinkle Cut  Chips Chicken 170g"
## [4] "Smiths Chip Thinly  S/Cream&Onion 175g"
## [5] "Kettle Tortilla ChpsHny&Jlpno Chili 150g"
## [6] "Old El Paso Salsa   Dip Tomato Mild 300g"
```

Determining if all chips are potato chips

```
#### Examine the words in PROD_NAME to see if there are any incorrect entries
productWords = data.table ( unlist (strsplit (unique (transactionData [, PROD_NAME]),"")))
setnames(productWords, 'words')

#### Removing any entries that are not strictly alphabetical characters using grepl
productWords = productWords [!grepl ('[^[:alpha:]]', productWords$words )]
print(productWords)
```

```
##       words
##    1:     N
##    2:     a
##    3:     t
##    4:     u
##    5:     r
##   ---
## 2719:     M
## 2720:     i
## 2721:     l
## 2722:     d
## 2723:     g
```

```
### Sorting by word frequency
head (sort(table(productWords$words), decreasing = T), 30)
```

```
##
## i e s C t l g r a n h o S p u m c k d R
## 234 228 168 161 161 159 157 150 143 139 136 135 102 66 53 52 42 38 32 32
## T y D P O W B w M K
## 32 28 27 26 22 22 20 17 16 13
```

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]
```

Summarizing the data to check for nulls and possible outliers

```r
#### Summarize 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
```

```r
#### Filtering the data set to examine the transactions in question

sort (table(transactionData$PROD_QTY), decreasing = T)
```

```
##
##      2      1      5      3      4    200
## 220070  25476    415    408    371      2
```

```r
print(transactionData[PROD_QTY > 226201])
```

```
## Empty data.table (0 rows and 8 cols):
DATE,STORE_NBR,LYLTY_CARD_NBR,TXN_ID,PROD_NBR,PROD_NAME...
```

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

sort (table(transactionData$PROD_QTY), decreasing = T)
```

```
##
##      2      1      5      3      4    200
## 220070  25476    415    408    371      2
```

```r
# Checking the number of PROD_QNT = 200 transactions
print (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
```

```r
# Checking to see if the cx had other transactions based on loyalty card number
print (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
```

```r
# Removing the cx
transactionData = transactionData[LYLTY_CARD_NBR != 226000]
```

Confirming that there are no missing data

```r
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
```

```r
missingData = transactionData[apply(transactionData, 1, function(x) any (!nzchar(x)) || any(is.na(x))),]

print (missingData)
```

```
## Empty data.table (0 rows and 8 cols):
DATE,STORE_NBR,LYLTY_CARD_NBR,TXN_ID,PROD_NBR,PROD_NAME...
```

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 dates.

```r
#### Counting the number of transactions by date
numDates = length (unique (transactionData$DATE))
print (numDates)
```

```
## [1] 364
```

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.

```r
#### Create a sequence of dates and join this the count of transactions by date

partialYear = as.Date(unique(transactionData$DATE), origin = "1899-12-30")
fullYear = seq(as.Date("2018/7/1"), by = "day", length.out = 365)
```

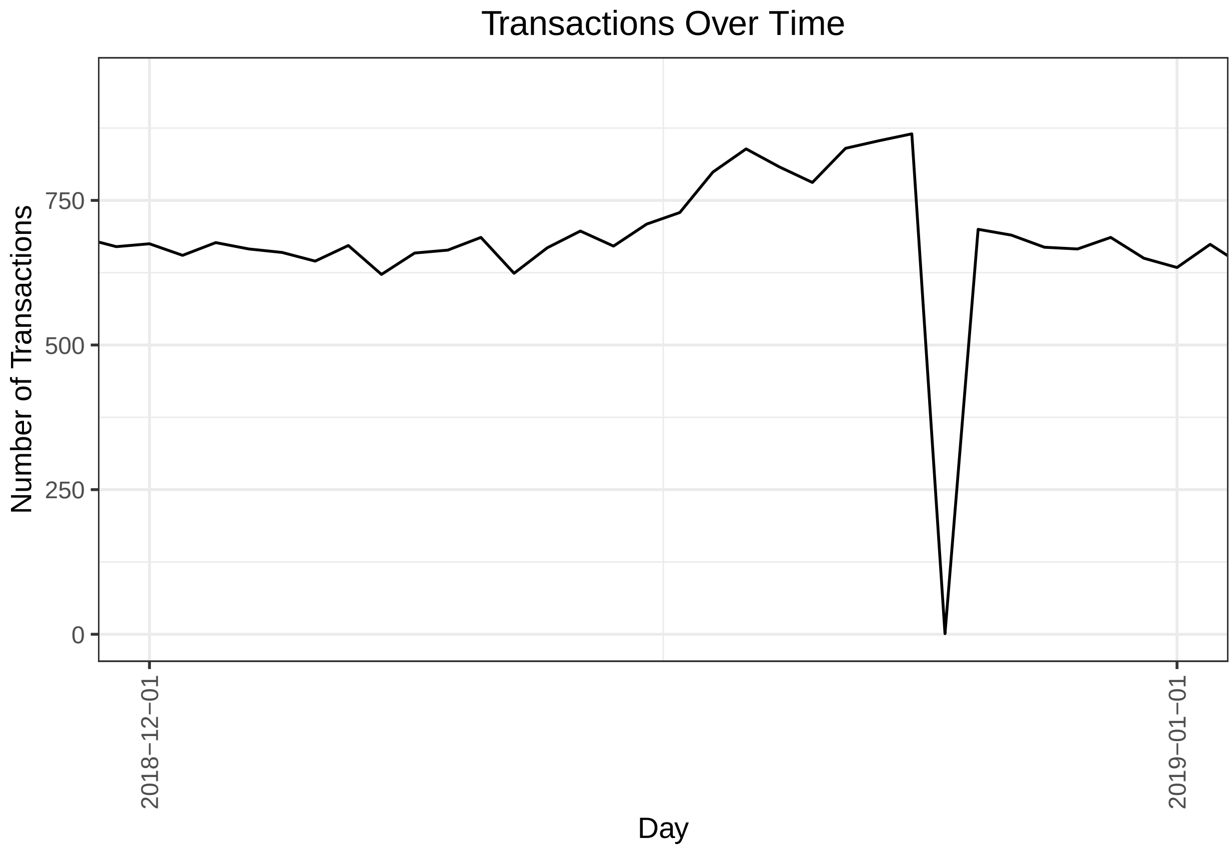
```r
missingDate = fullYear[!(fullYear %in% partialYear)]

print (missingDate)
```

```
## [1] "2018-12-25"
```

```r
transactionsByDay = data.table (table (c (as.Date(transactionData$DATE, origin = "1899-12-30"), missingI
setnames(transactionsByDay, c('day', 'count'))
transactionsByDay$day = as.Date(transactionsByDay$day)

str (transactionsByDay)
```

```
## Classes 'data.table' and 'data.frame':   365 obs. of  2 variables:
##  $ day  : Date, format: "2018-07-01" "2018-07-02" ...
##  $ count: int  663 650 674 669 660 711 695 653 692 650 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

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


#### Plot Transactions over time
ggplot (transactionsByDay, aes(x = transactionsByDay$day, y = transactionsByDay$count)) +
  geom_line() +
  labs (x = "Day", y = "Number of Transactions", title = "Transactions Over Time") + scale_x_date(breaks

  theme(axis.text.x = element_text(angle = 90, vjust = 0.5)) +

  #### Filtering to December and look individual days ( remove this to zoom out )
  coord_cartesian(xlim = c(as.Date('2018-12-01'),as.Date('2019-01-01')), ylim=c(0, 950))
```




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.

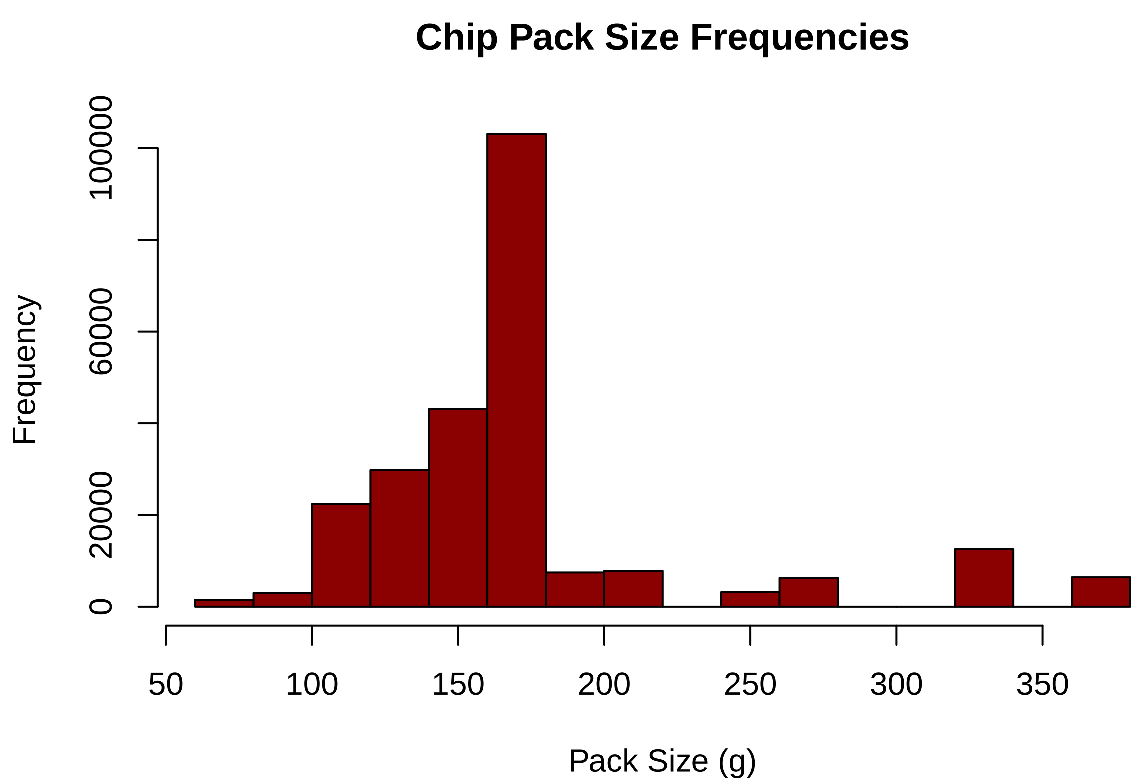
```r
#### Pack Size
#### We can work this out by taking the digits that are in PROD_NAME
transactionData [, PACK_SIZE := parse_number (PROD_NAME)]

#### Always Check 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
```

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 cont
hist(transactionData$PACK_SIZE,
     main = "Chip Pack Size Frequencies",
      xlab = "Pack Size (g)",
      ylab = "Frequency",
      col = "darkred")
```




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

```
#### Create Brand Names
transactionData [, BRAND := tstrsplit(PROD_NAME, " ", fixed = TRUE)[1]]
print (transactionData[, .N, BRAND] [order (BRAND)])
```

```
##          BRAND     N
##   1:    Burger  1564
##   2:       CCs  4551
##   3:   Cheetos  2927
```

```
##  4:   Cheezels  4603
##  5:       Cobs  9693
##  6:     Dorito  3183
##  7:    Doritos 22041
##  8:     French  1418
##  9:      Grain  6272
## 10:    GrnWves  1468
## 11:  Infuzions 11057
## 12:     Infzns  3144
## 13:     Kettle 41288
## 14:        NCC  1419
## 15:    Natural  6050
## 16:   Pringles 25102
## 17:        RRD 11894
## 18:        Red  4427
## 19:      Smith  2963
## 20:     Smiths 27390
## 21:      Snbts  1576
## 22:   Sunbites  1432
## 23:      Thins 14075
## 24:   Tostitos  9471
## 25:   Twisties  9454
## 26:   Tyrrells  6442
## 27:         WW 10320
## 28: Woolworths  1516
##          BRAND     N
```

```
#### Clean Brand Names
transactionData [ BRAND == "Red", BRAND := "RRD" ]
transactionData [ BRAND == "Infzns", BRAND := "Infuzions" ]
transactionData [ BRAND == "Snbts", BRAND := "Sunbites" ]
transactionData [ BRAND == "Smith", BRAND := "Smiths" ]
transactionData [ BRAND == "Dorito", BRAND := "Doritos" ]
transactionData [ BRAND == "Grain", BRAND := "GrnWves" ]
transactionData [ BRAND == "WW", BRAND := "Woolworths" ]



#### Confirming if the edits were successfull
print (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:        NCC  1419
## 12:    Natural  6050
## 13:   Pringles 25102
```

```
## 14:        RRD 16321
## 15:     Smiths 30353
## 16:   Sunbites  3008
## 17:      Thins 14075
## 18:   Tostitos  9471
## 19:   Twisties  9454
## 20:   Tyrrells  6442
## 21: Woolworths 11836
##          BRAND     N
```

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

```r
str(customerData)
```

**Examining Customer Data**

```
## 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>
```

```r
# Get Data Summary
summary (customerData)
```

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

```r
print (customerData [, .N, LIFESTAGE] [order(N, decreasing = 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
print (customerData [, .N, PREMIUM_CUSTOMER] [order(N, decreasing = TRUE)])
```

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

```r
#### Checking for any missing entries
print (customerData [is.null(PREMIUM_CUSTOMER), .N])
```

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

```
#### 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.

```
#### Checking for missing cx data
print (data [is.null(PREMIUM_CUSTOMER), .N])
```

```
## [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(filePath,"QVI_data.csv"))
```

Data exploration is now complete!

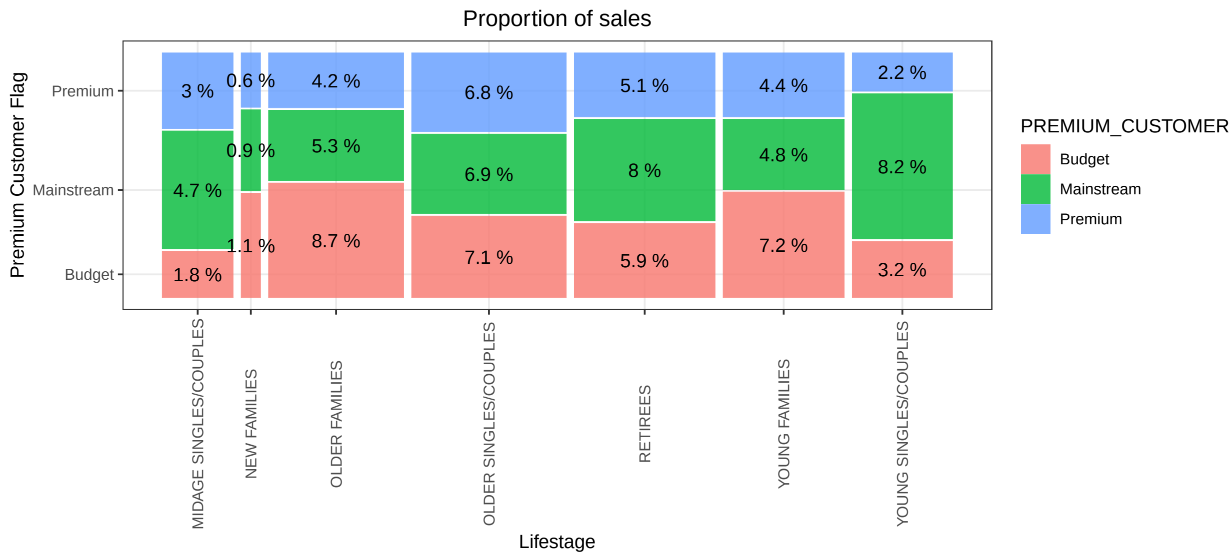
## Data Analysis

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.

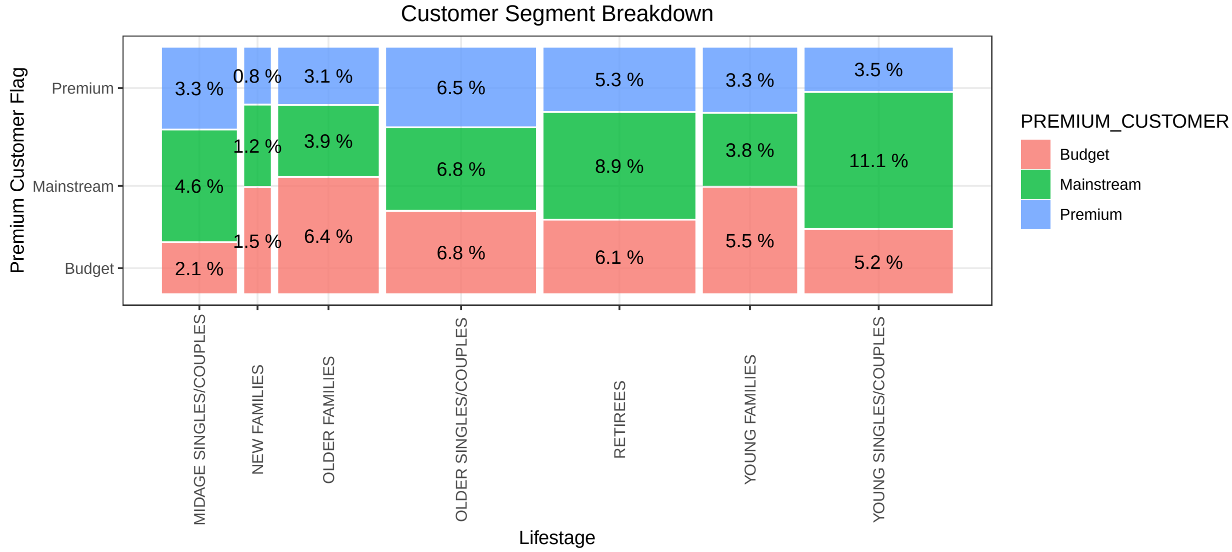
Let's start with calculating total sales by LIFESTAGE and PREMIUM_CUSTOMER and plotting the split by these segments to describe which customer segment contribute most to chip sales.

```
#### Total sales by LIFESTAGE and PREMIUM_CUSTOMER
sales <- data[,list(SALES=sum(TOT_SALES), packets=sum(PROD_QTY)), by=c('LIFESTAGE', 'PREMIUM_CUSTOMER')]


#### Creating a plot
p = ggplot(data = sales) +
  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, '%'))))
```

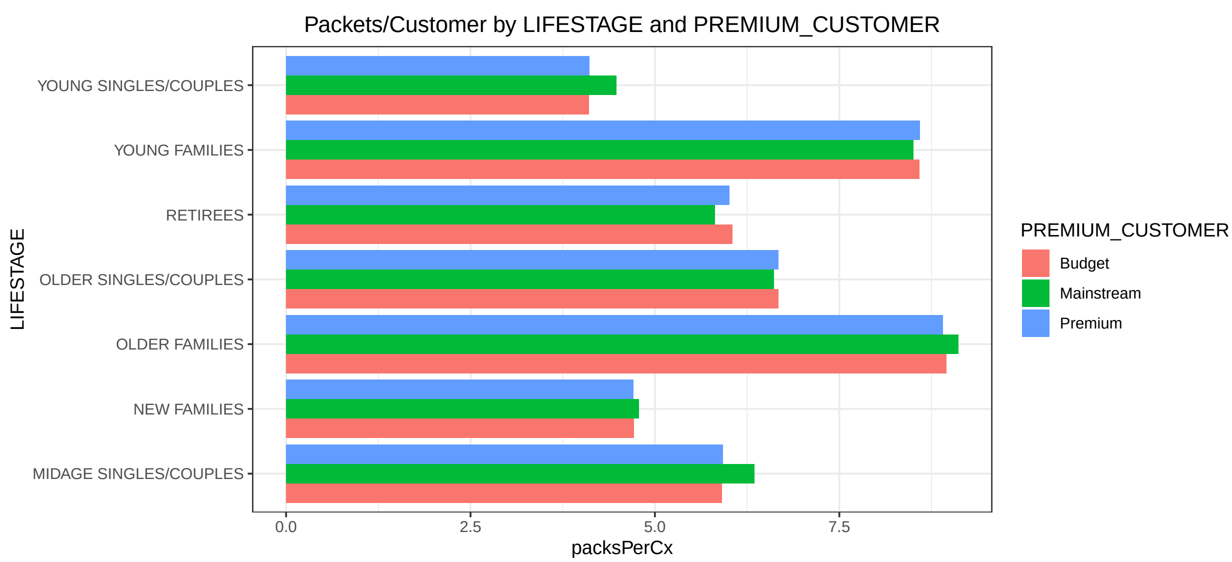



Sales are coming mainly from Budget - older families, Mainstream - young singles/couples, and Mainstream - retirees Let's see if the higher sales are due to there being more customers who buy chips.

```
#### Number of customers by LIFESTAGE and PREMIUM_CUSTOMER

cxBySegment = customerData [, .N, by = c('LIFESTAGE', 'PREMIUM_CUSTOMER')]


#### Creating a plot
p = ggplot(data = cxBySegment) +
  geom_mosaic(aes (weight = N, x = product(PREMIUM_CUSTOMER, LIFESTAGE), fill = PREMIUM_CUSTOMER)) +
  labs (x = "Lifestage", y = "Premium Customer Flag", title = "Customer Segment Breakdown") +
  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, '%'))))
```




```
#### Packets bought per customer by LIFESTAGE and PREMIUM_CUSTOMER
packetsPerCxBySegment = cxBySegment [, packsPerCx := sales$packets/N]

ggplot (packetsPerCxBySegment, aes (fill = PREMIUM_CUSTOMER, y = LIFESTAGE, x = packsPerCx)) +
  geom_bar(position = "dodge", stat = "identity") +
    ggtitle("Packets/Customer by LIFESTAGE and PREMIUM_CUSTOMER")
```



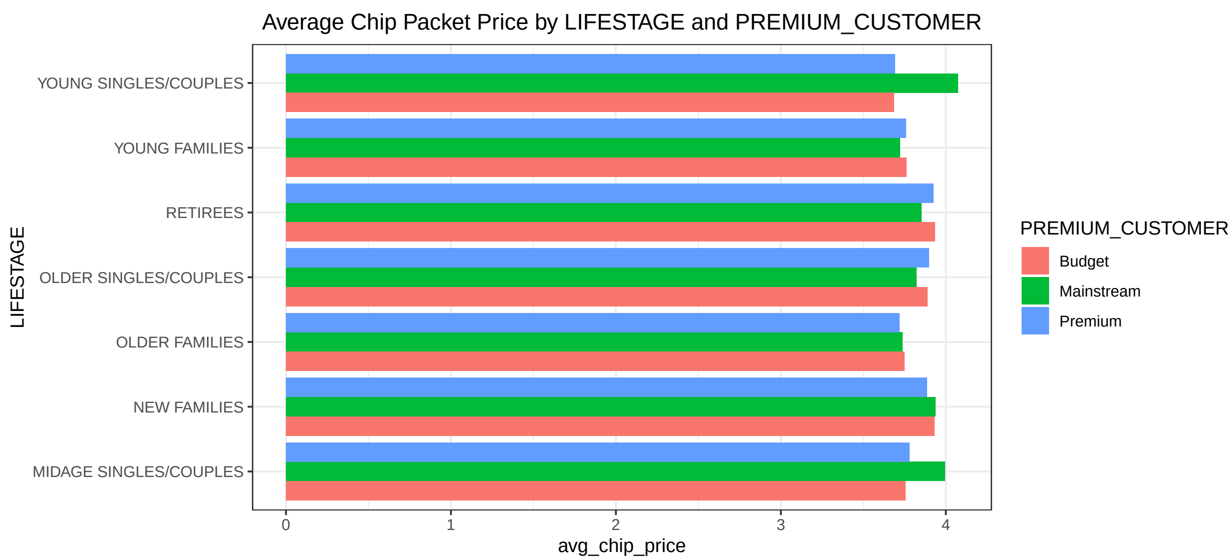

There are more Mainstream - young singles/couples and Mainstream - retirees who buy chips. This contributes to there being more sales to these customer segments but this is not a major driver for the Budget - Older families segment. Higher sales may also be driven by more units of chips being bought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

```
#### Average Price of units per customer by LIFESTAGE and PREMIUM_CUSTOMER

sales [, avg_chip_price := SALES / packets]
print(sales)
```

```
##                     LIFESTAGE PREMIUM_CUSTOMER     SALES packets avg_chip_price
```

```
##  1:  YOUNG SINGLES/COUPLES         Premium  39052.30   10575        3.692889
##  2:  YOUNG SINGLES/COUPLES      Mainstream 147582.20   36225        4.074043
##  3:         YOUNG FAMILIES          Budget 129717.95   34482        3.761903
##  4:  OLDER SINGLES/COUPLES      Mainstream 124648.50   32607        3.822753
##  5: MIDAGE SINGLES/COUPLES      Mainstream  84734.25   21213        3.994449
##  6:  YOUNG SINGLES/COUPLES          Budget  57122.10   15500        3.685297
##  7:           NEW FAMILIES         Premium  10760.80    2769        3.886168
##  8:         OLDER FAMILIES      Mainstream  96413.55   25804        3.736380
##  9:               RETIREES          Budget 105916.30   26932        3.932731
## 10:  OLDER SINGLES/COUPLES         Premium 123537.55   31695        3.897698
## 11:         OLDER FAMILIES          Budget 156863.75   41853        3.747969
## 12: MIDAGE SINGLES/COUPLES         Premium  54443.85   14400        3.780823
## 13:         OLDER FAMILIES         Premium  75242.60   20239        3.717703
## 14:               RETIREES      Mainstream 145168.95   37677        3.852986
## 15:               RETIREES         Premium  91296.65   23266        3.924037
## 16:         YOUNG FAMILIES      Mainstream  86338.25   23194        3.722439
## 17: MIDAGE SINGLES/COUPLES          Budget  33345.70    8883        3.753878
## 18:           NEW FAMILIES      Mainstream  15979.70    4060        3.935887
## 19:  OLDER SINGLES/COUPLES          Budget 127833.60   32883        3.887529
## 20:         YOUNG FAMILIES         Premium  78571.70   20901        3.759232
## 21:           NEW FAMILIES          Budget  20607.45    5241        3.931969
##                  LIFESTAGE PREMIUM_CUSTOMER     SALES packets avg_chip_price
```

```r
ggplot(sales, aes(fill=PREMIUM_CUSTOMER, y=LIFESTAGE, x= avg_chip_price)) +
    geom_bar(position="dodge", stat="identity") +
        ggtitle("Average Chip Packet Price by LIFESTAGE and PREMIUM_CUSTOMER")
```




```r
#### Perform an independent t-test between mainstream vs premium and budget midage and young singles an

data = data[, avg_Chip_Packet_Price := TOT_SALES/PROD_QTY]

mainstream = data [(LIFESTAGE == 'YOUNG SINGLES/COUPLES' | LIFESTAGE == 'MIDAGE SINGLES/COUPLES') &
                    PREMIUM_CUSTOMER == 'Mainstream', avg_Chip_Packet_Price]
```

```r
premium_Budget = data [(LIFESTAGE == 'YOUNG SINGLES/COUPLES' | LIFESTAGE == 'MIDAGE SINGLES/COUPLES') &
                      (PREMIUM_CUSTOMER == 'Budget' | PREMIUM_CUSTOMER == 'Premium'), avg_Chip_Packet_Pr


#### Running a t test to verify statistical significance
t.test(mainstream, premium_Budget, alternative = "greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  mainstream and premium_Budget
## t = 37.624, df = 54791, p-value < 0.0000000000000022
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.3187234       Inf
## sample estimates:
## mean of x mean of y
##  4.039786  3.706491
```

The t-test results in a p-value of 2.22e-16, i.e. the unit price for mainstream, young and mid-age singles and couples [ARE / ARE NOT] 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.

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

mainstream_Brands = data [LIFESTAGE == 'YOUNG SINGLES/COUPLES' & PREMIUM_CUSTOMER == 'Mainstream'] [, .N
old_Families_Brands = data [LIFESTAGE == 'OLDER FAMILIES' & PREMIUM_CUSTOMER == 'Budget'] [, .N, BRAND]

print(mainstream_Brands)
```

```
##          BRAND    N
##  1:     Kettle 3844
##  2:    Doritos 2379
##  3:   Pringles 2315
##  4:     Smiths 1921
##  5:  Infuzions 1250
##  6:      Thins 1166
##  7:   Twisties  900
##  8:   Tostitos  890
##  9:        RRD  875
## 10:       Cobs  864
## 11:    GrnWves  646
## 12:   Tyrrells  619
## 13: Woolworths  479
## 14:   Cheezels  346
## 15:    Natural  321
## 16:        CCs  222
## 17:    Cheetos  166
## 18:   Sunbites  128
```

```
## 19:     French   78
## 20:        NCC   73
## 21:     Burger   62
##          BRAND    N
```

```
print(old_Families_Brands)
```

```
##          BRAND    N
##  1:     Kettle 3320
##  2:     Smiths 2948
##  3:    Doritos 2032
##  4:   Pringles 1996
##  5:        RRD 1708
##  6: Woolworths 1213
##  7:  Infuzions 1185
##  8:      Thins 1171
##  9:   Twisties  810
## 10:       Cobs  760
## 11:   Tostitos  705
## 12:    GrnWves  671
## 13:    Natural  576
## 14:   Tyrrells  489
## 15:        CCs  451
## 16:   Cheezels  427
## 17:   Sunbites  305
## 18:    Cheetos  281
## 19:        NCC  165
## 20:     Burger  159
## 21:     French  142
##          BRAND    N
```

We can see that both share Kettle as their number 1 brand. If the client would like to target these segments then the Kettle brand would be ideal. Performing an affinity analysis.

```
#### Performing an Affinity Analysis

sector = data [LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER == "Mainstream", ]
other = data [!(LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER == "Mainstream"), ]

quantity_sector = sector [, sum(PROD_QTY)]
quantity_other = other [, sum(PROD_QTY)]

quantity_sector_Brand = sector [, .(targetSector = sum(PROD_QTY)/quantity_sector), by = BRAND]
quantity_other_Brand = other [, .(other = sum(PROD_QTY)/quantity_other), by = BRAND]

print (quantity_sector)
```

```
## [1] 36225
```

```
print (quantity_sector_Brand)
```

```
##          BRAND targetSector
##  1:        RRD  0.043809524
##  2:    Doritos  0.122760524
##  3:     Kettle  0.197984817
##  4:  Infuzions  0.064679089
##  5:     Smiths  0.096369910
```

```
##  6:    GrnWves  0.032712215
##  7:   Tyrrells  0.031552795
##  8:   Twisties  0.046183575
##  9:       Cobs  0.044637681
## 10:   Pringles  0.119420290
## 11:    Natural  0.015955832
## 12:   Cheezels  0.017971014
## 13:     Burger  0.002926156
## 14: Woolworths  0.024099379
## 15:   Sunbites  0.006349206
## 16:      Thins  0.060372671
## 17:   Tostitos  0.045410628
## 18:     French  0.003947550
## 19:        CCs  0.011180124
## 20:    Cheetos  0.008033126
## 21:        NCC  0.003643892
##           BRAND targetSector
```

```
brand_proportions = merge (quantity_sector_Brand, quantity_other_Brand) [, affinityToBrand := targetSec
brand_proportions[order(-affinityToBrand)]
```

```
##           BRAND targetSector       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  0.015955832 0.024980768       0.6387246
## 17:        NCC  0.003643892 0.005873221       0.6204248
## 18:        CCs  0.011180124 0.018895650       0.5916771
## 19:   Sunbites  0.006349206 0.012580210       0.5046980
## 20: Woolworths  0.024099379 0.049427188       0.4875733
## 21:     Burger  0.002926156 0.006596434       0.4435967
##           BRAND targetSector       other affinityToBrand
```

The affinity analysis takes into account the affinity of the target sectors for certain brands with respect to all the other sectors. From the analysis we can see that the mainstream young singles/couples are purchasing approximately 22.8% more Tyrell chips compared to the rest of the population. The analysis is in descending order and shows the purchasing differences between the mainstream young singles/couples and the other sectors. At the end, you can see that the Burger chips are not being purchased by the young singles/couples compared to the other sectors.

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
```

```r
mainstream_Packs = data [LIFESTAGE == 'YOUNG SINGLES/COUPLES' & PREMIUM_CUSTOMER == 'Mainstream'] [, .N
old_Families_Packs = data [LIFESTAGE == 'OLDER FAMILIES' & PREMIUM_CUSTOMER == 'Budget'] [, .N, PACK_SI

print(mainstream_Packs)
```

```
##     PACK_SIZE    N
##  1:       175 4997
##  2:       150 3080
##  3:       134 2315
##  4:       110 2051
##  5:       170 1575
##  6:       330 1195
##  7:       165 1102
##  8:       380  626
##  9:       270  620
## 10:       210  576
## 11:       135  290
## 12:       250  280
## 13:       200  179
## 14:       190  148
## 15:        90  128
## 16:       160  128
## 17:       180   70
## 18:        70   63
## 19:       220   62
## 20:       125   59
```

```r
print(old_Families_Packs)
```

```
##     PACK_SIZE    N
##  1:       175 5808
##  2:       150 3588
##  3:       134 1996
##  4:       110 1803
##  5:       170 1786
##  6:       165 1358
##  7:       330 1092
##  8:       270  532
##  9:       380  510
## 10:       210  505
## 11:       200  448
## 12:       190  312
## 13:       160  306
## 14:        90  305
## 15:       250  278
## 16:       135  268
## 17:       180  166
## 18:       220  159
## 19:       125  152
## 20:        70  142
```

```r
quantity_sector_Pack = sector [, .(targetSector = sum(PROD_QTY)/quantity_sector), by = PACK_SIZE]
quantity_other_Pack = other [, .(other = sum(PROD_QTY)/quantity_other), by = PACK_SIZE]

print (quantity_sector_Pack)
```

```
##     PACK_SIZE targetSector
##  1:       150  0.157598344
##  2:       170  0.080772947
##  3:       165  0.055652174
##  4:        70  0.003036577
##  5:       330  0.061283644
##  6:       180  0.003588682
##  7:       270  0.031828847
##  8:       110  0.106280193
##  9:       134  0.119420290
## 10:       380  0.032160110
## 11:       210  0.029123533
## 12:       175  0.254989648
## 13:       250  0.014354727
## 14:       220  0.002926156
## 15:       200  0.008971705
## 16:        90  0.006349206
## 17:       160  0.006404417
## 18:       135  0.014768806
## 19:       190  0.007481021
## 20:       125  0.003008972
```

```
print (quantity_sector_Brand)
```

```
##          BRAND targetSector
##  1:        RRD  0.043809524
##  2:    Doritos  0.122760524
##  3:     Kettle  0.197984817
##  4:  Infuzions  0.064679089
##  5:     Smiths  0.096369910
##  6:    GrnWves  0.032712215
##  7:   Tyrrells  0.031552795
##  8:   Twisties  0.046183575
##  9:       Cobs  0.044637681
## 10:   Pringles  0.119420290
## 11:    Natural  0.015955832
## 12:   Cheezels  0.017971014
## 13:     Burger  0.002926156
## 14: Woolworths  0.024099379
## 15:   Sunbites  0.006349206
## 16:      Thins  0.060372671
## 17:   Tostitos  0.045410628
## 18:     French  0.003947550
## 19:        CCs  0.011180124
## 20:    Cheetos  0.008033126
## 21:        NCC  0.003643892
##          BRAND targetSector
```

```
pack_proportions = merge (quantity_sector_Pack, quantity_other_Pack) [, affinityToPack := targetSector/
pack_proportions[order(-affinityToPack)]
```

```
##     PACK_SIZE targetSector       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
```

### Findings:

It was determined that the 2018-12-25 date was missing in the data. Since it was Christmas Day, it is assumed that it was closed. Sales increased int he days before Christmas.

Majority of the sales resulted from the Budget - older families, Mainstream - young singles/couples and Mainstream - retiree shoppers.

Mainstream, mid-age and young singles and couples are also more likely to pay more per packet of chips compared to other premium customers in their category.

Mainstream young singles and couples are 23% more likely to purchase Tyrrells chips compared to the rest of the population. It is suggested to increase the increase the category's performance by off-locating some Tyrrells and smaller packs of chips in discretionary space near segments where young singles and couples frequent more often to increase visibility and impulse behaviour.