Task 1: Data preparation and customer analytics

Xuan Fang

7/6/2021

Load required libraries and datasets

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

Load required libraries

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

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

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

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

Point the filePath to where you have downloaded the datasets to and

```
filePath <- "C:/Users/user/Desktop/Quantium_Internship/"
transactionData <- read.csv(paste0(filePath,"QVI_transaction_data.csv"))
customerData <- read.csv(paste0(filePath,"QVI_purchase_behaviour.csv"))
```

assign the data files to data.tables

Exploratory data analysis

Examining transaction data

We can use `str()` to look at the format of each column and see a sample of the data. As we have read in the dataset as a `data.table` object, we can also run `transactionData` in the console to see a sample of the data or use `head(transactionData)` to look at the first 10 rows.

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

```r
str(transactionData)
```

```
## '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 C
##  $ 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 ...
```

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

```
##     DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
## 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
## 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
```

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

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

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

```r
#### Examine PROD_NAME
transactionData %>% group_by(PROD_NAME) %>% count()
```

```
## # A tibble: 114 x 2
## # Groups:   PROD_NAME [114]
##    PROD_NAME                                    n
##    <chr>                                    <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
```

```
##  6 Cheetos Puffs 165g                      1448
##  7 Cheezels Cheese 330g                    3149
##  8 Cheezels Cheese Box 125g                1454
##  9 Cobs Popd Sea Salt  Chips 110g          3265
## 10 Cobs Popd Sour Crm  &Chives Chips 110g  3159
## # ... with 104 more rows
```

```
transactionData <- as.data.table(transactionData)
transactionData[ ,.N, by=PROD_NAME]
```

```
##                                     PROD_NAME    N
##   1:   Natural Chip        Compny SeaSalt175g 1468
##   2:                 CCs Nacho Cheese    175g 1498
##   3:   Smiths Crinkle Cut  Chips Chicken 170g 1484
##   4:   Smiths Chip Thinly  S/Cream&Onion 175g 1473
##   5: Kettle Tortilla ChpsHny&Jlpno Chili 150g 3296
##  ---
## 110:    Red Rock Deli Chikn&Garlic Aioli 150g 1434
## 111:      RRD SR Slow Rst     Pork Belly 150g 1526
## 112:                 RRD Pc Sea Salt     165g 1431
## 113:       Smith Crinkle Cut   Bolognese 150g 1451
## 114:                 Doritos Salsa Mild  300g 1472
```

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

```
#### Examine the words in PROD_NAME to see if there are any incorrect entries
#### such as products that are not chips
productWords <- data.table(unlist(strsplit(unique(transactionData$PROD_NAME), " ")))
setnames(productWords, 'words') # There are 823 words
```

As we are only interested in words that will tell us if the product is chips or not, let's remove all words with digits and special characters such as '&' from our set of product words. We can do this using `grepl()`. `grepl()` returns a logical vector indicating which element of a character vector contains the match

```
# Note:`\d:` matches any digit.
# Remember, to create a regular expression containing \d or \s,
# you'll need to escape the \ for the string, so you'll type "\\d" or "\\s"
productWords02 <- productWords[!grepl("\\d", productWords$words)]
```

Removing digits

```
# In other words, only keep alphabetic characters
productWords03 <- productWords02[grepl("[[:alpha:]]", productWords02$words)] #There are 485 words left

productWords04 <- productWords03[!grepl("[[:punct:]]", productWords03$words)] #There are 437 words left
```

Removing special characters

Let's look at the most common words by counting the number of times a word appears

```
productWords04[, .N, words][order(-N)]
```

And, sorting them by this frequency in order of highest to lowest frequency

```
##          words  N
##   1:     Chips 21
##   2:    Smiths 16
##   3:   Crinkle 14
##   4:       Cut 14
##   5:    Kettle 13
##  ---
## 164:       Rst  1
## 165:      Pork  1
## 166:     Belly  1
## 167:        Pc  1
## 168: Bolognese  1
```

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

```
transactionData02 <- transactionData[!grepl("salsa", tolower(transactionData$PROD_NAME)), ]
```

```
#  246742 records on new dataset vs. 264836 records on old one
dim(transactionData02)
```

Remove salsa products

```
## [1] 246742      8
```

```
dim(transactionData)
```

```
## [1] 264836      8
```

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).

```
#### Summarise the data to check for nulls and possible outliers
summary(transactionData02)
```

```
##       DATE              STORE_NBR     LYLTY_CARD_NBR        TXN_ID
##  Min.   :2018-07-01   Min.   :  1.0   Min.   :   1000   Min.   :      1
##  1st Qu.:2018-09-30   1st Qu.: 70.0   1st Qu.:  70015   1st Qu.:  67569
##  Median :2018-12-30   Median :130.0   Median : 130367   Median : 135183
##  Mean   :2018-12-30   Mean   :135.1   Mean   : 135531   Mean   : 135131
##  3rd Qu.:2019-03-31   3rd Qu.:203.0   3rd Qu.: 203084   3rd Qu.: 202654
##  Max.   :2019-06-30   Max.   :272.0   Max.   :2373711   Max.   :2415841
##     PROD_NBR       PROD_NAME            PROD_QTY         TOT_SALES
##  Min.   :  1.00   Length:246742      Min.   :  1.000   Min.   :  1.700
##  1st Qu.: 26.00   Class :character   1st Qu.:  2.000   1st Qu.:  5.800
##  Median : 53.00   Mode  :character   Median :  2.000   Median :  7.400
##  Mean   : 56.35                      Mean   :  1.908   Mean   :  7.321
##  3rd Qu.: 87.00                      3rd Qu.:  2.000   3rd Qu.:  8.800
##  Max.   :114.00                      Max.   :200.000   Max.   :650.000
```

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

```
#### Filter the dataset to find the outlier(s)
subset(transactionData02, PROD_QTY==200)
```

```
##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
## 1: 2018-08-19       226         226000 226201        4
## 2: 2019-05-20       226         226000 226210        4
##                            PROD_NAME PROD_QTY TOT_SALES
## 1: Dorito Corn Chp     Supreme 380g      200       650
## 2: Dorito Corn Chp     Supreme 380g      200       650
```

There are two transactions where 200 packets of chips are bought in one transaction and both of these transactions were by the same customer (LYLTY_CARD_NBR = 226000).

```
#### Let's see if the customer has had other transactions
transactionData02[transactionData02$LYLTY_CARD_NBR==226000, ]
```

```
##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
## 1: 2018-08-19       226         226000 226201        4
## 2: 2019-05-20       226         226000 226210        4
##                            PROD_NAME PROD_QTY TOT_SALES
## 1: Dorito Corn Chp     Supreme 380g      200       650
## 2: Dorito Corn Chp     Supreme 380g      200       650
```

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

```
#### Filter out the customer based on the loyalty card number
transactionData03 <- subset(transactionData02, LYLTY_CARD_NBR!=226000)
#### Re-examine transaction data
summary(transactionData03)
```

```
##       DATE              STORE_NBR     LYLTY_CARD_NBR        TXN_ID
##  Min.   :2018-07-01   Min.   :  1.0   Min.   :   1000   Min.   :      1
##  1st Qu.:2018-09-30   1st Qu.: 70.0   1st Qu.:  70015   1st Qu.:  67569
##  Median :2018-12-30   Median :130.0   Median : 130367   Median : 135182
##  Mean   :2018-12-30   Mean   :135.1   Mean   : 135530   Mean   : 135130
##  3rd Qu.:2019-03-31   3rd Qu.:203.0   3rd Qu.: 203083   3rd Qu.: 202652
##  Max.   :2019-06-30   Max.   :272.0   Max.   :2373711   Max.   :2415841
##     PROD_NBR        PROD_NAME           PROD_QTY       TOT_SALES
##  Min.   :  1.00   Length:246740      Min.   :1.000   Min.   : 1.700
##  1st Qu.: 26.00   Class :character   1st Qu.:2.000   1st Qu.: 5.800
##  Median : 53.00   Mode  :character   Median :2.000   Median : 7.400
##  Mean   : 56.35                      Mean   :1.906   Mean   : 7.316
##  3rd Qu.: 87.00                      3rd Qu.:2.000   3rd Qu.: 8.800
##  Max.   :114.00                      Max.   :5.000   Max.   :29.500
```

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

```
#### Count the number of transactions by date
transactions_by_day <- transactionData03[, .N, by = DATE]
transactions_by_day
```

```
##            DATE   N
##   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. Let's create a sequence of dates from 1 Jul 2018 to 30 Jun 2019 and use this to create a chart of number of transactions over time to find the missing date.

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

bd <- as.Date("2018-07-01")
ed <- as.Date("2019-06-30")
dates02 <- seq(bd, ed, by = 1)

#### Find the date that is missing
dates01 <- unique(transactionData03$DATE) # original
dates02[!dates02 %in% dates01]
```

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

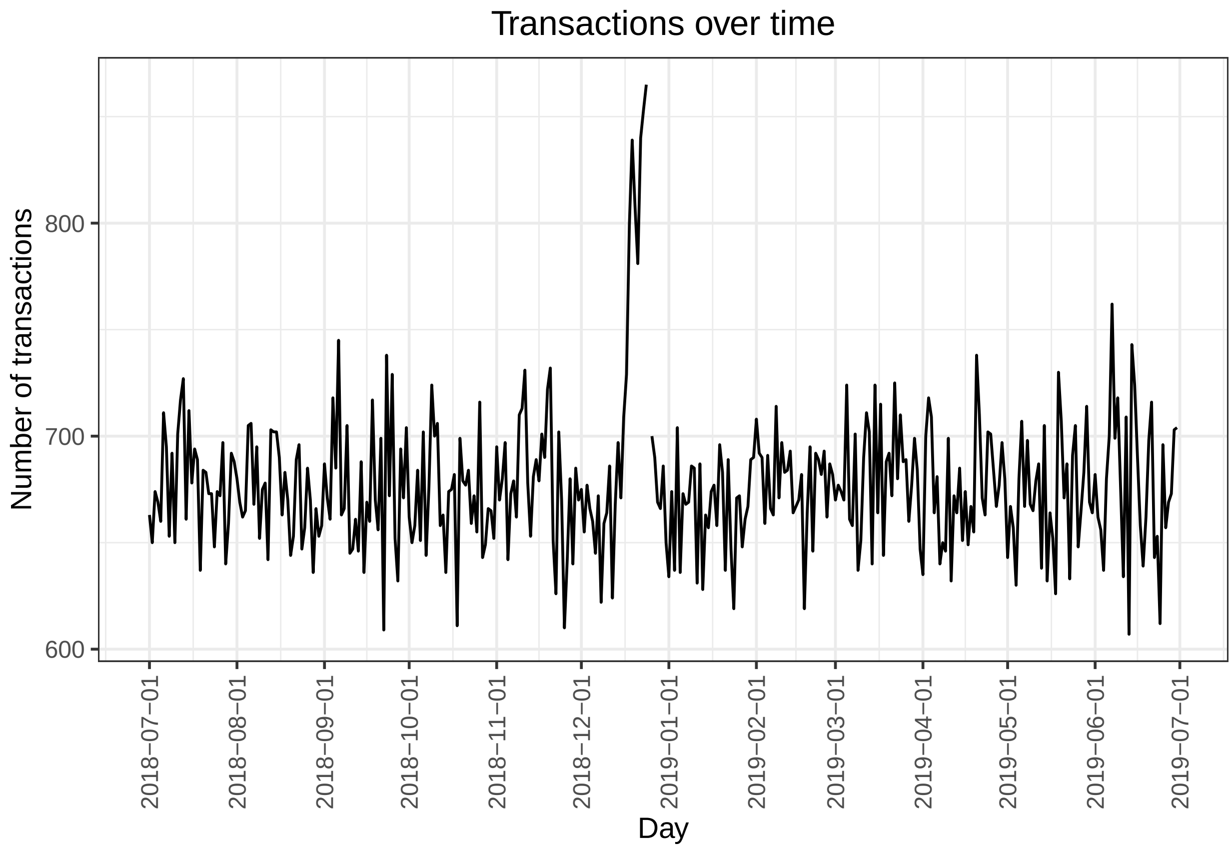
```
#### fill in the missing day.
transactions_by_day.new <- rbind(data.frame(DATE = as.Date("2018-12-25"), N=NA), transactions_by_day)

# Method 2
allDates <- data.frame(dates02)
setnames(allDates, "DATE")
transactions_by_day.new <- merge(allDates , transactions_by_day, all.x = TRUE)
```

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

Setting plot themes to format graphs

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

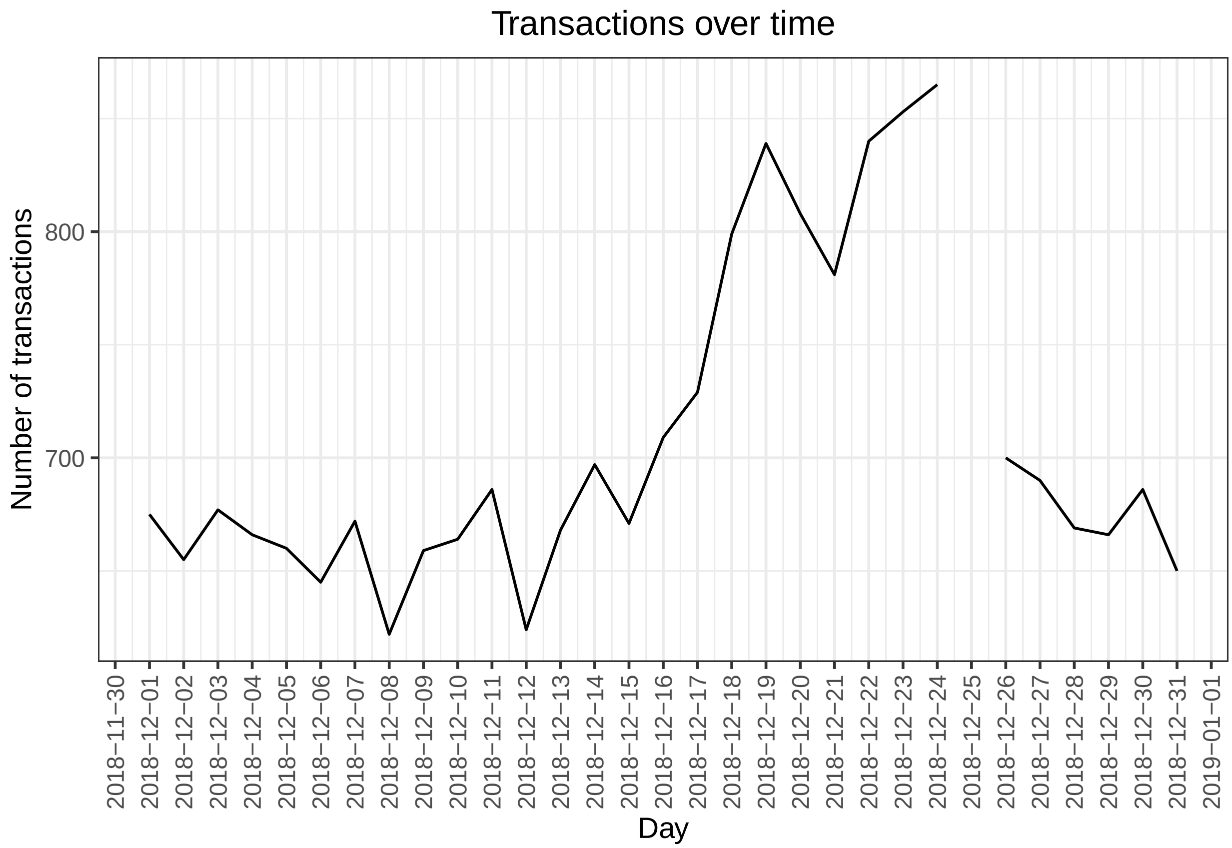
Plot transactions over time



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

Filter to December and look at individual days

```
ggplot(transactions_by_day.new[month(transactions_by_day.new$DATE)==12, ],
       aes(x = DATE, y = as.numeric(N))) + geom_line() +
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 that we are satisfied that the data no longer has outliers, we can move on to creating other features such as brand of chips or pack size from PROD_NAME. We will start with pack size.

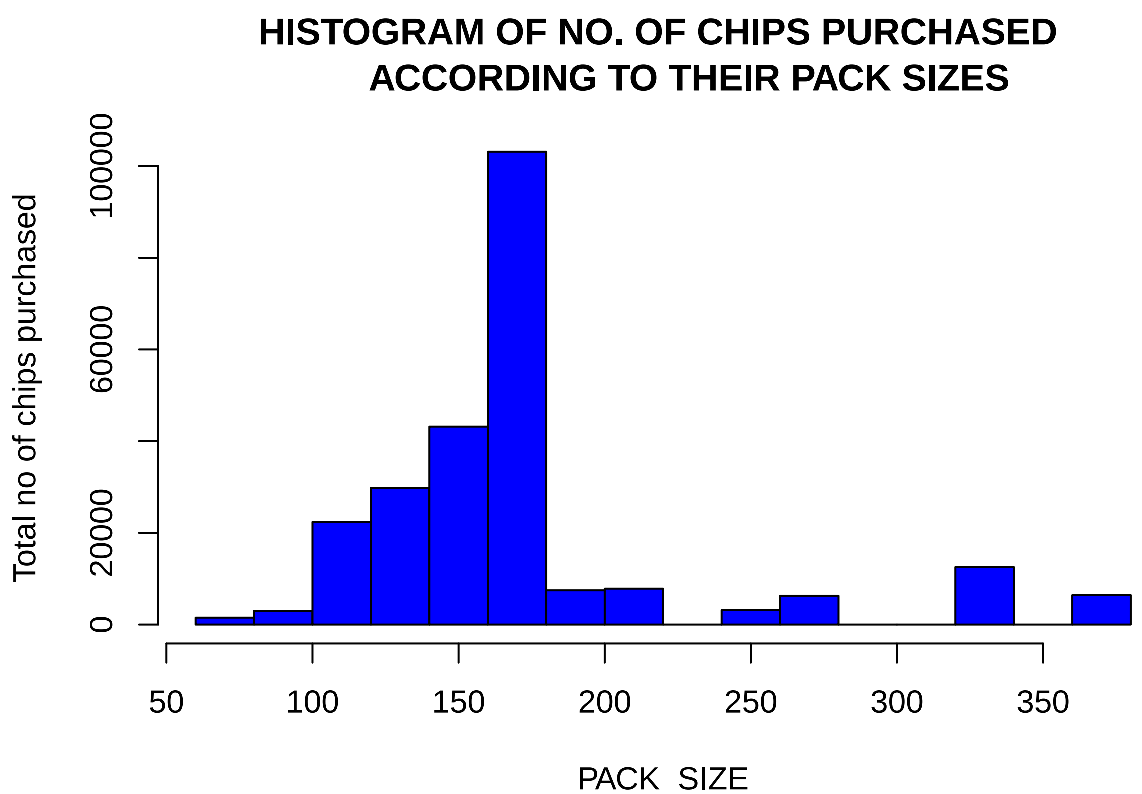
```
#### Pack size
# We can work this out by taking the digits that are in PROD_NAME
transactionData03[, PACK_SIZE := parse_number(as.character(PROD_NAME))]

# Let's check if the pack sizes look sensible
transactionData03[, .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! Plot a histogram showing the number of transactions by pack size.

```r
# plotting histogram of the packsize
options(scipen=999) # turn off scientific notations like 1e+05
hist(transactionData03$PACK_SIZE, col = "blue", border = "black" ,
xlab = "PACK  SIZE", ylab = "Total no of chips purchased",
main = "HISTOGRAM OF NO. OF CHIPS PURCHASED
     ACCORDING TO THEIR PACK SIZES")
```



Pack sizes created look reasonable with no outliers. From the plot it can be seen that the the packs of size 170-180 was purchased the most.

Now to create brands, we can use the first word in PROD_NAME to work out the brand name:

```r
regexp <- "^([[:alpha:]]+)"

##### extract first match using `regmatches()`
transactionData03$BRAND <- regmatches(transactionData03$PROD_NAME,
                                      regexpr(regexp, transactionData03$PROD_NAME))
```

```
#### Table of brand
table(transactionData03$BRAND)
```

```
##
##     Burger        CCs     Cheetos    Cheezels       Cobs      Dorito     Doritos
##       1564       4551        2927        4603       9693        3183       22041
##     French      Grain     GrnWves   Infuzions     Infzns      Kettle     Natural
##       1418       6272        1468       11057       3144       41288        6050
##        NCC   Pringles         Red         RRD      Smith      Smiths       Snbts
##       1419      25102        4427       11894       2963       27390        1576
##   Sunbites      Thins    Tostitos    Twisties   Tyrrells Woolworths          WW
##       1432      14075        9471        9454       6442        1516       10320
```

Some of the brand names look like they are of the same brands - such as RED and RRD, which are both Red Rock Deli chips. Let's combine these together.

Clean brand names

1) Replace "Red" with "RRD"

```
transactionData03$BRAND[which(transactionData03$BRAND == "Red")]<-"RRD"
table(transactionData03$BRAND)
```

```
##
##     Burger        CCs     Cheetos    Cheezels       Cobs      Dorito     Doritos
##       1564       4551        2927        4603       9693        3183       22041
##     French      Grain     GrnWves   Infuzions     Infzns      Kettle     Natural
##       1418       6272        1468       11057       3144       41288        6050
##        NCC   Pringles         RRD       Smith     Smiths       Snbts    Sunbites
##       1419      25102       16321        2963      27390        1576        1432
##      Thins   Tostitos    Twisties    Tyrrells Woolworths          WW
##      14075       9471        9454        6442       1516       10320
```

2) Replace "Smith" with " Smiths"

```
transactionData03$BRAND[which(transactionData03$BRAND == "Smith")] <-"Smiths"
table(transactionData03$BRAND)
```

```
##
##     Burger        CCs     Cheetos    Cheezels       Cobs      Dorito     Doritos
##       1564       4551        2927        4603       9693        3183       22041
##     French      Grain     GrnWves   Infuzions     Infzns      Kettle     Natural
##       1418       6272        1468       11057       3144       41288        6050
##        NCC   Pringles         RRD      Smiths      Snbts    Sunbites       Thins
##       1419      25102       16321       30353       1576        1432       14075
##   Tostitos   Twisties    Tyrrells Woolworths         WW
##       9471       9454        6442        1516      10320
```

3) Other replacements

```
transactionData03$BRAND[which(transactionData03$BRAND == "Snbts")] <-"Sunbites"
transactionData03$BRAND[which(transactionData03$BRAND == "Infzns")] <-"Infuzions"
transactionData03$BRAND[which(transactionData03$BRAND == "WOOLWORTHS")] <-"Woolworths"
transactionData03$BRAND[which(transactionData03$BRAND == "WW")] <-"Woolworths"
transactionData03$BRAND[which(transactionData03$BRAND == "NATURAL")] <-"Natural"
transactionData03$BRAND[which(transactionData03$BRAND == "Dorito")] <-"Doritos"
transactionData03$BRAND[which(transactionData03$BRAND == "Grain")] <-"GrnWves"
```

```
table(transactionData03$BRAND)
```

```
##
##     Burger        CCs     Cheetos    Cheezels       Cobs     Doritos      French
##       1564       4551       2927       4603       9693       25224        1418
##    GrnWves   Infuzions      Kettle     Natural        NCC    Pringles         RRD
##       7740      14201      41288       6050       1419       25102       16321
##     Smiths   Sunbites       Thins    Tostitos   Twisties    Tyrrells  Woolworths
##      30353       3008      14075       9471       9454        6442       11836
```

Examining customer data

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

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

```
## '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 SI
##  $ PREMIUM_CUSTOMER: chr  "Premium" "Mainstream" "Budget" "Mainstream" ...
```

```
head(customerData)
```

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

1) Examining `LIFESTAGE`

```
sort(table(customerData$LIFESTAGE), decreasing = TRUE)
```

```
##
##               RETIREEES  OLDER SINGLES/COUPLES  YOUNG SINGLES/COUPLES
##                  14805                  14609                  14441
##         OLDER FAMILIES         YOUNG FAMILIES MIDAGE SINGLES/COUPLES
##                   9780                   9178                   7275
##           NEW FAMILIES
##                   2549
```

2) Examining `PREMIUM_CUSTOMER`

```
sort(table(customerData$PREMIUM_CUSTOMER), decreasing = TRUE)
```

```
##
## Mainstream     Budget    Premium
##      29245      24470      18922
```

```
data <- merge(transactionData03, customerData,  all.x = TRUE)
```

Merge transaction data to customer data As the number of rows in `data` is the same as that of `transactionData03`, 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 `transactionData03` 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.

```
str(data)
```

```
## Classes 'data.table' and 'data.frame':   246740 obs. of  12 variables:
##  $ LYLTY_CARD_NBR  : int  1000 1002 1003 1003 1004 1005 1007 1007 1009 1010 ...
##  $ DATE            : Date, format: "2018-10-17" "2018-09-16" ...
##  $ STORE_NBR       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ TXN_ID          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ PROD_NBR        : int  5 58 52 106 96 86 49 10 20 51 ...
##  $ PROD_NAME       : chr  "Natural Chip        Compny SeaSalt175g" "Red Rock Deli Chikn&Garlic Aioli
##  $ PROD_QTY        : int  2 1 1 1 1 1 1 1 1 2 ...
##  $ TOT_SALES       : num  6 2.7 3.6 3 1.9 2.8 3.8 2.7 5.7 8.8 ...
##  $ PACK_SIZE       : num  175 150 210 175 160 165 110 150 330 170 ...
##  $ BRAND           : chr  "Natural" "RRD" "GrnWves" "Natural" ...
##  $ LIFESTAGE       : chr  "YOUNG SINGLES/COUPLES" "YOUNG SINGLES/COUPLES" "YOUNG FAMILIES" "YOUNG FA
##  $ PREMIUM_CUSTOMER: chr  "Premium" "Mainstream" "Budget" "Budget" ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "sorted")= chr "LYLTY_CARD_NBR"
```

Let's also check if some customers were not matched on by 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 the transaction data has been accounted for in the customer dataset.

Note that if you are continuing with Task 2, you may want to retain this dataset which you can write out as a csv

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

Data exploration is now complete!

Data analysis on customer segments

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

- Who spends the most on chips (total sales), describing customers by lifestage and how premium their general purchasing behaviour is
- How many customers are in each segment
- How many chips are bought per customer by segment
- 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

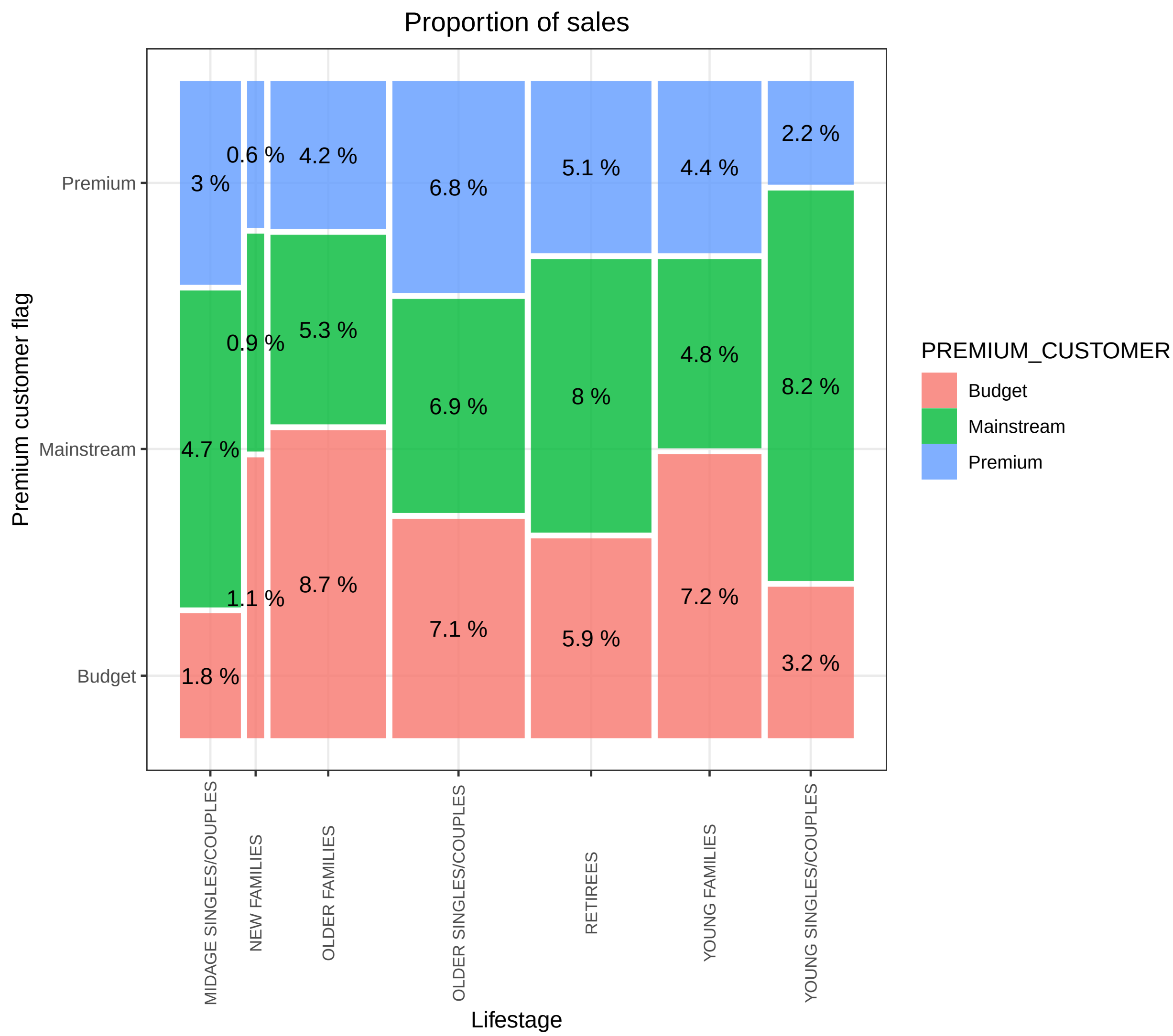
Let's start with calculating total sales by `LIFESTAGE` and `PREMIUM_CUSTOMER` and plotting the split by these segments to describe which customer segment contribute most to chip sales.

```
sales <- data %>% group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(Grandtotal_SALES = sum(TOT_SALES))
```

Total sales by LIFESTAGE and PREMIUM_CUSTOMER

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

```
# create plot
p1 <- ggplot(data = sales) +
  geom_mosaic(aes(weight = Grandtotal_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, size = 8))

# 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, '%'))))
```



Sales are coming mainly from Budget - older families (8.7%), Mainstream - young singles/couples (8.2%), and Mainstream - retirees (8%)

Let's see if the higher sales are due to there being more customers who buy chips.

```
no_of_customers  <- data %>% group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(customer_count = length(unique(LYLTY_CARD_NBR)))
```

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

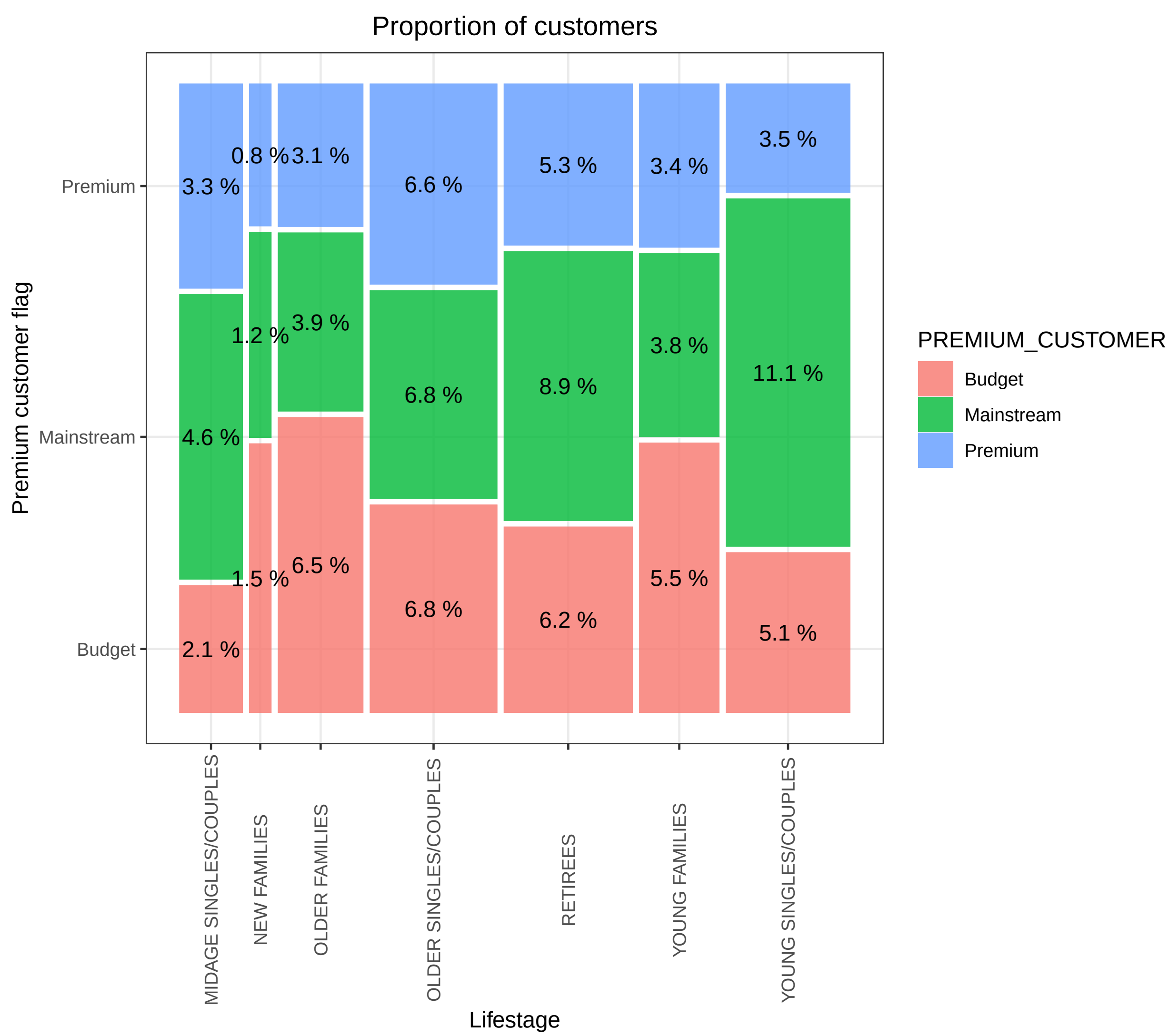
```
no_of_customers
```

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER customer_count
##    <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
## # ... with 11 more rows
```

Create plot

```
p2 <- ggplot(data = no_of_customers) + geom_mosaic(aes(weight = customer_count,
                                                        x = product(PREMIUM_CUSTOMER, LIFESTAGE),
                                                        fill = PREMIUM_CUSTOMER)) +
  labs(x = "Lifestage", y = "Premium customer flag", title = "Proportion of customers") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))

p2 + geom_text(data = ggplot_build(p2)$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 (11.1%) and Mainstream - retirees (8.9%) 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.

Calculate the summary of number of customers by those dimensions and create a plot.

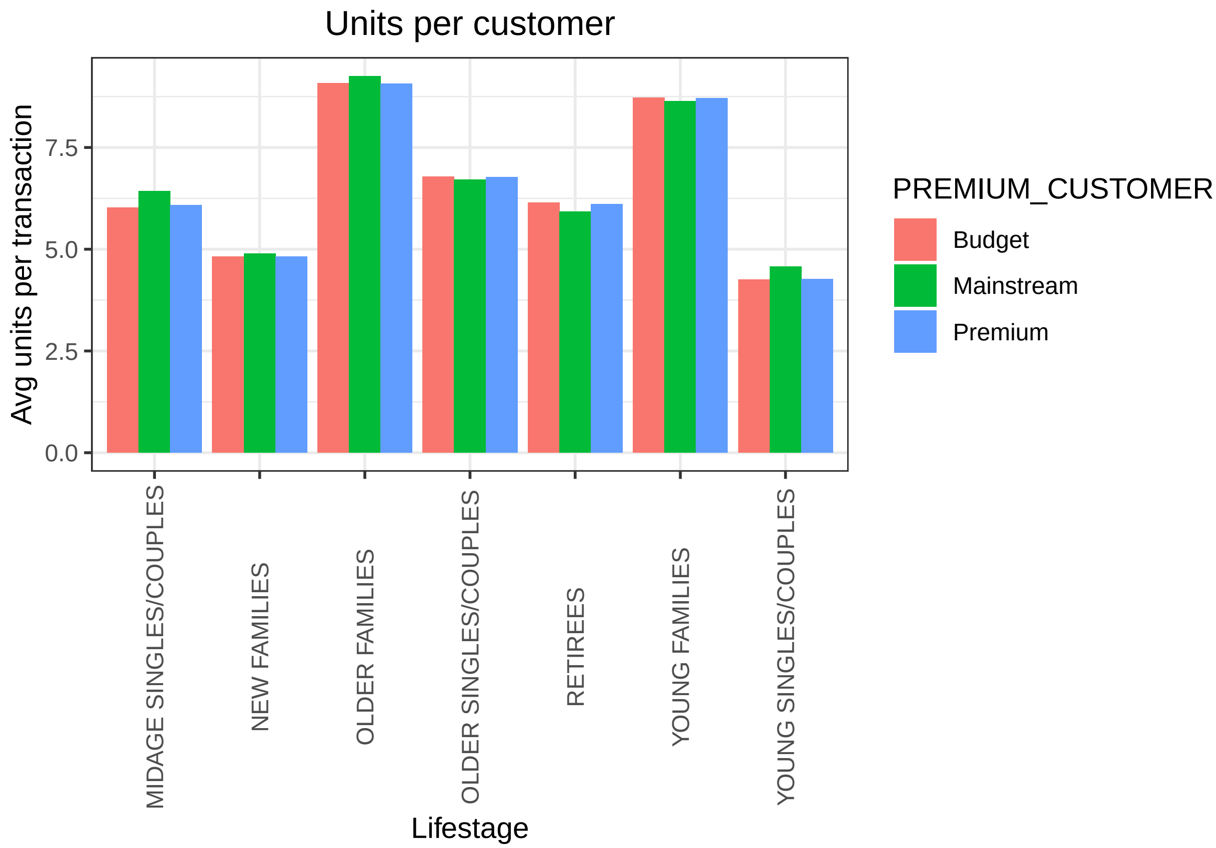
```
#### Number of customers by LIFESTAGE and PREMIUM_CUSTOMER
units <-  data %>% group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(units_count = (sum(PROD_QTY)/uniqueN(LYLTY_CARD_NBR)))
```

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

```
units
```

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER units_count
##    <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
## # ... with 11 more rows
```

```
ggplot(data = units, aes(weight = units_count, x = LIFESTAGE, fill = PREMIUM_CUSTOMER)) +
  geom_bar(position = position_dodge()) +
  labs(x = "Lifestage", y = "Avg units per transaction", title = "Units per customer") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```




```
units[order(units$units_count, decreasing = TRUE), ]
```

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER units_count
##    <chr>                  <chr>                  <dbl>
##  1 OLDER FAMILIES         Mainstream              9.26
##  2 OLDER FAMILIES         Budget                  9.08
##  3 OLDER FAMILIES         Premium                 9.07
##  4 YOUNG FAMILIES         Budget                  8.72
##  5 YOUNG FAMILIES         Premium                 8.72
##  6 YOUNG FAMILIES         Mainstream              8.64
##  7 OLDER SINGLES/COUPLES  Budget                  6.78
##  8 OLDER SINGLES/COUPLES  Premium                 6.77
##  9 OLDER SINGLES/COUPLES  Mainstream              6.71
## 10 MIDAGE SINGLES/COUPLES Mainstream              6.43
## # ... with 11 more rows
```

In general, older families and young families buy more chips per customer.

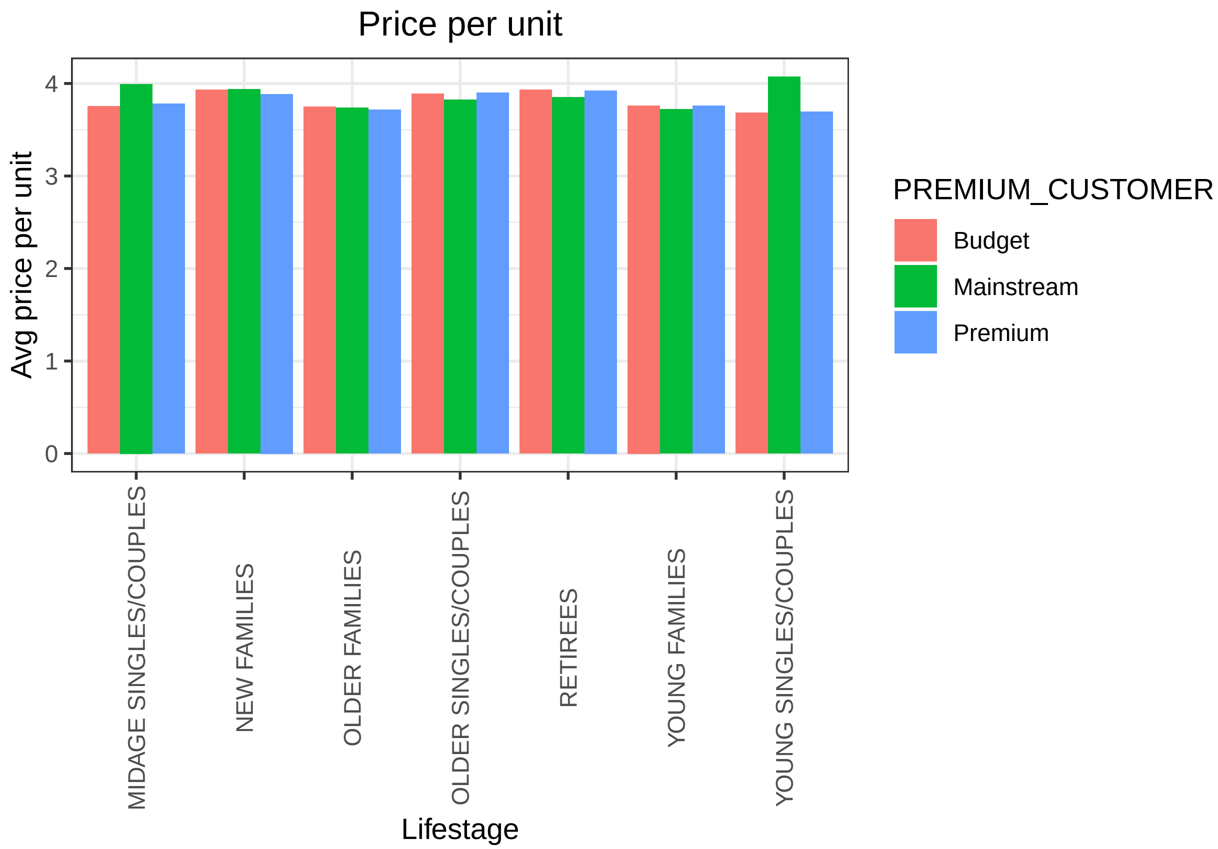
Let's also investigate the average price per unit chips bought for each customer segment as this is also a driver of total sales.

```
pricePerUnit <- data %>% group_by(LIFESTAGE, PREMIUM_CUSTOMER)%>%
  summarise(price_per_unit = (sum(TOT_SALES)/sum(PROD_QTY)))
```

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

Create plot

```r
ggplot(data=pricePerUnit, aes(weight = price_per_unit,x = LIFESTAGE,
                              fill = PREMIUM_CUSTOMER)) +
  geom_bar(position = position_dodge()) +
  labs(x = "Lifestage", y = "Avg price per unit", title = "Price per unit") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```




```r
pricePerUnit[order(pricePerUnit$price_per_unit, decreasing = TRUE), ]
```

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER price_per_unit
##    <chr>                  <chr>                     <dbl>
##  1 YOUNG SINGLES/COUPLES  Mainstream                 4.07
##  2 MIDAGE SINGLES/COUPLES Mainstream                 3.99
##  3 NEW FAMILIES           Mainstream                 3.94
##  4 RETIREES               Budget                     3.93
##  5 NEW FAMILIES           Budget                     3.93
##  6 RETIREES               Premium                    3.92
##  7 OLDER SINGLES/COUPLES  Premium                    3.90
##  8 OLDER SINGLES/COUPLES  Budget                     3.89
##  9 NEW FAMILIES           Premium                    3.89
## 10 RETIREES               Mainstream                 3.85
## # ... with 11 more rows
```

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.

```
#### If this p-value is above .05, then there is not a significant difference in test scores.
data$price <-data$TOT_SALES/data$PROD_QTY # calculate price for each obs from dataset

t1 <-  data$price[data$LIFESTAGE %in% c("YOUNG SINGLES/COUPLES",
                                        "MIDAGE SINGLES/COUPLES") &
                    data$PREMIUM_CUSTOMER == "Mainstream"]
t2 <-  data$price[data$LIFESTAGE %in% c("YOUNG SINGLES/COUPLES",
                                        "MIDAGE SINGLES/COUPLES") &
                    data$PREMIUM_CUSTOMER != "Mainstream"]

t.test(t1, t2, alternative = "greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  t1 and t2
## t = 37.624, df = 54791, p-value < 0.00000000000000022
## 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 2.2e-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 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.

```

segment1 <- subset(data, LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER == "Mainstream")
others <- subset(data, !(LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER == "Mainstream"))

quantity_segment1 <- sum(segment1$PROD_QTY)
quantity_others <- sum(others$PROD_QTY)

quantity_segment1_by_brand <- segment1 %>% group_by(BRAND) %>%
  summarise(targetSegment = sum(PROD_QTY)/quantity_segment1)

quantity_other_by_brand <- others %>% group_by(BRAND) %>%
  summarise(other = sum(PROD_QTY)/quantity_others)

brand_proportions <- merge(quantity_segment1_by_brand, quantity_other_by_brand) %>%
```

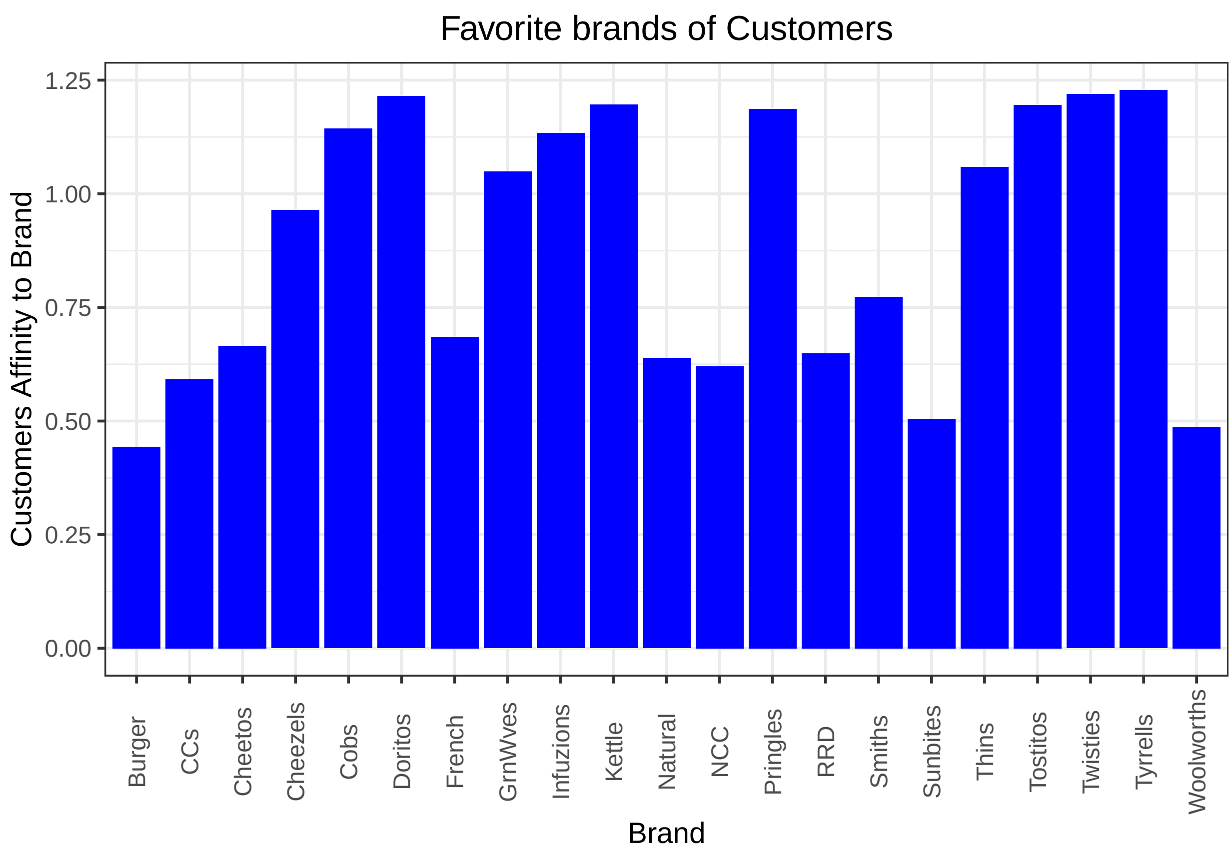
```r
  mutate(affinityToBrand = targetSegment/other)%>% arrange(-affinityToBrand)
brand_proportions
```

Deep dive into Mainstream, young singles/couples

```
##         BRAND targetSegment       other affinityToBrand
## 1    Tyrrells   0.031552795 0.025692464       1.2280953
## 2    Twisties   0.046183575 0.037876520       1.2193194
## 3     Doritos   0.122760524 0.101074684       1.2145526
## 4      Kettle   0.197984817 0.165553442       1.1958967
## 5    Tostitos   0.045410628 0.037977861       1.1957131
## 6    Pringles   0.119420290 0.100634769       1.1866703
## 7        Cobs   0.044637681 0.039048861       1.1431238
## 8   Infuzions   0.064679089 0.057064679       1.1334347
## 9       Thins   0.060372671 0.056986370       1.0594230
## 10    GrnWves   0.032712215 0.031187957       1.0488733
## 11   Cheezels   0.017971014 0.018646902       0.9637534
## 12     Smiths   0.096369910 0.124583692       0.7735355
## 13     French   0.003947550 0.005758060       0.6855694
## 14    Cheetos   0.008033126 0.012066591       0.6657329
## 15        RRD   0.043809524 0.067493678       0.6490908
## 16    Natural   0.015955832 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
```

Plot

```r
ggplot(brand_proportions, aes(BRAND, affinityToBrand)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(x = "Brand", y = "Customers Affinity to Brand",
       title = "Favorite brands of Customers") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



Mainstream young singles/couples are 23% more likely to purchase Tyrrells chips compared to the rest of the population; And, mainstream young singles/couples are 56% less likely to purchase Burger Rings compared to the rest of the population

[INSIGHTS] Let's also find out if our target segment tends to buy larger packs of chips.

```
quantity_segment1_by_pack <- segment1 %>% group_by(PACK_SIZE) %>%
  summarise(targetSegment = sum(PROD_QTY)/quantity_segment1)

quantity_others_by_pack <- others %>% group_by(PACK_SIZE) %>%
  summarise(other = sum(PROD_QTY)/quantity_others)
```

```
pack_proportions <- merge(quantity_segment1_by_pack, quantity_others_by_pack) %>%
  mutate(affinityToBrand = targetSegment/other)%>% arrange(-affinityToBrand)
pack_proportions
```

```
##    PACK_SIZE targetSegment       other affinityToBrand
## 1        270   0.031828847 0.025095929       1.2682873
## 2        380   0.032160110 0.025584213       1.2570295
## 3        330   0.061283644 0.050161917       1.2217166
## 4        134   0.119420290 0.100634769       1.1866703
## 5        110   0.106280193 0.089791190       1.1836372
## 6        210   0.029123533 0.025121265       1.1593180
## 7        135   0.014768806 0.013075403       1.1295106
## 8        250   0.014354727 0.012780590       1.1231662
## 9        170   0.080772947 0.080985964       0.9973697
## 10       150   0.157598344 0.163420656       0.9643722
## 11       175   0.254989648 0.270006956       0.9443818
```

```
## 12        165   0.055652174 0.062267662       0.8937572
## 13        190   0.007481021 0.012442016       0.6012708
## 14        180   0.003588682 0.006066692       0.5915385
## 15        160   0.006404417 0.012372920       0.5176157
## 16         90   0.006349206 0.012580210       0.5046980
## 17        125   0.003008972 0.006036750       0.4984423
## 18        200   0.008971705 0.018656115       0.4808989
## 19         70   0.003036577 0.006322350       0.4802924
## 20        220   0.002926156 0.006596434       0.4435967
```

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

```
data %>% filter(PACK_SIZE == 270) %>% distinct(PROD_NAME)
```

```
##                    PROD_NAME
## 1: Twisties Cheese     270g
## 2:     Twisties Chicken270g
```

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 23% more likely to purchase Tyrrells chips compared to the rest of the population. And, mainstream young singles/couples are 27% more likely to purchase a 270g pack of chips compared to the rest of the population