Quantium Virtual Internship - Retail Strategy and Analytics - Task 1

```
#### Example code to install packages
#install.packages("data.table")
#### Load required libraries
library(data.table)
library(ggplot2)
library(ggmosaic)
```

```
## Warning: package 'ggmosaic' was built under R version 4.4.2
```

```
library(readr)
library(stringr)
library(dplyr)
```

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

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

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

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

```
#### Point the filePath to where you have downloaded the datasets to and
#### assign the data files to data.tables
# over to you! fill in the path to your working directory. If you are on a Windows
↪  machine, you will need to use forward slashes (/) instead of backshashes (\)
setwd("F:/Berkeley/Forage/Quantium_DA/Task_1")
filePath <- "F:/Berkeley/Forage/Quantium_DA/Task_1/"
transactionData <- fread(paste0(filePath,"QVI_transaction_data.csv"))
customerData <- fread(paste0(filePath,"QVI_purchase_behaviour.csv"))
```

Exploratory data analysis

The first step in any analysis is to first understand the data. Let's take a look at each of the datasets provided.

Examining transaction data

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

```
#### Examine transaction data
# Over to you! Examine the data using one or more of the methods described above.
head(transactionData)
```

```
##     DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##    <int>     <int>          <int>  <int>    <int>
## 1: 43390         1           1000      1        5
## 2: 43599         1           1307    348       66
## 3: 43605         1           1343    383       61
## 4: 43329         2           2373    974       69
## 5: 43330         2           2426   1038      108
## 6: 43604         4           4074   2982       57
##                                 PROD_NAME PROD_QTY TOT_SALES
##                                    <char>    <int>     <num>
## 1:   Natural Chip        Compny SeaSalt175g        2       6.0
## 2:                 CCs Nacho Cheese    175g        3       6.3
## 3:   Smiths Crinkle Cut  Chips Chicken 170g        2       2.9
## 4:   Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0
## 5: Kettle Tortilla ChpsHny&Jlpno Chili 150g        3      13.8
## 6: Old El Paso Salsa   Dip Tomato Mild 300g        1       5.1
```

```
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
```

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

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

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

```
#### Examine PROD_NAME
# Over to you! Generate a summary of the PROD_NAME column.
summary.factor(transactionData$PROD_NAME)
```

```
##                         Burger Rings 220g
##                                      1564
##                  CCs Nacho Cheese    175g
##                                      1498
##                         CCs Original 175g
##                                      1514
##                  CCs Tasty Cheese    175g
##                                      1539
##            Cheetos Chs & Bacon Balls 190g
##                                      1479
##                          Cheetos Puffs 165g
##                                      1448
##                       Cheezels Cheese 330g
##                                      3149
##                  Cheezels Cheese Box 125g
##                                      1454
##            Cobs Popd Sea Salt  Chips 110g
##                                      3265
##   Cobs Popd Sour Crm  &Chives Chips 110g
##                                      3159
## Cobs Popd Swt/Chlli &Sr/Cream Chips 110g
##                                      3269
##         Dorito Corn Chp     Supreme 380g
##                                      3185
##         Doritos Cheese      Supreme 330g
##                                      3052
##  Doritos Corn Chip Mexican Jalapeno 150g
##                                      3204
##  Doritos Corn Chip Southern Chicken 150g
##                                      3172
##  Doritos Corn Chips  Cheese Supreme 170g
##                                      3217
##    Doritos Corn Chips  Nacho Cheese 170g
##                                      3160
##        Doritos Corn Chips  Original 170g
##                                      3121
##                 Doritos Mexicana    170g
##                                      3115
##          Doritos Salsa       Medium 300g
##                                      1449
##                 Doritos Salsa Mild  300g
##                                      1472
##           French Fries Potato Chips 175g
##                                      1418
##    Grain Waves         Sweet Chilli 210g
##                                      3167
##    Grain Waves Sour    Cream&Chives 210G
##                                      3105
##    GrnWves Plus Btroot & Chilli Jam 180g
##                                      1468
##  Infuzions BBQ Rib   Prawn Crackers 110g
##                                      3174
##  Infuzions Mango     Chutny Papadums 70g
##                                      1507
```

```
## Infuzions SourCream&Herbs Veg Strws 110g
##                                     3134
## Infuzions Thai SweetChili PotatoMix 110g
##                                     3242
##   Infzns Crn Crnchers Tangy Gcamole 110g
##                                     3144
##             Kettle 135g Swt Pot Sea Salt
##                                     3257
##                       Kettle Chilli 175g
##                                     3038
##         Kettle Honey Soy    Chicken 175g
##                                     3148
##   Kettle Mozzarella   Basil & Pesto 175g
##                                     3304
##                     Kettle Original 175g
##                                     3159
##     Kettle Sea Salt     And Vinegar 175g
##                                     3173
##       Kettle Sensations   BBQ&Maple 150g
##                                     3083
## Kettle Sensations   Camembert & Fig 150g
##                                     3219
##    Kettle Sensations   Siracha Lime 150g
##                                     3127
##  Kettle Sweet Chilli And Sour Cream 175g
##                                     3200
##  Kettle Tortilla ChpsBtroot&Ricotta 150g
##                                     3146
##     Kettle Tortilla ChpsFeta&Garlic 150g
##                                     3138
## Kettle Tortilla ChpsHny&Jlpno Chili 150g
##                                     3296
##   Natural Chip        Compny SeaSalt175g
##                                     1468
##  Natural Chip Co     Tmato Hrb&Spce 175g
##                                     1572
##   Natural ChipCo      Hony Soy Chckn175g
##                                     1460
##   Natural ChipCo Sea  Salt & Vinegr 175g
##                                     1550
##   NCC Sour Cream &    Garden Chives 175g
##                                     1419
## Old El Paso Salsa   Dip Chnky Tom Ht300g
##                                     3125
##  Old El Paso Salsa   Dip Tomato Med 300g
##                                     3114
## Old El Paso Salsa   Dip Tomato Mild 300g
##                                     3085
##                 Pringles Barbeque   134g
##                                     3210
##      Pringles Chicken    Salt Crips 134g
##                                     3104
##         Pringles Mystery    Flavour 134g
##                                     3114
```

```
##          Pringles Original   Crisps 134g
##                                     3157
##                Pringles Slt Vingar 134g
##                                     3095
##           Pringles SourCream  Onion 134g
##                                     3162
##         Pringles Sthrn FriedChicken 134g
##                                     3083
##             Pringles Sweet&Spcy BBQ 134g
##                                     3177
##    Red Rock Deli Chikn&Garlic Aioli 150g
##                                     1434
##  Red Rock Deli Sp    Salt & Truffle 150G
##                                     1498
## Red Rock Deli SR    Salsa & Mzzrlla 150g
##                                     1458
##     Red Rock Deli Thai  Chilli&Lime 150g
##                                     1495
##         RRD Chilli&         Coconut 150g
##                                     1506
##         RRD Honey Soy       Chicken 165g
##                                     1513
##                 RRD Lime & Pepper   165g
##                                     1473
##                 RRD Pc Sea Salt     165g
##                                     1431
##                 RRD Salt & Vinegar  165g
##                                     1474
##      RRD SR Slow Rst     Pork Belly 150g
##                                     1526
##     RRD Steak &         Chimuchurri 150g
##                                     1455
##      RRD Sweet Chilli &  Sour Cream 165g
##                                     1516
##       Smith Crinkle Cut   Bolognese 150g
##                                     1451
##    Smith Crinkle Cut   Mac N Cheese 150g
##                                     1512
##    Smiths Chip Thinly  Cut Original 175g
##                                     1614
##   Smiths Chip Thinly  CutSalt/Vinegr175g
##                                     1440
##   Smiths Chip Thinly  S/Cream&Onion 175g
##                                     1473
##        Smiths Crinkle      Original 330g
##                                     3142
## Smiths Crinkle Chips Salt & Vinegar 330g
##                                     3197
##  Smiths Crinkle Cut  Chips Barbecue 170g
##                                     1489
##   Smiths Crinkle Cut  Chips Chicken 170g
##                                     1484
##  Smiths Crinkle Cut  Chips Chs&Onion170g
##                                     1481
```

```
##  Smiths Crinkle Cut  Chips Original 170g
##                                     1461
## Smiths Crinkle Cut  French OnionDip 150g
##                                     1438
##  Smiths Crinkle Cut  Salt & Vinegar 170g
##                                     1455
##      Smiths Crinkle Cut  Snag&Sauce 150g
##                                     1503
##    Smiths Crinkle Cut  Tomato Salsa 150g
##                                     1470
##   Smiths Crnkle Chip  Orgnl Big Bag 380g
##                                     3233
## Smiths Thinly       Swt Chli&S/Cream175G
##                                     1461
##   Smiths Thinly Cut   Roast Chicken 175g
##                                     1519
##     Snbts Whlgrn Crisps Cheddr&Mstrd 90g
##                                     1576
## Sunbites Whlegrn    Crisps Frch/Onin 90g
##                                     1432
##   Thins Chips         Originl saltd 175g
##                                     1441
##           Thins Chips Light&  Tangy 175g
##                                     3188
##         Thins Chips Salt &  Vinegar 175g
##                                     3103
##         Thins Chips Seasonedchicken 175g
##                                     3114
##     Thins Potato Chips  Hot & Spicy 175g
##                                     3229
##          Tostitos Lightly    Salted 175g
##                                     3074
##        Tostitos Smoked     Chipotle 175g
##                                     3145
##            Tostitos Splash Of  Lime 175g
##                                     3252
##                 Twisties Cheese     270g
##                                     3115
##          Twisties Cheese     Burger 250g
##                                     3169
##                     Twisties Chicken270g
##                                     3170
##   Tyrrells Crisps     Ched & Chives 165g
##                                     3268
##  Tyrrells Crisps     Lightly Salted 165g
##                                     3174
##           Woolworths Cheese   Rings 190g
##                                     1516
##           Woolworths Medium   Salsa 300g
##                                     1430
##           Woolworths Mild     Salsa 300g
##                                     1491
##         WW Crinkle Cut      Chicken 175g
##                                     1467
```

```
##        WW Crinkle Cut      Original 175g
##                                       1410
##        WW D/Style Chip     Sea Salt 200g
##                                       1469
##           WW Original Corn    Chips 200g
##                                       1495
##           WW Original Stacked Chips 160g
##                                       1487
##   WW Sour Cream &OnionStacked Chips 160g
##                                       1483
##      WW Supreme Cheese   Corn Chips 200g
##                                       1509
```

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

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

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

```
# Over to you! Remove digits, and special characters, and then sort the distinct words by
↪  frequency of occurrence.
#### Removing digits
productWords[, DIGIT_PRESENT := grepl("\\d", words)]
productWords <- productWords[DIGIT_PRESENT == FALSE, ][, DIGIT_PRESENT := NULL]
#### Removing special characters
productWords[, SPECIAL_PRESENT := grepl("[^A-Za-z]", words)]
productWords <- productWords[SPECIAL_PRESENT == FALSE, ][, SPECIAL_PRESENT := NULL]
#### 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
uniqueProductWords <- data.table(unlist(unique(productWords[, words])))
setnames(uniqueProductWords, 'words')
uniqueProductWordCount <- c()
for (word in uniqueProductWords$words) {
  uniqueProductWordCount <- c(uniqueProductWordCount, sum(str_count(productWords$words,
↪  word)))
}
uniqueProductWords$Frequency <- uniqueProductWordCount
uniqueProductWords <- uniqueProductWords |> arrange(desc(Frequency))
```

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

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

Next, we can use `summary()` to check summary statistics such as mean, min and max values for each feature to see if there are any obvious outliers in the data and if there are any nulls in any of the columns (`NA's : number of nulls` will appear in the output if there are any nulls).

```
#### Summarise the data to check for nulls and possible outliers
# Over to you!
summary.data.frame(transactionData)
```

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

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

```
#### Filter the dataset to find the outlier
# Over to you! Use a filter to examine the transactions in question.
transactionData |> filter(PROD_QTY == 200)
```

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

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

```
#### Let's see if the customer has had other transactions
# Over to you! Use a filter to see what other transactions that customer made.
cust200ID <- transactionData |> filter(PROD_QTY == 200) |> select(LYLTY_CARD_NBR)
cust200ID <- unique(cust200ID$LYLTY_CARD_NBR)
transactionData |> filter(LYLTY_CARD_NBR == cust200ID)
```

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

```
## 2: 2019-05-20       226         226000 226210        4
##                           PROD_NAME PROD_QTY TOT_SALES
##                              <char>    <int>     <num>
## 1: Dorito Corn Chp     Supreme 380g      200       650
## 2: Dorito Corn Chp     Supreme 380g      200       650
```

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
# Over to you!
transactionData <- transactionData |> filter(LYLTY_CARD_NBR != cust200ID)
#### Re-examine transaction data
# Over to you!
summary.data.frame(transactionData)
```

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

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

```
#### Count the number of transactions by date
# Over to you! Create a summary of transaction count by date.
transactions_by_day <- transactionData |> group_by(DATE) |> summarise(N = n())
transactions_by_day
```

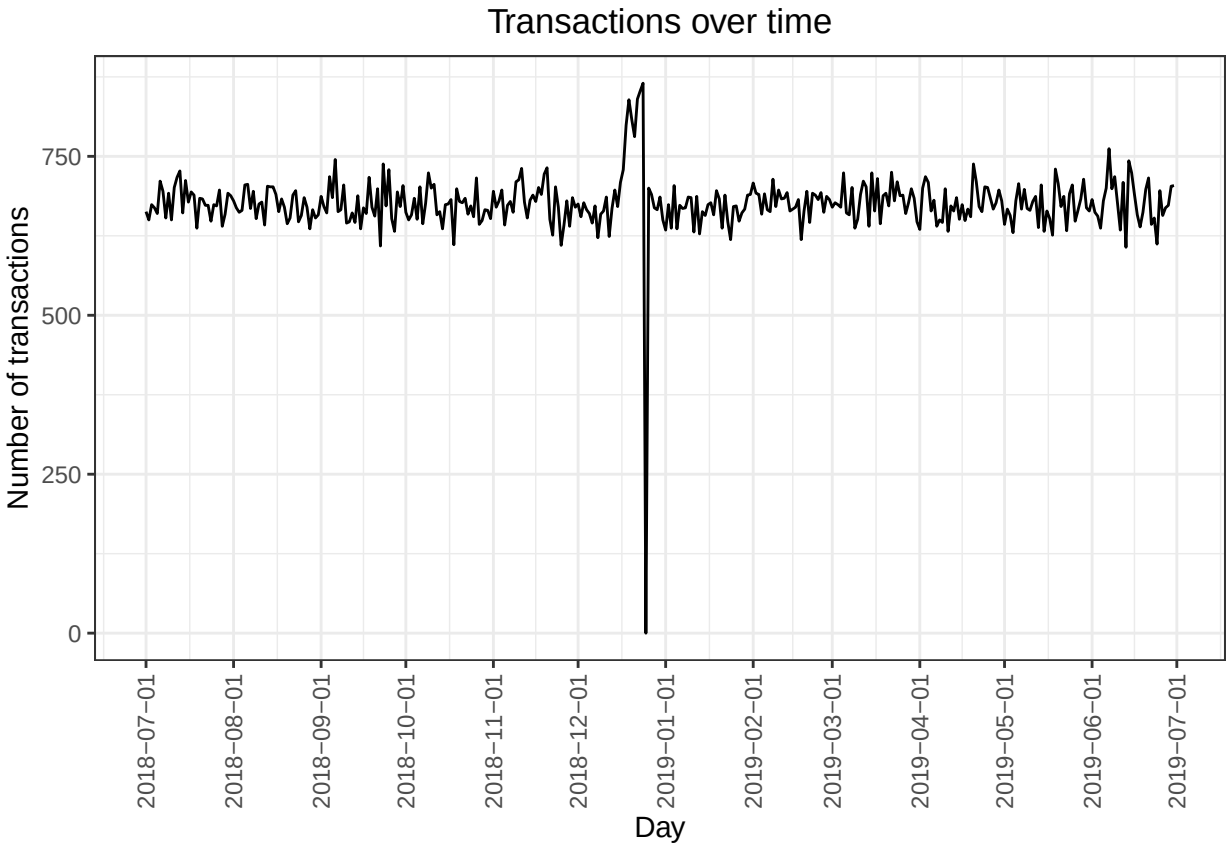
```
## # A tibble: 364 x 2
##    DATE           N
##    <date>     <int>
##  1 2018-07-01   663
##  2 2018-07-02   650
##  3 2018-07-03   674
##  4 2018-07-04   669
##  5 2018-07-05   660
##  6 2018-07-06   711
##  7 2018-07-07   695
##  8 2018-07-08   653
##  9 2018-07-09   692
```

```
## 10 2018-07-10   650
## # i 354 more rows
```

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

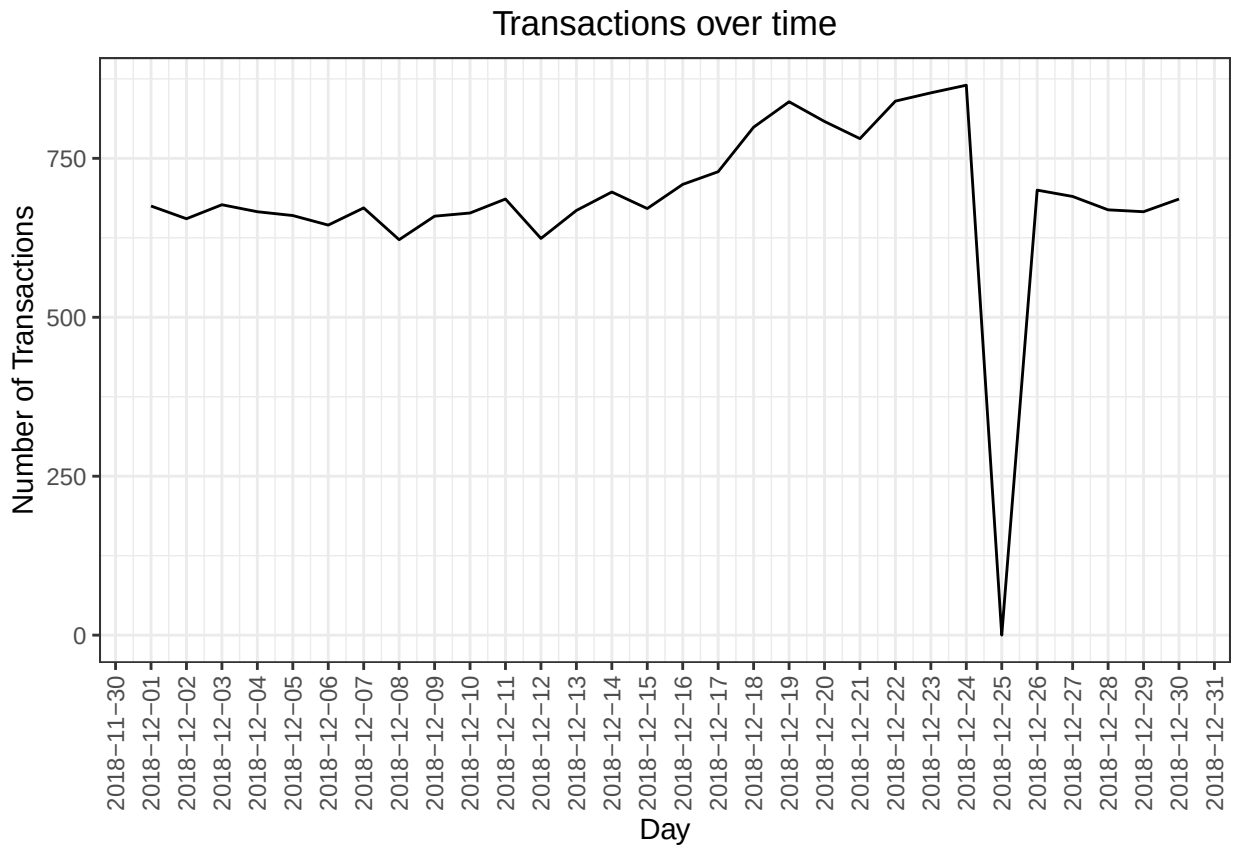
```
#### Create a sequence of dates and join this the count of transactions by date
# Over to you - create a column of dates that includes every day from 1 Jul 2018 to 30
↪  Jun 2019, and join it onto the data to fill in the missing day.
dates <- seq(as.Date("2018-07-01"), as.Date("2019-06-30"), by="day")
for (date in dates) {
  if (as.Date(date) %in% transactions_by_day$DATE){

  }
  else {
    transactions_by_day[nrow(transactions_by_day) + 1, ] <- NA
    transactions_by_day$DATE[nrow(transactions_by_day)] <- as.Date(date)
    transactions_by_day$N[nrow(transactions_by_day)] <- 0
  }
}
#### Setting plot themes to format graphs
theme_set(theme_bw())
theme_update(plot.title = element_text(hjust = 0.5))
#### Plot transactions over time
ggplot(transactions_by_day, aes(x = DATE, y = N)) +
geom_line() +
labs(x = "Day", y = "Number of transactions", title = "Transactions over time") +
scale_x_date(breaks = "1 month") +
theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



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

```
#### Filter to December and look at individual days
# Over to you - recreate the chart above zoomed in to the relevant dates.
decemberTransactions <- transactions_by_day |> filter(DATE >= as.Date("2018-12-01"), DATE
↪  <= as.Date("2018-12-30"))
ggplot(decemberTransactions, aes(x = DATE, y = 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
transactionData[, PACK_SIZE := parse_number(PROD_NAME)]
```

```
## Warning in `[.data.table`(transactionData, , `:=`(PACK_SIZE,
## parse_number(PROD_NAME))): Invalid .internal.selfref detected and fixed by
## taking a (shallow) copy of the data.table so that := can add this new column by
## reference. At an earlier point, this data.table has been copied by R (or was
## created manually using structure() or similar). Avoid names<- and attr<- which
## in R currently (and oddly) may copy the whole data.table. Use set* syntax
## instead to avoid copying: ?set, ?setnames and ?setattr. If this message doesn't
## help, please report your use case to the data.table issue tracker so the root
## cause can be fixed or this message improved.
```

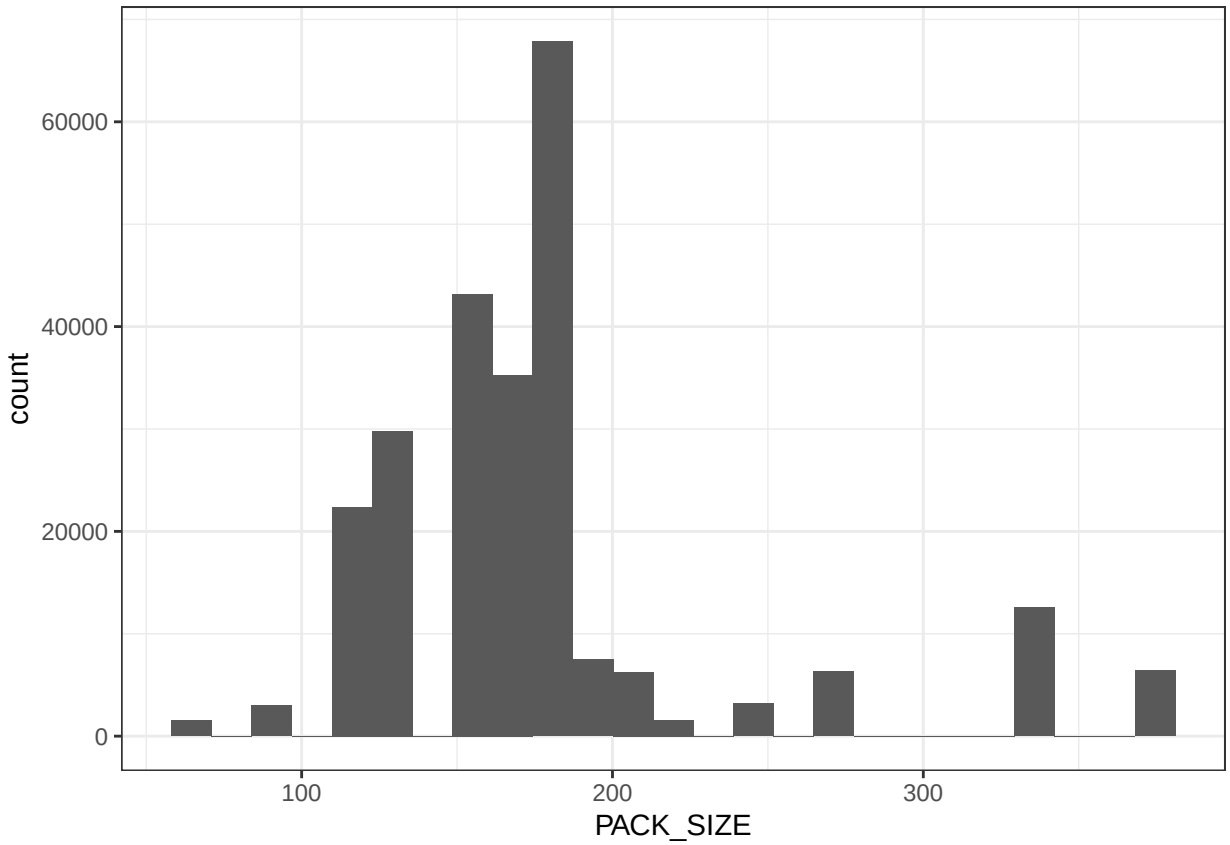
```
#### Always check your output
#### Let's check if the pack sizes look sensible
transactionData[, .N, PACK_SIZE][order(PACK_SIZE)]
```

```
##     PACK_SIZE     N
```

```
##         <num> <int>
##  1:        70  1507
##  2:        90  3008
##  3:       110 22387
##  4:       125  1454
##  5:       134 25102
##  6:       135  3257
##  7:       150 40203
##  8:       160  2970
##  9:       165 15297
## 10:       170 19983
## 11:       175 66390
## 12:       180  1468
## 13:       190  2995
## 14:       200  4473
## 15:       210  6272
## 16:       220  1564
## 17:       250  3169
## 18:       270  6285
## 19:       330 12540
## 20:       380  6416
##     PACK_SIZE     N
```

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

```
#### 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.
# Over to you! Plot a histogram showing the number of transactions by pack size.
ggplot(transactionData, aes(x = PACK_SIZE)) + geom_histogram(bins = 25)
```



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

```
#### Brands
# Over to you! Create a column which contains the brand of the product, by extracting it
↪  from the product name.

transactionData[, BRAND := parse_character(PROD_NAME)]
for (brand in transactionData$BRAND) {
  shortened <- strsplit(brand, " ")[[1]][1]
  transactionData[BRAND == brand, BRAND := shortened]
}
#### Checking brands
# Over to you! Check the results look reasonable.
transactionData[, .N, BRAND][order(BRAND)]
```

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

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

```
#### Clean brand names
transactionData[BRAND == "RED", BRAND := "RRD"]
# Over to you! Add any additional brand adjustments you think may be required.
#transactionData[BRAND == "Grain Waves", BRAND := "GrnWves"]
transactionData[BRAND == "Grain", BRAND := "GrnWves"]
transactionData[BRAND == "Infuzions", BRAND := "Infzns"]
#transactionData[BRAND == "Natural Chip Compny", BRAND := "NCC"]
#transactionData[BRAND == "Natural Chip Co", BRAND := "NCC"]
#transactionData[BRAND == "Natural ChipCo", BRAND := "NCC"]
transactionData[BRAND == "Natural", BRAND := "NCC"]
transactionData[BRAND == "Sunbites", BRAND := "Snbts"]
transactionData[BRAND == "Woolworths", BRAND := "WW"]
transactionData[BRAND == "Doritos", BRAND := "Dorito"]
transactionData[BRAND == "Smiths", BRAND := "Smith"]
#### Check again
# Over to you! Check the results look reasonable.
transactionData[, .N, BRAND][order(BRAND)]
```

```
##        BRAND     N
##       <char> <int>
##  1:   Burger  1564
##  2:      CCs  4551
##  3:  Cheetos  2927
##  4: Cheezels  4603
##  5:     Cobs  9693
##  6:   Dorito 25224
##  7:   French  1418
##  8:  GrnWves  7740
```

```
##  9:   Infzns 14201
## 10:   Kettle 41288
## 11:      NCC  7469
## 12: Pringles 25102
## 13:      RRD 11894
## 14:      Red  4427
## 15:    Smith 30353
## 16:    Snbts  3008
## 17:    Thins 14075
## 18: Tostitos  9471
## 19: Twisties  9454
## 20: Tyrrells  6442
## 21:       WW 11836
##        BRAND     N
```

Examining customer data

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

```
#### Examining customer data
# Over to you! Do some basic summaries of the dataset, including distributions of any key
↪  columns.
print(summary.data.frame(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
```

```
customerData |> group_by(LIFESTAGE, PREMIUM_CUSTOMER) |> summarise(count = n())
```

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

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER count
##    <chr>                  <chr>            <int>
##  1 MIDAGE SINGLES/COUPLES Budget            1504
##  2 MIDAGE SINGLES/COUPLES Mainstream        3340
##  3 MIDAGE SINGLES/COUPLES Premium           2431
##  4 NEW FAMILIES           Budget            1112
##  5 NEW FAMILIES           Mainstream         849
##  6 NEW FAMILIES           Premium            588
##  7 OLDER FAMILIES         Budget            4675
##  8 OLDER FAMILIES         Mainstream        2831
##  9 OLDER FAMILIES         Premium           2274
## 10 OLDER SINGLES/COUPLES  Budget            4929
## # i 11 more rows
```

```
#### 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.

```
# Over to you! See if any transactions did not have a matched customer.
for (transactionLoyalty in transactionData$LYLTY_CARD_NBR) {
  if (transactionLoyalty %in% customerData$LYLTY_CARD_NBR) {

  }
  else {
    print(transactionLoyalty)
  }
}
```

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 on customer segments Now that the data is ready for analysis, we can define some metrics of interest to the client: - Who spends the most on chips (total sales), describing customers by lifestage and how premium their general purchasing behaviour is - How many customers are in each segment- How many chips are bought per customer by segment - What's the average chip price by customer segment We could also ask our data team for more information. Examples are: - The customer's total spend over the period and total spend for each transaction to understand what proportion of their grocery spend is on chips - Proportion of customers in each customer segment overall to compare against the mix of customers who purchase chips

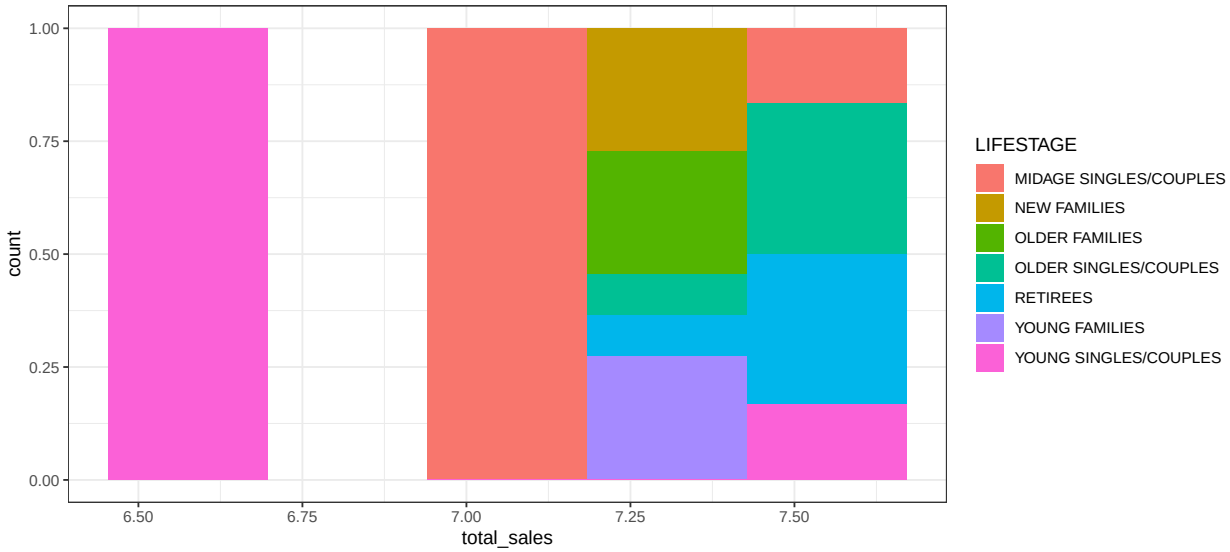
Let's start with calculating total sales by LIFESTAGE and PREMIUM_CUSTOMER and plotting the split by these segments to describe which customer segment contribute most to chip sales.

```
#### Total sales by LIFESTAGE and PREMIUM_CUSTOMER
# Over to you! Calculate the summary of sales by those dimensions and create a  plot.
salesbyPremium_Lifestage <- data |> group_by(LIFESTAGE, PREMIUM_CUSTOMER) |>
↪  summarise(total_sales = mean(TOT_SALES))
```

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

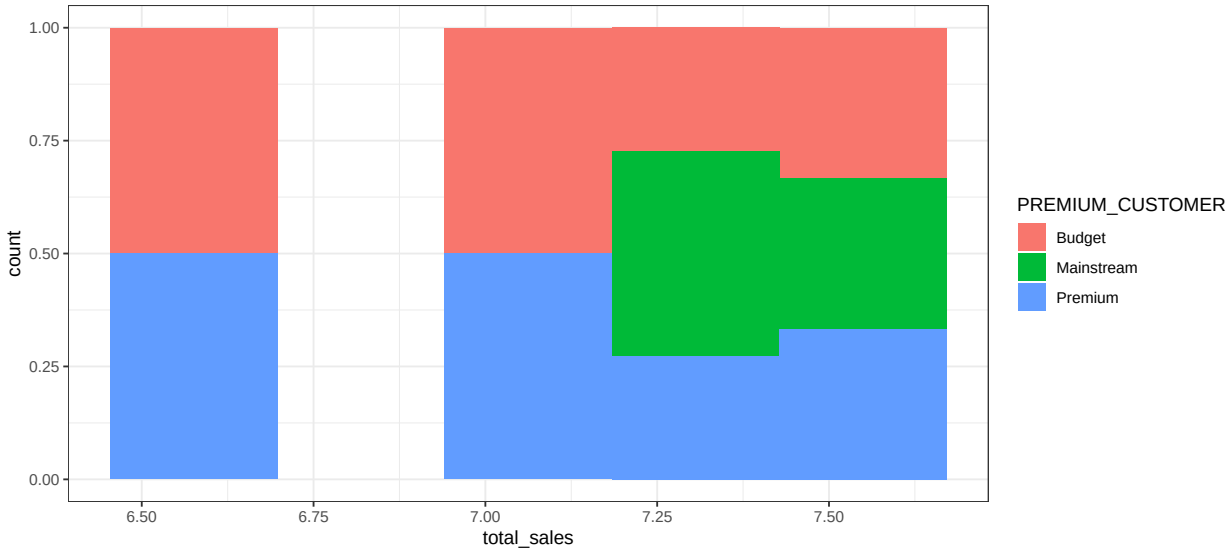
```
ggplot(salesbyPremium_Lifestage, aes(x = total_sales, fill = LIFESTAGE)) +
↪  geom_histogram(position = "fill", bins = 5)
```

```
## Warning: Removed 7 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```




```
ggplot(salesbyPremium_Lifestage, aes(x = total_sales, fill = PREMIUM_CUSTOMER)) +
↪  geom_histogram(position = "fill", bins = 5)
```

```
## Warning: Removed 3 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```

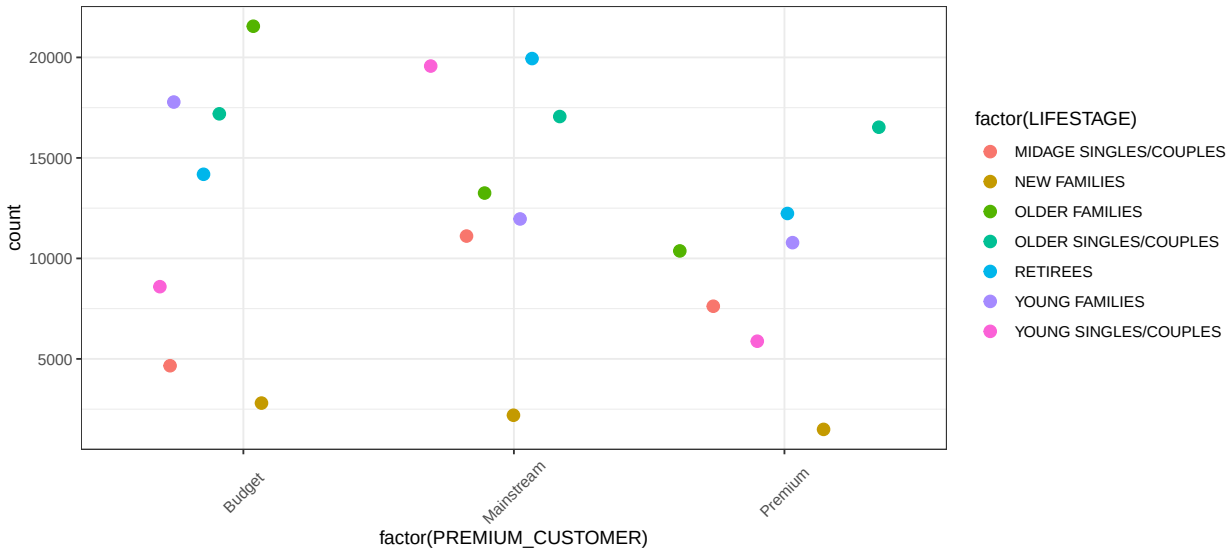


Sales are coming mainly from Budget - older families, Mainstream - young singles/couples, and Mainstream - retirees Let's see if the higher sales are due to there being more customers who buy chips.

```
#### Number of customers by LIFESTAGE and PREMIUM_CUSTOMER
# Over to you! Calculate the summary of number of customers by those dimensions and
↪  create a plot.
customersbyPremium_Lifestage <- data |> group_by(LIFESTAGE, PREMIUM_CUSTOMER) |>
↪  summarise(count = n())
```

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

```
ggplot(customersbyPremium_Lifestage, aes(x = factor(PREMIUM_CUSTOMER), y = count, colour
↪  = factor(LIFESTAGE))) + geom_jitter(size = 3) + theme(axis.text.x =
↪  element_text(angle = 45, vjust = 0.5))
```

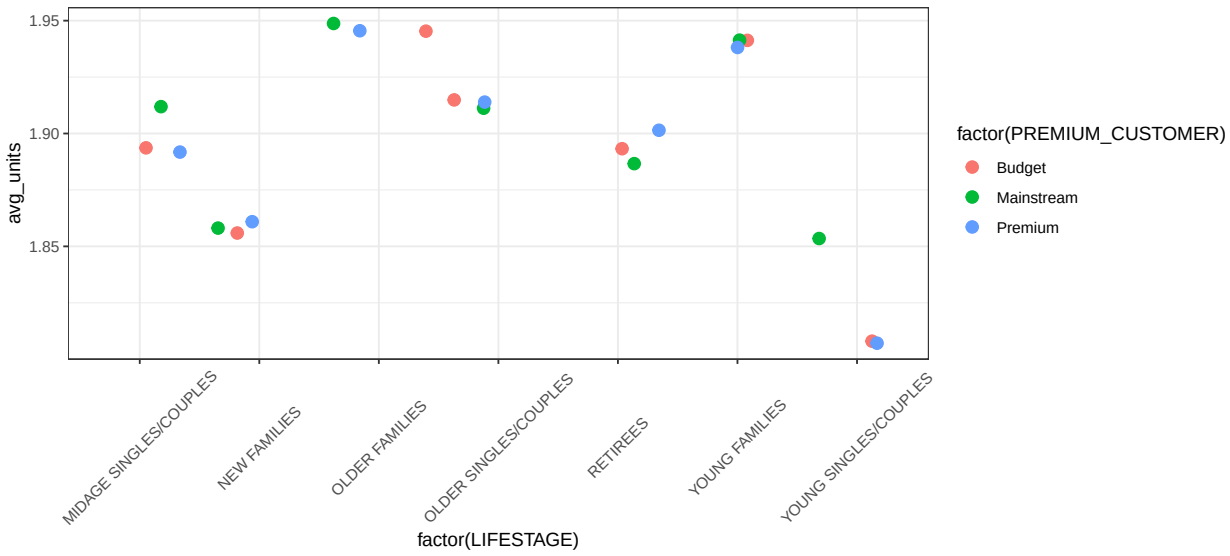


There are more Mainstream - young singles/couples and Mainstream - retirees who buy chips. This contributes to there being more sales to these customer segments but this is not a major driver for the Budget - Older families segment. Higher sales may also be driven by more units of chips being bought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

```
#### Average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER
# Over to you! Calculate and plot the average number of units per customer by those two
↪  dimensions.
avgUnitsbyPremium_Lifestage <- data |> group_by(PREMIUM_CUSTOMER, LIFESTAGE) |>
↪  summarise(avg_units = mean(PROD_QTY))
```

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

```
ggplot(avgUnitsbyPremium_Lifestage, aes(x = factor(LIFESTAGE), y = avg_units, colour =
↪  factor(PREMIUM_CUSTOMER))) + geom_jitter(size = 3) + theme(axis.text.x =
↪  element_text(angle = 45, vjust = 0.5))
```

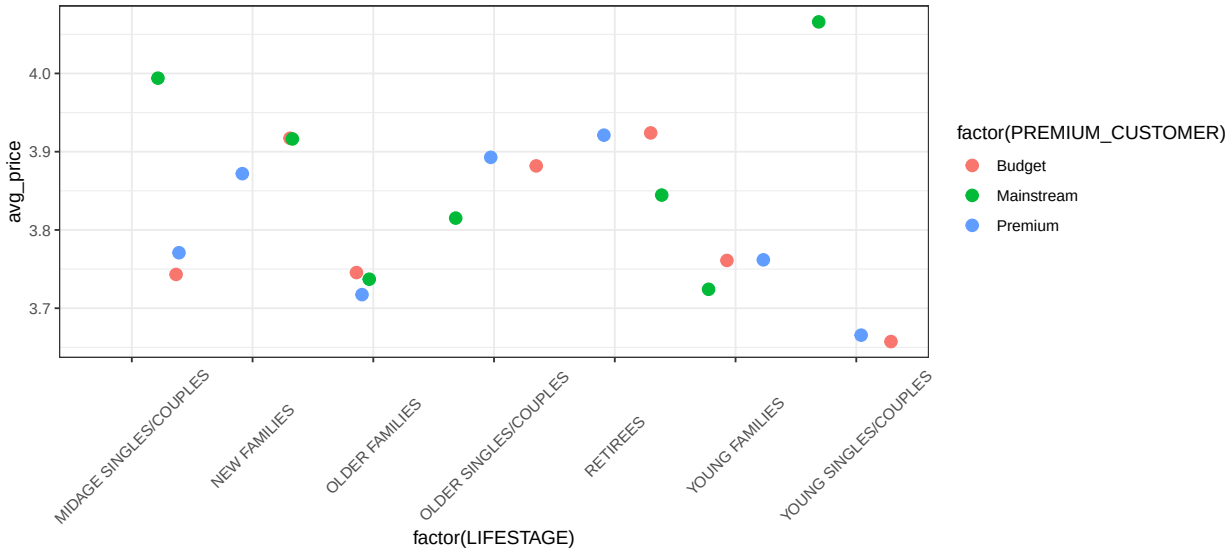


Older families and young families in general buy more chips per customer Let's also investigate the average price per unit chips bought for each customer segment as this is also a driver of total sales.

```
#### Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER
# Over to you! Calculate and plot the average price per unit sold (average sale price) by
↪  those two customer dimensions.
avgPricebyPremium_Lifestage <- data |> group_by(LIFESTAGE, PREMIUM_CUSTOMER) |>
↪  summarise(avg_price = mean(TOT_SALES/PROD_QTY))
```

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

```
ggplot(avgPricebyPremium_Lifestage, aes(x = factor(LIFESTAGE), y = avg_price, colour =
↪  factor(PREMIUM_CUSTOMER))) + geom_jitter(size = 3) + theme(axis.text.x =
↪  element_text(angle = 45, vjust = 0.5))
```



```
avgPricebyPremium_Lifestage
```

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER avg_price
##    <chr>                  <chr>                <dbl>
##  1 MIDAGE SINGLES/COUPLES Budget                3.74
##  2 MIDAGE SINGLES/COUPLES Mainstream            3.99
##  3 MIDAGE SINGLES/COUPLES Premium               3.77
##  4 NEW FAMILIES           Budget                3.92
##  5 NEW FAMILIES           Mainstream            3.92
##  6 NEW FAMILIES           Premium               3.87
##  7 OLDER FAMILIES         Budget                3.75
##  8 OLDER FAMILIES         Mainstream            3.74
##  9 OLDER FAMILIES         Premium               3.72
## 10 OLDER SINGLES/COUPLES  Budget                3.88
## # i 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
# Over to you! Perform a t-test to see if the difference is significant.

justMidandYoung <- avgPricebyPremium_Lifestage |> filter(LIFESTAGE == "MIDAGE
↪  SINGLE/COUPLES" | LIFESTAGE == "YOUNG SINGLES/COUPLES")
justMidandYoung$group <- ifelse(justMidandYoung$PREMIUM_CUSTOMER == "Mainstream",
↪  "Mainstream", "Premium_Budget")

t.test(avg_price ~ group, var.equal = TRUE, data = justMidandYoung)
```

```
##
## Two Sample t-test
##
## data: avg_price by group
## t = 58.005, df = 1, p-value = 0.01097
## alternative hypothesis: true difference in means between group Mainstream
and group Premium_Budget is not equal to 0
## 95 percent confidence interval:
## 0.3156992 0.4928044
## sample estimates:
## mean in group Mainstream mean in group Premium_Budget
## 4.065642 3.661390
```

The t-test results in a p-value of 0.01097, i.e. the unit price for mainstream, young and mid-age singles and couples [ARE] significantly higher than that of budget or premium, young and midage singles and couples.

Deep dive into specific customer segments for insights We have found quite a few interesting insights that we can dive deeper into. We might want to target customer segments that contribute the most to sales to retain them or further increase sales. Let's look at Mainstream - young singles/couples. For instance, let's find out if they tend to buy a particular brand of chips.

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

```
## Warning: package 'arules' was built under R version 4.4.2
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
mainstream_young <- data |> filter((LIFESTAGE == "MIDAGE SINGLE/COUPLES" | LIFESTAGE ==
↪  "YOUNG SINGLES/COUPLES") & PREMIUM_CUSTOMER == "Mainstream")
mainstream_young |> group_by(BRAND) |> summarise(count = n()) |> arrange(desc(count))
```

```
## # A tibble: 21 x 2
##    BRAND    count
##    <chr>    <int>
##  1 Kettle    3844
##  2 Dorito    2379
##  3 Pringles  2315
##  4 Smith     1921
##  5 Infzns    1250
##  6 Thins     1166
##  7 Twisties   900
##  8 Tostitos   890
##  9 Cobs       864
## 10 GrnWves    646
## # i 11 more rows
```

```
transactions <- as(split(mainstream_young$BRAND, seq(nrow(mainstream_young))),
↪  "transactions")
rules <- apriori(transactions, parameter = list(support = 0.1, confidence = 0.8))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5     0.1      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 1954
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[21 item(s), 19544 transaction(s)] done [0.00s].
## sorting and recoding items ... [3 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 done [0.00s].
## writing ... [0 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
head(rules)
```

```
## set of 0 rules
```

We can see that: Kettle and Dorito chips were the most preferred chips [INSIGHTS] 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
# Over to you! Do the same for pack size.
mainstream_young |> group_by(PACK_SIZE) |> summarise(count = n()) |> arrange(desc(count))
```

```
## # A tibble: 20 x 2
##    PACK_SIZE count
##        <dbl> <int>
##  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
```

[INSIGHTS]