Quantium-Task1

2024-02-08

Solution template for Task 1

This file is a solution template for the Task 1 of the Quantium Virtual Internship. It will walk you through the analysis, providing the scaffolding for your solution with gaps left for you to fill in yourself.

Look for comments that say 'over to you' for places where you need to add your own code!

Often, there will be hints about what to do or what function to use in the text leading up to a code block - if you need a bit of extra help on how to use a function, the internet has many excellent resources on R coding, which you can find using your favourite search engine."

Load required libraries and datasets Example code to install packages: install.packages("data.table") install.packages("data.table") install.packages("ggplot2") install.packages("ggmosaic") install.packages("readr")

```
library(data.table)
```

Load required libraries

```
## Warning: package 'data.table' was built under R version 4.2.3
```

```
library(ggplot2)
library(ggmosaic)
library(readr)
```

```
## Warning: package 'readr' was built under R version 4.2.3
```

Point the filePath to where you have downloaded the datasets to and assign the data files to data.tables

```
filePath <- "~/Desktop/Forage/Quantium/"
transactionData <- fread(paste0(filePath,"QVI_transaction_data.csv"))
customerData <- fread(paste0(filePath,"QVI_purchase_behaviour.csv"))
```

Over to you! Fill in the path to your working directory.

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."

```
head(transactionData)
```

Over to you! Examine the data using one or more of the methods described above.

```
##     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
```

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

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

A quick search online tells us that CSV and Excel integer dates begin on 30 Dec 1899 We should check that we are looking at the right products by examining PROD_NAME.

Examine PROD_NAME

```
table(transactionData$PROD_NAME) # We can use table to count.
```

Over to you! Generate a summary of the PROD_NAME column.

```
##
##                         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
```

```
transactionData[, .N, PROD_NAME] # N is the number of rows
```

```
##                                     PROD_NAME     N
##                                        <char> <int>
##   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 summarizing the individual words in the product name.

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

Examine the words in PROD_NAME to see if there are any incorrect entries such as products that are not chips 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()`.

grep(), grepl() – return the indices of strings containing a match (grep()) or a logical vector showing which strings contain a match (grepl()).

Over to you! Remove digits, and special characters, and then sort the distinct words by frequency of occurrence.

```
productWords <- productWords[grepl("\\d", words) == FALSE, ]

# this function says keep words that DO NOT have numbers
# \d is equivalent to [0-9] meaning it matches any number
# \\d is saying match any digits
```

Removing digits

```
productWords <- productWords[grepl("[:alpha:]", words), ]

# [:alpha:] : alphabetic characters, equivalent to [[:lower:][:upper:]] or [A-z]
# this function says keep words that have alphabetic characters
```

Removing special characters

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

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

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

```
# productWords[, .N, words] select rows from column words
# productWords[, .N] returns 323 (number of rows)
```

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

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

Remove salsa products 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(transactionData)
```

Over to you!

```
##       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

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

Over to you! Use a filter to examine the transactions in question.

```
##          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.

customer Loyalty Number is 226000

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

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

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[LYLTY_CARD_NBR != 226000]
```

Show rows that DO NOT match the loyalty card number.

Re-examine transaction data

```
summary(transactionData)
```

Over to you!

```
##       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
```

```
# the new max for PROD_QTY is 5, which is much closer to the other data points!
```

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

```
transaction_by_day <- transactionData[, .N, DATE]
```

Over to you! Create a summary of transaction count by date. 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.

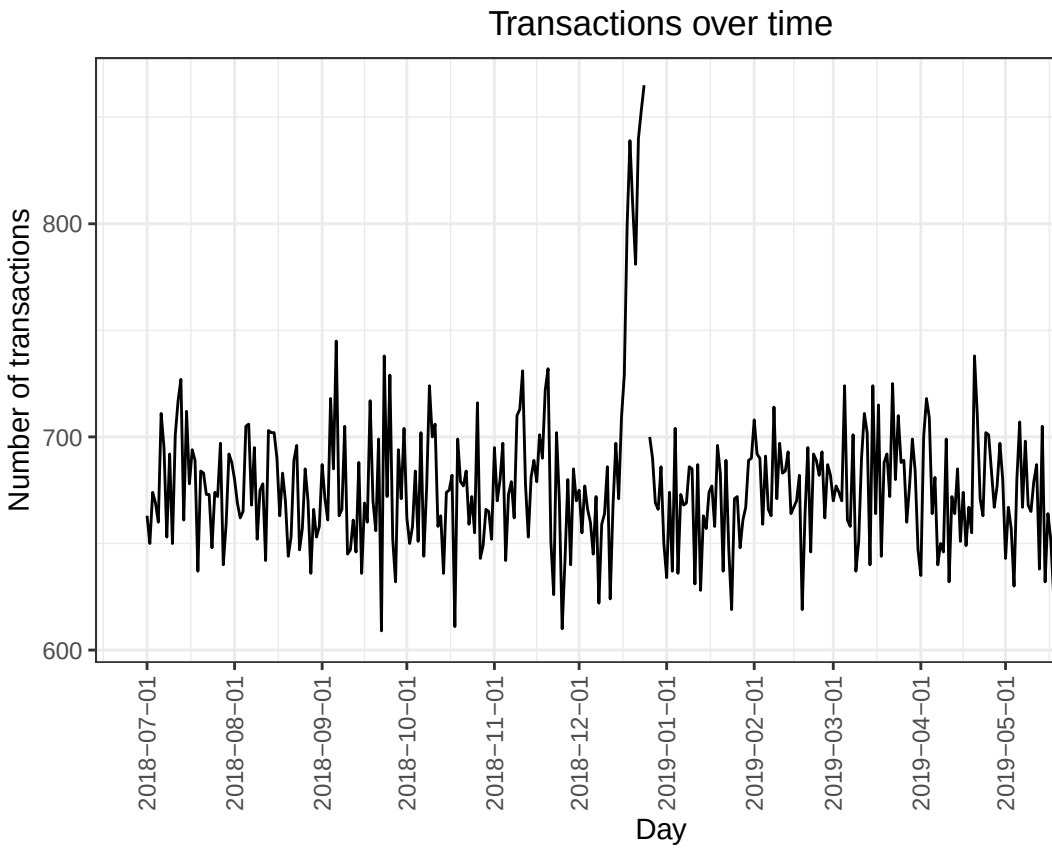
Create a sequence of dates and join this the count of transactions by date

```
# seq(from, to, by)
dates <- data.table(seq(as.Date("2018/07/01"), as.Date("2019/06/30"), by = "day"))
# We need to change the column name to match what's in our table.
# Otherwise we'll see "Error: A non-empty vector of column names for `by` is required."
dates <- setNames(dates, "DATE") # change from V1 to DATE
# Keep all rows from X, but only those from Y that match ("left join")
# Z <- merge(X, Y, all.x = T)
# In our case, we want to keep all rows from daily sequence and only those that match in the table.
transactions_by_day <- merge(dates, transaction_by_day, all.x = T)
```

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.

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

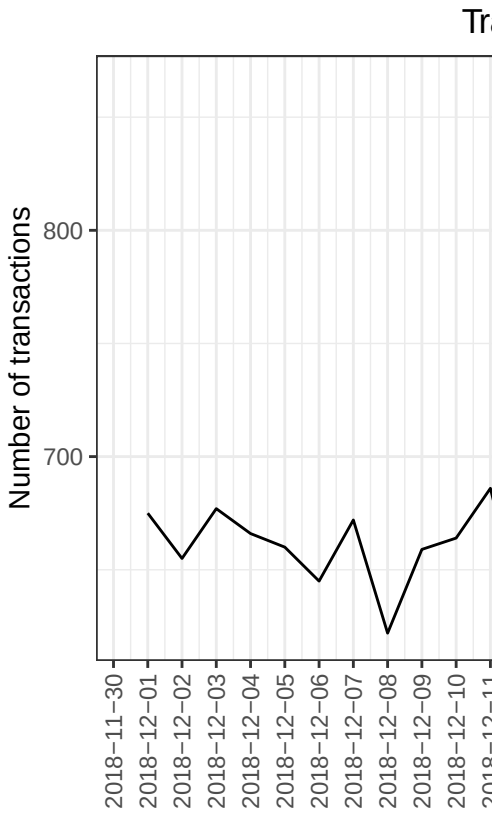
Setting plot themes to format graphs



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



Over to you - recreate the chart above zoomed in to the relevant dates.

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

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

We can work this out by taking the digits that are in PROD_NAME

Always check your output

```
transactionData[, .N, PACK_SIZE][order(PACK_SIZE)]
```

Let's check if the pack sizes look sensible

```
##     PACK_SIZE     N
##         <num> <int>
```

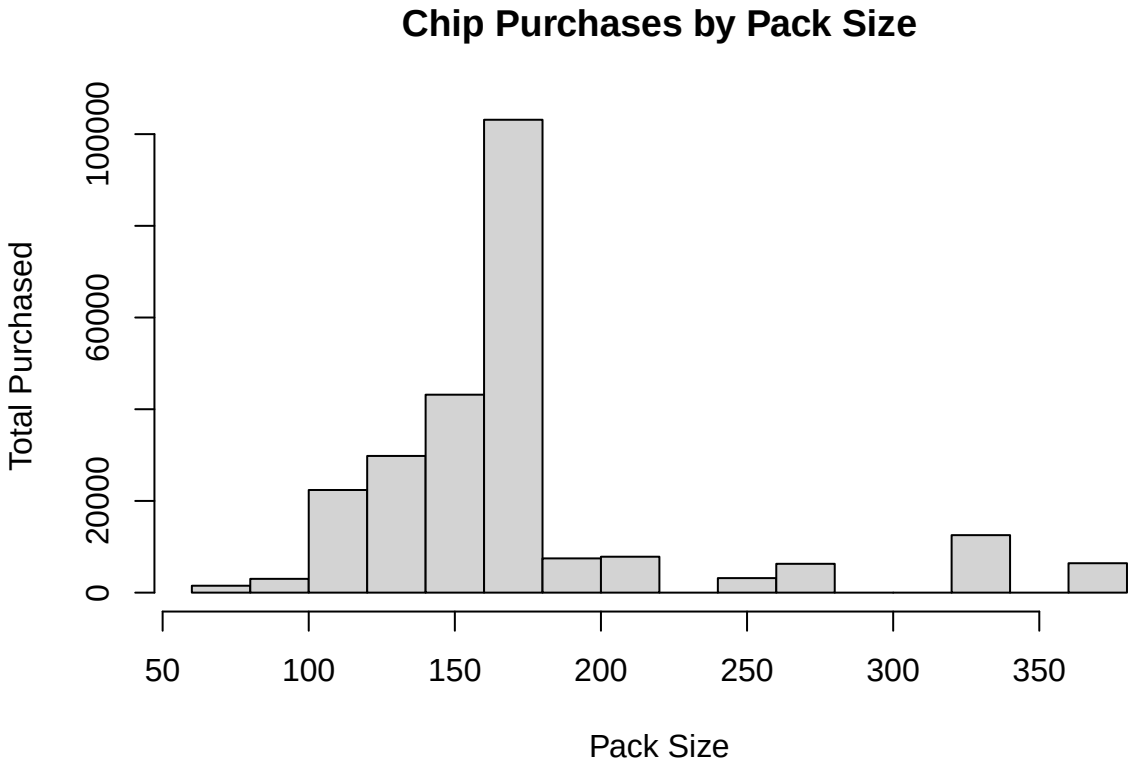
```
##  1:         70  1507
##  2:         90  3008
##  3:        110 22387
##  4:        125  1454
##  5:        134 25102
##  6:        135  3257
##  7:        150 40203
##  8:        160  2970
##  9:        165 15297
## 10:        170 19983
## 11:        175 66390
## 12:        180  1468
## 13:        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.

```
options(scipen=999) # remove scientific notations for 10000s
hist(transactionData$PACK_SIZE, xlab = "Pack Size", ylab = "Total Purchased",
     main = "Chip Purchases by Pack Size")
```

Over to you! Plot a histogram showing the number of transactions by pack size.



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

```
head(transactionData$PROD_NAME)
```

Over to you! Create a column which contains the brand of the product, by extracting it from the product 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] "Smiths Crinkle Chips Salt & Vinegar 330g"
```

```
# This shows us that the brand is usually contained within the first word.
# Brands like Natural ChipCo or Red Rock Deli can also be identified by their first words.
# We want to extract the first word in PROD_NAME.
# You can use a regex ("([A-Za-z]+)" or "([[:alpha:]]+)"or "(\\w+)") to grab the first word
```

```
# Dataframe1$COL2 <- gsub("([A-Za-z]+).*", "\\1", Dataframe1$COL1)
transactionData$BRAND <- gsub("([A-Za-z]+).*", "\\1", transactionData$PROD_NAME)
```

Checking brands

```
unique(transactionData$BRAND)
```

Over to you! Check the results look reasonable.

```
##  [1] "Natural"    "CCs"        "Smiths"     "Kettle"     "Grain"
##  [6] "Doritos"    "Twisties"   "WW"         "Thins"      "Burger"
## [11] "NCC"        "Cheezels"   "Infzns"     "Red"        "Pringles"
## [16] "Dorito"     "Infuzions"  "Smith"      "GrnWves"    "Tyrrells"
## [21] "Cobs"       "French"     "RRD"        "Tostitos"   "Cheetos"
## [26] "Woolworths" "Snbts"      "Sunbites"
```

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.

```
transactionData[BRAND == "RED", BRAND := "RRD"]
```

Clean brand names

```
unique(transactionData$BRAND)
```

Over to you! Add any additional brand adjustments you think may be required.

```
##  [1] "Natural"    "CCs"        "Smiths"     "Kettle"     "Grain"
##  [6] "Doritos"    "Twisties"   "WW"         "Thins"      "Burger"
## [11] "NCC"        "Cheezels"   "Infzns"     "Red"        "Pringles"
## [16] "Dorito"     "Infuzions"  "Smith"      "GrnWves"    "Tyrrells"
## [21] "Cobs"       "French"     "RRD"        "Tostitos"   "Cheetos"
## [26] "Woolworths" "Snbts"      "Sunbites"
```

```
# Snbts and Sunbites
transactionData[BRAND == "Snbts", BRAND := "Sunbites"]

# NCC and Natural Chip Company
transactionData[BRAND == "NCC", BRAND := "Natural"]

# Infzns and Infuzions
transactionData[BRAND == "Infzns", BRAND := "Infuzions"]
```

```
# Smith and Smiths
transactionData[BRAND == "Smith", BRAND := "Smiths"]

# WW and Woolworths
transactionData[BRAND == "WW", BRAND := "Woolworths"]

# Dorito and Doritos
transactionData[BRAND == "Dorito", BRAND := "Doritos"]

# Grain and GRNWVES
transactionData[BRAND == "Grain", BRAND := "GrnWves"]
```

Check again

```
unique(transactionData$BRAND)
```

Over to you! Check the results look reasonable.

```
##  [1] "Natural"    "CCs"        "Smiths"     "Kettle"     "GrnWves"
##  [6] "Doritos"    "Twisties"   "Woolworths" "Thins"      "Burger"
## [11] "Cheezels"   "Infuzions"  "Red"        "Pringles"   "Tyrrells"
## [16] "Cobs"       "French"     "RRD"        "Tostitos"   "Cheetos"
## [21] "Sunbites"
```

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

Examining customer data

```
head(customerData)
```

Over to you! Do some basic summaries of the dataset, including distributions of any key columns.

```
##    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
```

```
# Loyalty Number, Lifestage, Premium/Mainstream/Budget Customer

customerData[, .N, by = LYLTY_CARD_NBR][order(-N)]
```

```
##        LYLTY_CARD_NBR     N
##                 <int> <int>
##     1:           1000     1
##     2:           1002     1
##     3:           1003     1
##     4:           1004     1
##     5:           1005     1
##    ---
## 72633:        2370651     1
## 72634:        2370701     1
## 72635:        2370751     1
## 72636:        2370961     1
## 72637:        2373711     1
```

```
# This shows us that each number is used once, which makes sense.

customerData[, .N, by = LIFESTAGE][order(-N)]
```

```
##                 LIFESTAGE     N
##                    <char> <int>
## 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
```

```
# This shows that RETIREES have the most numbers, followed by OLDER SINGES/COUPLES
# and YOUNG SINGLES/COUPLES. Families seem to have less numbers.

customerData[, .N, by = PREMIUM_CUSTOMER][order(-N)]
```

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

```
# Mainstream customers are the most common, followed by Budget.
```

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

Merge transaction data to customer data As the number of rows in `data` is the same as that of `transactionData`, we can be sure that no duplicates were created. This is because we created `data` by setting `all.x = TRUE` (in other words, a left join) which means take all the rows in `transactionData` and find rows with matching values in shared columns and then joining the details in these rows to the `x` or the first mentioned table.

Let's also check if some customers were not matched on by checking for nulls.

```
# sum(is.na(data)) checks if the entire df has any nulls
# colSums: Form row and column sums and means for numeric arrays (or data frames).
colSums(is.na(data)) # checks all individual columns
```

Over to you! See if any transactions did not have a matched customer.

```
##   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
```

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

```
head(data)
```

Over to you! Calculate the summary of sales by those dimensions and create a plot.

```
## Key: <LYLTY_CARD_NBR>
##    LYLTY_CARD_NBR       DATE STORE_NBR TXN_ID PROD_NBR
##             <int>     <Date>     <int>  <int>    <int>
## 1:           1000 2018-10-17         1      1        5
```

```
## 2:           1002 2018-09-16         1      2       58
## 3:           1003 2019-03-07         1      3       52
## 4:           1003 2019-03-08         1      4      106
## 5:           1004 2018-11-02         1      5       96
## 6:           1005 2018-12-28         1      6       86
##                                 PROD_NAME PROD_QTY TOT_SALES PACK_SIZE
##                                    <char>    <int>     <num>     <num>
## 1: Natural Chip        Compny SeaSalt175g        2       6.0       175
## 2:  Red Rock Deli Chikn&Garlic Aioli 150g        1       2.7       150
## 3:  Grain Waves Sour    Cream&Chives 210G        1       3.6       210
## 4: Natural ChipCo      Hony Soy Chckn175g        1       3.0       175
## 5:         WW Original Stacked Chips 160g        1       1.9       160
## 6:                     Cheetos Puffs 165g        1       2.8       165
##         BRAND               LIFESTAGE PREMIUM_CUSTOMER
##        <char>                  <char>           <char>
## 1:    Natural  YOUNG SINGLES/COUPLES          Premium
## 2:        Red  YOUNG SINGLES/COUPLES       Mainstream
## 3:    GrnWves          YOUNG FAMILIES           Budget
## 4:    Natural          YOUNG FAMILIES           Budget
## 5: Woolworths  OLDER SINGLES/COUPLES       Mainstream
## 6:    Cheetos MIDAGE SINGLES/COUPLES       Mainstream
```

```
# data[rows, columns, by]
sales <- data[, .(SALES = sum(TOT_SALES)), .(LIFESTAGE, PREMIUM_CUSTOMER)]
sales
```

```
##                  LIFESTAGE PREMIUM_CUSTOMER     SALES
##                     <char>           <char>     <num>
##  1:  YOUNG SINGLES/COUPLES          Premium  39052.30
##  2:  YOUNG SINGLES/COUPLES       Mainstream 147582.20
##  3:          YOUNG FAMILIES           Budget 129717.95
##  4:  OLDER SINGLES/COUPLES       Mainstream 124648.50
##  5: MIDAGE SINGLES/COUPLES       Mainstream  84734.25
##  6:  YOUNG SINGLES/COUPLES           Budget  57122.10
##  7:            NEW FAMILIES          Premium  10760.80
##  8:          OLDER FAMILIES       Mainstream  96413.55
##  9:                RETIREES           Budget 105916.30
## 10:  OLDER SINGLES/COUPLES          Premium 123537.55
## 11:          OLDER FAMILIES           Budget 156863.75
## 12: MIDAGE SINGLES/COUPLES          Premium  54443.85
## 13:          OLDER FAMILIES          Premium  75242.60
## 14:                RETIREES       Mainstream 145168.95
## 15:                RETIREES          Premium  91296.65
## 16:          YOUNG FAMILIES       Mainstream  86338.25
## 17: MIDAGE SINGLES/COUPLES           Budget  33345.70
## 18:            NEW FAMILIES       Mainstream  15979.70
## 19:  OLDER SINGLES/COUPLES           Budget 127833.60
## 20:          YOUNG FAMILIES          Premium  78571.70
## 21:            NEW FAMILIES           Budget  20607.45
##                  LIFESTAGE PREMIUM_CUSTOMER     SALES
```

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.

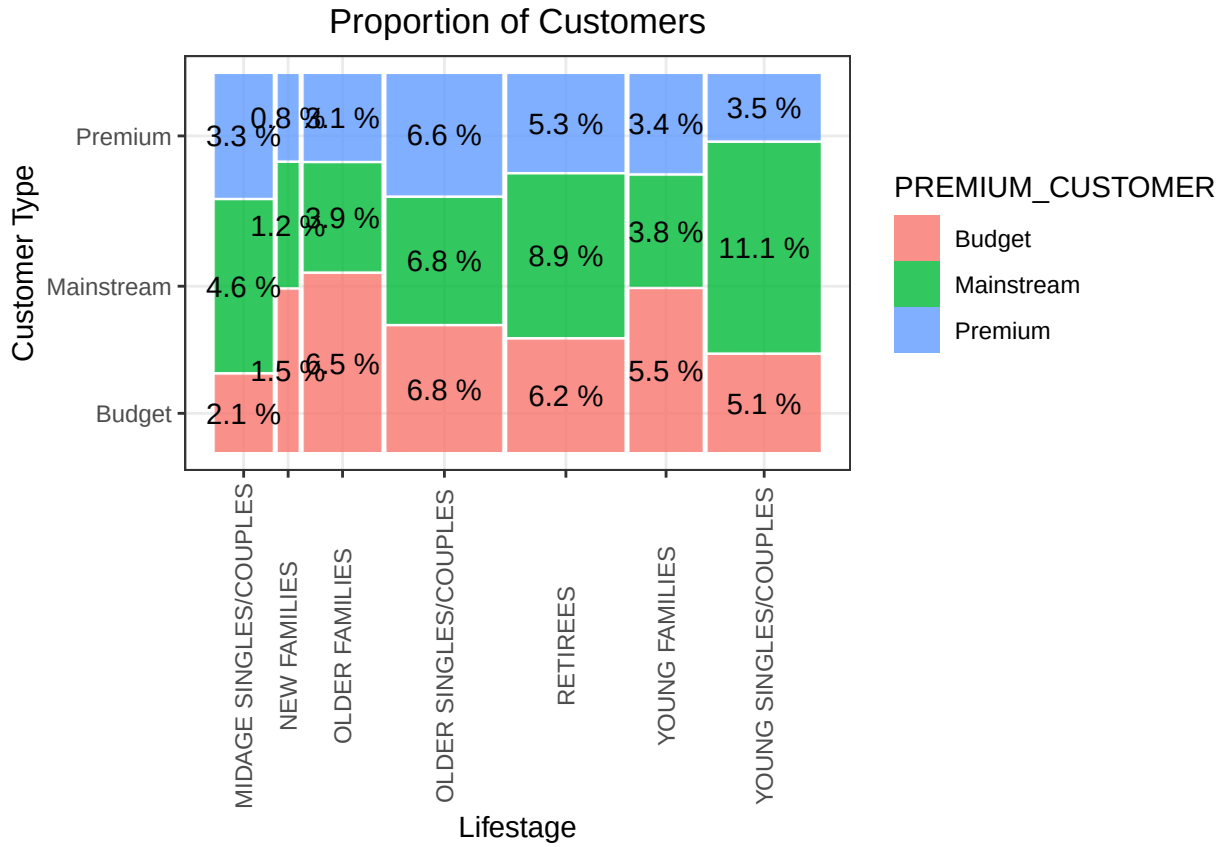
Number of customers by LIFESTAGE and PREMIUM_CUSTOMER

```
customers <- data[, .(CUSTOMERS = uniqueN(LYLTY_CARD_NBR)), .(LIFESTAGE, PREMIUM_CUSTOMER)]
customers
```

Over to you! Calculate the summary of number of customers by those dimensions and create a plot.

```
##                   LIFESTAGE PREMIUM_CUSTOMER CUSTOMERS
##                      <char>           <char>     <int>
##  1:  YOUNG SINGLES/COUPLES          Premium      2480
##  2:  YOUNG SINGLES/COUPLES       Mainstream      7917
##  3:          YOUNG FAMILIES           Budget      3953
##  4:  OLDER SINGLES/COUPLES       Mainstream      4858
##  5: MIDAGE SINGLES/COUPLES       Mainstream      3298
##  6:  YOUNG SINGLES/COUPLES           Budget      3647
##  7:            NEW FAMILIES          Premium       575
##  8:          OLDER FAMILIES       Mainstream      2788
##  9:                RETIREES           Budget      4385
## 10:  OLDER SINGLES/COUPLES          Premium      4682
## 11:          OLDER FAMILIES           Budget      4611
## 12: MIDAGE SINGLES/COUPLES          Premium      2369
## 13:          OLDER FAMILIES          Premium      2231
## 14:                RETIREES       Mainstream      6358
## 15:                RETIREES          Premium      3812
## 16:          YOUNG FAMILIES       Mainstream      2685
## 17: MIDAGE SINGLES/COUPLES           Budget      1474
## 18:            NEW FAMILIES       Mainstream       830
## 19:  OLDER SINGLES/COUPLES           Budget      4849
## 20:          YOUNG FAMILIES          Premium      2398
## 21:            NEW FAMILIES           Budget      1087
##                   LIFESTAGE PREMIUM_CUSTOMER CUSTOMERS
```

```
## Warning: `unite_()` was deprecated in tidyr 1.2.0.
## i Please use `unite()` instead.
## i The deprecated feature was likely used in the ggmosaic package.
##   Please report the issue at <https://github.com/haleyjeppson/ggmosaic>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



There are more Mainstream - young singles/couples (11.1%) and Mainstream - retirees (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 (6.4%). Higher sales may also be driven by more units of chips being bought per customer. Let's have a look at this next.

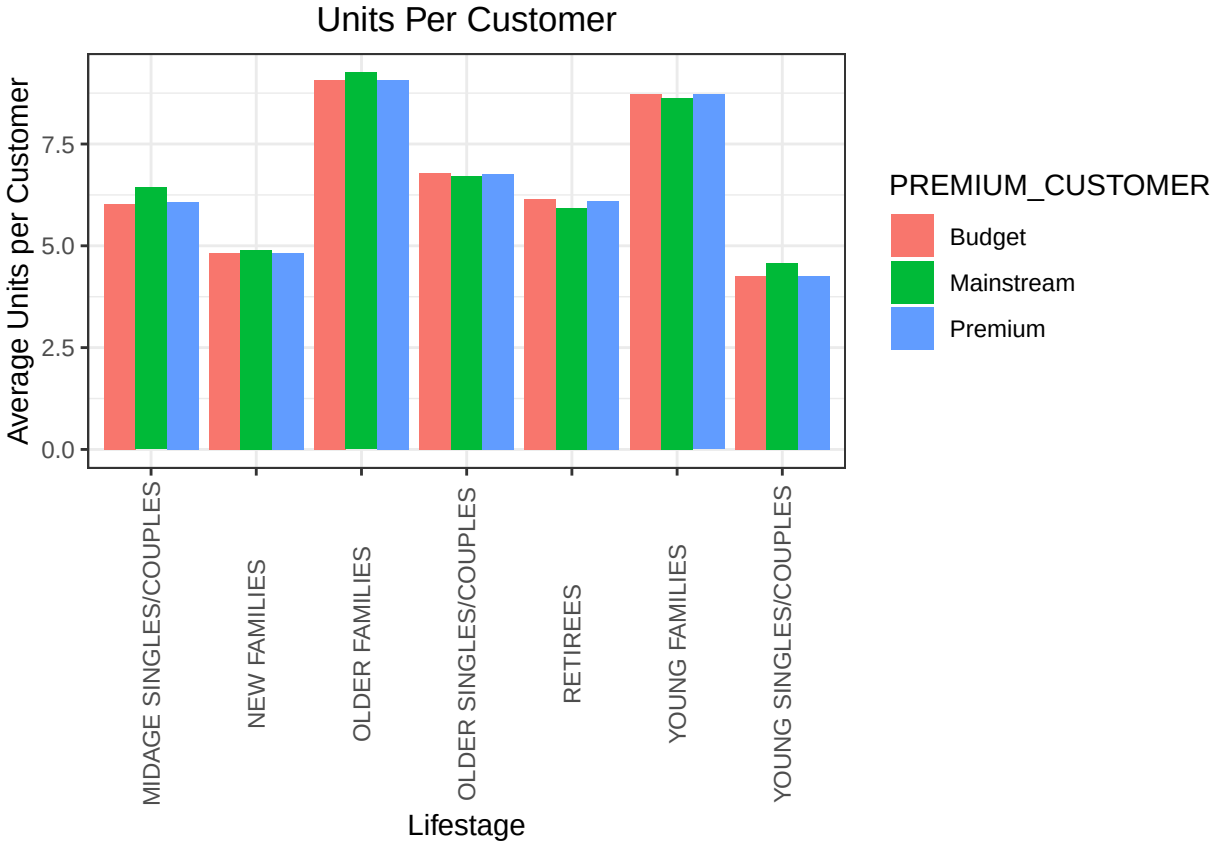
Average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER

```
# data[rows, columns, by]
# [order(-AVERAGE)] would sort avg_units in decreasing order
avg_units <- data[, .(AVERAGE = sum(PROD_QTY)/uniqueN(LYLTY_CARD_NBR)), .(LIFESTAGE, PREMIUM_CUSTOMER)]
avg_units
```

Over to you! Calculate and plot the average number of units per customer by those two dimensions.

```
##                 LIFESTAGE PREMIUM_CUSTOMER  AVERAGE
##                    <char>           <char>    <num>
##  1:  YOUNG SINGLES/COUPLES          Premium 4.264113
##  2:  YOUNG SINGLES/COUPLES       Mainstream 4.575597
##  3:          YOUNG FAMILIES           Budget 8.722995
##  4:  OLDER SINGLES/COUPLES       Mainstream 6.712021
##  5: MIDAGE SINGLES/COUPLES       Mainstream 6.432080
```

```
##  6:  YOUNG SINGLES/COUPLES           Budget 4.250069
##  7:           NEW FAMILIES          Premium 4.815652
##  8:         OLDER FAMILIES       Mainstream 9.255380
##  9:               RETIREES           Budget 6.141847
## 10:  OLDER SINGLES/COUPLES          Premium 6.769543
## 11:         OLDER FAMILIES           Budget 9.076773
## 12: MIDAGE SINGLES/COUPLES          Premium 6.078514
## 13:         OLDER FAMILIES          Premium 9.071717
## 14:               RETIREES       Mainstream 5.925920
## 15:               RETIREES          Premium 6.103358
## 16:         YOUNG FAMILIES       Mainstream 8.638361
## 17: MIDAGE SINGLES/COUPLES           Budget 6.026459
## 18:           NEW FAMILIES       Mainstream 4.891566
## 19:  OLDER SINGLES/COUPLES           Budget 6.781398
## 20:         YOUNG FAMILIES          Premium 8.716013
## 21:           NEW FAMILIES           Budget 4.821527
##                  LIFESTAGE PREMIUM_CUSTOMER  AVERAGE
```



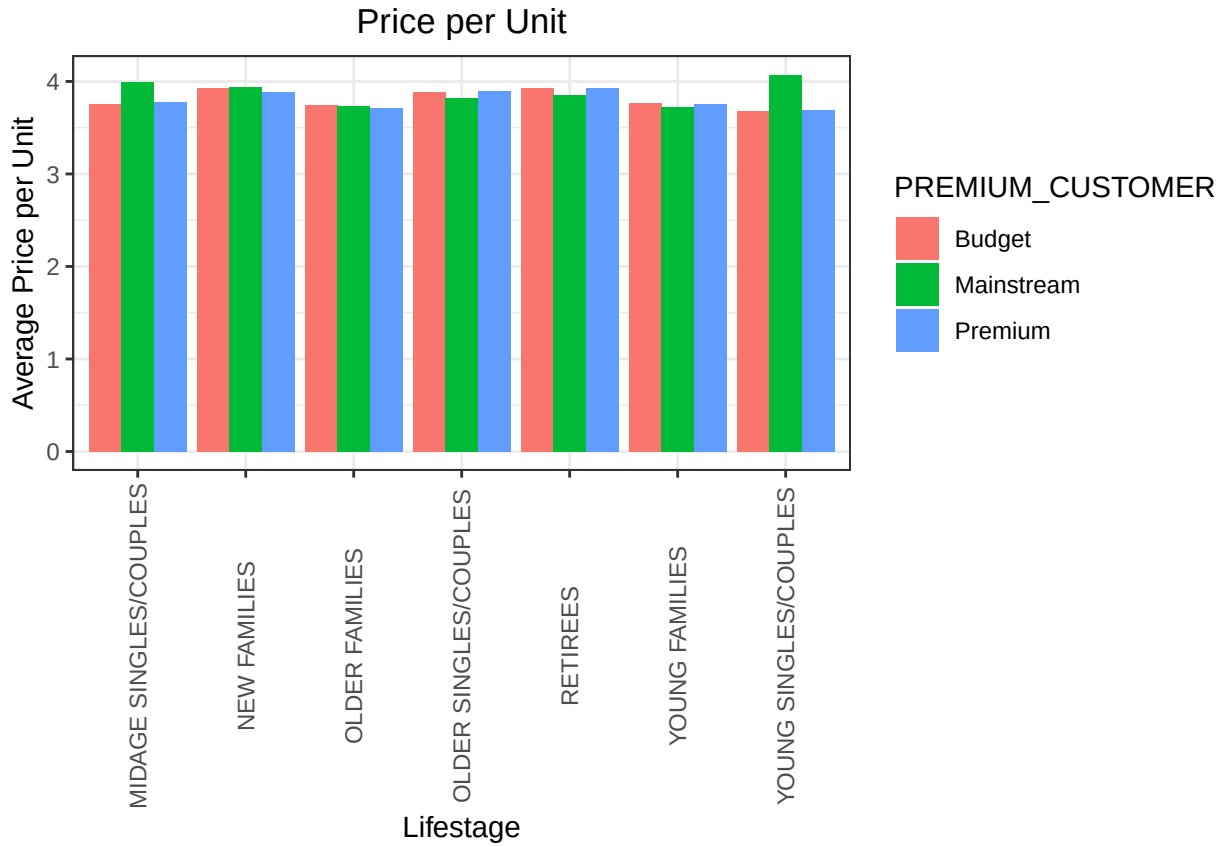
Older families and young families in general buy more chips per customer. Let's also investigate the average price per unit chips bought for each customer segment as this is also a driver of total sales.

Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER

```
# price is TOT_SALES
avg_price <- data[, .(AVERAGE = sum(TOT_SALES)/sum(PROD_QTY)), .(LIFESTAGE, PREMIUM_CUSTOMER)]
avg_price
```

Over to you! Calculate and plot the average price per unit sold (average sale price) by those two customer dimensions.

```
##                 LIFESTAGE PREMIUM_CUSTOMER  AVERAGE
##                    <char>           <char>    <num>
##  1:  YOUNG SINGLES/COUPLES          Premium 3.692889
##  2:  YOUNG SINGLES/COUPLES       Mainstream 4.074043
##  3:         YOUNG FAMILIES           Budget 3.761903
##  4:  OLDER SINGLES/COUPLES       Mainstream 3.822753
##  5: MIDAGE SINGLES/COUPLES       Mainstream 3.994449
##  6:  YOUNG SINGLES/COUPLES           Budget 3.685297
##  7:           NEW FAMILIES          Premium 3.886168
##  8:         OLDER FAMILIES       Mainstream 3.736380
##  9:               RETIREES           Budget 3.932731
## 10:  OLDER SINGLES/COUPLES          Premium 3.897698
## 11:         OLDER FAMILIES           Budget 3.747969
## 12: MIDAGE SINGLES/COUPLES          Premium 3.780823
## 13:         OLDER FAMILIES          Premium 3.717703
## 14:               RETIREES       Mainstream 3.852986
## 15:               RETIREES          Premium 3.924037
## 16:         YOUNG FAMILIES       Mainstream 3.722439
## 17: MIDAGE SINGLES/COUPLES           Budget 3.753878
## 18:           NEW FAMILIES       Mainstream 3.935887
## 19:  OLDER SINGLES/COUPLES           Budget 3.887529
## 20:         YOUNG FAMILIES          Premium 3.759232
## 21:           NEW FAMILIES           Budget 3.931969
##                 LIFESTAGE PREMIUM_CUSTOMER  AVERAGE
```



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.

```
##
##  Welch Two Sample t-test
##
## data:  mainstream_young_mid and premium_budget_young_mid
## 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
```

"Welch Two Sample t-test

data: mainstream_young_mid and premium_budget_young_mid t = 37.624, df = 54791, p-value < 2.2e-16 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 that is < 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."

Deep dive into Mainstream, young singles/couples

```
# arules (apriori) analysis requires our data to be in "transaction" format
# see data(Groceries)

mainstream_ysc <- data[data$PREMIUM_CUSTOMER == "Mainstream" & LIFESTAGE == "YOUNG SINGLES/COUPLES"]
mainstream_ysc
```

Over to you! Work out if 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)

```
## Key: <LYLTY_CARD_NBR>
##        LYLTY_CARD_NBR       DATE STORE_NBR TXN_ID PROD_NBR
##                 <int>     <Date>     <int>  <int>    <int>
##     1:           1002 2018-09-16         1      2       58
##     2:           1010 2018-09-09         1     10       51
##     3:           1018 2018-09-03         1     22        3
##     4:           1018 2018-11-28         1     23       97
##     5:           1018 2019-06-20         1     24       38
##    ---
## 19540:         272391 2018-12-07       272 270205       63
## 19541:        2330041 2018-09-23        77 236718       24
## 19542:        2330321 2018-07-30        77 236756       71
## 19543:        2370181 2018-08-02        88 240146       36
## 19544:        2373711 2018-12-14        88 241815       16
##                                      PROD_NAME PROD_QTY TOT_SALES PACK_SIZE
##                                         <char>    <int>     <num>     <num>
##     1:    Red Rock Deli Chikn&Garlic Aioli 150g        1       2.7       150
##     2:                 Doritos Mexicana    170g        2       8.8       170
##     3: Kettle Sensations   Camembert & Fig 150g        1       4.6       150
##     4:                 RRD Salt & Vinegar  165g        1       3.0       165
##     5:  Infuzions Mango     Chutny Papadums 70g        1       2.4        70
##    ---
## 19540:             Kettle 135g Swt Pot Sea Salt        2       8.4       135
```

```
## 19541:    Grain Waves         Sweet Chilli 210g        2       7.2       210
## 19542:          Twisties Cheese     Burger 250g        2       8.6       250
## 19543:                        Kettle Chilli 175g        2      10.8       175
## 19544: Smiths Crinkle Chips Salt & Vinegar 330g        2      11.4       330
##            BRAND              LIFESTAGE PREMIUM_CUSTOMER price
##           <char>                 <char>           <char> <num>
##     1:       Red YOUNG SINGLES/COUPLES       Mainstream   2.7
##     2:   Doritos YOUNG SINGLES/COUPLES       Mainstream   4.4
##     3:    Kettle YOUNG SINGLES/COUPLES       Mainstream   4.6
##     4:       RRD YOUNG SINGLES/COUPLES       Mainstream   3.0
##     5: Infuzions YOUNG SINGLES/COUPLES       Mainstream   2.4
##    ---
## 19540:    Kettle YOUNG SINGLES/COUPLES       Mainstream   4.2
## 19541:   GrnWves YOUNG SINGLES/COUPLES       Mainstream   3.6
## 19542:  Twisties YOUNG SINGLES/COUPLES       Mainstream   4.3
## 19543:    Kettle YOUNG SINGLES/COUPLES       Mainstream   5.4
## 19544:    Smiths YOUNG SINGLES/COUPLES       Mainstream   5.7
```

```
## Apriori Algorithm ##

# load libraries -- uploaded through Tools > Install
library(arules)
```

```
## Warning: package 'arules' was built under R version 4.2.3
```

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

```
## Warning: package 'Matrix' was built under R version 4.2.3
```

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

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

```
library(arulesViz)
library(RColorBrewer)

# Data Frame: card number & brand
"The part that I struggled to understand was that I wanted to have a DF with a
transaction ID and item, rather than a DF with a list of items."
```

```
## [1] "The part that I struggled to understand was that I wanted to have a DF with a\ntransaction ID ar
```

```
main_t <- mainstream_ysc[, c("LYLTY_CARD_NBR", "BRAND")]

# convert the data to transaction class
transactions_main <- as(split(main_t$BRAND, main_t$LYLTY_CARD_NBR), "transactions")
```

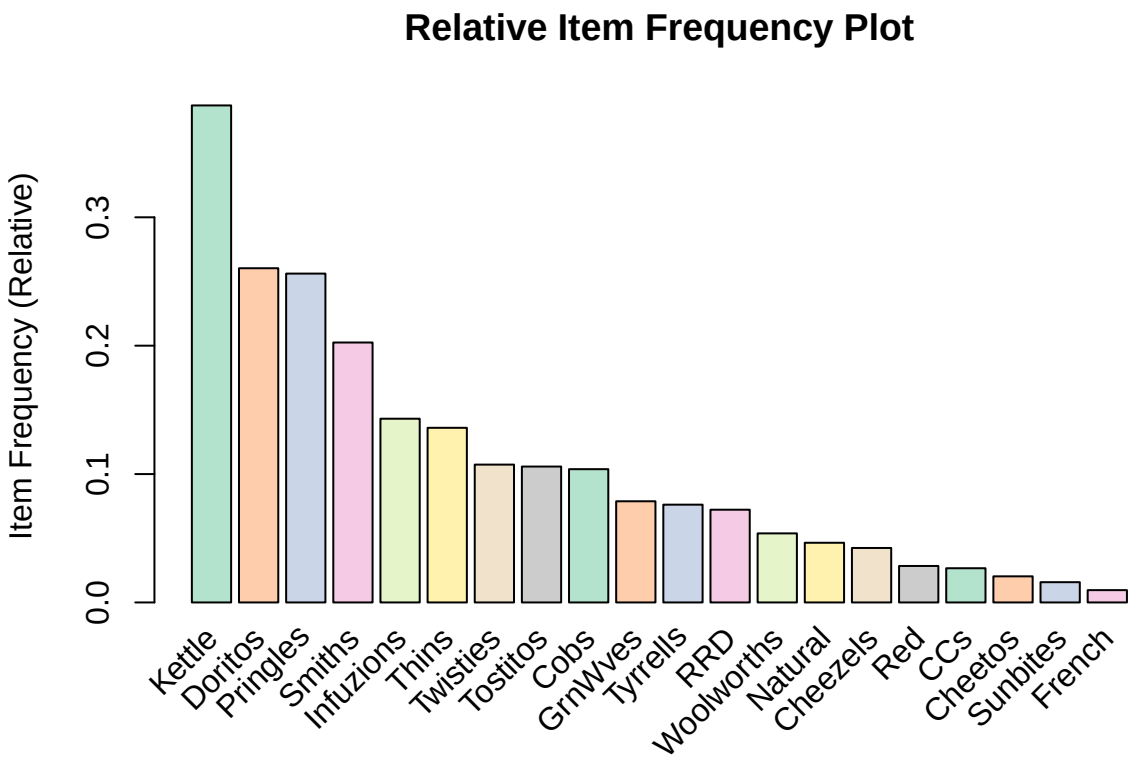
```
## Warning in asMethod(object): removing duplicated items in transactions
```

```r
# using apriori() function
rules_main <- apriori(transactions_main, parameter = list(supp = 0.01, conf = 0.2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.2    0.1    1 none FALSE            TRUE       5    0.01      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: 79
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[21 item(s), 7917 transaction(s)] done [0.00s].
## sorting and recoding items ... [19 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [96 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
# inspect() the first 10 strong associations
inspect(rules_main[1:10])
```

```
##      lhs           rhs        support    confidence coverage   lift      count
## [1]  {}         => {Smiths}   0.20247569 0.2024757  1.00000000 1.0000000 1603
## [2]  {}         => {Pringles} 0.25615764 0.2561576  1.00000000 1.0000000 2028
## [3]  {}         => {Doritos}  0.26032588 0.2603259  1.00000000 1.0000000 2061
## [4]  {}         => {Kettle}   0.38714159 0.3871416  1.00000000 1.0000000 3065
## [5]  {Red}      => {Smiths}   0.01048377 0.3688889  0.02841986 1.8218923   83
## [6]  {Red}      => {Kettle}   0.01086270 0.3822222  0.02841986 0.9872931   86
## [7]  {CCs}      => {Smiths}   0.01035746 0.3886256  0.02665151 1.9193692   82
## [8]  {CCs}      => {Kettle}   0.01035746 0.3886256  0.02665151 1.0038332   82
## [9]  {Cheezels} => {Pringles} 0.01111532 0.2619048  0.04244032 1.0224359   88
## [10] {Cheezels} => {Doritos}  0.01061008 0.2500000  0.04244032 0.9603348   84
```




```
mainstream_ysc[, .(SUM = sum(PROD_QTY)), .(BRAND)][order(-SUM)]
```

What's the brand with the most units purchased?

```
##          BRAND   SUM
##         <char> <int>
##  1:     Kettle  7172
##  2:    Doritos  4447
##  3:   Pringles  4326
##  4:     Smiths  3491
##  5:  Infuzions  2343
##  6:      Thins  2187
##  7:   Twisties  1673
##  8:   Tostitos  1645
##  9:       Cobs  1617
## 10:    GrnWves  1185
## 11:        RRD  1160
## 12:   Tyrrells  1143
## 13: Woolworths   873
## 14:    Natural   710
## 15:   Cheezels   651
## 16:        Red   427
## 17:        CCs   405
```

```
## 18:    Cheetos   291
## 19:   Sunbites   230
## 20:     French   143
## 21:     Burger   106
##          BRAND   SUM
```

Of all of the brands, Kettle is purchased most often.

We can see that for Mainstream, young singles/couples, some association rules are: • If Red is bought, Smiths is also bought. In addition, if Red is bough, Kettle is also bought. • If CCs is bought, Smiths is also bought. Likewise, if CCs is bought, Kettle is also bought. • If Cheezels is bought, Pringles is also bought. Similarly, if Cheezels is bought, Doritos is also bought.

```
others <- data[!(data$PREMIUM_CUSTOMER == "Mainstream" & LIFESTAGE == "YOUNG SINGLES/COUPLES")]
others
```

For safety, let's also do what we did above for everyone else.

```
## Key: <LYLTY_CARD_NBR>
##         LYLTY_CARD_NBR       DATE STORE_NBR TXN_ID PROD_NBR
##                  <int>     <Date>     <int>  <int>    <int>
##      1:           1000 2018-10-17         1      1        5
##      2:           1003 2019-03-07         1      3       52
##      3:           1003 2019-03-08         1      4      106
##      4:           1004 2018-11-02         1      5       96
##      5:           1005 2018-12-28         1      6       86
##     ---
## 227192:        2370581 2018-12-26        88 240318        9
## 227193:        2370651 2018-08-03        88 240350        4
## 227194:        2370701 2018-12-08        88 240378       24
## 227195:        2370751 2018-10-01        88 240394       60
## 227196:        2370961 2018-10-24        88 240480       70
##                                      PROD_NAME PROD_QTY TOT_SALES PACK_SIZE
##                                         <char>    <int>     <num>     <num>
##      1:  Natural Chip        Compny SeaSalt175g        2       6.0       175
##      2:   Grain Waves Sour    Cream&Chives 210G        1       3.6       210
##      3:  Natural ChipCo      Hony Soy Chckn175g        1       3.0       175
##      4:          WW Original Stacked Chips 160g        1       1.9       160
##      5:                      Cheetos Puffs 165g        1       2.8       165
##     ---
## 227192: Kettle Tortilla ChpsBtroot&Ricotta 150g        2       9.2       150
## 227193:        Dorito Corn Chp     Supreme 380g        2      13.0       380
## 227194:   Grain Waves         Sweet Chilli 210g        2       7.2       210
## 227195:    Kettle Tortilla ChpsFeta&Garlic 150g        2       9.2       150
## 227196: Tyrrells Crisps     Lightly Salted 165g        2       8.4       165
##              BRAND              LIFESTAGE PREMIUM_CUSTOMER price
##             <char>                 <char>           <char> <num>
##      1:    Natural  YOUNG SINGLES/COUPLES          Premium   3.0
##      2:    GrnWves          YOUNG FAMILIES           Budget   3.6
##      3:    Natural          YOUNG FAMILIES           Budget   3.0
##      4: Woolworths  OLDER SINGLES/COUPLES       Mainstream   1.9
```

```
##      5:    Cheetos MIDAGE SINGLES/COUPLES       Mainstream   2.8
##     ---
## 227192:     Kettle  OLDER SINGLES/COUPLES           Budget   4.6
## 227193:    Doritos MIDAGE SINGLES/COUPLES       Mainstream   6.5
## 227194:    GrnWves         YOUNG FAMILIES       Mainstream   3.6
## 227195:     Kettle         YOUNG FAMILIES          Premium   4.6
## 227196:   Tyrrells         OLDER FAMILIES           Budget   4.2
```

```
others_t <- others[, c("LYLTY_CARD_NBR", "BRAND")]

transactions_others <- as(split(others_t$BRAND, others_t$LYLTY_CARD_NBR), "transactions")
```

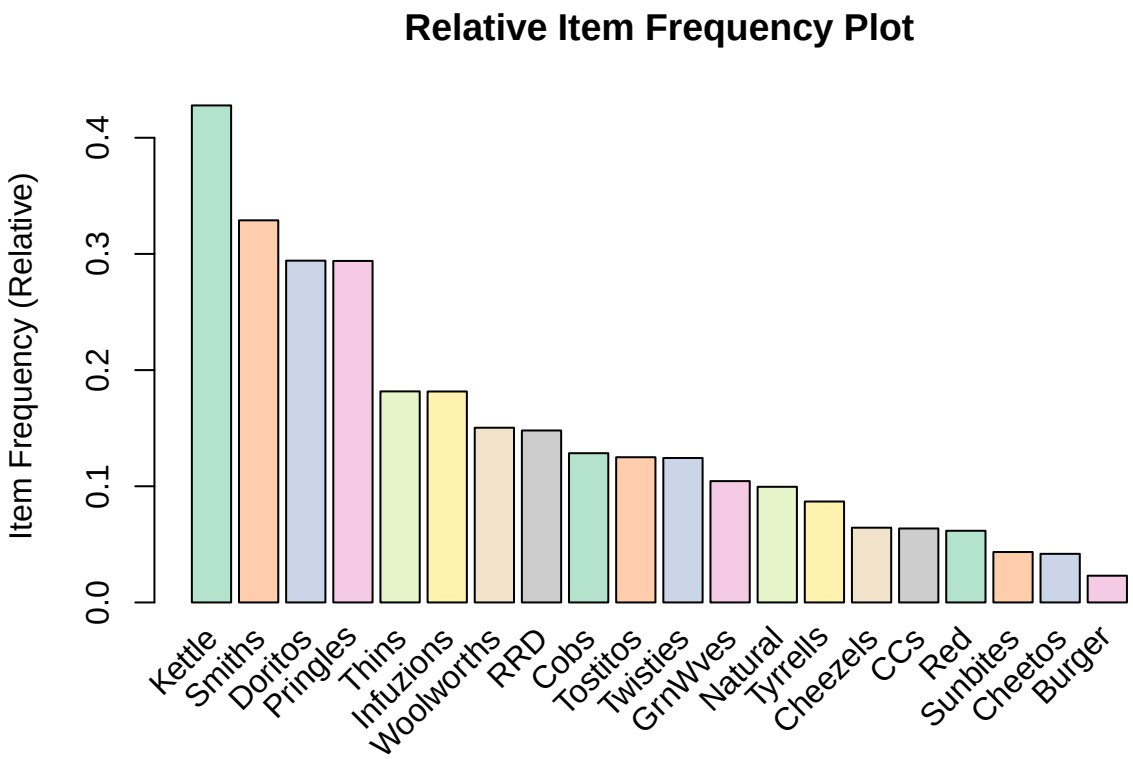
```
## Warning in asMethod(object): removing duplicated items in transactions
```

```
rules_others <- apriori(transactions_others, parameter = list(supp = 0.01, conf = 0.2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.2    0.1    1 none FALSE            TRUE       5    0.01      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: 633
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[21 item(s), 63370 transaction(s)] done [0.01s].
## sorting and recoding items ... [21 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [473 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
inspect(rules_others[1:10])
```

```
##      lhs          rhs          support    confidence coverage   lift     count
## [1]  {}        => {Pringles}   0.29397191 0.2939719  1.00000000 1.000000 18629
## [2]  {}        => {Doritos}    0.29420862 0.2942086  1.00000000 1.000000 18644
## [3]  {}        => {Smiths}     0.32894114 0.3289411  1.00000000 1.000000 20845
## [4]  {}        => {Kettle}     0.42782074 0.4278207  1.00000000 1.000000 27111
## [5]  {French}  => {Smiths}     0.01062017 0.5157088  0.02059334 1.567784   673
## [6]  {Burger}  => {Smiths}     0.01207196 0.5243317  0.02302351 1.593999   765
## [7]  {Burger}  => {Kettle}     0.01028878 0.4468814  0.02302351 1.044553   652
## [8]  {Cheetos} => {RRD}        0.01289254 0.3080694  0.04184946 2.081053   817
## [9]  {Cheetos} => {Woolworths} 0.01270317 0.3035445  0.04184946 2.018428   805
## [10] {Cheetos} => {Pringles}   0.01265583 0.3024133  0.04184946 1.028715   802
```




```
others[, .(SUM = sum(PROD_QTY)), .(BRAND)][order(-SUM)]
```

What's the brand with the most units purchased?

```
##          BRAND   SUM
##         <char> <int>
##  1:     Kettle 71879
##  2:     Smiths 54091
##  3:    Doritos 43884
##  4:   Pringles 43693
##  5:  Infuzions 24776
##  6:      Thins 24742
##  7: Woolworths 21460
##  8:        RRD 21340
##  9:       Cobs 16954
## 10:   Tostitos 16489
## 11:   Twisties 16445
## 12:    GrnWves 13541
## 13:    Natural 13396
## 14:   Tyrrells 11155
## 15:        CCs  8204
## 16:   Cheezels  8096
## 17:        Red  7964
```

```
## 18:   Sunbites  5462
## 19:    Cheetos  5239
## 20:     Burger  2864
## 21:     French  2500
##          BRAND   SUM
```

Once again, Kettle is the brand purchased most often.

For everyone else, some association rules are: • If French is bought, Smiths are bought. • If Burger is bought, Smiths are bought. Kettle is also bought. • If Cheetos are bought, RRD, Woolworths, and Pringles are also bought.

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

```
main_size <- mainstream_ysc[, c("LYLTY_CARD_NBR", "PACK_SIZE")]
main_size
```

Over to you! Do the same for pack size.

```
## Key: <LYLTY_CARD_NBR>
##        LYLTY_CARD_NBR PACK_SIZE
##                 <int>     <num>
##     1:           1002       150
##     2:           1010       170
##     3:           1018       150
##     4:           1018       165
##     5:           1018        70
##    ---
## 19540:         272391       135
## 19541:        2330041       210
## 19542:        2330321       250
## 19543:        2370181       175
## 19544:        2373711       330
```

```
transactions_msize <- as(split(main_size$PACK_SIZE, main_t$LYLTY_CARD_NBR), "transactions")
```

```
## Warning in asMethod(object): removing duplicated items in transactions
```

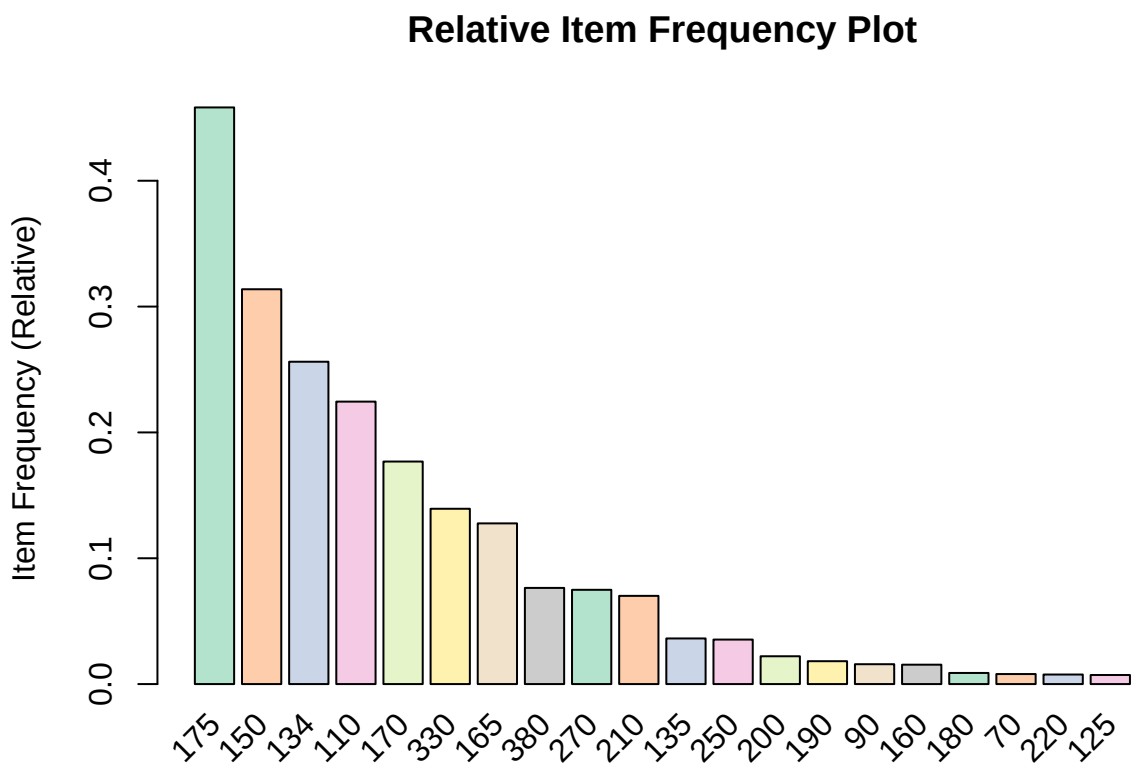
```
rules_msize <- apriori(transactions_msize, parameter = list(supp = 0.01, conf = 0.2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.2    0.1    1 none FALSE            TRUE       5    0.01      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: 79
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[20 item(s), 7917 transaction(s)] done [0.00s].
## sorting and recoding items ... [16 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [115 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
inspect(rules_msize[1:10])
```

```
##      lhs      rhs   support    confidence coverage   lift      count
## [1]  {}    => {110} 0.22445371 0.2244537  1.00000000 1.0000000 1777
## [2]  {}    => {134} 0.25615764 0.2561576  1.00000000 1.0000000 2028
## [3]  {}    => {150} 0.31375521 0.3137552  1.00000000 1.0000000 2484
## [4]  {}    => {175} 0.45825439 0.4582544  1.00000000 1.0000000 3628
## [5]  {200} => {150} 0.01035746 0.4685714  0.02210433 1.4934300   82
## [6]  {200} => {175} 0.01338891 0.6057143  0.02210433 1.3217861  106
## [7]  {250} => {150} 0.01048377 0.2964286  0.03536693 0.9447766   83
## [8]  {250} => {175} 0.01490464 0.4214286  0.03536693 0.9196389  118
## [9]  {135} => {150} 0.01263105 0.3484321  0.03625111 1.1105220  100
## [10] {135} => {175} 0.01692560 0.4668990  0.03625111 1.0188641  134
```




```
mainstream_ysc[, .(COUNT = sum(PROD_QTY)), .(PACK_SIZE)][order(-COUNT)]
```

What's the commonly purchased pack size?

```
##     PACK_SIZE COUNT
##         <num> <int>
##  1:       175  9237
##  2:       150  5709
##  3:       134  4326
##  4:       110  3850
##  5:       170  2926
##  6:       330  2220
##  7:       165  2016
##  8:       380  1165
##  9:       270  1153
## 10:       210  1055
## 11:       135   535
## 12:       250   520
## 13:       200   325
## 14:       190   271
## 15:       160   232
## 16:        90   230
## 17:       180   130
```

```
## 18:        70   110
## 19:       125   109
## 20:       220   106
##     PACK_SIZE COUNT
```

```
mainstream_ysc[PACK_SIZE == 175, unique(PROD_NAME)]
```

```
##  [1] "Kettle Original 175g"
##  [2] "Natural Chip Co     Tmato Hrb&Spce 175g"
##  [3] "Kettle Sweet Chilli And Sour Cream 175g"
##  [4] "Smiths Chip Thinly  CutSalt/Vinegr175g"
##  [5] "Thins Chips         Originl saltd 175g"
##  [6] "Natural ChipCo      Hony Soy Chckn175g"
##  [7] "Kettle Chilli 175g"
##  [8] "Tostitos Splash Of  Lime 175g"
##  [9] "Thins Chips Light&  Tangy 175g"
## [10] "WW Crinkle Cut      Original 175g"
## [11] "Tostitos Lightly    Salted 175g"
## [12] "Tostitos Smoked     Chipotle 175g"
## [13] "Kettle Honey Soy    Chicken 175g"
## [14] "Natural Chip        Compny SeaSalt175g"
## [15] "Smiths Thinly Cut   Roast Chicken 175g"
## [16] "Kettle Mozzarella   Basil & Pesto 175g"
## [17] "Thins Chips Salt &  Vinegar 175g"
## [18] "Kettle Sea Salt     And Vinegar 175g"
## [19] "Thins Chips Seasonedchicken 175g"
## [20] "Thins Potato Chips  Hot & Spicy 175g"
## [21] "French Fries Potato Chips 175g"
## [22] "CCs Tasty Cheese    175g"
## [23] "Smiths Chip Thinly  Cut Original 175g"
## [24] "NCC Sour Cream &    Garden Chives 175g"
## [25] "CCs Nacho Cheese    175g"
## [26] "Natural ChipCo Sea  Salt & Vinegr 175g"
## [27] "Smiths Thinly       Swt Chli&S/Cream175G"
## [28] "Smiths Chip Thinly  S/Cream&Onion 175g"
## [29] "WW Crinkle Cut      Chicken 175g"
## [30] "CCs Original 175g"
```

This tells us that for Mainstream, young singles/couples, 175 is the most common pack size.

Our affinity analysis tells us that some association rules are: • If 200 g bags are purchased, 150 g and 175 g bags are also purchased. • If 250 g bags are purchased, 150 g and 175 g bags are also purchased • If 135 g bags are purchased, 150 g and 175 g bags are also purchased.

One limitation of this affinity analysis was that we didn't factor in the number of packs per brand purchased in a single transaction.

Now, let's check this information for everyone else.

```
others_size <- others[, c("LYLTY_CARD_NBR", "PACK_SIZE")]
others_size
```

```
## Key: <LYLTY_CARD_NBR>
##         LYLTY_CARD_NBR PACK_SIZE
```

```
##                   <int>     <num>
##      1:            1000       175
##      2:            1003       210
##      3:            1003       175
##      4:            1004       160
##      5:            1005       165
##     ---
## 227192:         2370581       150
## 227193:         2370651       380
## 227194:         2370701       210
## 227195:         2370751       150
## 227196:         2370961       165
```

```
transactions_osize <- as(split(others_size$PACK_SIZE, others_size$LYLTY_CARD_NBR), "transactions")
```

```
## Warning in asMethod(object): removing duplicated items in transactions
```

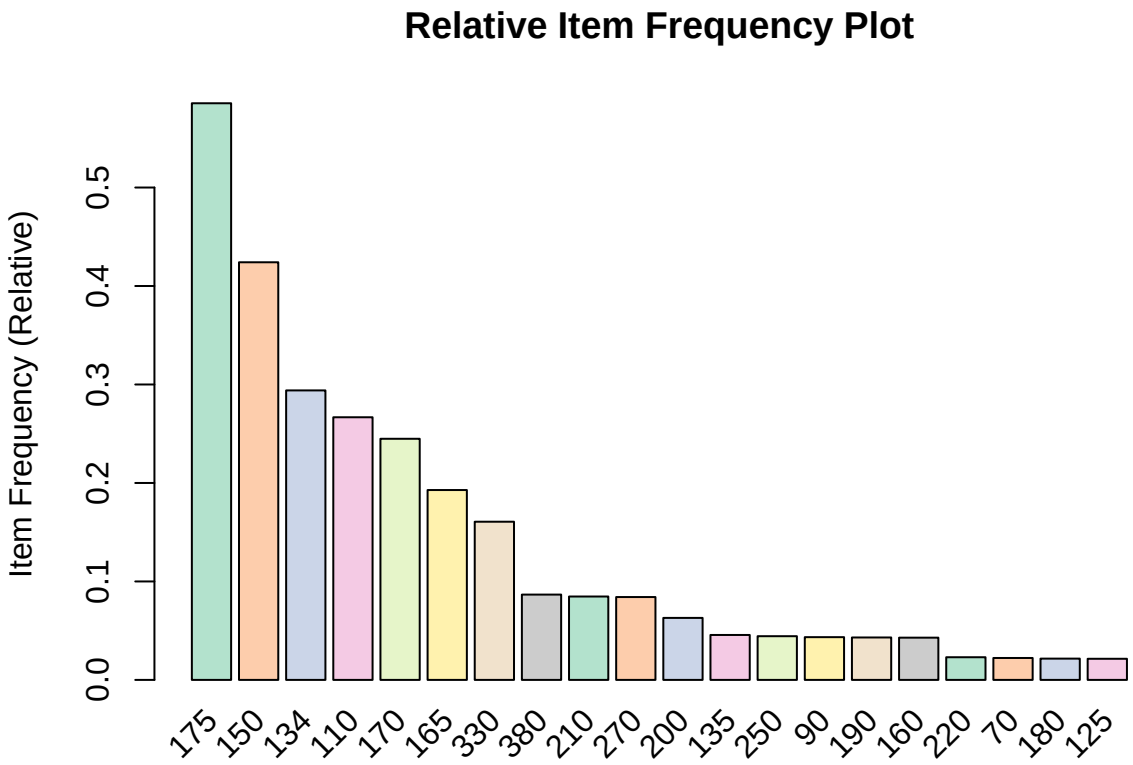
```
rules_osize <- apriori(transactions_osize, parameter = list(supp = 0.01, conf = 0.2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.2    0.1    1 none FALSE            TRUE       5    0.01      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: 633
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[20 item(s), 63370 transaction(s)] done [0.01s].
## sorting and recoding items ... [20 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 5 done [0.00s].
## writing ... [406 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
inspect(rules_osize[1:10])
```

```
##      lhs      rhs    support    confidence coverage   lift     count
## [1]  {}    => {170} 0.24484772 0.2448477  1.00000000 1.000000 15516
## [2]  {}    => {110} 0.26670349 0.2667035  1.00000000 1.000000 16901
## [3]  {}    => {134} 0.29397191 0.2939719  1.00000000 1.000000 18629
## [4]  {}    => {150} 0.42409658 0.4240966  1.00000000 1.000000 26875
## [5]  {}    => {175} 0.58557677 0.5855768  1.00000000 1.000000 37108
## [6]  {125} => {150} 0.01121982 0.5231788  0.02144548 1.233631   711
## [7]  {125} => {175} 0.01456525 0.6791759  0.02144548 1.159841   923
## [8]  {180} => {150} 0.01194572 0.5525547  0.02161906 1.302898   757
```

```
## [9]  {180} => {175} 0.01492820 0.6905109  0.02161906 1.179198   946
## [10] {70}  => {150} 0.01177213 0.5287030  0.02226606 1.246657   746
```




```
others[, .(COUNT = sum(PROD_QTY)), .(PACK_SIZE)][order(-COUNT)]
```

What's the commonly purchased pack size?

```
##     PACK_SIZE  COUNT
##         <num>  <int>
##  1:       175 117230
##  2:       150  70953
##  3:       134  43693
##  4:       110  38985
##  5:       170  35162
##  6:       165  27035
##  7:       330  21779
##  8:       380  11108
##  9:       210  10907
## 10:       270  10896
## 11:       200   8100
## 12:       135   5677
## 13:       250   5549
```

```
## 14:        90   5462
## 15:       190   5402
## 16:       160   5372
## 17:       220   2864
## 18:        70   2745
## 19:       180   2634
## 20:       125   2621
##     PACK_SIZE  COUNT
```

```
others[PACK_SIZE == 175, unique(PROD_NAME)]
```

```
##  [1] "Natural Chip        Compny SeaSalt175g"
##  [2] "Natural ChipCo      Hony Soy Chckn175g"
##  [3] "CCs Tasty Cheese    175g"
##  [4] "Tostitos Splash Of  Lime 175g"
##  [5] "Smiths Chip Thinly  S/Cream&Onion 175g"
##  [6] "Natural ChipCo Sea  Salt & Vinegr 175g"
##  [7] "Natural Chip Co     Tmato Hrb&Spce 175g"
##  [8] "Smiths Thinly       Swt Chli&S/Cream175G"
##  [9] "Thins Chips Seasonedchicken 175g"
## [10] "Smiths Thinly Cut   Roast Chicken 175g"
## [11] "Smiths Chip Thinly  Cut Original 175g"
## [12] "Thins Chips Light&  Tangy 175g"
## [13] "Kettle Chilli 175g"
## [14] "WW Crinkle Cut      Chicken 175g"
## [15] "Smiths Chip Thinly  CutSalt/Vinegr175g"
## [16] "Tostitos Smoked     Chipotle 175g"
## [17] "CCs Nacho Cheese    175g"
## [18] "French Fries Potato Chips 175g"
## [19] "Kettle Mozzarella   Basil & Pesto 175g"
## [20] "CCs Original 175g"
## [21] "Tostitos Lightly    Salted 175g"
## [22] "Kettle Original 175g"
## [23] "Kettle Sea Salt     And Vinegar 175g"
## [24] "WW Crinkle Cut      Original 175g"
## [25] "Thins Chips Salt &  Vinegar 175g"
## [26] "Kettle Sweet Chilli And Sour Cream 175g"
## [27] "Kettle Honey Soy    Chicken 175g"
## [28] "NCC Sour Cream &    Garden Chives 175g"
## [29] "Thins Potato Chips  Hot & Spicy 175g"
## [30] "Thins Chips         Originl saltd 175g"
```

For everyone else, 175 is also the most common pack size.

Our affinity analysis tells us that some association rules are: • If 125 g bags are purchased, 150 g and 175 g bags are also purchased. • If 180 g bags are purchased, 150 g and 175 g bags are also purchased • If 70 g bags are purchased, 150 g bags are also purchased.

One limitation of this affinity analysis was that we didn't factor in the number of packs per brand purchased in a single transaction. However, note that it appears that Mainstream young singles/couples are more likely to buy larger pack sizes compared to the rest of the population.