– title: “Quantium Virtual Internship - Retail Strategy and Analytics - Task 1” mainfont: Roboto monofont: Consolas output: pdf_document: df_print: default highlight: tango keep_tex: yes latex_engine: xelatex header-includes:
–

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.

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("ggmosaic")
#install.packages("readr")

#### Load required libraries
library(data.table)
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
```

```
library(ggplot2)
library(ggmosaic)
library(readr)
library(readxl)
library(stringr)
library(tidyr)

#file.choose()
transaction <- read_excel("D:\\UMP\\Extra Program\\Virtual Internship (Forage)\\Quantium\\Tas
k 1\\QVI_transaction_data.xlsx")
transaction

customer <- read.csv("D:\\UMP\\Extra Program\\Virtual Internship (Forage)\\Quantium\\Task 1
\\QVI_purchase_behaviour.csv")
customer
```

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 `transaction` in the console to see a sample of the data or use `head(transaction )` 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
str(transaction)
```

```
## tibble [264,836 × 8] (S3: tbl_df/tbl/data.frame)
## $ DATE : num [1:264836] 43390 43599 43605 43329 43330 ...
## $ STORE_NBR : num [1:264836] 1 1 1 2 2 4 4 4 5 7 ...
## $ LYLTY_CARD_NBR: num [1:264836] 1000 1307 1343 2373 2426 ...
## $ TXN_ID : num [1:264836] 1 348 383 974 1038 ...
## $ PROD_NBR : num [1:264836] 5 66 61 69 108 57 16 24 42 52 ...
## $ PROD_NAME : chr [1:264836] "Natural Chip Compny SeaSalt175g" "CCs Nacho
Cheese 175g" "Smiths Crinkle Cut Chips Chicken 170g" "Smiths Chip Thinly
S/Cream&Onion 175g" ...
## $ PROD_QTY : num [1:264836] 2 3 2 5 3 1 1 1 1 2 ...
## $ TOT_SALES : num [1:264836] 6 6.3 2.9 15 13.8 5.1 5.7 3.6 3.9 7.2 ...
```

```
head(transaction)
```

```
## # A tibble: 6 × 8
## DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME PROD_QTY TOT_SALES
## <dbl> <dbl> <dbl> <dbl> <dbl> <chr> <dbl> <dbl>
## 1 43390 1 1000 1 5 Natural Chi… 2 6
## 2 43599 1 1307 348 66 CCs Nacho C… 3 6.3
## 3 43605 1 1343 383 61 Smiths Crin… 2 2.9
## 4 43329 2 2373 974 69 Smiths Chip… 5 15
## 5 43330 2 2426 1038 108 Kettle Tort… 3 13.8
## 6 43604 4 4074 2982 57 Old El Paso… 1 5.1
```

```
summary(transaction)
```

```
##       DATE         STORE_NBR     LYLTY_CARD_NBR        TXN_ID
##  Min.   :43282   Min.   :  1.0   Min.   :   1000   Min.   :      1
##  1st Qu.:43373   1st Qu.: 70.0   1st Qu.:  70021   1st Qu.:  67602
##  Median :43464   Median :130.0   Median : 130358   Median : 135138
##  Mean   :43464   Mean   :135.1   Mean   : 135550   Mean   : 135158
##  3rd Qu.:43555   3rd Qu.:203.0   3rd Qu.: 203094   3rd Qu.: 202701
##  Max.   :43646   Max.   :272.0   Max.   :2373711   Max.   :2415841
##     PROD_NBR       PROD_NAME            PROD_QTY         TOT_SALES
##  Min.   :  1.00   Length:264836      Min.   :  1.000   Min.   :  1.500
##  1st Qu.: 28.00   Class :character   1st Qu.:  2.000   1st Qu.:  5.400
##  Median : 56.00   Mode  :character   Median :  2.000   Median :  7.400
##  Mean   : 56.58                      Mean   :  1.907   Mean   :  7.304
##  3rd Qu.: 85.00                      3rd Qu.:  2.000   3rd Qu.:  9.200
##  Max.   :114.00                      Max.   :200.000   Max.   :650.000
```

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

```
## # A tibble: 264,836 × 8
## DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME PROD_QTY
## <date> <dbl> <dbl> <dbl> <dbl> <chr> <dbl>
## 1 2018-10-17 1 1000 1 5 Natural Chip … 2
## 2 2019-05-14 1 1307 348 66 CCs Nacho Chees… 3
## 3 2019-05-20 1 1343 383 61 Smiths Crinkle … 2
## 4 2018-08-17 2 2373 974 69 Smiths Chip Thi… 5
## 5 2018-08-18 2 2426 1038 108 Kettle Tortilla… 3
## 6 2019-05-19 4 4074 2982 57 Old El Paso Sal… 1
## 7 2019-05-16 4 4149 3333 16 Smiths Crinkle … 1
## 8 2019-05-16 4 4196 3539 24 Grain Waves … 1
## 9 2018-08-20 5 5026 4525 42 Doritos Corn Ch… 1
## 10 2018-08-18 7 7150 6900 52 Grain Waves Sou… 2
## # ℹ 264,826 more rows
## # ℹ 1 more variable: TOT_SALES <dbl>
```

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

```
#### Examine PROD_NAME
summary(transaction$PROD_NAME)
```

```
##    Length     Class      Mode
##    264836 character character
```

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 product
s that are not chips
product_words <- data.table(unlist(strsplit(as.character(unique(transaction$PROD_NAME)), "
")))
# Set the column name to 'words'
setnames(product_words, '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()`.

```
# Remove digits, and special characters, and then sort the distinct words by frequency of occ
urrence.
clean_product_words <- product_words[!grepl("[0-9&@%$#]", words)]

wordFrequency <- clean_product_words[, .N, by = words][order(-N)]
wordFrequency
```

```
##          words     N
##         <char> <int>
##   1:             234
##   2:     Chips    21
##   3:    Smiths    16
##   4:   Crinkle    14
##   5:       Cut    14
##  ---
## 168:       Rst     1
## 169:      Pork     1
## 170:     Belly     1
## 171:        Pc     1
## 172: Bolognese     1
```

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

```
# # Convert the transaction data to data.table
setDT(transaction)

#### Remove salsa products
transaction[, SALSA := grepl("salsa", tolower(PROD_NAME))]
transaction <- transaction[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
summary(transaction)
```

```
##       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
```

```
dim(transaction)
```

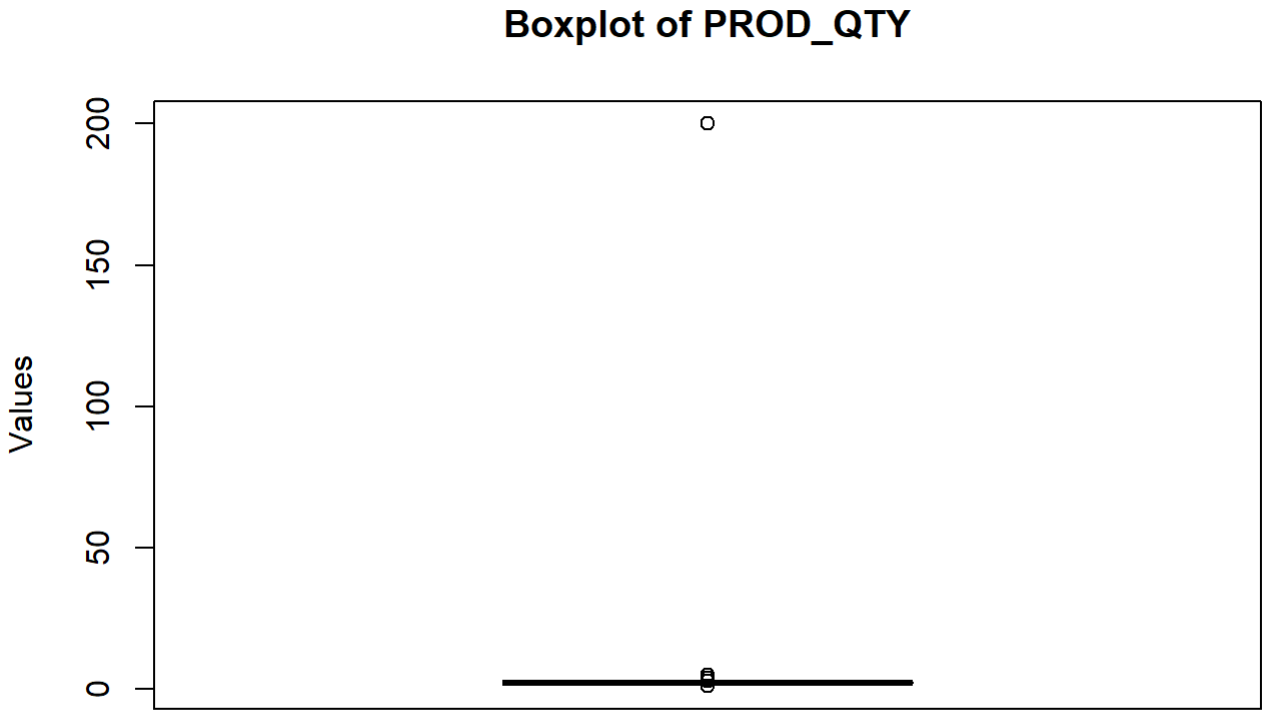
```
## [1] 246742      8
```

```
# check missing values
sum(is.na(transaction))
```

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

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
boxplot(transaction$PROD_QTY,
        names = 'PROD_QTY',
        main = "Boxplot of PROD_QTY",
        ylab = "Values",
        col = "lightblue")
```



Based on the boxplot, it shows that the outliers near 200 and above. There are two transactions purchase with the same product name which is “Dorito Corn Chp Supreme 380g” with the same product quantity with 200 packs. Both of these transactions were the same customer.

```
#### Let's see if the customer has had other transactions
# shows outliers
outliers <- transaction[PROD_QTY >= 200, ]
outliers
```

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

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

```
#### Filter out the customer based on the loyalty card number
transaction <- transaction[LYLTY_CARD_NBR != 226000, ]


#### Re-examine transaction data
numberOfTransactionsByDate <- data.frame(sort(table(transaction$DATE), decreasing = TRUE))
setnames(numberOfTransactionsByDate, c('date', 'freq'))
numberOfTransactionsByDate <- numberOfTransactionsByDate[order(as.Date(numberOfTransactionsBy
Date$date)),]
numberOfTransactionsByDate
```

```
##           date freq
## 247 2018-07-01  663
## 301 2018-07-02  650
## 177 2018-07-03  674
## 212 2018-07-04  669
## 265 2018-07-05  660
## 37  2018-07-06  711
## 88  2018-07-07  695
## 288 2018-07-08  653
## 96  2018-07-09  692
## 302 2018-07-10  650
## 65  2018-07-11  701
## 28  2018-07-12  717
## 18  2018-07-13  727
## 260 2018-07-14  661
## 36  2018-07-15  712
## 163 2018-07-16  678
## 92  2018-07-17  694
## 110 2018-07-18  689
## 335 2018-07-19  637
## 134 2018-07-20  684
## 140 2018-07-21  683
## 184 2018-07-22  673
## 185 2018-07-23  673
## 308 2018-07-24  648
## 178 2018-07-25  674
## 189 2018-07-26  672
## 80  2018-07-27  697
## 329 2018-07-28  640
## 269 2018-07-29  659
## 97  2018-07-30  692
## 117 2018-07-31  688
## 152 2018-08-01  680
## 213 2018-08-02  669
## 254 2018-08-03  662
## 235 2018-08-04  665
## 50  2018-08-05  705
## 48  2018-08-06  706
## 219 2018-08-07  668
## 89  2018-08-08  695
## 293 2018-08-09  652
## 174 2018-08-10  675
## 164 2018-08-11  678
## 326 2018-08-12  642
## 57  2018-08-13  703
## 59  2018-08-14  702
## 60  2018-08-15  702
## 103 2018-08-16  690
## 248 2018-08-17  663
## 141 2018-08-18  683
## 203 2018-08-19  670
## 320 2018-08-20  644
## 289 2018-08-21  653
## 111 2018-08-22  689
## 85  2018-08-23  696
```

```
## 311 2018-08-24  647
## 279 2018-08-25  657
## 129 2018-08-26  685
## 204 2018-08-27  670
## 340 2018-08-28  636
## 229 2018-08-29  666
## 290 2018-08-30  653
## 273 2018-08-31  658
## 121 2018-09-01  687
## 196 2018-09-02  671
## 261 2018-09-03  661
## 25  2018-09-04  718
## 130 2018-09-05  685
## 9   2018-09-06  745
## 249 2018-09-07  663
## 230 2018-09-08  666
## 51  2018-09-09  705
## 318 2018-09-10  645
## 312 2018-09-11  647
## 262 2018-09-12  661
## 315 2018-09-13  646
## 118 2018-09-14  688
## 341 2018-09-15  636
## 214 2018-09-16  669
## 266 2018-09-17  660
## 29  2018-09-18  717
## 205 2018-09-19  670
## 283 2018-09-20  656
## 73  2018-09-21  699
## 363 2018-09-22  609
## 11  2018-09-23  738
## 190 2018-09-24  672
## 16  2018-09-25  729
## 294 2018-09-26  652
## 348 2018-09-27  632
## 93  2018-09-28  694
## 197 2018-09-29  671
## 54  2018-09-30  704
## 255 2018-10-01  662
## 303 2018-10-02  650
## 274 2018-10-03  658
## 135 2018-10-04  684
## 297 2018-10-05  651
## 61  2018-10-06  702
## 321 2018-10-07  644
## 172 2018-10-08  676
## 20  2018-10-09  724
## 69  2018-10-10  700
## 49  2018-10-11  706
## 275 2018-10-12  658
## 250 2018-10-13  663
## 342 2018-10-14  636
## 179 2018-10-15  674
## 175 2018-10-16  675
## 145 2018-10-17  682
## 361 2018-10-18  611
```

```
## 74  2018-10-19  699
## 159 2018-10-20  679
## 166 2018-10-21  677
## 136 2018-10-22  684
## 270 2018-10-23  659
## 191 2018-10-24  672
## 285 2018-10-25  655
## 30  2018-10-26  716
## 323 2018-10-27  643
## 306 2018-10-28  649
## 231 2018-10-29  666
## 236 2018-10-30  665
## 295 2018-10-31  652
## 90  2018-11-01  695
## 206 2018-11-02  670
## 153 2018-11-03  680
## 81  2018-11-04  697
## 327 2018-11-05  642
## 186 2018-11-06  673
## 160 2018-11-07  679
## 256 2018-11-08  662
## 39  2018-11-09  710
## 35  2018-11-10  713
## 14  2018-11-11  731
## 165 2018-11-12  678
## 291 2018-11-13  653
## 150 2018-11-14  681
## 112 2018-11-15  689
## 161 2018-11-16  679
## 66  2018-11-17  701
## 104 2018-11-18  690
## 24  2018-11-19  722
## 13  2018-11-20  732
## 298 2018-11-21  651
## 354 2018-11-22  626
## 62  2018-11-23  702
## 207 2018-11-24  670
## 362 2018-11-25  610
## 328 2018-11-26  642
## 154 2018-11-27  680
## 330 2018-11-28  640
## 131 2018-11-29  685
## 208 2018-11-30  670
## 176 2018-12-01  675
## 286 2018-12-02  655
## 167 2018-12-03  677
## 232 2018-12-04  666
## 267 2018-12-05  660
## 319 2018-12-06  645
## 192 2018-12-07  672
## 357 2018-12-08  622
## 271 2018-12-09  659
## 239 2018-12-10  664
## 126 2018-12-11  686
## 356 2018-12-12  624
## 220 2018-12-13  668
```

```
## 82  2018-12-14  697
## 198 2018-12-15  671
## 42  2018-12-16  709
## 17  2018-12-17  729
## 6   2018-12-18  799
## 4   2018-12-19  839
## 5   2018-12-20  808
## 7   2018-12-21  781
## 3   2018-12-22  840
## 2   2018-12-23  853
## 1   2018-12-24  865
## 70  2018-12-26  700
## 105 2018-12-27  690
## 215 2018-12-28  669
## 233 2018-12-29  666
## 127 2018-12-30  686
## 304 2018-12-31  650
## 345 2019-01-01  634
## 180 2019-01-02  674
## 336 2019-01-03  637
## 55  2019-01-04  704
## 343 2019-01-05  636
## 187 2019-01-06  673
## 221 2019-01-07  668
## 216 2019-01-08  669
## 128 2019-01-09  686
## 132 2019-01-10  685
## 351 2019-01-11  631
## 122 2019-01-12  687
## 353 2019-01-13  628
## 251 2019-01-14  663
## 280 2019-01-15  657
## 181 2019-01-16  674
## 168 2019-01-17  677
## 276 2019-01-18  658
## 86  2019-01-19  696
## 142 2019-01-20  683
## 337 2019-01-21  637
## 113 2019-01-22  689
## 313 2019-01-23  647
## 358 2019-01-24  619
## 199 2019-01-25  671
## 193 2019-01-26  672
## 309 2019-01-27  648
## 263 2019-01-28  661
## 223 2019-01-29  667
## 114 2019-01-30  689
## 106 2019-01-31  690
## 45  2019-02-01  708
## 98  2019-02-02  692
## 107 2019-02-03  690
## 272 2019-02-04  659
## 101 2019-02-05  691
## 234 2019-02-06  666
## 252 2019-02-07  663
## 33  2019-02-08  714
```

```
## 200 2019-02-09  671
## 83  2019-02-10  697
## 143 2019-02-11  683
## 137 2019-02-12  684
## 94  2019-02-13  693
## 240 2019-02-14  664
## 224 2019-02-15  667
## 209 2019-02-16  670
## 146 2019-02-17  682
## 359 2019-02-18  619
## 241 2019-02-19  664
## 91  2019-02-20  695
## 316 2019-02-21  646
## 99  2019-02-22  692
## 115 2019-02-23  689
## 147 2019-02-24  682
## 95  2019-02-25  693
## 257 2019-02-26  662
## 123 2019-02-27  687
## 148 2019-02-28  682
## 210 2019-03-01  670
## 169 2019-03-02  677
## 182 2019-03-03  674
## 211 2019-03-04  670
## 21  2019-03-05  724
## 264 2019-03-06  661
## 277 2019-03-07  658
## 67  2019-03-08  701
## 338 2019-03-09  637
## 299 2019-03-10  651
## 108 2019-03-11  690
## 38  2019-03-12  711
## 63  2019-03-13  702
## 331 2019-03-14  640
## 22  2019-03-15  724
## 242 2019-03-16  664
## 32  2019-03-17  715
## 322 2019-03-18  644
## 119 2019-03-19  688
## 100 2019-03-20  692
## 194 2019-03-21  672
## 19  2019-03-22  725
## 155 2019-03-23  680
## 40  2019-03-24  710
## 120 2019-03-25  688
## 116 2019-03-26  689
## 268 2019-03-27  660
## 170 2019-03-28  677
## 75  2019-03-29  699
## 138 2019-03-30  684
## 314 2019-03-31  647
## 344 2019-04-01  635
## 71  2019-04-02  700
## 26  2019-04-03  718
## 43  2019-04-04  709
## 243 2019-04-05  664
```

```
## 151 2019-04-06  681
## 332 2019-04-07  640
## 305 2019-04-08  650
## 317 2019-04-09  646
## 76  2019-04-10  699
## 349 2019-04-11  632
## 195 2019-04-12  672
## 244 2019-04-13  664
## 133 2019-04-14  685
## 300 2019-04-15  651
## 183 2019-04-16  674
## 307 2019-04-17  649
## 225 2019-04-18  667
## 287 2019-04-19  655
## 12  2019-04-20  738
## 41  2019-04-21  710
## 201 2019-04-22  671
## 253 2019-04-23  663
## 64  2019-04-24  702
## 68  2019-04-25  701
## 139 2019-04-26  684
## 226 2019-04-27  667
## 171 2019-04-28  677
## 84  2019-04-29  697
## 156 2019-04-30  680
## 324 2019-05-01  643
## 227 2019-05-02  667
## 281 2019-05-03  657
## 352 2019-05-04  630
## 157 2019-05-05  680
## 46  2019-05-06  707
## 228 2019-05-07  667
## 78  2019-05-08  698
## 222 2019-05-09  668
## 237 2019-05-10  665
## 162 2019-05-11  679
## 124 2019-05-12  687
## 334 2019-05-13  638
## 52  2019-05-14  705
## 350 2019-05-15  632
## 245 2019-05-16  664
## 296 2019-05-17  652
## 355 2019-05-18  626
## 15  2019-05-19  730
## 47  2019-05-20  707
## 202 2019-05-21  671
## 125 2019-05-22  687
## 347 2019-05-23  633
## 102 2019-05-24  691
## 53  2019-05-25  705
## 310 2019-05-26  648
## 238 2019-05-27  665
## 144 2019-05-28  683
## 34  2019-05-29  714
## 217 2019-05-30  669
## 246 2019-05-31  664
```

```
## 149 2019-06-01  682
## 258 2019-06-02  662
## 284 2019-06-03  656
## 339 2019-06-04  637
## 158 2019-06-05  680
## 72  2019-06-06  700
## 8   2019-06-07  762
## 77  2019-06-08  699
## 27  2019-06-09  718
## 173 2019-06-10  676
## 346 2019-06-11  634
## 44  2019-06-12  709
## 364 2019-06-13  607
## 10  2019-06-14  743
## 23  2019-06-15  724
## 109 2019-06-16  690
## 278 2019-06-17  658
## 333 2019-06-18  639
## 259 2019-06-19  662
## 79  2019-06-20  698
## 31  2019-06-21  716
## 325 2019-06-22  643
## 292 2019-06-23  653
## 360 2019-06-24  612
## 87  2019-06-25  696
## 282 2019-06-26  657
## 218 2019-06-27  669
## 188 2019-06-28  673
## 58  2019-06-29  703
## 56  2019-06-30  704
```

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
count_unique_dates <- sort(unique(transaction$DATE, asc=TRUE))
count_unique_dates
```

```
##   [1] "2018-07-01" "2018-07-02" "2018-07-03" "2018-07-04" "2018-07-05"
##   [6] "2018-07-06" "2018-07-07" "2018-07-08" "2018-07-09" "2018-07-10"
##  [11] "2018-07-11" "2018-07-12" "2018-07-13" "2018-07-14" "2018-07-15"
##  [16] "2018-07-16" "2018-07-17" "2018-07-18" "2018-07-19" "2018-07-20"
##  [21] "2018-07-21" "2018-07-22" "2018-07-23" "2018-07-24" "2018-07-25"
##  [26] "2018-07-26" "2018-07-27" "2018-07-28" "2018-07-29" "2018-07-30"
##  [31] "2018-07-31" "2018-08-01" "2018-08-02" "2018-08-03" "2018-08-04"
##  [36] "2018-08-05" "2018-08-06" "2018-08-07" "2018-08-08" "2018-08-09"
##  [41] "2018-08-10" "2018-08-11" "2018-08-12" "2018-08-13" "2018-08-14"
##  [46] "2018-08-15" "2018-08-16" "2018-08-17" "2018-08-18" "2018-08-19"
##  [51] "2018-08-20" "2018-08-21" "2018-08-22" "2018-08-23" "2018-08-24"
##  [56] "2018-08-25" "2018-08-26" "2018-08-27" "2018-08-28" "2018-08-29"
##  [61] "2018-08-30" "2018-08-31" "2018-09-01" "2018-09-02" "2018-09-03"
##  [66] "2018-09-04" "2018-09-05" "2018-09-06" "2018-09-07" "2018-09-08"
##  [71] "2018-09-09" "2018-09-10" "2018-09-11" "2018-09-12" "2018-09-13"
##  [76] "2018-09-14" "2018-09-15" "2018-09-16" "2018-09-17" "2018-09-18"
##  [81] "2018-09-19" "2018-09-20" "2018-09-21" "2018-09-22" "2018-09-23"
##  [86] "2018-09-24" "2018-09-25" "2018-09-26" "2018-09-27" "2018-09-28"
##  [91] "2018-09-29" "2018-09-30" "2018-10-01" "2018-10-02" "2018-10-03"
##  [96] "2018-10-04" "2018-10-05" "2018-10-06" "2018-10-07" "2018-10-08"
## [101] "2018-10-09" "2018-10-10" "2018-10-11" "2018-10-12" "2018-10-13"
## [106] "2018-10-14" "2018-10-15" "2018-10-16" "2018-10-17" "2018-10-18"
## [111] "2018-10-19" "2018-10-20" "2018-10-21" "2018-10-22" "2018-10-23"
## [116] "2018-10-24" "2018-10-25" "2018-10-26" "2018-10-27" "2018-10-28"
## [121] "2018-10-29" "2018-10-30" "2018-10-31" "2018-11-01" "2018-11-02"
## [126] "2018-11-03" "2018-11-04" "2018-11-05" "2018-11-06" "2018-11-07"
## [131] "2018-11-08" "2018-11-09" "2018-11-10" "2018-11-11" "2018-11-12"
## [136] "2018-11-13" "2018-11-14" "2018-11-15" "2018-11-16" "2018-11-17"
## [141] "2018-11-18" "2018-11-19" "2018-11-20" "2018-11-21" "2018-11-22"
## [146] "2018-11-23" "2018-11-24" "2018-11-25" "2018-11-26" "2018-11-27"
## [151] "2018-11-28" "2018-11-29" "2018-11-30" "2018-12-01" "2018-12-02"
## [156] "2018-12-03" "2018-12-04" "2018-12-05" "2018-12-06" "2018-12-07"
## [161] "2018-12-08" "2018-12-09" "2018-12-10" "2018-12-11" "2018-12-12"
## [166] "2018-12-13" "2018-12-14" "2018-12-15" "2018-12-16" "2018-12-17"
## [171] "2018-12-18" "2018-12-19" "2018-12-20" "2018-12-21" "2018-12-22"
## [176] "2018-12-23" "2018-12-24" "2018-12-26" "2018-12-27" "2018-12-28"
## [181] "2018-12-29" "2018-12-30" "2018-12-31" "2019-01-01" "2019-01-02"
## [186] "2019-01-03" "2019-01-04" "2019-01-05" "2019-01-06" "2019-01-07"
## [191] "2019-01-08" "2019-01-09" "2019-01-10" "2019-01-11" "2019-01-12"
## [196] "2019-01-13" "2019-01-14" "2019-01-15" "2019-01-16" "2019-01-17"
## [201] "2019-01-18" "2019-01-19" "2019-01-20" "2019-01-21" "2019-01-22"
## [206] "2019-01-23" "2019-01-24" "2019-01-25" "2019-01-26" "2019-01-27"
## [211] "2019-01-28" "2019-01-29" "2019-01-30" "2019-01-31" "2019-02-01"
## [216] "2019-02-02" "2019-02-03" "2019-02-04" "2019-02-05" "2019-02-06"
## [221] "2019-02-07" "2019-02-08" "2019-02-09" "2019-02-10" "2019-02-11"
## [226] "2019-02-12" "2019-02-13" "2019-02-14" "2019-02-15" "2019-02-16"
## [231] "2019-02-17" "2019-02-18" "2019-02-19" "2019-02-20" "2019-02-21"
## [236] "2019-02-22" "2019-02-23" "2019-02-24" "2019-02-25" "2019-02-26"
## [241] "2019-02-27" "2019-02-28" "2019-03-01" "2019-03-02" "2019-03-03"
## [246] "2019-03-04" "2019-03-05" "2019-03-06" "2019-03-07" "2019-03-08"
## [251] "2019-03-09" "2019-03-10" "2019-03-11" "2019-03-12" "2019-03-13"
## [256] "2019-03-14" "2019-03-15" "2019-03-16" "2019-03-17" "2019-03-18"
## [261] "2019-03-19" "2019-03-20" "2019-03-21" "2019-03-22" "2019-03-23"
## [266] "2019-03-24" "2019-03-25" "2019-03-26" "2019-03-27" "2019-03-28"
## [271] "2019-03-29" "2019-03-30" "2019-03-31" "2019-04-01" "2019-04-02"
```

```
## [276] "2019-04-03" "2019-04-04" "2019-04-05" "2019-04-06" "2019-04-07"
## [281] "2019-04-08" "2019-04-09" "2019-04-10" "2019-04-11" "2019-04-12"
## [286] "2019-04-13" "2019-04-14" "2019-04-15" "2019-04-16" "2019-04-17"
## [291] "2019-04-18" "2019-04-19" "2019-04-20" "2019-04-21" "2019-04-22"
## [296] "2019-04-23" "2019-04-24" "2019-04-25" "2019-04-26" "2019-04-27"
## [301] "2019-04-28" "2019-04-29" "2019-04-30" "2019-05-01" "2019-05-02"
## [306] "2019-05-03" "2019-05-04" "2019-05-05" "2019-05-06" "2019-05-07"
## [311] "2019-05-08" "2019-05-09" "2019-05-10" "2019-05-11" "2019-05-12"
## [316] "2019-05-13" "2019-05-14" "2019-05-15" "2019-05-16" "2019-05-17"
## [321] "2019-05-18" "2019-05-19" "2019-05-20" "2019-05-21" "2019-05-22"
## [326] "2019-05-23" "2019-05-24" "2019-05-25" "2019-05-26" "2019-05-27"
## [331] "2019-05-28" "2019-05-29" "2019-05-30" "2019-05-31" "2019-06-01"
## [336] "2019-06-02" "2019-06-03" "2019-06-04" "2019-06-05" "2019-06-06"
## [341] "2019-06-07" "2019-06-08" "2019-06-09" "2019-06-10" "2019-06-11"
## [346] "2019-06-12" "2019-06-13" "2019-06-14" "2019-06-15" "2019-06-16"
## [351] "2019-06-17" "2019-06-18" "2019-06-19" "2019-06-20" "2019-06-21"
## [356] "2019-06-22" "2019-06-23" "2019-06-24" "2019-06-25" "2019-06-26"
## [361] "2019-06-27" "2019-06-28" "2019-06-29" "2019-06-30"
```

```
summary(count_unique_dates)
```

```
## Min.  1st Qu.  Median Mean 3rd Qu.  Max.
## "2018-07-01" "2018-09-29" "2018-12-30" "2018-12-30" "2019-03-31"
"2019-06-30"
```

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

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

# Create a sequence of dates from 1 Jul 2018 to 30 Jun 2019
seqOfDates <- data.table(seq(as.Date('2018-07-01'),as.Date('2019-06-30'), by = "day"))
setnames(seqOfDates,"date")

# Ensure seqOfDates$date is of Date type
seqOfDates$date <- as.Date(seqOfDates$date)

# Ensure the date column in numberOfTransactionsByDate is of Date type
numberOfTransactionsByDate$date <- as.Date(numberOfTransactionsByDate$date)

# Merging the sequence of dates with the transaction counts
transactions_by_days <- merge(x = seqOfDates,
                              y = numberOfTransactionsByDate,
                              by = "date",
                              all.x = TRUE)
transactions_by_days
```

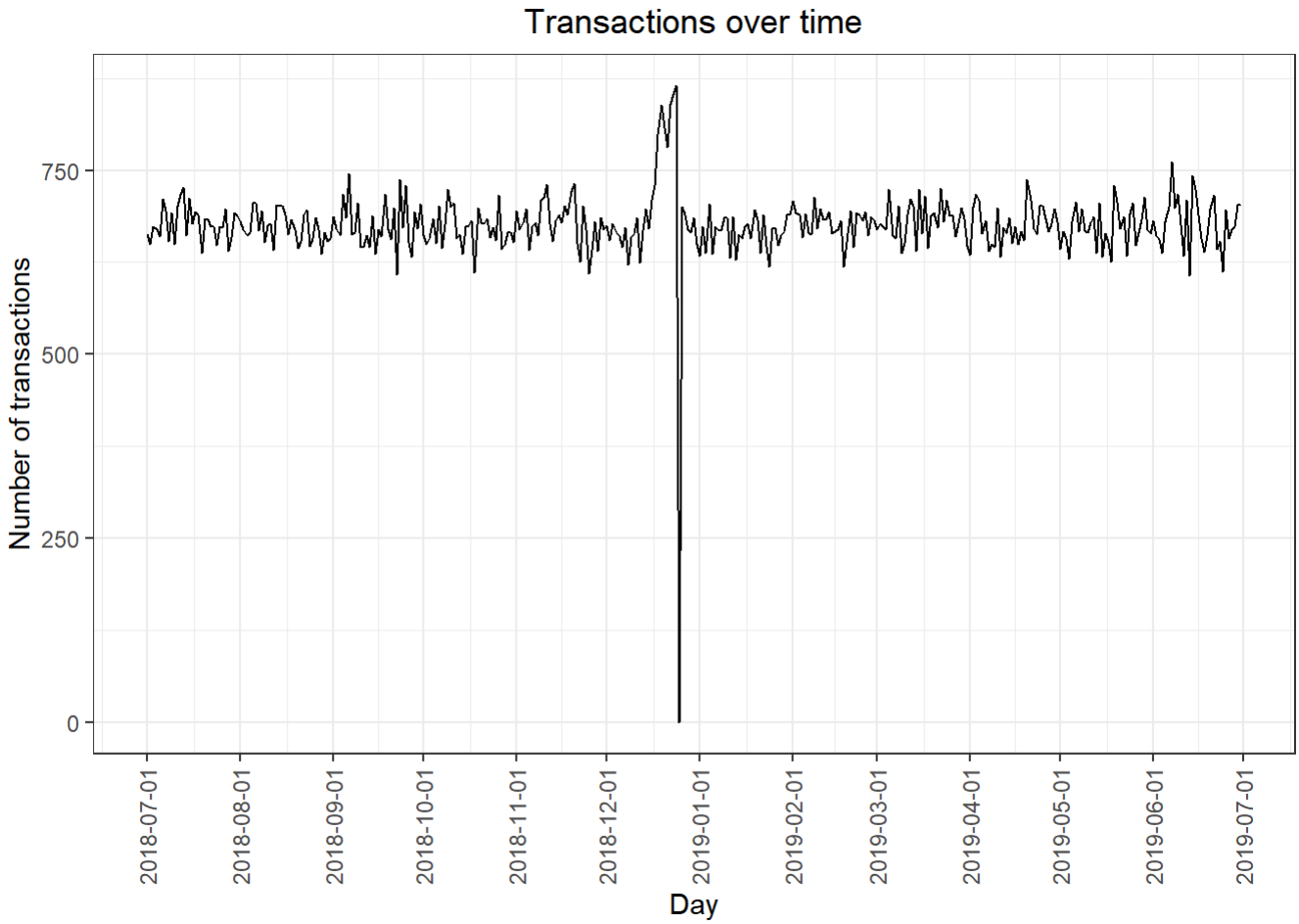
```
## Key: <date>
##            date  freq
##          <Date> <int>
##   1: 2018-07-01   663
##   2: 2018-07-02   650
##   3: 2018-07-03   674
##   4: 2018-07-04   669
##   5: 2018-07-05   660
##  ---
## 361: 2019-06-26   657
## 362: 2019-06-27   669
## 363: 2019-06-28   673
## 364: 2019-06-29   703
## 365: 2019-06-30   704
```

```
# setDT(transactions_by_days)

# Replace NA values in the freq column with 0 for days with no transactions
transactions_by_days[is.na(freq), freq := 0]

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

#### Plot transactions over time
trans_over_time <- ggplot(transactions_by_days, aes(x = date, y = freq)) +
  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))
trans_over_time
```

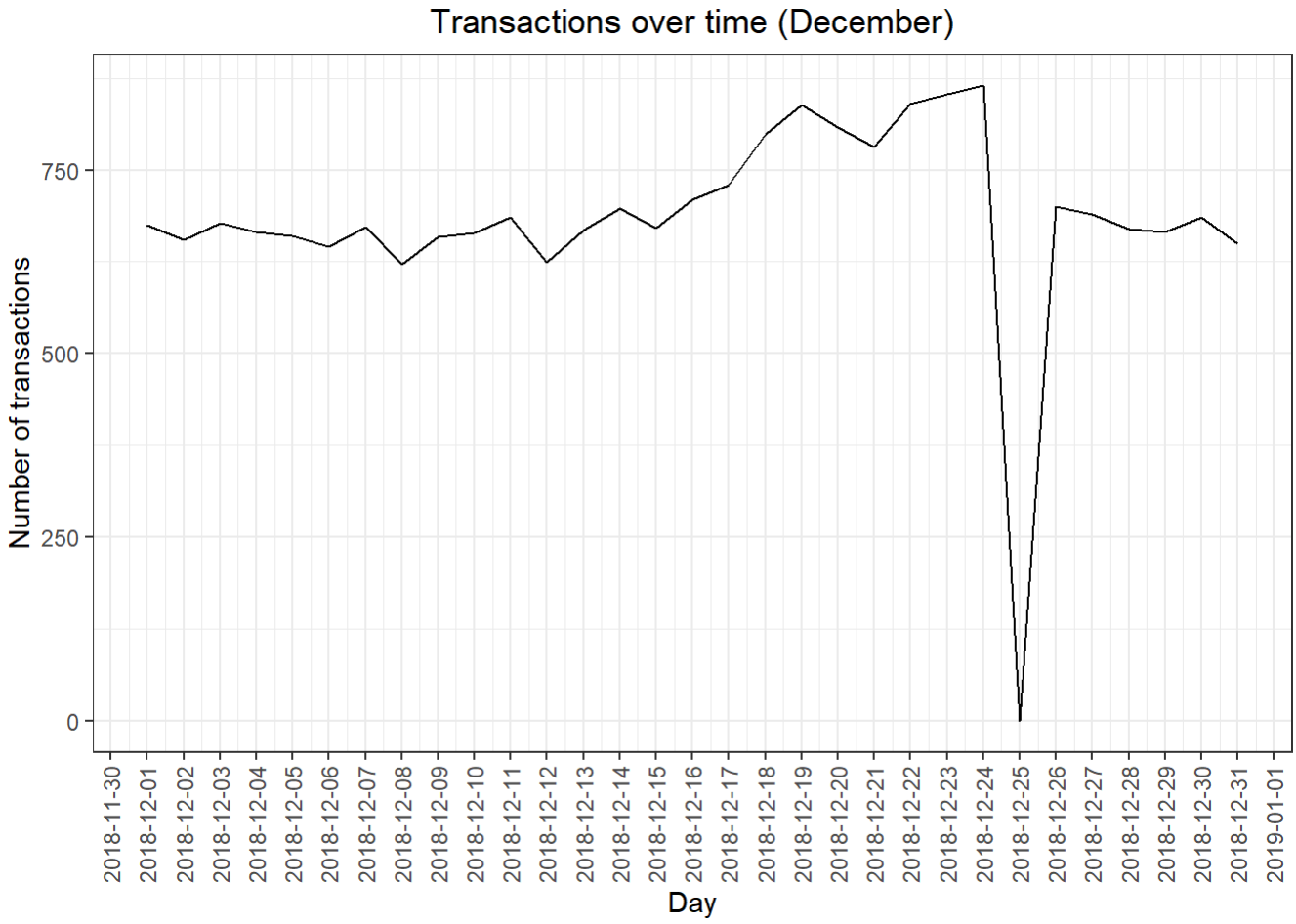



```
ggsave("transactions_over_time.png", plot = trans_over_time, width = 12, height = 6, dpi = 30
0)
```

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
december <- transactions_by_days[(date >= "2018-12-01" & date <= "2018-12-31"), ]

#### Plot transactions over time
trans_over_time_Dec2018 <- ggplot(december, aes(x = date, y = freq)) +
  geom_line() +
  labs(x = "Day",
       y = "Number of transactions",
       title = "Transactions over time (December)") +
  scale_x_date(breaks = "1 day") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
trans_over_time_Dec2018
```




```
ggsave("transactions_over_time (Dec).png", plot = trans_over_time_Dec2018, width = 12, height
= 6, dpi = 300)
```

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.

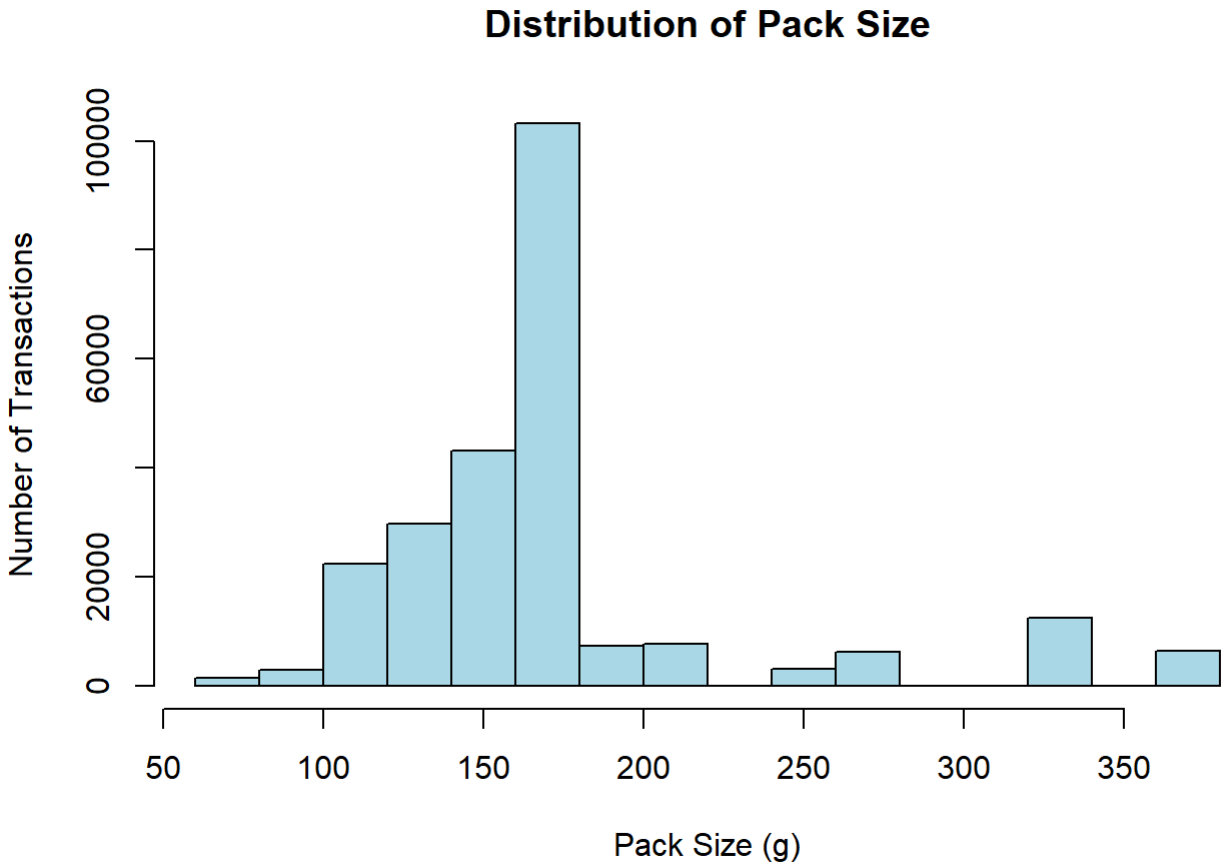
```
#### Pack size
#### We can work this out by taking the digits that are in PROD_NAME
transaction[, PACK_SIZE := parse_number(PROD_NAME)]

#### Let's check if the pack sizes look sensible
pack_size <- transaction[, .N, PACK_SIZE][order(PACK_SIZE)]
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.
# Plot a histogram showing the number of transactions by pack size.
options(scipen=999)
hist(transaction[, PACK_SIZE],
             col = 'lightblue',
             border='black',
             xlab = 'Pack Size (g)',
             ylab = 'Number of Transactions',
             main = 'Distribution of 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.

```
#### create Brands column & extract from the product name
transaction[, BRAND := substr(PROD_NAME, 1, regexpr(' ', PROD_NAME) - 1)]
summary(transaction)
```

```
##       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
##    PACK_SIZE        BRAND
##  Min.   : 70.0   Length:246740
##  1st Qu.:150.0   Class :character
##  Median :170.0   Mode  :character
##  Mean   :175.6
##  3rd Qu.:175.0
##  Max.   :380.0
```

```
#### Checking brands
unique(transaction$BRAND)
```

```
##  [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.

```
#### Clean brand names
transaction[BRAND == "Red", BRAND := "RRD"]
transaction[BRAND == "Smiths", BRAND := "Smith"]
transaction[BRAND == "GrnWves", BRAND := "Sunbites"]
transaction[BRAND == "Grain", BRAND := "Sunbites"]
transaction[BRAND == "Doritos", BRAND := "Dorito"]
transaction[BRAND == "NCC", BRAND := "Natural"]
transaction[BRAND == "WW", BRAND := "Woolworths"]
transaction[BRAND == "Infzns", BRAND := "Infuzions"]
transaction[BRAND == "Snbts", BRAND := "Sunbites"]

#### Check again
unique(transaction$BRAND)
```

```
##  [1] "Natural"    "CCs"        "Smith"      "Kettle"     "Sunbites"
##  [6] "Dorito"     "Twisties"   "Woolworths" "Thins"      "Burger"
## [11] "Cheezels"   "Infuzions"  "RRD"        "Pringles"   "Tyrrells"
## [16] "Cobs"       "French"     "Tostitos"   "Cheetos"
```

Examining customer data

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

```
#### Examining customer data

# summary of customer dataset
summary(customer)
```

```
##  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
```

```
# check the structure dataset
str(customer)
```

```
## 'data.frame': 72637 obs. of 3 variables:
## $ LYLTY_CARD_NBR : int 1000 1002 1003 1004 1005 1007 1009 1010 1011 1012 ...
## $ LIFESTAGE : chr "YOUNG SINGLES/COUPLES" "YOUNG SINGLES/COUPLES" "YOUNG
FAMILIES" "OLDER SINGLES/COUPLES" ...
## $ PREMIUM_CUSTOMER: chr "Premium" "Mainstream" "Budget" "Mainstream" ...
```

```
# capital first letter for LIFESTAGE column
customer$LIFESTAGE <- str_to_title(customer$LIFESTAGE)
head(customer)
```

```
##   LYLTY_CARD_NBR              LIFESTAGE PREMIUM_CUSTOMER
## 1           1000  Young Singles/Couples          Premium
## 2           1002  Young Singles/Couples       Mainstream
## 3           1003         Young Families           Budget
## 4           1004  Older Singles/Couples       Mainstream
## 5           1005 Midage Singles/Couples       Mainstream
## 6           1007  Young Singles/Couples           Budget
```

```
# check missing value
sum(is.na(customer))
```

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

```
#### Merge transaction data to customer data
data <- merge(transaction, customer, all.x = TRUE)
data
```

```
## Key: <LYLTY_CARD_NBR>
##         LYLTY_CARD_NBR       DATE STORE_NBR TXN_ID PROD_NBR
##                  <int>     <Date>     <num>  <num>    <num>
##      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
##     ---
## 246736:        2370651 2018-08-03        88 240350        4
## 246737:        2370701 2018-12-08        88 240378       24
## 246738:        2370751 2018-10-01        88 240394       60
## 246739:        2370961 2018-10-24        88 240480       70
## 246740:        2373711 2018-12-14        88 241815       16
##                                     PROD_NAME PROD_QTY TOT_SALES PACK_SIZE
##                                         <char>    <num>     <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
##     ---
## 246736:         Dorito Corn Chp     Supreme 380g        2      13.0       380
## 246737:    Grain Waves         Sweet Chilli 210g        2       7.2       210
## 246738:     Kettle Tortilla ChpsFeta&Garlic 150g        2       9.2       150
## 246739:  Tyrrells Crisps     Lightly Salted 165g        2       8.4       165
## 246740: Smiths Crinkle Chips Salt & Vinegar 330g        2      11.4       330
##              BRAND              LIFESTAGE PREMIUM_CUSTOMER
##             <char>                 <char>           <char>
##      1:    Natural  Young Singles/Couples          Premium
##      2:        RRD  Young Singles/Couples       Mainstream
##      3:   Sunbites         Young Families           Budget
##      4:    Natural         Young Families           Budget
##      5: Woolworths  Older Singles/Couples       Mainstream
##     ---
## 246736:     Dorito Midage Singles/Couples       Mainstream
## 246737:   Sunbites         Young Families       Mainstream
## 246738:     Kettle         Young Families          Premium
## 246739:   Tyrrells         Older Families           Budget
## 246740:      Smith  Young Singles/Couples       Mainstream
```

As the number of rows in `data` is the same as that of `transaction`, 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 `transaction` 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.

```
# check missing values
sum(is.na(data))
```

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

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

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

```
write.csv(data, "D:\\UMP\\Extra Program\\Virtual Internship (Forage)\\Quantium\\Task 1\\QVI_data.csv", row.names=FALSE)
```

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:

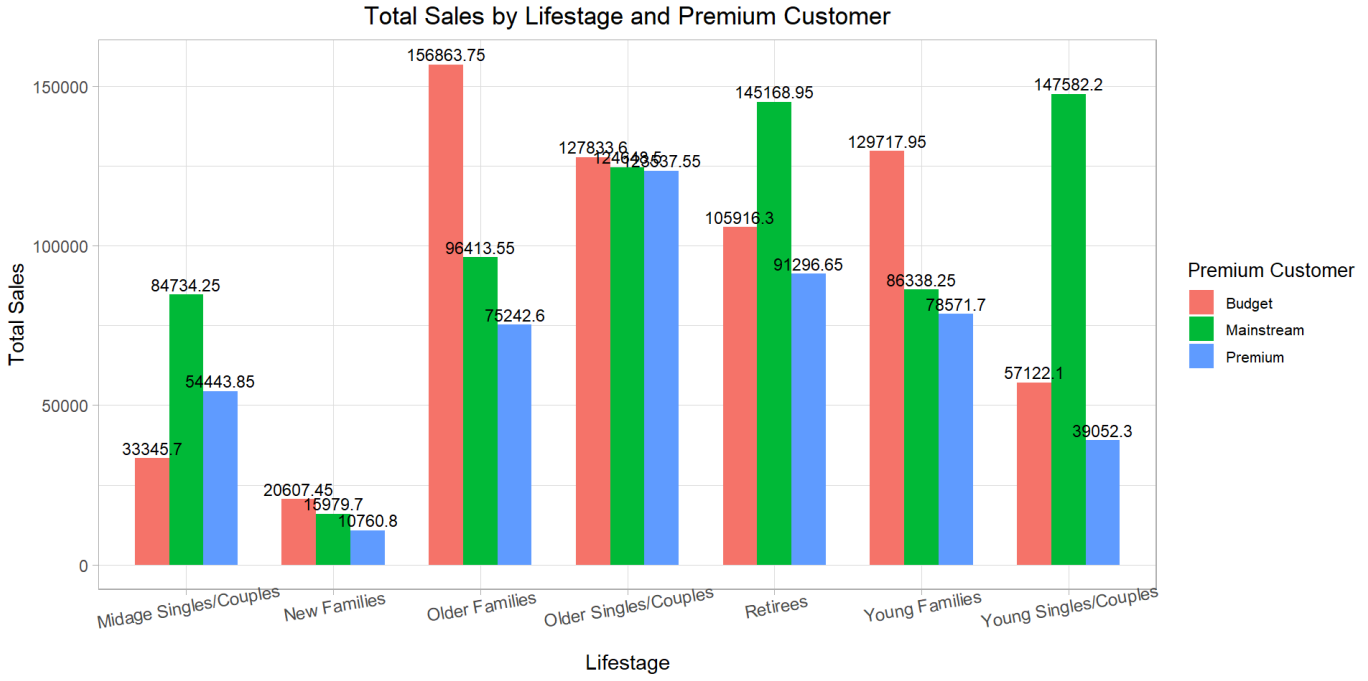
- The customer's total spend over the period and total spend for each transaction to understand what proportion of their grocery spend is on chips
- Proportion of customers in each customer segment overall to compare against the mix of customers who purchase chips

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_data <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(Total_Sales = sum(TOT_SALES, na.rm = TRUE), .groups = "drop")
total_sales_data
```

```
## # A tibble: 21 × 3
##    LIFESTAGE              PREMIUM_CUSTOMER Total_Sales
##    <chr>                  <chr>                 <dbl>
##  1 Midage Singles/Couples Budget               33346.
##  2 Midage Singles/Couples Mainstream           84734.
##  3 Midage Singles/Couples Premium              54444.
##  4 New Families           Budget               20607.
##  5 New Families           Mainstream           15980.
##  6 New Families           Premium              10761.
##  7 Older Families         Budget              156864.
##  8 Older Families         Mainstream           96414.
##  9 Older Families         Premium              75243.
## 10 Older Singles/Couples  Budget              127834.
## # ℹ 11 more rows
```

```
#### Total sales by LIFESTAGE and PREMIUM_CUSTOMER
total_sales_lifestage_premium_customer <- ggplot(
  total_sales_data, aes(x = LIFESTAGE,
                        y = Total_Sales,
                        fill = PREMIUM_CUSTOMER)) +
  # bar width
  geom_bar(stat = "identity",
           position = "dodge",
           width = 0.7) +
  # add labels
  geom_text(aes(label = Total_Sales),
            position = position_dodge(width = 0.7),
            size = 3,
            vjust = -0.3) +
  # add titles & minimal theme
  labs(title = "Total Sales by Lifestage and Premium Customer",
       x = "Lifestage",
       y = "Total Sales",
       fill = "Premium Customer") +
  theme_light() +
  theme(plot.title = element_text(hjust = 0.5), # center the title
        legend.position = 'right',
        legend.title = element_text(size = 10),
        legend.text = element_text(size = 8),
        legend.key.size = unit(0.5, "cm"),
        # font size of lables (lifestage)
        axis.text.x = element_text(size = 9,
                                   angle = 10,
                                   hjust = 0.5))
total_sales_lifestage_premium_customer
```




```
ggsave("total_sales_lifestage_premium_customer.png", plot = total_sales_lifestage_premium_cus
tomer)
```

```
## Saving 10 x 5 in image
```

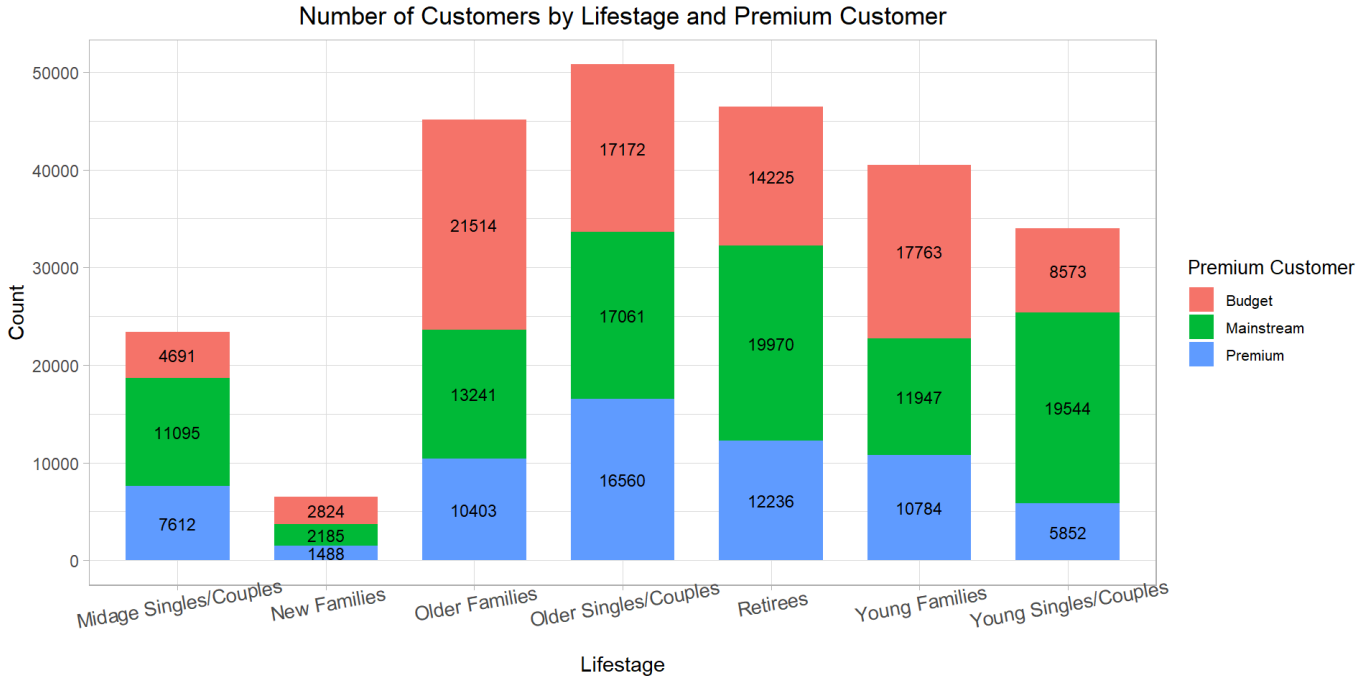
The sales mainly coming from the top 3 which are Budget with older families (56863.75), Mainstream with young singles/couples (147582.2) and Mainstream with retirees (145168.95).

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
num_customers <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(Num_Customers = n(), .groups = 'drop')
num_customers
```

```
## # A tibble: 21 × 3
##    LIFESTAGE              PREMIUM_CUSTOMER Num_Customers
##    <chr>                  <chr>                    <int>
##  1 Midage Singles/Couples Budget                    4691
##  2 Midage Singles/Couples Mainstream               11095
##  3 Midage Singles/Couples Premium                   7612
##  4 New Families           Budget                    2824
##  5 New Families           Mainstream                2185
##  6 New Families           Premium                   1488
##  7 Older Families         Budget                   21514
##  8 Older Families         Mainstream               13241
##  9 Older Families         Premium                  10403
## 10 Older Singles/Couples  Budget                   17172
## # ℹ 11 more rows
```

```
num_cust <- ggplot(
  data, aes(x = LIFESTAGE,
            fill = PREMIUM_CUSTOMER)) +
  # bar width
  geom_bar(position = "stack",
           width = 0.7) +
  # add labels with count values using "after_stat(count)"
  geom_text(stat = "count",
            aes(label = after_stat(count)),
            position = position_stack(vjust = 0.5),
            size = 3) +
  # add titles & minimal theme
  labs(title = "Number of Customers by Lifestage and Premium Customer",
       x = "Lifestage",
       y = "Count",
       fill = "Premium Customer") +
  theme_light() +
  theme(plot.title = element_text(hjust = 0.5), # center the title
        legend.position = 'right',
        legend.title = element_text(size = 10),
        legend.text = element_text(size = 8),
        legend.key.size = unit(0.5, "cm"),
        # font size of lables (lifestage)
        axis.text.x = element_text(size = 10,
                                   angle = 10,
                                   hjust = 0.5))
num_cust
```




```
ggsave("num_customers_lifestage_premium_customer.png", plot = num_cust)
```

```
## Saving 10 x 5 in image
```

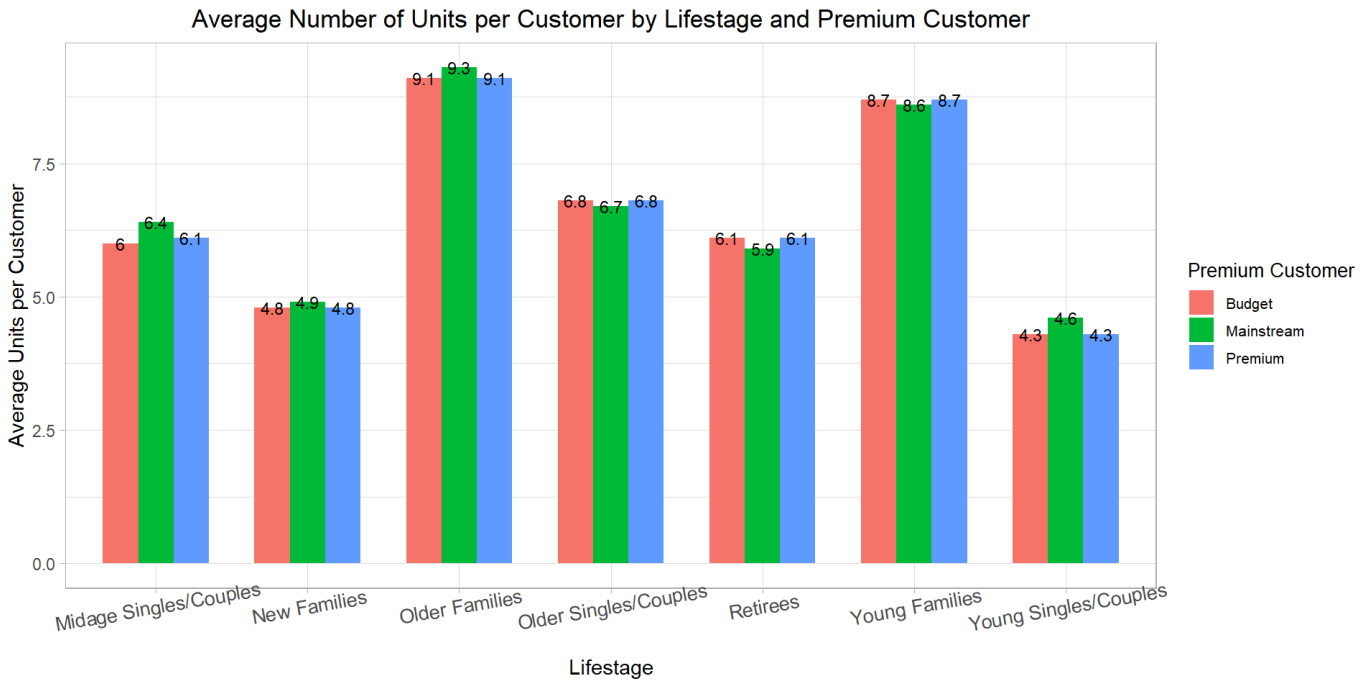
There are more Budget with older families (21514) and Mainstream with retirees (19970) who buy chips. This contributes to there being more sales to these customer segments but this is not a major driver for the Mainstream with young singles/couples segment.

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
avg_units_per_cust <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(Average_Units = round(sum(PROD_QTY, na.rm=TRUE) / n_distinct(LYLTY_CARD_NBR), 1),
            .groups = 'drop')
avg_units_per_cust
```

```
## # A tibble: 21 × 3
##    LIFESTAGE              PREMIUM_CUSTOMER Average_Units
##    <chr>                  <chr>                    <dbl>
##  1 Midage Singles/Couples Budget                     6
##  2 Midage Singles/Couples Mainstream                 6.4
##  3 Midage Singles/Couples Premium                    6.1
##  4 New Families           Budget                     4.8
##  5 New Families           Mainstream                 4.9
##  6 New Families           Premium                    4.8
##  7 Older Families         Budget                     9.1
##  8 Older Families         Mainstream                 9.3
##  9 Older Families         Premium                    9.1
## 10 Older Singles/Couples  Budget                     6.8
## # ℹ 11 more rows
```

```
avg_units <- ggplot(
  avg_units_per_cust,
  aes(x = LIFESTAGE,
      y = Average_Units,
      fill = PREMIUM_CUSTOMER)) +
  # bar width
  geom_bar(stat = "identity",
           position = "dodge",
           width = 0.7) +
  # add labels with count values using "after_stat(count)"
  geom_text(aes(label = round(Average_Units, 1)),
            position = position_dodge(width = 0.7),
            size = 3) +
  # add titles & minimal theme
  labs(title = "Average Number of Units per Customer by Lifestage and Premium Customer",
       x = "Lifestage",
       y = "Average Units per Customer",
       fill = "Premium Customer") +
  theme_light() +
  theme(plot.title = element_text(hjust = 0.5), # center the title
        legend.position = 'right',
        legend.title = element_text(size = 10),
        legend.text = element_text(size = 8),
        legend.key.size = unit(0.5, "cm"),
        # font size of lables (lifestage)
        axis.text.x = element_text(size = 10,
                                   angle = 10,
                                   hjust = 0.5))
avg_units
```




```
ggsave("avg_num_per_customers_lifestage_premium_customer.png", plot = avg_units)
```

```
## Saving 10 x 5 in image
```

Older families (between 9.1 and 9.3) and young families (between 8.6 and 8.7) 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
avg_price_per_unit <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(Average_Price = round(sum(TOT_SALES, na.rm=TRUE)/sum(PROD_QTY), 1),
            .groups = 'drop')
avg_price_per_unit
```

```
## # A tibble: 21 × 3
##    LIFESTAGE              PREMIUM_CUSTOMER Average_Price
##    <chr>                  <chr>                    <dbl>
##  1 Midage Singles/Couples Budget                     3.8
##  2 Midage Singles/Couples Mainstream                 4
##  3 Midage Singles/Couples Premium                    3.8
##  4 New Families           Budget                     3.9
##  5 New Families           Mainstream                 3.9
##  6 New Families           Premium                    3.9
##  7 Older Families         Budget                     3.7
##  8 Older Families         Mainstream                 3.7
##  9 Older Families         Premium                    3.7
## 10 Older Singles/Couples  Budget                     3.9
## # ℹ 11 more rows
```

```
avg_price <- ggplot(
  avg_price_per_unit,
  aes(x = LIFESTAGE,
      y = Average_Price,
      fill = PREMIUM_CUSTOMER)) +
  # bar width
  geom_bar(stat = "identity",
           position = "dodge",
           width = 0.7) +
  # add labels with count values using "after_stat(count)"
  geom_text(aes(label = round(Average_Price, 1)),
            position = position_dodge(width = 0.7),
            size = 3) +
  # add titles & minimal theme
  labs(title = "Average Price of per Unit by Lifestage and Premium Customer",
       x = "Lifestage",
       y = "Average Price per Customer",
       fill = "Premium Customer") +
  theme_light() +
  theme(plot.title = element_text(hjust = 0.5), # center the title
        legend.position = 'right',
        legend.title = element_text(size = 10),
        legend.text = element_text(size = 8),
        legend.key.size = unit(0.5, "cm"),
        # font size of lables (lifestage)
        axis.text.x = element_text(size = 10,
                                   angle = 10,
                                   hjust = 0.5))
avg_price
```


```
ggsave("avg_price_per_units_lifestage_premium_customer.png", plot = avg_units)
```

```
## Saving 10 x 5 in image
```

Both Mainstream midage (4) and young (4.1) singles/couples more willing to pay more per packet of chips compared to their budget (3.7, 3.8) and premium (3.7, 3.8) 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/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

## step 1: filter the data for two groups
group1 <- data %>%
  filter(PREMIUM_CUSTOMER == "Mainstream" &
           (LIFESTAGE == "Young Singles/Couples " | LIFESTAGE == "Midage Singles/Couples"))

group2 <- data %>%
  filter((PREMIUM_CUSTOMER == "Budget" | PREMIUM_CUSTOMER == "Premium") &
           (LIFESTAGE == "Young Singles/Couples" | LIFESTAGE == "Midage Singles/Couples"))


## step 2: perform an independent t-test
t_test <- t.test(group1$TOT_SALES / group1$PROD_QTY,
                 group2$TOT_SALES / group2$PROD_QTY,
                 alternative =  "two.sided")
t_test
```

```
##
##  Welch Two Sample t-test
##
## data:  group1$TOT_SALES/group1$PROD_QTY and group2$TOT_SALES/group2$PROD_QTY
## t = 24.248, df = 21967, p-value < 0.00000000000000022
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.2644896 0.3110091
## sample estimates:
## mean of x mean of y
##  3.994241  3.706491
```

\(H_{0}\): The mean unit price of the Mainstream group (young and mid-age singles/couples) is equal to the mean unit price of the Budget or Premium group (young and mid-age singles/couples).

\(H_{1}\): The mean unit price of the Mainstream group is not equal to the mean unit price of the Budget or Premium group.

p-value = 0

Since (p-value = 0) < (α = 0.05), reject \(H_{0}\).

At α=0.05, the mean unit price of the Mainstream group is not equal to the mean unit price of the Budget or Premium group.

The t-test results in a p-value of 0, 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 with 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
library(data.table)

# Filter for the target & other segment
target_seg <- data %>%
  filter(PREMIUM_CUSTOMER == "Mainstream" & LIFESTAGE == "Young Singles/Couples")
other_seg <- data %>%
  filter(!(PREMIUM_CUSTOMER == "Mainstream" & LIFESTAGE == "Young Singles/Couples"))

# calc total quantity
total_quantity_target <- sum(target_seg$PROD_QTY, na.rm = TRUE)
total_quantity_other <- sum(other_seg$PROD_QTY, na.rm = TRUE)

# calc proportion of product quantity
quantity_target_brand <- target_seg[, .(Proportion_Target = sum(PROD_QTY, na.rm = TRUE) / tot
al_quantity_target), by = BRAND]
quantity_other_brand <- other_seg[, .(Proportion_Other = sum(PROD_QTY, na.rm = TRUE) / total_
quantity_other), by = BRAND]

# Merge
brand_affinity_comparison <- merge(quantity_target_brand, quantity_other_brand,
                                   by = "BRAND",
                                   all = TRUE)

# calc affinity index (how much more the target segment prefers each brand)
brand_affinity_comparison[, Affinity_Index := Proportion_Target / Proportion_Other]

# Round values to 2 decimal places
brand_affinity_comparison[, `:=`(
  Proportion_Target = round(Proportion_Target, 4),
  Proportion_Other = round(Proportion_Other, 4),
  Affinity_Index = round(Affinity_Index, 4)
)]

# Order the results by Affinity Index in desc order
brand_affinity_sorted <- brand_affinity_comparison[order(-Affinity_Index)]
brand_affinity_sorted
```

```
##          BRAND Proportion_Target Proportion_Other Affinity_Index
##         <char>             <num>            <num>          <num>
##  1:   Tyrrells            0.0316           0.0257         1.2281
##  2:   Twisties            0.0462           0.0379         1.2193
##  3:     Dorito            0.1228           0.1011         1.2146
##  4:     Kettle            0.1980           0.1656         1.1959
##  5:   Tostitos            0.0454           0.0380         1.1957
##  6:   Pringles            0.1194           0.1006         1.1867
##  7:       Cobs            0.0446           0.0390         1.1431
##  8:  Infuzions            0.0647           0.0571         1.1334
##  9:      Thins            0.0604           0.0570         1.0594
## 10:   Cheezels            0.0180           0.0186         0.9638
## 11:   Sunbites            0.0391           0.0438         0.8925
## 12:      Smith            0.0964           0.1246         0.7735
## 13:     French            0.0039           0.0058         0.6856
## 14:    Cheetos            0.0080           0.0121         0.6657
## 15:        RRD            0.0438           0.0675         0.6491
## 16:    Natural            0.0196           0.0309         0.6352
## 17:        CCs            0.0112           0.0189         0.5917
## 18: Woolworths            0.0241           0.0494         0.4876
## 19:     Burger            0.0029           0.0066         0.4436
```

We can see that :

- Brands like Tyrrells, Twisties, Dorito, Kettle, and Tostitos have an affinity index greater than 1, indicating that young singles/couples in the Mainstream segment tend to prefer these brands significantly more than other customer segments. For instance, Tyrrells has an affinity index of 1.2281, suggesting a strong preference relative to the broader market.
- Brands like Pringles, Cobs and Infuzions also show positive affinity indices (1.1867, 1.1431 and 1.1334, respectively), indicating a favorable preference among the target segment, but not as pronounced as the top brands.
- Brands such as Cheezels, Sunbites, Smith, and French exhibit affinity indices below 1, suggesting that these brands are less favored by the target segment compared to others. For example, Cheezels has an affinity index of 0.9638, indicating that its popularity among young singles/couples is slightly lower than in other segments.
- A few brands, such as Burger (affinity index of 0.4436) and Woolworths (affinity index of 0.4876), have significantly lower affinity indices, indicating that they are not favored by the young singles/couples in the Mainstream segment compared to the broader customer base. This suggests that marketing efforts for these brands may need reevaluation for this specific demographic.

In short, the Mainstream with young singles/couples is 23% more likely to purchase Tyrrells (1.23) chips compared to the rest of the population. Meanwhile, the Mainstream young singles/couples are 56% less likely to purchase Burger Rings (0.44) compared to the rest of the population.

Recommendations:

1. Targeted Marketing Opportunities: The brands with high affinity indices represent strong candidates for targeted marketing campaigns aimed at young singles/couples, as these customers are likely to respond favorably to promotions or offerings related to these brands.
2. Brand Positioning and Strategy: Brands with lower affinity scores might consider revising their marketing strategies or product offerings to better align with the preferences of this demographic, possibly through targeted advertising, promotional offers, or product variations.

3. Customer Insights: Understanding these preferences can help businesses strategize their inventory and marketing efforts to maximize sales within the young singles/couples segment, ensuring that they cater to the tastes and preferences of this key consumer group.

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

# filter target & other segment
target_seg <- data %>%
  filter(PREMIUM_CUSTOMER == "Mainstream" & LIFESTAGE == "Young Singles/Couples")

other_seg <- data %>%
  filter(!(PREMIUM_CUSTOMER == "Mainstream" & LIFESTAGE == "Young Singles/Couples"))

# Calc total quality
total_quantity_target <- sum(target_seg$PROD_QTY, na.rm = TRUE)
total_quantity_other <- sum(other_seg$PROD_QTY, na.rm = TRUE)

# calc proportion of product size
quantity_target_product_size <- target_seg[, .(Proportion_Target = sum(PROD_QTY, na.rm = TRU
E) / total_quantity_target), by = PACK_SIZE]
quantity_other_product_size <- other_seg[, .(Proportion_Other = sum(PROD_QTY, na.rm = TRUE) /
total_quantity_other), by = PACK_SIZE]

# Merge
size_pack_affinity_comparison <- merge(quantity_target_product_size,
                                       quantity_other_product_size,
                                       by = "PACK_SIZE",
                                       all = TRUE)

# calc affinity index
size_pack_affinity_comparison[, Affinity_Index := Proportion_Target / Proportion_Other]

# Round off
size_pack_affinity_comparison[, `:=`(
  Proportion_Target = round(Proportion_Target, 4),
  Proportion_Other = round(Proportion_Other, 4),
  Affinity_Index = round(Affinity_Index, 4)
)]

# order the Affinity Index
size_pack_affinity_sorted <- size_pack_affinity_comparison[order(-Affinity_Index)]
size_pack_affinity_sorted
```

```
##     PACK_SIZE Proportion_Target Proportion_Other Affinity_Index
##         <num>             <num>            <num>          <num>
##  1:       270            0.0318           0.0251         1.2683
##  2:       380            0.0322           0.0256         1.2570
##  3:       330            0.0613           0.0502         1.2217
##  4:       134            0.1194           0.1006         1.1867
##  5:       110            0.1063           0.0898         1.1836
##  6:       210            0.0291           0.0251         1.1593
##  7:       135            0.0148           0.0131         1.1295
##  8:       250            0.0144           0.0128         1.1232
##  9:       170            0.0808           0.0810         0.9974
## 10:       150            0.1576           0.1634         0.9644
## 11:       175            0.2550           0.2700         0.9444
## 12:       165            0.0557           0.0623         0.8938
## 13:       190            0.0075           0.0124         0.6013
## 14:       180            0.0036           0.0061         0.5915
## 15:       160            0.0064           0.0124         0.5176
## 16:        90            0.0063           0.0126         0.5047
## 17:       125            0.0030           0.0060         0.4984
## 18:       200            0.0090           0.0187         0.4809
## 19:        70            0.0030           0.0063         0.4803
## 20:       220            0.0029           0.0066         0.4436
##     PACK_SIZE Proportion_Target Proportion_Other Affinity_Index
```

Key Insights:

1. Pack sizes with higher affinity

- 270g (1.2683), 380g (1.2570), 330g (1.2217), 134g (1.1867), 110g (1.1836)
- These pack sizes have an Affinity Index greater than 1, indicating that target customers are more likely to buy these pack sizes compared to other customers.
- For example, the 270g pack has an Affinity Index of 1.2683, meaning target customers are about 26.8% more likely to buy this pack size compared to others.

2. Pack sizes with lower affinity

- 170g (0.9974), 150g (0.9644), 175g (0.9444), 165g (0.8938)
- These pack sizes have an Affinity Index close to 1 but slightly less, meaning target customers are about as likely or slightly less likely to buy these sizes compared to others.
- The 175g pack has a lower Affinity Index of 0.9444, meaning target customers are 5.6% less likely to buy this pack size compared to others.

3. Least favored pack sizes by target customers

- 190g (0.6013), 180g (0.5915), 160g (0.5176), 90g (0.5047), 125g (0.4984):
- These pack sizes have significantly lower Affinity Index values, indicating that target customers are much less likely to prefer them. For example, the 190g pack is 39.87% less likely to be purchased by the target customers compared to others.

In short, the Mainstream with young singles/couples is 27% more likely to purchase 270g of chips (1.27) compared to the rest of the population.

```
data[PACK_SIZE == 270, unique(BRAND)]
```

```
## [1] "Twisties"
```

Twisties are the only brand offering 270g packs and so this may instead be reflecting a higher likelihood of purchasing Twisties.

CONCLUSIONS:

Sales have mainly been due to Budget with older families (156863.75), Mainstream with young singles/couples (147582.2), and Mainstream with retirees (145168.95) shoppers. We found that the high spend in chips for budget with older families (21514) and Mainstream with retirees (19970) are due to there being more of them than other buyers. Mainstream, midage and young singles and couples are also more likely to pay more per packet of chips. This is indicative of impulse buying behaviour. We've also found that Mainstream young singles and couples are 23% more likely to purchase Tyrrells (1.23) chips compared to the rest of the population. The Category Manager may want to increase the category's performance by off-locating some Tyrrells and smaller packs of chips in discretionary space near segments where young singles and couples frequent more often to increase visibilty and impulse behaviour.

Quantium can help the Category Manager with recommendations of where these segments are and further help them with measuring the impact of the changed placement. We'll work on measuring the impact of trials in the next task and putting all these together in the third task.