Quantium Virtual Internship - Retail Strategy and Analytics - Task 1

Upasana Sharma

Solution for Task 1

```
#### Loading all necessary libraries
library(data.table)
library(datasets)
library(readr)
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(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

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

```
library(stringr)
library (tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v forcats 0.5.1
## v tidyr   1.2.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::as.difftime() masks base::as.difftime()
## x dplyr::between()         masks data.table::between()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks stats::filter()
## x dplyr::first()           masks data.table::first()
## x lubridate::hour()        masks data.table::hour()
## x lubridate::intersect()   masks base::intersect()
## x lubridate::isoweek()     masks data.table::isoweek()
## x dplyr::lag()             masks stats::lag()
## x dplyr::last()            masks data.table::last()
## x lubridate::mday()        masks data.table::mday()
## x lubridate::minute()      masks data.table::minute()
## x lubridate::month()       masks data.table::month()
## x lubridate::quarter()     masks data.table::quarter()
## x lubridate::second()      masks data.table::second()
## x lubridate::setdiff()     masks base::setdiff()
## x purrr::transpose()       masks data.table::transpose()
## x lubridate::union()       masks base::union()
## x lubridate::wday()        masks data.table::wday()
## x lubridate::week()        masks data.table::week()
## x lubridate::yday()        masks data.table::yday()
## x lubridate::year()        masks data.table::year()
```

```
library(ggplot2)
library(ggmosaic)

#### Opening dataset files
filePath <- "C:/Quantium/"
transactionData <- fread(paste0(filePath,"QVI_transaction_data.csv"))
customerData <- fread(paste0(filePath,"QVI_purchase_behaviour.csv"))
```

Exploratory data analysis

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

Examining transaction data

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

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

```
#### View the structure of transactionData table
str(transactionData)
```

```
## Classes 'data.table' and 'data.frame':   264836 obs. of  8 variables:
##  $ DATE          : int  43390 43599 43605 43329 43330 43604 43601 43601 43332 43330 ...
##  $ STORE_NBR     : int  1 1 1 2 2 4 4 4 5 7 ...
##  $ LYLTY_CARD_NBR: int  1000 1307 1343 2373 2426 4074 4149 4196 5026 7150 ...
##  $ TXN_ID        : int  1 348 383 974 1038 2982 3333 3539 4525 6900 ...
##  $ PROD_NBR      : int  5 66 61 69 108 57 16 24 42 52 ...
##  $ PROD_NAME     : chr  "Natural Chip        Compny SeaSalt175g" "CCs Nacho Cheese    175g" "Sm
iths Crinkle Cut  Chips Chicken 170g" "Smiths Chip Thinly  S/Cream&Onion 175g" ...
##  $ PROD_QTY      : int  2 3 2 5 3 1 1 1 1 2 ...
##  $ TOT_SALES     : num  6 6.3 2.9 15 13.8 5.1 5.7 3.6 3.9 7.2 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
summary(transactionData)
```

```
##       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  
```

```
head(transactionData)
```

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

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

```
#### Converting the date from integer format to date format
transactionData$DATE <- as.Date(transactionData$DATE, origin = "1899-12-30")
```

Lets check that we are looking at the right products by taking a closer look at PROD_NAME.

```
#### Checking the column PROD_NAME by its summary
transactionData[, .N, by = PROD_NAME]
```

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

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

```
#### Checking for incorrect entries other than chips
productWords <- data.table(unlist(strsplit(unique(transactionData[, PROD_NAME]), "
")))
setnames(productWords, 'words')

#### Creating list of products
productInfo <- data.table(PROD_NBR = unique(transactionData$PROD_NBR),
                          PROD_NAME = unique(toupper(transactionData$PROD_NAME)))
productInfo <- productInfo %>% arrange(PROD_NBR)
```

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.

```
#### Filtering data for digits and special characters
transactionData$PROD_NAME = substr(transactionData$PROD_NAME, 1, nchar(transactionData$PROD_NAME))
transactionData$PROD_NAME = gsub("\\s+", " ", transactionData$PROD_NAME)
head(transactionData)
```

```
##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
## 1: 2018-10-17         1           1000      1        5
## 2: 2019-05-14         1           1307    348       66
## 3: 2019-05-20         1           1343    383       61
## 4: 2018-08-17         2           2373    974       69
## 5: 2018-08-18         2           2426   1038      108
## 6: 2019-05-19         4           4074   2982       57
##                                   PROD_NAME PROD_QTY TOT_SALES
## 1:          Natural Chip Compny SeaSalt175g        2       6.0
## 2:                    CCs Nacho Cheese 175g        3       6.3
## 3:    Smiths Crinkle Cut Chips Chicken 170g        2       2.9
## 4:    Smiths Chip Thinly S/Cream&Onion 175g        5      15.0
## 5: Kettle Tortilla ChpsHny&Jlpno Chili 150g        3      13.8
## 6:   Old El Paso Salsa Dip Tomato Mild 300g        1       5.1
```

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

```
#### Remove salsa from the items list
transactionData[, SALSA := grepl("salsa", tolower(PROD_NAME))]
transactionData <- transactionData[SALSA == FALSE, ][, SALSA := NULL]
summary(transactionData)
```

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

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).

```
#### Remove salsa from the items list
transactionData[, SALSA := grepl("salsa", tolower(PROD_NAME))]
transactionData <- transactionData[SALSA == FALSE, ][, SALSA := NULL]
summary(transactionData)
```

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

```
#### Check for nulls
transactionData[is.null(PROD_NAME), .N]
```

```
## [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 data to find the outlier
transactionData[PROD_QTY == 200]
```

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

```
#### See if this customer has had any other transactions
transactionData %>%
  filter(LYLTY_CARD_NBR == 226000)
```

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

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

```
#### Filtering the customer based on the loyalty card number
transactionData = filter(transactionData, LYLTY_CARD_NBR != 226000)
summary(transactionData)
```

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

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.

```
#### Filtering the customer based on the loyalty card number
transactionData = filter(transactionData, LYLTY_CARD_NBR != 226000)
summary(transactionData)
```

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

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.

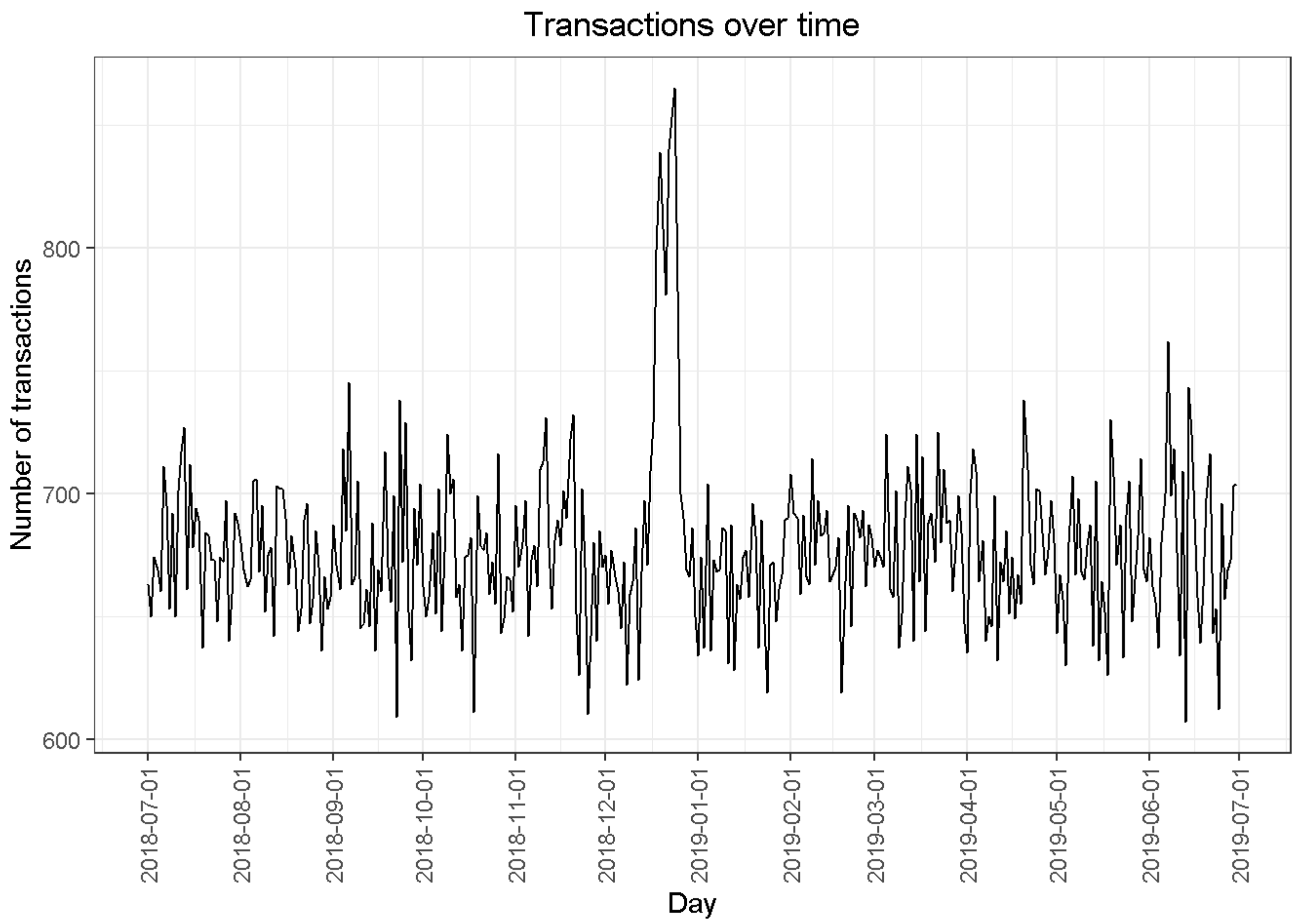
```
#### Create a summary of transaction count by date
transactions_by_date <- transactionData %>%
  group_by(DATE) %>%
  summarise(count = n())
transactions_by_date
```

```
## # A tibble: 364 x 2
##    DATE       count
##    <date>     <int>
##  1 2018-07-01   663
##  2 2018-07-02   650
##  3 2018-07-03   674
##  4 2018-07-04   669
##  5 2018-07-05   660
##  6 2018-07-06   711
##  7 2018-07-07   695
##  8 2018-07-08   653
##  9 2018-07-09   692
## 10 2018-07-10   650
## # ... with 354 more rows
```

```
# Creating a sequence of dates and join this the count of transactions by date
transactions_by_date <- as.data.table(transactionData)
transactions_by_date <- data.frame(transactions_by_date[, .N, by = DATE])
```

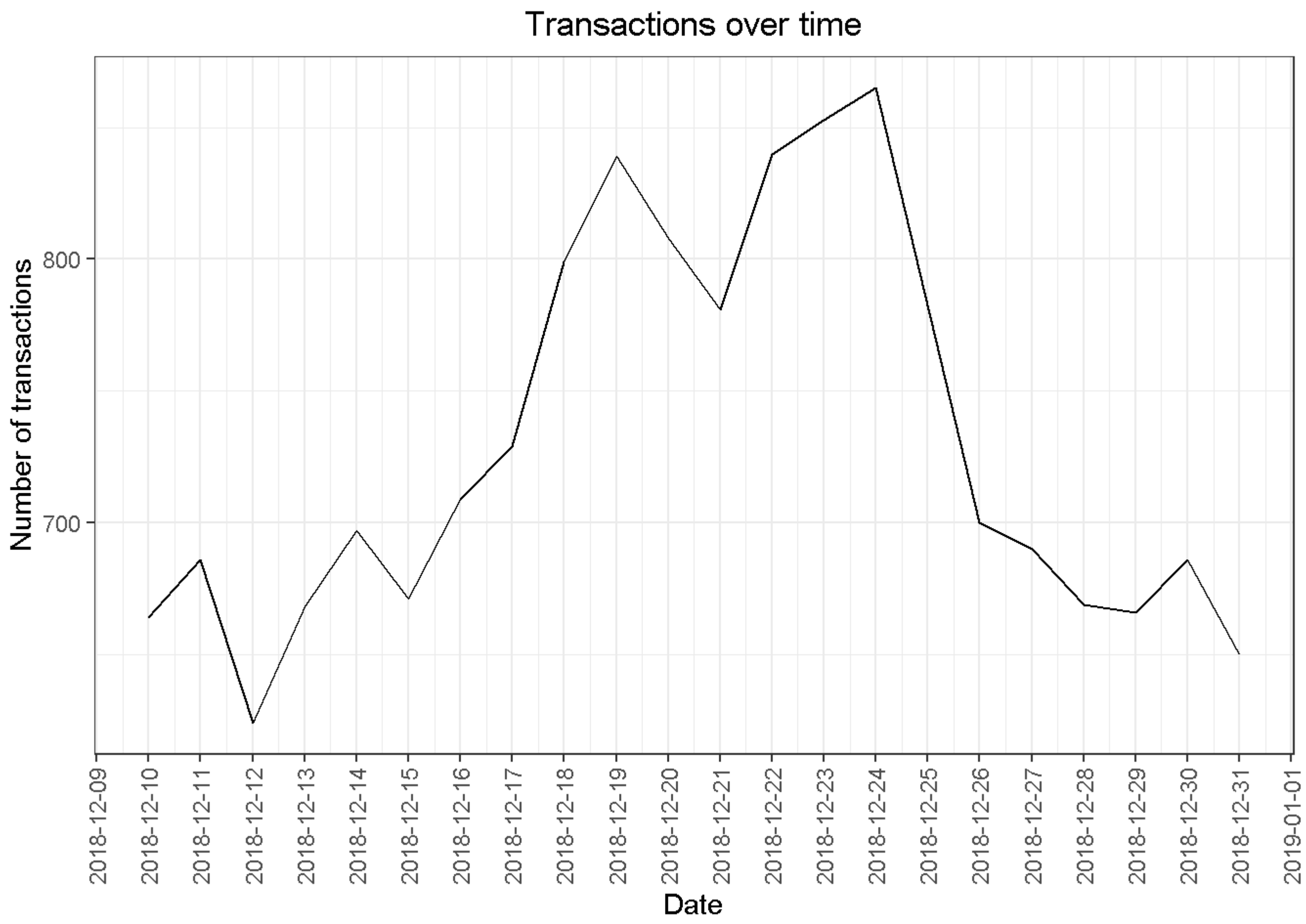
There’s only 364 rows, meaning only 364 dates which indicates a missing date. Let’s create a sequence of dates from 1 Jul 2018 to 30 Jun 2019 and use this to create a chart of number of transactions over time to find the missing date.

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

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



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

```
#### Filter to December and look at individual days
ggplot(transactions_by_date[transactions_by_date$DATE >= "2018-12-10" & transactions_by_date$DATE
        <= "2018-12-31", ], aes(x = DATE, y = N)) +
  geom_line() +
  labs(x = "Date", y = "Number of transactions", title = "Transactions over time") +  scale_x_date
        (breaks = "1 day") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



We can see that the increase in sales occurs in the lead-up to Christmas and that there are zero sales on Christmas day itself. This is due to shops being closed on Christmas day. Now that we are satisfied that the data no longer has outliers, we can move on to creating other features such as brand of chips or pack size from PROD_NAME. We will start with pack size.

```
#### Create pack size
transactionData[, PACK_SIZE := parse_number(PROD_NAME)]
```

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

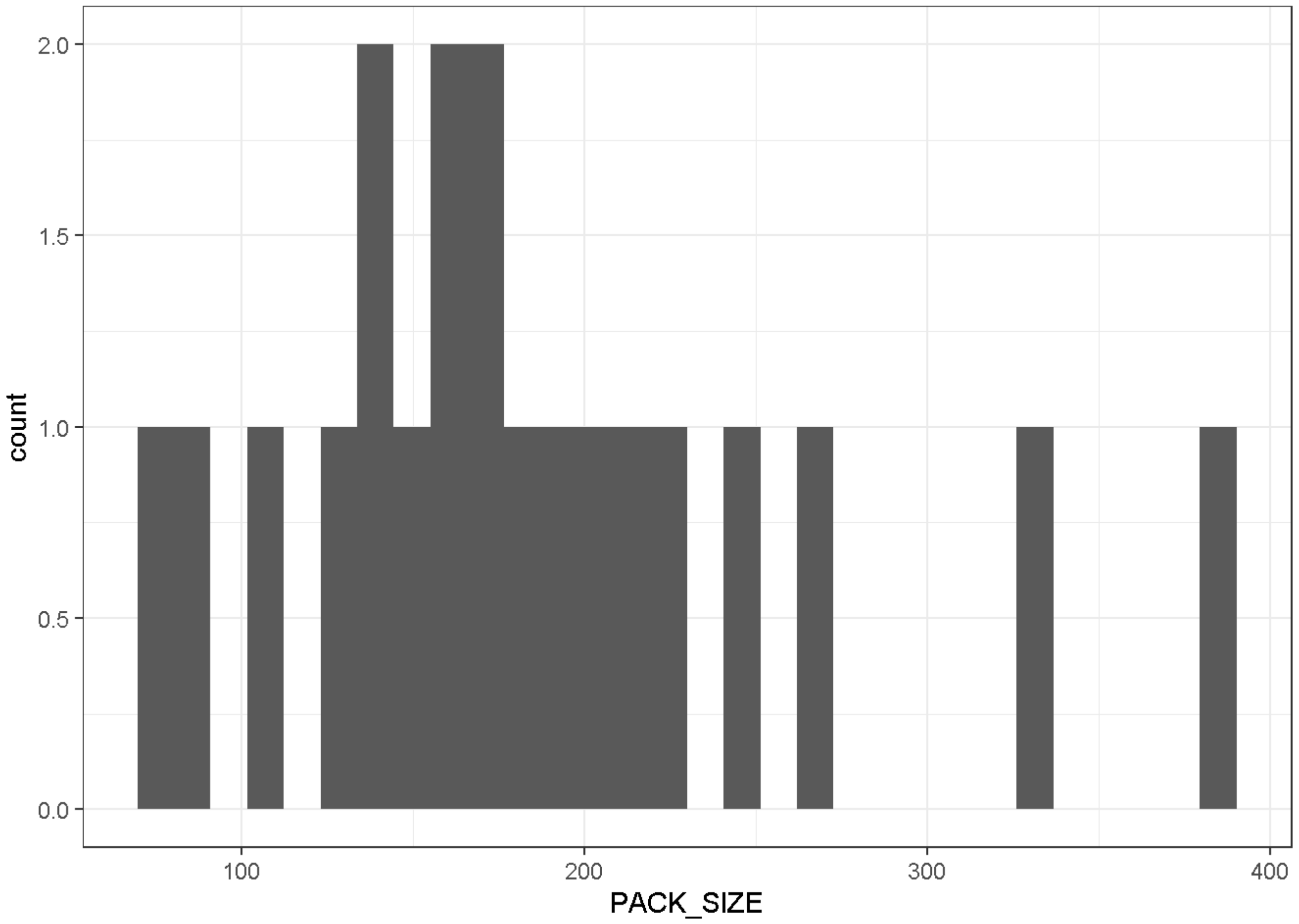
```
#### Checking for any outliers
transactionData[, .N, PACK_SIZE][order(PACK_SIZE)]
```

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

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

```
#### Plotting a histogram showing the number of transactions by pack size
ggplot(transactionData[, .N, by = PACK_SIZE][order(N)], aes(x = PACK_SIZE)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



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

```
#### Creating a column of brand names from the first word of column Product names
transactionData$BRAND <- gsub("([A-Za-z]+).*", "\\1", transactionData$PROD_NAME)
View(transactionData)

#### Identify unique brand names
transactionData[, .N, by = BRAND][order(-N)]
```

```
##          BRAND     N
##  1:     Kettle 41288
##  2:     Smiths 27390
##  3:   Pringles 25102
##  4:    Doritos 22041
##  5:      Thins 14075
##  6:        RRD 11894
##  7:  Infuzions 11057
##  8:         WW 10320
##  9:       Cobs  9693
## 10:   Tostitos  9471
## 11:   Twisties  9454
## 12:   Tyrrells  6442
## 13:      Grain  6272
## 14:    Natural  6050
## 15:   Cheezels  4603
## 16:        CCs  4551
## 17:        Red  4427
## 18:     Dorito  3183
## 19:     Infzns  3144
## 20:      Smith  2963
## 21:    Cheetos  2927
## 22:      Snbts  1576
## 23:     Burger  1564
## 24: Woolworths  1516
## 25:    GrnWves  1468
## 26:   Sunbites  1432
## 27:        NCC  1419
## 28:     French  1418
##          BRAND     N
```

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

```
#### Clean Brand Names
transactionData[BRAND == "RED", BRAND := "RRD"]
transactionData[BRAND == "SNBTS", BRAND := "SUNBITES"]
transactionData[BRAND == "INFZNS", BRAND := "INFUSIONS"]
transactionData[BRAND == "WW", BRAND := "WOOLWORTHS"]
transactionData[BRAND == "SMITH", BRAND := "SMITHS"]
transactionData[BRAND == "NCC", BRAND := "NATURAL"]
transactionData[BRAND == "DORITO", BRAND := "DORITOS"]
transactionData[BRAND == "GRAIN", BRAND := "GRNWVES"]

#### Checking Again
transactionData[, .N, by = BRAND][order(-N)]
```

```
##           BRAND     N
##  1:      Kettle 41288
##  2:      Smiths 27390
##  3:    Pringles 25102
##  4:     Doritos 22041
##  5:       Thins 14075
##  6:         RRD 11894
##  7:   Infuzions 11057
##  8: WOOLWORTHS 10320
##  9:        Cobs  9693
## 10:    Tostitos  9471
## 11:    Twisties  9454
## 12:    Tyrrells  6442
## 13:       Grain  6272
## 14:     Natural  6050
## 15:    Cheezels  4603
## 16:         CCs  4551
## 17:         Red  4427
## 18:      Dorito  3183
## 19:      Infzns  3144
## 20:       Smith  2963
## 21:     Cheetos  2927
## 22:       Snbts  1576
## 23:      Burger  1564
## 24: Woolworths  1516
## 25:     GrnWves  1468
## 26:    Sunbites  1432
## 27:     NATURAL  1419
## 28:      French  1418
##           BRAND     N
```

Examining customer data

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

```
#### View the structure of customerData table
str(customerData)
```

```
## Classes 'data.table' and 'data.frame':   72637 obs. of  3 variables:
##  $ LYLTY_CARD_NBR  : int  1000 1002 1003 1004 1005 1007 1009 1010 1011 1012 ...
##  $ LIFESTAGE       : chr  "YOUNG SINGLES/COUPLES" "YOUNG SINGLES/COUPLES" "YOUNG FAMILIES" "OLD
ER SINGLES/COUPLES" ...
##  $ PREMIUM_CUSTOMER: chr  "Premium" "Mainstream" "Budget" "Mainstream" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
summary(customerData)
```

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

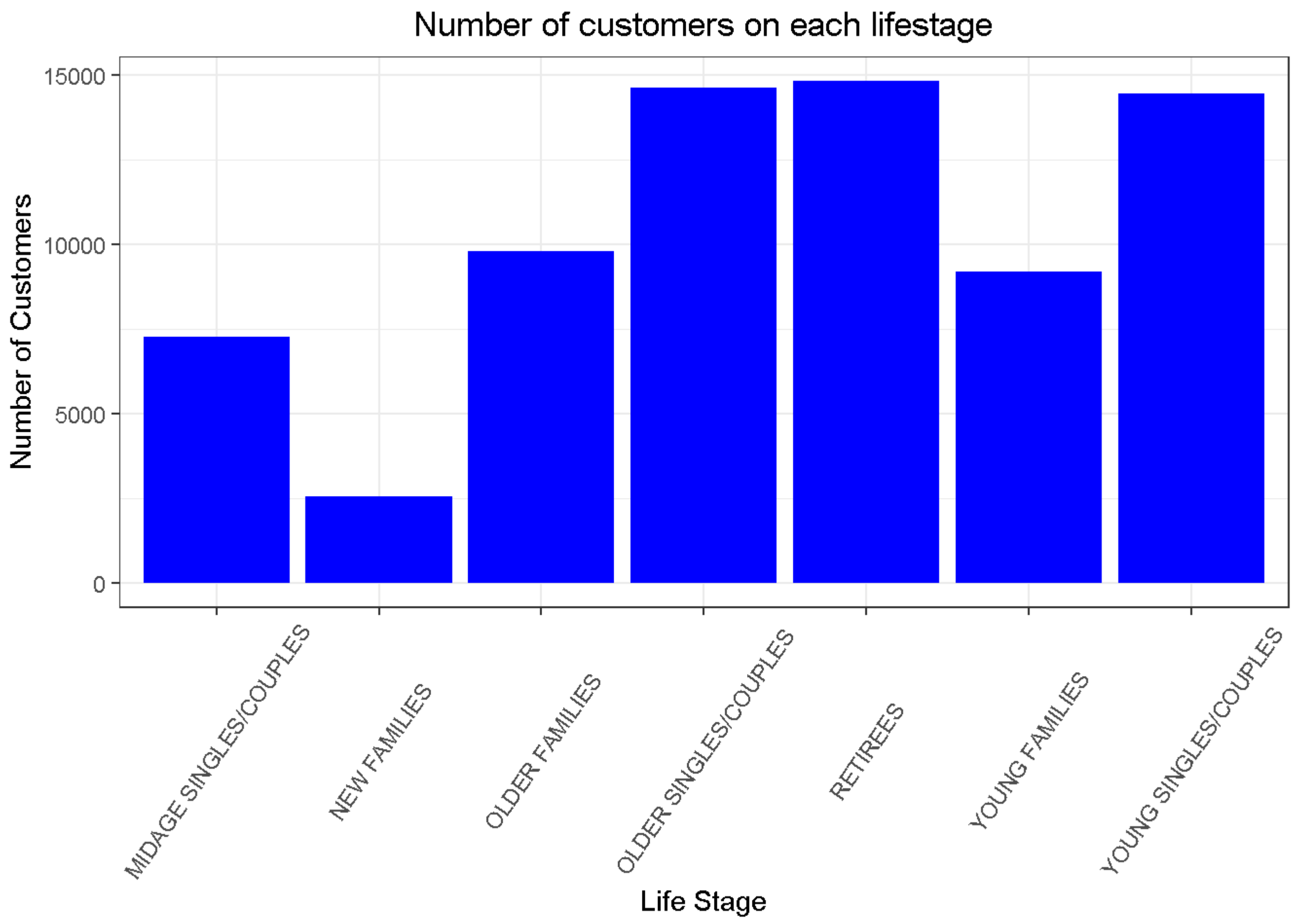
```
head(customerData)
```

```
##    LYLTY_CARD_NBR              LIFESTAGE PREMIUM_CUSTOMER
## 1:           1000  YOUNG SINGLES/COUPLES          Premium
## 2:           1002  YOUNG SINGLES/COUPLES       Mainstream
## 3:           1003         YOUNG FAMILIES           Budget
## 4:           1004  OLDER SINGLES/COUPLES       Mainstream
## 5:           1005 MIDAGE SINGLES/COUPLES       Mainstream
## 6:           1007  YOUNG SINGLES/COUPLES           Budget
```

```
#### Checking customer data by lifestage column
customerData[, .N, by = LIFESTAGE][order(N)]
```

```
##                 LIFESTAGE     N
## 1:           NEW FAMILIES  2549
## 2: MIDAGE SINGLES/COUPLES  7275
## 3:         YOUNG FAMILIES  9178
## 4:         OLDER FAMILIES  9780
## 5:  YOUNG SINGLES/COUPLES 14441
## 6:  OLDER SINGLES/COUPLES 14609
## 7:               RETIREES 14805
```

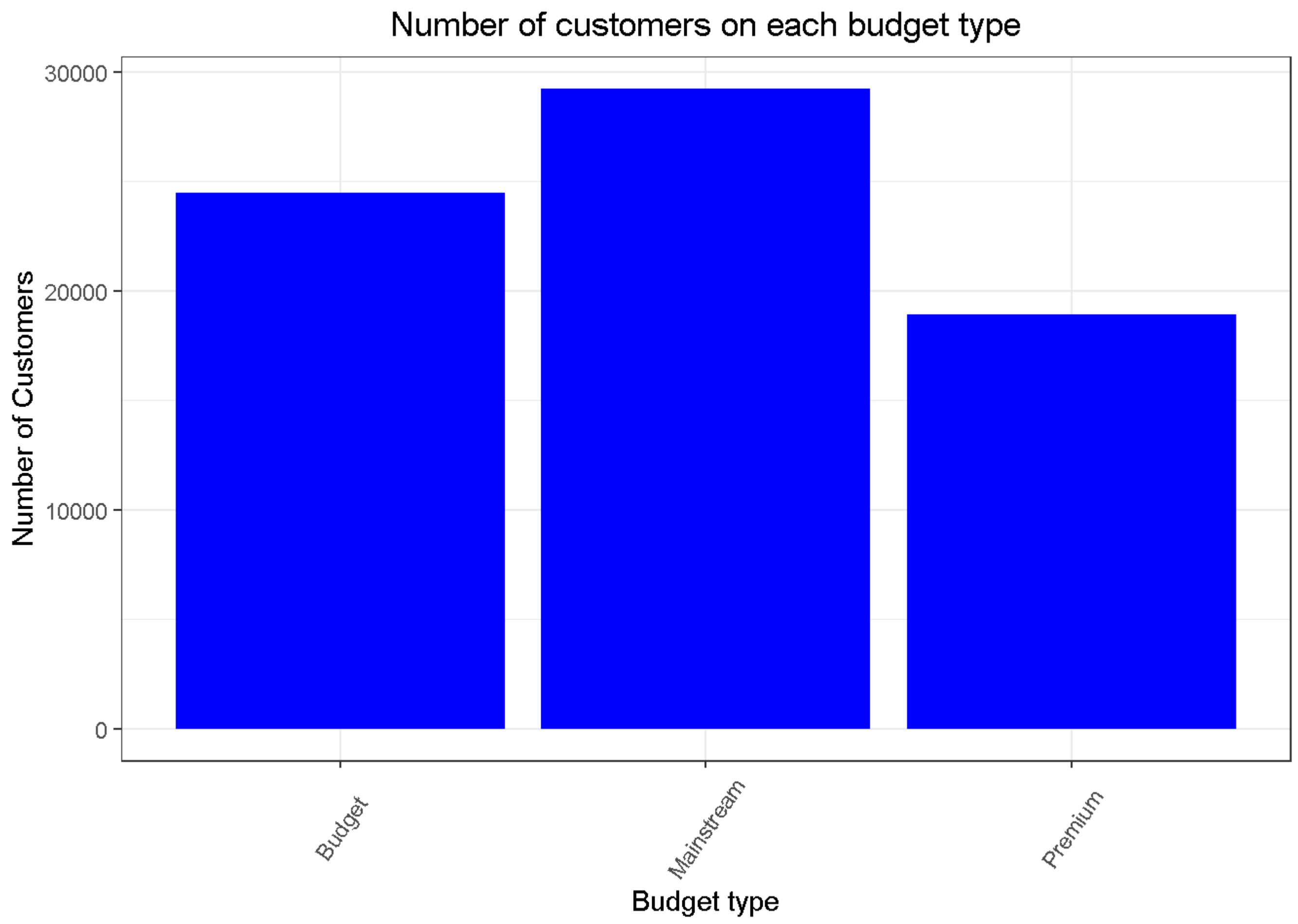
```
#### Plotting a graph
ggplot(customerData[, .N, by = LIFESTAGE][order(N)], aes(x = LIFESTAGE, y = N)) +
  geom_bar(stat = "identity", fill = "blue") + labs(x= "Life Stage", y = "Number of Customers", ti
         tle = "Number of customers on each lifestage") +
  theme(axis.text.x = element_text(angle = 55, vjust = 0.5))
```



```
#### Checking customer data by permium customers column
customerData[, .N, by = PREMIUM_CUSTOMER][order(N)]
```

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

```
#### Plotting a graph
ggplot(customerData[, .N, by = PREMIUM_CUSTOMER][order(N)], aes(x = PREMIUM_CUSTOMER, y = N)) +
  geom_bar(stat = "identity", fill = "blue") + labs(x= "Budget type", y = "Number of Customers", t
         itle = "Number of customers on each budget type") +
  theme(axis.text.x = element_text(angle = 55, vjust = 0.5))
```



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

```
#### Checking for Nulls in data
data[is.null(LIFESTAGE), .N]
```

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

```
data[is.null(PREMIUM_CUSTOMER), .N]
```

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

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

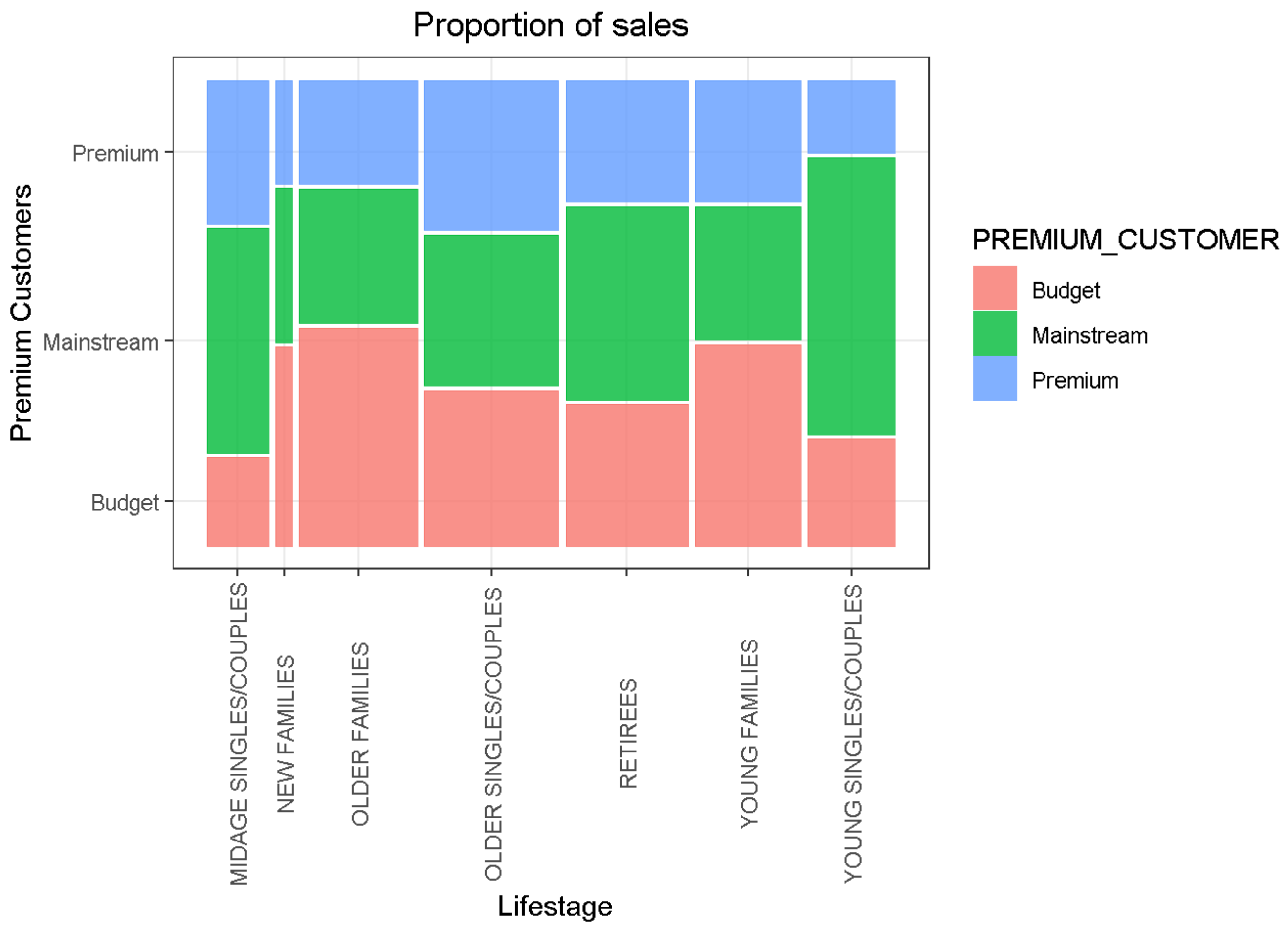
Data exploration is now complete!

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.

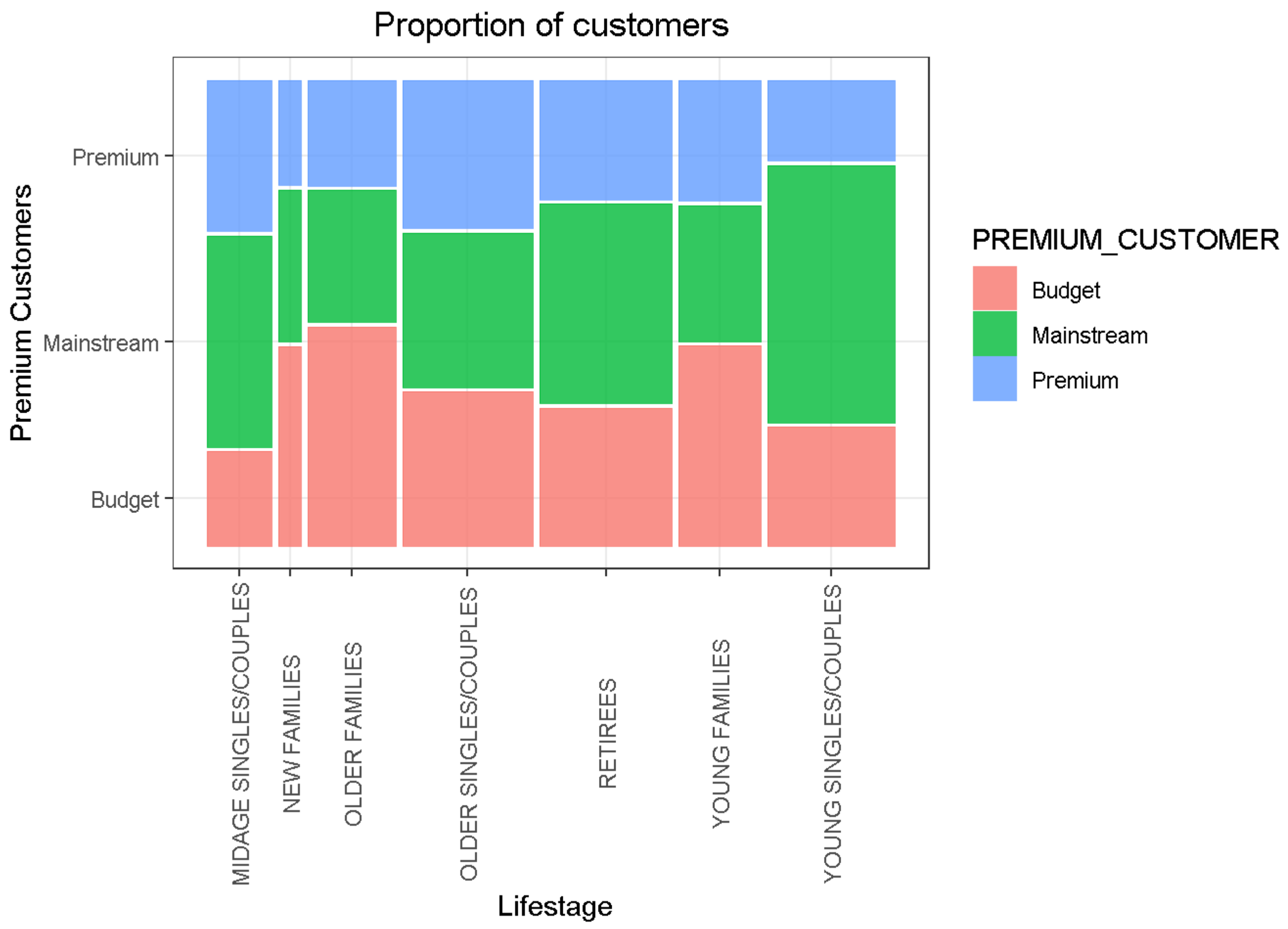
```
#### calculating total sales by LIFESTAGE and PREMIUM_CUSTOMER
sales <- data[, .(SALES = sum(TOT_SALES)), .(LIFESTAGE, PREMIUM_CUSTOMER)]

#### Plotting a graph
ggplot(data = sales) +
  geom_mosaic(aes(weight = SALES, x = product(PREMIUM_CUSTOMER, LIFESTAGE),
  fill = PREMIUM_CUSTOMER)) +
  labs(x = " Lifestage", y = "Premium Customers", title = "Proportion of sales") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```

```
## Warning: `unite_()` was deprecated in tidyr 1.2.0.
## Please use `unite()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```



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.

```
#### Calculating number of customers by LIFESTAGE and PREMIUM_CUSTOMER
customers <- data[, .(CUSTOMERS = uniqueN(LYLTY_CARD_NBR)), .(LIFESTAGE, PREMIUM_CUSTOMER)][order(
         -CUSTOMERS)]

#### Plotting a graph
ggplot(data = customers) +
  geom_mosaic(aes(weight = CUSTOMERS, x = product(PREMIUM_CUSTOMER, LIFESTAGE),
                fill = PREMIUM_CUSTOMER)) +
  labs(x = " Lifestage", y = "Premium Customers", title = "Proportion of customers") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```

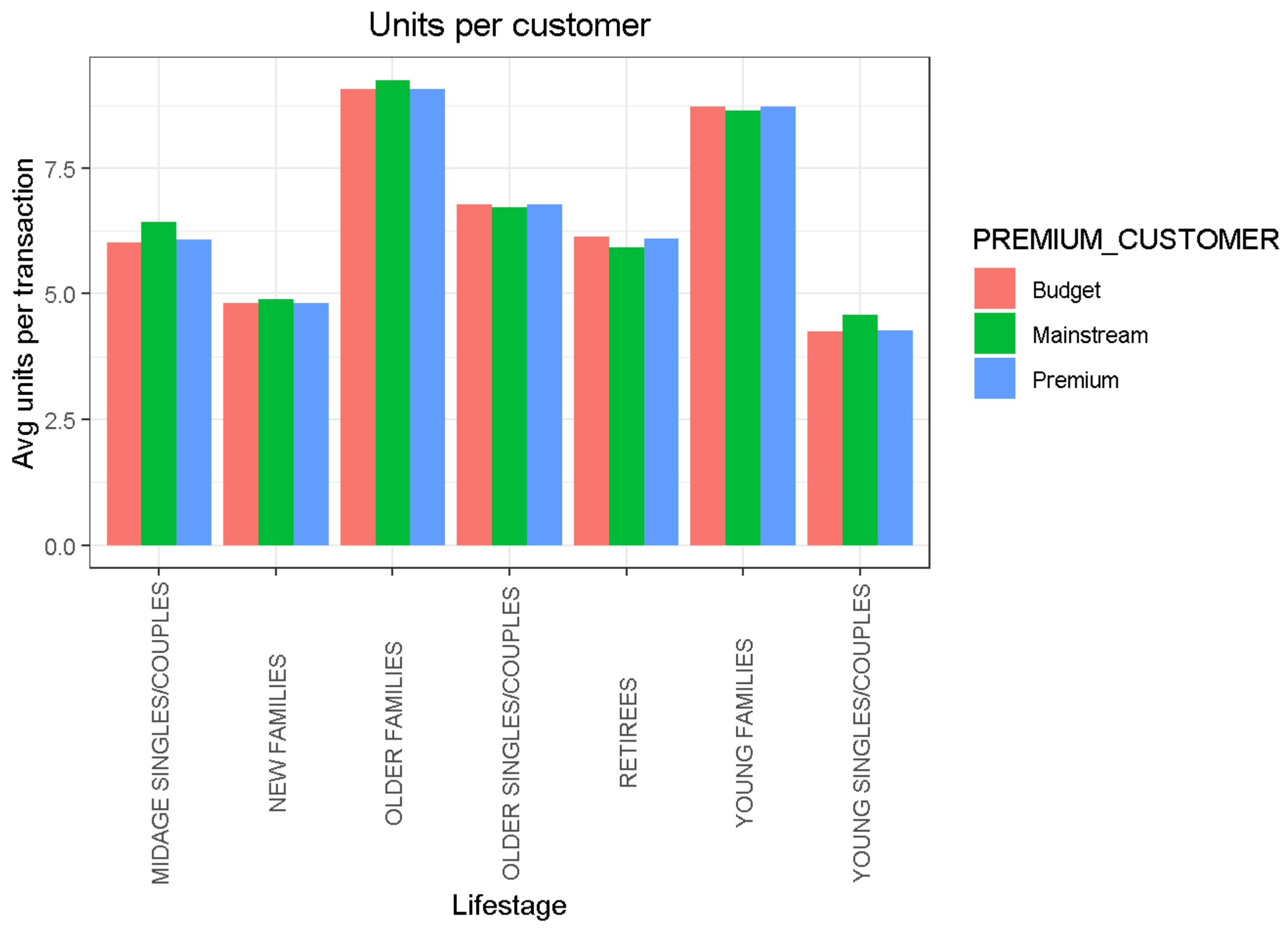


There are more Mainstream - young singles/couples and Mainstream - retirees who buy chips. This contributes to there being more sales to these customer segments but this is not a major driver for the Budget - Older families segment.

Higher sales may also be driven by more units of chips being bought per customer.

Let’s have a look at this next.

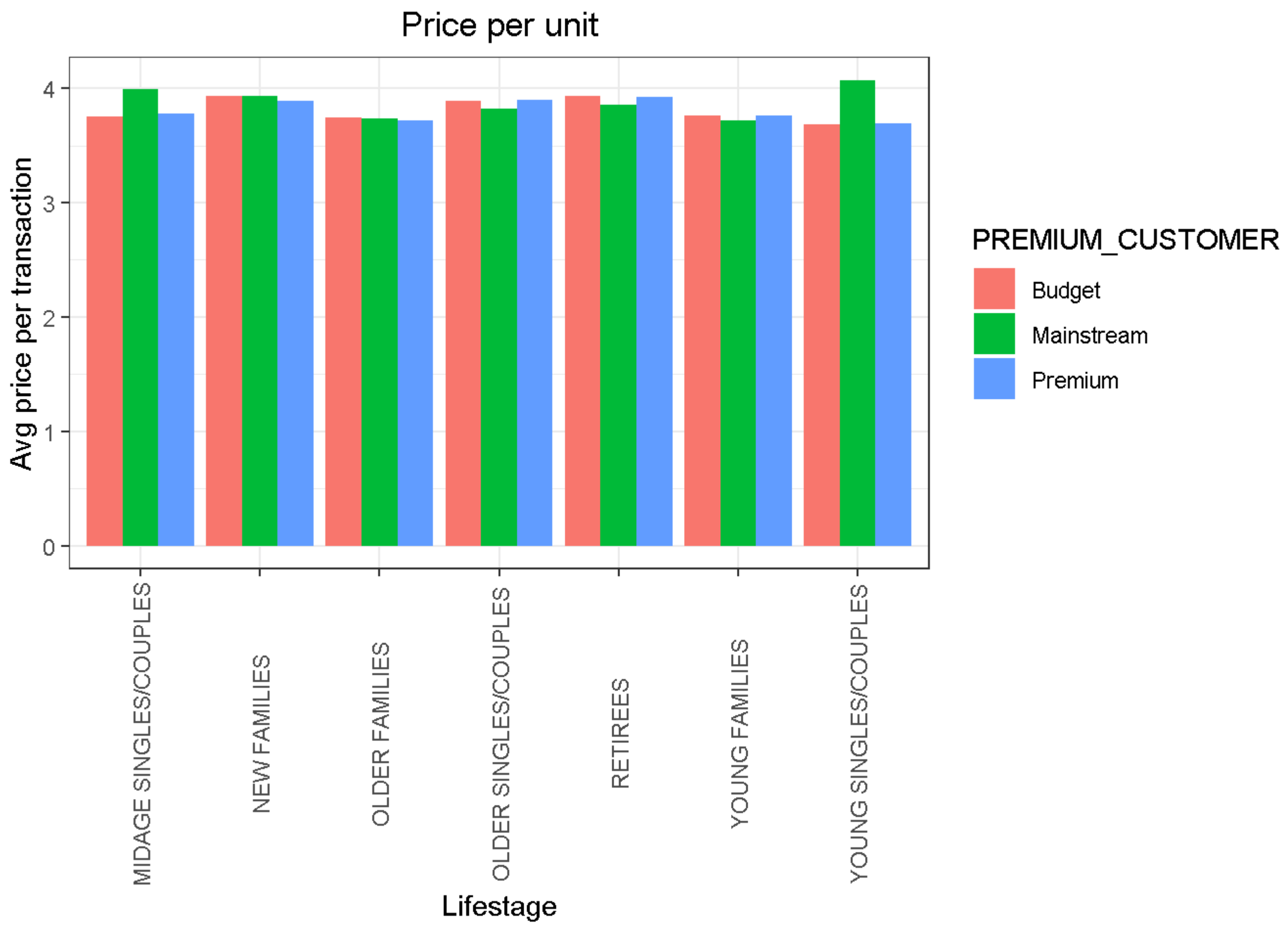
```
#### Average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER
avg_units <- data[, .(AVERAGE = sum(PROD_QTY) / uniqueN(LYLTY_CARD_NBR)), .(LIFESTAGE, PREMIUM_CUS
         TOMER)][order(-AVERAGE)]

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



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.

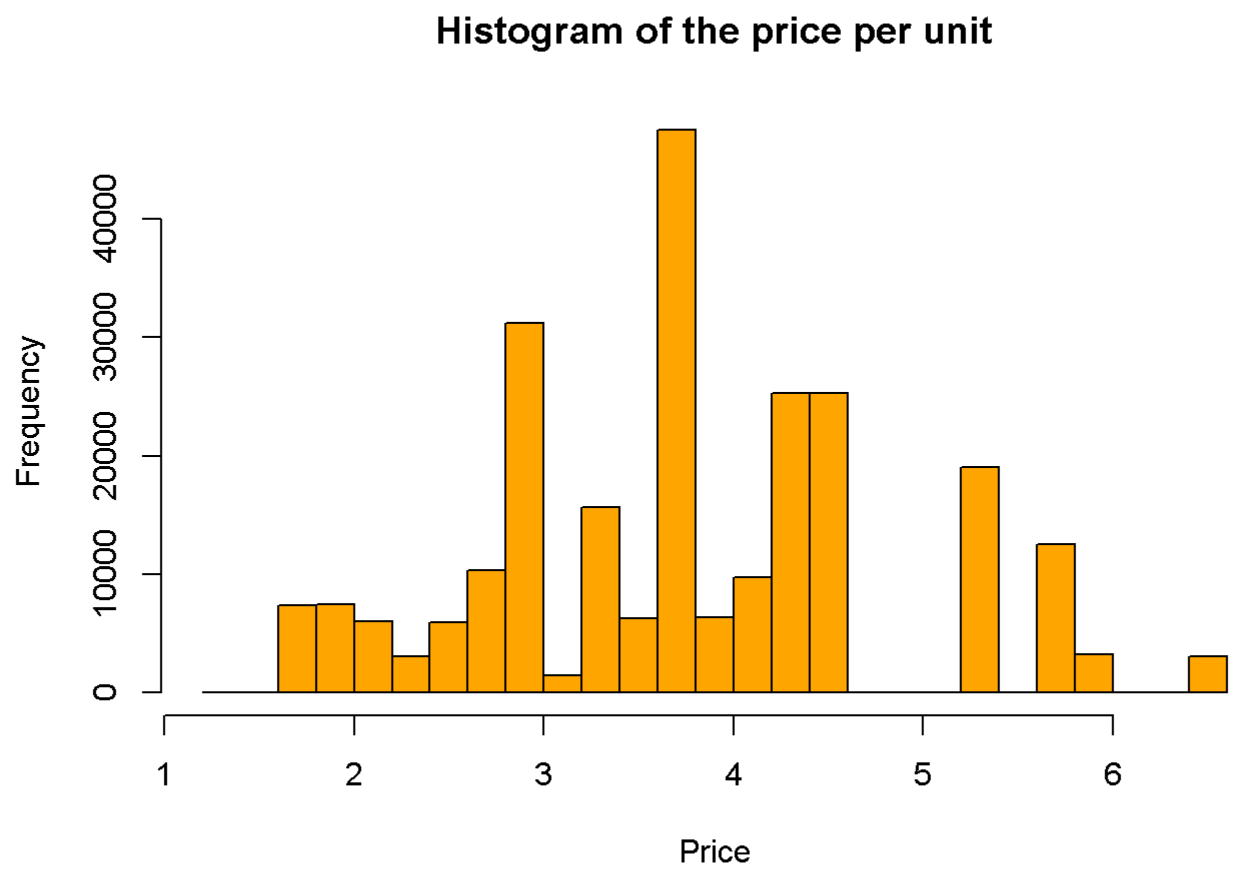
```
#### Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER
avg_price <- data[, .(AVERAGE = sum(TOT_SALES) / sum(PROD_QTY)), .(LIFESTAGE, PREMIUM_CUSTOMER)][o
         rder(-AVERAGE)]

#### Plotting a graph
ggplot(data = avg_price, aes(weight = AVERAGE, x = LIFESTAGE, fill = PREMIUM_CUSTOMER)) +
  geom_bar(position = position_dodge()) +
  labs(x = "Lifestage", y = "Avg price per transaction", title = "Price per unit") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



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 is not large, we can check if this difference is statistically significant. To do so, I will perform an independent t-test between mainstream vs premium midage young singles and couplesto see if thedifference is significant.Our data will yield revelant results about the statistical significance of the price diference. we have all the data which uses ordinal scale as measurement applied to the data, if we plot price data it result on a normal distribution, we can rely on the results with reasonable assurance.

```
priceperunit <- data[, price := TOT_SALES / PROD_QTY]
hist(priceperunit[, price], main = "Histogram of the price per unit",
     xlab = "Price", col = "orange")
```



Let's proceed with the t-test

```
#### Performing an independent t-test between mainstream vs premium and budget midage and young si
         ngles and couples
t.test(data[LIFESTAGE %in% c("YOUNG SINGELS/COUPLES", "MIDAGE SINGLES/COUPLES")
            & PREMIUM_CUSTOMER == "Mainstream", price]
       , data[LIFESTAGE %in% c("YOUNG SINGELS/COUPLES", "MIDAGE SINGLES/COUPLES")
            & PREMIUM_CUSTOMER != "Mainstream", price],
       alternative = "greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  data[LIFESTAGE %in% c("YOUNG SINGELS/COUPLES", "MIDAGE SINGLES/COUPLES") & PREMIUM_CUSTO
MER == "Mainstream", price] and data[LIFESTAGE %in% c("YOUNG SINGELS/COUPLES", "MIDAGE SINGLES/COU
PLES") & PREMIUM_CUSTOMER != "Mainstream", price]
## t = 16.864, df = 23345, p-value < 2.2e-16
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.2111567       Inf
## sample estimates:
## mean of x mean of y
##  3.994241  3.760262
```

The t-test results in a p-value < 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
segment_1 <- data[LIFESTAGE == "YOUNG SINGELS/COUPLES" & PREMIUM_CUSTOMER == "Mainstream", ]
others <- data[!(LIFESTAGE == "YOUNG SINGELS/COUPLES" & PREMIUM_CUSTOMER == "Mainstream"), ]

#### Comparing brand affinity to the rest of the population
quantity_segment_1 <- segment_1[, sum(PROD_QTY)]
quantity_others <- others[, sum(PROD_QTY)]
quantity_segment_1_by_brand <- segment_1[, .(targetSegment = sum(PROD_QTY) / quantity_segment_1),
          by = BRAND]
quantity_others_by_brand <- others[, .(others = sum(PROD_QTY) / quantity_others), by = BRAND]
brand_proportions <- merge(quantity_segment_1_by_brand, quantity_others_by_brand)[, affinityToBran
         d := targetSegment / others]
brand_proportions[order(-affinityToBrand)]
```

```
## Empty data.table (0 rows and 4 cols): BRAND,targetSegment,others,affinityToBrand
```

We can see that :

• Mainstream young singles/couples are 23% more likely to purchase Tyrrells chips compared to the rest of the population.

• Mainstream young singles/couples are 56% less likely to purchase Burger Rings compared to the rest of the population

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
quantity_segment_1_by_pack <- segment_1[, .(targetSegment = sum(PROD_QTY) / quantity_segment_1), b
         y = PACK_SIZE]
quantity_others_by_pack <- others[, .(others = sum(PROD_QTY) / quantity_others), by = PACK_SIZE]
pack_proportions <- merge(quantity_segment_1_by_pack, quantity_others_by_pack)[, affinityToPack :=
         targetSegment / others]
pack_proportions[order(-affinityToPack)]
```

```
## Empty data.table (0 rows and 4 cols): PACK_SIZE,targetSegment,others,affinityToPack
```

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

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

```
## [1] "Twisties Cheese 270g" "Twisties Chicken270g"
```

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

Conclusion.

Let’s recap what we’ve found!

Sales have mainly been due to Budget - older families, Mainstream - young singles/couples, and Mainstream - retirees shoppers. We found that the high spend in chips for mainstream young singles/couples and retirees is due to there being more of them than other buyers. Mainstream, midage and young singles and couples are also more likely to pay more per packet of chips. This is indicative of impulse buying behaviour. We’ve also found that Mainstream young singles and couples are 23% more likely to purchase Tyrrells chips compared to the rest of the population. 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.