Quantium Virtual Internship - Retail Strategy and Analytics - Task 1

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

```
## Warning: package 'data.table' was built under R version 4.1.3
```

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

```r
library(ggmosaic)
```

```
## Warning: package 'ggmosaic' was built under R version 4.1.3
```

```r
library(readr)
```

```
## Warning: package 'readr' was built under R version 4.1.3
```

```r
filePath <- "D:/DOC/"
transactionData <- fread(paste0(filePath,"QVI_transaction_data.csv"))
customerData <- fread(paste0(filePath,"QVI_purchase_behaviour.csv"))
```

Exploratory data analysis

Examining transaction data

```r
#### Examine transaction data
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"
"Smiths 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>
```

```
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.

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

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

```
#### Examine PROD_NAME
str(transactionData$PROD_NAME)
```

```
##  chr [1:264836] "Natural Chip        Compny SeaSalt175g" ...
```

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

```
## [1] "Natural Chip        Compny SeaSalt175g"
## [2] "CCs Nacho Cheese    175g"
## [3] "Smiths Crinkle Cut  Chips Chicken 170g"
## [4] "Smiths Chip Thinly  S/Cream&Onion 175g"
## [5] "Kettle Tortilla ChpsHny&Jlpno Chili 150g"
## [6] "Old El Paso Salsa   Dip Tomato Mild 300g"
```

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

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

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

```
# Remove digits, and special characters, and then sort the distinct words by frequency of occurrence.
#### Removing digits
productWords <- productWords[grepl("\\d", words) == FALSE, ]

#### Removing special characters
productWords <- productWords[grepl("[:alpha:]" , words), ]

#### Let's look at the most common words by counting the number of times a word appears and
#### sorting them by this frequency in order of highest to lowest frequency
productWords[, .N, words][order(N, decreasing = TRUE)]
```

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

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

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

Next, we can use `summary()` to check summary statistics such as mean, min and max values for each feature to see if there are any obvious outliers in the data and if there are any nulls in any of the columns

```
#### Summarise the data to check for nulls and possible outliers
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
```

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

```
#### Filter the dataset to find the outlier
transactionData[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
```

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

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

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

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

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

#### Re-examine transaction data
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
```

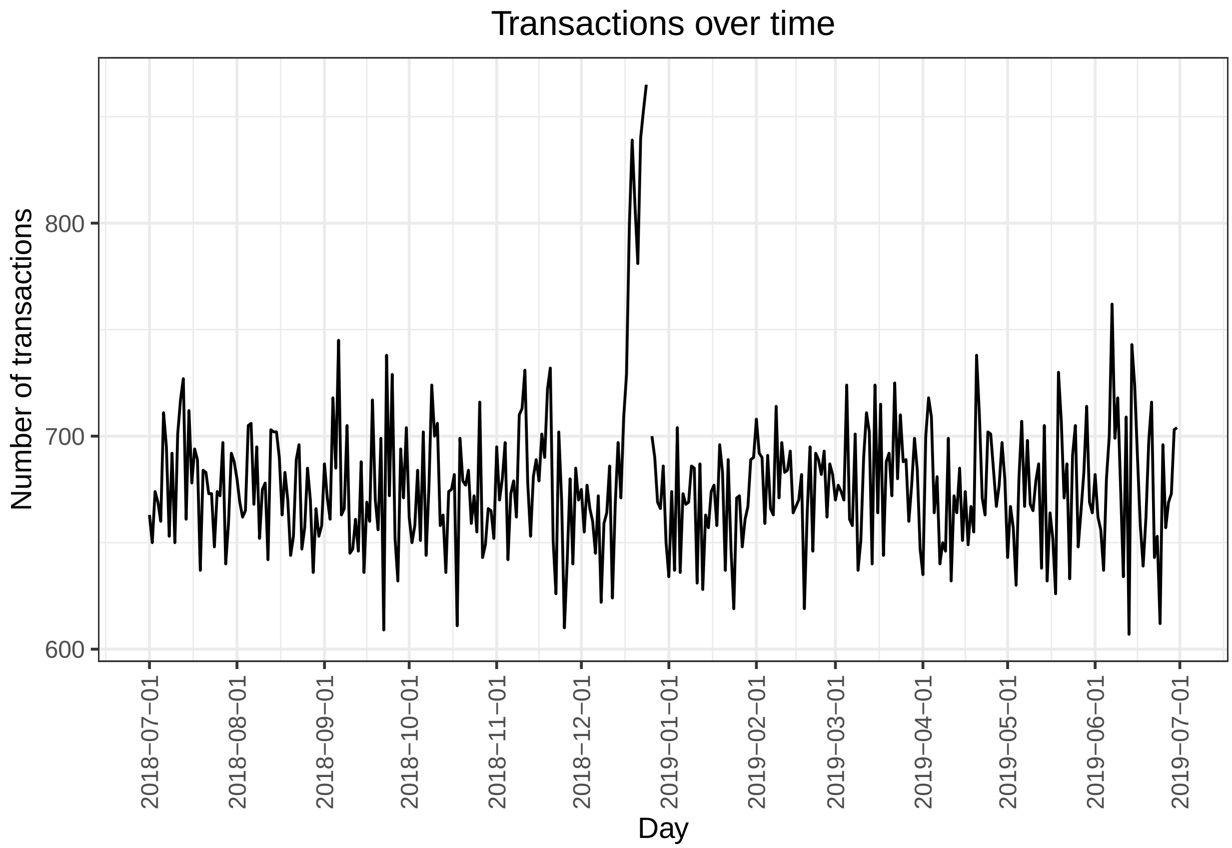
That's better. Now, let's look at the number of transaction lines over time to see if there are any obvious data issues such as missing data.

```r
#### Count the number of transactions by date
transactions_by_day <- transactionData[, .N, DATE][order(DATE)]
transactions_by_day
```

```
##            DATE   N
##   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
##  ---
## 360: 2019-06-26 657
## 361: 2019-06-27 669
## 362: 2019-06-28 673
## 363: 2019-06-29 703
## 364: 2019-06-30 704
```

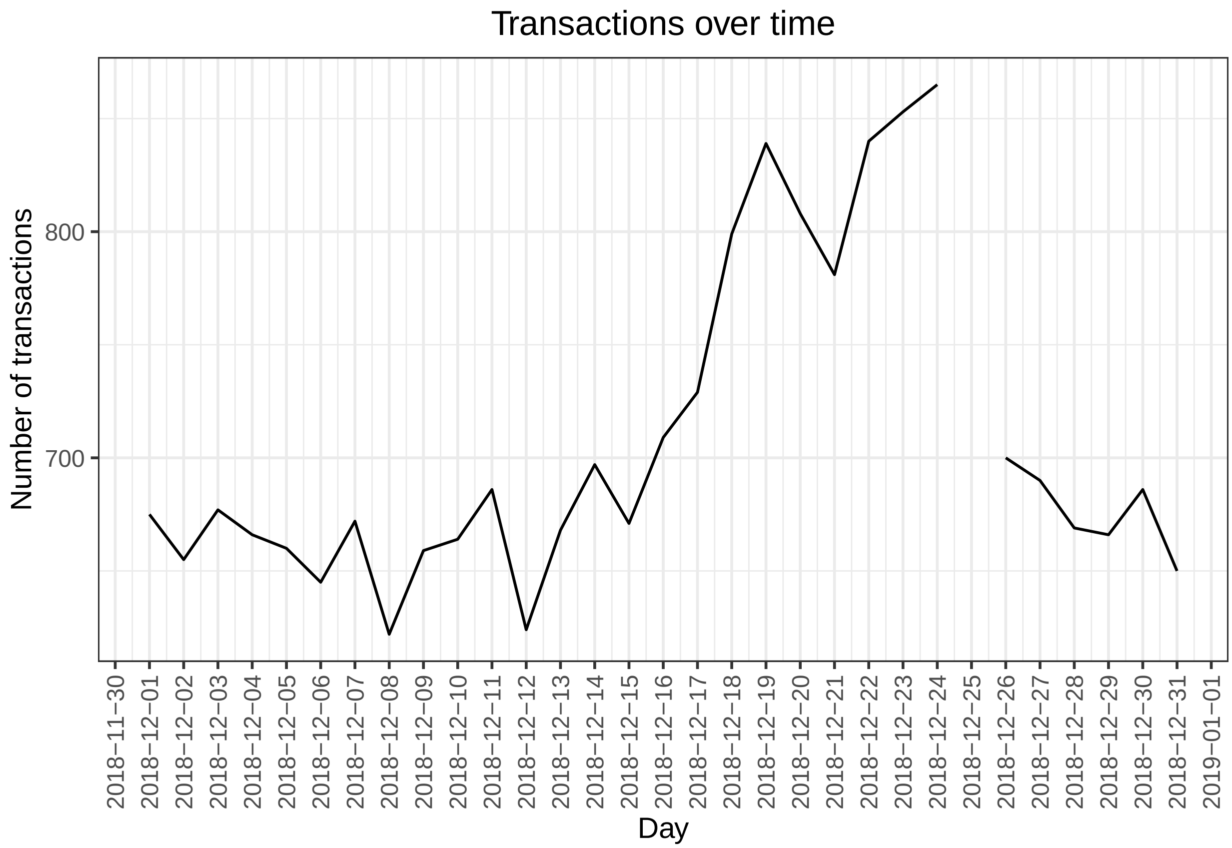
There's only 364 rows, meaning only 364 dates which indicates a missing date. Let's create a sequence of dates from 1 Jul 2018 to 30 Jun 2019 and use this to create a chart of number of transactions over time to find the missing date.

```r
#### Create a sequence of dates and join this the count of transactions by date
# create a column of dates that includes every day from 1 Jul 2018 to 30 Jun 2019, and join it onto the
new_date <- data.frame(DATE=as.Date(seq(as.Date("2018-07-01"), as.Date("2019-06-30"), "day")))
transactions_by_day <- merge(x=new_date,y=transactions_by_day,by="DATE",all.x=TRUE)
#### Setting plot themes to format graphs
theme_set(theme_bw())
theme_update(plot.title = element_text(hjust = 0.5))
#### Plot transactions over time
ggplot(transactions_by_day, aes(x = DATE, y = N)) +
 geom_line() +
 labs(x = "Day", y = "Number of transactions", title = "Transactions over time") +
 scale_x_date(breaks = "1 month") +
 theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



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

```
dec <- transactions_by_day[month(transactions_by_day$DATE) == 12, ]
#### Setting plot themes to format graphs
theme_set(theme_bw())
theme_update(plot.title = element_text(hjust = 0.5))
#### Plot transactions over time
ggplot(dec, aes(x = DATE, y = N)) +
 geom_line() +
 labs(x = "Day", y = "Number of transactions", title = "Transactions over time") +
 scale_x_date(breaks = "1 day") +
 theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



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

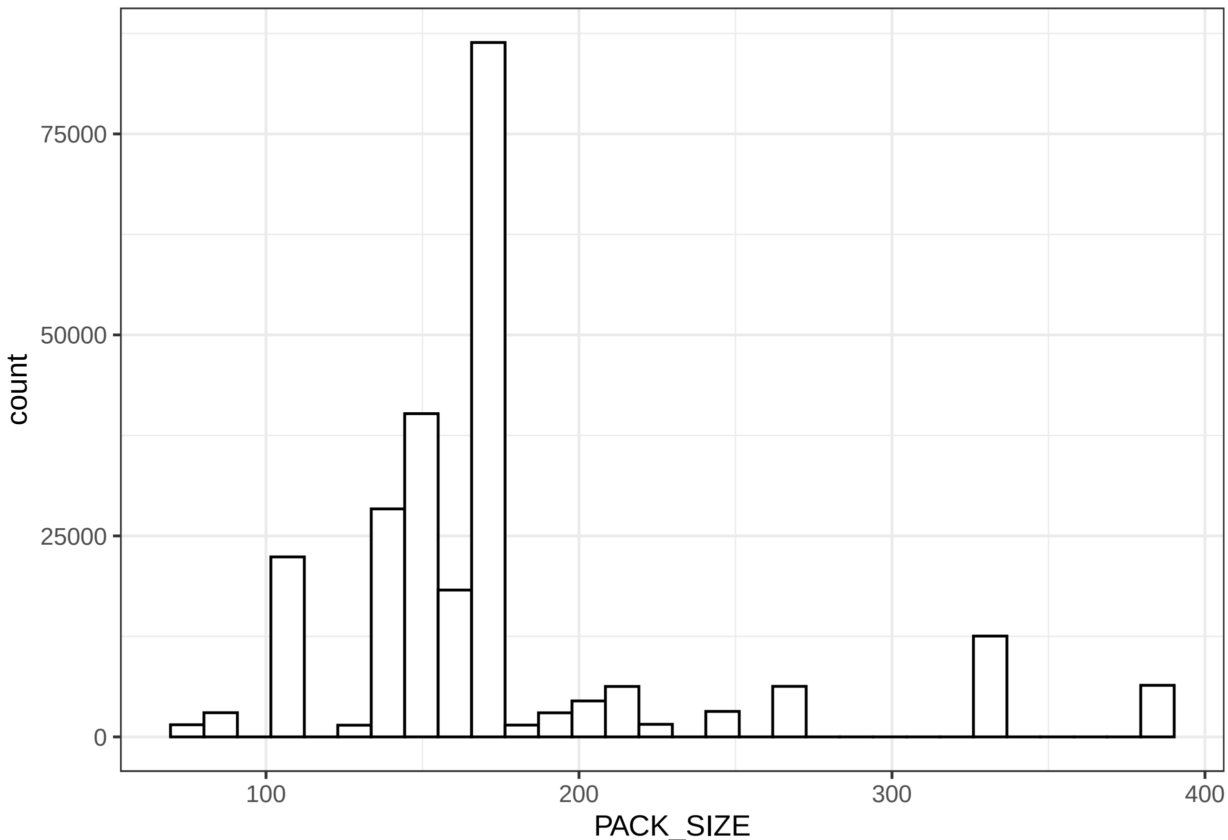
```
#### Pack size
#### We can work this out by taking the digits that are in PROD_NAME
transactionData[, PACK_SIZE := parse_number(PROD_NAME)]
#### Always check your output
#### Let's check if the pack sizes look sensible
transactionData[, .N, PACK_SIZE][order(PACK_SIZE)]
```

```
##     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!

```
#### Let's plot a histogram of PACK_SIZE since we know that it is a categorical variable and not a cont
ggplot(transactionData, aes(x=PACK_SIZE))+
  geom_histogram(colour="black", fill="white", bins = 30)
```



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

```
#### Brands
# Create a column which contains the brand of the product, by extracting it from the product name.
transactionData[, BRAND := toupper(substr(PROD_NAME, 1, regexpr(pattern = ' ', PROD_NAME) -1))]
#### Checking brands
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 := "INFUZIONS"]
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"]

#### Check again
transactionData[, .N, BRAND][order(N)]
```

```
##          BRAND     N
##  1:     FRENCH  1418
##  2:     BURGER  1564
##  3:    CHEETOS  2927
##  4:   SUNBITES  3008
##  5:        CCS  4551
##  6:   CHEEZELS  4603
##  7:   TYRRELLS  6442
```

```
##  8:    NATURAL  7469
##  9:    GRNWVES  7740
## 10:   TWISTIES  9454
## 11:   TOSTITOS  9471
## 12:       COBS  9693
## 13: WOOLWORTHS 11836
## 14:      THINS 14075
## 15:  INFUZIONS 14201
## 16:        RRD 16321
## 17:   PRINGLES 25102
## 18:    DORITOS 25224
## 19:     SMITHS 30353
## 20:     KETTLE 41288
```

Examining customer data

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

```
#### Examining customer data
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
```

```
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
```

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

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

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

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

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

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

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

```r
lapply(data, function(x)sum(is.na(x)))
```

```
## $LYLTY_CARD_NBR
## [1] 0
##
## $DATE
## [1] 0
##
## $STORE_NBR
## [1] 0
##
## $TXN_ID
## [1] 0
##
## $PROD_NBR
## [1] 0
##
## $PROD_NAME
## [1] 0
##
## $PROD_QTY
## [1] 0
##
## $TOT_SALES
## [1] 0
##
## $PACK_SIZE
## [1] 0
##
## $BRAND
## [1] 0
##
## $LIFESTAGE
## [1] 0
##
## $PREMIUM_CUSTOMER
## [1] 0
```

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

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

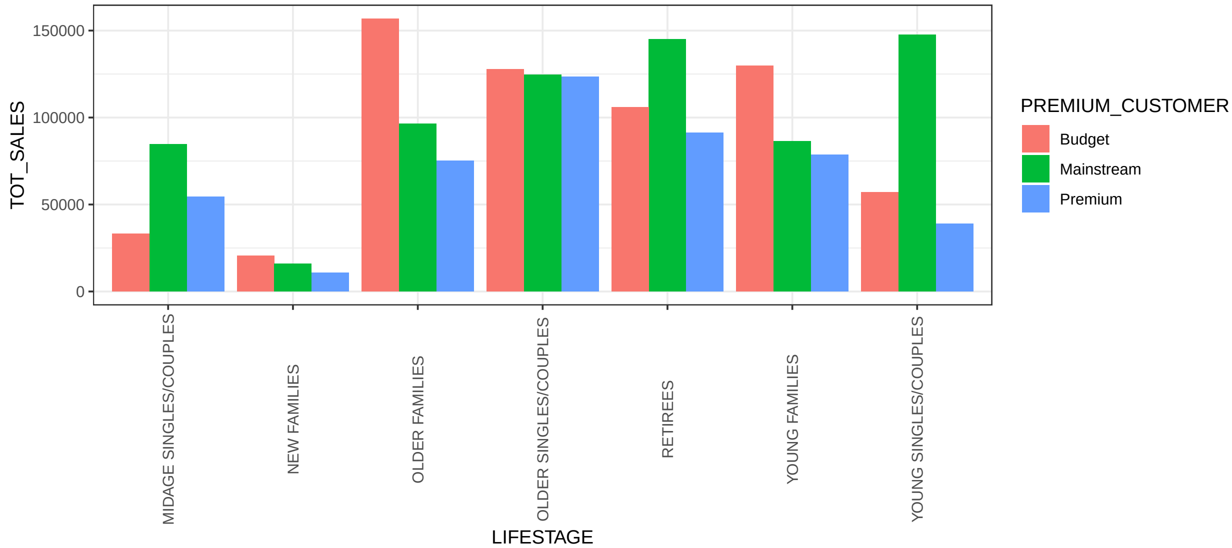
Data analysis on customer segments

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

```
#### Total sales by LIFESTAGE and PREMIUM_CUSTOMER
(sales <- data[, .(TOT_SALES = sum(TOT_SALES)), .(LIFESTAGE, PREMIUM_CUSTOMER)][order(-TOT_SALES)])
```

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

```
ggplot(sales, aes(x = LIFESTAGE, y = TOT_SALES, fill = PREMIUM_CUSTOMER))+
  geom_col(position = "dodge")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



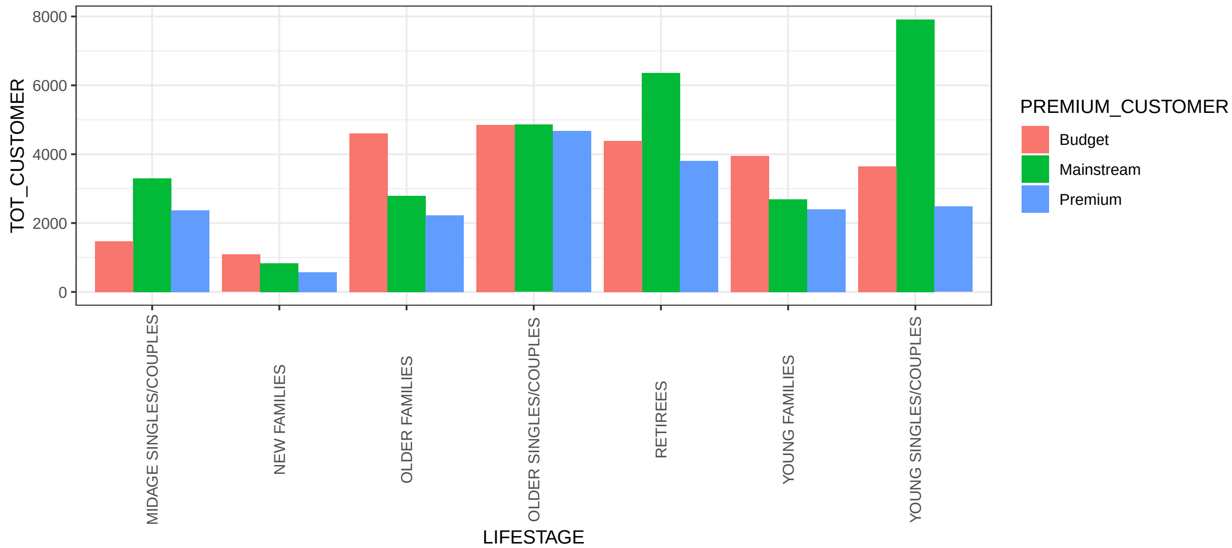
Sales are coming mainly from Budget - older families, Mainstream - young singles/couples, and Mainstream - retirees

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

```
#### Number of customers by LIFESTAGE and PREMIUM_CUSTOMER
(customer <- data[, .(TOT_CUSTOMER=uniqueN(LYLTY_CARD_NBR)), .(LIFESTAGE, PREMIUM_CUSTOMER)][order(-TOT
```

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

```
ggplot(customer, aes(x = LIFESTAGE, y = TOT_CUSTOMER, fill = PREMIUM_CUSTOMER))+
  geom_col(position = "dodge")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



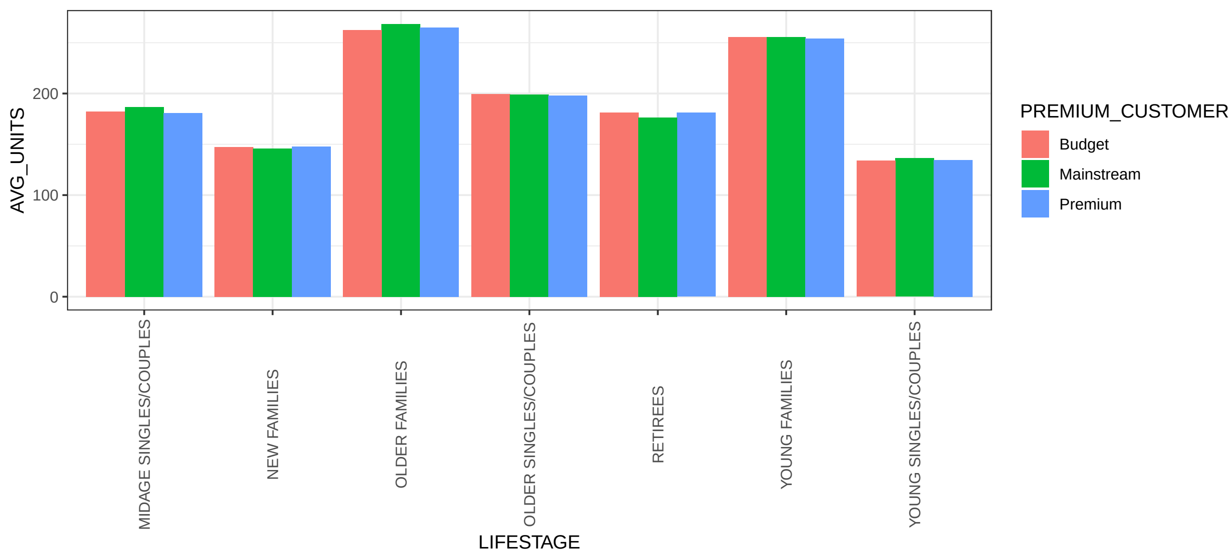
There are more Mainstream - young singles/couples and Mainstream - retirees who buy chips. This contributes to there being more sales to these customer segments but this is not a major driver for the Budget - Older families segment.

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
(units <- data[, .(AVG_UNITS=sum(PROD_NBR) / uniqueN(LYLTY_CARD_NBR)), .(LIFESTAGE, PREMIUM_CUSTOMER)][
```

```
##                  LIFESTAGE PREMIUM_CUSTOMER AVG_UNITS
##  1:          OLDER FAMILIES       Mainstream  268.5430
##  2:          OLDER FAMILIES          Premium  264.8861
##  3:          OLDER FAMILIES           Budget  262.3763
##  4:          YOUNG FAMILIES           Budget  255.7463
##  5:          YOUNG FAMILIES       Mainstream  255.7400
##  6:          YOUNG FAMILIES          Premium  254.3349
##  7:   OLDER SINGLES/COUPLES           Budget  199.5490
##  8:   OLDER SINGLES/COUPLES       Mainstream  198.8720
##  9:   OLDER SINGLES/COUPLES          Premium  197.9374
## 10: MIDAGE SINGLES/COUPLES       Mainstream  186.5919
## 11: MIDAGE SINGLES/COUPLES           Budget  182.1079
## 12:                RETIREES           Budget  181.3432
## 13:                RETIREES          Premium  181.0614
## 14: MIDAGE SINGLES/COUPLES          Premium  181.0038
## 15:                RETIREES       Mainstream  176.4344
## 16:            NEW FAMILIES          Premium  147.9722
## 17:            NEW FAMILIES           Budget  147.2015
## 18:            NEW FAMILIES       Mainstream  145.9663
## 19:   YOUNG SINGLES/COUPLES       Mainstream  136.3204
## 20:   YOUNG SINGLES/COUPLES          Premium  134.5512
## 21:   YOUNG SINGLES/COUPLES           Budget  133.7867
##                  LIFESTAGE PREMIUM_CUSTOMER AVG_UNITS
```

```
ggplot(units, aes(x = LIFESTAGE, y = AVG_UNITS, fill = PREMIUM_CUSTOMER))+
  geom_col(position = "dodge")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



Older families and young families in general buy more chips per customer.

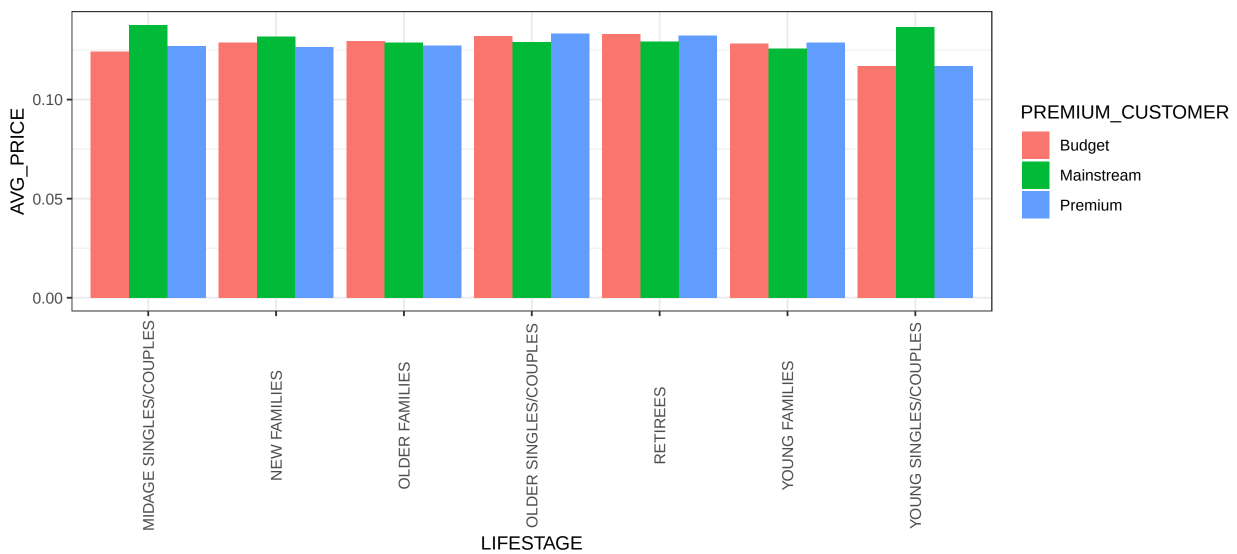
Let's also investigate the average price per unit chips bought for each customer segment as this is also a driver of total sales.

```
#### Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER
(price <- data[, .(AVG_PRICE=sum(TOT_SALES)/sum(PROD_NBR)), .(LIFESTAGE, PREMIUM_CUSTOMER)][order(-AVG_
```

```
##                  LIFESTAGE PREMIUM_CUSTOMER AVG_PRICE
##  1: MIDAGE SINGLES/COUPLES       Mainstream 0.1376942
##  2:  YOUNG SINGLES/COUPLES       Mainstream 0.1367453
##  3:  OLDER SINGLES/COUPLES          Premium 0.1333029
##  4:               RETIREES           Budget 0.1331962
##  5:               RETIREES          Premium 0.1322745
##  6:  OLDER SINGLES/COUPLES           Budget 0.1321123
##  7:           NEW FAMILIES       Mainstream 0.1318979
##  8:         OLDER FAMILIES           Budget 0.1296591
##  9:               RETIREES       Mainstream 0.1294106
## 10:  OLDER SINGLES/COUPLES       Mainstream 0.1290197
## 11:         YOUNG FAMILIES          Premium 0.1288282
## 12:           NEW FAMILIES           Budget 0.1287901
## 13:         OLDER FAMILIES       Mainstream 0.1287750
## 14:         YOUNG FAMILIES           Budget 0.1283110
## 15:         OLDER FAMILIES          Premium 0.1273224
## 16: MIDAGE SINGLES/COUPLES          Premium 0.1269685
## 17:           NEW FAMILIES          Premium 0.1264727
## 18:         YOUNG FAMILIES       Mainstream 0.1257362
## 19: MIDAGE SINGLES/COUPLES           Budget 0.1242263
## 20:  YOUNG SINGLES/COUPLES           Budget 0.1170727
```

```
## 21:  YOUNG SINGLES/COUPLES          Premium 0.1170327
##                 LIFESTAGE PREMIUM_CUSTOMER AVG_PRICE
```

```
ggplot(price, aes(x = LIFESTAGE, y = AVG_PRICE, fill = PREMIUM_CUSTOMER))+
  geom_col(position = "dodge")+
  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 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
mainstream <- data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & PREMIUM_CUSTOM
premium_and_budget <- data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & PREMIU
t.test(mainstream, premium_and_budget)
```

```
##
##  Welch Two Sample t-test
##
## data:  mainstream and premium_and_budget
## t = 13.902, df = 67402, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.09135741 0.12134651
## sample estimates:
## mean of x mean of y
##  4.039786  3.933434
```

The t-test results in a p-value of 2.2e-16, i.e. the unit price for mainstream, young and mid-age singles and couples ARE significantly higher than that of budget or premium, young and midage singles and couples.

Deep dive into specific customer segments for insights

We have found quite a few interesting insights that we can dive deeper into.

We might want to target customer segments that contribute the most to sales to retain them or further increase sales. Let's look at Mainstream - young singles/couples. For instance, let's find out if they tend to buy a particular brand of chips.

```
#### Deep dive into Mainstream, young singles/couples
# Over to you! Work out of there are brands that these two customer segments prefer more than others. Y
segment1 <- data[LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER == "Mainstream",]
segment2 <- data[!(LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER == "Mainstream"),]

Total1 <- segment1[, sum(PROD_QTY)]
Total2 <- segment2[, sum(PROD_QTY)]

Total1_BRAND <- segment1[, .(targetSegment = sum(PROD_QTY)/Total1), BRAND]
Total2_BRAND <- segment2[, .(Other = sum(PROD_QTY)/Total2), BRAND]

(merge(Total1_BRAND, Total2_BRAND, by='BRAND')[, Affinity := targetSegment/Other])[order(-Affinity)]
```

```
##          BRAND targetSegment       Other  Affinity
##  1:   TYRRELLS   0.031552795 0.025692464 1.2280953
##  2:   TWISTIES   0.046183575 0.037876520 1.2193194
##  3:    DORITOS   0.122760524 0.101074684 1.2145526
##  4:     KETTLE   0.197984817 0.165553442 1.1958967
##  5:   TOSTITOS   0.045410628 0.037977861 1.1957131
##  6:   PRINGLES   0.119420290 0.100634769 1.1866703
##  7:       COBS   0.044637681 0.039048861 1.1431238
##  8:  INFUZIONS   0.064679089 0.057064679 1.1334347
##  9:      THINS   0.060372671 0.056986370 1.0594230
## 10:    GRNWVES   0.032712215 0.031187957 1.0488733
## 11:   CHEEZELS   0.017971014 0.018646902 0.9637534
## 12:     SMITHS   0.096369910 0.124583692 0.7735355
## 13:     FRENCH   0.003947550 0.005758060 0.6855694
## 14:    CHEETOS   0.008033126 0.012066591 0.6657329
## 15:        RRD   0.043809524 0.067493678 0.6490908
## 16:    NATURAL   0.019599724 0.030853989 0.6352412
## 17:        CCS   0.011180124 0.018895650 0.5916771
## 18:   SUNBITES   0.006349206 0.012580210 0.5046980
## 19: WOOLWORTHS   0.024099379 0.049427188 0.4875733
## 20:     BURGER   0.002926156 0.006596434 0.4435967
```

We can see that young singles/couples trend to buy brands such as TYRRELLS, TWISTIES, and DORITOS than others.

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
Total1_BRAND <- segment1[, .(targetSegment = sum(PROD_QTY)/Total1), PACK_SIZE]
Total2_BRAND <- segment2[, .(Other = sum(PROD_QTY)/Total2), PACK_SIZE]

(merge(Total1_BRAND, Total2_BRAND, by='PACK_SIZE')[, Affinity := targetSegment/Other])[order(-Affinity)]
```

```
##     PACK_SIZE targetSegment       Other  Affinity
##  1:       270   0.031828847 0.025095929 1.2682873
##  2:       380   0.032160110 0.025584213 1.2570295
##  3:       330   0.061283644 0.050161917 1.2217166
##  4:       134   0.119420290 0.100634769 1.1866703
##  5:       110   0.106280193 0.089791190 1.1836372
##  6:       210   0.029123533 0.025121265 1.1593180
##  7:       135   0.014768806 0.013075403 1.1295106
##  8:       250   0.014354727 0.012780590 1.1231662
##  9:       170   0.080772947 0.080985964 0.9973697
## 10:       150   0.157598344 0.163420656 0.9643722
## 11:       175   0.254989648 0.270006956 0.9443818
## 12:       165   0.055652174 0.062267662 0.8937572
## 13:       190   0.007481021 0.012442016 0.6012708
## 14:       180   0.003588682 0.006066692 0.5915385
## 15:       160   0.006404417 0.012372920 0.5176157
## 16:        90   0.006349206 0.012580210 0.5046980
## 17:       125   0.003008972 0.006036750 0.4984423
## 18:       200   0.008971705 0.018656115 0.4808989
## 19:        70   0.003036577 0.006322350 0.4802924
## 20:       220   0.002926156 0.006596434 0.4435967
```

Our target group tends to buy package sizes like 270g, 380g and 330g, indicating that they prefer larger package sizes compared to the rest of the population.