# Task 1 - Retail Strategy and Analytics - Quantium Virtual Internship

Gifty Cheruvallimalayil

24/05/2022

---

---

## Setting the path and loading required libraries and data sets

```
setwd("C://Users//Gifty//OneDrive//Documents//quantium_virtual_internship//")

# install.packages("ggmosaic")

library(data.table)
library(ggplot2)
library(ggmosaic)
library(readr)
library(readxl)
library(tidyverse)

transactionData <- as.data.table(read_excel('QVI_transaction_data.xlsx'))
customerData <- as.data.table(read.csv("QVI_purchase_behaviour.csv"))
```

---

---

## Exploratory data analysis

The first step in any analysis is to first understand the data. Lets take a look at each of the data sets provided.

```
View(customerData)
View(transactionData)

names(transactionData)
```

```
## [1] "DATE"           "STORE_NBR"      "LYLTY_CARD_NBR" "TXN_ID"
## [5] "PROD_NBR"       "PROD_NAME"      "PROD_QTY"       "TOT_SALES"
```

```
names(customerData)
```

```
## [1] "LYLTY_CARD_NBR"   "LIFESTAGE"        "PREMIUM_CUSTOMER"
```

---

# Examining transaction data

---

```
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
```

```
str(transactionData)
```

```
## Classes 'data.table' and 'data.frame':   264836 obs. of  8 variables:
##  $ DATE          : num  43390 43599 43605 43329 43330 ...
##  $ STORE_NBR     : num  1 1 1 2 2 4 4 4 5 7 ...
##  $ LYLTY_CARD_NBR: num  1000 1307 1343 2373 2426 ...
##  $ TXN_ID        : num  1 348 383 974 1038 ...
##  $ PROD_NBR      : num  5 66 61 69 108 57 16 24 42 52 ...
##  $ PROD_NAME     : chr  "Natural Chip        Compny SeaSalt175g" "CCs Nacho Cheese    175g" "Smiths Crinkle Cut  Ch
ips Chicken 170g" "Smiths Chip Thinly  S/Cream&Onion 175g" ...
##  $ PROD_QTY      : num  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>
```

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

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

#### Examining PROD_NAME
# Generating a summary of the PROD_NAME column.
summary(transactionData$PROD_NAME)
```

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

```
transactionData[, .N, 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
```

```
#### Examining 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.

```
library(stringr)
library(stringi)

#### Removing digits
productWords$words <- str_replace_all(productWords$words,"[0-9]"," ")
productWords$words <- str_replace_all(productWords$words,"[gG]"," ")

#### Removing special characters
productWords$words <- str_replace_all(productWords$words,"[[:punct:]]"," ")
view(productWords)
```

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

```
cmm_words <- strsplit(productWords$words," ")
freq_words<-table(unlist(cmm_words))

freq_words <-  as.data.frame(freq_words)
freq_words <- freq_words[order(freq_words$Freq, decreasing = T),]
freq_words
```

```
##    Var1 Freq
## 1       1259
## 16    i  234
## 10    e  228
## 35    s  168
## 7     C  161
## 37    t  161
## 21    l  159
## 33    r  150
## 2     a  143
## 25    n  139
## 14    h  136
## 27    o  135
## 36    S  102
## 29    p   66
## 39    u   53
## 23    m   52
## 6     c   42
## 19    k   38
## 8     d   32
## 34    R   32
## 38    T   32
## 45    y   28
## 9     D   27
## 30    P   26
## 28    O   22
## 43    W   22
## 5     B   20
## 42    w   17
## 24    M   16
## 20    K   13
## 4     b   11
## 46    z   11
## 12    f   10
## 41    V    9
## 13    F    8
## 15    H    8
## 26    N    8
## 40    v    8
## 22    L    7
## 17    I    5
## 44    x    4
## 3     A    3
## 11    E    3
## 18    J    3
## 32    Q    3
## 31    q    1
```

We can see that we have 732 whitespaces and the word chips is appearing 21 times
There are salsa products in the dataset but we are only interested in the chips category, so let’s remove these.

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

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

```
#### Summarize 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.

```
library(tidyverse)
library(dplyr)

prod_qty_200 <- transactionData %>% filter(PROD_QTY==200)
count(prod_qty_200)
```

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
same_customer <- transactionData %>% filter(LYLTY_CARD_NBR == 226000)
same_customer
```

```
##          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.

```
#### Filtering out the customer based on the loyalty card number
transaction_data <- transactionData[!(transactionData$LYLTY_CARD_NBR == 226000)]

#### Re-examine transaction data
summary(transaction_data)
```

```
##       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
```

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
countByDate <- count(transaction_data, transaction_data$DATE)
countByDate
```

```
##      transaction_data$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
```

```
nrow(countByDate)
```

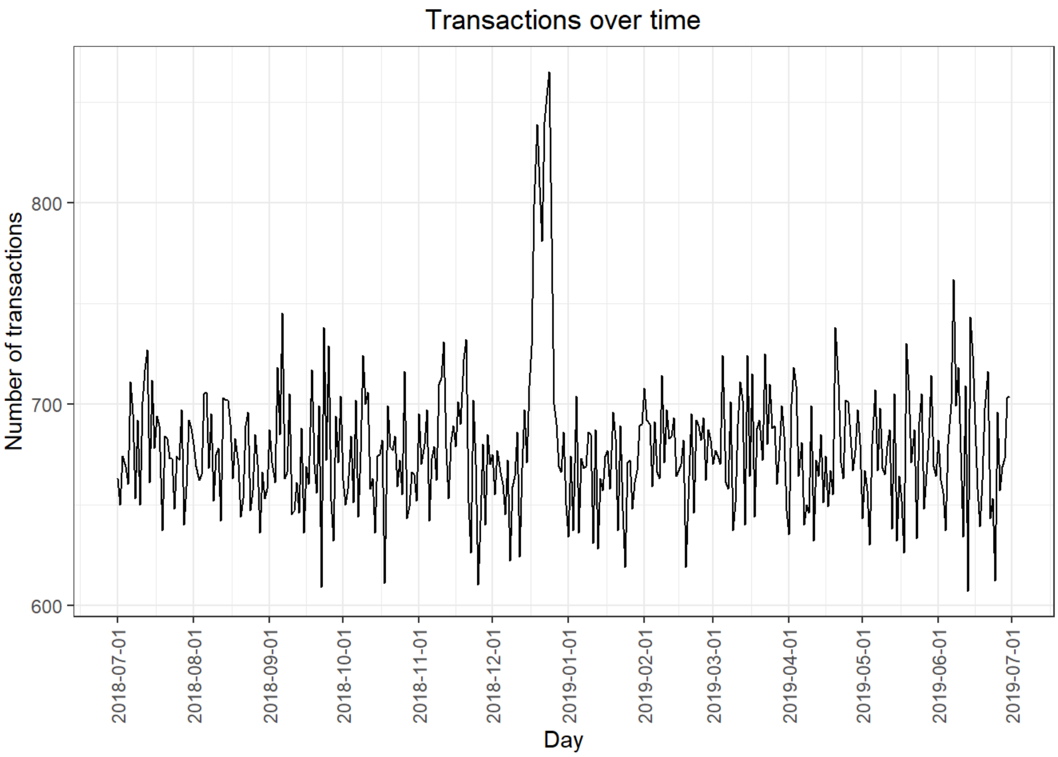
```
## [1] 364
```

```
summary(countByDate)
```

```
##  transaction_data$DATE       n
##  Min.   :2018-07-01    Min.   :607.0
##  1st Qu.:2018-09-29    1st Qu.:658.0
##  Median :2018-12-30    Median :674.0
##  Mean   :2018-12-30    Mean   :677.9
##  3rd Qu.:2019-03-31    3rd Qu.:694.2
##  Max.   :2019-06-30    Max.   :865.0
```

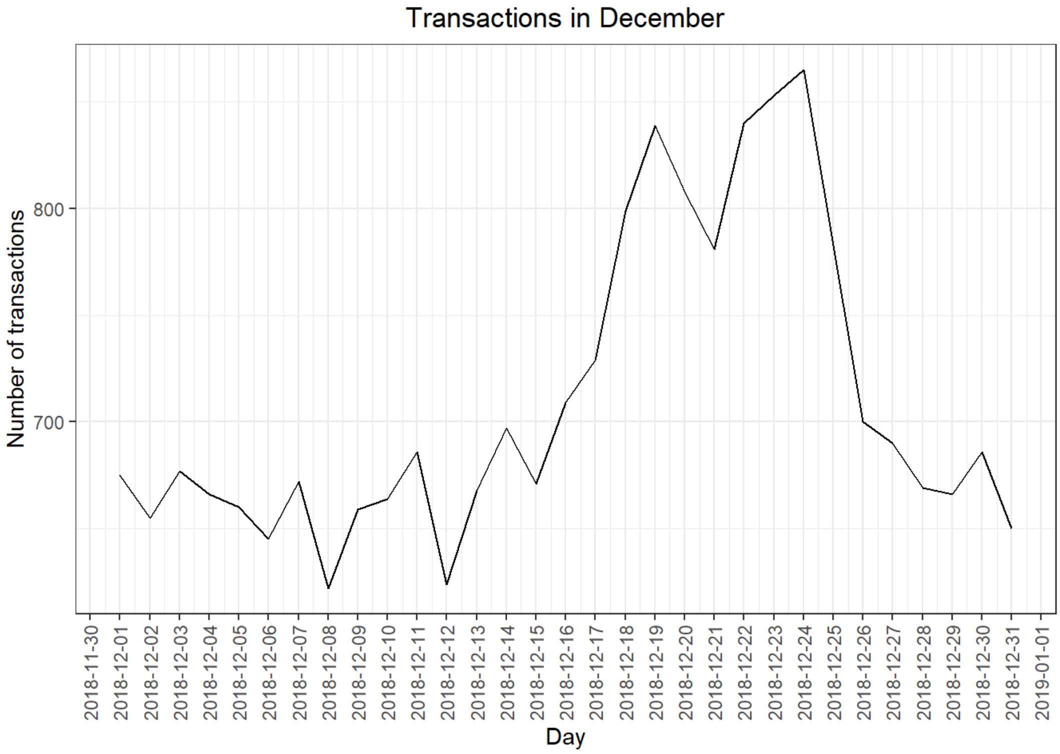
There's only 364 rows, meaning only 364 dates which indicates a missing date. Let's create a sequence of dates from 1 Jul 2018 to 30 Jun 2019 and use this to create a chart of number of transactions over time to find the missing date.

```
#### Create a sequence of dates and join this the count of transactions by date
seq_dates <- transaction_data[order(DATE),]

#### create a column of dates that includes every day from 1 Jul 2018 to
# 30 Jun 2019, and join it onto the data to fill in the missing day.

#### 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(countByDate,
                aes(x = countByDate$`transaction_data$DATE`, y = countByDate$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))
trans_over_time
```




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

```
#### Filter to December and look at individual days
filterData <- countByDate[countByDate$`transaction_data$DATE` >= "2018-12-01" & countByDate$`transaction_data$DATE` <=
"2018-12-31"]
ggplot(filterData, aes(x = filterData$`transaction_data$DATE`, y = filterData$n)) +
  geom_line() +
  labs(x = "Day", y = "Number of transactions", title = "Transactions in December") +
  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.

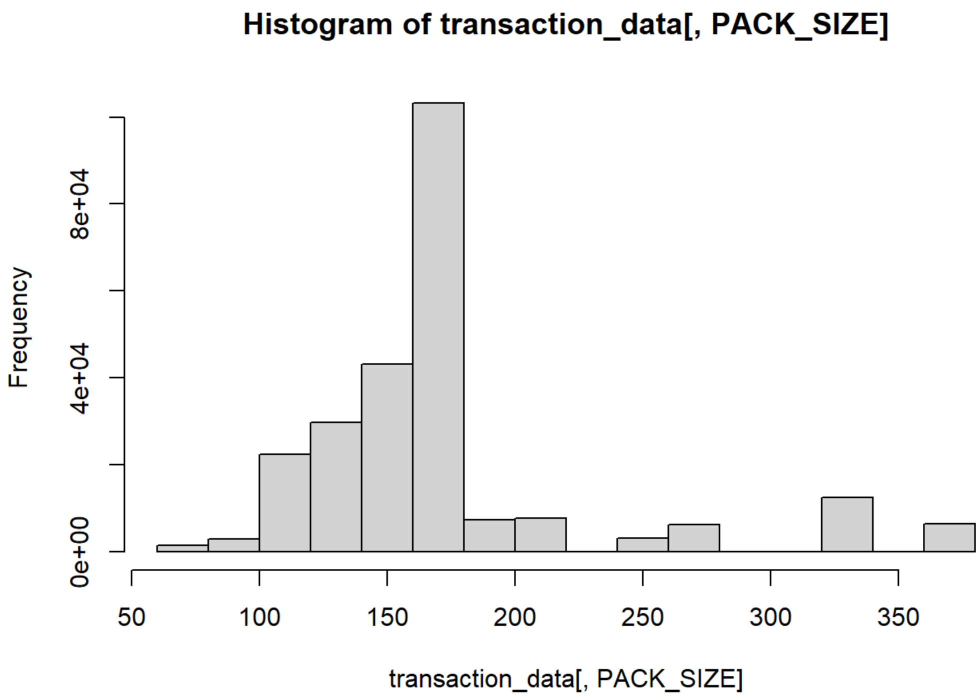
Now that we are satisfied that the data no longer has outliers, we can move on to creating other features such as brand of chips or pack size from PROD_NAME. We will start with pack size.

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

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

```
##     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 continuous variable even though it is numeric.
hist(transaction_data[, PACK_SIZE])
```




Pack sizes created look reasonable.

Now to create brands, we can use the first word in PROD_NAME to work out the brand name…

```
#### Brands
transaction_data$BRAND <- gsub("([A-Za-z]+).*", "\\1", transaction_data$PROD_NAME)

#### Checking brands
transaction_data[, .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
transaction_data[BRAND == "RED", BRAND := "RRD"]
transaction_data[BRAND == "SNBTS", BRAND := "SUNBITES"]
transaction_data[BRAND == "INFZNS", BRAND := "INFUZIONS"]
transaction_data[BRAND == "WW", BRAND := "WOOLWORTHS"]
transaction_data[BRAND == "SMITH", BRAND := "SMITHS"]
transaction_data[BRAND == "NCC", BRAND := "NATURAL"]
transaction_data[BRAND == "DORITO", BRAND := "DORITOS"]
transaction_data[BRAND == "GRAIN", BRAND := "GRNWVES"]

#### Check again
transaction_data[, .N, by = BRAND][order(BRAND)]
```

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

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
```

```
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" "OLDER 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
```

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

As the number of rows in `data` is the same as that of `transaction_data`, 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_data` and find rows with matching values in shared columns and then joining the details in these rows to the `x` or the first mentioned table.

```
#### Let's also check if some customers were not matched on by checking for nulls.
sum(is.na(data))
```

```
## [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,"QVI_data.csv")
```

---

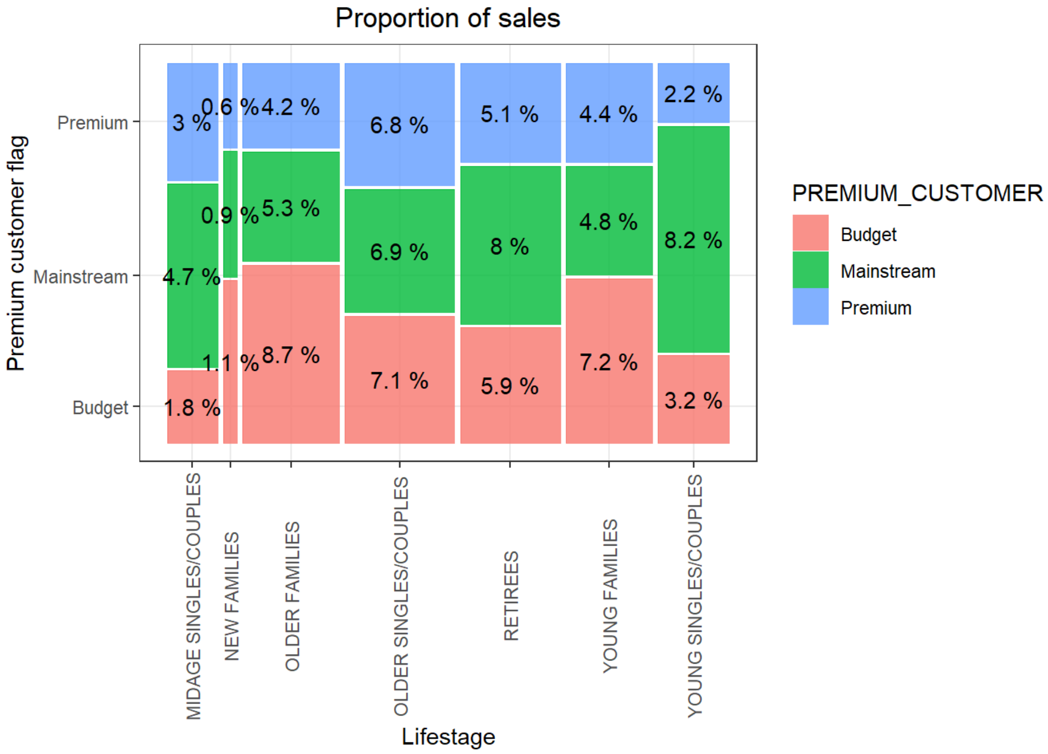
# Data analysis on customer segments

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

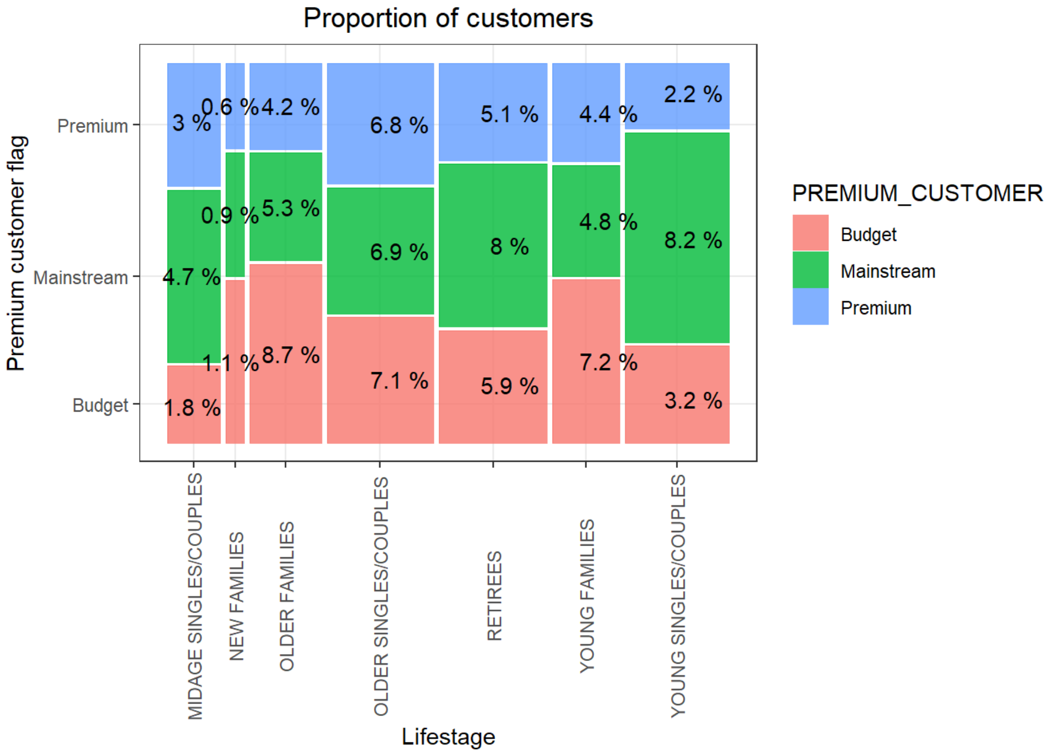
```
#### Total sales by LIFESTAGE and PREMIUM_CUSTOMER
total_sales <- data %>% group_by(LIFESTAGE,PREMIUM_CUSTOMER)
pf.total_sales <- summarise(total_sales,sales_count=sum(TOT_SALES))
summary(pf.total_sales)

p <- ggplot(pf.total_sales) + geom_mosaic(aes(weight = sales_count, x = product(PREMIUM_CUSTOMER, LIFESTAGE),fill = PR
EMIUM_CUSTOMER)) + labs(x = "Lifestage", y = "Premium customer flag", title = "Proportion of sales") + theme(axis.tex
t.x = element_text(angle = 90, vjust = 0.5))
p +geom_text(data = ggplot_build(p)$data[[1]], aes(x = (xmin + xmax)/2 , y = (ymin + ymax)/2, label = as.character(pas
te(round(.wt/sum(.wt),3)*100, '%'))), inherit.aes = F)
```




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.

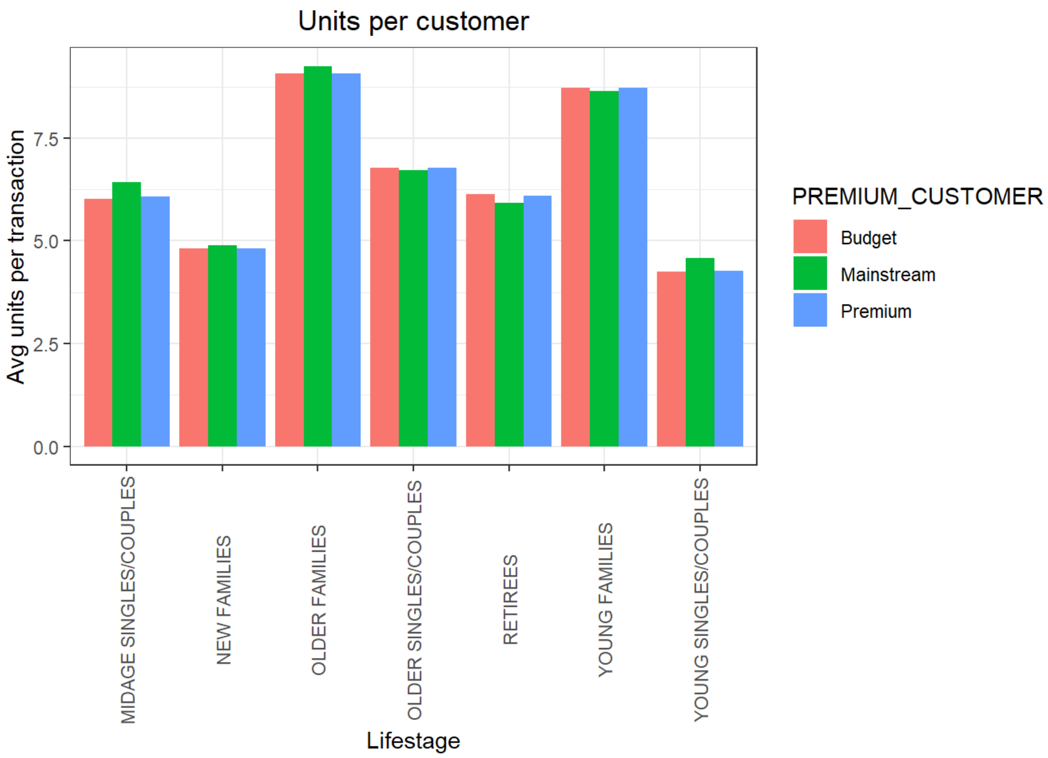
```
#### Number of customers by LIFESTAGE and PREMIUM_CUSTOMER
total_sales <- data %>% group_by(LIFESTAGE,PREMIUM_CUSTOMER)
no_of_customers <- summarise(total_sales,customer_count = length(unique(LYLTY_CARD_NBR)))

summary(no_of_customers)

p <- ggplot(data = no_of_customers) +
  geom_mosaic(aes(weight = customer_count, x = product(PREMIUM_CUSTOMER, LIFESTAGE), fill = PREMIUM_CUSTOMER)) +
  labs(x = "Lifestage", y = "Premium customer flag", title = "Proportion of customers") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))+
  geom_text(data = ggplot_build(p)$data[[1]], aes(x = (xmin + xmax)/2 , y = (ymin + ymax)/2, label = as.character(past
e(round(.wt/sum(.wt),3)*100, '%'))))

p
```




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.

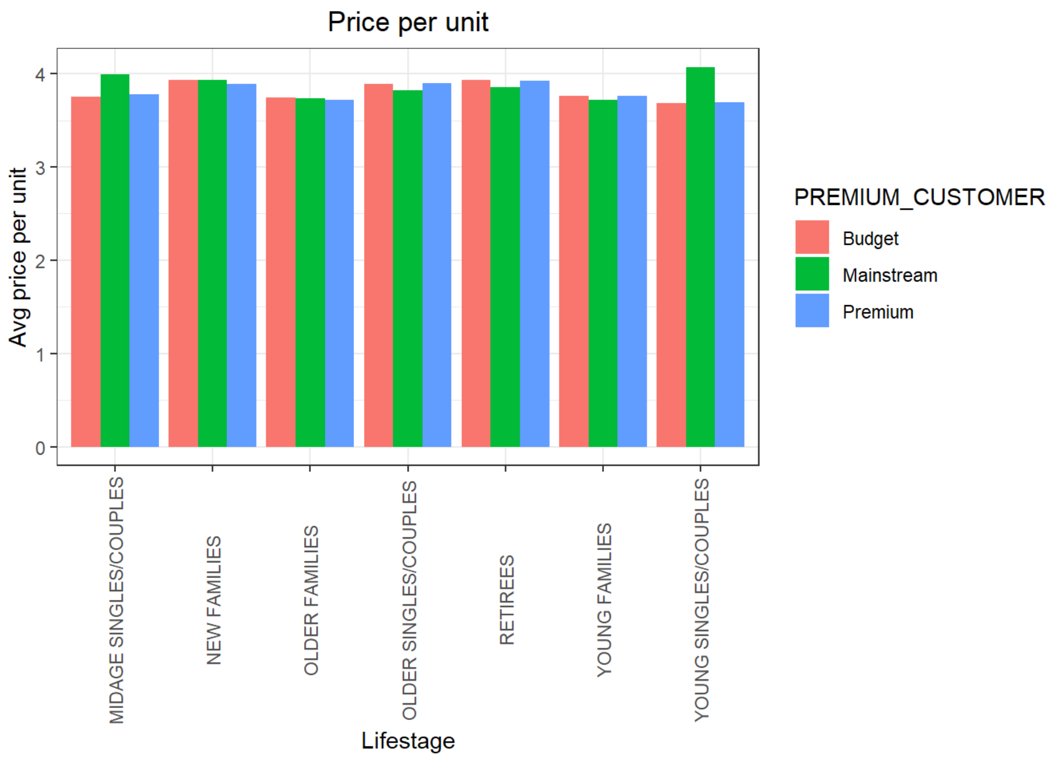
```
#### Average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER
total_sales_1 <-data %>% group_by(LIFESTAGE,PREMIUM_CUSTOMER)
units <-  summarise(total_sales_1, units_count = (sum(PROD_QTY)/uniqueN(LYLTY_CARD_NBR)))

summary(units)

ggplot(data = units, aes(weight = units_count, 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
total_sales_2 <-data %>% group_by(LIFESTAGE,PREMIUM_CUSTOMER)
pricePerUnit <-  summarise(total_sales_2, price_per_unit = (sum(TOT_SALES)/sum(PROD_QTY)))

ggplot(data=pricePerUnit, aes(weight = price_per_unit,x = LIFESTAGE, fill = PREMIUM_CUSTOMER)) +
  geom_bar(position = position_dodge()) +
  labs(x = "Lifestage", y = "Avg price per unit", 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 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
pricePerUnit <- data[, price := TOT_SALES/PROD_QTY]
t.test(data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & PREMIUM_CUSTOMER == "Mainstream", pr
ice],data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & PREMIUM_CUSTOMER != "Mainstream", pric
e], alternative = "greater")
```

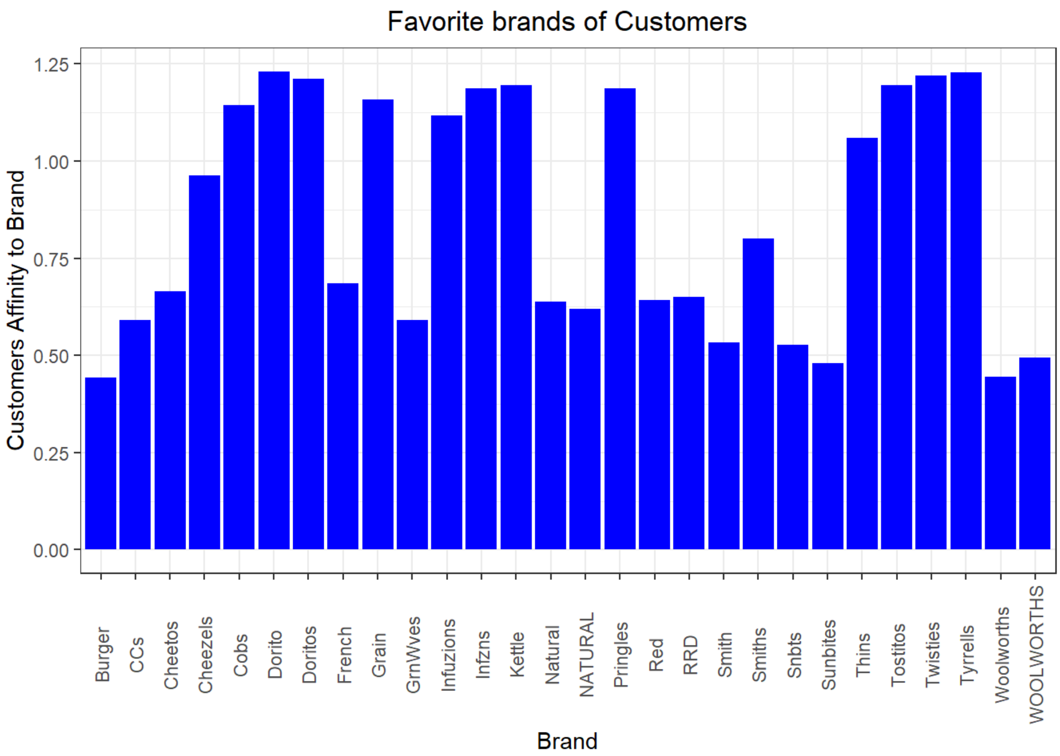
The t-test results in a p-value < 2.2e-16, i.e. the unit price for mainstream, young and mid-age singles and couples ARE significantly higher than that of budget or premium, young and midage singles and couples.

# 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
segment1 <- data[LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER == "Mainstream",]
other <- data[!(LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER =="Mainstream"),]

#### Brand affinity compared to the rest of the population
quantity_segment1 <- segment1[, sum(PROD_QTY)]

quantity_other <- other[, sum(PROD_QTY)]

quantity_segment1_by_brand <- segment1[, .(targetSegment = sum(PROD_QTY)/quantity_segment1), by = BRAND]

quantity_other_by_brand <- other[, .(other = sum(PROD_QTY)/quantity_other), by = BRAND]

brand_proportions <- merge(quantity_segment1_by_brand, quantity_other_by_brand)[, affinityToBrand := targetSegment/oth
er]

brand_proportions[order(-affinityToBrand)]

ggplot(brand_proportions, aes(brand_proportions$BRAND,brand_proportions$affinityToBrand)) +
  geom_bar(stat = "identity",fill = "blue") +
  labs(x = "Brand", y = "Customers Affinity to Brand", title = "Favorite brands of Customers") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```

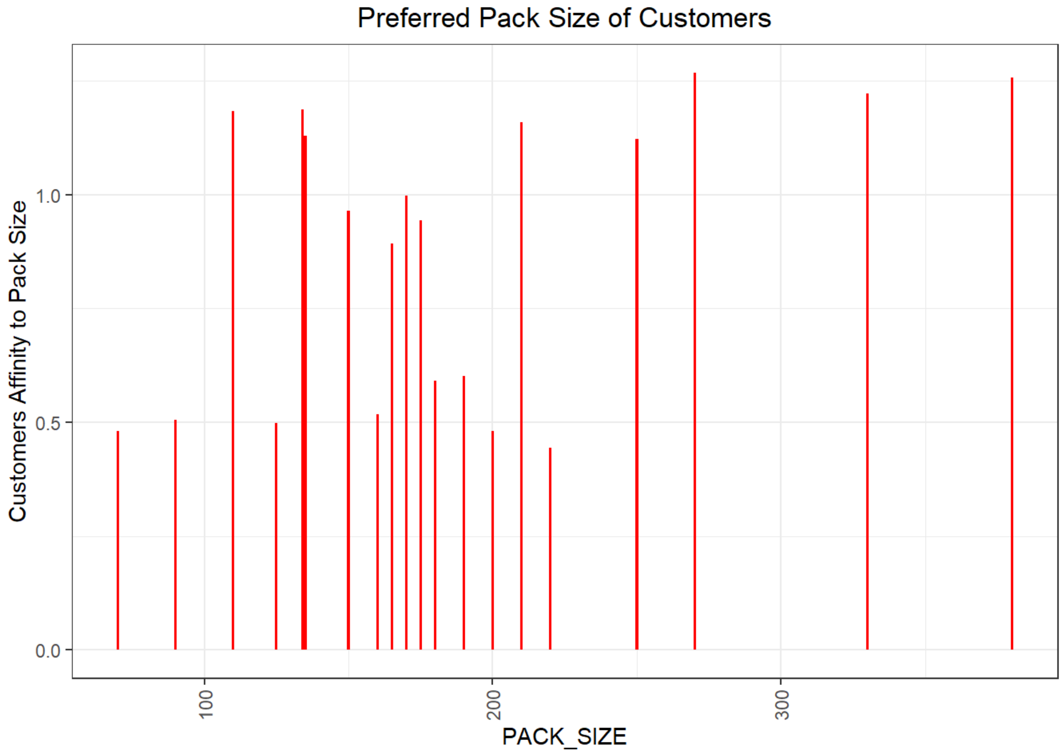



[INSIGHTS] Here, We can see that:

- Mainstream young singles/couples are 56% less likely to purchase Burger Rings compared to the rest of the population
- Mainstream young singles/couples are 23% more likely to purchase Tyrrells chips 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_segment1_by_pack <- segment1[, .(targetSegment = sum(PROD_QTY)/quantity_segment1), by = PACK_SIZE]
quantity_other_by_pack <- other[, .(other = sum(PROD_QTY)/quantity_other), by = PACK_SIZE]

pack_proportions <- merge(quantity_segment1_by_pack, quantity_other_by_pack)[, affinityToPack := targetSegment/other]
pack_proportions[order(-affinityToPack)]

ggplot(pack_proportions, aes(pack_proportions$PACK_SIZE,pack_proportions$affinityToPack)) +
  geom_bar(stat = "identity",fill = "red") +
  labs(x = "PACK_SIZE", y = "Customers Affinity to Pack Size", title = "Preferred Pack Size of Customers") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```




[INSIGHTS] Here, We can see that the preferred `PACK_SIZE` is 270g: