# Quantium Virtual Intership Task 1

Lizbeth Santiago

## Background information

The Category Manager for Chips wants understand the types of customers who purchase Chips and their purchasing behaviour within the region. The insights from the analysis will feed into the supermarket's strategic plan for the chip category in the next half year.

## Exploratory data analysis

### Load required libraries

```
library(data.table)
library(ggplot2)
library(readr)
library(ggmosaic)
```

### Reading data

```
filePath <- "C:/Users/Liz/Documents/Virtual_Intership/"
transactionData <- fread(paste0(filePath,"QVI_transaction_data.csv"))
customerData <- fread(paste0(filePath,"QVI_purchase_behaviour.csv"))
```

### 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
```

```r
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 
##  $ 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>
```

The column DATE is in an integer format, so it is necessary to change this to a date format

#### Convert column DATE to date format

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

#### Verify information - examine PROD_NAME

To check if the products are those that the analysis need. Choose the unique product names, and examine the words in PROD_NAME to see if there are any incorrect entries such as products that are not chips

```r
#unique(transactionData[, PROD_NAME])

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

#### Removing digits and special character

To focus just in words that will tell us if the product is chips or not, remove all words with digits and special characters.

```r
patterns <- c('^[123456789]','&')
words2 <- productWords [! grepl (paste(patterns, collapse='|'), productWords$words),]
```

#### Most common words and sorting them

```r
commonWords <- words2[, .(.N), .(words)][order(-N)]
```

There are some salsa products in the dataset that we don't need.

#### Remove Salsa products

```r
transactionData[, SALSA := grepl("salsa", tolower(PROD_NAME))]# new column that marks a salsa product
transactionData <- transactionData[SALSA == FALSE, ][, SALSA := NULL]
```

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

```r
summary(transactionData) #there is a case where 200 packets of chips are bought in one transaction.
```

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

There is a case where 200 packets of chips are bought in one transaction.

```r
transactionData[PROD_QTY>10]
```

```
##          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
```

```r
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
```

Actually we found that there are two transactions by the same customer, with card number 226000, who just made these two transactions of 200 packets.

### Removing these two transactions

```r
transactionData <- transactionData[LYLTY_CARD_NBR!=226000]
```

**Re-examine transaction data**

```r
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
```

**Count the number of transactions by date**

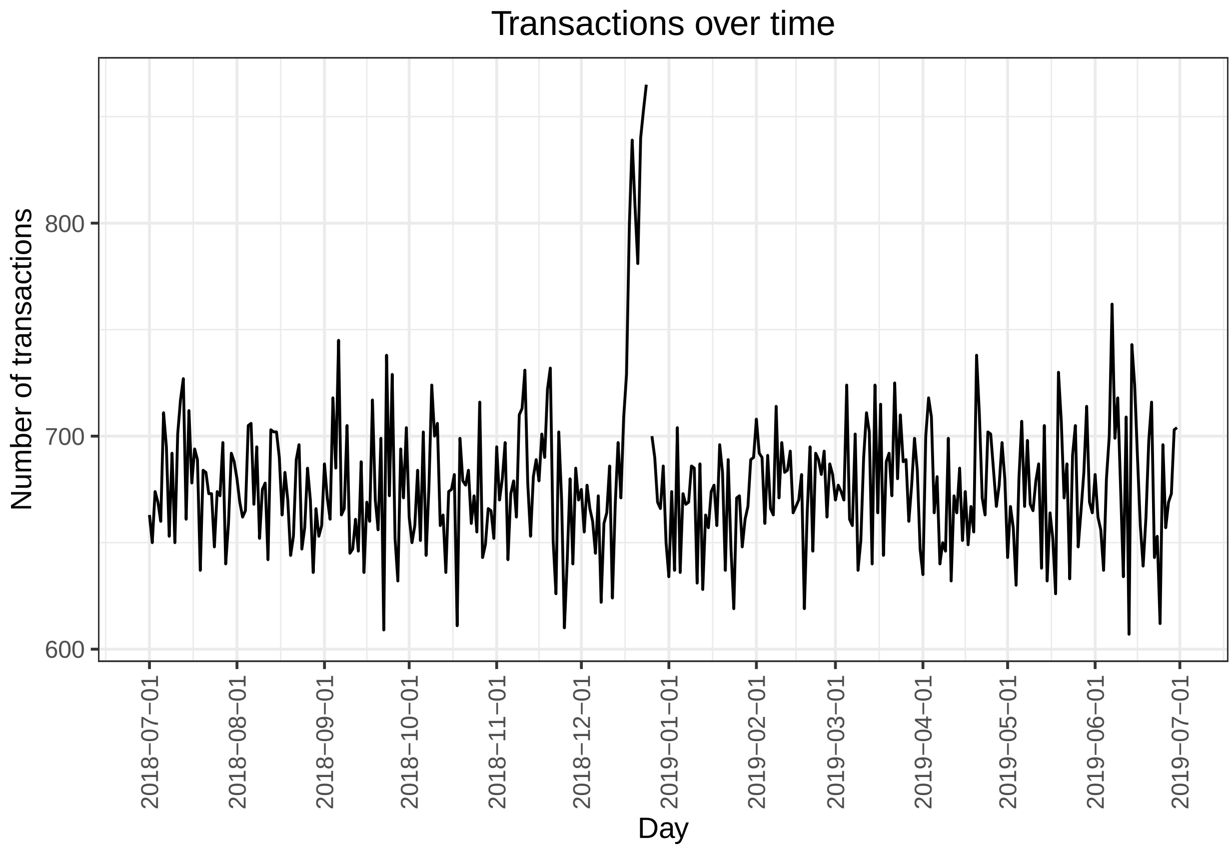
```r
countByDate<- transactionData[, .(.N), .(DATE)]#a missing date
```

There is a missing date, so create a sequence of dates and use this to create a chart of number of transactions over time to find the missing date.

```r
s <- as.Date('2018-07-01')
e <- as.Date('2019-06-30')
completeDates <- data.table(seq(from=s, to=e, by=1))

countByDate2 <- merge(countByDate, completeDates,by.x="DATE", by.y="V1", all=T)
```

**Plot transactions over time**

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

ggplot(countByDate2, 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.

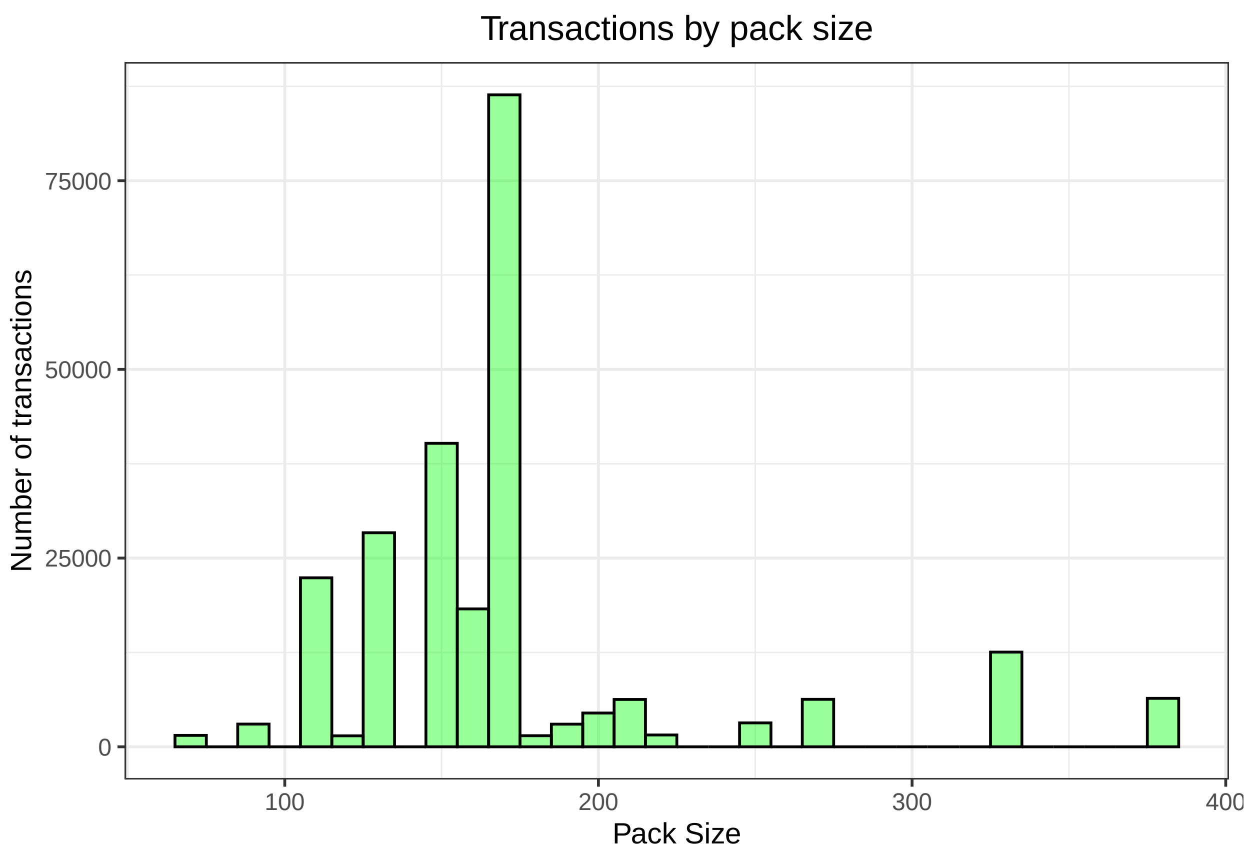
#### Ploting transaction of December and January

```
dec_jan = countByDate2[DATE>as.Date('2018-12-07') & DATE<as.Date('2019-01-07')]

ggplot(dec_jan, 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))
```


The missing date is on December 25th, that day shops are closed. 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, then decrease to the usual sales.

### Checking information about Pack size

```
transactionData[, PACK_SIZE := parse_number(PROD_NAME)]
pack_sizes <- transactionData[, .N, PACK_SIZE][order(PACK_SIZE)]

ggplot(transactionData, aes(x = PACK_SIZE)) +
  geom_histogram(binwidth = 10, col='black', fill='green', alpha=0.4) +
  labs(x = "Pack Size", y = "Number of transactions", title = "Transactions by pack size")
```




Most of the sales are from 175g pack size

## Checking Brands

To obtain the brands we extract just the first word of product names

```
library(stringr)
transactionData[, BRAND := str_split(transactionData$PROD_NAME, " ", simplify = TRUE)[,1] ] #First word
unique(transactionData[, BRAND])
```

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

Some of the brands have wrong spelling

### Brand adjustments

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

#### Re-examine of brands

```
unique(transactionData[, BRAND])
```

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

### 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
```

### Merge both datasets

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

**Checking for nulls.**

```
sum(is.na(data))
```

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

**Save 'data'**

```
filePath <- "C:/Users/Liz/Documents/Virtual_Intership/"
fwrite(data, paste0(filePath,"QVI_data.csv"))
```

# Data analysis on customer segments

We want to find:

- 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

```
library(tidyverse)
Data <- read.csv('QVI_data.csv')
```

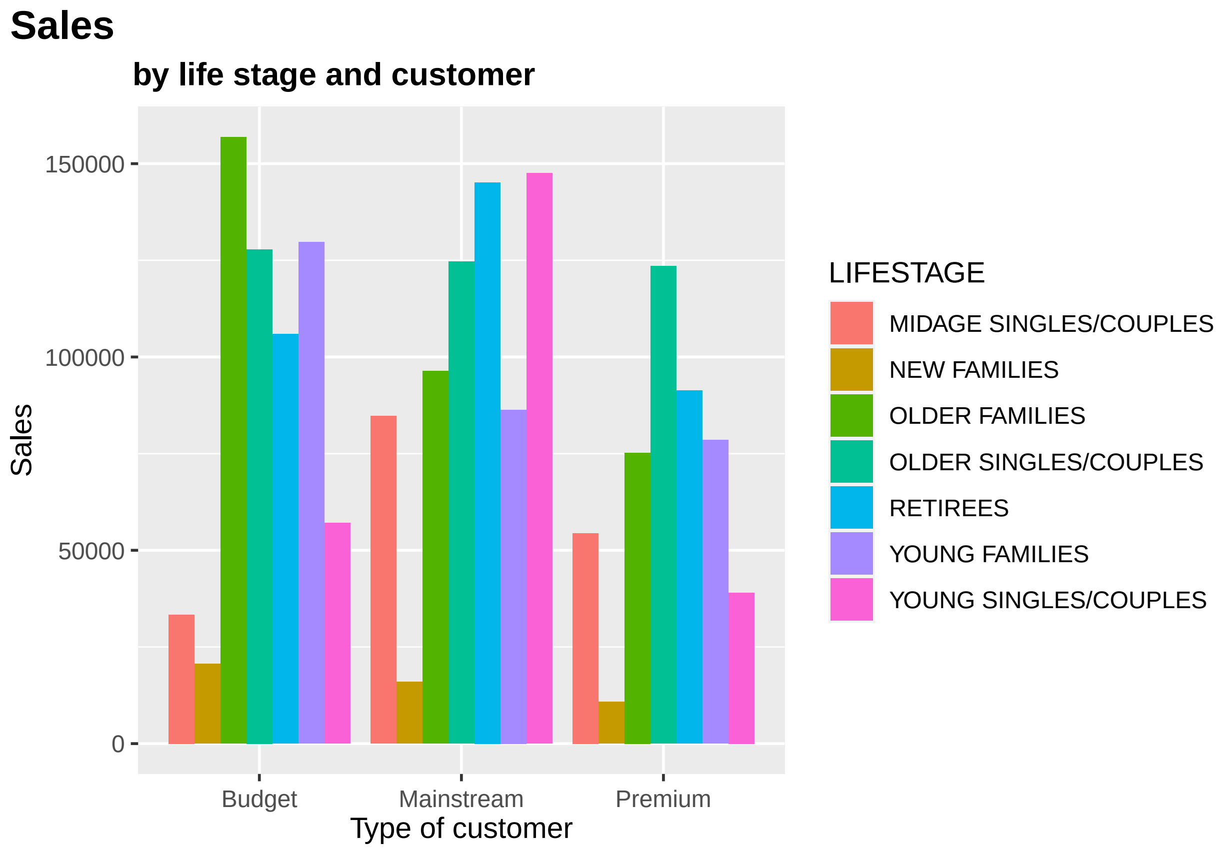
## TOTAL SALES by lifestage and premium_customer

Table with summarized information

```
lifestage_customer <- Data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(sales = sum(TOT_SALES)) %>%
  arrange(LIFESTAGE, PREMIUM_CUSTOMER)
```

Create a bar graph with the last table

```
ggplot(data = lifestage_customer,
       aes(x = PREMIUM_CUSTOMER, y = sales, fill = LIFESTAGE)) +
  geom_col(position = 'dodge') +
  labs(x ='Type of customer', y ='Sales',
       title ='Sales', subtitle ='by life stage and customer') +
  theme(axis.text.x = element_text(vjust = 0.5),
        plot.title.position = 'plot', plot.title = element_text(size = 15, fac = 'bold'),
  plot.subtitle = element_text(size = 12,fac = 'bold',hjust = 0.15))
```

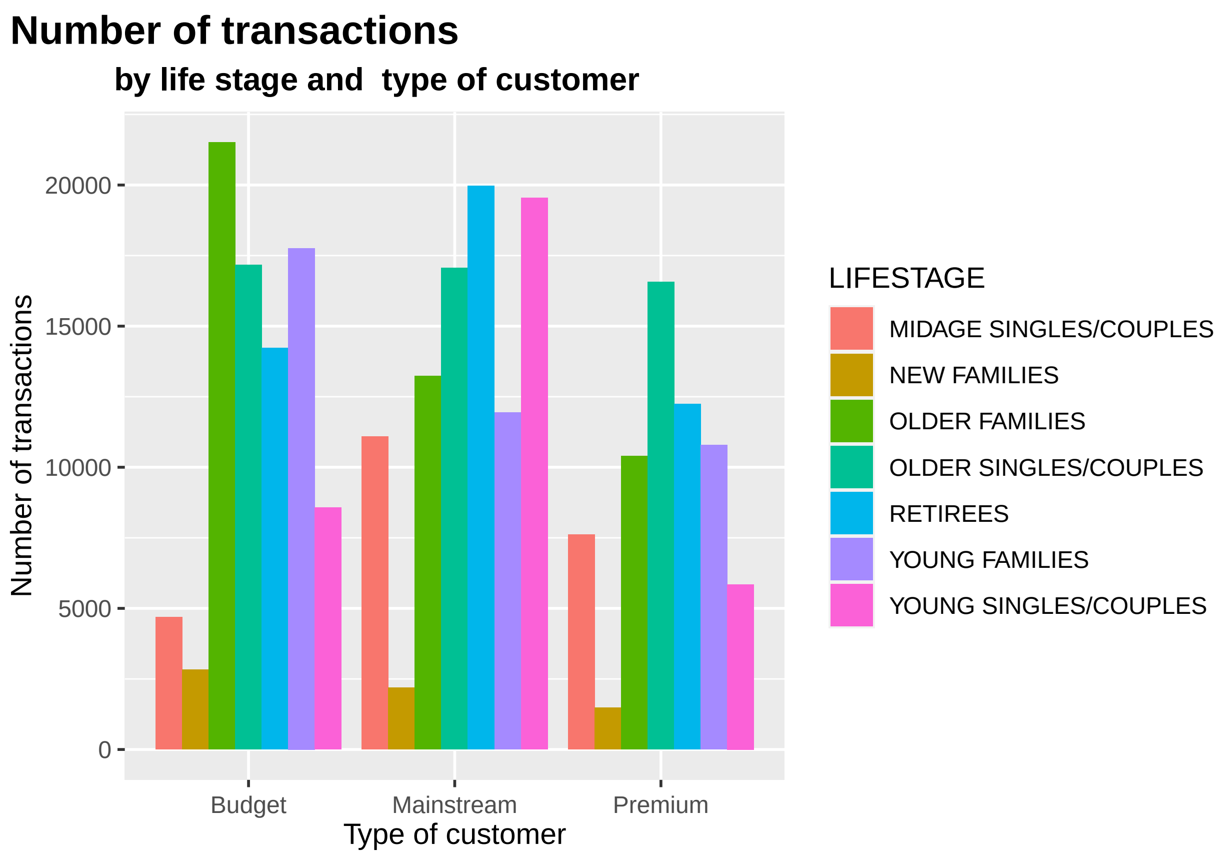



Sales are coming mainly from Budget - older families, Mainstream - young singles/couples, and Mainstream - retirees

Let's see the quantity of transaction by life stage and type of customer

### NUMBER OF TRANSACTIONS by lifestage and premium_customer

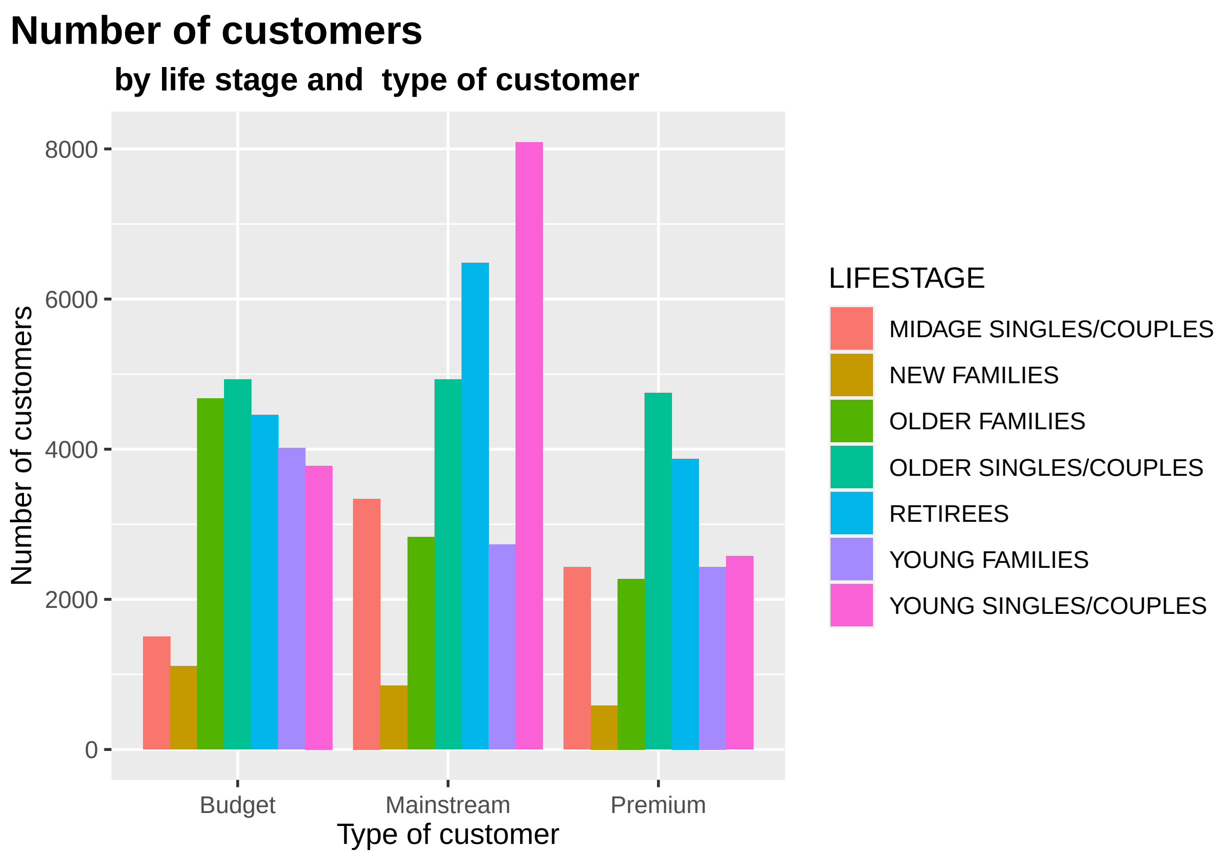
```
n_transactions <- Data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(nbr_transaction = n())

ggplot(data = n_transactions,
       aes(x= PREMIUM_CUSTOMER, y = nbr_transaction, fill = LIFESTAGE)) +
  geom_col(positio = 'dodge') +
  labs(x ='Type of customer', y ='Number of transactions',
       title ='Number of transactions', subtitle ='by life stage and  type of customer') +
  theme(axis.text.x = element_text(vjust = 0.5),
        plot.title.position = 'plot', plot.title = element_text(size = 15, fac = 'bold'),
        plot.subtitle = element_text(size = 12,fac = 'bold',hjust = 0.15))
```




Most of the transactions come from Budget Older families and Mainstream Young Single/Couple and Retirees. Similar to total sales.

### NUMBER OF CUSTOMERS by lifestage and premium_customer

```
data_customer <- read.csv('QVI_purchase_behaviour.csv')

n_customer <- data_customer %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(nbr_customers = n()) %>%
  arrange(LIFESTAGE, PREMIUM_CUSTOMER)

ggplot(data = n_customer,
       aes(x= PREMIUM_CUSTOMER, y = nbr_customers, fill = LIFESTAGE)) +
  geom_col(positio = 'dodge') +
  labs(x ='Type of customer', y ='Number of customers',
       title ='Number of customers', subtitle ='by life stage and  type of customer') +
  theme(axis.text.x = element_text(vjust = 0.5),
        plot.title.position = 'plot', plot.title = element_text(size = 15, fac = 'bold'),
        plot.subtitle = element_text(size = 12,fac = 'bold',hjust = 0.15))
```




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.

### AVERAGE NUMBER OF UNITS PER CUSTOMER by lifestage and premium_customer

To obtain the average by customer, and not by transaction, we group by card number

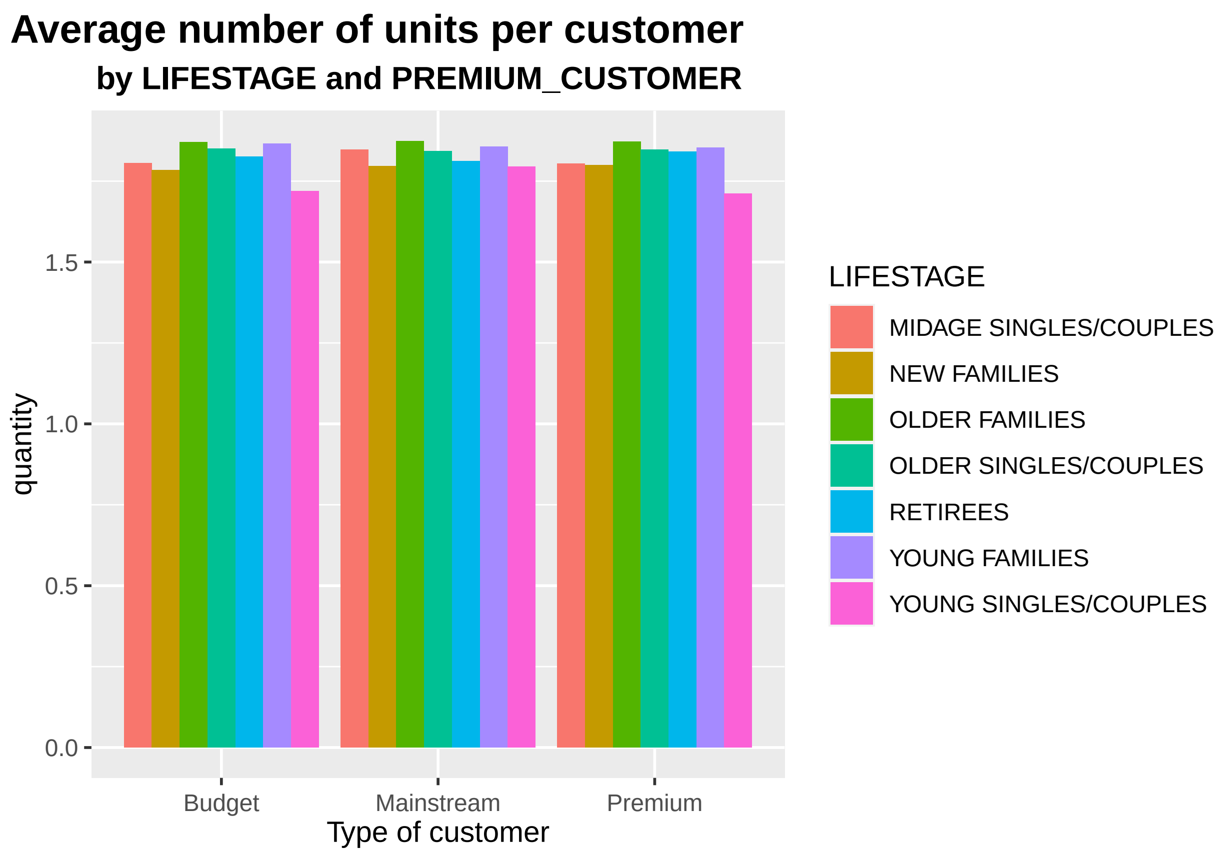
```
n_units <- Data %>%
  group_by(LYLTY_CARD_NBR, LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(avg = mean(PROD_QTY)) %>%
  arrange(LIFESTAGE, PREMIUM_CUSTOMER)
```

Then we calculate the average again, to obtain the average number of units per customer

```
n_unitsT <- n_units %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(avg2 = mean(avg))
```

```
ggplot(data = n_unitsT,
       aes(x= PREMIUM_CUSTOMER, y = avg2, fill = LIFESTAGE)) +
  geom_col(positio = 'dodge') +
  labs(x ='Type of customer', y ='quantity',
```

```
      title ='Average number of units per customer', subtitle ='by LIFESTAGE and PREMIUM_CUSTOMER') +
  theme(axis.text.x = element_text(vjust = 0.5),
        plot.title.position = 'plot', plot.title = element_text(size = 15, fac = 'bold'),
        plot.subtitle = element_text(size = 12,fac = 'bold',hjust = 0.15))
```



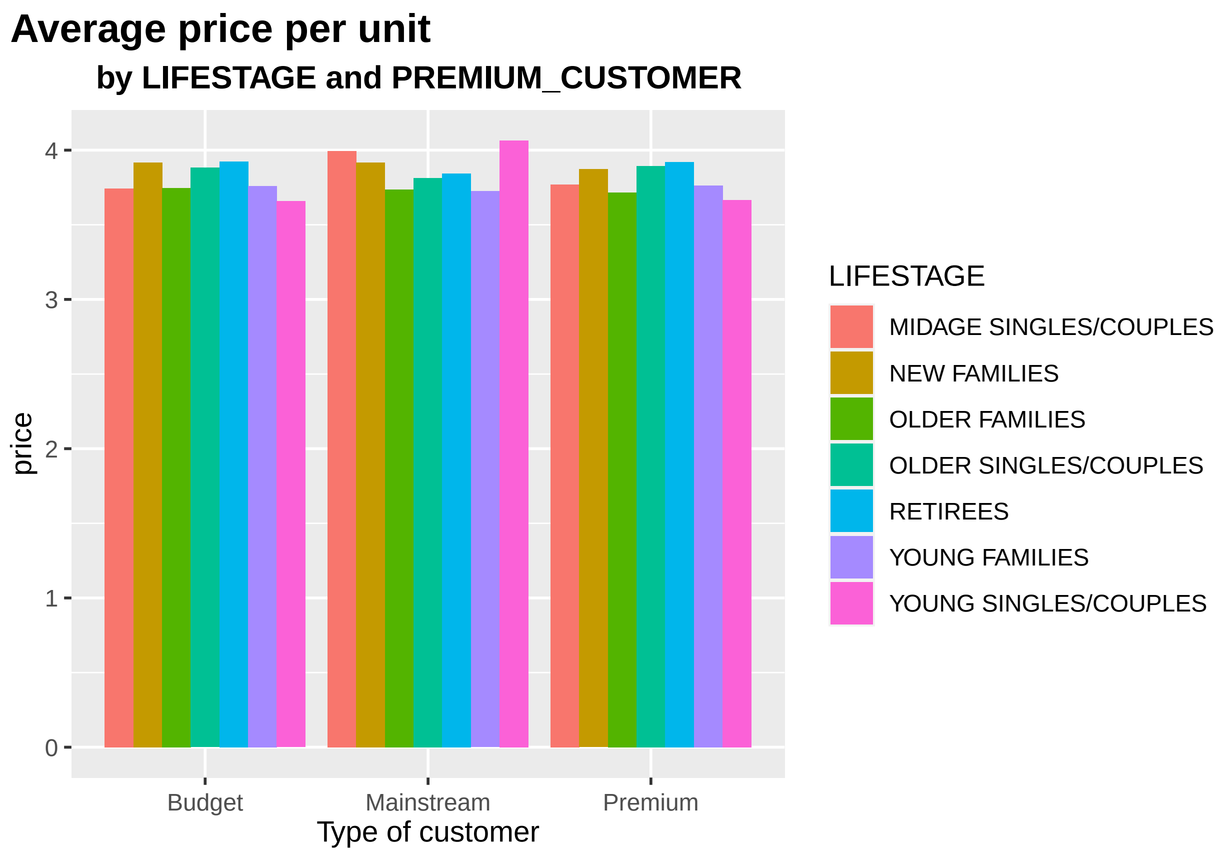

Older families and young families in general buy more chips per customer

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

```
Data$UNIT_PRICE <- Data$TOT_SALES/Data$PROD_QTY

price_unit <- Data %>%
  group_by(LIFESTAGE,PREMIUM_CUSTOMER) %>%
  summarise(avg_price = mean(UNIT_PRICE))

ggplot(data = price_unit,
       aes(x= PREMIUM_CUSTOMER, y = avg_price, fill = LIFESTAGE)) +
  geom_col(positio = 'dodge') +
  labs(x ='Type of customer', y ='price',
       title ='Average price per unit', subtitle ='by LIFESTAGE and PREMIUM_CUSTOMER') +
  theme(axis.text.x = element_text(vjust = 0.5),
```

```
      plot.title.position = 'plot', plot.title = element_text(size = 15, fac = 'bold'),
      plot.subtitle = element_text(size = 12,fac = 'bold',hjust = 0.15))
```




Mainstream midage and young singles and couples are more willing to pay more per packet of chips compared to their budget and premium counterparts. To check if the difference is statistically different we use a t-test.

## Perform an independent t-test between mainstream vs premium and budget, midage and young singles and couples

### Young Singles and Couples

#### Mainstream vs Premium

Select the data from young singles and couples from maintream and premium customers.

```
young_mainstream_premium <- Data[Data$LIFESTAGE == 'YOUNG SINGLES/COUPLES' &
                                    (Data$PREMIUM_CUSTOMER == 'Mainstream' |
                                       Data$PREMIUM_CUSTOMER == 'Premium'),11:13]
```

t-test

```r
t.test(UNIT_PRICE ~ PREMIUM_CUSTOMER, data = young_mainstream_premium)
```

```
##
##  Welch Two Sample t-test
##
## data:  UNIT_PRICE by PREMIUM_CUSTOMER
## t = 24.777, df = 8897.4, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group Mainstream and group Premium is not e
## 95 percent confidence interval:
##  0.3685646 0.4318916
## sample estimates:
## mean in group Mainstream    mean in group Premium
##                 4.065642                 3.665414
```

t = 24.777, df = 8897.4, p-value < 2.2e-16

#### Maistream vs Budget

```r
young_mainstream_budget <- Data[Data$LIFESTAGE == 'YOUNG SINGLES/COUPLES' &
                                  (Data$PREMIUM_CUSTOMER == 'Mainstream' |
                                     Data$PREMIUM_CUSTOMER == 'Budget'),11:13]
t.test(UNIT_PRICE ~ PREMIUM_CUSTOMER, data = young_mainstream_budget)
```

```
##
##  Welch Two Sample t-test
##
## data:  UNIT_PRICE by PREMIUM_CUSTOMER
## t = -29.522, df = 15099, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group Budget and group Mainstream is not eq
## 95 percent confidence interval:
##  -0.4353828 -0.3811682
## sample estimates:
##     mean in group Budget mean in group Mainstream
##                 3.657366                 4.065642
```

t = -29.522, df = 15099, p-value < 2.2e-16

#### Premium vs Budget

```r
young_premium_budget <- Data[Data$LIFESTAGE == 'YOUNG SINGLES/COUPLES' &
                               (Data$PREMIUM_CUSTOMER == 'Premium' |
                                  Data$PREMIUM_CUSTOMER == 'Budget'),11:13]
t.test(UNIT_PRICE ~ PREMIUM_CUSTOMER, data = young_premium_budget)
```

```
##
##  Welch Two Sample t-test
##
## data:  UNIT_PRICE by PREMIUM_CUSTOMER
```

```
## t = -0.43028, df = 12477, p-value = 0.667
## alternative hypothesis: true difference in means between group Budget and group Premium is not equal
## 95 percent confidence interval:
##  -0.04470709  0.02861232
## sample estimates:
##  mean in group Budget mean in group Premium
##              3.657366              3.665414
```

t = -0.43028, df = 12477, p-value = 0.667

## Midage Singles and Couples

### Mainstream vs Premium

```
midage_mainstream_premium <- Data[Data$LIFESTAGE == 'MIDAGE SINGLES/COUPLES' &
                                          (Data$PREMIUM_CUSTOMER == 'Mainstream' |
                                           Data$PREMIUM_CUSTOMER == 'Premium'),11:13]

t.test(UNIT_PRICE ~ PREMIUM_CUSTOMER, data = midage_mainstream_premium)
```

```
##
##  Welch Two Sample t-test
##
## data:  UNIT_PRICE by PREMIUM_CUSTOMER
## t = 14.059, df = 15715, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group Mainstream and group Premium is not e
## 95 percent confidence interval:
##  0.1923758 0.2547104
## sample estimates:
## mean in group Mainstream    mean in group Premium
##                 3.994241                 3.770698
```

t = 14.059, df = 15715, p-value < 2.2e-16

### Mainstream vs Budget

```
midage_mainstream_budget <- Data[Data$LIFESTAGE == 'MIDAGE SINGLES/COUPLES' &
                                    (Data$PREMIUM_CUSTOMER == 'Mainstream' |
                                       Data$PREMIUM_CUSTOMER == 'Budget'),11:13]
t.test(UNIT_PRICE ~ PREMIUM_CUSTOMER, data = midage_mainstream_budget)
```

```
##
##  Welch Two Sample t-test
##
## data:  UNIT_PRICE by PREMIUM_CUSTOMER
## t = -13.46, df = 8422.7, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group Budget and group Mainstream is not equ
## 95 percent confidence interval:
##  -0.2874536 -0.2143724
```

```
## sample estimates:
##     mean in group Budget mean in group Mainstream
##                 3.743328                 3.994241
```

t = -13.46, df = 8422.7, p-value < 2.2e-16

**Premium vs Budget**

```
midage_premium_budget <- Data[Data$LIFESTAGE == 'MIDAGE SINGLES/COUPLES' &
                                    (Data$PREMIUM_CUSTOMER == 'Premium' |
                                       Data$PREMIUM_CUSTOMER == 'Budget'),11:13]
t.test(UNIT_PRICE ~ PREMIUM_CUSTOMER, data = midage_premium_budget)
```

```
##
##  Welch Two Sample t-test
##
## data:  UNIT_PRICE by PREMIUM_CUSTOMER
## t = -1.3537, df = 9975.6, p-value = 0.1758
## alternative hypothesis: true difference in means between group Budget and group Premium is not equal
## 95 percent confidence interval:
##  -0.06700111  0.01226125
## sample estimates:
##  mean in group Budget mean in group Premium
##              3.743328              3.770698
```

t = -1.3537, df = 9975.6, p-value = 0.1758

When we compare Mainstream customers vs Budget or Premium the t-test results in a very small p-value (when we compare Budget vs Premium, this doesn´t happen), so we can say that the unit price for mainstream, young and mid-age singles and couples are significantly higher compared to their budget and premium counterparts.

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.

# A-priori analysis to deep dive into Mainstream, young singles/couples

## Brands that Mainstream, young singles/couples prefer

### List of young customers, Brands and pack size they bought

```
transaction_Customer_Young <- Data[Data$LIFESTAGE == 'YOUNG SINGLES/COUPLES' &
                              Data$PREMIUM_CUSTOMER == 'Mainstream',c(1,9,10)]
str(transaction_Customer_Young)
```

```
## 'data.frame':    19544 obs. of  3 variables:
##  $ LYLTY_CARD_NBR: int  1002 1010 1018 1018 1018 1020 1020 1020 1060 1060 ...
##  $ PACK_SIZE     : int  150 170 150 165 70 150 330 180 165 270 ...
##  $ BRAND         : chr  "RRD" "Doritos" "Kettle" "RRD" ...
```

#### Change single to basket format and save it

```
library(plyr)
brands_customer <- ddply(transaction_Customer_Young, c("LYLTY_CARD_NBR"),
                         function(df1)paste(df1$BRAND, collapse = ","))
brands_customer$LYLTY_CARD_NBR <- NULL
write.csv(brands_customer, "Brands_Young.csv" , quote = FALSE, row.names = FALSE)
```

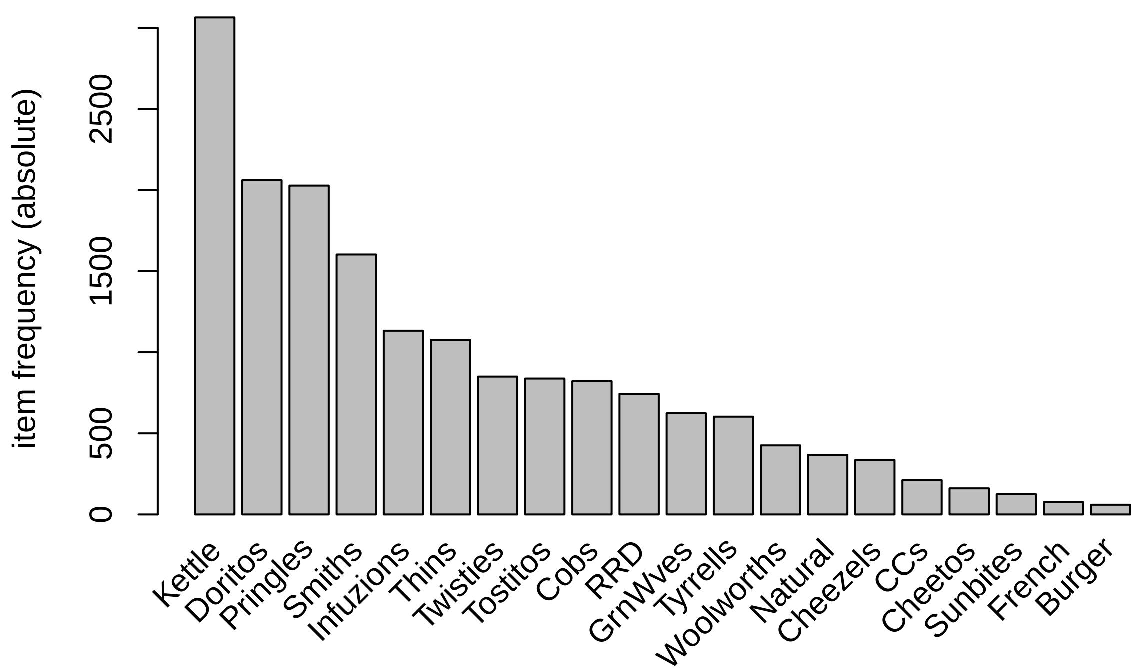
#### Reading the document with brands in the new format

```
library(arules)
transactionYoung <- read.transactions(file = "Brands_Young.csv",
                                      format = "basket",
                                      sep = ",",
                                      header = TRUE)
```

#### Frequency of brands

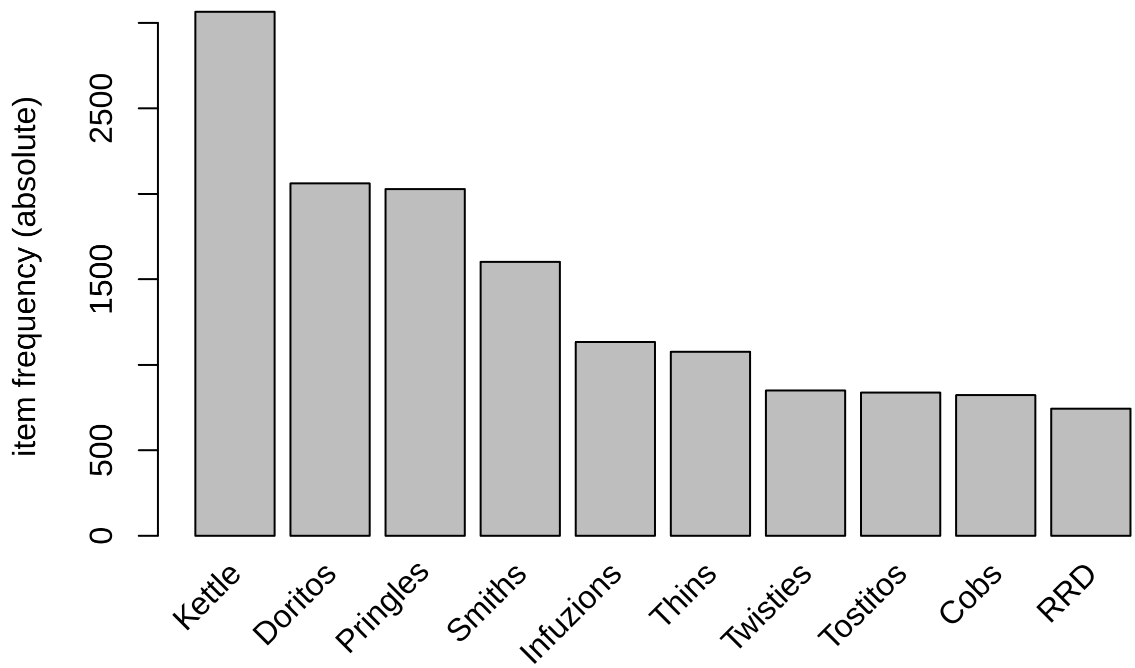
all Brands in Data (20)

```
itemFrequencyPlot(transactionYoung, topN= 20, type='absolute')
```




Top 10 of brands

```r
itemFrequencyPlot(transactionYoung, topN= 10, type='absolute')
```




**A-priori analysis to find frequent itemsets**

We tried the target = 'rules', but any rule was find. So we used 'frequent itemset' instead.

```r
soporte <- 100 / dim(transactionYoung)[1]
rules <- apriori(transactionYoung, parameter = list(supp= soporte,
                                                    conf= 0.7,
                                                    minlen= 1,
                                                    maxlen = 5,
                                                    target = "frequent itemset"))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime    support minlen
##          NA    0.1    1 none FALSE            TRUE       5 0.01263105      1
##  maxlen            target  ext
##       5 frequent itemsets TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
```

```
##
## Absolute minimum support count: 100
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[20 item(s), 7917 transaction(s)] done [0.00s].
## sorting and recoding items ... [18 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## sorting transactions ... done [0.00s].
## writing ... [84 set(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

#### Inspect the itemsets

```r
summary(rules)
```

```
## set of 84 itemsets
##
## most frequent items:
##    Kettle   Doritos    Smiths  Pringles Infuzions   (Other)
##        24        19        19        18        13        67
##
## element (itemset/transaction) length distribution:sizes
##  1  2  3
## 18 56 10
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   1.905   2.000   3.000
##
## summary of quality measures:
##     support             count
##  Min.   :0.01276   Min.   : 101.0
##  1st Qu.:0.01715   1st Qu.: 135.8
##  Median :0.02551   Median : 202.0
##  Mean   :0.04715   Mean   : 373.3
##  3rd Qu.:0.04664   3rd Qu.: 369.2
##  Max.   :0.38714   Max.   :3065.0
##
## includes transaction ID lists: FALSE
##
## mining info:
##              data ntransactions    support confidence
##  transactionYoung          7917 0.01263105          1
##
##  apriori(data = transactionYoung, parameter = list(supp = soporte, conf = 0.7, minlen = 1, maxlen = 5
```

```r
#inspect(rules)

duplicated(rules)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

**Show the first 20 itemsets and save them**

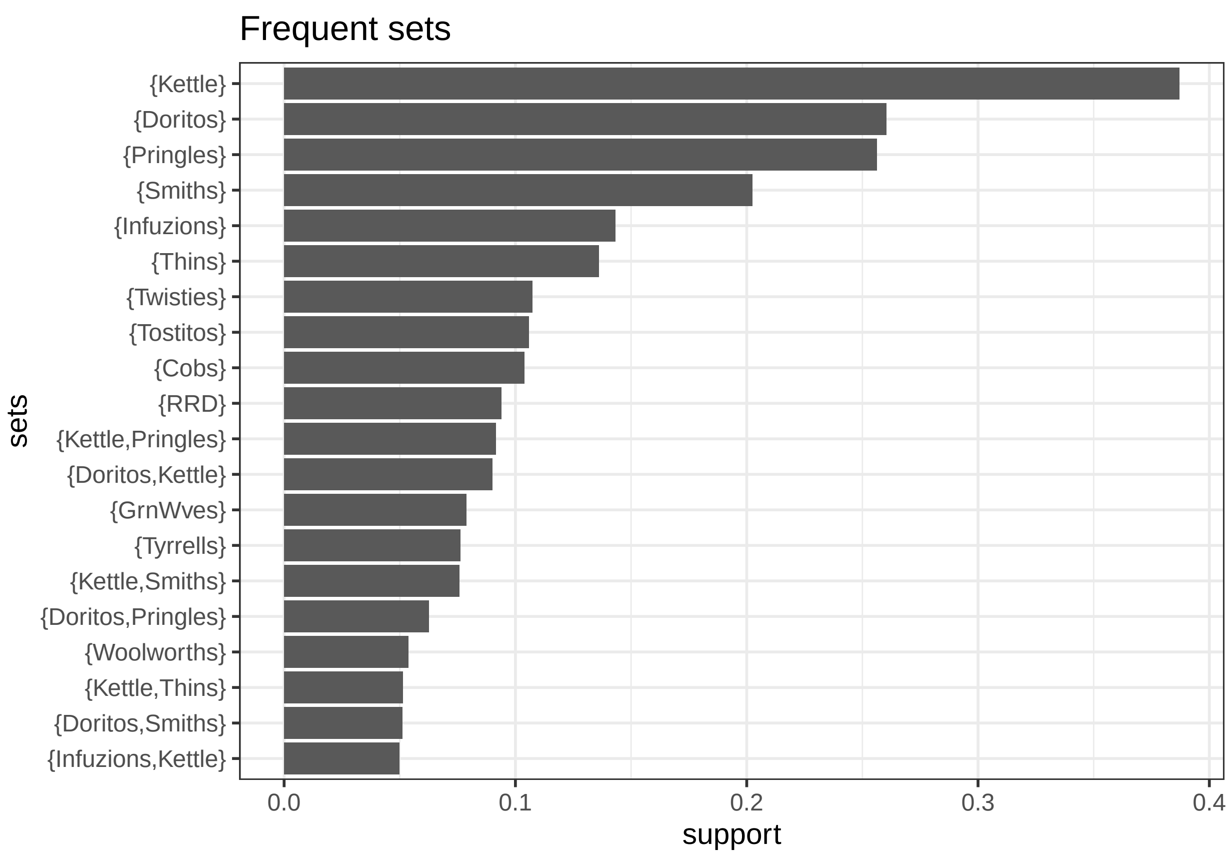
```
top_20_items <- sort(rules, by = "support", decreasing = TRUE)[1:20]
inspect(top_20_items)
```

```
##      items               support    count
## [1]  {Kettle}            0.38714159 3065
## [2]  {Doritos}           0.26032588 2061
## [3]  {Pringles}          0.25615764 2028
## [4]  {Smiths}            0.20247569 1603
## [5]  {Infuzions}         0.14310976 1133
## [6]  {Thins}             0.13603638 1077
## [7]  {Twisties}          0.10736390  850
## [8]  {Tostitos}          0.10584817  838
## [9]  {Cobs}              0.10382721  822
## [10] {RRD}               0.09397499  744
## [11] {Kettle, Pringles}  0.09144878  724
## [12] {Doritos, Kettle}   0.08993306  712
## [13] {GrnWves}           0.07881773  624
## [14] {Tyrrells}          0.07616521  603
## [15] {Kettle, Smiths}    0.07565997  599
## [16] {Doritos, Pringles} 0.06264999  496
## [17] {Woolworths}        0.05380826  426
## [18] {Kettle, Thins}     0.05128205  406
## [19] {Doritos, Smiths}   0.05115574  405
## [20] {Infuzions, Kettle} 0.04976633  394
```

```
df_top20 <- as(top_20_items, Class = "data.frame")
write.csv(df_top20, "top20_young.csv")
```

**Plot top 20 itemsets**

```
ggplot(data = df_top20,
       aes(x = reorder(items, support), y = support)) +
  geom_col() +
  coord_flip() +
  labs(title = "Frequent sets", x = "sets") +
  theme_bw()
```



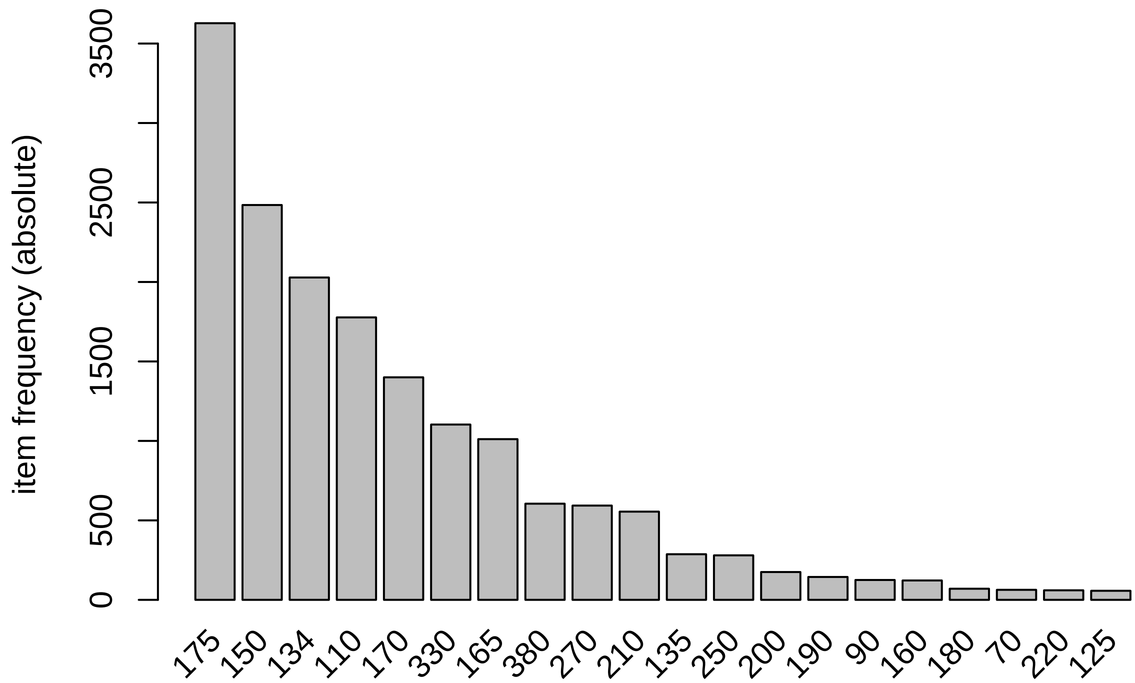

### Pack size that Mainstream, young singles/couples prefer

```
packsize_young <- ddply(transaction_Customer_Young, c("LYLTY_CARD_NBR"),
                        function(df1)paste(df1$PACK_SIZE, collapse = ","))
packsize_young$LYLTY_CARD_NBR <- NULL
write.csv(packsize_young, "PackSize_Young.csv" , quote = FALSE, row.names = FALSE)
```

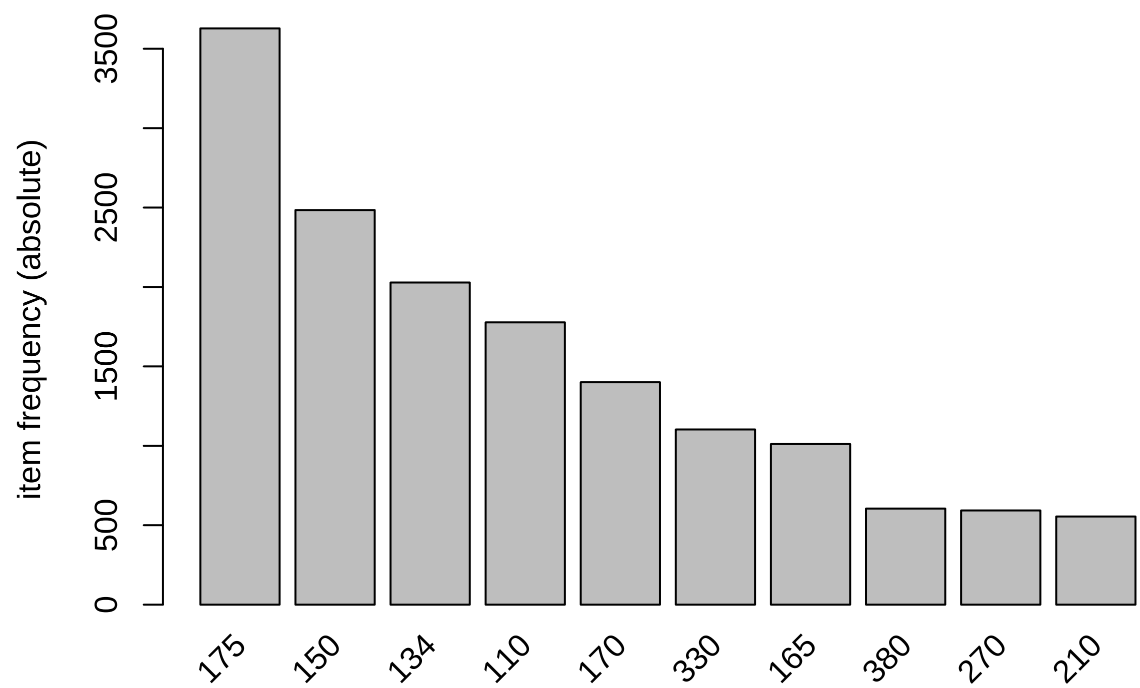
```
packsizeYoung <- read.transactions(file = "PackSize_Young.csv",
                                    format = "basket",
                                    sep = ",",
                                    header = TRUE)
```

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

```r
itemFrequencyPlot(packsizeYoung, topN= 20, type='absolute')#all sizes in Data(20)
```




```r
itemFrequencyPlot(packsizeYoung, topN= 10, type='absolute')
```




```r
soportePackYoung <- 100 / dim(packsizeYoung)[1]
rulesSizeYoung <- apriori(packsizeYoung, parameter = list(supp= soportePackYoung,
                                                          conf= 0.7,
                                                          minlen= 1,
                                                          maxlen = 5,
                                                          target = "frequent itemset"))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime    support minlen
##          NA    0.1    1 none FALSE            TRUE       5 0.01263105      1
##  maxlen            target  ext
##       5 frequent itemsets TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 100
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[20 item(s), 7917 transaction(s)] done [0.00s].
## sorting and recoding items ... [16 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
```

```
## checking subsets of size 1 2 3 4 done [0.00s].
## sorting transactions ... done [0.00s].
## writing ... [73 set(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
summary(rulesSizeYoung)
```

```
## set of 73 itemsets
##
## most frequent items:
##     175     150     134     110     170 (Other)
##      27      21      19      17      15      50
##
## element (itemset/transaction) length distribution:sizes
##  1  2  3
## 16 38 19
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   2.041   3.000   3.000
##
## summary of quality measures:
##     support            count
##  Min.   :0.01263   Min.   : 100.0
##  1st Qu.:0.01730   1st Qu.: 137.0
##  Median :0.02842   Median : 225.0
##  Mean   :0.05422   Mean   : 429.2
##  3rd Qu.:0.05772   3rd Qu.: 457.0
##  Max.   :0.45825   Max.   :3628.0
##
## includes transaction ID lists: FALSE
##
## mining info:
##          data ntransactions    support confidence
##  packsizeYoung          7917 0.01263105          1
##
##  apriori(data = packsizeYoung, parameter = list(supp = soportePackYoung, conf = 0.7, minlen = 1, max
```

```r
#inspect(rulesSizeYoung)

duplicated(rulesSizeYoung)
```

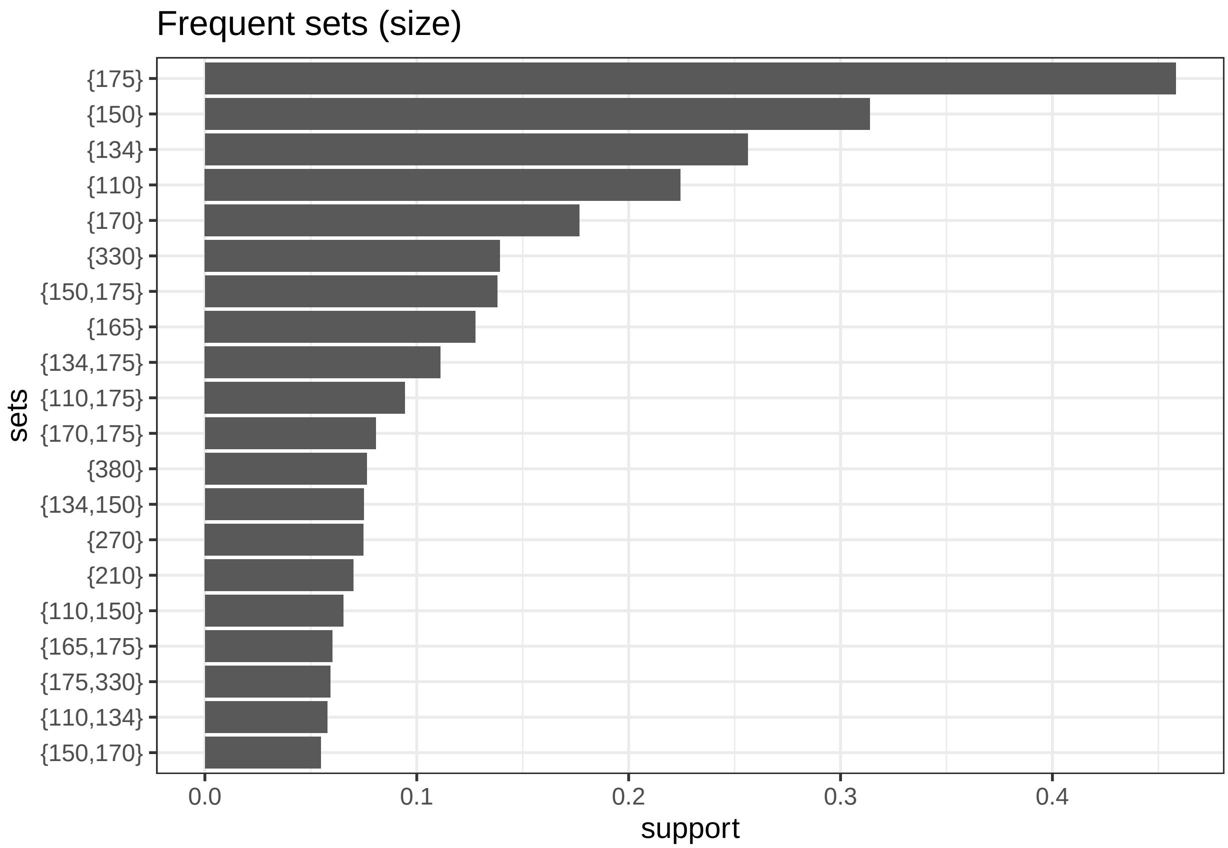
```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [73] FALSE
```

```r
top_20_sizes <- sort(rulesSizeYoung, by = "support", decreasing = TRUE)[1:20]
inspect(top_20_sizes)
```

```
##      items      support    count
## [1]  {175}      0.45825439 3628
## [2]  {150}      0.31375521 2484
## [3]  {134}      0.25615764 2028
## [4]  {110}      0.22445371 1777
## [5]  {170}      0.17683466 1400
## [6]  {330}      0.13932045 1103
## [7]  {150, 175} 0.13793103 1092
## [8]  {165}      0.12769989 1011
## [9]  {134, 175} 0.11115321  880
## [10] {110, 175} 0.09448023  748
## [11] {170, 175} 0.08058608  638
## [12] {380}      0.07641784  605
## [13] {134, 150} 0.07502842  594
## [14] {270}      0.07490211  593
## [15] {210}      0.07010231  555
## [16] {110, 150} 0.06530251  517
## [17] {165, 175} 0.06012378  476
## [18] {175, 330} 0.05911330  468
## [19] {110, 134} 0.05772389  457
## [20] {150, 170} 0.05469243  433
```

```r
df_top20sizes <- as(top_20_sizes, Class = "data.frame")
write.csv(df_top20sizes, "top20Sizes_young.csv")
```

```r
ggplot(data = df_top20sizes,
       aes(x = reorder(items, support), y = support)) +
  geom_col() +
  coord_flip() +
  labs(title = "Frequent sets (size)", x = "sets") +
  theme_bw()
```




## A-priori analysis to deep dive into Budget Older families

Although there aren't too many customers in the Budget Older families segment, they make more transactions and contribute to the sales, so we can also check their preferences.

### Brands that Budget Older families prefer

```
transaction_Budget_OldFam <- Data[Data$LIFESTAGE == 'OLDER FAMILIES' &
                                     Data$PREMIUM_CUSTOMER == 'Budget',c(1,9,10)]
str(transaction_Budget_OldFam)
```

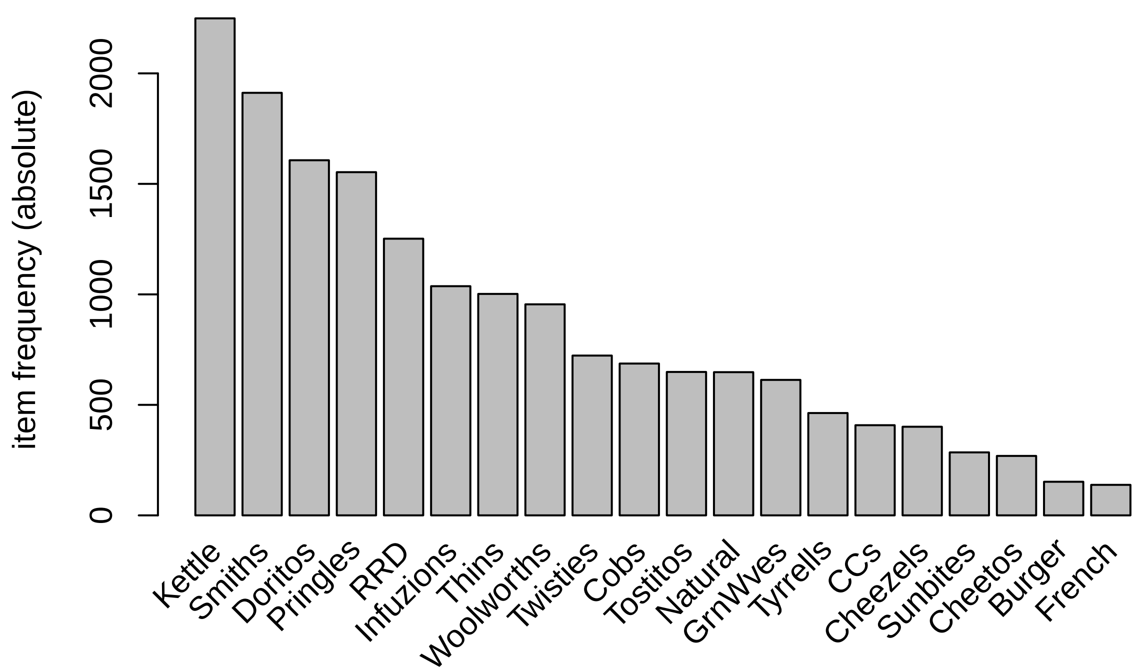
```
## 'data.frame':    21514 obs. of  3 variables:
##  $ LYLTY_CARD_NBR: int  1022 1090 1102 1103 1136 1190 1226 1235 1250 1281 ...
##  $ PACK_SIZE     : int  150 165 175 170 175 150 150 175 170 150 ...
##  $ BRAND         : chr  "Kettle" "RRD" "CCs" "Smiths" ...
```

```
brands_OldFam <- ddply(transaction_Budget_OldFam, c("LYLTY_CARD_NBR"),
                         function(df1)paste(df1$BRAND, collapse = ","))
brands_OldFam$LYLTY_CARD_NBR <- NULL
write.csv(brands_OldFam, "Brands_Butget_OldFam.csv" , quote = FALSE, row.names = FALSE)
```

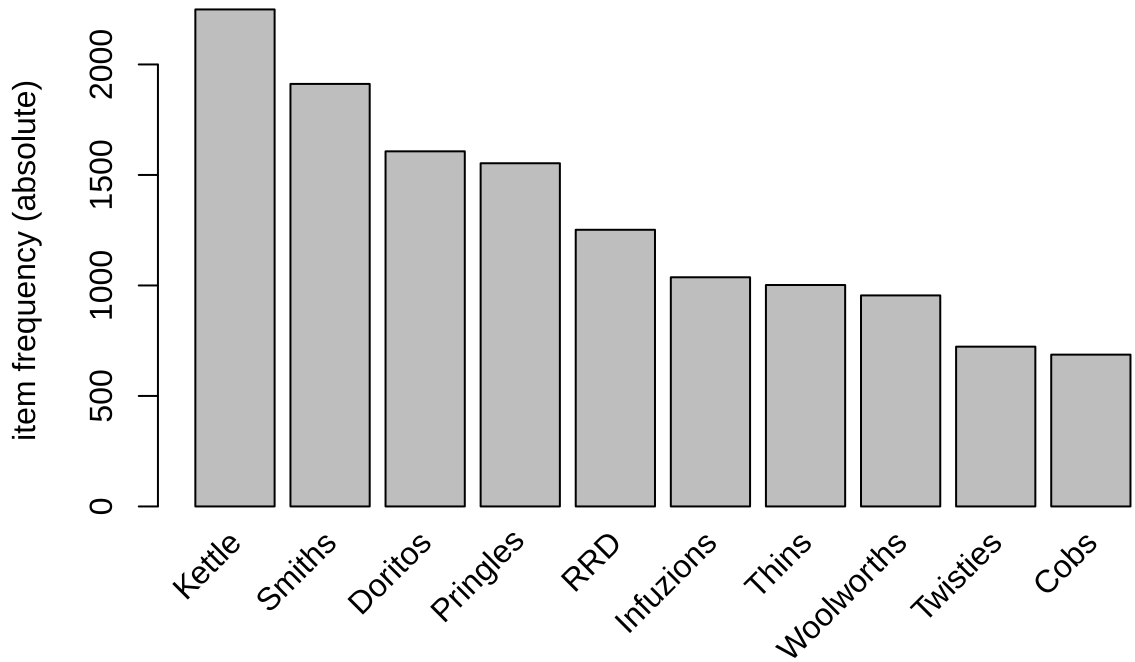
```r
transactionOldFam <- read.transactions(file = "Brands_Butget_OldFam.csv",
                                       format = "basket",
                                       sep = ",",
                                       header = TRUE)
```

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

```r
itemFrequencyPlot(transactionOldFam, topN= 20, type='absolute')#all Brands in Data(23)
```




```r
itemFrequencyPlot(transactionOldFam, topN= 10, type='absolute')#all Brands in Data(23)
```




```r
soporteBrandOldFam <- 200 / dim(transactionOldFam)[1]
rulesBrandOldFam <- apriori(transactionOldFam, parameter = list(supp= soporteBrandOldFam,
                                                    conf= 0.7,
                                                    minlen= 1,
                                                    maxlen = 5,
                                                  target = "frequent itemset"))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime    support minlen
##          NA    0.1    1 none FALSE            TRUE       5 0.04337454      1
##  maxlen            target  ext
##       5 frequent itemsets TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 200
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[20 item(s), 4611 transaction(s)] done [0.00s].
## sorting and recoding items ... [18 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
```

```
## checking subsets of size 1 2 3 4 done [0.00s].
## sorting transactions ... done [0.00s].
## writing ... [101 set(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
summary(rulesBrandOldFam)
```

```
## set of 101 itemsets
##
## most frequent items:
##   Smiths   Kettle  Doritos Pringles      RRD  (Other)
##       32       31       22       21       21       81
##
## element (itemset/transaction) length distribution:sizes
##  1  2  3
## 18 59 24
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   2.059   2.000   3.000
##
## summary of quality measures:
##     support             count
##  Min.   :0.04337   Min.   : 200.0
##  1st Qu.:0.05357   1st Qu.: 247.0
##  Median :0.06940   Median : 320.0
##  Mean   :0.09978   Mean   : 460.1
##  3rd Qu.:0.10280   3rd Qu.: 474.0
##  Max.   :0.48775   Max.   :2249.0
##
## includes transaction ID lists: FALSE
##
## mining info:
##               data ntransactions    support confidence
##  transactionOldFam          4611 0.04337454          1
##
##  apriori(data = transactionOldFam, parameter = list(supp = soporteBrandOldFam, conf = 0.7, minlen = 
```

```r
duplicated(rulesBrandOldFam)
```

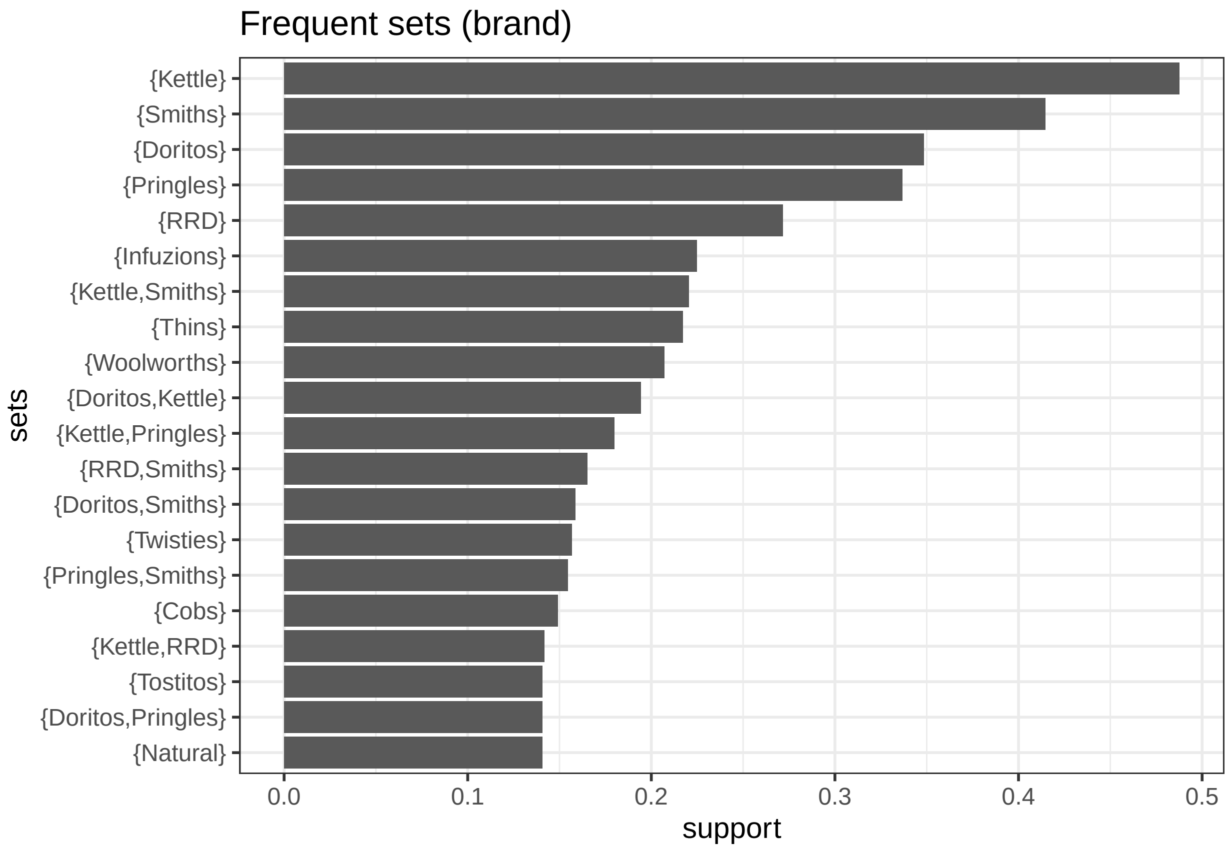
```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE FALSE FALSE
```

```r
top20_items_OldFam <- sort(rulesBrandOldFam, by = "support", decreasing = TRUE)[1:20]
inspect(top20_items_OldFam)
```

```
##      items               support   count
## [1]  {Kettle}            0.4877467 2249
## [2]  {Smiths}            0.4146606 1912
## [3]  {Doritos}           0.3485144 1607
## [4]  {Pringles}          0.3368033 1553
## [5]  {RRD}               0.2715246 1252
## [6]  {Infuzions}         0.2248970 1037
## [7]  {Kettle, Smiths}    0.2205595 1017
## [8]  {Thins}             0.2173064 1002
## [9]  {Woolworths}        0.2071134  955
## [10] {Doritos, Kettle}   0.1943179  896
## [11] {Kettle, Pringles}  0.1797875  829
## [12] {RRD, Smiths}       0.1650401  761
## [13] {Doritos, Smiths}   0.1587508  732
## [14] {Twisties}          0.1567990  723
## [15] {Pringles, Smiths}  0.1546302  713
## [16] {Cobs}              0.1489915  687
## [17] {Kettle, RRD}       0.1418347  654
## [18] {Tostitos}          0.1407504  649
## [19] {Doritos, Pringles} 0.1407504  649
## [20] {Natural}           0.1405335  648
```

```
df_top20_OldFam <- as(top20_items_OldFam, Class = "data.frame")
write.csv(df_top20_OldFam, "top20_Budget_OldFam.csv")
```

```
ggplot(data = df_top20_OldFam,
       aes(x = reorder(items, support), y = support)) +
  geom_col() +
  coord_flip() +
  labs(title = "Frequent sets (brand)", x = "sets") +
  theme_bw()
```



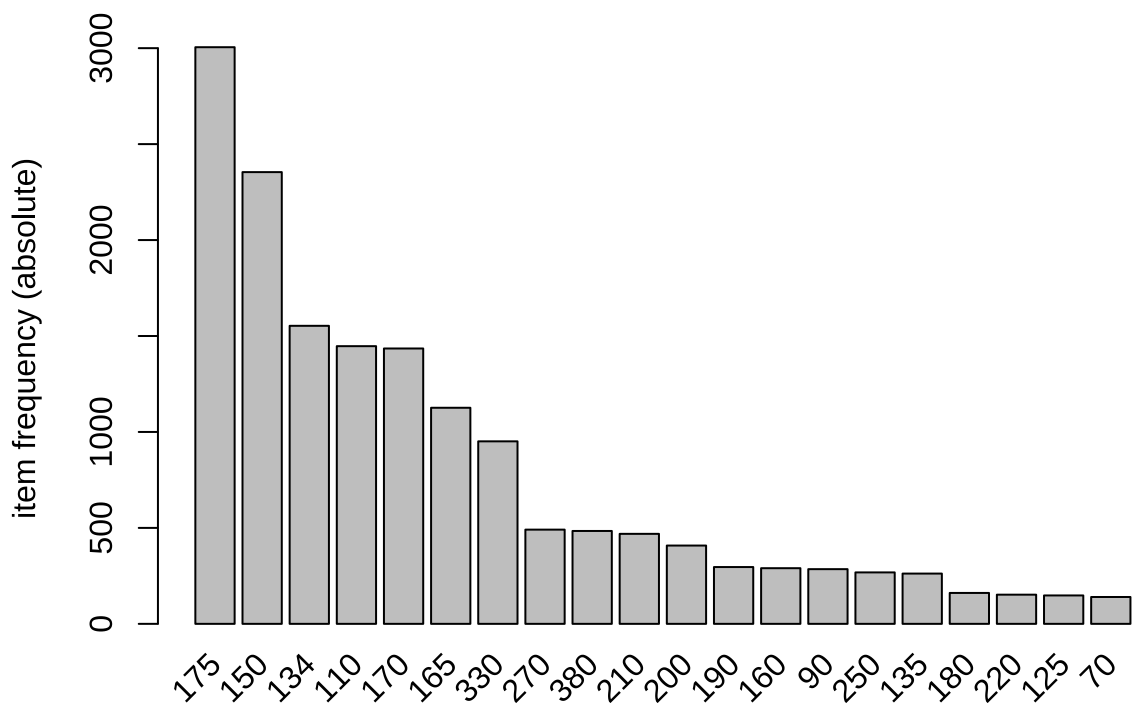

### Pack size taht Budget Older families prefer

```
packsize_OldFam <- ddply(transaction_Budget_OldFam, c("LYLTY_CARD_NBR"),
                         function(df1)paste(df1$PACK_SIZE, collapse = ","))
packsize_OldFam$LYLTY_CARD_NBR <- NULL
write.csv(packsize_OldFam, "PackSize_Buget_OldFam.csv" , quote = FALSE, row.names = FALSE)
```

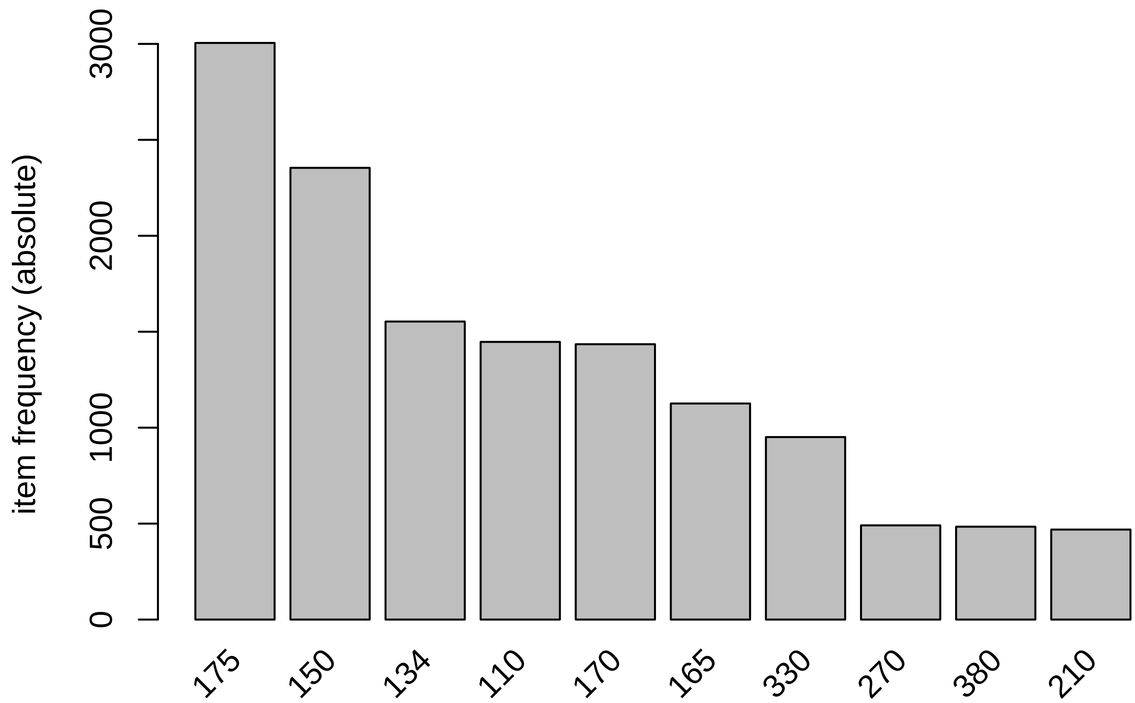
```
packsizeOldFam <- read.transactions(file = "PackSize_Buget_OldFam.csv",
                                    format = "basket",
                                    sep = ",",
                                    header = TRUE)
```

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

```r
itemFrequencyPlot(packsizeOldFam, topN= 20, type='absolute')#all sizes in Data(20)
```




```r
itemFrequencyPlot(packsizeOldFam, topN= 10, type='absolute')
```




```r
soportePackOldFam <- 200 / dim(packsizeOldFam)[1]
rulesSizeOldFam <- apriori(packsizeOldFam, parameter = list(supp= soportePackOldFam,
                                                           conf= 0.7,
                                                           minlen= 1,
                                                           maxlen = 5,
                                                           target = "frequent itemset"))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime    support minlen
##          NA    0.1    1 none FALSE            TRUE       5 0.04337454      1
##  maxlen            target  ext
##       5 frequent itemsets TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 200
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[20 item(s), 4611 transaction(s)] done [0.00s].
## sorting and recoding items ... [16 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
```

```
## checking subsets of size 1 2 3 4 done [0.00s].
## sorting transactions ... done [0.00s].
## writing ... [85 set(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
summary(rulesSizeOldFam)
```

```
## set of 85 itemsets
##
## most frequent items:
##     175     150     134     110     170 (Other)
##      39      36      21      20      20      59
##
## element (itemset/transaction) length distribution:sizes
##  1  2  3  4
## 16 34 29  6
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   2.294   3.000   4.000
##
## summary of quality measures:
##     support            count
##  Min.   :0.04403   Min.   : 203.0
##  1st Qu.:0.05769   1st Qu.: 266.0
##  Median :0.07395   Median : 341.0
##  Mean   :0.11381   Mean   : 524.8
##  3rd Qu.:0.12058   3rd Qu.: 556.0
##  Max.   :0.65170   Max.   :3005.0
##
## includes transaction ID lists: FALSE
##
## mining info:
##            data ntransactions    support confidence
##  packsizeOldFam          4611 0.04337454          1
##
##  apriori(data = packsizeOldFam, parameter = list(supp = soportePackOldFam, conf = 0.7, minlen = 1, ma
```

```r
duplicated(rulesSizeOldFam)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [85] FALSE
```

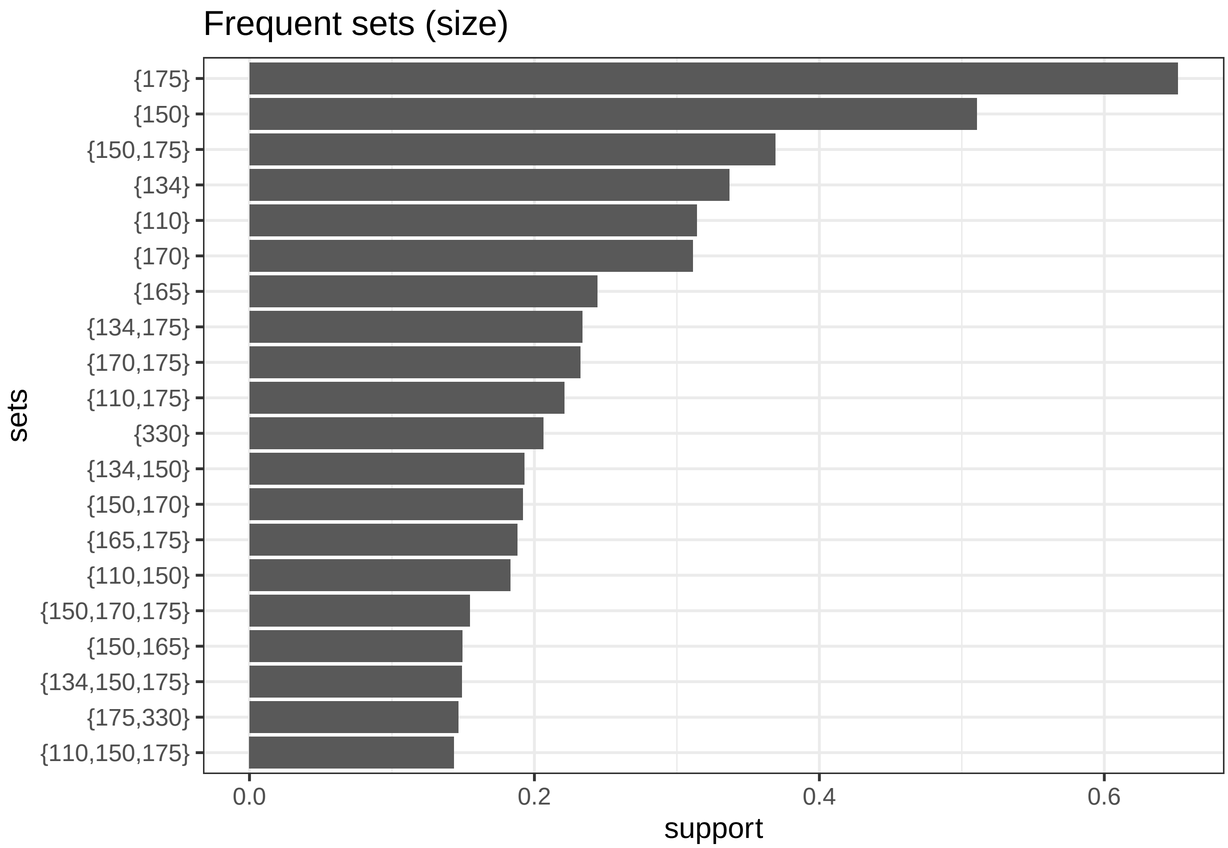
```r
top_20_sizes_OF <- sort(rulesSizeOldFam, by = "support", decreasing = TRUE)[1:20]
inspect(top_20_sizes_OF)
```

```
##      items           support   count
```

```
## [1]  {175}           0.6517025 3005
## [2]  {150}           0.5105183 2354
## [3]  {150, 175}      0.3691173 1702
## [4]  {134}           0.3368033 1553
## [5]  {110}           0.3138148 1447
## [6]  {170}           0.3112123 1435
## [7]  {165}           0.2441987 1126
## [8]  {134, 175}      0.2335719 1077
## [9]  {170, 175}      0.2322707 1071
## [10] {110, 175}      0.2209933 1019
## [11] {330}           0.2062459  951
## [12] {134, 150}      0.1927998  889
## [13] {150, 170}      0.1917155  884
## [14] {165, 175}      0.1880286  867
## [15] {110, 150}      0.1830406  844
## [16] {150, 170, 175} 0.1546302  713
## [17] {150, 165}      0.1492084  688
## [18] {134, 150, 175} 0.1489915  687
## [19] {175, 330}      0.1463891  675
## [20] {110, 150, 175} 0.1435697  662
```

```
df_top20sizesOldFam <- as(top_20_sizes_OF, Class = "data.frame")
write.csv(df_top20sizesOldFam, "top20Sizes_OldFam.csv")
```

```
ggplot(data = df_top20sizesOldFam,
       aes(x = reorder(items, support), y = support)) +
  geom_col() +
  coord_flip() +
  labs(title = "Frequent sets (size)", x = "sets") +
  theme_bw()
```




## Conclusion

Recapitulation of what we have found.

Sales are coming mainly from Budget - older families, Mainstream - young singles/couples, and Mainstream - retirees shoppers, the same for the number of transactions. There are more Mainstream - young singles/couples and retirees who buy chips. This contributes to there being more sales from these customer segments but in the case of the Budget - Older families segment, they contribute making more transactions of chips.

Older families and young families in general buy more chips per customer, maybe one member of the family buys for all the other members.

Mainstream midage and young singles/couples are more willing to pay more per packet of chips compared to their budget and premium counterparts.

Three of the first brands that Mainstream young singles and couples prefer are Kettle, Doritos, and Pringles in packet sizes of 175g or 150g.

Although there aren't too many customers in the Budget Older families segment, they make more transactions and contribute to the sales, three of the first brands that they prefer are Kettle, Smiths, and Doritos in packet sizes of 175g or 150g.

Both segments coincide in the brand Kettle and Doritos and the packet sizes.