quantium_Task_1

Rajesh Dhungna

2025-01-06

install.packages("dplyr") install.packages("tidyverse") install.packages("readxl") install.packages("data.table") install.packages("ggplot2") install.packages("ggmosaic") install.packages("readr")

```r
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.3.3

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

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

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.3

## Warning: package 'ggplot2' was built under R version 4.3.3

## Warning: package 'tidyr' was built under R version 4.3.3

## Warning: package 'readr' was built under R version 4.3.3

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ forcats   1.0.0     ✓ readr     2.1.5
## ✓ ggplot2   3.5.1     ✓ stringr   1.5.0
## ✓ lubridate 1.9.3     ✓ tibble    3.2.1
## ✓ purrr     1.0.2     ✓ tidyr     1.3.1

## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(readxl)

## Warning: package 'readxl' was built under R version 4.3.3

library(data.table)

## Warning: package 'data.table' was built under R version 4.3.3

##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following object is masked from 'package:purrr':
##
##     transpose
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last

library(ggplot2)
library(ggmosaic)

## Warning: package 'ggmosaic' was built under R version 4.3.3

library(readr)

transaction <- read_excel("transaction_data.xlsx")

customer <- read.csv("purchase_behaviour.csv")
```

Exploratory Data Analysis

Lets deep dive into transaction data

```r
#Shape of transaction data
dim(transaction)

## [1] 264836      8

#structure of transaction data
str(transaction)

## tibble [264,836 × 8] (S3: tbl_df/tbl/data.frame)
##  $ DATE          : num [1:264836] 43390 43599 43605 43329 43330 ...
##  $ STORE_NBR     : num [1:264836] 1 1 1 2 2 4 4 4 5 7 ...
##  $ LYLTY_CARD_NBR: num [1:264836] 1000 1307 1343 2373 2426 ...
##  $ TXN_ID        : num [1:264836] 1 348 383 974 1038 ...
##  $ PROD_NBR      : num [1:264836] 5 66 61 69 108 57 16 24 42 52 ...
```

```
## $ PROD_NAME     : chr [1:264836] "Natural Chip        Compny SeaSalt175g"
"CCs Nacho Cheese    175g" "Smiths Crinkle Cut  Chips Chicken 170g" "Smiths
Chip Thinly  S/Cream&Onion 175g" ...
## $ PROD_QTY      : num [1:264836] 2 3 2 5 3 1 1 1 1 2 ...
## $ TOT_SALES     : num [1:264836] 6 6.3 2.9 15 13.8 5.1 5.7 3.6 3.9 7.2
...
```

```r
#First 5 rows of transaction data
head(transaction)
```

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

```r
# Checking for any missing values
any(is.na(transaction))
```

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

```r
#Data type of Date column
class(transaction$DATE)
```

```
## [1] "numeric"
```

```r
#changing Date column to date type
transaction$DATE <- as.Date(transaction$DATE, origin = "1899-12-10")
```

```r
#Rechecking the data type of Date
class(transaction$DATE)
```

```
## [1] "Date"
```

Lets focus into PROD_NAME and get some insights

```r
#Checking the summary of PROD_NAME
summary(transaction$PROD_NAME)
```

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

It doesn't give us much information about the PROD_NAME, so lets try other way.

```
#Listing the unique values with number of occurrence
#table(transaction$PROD_NAME)
```

Lets organize above result to understand the data more clearly.

```
#Grouping frequently occurring words and arranging them in descending order
freq_words <- transaction %>% group_by(PROD_NAME)%>% summarize(count = n())
%>%
arrange(desc(count))

freq_words

## # A tibble: 114 × 2
##    PROD_NAME                                count
##    <chr>                                    <int>
##  1 Kettle Mozzarella   Basil & Pesto 175g    3304
##  2 Kettle Tortilla ChpsHny&Jlpno Chili 150g  3296
##  3 Cobs Popd Swt/Chlli &Sr/Cream Chips 110g  3269
##  4 Tyrrells Crisps     Ched & Chives 165g    3268
##  5 Cobs Popd Sea Salt  Chips 110g            3265
##  6 Kettle 135g Swt Pot Sea Salt              3257
##  7 Tostitos Splash Of  Lime 175g             3252
##  8 Infuzions Thai SweetChili PotatoMix 110g  3242
##  9 Smiths Crnkle Chip  Orgnl Big Bag 380g    3233
## 10 Thins Potato Chips  Hot & Spicy 175g      3229
## # ℹ 104 more rows
```

Since we are interested in the word chip or chips, lets split the product name into the individual words and count the frequency of occurrence of each words

```
#Creating dictionary words table
productWords <- data.table(unlist(strsplit(unique(transaction$PROD_NAME), "
")))

#Changing column name to words
setnames(productWords, 'words')

#productWords
```

Lets remove any digits and special characters from the words. We will utilize regular expression for this task.

```
#Removing digits and white space using regular expression
productWords <- productWords[grepl("^[a-zA-Z]+$", words)]
```

Further, lets arrange these words in descending order of their frequency

```r
#Generating word frequency and sorting in descending order
productWords <- productWords %>%
  group_by(words) %>%
  summarise(frequency = n(), .groups = 'drop') %>%
  arrange(desc(frequency))
```

Lets see the result

```r
productWords

## # A tibble: 168 × 2
##    words     frequency
##    <chr>         <int>
##  1 Chips            21
##  2 Smiths           16
##  3 Crinkle          14
##  4 Cut              14
##  5 Kettle           13
##  6 Cheese           12
##  7 Salt             12
##  8 Original         10
##  9 Chip              9
## 10 Doritos           9
## # ℹ 158 more rows

#Changing our transaction data frame into table
transaction <- data.table(transaction)
```

Lets get rid of word like salsa from our transaction data

```r
#Removing salsa product from transaction data
transaction[, SALSA := grepl("salsa", tolower(PROD_NAME))]
transaction <- transaction[SALSA == FALSE, ][, SALSA := NULL]
```

Checking the summary of our transaction data

```r
#Stastistical values of columns
summary(transaction)

##       DATE                STORE_NBR     LYLTY_CARD_NBR        TXN_ID
##  Min.   :2018-06-11   Min.   :  1.0   Min.   :   1000   Min.   :      1
##  1st Qu.:2018-09-10   1st Qu.: 70.0   1st Qu.:  70015   1st Qu.:  67569
##  Median :2018-12-10   Median :130.0   Median : 130367   Median : 135183
##  Mean   :2018-12-10   Mean   :135.1   Mean   : 135531   Mean   : 135131
##  3rd Qu.:2019-03-11   3rd Qu.:203.0   3rd Qu.: 203084   3rd Qu.: 202654
##  Max.   :2019-06-10   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
```

From above summary we can see that 200 items of Doritos Corn Chip Supreme 380g was bought by same customer. We will investigate the]is transaction to see if its is outlier or regular transaction.

```
#Lets organize our transaction data to see possible outliers
transaction[order(-PROD_QTY),]

##               DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##             <Date>     <num>          <num>  <num>    <num>
##      1: 2018-07-30       226         226000 226201        4
##      2: 2019-04-30       226         226000 226210        4
##      3: 2018-07-28         2           2373    974       69
##      4: 2018-07-31         8           8294   8221      114
##      5: 2019-04-26        74          74336  73182       84
##     ---
## 246738: 2018-09-17       268         268396 264841        8
## 246739: 2018-10-02       268         268463 264916       87
## 246740: 2019-04-08       268         268491 264947       56
## 246741: 2019-02-21       272         272193 269906        9
## 246742: 2018-07-24       272         272358 270154       74
##                                        PROD_NAME PROD_QTY TOT_SALES
##                                           <char>    <num>     <num>
##      1:         Dorito Corn Chp     Supreme 380g      200     650.0
##      2:         Dorito Corn Chp     Supreme 380g      200     650.0
##      3:  Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0
##      4:   Kettle Sensations   Siracha Lime 150g        5      23.0
##      5:   GrnWves Plus Btroot & Chilli Jam 180g        5      15.5
##     ---
## 246738: Smiths Crinkle Cut  Chips Original 170g        1       2.9
## 246739: Infuzions BBQ Rib   Prawn Crackers 110g        1       3.8
## 246740:               Cheezels Cheese Box 125g        1       2.1
## 246741: Kettle Tortilla ChpsBtroot&Ricotta 150g        1       4.6
## 246742:            Tostitos Splash Of  Lime 175g        1       4.4

#Lets filter our transaction data to see the transaction of particular
customer
trans_outlier <- transaction[transaction$LYLTY_CARD_NBR == 226000,]

#Lets check our result
trans_outlier

##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##        <Date>     <num>          <num>  <num>    <num>
## 1: 2018-07-30       226         226000 226201        4
## 2: 2019-04-30       226         226000 226210        4
##                             PROD_NAME PROD_QTY TOT_SALES
##                                <char>    <num>     <num>
```

```
## 1: Dorito Corn Chp     Supreme 380g      200       650
## 2: Dorito Corn Chp     Supreme 380g      200       650
```

We can see that this customer has had only 2 transaction in a year. This might be for commercial purpose, hence we will remove this transaction for further analysis.

```
#Removing rows with following LYLTY_CARD_NBR
transaction <- subset(transaction,LYLTY_CARD_NBR != 226000)

#Checking the result
transaction

##               DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##             <Date>     <num>          <num>  <num>    <num>
##       1: 2018-09-27         1           1000      1        5
##       2: 2019-04-24         1           1307    348       66
##       3: 2019-04-30         1           1343    383       61
##       4: 2018-07-28         2           2373    974       69
##       5: 2018-07-29         2           2426   1038      108
##      ---
## 246736: 2019-02-17       272         272319 270088       89
## 246737: 2018-07-24       272         272358 270154       74
## 246738: 2018-10-17       272         272379 270187       51
## 246739: 2018-12-07       272         272379 270188       42
## 246740: 2018-09-02       272         272380 270189       74
##                                        PROD_NAME PROD_QTY TOT_SALES
##                                           <char>    <num>     <num>
##       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
##      ---
## 246736:  Kettle Sweet Chilli And Sour Cream 175g        2      10.8
## 246737:            Tostitos Splash Of  Lime 175g        1       4.4
## 246738:                 Doritos Mexicana    170g        2       8.8
## 246739:  Doritos Corn Chip Mexican Jalapeno 150g        2       7.8
## 246740:            Tostitos Splash Of  Lime 175g        2       8.8
```

Lets organize our data by transaction date

```
#Lets create a table with two columns date and frequency
transactionDate <- transaction %>%
  group_by(DATE) %>%
  summarise(frequency = n(), .groups = 'drop')

#Lets see the result in descending order of frequency
transactionDate[order(-transactionDate$frequency),]

## # A tibble: 364 × 2
##    DATE       frequency
```

```
##    <date>         <int>
##  1 2018-12-04       865
##  2 2018-12-03       853
##  3 2018-12-02       840
##  4 2018-11-29       839
##  5 2018-11-30       808
##  6 2018-11-28       799
##  7 2018-12-01       781
##  8 2019-05-18       762
##  9 2018-08-17       745
## 10 2019-05-25       743
## # ℹ 354 more rows
```

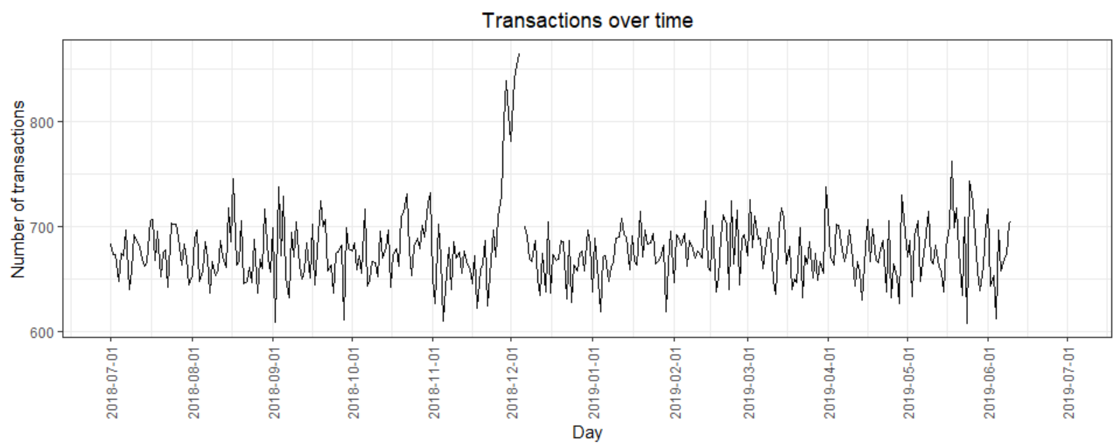
```r
#Creating a date chart starting from 1 Jul 2018 to 30 Jun 2019
dateChart <- data.table(
  DATE = seq(from = as.Date("2018-07-01"),
             to = as.Date("2019-06-30"),
             by = "day")
)

#dateChart

#Lets join two tables transactionDate and dateChart to see the missing
transaction date

missingtrans_Date <- merge.data.table(dateChart, transactionDate, by= "DATE",
all.x = TRUE)
missingtrans_Date
```

```
## Key: <DATE>
##            DATE frequency
##          <Date>     <int>
##   1: 2018-07-01       683
##   2: 2018-07-02       673
##   3: 2018-07-03       673
##   4: 2018-07-04       648
##   5: 2018-07-05       674
##  ---
## 361: 2019-06-26        NA
## 362: 2019-06-27        NA
## 363: 2019-06-28        NA
## 364: 2019-06-29        NA
## 365: 2019-06-30        NA
```

Visualizing the transaction trend over date

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

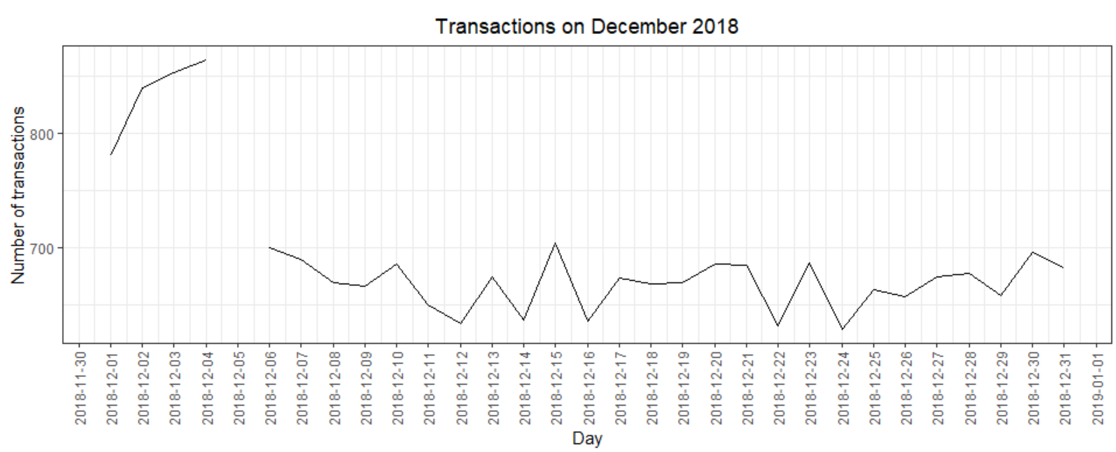
```r
# Plot transactions over time
ggplot(missingtrans_Date, aes(x = DATE, y = frequency)) +
geom_line() +
labs(x = "Day", y = "Number of transactions", title = "Transactions over
time") +
scale_x_date(breaks = "1 month") +
theme(axis.text.x = element_text(angle = 90, vjust = 0.5))

## Warning: Removed 20 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```



From this visualization we can see that there is increase in sales in the month of December but the graph breaks during the late December. Lets zoom into the month of December to see the sales trends.

```r
#Lets filter our table for transaction that occurred in December only.

trans_dec <- subset(missingtrans_Date, DATE >= "2018-12-01" & DATE < "2019-
01-01")
trans_dec

## Key: <DATE>
##            DATE frequency
##          <Date>     <int>
##  1: 2018-12-01       781
##  2: 2018-12-02       840
##  3: 2018-12-03       853
##  4: 2018-12-04       865
##  5: 2018-12-05        NA
##  6: 2018-12-06       700
##  7: 2018-12-07       690
##  8: 2018-12-08       669
##  9: 2018-12-09       666
## 10: 2018-12-10       686
## 11: 2018-12-11       650
## 12: 2018-12-12       634
```

```
## 13: 2018-12-13       674
## 14: 2018-12-14       637
## 15: 2018-12-15       704
## 16: 2018-12-16       636
## 17: 2018-12-17       673
## 18: 2018-12-18       668
## 19: 2018-12-19       669
## 20: 2018-12-20       686
## 21: 2018-12-21       685
## 22: 2018-12-22       631
## 23: 2018-12-23       687
## 24: 2018-12-24       628
## 25: 2018-12-25       663
## 26: 2018-12-26       657
## 27: 2018-12-27       674
## 28: 2018-12-28       677
## 29: 2018-12-29       658
## 30: 2018-12-30       696
## 31: 2018-12-31       683
##             DATE frequency

#Lets visualize the result
ggplot(trans_dec, aes(x = DATE, y = frequency)) +
geom_line() +
labs(x = "Day", y = "Number of transactions", title = "Transactions on
December 2018") +
scale_x_date(breaks = "1 day") +
theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



From the above graph we can see that there was no any transaction on 5 Dec 2018. There is no specific reason for this to happen. May be the data for this date is missing due to technical reason. So, we will remove transaction for this date. Further, we see fluctuating sales trend from 10 to 17 Dec and 22 to 25 Dec.

Creating a pack size from transaction data

```r
#Creating a data table named PACK_SIZE from transaction data
transaction[, PACK_SIZE := parse_number(PROD_NAME)]

#Lets Check our result
transaction

##               DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##             <Date>     <num>          <num>  <num>    <num>
##      1: 2018-09-27         1           1000      1        5
##      2: 2019-04-24         1           1307    348       66
##      3: 2019-04-30         1           1343    383       61
##      4: 2018-07-28         2           2373    974       69
##      5: 2018-07-29         2           2426   1038      108
##     ---
## 246736: 2019-02-17       272         272319 270088       89
## 246737: 2018-07-24       272         272358 270154       74
## 246738: 2018-10-17       272         272379 270187       51
## 246739: 2018-12-07       272         272379 270188       42
## 246740: 2018-09-02       272         272380 270189       74
##                                        PROD_NAME PROD_QTY TOT_SALES
PACK_SIZE
##                                           <char>    <num>     <num>
<num>
##      1:   Natural Chip        Compny SeaSalt175g        2       6.0
175
##      2:                 CCs Nacho Cheese    175g        3       6.3
175
##      3:   Smiths Crinkle Cut  Chips Chicken 170g        2       2.9
170
##      4:   Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0
175
##      5: Kettle Tortilla ChpsHny&Jlpno Chili 150g        3      13.8
150
##     ---
## 246736:  Kettle Sweet Chilli And Sour Cream 175g        2      10.8
175
## 246737:            Tostitos Splash Of  Lime 175g        1       4.4
175
## 246738:                 Doritos Mexicana    170g        2       8.8
170
## 246739:  Doritos Corn Chip Mexican Jalapeno 150g        2       7.8
150
## 246740:            Tostitos Splash Of  Lime 175g        2       8.8
175
```

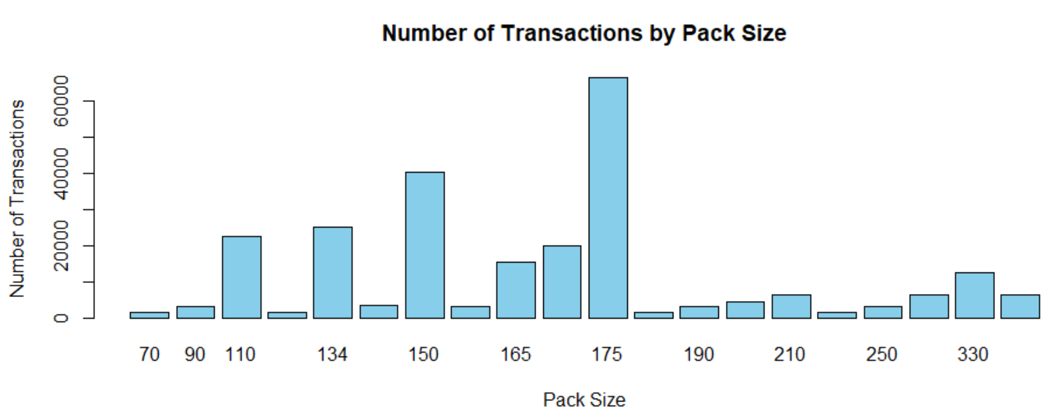
Lets organize our result by sorting transaction as per pack size and number of transactions per pack size

```r
packSizeFre <- transaction[, .N, PACK_SIZE][order(PACK_SIZE)]
packSizeFre
```

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

Lets visualize above result using a histogram showing the number of transaction by pack size

```r
barplot(
  packSizeFre$N,
  names.arg = packSizeFre$PACK_SIZE,
  main = "Number of Transactions by Pack Size",
  xlab = "Pack Size",
  ylab = "Number of Transactions",
  col = "skyblue",
  border = "black"
)
```



From the above histogram we can see that the highest number of transaction was for the chip with size of 175 gram.

Now lets see which chips brand has the highest transactions.Looking into the PROD_NAME we can see that the first word is brand name. So lets extract first words from PROD_NAME

```
#Creating a colum BRAD in transaction data
transaction[, BRAND := word(PROD_NAME,1)]
```

Lets investigate further into Brands

```
#Creating a frequency count of each brands
transBrand <- transaction[, .N, BRAND][order(-N)]
transBrand

##          BRAND     N
##         <char> <int>
##  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
```

As per the instruction brand like RED abd RRD are similar. So lets rename one of them and see the result again

```
transaction[BRAND == "Red", BRAND := "RRD"]
```

```r
transBrand <- transaction[, .N, BRAND][order(-N)]
transBrand
```

```
##          BRAND     N
##         <char> <int>
##  1:     Kettle 41288
##  2:     Smiths 27390
##  3:   Pringles 25102
##  4:    Doritos 22041
##  5:        RRD 16321
##  6:      Thins 14075
##  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:     Dorito  3183
## 18:     Infzns  3144
## 19:      Smith  2963
## 20:    Cheetos  2927
## 21:      Snbts  1576
## 22:     Burger  1564
## 23: Woolworths  1516
## 24:    GrnWves  1468
## 25:   Sunbites  1432
## 26:        NCC  1419
## 27:     French  1418
##          BRAND     N
```

We can see that Kettle is a top selling brand with 41288 transactions.

```r
transaction
```

```
##               DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##             <Date>     <num>          <num>  <num>    <num>
##      1: 2018-09-27         1           1000      1        5
##      2: 2019-04-24         1           1307    348       66
##      3: 2019-04-30         1           1343    383       61
##      4: 2018-07-28         2           2373    974       69
##      5: 2018-07-29         2           2426   1038      108
##     ---
## 246736: 2019-02-17       272         272319 270088       89
## 246737: 2018-07-24       272         272358 270154       74
## 246738: 2018-10-17       272         272379 270187       51
## 246739: 2018-12-07       272         272379 270188       42
## 246740: 2018-09-02       272         272380 270189       74
```

```
##                                       PROD_NAME PROD_QTY TOT_SALES
PACK_SIZE
##                                          <char>    <num>     <num>
<num>
##      1:   Natural Chip        Compny SeaSalt175g        2       6.0
175
##      2:                 CCs Nacho Cheese    175g        3       6.3
175
##      3:   Smiths Crinkle Cut  Chips Chicken 170g        2       2.9
170
##      4:   Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0
175
##      5: Kettle Tortilla ChpsHny&Jlpno Chili 150g        3      13.8
150
##     ---
## 246736:  Kettle Sweet Chilli And Sour Cream 175g        2      10.8
175
## 246737:            Tostitos Splash Of  Lime 175g        1       4.4
175
## 246738:                 Doritos Mexicana    170g        2       8.8
170
## 246739:  Doritos Corn Chip Mexican Jalapeno 150g        2       7.8
150
## 246740:            Tostitos Splash Of  Lime 175g        2       8.8
175
##            BRAND
##           <char>
##      1:  Natural
##      2:      CCs
##      3:   Smiths
##      4:   Smiths
##      5:   Kettle
##     ---
## 246736:   Kettle
## 246737: Tostitos
## 246738:  Doritos
## 246739:  Doritos
## 246740: Tostitos
```

Now that we are happy with our transaction data. We will dip dive into the customer data.

```
#Lets check the structure of customer data
str(customer)
```

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

```r
#Lets see first 5 rows
head(customer)
```

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

```r
#Lets convert customer data frame into data tables
customer <- data.table(customer)
```

Lets see the frequency of category of PREMIUM_CUSTOMER

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

```
##    PREMIUM_CUSTOMER     N
##              <char> <int>
## 1:       Mainstream 29245
## 2:           Budget 24470
## 3:          Premium 18922
```

Similarly, lets see the frequency of LIFESTAGE of customer

```r
customer[, .N, LIFESTAGE][order(-N)]
```

```
##                 LIFESTAGE     N
##                    <char> <int>
## 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
#Dimension of customer data
dim(customer)
```

```
## [1] 72637     3
```

Since LYLTY_CARD_NBR is our unique value in table. We will see how much unique values are there in both transaction and customer table

```r
#Unique LYLTY_CARD_NBR in transaction
count(distinct(transaction, LYLTY_CARD_NBR))
```

```
##        n
##    <int>
## 1: 71287
```

```
#Unique LYLTY_CARD_NBR in transaction
count(distinct(customer, LYLTY_CARD_NBR))

##         n
##     <int>
## 1: 72637
```

We will merge our trnsaction data with customer data for further analysis.

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

head(data)

## Key: <LYLTY_CARD_NBR>
##    LYLTY_CARD_NBR       DATE STORE_NBR TXN_ID PROD_NBR
##             <int>     <Date>     <num>  <num>    <num>
## 1:           1000 2018-09-27         1      1        5
## 2:           1002 2018-08-27         1      2       58
## 3:           1003 2019-02-15         1      3       52
## 4:           1003 2019-02-16         1      4      106
## 5:           1004 2018-10-13         1      5       96
## 6:           1005 2018-12-08         1      6       86
##                                 PROD_NAME PROD_QTY TOT_SALES PACK_SIZE
BRAND
##                                    <char>    <num>     <num>     <num>
<char>
## 1: Natural Chip        Compny SeaSalt175g        2       6.0       175
Natural
## 2:  Red Rock Deli Chikn&Garlic Aioli 150g        1       2.7       150
RRD
## 3:  Grain Waves Sour    Cream&Chives 210G        1       3.6       210
Grain
## 4: Natural ChipCo      Hony Soy Chckn175g        1       3.0       175
Natural
## 5:         WW Original Stacked Chips 160g        1       1.9       160
WW
## 6:                     Cheetos Puffs 165g        1       2.8       165
Cheetos
##                 LIFESTAGE PREMIUM_CUSTOMER
##                    <char>           <char>
## 1:  YOUNG SINGLES/COUPLES          Premium
## 2:  YOUNG SINGLES/COUPLES       Mainstream
## 3:         YOUNG FAMILIES           Budget
## 4:         YOUNG FAMILIES           Budget
## 5:  OLDER SINGLES/COUPLES       Mainstream
## 6: MIDAGE SINGLES/COUPLES       Mainstream
```

Lets check for any na values in data set

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

```
## [1] 0

#Lets check for any duplicates values in the data set
sum(duplicated(data))

## [1] 1
```

We can see that we have one duplicate row in our data. Lets find out the values.

```
which(duplicated(data))

## [1] 99028

#Converting our data into data tables
data <- data.table(data)
```

Lets see the duplicated row

```
data[99028,]

## Key: <LYLTY_CARD_NBR>
##    LYLTY_CARD_NBR       DATE STORE_NBR TXN_ID PROD_NBR
##             <int>     <Date>     <num>  <num>    <num>
## 1:         107024 2018-09-11       107 108462       45
##                                PROD_NAME PROD_QTY TOT_SALES PACK_SIZE
BRAND
##                                   <char>    <num>     <num>     <num>
<char>
## 1: Smiths Thinly Cut   Roast Chicken 175g        2         6       175
Smiths
##                LIFESTAGE PREMIUM_CUSTOMER
##                   <char>           <char>
## 1: OLDER SINGLES/COUPLES          Premium
```

Lets see all duplicated values with following LYLTY_CARD_NBR

```
data[duplicated(data) | duplicated(data, fromLast = TRUE)]

## Key: <LYLTY_CARD_NBR>
##    LYLTY_CARD_NBR       DATE STORE_NBR TXN_ID PROD_NBR
##             <int>     <Date>     <num>  <num>    <num>
## 1:         107024 2018-09-11       107 108462       45
## 2:         107024 2018-09-11       107 108462       45
##                                PROD_NAME PROD_QTY TOT_SALES PACK_SIZE
BRAND
##                                   <char>    <num>     <num>     <num>
<char>
## 1: Smiths Thinly Cut   Roast Chicken 175g        2         6       175
Smiths
## 2: Smiths Thinly Cut   Roast Chicken 175g        2         6       175
Smiths
##                LIFESTAGE PREMIUM_CUSTOMER
```

```
##                      <char>           <char>
## 1: OLDER SINGLES/COUPLES          Premium
## 2: OLDER SINGLES/COUPLES          Premium
```

We have identified that the above entries are duplicate values. Hence we will remove one of them.

```
data <- distinct(data)
```

Lets verify our result

```
which(duplicated(data))

## integer(0)
```

Now that we have completed our data exploration part, lets save this data as CSV.

```
write.csv(data, file = "QVI_data.csv", row.names = FALSE)
```

Analysis on Customer Segment

Total Sales by Lifestage and Premium Customer

```
#Grouping customer by Lifestage and Premium Customer
grouped_Cus <- data %>% group_by(LIFESTAGE,PREMIUM_CUSTOMER)
grouped_Cus

## # A tibble: 246,739 × 12
## # Groups:   LIFESTAGE, PREMIUM_CUSTOMER [21]
##    LYLTY_CARD_NBR DATE       STORE_NBR TXN_ID PROD_NBR PROD_NAME
PROD_QTY
##             <int> <date>         <dbl>  <dbl>    <dbl> <chr>
<dbl>
##  1           1000 2018-09-27         1      1        5 Natural Chip   …
2
##  2           1002 2018-08-27         1      2       58 Red Rock Deli C…
1
##  3           1003 2019-02-15         1      3       52 Grain Waves Sou…
1
##  4           1003 2019-02-16         1      4      106 Natural ChipCo …
1
##  5           1004 2018-10-13         1      5       96 WW Original Sta…
1
##  6           1005 2018-12-08         1      6       86 Cheetos Puffs 1…
1
##  7           1007 2018-11-14         1      7       49 Infuzions SourC…
1
##  8           1007 2018-11-15         1      8       10 RRD SR Slow Rst…
1
##  9           1009 2018-10-31         1      9       20 Doritos Cheese …
1
## 10           1010 2018-08-20         1     10       51 Doritos Mexican…
```

```
2
## # ℹ 246,729 more rows
## # ℹ 5 more variables: TOT_SALES <dbl>, PACK_SIZE <dbl>, BRAND <chr>,
## #   LIFESTAGE <chr>, PREMIUM_CUSTOMER <chr>
```

Now lets calculate the total sales for each customer segment

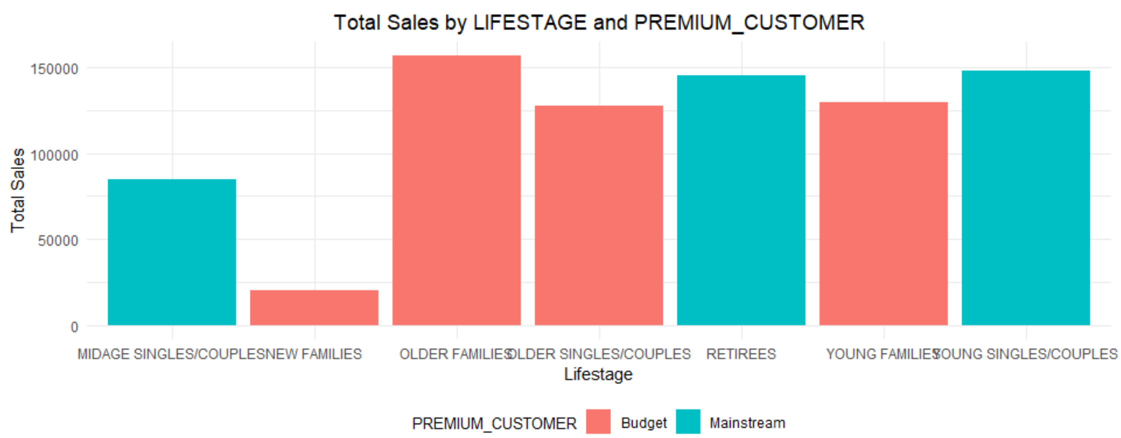
```
#Total sales per segement and arranging in descending order
grouped_Sales <- grouped_Cus %>% summarise(Total_Sales = sum(TOT_SALES)) %>%
arrange(desc(Total_Sales))

## `summarise()` has grouped output by 'LIFESTAGE'. You can override using
the
## `.groups` argument.

grouped_Sales

## # A tibble: 21 × 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER Total_Sales
##    <chr>                  <chr>                  <dbl>
##  1 OLDER FAMILIES         Budget               156864.
##  2 YOUNG SINGLES/COUPLES  Mainstream           147582.
##  3 RETIREES               Mainstream           145169.
##  4 YOUNG FAMILIES         Budget               129718.
##  5 OLDER SINGLES/COUPLES  Budget               127834.
##  6 OLDER SINGLES/COUPLES  Mainstream           124648.
##  7 OLDER SINGLES/COUPLES  Premium              123532.
##  8 RETIREES               Budget               105916.
##  9 OLDER FAMILIES         Mainstream            96414.
## 10 RETIREES               Premium               91297.
## # ℹ 11 more rows
```

Now that we have identified total sales for each customer segments. Lets find out the top spending group

```
high_Spd_Grp <- grouped_Sales %>% slice(1) %>% arrange(desc(Total_Sales))
high_Spd_Grp

## # A tibble: 7 × 3
## # Groups:   LIFESTAGE [7]
##   LIFESTAGE              PREMIUM_CUSTOMER Total_Sales
##   <chr>                  <chr>                  <dbl>
## 1 OLDER FAMILIES         Budget               156864.
## 2 YOUNG SINGLES/COUPLES  Mainstream           147582.
## 3 RETIREES               Mainstream           145169.
## 4 YOUNG FAMILIES         Budget               129718.
## 5 OLDER SINGLES/COUPLES  Budget               127834.
## 6 MIDAGE SINGLES/COUPLES Mainstream            84734.
## 7 NEW FAMILIES           Budget                20607.
```

Lets see these result in visualization

```r
ggplot(high_Spd_Grp, aes(x = LIFESTAGE, y = Total_Sales, fill =
PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(
    title = "Total Sales by LIFESTAGE and PREMIUM_CUSTOMER",
    x = "Lifestage",
    y = "Total Sales"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5),
    legend.title = element_text(size = 10),
    legend.position = "bottom")
```



We can see that that the highest spending customer segment are Older Families- Budget, Young Singles/Couples - Mainstream, and Retirees - Mainstream.

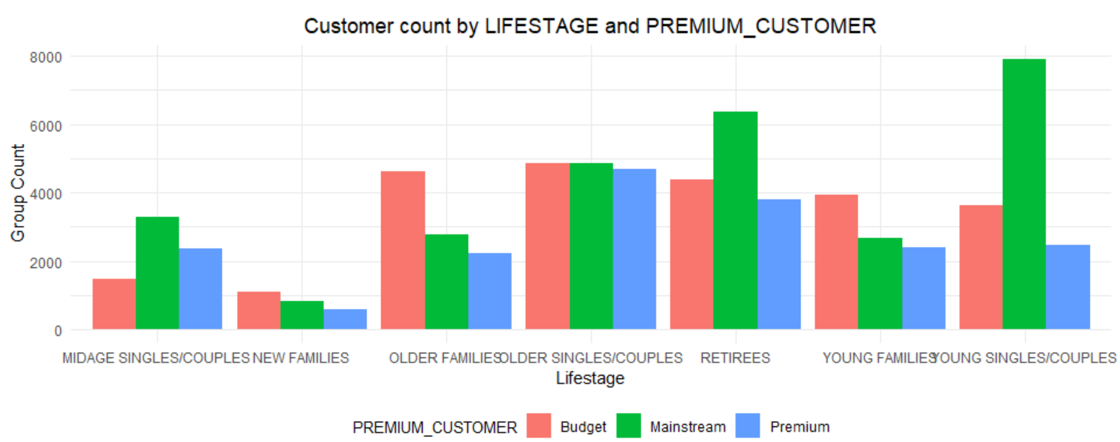
Total Customer in each segments

```r
#Number of Customer by segment
grouped_Count <- grouped_Cus %>% summarise(Grp_Count =
n_distinct(LYLTY_CARD_NBR)) %>% arrange(desc(Grp_Count))

## `summarise()` has grouped output by 'LIFESTAGE'. You can override using
the
## `.groups` argument.

grouped_Count

## # A tibble: 21 × 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE             PREMIUM_CUSTOMER Grp_Count
##    <chr>                 <chr>                <int>
##  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
## # ℹ 11 more rows
```

Lets plot our result

```
ggplot(grouped_Count, aes(x = LIFESTAGE, y = Grp_Count, fill =
PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(
    title = "Customer count by LIFESTAGE and PREMIUM_CUSTOMER",
    x = "Lifestage",
    y = "Group Count"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5),
    legend.title = element_text(size = 10),
    legend.position = "bottom")
```



There are more Mainstream customers in both Retirees and Single/Couples segments.

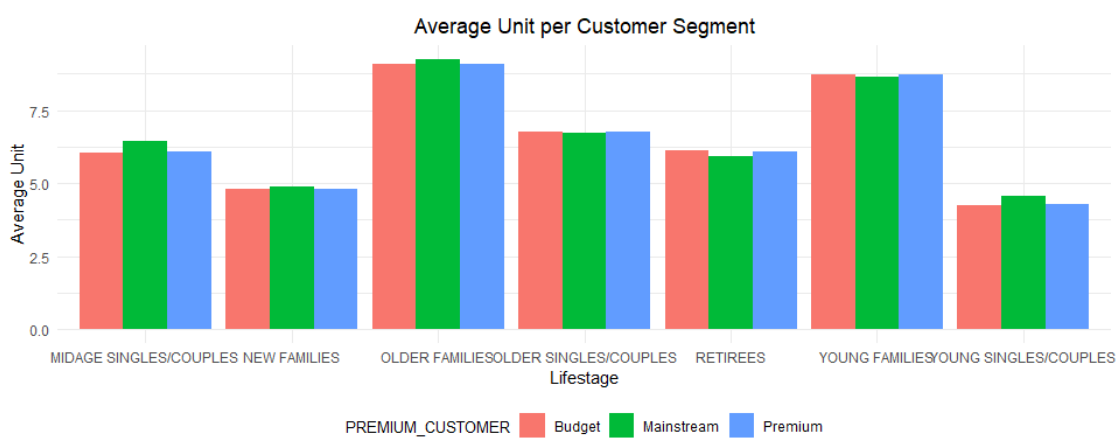
Number of Chips bought per customer by segment

```
#Average unit per customer
Avg_Unit_per_Cus <- grouped_Cus %>% summarise(Total_Units = sum(PROD_QTY),
Unique_Cus =  n_distinct(LYLTY_CARD_NBR),
Avg_Unit = Total_Units / Unique_Cus
) %>% arrange(desc(Avg_Unit))

## `summarise()` has grouped output by 'LIFESTAGE'. You can override using
the
## `.groups` argument.
```

```
Avg_Unit_per_Cus

## # A tibble: 21 × 5
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER Total_Units Unique_Cus Avg_Unit
##    <chr>                  <chr>                  <dbl>      <int>    <dbl>
##  1 OLDER FAMILIES         Mainstream             25804       2788     9.26
##  2 OLDER FAMILIES         Budget                 41853       4611     9.08
##  3 OLDER FAMILIES         Premium                20239       2231     9.07
##  4 YOUNG FAMILIES         Budget                 34482       3953     8.72
##  5 YOUNG FAMILIES         Premium                20901       2398     8.72
##  6 YOUNG FAMILIES         Mainstream             23194       2685     8.64
##  7 OLDER SINGLES/COUPLES  Budget                 32883       4849     6.78
##  8 OLDER SINGLES/COUPLES  Premium                31693       4682     6.77
##  9 OLDER SINGLES/COUPLES  Mainstream             32607       4858     6.71
## 10 MIDAGE SINGLES/COUPLES Mainstream             21213       3298     6.43
## # ℹ 11 more rows
```

Lets plot our result

```
#Average Unit per Customer
ggplot(Avg_Unit_per_Cus, aes(x = LIFESTAGE, y = Avg_Unit, fill =
PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(
    title = "Average Unit per Customer Segment",
    x = "Lifestage",
    y = "Average Unit"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5),
    legend.title = element_text(size = 10),
    legend.position = "bottom")
```



The above result shows that Older Families and Young Families buy more chips per customer in these customer segements.

Average price per unit by Customer Segments

```
#Average price per unit by customer segement
Avg_Price_per_Cus <- grouped_Cus %>% summarise(Total_Units = sum(PROD_QTY),
Total_Sales = sum(TOT_SALES),
Avg_Price = Total_Sales/ Total_Units
) %>% arrange(desc(Avg_Price))
```

```
## `summarise()` has grouped output by 'LIFESTAGE'. You can override using
the
## `.groups` argument.
```

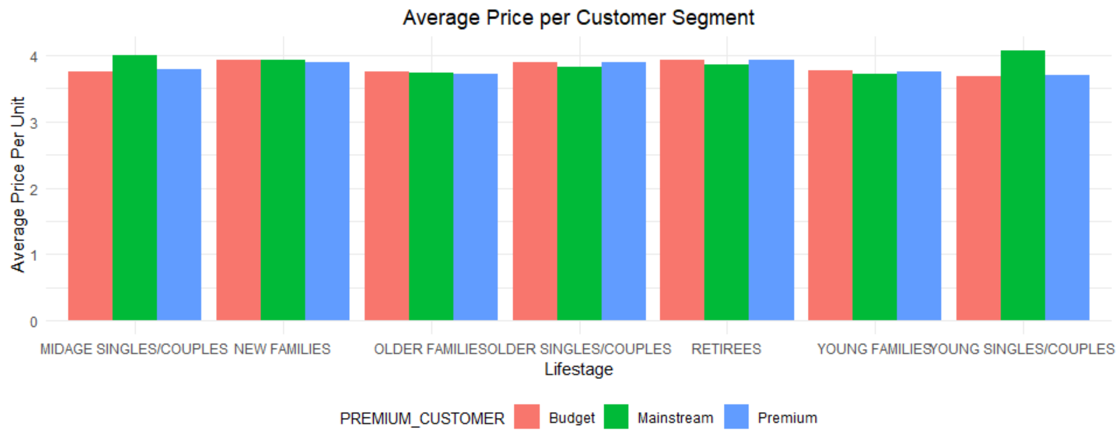
```
Avg_Price_per_Cus
```

```
## # A tibble: 21 × 5
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER Total_Units Total_Sales
Avg_Price
##    <chr>                  <chr>                  <dbl>       <dbl>
<dbl>
##  1 YOUNG SINGLES/COUPLES  Mainstream             36225     147582.
4.07
##  2 MIDAGE SINGLES/COUPLES Mainstream             21213      84734.
3.99
##  3 NEW FAMILIES           Mainstream              4060      15980.
3.94
##  4 RETIREES               Budget                 26932     105916.
3.93
##  5 NEW FAMILIES           Budget                  5241      20607.
3.93
##  6 RETIREES               Premium                23266      91297.
3.92
##  7 OLDER SINGLES/COUPLES  Premium                31693     123532.
3.90
##  8 OLDER SINGLES/COUPLES  Budget                 32883     127834.
3.89
##  9 NEW FAMILIES           Premium                 2769      10761.
3.89
## 10 RETIREES               Mainstream             37677     145169.
3.85
## # ℹ 11 more rows
```

Lets Plot this result

```
#Average price per unit
ggplot(Avg_Price_per_Cus, aes(x = LIFESTAGE, y = Avg_Price, fill =
PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(
    title = "Average Price per Customer Segment",
    x = "Lifestage",
```

```
    y = "Average Price Per Unit"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5),
    legend.title = element_text(size = 10),
    legend.position = "bottom")
```



From the above result we can see that the average price per customer segment is almost similar for each segment. But Mainstream- Midage and Young singles and couples spends more on buying the chips than Budget and Premium counter part.

So, lets figure this out through t-test method.

Before moving into the t-test, lets calculate the mean of Lifestage Midage Single/Couple for both Mainstream, and Premium and Budget Segment

```
#Average_Price for midage- mainstream customers

midage_mainstream <- Avg_Price_per_Cus %>% filter(LIFESTAGE == "MIDAGE
SINGLES/COUPLES" & PREMIUM_CUSTOMER == "Mainstream") %>% pull(Avg_Price)
midage_mainstream

## [1] 3.994449

#Average Price for Midage- Premium, Budget customers
midage_premium_budget <- Avg_Price_per_Cus %>% filter(LIFESTAGE == "MIDAGE
SINGLES/COUPLES" & PREMIUM_CUSTOMER %in% c("Premium","Budget")) %>%
pull(Avg_Price)
midage_premium_budget

## [1] 3.780823 3.753878
```

Lets perform t-test

```
t_test_midage <- t.test(midage_mainstream,midage_premium_budget, var.equal =
TRUE)
print(t_test_midage)
```

```
## 
##  Two Sample t-test
## 
## data:  midage_mainstream and midage_premium_budget
## t = 9.7322, df = 1, p-value = 0.06519
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.06939832  0.52359553
## sample estimates:
## mean of x mean of y 
##  3.994449  3.767351
```

From the above t-test the p-value is 0.06519 which is greater than 0.05, so we fail to reject null hypothesis and we don't have enough evidence to conclude a significant mean difference between Midage - Mainstream and Midage - Premium_budget segment. Also, the confidence interval contain 0 which further support the difference in mean is not statistically significant.

In simple term, there is no statistical evidence of Mainstream - Midage ,and Young singles and couples spending more on buying the chips than Budget and Premium counter part.

t-test for Young- Mainstream , and Premium, Budget Customers

```r
#Average_Price for young- mainstream customers

young_mainstream <- Avg_Price_per_Cus %>% filter(LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER == "Mainstream") %>% pull(Avg_Price)
young_mainstream

## [1] 4.074043

#Average_Price for young- premium and budget customers

young_premium_budget <- Avg_Price_per_Cus %>% filter(LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER %in% c("Premium", "Budget")) %>% pull(Avg_Price)
young_premium_budget

## [1] 3.692889 3.685297
```

Lets perform t-test on above mean

```r
t_test_young <- t.test(young_mainstream, young_premium_budget, var.equal = TRUE)
print(t_test_young)

## 
##  Two Sample t-test
## 
## data:  young_mainstream and young_premium_budget
## t = 58.548, df = 1, p-value = 0.01087
```

```
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.3014071 0.4684928
## sample estimates:
## mean of x mean of y
##  4.074043  3.689093
```

From the above result we can see that the p-value is 0.01087 which is less than 0.05, hence we reject the null hypothesis. Also, the confidence interval doesn't contain 0 which further supports this claim.

In simple term, Young- Mainstream customers segments spend more on buying chips than Young- Premium and Budget counter part.

Now lets deep dive into young- mainstream customer segment.

Lets see if this customer segment has any preference for specific brands.

```
#Filter data for Young- Mainstream customer segments
filtered_data <- data %>%
  filter(LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER ==
"Mainstream")
filtered_data

## Key: <LYLTY_CARD_NBR>
##        LYLTY_CARD_NBR       DATE STORE_NBR TXN_ID PROD_NBR
##                 <int>     <Date>     <num>  <num>    <num>
##     1:           1002 2018-08-27         1      2       58
##     2:           1010 2018-08-20         1     10       51
##     3:           1018 2018-08-14         1     22        3
##     4:           1018 2018-11-08         1     23       97
##     5:           1018 2019-05-31         1     24       38
##    ---
## 19540:         272391 2018-11-17       272 270205       63
## 19541:        2330041 2018-09-03        77 236718       24
## 19542:        2330321 2018-07-10        77 236756       71
## 19543:        2370181 2018-07-13        88 240146       36
## 19544:        2373711 2018-11-24        88 241815       16
##                                      PROD_NAME PROD_QTY TOT_SALES
PACK_SIZE
##                                         <char>    <num>     <num>
<num>
##     1:    Red Rock Deli Chikn&Garlic Aioli 150g        1       2.7
150
##     2:                 Doritos Mexicana    170g        2       8.8
170
##     3: Kettle Sensations   Camembert & Fig 150g        1       4.6
150
##     4:                 RRD Salt & Vinegar  165g        1       3.0
165
##     5:  Infuzions Mango     Chutny Papadums 70g        1       2.4
```

```
70
##    ---
## 19540:             Kettle 135g Swt Pot Sea Salt        2       8.4
135
## 19541:    Grain Waves         Sweet Chilli 210g        2       7.2
210
## 19542:          Twisties Cheese     Burger 250g        2       8.6
250
## 19543:                      Kettle Chilli 175g        2      10.8
175
## 19544: Smiths Crinkle Chips Salt & Vinegar 330g        2      11.4
330
##            BRAND             LIFESTAGE PREMIUM_CUSTOMER
##           <char>                <char>           <char>
##     1:       RRD YOUNG SINGLES/COUPLES       Mainstream
##     2:   Doritos YOUNG SINGLES/COUPLES       Mainstream
##     3:    Kettle YOUNG SINGLES/COUPLES       Mainstream
##     4:       RRD YOUNG SINGLES/COUPLES       Mainstream
##     5: Infuzions YOUNG SINGLES/COUPLES       Mainstream
##    ---
## 19540:    Kettle YOUNG SINGLES/COUPLES       Mainstream
## 19541:     Grain YOUNG SINGLES/COUPLES       Mainstream
## 19542:  Twisties YOUNG SINGLES/COUPLES       Mainstream
## 19543:    Kettle YOUNG SINGLES/COUPLES       Mainstream
## 19544:    Smiths YOUNG SINGLES/COUPLES       Mainstream
```

install.packages("arules") install.packages("arulesViz")

```
library(arules)

## Warning: package 'arules' was built under R version 4.3.3

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

##
## Attaching package: 'arules'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```r
library(arulesViz)

## Warning: package 'arulesViz' was built under R version 4.3.3
```

First lets convert our data for Young-Mainstream customer segment into transactions

```r
#Converting data into tansaction for market basket analysis
brand_trans <- as(split(filtered_data$BRAND, filtered_data$LYLTY_CARD_NBR),
"transactions")

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

brand_trans

## transactions in sparse format with
##  7917 transactions (rows) and
##  27 items (columns)

#Checking first 5 transactions
head(as(brand_trans, "list"), 5)

## $`1002`
## [1] "RRD"
##
## $`1010`
## [1] "Doritos"
##
## $`1018`
## [1] "Infuzions" "Kettle"    "RRD"
##
## $`1020`
## [1] "GrnWves" "Smiths"
##
## $`1060`
## [1] "Doritos"  "Twisties" "Tyrrells"

#Lets apply apriori algorithm in our brand transaction
rules <- apriori(brand_trans, parameter = list(supp = 0.001, conf = 0.3,
target = "rules"))

## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.3    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
```

```
## Absolute minimum support count: 7
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[27 item(s), 7917 transaction(s)] done [0.00s].
## sorting and recoding items ... [27 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 done [0.00s].
## writing ... [1156 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
# Lets inspect top 10 rules by lift
inspect(sort(rules, by = "lift")[1:10])
```

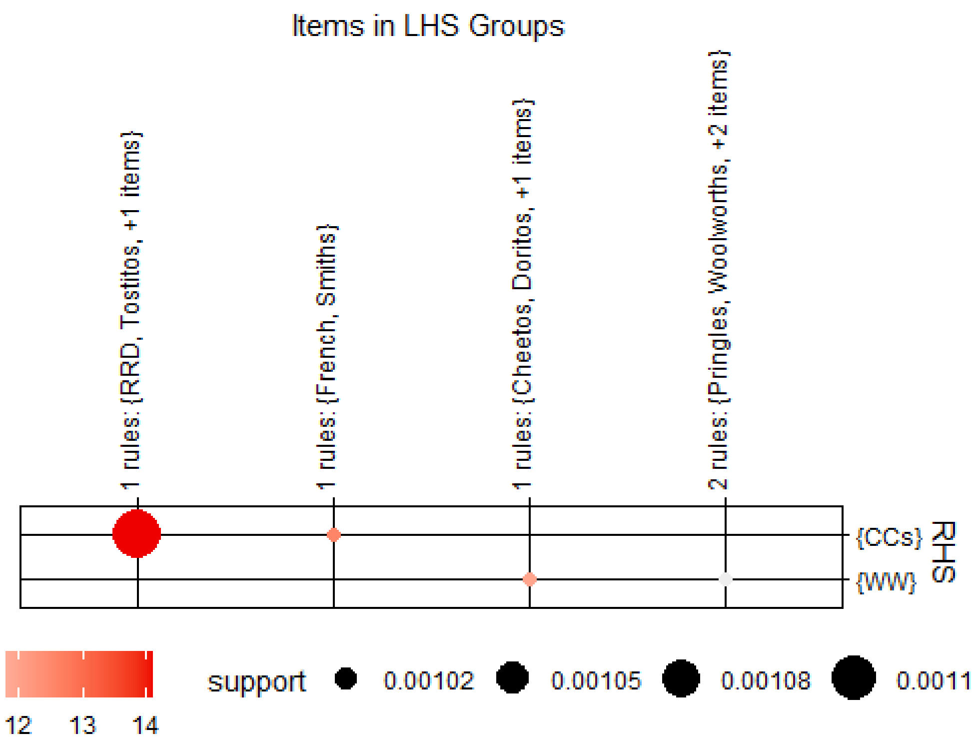
```
##      lhs                               rhs       support     confidence
## [1]  {Doritos, RRD, Tostitos}    => {CCs}     0.001136794 0.3750000
## [2]  {French, Smiths}            => {CCs}     0.001010484 0.3333333
## [3]  {Cheetos, Doritos, Smiths}  => {WW}      0.001010484 0.5714286
## [4]  {Pringles, Woolworths}      => {WW}      0.001010484 0.5000000
## [5]  {Cheetos, Pringles, Smiths} => {WW}      0.001010484 0.5000000
## [6]  {Smiths, Woolworths}        => {WW}      0.001136794 0.4285714
## [7]  {Kettle, RRD, Smiths, WW}   => {Natural} 0.001136794 0.3214286
## [8]  {CCs, Doritos, Tostitos}    => {RRD}     0.001136794 0.7500000
## [9]  {CCs, Infuzions, Smiths}    => {RRD}     0.001515726 0.7500000
## [10] {CCs, Doritos, Smiths}      => {WW}      0.001010484 0.3809524
##      coverage    lift      count
## [1]  0.003031451 14.070498  9
## [2]  0.003031451 12.507109  8
## [3]  0.001768347 11.874016  8
## [4]  0.002020968 10.389764  8
## [5]  0.002020968 10.389764  8
## [6]  0.002652520  8.905512  9
## [7]  0.003536693  8.370888  9
## [8]  0.001515726  7.980847  9
## [9]  0.002020968  7.980847 12
## [10] 0.002652520  7.916010  8
```

Lets select rules with some significant brands

```
top_rules <- subset(rules, lift >10)

inspect(sort(top_rules, by = "lift"))
```

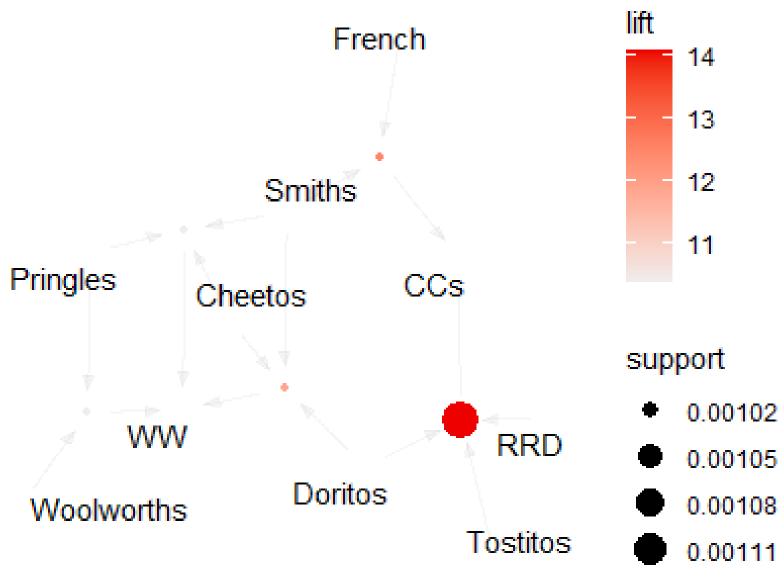
```
##     lhs                              rhs   support     confidence coverage
## [1] {Doritos, RRD, Tostitos}    => {CCs} 0.001136794 0.3750000
0.003031451
## [2] {French, Smiths}            => {CCs} 0.001010484 0.3333333
0.003031451
## [3] {Cheetos, Doritos, Smiths}  => {WW}  0.001010484 0.5714286
0.001768347
## [4] {Pringles, Woolworths}      => {WW}  0.001010484 0.5000000
0.002020968
```

```
## [5] {Cheetos, Pringles, Smiths} => {WW}  0.001010484 0.5000000
0.002020968
##     lift     count
## [1] 14.07050 9
## [2] 12.50711 8
## [3] 11.87402 8
## [4] 10.38976 8
## [5] 10.38976 8

# Visualize rules
plot(top_rules, method = "grouped")
```




```
# Example: Graph-based Visualization
plot(top_rules, method = "graph", control = list(type = "items"))

## Warning: Unknown control parameters: type

## Available control parameters (with default values):
## layout    =  stress
## circular  =  FALSE
## ggraphdots    =  NULL
## edges     =  <environment>
## nodes     =  <environment>
## nodetext  =  <environment>
## colors    =  c("#EE0000FF", "#EEEEEEFF")
## engine    =  ggplot2
```

```
## max    =  100
## verbose    =  FALSE
```




```
inspect(sort(rules, by = "count")[1:10])

##      lhs              rhs      support    confidence coverage   lift
count
## [1]  {}            => {Kettle} 0.38714159 0.3871416  1.00000000 1.0000000
3065
## [2]  {Pringles}    => {Kettle} 0.09144878 0.3570020  0.25615764 0.9221483
724
## [3]  {Doritos}     => {Kettle} 0.07907035 0.3435785  0.23013768 0.8874750
626
## [4]  {Smiths}      => {Kettle} 0.07212328 0.3759052  0.19186561 0.9709760
571
## [5]  {Thins}       => {Kettle} 0.05128205 0.3769731  0.13603638 0.9737344
406
## [6]  {Infuzions} => {Kettle} 0.03953518 0.3524775  0.11216370 0.9104614
313
## [7]  {Tostitos}    => {Kettle} 0.03764052 0.3556086  0.10584817 0.9185492
298
## [8]  {Cobs}        => {Kettle} 0.03726159 0.3588808  0.10382721 0.9270013
295
## [9]  {Twisties}    => {Kettle} 0.03625111 0.3376471  0.10736390 0.8721539
287
## [10] {RRD}         => {Kettle} 0.03410383 0.3629032  0.09397499 0.9373915
270
```

Analysis From our analysis we can see that rule 1 {Doritos, RRD,Tostitos} => {CCs} has highest lift of 14.07 meaning these brands have strong association or if customer bought Doritos, RDD and Tostitos they are more likely to buy CCs.

For this customer segment Pringles and Kettle are the most preferred brand which appeared in 724 transactions.

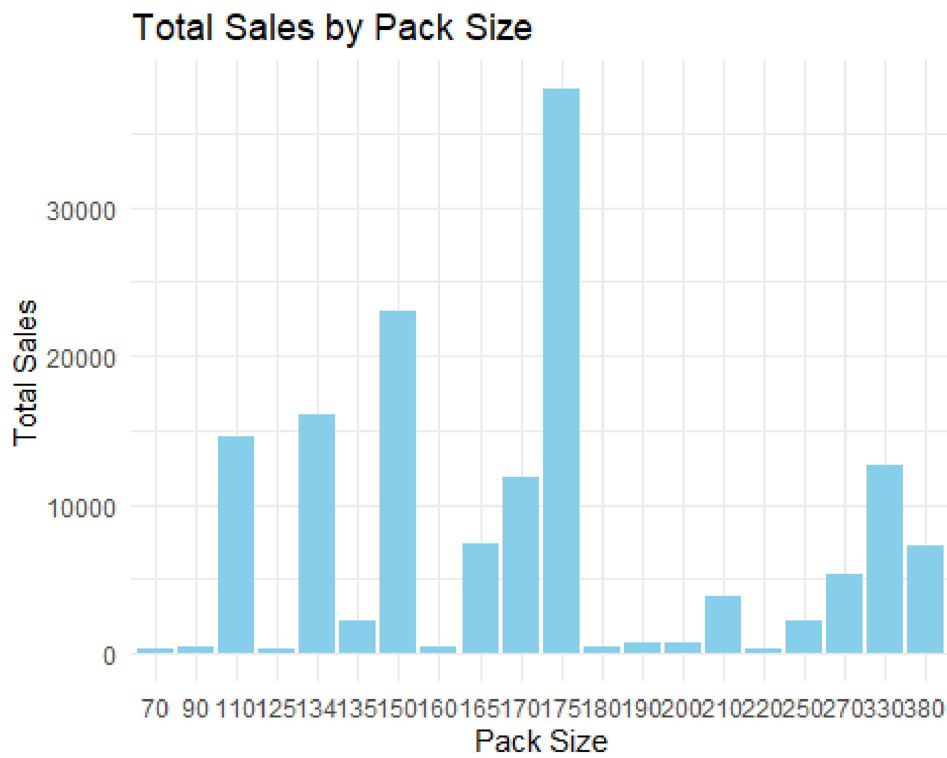
Chips Size Analysis First lets group our data according to pack size and calculate the total sales and average quantity for each pack size

```
packsize_sales <- filtered_data %>%
  group_by(PACK_SIZE) %>%
  summarise(
    Total_Sales = sum(TOT_SALES, na.rm = TRUE),
    Avg_Quantity = mean(PROD_QTY, na.rm = TRUE),
    Transaction_Count = n()
  )
packsize_sales
```

```
## # A tibble: 20 × 4
##    PACK_SIZE Total_Sales Avg_Quantity Transaction_Count
##        <dbl>       <dbl>        <dbl>             <int>
##  1        70         264         1.75                63
##  2        90         391         1.80               128
##  3       110       14630         1.88              2051
##  4       125         229.        1.85                59
##  5       134       16006.        1.87              2315
##  6       135        2247         1.84               290
##  7       150       22946.        1.85              3080
##  8       160         441.        1.81               128
##  9       165        7395         1.83              1102
## 10       170       11893.        1.86              1575
## 11       175       37968.        1.85              4997
## 12       180         403         1.86                70
## 13       190         740.        1.83               148
## 14       200         618.        1.82               179
## 15       210        3798         1.83               576
## 16       220         244.        1.71                62
## 17       250        2236         1.86               280
## 18       270        5304.        1.86               620
## 19       330       12654         1.86              1195
## 20       380        7176.        1.86               626
```

Lets visualize this result

```
#Total sales by pack size
ggplot(packsize_sales, aes(x = factor(PACK_SIZE), y = Total_Sales)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Total Sales by Pack Size", x = "Pack Size", y = "Total
Sales") +
  theme_minimal()
```



From the above analysis we can see that for Young-Mainstream customers pack size of 175g is mostly preferred. Also, they buy smaller pack size more often rather than bigger ones.