Data Preparation and Customer Analytics Task

[Quantium Virtual Internship Task 1]

Cwen Fernandes

- **Loading required libraries and datasets**

```
install.packages("data.table")
#### Loading required libraries
library(data.table)
library(ggplot2)
library(ggmosaic)
library(readr)

#### Loading chips transaction data
transaction_data <- read.csv("R/transaction_data.csv", header=TRUE)
head(transaction_data)
```

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

```
#### Loading Customer Purchase data
pb = read.csv("R/PurchaseBehaviour.csv", header=TRUE)
head(pb)
```

```
  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
>
```

➢ Exploratory Data Analysis

First step in analysing the data is to understand the data.

▪ Examining transaction data:

We use str() to analyze every column in our dataset

str(transaction_data)

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

##Convert DATE column to a date format

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

head(transaction_data$DATE)

```
> head(transaction_data$DATE)
[1] "2018-10-17" "2019-05-14" "2019-05-20" "2018-08-17" "2018-08-18" "2019-05-19"
> 
```

head(transaction_data)

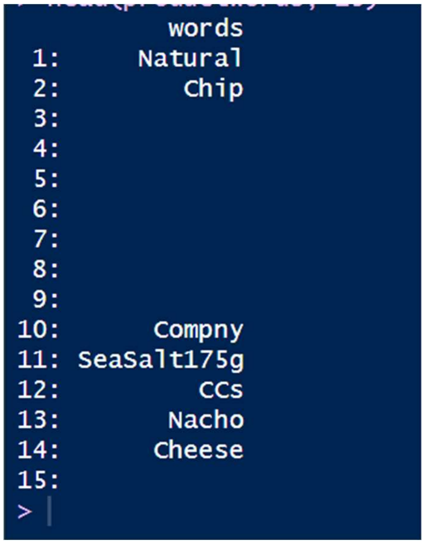
```
> head(transaction_data)
        DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
1 2018-10-17         1           1000      1        5
2 2019-05-14         1           1307    348       66
3 2019-05-20         1           1343    383       61
4 2018-08-17         2           2373    974       69
5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

##Examine PROD_NAME

summary(transaction_data$PROD_NAME)

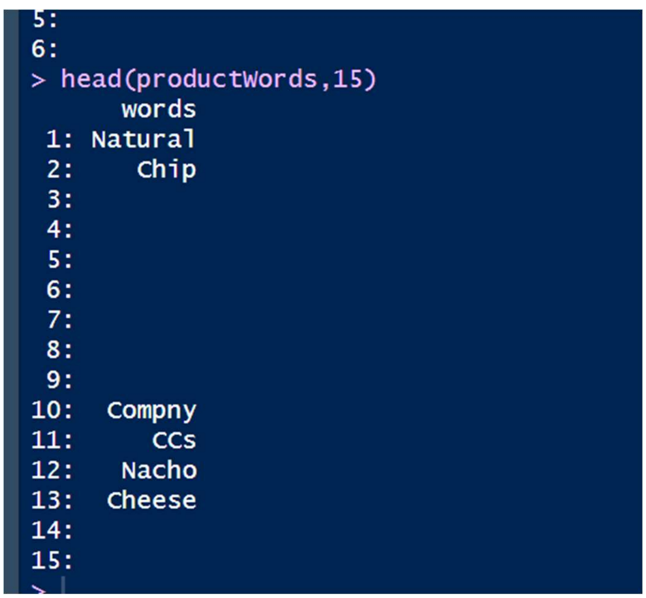
```
> summary(transaction_data$PROD_NAME)
   Length     Class      Mode 
   264836 character character 
> 
```

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



```
          words
 1:     Natural
 2:        Chip
 3:
 4:
 5:
 6:
 7:
 8:
 9:
10:      Compny
11: SeaSalt175g
12:         CCs
13:       Nacho
14:      Cheese
15:
>
```

##Removing digits
productWords[, SPECIAL := grepl("[[:digit:]]", words)]
productWords <- productWords[SPECIAL == FALSE,] [,SPECIAL := NULL]



```
 5:
 6:
> head(productWords,15)
      words
 1: Natural
 2:    Chip
 3:
 4:
 5:
 6:
 7:
 8:
 9:
10:  Compny
11:     CCs
12:   Nacho
13:  Cheese
14:
15:
```

##Removing Special Characters
##Removing punctuation
productWords[,SPECIAL := grepl("[[:punct:]]", words)]
productWords <- productWords[SPECIAL == FALSE,] [,SPECIAL := NULL]

##changing empty strings to NA
productWords[words == ""] <- NA

```
> head(productWords, 15)
      words
 1: Natural
 2:    Chip
 3:    <NA>
 4:    <NA>
 5:    <NA>
 6:    <NA>
 7:    <NA>
 8:    <NA>
 9:    <NA>
10:  Compny
11:     CCs
12:   Nacho
13:  Cheese
14:    <NA>
15:    <NA>
>
```

##removing all empty cells
productWords <- productWords[complete.cases(productWords),]

```
> productWords <- productWords[comple
> head(productWords,15)
      words
 1: Natural
 2:    Chip
 3:  Compny
 4:     CCs
 5:   Nacho
 6:  Cheese
 7:  Smiths
 8: Crinkle
 9:     Cut
10:   Chips
11: Chicken
12:  Smiths
13:    Chip
14:  Thinly
15:  Kettle
>
```

##creating a frequency table for out set of words, sorted
productWords <- data.frame(sort(table(productWords), decreasing = TRUE))

```
> head(productWords)
    words Freq
1   Chips   21
2  Smiths   16
3 Crinkle   14
4     Cut   14
5  Kettle   13
6  Cheese   12
>
```

##Remove salsa products
transaction_data <- data.table(transaction_data)
transaction_data[, SALSA := grepl("salsa", tolower(PROD_NAME))]
transaction_data <- transaction_data[SALSA == FALSE,][, SALSA := NULL]

##Summarise data to check for nulls and possible outliers

summary(transaction_data)

```
> summary(transaction_data)
      DATE              STORE_NBR     LYLTY_CARD_NBR        TXN_ID
 Min.   :2018-07-01   Min.   :  1.0   Min.   :   1000   Min.   :      1
 1st Qu.:2018-09-30   1st Qu.: 70.0   1st Qu.:  70015   1st Qu.:  67569
 Median :2018-12-30   Median :130.0   Median : 130367   Median : 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
>
```

sum(is.na(transaction_data))

```
 Max.   :114.00
> sum(is.na(transaction_data))
[1] 0
>
```

There are no nulls in the columns but product quantity appears to have an outlier which should be investigated. Case refers where 200 packets of chips are bought in one transaction.

##Filter the dataset to find the outlier
outlier <- transaction_data[PROD_QTY == 200,]
print(outlier)

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

There are two such transactions by the same customer based on Loyalty card number. Let’s now try to track other transactions of the customers.

##Filter out the customers based on the loyalty card number

outlierTransactions <- transaction_data[LYLTY_CARD_NBR == 226000,]

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

#Filter out the customer based on loyalty card number
Transaction_data <- transaction_data[LYLTY_CARD_NBR != 226000,]

##Re-examine transaction data

Summary(transaction_data)

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

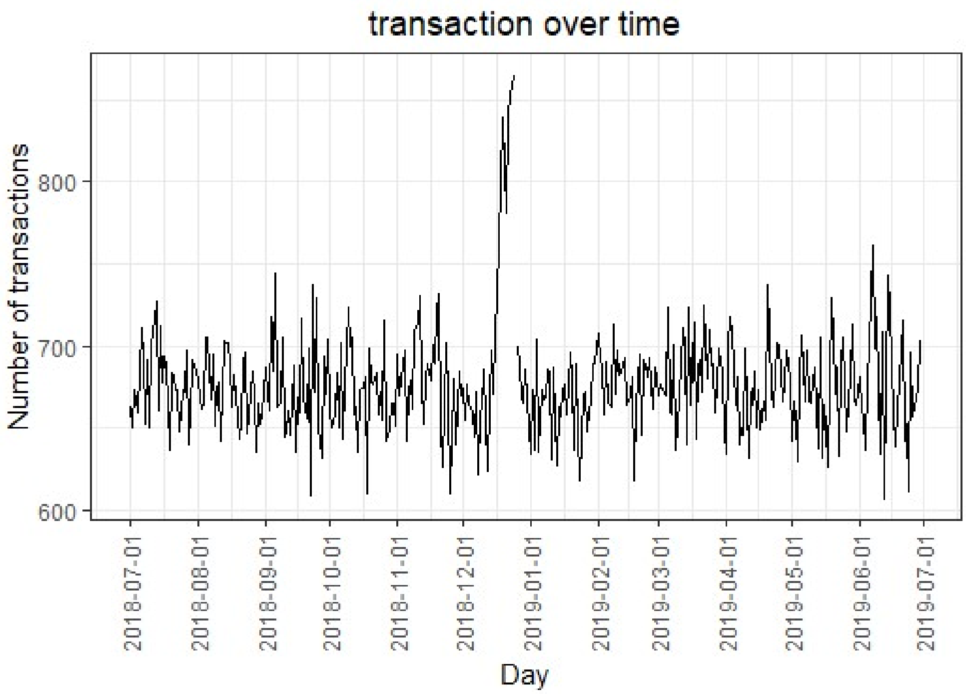
This looks much better. Now we will focus on the number of transaction lines over time to see any obvious issues such as missing data.

##Count the number of transactions by data
Transaction_data[, .N, by= DATE]

```
> transaction_data[, .N, by= DATE]
           DATE   N
  1: 2018-10-17 682
  2: 2019-05-14 705
  3: 2019-05-20 707
  4: 2018-08-17 663
  5: 2018-08-18 683
 ---
360: 2018-12-08 622
361: 2019-01-30 689
362: 2019-02-09 671
363: 2018-08-31 658
364: 2019-02-12 684
>
```

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

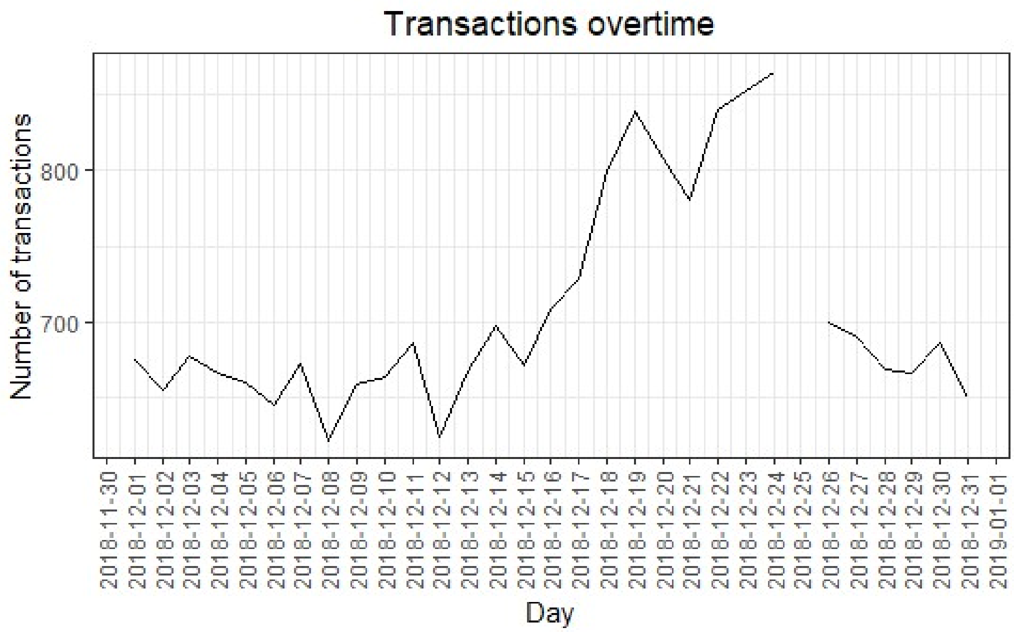
```
###Create a sequence of dates and join this the count of transactions by date
allDates <- data.table(seq(as.Date("2018/07/01"), as.Date("2019/06/30"), by = "day"))
setnames(allDates, "DATE")
transactions_by_data <- merge(allDates, transaction_data[, .N, by=DATE], all.x=TRUE)

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

ggplot(transactions_by_data, aes(x=DATE, y=N))+
  geom_line()+
  labs(x = "Day", y="Number of transactions", title="transaction 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 purchase in December and a break in a late December.

###Filter to December and look at individual days

```
ggplot(transactions_by_day[month(DATE) == 12, ], aes(x = DATE, y = N)) +
geom_line() +
labs(x = "Day", y = "Number of transactions", title = "Transactions over ↪ time") +
scale_x_date(breaks = "1 day") +
theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



We can see that the increase in sales occurs in the lead-up to Christmas and that there are zero sales on Christmas day itself. This is due to shops being closed on Christmas day.

Now that we are satisfied that the data no longer has outliers, we can move on to creating other features such as brand of chips, pack isze from PROD_NAME

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

Always check your output
Let's check if the pack sizes look sensible transactionData[, .N, PACK_SIZE][order(PACK_SIZE)]

```
> transaction_data[, .N, PACK_SIZE][order(PACK_SIZE)]
    PACK_SIZE     N
 1:        70  1507
 2:        90  3008
 3:       110 22387
 4:       125  1454
 5:       134 25102
 6:       135  3257
 7:       150 40203
 8:       160  2970
 9:       165 15297
10:       170 19983
11:       175 66390
12:       180  1468
13:       190  2995
14:       200  4473
```

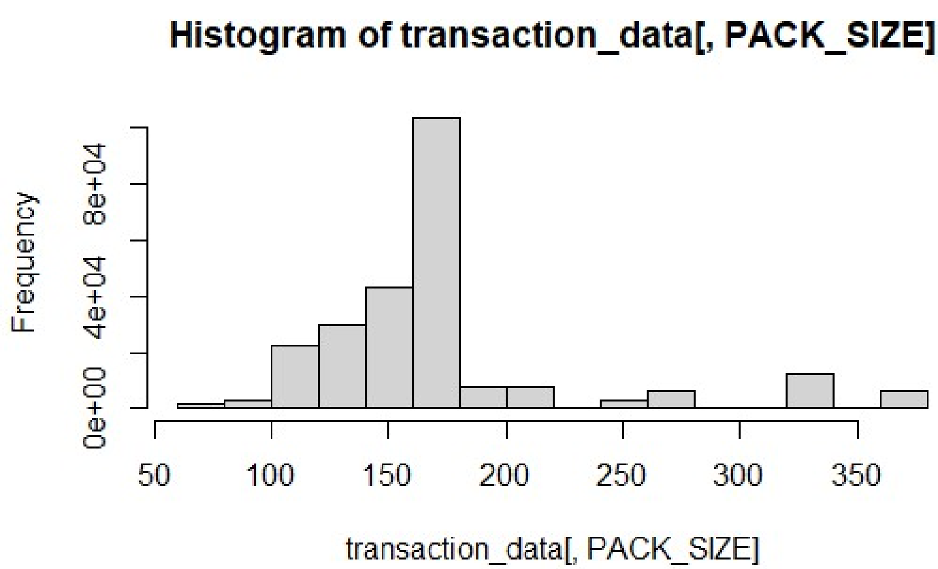
```
15:         210  6272
16:         220  1564
17:         250  3169
18:         270  6285
19:         330 12540
20:         380  6416
```

The largest size is 380g and the smallest pack is 70g

Let's check the output of the first few rows to see if we have indeed ↪ picked out pack
####size.
 transactionData

```
> transaction_data
            DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR                                 PROD_NAME PROD_QTY TOT_SALES PACK_SIZE
     1: 2018-10-17         1           1000      1        5   Natural Chip        Compny SeaSalt175g        2       6.0       175
     2: 2019-05-14         1           1307    348       66                 CCs Nacho Cheese    175g        3       6.3       175
     3: 2019-05-20         1           1343    383       61   Smiths Crinkle Cut  Chips Chicken 170g        2       2.9       170
     4: 2018-08-17         2           2373    974       69   Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0       175
     5: 2018-08-18         2           2426   1038      108 Kettle Tortilla ChpsHny&Jlpno Chili 150g        3      13.8       150
    ---
246736: 2019-03-09       272         272319 270088       89  Kettle Sweet Chilli And Sour Cream 175g        2      10.8       175
246737: 2018-08-13       272         272358 270154       74             Tostitos Splash Of  Lime 175g        1       4.4       175
246738: 2018-11-06       272         272379 270187       51                 Doritos Mexicana    170g        2       8.8       170
246739: 2018-12-27       272         272379 270188       42  Doritos Corn Chip Mexican Jalapeno 150g        2       7.8       150
246740: 2018-09-22       272         272380 270189       74             Tostitos Splash Of  Lime 175g        2       8.8       175
>
```

#let's plot the a histogram of pack size since we know that it is a categorical variable not a continuous variable even though it is numeric.
Hist(transaction_data[, Pack_Size])



Pack sizes look reasonable and now to create brands, we can ue the first word PROD_NAME to work out the brand name

###Brands
transaction_data[, BRAND := toupper(substr(PROD_NAME, 1, regexpr(patter=' ', PROD_NAME) - 1))]

Checking brands
transaction_data[, .N, by = BRAND][order(-N)]

```
> ## Checking brands
> transaction_data[, .N, by = BRAND][order(-N)]
         BRAND     N
 1:     KETTLE 41288
 2:     SMITHS 27390
 3:   PRINGLES 25102
 4:    DORITOS 22041
 5:      THINS 14075
 6:        RRD 11894
 7:  INFUZIONS 11057
 8:         WW 10320
 9:       COBS  9693
10:   TOSTITOS  9471
11:   TWISTIES  9454
12:   TYRRELLS  6442
13:      GRAIN  6272
14:    NATURAL  6050
15:   CHEEZELS  4603
16:        CCS  4551
17:        RED  4427
18:     DORITO  3183
```

```
19:     INFZNS  3144
20:      SMITH  2963
21:    CHEETOS  2927
22:      SNBTS  1576
23:     BURGER  1564
24: WOOLWORTHS  1516
25:    GRNWVES  1468
26:   SUNBITES  1432
27:        NCC  1419
28:     FRENCH  1418
         BRAND     N
```

Some of the brand names look like they are the same brands – such as RED and RRD, which are both Red Rock Deli Chips. Let's combines these together.

###Cleaning brand names
transaction_data[BRAND == "RED", BRAND := "RRD"]
transaction_data[BRAND == "SNBTS", BRAND := "SUNBITES"]
transaction_data[BRAND == "INFSNS", BRAND := "INFUZIONS"]
transaction_data[BRAND == "WW", BRAND := "WOOLWORTHS"]
transaction_data[BRAND == "SMITH", BRAND := "SMITHS"]
transaction_data[BRAND == "NCC", BRAND := "NATURAL"]
transaction_data[BRAND == "DORITO", BRAND := "DORITOS"]
transaction_data[BRAND == "GRAIN", BRAND := "GRNWVES"]

##Check again

transaction_data[, .N, by = BRAND][order(BRAND)]

```
> transaction_data[, .N, by = BRAND][order(BRAND)]
          BRAND     N
 1:      BURGER  1564
 2:         CCS  4551
 3:     CHEETOS  2927
 4:    CHEEZELS  4603
 5:        COBS  9693
 6:     DORITOS 25224
 7:      FRENCH  1418
 8:     GRNWVES  7740
 9:   INFUZIONS 11057
10:      INFZNS  3144
11:      KETTLE 41288
12:     NATURAL  7469
13:    PRINGLES 25102
14:         RRD 16321
15:      SMITHS 30353
16:    SUNBITES  3008
17:       THINS 14075
18:    TOSTITOS  9471
19:    TWISTIES  9454
20:    TYRRELLS  6442
21:  WOOLWORTHS 11836
          BRAND     N
```

- **Examining customer data:**

##Examing Customer data

str(pb)

```
> str(pb)
'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" ...
>
```

##Some basic summaries of dataset

summary(pb)

```
...
> ##Some basic summaries of dataset
> summary(pb)
 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                                        
>
```

#Examining the values of lifestage and premium customers
customer_data[, .N, by = LIFESTAGE][order(-N)]

```
> customer_data[, .N, by = LIFESTAGE][order(-N)]
                LIFESTAGE     N
1:               RETIREES 14805
2:  OLDER SINGLES/COUPLES 14609
3:  YOUNG SINGLES/COUPLES 14441
4:         OLDER FAMILIES  9780
5:         YOUNG FAMILIES  9178
6: MIDAGE SINGLES/COUPLES  7275
7:           NEW FAMILIES  2549
>
```

customer_data[, .N, by = PREMIUM_CUSTOMER][order(-N)]

```
> customer_data[, .N, by = PREMIUM_CUSTOMER][order(-N)]
   PREMIUM_CUSTOMER     N
1:       Mainstream 29245
2:           Budget 24470
3:          Premium 18922
>
```

As there do not seem to be issues with customer data, we can go ahead and join the transaction data and customer data.

##merging transaction data to customer data
data <- merge(transaction_data, customer_data, all.x = TRUE)

As the number of rows in data is the same as that of transaction data, we can be sure that no duplicates were created. This is because we create data by setting all.x =true (in other words left join).

#checking for nulls
Data[is.null(LIFESTAGE), .N]

Data[is.null(PREMIUM_CUSTOMER), .N]

```
> #checking for nulls
> data[is.null(LIFESTAGE), .N]
[1] 0
> Data[is.null(PREMIUM_CUSTOMER), .N]
Error: object 'Data' not found
> data[is.null(PREMIUM_CUSTOMER), .N]
[1] 0
>
```

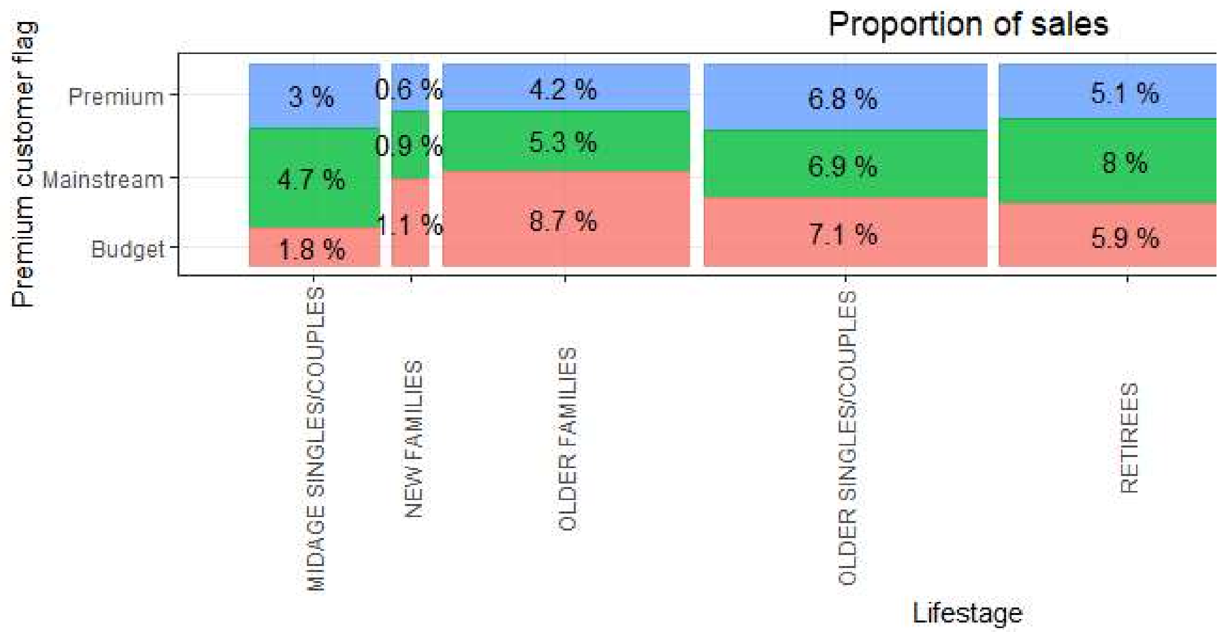
Great, there are no nulls so all our customers in the transaction data has been accounted for in customer dataset.

- **Data analysis on customer segments**

Now, that the data is ready for analysis, we can define some metrics of interest to the client:

- Who spends the most on chips (total sales), describing customers by life stage 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
- The customer's total spends over the period and total spend for each transaction to understand what proportion of their grocery spend is on chip.
- Proportion of customers in each customer segment overall to compare against the mix of customers who purchase chips

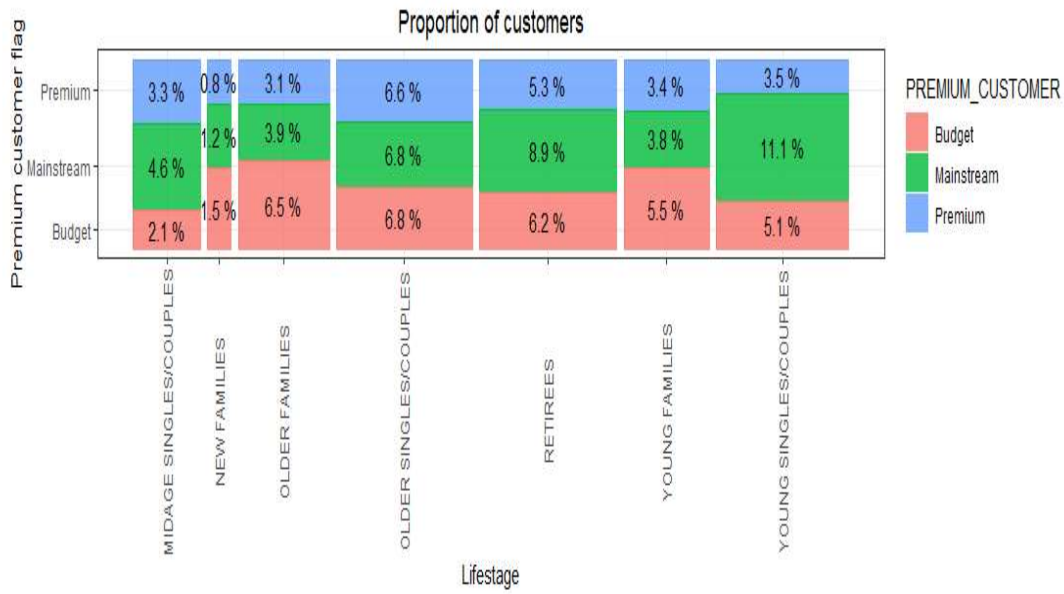
```
##Total Sales by LIFESTAGE and PREMIUM_CUSTOMER
sales <- data[, .(SALES = sum(TOT_SALES)), .(LIFESTAGE, PREMIUM_CUSTOMER)]

##Create plot

p <- ggplot(data = sales)+
  geom_mosaic(aes(weight = SALES, x = product(PREMIUM_CUSTOMER, LIFESTAGE),
              fill = PREMIUM_CUSTOMER))+
  labs(x = "Lifestage", y="Premium customer flag", title="Proportion of sales")+
  theme(axis.text.x = element_text(angle= 90, vjust = 0.5))

## Plot and label with proportion of sales
p + geom_text(data = ggplot_build(p)$data[[1]], aes(x = (xmin + xmax)/2 , y =
                                  (ymin + ymax)/2, label =
as.character(paste(round(.wt/sum(.wt),3)*100,
                                                         '%'))))
```



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

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

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

#### Create plot
p = ggplot(data = customer) +
  geom_mosaic(aes(weight = CUSTOMERS, x = product(PREMIUM_CUSTOMER, LIFESTAGE),
fill = PREMIUM_CUSTOMER)) +
  labs(x = "Lifestage", y = "Premium customer flag", title = "Proportion of customers") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))

#### Plot and label with proportion of customers
p + geom_text(data = ggplot_build(p)$data[[1]], aes(x = (xmin + xmax)/2 , y =
                                 (ymin + ymax)/2, label =
as.character(paste(round(.wt/sum(.wt),3)*100,
'%'))))
```



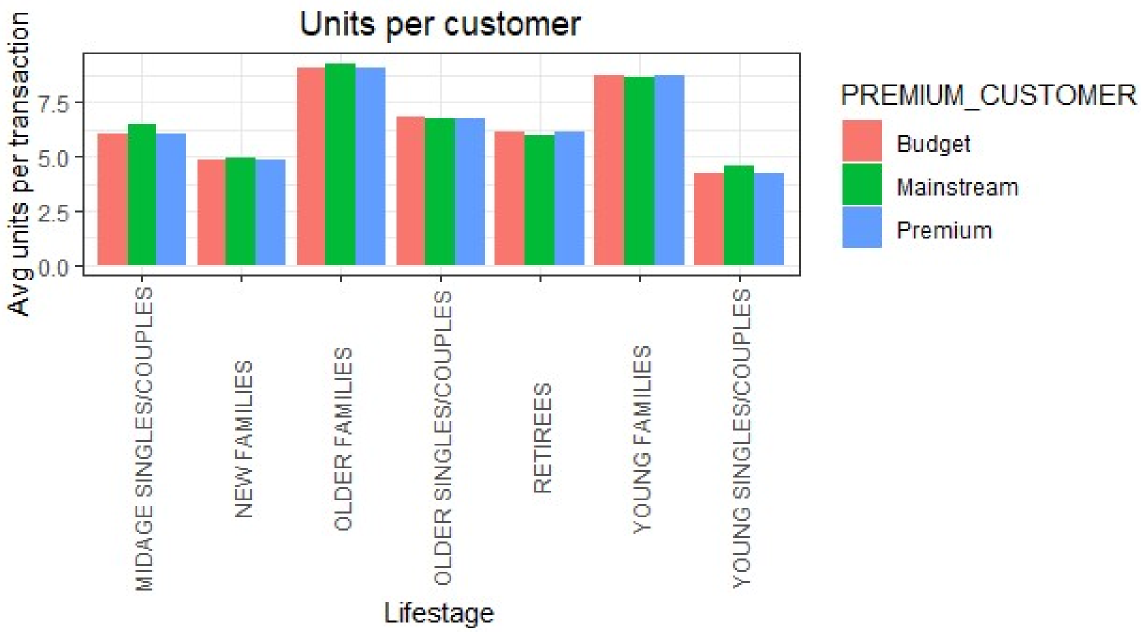
There are more Mainstream – young singles/couples and Mainstream – retirees who buy chips. This contributes to there bring more sales to these customers segments but theis is not a major driver for the budget -older families' segment.

Higher sales may also be driven by more units of chips being bought per customer. Let's have a look at this next.

Average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER
avg_units = data[, .(AVG = sum(PROD_QTY)/uniqueN(LYLTY_CARD_NBR)), .(LIFESTAGE, PREMIUM_CUSTOMER)][order(-AVG)]

Create plot

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



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

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

Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER

avg_price = data[, .(AVG = sum(TOT_SALES)/sum(PROD_QTY)), .(LIFESTAGE, PREMIUM_CUSTOMER)][order(-AVG)]

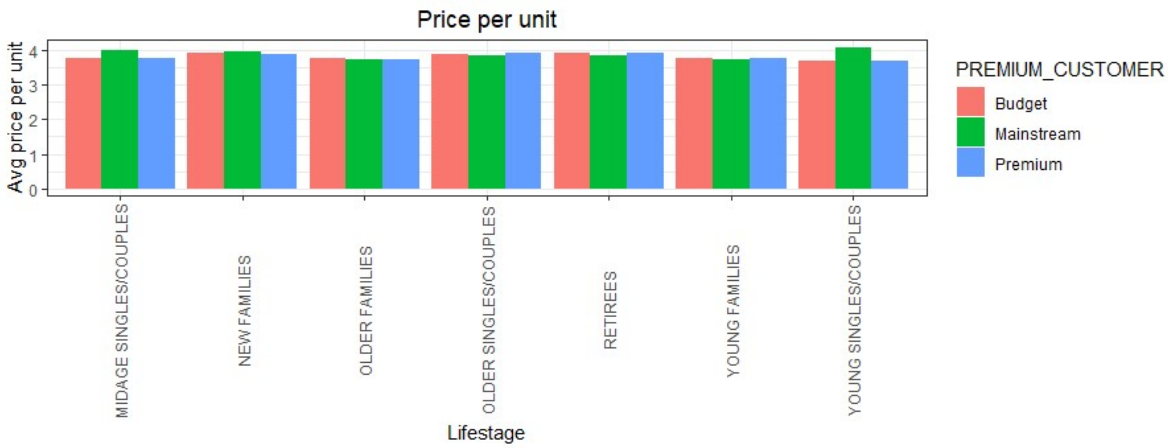
Create plot

ggplot(data = avg_price, aes(weight = AVG, x = LIFESTAGE, fill = PREMIUM_CUSTOMER)) +

 geom_bar(position = position_dodge()) +

 labs(x = "Lifestage", y = "Avg price per unit", title = "Price per unit") +

 theme(axis.text.x = element_text(angle = 90, vjust = 0.5))



Mainstream midage and young singles and couples are more willing to pay more per packet of chips compared to their budget and premium counterparts. This may be due to premium shoppers being more likely to buy healthy snacks and when they buy chips, this is mainly for entertainment purposes rather than their own consumption. This is also supported by there being fewer premium midage and young singles and couples buying chips compared to their mainstream counterparts.

As the difference in average price per unit isn't large, we can check if this difference is statistically different.

Perform an independent t-test between mainstream vs premium and budget midage and

young singles and couples

pricePerUnit = data[, price := TOT_SALES/PROD_QTY]

t.test(data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & PREMIUM_CUSTOMER == "Mainstream", price]

, data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & PREMIUM_CUSTOMER != "Mainstream", price]

, alternative = "greater")

```
        Welch Two Sample t-test

data:  data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") &
PREMIUM_CUSTOMER == "Mainstream", price] and data[LIFESTAGE %in% c("YOUNG SINGLE
S/COUPLES", "MIDAGE SINGLES/COUPLES") & PREMIUM_CUSTOMER != "Mainstream", price]
t = 37.624, df = 54791, p-value < 2.2e-16
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 0.3187234       Inf
sample estimates:
mean of x mean of y
 4.039786  3.706491

>
```

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

➢ Deep Dive into specific customer segments for insights

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

```
#### Deep dive into Mainstream, young singles/couples

segment1 <- data[LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER ==
"Mainstream",]

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


#### Brand affinity compared to the rest of the population

quantity_segment1 <- segment1[, sum(PROD_QTY)]

quantity_other <- other[, sum(PROD_QTY)]

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

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

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

brand_proportions[order(-affinityToBrand)]
```

```
> brand_proportions[order(-affinityToBrand)]
         BRAND targetSegment       other affinityToBrand
 1:   TYRRELLS   0.031552795 0.025692464       1.2280953
 2:   TWISTIES   0.046183575 0.037876520       1.2193194
 3:    DORITOS   0.122760524 0.101074684       1.2145526
 4:     KETTLE   0.197984817 0.165553442       1.1958967
 5:   TOSTITOS   0.045410628 0.037977861       1.1957131
 6:     INFZNS   0.014934438 0.012573300       1.1877898
 7:   PRINGLES   0.119420290 0.100634769       1.1866703
 8:       COBS   0.044637681 0.039048861       1.1431238
 9:  INFUZIONS   0.049744651 0.044491379       1.1180739
10:      THINS   0.060372671 0.056986370       1.0594230
11:    GRNWVES   0.032712215 0.031187957       1.0488733
12:   CHEEZELS   0.017971014 0.018646902       0.9637534
13:     SMITHS   0.096369910 0.124583692       0.7735355
14:     FRENCH   0.003947550 0.005758060       0.6855694
15:    CHEETOS   0.008033126 0.012066591       0.6657329
16:        RRD   0.043809524 0.067493678       0.6490908
17:    NATURAL   0.019599724 0.030853989       0.6352412
18:        CCS   0.011180124 0.018895650       0.5916771
19:   SUNBITES   0.006349206 0.012580210       0.5046980
20: WOOLWORTHS   0.024099379 0.049427188       0.4875733
21:     BURGER   0.002926156 0.006596434       0.4435967
         BRAND targetSegment       other affinityToBrand
>
```

We can see that:

- Mainstream young singles/ couples are 23% more likely to purchase Tyrrells chips compared to the rest of the population.
- Mainstream young singles. Couples are 56% less likely to purchase Burger Rings compared to the rest of the population.

Let's also try to find out if our target segment tends to buy larger packs of chips.

Preferred pack size compared to the rest of the population

quantity_segment1_by_pack <- segment1[, .(targetSegment = sum(PROD_QTY)/quantity_segment1), by = PACK_SIZE]

quantity_other_by_pack <- other[, .(other = sum(PROD_QTY)/quantity_other), by = PACK_SIZE]

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

pack_proportions[order(-affinityToPack)]

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

It looks like Mainstream young singles/couples are 27% more likely top purchase a 270g of chip

Compared to the rest of the population but let's dive into what brands sell this pack size.

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

```
> data[PACK_SIZE == 270, unique(PROD_NAME)]
[1] "Twisties Cheese     270g" "Twisties Chicken270g"
>
```

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

Let's recap what we've found! Sales have mainly been due to Budget - older families, Mainstream - young singles/couples, and Mainstream - retirees shoppers. We found that the high spend in chips for mainstream young singles/couples and retirees is due to there being more of them than other buyers. Mainstream, midage and young singles and couples are also more likely to pay more per packet of chips. This is indicative of impulse buying behaviour. We've also found that Mainstream young singles and couples are 23% more likely to purchase Tyrrells chips compared to the rest of the population. The Category Manager may want to increase the category's performance by off-locating some Tyrrells and smaller packs of chips in discretionary space near segments where young singles and couples frequent more often to increase visibilty and impulse behaviour.