Task 1

Lakmini Herath

2024-07-31

```r
##Load the required libraries
library(data.table)
library(ggplot2)
library(ggmosaic)
library(readr)
library(stringr)
library(stringi)
library(tidyverse)
library(dplyr)
library(writexl)
```

```r
## assign the data files to data.tables
filePath <- "E:/DevOP/quantum/"
transactionData <- fread(paste0(filePath,"QVI_transaction_data.csv"))
customerData <- fread(paste0(filePath,"QVI_purchase_behaviour.csv"))
```

Exploratory data analysis

```r
str(customerData)
```

```
## Classes ’data.table’ and ’data.frame’:   72637 obs. of  3 variables:
##  $ LYLTY_CARD_NBR  : int  1000 1002 1003 1004 1005 1007 1009 1010 1011 1012 ...
##  $ LIFESTAGE       : chr  "YOUNG SINGLES/COUPLES" "YOUNG SINGLES/COUPLES" "YOUNG FAMILIES" "OLDER SI
##  $ PREMIUM_CUSTOMER: chr  "Premium" "Mainstream" "Budget" "Mainstream" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```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 C
##  $ 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>
```

```r
head(transactionData)
```

```
##     DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##    <int>     <int>          <int>  <int>    <int>
## 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
##                                       <char>    <int>     <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
## 6: Old El Paso Salsa   Dip Tomato Mild 300g        1       5.1
```

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

```r
#### Examine PROD_NAME
transactionData[, .N, PROD_NAME]
```

```
##                                      PROD_NAME     N
##                                         <char> <int>
##   1:   Natural Chip        Compny SeaSalt175g  1468
##   2:                 CCs Nacho Cheese    175g  1498
##   3:   Smiths Crinkle Cut  Chips Chicken 170g  1484
##   4:   Smiths Chip Thinly  S/Cream&Onion 175g  1473
##   5: Kettle Tortilla ChpsHny&Jlpno Chili 150g  3296
##  ---
## 110:    Red Rock Deli Chikn&Garlic Aioli 150g  1434
## 111:      RRD SR Slow Rst     Pork Belly 150g  1526
## 112:                 RRD Pc Sea Salt     165g  1431
## 113:       Smith Crinkle Cut   Bolognese 150g  1451
## 114:                 Doritos Salsa Mild  300g  1472
```

```r
####Examine the words in PROD_NAME to see if there are any incorrect entries such as products that are 

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

```r
####Remove digits, and special characters, and then sort the distinct words
####by frequency of occurrence.

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

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

#### Let's look at the most common words by counting the number of times a word appears
wordsSep <- strsplit(productWords$words," ")
words.freq<-table(unlist(wordsSep))

#### sorting them by this frequency in order of highest to lowest frequency
words.freq <-  as.data.frame(words.freq)
words.freq <- words.freq[order(words.freq$Freq, decreasing = T),]
```

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

```r
#### Summarise the data to check for nulls and possible outliers
summary(transactionData)
```

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

```r
#### Filter the dataset to find the outlier
#### investigate further the case where 200 packets of chips are bought in one transaction.
prod_qty_200 <- transactionData %>% filter(PROD_QTY==200)
prod_qty_200
```

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

```r
#### Let's see if the customer has had other transactions
same_customer <- transactionData %>% filter(LYLTY_CARD_NBR == 226000)
same_customer
```

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

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

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

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

```
#### Count the number of transactions by date

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

```
##      transactionData$DATE     n
##                    <Date> <int>
##   1:           2018-07-01   663
##   2:           2018-07-02   650
##   3:           2018-07-03   674
##   4:           2018-07-04   669
##   5:           2018-07-05   660
##  ---
## 360:           2019-06-26   657
## 361:           2019-06-27   669
## 362:           2019-06-28   673
## 363:           2019-06-29   703
## 364:           2019-06-30   704
```

```
nrow(countByDate)
```

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

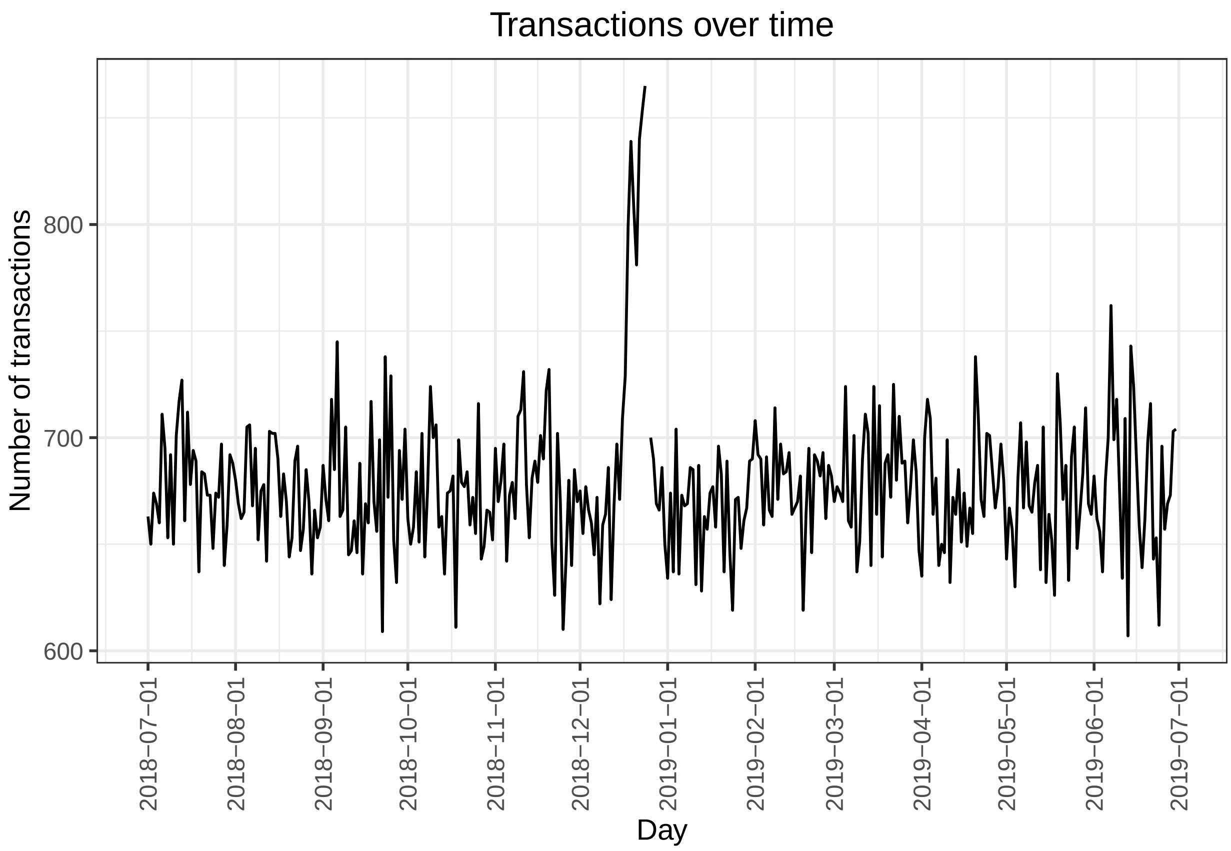
```r
##Create a summary of transaction count by date.
summary(countByDate)
```

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

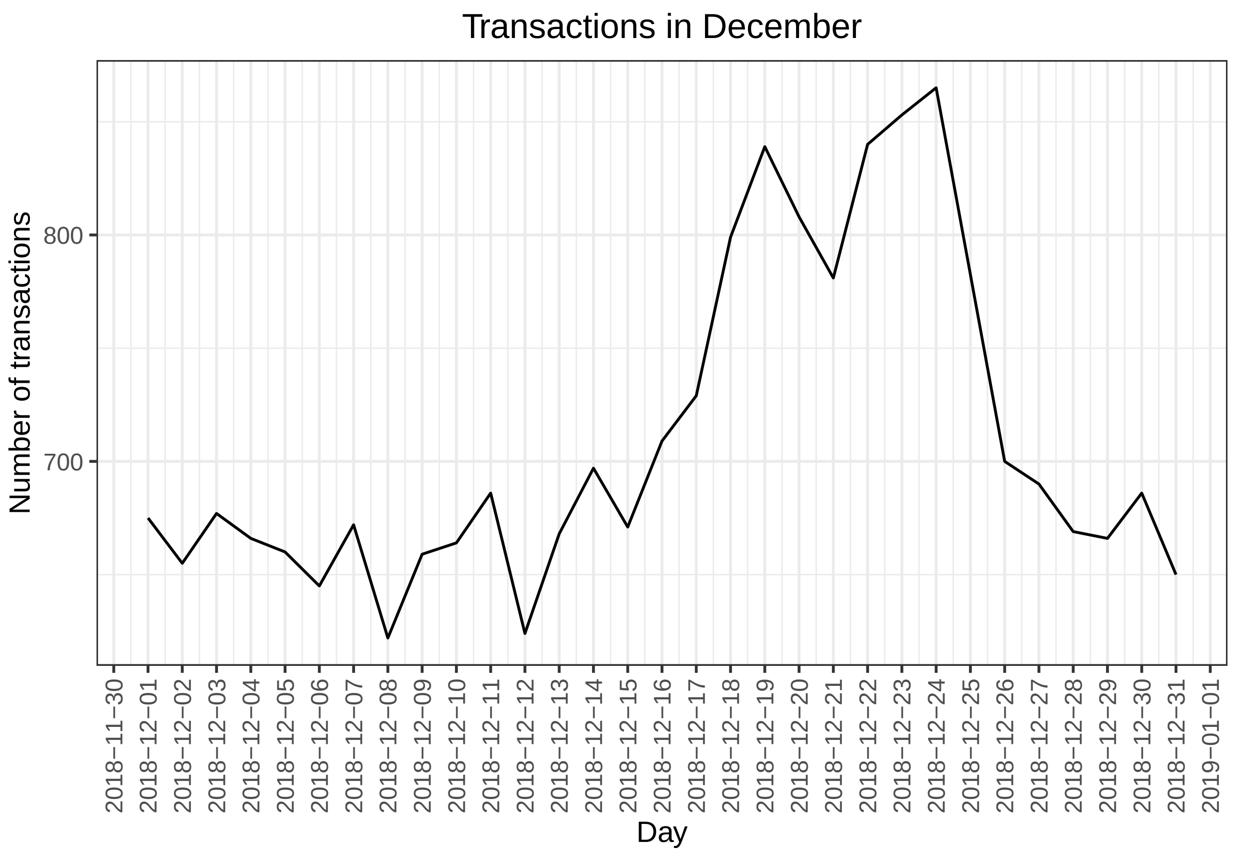
```r
#### Count the number of transactions by date
transactionData[, .N, by = DATE]
```

```
##            DATE     N
##          <Date> <int>
##   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
```

```r
#### Create a sequence of dates and join this the count of transactions by date

####create a column of dates that includes every day from 1 Jul 2018 to 30 Jun 2019, ####join it onto t
#transaction_by_day <- transactionData[order(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_day<- merge(allDates, transactionData[, .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))

#write_xlsx(transactions_by_day, "alldata.xlsx")
#### Plot transactions over time
#ggplot(countByDate, aes(x = countByDate$`transactionData$DATE`, y = countByDate$n)) +
 #geom_line() +
 #labs(x = "Day", y = "Number of transactions", title = "Transactions over time") +
 #scale_x_date(breaks = "1 month") +
 #theme(axis.text.x = element_text(angle = 90, vjust = 0.5))

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



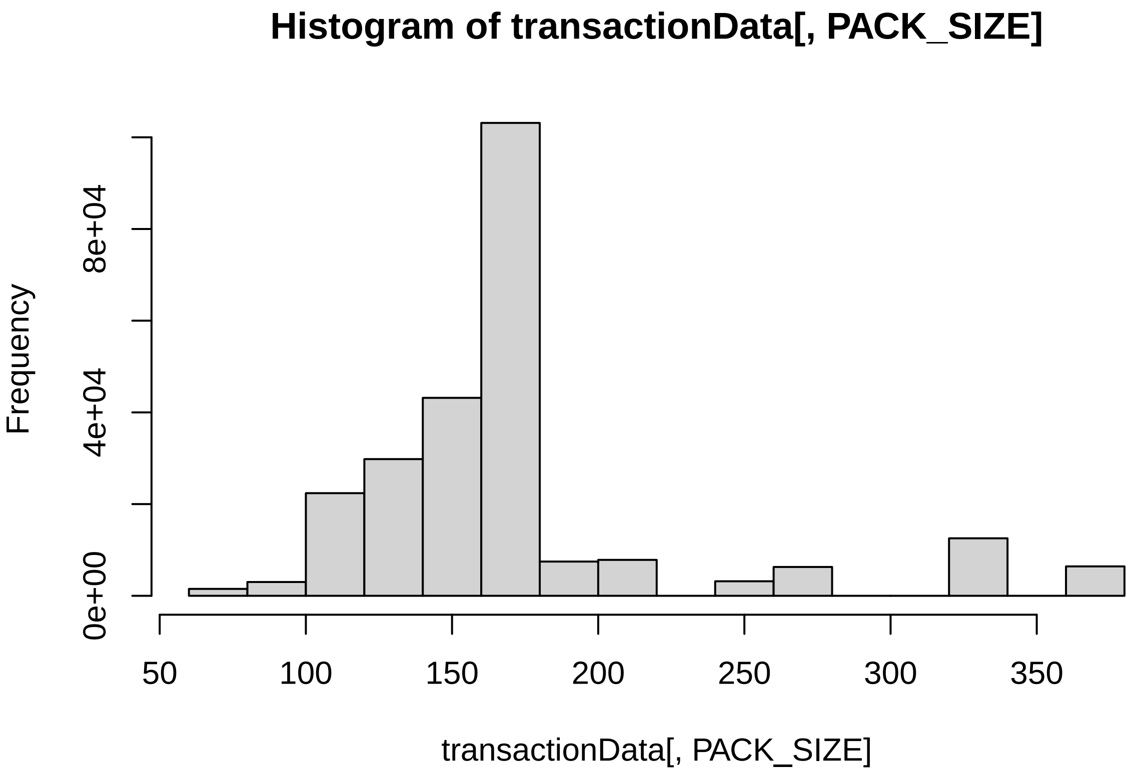
```
#### Filter to December and look at individual days
filterData <- countByDate[countByDate$`transactionData$DATE` >= "2018-12-01" & countByDate$`transaction

#write_xlsx(filterData, "data.xlsx")

ggplot(filterData, aes(x = filterData$`transactionData$DATE`, y = filterData$n)) +
 geom_line() +
 labs(x = "Day", y = "Number of transactions", title = "Transactions in December") +
 scale_x_date(breaks = "1 day") +
 theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



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

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

```
##     PACK_SIZE     N
##         <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
```

```
#### Let's plot a histogram of PACK_SIZE since we know that it is a categorical variable
####and not a continuous variable even though it is numeric.
hist(transactionData[, PACK_SIZE])
```



```
#### Create a column which contains the brand of the product,by extracting it from the product name.

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

#### Checking brands
transactionData[, .N, by = BRAND][order(-N)]
```

```
##          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
```

```
#### Clean brand names
transactionData[BRAND == "RED", BRAND := "RRD"]
transactionData[BRAND == "SNBTS", BRAND := "SUNBITES"]
transactionData[BRAND == "INFZNS", BRAND := "INFUZIONS"]
transactionData[BRAND == "WW", BRAND := "WOOLWORTHS"]
transactionData[BRAND == "SMITH", BRAND := "SMITHS"]
transactionData[BRAND == "NCC", BRAND := "NATURAL"]
transactionData[BRAND == "DORITO", BRAND := "DORITOS"]
transactionData[BRAND == "GRAIN", BRAND := "GRNWVES"]

#### Check again # Over to you! Check the results look reasonable.
transactionData[, .N, by = BRAND][order(-N)]
```

```
##          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: WOOLWORTHS 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:    NATURAL  1419
## 28:     French  1418
##           BRAND     N
```

Examining customer data

```
#### Examining customer data
head(customerData)
```

```
##    LYLTY_CARD_NBR              LIFESTAGE PREMIUM_CUSTOMER
##             <int>                 <char>           <char>
## 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 transaction data to customer data
data <- merge(transactionData, customerData, all.x = TRUE)
```

```
apply(data, 2, function(x) any(is.na(x)))
```

```
##   LYLTY_CARD_NBR             DATE        STORE_NBR           TXN_ID
##            FALSE            FALSE            FALSE            FALSE
##         PROD_NBR        PROD_NAME         PROD_QTY        TOT_SALES
##            FALSE            FALSE            FALSE            FALSE
##        PACK_SIZE            BRAND        LIFESTAGE PREMIUM_CUSTOMER
##            FALSE            FALSE            FALSE            FALSE
```

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

Data analysis on customer segments

```
#### Total sales by LIFESTAGE and PREMIUM_CUSTOMER

total_sales <- data %>% group_by(LIFESTAGE,PREMIUM_CUSTOMER)

total_sales
```

```
## # A tibble: 246,740 x 12
## # Groups:   LIFESTAGE, PREMIUM_CUSTOMER [21]
##    LYLTY_CARD_NBR DATE       STORE_NBR TXN_ID PROD_NBR PROD_NAME        PROD_QTY
##             <int> <date>         <int>  <int>    <int> <chr>               <int>
##  1           1000 2018-10-17         1      1        5 Natural Chip   ~        2
##  2           1002 2018-09-16         1      2       58 Red Rock Deli C~        1
##  3           1003 2019-03-07         1      3       52 Grain Waves Sou~        1
##  4           1003 2019-03-08         1      4      106 Natural ChipCo ~        1
##  5           1004 2018-11-02         1      5       96 WW Original Sta~        1
##  6           1005 2018-12-28         1      6       86 Cheetos Puffs 1~        1
##  7           1007 2018-12-04         1      7       49 Infuzions SourC~        1
##  8           1007 2018-12-05         1      8       10 RRD SR Slow Rst~        1
##  9           1009 2018-11-20         1      9       20 Doritos Cheese ~        1
## 10           1010 2018-09-09         1     10       51 Doritos Mexican~        2
## # i 246,730 more rows
## # i 5 more variables: TOT_SALES <dbl>, PACK_SIZE <dbl>, BRAND <chr>,
## #   LIFESTAGE <chr>, PREMIUM_CUSTOMER <chr>
```

```
pf.total_sales <- summarise(total_sales,sales_count=sum(TOT_SALES))
```

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

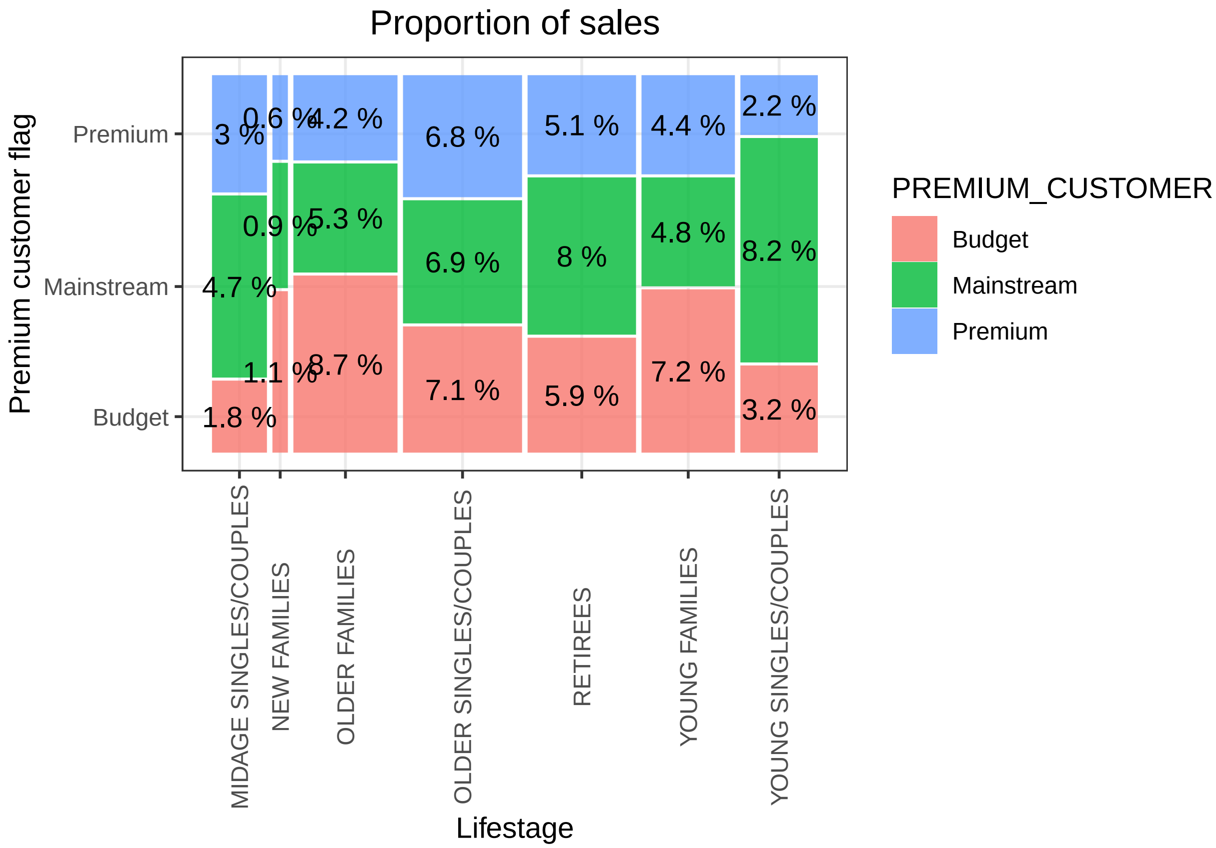
```
summary(pf.total_sales)
```

```
##   LIFESTAGE         PREMIUM_CUSTOMER    sales_count
##  Length:21          Length:21          Min.   : 10761
##  Class :character   Class :character   1st Qu.: 54444
##  Mode  :character   Mode  :character   Median : 86338
##                                        Mean   : 85961
##                                        3rd Qu.:124649
##                                        Max.   :156864
```

```
#write_xlsx(pf.total_sales, "plot_data.xlsx")

#### Create plot
p <- ggplot(pf.total_sales) + geom_mosaic(aes(weight = sales_count, x = product(PREMIUM_CUSTOMER, LIFEST
p +geom_text(data = ggplot_build(p)$data[[1]], aes(x = (xmin + xmax)/2 , y = (ymin + ymax)/2, label = as
## Warning: The ‘scale_name‘ argument of ‘continuous_scale()‘ is deprecated as of ggplot2
## 3.5.0.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

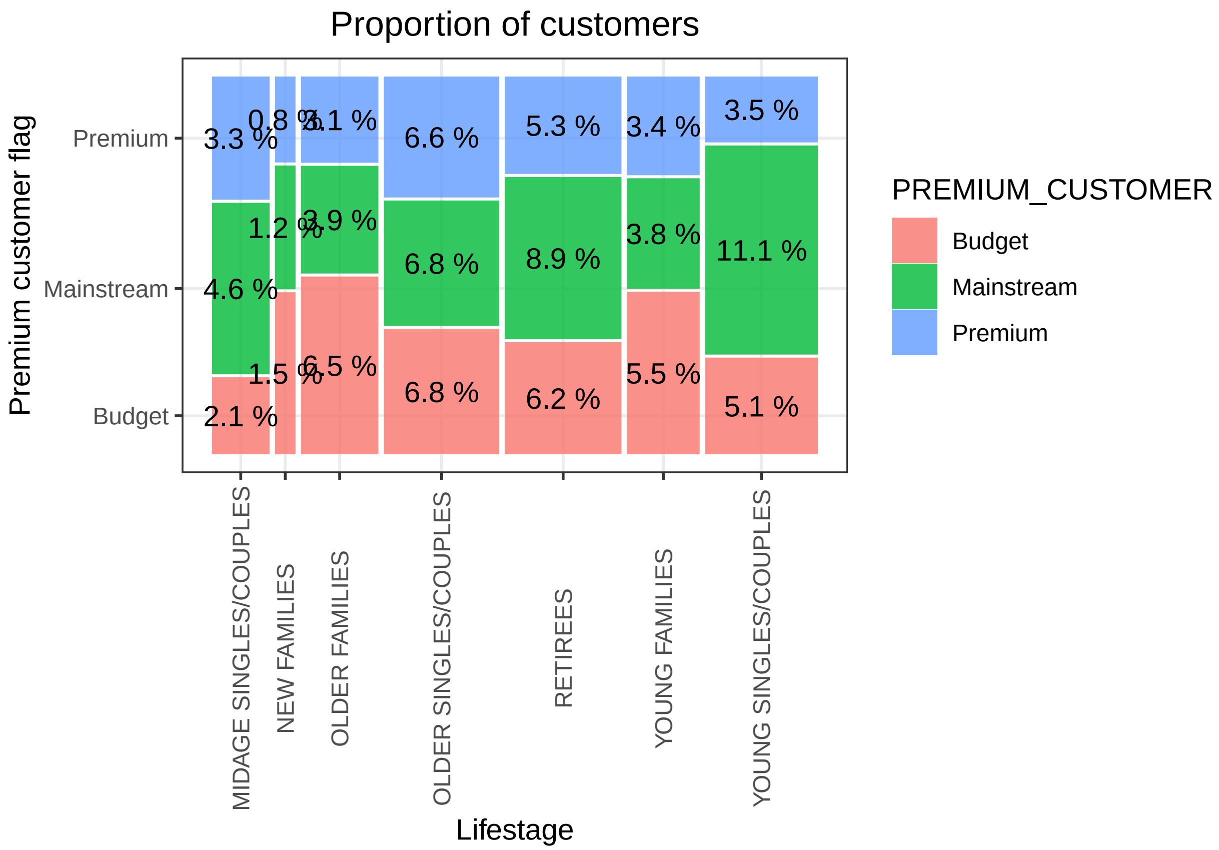
```
## Warning: The ‘trans‘ argument of ‘continuous_scale()‘ is deprecated as of ggplot2 3.5.0.
## i Please use the ‘transform‘ argument instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
## Warning: ‘unite_()‘ was deprecated in tidyr 1.2.0.
## i Please use ‘unite()‘ instead.
## i The deprecated feature was likely used in the ggmosaic package.
##   Please report the issue at <https://github.com/haleyjeppson/ggmosaic>.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



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

write_xlsx(customers, "customer.xlsx")
```

```
p <- ggplot(data = customers) +
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,
'%'))))
```



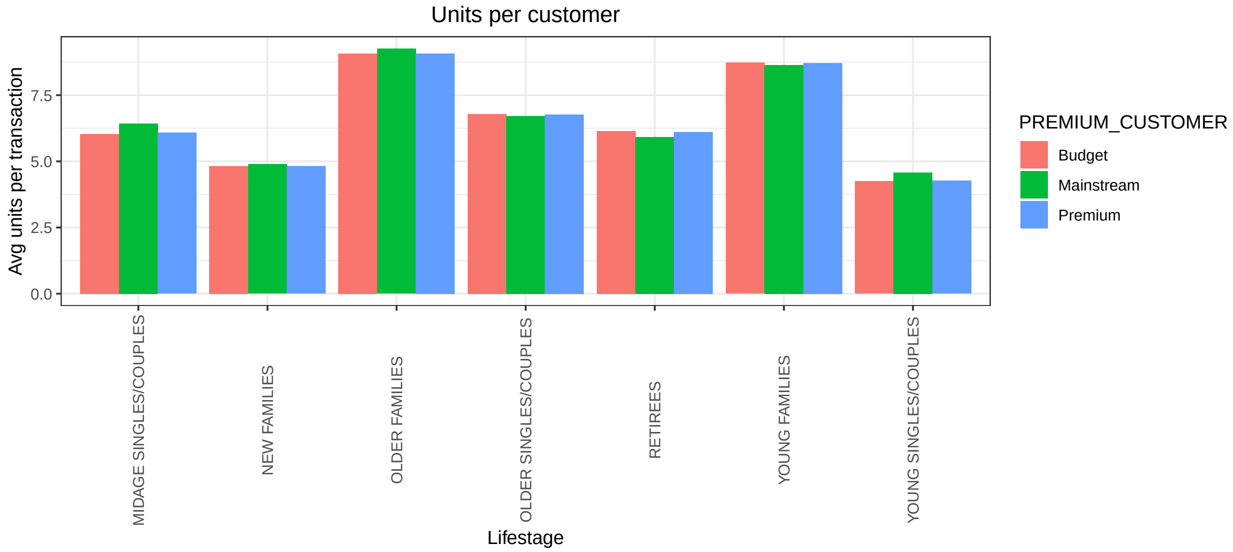
```
#### Average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER - Calculate and plot the av

total_sales_1 <-data %>% group_by(LIFESTAGE,PREMIUM_CUSTOMER)
units <-  summarise(total_sales_1, units_count = (sum(PROD_QTY)/uniqueN(LYLTY_CARD_NBR)))
```

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

```
summary(units)
```

```
##   LIFESTAGE          PREMIUM_CUSTOMER    units_count
##  Length:21          Length:21          Min.   :4.250
##  Class :character   Class :character   1st Qu.:4.892
##  Mode  :character   Mode  :character   Median :6.142
##                     Mean   :6.575
##                     3rd Qu.:8.638
##                     Max.   :9.255
```

```
#write_xlsx(units, "units.xlsx")
###create plot
ggplot(data = units, aes(weight = units_count, x = LIFESTAGE, fill = PREMIUM_CUSTOMER)) + geom_bar(posi
labs(x = "Lifestage", y = "Avg units per transaction", title = "Units per customer") + theme(axis.text.
```

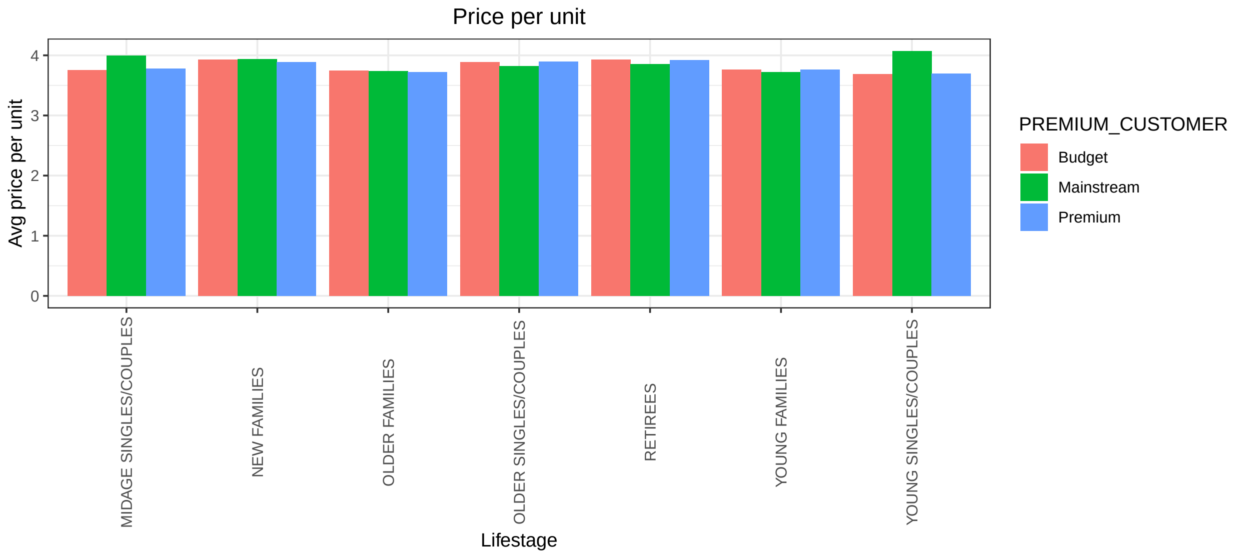


```
check <- units[order(units$units_count, decreasing = T),]
```

```
#### Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER Calculate and plot the average price per

total_sales_2 <-data %>% group_by(LIFESTAGE,PREMIUM_CUSTOMER)

pricePerUnit <-  summarise(total_sales_2, price_per_unit = (sum(TOT_SALES)/sum(PROD_QTY)))
```

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

```
#write_xlsx(pricePerUnit, "price.xlsx")
####plot
ggplot(data=pricePerUnit, aes(weight = price_per_unit,x = LIFESTAGE, fill = PREMIUM_CUSTOMER)) + geom_ba
```



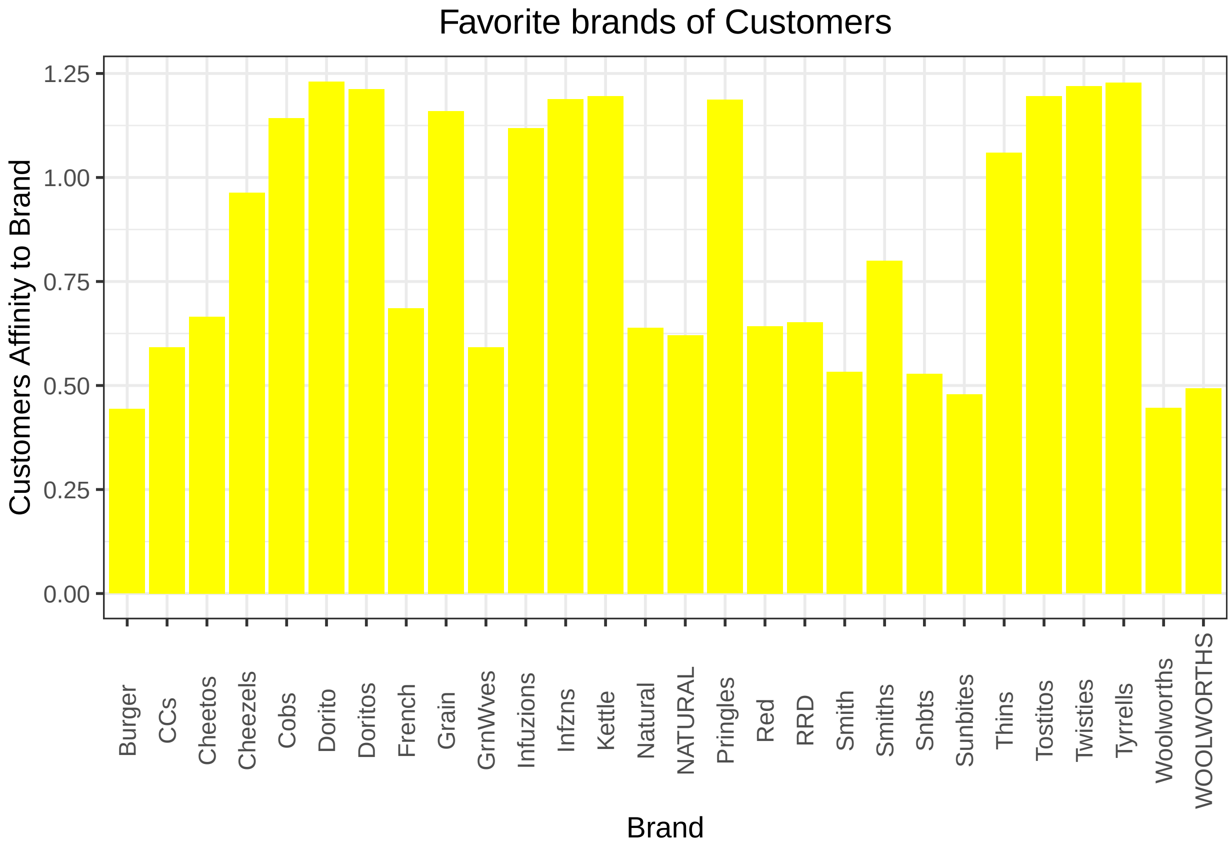
```
#### Perform an independent t-test between mainstream vs premium and budget midage and #### young singl

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

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

```
##
##  Welch Two Sample t-test
##
## data:  data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & PREMIUM_CUSTOMER ==
## 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
```

```
#### 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 := ta
brand_proportions[order(-affinityToBrand)]
```

```
##          BRAND targetSegment       other affinityToBrand
##         <char>        <num>       <num>          <num>
##  1:     Dorito  0.015707384 0.012759861      1.2309996
##  2:   Tyrrells  0.031552795 0.025692464      1.2280953
##  3:   Twisties  0.046183575 0.037876520      1.2193194
##  4:    Doritos  0.107053140 0.088314823      1.2121764
##  5:     Kettle  0.197984817 0.165553442      1.1958967
##  6:   Tostitos  0.045410628 0.037977861      1.1957131
##  7:     Infzns  0.014934438 0.012573300      1.1877898
##  8:   Pringles  0.119420290 0.100634769      1.1866703
##  9:      Grain  0.029123533 0.025121265      1.1593180
## 10:       Cobs  0.044637681 0.039048861      1.1431238
## 11:   Infuzions 0.049744651 0.044491379      1.1180739
## 12:      Thins  0.060372671 0.056986370      1.0594230
## 13:   Cheezels  0.017971014 0.018646902      0.9637534
## 14:     Smiths  0.089772257 0.112215379      0.7999996
## 15:     French  0.003947550 0.005758060      0.6855694
## 16:    Cheetos  0.008033126 0.012066591      0.6657329
## 17:        RRD  0.032022084 0.049150801      0.6515069
## 18:        Red  0.011787440 0.018342876      0.6426168
## 19:    Natural  0.015955832 0.024980768      0.6387246
## 20:    NATURAL  0.003643892 0.005873221      0.6204248
## 21:        CCs  0.011180124 0.018895650      0.5916771
## 22:    GrnWves  0.003588682 0.006066692      0.5915385
## 23:      Smith  0.006597654 0.012368313      0.5334320
## 24:      Snbts  0.003478261 0.006587221      0.5280316
## 25: WOOLWORTHS  0.021256039 0.043049561      0.4937574
## 26:   Sunbites  0.002870945 0.005992989      0.4790507
## 27: Woolworths  0.002843340 0.006377627      0.4458304
## 28:     Burger  0.002926156 0.006596434      0.4435967
##          BRAND targetSegment       other affinityToBrand
```

```
ggplot(brand_proportions, aes(brand_proportions$BRAND,brand_proportions$affinityToBrand)) + geom_bar(sta
```



```
#### Preferred pack size compared to the rest of the population

quantity_segment1_by_pack <- segment1[, .(targetSegment = sum(PROD_QTY)/quantity_segment1), by = PACK_SI
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 := targetS
pack_proportions[order(-affinityToPack)]
```

```
##     PACK_SIZE targetSegment       other affinityToPack
##         <num>         <num>       <num>          <num>
##  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
##     PACK_SIZE targetSegment       other affinityToPack
```

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

```
## [1] "Twisties Cheese     270g" "Twisties Chicken270g"
```