Quantium Virtual Internship - Retail Strategy and Analytics - Task 1

Solution template for Task 1

This file is a solution template for the Task 1 of the Quantium Virtual Internship. It will walk you through the analysis, providing the scaffolding for your solution with gaps left for you to fill in yourself.

Look for comments that say "over to you" for places where you need to add your own code! Often, there will be hints about what to do or what function to use in the text leading up to a code block - if you need a bit of extra help on how to use a function, the internet has many excellent resources on R coding, which you can find using your favourite search engine. ## Load required libraries and datasets Note that you will need to install these libraries if you have never used these before.

```
#### Example code to install packages
#install.packages("data.table")
#### Load required libraries
library(data.table)
library(ggplot2)
library(ggmosaic)
library(readr)
library(stringr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

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

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

```
library(arules)
```

```
## Loading required package: Matrix

##
## 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)
```

```
## Loading required package: grid
```

```
## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus
```

```r
library(datasets)
#### Point the filePath to where you have downloaded the datasets to and
#### assign the data files to data.tables
# over to you! fill in the path to your working directory. If you are on a Windows
↪  machine, you will need to use forward slashes (/) instead of backshashes (\)
filePath <- ""
transactionData <- fread(paste0(filePath,"QVI_transaction_data.csv"))
customerData <- fread(paste0(filePath,"QVI_purchase_behaviour.csv"))
```

Exploratory data analysis

The first step in any analysis is to first understand the data. Let's take a look at each of the datasets provided. ### Examining transaction data We can use `str()` to look at the format of each column and see a sample of the data. As we have read in the dataset as a `data.table` object, we can also run `transactionData` in the console to see a sample of the data or use `head(transactionData)` to look at the first 10 rows. Let's check if columns we would expect to be numeric are in numeric form and date columns are in date format.

```r
#### Examine transaction data
head(transactionData)
```

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

```r
str(transactionData)
```

```
## Classes 'data.table' and 'data.frame': 264836 obs. of 8 variables:
## $ DATE : int 43390 43599 43605 43329 43330 43604 43601 43601 43332 43330 ...
## $ STORE_NBR : int 1 1 1 2 2 4 4 4 5 7 ...
## $ LYLTY_CARD_NBR: int 1000 1307 1343 2373 2426 4074 4149 4196 5026 7150 ...
## $ TXN_ID : int 1 348 383 974 1038 2982 3333 3539 4525 6900 ...
## $ PROD_NBR : int 5 66 61 69 108 57 16 24 42 52 ...
## $ PROD_NAME : chr "Natural Chip Compny SeaSalt175g" "CCs Nacho Cheese 175g"
"Smiths 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 ...
## - attr(*, ".internal.selfref")=<externalptr>
```

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

```r
#### Convert DATE column to a date format
#### A quick search online tells us that CSV and Excel integer dates begin on 30 Dec 1899
transactionData$DATE <- as.Date(transactionData$DATE, origin = "1899-12-30")
```

We should check that we are looking at the right products by examining PROD_NAME.

```r
#### Examine PROD_NAME
summary(transactionData$PROD_NAME)
```

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

```r
unique(transactionData$PROD_NAME)
```

```
##   [1] "Natural Chip        Compny SeaSalt175g"
##   [2] "CCs Nacho Cheese    175g"
##   [3] "Smiths Crinkle Cut  Chips Chicken 170g"
##   [4] "Smiths Chip Thinly  S/Cream&Onion 175g"
##   [5] "Kettle Tortilla ChpsHny&Jlpno Chili 150g"
##   [6] "Old El Paso Salsa   Dip Tomato Mild 300g"
##   [7] "Smiths Crinkle Chips Salt & Vinegar 330g"
##   [8] "Grain Waves         Sweet Chilli 210g"
##   [9] "Doritos Corn Chip Mexican Jalapeno 150g"
##  [10] "Grain Waves Sour    Cream&Chives 210G"
##  [11] "Kettle Sensations   Siracha Lime 150g"
##  [12] "Twisties Cheese     270g"
##  [13] "WW Crinkle Cut      Chicken 175g"
##  [14] "Thins Chips Light&  Tangy 175g"
##  [15] "CCs Original 175g"
##  [16] "Burger Rings 220g"
##  [17] "NCC Sour Cream &    Garden Chives 175g"
##  [18] "Doritos Corn Chip Southern Chicken 150g"
##  [19] "Cheezels Cheese Box 125g"
##  [20] "Smiths Crinkle      Original 330g"
##  [21] "Infzns Crn Crnchers Tangy Gcamole 110g"
##  [22] "Kettle Sea Salt     And Vinegar 175g"
##  [23] "Smiths Chip Thinly  Cut Original 175g"
##  [24] "Kettle Original 175g"
##  [25] "Red Rock Deli Thai  Chilli&Lime 150g"
##  [26] "Pringles Sthrn FriedChicken 134g"
##  [27] "Pringles Sweet&Spcy BBQ 134g"
##  [28] "Red Rock Deli SR    Salsa & Mzzrlla 150g"
##  [29] "Thins Chips         Originl saltd 175g"
##  [30] "Red Rock Deli Sp    Salt & Truffle 150G"
##  [31] "Smiths Thinly       Swt Chli&S/Cream175G"
##  [32] "Kettle Chilli 175g"
##  [33] "Doritos Mexicana    170g"
##  [34] "Smiths Crinkle Cut  French OnionDip 150g"
##  [35] "Natural ChipCo      Hony Soy Chckn175g"
##  [36] "Dorito Corn Chp     Supreme 380g"
```

```
##  [37] "Twisties Chicken270g"
##  [38] "Smiths Thinly Cut   Roast Chicken 175g"
##  [39] "Smiths Crinkle Cut  Tomato Salsa 150g"
##  [40] "Kettle Mozzarella   Basil & Pesto 175g"
##  [41] "Infuzions Thai SweetChili PotatoMix 110g"
##  [42] "Kettle Sensations   Camembert & Fig 150g"
##  [43] "Smith Crinkle Cut   Mac N Cheese 150g"
##  [44] "Kettle Honey Soy    Chicken 175g"
##  [45] "Thins Chips Seasonedchicken 175g"
##  [46] "Smiths Crinkle Cut  Salt & Vinegar 170g"
##  [47] "Infuzions BBQ Rib   Prawn Crackers 110g"
##  [48] "GrnWves Plus Btroot & Chilli Jam 180g"
##  [49] "Tyrrells Crisps     Lightly Salted 165g"
##  [50] "Kettle Sweet Chilli And Sour Cream 175g"
##  [51] "Doritos Salsa       Medium 300g"
##  [52] "Kettle 135g Swt Pot Sea Salt"
##  [53] "Pringles SourCream  Onion 134g"
##  [54] "Doritos Corn Chips  Original 170g"
##  [55] "Twisties Cheese     Burger 250g"
##  [56] "Old El Paso Salsa   Dip Chnky Tom Ht300g"
##  [57] "Cobs Popd Swt/Chlli &Sr/Cream Chips 110g"
##  [58] "Woolworths Mild     Salsa 300g"
##  [59] "Natural Chip Co     Tmato Hrb&Spce 175g"
##  [60] "Smiths Crinkle Cut  Chips Original 170g"
##  [61] "Cobs Popd Sea Salt  Chips 110g"
##  [62] "Smiths Crinkle Cut  Chips Chs&Onion170g"
##  [63] "French Fries Potato Chips 175g"
##  [64] "Old El Paso Salsa   Dip Tomato Med 300g"
##  [65] "Doritos Corn Chips  Cheese Supreme 170g"
##  [66] "Pringles Original   Crisps 134g"
##  [67] "RRD Chilli&         Coconut 150g"
##  [68] "WW Original Corn    Chips 200g"
##  [69] "Thins Potato Chips  Hot & Spicy 175g"
##  [70] "Cobs Popd Sour Crm  &Chives Chips 110g"
##  [71] "Smiths Crnkle Chip  Orgnl Big Bag 380g"
##  [72] "Doritos Corn Chips  Nacho Cheese 170g"
##  [73] "Kettle Sensations   BBQ&Maple 150g"
##  [74] "WW D/Style Chip     Sea Salt 200g"
##  [75] "Pringles Chicken    Salt Crips 134g"
##  [76] "WW Original Stacked Chips 160g"
##  [77] "Smiths Chip Thinly  CutSalt/Vinegr175g"
##  [78] "Cheezels Cheese 330g"
##  [79] "Tostitos Lightly    Salted 175g"
##  [80] "Thins Chips Salt &  Vinegar 175g"
##  [81] "Smiths Crinkle Cut  Chips Barbecue 170g"
##  [82] "Cheetos Puffs 165g"
##  [83] "RRD Sweet Chilli &  Sour Cream 165g"
##  [84] "WW Crinkle Cut      Original 175g"
##  [85] "Tostitos Splash Of  Lime 175g"
##  [86] "Woolworths Medium   Salsa 300g"
##  [87] "Kettle Tortilla ChpsBtroot&Ricotta 150g"
##  [88] "CCs Tasty Cheese    175g"
##  [89] "Woolworths Cheese   Rings 190g"
##  [90] "Tostitos Smoked     Chipotle 175g"
```

```
##  [91] "Pringles Barbeque   134g"
##  [92] "WW Supreme Cheese   Corn Chips 200g"
##  [93] "Pringles Mystery    Flavour 134g"
##  [94] "Tyrrells Crisps     Ched & Chives 165g"
##  [95] "Snbts Whlgrn Crisps Cheddr&Mstrd 90g"
##  [96] "Cheetos Chs & Bacon Balls 190g"
##  [97] "Pringles Slt Vingar 134g"
##  [98] "Infuzions SourCream&Herbs Veg Strws 110g"
##  [99] "Kettle Tortilla ChpsFeta&Garlic 150g"
## [100] "Infuzions Mango     Chutny Papadums 70g"
## [101] "RRD Steak &         Chimuchurri 150g"
## [102] "RRD Honey Soy       Chicken 165g"
## [103] "Sunbites Whlegrn    Crisps Frch/Onin 90g"
## [104] "RRD Salt & Vinegar  165g"
## [105] "Doritos Cheese      Supreme 330g"
## [106] "Smiths Crinkle Cut  Snag&Sauce 150g"
## [107] "WW Sour Cream &OnionStacked Chips 160g"
## [108] "RRD Lime & Pepper   165g"
## [109] "Natural ChipCo Sea  Salt & Vinegr 175g"
## [110] "Red Rock Deli Chikn&Garlic Aioli 150g"
## [111] "RRD SR Slow Rst     Pork Belly 150g"
## [112] "RRD Pc Sea Salt     165g"
## [113] "Smith Crinkle Cut   Bolognese 150g"
## [114] "Doritos Salsa Mild  300g"
```

```
# Over to you! Generate a summary of the PROD_NAME column.
```

Looks like we are definitely looking at potato chips but how can we check that these are all chips? We can do some basic text analysis by summarising the individual words in the product name.

```
#### 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(transactionData[, PROD_NAME]), "
")))
setnames(productWords, 'words')
```

As we are only interested in words that will tell us if the product is chips or not, let's remove all words with digits and special characters such as '&' from our set of product words. We can do this using `grepl()`.

```
# Over to you! Remove digits, and special characters, and then sort the distinct words by
↪  frequency of occurrence.
productWords = gsub('[0-9]+', '', productWords) #removing digits
productWords = gsub('[^[:alnum:]]',',',productWords) #removing special char
#### Let's look at the most common words by counting the number of times a word appears
↪  and
productWords = strsplit(productWords, split=',')
#### sorting them by this frequency in order of highest to lowest frequency
sort(table(productWords),decreasing=TRUE)
```

```
## productWords
## g Chips Smiths Crinkle
## 622 105 21 16 14
## Cut Kettle Cheese Salt Original
## 14 13 12 12 10
## Chip Doritos Salsa Corn Pringles
```

```
## 9 9 9 8 8
## RRD Chicken Chilli Cream WW
## 8 7 7 7 7
## Sea Sour Crisps Thinly Thins
## 6 6 5 5 5
## Vinegar Chives Deli Infuzions Lime
## 5 4 4 4 4
## Natural Red Rock Supreme Sweet
## 4 4 4 4 4
## BBQ CCs Cobs Dip El
## 3 3 3 3 3
## Mild Old Paso Popd Sensations
## 3 3 3 3 3
## Soy Swt Tomato Tortilla Tostitos
## 3 3 3 3 3
## Twisties Woolworths And Burger Cheetos
## 3 3 2 2 2
## Cheezels ChipCo Chs French G
## 2 2 2 2 2
## Garlic Grain Honey Lightly Medium
## 2 2 2 2 2
## Nacho Onion Potato Rings S
## 2 2 2 2 2
## Salted Smith SourCream SR Tangy
## 2 2 2 2 2
## Thai Tyrrells Waves Aioli Bacon
## 2 2 2 1 1
## Bag Balls Barbecue Barbeque Basil
## 1 1 1 1 1
## Belly Big Bolognese Box Btroot
## 1 1 1 1 1
## c Camembert Chckng Ched Cheddr
## 1 1 1 1 1
## Chickeng Chikn Chili Chimuchurri Chipotle
## 1 1 1 1 1
## Chli Chlli Chnky Chp ChpsBtroot
## 1 1 1 1 1
## ChpsFeta ChpsHny Chutny Co Coconut
## 1 1 1 1 1
## Compny Crackers CreamG Crips Crm
## 1 1 1 1 1
## Crn Crnchers Crnkle CutSalt D
## 1 1 1 1 1
## Dorito Fig Flavour Frch FriedChicken
## 1 1 1 1 1
## Fries Garden Gcamole GrnWves Herbs
## 1 1 1 1 1
## Hony Hot Hrb Htg Infzns
## 1 1 1 1 1
## Jalapeno Jam Jlpno Light Mac
## 1 1 1 1 1
## Mango Maple Med Mexican Mexicana
## 1 1 1 1 1
## Mozzarella Mstrd Mystery Mzzrlla N
```

```
## 1 1 1 1 1
## NCC Of Onin OnionDip Oniong
## 1 1 1 1 1
## OnionStacked Orgnl Originl Papadums Pc
## 1 1 1 1 1
## Pepper Pesto Plus Pork Pot
## 1 1 1 1 1
## PotatoMix Prawn Puffs Rib Ricotta
## 1 1 1 1 1
## Roast Rst saltd Sauce SeaSaltg
## 1 1 1 1 1
## Seasonedchicken Siracha Slow Slt Smoked
## 1 1 1 1 1
## Snag Snbts Southern Sp Spce
## 1 1 1 1 1
## Spcy Spicy Splash Sr Stacked
## 1 1 1 1 1
## Steak Sthrn Strws Style Sunbites
## 1 1 1 1 1
## SweetChili Tasty Tmato Tom Truffle
## 1 1 1 1 1
## Veg Vinegr Vinegrg Vingar Whlegrn
## 1 1 1 1 1
## Whlgrn
## 1
```

There are salsa products in the dataset but we are only interested in the chips category, so let's remove these.

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

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

```
#### 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
```

There are no nulls in the columns but product quantity appears to have an outlier which we should investigate further. Let's investigate further the case where 200 packets of chips are bought in one transaction.

```r
#### Filter the dataset to find the outlier
outlier = transactionData[transactionData$PROD_QTY > 199]
outlier
```

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

There are two transactions where 200 packets of chips are bought in one transaction and both of these transactions were by the same customer.

```r
#### Let's see if the customer has had other transactions
customer_outlier = transactionData[transactionData$LYLTY_CARD_NBR == 226000]
customer_outlier
```

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

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

```r
#### 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
```

That's better. Now, let's look at the number of transaction lines over time to see if there are any obvious data issues such as missing data.

```r
#### Count the number of transactions by date
nr.of.transactions = aggregate(x=transactionData$DATE, by=list(sequence_of_dates =
↪  transactionData$DATE), FUN=length)
nr.of.transactions = as.data.frame(nr.of.transactions)
```

```
nr.of.transactions$unique.values
```

```
## NULL
```

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

```
#### Create a sequence of dates and join this the count of transactions by date
sequence_of_dates = seq(as.Date('2018-07-01'), as.Date('2019-06-30'), by='day')
sequence_of_dates = as.data.frame(sequence_of_dates)
transactions_by_day = left_join(sequence_of_dates, nr.of.transactions,
↪  by='sequence_of_dates')
transactions_by_day[is.na(transactions_by_day)] = 0#as.integer((865+700)/2)
transactions_by_day # Create a column of dates that includes every day from 1 Jul 2018 to
↪  30 Jun 2019, and join it onto the data to fill in the missing day.
```

```
##     sequence_of_dates   x
## 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
## 6          2018-07-06 711
## 7          2018-07-07 695
## 8          2018-07-08 653
## 9          2018-07-09 692
## 10         2018-07-10 650
## 11         2018-07-11 701
## 12         2018-07-12 717
## 13         2018-07-13 727
## 14         2018-07-14 661
## 15         2018-07-15 712
## 16         2018-07-16 678
## 17         2018-07-17 694
## 18         2018-07-18 689
## 19         2018-07-19 637
## 20         2018-07-20 684
## 21         2018-07-21 683
## 22         2018-07-22 673
## 23         2018-07-23 673
## 24         2018-07-24 648
## 25         2018-07-25 674
## 26         2018-07-26 672
## 27         2018-07-27 697
## 28         2018-07-28 640
## 29         2018-07-29 659
## 30         2018-07-30 692
## 31         2018-07-31 688
## 32         2018-08-01 680
## 33         2018-08-02 669
## 34         2018-08-03 662
## 35         2018-08-04 665
## 36         2018-08-05 705
```

```
## 37          2018-08-06 706
## 38          2018-08-07 668
## 39          2018-08-08 695
## 40          2018-08-09 652
## 41          2018-08-10 675
## 42          2018-08-11 678
## 43          2018-08-12 642
## 44          2018-08-13 703
## 45          2018-08-14 702
## 46          2018-08-15 702
## 47          2018-08-16 690
## 48          2018-08-17 663
## 49          2018-08-18 683
## 50          2018-08-19 670
## 51          2018-08-20 644
## 52          2018-08-21 653
## 53          2018-08-22 689
## 54          2018-08-23 696
## 55          2018-08-24 647
## 56          2018-08-25 657
## 57          2018-08-26 685
## 58          2018-08-27 670
## 59          2018-08-28 636
## 60          2018-08-29 666
## 61          2018-08-30 653
## 62          2018-08-31 658
## 63          2018-09-01 687
## 64          2018-09-02 671
## 65          2018-09-03 661
## 66          2018-09-04 718
## 67          2018-09-05 685
## 68          2018-09-06 745
## 69          2018-09-07 663
## 70          2018-09-08 666
## 71          2018-09-09 705
## 72          2018-09-10 645
## 73          2018-09-11 647
## 74          2018-09-12 661
## 75          2018-09-13 646
## 76          2018-09-14 688
## 77          2018-09-15 636
## 78          2018-09-16 669
## 79          2018-09-17 660
## 80          2018-09-18 717
## 81          2018-09-19 670
## 82          2018-09-20 656
## 83          2018-09-21 699
## 84          2018-09-22 609
## 85          2018-09-23 738
## 86          2018-09-24 672
## 87          2018-09-25 729
## 88          2018-09-26 652
## 89          2018-09-27 632
## 90          2018-09-28 694
```

```
## 91          2018-09-29 671
## 92          2018-09-30 704
## 93          2018-10-01 662
## 94          2018-10-02 650
## 95          2018-10-03 658
## 96          2018-10-04 684
## 97          2018-10-05 651
## 98          2018-10-06 702
## 99          2018-10-07 644
## 100         2018-10-08 676
## 101         2018-10-09 724
## 102         2018-10-10 700
## 103         2018-10-11 706
## 104         2018-10-12 658
## 105         2018-10-13 663
## 106         2018-10-14 636
## 107         2018-10-15 674
## 108         2018-10-16 675
## 109         2018-10-17 682
## 110         2018-10-18 611
## 111         2018-10-19 699
## 112         2018-10-20 679
## 113         2018-10-21 677
## 114         2018-10-22 684
## 115         2018-10-23 659
## 116         2018-10-24 672
## 117         2018-10-25 655
## 118         2018-10-26 716
## 119         2018-10-27 643
## 120         2018-10-28 649
## 121         2018-10-29 666
## 122         2018-10-30 665
## 123         2018-10-31 652
## 124         2018-11-01 695
## 125         2018-11-02 670
## 126         2018-11-03 680
## 127         2018-11-04 697
## 128         2018-11-05 642
## 129         2018-11-06 673
## 130         2018-11-07 679
## 131         2018-11-08 662
## 132         2018-11-09 710
## 133         2018-11-10 713
## 134         2018-11-11 731
## 135         2018-11-12 678
## 136         2018-11-13 653
## 137         2018-11-14 681
## 138         2018-11-15 689
## 139         2018-11-16 679
## 140         2018-11-17 701
## 141         2018-11-18 690
## 142         2018-11-19 722
## 143         2018-11-20 732
## 144         2018-11-21 651
```

```
## 145         2018-11-22 626
## 146         2018-11-23 702
## 147         2018-11-24 670
## 148         2018-11-25 610
## 149         2018-11-26 642
## 150         2018-11-27 680
## 151         2018-11-28 640
## 152         2018-11-29 685
## 153         2018-11-30 670
## 154         2018-12-01 675
## 155         2018-12-02 655
## 156         2018-12-03 677
## 157         2018-12-04 666
## 158         2018-12-05 660
## 159         2018-12-06 645
## 160         2018-12-07 672
## 161         2018-12-08 622
## 162         2018-12-09 659
## 163         2018-12-10 664
## 164         2018-12-11 686
## 165         2018-12-12 624
## 166         2018-12-13 668
## 167         2018-12-14 697
## 168         2018-12-15 671
## 169         2018-12-16 709
## 170         2018-12-17 729
## 171         2018-12-18 799
## 172         2018-12-19 839
## 173         2018-12-20 808
## 174         2018-12-21 781
## 175         2018-12-22 840
## 176         2018-12-23 853
## 177         2018-12-24 865
## 178         2018-12-25   0
## 179         2018-12-26 700
## 180         2018-12-27 690
## 181         2018-12-28 669
## 182         2018-12-29 666
## 183         2018-12-30 686
## 184         2018-12-31 650
## 185         2019-01-01 634
## 186         2019-01-02 674
## 187         2019-01-03 637
## 188         2019-01-04 704
## 189         2019-01-05 636
## 190         2019-01-06 673
## 191         2019-01-07 668
## 192         2019-01-08 669
## 193         2019-01-09 686
## 194         2019-01-10 685
## 195         2019-01-11 631
## 196         2019-01-12 687
## 197         2019-01-13 628
## 198         2019-01-14 663
```

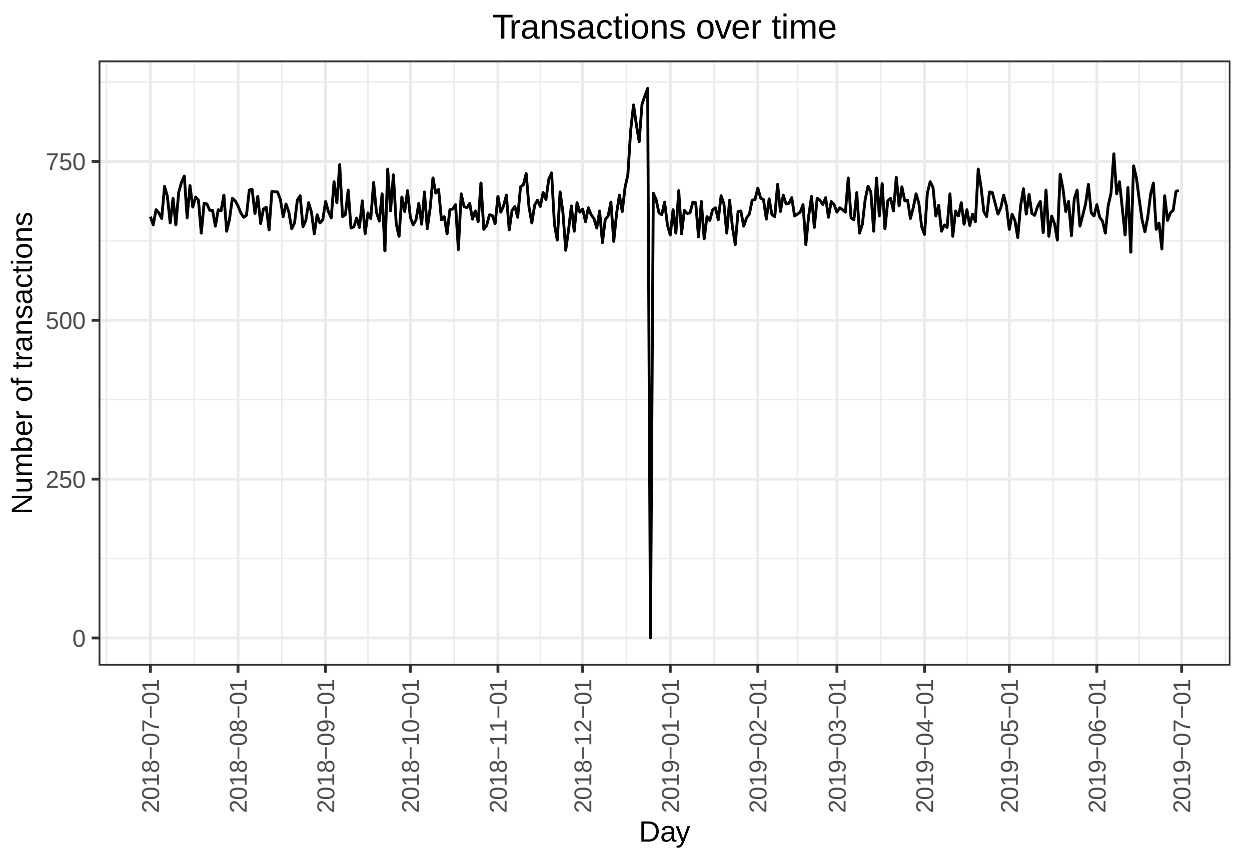
```
## 199          2019-01-15 657
## 200          2019-01-16 674
## 201          2019-01-17 677
## 202          2019-01-18 658
## 203          2019-01-19 696
## 204          2019-01-20 683
## 205          2019-01-21 637
## 206          2019-01-22 689
## 207          2019-01-23 647
## 208          2019-01-24 619
## 209          2019-01-25 671
## 210          2019-01-26 672
## 211          2019-01-27 648
## 212          2019-01-28 661
## 213          2019-01-29 667
## 214          2019-01-30 689
## 215          2019-01-31 690
## 216          2019-02-01 708
## 217          2019-02-02 692
## 218          2019-02-03 690
## 219          2019-02-04 659
## 220          2019-02-05 691
## 221          2019-02-06 666
## 222          2019-02-07 663
## 223          2019-02-08 714
## 224          2019-02-09 671
## 225          2019-02-10 697
## 226          2019-02-11 683
## 227          2019-02-12 684
## 228          2019-02-13 693
## 229          2019-02-14 664
## 230          2019-02-15 667
## 231          2019-02-16 670
## 232          2019-02-17 682
## 233          2019-02-18 619
## 234          2019-02-19 664
## 235          2019-02-20 695
## 236          2019-02-21 646
## 237          2019-02-22 692
## 238          2019-02-23 689
## 239          2019-02-24 682
## 240          2019-02-25 693
## 241          2019-02-26 662
## 242          2019-02-27 687
## 243          2019-02-28 682
## 244          2019-03-01 670
## 245          2019-03-02 677
## 246          2019-03-03 674
## 247          2019-03-04 670
## 248          2019-03-05 724
## 249          2019-03-06 661
## 250          2019-03-07 658
## 251          2019-03-08 701
## 252          2019-03-09 637
```

```
## 253          2019-03-10  651
## 254          2019-03-11  690
## 255          2019-03-12  711
## 256          2019-03-13  702
## 257          2019-03-14  640
## 258          2019-03-15  724
## 259          2019-03-16  664
## 260          2019-03-17  715
## 261          2019-03-18  644
## 262          2019-03-19  688
## 263          2019-03-20  692
## 264          2019-03-21  672
## 265          2019-03-22  725
## 266          2019-03-23  680
## 267          2019-03-24  710
## 268          2019-03-25  688
## 269          2019-03-26  689
## 270          2019-03-27  660
## 271          2019-03-28  677
## 272          2019-03-29  699
## 273          2019-03-30  684
## 274          2019-03-31  647
## 275          2019-04-01  635
## 276          2019-04-02  700
## 277          2019-04-03  718
## 278          2019-04-04  709
## 279          2019-04-05  664
## 280          2019-04-06  681
## 281          2019-04-07  640
## 282          2019-04-08  650
## 283          2019-04-09  646
## 284          2019-04-10  699
## 285          2019-04-11  632
## 286          2019-04-12  672
## 287          2019-04-13  664
## 288          2019-04-14  685
## 289          2019-04-15  651
## 290          2019-04-16  674
## 291          2019-04-17  649
## 292          2019-04-18  667
## 293          2019-04-19  655
## 294          2019-04-20  738
## 295          2019-04-21  710
## 296          2019-04-22  671
## 297          2019-04-23  663
## 298          2019-04-24  702
## 299          2019-04-25  701
## 300          2019-04-26  684
## 301          2019-04-27  667
## 302          2019-04-28  677
## 303          2019-04-29  697
## 304          2019-04-30  680
## 305          2019-05-01  643
## 306          2019-05-02  667
```

```
## 307          2019-05-03 657
## 308          2019-05-04 630
## 309          2019-05-05 680
## 310          2019-05-06 707
## 311          2019-05-07 667
## 312          2019-05-08 698
## 313          2019-05-09 668
## 314          2019-05-10 665
## 315          2019-05-11 679
## 316          2019-05-12 687
## 317          2019-05-13 638
## 318          2019-05-14 705
## 319          2019-05-15 632
## 320          2019-05-16 664
## 321          2019-05-17 652
## 322          2019-05-18 626
## 323          2019-05-19 730
## 324          2019-05-20 707
## 325          2019-05-21 671
## 326          2019-05-22 687
## 327          2019-05-23 633
## 328          2019-05-24 691
## 329          2019-05-25 705
## 330          2019-05-26 648
## 331          2019-05-27 665
## 332          2019-05-28 683
## 333          2019-05-29 714
## 334          2019-05-30 669
## 335          2019-05-31 664
## 336          2019-06-01 682
## 337          2019-06-02 662
## 338          2019-06-03 656
## 339          2019-06-04 637
## 340          2019-06-05 680
## 341          2019-06-06 700
## 342          2019-06-07 762
## 343          2019-06-08 699
## 344          2019-06-09 718
## 345          2019-06-10 676
## 346          2019-06-11 634
## 347          2019-06-12 709
## 348          2019-06-13 607
## 349          2019-06-14 743
## 350          2019-06-15 724
## 351          2019-06-16 690
## 352          2019-06-17 658
## 353          2019-06-18 639
## 354          2019-06-19 662
## 355          2019-06-20 698
## 356          2019-06-21 716
## 357          2019-06-22 643
## 358          2019-06-23 653
## 359          2019-06-24 612
## 360          2019-06-25 696
```

```
## 361        2019-06-26 657
## 362        2019-06-27 669
## 363        2019-06-28 673
## 364        2019-06-29 703
## 365        2019-06-30 704
```

```r
#### Setting plot themes to format graphs
theme_set(theme_bw())
theme_update(plot.title = element_text(hjust = 0.5))
#### Plot transactions over time
ggplot(transactions_by_day, aes(x = sequence_of_dates, y = x)) +
geom_line() +
labs(x = "Day", y = "Number of transactions", title = "Transactions over time") +
scale_x_date(breaks = "1 month") +
theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



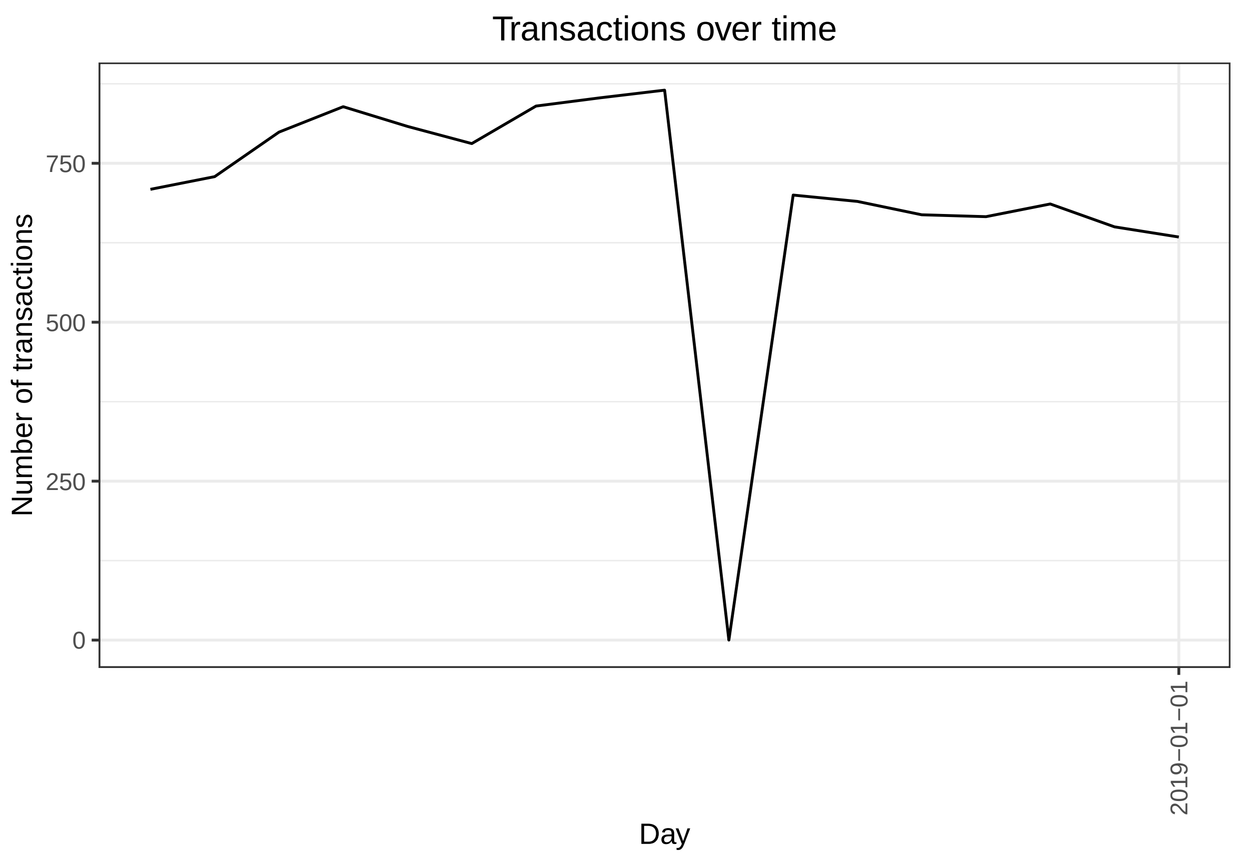
We can see that there is an increase in purchases in December and a break in late December. Let's zoom in on this.

```r
#### Filter to December and look at individual days
subset_dates = subset(transactions_by_day, (transactions_by_day$sequence_of_dates >
↪  as.Date('2018-12-15') & transactions_by_day$sequence_of_dates <
↪  as.Date('2019-01-02')))
subset_dates
```

```
##     sequence_of_dates   x
## 169        2018-12-16 709
## 170        2018-12-17 729
```

```
## 171        2018-12-18 799
## 172        2018-12-19 839
## 173        2018-12-20 808
## 174        2018-12-21 781
## 175        2018-12-22 840
## 176        2018-12-23 853
## 177        2018-12-24 865
## 178        2018-12-25   0
## 179        2018-12-26 700
## 180        2018-12-27 690
## 181        2018-12-28 669
## 182        2018-12-29 666
## 183        2018-12-30 686
## 184        2018-12-31 650
## 185        2019-01-01 634
```

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



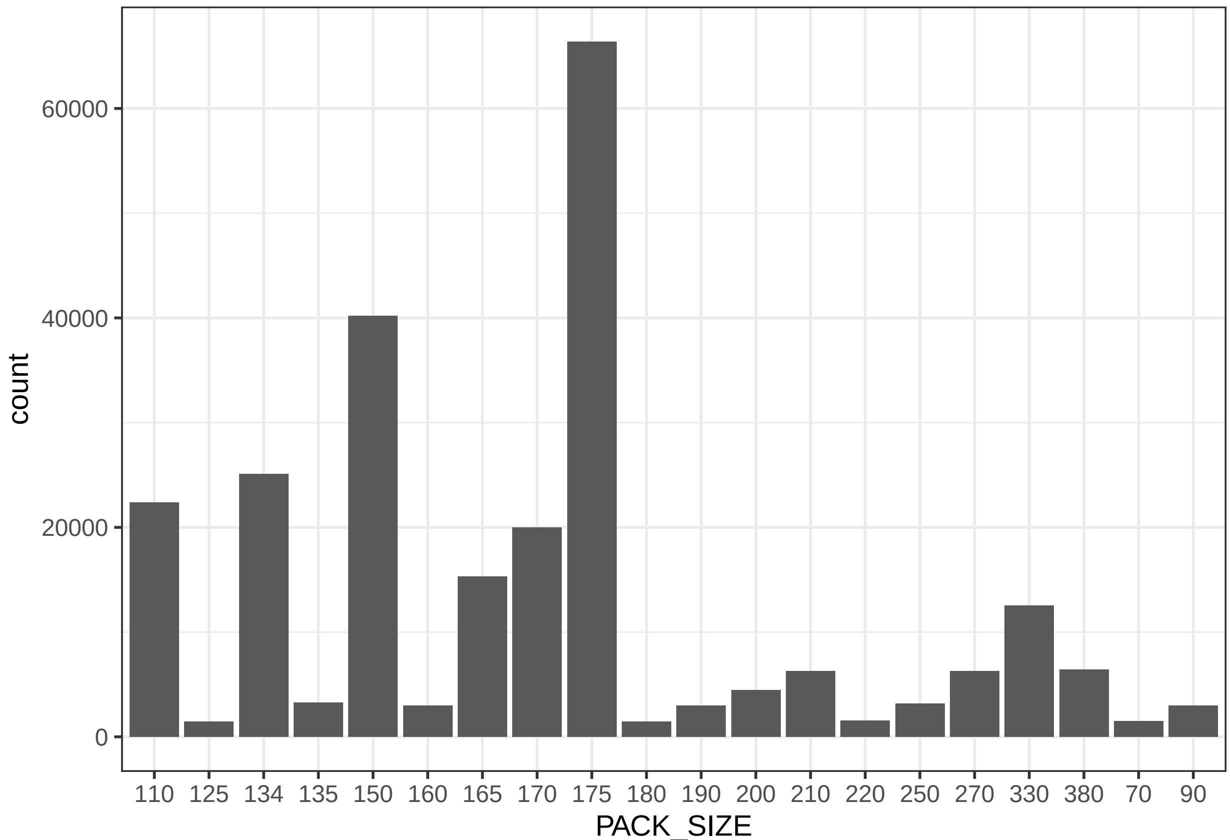
We can see that the increase in sales occurs in the lead-up to Christmas and that there are zero sales on Christmas day itself. This is due to shops being closed on Christmas day. Now that we are satisfied that the data no longer has outliers, we can move on to creating other features such as brand of chips or pack size from PROD_NAME. We will start with pack size.

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

The largest size is 380g and the smallest size is 70g - seems sensible!

```r
#### 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.
x = transactionData[, .N, PACK_SIZE][order(PACK_SIZE)]
transactionData$PACK_SIZE = as.character(transactionData$PACK_SIZE)
ggplot(transactionData, aes(x = PACK_SIZE)) +
    geom_bar()
```



Pack sizes created look reasonable. Now to create brands, we can use the first word in PROD_NAME to work out the brand name...

```
#### Brands
transactionData$BRAND = word(transactionData$PROD_NAME,1)
#### Checking brands
unique(transactionData$BRAND)
```

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

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

```
#### Clean brand names
transactionData[BRAND == "Red", BRAND := "RRD"]
transactionData[BRAND == 'Snbts', BRAND := 'Sunbites']
transactionData[BRAND == 'Grain', BRAND := 'GrainWaves']
transactionData[BRAND == 'GrnWves', BRAND := 'GrainWaves']
transactionData[BRAND == 'Infzns', BRAND := 'Infuzions']
transactionData[BRAND == 'WW', BRAND := 'Woolworths']

unique(transactionData$BRAND)
```

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

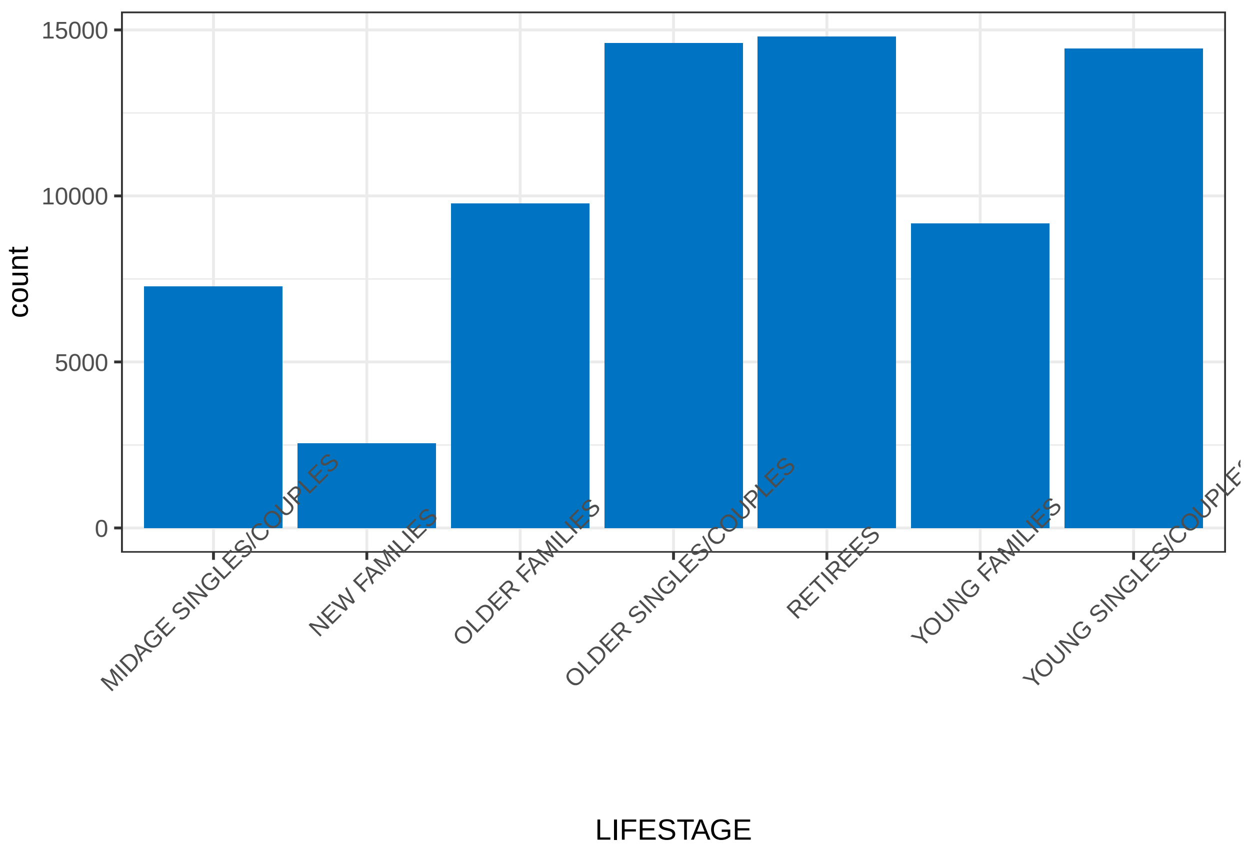
Examining customer data

Now that we are happy with the transaction dataset, let's have a look at the customer dataset.

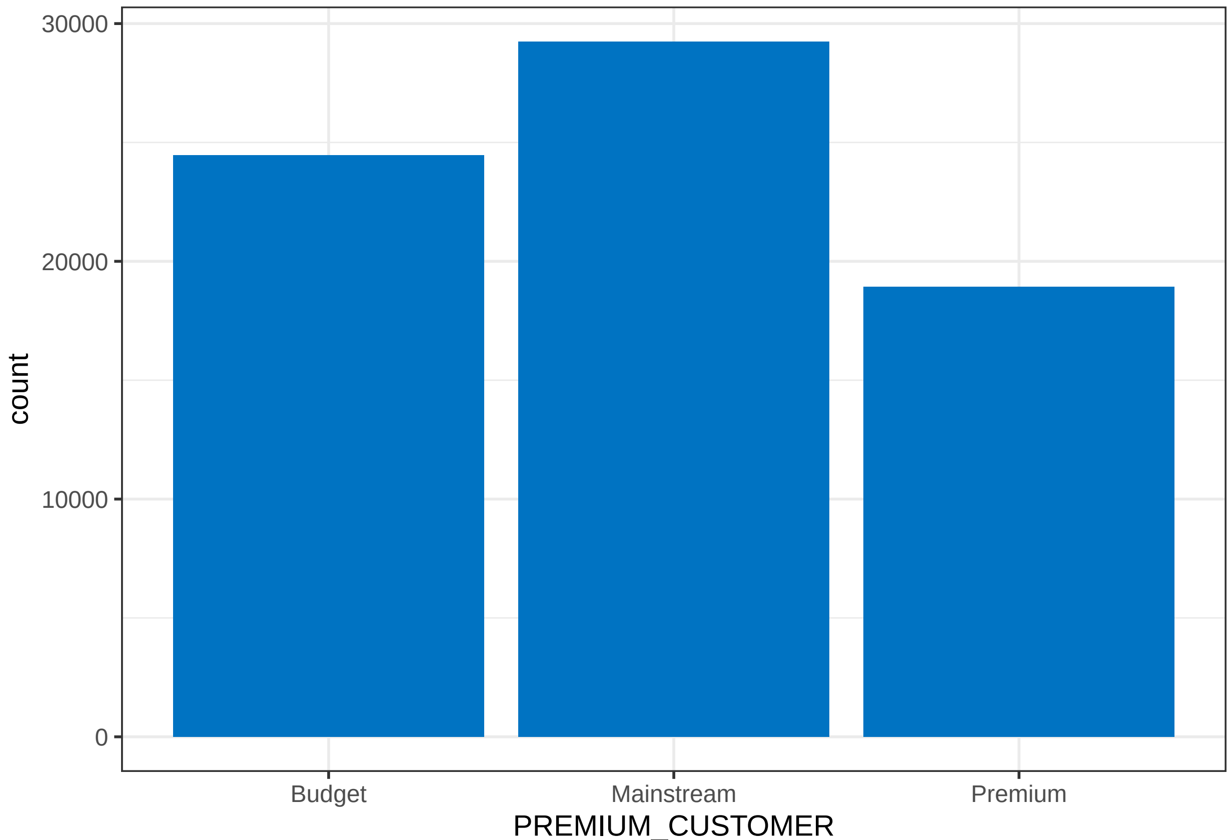
```r
#### Examining customer data
summary(customerData)
```

```
##  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
```

```r
ggplot(customerData, aes(LIFESTAGE)) +
  geom_bar(fill = "#0073C2FF") +
  theme(axis.text.x = element_text(angle = 45))
```




```r
ggplot(customerData, aes(PREMIUM_CUSTOMER)) +
  geom_bar(fill = "#0073C2FF")
```




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

As the number of rows in `data` is the same as that of `transactionData`, we can be sure that no duplicates were created. This is because we created `data` by setting `all.x = TRUE` (in other words, a left join) which means take all the rows in `transactionData` and find rows with matching values in shared columns and then joining the details in these rows to the `x` or the first mentioned table. Let's also check if some customers were not matched on by checking for nulls.

```r
# Over to you! See if any transactions did not have a matched customer.
data[which(is.na(data))]
```

```
## Empty data.table (0 rows and 12 cols):
LYLTY_CARD_NBR,DATE,STORE_NBR,TXN_ID,PROD_NBR,PROD_NAME...
```

Great, there are no nulls! So all our customers in the transaction data has been accounted for in the customer dataset. Note that if you are continuing with Task 2, you may want to retain this dataset which you can write out as a csv

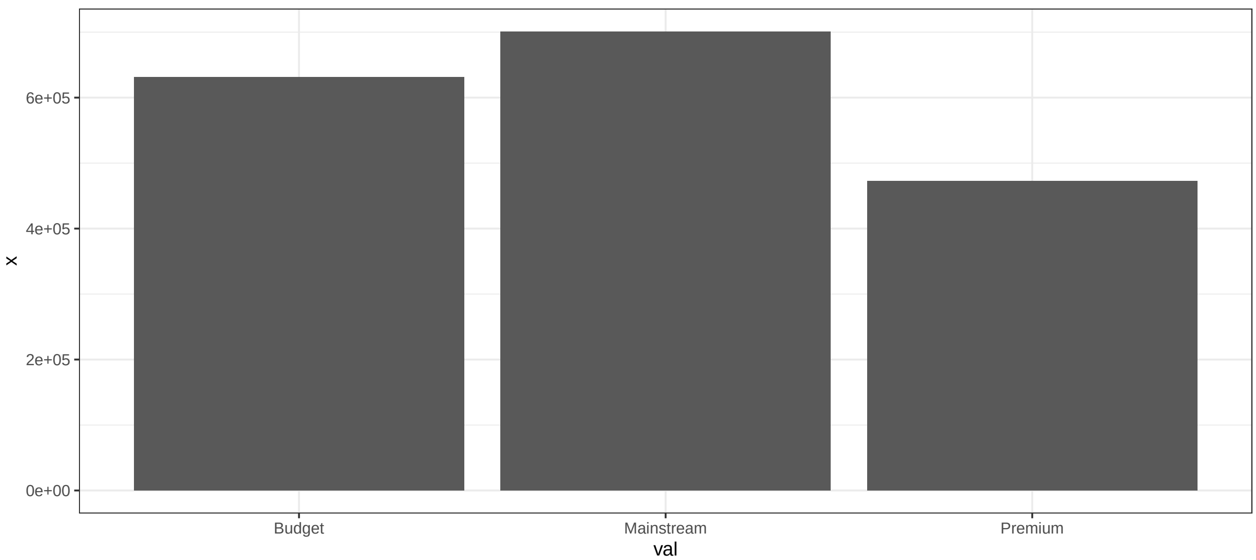
```r
fwrite(data, paste0(filePath,"QVI_data.csv"))
```

Data exploration is now complete! ## Data analysis on customer segments Now that the data is ready for analysis, we can define some metrics of interest to the client: - Who spends the most on chips (total sales), describing customers by lifestage and how premium their general purchasing behaviour is - How many customers are in each segment - How many chips are bought per customer by segment - What's the average chip price by customer segment We could also ask our data team for more information. Examples are: - The customer's total spend over the period and total spend for each transaction to understand what proportion of their grocery spend is on chips - Proportion of customers in each customer segment overall to compare against the mix of customers who purchase chips Let's start with calculating total sales by LIFESTAGE and PREMIUM_CUSTOMER and plotting the split by these segments to describe which customer segment contribute most to chip sales.

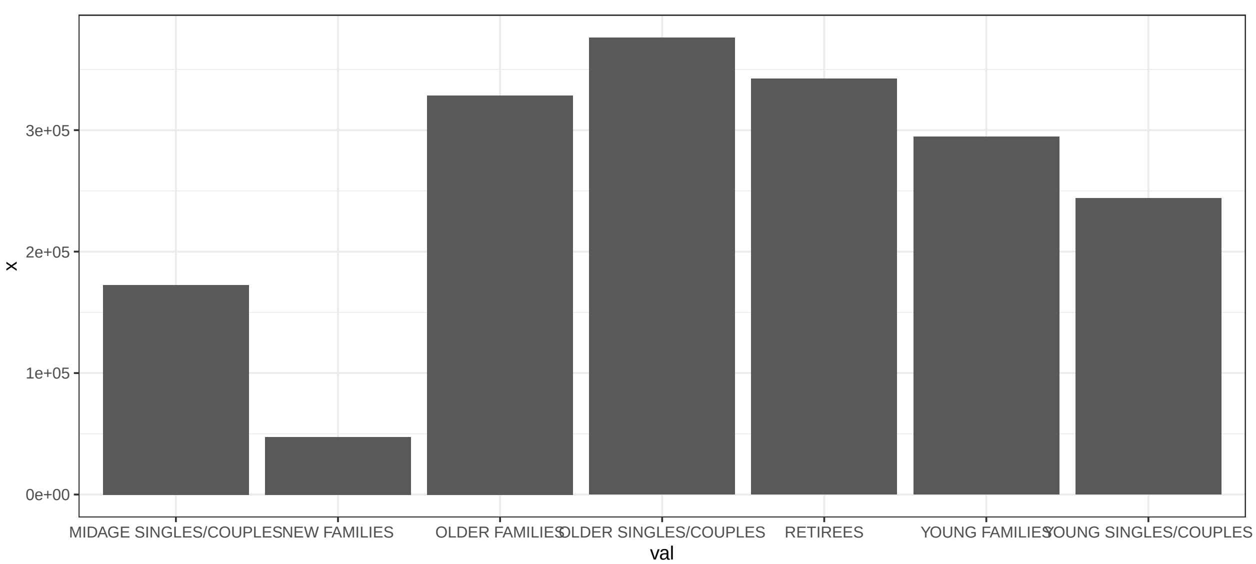
```
#### Total sales by LIFESTAGE and PREMIUM_CUSTOMER
total_lifestage = aggregate(x=data$TOT_SALES, by=list(val =data$LIFESTAGE), FUN=sum)
total_premium_customer = aggregate(x=data$TOT_SALES, by=list(val =
↪  data$PREMIUM_CUSTOMER), FUN=sum)
total_premium_customer
```

```
##          val        x
## 1     Budget 631406.9
## 2 Mainstream 700865.4
## 3    Premium 472905.5
```

```
ggplot(total_premium_customer, aes(x = val, y = x)) + geom_bar(stat="identity")
```



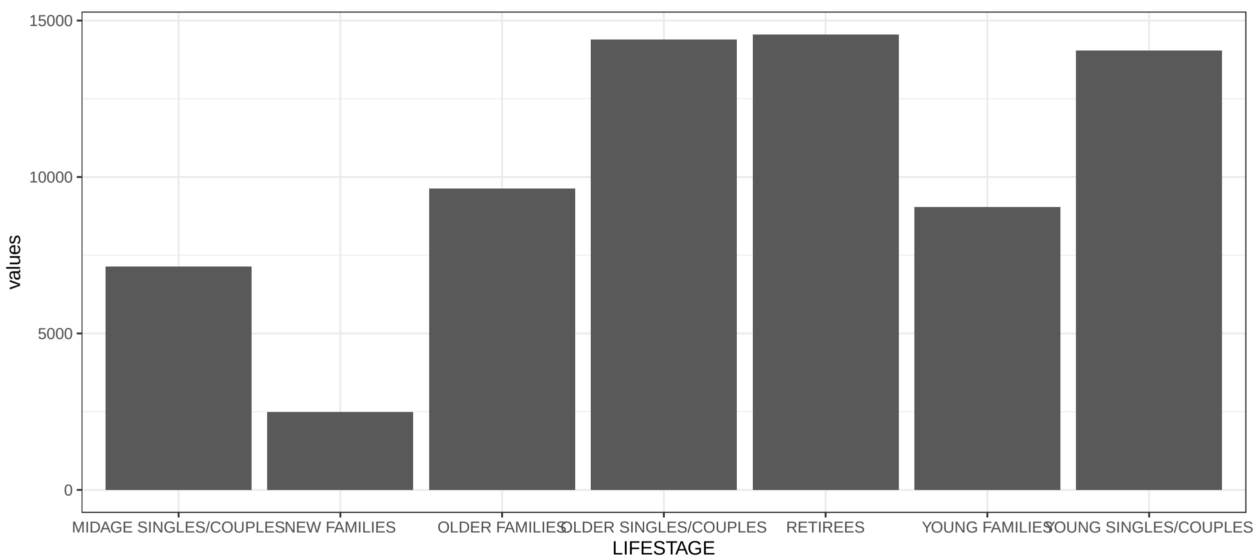

```
ggplot(total_lifestage, aes(x = val, y = x)) + geom_bar(stat="identity")
```



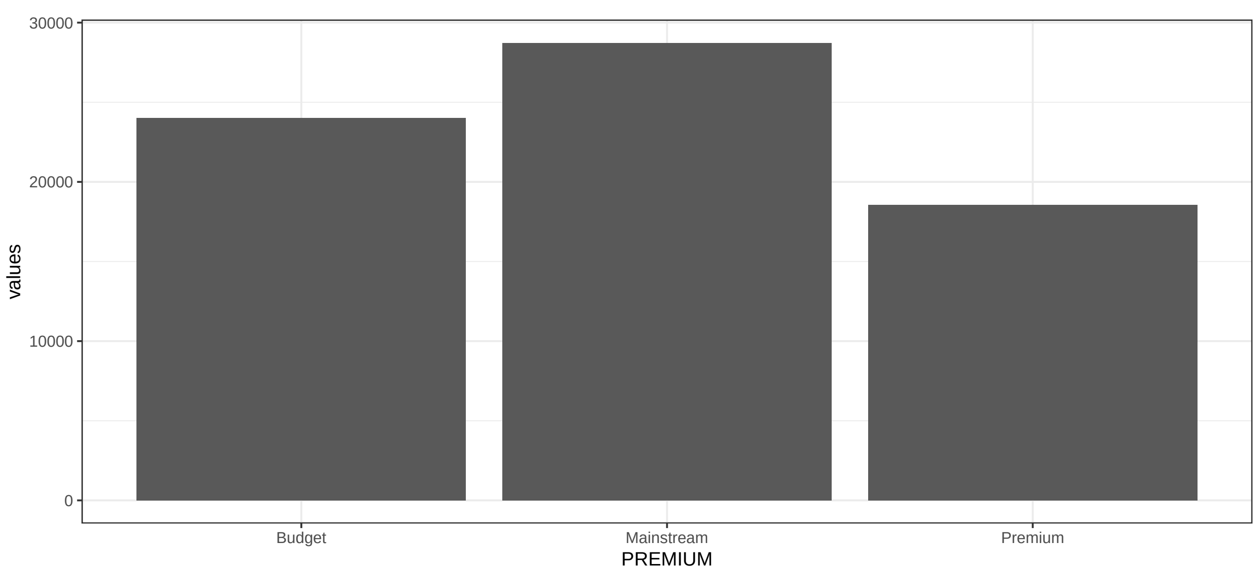
Sales are coming mainly from Budget - older families, Mainstream - young singles/couples, and Mainstream - retirees Let's see if the higher sales are due to there being more customers who buy chips.

```
#### Number of customers by LIFESTAGE and PREMIUM_CUSTOMER
total_customer_lifestage = aggregate(x=data$LYLTY_CARD_NBR, by=list(val =data$LIFESTAGE),
↪  FUN = unique)

total_customer_premium = aggregate(x=data$LYLTY_CARD_NBR, by=list(val
↪  =data$PREMIUM_CUSTOMER), FUN = unique)
total_customer_lifestage = data.frame("LIFESTAGE" = total_customer_lifestage$val,
↪  "values" = c(7141, 2492,9630,14389,14555,9036,14044))
total_customer_premium = data.frame("PREMIUM" = total_customer_premium$val, "values" =
↪  c(24006,28734,18547))
ggplot(total_customer_lifestage, aes(x =LIFESTAGE, y = values)) +
↪  geom_bar(stat="identity")
```




```
ggplot(total_customer_premium, aes(x = PREMIUM, y = values)) + geom_bar(stat="identity")
```

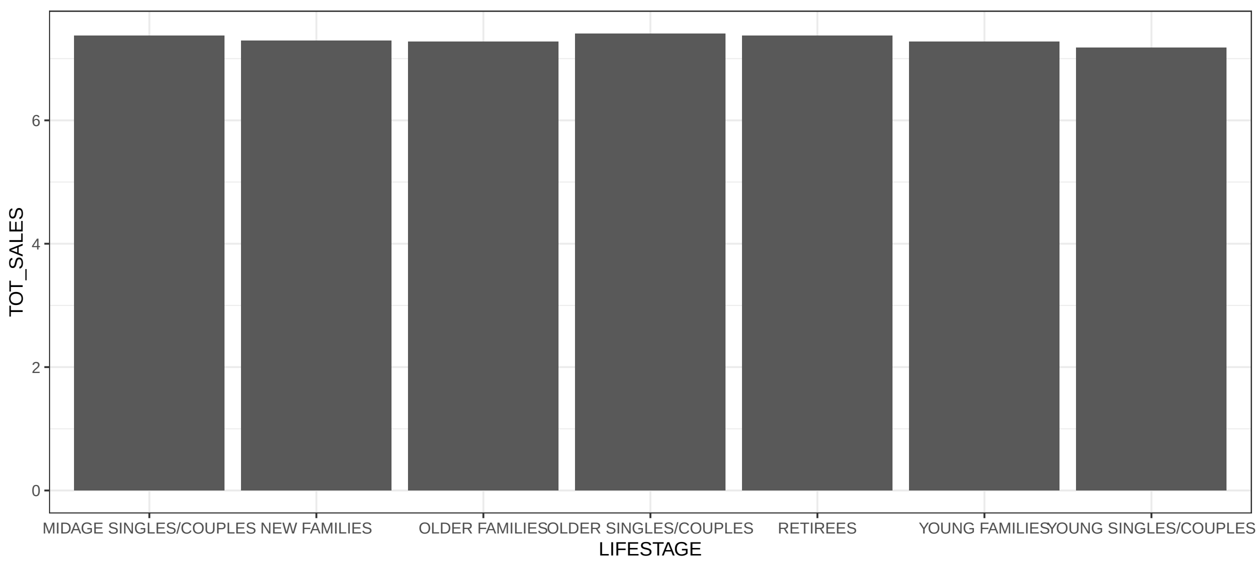


There are more Mainstream - young singles/couples and Mainstream - retirees who buy chips. This contributes to there being more sales to these customer segments but this is not a major driver for the Budget - Older families segment. Higher sales may also be driven by more units of chips being bought per customer. Let's have a look at this next.

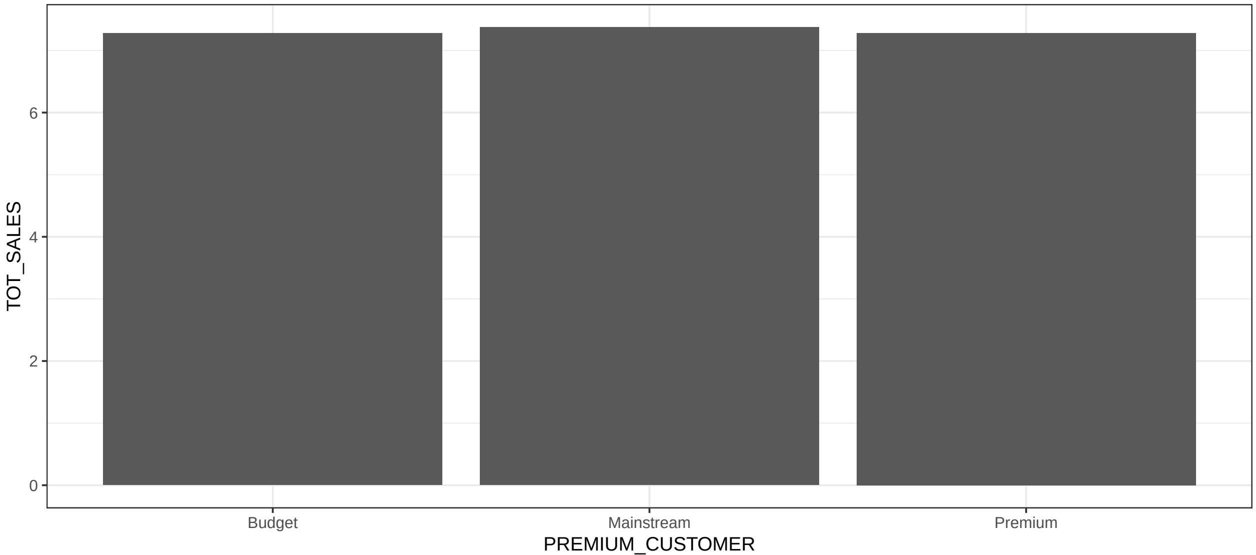
```r
#### Average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER
average_per_customer_lifestage = summarise_at(group_by(data, LIFESTAGE, LYLTY_CARD_NBR),
↪   vars(TOT_SALES), funs(mean))
```

```
## Warning: `funs()` is deprecated as of dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```r
average_per_customer_lifestage = summarise_at(group_by(data, LIFESTAGE), vars(TOT_SALES),
↪   funs(mean))
ggplot(average_per_customer_lifestage, aes(x = LIFESTAGE, y = TOT_SALES)) +
↪   geom_bar(stat="identity")
```




```r
average_per_customer_premium = summarise_at(group_by(data, PREMIUM_CUSTOMER,
↪   LYLTY_CARD_NBR), vars(TOT_SALES), funs(mean))
average_per_customer_premium = summarise_at(group_by(data, PREMIUM_CUSTOMER),
↪   vars(TOT_SALES), funs(mean))
ggplot(average_per_customer_premium, aes(x = PREMIUM_CUSTOMER, y = TOT_SALES)) +
↪   geom_bar(stat="identity")
```



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
# Over to you! Perform a t-test to see if the difference is significant.
mainstream = data[((data$LIFESTAGE == "MIDAGE SINGLES/COUPLES") | (data$LIFESTAGE ==
↪  "YOUNG SINGLES/COUPLES")) &  (data$PREMIUM_CUSTOMER == 'Mainstream')]
premium = data[((data$LIFESTAGE == "MIDAGE SINGLES/COUPLES") | (data$LIFESTAGE == "YOUNG
↪  SINGLES/COUPLES")) &  (data$PREMIUM_CUSTOMER == 'Premium')]


t.test(premium$TOT_SALES, mainstream$TOT_SALES)
```

```
##
##  Welch Two Sample t-test
##
## data:  premium$TOT_SALES and mainstream$TOT_SALES
## t = -24.24, df = 24455, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.6898246 -0.5866125
## sample estimates:
## mean of x mean of y
##  6.944158  7.582377
```

The t-test results in a p-value of ≈ 0, i.e. the unit price for mainstream, young and mid-age singles and couples ARE significantly higher than that of budget or premium, young and midage singles and couples. ## Deep dive into specific customer segments for insights We have found quite a few interesting insights that we can dive deeper into. We might want to target customer segments that contribute the most to sales to retain them or further increase sales. Let's look at Mainstream - young singles/couples. For instance, let's find out if they tend to buy a particular brand of chips.

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

new = select(mainstream, c('LYLTY_CARD_NBR','PROD_QTY', 'BRAND'))
#affinity analysis
rules <- apriori(new, parameter = list(supp = 0.001, conf = 0.8))
```

```
## Warning: Column(s) 1, 2, 3 not logical or factor. Applying default
## discretization (see '? discretizeDF').

## Warning in discretize(x = c(1L, 1L, 2L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, : The calculated breaks are:
##   Only unique breaks are used reducing the number of intervals. Look at ? discretize for details.

## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    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: 30
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[28 item(s), 30639 transaction(s)] done [0.01s].
## sorting and recoding items ... [28 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [87 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
# Show the top 5 rules, but only 2 digits
rules<-sort(rules, by="confidence", decreasing=TRUE)
options(digits=2)
inspect(rules[1:5])
```

```
## lhs rhs support confidence coverage lift count
## [1] {LYLTY_CARD_NBR=[1.79e+05,2.37e+06],
## BRAND=Smith} => {PROD_QTY=[2,5]} 0.0021 0.93 0.0023 1.1 65
## [2] {LYLTY_CARD_NBR=[1.79e+05,2.37e+06],
## BRAND=Dorito} => {PROD_QTY=[2,5]} 0.0049 0.92 0.0053 1.1 150
## [3] {LYLTY_CARD_NBR=[8.73e+04,1.79e+05),
## BRAND=French} => {PROD_QTY=[2,5]} 0.0014 0.92 0.0016 1.1 44
## [4] {LYLTY_CARD_NBR=[8.73e+04,1.79e+05),
## BRAND=Infuzions} => {PROD_QTY=[2,5]} 0.0189 0.91 0.0208 1.0 578
## [5] {LYLTY_CARD_NBR=[1e+03,8.73e+04),
## BRAND=Cheezels} => {PROD_QTY=[2,5]} 0.0060 0.90 0.0067 1.0 184
```

```r
new = select(premium, c('LYLTY_CARD_NBR','PROD_QTY', 'BRAND'))
#affinity analysis
rules <- apriori(new, parameter = list(supp = 0.001, conf = 0.8))
```

```
## Warning: Column(s) 1, 2, 3 not logical or factor. Applying default
## discretization (see '? discretizeDF').

## Warning in discretize(x = c(2L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, : The calculated breaks are:
##   Only unique breaks are used reducing the number of intervals. Look at ? discretize for details.

## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    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: 13
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[28 item(s), 13464 transaction(s)] done [0.00s].
## sorting and recoding items ... [28 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [65 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
# Show the top 5 rules, but only 2 digits
rules<-sort(rules, by="confidence", decreasing=TRUE)
options(digits=2)
inspect(rules[1:5])
```

```
## lhs rhs support confidence coverage lift count
## [1] {LYLTY_CARD_NBR=[9.31e+04,1.82e+05),
## BRAND=Pringles} => {PROD_QTY=[2,5]} 0.0295 0.92 0.0322 1.1 397
## [2] {LYLTY_CARD_NBR=[1.82e+05,2.33e+06],
## BRAND=Doritos} => {PROD_QTY=[2,5]} 0.0240 0.92 0.0262 1.1 323
## [3] {LYLTY_CARD_NBR=[1.82e+05,2.33e+06],
## BRAND=Twisties} => {PROD_QTY=[2,5]} 0.0108 0.91 0.0119 1.1 146
## [4] {LYLTY_CARD_NBR=[1e+03,9.31e+04),
## BRAND=Dorito} => {PROD_QTY=[2,5]} 0.0037 0.91 0.0041 1.1 50
## [5] {LYLTY_CARD_NBR=[1.82e+05,2.33e+06],
## BRAND=Cheezels} => {PROD_QTY=[2,5]} 0.0058 0.90 0.0065 1.1 78
```

```r
# Over to you! Work out of there are brands that these two customer segments prefer more
↪  than others. You could use a technique called affinity analysis or a-priorianalysis
↪  (or any other method if you prefer)
```

We can see that : For premium customers the top 3 brands are Pringles, Doritos, Twisties For mainstream customers the top 3 brands are Smith, Doritos, French Let's also find out if our target segment tends to buy larger packs of chips.

```r
#### Preferred pack size compared to the rest of the population
new = select(mainstream, c('LYLTY_CARD_NBR','PROD_QTY', 'PACK_SIZE'))
rules <- apriori(new, parameter = list(supp = 0.001, conf = 0.8))
```

```
## Warning: Column(s) 1, 2, 3 not logical or factor. Applying default
## discretization (see '? discretizeDF').

## Warning in discretize(x = c(1L, 1L, 2L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, : The calculated breaks are:
##   Only unique breaks are used reducing the number of intervals. Look at ? discretize for details.

## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    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: 30
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[25 item(s), 30639 transaction(s)] done [0.01s].
## sorting and recoding items ... [25 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [73 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules<-sort(rules, by="confidence", decreasing=TRUE)
options(digits=2)
inspect(rules[1:5])
```

```
## lhs rhs support confidence coverage lift count
## [1] {LYLTY_CARD_NBR=[1.79e+05,2.37e+06],
## PACK_SIZE=380} => {PROD_QTY=[2,5]} 0.0091 0.93 0.0098 1.1 279
## [2] {LYLTY_CARD_NBR=[8.73e+04,1.79e+05),
## PACK_SIZE=70} => {PROD_QTY=[2,5]} 0.0012 0.90 0.0014 1.0 38
## [3] {LYLTY_CARD_NBR=[8.73e+04,1.79e+05),
## PACK_SIZE=110} => {PROD_QTY=[2,5]} 0.0305 0.90 0.0339 1.0 933
## [4] {LYLTY_CARD_NBR=[8.73e+04,1.79e+05),
## PACK_SIZE=160} => {PROD_QTY=[2,5]} 0.0022 0.89 0.0025 1.0 68
## [5] {LYLTY_CARD_NBR=[8.73e+04,1.79e+05),
## PACK_SIZE=190} => {PROD_QTY=[2,5]} 0.0028 0.89 0.0031 1.0 85
```

```
new = select(data[!mainstream], c('LYLTY_CARD_NBR','PROD_QTY', 'PACK_SIZE'))
rules <- apriori(new, parameter = list(supp = 0.001, conf = 0.8))
```

```
## Warning: Column(s) 1, 2, 3 not logical or factor. Applying default
## discretization (see '? discretizeDF').

## Warning in discretize(x = c(2L, 1L, 1L, 1L, 1L, 1L, 1L, 2L, 1L, 1L, 1L, : The calculated breaks are:
##   Only unique breaks are used reducing the number of intervals. Look at ? discretize for details.

## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
```

```
##         0.8    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: 216
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[25 item(s), 216101 transaction(s)] done [0.04s].
## sorting and recoding items ... [25 item(s)] done [0.00s].
## creating transaction tree ... done [0.04s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [84 rule(s)] done [0.00s].
## creating S4 object  ... done [0.02s].
```

```
rules<-sort(rules, by="confidence", decreasing=TRUE)
options(digits=2)
inspect(rules[1:5])
```

```
## lhs rhs support confidence coverage lift count
## [1] {LYLTY_CARD_NBR=[9.3e+04,1.79e+05),
## PACK_SIZE=135} => {PROD_QTY=[2,5]} 0.0039 0.92 0.0042 1 838
## [2] {LYLTY_CARD_NBR=[9.3e+04,1.79e+05),
## PACK_SIZE=250} => {PROD_QTY=[2,5]} 0.0038 0.92 0.0041 1 811
## [3] {LYLTY_CARD_NBR=[9.3e+04,1.79e+05),
## PACK_SIZE=270} => {PROD_QTY=[2,5]} 0.0070 0.92 0.0076 1 1506
## [4] {LYLTY_CARD_NBR=[1.79e+05,2.37e+06],
## PACK_SIZE=270} => {PROD_QTY=[2,5]} 0.0076 0.92 0.0083 1 1651
## [5] {LYLTY_CARD_NBR=[1e+03,9.3e+04),
## PACK_SIZE=110} => {PROD_QTY=[2,5]} 0.0284 0.92 0.0311 1 6145
```

Most preferred packet size by mainstream young & midage singles/couples is 380g whereas for the rest of population the most preferred packet size is 135g