task1_solution.R

User

2025-06-13

```
# set options for R markdown knitting
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(linewidth=80)

# set up line wrapping in MD knit output
library(knitr)
hook_output = knit_hooks$get("output")
knit_hooks$set(output = function(x, options)
{
  # this hook is used only when the line width option is not NULL
  if (!is.null(n <- options$linewidth))
  {
    x = knitr:::split_lines(x)
    # any lines wider than n should be wrapped
    if (any(nchar(x) > n))
      x = strwrap(x, width = n)
    x = paste(x, collapse = "\n")
  }
  hook_output(x, options)
})

options(repos = c(CRAN = "https://cran.rstudio.com/"))

#### Install packages
install.packages("data.table")
```

```
## Installing package into 'C:/Users/User/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)

## package 'data.table' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'data.table'

## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
## C:\Users\User\AppData\Local\R\win-library\4.4\00LOCK\data.table\libs\x64\data_table.dll
## to
## C:\Users\User\AppData\Local\R\win-library\4.4\data.table\libs\x64\data_table.dll:
## Permission denied

## Warning: restored 'data.table'
```

```
##
## The downloaded binary packages are in
##  C:\Users\User\AppData\Local\Temp\RtmpqKuKfs\downloaded_packages
```

```
# install.packages("tinytex")
# tinytex::install_tinytex(force = TRUE)

# --- Step 1: Load required libraries ---
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.4.3
```

```
library(ggplot2)
library(ggmosaic)
```

```
## Warning: package 'ggmosaic' was built under R version 4.4.3
```

```
library(readr)

# Setting plot themes to format graphs
theme_set(theme_bw())

#### Point the filePath to where you have downloaded the datasets to and
#### assign the data files to data.tables
filePath <- "C:/Users/User/OneDrive - Swinburne University/Desktop/forage/quantium/"
transactionData <- fread(paste0(filePath,"QVI_transaction_data.csv"))
customerData <- fread(paste0(filePath,"QVI_purchase_behaviour.csv"))

class(transactionData)
```

```
## [1] "data.table" "data.frame"
```

```
# --- Step 2: Exploratory data analysis ---
# --- 2.1 Examining transaction data ---
#### (1) Examining transaction data
str(transactionData) # check format of each column & see a sample of the data
```

```
## 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>
```

```
head(transactionData) # look at the first 6 rows
```

```
##     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
```

```
### Convert DATE column from an integer to a date format
transactionData$DATE <- as.Date(transactionData$DATE, origin = "1899-12-30")
# CSV and Excel integer dates begin on 30 Dec 1899

#### (2) Examining PROD_NAME, to see if we are looking for "chip"
colnames(transactionData)
```

```
## [1] "DATE"           "STORE_NBR"      "LYLTY_CARD_NBR" "TXN_ID"
## [5] "PROD_NBR"       "PROD_NAME"      "PROD_QTY"       "TOT_SALES"
```

```
# Count occurrences of each product name
transactionData[, .N, by = 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
```

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

# Keep only product words that relate to chip
```

```
# Remove digits
productWords <- productWords[grepl("\\d", words) == FALSE, ]
# Remove special characters
productWords <- productWords[grepl("[:alpha:]", words), ]
# Find the most common words
productWords[, .N, words][order(N, decreasing = TRUE)]
```

```
##             words     N
##            <char> <int>
##   1:        Chips    21
##   2:       Smiths    16
##   3:      Crinkle    14
##   4:       Kettle    13
##   5:       Cheese    12
##  ---
## 127: Chikn&Garlic     1
## 128:        Aioli     1
## 129:         Slow     1
## 130:        Belly     1
## 131:    Bolognese     1
```

```
# Remove salsa products
# because the word "salsa" appears in the PROD_NAME column of the transactionData
transactionData[, SALSA := grepl("salsa", tolower(PROD_NAME))]
transactionData <- transactionData[SALSA == FALSE, ][, SALSA := NULL]

#### (3) Check summary statistics 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 the product quantity of 200 packets
  # appears to be an outlier due to its significantly higher value compared to
  # the mean of 1.907 and the third quartile of 2.000, indicating an unusually
  # large transaction that warrants further investigation.

# Filter the dataset to find the outlier
transactionData[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
```

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

# Check if the customer has had other transactions
transactionData[LYLTY_CARD_NBR == 226000, ]
```

```
##          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
```

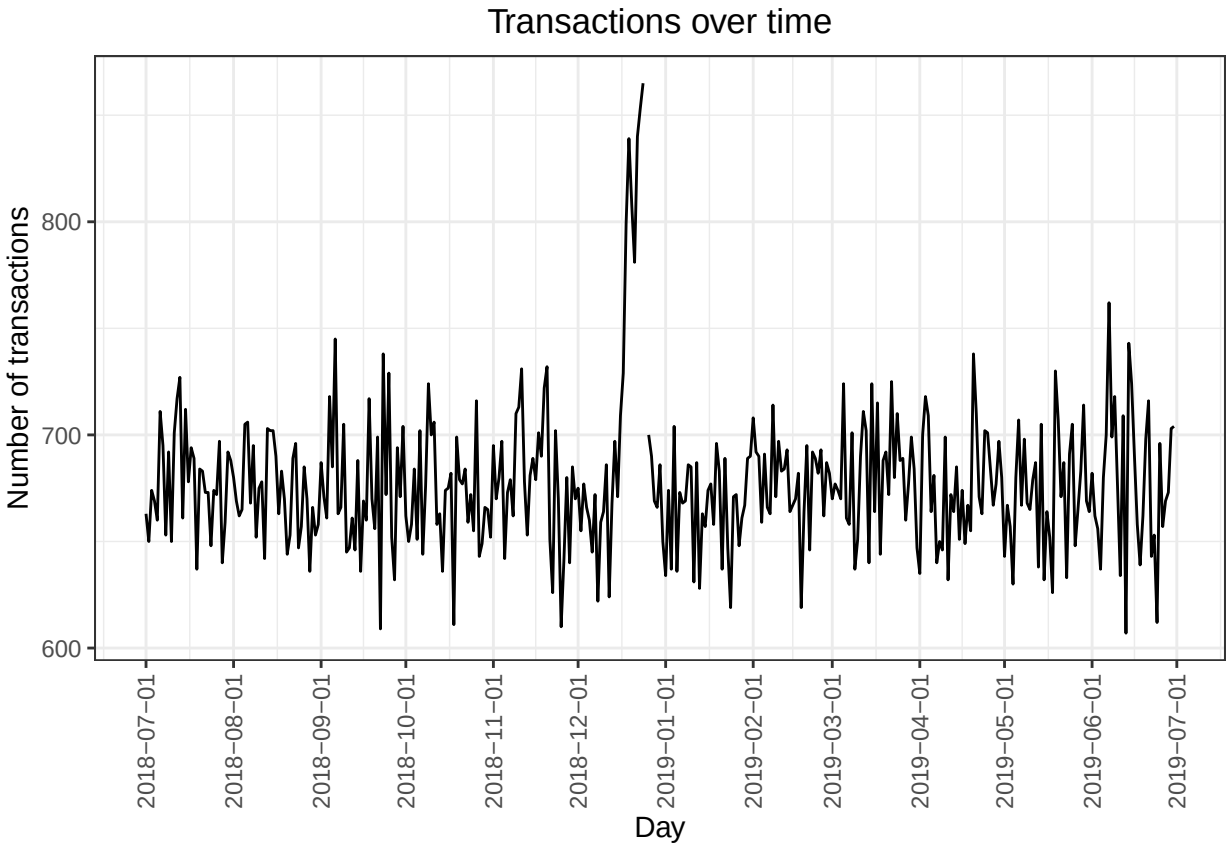
```
# It looks like this customer has only had the two transactions over the year
  # and is not an ordinary retail customer. The customer might be buying chips
  # for commercial purposes instead. We'll remove this loyalty card number from further analysis.

# Filter out the customer based on the loyalty card number
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
```

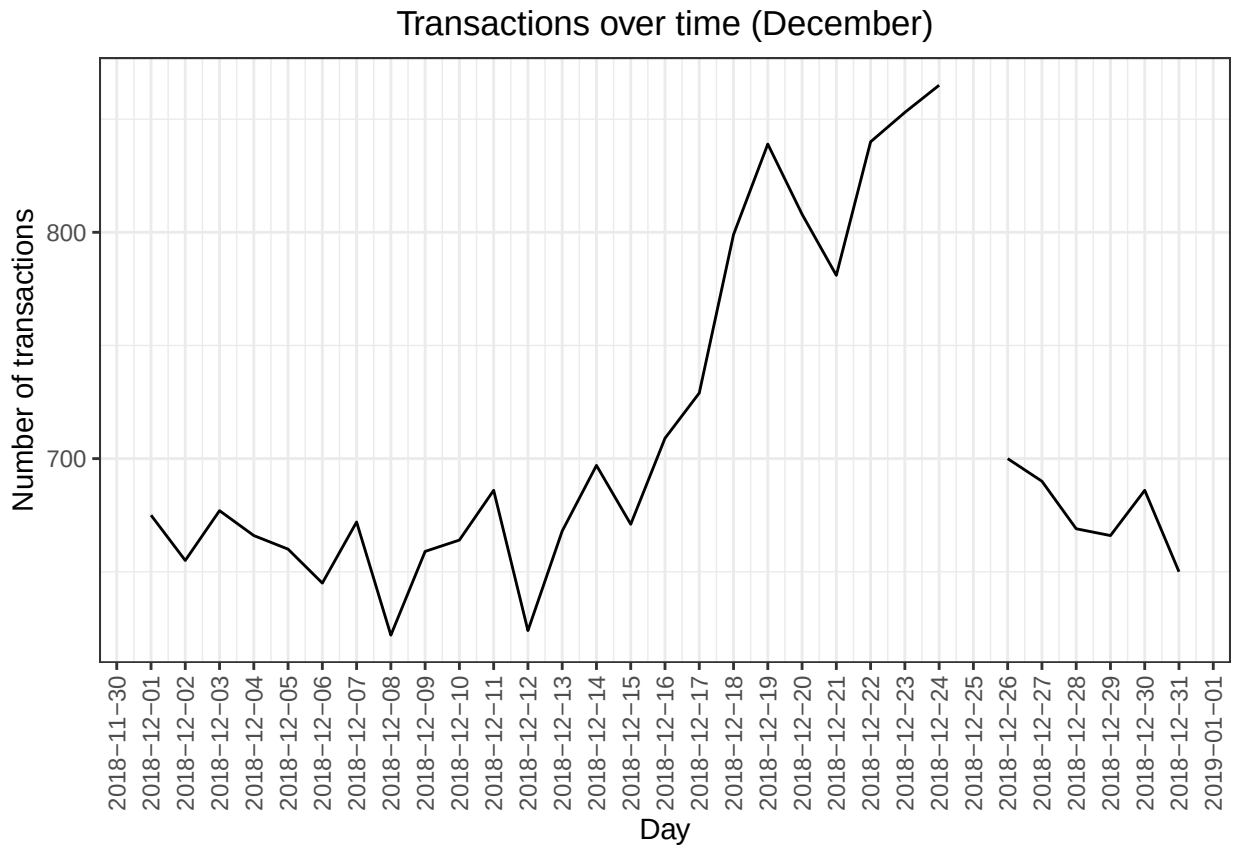
```
# Check the number of transaction lines over time to see if there are any
  # obvious data issues such as missing data.

# 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
# There are only 364 dates, which indicates a missing date.
# 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.
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))

# 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))
```




```
# There is an increase in purchases in December and a break in 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 (December)") +
  scale_x_date(breaks = "1 day") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```




```
# The increase in sales occurs in the lead-up to Christmas and that there are zero sales
  # on Christmas day itself due to the shop close.
# So no more outliers, can continue on create other features

#### (4) Create pack size from PROD_NAME
# Taking the digits that are in PROD_NAME
transactionData[,PACK_SIZE:= parse_number(PROD_NAME)]
# check if the output look sensible
transactionData[,.N,PACK_SIZE][order(PACK_SIZE)]
```

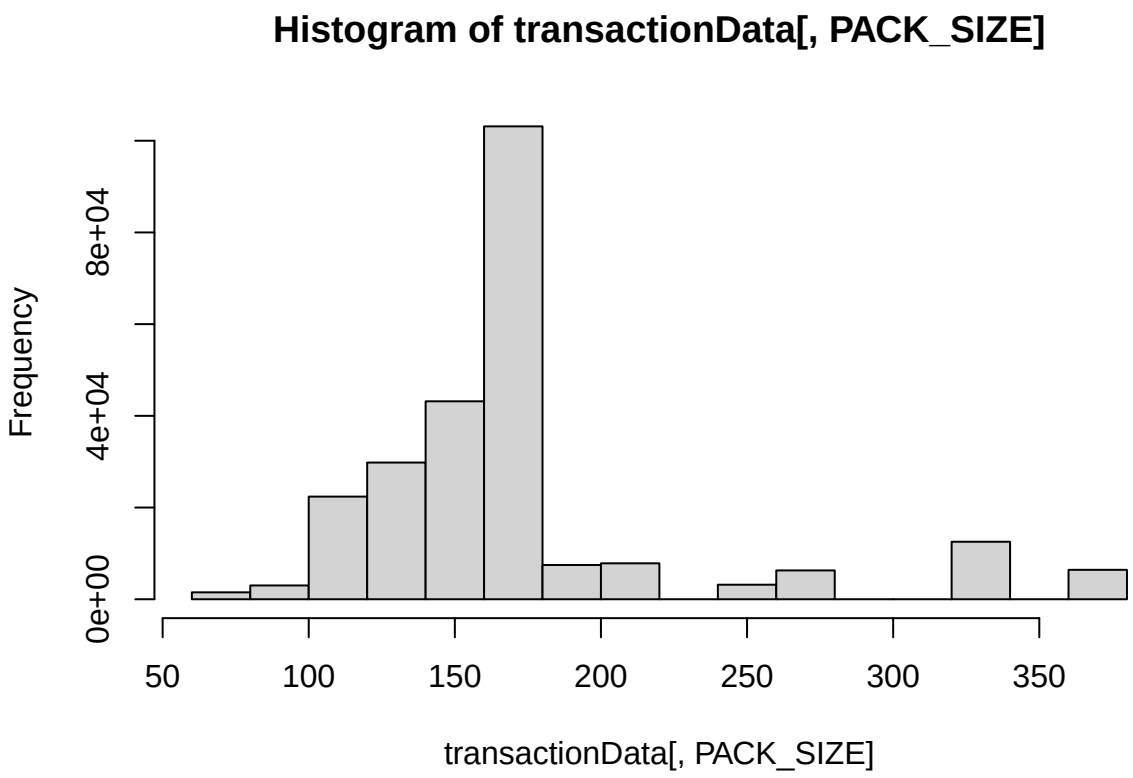
```
##     PACK_SIZE     N
##         <num> <int>
##  1:        70  1507
##  2:        90  3008
##  3:       110 22387
##  4:       125  1454
##  5:       134 25102
##  6:       135  3257
##  7:       150 40203
##  8:       160  2970
##  9:       165 15297
## 10:       170 19983
## 11:       175 66390
## 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
```

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

# Check the output of the first few rows to see if we have indeed pickedout packsize.
transactionData
```

```
##               DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##             <Date>     <int>          <int>  <int>    <int>
##      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
##     ---
## 246736: 2019-03-09       272         272319 270088       89
## 246737: 2018-08-13       272         272358 270154       74
## 246738: 2018-11-06       272         272379 270187       51
## 246739: 2018-12-27       272         272379 270188       42
## 246740: 2018-09-22       272         272380 270189       74
##                                       PROD_NAME PROD_QTY TOT_SALES PACK_SIZE
##                                          <char>    <int>     <num>     <num>
##      1:   Natural Chip        Compny SeaSalt175g        2       6.0       175
##      2:                 CCs Nacho Cheese    175g        3       6.3       175
##      3:   Smiths Crinkle Cut  Chips Chicken 170g        2       2.9       170
##      4:   Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0       175
##      5: Kettle Tortilla ChpsHny&Jlpno Chili 150g        3      13.8       150
##     ---
## 246736:  Kettle Sweet Chilli And Sour Cream 175g        2      10.8       175
## 246737:            Tostitos Splash Of  Lime 175g        1       4.4       175
## 246738:                 Doritos Mexicana    170g        2       8.8       170
## 246739:  Doritos Corn Chip Mexican Jalapeno 150g        2       7.8       150
## 246740:            Tostitos Splash Of  Lime 175g        2       8.8       175
```

```
# Plot a histogram of PACK_SIZE since we know that it is a categorical variable and
  # not a continuous variable eventhough it is numeric.
hist(transactionData[,PACK_SIZE])
```




```
# Pack sizes looks reasonable

#### (5) Create brand, using the first word in PROD_NAME to work out the brand name
# Brands
transactionData[, BRAND := toupper(substr(PROD_NAME, 1, regexpr(pattern = ' ', PROD_NAME) - 1))]
# 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
```

```
# 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 == "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
transactionData[, .N, by = BRAND][order(BRAND)]
```

```
##          BRAND     N
##         <char> <int>
##  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 14201
## 10:     KETTLE 41288
## 11:    NATURAL  7469
## 12:   PRINGLES 25102
## 13:        RRD 16321
## 14:     SMITHS 30353
## 15:   SUNBITES  3008
## 16:      THINS 14075
## 17:   TOSTITOS  9471
## 18:   TWISTIES  9454
## 19:   TYRRELLS  6442
## 20: WOOLWORTHS 11836
##          BRAND     N
```

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

# --- 2.2 Examining customer data ---
# Examining customer data
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>
```

```
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
```

```
# Examining the values of lifestage and premium_customer
customerData[, .N, by = LIFESTAGE][order(-N)]
```

```
##                 LIFESTAGE     N
##                    <char> <int>
## 1:               RETIREES 14805
## 2:  OLDER SINGLES/COUPLES 14609
## 3:  YOUNG SINGLES/COUPLES 14441
## 4:         OLDER FAMILIES  9780
## 5:         YOUNG FAMILIES  9178
## 6: MIDAGE SINGLES/COUPLES  7275
## 7:           NEW FAMILIES  2549
```

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

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

```
# As there are no issues with the customer data, we can now go ahead and join
  # the transaction  and customer data sets together
# Merge transaction data to customer data
data <- merge(transactionData, customerData, all.x = TRUE)

# Since 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 created data
```

```
  # by setting all.x = TRUE(inotherwords, aleftjoin) 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.

# Check if some customers were not matched on by checking for nulls.
data[is.null(LIFESTAGE), .N]
```

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

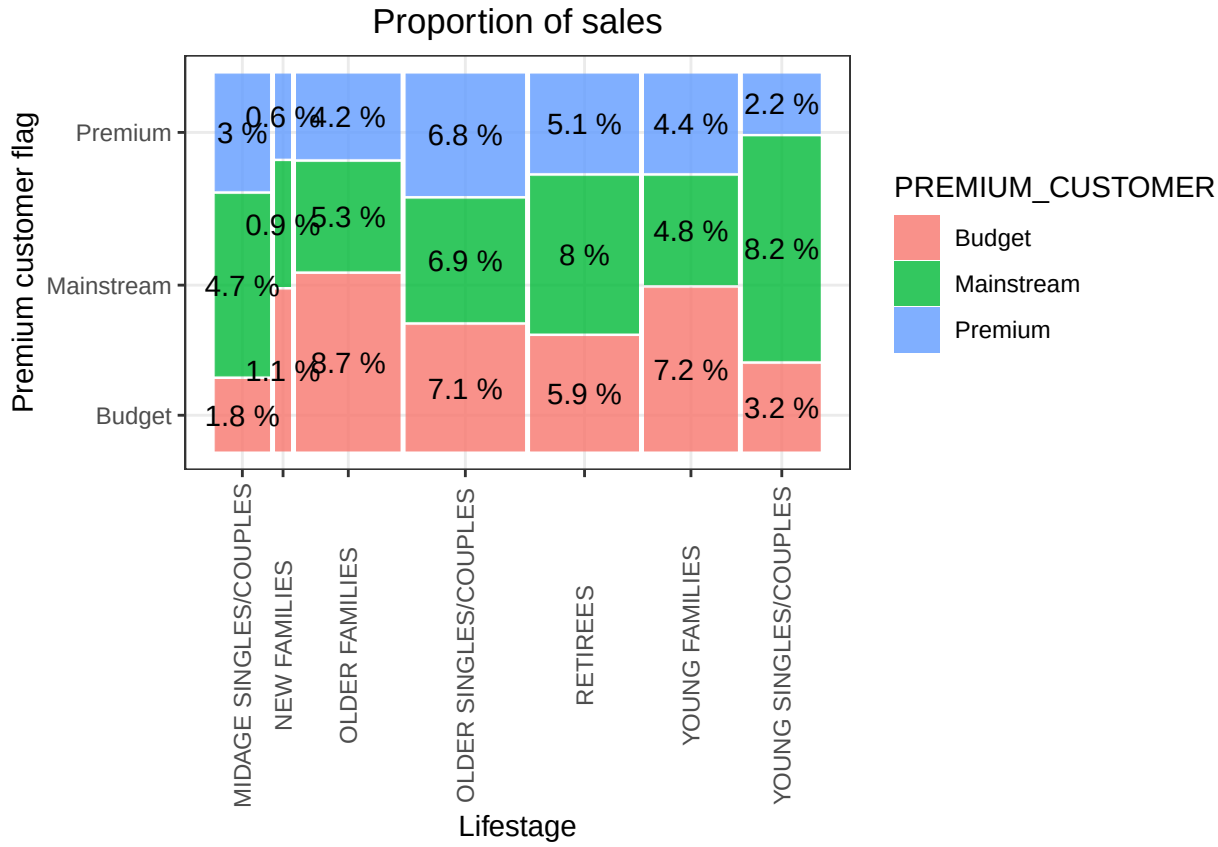
```
data[is.null(PREMIUM_CUSTOMER), .N]
```

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

```
# There are no nulls. So all our customers in the transaction data have been
  # accounted for in the customer dataset.
# fwrite(data, paste0(filePath,"QVI_data.csv"))

# --- Step 3: Data analysis on customer segments ---

#### Data analysis metrics for 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

#### (1) Describe which customer segment contribute most to chip sales
# Calculate total sales by LIFESTAGE and PREMIUM_CUSTOMER
sales <- data[, .(SALES = sum(TOT_SALES)), .(LIFESTAGE, PREMIUM_CUSTOMER)]

# Create plot for proportion of sales by LIFESTAGE and PREMIUM_CUSTOMER
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, '%'))))
```

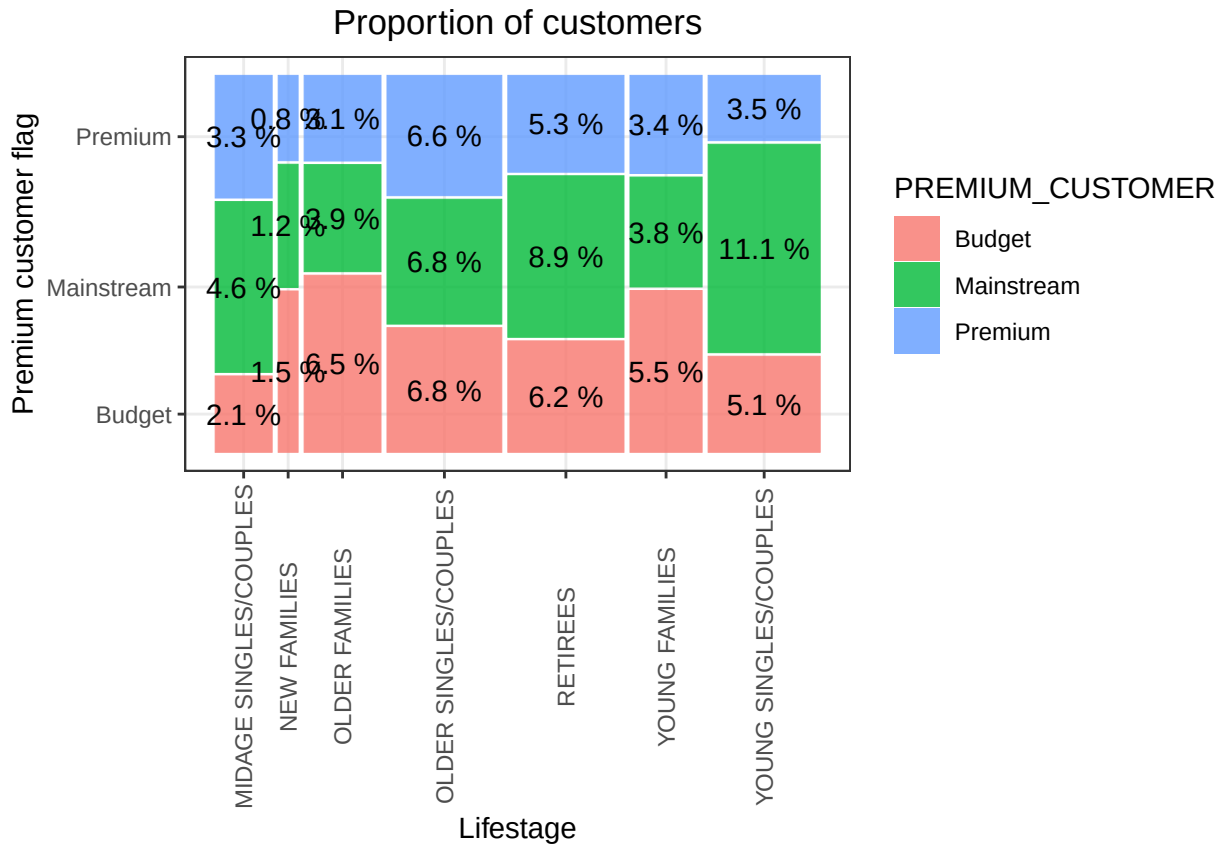
```
## Warning: The ‘scale_name‘ argument of ‘continuous_scale()‘ is deprecated as of ggplot2
## 3.5.0.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
## 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.
```

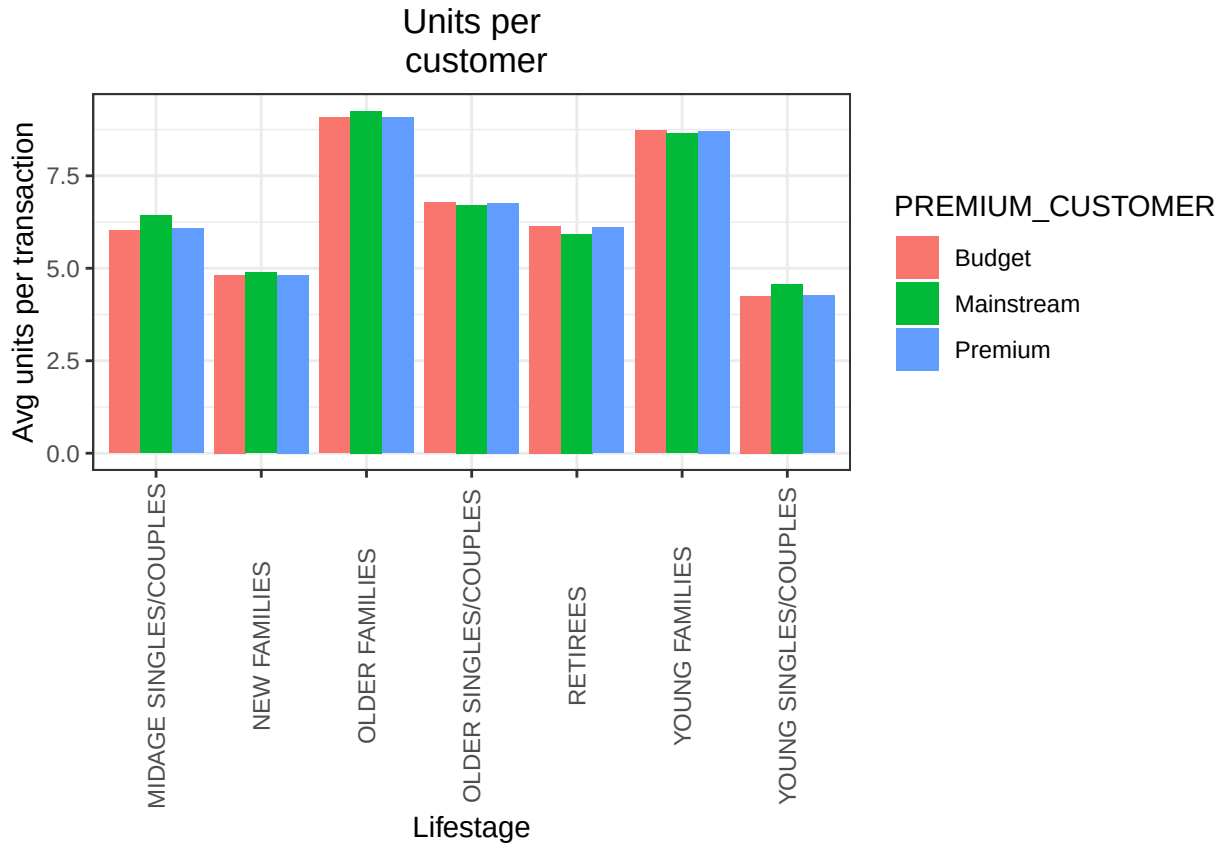



```
# Sales are coming mainly from Budget-olderfamilies,Mainstream-youngsingles/couples,
  # and Mainstream-retirees

## (2) Check if the higher sales are due to there being more customers who buy chips.
# Number of customers by LIFESTAGE and PREMIUM_CUSTOMER
customers <- data[, .(CUSTOMERS = uniqueN(LYLTY_CARD_NBR)),
                  .(LIFESTAGE, PREMIUM_CUSTOMER)][order(-CUSTOMERS)]

# Create plot for proportion of customers
p <- ggplot(data = customers) +
  geom_mosaic(aes(weight = CUSTOMERS, x = product(PREMIUM_CUSTOMER, LIFESTAGE),
```

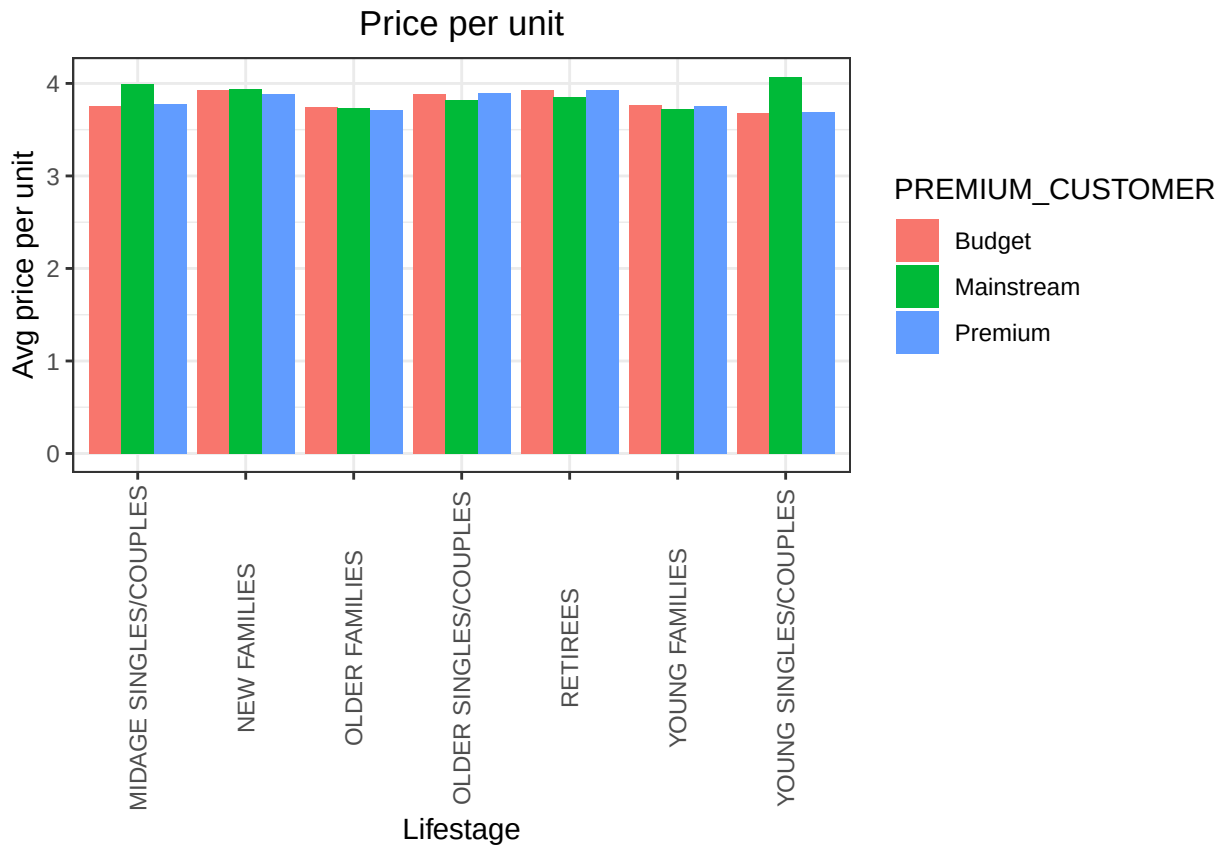
```
                  fill = PREMIUM_CUSTOMER)) +
  labs(x = "Lifestage", y = "Premium customer flag", title = "Proportion of customers") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))

# 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, '%'))))
```




```
# 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.

## (3) 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 is also a driver of total sales.

## (4) 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 mid-aged and young singles and couples pay more per chip packet.
# This may be due to premium shoppers prioritizing healthy snacks.
# They buy chips mainly for entertainment, not consumption.
# This is also supported by fewer premium mid-aged and young singles/couples
  # buying chips compared to mainstream.
# As the average price per unit difference is small, a statistical test
  # is needed to confirm its significance.

## (5) 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 == "
       data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & PREMIUM_CUSTOMER != "
       alternative = "greater")
```

```
##
##  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
```

```
# The t-test results in a p-value < 2.2e-16, 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.

# --- Step 4: Deep dive into specific customer segments for insights ---
# 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"),]
#### (1) 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 targetSegment       other affinityToBrand
##         <char>         <num>       <num>           <num>
##  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:   PRINGLES   0.119420290 0.100634769       1.1866703
##  7:       COBS   0.044637681 0.039048861       1.1431238
##  8:  INFUZIONS   0.064679089 0.057064679       1.1334347
##  9:      THINS   0.060372671 0.056986370       1.0594230
## 10:    GRNWVES   0.032712215 0.031187957       1.0488733
## 11:   CHEEZELS   0.017971014 0.018646902       0.9637534
## 12:     SMITHS   0.096369910 0.124583692       0.7735355
## 13:     FRENCH   0.003947550 0.005758060       0.6855694
## 14:    CHEETOS   0.008033126 0.012066591       0.6657329
## 15:        RRD   0.043809524 0.067493678       0.6490908
## 16:    NATURAL   0.019599724 0.030853989       0.6352412
## 17:        CCS   0.011180124 0.018895650       0.5916771
## 18:   SUNBITES   0.006349206 0.012580210       0.5046980
## 19: WOOLWORTHS   0.024099379 0.049427188       0.4875733
## 20:     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 youngsingles/couples are 56% less likely to purchase Burger Rings
  # compared to the rest of the population

#### (2)  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_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
```

```
#  It looks like Mainstream young singles/couples are 27% more likely to purchase
  # a 270g pack of chips com pared to the rest of the population but let's dive
  # into what brands sell this pack size.
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
```

```
# --- Step 5: Conclusion ---
# Chip sales are primarily driven by Budget - older families, Mainstream - young singles/couples,
# and Mainstream - retirees. High spend from mainstream young singles/couples and retirees is largely
# due to their higher numbers. Mainstream mid-age and young singles/couples also show impulse buying be
# paying more per chip packet. Notably, Mainstream young singles/couples
# are 23% more likely to buy Tyrrells chips.

# Recommendation: The Category Manager should increase visibility and impulse purchases
# by strategically placing Tyrrells and smaller chip packs in areas frequented
# by young singles/couples. Quantium can assist with segment location and trial impact measurement.
```