Quantinum Task_1 & Task_2

Shreyash S Sahare

2025-04-08

Contents

1 TASK 1

1.1 Task Details

1.2 Loading Libraries and Data Sets

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
```

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

```r
library(skimr)
library(janitor)
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```r
library(dplyr)
library(ggplot2)
library(lubridate)
installed.packages("data.table")
```

```
##      Package LibPath Version Priority Depends Imports LinkingTo Suggests
##      Enhances License License_is_FOSS License_restricts_use OS_type Archs
##      MD5sum NeedsCompilation Built
```

```r
installed.packages("ggmosaic")
```

```
##      Package LibPath Version Priority Depends Imports LinkingTo Suggests
##      Enhances License License_is_FOSS License_restricts_use OS_type Archs
##      MD5sum NeedsCompilation Built
```

```r
installed.packages("readr")
```

```
##      Package LibPath Version Priority Depends Imports LinkingTo Suggests
##      Enhances License License_is_FOSS License_restricts_use OS_type Archs
##      MD5sum NeedsCompilation Built
```

```r
library(data.table)
```

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

```r
library(ggmosaic)
```

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

```r
library(readr)
installed.packages("formatR")
```

```
##      Package LibPath Version Priority Depends Imports LinkingTo Suggests
##      Enhances License License_is_FOSS License_restricts_use OS_type Archs
##      MD5sum NeedsCompilation Built
```

```r
library(formatR)
```

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

```r
QVI_PB <- read.csv("QVI_purchase_behaviour.csv")
QVI_TD <- readxl::read_xlsx("QVI_transaction_data.xlsx")
```

1.3 Explotary Analysis on Transaction Data :-

In this analysis, the first step was to understand the data. I began by examining the datasets provided.

To get a clear picture, I used the str() function to inspect the format of each column and view a sample of the data. Alternatively, I used the head() function to display the first 10 rows.

My goal was to confirm that columns expected to contain numerical values were indeed in numeric format, and that columns representing dates were correctly formatted as dates.

```r
str(QVI_PB)
```

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

```r
head(QVI_PB)
```

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

```r
str(QVI_TD)
```

```
## tibble [264,836 x 8] (S3: tbl_df/tbl/data.frame)
##  $ DATE          : num [1:264836] 43390 43599 43605 43329 43330 ...
##  $ STORE_NBR     : num [1:264836] 1 1 1 2 2 4 4 4 5 7 ...
##  $ LYLTY_CARD_NBR: num [1:264836] 1000 1307 1343 2373 2426 ...
##  $ TXN_ID        : num [1:264836] 1 348 383 974 1038 ...
##  $ PROD_NBR      : num [1:264836] 5 66 61 69 108 57 16 24 42 52 ...
##  $ PROD_NAME     : chr [1:264836] "Natural Chip        Compny SeaSalt175g" "CCs Nacho Cheese    175g
##  $ PROD_QTY      : num [1:264836] 2 3 2 5 3 1 1 1 1 2 ...
##  $ TOT_SALES     : num [1:264836] 6 6.3 2.9 15 13.8 5.1 5.7 3.6 3.9 7.2 ...
```

```
head(QVI_TD)
```

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

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

1.3.1 Text Analysis

To determine if all products were indeed chips, I performed basic text analysis by summarizing the individual words in the product name.Since I was only interested in words that would tell me if the product is chips or not, I removed all words with digits and special characters such as '&' from the set of product words. Additionally, because there were salsa products in the dataset and the analysis was focused on the chips category, I removed those products as well.

```
class(QVI_TD)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
QVI_TD <- data.table(QVI_TD)
QVI_PB <- data.table(QVI_PB)
productWords <- data.table(unlist(strsplit(unique(QVI_TD[, PROD_NAME]),
    "
")))
setnames(productWords, "words")
View(productWords)

rows_to_clean <- grepl("[^a-zA-Z\\s]", productWords)
productWords[rows_to_clean, CLEANED_productWords := gsub("[^a-zA-Z\\s]",
    " ", words)]
productWords[!rows_to_clean, CLEANED_productWords := words]
View(productWords)

productWordsC <- productWords[, .(PROD_NAME = tolower(CLEANED_productWords))]
```

```
productWordsC <- productWordsC[, unlist(strsplit(PROD_NAME, " ")),
    by = PROD_NAME]
productWordsC <- productWordsC[V1 != ""]
productWordsC <- productWordsC[, .(count = .N), by = V1]  # Count word frequency
productWordsC <- productWordsC[order(-count)]  # Sort by frequency (descending)
setnames(productWordsC, "V1", "words")  #rename column
View(productWordsC)

QVI_TD[, SALSA := grepl("salsa", tolower(PROD_NAME))]
QVI_TD <- QVI_TD[SALSA == FALSE, ][, SALSA := NULL]
View(QVI_TD)
```

Next, I used the summary() function to check summary statistics, such as mean, min, and max values, for each feature. This allowed me to identify any obvious outliers in the data and check for null values in any of the columns (indicated by "NA" in the output, showing the number of nulls).

```
Summary <- QVI_TD %>%
    summarise(mean_prd_QTY = mean(PROD_QTY), min_prd_QTY = min(PROD_QTY),
        max_prd_QTY = max(PROD_QTY), mean_TOT_sales = mean(TOT_SALES),
        min_TOT_sales = min(TOT_SALES), max_TOT_sales = max(TOT_SALES),
        Total_QTY = sum(PROD_QTY), Total_sales = sum(TOT_SALES))
View(Summary)
```

There were no nulls in the columns, but product quantity appeared to have an outlier, which I investigated further. I specifically looked into the case where 200 packets of chips were bought in one transaction.

```
QVI_TD_outlier_OT <- QVI_TD %>%
    filter(LYLTY_CARD_NBR == 226000)

QVI_TD_RO <- QVI_TD %>%
    filter(LYLTY_CARD_NBR != 226000)
```

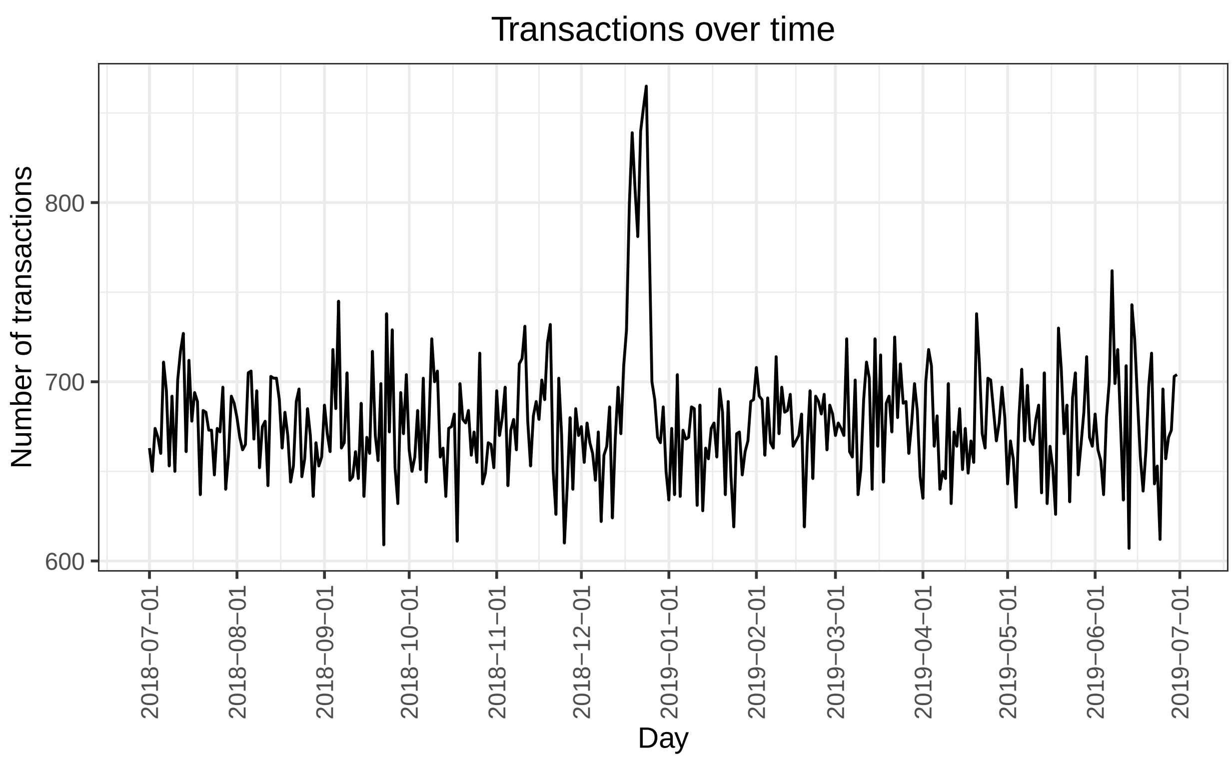
There were two transactions where 200 packets of chips were bought in one transaction, and both of these transactions were by the same customer. It looked like this customer had only had the two transactions over the year and was not an ordinary retail customer. The customer might have been buying chips for commercial purposes instead. I removed this loyalty card number from further analysis. That's better. Now, I'll look at the number of transaction lines over time to see if there are any obvious data issues such as missing data.

```
Tran_OVR_TIM <- QVI_TD_RO %>%
    group_by(DATE) %>%
    summarise(No_of_TRANS = n())
View(Tran_OVR_TIM)
```

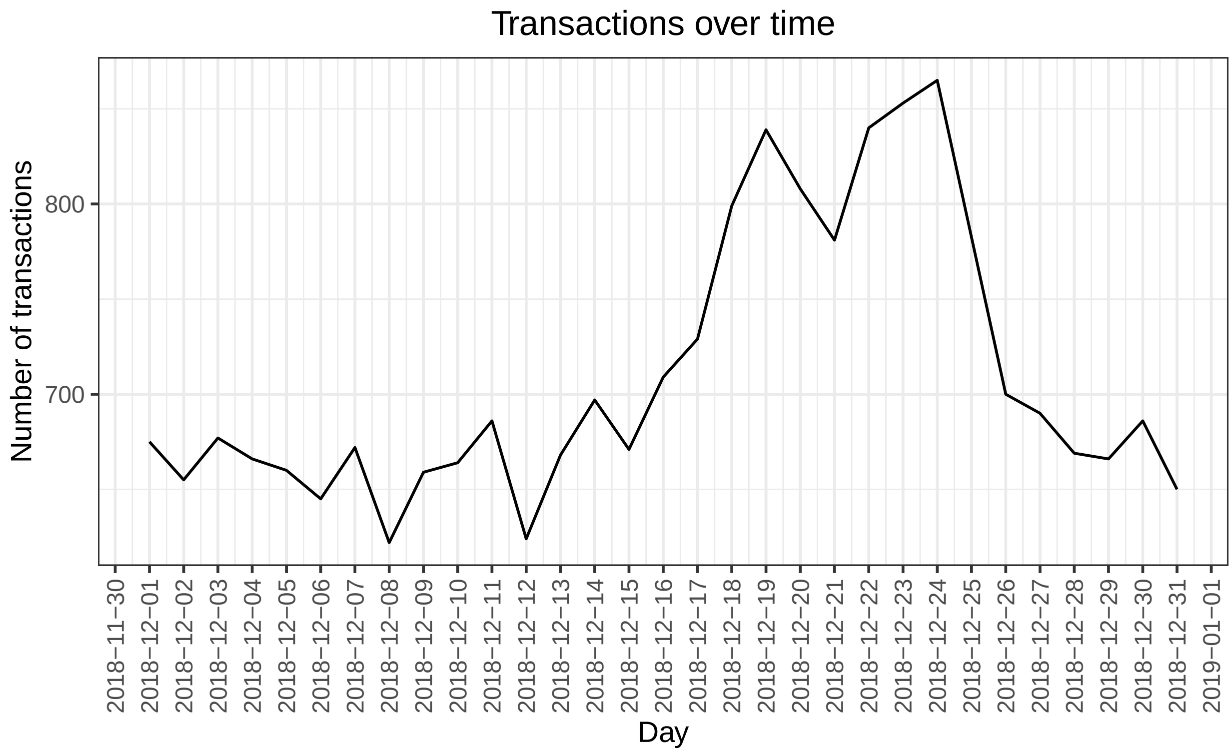
There are only 364 rows, meaning only 364 dates, which indicates a missing date. I'll create a sequence of dates from 1 Jul 2018 to 30 Jun 2019 and use this to create a chart of the number of transactions over time to find the missing date.

1.3.2 Plotting Transaction over time

```r
theme_set(theme_bw())
theme_update(plot.title = element_text(hjust = 0.5))
#### Plot transactions over time
P_TRANS <- Tran_OVR_TIM %>%
    ggplot(aes(x = DATE, y = No_of_TRANS)) + 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))
P_TRANS
```




```r
#### Plot transaction of month December
P_TRANS_DEC <- Tran_OVR_TIM %>%
    filter(month(DATE) == 12) %>%
    ggplot(aes(x = DATE, y = No_of_TRANS)) + geom_line() + labs(x = "Day",
    y = "Number of transactions", title = "Transactions over time") +
    scale_x_date(breaks = "1 day") + theme(axis.text.x = element_text(angle = 90,
    vjust = 0.5))
P_TRANS_DEC
```

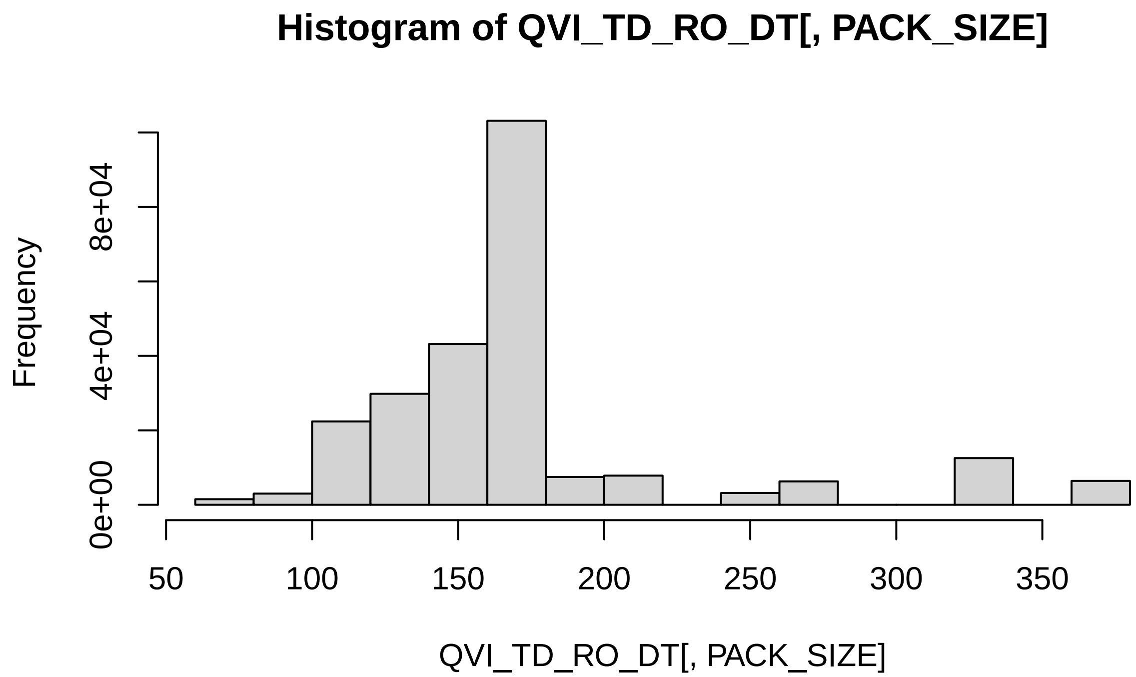


Further analysis of the transaction patterns revealed a notable increase in purchases in December, peaking in the lead-up to Christmas. Subsequently, there was a break in sales, with zero sales recorded on Christmas Day itself, which is likely due to store closures on that day.

Now that I am satisfied that the data no longer has outliers, I can move on to creating other features, such as brand of chips or pack size, from PROD_NAME. I will start with pack size.

1.3.3 Pack size

```
QVI_TD_RO_DT <- data.table(QVI_TD_RO)
QVI_TD_RO_DT[, PACK_SIZE := parse_number(PROD_NAME)]
H_Pack_size_dt <- QVI_TD_RO_DT[, .N, PACK_SIZE][order(PACK_SIZE)]
view(H_Pack_size_dt)
pack_size_labl <- c(70, 90, 110, 125, 134, 135, 150, 160, 165,
    170, 175, 180, 190, 200, 210, 220, 250, 270, 330, 380)
hist(QVI_TD_RO_DT[, PACK_SIZE])
```



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

1.3.4 Checking for Brand:-

To create brands, I used the first word in PROD_NAME to determine the brand name. Some of the brand names appeared to be the same, such as 'RED' and 'RRD', which both refer to Red Rock Deli chips, so I combined these together.

```
QVI_TD_RO_DT[, BRAND := word(PROD_NAME, 1)]
B_Tran_dt <- QVI_TD_RO_DT[, .N, BRAND][order(-N)]
QVI_TD_RO_DT[, Brand := NULL]
```

```
## Warning in `[.data.table`(QVI_TD_RO_DT, , `:=`(Brand, NULL)): Tried to assign
## NULL to column 'Brand', but this column does not exist to remove
```

```
View(B_Tran_dt)
print(B_Tran_dt)
```

```
##          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
```

```
QVI_TD_RO_DT[BRAND == "Red", BRAND := "RRD"]
QVI_TD_RO_DT[BRAND == "Smith", BRAND := "Smiths"]
QVI_TD_RO_DT[BRAND == "Infzns", BRAND := "Infuzions"]
QVI_TD_RO_DT[BRAND == "Snbts", BRAND := "Sunbites"]
QVI_TD_RO_DT[BRAND == "NCC", BRAND := "NATURAL"]
QVI_TD_RO_DT[BRAND == "DORITO", BRAND := "DORITOS"]
QVI_TD_RO_DT[BRAND == "GRAIN", BRAND := "GRNWVES"]
QVI_TD_RO_DT[BRAND == "WW", BRAND := "WOOLWORTHS"]

print(QVI_TD_RO_DT)
```

```
##               DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##             <Date>     <num>          <num>  <num>    <num>
##      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>    <num>     <num>     <num>
##      1:   Natural Chip        Compny SeaSalt175g        2       6.0       175
##      2:                 CCs Nacho Cheese    175g        3       6.3       175
##      3:   Smiths Crinkle Cut  Chips Chicken 170g        2       2.9       170
##      4:   Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0       175
##      5: Kettle Tortilla ChpsHny&Jlpno Chili 150g        3      13.8       150
```

```
##     ---
## 246736:  Kettle Sweet Chilli And Sour Cream 175g        2      10.8       175
## 246737:            Tostitos Splash Of  Lime 175g        1       4.4       175
## 246738:                 Doritos Mexicana    170g        2       8.8       170
## 246739:  Doritos Corn Chip Mexican Jalapeno 150g        2       7.8       150
## 246740:            Tostitos Splash Of  Lime 175g        2       8.8       175
##            BRAND
##           <char>
##      1:  Natural
##      2:      CCs
##      3:   Smiths
##      4:   Smiths
##      5:   Kettle
##     ---
## 246736:   Kettle
## 246737: Tostitos
## 246738:  Doritos
## 246739:  Doritos
## 246740: Tostitos
```

```
View(QVI_TD_RO_DT)
```

1.4 Examining customer data

```
View(QVI_PB)
str(QVI_PB)
```

```
## 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>
```

1.4.1 Examining the values of lifestage and premium_customer

Let's have a closer look at the LIFESTAGE and PREMIUM_CUSTOMER columns.

```
Customer_Summary_LS <- QVI_PB %>%
    group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
    summarise(No_of_customers = n())
```

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

```
print(Customer_Summary_LS)
```

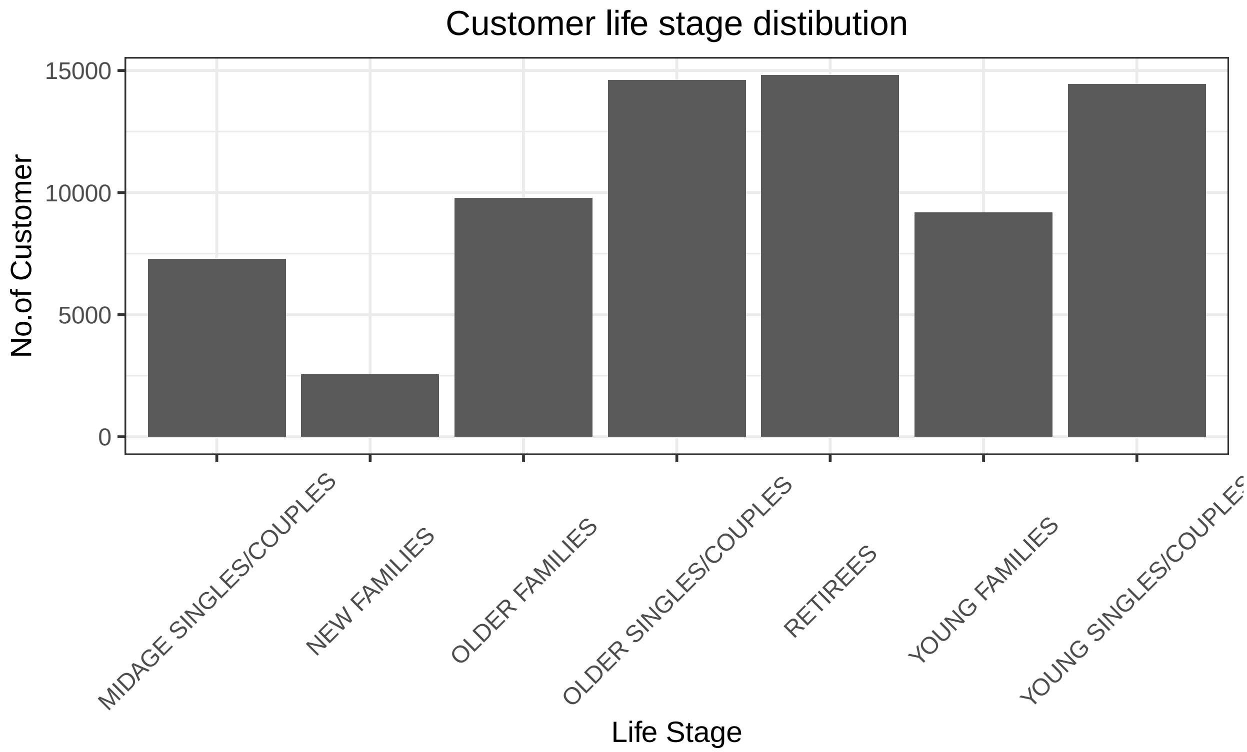
```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER No_of_customers
```

```
##    <chr>                  <chr>                      <int>
##  1 MIDAGE SINGLES/COUPLES Budget                      1504
##  2 MIDAGE SINGLES/COUPLES Mainstream                  3340
##  3 MIDAGE SINGLES/COUPLES Premium                     2431
##  4 NEW FAMILIES           Budget                      1112
##  5 NEW FAMILIES           Mainstream                   849
##  6 NEW FAMILIES           Premium                      588
##  7 OLDER FAMILIES         Budget                      4675
##  8 OLDER FAMILIES         Mainstream                  2831
##  9 OLDER FAMILIES         Premium                     2274
## 10 OLDER SINGLES/COUPLES  Budget                      4929
## # i 11 more rows
```

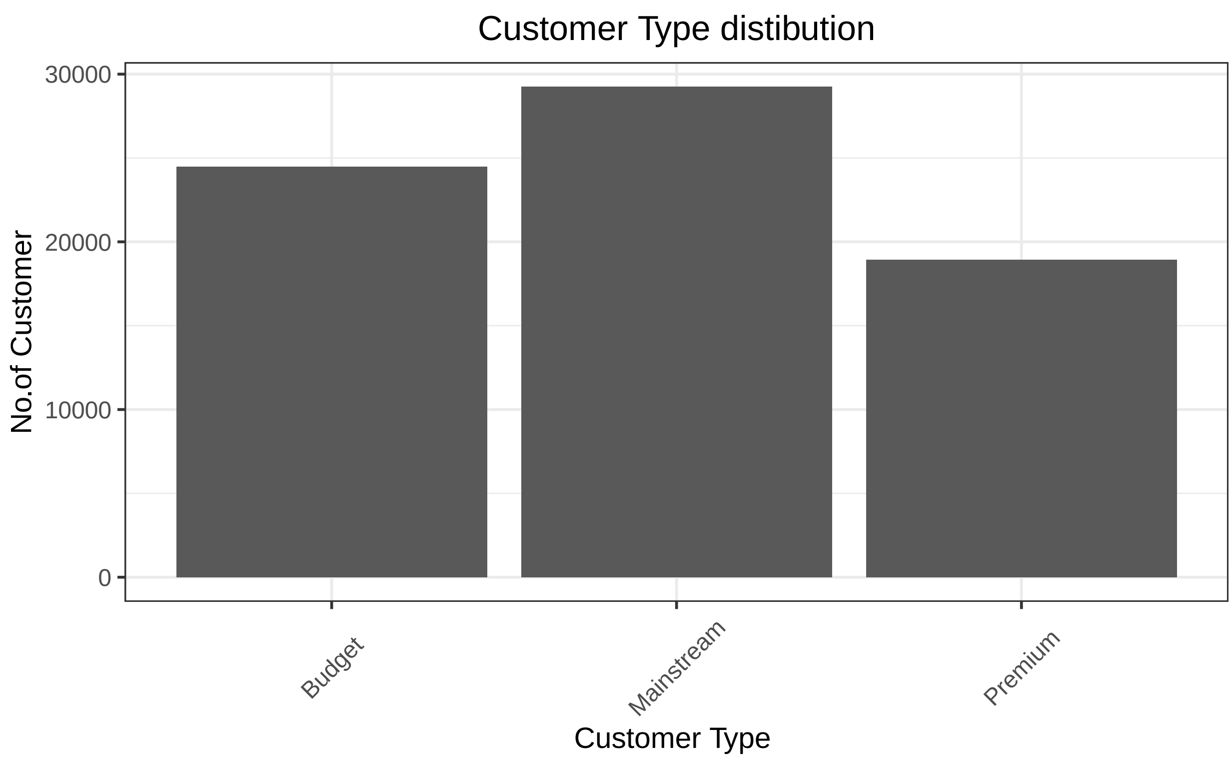
```r
Customer_Summary_PC <- QVI_PB %>%
    group_by(PREMIUM_CUSTOMER) %>%
    summarise(No_of_customers = n())
print(Customer_Summary_PC)
```

```
## # A tibble: 3 x 2
##   PREMIUM_CUSTOMER No_of_customers
##   <chr>                      <int>
## 1 Budget                     24470
## 2 Mainstream                 29245
## 3 Premium                    18922
```

```r
histo_Lifestage <- QVI_PB %>%
    ggplot(aes(x = LIFESTAGE)) + geom_bar(position = "dodge") +
    labs(title = "Customer life stage distibution", x = "Life Stage",
        y = "No.of Customer") + theme(axis.text.x = element_text(angle = 45,
    vjust = 0.5))
histo_Lifestage
```




```
histo_PC <- QVI_PB %>%
    ggplot(aes(x = PREMIUM_CUSTOMER)) + geom_bar(position = "dodge") +
    labs(title = "Customer Type distibution", x = "Customer Type ",
        y = "No.of Customer") + theme(axis.text.x = element_text(angle = 45,
    vjust = 0.5))
histo_PC
```



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

```
data <- merge(QVI_TD_RO_DT, QVI_PB, all.x = TRUE)
View(data)
na_values <- data %>%
    filter(is.na(PREMIUM_CUSTOMER))
print(na_values)
```

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

```
write.csv(data, file = "QVI_DATA.csv")
```

1.5 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
- Howmany 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

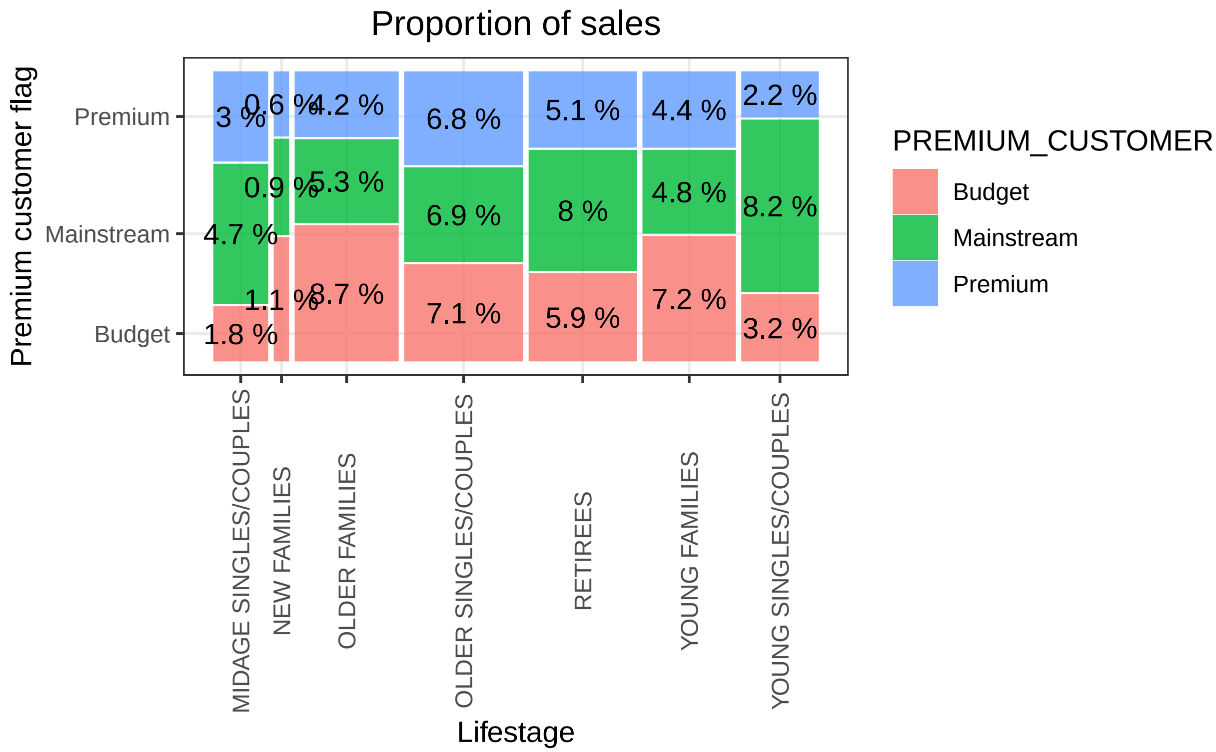
Let's start with calculating total sales by LIFESTAGE and PREMIUM_CUSTOMER and plotting the split by these segments to describe which customer segment contribute most to chip sales. ### Total Sales

```
# Total sales by lifestage and premium_Customer
Total_Sales_Summary <- data %>%
    group_by(PREMIUM_CUSTOMER, LIFESTAGE) %>%
    summarise(T_sales_PC_LS = sum(TOT_SALES), Average_sales = mean(TOT_SALES))
```

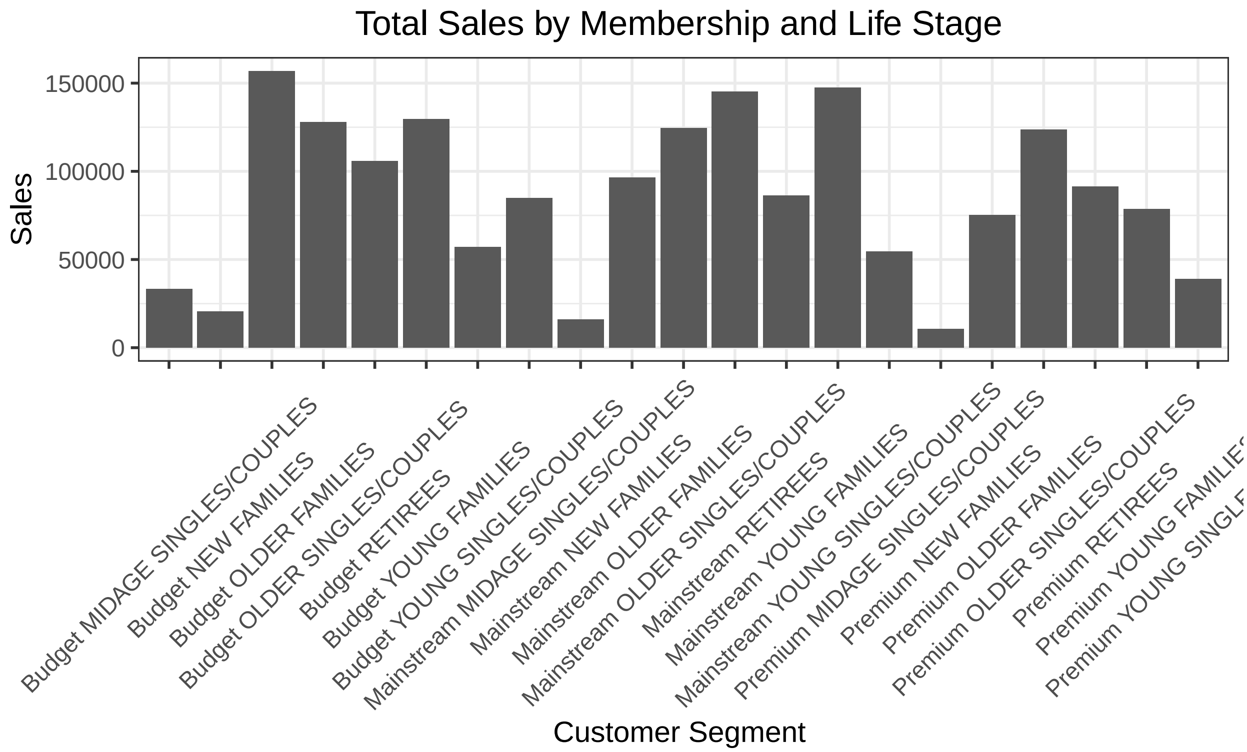
```
## `summarise()` has grouped output by 'PREMIUM_CUSTOMER'. You can override using
## the `.groups` argument.
```

```
View(Total_Sales_Summary)

p <- ggplot(data = Total_Sales_Summary) + geom_mosaic(aes(weight = T_sales_PC_LS,
    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))

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, "%"))))
```

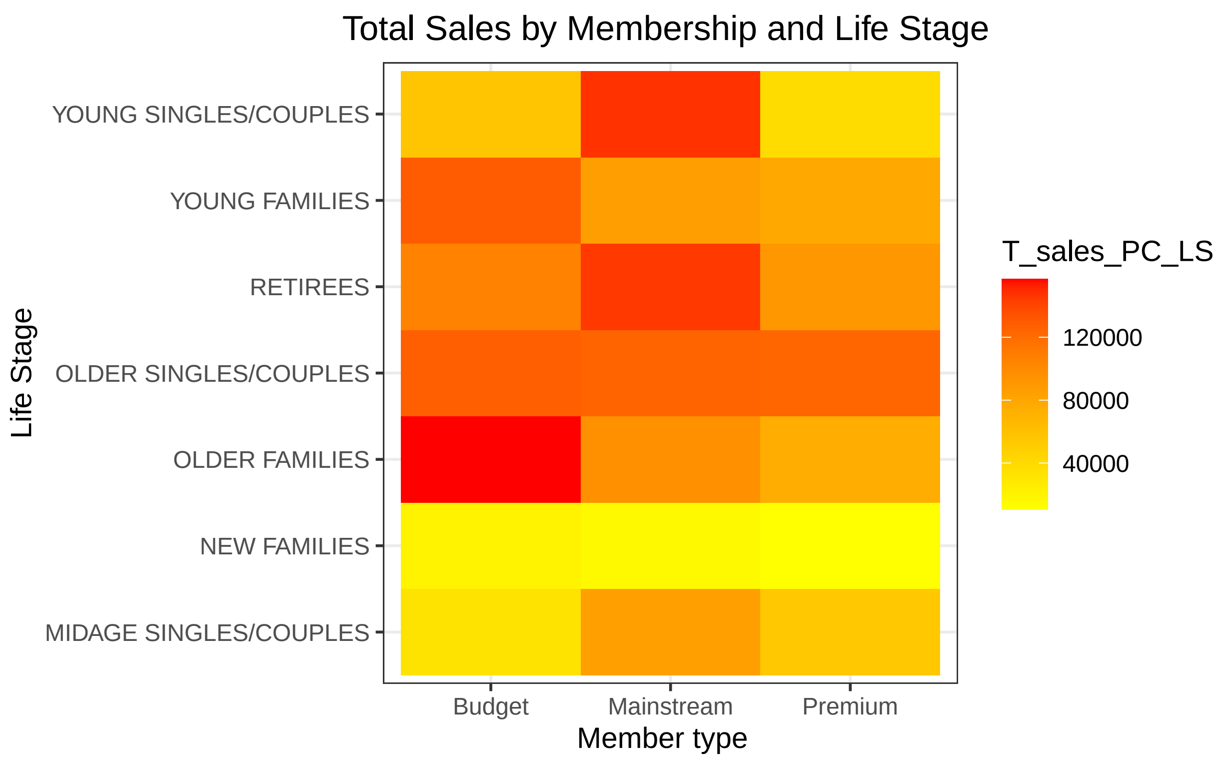
```
## Warning: The `scale_name` argument of `continuous_scale()` is deprecated as of ggplot2
## 3.5.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## 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.
```




```
Bar_tot_SalesPCLS <- Total_Sales_Summary %>%
    ggplot(aes(x = paste(PREMIUM_CUSTOMER, LIFESTAGE), y = T_sales_PC_LS)) +
    geom_bar(position = "dodge", stat = "Identity") + labs(title = "Total Sales by Membership and Life
    x = "Customer Segment ", y = "Sales") + theme(axis.text.x = element_text(angle = 45,
    vjust = 0.5))
Bar_tot_SalesPCLS
```



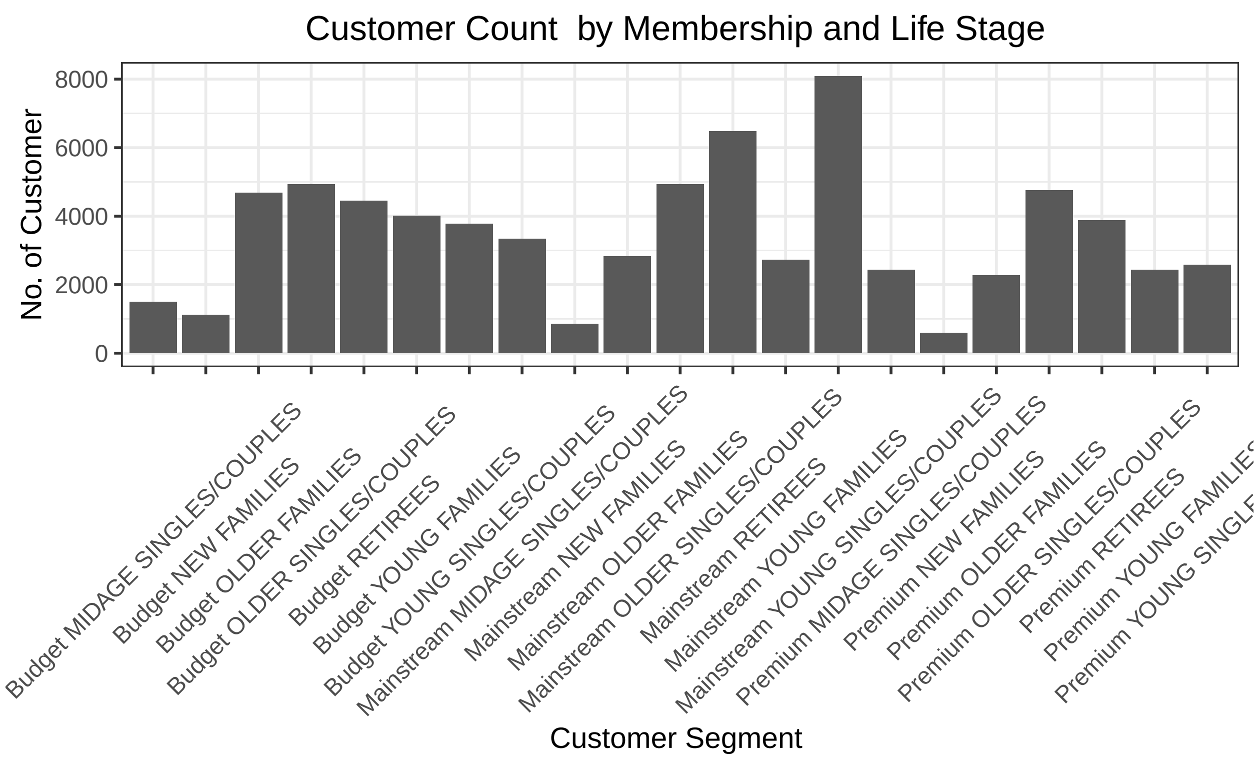

```
Heat_tot_SalesPCLS <- Total_Sales_Summary %>%
    ggplot(aes(x = PREMIUM_CUSTOMER, y = LIFESTAGE, fill = T_sales_PC_LS)) +
    geom_tile() + labs(title = "Total Sales by Membership and Life Stage ",
    x = "Member type  ", y = "Life Stage") + scale_fill_gradient(low = "yellow",
    high = "red")
Heat_tot_SalesPCLS
```



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

1.5.1 Customer

```
View(Customer_Summary_LS)
Bar_tot_customerPCLS <- Customer_Summary_LS %>%
    ggplot(aes(x = paste(PREMIUM_CUSTOMER, LIFESTAGE), y = No_of_customers)) +
    geom_bar(position = "dodge", stat = "Identity") + labs(title = "Customer Count  by Membership and Li
    x = "Customer Segment ", y = "No. of Customer") + theme(axis.text.x = element_text(angle = 45,
    vjust = 0.5))
Bar_tot_customerPCLS
```

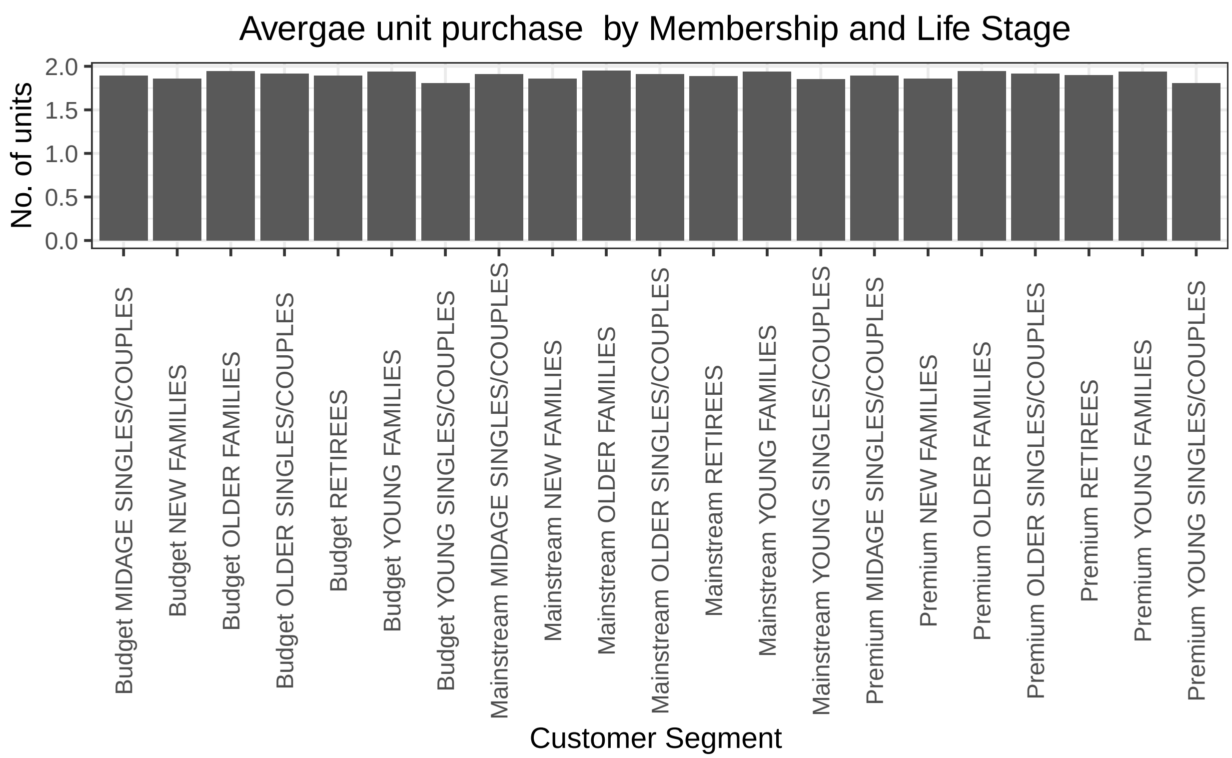


There are more Mainstream - young singles/couples and Mainstream - retirees who buy chips. This contributes to there being more sales to these customer segments but this is not a major driver for the Budget - Older families segment. Higher sales may also be driven by more units of chips being bought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

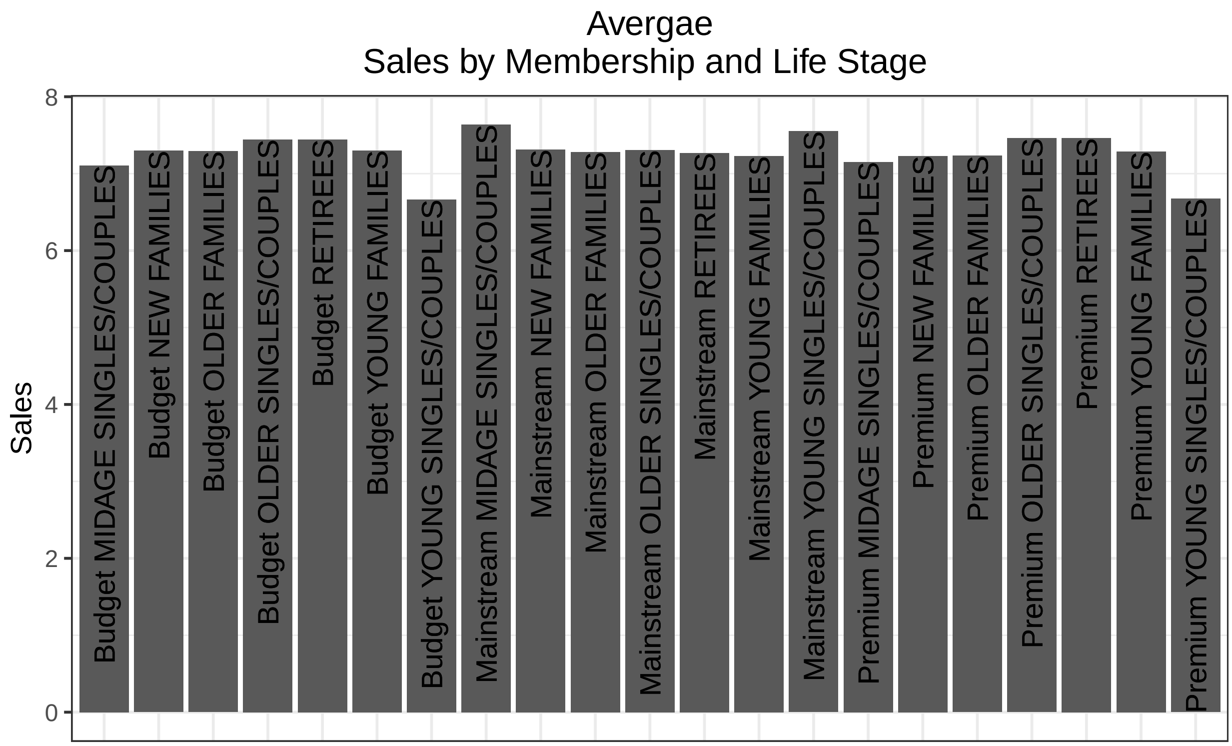
```
Average_Units_summary <- data %>%
    group_by(PREMIUM_CUSTOMER, LIFESTAGE) %>%
    summarise(Avergae_units = mean(PROD_QTY))
```

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

```
View(Average_Units_summary)
Average_unit_plot <- Average_Units_summary %>%
    ggplot(aes(x = paste(PREMIUM_CUSTOMER, LIFESTAGE), y = Avergae_units)) +
    geom_bar(position = "dodge", stat = "Identity") + labs(title = "Avergae unit purchase  by Membershi
    x = "Customer Segment ", y = "No. of units") + theme(axis.text.x = element_text(angle = 90,
    vjust = 0.5))
Average_unit_plot
```




```
Bar_Avg_SalesPCLS <- Total_Sales_Summary %>%
    ggplot(aes(x = paste(PREMIUM_CUSTOMER, LIFESTAGE), y = Average_sales,
        )) + geom_bar(position = "dodge", stat = "Identity") +
    labs(title = "Avergae
Sales by Membership and Life Stage ",
        x = "Customer Segment ", y = "Sales") + geom_text(aes(label = paste(PREMIUM_CUSTOMER,
    LIFESTAGE), vjust = 0.5, hjust = 1, angle = 90, )) + geom_text(aes(label = paste(PREMIUM_CUSTOMER,
    LIFESTAGE), vjust = 0.5, hjust = 1, angle = 90)) + theme(axis.text.x = element_blank(),
    axis.ticks.x = element_blank(), axis.title.x = element_blank())
Bar_Avg_SalesPCLS
```



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

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.

1.5.2 Perform an independent t-test between mainstream vs premium and budget midage and young singles and couples

```
Average_Units_PC_M_Y <- data %>%
    filter(PREMIUM_CUSTOMER == "Premium", LIFESTAGE == c("MIDAGE SINGLES/COUPLES",
        "YOUNG SINGLES/COUPLES"))
PC_T_Test <- Average_Units_PC_M_Y %>%
    select(PROD_QTY)
View(PC_T_Test)
Average_Units_MC_M_Y <- data %>%
    filter(PREMIUM_CUSTOMER == "Mainstream", LIFESTAGE == c("MIDAGE SINGLES/COUPLES",
        "YOUNG SINGLES/COUPLES"))
MC_T_Test <- Average_Units_MC_M_Y %>%
    select(PROD_QTY)
View(MC_T_Test)
View(Average_Units_MC_M_Y)
Average_Units_BC_M_Y <- data %>%
```

```r
    select(PROD_QTY, PREMIUM_CUSTOMER, LIFESTAGE) %>%
    filter(PREMIUM_CUSTOMER == "Budget", LIFESTAGE == c("MIDAGE SINGLES/COUPLES",
        "YOUNG SINGLES/COUPLES"))
View(Average_Units_BC_M_Y)
BC_T_Test <- Average_Units_BC_M_Y %>%
    select(PROD_QTY)
View(BC_T_Test)

t.test(MC_T_Test, PC_T_Test)
```

```
##
##  Welch Two Sample t-test
##
## data:  MC_T_Test and PC_T_Test
## t = 2.7346, df = 12167, p-value = 0.006255
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.00433689 0.02629155
## sample estimates:
## mean of x mean of y
##  1.872543  1.857228
```

```r
t.test(MC_T_Test, BC_T_Test)
```

```
##
##  Welch Two Sample t-test
##
## data:  MC_T_Test and BC_T_Test
## t = 5.5513, df = 11195, p-value = 2.9e-08
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.02122589 0.04439761
## sample estimates:
## mean of x mean of y
##  1.872543  1.839731
```

The t-test results in a p-value $< 2.2e\text{-}16$, 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.

1.5.3 Affinity Test

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.

1.5.4 Brand

```r
filtered_data <- data[LIFESTAGE == "YOUNG SINGLES/COUPLES" &
    PREMIUM_CUSTOMER == "Mainstream"]
View(filtered_data)
brand_frequencies <- filtered_data[, .N, by = BRAND][order(-N)]  # Count and sort by frequency
print(brand_frequencies)
```

```
##          BRAND     N
##         <char> <int>
##  1:     Kettle  3844
##  2:   Pringles  2315
##  3:    Doritos  2076
##  4:     Smiths  1921
##  5:  Infuzions  1250
##  6:      Thins  1166
##  7:   Twisties   900
##  8:   Tostitos   890
##  9:        RRD   875
## 10:       Cobs   864
## 11:   Tyrrells   619
## 12:      Grain   576
## 13: WOOLWORTHS   423
## 14:   Cheezels   346
## 15:    Natural   321
## 16:     Dorito   303
## 17:        CCs   222
## 18:    Cheetos   166
## 19:   Sunbites   128
## 20:     French    78
## 21:    NATURAL    73
## 22:    GrnWves    70
## 23:     Burger    62
## 24: Woolworths    56
##          BRAND     N
```

```r
total_transcation <- nrow(filtered_data)
brand_proportions <- filtered_data[, .N/total_transcation, by = BRAND][order(-V1)]
print(brand_proportions)
```

```
##          BRAND          V1
##         <char>       <num>
##  1:     Kettle 0.196684404
##  2:   Pringles 0.118450675
##  3:    Doritos 0.106221858
##  4:     Smiths 0.098291036
##  5:  Infuzions 0.063958248
##  6:      Thins 0.059660254
##  7:   Twisties 0.046049939
##  8:   Tostitos 0.045538273
##  9:        RRD 0.044770774
## 10:       Cobs 0.044207941
## 11:   Tyrrells 0.031672124
## 12:      Grain 0.029471961
```

```
## 13: WOOLWORTHS 0.021643471
## 14:   Cheezels 0.017703643
## 15:    Natural 0.016424478
## 16:     Dorito 0.015503479
## 17:        CCs 0.011358985
## 18:    Cheetos 0.008493655
## 19:   Sunbites 0.006549325
## 20:     French 0.003990995
## 21:    NATURAL 0.003735162
## 22:    GrnWves 0.003581662
## 23:     Burger 0.003172329
## 24: Woolworths 0.002865330
##          BRAND          V1
```

```r
installed.packages("arules")
```

```
##      Package LibPath Version Priority Depends Imports LinkingTo Suggests
##      Enhances License License_is_FOSS License_restricts_use OS_type Archs
##      MD5sum NeedsCompilation Built
```

```r
library(arules)
```

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

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

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

```
##
## Attaching package: 'arules'
```

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

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

```r
transactions <- as(split(filtered_data$BRAND, filtered_data$TXN_ID),
    "transactions")
```

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

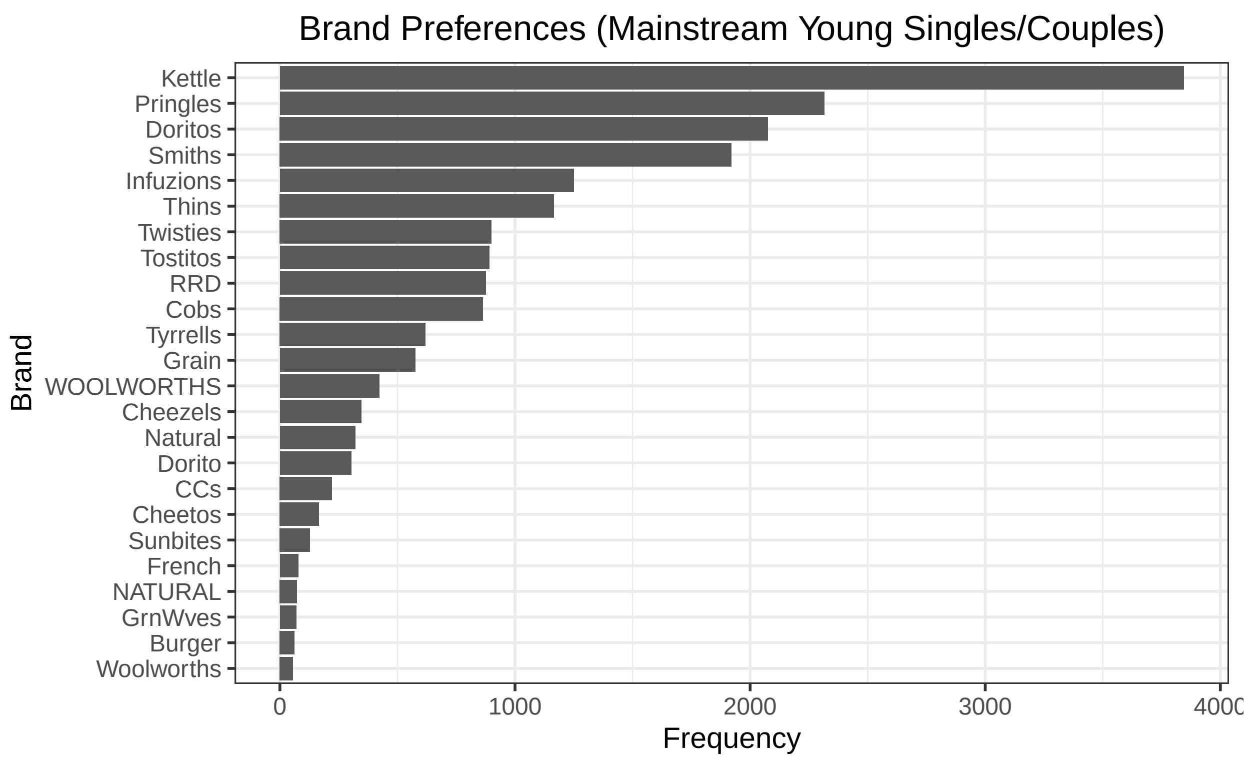
```r
rules <- apriori(transactions, parameter = list(support = 0.01,
    confidence = 0.1))  # Adjust support and confidence
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5    0.01      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: 194
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[24 item(s), 19482 transaction(s)] done [0.00s].
## sorting and recoding items ... [17 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [3 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
inspect(rules)
```

```
##     lhs    rhs        support   confidence coverage lift count
## [1] {}  => {Doritos}  0.1065086 0.1065086  1        1    2075
## [2] {}  => {Pringles} 0.1188276 0.1188276  1        1    2315
## [3] {}  => {Kettle}   0.1973103 0.1973103  1        1    3844
```

```r
ggplot(brand_frequencies, aes(x = reorder(BRAND, N), y = N)) +
    geom_bar(stat = "identity") + coord_flip() + labs(title = "Brand Preferences (Mainstream Young Singl
    x = "Brand", y = "Frequency")
```




```
#### Deep dive into Mainstream, young singles/couples
segment1 <- data[LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER ==
    "Mainstream", ]
other <- data[!(LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER ==
    "Mainstream"), ]
#### Brand affinity compared to the rest of the population
quantity_segment1 <- segment1[, sum(PROD_QTY)]
quantity_other <- other[, sum(PROD_QTY)]
quantity_segment1_by_brand <- segment1[, .(targetSegment = sum(PROD_QTY)/quantity_segment1),
    by = BRAND]
quantity_other_by_brand <- other[, .(other = sum(PROD_QTY)/quantity_other),
    by = BRAND]
brand_proportions <- merge(quantity_segment1_by_brand, quantity_other_by_brand)[,
    affinityToBrand := targetSegment/other]
brand_proportions[order(affinityToBrand)]
```

```
##          BRAND targetSegment       other affinityToBrand
##         <char>         <num>       <num>           <num>
##  1:     Burger   0.002926156 0.006596434       0.4435967
##  2: Woolworths   0.002843340 0.006377627       0.4458304
##  3: WOOLWORTHS   0.021256039 0.043049561       0.4937574
##  4:   Sunbites   0.006349206 0.012580210       0.5046980
##  5:    GrnWves   0.003588682 0.006066692       0.5915385
##  6:        CCs   0.011180124 0.018895650       0.5916771
##  7:    NATURAL   0.003643892 0.005873221       0.6204248
##  8:    Natural   0.015955832 0.024980768       0.6387246
##  9:        RRD   0.043809524 0.067493678       0.6490908
## 10:    Cheetos   0.008033126 0.012066591       0.6657329
```

```
## 11:      French   0.003947550 0.005758060       0.6855694
## 12:      Smiths   0.096369910 0.124583692       0.7735355
## 13:    Cheezels   0.017971014 0.018646902       0.9637534
## 14:       Thins   0.060372671 0.056986370       1.0594230
## 15:   Infuzions   0.064679089 0.057064679       1.1334347
## 16:        Cobs   0.044637681 0.039048861       1.1431238
## 17:       Grain   0.029123533 0.025121265       1.1593180
## 18:    Pringles   0.119420290 0.100634769       1.1866703
## 19:    Tostitos   0.045410628 0.037977861       1.1957131
## 20:      Kettle   0.197984817 0.165553442       1.1958967
## 21:     Doritos   0.107053140 0.088314823       1.2121764
## 22:    Twisties   0.046183575 0.037876520       1.2193194
## 23:    Tyrrells   0.031552795 0.025692464       1.2280953
## 24:      Dorito   0.015707384 0.012759861       1.2309996
##           BRAND targetSegment       other affinityToBrand
```

We can see that : • Mainstream young singles/couples are 23% more likely to purchase Tyrrells chips compared to the rest of the population • Mainstream young singles/couples are 56% less likely to purchase Burger Rings compared to the rest of the population

1.5.5 Pack Size

```
Pack_Size_frequencies <- filtered_data[, .N, by = PACK_SIZE][order(-N)]  # Count and sort by frequency
print(Pack_Size_frequencies)
```

```
##     PACK_SIZE     N
##         <num> <int>
##  1:       175  4997
##  2:       150  3080
##  3:       134  2315
##  4:       110  2051
##  5:       170  1575
##  6:       330  1195
##  7:       165  1102
##  8:       380   626
##  9:       270   620
## 10:       210   576
## 11:       135   290
## 12:       250   280
## 13:       200   179
## 14:       190   148
## 15:        90   128
## 16:       160   128
## 17:       180    70
## 18:        70    63
## 19:       220    62
## 20:       125    59
##     PACK_SIZE     N
```

```
total_transcation_P <- nrow(filtered_data)
Pack_Size_proportions <- filtered_data[, .N/total_transcation_P,
```

```r
    by = PACK_SIZE][order(-V1)]
print(Pack_Size_proportions)
```

```
##     PACK_SIZE          V1
##         <num>       <num>
##  1:       175 0.255679492
##  2:       150 0.157593123
##  3:       134 0.118450675
##  4:       110 0.104942693
##  5:       170 0.080587393
##  6:       330 0.061144085
##  7:       165 0.056385591
##  8:       380 0.032030291
##  9:       270 0.031723291
## 10:       210 0.029471961
## 11:       135 0.014838314
## 12:       250 0.014326648
## 13:       200 0.009158821
## 14:       190 0.007572657
## 15:        90 0.006549325
## 16:       160 0.006549325
## 17:       180 0.003581662
## 18:        70 0.003223496
## 19:       220 0.003172329
## 20:       125 0.003018829
##     PACK_SIZE          V1
```

```r
install.packages("arules")
```

```
## Warning: package 'arules' is in use and will not be installed
```

```r
library(arules)

transactions <- as(split(filtered_data$PACK_SIZE, filtered_data$TXN_ID),
    "transactions")
```

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

```r
rules <- apriori(transactions, parameter = list(support = 0.1,
    confidence = 0.1))  # Adjust support and confidence
```

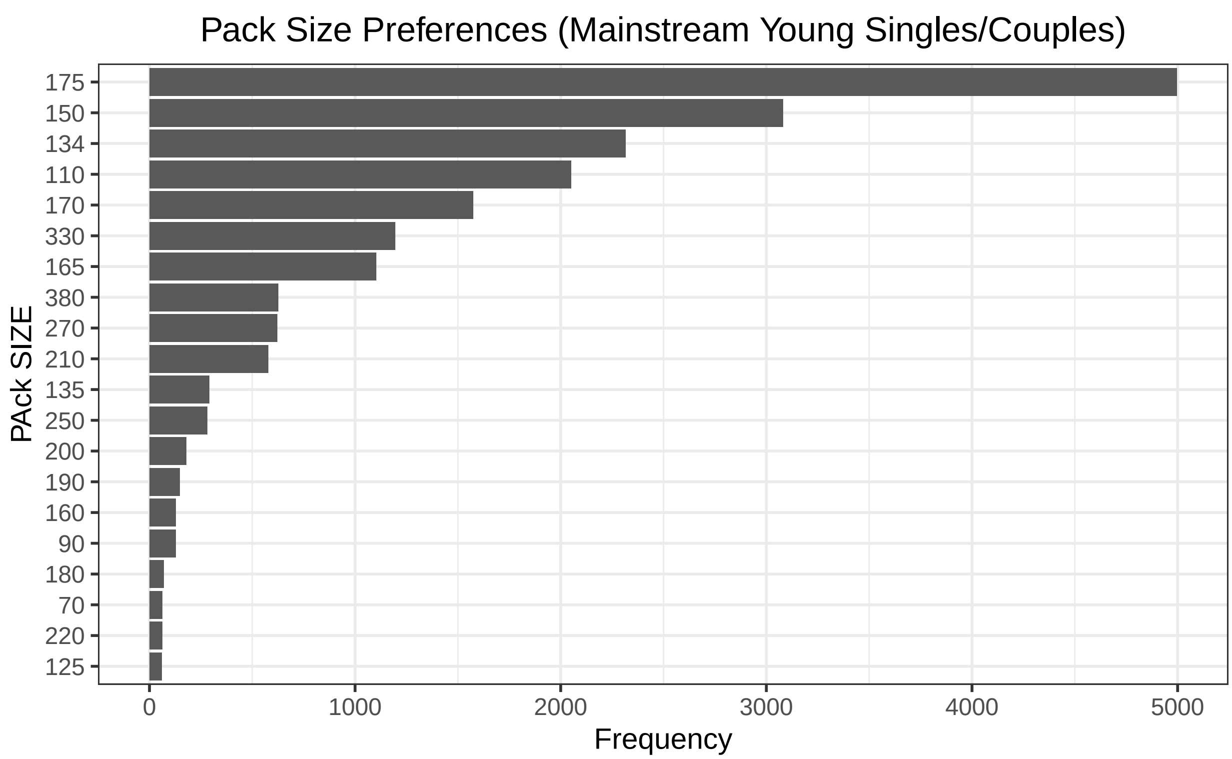
```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5     0.1      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt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

```
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 1948
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[20 item(s), 19482 transaction(s)] done [0.00s].
## sorting and recoding items ... [4 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [4 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
inspect(rules)
```

```
##     lhs    rhs   support   confidence coverage lift count
## [1] {}  => {110} 0.1052253 0.1052253 1        1    2050
## [2] {}  => {134} 0.1188276 0.1188276 1        1    2315
## [3] {}  => {150} 0.1579920 0.1579920 1        1    3078
## [4] {}  => {175} 0.2562879 0.2562879 1        1    4993
```

```r
ggplot(Pack_Size_frequencies, aes(x = reorder(PACK_SIZE, N),
    y = N)) + geom_bar(stat = "identity") + coord_flip() + labs(title = "Pack Size Preferences (Mainstre
    x = "PAck SIZE", y = "Frequency")
```




```r
#### Preferred pack size compared to the rest of the
#### population
quantity_segment1_by_pack <- segment1[, .(targetSegment = sum(PROD_QTY)/quantity_segment1),
```

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

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

```r
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.

1.6 Conclusion

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

2 Task 2

In this section, we will examine the performance of trial stores against control stores to provide recommendations for each location. The analysis will focus on the following areas:

- **Control Store Selection**: We will explore the data to define appropriate metrics for control store selection. This involves identifying the key drivers and visualizing them to ensure the chosen stores are suitable controls. A function may be created to streamline this process.
- **Trial Store Assessment**: We will assess the performance of each trial store individually in comparison to its respective control store. This will provide insights into the success of the trial stores.
- **Findings Collation**: We will summarize the findings for each store and provide recommendations regarding the impact on sales during the trial period.

Visualizations will be a key component of this analysis to aid in understanding the data. All visualizations will be saved for inclusion in the final report, which is due to be presented to the client in three weeks. The analysis is to be submitted by mid-next week to allow sufficient time for discussion and report compilation.

2.1 Select Store :-

The client has selected store numbers 77, 86 and 88 as trial stores and want control stores to be established stores that are operational for the entire observation period. We would want to match trial stores to control stores that are similar to the trial store prior to the trial period of Feb 2019 in terms of : • Monthly overall sales revenue • Monthly number of customers • Monthly number of transactions per customer Let's first create the metrics of interest and filter to stores that are present throughout the pre-trial period

Let's first create the metrics of interest and filter to stores that are present throughout the pre-trial period

```
data[, MONTH_ID := format(DATE, "%Y%m")]
View(data)

monthly_meterics <- data[, .(total_sales_revenue = sum(TOT_SALES),
    total_customer = length(unique(LYLTY_CARD_NBR)), Total_Transaction = .N,
    Chips_per_customer = sum(PROD_QTY)/uniqueN(LYLTY_CARD_NBR),
    Avg_price_per_unit = mean(TOT_SALES/PROD_QTY)), by = .(STORE_NBR,
    MONTH_ID)][order(STORE_NBR, MONTH_ID)]
monthly_meterics[, Transaction_per_customer := Total_Transaction/total_customer][order(STORE_NBR,
    MONTH_ID)]
```

```
##       STORE_NBR MONTH_ID total_sales_revenue total_customer Total_Transaction
##           <num>   <char>               <num>          <int>             <int>
##    1:         1   201807               188.9             47                49
##    2:         1   201808               168.4             41                41
##    3:         1   201809               268.1             57                59
##    4:         1   201810               175.4             39                40
##    5:         1   201811               184.8             44                45
##   ---
```

```
## 3161:       272   201902             385.3            44                 47
## 3162:       272   201903             421.9            48                 51
## 3163:       272   201904             445.1            54                 56
## 3164:       272   201905             314.6            34                 40
## 3165:       272   201906             301.9            33                 36
##       Chips_per_customer Avg_price_per_unit Transaction_per_customer
##                    <num>              <num>                    <num>
##    1:           1.234043           3.328571                 1.042553
##    2:           1.268293           3.303659                 1.000000
##    3:           1.245614           3.716949                 1.035088
##    4:           1.307692           3.457500                 1.025641
##    5:           1.250000           3.391111                 1.022727
##   ---
## 3161:           2.022727           4.342553                 1.068182
## 3162:           2.020833           4.321569                 1.062500
## 3163:           1.944444           4.248214                 1.037037
## 3164:           2.088235           4.437500                 1.176471
## 3165:           2.060606           4.405556                 1.090909
```

```r
View(monthly_meterics)
head(monthly_meterics)
```

```
##    STORE_NBR MONTH_ID total_sales_revenue total_customer Total_Transaction
##        <num>   <char>               <num>          <int>             <int>
## 1:         1   201807               188.9             47                49
## 2:         1   201808               168.4             41                41
## 3:         1   201809               268.1             57                59
## 4:         1   201810               175.4             39                40
## 5:         1   201811               184.8             44                45
## 6:         1   201812               160.6             37                40
##    Chips_per_customer Avg_price_per_unit Transaction_per_customer
##                 <num>              <num>                    <num>
## 1:           1.234043           3.328571                 1.042553
## 2:           1.268293           3.303659                 1.000000
## 3:           1.245614           3.716949                 1.035088
## 4:           1.307692           3.457500                 1.025641
## 5:           1.250000           3.391111                 1.022727
## 6:           1.297297           3.357500                 1.081081
```

```r
#### Filtering
Pre_trial_monthly_meterics <- monthly_meterics[MONTH_ID < 201902]
View(Pre_trial_monthly_meterics)
storesWithFullObs <- unique(monthly_meterics[, .N, STORE_NBR][N ==
    12, STORE_NBR])
# filtered stores with full observation pre trial period.

filtered_Pre_trial_monthly_meterics <- Pre_trial_monthly_meterics[STORE_NBR %in%
    storesWithFullObs]
View(filtered_Pre_trial_monthly_meterics)
```

To effectively rank the similarity of each potential control store to the trial store, we will calculate the correlation between their performance. To avoid repetitive calculations for each trial store and its potential control stores, we will write a function to automate this process.

```r
calculateCorrelation <- function(inputTable, metricCol, storeComparison) {
    calcCorrTable = data.table(Store1 = numeric(), Store2 = numeric(),
        corr_measure = numeric())
    storeNumbers <- unique(inputTable[STORE_NBR != storeComparison,
        STORE_NBR])

    for (i in storeNumbers) {
        trialStoreMetrics <- inputTable[STORE_NBR == storeComparison,
            get(metricCol)]
        controlStoreMetrics <- inputTable[STORE_NBR == i, get(metricCol)]
        calculatedMeasure = data.table(Store1 = storeComparison,
            Store2 = i, corr_measure = cor(trialStoreMetrics,
                controlStoreMetrics))

        calcCorrTable <- rbind(calcCorrTable, calculatedMeasure)
    }
    return(calcCorrTable)
}
```

Apart from correlation, we can also calculate a standardised metric based on the absolute difference between the trial store's performance and each control store's performance.

```r
# Create a function to calculate a standardized magnitude
# distance for a measure

calculateMagnitudeDistance <- function(inputTable, metricCol,
    storeComparison) {
    calcDistTable = data.table(Store1 = numeric(), Store2 = numeric(),
        YEARMONTH = numeric(), measure = numeric())
    storeNumbers <- unique(inputTable[STORE_NBR != storeComparison,
        STORE_NBR])

    for (i in storeNumbers) {
        calculatedMeasure = data.table(Store1 = storeComparison,
            Store2 = i, YEARMONTH = inputTable[STORE_NBR == storeComparison,
                MONTH_ID], measure = abs(inputTable[STORE_NBR ==
                storeComparison, eval(metricCol)] - inputTable[STORE_NBR ==
                i, eval(metricCol)]))
        calcDistTable <- rbind(calcDistTable, calculatedMeasure)
    }
    #### Standardize the magnitude distance so that the
    #### measure ranges from 0 to 1
    minMaxDist <- calcDistTable[, .(minDist = min(measure), maxDist = max(measure)),
        by = c("Store1", "YEARMONTH")]
    distTable <- merge(calcDistTable, minMaxDist, by = c("Store1",
        "YEARMONTH"))
    distTable[, magnitudeMeasure := 1 - (measure - minDist)/(maxDist -
        minDist)]

    finalDistTable <- distTable[, .(mag_measure = mean(magnitudeMeasure)),
        by = .(Store1, Store2)]
    return(finalDistTable)
}
```

Now let's use the functions to find the control stores! We'll select control stores based on how similar monthly total sales in dollar amounts and monthly number of customers are to the trial stores. So we will need to use our functions to get four scores, two for each of total sales and total customers.

```
trial_store <- 77  # Store number of the trial store
corr_nSales <- calculateCorrelation(filtered_Pre_trial_monthly_meterics,
    "total_sales_revenue", trial_store)
print(corr_nSales)
```

```
##      Store1 Store2 corr_measure
##       <num>  <num>        <num>
##   1:     77      1 -0.005382429
##   2:     77      2 -0.251182809
##   3:     77      3  0.660446832
##   4:     77      4 -0.347846468
##   5:     77      5 -0.139047983
##  ---
## 254:     77    268  0.395460337
## 255:     77    269 -0.466370424
## 256:     77    270  0.274854303
## 257:     77    271  0.195189898
## 258:     77    272 -0.179646952
```

```
corr_ncustomer <- calculateCorrelation(filtered_Pre_trial_monthly_meterics,
    "total_customer", trial_store)
print(corr_ncustomer)
```

```
##      Store1 Store2 corr_measure
##       <num>  <num>        <num>
##   1:     77      1  0.337865596
##   2:     77      2 -0.596491730
##   3:     77      3  0.755248715
##   4:     77      4 -0.305411652
##   5:     77      5  0.224768439
##  ---
## 254:     77    268  0.369735946
## 255:     77    269 -0.247580595
## 256:     77    270 -0.009181744
## 257:     77    271  0.023634941
## 258:     77    272  0.068677178
```

```
magnitude_nSales <- calculateMagnitudeDistance(filtered_Pre_trial_monthly_meterics,
    quote(total_sales_revenue), trial_store)
print(magnitude_nSales)
```

```
##      Store1 Store2 mag_measure
##       <num>  <num>       <num>
##   1:     77      1   0.9536909
##   2:     77      2   0.9372067
##   3:     77      3   0.3454316
##   4:     77      4   0.1810682
##   5:     77      5   0.5651305
```

```
## ---
## 254:     77    268   0.9636567
## 255:     77    269   0.4552162
## 256:     77    270   0.4584257
## 257:     77    271   0.5727032
## 258:     77    272   0.8928227
```

```
magnitude_nCustomers <- calculateMagnitudeDistance(filtered_Pre_trial_monthly_meterics,
    quote(total_customer), trial_store)
print(magnitude_nCustomers)
```

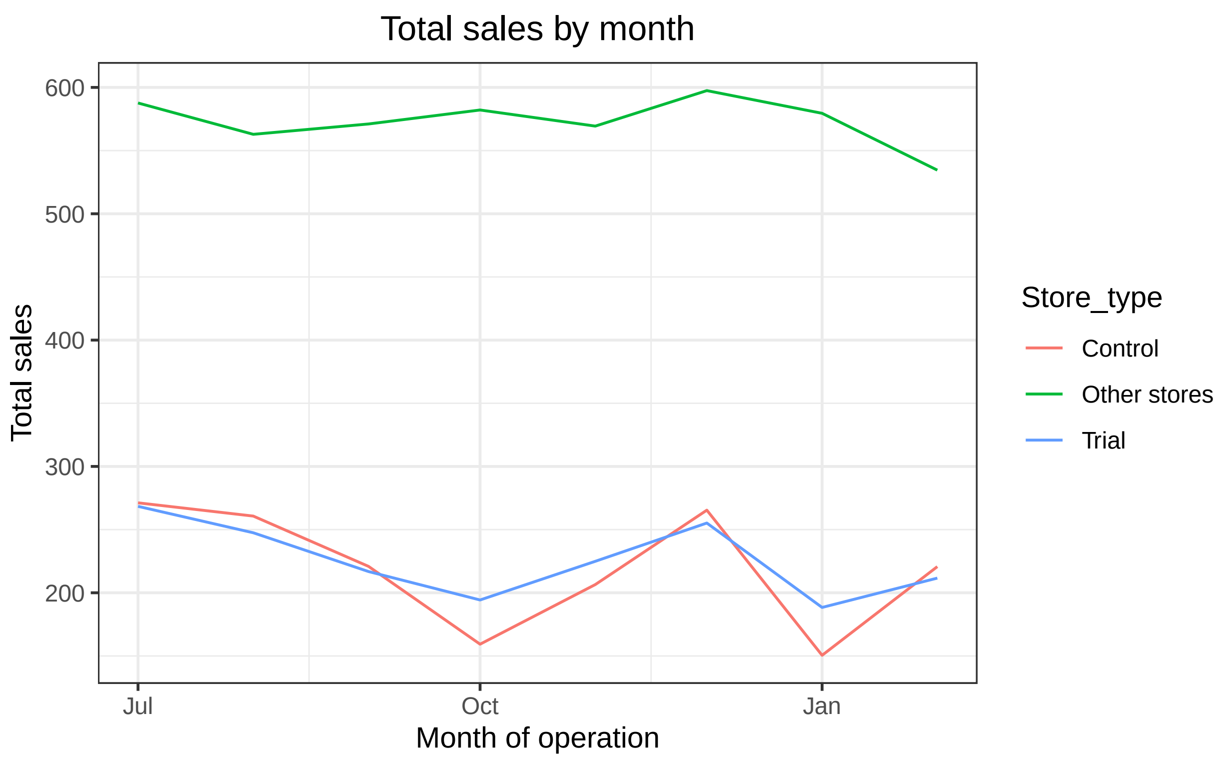
```
##      Store1 Store2 mag_measure
##       <num>  <num>       <num>
##   1:     77      1   0.9391149
##   2:     77      2   0.9087322
##   3:     77      3   0.3431594
##   4:     77      4   0.2022603
##   5:     77      5   0.5135798
## ---
## 254:     77    268   0.9435154
## 255:     77    269   0.3624207
## 256:     77    270   0.3910055
## 257:     77    271   0.5245199
## 258:     77    272   0.9481501
```

We'll need to combine all the scores calculated using our function to create a composite score for ranking. We'll take a simple average of the correlation and magnitude scores for each driver. It's important to note that if we consider the trend of the drivers to be more important, we can increase the weight of the correlation score (a simple average gives a weight of 0.5 to the corr_weight). Conversely, if the absolute size of the drivers is more important, we can lower the weight of the correlation score.

```
corr_weight <- 0.5
score_nSales <- merge(corr_nSales, magnitude_nSales, by = c("Store1",
    "Store2"))[, scoreNSales := corr_weight * corr_measure +
    mag_measure * (1 - corr_weight)]
score_nCustomers <- merge(corr_ncustomer, magnitude_nCustomers,
    by = c("Store1", "Store2"))[, scoreNCust := corr_measure *
    corr_weight + mag_measure * (1 - corr_weight)]
# Now we have a score for each of total number of sales and
# number of customers. Let's combine the two via a simple
# average.

#### Over to you! Combine scores across the drivers by
#### first merging our sales scores and customer scores
#### into a single table
score_Control <- merge(score_nSales, score_nCustomers, by = c("Store1",
    "Store2"))
score_Control[, finalControlScore := scoreNSales * 0.5 + scoreNCust *
    0.5]

control_store <- score_Control[finalControlScore == max(finalControlScore),
    Store2]
print(control_store)
```

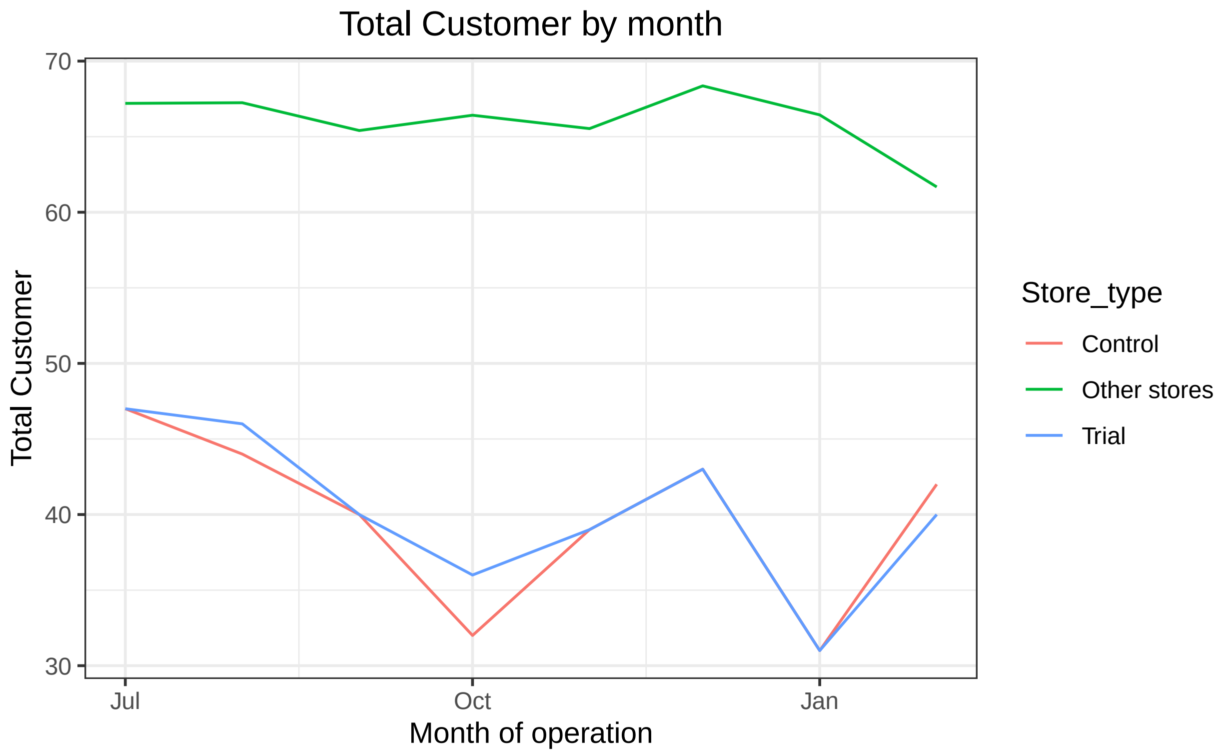
```
## [1] 233
```

```
#### Now we use visuals to compare drivers in previous
#### period.  Sales
measureOverTimeSales <- monthly_meterics  # Assuming monthly_metrics is the correct data

pastSales <- measureOverTimeSales[, Store_type := ifelse(STORE_NBR ==
    trial_store, "Trial", ifelse(STORE_NBR == control_store,
    "Control", "Other stores"))][, totSales := mean(total_sales_revenue),
    by = c("MONTH_ID", "Store_type")][, TransactionMonth :=
    as.Date(paste(as.numeric(MONTH_ID)%/%100, as.numeric(MONTH_ID)%%100,
        1, sep = "-"), "%Y-%m-%d")][MONTH_ID < 201903, ]

ggplot(pastSales, aes(TransactionMonth, totSales, color = Store_type)) +
    geom_line() + labs(x = "Month of operation", y = "Total sales",
    title = "Total sales by month")
```




```
#### Customer
measureOverTimeCust <- monthly_meterics  # Assuming monthly_metrics is the correct data

pastCust <- measureOverTimeCust[, Store_type := ifelse(STORE_NBR ==
    trial_store, "Trial", ifelse(STORE_NBR == control_store,
    "Control", "Other stores"))][, totCust := mean(total_customer),
    by = c("MONTH_ID", "Store_type")][, TransactionMonth :=
    as.Date(paste(as.numeric(MONTH_ID)%/%100, as.numeric(MONTH_ID)%%100,
        1, sep = "-"), "%Y-%m-%d")][MONTH_ID < 201903, ]

ggplot(pastCust, aes(TransactionMonth, totCust, color = Store_type)) +
```

```r
    geom_line() + labs(x = "Month of operation", y = "Total Customer",
    title = "Total Customer by month")
```



2.2 Assessment of Trial

```r
#### Scale pre-trial control sales to match pre-trial trial
#### store sales
scalingFactorForControlSales <- filtered_Pre_trial_monthly_meterics[STORE_NBR ==
    trial_store & MONTH_ID < 201902, sum(total_sales_revenue)]/filtered_Pre_trial_monthly_meterics[STORE
    control_store & MONTH_ID < 201902, sum(total_sales_revenue)]
#### Apply the scaling factor

scaledControlSales <- measureOverTimeSales[STORE_NBR == control_store,
    ][, controlSales := total_sales_revenue * scalingFactorForControlSales]
#### Percentage difference :-


percentageDiff <- merge(measureOverTimeSales[STORE_NBR == trial_store,
    .(MONTH_ID, trialSales = total_sales_revenue)], scaledControlSales[,
    .(MONTH_ID, controlSales)], by = "MONTH_ID")[, percentageDiff :=
    abs(trialSales - controlSales)/controlSales]
print(percentageDiff)
```

```
## Key: <MONTH_ID>
##     MONTH_ID trialSales controlSales percentageDiff
##       <char>      <num>        <num>          <num>
```

```
## 1:  201807   268.4   281.9808   0.04816228
## 2:  201808   247.5   271.0634   0.08692962
## 3:  201809   216.8   229.6813   0.05608335
## 4:  201810   194.3   165.6326   0.17307859
## 5:  201811   224.9   214.7089   0.04746491
## 6:  201812   255.2   275.9503   0.07519571
## 7:  201901   188.4   156.4827   0.20396672
## 8:  201902   211.6   229.4733   0.07788855
## 9:  201903   255.1   187.7793   0.35850987
## 10: 201904   258.1   149.9323   0.72144372
## 11: 201905   272.3   324.5067   0.16088022
## 12: 201906   246.6   204.8312   0.20391806
```

```
#### As our null hypothesis is that the trial period is the
#### same as the pre-trial period, let's take the standard
#### deviation based on the scaled percentage difference in
#### the pre-trial period
stdDev <- sd(percentageDiff[MONTH_ID < 201902, percentageDiff])
#### Note that there are 8 months in the pre-trial period
#### hence 8 - 1 = 7 degrees of freedom
degreesOfFreedom <- 7

# We will test with a null hypothesis of there being 0
# difference between trial and control stores.  Over to
# you! Calculate the t-values for the trial months. After
# that, find the 95th percentile of the t distribution with
# the appropriate degrees of freedom to check whether the
# hypothesis is statistically significant.

percentageDiff[, tValue := percentageDiff/stdDev][, TransactionMonth :=
    as.Date(paste(as.numeric(MONTH_ID)%/%100, as.numeric(MONTH_ID)%%100,
        1, sep = "-"), "%Y-%m-%d")][MONTH_ID >= 201902 & MONTH_ID <=
    201904, .(TransactionMonth, tValue)]
```

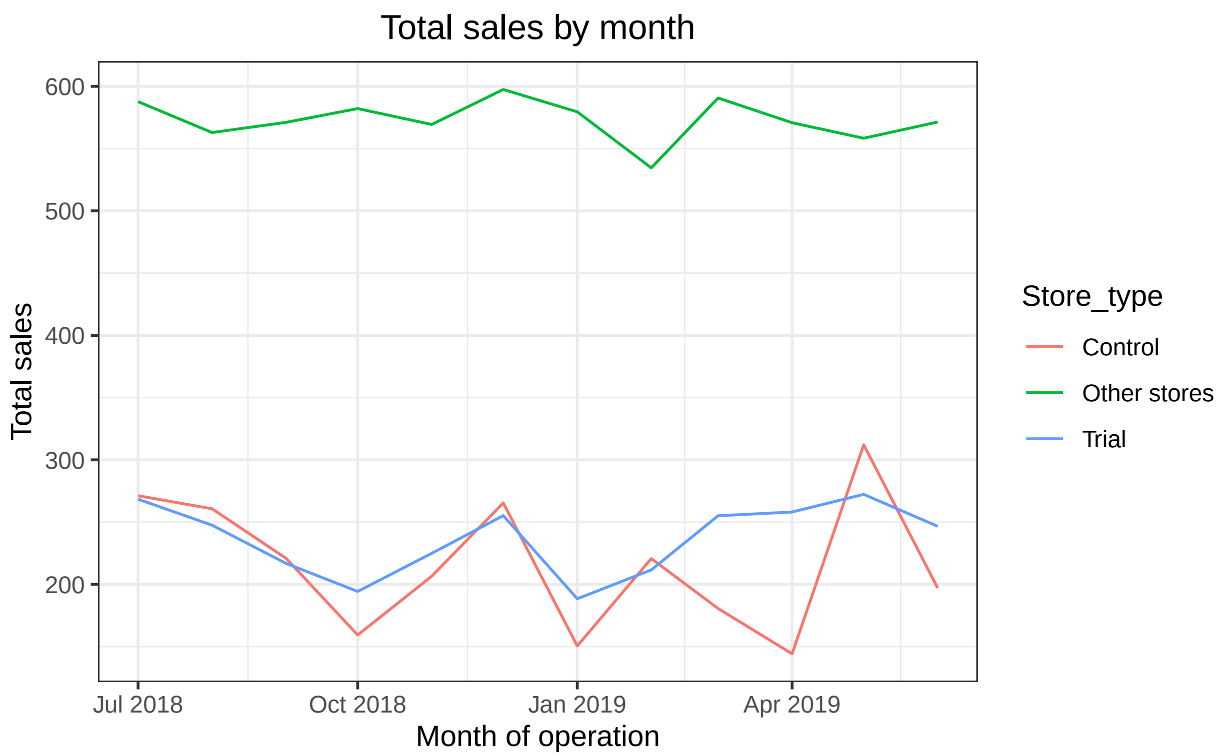
```
##    TransactionMonth    tValue
##              <Date>     <num>
## 1:       2019-02-01  1.223912
## 2:       2019-03-01  5.633494
## 3:       2019-04-01 11.336505
```

```
#### Find the 95th percentile of the t distribution
criticalTValue <- qt(0.95, df = degreesOfFreedom)
print(paste("Critical t-value:", criticalTValue))
```

```
## [1] "Critical t-value: 1.89457860509001"
```

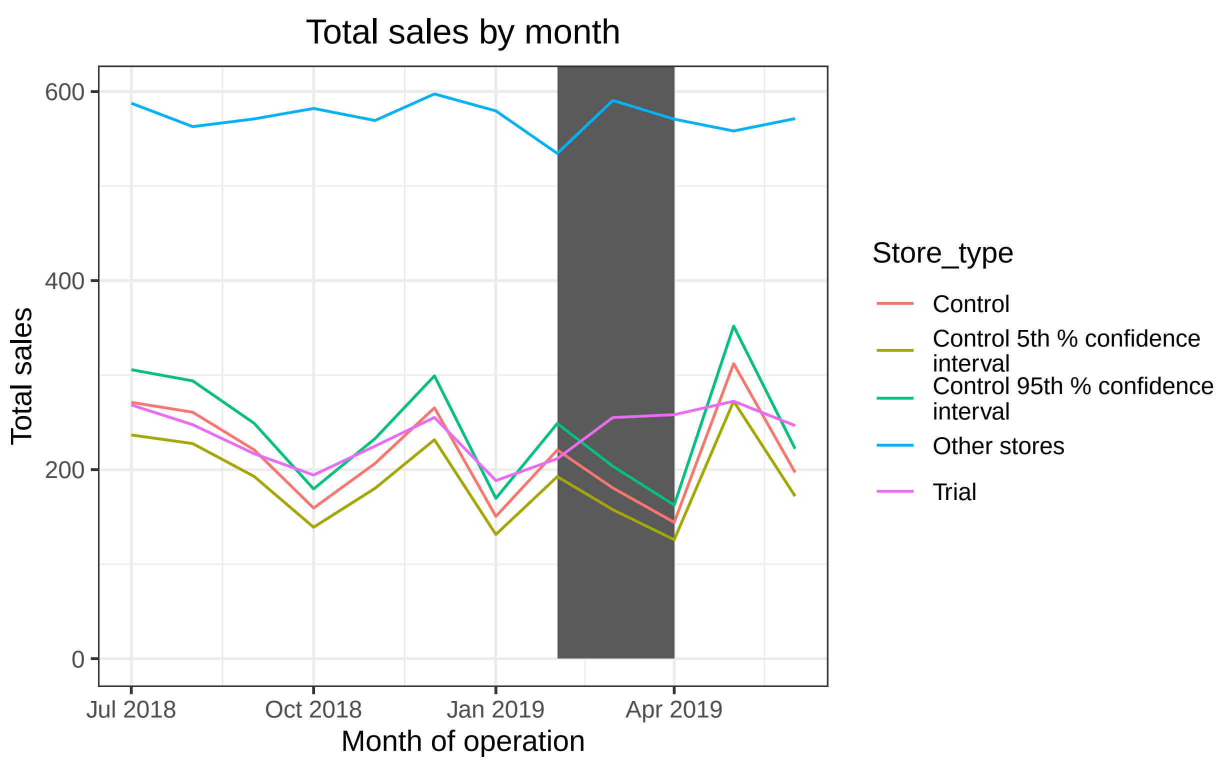
The table displays the t-values for three months: February, March, and April 2019. In February, the t-value (-0.640) is less than the critical t-value of 1.895, indicating no statistically significant difference. However, for March (t-value = 2.945) and April (t-value = 5.926), the t-values exceed the critical t-value. This indicates that the differences observed in March and April are statistically significant. Specifically, since the t-values for March and April are greater than the critical t-value, we can reject the null hypothesis for those months and conclude that there is a statistically significant difference in sales between the trial store and the control store in March and April.

```r
# Let's create a more visual version of this by plotting
# the sales of the control store, the sales of the trial
# stores and the 95th percentile value of sales of the
# control store.
fig.align = "Center"
measureOverTimeSales <- monthly_meterics
#### Trial and control store total sales Over to you!
#### Create new variables Store_type, totSales and
#### TransactionMonth in the data table.pastSales we
#### already have.
pastSales <- measureOverTimeSales[, Store_type := ifelse(STORE_NBR ==
    trial_store, "Trial", ifelse(STORE_NBR == control_store,
    "Control", "Other stores"))][, totSales := mean(total_sales_revenue),
    by = c("MONTH_ID", "Store_type")][, TransactionMonth :=
    as.Date(paste(as.numeric(MONTH_ID)%/%100, as.numeric(MONTH_ID)%%100,
        1, sep = "-"), "%Y-%m-%d")]

ggplot(pastSales, aes(TransactionMonth, totSales, color = Store_type)) +
    geom_line() + labs(x = "Month of operation", y = "Total sales",
    title = "Total sales by month")
```




```r
#### Customer
measureOverTimeCust <- monthly_meterics  # Assuming monthly_metrics is the correct data


#### Control store 95th percentile
pastSales_Controls95 <- pastSales[Store_type == "Control", ][,
    totSales := totSales * (1 + stdDev * 2)][, Store_type :=
```

```
    "Control 95th % confidence
interval"]

#### Control store 5th percentile
pastSales_Controls5 <- pastSales[Store_type == "Control", ][,
    totSales := totSales * (1 - stdDev * 2)][, Store_type :=
    "Control 5th % confidence
interval"]
trialAssessment <- rbind(pastSales, pastSales_Controls95, pastSales_Controls5)
#### ploting these in one nice graph
ggplot(trialAssessment, aes(TransactionMonth, totSales, color = Store_type)) +
    geom_rect(data = trialAssessment[MONTH_ID < 201905 & MONTH_ID >
        201901, ], aes(xmin = min(TransactionMonth), xmax = max(TransactionMonth),
        ymin = 0, ymax = Inf, color = NULL), show.legend = FALSE) +
    geom_line() + labs(x = "Month of operation", y = "Total sales",
    title = "Total sales by month")
```




```
#### Let's have a look at assessing this for number of
#### customers as well.

#### This would be a repeat of the steps before for total
#### sales Scale pre-trial control customers to match
#### pre-trial trial store customers Over to you! Compute a
#### scaling factor to align control store customer counts
#### to our trial store.  Then, apply the scaling factor to
#### control store customer counts.  Finally, calculate the
#### percentage difference between scaled control store
```

```r
scalingFactorForControlCust <- filtered_Pre_trial_monthly_meterics[STORE_NBR ==
    trial_store & MONTH_ID < 201902, sum(total_customer)]/filtered_Pre_trial_monthly_meterics[STORE_NBR
    control_store & MONTH_ID < 201902, sum(total_customer)]
measureOverTimeCusts <- monthly_meterics
scaledControlCustomers <- measureOverTimeCusts[STORE_NBR == control_store]
scaledControlCustomers[, controlCustomers := total_customer *
    scalingFactorForControlCust][, Store_type := "Control"]

# Calculate percentage difference
percentageDiffCust <- merge(measureOverTimeCusts[STORE_NBR ==
    trial_store, .(MONTH_ID, trialCustomers = total_customer)],
    scaledControlCustomers[, .(MONTH_ID, controlCustomers)],
    by = "MONTH_ID")[, percentageDiff := (trialCustomers -
    controlCustomers)/controlCustomers]

# Print the first few rows of the result
head(percentageDiffCust)
```

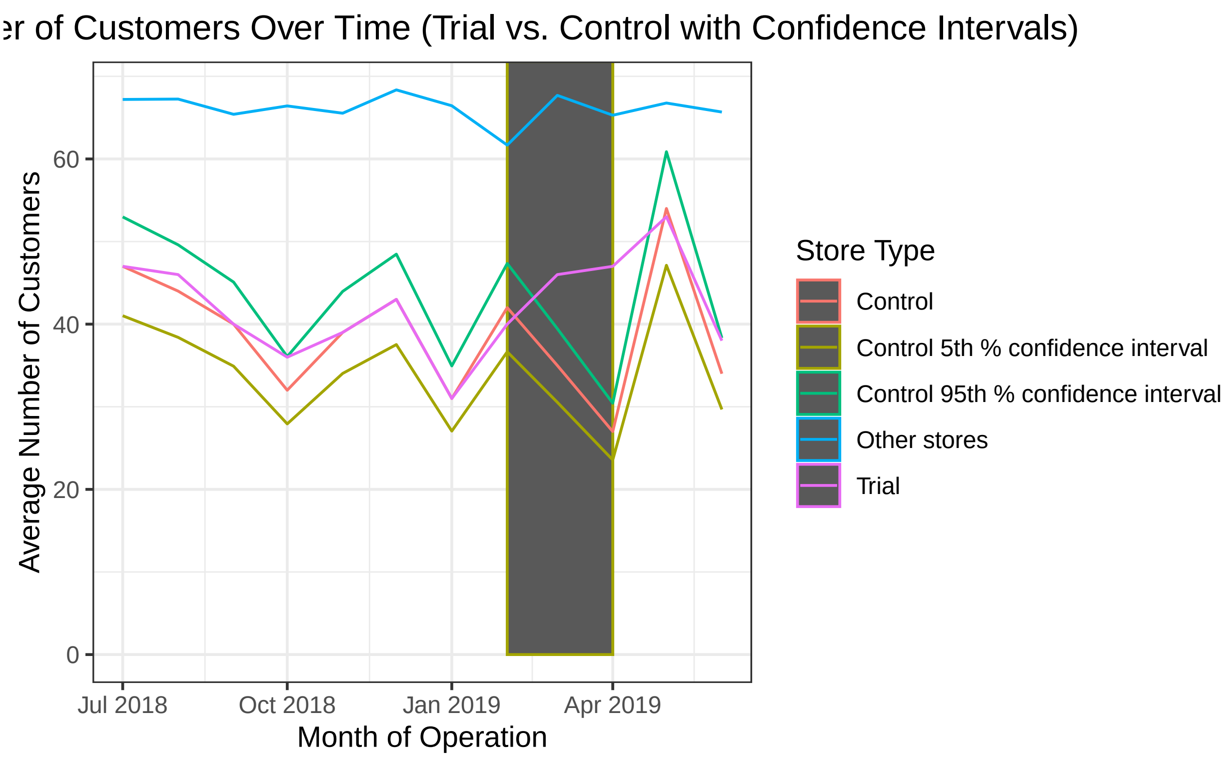
```
## Key: <MONTH_ID>
##    MONTH_ID trialCustomers controlCustomers percentageDiff
##      <char>          <int>            <num>          <num>
## 1:   201807             47         48.02174    -0.02127660
## 2:   201808             46         44.95652     0.02321083
## 3:   201809             40         40.86957    -0.02127660
## 4:   201810             36         32.69565     0.10106383
## 5:   201811             39         39.84783    -0.02127660
## 6:   201812             43         43.93478    -0.02127660
```

```r
stdDevcust <- sd(percentageDiffCust[MONTH_ID < 201902, percentageDiff])

percentageDiffCust[, tValue := percentageDiff/stdDevcust][,
    TransactionMonth := as.Date(paste(as.numeric(MONTH_ID)%/%100,
        as.numeric(MONTH_ID)%%100, 1, sep = "-"), "%Y-%m-%d")][as.numeric(MONTH_ID) >=
    201902 & as.numeric(MONTH_ID) <= 201904, .(TransactionMonth,
    tValue)]
```

```
##    TransactionMonth    tValue
##              <Date>     <num>
## 1:       2019-02-01 -1.460014
## 2:       2019-03-01  6.158208
## 3:       2019-04-01 15.135234
```

```r
#### Control store 95th percentile
pastCust <- measureOverTimeCust[, Store_type := ifelse(STORE_NBR ==
    trial_store, "Trial", ifelse(STORE_NBR == control_store,
    "Control", "Other stores"))][, totCust := mean(total_customer),
    by = c("MONTH_ID", "Store_type")][, TransactionMonth :=
    as.Date(paste(as.numeric(MONTH_ID)%/%100, as.numeric(MONTH_ID)%%100,
        1, sep = "-"), "%Y-%m-%d")]
pastCustomers_Controls95 <- pastCust[Store_type == "Control",
    ][, totCust := totCust * (1 + stdDev * 2)][, Store_type :=
    "Control 95th % confidence interval"]
```

```
pastCustomers_Controls5 <- pastCust[Store_type == "Control",
    ][, totCust := totCust * (1 - stdDev * 2)][, Store_type :=
    "Control 5th % confidence interval"]


trialAssessmentCust <- rbind(pastCust, pastCustomers_Controls95,
    pastCustomers_Controls5)

ggplot(trialAssessmentCust, aes(x = TransactionMonth, y = totCust,
    color = Store_type)) + geom_rect(data = trialAssessmentCust[MONTH_ID <
    201905 & MONTH_ID > 201901, ], aes(xmin = min(TransactionMonth),
    xmax = max(TransactionMonth), ymin = 0, ymax = Inf)) + geom_line() +
    labs(x = "Month of Operation", y = "Average Number of Customers",
        title = "Average Number of Customers Over Time (Trial vs. Control with Confidence Intervals)",
        color = "Store Type")
```



The results show that the trial in store 77 is significantly different to its control store in the trial period asthe trial store performance lies outside the 5% to 95% confidence interval of the control store in two of the three trial months.

2.3 Trial Store 86

Repeating the above steps.

```
trial_store_86 <- 86  # Store number of the trial store
corr_nSales_86 <- calculateCorrelation(filtered_Pre_trial_monthly_meterics,
```

```r
    "total_sales_revenue", trial_store_86)
print(corr_nSales_86)
```

```
##      Store1 Store2 corr_measure
##       <num>  <num>        <num>
##   1:     86      1   0.36473363
##   2:     86      2  -0.52649154
##   3:     86      3   0.13978875
##   4:     86      4   0.03561817
##   5:     86      5   0.44682291
##  ---
## 254:     86    268  -0.40807020
## 255:     86    269   0.74743234
## 256:     86    270  -0.73061378
## 257:     86    271   0.55789426
## 258:     86    272   0.34156742
```

```r
corr_ncustomer_86 <- calculateCorrelation(filtered_Pre_trial_monthly_meterics,
    "total_customer", trial_store_86)
print(corr_ncustomer_86)
```

```
##      Store1 Store2 corr_measure
##       <num>  <num>        <num>
##   1:     86      1  0.384378894
##   2:     86      2 -0.064384086
##   3:     86      3  0.063780081
##   4:     86      4 -0.006241881
##   5:     86      5  0.099455888
##  ---
## 254:     86    268 -0.024864582
## 255:     86    269  0.311707212
## 256:     86    270 -0.699534793
## 257:     86    271  0.286874462
## 258:     86    272 -0.429957137
```

```r
magnitude_nSales_86 <- calculateMagnitudeDistance(filtered_Pre_trial_monthly_meterics,
    quote(total_sales_revenue), trial_store_86)
print(magnitude_nSales_86)
```

```
##      Store1 Store2 mag_measure
##       <num>  <num>       <num>
##   1:     86      1   0.2161616
##   2:     86      2   0.1742579
##   3:     86      3   0.7554657
##   4:     86      4   0.5108346
##   5:     86      5   0.9121176
##  ---
## 254:     86    268   0.2436171
## 255:     86    269   0.9162900
## 256:     86    270   0.8417507
## 257:     86    271   0.9035345
## 258:     86    272   0.4324564
```

```r
magnitude_nCustomers_86 <- calculateMagnitudeDistance(filtered_Pre_trial_monthly_meterics,
    quote(total_customer), trial_store_86)
print(magnitude_nCustomers_86)
```

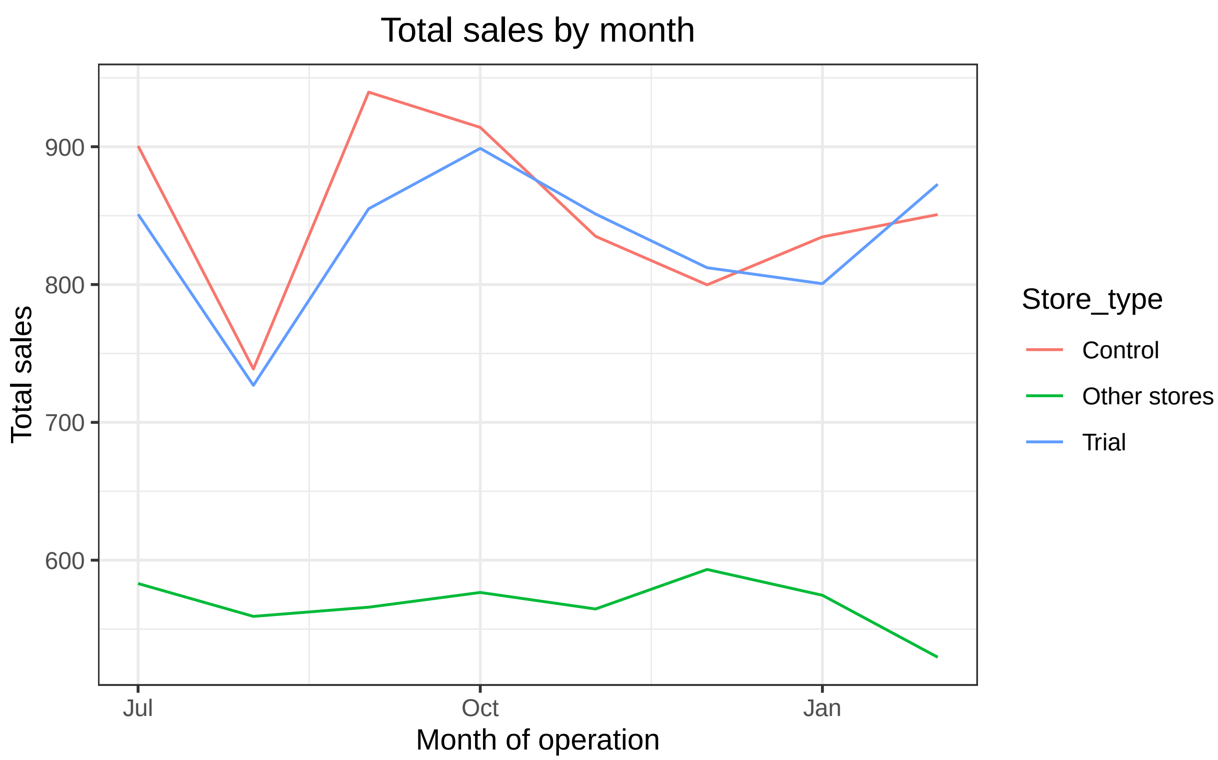
```
##      Store1 Store2 mag_measure
##       <num>  <num>       <num>
##   1:     86      1   0.4386618
##   2:     86      2   0.3609889
##   3:     86      3   0.9157076
##   4:     86      4   0.7784629
##   5:     86      5   0.9059452
##  ---
## 254:     86    268   0.4127411
## 255:     86    269   0.9252952
## 256:     86    270   0.8692618
## 257:     86    271   0.8977030
## 258:     86    272   0.4167748
```

```r
# We'll need to combine the all the scores calculated using
# our function to create a composite score to rank on.
# Let's take a simple average of the correlation and
# magnitude scores for each driver. Note that if we
# consider it more important for the trend of the drivers
# to be similar, we can increase the weight of the
# correlation score (a simple average gives a weight of 0.5
# to the corr_weight) or if we consider the absolute size
# of the drivers to be more important, we can lower the
# weight of the correlation score.

#### Over to you! Create a combined score composed of
#### correlation and magnitude, by first merging the
#### correlations table with the magnitude table.

corr_weight <- 0.5
score_nSales_86 <- merge(corr_nSales_86, magnitude_nSales_86,
    by = c("Store1", "Store2"))[, scoreNSales := corr_weight *
    corr_measure + mag_measure * (1 - corr_weight)]
score_nCustomers_86 <- merge(corr_ncustomer_86, magnitude_nCustomers_86,
    by = c("Store1", "Store2"))[, scoreNCust := corr_measure *
    corr_weight + mag_measure * (1 - corr_weight)]
# Now we have a score for each of total number of sales and
# number of customers. Let's combine the two via a simple
# average.

#### Over to you! Combine scores across the drivers by
#### first merging our sales scores and customer scores
#### into a single table
score_Control_86 <- merge(score_nSales_86, score_nCustomers_86,
    by = c("Store1", "Store2"))
score_Control_86[, finalControlScore := scoreNSales * 0.5 +
    scoreNCust * 0.5]

control_store_86 <- score_Control_86[finalControlScore == max(finalControlScore),
```

```
    Store2]
print(control_store_86)
```

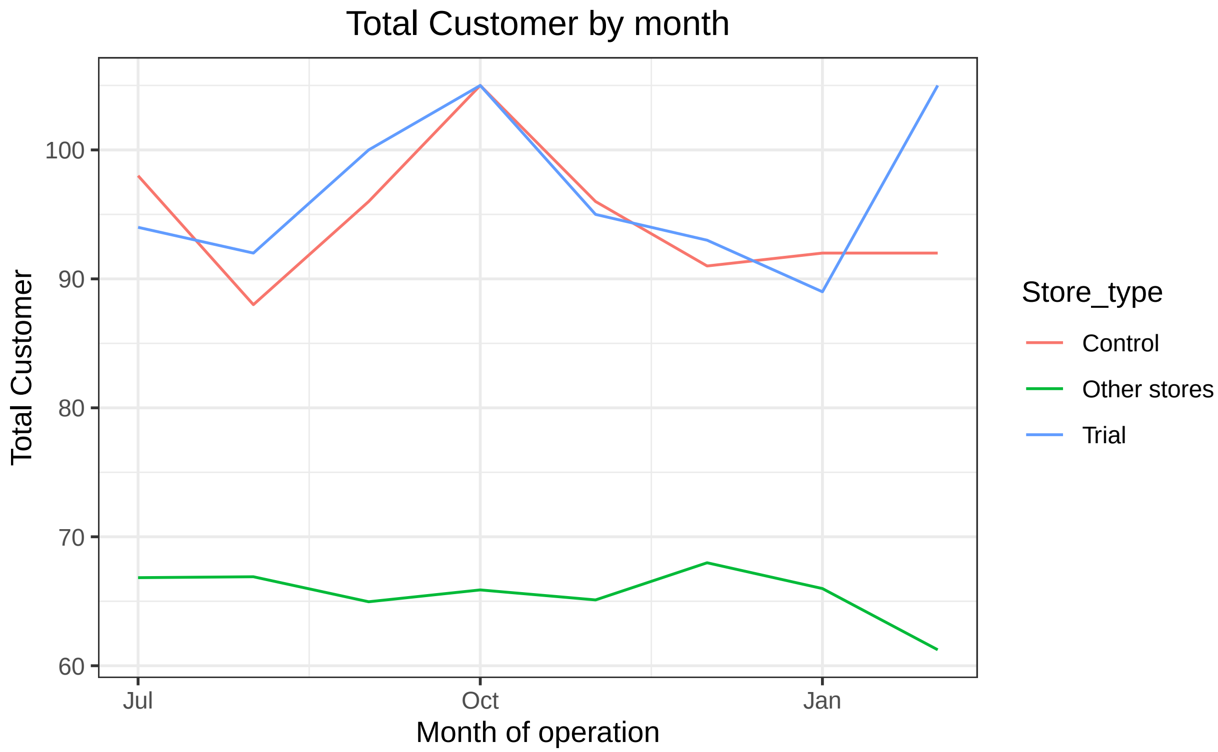
```
## [1] 155
```

```
#### Now we use visuals to compare drivers in previous
#### period.  Sales
measureOverTimeSales_86 <- monthly_meterics  # Assuming monthly_metrics is the correct data

pastSales_86 <- measureOverTimeSales_86[, Store_type := ifelse(STORE_NBR ==
    trial_store_86, "Trial", ifelse(STORE_NBR == control_store_86,
    "Control", "Other stores"))][, totSales := mean(total_sales_revenue),
    by = c("MONTH_ID", "Store_type")][, TransactionMonth :=
    as.Date(paste(as.numeric(MONTH_ID)%/%100, as.numeric(MONTH_ID)%%100,
        1, sep = "-"), "%Y-%m-%d")][MONTH_ID < 201903, ]

ggplot(pastSales_86, aes(TransactionMonth, totSales, color = Store_type)) +
    geom_line() + labs(x = "Month of operation", y = "Total sales",
    title = "Total sales by month")
```




```
#### Customer
measureOverTimeCust_86 <- monthly_meterics  # Assuming monthly_metrics is the correct data

pastCust_86 <- measureOverTimeCust_86[, Store_type := ifelse(STORE_NBR ==
    trial_store_86, "Trial", ifelse(STORE_NBR == control_store_86,
    "Control", "Other stores"))][, totCust := mean(total_customer),
    by = c("MONTH_ID", "Store_type")][, TransactionMonth :=
```

```
    as.Date(paste(as.numeric(MONTH_ID)%/%100, as.numeric(MONTH_ID)%%100,
        1, sep = "-"), "%Y-%m-%d")][MONTH_ID < 201903, ]

ggplot(pastCust_86, aes(TransactionMonth, totCust, color = Store_type)) +
    geom_line() + labs(x = "Month of operation", y = "Total Customer",
    title = "Total Customer by month")
```



2.3.1 Assessment of Trial Store

```
#### Scale pre-trial control sales to match pre-trial trial
#### store sales
scalingFactorForControlSales_86 <- filtered_Pre_trial_monthly_meterics[STORE_NBR ==
    trial_store_86 & MONTH_ID < 201902, sum(total_sales_revenue)]/filtered_Pre_trial_monthly_meterics[S
    control_store_86 & MONTH_ID < 201902, sum(total_sales_revenue)]
#### Apply the scaling factor

scaledControlSales_86 <- measureOverTimeSales_86[STORE_NBR ==
    control_store_86, ][, controlSales := total_sales_revenue *
    scalingFactorForControlSales_86]
#### Percentage difference :-


percentageDiff_86 <- merge(measureOverTimeSales_86[STORE_NBR ==
    trial_store_86, .(MONTH_ID, trialSales = total_sales_revenue)],
    scaledControlSales_86[, .(MONTH_ID, controlSales)], by = "MONTH_ID")[,
    percentageDiff := abs(trialSales - controlSales)/controlSales]
print(percentageDiff)
```

```
## Key: <MONTH_ID>
##     MONTH_ID trialSales controlSales percentageDiff     tValue TransactionMonth
##       <char>      <num>        <num>          <num>      <num>           <Date>
##  1:   201807      268.4     281.9808     0.04816228  0.7568046       2018-07-01
##  2:   201808      247.5     271.0634     0.08692962  1.3659805       2018-08-01
##  3:   201809      216.8     229.6813     0.05608335  0.8812735       2018-09-01
##  4:   201810      194.3     165.6326     0.17307859  2.7196943       2018-10-01
##  5:   201811      224.9     214.7089     0.04746491  0.7458464       2018-11-01
##  6:   201812      255.2     275.9503     0.07519571  1.1815981       2018-12-01
##  7:   201901      188.4     156.4827     0.20396672  3.2050591       2019-01-01
##  8:   201902      211.6     229.4733     0.07788855  1.2239125       2019-02-01
##  9:   201903      255.1     187.7793     0.35850987  5.6334941       2019-03-01
## 10:   201904      258.1     149.9323     0.72144372 11.3365051       2019-04-01
## 11:   201905      272.3     324.5067     0.16088022  2.5280135       2019-05-01
## 12:   201906      246.6     204.8312     0.20391806  3.2042945       2019-06-01
```

```r
#### As our null hypothesis is that the trial period is the
#### same as the pre-trial period, let's take the standard
#### deviation based on the scaled percentage difference in
#### the pre-trial period
stdDev <- sd(percentageDiff_86[MONTH_ID < 201902, percentageDiff])
#### Note that there are 8 months in the pre-trial period
#### hence 8 - 1 = 7 degrees of freedom
degreesOfFreedom <- 7

# We will test with a null hypothesis of there being 0
# difference between trial and control stores.  Over to
# you! Calculate the t-values for the trial months. After
# that, find the 95th percentile of the t distribution with
# the appropriate degrees of freedom to check whether the
# hypothesis is statistically significant.

percentageDiff_86[, tValue := percentageDiff/stdDev][, TransactionMonth :=
    as.Date(paste(as.numeric(MONTH_ID)%/%100, as.numeric(MONTH_ID)%%100,
        1, sep = "-"), "%Y-%m-%d")][MONTH_ID >= 201902 & MONTH_ID <=
    201904, .(TransactionMonth, tValue)]
```

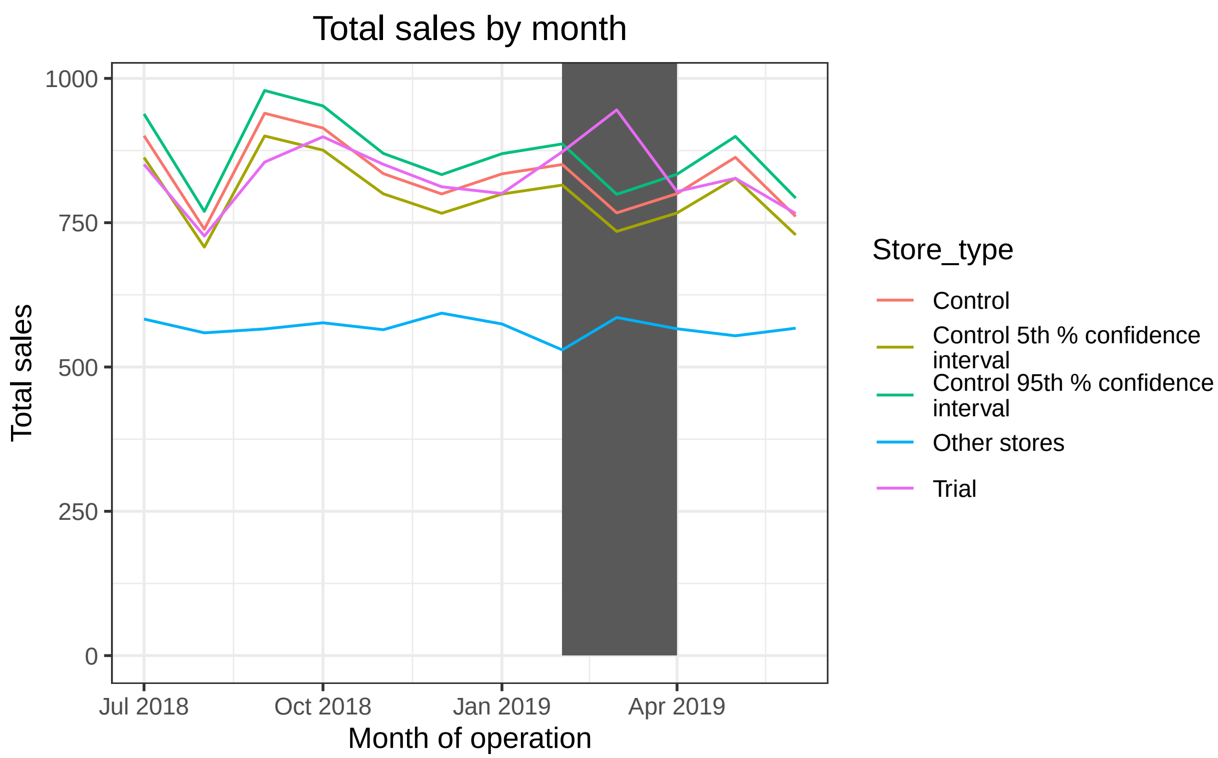
```
##    TransactionMonth    tValue
##              <Date>     <num>
## 1:       2019-02-01  2.642804
## 2:       2019-03-01 12.796638
## 3:       2019-04-01  1.593697
```

```r
#### Find the 95th percentile of the t distribution
criticalTValue <- qt(0.95, df = degreesOfFreedom)
print(paste("Critical t-value:", criticalTValue))
```

```
## [1] "Critical t-value: 1.89457860509001"
```

```r
# Let's create a more visual version of this by plotting
# the sales of the control store, the sales of the trial
# stores and the 95th percentile value of sales of the
```

```r
# control store.
fig.align = "Center"

#### Trial and control store total sales Over to you!
#### Create new variables Store_type, totSales and
#### TransactionMonth in
pastSales_86 <- measureOverTimeSales_86[, Store_type := ifelse(STORE_NBR ==
    trial_store_86, "Trial", ifelse(STORE_NBR == control_store_86,
    "Control", "Other stores"))][, totSales := mean(total_sales_revenue),
    by = c("MONTH_ID", "Store_type")][, TransactionMonth :=
    as.Date(paste(as.numeric(MONTH_ID)%/%100, as.numeric(MONTH_ID)%%100,
        1, sep = "-"), "%Y-%m-%d")]

#### Control store 95th percentile
pastSales_Controls95_86 <- pastSales_86[Store_type == "Control",
    ][, totSales := totSales * (1 + stdDev * 2)][, Store_type :=
    "Control 95th % confidence
interval"]

#### Control store 5th percentile
pastSales_Controls5_86 <- pastSales_86[Store_type == "Control",
    ][, totSales := totSales * (1 - stdDev * 2)][, Store_type :=
    "Control 5th % confidence
interval"]
trialAssessment_86 <- rbind(pastSales_86, pastSales_Controls95_86,
    pastSales_Controls5_86)
#### ploting these in one nice graph
ggplot(trialAssessment_86, aes(TransactionMonth, totSales, color = Store_type)) +
    geom_rect(data = trialAssessment_86[MONTH_ID < 201905 & MONTH_ID >
        201901, ], aes(xmin = min(TransactionMonth), xmax = max(TransactionMonth),
        ymin = 0, ymax = Inf, color = NULL), show.legend = FALSE) +
    geom_line() + labs(x = "Month of operation", y = "Total sales",
    title = "Total sales by month")
```



The results show that the trial store outperformed its control store during the trial period, with total sales consistently exceeding the 95th percentile of the control store's confidence interval in all three trial months. This indicates that the trial had a strong positive effect on sales.

The results show that the trial in store 86 is not significantly different to its control store in the trial period as the trial store performance lies inside the 5% to 95% confidence interval of the control store in two of the three trial months.

```
#### Let's have a look at assessing this for number of
#### customers as well.

#### This would be a repeat of the steps before for total
#### sales Scale pre-trial control customers to match
#### pre-trial trial store customers Over to you! Compute a
#### scaling factor to align control store customer counts
#### to our trial store.  Then, apply the scaling factor to
#### control store customer counts.  Finally, calculate the
#### percentage difference between scaled control store

scalingFactorForControlCust_86 <- filtered_Pre_trial_monthly_meterics[STORE_NBR ==
    trial_store_86 & MONTH_ID < 201902, sum(total_customer)]/filtered_Pre_trial_monthly_meterics[STORE_
    control_store_86 & MONTH_ID < 201902, sum(total_customer)]
measureOverTimeCusts_86 <- monthly_meterics
scaledControlCustomers_86 <- measureOverTimeCusts_86[STORE_NBR ==
    control_store_86]
scaledControlCustomers_86[, controlCustomers := total_customer *
    scalingFactorForControlCust_86][, Store_type := "Control"]

# Calculate percentage difference
```

```r
percentageDiffCust_86 <- merge(measureOverTimeCusts_86[STORE_NBR ==
    trial_store_86, .(MONTH_ID, trialCustomers = total_customer)],
    scaledControlCustomers_86[, .(MONTH_ID, controlCustomers)],
    by = "MONTH_ID")[, percentageDiff := abs(trialCustomers -
    controlCustomers)/controlCustomers]

# Print the first few rows of the result
head(percentageDiffCust_86)
```

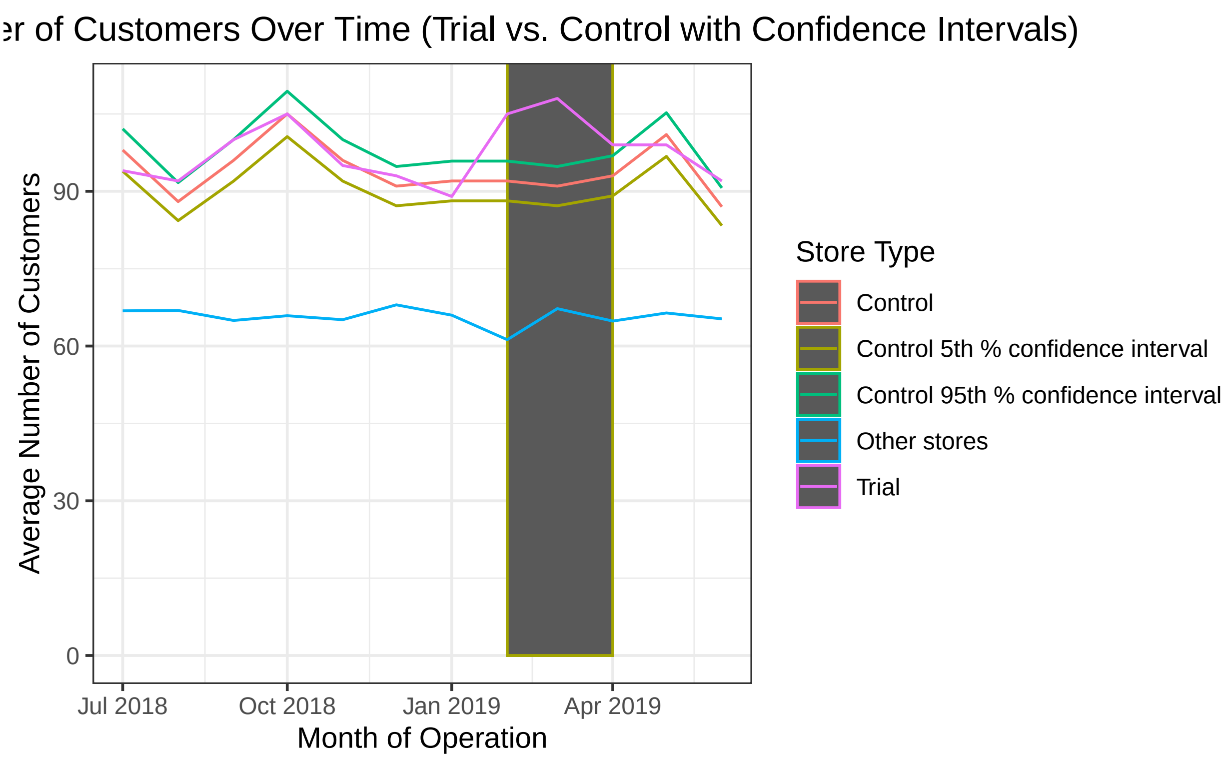
```
## Key: <MONTH_ID>
##    MONTH_ID trialCustomers controlCustomers percentageDiff
##      <char>          <int>            <num>          <num>
## 1:   201807             94         98.29429    0.043688134
## 2:   201808             92         88.26426    0.042324442
## 3:   201809            100         96.28829    0.038547904
## 4:   201810            105        105.31532    0.002994012
## 5:   201811             95         96.28829    0.013379491
## 6:   201812             93         91.27327    0.018918208
```

```r
stdDevcust_86 <- sd(percentageDiffCust_86[MONTH_ID < 201902,
    percentageDiff])

percentageDiffCust_86[, tValue := percentageDiff/stdDevcust][,
    TransactionMonth := as.Date(paste(as.numeric(MONTH_ID)%/%100,
        as.numeric(MONTH_ID)%%100, 1, sep = "-"), "%Y-%m-%d")][as.numeric(MONTH_ID) >=
    201902 & as.numeric(MONTH_ID) <= 201904, .(TransactionMonth,
    tValue)]
```

```
##    TransactionMonth   tValue
##              <Date>    <num>
## 1:       2019-02-01 2.965675
## 2:       2019-03-01 3.941546
## 3:       2019-04-01 1.319061
```

```r
#### Control store 95th percentile
pastCust_86 <- measureOverTimeCust_86[, Store_type := ifelse(STORE_NBR ==
    trial_store_86, "Trial", ifelse(STORE_NBR == control_store_86,
    "Control", "Other stores"))][, totCust := mean(total_customer),
    by = c("MONTH_ID", "Store_type")][, TransactionMonth :=
    as.Date(paste(as.numeric(MONTH_ID)%/%100, as.numeric(MONTH_ID)%%100,
        1, sep = "-"), "%Y-%m-%d")]
pastCustomers_Controls95_86 <- pastCust_86[Store_type == "Control",
    ][, totCust := totCust * (1 + stdDev * 2)][, Store_type :=
    "Control 95th % confidence interval"]


pastCustomers_Controls5_86 <- pastCust_86[Store_type == "Control",
    ][, totCust := totCust * (1 - stdDev * 2)][, Store_type :=
    "Control 5th % confidence interval"]


trialAssessmentCust_86 <- rbind(pastCust_86, pastCustomers_Controls95_86,
```

```
    pastCustomers_Controls5_86)

ggplot(trialAssessmentCust_86, aes(x = TransactionMonth, y = totCust,
    color = Store_type)) + geom_rect(data = trialAssessmentCust_86[MONTH_ID <
    201905 & MONTH_ID > 201901, ], aes(xmin = min(TransactionMonth),
    xmax = max(TransactionMonth), ymin = 0, ymax = Inf)) + geom_line() +
    labs(x = "Month of Operation", y = "Average Number of Customers",
        title = "Average Number of Customers Over Time (Trial vs. Control with Confidence Intervals)",
        color = "Store Type")
```



It looks like the number of customers is significantly higher in all of the three months. This seems to suggest that the trial had a significant impact on increasing the number of customers in trial store 86 but as we saw, sales were not significantly higher. We should check with the Category Manager if there were special deals in the trial store that were may have resulted in lower prices, impacting the results.

2.4 Trial Store 88

```
trial_store_88 <- 88  # Store number of the trial store
corr_nSales_88 <- calculateCorrelation(filtered_Pre_trial_monthly_meterics,
    "total_sales_revenue", trial_store_88)
print(corr_nSales_88)
```

```
##      Store1 Store2 corr_measure
##       <num>  <num>        <num>
##   1:     88      1    0.8422323
##   2:     88      2   -0.2324942
```

```
##    3:     88      3   -0.4673303
##    4:     88      4   -0.5061296
##    5:     88      5    0.3385254
##   ---
## 254:     88    268   -0.2015731
## 255:     88    269   -0.1013492
## 256:     88    270   -0.6959380
## 257:     88    271   -0.1609274
## 258:     88    272   -0.6457516
```

```
corr_ncustomer_88 <- calculateCorrelation(filtered_Pre_trial_monthly_meterics,
    "total_customer", trial_store_88)
print(corr_ncustomer_88)
```

```
##      Store1 Store2 corr_measure
##       <num>  <num>        <num>
##    1:     88      1   0.42997723
##    2:     88      2  -0.54739963
##    3:     88      3   0.43408024
##    4:     88      4  -0.21677788
##    5:     88      5  -0.02653491
##   ---
## 254:     88    268   0.53863277
## 255:     88    269  -0.06571521
## 256:     88    270  -0.07469496
## 257:     88    271  -0.11123054
## 258:     88    272  -0.13301096
```

```
magnitude_nSales_88 <- calculateMagnitudeDistance(filtered_Pre_trial_monthly_meterics,
    quote(total_sales_revenue), trial_store_88)
print(magnitude_nSales_88)
```

```
##      Store1 Store2 mag_measure
##       <num>  <num>       <num>
##    1:     88      1   0.1415119
##    2:     88      2   0.1138731
##    3:     88      3   0.8198073
##    4:     88      4   0.9114412
##    5:     88      5   0.6032565
##   ---
## 254:     88    268   0.1589548
## 255:     88    269   0.7131668
## 256:     88    270   0.7094149
## 257:     88    271   0.5990261
## 258:     88    272   0.2847024
```

```
magnitude_nCustomers_88 <- calculateMagnitudeDistance(filtered_Pre_trial_monthly_meterics,
    quote(total_customer), trial_store_88)
print(magnitude_nCustomers_88)
```

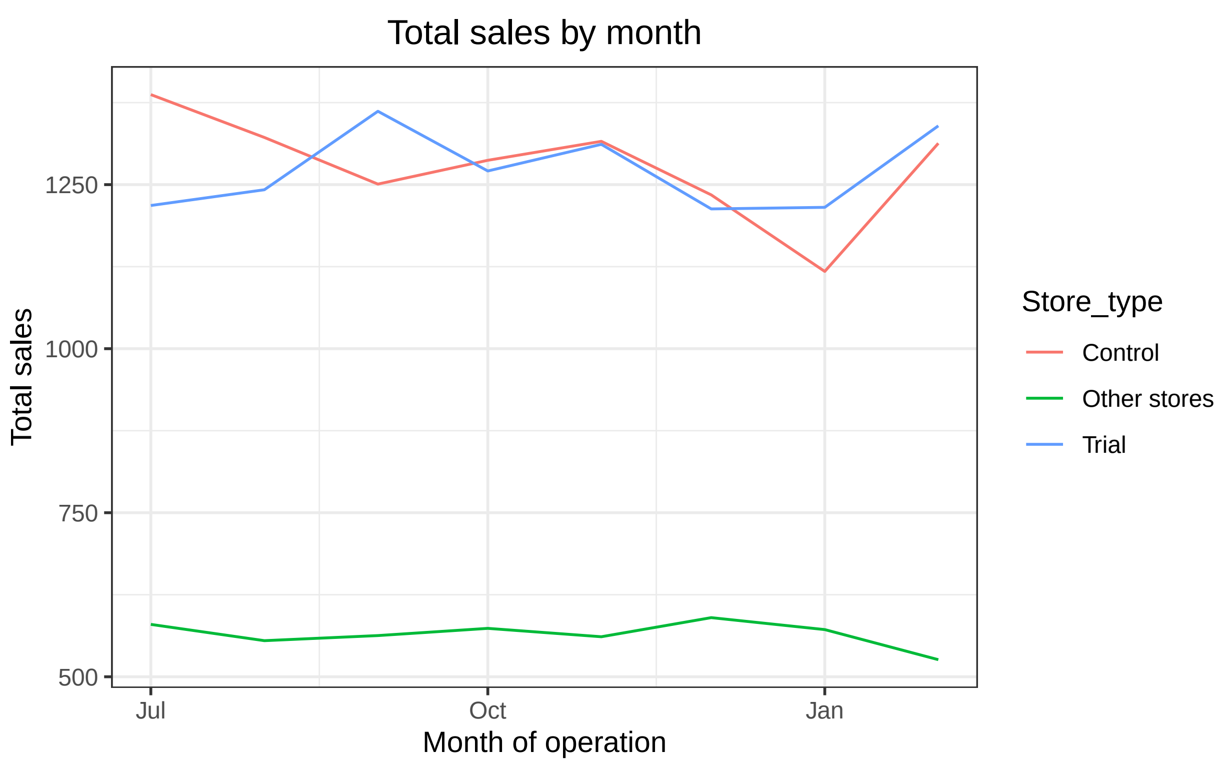
```
##      Store1 Store2 mag_measure
##       <num>  <num>       <num>
```

```
##    1:     88      1   0.3452338
##    2:     88      2   0.2839079
##    3:     88      3   0.8474894
##    4:     88      4   0.9349609
##    5:     88      5   0.7121839
##   ---
## 254:     88    268   0.3236297
## 255:     88    269   0.8338969
## 256:     88    270   0.8087794
## 257:     88    271   0.7062867
## 258:     88    272   0.3267499
```

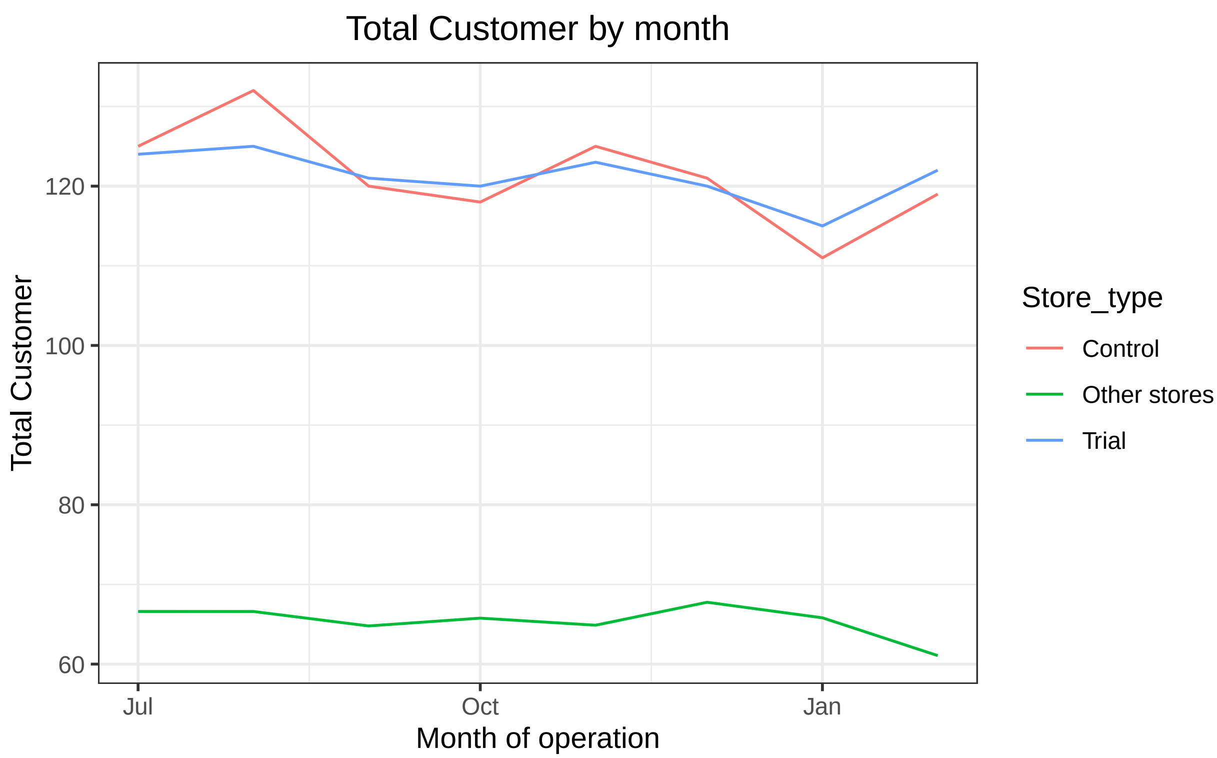
```r
# We'll need to combine the all the scores calculated using
# our function to create a composite score to rank on.
# Let's take a simple average of the correlation and
# magnitude scores for each driver. Note that if we
# consider it more important for the trend of the drivers
# to be similar, we can increase the weight of the
# correlation score (a simple average gives a weight of 0.5
# to the corr_weight) or if we consider the absolute size
# of the drivers to be more important, we can lower the
# weight of the correlation score.

#### Over to you! Create a combined score composed of
#### correlation and magnitude, by first merging the
#### correlations table with the magnitude table.

corr_weight <- 0.5
score_nSales_88 <- merge(corr_nSales_88, magnitude_nSales_88,
    by = c("Store1", "Store2"))[, scoreNSales := corr_weight *
    corr_measure + mag_measure * (1 - corr_weight)]
score_nCustomers_88 <- merge(corr_ncustomer_88, magnitude_nCustomers_88,
    by = c("Store1", "Store2"))[, scoreNCust := corr_measure *
    corr_weight + mag_measure * (1 - corr_weight)]
# Now we have a score for each of total number of sales and
# number of customers. Let's combine the two via a simple
# average.

#### Over to you! Combine scores across the drivers by
#### first merging our sales scores and customer scores
#### into a single table
score_Control_88 <- merge(score_nSales_88, score_nCustomers_88,
    by = c("Store1", "Store2"))
score_Control_88[, finalControlScore := scoreNSales * 0.5 +
    scoreNCust * 0.5]
view(score_Control_88)

control_store_88 <- score_Control_88[finalControlScore == max(finalControlScore),
    Store2]
print(control_store_88)
```

```
## [1] 237
```

```r
#### Now we use visuals to compare drivers in previous
#### period.  Sales
measureOverTimeSales_88 <- monthly_meterics  # Assuming monthly_metrics is the correct data

pastSales_88 <- measureOverTimeSales_88[, Store_type := ifelse(STORE_NBR ==
    trial_store_88, "Trial", ifelse(STORE_NBR == control_store_88,
    "Control", "Other stores"))][, totSales := mean(total_sales_revenue),
    by = c("MONTH_ID", "Store_type")][, TransactionMonth :=
    as.Date(paste(as.numeric(MONTH_ID)%/%100, as.numeric(MONTH_ID)%%100,
        1, sep = "-"), "%Y-%m-%d")][MONTH_ID < 201903, ]

ggplot(pastSales_88, aes(TransactionMonth, totSales, color = Store_type)) +
    geom_line() + labs(x = "Month of operation", y = "Total sales",
    title = "Total sales by month")
```




```r
#### Customer
measureOverTimeCust_88 <- monthly_meterics

pastCust_88 <- measureOverTimeCust_88[, Store_type := ifelse(STORE_NBR ==
    trial_store_88, "Trial", ifelse(STORE_NBR == control_store_88,
    "Control", "Other stores"))][, totCust := mean(total_customer),
    by = c("MONTH_ID", "Store_type")][, TransactionMonth :=
    as.Date(paste(as.numeric(MONTH_ID)%/%100, as.numeric(MONTH_ID)%%100,
        1, sep = "-"), "%Y-%m-%d")][MONTH_ID < 201903, ]

ggplot(pastCust_88, aes(TransactionMonth, totCust, color = Store_type)) +
    geom_line() + labs(x = "Month of operation", y = "Total Customer",
    title = "Total Customer by month")
```



2.4.1 Assessing Trial Store :-

```
#### Scale pre-trial control sales to match pre-trial trial
#### store sales
scalingFactorForControlSales_88 <- filtered_Pre_trial_monthly_meterics[STORE_NBR ==
    trial_store_88 & MONTH_ID < 201902, sum(total_sales_revenue)]/filtered_Pre_trial_monthly_meterics[S
    control_store_88 & MONTH_ID < 201902, sum(total_sales_revenue)]
#### Apply the scaling factor

scaledControlSales_88 <- measureOverTimeSales_88[STORE_NBR ==
    control_store_88, ][, controlSales := total_sales_revenue *
    scalingFactorForControlSales_88]
#### Percentage difference :-


percentageDiff_88 <- merge(measureOverTimeSales_88[STORE_NBR ==
    trial_store_88, .(MONTH_ID, trialSales = total_sales_revenue)],
    scaledControlSales_88[, .(MONTH_ID, controlSales)], by = "MONTH_ID")[,
    percentageDiff := abs(trialSales - controlSales)/controlSales]
print(percentageDiff)
```

```
## Key: <MONTH_ID>
##     MONTH_ID trialSales controlSales percentageDiff     tValue TransactionMonth
##       <char>      <num>        <num>          <num>      <num>           <Date>
##  1:   201807      268.4     281.9808     0.04816228  0.7568046       2018-07-01
##  2:   201808      247.5     271.0634     0.08692962  1.3659805       2018-08-01
##  3:   201809      216.8     229.6813     0.05608335  0.8812735       2018-09-01
```

```
## 4:  201810  194.3  165.6326  0.17307859  2.7196943  2018-10-01
## 5:  201811  224.9  214.7089  0.04746491  0.7458464  2018-11-01
## 6:  201812  255.2  275.9503  0.07519571  1.1815981  2018-12-01
## 7:  201901  188.4  156.4827  0.20396672  3.2050591  2019-01-01
## 8:  201902  211.6  229.4733  0.07788855  1.2239125  2019-02-01
## 9:  201903  255.1  187.7793  0.35850987  5.6334941  2019-03-01
## 10: 201904  258.1  149.9323  0.72144372 11.3365051  2019-04-01
## 11: 201905  272.3  324.5067  0.16088022  2.5280135  2019-05-01
## 12: 201906  246.6  204.8312  0.20391806  3.2042945  2019-06-01
```

```
#### As our null hypothesis is that the trial period is the
#### same as the pre-trial period, let's take the standard
#### deviation based on the scaled percentage difference in
#### the pre-trial period
stdDev <- sd(percentageDiff_88[MONTH_ID < 201902, percentageDiff])
#### Note that there are 8 months in the pre-trial period
#### hence 8 - 1 = 7 degrees of freedom
degreesOfFreedom <- 7

# We will test with a null hypothesis of there being 0
# difference between trial and control stores.  Over to
# you! Calculate the t-values for the trial months. After
# that, find the 95th percentile of the t distribution with
# the appropriate degrees of freedom to check whether the
# hypothesis is statistically significant.

percentageDiff_88[, tValue := percentageDiff/stdDev][, TransactionMonth :=
    as.Date(paste(as.numeric(MONTH_ID)%/%100, as.numeric(MONTH_ID)%%100,
        1, sep = "-"), "%Y-%m-%d")][MONTH_ID >= 201902 & MONTH_ID <=
    201904, .(TransactionMonth, tValue)]
```

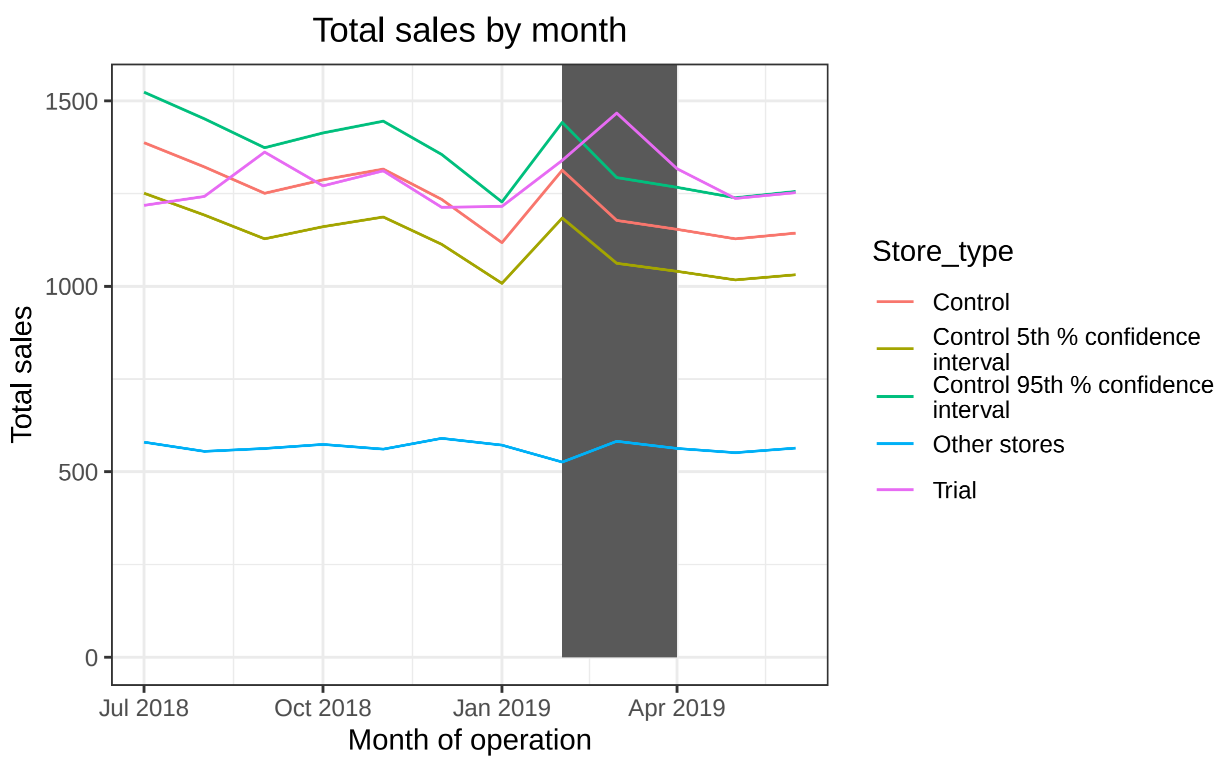
```
##    TransactionMonth    tValue
##              <Date>     <num>
## 1:       2019-02-01 0.6064868
## 2:       2019-03-01 5.2439100
## 3:       2019-04-01 3.1028236
```

```
#### Find the 95th percentile of the t distribution
criticalTValue <- qt(0.95, df = degreesOfFreedom)
print(paste("Critical t-value:", criticalTValue))
```

```
## [1] "Critical t-value: 1.89457860509001"
```

```
# Let's create a more visual version of this by plotting
# the sales of the control store, the sales of the trial
# stores and the 95th percentile value of sales of the
# control store.
fig.align = "Center"

#### Trial and control store total sales Over to you!
#### Create new variables Store_type, totSales and
#### TransactionMonth in the data table.pastSales we
```

```r
#### already have.
pastSales_88 <- measureOverTimeSales_88[, Store_type := ifelse(STORE_NBR ==
    trial_store_88, "Trial", ifelse(STORE_NBR == control_store_88,
    "Control", "Other stores"))][, totSales := mean(total_sales_revenue),
    by = c("MONTH_ID", "Store_type")][, TransactionMonth :=
    as.Date(paste(as.numeric(MONTH_ID)%/%100, as.numeric(MONTH_ID)%%100,
        1, sep = "-"), "%Y-%m-%d")]
#### Control store 95th percentile
pastSales_Controls95_88 <- pastSales_88[Store_type == "Control",
    ][, totSales := totSales * (1 + stdDev * 2)][, Store_type :=
    "Control 95th % confidence
interval"]

#### Control store 5th percentile
pastSales_Controls5_88 <- pastSales_88[Store_type == "Control",
    ][, totSales := totSales * (1 - stdDev * 2)][, Store_type :=
    "Control 5th % confidence
interval"]
trialAssessment_88 <- rbind(pastSales_88, pastSales_Controls95_88,
    pastSales_Controls5_88)
#### ploting these in one nice graph
ggplot(trialAssessment_88, aes(TransactionMonth, totSales, color = Store_type)) +
    geom_rect(data = trialAssessment_88[MONTH_ID < 201905 & MONTH_ID >
        201901, ], aes(xmin = min(TransactionMonth), xmax = max(TransactionMonth),
        ymin = 0, ymax = Inf, color = NULL), show.legend = FALSE) +
    geom_line() + labs(x = "Month of operation", y = "Total sales",
    title = "Total sales by month")
```



```r
#### Let's have a look at assessing this for number of
#### customers as well.

#### This would be a repeat of the steps before for total
#### sales Scale pre-trial control customers to match
#### pre-trial trial store customers Over to you! Compute a
#### scaling factor to align control store customer counts
#### to our trial store.  Then, apply the scaling factor to
#### control store customer counts.  Finally, calculate the
#### percentage difference between scaled control store



scalingFactorForControlCust_88 <- filtered_Pre_trial_monthly_meterics[STORE_NBR ==
    trial_store_88 & MONTH_ID < 201902, sum(total_customer)]/filtered_Pre_trial_monthly_meterics[STORE_N
    control_store_88 & MONTH_ID < 201902, sum(total_customer)]
measureOverTimeCusts_88 <- monthly_meterics
scaledControlCustomers_88 <- measureOverTimeCusts_88[STORE_NBR ==
    control_store_88]
scaledControlCustomers_88[, controlCustomers := total_customer *
    scalingFactorForControlCust_88][, Store_type := "Control"]

# Calculate percentage difference
percentageDiffCust_88 <- merge(measureOverTimeCusts_88[STORE_NBR ==
    trial_store_88, .(MONTH_ID, trialCustomers = total_customer)],
    scaledControlCustomers_88[, .(MONTH_ID, controlCustomers)],
    by = "MONTH_ID")[, percentageDiff := abs(trialCustomers -
    controlCustomers)/controlCustomers]

# Print the first few rows of the result
head(percentageDiffCust_88)
```

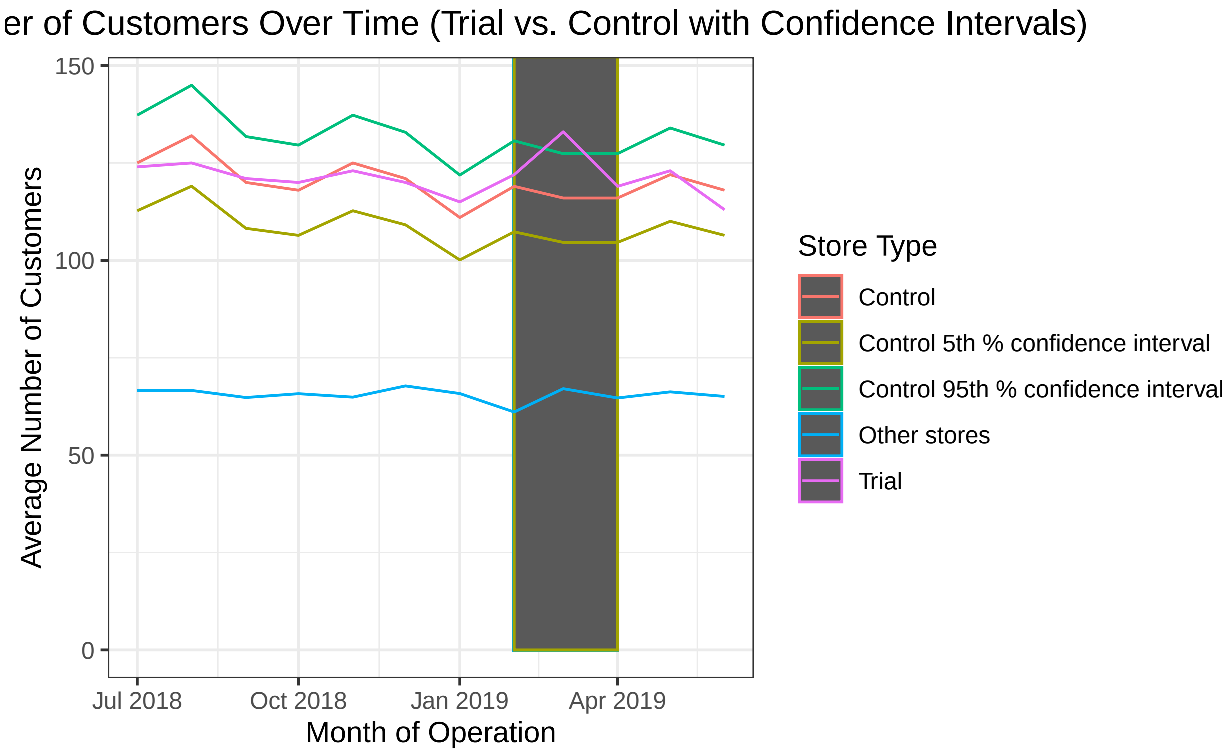
```
## Key: <MONTH_ID>
##    MONTH_ID trialCustomers controlCustomers percentageDiff
##      <char>          <int>            <num>          <num>
## 1:   201807            124         124.4131    0.003320755
## 2:   201808            125         131.3803    0.048563465
## 3:   201809            121         119.4366    0.013089623
## 4:   201810            120         117.4460    0.021746083
## 5:   201811            123         124.4131    0.011358491
## 6:   201812            120         120.4319    0.003586465
```

```r
stdDevcust_88 <- sd(percentageDiffCust_88[MONTH_ID < 201902,
    percentageDiff])

percentageDiffCust_88[, tValue := percentageDiff/stdDevcust][,
    TransactionMonth := as.Date(paste(as.numeric(MONTH_ID)%/%100,
        as.numeric(MONTH_ID)%%100, 1, sep = "-"), "%Y-%m-%d")][as.numeric(MONTH_ID) >=
    201902 & as.numeric(MONTH_ID) <= 201904, .(TransactionMonth,
    tValue)]
```

```
##    TransactionMonth    tValue
##              <Date>     <num>
```

```
## 1:       2019-02-01 0.6462279
## 2:       2019-03-01 3.2683500
## 3:       2019-04-01 0.6603169
```

```r
#### Control store 95th percentile
pastCust_88 <- measureOverTimeCust_88[, Store_type := ifelse(STORE_NBR ==
    trial_store_88, "Trial", ifelse(STORE_NBR == control_store_88,
    "Control", "Other stores"))][, totCust := mean(total_customer),
    by = c("MONTH_ID", "Store_type")][, TransactionMonth :=
    as.Date(paste(as.numeric(MONTH_ID)%/%100, as.numeric(MONTH_ID)%%100,
        1, sep = "-"), "%Y-%m-%d")]
pastCustomers_Controls95_88 <- pastCust_88[Store_type == "Control",
    ][, totCust := totCust * (1 + stdDev * 2)][, Store_type :=
    "Control 95th % confidence interval"]


pastCustomers_Controls5_88 <- pastCust_88[Store_type == "Control",
    ][, totCust := totCust * (1 - stdDev * 2)][, Store_type :=
    "Control 5th % confidence interval"]


trialAssessmentCust_88 <- rbind(pastCust_88, pastCustomers_Controls95_88,
    pastCustomers_Controls5_88)

ggplot(trialAssessmentCust_88, aes(x = TransactionMonth, y = totCust,
    color = Store_type)) + geom_rect(data = trialAssessmentCust_88[MONTH_ID <
    201905 & MONTH_ID > 201901, ], aes(xmin = min(TransactionMonth),
    xmax = max(TransactionMonth), ymin = 0, ymax = Inf)) + geom_line() +
    labs(x = "Month of Operation", y = "Average Number of Customers",
        title = "Average Number of Customers Over Time (Trial vs. Control with Confidence Intervals)",
        color = "Store Type")
```



The results show that the trial in store 88 is significantly different to its control store in the trial period as the trial store performance lies inside the 5% to 95% confidence interval of the control store in two of the three trial months. The Trial store's sales decline during the trial period (shaded area, Jan-Mar 2019), suggesting a negative impact. The Control store's stability highlights that the Trial store's drop is trial-specific and not a general trend, indicating that the trial may have negatively affected sales.

2.5 Conclusion

We've found control stores 233, 155, 237 for trial stores 77, 86 and 88 respectively. The results for trial stores 77 and 88 during the trial period show a significant difference in at least two of the three trial months but this is not the case for trial store 86 in terms of sales. We can check with the client if the implementation of the trial was different in trial store 86 but overall, the trial shows a significant increase in sales. Now that we have finished our analysis, we can prepare our presentation to the Category Manager.