Quantium Virtual Internship - Retail Strategy & Analytics - Task 2

Rounak Saha

2024-11-29

Load required libraries and datasets

```
options(repos = c(CRAN = "https://cloud.r-project.org"))
install.packages("tidyverse")
install.packages("ggplot2")
install.packages("lubridate")
install.packages("tidyr")
install.packages("data.table")
```

```
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   1.5.0
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     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
```

```
library(ggplot2)
library(lubridate)
library(tidyr)
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
```

```
# Load the dataset
data <- fread("/Users/ronsmackbook/Desktop/Quantium/QVI_data.csv", select=c(2:13))

## Set themes for plots
theme_update(plot.title = element_text(hjust = 0.5))
```

Select control stores

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.

```
# Select control stores
# Calculate these measures over time for each store
# Add a new month ID column in the data with the format yyyymm
data[, MONTHYEAR := year(DATE)*100 + month(DATE)]
data
```

```
##         LYLTY_CARD_NBR       DATE STORE_NBR TXN_ID PROD_NBR
##                  <int>     <IDat>     <int>  <num>    <int>
##      1:           1000 2018-10-17         1      1        5
##      2:           1002 2018-09-16         1      2       58
##      3:           1003 2019-03-07         1      3       52
##      4:           1003 2019-03-08         1      4      106
##      5:           1004 2018-11-02         1      5       96
##     ---
## 245298:        2370651 2018-08-03        88 240350        4
## 245299:        2370701 2018-12-08        88 240378       24
## 245300:        2370751 2018-10-01        88 240394       60
## 245301:        2370961 2018-10-24        88 240480       70
## 245302:        2373711 2018-12-14        88 241815       16
##                                      PROD_NAME PROD_QTY TOT_SALES packSize
##                                         <char>    <int>     <num>    <int>
##      1:   Natural Chip        Compny SeaSalt175g        2       6.0      175
##      2:    Red Rock Deli Chikn&Garlic Aioli 150g        1       2.7      150
##      3:    Grain Waves Sour    Cream&Chives 210G        1       3.6      210
##      4:   Natural ChipCo      Hony Soy Chckn175g        1       3.0      175
##      5:           WW Original Stacked Chips 160g        1       1.9      160
##     ---
```

```
## 245298:         Dorito Corn Chp     Supreme 380g        2      13.0      380
## 245299:    Grain Waves         Sweet Chilli 210g        2       7.2      210
## 245300:     Kettle Tortilla ChpsFeta&Garlic 150g        2       9.2      150
## 245301:  Tyrrells Crisps     Lightly Salted 165g        2       8.4      165
## 245302: Smiths Crinkle Chips Salt & Vinegar 330g        2      11.4      330
##              Brand              LIFESTAGE PREMIUM_CUSTOMER MONTHYEAR
##             <char>                 <char>           <char>     <num>
##      1:    NATURAL  YOUNG SINGLES/COUPLES          Premium    201810
##      2:        RRD  YOUNG SINGLES/COUPLES       Mainstream    201809
##      3:    GRNWVES         YOUNG FAMILIES           Budget    201903
##      4:    NATURAL         YOUNG FAMILIES           Budget    201903
##      5: WOOLWORTHS  OLDER SINGLES/COUPLES       Mainstream    201811
##     ---
## 245298:    DORITOS MIDAGE SINGLES/COUPLES       Mainstream    201808
## 245299:    GRNWVES         YOUNG FAMILIES       Mainstream    201812
## 245300:     KETTLE         YOUNG FAMILIES          Premium    201810
## 245301:   TYRRELLS         OLDER FAMILIES           Budget    201810
## 245302:     SMITHS  YOUNG SINGLES/COUPLES       Mainstream    201812
```

Next, we define the measure calculations to use during the analysis. For each store and month calculate total sales, number of customers, transactions per customer, chips per customer and the average price per unit.

```
#For each store and month calculate total sales, number of customers,
# transactions per customer, chips per customer and the average price per unit.
measureOverTime <- data[, .(TOTAL_SALES = sum(TOT_SALES),
         nCustomers =uniqueN(LYLTY_CARD_NBR),
         nTxnPerCust =uniqueN(TXN_ID)/uniqueN(LYLTY_CARD_NBR),
         nChipsPerTxn =sum(PROD_QTY)/uniqueN(TXN_ID),
         avgPricePerUnit = sum(TOT_SALES)/sum(PROD_QTY)),
         by = .(STORE_NBR,MONTHYEAR)][order(STORE_NBR,MONTHYEAR)]

measureOverTime
```

```
##       STORE_NBR MONTHYEAR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##           <int>     <num>       <num>      <int>       <num>        <num>
##    1:         1    201807       188.9         47    1.042553     1.183673
##    2:         1    201808       168.4         41    1.000000     1.268293
##    3:         1    201809       268.1         57    1.035088     1.203390
##    4:         1    201810       175.4         39    1.025641     1.275000
##    5:         1    201811       184.8         44    1.022727     1.222222
##   ---
## 3161:       272    201902       385.3         44    1.068182     1.893617
## 3162:       272    201903       421.9         48    1.062500     1.901961
## 3163:       272    201904       445.1         54    1.018519     1.909091
## 3164:       272    201905       314.6         34    1.176471     1.775000
## 3165:       272    201906       301.9         33    1.090909     1.888889
##       avgPricePerUnit
##                 <num>
##    1:        3.256897
##    2:        3.238462
##    3:        3.776056
##    4:        3.439216
##    5:        3.360000
```

```
##    ---
## 3161:        4.329213
## 3162:        4.349485
## 3163:        4.239048
## 3164:        4.430986
## 3165:        4.439706
```

```
# Filter to the pre-trial period and stores with full observation periods
storesWithFullObs <- unique(measureOverTime[, .N , STORE_NBR][N== 12, STORE_NBR])
preTrialMeasures <- measureOverTime[MONTHYEAR < 201902 & STORE_NBR %in%
storesWithFullObs, ]

storesWithFullObs
```

```
##   [1]   1   2   3   4   5   6   7   8   9  10  12  13  14  15  16  17  18  19
##  [19]  20  21  22  23  24  25  26  27  28  29  30  32  33  34  35  36  37  38
##  [37]  39  40  41  42  43  45  46  47  48  49  50  51  52  53  54  55  56  57
##  [55]  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75
##  [73]  77  78  79  80  81  82  83  84  86  87  88  89  90  91  93  94  95  96
##  [91]  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114
## [109] 115 116 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133
## [127] 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151
## [145] 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169
## [163] 170 171 172 173 174 175 176 178 179 180 181 182 183 184 185 186 187 188
## [181] 189 190 191 192 194 195 196 197 198 199 200 201 202 203 204 205 207 208
## [199] 209 210 212 213 214 215 216 217 219 220 221 222 223 224 225 226 227 228
## [217] 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246
## [235] 247 248 249 250 251 253 254 255 256 257 258 259 260 261 262 263 264 265
## [253] 266 267 268 269 270 271 272
```

```
preTrialMeasures
```

```
##       STORE_NBR MONTHYEAR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##           <int>     <num>       <num>      <int>       <num>        <num>
##    1:         1    201807       188.9         47    1.042553     1.183673
##    2:         1    201808       168.4         41    1.000000     1.268293
##    3:         1    201809       268.1         57    1.035088     1.203390
##    4:         1    201810       175.4         39    1.025641     1.275000
##    5:         1    201811       184.8         44    1.022727     1.222222
##   ---
## 1809:       272    201809       294.5         31    1.129032     1.971429
## 1810:       272    201810       405.1         41    1.146341     2.000000
## 1811:       272    201811       355.8         39    1.102564     1.930233
## 1812:       272    201812       363.1         43    1.000000     1.883721
## 1813:       272    201901       392.4         44    1.068182     1.914894
##       avgPricePerUnit
##                 <num>
##    1:        3.256897
##    2:        3.238462
##    3:        3.776056
##    4:        3.439216
##    5:        3.360000
##   ---
```

```
## 1809:        4.268116
## 1810:        4.309574
## 1811:        4.286747
## 1812:        4.482716
## 1813:        4.360000
```

We will construct a function computes how well a specific metric in the trial store correlates with the same metric in each potential control store, allowing the selection of stores most similar in performance to the trial store.

```
# Create function to calculate correlation
#Let's define inputTable as a metric table with potential comparison stores,
#metricCol as the store metric used to calculate correlation on, and storeComparison
#as the store number of the trial store.
calculate_corr <- function(input_table, metric_col, comparison_store){
  calc_corr_table <- data.table(trial_store= numeric() ,
                                comparison_store= numeric(), corr_calc= numeric())
  store_number <- unique(input_table[, STORE_NBR])

  for(i in store_number){
    calculated_measure <- data.table("trial_store"=
                                       comparison_store ,"comparison_store" = i,
                                     corr_calc= cor(input_table[STORE_NBR== comparison_store,
                                       eval(metric_col)],
                                       input_table[STORE_NBR== i,
                                        eval(metric_col)]))
    calc_corr_table <- rbind(calc_corr_table,calculated_measure)
  }
  return(calc_corr_table)
}
```

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.

```
# Create function to calculate magnitude distance
calculate_magnitude_distance <- function(input_table, metric_col, comparison_store){
calc_dist_table = data.table(trial_store = numeric(),
                             comparison_store = numeric(), MONTHYEAR =
numeric(), measure = numeric())
 store_number <- unique(input_table[, STORE_NBR])

 for (i in store_number) {
 calculated_measure = data.table("trial_store" = comparison_store
 , "comparison_store" = i
 , "MONTHYEAR" = input_table[STORE_NBR ==
comparison_store, MONTHYEAR]
 , "measure" = abs(input_table[STORE_NBR ==
comparison_store, eval(metric_col)]
 - input_table[STORE_NBR == i,
eval(metric_col)])
 )
 calc_dist_table <- rbind(calc_dist_table, calculated_measure)
 }
# Standardise the magnitude distance so that the measure ranges from 0 to 1
```

```
minMaxDist <- calc_dist_table[, .(minDist = min(measure), maxDist = max(measure)),
by = c("trial_store", "MONTHYEAR")]
 distTable <- merge(calc_dist_table, minMaxDist,
                    by = c("trial_store", "MONTHYEAR"))
#For each row in distTable:
#Standardize the measure using the below formula:
#This scales the distance to a range of [0, 1], where:
# 1 indicates the highest similarity (smallest distance).
# 0 indicates the lowest similarity (largest distance).
 distTable[, magnitudeMeasure := 1 - (measure - minDist)/(maxDist - minDist)]

 finalDistTable <- distTable[, .
                  (mag_measure = mean(magnitudeMeasure)), by =
.(trial_store, comparison_store)]
 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.

```
# Use functions to calculate correlation
trial_store <- 77
corr_nSales <- calculate_corr(preTrialMeasures, quote(TOTAL_SALES),trial_store )
corr_nCustomers <- calculate_corr(preTrialMeasures, quote(nCustomers),trial_store)
# Use functions to calculate correlation
magnitude_nSales <- calculate_magnitude_distance(preTrialMeasures, quote(TOTAL_SALES),
trial_store)
magnitude_nCustomers <- calculate_magnitude_distance(preTrialMeasures,
quote(nCustomers), trial_store)

corr_nSales
```

```
##      trial_store comparison_store   corr_calc
##            <num>            <num>       <num>
##   1:          77                1  0.01020652
##   2:          77                2 -0.24086601
##   3:          77                3  0.65629644
##   4:          77                4 -0.32320851
##   5:          77                5 -0.16464409
##  ---
## 255:          77              268  0.43035830
## 256:          77              269 -0.42325061
## 257:          77              270  0.31092862
## 258:          77              271  0.20661211
## 259:          77              272 -0.15986095
```

```
corr_nCustomers
```

```
##      trial_store comparison_store   corr_calc
##            <num>            <num>       <num>
##   1:          77                1  0.36131786
```

```
##   2:          77                2 -0.60429206
##   3:          77                3  0.73911351
##   4:          77                4 -0.25963818
##   5:          77                5  0.20376403
##  ---
## 255:          77              268  0.43820085
## 256:          77              269 -0.10738272
## 257:          77              270  0.04351795
## 258:          77              271  0.01059012
## 259:          77              272  0.11104792
```

```
magnitude_nSales
```

```
##      trial_store comparison_store mag_measure
##            <num>            <num>       <num>
##   1:          77                1   0.9522251
##   2:          77                2   0.9367420
##   3:          77                3   0.3446866
##   4:          77                4   0.1806104
##   5:          77                5   0.5702928
##  ---
## 255:          77              268   0.9616093
## 256:          77              269   0.4603068
## 257:          77              270   0.4623404
## 258:          77              271   0.5731995
## 259:          77              272   0.8909249
```

```
magnitude_nCustomers
```

```
##      trial_store comparison_store mag_measure
##            <num>            <num>       <num>
##   1:          77                1   0.9422514
##   2:          77                2   0.9104048
##   3:          77                3   0.3421515
##   4:          77                4   0.2014390
##   5:          77                5   0.5261624
##  ---
## 255:          77              268   0.9465978
## 256:          77              269   0.3731738
## 257:          77              270   0.3940384
## 258:          77              271   0.5259091
## 259:          77              272   0.9484559
```

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.

```
#Create a combined score composed of correlation and magnitude, by
#first merging the correlations table with the magnitude table.
# A simple average on the scores would be 0.5 * corr_measure + 0.5 *mag_measure
corr_weight <- 0.5
score_nSales <- merge(
  corr_nSales,
  magnitude_nSales,
  by = c("trial_store", "comparison_store")
)[, scoreNSales := corr_weight * corr_calc + corr_weight * mag_measure]

score_nCustomers <- merge(
  corr_nCustomers,
  magnitude_nCustomers,
  by = c("trial_store", "comparison_store")
)[, scoreNCust := corr_weight* corr_calc + corr_weight*mag_measure]

score_nSales
```

```
## Key: <trial_store, comparison_store>
##      trial_store comparison_store   corr_calc mag_measure scoreNSales
##            <num>            <num>       <num>       <num>       <num>
##   1:          77                1  0.01020652   0.9522251  0.48121579
##   2:          77                2 -0.24086601   0.9367420  0.34793800
##   3:          77                3  0.65629644   0.3446866  0.50049153
##   4:          77                4 -0.32320851   0.1806104 -0.07129907
##   5:          77                5 -0.16464409   0.5702928  0.20282434
##  ---
## 255:          77              268  0.43035830   0.9616093  0.69598381
## 256:          77              269 -0.42325061   0.4603068  0.01852812
## 257:          77              270  0.31092862   0.4623404  0.38663449
## 258:          77              271  0.20661211   0.5731995  0.38990580
## 259:          77              272 -0.15986095   0.8909249  0.36553197
```

```
score_nCustomers
```

```
## Key: <trial_store, comparison_store>
##      trial_store comparison_store   corr_calc mag_measure  scoreNCust
##            <num>            <num>       <num>       <num>       <num>
##   1:          77                1  0.36131786   0.9422514  0.65178465
##   2:          77                2 -0.60429206   0.9104048  0.15305636
##   3:          77                3  0.73911351   0.3421515  0.54063250
##   4:          77                4 -0.25963818   0.2014390 -0.02909958
##   5:          77                5  0.20376403   0.5261624  0.36496320
##  ---
## 255:          77              268  0.43820085   0.9465978  0.69239934
## 256:          77              269 -0.10738272   0.3731738  0.13289552
## 257:          77              270  0.04351795   0.3940384  0.21877817
## 258:          77              271  0.01059012   0.5259091  0.26824959
## 259:          77              272  0.11104792   0.9484559  0.52975190
```

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.

```
#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("trial_store","comparison_store"))
score_Control[, finalControlScore := scoreNSales * 0.5 + scoreNCust * 0.5]

score_Control
```

```
## Key: <trial_store, comparison_store>
##      trial_store comparison_store corr_calc.x mag_measure.x scoreNSales
##            <num>            <num>       <num>         <num>       <num>
##   1:          77                1  0.01020652     0.9522251  0.48121579
##   2:          77                2 -0.24086601     0.9367420  0.34793800
##   3:          77                3  0.65629644     0.3446866  0.50049153
##   4:          77                4 -0.32320851     0.1806104 -0.07129907
##   5:          77                5 -0.16464409     0.5702928  0.20282434
##  ---
## 255:          77              268  0.43035830     0.9616093  0.69598381
## 256:          77              269 -0.42325061     0.4603068  0.01852812
## 257:          77              270  0.31092862     0.4623404  0.38663449
## 258:          77              271  0.20661211     0.5731995  0.38990580
## 259:          77              272 -0.15986095     0.8909249  0.36553197
##      corr_calc.y mag_measure.y  scoreNCust finalControlScore
##            <num>         <num>       <num>             <num>
##   1:  0.36131786     0.9422514  0.65178465        0.56650022
##   2: -0.60429206     0.9104048  0.15305636        0.25049718
##   3:  0.73911351     0.3421515  0.54063250        0.52056202
##   4: -0.25963818     0.2014390 -0.02909958       -0.05019933
##   5:  0.20376403     0.5261624  0.36496320        0.28389377
##  ---
## 255:  0.43820085     0.9465978  0.69239934        0.69419158
## 256: -0.10738272     0.3731738  0.13289552        0.07571182
## 257:  0.04351795     0.3940384  0.21877817        0.30270633
## 258:  0.01059012     0.5259091  0.26824959        0.32907770
## 259:  0.11104792     0.9484559  0.52975190        0.44764194
```

The store with the highest score is then selected as the control store since it is most similar to the trial store.Select control stores based on the highest matching store (closest to 1 but not the store itself, i.e. the second ranked highest store)

```
#Select the most appropriate control store for trial store 77
#by finding the store with the highest final score.
control_store <-score_Control[order(-finalControlScore)][2,comparison_store]

control_store
```

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

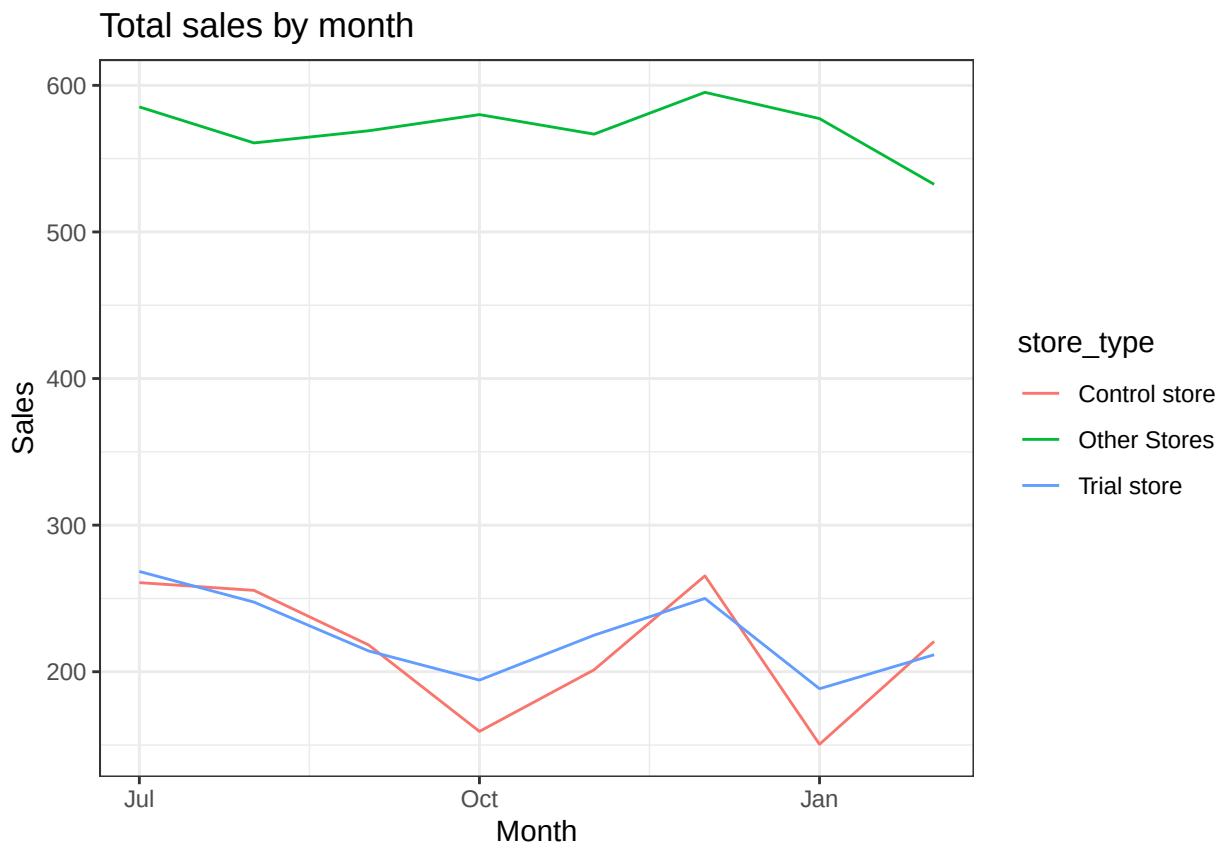
Now that we have found a control store, let's check visually if the drivers are indeed similar in the period before the trial.

We'll look at total sales first.

```
# Visual checks on monthly trends based on the total sales pre-trial
measurePreTrial_Sales <- measureOverTime[,
                   store_type:= ifelse(STORE_NBR== trial_store,
                          "Trial store", ifelse(STORE_NBR== control_store,
                                                "Control store", "Other Stores"))
                ][, .(sales= mean(TOTAL_SALES)),
                  by= .(store_type, MONTHYEAR)][, transaction_month:= as.Date(paste(
                  MONTHYEAR %/% 100, MONTHYEAR %% 100,1, sep ="-",
                  format= "%Y-%m-%d"))][ MONTHYEAR <201903,]

measurePreTrial_Sales
```

```
##        store_type MONTHYEAR    sales transaction_month
##            <char>     <num>    <num>            <Date>
##  1:  Other Stores    201807 585.2889        2018-07-01
##  2:  Other Stores    201808 560.7167        2018-08-01
##  3:  Other Stores    201809 568.9893        2018-09-01
##  4:  Other Stores    201810 580.0452        2018-10-01
##  5:  Other Stores    201811 566.7225        2018-11-01
##  6:  Other Stores    201812 595.2337        2018-12-01
##  7:  Other Stores    201901 577.3092        2019-01-01
##  8:  Other Stores    201902 532.4790        2019-02-01
##  9:   Trial store    201807 268.4000        2018-07-01
## 10:   Trial store    201808 247.5000        2018-08-01
## 11:   Trial store    201809 214.2000        2018-09-01
## 12:   Trial store    201810 194.3000        2018-10-01
## 13:   Trial store    201811 224.9000        2018-11-01
## 14:   Trial store    201812 250.0000        2018-12-01
## 15:   Trial store    201901 188.4000        2019-01-01
## 16:   Trial store    201902 211.6000        2019-02-01
## 17: Control store    201807 260.8000        2018-07-01
## 18: Control store    201808 255.5000        2018-08-01
## 19: Control store    201809 218.3000        2018-09-01
## 20: Control store    201810 159.3000        2018-10-01
## 21: Control store    201811 201.3000        2018-11-01
## 22: Control store    201812 265.4000        2018-12-01
## 23: Control store    201901 150.5000        2019-01-01
## 24: Control store    201902 220.7000        2019-02-01
##        store_type MONTHYEAR    sales transaction_month
```

```
ggplot(measurePreTrial_Sales, aes(x=transaction_month,y = sales, colour = store_type)) +
  geom_line()+ theme_bw() + labs(x= "Month", y = "Sales", title = "Total sales by month")
```



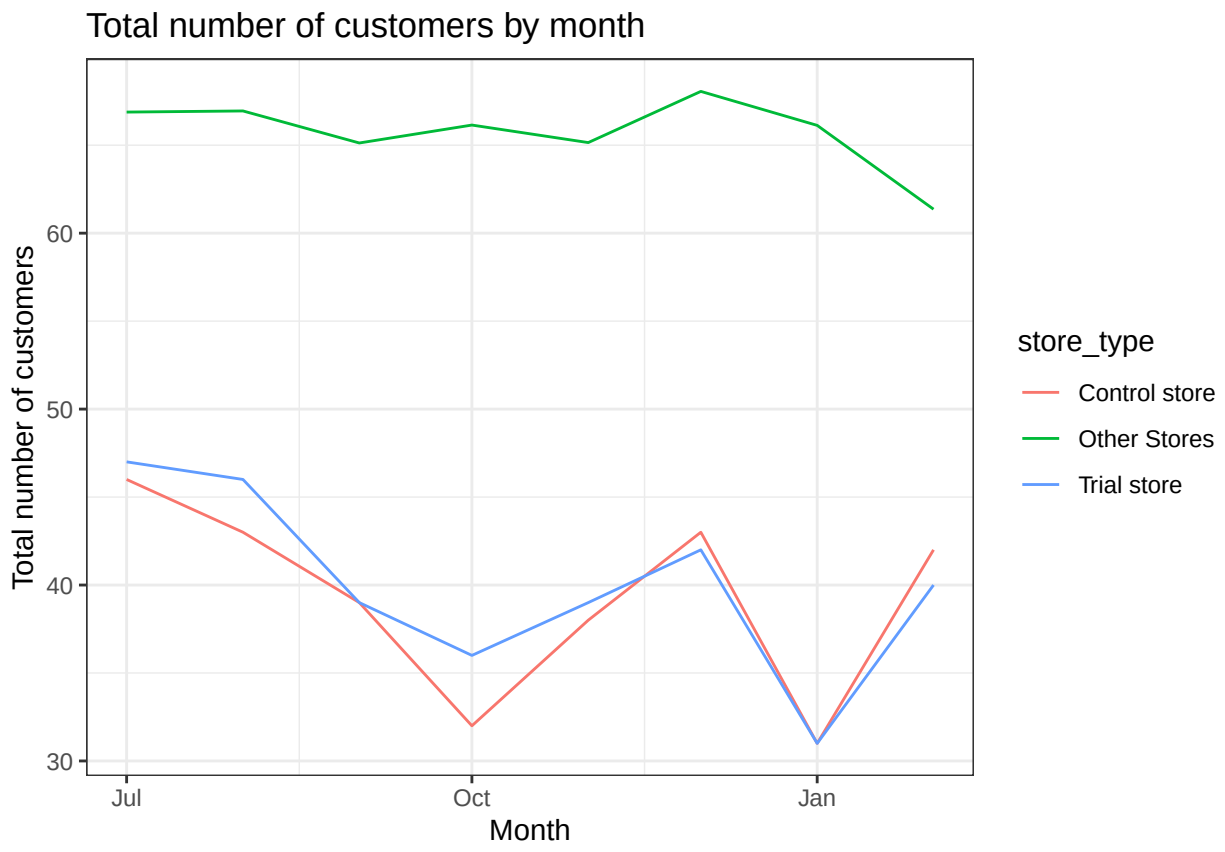
Next, number of customers

```
## Visual checks on monthly trends based on the total customers pre-trial
# Visual checks on monthly trends based on the total sales pre-trial
measurePreTrial_Customer <- measureOverTime[,
                                             store_type:= ifelse(STORE_NBR== trial_store,
                                                                 "Trial store",
                                                                 ifelse(STORE_NBR== control_store,
                                                                        "Control store","Other Stores"))
                ][, .(customer= mean(nCustomers)),
                  by= .(store_type, MONTHYEAR)][, transaction_month:= as.Date(paste(
                  MONTHYEAR %/% 100, MONTHYEAR %% 100,1, sep ="-",
                  format= "%Y-%m-%d"))][ MONTHYEAR <201903,]

measurePreTrial_Customer
```

```
##        store_type MONTHYEAR customer transaction_month
##            <char>     <num>    <num>            <Date>
##  1:  Other Stores    201807 66.88168        2018-07-01
##  2:  Other Stores    201808 66.94636        2018-08-01
##  3:  Other Stores    201809 65.12595        2018-09-01
##  4:  Other Stores    201810 66.14449        2018-10-01
##  5:  Other Stores    201811 65.14885        2018-11-01
##  6:  Other Stores    201812 68.05747        2018-12-01
##  7:  Other Stores    201901 66.12644        2019-01-01
##  8:  Other Stores    201902 61.36260        2019-02-01
```

```
##  9:   Trial store    201807 47.00000        2018-07-01
## 10:   Trial store    201808 46.00000        2018-08-01
## 11:   Trial store    201809 39.00000        2018-09-01
## 12:   Trial store    201810 36.00000        2018-10-01
## 13:   Trial store    201811 39.00000        2018-11-01
## 14:   Trial store    201812 42.00000        2018-12-01
## 15:   Trial store    201901 31.00000        2019-01-01
## 16:   Trial store    201902 40.00000        2019-02-01
## 17: Control store    201807 46.00000        2018-07-01
## 18: Control store    201808 43.00000        2018-08-01
## 19: Control store    201809 39.00000        2018-09-01
## 20: Control store    201810 32.00000        2018-10-01
## 21: Control store    201811 38.00000        2018-11-01
## 22: Control store    201812 43.00000        2018-12-01
## 23: Control store    201901 31.00000        2019-01-01
## 24: Control store    201902 42.00000        2019-02-01
##        store_type MONTHYEAR customer transaction_month
```

```
ggplot(measurePreTrial_Customer, aes(x=transaction_month,y = customer, colour = store_type)) +
  geom_line()+ theme_bw() + labs(x= "Month", y = "Total number of customers",
                                 title = "Total number of customers by month")
```



Assessment of trial

The trial period goes from the start of February 2019 to April 2019. We now want to see if there has been an uplift in overall chip sales.

We'll start with scaling the control store's sales to a level similar to control for any differences between the two stores outside of the trial period.

```
scalingFactor <- preTrialMeasures[
  STORE_NBR == trial_store & MONTHYEAR < 201902, sum(TOTAL_SALES)
]/ preTrialMeasures[
  STORE_NBR == control_store & MONTHYEAR < 201902, sum(TOTAL_SALES)
]
scalingFactor
```

```
## [1] 1.050692
```

```
# Apply the scaling factor
scaledControlSales<- measureOverTime[STORE_NBR ==
                                       control_store, ][,control_sales:=
                                         TOTAL_SALES *scalingFactor]
#Now that we have comparable sales figures for the control store, we can calculate
#the percentage difference between the scaled control sales and the trial store's
#sales during the trial period.
percentageDiff <- merge(scaledControlSales[, c("MONTHYEAR","control_sales")],
          measureOverTime[STORE_NBR==trial_store,
                          c("TOTAL_SALES","MONTHYEAR")],
          by= "MONTHYEAR")[, percentageDiff:= abs(control_sales-TOTAL_SALES)/control_sales]

scaledControlSales
```

```
##     STORE_NBR MONTHYEAR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##         <int>     <num>       <num>      <int>       <num>        <num>
##  1:       233    201807       260.8         46    1.021739     1.595745
##  2:       233    201808       255.5         43    1.023256     1.590909
##  3:       233    201809       218.3         39    1.076923     1.595238
##  4:       233    201810       159.3         32    1.000000     1.500000
##  5:       233    201811       201.3         38    1.026316     1.512821
##  6:       233    201812       265.4         43    1.046512     1.555556
##  7:       233    201901       150.5         31    1.000000     1.322581
##  8:       233    201902       220.7         42    1.023810     1.488372
##  9:       233    201903       178.0         34    1.029412     1.457143
## 10:       233    201904       144.2         27    1.037037     1.464286
## 11:       233    201905       312.1         54    1.037037     1.482143
## 12:       233    201906       197.0         34    1.000000     1.529412
##     avgPricePerUnit    store_type control_sales
##               <num>        <char>         <num>
##  1:        3.477333 Control store      274.0204
##  2:        3.650000 Control store      268.4517
##  3:        3.258209 Control store      229.3660
##  4:        3.318750 Control store      167.3752
##  5:        3.411864 Control store      211.5042
##  6:        3.791429 Control store      278.8535
```

```
##  7:        3.670732 Control store      158.1291
##  8:        3.448438 Control store      231.8876
##  9:        3.490196 Control store      187.0231
## 10:        3.517073 Control store      151.5097
## 11:        3.760241 Control store      327.9208
## 12:        3.788462 Control store      206.9862
```

```
percentageDiff
```

```
## Key: <MONTHYEAR>
##     MONTHYEAR control_sales TOTAL_SALES percentageDiff
##         <num>         <num>       <num>          <num>
##  1:    201807      274.0204       268.4     0.02051072
##  2:    201808      268.4517       247.5     0.07804641
##  3:    201809      229.3660       214.2     0.06612125
##  4:    201810      167.3752       194.3     0.16086518
##  5:    201811      211.5042       224.9     0.06333581
##  6:    201812      278.8535       250.0     0.10347201
##  7:    201901      158.1291       188.4     0.19143172
##  8:    201902      231.8876       211.6     0.08748904
##  9:    201903      187.0231       255.1     0.36400266
## 10:    201904      151.5097       258.1     0.70352105
## 11:    201905      327.9208       272.3     0.16961665
## 12:    201906      206.9862       236.2     0.14113868
```

Let's see if the difference is significant! This is to test whether the observed differences in sales between the trial store and the control store during the trial period are statistically significant.

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[MONTHYEAR < 201902 , percentageDiff])
stdDev
```

```
## [1] 0.05962571
```

```
# 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.

```
# Calculate the t-values for the trial months
percentageDiff[, tValue := (percentageDiff - 0)/stdDev
][, TransactionMonth := as.Date(paste(MONTHYEAR %/% 100,
                                      MONTHYEAR %% 100, 1, sep = "-"), "%Y-%m-%d")
][MONTHYEAR < 201905 & MONTHYEAR > 201901, .(TransactionMonth,tValue)]
```

```
##    TransactionMonth    tValue
##              <Date>     <num>
## 1:       2019-02-01  1.467304
## 2:       2019-03-01  6.104794
## 3:       2019-04-01 11.798954
```

```
# Find the 95th percentile of the t distribution with the appropriate
# degrees of freedom to compare against
qt(0.95, df = degreesOfFreedom)
```

```
## [1] 1.894579
```

We can observe that the t-value is much higher than the 95th percentile value of the t-distribution for March and April - i.e. the increase in sales in the trial store in March and April is statistically higher than in the control store.

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.

```
#Create new variables Store_type, totSales and TransactionMonth in the data table.
pastSales <- measureOverTime[, totSales := mean(TOTAL_SALES), by = c("MONTHYEAR", "store_type")
                    ][, TransactionMonth := as.Date(paste(MONTHYEAR %/% 100,
                                                          MONTHYEAR %% 100, 1,
                                                          sep = "-"), "%Y-%m-%d")
                    ][store_type %in% c("Trial store", "Control store"), ]
# Control store 95th percentile
pastSales_Controls95 <- pastSales[store_type == "Control store",
 ][, totSales := TOTAL_SALES * (1 + stdDev * 2)
 ][, store_type := "Control 95th % confidence interval"]
# Control store 5th percentile
pastSales_Controls5 <- pastSales[store_type == "Control store",
 ][, totSales := TOTAL_SALES * (1 - stdDev * 2)
 ][, store_type := "Control 5th % confidence interval"]

trialAssessment <- rbind(pastSales, pastSales_Controls95, pastSales_Controls5)

trialAssessment
```

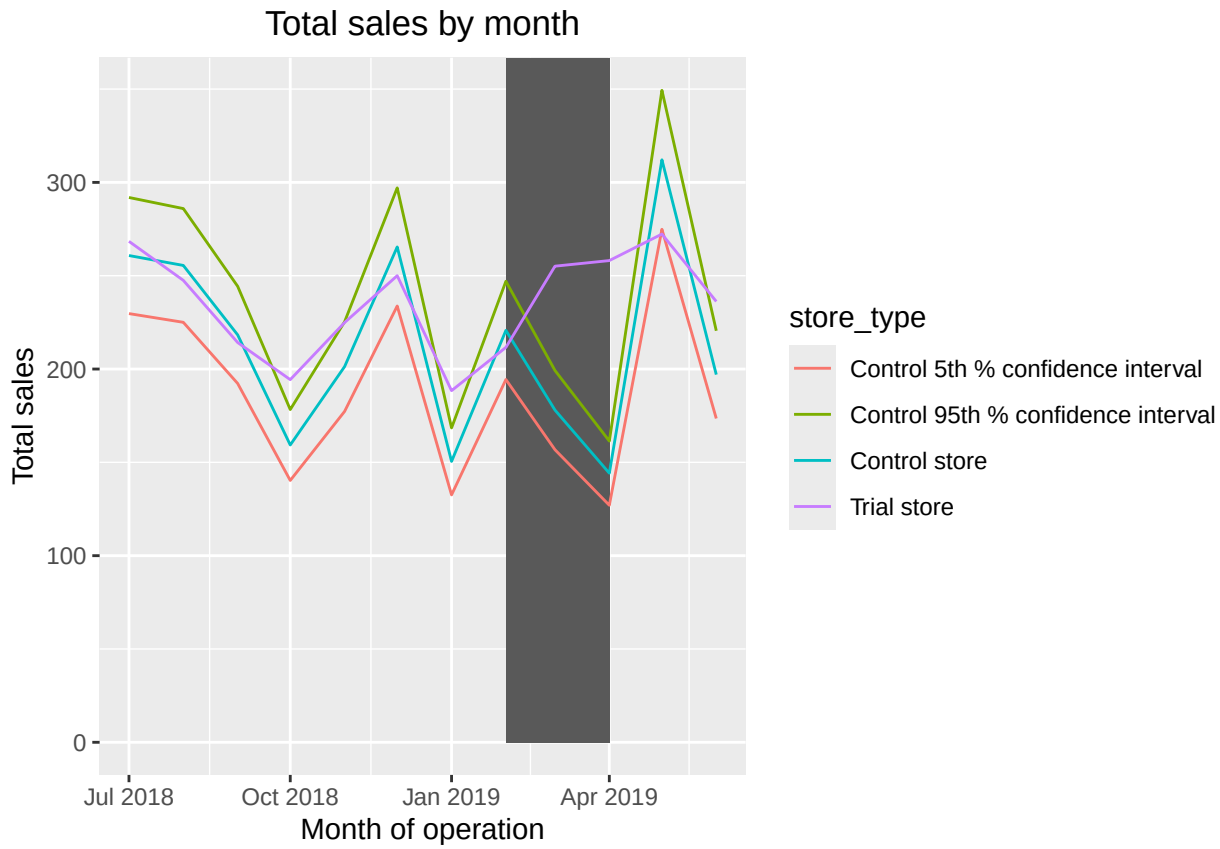
```
##     STORE_NBR MONTHYEAR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##         <int>     <num>       <num>      <int>       <num>        <num>
##  1:        77    201807       268.4         47    1.085106     1.509804
##  2:        77    201808       247.5         46    1.000000     1.543478
##  3:        77    201809       214.2         39    1.051282     1.585366
##  4:        77    201810       194.3         36    1.027778     1.351351
##  5:        77    201811       224.9         39    1.076923     1.500000
##  6:        77    201812       250.0         42    1.023810     1.511628
##  7:        77    201901       188.4         31    1.129032     1.657143
##  8:        77    201902       211.6         40    1.000000     1.675000
##  9:        77    201903       255.1         46    1.108696     1.490196
## 10:        77    201904       258.1         47    1.000000     1.617021
## 11:        77    201905       272.3         53    1.018868     1.444444
## 12:        77    201906       236.2         36    1.027778     1.648649
## 13:       233    201807       260.8         46    1.021739     1.595745
## 14:       233    201808       255.5         43    1.023256     1.590909
## 15:       233    201809       218.3         39    1.076923     1.595238
## 16:       233    201810       159.3         32    1.000000     1.500000
## 17:       233    201811       201.3         38    1.026316     1.512821
## 18:       233    201812       265.4         43    1.046512     1.555556
## 19:       233    201901       150.5         31    1.000000     1.322581
```

```
## 20:       233    201902       220.7         42    1.023810     1.488372
## 21:       233    201903       178.0         34    1.029412     1.457143
## 22:       233    201904       144.2         27    1.037037     1.464286
## 23:       233    201905       312.1         54    1.037037     1.482143
## 24:       233    201906       197.0         34    1.000000     1.529412
## 25:       233    201807       260.8         46    1.021739     1.595745
## 26:       233    201808       255.5         43    1.023256     1.590909
## 27:       233    201809       218.3         39    1.076923     1.595238
## 28:       233    201810       159.3         32    1.000000     1.500000
## 29:       233    201811       201.3         38    1.026316     1.512821
## 30:       233    201812       265.4         43    1.046512     1.555556
## 31:       233    201901       150.5         31    1.000000     1.322581
## 32:       233    201902       220.7         42    1.023810     1.488372
## 33:       233    201903       178.0         34    1.029412     1.457143
## 34:       233    201904       144.2         27    1.037037     1.464286
## 35:       233    201905       312.1         54    1.037037     1.482143
## 36:       233    201906       197.0         34    1.000000     1.529412
## 37:       233    201807       260.8         46    1.021739     1.595745
## 38:       233    201808       255.5         43    1.023256     1.590909
## 39:       233    201809       218.3         39    1.076923     1.595238
## 40:       233    201810       159.3         32    1.000000     1.500000
## 41:       233    201811       201.3         38    1.026316     1.512821
## 42:       233    201812       265.4         43    1.046512     1.555556
## 43:       233    201901       150.5         31    1.000000     1.322581
## 44:       233    201902       220.7         42    1.023810     1.488372
## 45:       233    201903       178.0         34    1.029412     1.457143
## 46:       233    201904       144.2         27    1.037037     1.464286
## 47:       233    201905       312.1         54    1.037037     1.482143
## 48:       233    201906       197.0         34    1.000000     1.529412
##     STORE_NBR MONTHYEAR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##     avgPricePerUnit                          store_type totSales
##               <num>                              <char>    <num>
##  1:       3.485714                       Trial store 268.4000
##  2:       3.485915                       Trial store 247.5000
##  3:       3.295385                       Trial store 214.2000
##  4:       3.886000                       Trial store 194.3000
##  5:       3.569841                       Trial store 224.9000
##  6:       3.846154                       Trial store 250.0000
##  7:       3.248276                       Trial store 188.4000
##  8:       3.158209                       Trial store 211.6000
##  9:       3.356579                       Trial store 255.1000
## 10:       3.396053                       Trial store 258.1000
## 11:       3.491026                       Trial store 272.3000
## 12:       3.872131                       Trial store 236.2000
## 13:       3.477333                     Control store 260.8000
## 14:       3.650000                     Control store 255.5000
## 15:       3.258209                     Control store 218.3000
## 16:       3.318750                     Control store 159.3000
## 17:       3.411864                     Control store 201.3000
## 18:       3.791429                     Control store 265.4000
## 19:       3.670732                     Control store 150.5000
## 20:       3.448438                     Control store 220.7000
## 21:       3.490196                     Control store 178.0000
## 22:       3.517073                     Control store 144.2000
```

```
## 23:        3.760241                      Control store 312.1000
## 24:        3.788462                      Control store 197.0000
## 25:        3.477333 Control 95th % confidence interval 291.9008
## 26:        3.650000 Control 95th % confidence interval 285.9687
## 27:        3.258209 Control 95th % confidence interval 244.3326
## 28:        3.318750 Control 95th % confidence interval 178.2968
## 29:        3.411864 Control 95th % confidence interval 225.3053
## 30:        3.791429 Control 95th % confidence interval 297.0493
## 31:        3.670732 Control 95th % confidence interval 168.4473
## 32:        3.448438 Control 95th % confidence interval 247.0188
## 33:        3.490196 Control 95th % confidence interval 199.2268
## 34:        3.517073 Control 95th % confidence interval 161.3961
## 35:        3.760241 Control 95th % confidence interval 349.3184
## 36:        3.788462 Control 95th % confidence interval 220.4925
## 37:        3.477333  Control 5th % confidence interval 229.6992
## 38:        3.650000  Control 5th % confidence interval 225.0313
## 39:        3.258209  Control 5th % confidence interval 192.2674
## 40:        3.318750  Control 5th % confidence interval 140.3032
## 41:        3.411864  Control 5th % confidence interval 177.2947
## 42:        3.791429  Control 5th % confidence interval 233.7507
## 43:        3.670732  Control 5th % confidence interval 132.5527
## 44:        3.448438  Control 5th % confidence interval 194.3812
## 45:        3.490196  Control 5th % confidence interval 156.7732
## 46:        3.517073  Control 5th % confidence interval 127.0039
## 47:        3.760241  Control 5th % confidence interval 274.8816
## 48:        3.788462  Control 5th % confidence interval 173.5075
##     avgPricePerUnit                           store_type totSales
##     TransactionMonth
##                <Date>
##  1:      2018-07-01
##  2:      2018-08-01
##  3:      2018-09-01
##  4:      2018-10-01
##  5:      2018-11-01
##  6:      2018-12-01
##  7:      2019-01-01
##  8:      2019-02-01
##  9:      2019-03-01
## 10:      2019-04-01
## 11:      2019-05-01
## 12:      2019-06-01
## 13:      2018-07-01
## 14:      2018-08-01
## 15:      2018-09-01
## 16:      2018-10-01
## 17:      2018-11-01
## 18:      2018-12-01
## 19:      2019-01-01
## 20:      2019-02-01
## 21:      2019-03-01
## 22:      2019-04-01
## 23:      2019-05-01
## 24:      2019-06-01
## 25:      2018-07-01
```

```
## 26:       2018-08-01
## 27:       2018-09-01
## 28:       2018-10-01
## 29:       2018-11-01
## 30:       2018-12-01
## 31:       2019-01-01
## 32:       2019-02-01
## 33:       2019-03-01
## 34:       2019-04-01
## 35:       2019-05-01
## 36:       2019-06-01
## 37:       2018-07-01
## 38:       2018-08-01
## 39:       2018-09-01
## 40:       2018-10-01
## 41:       2018-11-01
## 42:       2018-12-01
## 43:       2019-01-01
## 44:       2019-02-01
## 45:       2019-03-01
## 46:       2019-04-01
## 47:       2019-05-01
## 48:       2019-06-01
##     TransactionMonth
```

```r
# Plotting these in one nice graph
ggplot(trialAssessment, aes(TransactionMonth, totSales, color = store_type)) +
geom_rect(data = trialAssessment[ MONTHYEAR < 201905 & MONTHYEAR > 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 in store 77 is significantly different to its control store in the trial period as the trial store performance lies outside the 5% to 95% confidence interval of the control store in March and April months.

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

```
scalingFactor_cust <- preTrialMeasures[
  STORE_NBR == trial_store & MONTHYEAR < 201902, sum(nCustomers)
]/ preTrialMeasures[
  STORE_NBR == control_store & MONTHYEAR < 201902, sum(nCustomers)
]
scalingFactor_cust
```

```
## [1] 1.029412
```

```
# Apply the scaling factor
scaledControlCust <- measureOverTime[STORE_NBR ==
                                       control_store, ][, control_cust :=
                                                          nCustomers *
                                                          scalingFactor_cust]
#Now that we have comparable sales figures for the control store, we can calculate
#the percentage difference between the scaled control customer number and the trial store's
#customer number during the trial period.
percentageDiff_cust <- merge(scaledControlCust[, c("MONTHYEAR","control_cust")],
           measureOverTime[STORE_NBR==trial_store, c("nCustomers","MONTHYEAR")],
           by= "MONTHYEAR")[, percentageDiff_cust:= abs(control_cust-nCustomers)/control_cust]
```

```
scaledControlCust
```

```
##     STORE_NBR MONTHYEAR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##         <int>     <num>       <num>      <int>       <num>        <num>
##  1:       233    201807       260.8         46    1.021739     1.595745
##  2:       233    201808       255.5         43    1.023256     1.590909
##  3:       233    201809       218.3         39    1.076923     1.595238
##  4:       233    201810       159.3         32    1.000000     1.500000
##  5:       233    201811       201.3         38    1.026316     1.512821
##  6:       233    201812       265.4         43    1.046512     1.555556
##  7:       233    201901       150.5         31    1.000000     1.322581
##  8:       233    201902       220.7         42    1.023810     1.488372
##  9:       233    201903       178.0         34    1.029412     1.457143
## 10:       233    201904       144.2         27    1.037037     1.464286
## 11:       233    201905       312.1         54    1.037037     1.482143
## 12:       233    201906       197.0         34    1.000000     1.529412
##     avgPricePerUnit    store_type totSales TransactionMonth control_cust
##               <num>        <char>    <num>           <Date>        <num>
##  1:        3.477333 Control store    260.8       2018-07-01     47.35294
##  2:        3.650000 Control store    255.5       2018-08-01     44.26471
##  3:        3.258209 Control store    218.3       2018-09-01     40.14706
##  4:        3.318750 Control store    159.3       2018-10-01     32.94118
##  5:        3.411864 Control store    201.3       2018-11-01     39.11765
##  6:        3.791429 Control store    265.4       2018-12-01     44.26471
##  7:        3.670732 Control store    150.5       2019-01-01     31.91176
##  8:        3.448438 Control store    220.7       2019-02-01     43.23529
##  9:        3.490196 Control store    178.0       2019-03-01     35.00000
## 10:        3.517073 Control store    144.2       2019-04-01     27.79412
## 11:        3.760241 Control store    312.1       2019-05-01     55.58824
## 12:        3.788462 Control store    197.0       2019-06-01     35.00000
```

```
percentageDiff_cust
```

```
## Key: <MONTHYEAR>
##     MONTHYEAR control_cust nCustomers percentageDiff_cust
##         <num>        <num>      <int>               <num>
##  1:    201807     47.35294         47         0.007453416
##  2:    201808     44.26471         46         0.039202658
##  3:    201809     40.14706         39         0.028571429
##  4:    201810     32.94118         36         0.092857143
##  5:    201811     39.11765         39         0.003007519
##  6:    201812     44.26471         42         0.051162791
##  7:    201901     31.91176         31         0.028571429
##  8:    201902     43.23529         40         0.074829932
##  9:    201903     35.00000         46         0.314285714
## 10:    201904     27.79412         47         0.691005291
## 11:    201905     55.58824         53         0.046560847
## 12:    201906     35.00000         36         0.028571429
```

Let's see if the difference is significant! This is to test whether the observed differences in number of customers between the trial store and the control store during the trial period are statistically significant.

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_cust <- sd(percentageDiff_cust[MONTHYEAR < 201902 , percentageDiff_cust])
stdDev_cust
```

```
## [1] 0.03023928
```

```
# 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.

```
# Calculate the t-values for the trial months
percentageDiff_cust[, tValue := (percentageDiff_cust - 0)/stdDev_cust
][, TransactionMonth := as.Date(paste(MONTHYEAR %/% 100,
                                      MONTHYEAR %% 100, 1,
                                      sep = "-"), "%Y-%m-%d")
][MONTHYEAR < 201905 & MONTHYEAR > 201901, .(TransactionMonth,tValue)]
```

```
##    TransactionMonth    tValue
##              <Date>     <num>
## 1:       2019-02-01  2.474594
## 2:       2019-03-01 10.393294
## 3:       2019-04-01 22.851249
```

```
# Find the 95th percentile of the t distribution with the appropriate
# degrees of freedom to compare against
qt(0.95, df = degreesOfFreedom)
```

```
## [1] 1.894579
```

We can observe that the t-value is much higher than the 95th percentile value of the t-distribution for March and April - i.e. the increase in customer in the trial store in March and April is statistically higher than in the control store.

Let's create a more visual version of this by plotting the sales of the control store, the no of customer of the trial stores and the 95th percentile value of customer of the control store.

```
#Create new variables Store_type, totSales and TransactionMonth in the data table.
pastCust <- measureOverTime[, totCust := mean(nCustomers), by = c("MONTHYEAR", "store_type")
                    ][, TransactionMonth := as.Date(paste(MONTHYEAR %/% 100,
                                                          MONTHYEAR %% 100, 1,
                                                          sep = "-"), "%Y-%m-%d")
                    ][store_type %in% c("Trial store", "Control store"), ]
# Control store 95th percentile
pastCust_Controls95 <- pastCust[store_type == "Control store",
 ][, totCust := nCustomers * (1 + stdDev_cust * 2)
 ][, store_type := "Control 95th % confidence interval"]
# Control store 5th percentile
pastCust_Controls5 <- pastSales[store_type == "Control store",
```

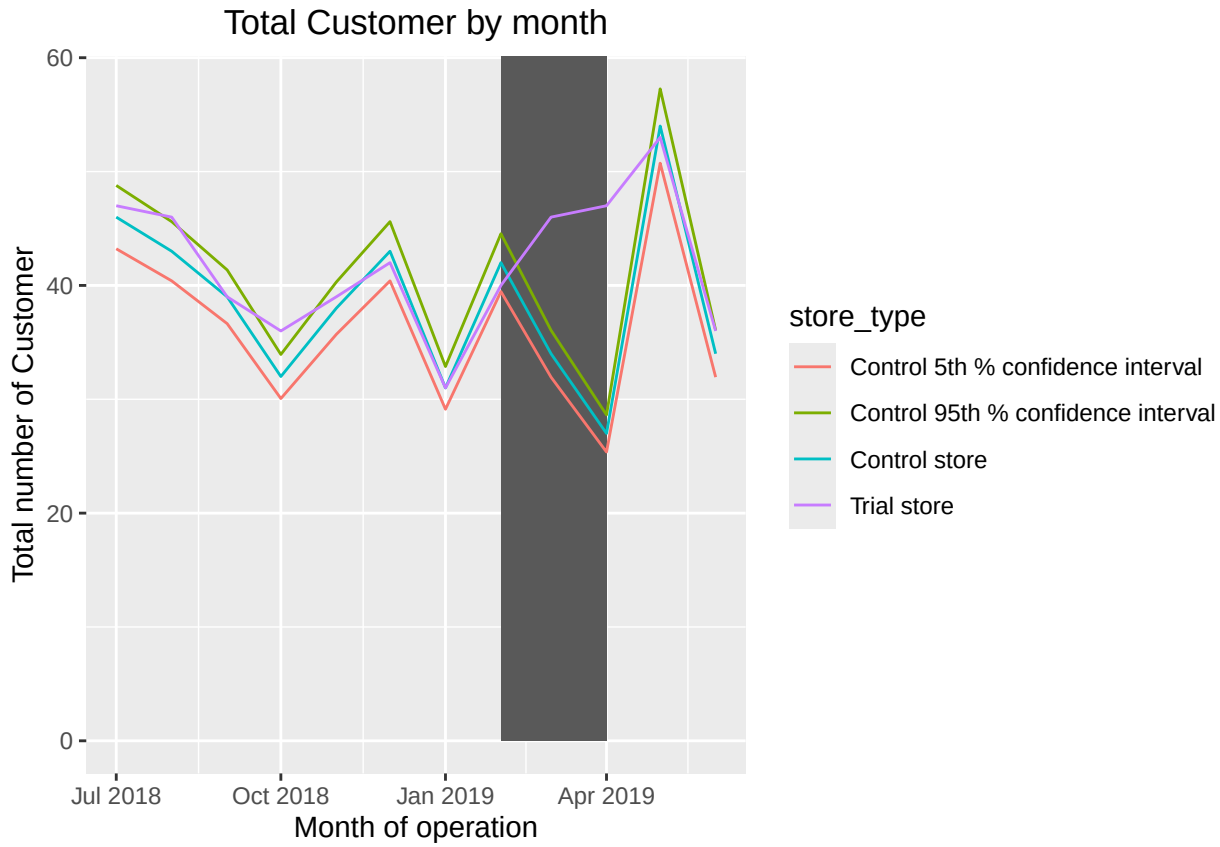
```
 ][, totCust := nCustomers * (1 - stdDev_cust * 2)
 ][, store_type := "Control 5th % confidence interval"]

trialAssessment_cust <- rbind(pastCust, pastCust_Controls95, pastCust_Controls5)

trialAssessment_cust
```

```
##     STORE_NBR MONTHYEAR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##         <int>     <num>       <num>      <int>       <num>        <num>
##  1:        77    201807       268.4         47    1.085106     1.509804
##  2:        77    201808       247.5         46    1.000000     1.543478
##  3:        77    201809       214.2         39    1.051282     1.585366
##  4:        77    201810       194.3         36    1.027778     1.351351
##  5:        77    201811       224.9         39    1.076923     1.500000
##  6:        77    201812       250.0         42    1.023810     1.511628
##  7:        77    201901       188.4         31    1.129032     1.657143
##  8:        77    201902       211.6         40    1.000000     1.675000
##  9:        77    201903       255.1         46    1.108696     1.490196
## 10:        77    201904       258.1         47    1.000000     1.617021
## 11:        77    201905       272.3         53    1.018868     1.444444
## 12:        77    201906       236.2         36    1.027778     1.648649
## 13:       233    201807       260.8         46    1.021739     1.595745
## 14:       233    201808       255.5         43    1.023256     1.590909
## 15:       233    201809       218.3         39    1.076923     1.595238
## 16:       233    201810       159.3         32    1.000000     1.500000
## 17:       233    201811       201.3         38    1.026316     1.512821
## 18:       233    201812       265.4         43    1.046512     1.555556
## 19:       233    201901       150.5         31    1.000000     1.322581
## 20:       233    201902       220.7         42    1.023810     1.488372
## 21:       233    201903       178.0         34    1.029412     1.457143
## 22:       233    201904       144.2         27    1.037037     1.464286
## 23:       233    201905       312.1         54    1.037037     1.482143
## 24:       233    201906       197.0         34    1.000000     1.529412
## 25:       233    201807       260.8         46    1.021739     1.595745
## 26:       233    201808       255.5         43    1.023256     1.590909
## 27:       233    201809       218.3         39    1.076923     1.595238
## 28:       233    201810       159.3         32    1.000000     1.500000
## 29:       233    201811       201.3         38    1.026316     1.512821
## 30:       233    201812       265.4         43    1.046512     1.555556
## 31:       233    201901       150.5         31    1.000000     1.322581
## 32:       233    201902       220.7         42    1.023810     1.488372
## 33:       233    201903       178.0         34    1.029412     1.457143
## 34:       233    201904       144.2         27    1.037037     1.464286
## 35:       233    201905       312.1         54    1.037037     1.482143
## 36:       233    201906       197.0         34    1.000000     1.529412
## 37:       233    201807       260.8         46    1.021739     1.595745
## 38:       233    201808       255.5         43    1.023256     1.590909
## 39:       233    201809       218.3         39    1.076923     1.595238
## 40:       233    201810       159.3         32    1.000000     1.500000
## 41:       233    201811       201.3         38    1.026316     1.512821
## 42:       233    201812       265.4         43    1.046512     1.555556
## 43:       233    201901       150.5         31    1.000000     1.322581
## 44:       233    201902       220.7         42    1.023810     1.488372
```

```
## 45:       233    201903       178.0         34    1.029412     1.457143
## 46:       233    201904       144.2         27    1.037037     1.464286
## 47:       233    201905       312.1         54    1.037037     1.482143
## 48:       233    201906       197.0         34    1.000000     1.529412
##     STORE_NBR MONTHYEAR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##     avgPricePerUnit                          store_type totSales
##               <num>                              <char>    <num>
##  1:       3.485714                         Trial store    268.4
##  2:       3.485915                         Trial store    247.5
##  3:       3.295385                         Trial store    214.2
##  4:       3.886000                         Trial store    194.3
##  5:       3.569841                         Trial store    224.9
##  6:       3.846154                         Trial store    250.0
##  7:       3.248276                         Trial store    188.4
##  8:       3.158209                         Trial store    211.6
##  9:       3.356579                         Trial store    255.1
## 10:       3.396053                         Trial store    258.1
## 11:       3.491026                         Trial store    272.3
## 12:       3.872131                         Trial store    236.2
## 13:       3.477333                       Control store    260.8
## 14:       3.650000                       Control store    255.5
## 15:       3.258209                       Control store    218.3
## 16:       3.318750                       Control store    159.3
## 17:       3.411864                       Control store    201.3
## 18:       3.791429                       Control store    265.4
## 19:       3.670732                       Control store    150.5
## 20:       3.448438                       Control store    220.7
## 21:       3.490196                       Control store    178.0
## 22:       3.517073                       Control store    144.2
## 23:       3.760241                       Control store    312.1
## 24:       3.788462                       Control store    197.0
## 25:       3.477333 Control 95th % confidence interval    260.8
## 26:       3.650000 Control 95th % confidence interval    255.5
## 27:       3.258209 Control 95th % confidence interval    218.3
## 28:       3.318750 Control 95th % confidence interval    159.3
## 29:       3.411864 Control 95th % confidence interval    201.3
## 30:       3.791429 Control 95th % confidence interval    265.4
## 31:       3.670732 Control 95th % confidence interval    150.5
## 32:       3.448438 Control 95th % confidence interval    220.7
## 33:       3.490196 Control 95th % confidence interval    178.0
## 34:       3.517073 Control 95th % confidence interval    144.2
## 35:       3.760241 Control 95th % confidence interval    312.1
## 36:       3.788462 Control 95th % confidence interval    197.0
## 37:       3.477333  Control 5th % confidence interval    260.8
## 38:       3.650000  Control 5th % confidence interval    255.5
## 39:       3.258209  Control 5th % confidence interval    218.3
## 40:       3.318750  Control 5th % confidence interval    159.3
## 41:       3.411864  Control 5th % confidence interval    201.3
## 42:       3.791429  Control 5th % confidence interval    265.4
## 43:       3.670732  Control 5th % confidence interval    150.5
## 44:       3.448438  Control 5th % confidence interval    220.7
## 45:       3.490196  Control 5th % confidence interval    178.0
## 46:       3.517073  Control 5th % confidence interval    144.2
## 47:       3.760241  Control 5th % confidence interval    312.1
```

```
## 48:        3.788462  Control 5th % confidence interval    197.0
##     avgPricePerUnit                           store_type totSales
##     TransactionMonth  totCust
##                <Date>    <num>
##  1:       2018-07-01 47.00000
##  2:       2018-08-01 46.00000
##  3:       2018-09-01 39.00000
##  4:       2018-10-01 36.00000
##  5:       2018-11-01 39.00000
##  6:       2018-12-01 42.00000
##  7:       2019-01-01 31.00000
##  8:       2019-02-01 40.00000
##  9:       2019-03-01 46.00000
## 10:       2019-04-01 47.00000
## 11:       2019-05-01 53.00000
## 12:       2019-06-01 36.00000
## 13:       2018-07-01 46.00000
## 14:       2018-08-01 43.00000
## 15:       2018-09-01 39.00000
## 16:       2018-10-01 32.00000
## 17:       2018-11-01 38.00000
## 18:       2018-12-01 43.00000
## 19:       2019-01-01 31.00000
## 20:       2019-02-01 42.00000
## 21:       2019-03-01 34.00000
## 22:       2019-04-01 27.00000
## 23:       2019-05-01 54.00000
## 24:       2019-06-01 34.00000
## 25:       2018-07-01 48.78201
## 26:       2018-08-01 45.60058
## 27:       2018-09-01 41.35866
## 28:       2018-10-01 33.93531
## 29:       2018-11-01 40.29819
## 30:       2018-12-01 45.60058
## 31:       2019-01-01 32.87484
## 32:       2019-02-01 44.54010
## 33:       2019-03-01 36.05627
## 34:       2019-04-01 28.63292
## 35:       2019-05-01 57.26584
## 36:       2019-06-01 36.05627
## 37:       2018-07-01 43.21799
## 38:       2018-08-01 40.39942
## 39:       2018-09-01 36.64134
## 40:       2018-10-01 30.06469
## 41:       2018-11-01 35.70181
## 42:       2018-12-01 40.39942
## 43:       2019-01-01 29.12516
## 44:       2019-02-01 39.45990
## 45:       2019-03-01 31.94373
## 46:       2019-04-01 25.36708
## 47:       2019-05-01 50.73416
## 48:       2019-06-01 31.94373
##     TransactionMonth  totCust
```

```
# Plotting these in one nice graph
ggplot(trialAssessment_cust, aes(TransactionMonth, totCust, color = store_type)) +
geom_rect(data = trialAssessment_cust[ MONTHYEAR < 201905 & MONTHYEAR > 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 number of Customer",
     title = "Total Customer by month")
```



Let's repeat finding the control store and assessing the impact of the trial for each of the other two trial stores.

Trial store 86

```
measureOverTime <- data[, .(TOTAL_SALES = sum(TOT_SALES),
         nCustomers =uniqueN(LYLTY_CARD_NBR),
         nTxnPerCust =uniqueN(TXN_ID)/uniqueN(LYLTY_CARD_NBR),
         nChipsPerTxn =sum(PROD_QTY)/uniqueN(TXN_ID),
         avgPricePerUnit = sum(TOT_SALES)/sum(PROD_QTY)),
         by = .(MONTHYEAR,STORE_NBR)][order(STORE_NBR,MONTHYEAR)]

measureOverTime
```

```
##       MONTHYEAR STORE_NBR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
```

```
##           <num>     <int>       <num>      <int>       <num>        <num>
##    1:    201807         1       188.9         47    1.042553     1.183673
##    2:    201808         1       168.4         41    1.000000     1.268293
##    3:    201809         1       268.1         57    1.035088     1.203390
##    4:    201810         1       175.4         39    1.025641     1.275000
##    5:    201811         1       184.8         44    1.022727     1.222222
##   ---
## 3161:    201902       272       385.3         44    1.068182     1.893617
## 3162:    201903       272       421.9         48    1.062500     1.901961
## 3163:    201904       272       445.1         54    1.018519     1.909091
## 3164:    201905       272       314.6         34    1.176471     1.775000
## 3165:    201906       272       301.9         33    1.090909     1.888889
##       avgPricePerUnit
##                 <num>
##    1:        3.256897
##    2:        3.238462
##    3:        3.776056
##    4:        3.439216
##    5:        3.360000
##   ---
## 3161:        4.329213
## 3162:        4.349485
## 3163:        4.239048
## 3164:        4.430986
## 3165:        4.439706
```

```
# Use the functions for calculating correlation
trial_store <- 86
corr_nSales <- calculate_corr(preTrialMeasures, quote(TOTAL_SALES),trial_store )
corr_nCustomers <- calculate_corr(preTrialMeasures, quote(nCustomers),trial_store)
# Use functions to calculate correlation
magnitude_nSales <- calculate_magnitude_distance(preTrialMeasures, quote(TOTAL_SALES),
trial_store)
magnitude_nCustomers <- calculate_magnitude_distance(preTrialMeasures,
quote(nCustomers), trial_store)

corr_nSales
```

```
##      trial_store comparison_store   corr_calc
##            <num>            <num>       <num>
##   1:          86                1  0.36800517
##   2:          86                2 -0.52950061
##   3:          86                3  0.13978875
##   4:          86                4  0.03561817
##   5:          86                5  0.47485248
##  ---
## 255:          86              268 -0.42996210
## 256:          86              269  0.73238121
## 257:          86              270 -0.73686576
## 258:          86              271  0.55489332
## 259:          86              272  0.34156742
```

```
corr_nCustomers
```

```
##      trial_store comparison_store    corr_calc
##            <num>            <num>        <num>
##   1:          86                1  0.417636359
##   2:          86                2 -0.055354489
##   3:          86                3  0.086547902
##   4:          86                4  0.002310019
##   5:          86                5  0.024497426
##  ---
## 255:          86              268 -0.060412646
## 256:          86              269  0.395986990
## 257:          86              270 -0.633430012
## 258:          86              271  0.248272495
## 259:          86              272 -0.447304451
```

```
magnitude_nSales
```

```
##      trial_store comparison_store mag_measure
##            <num>            <num>       <num>
##   1:          86                1   0.2161804
##   2:          86                2   0.1745721
##   3:          86                3   0.7459063
##   4:          86                4   0.5002491
##   5:          86                5   0.9122626
##  ---
## 255:          86              268   0.2411338
## 256:          86              269   0.9129628
## 257:          86              270   0.8400337
## 258:          86              271   0.9030423
## 259:          86              272   0.4343493
```

```
magnitude_nCustomers
```

```
##      trial_store comparison_store mag_measure
##            <num>            <num>       <num>
##   1:          86                1   0.4393829
##   2:          86                2   0.3627657
##   3:          86                3   0.9086162
##   4:          86                4   0.7672816
##   5:          86                5   0.9006030
##  ---
## 255:          86              268   0.4074107
## 256:          86              269   0.9251510
## 257:          86              270   0.8759369
## 258:          86              271   0.8998605
## 259:          86              272   0.4206454
```

```
#Create a combined score composed of correlation and magnitude, by
#first merging the correlations table with the magnitude table.
# A simple average on the scores would be 0.5 * corr_measure + 0.5 *mag_measure
```

```
corr_weight <- 0.5
score_nSales <- merge(
  corr_nSales,
  magnitude_nSales,
  by = c("trial_store", "comparison_store")
)[, scoreNSales := corr_weight * corr_calc + corr_weight * mag_measure]

score_nCustomers <- merge(
  corr_nCustomers,
  magnitude_nCustomers,
  by = c("trial_store", "comparison_store")
)[, scoreNCust := corr_weight* corr_calc + corr_weight*mag_measure]

score_nSales
```

```
## Key: <trial_store, comparison_store>
##      trial_store comparison_store   corr_calc mag_measure scoreNSales
##            <num>            <num>       <num>       <num>       <num>
##   1:          86                1  0.36800517   0.2161804  0.29209279
##   2:          86                2 -0.52950061   0.1745721 -0.17746424
##   3:          86                3  0.13978875   0.7459063  0.44284753
##   4:          86                4  0.03561817   0.5002491  0.26793362
##   5:          86                5  0.47485248   0.9122626  0.69355755
##  ---
## 255:          86              268 -0.42996210   0.2411338 -0.09441415
## 256:          86              269  0.73238121   0.9129628  0.82267198
## 257:          86              270 -0.73686576   0.8400337  0.05158399
## 258:          86              271  0.55489332   0.9030423  0.72896778
## 259:          86              272  0.34156742   0.4343493  0.38795837
```

```
score_nCustomers
```

```
## Key: <trial_store, comparison_store>
##      trial_store comparison_store    corr_calc mag_measure  scoreNCust
##            <num>            <num>        <num>       <num>       <num>
##   1:          86                1  0.417636359   0.4393829  0.42850961
##   2:          86                2 -0.055354489   0.3627657  0.15370558
##   3:          86                3  0.086547902   0.9086162  0.49758204
##   4:          86                4  0.002310019   0.7672816  0.38479582
##   5:          86                5  0.024497426   0.9006030  0.46255020
##  ---
## 255:          86              268 -0.060412646   0.4074107  0.17349903
## 256:          86              269  0.395986990   0.9251510  0.66056898
## 257:          86              270 -0.633430012   0.8759369  0.12125347
## 258:          86              271  0.248272495   0.8998605  0.57406651
## 259:          86              272 -0.447304451   0.4206454 -0.01332954
```

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.

```
#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("trial_store","comparison_store"))
score_Control[, finalControlScore := scoreNSales * 0.5 + scoreNCust * 0.5]

score_Control
```

```
## Key: <trial_store, comparison_store>
##      trial_store comparison_store corr_calc.x mag_measure.x scoreNSales
##            <num>            <num>       <num>         <num>       <num>
##   1:          86                1  0.36800517     0.2161804  0.29209279
##   2:          86                2 -0.52950061     0.1745721 -0.17746424
##   3:          86                3  0.13978875     0.7459063  0.44284753
##   4:          86                4  0.03561817     0.5002491  0.26793362
##   5:          86                5  0.47485248     0.9122626  0.69355755
##  ---
## 255:          86              268 -0.42996210     0.2411338 -0.09441415
## 256:          86              269  0.73238121     0.9129628  0.82267198
## 257:          86              270 -0.73686576     0.8400337  0.05158399
## 258:          86              271  0.55489332     0.9030423  0.72896778
## 259:          86              272  0.34156742     0.4343493  0.38795837
##       corr_calc.y mag_measure.y  scoreNCust finalControlScore
##             <num>         <num>       <num>             <num>
##   1:  0.417636359     0.4393829  0.42850961        0.36030120
##   2: -0.055354489     0.3627657  0.15370558       -0.01187933
##   3:  0.086547902     0.9086162  0.49758204        0.47021479
##   4:  0.002310019     0.7672816  0.38479582        0.32636472
##   5:  0.024497426     0.9006030  0.46255020        0.57805387
##  ---
## 255: -0.060412646     0.4074107  0.17349903        0.03954244
## 256:  0.395986990     0.9251510  0.66056898        0.74162048
## 257: -0.633430012     0.8759369  0.12125347        0.08641873
## 258:  0.248272495     0.8998605  0.57406651        0.65151715
## 259: -0.447304451     0.4206454 -0.01332954        0.18731441
```

The store with the highest score is then selected as the control store since it is most similar to the trial store.Select control stores based on the highest matching store (closest to 1 but not the store itself, i.e. the second ranked highest store)

```
#Select the most appropriate control store for trial store 86
#by finding the store with the highest final score.
control_store <-score_Control[order(-finalControlScore)][2,comparison_store]

control_store
```

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

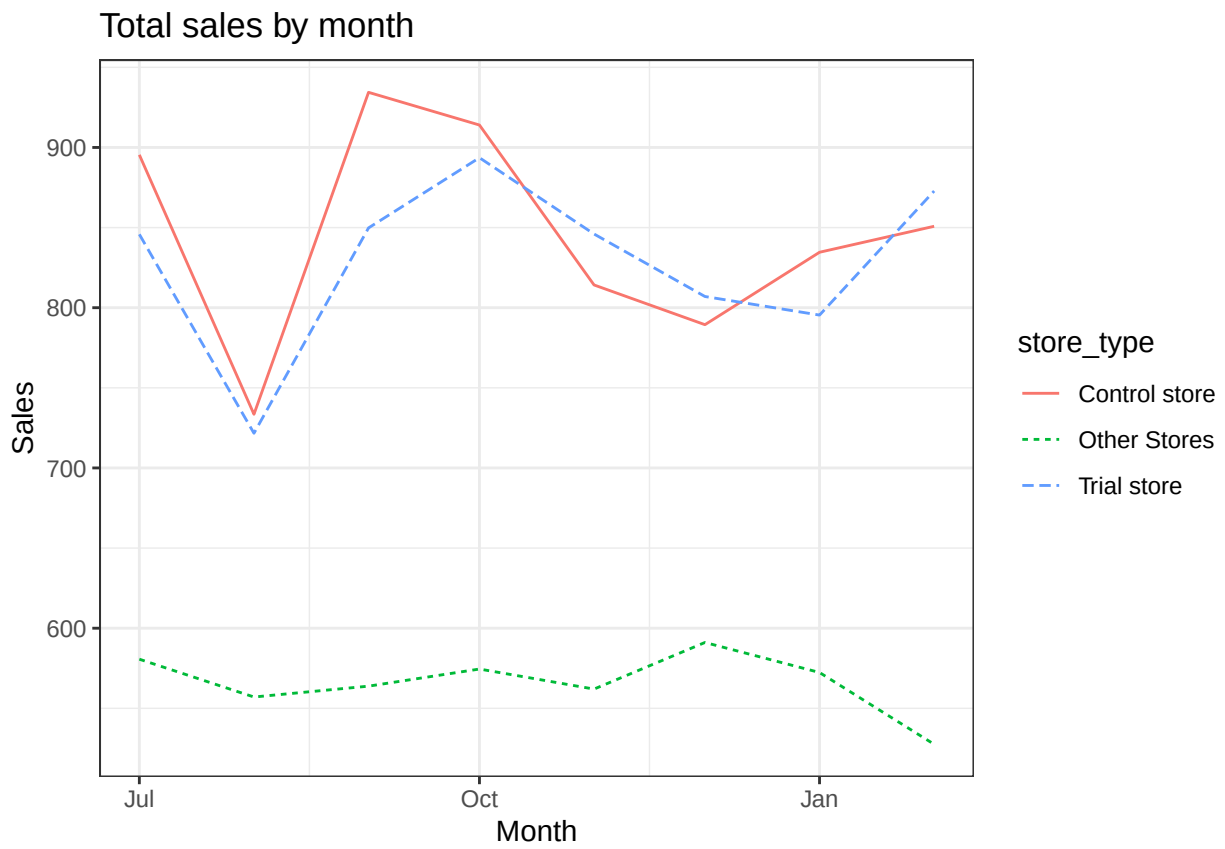
Looks like store 155 will be a control store for trial store 86. Again, let's check visually if the drivers are indeed similar in the period before the trial. We'll look at total sales first.

```
# Visual checks on monthly trends based on the total sales pre-trial
measurePreTrial_Sales <- measureOverTime[, store_type:= ifelse(STORE_NBR== trial_store,
                                                               "Trial store",
```

```
                                                        ifelse(STORE_NBR== control_store,
                                                               "Control store","Other Stores"))
                ][, .(sales= mean(TOTAL_SALES)),
                  by= .(store_type, MONTHYEAR)][,
                                                transaction_month:= as.Date(paste(
                  MONTHYEAR %/% 100, MONTHYEAR %% 100,1, sep ="-",
                  format= "%Y-%m-%d"))][ MONTHYEAR <201903,]

measurePreTrial_Sales
```

```
##        store_type MONTHYEAR    sales transaction_month
##            <char>     <num>    <num>            <Date>
##  1:  Other Stores    201807 580.6630        2018-07-01
##  2:  Other Stores    201808 557.0686        2018-08-01
##  3:  Other Stores    201809 563.8302        2018-09-01
##  4:  Other Stores    201810 574.5167        2018-10-01
##  5:  Other Stores    201811 562.0126        2018-11-01
##  6:  Other Stores    201812 591.0920        2018-12-01
##  7:  Other Stores    201901 572.3625        2019-01-01
##  8:  Other Stores    201902 527.5504        2019-02-01
##  9:   Trial store    201807 845.8000        2018-07-01
## 10:   Trial store    201808 721.6500        2018-08-01
## 11:   Trial store    201809 849.8000        2018-09-01
## 12:   Trial store    201810 893.6000        2018-10-01
## 13:   Trial store    201811 846.0000        2018-11-01
## 14:   Trial store    201812 807.0000        2018-12-01
## 15:   Trial store    201901 795.4000        2019-01-01
## 16:   Trial store    201902 872.8000        2019-02-01
## 17: Control store    201807 895.4000        2018-07-01
## 18: Control store    201808 733.5000        2018-08-01
## 19: Control store    201809 934.4000        2018-09-01
## 20: Control store    201810 914.0000        2018-10-01
## 21: Control store    201811 814.2000        2018-11-01
## 22: Control store    201812 789.4000        2018-12-01
## 23: Control store    201901 834.6000        2019-01-01
## 24: Control store    201902 850.8000        2019-02-01
##        store_type MONTHYEAR    sales transaction_month
```

```
ggplot(measurePreTrial_Sales, aes(x=transaction_month,y = sales, colour = store_type)) +
 geom_line(aes(linetype = store_type))+ theme_bw() +
  labs(x= "Month", y = "Sales", title = "Total sales by month")
```



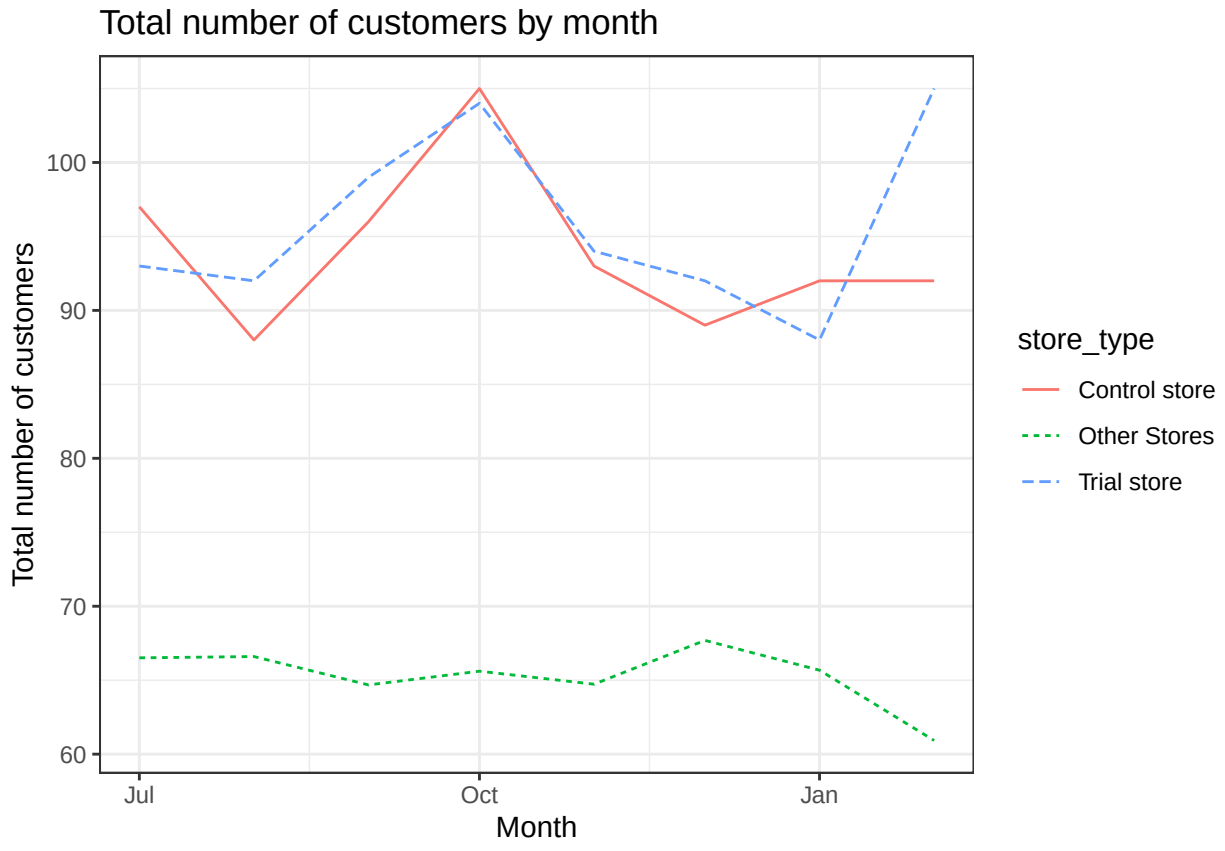
Great, sales are trending in a similar way. Next, number of customers.

```
# Visual checks on monthly trends based on the total customers pre-trial
# Visual checks on monthly trends based on the total sales pre-trial
measurePreTrial_Customer <- measureOverTime[, store_type:= ifelse(STORE_NBR== trial_store,
                                                                   "Trial store",
                                                                   ifelse(STORE_NBR== control_store,
                                                                          "Control store", "Other Stores"))
                ][, .(customer= mean(nCustomers)),
                  by= .(store_type, MONTHYEAR)][, transaction_month:= as.Date(paste(
                  MONTHYEAR %/% 100, MONTHYEAR %% 100,1, sep ="-",
                  format= "%Y-%m-%d"))][ MONTHYEAR <201903,]

measurePreTrial_Customer
```

```
##        store_type MONTHYEAR  customer transaction_month
##            <char>     <num>     <num>            <Date>
##  1:  Other Stores    201807  66.51145        2018-07-01
##  2:  Other Stores    201808  66.59770        2018-08-01
##  3:  Other Stores    201809  64.67939        2018-09-01
##  4:  Other Stores    201810  65.60837        2018-10-01
##  5:  Other Stores    201811  64.72901        2018-11-01
##  6:  Other Stores    201812  67.68966        2018-12-01
##  7:  Other Stores    201901  65.67433        2019-01-01
##  8:  Other Stores    201902  60.92366        2019-02-01
##  9:   Trial store    201807  93.00000        2018-07-01
```

```
## 10:   Trial store    201808  92.00000        2018-08-01
## 11:   Trial store    201809  99.00000        2018-09-01
## 12:   Trial store    201810 104.00000        2018-10-01
## 13:   Trial store    201811  94.00000        2018-11-01
## 14:   Trial store    201812  92.00000        2018-12-01
## 15:   Trial store    201901  88.00000        2019-01-01
## 16:   Trial store    201902 105.00000        2019-02-01
## 17: Control store    201807  97.00000        2018-07-01
## 18: Control store    201808  88.00000        2018-08-01
## 19: Control store    201809  96.00000        2018-09-01
## 20: Control store    201810 105.00000        2018-10-01
## 21: Control store    201811  93.00000        2018-11-01
## 22: Control store    201812  89.00000        2018-12-01
## 23: Control store    201901  92.00000        2019-01-01
## 24: Control store    201902  92.00000        2019-02-01
##        store_type MONTHYEAR  customer transaction_month
```

```
ggplot(measurePreTrial_Customer, aes(x=transaction_month,y = customer, colour = store_type)) +
  geom_line(aes(linetype = store_type))+ theme_bw() +
  labs(x= "Month", y = "Total number of customers", title = "Total number of customers by month")
```



Good, the trend in number of customers is also similar. Lets now assess the impact of the trial on sales.

We'll start with scaling the control store's sales to a level similar to control for any differences between the two stores outside of the trial period.

```
scalingFactor <- preTrialMeasures[
  STORE_NBR == trial_store & MONTHYEAR < 201902, sum(TOTAL_SALES)
]/ preTrialMeasures[
  STORE_NBR == control_store & MONTHYEAR < 201902, sum(TOTAL_SALES)
]
scalingFactor
```

```
## [1] 0.9735863
```

```
# Apply the scaling factor
scaledControlSales<- measureOverTime[STORE_NBR ==
                                       control_store, ][,control_sales:= TOTAL_SALES *scalingFactor]
#Now that we have comparable sales figures for the control store, we can calculate
#the percentage difference between the scaled control sales and the trial store's
#sales during the trial period.
percentageDiff <- merge(scaledControlSales[, c(1,9)],
          measureOverTime[STORE_NBR==trial_store, c(1,3)],
          by= "MONTHYEAR")[,percentageDiff:= abs(control_sales-TOTAL_SALES)/control_sales]

scaledControlSales
```

```
##     MONTHYEAR STORE_NBR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##         <num>     <int>       <num>      <int>       <num>        <num>
##  1:    201807       155      895.40         97    1.216495     2.016949
##  2:    201808       155      733.50         88    1.272727     1.910714
##  3:    201809       155      934.40         96    1.343750     2.015504
##  4:    201810       155      914.00        105    1.219048     2.000000
##  5:    201811       155      814.20         93    1.268817     2.033898
##  6:    201812       155      789.40         89    1.235955     2.018182
##  7:    201901       155      834.60         92    1.271739     2.017094
##  8:    201902       155      850.80         92    1.271739     2.034188
##  9:    201903       155      767.00         91    1.208791     2.036364
## 10:    201904       155      795.20         93    1.204301     2.017857
## 11:    201905       155      858.05        101    1.247525     1.920635
## 12:    201906       155      755.60         86    1.220930     2.019048
##     avgPricePerUnit    store_type control_sales
##               <num>        <char>         <num>
##  1:        3.762185 Control store      871.7492
##  2:        3.427570 Control store      714.1256
##  3:        3.593846 Control store      909.7191
##  4:        3.570313 Control store      889.8579
##  5:        3.392500 Control store      792.6940
##  6:        3.555856 Control store      768.5491
##  7:        3.536441 Control store      812.5552
##  8:        3.574790 Control store      828.3273
##  9:        3.424107 Control store      746.7407
## 10:        3.518584 Control store      774.1959
## 11:        3.545661 Control store      835.3858
## 12:        3.564151 Control store      735.6418
```

```
percentageDiff
```

```
## Key: <MONTHYEAR>
##     MONTHYEAR control_sales TOTAL_SALES percentageDiff
##         <num>         <num>       <num>          <num>
##  1:    201807      871.7492      845.80    0.029766829
##  2:    201808      714.1256      721.65    0.010536549
##  3:    201809      909.7191      849.80    0.065865473
##  4:    201810      889.8579      893.60    0.004205261
##  5:    201811      792.6940      846.00    0.067246631
##  6:    201812      768.5491      807.00    0.050030564
##  7:    201901      812.5552      795.40    0.021112610
##  8:    201902      828.3273      872.80    0.053689820
##  9:    201903      746.7407      945.40    0.266035145
## 10:    201904      774.1959      798.80    0.031780255
## 11:    201905      835.3858      826.90    0.010157894
## 12:    201906      735.6418      760.80    0.034198926
```

Let's see if the difference is significant! This is to test whether the observed differences in sales between the trial store and the control store during the trial period are statistically significant.

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[MONTHYEAR < 201902 , percentageDiff])
stdDev
```

```
## [1] 0.02576638
```

```
# 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.

```
# Calculate the t-values for the trial months
percentageDiff[, tValue := (percentageDiff - 0)/stdDev
][, TransactionMonth := as.Date(paste(MONTHYEAR %/% 100,
                                      MONTHYEAR %% 100, 1,
                                      sep = "-"), "%Y-%m-%d")
][MONTHYEAR < 201905 & MONTHYEAR > 201901, .(TransactionMonth,tValue)]
```

```
##    TransactionMonth    tValue
##              <Date>     <num>
## 1:       2019-02-01  2.083716
## 2:       2019-03-01 10.324893
## 3:       2019-04-01  1.233400
```

```
# Find the 95th percentile of the t distribution with the appropriate
# degrees of freedom to compare against
qt(0.95, df = degreesOfFreedom)
```

```
## [1] 1.894579
```

We can observe that the t-value is much higher than the 95th percentile value of the t-distribution for March - i.e. the increase in customer in the trial store in March is statistically higher than in the control store.

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.

```
#Create new variables Store_type, totSales and TransactionMonth in the data table.
pastSales <- measureOverTime[, totSales := mean(TOTAL_SALES),
                             by = c("MONTHYEAR", "store_type")
                    ][, TransactionMonth := as.Date(paste(MONTHYEAR %/% 100,
                                                          MONTHYEAR %% 100, 1,
                                                          sep = "-"), "%Y-%m-%d")
                    ][store_type %in% c("Trial store", "Control store"), ]
# Control store 95th percentile
pastSales_Controls95 <- pastSales[store_type == "Control store",
 ][, totSales := TOTAL_SALES * (1 + stdDev * 2)
 ][, store_type := "Control 95th % confidence interval"]
# Control store 5th percentile
pastSales_Controls5 <- pastSales[store_type == "Control store",
 ][, totSales := TOTAL_SALES * (1 - stdDev * 2)
 ][, store_type := "Control 5th % confidence interval"]

trialAssessment <- rbind(pastSales, pastSales_Controls95, pastSales_Controls5)

trialAssessment
```

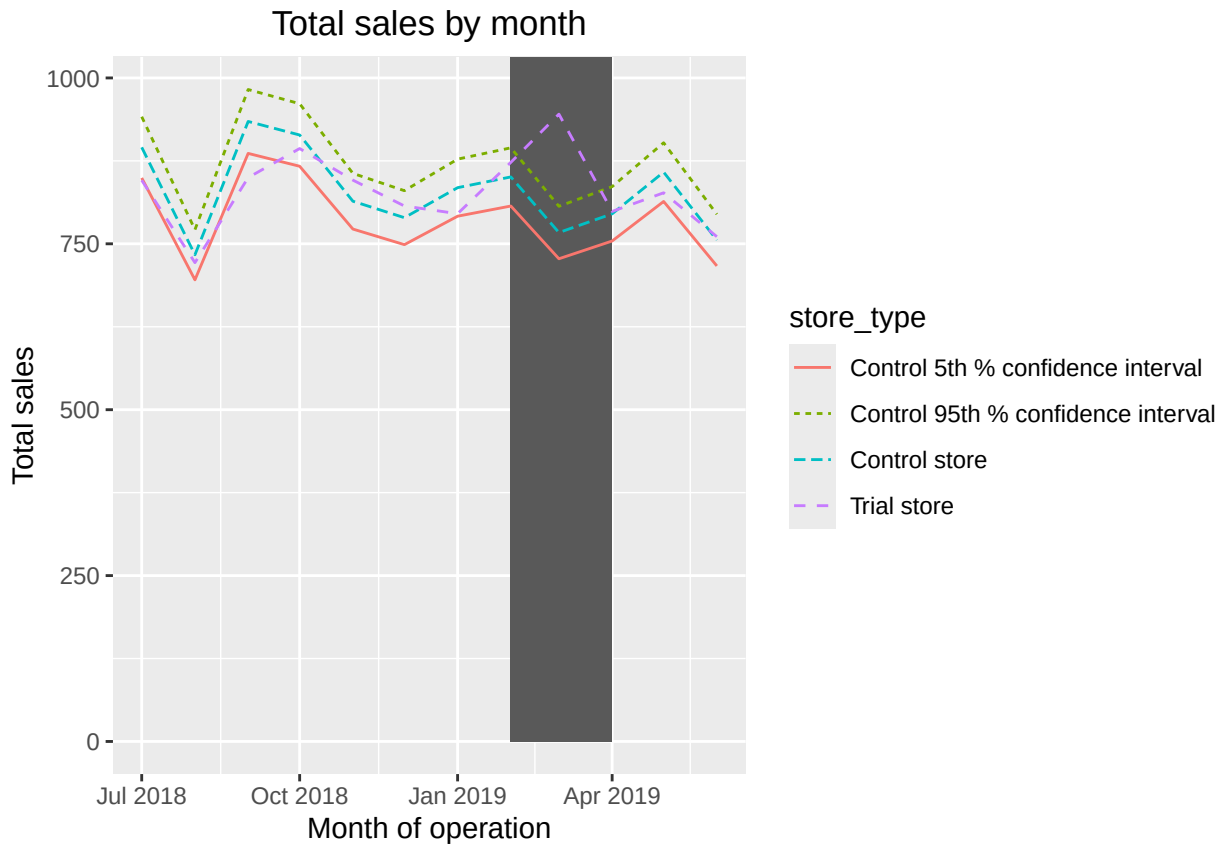
```
##     MONTHYEAR STORE_NBR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##         <num>     <int>       <num>      <int>       <num>        <num>
##  1:    201807        86      845.80         93    1.279570     1.991597
##  2:    201808        86      721.65         92    1.119565     1.951456
##  3:    201809        86      849.80         99    1.202020     2.016807
##  4:    201810        86      893.60        104    1.240385     2.000000
##  5:    201811        86      846.00         94    1.244681     2.017094
##  6:    201812        86      807.00         92    1.239130     2.000000
##  7:    201901        86      795.40         88    1.352273     2.016807
##  8:    201902        86      872.80        105    1.238095     2.007692
##  9:    201903        86      945.40        108    1.175926     2.015748
## 10:    201904        86      798.80         98    1.204082     2.016949
## 11:    201905        86      826.90         99    1.181818     2.017094
## 12:    201906        86      760.80         91    1.186813     2.018519
## 13:    201807       155      895.40         97    1.216495     2.016949
## 14:    201808       155      733.50         88    1.272727     1.910714
## 15:    201809       155      934.40         96    1.343750     2.015504
## 16:    201810       155      914.00        105    1.219048     2.000000
## 17:    201811       155      814.20         93    1.268817     2.033898
## 18:    201812       155      789.40         89    1.235955     2.018182
## 19:    201901       155      834.60         92    1.271739     2.017094
## 20:    201902       155      850.80         92    1.271739     2.034188
## 21:    201903       155      767.00         91    1.208791     2.036364
## 22:    201904       155      795.20         93    1.204301     2.017857
## 23:    201905       155      858.05        101    1.247525     1.920635
## 24:    201906       155      755.60         86    1.220930     2.019048
## 25:    201807       155      895.40         97    1.216495     2.016949
## 26:    201808       155      733.50         88    1.272727     1.910714
```

```
## 27:    201809       155      934.40         96    1.343750     2.015504
## 28:    201810       155      914.00        105    1.219048     2.000000
## 29:    201811       155      814.20         93    1.268817     2.033898
## 30:    201812       155      789.40         89    1.235955     2.018182
## 31:    201901       155      834.60         92    1.271739     2.017094
## 32:    201902       155      850.80         92    1.271739     2.034188
## 33:    201903       155      767.00         91    1.208791     2.036364
## 34:    201904       155      795.20         93    1.204301     2.017857
## 35:    201905       155      858.05        101    1.247525     1.920635
## 36:    201906       155      755.60         86    1.220930     2.019048
## 37:    201807       155      895.40         97    1.216495     2.016949
## 38:    201808       155      733.50         88    1.272727     1.910714
## 39:    201809       155      934.40         96    1.343750     2.015504
## 40:    201810       155      914.00        105    1.219048     2.000000
## 41:    201811       155      814.20         93    1.268817     2.033898
## 42:    201812       155      789.40         89    1.235955     2.018182
## 43:    201901       155      834.60         92    1.271739     2.017094
## 44:    201902       155      850.80         92    1.271739     2.034188
## 45:    201903       155      767.00         91    1.208791     2.036364
## 46:    201904       155      795.20         93    1.204301     2.017857
## 47:    201905       155      858.05        101    1.247525     1.920635
## 48:    201906       155      755.60         86    1.220930     2.019048
##     MONTHYEAR STORE_NBR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##     avgPricePerUnit                              store_type totSales
##                <num>                                  <char>    <num>
##  1:       3.568776                             Trial store 845.8000
##  2:       3.590299                             Trial store 721.6500
##  3:       3.540833                             Trial store 849.8000
##  4:       3.463566                             Trial store 893.6000
##  5:       3.584746                             Trial store 846.0000
##  6:       3.539474                             Trial store 807.0000
##  7:       3.314167                             Trial store 795.4000
##  8:       3.344061                             Trial store 872.8000
##  9:       3.692969                             Trial store 945.4000
## 10:       3.356303                             Trial store 798.8000
## 11:       3.503814                             Trial store 826.9000
## 12:       3.489908                             Trial store 760.8000
## 13:       3.762185                           Control store 895.4000
## 14:       3.427570                           Control store 733.5000
## 15:       3.593846                           Control store 934.4000
## 16:       3.570313                           Control store 914.0000
## 17:       3.392500                           Control store 814.2000
## 18:       3.555856                           Control store 789.4000
## 19:       3.536441                           Control store 834.6000
## 20:       3.574790                           Control store 850.8000
## 21:       3.424107                           Control store 767.0000
## 22:       3.518584                           Control store 795.2000
## 23:       3.545661                           Control store 858.0500
## 24:       3.564151                           Control store 755.6000
## 25:       3.762185 Control 95th % confidence interval 941.5424
## 26:       3.427570 Control 95th % confidence interval 771.2993
## 27:       3.593846 Control 95th % confidence interval 982.5522
## 28:       3.570313 Control 95th % confidence interval 961.1009
## 29:       3.392500 Control 95th % confidence interval 856.1580
```

```
## 30:        3.555856 Control 95th % confidence interval 830.0800
## 31:        3.536441 Control 95th % confidence interval 877.6092
## 32:        3.574790 Control 95th % confidence interval 894.6441
## 33:        3.424107 Control 95th % confidence interval 806.5256
## 34:        3.518584 Control 95th % confidence interval 836.1789
## 35:        3.545661 Control 95th % confidence interval 902.2677
## 36:        3.564151 Control 95th % confidence interval 794.5382
## 37:        3.762185  Control 5th % confidence interval 849.2576
## 38:        3.427570  Control 5th % confidence interval 695.7007
## 39:        3.593846  Control 5th % confidence interval 886.2478
## 40:        3.570313  Control 5th % confidence interval 866.8991
## 41:        3.392500  Control 5th % confidence interval 772.2420
## 42:        3.555856  Control 5th % confidence interval 748.7200
## 43:        3.536441  Control 5th % confidence interval 791.5908
## 44:        3.574790  Control 5th % confidence interval 806.9559
## 45:        3.424107  Control 5th % confidence interval 727.4744
## 46:        3.518584  Control 5th % confidence interval 754.2211
## 47:        3.545661  Control 5th % confidence interval 813.8323
## 48:        3.564151  Control 5th % confidence interval 716.6618
##     avgPricePerUnit                          store_type totSales
##     TransactionMonth
##                <Date>
##  1:      2018-07-01
##  2:      2018-08-01
##  3:      2018-09-01
##  4:      2018-10-01
##  5:      2018-11-01
##  6:      2018-12-01
##  7:      2019-01-01
##  8:      2019-02-01
##  9:      2019-03-01
## 10:      2019-04-01
## 11:      2019-05-01
## 12:      2019-06-01
## 13:      2018-07-01
## 14:      2018-08-01
## 15:      2018-09-01
## 16:      2018-10-01
## 17:      2018-11-01
## 18:      2018-12-01
## 19:      2019-01-01
## 20:      2019-02-01
## 21:      2019-03-01
## 22:      2019-04-01
## 23:      2019-05-01
## 24:      2019-06-01
## 25:      2018-07-01
## 26:      2018-08-01
## 27:      2018-09-01
## 28:      2018-10-01
## 29:      2018-11-01
## 30:      2018-12-01
## 31:      2019-01-01
## 32:      2019-02-01
```

```
## 33:       2019-03-01
## 34:       2019-04-01
## 35:       2019-05-01
## 36:       2019-06-01
## 37:       2018-07-01
## 38:       2018-08-01
## 39:       2018-09-01
## 40:       2018-10-01
## 41:       2018-11-01
## 42:       2018-12-01
## 43:       2019-01-01
## 44:       2019-02-01
## 45:       2019-03-01
## 46:       2019-04-01
## 47:       2019-05-01
## 48:       2019-06-01
##     TransactionMonth
```

```r
# Plotting these in one nice graph
ggplot(trialAssessment, aes(TransactionMonth, totSales, color = store_type)) +
geom_rect(data = trialAssessment[ MONTHYEAR < 201905 & MONTHYEAR > 201901 ,],
aes(xmin = min(TransactionMonth), xmax = max(TransactionMonth), ymin = 0 ,
ymax = Inf, color = NULL), show.legend = FALSE) + geom_line(aes(linetype = store_type)) +
labs(x = "Month of operation", y = "Total sales", title = "Total sales by month")
```



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.

```
scalingFactor_cust <- preTrialMeasures[
  STORE_NBR == trial_store & MONTHYEAR < 201902, sum(nCustomers)
]/ preTrialMeasures[
  STORE_NBR == control_store & MONTHYEAR < 201902, sum(nCustomers)
]
scalingFactor_cust
```

```
## [1] 1.00303
```

```
# Apply the scaling factor
scaledControlCust<- measureOverTime[STORE_NBR ==
                                      control_store, ][,control_cust:= nCustomers *scalingFactor_cust]
#Now that we have comparable sales figures for the control store, we can calculate
#the percentage difference between the scaled control customer number and the trial store's
#customer number during the trial period.
percentageDiff_cust <- merge(scaledControlCust[, c(1,11)],
           measureOverTime[STORE_NBR==trial_store, c(1,4)],
           by= "MONTHYEAR")[, percentageDiff_cust:= abs(control_cust-nCustomers)/control_cust]

scaledControlCust
```

```
##     MONTHYEAR STORE_NBR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##         <num>     <int>       <num>      <int>       <num>        <num>
##  1:    201807       155      895.40         97    1.216495     2.016949
##  2:    201808       155      733.50         88    1.272727     1.910714
##  3:    201809       155      934.40         96    1.343750     2.015504
##  4:    201810       155      914.00        105    1.219048     2.000000
##  5:    201811       155      814.20         93    1.268817     2.033898
##  6:    201812       155      789.40         89    1.235955     2.018182
##  7:    201901       155      834.60         92    1.271739     2.017094
##  8:    201902       155      850.80         92    1.271739     2.034188
##  9:    201903       155      767.00         91    1.208791     2.036364
## 10:    201904       155      795.20         93    1.204301     2.017857
## 11:    201905       155      858.05        101    1.247525     1.920635
## 12:    201906       155      755.60         86    1.220930     2.019048
##     avgPricePerUnit    store_type totSales TransactionMonth control_cust
##               <num>        <char>    <num>           <Date>        <num>
##  1:        3.762185 Control store   895.40       2018-07-01     97.29394
##  2:        3.427570 Control store   733.50       2018-08-01     88.26667
##  3:        3.593846 Control store   934.40       2018-09-01     96.29091
##  4:        3.570313 Control store   914.00       2018-10-01    105.31818
##  5:        3.392500 Control store   814.20       2018-11-01     93.28182
##  6:        3.555856 Control store   789.40       2018-12-01     89.26970
##  7:        3.536441 Control store   834.60       2019-01-01     92.27879
##  8:        3.574790 Control store   850.80       2019-02-01     92.27879
##  9:        3.424107 Control store   767.00       2019-03-01     91.27576
## 10:        3.518584 Control store   795.20       2019-04-01     93.28182
## 11:        3.545661 Control store   858.05       2019-05-01    101.30606
## 12:        3.564151 Control store   755.60       2019-06-01     86.26061
```

```
percentageDiff_cust
```

```
## Key: <MONTHYEAR>
##     MONTHYEAR control_cust nCustomers percentageDiff_cust
##         <num>        <num>      <int>               <num>
##  1:    201807     97.29394         93         0.044133678
##  2:    201808     88.26667         92         0.042296073
##  3:    201809     96.29091         99         0.028134441
##  4:    201810    105.31818        104         0.012516185
##  5:    201811     93.28182         94         0.007699055
##  6:    201812     89.26970         92         0.030584881
##  7:    201901     92.27879         88         0.046368055
##  8:    201902     92.27879        105         0.137856298
##  9:    201903     91.27576        108         0.183227648
## 10:    201904     93.28182         98         0.050579866
## 11:    201905    101.30606         99         0.022763304
## 12:    201906     86.26061         91         0.054942739
```

Let's see if the difference is significant! This is to test whether the observed differences in number of customers between the trial store and the control store during the trial period are statistically significant.

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_cust <- sd(percentageDiff_cust[MONTHYEAR < 201902 , percentageDiff_cust])
stdDev_cust
```

```
## [1] 0.01541253
```

```
# 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.

```
# Calculate the t-values for the trial months
percentageDiff_cust[, tValue := (percentageDiff_cust - 0)/stdDev_cust
][, TransactionMonth := as.Date(paste(MONTHYEAR %/% 100,
                                      MONTHYEAR %% 100, 1,
                                      sep = "-"), "%Y-%m-%d")
][MONTHYEAR < 201905 & MONTHYEAR > 201901, .(TransactionMonth,tValue)]
```

```
##    TransactionMonth    tValue
##              <Date>     <num>
## 1:       2019-02-01  8.944428
## 2:       2019-03-01 11.888224
## 3:       2019-04-01  3.281736
```

```
# Find the 95th percentile of the t distribution with the appropriate
# degrees of freedom to compare against
qt(0.95, df = degreesOfFreedom)
```

```
## [1] 1.894579
```

Let's create a more visual version of this by plotting the sales of the control store, the no of customer of the trial stores and the 95th percentile value of customer of the control store.

```
#Create new variables Store_type, totSales and TransactionMonth in the data table.
pastCust <- measureOverTime[, totCust := mean(nCustomers),
                            by = c("MONTHYEAR", "store_type")
                     ][, TransactionMonth := as.Date(paste(MONTHYEAR %/% 100,
                                                           MONTHYEAR %% 100, 1,
                                                           sep = "-"), "%Y-%m-%d")
                     ][store_type %in% c("Trial store", "Control store"), ]
# Control store 95th percentile
pastCust_Controls95 <- pastCust[store_type == "Control store",
 ][, totCust := nCustomers * (1 + stdDev_cust * 2)
 ][, store_type := "Control 95th % confidence interval"]
# Control store 5th percentile
pastCust_Controls5 <- pastSales[store_type == "Control store",
 ][, totCust := nCustomers * (1 - stdDev_cust * 2)
 ][, store_type := "Control 5th % confidence interval"]

trialAssessment_cust <- rbind(pastCust, pastCust_Controls95, pastCust_Controls5)

trialAssessment_cust
```

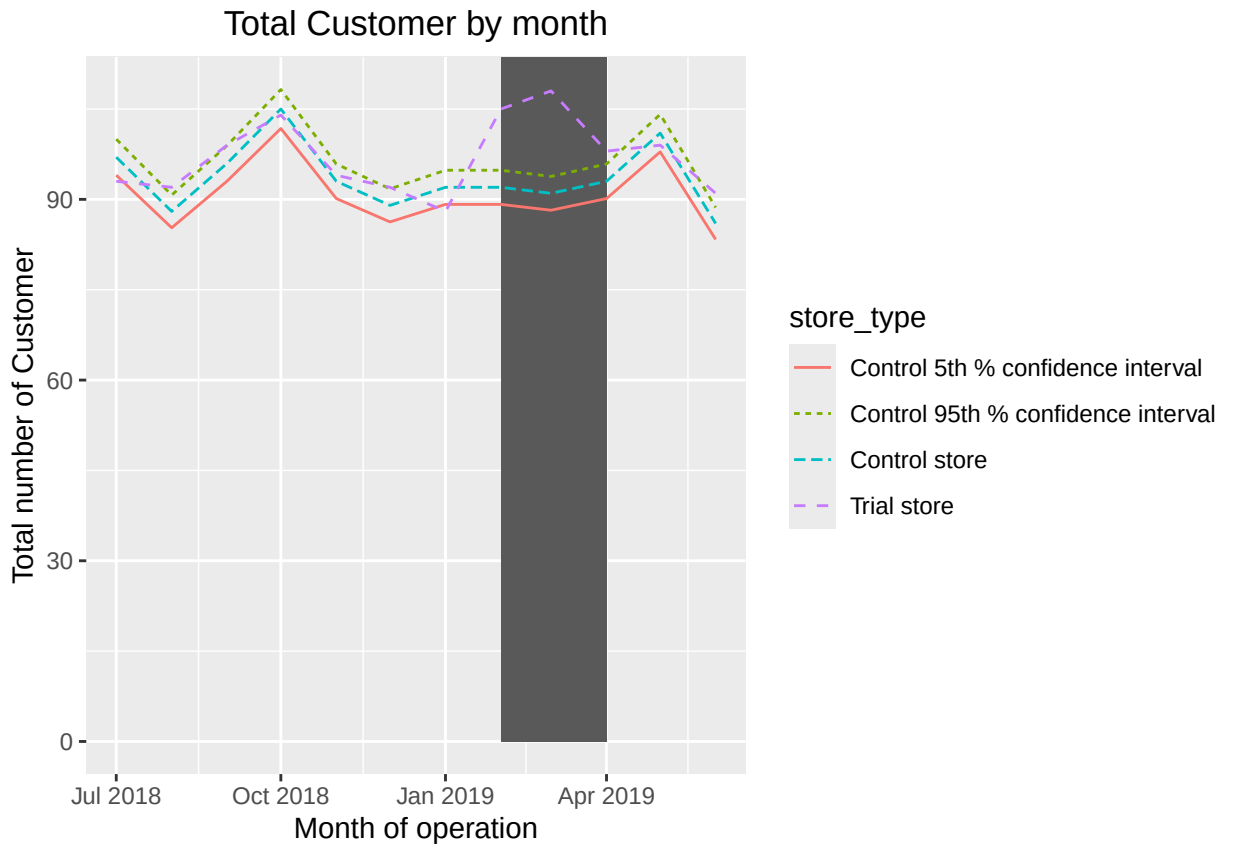
```
##     MONTHYEAR STORE_NBR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##         <num>     <int>       <num>      <int>       <num>        <num>
##  1:    201807        86      845.80         93    1.279570     1.991597
##  2:    201808        86      721.65         92    1.119565     1.951456
##  3:    201809        86      849.80         99    1.202020     2.016807
##  4:    201810        86      893.60        104    1.240385     2.000000
##  5:    201811        86      846.00         94    1.244681     2.017094
##  6:    201812        86      807.00         92    1.239130     2.000000
##  7:    201901        86      795.40         88    1.352273     2.016807
##  8:    201902        86      872.80        105    1.238095     2.007692
##  9:    201903        86      945.40        108    1.175926     2.015748
## 10:    201904        86      798.80         98    1.204082     2.016949
## 11:    201905        86      826.90         99    1.181818     2.017094
## 12:    201906        86      760.80         91    1.186813     2.018519
## 13:    201807       155      895.40         97    1.216495     2.016949
## 14:    201808       155      733.50         88    1.272727     1.910714
## 15:    201809       155      934.40         96    1.343750     2.015504
## 16:    201810       155      914.00        105    1.219048     2.000000
## 17:    201811       155      814.20         93    1.268817     2.033898
## 18:    201812       155      789.40         89    1.235955     2.018182
## 19:    201901       155      834.60         92    1.271739     2.017094
## 20:    201902       155      850.80         92    1.271739     2.034188
## 21:    201903       155      767.00         91    1.208791     2.036364
## 22:    201904       155      795.20         93    1.204301     2.017857
## 23:    201905       155      858.05        101    1.247525     1.920635
## 24:    201906       155      755.60         86    1.220930     2.019048
## 25:    201807       155      895.40         97    1.216495     2.016949
## 26:    201808       155      733.50         88    1.272727     1.910714
```

```
## 27:    201809       155      934.40         96    1.343750     2.015504
## 28:    201810       155      914.00        105    1.219048     2.000000
## 29:    201811       155      814.20         93    1.268817     2.033898
## 30:    201812       155      789.40         89    1.235955     2.018182
## 31:    201901       155      834.60         92    1.271739     2.017094
## 32:    201902       155      850.80         92    1.271739     2.034188
## 33:    201903       155      767.00         91    1.208791     2.036364
## 34:    201904       155      795.20         93    1.204301     2.017857
## 35:    201905       155      858.05        101    1.247525     1.920635
## 36:    201906       155      755.60         86    1.220930     2.019048
## 37:    201807       155      895.40         97    1.216495     2.016949
## 38:    201808       155      733.50         88    1.272727     1.910714
## 39:    201809       155      934.40         96    1.343750     2.015504
## 40:    201810       155      914.00        105    1.219048     2.000000
## 41:    201811       155      814.20         93    1.268817     2.033898
## 42:    201812       155      789.40         89    1.235955     2.018182
## 43:    201901       155      834.60         92    1.271739     2.017094
## 44:    201902       155      850.80         92    1.271739     2.034188
## 45:    201903       155      767.00         91    1.208791     2.036364
## 46:    201904       155      795.20         93    1.204301     2.017857
## 47:    201905       155      858.05        101    1.247525     1.920635
## 48:    201906       155      755.60         86    1.220930     2.019048
##     MONTHYEAR STORE_NBR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##     avgPricePerUnit                              store_type totSales
##                <num>                                  <char>    <num>
##  1:       3.568776                             Trial store   845.80
##  2:       3.590299                             Trial store   721.65
##  3:       3.540833                             Trial store   849.80
##  4:       3.463566                             Trial store   893.60
##  5:       3.584746                             Trial store   846.00
##  6:       3.539474                             Trial store   807.00
##  7:       3.314167                             Trial store   795.40
##  8:       3.344061                             Trial store   872.80
##  9:       3.692969                             Trial store   945.40
## 10:       3.356303                             Trial store   798.80
## 11:       3.503814                             Trial store   826.90
## 12:       3.489908                             Trial store   760.80
## 13:       3.762185                           Control store   895.40
## 14:       3.427570                           Control store   733.50
## 15:       3.593846                           Control store   934.40
## 16:       3.570313                           Control store   914.00
## 17:       3.392500                           Control store   814.20
## 18:       3.555856                           Control store   789.40
## 19:       3.536441                           Control store   834.60
## 20:       3.574790                           Control store   850.80
## 21:       3.424107                           Control store   767.00
## 22:       3.518584                           Control store   795.20
## 23:       3.545661                           Control store   858.05
## 24:       3.564151                           Control store   755.60
## 25:       3.762185 Control 95th % confidence interval   895.40
## 26:       3.427570 Control 95th % confidence interval   733.50
## 27:       3.593846 Control 95th % confidence interval   934.40
## 28:       3.570313 Control 95th % confidence interval   914.00
## 29:       3.392500 Control 95th % confidence interval   814.20
```

```
## 30:        3.555856 Control 95th % confidence interval   789.40
## 31:        3.536441 Control 95th % confidence interval   834.60
## 32:        3.574790 Control 95th % confidence interval   850.80
## 33:        3.424107 Control 95th % confidence interval   767.00
## 34:        3.518584 Control 95th % confidence interval   795.20
## 35:        3.545661 Control 95th % confidence interval   858.05
## 36:        3.564151 Control 95th % confidence interval   755.60
## 37:        3.762185  Control 5th % confidence interval   895.40
## 38:        3.427570  Control 5th % confidence interval   733.50
## 39:        3.593846  Control 5th % confidence interval   934.40
## 40:        3.570313  Control 5th % confidence interval   914.00
## 41:        3.392500  Control 5th % confidence interval   814.20
## 42:        3.555856  Control 5th % confidence interval   789.40
## 43:        3.536441  Control 5th % confidence interval   834.60
## 44:        3.574790  Control 5th % confidence interval   850.80
## 45:        3.424107  Control 5th % confidence interval   767.00
## 46:        3.518584  Control 5th % confidence interval   795.20
## 47:        3.545661  Control 5th % confidence interval   858.05
## 48:        3.564151  Control 5th % confidence interval   755.60
##     avgPricePerUnit                          store_type totSales
##     TransactionMonth   totCust
##                <Date>     <num>
##  1:       2018-07-01  93.00000
##  2:       2018-08-01  92.00000
##  3:       2018-09-01  99.00000
##  4:       2018-10-01 104.00000
##  5:       2018-11-01  94.00000
##  6:       2018-12-01  92.00000
##  7:       2019-01-01  88.00000
##  8:       2019-02-01 105.00000
##  9:       2019-03-01 108.00000
## 10:       2019-04-01  98.00000
## 11:       2019-05-01  99.00000
## 12:       2019-06-01  91.00000
## 13:       2018-07-01  97.00000
## 14:       2018-08-01  88.00000
## 15:       2018-09-01  96.00000
## 16:       2018-10-01 105.00000
## 17:       2018-11-01  93.00000
## 18:       2018-12-01  89.00000
## 19:       2019-01-01  92.00000
## 20:       2019-02-01  92.00000
## 21:       2019-03-01  91.00000
## 22:       2019-04-01  93.00000
## 23:       2019-05-01 101.00000
## 24:       2019-06-01  86.00000
## 25:       2018-07-01  99.99003
## 26:       2018-08-01  90.71261
## 27:       2018-09-01  98.95921
## 28:       2018-10-01 108.23663
## 29:       2018-11-01  95.86673
## 30:       2018-12-01  91.74343
## 31:       2019-01-01  94.83591
## 32:       2019-02-01  94.83591
```

```
## 33:       2019-03-01  93.80508
## 34:       2019-04-01  95.86673
## 35:       2019-05-01 104.11333
## 36:       2019-06-01  88.65096
## 37:       2018-07-01  94.00997
## 38:       2018-08-01  85.28739
## 39:       2018-09-01  93.04079
## 40:       2018-10-01 101.76337
## 41:       2018-11-01  90.13327
## 42:       2018-12-01  86.25657
## 43:       2019-01-01  89.16409
## 44:       2019-02-01  89.16409
## 45:       2019-03-01  88.19492
## 46:       2019-04-01  90.13327
## 47:       2019-05-01  97.88667
## 48:       2019-06-01  83.34904
##     TransactionMonth   totCust
```

```
# Plotting these in one nice graph
ggplot(trialAssessment_cust, aes(TransactionMonth, totCust, color = store_type)) +
geom_rect(data = trialAssessment_cust[ MONTHYEAR < 201905 & MONTHYEAR > 201901 ,],
aes(xmin = min(TransactionMonth), xmax = max(TransactionMonth), ymin = 0 ,
ymax = Inf, color = NULL), show.legend = FALSE) + geom_line(aes(linetype = store_type)) +
labs(x = "Month of operation", y = "Total number of Customer", title = "Total Customer by month")
```



It looks like the number of customers is significantly higher in 2 out 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.

Trial store 88

```
measureOverTime <- data[, .(TOTAL_SALES = sum(TOT_SALES),
         nCustomers =uniqueN(LYLTY_CARD_NBR),
         nTxnPerCust =uniqueN(TXN_ID)/uniqueN(LYLTY_CARD_NBR),
         nChipsPerTxn =sum(PROD_QTY)/uniqueN(TXN_ID),
         avgPricePerUnit = sum(TOT_SALES)/sum(PROD_QTY)),
         by = .(MONTHYEAR,STORE_NBR)][order(STORE_NBR,MONTHYEAR)]

measureOverTime
```

```
##       MONTHYEAR STORE_NBR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##           <num>     <int>       <num>      <int>       <num>        <num>
##    1:    201807         1       188.9         47    1.042553     1.183673
##    2:    201808         1       168.4         41    1.000000     1.268293
##    3:    201809         1       268.1         57    1.035088     1.203390
##    4:    201810         1       175.4         39    1.025641     1.275000
##    5:    201811         1       184.8         44    1.022727     1.222222
##   ---
## 3161:    201902       272       385.3         44    1.068182     1.893617
## 3162:    201903       272       421.9         48    1.062500     1.901961
## 3163:    201904       272       445.1         54    1.018519     1.909091
## 3164:    201905       272       314.6         34    1.176471     1.775000
## 3165:    201906       272       301.9         33    1.090909     1.888889
##       avgPricePerUnit
##                 <num>
##    1:        3.256897
##    2:        3.238462
##    3:        3.776056
##    4:        3.439216
##    5:        3.360000
##   ---
## 3161:        4.329213
## 3162:        4.349485
## 3163:        4.239048
## 3164:        4.430986
## 3165:        4.439706
```

```
# Use the functions for calculating correlation
trial_store <- 88
corr_nSales <- calculate_corr(preTrialMeasures, quote(TOTAL_SALES),trial_store )
corr_nCustomers <- calculate_corr(preTrialMeasures, quote(nCustomers),trial_store)
# Use functions to calculate correlation
magnitude_nSales <- calculate_magnitude_distance(preTrialMeasures, quote(TOTAL_SALES),
trial_store)
magnitude_nCustomers <- calculate_magnitude_distance(preTrialMeasures,
quote(nCustomers), trial_store)
```

```
corr_nSales
```

```
##      trial_store comparison_store  corr_calc
##            <num>            <num>      <num>
##   1:          88                1  0.8425111
##   2:          88                2 -0.2112472
##   3:          88                3 -0.4673303
##   4:          88                4 -0.5061296
##   5:          88                5  0.2786384
##  ---
## 255:          88              268 -0.2596568
## 256:          88              269 -0.1085843
## 257:          88              270 -0.6857001
## 258:          88              271 -0.1930628
## 259:          88              272 -0.6457516
```

```
corr_nCustomers
```

```
##      trial_store comparison_store   corr_calc
##            <num>            <num>       <num>
##   1:          88                1  0.48744608
##   2:          88                2 -0.58924154
##   3:          88                3  0.43408024
##   4:          88                4 -0.21677788
##   5:          88                5 -0.04505805
##  ---
## 255:          88              268  0.50443078
## 256:          88              269  0.06247298
## 257:          88              270 -0.01901026
## 258:          88              271 -0.14587332
## 259:          88              272 -0.13301096
```

```
magnitude_nSales
```

```
##      trial_store comparison_store mag_measure
##            <num>            <num>       <num>
##   1:          88                1   0.1398611
##   2:          88                2   0.1127836
##   3:          88                3   0.8145085
##   4:          88                4   0.9052605
##   5:          88                5   0.5935283
##  ---
## 255:          88              268   0.1555198
## 256:          88              269   0.7027886
## 257:          88              270   0.7000577
## 258:          88              271   0.5937072
## 259:          88              272   0.2830149
```

```
magnitude_nCustomers
```

```
##      trial_store comparison_store mag_measure
```

```
##            <num>             <num>       <num>
##   1:          88                 1   0.3415380
##   2:          88                 2   0.2817444
##   3:          88                 3   0.8445393
##   4:          88                 4   0.9315932
##   5:          88                 5   0.6990286
##  ---
## 255:          88               268   0.3151694
## 256:          88               269   0.8211373
## 257:          88               270   0.8021920
## 258:          88               271   0.7014746
## 259:          88               272   0.3256301
```

```
#Create a combined score composed of correlation and magnitude, by
#first merging the correlations table with the magnitude table.
# A simple average on the scores would be 0.5 * corr_measure + 0.5 *mag_measure
corr_weight <- 0.5
score_nSales <- merge(
  corr_nSales,
  magnitude_nSales,
  by = c("trial_store", "comparison_store")
)[, scoreNSales := corr_weight * corr_calc + corr_weight * mag_measure]

score_nCustomers <- merge(
  corr_nCustomers,
  magnitude_nCustomers,
  by = c("trial_store", "comparison_store")
)[, scoreNCust := corr_weight* corr_calc + corr_weight*mag_measure]

score_nSales
```

```
## Key: <trial_store, comparison_store>
##      trial_store comparison_store  corr_calc mag_measure  scoreNSales
##            <num>            <num>      <num>       <num>        <num>
##   1:          88                1  0.8425111   0.1398611  0.491186143
##   2:          88                2 -0.2112472   0.1127836 -0.049231772
##   3:          88                3 -0.4673303   0.8145085  0.173589123
##   4:          88                4 -0.5061296   0.9052605  0.199565421
##   5:          88                5  0.2786384   0.5935283  0.436083346
##  ---
## 255:          88              268 -0.2596568   0.1555198 -0.052068508
## 256:          88              269 -0.1085843   0.7027886  0.297102131
## 257:          88              270 -0.6857001   0.7000577  0.007178844
## 258:          88              271 -0.1930628   0.5937072  0.200322198
## 259:          88              272 -0.6457516   0.2830149 -0.181368345
```

```
score_nCustomers
```

```
## Key: <trial_store, comparison_store>
##      trial_store comparison_store   corr_calc mag_measure  scoreNCust
##            <num>            <num>       <num>       <num>       <num>
##   1:          88                1  0.48744608   0.3415380  0.41449203
##   2:          88                2 -0.58924154   0.2817444 -0.15374856
```

```
##   3:          88               3  0.43408024   0.8445393  0.63930975
##   4:          88               4 -0.21677788   0.9315932  0.35740767
##   5:          88               5 -0.04505805   0.6990286  0.32698526
##  ---
## 255:          88             268  0.50443078   0.3151694  0.40980012
## 256:          88             269  0.06247298   0.8211373  0.44180514
## 257:          88             270 -0.01901026   0.8021920  0.39159088
## 258:          88             271 -0.14587332   0.7014746  0.27780065
## 259:          88             272 -0.13301096   0.3256301  0.09630955
```

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.

```
#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("trial_store","comparison_store"))
score_Control[, finalControlScore := scoreNSales * 0.5 + scoreNCust * 0.5]

score_Control
```

```
## Key: <trial_store, comparison_store>
##      trial_store comparison_store corr_calc.x mag_measure.x  scoreNSales
##            <num>            <num>       <num>         <num>        <num>
##   1:          88                1   0.8425111     0.1398611  0.491186143
##   2:          88                2  -0.2112472     0.1127836 -0.049231772
##   3:          88                3  -0.4673303     0.8145085  0.173589123
##   4:          88                4  -0.5061296     0.9052605  0.199565421
##   5:          88                5   0.2786384     0.5935283  0.436083346
##  ---
## 255:          88              268  -0.2596568     0.1555198 -0.052068508
## 256:          88              269  -0.1085843     0.7027886  0.297102131
## 257:          88              270  -0.6857001     0.7000577  0.007178844
## 258:          88              271  -0.1930628     0.5937072  0.200322198
## 259:          88              272  -0.6457516     0.2830149 -0.181368345
##      corr_calc.y mag_measure.y  scoreNCust finalControlScore
##            <num>         <num>       <num>             <num>
##   1:  0.48744608     0.3415380  0.41449203         0.4528391
##   2: -0.58924154     0.2817444 -0.15374856        -0.1014902
##   3:  0.43408024     0.8445393  0.63930975         0.4064494
##   4: -0.21677788     0.9315932  0.35740767         0.2784865
##   5: -0.04505805     0.6990286  0.32698526         0.3815343
##  ---
## 255:  0.50443078     0.3151694  0.40980012         0.1788658
## 256:  0.06247298     0.8211373  0.44180514         0.3694536
## 257: -0.01901026     0.8021920  0.39159088         0.1993849
## 258: -0.14587332     0.7014746  0.27780065         0.2390614
## 259: -0.13301096     0.3256301  0.09630955        -0.0425294
```

The store with the highest score is then selected as the control store since it is most similar to the trial store.Select control stores based on the highest matching store (closest to 1 but not the store itself, i.e. the second ranked highest store)

```
#Select the most appropriate control store for trial
#store 88 by finding the store with the highest final score.
control_store <-score_Control[order(-finalControlScore)][2,comparison_store]

control_store
```

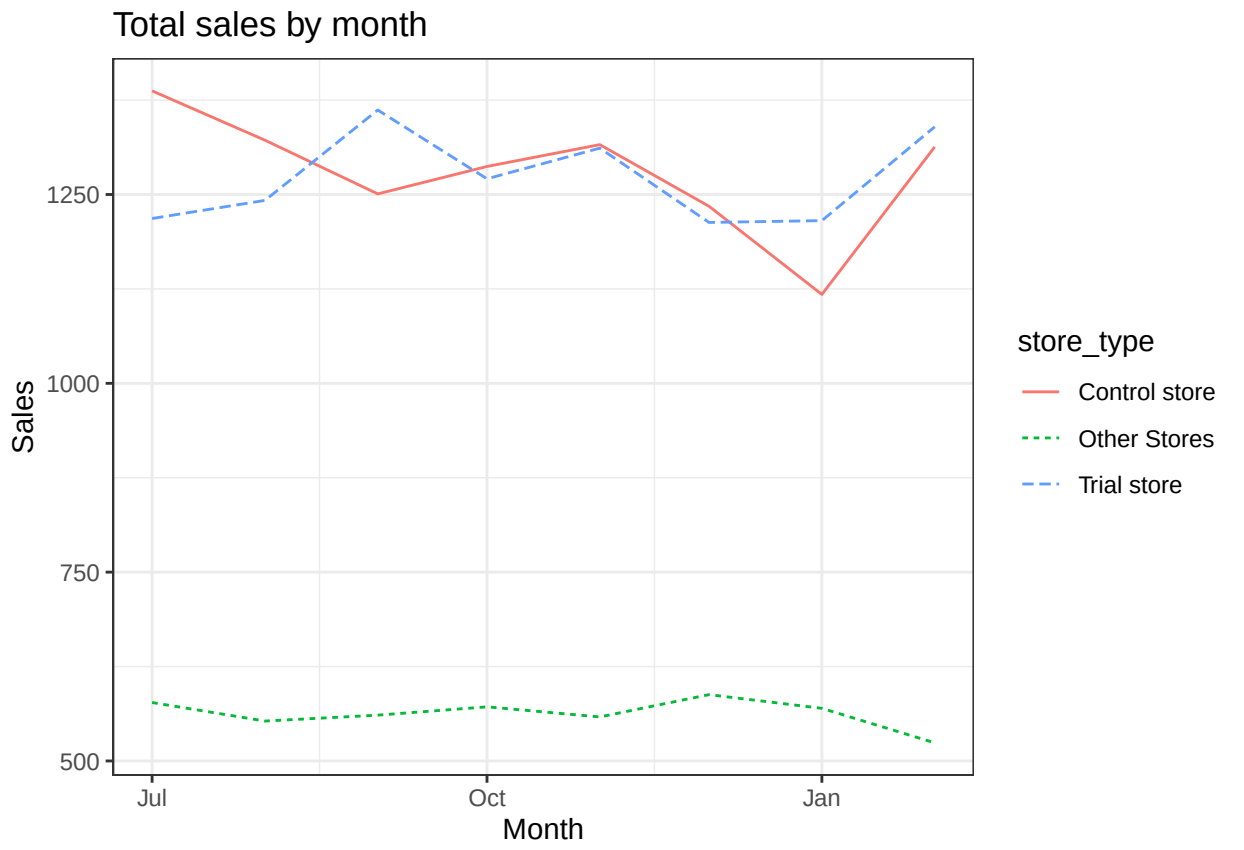
```
## [1] 237
```

Looks like store 237 will be a control store for trial store 88. Again, let's check visually if the drivers are indeed similar in the period before the trial. We'll look at total sales first.

```
# Visual checks on monthly trends based on the total sales pre-trial
measurePreTrial_Sales <- measureOverTime[,
                                         store_type:= ifelse(STORE_NBR== trial_store,
                                                             "Trial store",
                                                            ifelse(STORE_NBR== control_store,
                                                                   "Control store", "Other Stores"))
                ][, .(sales= mean(TOTAL_SALES)),
                  by= .(store_type, MONTHYEAR)][, transaction_month:= as.Date(paste(
                  MONTHYEAR %/% 100, MONTHYEAR %% 100,1, sep ="-",
                  format= "%Y-%m-%d"))][ MONTHYEAR <201903,]

measurePreTrial_Sales
```

```
##        store_type MONTHYEAR     sales transaction_month
##            <char>     <num>     <num>            <Date>
##  1:  Other Stores    201807  577.3645        2018-07-01
##  2:  Other Stores    201808  552.8197        2018-08-01
##  3:  Other Stores    201809  560.6683        2018-09-01
##  4:  Other Stores    201810  571.6639        2018-10-01
##  5:  Other Stores    201811  558.3210        2018-11-01
##  6:  Other Stores    201812  587.8314        2018-12-01
##  7:  Other Stores    201901  569.6686        2019-01-01
##  8:  Other Stores    201902  524.0046        2019-02-01
##  9:   Trial store    201807 1218.2000        2018-07-01
## 10:   Trial store    201808 1242.2000        2018-08-01
## 11:   Trial store    201809 1361.8000        2018-09-01
## 12:   Trial store    201810 1270.8000        2018-10-01
## 13:   Trial store    201811 1311.4000        2018-11-01
## 14:   Trial store    201812 1213.0000        2018-12-01
## 15:   Trial store    201901 1215.4000        2019-01-01
## 16:   Trial store    201902 1339.6000        2019-02-01
## 17: Control store    201807 1387.2000        2018-07-01
## 18: Control store    201808 1321.9000        2018-08-01
## 19: Control store    201809 1250.8000        2018-09-01
## 20: Control store    201810 1287.1000        2018-10-01
## 21: Control store    201811 1316.0000        2018-11-01
## 22: Control store    201812 1234.4000        2018-12-01
## 23: Control store    201901 1117.7000        2019-01-01
## 24: Control store    201902 1313.0000        2019-02-01
##        store_type MONTHYEAR     sales transaction_month
```

```
ggplot(measurePreTrial_Sales, aes(x=transaction_month,y = sales, colour = store_type)) +
 geom_line(aes(linetype = store_type))+ theme_bw() +
  labs(x= "Month", y = "Sales", title = "Total sales by month")
```



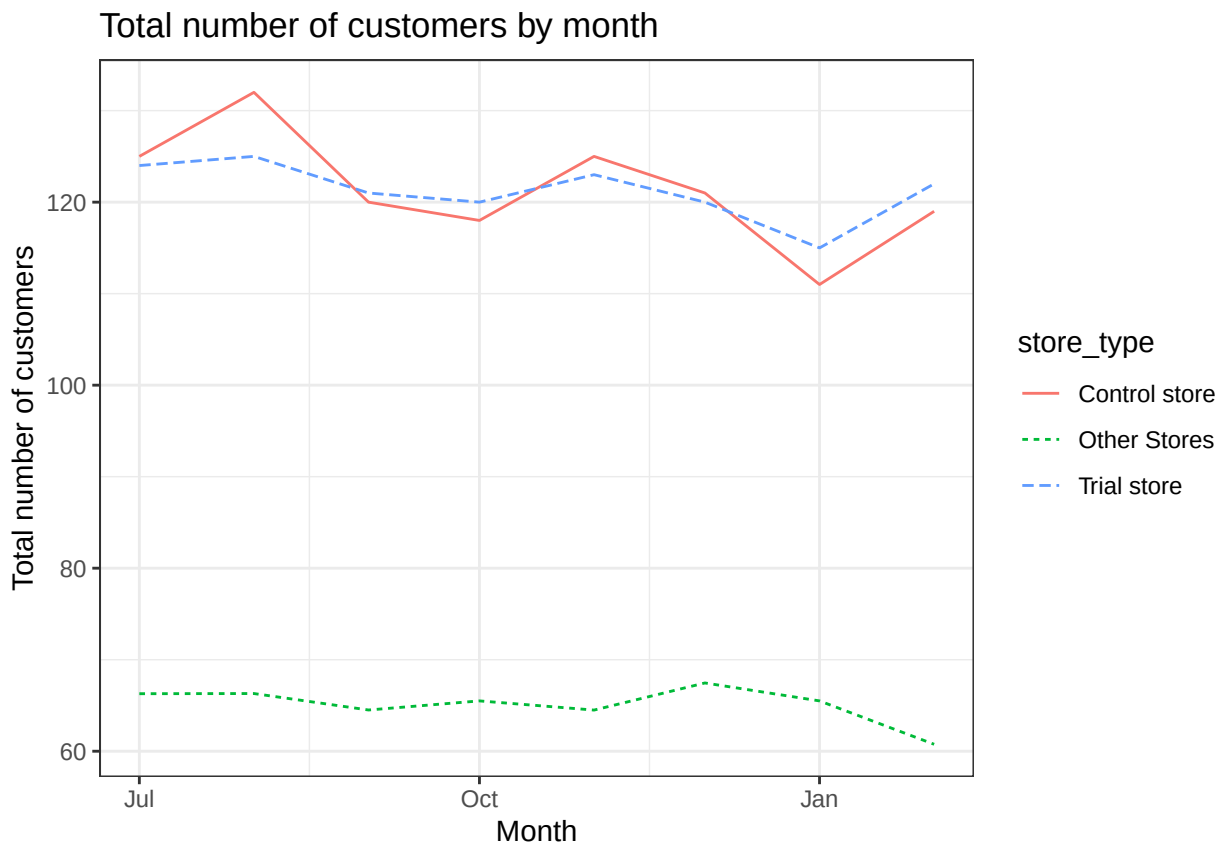
Great, sales are trending in a similar way. Next, number of customers.

```
## Visual checks on monthly trends based on the total customers pre-trial
# Visual checks on monthly trends based on the total sales pre-trial
measurePreTrial_Customer <- measureOverTime[,
                                            store_type:= ifelse(STORE_NBR== trial_store,
                                                                "Trial store",
                                                                ifelse(STORE_NBR== control_store,
                                                                       "Control store", "Other Stores"))
                ][, .(customer= mean(nCustomers)),
                  by= .(store_type, MONTHYEAR)][, transaction_month:= as.Date(paste(
                  MONTHYEAR %/% 100, MONTHYEAR %% 100,1, sep ="-",
                  format= "%Y-%m-%d"))][ MONTHYEAR <201903,]

measurePreTrial_Customer
```

```
##        store_type MONTHYEAR  customer transaction_month
##            <char>     <num>     <num>            <Date>
##  1:  Other Stores    201807  66.28626        2018-07-01
##  2:  Other Stores    201808  66.30268        2018-08-01
##  3:  Other Stores    201809  64.50382        2018-09-01
```

```
##  4:  Other Stores    201810  65.49810        2018-10-01
##  5:  Other Stores    201811  64.49618        2018-11-01
##  6:  Other Stores    201812  67.45977        2018-12-01
##  7:  Other Stores    201901  65.49808        2019-01-01
##  8:  Other Stores    201902  60.75573        2019-02-01
##  9:   Trial store    201807 124.00000        2018-07-01
## 10:   Trial store    201808 125.00000        2018-08-01
## 11:   Trial store    201809 121.00000        2018-09-01
## 12:   Trial store    201810 120.00000        2018-10-01
## 13:   Trial store    201811 123.00000        2018-11-01
## 14:   Trial store    201812 120.00000        2018-12-01
## 15:   Trial store    201901 115.00000        2019-01-01
## 16:   Trial store    201902 122.00000        2019-02-01
## 17: Control store    201807 125.00000        2018-07-01
## 18: Control store    201808 132.00000        2018-08-01
## 19: Control store    201809 120.00000        2018-09-01
## 20: Control store    201810 118.00000        2018-10-01
## 21: Control store    201811 125.00000        2018-11-01
## 22: Control store    201812 121.00000        2018-12-01
## 23: Control store    201901 111.00000        2019-01-01
## 24: Control store    201902 119.00000        2019-02-01
##        store_type MONTHYEAR  customer transaction_month
```

```
ggplot(measurePreTrial_Customer, aes(x=transaction_month,y = customer, colour = store_type)) +
  geom_line(aes(linetype = store_type))+ theme_bw() +
  labs(x= "Month", y = "Total number of customers", title = "Total number of customers by month")
```



Good, the trend in number of customers is also similar. Let's now assess the impact of the trial on sales.

We'll start with scaling the control store's sales to a level similar to control for any differences between the two stores outside of the trial period.

```
scalingFactor <- preTrialMeasures[
  STORE_NBR == trial_store & MONTHYEAR < 201902, sum(TOTAL_SALES)
]/ preTrialMeasures[
  STORE_NBR == control_store & MONTHYEAR < 201902, sum(TOTAL_SALES)
]
scalingFactor
```

```
## [1] 0.9907685
```

```
# Apply the scaling factor
scaledControlSales <- measureOverTime[STORE_NBR ==
                                        control_store, ][, control_sales :=
                                                           TOTAL_SALES *
                                                           scalingFactor]
#Now that we have comparable sales figures for the control store, we can calculate
#the percentage difference between the scaled control sales and the trial store's
#sales during the trial period.
percentageDiff <- merge(scaledControlSales[, c(1,9)],
          measureOverTime[STORE_NBR==trial_store, c(1,3)],
          by= "MONTHYEAR")[,percentageDiff:= abs(control_sales-TOTAL_SALES)/control_sales]

scaledControlSales
```

```
##     MONTHYEAR STORE_NBR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##         <num>     <int>       <num>      <int>       <num>        <num>
##  1:    201807       237      1387.2        125    1.248000     2.000000
##  2:    201808       237      1321.9        132    1.212121     1.900000
##  3:    201809       237      1250.8        120    1.183333     2.007042
##  4:    201810       237      1287.1        118    1.194915     2.035461
##  5:    201811       237      1316.0        125    1.224000     1.986928
##  6:    201812       237      1234.4        121    1.165289     2.007092
##  7:    201901       237      1117.7        111    1.162162     1.992248
##  8:    201902       237      1313.0        119    1.243697     2.000000
##  9:    201903       237      1177.6        116    1.129310     2.045802
## 10:    201904       237      1153.6        116    1.120690     2.015385
## 11:    201905       237      1127.9        122    1.155738     1.829787
## 12:    201906       237      1143.4        118    1.101695     2.000000
##     avgPricePerUnit    store_type control_sales
##               <num>        <char>         <num>
##  1:        4.446154 Control store      1374.394
##  2:        4.348355 Control store      1309.697
##  3:        4.388772 Control store      1239.253
##  4:        4.484669 Control store      1275.218
##  5:        4.328947 Control store      1303.851
##  6:        4.361837 Control store      1223.005
##  7:        4.349027 Control store      1107.382
##  8:        4.435811 Control store      1300.879
##  9:        4.394030 Control store      1166.729
## 10:        4.403053 Control store      1142.951
```

```
## 11:        4.371705 Control store      1117.488
## 12:        4.397692 Control store      1132.845
```

```
percentageDiff
```

```
## Key: <MONTHYEAR>
##     MONTHYEAR control_sales TOTAL_SALES percentageDiff
##         <num>         <num>       <num>          <num>
##  1:    201807      1374.394     1218.20    0.113645738
##  2:    201808      1309.697     1242.20    0.051536234
##  3:    201809      1239.253     1361.80    0.098887617
##  4:    201810      1275.218     1270.80    0.003464584
##  5:    201811      1303.851     1311.40    0.005789534
##  6:    201812      1223.005     1213.00    0.008180346
##  7:    201901      1107.382     1215.40    0.097543654
##  8:    201902      1300.879     1339.60    0.029765256
##  9:    201903      1166.729     1467.00    0.257361444
## 10:    201904      1142.951     1317.00    0.152280864
## 11:    201905      1117.488     1236.85    0.106813019
## 12:    201906      1132.845     1252.60    0.105712048
```

Let's see if the difference is significant! This is to test whether the observed differences in sales between the trial store and the control store during the trial period are statistically significant.

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[MONTHYEAR < 201902 , percentageDiff])
stdDev
```

```
## [1] 0.04907816
```

```
# 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.

```
# Calculate the t-values for the trial months
percentageDiff[, tValue := (percentageDiff - 0)/stdDev
][, TransactionMonth := as.Date(paste(MONTHYEAR %/% 100,
                                      MONTHYEAR %% 100, 1,
                                      sep = "-"), "%Y-%m-%d")
][MONTHYEAR < 201905 & MONTHYEAR > 201901, .(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 with the appropriate
# degrees of freedom to compare against
qt(0.95, df = degreesOfFreedom)
```

```
## [1] 1.894579
```

We can observe that the t-value is higher than the 95th percentile value of the t-distribution for March & April - i.e. the increase in sales in the trial store in March & April is statistically higher than in the control store.

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.

```
#Create new variables Store_type, totSales and TransactionMonth in the data table.
pastSales <- measureOverTime[, totSales := mean(TOTAL_SALES),
                             by = c("MONTHYEAR", "store_type")
                     ][, TransactionMonth := as.Date(paste(MONTHYEAR %/% 100,
                                                           MONTHYEAR %% 100, 1,
                                                           sep = "-"), "%Y-%m-%d")
                     ][store_type %in% c("Trial store", "Control store"), ]
# Control store 95th percentile
pastSales_Controls95 <- pastSales[store_type == "Control store",
 ][, totSales := TOTAL_SALES * (1 + stdDev * 2)
 ][, store_type := "Control 95th % confidence interval"]
# Control store 5th percentile
pastSales_Controls5 <- pastSales[store_type == "Control store",
 ][, totSales := TOTAL_SALES * (1 - stdDev * 2)
 ][, store_type := "Control 5th % confidence interval"]

trialAssessment <- rbind(pastSales, pastSales_Controls95, pastSales_Controls5)

trialAssessment
```

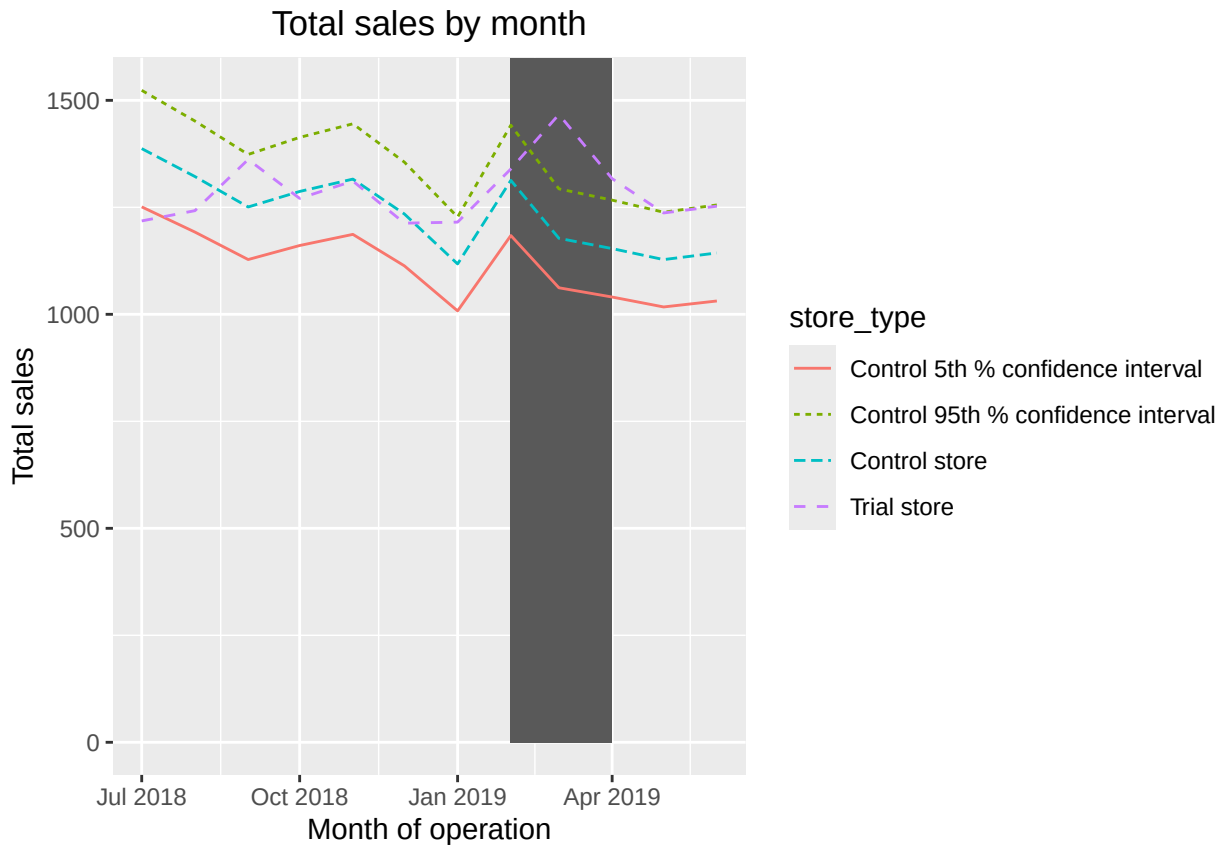
```
##     MONTHYEAR STORE_NBR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##         <num>     <int>       <num>      <int>       <num>        <num>
##  1:    201807        88     1218.20        124    1.161290     2.000000
##  2:    201808        88     1242.20        125    1.200000     1.913333
##  3:    201809        88     1361.80        121    1.247934     2.026490
##  4:    201810        88     1270.80        120    1.225000     2.040816
##  5:    201811        88     1311.40        123    1.211382     2.013423
##  6:    201812        88     1213.00        120    1.141667     2.014599
##  7:    201901        88     1215.40        115    1.217391     2.014286
##  8:    201902        88     1339.60        122    1.229508     2.013333
##  9:    201903        88     1467.00        133    1.263158     2.011905
## 10:    201904        88     1317.00        119    1.260504     2.000000
## 11:    201905        88     1236.85        123    1.195122     1.938776
## 12:    201906        88     1252.60        113    1.221239     2.028986
## 13:    201807       237     1387.20        125    1.248000     2.000000
## 14:    201808       237     1321.90        132    1.212121     1.900000
## 15:    201809       237     1250.80        120    1.183333     2.007042
## 16:    201810       237     1287.10        118    1.194915     2.035461
## 17:    201811       237     1316.00        125    1.224000     1.986928
## 18:    201812       237     1234.40        121    1.165289     2.007092
```

```
## 19:    201901       237     1117.70        111    1.162162     1.992248
## 20:    201902       237     1313.00        119    1.243697     2.000000
## 21:    201903       237     1177.60        116    1.129310     2.045802
## 22:    201904       237     1153.60        116    1.120690     2.015385
## 23:    201905       237     1127.90        122    1.155738     1.829787
## 24:    201906       237     1143.40        118    1.101695     2.000000
## 25:    201807       237     1387.20        125    1.248000     2.000000
## 26:    201808       237     1321.90        132    1.212121     1.900000
## 27:    201809       237     1250.80        120    1.183333     2.007042
## 28:    201810       237     1287.10        118    1.194915     2.035461
## 29:    201811       237     1316.00        125    1.224000     1.986928
## 30:    201812       237     1234.40        121    1.165289     2.007092
## 31:    201901       237     1117.70        111    1.162162     1.992248
## 32:    201902       237     1313.00        119    1.243697     2.000000
## 33:    201903       237     1177.60        116    1.129310     2.045802
## 34:    201904       237     1153.60        116    1.120690     2.015385
## 35:    201905       237     1127.90        122    1.155738     1.829787
## 36:    201906       237     1143.40        118    1.101695     2.000000
## 37:    201807       237     1387.20        125    1.248000     2.000000
## 38:    201808       237     1321.90        132    1.212121     1.900000
## 39:    201809       237     1250.80        120    1.183333     2.007042
## 40:    201810       237     1287.10        118    1.194915     2.035461
## 41:    201811       237     1316.00        125    1.224000     1.986928
## 42:    201812       237     1234.40        121    1.165289     2.007092
## 43:    201901       237     1117.70        111    1.162162     1.992248
## 44:    201902       237     1313.00        119    1.243697     2.000000
## 45:    201903       237     1177.60        116    1.129310     2.045802
## 46:    201904       237     1153.60        116    1.120690     2.015385
## 47:    201905       237     1127.90        122    1.155738     1.829787
## 48:    201906       237     1143.40        118    1.101695     2.000000
##     MONTHYEAR STORE_NBR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##     avgPricePerUnit                          store_type totSales
##                <num>                              <char>    <num>
##  1:        4.229861                         Trial store 1218.200
##  2:        4.328223                         Trial store 1242.200
##  3:        4.450327                         Trial store 1361.800
##  4:        4.236000                         Trial store 1270.800
##  5:        4.371333                         Trial store 1311.400
##  6:        4.394928                         Trial store 1213.000
##  7:        4.309929                         Trial store 1215.400
##  8:        4.435762                         Trial store 1339.600
##  9:        4.340237                         Trial store 1467.000
## 10:        4.390000                         Trial store 1317.000
## 11:        4.339825                         Trial store 1236.850
## 12:        4.473571                         Trial store 1252.600
## 13:        4.446154                       Control store 1387.200
## 14:        4.348355                       Control store 1321.900
## 15:        4.388772                       Control store 1250.800
## 16:        4.484669                       Control store 1287.100
## 17:        4.328947                       Control store 1316.000
## 18:        4.361837                       Control store 1234.400
## 19:        4.349027                       Control store 1117.700
## 20:        4.435811                       Control store 1313.000
## 21:        4.394030                       Control store 1177.600
```

```
## 22:        4.403053                      Control store 1153.600
## 23:        4.371705                      Control store 1127.900
## 24:        4.397692                      Control store 1143.400
## 25:        4.446154 Control 95th % confidence interval 1523.362
## 26:        4.348355 Control 95th % confidence interval 1451.653
## 27:        4.388772 Control 95th % confidence interval 1373.574
## 28:        4.484669 Control 95th % confidence interval 1413.437
## 29:        4.328947 Control 95th % confidence interval 1445.174
## 30:        4.361837 Control 95th % confidence interval 1355.564
## 31:        4.349027 Control 95th % confidence interval 1227.409
## 32:        4.435811 Control 95th % confidence interval 1441.879
## 33:        4.394030 Control 95th % confidence interval 1293.189
## 34:        4.403053 Control 95th % confidence interval 1266.833
## 35:        4.371705 Control 95th % confidence interval 1238.611
## 36:        4.397692 Control 95th % confidence interval 1255.632
## 37:        4.446154  Control 5th % confidence interval 1251.038
## 38:        4.348355  Control 5th % confidence interval 1192.147
## 39:        4.388772  Control 5th % confidence interval 1128.026
## 40:        4.484669  Control 5th % confidence interval 1160.763
## 41:        4.328947  Control 5th % confidence interval 1186.826
## 42:        4.361837  Control 5th % confidence interval 1113.236
## 43:        4.349027  Control 5th % confidence interval 1007.991
## 44:        4.435811  Control 5th % confidence interval 1184.121
## 45:        4.394030  Control 5th % confidence interval 1062.011
## 46:        4.403053  Control 5th % confidence interval 1040.367
## 47:        4.371705  Control 5th % confidence interval 1017.189
## 48:        4.397692  Control 5th % confidence interval 1031.168
##     avgPricePerUnit                          store_type totSales
##     TransactionMonth
##                <Date>
##  1:      2018-07-01
##  2:      2018-08-01
##  3:      2018-09-01
##  4:      2018-10-01
##  5:      2018-11-01
##  6:      2018-12-01
##  7:      2019-01-01
##  8:      2019-02-01
##  9:      2019-03-01
## 10:      2019-04-01
## 11:      2019-05-01
## 12:      2019-06-01
## 13:      2018-07-01
## 14:      2018-08-01
## 15:      2018-09-01
## 16:      2018-10-01
## 17:      2018-11-01
## 18:      2018-12-01
## 19:      2019-01-01
## 20:      2019-02-01
## 21:      2019-03-01
## 22:      2019-04-01
## 23:      2019-05-01
## 24:      2019-06-01
```

```
## 25:       2018-07-01
## 26:       2018-08-01
## 27:       2018-09-01
## 28:       2018-10-01
## 29:       2018-11-01
## 30:       2018-12-01
## 31:       2019-01-01
## 32:       2019-02-01
## 33:       2019-03-01
## 34:       2019-04-01
## 35:       2019-05-01
## 36:       2019-06-01
## 37:       2018-07-01
## 38:       2018-08-01
## 39:       2018-09-01
## 40:       2018-10-01
## 41:       2018-11-01
## 42:       2018-12-01
## 43:       2019-01-01
## 44:       2019-02-01
## 45:       2019-03-01
## 46:       2019-04-01
## 47:       2019-05-01
## 48:       2019-06-01
##     TransactionMonth
```

```r
# Plotting these in one nice graph
ggplot(trialAssessment, aes(TransactionMonth, totSales, color = store_type)) +
geom_rect(data = trialAssessment[ MONTHYEAR < 201905 & MONTHYEAR > 201901 ,],
aes(xmin = min(TransactionMonth), xmax = max(TransactionMonth), ymin = 0 ,
ymax = Inf, color = NULL), show.legend = FALSE) + geom_line(aes(linetype = store_type)) +
labs(x = "Month of operation", y = "Total sales", title = "Total sales by month")
```



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 outside of 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.

```
scalingFactor_cust <- preTrialMeasures[
  STORE_NBR == trial_store & MONTHYEAR < 201902, sum(nCustomers)
]/ preTrialMeasures[
  STORE_NBR == control_store & MONTHYEAR < 201902, sum(nCustomers)
]
scalingFactor_cust
```

```
## [1] 0.9953052
```

```
# Apply the scaling factor
scaledControlCust <- measureOverTime[STORE_NBR ==
                                       control_store, ][, control_cust :=
                                                          nCustomers *
                                                          scalingFactor_cust]
#Now that we have comparable sales figures for the control store, we can calculate
#the percentage difference between the scaled control customer number and the trial store's
#customer number during the trial period.
percentageDiff_cust <- merge(scaledControlCust[, c(1,11)],
           measureOverTime[STORE_NBR==trial_store, c(1,4)],
           by= "MONTHYEAR")[, percentageDiff_cust:= abs(control_cust-nCustomers)/control_cust]
```

```
scaledControlCust
```

```
##     MONTHYEAR STORE_NBR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##         <num>     <int>       <num>      <int>       <num>        <num>
##  1:    201807       237      1387.2        125    1.248000     2.000000
##  2:    201808       237      1321.9        132    1.212121     1.900000
##  3:    201809       237      1250.8        120    1.183333     2.007042
##  4:    201810       237      1287.1        118    1.194915     2.035461
##  5:    201811       237      1316.0        125    1.224000     1.986928
##  6:    201812       237      1234.4        121    1.165289     2.007092
##  7:    201901       237      1117.7        111    1.162162     1.992248
##  8:    201902       237      1313.0        119    1.243697     2.000000
##  9:    201903       237      1177.6        116    1.129310     2.045802
## 10:    201904       237      1153.6        116    1.120690     2.015385
## 11:    201905       237      1127.9        122    1.155738     1.829787
## 12:    201906       237      1143.4        118    1.101695     2.000000
##     avgPricePerUnit    store_type totSales TransactionMonth control_cust
##               <num>        <char>    <num>           <Date>        <num>
##  1:        4.446154 Control store   1387.2       2018-07-01     124.4131
##  2:        4.348355 Control store   1321.9       2018-08-01     131.3803
##  3:        4.388772 Control store   1250.8       2018-09-01     119.4366
##  4:        4.484669 Control store   1287.1       2018-10-01     117.4460
##  5:        4.328947 Control store   1316.0       2018-11-01     124.4131
##  6:        4.361837 Control store   1234.4       2018-12-01     120.4319
##  7:        4.349027 Control store   1117.7       2019-01-01     110.4789
##  8:        4.435811 Control store   1313.0       2019-02-01     118.4413
##  9:        4.394030 Control store   1177.6       2019-03-01     115.4554
## 10:        4.403053 Control store   1153.6       2019-04-01     115.4554
## 11:        4.371705 Control store   1127.9       2019-05-01     121.4272
## 12:        4.397692 Control store   1143.4       2019-06-01     117.4460
```

```
percentageDiff_cust
```

```
## Key: <MONTHYEAR>
##     MONTHYEAR control_cust nCustomers percentageDiff_cust
##         <num>        <num>      <int>               <num>
##  1:    201807     124.4131        124         0.003320755
##  2:    201808     131.3803        125         0.048563465
##  3:    201809     119.4366        121         0.013089623
##  4:    201810     117.4460        120         0.021746083
##  5:    201811     124.4131        123         0.011358491
##  6:    201812     120.4319        120         0.003586465
##  7:    201901     110.4789        115         0.040922998
##  8:    201902     118.4413        122         0.030045981
##  9:    201903     115.4554        133         0.151959987
## 10:    201904     115.4554        119         0.030701041
## 11:    201905     121.4272        123         0.012952366
## 12:    201906     117.4460        113         0.037855772
```

Let's see if the difference is significant! This is to test whether the observed differences in number of customers between the trial store and the control store during the trial period are statistically significant.

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_cust <- sd(percentageDiff_cust[MONTHYEAR < 201902 , percentageDiff_cust])
stdDev_cust
```

```
## [1] 0.01791538
```

```
# 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.

```
# Calculate the t-values for the trial months
percentageDiff_cust[, tValue := (percentageDiff_cust - 0)/stdDev_cust
][, TransactionMonth := as.Date(paste(MONTHYEAR %/% 100,
                                      MONTHYEAR %% 100, 1,
                                      sep = "-"), "%Y-%m-%d")
][MONTHYEAR < 201905 & MONTHYEAR > 201901, .(TransactionMonth,tValue)]
```

```
##    TransactionMonth   tValue
##              <Date>    <num>
## 1:       2019-02-01 1.677105
## 2:       2019-03-01 8.482095
## 3:       2019-04-01 1.713669
```

```
# Find the 95th percentile of the t distribution with the appropriate
# degrees of freedom to compare against
qt(0.95, df = degreesOfFreedom)
```

```
## [1] 1.894579
```

Let's create a more visual version of this by plotting the sales of the control store, the no of customer of the trial stores and the 95th percentile value of customer of the control store.

```
#Create new variables Store_type, totSales and TransactionMonth in the data table.
pastCust <- measureOverTime[, totCust := mean(nCustomers),
                            by = c("MONTHYEAR", "store_type")
                     ][, TransactionMonth := as.Date(paste(MONTHYEAR %/% 100,
                                                           MONTHYEAR %% 100, 1,
                                                           sep = "-"), "%Y-%m-%d")
                     ][store_type %in% c("Trial store", "Control store"), ]
# Control store 95th percentile
pastCust_Controls95 <- pastCust[store_type == "Control store",
 ][, totCust := nCustomers * (1 + stdDev_cust * 2)
 ][, store_type := "Control 95th % confidence interval"]
# Control store 5th percentile
pastCust_Controls5 <- pastSales[store_type == "Control store",
 ][, totCust := nCustomers * (1 - stdDev_cust * 2)
 ][, store_type := "Control 5th % confidence interval"]
```

```
trialAssessment_cust <- rbind(pastCust, pastCust_Controls95, pastCust_Controls5)

trialAssessment_cust
```

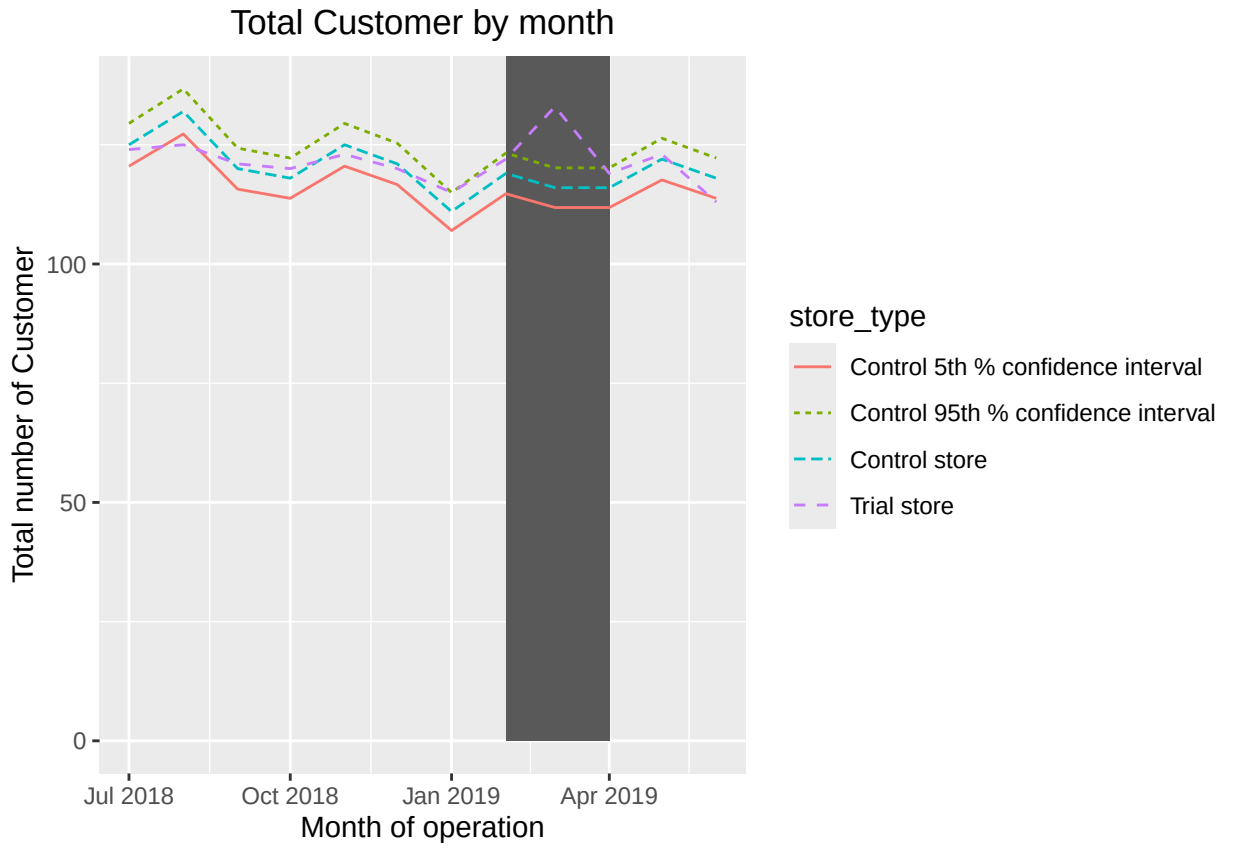
```
##     MONTHYEAR STORE_NBR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##         <num>     <int>       <num>      <int>       <num>        <num>
##  1:    201807        88     1218.20        124    1.161290     2.000000
##  2:    201808        88     1242.20        125    1.200000     1.913333
##  3:    201809        88     1361.80        121    1.247934     2.026490
##  4:    201810        88     1270.80        120    1.225000     2.040816
##  5:    201811        88     1311.40        123    1.211382     2.013423
##  6:    201812        88     1213.00        120    1.141667     2.014599
##  7:    201901        88     1215.40        115    1.217391     2.014286
##  8:    201902        88     1339.60        122    1.229508     2.013333
##  9:    201903        88     1467.00        133    1.263158     2.011905
## 10:    201904        88     1317.00        119    1.260504     2.000000
## 11:    201905        88     1236.85        123    1.195122     1.938776
## 12:    201906        88     1252.60        113    1.221239     2.028986
## 13:    201807       237     1387.20        125    1.248000     2.000000
## 14:    201808       237     1321.90        132    1.212121     1.900000
## 15:    201809       237     1250.80        120    1.183333     2.007042
## 16:    201810       237     1287.10        118    1.194915     2.035461
## 17:    201811       237     1316.00        125    1.224000     1.986928
## 18:    201812       237     1234.40        121    1.165289     2.007092
## 19:    201901       237     1117.70        111    1.162162     1.992248
## 20:    201902       237     1313.00        119    1.243697     2.000000
## 21:    201903       237     1177.60        116    1.129310     2.045802
## 22:    201904       237     1153.60        116    1.120690     2.015385
## 23:    201905       237     1127.90        122    1.155738     1.829787
## 24:    201906       237     1143.40        118    1.101695     2.000000
## 25:    201807       237     1387.20        125    1.248000     2.000000
## 26:    201808       237     1321.90        132    1.212121     1.900000
## 27:    201809       237     1250.80        120    1.183333     2.007042
## 28:    201810       237     1287.10        118    1.194915     2.035461
## 29:    201811       237     1316.00        125    1.224000     1.986928
## 30:    201812       237     1234.40        121    1.165289     2.007092
## 31:    201901       237     1117.70        111    1.162162     1.992248
## 32:    201902       237     1313.00        119    1.243697     2.000000
## 33:    201903       237     1177.60        116    1.129310     2.045802
## 34:    201904       237     1153.60        116    1.120690     2.015385
## 35:    201905       237     1127.90        122    1.155738     1.829787
## 36:    201906       237     1143.40        118    1.101695     2.000000
## 37:    201807       237     1387.20        125    1.248000     2.000000
## 38:    201808       237     1321.90        132    1.212121     1.900000
## 39:    201809       237     1250.80        120    1.183333     2.007042
## 40:    201810       237     1287.10        118    1.194915     2.035461
## 41:    201811       237     1316.00        125    1.224000     1.986928
## 42:    201812       237     1234.40        121    1.165289     2.007092
## 43:    201901       237     1117.70        111    1.162162     1.992248
## 44:    201902       237     1313.00        119    1.243697     2.000000
## 45:    201903       237     1177.60        116    1.129310     2.045802
## 46:    201904       237     1153.60        116    1.120690     2.015385
## 47:    201905       237     1127.90        122    1.155738     1.829787
```

```
## 48:    201906       237     1143.40        118    1.101695     2.000000
##     MONTHYEAR STORE_NBR TOTAL_SALES nCustomers nTxnPerCust nChipsPerTxn
##     avgPricePerUnit                           store_type totSales
##                <num>                               <char>    <num>
##  1:        4.229861                            Trial store  1218.20
##  2:        4.328223                            Trial store  1242.20
##  3:        4.450327                            Trial store  1361.80
##  4:        4.236000                            Trial store  1270.80
##  5:        4.371333                            Trial store  1311.40
##  6:        4.394928                            Trial store  1213.00
##  7:        4.309929                            Trial store  1215.40
##  8:        4.435762                            Trial store  1339.60
##  9:        4.340237                            Trial store  1467.00
## 10:        4.390000                            Trial store  1317.00
## 11:        4.339825                            Trial store  1236.85
## 12:        4.473571                            Trial store  1252.60
## 13:        4.446154                          Control store  1387.20
## 14:        4.348355                          Control store  1321.90
## 15:        4.388772                          Control store  1250.80
## 16:        4.484669                          Control store  1287.10
## 17:        4.328947                          Control store  1316.00
## 18:        4.361837                          Control store  1234.40
## 19:        4.349027                          Control store  1117.70
## 20:        4.435811                          Control store  1313.00
## 21:        4.394030                          Control store  1177.60
## 22:        4.403053                          Control store  1153.60
## 23:        4.371705                          Control store  1127.90
## 24:        4.397692                          Control store  1143.40
## 25:        4.446154 Control 95th % confidence interval  1387.20
## 26:        4.348355 Control 95th % confidence interval  1321.90
## 27:        4.388772 Control 95th % confidence interval  1250.80
## 28:        4.484669 Control 95th % confidence interval  1287.10
## 29:        4.328947 Control 95th % confidence interval  1316.00
## 30:        4.361837 Control 95th % confidence interval  1234.40
## 31:        4.349027 Control 95th % confidence interval  1117.70
## 32:        4.435811 Control 95th % confidence interval  1313.00
## 33:        4.394030 Control 95th % confidence interval  1177.60
## 34:        4.403053 Control 95th % confidence interval  1153.60
## 35:        4.371705 Control 95th % confidence interval  1127.90
## 36:        4.397692 Control 95th % confidence interval  1143.40
## 37:        4.446154  Control 5th % confidence interval  1387.20
## 38:        4.348355  Control 5th % confidence interval  1321.90
## 39:        4.388772  Control 5th % confidence interval  1250.80
## 40:        4.484669  Control 5th % confidence interval  1287.10
## 41:        4.328947  Control 5th % confidence interval  1316.00
## 42:        4.361837  Control 5th % confidence interval  1234.40
## 43:        4.349027  Control 5th % confidence interval  1117.70
## 44:        4.435811  Control 5th % confidence interval  1313.00
## 45:        4.394030  Control 5th % confidence interval  1177.60
## 46:        4.403053  Control 5th % confidence interval  1153.60
## 47:        4.371705  Control 5th % confidence interval  1127.90
## 48:        4.397692  Control 5th % confidence interval  1143.40
##     avgPricePerUnit                           store_type totSales
##     TransactionMonth  totCust
```

```
##                <Date>    <num>
##  1:       2018-07-01 124.0000
##  2:       2018-08-01 125.0000
##  3:       2018-09-01 121.0000
##  4:       2018-10-01 120.0000
##  5:       2018-11-01 123.0000
##  6:       2018-12-01 120.0000
##  7:       2019-01-01 115.0000
##  8:       2019-02-01 122.0000
##  9:       2019-03-01 133.0000
## 10:       2019-04-01 119.0000
## 11:       2019-05-01 123.0000
## 12:       2019-06-01 113.0000
## 13:       2018-07-01 125.0000
## 14:       2018-08-01 132.0000
## 15:       2018-09-01 120.0000
## 16:       2018-10-01 118.0000
## 17:       2018-11-01 125.0000
## 18:       2018-12-01 121.0000
## 19:       2019-01-01 111.0000
## 20:       2019-02-01 119.0000
## 21:       2019-03-01 116.0000
## 22:       2019-04-01 116.0000
## 23:       2019-05-01 122.0000
## 24:       2019-06-01 118.0000
## 25:       2018-07-01 129.4788
## 26:       2018-08-01 136.7297
## 27:       2018-09-01 124.2997
## 28:       2018-10-01 122.2280
## 29:       2018-11-01 129.4788
## 30:       2018-12-01 125.3355
## 31:       2019-01-01 114.9772
## 32:       2019-02-01 123.2639
## 33:       2019-03-01 120.1564
## 34:       2019-04-01 120.1564
## 35:       2019-05-01 126.3714
## 36:       2019-06-01 122.2280
## 37:       2018-07-01 120.5212
## 38:       2018-08-01 127.2703
## 39:       2018-09-01 115.7003
## 40:       2018-10-01 113.7720
## 41:       2018-11-01 120.5212
## 42:       2018-12-01 116.6645
## 43:       2019-01-01 107.0228
## 44:       2019-02-01 114.7361
## 45:       2019-03-01 111.8436
## 46:       2019-04-01 111.8436
## 47:       2019-05-01 117.6286
## 48:       2019-06-01 113.7720
##     TransactionMonth  totCust
```

```
# Plotting these in one nice graph
ggplot(trialAssessment_cust, aes(TransactionMonth, totCust, color = store_type)) +
geom_rect(data = trialAssessment_cust[ MONTHYEAR < 201905 & MONTHYEAR > 201901 ,],
```

```
aes(xmin = min(TransactionMonth), xmax = max(TransactionMonth), ymin = 0 ,
ymax = Inf, color = NULL), show.legend = FALSE) + geom_line(aes(linetype = store_type)) +
labs(x = "Month of operation", y = "Total number of Customer", title = "Total Customer by month")
```



Total number of customers in the trial period for the trial store is significantly higher than the control store for one out of three months, which indicates a positive trial effect.

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. 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.