Experiment and Uplift Testing

[Quantium Virtual Experience – Data Analytics]

Cwen Fernandes

Installing Required Libraries

library(data.table)

library(ggplot2)

library(tidyr)

Loading the dataset

data <- read.csv("R/QVI_data.csv")

head(data)

```
Console   Terminal   Background Jobs
R 4.3.1 · ~/
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
6           1005 2018-12-28          1      6       86
                                 PROD_NAME PROD_QTY TOT_SALES PACK_SIZE
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
6                      Cheetos Puffs 165g        1       2.8       165
  BRAND_NAME              LIFESTAGE PREMIUM_CUSTOMER
1    Natural  YOUNG SINGLES/COUPLES          Premium
2        Red  YOUNG SINGLES/COUPLES       Mainstream
3      Grain         YOUNG FAMILIES           Budget
4    Natural         YOUNG FAMILIES           Budget
5 Woolworths  OLDER SINGLES/COUPLES       Mainstream
6    Cheetos MIDAGE SINGLES/COUPLES       Mainstream
>
```

Set themes for plots

theme_set(theme_bw())

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.

Calculate these measures over time for each store #### Over to you! Add a new month ID column in the data with the format yyyymm.

data <- as.data.table(data)

Now, you can create the YEARMONTH column

data[, YEARMONTH := year(DATE) * 100 + month(DATE)]

```
gle symbol (e.g. DT[var]), data.table looks for var in calling scope.
> print(data[, YEARMONTH])
  [1] 201810 201809 201903 201903 201811 201812 201812 201812 201811 201809 201807 201812 201812 201903
 [15] 201906 201903 201903 201904 201906 201809 201811 201906 201901 201808 201810 201905 201810 201905
 [29] 201902 201811 201812 201807 201808 201906 201903 201903 201812 201810 201807 201902 201903 201904
 [43] 201902 201906 201902 201808 201809 201901 201904 201809 201809 201808 201904 201811 201902 201807
 [57] 201905 201807 201902 201810 201904 201905 201904 201811 201902 201902 201902 201809 201809 201810
 [71] 201807 201901 201901 201807 201904 201901 201905 201905 201811 201902 201906 201809 201902 201903
 [85] 201902 201807 201904 201809 201903 201904 201902 201812 201810 201903 201904 201901 201807 201811
 [99] 201901 201905 201810 201902 201807 201807 201903 201809 201905 201905 201905 201903 201807 201904
[113] 201812 201810 201903 201808 201809 201904 201809 201902 201904 201807 201808 201905 201809 201809
[127] 201905 201808 201902 201906 201811 201904 201903 201810 201901 201905 201902 201904 201906 201807
[141] 201809 201812 201809 201904 201811 201903 201905 201808 201905 201812 201902 201904 201807 201809
[155] 201902 201809 201812 201905 201905 201809 201808 201902 201901 201903 201905 201903 201808 201901
[169] 201810 201808 201809 201812 201810 201902 201906 201810 201812 201812 201905 201809 201809 201812
[183] 201808 201812 201811 201810 201811 201808 201807 201807 201809 201902 201905 201906 201809 201903
[197] 201904 201809 201905 201810 201809 201901 201902 201808 201901 201902 201807 201901 201902 201808
[211] 201807 201901 201901 201807 201906 201904 201808 201812 201809 201811 201903 201809 201810 201808
[225] 201809 201811 201812 201902 201903 201904 201807 201812 201901 201906 201809 201810 201903 201901
[239] 201902 201904 201809 201903 201812 201906 201904 201906 201807 201809 201901 201905 201903 201810
```

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[, .(totSales = 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 = c("STORE_NBR", "YEARMONTH")][order(STORE_NBR, YEARMONTH)]

Filter to the pre-trial period and stores with full observation periods

storesWithFullObs <- unique(measureOverTime[, .N, STORE_NBR][N == 12, STORE_NBR])

preTrialMeasures <- measureOverTime[YEARMONTH < 201902 & STORE_NBR %in%

```
                                      storesWithFullObs, ]

#### Create a function to calculate correlation for a measure, looping through each control store.
#### 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.
calculateCorrelation <- function(inputTable, metricCol, storeComparison) {
  calcCorrTable = data.table(Store1 = numeric(), Store2 = numeric(), corr_measure =
                               numeric())

  storeNumbers <- unique(inputTable[, STORE_NBR])

  for (i in storeNumbers) {
    calculatedMeasure = data.table("Store1" = storeComparison,
                                   "Store2" = i,
                                   "corr_measure" = cor( inputTable[STORE_NBR == storeComparison,
                                                        eval(metricCol)], inputTable[STORE_NBR == i,
                                                                      eval(metricCol)]))
    calcCorrTable <- rbind(calcCorrTable, calculatedMeasure)
  }
  return(calcCorrTable)
}

#### Create a function to calculate a standardised magnitude distance for a measure,
#### looping through each control store
calculateMagnitudeDistance <- function(inputTable, metricCol, storeComparison) {
  calcDistTable = data.table(Store1 = numeric(), Store2 = numeric(), YEARMONTH =
                               numeric(), measure = numeric())
  storeNumbers <- unique(inputTable[, STORE_NBR])
  for (i in storeNumbers) {
    calculatedMeasure = data.table("Store1" = storeComparison
```

```r
                        , "Store2" = i
                        , "YEARMONTH" = inputTable[STORE_NBR ==
                                          storeComparison, YEARMONTH]
                        , "measure" = abs(inputTable[STORE_NBR ==
                                           storeComparison, eval(metricCol)]
                                    - inputTable[STORE_NBR == i,
                                           eval(metricCol)])
   )
   calcDistTable <- rbind(calcDistTable, calculatedMeasure)
  }

#### Standardise the magnitude distance so that the measure ranges from 0 to 1
  minMaxDist <- calcDistTable[, .(minDist = min(measure), maxDist = max(measure)),
                    by = c("Store1", "YEARMONTH")]
  distTable <- merge(calcDistTable, minMaxDist, by = c("Store1", "YEARMONTH"))
  distTable[, magnitudeMeasure := 1 - (measure - minDist)/(maxDist - minDist)]
  finalDistTable <- distTable[, .(mag_measure = mean(magnitudeMeasure)), by =
                     .(Store1, Store2)]
  return(finalDistTable)
}
#### Use the function you created to calculate correlations
#### We will select control store vased on how similar monthly total sales in dollar amounts and
monthly number of customers are to the trial stores.
#### store 77 using total sales and number of customers.
trial_store <- 77
corr_nSales <- calculateCorrelation(preTrialMeasures, quote(totSales), trial_store)
corr_nSales[order(-corr_measure)]
corr_nCustomers <- calculateCorrelation(preTrialMeasures, quote(nCustomers), trial_store)
corr_nCustomers[order(-corr_measure)]
```

```
  return(calcCorrTable)
}
<bytecode: 0x000001686d5ed540>
> corr_nCustomers[order(-corr_measure)]
     Store1 Store2 corr_measure
  1:     77     77    1.0000000
  2:     77    233    0.9656821
  3:     77    119    0.9190639
  4:     77    113    0.9016299
  5:     77    254    0.9016105
 ---
255:     77    227   -0.7506291
256:     77    186   -0.7668731
257:     77    169   -0.7842412
258:     77      9   -0.8045038
259:     77     54   -0.8314799
> #### Then, use the functions for calculating magnitude.
> magnitude_nSales <- calculateMagnitudeDistance(preTrialMeasures, quote(totSales),
+                                                trial_store)
> magnitude_nCustomers <- calculateMagnitudeDistance(preTrialMeasures,
+                                                    quote(nCustomers), trial_store)
> |
```

```
#### Then, use the functions for calculating magnitude.

magnitude_nSales <- calculateMagnitudeDistance(preTrialMeasures, quote(totSales),
                                               trial_store)

magnitude_nCustomers <- calculateMagnitudeDistance(preTrialMeasures,
                                                   quote(nCustomers), trial_store)


#### 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: 0.5 * corr_measure + 0.5 * mag_measure

corr_weight <- 0.5

score_nSales <- merge(corr_nSales, magnitude_nSales, by =
                c("Store1","Store2"))[, scoreNSales := (corr_measure + mag_measure)/2 ]

score_nCustomers <- merge(corr_nCustomers, magnitude_nCustomers, by =
                    c("Store1", "Store2"))[, scoreNCust := (corr_measure + mag_measure)/2]



score_nSales[order(-scoreNSales)]
```

```
R 4.3.1 · ~/
> score_nSales[order(-scoreNSales)]
     Store1 Store2 corr_measure mag_measure scoreNSales
  1:     77     77    1.0000000   1.0000000  1.00000000
  2:     77    233    0.9736429   0.9864521  0.98004751
  3:     77     50    0.8977013   0.9758393  0.93677029
  4:     77     41    0.6591279   0.9595727  0.80935028
  5:     77    167    0.6546680   0.9566152  0.80564160
 ---
255:     77     55   -0.6187149   0.5005030 -0.05910599
256:     77      4   -0.3478465   0.2204245 -0.06371097
257:     77    247   -0.7109062   0.5525805 -0.07916288
258:     77    138   -0.6941490   0.5258652 -0.08414190
259:     77     75   -0.7952057   0.3479722 -0.22361678
>
```

score_nCustomers[order(-scoreNCust)]

```
259:     77     75   -0.7952057   0.3479722 -0.22361678
>
> score_nCustomers[order(-scoreNCust)]
     Store1 Store2 corr_measure mag_measure  scoreNCust
  1:     77     77    1.0000000   1.0000000  1.00000000
  2:     77    233    0.9656821   0.9909635  0.97832280
  3:     77    254    0.9016105   0.9295040  0.91555724
  4:     77     35    0.8927414   0.8995606  0.89615102
  5:     77     84    0.8515210   0.9230425  0.88728178
 ---
255:     77    165   -0.3647459   0.1809069 -0.09191945
256:     77    147   -0.7148957   0.5095139 -0.10269086
257:     77    102   -0.6371093   0.4300131 -0.10354806
258:     77     75   -0.5650164   0.3419888 -0.11151380
259:     77    227   -0.7506291   0.4317654 -0.15943183
>
```

Combine scores across the drivers by first merging our sales scores and customer scores into a single table

score_Control <- merge(score_nSales, score_nCustomers, by = c("Store1","Store2"))

score_Control[, finalControlScore := scoreNSales * 0.5 + scoreNCust * 0.5]

score_Control[order(-finalControlScore)]

```
>
> score_Control[order(-finalControlScore)]
     Store1 Store2 corr_measure.x mag_measure.x scoreNSales corr_measure.y mag_measure.y  scoreNCust
  1:     77     77      1.0000000     1.0000000  1.00000000      1.0000000     1.0000000  1.00000000
  2:     77    233      0.9736429     0.9864521  0.98004751      0.9656821     0.9909635  0.97832280
  3:     77     50      0.8977013     0.9758393  0.93677029      0.7093977     0.9319895  0.82069361
  4:     77     35      0.6910897     0.9125854  0.80183757      0.8927414     0.8995606  0.89615102
  5:     77    254      0.5848729     0.9235071  0.75419002      0.9016105     0.9295040  0.91555724
 ---
255:     77    247     -0.7109062     0.5525805 -0.07916288     -0.5436841     0.4531046 -0.04528974
256:     77    147     -0.6640142     0.5817530 -0.04113060     -0.7148957     0.5095139 -0.10269086
257:     77    227     -0.5040822     0.5294106  0.01266416     -0.7506291     0.4317654 -0.15943183
258:     77    138     -0.6941490     0.5258652 -0.08414190     -0.5483439     0.4165489 -0.06589749
259:     77     75     -0.7952057     0.3479722 -0.22361678     -0.5650164     0.3419888 -0.11151380
     finalControlScore
  1:        1.00000000
  2:        0.97918516
  3:        0.87873195
  4:        0.84899429
  5:        0.83487363
 ---
255:       -0.06222631
256:       -0.07191073
257:       -0.07338384
258:       -0.07501969
259:       -0.16756529
>
```

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[Store1 == trial_store,][order(-finalControlScore)][2, Store2]

control_store

```
> control_store <- score_Control[Store1 == trial_store, ][order(-finalControlScore)][2, Store2]
> control_store
[1] 233
>
```

Visual checks on trends based on the drivers

measureOverTimeSales <- measureOverTime

pastSales <- measureOverTimeSales[, Store_type := ifelse(STORE_NBR == trial_store,

"Trial",

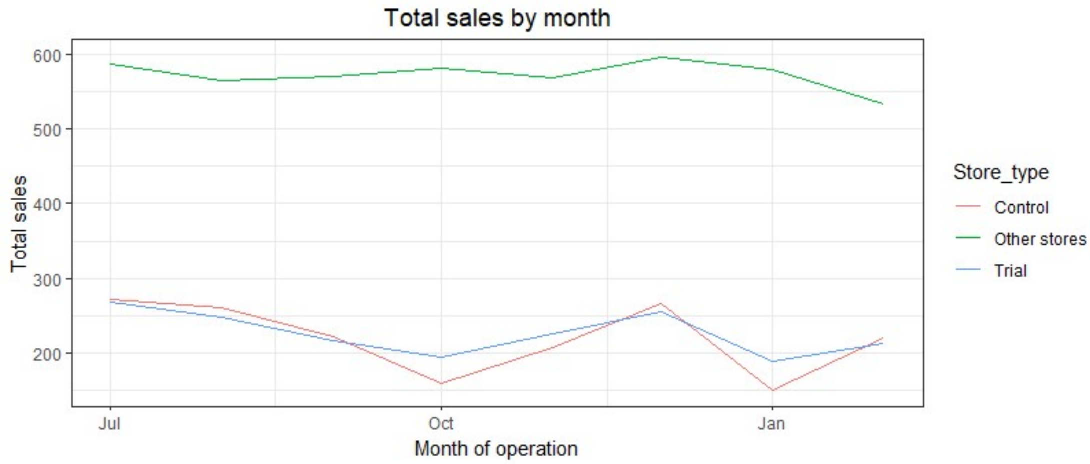
ifelse(STORE_NBR == control_store,

"Control", "Other stores"))][, totSales := mean(totSales), by = c("YEARMONTH", "Store_type")][, TransactionMonth := as.Date(paste(YEARMONTH %/%

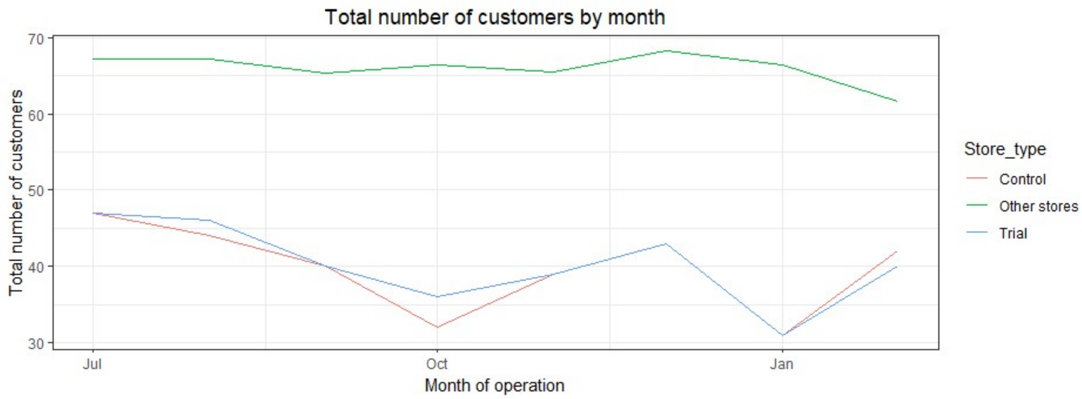
100, YEARMONTH %% 100, 1, sep = "-"), "%Y-%m-%d")][YEARMONTH < 201903 ,]

ggplot(pastSales, aes(TransactionMonth, totSales, color = Store_type)) +

```
 geom_line() +

 labs(x = "Month of operation", y = "Total sales", title = "Total sales by month")
```




```
#### Conduct visual checks on customer count trends by comparing the trial store

#### to the control store and other stores.

measureOverTimeCusts <- measureOverTime

pastCustomers <- measureOverTimeCusts[, Store_type := ifelse(STORE_NBR == trial_store, "Trial",

                                                             ifelse(STORE_NBR == control_store, "Control", "Other stores"))

][, numberCustomers := mean(nCustomers), by = c("YEARMONTH", "Store_type")

][, TransactionMonth := as.Date(paste(YEARMONTH %/% 100, YEARMONTH %% 100, 1, sep = "-"), "%Y-%m-%d")

][YEARMONTH < 201903 , ]

ggplot(pastCustomers, aes(TransactionMonth, numberCustomers, color = Store_type)) +

 geom_line() +

 labs(x = "Month of operation", 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 want to see if there has been an uplift overall chip sale.

Scale pre-trial control sales to match pre-trial trial store sales

```
scalingFactorForControlSales <- preTrialMeasures[STORE_NBR == trial_store &
                                    YEARMONTH < 201902, sum(totSales)]/preTrialMeasures[STORE_NBR ==
control_store &
                                                  YEARMONTH < 201902, sum(totSales)]
```

Apply the scaling factor

```
measureOverTimeSales <- measureOverTime
scaledControlSales <- measureOverTimeSales[STORE_NBR == control_store, ][ ,
                                        controlSales := totSales * scalingFactorForControlSales]
```

Calculate the percentage difference between scaled control sales and trial sales

```
percentageDiff <- merge(scaledControlSales[, c("YEARMONTH", "controlSales")],
              measureOverTime[STORE_NBR == trial_store, c("totSales", "YEARMONTH")],
              by = "YEARMONTH")[, percentageDiff := abs(controlSales-totSales)/controlSales]

percentageDiff # between control store sales and trial store sales
```

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[YEARMONTH < 201902 , percentageDiff])
```

Note that there are 8 months in the pre-trial period

hence 8 - 1 = 7 degrees of freedom

degreesOfFreedom <- 7

We will test with a null hypothesis of there being 0 difference between trial and control stores.

Calculate the t-values for the trial months. After that, find the 95th percentile of the t distribution with the appropriate degrees of freedom to check whether the hypothesis is statistically significant.

The test statistic here is (x - u)/standard deviation

percentageDiff[, tValue := (percentageDiff - 0)/stdDev

][, TransactionMonth := as.Date(paste(YEARMONTH %/% 100, YEARMONTH %% 100, 1,

sep = "-"), "%Y-%m-%d")

][YEARMONTH < 201905 & YEARMONTH > 201901, .(TransactionMonth,tValue)]

```
+                                                   sep =
+ ][YEARMONTH < 201905 & YEARMONTH > 201901,
   TransactionMonth    tValue
1:       2019-02-01  1.223912
2:       2019-03-01  5.633494
3:       2019-04-01 11.336505
>
```

Find the 95th percentile of the t distribution with the appropriate

degrees of freedom to compare against

qt(0.95, df = degreesOfFreedom)

```
> #### 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 larger 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 greater than in the control store.

measureOverTimeSales <- measureOverTime

Trial and control store total sales

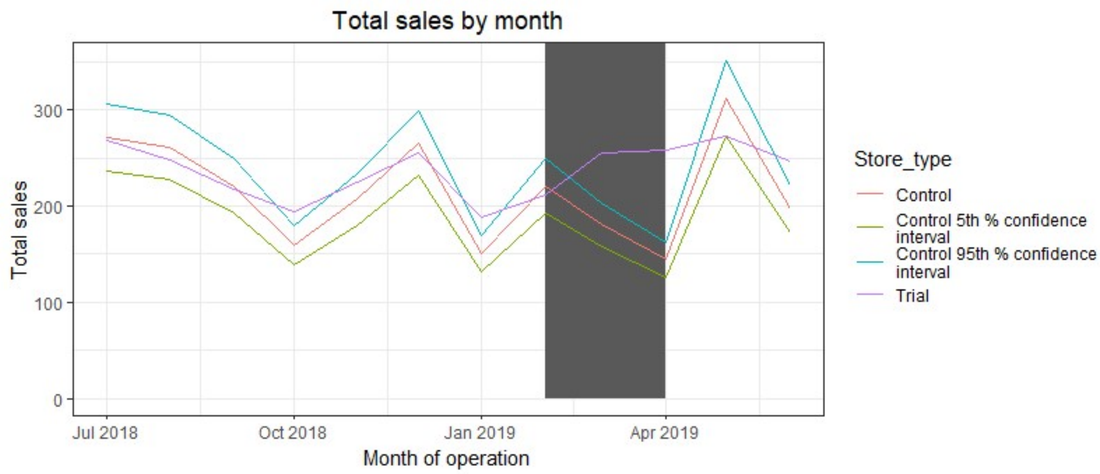
Create new variables Store_type, totSales and TransactionMonth in the data table.

pastSales <- measureOverTimeSales[, Store_type := ifelse(STORE_NBR == trial_store, "Trial",

ifelse(STORE_NBR == control_store, "Control", "Other stores"))

][, totSales := mean(totSales), by = c("YEARMONTH", "Store_type")

```
][, TransactionMonth := as.Date(paste(YEARMONTH %/% 100, YEARMONTH %% 100, 1, sep = "-"),
"%Y-%m-%d")
][Store_type %in% c("Trial", "Control"), ]


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


#### Control store 5th percentile
pastSales_Controls5 <- pastSales[Store_type == "Control",
][, totSales := totSales * (1 - stdDev * 2)
][, Store_type := "Control 5th % confidence
interval"]
trialAssessment <- rbind(pastSales, pastSales_Controls95, pastSales_Controls5)



#### Plotting these in one nice graph
ggplot(trialAssessment, aes(TransactionMonth, totSales, color = Store_type)) +
  geom_rect(data = trialAssessment[ YEARMONTH < 201905 & YEARMONTH > 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 trail 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 two of the three trial months.

This would be a repeat of the steps before for total sales

Scale pre-trial control customers to match pre-trial trial store customers

Compute a scaling factor to align control store customer counts to our trial store.

Then, apply the scaling factor to control store customer counts.

Finally, calculate the percentage difference between scaled control store customers and trial customers.

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

measureOverTimeCusts <- measureOverTime

scaledControlCustomers <- measureOverTimeCusts[STORE_NBR == control_store,
][ , controlCustomers := nCustomers * scalingFactorForControlCust
][, Store_type := ifelse(STORE_NBR ==trial_store, "Trial",
          ifelse(STORE_NBR == control_store,"Control", "Other stores"))]


percentageDiff <- merge(scaledControlCustomers[, c("YEARMONTH", "controlCustomers")],
          measureOverTimeCusts[STORE_NBR == trial_store,c("nCustomers", "YEARMONTH")],
          by = "YEARMONTH"
)[, percentageDiff := abs(controlCustomers-nCustomers)/controlCustomers]
```

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[YEARMONTH < 201902 , percentageDiff])

degreesOfFreedom <- 7

Trial and control store number of customers

pastCustomers <- measureOverTimeCusts[, nCusts := mean(nCustomers), by =

c("YEARMONTH", "Store_type")

][Store_type %in% c("Trial", "Control"),]

Control store 95th percentile

pastCustomers_Controls95 <- pastCustomers[Store_type == "Control",

][, nCusts := nCusts * (1 + stdDev * 2)

][, Store_type := "Control 95th % confidence

interval"]

Control store 5th percentile

pastCustomers_Controls5 <- pastCustomers[Store_type == "Control",

][, nCusts := nCusts * (1 - stdDev * 2)

][, Store_type := "Control 5th % confidence

interval"]

trialAssessment <- rbind(pastCustomers, pastCustomers_Controls95,

pastCustomers_Controls5)

Plotting a graph: geom_rect creates a rectangle in the plot. Use this to highlight the

trial period in our graph.

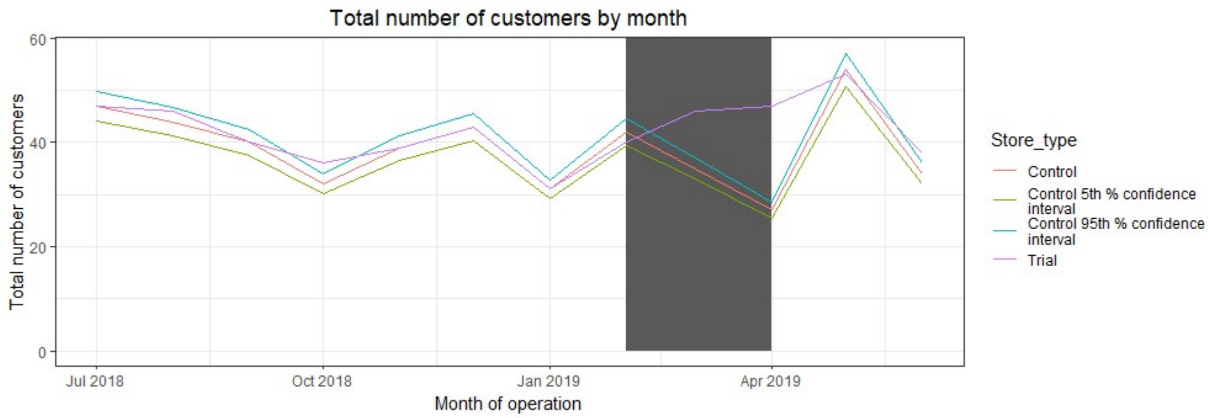
ggplot(trialAssessment, aes(TransactionMonth, nCusts, color = Store_type)) +

geom_rect(data = trialAssessment[YEARMONTH < 201905 & YEARMONTH > 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 customers", title = "Total number of customers by month")



Trial store 86

Calculate the metrics below as we did for the first trial store.

measureOverTime <- data[, .(totSales = 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 = c("STORE_NBR", "YEARMONTH")][order(STORE_NBR, YEARMONTH)]

Use the functions we created earlier to calculate correlations and magnitude for each potential control store

trial_store <- 86

corr_nSales <- calculateCorrelation(preTrialMeasures, quote(totSales),trial_store)

```
> corr_nSales <- calculateCorrelation(preTrialMeasures, quote(totSales),trial_store)
function(inputTable, metricCol, storeComparison) {
  calcCorrTable = data.table(Store1 = numeric(), Store2 = numeric(), corr_measure =
                              numeric())

  storeNumbers <- unique(inputTable[, STORE_NBR])
calculateCorrelation
print(calculateCorrelation)
print(calculateCorrelation)
for (i in storeNumbers) {
    calculatedMeasure = data.table("Store1" = storeComparison,
                                   "Store2" = i,
                                   "corr_measure" = cor( inputTable[STORE_NBR == storeComparison,
                                                                    eval(metricCol)], inputTable[STORE_NBR == i,
                                                                                                 eval(metricCol)]))
    calcCorrTable <- rbind(calcCorrTable, calculatedMeasure)
  }
  return(calcCorrTable)
}
<bytecode: 0x000001686d5ed540>
function(inputTable, metricCol, storeComparison) {
  calcCorrTable = data.table(Store1 = numeric(), Store2 = numeric(), corr_measure =
                              numeric())

  storeNumbers <- unique(inputTable[, STORE_NBR])
calculateCorrelation
print(calculateCorrelation)
print(calculateCorrelation)
for (i in storeNumbers) {
    calculatedMeasure = data.table("Store1" = storeComparison,
                                   "Store2" = i,
                                   "corr_measure" = cor( inputTable[STORE_NBR == storeComparison,
                                                                    eval(metricCol)], inputTable[STORE_NBR == i,
                                                                                                 eval(metricCol)]))
    calcCorrTable <- rbind(calcCorrTable, calculatedMeasure)
  }
  return(calcCorrTable)
```

corr_nCustomers <- calculateCorrelation(preTrialMeasures, quote(nCustomers), trial_store)

```
> corr_nCustomers <- calculateCorrelation(preTrialMeasures, quote(nCustomers), trial_store)
function(inputTable, metricCol, storeComparison) {
  calcCorrTable = data.table(Store1 = numeric(), Store2 = numeric(), corr_measure =
                              numeric())

  storeNumbers <- unique(inputTable[, STORE_NBR])
calculateCorrelation
print(calculateCorrelation)
print(calculateCorrelation)
for (i in storeNumbers) {
    calculatedMeasure = data.table("Store1" = storeComparison,
                                   "Store2" = i,
                                   "corr_measure" = cor( inputTable[STORE_NBR == storeComparison,
                                                                    eval(metricCol)], inputTable[STORE_NBR == i,
                                                                                                 eval(metricCol)]))
    calcCorrTable <- rbind(calcCorrTable, calculatedMeasure)
  }
  return(calcCorrTable)
}
<bytecode: 0x000001686d5ed540>
function(inputTable, metricCol, storeComparison) {
  calcCorrTable = data.table(Store1 = numeric(), Store2 = numeric(), corr_measure =
                              numeric())

  storeNumbers <- unique(inputTable[, STORE_NBR])
calculateCorrelation
print(calculateCorrelation)
print(calculateCorrelation)
for (i in storeNumbers) {
    calculatedMeasure = data.table("Store1" = storeComparison,
                                   "Store2" = i,
                                   "corr_measure" = cor( inputTable[STORE_NBR == storeComparison,
                                                                    eval(metricCol)], inputTable[STORE_NBR == i,
                                                                                                 eval(metricCol)]))
    calcCorrTable <- rbind(calcCorrTable, calculatedMeasure)
  }
```

magnitude_nSales <- calculateMagnitudeDistance(preTrialMeasures, quote(totSales), trial_store)

magnitude_nCustomers <- calculateMagnitudeDistance(preTrialMeasures, quote(nCustomers), trial_store)

Now, create a combined score composed of correlation and magnitude

corr_weight <- 0.5

score_nSales <- merge(corr_nSales, magnitude_nSales, by = c("Store1", "Store2"))[, scoreNSales := (corr_measure + mag_measure)/2] score_nCustomers <- merge(corr_nCustomers, magnitude_nCustomers, by = c("Store1", "Store2"))[, scoreNCust := (corr_measure + mag_measure)/2]

Finally, combine scores across the drivers using a simple average.

score_Control <- merge(score_nSales, score_nCustomers, by = c("Store1","Store2"))

score_Control[, finalControlScore := scoreNSales * 0.5 + scoreNCust * 0.5]

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 control store for trial store 86

control_store <- score_Control[Store1 == trial_store,

][order(-finalControlScore)][2, Store2]

control_store

```
+ ][order(-finalControl
> control_store
[1] 155
>
```

Looks like store 155 will be a control store for trial store 86.

Conduct visual checks on trends based on the drivers

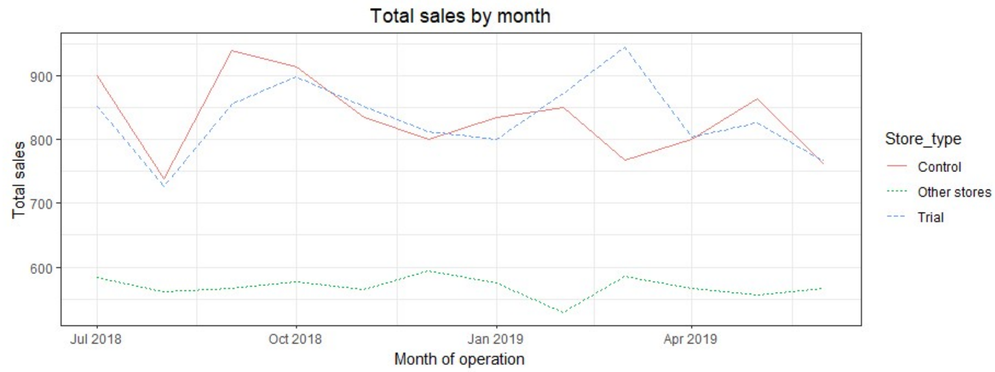
measureOverTimeSales <- measureOverTime

pastSales <- measureOverTimeSales[, Store_type:= ifelse(STORE_NBR == trial_store, "Trial", ifelse(STORE_NBR== control_store, "Control", "Other stores"))][, totSales := mean(totSales), by = c("YEARMONTH", "Store_type")][, TransactionMonth:= as.Date(paste(YEARMONTH%/%100, YEARMONTH%% 100, 1, sep = "-"), "%Y-%m-%d")][YEARMONTH <210903]

ggplot(pastSales, aes(TransactionMonth, totSales, color = Store_type)) +

geom_line(aes(linetype = Store_type)) +

labs(x = "Month of operation", y = "Total sales", title = "Total sales by month")



Conduct visual checks on trends based on the drivers

measureOverTimeCusts <- measureOverTime

pastCustomers <- measureOverTimeCusts[, Store_type := ifelse(STORE_NBR == trial_store, "Trial",

ifelse(STORE_NBR == control_store, "Control", "Other stores"))

][, numberCustomers := mean(nCustomers), by = c("YEARMONTH", "Store_type")

][, TransactionMonth := as.Date(paste(YEARMONTH %/%

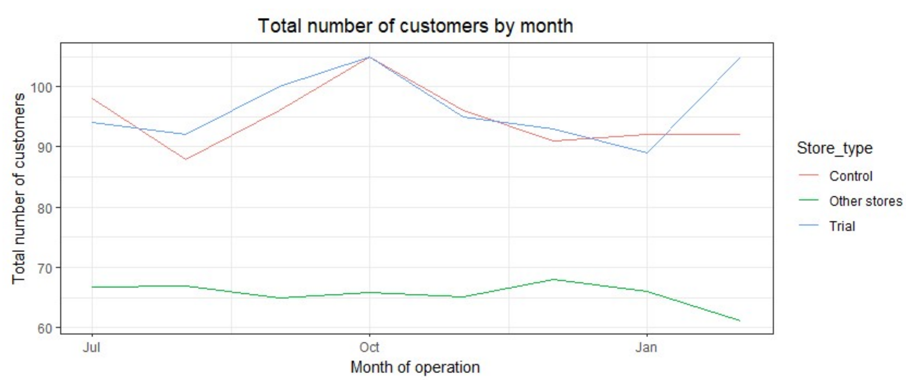
100, YEARMONTH %% 100, 1, sep = "-"), "%Y-%m-%d")

][YEARMONTH < 201903 ,]

ggplot(pastCustomers, aes(TransactionMonth, numberCustomers, color = Store_type)) +

geom_line() +

labs(x = "Month of operation", y = "Total number of customers", title = "Total number of customers by month")



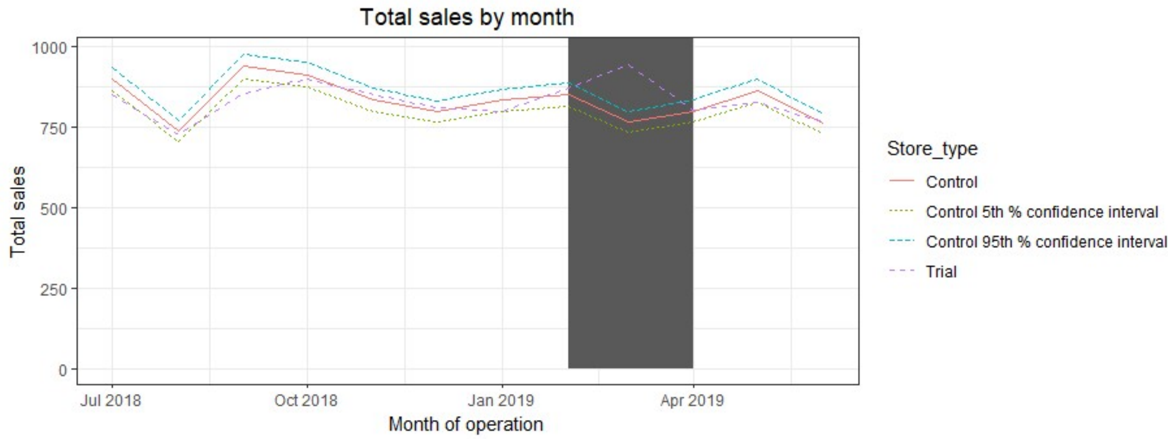
Good, the trend in the number of customers is also similar.

Scale pre-trial control sales to match pre-trial trial store sales

scalingFactorForControlSales <- preTrialMeasures[STORE_NBR == trial_store &

```
                                YEARMONTH < 201902, sum(totSales)]/preTrialMeasures[STORE_NBR ==
control_store &
                                                  YEARMONTH < 201902, sum(totSales)]

#### Apply the scaling factor

measureOverTimeSales <- measureOverTime

scaledControlSales <- measureOverTimeSales[STORE_NBR == control_store, ][ ,

                                  controlSales := totSales * scalingFactorForControlSales]

#### Calculate the percentage difference between scaled control sales and trial sales

#### When calculating percentage difference, remember to use absolute difference

percentageDiff <- merge(scaledControlSales[, c("YEARMONTH", "controlSales")],

        measureOverTime[STORE_NBR == trial_store, c("totSales", "YEARMONTH")],

        by = "YEARMONTH"

)[, percentageDiff := abs(controlSales-totSales)/controlSales]


#### 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

#### Calculate the standard deviation of percentage differences during the pre-trial period

stdDev <- sd(percentageDiff[YEARMONTH < 201902 , percentageDiff])

degreesOfFreedom <- 7


#### Trial and control store total sales

#### Create a table with sales by store type and month.

#### We only need data for the trial and control store.

measureOverTimeSales <- measureOverTime

pastSales <- measureOverTimeSales[, Store_type := ifelse(STORE_NBR == trial_store, "Trial",

                            ifelse(STORE_NBR == control_store, "Control", "Other stores"))

][, totSales := mean(totSales), by = c("YEARMONTH", "Store_type")

][, TransactionMonth := as.Date(paste(YEARMONTH %/%100, YEARMONTH %% 100, 1, sep = "-"),
"%Y-%m-%d")

][Store_type %in% c("Trial", "Control"), ]


#### Calculate the 5th and 95th percentile for control store sales.
```

```
#### The 5th and 95th percentiles can be approximated by using two standard deviations away from the mean.

#### Recall that the variable stdDev earlier calculates standard deviation in percentages, and not dollar sales.

#### Control store 95th percentile

pastSales_Controls95 <- pastSales[Store_type == "Control",

][, totSales := totSales * (1 + stdDev * 2)

][, Store_type := "Control 95th % confidence interval"]


#### Control store 5th percentile

pastSales_Controls5 <- pastSales[Store_type == "Control",

][, totSales := totSales * (1 - stdDev * 2)

][, Store_type := "Control 5th % confidence interval"]


#### Then, create a combined table with columns from pastSales, pastSales_Controls95 and pastSales_Controls5

trialAssessment <- rbind(pastSales, pastSales_Controls95, pastSales_Controls5)

#### Plotting a graph

ggplot(trialAssessment, aes(TransactionMonth, totSales, color = Store_type)) +

  geom_rect(data = trialAssessment[ YEARMONTH < 201905 & YEARMONTH > 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.

This would be a repeat of the steps before for total sales

Scale pre-trial control customers to match pre-trial trial store customers

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

Apply the scaling factor

```
measureOverTimeCusts <- measureOverTime
scaledControlCustomers <- measureOverTimeCusts[STORE_NBR == control_store,
][ , controlCustomers := nCustomers
  * scalingFactorForControlCust
][, Store_type := ifelse(STORE_NBR
                 == trial_store, "Trial",
                 ifelse(STORE_NBR == control_store,
                      "Control", "Other stores"))
]
```

Calculate the percentage difference between scaled control sales and trial sales

```
percentageDiff <- merge(scaledControlCustomers[, c("YEARMONTH",
```

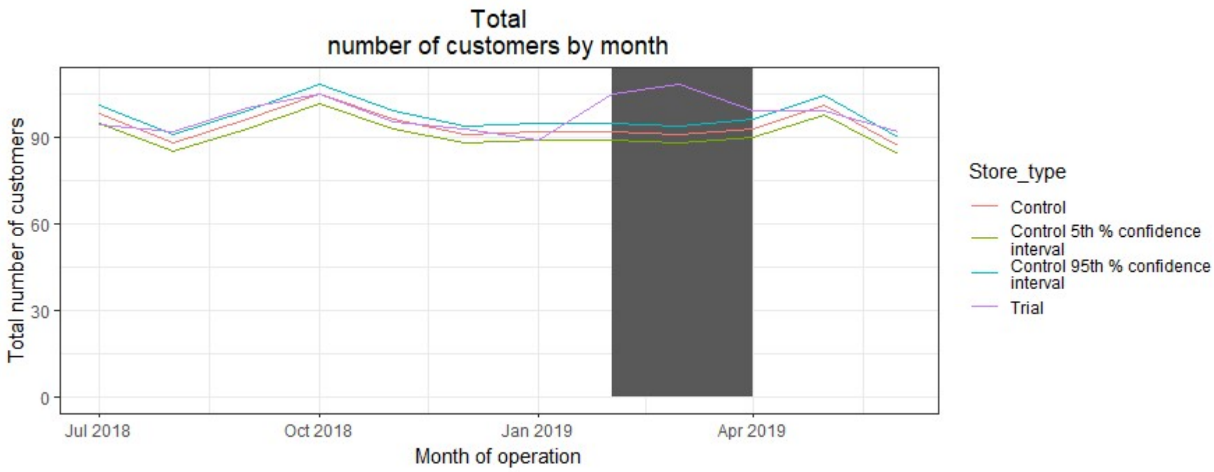
```
                                        "controlCustomers")],
                    measureOverTime[STORE_NBR == trial_store, c("nCustomers",
                                                     "YEARMONTH")],
                    by = "YEARMONTH"
)[, percentageDiff :=
   abs(controlCustomers-nCustomers)/controlCustomers]
```

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[YEARMONTH < 201902 , percentageDiff])

degreesOfFreedom <- 7

#### Trial and control store number of customers

pastCustomers <- measureOverTimeCusts[, nCusts := mean(nCustomers), by =
                               c("YEARMONTH", "Store_type")
][Store_type %in% c("Trial", "Control"), ]

#### Control store 95th percentile

pastCustomers_Controls95 <- pastCustomers[Store_type == "Control",
][, nCusts := nCusts * (1 + stdDev * 2)
][, Store_type := "Control 95th % confidence
interval"]

#### Control store 5th percentile

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

trialAssessment <- rbind(pastCustomers, pastCustomers_Controls95,
                     pastCustomers_Controls5)

#### Plotting a graph

ggplot(trialAssessment, aes(TransactionMonth, nCusts, color = Store_type)) +
  geom_rect(data = trialAssessment[ YEARMONTH < 201905 & YEARMONTH > 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 customers", title = "Total number of customers by month")



Trial store 88

Conduct the analysis on trial store 88.

```
measureOverTime <- data[, .(totSales = 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 = c("STORE_NBR", "YEARMONTH")][order(STORE_NBR, YEARMONTH)]
```

Use the functions from earlier to calculate the correlation of the sales and number of customers of each potential control store to the trial store

```
trial_store <- 88

corr_nSales <- calculateCorrelation(preTrialMeasures, quote(totSales),trial_store)
```

```
> corr_nSales <- calculateCorrelation(preTrialMeasures, quote(totSales),trial_store)
function(inputTable, metricCol, storeComparison) {
  calcCorrTable = data.table(Store1 = numeric(), Store2 = numeric(), corr_measure =
                             numeric())

  storeNumbers <- unique(inputTable[, STORE_NBR])
calculateCorrelation
print(calculateCorrelation)
print(calculateCorrelation)
for (i in storeNumbers) {
    calculatedMeasure = data.table("Store1" = storeComparison,
                                   "Store2" = i,
                                   "corr_measure" = cor( inputTable[STORE_NBR == storeCompari
n,
                                                                   eval(metricCol)], inputTa
e[STORE_NBR == i,

eval(metricCol)]))
    calcCorrTable <- rbind(calcCorrTable, calculatedMeasure)
  }
  return(calcCorrTable)
}
<bytecode: 0x000001686d5ed540>
function(inputTable, metricCol, storeComparison) {
  calcCorrTable = data.table(Store1 = numeric(), Store2 = numeric(), corr_measure =
                             numeric())

  storeNumbers <- unique(inputTable[, STORE_NBR])
calculateCorrelation
```

corr_nCustomers <- calculateCorrelation(preTrialMeasures, quote(nCustomers), trial_store)

```
> corr_nCustomers <- calculateCorrelation(preTrialMeasures, quote(nCustomers), trial_store)
function(inputTable, metricCol, storeComparison) {
  calcCorrTable = data.table(Store1 = numeric(), Store2 = numeric(), corr_measure =
                             numeric())

  storeNumbers <- unique(inputTable[, STORE_NBR])
calculateCorrelation
print(calculateCorrelation)
print(calculateCorrelation)
for (i in storeNumbers) {
    calculatedMeasure = data.table("Store1" = storeComparison,
                                   "Store2" = i,
                                   "corr_measure" = cor( inputTable[STORE_NBR == storeCompariso
n,
                                                                   eval(metricCol)], inputTabl
e[STORE_NBR == i,

eval(metricCol)]))
    calcCorrTable <- rbind(calcCorrTable, calculatedMeasure)
  }
  return(calcCorrTable)
}
<bytecode: 0x000001686d5ed540>
function(inputTable, metricCol, storeComparison) {
  calcCorrTable = data.table(Store1 = numeric(), Store2 = numeric(), corr_measure =
                             numeric())

  storeNumbers <- unique(inputTable[, STORE_NBR])
```

Use the functions from earlier to calculate the magnitude distance of the sales and number of customers of each potential control store to the trial store

magnitude_nSales <- calculateMagnitudeDistance(preTrialMeasures, quote(totSales), trial_store)

magnitude_nCustomers <- calculateMagnitudeDistance(preTrialMeasures, quote(nCustomers), trial_store)

Create a combined score composed of correlation and magnitude by merging the correlations table and the magnitudes table, for each driver.

corr_weight <- 0.5

score_nSales <- merge(corr_nSales, magnitude_nSales, by = c("Store1", "Store2"))[, scoreNSales := (corr_measure + mag_measure)/2]

score_nCustomers <- merge(corr_nCustomers, magnitude_nCustomers, by = c("Store1", "Store2"))[, scoreNCust := (corr_measure + mag_measure)/2]

Combine scores across the drivers by merging sales scores and customer scores, and compute a final combined score.

score_Control <- merge(score_nSales, score_nCustomers, by = c("Store1","Store2"))

score_Control[, finalControlScore := scoreNSales * 0.5 + scoreNCust * 0.5]

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 control store for trial store 88

control_store <- score_Control[Store1 == trial_store,][order(-finalControlScore)][2, Store2]

control_store

We've now found store 237 to be suitable control store for trial store 88./

Visual checks on trends based on the drivers

For the period before the trial, create a graph with total sales of the trial

store for each month, compared to the control store and other stores.

```
measureOverTimeSales <- measureOverTime

pastSales <- measureOverTimeSales[, Store_type := ifelse(STORE_NBR == trial_store, "Trial",

                                        ifelse(STORE_NBR == control_store, "Control", "Other stores"))

][, totSales := mean(totSales), by = c("YEARMONTH","Store_type")

][, TransactionMonth := as.Date(paste(YEARMONTH %/% 100, YEARMONTH %% 100, 1, sep = "-"), "%Y-%m-%d")

][YEARMONTH < 201903 , ]

ggplot(pastSales, aes(TransactionMonth, totSales, color = Store_type)) +

  geom_line(aes(linetype = Store_type)) +

  labs(x = "Month of operation", y = "Total sales", title = "Total sales by month")
```

Great, the trial and control stores have similar total sales.

Next, number of customers

Visual checks on trends based on the drivers

For the period before the trial, create a graph with customer counts of the

trial store for each month, compared to the control store and other stores.

measureOverTimeCusts <- measureOverTime

pastCustomers <- measureOverTimeCusts[, Store_type := ifelse(STORE_NBR == trial_store, "Trial",

ifelse(STORE_NBR == control_store, "Control", "Other stores"))

][, numberCustomers := mean(nCustomers), by = c("YEARMONTH", "Store_type")

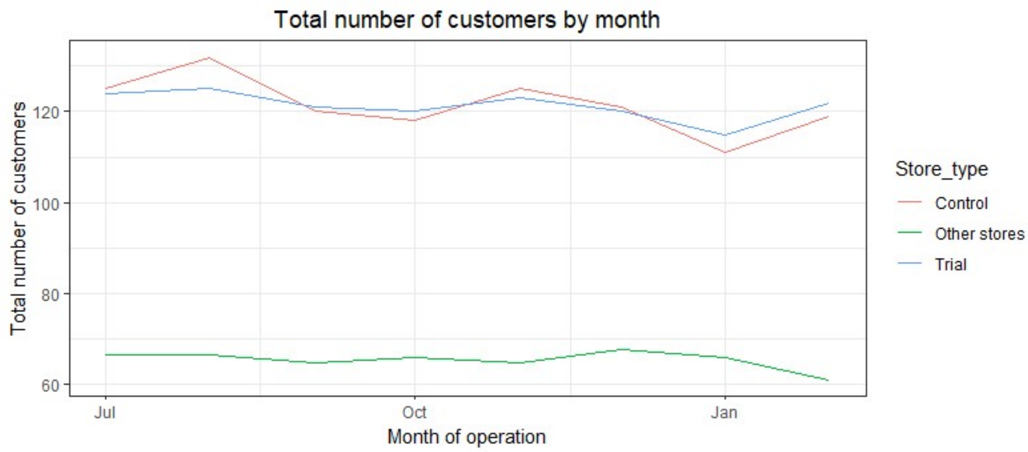
][, TransactionMonth := as.Date(paste(YEARMONTH %/%

100, YEARMONTH %% 100, 1, sep = "-"), "%Y-%m-%d")

][YEARMONTH < 201903 ,]

ggplot(pastCustomers, aes(TransactionMonth, numberCustomers, color = Store_type)) +

geom_line() + labs(x = "Month of operation", y = "Total number of customers", title = "Total number of customers by month")



Total number of customers of the control and trial stores are also similar.

Scale pre-trial control store sales to match pre-trial trial store sales

scalingFactorForControlSales <- preTrialMeasures[STORE_NBR == trial_store &

YEARMONTH < 201902, sum(totSales)]/preTrialMeasures[STORE_NBR ==

control_store & YEARMONTH < 201902,

sum(totSales)]

Apply the scaling factor

measureOverTimeSales <- measureOverTime

scaledControlSales <- measureOverTimeSales[STORE_NBR == control_store,][,controlSales := totSales * scalingFactorForControlSales]

Calculate the absolute percentage difference between scaled control sales and trial sales

percentageDiff <- merge(scaledControlSales[, c("YEARMONTH", "controlSales")],measureOverTime[STORE_NBR == trial_store, c("totSales", "YEARMONTH")],by = "YEARMONTH")[, percentageDiff := abs(controlSales-totSales)/controlSales]

As our null hypothesis is that the trial period is the same as the pre-trial period,

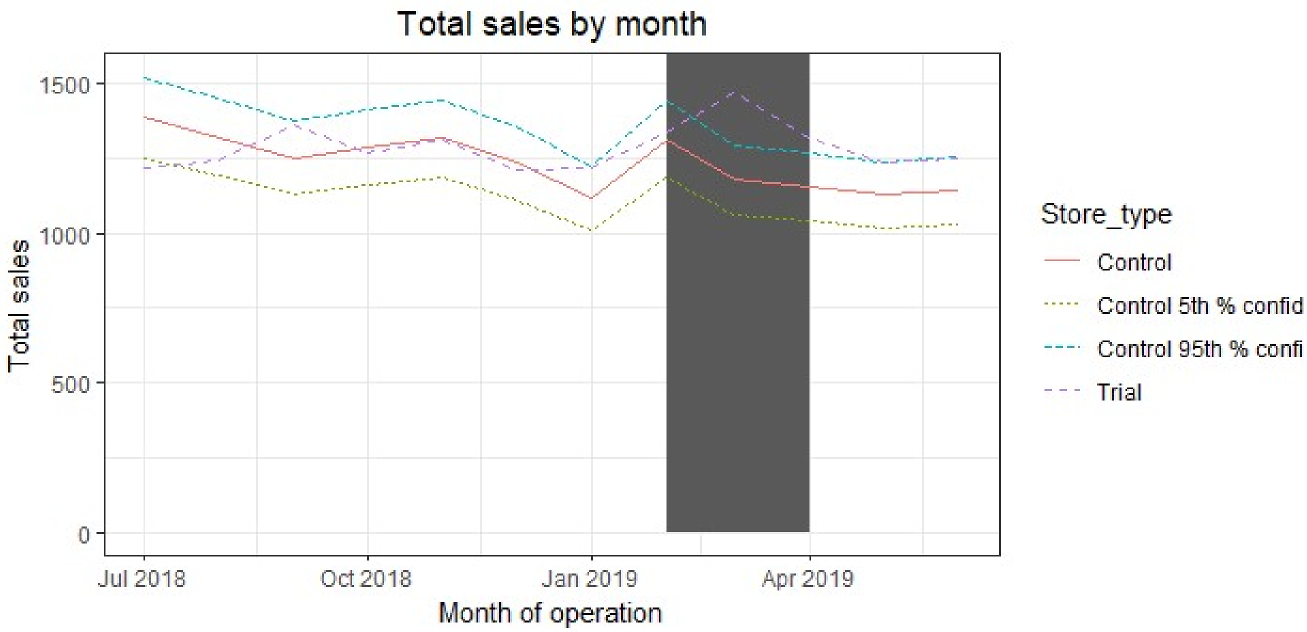
let's take the standard deviation based on the scaled percentage difference in the pre-trial period

stdDev <- sd(percentageDiff[YEARMONTH < 201902 , percentageDiff])

degreesOfFreedom <- 7

Trial and control store total sales

```
measureOverTimeSales <- measureOverTime

pastSales <- measureOverTimeSales[, Store_type := ifelse(STORE_NBR == trial_store, "Trial",

                                   ifelse(STORE_NBR == control_store, "Control", "Other stores"))

][, totSales := mean(totSales), by = c("YEARMONTH", "Store_type")

][, TransactionMonth := as.Date(paste(YEARMONTH %/%100, YEARMONTH %% 100, 1, sep = "-"),
"%Y-%m-%d")

][Store_type %in% c("Trial", "Control"), ]

#### Control store 95th percentile

pastSales_Controls95 <- pastSales[Store_type == "Control",

][, totSales := totSales * (1 + stdDev * 2)

][, Store_type := "Control 95th % confidence interval"]

#### Control store 5th percentile

pastSales_Controls5 <- pastSales[Store_type == "Control",

][, totSales := totSales * (1 - stdDev * 2)

][, Store_type := "Control 5th % confidence interval"]


## Combine the tables pastSales, pastSales_Controls95, pastSales_Controls5

trialAssessment <- rbind(pastSales, pastSales_Controls95, pastSales_Controls5)

#### Plotting a graph

ggplot(trialAssessment, aes(TransactionMonth, totSales, color = Store_type)) +

  geom_rect(data = trialAssessment[ YEARMONTH < 201905 & YEARMONTH > 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 store 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.

This would be a repeat of the steps before for total sales

Scale pre-trial control store customers to match pre-trial trial store customers

```
scalingFactorForControlCust <- preTrialMeasures[STORE_NBR == trial_store &

                                         YEARMONTH < 201902,
sum(nCustomers)]/preTrialMeasures[STORE_NBR ==

                                                                                    control_store & YEARMONTH < 201902,
sum(nCustomers)]
```

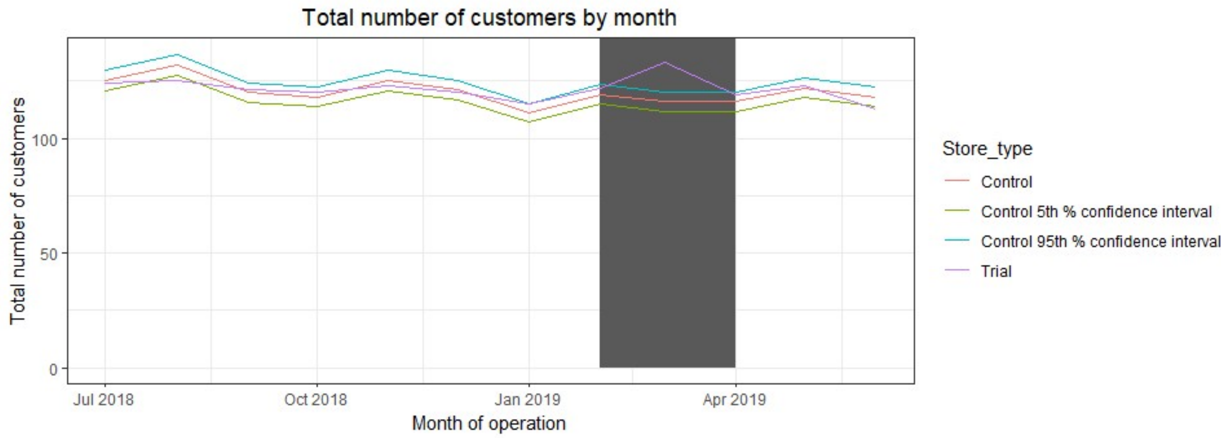
Apply the scaling factor

```
measureOverTimeCusts <- measureOverTime

scaledControlCustomers <- measureOverTimeCusts[STORE_NBR == control_store,

][ , controlCustomers := nCustomers * scalingFactorForControlCust

][, Store_type := ifelse(STORE_NBR == trial_store, "Trial",

                    ifelse(STORE_NBR == control_store,"Control", "Other stores"))

]
```

Calculate the absolute percentage difference between scaled control sales and trial sales

```
percentageDiff <- merge(scaledControlCustomers[,
c("YEARMONTH","controlCustomers")],measureOverTime[STORE_NBR == trial_store,
c("nCustomers", "YEARMONTH")],

                        by = "YEARMONTH")[, percentageDiff := abs(controlCustomers-
nCustomers)/controlCustomers]


#### 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[YEARMONTH < 201902 , percentageDiff])

degreesOfFreedom <- 7

# note that there are 8 months in the pre-trial period hence 8 - 1 = 7 degrees of freedom

#### Trial and control store number of customers

pastCustomers <- measureOverTimeCusts[, nCusts := mean(nCustomers), by = c("YEARMONTH",
"Store_type")

][Store_type %in% c("Trial", "Control"), ]


#### Control store 95th percentile

pastCustomers_Controls95 <- pastCustomers[Store_type == "Control",

][, nCusts := nCusts * (1 + stdDev * 2)

][, Store_type := "Control 95th % confidence interval"]

#### Control store 5th percentile

pastCustomers_Controls5 <- pastCustomers[Store_type == "Control",

][, nCusts := nCusts * (1 - stdDev * 2)

][, Store_type := "Control 5th % confidence interval"]

#### Combine the tables pastSales, pastSales_Controls95, pastSales_Controls5

trialAssessment <- rbind(pastCustomers, pastCustomers_Controls95,pastCustomers_Controls5)

#### Plotting a graph

ggplot(trialAssessment, aes(TransactionMonth, nCusts, color = Store_type)) +

  geom_rect(data = trialAssessment[ YEARMONTH < 201905 & YEARMONTH > 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 customers", title = "Total number of customers by month")



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

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.