Advertising Effects

Conor Dowd, Esha Banerjee, Jingying Bi, Kanyao Han, Minjia Zhu

January 22, 2018

```r
library(bit64)
library(data.table)
library(RcppRoll)
library(lfe)
library(stargazer)
library(knitr)
library(tidyverse)
```

Data preparation

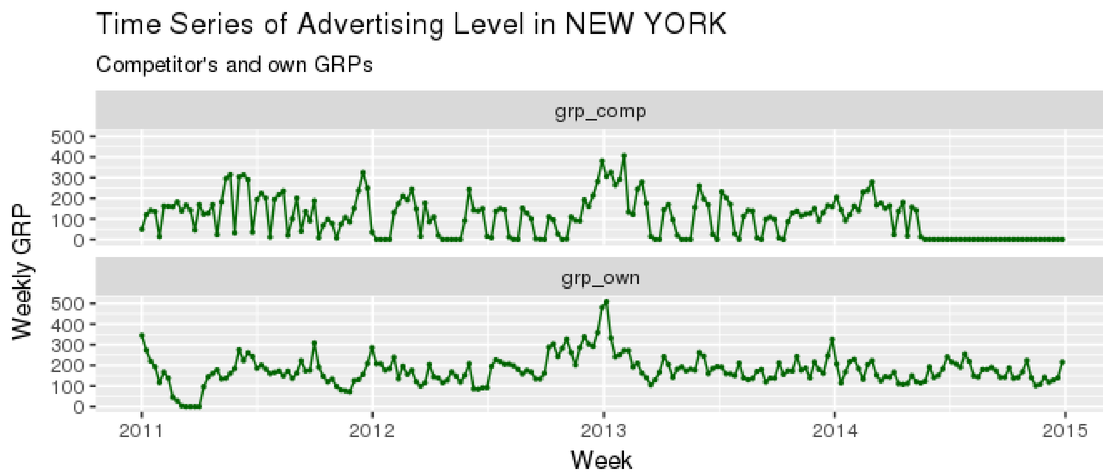
```r
load("/classes/37105/main/Assignment-4/Brands.RData")
load("/classes/37105/main/Assignment-4/Stores-DMA.RData")
load("/classes/37105/main/Assignment-4/move_8412.RData")
load("/classes/37105/main/Assignment-4/adv_8412.RData")


selected_module = 8412
selected_brand  = 621727

#Rename variables
setnames(move, old = c('units','promo_percentage'), new = c('quantity','promotion'))

# Create "brand_name"
move[, brand_name := if_else(brand_code_uc == selected_brand, "own", "comp")]

# Aggregate price, promotion and quantity for own and comp data at each store/week level
move = move[, .(quantity = sum(quantity), price = mean(price),
                promotion = mean(promotion)),
            keyby = .(store_code_uc, week_end, brand_name)]

# Extract the DMA and store code variables and retain only *unique* rows
stores_dma = unique(stores[, .(store_code_uc, dma_code)])

# Merge the `dma_code` with the movement data.
move = merge(move, stores_dma)

# “Fill the holes” in the data set by creating observations for all DMA/week combinations
brands    = unique(adv_DT$brand_code_uc)
dma_codes = unique(adv_DT$dma_code)
weeks     = seq(from = min(adv_DT$week_end), to = max(adv_DT$week_end), by = "week")

# Perform the cross join and set missing values to 0
setkey(adv_DT, brand_code_uc, dma_code, week_end)
adv_DT = adv_DT[CJ(brands,dma_codes,weeks)]
adv_DT[is.na(adv_DT)] = 0

# Create "brand_name"
adv_DT[, brand_name := if_else(brand_code_uc == selected_brand, "own", "comp")]

# Aggregate the data at the DMA/week level, seperately for 'own' and 'comp'
adv_DT = adv_DT[, .(grp_direct = sum(grp_direct), grp_indirect = sum(grp_indirect)),
                keyby = .(dma_code, week_end, brand_name)]
```

```r
#Sum direct and indirect grp as 'grp'
adv_DT[, grp := grp_direct + grp_indirect]


# Define the adstock parameters:  the number of lags and the carry-over factor delta
N_lags = 52
delta  = 0.9

# Calculate the geometric weights based on the carry-over factor
geom_weights = cumprod(c(1.0, rep(delta, times = N_lags)))
geom_weights = sort(geom_weights)

# Calculate the adstock variable using the `roll_sum` function in the `RcppRoll` package
setkey(adv_DT, brand_name, dma_code, week_end)
adv_DT[, adstock := roll_sum(log(1+grp), n = N_lags+1, weights = geom_weights,
                             normalize = FALSE, align = "right",  fill = NA),
        by = .(brand_name, dma_code)]

# Merge the advertising data with the scanner data based on brand name, DMA code, and week.
setkey(move, brand_name, dma_code, week_end)
move = merge(move, adv_DT)

# Reshape the data
move = dcast(move, dma_code + store_code_uc + week_end ~ brand_name,
             value.var = c("quantity", "price", "promotion", "adstock", "grp"))

# Remove missing values
move = move[complete.cases(move)]

#Create a time trend or index for each month/year combination in the data.
move[,`:=`(year = year(week_end), month = month(week_end))]
move[, month_index := 12 * (year - min(year)) + month]
```

Data inspection

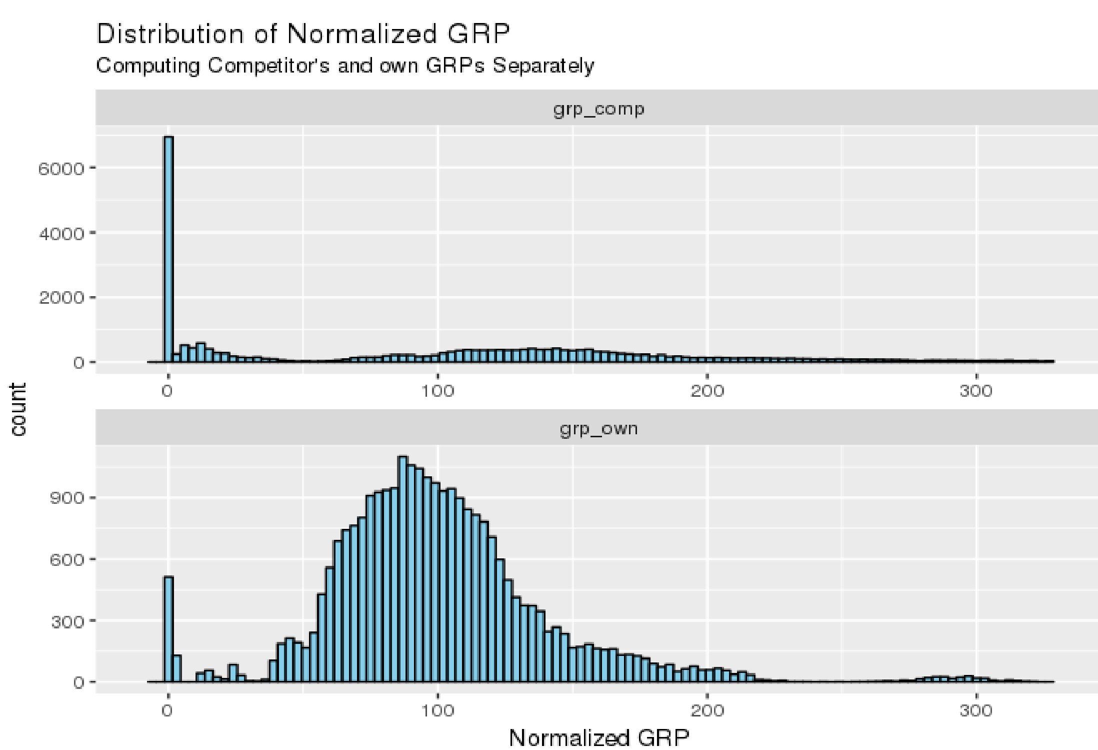
Time-series of advertising levels

```r
# pick "NEW YORK NY" which corresponds to dma_code = 501
# Plot the time-series of weekly GRP's

NY = move[dma_code == 501]

NY %>%
  group_by(week_end) %>%
  slice(1) %>%
  gather("grp_comp", "grp_own",
         key = "type", value = "value") %>%
  ggplot(aes(week_end, value)) +
  geom_point(size = 0.5, color = "dark green") +
  geom_line(color = 'dark green', size = 0.5) +
  facet_wrap(~ type, ncol = 1) +
  labs(title = "Time Series of Advertising Level in NEW YORK",
       subtitle = "Competitor's and own GRPs",
       x = 'Week',
       y = 'Weekly GRP')
```

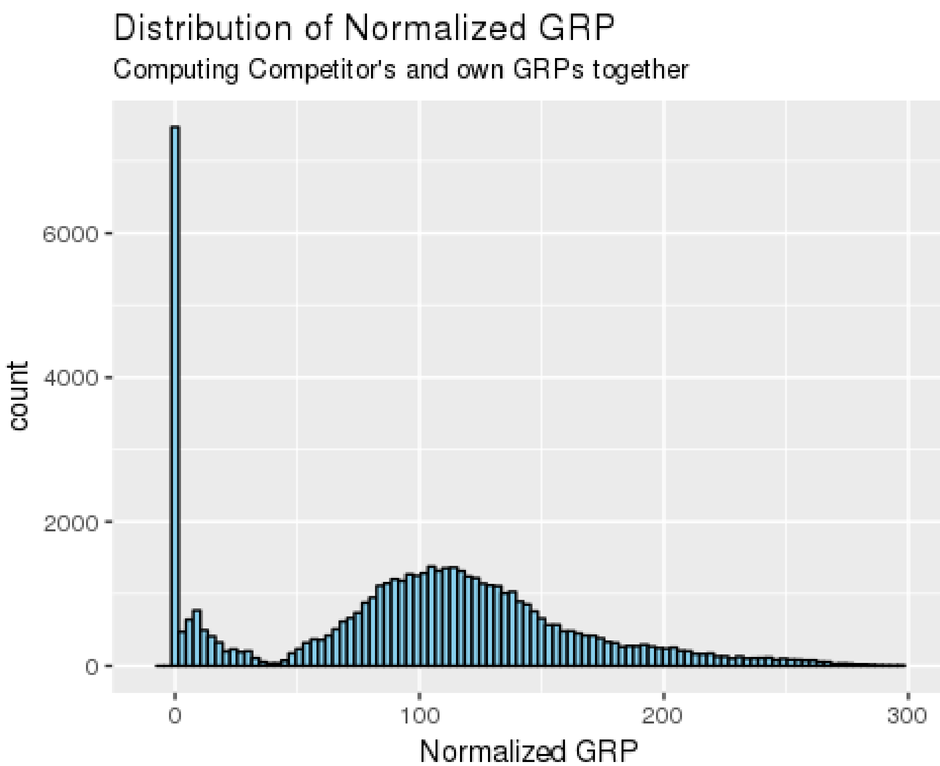


Overall advertising variation

```
move %>%
  group_by(week_end, dma_code) %>%
  slice(1) %>%
  gather("grp_comp", "grp_own",
         key = "type", value = "grp") %>%
  group_by(type, dma_code) %>%
  mutate(normalized_grp = 100 * grp / mean(grp)) %>%
  ggplot(aes(normalized_grp)) +
  geom_histogram(binwidth = 3, fill = 'sky blue', color = 'black') +
  facet_wrap(~ type, ncol = 1, scale = 'free') +
  xlim(-10, 330) +
  labs(title = "Distribution of Normalized GRP",
       subtitle = "Computing Competitor's and own GRPs Separately",
       x = "Normalized GRP")
```



```
move %>%
  group_by(week_end, dma_code) %>%
  slice(1) %>%
  gather("grp_comp", "grp_own",
         key = "type", value = "grp") %>%
  group_by(dma_code) %>%
  mutate(normalized_grp = 100 * grp / mean(grp)) %>%
  ggplot(aes(normalized_grp)) +
  geom_histogram(binwidth = 3, fill = 'sky blue', color = 'black') +
  xlim(-10, 300) +
  labs(title = "Distribution of Normalized GRP",
       subtitle = "Computing Competitor's and own GRPs together",
       x = "Normalized GRP")
```



Advertising effect estimation

```
fit_base = felm(log(1+quantity_own) ~ log(price_own) + log(price_comp) +
                  promotion_comp + promotion_own |store_code_uc + month_index,
                data = move)

fit_adstock = felm(log(1+quantity_own) ~ log(price_own) + log(price_comp) +
                     promotion_comp + promotion_own + adstock_comp +
                     adstock_own |store_code_uc + month_index,
                   data = move)

fit_notime = felm(log(1+quantity_own) ~ log(price_own) + log(price_comp) +
                    promotion_comp + promotion_own +
                    adstock_comp + adstock_own |store_code_uc,
                  data = move)

stargazer(fit_base, fit_adstock, fit_notime,
type = "text",
column.labels = c("Base", "Adstock", "Only Store FE"),
```

```r
dep.var.labels.include = FALSE, no.space = TRUE,
column.sep.width = "0.5pt",
font.size = "tiny", notes.align = "l")

## 
## ===================================================================================
##                                          Dependent variable:                      
##                        -------------------------------------------------------------
##                                Base                Adstock           Only Store FE  
##                                 (1)                  (2)                  (3)       
## -----------------------------------------------------------------------------------
## log(price_own)              -2.091***            -2.089***            -1.842***     
##                              (0.014)              (0.014)              (0.014)      
## log(price_comp)             -0.099***            -0.098***            -0.394***     
##                              (0.013)              (0.013)              (0.013)      
## promotion_comp               0.050***             0.051***             0.018***     
##                              (0.006)              (0.006)              (0.006)      
## promotion_own                0.913***             0.913***             0.981***     
##                              (0.003)              (0.003)              (0.003)      
## adstock_comp                                      0.002***             0.009***     
##                                                   (0.0003)            (0.00004)     
## adstock_own                                       0.010***            -0.003***     
##                                                   (0.001)              (0.0001)     
## -----------------------------------------------------------------------------------
## Observations                2,800,307            2,800,307            2,800,307     
## R2                            0.632                0.632                0.627       
## Adjusted R2                   0.630                0.631                0.625       
## Residual Std. Error 0.935 (df = 2786823) 0.935 (df = 2786821) 0.942 (df = 2786868)
## ===================================================================================
## Note:                  *p<0.1; **p<0.05; ***p<0.01

rm(fit_base, fit_adstock, fit_notime)
```

Estimation using border strategy

Merge border names

```r
stores[, border_name := as.factor(border_name)]

move = merge(move, stores[on_border == TRUE, .(store_code_uc, border_name)],
             allow.cartesian = TRUE)
```

```r
fit_interaction = felm(log(1+quantity_own) ~ log(price_own) +
                         log(price_comp) +
                         promotion_comp + promotion_own +
                         adstock_comp + adstock_own |store_code_uc +
                         border_name:month_index, data = move)

fit_clustered = felm(log(1+quantity_own) ~ log(price_own) +
                       log(price_comp) +
                       promotion_comp + promotion_own + adstock_comp +
                       adstock_own |border_name:month_index +
                       store_code_uc|0|dma_code, data = move)

stargazer(fit_interaction, fit_clustered,
type = "text",
column.labels = c("Interaction terms", "Clustered SE"),
dep.var.labels.include = FALSE, no.space = TRUE,
```

```
column.sep.width = "0.5pt",
font.size = "tiny", notes.align = "l")

##
## ==================================================================
##                                     Dependent variable:
##                                    ------------------------------
##                                    Interaction terms Clustered SE
##                                           (1)             (2)
## ------------------------------------------------------------------
## log(price_own)                          -2.051***      -2.051***
##                                         (0.020)         (0.142)
## log(price_comp)                         -0.242***      -0.242***
##                                         (0.017)         (0.074)
## promotion_comp                           0.021**        0.021
##                                         (0.008)         (0.015)
## promotion_own                           1.000***       1.000***
##                                         (0.004)         (0.038)
## adstock_comp                            0.004***       0.004***
##                                         (0.0001)       (0.0003)
## adstock_own                             0.007***       0.007***
##                                         (0.0002)       (0.0004)
## ------------------------------------------------------------------
## Observations                           1,714,507      1,714,507
## R2                                       0.605          0.605
## Adjusted R2                              0.603          0.603
## Residual Std. Error (df = 1708010)       0.972          0.972
## ==================================================================
## Note:                              *p<0.1; **p<0.05; ***p<0.01

rm(fit_interaction, fit_clustered)
```

Summarize and describe all your main estimation results.

The first model we estimated was a model regressing own and competitor's promotion and prices on the quantities, and store and month/year fixed effects were also controlled. In this model, both own and competitor prices are negatively related to sales, but the effect of the latter is relatively small. Besides, own promotions are shown to be incredibly important, while competitors promotions are also beneficial.

We then added own and competitor's adstock to the base model. It found that both own and competitor prices are negatively related to sales. The effect of the latter doubled in this model. Besides, while competitor adstocks benefitted a company, and its own adstock is detrimental.

This same regression was run again, but not controlled for time fixed effects. This time, the effect of competitor's prices further double, that of its adstock also quadruples and own adstock becomes negatively associated with sale quantities. However, the effect of own prices becomes moderately lower.

Moving into the more interesting results. Using a border estimation strategy, we look at the differences caused along advertising region borders. The hypothetical experiment is as follows -- pick two adjacent counties (they are likely very similar) -- then randomly choose one to recieve greater ad share than the other and look at purchasing patterns. Because we can't run this experiment, we instead look at adjacent counties which recieved (for somewhat exogenous reasons) different adstocks. By controlling for border and time fixed effects, we can try to focus on the differences caused by just the discontinuity across the border region. This estimation strategy found positive effects of own and competitor adstocks, as well as negative effects of own and competitor prices.

Finally, the most trustworthy of the routines we ran, is fundamentally the same as the above procedure. However, within an advertising area, the standard errors are clustered -- treating each DMA as a single observation with a bunch of correlated observed outcomes. This doesn't actually change the estimated values for the different covariates, but gives a better inferential procedure. Having done this, we conclude that there is a strong negative relationship between own promotion and sales, and no evidence of a relationship between competitor promotion and sales. Meanwhile, own adstock has a clear positive effect on sales, while competitor adstock has an effect of nearly half the size -- probably representing market expansion.

From the perspective of treating this as a causal estimate of the effect of ads, this is not the best we can do. First off, we should probably include an estimator looking at the difference between own price and competitor price, the lack of this kind of control is probably a fundamental failure. Moreover, while the adstock certainly looks to be beneficial, its possible that we are biasing our estimate by using a strange measure with bizzarre time series correlation. While clustering our SEs helps accomodate these concerns, we still may want to think harder about the exact functional form we are imposing on the adstock.

Beyond that, a major concern is that there are potentially very large spillovers between counties, and that the county threshold may not be as sharp as we would like. To the extent that a store on the edge of one county may attract many customers from the adjacent county, we have a bias issue. We would worry about this in the abscence of information about store locating decisions, however, we know for a fact that many stores deliberately locate near county borders in order to arbitrage differing county laws on things like alcohol sales -- so there may be an unusually large amount of crossover in this situation. Moreover, its not clear how confined to one DMA advertising really is. People moving back and forth could affect both purchasing behavior and ad exposure -- and thats only if the ads truly are limited in their range.

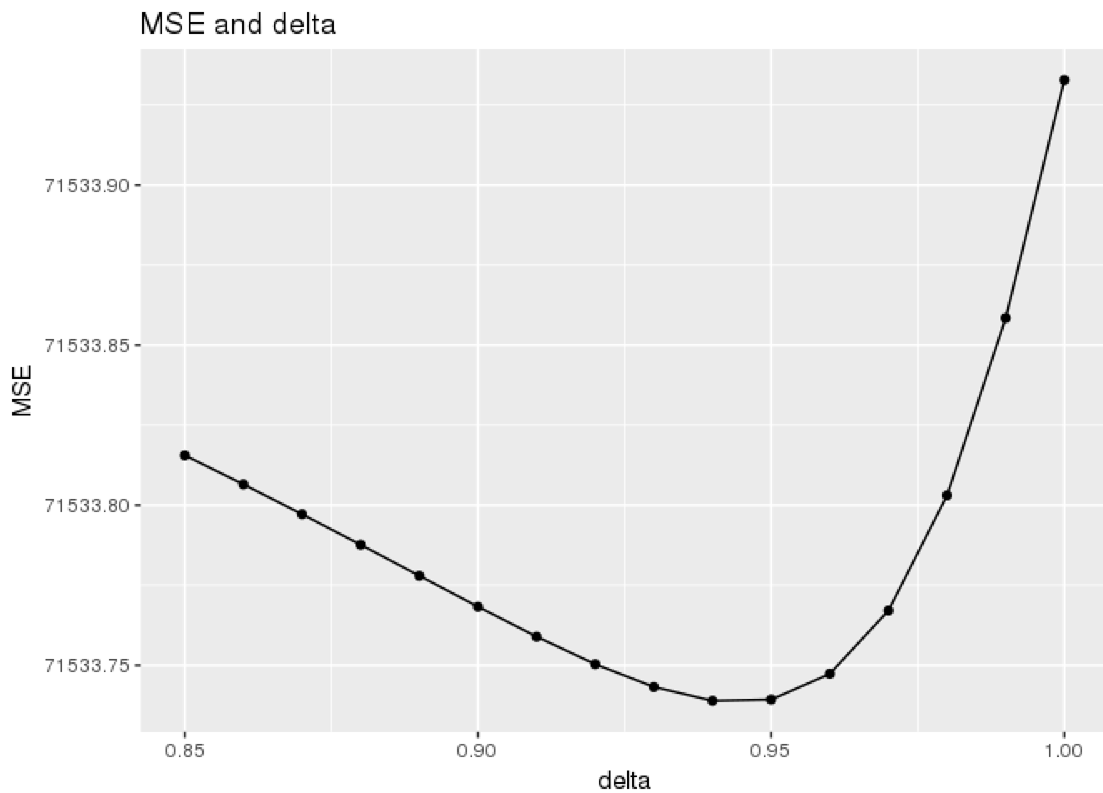
A different fundamental concern is that the DMAs are set up to be geographically contained. The LATE estimate provided by our border strategy -- if valid -- may still only apply to the borders between DMAs, which could be fundamentally different locations from the interiors of DMAs.

At the end of the day, this is why we A/B test things. Want to be sure that the effect is causal? Pick 4 totally separated DMAs, randomly allocate ads, and compare.

Optional extension: Estimate (calibrate) the carry-over parameter

```r
load("/classes/37105/main/Assignment-4/Brands.RData")
load("/classes/37105/main/Assignment-4/Stores-DMA.RData")
load("/classes/37105/main/Assignment-4/move_8412.RData")
load("/classes/37105/main/Assignment-4/adv_8412.RData")


selected_module = 8412
selected_brand  = 621727
setnames(move, old = c('units','promo_percentage'), new = c('quantity','promotion'))
move[, brand_name := if_else(brand_code_uc == selected_brand, "own", "comp")]
move = move[, .(quantity = sum(quantity), price = mean(price),
                promotion = mean(promotion)),
            keyby = .(store_code_uc, week_end, brand_name)]

stores_dma = unique(stores[, .(store_code_uc, dma_code)])

move1 = merge(move, stores_dma)

brands    = unique(adv_DT$brand_code_uc)
dma_codes = unique(adv_DT$dma_code)
weeks     = seq(from = min(adv_DT$week_end), to = max(adv_DT$week_end), by = "week")

setkey(adv_DT, brand_code_uc, dma_code, week_end)
adv_DT = adv_DT[CJ(brands,dma_codes,weeks)]
adv_DT[is.na(adv_DT)] = 0

adv_DT[, brand_name := if_else(brand_code_uc == selected_brand, "own", "comp")]

adv_DT = adv_DT[, .(grp_direct = sum(grp_direct), grp_indirect = sum(grp_indirect)),
                keyby = .(dma_code, week_end, brand_name)]
adv_DT1 = adv_DT[, grp := grp_direct + grp_indirect]


# Define the adstock parameters:  the number of lags and the carry-over factor delta
```

```r
N_lags = 52
delta  = (85 : 100) / 100
MSE = vector('double', length(delta))
for (i in 1: length(delta)){
  geom_weights = cumprod(c(1.0, rep(delta[i], times = N_lags)))
  geom_weights = sort(geom_weights)
  setkey(adv_DT1, brand_name, dma_code, week_end)
  adv_DT = adv_DT1[, adstock := roll_sum(log(1+grp), n = N_lags+1, weights =
geom_weights,normalize = FALSE, align = "right",  fill = NA),by = .(brand_name, dma_code)]

  setkey(move1, brand_name, dma_code, week_end)
  move = merge(move1, adv_DT)
  move = dcast(move, dma_code + store_code_uc + week_end ~ brand_name,
               value.var = c("quantity", "price", "promotion", "adstock", "grp"))
  move = move[complete.cases(move)]
  move[,`:=`(year = year(week_end), month = month(week_end))]
  move[, month_index := 12 * (year - min(year)) + month]
  stores[, border_name := as.factor(border_name)]
  move = merge(move, stores[on_border == TRUE, .(store_code_uc, border_name)],
               allow.cartesian = TRUE)

  fit = felm(log(1+quantity_own) ~ log(price_own) + log(price_comp) +
               promotion_comp + promotion_own + adstock_comp +
               adstock_own |border_name:month_index +
               store_code_uc|0|dma_code, data = move)

  MSE[i] = (sum((fit$fitted.values - move$quantity_own)^2)) / length(fit$fitted.values)
  rm(fit)
}

MSE <- cbind(delta, MSE) %>%
  as.data.table()

ggplot(MSE, aes(delta, MSE)) +
  geom_point() +
  geom_line() +
  labs(title = "MSE and delta")
```




```
kable(MSE[, rank := rank(MSE)])
```

delta	MSE	rank
0.85	71533.82	14
0.86	71533.81	13
0.87	71533.80	11
0.88	71533.79	10
0.89	71533.78	9
0.90	71533.77	8
0.91	71533.76	6
0.92	71533.75	5
0.93	71533.74	3
0.94	71533.74	1
0.95	71533.74	2
0.96	71533.75	4
0.97	71533.77	7
0.98	71533.80	12
0.99	71533.86	15
1.00	71533.93	16

When delta is around 0.94, The MSE are lowest, but the difference of MSE is not large when delta ranges from 0.85 to 1.