The Role of Competition in Explaining Law of One Price Deviations

Technical Note

Leandro Zipitria

February, 2018

Abstract

This note offer a detailed explanation of the construction of the database used in the paper "The Role of Competition in Explaining Law of One Price Deviations". It also offer the R scripts used to create the databases and run the regressions.

Introduction: databases and variables

This note provides a detailed exposition of the procedure to:

1- create the price differences databases,

2- calculate the distances between supermarkets,

3- create the dummy variable for local product.

We have two databases. The first one has information on the price, product, supermarket and date (price database). The supermarket and products are indexed. The second one, has information on the exact location of each supermarket -measured by the Universal Transverse Mercator, UTM- the number of cashiers, the city and the chain to which it belongs (supermarket database).

The variables in the price database are:

- Product: the number code for each product
- Year: the year of the observation
- Month: the month of the observation
- Day: the day of the observation
- Price: the price of the good for that month and year in the supermarket
- Super: the number code of supermarket

The database name is "base". The next code describes the calculation of the average, median, and mode monthly prices using the clean price database.

```
library("data.table") # requiered to load the database
library("dplyr")

# Load database
base <- fread("/location/to/database/PriceDatabase.csv", data.table = F))

# Calculate median, mode and average
setDT(base)[, median := median(Price), .(Year, Month, Super, Product)] #Calculate median
setDT(base)[, average := mean(Price), .(Year, Month, Super, Product)] # Calculate average
base <- base %>% group_by(Year, Month, Super, Product) %>%
  mutate(mode = as.numeric(names(sort(-table(Price)))[1L])) # Calculate mode
```

```
# Now we restrict the database to day one.
base <- base[base$Day == 1,] # restrict the database to day 1
```

Create local dummy

Now we create a dummy if, for each supermarket and month, the local dummy #X is present. X should be calculated for each local product in the database.

```
dbase$Price <- with(dbase, ave(Product, Year, Month, Super, FUN= function(x) X %in% x))
# Product X = local product in this example
```

Add supermarkets information

The supermarket information is in a separate file. We add supermarket information to the database.

```
# Load supermarket database
super <- read.csv("~/location/to/database/Supermarkets.csv", header = T, sep = ',')
dbase <- merge(dbase, supers, by = "Super") # Merge price and supermarket bases
```

The information available in the supermarket database is:

- Super: supermarket id
- chain: the chain to which each supermarket belongs (or not)
- city: the name of the city where the supermarkets is
- depto: the name of the state where the supermarket is
- X_UTM: Universal Transverse Mercator
- Y_UTM: Universal Transverse Mercator

Create price differences

The main point of the paper is to calculate price differences of goods. We have a database that contains information on price/supermarkets, and will create a different database that will have information on price differences. In what follows we will call the price/supermarkets database "base" and the price differences database "diff". The next code create an empty matrix called "dP" that will store the information on price differences, among other variables.

```
# First, we create a database that will hold the "diferential price" information

dP <- matrix(0,0,10)
colnames(dP) <- c("Year", "Month", "Product", "LocalP", "SameChain", "DifPrice",
                  "DifDepto", "DifCity", "DX_UTM", "DY_UTM")
```

In the next code, the database is transformed into a matrix. This make calculations more faster. But the key point is that columns have to be matched with variables. Take the case of the first part:

```
      l = t(outer(sub[,6], sub[,6], `+`)) # Add if local product = 1
```

This create a matrix of cross products for the variable in column 6. In this example, column 6 has the local dummy.

To create the price differences, we restrict for each Product, Year, and Month and perform the differences of the resulting sub matrix (called "sub") and store the information stacked

```
# Now create a table with price differences and then stack them and add to the file

for(i in unique(base$Product)) {
  for(j in unique(base$Year)) {
    for(h in unique(base$Month)) {
      sub <- subset(base, base$Product == i &
                    base$Year == j &
                    base$Month == h)
      as.matrix(sub)

      if (nrow(sub) < 2) {next}

# This code create vectors with the operations between observations
# Each number in "sub[,X]" must coincide with the column number (variable) in "base"
# "base" has the following variables, or column order: ["Product", "Year",
# "Month", "Local", "Chain", "Price", "State", "City", "X_UTM", "Y_UTM"].

      l = t(outer(sub[,4], sub[,4], `+`)) # Add if local product = 1
      diag(l) = NA # the information on the same observation is not needed (deleted)
      l[upper.tri(l)] <- NA # the matrix is symmetric, so one triangle is to be deleted (deleted)

      s = t(outer(sub[,5], sub[,5], "==")) # supermarket difference (same = 1)
      diag(s) = NA
      s[upper.tri(s)] <- NA

      p = t(outer(sub[,6], sub[,6], `-`)) # price difference
      diag(p) = NA
      p[upper.tri(p)] <- NA
      p = abs(p) # get absolute value of price differences

      d = t(outer(sub[,7], sub[,7], `!=`)) # State difference (dif = 1)
      diag(d) = NA
      d[upper.tri(d)] <- NA

      ci = t(outer(sub[,8], sub[,8], `!=`)) # City difference (dif = 1)
      diag(ci) = NA
      ci[upper.tri(ci)] <- NA

      xutm = t(outer(sub[,9], sub[,9], `-`)) # X_UTM differences
      diag(xutm) = NA
      xutm[upper.tri(xutm)] <- NA
      xutm = xutm * xutm # to calculate distances the square is needed

      yutm = t(outer(sub[,10], sub[,10], `-`)) # Y_UTM differences
      diag(yutm) = NA
      yutm[upper.tri(yutm)] <- NA
      yutm = yutm * yutm # to calculate distances the square is needed

# Now we need to create additional variables to stack the matrix created into columns.
# The next code create those variables (hY) and stack the information.
```

```
      h1= cbind(c(l))
      h2= cbind(c(s))
      h3= cbind(c(p))
      h4= cbind(c(d))
      h5= cbind(c(ci))
      h6= cbind(c(xutm))
      h7= cbind(c(yutm))

# Now a new variable (h8) is created that has has all the relevant information.
# Important: the order have to mach the names of the columns in dP.

      h8= cbind(j, h , i, h1, h2, h3, h4, h5, h6, h7)

# Now the rows are added to the empty matrix "diff".

      dP <- rbind(dP, h8)

# Now all NA are deleted (they were created to erase the diagonal and the triangles).

      dP <- na.omit(dP) # Delete prices with value NA

      }
    }
}

# Create a data frame
dfP <- as.data.frame(dP) # Transform the matrix into a data.frame
rm(dP)
```

Now the database has been created. Additional computations follows.

Calculate distance and other dummies

Now we calculate the distance between supermarkets. UTM easily allows to calculate distance as the Euclidean distance between two points (X, Y). In order to avoid negative numbers when distance is lower than 1 kilometer, we add 1. Also, as distance is in meters, we divide by 1000 to get distance in kilometers.

```
# Calculate distance
dfP$Distance <- log(1+(sqrt(dfP$DX_UTM + dfP$DY_UTM)/1000))
```

Now we calculate whether there is a local brand in one or both supermarkets.

```
# Dummy if both supermarkets sold a local brand
dfP$Both.local <- ifelse(dfP$LocalP == 2, 1, 0)

# Dummy if just one supermarket sold a local brand
dfP$One.local <- ifelse(dfP$LocalP == 1, 1, 0)
```

If any information is needed, please contact us at: leandro.zipitria@gmail.com