Introduction to Estimating Heterogeneous Individual Coefficients with Fixed Effect Controls

Install Packages

Uncomment the following block if packages are not installed.

```
# install.packages('devtools')
# library(devtools)
# install_github('georgegui/CommonFunctions_Public')
# install_github('georgegui/MarketingRegression')
```

```
library(data.table)
library(Matrix)
library(ggplot2)
library(CommonFunctions)
library(DemandEstimation)
```

Simulate Data

We generate the sales data based on a linear model. The default value for own elasticity, competing elasticities and promotion effect are -2, 0.2, 1 respectively, but they can be replaced with a random draw by providing the corresponding function. Missing products at a store or market can also be randomly generated, but for illustration purpose we make all products available at all stores.

```
set.seed(6)

dt <- GenerateEstimationTestData(
  n_zip3 = 2,
  n_brands = 5,
  n_store = 5,
  region_time_fe = TRUE,
  own_elas_draw = function() rnorm(1, -2),
  missing_assortment_draw = function() 0
  )
```

Elasticities for product 1

group_code is used to denote the product id. (This is probably a bad and confusing name originated from combining upc product group. I intended to change it to 'product' but there will be too many things that need to be changed accordingly.)

```
actual_elas <- unique(dt[ZIP3 == 1 & group_code == 1, .(
  actual_elasticity = own_elasticity, store_code_uc, group_code
  )])
actual_elas
```

```
##    actual_elasticity store_code_uc group_code
## 1:         -2.697340             1          1
## 2:         -2.102347             3          1
```

```
## 3:         -1.955564             4          1
## 4:         -2.154799             5          1
## 5:         -1.644488             2          1
```

Data Examples

```
head(dt[, .(week_end, ZIP3, store_code_uc, group_code, YearMonth, price)])
```

```
##      week_end ZIP3 store_code_uc group_code YearMonth       price
## 1: 2006-01-07    1             1          1   2006-01  0.26960598
## 2: 2006-01-07    1             3          4   2006-01 -0.62998541
## 3: 2006-01-07    1             5          3   2006-01  0.86865983
## 4: 2006-01-07    1             5          2   2006-01  1.72719552
## 5: 2006-01-07    2             4          3   2006-01  0.02418764
## 6: 2006-01-07    2             3          5   2006-01  0.36802518
```

Model Parameters

We generate a list of parameters that is necessary for estimation. This list of parameters is used throughout this project for testing different specifications. Not all parameters are used in the 'Estimate' function, as some are used for sample cleaning and selection.

```
cur_model <- data.table(
  model_name = 'SparseOLS',
  estimation_type = 'SparseOLS',
  time_control = 'YearMonth',
  regional_control = 'ZIP3',
  demean_before_estimation = 0
  )
cur_model <- as.list(CheckEstimationProfiles(
  cur_model, default_value_file = 'Default_Model_Values.csv'))

# class is importance because
head(cur_model)
```

Specify LHS Products and RHS Products

```
cur_model$lhs_groups <- '1'
cur_model$rhs_groups <- levels(dt$group_code)
```

Specify class of `model` as input to S3 function

For generating a flexible code structure, the function `Estimate` is a S3 function that will use different estimation method based on the class of its first parameter `model`. The potential class includes 'Sparse_OLS', 'SparseLASSO', 'OLS', 'Bayes'.

```
class(cur_model) <- c(cur_model$estimation_type, class(cur_model))
```

Estimation Result

The output of the result is a list of regional-level results, where each element contains the necessary information for formatting the results.

```
results <- Estimate(cur_model, dt)
```

Estimation result of the first ZIP3

The estimated coefficients is stored in the long format instead of the wide format because the assortment difference across stores and regions. The coefficient in the row with `variable` being 'price' and `group_code` being 2 is the effect of the price of product 2 on the sales of focal products.

```
result_region1 <- results[[1]]$out
head(result_region1, 6)
```

```
##    ZIP3 coefficient p_value column_index  variable group_code
## 1:    1    -2.69734       0            1     price          1
## 2:    1     0.20000       0            2     price          3
## 3:    1     0.20000       0            3     price          5
## 4:    1     0.20000       0            4     price          4
## 5:    1     0.20000       0            5     price          2
## 6:    1     1.00000       0            6 promotion          1
##    store_code_uc
## 1:             1
## 2:             1
## 3:             1
## 4:             1
## 5:             1
## 6:             1
```

```
own_price_coefficient <- result_region1[
  ZIP3 == 1 & variable == 'price' & group_code == '1', .(
    estimated_coef = coefficient, store_code_uc
    )]

own_price_coefficient <- merge(own_price_coefficient, actual_elas, by = 'store_code_uc')
own_price_coefficient
```

```
##    store_code_uc estimated_coef actual_elasticity group_code
## 1:             1      -2.697340         -2.697340          1
## 2:             2      -1.644488         -1.644488          1
## 3:             3      -2.102347         -2.102347          1
## 4:             4      -1.955564         -1.955564          1
## 5:             5      -2.154799         -2.154799          1
```

Comparison with wrong FWL usage

```
cur_model$model_name = 'OLS'
cur_model$estimation_type = 'OLS'
class(cur_model)     <- c(cur_model$estimation_type, class(cur_model))
# we demean the columns using the FWL theorem
dt <- DemeanCols(dt,
```

```
                relevant_cols = c('price', 'promotion', 'sales'),
                by_cols = c('YearMonth', 'ZIP3'))
result <- Estimate(cur_model, dt[ZIP3 == 1])

own_price_coefficient_fwl <- result[[1]]$out[
  rhs_var == 'price_1' , .(
    fwl_coef = coefficient, store_code_uc
  )]
```

```
own_price_coefficient <- merge(
  own_price_coefficient, own_price_coefficient_fwl,
  by = 'store_code_uc')

own_price_coefficient
```

```
##    store_code_uc estimated_coef actual_elasticity group_code  fwl_coef
## 1:             1      -2.697340         -2.697340          1 -2.573549
## 2:             2      -1.644488         -1.644488          1 -1.703334
## 3:             3      -2.102347         -2.102347          1 -2.131894
## 4:             4      -1.955564         -1.955564          1 -2.017112
## 5:             5      -2.154799         -2.154799          1 -2.164236
```

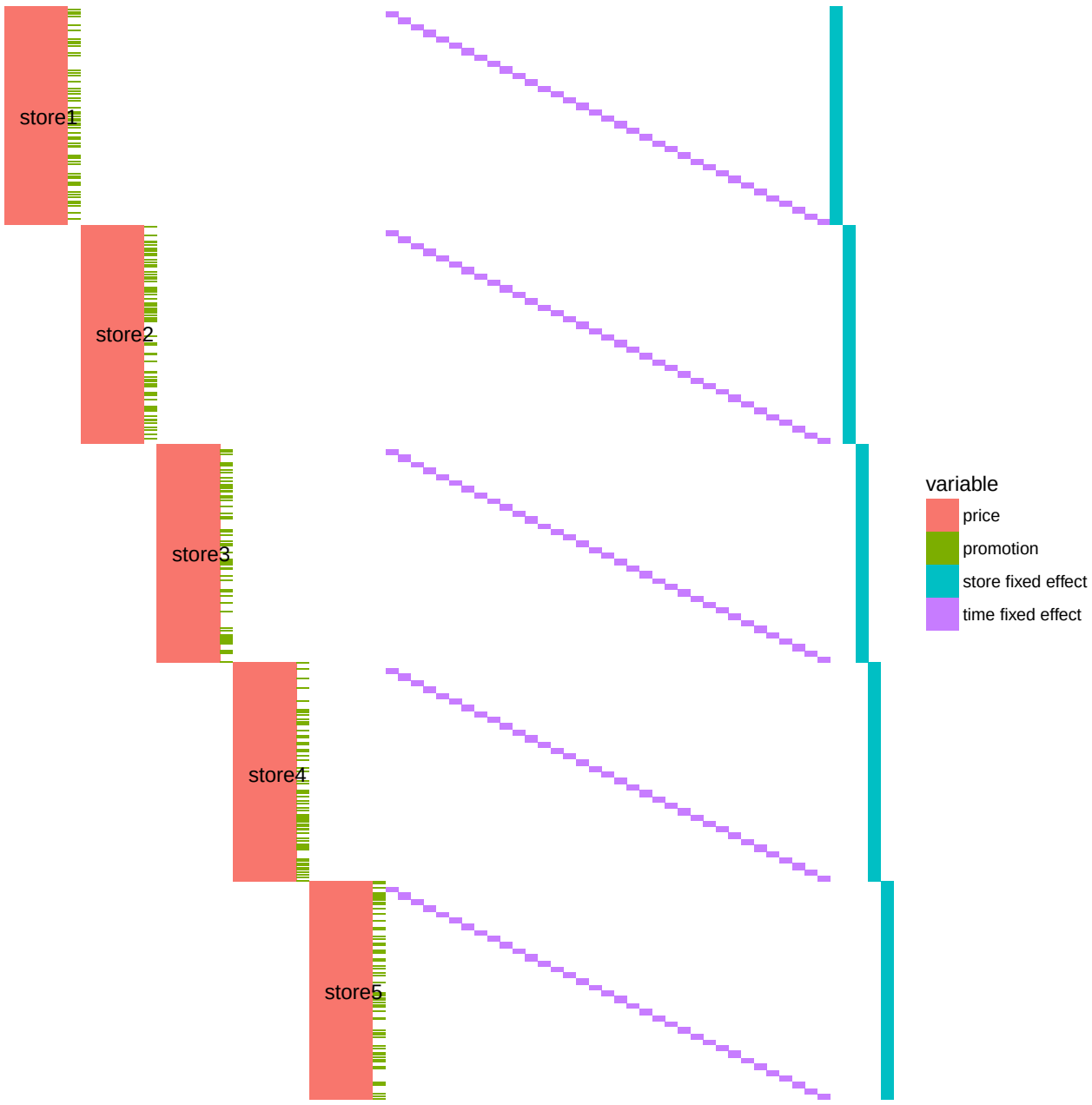
What does the regression matrix looks like?

`ConstructSparseRhs` is a function that I designed to construct the regression matrix. It is used in the `Estimate` function.

```
# if an observation is from a certain week_end and store_code_uc, which row it should be in.

small_dt <-  GenerateEstimationTestData(
  n_zip3 = 1,
  n_brands = 5,
  n_store = 5,
  region_time_fe = TRUE,
  own_elas_draw = function() rnorm(1, -2),
  missing_assortment_draw = function() 0
  )
small_dt <- small_dt[year(week_end) %between% c(2008, 2010)]
row_index <- small_dt[, .(row_index = as.integer(.GRP)),
                keyby = .(store_code_uc, week_end)]

rhs_sparse_matrix <- ConstructSparseRhs(
  cur_model,            # the model parameters
  small_dt,        # the price data.table
  '1',                  # the lhs product
  row_index)

column_definitions <- attr(rhs_sparse_matrix, 'relevant_cols')
setnames(column_definitions, 'store_code_uc', 'store_fixed_effect')

sparse_index <- data.table(which(rhs_sparse_matrix != 0, arr.ind = T))

# add column and row information
```

```r
setnames(sparse_index, c('V1', 'V2'), c('row_index', 'column_index'))
sparse_index <- merge(sparse_index, row_index, by = 'row_index')
sparse_index <- merge(sparse_index, column_definitions, by = 'column_index', all.x = TRUE)
n_price_cols <- nrow(column_definitions)
n_month <- attr(rhs_sparse_matrix, 'n_month_cols')
sparse_index[is.na(variable), variable := 'store fixed effect']
sparse_index[column_index %between% c(n_price_cols + 1, n_price_cols + n_month),
             variable := 'time fixed effect']

#
store_label_text <- sparse_index[, .(
  x = median(as.numeric(column_index)),
  y = -median(as.numeric(row_index))
  ), by = .(store_code_uc = paste0('store', store_code_uc))]

gg <- ggplot() +
  geom_tile(data = sparse_index, aes(x = column_index, y = -row_index, fill = variable)) +
  geom_text(data = store_label_text, aes(x = x, y = y, label = store_code_uc)) +
  theme_void() + labs(x = "", y = "") +
  scale_x_discrete(expand = c(0, 0)) +
  scale_y_discrete(expand = c(0, 0))
ggsave('matrix_example.pdf', gg, height = 8, width = 8)
```

```r
gg
```


store1
store2
store3
store4
store5
variable
price
promotion
store fixed effect
time fixed effect