Analysis of Household Buying Behavior: Data Preparation

Günter J. Hitsch

1/4/2017

Load the data

Note: Here, the data are located in the subfolder *Data* of the current working directory. `paste0` concatenates strings.

```
library(bit64)
library(data.table)

data_folder    = "./Data"
purchases_file = "purchases.RData"
products_file  = "Products.RData"

load(paste0(data_folder, "/", purchases_file))
load(paste0(data_folder, "/", products_file))
```

Extract all beverage purchase data (not including alcohol) and the corresponding product attributes

First, we merge the product group code of each product = `upc`/`upc_ver_uc` combination with the purchase data.

```
setkey(purchases, upc, upc_ver_uc)
setkey(products, upc, upc_ver_uc)

purchases = merge(purchases, products[, .(upc, upc_ver_uc, product_group_code)])
```

Note that the merge statement above works even without the initial `setkey` statements because `purchases` is already keyed on `household_code`, `purchase_date`, `upc`, `upc_ver_uc`, and `products` is keyed on `upc`, `upc_ver_uc`. Hence, the merge will be on `upc`, `upc_ver_uc`, the combination that defines a unique product, as desired. However, there's nothing wrong with writing code that is explicit about our intentions.

We only need data for the following product groups:

```
507   -  JUICE, DRINKS - CANNED, BOTTLED
1503  -  CARBONATED BEVERAGES
1508  -  SOFT DRINKS-NON-CARBONATED
2006  -  JUICES, DRINKS-FROZEN
```

Extract the purchase and product data:

```
selected_groups = c(507, 1503, 1508, 2006)

purchases = purchases[product_group_code %in% selected_groups]
products  = products[product_group_code %in% selected_groups]
```

Data cleaning

To clean the purchase data, we first remove the added attribute, `product_group_code`.

```
purchases[, product_group_code := NULL]
```

Then we convert the date string to an R *Date* object:

```
purchases[, purchase_date := as.Date(purchase_date)]
```

Note that converting string (character) data to a Date object can take a *very long* time, hence it's advisable to do this in an initial data-preparation step.

Now let's perform a quick check to see if the purchase and product tables contain any missing values (`NA`'s):

```
anyNA(purchases)
```

```
[1] FALSE
```

```
anyNA(products)
```

```
[1] TRUE
```

There are no missing values in `purchases`, but `products` contains missing values. Let's find the rows with missing values using `complete_cases`:

```
not_missing = complete.cases(products)
table(not_missing)
```

```
not_missing
 FALSE   TRUE 
     1 109068 
```

There is one missing value only. This happens all the time—all real-world data have imperfections. Let's remove this one row:

```
products = products[not_missing]
```

Sometimes, the brand description field changes, even though `brand_descr` refers to the same brand. Here we will fix two such instances manually:

```
products[brand_descr == "COCA-COLA CLASSIC R", brand_descr := "COCA-COLA R"]
products[brand_descr == "MTN DEW R", brand_descr := "MOUNTAIN DEW R"]
```

Save data

Key the data along household code, date, and product (`upc`/`upc_ver_uc`):

```
setkey(purchases, household_code, purchase_date, upc, upc_ver_uc)
```

And finally save the data:

```
purchases_output_file = "purchases_beverages.RData"
products_output_file  = "products_beverages.RData"

save(purchases, file = paste0(data_folder, "/", purchases_output_file))
save(products, file = paste0(data_folder, "/", products_output_file))
```