Analysis of Household Buying Behavior: Carbonated Soft Drinks and Other Beverages

Drew Ficken / Jace Herrmann / Krishna Kadiyala / Nuraddin Samadzade / Nao Azuma

March 29, 2019

Contents

1 Overview

We will perform an analysis of category buying and consumption behavior of beverages (not including alcohol or dairy), with a particular emphasis on sodas (carbonated soft drinks/CSD's). Sodas have been the subject of an intense debate linked to public health concerns, and one or two cent-per-ounce taxes have recently been passed in San Francisco, Cook County (although the tax was repealed in 2017), Philadelphia, Boulder, and other cities. Changes in buying behavior due to health concerns present challenges to the beverage manufacturers, but also opportunities if consumers are shifting their consumption to healthier substitutes.

2 Data

The Nielsen Homescan panel data set is an ideal data source to study broad trends in consumption behavior. This data set is available for research and teaching purposes from the Kilts Center for Marketing at Chicago Booth.

We will use Homescan panel data from 2004 to 2014. In many years we have information on the buying behavior of more than sixty thousand households. The corresponding full data set is large. Hence, to avoid memory and computing time issues, I extracted the beverage data that we will use for the analysis. I also took a *25 percent random subsample* of all the original obervations. Once you load the data, you can verify that even this subsample of the beverage data contains more than 10 million observations (use `nrow`, `ncol`, or `dim` to see the size of a data.table or data frame).

Although not necessary for this assignment, it is good to know how to create random subsamples. For more information, please consult the Appendix below.

The key data for the analysis are contained in a **purchase file** and a **product file**. Load the data:

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

#data_folder    = "./Data"
purchases_file = "purchases_beverages.RData"
products_file  = "products_beverages.RData"

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

Note that the data are in a sub-folder of *this* R Markdown document called `Data`, i.e. they are *assumed* to be in that folder! The `paste0` command merges strings. The bit64 package is used because the product UPC numbers are 64-bit (long) integers, a data type that is not part of base R.

3 Variable description

The **`purchases` data.table** contains information on the products bought by the households.

- **Inspect the data**

Households are identified based on a unique `household_code`. For each shopping trip we know the `purchase_date` and the `retailer_code` (for confidentiality, the Kilts Center data do not include the exact name of the retailer).

For each product we have information on the `total_price_paid` and the `quantity` purchased. A deal flag (0/1) and a `coupon_value` is also provided. The `total_price_paid` applies to *all* units purchased. Hence, if `total_price_paid = 7.98` and `quantity = 2`, then the *per-unit price* is $7.98/2 = 3.99$ dollars. Furthermore, if the `coupon_value` is positive, then the total dollar amount that the household spent is `total_price_paid - coupon_value`. The *per-unit cost* to the household is then even lower. For example, if `coupon_value = 3` in the example above, then the per-unit cost to the household is (7.98 - 3)/2 = 2.49 dollars.

I recommend to use the per-unit cost if the objective is to measure household dollar spending per unit purchased. If the objective is to measure the shelf-price of the product, use the per-unit price instead.

Important data notes

1. Products in the Nielsen Homescan and RMS scanner data are identified by a unique combination of `upc` and `upc_ver_uc`. Why not just the UPC code? — Because UPC's can change over time, for example if a UPC is assigned to a different product. The UPC version code captures such a change. From now on, whenever we refer to a *product*, we mean a `upc`/`upc_ver_uc` combination. To identify a unique product in data.table, use a `by = .(upc, upc_ver_uc)` statement.
2. The `panel_year` variable in `purchases` is intended *only* to link households to the corresponding household attribute data (income, age, etc.), which are updated yearly. At the beginning of a `panel_year` it need not exactly correspond to the calendar year. We will work with the household attribute data in one of the next assignments.

The **`products`** data.table contains product information for each `upc`/`upc_ver_uc` combination.

- **Inspect the data.table**

Note that products are organized into departments (e.g. DRY GROCERY), product groups (e.g. CARBONATED BEVERAGES), and product modules (e.g. SOFT DRINKS - CARBONATED), with corresponding codes and descriptions. Use `unique` or `table` to print all values for the variables. Or create a table that lists all the product module codes, product module descriptions, and group descriptions in the data:

```
module_DT = products[, head(.SD, 1), by = product_module_code,
                        .SDcols = c("product_module_descr", "product_group_descr")]
module_DT = module_DT[order(product_group_descr)]
```

We also have brand codes and descriptions, such as PEPSI R (Pepsi regular). You will often seen the brand description CTL BR, which stands for *control brand*, i.e. private label brand. Brands are identified using either the `brand_code_uc` or `brand_descr` variables, and are sold as different products (`upc`/`upc_ver_uc`) that differ along size, form (e.g. bottles vs. cans), or flavor.

`multi` indicates the number of units in a multi-pack (a multi-pack is a pack size such that `multi > 1`). More on the amount and unit variables below.

For many more details on the data, consult the *Consumer Panel Dataset Manual* (on Canvas).

4 Prepare the data for the analysis

We will calculate yearly summary statistics of customer buying behavior. To calculate the year corresponding to a purchase date, use `year()` in the data.table IDateTime class (see `?IDateTime` to learn about other conversion functions). Note that there are other methods for time aggregation that we will study later in the course.

```
purchases[, year := year(purchase_date)]
```

Note that we create this year-variable because the `panel_year` variable in `purchases` does not exactly correspond to the calendar year, as we already discussed above. You can verify that there is a tiny percentage of observations for the 2003 calender year, that "slipped" into the data set because the purchases are recorded for households in the 2004 `panel_year`.

- **Remove the 2003 observations**

```
purchases<-purchases[-which(year == 2003),]
```

4.1 Define categories

In the analysis we want to distinguish between carbonated soft drinks (CSD's), diet (low-calorie) CSD's, bottled water, and a category including all other beverages (juices, ...). Hence, we create a new `category` variable that allows us classify the beverage purchase observations.

- **First, in the `products` table, create a default `category` variable with a name such as "Other". Then find the product module codes for the three relevant categories and assign a corresponding name to `category`.**

```
products[, category := "Other"]

### bottled water - product module code = 1487
products$category[which(products$product_module_code==1487)] = "Water"

### CSD - product module code = 1484
products$category[which(products$product_module_code==1484)] = "CSD"

### Diet CSD - product module code = 1553
products$category[which(products$product_module_code==1553)] = "Diet CSD"
```

- **Document the number of observations, i.e. the number of products that belong to each of the categories.**

```
table(products$category)
```

```
     CSD Diet CSD    Other    Water
   24126    10267    58754    15921
```

Now merge the category variable with the purchase data.

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

Note that `by = .(upc, upc_ver_uc, ...)` provides the product-level link between the products and the purchase table.

4.2 Volume in equivalent units

To measure volume in *equivalent units*, we need the product-level information on the *units of measurement* of product volume (`size1_units`), such as ounces, and the corresponding volume in a pack size `(size1_amount1). This information neeeds to be merged with the purchase data. We also merge with`multi', which indicates multi-pack sizes.

- **Perform this merge**

```
purchases = merge(purchases, products[, .(upc, upc_ver_uc, size1_units,size1_amount, multi)])
```

For beverages, product volume is typically measured in ounces (`OZ`), less frequently in quarts (`QT`), and only rarely in counts (`CT`).

- **Document the number of observations by unit of measurement. Let's ignore counts, and remove all corresponding data from the purchases data.table. Then convert the `quantity` of units purchased into a common `volume` measure in gallons, using the `size1_amount`**

variable. Also incorporate the `multi` variable into the volume calculation—`multi` accounts for multi-packs.

```
table(purchases$size1_units)
```

```
     CT      OZ      QT
 126185 9666093  425636
```

```
purchases = purchases[-which(size1_units=="CT"),]

### quantity * multi * size1_amount = # of unit measurements
### OZ = quantity/128
### QT = quantity/4

# new_quantity=rep(0,nrow(purchases))

# for (i in 1:nrow(purchases)){
#   if(purchases$size1_units[i]=="OZ"){
#     new_quantity[i]=purchases$multi[i]*purchases$quantity[i]*purchases$size1_amount[i]/128
#   }else{ # size1_units = "QT"
#     new_quantity[i]=purchases$multi[i]*purchases$quantity[i]*purchases$size1_amount[i]/4
#   }
# }
#

#purchases[, volume := new_quantity]

purchases[,volume := ifelse(size1_units=="OZ",multi*quantity*size1_amount/128,multi*quantity*size1_amou
```

4.3 Number of households in the data

To calculate the number of households in the data by year, use:

```
purchases[, no_households := length(unique(household_code)), by = year]
```

- **Create and show a table with the number of households by year**

```
year_table =unique(purchases[, .(year,no_households)])

year_table = year_table[order(year)]

year_table
```

```
    year no_households
 1: 2004         10566
 2: 2005          9700
 3: 2006         10565
 4: 2007         16330
 5: 2008         16186
 6: 2009         16077
 7: 2010         16322
 8: 2011         15366
 9: 2012         15395
10: 2013         15764
11: 2014         15284
```

Note the expansion in the number of Homescan panelists in 2007!

5 Category-level analysis

Now we are ready to analyse the evolution of purchases and consumption in the four product categories. We want to calculate total and per capita (more precisely: per household) consumption metrics. First, we create the total dollar spend and the total purchase volume for each category/year combination. We use the data.table approach for aggregation:

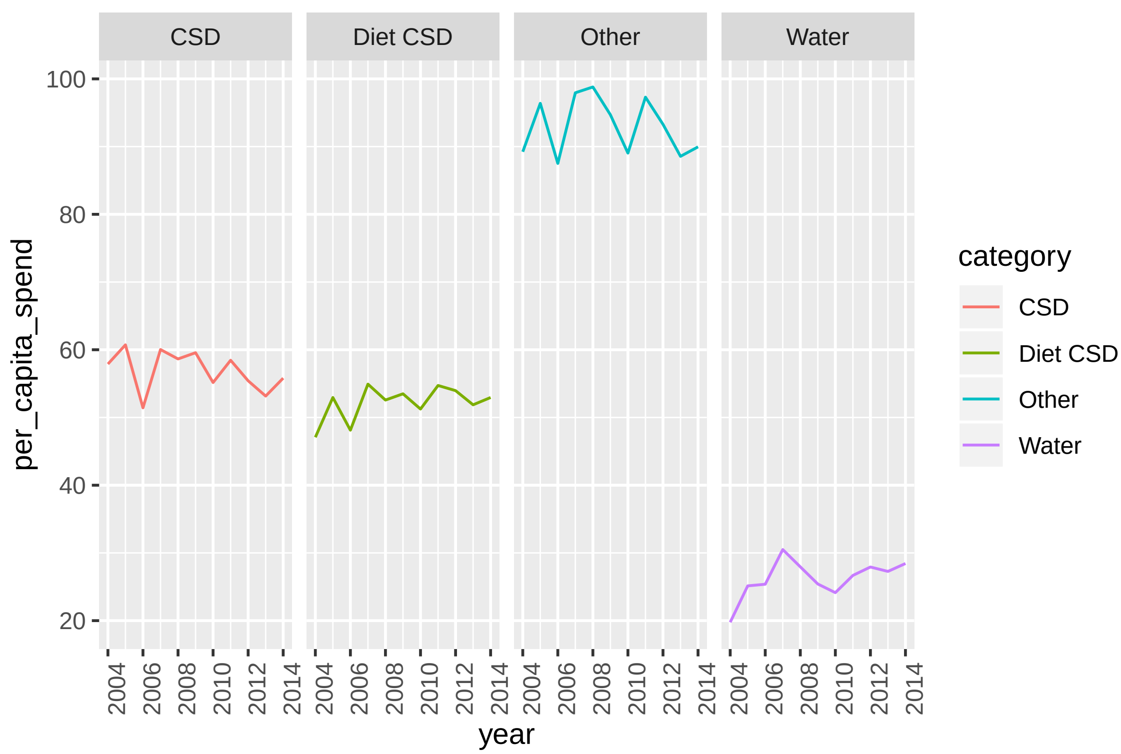
```
purchases_category = purchases[,
   .(spend           = sum(total_price_paid - coupon_value),
     purchase_volume = sum(volume),
     no_households   = head(no_households, 1)),
   keyby = .(category, year)]
```

- **Calculate per capita spend and purchase volume (in gallons) for each category separately. Then graph the evolution of the yearly per capita purchase volume for all four categories.**

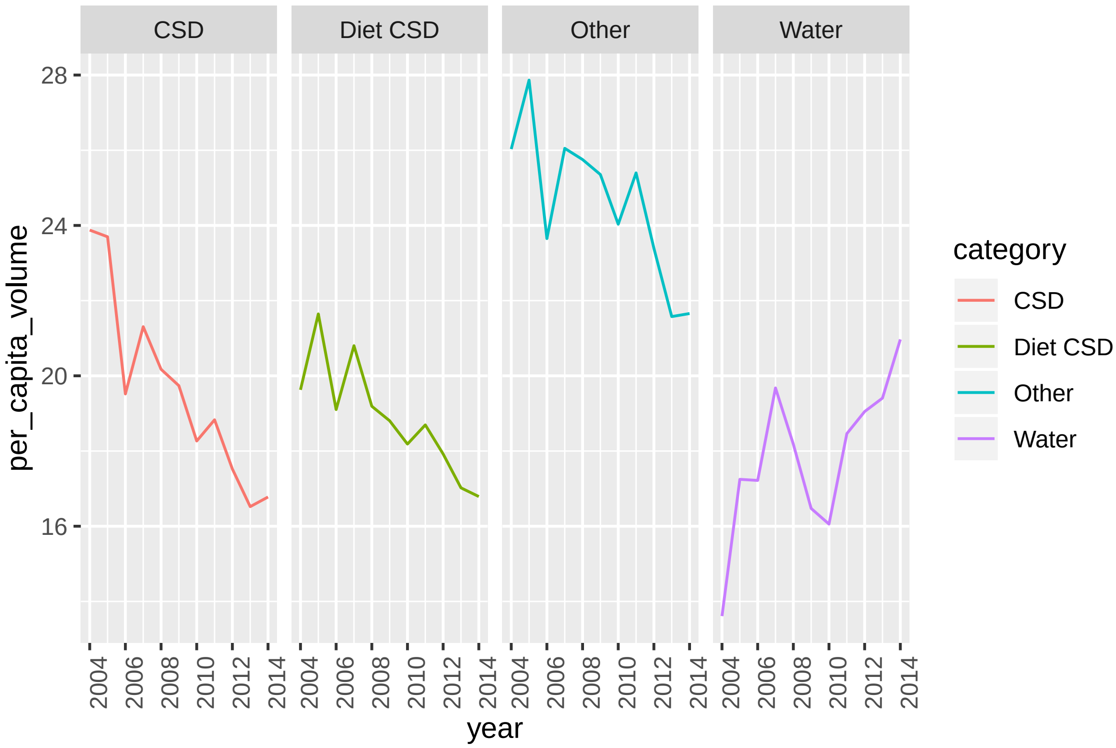
```
purchases_category = purchases_category[, per_capita_spend := (spend/no_households)]

purchases_category = purchases_category[, per_capita_volume :=(purchase_volume/no_households)]
```

Note: Instead of creating graphs for each of the four categories you can use a `facet_wrap` layer provided by ggplot2.

```
ggplot(purchases_category, aes(x=year, y=per_capita_spend, group = category, colour = category)) +
  geom_line() + facet_wrap(~ category, ncol=4) + theme(axis.text.x = element_text(angle = 90, hjust = 1
```

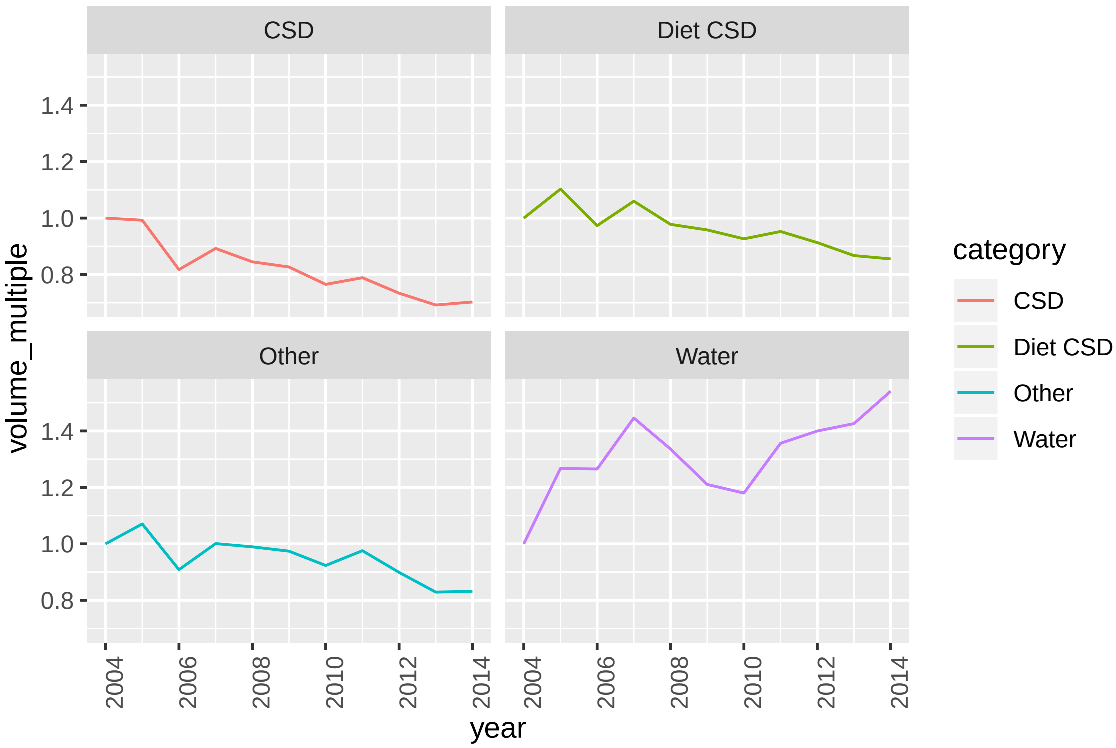



```
ggplot(purchases_category, aes(x=year, y=per_capita_volume, group = category, colour = category)) +
  geom_line() + facet_wrap(~ category, ncol=4) + theme(axis.text.x = element_text(angle = 90, hjust = 1
```



- **Express the purchase/consumption data as multiples of the 2004 values, such that per capita volume takes the value of 1.0 in all categories in 2004. Such a normalization allows us to compare the consumption series in each category directly in percentage terms. Then show the graphs of consumption (normalized to its 2004 value), and discuss the results.**

```
dividend = purchases_category[which(purchases_category$year==2004),.(category, per_capita_volume)]

colnames(dividend)<-c("category","dividend")

purchases_category = merge(purchases_category, dividend, by = "category")

purchases_category = purchases_category[, volume_multiple := (per_capita_volume/dividend)]

ggplot(purchases_category, aes(x=year, y=volume_multiple, group = category, colour = category)) +
  geom_line() + facet_wrap(~ category) + theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



The normalized per-capita-volume cosnumptions shows that over the decade 2004-2014:

CSD consumption reduced by almost 30% Diet CSD consumption reduced by 15% Other Drinks consumption reduced by 18-19% Bottled Water consumption increased by more than 50%

6 Brand-level analysis

Now we investigate the evolution of consumption for some of the key brands in the soda and bottled water categories.

- **First, merge the brand identifier `brand_descr` with the purchase data**

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

Then we rank brands by total dollar spend in each category separately. We can assign ranks either using the `rank` function in base R, or using `frankv` (or `frank`) in the data.table package. Simple usage: To rank a vector `x` in ascending order, use `frankv(x)`. To rank in descending order, use `frankv(x, order = -1)`. See `?frank` for more options.

First calculate total dollar spend by each category/brand combination, then assign the rank according to total spend:

```
brand_summary = purchases[, .(spend = sum(total_price_paid - coupon_value)),
                          by = .(category, brand_descr)]
brand_summary[, rank := frankv(spend, order = -1), by = category]

brand_summary = brand_summary[order(rank),]
```

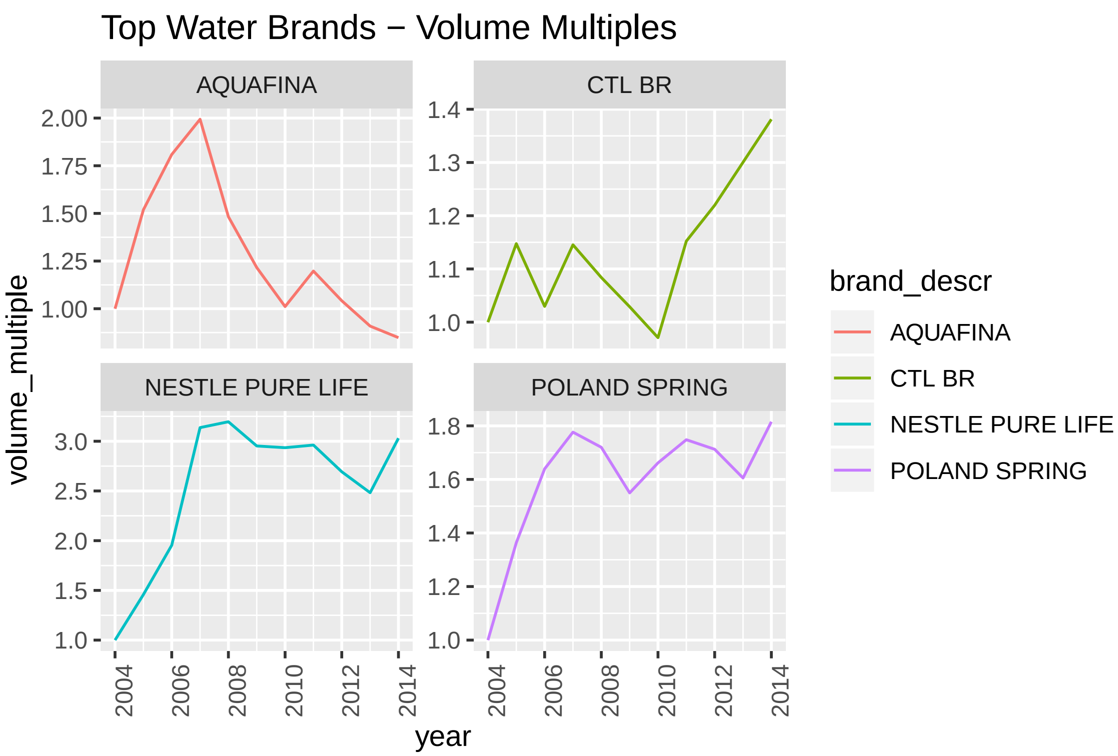
- **Merge the brand ranks in the `brand_summary` table with the `purchases` information. Aggregate to the brand level, as we did before at the category level. Then calculate per capita spending and volume, and normalize the per capita variables to 1.0 in 2004, as**

before at the category level. Plot the evolution of brand volume for the top four brands, separately for the CSD, diet CSD, and bottled water categories.

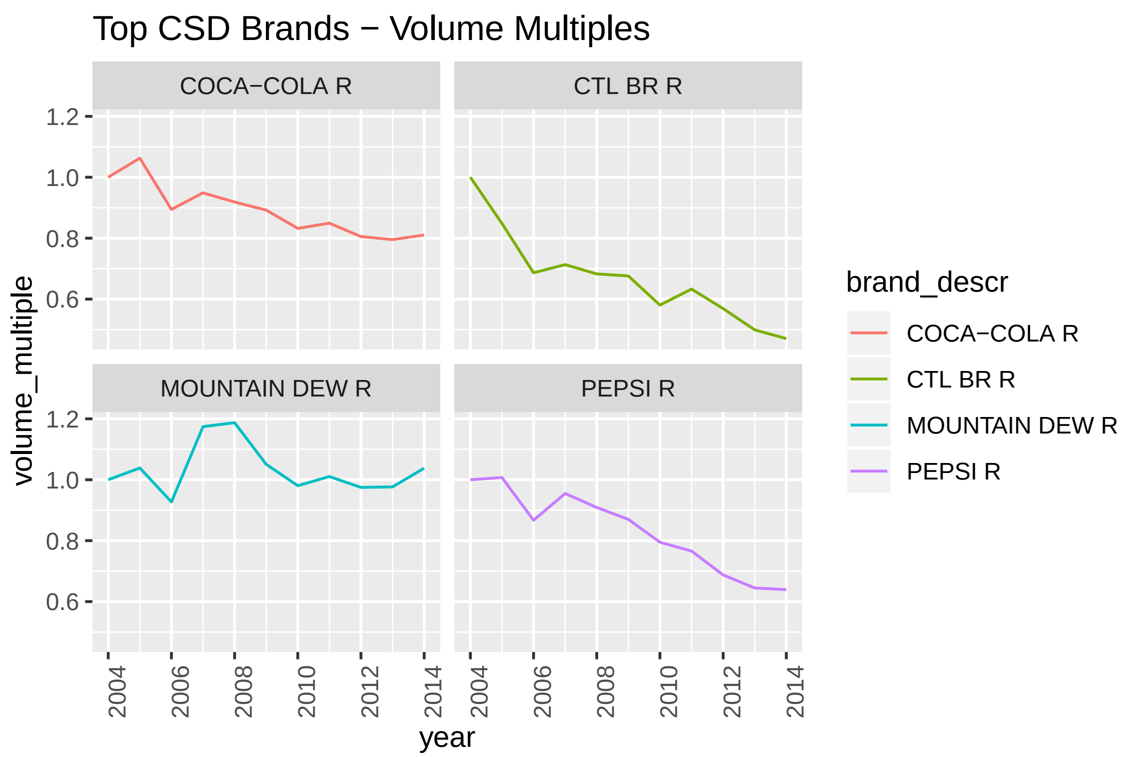
```
purchases = merge(purchases, brand_summary[, .(category,brand_descr,rank)], by=c("category", "brand_desc

# purchases_category = purchases[,
#    .(spend           = sum(total_price_paid - coupon_value),
#      purchase_volume = sum(volume),
#      no_households   = head(no_households, 1)),
#    keyby = .(category, year)]

purchases_brand = purchases[,
      .(spend          = sum(total_price_paid - coupon_value),
        purchase_volume= sum(volume),
        no_households = head(no_households,1)),
      keyby = .(brand_descr, year)]

purchases_brand = purchases_brand[, per_capita_spend := (spend/no_households)]

purchases_brand = purchases_brand[, per_capita_volume :=(purchase_volume/no_households)]

dividend_volume = purchases_brand[which(purchases_brand$year==2004),.(brand_descr, per_capita_volume)]

dividend_spend = purchases_brand[which(purchases_brand$year==2004),.(brand_descr, per_capita_spend)]

dividend_brand = merge(dividend_volume, dividend_spend)

colnames(dividend_brand)<-c("brand_descr","dividend_volume", "dividend_spend")

purchases_brand = merge(purchases_brand, dividend_brand, by = "brand_descr")

purchases_brand = purchases_brand[, volume_multiple := (per_capita_volume/dividend_volume)]

purchases_brand = purchases_brand[, spend_multiple := (per_capita_spend/dividend_spend)]

### Find top 4 'volume' brands by category

# brand_summary = purchases[, .(spend = sum(total_price_paid - coupon_value)),
#                              by = .(category, brand_descr)]
# brand_summary[, rank := frankv(spend, order = -1), by = category]
#
# brand_summary = brand_summary[order(rank),]

brand_vol_summary = purchases[, .(total_volume = sum(volume)),
                            by = .(category, brand_descr)]
brand_vol_summary[, rank := frankv(total_volume, order = -1), by = category]

brand_vol_summary = brand_vol_summary[order(rank),]

### Water
water_brands = (head(brand_vol_summary[category=="Water","brand_descr"],4))$brand_descr

### CSD
csd_brands = (head(brand_vol_summary[category=="CSD","brand_descr"],4))$brand_descr
```

```
### Diet CSD
diet_csd_brands = (head(brand_vol_summary[category=="Diet CSD","brand_descr"],4))$brand_descr

### Other
other_brands = (head(brand_vol_summary[category=="Other","brand_descr"],4))$brand_descr
```

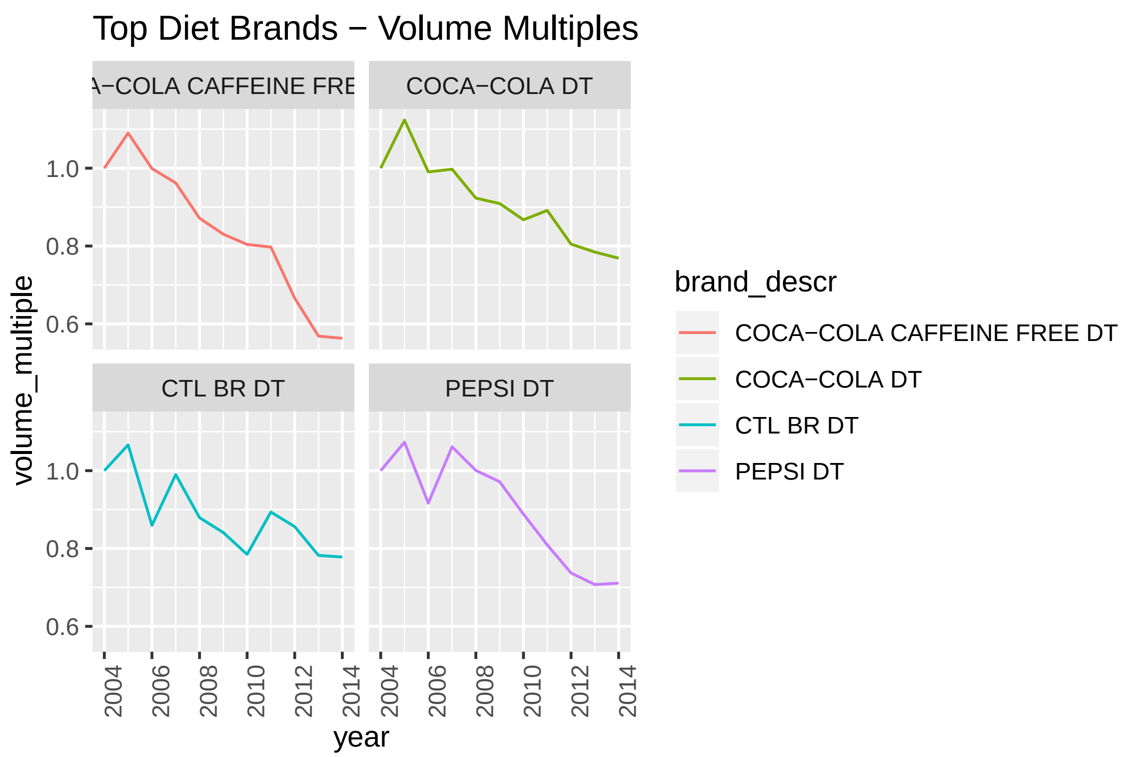
```
### Water plot - volume multiples
ggplot(purchases_brand[brand_descr %in% water_brands,], aes(x=year, y=volume_multiple, group = brand_des
```



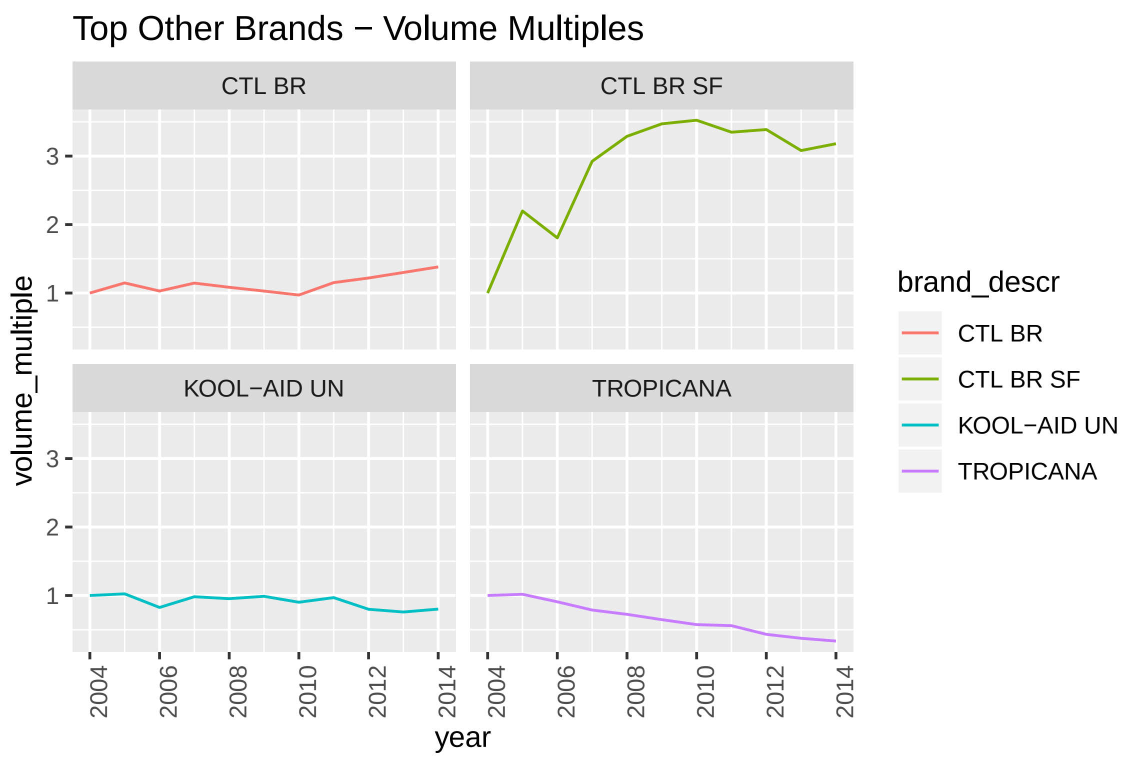

```
### CSD plot - volume multiples
ggplot(purchases_brand[brand_descr %in% csd_brands,], aes(x=year, y=volume_multiple, group = brand_desc
```



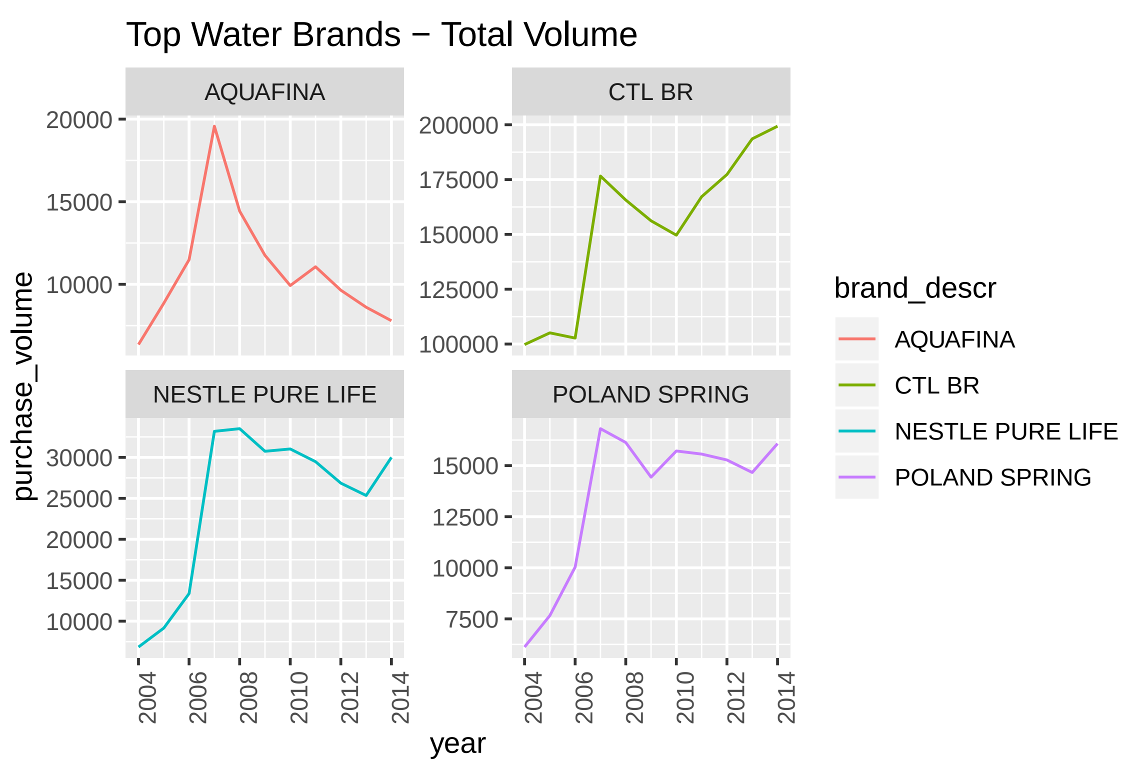

```
### Diet CSD plot - volume multiples
ggplot(purchases_brand[brand_descr %in% diet_csd_brands,], aes(x=year, y=volume_multiple, group = brand_
```



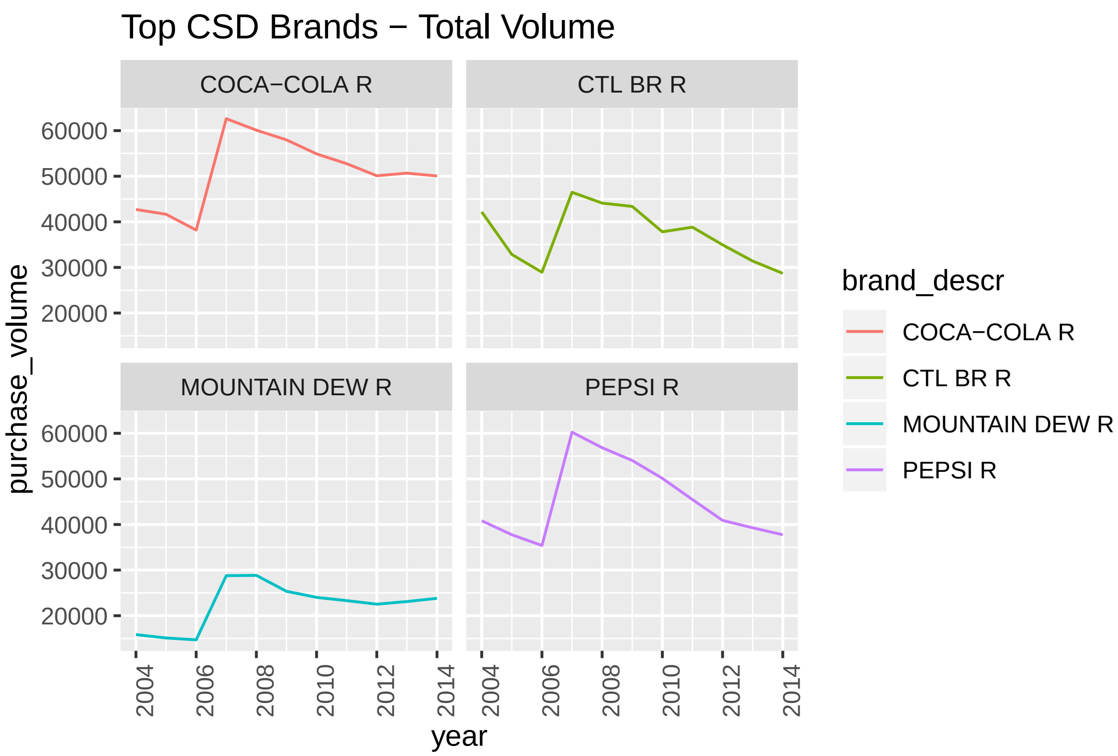

```
### Other plot - volume multiples
ggplot(purchases_brand[brand_descr %in% other_brands,], aes(x=year, y=volume_multiple, group = brand_des
```



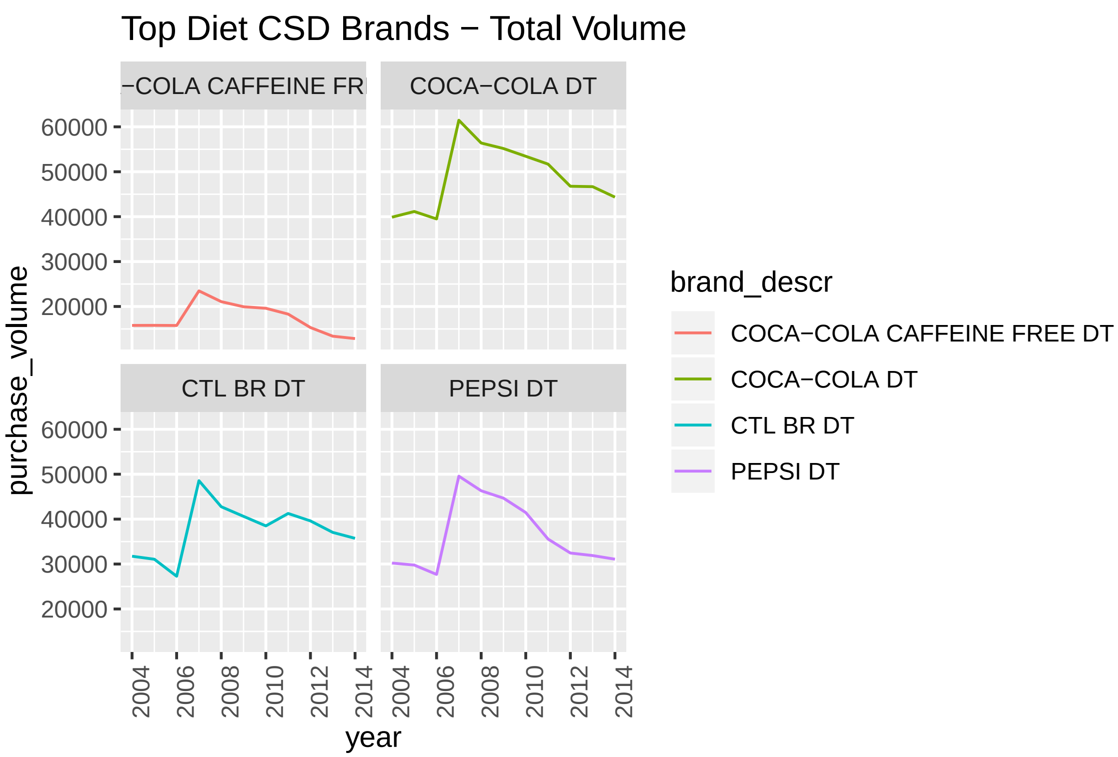

```
################################### overall volume #######################

### Water plot - total volume
ggplot(purchases_brand[brand_descr %in% water_brands,], aes(x=year, y=purchase_volume, group = brand_des
```



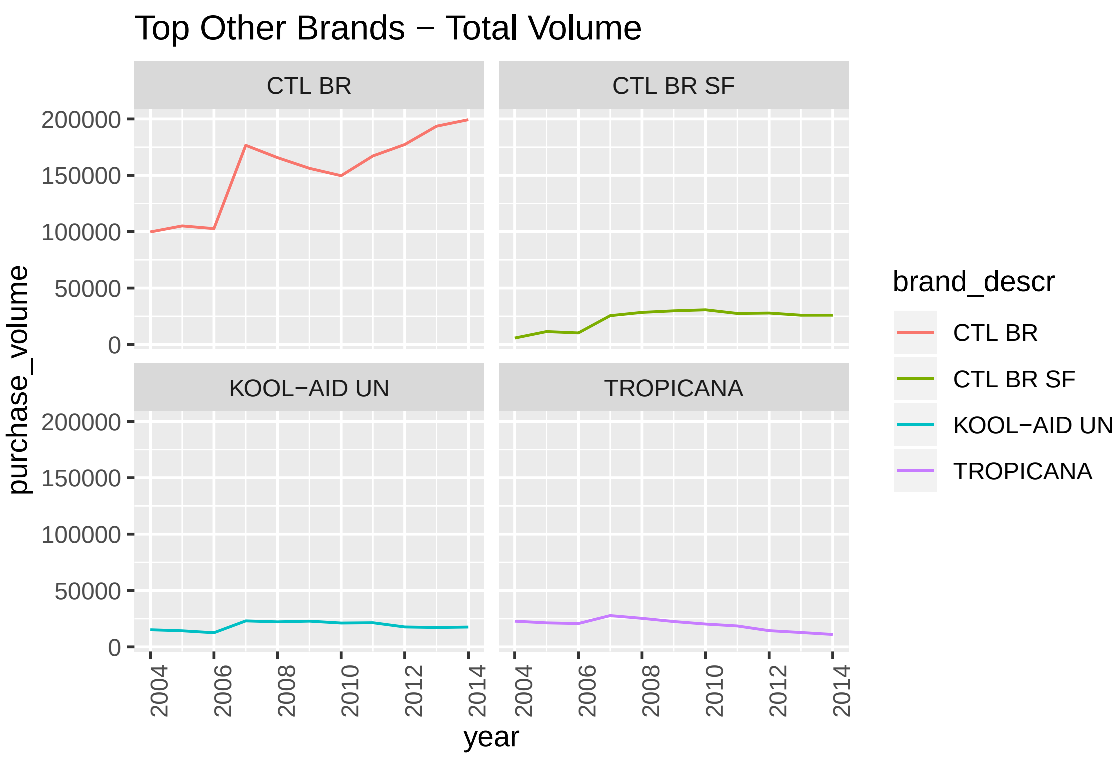
```r
### CSD plot - total volume
ggplot(purchases_brand[brand_descr %in% csd_brands,], aes(x=year, y=purchase_volume, group = brand_desc
```




```r
### Diet CSD plot - total volume
ggplot(purchases_brand[brand_descr %in% diet_csd_brands,], aes(x=year, y=purchase_volume, group = brand_
```




```
### Other plot - total volume
ggplot(purchases_brand[brand_descr %in% other_brands,], aes(x=year, y=purchase_volume, group = brand_des
```



For bottled water you may add the `scales = "free_y" option in`facet_wrap':

```r
  facet_wrap(..., scales = "free_y")
```

By default, the y-axes in a facet wrap are identical across all panels, but `free_y` let's ggplot2 choose different axes for each panel.

7 Discussion

Provide a brief discussion of the marketing implications of your findings.

Analyis of household buying behavior provides us with invaluable insights as to how the industry has transformed over the decade 2004 - 2014. This sort of analysis is useful to understand if the demand for a product is expanding/contracting, determine a company's overall market share, understand competitors' performance, current trends in the beverage space, etc.

From the category level analysis, the biggest and most important trend to be discussed is the upward trend in per_capita consumption volume of bottled water volume and downward trend in the per capita consumption volume of carbonated soft drinks, diet CSDs and other drinks -> meaning companies should look to shift towards offering healthier options.

From the brand level analysis 2004-2014:

For Bottled water, the per capita consumption of all the top brands except Aquafina increased consistently often by a large factor.

For CSD brands, the per capita consumption reduced all across the board, but the priavte label control brand took the biggest hit.

For Diet CSD brands, there was a steady decline in the per-capita consumption across all the major brands.

For Other drinks, the per-capita consumption remained fairly stable.

8 Appendix: Using a random subsample of the data

Although not necessary for the analysis in this assignment, especially as I already created a random subsample of the original data, it is useful to know how to create such a random subsample.

For example, to draw a random sample of 2 million observations without replacement, use:

```
purchases_sub = purchases[sample(.N, 2000000)]
```

For details, consult `?sample`, and note that `.N` is the number of rows in a data.table—a variable that the data.table package automatically supplies.

Alternatively, it may be even better to obtain *all* purchase data for a random sample of households. First, we obtain a 25 percent sample of all household codes in the data:

```
N_households = length(unique(purchases$household_code))
N_subsample  = round(0.25*N_households)

household_code_sub = sample(unique(purchases$household_code), N_subsample)
```

Then extract all data for the chosen household keys. As we already discussed in the data.table *Keys and Merging* overview, the first method is more readable yet somewhat slower, the second method is faster but also more confusing to the novice. The second method also does not keep its key, so you have to key the `purchases_sub_hh_a` data.table later.

```
purchases_sub_hh   = purchases[household_code %in% household_code_sub]
purchases_sub_hh_a = purchases[.(household_code_sub)]
```