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

Günter J. Hitsch

1/15/2017

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 (IL), Philadelphia, Boulder (CO), and other places. 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.

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.

Currently, the Homescan panel at the Kilts Center contains 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. Consult the `Buying-Behavior-Data-Preparation.Rmd` file to see the steps taken in the extraction and cleaning process. Once you load the data, you can verify that even the beverage data only contains more than 40 million observations (use `nrow`, `ncol`, or `dim` to see the size of a data.table or data frame).

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.

Using a subsample

Important trick: To develop and test your code, you can work with a subsample of your original data. For example, to draw a random sample of 4 million observations without replacement, use:

```
purchases_sub = purchases[sample(.N, 4000000)]
```

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 it's 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)]
```

For this analysis, you may just use the original data (40 million observations is quite easily managable using data.table on a fast computer), but it's nonetheless an important trick to remember! Also, it typically makes sense to create the subsample in an initial step and save it to a file.

To conserve memory we remove (delete) all three subsamples:

```
rm(purchases_sub)
```

```
Warning in rm(purchases_sub): object 'purchases_sub' not found
```

```
rm(list = c("purchases_sub_hh", "purchases_sub_hh_a"))
```

Variable description

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

```
head(purchases)
```

```
          upc upc_ver_uc household_code retailer_code purchase_date
1: 2114061712          1        2000000           248    2004-01-17
2: 7800008306          1        2000000           248    2004-01-17
3: 4900000551          1        2000000           248    2004-01-23
4: 7800008316          1        2000000            76    2004-02-06
5: 2500005404          1        2000000           248    2004-02-13
6: 7800008316          1        2000000           248    2004-02-28
   total_price_paid quantity coupon_value deal_flag_uc panel_year
1:             1.25        1          0.0            0       2004
2:             3.29        1          0.0            0       2004
3:             1.19        1          0.0            0       2004
4:             2.50        1          0.0            0       2004
5:             2.14        1          0.5            1       2004
6:             3.68        1          0.0            0       2004
```

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.

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

```
head(products)
```

```
       upc upc_ver_uc                  upc_descr product_module_code
1: 2002513          1             CTL BR AJ PL R                1033
2: 2003556          1        CTL BR OJ U NP PL R                1040
3: 2003557          1  CTL BR TNG/PA S JC U PL R                1044
4: 2003558          1  CTL BR TNG/KW S JC U PL R                1044
5: 2003559          1 CTL BR TNG/STB S JC U PL R                1044
6: 5399374          1        BALLYGOWAN R SMW NB                1484
                     product_module_descr product_group_code
1:                    FRUIT JUICE - APPLE                507
2: FRUIT JUICE - ORANGE - OTHER CONTAINER                507
3:                  FRUIT JUICE-REMAINING                507
4:                  FRUIT JUICE-REMAINING                507
5:                  FRUIT JUICE-REMAINING                507
6:               SOFT DRINKS - CARBONATED               1503
              product_group_descr department_code department_descr
1: JUICE, DRINKS - CANNED, BOTTLED               1      DRY GROCERY
2: JUICE, DRINKS - CANNED, BOTTLED               1      DRY GROCERY
3: JUICE, DRINKS - CANNED, BOTTLED               1      DRY GROCERY
4: JUICE, DRINKS - CANNED, BOTTLED               1      DRY GROCERY
5: JUICE, DRINKS - CANNED, BOTTLED               1      DRY GROCERY
6:            CARBONATED BEVERAGES               1      DRY GROCERY
   brand_code_uc  brand_descr multi size1_code_uc size1_amount size1_units
1:        536746       CTL BR     1         35160       59.000          OZ
2:        536746       CTL BR     1         32536        8.000          OZ
3:        536746       CTL BR     1         32536        8.000          OZ
4:        536746       CTL BR     1         32536        8.000          OZ
5:        536746       CTL BR     1         32536        8.000          OZ
6:        680566 BALLYGOWAN R     1         32900       11.154          OZ
   dataset_found_uc size1_change_flag_uc
1:              HMS                    0
2:              HMS                    0
3:              HMS                    0
4:              HMS                    0
5:              HMS                    0
```

```
6:              ALL                    0
```

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[order(product_group_descr)]
```

```
    product_module_code                        product_module_descr
 1:                1484                   SOFT DRINKS - CARBONATED
 2:                1553                  SOFT DRINKS - LOW CALORIE
 3:                7743                    FOUNTAIN BEVERAGE SYRUP
 4:                 452            REFERENCE CARD FOUNTAIN BEVERAGE
 5:                1033                          FRUIT JUICE - APPLE
 6:                1040      FRUIT JUICE - ORANGE - OTHER CONTAINER
 7:                1044                       FRUIT JUICE-REMAINING
 8:                1041                         FRUIT DRINKS-CANNED
 9:                1042                FRUIT DRINKS-OTHER CONTAINER
10:                1034                          FRUIT JUICE - GRAPE
11:                1031                                        CIDER
12:                1032 FRUIT JUICE - GRAPEFRUIT - OTHER CONTAINERS
13:                1030             FRUIT DRINKS & JUICES-CRANBERRY
14:                1055         VEGETABLE JUICE AND DRINK REMAINING
15:                1045                          FRUIT JUICE-NECTARS
16:                1036                    FRUIT JUICE - LEMON/LIME
17:                1035                FRUIT JUICE-GRAPEFRUIT-CANNED
18:                1037                    FRUIT JUICE-ORANGE-CANNED
19:                1054                    VEGETABLE JUICE - TOMATO
20:                1038                     FRUIT JUICE - PINEAPPLE
21:                1039                           FRUIT JUICE-PRUNE
22:                1051                                  CLAM JUICE
23:                2667               FRUIT JUICE - ORANGE - FROZEN
24:                2663           FRUIT JUICE - GRAPEFRUIT - FROZEN
25:                2666                FRUIT JUICE - APPLE - FROZEN
26:                2670               FRUIT DRINKS & MIXES - FROZEN
27:                2668                FRUIT JUICE - GRAPE - FROZEN
28:                2662       FRUIT JUICE - UNCONCENTRATED - FROZEN
29:                2674            FRUIT JUICE - REMAINING - FROZEN
30:                2669              FRUIT DRINKS - ORANGE - FROZEN
31:                1482                       COCKTAIL MIXES-LIQUID
32:                1487                              WATER-BOTTLED
33:                1050                      SOFT DRINKS - POWDERED
34:                1049  REMAINING DRINKS & SHAKES-NON REFRIGERATED
35:                1046                 FRUIT PUNCH BASES & SYRUPS
36:                1052                        ICE POPS - UNFROZEN
37:                1483                         COCKTAIL MIXES-DRY
38:                1048                BREAKFAST DRINKS - POWDERED
39:                1481           COCKTAIL PRODUCTS-BITTERS & HEADS
    product_module_code                        product_module_descr
             product_group_descr
 1:            CARBONATED BEVERAGES
 2:            CARBONATED BEVERAGES
```

```
 3:            CARBONATED BEVERAGES
 4:            CARBONATED BEVERAGES
 5: JUICE, DRINKS - CANNED, BOTTLED
 6: JUICE, DRINKS - CANNED, BOTTLED
 7: JUICE, DRINKS - CANNED, BOTTLED
 8: JUICE, DRINKS - CANNED, BOTTLED
 9: JUICE, DRINKS - CANNED, BOTTLED
10: JUICE, DRINKS - CANNED, BOTTLED
11: JUICE, DRINKS - CANNED, BOTTLED
12: JUICE, DRINKS - CANNED, BOTTLED
13: JUICE, DRINKS - CANNED, BOTTLED
14: JUICE, DRINKS - CANNED, BOTTLED
15: JUICE, DRINKS - CANNED, BOTTLED
16: JUICE, DRINKS - CANNED, BOTTLED
17: JUICE, DRINKS - CANNED, BOTTLED
18: JUICE, DRINKS - CANNED, BOTTLED
19: JUICE, DRINKS - CANNED, BOTTLED
20: JUICE, DRINKS - CANNED, BOTTLED
21: JUICE, DRINKS - CANNED, BOTTLED
22: JUICE, DRINKS - CANNED, BOTTLED
23:           JUICES, DRINKS-FROZEN
24:           JUICES, DRINKS-FROZEN
25:           JUICES, DRINKS-FROZEN
26:           JUICES, DRINKS-FROZEN
27:           JUICES, DRINKS-FROZEN
28:           JUICES, DRINKS-FROZEN
29:           JUICES, DRINKS-FROZEN
30:           JUICES, DRINKS-FROZEN
31:      SOFT DRINKS-NON-CARBONATED
32:      SOFT DRINKS-NON-CARBONATED
33:      SOFT DRINKS-NON-CARBONATED
34:      SOFT DRINKS-NON-CARBONATED
35:      SOFT DRINKS-NON-CARBONATED
36:      SOFT DRINKS-NON-CARBONATED
37:      SOFT DRINKS-NON-CARBONATED
38:      SOFT DRINKS-NON-CARBONATED
39:      SOFT DRINKS-NON-CARBONATED
               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).

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).

```
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 these observations.

```
purchases = purchases[year != 2003]
```

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 assigns each beverage purchase to one of these four categories.

```
products[, category := "Other"]
products[product_module_code == 1484, category := "CSD"]
products[product_module_code == 1553, category := "CSD (diet)"]
products[product_module_code == 1487, category := "Bottled water"]
```

The number of observations in each category:

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

```
Bottled water           CSD    CSD (diet)         Other
        15921         24126         10267         58754
```

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.

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.

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

For beverages, we can verify that product volume is typically measured in ounces (`OZ`), less frequently in quarts (`QT`), and only rarely in counts (`CT`):

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

```
     CT       OZ       QT
 477605 38592377  1690407
```

Let's ignore counts, and remove all corresponding data.

```
purchases = purchases[size1_units != "CT"]
```

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.

```
purchases[size1_units == "OZ", volume := (1/128)*size1_amount*multi*quantity]
purchases[size1_units == "QT", volume := (1/4)*size1_amount*multi*quantity]
```

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]
```

To view the number of households by year:

```
purchases[, head(.SD, 1), keyby = year, .SDcols = "no_households"]
```

```
    year no_households
 1: 2004         42264
 2: 2005         38781
 3: 2006         42096
 4: 2007         65441
 5: 2008         64573
 6: 2009         64391
 7: 2010         65360
 8: 2011         61786
 9: 2012         62011
10: 2013         63182
11: 2014         61269
```

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

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 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)]
```

Then, we calculate per capita spend and purchase volume (in gallons) for each category separately:

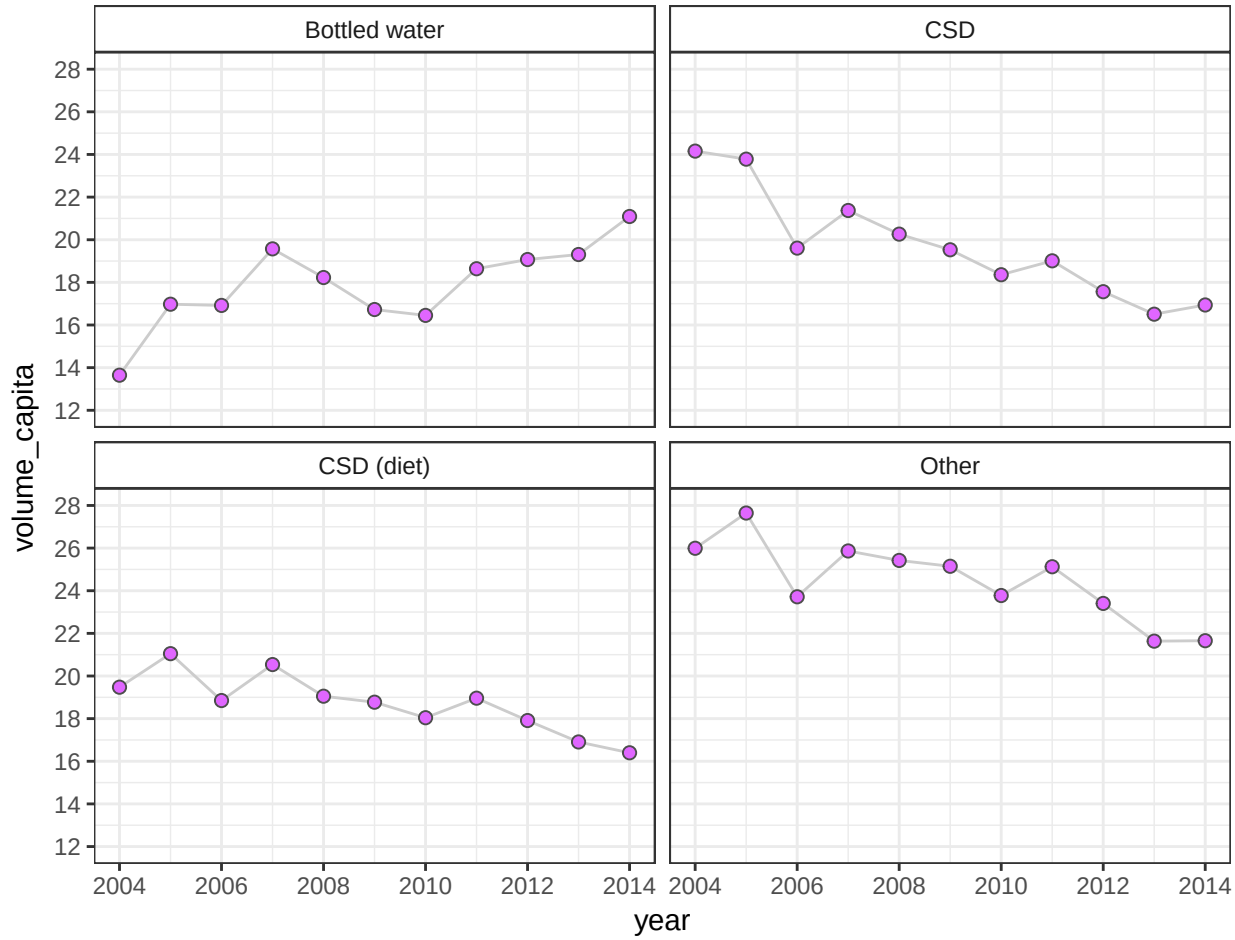
```
purchases_category[, `:=`(spend_capita  = spend/no_households,
                          volume_capita = purchase_volume/no_households)]
```

Now let's graph the evolution of the yearly per capita purchase volume for all four categories.

```
library(ggplot2)

ggplot(purchases_category, aes(x = year, y = volume_capita)) +
   geom_line(color = "gray80", size = 0.5) +
   geom_point(shape = 21, color = "gray30", fill = "mediumorchid1", size = 2, stroke = 0.5) +
   scale_y_continuous(limits = c(12, 28), breaks = seq(12, 28, 2)) +
   facet_wrap(~ category, ncol = 2) +
```

```
  theme_bw() +
  theme(strip.background=element_rect(fill="white"))
```

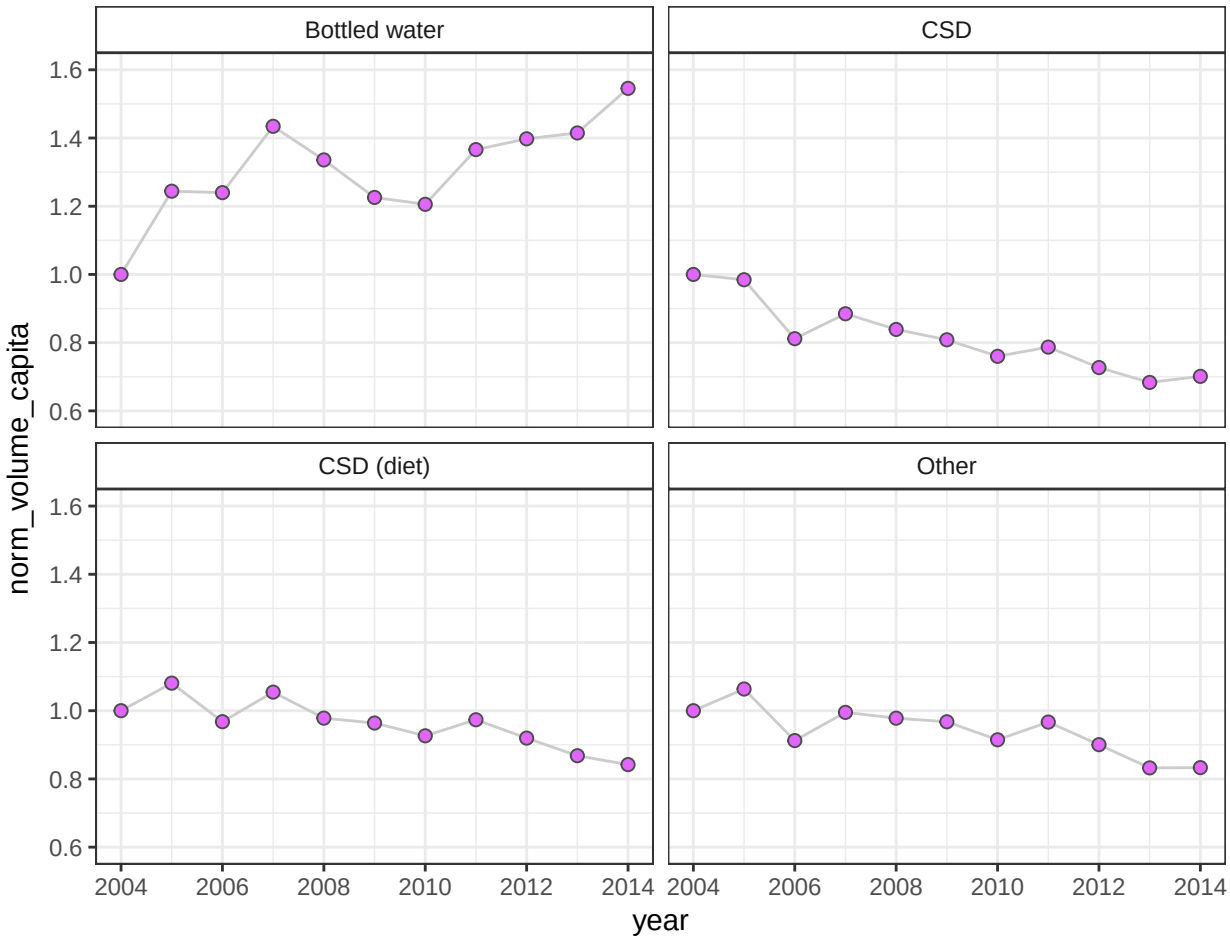


Let's express the purchase/consumption data as multiples of the 2004 values, such that per capita volume takes the value of 1.0 in all categories in 2004. Such a normalization allows us to compare the consumption series in each category directly in percentage terms.

```
purchases_category[, `:=`(norm_spend_capita  = spend_capita/head(spend_capita, 1),
                          norm_volume_capita = volume_capita/head(volume_capita, 1)),
                     by = category]
```

The corresponding graph:

```
ggplot(purchases_category, aes(x = year, y = norm_volume_capita)) +
  geom_line(color = "gray80", size = 0.5) +
  geom_point(shape = 21, color = "gray30", fill = "mediumorchid1", size = 2, stroke = 0.5) +
  scale_y_continuous(limits = c(0.6, 1.6), breaks = seq(0.6, 1.6, 0.2)) +
  facet_wrap(~ category, ncol = 2) +
  theme_bw() +
  theme(strip.background=element_rect(fill="white"))
```



We clearly see that category-consumption of sodas, both regular and diet, has declined strongly since 2004. For example, in 2014 regular CSD consumption had declined by about 30 percent relative to the 2004 consumption level. Interestingly, even diet soda consumption declined by about 23 percent. On the other hand, bottled water consumption increased by almost 60 percent over the same time horizon.

Brand-level analysis

Now let's investigate the evolution of consumption for some of the key brands in the soda and bottled water categories. First, we need to merge the brand identifier 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]
```

Now merge the brand ranks in the `brand_summary` table with the `purchases` information.

```
setkey(brand_summary, category, brand_descr)
setkey(purchases, category, brand_descr)
purchases = merge(purchases, brand_summary[, .(category, brand_descr, rank)])
```

Aggregate to the brand level, as we did before at the category level:

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

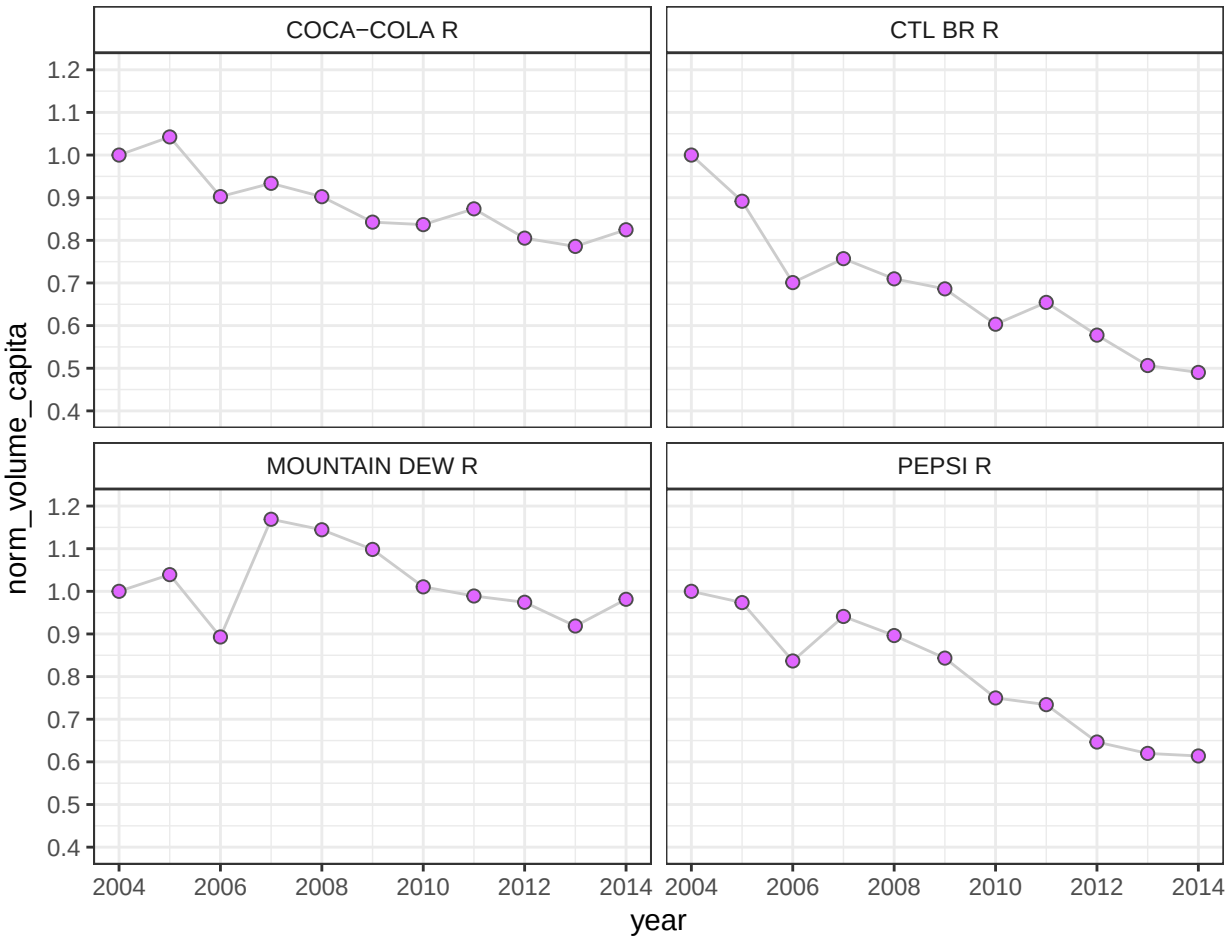
Then calculate per capita and normalize per capita consumption:

```
purchases_brand[, `:=`(spend_capita  = spend/no_households,
                       volume_capita = purchase_volume/no_households)]

purchases_brand[, `:=`(norm_spend_capita  = spend_capita/head(spend_capita, 1),
                       norm_volume_capita = volume_capita/head(volume_capita, 1)),
                  by = .(category, brand_descr)]
```

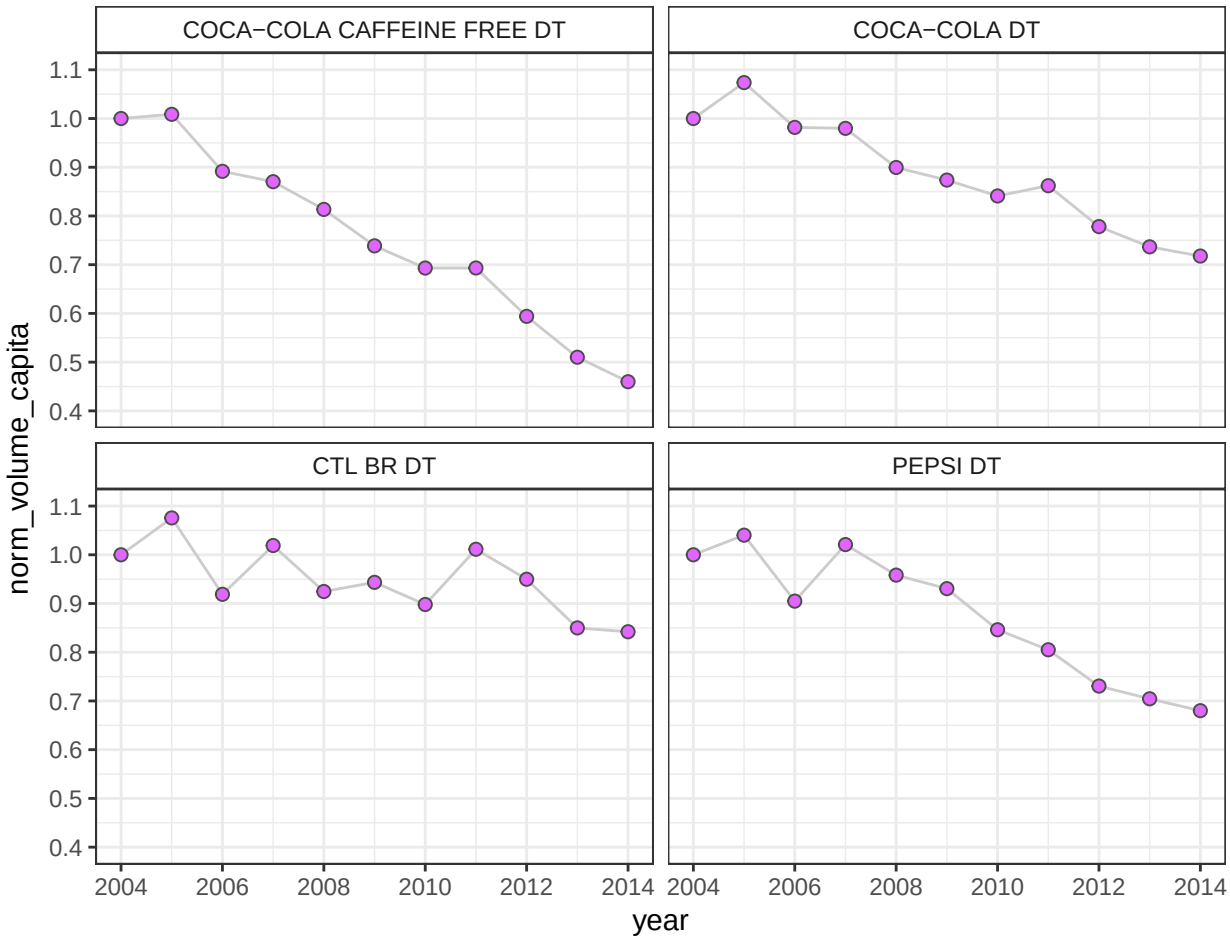
Now let's plot the evolution of brand volume for the top 4 brands, first in the CSD category:

```
ggplot(purchases_brand[rank <= 4 & category == "CSD"], aes(x = year, y = norm_volume_capita)) +
   geom_line(color = "gray80", size = 0.5) +
   geom_point(shape = 21, color = "gray30", fill = "mediumorchid1", size = 2, stroke = 0.5) +
   scale_y_continuous(limits = c(0.4, 1.2), breaks = seq(0.4, 1.2, 0.1)) +
   facet_wrap(~ brand_descr) +
   theme_bw() +
   theme(strip.background=element_rect(fill="white"))
```



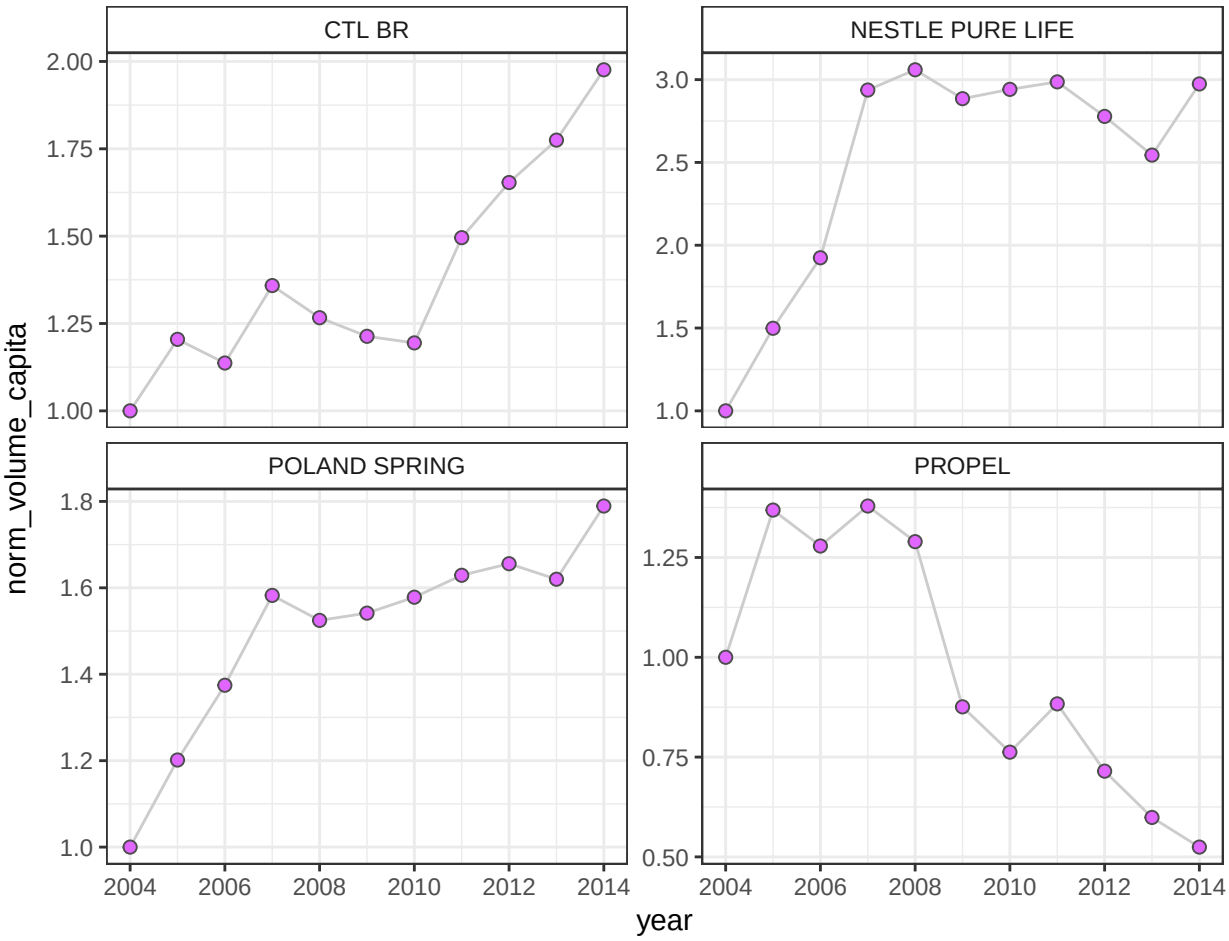
Then for diet sodas:

```
ggplot(purchases_brand[rank <= 4 & category == "CSD (diet)"], aes(x = year, y = norm_volume_capita)) +
   geom_line(color = "gray80", size = 0.5) +
   geom_point(shape = 21, color = "gray30", fill = "mediumorchid1", size = 2, stroke = 0.5) +
   scale_y_continuous(limits = c(0.4, 1.1), breaks = seq(0.4, 1.1, 0.1)) +
   facet_wrap(~ brand_descr) +
   theme_bw() +
   theme(strip.background=element_rect(fill="white"))
```



And finally for bottled water. Note the option `scales = "free_y"` in `facet_wrap`. 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. If you eliminate the option you will see why we used it in this graph.

```
ggplot(purchases_brand[rank <= 4 & category == "Bottled water"], aes(x = year, y = norm_volume_capita))
   geom_line(color = "gray80", size = 0.5) +
   geom_point(shape = 21, color = "gray30", fill = "mediumorchid1", size = 2, stroke = 0.5) +
   facet_wrap(~ brand_descr, scales = "free_y") +
   theme_bw() +
   theme(strip.background=element_rect(fill="white"))
```


CTL BR
NESTLE PURE LIFE
POLAND SPRING
PROPEL
norm_volume_capita
year
2.00
1.75
1.50
1.25
1.00
3.0
2.5
2.0
1.5
1.0
1.8
1.6
1.4
1.2
1.25
1.00
0.75
0.50
2004
2006
2008
2010
2012
2014