**Overview of the Basic Ideas in the Price Personalization Model/Code**

What we want to do is to find **an optimal price for each houshold**. We start with a *basic sales model* that is *sales = x\*beta* as we discussed in the Class 2. In this set-up *x* has variables that contain ***price*** and *other variables* that do not contain price (e.g., demographics). We first estimate the model to get *beta*.[[1]](#footnote-1)

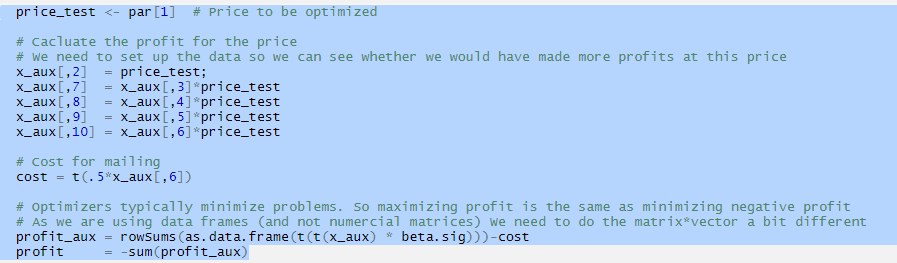
In this set-up, we want to find the **optimal price**. Say for business reason the optimal price can only come from the interval [5,7]. Typically, there are prices one can simply not charge. What we need to do is decide on a reasonable delta, let’s say 0.01, and try all prices: 5, 5.01, 5.02, …, 6.99, 7 (i.e., a grid search). If we do this “by hand” this is quite an exhausting undertaking in terms of time. There are two steps one would need to do

1. For each price to be tried, we would need to make ***x\_test*** for this price (recall that *x* includes price and non-price variables).
2. Calculate the expected profit given that price to be tested. Profit in our case is given by: *profit\_test* = *x\_test(price)\*beta-cost*.

Instead of doing this “by hand” we will use R to implement these two steps. As step 1 is needed to be repeated many times over (i.e., you will have to make the dataset *x\_test* for all prices to be tested, it is MUCH more efficient to use a function that can be called vs. coding every test by using a new piece of code.

**Setting up a FUNCTION – STEP 2 of the Code**

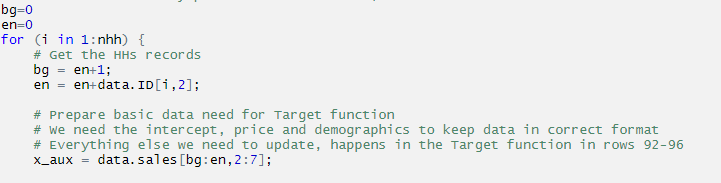
When you define a function it needs to be given a set of inputs. In our case that this a price to test (called par, short for parameter) and data (in our case called x\_aux, which are the x for one particular household). The function allows us to create a new dataset to be tested, *x\_test(price)*, for the household. Below in the code snapshot this is done in the x\_aux set of equations. The function allows us to calculate all variables that are depending on the price to be tested. It also calculates cost for the household, which is a function of where the household is located in our example. The profit equation sums up the profits for the household for the price to be tested. The function then gives the profits for the particular household back to the optimizer.



Now that we have a function that can calculate the profit for any given price for a single houshold (in my example we are only using prices between 5 and 7) – recall we are trying to find the optimal price for EACH household.

**Using the FUNCTION – Step 3 of the Code**

Now we can use the function **to test a set of prices for each household** to find the optimal price for a given household. The first part of STEP 3 (in which we loop over all households) “finds” the data for **ONE** household. In our case we have 200 households, so *nhh=200*.



As the data are in rows and not all households have the same number of purchases, we need to be able to look up the data for each HH separately. This is what *bg* and *en* (short for begin and end, counters that help me) are used for and they are updated in each step – these counters tell us where a household’s data begins and ends in the dataset.

Find below an example how the data look like. The first table is the *x* data. The second table is the *data.ID* data. The third table is what I am calculating with *bg* and *en* to look up the correct household records from the *x* database. For example, if we want to look up the data for household 3 we need to know where it begins (row 9) and where it ends (row 10), household 3 has two sales. One way to do this is, as shown above, to loop over all households (that computationally a lot less “expensive” then other methods to look up the household 3’s data). The loop begins with household 1 (naturally, bg=1 for the first household). Household 1 has 3 purchases, so the end of household 1’s data is in row 3, so en=3. Which means for household 2: bg = en+1 = 3+1 = 4, so household 2’s data starts in row 4. Household 2 has 5 sales, so its data end in row 8: en = en+5 = 3+5 = 8. Household 3’s data now starts in row 9: bg = en+1 = 8+1 = 9. Household 3 has two sales and thus household 3’s data end in row 10: en=en+2 = 8+2 = 10. 

**Optimizing the FUNCTION – Step 4 of the Code**

The next part of the code is the optimizer. This is the “solver” that tries all kinds of values for price from the interval [5,7] I have predefined. For each household it gives you back the optimal price.

1. Side Note: In a more typical setting (for example in the acquisition scoring we will discuss in Class 4) we would already have a set of data that we want to use to forecast, x\_forecast – to get sales\_forecast, we would simply do x\_forecast\*beta in such a setting.  
    [↑](#footnote-ref-1)