Workshop 11

Conjoint Analysis II: Market simulation & design optimization

MSBX-5130: Customer Analytics

In the previous workshop, we introduced conjoint analysis, an experimental technique to measure consumer preferences for goods and services.

We demonstrated how to perform conjoint analysis in a sequence of six steps:

1. Identify a set of relevant product attributes
2. Define reasonable levels for those attributes
3. Create product profiles
4. Obtain consumer preferences for profiles
5. Analyze the data for each respondent
6. Simulate market outcomes

In the previous workshop, we completely covered steps 1-5 and provided simplified examples of step 6 (market simulation). In today's workshop, we will consider market simuluations in greater depth, with the primary intent of optimizing a firm's product line design.

We will work with data similar to the last workshop, in that we use response data to the same survey (on tablet computer preferences). Here, we will use a respresentative sample of subjects (representative of our customer base), and use their responses to simulate market outcomes.

1) Setup

I have provided a data file that contains the survey design used for our workshop example, `survey_design.csv`. First, load this file into a dataframe called `design_DF`. Print the dataframe `design_DF`.

Next, load and summarize the subject responses (ratings), provided in the data file `respondent_data.csv`. Name the resulting dataframe `responses_DF`. Use `summary()` to summarize the dataframe `responses_DF`.

```
setwd("/Users/jeremygreen/Desktop/")

responses_DF <- read.csv('respondent_data.csv')
design_DF <- read.csv('survey_design.csv')
head(design_DF)
```

```
  Screen Cell Price Battery      OS
1      7    Y   300      12 Windows
2      7    Y   100       8 Windows
3     10    Y   500      12 Android
4      7    Y   300       4 Android
5      7    N   300       8     iOS
6     10    N   300      12 Windows
```

```r
library(psych)
describe(design_DF)
```

```
        vars  n  mean     sd median trimmed    mad min max range skew kurtosis
Screen     1 18   8.5   1.54    8.5     8.5   2.22   7  10     3    0    -2.11
Cell*      2 18   1.5   0.51    1.5     1.5   0.74   1   2     1    0    -2.11
Price      3 18 300.0 168.03  300.0   300.0 296.52 100 500   400    0    -1.66
Battery    4 18   8.0   3.36    8.0     8.0   5.93   4  12     8    0    -1.66
OS*        5 18   2.0   0.84    2.0     2.0   1.48   1   3     2    0    -1.66
           se
Screen   0.36
Cell*    0.12
Price   39.61
Battery  0.79
OS*      0.20
```

```r
describe(responses_DF)
```

```
              vars   n  mean    sd median trimmed   mad min max range  skew
respondent_id    1 114 74.17 43.16     73   74.40 57.08   1 145   144 -0.01
profile_1        2 114  4.00  1.61      4    4.07  1.48   1   7     6 -0.25
profile_2        3 114  4.68  1.67      5    4.79  1.48   1   7     6 -0.58
profile_3        4 114  3.07  1.63      3    2.95  1.48   1   7     6  0.44
profile_4        5 114  2.14  1.23      2    1.97  1.48   1   7     6  1.19
profile_5        6 114  3.54  1.48      4    3.55  1.48   1   7     6 -0.20
profile_6        7 114  3.86  1.69      4    3.89  1.48   1   7     6 -0.15
profile_7        8 114  2.19  1.30      2    2.03  1.48   1   7     6  1.00
profile_8        9 114  2.97  1.51      3    2.90  1.48   1   7     6  0.29
profile_9       10 114  1.79  1.16      1    1.57  0.00   1   6     5  1.57
profile_10      11 114  6.59  0.95      7    6.82  0.00   1   7     6 -3.00
profile_11      12 114  3.11  1.42      3    3.09  1.48   1   6     5  0.15
profile_12      13 114  4.77  1.79      5    4.93  1.48   1   7     6 -0.63
profile_13      14 114  3.20  1.51      3    3.18  1.48   1   7     6  0.09
profile_14      15 114  2.35  1.19      2    2.25  1.48   1   5     4  0.59
profile_15      16 114  3.16  1.54      3    3.07  1.48   1   7     6  0.49
profile_16      17 114  3.69  1.88      4    3.62  2.97   1   7     6  0.21
profile_17      18 114  3.39  1.44      3    3.38  1.48   1   7     6  0.09
profile_18      19 114  3.14  1.61      3    3.02  1.48   1   7     6  0.51
              kurtosis   se
respondent_id    -1.23 4.04
profile_1        -0.82 0.15
profile_2        -0.50 0.16
profile_3        -0.67 0.15
profile_4         1.57 0.11
profile_5        -0.66 0.14
profile_6        -0.97 0.16
profile_7         0.68 0.12
profile_8        -0.71 0.14
profile_9         2.13 0.11
profile_10       11.07 0.09
profile_11       -0.90 0.13
profile_12       -0.56 0.17
```

```
profile_13       -0.99 0.14
profile_14       -0.67 0.11
profile_15       -0.28 0.14
profile_16       -1.06 0.18
profile_17       -0.69 0.13
profile_18       -0.36 0.15
```

Discussion:

- Which profile has the highest mean rating? Lowest mean rating?

Profile 10 has the highest mean rating of 6.59 and profile 9 has the lowest rating of 1.79.

2) Analyze the data for each respondent (Conjoint step 5, revised)

In the last workshop, we analyzed survey responses for a single subject (you), using a "part-worth" model (encoding attribute levels as dummy variables) for all attributes.

Today, we will analyze survey responses for a representative sample of consumers – i.e., a group of individuals that are similar in composition to our (existing +/or prospective) customer base. Using a representative sample is critical, as we want market share predictions to reflect the expected behavior of the target market as a whole.

2.1) Revised utility model: Part-worth model for non-price attributes, linear model for price

When optimizing product designs (i.e. choosing the product design that is expected to maximize profits), it is frequently convenient to treat price as a continuous variable. That is, we often wish to relax the implicit assumption of the part-worth model that prices can only attain two or more discrete levels. If we instead treat price as a continuous regressor, we can test price levels other than those explicitly considered in the survey design. To keep matters simple, we will restrict attention to modeling price with a single linear effect.

Taking this approach entails estimating the following regression for *each* respondent:

```
lm(response~factor(Screen) + factor(Cell) + Price + factor(Battery) + factor(OS))
```

That is, we model `Screen`, `Cell`, `Battery` and `OS` attributes using the "part-worth" (dummy variable) model and `Price` using a simple linear model.

Note that, in some cases, firms may seek to optimize other attributes similar to price (by allowing for attribute levels other than the discrete levels included in the survey design). This can be achieved provided: a) the attribute in inherently continuous, and b) we are willing to take extra steps to optimize the additional variable(s). For example, we might consider `Battery` as a second continuous attribute, since in principle battery life could be engineered for values other than 4, 8 or 12 hours. In today's example, we will restrict attention to just optimizing `Price` as a continuous variable. This is consistent with assuming engineering contraints restrict non-price attributes.

#this is saying that price can now be other things than just 300 or 500 because it is now continuous

2.2) Computation and storage of multiple regression results

We will regress the consumer's preferences on the characteristics (i.e., attribute levels) of the various tablet PCs that he or she rated. This will allow us to disentangle the consumer's preferences for each attribute. Here we will generate similar estimates for each individual represented in our sample data, `responses_DF`.

A challenge to working with many individual-level regression results is how to organize and access those results in a programmatic way. There are many potential approaches to storing individual regression results. I will demonstrate the simplest approach, which involves storing regression results in a "list of lists". That is, we will construct an array of regression output objects, each of which is a structured list.

To do this, take the following steps:

1. Create an empty list-of-lists to hold the regression results for each individual. The length of this list-of-lists should be equal to the number of subjects in `responses_DF`. Call this list-of-lists `lm_res`.
    - HINT: You can use the `vector()` function to initialize a list-of-lists. For example, to create an empty list-of-lists with length 10, one can use: `lm_res = vector(mode="list", length=10)`
2. Loop over individuals. For each individual i:
    1. Create a dataframe that combines `design_DF` with the responses for individual i.
    2. Estimate the model specified in the previous subsection with individual i's responses as the dependent variable
    3. Store the model results to the i'th element of `lm_res`. HINT: Recall that to access or assign (top level) lists in a list-of-lists, we must use double bracket indexing, as in: `lm_res[[i]]`.

After completing and storing the regressions for all subjects, summarize the results for the first respondent. HINT: Use `summary()` for the results stored in `lm_res[[1]]`.

```
N = dim(responses_DF)[1]

lm_res = vector(mode = 'list', length = N)

for (i in 1:N) {

  response = as.numeric(responses_DF[i,2:ncol(responses_DF)])
  est_DF = cbind(design_DF, response = response)

  lm_res[[i]] = lm(response ~ factor(Screen) + factor(Cell)
                 + Price + factor(Battery) + factor(OS),
                 data = est_DF)

}

summary(lm_res[[1]])
```

```
Call:
lm(formula = response ~ factor(Screen) + factor(Cell) + Price +
    factor(Battery) + factor(OS), data = est_DF)

Residuals:
    Min      1Q  Median      3Q     Max
-1.5000 -0.4167  0.0000  0.5417  1.0000

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)        4.375e+00  6.466e-01   6.766 4.94e-05 ***
factor(Screen)10  -2.500e-01  3.992e-01  -0.626 0.545196
factor(Cell)Y      1.750e+00  3.992e-01   4.384 0.001370 **
```

```
Price            -7.083e-03  1.215e-03  -5.831 0.000166 ***
factor(Battery)8  -1.660e-16  4.859e-01   0.000 1.000000
factor(Battery)12  8.707e-16  4.859e-01   0.000 1.000000
factor(OS)iOS      8.333e-01  4.859e-01   1.715 0.117114
factor(OS)Windows  1.667e-01  4.859e-01   0.343 0.738702
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8416 on 10 degrees of freedom
Multiple R-squared:  0.8524,    Adjusted R-squared:  0.7491
F-statistic: 8.252 on 7 and 10 DF,  p-value: 0.001776
```

3) Simulate market outcomes (Conjoint step 6, revised)

One of the great strengths of conjoint analysis is the ability to evaluate "what if" scenarios, such as how would a hypothetical "new" product would fare in competition with existing products.

For the workshop example, we are focused on the market for tablet computers. Specifically, imagine you are the Toshiba company circa 2011. At that time, the 10" screen Apple iPad was the only tablet on the market. Toshiba is considering the introduction of a tablet computer, and wants to know was product design will be most profitable when competing against the Apple iPad. Toshiba has the capability to manufacture tablet computers similar to Apple, but Toshiba is restricted to using either the Android or Windows operating system (iOS is not an option).

In the context of our example, we want to predict how a new product by Toshiba will compete against the existing iPad.

Specifically, we assume the existing "iPad" product corresponds to:

- Screen = 10 (inches)
- Cell = "Y" (has cell connectivity)
- Price = 500 ($)
- Battery = 8 (hrs)
- OS = "iOS"

We also assume that Toshiba is initially considering two potential product designs, `Toshiba_A` and `Toshiba_B`.

The `Toshiba_A` product corresponds to:

- Screen = 7 (inches)
- Cell = "N" (no cell connectivity)
- Price = 300 ($)
- Battery = 12 (hrs)
- OS = "Android"

The `Toshiba_B` product corresponds to:

- Screen = 10 (inches)
- Cell = "N" (no cell connectivity)
- Price = 300 ($)
- Battery = 12 (hrs)
- OS = "Android"

Create and print a dataframe called `prods1` that contains the iPad (as defined above) as the first product (row) and the `Toshiba_A` product in the second row.

```r
iPad <- data.frame(Screen = 10, Cell = "Y", Price = 500, Battery = 8, OS = "iOS")
Toshiba_A <- data.frame(Screen = 7, Cell = "N", Price = 300, Battery = 12, OS = "Android")
row.names(iPad) <- "iPad"
row.names(Toshiba_A) <- "Toshiba_A"

prods1 <- rbind(iPad, Toshiba_A)

print(prods1)
```

```
          Screen Cell Price Battery      OS
iPad          10    Y   500       8     iOS
Toshiba_A      7    N   300      12 Android
```

Also create and print a dataframe called `prods2` that contains the iPad (as defined above) as the first product (row) and the `Toshiba_B` product in the second row.

```r
iPad <- data.frame(Screen = 10, Cell = "Y", Price = 500, Battery = 8, OS = "iOS")
Toshiba_B <- data.frame(Screen = 10, Cell = "N", Price = 300, Battery = 12, OS = "Android")
row.names(iPad) <- "iPad"
row.names(Toshiba_B) <- "Toshiba_B"

prods2 <- rbind(iPad, Toshiba_B)

print(prods2)
```

```
          Screen Cell Price Battery      OS
iPad          10    Y   500       8     iOS
Toshiba_B     10    N   300      12 Android
```

In section 3.1 below, we will devise functions to calculate expected demand, production costs, and firm profits, assuming the original iPad is competing with the `Toshiba_A` product. We also demonstrate how to optimize the price of `Toshiba_A`, assuming the other attribute levels remain fixed.

In section 3.2 below, we repeat the analysis, assuming the original iPad is competing with the `Toshiba_B` product. We also demonstrate how to optimize the price of `Toshiba_B`, assuming the other attribute levels remain fixed. By comparing expected profits for `Toshiba_A` and `Toshiba_B`, we can determine which product will bring the most profit to the firm.

Later (Section 3.3), we consider the full optimization of non-price attributes. That is, we allow Toshiba to sequentiually test all possible product designs (not just `Toshiba_A` and `Toshiba_B`) to compete against the original iPad.

3.1) Simulation of iPad vs Toshiba_A

We will use the above information on products to simulate the choices made by each consumer in our dataset, `responses_DF`. Using the part-worth (regression) estimates for each subject, we can estimate how much each subject would like each option, and therefore predict how he or she would choose. With knowledge of demand (product choice) and costs, we can calculate the expected profits for Toshiba when competing against the iPad.

To perform the simulation, we will first create "helper" functions to compute product demand, product costs, and firm profits.

3.1.1) Demand function

Here we engineer a function to compute demand (the number of consumers choosing each product), given: a) a set of products that will compete in the marketplace (e.g., `prods1`), and b) a set of regression results, one for each subject in our study (e.g. `lm_res`).

Below, write a function called `comp_demand` that takes two inputs:

- `res_list` – the list of regression output objects (for all subjects)
- `prods_DF` – the dataframe containing the products competing the market

The output of `comp_demand` should be a vector/list with the total number of subjects choosing each product in the market (e.g., if 2 products are in the market, the output should be a list of length 2).

HINT: One approach is to first loop over subjects within the function, predicting which product each subject will choose. For a given subject, this entails predicting ratings for each product and recording the product with the highest rating/utility. A second loop over products can then be used to calculate the total number of subjects choosing each product.

After completing your function, call it using the `prods1` market definition and the list of respondent regression results `lm_res`.

```
comp_demand <- function(res_list, prods_DF) {
  demand <- rep(0, nrow(prods_DF))

  for (i in seq_along(res_list)) {
    ratings <- predict(res_list[[i]], newdata = prods_DF)
    max_rating_idx <- which.max(ratings)
    demand[max_rating_idx] <- demand[max_rating_idx] + 1
  }

  return(demand)
}

demand <- comp_demand(lm_res, prods1)

cat("Demand for Toshiba_A:", demand[1], "\n")
```

```
Demand for Toshiba_A: 73
```

```
cat("Demand for iPad:", demand[2], "\n")
```

```
Demand for iPad: 41
```

```
#this is the number of subjects choosing each product
#switch the labels
```

Discussion:

- What is the expected demand for Toshiba_A, for the iPad?

The expected demand for Toshiba_A is 73 and 41 for the iPad

3.1.2) Cost function

Next, we create a function to calculate product production costs.

We are told from Toshiba's engineering team that the marginal cost to produce a tablet computer is approximately given by the following equation:

MC = 75 + 5*(Screen==10) + 20*(Cell==“Y”) + 5*(Battery==8) + 15*(Battery==12) + 3*(OS==“Windows”)

Below, write a function called `comp_cost` that takes one input:

- `prods_DF` – the dataframe containing the products competing the market

The output of `comp_cost` should be a vector/list with the production cost of each product in the market (e.g., if 2 products are in the market, the output should be a list of length 2).

HINT: One approach is loop over products, calculating the cost of each using the formula above.

After completing your function, call it using the `prods1` market definition to report the estimated production costs for each product.

```
comp_cost <- function(prods_DF) {

    cost <- 75 + 5*(prods_DF$Screen == 10) + 20*(prods_DF$Cell == "Y") +
    5*(prods_DF$Battery == 8) + 15*(prods_DF$Battery == 12) +
    3*(prods_DF$OS == "Windows")

  return(cost)
}


cost <- comp_cost(prods1)
cat("Cost for Toshiba_A:", cost[1], "\n")
```

```
Cost for Toshiba_A: 105
```

```
cat("Cost for iPad:", cost[2], "\n")
```

```
Cost for iPad: 90
```

```
#calculates cost for each product, then is used on prods1 which has Toshiba A and iPad.
#switch the labels
```

Discussion:

- What is the expected cost for Toshiba_A, for the iPad?

Toshiba_A = 105, iPad = 90

3.1.3) Profit function with pre-specified Toshiba_A price

We are now ready to compute profits for Toshiba, assuming the product designs are exactly as specified in `prods1`.

Below, write a function called `profit1` that takes three inputs:

- `res_list` – the list of regression output objects (for all subjects)
- `prods_DF` – the dataframe containing the products competing the market
- `sum_ndx` – index value(s) of product profits to be added up. Index values should correspond to products (rows of `prods_DF`) that are produced by Toshiba. Note that we allow for multiple products to be produced by the same company, as is common when firms offer multiple products in a product line – in such cases, we are interested in the total profit across all products produced by the focal firm (Toshiba in our example).

The output of `profit1` should be a scalar (number) with the total expected profit for Toshiba.

HINT: Within the function `profit1`, call `comp_demand()` and `comp_cost()` to calculate product demand and costs. Then loop over products specified in `sum_ndx` to calculate the total profit for Toshiba – here, `sum_ndx` should contain a single value (2, if you have defined `prods1` as requested at the beginning of section 3).

After completing your function, call it to report the expected profit for Toshiba.

```r
profit1 <- function(res_list, prods_DF, sum_ndx) {
  demand <- comp_demand(res_list, prods_DF)
  cost <- comp_cost(prods_DF)

  total_profit <- 0
  for (i in sum_ndx) {
    total_profit <- total_profit + (prods_DF$Price[i] - cost[i]) * demand[i]
  }

  return(total_profit)
}

profit_toshiba <- profit1(lm_res, prods1, 2)
print(profit_toshiba)
```

```
[1] 8610
```

```r
#this is the total expected profit for toshiba
```

Discussion:

- What is the expected profit for Toshiba?

$8610

3.1.4) Profit function with variable Toshiba_A price

In this section, we explore the possibility of improving the price (only) of Toshiba_A, holding the other attribute levels (for all products) fixed.

Optimizing the Toshiba_A price involves two steps: a) creation of an auxillary profit function that can evaluate profits at prices other than those in the pre-specified design, and b) using the auxillary profit function to evaluate profits over a set of possible prices for Toshiba_A. Generally, we call such an approach a "grid search" since we restrict attention to a finite set of alternative prices.

For the first step, create a function called `profit2` that takes four inputs:

- `res_list` – the list of regression output objects (for all subjects)
- `prods_DF` – the dataframe containing the products competing the market
- `sum_ndx` – index value(s) of product profits to be added up.
- `price` – the "new" price for Toshiba_A

The output of `profit2` should be a scalar (number) with the total expected profit for Toshiba.

HINT: Within the function `profit2`, update the value of the Toshiba_A price in `prods_DF`. Then call `profit1()` to calcuate Toshiba's profits.

After completing your function, call it to report the expected profit for Toshiba, assuming a price of $250.

```
profit2 <- function(res_list, prods_DF, sum_ndx, price) {
  prods_DF$Price[2] <- price

  total_profit <- profit1(res_list, prods_DF, sum_ndx)

  return(total_profit)
}

profit_toshiba_newprice <- profit2(lm_res, prods1, 2, 250)
print(profit_toshiba_newprice)
```

```
[1] 7040
```

```
#profit for toshiba using the price of 250 instead of 300
```

Discussion:

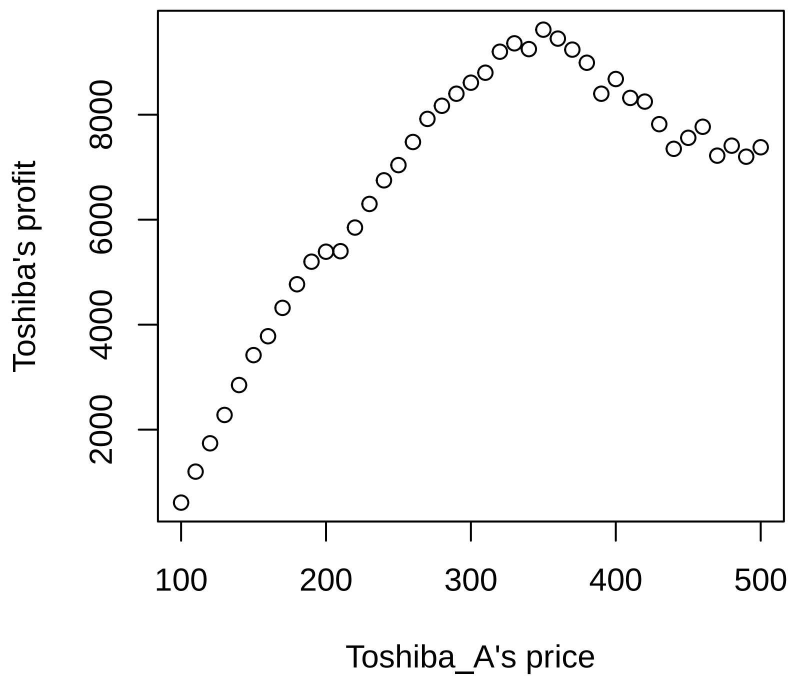
- What is the expected profit for Tobshiba assuming Toshiba_A is priced at $250? Is this better or worse than the initial price of $300?

The expected profit for Toshiba_A when priced at $250 is $7040 which is worse than the 8610 when priced at $300.

Now, for the second step, use `profit2()` to calcuate Toshiba's profits, assuming Toshiba_A's price can range from $100 to $500, in $10 increments. I.e., calculate profits asssuming Toshiba_A's price is 100, 110, 120, ..., 490, 500. We choose this range because Toshiba_A initial price is $300 and the adjacent price attribute levels are $100 and $500. We use $10 price increments to keep matters simple.

In your code, report the following: a) a plot of Toshiba's profit (Y) vs. Toshiba_A's price (X), b) the maximum profit obtained across all trial prices, and c) the profit-maximizing price.

```
pxs <- seq(100, 500, by = 10)
pft <- rep(0, length(pxs))
for (i in 1:length(pxs)) {
  pft[i] <- profit2(lm_res, prods1, 2, pxs[i]) #profit for given price
}
plot(pxs, pft, xlab = "Toshiba_A's price", ylab = "Toshiba's profit")
```




```
#maximum profit
max_profit <- max(pft)

#profit maximizing price- finds the price (pxs) of the max profit (pft)
max_profit_index <- which.max(pft)
profit_max_price <- pxs[max_profit_index]

cat("Maximum profit: $", round(max_profit, 2), "\n")
```

```
Maximum profit: $ 9620
```

```
cat("Profit-maximizing price: $", profit_max_price, "\n")
```

```
Profit-maximizing price: $ 350
```

Discussion:

- What is the profit-maximizing price for Toshiba? How much profit will it make at this price? How much more profit does Toshiba get, compared to the initial price of $300?

The profit maximizing price for Toshiba is $350, and they will make $9620 at this price.

- What do you notice about the shape of the profit function? Is it continuous (smooth) or discontinuous (has "jumps")?

The shape is discontinuous because once the price reaches a certain level, the profit will begin to drop as people wont want to pay that high of a price. After the price reaches $350 the profits begin to decline overall in this graph.

3.1) Simulation of iPad vs Toshiba_B

Building upon the prior work, we will now evaluate the second initial design under consideration, Toshiba_B.

First, evaluate Toshiba's expected profit assuming it produces Toshiba_B instead of Toshiba_A, assuming Toshiba_B's price is unchanged from its initial level (as in `prods2`.)

```
profit_toshiba_B <- profit1(lm_res, prods2, 2)
print(profit_toshiba_B)
```

```
[1] 9840
```

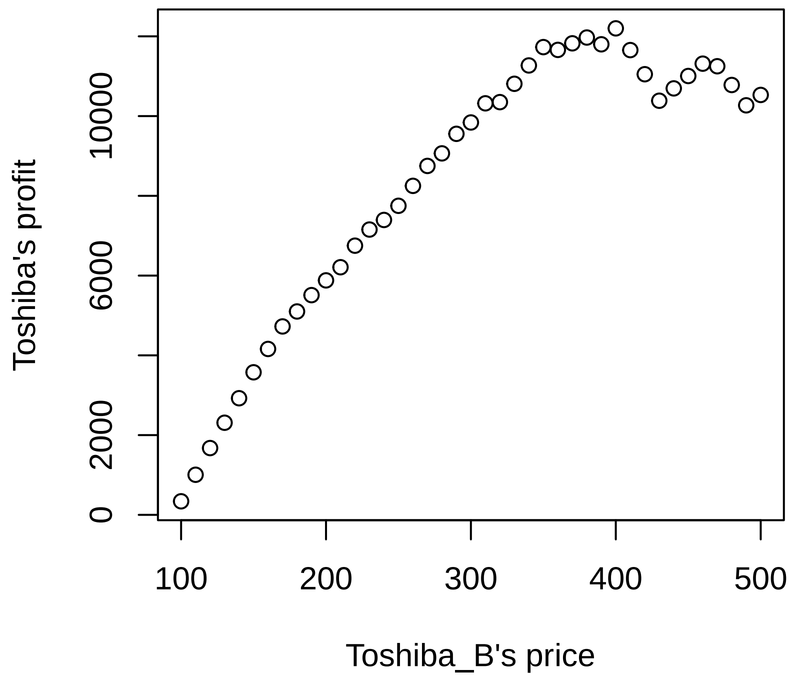
Discussion:

- How much profit will Toshiba make when it launches Toshiba_B (at a price of $300)? Is this better or worse than it can do with Toshiba_A?

Toshiba will make $9840 when it launches product B which is better than it can do with A (8610)

Next, determine the profit-maximizing price for Toshiba_B, considering prices from $100 to $500 in $10 increments.

```
pxs <- seq(100, 500, by = 10)
pft <- rep(0, length(pxs))
for (i in 1:length(pxs)) {
  pft[i] <- profit2(lm_res, prods2, 2, pxs[i]) #profit for given price
}
plot(pxs, pft, xlab = "Toshiba_B's price", ylab = "Toshiba's profit")
```




```r
#maximum profit
max_profit <- max(pft)

#profit maximizing price- finds the price (pxs) of the max profit (pft)
max_profit_index <- which.max(pft)
profit_max_price <- pxs[max_profit_index]

cat("Maximum profit: $", round(max_profit, 2), "\n")
```

```
Maximum profit: $ 12200
```

```r
cat("Profit-maximizing price: $", profit_max_price, "\n")
```

```
Profit-maximizing price: $ 400
```

Discussion:

- What is the profit-maximizing price for Toshiba when it launches Toshiba_B to compete with the iPad? How much profit will it make at this price? How much more profit does Toshiba get, compared to launching Toshiba_A at its optimal price?

The profit maximizing price for Toshiba B will be $400 with a profit of $12,200. This is $2580 better than product A which has a maximum price of 9620.

3.3) Full optimization of product design

We began the workshop by assuming Toshiba was only considering two potential designs, Toshiba_A and Toshiba_B. We now relax that assumption and allow Toshiba to search over all potential product designs for the profit-maximizing design.

To do this, we take the following steps:

1. Enumerate all possible designs that are feasible for Toshiba, assuming prices can only attain the pre-specified levels in the initial product design (i.e., levels of price in `design_DF`).
2. Loop over all possible designs, computing profits assuming each candidate design competes (only) with the iPad (as specified in `prods1` and `prods2`).
3. Taking the profit-maximizing design from (2), attempt to further optimize prices for the design. Test prices at $10 increments, up to a maximum of $700.

3.3.1 Enumerate all possible designs

For the first step, a useful R function is `expand.grid()`. `expand.grid()` creates a dataframe that contains all possible combinations of factor variables. The input to `expand.grid()` is a series of lists/vectors that contain the feasible attribute levels, one list per attribute. For example, to form all possible combinations of the numbers {1,2} and the letters {"a","b"}:

```
expand.grid(c(1,2),c("a","b"))
```

```
  Var1 Var2
1    1    a
2    2    a
3    1    b
4    2    b
```

Below, create a dataframe called `allprods_DF` that contains all feasible designs that Toshiba can produce. Note that Toshiba CANNOT produce products with the iOS operating system.

HINT: The `unique()` function can be useful to generate a list of valid attribute levels, e.g. `unique(design_DF$Screen)` will return the valid levels for screen size (7 and 10).

```
# Create lists of valid attribute levels for each attribute
screen_levels <- unique(design_DF$Screen)
cell_levels <- unique(design_DF$Cell)
price_levels <- unique(design_DF$Price)
battery_levels <- unique(design_DF$Battery)
os_levels <- unique(design_DF$OS)[unique(design_DF$OS) != "iOS"]

# Generate all feasible combinations of attribute levels
allprods <- expand.grid(Screen = screen_levels, Cell = cell_levels,
                        Price = price_levels, Battery = battery_levels, OS = os_levels)

# Create a dataframe with the same column names as prods1 and prods2
allprods_DF <- data.frame(Screen = allprods$Screen, Cell = allprods$Cell,
                           Price = allprods$Price, Battery = allprods$Battery, OS = allprods$OS)


print(allprods_DF)
```

	Screen	Cell	Price	Battery	OS
1	7	Y	300	12	Windows
2	10	Y	300	12	Windows
3	7	N	300	12	Windows
4	10	N	300	12	Windows
5	7	Y	100	12	Windows
6	10	Y	100	12	Windows
7	7	N	100	12	Windows
8	10	N	100	12	Windows
9	7	Y	500	12	Windows
10	10	Y	500	12	Windows
11	7	N	500	12	Windows
12	10	N	500	12	Windows
13	7	Y	300	8	Windows
14	10	Y	300	8	Windows
15	7	N	300	8	Windows
16	10	N	300	8	Windows
17	7	Y	100	8	Windows
18	10	Y	100	8	Windows
19	7	N	100	8	Windows
20	10	N	100	8	Windows
21	7	Y	500	8	Windows
22	10	Y	500	8	Windows
23	7	N	500	8	Windows
24	10	N	500	8	Windows
25	7	Y	300	4	Windows
26	10	Y	300	4	Windows
27	7	N	300	4	Windows
28	10	N	300	4	Windows
29	7	Y	100	4	Windows
30	10	Y	100	4	Windows
31	7	N	100	4	Windows
32	10	N	100	4	Windows
33	7	Y	500	4	Windows
34	10	Y	500	4	Windows
35	7	N	500	4	Windows
36	10	N	500	4	Windows
37	7	Y	300	12	Android
38	10	Y	300	12	Android
39	7	N	300	12	Android
40	10	N	300	12	Android
41	7	Y	100	12	Android
42	10	Y	100	12	Android
43	7	N	100	12	Android
44	10	N	100	12	Android
45	7	Y	500	12	Android
46	10	Y	500	12	Android
47	7	N	500	12	Android
48	10	N	500	12	Android
49	7	Y	300	8	Android
50	10	Y	300	8	Android
51	7	N	300	8	Android
52	10	N	300	8	Android
53	7	Y	100	8	Android

```
54      10     Y    100        8 Android
55       7     N    100        8 Android
56      10     N    100        8 Android
57       7     Y    500        8 Android
58      10     Y    500        8 Android
59       7     N    500        8 Android
60      10     N    500        8 Android
61       7     Y    300        4 Android
62      10     Y    300        4 Android
63       7     N    300        4 Android
64      10     N    300        4 Android
65       7     Y    100        4 Android
66      10     Y    100        4 Android
67       7     N    100        4 Android
68      10     N    100        4 Android
69       7     Y    500        4 Android
70      10     Y    500        4 Android
71       7     N    500        4 Android
72      10     N    500        4 Android
```

```
nrow(allprods_DF)
```

```
[1] 72
```

Discussion:

- How many possible designs are feasible for Toshiba?

72 designs

3.3.2 Loop over all possible designs, computing profits

Now loop over the possible Toshiba designs, computing profits assuming the design in question competes with the iPad (Screen=10,Cell="Y",Price=500,Battery=8,OS="iOS"). Store each profit value in a vector/list at each iteration of the loop.

```
prod <- rep(0, nrow(allprods_DF))

for (i in 1:nrow(allprods)) {
  prods_DF <- rbind(iPad, allprods [i,])
  prod[i] = profit1(lm_res, prods_DF, 2)

}

print (prod)
```

```
 [1] 14399 15288  9522 12928 -1144 -1674   497   160 16254 23684 11396 14472
[13] 11229 14784  7161  9328  -240  -696  1037   792 10322 15680  6672  9064
[25]  5454  6698  3996  4123   104  -195   704   714  4422  4764  2954  3336
[37] 10450 12395  8610  9840  -750 -1215   610   340 12870 15785  7380 10530
[49]  8200  9750  5720  7310     0  -345  1000   810  4400  7900  2940  5395
[61]  3075  4400  2475  2860   185     0   675   620  2430  2400  2125  2520
```

```
max(prod)
```

```
[1] 23684
```

Discussion:

- What is the profit-maximizing design for Toshiba, assuming prices are unchanged from the levels in the candidate designs? How much profit will Toshiba make with this design?

TBD

3.3.3 Further optimize prices for the design

Finally, attempt to improve prices using the preferred product design from the previous step. Test prices at $10 increments, up to a maximum of $700.

Discussion:

- What is the profit-maximizing price for Toshiba when it launches the globally optimal design to compete with the iPad? How much profit will it make at this price? How much more profit does Toshiba get, compared to launching Toshiba_B at its optimal price?

TBD

- Summarize your product design recommendation for Toshiba

TBD