MGSC 670 - Revenue Management

The Retail Markdown Game

Andrea Irina Yzeiri

Maguette Paye

Marek Krowicki

Shivangi Soni

Vivek Saahil

March 17th, 2021

Master of Management in Analytics

Revenue Management

McGill University

Contents

1 Introduction

Revenue Management in the retail industry is comprised of a number of crucial factors such as budget, product price, and demand forecasting, and strategies outlining how to implement markdowns and where original prices should be reduced to increase sales over time. Unlike promotional discounts, which are mostly temporary and target customer-segment strategies, a markdown is when a retailer permanently lowers the price of a product with the intention of driving sales. Overbuying is one of the major concerns in the retail industry and markdowns are specifically used to eliminate this problem and increase revenue.

This report employs *The Retail Markdown Game* to learn the best way to manage markdowns by testing out different algorithms.
The outline for the Retail Markdown Game is as follows:

Source Link: http://www.randhawa.us/games/retailer/nyu.html

Objective: To develop a generic markdown pricing strategy for a retailer **to maximize the total revenue** when selling some inventory over a limited time period (i.e. at the end of week 15).

Given Data:

- The initial stock is 2000 units
- The price is set at $60 for the first week
- The stock left-over at the end of the 15 weeks is lost i.e. there is no salvage value
- There are no other costs involved since the production costs for the items are already sunk

The historical sales data provided (Sales-Data.xlsx) includes four columns:

- **Week:** The season from week 1 to week 15 (i.e. 14 decisions)
- **Price:** The price charged on a given week in $
- **Sales:** The amount sold on a given week
- **Remaining Inventory:** The inventory remaining at the end of the week

2 Data Extraction, Pre-processing and Exploration

Data was extracted directly from the website using the Selenium library in python (see scraper_bike.py). First, a script was run to generate a list of all possible combinations of button presses. 60 was inserted at the beginning of each combination as the first decision is always predetermined, then each was indexed from 0 to 679 for a total of 680 combinations (see combos.csv).

Next, the site was scraped by automating the button presses according to the combinations list and saving the resulting table. 50 iterations were run for each combination, for a total of 34,000 trials. The iterations had to be run in chunks, as there would be significant slowdown after around 1000 iterations requiring a restart of the IDE. The results are stored in "raw_data.csv".

Finally, the raw data was separated into trials and results (see reps_only.csv and results_only.csv). A separate script (see data_transform.py) was used to return only the trials and results of the iterations that resulted in a difference between the revenue and optimal strategy of less than 5% (see reps_less_than_5_percent.csv and results_less_than_5_percent.csv). These were deemed the "good" results and were analyzed further to derive a markdown strategy.

3 Algorithm

Three algorithms were tested in this analysis.The first was dubbed the "Matrix Approach", where the decision rules were determined on a week by week basis according to the scarping results. The second used linear optimization to determine a general decision rule based only on the initial demand in week 1. The final algorithm utilized the difference between accumulated units sold and accumulated average units to be sold with a percentage thresh hold to indicate a markdown by week, dubbed "Mean Difference Algorithm".

3.1 Matrix Approach

The first algorithm was derived from the scraped data of trials with a difference less than 5%. The median sales were calculated for each week just prior to a change in pricing being triggered. There were six possible decisions for the lowering of the price each week, and the median was calculated for each of these conditions to determine the "trigger point". Should the sales fall below this point, the corresponding price change would be triggered. The price would be maintained otherwise. Some accommodations had to be made, as certain price changes had very little data (such as \$60 to \$36, and \$48 to \$ 36). The trigger point for these was set arbitrarily low, as the calculated values were observed to perform poorly. The decision thresholds are displayed in the table below (for rough work, see matrix_approach_work.xlsx):

Week	60 to 54	60 to 48	60 to 36	54 to 48	54 to 36	48 to 36
1	83	67	35	N/A	N/A	N/A
2	84	69	35	83	42	60
3	88	73	35	90	58	60
4	91	81	35	94	60	60
5	96	75	35	104	83	60
6	101	89	35	108	71	60
7	107	88	35	112	84	60
8	110	111	35	120	73	60
9	119	100	35	127	86	60
10	121	107	35	125	83	60
11	188	116	35	135	71	60
12	122	126	35	139	71	60
13	135	120	35	145	71	60
14	133	124	35	148	71	60

Table 1: Decision Threshold

This decision methodology was then implemented in a python script simulating 1000 trials on the website (see matrix_version_sim.py). The results are displayed below:

Mean:	6.9649%
Standard Deviation:	5.8897%
95% Confidence Interval:	(6.5994%, 7.3304%)

3.2 Linear Optimization

The problem can be set up as a Linear Optimization problem, where the objective is to maximize the revenue. Hence, the objective function is the sum of the product of price at which an item is sold, average weekly demand at each price point (product of average demand lift and weekly initial demand at full price), and number of weeks an item is sold at a respective price. Historical sales data was used for running the linear optimization model. The decision variables in this case are the number of weeks the item is sold at each price ($60, $54, $48, $36). First, the weekly average demand at each price for each item was calculated using the sales data. It was assumed here that sales is equal to demand. However, it is important to note that if the inventory was sold out before fifteen weeks, the sales did not imply demand, hence, the weeks where the items were sold out were not included while taking average. Since the historical data incorporated trials where the price was only switched once, from $60, it was easy to determine the demand jump once the price dropped to either $54, $48, or $36. The demand jump for each item was determined using the weekly average demand values calculated previously. It can be seen in Figure 1 that demand jump (lift) with respect to price is homogeneous across items. After this, the average of the demand jump for each price drop and incremental jump in demand lift was determined.

The results can be seen in the table below (Refer to Sales-Data.xlsx to see calculations):

Item	Price				Demand Jump	Average Demand Jump	Incremental Jump
	60	**54**	**48**	**36**			
1	58	76			1.3	1.31	31%
2	108	144			1.34		
3	59	82			1.39		
4	61	78			1.27		
5	93	114			1.23		
6	114		209		1.83	1.73	33%
7	67		120		1.77		
8	53		97		1.83		
9	74		132		1.79		
10	67		97		1.44		
11	100			264	2.63	2.81	62%
12	64			189	2.94		
13	66			197	3		
14	61			164	2.67		
15	62			175	2.81		

Table 2: Analysis on the Historical Sales Data

The linear optimization problem can be formulated as below:

$$Maximize\ (60 * 1.00 * d_{60} * x_{60}) + (54 * 1.31 * d_{60} * x_{54}) + (48 * 1.73 * d_{60} * x_{48}) + (36 * 2.81 * d_{36} * x_{36})$$

where

$$d_{60} = weekly\ demand\ at\ \$60$$

$$x_{60} = The\ number\ of\ weeks\ the\ item\ is\ sold\ at\ \$60\ and\ x_{54},\ x_{48},\ and\ x_{36}\ are\ defined\ similarly$$

subject to

Total-sales can not exceed 2000 units:

$$d_{60} * x_{60} + 1.31 * d_{60} * x_{54} + 1.73 * d_{60} * x_{48} + 2.81 * d_{36} * x_{36} \leq 2000$$

Selling season can not exceed 15 weeks (less than 15 weeks in case the items get sold out early):

$$x_{60} + x_{54} + x_{48} + x_{36} \leq 15$$

Item will be sold at full price, $ 60 for at least one week :

$$x_{60} \geq 1$$

Non-negativity constraints for rest of the decision variables:

$$x_{54}, x_{48}, x_{36} \geq 0$$

This decision methodology was then implemented in a python script simulating 1000 trials on the website (see matrix_version_sim.py). The results are displayed in the table below. The results from Linear Optimization model are further discussed in the "Final Strategy" section.

Mean:	3.6272%
Standard Deviation:	2.8587%
95% Confidence Interval:	(3.4502%, 3.8042%)

3.3 Mean Difference Algorithm

The final algorithm tested incorporated insights provided from research and by playing the game. First, to sell all 2000 units, the average number of units sold over fifteen weeks would have to be about 133 units a week. However, this alone did not provide the best basis to compare the difference to current units sold a week as it would not take into account past weeks where units were undersold compared to the average or oversold. Therefore, the accumulated number of units sold (133 for week 1, 266 for week 2, ...) was calculated by week, as well as the accumulated number of raw units sold. The difference between the later to the former was calculated by week and then we ran some tests to see the sensitivity of this difference to markdowns. As weeks passed, sensitivity increased, and for that reason the *Markdown Percentage Threshold by Weeks* table shows that weeks twelve to fourteen have very small thresholds so to remove the final remaining inventory. Each threshold was calculated based on trends and testing to determine the optimal difference by week for a markdown to occur.

It is important to note that there is a special case, where the model had to take into consideration that if the algorithm indicated a ten percent markdown more than once, that the second time indicated that the model should go to a twenty percent markdown. In all other cases, the next markdown would not occur unless indicated by a markdown jump.

Markdown	**W1**	**W2**	**W3**	**W4**	**W5**	**W6**	**W7**	**W8**	**W9**	**W10**	**W11**	**W12**	**W13**	**W14**
10%	0.45	0.45	0.45	0.45	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.035	0.035	0.035
20%	0.55	0.55	0.55	0.55	0.42	0.42	0.42	0.42	0.42	0.42	0.42	0.07	0.07	0.07
40%	0.74	0.74	0.74	0.74	0.62	0.62	0.62	0.62	0.62	0.62	0.62	0.15	0.15	0.15

Table 3: Markdown Percentage Threshold by Weeks

Mean:	6.6010%
Standard Deviation:	4.3546%
95% Confidence Interval:	(6.3308%, 6.8712%)

4 Final Strategy

The results of testing all three algorithms indicate that the optimal final strategy to employ would be Linear Optimization. It demonstrated that that even though the price breakdown was not simulated at the highest granularity possible (strategy was only tested at $50, $70, $90, and $110), the results were still better as compared to other approaches, which is why linear optimization was chosen as the best strategy.

For each run of the *Retail Markdown Game*, the Excel solver in *Sales-Data.xlsx* should be used to determine how long to sell units at the original price point. The final strategy is displayed in the table below:

Demand at full Price ($)	**x60**	**x53**	**x48**	**x36**	**Revenue**
40	1	0	0	14	$59,039.84
50	1	0	0	14	$73,522.01
60	1	0	7	7	$81,557.26
70	1	0	11	3	$89,592.51
80	1	1	13	0	$97,350.92
90	1	7	7	0	$102,104.70
100	1	12	2	0	$106,858.48
110	5	10	0	0	$111,038.04
120	10	5	0	0	$114,878.88

Table 4: Final Strategy

5 Conclusion

In the retail industry the implementation of markdowns to optimize profit and unit sales is imperative for the practice of revenue management in this field. Different strategies are employed and tested by managers to reach this point. This report outlines the effectiveness of different algorithms on *The Retail Markdown Game* and simulates trials to determine the best algorithm to employ that has the lowest difference between revenue with perfect information and the algorithm's revenue.

The analysis demonstrated that the best algorithm to employ is "Linear Optimization" over both the "Matrix Approach" and the "Mean Difference Algorithm". The former provided the lowest average difference between its generated revenue and the perfect information revenue, with a steady confidence interval. As the literature would suggest, this approach is a standardized method that is ideal despite not taking into consideration different consumer trends or varying sensitivity to unit stock.