

DSO 570: The Analytics Edge

Data, Models, and Effective Decisions

Final Project

**Contents**

**Executive Summary……………………………………………………………………………...3**

**Opportunity for Improvement……………………………..………………………………....3-4**

**Optimization Methodology…………………………………………………………………....4-7**

**Optimization Results…………………………………………………………………………..7-9**

**Discussion**

1. **Appropriateness of Methodology………………………………………….. 9-10**
2. **Final Recommendation……………………………………………………. 10-11**

**Technical Appendix**

**A1) Mathematical Formulation……………………………………………………11**

**A2) Discussion of Technical Details……………………..………………….…..12-13**

**Executive Summary-**

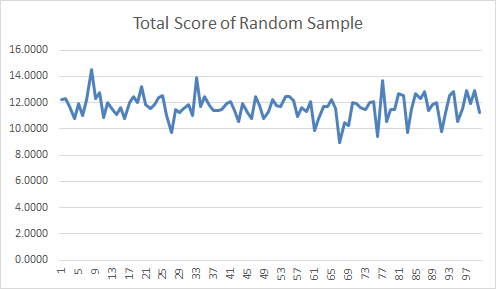
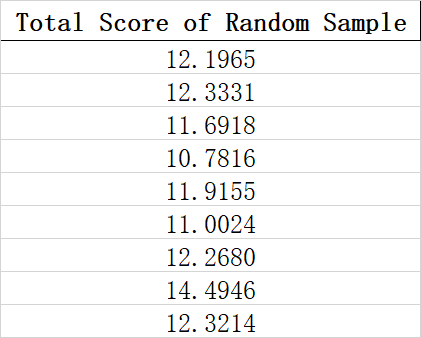
Being a subsidiary of PepsiCo, Frito-Lay owns 35 brands, including Lay’s, Fritos, Cheetos, etc., and it manufactures and sells chips and other snack foods. Each brand offers a combination of different flavors, packaging and sizes, and the number of total SKUs exceeds 1400. The US market is divided into 10 regions, and in each region, Frito-Lay operates its own sales team. Given the diverse product offerings, Frito-Lay wants to consolidate SKUs for each region. This project develops prototype optimization analysis and provides data-driven insights to help regional sales managers better select SKUs.

The current process by which directors of regional sales teams select SKUs is highly manual and based on the regional team’s intuition. This strategy is inefficient with respect to money and time, because the company needs to hire employees to execute it, and the results can be subjective and highly variable. Therefore, we felt it would be valuable to develop an optimization tool for more objective and product quality-driven results. The data in our model now are mainly from the Southwest region, which can be used to represent the operational data from all regions. We used these data to calculate a quality score for each product and we recommend that Frito-Lay use our tool to help with SKU selection and distribution. Our analysis is mainly based on small format products, or “eaches”. The company views its competitive edge as being able to service small stores across the nation, so the small format business is very important.

**Opportunity for Improvement-**

The current main weakness of small format SKUs selection is that the directors of regional sales teams select SKUs manually, which is based on the regional team’s intuition. Therefore, we conducted a simulation analysis to verify the necessity of change to the current product selection process.

The each module in distribution centers can support 250 distinct SKUs at a time. Effectively, each sales region can only support about 250 SKUs in small format. We did not have Frito Lay’s selection list, so we simulated it by randomly choosing 250 products and used our optimization tool to get a sum score for these 250 products (score is an evaluation method that we created to measure product quality). We cross validated 100 times to get an average score, which control the variance. The following chart shows us the distribution of the 100 scores, with the average score being approximately 11.68. However, if we use the optimization tool on all products and get the “250 product” list from our model, the average score will be 21.91. The difference is 10.23 and we improve efficiency by 88 %. We identify this as a huge opportunity to improve.

**Optimization Methodology-**

**Input Data:**

First, our input data can be divided into two parts. First part is the product quality score data, which are generated from the history data provided by Frito Lay, including sales, return, estimated margin, total distribution cost and platform capacity of each product. Second part is the constraints related data that can be input by the users, which include customized weights in quality score calculation, minimum requirement on innovation products, minimum requirement on each category and minimum requirement on each subcategory.

We then process the data to make it ready for objective functions. We used Z-scale to get a score between 0 and 1 for each record in each variable. We gave each variable a weight (which can be assigned by users) as shown in the table below.  Basically, the weight represents the importance of each variables’ effect on product quality. In our sample model, we set sales as the base of the score and assign weight 1. Returns, distribution, and over-capacity are the cost that are negatively correlated with the score. Margin, indicating profit, are positively correlated with the score. Here, we regard return cost as more important compared to the other factors when determining product quality, so the weight in the sample is set to be 20%. And for capacity, since its values are binary (0/1), we assign a relatively small weight to it. Based on this logic, users could assign weights based on understanding.



Now, all product data are ready for the optimization tool.

**Output of the optimization:**

The output consists of the 250 unique BDC’s with the highest total product quality score under the constraints. Sample outputs will be given later in this report.

**Decision Variables, Objectives, and Constraints:**

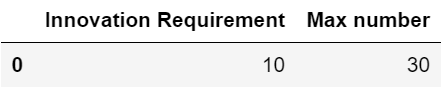
Our goal is to maximize the total quality score, and the decision variables are whether to select product i in the specific region’s market, or in another word, which BDC to carry in each sale region. The score’s formulation is shown as follows, and each product has its own score:



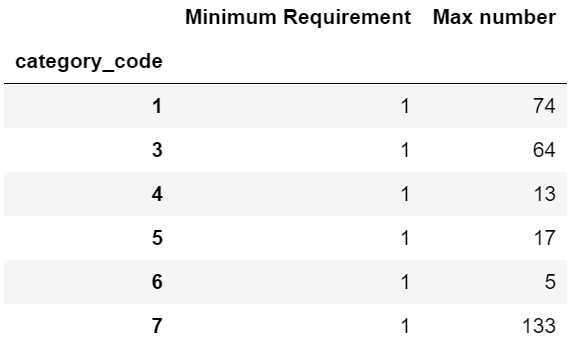
We implemented our optimization tool under the following four constraints.

First: small format quantity constraint. The eaches module in distribution centers can support 250 distinct SKUs at a time. Effectively, each sales region can only support about 250 SKUs in small format. Thus, the quantity constraint ensures our model outputs only 250 items.

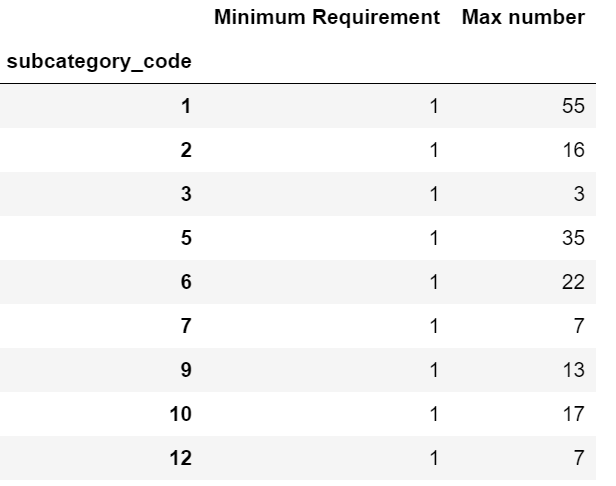
Second: innovation constraint. In the optimization, we need to include at least *X* number (decided by the user) of 2018-innovation products. In the example we provide, we set the minimum number of innovation products to be 10 and “Max number”, which is the upper bound a user can input, to be 30.



Third: category constraint. In the optimization, we need to include at least Ynumber (decided by user) of products for each category. In the sample, we assign one to all categories, meaning that our optimal choice should include all categories. “Max number” is the upper bound a user can input for each category.



Fourth: subcategory constraint. We need to include at least *Z* number (decided by user) of products for each subcategory. The Subcategory table is similar to the category table. An example is as follows:



**Usage:**

In the input dataset, there are a total of 7 sheets users should fill to use the tool.

1. **Product**. In this sheet, users should include data about sales, returns cost, total distribution cost, margin rate, etc. These data are generated from original data sets as row data need to be processed before   into objective function.
2. **Product\_Category**. In this sheet, users should specify each BDC belongs to which product category.
3. **Product\_Subcategory**. In this sheet, users should specify each BDC belongs to which product subcategory.
4. **score\_weight**. When the optimization tool calculates product quality score, each kind of component will need a weight. In this sheet, users can assign weights to each kind of component based on company strategy or other factors that they want to consider.
5. **category\_req**. In this sheet, users can specify that for each category, at least how many products have to be included. In this way, we can satisfy different customer need and be more well-known in various fields of market.
6. **subcategory\_req**. In this sheet, users can specify that for each subcategory, at least how many products have to be included. This sheet serves a similar function as “category\_req”
7. **inno\_req**. In this sheet, users can specify the minimum number of innovative products to be included in our optimal choice and can be reassigned by users.

When users fill the input-spreadsheet based on these instructions and format, they will be ready to use the optimization tool. All they need to do is run the tool, and the output data will be generated automatically. In the output excel file, the user will have a list of BDC numbers which are selected by the optimization tool. By selecting these products, the user will be guaranteed a maximized overall sum of product scores. (If you input other regions’ data into this optimization tool, you will get the optimal choice of products for those regions).

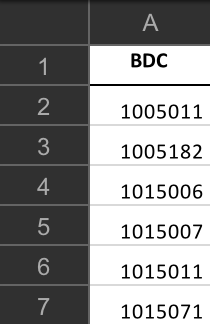
**Optimization Results-**

The results of our optimization are based on a few underlying assumptions, which we feel are both fair and intuitive. In order to utilize our tool, the user should be aware of these assumptions and how they affect the output

We hope the preceding section, *Optimization Methodology*, brought these points to light via a thorough explanation of our modeling procedure. Additional assumptions will be addressed in the technical appendix.

Perhaps the best feature of our tool is its simplistic output. While the methodology on the back-end is rooted in linear programming and mathematical formulae, the front-end is, ultimately, an n-dimensional vector. Each n, i.e. each row, contains the BDC code of a unique product which our tool has deemed “worthy” for selection. By “worthy” we mean the product who has their BDC in our output vector has specific qualities, such as sales returns, margin, and capacity, which combine to quantify the product’s value to Frito Lay in a positive way.

Below, we have an abbreviated sample output.

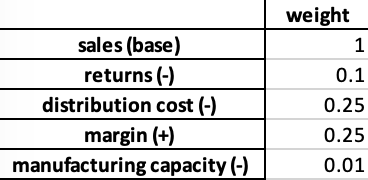
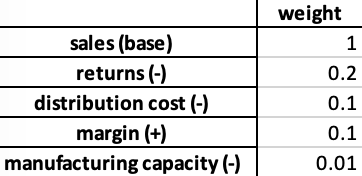


We understand that our domain knowledge lies in the field of analytics and statistics, not product management. For this reason, we chose not to order the BDC’s in terms of potential gain, potential loss, or any other metric. In order for our tool to be most effective and easy to use, we aim to introduce as little bias as possible. Although bias is tough to avoid in any type of statistical modeling, we feel a raw list output to be the most appropriate. In this form, regional sales directors can use their own analysis process, combined with ours, to make final decisions on SKU selection. Ultimately, our aim is not to make decisions for Frito Lay, but to provide them with actionable insights and statistically grounded recommendations.

Another excellent feature of our output is its flexibility. No matter how large or small the input file is, the output will always be an n-dimensional vector. Thus, complicated input procedures will always be boiled down to a simplistic output.

The simplicity and ease-of-use is simple to qualify. But at its core, our tool should help Frito Lay make decisions in order to increase revenue. A user-friendly output is only as valuable as the mathematics behind it, and the decisions made because of it.

When considering the output from our tool, one should recognize the output as heavily dependent on the pre-defined “score\_weight” inputs. Though our tool reads in a file, part of that file should be manipulated and cross-validated by the user. For example, below are two unique “score\_weight” inputs which could yield entirely different results:

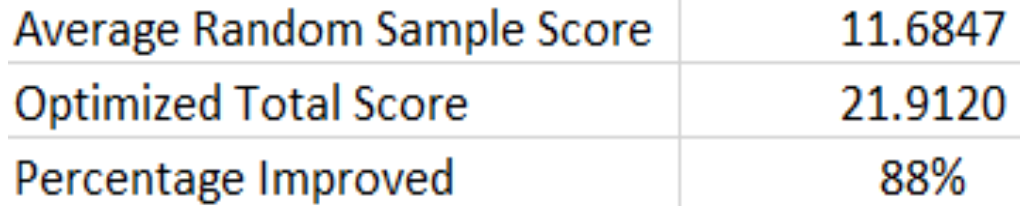
 

The weights represent, at a high level, which parts of the revenue cycle and supply chain system the user considers more or less important. If returns are a large detractor, then more weight should be assigned, etc. The numbers seen above are arbitrary and should be adjusted with attention.

By adjusting weights, the user can perform their own type of A/B testing with regards to SKU selection. If, in some scenario, distribution cost became virtually non-existent, then a weight of zero might be appropriate. To generalize, if and when progress and change occurs within Frito Lay, our tool can be adjusted to accommodate some of these changes! To be clear, our tool is not a one-size-fits-all optimization technique which can adjust to any kind of external factor. However, reasonable flexibility is certainly something we took into account when designing our tool.

Another use can be described as a type of “grouping.” If the user wanted the SKU’s who had the highest margin, while still satisfying all other pre-determined requirements, the margin weight could be given heavy inflation. The resulting output would then list all relevant products with the highest margin! The output file could then be interpreted in several ways. One, which we feel is an exciting option, is to take the output list and run it through further statistical modeling and testing. In this way, our tool can be utilized as a filter, which can automatically remove any product which fails to provide benefit in a certain category, such as margin or sales.

We will now provide a brief example of how our tool can outperform standard SKU procedures, which are based on simple random samples of 250 selected SKU’s. We have run a simulation, which was briefly touched upon on page 3, in which 100 iterations of 250 selected SKU’s are chosen. These products then have their respective scores calculated, and the resulting sum is then achieved. Over 100 trials, we have averaged the product score sums to be roughly 11.7. Meanwhile, we run our tool once, on standard constraints, and achieve an output score of approximately 22. These results are summarized below, and given our confidence in the product score’s correlation to Frito Lay success, we find the 88% increase so be a satisfactory baseline result.



**Discussion-**

1. **Appropriateness of Methodology**

As mentioned previously, we feel our tool is simple and un-biased. These are the two main reasons why we believe our proposed methodology is appropriate.

Our input data is comprehensive, concise, and captures the essential numbers pertaining to each unique product. Our objective function and constraints, which will be detailed in the technical appendix, utilize each variable in the “Product” tab of our input data, and our optimization technique iterates through the indexes of each of the subsequent tabs in the input data. To be short, our methodology selects what we feel is necessary, and then makes use of it. There are no frills, and each piece of data is employed.

In regard to the input data specifically, we make use of both categorical and numerical variables. The categorical variables, such as “pc0.95” and “innovation2018” have been made binary, with the usual 1 representing “on” or “yes” and the 0 representing the alternative. We feel this makes the input data cleaner, and the modeling and optimization process more efficient.

At the heart of Frito Lay’s revenue are sales, returns, distribution costs, and margins. We have included all of these variables in our input data, and as mentioned in the previous section, we provide a “weights” tab which enables the user to place emphasis where he or she sees fit.

From this comprehensive input data, our tool then calculates a “score” for each product. This score, multiplied by a binary decision variable, constitutes our objective function, which will be detailed in the technical appendix. In short, our tool maximizes the sum of the product scores for the combination of all products which meet our constraint requirements. Our objective function is succinct, and we feel it captures exactly what’s most important: helping Frito Lay executives make informed SKU selection.

As for constraints, we wanted to offer a diverse recommendation while minimizing bias. In order to do this, we created constraints to limit the overall number of products chosen while maximizing innovation, category, and sub-category requirements. In total, we required our output to contain a wide-range of SKU’s, while also keeping in mind quality of product and benefit to Frito Lay.

Since our optimization is based largely around this calculated “score,” it is important to note alternative approaches, which do not utilize this aggregate variable. A straightforward way to conduct this optimization would be to maximize sales and minimize costs. One could build the entire model around this concept, and the output it produces might be decent upon first glance. However, when diversity and minimum requirements enter the equation, one must be clever when determining which product is superior to another. For example, two distinct SKU’s might have the same sales numbers, the same distribution costs, and very similar profit margins. But if the goal of Frito Lay is to introduce a new product into a new market, maybe only one (or neither) of these competing SKU’s is satisfactory. A barebone, off-the-shelf model might miss this nuance, which is why we chose to build around the “score,” as well as diversity and innovation requirements. (It should be noted that if the user wanted to adjust these constraint requirements, a simple alteration to the input data would suffice.)

**2. Final Recommendation**

In this final recommendation section, we keep in mind domain knowledge and flexibility. With that being said, our recommendation is to utilize our tool in conjunction with existing technologies and expert opinions. We purposely do not rank our SKU selection in the output file. We leave it to someone in-house to do that. Our tool simply takes into account different requirements and builds a list of the best possible products which meet the needs of Frito Lay.

As for next steps, there are several directions we believe are appropriate. Since our tool can be scaled up or down in terms of input and output dimension, it can be utilized as both a filter and a decision-maker. Granted, both of these functions require intermediate steps, but we feel this flexibility is a huge bonus.

One possible course of action would be to use the tool as a filter, create unique and specific SKU lists, and then perhaps utilize these lists when testing pilot programs in different sales regions. For example, if Frito Lay wanted to test all SKU’s in the Southwest within a certain subcategory, while keeping the current SKU selection intact, our model can be manipulated to achieve such a goal. This concept ties back with the previously mentioned A/B testing idea, which serves as an excellent option when considering how to utilize this tool in the future.

Another route is to use the tool as a sort of weeding-out method. If the SKU’s in our model’s output are what we recommend, then it is appropriate to say the SKU’s not included are the products we do not recommend. If the user were to run the model several times, with the input constraints and weights slightly altered, similar SKU’s will be present in the output. Perhaps the user could keep track of how many times each product is showing up in the output and offer some possible solutions for the products who fail to show up. Again, these types of actions require specific domain knowledge, and we do not aim to infringe upon the knowledgeable professionals within Frito Lay. We merely want to arm them with a simple, powerful decision-making tool.

**Technical Appendix-**

**A1) Mathematical formulation**

Now let us define our decision variables, data variables, and our optimization formulation:

Xi: binary decision variable representing whether or not we include product *i* in our SKU list.

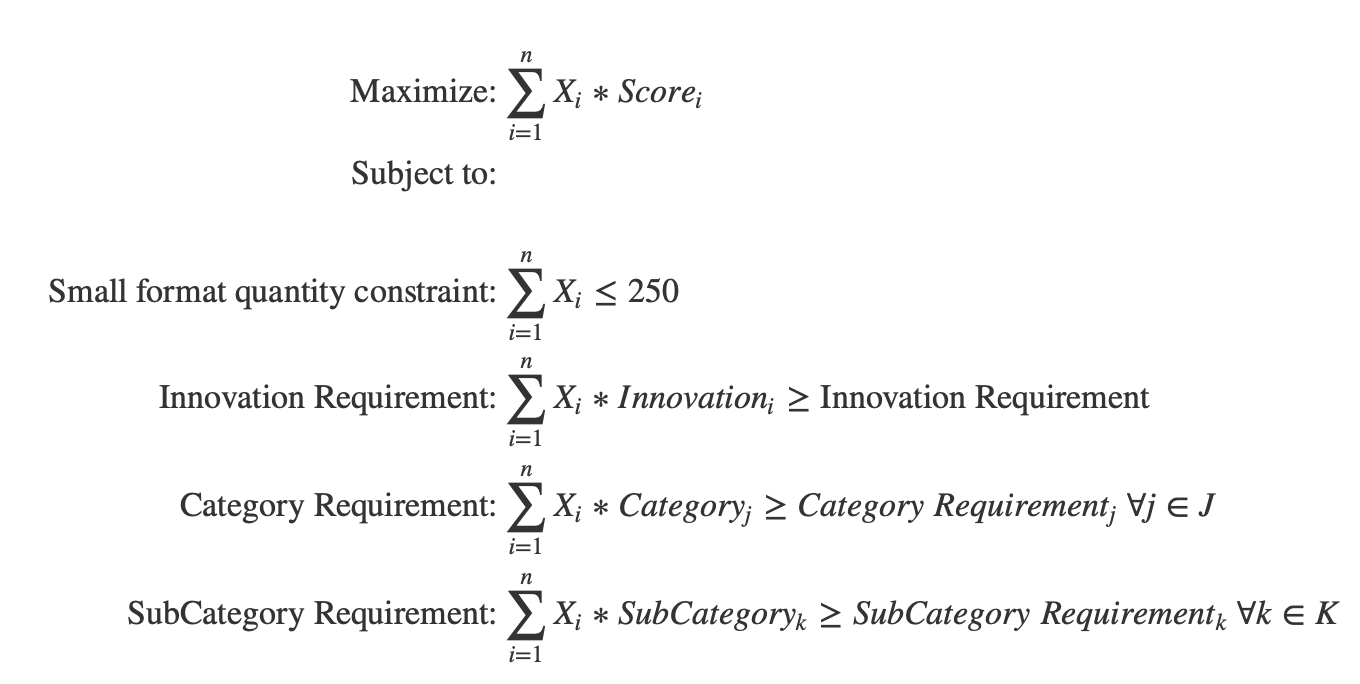
Scorei: the aggregated product score for product *i*, which we define below:



Innovationi: a binary variable representing whether product *i* is considered an “innovative” product, as defined for 2018 in our data set

J: set of Category requirements, with *Category Requirementj*being the jth category’s requirement

K: set of SubCategory requirements, with *SubCategory Requirementk* being the kth subcategory’s requirement



**A2) Discussion of Technical Details**

We will now address the rigor of our results, our underlying assumptions, and points which we feel can be improved in the future.

From a linear programming standpoint, our objective function and corresponding constraints are valid and appropriate. Non-linearity is not an issue, and our constraints are constructed such that upcoming trends and company dynamics can be efficiently accounted for with proper adjustment. This ability to adapt and flex to changing conditions should can be considered both a strength and weakness. In terms of strength, our formulation is built upon minimal constraints, and our objective function combined only essential metrics. By minimizing the moving parts, we enable our tool to be able to handle additional constraints and decision variables. In terms of weaknesses, our tool does not account for many external factors which certainly do play an important role in SKU selection. These factors could include flavor selection, seasonality, cannibalization, and competitor choice of SKU. Given more time, we could certainly perform some feature engineering and create dummy variables capturing these different factors. In turn, adjustments and additions to our objective function and constraints would take place.

While we tried to minimize bias as much as possible, it must be noted that our tool’s output is heavily dependent on the weight selections. This weight selection process can be considered a bias in its own right, and we will address it now.

Domain knowledge is a huge factor when considering weight selection. From our team’s point of view, we wanted to create sample outputs with very little rigor introduced to our weight selection. A seasoned Frito Lay executive will undoubtedly know more about the business than we do, and we anticipate their weight selection process to be thorough and informed. As was mentioned earlier in this report, the weight tuning process can act as a sort of statistical A/B test. Thus, it would not make sense for us to pre-define rigid weight values when part of our tool’s value comes from its flexibility on the user-end.

As for assumptions, our model is centralized around one key idea: the product score is a valid quantitative measure of product quality. As defined in the preceding section, our product score combines our weights and some of the key metrics involved with Frito Lay products in an aggregate manner. Clearly, the weight selection will play a huge role in defining the product score. Additionally, if some key product metric was left out of our formulation, then our objective function might miss out on an underlying trend. We are aware of this shortcoming, which is why we defined our product score to include several important product metrics.

Other key assumptions show up through our constraints. We assume minimum category and sub-category requirements, as well as a limit on total SKU selection. Since these constraints directly influence the output of our optimization tool, their adjustment will alter the output. However, we designed our tool to allow for painless adjustment of constraints, which can balance out the pitfalls of the assumptions to an extent. Naturally, some assumptions must be made in order to produce a working tool. We feel our assumptions are fair and realistic.

Thank you, and any additional questions/comments are welcome.