Optimization 1D

Fridge Stocking

1005154 Yong Zhe Rui Gabriel
1005351 Janani Sureshkumar

Engineering Systems & Design

1 Introduction

In land scarce Singapore, the battle for shelf space is ever prominent. In a bid to understand and better optimize the products to display or stock, our project will take a look at the scenario of stocking drinks in a fridge and how to best optimize it.

2 Understanding the Problem

The specific case we will be observing comes from Thirsty Craft Beer and due to a better delivery location, they have opened a 'cloud' fridge at Orchard and need to carefully select the type and number of products to stock in order to satisfy demand and maximize profit.

There are certain conditions (constraints) that we would need to keep in mind when stocking this fridge namely the following few:

- Demand: There is varying demand for each item and we need to satisfy the demand. The maximum amount of demand needs to be considered as well since some products have lower demand.
- Capacity: The fridge has a maximum amount that it can possibly stock.
- Variety: The fridge needs to stock a sufficient variety of craft beers as Thirsty prides themselves in their variety of craft beers
- Convenience: The items stocked in the fridge have to be convenient for accessing and accounting. Say the products are stocked in a row, it would not be ideal to place product A behind product B, as it would result in a lack of convenience to access product A
- Non-negativity and Integer constraints: Products can not be negatively stocked and must take integer values

3 Model Formulation

3.1 Key Assumptions

- Items are all Cuboids
- All items of the same type are of the same size
- Items are only of size $1 \times h$, where h is the height of the item
- If an item is placed, the whole row of items are the same. No situation exist such that product A is behind product B in a row.

3.2 Set-Up

The model works based on assignment of objects to specific coordinates (i, j) in a grid that is of height H and width W. This can be seen in the example shown in Figure 1, where the green dots represent assignment of the objects to coordinates $(3, 3)$ and $(7, 5)$.

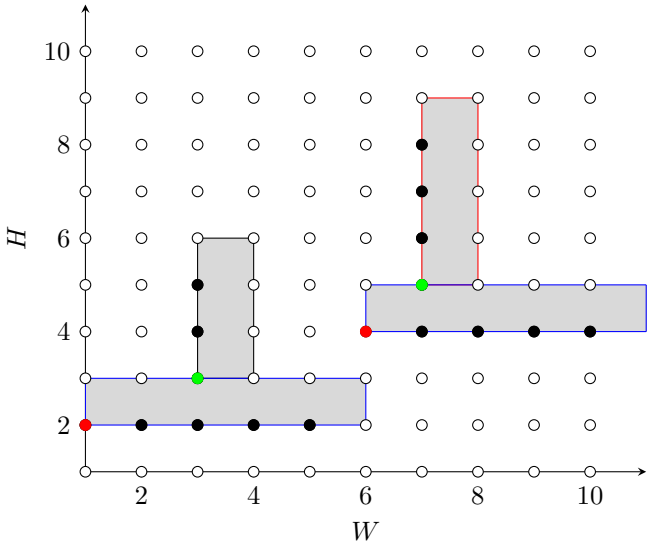

Figure 1: Item Grid

This decision to place an item k with height h_k in coordinate (i, j) is represented as the decision variable $x_{(i,j),k}$ and in this case $x_{(3,3),k} = 1$ and $x_{(7,5),l} = 1$ where k and l are two different types of items. This is represented by the green dots in the figure.

However, once an object is assigned to coordinate (i, j), it will take up space and other objects can no longer be placed in that area. Hence, another decision variable is required to disallow objects from being placed in coordinates that the object covers. This variable will be represented by $u_{(i,j),1}$.

In this case, due to the placement of the item at $(3, 3)$ as seen in Figure 1, $u_{(3,4),1} = 1$ and $u_{(3,5),1} = 1$. And in the case of the item placed at $(7, 5)$, we then need $u_{(7,6),1} = 1$, $u_{(7,7),1} = 1$ and $u_{(7,8),1} = 1$. These decision variables can be seen in Figure 1 as represented by the black dots.

Lastly, the final part of the model is that objects need to be placed on a shelf. That is represented by an object that has height 1 and width w_s units long. However, a shelf need not be placed across the entire width of the fridge. To account for that it is useful to introduce the idea of a grid for shelves. Continuing from the above example, the grid for shelves can be seen in Figure 2.

The decision variable on where to place shelves is $s_{(g,j)}$ where g in (g, j) is a coordinate in the shelf grid. In this case, $s_{(1,2)}$ and $s_{(2,4)}$ are 1. This works on the assumption that all shelves are of the same width, and the locations where shelves can be placed are evenly spaced. Another thing to note is that $s_{(2,4)}$ in the shelf grid corresponds to the coordinate $(6, 4)$ in the original grid since each shelf has a width of 5.

The variables $s_{(1,2)}$ and $s_{(2,4)}$ being 1 will thus have a corresponding effect on the decision variable $u_{(i,j),0}$ in the original grid so that other items can not be placed. As seen in the diagram, this is represented by the black dots that are to the right of the red dots. They represent the decision variable $u_{(i,2),0} = 1 \quad \forall i \in \{1, \ldots, 5\}$ and $u_{(i,4),0} = 1 \quad \forall i \in \{6, \ldots, 10\}$

Objects can then be placed on coordinate $(i, j + 1)$ if $u_{(i,j),0} = 1$

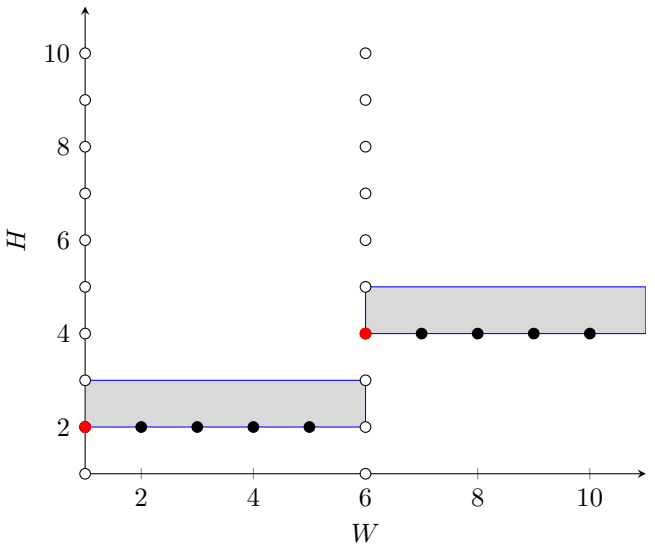

Figure 2: Shelf Grid

4 The Model

The model will first be presented, then the correctness will be explained in the next section.

$$\max \quad \sum_{i,j,k} 6p_k x_{(i,j),k} \tag{1}$$

$$\text{s.t.} \quad \sum_{k\in B} x_{(i,j),k} = \delta_{(i,j),b} \quad \forall i, 1 \le j \le H - h_b + 1 \tag{2}$$

$$\sum_{l=1}^{h_b-1} u_{(i,j+l),1} \ge (h_b - 1)\delta_{(i,j),b} \quad \forall i, 1 \le j \le H - h_b + 1 \tag{3}$$

$$\sum_{k\in C} x_{(i,j),k} = \delta_{(i,j),c} \quad \forall i, 1 \le j \le H - h_c + 1 \tag{4}$$

$$\sum_{l=1}^{h_c-1} u_{(i,j+l),1} \ge (h_c - 1)\delta_{(i,j),c} \quad \forall i, 1 \le j \le H - h_c + 1 \tag{5}$$

$$(w_s - 1)s_{(g,j)} = \sum_{l=1}^{w_s-1} u_{((g-1)w_s+l,j),0} \quad \forall g, j \tag{6}$$

$$\sum_{g,j} s_{(g,j)} \le N_s \tag{7}$$

$$x_{(i,j+1),k} \le u_{(i,j),0} \quad \forall i, 1 \le j \le H - h_b + 1, k \in B \text{ and } \forall i, 1 \le j \le h_c - 1, k \in C \tag{8}$$

$$x_{(i,j+1),k} \le u_{(i,j),0} + \delta_{(i,j-h_c+1),c} \quad \forall i, h_c \le j \le H - h_c + 1, k \in C \tag{9}$$

$$x_{(i,j+t\times h_c),k} \le 1 - \delta_{(i,j),c} \quad \forall i, 1 \le j \le H - t \times h_c + 1, k \in C \tag{10}$$

$$u_{(i,j),0} + u_{(i,j),1} + \sum_{k=1}^{N} x_{(i,j),k} \le 1 \quad \forall i, 1 \le j \le H - h_c + 1, k \in C \text{ and } \forall i, 1 < j \le H - h_b + 1, k \in B \tag{11}$$

$$u_{(i,j),0} + u_{(i,j),1} \le 1 \quad \forall i, H - h_c + 1 < j \le H, k \in C \text{ and } \forall i, H - h_b + 1 < j \le H, k \in B \tag{12}$$

$$\sum_{i,j} x_{(i,j),k} \le HWb_k \quad \forall k \tag{13}$$

$$b_k \le \sum_{i,j} x_{(i,j),k} \quad \forall k \tag{14}$$

$$\sum_{k\in T_c} b_k \ge n_c \quad \forall c \tag{15}$$

$$r_{L,k} \le \frac{\sum_{i,j} x_{(i,j),k}}{\sum_{i,j,k} x_{(i,j),k}} \le r_{U,k} \quad \forall k \tag{16}$$

$$d_{L,k} \le \sum_{i,j} 6x_{(i,j),k} \le d_{U,k} \quad \forall k \tag{17}$$

Where the decision variables and constants are defined as follows:

4.1 Constants

h_c	height of a can
h_b	height of a bottle
w_s	width of the shelf
p_k	price of item k
T_c	Set of items belonging in category c
n_c	Required number of category c items to be stocked
$r_{L,k}$	Required minimum proportion of item k to total number of items to be stocked
$r_{U,k}$	Required maximum proportion of item k to total number of items to be stocked
$d_{L,k}$	Required minimum number of item k to total number of items to be stocked
$d_{U,k}$	Required maximum number of item k to total number of items to be stocked
H	Height of the possible grid
W	Width of the possible grid
B	The set of all items k that are bottles
C	The set of all items k that are cans
N	The total number of types of items
N_s	Maximum number of shelves

4.2 Decision Variables

All the decision variables are binary variables.

$$x_{(i,j),k} = \begin{cases} 1 & \text{if the bottom-left of item k is placed at } (i,j) \\ 0 & \text{otherwise} \end{cases}$$

$$\delta_{(i,j),c} = \begin{cases} 1 & \text{if the bottom-left of a can is placed at } (i,j) \\ 0 & \text{otherwise} \end{cases}$$

$$\delta_{(i,j),b} = \begin{cases} 1 & \text{if the bottom-left of a bottle is placed at } (i,j) \\ 0 & \text{otherwise} \end{cases}$$

Where $x_{(i,j),k}$, $\delta_{(i,j),c}$ and $\delta_{(i,j),b}$ are defined for $1 \leq i \leq W$ and $1 \leq j \leq H - h_k + 1$ where h_k is the height of the object, and for all items k.

$$b_k = \begin{cases} 1 & \text{if item } k \text{ is placed} \\ 0 & \text{otherwise} \end{cases}$$

$$s_{(g,j)} = \begin{cases} 1 & \text{if bottom-left of shelf is placed at } (g,j) \\ 0 & \text{otherwise} \end{cases}$$

Where $s_{(g,j)}$ is defined for $1 \leq g \leq W/w_s$ and $1 \leq j \leq H$.

$$u_{(i,j),0} = \begin{cases} 1 & \text{if a shelf is taking up space} \\ 0 & \text{otherwise} \end{cases}$$

$$u_{(i,j),1} = \begin{cases} 1 & \text{if an item is taking up space} \\ 0 \text{ or } 1 & \text{otherwise} \end{cases}$$

5 Correctness of Model

5.1 Constraints

This brings us to the set of constraints that allow our set-up to function. The equation numbers are referenced from Section 4: The Model.

5.1.1 Feasibility

Since not more than one item can be placed at coordinate (i,j) and only x and u represent placement of objects, focusing on the constraint for cans shown in Equation (11) and (12):

$$u_{(i,j),0} + u_{(i,j),1} + \sum_{i=1}^{N} x_{(i,j),k} \leq 1 \quad \forall i, 1 \leq j \leq H - h_c + 1$$

$$u_{(i,j),0} + u_{(i,j),1} \leq 1 \quad \forall i, j > H - h_c + 1$$

The reason for the two different inequalities for different ranges of j is because $x_{(i,j),k}$ is only defined for $1 \leq j \leq H - h_c + 1$. Corresponding argument applies to bottles.

5.1.2 Volume

The following set of constraints allow corresponding $u_{(i,j),1} = 1$ if an item has been placed.

Do note that the assumption has been made that bottles and cans are of size $1 \times h$ and therefore, the constraints will only be concerned about the height of the object. This can however, be generalized and consider objects that have a width if desired.

Focusing once again on cans, if any can is placed on (i,j) then $\sum_{k \in C} x_{(i,j),k} = \delta_{(i,j),C} = 1$ because of equation (4) and then equation (5) becomes:

$$\sum_{l=1}^{h_c - 1} u_{(i,j+l),1} \geq (h_c - 1)$$

thus, making all the $u_{(i,j),1} = 1$ where necessary to disallow placement of other objects at those coordinates. This is however, not a strict relation and allows for $u_{(i,j),1} = 1$ even if an item is not placed. Note that this does not impact the model. except for Feasibility constraints. Hence in the goal of maximizing profit, it has no impact.

5.1.3 Shelf Placement

Subsequently, shelves are placed with the constraint shown in equation (6). This constraint allows the shelf to take up space on the item grid by making the corresponding $u_{(i,j),0}$ that represent the shelf become 1.

There is also a limit to the number of shelves that can be placed since there are only so many shelves. This is seen in equation (7).

5.1.4 Gravity

Lastly, objects can only be placed on shelves. This can be interpreted as the constraint as shown in Equation (8):

$$x_{(i,j+1),k} \leq u_{(i,j),0}$$

Taking $j = 1$ as the base of the fridge, the decision variable $u_{(i,j),0} = 1$ implies that the coordinate $(i, j+1)$ is now able to have any $x_{(i,j+1),k} = 1$. This is shown in the constraint since if $u_{(i,j),0} = 1$, the constraint is simply $x_{(i,j+1),k} \leq 1$. However, if $u_{(i,j),0} = 0$, then $(i, j+1)$ cannot accommodate a placement of items and therefore the constraint becomes $x_{(i,j+1),k} \leq 0$ and thus forcing $x_{(i,j+1),k} = 0$

A different case needs to be considered for cans as they are objects that can be stacked. Hence for cans, the constraint should be modified slightly once $j \geq h_c$. This is Equation (9)

$$x_{(i,j+1),k} \leq u_{(i,j,0)} + \delta_{(i,j-h_c+1),c}$$

This constraint simply checks if there is a can placed at $(i, j - h_c + 1)$ and if so the constraint will be $x_{(i,j+1),k} \leq 1$ and allow for placement of item k This means that if either $u_{(i,j,0)} = 1$ or $\delta_{(i,j-h_c+1),c} = 1$, then the constraint will become $x_{(i,j+1),k} \leq 1$ However, if both $u_{(i,j,0)} = 0$ and $\delta_{(i,j-h_c+1),c} = 0$, it means $x_{(i,j+1),k} \leq 0$ and no item can be placed at $(i, j+1)$.

Lastly, there is also a limit as to how high cans can be stacked in the fridge due to weight constraints as shown in Equation (10):

$$x_{(i,j+t\times h_c),k} \leq 1 - \delta_{(i,j),c}$$

This works because

$$\text{if } \delta_{(i,j),c} = 1 \implies x_{(i,j+t\times h_c),k} \leq 0$$

disallowing placement of an item in coordinate $(i, j+t\times h_c)$

5.1.5 Variety

5.1.5.1 Set-up b_k

Variety constraints require us to introduce a new decision variable b_k.

This is required because $\sum_{i,j} x_{(i,j),k}$ is representative of the total amount of item k that is stocked but is not boolean. However, for variety constraints, it would be easier to work with a boolean decision variable. Hence, we assign $b_k = 1$ if $\sum_{i,j} x_{(i,j),k} > 0$ and $b_k = 0$ if $\sum_{i,j} x_{(i,j),k} = 0$. The two constraints that are Equation (13) and Equation (14) create this assignment

These constraints work and can be understood through the various cases as shown below:

if $\sum_{i,j} x_{(i,j),k} > 0$:

$$\text{Equation (13)} \implies b_k = 1$$
$$\text{Equation (14)} \implies b_k = 1 \text{ or } 0$$

and if $\sum_{i,j} x_{(i,j),k} = 0$

$$\text{Equation (13)} \implies b_k = 1 \text{ or } 0$$
$$\text{Equation (14)} \implies b_k = 0$$

and correspondingly for b_k, if $b_k = 1$

$$\text{Equation (13)} \implies \sum_{i,j} x_{(i,j),k} \geq 0$$
$$\text{Equation (14)} \implies \sum_{i,j} x_{(i,j),k} \geq 1$$

and if $b_k = 0$

$$\text{Equation (13)} \implies \sum_{i,j} x_{(i,j),k} = 0$$
$$\text{Equation (14)} \implies \sum_{i,j} x_{(i,j),k} \geq 0$$

From the above case analysis, we see that those 2 constraints provide us with the effect that we desire on b_k.

5.1.5.2 Variety Constraints

Since items can be grouped into different categories, with b_k, constraints can be imposed on the number of items from each category such as IPAs, Stouts, Lagers, etc. This can be seen by the constraint as shown in Equation (15).

This enforce a minimum number of items, n_c, that belong to category T_c will be stocked.

Alternative constraints can possibly be formulated based on constraints that are used in the knapsack problem that could place restrictions on things like the brands of the items as well.

5.1.6 Demand

Another concern that is posed is the one of demand. This is a relatively easy constraint to construct. However, there exist 2 possible manners to construct it. We could either require that the items be stocked of a certain amount or a certain proportion.

- Proportion

  As shown in Equation (16)

- Fixed Amount

  As shown in Equation (17)

These are intuitive constraints that place upper and lower bounds on the amount of item k that can be placed. These work because $\sum_{i,j} 6x_{(i,j),k}$ represent the total amount of item k placed.

5.2 Objective function

Lastly, the objective of this program is to maximize the value of the items in the fridge so that the revenue of fridge is optimized. This is found by taking the price, p_k multiplied by the amount of item k stocked. Hence it is simply:

$$\max \quad \sum_{i,j,k} 6p_k x_{(i,j),k}$$

The factor 6 is attributed to the fact that if $x_{(i,j),k} = 1$, it represents a placement of item k in a row of the fridge, which means 6 of item k will be placed at coordinate (i, j).

6 Solving the Problem

Going back to the motivating case of the 'cloud' fridge at Orchard that is managed by Thirsty Craft Beer, we took note of all the items that they sold there as well as their prices.

We then had the opportunity to speak with the staff in charge of managing the fridge to understand how the fridge is like, what the demand for the products are and their constraints on variety.

Thus, the model considers 33 different types of items. They will be placed in a grid that is of size $H = 53$ and $W = 16$ with shelf width 8. The shelves can only be placed at $i = 1$ and $i = 9$ corresponding to a shelf grid of 2×53. There is also only a maximum of 8 shelves to be placed. The cans are recommended to only be stacked up to two layers.

The demand constraints placed lower bounds on items that were tested popular through other sales channels and bounds were placed on the remaining items accordingly based on their popularity.

For variety, a minimum number was imposed for different styles of beer such as IPAs, Stouts, Ales, Lagers and Sours. The values are as follows:

$n_{IPA} = 5, n_{Stout} = 2, n_{Ale} = 5, n_{Lager} = 1, n_{Sour} = 2$

The detailed values that were determined for the demand constraints are attached in the Appendix for further reference.

The model was implemented in JuMP and the Gurobi solver was used. The model took approximately an hour to solve and this could be due to the large amount of constraints placed on the model.

6.1 Interpretation

After solving it in Julia, we achieved a maximum fridge value of $4452 The breakdown of items to be placed are as follows:

Name	Slots	Amt
Kona Longboard Lager	14	84
Pabst Blue Ribbon Lager	5	30
Kona Big Wave Golden Ale	11	66
4 Pines Pacific Ale	9	54
New Belgium Voodoo Ranger IPA	9	54
Colonial Pale Ale	5	30
Colonial Draught Golden Ale	5	30
Lost Coast Revenant IPA	5	30
New Belgium Juicy Hazy IPA	6	36
Mother Earth Vanilla Cream Ale	6	36
Lost Coast Hazy IPA	2	12
Lost Coast Great White Wheat Ale	12	72
Green Flash West Coast IPA	5	30
Colonial Porter	5	30
Green Flash Session Hazy IPA	1	6
Green Flash Tropical DNA Hazy IPA	4	24
Hawkers Yuzu & Plum Sour	4	24
Lost Coast Sharkinator White IPA	1	6
Lost Coast Chocolate Milk Stout	1	6
Green Flash Soul Style IPA	1	6
Colonial IPA	2	12
Kaiju Metamorphosis IPA	3	18
Mother Earth Coffee Stout	3	18
Holgate Blush Wild Berry Sour Ale	1	6

Figure 3: Breakdown of Items

With the selected items, the number of items of each type are shown below:

Style	No. of Beers
Lager	2
Ale	6
IPA	12
Stout	2
Sour	2

Figure 4: Breakdown by Style

Overall, only 24 different type of items stocked in the fridge, with 9 of them not chosen. There are also a total of 720 bottles/cans stocked, with 96 being bottles and 624 being cans. This shows a possible preference for cans given their smaller size.

Not all the shelves were used and the solution corresponded to the fridge placement in Figure 5, where the black items are the shelves, green regions are cans and blue regions are bottles.

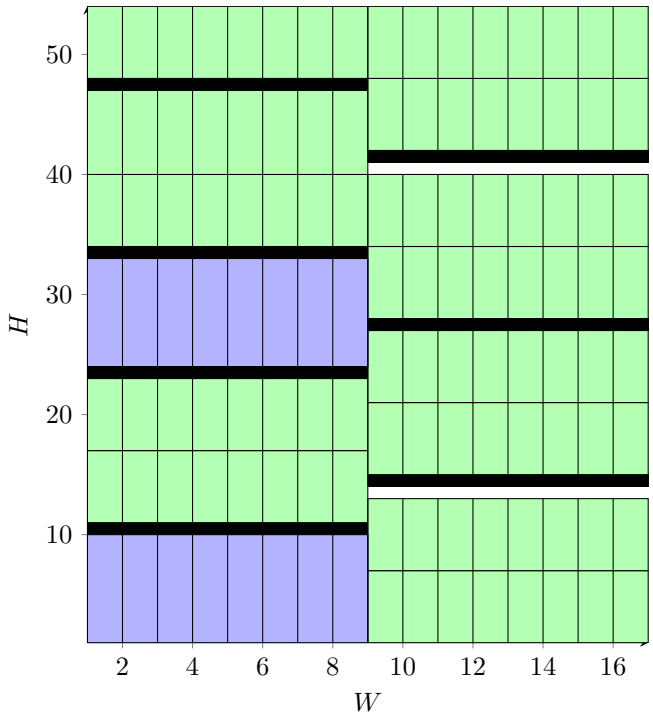

Figure 5: Item & Shelf Configuration

We can also see that the solution is feasible in reality and that the model is working as expected. However, the location to place a certain item k in is arbitrary and not shown in Figure 5. This can be understood since cans of 2 different types can exchange locations but the objective value will not change. Therefore, the decision on the exact location of item k is left to be decided by the person actually stocking the fridge as long as the breakdown of items shown in Figure 3 is satisfied.

7 Further Thoughts

The model has multiple optimal solutions since there exist many arrangement of a certain set of products. However, once a single arrangement is identified, a person can come in and easily reorganise it based on their convenience.

The model also seems to place a preference on higher priced items due to the nature of the objective function. It can thus be noted that the solution to the model is only as good as the demand and variety constraints that are defined. This is due to the fact that if the demand and variety constraints are not well defined, the model will simply return a fridge full of the highest priced items that may not sell well.

8 Appendix

8.1 Data Input

The following is the data that was put into the model, with the columns **Bottle/Can** and **Category** used as indicator variables to input into Julia.

The values for $r_{L,k}$ and $r_{U,k}$ was based off of actual data collected but due to confidentiality reasons will not be displayed.

Name	Price	$\mathbf{r}_{L,k}$	$\mathbf{r}_{U,k}$	Bottle/Can	Category
Kona Longboard Lager	5.5	0.11427	1	Can	Lager
Pabst Blue Ribbon Lager	4.5	0.0342	1	Can	Lager
Kona Big Wave Golden Ale	5.5	0.08674	1	Can	Ale
4 Pines Pacific Ale	5	0.0734	1	Bottle	Ale
New Belgium Voodoo Ranger IPA	7	0.03587	0.07587	Can	IPA
Colonial Pale Ale	6	0.0342	0.0742	Can	Ale
Colonial Draught Golden Ale	5.5	0.0342	0.0742	Can	Ale
Lost Coast Revenant IPA	7	0.03337	0.07337	Bottle	IPA
New Belgium Juicy Hazy IPA	7	0.0167	0.0567	Can	IPA
Mother Earth Vanilla Cream Ale	6.5	0.01586	0.05586	Can	Ale
Lost Coast Hazy IPA	6.5	0.01003	0.05003	Can	IPA
Lost Coast Great White Wheat Ale	6	0.09926	0.13926	Can	Ale
Green Flash West Coast IPA	7.5	0.00669	0.04669	Can	IPA
Colonial Porter	7.5	0.00335	0.04335	Can	Stout
Green Flash Session Hazy IPA	6	0.00252	0.04252	Can	IPA
Green Flash Tropical DNA Hazy IPA	6.5	0.00252	0.04252	Can	IPA
Hawkers Yuzu & Plum Sour	7	0.00085	0.04085	Can	Sour
Lost Coast Sharkinator White IPA	6	0.00502	0.04502	Bottle	IPA
New Belgium Fat Tire Amber Ale	6	0	0.03751	Can	Ale
Lost Coast Watermelon Wheat Ale	6	0	0.03751	Can	Ale
Lost Coast Tangerine Wheat Ale	6	0	0.03668	Can	Ale
Lost Coast Peanut Butter Chocolate Milk Stout	7	0.00585	0.04585	Bottle	Stout
Kaiju Main Squeeze Passion Guava Session Ale	6	0	0.03501	Can	Ale
Green Flash Soul Style IPA	6.5	0	0.03251	Can	IPA
Colonial IPA	6.5	0	0.03001	Can	IPA
Kaiju Krush Tropical Pale Ale	6	0	0.02917	Can	Ale
Lost Coast 8 Ball Stout	6	0	0.02834	Bottle	Stout
Kaiju Metamorphosis IPA	7	0	0.02834	Can	IPA
Holgate Session Pale Ale	5	0	0.02751	Can	Ale
Mother Earth Coffee Stout	7	0	0.02584	Can	IPA
Holgate Blush Wild Berry Sour Ale	6	0	0.025	Can	Sour
Hawkers Berry Kettle Sour	6	0	0.025	Can	Sour
Colonial Small Ale	5	0	0.025	Can	Ale

Figure 6: Data Input into Model

8.2 Julia Code

The Julia Code is on the next page.

It is also available on github