Minimize Cost of Food for CMU Students

21-393 Final Project Fall 2019

Cheyenne Ehman, Julia Keating, Shane Keating, Elizabeth Schulz

1 Abstract

The goal of this project is to determine a meal plan for a Carnegie Mellon University student that provides adequate nutrition while minimizing the cost. Nutrition is looked at over one month where a meal is either from an on campus restaurant or made up of food products from a predesignated list. The number of each unique food product and the number of meals at each restaurant throughout the month are first determined using linear programming based on nutritional constraints. After the linear program is run, the Greedy Algorithm is used to sort food products and restaurant meals into breakfast, lunch, and dinner for thirty days. The result of the linear program was a minimum cost of $344.30 per month and amounts of food specified in section 5.1. Then, the products and meals were separated into meals to create a full meal plan for the month which can be looked at in section 8.

2 Introduction

We will plan and assign meals throughout the course of the month with minimizing cost of the meals and meeting standard nutritional constraints for a CMU student. Our proposed algorithm will be comprised of 2 unique steps: First, we will calculate the total amounts of each food product and restaurant meals eaten over the entirety of the month while taking into account the nutritional constraints and minimizing costs. After we have the totals, they need to be partitioned into 90 separate meals for the entire month. To do so, we introduce a greedy algorithm that produces meals based on the recommended caloric intake for each meal type (Breakfast, Lunch, Dinner). Once we have created a list of 90 meals for the month, we can randomly permute them to introduce variety into our monthly recommended diet.

3 Assumptions

- You must eat three meals a day
- When eating a meal at a restaurant, you have one option
- a healthy individual should eat 2000 calories a day
- You can only eat out at most 9 times per month
- we assume there are 30 days in a month
- food is categorized as being either a breakfast food, lunch food, or a dinner food
- lunch foods and dinner foods can be interchangeable

4 Total products eaten in a month

In the first part of our algorithm, we will calculate the total servings of products and restaurant meals eaten over the course of month.

4.1 Objective Function

Our objective is to minimize the cost of all food bought over the course of the month.

$$min \sum_{p=1}^{P} c_p x_p + \sum_{r=1}^{R} k_r m_r$$

4.2 Variables

p = 1,2,... P # of food products

r = 1,2,... R # restaurants

c_p = cost of product p per serving

k_r = cost of one serving of food at restaurant r

x_p = servings of product p eaten in a month

m_r = servings eaten at restaurant r in a month

g_p = servings of grains in one serving of product p

v_p = servings of vegetables in one serving of product p

f_p = servings of fruits in one serving of product p

l_p = servings of protein in one serving of product p

s_p = servings of sugars in one serving of product p

G_r = servings of grains in meal from restaurant r

V_r = servings of vegetables in meal from restaurant r

F_r = servings of fruit in meal from restaurant r

L_r = servings of protein in meal from restaurant r

S_r = servings of sugar in meal from restaurant r

C_p = calories of a product p

K_r = calories of a meal r

4.3 Constraints

4.3.1 Maximum restaurant meals

Here, we assume that you can eat out a maximum of 30 times in the month.

$$\sum_{r=1}^{R} m_r \leq 30$$

4.3.2 Can buy at most 60 of any given food product

$$x_p \leq 60 \forall p$$

4.3.3 Calories

The recommended calorie intake per day 2000 calories, but since we are calculating meals for the month, you should consume at least 60,000 calories in total. Also, no one should eat more than 2500 calories per day, so there is an upper limit of 75,000 calories per month.

$$60,000 \leq \sum_{p=1}^{P} x_p C_p + \sum_{r=1}^{R} m_r K_r \leq 75,000$$

4.3.4 Grains

It is recommended to eat 9 servings of grains a day for 30 days in the month, so over the course of the month, you should consume 270 servings of grains.

$$\sum_{p=1}^{P} g_p x_p + \sum_{r=1}^{R} G_r m_r \geq 270$$

4.3.5 Fruits

It is recommended to eat 3 servings of fruits a day for 30 days in the month, so over the course of the month, you should consume 90 servings of fruits.

$$\sum_{p=1}^{P} f_p x_p + \sum_{r=1}^{R} F_r m_r \geq 90$$

4.3.6 Vegetables

It is recommended to eat 4 servings of vegetables a day for 30 days in the month, so over the course of the month, you should consume 120 servings of vegetables.

$$\sum_{p=1}^{P} v_p x_p + \sum_{r=1}^{R} V_r m_r \geq 120$$

4.3.7 Proteins

It is recommended to eat 2 servings of protein a day for 30 days in the month, so over the course of the month, you should consume 60 servings of protein.

$$\sum_{p=1}^{P} l_p x_p + \sum_{r=1}^{R} L_r m_r \geq 60$$

4.3.8 Sugars

It is recommended to eat no more than 2 servings of artificial sugar a day for 30 days in the month, so over the course of the month, you should consume between 30 and 60 servings of artificial sugar.

$$30 \leq \sum_{p=1}^{P} s_p x_p + \sum_{r=1}^{R} S_r m_r \leq 60$$

5 Data Collection

Data were collected on 13 different food products spanning categories such as proteins, fruits, vegetables, grains, and dairy. The amount of calories in each food product was collected based on averages given by the United States Department of Agriculture (USDA). The serving sizes for each food product were taken from recommendations by the United States Food and Drug Administration (FDA). Data regarding meals from six different campus restaurants were collected from the CMU Dining Services website. As per the assumptions in section 3, we have assumed that when going to a certain restaurant, you have only one meal option. The meals assigned to each restaurant can be found in the appendix in section 11.1. Some estimation was made based on serving sizes of food products in each meal from the USDA.

5.1 Linear Program Results

The linear program as described above was computationally solved using the simplex algorithm in Microsoft Excel. The output of the algorithm can be seen in the table below. The totals below was calculated with

the minimum cost of $334.30. It's important to note that there as many non-zero decision variables as there are constraints, so this is something to be taken into consideration in the future work section 9.

restaurant/product	value
abp	0
ug	11
resnik	17
tepper	0
exchange	0
inoodle	0
broccoli	0
carrots	60
spinach	60
bread	60
pasta	60
rice	60
chicken	0
cheese	0
egg	18
beans	25
orange	30
apple	0
banana	60

Table 1: Resulting totals from Linear Program

6 Partitioning the Totals into Meals

We combine all of totals from the linear programming output into one set of products/restaurant meals N.

$$N = \{x_1, x_2, ..., x_n, m_1, m_2, ...m_R\}$$

For any product x_p, if the total $x_p > 1$, we represent that product in N by listing the product in N, x_p times. (e.x. suppose the optimal LP output is 3 apples and 2 bananas, the respective $N = \{apple, apple, apple, banana, banana\}$)

We also categorize food products in the optimal N as being from a restaurant or not and if it is a breakfast item. Because we assumed that lunch and dinner items are interchangeable, if an item is not a breakfast item, it falls into this category.

$$R_i = \begin{cases} 1 & \text{if item } i \text{ is bought from a restaurant } \quad \forall i = 1, 2, ..., |N| \\ 0 & \text{otherwise} \end{cases}$$

$$B_i = \begin{cases} 1 & \text{if item } i \text{ is a breakfast item } \quad \forall i = 1, 2, ..., |N| \\ 0 & \text{otherwise} \end{cases}$$

6.1 Greedy Algorithm

To partition the optimal food items into 90 meals, we implement a modified greedy algorithm. For each of the 90 meals, it chooses the item with the highest amount of calories that is still less than the recommended calorie intake for a given meal with a certain error. This error is 100 calories, to account for some meals being slightly more due to variability. Once each meal is created, it is then added to S, the array of meals for the month. Note: This algorithm was coded using the program, R. the code used to implement the partitioning is included in the appendix in section 11.3.

Algorithm 1: Greedy Algorithm

Data: Finite set N of food items, non-negative function f

Result: array S : M ∈ S is a meal composed of products in N

S ← ∅ ;

while $|S| \leq 90$ **do**

  M ← ∅ ;

  while *there exists* $x \in N$ *such that* $f(M \cup \{x\}) > f(M)$ **do**

    add the best such element x to meal M ;

    $M \leftarrow M \cup \{x\}$;

    remove the used item from the set of items;

    $N \leftarrow N \setminus \{x\}$;

  end

  add the meal M to the array S;

  $S \leftarrow S \cup \{M\}$;

end

return S ;

6.2 Meal Calorie Requirement Function f(M)

It is recommended that you eat around 400 calories for breakfast and 800 calories for both lunch and dinner. A meal is considered a breakfast meal, if all items in the meal are categorized as breakfast items.

$$C_M = \begin{cases} \text{400 calories} & \text{if meal M is a breakfast meal: } B_i = 1 \quad \forall i \in M \\ \text{800 calories} & \text{otherwise} \end{cases}$$

$f(M)$ is the function that is being maximized in the greedy algorithm. It calculates the calories of a given meal given that the total calorie count is still less than the recommended calories for the type of meal plus an error of 100 calories. Because this function is being maximized, it ensures that each meal is fairly close to the recommended calorie amount as represented in C_M. The first case in $f(M)$ ensures that each meal is composed of at least 1 item, as $|M|$ with a value of 0 would never be the maximum value. The second case limits that amount of one specific item in any given meal, and the third case ensures that the meal does not go drastically over the recommended calorie intake, and the fourth case forces a meal to be all breakfast items of no breakfast items. Finally, the fifth case, counts the calories given that all of the previous requirements were met multiplied by the umber of unique items in the meal, and this is what is maximized.

$$f(M) = \begin{cases} 0 & \text{if } |M| = 0 \\ 0 & \text{if there is more than 10 of one item in M} \\ 0 & \text{if } \sum_{y \in M} Calories(y) > C_M + 100 \\ 0 & \text{if } B_i \neq B_j \quad \forall i \neq j \in M \\ (\text{\# unique items in M}) * \sum_{y \in M} Calories(y) & \text{otherwise} \end{cases}$$

6.3 Downfall of the Greedy Algorithm

It is important to note that the Greedy algorithm is not a perfect or incredibly efficient solution to our partitioning. The algorithm cannot guarantee the conditions in the f(M) function as it only maximizes it. When implementing the algorithm in code, it actually produced 97 meals rather than just 90.

7 Reordering of the Meals

The Greedy algorithm will result in an array S such that each element in S is a meal or a set food products. We assume that lunch and dinner meals are interchangeable, so the resulting array will be composed of breakfast and lunch/dinner meals. From this, 30 breakfast meals are separated, and we can then randomly permute the set of breakfast meals to introduce variety in the order of meals. The order of these meals will be assigned to the breakfast meal-slot for each of the 30 days. After we have the breakfasts in order, we

permute the remaining 60 lunch/dinner meals, and this will be the order or the lunches and dinner in the month. The first two meals in this order will be lunch and dinner of day 1, the second two meals will be lunch and dinner of day 2, and so on.

8 Final Meal Plan Results

	M	T	W	TH	F	S	SUN
Breakfast	Breadx2 Banana	Bread Bananax2 Egg Orange	Breadx2 Banana Orange	Bread Bananax2 Orange Egg	Bread Bananax2 Egg Orange	Breadx2 Banana	Breadx2 Banana Orange
Lunch	UG	Banana Breadx2	Resnik	Rice Pastax4 Carrotx8 Spinachx2	UG	Ricex3 Pastax2 Carrotx2 Spinachx2	Resnik
Dinner	Ricex3 Pastax2 Carrotx2 Spinachx2	Beansx2 Rice Pastax2 Carrot Spinach	Ricex3 Pastax2 Carrotx2 Spinachx2	UG	Ricex3 Pastax2 Carrotx2 Spinachx2	Resnik	UG

Table 2: Meals for One Week

8.1 Discussion

We were able to successfully create a meal plan for the month. Table 2 above shows a sample week in our resulting meal plan. Our initial methodology was successful in accomplish our goal of creating this meal plan while meeting nutritional constraints and minimizing costs. However, we see that despite our current efforts, there is not much variety in meals from day to day. The only restaurant meals present in this week are form the Underground and Resnik Cafe. Also, almost every day for breakfast, the plan suggests eating bread, bananas, and oranges. This may be a result of the limited breakfast foods in our initial data collection of the implications or the Greedy algorithm.

9 Future Work

In the future, there are several things that would could implement to improve our results. To increase the variety in the types of meals we could use a wider variety of food products like prepared or frozen foods as well as use all meals from a restaurant instead of one. As said earlier, we could also implement more variety constraints into the initial linear program so that are more non-zero decision variables in the final

output. We could also introduce some sort or time constraint as well because cooking at home is relatively costly in terms of time, especially for Carnegie Mellon students, who this study was designed for. We could also better modify the greedy algorithm such that calorie constraints are met per day. For the sake of customization, there should be an added option of "happiness" to customize the output to an individual person. We acknowledge that most people have preferences or allergies that greatly impact how and where they eat.

10 References

Axe, Josh. "How Many Grams of Sugar Per Day Should You Consume?" Dr. Axe, 24 Sept. 2018, draxe.com/nutrition/how-many-grams-of-sugar-per-day/.

Dietary Guidelines: Build a Healthy Base, health.gov/dietaryguidelines/dga2000/document/build.htm.

"Dining Locations." IIS Windows Server, apps.studentaffairs.cmu.edu/dining/conceptinfo/?page=listConcepts.

"FoodData Central." FoodData Central, fdc.nal.usda.gov/.

"Nutrition Facts Label." Accessdata.fda.gov, www.accessdata.fda.gov/scripts/interactivenutritionfactslabel.

11 Appendix

11.1 Restaurant Meal Assignments

- Au Bon Pain: Turkey Chipotle Sandwich
- Underground: Ultimate Brownie
- Resnik: Chicken Strips
- Tepper: Cheese Pizza
- Exchange: Pasta with Chicken and Marinara
- iNoodle: Noodle Bowl with Chicken and Vegetables

11.2 FDA Recommended Daily Servings

- 9 servings of grain
- 3 servings of fruit
- 4 servings of vegetable
- 2 servings of protein
- 2 servings of sugar

11.3 Greedy Algorithm Code

```
# Data Prep
library("readxl")
data <- read_excel("totals.xlsx",sheet = "Sheet1")

# restaurant totals
restaurants <- data[1:6,2:3]
restaurants <- restaurants[which(data$value[1:6] != 0 ),]
# how many restaurant meals
r_meals <- sum(restaurants$value)
# need to make h_meals homemade
h_meals <- 90 - r_meals

totals <- data[7:19,c(2,3,9,11)]

#all restaurant meals labeled out
r_meals_vec <- rep(restaurants$`restaurant/product`,restaurants$value)

# create N list

item <- rep(totals$`restaurant/product`, totals$value)
B <- rep(totals$Breakfast, totals$value)
Calories <- rep(totals$calories, totals$value)

N <- list(item = item,
          B = B,
          Calories = Calories)

f <- function(M){
  #meal not empty
  if(length(M) == 0){
    max <- 0
```

```
  }
   if(max(table(M$item)) > 10){
     max <- 0
   }

  # sum calories
  sum_cal <- sum(M$Calories)
  unique_factor <- length(unique(M$item))
  max <- unique_factor * sum_cal
  #is breakfast meal
  B <- prod(M$B)

  # 400 cal if breakfast meal
  CM <- ifelse(B == 1, 400, 800)

  # can't go over calorie amount
  if(sum_cal > CM + 100){
    max <- 0
  }

  # all items must be breakfast items or none
  if(length(unique(M$B)) > 1 ){
    max<- 0
  }
  return(max)
}

#initialize meal plan S
S <- list()
for (i in 1: length(r_meals_vec)){
  S <- append(S,r_meals_vec[i])
}
```

```
while (length(S) <= 100){
  M <- list(item = c(), B = c(), Calories = c())



  max_f <- 1
  while(max_f != 0){
    max_vec <- c()
  for (n in 1:length(N$item)){
    # Append new item to meal
    M_new <- mapply(append,M,
                    list(N$item[n], N$B[n], N$Calories[n]),
                    SIMPLIFY = FALSE)
    #calc f value for new M, add to vector
    max_vec <- c(max_vec, f(M_new))
  }

  #find which one is the max
  max_f <- max(max_vec)

  # if there is a max, add it to meal
  if(max_f != 0){
  max_index <- which(max_vec == max_f)[1]
  # add best item to meal
  M <- mapply(append,M, list(N$item[max_index],
                             N$B[max_index],
                             N$Calories[max_index]),
              SIMPLIFY = FALSE)
  # remove item used from total list
  N <- lapply(N, function(x) x[-1*max_index])}


  }
```

```
  # add meal to meal plan S
  S <- append(S, list(M$item))
}
# add whats left
S <- append(S, list(M$item))

names(S) <- paste0("meal",1:length(S))

# label which ones are breakfast meals
IS_B <- c()
for(meal in S){
  Breakfast <- c()
  for (i in 1:length(meal)){
    Breakfast <- c(Breakfast,
                   data$Breakfast[which(data$`restaurant/product` == meal[i])])
    boo <- as.logical(Breakfast)
  }
  IS_B <- c(IS_B,all(boo))
}

# Permute Breakfast Meals
B_meals <- S[which(IS_B == TRUE)]
samp_B_meals <- sample(B_meals)
B_meals <- samp_B_meals[1:round(length(S)/3)]

# Permute other meals
O_meals <- S[which(!names(S) %in% names(B_meals))]
O_meals <- sample(O_meals)

#separate into lunch and dinner meals
L_meals <- O_meals[1:(length(O_meals)/2)]
D_meals <- O_meals[((length(O_meals)/2) + 1):length(O_meals)]
```

```
# merge breakfast and other meals in order
meal_plan <- list()
# breakfast
meal_plan[seq(1,length(S),3)[1:round(length(S)/3)]] <- B_meals
# lunch
meal_plan[seq(2,length(S) - 1,3)] <- L_meals[1:length(seq(2,length(S)-1,3))]
# dinner
meal_plan[seq(3,length(S),3)] <- D_meals[1:length(seq(3,length(S),3))]


lapply(meal_plan,
       function(x) write.table( data.frame(t(x)) ,
                                'meal_plan.csv' ,
                                append= T, sep=',',
                                col.names = F))
```