Rank, Predict, Program, Play, Repeat:

An Introduction to Linear Programming

A SENTRY Reconnect Module

Authors: Melanie Brown, Catherine Buell, and Alison Marr

Abstract

This module is aimed to provide students beginning Linear Algebra an opportunity to play with advanced ideas of optimization and linear programming, typically reserved for a course in Optimization or Operations Research. The module builds upon introductory Linear Algebra ideas and implements code in MATLAB (using the Symbolic and Optimization Toolboxes) to allow students the ability to explore complex systems through framed examples and research questions. No previous knowledge of MATLAB or programming is required. It can be used as a classroom exercise or an out-of-class project.

Introduction

Students often inquire into the applications of the mathematical tools that they are learning in their courses. While all levels of mathematical theory have a beauty of their own and presenting applications is possible at various levels, glimpses into advanced techniques and courses can also be a motivating factor for students to continue their study. When students pose questions–inspired by real-life questions or scenarios–very often their access to techniques to solve such questions are a semester or two out of reach. This module is aimed to provide students in Linear Algebra an opportunity to explore and work with the advanced ideas of optimization and linear programming without the fundamentals of the advanced classes. This experience is achieved by building upon skills from Algebra, Calculus regarding optimization, and a matrix representation of a system of equations. However, the material and exercises are presented packaged in a program, currently MATLAB, using their Live Script feature. The program is self-contained, so students do not require any programming experience to use the tool and play through the scenarios. This module also looks to extend students' knowledge from Calculus and Linear Algebra. The exercises were inspired by two research papers, Sparks & Abrahamson (2005) and Bosch (1996). But this module is meant to give students access to research projects and extended activities while being self-contained and appropriate for a student who may never be able to take an Optimization, Linear Programming, or Operations Research course.

The module begins with a review of the concept of optimization through the lens of Calculus and provides some practical examples. Then we provide a brief refresher of Linear Algebra, specifically writing a system of equations as a matrix equation. Either of these sections could be skipped depending upon students' experience and desired scaffolding for the latter parts of the module. We provide a primer introducing MATLAB sufficient to work with the proposed problems in the module. Finally, students are introduced to the linear programming project. Students are encouraged to first mimic an example and then explore, discover, question, and extend the ideas. The themes presented are intended to be entertaining and engaging, but they are modeled from peer-reviewed, research papers.

Tools for Optimization from Calculus and Linear Algebra

Calculus I

First, we will start with the concept of optimization. While often associated with mathematics or related-fields, optimization is essentially the process of finding the best outcome and most effective use of resources to attain that outcome. The concept is familiar outside of a mathematics class, but perhaps first quantized for students in a Calculus 1 course. Students are introduced to the language of constraints (often the bounds of the problem) and an objective function (the outcome we desire to maximize or minimize). A common first example would be the garden example, where a rectangular garden is to be constructed using a rock wall/river/building as one side of the garden and wire fencing for the other three sides. Given 100 ft of wire fencing, determine the dimensions that would create a garden of maximum area. What is the maximum area? The process to solve involves writing equations for a variable, substitution into the objective function, and techniques of differentiation. Upon exposure to other fields like economics, business, engineering, or through their own curiosity, students discover that there can be a multitude of constraints and a variety of variables that aren't always going to be able to be incorporated with the tools they have in a Calculus I course. Some of these tools are introduced later in Multivariable Calculus, but others much later or not at all.

Linear Algebra

Upon adventuring into Linear Algebra, students are presented with solving systems of linear equations with a multitude of variables. Very often these linear equations may not be modeling a

particular situation as the class is working on understanding the theory. Of course, early examples can be pulled from engineering, chemistry, and business, but systems are typically consistent and standardized. Students learn techniques of Gaussian elimination to turn systems of equations into equivalent systems that are more easily solved. This technique, often first taught by hand, helps students to identify pivots to do these elementary operations, and this process may be a bit mysterious when students consider that a computer can automate these operations. However, those skills allow us to manipulate more complicated systems in linear programming where the choices of pivots can benefit from human intervention in the problem. The techniques of solving a linear programming problem (including the Simplex method and tableau) are outside the scope of this module, as they are usually taught over several weeks or months within a class dedicated to optimization. However, we may find that a linear programming solution does not appear to exist in the boxed MATLAB program function. Sometimes we can see why there is no solution; however other times a re-writing can create a feasible solution. This re-writing process is outside the scope of this module.

Combining Skills to Pose a Linear Programming Example

Similar to the Calculus I example, students quickly discover that there are restrictions to these new Linear Algebra skills and 1) we often cannot solve a system of linear equations and 2) very few questions posed require precise equality in the constraints and an inequality would better suit the expression of the question. The first of these two discoveries can lead to the conversations of the least-squares solution to a system of linear equations and minimizing the error, which is an optimization question. The second can start the conversation on linear programming, when constraints in the system are linear inequalities.

One problem-solving technique typically not taught in Linear Algebra is the idea of using a matrix equation as part of posing and solving a Calculus I linear optimization problem that has a system of constraints. These problems are typically not taught because most optimization problems posed are non-linear, and there are better techniques in the Calculus series to tackle these problems. Below we provide a demonstration of this technique through an example illustrating how an objective function and a system of equalities are interpreted in a matrix. We will also solve the problem using the familiar Calculus I techniques to scaffold this new process. Note that this is just an example, starting somewhere familiar, before we investigate systems of

linear inequalities the structure of linear programming. This example is a "standard" linear programming problem as the constraints are linear equalities and all the variables are assumed to be non-negative.

Example 1: A rectangular package mailed through the postal service can have a maximum combined length, width, and height of 108 inches. The length is twice the width. Find the dimensions of the package that maximize the volume.

The objective is to maximize the volume of the box. Let's let l be the length of the package, w be the width, and h be the height of the package. Then,

Objective Function, V =lwh, Constraint 1, l+w+h =108, Constraint 2, l=2w or l-2w=0

We will put this in the language of Linear Algebra and the common form of a linear programming problem with objective function z written as a linear combination of our variables and the matrix A containing our constraints.

Maximize V=lwh

*Subject to: A**x**=**b** and **x**>**0***

We will let V be represented as a new variable z, l=x1, w=x2, h=x3, then above is equivalent to Maximizing z = x1x2x3 subject to $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 0 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} = \begin{bmatrix} 108 \\ 0 \end{bmatrix}$ *where x1, x2, x3>0.*

Solving the system gives the parametric solution $\mathbf{x} = \begin{bmatrix} 72 - \frac{2}{3}t \\ 36 - \frac{1}{3}t \\ t \end{bmatrix}$ *for 0 < t < 108.*

We can now substitute into z = $\left(72 - \frac{2}{3}t\right)\left(36 - \frac{1}{3}t\right)t = 2592t - 48t^2 + \frac{2}{9}t^3$..

Solving z'=0 gives critical values of t = 108 or t = 36. A second derivative test verifies dimensions 48in x 24in x 36in give the maximum volume.

Certainly, this problem could be solved without using the language of linear algebra, but since our constraints were linear, we can solve the simultaneous equations using linear algebra. We modeled this example to prepare to use these skills for our linear optimization problem posed at the end.

An Introduction to Linear Programming and MATLAB

As mentioned in the introduction, we will not be covering the techniques associated with linear programming problems as these techniques are often taught over several weeks in a dedicated course. However, we will provide an introduction to posing these problems (encouraging students later to pose their own problems) and to using MATLAB's *linprog* solver.

Standard Form for a Linear Programming Problem in MATLAB

A linear programming problem, like the optimization problem in the previous section, consists of an objective function and constraints. Here, we can have multiple variables, but the objective function must be linear (and expressed as a dot product of two vectors) and the constraints a system of linear inequalities. To use the *linprog* solver in MATLAB, we will use the following standard form of a problem.

$$\textit{Minimize } z=\boldsymbol{c}^T\boldsymbol{x}$$

$$\textit{Subject to: } A\boldsymbol{x} \leq \boldsymbol{b}$$

The MATLAB *linprog* solver requires we re-write our objective as a minimization problem and all our objectives as "less than or equal to." Note that maximizing z can be rephrased as minimizing $-z$. The MATLAB solver can also do upper bounds, lower bounds, and constraints that are linear equalities; however, these will not be necessary for this simplification. We will practice phrasing a problem in this form.

Example 2: A company creates square boxes and triangular boxes. Square boxes take 2 minutes to make and sell for a profit of 4 dollars. Triangular boxes take 3 minutes to make and sell for a profit of 5 dollars. Their client wants at least 20 boxes and at least 8 of each type ready in under one hour. What is the best combination of square and triangular boxes to make so that the company makes the most profit from this client?

The objective is to maximize the profit. The constraints are the time and the minimal requirements of the order. Let x1 be the number of square boxes and x2 be the number of triangular boxes. We want to maximize profit = 4x1+5x2 subject to $2x1+3x2 \leq 60$, $x1+x2 \geq 20$, $x1 \geq 8$, *and* $x2 \geq 8$. *To use the MATLAB solver, we must change these constraints into the standard form to minimize* $z=\boldsymbol{c}^T\boldsymbol{x}$ *and subject to* $A\boldsymbol{x} \leq \boldsymbol{b}$. *First, instead of maximizing profit we*

will rename the objective and transform it to a minimization problem, so we will now minimize z=-4x1-5x2. We can write z as the dot product of a vector ***c*** *and vector* ***x****.*

$$z= \begin{bmatrix} -4 & -5 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \end{bmatrix}$$

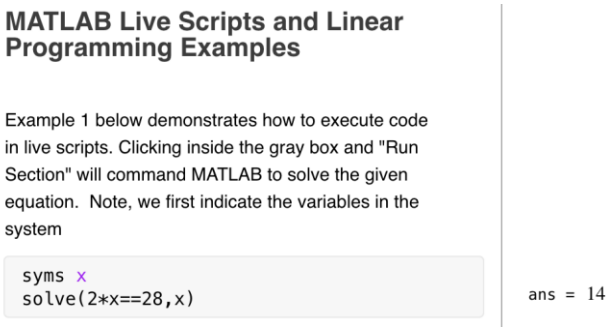
And we have four linear inequalities, we will write these in matrix form, but each must be written as "less than or equal to." Therefore, 2x1+3x2 ≤ 60, x1+x2 ≥ 20, x1 ≥ 8, and x2 ≥ 8 become 2x1+3x2 ≤ 60, -x1-x2 ≤ -20, -x1 ≤ -8, and -x2 ≤ -8 or in matrix form:

$$A= \begin{bmatrix} 2 & 3 \\ -1 & -1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \text{ and } b=. \begin{bmatrix} 60 \\ -20 \\ -8 \\ -8 \end{bmatrix}$$

Once the question is posed, we'd add this information into the MATLAB program. Usually, a linear programming system in standard form would be written in tableau format and solved; however, as one can see in (Nichols & Christogaro, 2016; Vanderbei, 2001) these techniques are complex and nuanced to reach a solution and beyond the scope of this module. In lieu of teaching these techniques, we will introduce MATLAB's *linprog* solver.

MATLAB's Live Script and *linprog*

We are utilizing MATLAB's Live Script to have a contained space for students to execute code. Packaged with this module (and featured in Appendix B) are the MATLAB codes. Below you'll find screenshots indicating the ways that we introduce variables in a system and set up linear programming problems. The example below is the first example of executable code in MATLAB1.mlx.



Next, we will consider Example 2 with MATLAB. Note, introducing the symbolic variables is not necessary in this particular program, but we want to get into the habit in order to program more advanced activities. The following code is in MATLAB1.mlx.

```
%%First we introduce the symbolic variables

syms x1 x2

%%Second we enter the problem

%% Vector c of constants for function to minimize
c = [-4 -5];

%% Matrix A and vector b for constraints
A = [2 3
     -1 -1
     -1 0
     0 -1];

b = [60
     -20
    -8
     -8];

%%Finally, we use linprog with these three components

Solution = linprog(c,A,b);
```

When we run the section, we see "Optimal solution found" and this solution can be found on the variable workspace. In this case the solution is [18,8] which means the company should make 18 square boxes and 8 triangular boxes. There is much going on in the background of this program and one can see this by changing some of the constraints. We encourage students to play with modifying the code. We have copied the code into the next blank space on the Live Script for this purpose. For example, modifying the minimal order to 5 of each box type creates the optimal solution [22.5,5] which is not possible. We cannot produce 22.5 square boxes, so since we are optimizing and have a time constraint, we will need to round down to say 22 square boxes and 5 triangular boxes. Another example would be if we asked for a minimum of 30 boxes. In this case, there is "No feasible solution found."

Before introducing the research project, we encourage readers to attempt the following example (solution at the end). A space has been left for this exploration in the MATLAB1.mlx file.

Exploration Question 1: A woman makes craft jewelry to sell at a seasonal craft show. She makes pins and earrings. Each pin takes her 1 hour to make and sells for a profit of 7 dollars. The pairs of earrings take 2 hours to make, but she gets a profit of 20 dollars. She likes to have variety, so she wants to have at least as many pins as pairs of earrings. She also knows that she has approximately 40 hours for creating jewelry between now and the start of the show. The craft show vendor wants sellers to have more than 20 items on display at the beginning of the show.

a) Assuming she sells all her inventory, how many pins and earring pairs should the woman make to maximize her profit?

b) How could you change the profit margins to minimize the time needed to create the work? Maybe in 30 hours? Try changing these components in the system.

Now that we have some basic examples onboard, we are going to describe the question and solution posed by Sparks & Abrahamson (2005), and then present a modification of the question for student exploration.

An Exploration with Linear Programming

The exploration in this module is loosely based and inspired by an exploration of the prediction of winners and ranking of the Cy Young Award winner from Sparks & Abrahamson (2005). The Cy Young Award winner is the highest ranked pitcher by a vote, the electors each rank the best pitchers for the season and first, second, and third are assigned points. The paper uses 20 years of data for the top three point-learners (and ultimately the top winner) of the Cy Young Award. The authors point out that no one really knows what criteria each voter is using to determine their top vote for the award. The goal of the paper was to see if a system could be posed and solved to determine appropriate weights to five chosen criterion or parameters (they chose wins (W), losses (L), earned run average (ERA), team winning percentage (TWP), and strikeouts (K)) to predict the top three ranked pitchers each year. The authors solved for the weights or constants in the system to determine what might be considered the most important parameters. Ultimately, the authors also used computational software to solve the system and even had to

remove certain years to have a feasible system. We will briefly present their findings and formulation before generalizing this ranking system in order to apply it to an exploration.

We will modify and simplify their presentation to just a single year to present the ideas, but we recommend interested students to see how an additional subscript on the variables allows for multiple years to be programmed into the system. For each pitcher *i*, they calculated a total score, S_i based on those five statistics mentioned above: wins (p_1), losses (p_2), earned run average (p_3), team winning percentage (p_4), and strikeouts (p_5). Those five statistics were normalized and reported for each pitcher. For the weights to make sense, the authors scaled each statistic to be a number between 0 and 10 through linearization: "These formulas are chosen with the idea in mind that a parameter value of zero should correspond to a performance of no value to voters, while a value of ten should be historic in the modern era of baseball." For example, for strikeouts, *K*, the scaling formula was $10\left(\frac{K-50}{333}\right)$. When the paper was written the record for most strikeouts in a season was 383, so a pitcher with this record would have a value of 10. The authors surmised that no pitcher with fewer than 50 strikeouts would even be considered. This step of normalizing the parameters is not necessary, but it does make the weights more meaningful, versus accounting for magnitude.

The goal of the project was to be able to correctly predict the ranking of the pitchers based on their scores S_i. So, if we consider the top three positions, we will want $S_1>S_2>S_3$. The score is calculated as $S_i=x_1p_{1i}+x_2p_{2i}+x_3p_{3i}+x_4p_{4i}+x_5p_{5i}$ where the $x = (x_1\ x_2\ x_3\ x_4\ x_5)$ are the weights that will force our ordering/ranking above and p_{ji} is the normalized value of parameter *j* for pitcher *i*. This means we can formulate our constraints.

1) $S_1>S_2$ implies $(x_1p_{11}+x_2p_{21}+x_3p_{31}+x_4p_{41}+x_5p_{51}) - (x_1p_{12}+x_2p_{22}+x_3p_{32}+x_4p_{42}+x_5p_{52}) > 0$ and $S_2>S_3$ implies $(x_1p_{12}+x_2p_{22}+x_3p_{32}+x_4p_{42}+x_5p_{52}) - (x_1p_{13}+x_2p_{23}+x_3p_{33}+x_4p_{43}+x_5p_{53}) > 0$. In preparation to use our MATLAB solver, we need to rephrase these as a linear system using "less than or equal to" and avoid the value of 0, so instead we will use a small value to limit the system. These systems can be written

$$\sum_{k=1}^{5} x_k(p_{k1} - p_{k2}) \geq 0.001 \ \Rightarrow \ \sum_{k=1}^{5} x_k(p_{k2} - p_{k1}) \leq -0.001$$

and

$$\sum_{k=1}^{5} x_k(p_{k2} - p_{k3}) \geq 0.001 \Rightarrow \sum_{k=1}^{5} x_k(p_{k3} - p_{k2}) \leq -0.001$$

We will further constrain the weights to force the score S_i to be a convex combination, meaning that the weights sum to 1 ($\sum_{k=1}^{5} x_k = 1$) and we force the weights to be positive ($x_k \geq 0$, *k=1...5*). When translating these constraints into our system, we will need to be clever to require equality.

$$-x_k \leq 0, k=1...5$$

$$1 \leq \sum_{k=1}^{5} x_k \leq 1 \Rightarrow \sum_{k=1}^{5} x_k \leq 1 \text{ and } \sum_{k=1}^{5} -x_k \leq -1$$

Finally, we have our constraints! There must still be an objective function. The authors chose to maximize the score of the winning pitcher each year. Since we are only modeling one year, we will want to maximize S_1, but again, for our system this will mean we want to minimize $-S_1$.

We are now prepared to state the standard form of the problem:

$$\textit{Minimize } z=-(x_1p_{11}+x_2p_{21}+x_3p_{31}+x_4p_{41}+x_5p_{51})$$

$$\textit{Subject to: } \sum_{k=1}^{5} x_k(p_{k2} - p_{k1}) \leq -0.001$$

$$\sum_{k=1}^{5} x_k(p_{k3} - p_{k2}) \leq -0.001$$

$$, \quad -x_k \leq 0 \quad k=1...5 \quad \sum_{k=1}^{5} x_k \leq 1 \quad \sum_{k=1}^{5} -x_k \leq -1$$

Note, this system above technically models ANY ranking problem where you want to understand parameters that contribute to a ranking system. Before we demonstrate some fun applications of this model, provide the code, and pose several exploratory examples for students to investigate, we will compare the 2022 Cy Young Award results with the weights from the paper that used the older data. The authors found the weights listed below on the left, indicating that the Wins was the highest weight. The table on the right indicates the votes for 2022 Cy Young Awards with Justin Verlander chosen as the winner.

$x_1 = 0.578084,$ (Wins)
$x_2 = 0.00999357,$ (Losses)
$x_3 = 0.196700,$ (ERA)
$x_4 = 0.0784757,$ (Team Winning Proportion)
$x_5 = 0.136747,$ (Strikeouts)

Justin Verlander*	HOU	210 pts
Sandy Alcantara	MIA	210 pts
Dylan Cease	CWS	97 pts

We've calculated the five statistics for these three pitchers and present their vectors and scores using the weights from the 2005 paper (note, their data stopped before 2005).

$$p_{\cdot 1} = (6,\ 7.333,\ 8.125,\ 8.086,\ 4.054)$$
$$p_{\cdot 2} = (4.667,\ 4,\ 6.8,\ 3.519,\ 4.715)$$
$$p_{\cdot 3} = (4.667,\ 4.667,\ 7,\ 6.358,\ 5.315)$$

Therefore, using the 2005 weights, we'd expect:

$$S_1 = x^T p_{\cdot 1} = 6.329$$
$$S_2 = x^T p_{\cdot 2} = 4.996$$
$$S_3 = x^T p_{\cdot 3} = 5.347$$

Justin does have the highest score! But we see that Sandy received the same number of votes as Justin in the Cy Young Award scored ranking. Sandy, in fact, scores overall less than Dylan. It is worth running our year to see what the weights would be. Let's recall our original standard form and then, **in bold**, the system with our values in the system.

Minimize $z=-(x_1p_{11}+x_2p_{21}+x_3p_{31}+x_4p_{41}+x_5p_{51})$

z=-(6x1+7.333x2+8.215x3+8.086x4+4.054x5)

Subject to: $\sum_{k=1}^{5} x_k(p_{k2} - p_{k1}) \leq -0.001$ and $\sum_{k=1}^{5} x_k(p_{k3} - p_{k2}) \leq -0.001$

x1(4.667-6) + x2(4-7.333) + x3(6.8-8.125) + x4(3.519-8.086) + x5(4.715-4.054) ≤ -0.001

x1(-1.333) + x2(-3.333) + x3(-1.325) + x4(-4.567) + x5(0.661) ≤ -0.001

and

x1(4.667-4.667) + x2(4.667-4) + x3(7-6.8) + x4(6.358-3.519) + x5(5.315-4.715) ≤ -0.001

x1(0) + x2(0.667) + x3(0.2) + x4(2.839) + x5(0.6) ≤ -0.001

$$-x_k \leq 0 \qquad \sum_{k=1}^{5} x_k \leq 1 \qquad \sum_{k=1}^{5} -x_k \leq -1$$

Notice that none of Sandy's five statistics were higher than Dylan's five statistics. Therefore, the second requirement (highlighted above) cannot be attained since we require all the weights to be positive. We can already see that we will not have a solution. The example, as written above, has been programmed in MATLAB2.mlx. You can see when the program returns "No feasible solutions found."

Exploration Question 2: What parameter could be missing? Find some parameter that might allow Sandy to out-score Dylan in the 2022 rankings, then scale it linearly to a value between 0 and 10. A hint for linearization: look-up record highs and lows for top players in the rankings. See Spark & Abrahamson for more examples of linearization. Modify the code in the beginning of MATLAB2.mlx to add and remove parameters and determine and interpret the new weights.

Experimenting with Ranking Systems and Weights

Perhaps baseball is not your thing–you aren't alone. Above, we mentioned that the system can help develop an understanding of preferences for any ranking of "top three." In Appendix A, there are three data sets each of the "top-three" (or more) in that category. We also provide several parameters for each. We will explore the first data set in this module, then leave the remaining sets and open questions for student exploration and experimentation.

Ranker.com highlights that the Best Girl Scout cookies are 1) Thin Mints®, 2) Samoas®/ Caramel deLites, and 3) Tagalongs®/Peanut Butter Patties. What parameters do voters use to determine their ranking? Below is the table of several variables retrieved regarding these three cookies.

Cookie Name	Calories per serving	Cookies per box	Year first introduced	Percentage of sales	Grams of sugar per serving	Gram of fat per serving
Thin Mints®	160	32	1941	25	10	7
Samoas®	150	15	1974	19	11	8
Tagalongs®	140	15	1976	13	8	8

We need not choose all six categories, but we will discuss normalizing all six parameters and then choosing three to four for our program. Suppose America's favorite cookie is the Toll House chocolate chip cookie, and per serving: 180 calories, 14g sugar, and 8g fat. Clearly America's least favorite cookie is the Fig Newton, and per serving: 100 calories, 12g sugar, and 2g fat. We can use these to get a sense of preferences. Other important facts are that the Girl Scouts started to commercially sell cookies in the 1930's; we will surmise a long-life span would indicate preference. The largest box of Girl Scout cookies has 44 cookies and the least has 14. And finally, there are 11 different types of cookies, so a popular cookie should have more than 9.09% of the sales. For each, we will find a slope or appropriate scale, then modify each statistic. Notice that each scaling formula follows the Sparks and Abrahamson convention of returning a value between 0 and 10.

Calories per serving: (cal-100)/8	Cookies per box: (cookies-10)/4	Year Introduced: -(year-2022)/9
Percentage of Sales: percent/4	Grams of Sugar: 10*(grams-6)/9	Grams of Fat: grams (no scale needed!)

Scaled Statistics:

Cookie Name	Calories per serving	Cookies per box	Year introduced	Percentage of sales	Grams of sugar per serving	Grams of fat per serving
Thin Mints®	7.5	8	9	6.25	4.44	7
Samoas®	6.25	1.25	5.33	4.75	5.56	8
Tagalongs®	5	1.25	5.11	3.25	2.22	8
Trefoils®	7.5	7.5	7.89	1.75	1.11	7

Upon scaling the statistics, we notice that Samoas® beats out Tagalongs® in every statistic, similar, this makes the opposite (and boring) situation that any linear combination will have a score for Samoas® higher than Tagalongs®, so we've added Trefoils®/Shortbread (ranked 4th!). Note, while it looks like the percentage of sales could decide it all, the 6th-ranked cookie, Do-si-dos® (featured in the appendix data), has 16% of the shares of sales. We will use Calories ($p_{\cdot 1}$), Grams of fat ($p_{\cdot 2}$), Year introduced ($p_{\cdot 3}$), and Grams of sugar ($p_{\cdot 4}$) as our four variables to see if we can achieve this ranking.

Like the model above, we want to minimize the negation of Thin Mints's® score, but we also must modify the original model to have four parameters instead of five and ranking four cookies instead of three pitchers. An additional cookie adds one more constraint.

$$\textit{Minimize } z=-(x_1p_{11}+x_2p_{21}+x_3p_{31}+x_4p_{41})$$

$$\boldsymbol{z=-(7.5x1+7x2+9x3+4.44x4)}$$

Subject to:

$$\sum_{k=1}^{4} x_k(p_{k2} - p_{k1}) \leq -0.001 \ , \ \sum_{k=1}^{4} x_k(p_{k3} - p_{k2}) \leq -0.001 \, , \ \sum_{k=1}^{4} x_k(p_{k4} - p_{k3}) \leq -0.001$$

$$\boldsymbol{x1(-1.25) + x2(1) + x3(-3.67) + x4(1.11) \leq -0.001}$$

$$\boldsymbol{x1(-1.25) + x2(0) + x3(-0.22) + x4(-3.33) \leq -0.001}$$

$$\boldsymbol{x1(2.5) + x2(-1) + x3(2.78) + x4(-1.11) \leq -0.001}$$

$$-x_k \leq 0 \qquad \sum_{k=1}^{4} x_k \leq 1 \qquad \sum_{k=1}^{4} -x_k \leq -1$$

This system is programmed into MATLAB2.mlx. The weights found are *x1=0, x2=0.7357, x3=0.2643,* and *x4=0.* The weight *x3* implies that the time the cookies have around been is a factor, but a higher fat content holds more weight, as demonstrated in the weight *x2*. Our cookie scores are in the table below. Of course, these results are based on the parameters chosen. We can see we've clearly met this ranking.

Cookie Name	Score
Thin Mints®	7.5286
Samoas®	7.2952
Tagalongs®	7.23646667
Trefoils®	7.23493333

Exploration Question 3: There were more possible parameters presented (and even more in Appendix A). Can you determine sets of parameters that provide feasible vs. infeasible linear programming problems? Can you explain why? Believe it or not, the 6th ranked cookie is the well-known Do-si-dos has the following parameters:

	Calories per serving	Cookies per box	Year introduced	Percentage of sales	Grams of sugar per serving	Grams of fat per serving	Score
Do-si-dos®	7.5	2	7.89	6.25	4.44	7	7.235227

Under the current weights, the cookie just sneaks into 4th position. Can you add this fifth cookie to the system? Can you find a feasible system that tells us about the weights of this ranking? Modify the code in MATLAB2.mlx.

Exploration Question 4: Appendix A has several ranked data sets from a variety of polls or community-sourced websites with a variety of parameters to consider. Choose a data set and desired parameters to test for a solution. You will need to scale the statistics through your own research, explore different parameter combinations, and search for the limits of a feasible system while interpreting the resulting weights.

Glossary (Definitions from Oxford Languages)

Feasible Solution

a solution which satisfies all the constraints of an optimization problem.

Linear Programming

a mathematical technique for maximizing or minimizing a linear function of several variables, such as output or cost.

Optimization

the action of making the best or most effective use of a situation or resource.

Simplex Method and Tableau

a standard method of maximizing a linear function of several variables under several constraints on other linear functions. The tableau is used to perform row operations on the linear programming model as well as to check a solution for optimality.

References

Bosch, Robert A. (1996) The battle of the burger chains: Which is best—Burger King, McDonald's or Wendy's?, *Socio-Economic Planning Sciences*, Volume 30, Issue 3, pages 157-162, https://doi.org/10.1016/0038-0121(96)00015-8.

Sparks, Rebecca L. & David L. Abrahamson (2005) A Mathematical Model to Predict Award Winners, *Math Horizons*, Volume 12, Issue 4, pages 5-13, DOI: 10.1080/10724117.2005.12021813

Nichols, Laurel & Gina Christogaro .(2015) Explanation of Simplex Method, *IMSE Concept Library*, https://www.imse.iastate.edu/files/2015/08/Explanation-of-Simplex-Method.docx

Vanderbei, Robert J. (2001) *Linear Programming: Foundations and Extensions*, 2nd edition, International Series in Operations Research & Management Science, Springer, NY, NY.

Appendix A - Ranked Datasets for Linear Programming Ranking Exploration

Best Girl Scout Cookies	https://www.ranker.com/crowdranked-list/the-best-girl-scout-cookies accessed 12/30/22						
	Calories per serving	Cookies per serving	Cookies per box	Year first produced	Percentage of sales	Grams of sugar per serving	Gram of fat per serving
Thin Mints®	160	4	32	1941	25%	10	7
Samoas®	150	2	15	1974	19%	11	8
Tagalongs®	140	2	15	1976	13%	8	8
Trefoils®	160	5	40	1951	7%	7	7
Do Si Dos® (ranked 6th)	160	3	18	1951	16%	11	7

Most Memorable Movie Sidekicks	https://www.ranker.com/list/the-most-memorable-film-sidekicks-ever/marc-cuenco - accessed 12/30/22						
	# of movies they appear	Total movies in series	Worldwide Box Office (in millions)	Total Budget (in millions)	# of Oscars for the series	Minutes of Screen Time	Height (m)
Samwise Gamgee	3	8	5,083.07	981	17	78	1.27
Chewbacca	8	12	10,318.20	1,738	10	113	2.29
Donkey	4	9	3,696.22	575	1	unknown	2.13
R2D2	11	12	10,318.20	1,738	10	91	1.07
Hermione Granger	8	13	9,586.20	1,734	0	205	1.65

Best Fast Food Brands	https://www.ranker.com/crowdranked-list/top-fast-food-brands -accessed 12/30/22						
	# of locations	Cost of most popular menu item	US States with locations	# of Dipping Sauces	Profit in 2021 (in millions)	Social Media Followers (in millions)	# of Tweets (in thousands)
Chick-fil-a	2,732	2.11	47	8	16,700	11.674	217.9
Five Guys	1,390	11.12	49	7	2,093	1.648	43.2
Wendy's	5,938	2.55	50	9	11,111	14.432	222.6
Taco Bell	7,002	3.44	50	7	12,600	15.123	806.3
In-N-Out Burger	370	4.04	4	3	1,175	3.399	0.673

Appendix B - Code for MATLAB programs (MATLAB1.mlx, MATLAB2.mlx)

MATLAB1.mlx

Example 1

```
syms x
solve(2*x==28,x)
```

Example 2

```
%%First, we introduce the symbolic variables

syms x1 x2

%%Second, we enter the problem

%% Vector c of constants for function to minimize
c = [-4 -5];

%% Matrix A and vector b for constraints
A = [2 3
    -1 -1
    -1 0
    0 -1];

b = [60
    -20
   -8
    -8];

%%Finally, we use linprog with these three components

Solution = linprog(c,A,b);
```

Code to modify and test:

```
%%First, we introduce the symbolic variables

syms x1 x2

%%Second, we enter the problem

%% Vector c of constants for function to minimize
c = [-4 -5];
```

```
%% Matrix A and vector b for constraints
A = [2 3
    -1 -1
    -1 0
    0 -1];

b = [60
    -20
   -8
    -8];

%%Finally, we use linprog with these three components

Solution2 = linprog(c,A,b);
```

MATLAB2.mlx

Cy Young Award Exploration

```
%%First, we introduce the symbolic variables

syms x1 x2 x3 x4 x5

%%Second, we enter the problem

%% Vector c of constants for function to minimize
cz = [-6 -7.333 -8.215 -8.086 -4.054];

%% Matrix A and vector b for constraints
A = [-1.333 -3.333 -1.325 -4.567 0.661
    0  0.667   0.2 2.839   0.06
    -1 0 0 0 0
    0 -1 0 0 0
    0 0 -1 0 0
    0 0 0 -1 0
    0 0 0 0 -1
    1 1 1 1 1
    -1 -1 -1 -1 -1];

b = [-0.001
    -0.001
   0
   0
   0
```

```
    0
    0
    1
    -1];

%%Finally, we use linprog with these three components

Solution = linprog(c,A,b);
```

Girl Scout Cookie Exploration:

```
%%First we introduce the symbolic variables

syms x1 x2 x3 x4

%%Second we enter the problem

%% Vector c of constants for function to minimize
c = [-7.5 -7 -9.22  -4.44];

%% Matrix A and vector b for constraints
A = [-1.25 1 -3.67 1.11
    -1.25  0   -0.22 -3.33
    2.5  -1  2.78  -1.11
    -1 0 0 0
    0 -1 0 0
    0 0 -1 0
    0 0 0 -1
    1 1 1 1
    -1 -1 -1 -1 ];

b = [-0.001
    -0.001
    -0.001
    0
    0
    0
    0
    1
    -1];
%%Finally, we use linprog with these three components

Solution2 = linprog(c,A,b);
```