3.1 – Modeling with Prescriptive Analytics

1 Prescriptive Analytics

Prescriptive analytics uses a mathematical model to recommend the best behavior or decision to enable an organization to meet its goals [3]. Prescriptive analytics often prioritizes use of optimization models which seek to find values of variables that "optimize" some function given constraints [3]. The optimization model we will use for this course is a linear programming model (LP).

2 Types of Questions

Linear programming is one of the most common and widely used optimization models. It's used to answer questions such as:

- How should a company allocate labor to maximize profit?
- What combination of ingredients should be used to minimize cost while meeting nutritional needs?
- How should I allocate training time to maximize fitness while balancing academic and company time commitments?
- How many of each product type should a company produce to maximize profit under resource constraints?

Note that all of these questions focus on maximizing or minimizing a quantity under constraints. They also seek to identify what *should* happen, not what is likely to happen. We are going beyond making a prediction to answer a question that directly relates to making a decision.

3 Applying the Modeling Triangle

As we've shown in the last several blocks, the mathematical modeling triangle can be tailored to LP. Our *transform*, *solve*, and *interpret* steps can be more specific to building a LP model. The general triangle with the customizations for LP models are found in Figure 1.

3.1 Transform

As shown in Figure 1, the flow through the transform step is the same. We start with what we are *given*, the research question and other context. We also need to understand the decision to be made; this is what we need to *find*. Once the inputs are understood, we *explore* the different possibilities to make our model. Even the sub-steps, or components of exploration are the same: define variables, make assumptions, and develop model. However, what is unique, is what we do within those steps.

3.1.1 Define Variables

We change the variables in the LP model to influence the performance of the system [3]. Another way to think about this is that the variables are representative of a decision that needs to be made regarding a certain activity

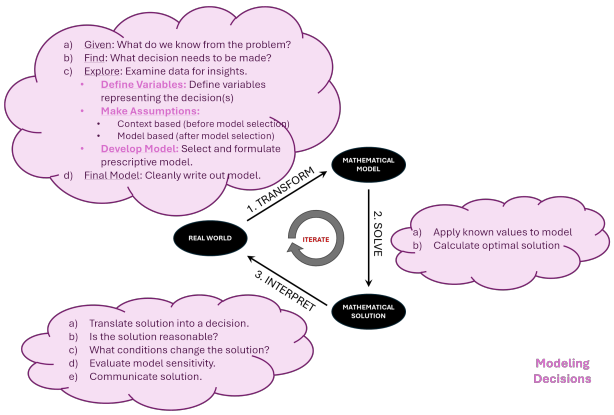

Figure 1: The mathematical modeling triangle as it is applied to prescriptive analytics and specifically linear programming. The pink writing identifies where the modeling decisions are made.

[2]. For example, if we are trying to identify the number of windows and doors to make to maximize profit, our variables should be the number of windows produced and the number or doors produced. In this way, our variables represent the decision of the number of each product to make. Within linear programming, understanding the decision to be made is essential to defining variables and eventually solving the problem.

3.1.2 Make Assumptions

There are two different types of assumptions we need to consider throughout prescriptive analytics, assumptions about the context of the problem and assumptions based on the model we select.

Assumptions made about the context of the problem are generally made prior to selecting a model. These include how we frame the objective, how we determine parameter values, and what kind of environment the decision is being made in. These context-based assumptions help us shape the model and prepare us to analyze the appropriateness of model-based assumptions.

Model-based assumptions are assumptions that need to be made after selecting the model type. In this instance, the model type we select is linear programming. Linear programming problems have four assumptions that must be made to use the model.

- **Proportionality**. The proportionality assumption states that the contribution of all variables in the objective function and constraints is linearly proportional to the value of the variable. This means that the effect of a decision, regardless of what it is, changes in direct proportion to how much of that decision you have [3]. For example, if making one door costs $10 and uses 2 hours of labor, then making 10 doors costs $100 and uses 20 hours of labor. There are no bulk discounts, set up fees, and no increasing

returns [2].

- **Additivity**. The additivity assumption is that each decision contributes to the total independently of the others [3]. For example, if building a door costs \$10 and building a window costs \$20, then building one of each costs \$30. There is no interaction between decisions; meaning that if you choose to make both there is not a discount and choosing to make a door does not make it more difficult to make a window [2]. This also means that each decision has a fixed contribution, regardless of what else happens. The impact of making 5 doors is the same whether you make 0 or 100 windows.

- **Continuous-Variable Assumption**. Also known as the divisibility assumption, the continuous-variable assumption requires that each decision variable be allowed to assume fractional values [3]. This means that we can do part of a task, like produce 3.2 windows. Of course, in the real world, some things can't be split. I can't assign half a person or send 2.6 trucks. Many LP problems ask us to determine an optimal rate, for example 3.2 windows per week instead of 3.2 windows, which can be fractional. This allows us to make sense of a fractional answer: we might complete part of a window in one week and finish it the next.

  Sometimes the continuous-variable assumption is still reasonable if fractional results can be rounded without changing the decision by much [3]. For example, if the solution is to make 3.2 windows, it may make sense to round down to 3 or up to 4, depending on context. However, it is important to recognize that sometimes this assumption won't always be appropriate. If rounding leads to a major change in feasibility or cost, or if dealing with small numbers where every unit counts, then a linear program may not be the right tool [3]. In those cases, more advanced techniques are used, but often LP provides a useful and practical approximation.

- **Certainty**. The certainty assumption specifies that all the parameters, or numbers being used in the model, are known with certainty [3]. That doesn't mean that these values will be constant forever, but that we know them well enough to treat them as fixed while making the decision [2]. For example, if we think it takes 2 hours of labor to produce a door, we assume that number is accurate and dependable when solving the model. We are not modeling variability in those estimates; we are using best estimates or averages as if they are exact. This is often a reasonable simplification, but it's important to recognize that it leaves out risk, fluctuation, and unexpected changes. If the results are very sensitive to the parameters being changed, then a different modeling approach may be needed.

3.1.3 Develop Model

Before we dive in: You're about to see quite a bit of math notation, some of it may feel unfamiliar or even overwhelming. That's okay. We don't expect you to fully understand everything right now. This section is here to give you a high-level preview and a reference you can return to later. In the next several lessons, we'll break down each part of this structure: what it means, how to build it, and how to use it to solve real-world problems. For now, just focus on the big picture and start getting comfortable with the language of optimization.

A linear programming problem (LP) is an optimization model for which we do the following:

1. We attempt to minimize (or maximize) a *linear* function of the variables [3].

2. The values of the variables must satisfy a set of constraints. Each constraint must be a linear equation or linear inequality [3].

3. A sign restriction is associated with each variable, meaning each variable is either constrained to be nonnegative or left unrestricted [3]. We will most often restrict the sign to be nonnegative.

The general form of a minimization problem is written as [1]:

$$\begin{aligned} \text{Minimize} \quad & \sum_{j=1}^{n} c_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \qquad i = 1, 2, \ldots, m \\ & x_j \geq 0, \qquad j = 1, 2, \ldots, n \end{aligned}$$

In this formulation, Minimize $\sum_{j=1}^{n} c_j x_j$, or equivalently $c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$, is the objective function, representing what we want to minimize [1]. The coefficients $c_1, c_2, \ldots, c_n$ are the cost coefficients and $x_1, x_2, \ldots, x_n$ are the variables [1]. Note that while we generally use the term *cost* coefficient within the objective function, the coefficient may represent other quantities such as time, energy, or risk, depending on the context.

Definition 3.1.1 (Objective Function [2])
The mathematical expression that represents the goal of the model, typically something to be maximized (profit or efficiency) or minimized (cost or time).

The *subject to* begins the specification of constraints on the variables. The inequality $\sum_{j=1}^{n} a_{ij} x_j \geq b_i$ represents the ith constraint in the LP formulation.

Definition 3.1.2 (Constraint [3])
A function or inequality that limits or restricts the possible values of the variables.

The coefficients a_{ij} for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$ are the technological coefficients that describe how much of resource i is used by one unit of variable j [1]. The values b_i represent the total amount of resource i available, or required, and must be satisfied by any solution [1]. Finally, the constraints $x_1, x_2, \ldots, x_n \geq 0$ are the nonnegativity constraints [1].

Note: In this step, we define the structure of the model using symbolic parameters (such as c_j, a_{ij}, and b_i), which represent the relationships and resources in abstract terms. We do not yet substitute specific numerical values. Instead, we focus on expressing the real-world problem in mathematical terms that can be solved later. The solve step begins when we substitute known values for these parameters to analyze or optimize the model.

Using our knowledge of vectors and matrices we can rewrite each of these components as vectors and matrices [1]: $\vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ is the variable vector, $\vec{c} = [c_1 \quad \ldots \quad c_n]$ is the cost vector, $\vec{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$ is the the right-hand side vector, and

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix}$$

is called the constraint matrix that contains the technological coefficients [1].

We can therefore rewrite our formulation as:

$$\begin{aligned} \text{Minimize} \quad & \vec{c}\,\vec{x} \\ \text{subject to} \quad & \boldsymbol{A}\vec{x} \geq \vec{b} \\ & \vec{x} \geq \vec{0} \end{aligned}$$

We can use a similar formulation for maximization problems. We will explore this structure, and dive into more detail on each component of the formulation, in the next several readings.

Note: a keen observer may find this constraint notation a bit nebulous. Let's take a closer look at the left and right sides of our constraint expression. The left side $A\vec{x}$ is the product of a m x n matrix and an n x 1 column vector which results in an m x 1 column vector. The right side of our constraint expression is also an m x 1 column vector. What we mean when we say that $A\vec{x} \geq \vec{b}$ is that each individual component of the column vector that results from $A\vec{x}$ is greater or equal to its corresponding component in $\vec{b}$. This is written in matrix and vector notation below.

If

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix},$$

and we let

$$\boldsymbol{A_1} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \end{bmatrix},$$

and

$$\boldsymbol{A_2} = \begin{bmatrix} a_{21} & a_{22} & \ldots & a_{2n} \end{bmatrix},$$

and so on. Then,

$$\boldsymbol{A} = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_m \end{bmatrix},$$

and the constraint equations become

$$\begin{aligned} \boldsymbol{A}_1\vec{x} &\geq b_1, \\ \boldsymbol{A}_2\vec{x} &\geq b_2, \\ &\vdots \\ \boldsymbol{A}_m\vec{x} &\geq b_m. \end{aligned}$$

3.2 Solve

Once we have developed our generic model, we are going to apply the parameters to the model and solve one of two ways: graphically or using the Solver package in Excel.

3.3 Interpret

During the interpret step you will translate your solution into a decision, then provide some analysis of the decision. Is it reasonable? What conditions change your solution? What is your model sensitivity? Finally, you'll communicate your solution.

4 Ethical Checklist

Remember that the ethical checklist we use requires three things: data validity, model validity, and clear communication. Below are some questions to think through as you consider whether your model is ethical.

- **Data validity**. Are the cost coefficients and technological coefficients realistic and justifiable? Where did those values come from?

- **Model validity**. Do the assumptions of linear programming hold in the real-world? If we've chosen to round our solution, how does that impact feasibility? Did we simplify a non-linear relationship inappropriately? Did I make an assumption that may not be true?

- **Communication**. Have I clearly specified my assumptions? Have I explained the tradeoffs involved in the decision and how sensitive the results are to changes in key values?

References

[1] Mokhtar S. Bazaraa, John J. Jarvis, and Hanif D. Sherali. *Linear Programming and Network Flows*. Wiley, 2010.

[2] Frederick Hillier et al. *MA103 Mathematical Modeling: Introduction to Management, Science, & Business Analytics with Connect*. McGraw-Hill, 2024.

[3] Wayne Winston and Munirpallam Venkataramanan. *Introduction to Mathematical Programming*. Brooks/Cole, 2003.