Chapter 11. Simulation as a path to understanding

Steve Elston

12/16/2020

Introduction

Simulation enables data scientists to study the behavior of stochastic processes with complex probability distributions. Simple processes might be approximated by a known, or 'named' distribution. In these simple cases, it might even be possible to derive analytical results. However, most real-world processes have complex behavior, resulting in complex distributions of output values. In these cases, simulation is a practical approach to understanding these processes. The two main purposes of simulation can be summarized as:
- **Testing models:** If data simulated from the model do not resemble the original data, something is likely wrong with the model.
- **Understand processes with complex probability distributions:** In these cases, simulation provides a powerful and flexible computational technique to understand behavior.

As cheap computational power has become ubiquitous, simulation has become a widely used technique in the data scientist's tool box. Simulations compute a large number of cases, or realizations. The computing cost of each realization must be low in any practical simulation.

The realizations are drawn from complex probability distributions of the process being studied. In many cases, the realizations are computed using conditional probability distributions. The final or posterior distribution of the process being simulated is comprised of these realizations.

Creating simulations

Creating, testing and debugging simulation software can be tricky. Some techniques which can make your life easier are the same as you would use when developing any analytical software, or even software in general. But, given the stochastic nature of simulation, testing and debugging can be more difficult than other types of software. Some commonly employed techniques which will help you include:
- Build your simulation as a series of small, easily tested chunks. The overall simulation will comprise many of these small chunks, typically in the form of functions.
- Test each small functional unit individually. These tests should include at least testing some typical cases, as well as boundary or extreme cases. Sensible behavior with extreme or limiting cases is a requirement for a stable simulation. Both tabular and graphical output can be useful for evaluating these tests.
- Test your overall simulation each time you add a new functional component. This processes ensures that all the components work together. Attempts at *'big bang'* integration typically end in failure and confusion, with considerable waste of time.
- Simulations are inherently stochastic. If you want to create identical numerical results, say for automated testing, set a seed before you begin tests. It is impossible to create and maintain quality simulation code without having detailed reproducible test cases.

The Scenario

Simulation employs several key concepts; conditional probability distributions and sampling theory. In this chapter you will perform a series of exercises to simulate the profitability of a sandwich shop. Not surprisingly, the sandwich shop earns money every time a customer buys a sandwich. However, the inputs to the sandwich cost money. The daily profit is the amount customers pay for the sandwiches minus the costs of the inputs.

The cost of bread is an input which is particularly difficult to manage. The shop bakes its own bread before it opens, and the bread must be used on the day it is made. The customers can select one of three types of bread, white, wheat, and multigrain. The customers are unusually picky. If the shop does not have the bread of the customer's choice available, the customer will leave the shop without buying a sandwich. However, any extra bread left at the end of the day is discarded, and the cost reduces the profitability of the shop.

To keep this simulation simple, several assumptions are made:
- The number of customers arriving at the shop each day is stationary with time and known to have a mean rate of 100 per day.
- The probability that each customer chooses a particular type of bread is stationary and known. These probabilities are 50% for white bread, 25% for wheat and 25% for multigrain.
- If a customer's choice of bread is not available the customer leaves the shop without buying a sandwich.
- If the customer buys a sandwich the mean profit to the shop is $1.00.
- The only perishable input which must be discarded at the end of each day is the bread. The discarded bread has a cost of $0.25.
- Customers do not stop coming to the sandwich shop as a result of not finding their bread choice. In other words, the number of arrivals is independent of how many customers actually buy a sandwich.

From this problem description it is clear there is a trade-off between profitability of the shop and the amount of bread baked. If the shop manager order too little bread, opportunities for profitable sandwich sales are lost. On the other hand, over-production of bread leads to loss of profit from the waste.

In reality these are questionable assumptions, and a real-world situation would be more complex. The simulation techniques used here can be applied to much more complex and realistic simulations.

Representation as Directed Acyclic Graphical Model

When creating a simulation with multiple conditionally dependent variables it is useful to draw a directed graph.. Such a representation is known as a **directed acyclic graphical model or DAG** The graph is a communications device showing which variables are independent and which are conditionally dependent on others. The shapes used in the graph help with understanding.
1. **Probability distributions** of the variables are shown as ellipses. Conditional distributions have **directed edges** leading to them, which specify the dependencies. These distributions have parameters which must be estimated.
2. **Decision variables** are deterministic and are shown as rectangles. Decisions are determined by variables. Setting decision variables can be performed either manually or automatically.
3. **Utility nodes**, profit in this case, are shown as diamonds. These nodes represent a **utility function** given the dependencies in the graphs. Utility calculations are deterministic given the input values.
4. **Directed edges** show the dependency structure of the distributions. The arrows point to **child nodes** which are dependent on **parent nodes** where the arrows originate.

Values of decision variables are selected to optimize the utility. The directed graph for this simulation is shown in the figure below.

Exercise: Answer the following questions about the graph:
1. Which are the independent variable(s) in the graph? Are the independent varaiables the only ones for which parameters must be estiamted or known apriori.
2. Which are the dependent variable(s) in the graph?
3. Which is the decision variable?
4. Given the values of the other varibles, is profit deterministic and why?

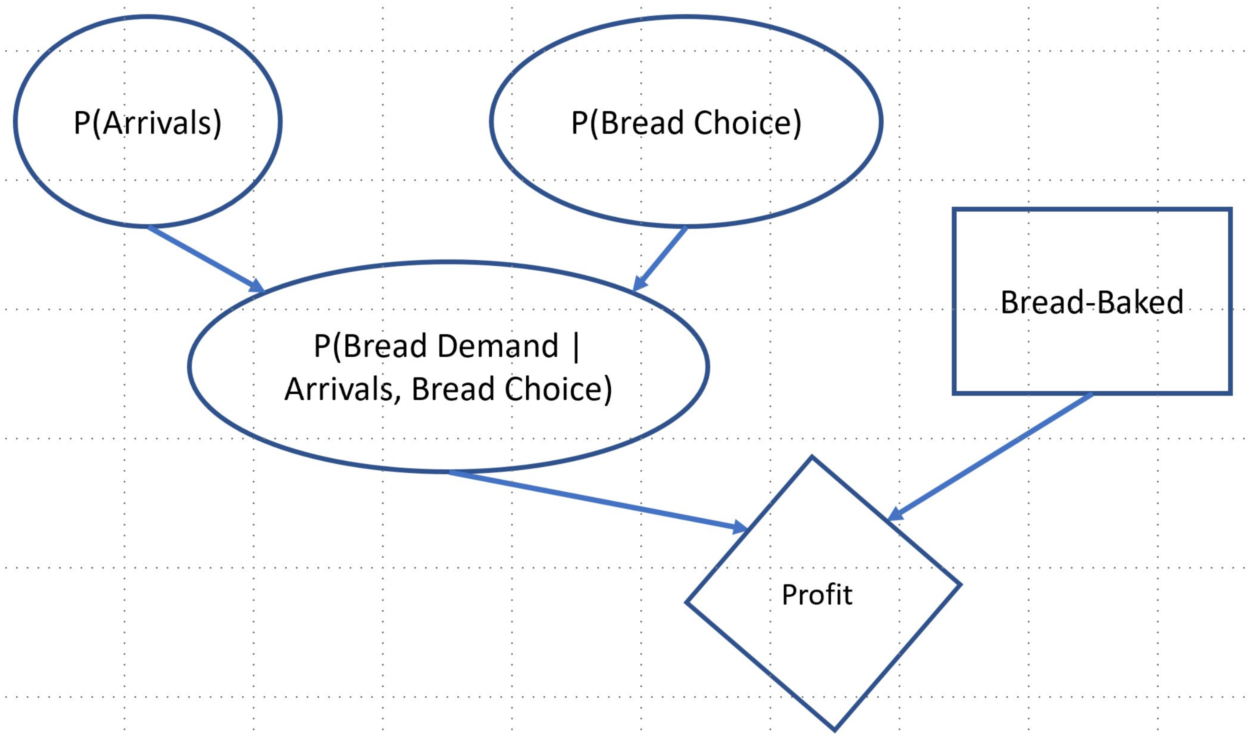

Figure 1: Directed graph of the distributions for profit simulation

Realizations of Distribution

The ability to compute large numbers of realizations from probability distributions is fundamental to simulation. Several questions naturally arise.
1. Which distribution to choose? The answer to this question is typically determined through a combination of domain knowledge and significant data exploration. Usually, several choices are tested and compared.
2. Are the distributions of the variables conditionally dependent? That is, do we need to know the distribution of some **parent variable** before we can compute the distribution of a **dependent child variable**.
3. How are the parameters of the distributions determined? Typically, maximum likelihood estimation is applied to observations of the process to be simulated. If the process is non-stationary more complex methods, beyond the scope of our discussion, are required.
4. How many realizations are required? The answer will depend on the accuracy you require from your simulation and how patient you are. Typically, some tests will indicate an appropriate number of realizations.

The following code loads the suggested packages and configures the Seaborn graphics style.

```
## Load packages we will need
import numpy as np
import numpy.random as nr
import pandas as pd
import scipy
import scipy.stats as ss
import matplotlib.pyplot as plt
import seaborn as sns
# Configure default plot style.
sns.set_palette('muted')
sns.set_style('ticks')
```

Building the Simulation

Follow the steps in this section to build, test and evaluate a simulation of sandwich shop profitability. The goal is to build a simple decision tool that will help the shop manager determine how much bread to bake each day.

> **Coding note:** The exercises described in this section lead to construction a of simulation using functions. This approach has some advantages for statistical computing, since there is a natural correspondance with statical functions. However, this approach can lead to ackward function calls with a great many arguments. Some encapsulation inherent in an object model reduces this complexity. If you have skills in object oriented design, you may wish to undertake these exercises using an object system.

Poisson Distribution for Arrival Rates

Poisson distributions are often used to model arrival processes. The Poisson distribution has a single parameter, the arrival rate or intensity, λ, per time period. We have reviewed key properties of the Poisson distribution in Chapter XXXXXXX. We can express the density function of arrivals as:

$$f(arrivals) = Pos(\lambda)$$

> **Exercise:** You will now create and test a function, named `customer_arrivals`, to simulate the daily number of customer arrivals at the sandwich shop. The argument to this function is the mean arrival rate and the function should return the integer number of arrivals for each daily realization. Include some test cases for your function and write an explaination of the tests.

Simulate bread choice

The demand for the bread is simulated from a **categorical distribution**. Recalling the discussion in Chapter XXXX, for each customer arrival a choice of bread is made with probabilities, $\Pi = [\pi_1, \pi_2, \pi_3] = [0.5, 0.25, 0.25]$, for $[white, wheat, multigrain]$. We express density function of bread choice as:

$$f(white, wheat, multigrain) = Cat(\Pi) \quad (1)$$
$$\Pi = [\pi_1, \pi_2, \pi_3] \quad (2)$$

> **Exercise:** Create and test a function, named `bread_choice`, to simulate arriving customers' bread demand. The function takes three arguments, the number of arrivals, a list of bread choices, and the probabilities of those choices. The function returns a bread choice as a Numpy array of strings. Include some test cases for your function and write an explaination of the tests. In particular, test some extreme cases, such as where some $\pi_i = 0$.

> **Computational Note:** To create the function described you can use the numpy.random.choice function on the list of bread choices. The size argument is the number of customer arrivals, in this case. Thus, all the bread choices for a given day are made with one function call. In general, computing a vector of results with a single call is a computationally efficient approach. Many computational libraries, including Numpy, are optimized for **vectorized** calls.

Simulate bread demand

Before you can compute daily profitability you must simulate bread demand for each day. The probabilities of demand for each type of bread are conditional on the number of arrivals and the bread choices. In mathematical terms we can write this relationship as $P(demand \mid arrivals, choice)$.

> **Exercise:** To compute this conditional distribution, you will write a function `bread_demand` with arguments of mean arrival rate of customers, the choices of breads and the probabilities of bread choice. The funciton will can the `customer_arrivals` and `bread_choice` funcitons. The function will return a numpy array of demand by bread type. Include some test cases for your function and write an explaination of the tests.

> **Exercise:** The conditional distribution of the bread demand might not be obvious at first. To get a feel for this distribution you will now create histograms for the demand of each bread type over an approximately 4-year period. There are about 250 business days in a year, so that the histograms show the distribution of bread demand for 1,000 days. Make sure these histograms have a reasonable aspect ratio (not elongated), and have proper labels. Once you display the histograms examine the results. Recall that a Poisson distribution approaches the Normal for a large mean arrival rate. Do these histograms appear approximately Normal? Is this behavior expected, given that the distribution is conditional on a Poisson process and a uniformally distributed process?

Simulate Bread Baked

The number of each type of bread baked in the sandwich shop is deterministic. The shop manager has a plan for the day, and the bread is baked in advance of the customer arrivals. Naturally, the manager will choose the proportion of each bread type based on the historical expected demand. This is a decision function. The decision to be made is how many total loaves to bake?

> **Exercise:** Create a function `bread_baked` with aguments, total number of loaves, proportion of each bread type to bake, and the types of breads. The function will return a list (or Numpy array) with the number of each type of bread baked. Include some test cases for your function and write an explaination of the tests.

Simulate and plot profit

The last function you need to execute the full model represented by the DAG computes the daily profit. This function is therefore the utility function. The calculation is deterministic, given the inputs of bread demand and the amount of bread baked. But don't be confused, while the utility function is deterministic, the bread demands is stochastic. As a result, profit is a random variable.

> **Exercise:** You will now write the function, `daily_profit`, that computes and returns the daily profit. The function calls the `bread_demand` and `compute_bread_bake` functions. The arguments to this function are:
> - The daily total of bread baked,
> - Proportion of each bread type baked daily,
> - Mean daily arrival rate,
> - Probabilities of bread choices,
> - Amount earned per sandwich sold,
> - Cost of discarded bread,
> - Bread choices.

> **Exercise:** The distribution of daily profit is, as yet, unknown. Further, you will need to test your `daily_profit` function. In order to work towad these goals, you will now simulate a year's (250 business days) profits and plot the result as a histogram. Use a mean arrival rate of 100 and 100 breads baked per day. Examine your histogram:
> - Does the distribution of profit resemble the distribution of arrivals, or the distribuiton of bread bakced?
> - Does this result indicate that it can mostly likely only be found by a computational method?

Stochastic Nature of Simulation

The key question of interest to the manager of the sandwich shop is the number of loaves of bread to bake to simulate profitability. Thus, there is only one decision variable in this case. We should be able to search a reasonable space of values to find the one which optimizes the profit of the shop.

But, there is still the issue of how much the results of this simulation vary from run to run. This means, there is considerable uncertainty or variability in the daily profit. Running a simulation multiple times yields the **posterior distribution** of the variables being simulated. In this case we are interested in the posterior distribution of the profitability vs. the number of breads baked.

> **Exercise:** Running the simulation of daily profit for different numbers of bread baked will help to quantify the choice and uncertainty in the number of breads to bake each day. You will perform the simulations for 4 business years (1000 days) and for number of bread baked $= [60, 80, 100, 120, 140, 160]$. Therefore, you will do the following:
> 1. Create an empty Pandas data frame columns, `num_baked`, and `profit`. The number of rows is the different numbers of bread baked by the number of realizations of profit for each quantity of bread.
> 2. Iteratie over the number of bread baked. For each number of breads, compute 1000 realizations of customer arrivals along with the profit for the sandwich shop.
> 3. Create a box plot showing the realizations of profit for each number of breads baked.
> 4. Group the data frame by `num_baked` and then apply the `describe` method.
>
> Answer these questions:
> - How does the variability of daily profit change with the number of breads baked?
> - Which number of bread baked have the highest mean and median daily profits?
> - Given these mean and median values and the risk of low daily profits seen in the variabilit, y which number of bread baked should the shop manager choose and why?

Extensions of the Simulation

The foregoing simulation has given some insight into the relationship of bread baked and profitability of the sandwich shop. But of course, the real-world is much more complicated. Some examples of real-world complexity that could be accounted for in a more complete simulation include:
- The arrival rate of customers may change with the day of the week or the season of the year.
- Other inventory items, besides bread, that are also perishable and have costs.
- The labor costs for the shop can be variable depending on a finer grain time scale. For example, if most customers arrive within a short window of time more employees are required to meet the surge in demand.

> **Exercise:** Consider some other real-world complicaitons which could be accounted for in a more complete simulation. What are they? How could you model these factors?

Bibliography