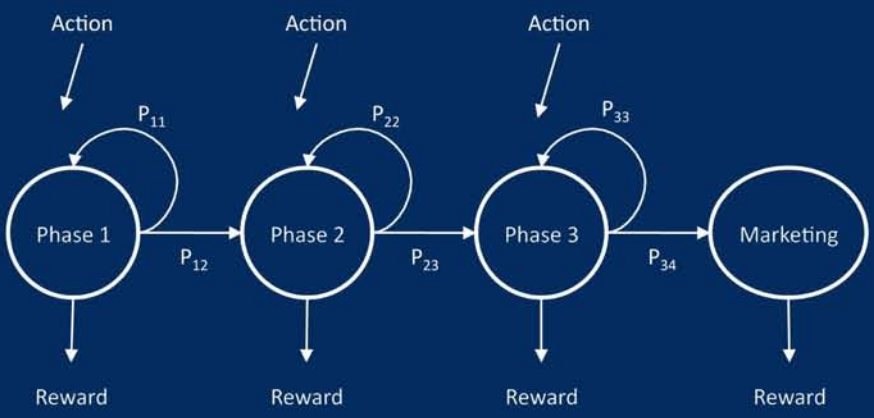
Chapman & Hall/CRC Biostatistics Series

Monte Carlo Simulation for the Pharmaceutical Industry

Concepts, Algorithms, and Case Studies



Mark Chang

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Monte Carlo Simulation for the Pharmaceutical Industry

Concepts, Algorithms, and Case Studies

Chapman & Hall/CRC Biostatistics Series

Chapman & Hall/CRC Biostatistics Series

Published Titles

1. *Design and Analysis of Animal Studies in Pharmaceutical Development,* Shein-Chung Chow and Jen-pei Liu
2. *Basic Statistics and Pharmaceutical Statistical Applications,* James E. De Muth
3. *Design and Analysis of Bioavailability and Bioequivalence Studies, Second Edition, Revised and Expanded,* Shein-Chung Chow and Jen-pei Liu
4. *Meta-Analysis in Medicine and Health Policy,* Dalene K. Stangl and Donald A. Berry
5. *Generalized Linear Models: A Bayesian Perspective,* Dipak K. Dey, Sujit K. Ghosh, and Bani K. Mallick
6. *Difference Equations with Public Health Applications,* Lemuel A. Moyé and Asha Seth Kapadia
7. *Medical Biostatistics,* Abhaya Indrayan and Sanjeev B. Sarmukaddam
8. *Statistical Methods for Clinical Trials,* Mark X. Norleans
9. *Causal Analysis in Biomedicine and Epidemiology: Based on Minimal Sufficient Causation,* Mikel Aickin
10. *Statistics in Drug Research: Methodologies and Recent Developments,* Shein-Chung Chow and Jun Shao
11. *Sample Size Calculations in Clinical Research,* Shein-Chung Chow, Jun Shao, and Hansheng Wang
12. *Applied Statistical Design for the Researcher,* Daryl S. Paulson
13. *Advances in Clinical Trial Biostatistics,* Nancy L. Geller
14. *Statistics in the Pharmaceutical Industry, Third Edition,* Ralph Buncher and Jia-Yeong Tsay
15. *DNA Microarrays and Related Genomics Techniques: Design, Analysis, and Interpretation of Experiments,* David B. Allsion, Grier P. Page, T. Mark Beasley, and Jode W. Edwards
16. *Basic Statistics and Pharmaceutical Statistical Applications, Second Edition,* James E. De Muth
17. *Adaptive Design Methods in Clinical Trials,* Shein-Chung Chow and Mark Chang
18. *Handbook of Regression and Modeling: Applications for the Clinical and Pharmaceutical Industries,* Daryl S. Paulson
19. *Statistical Design and Analysis of Stability Studies,* Shein-Chung Chow
20. *Sample Size Calculations in Clinical Research, Second Edition,* Shein-Chung Chow, Jun Shao, and Hansheng Wang
21. *Elementary Bayesian Biostatistics,* Lemuel A. Moyé
22. *Adaptive Design Theory and Implementation Using SAS and R,* Mark Chang
23. *Computational Pharmacokinetics,* Anders Källén
24. *Computational Methods in Biomedical Research,* Ravindra Khattree and Dayanand N. Naik
25. *Medical Biostatistics, Second Edition,* A. Indrayan
26. *DNA Methylation Microarrays: Experimental Design and Statistical Analysis,* Sun-Chong Wang and Arturas Petronis
27. *Design and Analysis of Bioavailability and Bioequivalence Studies, Third Edition,* Shein-Chung Chow and Jen-pei Liu
28. *Translational Medicine: Strategies and Statistical Methods,* Dennis Cosmatos and Shein-Chung Chow
29. *Bayesian Methods for Measures of Agreement,* Lyle D. Broemeling
30. *Data and Safety Monitoring Committees in Clinical Trials,* Jay Herson
31. *Design and Analysis of Clinical Trials with Time-to-Event Endpoints,* Karl E. Peace
32. *Bayesian Missing Data Problems: EM, Data Augmentation and Noniterative Computation,* Ming T. Tan, Guo-Liang Tian, and Kai Wang Ng
33. *Multiple Testing Problems in Pharmaceutical Statistics,* Alex Dmitrienko, Ajit C. Tamhane, and Frank Bretz
34. *Bayesian Modeling in Bioinformatics,* Dipak K. Dey, Samiran Ghosh, and Bani K. Mallick
35. *Clinical Trial Methodology,* Karl E. Peace and Ding-Geng (Din) Chen
36. *Monte Carlo Simulation for the Pharmaceutical Industry: Concepts, Algorithms, and Case Studies,* Mark Chang

Chapman & Hall/CRC Biostatistics Series

Monte Carlo Simulation for the Pharmaceutical Industry

Concepts, Algorithms, and Case Studies

Mark Chang

Amag Pharmaceuticals

Lexington, Massachusetts, U.S.A.

CRC Press

Taylor & Francis Group

Boca Raton London New York

CRC Press is an imprint of the

Taylor & Francis Group, an **informa** business

A CHAPMAN & HALL BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Version Date: 20140515

International Standard Book Number-13: 978-1-4398-3593-7 (eBook - PDF)

To my teachers and my family.

Contents

Preface

Drug development, aimed at improving people's health, becomes more costly every year. The pharmaceutical industry must join its efforts with government and the health professions to seek new, innovative, and cost-effective approaches in the development process. During this evolutionary process over the next decades, computer simulations will no doubt play a critical role. Computer simulation or Monte Carlo simulation is the technique of simulating a dynamic system or process using a computer program. Computer simulations, as an efficient and effective research tool, have been used in virtually every area of engineering, science, mathematics, etc.

In this book, I present *concepts*, *theories*, *algorithms*, and *cases studies* of Monte Carlo simulation in the pharmaceutical and health industries. The concepts refer not only to simulation in general, but also to various types of simulations in drug development. The theory will include virtual data sampling, game theory, deterministic and stochastic decision theories, adaptive design methods, Petri net, genetic programming, resampling methods, and other strategies. These theories and methods are necessary either to carry out the simulations or to make the simulations more efficient, even though there are many practical problems that can be simulated directly in an ad hoc fashion without any theory about their efficiency or convergence considerations. The algorithms, which can be descriptive, computer pseudocode, or a combination of both, provide the basis for implementation of simulation methods. The case studies or applications are simplified versions of real-world problems. These simplifications are necessary because a single case could otherwise occupy the whole book, preventing readers from exploring broad issues. There are also examples of how simulation can be formulated to address interesting questions that have not been studied before. In my view, building simulation models is, to a large extent, a creative process; it is an innovation in application, which can impact our lives more positively than other innovations or creations. Simulation often

requires knowledge in several disciplines, including mathematics/statistics, programming, and the subject field.

The overarching goals of this book are not limited just to the scope of simulation methodologies and computer algorithms, but also include sharing personal thoughts, experiences, and views about how to become a visionary, to be creative, a logical thinker and, a skillful "simulator." Being visionary can be helpful in formulating the questions and building simulation models; on the other hand, using simulations we can answer visionary questions. Being visionary requires having a big-picture view and understanding the issues profoundly. Being visionary is helpful in facilitating communications and getting one's ideas across to key stockholders in a company. This belief is reflected in the chapters on meta-simulation and macro-simulation, and throughout the book. Being creative requires wide knowledge in the field and cross-disciplines and skill in identifying the similarities among different things and making analogies. For this reason, I decided to cover a broad range of problems in drug development and to provide introductory examples from many different fields. Being logical implies being capable of abstract thought. Simulation processes often appear to be intuitive and concrete. However, to build the model or convert a practical problem into a simulation problem and develop an algorithm require a high degree of abstract thinking and the ability to visualize all the steps in the simulation process. After the algorithm is developed, implementation using any computer language or software is a relatively straightforward task if one knows the programming language well.

Many academic and industry professionals, myself included, share the view that, despite our great efforts, there is still a gap between what students learn in school and what they actually need in their work. To narrow this gap, teaching materials or books that can bridge the gap between academia and industry are necessary. This book is an attempt in this direction. The style of the book is unique in the sense that it is a mix of textbook and monograph. Having said that, industry statisticians, scientists, and software engineers/programmers are intended to be its primary readers.

The second unique characteristic of this book is the broad coverage of subject fields from drug discovery (molecular design, disease modeling, and biological pathway simulation), preclinical aspects (pharmacokinetics, pharmacodynamics), clinical development (adaptive clinical trials, trial management, and execution), and prescription drug commercialization. In contrast, most monographs deal with only a very specific field.

Before discussing specific Monte Carlo techniques, background information, and the issues and/or trends in the field are covered. This is partic-

ularly important for Monte Carlo model building and for communicating simulation results back to the team or any concerned parties. You will be amazed when you find out, in a successful Monte Carlo project, how much more time and effort are expended on team communication than on actual simulations. Most simulation books are either completely mathematical/statistical oriented or resemble a software user manual, and neglect the background topics altogether.

The third unique feature is in the exercises. Typically, exercises in a textbook are provided with all the information for students to answer the questions. However, in the real world, especially in drug development, virtually every challenging problem requires that we make judgments on what information is needed and where to get it in order to solve the problem at hand, and most times appropriate assumptions are also required. Because of the assumptions and uncertainties of source information, how to interpret the results also becomes an essential part of the task, which, from my experience, is often overlooked in the classroom and by students. In light of this, for many exercises in this book, readers are expected to make a judgment as to whether the information provided is sufficient or not; if not, they must identify and use other sources or make appropriate assumptions, and then finally use the methods/tools discussed to solve the problem.

Road Map

The book studies a broad category of computer simulations, virtually covering the whole spectrum of drug development in the thirteen chapters.

Chapter 1, Simulation, Simulation Everywhere, covers the general concepts of simulations, emphasizing the importance of analogy and simulation using various examples from daily life, art, music, bilingualism, strategies for commuting, economics, math, science, finance, optimization, and others.

Chapter 2, Virtual Sampling Techniques, discusses general methods for the generation of random numbers, methods for variance reduction, and methods for generating random numbers from specific distributions. The discussions are brief, especially on methods for specific distributions.

Chapter 3, Overview of Drug Development, provides basic knowledge about different stages of drug development, from discovery through preclinical and clinical development. This knowledge is important to readers who have little knowledge about drug development and want to develop the ability to confidently model a practical problem as a Monte Carlo problem.

Simulations are divided into meta, macro, and micro simulations. Chapter 4, Meta-Simulation for the Pharmaceutical Industry, investigates the characteristics of competition and collaboration between independent business entities or pharmaceutical companies using simulations, including pharmaceutical gaming and prescription drug global pricing. Those sim-

ulation methods are constructed on the basis of game theory; therefore, the chapter also serves as an introduction to game theory.

Chapter 5, Meta-Simulation for Pharmaceutical Research and Development, studies meta-simulation, in which different stages of drug development and different drug candidates are considered simultaneously in simulation models. The simulation approaches are constructed based on the Markov decision process. The chapter provides materials to smoothly transfer from a deterministic sequential decision process, to a Markov decision process, to a pharmaceutical decision process, and to extensions of Markov decision processes.

Chapter 6, Clinical Trial Simulations (CTS), deals with simulations in classical and adaptive trials, which is of the most interest to the pharmaceutical industry and where simulation finds most of its applications in drug development. Unified adaptive design methods are also introduced.

Chapter 7, Clinical Trial Management and Execution, focuses on the various challenges in clinical trial management and execution (e.g., trial management, patient recruitment, adaptive randomization, dynamic drug supply, and adaptive trial monitoring) and the reformulation of the problems into Monte Carlo simulation in conjunction with other theoretical methods such as critical path analysis.

Chapter 8, Prescription Drug Commercialization, covers the dynamics of prescription drug marketing, the stock-flow dynamic model for brand planning, and a competitive drug marketing strategy. Different models such as a stochastic market game are discussed in the simulations.

Chapter 9, Molecular Design and Simulation, comprises various topics including why molecular design and simulation, molecular similarity search, overview of molecular docking, small molecule conformation analysis, ligand–receptor interaction, docking algorithms, and scoring functions.

Chapter 10, Disease Modeling and Biological Pathway Simulation, discusses computational system biology, petri nets, and biological pathway simulation with PN.

Chapter 11, Pharmacokinetic Simulation, gives an overview of abortion, distribution, metabolism, and excretion (ADEM) and their modeling, especially physiologically based pharmacokinetic modeling.

Chapter 12, Pharmacodynamic Simulation, provides an overview of pharmacodynamics, enzyme kinetics, pharmacodynamic models, drug–drug interactions, and case studies.

Chapter 13, Monte Carlo for Inference and Beyond, includes several important topics, including sorting algorithms, resampling methods, and genetic programming.

In the appendices, implementations of three algorithms, representing

easy, moderate, and difficult levels of coding, are presented as examples, in Javascript language. The code is embedded in an html file; thus only Internet Explorer is needed to run the programs. More algorithms are implemented and made available at www.statisticians.org.

The book deals with multiple disciplines employing different syntax conventions. There is a balance between being syntax consistent throughout the book and respecting syntax/conventions in the individual subject fields. Since the levels of Monte Carlo development in different areas of the pharmaceutical industry are very different, I try to not overstress consistency in the complexity level of the mathematical models or in the simulation technologies actually used. For example, clinical trial simulations are more developed, but simulation in prescription drug commercialization is relatively naive from a mathematical perspective. As a second example, the subject of molecular design and simulation is usually accomplished using commercial software, not because of the mathematical complexity but because of the necessarily lengthy coding for ligand-protein 3D geometry, molecular or quantum mechanics, and the software user interface. Thus, I spend less time in discussing the algorithms, but focus instead on the issues concerned when conducting these kinds of simulations.

I candidly admit that my goals are ambitiously high, and my approach in this book needs to be tested. Therefore, any comments or criticisms are encouraged. I can be reached at www.statisticians.org.

Finally, thanks to Dr. Robert Pierce; his valuable comments have greatly improved the manuscript. Thanks also to Acquisitions Editor David Grubbs from Taylor & Francis for providing me the opportunity to work on this project.

Mark Chang

Lexington, MA, USA
www.statisticians.org

Chapter 1

Simulation, Simulation Everywhere

This chapter will cover the following topics:

- Modeling and Simulation
- Introductory Monte Carlo Examples
- Simulations in Drug Development

1.1 Modeling and Simulation

1.1.1 *The Art of Simulations*

Simulation is the imitation of a physical or conceptual system (process) in order to gain insight into its functioning and to optimize its performance. The act of simulating generally entails representing certain key characteristics or behaviors of a selected physical or abstract system. Key issues in simulation include acquisition of valid source of information about the referent, selection of key characteristics and behaviors, the use of simplifying approximations and assumptions within the simulation, and fidelity and validity of the simulation outcomes.

A Monte Carlo method, also called Monte Carlo, Monte Carlo simulation, or computer simulation, is a technique that usually involves using computer-generated random numbers and theory of probability to solve problems. The term Monte Carlo method was coined by S. Ulam and Nicholas Metropolis in reference to games of chance, a popular attraction in Monte Carlo, Monaco (Hoffman, 1998; Metropolis and Ulam, 1949).

The term simulation is often associated with the term modeling. Modeling can be mechanical modeling, mathematical or statistical modeling, or analogy. Mathematical or statistical modeling of phenomena can have analytical closed form solutions or more often numerical solutions. For many complicated situations the solutions have to be obtained with the assistance

of computer simulation.

The terms modeling and simulation are often used interchangeably because they both use analogy. However, there are some fine differences — simulations often involve repetitions of virtual random data sets, whereas modeling often applies to an observed data set. In practice, modeling and simulation are often combined to effectively solve problems. In engineering, modeling can mimic behavior using a different scale from the original object being modeled. As an example, before constructing a large dam for a reservoir, engineers usually use a physical model (prototype) in 1:100 to 1:1000 scale in the laboratory and evaluate the safety of the dam as it undergoes various forces. They use the results to infer the performance of the actual dam to be built. This is reasonable because there are underlying relationships between the model and the original object. The theory of the study of the relationships is called dimensional analysis in mechanics (Szirtes, 2007). Spaceship launching is another example: before the launch, numerous tests or simulations have to be carried out.

Like many other fields, computer simulation can also be an art — there are many ways to achieve the goal, and some are better than others. In addition to the necessary mathematical/statistical and computer knowledge, the keys to a successful simulation are the ability to make analogies and a reasonably good knowledge of the subject field. If you can use logic and analogy to transfer the problem into a sensible Monte Carlo simulation, you are halfway to solving the problem. Given the importance of basic subject knowledge, I have included as a chapter in this book background materials for drug development, so that the reader can fully understand the Monte Carlo methods without frequently referring to other texts. Simulation in drug development often requires strong interaction among people from different disciplines, and this subject knowledge will help facilitate your communications. You will be amazed how much time and effort you need to put into collaboration — getting input for your model and convincing others of your proposal.

Before we discuss simulation techniques in the pharmaceutical and health industries, it is helpful to take a glance at some entertaining examples. These examples are selected because they are simple, but also because they demonstrate the amazing power of simulations.

1.1.2 *Genetic Programming in Art Simulation*

Simulation can be used in many fields (with or without mathematical modeling): science, engineering, finance, art, and music (Romero & Machdo, 2008; Cope, 2001). The portrait of Lisa Gherardini (the Mona Lisa) in the

Louvre in Paris, painted by Leonardo da Vinci during the Italian Renaissance, is perhaps the most famous and iconic painting in the world. Roger Alsing used a special Monte Carlo method called genetic programming (GP) to successfully reproduce the portrait with only 50 semi-transparent polygons (see Figure 1.1). The result is astonishing! One implication of this successful simulation using GP is that GP can be used to compress/zip the picture with truly tiny computer storage — only information containing 50 semi-transparent polygons in this case! We will return to GP later in this section.

Figure 1.1: The Mona Lisa Generated Using GP

1.1.3 *Artificial Neural Network in Music Machinery*

Virtual music represents a broad category of machine-created composition which attempts to replicate the style but not the actual notes of existing music (Cope, 1993). One of the first formal types of algorithms in music theory, and another good example of virtual music, is the eighteenth century Musikalisches Wurfelspiel, or musical dice game. The idea behind this musically sophisticated game involved composing a series of measures of music that could be recombined in many different ways and still be stylistically viable — virtual music. Following this process, even a very simple piece becomes a source of innumerable new works (Cope, 2001).

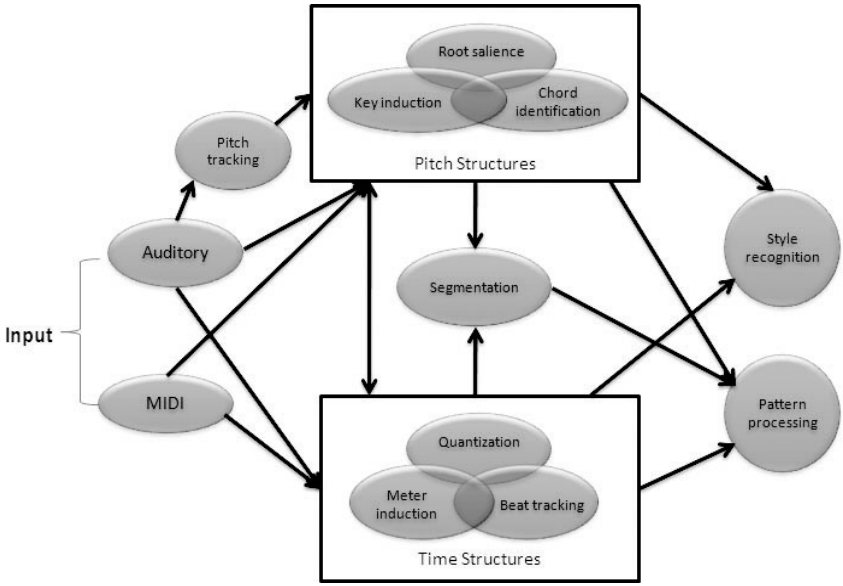


Figure 1.2: Music Scores and Structure Process for Monte Carlo (Source: Adapted from Rowe, 2001)

Rowe (2001) used an artificial neural network to simulate the subsymbolic process (Figure 1.2). Neural networks are a class of algorithms that learn relations between inputs and outputs. Their structure is derived from a schematic model of the neurons of the brain. A typical artificial neural network (ANN) consists of layers of connected nodes. A weight is placed on each connection in a neural network. Activation travels from one node to another across a connection. The weighted sum of activations from nodes at a previous layer is compared to a threshold to determine the activation of the current node.

The initial connection weights can be set arbitrarily. One of the great attractions of ANNs is that they are able to learn: the weights can be adjusted automatically from data. To accomplish this, a training set of input data with known answers attached is presented to the network. Over the course of a training session, the connection weights are gradually adjusted by the neural network itself until they reach convergence. If the training set captures the regularities of a wider class of inputs, the trained network will then be able to correctly classify inputs not found in the training set as well. Such a process is an example of supervised learning, in which a teacher (the training set) is used to guide the network in acquiring the necessary knowledge (connection weights).

After the ANN is well trained, it can be used as a music teacher to classify musical works in teaching and can even become a virtual musician, producing music pieces of various kinds. As far as the applications of ANN in drug development, we will discuss the topic in detail in later chapters of this book.

1.1.4 *Bilingual Bootstrapping in Word Translation*

Li and Li (2004) developed an effective machine learning technique (bilingual bootstrapping) for word translation disambiguation. It makes use of a small amount of classified data and a large amount of unclassified data in both the source and the target languages. It repeatedly constructs classifiers in the two languages in parallel and boosts the performance of the classifiers by classifying unclassified data in the two languages and by exchanging information regarding classified data between the two languages (Figure 1.3).

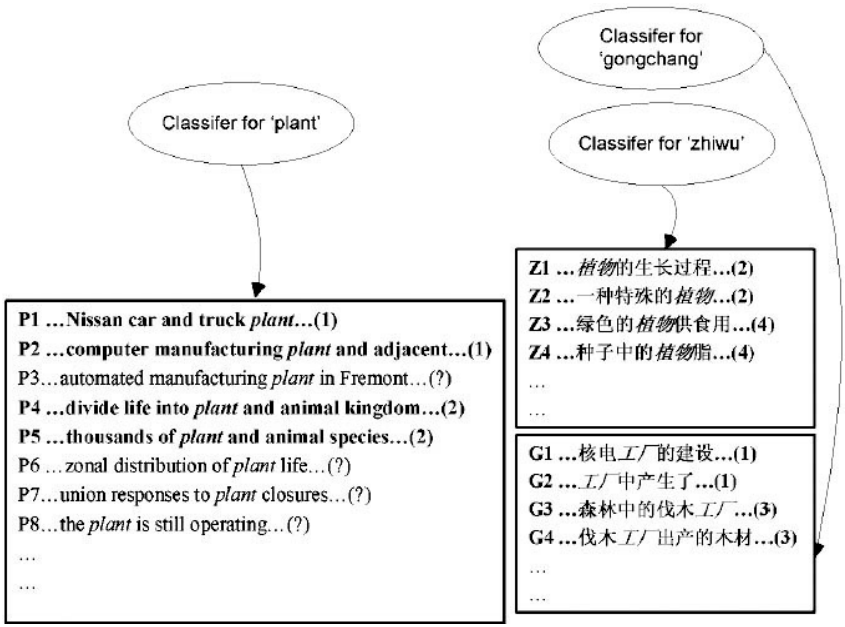


Figure 1.3: Bilingual Bootstrapping
(Source: Li and Li, 2004)

Bootstrapping is a statistical method for estimating the sampling distribution of an estimator by sampling with replacement from the original

sample, most often with the purpose of deriving robust estimates of standard errors and confidence intervals of a population parameter such as a mean, median, proportion, odds ratio, correlation coefficient, or regression coefficient. It can also be used for constructing hypothesis tests. It is often used as a robust alternative to an inference based on parametric assumptions when those assumptions are in doubt, or where parametric inference is impossible or requires very complicated formulas for the calculation of standard errors. When large data sets are available, we can use random sampling with or without replacement to validate the neural network method. We will discuss more about bootstrapping methods in Chapter 13.

1.2 Introductory Monte Carlo Examples

The best way to describe Monte Carlo simulation may be through examples. In the next section, we give some simple examples, walking through the steps taken to solve the problems using Monte Carlo simulation. Keep in mind that to solve problems effectively, Monte Carlo simulations are often combined with other modeling methods or theories, such as game and decision theories. In addition, basic knowledge of the subject field of the underlying problem is critical to the success of the simulation.

1.2.1 *USA Territory*

Suppose we are asked to find the size of the USA's territory. The only thing we have is a map (Figure 1.4), in 1:20000 scale, on a rectangular piece of paper of 24 in × 36 in. Can we accomplish the mission?

Let's analyze the situation: The map has an irregular shape and there seems no simple and direct way to calculate the size (area). However, if we can find the ratio of map size to rectangle size, the size of the USA's territory can be calculated using the following equation:

$$A_{usa} = RA_{reg}S_{map} = 20000RA_{reg} \tag{1.1}$$

where R is the ratio of the areas (USA over the rectangle) $\frac{A_{usa}}{A_{reg}}$; S_{map} is the scale of the map; A_{reg} is the area of the rectangle, which is easy to measure.

It is clear that the key to the problem is to find the ratio

$$R = \frac{A_{usa}}{A_{reg}}. \tag{1.2}$$

Among other alternatives, we can use mechanical simulation to estimate the ratio through the following steps: (1) Cut the map into many small

Figure 1.4: Map of the US

pieces randomly. Based on the dominant color on the piece, some of the pieces will be considered white and some gray (the map part). (2) Fully mix all the pieces. (3) Randomly draw a piece N times with replacement and record the number of gray pieces (N_{usa}). A simple probabilistic fact tells us that when $n \to \infty$, the following equation holds with a probability of 1:

$$R = \frac{A_{usa}}{A_{reg}} = \frac{N_{usa}}{N}. \tag{1.3}$$

But what if we don't want to destroy the map? Well, that is easy too. Get a small object, e.g., a needle; throw it randomly toward the map many times; record the number of times (N_{usa}) the needle tip falls on the USA map and the number of times (N) the needle tip falls on the entire rectangle; (1.3) and (1.1) are still valid. Note that the paper used to draw the map does not have to be a rectangle, but can be any shape. For convenience, we call the paper or area that covers the map the "majorization space."

1.2.2 π *Simulation*

We now consider how we can use computer simulation to perform this type of simulation more efficiently. To make it more interesting, let's estimate

the constant π. To this end, we change the USA map to a disk with a known radius (for simplicity let the radius $r = 1$ and the majorization space be a square with side length of $2r = 2$). See Figure 1.5. From (1.2), the ratio of the areas can be expressed as

$$R = \frac{\pi r^2}{(2r)^2} = \frac{\pi}{4}. \tag{1.4}$$

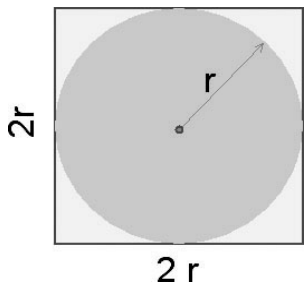


Figure 1.5: Pi by Simulation

From (1.4), we have the equality $\pi = 4R$ for the π simulation specified below:

(1) Generate two random numbers independently from $U(0, 1)$, the uniform distribution over range $[0, 1]$, representing a point (x, y) in an XY plane.
(2) Check if the point falls in the circle (i.e., if $(x-0.5)^2+(y-0.5)^2 < 0.25$).
(3) If the point falls in the circle, increase the count n by 1.
(4) Repeat steps (1)–(3) m times.
(5) Calculate the final ratio $R = \frac{n}{m}$ and the estimate $\pi = 4R$.

Let's write the steps using computer pseudocode or an algorithm:

Algorithm 1.1: Pi Simulation
Objective: return an estimate for the constant π
input number of simulation nRuns
$m := 0$
For $iRun := 1$ **To** nRuns
    Generate x from $U(0, 1)$
    Generate y from $U(0, 1)$
    $d := (x - 0.5)^2 + (y - 0.5)^2$

If $d < 0.25$ **Then** $m := m + 1$
Endfor
$Pi := 4\frac{m}{\text{nRuns}}$
Return Pi
§

There are other simulation algorithms for calculating π up to 2398 digits (e.g., Press, et al. 2007, p. 1194). The most famous example of π simulation is the Buffon Needle experiment.

1.2.3 *Definite Integrals*

Suppose we want to calculate the integral

$$I = \int_a^b f(x)dx, (c \geq f(x) \geq 0, b > a). \tag{1.5}$$

The interpretation of the integral is the area under curve $f(x)$ (Figure 1.6), which naturally links to the previous simulations. Indeed, we can develop a simulation algorithm similar to Algorithm 1.1 to calculate the integral. Note that the area of the majorization space is $c \times (b - a)$ in the current case.

Algorithm 1.2 Monte Carlo for a Definite Integral
Objective: return the numerical value of integral I defined by (1.5)
Input number of simulation nRuns
$m := 0$
For $iRun := 1$ **To** nRuns
generate x from $U(a, b)$
generate y from $U(0, c)$
If $y < f(x)$ **Then** $m := m + 1$
Endfor
$I := c(b - a)\frac{m}{\text{nRuns}}$
Return I
§

A better simulation approach is to calculate the mean value $\bar{f}$ of $f(x)$ over (a, b) and then obtain $I = (b - a)\bar{f}$. The mean $\bar{f}$ can be estimated using simulation as $\bar{f} \approx \frac{1}{m_{\max}} \sum_{i=1}^{m_{\max}} f(x_i)$, where x_i is a random number from the uniform distribution $U(a, b)$. We formalize this approach in Algorithm 1.3.

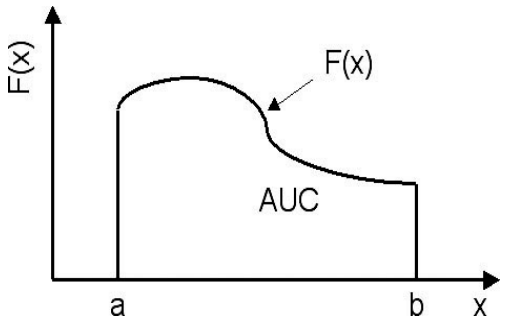


Figure 1.6: Area Under Curve (AUC)

Algorithm 1.3 Improved Monte Carlo for a Definite Integral
Objective: return the numerical value of integral I defined by (1.5)
input number of simulation nRuns
$sum := 0$
For $iRun := 1$ **To** nRuns
    generate x from $U(a, b)$
    $sum := sum + f(x)$
Endfor
$I := (b-a)\frac{sum}{\text{nRuns}}$
Return I
§

We can see that only one random number is needed for each simulation run using Algorithm 1.3, whereas two random numbers are required for each simulation run in Algorithm 1.2.

The validity of the algorithm is based on the law of large numbers in statistics, which ensures:

$$\frac{1}{b-a}\int_a^b f(x)\,dx \approx \bar{f}. \tag{1.6}$$

Theorem 1.1 ***Weak Law of Large Numbers:*** *Suppose that the expectation $E(X)$ of X is finite. Then the sample mean $\bar{X}_m$ converges in probability to $E(X)$; thus $lim_{m\to\infty}P\left(\left|\bar{X}_m - E(X)\right| > \varepsilon\right) = 0$ for every $\varepsilon > 0$, where $\bar{X}_m = \frac{1}{m}\sum_{i=1}^{m} X_i$.*

Theorem 1.2 ***Strong Law of Large Numbers:*** *Suppose that the expectation $E(X)$ of X is finite and then converges almost surely to $E(X)$. Thus $lim_{m\to\infty}P\left(\bar{X}_m = E(X)\right) = 1$.*

Theorem 1.3 ***Central Limit Theory****: For any* $\lambda_\alpha > 0$*,*

$$P\left(\left|\bar{X}_m - I\right| < \frac{\lambda_\alpha \sigma}{\sqrt{m}}\right) \approx \frac{2}{\sqrt{2\pi}} \int_0^{\lambda_\alpha} \exp\left(-\frac{t^2}{2}\right) dt = 1 - \alpha. \tag{1.7}$$

In other words,

$$\sqrt{m}\left(\hat{I}_m - I\right) \to N\left(0, \sigma^2\right), \text{ in distribution,}$$

where $\sigma^2 = var\left\{g\left(x\right)\right\}$*,* $g\left(\cdot\right) =$ real *function.*

Therefore the error of this Monte Carlo approximation for an integral is $O\left(\sqrt{m}\right)$ regardless of the dimension of the integral. This a major advantage of using simulation for a high-dimensional definite integral in comparison to numerical integration.

An interesting question is: how does the size of the majorization space affect the efficiency of the program? Will the simulation reach its maximum efficiency when the ratio of areas is close to 0.5? We leave the reader to find the answer (Exercise 1.2). In Chapter 2, we will discuss how to improve the efficiency of the sampling procedure.

1.2.4 *Fastest Route*

Industry and technology revolution do not always make life easier. Traffic jams often make one think: I am better off riding a bicycle. Using backward induction (see Chapter 5 for details) in decision theory, we can easily find the shortest route back home from the office. However, finding the fastest route is a quite different story, and much more challenging because there is random traffic involved. You may use Yahoo or Google Map to search for the shortcut, but whether this "shortcut" is real or not is dependent on how many other drivers utilize the same information. If every driver uses the same information and takes the same "shortcut," it will not be the fastest path at all. In real life, people often try different roads and record (by heart) the driving time taken each time. If they find a trip is faster than a previous trip on a different path, they may increase the probability of taking this road. By doing this, they hope to find the best path or strategy in the long run or on average. Can we prove this strategy works (and how well it works) using Monte Carlo? Many practical problems involve this type of competition and are well studied in game theory (see Chapter 4). In fact, this commuting problem is similar to the random-play-the-winner strategy in Casino and the response-adaptive randomization in clinical trial design (Chang, 2007b, 2008 and Chapter 9 of this book). But for now, let's

develop an algorithm (Algorithm 1.4) to find the fastest path before we leave home today.

Figure 1.7: Finding the Fastest Route

To fit the scope of this introductory chapter, we need to make some necessary assumptions to simplify the problem so that we have a concise algorithm. The assumptions in Algorithm 1.4 are: all paths have the same distance, driver k may (but not necessarily) switch his/her path with the probability of switching $= \frac{n_{ij}}{n_{i-1,j}+n_{ij}}$, where n_{ij} = number of cars on the j^{th} road on the i^{th} day. $U(M)$ is the probability mass function or random number generator that generates integers 1 to M with equal probability. For a typical day i, on a typical path j, we need to determine whether a typical car k will switch paths based on the probability of switching $\frac{n_{ij}}{n_{i-1,j}+n_{ij}}$. If yes, the path he/she is switching to is based on the probability $U(M)$. Note that even if a driver intends to switch, there is still a $1/M$ probability that he will turn out to be a nonswitcher. Therefore the actual switching probability is $\frac{M-1}{M}\frac{n_{ij}}{n_{i-1,j}+n_{ij}}$. When a switch does occur, the number of cars on the two paths involved in the switching need to be updated (reduced or increased by 1) for the next day. Here is the algorithm in computer pseudocode.

Algorithm 1.4: The Fastest Route
Objective: return $\{n_{ij}\}$ the number of cars on the j^{th} path on day i.
Input nRuns (days of drive), M (number of paths available).
Assign any initial n_{1j} cars on each path j on day 1
For $i := 1$ **To** nRuns
    For $j := 1$ **To** M
        For $k := 1$ **To** n_{ij}
            Generate x from $U(0,1)$

If $x \leq \frac{n_{ij}}{n_{i-1,j}+n_{ij}}$ **Then**
generate m from $U(M)$
$n_{i+1,m} := n_{im} + 1$
$n_{i+1,k} := n_{ik} - 1$
Endif
Endfor
Endfor
Endfor
Return $\{n_{ij}\}$
§

We always make decisions based on previous experiences when we are facing a set of options; therefore this kind of Monte Carlo can be used in many situations within and outside drug development (see Chapters 4 and 5). One can build very interesting scenarios to study how the drug industry and even the global economy works.

1.2.5 *Economic Globalization*

Globalization in its literal sense is the transformation of local or regional phenomena into global ones. It can be described as a process by which the people of the world are unified into a single society and function together. This process is a combination of economic, technological, sociocultural, and political forces. Globalization is often used to refer to economic globalization, that is, integration of national economies into the international economy through trade, foreign direct investment, capital flows, migration, and the spread of technology (Wikipedia, 2009). Advances in communication and transportation technology, combined with free-market ideology, have given goods, services, and capital unprecedented mobility.

Despite the complicated reasons behind individual cases, there is a very general statistical or mathematical model for such globalization processes, i.e., random (Brownian) motion at the microscopic level or the diffusion equation at the macroscopic level.

At the macroscopic level, diffusion can be modeled by the differential equation:

$$\frac{du}{dt} = -ku, \tag{1.8}$$

where u is the quantity of interest, e.g., electrical voltage or the level of technology or knowledge in different regions. The solution for (1.8) is the exponential function

$$u = u_0 e^{-kt},$$

where u_0 is determined by the initial condition.

Equation (1.8) states that when there is no external force, there is a general tendency that quantity u moves from an area with a higher value to another with a lower value. The speed of this homogenization process (diffusion) is characterized by a constant, k. In the globalization process, government policies of a country play a critical role in determining the value of k.

Diffusions are everywhere — we can say that if there is a difference, there will be a diffusion: water flow from higher to lower, technology diffusion from first world countries to third world countries, diffusion due to the difference in labor costs between the US and China. The difference in the percentage of English speakers in the US and China will also create diffusion; even overall cultural differences in two countries can lead to diffusion. These multi-channel diffusion processes from one region to another are not isolated or statistically dependent, but are governed by the following multi-dimension diffusion equation:

$$\frac{du_i}{dt} = -k_{ij} u_j, \tag{1.9}$$

where the diffusion coefficients k_{ij} are usually a function of u_{ij}, which means (1.9) is a nonlinear differential equation system. Such a system is virtually impossible to solve analytically, but can be easily solved using Monte Carlo simulation. Many commercial and noncommercial software tools (e.g., ExtendSim) are available for Monte Carlo simulations. Algorithm 1.7 later in this chapter can be used to find the steady state solution for this problem.

In Chapter 11 (Pharmacokinetic Simulation), we will discuss how microscopic Brownian motion turns out mathematically to be a diffusion equation at the macroscopic level and we present Monte Carlo algorithms for multi-channel diffusion of drug substances.

The diffusion model is a deterministic approach, which models the mean or overall behavior without variability considerations. In contrast, Monte Carlo simulations at the microscopic level provide both mean and variability evaluations for random phenomena.

1.2.6 *Percolation and Chaos*

In physics, chemistry, materials science, and geography, percolation concerns the movement of fluids through porous materials or random media.

Examples include the movement of solvents through filter paper and the seepage or movement of liquids (e.g., water or petroleum) through soil. Electrical analogs include the flow of electricity through random networks. Another interesting application is coffee percolation, where the solvent is water, the permeable substance is the coffee grounds, and the soluble constituents are the chemical compounds that give coffee its color, taste, and aroma.

Percolation is recognized as important in studying random media because when it happens the system will degenerate into chaos. A simple question to ask is: what is the average proportion of void-ratio (void volume to solid volume) that will cause percolation? Let's develop a Monte Carlo algorithm to answer the question.

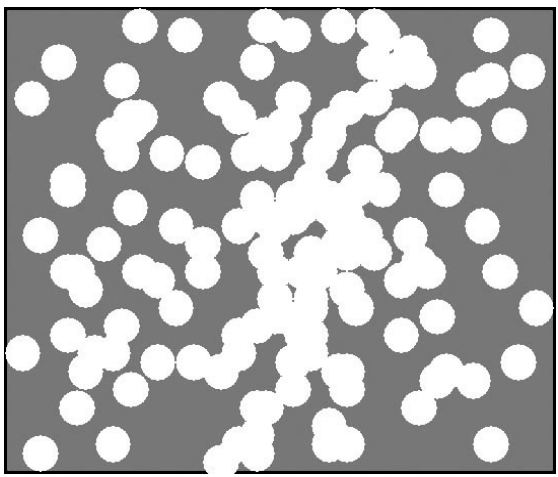


Figure 1.8: Simulation of Percolation

The basic idea of percolation simulation is, in a bounded space (e.g., a square, representing a piece of solid material), continually generate small disks (holes) at random locations in each other until the piece of material is percolated (Figure 1.8). The void ratio at percolation is then calculated. The occurrence of percolation can be either visually identified or computed in the simulation. To compute the percolation condition, a simple way is to use a "laser-beam" to scan the area, e.g., from left (top) to right (bottom), whenever a new disk is generated. Every time the laser beam moves right a very small unit it checks if it hits or intersects any disk; if no intersection, no percolation; if the laser beam always intersects some disk(s) during the scanning, it is percolated. Algorithm 1.5 is developed from this idea.

Algorithm 1.5: Simulation of Percolation
Objective: return average number of random holes at percolation.
Input: nRuns, small disk radius r
$nAve := 0$
For $iRun := 1$ **To** nRuns
    $i := 0$
    percolation := False
    While Not percolation:
        Generate x from $U(0,1)$
        Generate y from $U(0,1)$
        Draw a circle centered at (x, y) with radius r
        $i := i + 1$
        Check percolation
        If percolation **Then** $nAve := nAve + i/$nRuns
    Endwhile
Endfor
Return $nAve$
§

Note that we have ignored the disk overlapping in Algorithm 1.5 for simplicity. The line Check percolation is brief. The analytic solution is found to be 1/3 when the defects are completely random or uniformly distributed over the 3D space.

1.2.7 *Fish Pond*

If you want to measure the volume of water (V_w) in a pond, an easy way is to pour a cup (volume v) of a (colored) testing liquid into a pond and let it sufficiently mix (diffuse). Then take a cup of water from the pond and measure the concentration (c) of the test liquid. We now can calculate the volume of water in the pond: $V = \frac{v}{c} - v$ because the concentration is $c = v/(v + V)$.

Suppose we want to estimate the number of fish in a pond without pumping out the water (Figure 1.9). We can use a similar approach. Get a certain number (n_1) of fish (not from the pond), mark them, and put them into the pond. On the next day, we catch some (n_2) of the fish, and find r out of n_2 have been marked (r/n_2 equivalent to c). We now can estimate the total number of fish in the pond: $n = \frac{n_1 n_2}{r} - n_1$.

However, a commonly used method for solving this kind of problem is the so-called mark-recapture method (MRM). The simplest version of this method is the two-sample method: Capture some fish (n_1), mark them, and

put them back into the pond. On the next day you capture some fish (n_2). Of these, r were marked the day before. Then the estimated population size (N) is given by

$$N = \frac{n_1 n_2}{r}. \tag{1.10}$$

An approximately unbiased variance of N, or $var(N)$, can be estimated as:

$$var(N) = \frac{(n_1 + 1)(n_2 + 1)\,(n_1 - r)\,(n_2 - r)}{(r + 1)^2\,(r + 2)}. \tag{1.11}$$

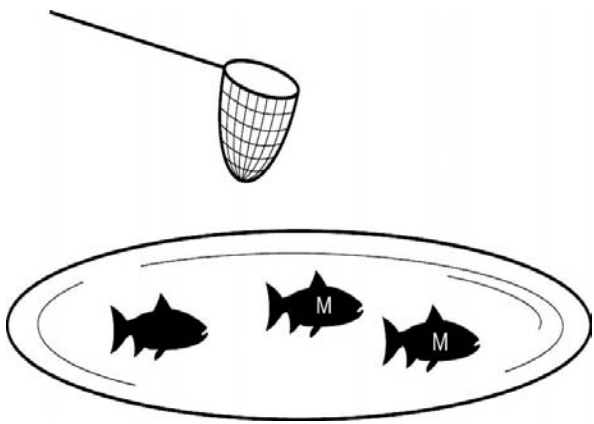


Figure 1.9: Capture-Recapture Method
(Source: www.figurethis.org)

This method is called the Lincoln–Petersen method. It can be used to estimate population size if (1) only two independent samples are taken, (2) each individual has an equal probability of being selected (or captured), and (3) the study population is closed, meaning no deaths or births and no migration between visits.

MRM is a method commonly used in ecology to estimate population size, population vital rates (i.e., survival, movement, and growth), and the number of people needing particular services (e.g., people infected with HIV). MRM can also be used to study the error rate in software source code, in clinical trials, databases, etc. This approach is useful when the total population size is unknown.

1.2.8 *Competing Risks*

Medical science and health technology have been improved dramatically in the past decades. The resulting health status and life-expectancy changes are particularly interesting to us. According to National Vital Statistics Reports (Vol. 56, No. 10, April 24, 2008), the five leading causes of death in the USA for 2005 were heart disease (652,091), cancer (559,312), stroke (143,579), chronic lower respiratory diseases (130,933), and accidents (unintentional injuries, 117,809). Among the fifteen leading causes of death, age-adjusted death rates decreased significantly from 2004 to 2005 for the top three leading causes, heart disease, cancer, and stroke, as long-term decreasing trends for these causes continued. Significant increases occurred for chronic lower respiratory diseases, unintentional injuries, Alzheimer's disease, influenza and pneumonia, hypertension, Parkinson's disease, and homicide. Despite the disease spectrum shifts, the life expectancy of the US population was 77.8 years, the same as that in 2004. What does it mean that life expectancies remained unchanged for the total population? Are our great efforts in medicine and health being balanced out by increasing negative causes (war, pollution, epidemiological disease)? Is it a phenomenon of simultaneous age-adjusted life expectancy? Is it because we allocate our health resources inefficiently? In what follows we will study how to allocate health resources appropriately and see the impact of inappropriate allocations.

We all face potential multiple causes of death. A cancer patient's response to cancer treatment effectively may fail due to other diseases such as stroke. To study the mechanism of life expectancy in disease complication is an interesting topic to governments for resource allocation. Algorithm 1.6 can be used to study the effect of single-cause life expectancy change on overall life expectancy ($aveLife$). This algorithm can be easily used for resource allocation purposes as described in the following: Suppose we just consider two causes of death, cancer and heart disease. Assume life expectancy will be T_1 without heart disease and T_2 without cancer. The cost for an increase of 1% in T_1 is C_1 (cancer treatment cost) and the cost for an increase of 1% in T_2 is C_2 (treatment cost for heart disease). How should we allocate resource C, where $C_1 + C_2 = C$, so that life expectancy is maximized? Readers should be able to solve this problem using Monte Carlo techniques after reading Chapter 2.

Algorithm 1.6: Life Expectancy under Competing Risks
Objective: average life expectancy under two potential causes of death.
Input: population size N, single cause life expectancies T_1 and T_2.

$aveLife := 0$
For $i := 1$ **To** N
    Generate t_1 from exponential distribution with hazard rate $1/T_1$.
    Generate t_2 from exponential distribution with hazard rate $1/T_2$.
    $L_i := \min(t_1, t_2)$
    $aveLife := aveLife + L_i/N$
Endfor
Return $aveLife$
§

1.2.9 *Pandemic Disease Modeling*

Endemic disease is a disease that exists permanently in a particular region or population. Malaria is a constant worry in parts of Africa. Epidemic disease refers to an outbreak of disease that attacks many people at about the same time and may spread through one or several communities. Pandemic disease occurs when an epidemic spreads throughout the world. SARS and swine flu (H1N1) are two examples of pandemic diseases.

To model a pandemic disease (e.g., swine flu), let's assume the rate of disease infection is proportional to the number of infected and the number of potentially infected, i.e.,

$$\frac{dn\,(t)}{dt} = kn\,(t)\,(N - n\,(t) - rt)\,, \tag{1.12}$$

where k is a constant to be determined, N is the overall population size, $n\,(t)$ is the infected population at time t, and the constant r is the spreading rate of H1N1 vaccine (number of patients who get the vaccine per unit time). In general, (1.12) can be solved using Monte Carlo (Exercise 1.10). However, if the term rt is neglected because no vaccine is available, we can solve (1.12) analytically as follows.

Equation (1.12) can be written as

$$\frac{1}{N - rt}\left(\frac{1}{n\,(t)} + \frac{1}{N - n\,(t)}\right) dn\,(t) = k\,dt. \tag{1.13}$$

Assume at time $t = 0$ there is only one person who carries the disease, i.e., $n\,(0) = 1$ and define the proportion of the infected population $p(t) = n\,(t)\,/N$. Then from (1.13), after integrating, we can obtain the logistic model.

$$p\,(t) = \frac{e^{kNt}}{1 + e^{kNt}}. \tag{1.14}$$

Suppose you now have the opportunity to have a swine flu vaccine that virtually guarantees you will not get the disease. However, there is a slim chance (probability p_0) of having a side effect that is as serious as swine flu. What should you do? You may want to get the vaccine shot right away if you can. But if you can't you may want to compare the probability p_0 and $p(t)$ at time t and decide to get the shot (if $p(t) > p_0$) or not (if $p(t) \leq p_0$).

Practically, k is unknown; thus (1.14) can be used to assess k, where $p(t) = n(t)/N$ is calculated based on the observed number of infections $n(t)$ at time t.

Considering that $p(t)$ is the overall measure of the infection rate at time t, we may, for our decision making, want to use the instantaneous probability within the unit time interval at time t, i.e., $\frac{dn(t)}{Ndt}$, or the conditional probability (given no disease at time t) of having the infection during the time interval t to $L_{\exp}$ (life expectancy):

$$P_c(t) = \int_t^{L_{\exp}} \left[\frac{dn(x)}{Ndx}\right] dx. \tag{1.15}$$

Substituting (1.12) and $p(t) = n(t)/N$ into (1.15), we have

$$P_c(t) = kN \int_t^{L_{\exp}} p(x)(1 - p(x))\, dx. \tag{1.16}$$

Further substituting (1.16) and carrying out the integral, we obtain the conditional probability of having the infection at time t and beyond:

$$P_c(t) = \frac{1}{1 + e^{kNt}} - \frac{1}{1 + e^{kNL_{\exp}}}. \tag{1.17}$$

Equation (1.17) can be compared with side-effect probability p_0 to help you decide whether you should get the flu vaccine or not. Clearly, when t gets larger or close to the life expectancy $L_{\exp}$, $P_c(t)$ gets smaller than p_0. In such a case, there is no point in having the flu shot.

However, this is a simple case; if you want to consider regional differences in k and other complications, then Monte Carlo simulation is a simple way to go.

1.2.10 *Random Walk and Integral Equation*

The Laplace partial differential equation has been used to describe many phenomena in physics. It is a basic law governing any potential field $u(x, y)$.

Mathematically, it is written as

$$\begin{cases} \Delta u(x,y) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \ (x,y) \in \Omega, \\ u|_S = g(S), \end{cases} \tag{1.18}$$

where Ω is a domain in 2D real space and S is the boundary of Ω. The function $g(\cdot)$ is given. The goal is to solve for the unknown function $u(Q) = u(x,y)$ numerically. There are several ways to find $u(Q)$, such as finite difference, finite element, or boundary element methods. But here we are going to use the Monte Carlo method to solve the problem. Monte Carlo is efficient here if we are only interested in the potential $u(Q)$ at a small set of points Q.

The mathematical basis for the Monte Carlo simulation is described as follows:

Draw a circle C centered at $Q \in \Omega$ with $C \subset \Omega$. Thus we have

$$\begin{aligned} \{S|Q\} &= \{S|Q\} \bigcup_{\varphi} \{Q \to C(\varphi)\} \\ &= \bigcup_{\varphi} \{\{S|Q\}\{Q \to C(\varphi)\}\} \\ &= \bigcup_{\varphi} \{\{S|C(\varphi)\}\{Q \to C(\varphi)\}\}, \end{aligned} \tag{1.19}$$

where $C(\varphi)$ is the point on the circle with the parameter angle φ and $\to$ implies the direction of the random walk.

Therefore,

$$P\{S|Q\} = \sum_{\varphi} P\{Q \to C(\varphi)\} P\{S|C(\varphi)\} \tag{1.20}$$

$$\begin{aligned} u(Q) &= \int_S g(S) P\{S|Q\} \\ &= \sum_{\varphi} P\{Q \to C(\varphi)\} \int_S g(S) P\{S|C(\varphi)\} \\ &= \frac{1}{2\pi} \int_0^{2\pi} u(C(\varphi)) d\varphi \\ &= \frac{1}{m} \sum_{i=1}^{m} u(C(\varphi_i)) = \tilde{u}_c \end{aligned} \tag{1.21}$$

where φ_i is randomly sampled from uniform distribution $U(0, 2\pi)$. If $\varphi_i \notin S$, we can use the average u from another circle C_2 centered at $C(\varphi_i) \cdots$ until the boundary is reached or close enough (Figure 1.10).

Based on the result (1.21), we now can construct a random walk algorithm to solve the Laplace problem (Muller, 1956):

Algorithm 1.7: Random Walk-Solving Integral Equation
Objective: return a numerical solution $u(Q)$ for the Laplace problem

(1) Draw the maximum circle C that is centered at Q without crossing the boundary S, i.e., $C \cap S = \phi$.
(2) Generate a random point $Q_1 = 2\pi\xi_1$, ξ_1 from $U(0,1)$; draw the maximum circle C_1 centered at Q_1 such that $C_1 \cap S = \phi$.
(3) Generate a random point $Q_2 = 2\pi\xi_2$, ξ_2 from $U(0,1)$. The process continues until the radius of the maximum circle is smaller than constant $\delta > 0$. Record $g(\Gamma_1)$, where Γ_1 is the closest point on the boundary to the final point of the random walk (in Figure 1.10, $\Gamma_1 = Q_3$).
(4) This finishes a simulation run. Repeat the random walk process n times. The solution at point Q is given by

$$u(Q) \approx \frac{1}{n} \sum_{i=1}^{n} g(\Gamma_i) \tag{1.22}$$

§

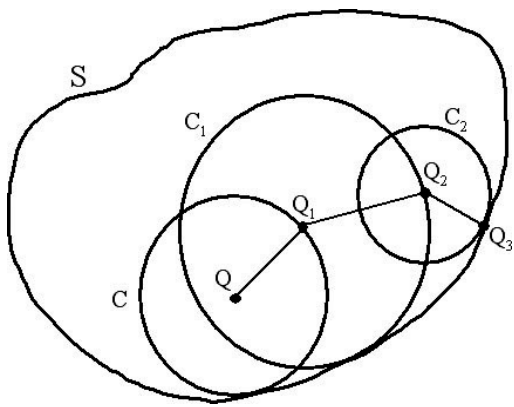


Figure 1.10: Monte Carlo for the Laplace Problem

For a three or higher dimensional Laplace problem, the Monte Carlo method is similar. There are many other random walk methods (RWM), e.g., RWM based finite difference.

1.2.11 *Financial Index and α Stable Distribution*

Market risks are the prospect of financial losses or gains due to unexpected changes in market prices and rates. Evaluating exposure to such risks is the primary concern of risk management in financial and nonfinancial institutions alike. Until the late 1980s market risks were estimated through gap and duration analysis (interest rates), portfolio theory (securities), sensitivity analysis (derivatives), or "what-if" scenarios. These traditional methods are only applicable to very specific assets and/or based on subjective reasoning (Weron, 2004).

Since the early 1990s a commonly used market risk estimation methodology has been the Value at Risk (VaR). VaR is the percentile defined by

$$\Pr\left(L > \text{VaR}\right) \leq 1 - c \tag{1.23}$$

where $L = -\Delta X\left(\tau\right)$ with $\Delta X\left(\tau\right)$ being the relative change (return) in portfolio value over the time horizon τ. Hence, large values of L correspond to large losses (or large negative returns).

The VaR provides a common consistent measure of risk across different positions and risk factors and takes into account the correlations or dependencies between different risk factors. Artzner et al. (1999) proposed a coherent measure, the Expected Shortfall (ES), also called Expected Tail Loss or Conditional VaR, as the expected value of the losses in excess of VaR:

$$ES = E\left(L|L > \text{VaR}\right). \tag{1.24}$$

The calculations were based on the normality assumption. However, it has long been known that asset returns are not normally distributed. Rather, empirical observations exhibit excess kurtosis (fat tails). The Dow Jones Industrial Average (DJIA) index is a prominent example, where the contrast with the Gaussian law is striking.

An appropriate model is the so-called Stable Distribution (α-stable). It is often argued that financial asset returns are the cumulative outcome of a vast number of pieces of information and individual decisions arriving almost continuously in time (McCulloch, 1996; Rachev and Mittnik, 2000). It seems that the Gaussian distribution should be fine, thanks to the Central Limit Theorem, which states that the sum of a large number of independent, identically distributed variables from a finite-variance distribution will tend to be normally distributed. However, financial asset returns usually have heavier tails. As indicated by Laha and Rohatgi (1979), there are at least two reasons to use α-stable: (1) it is supported by the generalized Central Limit Theorem, which states that stable laws are the only possible limit

distributions for properly normalized and centered sums of independent, identically distributed random variables, and (2) α-stable distributions are leptokurtic: since they can accommodate the fat tails and asymmetry, they fit empirical distributions much better.

An α-stable distribution requires four parameters for complete description: an index of stability $\alpha \in (0, 2]$, also called the tail index, tail exponent, or characteristic exponent, a skewness parameter $\beta \in [-1, 1]$, a scale parameter $\sigma > 0$, and a location parameter $\mu \in \mathbb{R}$. The Paretian-Lévy stable or α-stable distributions (Lévy, 1925) don't require a closed form. However, the most popular parameterization of the characteristic function of $X \sim S_\alpha(\sigma, \beta, \mu)$, i.e., an α-stable random variable with parameters α, σ, β, and μ, is given by

$$\ln \phi(t) = \begin{cases} -\sigma^\alpha |t|^\alpha \left(1 - i\beta sign(t) \tan \frac{\pi\alpha}{2}\right) + i\mu t, & \alpha \neq 1, \\ -\sigma|t| \left(1 + i\beta sign(t) \frac{2}{\pi} \ln(t)\right) + i\mu t, & \alpha = 1. \end{cases} \tag{1.25}$$

The following efficient algorithm for sampling from $S_\alpha(1, \beta, 0)$ was proposed by Chambers, Mallows and Stuck (1976).

Algorithm 1.8: α-Stable (Chambers, Mallows, and Stuck, 1976)

(1) Generate a random variable U uniformly distributed on $(-\frac{\pi}{2}, -\frac{\pi}{2})$ and an independent exponential random variable w with mean 1;
(2) If $\alpha \neq 1$, return

$$X = \left[1 + \left(\beta \tan \frac{\pi\alpha}{2}\right)^2\right]^{\frac{1}{2\alpha}} \frac{\sin(\alpha(U + \xi))}{(\cos u)^{1/\alpha}} \left[\frac{\cos(U - \alpha(U + \xi))}{w}\right]^{\frac{1-\alpha}{\alpha}}, \tag{1.26}$$

otherwise, return

$$X = \frac{2}{\pi} \left\{ \left(\frac{\pi}{2} + \beta U\right) \tan U - \beta \ln \left(\frac{\frac{\pi}{2} W \cos U}{\frac{\pi}{2} + \beta U} \right) \right\}. \tag{1.27}$$

§

For sampling from the general α-stable distribution $S_\alpha(\sigma, \beta, \mu)$, we can use the following property: if $X \sim S_\alpha(1, \beta, 0)$, then $Y \sim S_\alpha(\sigma, \beta, \mu)$ can be obtained by variable transform:

$$Y = \begin{cases} \sigma X + \mu, & \alpha \neq 1, \\ \sigma X + \frac{2}{\pi} \beta\sigma \ln \sigma + \mu, & \alpha = 1. \end{cases} \tag{1.28}$$

The random numbers generated from $S_\alpha(\sigma, \beta, \mu)$ can be used to study the strategies to maximize gain in stock trading.

1.2.12 *Nonlinear Equation System Solver*

Many problems lead to nonlinear equation systems, which are often impossible to solve analytically. Local linearization approaches are often used, but the methods are not always effective. A Monte Carlo method provides an alternative.

Suppose we have the following system of equations to be solved:

$$f_i\left(x_1, x_2, ..., x_n\right) = 0, i = 1, 2, ..., n, \tag{1.29}$$

where x_i are real and f_i are nonlinear functions.

To find a solution $\boldsymbol{x} = \{x_1, x_2, ..., x_n\}$ to the equation system (1.23), we define an objective function or loss function:

$$L\left(\boldsymbol{x}\right) = \sum_{i=1}^{n} f_i^2\left(\boldsymbol{x}\right). \tag{1.30}$$

The vector $\boldsymbol{x}$ will be considered as an approximate solution to (1.23) or (1.24) if it satisfies the following inequality:

$$L\left(\boldsymbol{x}\right) < \varepsilon, \tag{1.31}$$

where $\varepsilon > 0$ is a predefined small positive value. This is a typical classic optimization problem without constraints.

The classic unconstrained optimization problem can formally be presented as finding the set:

$$\Theta^* = \arg\min_{\theta \in \Theta} L\left(\boldsymbol{\theta}\right) = \left\{\boldsymbol{\theta}^* \in \Theta : L\left(\boldsymbol{\theta}^*\right) \leq L\left(\boldsymbol{\theta}\right) \text{ for all } \boldsymbol{\theta} \in \Theta\right\}, \tag{1.32}$$

where $L\left(\boldsymbol{\theta}\right)$ is called the loss function, θ is the p-dimensional vector of parameters that are being adjusted, and $\Theta \in \mathbb{R}^p$. Note that $\boldsymbol{\theta}^*$ may not be unique, i.e., $\boldsymbol{\Theta}^*$ may have more than one element.

For classic optimization (1.32), many algorithms can be used, e.g., Blind Random Search (Algorithm 1.9).

Algorithm 1.9: Blind Random Search
Objective: return an optimal value $\boldsymbol{\theta} \in \Theta^*$ in (1.32)
rejection := **True**
While rejection:
    Generate $\boldsymbol{\theta} \in \Theta$ based on a probability distribution.
    If $L\left(\boldsymbol{\theta}\right) < \varepsilon$ **Then** rejection := **False**
Endwhile
Return $\boldsymbol{\theta}$
§

The blind search algorithm is the simplest one, but not very efficient. The next method is to mimic the way a blind man climbs a mountain. He detects the heights nearby using a stick; if he finds a place higher than where he is standing, he steps up to that location. The process continues until he can't find a higher point. In this way, he hopes he can find the peak of the mountain.

Algorithm 1.10: Blind-Man Search for Optima
Objective: return optima

(1) Randomly select a starting point X_0, neighboring length L_0
(2) Randomly search m neighboring points $(X_{0i}, i = 1, ..., m)$,

If $\theta(X_{0i}) > \theta(X_0)$, then
select X_{0i} as a new starting point and go to Step 1.
If $\theta(X_{0i}) \leq \theta(X_0)$, then continue to search the neighboring points.
If $i = m$ and $\theta(X_{0i}) \leq \theta(X_0)$, $\forall i \in \{1, 2, ..., m\}$, then
change (randomly or not) L_0 and return to Step 1.

(3) If the search criteria have been met, then

return the final X_{0i} and $\theta(X_{0i})$.
§

This randomized search method allows for searching for global optima, where many deterministic approaches can only find the local optima.

1.2.13 *Stochastic Optimization*

In stochastic optimization, we are dealing with optimization with random noise $\varepsilon(\boldsymbol{\theta})$ in the objective function

$$y(\theta) = L(\boldsymbol{\theta}) + \varepsilon(\boldsymbol{\theta}). \tag{1.33}$$

The noise $\varepsilon(\boldsymbol{\theta})$ is a function of θ. Because of this noise, it fundamentally alters the search or optimization process as illustrated in Figure 1.11.

Stochastic approximation (SA) is a basis for stochastic optimization. Robbins and Monro (1951) introduced SA as a general root-finding method when measurements of the underlying function involve random noise.

If the objective function $L(\theta)$ is known and differentiable, then the optimization can be equivalent to solving the equation

$$\frac{\partial L(\theta)}{\partial \theta} = 0 \tag{1.34}$$

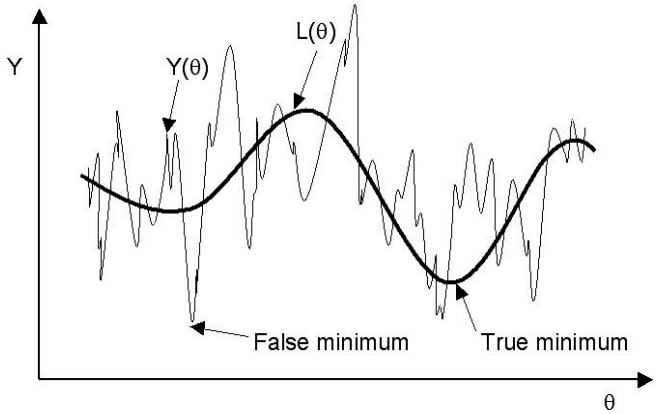


Figure 1.11: Classic versus Stochastic Optimization

for θ.

The gradient method is based on the formula to obtain the value $\hat{\theta}_{k+1}$ for the $(k+1)^{th}$ iteration from the value $\hat{\theta}_k$ at the k^{th} iteration:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k g\left(\hat{\theta}_k\right), \tag{1.35}$$

where constant $a_k > 0$ is the step size, and $g(\theta) = \frac{\partial L(\theta)}{\partial \theta}$.

However, because $g(\theta)$ is unknown in SA, we use an estimate $\hat{g}(\theta)$ to replace $g(\theta)$ in (1.35), which leads to

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}\left(\hat{\theta}_k\right), \tag{1.36}$$

where $\hat{g}$ is estimated using the so-called simultaneous perturbation (SP).

For two-sided SP gradient approximation, this leads to

$$\hat{g}_k\left(\hat{\theta}_k\right) = \begin{bmatrix} \frac{y\left(\hat{\theta}_k + c_k\Delta_k\right) - y\left(\hat{\theta}_k - c_k\Delta_k\right)}{2c_k\Delta_{k1}} \\ \vdots \\ \frac{y\left(\hat{\theta}_k + c_k\Delta_k\right) - y\left(\hat{\theta}_k - c_k\Delta_k\right)}{2c_k\Delta_{kp}} \end{bmatrix}. \tag{1.37}$$

Equation (1.37) provides an optimization algorithm, called SPSA. Because the numerator is the same in all p components of $\hat{g}_k\left(\hat{\theta}_k\right)$, the number of loss measurements needed to estimate the gradient in SPSA is two, regardless of the dimension p.

Applications of SPSA include queuing systems, pattern recognition, industrial quality improvement, aircraft design, simulation-based optimization, bioprocess control, neural network training, chemical process control,

fault detection, human-machine interaction, sensor placement and configuration, and vehicle traffic management.

The choice of the distribution for generating the Δ_k is important to the performance of the algorithm. One simple and popular distribution that satisfies the inverse moments condition is the symmetric Bernoulli ± 1 distribution. Two common mean-zero distributions that do not satisfy the inverse moments condition are symmetric uniform and normal with mean zero. The failure of both of these distributions is a consequence of the amount of probability mass near zero (Gentle and Härdle, 2004, p. 170–195)

1.2.14 *Symbolic Regression*

Genetic programming (GP), inspired by biological evolution, is an evolutionary computation (EC) technique that automatically solves problems without requiring the user to know or specify the form or structure of the solution in advance. At the most abstract level GP is a systematic, domain-independent method for getting computers to solve problems automatically starting from a high-level statement of what needs to be done (Oli, Langdon, and McPhee, 2008). The idea of genetic programming is to evolve a population of computer programs. Hopefully, generation by generation, GP stochastically transforms populations of programs into new populations of programs that will effectively solve the problem under consideration. Like evolution in nature, GP has been very successful at developing novel and unexpected ways of solving problems.

Representation: Syntax Tree

To study GP, it is convenient to express programs using syntax trees in GP rather than as lines of code. For example, the programs $(x + y) + 3$, $(y + 1) \times (x/2)$, and $(x/2) + 3$ can be represented by the three syntax trees in Figure 1.12, respectively. The variables and constants in the program (x, y, 1, 2, and 3) are leaves of the tree, called terminals, while the arithmetic operations ($+$, $\times$, and max) are internal nodes called functions. The sets of allowed functions and terminals together form the primitive set of a GP system.

Reproduction Mechanism

For the program to evaluate, GP must have the reproduction mechanism to generate new programs or offspring. There are two common ways to generate offspring: crossover and mutation. Crossover is the primary way (about 90% of new generations evolve by crossover, and 10% by mutation) to reduce the chance of chaos because the crossover leads to much similarity between parent and child. Crossover is a selection of a subtree for crossover, whereas mutation randomly generates a subtree to replace a randomly se-

lected subtree from a randomly selected individual. Crossover and mutation are illustrated in Figure 1.12, where the trees on the left are actually copies of the parents; their genetic material can freely be used without altering the original individuals.

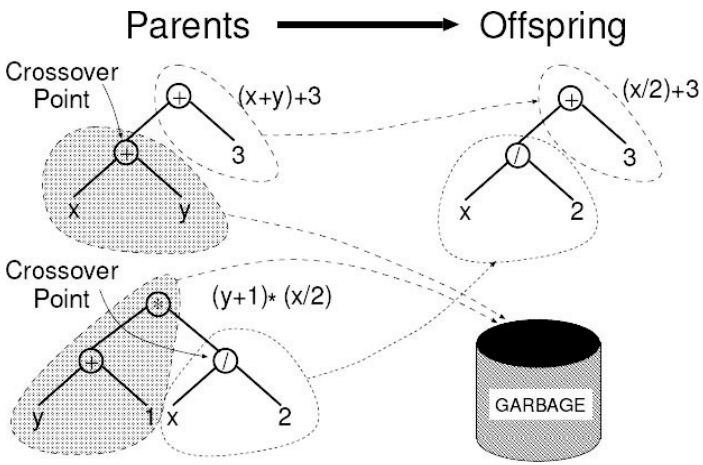


Figure 1.12: Crossover in GP
(Source: Oli et al., 2008)

Survival Fitness

The second mechanism required for program evolution is survival fitness. There are many possible ways to define the fitness measure. As an example, for the problem of finding a function $g(x)$ to approximate the target function $f(x)$, it is the mean square error between the two functions.

Algorithm 1.11: Genetic Programming
Objective: return best-fit-individual based GP.
Input generation size M, population size N, and target fitness V_t.
Create an initial population of programs and assess their fitness.
For $i := 1$ **To** M
  For $j := 1$ **To** N
    Determine crossover or mutation operation probabilistically.
    If crossover Then randomly select two trees and cross nodes.
    If mutation Then randomly select one tree and a node.
    Produce an offspring and evaluate its fitness.
  Endfor
  If fitness $\geq V_t$ **Then** Exitfor.

Endfor
Return the best-so-far individual
§

An algorithm for genetic programming is presented in Algorithm 1.11 and elaborated as follows.

M is the number of generations, N is the population size for each generation, and V_t is the target fitness. To create an initial population of programs, we use so-called primitives that include a terminal set (leaves) and a function set (nodes). The terminal set typically consists of variables and constants. The function set is driven by the nature of the problem's domain. In a simple numeric problem, for example, the function set may consist of merely the arithmetic functions $(+, -, \times, /)$, but other functions can also be used.

There are two iterative loops: one for generation, and the other for individuals within the generation. Each individual, whether generated by crossover or mutation, will have his fitness assessed. Individuals with lower fitness may be removed from the population. Individuals with higher fitness may have higher probabilities to be selected for generating their offspring for the next generation. The crossover usually operates within the same generation, but theoretically it can be performed between two generations. Fitness (function) can be measured in many ways, for example, in terms of (1) the amount of error between its output and the desired output, (2) the amount of time required to bring a system to a desired target state, (3) the accuracy of the program in recognizing patterns or classifying objects, or (4) the compliance of a structure with user-specified design criteria.

The usual termination criterion is that an individual's fitness should exceed a target value, but could instead be a problem-specific success predicate, or some other criterion. Typically, the single best-so-far individual is then harvested and designated as the result of the run.

Koza (Banzhaf et al., 1998) studied the symbolic regression

$$y = f(x) = \frac{x^2}{2},\ x \in [0, 1].$$

Using the terminal set: $x \in [-5, 5]$, function set: $+, -, \times$, and protected division %, and 500 individuals in each generation, Koza was able to obtain the best individual (function) f_i in generation i, where $f_0 = \frac{x}{3}$, $f_1 = \frac{x}{6-3x}$, $f_2 = \frac{x}{x(x-4)-1+4/x-(9(x+1)/(5x)+x)/(6-3x)}$, and $f_3 = \frac{x^2}{2}$. Therefore, at generation 3, the correct solution is found. However, as the generation number increases, the best fit function starts to expand again. We will return to this topic in Chapter 13.

1.3 Simulations in Drug Development

1.3.1 *Challenges in the Pharmaceutical Industry*

It is of great concern that the pharmaceutical industry may be undergoing a productivity crisis caused in part by the pragmatic definition of the number of new drugs or new molecular entities (NMEs) approved each year. The number of NMEs and priority review drug approvals have remained relatively flat in the past decades (Figure 1.13). However, the amount of spending in Research and Development has consistently increased yearly from approximately 4B in 1976 to 36B in 2004 based on a 9% inflation adjusted rate for 2002.

Moreover, from 1990 to 1994, 11 new drugs had reached the "top 100 drugs" category in terms of global sales. From 1995 to 1999, ten new approved drugs made it into the "top 100 drugs" category. However, during the period from 2000–2004, only two new approvals broke into the group of top 100 revenue generators. During 2005 to 2007, the FDA approved only half the number of new compounds as it had only a decade before. And fewer than 10% of these newly approved compounds are expected to ultimately generate sales of even $350 million annually (Simon and Pecker, 2003). Five different blockbuster drugs went off-patent in 2006 and more such transitions loom large on the horizon. Price pressures from the public and private sectors have made headlines nationwide. These situations have made the pharmaceutical industry as a whole seem vulnerable in the face of new challenges, new realities regarding drug development, new competition from biotechnology and the emerging world of genomics, and new expectations on the part of consumers and managed care providers (Paich et al., 2009). There are many reasons why this is happening (Woodcock, 2004, Chang, 2007b). Among them is insufficient technology innovation, such as adaptive design and computer simulation.

Traditional drug development is subjective to a large extent, and intuitive decision-making processes are primarily based on individual experiences. Therefore, optimal design is often not achieved. Monte Carlo (MC) is a powerful evaluation tool for development plans in all stages and study designs to support strategic decision making. MC is intuitive and easy to implement with minimal cost and can be done in a short time. The utilities of MC include, but are not limited to (1) sensitivity analysis and risk assessment, (2) estimation of probability of success (power), (3) design evaluation and optimization, (4) cost, time, and risk reduction, (5) clinical development program evaluation and prioritization, (6) trial monitoring and interim prediction of future outcomes, (7) prediction of long-term benefits using short-

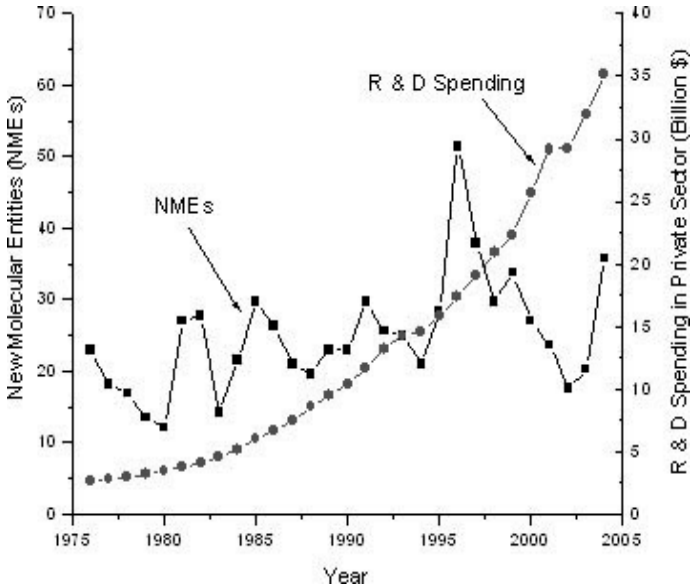


Figure 1.13: Research and Development Spending and NMEs approved by the FDA (Data source: Pammolli and Riccaboni, 2007)

term outcomes, (8) validation of trial design and statistical methods, and (9) streamlining communication among different parties. Within regulatory bodies, MC can be and has been used for assessing the robustness of results, validating statistical methodology, and predicting long-term benefits in accelerated approvals. Monte Carlo is often used for power simulation in hypothesis tests and for selection of the best statistical test method among several alternatives, but these are just mostly basic applications of Monte Carlo. Simulation should go well beyond this limited scope, as outlined in this book.

1.3.2 *Classification of Simulations in Drug Development*

Based on the scope of studies, Monte Carlo simulations can be classified into meta-simulation, macro-simulation, and micro-simulation (Figure 1.14). Meta-simulations target multiple sectors or drug companies. Because of the nature of competition and collaboration, Monte Carlo can combine with game and decision theory to solve many problems. The examples of interest are impact analysis of a technology platform, drug development globalization, and drug industry partnerships. Macro-simulations deal with problems involving a single business entity or company. Thus, decision the-