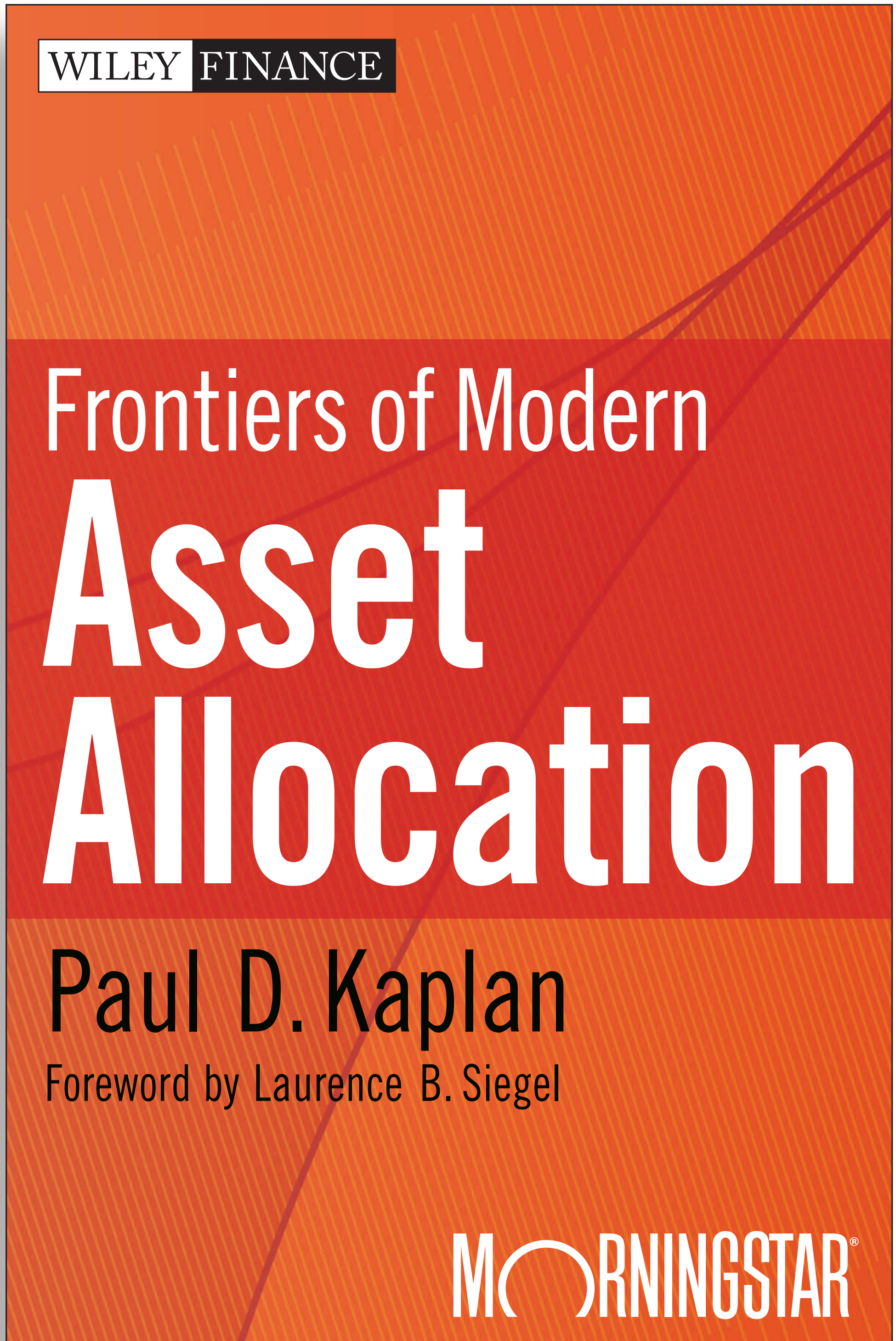
An Excerpt from

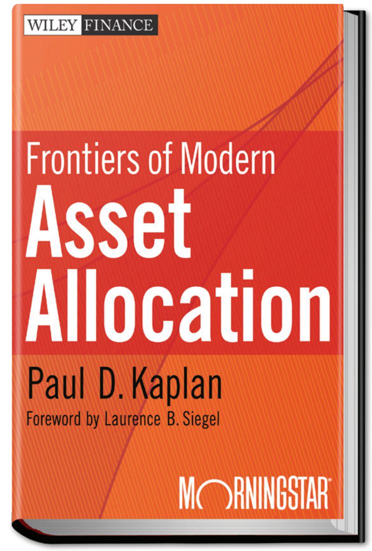


Available at www.wiley.com/buy/9781118115060 and wherever books and eBooks are sold.
Discounts ranging from 50% to 70% are available for bulk purchases. Please contact Laura Walsh at lwalsh@wiley.com or (201) 748-8660 to learn more.

WILEY
Now you know.
wiley.com

Innovative approaches to putting asset allocation into practice

Building on more than 15 years of asset-allocation research, Paul D. Kaplan, who led the development of the methodologies behind the Morningstar Rating™ and the Morningstar Style Box™, tackles key challenges investor professionals face when putting asset-allocation theory into practice.



978-1-118-11506-0 • Cloth
US $95.00 • CAN $114.00 • £65.00

Table of Contents

Available at www.wiley.com/buy/9781118115060 and wherever books and eBooks are sold.

CHAPTER 25

Updating Monte Carlo Simulation for the Twenty-First Century*

Paul D. Kaplan and Sam Savage

In 1946, Stanislaw Ulam, a Polish-born mathematician and member of the Manhattan atomic bomb project, was whiling away the time while recovering from an illness by playing solitaire, and began to wonder about the likelihood of success. So he stopped playing with the cards and returned to his profession of mathematics by trying to calculate the percentage of successful games out of all possible shuffles. This turned out to be harder than he thought. So he came up with an alternative method using the power of an early computer to simulate 100 card shuffles and then simply count the number of winning hands.

Thus was born a computational technique now known as Monte Carlo simulation because the basic building block was none other than a computerized version of a roulette wheel with many billions of numbers around the edge. Although it took decades to work out all the kinks in the computerized roulette wheel, Monte Carlo simulation has become a standard tool of risk management. And its latest incarnations offer several bold advances.

**Morningstar Alternative Investments Observer*, Fourth Quarter 2010. A longer version appears as "Monte Carlo: Shining a Light on Uncertainty," *Investments and Wealth Monitor*, March/April 2011. The appendix was added for this book.

AN EARLY APPLICATION OF MONTE CARLO SIMULATION TO ASSET ALLOCATION

In 1976, Roger Ibbotson, then an assistant professor at the University of Chicago, and Rex Sinquefield published a paper called "Stocks, Bonds, Bills, and Inflation: Simulations of the Future (1976–2000)" as a companion piece to their historical study of asset class returns.[1] They used the Monte Carlo method developed by Ulam to make probabilistic predictions of the form "there is an *X* percent chance that \$1 invested in the portfolio will grow to \$*Y* or more in *Z* years." Putting together past history with the forecasts, they generated tulip or fan charts similar to Figure 25.1.

Like Harry Markowitz's 1952 mean-variance model, the Ibbotson-Sinquefield simulation model was an early attempt to cure what Sam Savage (2009) has dubbed the flaw of averages. In general, the flaw of averages is a set of systematic errors that occurs when people use single numbers (usually averages) to describe uncertain future quantities. For example, if you plan to rob a bank of \$10 million and have one chance in 100 of getting away with it, your average take is \$100,000. If you described your activity beforehand as "making \$100,000," you would be correct, on average. But this is a terrible characterization of a bank heist! Yet, this very mistake is made all the time in business practice. It helps explain why everything is behind schedule, beyond budget, and below projections, and it was an accessory to the economic catastrophe that culminated in 2008.

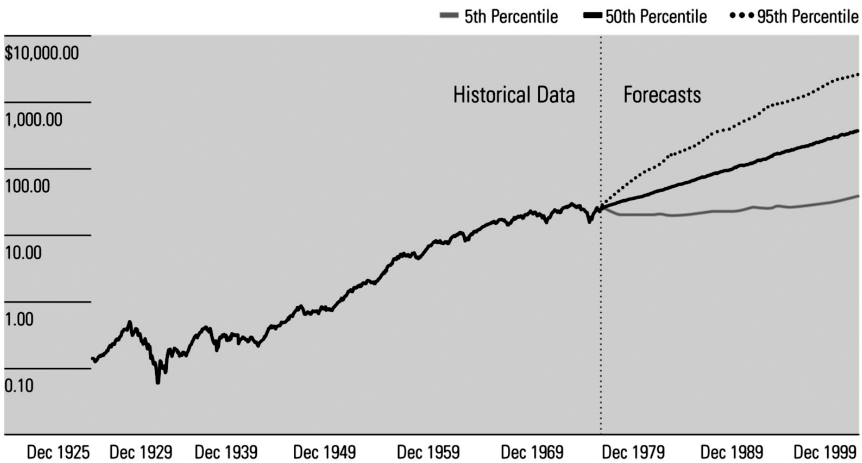


FIGURE 25.1 Ibbotson-Sinquefield Simulation Chart
Source: Morningstar (2010).

Ibbotson and Sinquefield simulated each future month's return on a portfolio from historical monthly returns over the period 1926 to 1974, a period of 588 months. Like Ulam, Ibbotson and Sinquefield used a computer program to spin a roulette wheel with 588 spots 300 times for each simulated future. By running only a few thousand possible futures, they were able to complete the calculations on a mainframe computer of the era in time for publication.

IBBOTSON AND SINQUEFIELD WITHOUT MONTE CARLO SIMULATION

While there was interest in the Ibbotson-Sinquefield simulation model at the time of its publication, technology for running Monte Carlo simulations was not readily available to many in the investment community. But four years later, four researchers (Lewis, Kassouf, Brehm, and Johnston 1980) published a paper that showed that to a large degree the results of the Ibbotson-Sinquefield simulations could be replicated without Monte Carlo simulation. This paper showed that by making a number of simplifying assumptions and applying the Central Limit Theorem,[2] probabilistic forecasts of cumulative wealth can be made using mathematical formulas.[3] The Lewis et al. model became the standard method for probabilistic forecasting and is in wide use today.

However, as powerful as the Lewis et al. model is, it is not up to the task of forecasting problems other than simple wealth accumulation with no inflows or outflows. Consider the problem of forecasting how long a retiree can make a given amount of wealth last before going broke, assuming that he invests unspent wealth in a portfolio of risky assets. If we were to assume a fixed rate of return on investments during retirement and solve for the year in which the retiree runs out of money, we would run afoul of the flaw of averages because there are many plausible scenarios in which poor returns in the early years cause the retiree to go broke well before the time forecasted. Except under highly simplified assumptions, the only practical way to approach this problem is Monte Carlo simulation.[4] Hence, the Monte Carlo approach has become the most common method for modeling drawing down wealth during retirement.

Furthermore, the capital markets do not always behave in the way that the simplified models assume. History is replete with fat tail events that are not captured by models based on the bell curve (as all of the simplified models are).[5] This is another reason why Monte Carlo simulation is usually the most practical approach to investment forecasting.

This is not to say that Monte Carlo simulation is a silver bullet. There are a number of practical issues when implementing a Monte Carlo model that must be taken into consideration. Michele Gambera (2002) summarizes a number of these issues, namely:

1. The accuracy of the results is limited by the number of simulated histories. Hence, there is a trade-off between the accuracy of the model and the time it takes to run it.
2. The amount of time needed to run enough simulated histories might be too long to be practical to obtain enough accuracy to make the model useful.
3. The amount of computer storage needed to run a model might be impractically large. For example, to store 1,000 simulated histories over a 25-year period of monthly returns requires storing 300,000 numbers per asset class.

A TWENTY-FIRST CENTURY UPDATE

Fortunately, twenty-first century technology addresses these issues and not only makes Monte Carlo simulation practical, but also interactive and highly flexible. This is due to three computer technologies that have recently come together: interactive simulation, the Distribution String, and cloud computing.

1. **Interactive Simulation**. The central processing unit in today's iPhone is hundreds of times more powerful than the machine used by Ibbotson and Sinquefield, and many times faster than that used in 2002, the date of Gambera's publication. Furthermore, several recent software breakthroughs have focused specifically on the speed of Monte Carlo simulation. Risk Solver Platform, for example, from Nevada-based Frontline Systems, can simulate 100,000 spins of the roulette wheel in Microsoft Excel before the user's finger has left the "Enter" key of their computer. The resulting "interactive" simulation provides a new level of intuition into uncertainty. And more speed is on the way. Not only are CPUs getting faster, but machines are being fitted with parallel processors. Many applications cannot be programmed to take advantage of multiple processors. Monte Carlo simulation is a notable exception, and is known in the trade as "embarrassingly parallel." It may not be long before specialized machines are developed for the sole purpose of running simulations.
2. **The Distribution String**. The Distribution String is a new standard for packaging thousands of Monte Carlo scenarios into a single data

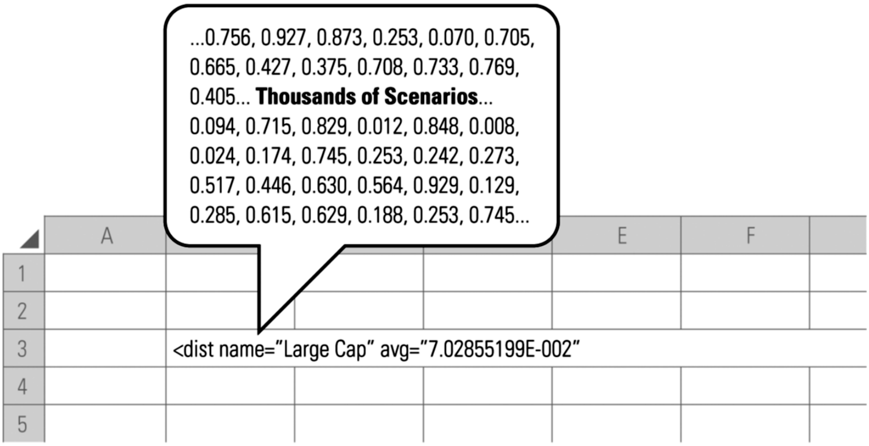


FIGURE 25.2 The Distribution String or DIST
Source: VectorEconomics.com.

element (Figure 25.2). Known as DIST, it was developed by Savage, in collaboration with Oracle Corp., SAS Institute, and Frontline Systems, and along with interactive simulation, addresses the issues raised by Gambera. If interactive simulation is a new lightbulb for illuminating uncertainty, then the Distribution String is the AC current that lights the bulb. The 300,000 data elements required to store a 25-year simulation is reduced to 300 DIST elements. And when people say that size does not matter, this does not apply to factors of 1,000.

3. **Cloud Computing**. The DIST standard is so compact that thousands of Monte Carlo trials may be downloaded over the web in seconds. This provides a collaborative network in which specialists in financial statistics can produce probability distributions for immediate consumption by a wide array of investors, worldwide. Hence, it may unleash an industry in DIST distribution.

IMPLICATIONS FOR TOMORROW

These recent technological advances in Monte Carlo simulation allow for a probability power grid that can drive asset allocation, retirement models, and valuations on everything from laptops to BlackBerries and iPads.

Furthermore, Monte Carlo models built with DISTs are also highly flexible, allowing for almost any type of return distribution or underlying probability model. Today, in light of the recent global financial crisis, there is much debate about how to best model the probability distributions of asset-class returns. Some researchers are proposing that we replace models based on the bell curve or normal distribution (which are tractable from a theoretical perspective) with fat-tail models in which extreme events occur (which require simulation to analyze).[6] Others argue that the models based on the normal distribution are adequate. Distribution Strings are agnostic regarding this debate.

Similarly, there is debate about the usefulness of correlation matrixes to represent the co-movements of asset-class returns, with many arguing that during down markets, asset classes become more correlated. Again, the DIST approach allows any pattern of co-movements to be modeled. As an extreme, the scatter plot in Figure 25.3 of asset classes HAP and PY is a happy face, which is certainly a type of relationship. Although the correlation coefficient between HAP and PY is almost zero, compressing the underlying data into a pair of DISTs preserves the relationship in its entirety.

FIGURE 25.3 "Uncorrelated" Asset Returns
Source: VectorEconomics.com.

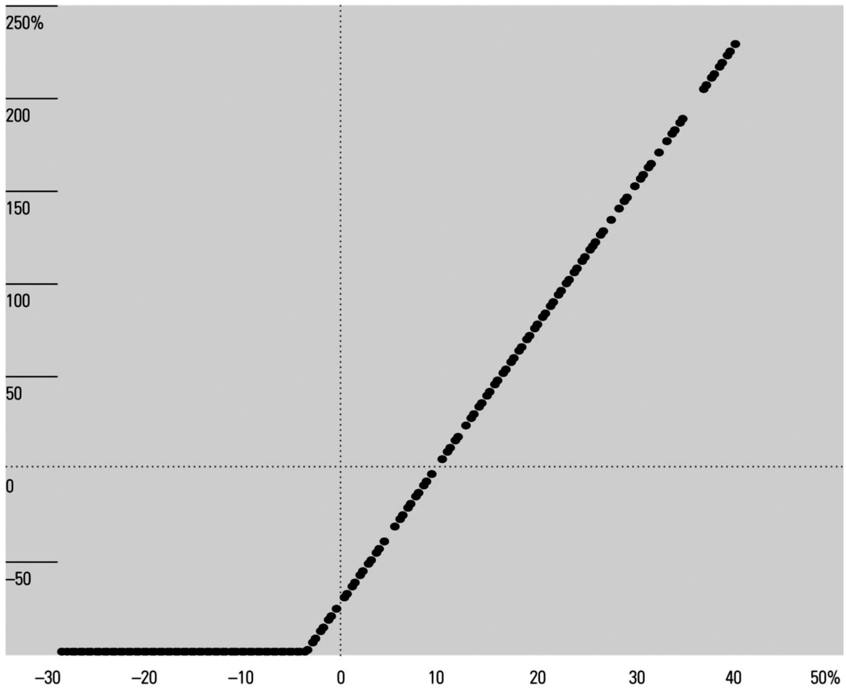


FIGURE 25.4 Asset Return versus Call Option Return

The ability to model nonlinear relationships between return distributions has important real-life applications. Consider Figure 25.4, which is a scatter plot of returns on a stock index and a call option on the index. The DIST approach allows us to preserve the exact hockey stick relationship between these returns of these two assets that cannot be captured by a correlation coefficient. This is important if options are being considered as part of the portfolio.

These examples illustrate the importance of preserving underlying relationships between assets when creating a Monte Carlo model out of DISTs. Sets of DISTs that preserve such relationships are said to be coherent. The creation of coherent DIST libraries is one of the most important functions of probability management, a field devoted to managing databases not of numbers but probability distributions (see www.ProbabilityManagement.org).

WHERE TO FIND IT

The power of DIST technology is beginning to appear in several programming tools for the computer savvy investment professional. It is currently supported by three software packages. The first two are spreadsheet add-ins: Risk Solver (http://Solver.com) and XLSim (http://VectorEconomics.com). The last is a multidimensional modeling tool: Analytica (www.Lumina.com). For those who want ready-to-use interactive asset allocation software with Monte Carlo models, Morningstar is in the process of creating new tools based on the DIST technology. In the near future, it will be possible to include many types of distributions, including those that model the occasional financial crisis, in an interactive environment on the desktop or laptop.

APPENDIX 25A: TECHNICAL DETAILS ON DISTRIBUTION STRINGS

A Distribution String, or DIST, is an XML string that represents a vector of numbers in a compressed format. Compression is achieved by creating a limited number of bins between the minimum and maximum values of the original data and recording the bin number of each entry rather than the entry itself.

There are several DIST formats, but for the purpose of simulating returns, the DOUBLE format, which maps each entry into one of 2^{16} or 65,536 bins, is the most appropriate. Hence, each entry is mapped into a 16-element bit string. Each group of three consecutive 16-bit strings is concatenated together into a 48-bit string, which is then redivided into eight consecutive 6-bit strings. Each resulting 6-bit string is then mapped into one of 64 characters as shown in Table 25A.1. The result is a compression ratio of vector entries to characters of 3 to 8.

Compressing a vector of double precision numbers into a DIST results in a certain loss of accuracy, but the magnitude of the errors is almost always small enough to ignore in return data. However, the DIST does include the values of the minimum, maximum, and average values of the original data so that these are always accurately recovered when a DIST is decompressed.

The DIST standard requires providing two routines to create a DIST from data and decompress a DIST, called *DstCreate* and *Dst* respectively. *DstCreate* has three optional parameters:

1. *Name*. A string to name the DIST. If not provided, defaults to Unnamed—.
2. *Type*. A string to indicate the type of compression. Defaults to Double.
3. *Origin*. A string that describes the origin of the data. Defaults to the null string.

I have written MatLab versions of *DstCreate* and *Dst* to illustrate how to create DISTs and decompress them in practice. Figures 25A.1 and 25A.2

TABLE 25A.1 Mapping Between 6-Bit Strings and DIST Characters

6-Bit String	Decimal Value	DIST Character	6-Bit String	Decimal Value	DIST Character
000000	0	A	100000	32	g
000001	1	B	100001	33	h
000010	2	C	100010	34	i
000011	3	D	100011	35	j
000100	4	E	100100	36	k
000101	5	F	100101	37	l
000110	6	G	100110	38	m
000111	7	H	100111	39	n
001000	8	I	101000	40	o
001001	9	J	101001	41	p
001010	10	K	101010	42	q
001011	11	L	101011	43	r
001100	12	M	101100	44	s
001101	13	N	101101	45	t
001110	14	O	101110	46	u
001111	15	P	101111	47	v
010000	16	Q	110000	48	w
010001	17	R	110001	49	x
010010	18	S	110010	50	y
010011	19	T	110011	51	z
010100	20	U	110100	52	0
010101	21	V	110101	53	1
010110	22	W	110110	54	2
010111	23	X	110111	55	3
011000	24	Y	111000	56	4
011001	25	Z	111001	57	5
011010	26	a	111010	58	6
011011	27	b	111011	59	7
011100	28	c	111100	60	8
011101	29	d	111101	61	9
011110	30	e	111110	62	+
011111	31	f	111111	63	/

Source: VectorEconomics.com.

```
function dist = DstCreate(sip,dname,dtype,dorigin)
% DSTCREATE. Creates a DIST Version 1.1
% sip = column vector to convert to a DIST.
% dname = A string that gives the name of the DIST
% dtype = Case independent string indicating compression type
% dorigin = string documenting origin of data
% dist  = The resulting DIST
% dname, dtype, and dorigin should be optional parameters, but I made
% them mandatory for this illustration.
% Set basic data
Code_Values = ...
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/';
n = length(sip);
switch lower(dtype)
   case 'binary'
      data = sip~=0;
   otherwise
      data = sip;
end
minsip = min(data);
maxsip = max(data);
avgsip = mean(data);
switch lower(dtype)
    case 'double' % Only case shown
       % Assign each trial to a bin from 0 to 65535.
       bins = fix(0.5+65535*(sip-minsip)/(maxsip-minsip));
       % If n is not divisible by 3, pad the bin vector with zeros.
       bins = [bins; zeros(mod(3-mod(n,3),3),1)];
       % Rearrange bins into a 3 column matrix with each row being 3
       % consecutive trials. I call each row a trio.
```

FIGURE 25A.1 *DstCreate* in MatLab

```
        ntrios = length(bins)/3;
        trios = reshape(bins,3,ntrios)';
        % Convert each bin number into a 16-bit binary string and
        % concatenate the results from each trio to form a 48-bit
        % string.
         btrios = [dec2bin(trios(:,1),16) dec2bin(trios(:,2),16) ...
                   dec2bin(trios(:,3),16)];
        % Split each 48-bit string into 8 6-bit strings and convert
        % each resulting 6-bit string into a number between 0 and 63.
        idx8s = [];
        for j = 1:8
            idx8s = [idx8s bin2dec(btrios(:,(6*j-5):(6*j)))];
        end
        % Map each number between 0 and 63 into its corresponding
        % character in Code_Values, to form the final string
        % representation of the vector of trials values.
        dstring = Code_Values(reshape(idx8s',8*ntrios,1)'+1);
end
% Form the DIST XML string.
dist = sprintf(['<dist name="%s" avg="%16.8E" min="%16.8E" ' ...
                 'max="%16.8E" count="%.0f" ' ...
                  'type="%s" origin="%s" ' ...
                  'ver="1.1">%s</dist>'] ...,
                dname,avgsip,minsip,maxsip,n, ...
               dtype,dorigin,dstring);

end
```

FIGURE 25A.1 (Continued)

```
function sip = Dst(dist)
% DST. Convert a DIST (version 1.1) into a data vector, sip
Code_Values = ...
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/';
% Extract the substring that contains the fields.
minsip = code not shown; % value of min field
maxsip = code not shown; % value of max field
avgsip = code not shown; % value of avg field
n = code not shown;      % value of count field
dtype = code not shown;  % value of type field
% Extract the substring of dist that contains the compressed data.
dstring = code not shown;
% Map each character in dstring into a 6-bit binary string.
m = length(dstring);
bin6Mat = repmat('0',m,6);
for charNo = 1:64
    locs = dstring==Code_Values(charNo);
    bin6Mat(locs,:) = repmat(dec2bin(charNo-1,6), ...
                             length(find(locs)),1);
end
% Place the 6-bit strings end-to-end to form a binary string,
binString = reshape(bin6Mat',1,6*m);
% Convert binString to sip based on dtype.
switch lower(dtype)
    case 'double' % Only case shown
       % Divide binString into 16-bit binary strings.
```

FIGURE 25A.2 *Dst* in MatLab

```
        bin16Mat = reshape(binString,16,length(binString)/16)';
        % Convert the binary strings into n bin numbers.
        bins = bin2dec(bin16Mat(1:n,:));
        % Map bin numbers into data (Version 1.1)
        decoded = minsip+bins*(maxsip-minsip)/65535;
        end
end
% Make sure that the mins and maxs are correct.
decoded(bins==min(bins)) = minsip;
decoded(bins==max(bins)) = maxsip;
% We need to adjust the decoded numbers that are neither mins or maxs
% so that the average of the resulting vector equals the average
% specified in the DIST.
fixlocs = (decoded~=minsip) & (decoded~=maxsip);
sip = decoded;
sip(fixlocs) = decoded(fixlocs)—...
               sum(decoded-avgsip)/length(find(fixlocs));
end
```

FIGURE 25A.2 (Continued)

provide excerpts from these routines to show how to compress and decompress data using DISTs.

NOTES

1. I.e., Ibbotson and Sinquefield (1976b) as the companion piece to Ibbotson and Sinquefield (1976a).
2. The Central Limit Theorem states that when a large number of statistically independent variables with the same distribution are added, the result is a normal (bell-shaped) distribution, regardless of the distribution of the underlying variables. An exception to the theorem occurs with the sort of fat-tail distributions discussed in Kaplan (2009) that appears as Chapter 18 in this book.

3. For a description of the model and the formulas, see pages 113 to 118 in Morningstar (2010).
4. Milevsky and Robinson (2005) present a probabilistic formula for sustainable spending rate that they derive under a set of assumptions, one of which is that the spending rate is constant. However, for other spending patterns, such as including occasional lump-sum amounts to finance, say, a child's wedding, a grandchild's education, or a vacation home for oneself, in addition to regular spending, Monte Carlo simulation remains the only practical option.
5. See Part III of this book.
6. See Chapters 18 and 19 of this book.

REFERENCES

Gambera, Michele. 2002. "It's a Long Way to Monte Carlo," *Business Economics* (July).

Ibbotson, Roger G., and Rex A. Sinquefield. 1976a. "Stocks, Bonds, Bills, and Inflation: Year-by-Year Historical Returns (1926–1978)." *Journal of Business* 49: 11–47.

Ibbotson, Roger G., and Rex A. Sinquefield. 1976b. "Stocks, Bonds, Bills, and Inflation: Simulations of the Future (1976–2000)." *Journal of Business* 49: 313–338.

Kaplan, Paul D. 2009. "Déjà Vu All Over Again." *Morningstar Advisor* (February/March). [Chapter 18].

Lewis, Alan L., Sheen T. Kassouf, R. Dennis Brehm, and Jack Johnston. 1980. "The Ibbotson-Sinquefield Simulation Made Easy." *Journal of Business* 53: 205–214.

Milevsky, Moshe A., and Chris Robinson. 2005. "A Sustainable Spending Rate without Simulation." *Financial Analysts Journal* 61: 89–100.

Morningstar, Inc. 2010. *Ibbotson Stocks, Bonds, Bills, and Inflation (SBBI) Classic Yearbook*. Morningstar, Inc.

Savage, Sam. 2009. *The Flaw of Averages*. Hoboken, NJ: John Wiley & Sons.