什麼是「蒙地卡羅方法」？它是一種數值方法，利用亂數取樣 (random sampling) 模擬來解決數學問題。一般公認蒙地卡羅方法一詞為著名數學家 John von Neumann 等人於 1949 年一篇名為「The Monte Carlo method」所提出。其實，此方法的理論基礎於更早時候已為人所知，只不過用手動產生亂數來解決問題，是一件費時又費力的繁瑣工作，必須等到電腦時代，使此繁複計算工作才變得實際可執行

為什麼取名「蒙地卡羅方法」?

在數學上，所謂產生亂數，就是從一給定的數集合中選出的數，若從集合中不按順序隨機選取其中數字，這就叫是亂數，若是被選到的機率相同，這就叫是一均勻亂數。例如擲骰子，1 點至 6 點骰子出現機率均等。雖然，現在科學研究中經常是利用電腦產生均勻分佈於 [0, 1] 之間的數，但早期最簡單方式是由賭場輪盤以機械方式產生亂數。這也是為何以摩洛歌首都--蒙地卡羅(賭城)為名的緣故

[例題：求圓周率 π]

利用電腦產生均勻分佈於 [0, 1] 之間的數為點 X、Y 座標位置，由於點座標均勻分佈在 0 ~ 1 之間，所以落在某區域之點數目與其面積成正比，只要將落在圓內點數目除以點模擬總數目，即可得圓周率 π 之數值。

[蒙地卡羅方法基本原理]

蒙地卡羅方法到底適合解決哪些問題？ 舉凡所有具有隨機效應的過程，均可能以蒙地卡羅方法大量模擬單一事件，藉統計上平均值獲得某設定條件下實際最可能測量值。現今此方法已被應用在許多領域中。

蒙地卡羅方法的基本原理是將所有可能結果發生的機率，定義出一機率密度函數。將此機率密度函數累加成累積機率函數，調整其值最大值為 1，此稱為歸一化(Normalization)。這也將正確反應出所有事件出現的總機率為 1 的機率特性，這也為亂數取樣與實際問題模擬建立起連結。也就是說我們將電腦所產生均勻分佈於 [0, 1] 之間的亂數，透過所欲模擬的過程所具有機率分佈函數，模擬出實際問題最可能結果。

Monte Carlo Simulation Basics

A **Monte Carlo method** is a technique that involves using random numbers and 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).

Computer simulation has to do with using computer models to imitate real life or make predictions. When you create a model, you have a certain number of input parameters and a few equations that use those inputs to give you a set of outputs (or *response* variables). This type of model is usually **deterministic**, meaning that you get the same results no matter how many times you re-calculate.

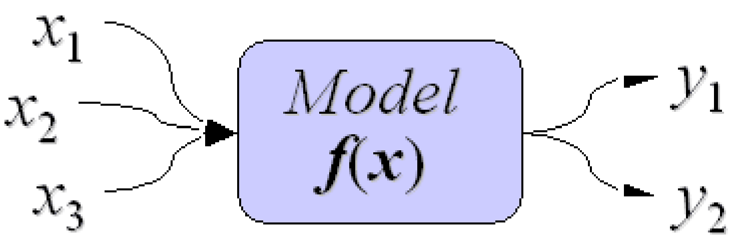


Figure 1: A parametric deterministic model maps a set of input variables to a set of output variables.

Monte Carlo simulation is a method for iteratively evaluating a deterministic model using sets of random numbers as inputs. This method is often used when the model is complex, nonlinear, or involves more than just a couple uncertain parameters. A simulation can typically involve over 10,000 evaluations of the model, a task which in the past was only practical using super computers.

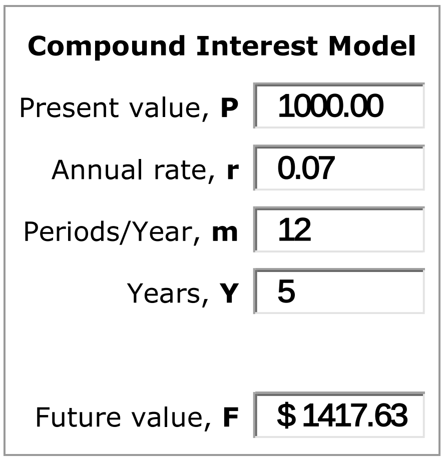
[Example : A Deterministic Model for Compound Interest]

Deterministic Model Example

An example of a **deterministic model** is a calculation to determine the **return on a 5-year investment** with an annual interest rate of 7%, compounded monthly. The **model** is just the equation below:

$$F = P\left(1 + r/m\right)^{Ym}$$

The **inputs** are the initial investment (*P* = $1000), annual interest rate (*r* = 7% = 0.07), the compounding period (*m* = 12 months), and the number of years (*Y* = 5).



Compound Interest Model

Present value, **P**	1000.00
Annual rate, **r**	0.07
Periods/Year, **m**	12
Years, **Y**	5
Future value, **F**	$ 1417.63

One of the purposes of a model such as this is to make predictions and try **"What If?" scenarios**. You can change the inputs and recalculate the model and you'll get a new answer. You might even want to plot a graph of the future value (F) vs. years (Y). In some cases, you may have a fixed interest rate, but what do you do if the interest rate is allowed to change? For this simple equation, you might only care to know a **worst/best case scenario**, where you calculate the future value based upon the lowest and highest interest rates that you might expect.

By using **random inputs**, you are essentially turning the deterministic model into a stochastic model. Example below demonstrates this concept with a very simple problem.

Stochastic Model Example

A stochastic model is one that involves probability or randomness. In this example, we have an assembly of 4 parts that make up a hinge, with a pin or bolt through the centers of the parts. Looking at the figure below, **if A + B + C is greater than D**, we're going to have a hard time putting this thing together.

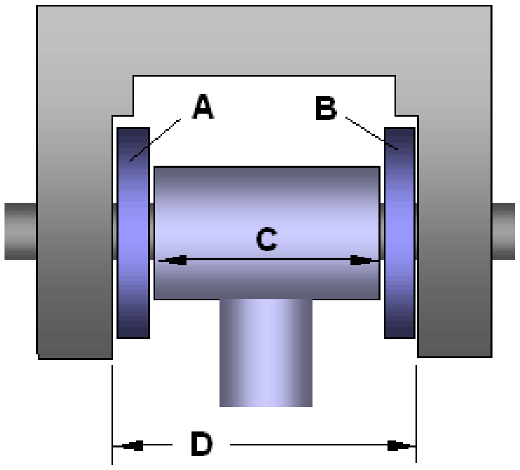


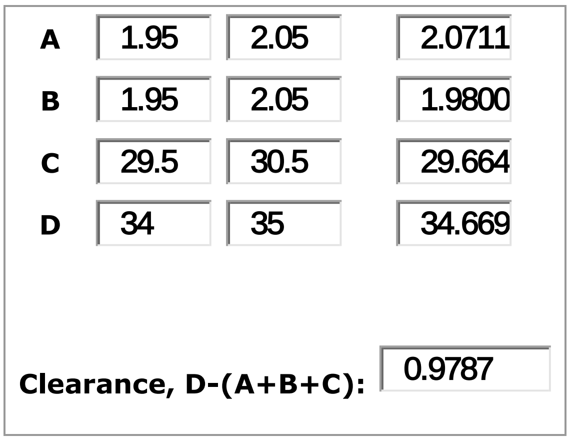
Figure : A hinge.

Let's say we have a million of each of the different parts, and we randomly select the parts we need in order to assemble the hinge. **No two parts are going to be exactly the same size!** But, if we have an idea of the range of sizes for each part, then we can **simulate** the selection and assembly of the parts mathematically.

The table below demonstrates this. Each time you press **"Calculate"**, you are simulating the creation of an assembly from a random set of parts. If you ever get a **negative clearance**, then that means the combination of the parts you have selected will be too large to fit within dimension **D**. Do you ever get a negative clearance?

Tolerance Stack-Up Model

Part	Min	Max	Random



A	1.95	2.05	2.0711
B	1.95	2.05	1.9800
C	29.5	30.5	29.664
D	34	35	34.669

Clearance, D-(A+B+C): 0.9787

This example demonstrates almost all of the steps in a Monte Carlo simulation. The **deterministic model** is simply **D-(A+B+C)**. We are using **uniform distributions** to generate the values for each input. All we need to do now is press the "calculate" button a few thousand times, record all the results, create a histogram to visualize the data, and calculate the probability that the parts cannot be assembled.

Of course, you don't want to do this manually. That is why there is so much software for automating Monte Carlo simulation.

In Example above, we used simple *uniform random numbers* as the inputs to the model. However, a uniform distribution is not the only way to represent uncertainty. Before describing the steps of the general MC simulation in detail, a little word about uncertainty propagation:

The Monte Carlo method is just one of many methods for analyzing **uncertainty propagation**, where the goal is to determine how random variation, lack of knowledge, or error affects the sensitivity, performance, or reliability of the system that is being modeled. Monte Carlo simulation is categorized as a **sampling method** because the inputs are randomly generated from *probability distributions* to simulate the process of sampling from an actual population. So, we try to choose a distribution for the inputs that most closely matches data we already have, or best represents our current state of knowledge. The data generated from the simulation can be represented as probability distributions (or histograms) or converted to error bars, reliability predictions, tolerance zones, and confidence intervals. (See Figure below).

Uncertainty Propagation

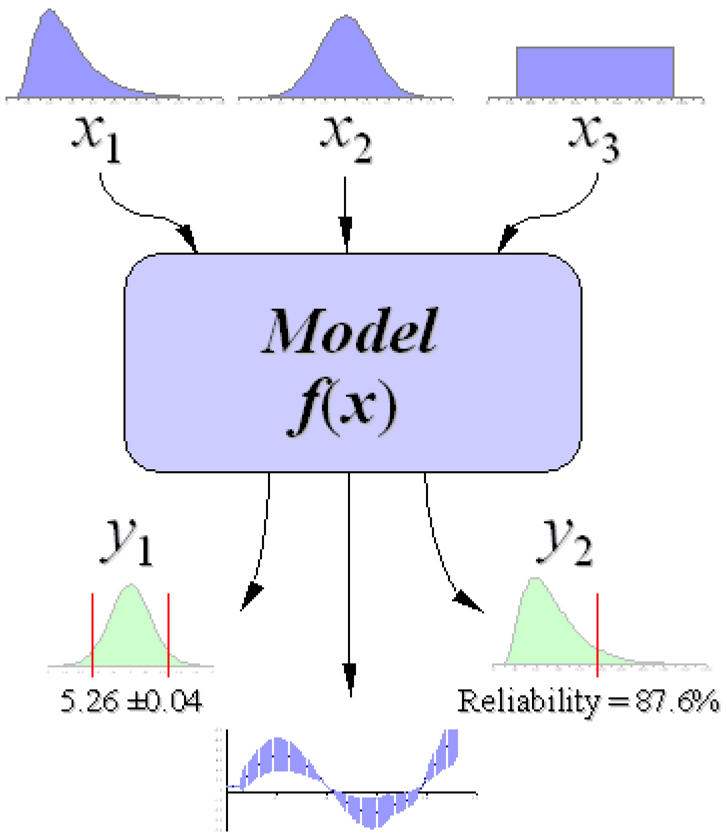


Figure : Schematic showing the principal of stochastic uncertainty propagation. (The basic principle behind Monte Carlo simulation.)

If you have made it this far, **congratulations**! Now for the fun part! The steps in Monte Carlo simulation corresponding to the uncertainty propagation shown in Figure above are fairly simple, and can be easily implemented in software for simple models. All we need to do is follow the **five simple steps** listed below:

Step 1: Create a parametric model, $y = f(x_1, x_2, ..., x_q)$.

Step 2: Generate a set of random inputs, $x_{i1}, x_{i2}, ..., x_{iq}$.

Step 3: Evaluate the model and store the results as y_i.

Step 4: Repeat steps 2 and 3 for $i = 1$ to n.

Step 5: Analyze the results using histograms, summary statistics, confidence intervals, etc.

On to an example problem

Sales Forecasting Example

Our example of Monte Carlo simulation will be a simplified **sales forecast model**. Each step of the analysis will be described in detail.

The Scenario: Company XYZ wants to know how profitable it will be to market their new gadget, realizing there are many uncertainties associated with market size, expenses, and revenue.

The Method: Use a Monte Carlo Simulation to estimate profit and evaluate risk.

Step 1: Creating the Model

We are going to use a top-down approach to create the sales forecast model, starting with:

Profit = Income - Expenses

Both income and expenses are uncertain parameters, but we aren't going to stop here, because one of the purposes of developing a model is to try to break the problem down into more fundamental quantities. **Ideally, we want all the inputs to be *independent***. *Does income depend on expenses?* If so, our model needs to take this into account somehow.

We'll say that *Income* comes solely from the **number of sales** (**S**) multiplied by the **profit per sale** (**P**) resulting from an individual purchase of a gadget, so ***Income* = S*P**. The *profit per sale* takes into account the sale price, the initial cost to manufacturer or purchase the product wholesale, and other transaction fees (credit cards, shipping, etc.). For our purposes, we'll say the **P** may fluctuate between $47 and $53.

We could just leave the number of sales as one of the primary variables, but for this example, Company XYZ generates sales through purchasing leads. The number of sales per month is the **number of leads per month** (**L**) multiplied by the **conversion rate** (**R**) (the percentage of leads that result in sales). So our final equation for *Income* is:

Income = L*R*P

We'll consider the *Expenses* to be a combination of **fixed overhead** (**H**) plus the total cost of the leads. For this model, the **cost of a single lead** (**C**) varies between $0.20 and $0.80. Based upon some market research, Company XYZ expects the number of leads per month (**L**) to vary between 1200 and 1800. Our final model for Company XYZ's sales forecast is:

$$\textbf{Profit} = L*R*P - (H + L*C)$$

$$Y = \textbf{Profits}$$
$$X_1 = L$$
$$X_2 = C$$
$$X_3 = R$$
$$X_4 = P$$

Notice that **H** is also part of the equation, but we are going to treat it as a constant in this example. The inputs to the Monte Carlo simulation are just the uncertain parameters (X_i).

This is not a comprehensive treatment of modeling methods, but I used this example to demonstrate an important concept in uncertainty propagation, namely *correlation*. After breaking *Income* and *Expenses* down into more fundamental and measurable quantities, we found that the number of leads (**L**) affected both income *and* expenses. Therefore, income and expenses are **not independent**. We could probably break the problem down even further, but we won't in this example. We'll assume that L, R, P, H, and C are all independent.

Note: In my opinion, it is easier to decompose a model into independent variables (when possible) than to try to mess with correlation between random inputs.

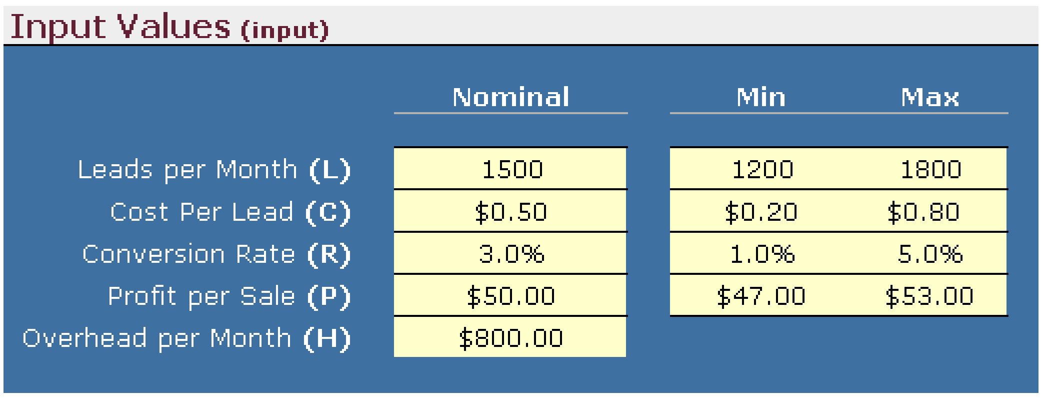
Generating Random Numbers using software

Sales Forecast Example - Part II

Step 2: Generating Random Inputs

The key to Monte Carlo simulation is generating the set of random inputs. As with any modeling and prediction method, the "garbage in equals garbage out" principle applys. For now, I am going to avoid the questions "*How do I know what distribution to use for my inputs?*" and "*How do I make sure I am using a good* *random number generator?*" and get right to the details of how to implement the method in software.

For this example, we're going to use a **Uniform Distribution** to represent the four uncertain parameters. The inputs are summarized in the table shown below.



Input Values **(input)**

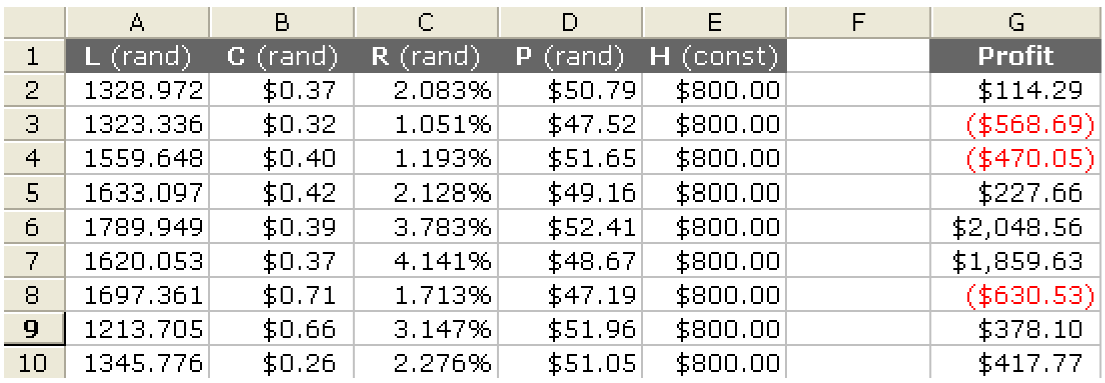
	Nominal	Min	Max
Leads per Month **(L)**	1500	1200	1800
Cost Per Lead **(C)**	$0.50	$0.20	$0.80
Conversion Rate **(R)**	3.0%	1.0%	5.0%
Profit per Sale **(P)**	$50.00	$47.00	$53.00
Overhead per Month **(H)**	$800.00		

The table above uses "Min" and "Max" to indicate the uncertainty in L, C, R, and P. To **generate a random number between "Min" and "Max"**, we use the following formula in software(Replacing "min" and "max" with cell references):

= *min* + RAND()*(*max-min*)

You can also use the **Random Number Generation** tool to kick out a bunch of static random numbers for a few distributions. However, in this example we are going to make use of **RAND()** formula so that every time the worksheet recalculates, a new random number is generated.

Let's say we want to **run *n*=5000 evaluations** of our model. This is a fairly low number when it comes to Monte Carlo simulation, and you will see why once we begin to analyze the results.



	A	B	C	D	E	F	G
1	**L** (rand)	**C** (rand)	**R** (rand)	**P** (rand)	**H** (const)		**Profit**
2	1328.972	$0.37	2.083%	$50.79	$800.00		$114.29
3	1323.336	$0.32	1.051%	$47.52	$800.00		($568.69)
4	1559.648	$0.40	1.193%	$51.65	$800.00		($470.05)
5	1633.097	$0.42	2.128%	$49.16	$800.00		$227.66
6	1789.949	$0.39	3.783%	$52.41	$800.00		$2,048.56
7	1620.053	$0.37	4.141%	$48.67	$800.00		$1,859.63
8	1697.361	$0.71	1.713%	$47.19	$800.00		($630.53)
9	1213.705	$0.66	3.147%	$51.96	$800.00		$378.10
10	1345.776	$0.26	2.276%	$51.05	$800.00		$417.77

Figure : the example sales forecast spreadsheet.

To generate 5000 random numbers for **L**, you simply copy the formula down 5000 rows. You repeat the process for the other variables (except for **H**, which is constant).

Step 3: Evaluating the Model

Since our model is very simple, all we need to do to evaluate the model for each run of the simulation is put the equation in another column next to the inputs, as shown in Figure (the **Profit** column).

Step 4: Run the Simulation

We don't need to write a fancy macro for this example in order to iteratively evaluate our model. We simply copy the formula for profit down 5000 rows, making sure that we use relative references in the formula

Rerun the Simulation:

Although we still need to analyze the data, we have essentially completed a Monte Carlo simulation. Because we have used the volatile RAND() formula, to re-run the simulation all we have to do is recalculate the worksheet.

This may seem like a strange way to implement Monte Carlo simulation, but think about what is going on behind the scenes every time the Worksheet recalculates: **(1)** 5000 sets of random inputs are generated **(2)** The model is evaluated for all 5000 sets. Software is handling all of the iteration.

A Few Other Distributions

To generate a random number from a **Normal (Gaussian) distribution**

=NORMINV(rand(),*mean*,*standard_dev*)

To generate a random number from a **Lognormal distribution** with median = exp(meanlog), and shape = sdlog, you would use the following formula

=LOGINV(RAND(),*meanlog*,*sdlog*)

To generate a random number from a (2-parameter) Weibull distribution with **scale = *c***, and **shape = *m***, you would use the following formula in Excel:

=*c(-LN(1-RAND()))^(1/*m*)**

MORE Distribution Functions: check

Creating a Histogram

Sales Forecast Example - Part III

In Part II of this Monte Carlo Simulation example, we completed the actual simulation. We ended up with a column of 5000 possible values (observations) for our single response variable, ***profit***. The last step is to analyze the results. We will start off by creating a **histogram**, a graphical method for visualizing the results.

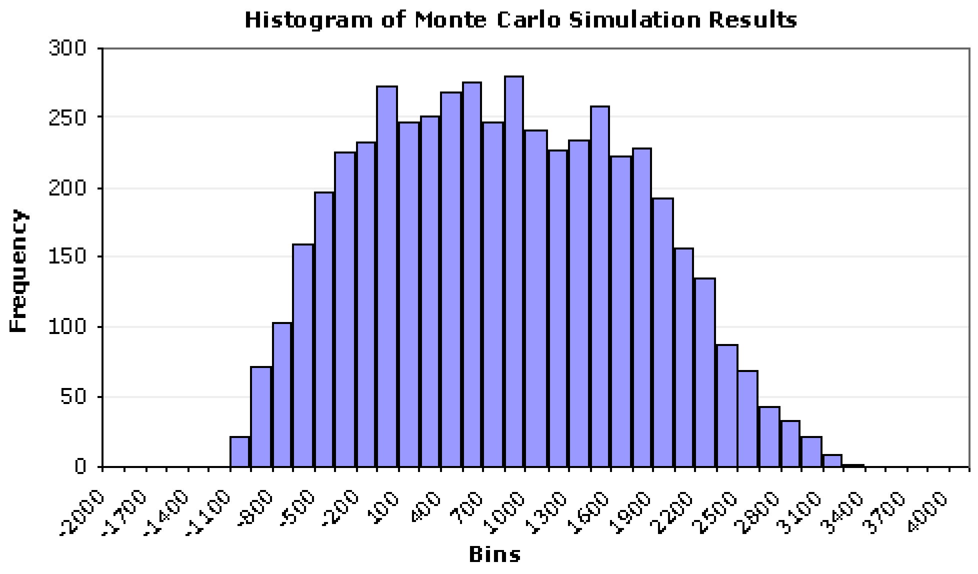


Figure 1: A Histogram created using a Bar Chart.

(From a Monte Carlo simulation using *n* = 5000 points and 40 bins).

We can glean a lot of information from this histogram:

- It looks like profit will be positive, *most* of the time.
- The uncertainty is quite large, varying between -1000 to 3400.
- The distribution does not look like a perfect Normal distribution.
- There doesn't appear to be outliers, truncation, multiple modes, etc.

The histogram tells a good story, but in many cases, we want to estimate the **probability** of being below or above some value, or between a set of specification limits.

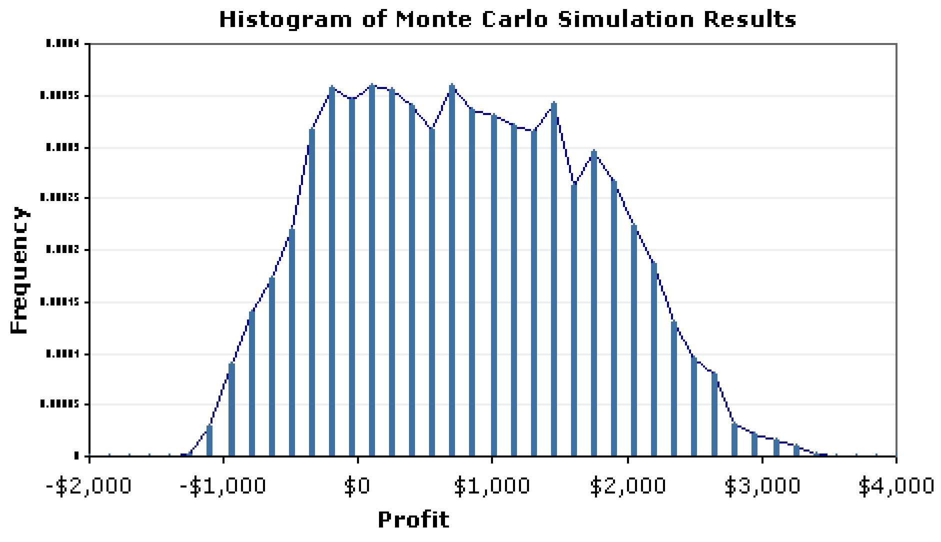


Figure 4: Example Histogram Created Using a Scatter Plot and Error Bars.

Summary Statistics

Sales Forecast Example - Part IV of V

In Part III of this Monte Carlo Simulation example, we plotted the results as a histogram in order to **visualize** the uncertainty in profit. In order to provide a concise summary of the results, it is customary to report the **mean**, **median**, **standard deviation**, standard error, and a few other summary statistics to describe the resulting distribution.

Statistics Formulas

Sample Size (*n*):
Sample Mean:
Median:
Sample Standard Deviation (σ)
Maximum:
Mininum:

Sample Size (*n*)

The **sample size**, ***n***, is the number of **observations** or data points from a single MC simulation. For this example, we obtained *n* = 5000 simulated observations. Because the Monte Carlo method is **stochastic**, if we repeat the simulation, we will end up calculating a different set of summary statistics. The larger the sample size, the smaller the difference will be between the repeated simulations.

Central Tendency: Mean and Median

The **sample mean** and **median** statistics describe the **central tendency** or "location" of the distribution. The **arithmetic mean** is simply the **average** value of the observations.
If you sort the results from lowest to highest, the **median** is the "middle" value or the 50th Percentile, meaning that **50%** of the results from the simulation are less than the median. If there is an even number of data points, then the median is the average of the middle two points.
Extreme values can have a large impact on the mean, but the median only depends upon the middle point(s). This property makes the median useful for describing the center of skewed distributions such as the Lognormal distribution. If the distribution is symmetric (like the Normal distribution), then the mean and median will be identical.

Confidence Intervals for the True Population

Mean

The **sample mean** is just an estimate of the **true population mean**. How accurate is the estimate? You can see by repeating the simulation that the mean is not the same for each simulation.

Standard Error

If you repeated the Monte Carlo simulation and recorded the sample mean each time, the distribution of the sample mean would end up following a Normal distribution (based upon the Central Limit Theorem). The **standard error** is a good estimate of the **standard deviation** of this distribution, assuming that the sample is sufficiently large (*n* >= 30).
The **standard error** is calculated using the following formula:

$$StErr = \frac{s}{\sqrt{n}}$$

95% Confidence Interval

The standard error can be used to calculate **confidence intervals for the true population mean**. For a 95% **2-sided** confidence interval, the Upper Confidence Limit (UCL) and Lower Confidence Limit (LCL) are calculated as:

$$95\%UCL = Mean + 1.96 \times StErr = \bar{y} + 1.96\frac{s}{\sqrt{n}}$$

$$95\%LCL = Mean - 1.96 \times StErr = \bar{y} - 1.96\frac{s}{\sqrt{n}}$$

To get a 90% or 99% confidence interval, you would change the value 1.96 to 1.645 or 2.575, respectively. The value 1.96

represents the **97.5** percentile of the standard normal distribution. (You may often see this number rounded to 2).

Percentiles and Cumulative Probabilities

Sales Forecast Example - Part V of V

As a final step in the sales forecast example, we are going to look at how to use the **percentile function** and **percent rank function** to estimate important summary statistics from our Monte Carlo simulation results. But first, it will be helpful to talk a bit about the cumulative probability distribution.

Creating a Cumulative Distribution

In Part III of this Monte Carlo Simulation example, we plotted the results as a histogram in order to **visualize** the uncertainty in profit. We are going to augment the histogram by including a graph of the **estimated cumulative distribution function** (CDF) as shown below.

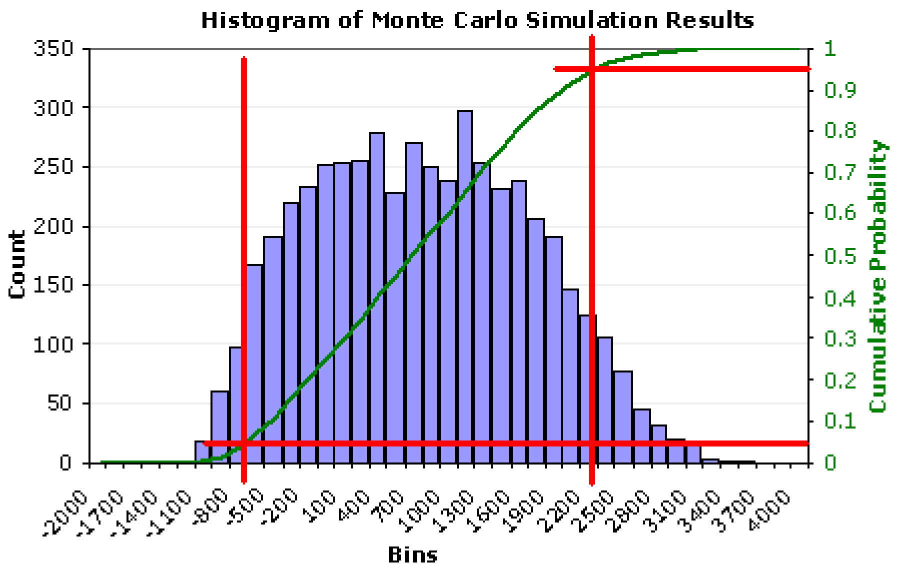


Figure 1: Graph of the estimated cumulative distribution.

The reason for showing the CDF along with the histogram is to demonstrate that an estimate of the cumulative probability is simply the **percentage** of the **data points** to the left of the point of interest.

For example, we might want to know **what percentage of the results was less than -$700.00** (the vertical red line on the left). From the graph, the corresponding cumulative probability is about 0.05 or 5%. Similarly, we can draw a line at $2300 and find that about 95% of the results are less than $2300.

It is fairly simple to create the cumulative distribution. Figure 2 shows how you can estimate the CDF by calculating the probabilities using a **cumulative sum of the count** from the frequency function. You simply divide the cumulative sum by the total number of points.