12 Simulation methods

Learning Outcomes

In this chapter, you will learn how to

- Design simulation frameworks to solve a variety of problems in finance
- Explain the difference between pure simulation and bootstrapping
- Describe the various techniques available for reducing Monte Carlo sampling variability
- Implement a simulation analysis in EViews

12.1 Motivations

There are numerous situations, in finance and in econometrics, where the researcher has essentially no idea what is going to happen! To offer one illustration, in the context of complex financial risk measurement models for portfolios containing large numbers of assets whose movements are dependent on one another, it is not always clear what will be the effect of changing circumstances. For example, following full European monetary union (EMU) and the replacement of member currencies with the euro, it is widely believed that European financial markets have become more integrated, leading the correlation between movements in their equity markets to rise. What would be the effect on the properties of a portfolio containing equities of several European countries if correlations between the markets rose to 99%? Clearly, it is probably not possible to be able to answer such a question using actual historical data alone, since the event (a correlation of 99%) has not yet happened.

The practice of econometrics is made difficult by the behaviour of series and inter-relationships between them that render model assumptions at best questionable. For example, the existence of fat tails, structural breaks and bi-directional causality between dependent and independent variables, etc. will make the process of parameter estimation and inference less reliable. Real data is messy, and no one really knows all of the features that lurk inside it. Clearly, it is important for researchers to have an idea of what the effects of such phenomena will be for model estimation and inference.

By contrast, simulation is the econometrician's chance to behave like a real scientist, conducting experiments under controlled conditions. A simulations experiment enables the econometrician to determine what the effect of changing one factor or aspect of a problem will be, while leaving all other aspects unchanged. Thus, simulations offer the possibility of complete flexibility. Simulation may be defined as an approach to modelling that seeks to mimic a functioning system as it evolves. The simulations model will express in mathematical equations the assumed form of operation of the system. In econometrics, simulation is particularly useful when models are very complex or sample sizes are small.

12.2 Monte Carlo simulations

Simulations studies are usually used to investigate the properties and behaviour of various statistics of interest. The technique is often used in econometrics when the properties of a particular estimation method are not known. For example, it may be known from asymptotic theory how a particular test behaves with an infinite sample size, but how will the test behave if only 50 observations are available? Will the test still have the desirable properties of being correctly sized and having high power? In other words, if the null hypothesis is correct, will the test lead to rejection of the null 5% of the time if a 5% rejection region is used? And if the null is incorrect, will it be rejected a high proportion of the time?

Examples from econometrics of where simulation may be useful include:

- Quantifying the simultaneous equations bias induced by treating an endogenous variable as exogenous
- Determining the appropriate critical values for a Dickey–Fuller test
- Determining what effect heteroscedasticity has upon the size and power of a test for autocorrelation.

Box 12.1 Conducting a Monte Carlo simulation

(1) Generate the data according to the desired data generating process (DGP), with the errors being drawn from some given distribution
(2) Do the regression and calculate the test statistic
(3) Save the test statistic or whatever parameter is of interest
(4) Go back to stage 1 and repeat N times.

Simulations are also often extremely useful tools in finance, in situations such as:

- The pricing of exotic options, where an analytical pricing formula is unavailable
- Determining the effect on financial markets of substantial changes in the macroeconomic environment
- 'Stress-testing' risk management models to determine whether they generate capital requirements sufficient to cover losses in all situations.

In all of these instances, the basic way that such a study would be conducted (with additional steps and modifications where necessary) is shown in box 12.1.

A brief explanation of each of these steps is in order. The first stage involves *specifying the model* that will be used to generate the data. This may be a pure time series model or a structural model. Pure time series models are usually simpler to implement, as a full structural model would also require the researcher to specify a data generating process for the explanatory variables as well. Assuming that a time series model is deemed appropriate, the next choice to be made is of the *probability distribution* specified for the errors. Usually, standard normal draws are used, although any other empirically plausible distribution (such as a Student's t) could also be used.

The second stage involves estimation of the parameter of interest in the study. The parameter of interest might be, for example, the value of a coefficient in a regression, or the value of an option at its expiry date. It could instead be the value of a portfolio under a particular set of scenarios governing the way that the prices of the component assets move over time.

The quantity N is known as the number of replications, and this should be as large as is feasible. The central idea behind Monte Carlo is that of random sampling from a given distribution. Therefore, if the number of replications is set too small, the results will be sensitive to 'odd' combinations of random number draws. It is also worth noting that asymptotic arguments apply in Monte Carlo studies as well as in other areas of econometrics. That is, the results of a simulation study will be equal to their analytical counterparts (assuming that the latter exist) asymptotically.

12.3 Variance reduction techniques

Suppose that the value of the parameter of interest for replication i is denoted x_i. If the average value of this parameter is calculated for a set of, say, $N = 1{,}000$ replications, and another researcher conducts an otherwise identical study with different sets of random draws, a different average value of x is almost certain to result. This situation is akin to the problem of selecting only a sample of observations from a given population in standard regression analysis. The sampling variation in a Monte Carlo study is measured by the standard error estimate, denoted S_x

$$S_x = \sqrt{\frac{\text{var}(x)}{N}} \tag{12.1}$$

where var(x) is the variance of the estimates of the quantity of interest over the N replications. It can be seen from this equation that to reduce the Monte Carlo standard error by a factor of 10, the number of replications must be increased by a factor of 100. Consequently, in order to achieve acceptable accuracy, the number of replications may have to be set at an infeasibly high level. An alternative way to reduce Monte Carlo sampling error is to use a variance reduction technique. There are many variance reduction techniques available. Two of the intuitively simplest and most widely used methods are the method of *antithetic variates* and the method of *control variates*. Both of these techniques will now be described.

12.3.1 *Antithetic variates*

One reason that a lot of replications are typically required of a Monte Carlo study is that it may take many, many repeated sets of sampling before the entire probability space is adequately covered. By their very nature, the values of the random draws are random, and so after a given number of replications, it may be the case that not the whole range of possible outcomes has actually occurred.[1] What is really required is for successive replications to cover different parts of the probability space – that

[1] Obviously, for a continuous random variable, there will be an infinite number of possible values. In this context, the problem is simply that if the probability space is split into arbitrarily small intervals, some of those intervals will not have been adequately covered by the random draws that were actually selected.

is, for the random draws from different replications to generate outcomes that span the entire spectrum of possibilities. This may take a long time to achieve naturally.

The antithetic variate technique involves taking the complement of a set of random numbers and running a parallel simulation on those. For example, if the driving stochastic force is a set of T $N(0, 1)$ draws, denoted u_t, for each replication, an additional replication with errors given by $-u_t$ is also used. It can be shown that the Monte Carlo standard error is reduced when antithetic variates are used. For a simple illustration of this, suppose that the average value of the parameter of interest across 2 sets of Monte Carlo replications is given by

$$\bar{x} = (x_1 + x_2)/2 \tag{12.2}$$

where x_1 and x_2 are the average parameter values for replications sets 1 and 2, respectively. The variance of $\bar{x}$ will be given by

$$\text{var}(\bar{x}) = \frac{1}{4}(\text{var}(x_1) + \text{var}(x_2) + 2\text{cov}(x_1, x_2)) \tag{12.3}$$

If no antithetic variates are used, the two sets of Monte Carlo replications will be independent, so that their covariance will be zero, i.e.

$$\text{var}(\bar{x}) = \frac{1}{4}(\text{var}(x_1) + \text{var}(x_2)) \tag{12.4}$$

However, the use of antithetic variates would lead the covariance in (12.3) to be negative, and therefore the Monte Carlo sampling error to be reduced.

It may at first appear that the reduction in Monte Carlo sampling variation from using antithetic variates will be huge since, by definition, $\text{corr}(u_t, -u_t) = \text{cov}(u_t, -u_t) = -1$. However, it is important to remember that the relevant covariance is between the simulated quantity of interest for the standard replications and those using the antithetic variates. But the perfect negative covariance is between the random draws (i.e. the error terms) and their antithetic variates. For example, in the context of option pricing (discussed below), the production of a price for the underlying security (and therefore for the option) constitutes a non-linear transformation of u_t. Therefore the covariances between the terminal prices of the underlying assets based on the draws and based on the antithetic variates will be negative, but not -1.

Several other variance reduction techniques that operate using similar principles are available, including stratified sampling, moment-matching and low-discrepancy sequencing. The latter are also known as *quasi-random sequences* of draws. These involve the selection of a specific sequence of representative samples from a given probability distribution. Successive samples are selected so that the unselected gaps left in the probability distribution are filled by subsequent replications. The result is a set of random draws that are appropriately distributed across all of the outcomes of interest. The use of low-discrepancy sequences leads the Monte Carlo standard errors to be reduced in direct proportion to the number of replications rather than in proportion to the square root of the number of replications. Thus, for example, to reduce the Monte Carlo standard error by a factor of 10, the number of replications would have to be increased by a factor of 100 for standard Monte Carlo random sampling, but only 10 for low-discrepancy sequencing. Further details of low-discrepancy techniques are beyond the scope of this text, but can be seen in Boyle (1977) or Press *et al*. (1992). The former offers a detailed and relevant example in the context of options pricing.

12.3.2 Control variates

The application of control variates involves employing a variable similar to that used in the simulation, but whose properties are known prior to the simulation. Denote the variable whose properties are known by y, and that whose properties are under simulation by x. The simulation is conducted on x and also on y, with the same sets of random number draws being employed in both cases. Denoting the simulation estimates of x and y by $\hat{x}$ and $\hat{y}$, respectively, a new estimate of x can be derived from

$$x^* = y + (\hat{x} - \hat{y}) \tag{12.5}$$

Again, it can be shown that the Monte Carlo sampling error of this quantity, x^*, will be lower than that of x provided that a certain condition holds. The control variates help to reduce the Monte Carlo variation owing to particular sets of random draws by using the same draws on a related problem whose solution is known. It is expected that the effects of sampling error for the problem under study and the known problem will be similar, and hence can be reduced by calibrating the Monte Carlo results using the analytic ones.

It is worth noting that control variates succeed in reducing the Monte Carlo sampling error only if the control and simulation problems are very closely related. As the correlation between the values of the control statistic and the statistic of interest is reduced, the variance reduction is weakened. Consider again (12.5), and take the variance of both sides

$$\text{var}(x^*) = \text{var}(y + (\hat{x} - \hat{y})) \tag{12.6}$$

$\text{var}(y) = 0$ since y is a quantity which is known analytically and is therefore not subject to sampling variation, so (12.6) can be written

$$\text{var}(x^*) = \text{var}(\hat{x}) + \text{var}(\hat{y}) - 2\text{cov}(\hat{x}, \hat{y}) \tag{12.7}$$

The condition that must hold for the Monte Carlo sampling variance to be lower with control variates than without is that $\text{var}(x^*)$ is less than $\text{var}(\hat{x})$. Taken from (12.7), this condition can also be expressed as

$$\text{var}(\hat{y}) - 2\text{cov}(\hat{x}, \hat{y}) < 0$$

or

$$\text{cov}(\hat{x}, \hat{y}) > \frac{1}{2}\text{var}(\hat{y})$$

Divide both sides of this inequality by the products of the standard deviations, i.e. by $(\text{var}(\hat{x}), \text{var}(\hat{y}))^{1/2}$, to obtain the correlation on the LHS

$$\text{corr}(\hat{x}, \hat{y}) > \frac{1}{2}\sqrt{\frac{\text{var}(\hat{y})}{\text{var}(\hat{x})}}$$

To offer an illustration of the use of control variates, a researcher may be interested in pricing an arithmetic Asian option using simulation. Recall that an arithmetic Asian option is one whose payoff depends on the arithmetic average value of the underlying asset over the lifetime of the averaging; at the time of writing, an analytical (closed-form) model is not yet available for pricing such options. In this context, a control variate price could be obtained by finding the price via simulation of a similar derivative whose value is known analytically – e.g. a vanilla European option. Thus, the Asian and vanilla options would be priced using simulation, as shown below, with the simulated price given by P_A and P_{BS}^*, respectively. The price of the vanilla option, P_{BS} is also calculated using an analytical formula, such as Black–Scholes. The new estimate of the Asian option price, P_A^*, would then be given by

$$P_A^* = (P_A - P_{BS}) + P_{BS}^* \tag{12.8}$$

12.3.3 Random number re-usage across experiments

Although of course it would not be sensible to re-use sets of random number draws within a Monte Carlo experiment, using the same sets of draws across experiments can greatly reduce the variability of the difference in the estimates across those experiments. For example, it may be of interest to examine the power of the Dickey–Fuller test for samples of size 100 observations and for different values of ϕ (to use the notation of chapter 7). Thus, for each experiment involving a different value of ϕ, the same set of standard normal random numbers could be used to reduce the sampling variation across experiments. However, the accuracy of the actual estimates in each case will not be increased, of course.

Another possibility involves taking long series of draws and then slicing them up into several smaller sets to be used in different experiments. For example, Monte Carlo simulation may be used to price several options of different times to maturity, but which are identical in all other respects. Thus, if 6-month, 3-month and 1-month horizons were of interest, sufficient random draws to cover 6 months would be made. Then the 6-months' worth of draws could be used to construct two replications of a 3-month horizon, and six replications for the 1-month horizon. Again, the variability of the simulated option prices across maturities would be reduced, although the accuracies of the prices themselves would not be increased for a given number of replications.

Random number re-usage is unlikely to save computational time, for making the random draws usually takes a very small proportion of the overall time taken to conduct the whole experiment.

12.4 Bootstrapping

Bootstrapping is related to simulation, but with one crucial difference. With simulation, the data are constructed completely artificially. Bootstrapping, on the other hand, is used to obtain a description of the properties of empirical estimators by using the sample data points themselves, and it involves sampling repeatedly with replacement from the actual data. Many econometricians were initially highly sceptical of the usefulness of the technique, which appears at first sight to be some kind of magic trick – creating useful additional information from a given sample. Indeed, Davison and Hinkley (1997, p. 3), state that the term 'bootstrap' in this context comes from an analogy with the fictional character Baron Munchhausen, who got out from the bottom of a lake by pulling himself up by his bootstraps.

Suppose a sample of data, $\mathbf{y} = y_1, y_2, \ldots, y_T$ are available and it is desired to estimate some parameter θ. An approximation to the statistical properties of $\hat{\theta}_T$ can be obtained by studying a sample of bootstrap estimators. This is done by taking N samples of size T with replacement from $\mathbf{y}$ and re-calculating $\hat{\theta}$ with each new sample. A series of $\hat{\theta}$ estimates is then obtained, and their distribution can be considered.

The advantage of bootstrapping over the use of analytical results is that it allows the researcher to make inferences without making strong distributional assumptions, since the distribution employed will be that of the actual data. Instead of imposing a shape on the sampling distribution of the $\hat{\theta}$ value, bootstrapping involves empirically estimating the sampling distribution by looking at the variation of the statistic within-sample.

A set of new samples is drawn with replacement from the sample and the test statistic of interest calculated from each of these. Effectively, this involves sampling from the sample, i.e. treating the sample as a population from which samples can be drawn. Call the test statistics calculated from the new samples $\hat{\theta}^*$. The samples are likely to be quite different from each other and from the original $\hat{\theta}$ value, since some observations may be sampled several times and others not at all. Thus a distribution of values of $\hat{\theta}^*$ is obtained, from which standard errors or some other statistics of interest can be calculated.

Along with advances in computational speed and power, the number of bootstrap applications in finance and in econometrics have increased rapidly in previous years. For example, in econometrics, the bootstrap has been used in the context of unit root testing. Scheinkman and LeBaron (1989) also suggest that the bootstrap can be used as a 'shuffle diagnostic', where as usual the original data are sampled with replacement to form new data series. Successive applications of this procedure should generate a collection of data sets with the same distributional properties, on average, as the original data. But any kind of dependence in the original series (e.g. linear or non-linear autocorrelation) will, by definition, have been removed. Applications of econometric tests to the shuffled series can then be used as a benchmark with which to compare the results on the actual data or to construct standard error estimates or confidence intervals.

In finance, an application of bootstrapping in the context of risk management is discussed below. Another important recent proposed use of the bootstrap is as a method for detecting data snooping (data mining) in the context of tests of the profitability of technical trading rules. Data snooping occurs when the same set of data is used to construct trading rules and also to test them. In such cases, if a sufficient number of trading rules are examined, some of them are bound, purely by chance alone, to generate statistically significant positive returns. Intra-generational data snooping is said to occur when, over a long period of time, technical trading rules that 'worked' in the past continue to be examined, while the ones that did not fade away. Researchers are then made aware of only the rules that worked, and not the other, perhaps thousands, of rules that failed.

Data snooping biases are apparent in other aspects of estimation and testing in finance. Lo and MacKinlay (1990) find that tests of financial asset pricing models (CAPM) may yield misleading inferences when properties of the data are used to construct the test statistics. These properties relate to the construction of portfolios based on some empirically motivated characteristic of the stock, such as market capitalisation, rather than a theoretically motivated characteristic, such as dividend yield.

Sullivan, Timmermann and White (1999) and White (2000) propose the use of a bootstrap to test for data snooping. The technique works by placing the rule under study in the context of a 'universe' of broadly similar trading rules. This gives some empirical content to the notion that a variety of rules may have been examined before the final rule is selected. The bootstrap is applied to each trading rule, by sampling with replacement from the time series of observed returns for that rule. The null hypothesis is that there does not exist a superior technical trading rule. Sullivan, Timmermann and White show how a p-value of the 'reality check' bootstrap-based test can be constructed, which evaluates the significance of the returns (or excess returns) to the rule after allowing for the fact that the whole universe of rules may have been examined.

12.4.1 *An example of bootstrapping in a regression context*

Consider a standard regression model

$$y = X\beta + u \tag{12.9}$$

The regression model can be bootstrapped in two ways.

Re-sample the data

This procedure involves taking the data, and sampling the entire rows corresponding to observation i together. The steps would then be as shown in box 12.2.

A methodological problem with this approach is that it entails sampling from the regressors, and yet under the CLRM, these are supposed to be

Box 12.2 Re-sampling the data

(1) Generate a sample of size T from the original data by sampling with replacement from the whole rows taken together (that is, if observation 32 is selected, take y_{32} and all values of the explanatory variables for observation 32).
(2) Calculate $\hat{\beta}^*$, the coefficient matrix for this bootstrap sample.
(3) Go back to stage 1 and generate another sample of size T. Repeat these stages a total of N times. A set of N coefficient vectors, $\hat{\beta}^*$, will thus be obtained and in general they will all be different, so that a distribution of estimates for each coefficient will result.

Box 12.3 Re-sampling from the residuals

(1) Estimate the model on the actual data, obtain the fitted values $\hat{y}$, and calculate the residuals, $\hat{u}$
(2) Take a sample of size T with replacement from these residuals (and call these $\hat{u}^*$), and generate a bootstrapped-dependent variable by adding the fitted values to the bootstrapped residuals

$$y^* = \hat{y} + \hat{u}^* \tag{12.10}$$

(3) Then regress this new dependent variable on the original X data to get a bootstrapped coefficient vector, $\hat{\beta}^*$
(4) Go back to stage 2, and repeat a total of N times.

fixed in repeated samples, which would imply that they do not have a sampling distribution. Thus, resampling from the data corresponding to the explanatory variables is not in the spirit of the CLRM.

As an alternative, the only random influence in the regression is the errors, u, so why not just bootstrap from those?

Re-sampling from the residuals

This procedure is ‘theoretically pure’ although harder to understand and to implement. The steps are shown in box 12.3.

12.4.2 *Situations where the bootstrap will be ineffective*

There are at least two situations where the bootstrap, as described above, will not work well.

Outliers in the data

If there are *outliers* in the data, the conclusions of the bootstrap may be affected. In particular, the results for a given replication may depend critically on whether the outliers appear (and how often) in the bootstrapped sample.

Non-independent data

Use of the bootstrap implicitly assumes that the data are *independent of one another*. This would obviously not hold if, for example, there were autocorrelation in the data. A potential solution to this problem is to use a ‘moving block bootstrap’. Such a method allows for the dependence in the series by sampling whole blocks of observations at a time. These, and many other issues relating to the theory and practical usage of the bootstrap are given in Davison and Hinkley (1997); see also Efron (1979;1982).

It is also worth noting that variance reduction techniques are also available under the bootstrap, and these work in a very similar way to those described above in the context of pure simulation.

12.5 Random number generation

Most econometrics computer packages include a random number generator. The simplest class of numbers to generate are from a uniform (0,1) distribution. A uniform (0,1) distribution is one where only values between zero and one are drawn, and each value within the interval has an equal chance of being selected. Uniform draws can be either discrete or continuous. An example of a discrete uniform number generator would be a die or a roulette wheel. Computers generate continuous uniform random number draws.

Numbers that are a continuous uniform (0,1) can be generated according to the following recursion

$$y_{i+1} = (ay_i + c) \text{ modulo } m,\ i = 0, 1, \ldots, T \tag{12.11}$$

then

$$R_{i+1} = y_{i+1}/m \text{ for } i = 0, 1, \ldots, T \tag{12.12}$$

for T random draws, where y_0 is the seed (the initial value of y), a is a multiplier and c is an increment. All three of these are simply constants. The 'modulo operator' simply functions as a clock, returning to one after reaching m.

Any simulation study involving a recursion, such as that described by (12.11) to generate the random draws, will require the user to specify an initial value, y_0, to get the process started. The choice of this value will, undesirably, affect the properties of the generated series. This effect will be strongest for y_1, $y_2, \ldots$, but will gradually die away. For example, if a set of random draws is used to construct a time series that follows a GARCH process, early observations on this series will behave less like the GARCH process required than subsequent data points. Consequently, a good simulation design will allow for this phenomenon by generating more data than are required and then dropping the first few observations. For example, if 1,000 observations are required, 1,200 observations might be generated, with observations 1 to 200 subsequently deleted and 201 to 1,200 used to conduct the analysis.

These computer-generated random number draws are known as *pseudo-random numbers*, since they are in fact not random at all, but entirely deterministic, since they have been derived from an exact formula! By carefully choosing the values of the user-adjustable parameters, it is possible to get the pseudo-random number generator to meet all the statistical properties of true random numbers. Eventually, the random number sequences will start to repeat, but this should take a long time to happen. See Press *et al*. (1992) for more details and Fortran code, or Greene (2002) for an example.

The U(0,1) draws can be transformed into draws from any desired distribution – for example a normal or a Student's *t*. Usually, econometric software packages with simulations facilities would do this automatically.

12.6 Disadvantages of the simulation approach to econometric or financial problem solving

- *It might be computationally expensive*
  That is, the number of replications required to generate precise solutions may be very large, depending upon the nature of the task at hand. If each replication is relatively complex in terms of estimation issues, the problem might be computationally infeasible, such that it could take days, weeks or even years to run the experiment. Although CPU time is becoming ever cheaper as faster computers are brought to market, the technicality of the problems studied seems to accelerate just as quickly!
- *The results might not be precise*
  Even if the number of replications is made very large, the simulation experiments will not give a precise answer to the problem if some unrealistic assumptions have been made of the data generating process. For example, in the context of option pricing, the option valuations obtained from a simulation will not be accurate if the data generating process assumed normally distributed errors while the actual underlying returns series is fat-tailed.
- *The results are often hard to replicate*
  Unless the experiment has been set up so that the sequence of random draws is known and can be reconstructed, which is rarely done in practice, the results of a Monte Carlo study will be somewhat specific to the given investigation. In that case, a repeat of the experiment would involve different sets of random draws and therefore would be likely to yield different results, particularly if the number of replications is small.
- *Simulation results are experiment-specific*
  The need to specify the data generating process using a single set of equations or a single equation implies that the results could apply to

only that exact type of data. Any conclusions reached may or may not hold for other data generating processes. To give one illustration, examining the power of a statistical test would, by definition, involve determining how frequently a wrong null hypothesis is rejected. In the context of DF tests, for example, the power of the test as determined by a Monte Carlo study would be given by the percentage of times that the null of a unit root is rejected. Suppose that the following data generating process is used for such a simulation experiment

$$y_t = 0.99y_{t-1} + u_t, \qquad u_t \sim \mathrm{N}(0, 1) \tag{12.13}$$

Clearly, the null of a unit root would be wrong in this case, as is necessary to examine the power of the test. However, for modest sample sizes, the null is likely to be rejected quite infrequently. It would not be appropriate to conclude from such an experiment that the DF test is generally not powerful, since in this case the null ($\phi = 1$) is not very wrong! This is a general problem with many Monte Carlo studies. The solution is to run simulations using as many different and relevant data generating processes as feasible. Finally, it should be obvious that the Monte Carlo data generating process should match the real-world problem of interest as far as possible.

To conclude, simulation is an extremely useful tool that can be applied to an enormous variety of problems. The technique has grown in popularity over the past decade, and continues to do so. However, like all tools, it is dangerous in the wrong hands. It is very easy to jump into a simulation experiment without thinking about whether such an approach is valid or not.

12.7 An example of Monte Carlo simulation in econometrics: deriving a set of critical values for a Dickey–Fuller test

Recall, that the equation for a Dickey–Fuller (DF) test applied to some series y_t is the regression

$$y_t = \phi y_{t-1} + u_t \tag{12.14}$$

so that the test is one of H$_0$: $\phi = 1$ against H$_1$: $\phi < 1$. The relevant test statistic is given by

$$\tau = \frac{\hat{\phi} - 1}{SE(\hat{\phi})} \tag{12.15}$$

Box 12.4 Setting up a Monte Carlo simulation

(1) Construct the data generating process under the null hypothesis – that is, obtain a series for y that follows a unit root process. This would be done by:
- Drawing a series of length T, the required number of observations, from a normal distribution. This will be the error series, so that $u_t \sim N(0,1)$.
- Assuming a first value for y, i.e. a value for y at time $t = 1$.
- Constructing the series for y recursively, starting with y_2, y_3, and so on

$$\begin{aligned} y_2 &= y_1 + u_2 \\ y_3 &= y_2 + u_3 \\ &\dots \\ y_T &= y_{T-1} + u_T \end{aligned} \tag{12.16}$$

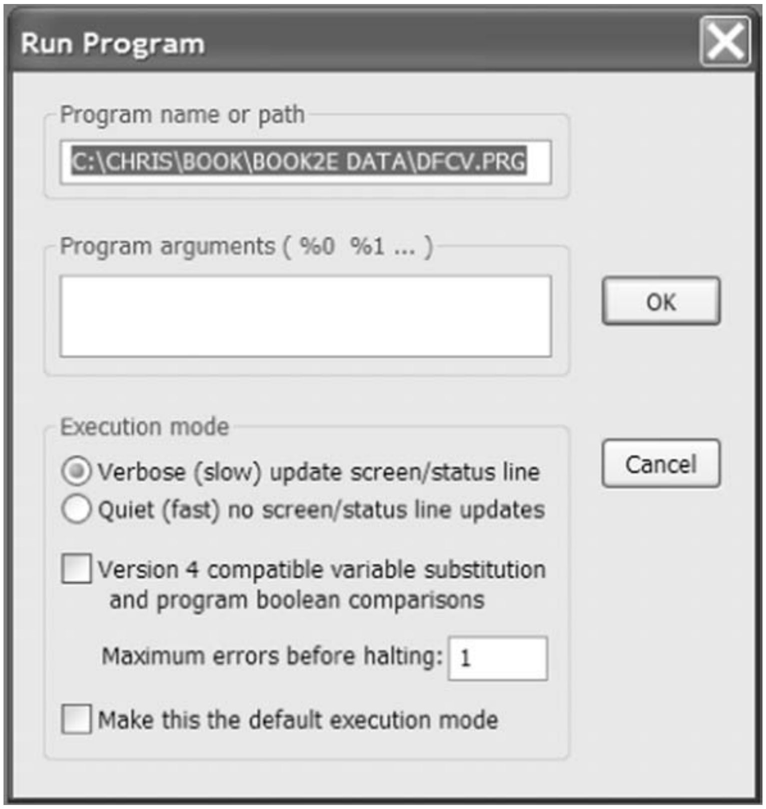
(2) Calculating the test statistic, τ.
(3) Repeating steps 1 and 2 N times to obtain N replications of the experiment. A distribution of values for τ will be obtained across the replications.
(4) Ordering the set of N values of τ from the lowest to the highest. The relevant 5% critical value will be the 5th percentile of this distribution.

Under the null hypothesis of a unit root, the test statistic does not follow a standard distribution, and therefore a simulation would be required to obtain the relevant critical values. Obviously, these critical values are well documented, but it is of interest to see how one could generate them. A very similar approach could then potentially be adopted for situations where there has been less research and where the results are relatively less well known.

The simulation would be conducted in the four steps shown in box 12.4. Some EViews code for conducting such a simulation is given below. The objective is to develop a set of critical values for Dickey–Fuller test regressions. The simulation framework considers sample sizes of 1,000, 500 and 100 observations. For each of these sample sizes, regressions with no constant or trend, a constant but no trend, and a constant and trend are conducted. 50,000 replications are used in each case, and the critical values for a 1-sided test at the 1%, 5% and 10% levels are determined. The code can be found pre-written in a file entitled ‘dfcv.prg’.

EViews programs are simply sets of instructions saved as plain text, so that they can be written from within EViews, or using a word processor or text editor. EViews program files must have a ‘.PRG’ suffix. There are several ways to run the programs once written, but probably the simplest is to write all of the instructions first, and to save them. Then open the EViews software and choose **File, Open and Program**, and when prompted select the directory and file for the instructions. The program containing the instructions will then appear on the screen. To run the program, click on the **Run** button. EViews will then open a dialog box with several options, including whether to run the program in 'Verbose' or 'Quiet' mode. Choose Verbose mode to see the instruction line that is being run at each point in its execution (i.e. the screen is continually updated). This is useful for debugging programs or for running short programs. Choose Quiet to run the program without updating the screen display as it is running, which will make it execute (considerably) more quickly. The screen would appear as in screenshot 12.1.

Screenshot 12.1
Running an EViews program



Program name or path: C:\CHRIS\BOOK\BOOK2E DATA\DFCV.PRG

Program arguments (%0 %1 ...):

Execution mode
[✓ Verbose (slow) update screen/status line]
Quiet (fast) no screen/status line updates

☐ Version 4 compatible variable substitution and program boolean comparisons

Maximum errors before halting: 1

☐ Make this the default execution mode

Then click **OK** and off it goes! The following lists the instructions that are contained in the program, and the discussion below explains what each line does.

```
'NEW WORKFILE CREATED CALLED DF_CV, UNDATED
'WITH 50000 OBSERVATIONS
  WORKFILE DF_CV U 50000
  RNDSEED 12345
```

```
SERIES T1
SERIES T2
SERIES T3
SCALAR K1
SCALAR K2
SCALAR K3
SCALAR K4
SCALAR K5
SCALAR K6
SCALAR K7
SCALAR K8
SCALAR K9
!NREPS=50000
!NOBS=1000
FOR !REPC=1 TO !NREPS
SMPL @FIRST @FIRST
SERIES Y1=0
SMPL @FIRST+1   !NOBS+200
SERIES Y1=Y1(−1)+NRND
SERIES DY1=Y1-Y1(−1)
SMPL @FIRST+200   !NOBS+200
EQUATION EQ1.LS DY1 Y1(−1)
T1(!REPC)=@TSTATS(1)
EQUATION EQ2.LS DY1 C Y1(−1)
T2(!REPC)=@TSTATS(2)
EQUATION EQ3.LS DY1 C @TREND Y1(−1)
T3(!REPC)=@TSTATS(3)
NEXT
SMPL @FIRST !NREPS
K1=@QUANTILE(T1,0.01)
K2=@QUANTILE(T1,0.05)
K3=@QUANTILE(T1,0.1)
K4=@QUANTILE(T2,0.01)
K5=@QUANTILE(T2,0.05)
K6=@QUANTILE(T2,0.1)
K7=@QUANTILE(T3,0.01)
K8=@QUANTILE(T3,0.05)
K9=@QUANTILE(T3,0.1)
```

Although there are probably more efficient ways to structure the program than that given above, this sample code has been written in a style to make it easy to follow. The program would be run in the way described above. That is, it would be opened from within EViews, and then the Run button would be pressed and the mode of execution (Verbose or Quiet) chosen.

A first point to note is that comment lines are denoted by a ′ symbol in EViews. The first line of code, 'WORKFILE DF_CV U 50000' will set up a new EViews workfile called DF_CV.WK1, which will be undated (U) and will contain series of length 50,000. This step is required for EViews to have a place to put the output series since no other workfile will be opened by this program! In situations where the program requires an already existing workfile containing data to be opened, this line would not be necessary since any new results and objects created would be appended to the original workfile. RNDSEED 12345 sets the random number seed that will be used to start the random draws.

'SERIES T1' creates a new series T1 that will be filled with NA elements. The series T1, T2 and T3, will hold the Dickey–Fuller test statistics for each replication, for the three cases (no constant or trend, constant but no trend, constant and trend, respectively). 'SCALAR K1' sets up a scalar (single number) K1. K1, . . . , K9 will be used to hold the 1%, 5% and 10% critical values for each of the three cases. !NREPS=50000 and !NOBS=1000 set the number of replications that will be used to 50,000 and the number of observations to be used in each time series to 1,000. The exclamation marks enable the scalars to be used without previously having to define them using the SCALAR instruction. Of course, these values can be changed as desired. Loops in EViews are defined as FOR at the start and NEXT at the end, in a similar way to visual basic code. Thus FOR !REPC=1 TO !NREPS starts the main replications loop, which will run from 1 to NREPS.

```
SMPL @FIRST @FIRST
SERIES Y1=0
```

The two lines above set the first observation of a new series Y1 to zero (so @FIRST is EViews method of denoting the first observation in the series, and the final observation is denoted by, you guessed it, @LAST). Then

```
SMPL @FIRST+1 !NOBS+200
SERIES Y1=Y1(−1)+NRND
SERIES DY1=Y1-Y1(−1)
```

will set the sample to run from observation 2 to observation !NOBS+200 (1200). This enables the program to generate 200 additional startup observations. It is very easy in EViews to construct a series following a random walk process, and this is done by the second of the above three lines. The current value of Y1 is set to the previous value plus a standard normal random draw (NRND). In EViews, draws can be taken from a wide array of distributions (see the User Guide). SERIES DY1 . . . creates a new series called DY1 that contains the first difference of Y.

```
SMPL @FIRST+200 !NOBS+200
EQUATION EQ1.LS DY1 Y1(−1)
```

The first of the two lines above sets the sample to run from observation 201 to observation 1200, thus dropping the 200 startup observations. The following line actually conducts an OLS estimation ('.LS'), in the process creating an equation object called EQ1. The dependent variable is DY1 and the independent variable is the lagged value of Y, Y(−1).

Following the equation estimation, several new quantities will have been created. These quantities are denoted by a '@' in EViews. So the line 'T1(!REPC)=@TSTATS(1)' will take the *t*-ratio of the coefficient on the first (and in this case only) independent variable, and will place it in the !REPC row of the series T1. Similarly, the *t*-ratios on the lagged value of Y will be placed in T2 and T3 for the regressions with constant and constant and trend respectively. Finally, NEXT will finish the replications loop and SMPL @FIRST !NREPS will set the sample to run from 1 to 50000, and the 1%, 5%, and 10% critical values for the no constant or trend case will then be found in K1, K2 and K3. The '@QUANTILE(T1,0.01)' instruction will take the 1% quantile from the series T1, which avoids sorting the series.

The critical value obtained by running the above instructions, which are virtually identical to those found in the statistical tables at the end of this book, are (to two decimal places)

	1%	5%	10%
No constant or trend	−2.58	−1.95	−1.63
Constant but no trend	−3.45	−2.85	−2.56
Constant and trend	−3.93	−3.41	−3.43

This is to be expected, for the use of 50,000 replications should ensure that an approximation to the asymptotic behaviour is obtained. For example, the 5% critical value for a test regression with no constant or trend and 500 observations is −1.945 in this simulation, and −1.95 in Fuller (1976). Although the Dickey–Fuller simulation was unnecessary in the sense that the critical values for the resulting test statistics are already well known and documented, a very similar procedure could be adopted for a variety of problems. For example, a similar approach could be used for constructing critical values or for evaluating the performance of statistical tests in various situations.

12.8 An example of how to simulate the price of a financial option

A simple example of how to use a Monte Carlo study for obtaining a price for a financial option is shown below. Although the option used for illustration here is just a plain vanilla European call option which could be valued analytically using the standard Black–Scholes (1973) formula, again, the method is sufficiently general that only relatively minor modifications would be required to value more complex options. Boyle (1977) gives an excellent and highly readable introduction to the pricing of financial options using Monte Carlo.

The steps involved are shown in box 12.5.

12.8.1 *Simulating the price of a financial option using a fat-tailed underlying process*

A fairly limiting and unrealistic assumption in the above methodology for pricing options is that the underlying asset returns are normally distributed, whereas in practice, it is well know that asset returns are fat-tailed. There are several ways to remove this assumption. First, one could employ draws from a fat-tailed distribution, such as a Student's t, in step

Box 12.5 Simulating the price of an Asian option

(1) *Specify a data generating process for the underlying asset*. A random walk with drift model is usually assumed. Specify also the assumed size of the drift component and the assumed size of the volatility parameter. Specify also a strike price K, and a time to maturity, T.
(2) Draw a series of length T, the required number of observations for the life of the option, from a normal distribution. This will be the *error series*, so that $\varepsilon_t \sim \text{N}(0, 1)$.
(3) Form a series of observations of length T on the *underlying asset*.
(4) *Observe the price of the underlying asset at maturity observation* T. For a call option, if the value of the underlying asset on maturity date, $P_T \leq K$, the option expires worthless for this replication. If the value of the underlying asset on maturity date, $P_T > K$, the option expires in the money, and has value on that date equal to $P_T - K$, which should be discounted back to the present day using the risk-free rate. Use of the risk-free rate relies upon risk-neutrality arguments (see Duffie, 1996).
(5) Repeat steps 1 to 4 a total of N times, and take the average value of the option over the N replications. This average will be the *price of the option*.

Box 12.6 Generating draws from a GARCH process

(1) Draw a series of length T, the required number of observations for the life of the option, from a normal distribution. This will be the error series, so that $\varepsilon_t \sim \mathrm{N}(0, 1)$.
(2) Recall that one way of expressing a GARCH model is

$$r_t = \mu + u_t \qquad u_t = \varepsilon_t \sigma_t \qquad \varepsilon_t \sim \mathrm{N}(0, 1) \tag{12.17}$$

$$\sigma_t^2 = \alpha_0 + \alpha_1 u_{t-1}^2 + \beta \sigma_{t-1}^2 \tag{12.18}$$

A series of ε_t, have been constructed and it is necessary to specify initialising values y_1 and σ_1^2 and plausible parameter values for α_0, α_1, β. Assume that y_1 and σ_1^2 are set to μ and one, respectively, and the parameters are given by $\alpha_0 = 0.01$, $\alpha_1 = 0.15$, $\beta = 0.80$. The equations above can then be used to generate the model for r_t as described above.

2 above. Another method, which would generate a distribution of returns with fat tails, would be to assume that the errors and therefore the returns follow a GARCH process. To generate draws from a GARCH process, do the steps shown in box 12.6.

12.8.2 *Simulating the price of an Asian option*

An Asian option is one whose payoff depends upon the average value of the underlying asset over the averaging horizon specified in the contract. Most Asian options contracts specify that arithmetic rather than geometric averaging should be employed. Unfortunately, the arithmetic average of a unit root process with a drift is not well defined. Additionally, even if the asset prices are assumed to be log-normally distributed, the arithmetic average of them will not be. Consequently, a closed-form analytical expression for the value of an Asian option has yet to be developed. Thus, the pricing of Asian options represents a natural application for simulations methods. Determining the value of an Asian option is achieved in almost exactly the same way as for a vanilla call or put. The simulation is conducted identically, and the only difference occurs in the very last step where the value of the payoff at the date of expiry is determined.

12.8.3 *Pricing Asian options using EViews*

A sample of EViews code for determining the value of an Asian option is given below. The example is in the context of an arithmetic Asian option on the FTSE 100, and two simulations will be undertaken with different strike prices (one that is out of the money forward and one that is in the money forward). In each case, the life of the option is 6 months, with daily averaging commencing immediately, and the option value is given for both calls and puts in terms of index points. The parameters are given as follows, with dividend yield and risk-free rates expressed as percentages:

Simulation 1: strike=6500, risk-free=6.24, dividend yield=2.42, 'today's' FTSE=6289.70, forward price=6405.35, implied volatility=26.52
Simulation 2: strike=5500, risk-free=6.24, dividend yield=2.42, 'today's' FTSE=6289.70, forward price=6405.35, implied volatility=34.33

Any other programming language or statistical package would be equally applicable, since all that is required is a Gaussian random number generator, the ability to store in arrays and to loop. Since no actual estimation is performed, differences between packages are likely to be negligible. All experiments are based on 25,000 replications and their antithetic variates (total: 50,000 sets of draws) to reduce Monte Carlo sampling error.

Some sample code for pricing an ASIAN option for Normally distributed errors using EViews is given as follows:

```
'NEW WORKFILE CREATED CALLED ASIAN_P, UNDATED
'WITH 50000 OBSERVATIONS
WORKFILE ASIAN_P U 50000
RNDSEED 12345
!N=125
!TTM=0.5
!NREPS=50000
!IV=0.28
!RF=0.0624
!DY=0.0242
!DT=!TTM / !N
!DRIFT=(!RF-!DY-(!IV^2/2.0))*!DT
!VSQRDT=!IV*(!DT^0.5)
!K=5500
!S0=6289.7
SERIES APVAL
SERIES ACVAL
SERIES SPOT
SCALAR AV
SCALAR CALLPRICE
SCALAR PUTPRICE
SERIES RANDS
'GENERATES THE DATA
FOR !REPC=1 TO !NREPS STEP 2
RANDS=NRND
```

```
SERIES SPOT=0
SMPL @FIRST @FIRST
SPOT(1)=!S0*EXP(!DRIFT+!VSQRDT*RANDS(1))
SMPL 2 !N
SPOT=SPOT(−1)*EXP(!DRIFT+!VSQRDT*RANDS(!N))
'COMPUTE THE DAILY AVERAGE
SMPL @FIRST !N
AV=@MEAN(SPOT)
IF AV>!K THEN
   ACVAL(!REPC)=(AV-!K)*EXP(-!RF*!TTM)
ELSE
   ACVAL(!REPC)=0
ENDIF
IF AV<!K THEN
   APVAL(!REPC)=(!K-AV)*EXP(-!RF*!TTM)
ELSE
   APVAL(!REPC)=0
ENDIF
RANDS=-RANDS
SERIES SPOT=0
SMPL @FIRST @FIRST
SPOT(1)=!S0*EXP(!DRIFT+!VSQRDT*RANDS(1))
SMPL 2 !N
SPOT=SPOT(−1)*EXP(!DRIFT+!VSQRDT*RANDS(!N))
'COMPUTE THE DAILY AVERAGE
SMPL @FIRST !N
AV=@MEAN(SPOT)
IF AV>!K THEN
   ACVAL(!REPC+1)=(AV-!K)*EXP(-!RF*!TTM)
ELSE
   ACVAL(!REPC+1)=0
ENDIF
IF AV<!K THEN
   APVAL(!REPC+1)=(!K-AV)*EXP(-!RF*!TTM)
ELSE
   APVAL(!REPC+1)=0
ENDIF
NEXT
SMPL @FIRST !NREPS
CALLPRICE=@MEAN(ACVAL)
PUTPRICE=@MEAN(APVAL)
```

Many parts of the program above use identical instructions to those given for the DF critical value simulation, and so annotation will now focus on the construction of the program and on previously unseen commands. The first block of commands set up a new workfile called ‘ASIAN_P’ that will hold all of the objects and output. Then the following lines specify the parameters for the simulation of the path of the price of the underlying asset (the drift, the implied volatility, etc.).

‘!=DT=!TTM/!N’ splits the time to maturity (0.5 years) into N discrete time periods. Since daily averaging is required, it is easiest to set N = 125 (the approximate number of trading days in half a year), so that each time period DT represents one day. The model assumes that the log of the underlying asset price follows a geometric Brownian motion, which could be given by

$$S + dS = S\exp\left[\left(rf - dy - \frac{1}{2}\sigma^2\right)dt + \sigma dz\right] \tag{12.19}$$

where *dz* is a standard Wiener process. Further details of this continuous time representation of the movement of the underlying asset over time are beyond the scope of this book. A treatment of this and many other useful option pricing formulae and computer code are given in Haug (1998). The discrete time approximation to this can be written

$$S_t = S_{t-1}\exp\left[\left(rf - dy - \frac{1}{2}\sigma^2\right)dt + \sigma\sqrt{dt}\,u_t\right] \tag{12.20}$$

The following instructions set up the arrays for the underlying spot price (called ‘SPOT’), and for the discounted values of the put (‘APVAL’) and call (‘ACVAL’). Note that by default, arrays of the length given by the ‘workfile’ definition statement (50000) will be created.

The command ‘FOR !REPC=1 TO !NREPS DO REPC=1, NREPS,2’ starts the main do loop for the simulation, looping up to the number of replications, in steps of 2. The loop ends at ‘END DO REPC’. Steps of 2 are used because antithetic variates are also used for each replication, which will create another simulated path for the underlying asset prices and option value.

The random N(0,1) draws are made, which are then constructed into a series of future prices of the underlying asset for the next 125 days. ‘AV=@MEAN(SPOT)’ will compute the average price of the underlying over the lifetime of the option (125 days). The following two statements construct the terminal payoffs for the call and the put options respectively.

For the call, 'ACVAL' is set to the average underlying price less the strike price if the average is greater than the strike (i.e. if the option expires in the money), and zero otherwise. Vice versa for the put. The payoff at expiry is discounted back to the present using the risk-free rate, and placed in the REPC row of the 'ACVAL' or 'APVAL' array for the calls and puts, respectively.

The process then repeats using the antithetic variates, constructed using 'RANDS = -RANDS'. The call and put present values for these paths are put in the even rows of 'ACVAL' and 'APVAL'.

This completes one cycle of the REPC loop, which starts again with REPC=3, then 5, 7, 9, . . . , 49999. The result will be 2 arrays 'ACVAL' and 'APVAL', which will contain 50,000 rows comprising the present value of the call and put option for each simulated path. The option prices would then simply be given by the averages over the 50,000 replications.

Note that both call values and put values can be calculated easily from a given simulation, since the most computationally expensive step is in deriving the path of simulated prices for the underlying asset. The results are given in table 10.1, along with the values derived from an analytical approximation to the option price, derived by Levy, and estimated using VBA code in Haug (1998, pp. 97–100).

The main difference between the way that the simulation is conducted here and the method used for EViews simulation of the Dickey–Fuller critical values is that here, the random numbers are generated by opening a new series called 'RANDS' and filling it with the random number draws. The reason that this must be done is so that the negatives of the elements of RANDS can later be taken to form the antithetic variates. Finally, for each replication, the IF clause will set out of the money call prices (where K>AV) and out of the money put prices (K<AV) to zero. Then the call and put prices for each replication are discounted back to the present using the risk-free rate, and outside the replications loop, the options prices are the averages of these discounted prices across the 50,000 replications.

The workfile 'ASIAN_P' will contain quite a few objects by the end of the simulation, including the scalars CALLPRICE and PUTPRICE, which will be the call and put prices. Also, the series ACVAL and APVAL will contain the current value of the option for each of the 50,000 simulated paths. Having the whole series across all replications can be useful for constructing standard errors, and for checking that the program appears to have been working correctly.

Applying the instructions above (with K = 5500, and implied volatility at 28%) gives simulated call and put prices as given in the following table.

Strike = 6500, IV = 26.52		Strike = 5500, IV = 34.33	
CALL	Price	*CALL*	Price
Analytical Approximation	203.45	Analytical Approximation	888.55
Monte Carlo Normal	204.22	Monte Carlo Normal	885.29
PUT	Price	*PUT*	Price
Analytical Approximation	348.7	Analytical Approximation	64.52
Monte Carlo Normal	349.43	Monte Carlo Normal	61.52

In both cases, the simulated options prices are quite close to the analytical approximations, although the Monte Carlo seems to overvalue the out-of-the-money call and to undervalue the out-of-the-money put. Some of the errors in the simulated prices relative to the analytical approximation may result from the use of a discrete-time averaging process using only 125 points.

12.9 An example of bootstrapping to calculate capital risk requirements

12.9.1 *Financial motivation*

Risk management modelling has, in this author's opinion, been one of the most rapidly developing areas of application of econometric techniques over the past decade or so. One of the most popular approaches to risk measurement is by calculating what is known as an institution's 'value-at-risk', denoted VaR. Broadly speaking, value-at-risk is an estimation of the *probability of likely losses which could arise from changes in market prices*. More precisely, it is defined as the money-loss of a portfolio that is expected to occur over a pre-determined horizon and with a pre-determined degree of confidence. The roots of VaR's popularity stem from the simplicity of its calculation, its ease of interpretation and from the fact that VaR can be suitably aggregated across an entire firm to produce a single number which broadly encompasses the risk of the positions of the firm as a whole. The value-at-risk estimate is also often known as the position risk requirement or minimum capital risk requirement (MCRR); the three terms will be used interchangeably in the exposition below.

There are various methods available for calculating value at risk, including the 'delta-normal' method; historical simulation, involving the estimation of the quantile of returns of the portfolio; and structured Monte Carlo simulation; see Dowd (1998) or Jorion (2006) for thorough introductions to value-at-risk.

The *Monte Carlo* approach involves two steps. First, a data generating process is specified for the underlying assets in the portfolio. Second, possible future paths are simulated for those assets over given horizons, and the value of the portfolio at the end of the period is examined. Thus the returns for each simulated path are obtained, and from this distribution across the Monte Carlo replications, the VaR as a percentage of the initial value of the portfolio can be measured as the first or fifth percentile.

The Monte Carlo method is clearly a very powerful and flexible method for generating VaR estimates, since any stochastic process for the underlying assets can be specified. The effect of increasing variances or correlations, etc. can easily be incorporated into the simulation design. However, there are at least two drawbacks with the use of Monte Carlo simulation for estimating VaR. First, for a large portfolio, the computational time required to compute the VaR may be excessively great. Second, and more fundamentally, the calculated VaR may be inaccurate if the stochastic process that has been assumed for the underlying asset is inappropriate. In particular, asset prices are often assumed to follow a random walk or a random walk with drift, where the driving disturbances are random draws from a normal distribution. Since it is well known that asset returns are fat-tailed, the use of Gaussian draws in the simulation is likely to lead to a systematic underestimate of the VaR, as extremely large positive or negative returns are more likely in practice than would arise under a normal distribution. Of course, the normal random draws could be replaced by draws from a *t*-distribution, or the returns could be assumed to follow a GARCH process, both of which would generate an unconditional distribution of returns with fat tails. However, there is still some concern as to whether the distribution assumed in designing the simulations framework is really appropriate.

An alternative approach, that could potentially overcome this criticism, would be to use bootstrapping rather than Monte Carlo simulation. In this context, the future simulated prices are generated using random draws with replacement from the actual returns themselves, rather than artificially generating the disturbances from an assumed distribution. Such an approach is used in calculating MCRRs by Hsieh (1993) and by Brooks, Clare and Persand (2000). The methodology proposed by Hsieh will now be examined.

Hsieh (1993) employs daily log returns on foreign currency (against the US dollar) futures series from 22 February 1985 until 9 March 1990 (1,275 observations) for the British pound (denoted BP), the German mark (GM), the Japanese yen (JY) and the Swiss franc (SF). The first stage in setting up the bootstrapping framework is to form a model that fits the data and adequately describes its features. Hsieh employs the BDS test (discussed briefly in chapter 8) to determine an appropriate class of models. An application of the test to the raw returns data shows that the data are not random, and that there is some structure in the data. The dependence in the series, shown in the rejection of randomness by the test implies that there is either:

- a linear relationship between y_t and $y_{t-1}, y_{t-2}, \ldots$ or
- a non-linear relationship between y_t and $y_{t-1}, y_{t-2}, \ldots$

The Box–Pierce Q test is applied to test for both, on the returns for the former, and on the squared or absolute values of the returns for the latter. The results of this test are not shown but effectively rule out the possibility of linear dependence (so that, for example, an ARMA model would not be appropriate for the returns), but there appears to be evidence of non-linear dependence in the series. Therefore, a second question, is whether the non-linearity is in-mean or in-variance (see chapter 8 for elucidation). Hsieh uses a bicorrelation test to show that there is no evidence for non-linearity in-mean. Therefore, the most appropriate class of models for the returns series is a model which has time-varying (conditional) variances. Hsieh employs two types of model: EGARCH and autoregressive volatility (ARV) models. The coefficient estimates for the EGARCH model are reported in table 12.1.

Several features of the EGARCH estimates are worth noting. First, as one may anticipate for a set of currency futures returns, the asymmetry terms (i.e. the estimated values of γ) are not significant for any of the four series. The high estimated values of β suggest a high degree of persistence in volatility in all cases except the Japanese yen. Brooks, Clare and Persand (2000) suggest that such persistence may be excessive in the sense that the volatility implied by the estimated conditional variance is too persistent to reproduce the profile of the volatility of the actual returns series. Such excessive volatility persistence could lead to an overestimate of the VaR. Leaving this issue aside, Hsieh continues to evaluate the effectiveness of the EGARCH models in capturing all of the non-linear dependence in the data. This is achieved by reapplying the BDS test to the standardised residuals, constructed by taking the residuals from the estimated models, and dividing them by their respective conditional standard deviations. If

Table 12.1 EGARCH estimates for currency futures returns

$$x_t = \mu + \sigma_t \eta_t$$
$$\eta_t \sim \mathrm{N}(0,1)$$
$$\log \sigma_t^2 = \alpha + \beta \log \sigma_{t-1}^2 + \phi(|\eta_{t-1}| - (2/\pi)^{1/2}) + \gamma \eta_{t-1}$$

Coefficient	BP	*DM*	JY	SF
μ	0.000319	0.000377	0.000232	0.000239
	(0.000208)	(0.000214)	(0.000189)	(0.000235)
α	−0.688127	−1.072229	−4.438289	−0.993241
	(0.030088)	(0.041828)	(0.756704)	(0.032479)
β	0.928780	0.889511	0.550707	0.895527
	(0.002995)	(0.004386)	(0.075851)	(0.003508)
ϕ	0.135854	0.187005	0.282167	0.157669
	(0.019961)	(0.028388)	(0.093357)	(0.024013)
γ	−0.110718	0.084173	0.313274	0.129035
	(0.177458)	(0.147279)	(0.201531)	(0.166507)

Notes: Standard errors in parentheses.
Source: Hsieh (1993). Reprinted with the permission of School of Business Administration, University of Washington.

the model has captured all of the important features of the data, the standardised residual series should be completely random. It is observed that the EGARCH model cannot capture all of the non-linear dependence in the mark or franc series.

A second approach to modelling volatility is derived from a high/low volatility estimator. A daily volatility series is thus constructed using a re-scaled estimate of the range over the trading day

$$\sigma_{P,t} = (0.361 \times 1440/M)^{1/2}\, log(\, High_t / Low_t) \tag{12.21}$$

where $High_t$ and Low_t are the highest and lowest transacted prices on day t and M is the number of trading minutes during the day. The volatility series, $\sigma_{P,t}$ can now be modelled as any other series. A natural model to propose, given the dependence (or persistence) in volatility over time, is an autoregressive model in the volatility. The formulation used for the price series is known as an autoregressive volatility (ARV) model

$$x_t = \sigma_{P,t} u_t \tag{12.22}$$
$$\ln \sigma_{P,t} = \alpha + \sum_i \beta_i \ln \sigma_{P,t-i} + \nu_t \tag{12.23}$$

where ν_t is an error term. The appropriate lag length for the ARV model is determined using Schwarz's information criterion, which suggests that

Table 12.2 Autoregressive volatility estimates for currency futures returns

$$x_t = \sigma_{P,t}\, u_t$$
$$\ln \sigma_{P,t} = \alpha + \sum_i \beta_i \ln \sigma_{P,t-i} + \nu_t$$

Coefficient	BP	DM	JY	SF
α	−1.037	−1.139	−1.874	−1.219
	(0.171)	(0.187)	(0.199)	(0.193)
β_1	0.192	0.153	0.208	0.115
	(0.028)	(0.028)	(0.028)	(0.028)
β_2	0.134	0.111	0.137	0.106
	(0.029)	(0.028)	(0.028)	(0.028)
β_3	0.062	0.052	0.058	0.068
	(0.029)	(0.028)	(0.029)	(0.028)
β_4	0.069	0.092	0.109	0.091
	(0.029)	(0.028)	(0.028)	(0.028)
β_5	0.137	0.091	0.112	0.118
	(0.028)	(0.028)	(0.028)	(0.028)
β_6	0.027	0.072		0.074
	(0.029)	(0.028)		(0.028)
β_7	0.073	0.110		0.086
	(0.028)	(0.028)		(0.028)
β_8	0.088	0.079		0.078
	(0.028)	(0.028)		(0.028)
$\bar{R}^2$	0.274	0.227	0.170	0.193

Source: Hsieh (1993). Reprinted with the permission of School of Business Administration, University of Washington.

8, 8, 5 and 8 lags should be used for the pound, mark, yen and franc series, respectively. The coefficient estimates for the ARV models are given in table 12.2.

The degrees of persistence for each exchange rate series implied by the ARV estimates is given by the sums of the β coefficients, which are 0.78, 0.76, 0.62, 0.74, respectively. These figures are high, although less so than under the EGARCH formulation. The standardised residuals from this model are given by $x_t/\hat{\sigma}_{P,t}$, where $\hat{\sigma}_{P,t}$ are the fitted values of volatility. An application of the BDS test to these standardised residuals shows no evidence of further structure apart from in the Swiss franc case, where the test statistics are marginally significant. Thus, since these standardised residuals are iid, it is valid to sample from them using the bootstrap technique.

To summarise, it is concluded that both the EGARCH and ARV models present reasonable descriptions of the futures returns series, which are then employed in conjunction with the bootstrap to estimate the value at risk estimates. This is achieved by simulating the future values of the futures price series, using the parameter estimates from the two models, and using disturbances obtained by sampling with replacement from the standardised residuals ($\hat{\eta}_t/\hat{h}_t^{1/2}$) for the EGARCH model and from u_t and ν_t for ARV models. In this way, 10,000 possible future paths of the series are simulated (i.e. 10,000 replications are used), and in each case, the maximum drawdown (loss) can be calculated over a given holding period by

$$Q = (P_0 - P_1) \times \textit{number of contracts} \tag{12.24}$$

where P_0 is the initial value of the position, and P_1 is the lowest simulated price (for a long position) or highest simulated price (for a short position) over the holding period. The maximum loss is calculated assuming holding periods of 1, 5, 10, 15, 20, 25, 30, 60, 90 and 180 days. It is assumed that the futures position is opened on the final day of the sample used to estimate the models, 9 March 1990.

The 90th percentile of these 10,000 maximum losses can be taken to obtain a figure for the amount of capital required to cover losses on 90% of days. It is important for firms to consider the maximum daily losses arising from their futures positions, since firms will be required to post additional funds to their margin accounts to cover such losses. If funds are not made available to the margin account, the firm is likely to have to liquidate its futures position, thus destroying any hedging effects that the firm required from the futures contracts in the first place.

However, Hsieh (1993) uses a slightly different approach to the final stage, which is as follows. Assuming (without loss of generality) that the number of contracts held is 1, the following can be written for a long position

$$\frac{Q}{x_0} = \left(1 - \frac{x_1}{x_0}\right) \tag{12.25}$$

or

$$\frac{Q}{x_0} = \left(\frac{x_1}{x_0} - 1\right) \tag{12.26}$$

for a short position. x_1 is defined as the minimum price for a long position (or the maximum price for a short position) over the horizon that the position is held. In either case, since x_0 is a constant, the distribution of Q will depend on the distribution of x_1. Hsieh (1993) assumes that prices are lognormally distributed, i.e. that the logs of the ratios of the prices,

$$\ln\left(\frac{x_1}{x_0}\right)$$

are normally distributed. This being the case, an alternative estimate of the fifth percentile of the distribution of returns can be obtained by taking the relevant critical value from the normal statistical tables, multiplying it by the standard deviation and adding it to the mean of the distribution.

The MCRRs estimated using the ARV and EGARCH models are compared with those estimated by bootstrapping from the price changes themselves, termed the 'unconditional density model'. The estimated MCRRs are given in table 12.3.

The entries in table 12.3 refer to the amount of capital required to cover 90% of expected losses, as percentages of the initial values of the positions. For example, according to the EGARCH model, approximately 14% of the initial value of a long position should be held in the case of the yen to cover 90% of expected losses for a 180-day horizon. The results contain several interesting features. First, the MCRRs derived from bootstrapping the price changes themselves (the 'unconditional approach') are in most cases higher than those generated from the other two methods, especially at short investment horizons. This is argued to have occurred owing to the fact that the level of volatility at the start of the MCRR calculation period was low relative to its historical level. Therefore, the conditional estimation methods (EGARCH and ARV) will initially forecast volatility to be lower than the historical average. As the holding period increases from 1 towards 180 days, the MCRR estimates from the ARV model converge upon those of the unconditional densities. On the other hand, those of the EGARCH model do not converge, even after 180 days (in fact, in some cases, the EGARCH MCRR seems oddly to diverge from the unconditionally estimated MCRR as the horizon increases). It is thus argued that the EGARCH model may be inappropriate for MCRR estimation in this application.

It can also be observed that the MCRRs for short positions are larger than those of comparative long positions. This could be attributed to an upward drift in the futures returns over the sample period, suggesting that on average an upwards move in the futures price was slightly more likely than a fall.

A further step in the analysis, which Hsieh did not conduct, but which is shown in Brooks, Clare and Persand (2000), is to evaluate the performance of the MCRR estimates in an out-of-sample period. Such an exercise would evaluate the models by assuming that the MCRR estimated from

Table 12.3 Minimum capital risk requirements for currency futures as a percentage of the initial value of the position

		Long position			Short position		
	No. of days	AR	Unconditional density	EGARCH	AR	Unconditional density	EGARCH
BP	1	0.73	0.91	0.93	0.80	0.98	1.05
	5	1.90	2.30	2.61	2.18	2.76	3.00
	10	2.83	3.27	4.19	3.38	4.22	4.88
	15	3.54	3.94	5.72	4.45	5.48	6.67
	20	4.10	4.61	6.96	5.24	6.33	8.43
	25	4.59	5.15	8.25	6.20	7.36	10.46
	30	5.02	5.58	9.08	7.11	8.33	12.06
	60	7.24	7.44	14.50	11.64	12.87	20.71
	90	8.74	8.70	17.91	15.45	16.90	28.03
	180	11.38	10.67	24.25	25.81	27.36	48.02
DM	1	0.72	0.87	0.83	0.89	1.00	0.95
	5	1.89	2.18	2.34	2.23	2.70	2.91
	10	2.77	3.14	3.93	3.40	4.12	5.03
	15	3.52	3.86	5.37	4.36	5.30	6.92
	20	4.05	4.45	6.54	5.19	6.14	8.91
	25	4.55	4.90	7.86	6.14	7.21	10.69
	30	4.93	5.37	8.75	7.02	7.88	12.36
	60	7.16	7.24	13.14	11.36	12.38	20.86
	90	8.87	8.39	16.06	14.68	16.16	27.75
	180	11.38	10.35	21.69	24.25	26.25	45.68
JY	1	0.56	0.74	0.72	0.68	0.87	0.86
	5	1.61	1.99	2.22	1.92	2.36	2.73
	10	2.59	2.82	3.46	3.06	3.53	4.41
	15	3.30	3.46	4.37	4.11	4.60	5.79
	20	3.95	4.10	5.09	5.13	5.45	6.77
	25	4.42	4.58	5.78	5.91	6.30	7.98
	30	4.95	4.92	6.34	6.58	6.85	8.81
	60	6.99	6.84	8.72	10.53	10.74	13.58
	90	8.43	8.00	10.51	13.61	14.00	17.63
	180	10.97	10.27	13.99	21.86	22.21	27.39
SF	1	0.82	0.97	0.89	0.93	1.12	0.98
	5	1.99	2.51	2.48	2.23	2.93	2.98
	10	2.87	3.60	4.12	3.37	4.53	5.09
	15	3.67	4.35	5.60	4.22	5.67	7.03
	20	4.24	5.10	6.82	5.09	6.69	8.86
	25	4.81	5.65	8.12	5.90	7.77	10.93
	30	5.23	6.20	9.12	6.70	8.47	12.50
	60	7.69	8.41	13.73	10.55	13.10	21.27
	90	9.23	9.93	16.89	13.60	17.06	27.80
	180	12.18	12.57	22.92	21.72	27.45	45.47

Source: Hsieh (1993). Reprinted with the permission of School of Business Administration, University of Washington.

the model had been employed, and by tracking the change in the value of the position over time. If the MCRR is adequate, the 90% nominal estimate should be sufficient to cover losses on 90% of out-of-sample testing days. Any day where the MCRR is insufficient to cover losses is termed an ‘exceedence’ or an ‘exception’. A model that leads to more than 10% exceptions for a nominal 90% coverage is deemed unacceptable on the grounds that on average, the MCRR was insufficient. Equally, a model that leads to considerably less than the expected 10% exceptions would also be deemed unacceptable on the grounds that the MCRR has been set at an inappropriately high level, leading capital to be unnecessarily tied up in a liquid and unprofitable form. Brooks, Clare and Persand (2000) observe, as Hsieh’s results forewarn, that the MCRR estimates from GARCH-type models are too high, leading to considerably fewer exceedences than the nominal proportion.

12.9.2 *VaR estimation using bootstrapping in EViews*

Following the discussion above concerning the Hsieh (1993) and Brooks, Clare and Persand (2000) approaches to calculating minimum capital risk requirements, the following EViews code can be used to calculate the MCRR for a 10-day holding period (the length that regulators require banks to employ) using daily S&P500 data, which is found in the file ‘sp500.wf1’. The code is presented, followed by an annotated copy of some of the key lines.

```
'THIS PROGRAM APPLIES THE BOOTSTRAP TO THE
'CALCULATION OF
'MCRR FOR A 10-DAY HORIZON PERIOD
'LOAD WORKFILE
LOAD ‘‘D:\CHRIS\BOOK\SP500.WF1”
RNDSEED 12345
!NREPS=10000
SERIES RT
SERIES U
SERIES H
SERIES MIN
SERIES MAX
SERIES L1
SERIES S1
SCALAR MCRRL
SCALAR MCRRS
RT=LOG(SP500/SP500(−1))
```

```
EQUATION EQ1.ARCH(M=100,C=1E-5) RT C
EQ1.MAKEGARCH H
EXPAND 1 10000
SERIES HSQ=H^0.5
SERIES RESI=RT-@COEFS(1)
SERIES SRES=RESI/HSQ
EQ1.FORECAST RTF YSE HF
'BOOTSTRAP LOOP
FOR !Z=1 TO !NREPS
   SMPL 3 2610
   GROUP G1 SRES
   G1.RESAMPLE
   SMPL 2611 2620
   RT=@COEFS(1)+@SQRT(HF(-2610))*SRES_B(−10)
   SP500=SP500(−1)*EXP(RT)
MIN(!Z)=@MIN(SP500)
MAX(!Z)=@MAX(SP500)
NEXT
SMPL 1 10000
'LONG POSITION
L1=LOG(MIN/1138.73)
MCRRL=1-(EXP((−1.645*@STDEV(L1))+@MEAN(L1)))
'SHORT POSITION
S1=LOG(MAX/1138.73)
MCRRS=(EXP((1.645*@STDEV(S1))+@MEAN(S1)))−1
```

Again, annotation of the EViews code above will concentrate on commands that have not been discussed previously. The 'SERIES . . .' and 'SCALAR . . .' statements set up the arrays that will hold the series and the scalars (i.e. single numbers) respectively.

Then 'EQUATION EQ1.ARCH(M=100,C=1E-5) RT C' estimates an ARCH model, denoting the equation object created by 'EQ1', and allowing the process to perform up to 100 iterations with a convergence criterion of 10^{-5}, with the dependent variable RT (which is the returns series) and the conditional mean equation containing a constant only. The line 'EQ1.MAKEGARCH H' will generate a series of fitted conditional variance values, denoted by H. The 'EXPAND 1 10000' instruction will increase the size of the arrays in the workfile to 10000 from the original length of the S&P series (2,610 observations).

The three lines SERIES HSQ=H^0.5, SERIES RESI=RT-@COEFS(1) and SERIES SRES=RESI/HSQ will construct a set of standardised residuals.

The next step is to forecast the conditional variances for 10 observations 2611 to 2620 using the command 'EQ1.FORECAST RTF YSE HF', which will construct forecasts of the conditional mean (placed into RTF), the conditional standard deviation (YSE) and the conditional variance (HF), respectively.

Next follows the core of the program, which is the bootstrap loop, Z. The number of replications '!NREPS' has been defined as 10,000. The instructions GROUP G1 SRES and G1.RESAMPLE construct a group (in this case, containing only one element SRES), which is then resampled. The re-sampled series is then placed in SRES_B. The future paths of the series over the 10-day holding period are then constructed, and the maximum and minimum price achieved over that period (observations 2611 to 2620) are saved in the arrays MAX and MIN, respectively. Finally, NEXT finishes the bootstrapping loop.

The following SMPL instruction is necessary to reset the sample period used to cover all observation numbers from 1 to 10,000 (i.e. to incorporate all of the 10,000 bootstrap replications). By default, if this statement was not included, EViews would have continued to use the most recent sample statement, conducting analysis using only observations 2611 to 2620:

```
SMPL110000
```

The following block of two commands generates the MCRR for the long position. The first stage is to construct the log returns for the maximum loss over the 10-day holding period. Notice that the command will automatically do this calculation for every element of the 'MIN' array – i.e. for all 10,000 replications. In order to use information from all of the replications, and under the assumption that the L1 statistic is normally distributed across the replications, the MCRR can be calculated using the command given (rather than using the fifth percentile of the empirical distribution). This works as follows. Assuming that $\ln(\frac{x_1}{x_0})$ is normally distributed with some mean m and standard deviation sd, a standard normal variable can be constructed by subtracting the mean and dividing by the standard deviation

$$\frac{\ln\left(\dfrac{x_1}{x_0}\right) - m}{sd} \sim N(0, 1).$$

The 5% lower tail critical value for a standard normal is -1.645, so to find the fifth percentile

$$\frac{\ln\left(\dfrac{x_1}{x_0}\right) - m}{sd} = -1.645 \tag{12.27}$$

Rearranging (12.27)

$$\frac{x_1}{x_0} = \exp\left[-1.645sd + m\right] \tag{12.28}$$

From (12.25), (12.28) can also be written

$$\frac{Q}{\bar{x}_0} = 1 - \exp\left[-1.645sd + m\right] \tag{12.29}$$

which will give the maximum loss or draw down on a long position over the simulated 10 days. The maximum draw down for a short position will be given by

$$\frac{Q}{\bar{x}_0} = \exp\left[-1.645sd + m\right] - 1 \tag{12.30}$$

The following two lines then repeat the above procedure, but replacing the 'MIN' array with 'MAX' to calculate the MCRR for a short position: The results that would be generated by running the above program are approximately:

MCRR = 0.04035
MCRR = 0.04814

These figures represent the minimum capital risk requirement for a long and short position, respectively, as a percentage of the initial value of the position for 95% coverage over a 10-day horizon. This means that, for example, approximately 4% of the value of a long position held as liquid capital will be sufficient to cover losses on 95% of days if the position is held for 10 days. The required capital to cover 95% of losses over a 10-day holding period for a short position in the S&P500 index would be around 4.8%. This is as one would expect since the index had a positive drift over the sample period. Therefore, the index returns are not symmetric about zero as positive returns are slightly more likely than negative returns. Higher capital requirements are thus necessary for a short position since a loss is more likely than for a long position of the same magnitude.

Key concepts

The key terms to be able to define and explain from this chapter are

- simulation
- Monte Carlo sampling variability
- antithetic variates
- bootstrapping
- pseudo-random number
- control variates

Review questions

1. (a) Present two examples in finance and two in econometrics (ideally other than those listed in this chapter!) of situations where a simulation approach would be desirable. Explain in each case why simulations are useful.
   (b) Distinguish between pure simulation methods and bootstrapping. What are the relative merits of each technique? Therefore, which situations would benefit more from one technique than the other?
   (c) What are variance reduction techniques? Describe two such techniques and explain how they are used.
   (d) Why is it desirable to conduct simulations using as many replications of the experiment as possible?
   (e) How are random numbers generated by a computer?
   (f) What are the drawbacks of simulation methods relative to analytical approaches, assuming that the latter are available?
2. A researcher tells you that she thinks the properties of the Ljung–Box test (i.e. the size and power) will be adversely affected by ARCH in the data. Design a simulations experiment to test this proposition.
3. (a) Consider the following AR(1) model

$$y_t = \phi y_{t-1} + u_t \tag{12.31}$$

   Design a simulation experiment (with code for EViews) to determine the effect of increasing the value of ϕ from 0 to 1 on the distribution of the *t*-ratios.
   (b) Consider again the AR(1) model of (12.31). As stated in chapter 4, the explanatory variables in a regression model are assumed to be non-stochastic, and yet y_{t-1} is stochastic. The result is that the estimator for ϕ will be biased in small samples. Design a simulation experiment to investigate the effect of the value of ϕ and the sample size on the extent of the bias.
4. A barrier option is a path-dependent option whose payoff depends on whether the underlying asset price traverses a barrier. A knock-out call is a call option that ceases to exist when the underlying price falls below a given barrier level H. Thus the payoff is given by

$$\begin{array}{rl} \max[0, S_T - K] & \text{if } S_t > H \; \forall \, t \leq T \\ 0 & \text{if } S_t \leq H \text{ for any } t \leq T. \end{array}$$

   where S_T is the underlying price at expiry date T, and K is the exercise price. Suppose that a knock-out call is written on the FTSE 100 Index.

The current index value, $S_0 = 5000$, $K = 5100$, time to maturity $= 1$ year, $H = 4900$, $IV = 25\%$, risk-free rate $= 5\%$, dividend yield $= 2\%$.

Design a Monte Carlo simulation to determine the fair price to pay for this option. Using the same set of random draws, what is the value of an otherwise identical call without a barrier? Design computer code in EViews to test your experiment.