Option Pricing with `S+FinMetrics`

PETER FULEKY
Department of Economics
University of Washington

August 27, 2007

Contents

1 Introduction

In this guide, the reader will find a summary of basic option pricing theory[1] along with examples of option pricing functions[2] implemented in `S+FinMetrics`.

A derivative security is a financial instrument whose value is derived from the values of other underlying variables, such as the prices of traded assets. Option contracts are derivative securities that give the holder the right, but not the obligation, to engage in a future transaction on some underlying asset. Once the holder of the option decides to exercise the option, the party that sold (wrote) the option must fulfill the terms specified in the contract.

1.1 Terminology

Amount and Class of underlying asset: e.g. 100 shares of XYZ Corp.

Strike Price / Exercise Price: the price at which the underlying transaction will occur once the holder exercises the option.

Expiration / Maturity Date: the last date when the option can be exercised.

Option Price / Premium: the price the holder of the option paid for the option, that is, for the right to buy or sell a security at the strike price in the future.

Settlement Terms: for example, whether the writer must deliver the underlying asset or may transfer the equivalent cash amount.

Call Option: gives the holder the right to buy the underlying asset by the expiration date for the strike price.

Put Option: gives the holder then right to sell the underlying asset by the expiration date for the strike price.

Option Style

- **Vanilla Options**
  - **American Option:** might be exercised any day on or prior to expiration date.
  - **European Option:** might be exercised only on the expiration date.
- **Exotic Options:** more unusual styles in which the pay-off structure and exercise timing are different from those of the vanilla options.

1.2 Option Positions

The *holder* of an option is said to be in a long position, and the *writer* of an option contract is said to be in a short position. On the maturity date, the payoff to a holder of a call option is $\max(S_T - K, 0)$, and the payoff to a holder of a put is $\max(K - S_T, 0)$, where S_T is the market price of the underlying asset on the maturity date, and K is the strike price (for details see section 2).

[1]For a detailed exposition of option pricing theory the reader is referred to Hull (2005)

[2]For a collection of option pricing formulas the reader is referred to Haug (2006)

The diagrams in Figure 1 are displaying various payoffs for European call and put options with strike price $K = 100$. The `S+FinMetrics` code generating the diagrams uses the `bsmEU` function to be introduced in Example 1. (The Appendix gives the code for generating the plot.)

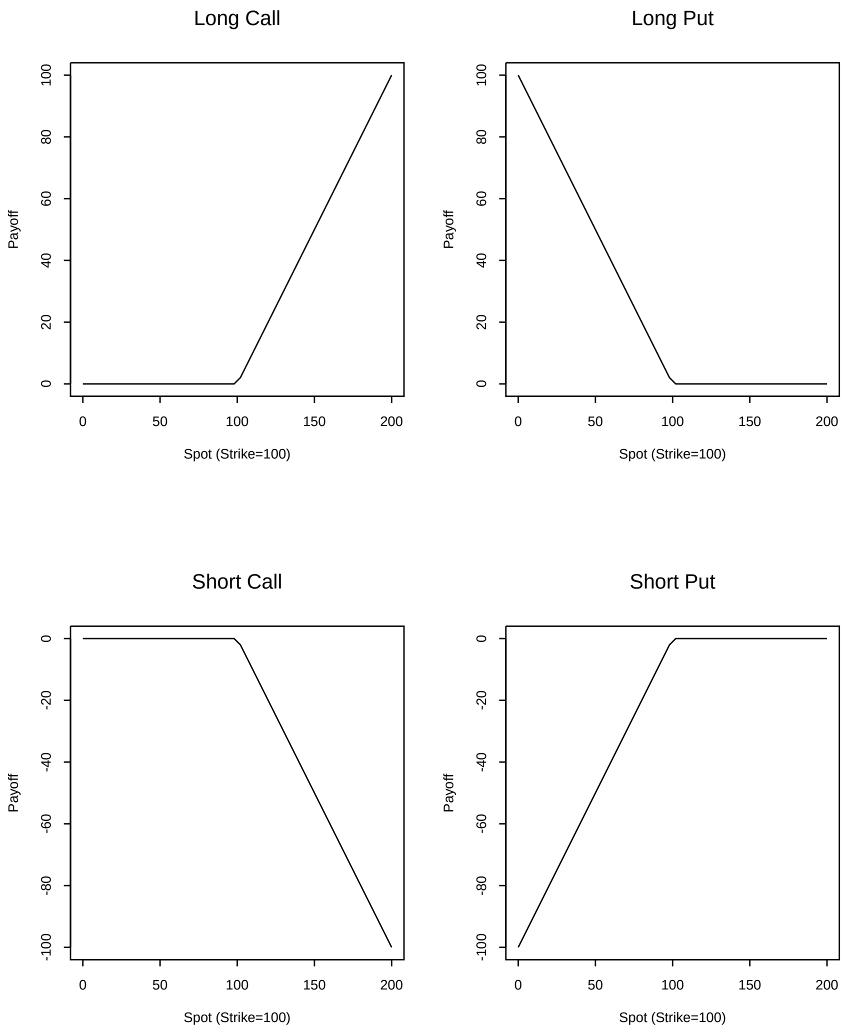

Figure 1: Payoffs as functions of underlying asset price for different option positions with strike price $K = 100$. The Appendix gives the code for generating the plot.

2 Valuation of Vanilla Options

In general, the price of vanilla options depends on the following factors:

- The current market price of the underlying asset, S_0.
- The strike price of the option, K.
- The time to expiration, T, together with any restrictions on when exercise may occur.
- An estimate of the future volatility of the underlying asset's price, σ.
- The cost of carry, b, where $b = r - q$ (interest rate minus the dividend yield or other positive cash flows). The cost of carry is the cumulative cost required to hold a position in the underlying security.

Figure 2 shows the influence of these factors on the price of vanilla call options. The diagrams are drawn for default values of $S_0 = 100,\ K = 100,\ T = 0.5,\ r = 0.05,\ b = 0.05,\ \sigma = 0.2$. If an option were exercised immediately, its value would be determined by the strike price and the price of the underlying asset. This is the *intrinsic value* of an option. For a call, it is defined as $\max(S_0 - K, 0)$, and for a put as $\max(K - S_0, 0)$.

As long as the intrinsic value is positive, the option is said to be *in-the-money.* In the absence of transaction costs, all in-the-money options will be exercised by the expiration date. If $S_0 = K$, the option is said to be *at-the-money.* Options that do not fall into either one of the previous two categories are *out-of-the-money.*

If the option price exceeds its intrinsic value, it is said to have time value: $Time\ Value = Option\ Price - Intrinsic\ Value$. More specifically, an option's time value captures the possibility that the option might increase in value due to volatility in the underlying asset. The option's time value depends on the time until the expiration date and the volatility of the underlying instrument's price. The panels in the top row of Figure 2 illustrate the price of a call option with six months to expiration. The option price increases with the spot price of the underlying asset, and decreases with the strike price. Because there are six months to expiration, the option has time value: as apparent from both top panels, an at-the-money option with $S_0 = 100,\ K = 100$ has positive price.

In the absence of discrete cash flows from the underlying security, options become more valuable as the time to maturity increases. The longer the time to expiration, the greater the probability that an out-of-money European option will end up in-the-money. If there is more time left until the expiration date, the holder of an American option has more exercise opportunities, which in turn implies higher option value. This positive relationship between time to maturity and option price is illustrated in the third panel of Figure 2.

Higher volatility of the underlying asset price results in a higher option price. The larger volatility implies that an out-of-money option may end up deep in-the-money. However, the downside risk is limited for the option holder, because the option does not have to be exercised, that is, it does not matter how "deep" out-of-the-money the option is. This positive relationship between the volatility

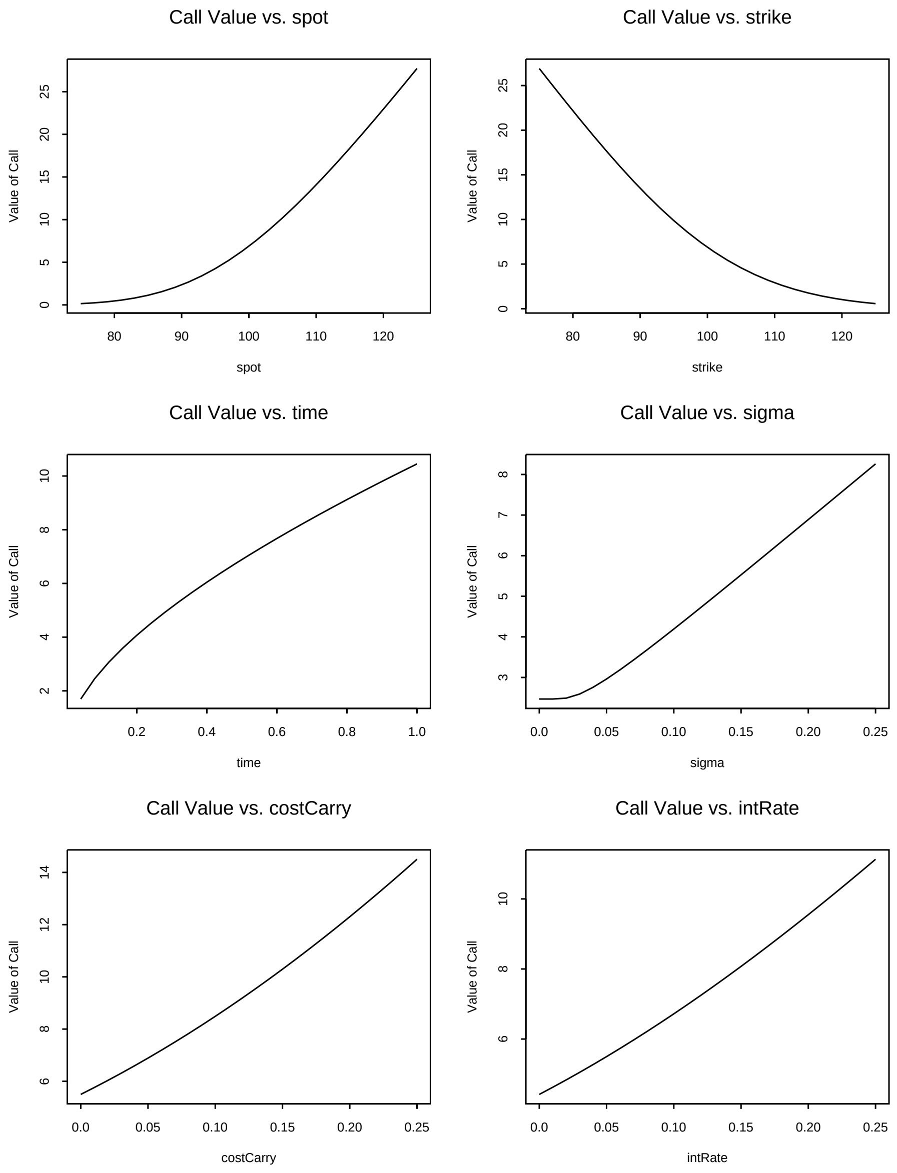

Figure 2: Call option sensitivity to various parameters. The Appendix gives the code for generating the plot.

of the underlying asset price and option price is illustrated in the fourth panel of Figure 2.

The higher is the cumulative cost required to hold a position in the underlying security, the larger is the possible saving by buying a call option instead. Also, high interest rates reduce the present value of the strike price. Therefore, high interest rates and low dividends result in high call option value. This positive relationship between the cost of carry ($b = r - q$) and the price of a call option is illustrated in the bottom left panel of Figure 2.

In the bottom right panel of Figure 2, dividends are held constant, and the interest rate is rising, which also implies an increasing call option value. Notice that if the positive cash flows to the holder of the underlying security exceed the interest rate, the cost of carry becomes negative.

2.1 Put Call Parity

Put-call parity defines a relationship between the price of a European call option and a European put option with identical strike price and expiration date on the same underlying asset. In the absence of arbitrage opportunities, put-call parity implies a unique price for the put option if the price of the call option is known, or vice versa.

First, consider a portfolio that consists of one call option and an amount of cash equal to Ke^{-rT} invested at the risk free interest rate r. On the expiration date, this portfolio has value:

- K if $S_T \leq K$ (the call has value 0 and the invested cash has value K).
- S_T if $S_T \geq K$ (the call has value $(S_T - K)$ and the invested cash has value K).

Second, consider a portfolio that consists of one put option and one unit of the underlying asset. On the expiration date, this portfolio has value:

- K if $S_T \leq K$ (the put has value $(K - S_T)$ and the asset has value S_T).
- S_T if $S_T \geq K$ (the put has value 0 and the asset has value S_T).

Notice that both portfolios have the same value at maturity:

$$\max(S_T, K)$$

Because the options cannot be exercised before the expiration date, no arbitrage implies that these two portfolios must have the same value today.

$$c + Ke^{-rT} = p + S_0$$

where c and p are the values of the call and put options today, respectively. Using the above relationship, for a given price of the call option, the underlying asset price, and the risk free interest rate, one can compute the implied price of the put option.

Example 1 *Put-Call Parity*

Consider two European call option with six months to expiry. The underlying stock price is \$100, the strike price on the first one is \$100, and on the second one is \$110. The risk free interest rate is 5% per year, and volatility is 20%. The following `S-PLUS` code generates the price of the corresponding put option. The first input argument of the `bsmEUParity` function is a European option object of class `opEuOption`. This can be created by another European option pricing function, such as `bsmEU`.

The `bsmEU` function computes the price of European call and put options using the Black Scholes formula to be described in section 2.3. It takes the following arguments:

`spot`: A numeric vector representing the current price of the underlying asset.
`strike`: A numeric vector representing the exercise price of the option.
`time`: A numeric vector representing the time to expiration of the option, expressed in years.
`intRate`: A numeric vector representing the annualized risk-free interest rate.
`sigma`: A numeric vector representing the annualized asset price standard deviation.
`costCarry`: A numeric vector representing the cost-of-carry rate, defined as $b = r - q$. Notice that, if the positive cash flows to the holder of the underlying security exceed the interest rate, the cost of carry becomes negative. If missing, the interest rate will be assigned as the cost-of-carry rate.
`type`: (optional) The type of the option as `call` (default) or `put`.

The length of the numeric vectors above must be of equal length to the number of options to be priced unless one value is to be used for all options. The `bsmEU` function returns an object of class `opEuOption`, which is a list specifying the spot price, strike price, time to maturity, interest rate, volatility, cost of carry, option type and price.

```
> call.opt.obj <- bsmEU(spot = 100, strike = c(100, 110),
+    time = 0.5, intRate = 0.05, costCarry = 0.05, sigma = 0.2,
+    type = "call")

> call.opt.obj

 2 European call(s)

     spot strike maturity interest.rate sigma cost.carry price
[1,]  100    100      0.5          0.05   0.2       0.05 6.889
[2,]  100    110      0.5          0.05   0.2       0.05 2.906

Valuation method --  black-scholes
```

Once the European option object of class `opEuOption` has been created, it can be passed to `bsmEUParity`. The second input argument is the type of the option whose price is to be determined by the put-call parity.

```
> bsmEUParity(call.opt.obj, type = "put")

 2 European put(s)

     spot strike maturity interest.rate sigma cost.carry price
[1,]  100    100      0.5          0.05   0.2       0.05  4.42
[2,]  100    110      0.5          0.05   0.2       0.05 10.19

Valuation method --  black-scholes
```

Similarly to `bsmEU`, the `bsmEUParity` function also returns a European option object of class `opEuOption`.

2.2 Valuation of Options in Discrete Time: The Binomial Option Pricing Model

The option pricing model developed by Cox, Ross, and Rubinstein (CRR) models the dynamics of the option's theoretical value f in discrete time. The model consists of a binomial tree of possible future underlying asset prices S over the life of the option.

2.2.1 Risk Neutral Valuation

The simplest binomial tree is the one-step tree with two states of nature at the end of the period. Assume that the length of period is δt, and that the asset price can either move up to $S_0 u$, where $u > 1$, or down to $S_0 d$, where $d < 1$. If the asset price moves up, the payoff from the option is f_u; if the asset price moves down, the payoff from the option is f_d.

Next, imagine a portfolio consisting of a long position in Δ units of the underlying asset, and a short position in one option. If the asset price moves up, the value of the portfolio at the end of the period is $S_0 u\Delta - f_u$; if the asset price moves down, the value of the portfolio at the end of the period is $S_0 d\Delta - f_d$.

The portolio is riskless if the payoffs in the two possible states of nature are equal:

$$S_0 u\Delta - f_u = S_0 d\Delta - f_d$$

which implies

$$\Delta = \frac{f_u - f_d}{S_0 u - S_0 d}$$

That is, the number of units of the underlying asset that makes the portfolio risk-neutral is Δ. Correspondingly, this risk-neutral portfolio is discounted at the risk-free interest rate. Its present value is

$$(S_0 u\Delta - f_u)e^{-r\delta t}$$

which must equal the cost of setting up the portfolio

$$(S_0 u\Delta - f_u)e^{-r\delta t} = S_0\Delta - f$$

This implies

$$f = e^{-r\delta t}(pf_u + (1-p)f_d)$$

where

$$p = \frac{e^{r\delta t} - d}{u - d}$$

is the risk-neutral probability of an upward movement of the underlying asset price.

In general, the risk-neutral probability of an upward movement is different from the real world (actual) probability of an upward movement. The real world probability of future movements in the price of the underlying asset is already incorporated into the asset price. Therefore, it does not have to be taken into account again when valuing the option in terms of the asset price as above. Thus, the value of an option is its expected payoff in a risk neutral world discounted at the risk-free rate.

2.2.2 Matching the Asset and the Tree

To calculate the parameters p, u and d, one needs three conditions. In the risk-neutral world, the expected return on the asset is the risk free interest rate r. One can match the expected price of the asset with the expected price on the binomial tree at the end of a very short time period δt:

$$S_0 e^{r\delta t} = pS_0 u + (1-p)S_0 d \tag{1}$$

or

$$e^{r\delta t} = pu + (1-p)d$$

which gives

$$p = \frac{e^{r\delta t} - d}{u - d} \tag{2}$$

Next, we match the variance of the return on the asset with the variance on the binomial tree:

$$\sigma^2 \delta t = pu^2 + (1-p)d^2 - [pu + (1-p)d]^2$$

By substituting for p, and ignoring the quadratic and higher order terms of δt in the power series approximation of $e^{\mu\delta t}$, and imposing the third condition

$$u = \frac{1}{d}$$

we get the values of u and d proposed by Cox, Ross, and Rubinstein (1979)

$$u = e^{\sigma\sqrt{\delta t}}$$
$$d = e^{-\sigma\sqrt{\delta t}}$$

If there are any positive cash flows, q, to the owner of the underlying asset, the asset must on average — in a risk neutral world — provide a return $b = r - q$. Therefore, equation (1) becomes:

$$S_0 e^{b\delta t} = pS_0 u + (1-p)S_0 d$$

so that

$$e^{b\delta t} = pu + (1-p)d$$

which gives

$$p = \frac{e^{b\delta t} - d}{u - d} \tag{3}$$

2.2.3 Binomial Lattice

The third condition $u = 1/d$ enables us to create a binomial tree centered at the initial asset price whose nodes recombine. That is, we get to the same price S_0 after an up-down move S_0ud and a down-up move S_0du. Therefore, at time δt, there are 2 possible asset prices S_0u, S_0d; at time $2\delta t$, there are 3 possible asset prices S_0u^2, S_0, S_0d^2; and so on. At time $i\delta t$, we have to consider $i+1$ asset prices

$$S_0u^jd^{i-j}, \quad j = 0, 1, \ldots, i$$

On the expiration date, the value of the option is equal to its intrinsic value. The value of the option on each node at time $T - \delta t$ can be calculated as the expected value at expiration in a risk neutral world discounted at the risk-free rate for a time period δt. Similarly, the value of the option on each node at time $T - 2\delta t$ can be calculated as the expected value at time $T - \delta t$ in a risk neutral world discounted at the risk-free rate for a time period δt.

Working backward through the whole tree, we get the value of the option at time zero. Note that the value of an American option at any given node also depends on its early exercise (intrinsic) value. If early exercise is preferred on the given node, the option's intrinsic value is considered instead of its discounted expected value.

2.2.4 Algorithm for Binomial Option Pricing

Define $f_{i,j}$ as the the value of the option at the (i, j) node, where $0 \leq i \leq N$ is the index in the time dimension, and $0 \leq j \leq i$ is the index in the asset price dimension. The price of the asset at the (i, j) node is $S_0u^jd^{i-j}$. The value of an American call option on the expiration date is $\max(K - S_T, 0)$, therefore

$$f_{N,j} = \max(S_0u^jd^{N-j} - K, 0), \quad j = 0, 1, \ldots, N$$

Assuming no early exercise at $(N-1)\delta t$, risk neutral valuation gives

$$f_{N-1,j} = e^{-r\delta t}[pf_{N,j+1} + (1-p)f_{N,j}], \quad j = 0, 1, \ldots, N-1$$

for time period $(N-1)\delta t$, or

$$f_{i,j} = e^{-r\delta t}[pf_{i+1,j+1} + (1-p)f_{i+1,j}]$$

for $0 \leq i \leq N-1$ and $0 \leq j \leq i$. When early exercise is taken into account, the discounted expected value of the option above has to be compared to its intrinsic value, and the value of the option is

$$f_{i,j} = \max\{S_0u^jd^{N-j} - K,\ e^{-r\delta t}[pf_{i+1,j+1} + (1-p)f_{i+1,j}]\}$$

Working backward this way, the value of the option at time $i \cdot \delta t$ captures the effect of early exercise possibilities at all subsequent times. In the limit, as the length of time subintervals δt tends to zero and the number of subintervals N tends to infinity, an exact value for an American call is obtained.

Again, if there are any positive cash flows, q, to the owner of the underlying asset, the asset must on average — in a risk neutral world — provide a return $b = r - q$. Therefore, the risk-neutral probability of an upward movement of the underlying asset price is given by equation (3) instead of equation (2).

Example 2 *Binomial Tree*

Consider an American call option with six months to expiry. The underlying stock price is \$100, the strike price is \$100, the risk free interest rate is 5% per year, the dividend yield is 8% per year, and the volatility is 20%. First, calculate a 5 step tree of asset prices, displayed in figure 3.

The `opBinomTree` function computes the asset prices in the binomial tree using risk-neutral valuation. The `opBinomTree` function requires the following input arguments in addition to the parameters of the `bsmEU` function described in Example 1 (except `type`):

`steps`: A numeric value representing the number of steps to be used in the construction of the binomial tree. Defaults to 1000 steps. Note that the step size is determined by time/steps.

`method`: (optional) A character string that indicates whether the binomial tree to be built is an additive (`addi`) or multiplicative (`multi`) recombining tree. The tree constructed using the additive method reflects that the underlying asset follows an arithmetic Brownian motion, while a tree constructed using the multiplicative method assumes that the underlying asset follows a geometric Brownian motion. Defaults to `addi`. All options to be priced must use the same method.

`nSteps`: (optional) A sequence of numbers of equal length to the number of time steps. It is used to label the columns of the binomial tree. If mising, the column labels will start from step 0 up to the total number of steps.

The function `opBinomTree` returns a list of asset prices at each step in the binomial tree along with the time, probability of upward move, time to expiration, time steps, step size, and method used to compute the prices.

```
> n.steps <- 5
> asset.tree <- opBinomTree(spot = 100, time = 0.5,
+    intRate = 0.05, costCarry = 0.05-0.08, method = "multi",
+    sigma = 0.2, steps = n.steps)
> values <- asset.tree$prices != 0
> plot(rep(1:(n.steps + 1), 1:(n.steps + 1)),
+    asset.tree$prices[values], xlab = "Step",
+    ylab = "Asset Price", labex = 2)
> text(rep(1:(n.steps + 1), 1:(n.steps + 1)),
+    asset.tree$prices[values] + 2,
+    round(asset.tree$prices[values], dig = 2))
> title(main = "Binomial Tree")
```

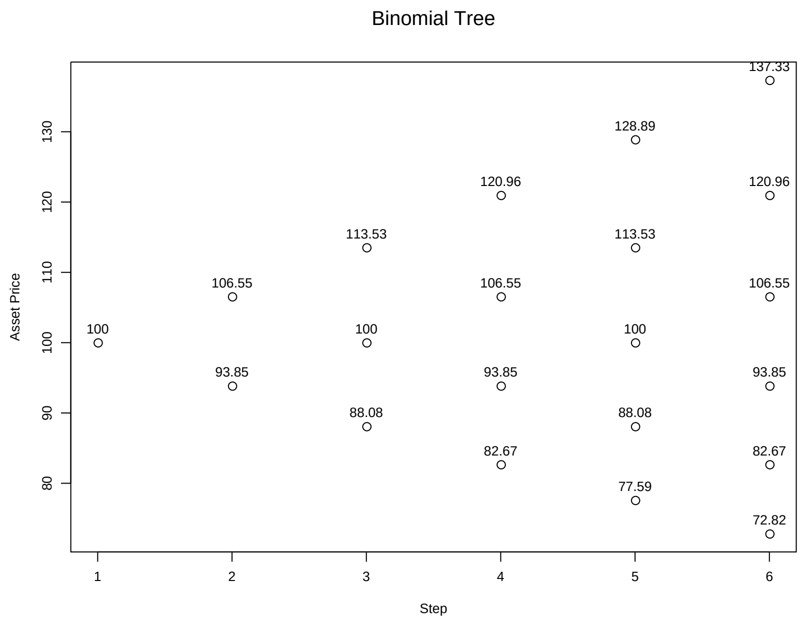

Figure 3: Binomial tree of asset prices.

Second, calculate the option price as the present value of the expected cash flow. The `opBinomCashFlow` function computes the price of European and American call and put options based on the binomial tree built for the underlying asset. It requires the following input arguments:

`binomTree`: A list object representing a binomial tree built by another function such as opBinomTree.

`strike`: A numeric value representing the exercise price of the option.

`intRate`: A numeric value representing the annualized risk-free interest rate.

`type`: (optional) A character string representing the type of the option as either `call` (default) or `put`.

`euroAmFlag`: (optional) A character string representing the type of the option as either the default type as a European option (EURO) or an American option (`am`).

`method`: (optional) A character string representing the method used to price the option. The default method `induction` uses iterative backward induction while `AD` uses the Arrow Debreu method. The binomial tree constructed by opBinomTree does not have Arrow Debreu information in the tree and therefore this method cannot be implemented.

```
> opBinomCashFlow(asset.tree, intRate = 0.05,
+    strike = 100, type = "call", euroAmFlag = "am")
[1] 5.204593
```

The `opBinomCashFlow` function returns the price of the option computed by the binomial cashflow tree. It can only price one option at a time. For a consistency check, one can compare this result with the one using the `bsmAMBAW` or `bsmAMBS` function in section 2.5. Increasing the number of steps in the tree increases the precision of the results. For example, `steps = 50` in the `opBinomTree` function brings the option price down to 4.915162.

2.3 Valuation of European Options in Continuous Time: The Black-Scholes-Merton Model

Over a very short period of time, the price of an option is perfectly correlated with the price of the underlying asset; therefore, a risk neutral portfolio, consisting of the option and the underlying asset, can be set up. By employing the technique of constructing such a risk neutral portfolio that replicates the returns of holding an option, Fisher Black, Myron Scholes, and Robert Merton produced a closed-form solution for a European option's theoretical price in continuous time.

2.3.1 Brownian Motion

In a *Markov process*, the state of the process in the future is conditionally independent of the history of the process in the past. That is, the conditional probability distribution of the process in the future depends only on the current state. Let X_n be a random variable representing the state at time n. $\{X_n\}$ is a Markov process if

$$E(X_{n+1}|X_n, \ldots, X_1) = E(X_{n+1}|X_n)$$

A Markov process with mean change zero is a *martingale*, that is, a stochastic process such that the conditional expected value of an observation at some future time, given all the observations up to present, is equal to the observation at present:

$$E(X_{n+1}|X_n, \ldots, X_1) = X_n$$

A Markov process with mean change zero and a variance rate of 1.0 per time period is called a *Wiener process*. Formally, a variable W follows a Wiener process if its increments δW are independent, and

$$\delta W = \epsilon\sqrt{\delta t}, \quad \text{where} \quad \epsilon \sim N(0,1)$$

In general, asset price changes have nonzero means and nonunity variances. To account for these specifics, the Wiener process can be replaced by a *generalized Brownian motion* consisting of a deterministic drift term and a stochastic diffusion term:

$$dX = a\ dt + b\ dW \tag{4}$$

where a and $b > 0$ are constants.

Even more flexibility can be achieved by considering an *Itô process*. The underlying asset's price, S, is said to follow an Itô process if the expected drift rate, $a(S,t)$, and variance rate, $b(S,t)^2$, depend on S itself and time:

$$dS = a(S,t)\ dt + b(S,t)\ dW \tag{5}$$

The generalized Brownian motion with constant coefficients a and b in (4) is normally distributed, which means that there is some probability of observing negative asset prices. In addition, the increments of (4) have constant expectation, which implies that the incremental percentage change of asset prices, or the percentage rate of return on the asset, would be declining as the asset price rises. To avoid such problems, asset prices S are modeled as variables following a *geometric Brownian motion*, e^X, with X following a generalized Brownian motion with constant coefficients defined in equation (4).

Asset prices, S, are said to follow a geometric Brownian motion if they satisfy the following stochastic differential equation:

$$dS = \mu S\ dt + \sigma S\ dW \tag{6}$$

Equation (6) is a specific form of (5) describing an Itô process. The returns on the asset can be written as

$$\frac{dS}{S} = \mu\ dt + \sigma\ dW$$

that is, for short time increments δt

$$\frac{\delta S}{S} \sim N(\mu\ \delta t, \sigma\ \delta W) = N(\mu\ \delta t, \sigma\sqrt{\delta t})$$

where, in line with the assumptions of the Black-Scholes-Merton model, μ is the constant expected rate of return on the asset, and σ is the constant volatility of the asset price.

2.3.2 Itô's Lemma

The price of an option $f(S,t)$ depends on the underlying asset's price (a stochastic variable) and time. According to Itô's lemma, a function G of an Itô process S and time t follows the process

$$dG(S,t) = \left(\frac{\partial f}{\partial S}a + \frac{\partial f}{\partial t} + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}b^2\right)dt + \frac{\partial f}{\partial S}b\ dW \tag{7}$$

In other words, the process for the option price can be written as

$$df = \left(\frac{\partial f}{\partial S}\mu S + \frac{\partial f}{\partial t} + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2\right)dt + \frac{\partial f}{\partial S}\sigma S\ dW \tag{8}$$

Thus, f also follows an Itô process, with drift rate

$$\frac{\partial f}{\partial S}\mu S + \frac{\partial f}{\partial t} + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2$$

and variance rate

$$\left(\frac{\partial f}{\partial S}\right)^2 \sigma^2 S^2$$

Note, both S and f are affected by the same source of uncertainty, dW.

2.3.3 The Lognormal Property of Asset Prices

In the Black-Scholes-Merton model, the asset price S is assumed to follow a geometric Browninan motion described by equation (6) and to have a lognormal distribution. The stochastic process followed by $\ln S$ can be derived by using Itô's lemma. Define

$$G = \ln S$$

It follows that

$$\frac{\partial G}{\partial S} = \frac{1}{S}\ , \quad \frac{\partial^2 G}{\partial S^2} = -\frac{1}{S^2}\ , \quad \frac{\partial G}{\partial t} = 0$$

Substituting these values into Itô's lemma in equation (7) gives the process followed by $\ln S$:

$$d \ln S = \left(\mu - \frac{\sigma^2}{2}\right) dt + \sigma\ dW$$

Thus, $\ln S$ follows a generalized Brownian motion with constant drift rate $\mu - \sigma^2/2$ and constant variance rate σ^2. The change in $\ln S$ is normally distributed:

$$\ln S_T - \ln S_0 \sim N\left[\left(\mu - \frac{\sigma^2}{2}\right) T, \sigma\sqrt{T}\right]$$

where S_T is the stock price at a future time T, and S_0 is the stock price at time 0. The drift rate $\mu - \sigma^2/2$ equals the expected value of the *continuously compounded* rate of return.

2.3.4 The Black-Scholes-Merton Differential Equation

By choosing a risk-neutral portfolio of the option and the underlying asset, the Wiener process can be eliminated from equation (6) and (8). The portfolio will consist of

- short 1 option.
- long $\partial f/\partial S$ units of the underlying asset.

The value of the portfolio is

$$\Pi = -f + \frac{\partial f}{\partial S} S$$

The change in the value of the portfolio, $d\Pi$, in an small time interval, dt, is

$$d\Pi = -df + \frac{\partial f}{\partial S} dS$$

After substituting equation (6) and (8), for dS and df respectively, we get

$$d\Pi = \left(-\frac{\partial f}{\partial t} - \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2\right) dt$$

Because this equation does not involve dW, the portfolio must be riskless during the time interval dt. The no-arbitrage argument implies that the portfolio must earn the risk free interest rate. Therefore

$$d\Pi = r\Pi dt$$

where r is the risk free interest rate. Substituting for Π and $d\Pi$, we get

$$\left(\frac{\partial f}{\partial t} + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2\right) dt = r\left(f - \frac{\partial f}{\partial S}S\right) dt$$

so that

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2 = rf$$

This is the Black-Scholes-Merton partial differential equation. Its solution depends on the boundary conditions. At the expiration date, the boundary condition of a European call option is

$$f = \max(S_T - K, 0)$$

and for a European put option is

$$f = \max(K - S_T, 0)$$

The Black-Scholes formulas for the time zero prices of European call and put options on a non-dividend-paying asset are

$$\begin{aligned} c &= S_0\Phi(d_1) - Ke^{-rT}\Phi(d_2) \\ p &= Ke^{-rT}\Phi(-d_2) - S_0\Phi(-d_1) \end{aligned}$$

where

$$\begin{aligned} d_1 &= \frac{\ln(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}} \\ d_2 &= \frac{\ln(S_0/K) + (r - \sigma^2/2)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T} \end{aligned}$$

and the function $\Phi(\cdot)$ is the cumulative distribution function for a standard normal random variable.

2.3.5 The Generalized Black-Scholes-Merton Formula

When options on dividend paying stock, futures, or currency options are being analyzed, the cost of holding the underlying asset, the cost of carry b, has to be taken into account . The generalized Black-Scholes-Merton formula incorporates these cases. The formulas for the prices of European call and put options are

$$\begin{aligned} c &= S_0e^{(b-r)T}\Phi(d_1) - Ke^{-rT}\Phi(d_2) \\ p &= Ke^{-rT}\Phi(-d_2) - S_0e^{(b-r)T}\Phi(-d_1) \end{aligned}$$

where

$$d_1 = \frac{\ln(S_0/K) + (b + \sigma^2/2)T}{\sigma\sqrt{T}}$$
$$d_2 = \frac{\ln(S_0/K) + (b - \sigma^2/2)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T}$$

Depending on the underlying asset, the cost of carry, b, will take on the following values:

- $b = r$ gives the Black Scholes (1972) pricing formula for an option on a non-dividend paying stock.
- $b = r - q$ gives the Merton (1973) pricing formula for an option paying dividend yield q.
- $b = r$ gives the Black (1976) pricing formula for futures options.
- $b = r - r_f$ gives the Garman and Kohlhagen (1983) pricing formula for currency options with foreign risk-free interest rate r_f.

Besides the pricing function `bsmEU`, based on the generalized Black-Scholes-Merton formula involving the of carry, `S+FinMetrics` offers further functions that are tailored to deal with the various cases mentioned above. They are listed in Table 1.

Name	**Description**
`bsmEU`	European Option Pricing with the Generalized BSM Formula
`bsmStockEU`	Pricing function tailored for European Stock Options
`bsmFuturesEU`	Pricing function tailored for European Futures Options
`bsmFXEU`	Pricing function tailored for European Foreign Exchange Options

Table 1: List of `S+FinMetrics` functions for pricing vanilla European put and call options.

In the following example, the currency options are first priced with the `bsmFXEU` function and then with the `bsmEU` function.

Example 3 *Currency Options*

Consider two European call options on the British Pound with six and twelve months to expiry respectively. The exchange rate is 1.5\$/£. The strike price on the first option is 1.5\$/£, and on the second 1.6\$/£. The US risk free interest rate is 5% per year, the UK risk free interest rate is 8% per year, and the volatility of the exchange rate is 20%. First apply the specialized BSM function for foreign currency options, `bsmFXEU`, that requires the following input arguments:

`fxrate`: A numeric vector representing the current foreign exchange rate.
`strike`: A numeric vector representing the exercise price of the option.
`time`: A numeric vector representing the time to expiration of the option, expressed in years.
`intRate`: A numeric vector representing the annualized domestic risk-free interest rate.
`intRate.foreign`: A numeric vector representing the annualized risk-free foreign interest rate.
`sigma`: A numeric vector representing the annualized foreign exchange rate volatility.
`type`: (optional) The type of the option as `call` (default) or `put`.

The length of the numeric vectors above must be of equal length to the number of options to be priced unless one value is to be used for all options. The `bsmFXEU` function returns an object of class `opEuOption`, which is a list specifying the spot price, strike price, time to maturity, interest rate, volatility, cost of carry, option type and price.

```
> bsmFXEU(fxrate = 1.5, strike = c(1.5, 1.6), time = c(0.5, 1),
+         intRate = 0.05,intRate.foreign = 0.08, sigma = 0.2,
+         type = "call")

 2 European call(s)

     spot strike maturity interest.rate sigma cost.carry   price
[1,]  1.5    1.5      0.5          0.05   0.2      -0.03 0.07143
[2,]  1.5    1.6      1.0          0.05   0.2      -0.03 0.05976

Valuation method --  black-scholes
```

The same result obtains using the general BSM function `bsmEU`, already encountered in Example 1. Instead of the domestic and foreign interest rates, the `bsmEU` function requires the cost of carry to be specified ($b = r - r_f = 0.05 - 0.08 = -0.03$).

```
> bsmEU(spot = 1.5, strike = c(1.5, 1.6), time = c(0.5, 1),
+       intRate = 0.05, costCarry = -0.03, sigma = 0.2,
+       type = "call")

 2 European call(s)

     spot strike maturity interest.rate sigma cost.carry   price
[1,]  1.5    1.5      0.5          0.05   0.2      -0.03 0.07143
[2,]  1.5    1.6      1.0          0.05   0.2      -0.03 0.05976

Valuation method --  black-scholes
```

2.3.6 Implied Volatility

One parameter of the Black-Scholes-Merton (BSM) formula that cannot be directly observed is the volatility of the underlying asset's price, σ. However, its value can be approximated by the volatility implied by the option prices observed in the market.

The `S+FinMetrics` function `opImpVol` computes the implied volatility of an option. It takes as input an option object, which can contain several individual options, for which the option prices have been defined. It then estimates the volatility (or volatilities) of the underlying asset(s) that would give these prices, assuming that all other parameters remain fixed. In order to do this, `opImpVol` uses the formula or algorithm that was used to calculate the option prices, which is identified by the class of the option object. For example, an object with class `opEuOption` will be priced using the Black-Scholes-Merton formula for European options.

Example 4 *Implied Volatility*

Consider a European call option with six months to expiry. The underlying stock price is \$100, the strike price is \$100, the risk free interest rate is 5% per year. The price of the option observed in the market is \$10. First create an object with class `opEuOption` with an initial estimate of the volatility of the underlying asset, σ.

```
> opt.obj <- bsmEU(spot = 100, strike = 100, time = 0.5,
+     intRate = 0.05, costCarry = 0.05, sigma = 0.2,
+     type = "call")
> opt.obj

 1 European call(s)

     spot strike maturity interest.rate sigma cost.carry price
[1,]  100    100      0.5          0.05   0.2       0.05 6.889

Valuation method --  black-scholes
```

The following input arguments can be specified for the `opImpVol` function:

`object:` (required) an option object created by an option pricing function.

`method:` a character string representing the method used to compute the implied volatility. The allowable methods are bisection, secant, nr for the Newton-Rahpson method, and appcm for the Corrado and Miller approximation method for calculating the implied volatility of options of class opEuOption. The default is the bisection method.

`sigma.low:` a numeric object representing the initial lower bound on implied volatility, which is used by the bisection and secant methods. The default value of sigma.low is 0.01.

sigma.high: a numeric object representing the initial upper bound on implied volatility, which is used by the bisection and secant methods. The default value of sigma.high is 1.0.

sigma0: a numeric object that gives the initial estimate for implied volatility used in the Newton-Raphson method. If no initial estimate is supplied then a value of 0.40 is used, unless the option is of class opEuOption, in which case the Manaster and Koehler formula is used to derive a more accurate starting value.

n.iter: a numeric object representing the maximum number of iterations allowed to find the implied volatility, which is used by the bisection, secant and Newton-Raphson methods. The default number of iterations is 30.

eps: a numeric object representing epsilon, which is the tolerance used for the bisection, secant and Newton-Rahpson methods. That is, these methods will stop when they find volatilities that give prices within epsilon of the original prices, provided that the limit on the number of iterations has not been exceeded. The default value for eps is 1e-06.

eps.vega: a numeric object representing epsilon which is used in numerical differentiation when computing Vega, which is the derivative of the option price relative to changes in volatility. This is used in the Newton-Raphson method. The default value for eps.vega is 1e-04.

The `opImpVol` function returns a list of the implied volatility values:

```
> opImpVol(opt.obj, method = "nr", sigma0 = 0.2)
$sigma:
[1] 0.2

$iter:
[1] 1

$eps:
[1] 1e-006
```

The above result is just a consistency check for the `bsmEU` and `opImpVol` functions. The option price was calculated based on the estimated volatility. Therefore, the implied volatility based on the BSM price has to equal the volatility supplied to the `bsmEU` function. To get the implied volatility based on the market price, the price stored in the option object has to be changed:

```
> opt.obj$price <- 10
> opImpVol(opt.obj, method = "nr", sigma0 = 0.2)
$sigma:
[1] 0.3132713

$iter:
[1] 3
```

```
$eps:
[1] 1e-006
```

Note that, the prices of deep-in-the-money and deep-out-of-money options are relatively insensitive to volatility. Therefore, implied volatilities calculated from these options tend to be unreliable.

The implied volatility of a European put is the same as the implied volatility of a European call when the two options have the same strike price and maturity date. To see this, compare the put call parity for the BSM model and for the actual market prices of options in the absence of arbitrage opportunities:

$$\begin{aligned} p_{BSM} + S_0 e^{(b-r)t} &= c_{BSM} + K e^{-rt} \\ p_{mkt} + S_0 e^{(b-r)t} &= c_{mkt} + K e^{-rt} \end{aligned}$$

subtracting the two equations, we get

$$p_{BSM} - p_{mkt} = c_{BSM} - c_{mkt} \tag{9}$$

That is, the pricing errors for the put and call options are the same. Now, suppose that the implied volatility of the put option is 20%. This means that $p_{BSM} = p_{mkt}$ when 20% volatility is used in the BSM model. Then, from equation (9), it follows that $c_{BSM} = c_{mkt}$, and the implied volatility of the call is also 20%.

2.4 Measuring Option Risk

The sensitivity of options to changes in market conditions can be measured by the "Greeks" — that is, sensitivity measures denoted by Greek letters. Each Greek letter measures a different dimension to the risk in an option position. In the case of a vanilla option, the strike price is fixed in advance and does not change. However, there is a nonlinear relationship between the value of an option and the change in any one of the remaining variables underlying the price of the option.

Besides the analytical risk measures based on the Black-Scholes-Merton formulas, `S+FinMetrics` has built in functions with numerical approximation of option sensitivities for American and Exotic options. Table 2 lists the two types side by side.

BSM based functions	**Numerical approximation**
`opDeltaBSM`	`opDelta`
`opLambdaBSM`	`opLambda`
`opGammaBSM`	`opGamma`
`opThetaBSM`	`opTheta`
`opVegaBSM`	`opVega`
`opRhoBSM`	`opRho`
`opSenCostCarryBSM`	`opSenCostCarry`

Table 2: List of `S+FinMetrics` functions computing sensitivity measures.

Figure 5 shows the sensitivity measures of a vanilla European call option with respect to the price of the underlying asset. The diagrams are drawn for default values of $K = 100,\ r = 0.05,\ b = 0.05,\ \sigma = 0.2$. The different line types represent different times to maturity, ranging from 1 day (full line) to 6 months (short-dashed line). A detailed description of each risk measure follows below.

2.4.1 Delta (Δ)

Delta measures the sensitivity of the option price to a change in the price of the underlying asset. For European options it is defined as

$$\Delta_{call} = \frac{\partial c}{\partial S} = e^{(b-r)T}\Phi(d_1)$$
$$\Delta_{put} = \frac{\partial p}{\partial S} = e^{(b-r)T}(\Phi(d_1) - 1)$$

The fact that Δ measures the rate of change in the option price relative to the change in the price of the underlying asset can be applied in delta-hedging. As shown in the sections on risk-neutral valuation and the BSM differential equation, a portfolio of short 1 option and long Δ units of the underlying asset is delta neutral for a short period of time. That is, it should earn the risk-free interest rate.

Example 5 *Delta based on the BSM model*

The `opDeltaBSM` function requires an object of class `opEuOption`, and returns a numeric object representing the Delta of the option. Consider two European call options with six months to expiry. The underlying stock price is \$100, and the strike price is \$100 and \$110, respectively. The risk free interest rate is 5% per year and the volatility of the underlying asset price is 20% per year. The object of class `opEuOption` can be created by:

```
> opt.obj.BSM <- bsmEU(spot = 100, strike = c(100, 110),
+    time = 0.5, intRate = 0.05, costCarry = 0.05,
+    sigma = 0.2, type = "call")
> opt.obj.BSM

 2 European call(s)

     spot strike maturity interest.rate sigma cost.carry price
[1,]  100    100      0.5          0.05   0.2       0.05 6.889
[2,]  100    110      0.5          0.05   0.2       0.05 2.906

Valuation method --  black-scholes
```

Then, the object of class `opEuOption` can be supplied to the `opDeltaBSM` function, which computes the Delta of the option:

```
> opDeltaBSM(opt.obj.BSM)
[1] 0.5977345 0.3348873
```

Alternatively, the parameters of the BSM model can directly be specified in the `opDeltaBSM` function:

```
> opDeltaBSM(spot = 100, strike = c(100, 110),
+    time = 0.5, intRate = 0.05, costCarry = 0.05,
+    sigma = 0.2, type = "call")
[1] 0.5977345 0.3348873
```

The analytical formulas for option sensitivities described above are based on the Black-Scholes-Merton formulas and are valid for European options. In the general case, for any option type, finite difference approximations can be used to calculate the sensitivities. This method requires that the value of the option be calculated accurately.

For example, first order partial derivatives such as Δ can be approximated by the two sided finite difference method

$$\Delta_{call} = \frac{\partial c}{\partial S} \approx \frac{c(S + \delta S, K, T, r, b, \sigma_1) - c(S - \delta S, K, T, r, b, \sigma_2)}{2\delta S}$$

Notice that in the above approximation, the volatility does not have to be fixed.

Example 6 *Delta based on numerical approximation*

The `opDelta` function numerically approximates the Delta of any option. It requires an option object that is identified by its class, and returns a numeric object representing the Delta of the option. For example the `bsmSimpleChooserEU` function (specified in Example 20) returns an object of class `opChooserOption`, which is a list specifying the spot, strike, time to maturity, time to choose, interest rate, sigma, cost carry and computed option price.

```
> opt.obj.NUM <- bsmSimpleChooserEU(spot = 100,
+    strike = c(100, 110), time = 1, intRate = 0.05,
+    costCarry = 0.05, sigma = 0.2, t1 = c(0.25, 0.5))
> opt.obj.NUM

 2 Chooser option

                 [,1]   [,2]
           spot 100.00 100.00
         strike 100.00 110.00
       maturity   1.00   1.00
 time.to.choose   0.25   0.50
  interest.rate   0.05   0.05
          sigma   0.20   0.20
     cost.carry   0.05   0.05
          price  12.38  14.42
```

Upon passing this object to the `opDelta` function, `opDelta` returns a numeric object representing the Delta of the option.

```
> opDelta(opt.obj.NUM)
$spot:
[1]  0.3456710 -0.1489349
```

For the European option object cretated above, `opDelta` gives the same result as `opDeltaBSM`.

```
> opDelta(opt.obj.BSM)
$spot:
[1] 0.5977345 0.3348873
```

The following code produces a 3D diagram of numerical deltas as functions of S and T displayed in Figure 4.

```
> nr.nodes <- 26
> S.margin <- seq(from = 75, to = 125, length = nr.nodes)
> t.margin <- seq(from = 2/52, to = 1, length = nr.nodes)
> St.grid <- expand.grid(list(S = S.margin, t = t.margin))
> opt.obj <- bsmEU(spot = St.grid[1], strike = 100,
+    time = St.grid[2], intRate = 0.05, costCarry = 0.05,
+    sigma = 0.2, type = "call")

> Delta <- matrix(opDelta(opt.obj, eps = 0.0001)[[1]],
+    ncol = nr.nodes)

> persp.data <- list(x = S.margin, y = t.margin, z = Delta)
> persp.data.range <- c(diff(range(S.margin, na.rm = T)),
+    diff(range(t.margin, na.rm = T)),
+    diff(range(Delta, na.rm = T)))
> eye.level <- c(-6, -8, 5)*persp.data.range
> persp(persp.data, xlab = "S", ylab = "T", zlab = "Delta",
+    axes = T, box = T, eye = eye.level)
> title(main = "Delta")
```

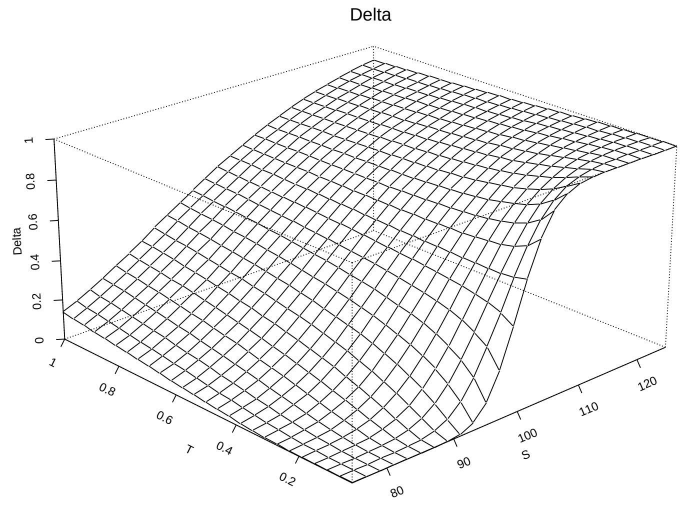

Figure 4: Numerical Delta in 3D as function of time to maturity and spot price of underlying asset.

2.4.2 Lambda (Λ)

Lambda measures the pecentage change of the option price to a one percent change in the price of the underlying asset. For European options, it is defined as

$$\Lambda_{call} = \frac{\partial c}{\partial S}\frac{S}{c} = \Delta_{call}\frac{S}{c} = e^{(b-r)T}\Phi(d_1)\frac{S}{c}$$
$$\Lambda_{put} = \frac{\partial p}{\partial S}\frac{S}{p} = \Delta_{put}\frac{S}{p} = e^{(b-r)T}(\Phi(d_1)-1)\frac{S}{p}$$

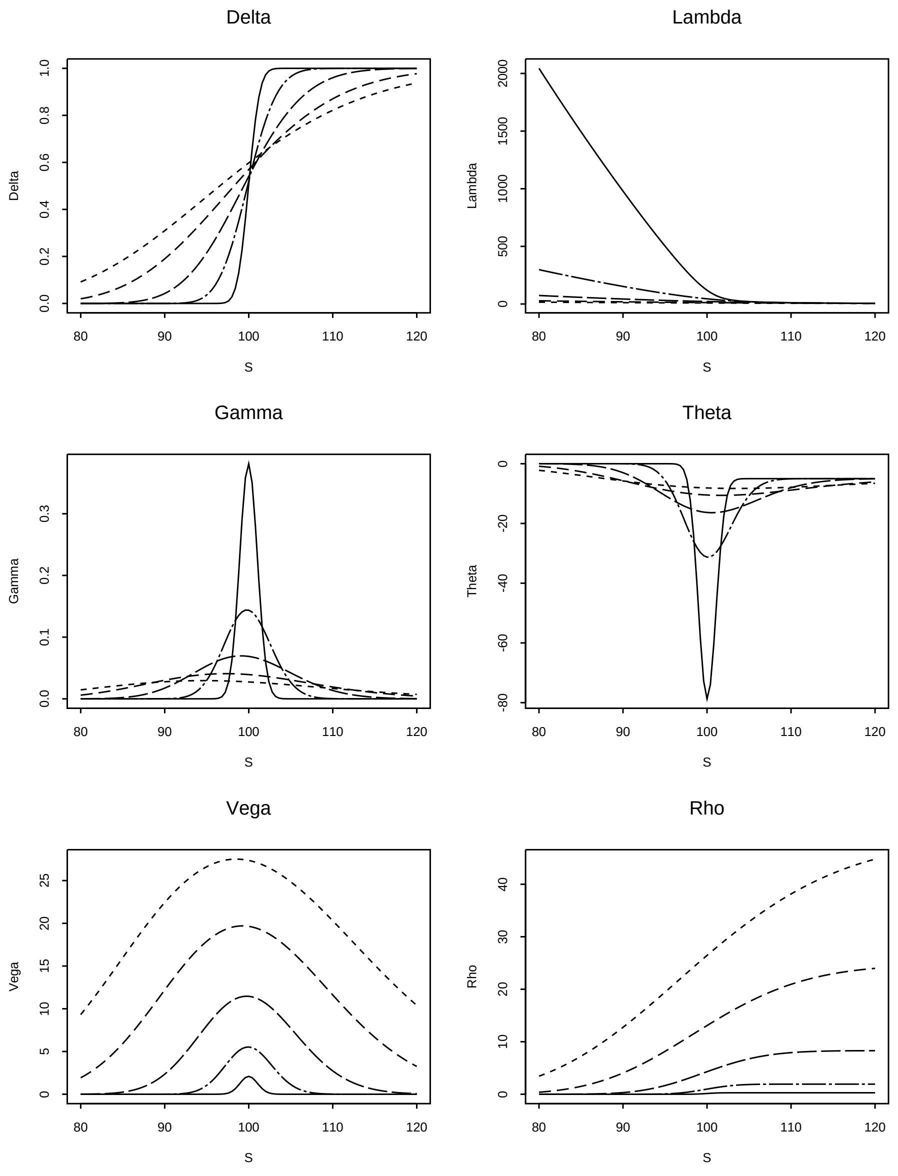

Figure 5: Greeks of a European call option in 2D. Dashed lines represent increasing time to maturity: $T = \{$1 day (full line), 1 week, 1 month, 3 months, 6 months$\}$. The Appendix gives the code for generating the plot.

In other words, Λ measures the elasticity of the option price with respect to the price of the underlying asset; therefore, indicates the leverage built in the option.

Example 7 *Lambda*

The `opLambdaBSM` function requires an object of class `opEuOption`, and returns a numeric object representing the Lambda of the option. The `opLambda` requires an option object that is identified by its class, and returns a numeric object representing the Delta of the option. The option objects `opt.obj.BSM` and `opt.obj.NUM` were defined in Example 5 and 6, respectively.

```
> opLambdaBSM(opt.obj.BSM)
[1]  8.676993 11.522126
> opLambda(opt.obj.BSM)
$spot:
[1]  8.676993 11.522126

> opLambda(opt.obj.NUM)
$spot:
[1]  2.792515 -1.032941
```

2.4.3 Gamma (Γ)

Gamma measures the sensitivity of the option's delta to a change in the price of the underlying asset. For European options, it is defined as

$$\Gamma_{call} = \frac{\partial^2 c}{\partial S^2} = \frac{\Phi'(d_1)e^{(b-r)T}}{S_0\sigma\sqrt{T}}$$
$$\Gamma_{put} = \frac{\partial^2 p}{\partial S^2} = \frac{\Phi'(d_1)e^{-(b-r)T}}{S_0\sigma\sqrt{T}}$$

where $\Phi'(d_1)$ is the probability density function for a standard normal random variable evaluated at d_1. If Γ is large, Δ is highly sensitive to the price of the underlying asset. Γ also measures the curvature of the relationship between the option price and the underlying asset price. When there is a movement in the underlying price, large Γ implies a large amount of rebalancing to maintain a delta-neutral portfolio.

Example 8 *Gamma*

The `opGammaBSM` function requires an object of class `opEuOption`, and returns a numeric object representing the Gamma of the option. The `opGamma` requires an option object that is identified by its class, and returns a numeric object representing the Delta of the option. The option objects `opt.obj.BSM` and `opt.obj.NUM` were defined in Example 5 and 6, respectively.

```
> opGammaBSM(opt.obj.BSM)
[1] 0.02735866 0.02575748
> opGamma(opt.obj.BSM)
$spot:
[1] 0.02735874 0.02575611

> opGamma(opt.obj.NUM)
$spot:
[1] 0.05305800 0.04713172
```

2.4.4 Theta (Θ)

Theta measures the sensitivity of the option price to a change in time to maturity. For European options, it is defined as

$$\Theta_{call} = \frac{\partial c}{\partial T} = -\frac{S_0\Phi'(d_1)\sigma e^{(b-r)T}}{2\sqrt{T}} + qS_0\Phi(d_1)e^{(b-r)T} - rKe^{-rT}\Phi(d_2)$$
$$\Theta_{put} = \frac{\partial p}{\partial T} = -\frac{S_0\Phi'(d_1)\sigma e^{(b-r)T}}{2\sqrt{T}} - qS_0\Phi(-d_1)e^{(b-r)T} + rKe^{-rT}\Phi(-d_2)$$

As an option approaches its expiration date, its time value decays, and Θ measures how much time value is lost as a given period elapses.

Example 9 *Theta*

The `opThetaBSM` function requires an object of class `opEuOption`, and returns a numeric object representing the Theta of the option. The `opTheta` requires an option object that is identified by its class, and returns a numeric object representing the Theta of the option. The option objects `opt.obj.BSM` and `opt.obj.NUM` were defined in Example 5 and 6, respectively.

```
> opThetaBSM(opt.obj.BSM)
[1] -8.115968 -6.680609
> opTheta(opt.obj.BSM)
$time:
[1] -8.115968 -6.680609

> opTheta(opt.obj.NUM)
$time:
[1] -4.861834 -2.492004

$t1:
[1] -6.858877 -5.468749
```

The last line in the results is specific to an object of class `opChooserOption`. It represents the sensitivity of the price of a chooser option to the time until the choice between a put and a call has to be made.

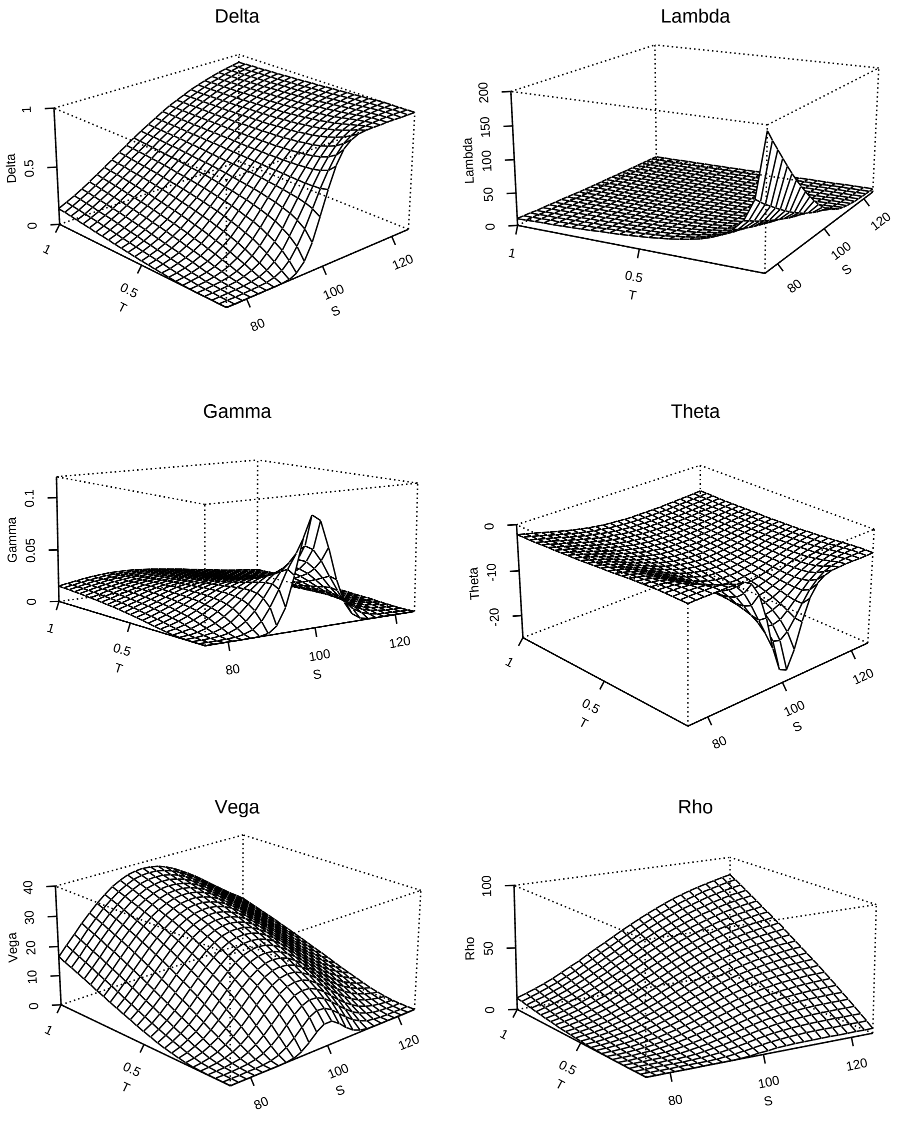

Figure 6: Greeks of a European call option in 3D as functions of time to maturity and spot price of underlying asset. The Appendix gives the code for generating the plot.

2.4.5 Vega (ν)

Vega is not a Greek letter, but ν is used to represent it sometimes. Vega measures the sensitivity of the option price to a change in the volatility of the underlying asset. For European options it is defined as

$$\nu_{call} = \nu_{put} = \frac{\partial c}{\partial \sigma} = \frac{\partial p}{\partial \sigma} = S_0 \sqrt{T} e^{(b-r)T} \Phi'(d_1)$$

Although the Black-Scholes-Merton model assumes constant volatility of the price of the underlying asset, in practice volatility changes over time. Higher volatility means higher uncertainty; therefore, it results in higher option value.

Example 10 *Vega*

The `opVegaBSM` function requires an object of class `opEuOption`, and returns a numeric object representing the Vega of the option. The `opVega` requires an option object that is identified by its class, and returns a numeric object representing the Vega of the option. The option objects `opt.obj.BSM` and `opt.obj.NUM` were defined in Example 5 and 6, respectively.

```
> opVegaBSM(opt.obj.BSM)
[1] 27.35866 25.75748
> opVega(opt.obj.BSM)
$sigma:
[1] 27.35866 25.75748

> opVega(opt.obj.NUM)
$sigma:
[1] 54.67123 66.91979
```

2.4.6 Rho (ρ)

Rho measures the sensitivity of the option price to a change in the risk free interest rate. For European options, it is defined as

$$\rho_{call} = \frac{\partial c}{\partial r} = KTe^{-rT}\Phi(d_2)$$
$$\rho_{put} = \frac{\partial p}{\partial r} = -KTe^{-rT}\Phi(-d_2)$$

when $b \neq 0$, and

$$\rho_{call} = \frac{\partial c}{\partial r} = -cT$$
$$\rho_{put} = \frac{\partial p}{\partial r} = -pT$$

when $b = 0$, e.g. when the underlying asset is a futures contract.

Example 11 *Rho*

The `opRhoBSM` function requires an object of class `opEuOption`, and returns a numeric object representing the Rho of the option. The `opRho`[3] requires an option object that is identified by its class, and returns a numeric object representing the Rho of the option. The option objects `opt.obj.BSM` and `opt.obj.NUM` were defined in Example 5 and 6, respectively.

```
> opRhoBSM(opt.obj.BSM)
[1] 26.44236 15.29113
> opRho(opt.obj.BSM)
$intRate:
[1] -3.444364 -1.453236

> opRho(opt.obj.NUM)
$intRate:
[1] -12.37848 -14.41853
```

2.4.7 Sensitivity to the Cost of Carry

The sensitivity of a European option to a marginal change in the cost of carry rate is defined as

$$CC_{call} = \frac{\partial c}{\partial b} = S_0 T e^{(b-r)T}\Phi(d_1)$$
$$CC_{put} = \frac{\partial p}{\partial b} = -S_0 T e^{(b-r)T}\Phi(-d_1)$$

Example 12 *Sensitivity to Cost of Carry*

The `opSenCostCarryBSM` function requires an object of class `opEuOption`, and returns a numeric object representing the sensitivity to the cost of carry of the option. The `opSenCostCarry` requires an option object that is identified by its class, and returns a numeric object representing the sensitivity to the cost of carry of the option. The option objects `opt.obj.BSM` and `opt.obj.NUM` were defined in Example 5 and 6, respectively.

```
> opSenCostCarryBSM(opt.obj.BSM)
[1] 29.88672 16.74437
> opSenCostCarry(opt.obj.BSM)
$costCarry:
[1] 29.88672 16.74437

> opSenCostCarry(opt.obj.NUM)
$costCarry:
[1]  34.56709 -14.89349
```

[3]As of this writing `opRho` has a bug: it gives a result that is different from `opRhoBSM`

2.5 Valuation of American Options in Continuous Time: Analytical Pricing Formulas

The key issues to consider when pricing American options are

- The posibility of early exercise.
- The payment of dividends by the underlying asset during the options life.

If there is no dividend paid during the life of the option, there is no incentive to exercise the option early, and the option might be priced as a European one. The functions based on closed form approximations for the pricing of American options in S+FinMetrics are `bsmCallOneDivAM`, `bsmAMBAW`, `bsmAMBS`. Their use is illustrated in the examples below.

2.5.1 American Calls on Stocks with Known Dividend

American options are exercised just before the ex-dividend date if the benefit from the dividend to be paid is greater than that from waiting until expiry and losing the dividend. Roll (1977), Geske (1979) and Whaley (1982) developed an adapted version of the Black Scholes model for the valuation of an American call option on a stock paying a single dividend of D, with time to dividend payout t.

$$\begin{aligned} C = {} & S_0[\Phi(b_1) + \Phi(a_1, -b_1, -\sqrt{\frac{t}{T}})] + De^{-rT}\Phi(b_2) \\ & - Ke^{-rT}[e^{r(T-t)}\Phi(b_2) + \Phi(a_2, -b_2, -\sqrt{\frac{t}{T}}) - (K-D)e^{-rt}] \end{aligned}$$

where

$$a_1 = \frac{ln[(S_0)/K] + (r + \sigma^2/2)T}{\sigma\sqrt{T}}\ , \quad a_2 = a_1 - \sigma\sqrt{T}$$
$$b_1 = \frac{ln[(S_0)/S_{xd}] + (r + \sigma^2/2)t}{\sigma\sqrt{t}}\ , \quad b_2 = b_1 - \sigma\sqrt{t}$$

and $\Phi(a, b, \rho)$ is the cumulative bivariate normal distribution function with upper integral limits a and b and correlation coefficient ρ. S_{xd} is the critical ex-dividend stock price that satisfies

$$c(S_{xd}, K, T-t) = S_{xd} + D - K$$

where $c(S_{xd}, K, T)$ is the value of the European call with stock price S_{xd} and time to maturity T.

Example 13 *Known Cash Dividend Payment*

Consider an American call option with six months to expiry. The underlying stock price is \$100, the strike price is \$100, the risk free interest rate is 5% per year, a dividend of \$4 will be paid in three months, and the volatility is 20% per year.

The `bsmCallOneDivAM` function computes the value of an option with a single cash dividend payment. In addition to the parameters of the `bsmEU` function described in Example 1, the `bsmCallOneDivAM` function requires the following input argument:

`divCash`: A series object representing one cash dividend per American option to be priced. In this example, the `signalSeries` function constructs a `signalSeries` object from positions and data. This object contains the information that a dividend amount of $4 will be paid in three months.

The `bsmCallOneDivAM` function returns an object of class `opAmOption` which is a list specifying the spot, strike, time to maturity, interest rate, sigma, cost carry, type, price and discrete cash dividends of options to be priced.

```
> bsmCallOneDivAM(spot = 100, strike = 100, time = 0.5,
+    intRate = 0.05, sigma = 0.2,
+    divCash = signalSeries(4, pos = 0.25))

 1 American call(s)

     spot strike maturity interest.rate sigma cost.carry price
[1,]  100    100      0.5          0.05   0.2       0.05 5.208

Discrete Cash Dividend series --
     t=0.25
[1,]      4
```

2.5.2 Barone-Adesi and Whaley Approximation

The quadratic approximation method by Barone-Adesi and Whaley (1987) can be used to price American call and put options on an underlying asset with cost-of-carry rate b. When $b \geq r$, the American call value is equal to the European call value, and can be found by using the generalized Black-Scholes-Merton formula. The Barone-Adesi and Whaley formulas are

$$\begin{aligned} C(S,K,T) &= \begin{cases} c_{BSM}(S,K,T) + A_1(S/S^*)^{q_1} & \text{when } S < S^* \\ S - K & \text{when } S \geq S^* \end{cases} \\ P(S,K,T) &= \begin{cases} p_{BSM}(S,K,T) + A_2(S/S^{**})^{q_2} & \text{when } S > S^{**} \\ K - S & \text{when } S \leq S^{**} \end{cases} \end{aligned}$$

where $c_{BSM}(S,K,T)$ and $p_{BSM}(S,K,T)$ are the general Black-Scholes-Merton call and put formula, respectively, and

$$A_1 = \frac{S^*}{q1}[1 - e^{(b-r)T}\Phi(d_1(S^*))]$$

$$A_2 = -\frac{S^{**}}{q2}[1 - e^{(b-r)T}\Phi(-d_1(S^{**}))]$$

$$d_1(S) = \frac{\ln(S/K) + (b + \sigma^2/2)T}{\sigma\sqrt{T}}$$
$$q_{1,2}(S) = \frac{-(N-1) \pm \sqrt{(N-1)^2 + 4M/Q}}{2}$$
$$M = 2r/\sigma^2 \ , \quad N = 2b/\sigma^2 \ , \quad Q = 1 - e^{-rT}$$

where S^* and S^{**} are the critical asset prices for the call and put options, respectively, that satisfy

$$S^* - K = c(S^*, K, T) + [1 - e^{(b-r)T}\Phi(d_1(S^*))]S^*\frac{1}{q_1}$$

$$K - S^{**} = p(S^{**}, K, T) - [1 - e^{(b-r)T}\Phi(-d_1(S^{**}))]S^{**}\frac{1}{q_2}$$

These equations can be solved by using the Newton-Raphson algorithm.

Example 14 *Barone-Adesi and Whaley Approximation*

Consider an American call option with six months to expiry. The underlying stock price is \$100, the strike price is \$100, the risk-free interest rate is 5% per year, the dividend yield is 8% per year, and the volatility is 20% per year.

The `bsmAMBAW` function computes the price of Amercian options on an underlying asset with continuous dividend payout using the Barone-Adesi and Whaley (BAW) approximation technique. The `bsmAMBAW` function requires the same parameters as the `bsmEU` function described in Example 1.

```
> bsmAMBAW(spot = 100, strike = 100, time = 0.5,
+    intRate = 0.05, costCarry = 0.05-0.08, sigma = 0.2,
+    type = "call")

 1 American call(s)

     spot strike maturity interest.rate sigma cost.carry price
[1,]  100    100      0.5          0.05   0.2      -0.03 4.933

Valuation method --  Barone-Adesi&Whaley87
```

The `bsmAMBAW` function returns an object of class `opAmOption` described in Example 13.

2.5.3 Bjerksund and Stensland Approximation

The Bjerksund and Stensland (1993) approximation can be used to price American options on stocks, futures and currencies. It is based on an exercise strategy corresponding on a flat boundary I (trigger price). Numerical investigation indicates that this model is somewhat more accurate for long-term options than the Barone-Adesi and Whaley model presented above.

$$C = \alpha S^{\beta} - \alpha\phi(S,T,\beta,I,I) + \phi(S,T,1,I,I)$$
$$- \phi(S,T,1,X,I) - X\phi(S,T,0,I,I) + X\phi(S,T,0,X,I)$$

where

$$\alpha = (I - X)I^{-\beta}$$
$$\beta = \left(\frac{1}{2} - \frac{b}{\sigma^2}\right) + \sqrt{\left(\frac{b}{\sigma^2} - \frac{1}{2}\right)^2 + 2\frac{r}{\sigma^2}}$$

The function $\phi(S,T,\gamma,H,I)$ is given by

$$\phi(S,T,\gamma,H,I) = e^{\lambda}S^{\gamma}\left[\Phi(d) - \left(\frac{I}{S}\right)^{\kappa}\Phi\left(d - \frac{2\ln(I/S)}{\sigma\sqrt{T}}\right)\right]$$
$$\lambda = \left[-r + \gamma b + \frac{1}{2}\gamma(\gamma-1)\sigma^2\right]T$$
$$d = -\frac{\ln(S/H) + [b + (\gamma - 1/2)\sigma^2]T}{\sigma\sqrt{T}}$$
$$\kappa = \frac{2b}{\sigma^2} + (2\gamma - 1)$$

and the trigger price is defined as

$$I = B_0 + (B_\infty - B_0)(1 - e^{h(T)})$$
$$h(T) = -(bT + 2\sigma\sqrt{T})\left(\frac{B_0}{B_\infty - B_0}\right)$$
$$B_\infty = \frac{\beta}{\beta - 1}X$$
$$B_0 = \max\left[X, \left(\frac{r}{r-b}\right)X\right]$$

If $S \geq I$, it is optimal to exercise the option immediately, and the value must be equal to the intrinsic value $S - X$. On the other hand, if $b \geq r$, it will never be optimal to exercise the American call option before expiration, and the value can be found using the generalized Black-Scholes formula. The value of the American put is given by the Bjerksund and Stensland put-call transformation

$$P(S,X,T,r,b,\sigma) = C(X,S,T,r-b,-b,\sigma)$$

where $C(\cdot)$ is the value of the American call with risk free rate $r-b$ and drift $-b$. With the use of this transformation, it is not necessary to develop a separate formula for an American put option.

Example 15 *Bjerksund and Stensland Approximation*

Consider the same American call option as above with six months to expiry. The underlying stock price is \$100, the strike price is \$100, the risk-free interest rate is 5% per year, the dividend yield is 8% per year, and the volatility is 20% per year.

The bsmAMBS function computes the price of Amercian options on an underlying asset with continuous dividend payout using the Bjerksund and Stensland (BS) approximation technique. The bsmAMBS function requires the same parameters as the bsmEU function described in Example 1.

```
> bsmAMBS(spot = 100, strike = 100, time = 0.5,
+    intRate = 0.05, costCarry = 0.05-0.08, sigma = 0.2,
+    type = "call")

 1 American call(s)

     spot strike maturity interest.rate sigma cost.carry price
[1,]  100    100      0.5          0.05   0.2      -0.03 4.875

Valuation method --  Bjerksund&Stensland93
```

The bsmAMBS function returns an object of class opAmOption described in Example 13.

2.6 Monte Carlo Simulation

Monte Carlo simulation[4] is a numerical method that is useful in many situations when no closed form solution is available. It is based on random sampling of the underlying asset price paths. The value of an option is derived using the risk neutral valuation result. That is, the expected payoff in a risk neutral world is discounted at the risk free interest rate. The steps of the algorithm are:

1. Sample a random path for S in a risk neutral world. For example, for lognormally distributed asset prices construct a path for S (see section 2.3.3)

$$S_{t+\delta t} = S_t \exp\left[\left(\mu - \frac{\sigma^2}{2}\right)\delta t + \sigma\epsilon\sqrt{\delta t}\right] \;, \quad \text{where} \quad \epsilon \sim N(0,1) \qquad (10)$$

   One simulation trial involves constructing a complete path for S using N random samples from the standard normal distribution, where N is the number of intervals of length δt during the life of the option. In the case of lognormally distributed asset prices, equation (10) is true for all δt, and so it follows that

$$S_T = S_0 \exp\left[\left(\mu - \frac{\sigma^2}{2}\right)T + \sigma\epsilon\sqrt{T}\right] \;, \quad \text{where} \quad \epsilon \sim N(0,1)$$

2. Calculate the payoff from the derivative.

3. Repeat steps 1 and 2 to get many sample values of the payoff from the derivative in a risk neutral world. A minimum of 10,000 simulations are typically necessary to price an option with satisfactory accuracy.

4. Calculate the mean of the sample payoffs to get an estimate of the expected payoff in a risk neutral world.

5. Discount the expected payoff at the risk free rate to get an estimate of the value of the derivative. For example, in the case of vanilla European options, the payoff depends on the final value of the asset. The value of a European call and put option is given by:

$$c = \frac{e^{-rT}}{n}\sum_{i=1}^{n} max[Se^{(b-\sigma^2/2)T+\sigma\epsilon_i\sqrt{T}} - K, 0]$$
$$p = \frac{e^{-rT}}{n}\sum_{i=1}^{n} max[K - Se^{(b-\sigma^2/2)T+\sigma\epsilon_i\sqrt{T}}, 0] \;, \quad \text{where} \quad \epsilon_i \sim N(0,1)$$

The key advantage of Monte Carlo simulation is that it can be used when the payoff depends on the path followed by the underlying variable S as well as

[4]For a detailed exposition of Monte Carlo methods in finance, the reader is referred to Jaeckel (2002) and Glasserman (2003)

when it depends only on the final value of$S$. As apparent in step 3 above, the main drawback of Monte Carlo simulation is that it is computer intensive.

S+FinMetrics offers two Monte Carlo simulation based option pricing functions: the cevEUMC function based on the Generalized Constant Elasticity of Variance (CEV) Model and the svolEUMC function based on the Continuous Stochastic Volatility Model.

2.6.1 Variance Reduction Procedures

As described above, a very large number of trials is usually necessary to get reasonably accurate estimates of the option prices. S+FinMetrics implements two variance reduction techniques that speed up the estimation process and make the results more accurate:

Anthetic Variable Technique: involves calculating two values of the derivative. First, the path is simulated as described in step 1 above, and then a mirror path is generated by switching the sign of the random variables on the original path. (If ϵ is the value of the random variable used to calculate the first payoff, then $-\epsilon$ is the corresponding random variable used to calculate the second one. This technique reduces the number of simulated random paths and the variance of the estimated option price by half.

Control Variate Technique: [5]

2.6.2 The Constant Elasticity of Variance Model

The Black-Scholes-Merton formula assumes that volatility is constant for the duration of the option contract. I also assumer that the asset price follows a Geometric Brownian Motion which implies that asset returns are normally distributed. However, empirical evidence from stock markets shows that the volatility is often negatively correlated with the stock price level. That is, volatility tends to increase as the stock price decreases.

The Constant Elasticity of Variance model, developed by Cox (1975) and Cox and Ross (1976), is an adjustment that enables modeling the possibility that the volatility of the underlying asset could be dependent upon the price of the underlying asset:

$$dS = \mu S\ dt + \sigma S^{\beta}\ dW$$

where μ is the expected rate of return on the asset, σ is the instantaneous volatility of the asset price, β is the elasticity parameter, and W follows a Wiener process. $\beta = 1$ yields the BSM stock price model.

Example 16 *European Option pricing with Monte Carlo Simulation under the Generalized Constant Elasticity of Variance Model*

[5]to be completed

Consider a European call option with six months to expiry. The underlying stock price is \$100, and the strike price is \$100 and \$110, respectively. The risk-free interest rate is 5% per year, the dividend yield is 8% per year, and the volatility is 20% per year.

The `cevEUMC` function computes the European option price with Monte Carlo Simulation under the Constant Elasticity of Variance model. In addition to the parameters of the `bsmEU` function described in Example 1, the `cevEUMC` function requires the following input arguments:

`beta`: A numeric object representing the coefficient of elasticity. If `beta = 0`, the CEV model becomes the standard geometric Brownian motion model. When `beta` < 0, the model creates a probability distribution with a heavy left tail and a less heavy right tail. When `beta` > 0, the model produces a probability distribution with a heavy right tail and a less heavy left tail. Defaults to -0.5 which denotes the standard square root CIR process.

`method`: A character string representing the method to be used in generating sample paths for the Monte Carlo simulation. The only method supported is the `psuedo` method, in which the sample path generated will be from a standard normal distribution.

`seed`: A numeric value representing the random seed to be used for the Monte Carlo simulation. The random seed is used to initialize a pseudo-random number generator.

`antithetic`: A logical flag either `T` or `F` to denote whether an antithetic variate technique is to be used for variance reduction purposes. As default, the antithetic variable technique is implemented.

`control.variate`: A character string representing whether the control variate technique is to be implemented. If `none`, the control variate technique is not used for variance reduction purposes. To implement the control variate method using the delta or delta-gamma approximation, set to either `delta` or `delta&gamma` (default) respectively.

`steps`: A numeric value specifying the number of steps to be used in the discretization of a one year period. Defaults to 250 steps.

`nSim`: A numeric value representing the number of Monte Carlo simulations. Defaults to 10000.

The `cevEUMC` function returns a list specifying the spot, strike, maturity, interest rate, sigma, cost of carry, and the computed European option price.

```
> cevEUMC(spot = 100, strike = c(100, 110), time = 0.5,
+    intRate = 0.05, sigma = 0.2, costCarry = 0.05-0.08,
+    beta = -0.5, type = "call", method = "pseudo",
+    seed = NULL, antithetic = T,
+    control.variate = "delta&gamma", steps = 250,
+    nSim = 10000)

 2 European call(s)
```

```
     spot strike maturity interest.rate sigma cost.carry price
[1,]  100    100      0.5          0.05   0.2      -0.03 4.691
[2,]  100    110      0.5          0.05   0.2      -0.03 1.763

Valuation method --  Monte-Carlo pseudo
```

2.6.3 The Continuous Stochastic Volatility Model

[6]

Example 17 *European Option pricing with Monte Carlo Simulation under the Continuous Stochastic Volatility Model*

Consider two European call options with six months to expiry. The underlying stock price is $100, and the strike price is $100 and $110, respectively. The risk-free interest rate is 5% per year, the dividend yield is 8% per year, and the volatility is 20% per year.

The `svolEUMC` function computes the European option price with Monte Carlo Simulation under the continuous stochastic volatility models of Cox-Intersoll-Ross or Log-Ornstein-Uhlenbeck. In addition to the parameters of the `bsmEU` function described in Example 1, the `cevEUMC` function requires the following input arguments:

`method`: A character string representing the method to be used in generating sample paths for the Monte Carlo simulation. The only method supported is the `psuedo` method, in which the sample path generated will be from a standard normal distribution.

`seed`: A numeric value representing the random seed to be used for the Monte Carlo simulation. The random seed is used to initialize a pseudo-random number generator.

`antithetic`: A logical flag either `T` or `F` to denote whether an antithetic variate technique is to be used for variance reduction purposes. As default, the antithetic variable technique is implemented.

`control.variate`: A character string representing whether the control variate technique is to be implemented. If `none`, the control variate technique is not used for variance reduction purposes. Else, to implement the control variate method using the delta or delta-gamma approximation, set to either `delta` or `delta&gamma` (default) respectively.

`steps`: A numeric value specifying the number of steps to be used in the discretization of a one year period. Defaults to 250 steps.

`nSim`: A numeric value representing the number of Monte Carlo simulations. Defaults to 10000.

`svol`: A character string representing the stochastic volatility model to be implented in the Monte Carlo simulation. If `none` (default), the stochastic volatility model will not be implemented and the risk will be quantified

[6] to be completed

by a constant volatility parameter. Otherwise, a continuous stochastic volatility model will be implemented and it can be either one of the Cox-Ingersoll-Ross method (`cir`) or the Log-Ornstein-Uhlenbeck method (`log`).

`param`: A vector of parameters for the implementation of the stochastic volatility model. If the Cox-Ingersoll-Ross (`cir`) method is implemented, then a positive value for `kappa`, `theta`, `sigma` and `rho` must be specified for a positive stochastic volatility process. Else if the `log` method is chosen, `kappa`, `sigma` and `theta` must be specified, where `kappa` and `sigma` are positive and the exponential value of `theta` must also be positive for a positive stochastic volatility process. Defaults to `param = c(kappa = 1.0, theta = 0.15, sigma = 0.05, rho = 0.5)`.

The `svolEUMC` function returns a list specifying the spot, strike, time to maturity, interest rate, sigma, cost-of-carry, and the computed European option price.

```
> svolEUMC(spot = 100, strike = c(100, 110), time = 0.5,
+    intRate = 0.05, sigma = 0.2, costCarry = 0.05-0.08,
+    type = "call", method = "pseudo", seed = NULL,
+    antithetic = T, control.variate = "none", svol = "none",
+    param = c(kappa = 1., theta = 0.15, sigma = 0.05, rho = 0.5),
+    steps = 52, nSim = 10000)

 2 European call(s)

     spot strike maturity interest.rate sigma cost.carry price
[1,]  100    100      0.5          0.05   0.2      -0.03 4.812
[2,]  100    110      0.5          0.05   0.2      -0.03 1.778

Valuation method --  Monte-Carlo pseudo
```

The Monte Carlo simulation based option prices can be compared to the option prices based on the analytical Black-Scholes-Merton formulas. The corresponding output of the `bsmEU` function is displayed below:

```
> bsmEU(spot = 100, strike = c(100, 110), time = 0.5,
+    intRate = 0.05, costCarry = 0.05-0.08, sigma = 0.2, type = "call")

 2 European call(s)

     spot strike maturity interest.rate sigma cost.carry price
[1,]  100    100      0.5          0.05   0.2      -0.03 4.762
[2,]  100    110      0.5          0.05   0.2      -0.03 1.785

Valuation method --  black-scholes
```

3 Valuation of Exotic Options

Exotic options differ from vanilla options in at least one of the standard contract terms. Those are either variations on the payoff profiles of the plain vanilla options, or they have further optionality embedded in them. The value of an exotic option is a function of "exotic" contract terms. Depending on the type of deviation form the traditional structure, exotic options can be classified into the following categories:

- Options with contract variations
- Path-dependent options
- Limit-dependent options
- Multi-factor options

3.1 Options with Contract Variations

3.1.1 Binary Options

The payoff from a binary option is either some fixed amount of some asset if it is in-the-money, or nothing at all if it is out-of-money. The payoff does not depend on how far in-the-money the option is. Two types of binary options are

- Cash-or-Nothing, which pays some fixed amount of cash if the option expires in-the-money. (See Reiner and Rubinstein (1991b)).
- Asset-or-Nothing, which pays the value of the underlying security at the expiration date. (See Cox and Rubinstein (1985)).

Thus, the options are binary in nature because there are only two possible outcomes. They are also called all-or-nothing options or digital options.

Example 18 *Cash or Nothing*

Consider two cash-or-nothing European call options paying \$10 and \$20, respectively, if they end up in-the-money with six months to expiration. The underlying asset price is \$100, and the strike price is \$100 and \$110, respectively. The dividend yield is 8% per year, the risk free interest rate is 5% per year, and the volatility is 20% per year.

The `bsmBinaryEU` function computes the price of European cash-or-nothing and asset-or-nothing binary options using Cox and Rubinstein formulas (1985). In addition to the parameters of the `bsmEU` function described in Example 1, the `bsmBinaryEU` function requires the following input argument:

`payoff:` A numeric object representing a cash amount paid at expiry if the cash-or-nothing option ends up in-the-money or a character string "asset" if an asset-or-nothing option is being priced. The length of the object must be equal to the number of options to be priced unless one payoff value is to be used for all options.

The `bsmBinaryEU` function returns an object of class `opBinaryOption` which is a list specifying the type, spot, strike, time to maturity, interest rate, sigma, cost carry, and computed option price.

```
> bsmBinaryEU(spot = 100, strike = c(100, 110),
+    payoff = c(10, 20), intRate = 0.05, time = 0.5,
+    costCarry = 0.05-0.08, sigma = 0.2, type = "call")

 2   Binary European call(s)

     spot strike maturity interest.rate sigma cost.carry price
[1,]  100    100      0.5          0.05   0.2      -0.03 4.192
[2,]  100    110      0.5          0.05   0.2      -0.03 3.852

Payoff --
10 20
```

Example 19 *Asset or Nothing*

Consider two asset-or-nothing European call options with six months to expiration. The underlying asset price is $100, the strike price is $100 and $110, respectively. The dividend yield is 8% per year, the risk free interest rate is 5% per year, and the volatility is 20% per year.

To compute the price of the asset-or-nothing options, the `payoff` parameter has to be set to `"asset"` in the `bsmBinaryEU` function.

```
> bsmBinaryEU(spot = 100, strike = c(100, 110),
+    payoff = "asset", intRate = 0.05, time = 0.5,
+    costCarry = 0.05-0.08, sigma = 0.2, type = "call")

 2   Binary European call(s)

     spot strike maturity interest.rate sigma cost.carry price
[1,]  100    100      0.5          0.05   0.2      -0.03 46.68
[2,]  100    110      0.5          0.05   0.2      -0.03 22.97

Payoff --
asset
```

The counterpart of the `bsmBinaryEU` function for pricing American options is `bsmBinaryAM`[7]. It takes the same arguments and also returns an object of class `opBinaryOption`.

[7]As of this writing the `bsmBinaryAM` function has a bug: the `payoff` argument requires a numeric value for asset-or-nothing options.

3.1.2 Chooser Options

Chooser options allow the holder to choose whether their option is a call or a put at a particular date. (See Rubinstein (1991c)) Chooser options are usually more expensive than vanilla options due to the added flexibility. Two types of chooser options are:

- Simple Chooser, with same time to maturity and strike for the call and the put.
- Complex Chooser, with different time to maturity and/or strike price for the call and the put. The holder has the right to choose at time, t, if the option is to be a call with time to maturity T_c and strike price K_c, or a put with time to maturity T_p and strike price K_p. (See Rubinstein (1991c))

Chooser options provide the benefit of allowing hedging against both price increases and decreases without purchasing both a call and a put. The value of the choser option at the time of the choice is $\max(c, p)$, where c and p are the values of the underlying call and put options, respectively.

Example 20 *Simple Chooser*

Consider two simple chooser options with one year to expiration. The choice between a put and a call can be made in three and six months respectively. The underlying stock price is \$100, and the strike price is \$100 and \$110, respectively. The risk free interest rate is 5% per year, the dividend yield is 8% per year, and the volatility is 20% per year.

The `bsmSimpleChooserEU` function computes the price of simple chooser put and call options using the Rubinstein analytical formula (1991). The `bsmSimpleChooserEU` function requires the following input argument in addition to the parameters of the `bsmEU` function described in Example 1 (except `type`):

`t1`: The predetermined time after which the holder has the right to choose whether the option is to be a standard call or put option. The length of the object must be equal to the number of options to be priced unless one t1 value is to be used for all options.

The `bsmSimpleChooserEU` function returns an object of class `opChooserOption`, which is a list specifying the spot, strike, time to maturity, time to choose, interest rate, sigma, cost carry and computed option price.

```
> bsmSimpleChooserEU(spot = 100, strike = c(100, 110),
+    time = 1, intRate = 0.05, costCarry = 0.05,
+    sigma = 0.2, t1 = c(0.25, 0.5))

 2 Chooser option

                 [,1]   [,2]
          spot 100.00 100.00
```

```
        strike 100.00 110.00
      maturity   1.00   1.00
time.to.choose   0.25   0.50
 interest.rate   0.05   0.05
         sigma   0.20   0.20
    cost.carry   0.05   0.05
         price  12.38  14.42
```

Example 21 *Complex Chooser*

Consider a complex chooser option that gives the right to the holder to choose between a call with six months to expiration and strike price \$110, and a put with seven months to expiration and strike price \$90. The choice between the call and the put can be made in three months. The underlying stock price is \$100, the risk free interest rate is 5% per year, the dividend yield is 8% per year, and the volatility is 20% per year.

The `bsmComplexChooserEU` function computes the price of complex chooser put and call options using the method proposed by Rubinstein (1991). The `bsmSimpleChooserEU` function requires the following input arguments in addition to the parameters of the `bsmEU` function described in Example 1 (except `type`):

`Xc`: A numeric object representing the exercise price of the call option.
`Xp`: A numeric object representing the exercise price of the put option.
`Tc`: A numeric object representing the time to expiration of the standard call option expressed in years, if it were chosen.
`Tp`: A numeric object representing the time to expiration of the standard put option expressed in years, if it were chosen.
`epsilon`: (optional) A numeric object representing the Newton-Raphson critical value. Defaults to 0.001.

Similarly to the `bsmSimpleChooserEU` function, `bsmComplexChooserEU` also returns an object of class `opChooserOption`.

```
> bsmComplexChooserEU(spot = 100, Xc = 110, Xp = 90,
+    time = 0.25, Tc = 0.5, Tp = 7/12, intRate = 0.05,
+    costCarry = 0.05-0.08, sigma = 0.2, epsilon = 0.001)

 1 Chooser option

                    [,1]
          spot  100.0000
   call.strike  110.0000
    put.strike   90.0000
time.to.choose    0.2500
 call.maturity    0.5000
  put.maturity    0.5833
```

```
interest.rate    0.0500
        sigma    0.2000
   cost.carry   -0.0300
        price    3.8926
```

3.2 Path-Dependent Options

Asian Options

The payoff from an Asian option depends on the average of the prices of the underlying asset over a specified period of time, rather than on the price of the underlying asset on the expiration date. Two types of Asian options are:

- Average Price, where an in-the-money option pays the difference between the average price of the asset and the predefined strike price.
- Average Strike, where the strike price is set equal to the average price of the asset, and an in-the-money option pays the difference between this average asset price and the asset price on the option's maturity date.

The most common type uses the arithmetic average of prices recorded for the underlying asset during the life of the option which is then compared against the strike price when calculating the final payoff. (See Turnbull and Wakeman (1991), Levy (1992), Haug and Margabe (2003), Curran (1992)). For geometric average rate option, the valuation is similar to a vanilla option, because the assumption of lognormal distribution of asset price returns still holds. (See Kemna and Vorst (1990).)

Example 22 *Geometric Average Price Option*

Consider two geometric average rate put options with three months to expiration and strike price \$100 and \$110, respectively. The underlying asset price is \$100, the risk free interest rate is 5% per year, and the volatility is 20% per year.

The `bsmAsianGeomEU` function computes the price of asian geometric average rate options using the method introduced by Kemna and Vorst (1990). The `bsmAsianGeomEU` function requires the same input arguments as the `bsmEU` function described in Example 1. The `bsmAsianGeomEU` function returns an object of class `opAsianOption` which is a list specifying the spot, strike, time to maturity, interest rate, sigma, cost carry, type, and computed price of the Asian option.

```
> bsmAsianGeomEU(spot = 100, strike = c(100, 110), time = 0.5,
+    intRate = 0.05, costCarry = 0.05, sigma = 0.2,
+    type = "call")

 2 Asian call(s)
     spot strike maturity interest.rate sigma cost.carry  price
[1,]  100    100      0.5          0.05   0.2       0.05 3.7526
[2,]  100    110      0.5          0.05   0.2       0.05 0.6544
```

Example 23 *Arithmetic Average Price Option*

Consider two arithmetic average currency options with six months to expiration and strike price \$100 and \$110, respectively. The spot price is \$90, the observed average spot price is \$90, the domestic risk free interest rate is 5% per year, the foreign interest rate is 10% per year, and the volatility of the spot rate is 20% per year. `S+FinMetrics` offers two functions for asian arithmetic average-rate option pricing: `bsmAsianArithFixedEULevy` and `bsmAsianArithFixedEUTW`. This example is solved using the former one; that is, using Levy's approximation.

The `bsmAsianArithFixedEULevy` function requires the following input argument in addition to the parameters of the `bsmEU` function described in Example 1:

`spot.avg`: A numeric object representing the arithmetic average of the known asset price fixings.
`t2`: A numeric object representing the remaining time to expiration of the option, expressed in years.

Similarly to the `bsmAsianGeomEU` function, `bsmAsianArithFixedEULevy` also returns an object of class `opAsianOption`.

```
> bsmAsianArithFixedEULevy(spot = 90, spot.avg = 90,
+    strike = c(100, 110), time = 0.5, t2 = 0.5,
+    intRate = 0.05, costCarry = 0.05-0.10, sigma = 0.2,
+    type = "call")

 2 Asian call(s)
     spot strike maturity interest.rate sigma cost.carry  price
[1,]   90    100      0.5          0.05   0.2      -0.05 0.2476
[2,]   90    110      0.5          0.05   0.2      -0.05 0.0109

Remaining time to maturity --  0.5
Observed arithmetic average price so far --  90
```

3.2.1 Lookback Options

The payoff from a lookback option depends on the maximum or minimum asset price over the life of the option. The holder can "look back" and choose the most advantageous price that was recorded by the asset during the lookback period. Two types of lookback options are:

- Floating, where the strike price is determined as the minimum value (for a call) or maximum value (for a put) of the underlying asset observed over the life of the option. (See Goldman, Sosin, and Gatto (1979), and Garman (1989).)
- Fixed, where the strike price is predetermined at inception, and the payoff for an in-the-money call is the difference between the maximum asset price

over the life of the option, S_{max}, and the strike price K. For an in-the-money put it is the difference between the minimum asset price over the life of the option, S_{min}, and the strike price K. (See Conze and Viswanathan (1991).)

The lookback option gives the holder the right to buy an asset at its lowest price or sell it at its highest price attained over the life of the option. The holder of a lookback option can never miss the best underlying asset price, and therefore a lookback option can never be out-of-the money. `S+FinMetrics` offers the functions listed in Table 3 for lookback option pricing.

Name	**Description**
`bsmLookbackExtremeSpreadEU`	Extreme Spread Option Pricing
`bsmLookbackFixedEU`	Fixed-Strike Lookback Option Pricing
`bsmLookbackFloatEU`	Floating-Strike Lookback Option Pricing
`bsmLookbackPartialFixedEU`	Partial-Time Fixed-Strike Lookback Option Pricing
`bsmLookbackPartialFloatEU`	Partial-Time Floating-Strike Lookback Option Pricing

Table 3: List of `S+FinMetrics` functions for pricing lookback options.

Example 24 *Floating Lookback Option*

Consider a floating lookback call option with six months to expiration, which gives the right to buy the underlying stock index at its lowest price. The underlying stock price is currently \$100, the so far observed minimum stock price is \$80, the risk-free interest rate is 5% per year, the dividend yield is 8% per year, and the volatility is 20% per year.

The `bsmLookbackFloatEU` function requires the following input argument in addition to the parameters of the `bsmEU` function described in Example 1:

`sMinOrMax`: A numeric object representing the lowest price observed during the lifetime of the underlying asset in the case of the floating-strike lookback call or the highest price observed for the floating-strike lookback put option.

The `bsmLookbackFloatEU` function returns an object of class `opLookbackOption`, which is a list specifying the type, spot, minimum or maximum price of the underlying, time to maturity, interest rate, cost of carry, sigma, and computed option price.

```
> bsmLookbackFloatEU(spot = 100, sMinOrMax = 80, time = 0.5,
+    intRate = 0.05, costCarry = 0.05-0.08, sigma = 0.3,
+    type = "call")
```

```
 1 Floating Strike Lookback call(s)

     spot min/max maturity interest.rate cost.carry sigma price
[1,]  100      80      0.5          0.05      -0.03   0.3 20.98
```

3.3 Limit-Dependent Options

3.3.1 Barrier Options

Barrier options are extinguished or activated when the underlying asset price reaches a predetermined barrier. Four types of barrier options are:

- Down and Out, where the option is canceled or knocked-out if the asset falls to a predetermined boundary price.
- Down and In, where the option is activated or knocked-in if the asset falls to a predetermined boundary price.
- Up and Out, where the option is canceled or knocked-out if the asset rises to a predetermined boundary price.
- Up and In, where the option is activated or knocked-in if the asset rises to a predetermined boundary price.

The premium for barrier options is lower than standard options, because the barrier option has value within a smaller price range than the standard option. (See Merton (1973, Reiner and Rubinstein (1991a), and Rich (1994).)

Similar to the put-call parity for vanilla options, there exists an in-out parity for barrier options. If we combine one *Down and In* and one *Down and Out* call option with the same strike price and expiration date, we get the price of a vanilla call option: $c = c_{in} + c_{out}$. Simultaneously holding the "in" and the "out" option guarantees that one and only one of the two will pay off.

Barrier options are sometimes accompanied by a rebate, which is a payoff to the option holder in case of a barrier event. Rebates can be paid either at the time of the event or at expiration. For a discrete barrier option, barrier events are checked for at discrete times, rather than continuously. `S+FinMetrics` offers the functions listed in Table 3 for lookback option pricing.

Example 25 *Double Barrier Option*

Consider two double knock out call options with six months to expiration, lower boundary \$80 and upper boundary \$120. The underlying stock price is \$100, the strike price is \$100 and \$110, respectively for the two options. The risk free interest rate is 5% per year, the dividend yield is 8% per year, and the volatility is 20% per year. The factors determining the curvature of the boundaries are 0; that is, the boundaries are flat.

The `bsmBarrierDoubleEU` function requires the following input arguments in addition to the parameters of the `bsmEU` function described in Example 1 (except `type`):

Name	Description
bsmBarrier	European and American Barrier Option Pricing
bsmBarrierBinom	European and American Barrier Option Pricing with a Binomial Tree Approximation
bsmBarrierDoubleEU	European Double Barrier Option Pricing
bsmBarrierLookEU	Look-Barrier Option Pricing
bsmBarrierPartialEU	Partial-Time Barrier Option Pricing
bsmBarrierPartialTwoAssetEU	Partial-Time Two-Asset Barrier Option Pricing
bsmBarrierSoftEU	Soft-Barrier Option Pricing
bsmBarrierTwoAssetEU	Two-Asset Barrier Option Pricing
opBarrierAdj	Adjustment of Barrier Option Prices to Allow for Discrete Monitoring

Table 4: List of S+FinMetrics functions for pricing barrier options.

`typeFlag`: A character string representing the type of the Double Barrier Option. `co` denotes an up-and-out-down-and-out call, `ci` denotes an up-and-in-down-and-in call, `po` denotes an up-and-out-down-and-out put, and `ci` denotes an up-and-in-down-and-in put double barrier option. All options to be priced must be of the same type.

`L`: A numeric object representing the lower boundary to be hit by the double barrier option.

`U`: A numeric object representing the upper boundary to be hit by the double barrier option.

`delta1, delta2`: Numeric objects that determine the curvature of the lower and upper boundary values of the double barrier option. If `delta1 = delta2 = 0` (default), this corresponds to two flat boundaries.

`H.dt`: A numeric object representing the time between monitoring instants defined as `H.dt` $= T/m$, where T is the time to maturity of the barrier option, and m is the number of monitored points. Defaults to `H.dt = 0`, which denotes that the barrier is monitored in continuous time. This value is used for the continuity correction for discrete barrier options pricing formulas.

The `bsmBarrierDoubleEU` function returns an object of class `opBarrierOption`, which is a list of spot, strike, upper and lower barriers, time to maturity, interest rate, cost of carry, sigma, type and computed option price.

```
> bsmBarrierDoubleEU(typeFlag = "co", spot = 100,
+    strike = c(100, 110), L = 80, U = 120,
+    time = 0.5, intRate = 0.05, costCarry = 0.05-0.08,
+    sigma = 0.2, delta1 = 0, delta2 = 0, H.dt = 0)
```

```
 2 Double Barrier options:
 --  up-and-out-down-and-out call

                   [,1]      [,2]
         spot  100.0000  100.0000
       strike  100.0000  110.0000
upper.barrier  120.0000  120.0000
lower.barrier   80.0000   80.0000
     maturity    0.5000    0.5000
interest.rate    0.0500    0.0500
   cost.carry   -0.0300   -0.0300
        sigma    0.2000    0.2000
        price    1.8074    0.2175

Lower barrier curvature --  0
Upper barrier curvature --  0

Discrete monitoring interval --  0
```

Example 26 *Soft Barrier Option*

Consider a soft down and out call option with six months to expiration, and with a soft-barrier range \$90 to \$80. This type of option is knocked out proportionately. For example, if the lowest asset price during the lifetime of the option is \$86, then 40% of the call is knocked out. The underlying stock price is \$100, the strike price is \$100, the risk free interest rate is 5% per year, the dividend yield is 8% per year, and the volatility is 20% per year.

The `bsmBarrierSoftEU` function requires the same input arguments as the `bsmBarrierDoubleEU` function, and also returns an object of class `opBarrierOption`.

```
> bsmBarrierSoftEU(typeFlag = "cdo", spot = 100,
+    strike = 100, L = 80, U = 90, time = 0.5,
+    intRate = 0.05, costCarry = 0.05-0.08, sigma = 0.2)

 1 Soft Barrier options:
 --  Down-and-out call

                   [,1]
         spot  100.0000
       strike  100.0000
upper.barrier   90.0000
lower.barrier   80.0000
     maturity    0.5000
interest.rate    0.0500
   cost.carry   -0.0300
        sigma    0.2000
        price    4.6703
```

3.4 Multi-Factor Options

3.4.1 Quanto Options

The payoff of a quanto option is determined in one asset, but its value is determined with respect to another underlying asset. One of the factors that have to be taken into account when evaluating quanto options is the correlation between the price volatility of the two assets.

Example 27 *Quanto Option*

Consider a fixed exchange rate foreign equity call option with six months to expiration. The underlying stock price is £100, the strike price is £100, the predetermined exchange rate is 1.5$/£, the domestic risk free interest rate is 5% per year, the foreign risk free interest rate is 4% per year, the dividend yield is 8% per year, the volatility of the foreign stock is 20% per year, the volatility of the foreign currency is 30% per year, and the correlation between the stock and the foreign currency is 0.3.

The `bsmQuantoEU` computes the price of currency-translated options. Itrequires the following input arguments in addition to the parameters of the `bsmEU` function described in Example 1:

`E`: A numeric object representing the spot exchange rate specified in units of the domestic currency per unit of the foreign currency.

`intRateForeign`: A numeric object representing the annualized foreign risk-free interest rate.

`divYield`: A numeric object representing the instantaneous proportional dividend payout rate of the underlying asset.

`sigmaE`: A numeric object representing the annualized asset price volatility of the domestic exchange rate.

`rho`: A numeric object representing the correlation of the returns for the two assets.

The `bsmQuantoEU` function returns an object of class `opFXTranslatedOption`, which is a list specifying the type, spot, exchange rate, strike, time to maturity, interest rates, dividend yield, sigmas, correlation and computed option price.

```
> bsmQuantoEU(spot = 100, E = 1.5, strike = 100,
+    time = 0.5, intRate = 0.05, intRateForeign = 0.04,
+    divYield = 0.08, sigma = 0.2, sigmaE = 0.3, rho = 0.3,
+    type = "call")

 1 quanto call(s)
fixed exchange-rate foreign equity options

                    [,1]
           spot  100.000
        fx.rate    1.500
         strike  100.000
```

```
          maturity   0.500
     interest.rate   0.050
 interest.rate.for   0.040
          divYield   0.080
        sigma.spot   0.200
          sigma.fx   0.300
       correlation   0.300
             price   6.208
```

4 Appendix

4.1 List of Option Pricing Functions in `S+FinMetrics`

Name	**Description**
`bsmAMBAW`	Barone-Adesi and Whaley Approximation of American Option Prices
`bsmAMBS`	Black-Scholes Approximation of American Option Prices
`bsmAMParity`	Parity for American options
`bsmAsianArithFixedEULevy`	Asian Arithmetic Average-Rate Option Pricing with Levy's Approximation
`bsmAsianArithFixedEUTW`	Asian Arithmetic Average-Rate Option Pricing with the Turnbull and Wakeman Approximation
`bsmAsianGeomEU`	Asian Geometric Average-Rate Option Pricing
`bsmBarrier`	European and American Barrier Option Pricing
`bsmBarrierBinom`	European and American Barrier Option Pricing with a Binomial Tree Approximation
`bsmBarrierDoubleEU`	European Double Barrier Option Pricing
`bsmBarrierLookEU`	Look-Barrier Option Pricing
`bsmBarrierPartialEU`	Partial-Time Barrier Option Pricing
`bsmBarrierPartialTwoAssetEU`	Partial-Time Two-Asset Barrier Option Pricing
`bsmBarrierSoftEU`	Soft-Barrier Option Pricing
`bsmBarrierTwoAssetEU`	Two-Asset Barrier Option Pricing
`bsmBermudanBinom`	Bermudan Option Pricing with a Binomial Tree Approximation
`bsmBinaryAM`	American Cash-or-Nothing and Asset-or-Nothing Binary Option Pricing
`bsmBinaryBarrierEU`	Binary Cash-Or-Nothing and Asset-Or-Nothing Barrier European Option Pricing
`bsmBinaryEU`	European Cash-or-Nothing and Asset-or-Nothing Binary Option Pricing
`bsmBinaryTwoAssetEU`	Two-Asset-Cash-or-Nothing Binary Option Pricing
`bsmCallOneDivAM`	American Call Option Pricing with One Cash Dividend
`bsmComplexChooserEU`	Complex Chooser Option Pricing
`bsmCompoundOptionEU`	Compound Option Pricing

Name	**Description**
`bsmCondivBinom`	European and American Option Pricing with the Binomial Tree Method
`bsmCondivFD`	European and American Option Pricing with the Finite Difference Method
`bsmCondivTrinom`	European and American Option Pricing with the Trinomial Tree Method
`bsmDiscDivAMBinom`	American Call and Put Option Pricing with Discrete Cash Dividends using a Binomial Tree Approximation
`bsmDualStrikeBinom`	European and American Dual Strike Option Pricing with a Binomial Tree Approximation
`bsmEqLinkedFXEU`	Equity Linked Foreign Exchange Option Pricing
`bsmEU`	European Option Pricing
`bsmEUParity`	Put-Call Parity for European Options
`bsmExchangeAM`	American Exchange Option Pricing
`bsmExchangeEU`	European Exchange Option Pricing
`bsmExecutiveEU`	Executive Stock Options Pricing
`bsmForEqInDomCurEU`	Foreign Equity Options Struck in Domestic Currency Option Pricing
`bsmForwardStartEU`	Forward Start Option Pricing
`bsmFuturesAM`	American Futures Option Pricing
`bsmFuturesEU`	European Futures Option Pricing
`bsmFXAM`	American Foreign Exchange Option Pricing
`bsmFXEU`	European Foreign Exchange Option Pricing
`bsmGapEU`	Gap Option Pricing
`bsmLookbackExtremeSpreadEU`	Extreme Spread Option Pricing
`bsmLookbackFixedEU`	Fixed-Strike Lookback Option Pricing
`bsmLookbackFloatEU`	Floating-Strike Lookback Option Pricing
`bsmLookbackPartialFixedEU`	Partial-Time Fixed-Strike Lookback Option Pricing
`bsmLookbackPartialFloatEU`	Partial-Time Floating-Strike Lookback Option Pricing
`bsmMaxMinBinom`	Option Pricing on the Minimum or the Maximum of Two Risky Assets with a Binomial Tree Approximation
`bsmMaxMinEU`	Option Pricing on the Minimum or the Maximum of Two Risky Assets
`bsmPerpetualAM`	American Perpetual Option Pricing
`bsmQuantoEU`	Quanto Option Pricing
`bsmRatchetEU`	Ratchet Option Pricing

Name	Description
`bsmSimpleChooserEU`	Simple Chooser Option Pricing
`bsmSpreadBinom`	European and American Spread Option Pricing with with a Binomial Tree Approximation
`bsmSpreadEU`	European Spread Option Pricing
`bsmStockAM`	American Stock Option Pricing
`bsmStockEU`	European Stock Option Pricing
`bsmSupershareEU`	Supershare Option Pricing
`bsmTimeSwitchEU`	Time-Switch Option Pricing
`bsmTwoAssetCorEU`	Two-Asset Correlation Option Pricing
`bsmWriterExtendEU`	Writer-Extendible Option Pricing
`cevEUMC`	European Option pricing with Monte Carlo Simulation under the Generalized Constant Elasticity of Variance (CEV) Model
`impTreeEUTrinom`	European Option Pricing with an Implied Trinomial Tree
`jdEU`	Jump-Diffusion Model for European Options
`opBarrierAdj`	Adjustment of Barrier Option Prices to Allow for Discrete Monitoring
`opBinomCashFlow`	European and American Option Pricing using a Binomial Cashflow Tree
`opBinomTree`	Binomial Tree
`opDelta`	Delta of Any Generic Options
`opDeltaBSM`	Delta of Black Scholes Option Prices
`opGamma`	Gamma of Any Generic Options
`opGammaBSM`	Gamma of Black Scholes Option Prices
`opImpVol`	Implied Volatility
`opLambda`	Lambda of Any Generic Options
`opLambdaBSM`	Lambda of Black Scholes Option Prices
`opRho`	Rho of Any Generic Options
`opRhoBSM`	Rho of Black Scholes Option Prices
`opSenCostCarry`	Cost-of-carry Sensitivity of Any Generic Options
`opSenCostCarryBSM`	Cost-of-carry Sensitivity of Black Scholes Option Prices
`opTheta`	Theta of Any Generic Options
`opThetaBSM`	Theta of Black Scholes Option Prices
`opVega`	Vega of Any Generic Options
`opVegaBSM`	Vega of Black Scholes Option Prices
`svolEUMC`	European Option Pricing with Monte Carlo Simulation

4.2 S-PLUS Code for Full Page Figures

```
################################
# Payoff Profiles 2D (Figure 1)
################################
> S.T <- seq(from = 0, to = 200, length = 50)
> call.bs <- bsmEU(spot = S.T, strike = 100, time = 0, intRate = 0.05,
+    costCarry = 0.05, sigma = 0.2, type = "call")$price
> put.bs <- bsmEU(spot = S.T, strike = 100, time = 0, intRate = 0.05,
+    costCarry = 0.05, sigma = 0.2, type = "put")$price
> payoff = list(call.bs, put.bs, -call.bs, -put.bs)
> titles = c("Long Call", "Long Put", "Short Call", "Short Put")
> pdf.graph(file = "payoffs.pdf", horizontal = T)
> par(mfrow=c(2,2))
> for (i in 1:4){
+    plot(S.T, payoff[[i]], xlab="Spot (Strike=100)", ylab="Payoff",
+       type="l")
+    title(main = titles[i])
+ }
> dev.off()

###################################
# Option Sensitivity 2D (Figure 2)
###################################
> nr.nodes <- 26
> spot.margin <- seq(from = 75, to = 125, length = nr.nodes)
> strike.margin <- seq(from = 75, to = 125, length = nr.nodes)
> time.margin <- seq(from = 0, to = 1, length = nr.nodes)
> intRate.margin <- seq(from = 0, to = 0.25, length = nr.nodes)
> costCarry.margin <- seq(from = 0, to = 0.25, length = nr.nodes)
> sigma.margin <- seq(from = 0, to = 0.25, length = nr.nodes)

> margin <- list(spot.margin, strike.margin, time.margin,
+    intRate.margin, costCarry.margin, sigma.margin)
> default.args <- list(spot = 100, strike = 100, time = 0.5,
+    intRate = 0.05, costCarry = 0.05, sigma = 0.2, type = "call")

> par(mfrow=c(3,2))
> for (i in 1:6){
+    sens.args <- default.args
+    sens.args[[i]] <- margin[[i]]
+    plot(margin[[i]], do.call("bsmEU", sens.args)$price,
+       xlab = names(default.args)[i], ylab = "Value of Call",
+       type = "l")
+    title(main = paste("Call Value vs.", names(default.args)[i]))
+ }
```

```
#######################
# Greeks 2D (Figure 5)
#######################
> greeks <- list(Delta = NULL, Lambda = NULL, Gamma = NULL,
+    Theta = NULL, Vega = NULL, Rho = NULL)
> greeks.names <- names(greeks)
> nr.nodes <- 101
> S.margin <- seq(from = 80, to = 120, length = nr.nodes)
> t.margin <- c(1/365, 1/52, 1/12, 1/4, 1/2)
> t.line.type <- c(1, 6, 5, 4, 3)
> St.grid <- expand.grid(list(S = S.margin, t = t.margin))

> opt.obj <- bsmEU(spot = St.grid[1], strike = 100, time = St.grid[2],
+    intRate = 0.05, costCarry = 0.05, sigma = 0.2, type = "call")
> methods <- expression(opDeltaBSM(opt.obj), opLambdaBSM(opt.obj),
+    opGammaBSM(opt.obj), opThetaBSM(opt.obj), opVegaBSM(opt.obj),
+    opRhoBSM(opt.obj))

> par(mfrow=c(3,2))
> for (g in 1:6){
+    greeks[[g]] <- matrix(eval(methods[[g]]), ncol = 5)
+    y.limits <- range(greeks[[g]], na.rm = T)
+    plot(x = S.margin, y = greeks[[g]][,1], xlab = "S",
+       ylab = greeks.names[g], ylim = y.limits, type = "l",
+       lty = t.line.type[1])
+    title(main = greeks.names[g])
+    for (t in 2:5){
+       lines(x = S.margin, y = greeks[[g]][,t], lty = t.line.type[t])
+    }
+ }

#######################
# Greeks 3D (Figure 6)
#######################
> greeks <- list(Delta = NULL, Lambda = NULL, Gamma = NULL,
+    Theta = NULL, Vega = NULL, Rho = NULL)
> greeks.names <- names(greeks)
> nr.nodes <- 26
> S.margin <- seq(from = 75, to = 125, length = nr.nodes)
> t.margin <- seq(from = 2/52, to = 1, length = nr.nodes)
> St.grid <- expand.grid(list(S = S.margin, t = t.margin))
> persp.data <- list(x = S.margin, y = t.margin, z = NA)
> eye.multip <- list(c(-6, -8, 5), c(-6, -4, 3), c(-6, -10, 3),
+    c(-6, -8, 5), c(-6, -8, 5), c(-6, -12, 5))
```

```
> opt.obj <- bsmEU(spot = St.grid[1], strike = 100, time = St.grid[2],
+    intRate = 0.05, costCarry = 0.05, sigma = 0.2, type = "call")
> methods <- expression(opDeltaBSM(opt.obj), opLambdaBSM(opt.obj),
+    opGammaBSM(opt.obj), opThetaBSM(opt.obj), opVegaBSM(opt.obj),
+    opRhoBSM(opt.obj))

> par(mfrow=c(3,2), lab=c(3,3,7))
> for (g in 1:6){
+    greeks[[g]] <- matrix(eval(methods[[g]]), ncol = nr.nodes)
+    persp.data$z <- greeks[[g]]
+    persp.data.range <- c(diff(range(S.margin, na.rm = T)),
+       diff(range(t.margin, na.rm = T)), diff(range(greeks[[g]],
+       na.rm = T)))
+    eye.value <- eye.multip[[g]]*persp.data.range
+    persp(persp.data, xlab = "S", ylab = "T", zlab = greeks.names[g],
+       axes = T, box = T, eye = eye.value)
+    title(main = greeks.names[g])
+ }
```

5 References

Espen Gaarder Haug (2006), *The Complete Guide to Option Pricing Formulas, 2nd edition*, McGraw-Hill

John C. Hull (2005), *Options, Futures, and Other Derivatives, 6th edition*, Prentice Hall

Diethelm Würtz (2004), *Computing with `R` and `S-PLUS` For Financial Engineers, Part IV, The Valuation of Options*

Eric Zivot and Jiahui Wang (2006), *Modeling Financial Time Series with `S-PLUS`, 2nd edition*, Springer

`S-PLUS` 8 for Windows Documentation (2007), Insightful Corporation