# Binomial model

Oleh Burdukov

January 25, 2026

This project contains essential computation and illustration functions for valuation of call and put options by well known binomial model. The whole project was created as an extended version of my university project on the same topic. Taking this opportunity, I'd like to express my gratitude to Associate Professor Dr. Josef Leydold for conducting the course of Computing and for the clear and reasonable assignment design that inspired me to proceed my work on this project using the initial tasks as an appropriate basis. I'd also like to pay a tribute to the Group 1 from this course for their commitment on our common work item and to my trusted peer advisor Ivan Tsots for his organisational advice on my project.

In this introductory appendix to my project, I'd like to show what it consists of, as well as its application basics.

## 1 Value-estimating functions

Since almost all functions in this project are similar to some degree, I find it more efficient to discuss all of them step by step describing how they work. In order to make sure that reader understands every function's mechanism, I'm going to refer to some important specifics of my code separately.

### 1.1 Checks

The first thing one can observe in my code are the checks for appropriate input at the beginning of every function. Here is an example:

```
if (!is.numeric(r) || length(r) != 1) {
    warning("r must be a single numeric value. 0 returned")
    return(0)
  }
```

Essentially, most of these are same, although some additional checks are included where required.

In every check, one can observe one warning as well as one return command. I've set 0 as an output to return in order to make my code a bit more comfortable

to use. This approach prevents user from dealing with 'stop' commands that interrupt the code execution every time the wrong input is used.

## 1.2 Additional variables computation

Every function of the code is based on some input variables such as volatility rate, PV of the underlying etc. Perhaps there are more variables to employ.

```
deltat <- T/n
u <- exp(s*sqrt(deltat))
d <- exp(-s*sqrt(deltat))
p <- (exp(r*deltat)-d)/(u-d)
```

The formulas of these are compounded from the initial inputs and play a crucial role in estimation process.

## 1.3 Estimation of values across the binomial tree

This step differs rather more in every function. Nevertheless, its task is the same. Its main goal is to assign 2 to 3 values for every node of the binomial tree depending on what kind of option is observed.

And so, the code, firstly, works with the underlying value estimating its current spot rate at every period between the present time and the maturity date.

```
St <- numeric(length = n+1)
  for(j in 0:n) {
    St[j+1] <- S0*u^j*d^(n-j)
  }
```

This part is basically same, at least it's computation approach. Perhaps, for american options I separately consider the case of the underlying that pays dividends within the period an option is meant to be on the books. Taking american calls as an example, you can observe this part

```
if(!is.null(div)) {
    St[[1]] <- S0 - div/((1+r)^(deltat*exdivdate))
  } else {
    St[[1]] <- S0
  }
```

Here I make sure the PV of the dividends is subtracted from the initial value, so that later in this part

```
if(!is.null(div)) {
    for(i in 1:(exdivdate)) {
      for(j in 0:(i-1)) {
        St[[i]][[j+1]] <- St[[i]][[j+1]] +
                          div/((1+r)^(deltat*(exdivdate+1-i)))
      }
    }
}
```

the code adds back the FV of dividends considering the period along the tree.

Secondly, the code estimates the corresponding value of the option in case of the exercise.

```
Ct <- numeric(length = n+1)
  for(j in 0:n) {
    Ct[j+1] <- max(St[j+1]-E,0)
  }
```

In European options, these values are estimated only for the maturity date. Later, these estimates will be used to calculate the PV of the option by means of the risk-neutral valuation approach:

$$\frac{pC_u+(1-p)C_d}{1+r(\delta*t)}$$

Here is how it looks in the code for European calls:

```
for(i in (n-1):0) {
    C0 <- numeric(i+1)
    for(j in 0:i) {
      C0[j+1] <- (p*Ct[j+2]+(1-p)*Ct[j+1])/(1+r)^deltat
    }
  }
```

On the other hand, american options always require computation of the exercise price at every node along the binomial tree.

```
for(i in n:1) {
    Ct[[i]] <- numeric(i)
    H[[i]] <- numeric(i)
    for(j in 1:i) {
      Ct[[i]][j] <- EX[[i]][j]
      H[[i]][j] <- (p*Ct[[i+1]][j+1]+
                    (1-p)*Ct[[i+1]][j])/(1+r)^deltat
      if(H[[i]][j]>EX[[i]][j]) {
```

```
        Ct[[i]][j] <- H[[i]][j]
      }
    }
  }
```

Although the Put-Call-Parity

$$C_0 - P_0 = S_t - K$$

implies that it never pays to exercise a call prematurely, I still take into account these values for computational fairness. Due to this feature of american options, every node possesses 3 values, namely

1. the spot rate of the underlying $S_t$
2. the exercise price EX
3. the value of holding the option until the next period H

Among the latter 2 the code chooses the greatest value that he further uses in the risk-neutral probability valuation.

## 1.4 The Payoff Plots

For the plots it is necessary to consider such a detail as position in the option. In order to make sure the code shows correct plot, I used if statements. For every position, the code's structure is same, though the input sometimes differs. Observe an example for the European call:

```
if(position == 'long') {
      frame_plot <- data.frame(x = St,
                               y = pmax(St-E,0)-C0*(1+r)^T)
      Theplot <- ggplot(frame_plot, aes(x, y)) +
        geom_glowline(color = 'darkkhaki',
                      linetype = 'solid') +
        labs(x = 'St',
             y = 'Ct(incl. premium)',
             title = "The Option's Payoff",
             subtitle = '(long European call)') +
        theme(plot.background = element_rect(fill = "grey10"),
              panel.background = element_rect(fill = "grey10"),
              plot.title = element_text(hjust = 0.5,
                                        color = 'white'),
              plot.subtitle = element_text(hjust = 0.5,
                                           color = 'white'),
              axis.title.x = element_text(color = 'white'),
```

```
              axis.title.y = element_text(color = 'white'))
      print(Theplot)
}
```

As we can see, this part creates the plot for the long position. The same structure reader can find for the case of the short position. Basically, the plots are created by the same code for all options reader can find in the project.

In my opinion, it is of utter importance to create good and readable (perhaps, maybe also stylish and fancy) plots. That's why I employed the ggplot2 package: not only does it deliver much more freedom in the plot design, but also allows for the use of some useful extensions.

### 1.5 The output

The final part of each code assigns the proper return. In every function, this return appears quite explainable, once the code is read from the beginning to this part. Additionally to the 'return' command, I assign some reasonable text part, which is aimed to make the function more user-friendly. Here we see this part on example of European call as previously:

```
if(position == 'long') {
    cat('The premium due (C0)')
    return(C0)
  } else {
    if(position == 'short') {
      cat('The premium to ask(C0)')
      return(C0)
    } else {
      cat('The premium due (C0)')
      return(C0)
    }
  }
```

From this part, we might notice that, in any unclear case, the code assumes a long position in the contract.

## 2 The sample-path function

This file is still in progress. For further information and assistance, please, contact the author per E-Mail: burdukovoleh@gmail.com