FE 620 - American Options

SUMMER 2020

Waldyr Faustini

Sandeep Ranjan

Zach Townley-Smith

Steven Paker

Contents

List of Tables

List of Figures

Brief History of Options Market

Stock options in the United States started in the late 1800's and during that time the market was over-the-counter and totally decentralized. At that time, options were very illiquid assets. It was a common practice to put options interest in the newspaper and wait for counterparties to come in with their interest. Once you owned the option, you'd either have to wait to see what happened at expiration or place a new ad in a financial journal in order to resell it.

In 1929, Congress stepped in the financial marketplace after the stock market crash. Then, in 1935, the Securities and Exchange Commission (SEC} began regulating the over-the-counter options market giving the Chicago Board of Trade (CBOT) a license to register as a national securities exchange.

Given the low volume in the commodity futures market, in 1968 the CBOT was forced to look for other ways to expand its business. The idea was to create an open-outcry exchange for stock options, similar to the method for trading futures. The Chicago Board Options Exchange (CBOE) was then created.

In the year of 1973 Fischer Black and Myron Scholes published an article titled "The Pricing of Options and Corporate Liabilities" (*University of Chicago's Journal of Political Economy*). During that period, the market players started to analyze options. The Options Clearing Corporation (OCC) also started around the same time and it was created to ensure the obligations associated with options contracts were duly followed.

Most of the features of equity markets appeared after the 1987 crash, around the mid 90's, online trading options became available.

Introduction to Options

For this project, we are going to work through American Options. So, the first thing to do here is to show a brief description about options and specifically about American Options. An option is a contract which gives the buyer (the owner or holder of the option) the right, but not the obligation, to buy or sell an underlying asset or instrument at a specified strike price prior to or on a specified date, depending on the form of the option.

An American option is a version of an options contract that allows holders to exercise the option rights at any time before and including the day of expiration. The exercise style of listed options are American by default, except for options on equity market indexes such as the S&P 500 index. In other words, options on individual stocks and ETFs are exercised in American style.
The last day to exercise a weekly American option is normally on the Friday of the week in which the option contract expires. Conversely, the last day to exercise a monthly American option is normally the third Friday of the month.

Basic Concepts about Options

We will try to show some basic concepts about Options (and American Options) without much mathematical formalism and in the simple way to be easy to understand some concepts in our project.

Exercise: just the options holders can exercise an Option. The act of exercising an option just makes sense if the Option is In the Money (ITM), or in other words, for a Call option if the underlying asset is above the strike price and for a Put option if the underlying is below the strike price. So, exercising the Call Option is to buy the underlying asset according to the quantity prescribed in the contract. And exercising the Put Option is to sell the underlying asset according to the quantity prescribed in the contract

Expiry Date: the final day in the life of an Option. So, an American Option just can be exercised between the trade date and expiry date.

Strike: the strike price is also known as exercise price. The strike is very important to decide if the option will be exercised or not. For American options, let's first consider a Call Option, so in any moment before the expiry date, if the underlying asset trade above the strike price the option holder can (the holder can, is not obliged) exercise the call option. Now considering a Put Option, so in any moment before the expiry date, if the underlying asset trade below the strike price the option holder can (the holder can, is not obliged) exercise the Put Option

Volatility: We can define the volatility as the amount of variability in the returns of the underlying asset. In this case we are measuring the number as summing up the magnitude of all deviations, for volatility we are not interested if the market goes up or down, but in how big are the movements, so in this case the magnitude of the deviations are more important than the sign of the deviations.

Implied Volatility: The implied volatility is computed from market options price.

Historic Volatility: Historic Volatility is useful to compare prices between different periods of time. For example, now during the COVID-19 pandemic the best approach to try to understand how is the market behavior during a huge stress period is to analyze the past during the 2007-2009 crisis for example. Of course, are totally different situations, but it's a good proxy to predict how volatile the market can be during a stress period. Also, Historic Volatility can be used to calibrate risk systems in financial institutions. A very good exercise to try to understand the financial markets is to compare the implied volatility vs historical volatility during the same time period.

<u>Greeks:</u>

<u>Delta:</u> It is defined as the rate of change of the option price with respect to the price of the underlying asset. It's a very good measure to analyze the probabilities of the Option being exercised considering the strike price, underlying price and expiry date. So, let's analyze some examples for Call Options. In-the-money (ITM) call options get closer to 1 as their expiration approaches. At-the-money call options typically have a delta of 0.5, and the delta of out-of-the-money call options approaches 0 as expiration nears. The deeper in-the-money the call option, the closer the delta will be to 1, and the more the option will behave like the underlying asset. A very good interpretation for a Delta Option is in terms of probability, because a 0 Delta option for example is basically implying very low chances to be exercised until the expiry date. On the other hand, 1 Delta Option means the Option is already ITM.

On the trade moment and/or during the Option life, Low Delta Options are much cheaper than a High Delta Option (if the Options price follows the smile curve), because those prices are implying the probabilities of the Option being exercised or not.

<u>Gamma:</u> Gamma can be defined in how fast the Delta Option is increasing or decreasing. Some basic things to keep in mind, a short date Option has more gamma than a long date Option, or in other words, you have more gamma approaching the expiry than in the trade date. And you have more gamma approaching the strike price (in any moment during the life option) than moving away from the strike price (in any moment during the life option).
A very common way to describe the gamma in the portfolio is for example to say “The portfolio is long (short) 10 units of gamma”, that means if the underlying price goes up 1% the portfolio size will increase (decrease) 10 units of the underlying asset and if the market goes down 1% the portfolio size will decrease (increase) 10 units of the underlying asset.

<u>Theta:</u> Each day of a life option has a decay, not the same amount of decay because the underlying asset price is moving, and the decay price will change as well. So, Theta is the time decay of an Option, as time goes by the Option value will decline. If you are an Option Holder you are paying decay to keep the optionality to hold this Option contract, on the other hand, if you are an Option Writer you are receiving decay to give the optionality to the Option Holder.

<u>Vega:</u> Vega is the greek to measure the market volatility in the Option contract. For large movements we are going to see the volatility increasing and for small movements we will see the

volatility falling. If you are an Options holder you are Long Vega, and if you are an Option Writer you are Short Vega.

Rho: Rho measures the sensitivity of an option or options portfolio to a change in interest rate. The most sensitive options to Rho risk are the longest time to expiration and closest of the strike price option.

To summarize the Greeks risk:

- If you are Long Options (Option Holder): Long Gamma, Long Vega, Long Rho and Paying Theta.
- If you are Short Options (Option Writer): Short Gamma, Short Vega, Short Rho and Receiving Theta.

Project Objective

This project will compare different methods to price American call and put options. Two models are presented for pricing American options: the binomial tree method and trinomial tree.

The project scope will include just stock options. Researching about the stocks, we found that most of them pay dividends with some frequency. For the simplicity of pricing our model, we would like to work only with stocks that do not pay dividends but also have liquidity to find data with good quality and accuracy. That way, we got to choose our 4 stocks for the project: Berkshire Hathaway (ticker: BRK.B), Facebook (ticker: FB), United Airlines Holdings (ticker: UAL) and Amazon (ticker: AMZN). In addition, we will also do the price modeling of the S&P 500 Index.

So the initial idea is to price these Stock Options for at least 4 different maturities and consider one Call and one Put for each maturity, for a total of 40 options for each method.

Data Collection and Analysis

The source of equity and option data for the project is Yahoo Finance and Bloomberg due to its availability in R. The two tree models will be judged with how closely they match the actual price of each option. In addition to the option price, this project also will compare the Greeks (Theta, Delta, Vega, Gamma) between the binomial tree and trinomial tree models. The same risk-free rate will be used for all models for valuing options of the same maturity. The risk-free rate will be US 13-week treasury bill rates, held constant from 4/7/20.

To collect the options data, we selected 4 different maturities (near term, 1 month, 2 months, and 3 months) for both call and put options. The main idea here is to analyze how the greeks will change as the time goes for different strikes and maturities, we know for example it will be easier to analyze the Gamma for a short date option and the Vega for a long date option. Historical stock data was obtained using the Quantmod package in R, which retrieves from YAHOO finance. In determining the asset volatility, a period of log returns spanning 7/15/19 -

4/13/20 was considered. Historical option prices were obtained using Bloomberg, which provides options prices as far as 120 days in the past. As such, our option prices are all coinciding with the 1st half of April, when the coronavirus pandemic began spreading across the U.S.

Results - Pricing Market Options

Our project considers 40 American style options, whose prices were obtained from the Bloomberg historical repository, with a price date in early April 2020. Equities in the scope of the analysis include Amazon, Facebook, United Airlines, Berkshire Hathway, and the S&P500 index. For each equity, a near term, 1 month, 2 months, and 3 months expiration option for both calls / puts was considered. The risk-free rate was taken from the 12-week treasury bill rate (^IRX) on 4/7/20 and held constant for all options. The following table gives market option parameters which were used as inputs into our Binomial and Trinomial tree models.

Market Option Data from Bloomberg – AMZN (Amazon) – Stock Price: 2011.59						
Underlying	**Strike**	**Maturity**	**Bid**	**Ask**	**Type**	**Price Date**
AMZN	2000	4/17/2020	47.25	49.50	Call	4/7/2020
AMZN	2000	5/15/2020	102.35	106.00	Call	4/7/2020
AMZN	2000	6/19/2020	127.35	133.20	Call	4/7/2020
AMZN	2000	7/17/2020	140.65	148.00	Call	4/7/2020
AMZN	2000	4/17/2020	32.00	33.95	Put	4/7/2020
AMZN	2000	5/15/2020	86.90	90.30	Put	4/7/2020
AMZN	2000	6/19/2020	110.40	114.00	Put	4/7/2020
AMZN	2000	7/17/2020	123.50	129.70	Put	4/7/2020

Market Option Data from Bloomberg – FB (Facebook) – Stock Price: 168.83						
Underlying	**Strike**	**Maturity**	**Bid**	**Ask**	**Type**	**Price Date**
FB	170	4/17/2020	4.45	4.70	Call	4/7/2020
FB	170	5/15/2020	9.35	9.70	Call	4/7/2020
FB	170	6/19/2020	11.40	12.95	Call	4/7/2020
FB	170	9/18/2020	16.05	17.60	Call	4/7/2020
FB	170	4/17/2020	5.15	5.45	Put	4/7/2020
FB	170	5/15/2020	9.90	10.50	Put	4/7/2020
FB	170	6/19/2020	11.85	12.60	Put	4/7/2020
FB	170	9/18/2020	16.00	17.80	Put	4/7/2020

Market Option Data from Bloomberg – UAL (United Airlines Holding) – Stock Price: 24.48						
Underlying	**Strike**	**Maturity**	**Bid**	**Ask**	**Type**	**Price Date**
UAL	25	4/17/2020	2.17	2.38	Call	4/7/2020
UAL	25	5/15/2020	4.10	4.20	Call	4/7/2020
UAL	25	6/19/2020	4.80	5.25	Call	4/7/2020
UAL	25	7/17/2020	5.10	5.80	Call	4/7/2020
UAL	25	4/17/2020	2.70	2.87	Put	4/7/2020
UAL	25	5/15/2020	4.60	4.75	Put	4/7/2020
UAL	25	6/19/2020	5.30	5.55	Put	4/7/2020
UAL	25	7/17/2020	5.75	6.20	Put	4/7/2020
Market Option Data from Bloomberg – BRK (Berkshire Hathaway) – Stock Price: 185.25						
Underlying	**Strike**	**Maturity**	**Bid**	**Ask**	**Type**	**Price Date**
BRK	190	4/17/2020	1.37	2.87	Call	4/7/2020
BRK	190	5/15/2020	5.20	5.90	Call	4/7/2020
BRK	190	6/19/2020	7.60	9.25	Call	4/7/2020
BRK	190	8/21/2020	8.35	11.45	Call	4/7/2020
BRK	190	4/17/2020	3.85	7.30	Put	4/7/2020
BRK	190	5/15/2020	9.10	10.10	Put	4/7/2020
BRK	190	6/19/2020	11.65	12.50	Put	4/7/2020
BRK	190	8/21/2020	14.30	16.85	Put	4/7/2020
Market Option Data from Bloomberg – SPX (S&P Index) – Index Price: 2846.06						
Underlying	**Strike**	**Maturity**	**Bid**	**Ask**	**Type**	**Price Date**
SPX	2875	5/15/2020	102.60	104.20	Call	4/13/2020
SPX	2865	6/19/2020	156.10	157.60	Call	4/13/2020
SPX	2850	8/21/2020	220.00	222.20	Call	4/13/2020
SPX	2850	9/18/2020	236.40	238.60	Call	4/13/2020
SPX	2870	5/15/2020	101.00	102.30	Put	4/13/2020
SPX	2865	6/19/2020	149.90	151.40	Put	4/13/2020
SPX	2850	8/21/2020	204.20	206.20	Put	4/13/2020
SPX	2850	9/18/2020	223.30	225.00	Put	4/13/2020

Table 1: Market Data from BBG: AMZN, FB, UAL, BRK and S&P Index

Valuation Algorithms

This project compares two different valuation methods for American options: the binomial tree and trinomial tree. The dynamics of the equity's price movement will be governed by geometric Brownian motion in all models. The stochastic differential equation for the log-returns of the asset is of the form:

$$dX = \left(r - \frac{\sigma^2}{2}\right)dt + \sigma dWt$$

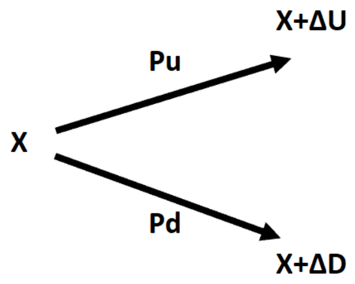
Where *r* is the risk-free rate, σ is the volatility of the asset, *X* is the log returns of the asset, and *Wt* is a Brownian motion with expected value 0 and variance equal to the length of time of the motion. Note that the expected return and variance of expected returns for the asset are given by:

$$E[X] = \left(r - \frac{\sigma^2}{2}\right)\Delta t \qquad V[X] = \sigma^2 \Delta t + \left(r - \frac{\sigma^2}{2}\right)^2 \Delta t^2$$

Binomial Tree Algorithm:

We will use an additive binomial tree in the log-return space to price American style options. The model is simplistic with four parameters:

- *pu* - the probability of the log stock price increasing
- *pd* - the probability of the log stock price decreasing
- Δu- the amount that the log stock price increments in the positive direction
- Δd - the amount that the log stock price increments in the negative direction



In order for this model to match the dynamics of geometric Brownian motion, we must set the first and second moments of this tree-based model equal to that of geometric Brownian motion. This leads us to the system of equations:

$$Pu * \Delta u + Pd * \Delta d = \left(r - \frac{\sigma^2}{2}\right)\Delta t$$

$$Pu * \Delta u^2 + Pd * \Delta d^2 = \sigma^2 \Delta t + \left(r - \frac{\sigma^2}{2}\right)^2 \Delta t^2$$

$$Pu + Pd = 1$$

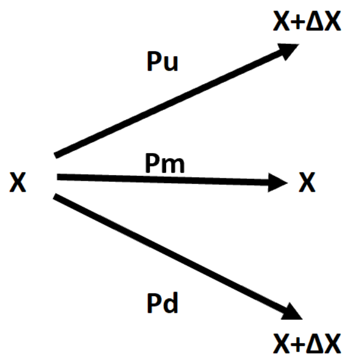
Given that we have three equations with four unknowns, we will use $\Delta u = \Delta d = \Delta x$ to define our specific solution. Our tree can then finally be built by:

$$\Delta x = \sqrt{\sigma^2 \Delta t + \left(r - \frac{\sigma^2}{2}\right)^2 \Delta t^2} \qquad Pu = \frac{1}{2} + \frac{1}{2}\frac{\left(r - \frac{\sigma^2}{2}\right)\Delta t}{\Delta x} \qquad Pd = 1 - Pu$$

The option is priced by working backward in the tree, comparing the immediate exercise value of the option with the discounted, expected return of continuation for each tree node. The value of the option at that node (X, t) is taken as the maximum of the immediate exercise and continuation values.

Trinomial Tree Algorithm:

The trinomial tree is an extension of the binomial tree, with the addition that in addition to the possibility of the log stock price increasing or decreasing, there is also a probability that the stock price remains constant (*Pm*).



In order for the trinomial tree to be recombining, $\Delta u = \Delta d = \Delta x$. Setting the first and second moments of this trinomial tree equal to those given by geometric brownian motion defines our trinomial tree as:

$$Pu = \frac{1}{2}\left(\frac{\sigma^2 \Delta t + D^2 \Delta t^2}{\Delta x^2} + \frac{D\Delta t}{\Delta x}\right) \qquad Pm = 1 - \frac{\sigma^2 \Delta t + D^2 \Delta t^2}{\Delta x^2} \quad Pd = \frac{1}{2}\left(\frac{\sigma^2 \Delta t + D^2 \Delta t^2}{\Delta x^2} - \frac{D\Delta t}{\Delta x}\right)$$
$$\Delta x = \sigma\sqrt{3\Delta t} \qquad \text{where } D = r - \frac{\sigma^2}{2}$$

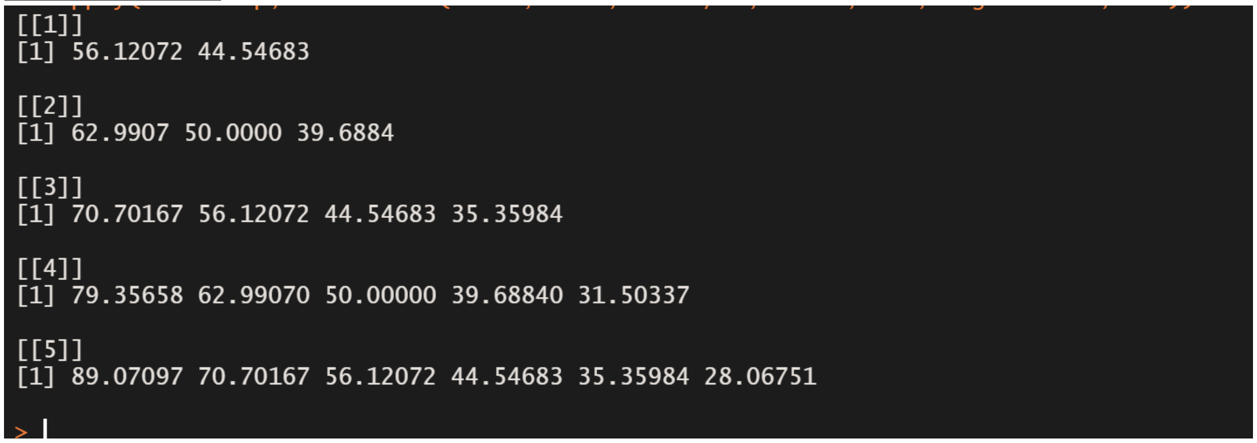
The choice of Δx here is selected such that *Pu, Pm, and Pd* are all [0,1], which holds for any $\Delta x \geq \sigma\sqrt{3\Delta t}$.

As in the binomial tree, the option is priced by working backward, comparing the immediate exercise value of the option with the discounted, expected return of continuation for each tree node. The value of the option at that node (X, t) is taken as the maximum of the immediate exercise and continuation values.

Results - Test Pricing Functions against known results

Hull example 21.1 considers a 5 month American put option on non dividend paying stock with the following characteristics: (S0=50, K=50, rf=10%, vol=40%). The tree developed using the Binomial Tree method is given in figure 21.3 which utilizes 5 time steps. As a check on the function we've developed, we have included a snip of our R code's output for both the stock price and put option value calculated along the tree. The R simulation was found to be in agreement with the literature laid out in Hull.

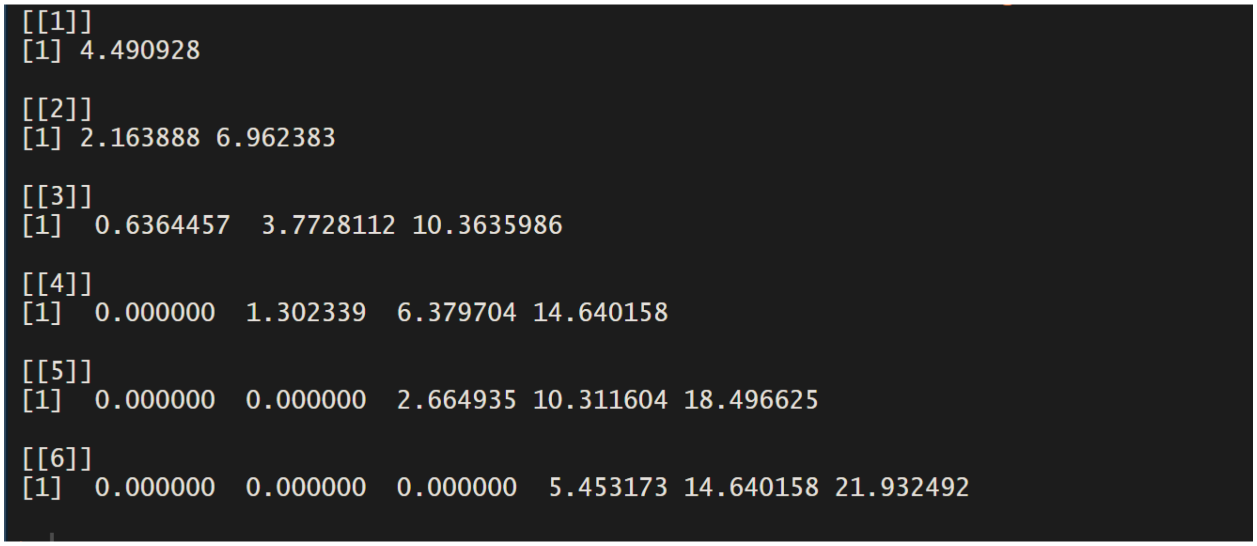
Stock Price Tree:



```
[[1]]
[1] 56.12072 44.54683

[[2]]
[1] 62.9907 50.0000 39.6884

[[3]]
[1] 70.70167 56.12072 44.54683 35.35984

[[4]]
[1] 79.35658 62.99070 50.00000 39.68840 31.50337

[[5]]
[1] 89.07097 70.70167 56.12072 44.54683 35.35984 28.06751

>
```

Figure 1: Stock Price Tree

Put Option Value Tree:



```
[[1]]
[1] 4.490928

[[2]]
[1] 2.163888 6.962383

[[3]]
[1]  0.6364457  3.7728112 10.3635986

[[4]]
[1]  0.000000  1.302339  6.379704 14.640158

[[5]]
[1]  0.000000  0.000000  2.664935 10.311604 18.496625

[[6]]
[1]  0.000000  0.000000  0.000000  5.453173 14.640158 21.932492
```

Figure 2: Put Option Value Tree

For comparison purposes, we have included the known formulation for the Binomial Tree of this options, as laid out in Hull:

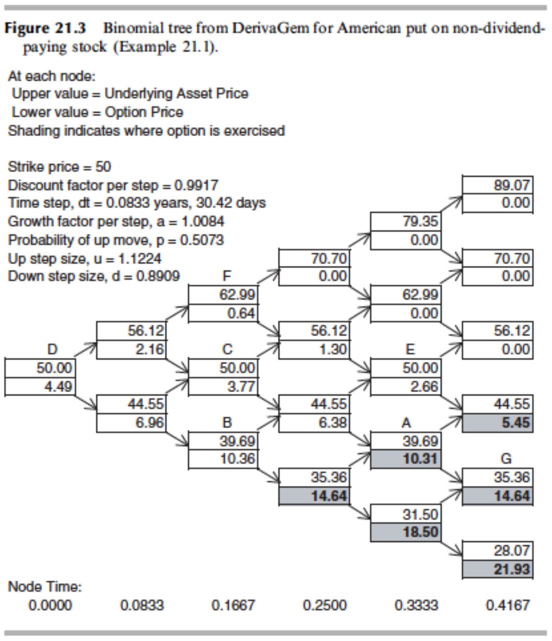


Figure 21.3 Binomial tree from DerivaGem for American put on non-dividend-paying stock (Example 21.1).
At each node:
Upper value = Underlying Asset Price
Lower value = Option Price
Shading indicates where option is exercised
Strike price = 50
Discount factor per step = 0.9917
Time step, dt = 0.0833 years, 30.42 days
Growth factor per step, a = 1.0084
Probability of up move, p = 0.5073
Up step size, u = 1.1224
Down step size, d = 0.8909
D
50.00
4.49
56.12
2.16
44.55
6.96
F
62.99
0.64
C
50.00
3.77
B
39.69
10.36
70.70
0.00
56.12
1.30
44.55
6.38
35.36
14.64
79.35
0.00
62.99
0.00
E
50.00
2.66
A
39.69
10.31
31.50
18.50
89.07
0.00
70.70
0.00
56.12
0.00
44.55
5.45
G
35.36
14.64
28.07
21.93
Node Time:
0.0000
0.0833
0.1667
0.2500
0.3333
0.4167

Figure 3: Hull - Options Futures and Other Derivatives p.453

Evaluation of the Estimated option price for increasing number of timesteps

As a further test to our pricing function in R, we ran a series of cases to demonstrate that as the number of timesteps in your tree approaches infinity, the simulated option price approaches the analytic Black Scholes Merton result. In evaluating this convergence, an American Call option with the following characteristics was utilized (S0=50, K=50, TTM = 5M, r=10%, Vol=40%). The estimated price of this option from the Binomial Tree for a varying number of timesteps (5, 10, 50, 100, 500 timesteps evaluated) is given in the plot below. We were able to demonstrate that this limit does indeed hold with the Tree value of the option converging to the Black Scholes result for high values N.

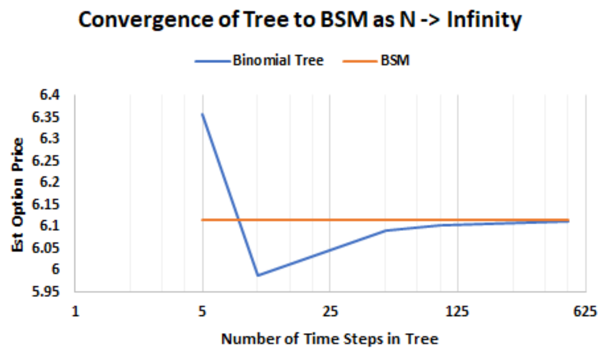
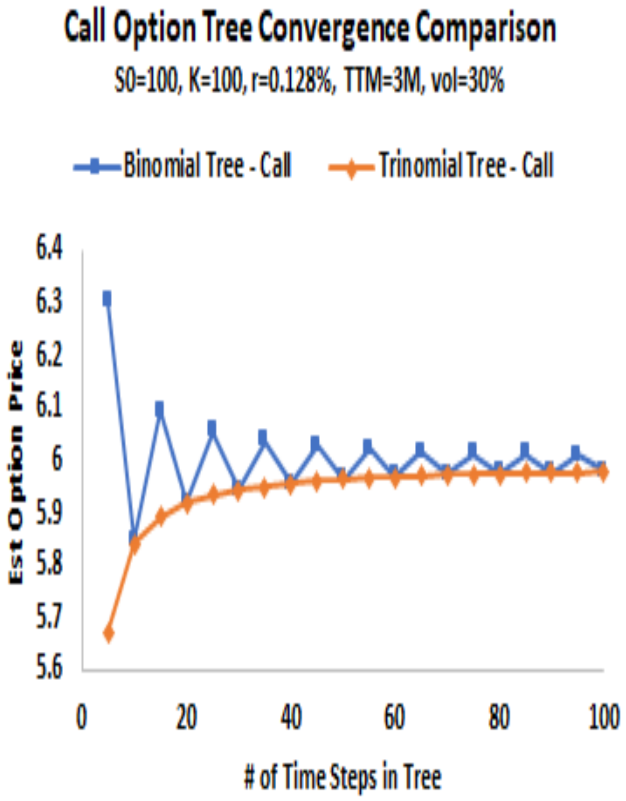


Figure 4: Convergence of Tree to BSM as N goes to Infinity

Sensitivity Analysis on Tree parameter N, the number of timesteps



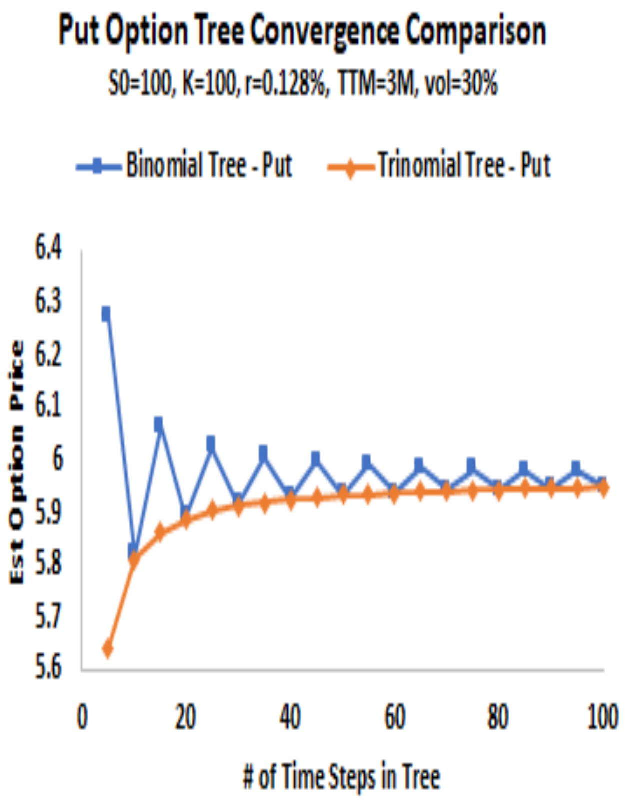


Figure 5: Call Option Tree Convergence Comparison

Figure 6: Put Option Tree Convergence Comparison

To determine the appropriate number of timesteps to use in pricing our options, the longest maturity being 3 months, we ran a series of cases to see the price convergence of American Call / Put Options via the Binomial and Trinomial trees. It was interesting to note that the Trinomial tree approached the BSM result monotonically, while the Binomial tree honed in from both the positive and negative direction. We found that at 50 timesteps, the Binomial price was convergent within 1% of the stable result and the Trinomial tree was convergent within 0.4%. We also found that higher number of timesteps increased the computation time significantly and so all options were priced using the optimized N of 50.

Results Discussion: Pricing of Amazon (AMZN) Options

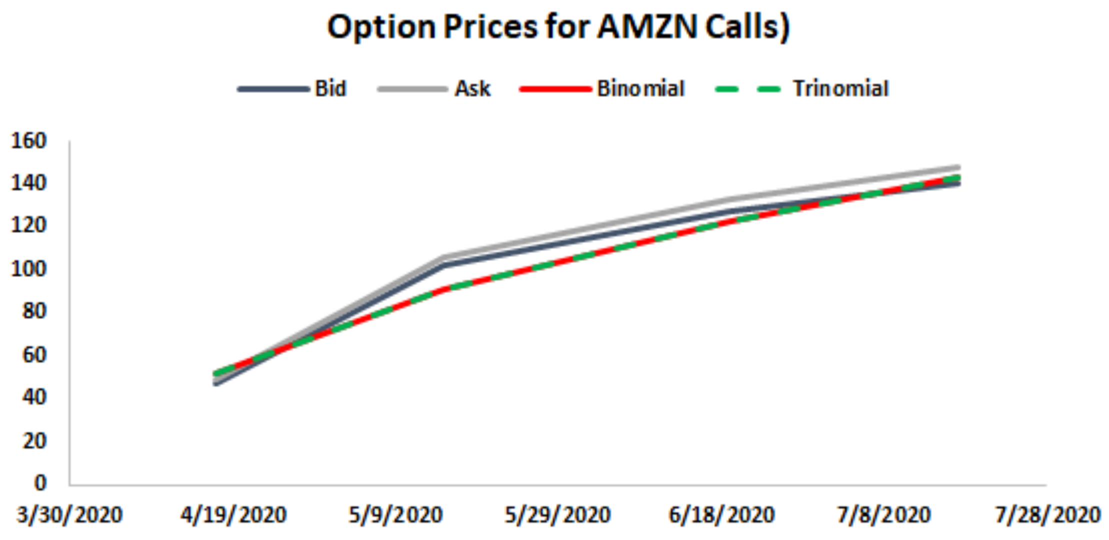


Figure 7: Options Prices for AMZ (Amazon) Calls

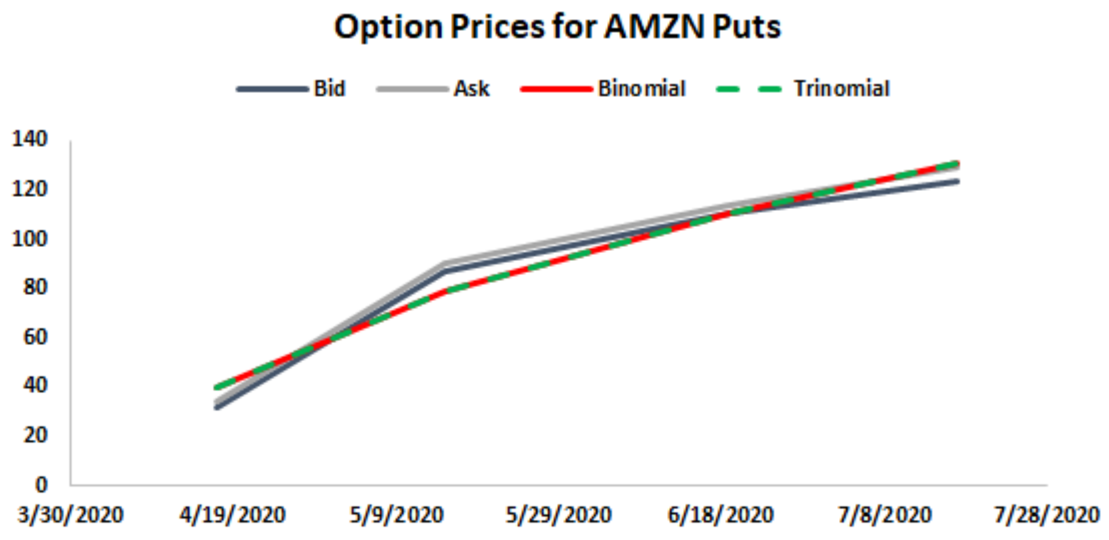


Figure 8: Options Prices for AMZ (Amazon) Puts

For **Amazon,** we determined the historic volatility of log - returns to be **31.88%** using Yahoo closing prices from 7/15/2019 - 4/13/2020. The option prices output by the binomial and trinomial tree models were in agreement, being within **0.09$** of each other on average. The maturity and strike whose price most closely matched the numerical approximation to the Black Scholes Model was the **7/17/20 maturity call (K=2000**), with a **1.11$** offset from the model. The maturity and strike whose price least closely matched the numerical approximation to the Black Scholes Model was the **5/15/20 maturity put (K=2000)**, with a **9.36$** offset from the model.

Results Discussion: Pricing of Facebook (FB) Options

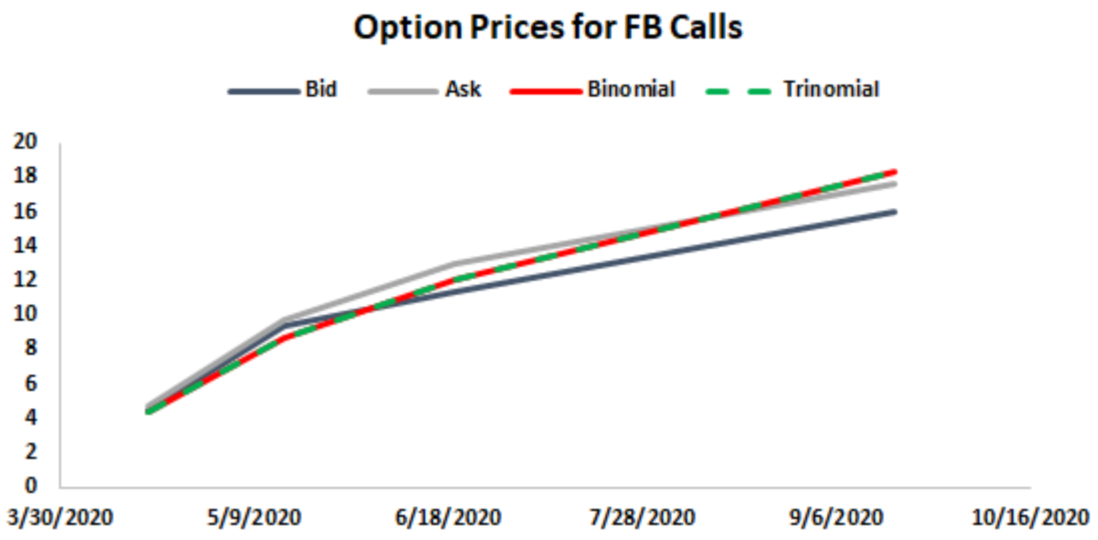


Figure 9: Options Prices for FB (Facebook) Calls

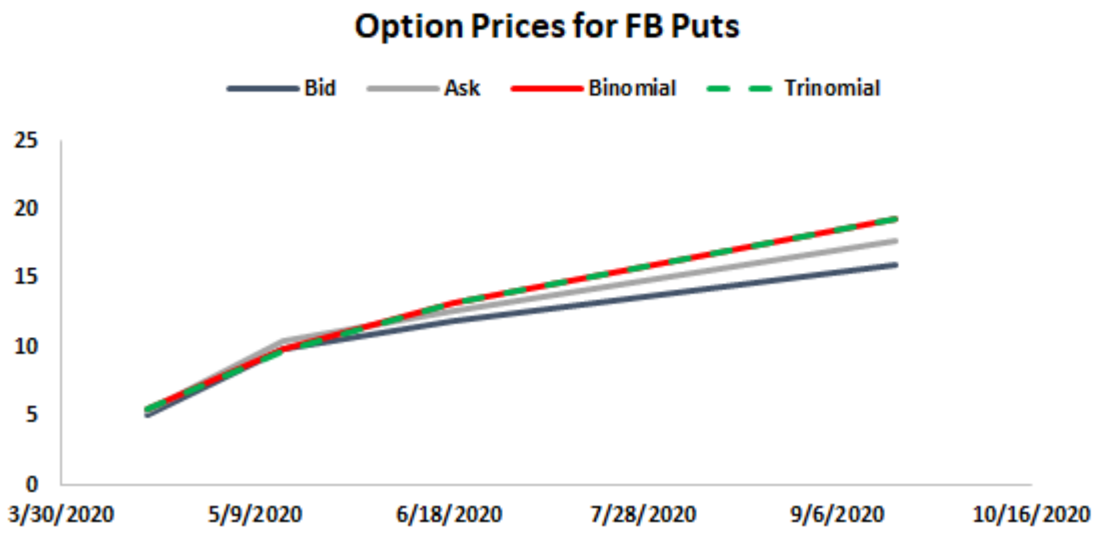


Figure 10: Options Prices for FB (Facebook) Puts

For **Facebook,** we determined the historic volatility of log - returns to be **40.96%** using Yahoo closing prices from 7/15/2019 - 4/13/2020. The option prices output by the binomial and trinomial tree models were in agreement, being within **0.01$** of each other on average. The maturity and strike whose price most closely matched the numerical approximation to the Black Scholes Model was the **6/19/20 maturity call (K=170)**, with a **0.06$** offset from the model. The maturity and strike whose price least closely matched the numerical approximation to the Black Scholes Model was the **9/18/20 maturity put (K=170)**, with a **2.48$** offset from the model.

Results Discussion: Pricing of United Airlines (UAL) Options

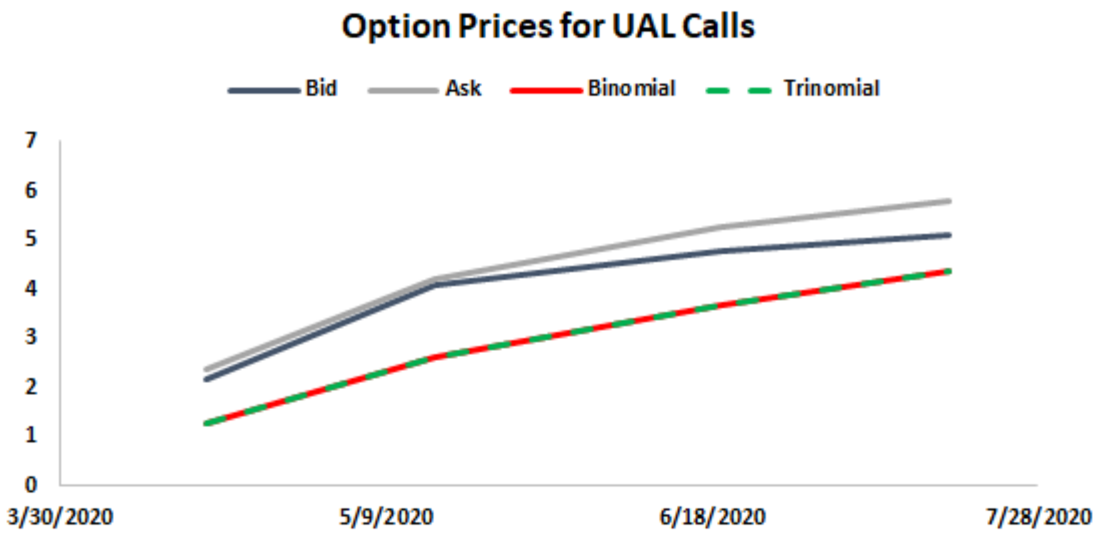


Figure 11: Options Prices for UAL (United Airlines Holding) Calls

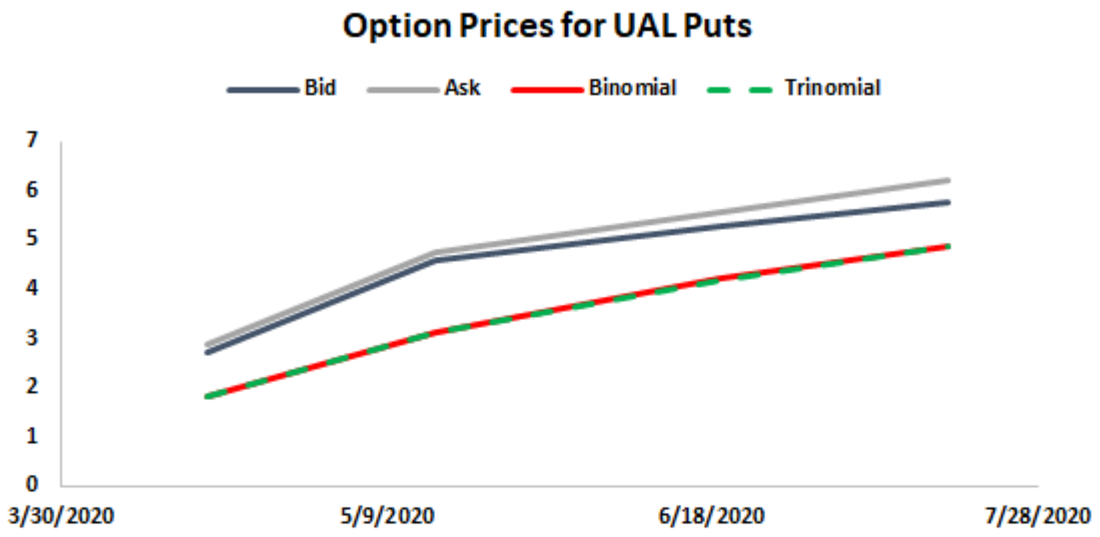


Figure 12: Options Prices for UAL (United Airlines Holding) Puts

For **United Airlines,** we determined the historic volatility of log - returns to be **87.55%** using Yahoo closing prices from 7/15/2019 - 4/13/2020. The option prices output by the binomial and trinomial tree models were in agreement, being within **0.005$** of each other on average. The maturity and strike whose price most closely matched the numerical approximation to the Black Scholes Model was the **4/17/20 maturity put (K=25**), with a **0.97$** offset from the model. The maturity and strike whose price least closely matched the numerical approximation to the Black Scholes Model was the **5/15/20 maturity put (K=25)**, with a **1.53$** offset from the model.

Results Discussion: Pricing of Berkshire Hathaway (BRK) Options

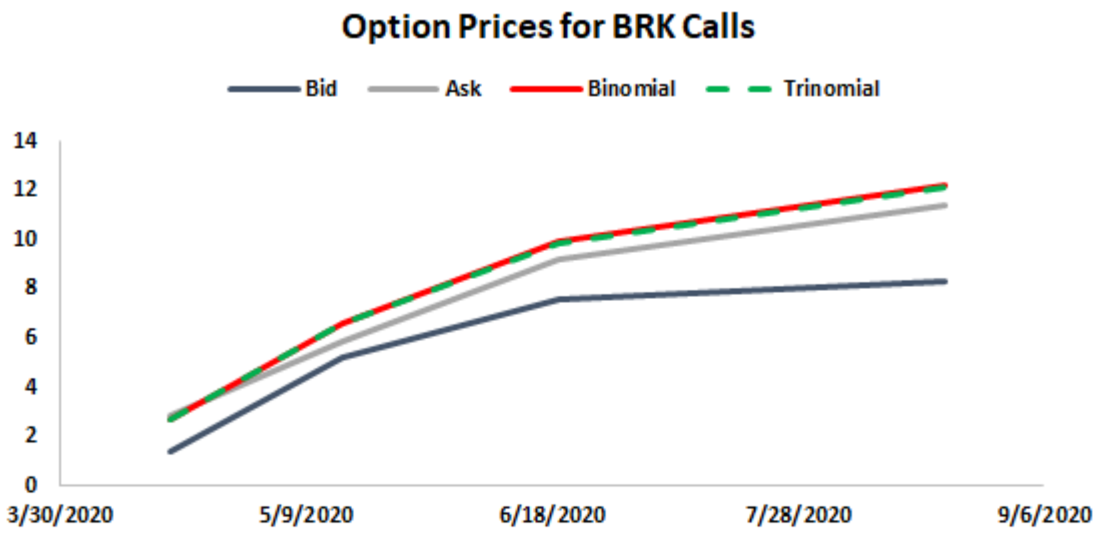


Figure 13: Options Prices for BRK (Berkshire Hathaway) Calls

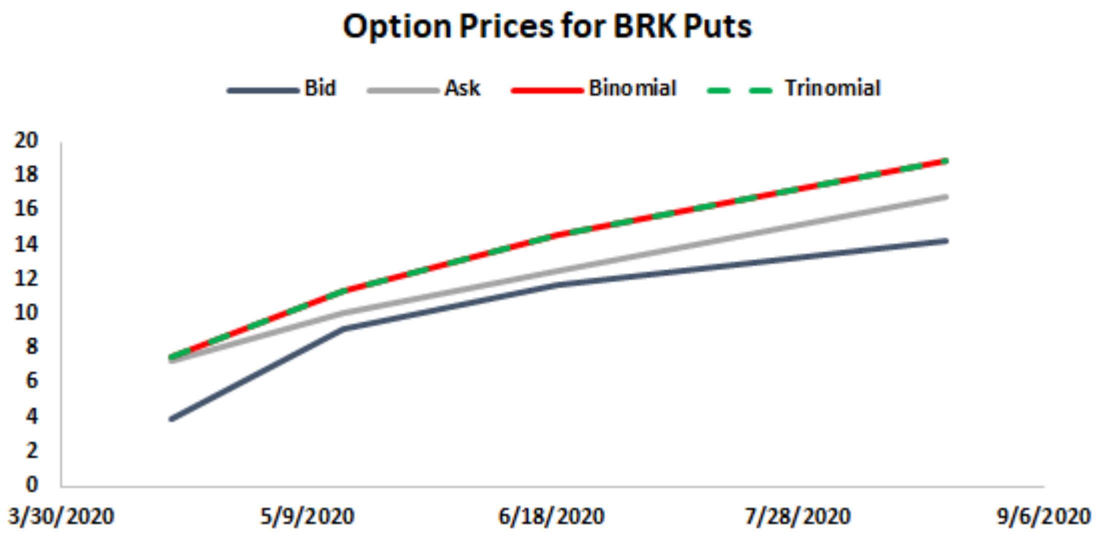


Figure 14: Options Prices for BRK (Berkshire Hathaway) Puts

For **Berkshire Hathaway,** we determined the historic volatility of log - returns to be **35.15%** using Yahoo closing prices from 7/15/2019 - 4/13/2020. The option prices output by the binomial and trinomial tree models were in agreement, being within **0.03$** of each other on average. The maturity and strike whose price most closely matched the numerical approximation to the Black Scholes Model was the **4/17/20 maturity call (K=190**), with a **0.59$** offset from the model. The maturity and strike whose price least closely matched the numerical approximation to the Black Scholes Model was the **8/21/20 maturity put (K=190)**, with a **3.29$** offset from the model.

Results Discussion: Pricing of S&P500 Index (SPX) Options

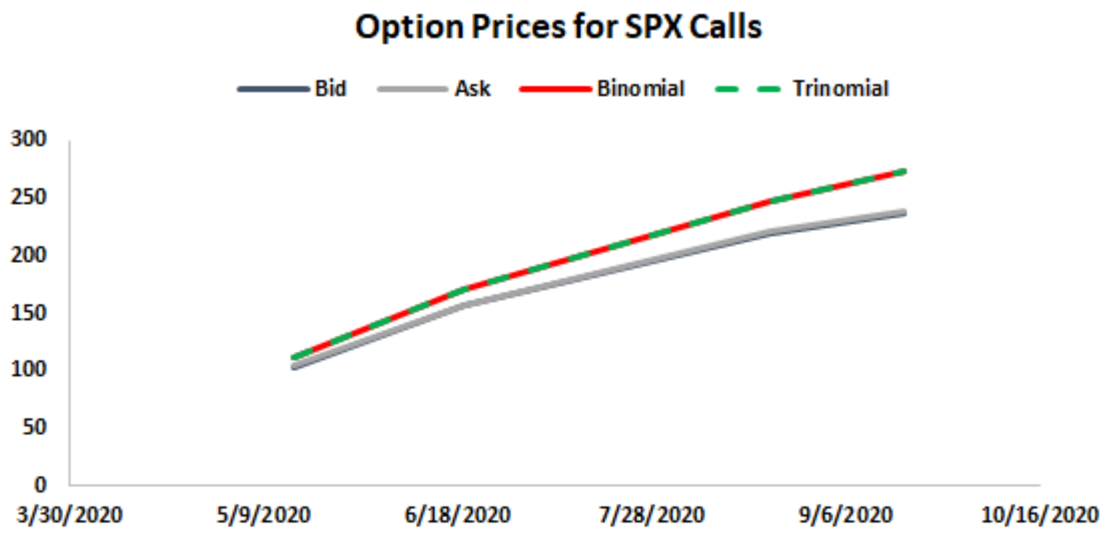


Figure 15: Options Prices for SPX (S&P Index) Calls

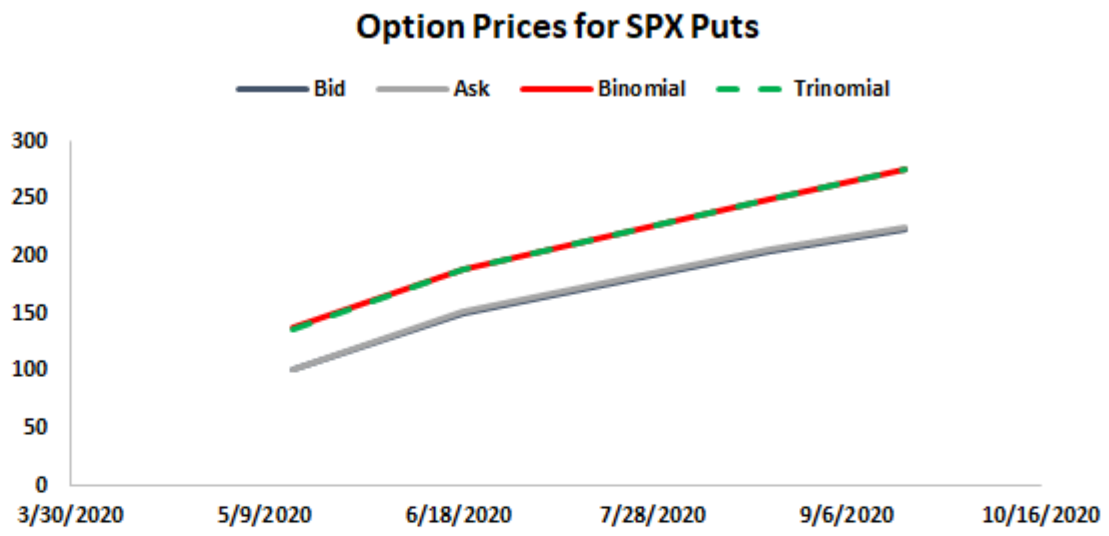


Figure 16: Options Prices for SPX (S&P Index) Puts

For the **S&P500 Index,** we determined the historic volatility of log - returns to be **36.21%** using Yahoo closing prices from 7/15/2019 - 4/13/2020. The option prices output by the binomial and trinomial tree models were in agreement, being within **0.12$** of each other on average. The maturity and strike whose price most closely matched the numerical approximation to the Black Scholes Model was the **5/15/20 maturity call (K=2875**), with a **8.20$** offset from the model. The maturity and strike whose price least closely matched the numerical approximation to the Black Scholes Model was the **9/18/20 maturity put (K=2850)**, with a **51.15$** offset from the model.

Results - Hedging Analysis

A sample of at 10 daily stock prices was generated as a random realization of the Black Scholes model. For each day we computed the American put option price, delta of the put option and portfolio value for a delta hedged portfolio (P - delta * Si). In simulating this, a stock / option of the following characteristics was considered: (S0=100, K=100, TTM=1M, r=5%, vol=40%). In the particular random pathway plotted below, the stock price first climbed above 100 $/share before then falling below 100 $/share and finishing the period at 91.56$. The effect of this on our Put call is a lowering of the option price followed by an increase in option value as the stock price moves the option further into the money. This directionality is also seen in the delta-hedged portfolio, though the portfolio value is of a different scale than the lone option price. However, when comparing the returns in holding the put option vs the hedged portfolio on a percentage basis, you can more clearly see the benefit of hedging which has suppressed our downside risk (and also upside gain) with regard to changes in stock price.

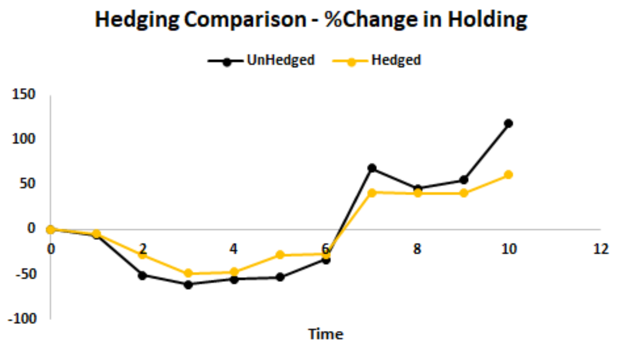


Figure 17: Hedging Comparison

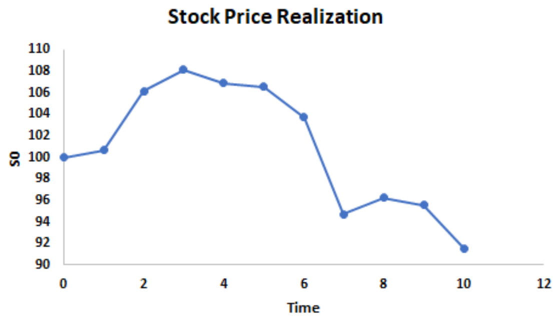


Figure 18: Stock Price Realization

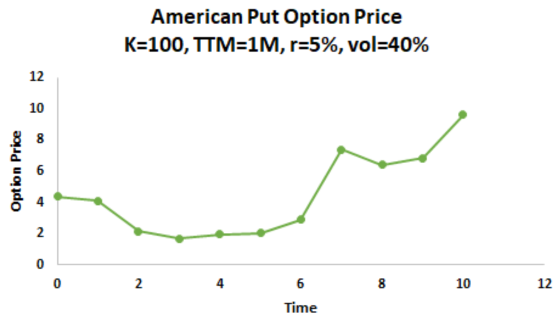


Figure 19: American Put Option Price

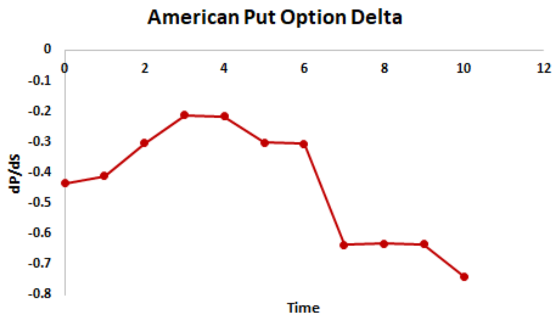


Figure 20: American Put Option Delta

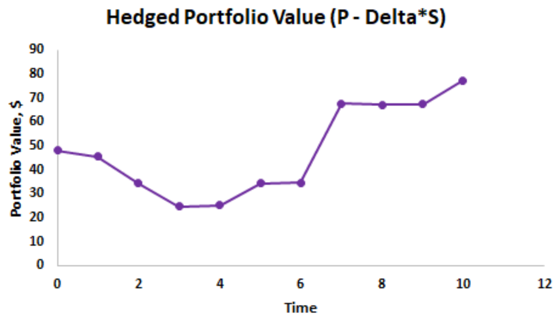
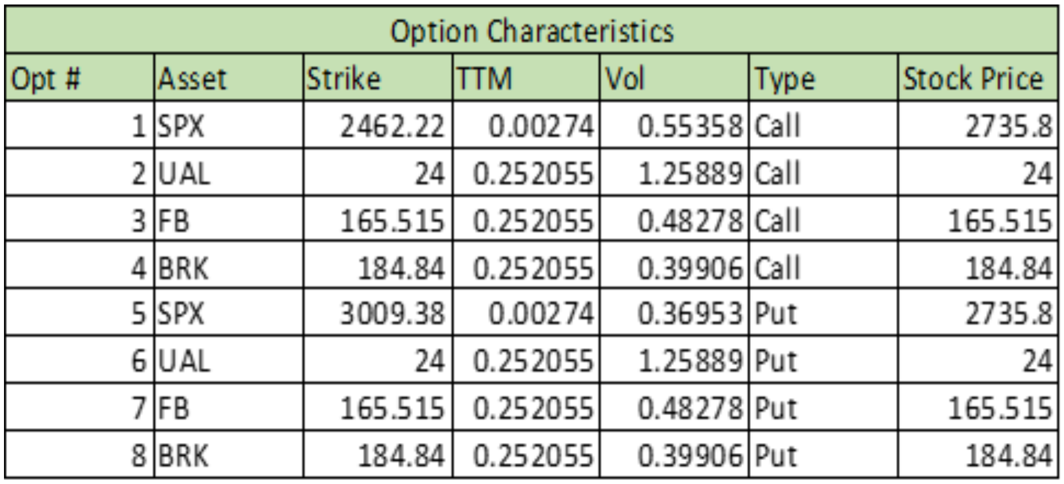


Figure 21: Hedged Portfolio Delta

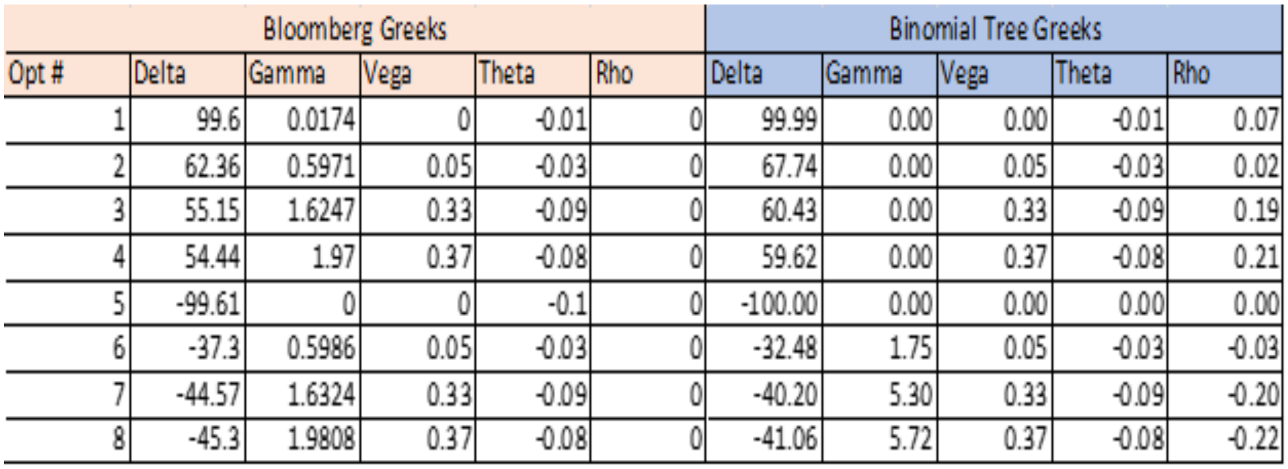
Results - Numerical Approximation of Option Derivatives

In this part of the project we choose 8 options (4 Calls and 4 Puts) and then using the Options details we computed the main greeks for each option (delta, gamma, vega, theta and Rho) using the Binomial Tree Greeks. After that we compared the computed greeks with the market greeks using the Bloomberg data. We achieved very good results using this comparison as can be analyzed in the Table 3 and Table 4 below.



Option Characteristics						
Opt #	Asset	Strike	TTM	Vol	Type	Stock Price
1	SPX	2462.22	0.00274	0.55358	Call	2735.8
2	UAL	24	0.252055	1.25889	Call	24
3	FB	165.515	0.252055	0.48278	Call	165.515
4	BRK	184.84	0.252055	0.39906	Call	184.84
5	SPX	3009.38	0.00274	0.36953	Put	2735.8
6	UAL	24	0.252055	1.25889	Put	24
7	FB	165.515	0.252055	0.48278	Put	165.515
8	BRK	184.84	0.252055	0.39906	Put	184.84

Table 2: Options Characteristics – Numerical Approximation of Options Derivatives



	Bloomberg Greeks					Binomial Tree Greeks				
Opt #	Delta	Gamma	Vega	Theta	Rho	Delta	Gamma	Vega	Theta	Rho
1	99.6	0.0174	0	-0.01	0	99.99	0.00	0.00	-0.01	0.07
2	62.36	0.5971	0.05	-0.03	0	67.74	0.00	0.05	-0.03	0.02
3	55.15	1.6247	0.33	-0.09	0	60.43	0.00	0.33	-0.09	0.19
4	54.44	1.97	0.37	-0.08	0	59.62	0.00	0.37	-0.08	0.21
5	-99.61	0	0	-0.1	0	-100.00	0.00	0.00	0.00	0.00
6	-37.3	0.5986	0.05	-0.03	0	-32.48	1.75	0.05	-0.03	-0.03
7	-44.57	1.6324	0.33	-0.09	0	-40.20	5.30	0.33	-0.09	-0.20
8	-45.3	1.9808	0.37	-0.08	0	-41.06	5.72	0.37	-0.08	-0.22

Table 3: Bloomberg Greeks

Table 4: Binomial Tree Greeks

References:

[1] Florescu, Ionut and Maria Mariani. 2020. *Quantitative Finance.* New Jersey: John Wiley & Sons.
[2] Hull, John. 2018. *Options, Futures and Other Derivatives*. New York: Pearson Education.
[3] Taleb, Nassim Nicholas.1997. *Dynamic Hedging : Managing Vanilla and Exotic Options.* Wiley
[4] Connolly, Kevin B. 1997. *Buying and Selling Volatility*. Wiley

Appendix

<u>R Code:</u>

```
library(quantmod)
library(data.table)
library(bizdays)
setwd("C:/Grad School/FE 620 Pricing and Hedging/Project")



##################################################################################
##########################
#   PART 1: Data Gathering
##################################################################################
##########################

#Read in Option Data from Bloomberg Excel File
Option.Data <- read.csv(file = "Option_Data.csv")
Option.Data



##################################################################################
##########################
#   PART 2: Data Cleaning
##################################################################################
##########################

# Load in previously generated datasets



#Short Term Interest Rates
# Risk free interest rate taken as 13 week T-Bill rate ^IRX on 4/7/20
int <- 0.128/100



#Calculate asset volatilities from 1 yr of Historical closing prices
#getSymbols(Symbols = c("^GSPC","UAL","AMZN", "BRK-B", "FB"), from = "2019-07-15", to =
"2020-07-15")
getSymbols(Symbols = c("^GSPC","UAL","AMZN", "BRK-B", "FB"), from = "2019-07-15", to =
"2020-04-15")
```

```
BRK <- `BRK-B`
SPX <- GSPC
colnames(SPX) <- c("SPX.Open", "SPX.High", "SPX.Low", "SPX.Close", "SPX.Volume",
"SPX.Adjusted")


Historic <- as.data.frame(cbind(SPX$SPX.Adjusted, UAL$UAL.Adjusted, AMZN$AMZN.Adjusted,
BRK$`BRK-B.Adjusted`, FB$FB.Adjusted))
Historic <- cbind(rownames(Historic), Historic)
colnames(Historic) <- c("Date", "SPX.Adjusted", "UAL.Adjusted", "AMZN.Adjusted",
"BRK.B.Adjusted", "FB.Adjusted")
rownames(Historic) <- NULL

Historic$SPXrets <- c(NA,log(Historic$SPX.Adjusted[-1]/Historic$SPX.Adjusted[-
length(Historic$SPX.Adjusted)]))
Historic$UALrets <- c(NA,log(Historic$UAL.Adjusted[-1]/Historic$UAL.Adjusted[-
length(Historic$UAL.Adjusted)]))
Historic$AMZNrets <- c(NA,log(Historic$AMZN.Adjusted[-1]/Historic$AMZN.Adjusted[-
length(Historic$AMZN.Adjusted)]))
Historic$BRKrets <- c(NA,log(Historic$BRK.B.Adjusted[-1]/Historic$BRK.B.Adjusted[-
length(Historic$BRK.B.Adjusted)]))
Historic$FBrets <- c(NA,log(Historic$FB.Adjusted[-1]/Historic$FB.Adjusted[-
length(Historic$FB.Adjusted)]))

SPX.vol <- sd(Historic$SPXrets, na.rm = TRUE)/sqrt(1/252)
UAL.vol <- sd(Historic$UALrets, na.rm = TRUE)/sqrt(1/252)
AMZN.vol <- sd(Historic$AMZNrets, na.rm = TRUE)/sqrt(1/252)
BRK.vol <- sd(Historic$BRKrets, na.rm = TRUE)/sqrt(1/252)
FB.vol <- sd(Historic$FBrets, na.rm = TRUE)/sqrt(1/252)



# Add Current Stock price (at time of option data) and volatility to datasets
for(i in 1:nrow(Option.Data)){
  if(Option.Data[i, "Underlying"] == "AMZN"){
      Option.Data[i, "Volatility"] = AMZN.vol
      temp = Historic[Historic$Date == as.character(as.Date(Option.Data[i, "Price.Date"], format =
"%m/%d/%Y")), "AMZN.Adjusted"]
      Option.Data[i, "Stock.Price"] = temp

  }else if(Option.Data[i, "Underlying"] == "SPX"){
    Option.Data[i, "Volatility"] = SPX.vol
    temp = Historic[Historic$Date == as.character(as.Date(Option.Data[i, "Price.Date"], format =
"%m/%d/%Y")), "SPX.Adjusted"]
```

```
    Option.Data[i, "Stock.Price"] = temp

  }else if(Option.Data[i, "Underlying"] == "UAL"){
    Option.Data[i, "Volatility"] = UAL.vol
    temp = Historic[Historic$Date == as.character(as.Date(Option.Data[i, "Price.Date"], format =
"%m/%d/%Y")), "UAL.Adjusted"]
    Option.Data[i, "Stock.Price"] = temp

  }else if(Option.Data[i, "Underlying"] == "BRK"){
    Option.Data[i, "Volatility"] = BRK.vol
    temp = Historic[Historic$Date == as.character(as.Date(Option.Data[i, "Price.Date"], format =
"%m/%d/%Y")), "BRK.B.Adjusted"]
    Option.Data[i, "Stock.Price"] = temp

  }else{
    Option.Data[i, "Volatility"] = FB.vol
    temp = Historic[Historic$Date == as.character(as.Date(Option.Data[i, "Price.Date"], format =
"%m/%d/%Y")), "FB.Adjusted"]
    Option.Data[i, "Stock.Price"] = temp
  }

}



#Add Time to Maturity for each option
working_calendar <-  create.calendar(name = "mycal", weekdays=c("saturday", "sunday"))
Option.Data$TTM = bizdays(from = as.Date(Option.Data[,"Price.Date"], format = "%m/%d/%Y"),
to = as.Date(Option.Data[,"Maturity"], format = "%m/%d/%Y"), working_calendar)/252



################################################################################
##########################
#   PART 3: Definition of Option Pricing Models
################################################################################
##########################

#Define a function to output an additive Binomial Tree of potential X's (log returns)
#The change up / down was held constant. Xd = Xu (Trigeorgis Tree Construction)
Binom.Tree <- function(S0, K, TTM, r, d, sigma, N){
  del.t <- TTM/N
  del.x <- sqrt((r-d-sigma^2/2)^2*del.t^2+sigma^2*del.t)
  pu <- 1/2 + 1/2*(r-d-sigma^2/2)*del.t/del.x
  pd <- 1 - pu

  Xtree <- list()

  for(i in 1:N){
    jvec <- vector("numeric", length = i+1)
    for(j in 0:i){
      jvec[j+1] <- log(S0) + (i-j)*del.x - j*del.x
    }
    Xtree[[i]] <- jvec
  }

  return(Xtree)
}
#Function Testing
#Binom.Tree (S0 = 30, K = 30, TTM = 1, r = 5/100, d=0, sigma = 0.25, N = 5)


#Define Function for pricing American Calls via the Binomial Tree
American.Call <- function(S0, K, TTM, r, d, sigma, N){

  Xtree <- Binom.Tree (S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
  del.t <- TTM/N
  del.x <- sqrt((r-d-sigma^2/2)^2*del.t^2+sigma^2*del.t)
  pu <- 1/2 + 1/2*(r-d-sigma^2/2)*del.t/del.x
  pd <- 1 - pu

  temp <- Xtree[[length(Xtree)]]
  terminal <- exp(temp)-K
  terminal <- mapply(FUN = max, terminal, 0)

  #Note loop terminates at time 2, due to lookback from immediate excersise condition
  for(i in length(Xtree):2){
    temp1 <- vector("numeric", length = i)
    temp2 <- vector("numeric", length = i)
    temp <- vector("numeric", length = i)
    for(j in 1:i){

      #temp1 vector for expected option value working backwards
      temp1[j] <- exp(-r*del.t)*(pu*terminal[j]+pd*terminal[j+1])

      #temp2 vector for immediate excersise condition
      temp2[j] <- exp(Xtree[[i-1]][j])-K
    #temp vector to do the optimum of holding option or excersising at that time
    temp <- mapply(FUN = max, temp1, temp2)
   }
   terminal <- temp
  }

  #Complete last iteration
  temp1 <- vector("numeric")
  temp2 <- vector("numeric")
  temp <- vector("numeric")

  temp1 <- exp(-r*del.t)*(pu*terminal[1]+pd*terminal[2])
  temp2 <- S0-K
  option.price <- mapply(FUN = max, temp1, temp2)

  return(option.price)
}

#Testing Function
#American.Call(S0 = 30, K = 30, TTM = 1, r = 5/100, d=0, sigma = 0.25, N = 100)


#Define Function for pricing European Calls via the Binomial Tree
European.Call <- function(S0, K, TTM, r, d, sigma, N){

  Xtree <- Binom.Tree (S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
  del.t <- TTM/N
  del.x <- sqrt((r-d-sigma^2/2)^2*del.t^2+sigma^2*del.t)
  pu <- 1/2 + 1/2*(r-d-sigma^2/2)*del.t/del.x
  pd <- 1 - pu

  temp <- Xtree[[length(Xtree)]]
  terminal <- exp(temp)-K
  terminal <- mapply(FUN = max, terminal, 0)

  for(i in length(Xtree):1){
   temp <- vector("numeric", length = i)
   for(j in 1:i){
    temp[j] <- exp(-r*del.t)*(pu*terminal[j]+pd*terminal[j+1])
   }

   terminal <- temp
  }
  option.price <- terminal
  return(option.price)
}
#Testing Function
#European.Call(S0 = 30, K = 30, TTM = 1, r = 5/100, d=0, sigma = 0.25, N = 100)


#Define Function for pricing American Puts via the Binomial Tree
American.Put <- function(S0, K, TTM, r, d, sigma, N){

  Xtree <- Binom.Tree (S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
  del.t <- TTM/N
  del.x <- sqrt((r-d-sigma^2/2)^2*del.t^2+sigma^2*del.t)
  pu <- 1/2 + 1/2*(r-d-sigma^2/2)*del.t/del.x
  pd <- 1 - pu

  temp <- Xtree[[length(Xtree)]]
  terminal <- K-exp(temp)
  terminal <- mapply(FUN = max, terminal, 0)

  #Note loop terminates at time 2, due to lookback from immediate excersise condition
  for(i in length(Xtree):2){
    temp1 <- vector("numeric", length = i)
    temp2 <- vector("numeric", length = i)
    temp <- vector("numeric", length = i)
    for(j in 1:i){

      #temp1 vector for expected option value working backwards
      temp1[j] <- exp(-r*del.t)*(pu*terminal[j]+pd*terminal[j+1])

      #temp2 vector for immediate excersise condition
      temp2[j] <- K-exp(Xtree[[i-1]][j])

      #temp vector to do the optimum of holding option or excersising at that time
      temp <- mapply(FUN = max, temp1, temp2)
    }
    terminal <- temp
  }

  #Complete last iteration
  temp1 <- vector("numeric")
  temp2 <- vector("numeric")
  temp <- vector("numeric")
  temp1 <- exp(-r*del.t)*(pu*terminal[1]+pd*terminal[2])
  temp2 <- K-S0
  option.price <- mapply(FUN = max, temp1, temp2)

  return(option.price)
}
#Testing Function
#American.Put(S0 = 30, K = 30, TTM = 1, r = 5/100, d=0, sigma = 0.25, N = 100)


#Define Function for pricing European Puts via the Binomial Tree
European.Put <- function(S0, K, TTM, r, d, sigma, N){

  Xtree <- Binom.Tree (S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
  del.t <- TTM/N
  del.x <- sqrt((r-d-sigma^2/2)^2*del.t^2+sigma^2*del.t)
  pu <- 1/2 + 1/2*(r-d-sigma^2/2)*del.t/del.x
  pd <- 1 - pu

  temp <- Xtree[[length(Xtree)]]
  terminal <- K-exp(temp)
  terminal <- mapply(FUN = max, terminal, 0)

  for(i in length(Xtree):1){
    temp <- vector("numeric", length = i)
    for(j in 1:i){
      temp[j] <- exp(-r*del.t)*(pu*terminal[j]+pd*terminal[j+1])
    }

    terminal <- temp
  }

  option.price <- terminal
  return(option.price)
}
#Testing Function
#European.Put(S0 = 30, K = 30, TTM = 1, r = 5/100, d=0, sigma = 0.25, N = 100)


#Define a function to output an additive Trinomial Tree of potential X's (log returns)
#The change up / down was held constant. Xd = Xu (Trigeorgis Tree Construction)
Trinom.Tree <- function(S0, K, TTM, r, d, sigma, N){
  del.t <- TTM/N
  del.x <- sigma*sqrt(3*del.t)
  D <- r-d-sigma^2/2
  pu <- 1/2*((sigma^2*del.t+D^2*del.t^2)/(del.x^2)+D*del.t/del.x)
  pm <- 1 - (sigma^2*del.t+D^2*del.t^2)/(del.x^2)
  pd <- 1/2*((sigma^2*del.t+D^2*del.t^2)/(del.x^2)-D*del.t/del.x)

  Xtree <- list()

  for(i in 1:N){
   jvec <- vector("numeric", length = 2*i+1)
   for(j in -i:i){
    jvec[j+i+1] <- log(S0) - j*del.x
   }
   Xtree[[i]] <- jvec
  }

  return(Xtree)
}
#Function Testing
#Trinom.Tree (S0 = 30, K = 30, TTM = 1, r = 5/100, d=0, sigma = 0.25, N = 5)


#Define Function for pricing American Calls via the Trinomial Tree
Tri.American.Call <- function(S0, K, TTM, r, d, sigma, N){

  Xtree <- Trinom.Tree (S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
  del.t <- TTM/N
  del.x <- sigma*sqrt(3*del.t)
  D <- r-d-sigma^2/2
  pu <- 1/2*((sigma^2*del.t+D^2*del.t^2)/(del.x^2)+D*del.t/del.x)
  pm <- 1 - (sigma^2*del.t+D^2*del.t^2)/(del.x^2)
  pd <- 1/2*((sigma^2*del.t+D^2*del.t^2)/(del.x^2)-D*del.t/del.x)

  temp <- Xtree[[length(Xtree)]]
  terminal <- exp(temp)-K
  terminal <- mapply(FUN = max, terminal, 0)

  #Note loop terminates at time 2, due to lookback from immediate excersise condition
  for(i in length(Xtree):2){
   temp1 <- vector("numeric", length = 2*i-1)
   temp2 <- vector("numeric", length = 2*i-1)
   temp <- vector("numeric", length = 2*i-1)
   for(j in 1:(2*i-1)){

    #temp1 vector for expected option value working backwards
```

```
      temp1[j] <- exp(-r*del.t)*(pu*terminal[j]+pm*terminal[j+1]+pd*terminal[j+2])

      #temp2 vector for immediate exercise condition
      temp2[j] <- exp(Xtree[[i-1]][j])-K

      #temp vector to do the optimum of holding option or excersising at that time
      temp <- mapply(FUN = max, temp1, temp2)
    }
    terminal <- temp
  }

  #Complete last iteration
  temp1 <- vector("numeric")
  temp2 <- vector("numeric")
  temp <- vector("numeric")

  temp1 <- exp(-r*del.t)*(pu*terminal[1]+pm*terminal[2]+pd*terminal[3])
  temp2 <- S0-K
  option.price <- mapply(FUN = max, temp1, temp2)

  return(option.price)
}
#Testing Function
#Tri.American.Call(S0 = 30, K = 30, TTM = 1, r = 5/100, d=0, sigma = 0.25, N = 100)



#Define Function for pricing European Calls via the Trinomial Tree
Tri.European.Call <- function(S0, K, TTM, r, d, sigma, N){

  Xtree <- Trinom.Tree (S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
  del.t <- TTM/N
  del.x <- sigma*sqrt(3*del.t)
  D <- r-d-sigma^2/2
  pu <- 1/2*((sigma^2*del.t+D^2*del.t^2)/(del.x^2)+D*del.t/del.x)
  pm <- 1 - (sigma^2*del.t+D^2*del.t^2)/(del.x^2)
  pd <- 1/2*((sigma^2*del.t+D^2*del.t^2)/(del.x^2)-D*del.t/del.x)

  temp <- Xtree[[length(Xtree)]]
  terminal <- exp(temp)-K
  terminal <- mapply(FUN = max, terminal, 0)

  for(i in length(Xtree):1){
    temp <- vector("numeric", length = 2*i-1)
    for(j in 1:(2*i-1)){
      temp[j] <- exp(-r*del.t)*(pu*terminal[j]+pm*terminal[j+1]+pd*terminal[j+2])
    }

    terminal <- temp
  }

  option.price <- terminal
  return(option.price)
}
#Function Testing
#Tri.European.Call(S0 = 30, K = 30, TTM = 1, r = 5/100, d=0, sigma = 0.25, N = 100)



#Define Function for pricing American Puts via the Trinomial Tree
Tri.American.Put <- function(S0, K, TTM, r, d, sigma, N){

  Xtree <- Trinom.Tree (S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
  del.t <- TTM/N
  del.x <- sigma*sqrt(3*del.t)
  D <- r-d-sigma^2/2
  pu <- 1/2*((sigma^2*del.t+D^2*del.t^2)/(del.x^2)+D*del.t/del.x)
  pm <- 1 - (sigma^2*del.t+D^2*del.t^2)/(del.x^2)
  pd <- 1/2*((sigma^2*del.t+D^2*del.t^2)/(del.x^2)-D*del.t/del.x)

  temp <- Xtree[[length(Xtree)]]
  terminal <- K-exp(temp)
  terminal <- mapply(FUN = max, terminal, 0)

  #Note loop terminates at time 2, due to lookback from immediate excersise condition
  for(i in length(Xtree):2){
    temp1 <- vector("numeric", length = 2*i-1)
    temp2 <- vector("numeric", length = 2*i-1)
    temp <- vector("numeric", length = 2*i-1)
    for(j in 1:(2*i-1)){

      #temp1 vector for expected option value working backwards
      temp1[j] <- exp(-r*del.t)*(pu*terminal[j]+pm*terminal[j+1]+pd*terminal[j+2])

      #temp2 vector for immediate exercise condition
      temp2[j] <- K-exp(Xtree[[i-1]][j])

      #temp vector to do the optimum of holding option or excersising at that time
      temp <- mapply(FUN = max, temp1, temp2)
    }
    terminal <- temp
  }

  #Complete last iteration
  temp1 <- vector("numeric")
  temp2 <- vector("numeric")
  temp <- vector("numeric")

  temp1 <- exp(-r*del.t)*(pu*terminal[1]+pm*terminal[2]+pd*terminal[3])
  temp2 <- K-S0
  option.price <- mapply(FUN = max, temp1, temp2)

  return(option.price)
}
#Testing Function
#Tri.American.Put(S0 = 30, K = 30, TTM = 1, r = 5/100, d=0, sigma = 0.25, N = 100)


#Define Function for pricing European Puts via the Trinomial Tree
Tri.European.Put <- function(S0, K, TTM, r, d, sigma, N){

  Xtree <- Trinom.Tree (S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
  del.t <- TTM/N
  del.x <- sigma*sqrt(3*del.t)
  D <- r-d-sigma^2/2
  pu <- 1/2*((sigma^2*del.t+D^2*del.t^2)/(del.x^2)+D*del.t/del.x)
  pm <- 1 - (sigma^2*del.t+D^2*del.t^2)/(del.x^2)
  pd <- 1/2*((sigma^2*del.t+D^2*del.t^2)/(del.x^2)-D*del.t/del.x)

  temp <- Xtree[[length(Xtree)]]
  terminal <- K-exp(temp)
  terminal <- mapply(FUN = max, terminal, 0)

  for(i in length(Xtree):1){
    temp <- vector("numeric", length = 2*i-1)
    for(j in 1:(2*i-1)){
      temp[j] <- exp(-r*del.t)*(pu*terminal[j]+pm*terminal[j+1]+pd*terminal[j+2])
    }

    terminal <- temp
  }
  option.price <- terminal
  return(option.price)
}
#Function Testing
#Tri.European.Put(S0 = 30, K = 30, TTM = 1, r = 5/100, d=0, sigma = 0.25, N = 100)



#####################################################################################
###########################
#   PART 4: Run Pricing models on real option data
#####################################################################################
###########################

# First Look at Convergence of Binomial and Trinomial Trees on 3Mo option to determine
number of tree nodes
Nvec <- c(1:20*5)
Sens.Binomial <- mapply(FUN = American.Call,
                  S0 = 100,
                  K = 100,
                  TTM = 0.25,
                  r = int,
                  d = 0,
                  sigma = 0.3,
                  N = Nvec
)

Sens.Trinomial <- mapply(FUN = Tri.American.Call,
                  S0 = 100,
                  K = 100,
                  TTM = 0.25,
                  r = int,
                  d = 0,
                  sigma = 0.3,
                  N = Nvec
)

Sens.Binomial.Put <- mapply(FUN = American.Put,
                  S0 = 100,
                  K = 100,
                  TTM = 0.25,
                  r = int,
                  d = 0,
                  sigma = 0.3,
                    N = Nvec
)

Sens.Trinomial.Put <- mapply(FUN = Tri.American.Put,
                    S0 = 100,
                    K = 100,
                    TTM = 0.25,
                    r = int,
                    d = 0,
                    sigma = 0.3,
                    N = Nvec
)

Sensitivity <- cbind(Nvec, Sens.Binomial, Sens.Trinomial, Sens.Binomial.Put, Sens.Trinomial.Put)
#write.csv(Sensitivity, "Sensitivity.csv")

#Price American Call Options under Binomial Tree
Option.Data$Binomial[Option.Data$Type=="Call" & Option.Data$Underlying!="SPX"] <-
mapply(FUN = American.Call,
                                        S0 = Option.Data$Stock.Price[Option.Data$Type=="Call" &
Option.Data$Underlying!="SPX"],
                                        K = Option.Data$Strike[Option.Data$Type=="Call" &
Option.Data$Underlying!="SPX"],
                                        TTM = Option.Data$TTM[Option.Data$Type=="Call" &
Option.Data$Underlying!="SPX"],
                                        r = int,
                                        d = 0,
                                        sigma = Option.Data$Volatility[Option.Data$Type=="Call" &
Option.Data$Underlying!="SPX"],
                                        N = 50
                                         )


#Price American Put Options under Binomial Tree
Option.Data$Binomial[Option.Data$Type=="Put" & Option.Data$Underlying!="SPX"] <-
mapply(FUN = American.Put,
                                        S0 = Option.Data$Stock.Price[Option.Data$Type=="Put" &
Option.Data$Underlying!="SPX"],
                                        K = Option.Data$Strike[Option.Data$Type=="Put" &
Option.Data$Underlying!="SPX"],
                                        TTM = Option.Data$TTM[Option.Data$Type=="Put" &
Option.Data$Underlying!="SPX"],
                                        r = int,
                                        d = 0,
                                        sigma = Option.Data$Volatility[Option.Data$Type=="Put" &
Option.Data$Underlying!="SPX"],
                                        N = 50
)

#Price American Call Options under Trinomial Tree
Option.Data$Trinomial[Option.Data$Type=="Call" & Option.Data$Underlying!="SPX"]<-
mapply(FUN = Tri.American.Call,
                                        S0 = Option.Data$Stock.Price[Option.Data$Type=="Call" &
Option.Data$Underlying!="SPX"],
                                        K = Option.Data$Strike[Option.Data$Type=="Call" &
Option.Data$Underlying!="SPX"],
                                        TTM = Option.Data$TTM[Option.Data$Type=="Call" &
Option.Data$Underlying!="SPX"],
                                        r = int,
                                        d = 0,
                                        sigma = Option.Data$Volatility[Option.Data$Type=="Call" &
Option.Data$Underlying!="SPX"],
                                        N = 50
)

#Price American Put Options under Binomial Tree
Option.Data$Trinomial[Option.Data$Type=="Put" & Option.Data$Underlying!="SPX"] <-
mapply(FUN = Tri.American.Put,
                                       S0 = Option.Data$Stock.Price[Option.Data$Type=="Put" &
Option.Data$Underlying!="SPX"],
                                       K = Option.Data$Strike[Option.Data$Type=="Put" &
Option.Data$Underlying!="SPX"],
                                       TTM = Option.Data$TTM[Option.Data$Type=="Put" &
Option.Data$Underlying!="SPX"],
                                       r = int,
                                       d = 0,
                                       sigma = Option.Data$Volatility[Option.Data$Type=="Put" &
Option.Data$Underlying!="SPX"],
                                       N = 50
)


#################################################################################
#####
##### Price SPX As European

#Price European Call Options under Binomial Tree
```

```
Option.Data$Binomial[Option.Data$Type=="Call" & Option.Data$Underlying=="SPX"] <-
mapply(FUN = European.Call,
                              S0 = Option.Data$Stock.Price[Option.Data$Type=="Call" &
Option.Data$Underlying=="SPX"],
                              K = Option.Data$Strike[Option.Data$Type=="Call" &
Option.Data$Underlying=="SPX"],
                              TTM = Option.Data$TTM[Option.Data$Type=="Call" &
Option.Data$Underlying=="SPX"],
                              r = int,
                              d = 0,
                              sigma = Option.Data$Volatility[Option.Data$Type=="Call" &
Option.Data$Underlying=="SPX"],
                              N = 50
)


#Price American Put Options under Binomial Tree
Option.Data$Binomial[Option.Data$Type=="Put" & Option.Data$Underlying=="SPX"] <-
mapply(FUN = European.Put,
                                                  S0 =
Option.Data$Stock.Price[Option.Data$Type=="Put" & Option.Data$Underlying=="SPX"],
                                                  K =
Option.Data$Strike[Option.Data$Type=="Put" & Option.Data$Underlying=="SPX"],
                                                  TTM =
Option.Data$TTM[Option.Data$Type=="Put" & Option.Data$Underlying=="SPX"],
                                                  r = int,
                                                  d = 0,
                                                  sigma =
Option.Data$Volatility[Option.Data$Type=="Put" & Option.Data$Underlying=="SPX"],
                                                  N = 50
)

#Price American Call Options under Trinomial Tree
Option.Data$Trinomial[Option.Data$Type=="Call" & Option.Data$Underlying=="SPX"]<-
mapply(FUN = Tri.European.Call,
                                                  S0 =
Option.Data$Stock.Price[Option.Data$Type=="Call" & Option.Data$Underlying=="SPX"],
                                                  K =
Option.Data$Strike[Option.Data$Type=="Call" & Option.Data$Underlying=="SPX"],
                                                  TTM =
Option.Data$TTM[Option.Data$Type=="Call" & Option.Data$Underlying=="SPX"],
                                                  r = int,
                                                  d = 0,
                                                    sigma =
Option.Data$Volatility[Option.Data$Type=="Call" & Option.Data$Underlying=="SPX"],
                                                    N = 50
)

#Price American Put Options under Binomial Tree
Option.Data$Trinomial[Option.Data$Type=="Put" & Option.Data$Underlying=="SPX"] <-
mapply(FUN = Tri.European.Put,
                                                    S0 =
Option.Data$Stock.Price[Option.Data$Type=="Put" & Option.Data$Underlying=="SPX"],
                                                    K =
Option.Data$Strike[Option.Data$Type=="Put" & Option.Data$Underlying=="SPX"],
                                                    TTM =
Option.Data$TTM[Option.Data$Type=="Put" & Option.Data$Underlying=="SPX"],
                                                    r = int,
                                                    d = 0,
                                                    sigma =
Option.Data$Volatility[Option.Data$Type=="Put" & Option.Data$Underlying=="SPX"],
                                                    N = 50
)




#Final Data Table
write.csv(Option.Data, "Results_Output.csv")

#Test Using Bloomberg IVM
#Option.Data$Test.Option.Value[Option.Data$Type=="Call"] <- mapply(FUN =
Tri.American.Call,
#                                   S0 = Option.Data$Stock.Price[Option.Data$Type=="Call"],
#                                   K = Option.Data$Strike[Option.Data$Type=="Call"],
#                                   TTM = Option.Data$TTM[Option.Data$Type=="Call"],
#                                   r = int,
#                                   d = 0,
#                                   sigma = Option.Data$IVM[Option.Data$Type=="Call"]/100,
#                                   N = 50
#)


#Option.Data$Test.Option.Value[Option.Data$Type=="Put"] <- mapply(FUN = Tri.American.Put,
#                                    S0 = Option.Data$Stock.Price[Option.Data$Type=="Put"],
#                                    K = Option.Data$Strike[Option.Data$Type=="Put"],
#                                       TTM = Option.Data$TTM[Option.Data$Type=="Put"],
#                                       r = int,
#                                       d = 0,
#                                       sigma = Option.Data$IVM[Option.Data$Type=="Put"]/100,
#                                       N = 50
#)




###################################################################################
###########################
#   2.) Testing Functions against HULL Example
###################################################################################
###########################
#Test the pricer on the Example 21.1 in Hull. The tree should reproduce the numbers
#in the tree on page 21.3 for an American put option with parameters
#S0 = K = 50 , ?? = 0.4 , r = 0.1 , q = 0

#Defining a modification on Binomial Tree function to return the tree instead of the option
price, to match HULL example
American.Put.Tree <- function(S0, K, TTM, r, d, sigma, N){

  Xtree <- Binom.Tree (S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
  del.t <- TTM/N
  del.x <- sqrt((r-d-sigma^2/2)^2*del.t^2+sigma^2*del.t)
  pu <- 1/2 + 1/2*(r-d-sigma^2/2)*del.t/del.x
  pd <- 1 - pu

  Value_tree <- list()

  temp <- Xtree[[length(Xtree)]]
  terminal <- K-exp(temp)
  terminal <- mapply(FUN = max, terminal, 0)
  Value_tree[[length(Xtree)+1]] <- terminal

  #Note loop terminates at time 2, due to lookback from immediate excersise condition
  for(i in length(Xtree):2){
    temp1 <- vector("numeric", length = i)
    temp2 <- vector("numeric", length = i)
    temp <- vector("numeric", length = i)
    for(j in 1:i){
      #temp1 vector for expected option value working backwards
      temp1[j] <- exp(-r*del.t)*(pu*terminal[j]+pd*terminal[j+1])

      #temp2 vector for immediate excersise condition
      temp2[j] <- K-exp(Xtree[[i-1]][j])

      #temp vector to do the optimum of holding option or excersising at that time
      temp <- mapply(FUN = max, temp1, temp2)
    }
    terminal <- temp
    Value_tree[[i]] <- terminal
  }

  #Complete last iteration
  temp1 <- vector("numeric")
  temp2 <- vector("numeric")
  temp <- vector("numeric")

  temp1 <- exp(-r*del.t)*(pu*terminal[1]+pd*terminal[2])
  temp2 <- K-S0
  option.price <- mapply(FUN = max, temp1, temp2)
  Value_tree[[1]] <- option.price

  return(Value_tree)
}


# Display Stock Price Tree from Figure 21.3
lapply(FUN = exp, Binom.Tree(S0=50, K=50, TTM=5/12, r=0.1, d=0, sigma = 0.4, N=5))

# Display Option Value Tree from Figure 21.3
American.Put.Tree(S0 = 50, K = 50, TTM = 5/12, r = 0.1, d=0, sigma = 0.4, N = 5)

#Run test at 30, 50, 100, 500 timesteps, to match HULL

# N=30
American.Put(S0 = 50, K = 50, TTM = 5/12, r = 0.1, d=0, sigma = 0.4, N = 30)

# N=50
American.Put(S0 = 50, K = 50, TTM = 5/12, r = 0.1, d=0, sigma = 0.4, N = 50)

# N=100
American.Put(S0 = 50, K = 50, TTM = 5/12, r = 0.1, d=0, sigma = 0.4, N = 100)
```

```
# N=500
American.Put(S0 = 50, K = 50, TTM = 5/12, r = 0.1, d=0, sigma = 0.4, N = 500)


#Show that as n-> infinity, the price of an American Call approaches the BSM value

#Define Analytic BSM functions for testing results
BSM_call.div <- function(S0, K, TTM, r, d, sigma){
  dplus <- 1/sigma/sqrt(TTM)*(log(S0/K)+(r-d+sigma^2/2)*TTM)
  dmin <- dplus - sigma*sqrt(TTM)
  calc.price <- S0*exp(-d*TTM)*pnorm(q=dplus)-K*exp(-r*TTM)*pnorm(q=dmin)
  return(calc.price)
}

#Testing with N = 5, 10, 50, 100, 500
nvec <- c(5, 10, 50, 100, 500)
am_call_limit <- mapply(FUN = American.Call, S0 = 50, K = 50, TTM = 5/12, r = 0.1, d=0, sigma =
0.4, N = nvec)
bsm_call_results <- BSM_call.div(S0=50, K=50, TTM=5/12, r=0.1, d=0, sigma=0.4)
bsm_call_results <- rep(bsm_call_results, 5)

limit.df <- cbind(nvec, am_call_limit, bsm_call_results)
colnames(limit.df) <- c("N", "Binom Tree Call", "BSM Call")
limit.df



##################################################################################
##########################
#  Compute Greeks
##################################################################################
##########################

#Define a Function for Computing Delta, Calling the Binomial Tree
Binom_Delta <- function(S0, K, TTM, r, d, sigma, N, type){
  if (type == "Call"){
    Y1 = American.Call(S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
    Y2 = American.Call(S0=S0+0.1, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
  } else{
    Y1 = American.Put(S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
    Y2 = American.Put(S0=S0+0.1, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
  }
  Delta = (Y2-Y1)/0.1
  #Multiplying by 100% to be consistent with Bloomberg Reporting
  Delta = Delta*100
  return(Delta)
}
# Function Testing
#Binom_Delta(S0=100, K=100, TTM=1, r=0.05, d=0, sigma=0.2, N=50, type="Call")


#Define a Function for Computing Gamma, Calling the Binomial Tree
Binom_Gamma <- function(S0, K, TTM, r, d, sigma, N, type){
  if (type == "Call"){
    Y1 = American.Call(S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
    Y2 = American.Call(S0=S0+0.1, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
    Y3 = American.Call(S0=S0+0.2, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
  } else{
    Y1 = American.Put(S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
    Y2 = American.Put(S0=S0+0.1, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
    Y3 = American.Put(S0=S0+0.2, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
  }
  Delta1 = (Y2-Y1)/0.1
  Delta2 = (Y3-Y2)/0.1
  Gamma = (Delta2 - Delta1)/0.1

  #Multiplying by 100% to be consistent with Bloomberg Reporting
  Gamma = Gamma*100

  return(Gamma)
}
# Function Testing
#Binom_Gamma(S0=100, K=100, TTM=1, r=0.05, d=0, sigma=0.2, N=50, type="Put")



#Define a Function for Computing Vega, Calling the Binomial Tree
Binom_Vega <- function(S0, K, TTM, r, d, sigma, N, type){
  if (type == "Call"){
    Y1 = American.Call(S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
    Y2 = American.Call(S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma+0.005, N=N)
  } else{
    Y1 = American.Put(S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
    Y2 = American.Put(S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma+0.005, N=N)
  }
  Vega = (Y2-Y1)/0.005
  # Dividing by 100 % to be consistent with Bloomberg Reporting
  Vega = Vega/100
  return(Vega)
}
# Function Testing
#Binom_Vega(S0=100, K=100, TTM=1, r=0.05, d=0, sigma=0.2, N=50, type="Put")



#Define a Function for Computing Theta, Calling the Binomial Tree
Binom_Theta <- function(S0, K, TTM, r, d, sigma, N, type){
  if (type == "Call"){
    Y1 = American.Call(S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
    Y2 = American.Call(S0=S0, K=K, TTM=TTM-0.001, r=r, d=d, sigma=sigma, N=N)
  } else{
    Y1 = American.Put(S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
    Y2 = American.Put(S0=S0, K=K, TTM=TTM-0.001, r=r, d=d, sigma=sigma, N=N)
  }
  Theta = (Y2-Y1)/0.001

  #Diving by 365 days to be consistent with Bloomberg Reporting
  Theta = Theta/365
  return(Theta)
}
# Function Testing
#Binom_Theta(S0=100, K=100, TTM=1, r=0.05, d=0, sigma=0.2, N=50, type="Call")


#Define a Function for Computing Rho, Calling the Binomial Tree
Binom_Rho <- function(S0, K, TTM, r, d, sigma, N, type){
  if (type == "Call"){
    Y1 = American.Call(S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
    Y2 = American.Call(S0=S0, K=K, TTM=TTM, r=r+0.0001, d=d, sigma=sigma, N=N)
  } else{
    Y1 = American.Put(S0=S0, K=K, TTM=TTM, r=r, d=d, sigma=sigma, N=N)
    Y2 = American.Put(S0=S0, K=K, TTM=TTM, r=r+0.0001, d=d, sigma=sigma, N=N)
  }
  Rho = (Y2-Y1)/0.0001

  #Diving by 100% to be consistent with Bloomberg Reporting
  Rho = Rho/100
  return(Rho)
}
# Function Testing
```

```
#Binom_Rho(S0=100, K=100, TTM=1, r=0.05, d=0, sigma=0.2, N=50, type="Call")


# Read in Option Data from Bloomberg for Greek Comparison
Greeks.Data <- read.csv(file = "Greeks_Data.csv")


#Calculate Delta From Binomial Tree
Greeks.Data$Delta <- mapply(FUN = Binom_Delta,
                S0 = Greeks.Data$Stock.Price,
                K = Greeks.Data$Strike,
                TTM = Greeks.Data$TTM,
                r = int,
                d = 0,
                sigma = Greeks.Data$Vol,
                N = 50,
                type = Greeks.Data$Type)

#Calculate Gamma From Binomial Tree
Greeks.Data$Gamma <- mapply(FUN = Binom_Gamma,
                S0 = Greeks.Data$Stock.Price,
                K = Greeks.Data$Strike,
                TTM = Greeks.Data$TTM,
                r = int,
                d = 0,
                sigma = Greeks.Data$Vol,
                N = 50,
                type = Greeks.Data$Type)

#Calculate Vega From Binomial Tree
Greeks.Data$Vega<- mapply(FUN = Binom_Vega,
                S0 = Greeks.Data$Stock.Price,
                K = Greeks.Data$Strike,
                TTM = Greeks.Data$TTM,
                r = int,
                d = 0,
                sigma = Greeks.Data$Vol,
                N = 50,
                type = Greeks.Data$Type)

#Calculate Theta From Binomial Tree
Greeks.Data$Theta <- mapply(FUN = Binom_Theta,
               S0 = Greeks.Data$Stock.Price,
               K = Greeks.Data$Strike,
                    TTM = Greeks.Data$TTM,
                    r = int,
                    d = 0,
                    sigma = Greeks.Data$Vol,
                    N = 50,
                    type = Greeks.Data$Type)

#Calculate Rho From Binomial Tree
Greeks.Data$Rho <- mapply(FUN = Binom_Rho,
                     S0 = Greeks.Data$Stock.Price,
                     K = Greeks.Data$Strike,
                     TTM = Greeks.Data$TTM,
                     r = int,
                     d = 0,
                     sigma = Greeks.Data$Vol,
                     N = 50,
                     type = Greeks.Data$Type)

#Final Data Table
write.csv(Greeks.Data, "Greeks_Output.csv")

#####################################################################################
##########################
#   5.) Hedging Exercise
#####################################################################################
##########################
# Generate a sample of e.g. 10 daily stock prices Si following from the Black-Scholes model.
# For each day, compute: American put option prices P(ti), Delta of the option ???(ti).

#Plot the option prices P(ti) vs ti, and compare with the plot of the prices of the hedged
portfolio P(ti) ??? ???(ti)Si

#For this trial we consider a stock with S0 = 100, vol = 40%, K = 100, TTM = 1M, r = 5%, u = 8%

#Define a function to output a realization of the stock price by the BSM model, using Eulers
Method
BSM_Stock_Path <- function(n, S0, r, d, TTM, vol){
  dW <- rnorm(n, mean = 0, sd = 1)
  dT <- TTM/n
  Xnext <- log(S0)
  Xpath <- vector("numeric")
  Xpath[1] <- Xnext
  for(i in 1:n){
    Xnext = (r-d-1/2*vol^2)*dT+vol*dW[i]*sqrt(dT)+Xnext
    Xpath[i+1] <- Xnext
  }
  return(exp(Xpath))
}

# Generate 10 daily stock price realizations via the BSM model
stk_path <- BSM_Stock_Path(n=10, S0=100, r=0.05, d=0, TTM=1/12, vol = 0.4)

# Use American.Put Function to price options based on the Binomial Tree
opt_prices <- mapply(FUN = American.Put, S0 = stk_path, K = 100, TTM = 1/12, r = 0.05, d=0,
sigma = 0.4, N = 50)

# Compute the option delta using the Binomial Tree, and a finite difference dS of 0.1
inc_stk_path <- stk_path + 0.1
inc_opt_prices <-  mapply(FUN = American.Put, S0 = inc_stk_path, K = 100, TTM = 1/12, r =
0.05, d=0, sigma = 0.4, N = 50)
opt_deltas <- (inc_opt_prices - opt_prices)/0.1

#Create a vector of delta changes for computing hedge requirement
#shares <- opt_deltas[-1] - opt_deltas[-length(opt_deltas)]
#shares <- c(opt_deltas[1], shares)


# Create a vector of delta stock movements for hedged portfolio comparison
#delta_stk <- stk_path[-1] - stk_path[-length(stk_path)]
#delta_stk <- c(0, delta_stk)

# Calculate value of delta hedged portfolio
hedged_port = opt_prices - opt_deltas*stk_path
opt_pct_change = opt_prices / opt_prices[1] * 100 - 100
port_pct_change = hedged_port / hedged_port[1] * 100 - 100

# Result DataFrame
df <- cbind(1:length(stk_path)-1, stk_path, opt_prices, opt_deltas, hedged_port,
opt_pct_change, port_pct_change)
colnames(df) <- c("ti", "Si", "Pi", "Deltas", "Hedged Port", "Pi % Change", "Hedged % Change")
df
```