Project: American Options on GOOG

By Felipe Stanley, Jyoti Panday & Sishir Yerra Doddi

FE-620

8th October 2020

Introduction

Team 11 works aims at pricing American options on Alphabet Inc. (GOOG) as of December 4, 2020. For this project, The American Options on GOOG are priced in the Black-Scholes model using Binomial tree.

Alphabet Inc is a holding company that gives ambitious projects, resources, freedom and focus to make ideas happen. Alphabet was established in October 2015. The current CEO is Sundar Pichai. In 2019, Alphabet was valued at $162 Billion dollars. 80 percent of googles revenue comes from advertisement. The reason we choose google is because it doesn't pay dividends.

The project follows the following steps:

1. Binomial Tree Pricing of Alphabet using last 6 months of GOOG prices from Yahoo with the help of Python
2. Comparison of market data, Binomial Tree, and analytic formula Black-Scholes
3. Implied Volatility
4. Greek Calculations
5. Hedging of our position on google

Market Data Analysis:

Spot price of Alphabet Inc. On December 4th, 2020 GOOG closed at $S_0 = 1827.50$

Historical Volatility = 0.18

Risk free rate on December 4th, 2020 = .0750

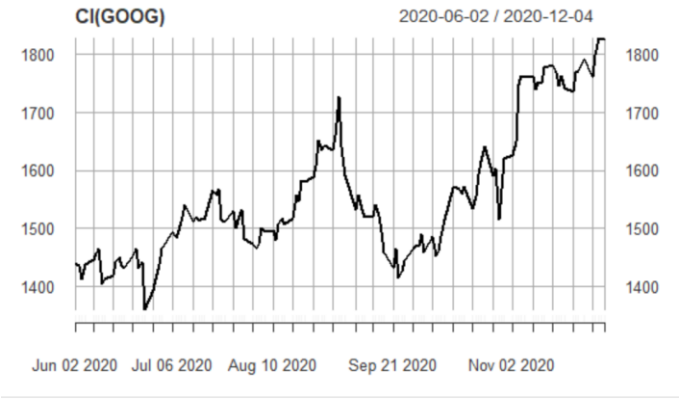


Figure 1: Closing price of GOOG for last 6 months

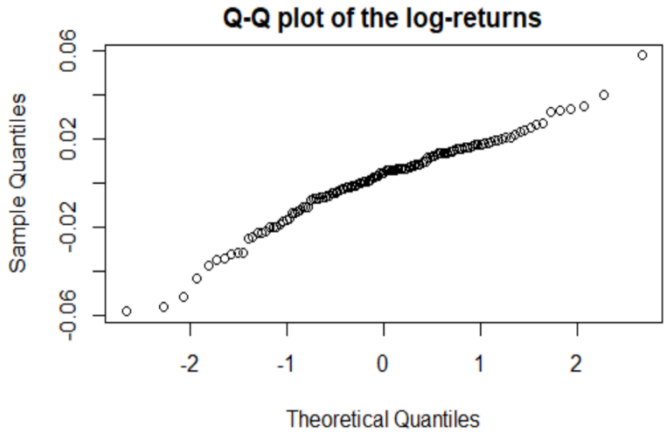


Figure 2: Q-Q plot of log-returns

Figure 3 shows the market data of 'GOOG' Options as of Friday 12/04/2020. Expirations date of 'At the Money' call option at 12/11/2020.

Contract Name	Last Trade Date	Strike	Last Price	Bid	Ask	Change	% Change	Volume	Open Interest	Implied Volatility
GOOG201211C01810000	2020-12-04 3:33PM EST	1,810.00	36.9	31.5	37.2	1.85	5.28%	15	186	29.31%
GOOG201211C01812500	2020-12-04 11:27AM EST	1,812.50	32.5	29.2	36.5	32.5	-	3	0	30.16%
GOOG201211C01815000	2020-12-04 2:41PM EST	1,815.00	30.3	28.5	33.9	-1.5	-4.72%	34	61	28.88%
GOOG201211C01817500	2020-12-04 3:31PM EST	1,817.50	31.83	26.4	33	31.83	-	3	0	29.43%
GOOG201211C01820000	2020-12-04 3:59PM EST	1,820.00	27.95	25.7	31	-1.15	-3.95%	66	92	28.74%
GOOG201211C01822500	2020-12-04 3:28PM EST	1,822.50	27.59	25.5	29.9	27.59	-	10	1	28.99%
GOOG201211C01825000	2020-12-04 3:59PM EST	1,825.00	25.25	23.1	28.1	-0.95	-3.63%	165	88	28.45%
GOOG201211C01827500	2020-12-04 3:00PM EST	1,827.50	24.9	21.1	27.3	24.9	-	31	2	28.94%
GOOG201211C01830000	2020-12-04 3:59PM EST	1,830.00	22.98	22	23.8	-0.17	-0.73%	100	193	26.50%
GOOG201211C01832500	2020-12-04 3:33PM EST	1,832.50	24.1	19	24.4	24.1	-	17	12	28.41%
GOOG201211C01835000	2020-12-04 3:54PM EST	1,835.00	20.48	18.3	22.8	-0.76	-3.58%	337	323	27.92%
GOOG201211C01837500	2020-12-04 3:54PM EST	1,837.50	20.05	16.5	22.4	20.05	-	4	5	28.68%
GOOG201211C01840000	2020-12-04 3:59PM EST	1,840.00	18	16.1	20.4	-2	-10.00%	519	331	27.68%
GOOG201211C01842500	2020-12-04 2:54PM EST	1,842.50	17.8	14	20.1	17.8	-	7	5	28.48%
GOOG201211C01845000	2020-12-04 3:37PM EST	1,845.00	16.83	13	19.1	16.83	-	10	24	28.48%
GOOG201211C01847500	2020-12-04 3:59PM EST	1,847.50	15.47	12.1	18.2	15.47	-	12	3	28.55%

To start pricing ‘GOOG’ options using Binomial Tree we need to stablish number of time steps and delta time for each of these steps. We are going to generate a binomial tree with 7 steps (‘n’ = 7) and each step represents one day.

$$T = 7/252$$

$$Dt = 1/252$$

Now, to compare Market Data, Binomial Tree, and analytic formula of Black-Scholes we are taking the list of strikes from table above and insert the following parameters to price models:

Binomial Inputs	
N	**7**
Dt	**0.003968**
S0	**1827.5**
Sigma	**0.18**
Rate	**0.01538**
K	**1827.5**

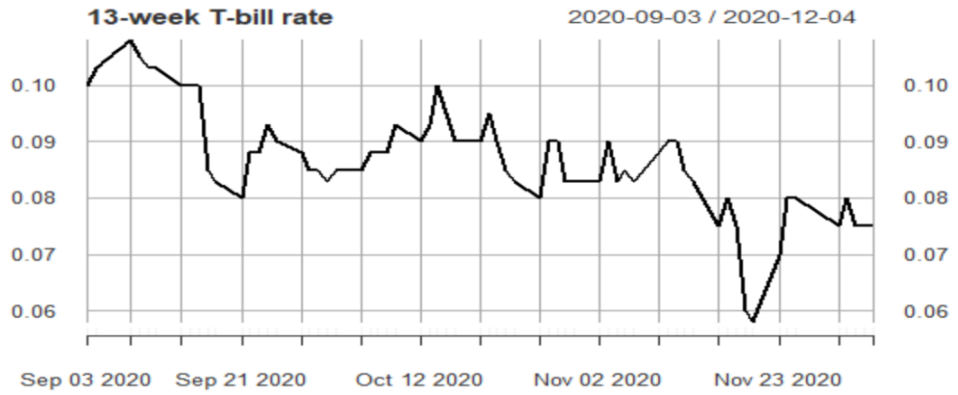


Figure 4: 13 weeks T-bill rate ending on December 4th

The Following Table Explains the Comparison of Market Put and Call prices with Binominal tree and Black-Scholes model prices of Put and Calls. Tree Call and Tree Put show Column show the results of pricing using tree model with n = 7 and

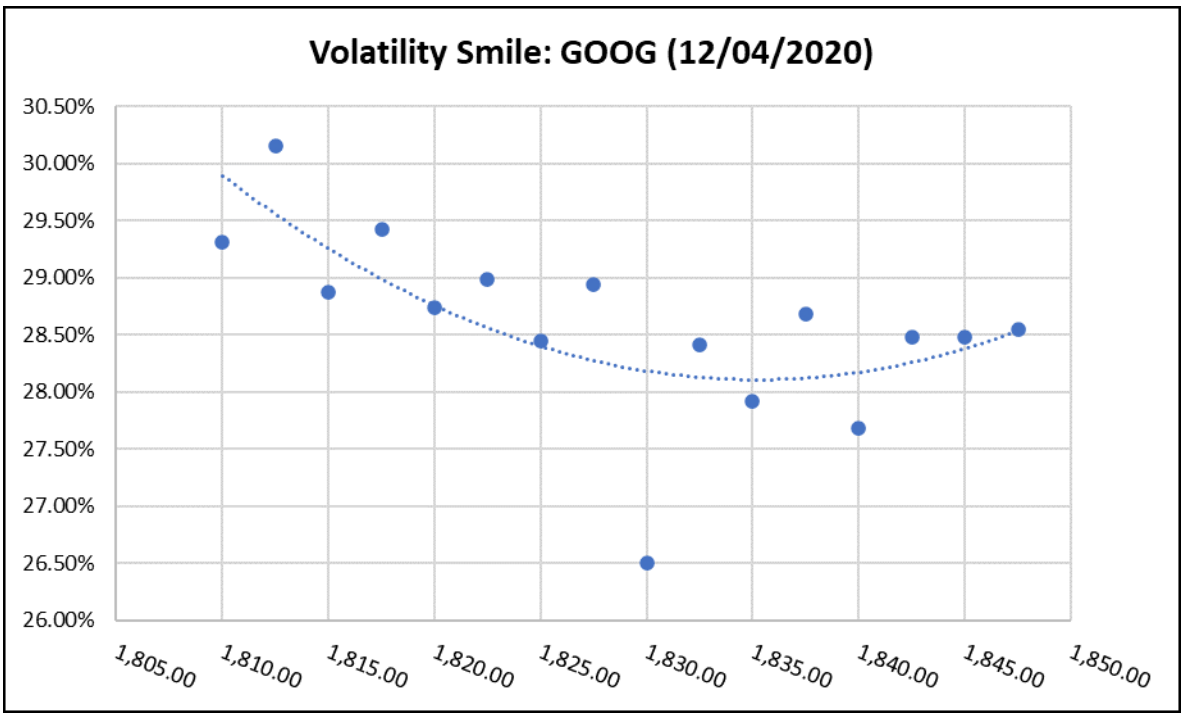
Historical Volatility estimated at 18% with a Risk Free Rate of 7.5% (Dec 4th).

Option closest to At The Money is shows in the marked box.

Model Comparison						
Strike	***Bid***	***Ask***	***Tree Call***	***BSM Call***	***Tree Put***	***BSM Put***
$1,810.00	**$ 36.90**	**$ 31.50**	$ 31.79	$ 32.11	$ 13.58	$ 13.83
$1,812.50	**$ 32.50**	**$ 29.20**	$ 30.54	$ 30.57	$ 14.83	$ 14.79
$1,815.00	**$ 30.30**	**$ 28.50**	$ 29.29	$ 29.07	$ 16.08	$ 15.79
$1,817.50	**$ 31.83**	**$ 26.40**	$ 28.05	$ 27.62	$ 17.33	$ 16.84
$1,820.00	**$ 27.95**	**$ 25.70**	$ 26.80	$ 26.21	$ 18.58	$ 17.93
$1,822.50	**$ 27.59**	**$ 25.50**	$ 25.55	$ 24.85	$ 19.84	$ 19.07
$1,825.00	**$ 25.25**	**$ 23.10**	$ 24.30	$ 23.53	$ 21.09	$ 20.25
$1,827.50	**$ 24.90**	**$ 21.10**	$ 23.05	$ 22.26	$ 22.34	$ 21.48
$1,830.00	**$ 22.98**	**$ 22.00**	$ 21.80	$ 21.03	$ 23.59	$ 22.75
$1,832.50	**$ 24.10**	**$ 19.00**	$ 20.55	$ 19.85	$ 24.84	$ 24.07
$1,835.00	**$ 20.48**	**$ 18.30**	$ 19.30	$ 18.72	$ 26.09	$ 25.43
$1,837.50	**$ 20.05**	**$ 16.50**	$ 18.06	$ 17.63	$ 27.34	$ 26.84
$1,840.00	**$ 18.00**	**$ 16.10**	$ 16.81	$ 16.58	$ 28.59	$ 28.29
$1,842.50	**$ 17.80**	**$ 14.00**	$ 15.56	$ 15.58	$ 29.84	$ 29.79
$1,845.00	**$ 16.83**	**$ 13.00**	$ 14.31	$ 14.62	$ 31.09	$ 31.33
$1,847.50	**$ 15.47**	**$ 12.10**	$ 13.06	$ 13.70	$ 32.38	$ 32.91

Figure 5

The Following Table Shows the Implied Volatility of the American Option GOOG as of December 4th, 2020 (This has to be compared with Implied Volatility of the Expiry date December 11th, 2020)


Volatility Smile: GOOG (12/04/2020)
30.50%
30.00%
29.50%
29.00%
28.50%
28.00%
27.50%
27.00%
26.50%
26.00%
1,805.00
1,810.00
1,815.00
1,820.00
1,825.00
1,830.00
1,835.00
1,840.00
1,845.00
1,850.00

Greeks

Since ‘GOOG’ does not pay dividends Black-Scholes model for European Option is a good approximation of its Greeks value. To validate this statement, we conducted a test where we manually computed Delta of American Call option using the following equation.

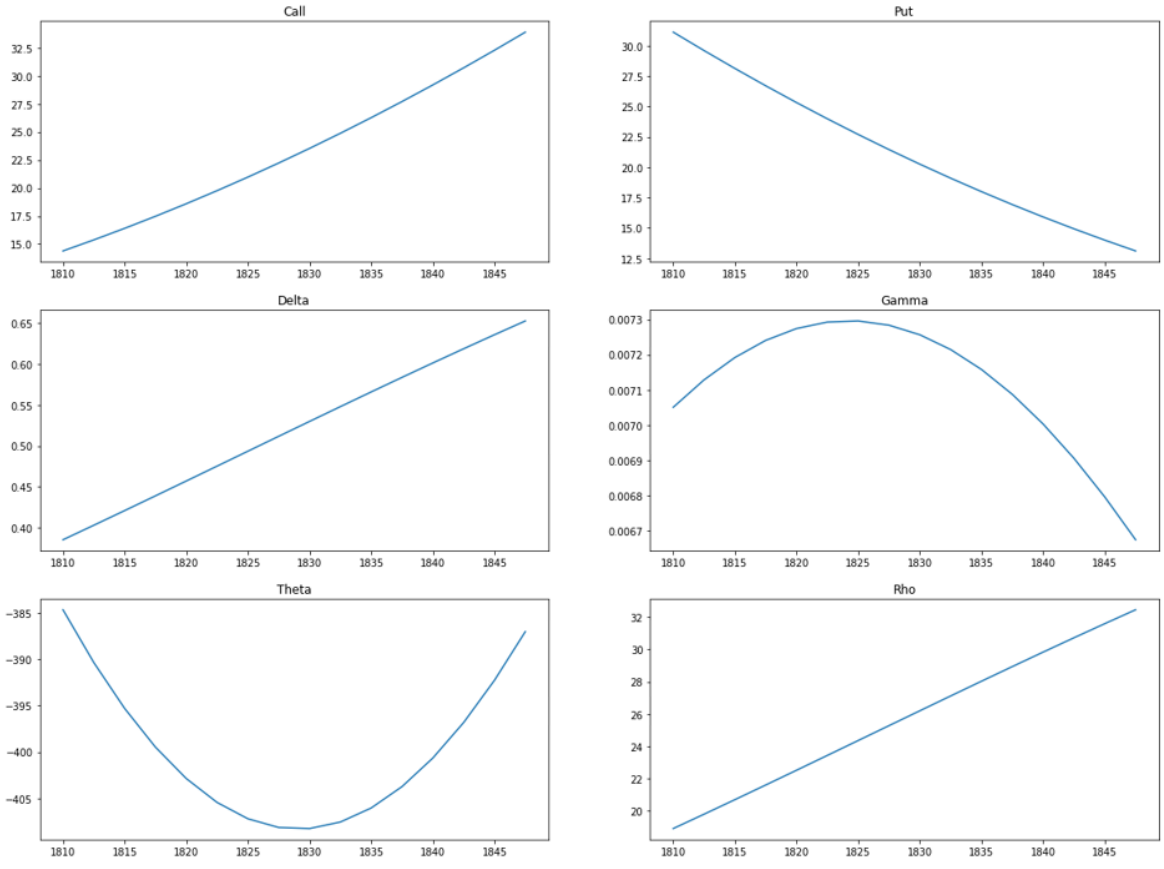
$$D = (P(S0+0.1)-P(S0-0.1))/0.2$$

And we compared its value against Black-Scholes Model (see appendix). Both approaches yield Delta of roughly 0.55 which suggests that a one dollar up move on the underlying asset would result on a fifty-five cents up move on the call premium.

To compute the equivalent European Option price through Black-Scholes Model to the price we obtained for the American Option via Binomial Tree we needed to make one alteration on the input value. Delta Time on the Binomial Tree was set as (1/252) because each time step was represented moving one business day forward on our time horizon, and there were 7 steps before maturity date.

To generate a similar result using Black-Scholes Model, Delta Time was set as (7/252) because there were ten days before expiration and there are 252 business days in one year. The following table and graph illustrate complete list of Greeks of ‘GOOG’ options for the series of strikes we retrieved from market data.

Options' Greeks							
Spot	***Call***	***Put***	***Delta***	***Gamma***	***Vega***	***Theta***	***Rho***
$1,810.00	**14.38632**	**31.10792**	0.385149	0.00705	115.1637	-384.678	18.9117
$1,812.50	**15.37131**	**29.59291**	0.402875	0.007128	116.7606	-390.361	19.80106
$1,815.00	**16.40084**	**28.12245**	0.420778	0.007192	118.127	-395.3	20.70054
$1,817.50	**17.47532**	**26.69692**	0.438822	0.007241	119.2552	-399.47	21.60832
$1,820.00	**18.59504**	**25.31664**	0.456969	0.007274	120.1393	-402.85	22.52255
$1,822.50	**19.76022**	**23.98182**	0.475181	0.007293	120.7747	-405.424	23.44132
$1,825.00	**20.97097**	**22.69257**	0.49342	0.007296	121.1585	-407.183	24.3627
$1,827.50	**22.22731**	**21.44891**	0.511647	0.007284	121.2892	-408.119	25.28479
$1,830.00	**23.52916**	**20.25077**	0.529826	0.007257	121.1672	-408.234	26.20565
$1,832.50	**24.87636**	**19.09797**	0.547918	0.007214	120.7942	-407.532	27.12337
$1,835.00	**26.26865**	**17.99025**	0.565886	0.007158	120.1735	-406.022	28.03607
$1,837.50	**27.70566**	**16.92727**	0.583695	0.007087	119.31	-403.72	28.94192
$1,840.00	**29.18696**	**15.90857**	0.60131	0.007003	118.21	-400.644	29.83911
$1,842.50	**30.71202**	**14.93363**	0.618698	0.006905	116.8811	-396.818	30.72592
$1,845.00	**32.28024**	**14.00184**	0.635826	0.006795	115.3323	-392.272	31.60069
$1,847.50	**33.89091**	**13.11252**	0.652665	0.006674	113.5736	-387.036	32.46182

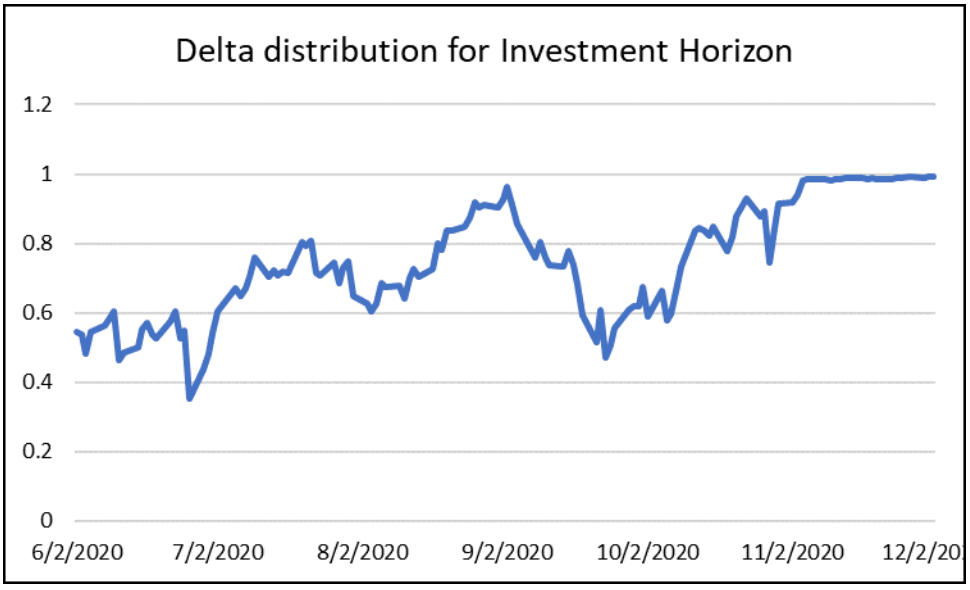


Delta Hedging

Lastly, we need to develop a hedging mechanism. For that, lets suppose we are selling GOOG's call option and to protect this initial position we are going to sell short shares of the underlying asset to mitigate potential losses resulted from a rally in price of the spot price.

This mechanism is also known as `Delta Hedging` and it receives this name because the value of the option's delta dictates the number of shares need to hold to offset market risk of initial option's position.

Our Delta Hedging simulation is initiating a position on 06/02/2020 and we are going to use historical price of GOOG to compute delta of the option for that investment horizon. Figure below displays the distribution of delta:



Next table is showing how Delta was calculated for the first 17-time steps.

t	Time	Date	S(t)	T-t	d1	d2	Call	Delta
0	0	6/2/2020	1439.22	1	0.1266665	0.006666	74.4703	0.546012
1	0.003968	6/3/2020	1436.38	0.996032	0.10992203	-0.00984	72.75821	0.539449
2	0.007937	6/4/2020	1412.18	0.992063	-0.0325234	-0.15205	60.12595	0.483177
3	0.011905	6/5/2020	1438.39	0.988095	0.12107443	0.001791	73.5381	0.543868
4	0.015873	6/8/2020	1446.61	0.984127	0.16867998	0.049636	77.96185	0.56253
5	0.019841	6/9/2020	1456.16	0.980159	0.22389876	0.105095	83.3194	0.583985
6	0.02381	6/10/2020	1465.85	0.97619	0.27978465	0.161222	88.96666	0.605432
7	0.027778	6/11/2020	1403.84	0.972222	-0.0854638	-0.20379	55.38082	0.462336
8	0.031746	6/12/2020	1413.18	0.968254	-0.0299909	-0.14807	59.68086	0.484271
9	0.035714	6/15/2020	1419.85	0.964286	0.00939477	-0.10844	62.83121	0.499877
10	0.039683	6/16/2020	1442.72	0.960317	0.1447828	0.027188	74.80954	0.553292
11	0.043651	6/17/2020	1451.12	0.956349	0.19403955	0.076688	79.41254	0.57253
12	0.047619	6/18/2020	1435.96	0.952381	0.10424973	-0.01286	70.77147	0.537404
13	0.051587	6/19/2020	1431.72	0.948413	0.07864773	-0.03822	68.33566	0.527327
14	0.055556	6/22/2020	1451.86	0.944444	0.19807862	0.08146	79.35155	0.574154
15	0.059524	6/23/2020	1464.41	0.940476	0.27193747	0.155564	86.62876	0.602614
16	0.063492	6/24/2020	1431.97	0.936508	0.07909133	-0.03704	67.98355	0.527553

Expanding on the information of previous table, we can compute the number of shares we need to hold to offset Market Risk of our option's position.

Shares Purchased	Daily Cost	Cumulative
546.0122021	$ 785.83	$ 785.83
-6.563444357	$ (9.43)	$ 776.40
-56.27141953	$ (79.47)	$ 696.94
60.69043702	$ 87.30	$ 784.24
18.66175883	$ 27.00	$ 811.23
21.45524339	$ 31.24	$ 842.47
21.44721177	$ 31.44	$ 873.91
-143.095604	$ (200.88)	$ 673.03
21.93503661	$ 31.00	$ 704.03
15.60539021	$ 22.16	$ 726.18
53.41495165	$ 77.06	$ 803.25
19.23866612	$ 27.92	$ 831.16
-35.12616164	$ (50.44)	$ 780.72
-10.07688677	$ (14.43)	$ 766.30
46.82636236	$ 67.99	$ 834.28
28.46015591	$ 41.68	$ 875.96
-75.0611903	$ (107.49)	$ 768.47

This yields the following result:

Sell Shares	$	1,827,949.95
Cost	$	1,568,454.91
Hedge Cost	**$**	**(259,495.04)**

The last row of the table above is showing we did not stablish a perfect hedging mechanism. This residual error can be attributed to the fact Binomial Tree as well as Black- Scholes models are only an approximation of market behavior and not an exact prediction. This approximation is evident by the fact that our models are assuming constant volatility, and that expected rate of return for the underlying asset is the same rate as risk -free rate which not only has the same value as T-Bill rates for 13 rate and that it is also constant.

Plot Code using R-Studio.

```
# Historical Data of GOOG
getSymbols("GOOG", src="yahoo", from="2020-06-02", to="2020-12-02")
# visualizing the Data
names(GOOG)
plot(Cl(GOOG))
# Calculating Historical Volatility using log of past prices
prices <- Cl(GOOG)
head(prices)
tail(prices)
ndays <- length(prices)
#computing daily log-returns
logret <- periodReturn(prices, period="daily", type="log")
plot(logret, main="GOOG daily log returns")
# check the normality of the log-returns
qqnorm(logret, main="Q-Q plot of the log-returns")

# Risk-free interest rate ^IRX = 13-week T-bill rate
getSymbols("^IRX", src="yahoo", from="2020-09-03", to="2020-12-02")
names(IRX)
rfrate <- na.omit(Cl(IRX))
nrf <- length(rfrate)
plot(rfrate, main="13-week T-bill rate")
tail(rfrate)
```

Python Part of the Project

Modifications we made

1. created an array ('vector') to store results of calculations of 'dt', 'u', 'd', and 'p'.

```
dt = T/N
u = numpy.exp(sigma*numpy.sqrt(dt))
d = 1/u
p = (numpy.exp(r*dt)-d)/(u-d)
vector = [dt, u, d, p, 1-p]
```

2. Created two new variables 'pv_price' and 'exercise'. The first one is storing the calculation of present value of node's option price. The second one is storing the calculation of payout of early exercise. Then, inserted the highest of them on the 'option' array.

```
        pv_price = numpy.exp(-r*dt)*(p*option[j, i+1]+(1-p)*option[j+1, i+1]) # present value for current node
        exercise = numpy.maximum(price_tree[j, i]-k,0)                        # payoff of early exercise
        option[j, i] = numpy.maximum(pv_price, exercise)                      # populate option array
```

3. Used the same approach taken to declare and populate 'option' array to create an additional array to store early payoff of each node of the tree. This new array was created so it can be used to validate algorithm.

```
early = numpy.zeros([N+1, N+1])
for i in numpy.arange(N-1, -1, -1):
    for j in numpy.arange(0, i+1):
        pv_price = numpy.exp(-r*dt)*(p*option[j, i+1]+(1-p)*option[j+1, i+1])
        early[j, i] = numpy.maximum(price_tree[j, i]-k, 0)
```

4. Transformed all three arrays into a Data Frame object so we can have easy access, visualize, and manipulate their data later. In addition to these three Data Frames we also included one variable 'price' for final price of the option and array 'vector' on a list of all the outputs of the function.

```
#..................................................................... output
price     = option[0,0]
spot_df   = pd.DataFrame(price_tree)
option_df = pd.DataFrame(option)
early_df  = pd.DataFrame(early)
#.....................................................................................................


if complete:
    return [price, spot_df, option_df, early_df, vector]
else:
    return price
```

How we validated the algorithm

1. The main modification we made on original code is including computation early exercise payoff, and to validate this modification yielded correct result we pasted Data Frame 'early_df' on excel while also inserting data frame 'option_df' from original code (which does not compute early exercise) and manual compared them to make sure final code is computing it correctly.

Modifications we still need to perform

1. Include Put Option's payoff.
2. Modify calculation of delta time 'dt' so we can have more flexibility adjusting it for our own convenience.

Complete Source Code

```
def binom_tree_call(N, T, S0, sigma, r, k, complete=False):

    #..................................................... tree's parameters
    dt = T/N
    u = numpy.exp(sigma*numpy.sqrt(dt))
    d = 1/u
    p = (numpy.exp(r*dt)-d)/(u-d)
    vector = [dt, u, d, p, 1-p]
    #.....................................................................................


    #..................................................... price matrix
    price_tree = numpy.zeros([N+1, N+1])
    for i in range(N+1):
        for j in range(i+1):
            price_tree[j, i] = S0*(d**j)*(u**(i-j))
    #.....................................................................................


    #..................................................... option's value at maturity
    option = numpy.zeros([N+1, N+1])
    option[:, N] = numpy.maximum(numpy.zeros(N+1), price_tree[:, N]-k)
    #.....................................................................................


    #..................................................... option's value at T= 0
    for i in numpy.arange(N-1, -1, -1):
        for j in numpy.arange(0, i+1):
            pv_price = numpy.exp(-r*dt)*(p*option[j, i+1]+(1-p)*option[j+1, i+1]) # present value for current node
            exercise = numpy.maximum(price_tree[j, i]-k,0)
            option[j, i] = numpy.maximum(pv_price, exercise)
    #.....................................................................................


    #..................................................... early exercise's value
    early = numpy.zeros([N+1, N+1])
    for i in numpy.arange(N-1, -1, -1):
```

```
    for j in numpy.arange(0, i+1):
        pv_price = numpy.exp(-r*dt)*(p*option[j, i+1]+(1-p)*option[j+1, i+1]) # present value for current node
        early[j, i] = numpy.maximum(price_tree[j, i]-k, 0)
#.....................................................................................................


#..................................................... output
price     = option[0,0]
spot_df   = pd.DataFrame(price_tree)
option_df = pd.DataFrame(option)
early_df  = pd.DataFrame(early)
#.....................................................................................................


if complete:
    return [price, spot_df, option_df, early_df, vector]
else:
    return price
```