Monte Carlo Simulation in Option Profit Testing

Group 13: Beihan Niu, Kemei Zhuo, Zachary Cleary, Nicholas Liotti, Jinning Liu

May 3, 2018

1. Introduction

This project assesses how effective simulation is in predicting option profits. We used two datasets in our profit testing: the historical S&P 500 Index price from January 20th, 2017 to April 10th, 2018 and weekly option prices from April 10th, 2018 listed on Cboe Option Exchange Website. These weekly options all have a specified date of April 20th, 2018.

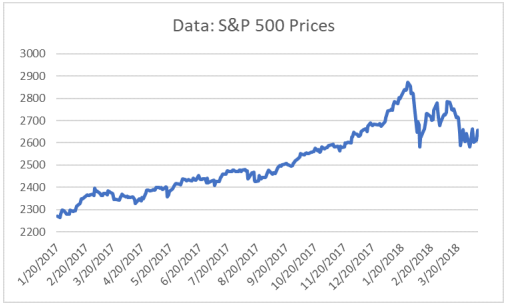


Figure 1: S&P prices over the period of intrest

Calls — APRIL 2018 (EXPIRATION: 04/20)

Strike	Last	Net	Bid	Ask	Vol	Int
SPX1820D2635-E	36.10	-9.05	32.90	34.70	27	483
SPXW1820D2635-E	38.30	-13.35	34.50	35.70	118	286
SPX1820D2640-E	31.00	-19.00	29.90	31.80	56	2239
SPXW1820D2640-E	32.28	-16.09	31.70	32.70	155	540
SPX1820D2645-E	28.60	-10.20	27.20	28.90	50	3137
SPXW1820D2645-E	29.55	-17.45	28.90	29.80	123	490
SPX1820D2650-E	26.00	-13.00	24.50	26.20	7267	38578
SPXW1820D2650-E	27.40	-10.10	26.20	27.00	636	1771

Puts — APRIL 2018 (EXPIRATION: 04/20)

Strike	Last	Net	Bid	Ask	Vol	Int
SPX1820P2635-E	21.94	-1.36	26.70	28.40	10	2720
SPXW1820P2635-E	26.40	+0.12	28.30	29.10	101	671
SPX1820P2640-E	28.50	+2.70	28.70	30.50	236	3687
SPXW1820P2640-E	30.42	+4.07	30.30	31.30	201	1144
SPX1820P2645-E	32.40	+3.30	30.80	32.70	82	2105
SPXW1820P2645-E	32.00	+2.00	32.40	33.50	167	335
SPX1820P2650-E	33.45	+0.45	33.10	35.00	6375	39794
SPXW1820P2650-E	34.39	+2.69	34.70	35.80	633	2382

Figure 2: Options

Options are a type of contract that gives the buyer the right, but not the obligation, to buy or sell an underlying asset at a specified strike price on a specified date. Option trading is very popular due to its four advantages. First, options are cost efficient, which means an investor pays less for an option than a stock. Second, options can be used to reduce risks. Third, options have higher potential returns. Fourth, different options can be combined to reduce risks and increase the potential profit. Due to these advantages, we chose to test option profits instead of stocks in our project.

There are two basic types of options: call and put. A call option is a financial contract between two parties: buyer and seller that gives the buyer the right, but not the obligation, to buy a stock, bond, commodity or other instrument at a specified price on a specific date. A put option is a financial contract between two parties: buyer and seller that gives the buyer the right, but not the obligation, to sell a stock, bond,

commodity or other instrument at a specified price on a specific date. An investor can choose to buy an option, short sell an option or make any combination of these two basic options with different strike price.

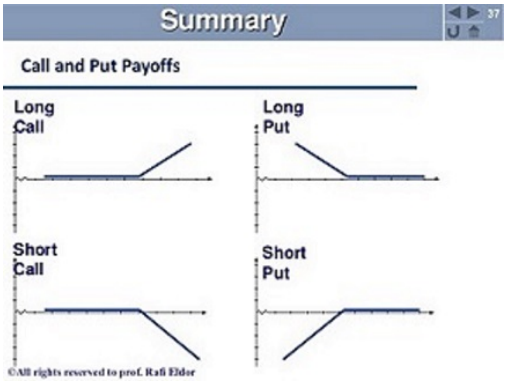


Figure 3: Payoff by buying or selling of one call or put option

We used these options to form four different strategies. Our first strategy is to buy one call at strike price K. The strategy captures the profit of a rising stock price and also reduces the possible loss to the price an investor would pay to buy this call option. Our second strategy is to sell one put at strike price K. With this strategy, an investor would gain certain amount of money by short selling the put option but he is exposed to possible loss. Our third strategy is to buy one call at strike price at K_1 and sell one put at strike price $K_2(K1 \geqslant K_2)$. This strategy would win investors profits when stock price is higher than K_1 and it protects itself from loss if stock price falls within K_1 and K_2. The investor would have a loss if the stock price is lower than K_2. Our fourth strategy is to buy one call at K_1 and sell one call at $K_2(K_1 > K_2)$. This strategy would limit both the profit and loss if the stock price fall out of K_1 and K_2.

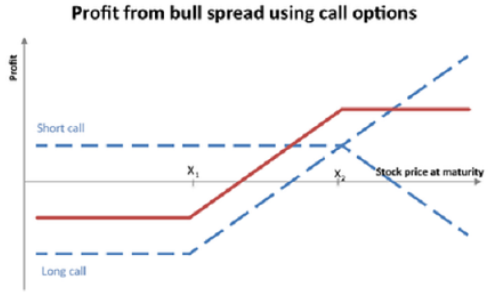


Figure 4: Strategy 3: Bull Spread

Our goal is to first predict the S&P 500 index price on April 20th, 2018 based on historical S&P 500 index and then compare the profits of different option combinations under simulated S&P 500 index price and real S&P 500 index price.

2. Methods and Results

We first used Monte Carlo Simulation to simulate the S&P 500 price on April 20th, 2018 and from that derived the option combination profits. Secondly, we used Bayesian Monte Carlo Integration to simulate the stock price on April 20th, 2018 and recalculated the option combination profits. In the end, we compared our two predictions with the real option profits given the actual S&P 500 index price of April 20th, 2018.

Our stock stimulation is based on the assumption that stock prices follow Geometric Brownian Motion. Geometric Brownian Motion is widely used in stock market because of many fundamental similarities between

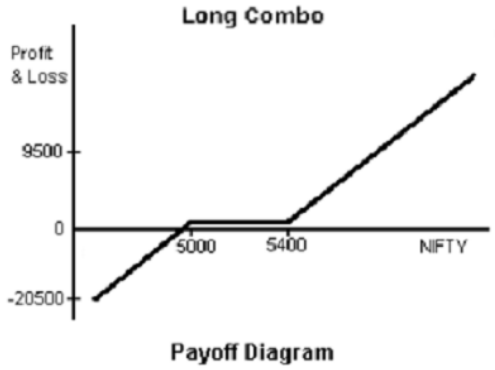


Figure 5: Strategy 4: Long Combo

GBM process and equity prices. One example would be that the expected returns from GBM processes are independent of the initial value just like it happens in the stock market. Another example would be that GBM exhibit volatility similar to stock prices.

This is the Geometric Brownian equation. S_t stands for the stock price at time t and for any fixed $t > 0$ and $S_t > 0$, we have that:

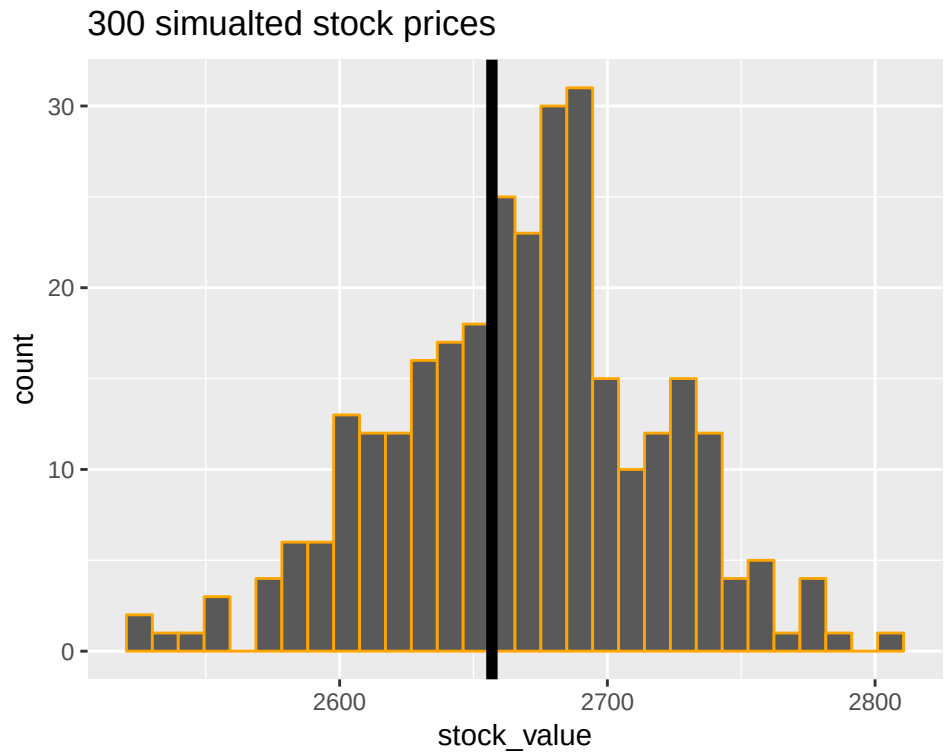
$$S_t = S_0 \exp\{(\mu - \frac{\sigma^2}{2}t + W_t)\},$$

where W_t is assumed to be normally distributed as $\mathcal{N}(0, \sigma^2 t)$. To obtain the mu and sigma in GBM equation, we first calculated the index daily return. The formula to calculate daily return is: $\frac{S_{i+1}}{S_i} - 1$, S_i is the close price at date i. The μ is calculated as the mean of daily return and the σ is the standard deviation of daily return.

2.1 Monte Carlo Simulation and Profit Calculation

2.1.1 Monte Carlo Simulation

We took the index price on April 10th, 2018 as our initial stock price. We then randomly generated 8 consecutive daily stock prices and kept the stock price on the last day. We repeated this process 300 times and the results are displayed in the histogram below. The black line here is the index price on April 10th, 2018. The simulated price is concentrated in the middle with some located far away at two sides.



2.1.2 Profit Calculation

The profits of different option combinations are calculated according to the following formulas:

- Buy one call at strike price K: Profit $= \text{Max}(S - K, 0) - c$
- Sell one put at strike price K: Profit $= -\text{Max}(K - S, 0) + p$
- Buy one call at strike price K_1 and sell one put at strike price K_2: Profit $= \text{Max}(S - K_1, 0) - c - \text{Max}(K_2 - S, 0) + p$
- Buy one call at strike price K_1 and sell one call at strike price K_2: Profit $= \text{Max}(S - K_1, 0) - c_1 - \text{Max}(S - K_2, 0) + c_2$

(S is stock price on the specified date; c is the price of call option; p is the price of put option)

The predicted profit for each approach is summarized in the table below. The options are arranged in the order they are presented above, with four prices considered per strategy. The third strategy: buying a call and selling a put, stands out as having relatively high predicted profits.

option1	option2	Profit Mean	Profit Sd
buy a 2635 call		5.488	0.068
buy a 2640 call		4.757	0.066
buy a 2645 call		4.221	0.063
buy a 2650 call		3.475	0.061
sell a 2635 put		20.518	0.034
sell a 2640 put		21.242	0.037
sell a 2645 put		21.906	0.041
sell a 2650 put		22.648	0.043
buy a 2635 call	sell a 2635 put	25.979	0.088
buy a 2640 call	sell a 2635 put	25.379	0.085
buy a 2640 call	sell a 2640 put	26.034	0.089
buy a 2645 call	sell a 2635 put	24.691	0.085
buy a 2645 call	sell a 2640 put	25.349	0.086
buy a 2645 call	sell a 2645 put	26.183	0.087
buy a 2650 call	sell a 2635 put	29.864	0.080
buy a 2650 call	sell a 2640 put	24.481	0.083
buy a 2650 call	sell a 2645 put	25.321	0.086
buy a 2650 call	sell a 2650 put	26.248	0.090
buy a 2635 call	sell a 2640 call	-0.280	0.125
buy a 2635 call	sell a 2645 call	0.476	0.250
buy a 2635 call	sell a 2650 call	1.219	0.378
buy a 2640 call	sell a 2640 call	0.756	0.129
buy a 2640 call	sell a 2645 call	1.499	0.260
buy a 2645 call	sell a 2650 call	0.743	0.133

2.1.3 Bootstrap and Jackknife Validation of Monte Carlo Results

The μ and σ estimates are of central importance to the Monte Carlo Simulation. Thus, we utilized jackknife analysis to find the standard error of our parameter estimates. Using this we calculated a 95% confidence interval for our parameter estimates. As can be seen in the table below the jackknife estimate of error for both parameters is relatively small. We then input the extremes of these bounds into our simulation to test its sensitivity to changes in the parameter estimates. Fortunately, the final results displayed below indicate that our model is not overly sensitive to variance in the parameters.

	95% Lower Bound	Estimate	95% Upper Bound
Mu	0.000	0.001	0.001
Sigma	0.006	0.007	0.008

	95% Lower Bound	Estimate of Mean	95% Upper Bound
Stock Price(Mu tested)	2650.399	2668.421	2681.540
Stock Price(Sigma tested)	2664.483	2668.421	2667.789

We also were concerned that the Monte Carlo Simulation might not be internally consistent. To evaluate if this was an issue we recalculated the parameters at the end of the simulation period and compared them to our initial results. We used bootstrap re-sampling on the simulated returns so as to get a very large sample without undue computational stress. The jackknife after bootstrap provides us with an estimate of the standard error of our bootstrapped parameter estimates. We can use simple z-test for difference of means to compare the new and old estimates. This returns a p-value of 0.2202653 which indicates no significant difference. Thus, our concern that the model was internally inconsistent is addressed. Although the new

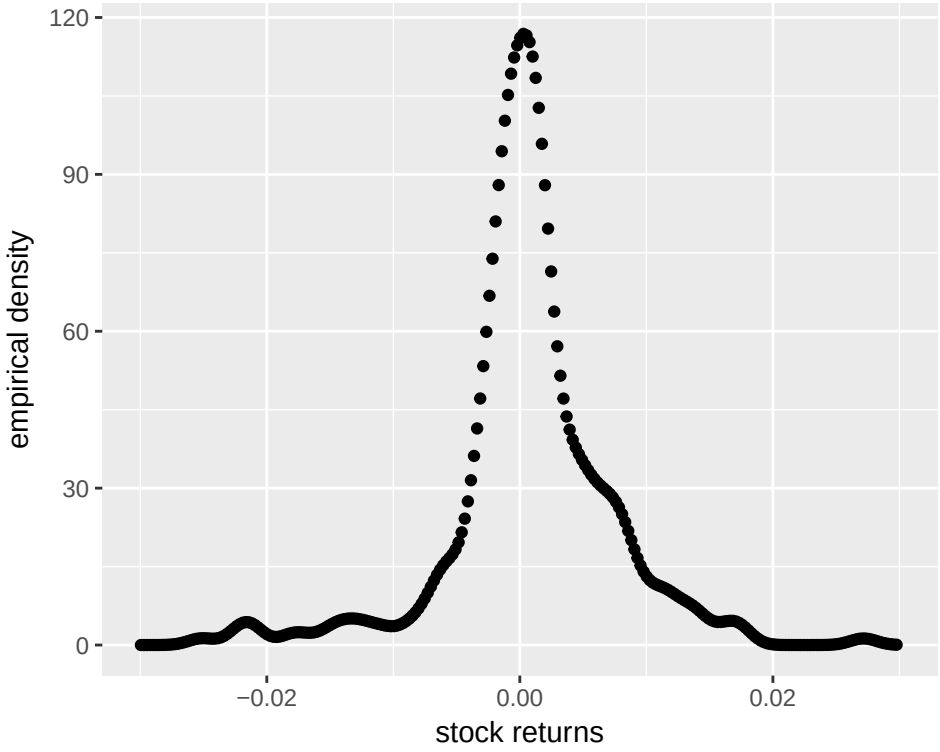
estimates might at first appear to differ from the old, when we account for error in our estimate the difference is insignificant.

	Original	New	SE of New
Mu	0.001	0.000	0
Sigma	0.007	0.007	0

2.2 Bayesian Monte Carlo Integration and Profit Calculation

2.2.1 Bayesian Monte Carlo Integration

Additionally, we conducted Bayesian analysis of the stock prices on a specified date of April 20th, 2018. Given that there should be a random parameter and prior distribution in Bayesian statistics, we used the parameter μ in Geometric Brownian Motion as a random variable and it follows a prior distribution. Moreover, conditioning on μ, the future stock prices follow Geometric Brownian Motion with σ estimated using MLE of the history data. As can be seen in the scatter-plot of history data for the stock returns, it shows a pattern similar to Laplace distribution. Therefore, we used Laplace distribution to fit the prior distribution.




```
## [1] 2657.004
```

After fitting the prior distribution, we figured out the posterior distribution given stock prices. Here, we worked out the posterior distribution using the last several records of stock prices, and the Laplace distribution as prior distribution of μ. Based on this posterior distribution, we calculated the expected stock price on April 20th, 2018 and the expected profits of different option strategies. Hence we compared the outcome of the two different methods.

2.2.2 Profit Calculation

After we obtained the estimated stock price from Bayesian Monte Carlo Simulation, we used it as a criterion to calculate the profit of 21 options. Recall the selection is comprised of four options simply buying a call,

four options simply selling a put, ten options including buying a call and selling a put, and three options including buying a call and selling a call. The formulas we used to calculate the profit was mentioned before under the part of "Monte Carlo Simulation and Profit Calculation". We calculated the profit of those 21 options using Bayesian Monte Carlo and Monte Carlo Simulation, and we tried to figure out the differences between them and real world profits.

option1	option2	Profit Mean
buy a 2635 call		-13.83
buy a 2640 call		-15.83
buy a 2645 call		-17.93
buy a 2650 call		-20.13
sell a 2635 put		28.30
sell a 2640 put		30.30
sell a 2645 put		32.40
sell a 2650 put		34.70
buy a 2635 call	sell a 2635 put	14.47
buy a 2640 call	sell a 2635 put	12.47
buy a 2640 call	sell a 2640 put	14.57
buy a 2645 call	sell a 2635 put	10.37
buy a 2645 call	sell a 2640 put	12.47
buy a 2645 call	sell a 2645 put	14.47
buy a 2650 call	sell a 2635 put	8.17
buy a 2650 call	sell a 2640 put	9.27
buy a 2650 call	sell a 2645 put	11.37
buy a 2650 call	sell a 2650 put	14.57
buy a 2635 call	sell a 2640 call	1.00
buy a 2635 call	sell a 2645 call	3.20
buy a 2635 call	sell a 2650 call	5.50
buy a 2640 call	sell a 2640 call	2.20
buy a 2640 call	sell a 2645 call	4.50
buy a 2645 call	sell a 2650 call	2.30

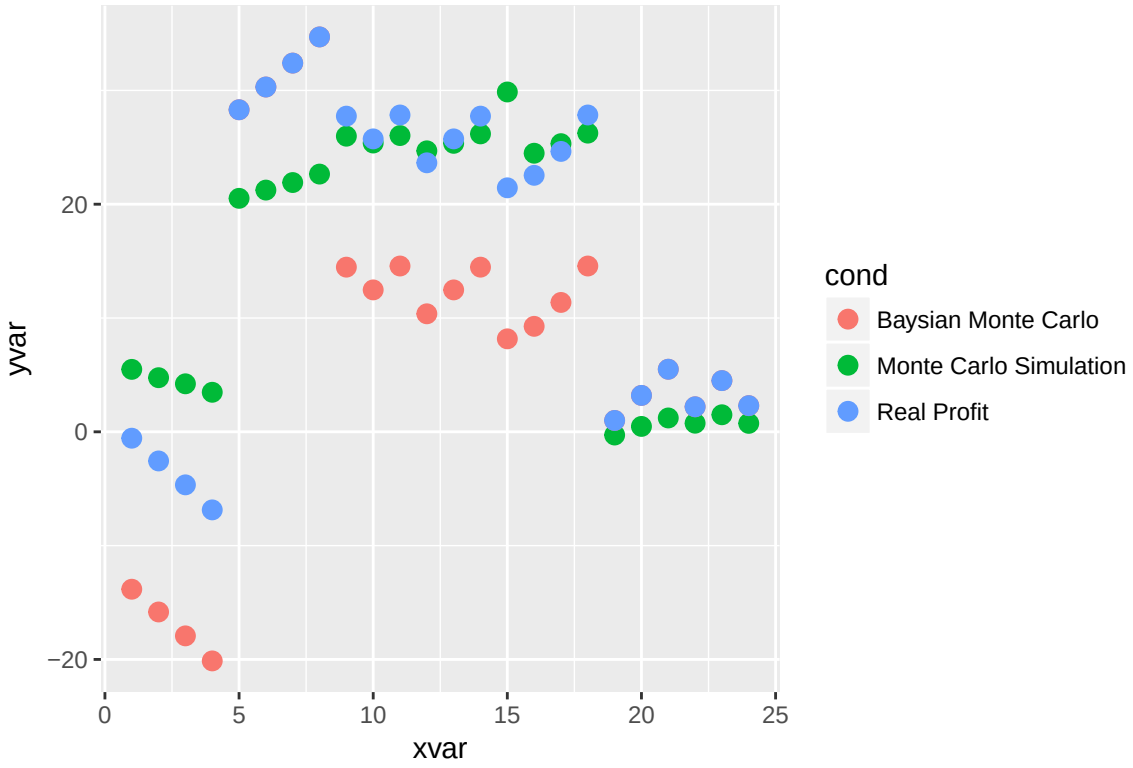
We found out that for Bayesian Monte Carlo, the four options of selling a put yielded the best profit.

2.3 Real Option Porfits and Profit Comparisons

The real S&P 500 index price on April 20th, 2018 turns out to be 2670.14. We used this price to calculate our option strategies profits. We then visualized simulated option prices with the real option price on the plot below.

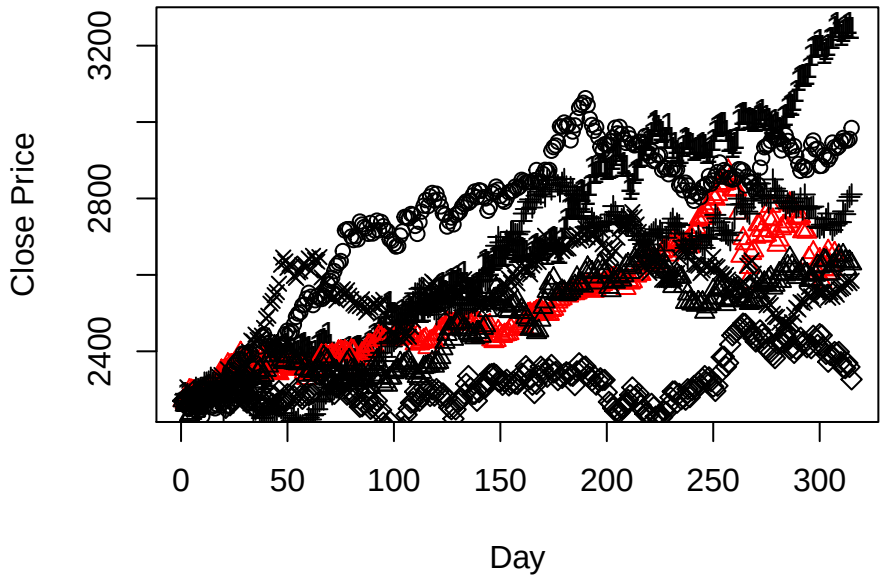
The Monte Carlo Simulation had an overall closer performance with the real profit, even though the last six dots of Bayesian Monte Carlo overlapped with the real world profit. In the future, we recommended using Monte Carlo Simulation to simulate option profit.

option1	option2	Profit Mean
buy a 2635 call		-0.56
buy a 2640 call		-2.56
buy a 2645 call		-4.66
buy a 2650 call		-6.86
sell a 2635 put		28.30
sell a 2640 put		30.30
sell a 2645 put		32.40
sell a 2650 put		34.70
buy a 2635 call	sell a 2635 put	27.74
buy a 2640 call	sell a 2635 put	25.74
buy a 2640 call	sell a 2640 put	27.84
buy a 2645 call	sell a 2635 put	23.64
buy a 2645 call	sell a 2640 put	25.74
buy a 2645 call	sell a 2645 put	27.74
buy a 2650 call	sell a 2635 put	21.44
buy a 2650 call	sell a 2640 put	22.54
buy a 2650 call	sell a 2645 put	24.64
buy a 2650 call	sell a 2650 put	27.84
buy a 2635 call	sell a 2640 call	1.00
buy a 2635 call	sell a 2645 call	3.20
buy a 2635 call	sell a 2650 call	5.50
buy a 2640 call	sell a 2640 call	2.20
buy a 2640 call	sell a 2645 call	4.50
buy a 2645 call	sell a 2650 call	2.30



2.4 Further Analysis

The next thing we wanted to do was to test our Monte Carlo simulation against known data in the short and long term to see how well it models the actual closing price. So we used the index prices from April 20th, 2016 to January 10th, 2017 (1 week before the original data set's start) to simulate the entire original data set, and see how closely the simulation matches the known values. After running 300 simulations and comparing them with the original data, the results were as you would expect:



Most of the simulations follow the original data pretty closely, but as you move further along in time, the worse the simulation matches the original data, which you would expect since the stock market is so volatile it is hard to predict what will happen like 2 months down the road. So, this simulation method appears to do a pretty good job of modeling S&P 500 index prices in short term situations.

3. Conclusion

In conclusion, we found that simulation is effective in estimating options profits and future index prices. Both methods we used did a good job of predicting options profits, but we found that the Monte Carlo simulation performed better than the Bayesian Monte Carlo Integration. Our Monte Carlo simulation predicts that the 15 th option (buy 2650, sell 2635) would be the most profitable. We further confirmed that our Monte Carlo simulation was robust and capable of handling minor inaccuracies in the parameter estimates used. We also found that using our Monte Carlo simulation to model future index prices is also pretty accurate in the short term, but not the long term. This result is unsurprising for two reasons. First, the model's performance naturally degrades as small errors accumulate over time. Secondly, stock prices are frequently impacted by external events that our model does not and can not account for.

Code Appendix

Monte Carlo Simulation

```
X_GSPC = read.csv("^GSPC.csv")
stock_price = subset(X_GSPC, select = c(1, 5))  #select Date and Close price
set.seed(0)
stock_price$dailyreturn = 0  # column initialization
for (i in 1:306) {
    # rate of return
    stock_price$dailyreturn[i + 1] = round(stock_price$Close[i + 1]/stock_price$Close[i] -
        1, 6)
}
stock_return = stock_price[-1, ]
mu = mean(stock_return$dailyreturn)
sigma = sd(stock_return$dailyreturn)
set.seed(2567)
# simulate the sp500 price on April 20th.
S = numeric(8)
S[1] = stock_price$Close[307]  #initial price
path = 300  #sample size
stock_value = numeric(path)
pred_return = numeric(path)
for (i in 1:path) {
    for (j in 1:8) {
        z = rnorm(1)  #randomly generate a number z
        S[j + 1] = S[j] * exp((mu - 1/2 * sigma^2) + sigma * z)
    }  # stock price in the next day
    stock_value[i] = S[8]  #stock price on April 20th
    pred_return[i] = round(S[8]/S[7] - 1, 6)
}  ## daily return
```

Baysian Monte Carlo Intergation

```
return_absolutevalue = abs(stock_return$dailyreturn)
fit = fitdistr(return_absolutevalue, "exponential")

link.function = function(x, mu) {
    f = c()
    for (i in 1:length(x)) {
        f[i] = dnorm(mu - sigma^2/2, sigma) * 1/2 * dexp(abs(mu),
            rate = fit$estimate, log = FALSE)
    }
    prod(f)
}
hist_data = stock_price$Close[303:307]

### Generate sequence of mu to calculate constant C in the
### posterior distribution
mu_hat = seq(quantile(stock_return$dailyreturn, 0.05), quantile(stock_return$dailyreturn,
    0.95), by = 1e-05)
theta = c()
for (i in 1:length(mu_hat)) {
    theta[i] = link.function(hist_data, mu_hat[i])
}
```

```
C = mean(theta)
expected_S = c()
for (i in 1:length(mu_hat)) {
    expected_S[i] = stock_price$Close[307] * exp(mu_hat[i] *
        8) * link.function(hist_data, mu_hat[i])/C
}
mean(expected_S)
```

Bootstrap

```
## The code does not have to go exactly here. You can move
## this stuff around
set.seed(25674)
return_indices = matrix(0, nrow = 1000, ncol = 300)
mean_est = numeric(1000)
sd_est = numeric(1000)
mean_est_jackse = numeric(1000)
sd_est_jackse = numeric(1000)
for (i in 1:1000) {
    index = sample(1:300, size = 300, replace = TRUE)
    tempsample = pred_return[index]
    mean_est[i] = mean(tempsample)
    sd_est[i] = sd(tempsample)
    return_indices[i, ] = index
}  ## This is the bootstrap
mu2 = mean(mean_est)
sigma2 = mean(sd_est)
for (i in 1:300) {
    keep = (1:1000)[apply(return_indices, MARGIN = 1, FUN = function(k) {
        !any(k == i)
    })]
    mean_est_jackse[i] = sd(mean_est[keep])
    sd_est_jackse[i] = sd(sd_est[keep])
}  ## This is the jacknife
mu2_jackse = mean(mean_est_jackse)
sigma2_jackse = mean(sd_est_jackse)

if (mu > mu2) {
    mu_pval = pnorm((mu - mu2)/mu2_jackse, lower.tail = FALSE)
} else {
    mu_pval = pnorm((mu - mu2)/mu2_jackse)
}
mu_stats = c(mu, mu2, mu2_jackse)
sigma_stats = c(sigma, sigma2, sigma2_jackse)
jackboot_results = rbind(mu_stats, sigma_stats)
colnames(jackboot_results) = c("Original", "New", "SE of New")
rownames(jackboot_results) = c("Mu", "Sigma")
```