Report for project – Peng Shiyu (A0186056Y)

Key steps:

1. Get Bitcoin – USD exchange rate from yahoo finance. Take adjusted close price 3,874.98 on March 07.

2. Use getline instruction to read csv files, and load the data into the matrixes.

3. Calculate the average of forward prices of each maturity

4. Use rootbracketing function, rootfinding function and rfbisect method to calculate the implied volatility. Take the average of bid and ask as the option price.

5. Design the adjustment of volatility which is correspondent to the put and call option with the same strike.

   First, detect which strike can be of a call option and a put option.

   Second, calculate the volatility by using the bid and ask price respectively for both call and put option.

   Third, determine the range within the volatility from bid price and the volatility from ask price of an option. If the ranges of call option and put option have no intersection, that means there is an opportunity of arbitrage. If the ranges have an intersection, take the middle point as the volatility, so that the volatility can be within both call and put option's range. And the call and put option with the same strike can have the same volatility.

   Define a function as follow:

```
double voladj(string Type, double str, double t, double ave, double bid1, double bid2, double ask1, double ask2){
    double vb1=0, va1=0, vb2=0, va2=0, max1=0, min1=0, max2=0, min2=0;
    double vol=0;
    if(Type=="call"){vb1 = rootfinding1(str, t, ave, bid1);
                     va1 = rootfinding1(str, t, ave, ask1);
                     vb2 = rootfinding2(str, t, ave, bid2);
                     va2 = rootfinding2(str, t, ave, ask2);}
                     else{
                     vb1 = rootfinding2(str, t, ave, bid1);
                     va1 = rootfinding2(str, t, ave, ask1);
                     vb2 = rootfinding1(str, t, ave, bid2);
                     va2 = rootfinding1(str, t, ave, ask2);}
                     if(vb1>=va1){max1=vb1;min1=va1;}else{max1=va1;min1=vb1;}
                     if(vb2>=va2){max2=vb2;min2=va2;}else{max2=va2;min2=vb2;}
                     if((min2>=min1)&&(max2<=max1)){vol = (max2+min2)/2;}
                     else if((min2<=min1)&&(max2>=max1)){vol = (max1+min1)/2;}
                     else if((min2<=min1)&&(max2>=min1)&&(max2<=max1)){vol = (min1+max2)/2;}
                     else if((min2>=min1)&&(min2<=max1)&&(max2>=max1)){vol = (min2+max1)/2;}
                     else{std::cout << "There is a call-put arbitage when strike is " << str << ", and maturity " << t <<std::endl;}
    return vol;

}
```

There are 3 arbitrage opportunities when the expiry date is 2019/06/28.

```
There is a call-put arbitage when strike is 1500, and maturity 0.309589
There is a call-put arbitage when strike is 9000, and maturity 0.309589
There is a call-put arbitage when strike is 10000, and maturity 0.309589
```

6. Build volatility smiles. Read the data of strikes and correspondent volatilities in to smile class. Avoid to read the same strike-vol pair duplicately. Construct pillarsmiles.

7. Get the maturity date from the input. Using date class operator "-" to transform the maturity into a number that can be calculated by the program.

8. Do the interpolation and extrapolation for the forward price. Get the input of maturity, if the maturity is off the range of the input option expiry dates, set the forward price equal to the nearest endpoint's forward price. If the maturity is within the range, use the nearest two time points' forward prices to do the linear interpolation. If the maturity is exactly an input date, just set the forward price equal to that date's forward price.

```
double s;
int i;
for (i=0; i < pillarSmiles.size(); i++ )
if (t < pillarSmiles[i].first ) break; // i stores the index of the right end of the bracket
if (i == 0){s = price1;}
if (i == pillarSmiles.size()){s = price6;}
double t1 = pillarSmiles[i-1].first;
double t2 = pillarSmiles[i].first;
double price[6]; price[1]=price1; price[2]=price2; price[3]=price3; price[4]=price4; price[5]=price5; price[6]=price6;
s = ((t2 - t) / (t2 - t1))*price[i-1] + (1-((t2 - t) / (t2 - t1)))*price[i];
```

9. Using finite difference method to calculate the delta, vega and theta, so that they

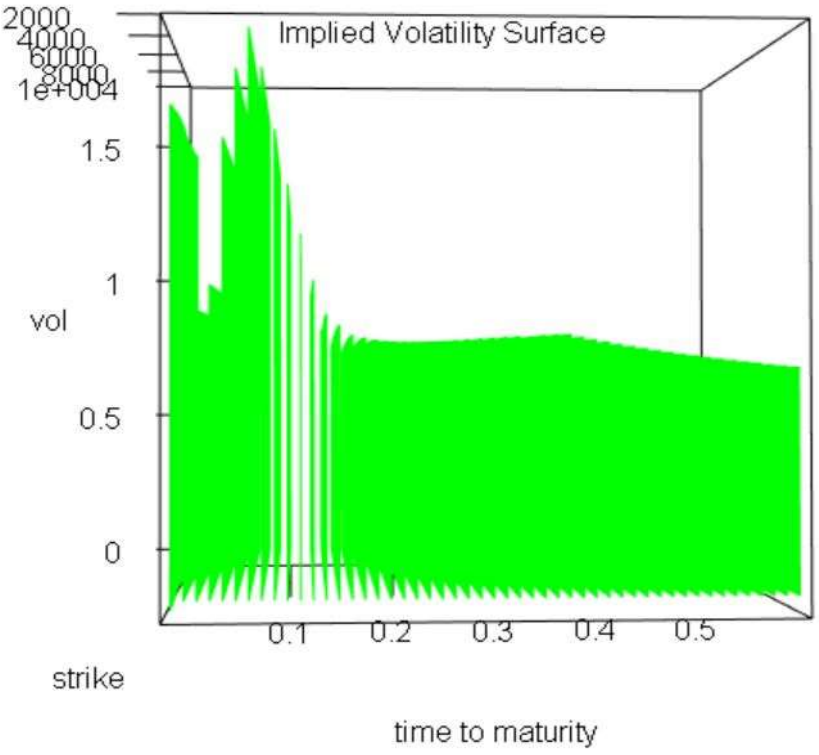
can be more precise. Take the difference as 0.001 in spot price, volatility and time, all use central finite difference. Then calculate the derivative.

```
(bsPricer(Call, k, t, s+0.001, iv.Vol(t, k), 0.0)-bsPricer(Call, k, t, s-0.001, iv.Vol(t, k), 0.0))/exchange_rate/0.002
(bsPricer(Call, k, t, s, iv.Vol(t, k)+0.001, 0.0)-bsPricer(Call, k, t, s, iv.Vol(t, k)-0.001, 0.0))/0.002
(bsPricer(Call, k, t+0.001, s, iv.Vol(t+0.001, k), 0.0)-bsPricer(Call, k, t-0.001, s, iv.Vol(t-0.001, k), 0.0))/0.002
```

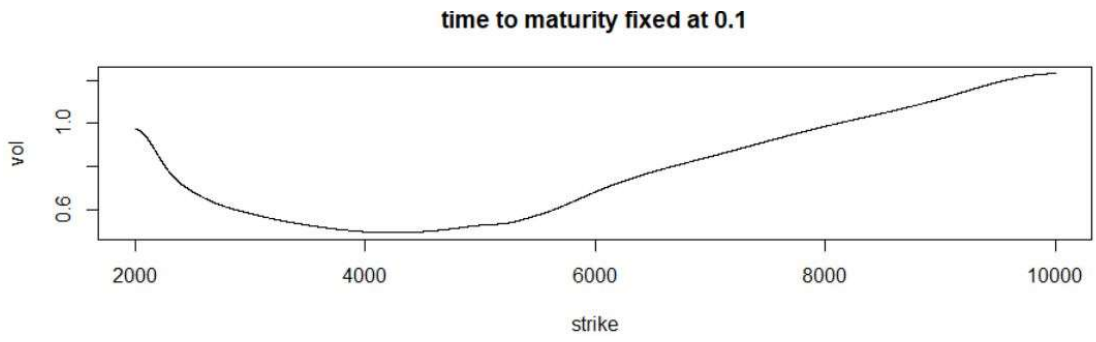
Test and Result

Build a volatility surface of half year from 2019 March to 2019 September. Get data from the program, the time interval is 0.01 from 0 to 0.5, strike interval is 1 from 2000 to 10000. Save the result in the file “ImpliedVol--Peng Shiyu.txt” (Code in the main() function).

Plot the result:

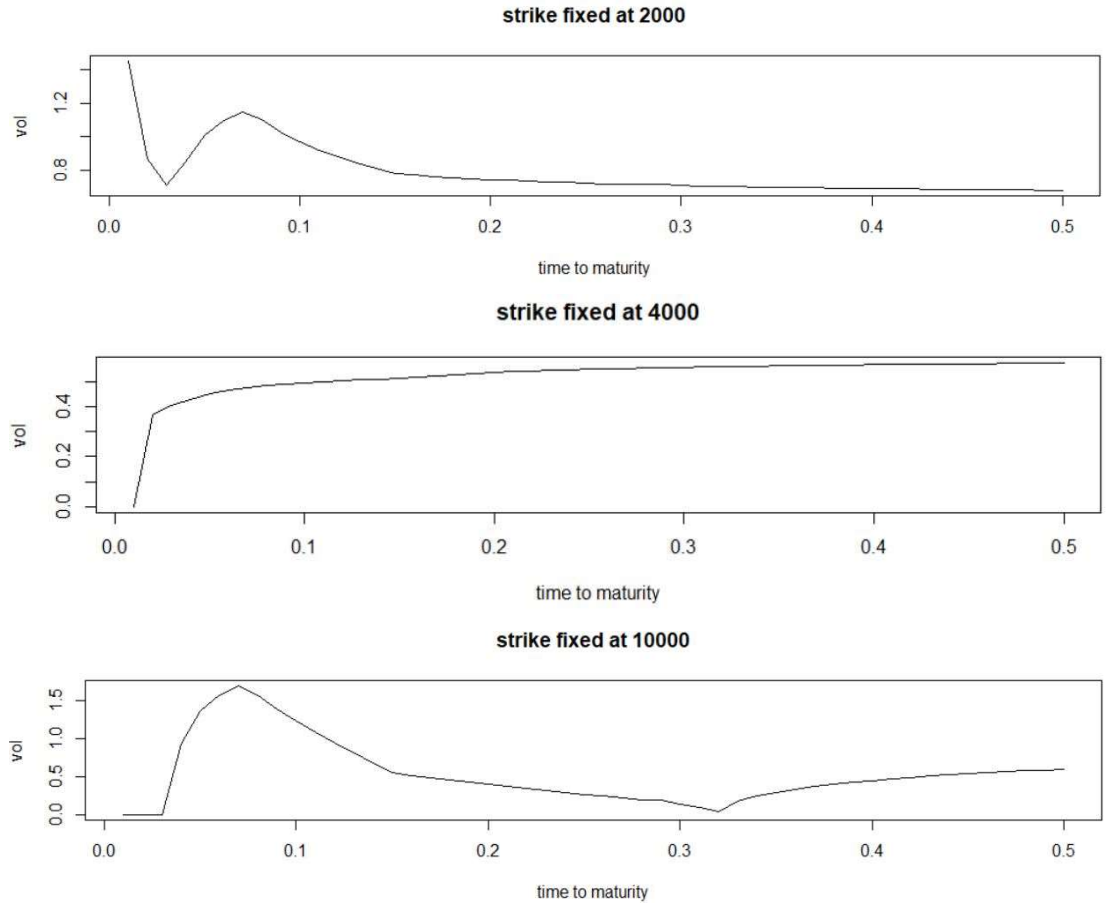


If we fix the time to maturity as 0.1, we will get a volatility smile:



If we fix the strike as 2000, 4000 and 10000, respectively we get 3 different-shaped

plots:



When execute the file "pro.out", enter the maturity as 2019-06-01 and strike as 5000, we get the mid-price, ask and bid price (calculated by volatility $\pm$ 0.5%), delta, vega and theta for both call and put option.

```
Enter maturity (Year) (format: 2019): 2019
Enter maturity (Month) (format: 4): 06
Enter maturity (Day) (format: 19): 01
Enter strike (USD): 5000
The mid price for call is (BTC)0.106286
The ask price for call is (BTC)0.107239
The bid price for call is (BTC)0.105334
The delta for call is (BTC)0.000143593
The vega for call is (USD)737.971
The theta for call is (USD)1006.9
The mid price for put is (BTC)0.103801
The ask price for put is (BTC)0.104753
The bid price for put is (BTC)0.102849
The delta for Put is (BTC)-0.000114472
The vega for Put is (USD)737.971
The theta for Put is (USD)1006.9
```