Analysis of implied volatility

Stochastic Finance

Adriano Salcuni

Contents

Introduction

This document aims to provide a theoretical and practical explanation of implied volatility. The draft is divided into three parts:

- a first part which will provide theoretical outlines on implied volatility and an implementation of the algorithm based on the Newton-Raphson method;
- a second part in which the price of the options will be calculated using the Monte Carlo simulation and compared with the price obtained using the Black and Scholes formula. Furthermore, the respective implied volatilities will be calculated and compared;
- a third party where the data of the Twitter Inc. company will be downloaded and the implied distribution will be calculated and compared with the log-normal distribution

The packages that will be used in the document are the following:

```
library("derivmkts")
library("tidyverse")
library("quantmod")
library("readr")
```

Theoretical framework

Black and Scholes's formula

The Black and Scholes model allows you to price a European option. The assumptions underlying the model are the following:

- the rate of return on the risk-free asset is constant;
- the stock does not pay dividends
- there are no arbitrage opportunities
- titles are infinitely divisible;
- short selling is allowed;
- agents are profit maximizers and price-takers;
- the market is open continuously.
- there are no transaction costs and taxes;
- the share price follows a geometric Brownian motion with constant μ and σ;

Its expression in the form of a differential equation is as follows:

$$\frac{\partial V}{\partial t} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

Its solution for a call is given by:

$$C(t) = S(t)N(d1) - Ke^{-r\tau}N(d2)$$

with:

$$d1 = \frac{log[S(t)/K] + (r + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

and:

$$d2 = d1 - \sigma\sqrt{\tau}$$

Greek

The greeks measure the sensitivity of the option value to the variation of one of the parameters. The following are the Greeks with respect to the call:

- Delta: sensitivity of the option value with respect to the underlying

$$\Delta = \frac{\partial C}{\partial S} > 0$$

- Theta: sensitivity of the option value with respect to time

$$\Theta = \frac{\partial C}{\partial t} < 0$$

- Rho: sensitivity of the option value with respect to the interest rate

$$\rho = \frac{\partial C}{\partial r} > 0$$

- Vega: sensitivity of the option value with respect to volatility

$$V = \frac{\partial C}{\partial \sigma} > 0$$

- Gamma: is the rate of change of the option with respect to the underlying

$$\Gamma = \frac{\partial^2 C}{\partial S^2} > 0$$

Vega

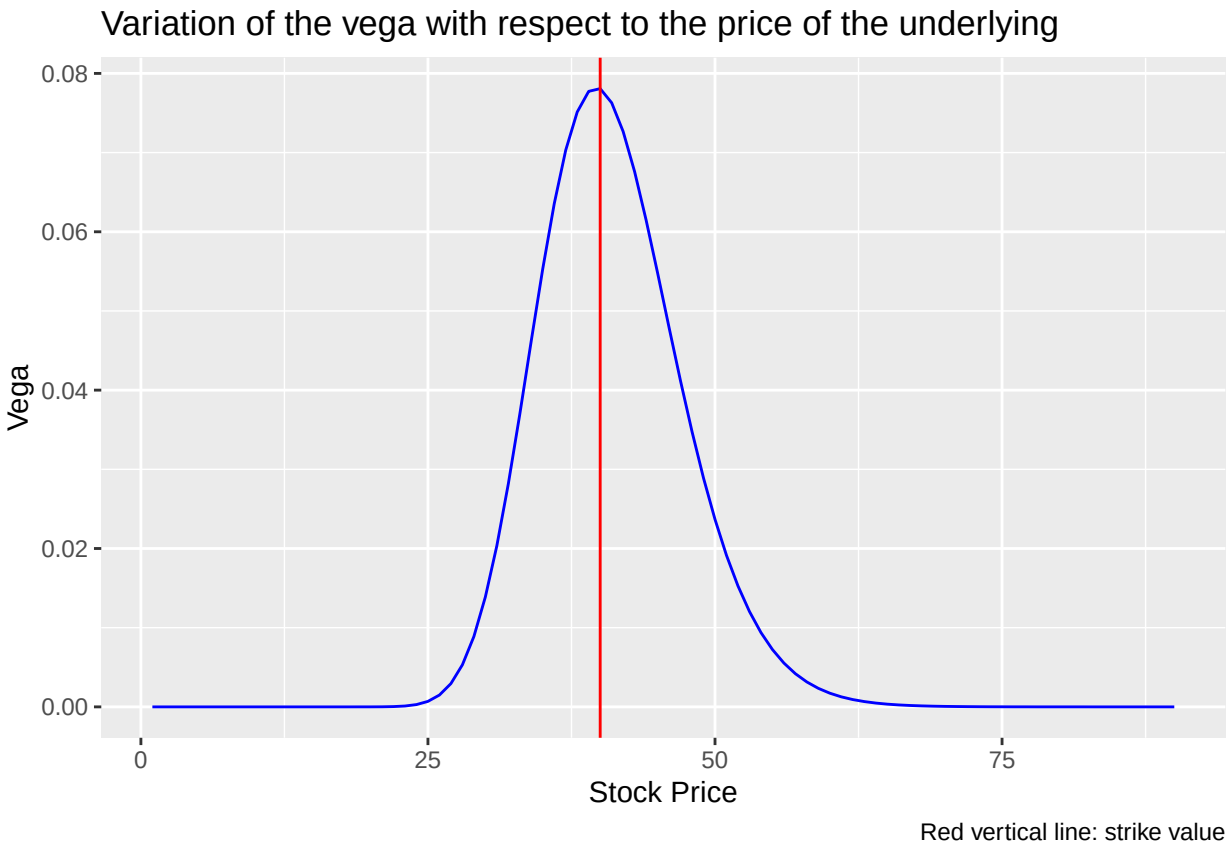
One of the assumptions made previously is the constancy of the volatility of the asset. In reality, however, voltatility changes with time. The value of a derivative is subject to change due to changes in volatility. The vega of a vanilla option is given by the following formula:

$$V = \frac{\partial \Pi}{\partial \sigma} = S\sqrt{\tau}N'(d1)$$

If the Vega is a very high positive or negative number, it means that the option price is highly sensitive to the volatility of the underlying asset.

Below is an example of the change in vega versus change in the value of the underlying.

```
k=40; v=0.30; r=0.08; tt=0.25; d=0

vega <- vector("numeric", length=90)
for (i in 1:90){
  vega[i] <- greeks(bscall(s=i, k, v, r, tt, d), complete=FALSE, long=FALSE, initcaps=TRUE)[4]
}

new_tbl_vega<- tibble::tibble(STOCK_PRICE = 1:90, VEGA = vega)
ggplot(data=new_tbl_vega, aes(y = VEGA, x = STOCK_PRICE))+
  geom_line(col="blue")+
  geom_vline(xintercept = k, col="red")+
  labs(x = "Stock Price", y = "Vega", caption = "Red vertical line: strike value")+
  ggtitle("Variation of the vega with respect to the price of the underlying ")
```



From the figure it can be seen that the vega is maximum when the option price is at the money. In fact, an increase or decrease in volatility can significantly change the value of an option if it is at-the-money.

Calculating the vega from the Black and Scholes model may seem strange because one of the assumptions is the constancy of the variance. Theoretically, in fact, it would be more correct to compute vega from a model in which variability is assumed to be stochastic. However, it turns out that the vega calculated from a stochastic volatility model is very similar to the vega calculated from the B&S model. So in practice calculating vega from a model where volatility is constant works reasonably well.

Implied Volatility

The implied volatility represents the volatility of the underlying which, inserted in the Black and Scholes forum, provides a theoretical value equal to the market price of the option. It could be said that it provides a view of the volatility that the market has for the period still remaining to the expiry of the option.

Since there is no closed formula to derive the implied volatility as a function of the other arguments and the option, it is necessary to solve the equation:

$$V_{BS}(S_0,\ t_0,\ \sigma,\ r,\ E,\ T) = known\ value$$

for σ, where V_{BS} is the closed Black and Scholes formula. The current price of the underlying is S_0, the valuation date is t_0 and everything is known except σ. The Newton-Raphson method can be used to solve this equation.

Newton-Raphson method

The iterative process is as follows:

1. Calculate the theoretical price of the option with the BS formula by inserting an arbitrarily chosen σ;
2. The price obtained is compared with the market price;
3. If the result is lower (higher) the volatility increases (decreases);
4. The new volatility datum is entered until the theoretical value approaches the market price.

The forumula that allows us to calculate the new σ is the following:

$$\sigma_{new} = \sigma_{old} - \frac{BS(\sigma_{old}) - C_m}{C'(\sigma_{old})} = \sigma_{old} - \frac{BS(\sigma_{old}) - C_m}{Vega}$$

where $BS(\sigma_{old})$ is the call price calculated with the B&S formula; C_m is the market price of the call and $C'(\sigma_{old})$ is the $Vega$.

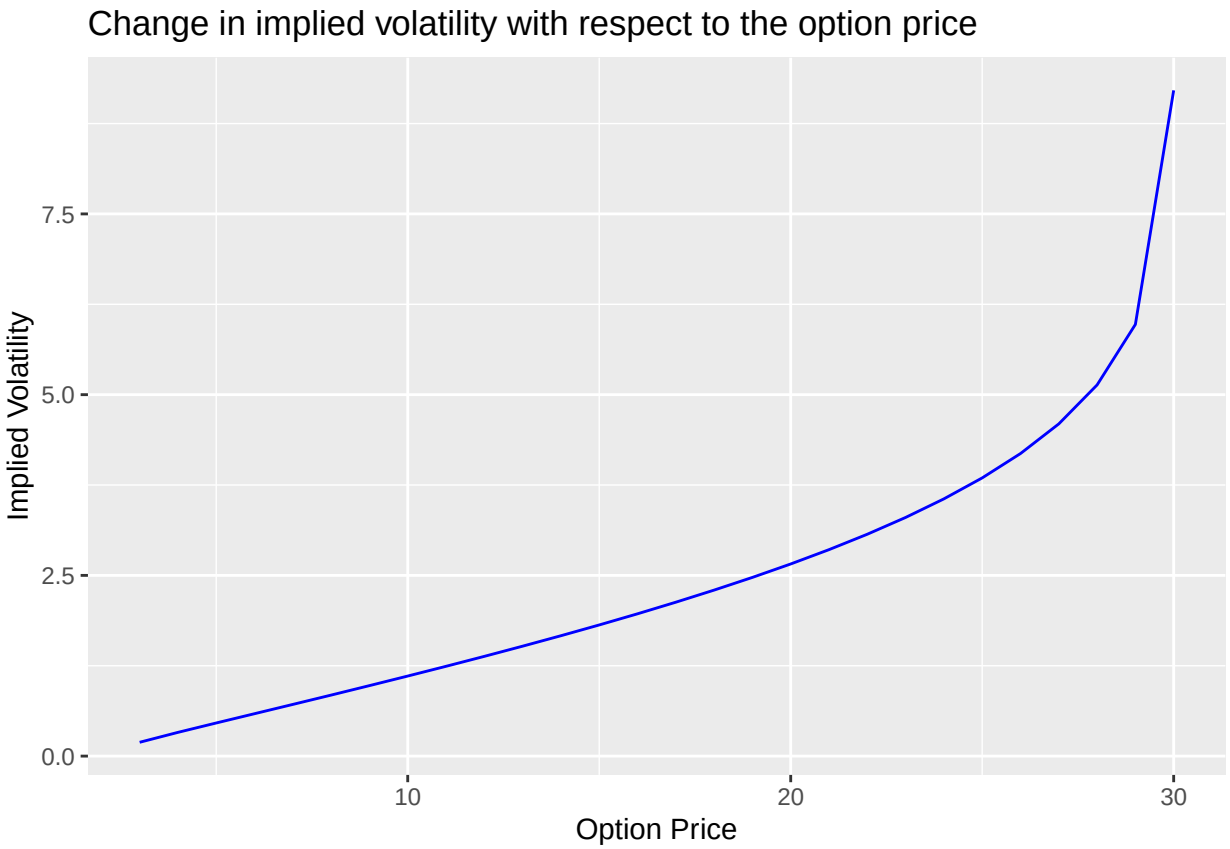
An algorithm for calculating the implied volatility for a call is implemented below:

```
vol_old=0.3

implied_vol <- function (S0, K, tt, r, market_price, vol_old=0.3){


      "
      Implementation of the function for calculating the implied volatility of       a European optio

      S0: price of the underlying
      K: exercise price
      tt: expiry time
      r: risk-free return
      market_price: option price
      vol_old= variance initialization
      "

  tol=0.0001 #initialization of the tolerance used to define a threshold
  max_iter <- 200 #maximum number of iterations of the algorithm

  for (k in 1:max_iter){

    #calculation of the price of the call assuming that the variance is vol_old
    bs_price <- bscall(s=S0, k=K, v=vol_old, r, tt, d=0)

     #calculation of vega
     # inside the "derivmkts" package the vega is calculated as follows:
     # Vega <- .FirstDer(funcname, 'v', x)/100
     # so to get the derivative just multiply by 100
     # extrapolation of the fourth position, [4], to get only the vega among all the greeks
    c_primo <- greeks(bscall(s=S0, k=K, v=vol_old, r, tt, d=0))[4]*100

    #calculation of the new volatility
    vol_new <- vol_old - (bs_price - market_price)/c_primo
```

```
    #calculate new price
    new_bs_price <- bscall(s=S0, k=K, v=vol_new, r, tt, d=0)

    #calculation of the difference in absolute value of volatility and price
    diff_vol <- abs(vol_old - vol_new)
    diff_price <- abs(new_bs_price-market_price)

    #condition to exit the for
    if (diff_vol < tol) {
      break
    }

    if (diff_price < tol){
      break
    }

    vol_old <- vol_new

  }
  return(vol_new)

}
```

Instead, below is a graph of the variation of the implied volatility with respect to the option using the implemented algorithm.

```
S0=30; K=28; r=0.025; tt=0.5; d=0

new_vol <-vector("numeric", length=28)
market_price<-3:30
imp_vol <- for (i in 1:length(market_price)){
  new_vol[i] <- implied_vol(S0, K, tt, r, market_price[i])

}

new_tbl<- tibble::tibble(PRICE_OPTION = market_price, IMPLIED_VOLATILITY = new_vol)


ggplot(new_tbl, aes(x=PRICE_OPTION, y=IMPLIED_VOLATILITY))+
  geom_line(col="blue")+
  labs(x = "Option Price", y = "Implied Volatility")+
  ggtitle("Change in implied volatility with respect to the option price")
```

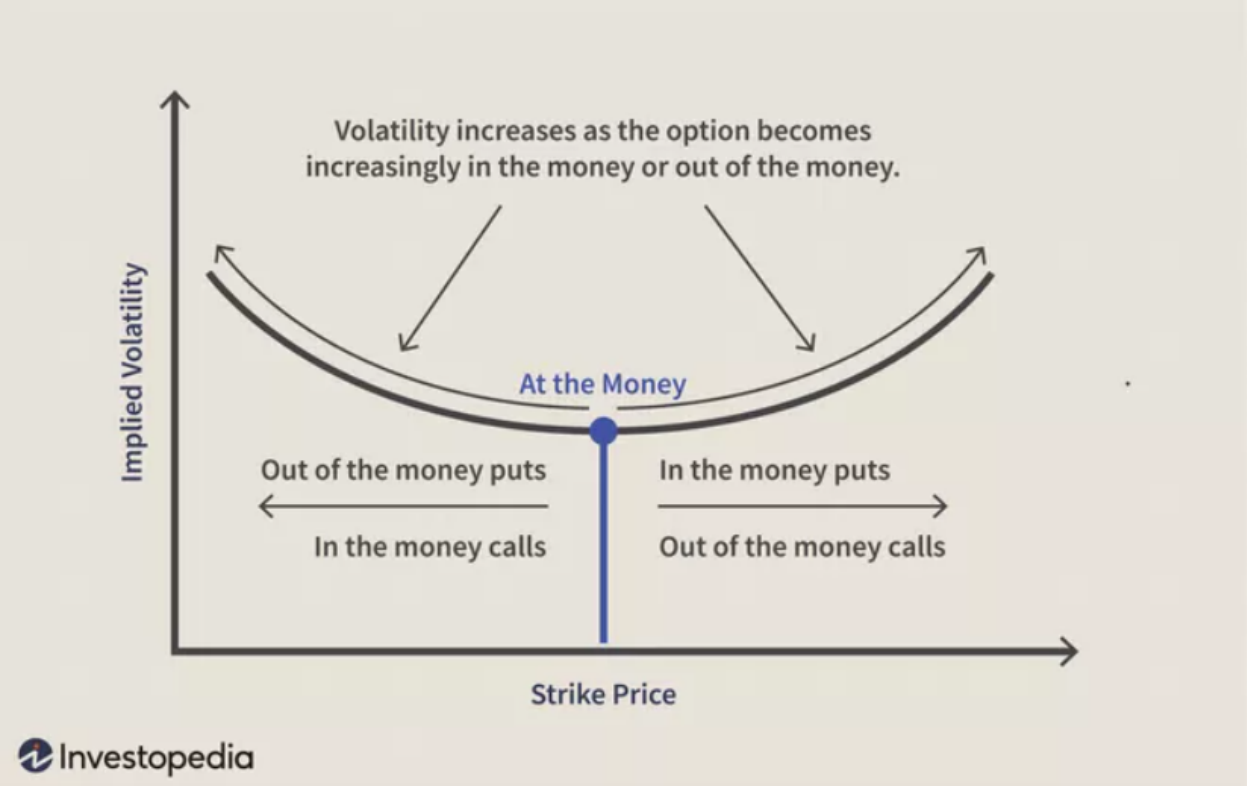


From the graph it can be seen, as one might expect, that the implied volatility increases as the option price increases.

Volatility Smile

The volatility smile is the graph of the implied volatility of the option with the same duration and priced on the same underlying as a function of its strike.

It is called this because the implied volatility increases progressively the further it is from being "at the money", therefore the implied volatility increases the more it is "out-the-money" or "in-the-money", as it is shown in the figure.



From the put-call parity equation we know that:

$$p + S_0 = c + Ke^{-rT}$$

and therefore also for the options priced with the Black-Scholes model:

$$p_{bs} + S_0 = c_{bs} + Ke{-}rT$$

and that in the absence of arbitrage the same equation is also valid for market prices:

$$p_{mkt} + S_0 = c_{mkt} + Ke{-}rT$$

That is, in the absence of arbitrage, the options priced with the Black-Scholes model coincide with market prices.

Subtracting these last two equations we get:

$$p_{bs} - p_{mkt} = c_{bs} - c_{mkt}$$

When we price options with the Black-Scholes model using implied volatility the spread is zero and the two quantities must match. This argument shows that the implied volatility of a European call option is always the same as the implied volatility of a European put option when the two have the same strike price and expiration date.

The volatility smile is not predicted by the Black-Scholes model, as it predicts that the implied volatility curve is flat when plotted against different strike prices. Yet, in the real world this is not the case. This "anomaly" allows us to calculate what is called the "implicit distribution" which we will see later.

Simulated data

An analysis of the implied volatility through the Monte Carlo simulation is proposed below. The importance of this analysis lies in understanding the advantages, but above all the limits of implied volatility.

Assuming that the price process is a geometric Brawnian motion described by the following stochastic equation:

$$dS(t) = \mu S(t)dt + \sigma S(t)dZ(t)$$

Which has as a solution:

$$S_t = S_0 e^{(\mu - \sigma^2/2)t + \sigma dW_t}$$

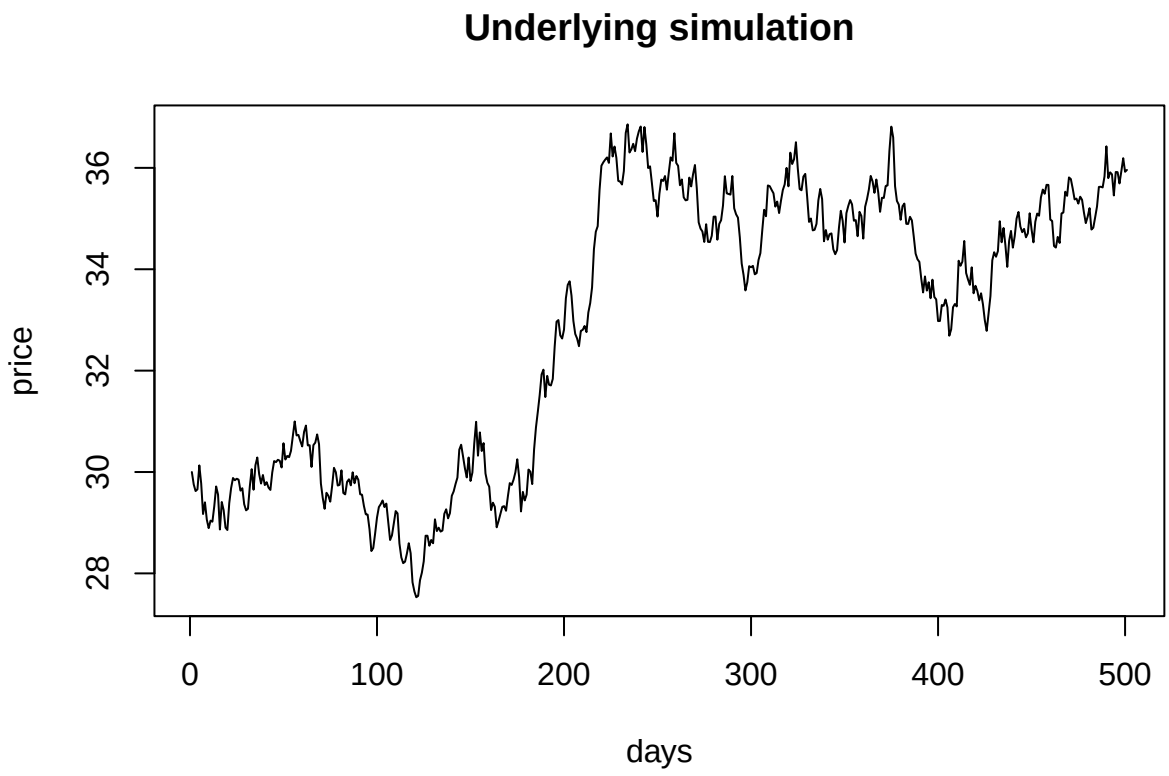
And by dividing the interval [0, T] into n subintervals of equal size $\Delta t = T/n$, we obtain:

$$S_t = S_{t-1} e^{(\mu - \sigma^2/2)\Delta t + \sigma\sqrt{\Delta t}\epsilon_t}, \quad \epsilon \sim N(0,1)$$

We simulate the trajectory for the price evolution assuming:

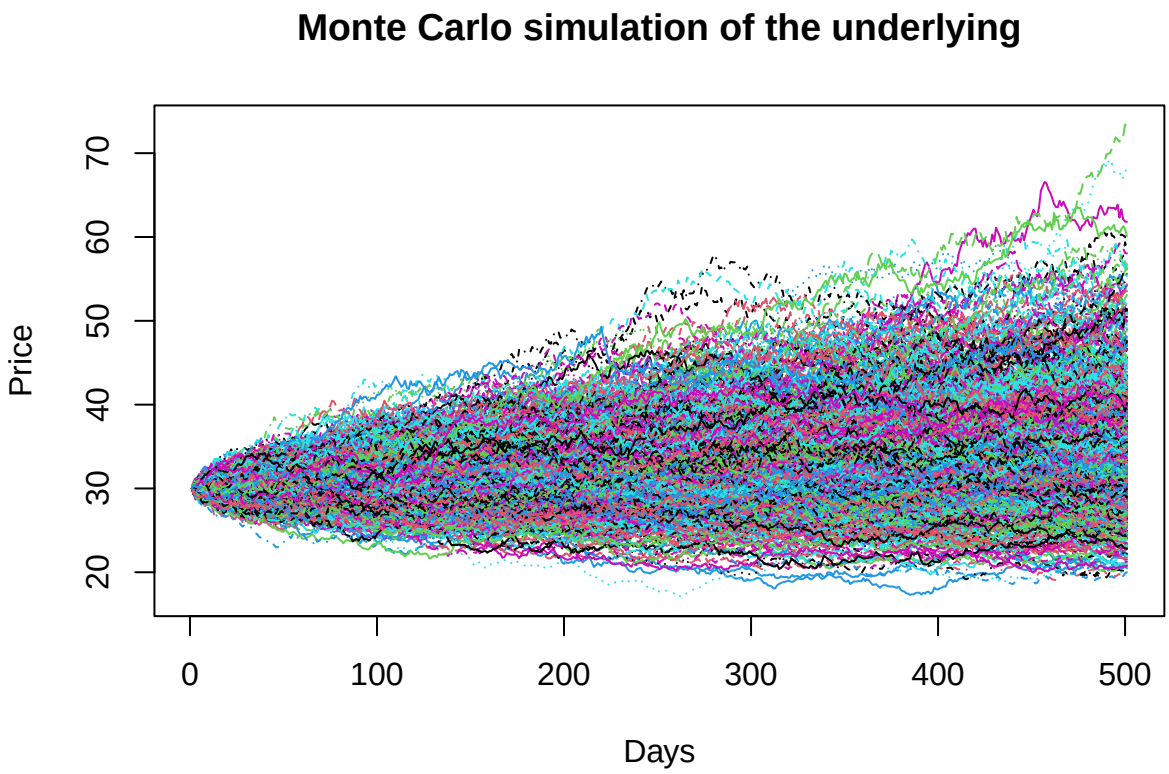
- $S_0 = 30$
- $\mu = 0.08$
- $\sigma = 0.15$
- $T = 2$
- $r = 0.025$

```
#initialization values
S0=30
mu=0.08
sigma=0.15
T=2
n=500 #the trading days are 250 per year
S=vector("numeric", length=501)

S[1]=S0

for(i in 1:(length(S)-1)){

    S[i+1]=S[i]*exp((mu-sigma^2/2)*T/n+sigma*sqrt(T/n)*rnorm(1,0,1))}

plot(S, xlab="days", ylab="price",type="l", main="Underlying simulation")
```



Let's create a matrix of simulations. Precisely we simulate the underlying path 1000 times.

```
Nsim = 1000

S = matrix(0, ncol=Nsim, nrow=501)

S[1,] = S0

for(j in 1:Nsim){
    for(i in 1:n){

        S[i+1,j]=S[i,j]*exp((mu-sigma^2/2)*T/n+sigma*sqrt(T/n)*rnorm(1,0,1))

  }
}

matplot(S, xlab="Days", ylab="Price",type="l", main = "Monte Carlo simulation of the underlying")
```



Finally, we calculate the value of the call both as the arithmetic mean of the simulated payoffs $Max(S_t - K, 0)$ and through the Black and Scholes formula.

```
K = 30
r = 0.03
payoff = ifelse(S[n+1,]-K>0,S[n+1,]-K,0)
Ct = mean(payoff)

C0_sim=Ct*exp(-r*T)
imp_vol_sim <- implied_vol(S0, K, tt=T, r, market_price=C0_sim)

C0_bs <- bscall(S0,K,sigma,r,T,d=0)
imp_vol_bs <- implied_vol(S0, K, tt=T, r, market_price=C0_bs)
```

	Simulation	Black and Scholes
Call Price	5.4886089	3.4312047
Implied volatility	0.2795472	0.1499585

From the table we note a difference between the implied volatility calculated using the call price obtained from the simulated pay-offs and the implied volatility calculated using the call price using the Black and Scholes formula. This is an important lesson in understanding and applying implied volatility: it is calculated assuming that the call price is calculated using the Black and Scholes formula. In fact, from the previous calculations this emerges clearly. Only when we use the call priced with the B&S forumula are we able to precisely predict the volatility, which in the example is assigned equal to 0.15.

Real Data

Now let's analyze real data. In particular, from yahoo-finance we download the prices and options of the company Twitter, Inc. By combining the information on prices and options we can calculate the implied distribution as will be seen below.

Below is a history of the asset's prices.

```
getSymbols("TWTR",src="yahoo",return.class='zoo')
```

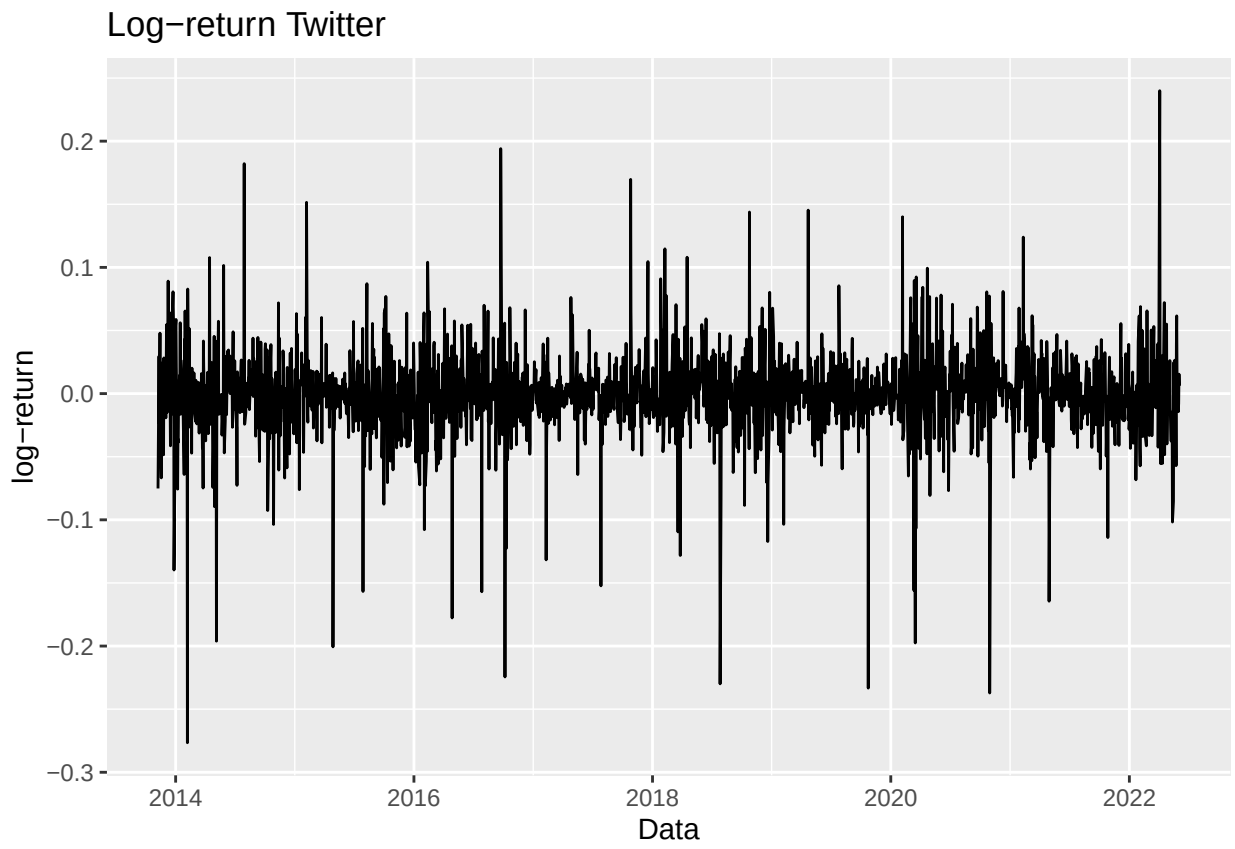
```
## Warning: TWTR contains missing values. Some functions will not work if objects
## contain missing values in the middle of the series. Consider using na.omit(),
## na.approx(), na.fill(), etc to remove or replace them.
```

```
## [1] "TWTR"
```

```
prezzo_with_na <- TWTR[,6]
prezzo_with_na <- window(prezzo_with_na, end= "2022-06-03")
prezzo <- na.omit(prezzo_with_na)

autoplot(prezzo) +
  labs(x = "Data", y = "Price")+
  ggtitle("Twitter price history")
```

Let's calculate the log returns.

```r
d_ln_Stock=diff(log(prezzo))
autoplot(d_ln_Stock)+
  labs(x = "Data", y = "log-return")+
  ggtitle("Log-return Twitter")
```




```r
#we value mu and sigma
dt=1/250
mu=mean(d_ln_Stock)/dt
sigma=sd(d_ln_Stock)/sqrt(dt)
S0=prezzo[length(prezzo)] #-1 as we want the value of Friday 03/06
```

Once we have obtained the estimates of the mean and the standard deviation we proceed with the study of the implied distribution and we will compare it with the log-normal distribution.

Implicit distribution

Knowing that the price of a call option with a strike price K and an expiry time T is given by:

$$c = e^{-rT} \int_{S_T=K}^{\infty} (S_t - K) g(S_T) dS_t$$

Differentiating with respect to K we get:

$$\frac{\partial c}{\partial K} = -e^{-rT} \int_{S_T=K}^{\infty} g(S_T) dS_t$$

Differentiating again with respect to K we get:

$$\frac{\partial^2 c}{\partial K^2} = e^{-rT} g(K)$$

This shows that the probability density function is given by:

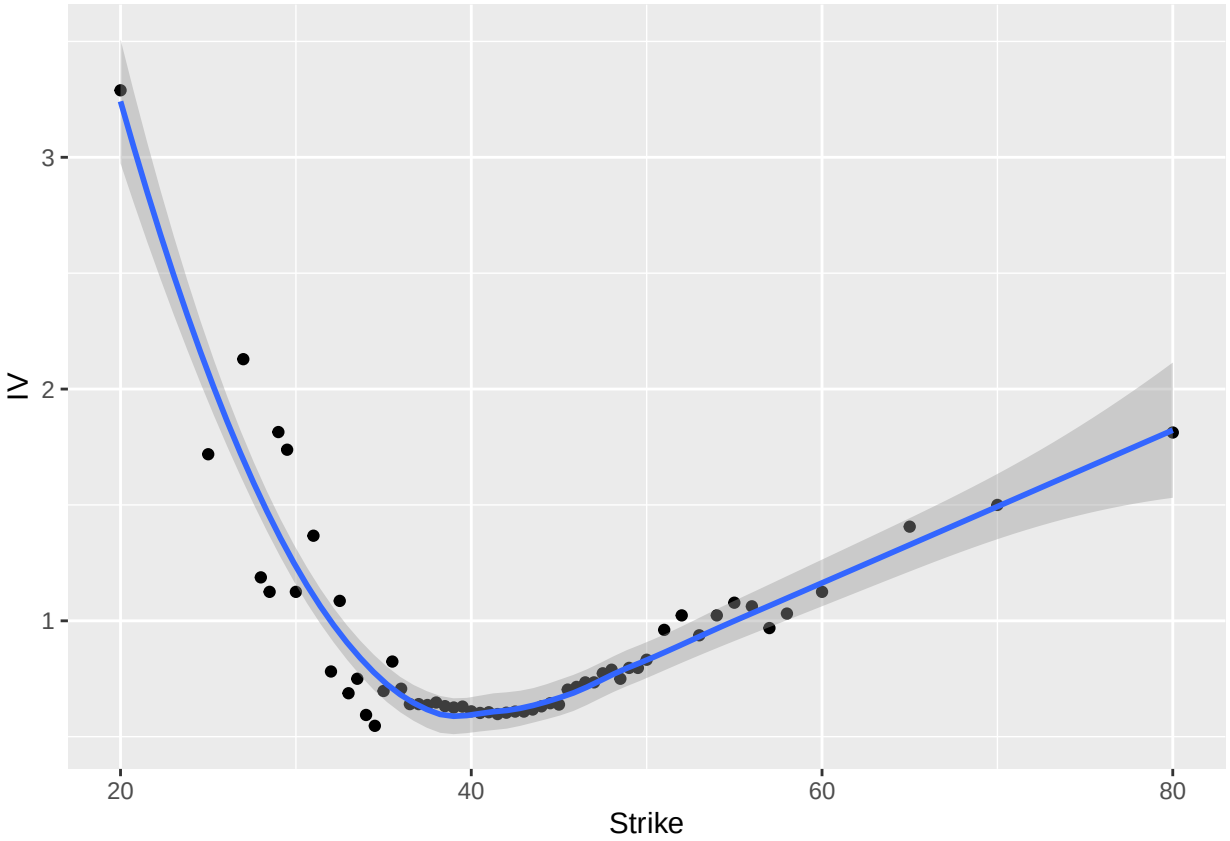
$$g(K) = e^{-rT} \frac{\partial^2 c}{\partial K^2}$$

This result was proved by Breeden and Litzenberger (1978) and allows us to estimate risk-neutral probability distributions from the volatility smile. We also assume that c_1, c_2 and c_3 are the prices of European call options with maturity T with strike price $K - \delta$, K and $K + \delta$ respectively. Assuming that δ is small we can obtain an estimate of $g(k)$ by approximating the second partial derivative with the following equation:

$$e^{rT} \frac{c_1 + c_3 - 2c_2}{\delta^2}$$

Below we implement an algorithm for calculating the implied distribution. First of all let's download the twitter options dataset from yahoo finance. The analysis was carried out with the data of 06/05/2022 with a deadline of 06/10/2022.

```
#option_twtr <- getOptionChain("TWTR",  src="yahoo")

call_twtr <- read_csv("call.csv")

ggplot(call_twtr, aes(x=Strike, y=IV))+ geom_point() + geom_smooth(method = "loess", show_col_types = F
```



From the graph we note that the implied volatility with respect to the strike behaves according to the volatility smile theory.

Next, we interpolate the implied volatility values against the strike with a local regression. The financial literature has also developed more sophisticated methods for interpolating volatility smiles, such as the SABR method, for example. In this document, however, to have a better understanding of the results and the methods used, we will estimate the volatility smiles with local regression.

```
#Data interpolation
predict_lo <- loess(IV ~ Strike, call_twtr)
```

After that, we forecast the new implied volatilities against simulated strike prices.

```
tt <- 5/250 #5 days left
r=0.025

#forecast of implied volatilities
new_strike <- seq(20,80, by=0.5)
vol_imp_pred <- predict(predict_lo, new_strike,type="response")
```

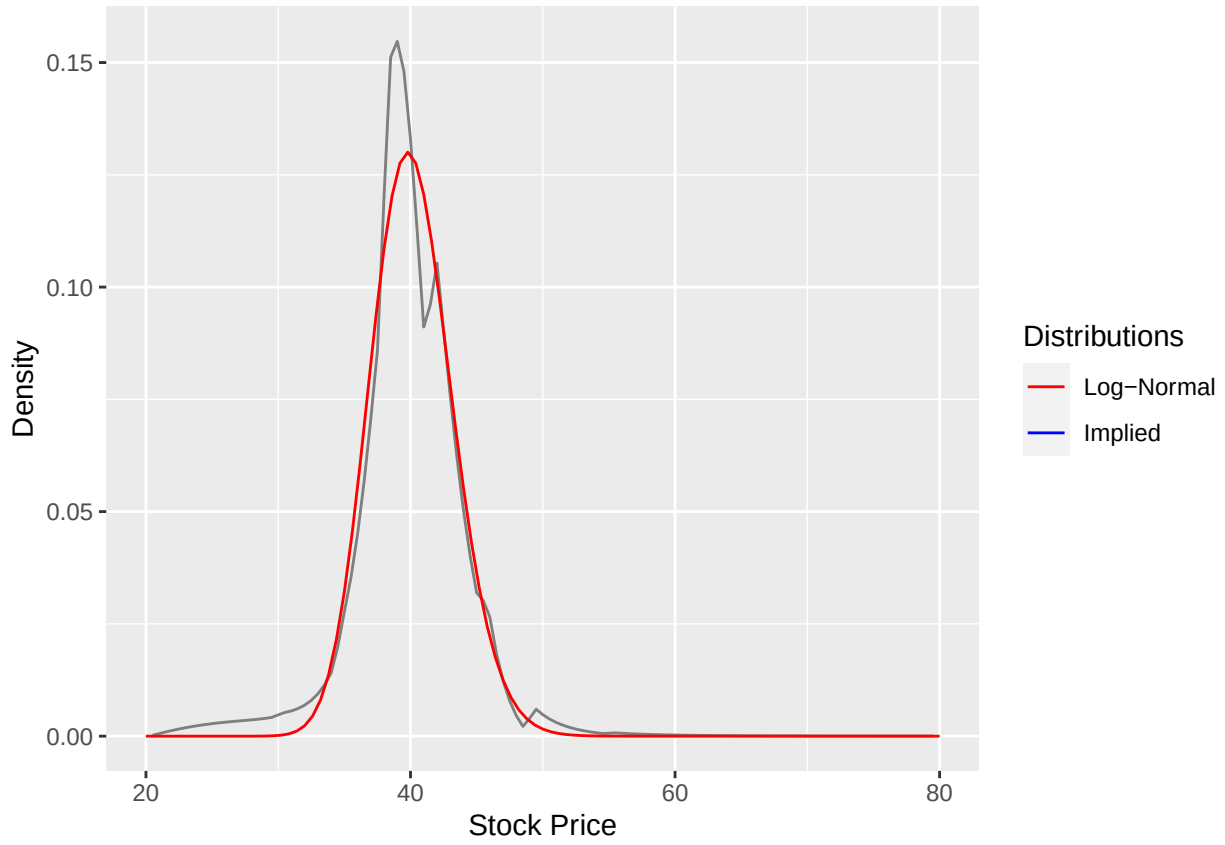
Now let's build the probability distribution function:

```
#let's initialize a vector for calculating call prices
call_new <- vector("numeric", length = 121)

#calculation of new call prices
```

```
for (i in 1:length(new_strike)){

  call_new[i]<- bscall(s=40.16, k=new_strike[i], v=vol_imp_pred[i], r=0.025, tt, d=0)

}

#calculate the implied distribution
g_k <- vector() # implicit distribution initialization
for (i in 2:(length(new_strike))){

  g_k[i] <- (exp(r*tt)*(call_new[i-1] + call_new[i+1] - 2*call_new[i]) ) / (0.5^2)

}
```

We graph the implied distribution and the log-normal

```
df_implicita <- tibble(Density = g_k, STRIKE = new_strike)




#calculation of the mean and standard deviation of the log-normal
mean_log <- log(S0) + (mu - (sigma^2)/2) *tt
sigma_log <- sigma * sqrt(tt)


ggplot(data = df_implicita, aes(x=STRIKE, y=Density, color="Implicita"))+ geom_line()+
  stat_function(fun="dlnorm", args = list(meanlog =  mean_log, sdlog = sigma_log), N=100, color="red") +
  scale_color_manual(name = "Distributions",
                         breaks = c("Log-Normal", "Implied"),
                         values = c("Log-Normal" = "red", "Implied" = "blue"))+
  labs(x = "Stock Price")
```



Black, Scholes and Merton assume in their model that the underlying is log-normally distributed. This analysis shows that this assumption, at least if we consider short-term options, is false. We note from the graph that the implied distribution of the underlying has heavier tails than those predicted by the log-normal distribution.

You might then decide to adopt the following strategy: buy what author Hull calls "deep-out-of-the-money" calls or puts. These options are inexpensive, and many of them will close "in-the-money" than the log-normal model predicts. On average the present value of the earnings will be greater than the cost of the options.

Conclusions

With this document, starting from the theoretical concepts of implied volatility, we have provided an algorithm based on the Newton-Raphson method for calculating it. We then obtained the option price using Monte Carlo simulation and the Black and Scholes formula. The respective implied volatilities were calculated using the latter. The analysis emphasized an important aspect: the value of the implied volatility is obtained assuming that the options are priced using the Black-Scholes model.

We then analyzed data from Twitter Inc. Using the prices of the underlying and the options, we interpolated the implied volatilities for various strike price values, with which the prices of the respective options were calculated. And then, exploiting the theory of Breeden and Litzenberger, the implied distribution was calculated and compared with the log-normal distribution. The analysis showed that the implied distribution has heavier tails than the log-normal. This allowed us to hypothesize a trading strategy based on "deep-out-of-the-money" calls or puts.

References

Breeden, D. T., and R. H. Litzenberger. 1978. "Prices of State-Contingent Claims Implicit in Options Prices." *Journal of Business*.

Bruno Rémillard. 2013. *Statistical Methods for Financial Engineering*.

G. Castellani, M.D. Felice, F. Moriconi. 2019. *Manuale Di Finanza III*.

Hull J.C. 2014. *Options, Futures and Other Derivatives*.

Paul Wilmotti. 2003. *Introduzione Alla Finanza*.

R Core Team. 2019. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org.

Wikipedia contributors. 2022. "SABR Volatility Model — Wikipedia, the Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=SABR_volatility_model&oldid=1090241278.