Finance Quantitative

Pricing Vanna-Volga

Patrick Hénaff

Version: 30 mars 2023

The purpose of this problem set is to explore the Vanna-Volga pricing model.

Getting started

- Using the Rmetrics library (fOptions), verify that you know how to compute the price and the "greeks" of a vanilla option.
- Identify or write a robust function to compute the implied volatility, given the price of a vanilla option.

Questions

Volatility Interpolation

Given the implied volatility at three strikes, we will use the Vanna-Volga pricing method to interpolate the volatility curve. Assume $r = 0, b = 0, T = 1, \text{Spot} = 100$. The Black-Scholes volatility for three strikes is given below.

Strike	80.00	100.0	120.000
Volatility	0.32	0.3	0.315

```
T <- 1
Spot <- 100
r <- 0
b <- 0
eps <- 1.e-3
sigma <- .3
# Benchmark data: (strike, volatility)
VolData <- list(c(80, .32), c(100, .30), c(120, .315))
```

Fonction de calcul de la volatilité implicite

```
ImpliedVolNewton <- function(p, TypeFlag, S, X, Time, r, b,
                             sigma=NULL, maxiter=500, tol=1.e-5) {

  if(is.null(sigma))
    s <- sqrt(2*abs(log(S/(X*exp((b*T)))))/T)
```

```
  else
    s <- sigma

  not_converged <- T
  i=1
  vega <- GBSGreeks('vega', TypeFlag, S, X, Time, r, b, s)
  while(not_converged & (i<maxiter)) {
    err <- (p-GBSOption(TypeFlag, S, X, Time, r, b, s)@price)
    s <- s + err/vega
    # print(paste('i:', i, 's:', s))
    not_converged <- (abs(err/vega) > tol)
    i <- i+1
  }
s
}
```

Let's first define an array of pricing functions for the benchmark instruments:

```
C1 <- function(vol=sigma, spot=Spot) GBSOption(TypeFlag='c', S=spot, X=VolData[[1]][1], Time=T, r=r, b=

C2 <- function(vol=sigma, spot=Spot) GBSOption(TypeFlag='c', S=spot, X=VolData[[2]][1], Time=T, r=r, b=

C3 <- function(vol=sigma, spot=Spot) GBSOption(TypeFlag='c', S=spot, X=VolData[[3]][1], Time=T, r=r, b=

C <- c(C1, C2, C3)
```

1. Write a utility functions to compute the risk indicators, all by finite difference:

Solution

```
Vega <- function(f, vol, spot=Spot) (f(vol+eps, spot)-f(vol-eps, spot))/(2*eps)

Vanna <- function(f, vol, spot=Spot) {
  (Vega(f, vol, spot+1)-Vega(f, vol, spot-1))/2
}

Volga <- function(f, vol) {
    (Vega(f,vol+eps)-Vega(f,vol-eps))/(2*eps)
}
```

2. Compute vectors of vega, vanna, volga for the three hedge instruments

Solution

```
B.vega <- sapply(1:3, function(i) Vega(C[[i]], sigma))
B.vanna <- sapply(1:3, function(i) Vanna(C[[i]], sigma))
B.volga <- sapply(1:3, function(i) Volga(C[[i]], sigma))
```

Strike	Vol	Vega	Vanna	Volga
80	0.320	26.757	-0.529	47.339
100	0.300	39.448	0.197	-2.959
120	0.315	35.926	0.907	41.536

3. Choose a new strike for which we want to compute the implied volatility.
4. Compute the risk indicators for a call option struck at that strike.
5. Compute the Vanna-Volga price adjustment and the corresponding implied volatility.

Solution

On désire interpoler la volatilité au strike $Knew = 90$.

```
Knew <- 90

O <- function(vol=sigma, spot=Spot) GBSOption('c', S=spot,
       X=Knew, Time=T, r=r, b=b, sigma=vol)@price

# Fonction de prix Black-Scholes
O.BS <- O()
```

Fonctions de calcul des indicateurs de risque:

```
O.vega <- Vega(O, sigma)
O.vanna <- Vanna(O, sigma)
O.volga <- Volga(O, sigma)

# Difference entre les prix de marché et les prix Black-Scholes
B.cost <- sapply(1:3, function(i) C[[i]](VolData[[i]][2]) - C[[i]](sigma))
```

Calcul de la correction de prix Vanna-Volga:

```
A <- t(matrix(c(B.vega, B.vanna, B.volga),  nrow=3))
x <- matrix(c(O.vega, O.vanna, O.volga), nrow=3)
w <- solve(A, x)
vanna.volga.cor <- t(w) %*% matrix(B.cost, nrow=3)

O.Price <- as.numeric(O.BS + vanna.volga.cor)
```

Volatilité implicite correspondante:

```
# implied volatility
O.iv <- ImpliedVolNewton(O.Price, 'c', Spot, Knew, T, r, b,
                         sigma=sigma)
```

Call de strike $K = 90$: Prix Black-Scholes (vol ATM): 17.01, Prix avec ajustement Vanna-Volga: 17.16.

6. Wrap the above logic in a function in order to interpolate/extrapolate the vol curve from $K = 70$ to $K = 130$

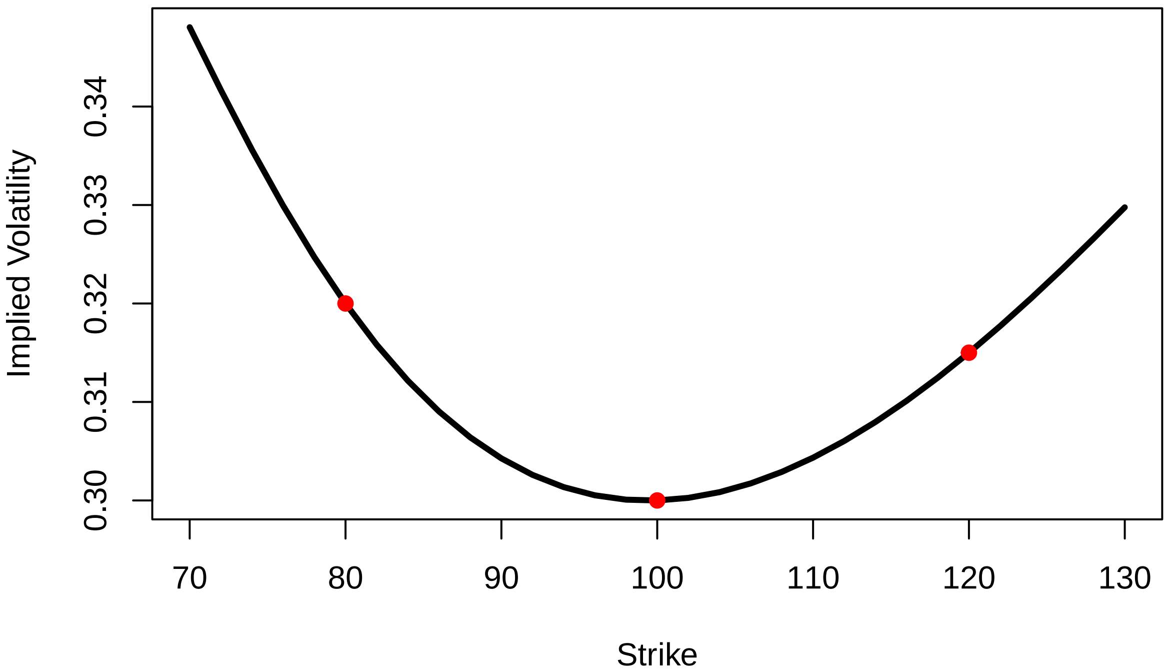
Solution

```
VVVol <- function(K) {

## Calcul de la vol implicite pour un strike K donné

  O <- function(vol=sigma, spot=Spot) GBSOption('c', S=spot,
       X=K, Time=T, r=r, b=b, sigma=vol)@price

  # Its Black-Scholes price
  O.BS <- O()

  # risk indicators for new option
  O.vega <- Vega(O, sigma)
  O.vanna <- Vanna(O, sigma)
  O.volga <- Volga(O, sigma)

  # calculation of price adjustment
  A <- t(matrix(c(B.vega, B.vanna, B.volga),  nrow=3))
  x <- matrix(c(O.vega, O.vanna, O.volga), nrow=3)
  w <- solve(A, x)
  CF <- t(w) %*% matrix(B.cost, nrow=3)

  # implied volatility
  iv <- ImpliedVolNewton(as.numeric(O.BS+CF), 'c', Spot, K, T, r, b,
                         sigma=sigma)
  iv
}
```

On éxécute cette fonction pour une plage de strikes:

```
v <- sapply(seq(70, 130, 2), VVVol)
```

La courbe de volatilité interpolée figure ci-dessous. On vérifie bien que l'interpolation passe par les 3 points de référence.



Pricing a digital call

Recall that a digital call with strike K pays one euro if $S_T \geq K$, and nothing otherwise.

Using the same logic as in the previous question, price a digital call, maturity $T = 1$, struck at $K = 105$.

Solution

Les données du problème:

```
T <- 1
Spot <- 100
r <- 0
b <- 0

# Vol ATM
sigma <- .30

# strike
Strike <- 105

# Fonction de prix BS d'un call digital

BinaryPrice <- function(PutCall, S, K, T, r, b, sigma) {
  d1 <- (log(S/K) + (b+sigma^2/2)*T)/(sigma*sqrt(T))
  d2 <- d1 - sigma*sqrt(T)


  if(PutCall == 'c')
    px <- 100*exp(-r*T)*pnorm(d2)
```

```
  else
    px <- 100*exp(-r*T)*pnorm(-d2)

px
}

Bin <- function(vol=sigma, spot=Spot) BinaryPrice('c', S=spot,
       K=Strike, T=T, r=r, b=b, sigma=vol)

# Prix BS d'un call digital de strike K=105
Bin.BS <- Bin()
```

Les instruments de référence sont les mêmes que dans la question précédente. Il reste à calculer le vega, vanna, volga du call digital, et la correction de prix.

```
Bin.vega <- Vega(Bin, sigma)
Bin.vanna <- Vanna(Bin, sigma)
Bin.volga <- Volga(Bin, sigma)


A <- t(matrix(c(B.vega, B.vanna, B.volga),  nrow=3))
x <- matrix(c(Bin.vega, Bin.vanna, Bin.volga), nrow=3)
w <- solve(A, x)
CF <- t(w) %*% matrix(B.cost, nrow=3)
```

Le prix corrigé est finalement:

```
Bin.prix.VV <- Bin.BS + CF
```

Call digital de strike 105:

- Prix Black-Scholes: 37.73
- Prix avec correction Vanna-Volga: 35.96

Pour confirmation, on peut comparer cette évaluation à celle donnée par la densité de S_T implicite au smile (voir TP-Shimko). Pour cela, ajustons une forme quadratique au smile de volatilité:

```
lm.smile <- lm(V ~ poly(K,2,raw=TRUE))
coef <- lm.smile$coefficients
smileVol <- function(K) {
  sum(coef * c(1, K, K*K))
}
```

Calculons la densité de S_T par la formule de Breeden-Litzenberger:

```
  d2CdK2 <- function(vol, S, K, T, r, b) {
   dK <- K/10000
   c <- GBSOption('c', S, K, T, r, b, vol(K))@price
   cPlus <- GBSOption('c', S, K+dK, T, r, b, vol(K+dK))@price
   cMinus <- GBSOption('c', S, K-dK, T, r, b, vol(K-dK))@price
   (cPlus-2*c+cMinus)/(dK^2)
```

```
  }

smile.pdf <- function(S0, K, T, r, b) {
  d2CdK2(smileVol, S0, K, T, r, b) * exp(r*T)
}
```

Le prix de l'option digitale est calculé par intégration numérique:

```
# Valeur à maturité
digital.payoff <- function(S.T) {
  if(S.T>Strike)
    100
  else
    0
}

# Fonctions à intégrer numériquement:
digital.smile <- function(K){
  digital.payoff(K)*smile.pdf(Spot, K, T, r, b)
}

Bin.prix.smile <- exp(-r*T)*integrate(
  Vectorize(digital.smile),
  lower=Strike, upper=700)$value
```

Finalement, on obtient les estimations suivantes:

- Prix Black-Scholes: 37.73
- Prix avec correction Vanna-Volga: 35.96
- Prix à partir de la distribution implicite à maturité: 36.48