Chapter 13

Building an Equity Volatility Surface

The purpose of this chapter is to present a step-by-step procedure for building a volatility surface, using the S&P 500 index options for illustration.

13.1 The Data

Settlement prices for call and put options on the S&P 500 index are available for download at www.cboe.com/DelayedQuote/QuoteTableDownload.aspx. A fragment of the data file is shown in Table 13.1.

Listing 13.1: Fragment of S&P 500 option settlement data downloaded from CBOE website. Data for calls are shown here. Data for puts follow in the next seven columns, with the same structure.

```
SPX (SP 500 INDEX)     1290.59    +7.24
Jan 24 2011 @ 14:03 ET
Calls                                Last Sale   Net Bid     Ask      Vol Open Int ...
11 Jan 1075.00 (SPXW1128A1075-E) 0.0          0.0 215.30 217.00 0    0          ...
11 Jan 1100.00 (SPXW1128A1100-E) 0.0          0.0 190.60 191.80 0    0          ...
```

The ticker for the S&P 500 index options (i.e. SPXW1128A1075-E) should be interpreted as follows:

- characters 5-9: the expiry date of the option, and the option type (call or put):
  - characters 5-6: year;
  - characters 7-8: day of month;
  - character 9: month, coded differently for calls and puts. For calls: A (January) to L (December); for puts: M (January) to X (December);

- characters 10-13: option strike.

For example, SPXW1128A1075-E is the ticker of a European call, strike 1075, expiring on January 28, 2011.
The ticker is parsed with the following function:

```
> SPX_months <- hash(keys = unlist(strsplit("ABCDEFGHIJKLMNOPQRSTUVWX",
     split = "")), values = c(1:12, 1:12))
> spx_symbol = "\\(SPX(1[0-9])([0-9]{2})([A-Z])([0-9]{3,4})-E\\)"
> parse_SPX_expiry <- function(tokens) {
     year = 2000 + as.integer(tokens[2])
     day = as.integer(tokens[3])
     month = SPX_months[[tokens[4]]]
     myDate(sprintf("%02d-%02d-%04d", day, month, year))
 }
```

The next function reads the csv file and builds a data frame of call and put prices:

```
> read_SPX_file <- function(option_data_file) {
     df = read.csv(option_data_file, sep = ",", header = F, nrow = 2)
     dtTradeTokens = unlist(strsplit(as.character(df[2, "V1"]),
         " ", fixed = T)[1:3])
     dtTrade = myDate(paste(dtTradeTokens[c(2, 1, 3)], collapse = ""))
     spot = df[1, "V2"]
     df = read.csv(option_data_file, sep = ",", header = T, skip = 2)
     call_df = df[, c("Calls", "Bid", "Ask")]
     colnames(call_df) = c("Spec", "PBid", "PAsk")
     call_df["CP"] = "C"
     put_df = df[, c("Puts", "Bid.1", "Ask.1")]
     colnames(put_df) = c("Spec", "PBid", "PAsk")
     put_df["CP"] = "P"
     df_all = rbind(call_df, put_df)
     df_all$Type = "Euro"
     list(spot = spot, dtTrade = dtTrade, quotes = df <- all)
 }
```

Finally, the following script filters out bad data, and creates a data frame with columns:

- dtTrade: quote date, or time stamp
- Strike
- dtExpiry: option expiry date
- CP: call/put flag, coded as C/P or Call/Put
- Spot: price of underlying asset
- Type: European/American
- PBid: bid price

- PAsk: ask price

```
option_data_file = <path to csv file>

res = read_SPX_file(option_data_file)

df_all <- res$df_all

# parse Spec column for strike and expiry date
# only retain valid data
tokens <- str_match(df_all[,'Spec'], spx_symbol)

good_rows <- apply(tokens, 1, function(x) all(!is.na(x)))

df_all <- df_all[good_rows,]
tokens <- tokens[good_rows,]
df_all['Strike'] = as.numeric(tokens[,5])

dtExpiry = apply(tokens, 1, parse_SPX_expiry)
df_all['dtExpiry'] = as.Date(dtExpiry, origin='1970-01-01')

df_all = df_all[(df_all['Strike'] > 0) & (df_all['PBid']>0)
                    & (df_all['PAsk']>0),]

df_all['dtTrade'] = res$dtTrade
df_all['Spot'] = res$spot

print(paste('Nb of records processed:', length(df_all$PBid)))
save(df_all, file='df_SPX.rda')
write.csv(df_all, 'df_SPX.csv')
```

13.2 ESTIMATION OF THE VOLATILITY SURFACE

Implied risk-free rate and dividend yield Rather than using exogenous information, we will estimate the risk-free rate and dividend yield implied by the option market itself.
Recall the put-call parity relationship for European options with continuous dividends:

$$C_t(T) - P_t(T) = S_t e^{-d(T)(T-t)} - K e^{-r(T)(T-t)}$$

where

$T =$ expiry time

$C_t(T) =$ price at time t of call maturing at T

$P_t(T) =$ price at time t of put maturing at T

$S_t =$ spot price of underlying asset

$d(T) =$ continuous dividend yield

$r(T) =$ risk-free rate

Because of measurements errors and bid-ask spreads, this relationship does not hold exactly. However, for each maturity, we can estimate the terms $e^{-d(T-t)}$ and $e^{-r(T-t)}$ by linear regression. Consider the linear model:

$$C_t(T) - P_t(T) = a_0 + a_1 K$$

which yields the following estimates for the risk-free rate and dividend yield of maturity T.

$$\begin{aligned} r_(T) &= -\frac{1}{T}\ln(-a_1) \\ d_t(T) &= \frac{1}{T}\ln\left(\frac{S_t}{a_0}\right) \end{aligned}$$

Forward at-the-money volatility The next step is to estimate the implied volatility of an option struck at the forward price. In general, such option is not traded, and the volatility must therefore be estimated. The calculation involves 3 steps, performed separately on calls and puts:

1. Estimate the bid ($\sigma_b(K)$) and ask ($\sigma_a(K)$) Black-Scholes volatility for each strike K.

2. Compute a mid-market implied volatility for each strike:

$$\sigma(K) = \frac{\sigma_b(K) + \sigma_a(K)}{2}$$

3. Let F be the forward price. The corresponding mid-market implied volatility is computed by linear interpolation between the two strikes bracketing F.

4. The forward ATM volatility is the average of the volatilities computed on calls and puts.

Quick delta The "quick delta" (QD) is a popular measure of moneyness, inspired by the definition of the delta of a European call:

$$QD(K) = N\left(\frac{1}{\sigma\sqrt{T}}\ln\left(\frac{F_T}{K}\right)\right)$$

Note that $QD(F_T) = 0.5$, for all maturities, while the regular forward delta is a function of time to expiry. This property of "quick delta" makes it

easy to interpret. In addition, being bounded by 0 and 1, it is a convenient measure of moneyness for representing the volatility smile.

The function `compute.iv()`, reproduced below, performs these calculations for one expiry date.

```
> compute.iv <- function(group, tMin = 0, nMin = 0, QDMin = 0,
    QDMax = 1, keepOTMData = TRUE) {
    df.out = data.frame()
    dtTrade = group$dtTrade[1]
    dtExpiry = group$dtExpiry[1]
    spot = group$Spot[1]
    daysToExpiry = as.numeric(dtExpiry - dtTrade)
    timeToMaturity = daysToExpiry/365
    if (timeToMaturity < tMin)
        return(df.out)
    df_call = subset(group, Type == "C", select = c(Strike, PBid,
        PAsk))
    df_put = subset(group, Type == "P", select = c(Strike, PBid,
        PAsk))
    if ((nrow(df_call) < nMin) | (nrow(df_put) < nMin))
        return(df.out)
    df_call$PremiumC = (df_call$PBid + df_call$PAsk)/2
    df_put$PremiumP = (df_put$PBid + df_put$PAsk)/2
    df_all = merge(df_call, df_put, by = "Strike", all = TRUE)
    df_all$CP = df_all$PremiumC - df_all$PremiumP
    model = lm(CP ~ Strike, df_all)
    a0 = model$coefficients[1]
    a1 = model$coefficients[2]
    iRate = -log(-a1)/timeToMaturity
    dRate = log(spot/a0)/timeToMaturity
    discountFactor = exp(-iRate * timeToMaturity)
    Fwd = spot * exp((iRate - dRate) * timeToMaturity)
    print(paste("Fwd: ", round(Fwd, 2), " int rate: ", round(iRate *
        100, 2), "div yield: ", round(dRate * 100, 2)))
    impvol <- function(aRow, cp, p.column) {
        res = try(GBSVolatility(price = aRow[p.column], TypeFlag = cp,
            S = spot, X = aRow["Strike"], Time = timeToMaturity,
            r = iRate, b = iRate - dRate))
        if (!is.numeric(res))
            res = NA
        res
    }
    df_call$IVBid <- apply(df_call, 1, FUN = impvol, cp = "c",
        p.column = "PBid")
    df_call$IVAsk = apply(df_call, 1, FUN = impvol, cp = "c",
        p.column = "PAsk")
    df_call$IVMid <- (df_call$IVBid + df_call$IVAsk)/2
    df_put$IVBid = apply(df_put, 1, FUN = impvol, cp = "p", p.column = "PBid")
    df_put$IVAsk = apply(df_put, 1, FUN = impvol, cp = "p", p.column = "PAsk")
    df_put$IVMid <- (df_put$IVBid + df_put$IVAsk)/2
    atm.vol.c = approx(df_call$Strike, df_call$IVMid, xout = Fwd)$y
    atm.vol.p = approx(df_put$Strike, df_put$IVMid, xout = Fwd)$y
    atmVol = (atm.vol.c + atm.vol.p)/2
    print(paste("ATM vol: ", atmVol))
```

```
        df_call$QuickDelta = pnorm(log(Fwd/df_call$Strike)/(atmVol *
            sqrt(timeToMaturity)))
        df_put$QuickDelta = pnorm(log(Fwd/df_put$Strike)/(atmVol *
            sqrt(timeToMaturity)))
        df_call = df_call[(df_call$QuickDelta >= QDMin) & (df_call$QuickDelta <=
            QDMax), ]
        df_put = df_put[(df_put["QuickDelta"] >= QDMin) & (df_put["QuickDelta"] <=
            QDMax), ]
        df_call$PremiumC <- NULL
        df_call$Type <- "C"
        df_put$PremiumP <- NULL
        df_put$Type <- "P"
        df_cp = rbind(df_call, df_put)
        df_cp$iRate = iRate
        df_cp$iDiv = dRate
        df_cp$ATMVol = atmVol
        df_cp$Fwd = Fwd
        df_cp$dtTrade = dtTrade
        df_cp$dtExpiry = dtExpiry
        df_cp
    }
```

The function must be called for each expiry date. The implied volatility surface is built one expiry date at a time, as follows:

```
  load(file = 'df_SPX.rda')
  SPX.IV = data.frame()
  for(dt in unique(df_SPX$dtExpiry)) {
  dt.2 <- as.Date(dt, origin='1970-01-01')
  print(paste('DtExpiry:', dt.2))
  tmp = subset(df_SPX, dtExpiry == dt.2)
  df.tmp = compute.iv(tmp, tMin=1/12, nMin=6, QDMin=.2, QDMax=.8,
       keepOTMData=TRUE)
  SPX.IV = rbind(SPX.IV, df.tmp)
}
```

The resulting data set may be found in package `empfin`. The data is represented in Figure 13.1.

```
    > data(SPX.IV, package = "empfin")
    > df_call <- subset(SPX.IV, Type == "C")
    > x <- df_call$QuickDelta
    > y <- as.numeric(df_call$dtExpiry)
    > z <- df_call$IVMid
    > s <- interp(x, y, z)
    > nrz = nrow(s$z)
    > ncz = ncol(s$z)
    > jet.colors = colorRampPalette(c("blue", "green"))
    > nbcol = 100
    > color = jet.colors(nbcol)
    > zfacet = s$z[-1, -1] + s$z[-1, -ncz] + s$z[-nrz, -1] + s$z[-nrz,
        -ncz]
    > facetcol = cut(zfacet, nbcol)
    > TTM <- as.numeric(as.Date(s$y, origin = "1970-01-01") - df_call$dtTrade[1])/365
    > persp(s$x, TTM, s$z * 100, col = color[facetcol], xlab = "\nQuick Delta",
```

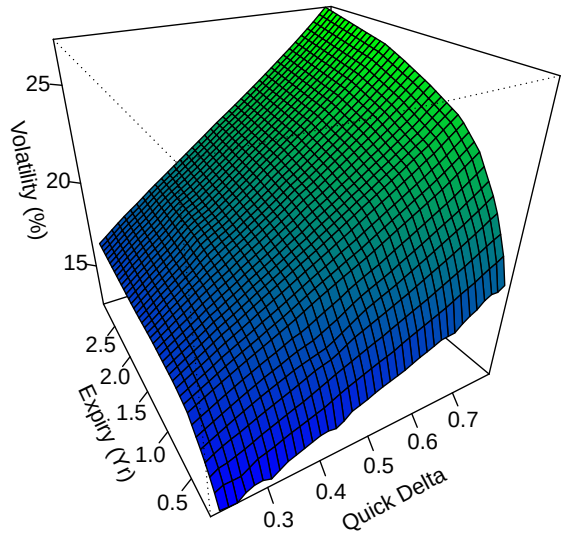

FIGURE 13.1: Implied volatility surface, SPX index options, January 24, 2011

```
ylab = "\nExpiry (Yr)", zlab = "\nVolatility (%)", theta = -30,
phi = 30, ticktype = "detailed")
```