1. Consider the parameterization of the N-step binomial model with $\tilde{p} = \tilde{q} = 1/2$, $\Delta t = T/N$, continuously compounded interest rate R,

$$u = e^{R\Delta t}(1 + \sqrt{e^{\sigma^2\Delta t} - 1}), \quad \text{and}$$
$$d = e^{R\Delta t}(1 - \sqrt{e^{\sigma^2\Delta t} - 1})$$

(a) Show that

$$u = 1 + \sigma\sqrt{\Delta t} + R\Delta t + O(\Delta t^{3/2}) \quad \text{and}$$
$$d = 1 - \sigma\sqrt{\Delta t} + R\Delta t + O(\Delta t^{3/2})$$

as $\Delta t \to 0$.
[Hint: recall that $\sqrt{1+x} = 1 + \frac{1}{2}x + O(x^2)$*.]*
Solution:
Assume $0 \leq \Delta t \leq \min(1, \sigma^{-1})$. Then, by Taylor's Theorem we have that

$$e^{\sigma^2\Delta t} = 1 + \sigma^2\Delta t + K(\sigma^2\Delta t)^2$$

for some (positive) constant K. Therefore, we may write

$$u = e^{R\Delta t}(1 + \sqrt{\sigma^2\Delta t + K(\sigma^2\Delta t)^2}) = e^{R\Delta t}(1 + \sigma\sqrt{\Delta t}\sqrt{1 + K\sigma^2\Delta t}).$$

Applying the hint with $x = K\sigma^2\Delta t$ gives that (for sufficiently small Δt)

$$\begin{aligned} u &= e^{R\Delta t}(1 + \sigma\sqrt{\Delta t}(1 + \frac{1}{2}K\sigma^2\Delta t + O([K\sigma^2\Delta t]^2))) \\ &= e^{R\Delta t}(1 + \sigma\sqrt{\Delta t} + \frac{1}{2}K\sigma^3(\Delta t)^{3/2} + \sigma\sqrt{\Delta t}O(\Delta t^2)) \\ &= e^{R\Delta t}(1 + \sigma\sqrt{\Delta t} + O(\Delta t^{3/2})). \end{aligned}$$

Further, using $e^x = 1 + x + O(x^2)$ with $x = R\Delta t$ (assume additionally that $\Delta t \leq R^{-1}$) we have

$$\begin{aligned} u &= (1 + R\Delta t + O(\Delta t^2))(1 + \sigma\sqrt{\Delta t} + O(\Delta t^{3/2})) \\ &= 1 + R\Delta t + O(\Delta t^2) + \sigma\sqrt{\Delta t}(1 + R\Delta t + O(\Delta t^2)) + O(\Delta t^{3/2}))(1 + R\Delta t + O(\Delta t^2)) \\ &= 1 + R\Delta t + \sigma\sqrt{\Delta t} + \sigma\sqrt{\Delta t}(R\Delta t + O(\Delta t^2)) + O(\Delta t^{3/2}))(1 + R\Delta t + O(\Delta t^2)) \\ &= 1 + R\Delta t + \sigma\sqrt{\Delta t} + O(\Delta t^{3/2}) \end{aligned}$$

as desired. The proof that $d = 1 + R\Delta t - \sigma\Delta t + O(\Delta t^{3/2})$ is similar.

(b) **[MAST 729H/881 Only]** Consider the parameterization of the N-step binomial model with $\Delta t = T/N$, continuously compounded interest rate R,

$$v = e^{\sigma^2\Delta t} \tag{1}$$
$$u = \frac{1}{2}ve^{R\Delta t}(v + 1 + \sqrt{v^2 + 2v - 3}) \tag{2}$$
$$d = \frac{1}{2}ve^{R\Delta t}(v + 1 - \sqrt{v^2 + 2v - 3}) \tag{3}$$

and risk-neutral probability of an up-step $\tilde{p} = \frac{e^{R\Delta t}-d}{u-d}$.

Show that the expected return of the stock price over an interval of length Δt in the above model is given by $r = 1 - e^{R\Delta t}$. *[Note that this matches the expected return, under the risk-neutral measure, of the stock price over the same time interval in the Black-Scholes model.]*

Solution:

We wish to calculate $r = E_m^Q\left[\frac{S_{m+1}-S_m}{S_m}\right]$. Fixing the outcome of the coin-tosses up to time m we have

$$S_{m+1}(\omega_1 \cdots \omega_m \omega_{m+1}) = S_m(\omega_1 \cdots \omega_m)\xi_{m+1}$$

where

$$\xi_{m+1} = \begin{cases} u & \text{if} \omega + m + 1 = H \\ d & \text{if} \omega + m + 1 = T \end{cases}$$

where H denotes *heads* (an up-step) and T denotes *tails* (a down-step) on the $m+1$-th coin toss. Therefore,

$$E_m^Q\left[\frac{S_{m+1}}{S_m}\right] = E_m^Q[\xi_{m+1}] = [\tilde{p}u + (1-\tilde{p})d].$$

However, since $\tilde{p}$ is the risk-neutral probability, we must have

$$\tilde{p}u + (1-\tilde{p})d = e^{R\Delta t}$$

otherwise there is arbitrage (you can also check this algebraically). Hence

$$E_m^Q\left[\frac{S_{m+1}}{S_m}\right] = e^{R\Delta t}.$$

That is, $r = 1 - e^{R\Delta t}$.

2. **[Binomial Implied Volatility]** Consider the N-step binomial model for the asset price

$$S_n^{i,N} = u^n d^{N-n} S_0$$

where $0 \le n \le i$, $0 \le i \le N$, and n denotes the number of *heads* or *up-steps* in the binomial tree for S up to time-step i. Let $V_n^{i,N}$ denote the corresponding value of a European option on S at node (i, n) . Then we can show that

$$V_0^{0,N} = e^{-rT}\sum_{k=0}^{N}\binom{N}{k}\tilde{p}^k\tilde{q}^{N-k}V_k^{N,N} \tag{4}$$

where $\binom{N}{k}$ denotes the binomial coefficient $\binom{N}{k} = \frac{N!}{k!(N-k)!}$.

(a) With $\tilde{p} = \tilde{q} = 1/2$, $\Delta t = T/N$, and continuously compounded interest rate R where u and d are given by

$$u = e^{\sigma\sqrt{\Delta t}+(R-\frac{1}{2}\sigma^2)\Delta t}, \quad \text{and} \tag{5}$$

$$d = e^{-\sigma\sqrt{\Delta t}+(R-\frac{1}{2}\sigma^2)\Delta t}. \tag{6}$$

write a program which implements formula (4) as a function of the inputs $(S_0, R, \sigma, K, \tau, N)$ to find the price, C_0^N, at time zero of a European call option. Use small scale examples you can

work out by hand to test your program. Create a corresponding program for put options. *[Hint: For large values of N you may run into computational problems calculating the binomial coefficients. Consider using log-binomial coefficients and Stirling's approximation:*

$$\ln(n!) = n\ln(n) - n + O(\ln(n)) \quad \textit{or} \quad n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n.$$

There are other simplifications that can be made to avoid computational difficulties.] Solution: The following *R* code implements the formula (4) directly. To simplify the implementation I have used the built in binomial coefficient function *choose*. Students should create their own *choose* function.

```
BinCall <- function(S0,R,sigma,K,T,N){
        pQ <- 0.5
        qQ <- 0.5
        SN <- rep(0,N+1)
        CN <- rep(0,N+1)
        bcoef <- rep(0,N+1)
        tempC <- rep(0,N+1)
        C0 <- 0
        Dt <- T/N
        u <- exp(sigma*sqrt(Dt)+(R-0.5*sigma^2)*Dt)
        d <- exp(-sigma*sqrt(Dt)+(R-0.5*sigma^2)*Dt)
        for (j in 1:(N+1)){
           SN[j] <- S0*u^(j-1)*d^(N-(j-1))
           CN[j] <- max(0,SN[j]-K)
           bcoef[j] <- choose(N,j-1)
           tempC[j] <- CN[j]*bcoef[j]*pQ^(j-1)*(qQ)^(N-(j-1))
        }
        C0 <- exp(-R*T)*sum(tempC)
        return(C0)
}
```

Unfortunately for very large values of N the binomial coefficients are too large for *R* to handle. To speed up the computations we ignore terms where the option payoff is out of the money. We consider the logs of the individual (non-zero) terms in the sum:

$$lV_j := \ln V_j^{N,N} + \ln\binom{N}{j} + N\ln(\tilde{p}) - RT.$$

We also use the fact that $\tilde{p} = \tilde{q} = 0.5$ to decrease the number of computations. We may also scale by the largest (non-zero) log value $l\bar{V} := \max_j lV_j$ to write the sum as

$$e^{l\bar{V}} \sum_{j=0}^{N} e^{lV_j - l\bar{V}}$$

since for very large values of lV_j the difference $lV_j - l\bar{V}$ will be small thus increasing the accuracy of the computation.

The following code uses the log-binomial coefficients (*lchoose*) which can handle larger values of N.

```
BinCall2 <- function(S0,R,sigma,K,T,N){
   pQ <- 0.5
   qQ <- 0.5
   SN <- rep(0,N+1)
   CN <- rep(0,N+1)
   lbcoef <- rep(0,N+1)
   tempC <- rep(0,N+1)
   C0 <- 0
   Dt <- T/N
   u <- exp(sigma*sqrt(Dt)+(R-0.5*sigma^2)*Dt)
   d <- exp(-sigma*sqrt(Dt)+(R-0.5*sigma^2)*Dt)
   for (j in 1:(N+1)){
      SN[j] <- S0*u^(j-1)*d^(N-(j-1))
      CN[j] <- max(0,SN[j]-K)
      }
   for (j in 1:(N+1)){
      if (CN[j]>0){
      lbcoef[j] <- lchoose(N,j-1)
      tempC[j] <- log(CN[j]) + N*log(0.5) + lbcoef[j] - R*T
      }
   }
   tempCmax <- max(tempC)
   ltempCmax <- 0
   for (j in 1:(N+1)){
      if (CN[j]>0){
         ltempCmax <- ltempCmax + exp(tempC[j]-tempCmax)
      }
   }
   C0 <- ltempCmax*exp(tempCmax)
   return(C0)
}
```

You can adapt the above code to not use the built in function *lchoose* the log-binomial coefficients accurately for large N using the approximation

$$\frac{1}{N}\ln\binom{N}{k} = \ln\left(\frac{N}{N-k}\right) + \frac{k}{N}\ln\left(\frac{N-k}{k}\right) + \frac{O(\ln(N))}{N}$$

that can be derived from the approximation given in the hint.

(b) Write a program that applies the bisection algorithm to the program from part (a), given starting values that satisfy the appropriate conditions, to solve

$$C_0^{0,N} = C_0^{obs}$$

for the **binomial implied volatility** where C_0^{obs} is an observed market price. Specify an error tolerance parameter ϵ such that the bisection algorithm stops when $|C_0^{0,N} - C_0^{obs}| < \epsilon$. Modify your code to create a program which calibrates to observed put option prices.

Solution:

The following code implements the bisection algorithm for the observed call prices for various values of N

```
# Starting Parameters

S0 <- 4.75
R <- 0.0492
T <- 59/365

# Observed Values

C_strike <- c(4.5,4.75,5,5.25,5.5,5.75)
C_obs <- c(.33,.16,.06,.02,.01,.01)
Calls <- data.frame(C_strike,C_obs)

n_calls <- length(Calls$C_obs)

for (k in 1:5){
   N <- 10^k
   sigma_imp <- rep(0,n_calls)
   for (i in 1:n_calls){
      sigmaa <- 0.005
      sigmab <- 1
      error <- 1
      while (error > 10^-6){
       sigma_imp[i] <- (sigmaa + sigmab)/2
       if (BinCall2(S0,R,sigma_imp[i],Calls$C_strike[i],T,N) < Calls$C_obs[i]){
          sigmaa <- sigma_imp[i]
       }
       if (BinCall2(S0,R,sigma_imp[i],Calls$C_strike[i],T,N) > Calls$C_obs[i]){
          sigmab <- sigma_imp[i]
       }
       error <- abs(BinCall2(S0,R,sigma_imp[i],Calls$C_strike[i],T,N)-Calls$C_obs[i])
      }
   }
   print(sigma_imp)
}
```

For the put option we adjust our code from part (a) to use the formula (4) to calculate put prices.

```
BinPut2 <- function(S0,R,sigma,K,T,N){
   pQ <- 0.5
   qQ <- 0.5
   SN <- rep(0,N+1)
   PN <- rep(0,N+1)
   lbcoef <- rep(0,N+1)
   tempP <- rep(0,N+1)
   P0 <- 0
   Dt <- T/N
   u <- exp(sigma*sqrt(Dt)+(R-0.5*sigma^2)*Dt)
   d <- exp(-sigma*sqrt(Dt)+(R-0.5*sigma^2)*Dt)
   for (j in 1:(N+1)){
      SN[j] <- S0*u^(j-1)*d^(N-(j-1))
      PN[j] <- max(0,K-SN[j])
      }
   for (j in 1:(N+1)){
      if (PN[j]>0){
      lbcoef[j] <- lchoose(N,j-1)
      tempP[j] <- log(PN[j]) + N*log(0.5) + lbcoef[j] - R*T
      }
   }
   tempPmax <- max(tempP)
   ltempPmax <- 0
   for (j in 1:(N+1)){
      if (PN[j]>0){
         ltempPmax <- ltempPmax + exp(tempP[j]-tempPmax)
      }
   }
   P0 <- ltempPmax*exp(tempPmax)
   return(P0)
}
```

We also adjust the above code to implement the bisection algorithm to calculate binomial implied volatilities for put options.

```
# Starting Parameters

S0 <- 4.75
R <- 0.0492
T <- 59/365

# Observed Values

P_strike <- c(4,4.25,4.5,4.75,5,5.25)
P_obs <- c(.02,.04,.09,.2,.38,.59)
Puts <- data.frame(P_strike,P_obs)

n_puts <- length(Puts$P_obs)

# Bisection

for (k in 1:5){
   N <- 10^k
   sigma_imp <- rep(0,n_puts)
   for (i in 1:n_puts){
      sigmaa <- 0.005
      sigmab <- 1
      error <- 1
      while (error > 10^-6){
        sigma_imp[i] <- (sigmaa + sigmab)/2
        if (BinPut2(S0,R,sigma_imp[i],Puts$P_strike[i],T,N) < Puts$P_obs[i]){
          sigmaa <- sigma_imp[i]
        }
        if (BinPut2(S0,R,sigma_imp[i],Puts$P_strike[i],T,N) > Puts$P_obs[i]){
          sigmab <- sigma_imp[i]
        }
        error <- abs(BinPut2(S0,R,sigma_imp[i],Puts$P_strike[i],T,N)-Puts$P_obs[i])
      }
   }
   print(sigma_imp)
}
```

(c) Apply your programs from part (a)-(b) to the observed price data from July 29, 2002 given below to calculate the binomial implied volatilities for call and put option ask prices which expire on September 26, 2002. Assume $S_0 = 4.75$, and $R = 0.0492$ per annum (use a time scale in units of years with 365 days per year, you will need to calculate the number of years T to expiry). Assume $\epsilon = 10^{-6}$. Calculate each of the implied volatilities for increasing $N = 10^k$, $k = 1, 2, \ldots, 5$ (or more if your programs allow) and report your results in a table. Comment on convergence.

K	Call price-ask	Put price-ask
4.00	-	0.02
4.25	-	0.04
4.50	0.33	0.09
4.75	0.16	0.20
5.00	0.06	0.38
5.25	0.02	0.59
5.50	0.01	-
5.75	0.01	-

Solution:
Using the code from part (b) we find that the binomial implied volatilities for the observed call option prices are:

Call Strike K						
$N = 10^k$	4.50	4.75	5.00	5.25	5.50	5.75
k=1	0.1959958	0.1861423	0.1854591	0.1924051	0.2141313	0.2691982
k=2	0.1953581	0.1846621	0.1810657	0.1872241	0.2147462	0.2671258
k=3	0.1955441	0.1850530	0.1810638	0.1873190	0.2145108	0.2667234
k=4	0.1955327	0.1850340	0.1810126	0.1872696	0.2144425	0.2666703
k=5	0.1955346	0.1850397	0.1810164	0.1872659	0.2144425	0.2666551

Similarly, the binomial implied volatilities for the observed put prices are:

Put Strike K						
$N = 10^k$	4.00	4.25	4.50	4.75	5.00	5.25
k=1	0.3284631	0.2810405	0.2712118	0.2944049	0.3290666	0.3953078
k=2	0.3177290	0.2864645	0.2721759	0.2882598	0.3365477	0.3857713
k=3	0.3178505	0.2861039	0.2728306	0.2878195	0.3360429	0.3863084
k=4	0.3178429	0.2861570	0.2727888	0.2878214	0.3360239	0.3863691
k=5	0.3178353	0.2861532	0.2727926	0.2878271	0.3360315	0.3863767

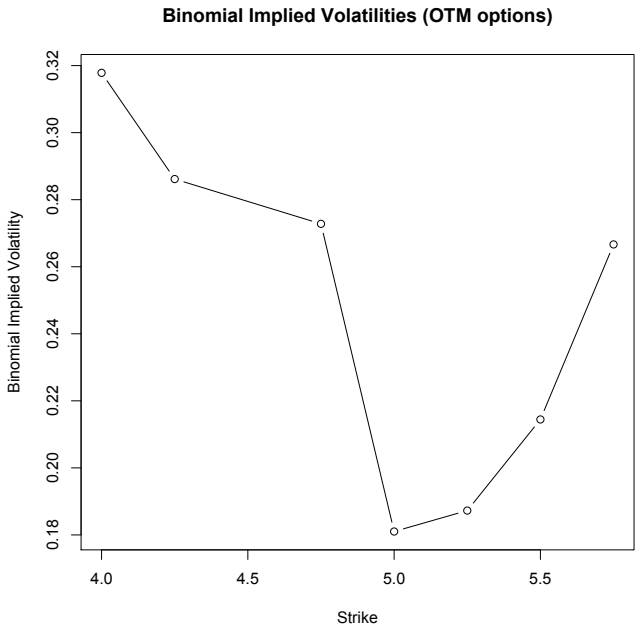
It is clear that as N increases the various values are converging, however, it would be better to have more intermediate values to better observe the convergence.

Note that the implied volatilities calculated using put prices and call prices with the same strike are not the same. This is a general feature of options price data.

(d) Implied volatilities should be the same, all else equal, whether we calibrate to market call or put prices. Practitioners calibrate to out-of-the-money option prices to estimate implied volatility. Plot your results from part (b) as a function of strike-price for the out-of-the-money options using largest value N for you were able to calculate. Is there a 'volatility smile'?

Solution:

We have the following plot of implied volatilities calculated for OTM observed option prices with $N = 10^5$



and there is a smile (but it is a bit crooked).