TP2 Risk Management

TP2: Hair Parra , Alessio Bressan, Ioan Catalin

2023-04-09

Libraries

Risk Management: European Options Portfolio

The objective is to implement (part of) the risk management framework for estimating the risk of a book of European call options by taking into account the risk drivers such as underlying and implied volatility.

Data

Load the database Market. Identify the price of the **SP500**, the **VIX index**, the term structure of interest rates (current and past), and the traded options (calls and puts).

```
# load dataset into environment
load(file = here("data_raw", "Market.rda"))

# reassign name and inspect structure of loaded data
mkt <- Market
summary(mkt)
```

```
##       Length Class  Mode
## sp500 3410   xts    numeric
## vix   3410   xts    numeric
## rf      14   -none- numeric
## calls 1266   -none- numeric
## puts  2250   -none- numeric
```

```
str(mkt)
```

```
## List of 5
##  $ sp500:An xts object on 2000-01-03 / 2013-09-10 containing:
##   Data:    double [3410, 1]
##   Index:   Date [3410] (TZ: "UTC")
##  $ vix  :An xts object on 2000-01-03 / 2013-09-10 containing:
##   Data:    double [3410, 1]
##   Index:   Date [3410] (TZ: "UTC")
##  $ rf   : num [1:14, 1] 0.00071 0.00098 0.00128 0.00224 0.00342 ...
##   ..- attr(*, "names")= chr [1:14] "0.00273972602739726" "0.0192307692307692" "0.0833333333333333" "0.25" .
##  $ calls: num [1:422, 1:3] 1280 1370 1380 1400 1415 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:3] "K" "tau" "IV"
##  $ puts : num [1:750, 1:3] 1000 1025 1050 1075 1100 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:3] "K" "tau" "IV"
```

Let's unpack these into the env. individually:

```
# unpack each of the elements in the mkt list
sp500 <- mkt$sp500
vix <- mkt$vix
Rf <- mkt$rf # risk-free rates
calls <- mkt$calls
puts <- mkt$puts

# assign colname for aesthetic
colnames(sp500) <- "sp500"
colnames(vix) <- "vix"
```

SP500 and VIX

By inspection, we observe that we the SP500 and VIX indices are contained in the `sp500` and `vix` xts objects respectively.

```
# show head of both indexes
head(sp500)
```

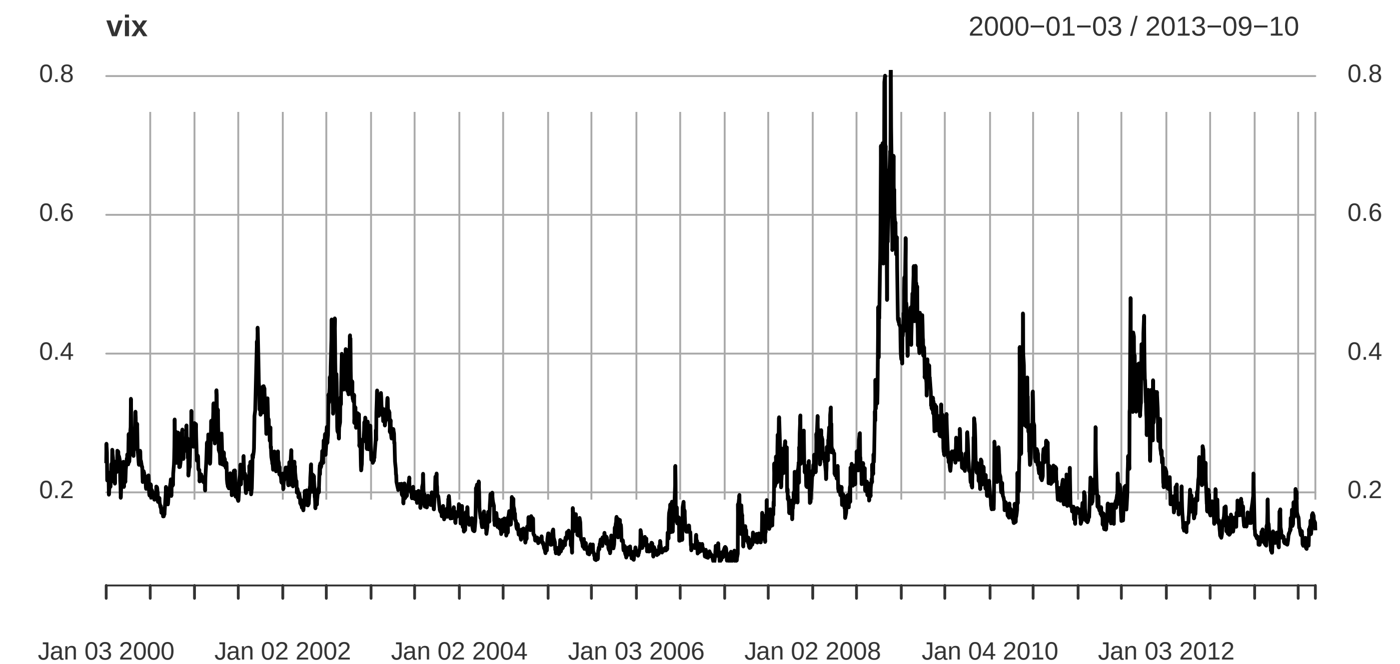
```
##              sp500
## 2000-01-03 1455.22
## 2000-01-04 1399.42
## 2000-01-05 1402.11
## 2000-01-06 1403.45
## 2000-01-07 1441.47
## 2000-01-10 1457.60
```

```
head(vix)
```

```
##               vix
## 2000-01-03 0.2421
## 2000-01-04 0.2701
## 2000-01-05 0.2641
## 2000-01-06 0.2573
## 2000-01-07 0.2172
## 2000-01-10 0.2171
```

```
par(mfrow = c(2,1))

# plot both series on top of each other
plot(sp500)
plot(vix)
```



Interest Rates

The **interest rates** are given in the `$rf` attribute. We can see that

```
Rf
```

```
##                [,1]
## [1,] 0.0007099993
## [2,] 0.0009799908
## [3,] 0.0012799317
## [4,] 0.0022393730
## [5,] 0.0034170792
## [6,] 0.0045123559
## [7,] 0.0043206525
```

```
##  [8,] 0.0064284968
##  [9,] 0.0090558654
## [10,] 0.0117237591
## [11,] 0.0141196498
## [12,] 0.0176131823
## [13,] 0.0207989304
## [14,] 0.0203526819
## attr(,"names")
##  [1] "0.00273972602739726" "0.0192307692307692"  "0.0833333333333333"
##  [4] "0.25"                "0.5"                 "0.75"
##  [7] "1"                   "2"                   "3"
## [10] "4"                   "5"                   "7"
## [13] "10"                  "30"
```

These represent the interest rates at different maturities. The maturities are given as follows:

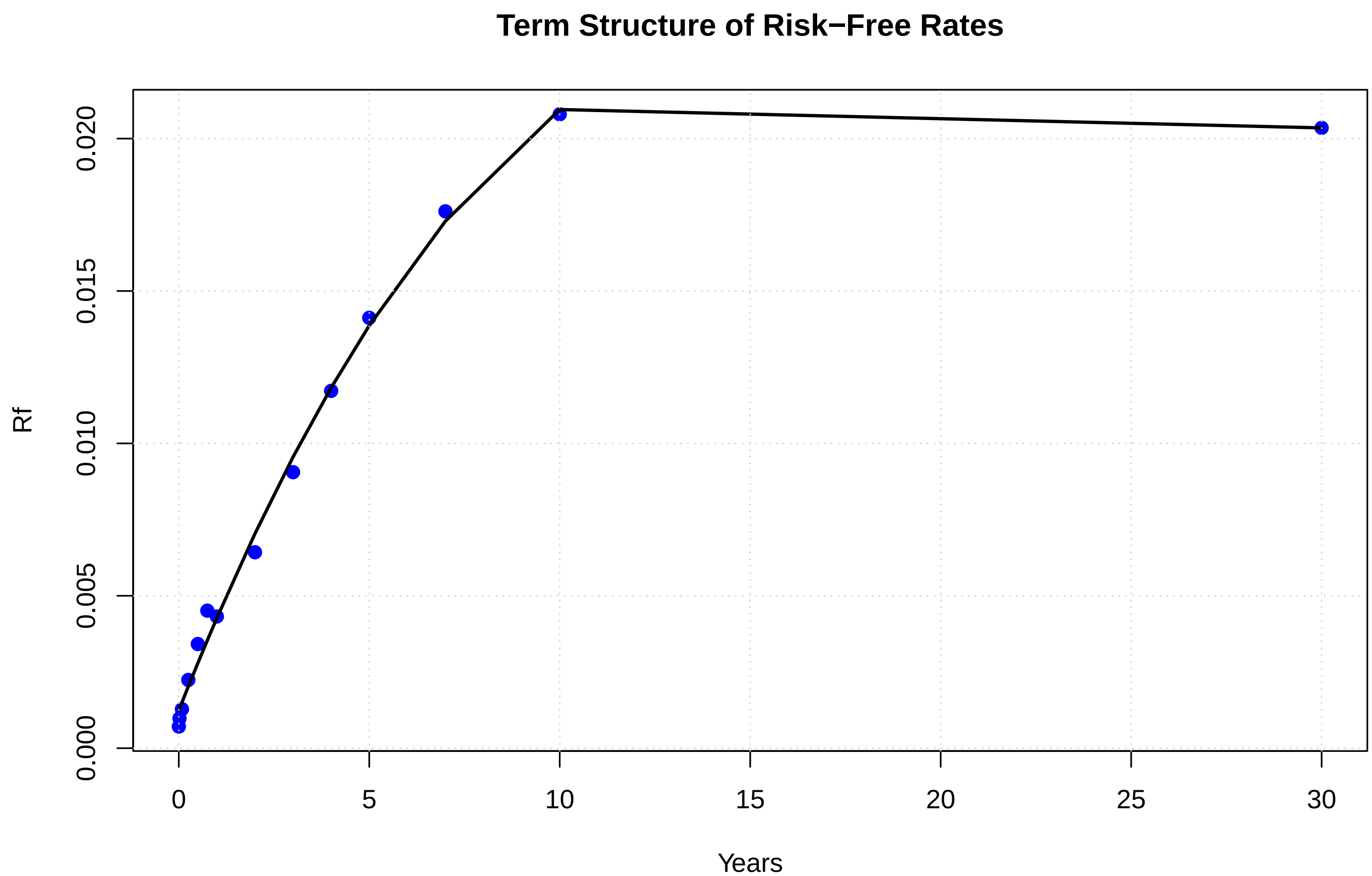
```r
r_f <- as.vector(Rf)
names(r_f) <- c("1d","1w", "1m", "3m", "6m", "9m", "1y", "2y", "3y", "4y", "5y", "7y","10y", "30y")
r_f
```

```
##           1d           1w           1m           3m           6m           9m
## 0.0007099993 0.0009799908 0.0012799317 0.0022393730 0.0034170792 0.0045123559
##           1y           2y           3y           4y           5y           7y
## 0.0043206525 0.0064284968 0.0090558654 0.0117237591 0.0141196498 0.0176131823
##          10y          30y
## 0.0207989304 0.0203526819
```

Further, we can pack different sources of information in a matrix:

```r
# pack Rf into a matrix with rf, years, and days
rf_mat <- as.matrix(r_f)
rf_mat <- cbind(rf_mat, as.numeric(names(Rf)))
rf_mat <- cbind(rf_mat, rf_mat[, 2]*360)
colnames(rf_mat) <- c("rf", "years", "days")
rf_mat
```

```
##               rf        years          days
## 1d  0.0007099993  0.002739726     0.9863014
## 1w  0.0009799908  0.019230769     6.9230769
## 1m  0.0012799317  0.083333333    30.0000000
## 3m  0.0022393730  0.250000000    90.0000000
## 6m  0.0034170792  0.500000000   180.0000000
## 9m  0.0045123559  0.750000000   270.0000000
## 1y  0.0043206525  1.000000000   360.0000000
## 2y  0.0064284968  2.000000000   720.0000000
## 3y  0.0090558654  3.000000000  1080.0000000
## 4y  0.0117237591  4.000000000  1440.0000000
## 5y  0.0141196498  5.000000000  1800.0000000
## 7y  0.0176131823  7.000000000  2520.0000000
## 10y 0.0207989304 10.000000000  3600.0000000
## 30y 0.0203526819 30.000000000 10800.0000000
```



Calls

The `calls` object displays the different values of K (**Strike Price**), τ (**time to maturity**) and $\sigma = IV$ (**Implied Volatilty**)

```
dim(calls)
```

```
## [1] 422   3
```

```
head(calls)
```

```
##         K        tau        IV
## [1,] 1280 0.02557005 0.7370370
## [2,] 1370 0.02557005 0.9691616
## [3,] 1380 0.02557005 0.9451401
## [4,] 1400 0.02557005 0.5274481
## [5,] 1415 0.02557005 0.5083375
## [6,] 1425 0.02557005 0.4820041
```

Add `days` column for convenience:

```
calls <- cbind(calls, calls[, "tau"]*250)
colnames(calls) <- c("K","tau", "IV", "tau_days")
head(calls)
```

```
##         K        tau        IV tau_days
## [1,] 1280 0.02557005 0.7370370 6.392513
## [2,] 1370 0.02557005 0.9691616 6.392513
## [3,] 1380 0.02557005 0.9451401 6.392513
```

```
## [4,] 1400 0.02557005 0.5274481 6.392513
## [5,] 1415 0.02557005 0.5083375 6.392513
## [6,] 1425 0.02557005 0.4820041 6.392513
```

```
tail(calls)
```

```
##            K      tau        IV tau_days
## [417,] 1925 2.269406 0.1605208 567.3514
## [418,] 1975 2.269406 0.1602093 567.3514
## [419,] 2000 2.269406 0.1559909 567.3514
## [420,] 2100 2.269406 0.1480259 567.3514
## [421,] 2500 2.269406 0.1441222 567.3514
## [422,] 3000 2.269406 0.1519319 567.3514
```

Puts

```
dim(puts)
```

```
## [1] 750   3
```

```
head(puts)
```

```
##          K        tau        IV
## [1,] 1000 0.02557005 1.0144250
## [2,] 1025 0.02557005 1.0083110
## [3,] 1050 0.02557005 0.9622093
## [4,] 1075 0.02557005 0.9170457
## [5,] 1100 0.02557005 0.8728757
## [6,] 1120 0.02557005 0.8381910
```

```
puts <- cbind(puts, puts[, "tau"]*250)
colnames(puts) <- c("K","tau", "IV", "tau_days")
head(puts)
```

```
##          K        tau        IV tau_days
## [1,] 1000 0.02557005 1.0144250 6.392513
## [2,] 1025 0.02557005 1.0083110 6.392513
## [3,] 1050 0.02557005 0.9622093 6.392513
## [4,] 1075 0.02557005 0.9170457 6.392513
## [5,] 1100 0.02557005 0.8728757 6.392513
## [6,] 1120 0.02557005 0.8381910 6.392513
```

```
tail(puts)
```

```
##            K      tau        IV tau_days
## [745,] 1750 2.269406 0.1899088 567.3514
## [746,] 1800 2.269406 0.1698365 567.3514
## [747,] 1825 2.269406 0.1986200 567.3514
## [748,] 1850 2.269406 0.1853406 567.3514
## [749,] 2000 2.269406 0.1520378 567.3514
## [750,] 3000 2.269406 0.2759397 567.3514
```

Pricing a Portfolio of Options

Black-Scholes

Notation:

- S_t = Current value of underlying asset price
- K = Options **strike price**
- T = Option **maturity** (in years)
- t = **time** in years
- $\tau = T - t$ = **Time to maturity**
- r = **Risk-free rate**
- y **Dividend yield**
- $R = r - y$
- σ = **Implied volatility**
- c = **Price Call Option**
- p = **Price Put Option**

Proposition 1 (Black-Scholes Model)**.** Assume the notation before, and let $N(\cdot)$ be the cumulative standard normal distribution function. Under certain assumptions, the Black-Scholes models prices Call and Put options as follows:

$$\begin{cases} C(S_t, t) = Se^{yT}N(d_1) - Ke^{-r\times\tau}N(d_2), \\ \\ P(S_t, t) = Ke^{-r\times\tau}(1 - N(d_2)) - Se^{y\times T}(1 - N(d_1)), \end{cases}$$

where:

$$\begin{cases} d_1 = \dfrac{\ln\left(\dfrac{S_t}{K}\right) + \tau\left(r + \dfrac{\sigma^2}{2}\right)}{\sigma\sqrt{\tau}} \\ d_2 = d_1 - \sigma\sqrt{\tau} \end{cases}$$

, further the Put Option price corresponds to the **Put-Call parity**, given by:

$$C(S_t, t) + Ke^{-r\times\tau} = P(S_t, t) + S_t$$

Note As here we don't have dividends, then $y = 0$, and so

$$\begin{cases} C(S_t, t) = S_tN(d_1) - Ke^{-r\times\tau}N(d_2), \\ \\ P(S_t, t) = Ke^{-r\times\tau}(1 - N(d_2)) - S_t(1 - N(d_1)), \end{cases}$$

Implementation

```
get_d1 <- function(S_t, K, tau, r, sigma){
  ### Compute d1 for the Black-Scholes model
  # INPUTS
  #   S_t:  Current value of underlying asset price
  #   K:    Strike Price
  #   tau:  T- t, where T=maturity, and t=current time
  #   r:    risk-free rate
  #   sigma   Implied volatility (i.e. sigma)

  num <- (log(S_t/K) - tau*(r + 0.5*sigma**2)) # numerator
  denom <- sigma * sqrt(tau) # denominator

  return(num/denom)
```

```
}

get_d2 <- function(d1, sigma, tau){
  ### Compute d2 for the Black-Scholes model
  # INPUTS
  #   d1:  d1 factor calculated by the get_d1 function
  #   tau:  T- t, where T=maturity, and t=current time
  #   sigma   Implied volatility (i.e. sigma)

  return(d1 - sigma * sqrt(tau))
}

# Function to implement the Black-Scholes model
black_scholes <- function(S_t, K, r, tau, sigma, put=FALSE){
  # Calculates a Call (or Option) price using Black-Scholes
  # INPUTS
  #   S_t:      [numeric] Current value of underlying asset price
  #   K:        [numeric] Strike Price
  #   r:        [numeric] risk-free rate
  #   tau:      [numeric] T- t, where T=maturity, and t=current time
  #   sigma:    [numeric] Implied volatility (i.e. sigma)
  #   put:      [logical] if TRUE, calculate a Put, if FALSE, calculate a Call.
  #             FALSE by default (Call).
  #
  # OUTPUTS:
  #   P or C: [numeric] Option value according to Black-scholes

  # calculate d1 & d2
  d1 <- get_d1(S_t, K, tau, r, sigma)
  d2 <- get_d2(d1, sigma, tau)

  if(put==TRUE){
    # calculate a Put option
    P <- K*exp(-r*tau)*(1 - pnorm(d2)) - S_t * (1 - pnorm(d1))
    P <- as.numeric(P)
    return( round(P,6))
  }
  # else calculate a Call option (default)
  C <- S_t * pnorm(d1) - K*exp(-r*tau) * pnorm(d2)
  return( round(as.numeric(C),6) )
}

# Test: Call Option
S_t = 1540
K = 1600
r = 0.03
tau = 10/360
sigma = 1.05
black_scholes(S_t, K, r, tau, sigma)
```

```
## [1] 80.81672
```

Book of Options

Assume the following book of **European Call Options:**

1. **1x** strike $K = 1600$ with maturity $T = 20d$
2. **1x** strike $K = 1605$ with maturity $T = 40d$

3. **1x** strike $K = 1800$ with maturity $T = 40d$

Find the price of this book given **the last underlying price** and the **last implied volatility** (take the VIX for all options). Use **Black-Scholes** to price the options. Take the current term structure and **linearly interpolate** to find the corresponding rates. Use 360 days/year for the term structure and **250 days/year** for the maturity of the options.

Nearest values

This function will obtain the two nearest values a, b for a number x in a vector v, such that $a < x < b$.

```
# Obtain the two nearest values of x in vec.
get_nearest<- function(x, vec){
  # find all the numbers that are bigger and smaller than x in vec
  bigger <- vec >= x
  smaller <- vec <= x

  # filter only values with TRUE
  bigger <- bigger[bigger == TRUE]
  smaller <- smaller[smaller == TRUE]

  # obtain the indexes for the left and upper bound
  a_idx <- length(smaller)
  b_idx <- length(smaller)+1

  # retrieve values from original vector
  a <- vec[a_idx]
  b <- vec[b_idx]

  # return the retrieved values
  return( c(a,b) )
}

# Test
days <- rf_mat[, "days"]
get_nearest(40, rf_mat[, "days"]) # nearest day values
```

```
## 1m 3m
## 30 90
```

Linear Interpolation

Given two known values (x_1, y_1) and (x_2, y_2), we can estimate the y-value for some x-value with:

$$y = y_1 + \frac{(x - x_1)(y_2 - y_1)}{(x_2 - x_1)}$$

```
# Function to interpolate y given two points
interpolate <- function(x,x1=1,y1=1,x2=2,y2=2){
  y1 + (x-x1)*(y2-y1)/(x2-x1)
}
```

Finding the rates through interpolation

The **yield curve** for the given structure of interest rates can be modeled a function $r_f = f(x)$,where x is the number of years. Then, we can interapolate the values as follows:

```
# Interest rates
rf_mat
```

```
##                rf       years        days
## 1d  0.0007099993  0.002739726     0.9863014
## 1w  0.0009799908  0.019230769     6.9230769
## 1m  0.0012799317  0.083333333    30.0000000
## 3m  0.0022393730  0.250000000    90.0000000
## 6m  0.0034170792  0.500000000   180.0000000
## 9m  0.0045123559  0.750000000   270.0000000
## 1y  0.0043206525  1.000000000   360.0000000
## 2y  0.0064284968  2.000000000   720.0000000
## 3y  0.0090558654  3.000000000  1080.0000000
## 4y  0.0117237591  4.000000000  1440.0000000
## 5y  0.0141196498  5.000000000  1800.0000000
## 7y  0.0176131823  7.000000000  2520.0000000
## 10y 0.0207989304 10.000000000  3600.0000000
## 30y 0.0203526819 30.000000000 10800.0000000
```

```
head(calls)
```

```
##          K        tau        IV tau_days
## [1,] 1280 0.02557005 0.7370370 6.392513
## [2,] 1370 0.02557005 0.9691616 6.392513
## [3,] 1380 0.02557005 0.9451401 6.392513
## [4,] 1400 0.02557005 0.5274481 6.392513
## [5,] 1415 0.02557005 0.5083375 6.392513
## [6,] 1425 0.02557005 0.4820041 6.392513
```

ex.: **1x** strike $K = 1600$ with maturity $T = 20d$

```
price_option <- function(T, K, calls, rf_mat, stock=NA, S_t=NA, IV = NA, put=FALSE){
  # Calculates the price of an European option using input parameters
  # INPUTS
  #   T:        [numeric] maturity of option (in days)
  #   K:        [numeric] Strike Price
  #   calls:    [matrix] matrix containing information about tau and IV for different strike prices
  #   rf_mat:       [matrix] matrix containing risk-free term structure
  #   stock:     [xts OR zoo like object] object containing stock prices for a single stock
  #   S_t:      [numeric] Specific price at time t
  #   IV:    [float] Implied volatility of the underlying
  #   put:     [logical] if TRUE, calculate a Put, if FALSE, calculate a Call.
  #            FALSE by default (Call).
  #
  # OUTPUTS:
  #   LIST containing:
  #     - P or C: [numeric] Option value according to Black-scholes and available information
  #     - r_interp: [numeric] Interpolated risk-free rate given risk-free term structure
  #     - calls [matrix] relevant set of calls information
  #     - rates [matrix] relevant set of risk-free rates used for the interpolation

  # Sanity check
  if(!is.matrix(calls) | !('tau_days' %in% colnames(calls)) ){
    stop("calls should be a matrix with columns c('K', 'tau', 'IV', 'tau_days')")
  }

  # Inputs
  tau = T/250 # days --> years
```

```
days_calls <- calls[,"tau_days"] # extract days column
days_rf <- rf_mat[, "days"] # extract days from rf_mat

# extract the calls values
ab <- get_nearest(T, days_calls) # search lower and upper nearest days to T
valid_days <- calls[, "tau_days"] == ab[1] | calls[, "tau_days"] == ab[2] # where match
calls_sub <- calls[ valid_days,  ] # subset valid rows
calls_sub <- calls_sub[calls_sub[,"K"]==K, ] # subset matching K

# test whether matrix is empty (i.e. no matching K found)
if(all(is.na(calls_sub))){
  warning("No values matching K in Calls data\n")
}

# extract interpolated risk rates
ab <- get_nearest(T, days_rf) # obtain nearest days to T available in rf_mat
valid_days_rf <- rf_mat[, "days"] == ab[1] | rf_mat[, "days"] == ab[2] # where match
rates <- rf_mat[valid_days_rf, ] # subset for valid days

# interpolate risk free rate for Option given maturity
r <- interpolate(tau,
                 x1=rates[1,2],
                 y1=rates[1,1],
                 x2=rates[2,2],
                 y2=rates[2,1])

# use provided sigma by default, else calculate from calls matrix
if(is.na(sigma)){

  # retrieve implied volatility for option
  if(is.matrix(calls_sub)){
    # average between lower and upper values
    sigma <- (calls_sub[1, "IV"] + calls_sub[2, "IV"])/2

  } else{
    # retrive from numeric vector (single match)
    sigma <- calls_sub["IV"]
  }

}
else{
  # rename for convenience
  sigma <- IV
}

# if price at t is not provided
if(is.na(S_t) & !is.na(stock)){
  # retrieve last price for option from input index
  warning("Using last day's S_t from input index\n")
  S_t <- as.numeric( stock[length(stock)])
}

# Calculate Option price
if(put==TRUE){
  C <- NA
  P <- black_scholes(S_t, K, r, tau, sigma, put=TRUE)
}
else{
  C <- black_scholes(S_t, K, r, tau, sigma, put=FALSE)
```

```
    P <- NA
  }

  # pack everything into a List and return
  return(list(Call = C,
              Put = P,
              S = as.numeric(S_t)[[1]],
              K = K,
              r_interp = r,
              calls = calls_sub, # subset of calls used
              rates = rates # subset of rates used
              ))
}
```

```
S_t = sp500[length(sp500)] # last price of underlying
IV = vix[length(vix)] # last volatility

## test: specific price
price_option(T=20, K=1600, calls = calls, rf_mat =  rf_mat, stock = NA, S_t = S_t, IV = IV)
```

```
## $Call
## [1] 87.56885
##
## $Put
## [1] NA
##
## $S
## [1] 1683.99
##
## $K
## [1] 1600
##
## $r_interp
## [1] 0.001264335
##
## $calls
##         K        tau        IV  tau_days
## [1,] 1600 0.02557005 0.1817481  6.392513
## [2,] 1600 0.10228238 0.1701946 25.570595
##
## $rates
##              rf      years      days
## 1w 0.0009799908 0.01923077  6.923077
## 1m 0.0012799317 0.08333333 30.000000
```

Next, using the function above we price the book of options given:

1. **1x** strike $K = 1600$ with maturity $T = 20d$
2. **1x** strike $K = 1605$ with maturity $T = 40d$
3. **1x** strike $K = 1800$ with maturity $T = 40d$

First, we retrieve the latest value for the underlying (SP500) and the latest implied volatility (VIX):

```
S_t = sp500[length(sp500)] # last price of underlying
IV = vix[length(vix)] # last volatility
```

Then, we price the options accordingly:

```r
# First Call Option
price_option(T=20, K=1600, calls=calls, rf_mat=rf_mat, S_t = S_t, IV = IV)
```

```
## $Call
## [1] 87.56885
##
## $Put
## [1] NA
##
## $S
## [1] 1683.99
##
## $K
## [1] 1600
##
## $r_interp
## [1] 0.001264335
##
## $calls
##         K        tau        IV  tau_days
## [1,] 1600 0.02557005 0.1817481  6.392513
## [2,] 1600 0.10228238 0.1701946 25.570595
##
## $rates
##              rf      years      days
## 1w 0.0009799908 0.01923077  6.923077
## 1m 0.0012799317 0.08333333 30.000000
```

```r
# Second Call Option
price_option(T=40, K=1605, calls=calls, rf_mat=rf_mat, S_t = S_t, IV = IV)
```

```
## $Call
## [1] 90.22871
##
## $Put
## [1] NA
##
## $S
## [1] 1683.99
##
## $K
## [1] 1605
##
## $r_interp
## [1] 0.001721275
##
## $calls
##            K          tau           IV     tau_days
## 1605.0000000    0.1022824    0.1676923   25.5705949
##
## $rates
##             rf      years days
## 1m 0.001279932 0.08333333   30
## 3m 0.002239373 0.25000000   90
```

```r
# Third Call Option
price_option(T=40, K=1800, calls=calls, rf_mat=rf_mat, S_t = S_t, IV = IV)
```

```
## $Call
## [1] 6.34395
##
## $Put
## [1] NA
##
## $S
## [1] 1683.99
##
## $K
## [1] 1800
##
## $r_interp
## [1] 0.001721275
##
## $calls
##         K       tau        IV tau_days
## [1,] 1800 0.1022824 0.1057523 25.57059
## [2,] 1800 0.1789947 0.1044115 44.74868
##
## $rates
##             rf      years days
## 1m 0.001279932 0.08333333   30
## 3m 0.002239373 0.25000000   90
```

Two risk drivers and copula-marginal model (Student-t and Gaussian Copula)

1. Compute the daily log-returns of the underlying stock
2. Assume the first invariant is generated using a Student-t distribution with $\nu = 10$ df and the second invariant is generated using a Student-t distribution with $\nu = 5$ df.
3. Assume the **normal copula** to merge the marginals.
4. Generate 10000 scenarios for the one-week ahead price for the underlying and the one-week ahead VIX value using the copula.
5. Determine the P&L distribution of the book of options, using the simulated values.
6. Take interpolated rates for the term structure.

Gaussian Copula with two Student-t marginals

A bivariate distribution H can be formed via a copula C from two marginal distributions with CDFs F and G via:

$$H(x, y) = C(F(x), G(y)) = C(F^{-1}(u), G^{-1}(u))$$

with density

$$h(x, y) = c(F(x), G(y))f(x)g(y)$$

The **Gaussian Copula** is given by:

$$C_\rho^{\text{Gauss}}(u, v) = \Phi_\rho(\Phi^{-1}(u), \Phi^{-1}(v)).$$

In this case, a Gaussian copula with two Student-t marginals with CDFs $t(\nu_1)$ with ν_1 degrees of freedom and $t(\nu_2)$ with ν_2 degrees of freedom is given by:

$$C_\rho^{\text{Gauss}}(u, v) = \Phi_\rho(F_{\nu_1}^{-1}(u), F_{\nu_1}^{-1}(v)),$$

where F_{ν_1} and F_{ν_2} are their respective CDFs.

Log-returns

The **discrete returns** are given by:

$$R_{t+1} = \frac{P_{t+1} - P_t}{P_t}$$

and the next ahead log-returns are given by:

$$\log(R_{t+1}) = \log(P_{t+1} - P_t) - \log(P_t)$$

```
# load reqruired libraries
library("PerformanceAnalytics")

# calculate returns
sp500_rets <- PerformanceAnalytics::CalculateReturns(sp500, method="log")
vix_rets <- PerformanceAnalytics::CalculateReturns(vix, method="log")

# remove nas
sp500_rets <- sp500_rets[rowSums(is.na(sp500_rets)) == 0,]
vix_rets <- vix_rets[rowSums(is.na(vix_rets)) == 0,]

# display
head(sp500_rets)
```

```
##                     sp500
## 2000-01-04 -0.0390992269
## 2000-01-05  0.0019203798
## 2000-01-06  0.0009552461
## 2000-01-07  0.0267299353
## 2000-01-10  0.0111278213
## 2000-01-11 -0.0131486343
```

```
head(vix_rets)
```

```
##                       vix
## 2000-01-04  0.1094413969
## 2000-01-05 -0.0224644415
## 2000-01-06 -0.0260851000
## 2000-01-07 -0.1694241312
## 2000-01-10 -0.0004605112
## 2000-01-11  0.0357423253
```

Generating the simulation scenarios

Assumptions: - Marginal Student-t distributions - Disregard time dependence in the bootstrapping process

```
# Load required libraries
library("fGarch")
library("MASS")
library("Matrix")

# random seed for replication
set.seed(69)

##############################
### Setup & Initialization ###
```

```r
##############################

# Simulation parameters
n_sim = 10 # set number of simulations
n_ahead = 5 # days aheade to produce samples

# vector version since ignoring dates and using full past data
sp500_rets_vec <- as.vector(sp500_rets)
vix_rets_vec <- as.vector(vix_rets)

# preallocate matrices to store simulations
sim_rets_sp500 <- matrix(NA, nrow = n_sim, ncol=5)
sim_rets_vix <- matrix(NA, nrow = n_sim, ncol=5)

# assign days ahead
colnames(sim_rets_sp500) <- c("T+1", "T+2", "T+3", "T+4", "T+5")
colnames(sim_rets_vix) <- c("T+1", "T+2", "T+3", "T+4", "T+5")

#########################
### Fitting the model ###
#########################

# straight up fit on the data
b_sp500 <- sp500_rets_vec
b_vix <- vix_rets_vec

##  Fit a Gaussian Copula to model the dependence

# calculate the mean vector
mu <- c(mean(b_sp500), mean(b_vix))

# calculate the covariance
r <- cor(b_sp500, b_vix)[[1]] # correlation coefficient
sig <- c(sd(b_sp500), sd(b_vix)) # standard deviation
R <- matrix(data = c(1, r, r, 1), # correlation matrix
            nrow = 2,
            ncol = 2,
            byrow = TRUE)
Sigma <- diag(sig) %*% R %*% diag(sig) # covariance matrix
Sigma <- (Sigma + t(Sigma)) / 2
Sigma <- as.matrix(nearPD(Sigma)$mat)

##############################
### Running the simulation ###
##############################

# perform simulations
for(b in 1:n_sim){

  # # Obtain the bootstrapped samples from the data (?) >-- not sure if this is right?
  # b_sp500 <- sample(sp500_vec, size=length(sp500_vec), replace=TRUE)
  # b_vix <- sample(vix_vec, size=length(vix_vec), replace=TRUE)


  # Sample 5-days ahead from Gaussian Copula
  Z <- mvrnorm(n = n_ahead, mu = mu, Sigma = Sigma)

  # Draws from Gaussian Copula
  U1 <- pnorm(q = Z[, 1], mu[1], sig[1]) # first dimension (sp500)
```

```
  U2 <- pnorm(q = Z[, 2], mu[2], sig[2]) # second dimension  (vix)

  # Model marginals with student-t distributions & sample
  X1 <- qt(U1, df = 10) # simulated sp500
  X2 <- qt(U2, df = 5) # simulated vix

  # store simulation of log return in matrix
  sim_rets_sp500[b, ] <- X1
  sim_rets_vix[b ,] <- X2
}

# preview of simulated log returns
head(sim_rets_sp500)
```

```
##              T+1         T+2        T+3        T+4        T+5
## [1,] -0.85743562 -1.29266432  0.8733347  1.6045281  1.9194978
## [2,] -0.16262580  0.11739878  1.1240556 -0.6496687  1.2460947
## [3,] -2.18705327 -0.41753778 -0.4061295  1.8478907 -0.3060624
## [4,]  0.01400346  0.59591784  2.9462629  0.4690210 -0.8577066
## [5,] -0.76526470 -0.29864245 -0.9047900 -0.3773647  0.0248524
## [6,] -0.24827401  0.05753925 -0.8938464  0.7136188  1.1808194
```

```
head(sim_rets_vix)
```

```
##              T+1         T+2        T+3        T+4        T+5
## [1,]  0.05413839  0.36289352 -0.3331040 -1.0183794 -0.9962522
## [2,] -0.18827124  0.38595197 -1.1680597  0.4289229 -1.5088114
## [3,]  3.79647142 -0.06433122 -0.8837084 -1.4666625 -0.2640861
## [4,]  0.11565911 -2.62227260 -2.9784591 -0.2298235  1.3921182
## [5,]  0.49295453  0.27185842  2.9260609  0.6392685  0.3061394
## [6,]  0.87484203  0.48042192  0.3444034 -0.1963256 -2.6288166
```

Computing Prices from Returns

Next, we crate a function to forecast the 5 day ahead prices from the returns. Since:

$$
\begin{aligned}
\log(R_{t+1}) &= \log(P_{t+1} - P_t) - \log(P_t) \\
\implies \log(P_{t+1} - P_t) &= \log(R_{t+1}) + \log(P_t) \\
\implies P_{t+1} - P_t &= \exp\left(\log(R_{t+1}) + \log(P_t)\right) \\
\implies P_{t+1} &= \exp\left(\log(R_{t+1}) + \log(P_t)\right) + P_t \\
\implies P_t &= \exp\left(\log(R_t) + \log(P_{t-1})\right) + P_{t-1}
\end{aligned}
$$

```
# parameters
Pt <- sp500[length(sp500)][[1]] # initial price P_t
log_Pt <- log(Pt) # llog of current prices log(P_{t})
log_Rt_next <- sim_rets_sp500[, 1] # next set of returns log(R_{t+1})
Pt_next <- exp(log_Rt_next + log_Pt) + Pt

# display
Pt
```

```
## [1] 1683.99
```

```
log_Pt
```

```
## [1] 7.428921
```

```
log_Rt_next
```

```
##  [1] -0.85743562 -0.16262580 -2.18705327  0.01400346 -0.76526470 -0.24827401
##  [7] -0.68637014  1.18584191  0.69685820  0.02841854
```

```
log_Rt_next + log_Pt
```

```
##  [1] 6.571486 7.266295 5.241868 7.442925 6.663657 7.180647 6.742551 8.614763
##  [9] 8.125779 7.457340
```

```
exp(log_Rt_next + log_Pt)
```

```
##  [1]  714.4304 1431.2385  189.0229 1707.7376  783.4103 1313.7583  847.7206
##  [8] 5512.4429 3380.5019 1732.5330
```

```
Pt_next
```

```
##  [1] 2398.420 3115.229 1873.013 3391.728 2467.400 2997.748 2531.711 7196.433
##  [9] 5064.492 3416.523
```

```
f_next_Pt <- function(Pt, log_Rt_next){
  #### Wrapper for price_option for two vectors of prices and volatilities
  #
  # INPUTS
  #   Pt:           [numeric] Price of underlying at time t  (aka P_{t})
  #   log_Rt_next:  [vector numeric] vector of simulated log returns at time t+1 (aka log(R_{t+1}) )
  #
  # OUTPUTS:
  #   opt prices: [numeric vector] vector of prices computed from current underlying price Pt and
  #               the log returns for the next day ahead R_{t+1}, via the formula:
  #                P_{t+1} = exp( log(R_{t+1}) +  log(P_{t}) ) _ P_{t}

  # compute the log of Pt
  log_Pt <- log(Pt)

  # P_{t+1} = exp( log(R_{t+1}) +  log(P_{t}) ) _ P_{t}
  Pt_next <- exp(log_Rt_next + log_Pt) + Pt

  return(Pt_next)
}
```

```
# Obtain Initial values (last value of simulation)
spT <- sp500[length(sp500)][[1]]
vixT <- vix[length(vix)][[1]]

# Initialize empty matrices for the simulated sp500 and vix values
sim_val_mats <- initialize_sim_mats(sim_rets_sp500,
                                       lnames = c("sp500", "vix"), # <- this function comes from Utils.R
                                       num_mats = 2
                                       )

# Initialize the first prices
sim_val_mats$sp500[, 1] <- f_next_Pt(spT, sim_rets_sp500[, 1])
sim_val_mats$vix[, 1] <- f_next_Pt(spT, sim_rets_vix[, 1])

# for each day ahead
```

```r
for(t in 2:n_ahead){
  # obtain the values for P_{t-1}
  Pt_prev_sp500 <- sim_val_mats$sp500[, t-1]
  Pt_prev_vix <- sim_val_mats$vix[, t-1]

  # extract current returns R_{t}
  Rt_sp500 <- sim_rets_sp500[, t]
  Rt_vix <- sim_rets_vix[, t]

  # compute and assign next price ahead using current returns
  sim_val_mats$sp500[, t] <- f_next_Pt(Pt_prev_sp500, Rt_sp500)
  sim_val_mats$vix[, t] <- f_next_Pt(Pt_prev_vix, Rt_vix)
}

# unpack matrices
sim_price_sp500 <- sim_val_mats$sp500
sim_vol_vix <- sim_val_mats$vix
```

```r
# compare simulated returns with the price
head(sim_rets_sp500)
```

```
##              T+1         T+2        T+3        T+4        T+5
## [1,] -0.85743562 -1.29266432  0.8733347  1.6045281  1.9194978
## [2,] -0.16262580  0.11739878  1.1240556 -0.6496687  1.2460947
## [3,] -2.18705327 -0.41753778 -0.4061295  1.8478907 -0.3060624
## [4,]  0.01400346  0.59591784  2.9462629  0.4690210 -0.8577066
## [5,] -0.76526470 -0.29864245 -0.9047900 -0.3773647  0.0248524
## [6,] -0.24827401  0.05753925 -0.8938464  0.7136188  1.1808194
```

```r
head(sim_price_sp500)
```

```
##        T+1      T+2        T+3       T+4       T+5
## 1 2398.420 3056.879  10377.748  62012.35 484783.66
## 2 3115.229 6618.514  26985.730  41078.18 183896.30
## 3 1873.013 3106.704   5176.464  38028.47  66030.39
## 4 3391.728 9546.681 191264.768 496987.85 707777.13
## 5 2467.400 4297.779   6036.775  10175.99  20608.04
## 6 2997.748 6173.044   8698.308  26454.73 112618.90
```

```r
# compare simualted log rets with volatility
head(sim_rets_vix)
```

```
##              T+1         T+2        T+3        T+4        T+5
## [1,]  0.05413839  0.36289352 -0.3331040 -1.0183794 -0.9962522
## [2,] -0.18827124  0.38595197 -1.1680597  0.4289229 -1.5088114
## [3,]  3.79647142 -0.06433122 -0.8837084 -1.4666625 -0.2640861
## [4,]  0.11565911 -2.62227260 -2.9784591 -0.2298235  1.3921182
## [5,]  0.49295453  0.27185842  2.9260609  0.6392685  0.3061394
## [6,]  0.87484203  0.48042192  0.3444034 -0.1963256 -2.6288166
```

```r
head(sim_vol_vix)
```

```
##         T+1        T+2        T+3       T+4        T+5
## 1  3461.662   8437.740  14485.032  19716.73   26997.35
## 2  3078.990   7608.229   9974.157  25290.50   30884.07
## 3 76695.187 148611.833 210025.312 258476.92  456963.28
## 4  3574.459   3834.099   4029.144   7231.00   36323.94
## 5  4440.928  10269.206 201831.041 584319.85 1377926.52
## 6  5723.034  14975.786  36108.832  65781.07   70528.09
```

Pricing the simulation scenarios

Recall the initial (call) options:

1. **1x** strike $K = 1600$ with maturity $T = 20d$
2. **1x** strike $K = 1605$ with maturity $T = 40d$
3. **1x** strike $K = 1800$ with maturity $T = 40d$

Helper Functions

First, we code a function that will compute the option price, and an auxiliary function to create empty matrices of corresponding sizes:

```
prc_opt <- function(T, K, calls, rf_mat, price_vec, vol_vec){
  #### Wrapper for price_option for two vectors of prices and volatilities
  #
  # INPUTS
  #   T:            [numeric] time to maturity
  #   calls:        [matrix] matrix containing information about tau and IV for different strike prices
  #   rf_mat:       [matrix] matrix containing risk-free term structure
  #   price_vec:    [numeric vector] vector of stock (sp500) prices
  #   vol_vec:      [numeric vector] vector of corresponding volatilities
  #
  # OUTPUTS:
  #   opt prices: [numeric vector] vector of option prices

  # abstract price_opt function two arguments: S_t and IV
  price_opt_abstr <- function(x,y){price_option(T=T, # maturity
                                                K=K, # strike
                                                calls, # calls matrix
                                                rf_mat, # matrix of risk free structure
                                                stock = NA, # ignore
                                                S_t = x, # specific price
                                                IV = y)$Call} # implied volatility + extract Call price

  # pack both vectors into a dataframe
  vec_df <- data.frame(price_vec, vol_vec)

  # Calculate the options for all input S_t  and corresponding volatilities
  opt_prices <- mapply(price_opt_abstr, vec_df$price_vec, vec_df$vol_vec)

  return(opt_prices)
}
```

Option Pricing of Simulated Values

Next, we calculate the price of the book of options for the simulated values.

```
# random seed for replication
set.seed(123)

# Initialize empty matrices to store the simulated option prices (aka premiums)
opt_price_mats <- initialize_sim_mats(sim_price_sp500,
                                      lnames = c("opt1", "opt2", "opt3"),
                                      num_mats = 3
                                      )

#  maturities for each of the options
T1 <- 20
T2 <- 40
```

```
T3 <- 40

# Strikes for the options
K1 <- 1600
K2 <- 1605
K3 <- 1800

# looop through simulated prices (n_ahead days)
for(t in 1:ncol(sim_price_sp500)){

  # extract simulated prices for sp500 at T+t
  prices_t <- sim_price_sp500[, t]

  # extract implied volatility from vix at T+t
  vols_t <- sim_vol_vix[, t]

  # price first Call option
  c1_vec <- prc_opt(T1-t, K1, calls, rf_mat, prices_t, vols_t)
  opt_price_mats$opt1[ ,t] <- c1_vec
  # print(cbind(prices_t, vols_t, c1_vec)) # <-- uncomment for debugging

  # price first Call option
  c2_vec <- prc_opt(T2-t, K2, calls, rf_mat, prices_t, vols_t)
  opt_price_mats$opt2[ ,t] <- c2_vec

  # price first Call option
  c3_vec <- prc_opt(T3-t, K3, calls, rf_mat, prices_t, vols_t)
  opt_price_mats$opt3[ ,t] <- c3_vec
}
```

```
# overview of dataframes
head(opt_price_mats$opt1)
```

```
##   T+1 T+2 T+3 T+4 T+5
## 1   0   0   0   0   0
## 2   0   0   0   0   0
## 3   0   0   0   0   0
## 4   0   0   0   0   0
## 5   0   0   0   0   0
## 6   0   0   0   0   0
```

```
head(opt_price_mats$opt2)
```

```
##   T+1 T+2 T+3 T+4 T+5
## 1   0   0   0   0   0
## 2   0   0   0   0   0
## 3   0   0   0   0   0
## 4   0   0   0   0   0
## 5   0   0   0   0   0
## 6   0   0   0   0   0
```

```
head(opt_price_mats$opt3)
```

```
##   T+1 T+2 T+3 T+4 T+5
## 1   0   0   0   0   0
## 2   0   0   0   0   0
## 3   0   0   0   0   0
```

```
## 4    0    0    0    0    0
## 5    0    0    0    0    0
## 6    0    0    0    0    0
```

Distribution of the Profit and Loss for the Book Of Options

Recall the profit functions for European options:

Parameters

Parameters: - S: Spot price (current) - S_0: Spot price at the beginnin of the option - S_T: Spot price at maturity - T: Maturity of option - K: Strike price - c: Price of Call option - p: Price of Put option

Profit at Maturity

The profit functions of a long call and a long put are given by:

$$
\begin{aligned}
\pi^{\text{Long Call}} &= \max(S_T - K, 0) - c \\
\pi^{\text{Long Put}} &= \max(K - S_T, 0) - p
\end{aligned}
$$

Option profit function

```r
option_profit <- function(S,K,c=NULL, p=NULL, short=FALSE, N=1){
  #### Calculates Call and or Put profits
  #
  # INPUTS
  #   S:          [numeric or vector] array of prices to use
  #   K:          [numeric] Strike price for the option
  #   c:          [numeric or vector] array of premiums for a Call option
  #   sim_mat:    [matrix] (n_sim x n_days_ahead) matrix of simulation prices for
  #               n_days ahead, with n_sim simulations.
  #   lnames:     [character vector] vector with names for each of the created matrices
  #   num_mats:   [numeric] number of matrices to create
  #
  # OUTPUT:
  #   mats:       [list of matrices] List containing three matrices of compatible sizes as sim_mat
  #               initialized to NA  values

  # Initialize empty profit values
  profits <- list(call_profit=NA, put_profit=NA)
  call_profit = NA
  put_profit = NA

  # sanity check
  if(is.null(c) & is.null(p)){
    stop("At least one of c or p must be provided")
  }

  # if c, calculate the Call profit
  if(!is.null(c)){
    profits$call_profit <- (max(S - K, 0) - c)*N
  }

  # if p, calculate the Put profit
  if(!is.null(p)){
    profits$put_profit <- (max(K - S, 0) - p)*N
  }
```

```
  # inverse profit if short position
  if(short){
    profits <- lapply(profits, function(x){-x})
  }

  # multiply by size
  return(profits)
}
```

Calculating the profits

For each of the simulated prices and resulting premiums, we want to calculate the profit generated at each simulation timestep:

```
# Matrices of profit and loss for each of the options simulations
PL_mats <- initialize_sim_mats(sim_price_sp500,
                               lnames = c("PL1", "PL2", "PL3"),
                               num_mats = 3
                               )

# Calculate profit for all simulated options at each day ahead
for(t in 1:n_ahead){

  #spot price of underlying at day T+t
  spot <- sim_price_sp500[, t]

  # Option profit for K1 at time T+t with premiums c1
  c1 <- opt_price_mats$opt1[, t] # extract the premiums
  PL_mats$PL1[,t] <- option_profit(S=spot, K=K1, c=c1)$call_profit

  print(head(c1))
  print(head(spot))

  break

  # # Option profit for K1 at time T+t with premiums c2
  # c2 <- opt_price_mats$opt2[, t] # extract the premiums
  # PL_mats$PL2[,t] <- option_profit(S=spot, K=K2, c=c2)$call_profit
  #
  # # Option profit for K1 at time T+t with premiums c1
  # c3 <- opt_price_mats$opt3[, t] # extract the premiums
  # PL_mats$PL3[,t] <- option_profit(S=spot, K=K3, c=c3)$call_profit
}
```

```
## 1 2 3 4 5 6
## 0 0 0 0 0 0
##        1        2        3        4        5        6
## 2398.420 3115.229 1873.013 3391.728 2467.400 2997.748
```

```
# display profit matrices
head(PL_mats$PL1)
```

```
##        T+1 T+2 T+3 T+4 T+5
## 1 5596.433  NA  NA  NA  NA
## 2 5596.433  NA  NA  NA  NA
## 3 5596.433  NA  NA  NA  NA
## 4 5596.433  NA  NA  NA  NA
## 5 5596.433  NA  NA  NA  NA
## 6 5596.433  NA  NA  NA  NA
```

```
head(PL_mats$PL2)
```

```
##   T+1 T+2 T+3 T+4 T+5
## 1  NA  NA  NA  NA  NA
## 2  NA  NA  NA  NA  NA
## 3  NA  NA  NA  NA  NA
## 4  NA  NA  NA  NA  NA
## 5  NA  NA  NA  NA  NA
## 6  NA  NA  NA  NA  NA
```

```
head(PL_mats$PL3)
```

```
##   T+1 T+2 T+3 T+4 T+5
## 1  NA  NA  NA  NA  NA
## 2  NA  NA  NA  NA  NA
## 3  NA  NA  NA  NA  NA
## 4  NA  NA  NA  NA  NA
## 5  NA  NA  NA  NA  NA
## 6  NA  NA  NA  NA  NA
```

Distribution of Options P/L

Next, using all the simulated profits and losses for each of the options, we display a histogram for the distribution for each of the options, for the aggregated 5 days of simulation:

```
# flatten the matrices 5-days ahead simulated P/L for the three options
sim_pl_opt1 <- as.vector(PL_mats$PL1)
sim_pl_opt2 <- as.vector(PL_mats$PL2)
sim_pl_opt3 <- as.vector(PL_mats$PL3)

# plot the distribution for each of the options
par(mfrow = c(3,1))
# hist(sim_pl_opt1, nclass = round(10 * log(n_sim)), probability = TRUE)
# hist(sim_pl_opt2, nclass = round(10 * log(n_sim)), probability = TRUE)
# hist(sim_pl_opt3, nclass = round(10 * log(n_sim)), probability = TRUE)
```