TP2 Risk Management

TP2: Hair Parra , Alessio Bressan, Ioan Catalin

2023-04-07

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
```

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:

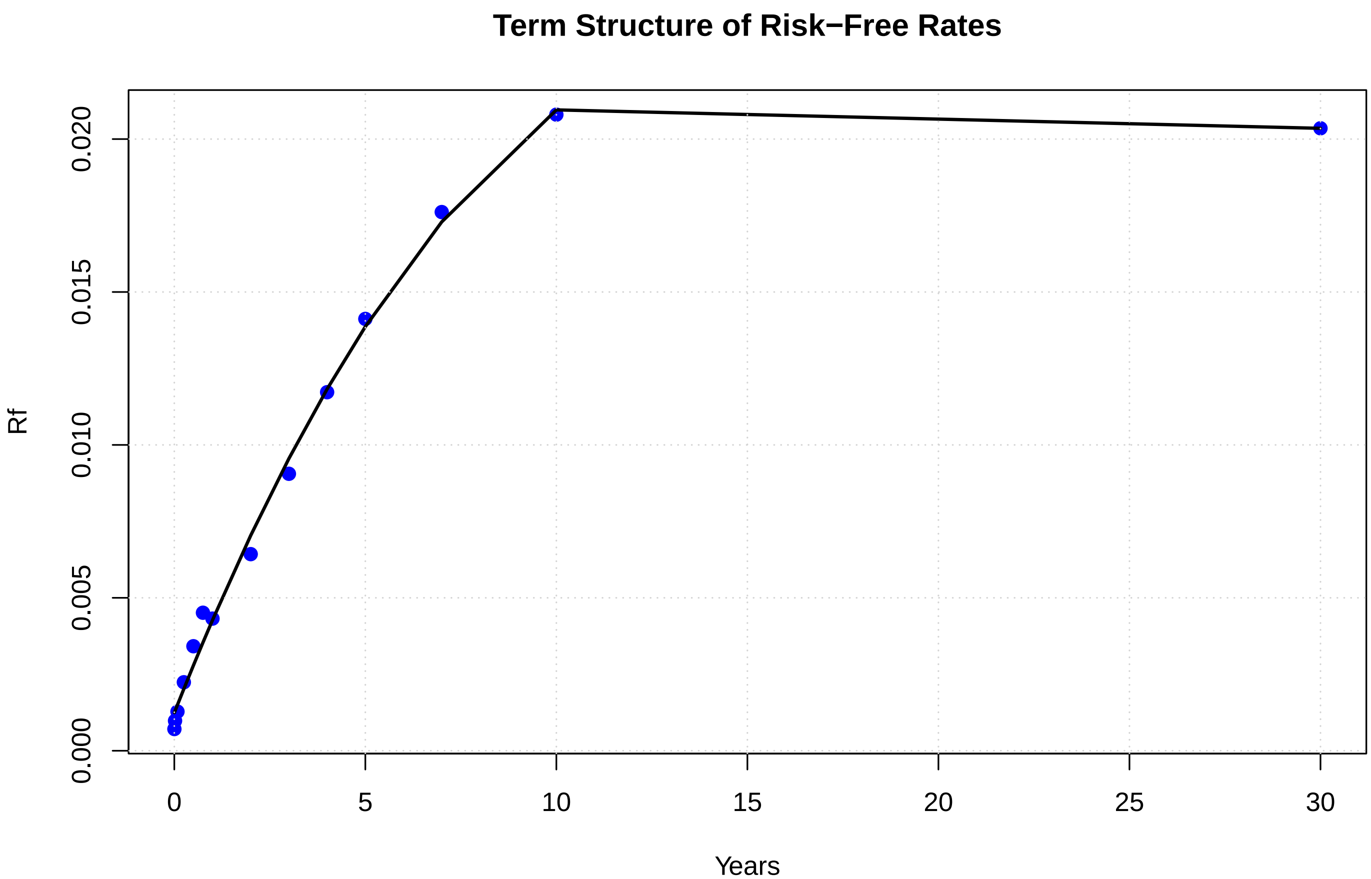
```
r_f <- as.vector(Rf)
names(r_f) <- c("1d","1w", "1m", "2m", "3m", "6m", "9m", "1y", "3y", "4y", "5y", "7y","10y", "30y")
r_f
```

```
##           1d           1w           1m           2m           3m           6m
## 0.0007099993 0.0009799908 0.0012799317 0.0022393730 0.0034170792 0.0045123559
##           9m           1y           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:

```
# 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
## 2m  0.0022393730  0.250000000    90.0000000
## 3m  0.0034170792  0.500000000   180.0000000
## 6m  0.0045123559  0.750000000   270.0000000
## 9m  0.0043206525  1.000000000   360.0000000
## 1y  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**)

```r
dim(calls)
```

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

```r
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:

```r
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)
```

```r
}

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 <- S_t * pnorm(d1) - K*exp(r*tau) * pnorm(d2)
    P <- as.numeric(P)
    return(P)
  }
  # else calculate a Call option (default)
  C <- K*exp(r*tau)*(1 - pnorm(d2)) - S_t * (1 - pnorm(d1))
  return( as.numeric(C) )
}

# Test: Call Option
S_t = 100
K = 1600
r = 0.03
tau = 10/360
sigma = 1.05
black_scholes(S_t, K, r, tau, sigma)
```

```
## [1] 1501.334
```

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 2m
## 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:

```r
# 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
## 2m  0.0022393730  0.250000000   90.0000000
## 3m  0.0034170792  0.500000000  180.0000000
## 6m  0.0045123559  0.750000000  270.0000000
## 9m  0.0043206525  1.000000000  360.0000000
## 1y  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
```

```r
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 $T = 20d$

```r
price_option <- function(T, K, calls, rf_mat, stock=NA, put=FALSE){
  # Calculates
  # 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
  #   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

  # 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])

  # 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"]
  }

  # retrieve last price for option from VIx
  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,
              r_interp = r,
              calls = calls_sub, # subset of calls used
              rates = rates # subset of rates used
              ))
}
```

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 $T = 40d$

```r
# First Call Option
price_option(T=20, K=1600, calls, rf_mat, stock = sp500)
```

```
## $Call
## [1] 6.452062
##
## $Put
## [1] NA
##
## $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, rf_mat, stock = sp500)
```

```
## $Call
## [1] 15.35329
##
## $Put
## [1] NA
##
## $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
## 2m 0.002239373 0.25000000   90
```

```r
# Third Call Option
price_option(T=40, K=1800, calls, rf_mat, stock = sp500)
```

```
## $Call
## [1] 118.2339
##
## $Put
## [1] NA
##
## $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
## 2m 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 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

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

$$H(x, y) = C(F(x), G(y))$$

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, instead of Gaussian marginals, we will use two Student-t distributions as marginals, say t

Log-returns

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

# calculate returns
rets <- 100 * PerformanceAnalytics::CalculateReturns(sp500, method="log")
rets <- rets[rowSums(is.na(rets)) == 0,] # remove nas
head(rets)
```

```
##                  sp500
## 2000-01-04 -3.90992269
## 2000-01-05  0.19203798
## 2000-01-06  0.09552461
## 2000-01-07  2.67299353
## 2000-01-10  1.11278213
## 2000-01-11 -1.31486343
```