TP2 Risk Management

TP2: Hair Parra , Alessio Bressan, Ioan Catalin

2023-04-18

Libraries

```
# Preload R libraries we will use
library("here")
library("xts")
library("zoo")
library("MASS")
library("Matrix")
library("fGarch") # to fit distr
library("copula") # to fit Copulas
library("plotly") # used to plot the volatility surface
library("mvtnorm") # multivariate Gaussian
library("quantmod")

# plotting
library("rgl")
library("akima")

# additional source code for this assg
source(here("code", "Utils.R")) # misc functions and utils
source(here("code", "GARCH.R")) # extended GARCH functions and code from class lectures
source(here("code", "OptionPricing.R")) # BlackScholes and Option pricing
source(here("code", "VolatilitySurface.R")) # Fitting and compu. vol. surface
```

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: num [1:3410, 1] 1455 1399 1402 1403 1441 ...
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
##  NULL
##  $ vix  :An 'xts' object on 2000-01-03/2013-09-10 containing:
##   Data: num [1:3410, 1] 0.242 0.27 0.264 0.257 0.217 ...
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
##  NULL
##  $ 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.

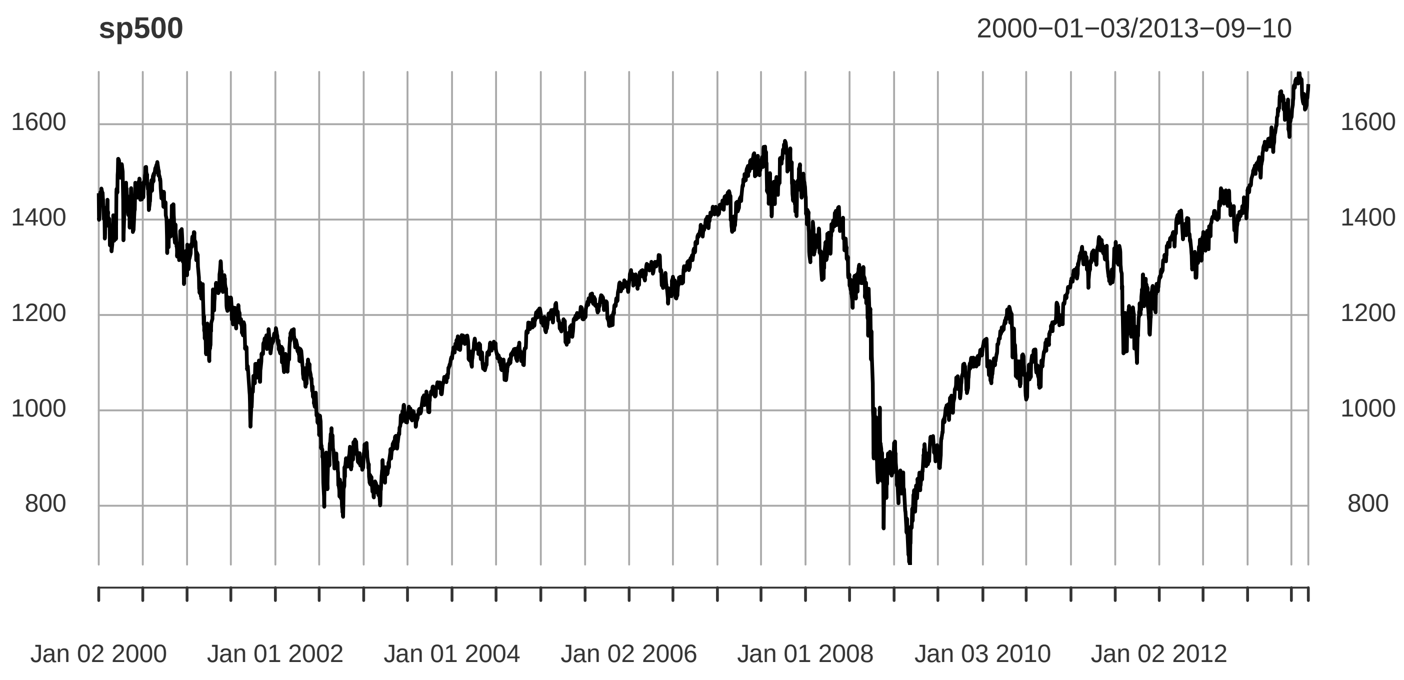
```
# show head of both indexes
head(sp500)
```

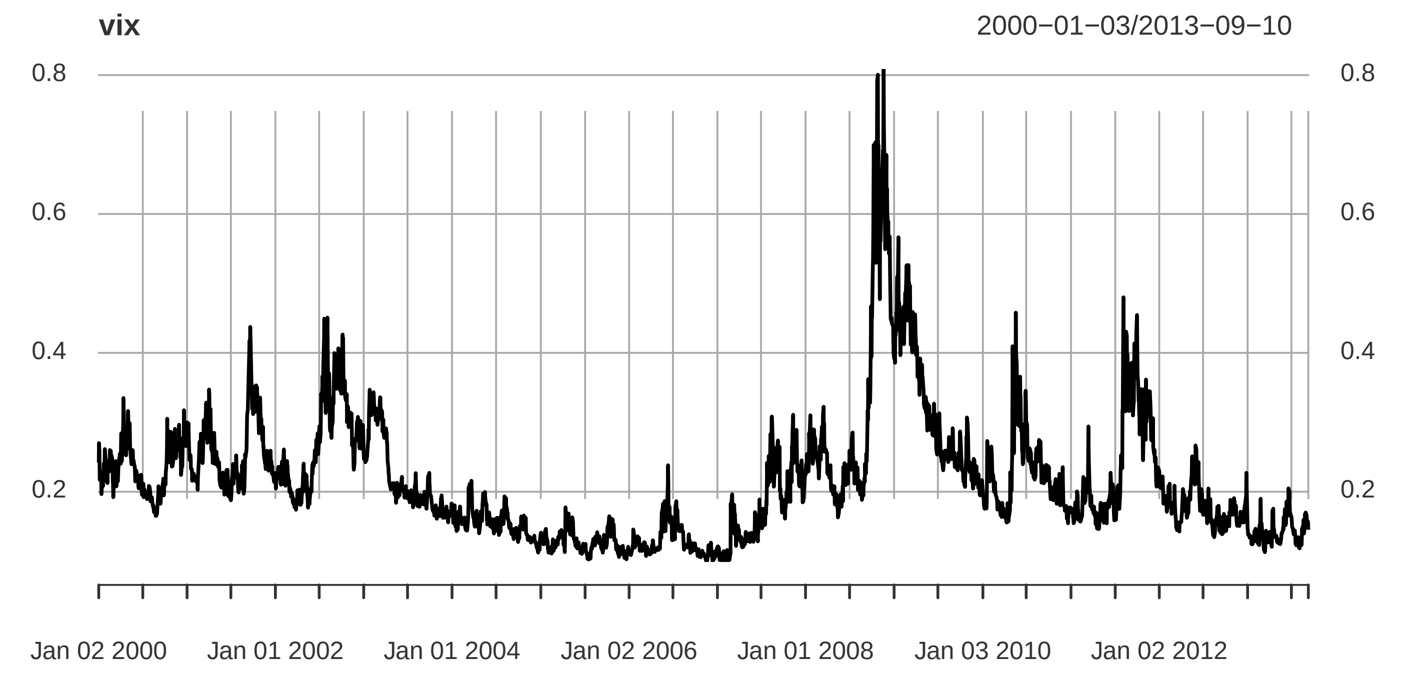
```
##               sp500
## 2000-01-03 1455.22
## 2000-01-04 1399.42
## 2000-01-05 1402.11
## 2000-01-06 1403.45
## 2000-01-07 1441.47
## 2000-01-10 1457.60
```

```r
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
```

```r
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:

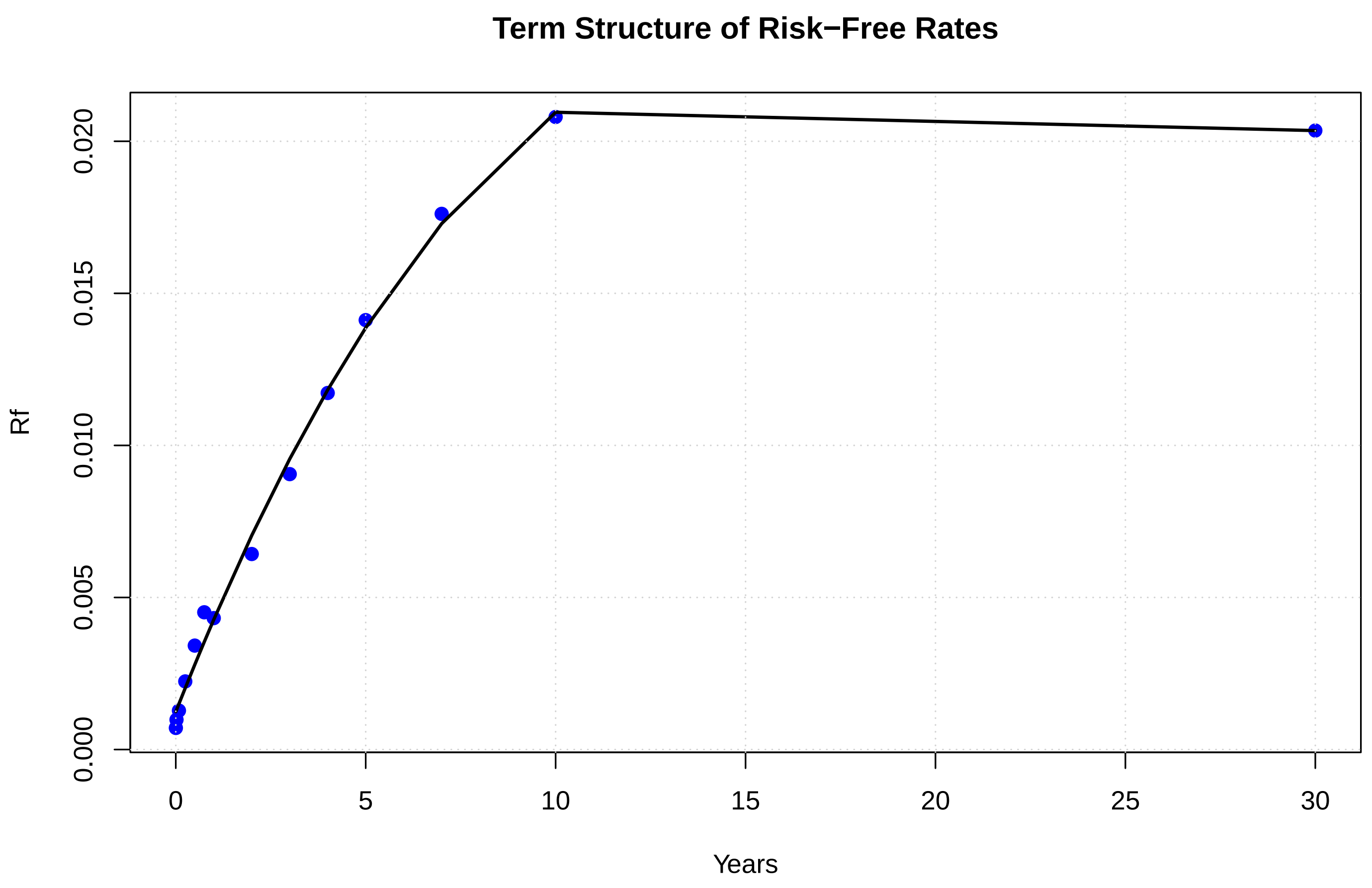
```
r_f <- as.vector(Rf)
names(r_f) <- c("1d","1w", "1m", "3m", "6m", "9m", "1y", "2y", "3y", "4y", "5y", "7y","10y", "30y")
r_f
```

```
##           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:

```
# 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

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

BlackScholes function

The Black-Scholes function is implemented under `OptionPricing.R::black-scholes()`:

```
# 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 $K = 1650$ with maturity $T = 20d$
3. **1x** strike $K = 1750$ with maturity $T = 40d$
4. **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.

We pack these into a list:

```
# Initialize strikes and maturities the options
T_vec <- c(20, 20, 40,40) #  maturities
K_vec <- c(1600, 1650, 1750, 1800) # Strikes
option_book <- list(T = T_vec, K = K_vec)
option_book
```

```
## $T
## [1] 20 20 40 40
##
## $K
## [1] 1600 1650 1750 1800
```

Nearest values

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

```
# Test: function used to get two nearest values in a vector (OptionsPricing.R)
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:

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

This function `interpolate()` is implemented under the `OptionPricing.R` script.

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 from `rf_mat`. This is done in the `price_option()` function under `code/OptionPricing.R`

Example

1x strike $K = 1600$ with maturity $T = 20d$

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

## test: specific price (func from OptionPricing.R)
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
```

where,

- `$Call`: The calculated Call option price
- `$Put`: The calculated Put option price (if `put=TRUE`. Set to `FALSE` by default).
- `$S`: Underlying price
- `$K`: Strike price
- `$r_intep`: Interpolated risk-free rate from the term structure of risk-free rates.
- `$calls`: Relevant values from the `calls` matrix.
- `$rates`: Rates used to find the interpolation.

Pricing the book of options

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

1. **1x** strike $K = 1600$ with maturity $T = 20d$
2. **1x** strike $K = 1650$ with maturity $T = 20d$
3. **1x** strike $K = 1750$ with maturity $T = 40d$
4. **1x** strike $K = 1800$ with maturity $T = 40d$

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

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

Then, we price the options accordingly:

```
# 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
```

```
# Second Call Option
price_option(T=20, K=1650, calls=calls, rf_mat=rf_mat, S_t = S_t, IV = IV)
```

```
## $Call
## [1] 47.70804
##
## $Put
## [1] NA
##
## $S
## [1] 1683.99
##
## $K
## [1] 1650
##
## $r_interp
## [1] 0.001264335
##
## $calls
##         K        tau        IV  tau_days
## [1,] 1650 0.02557005 0.1456375  6.392513
## [2,] 1650 0.10228238 0.1448237 25.570595
##
## $rates
##               rf      years      days
## 1w 0.0009799908 0.01923077  6.923077
## 1m 0.0012799317 0.08333333 30.000000
```

```
# Third Call Option
price_option(T=40, K=1750, calls=calls, rf_mat=rf_mat, S_t = S_t, IV = IV)
```

```
## $Call
## [1] 15.25057
##
## $Put
## [1] NA
##
## $S
## [1] 1683.99
##
## $K
## [1] 1750
##
## $r_interp
## [1] 0.001721275
##
## $calls
##         K       tau        IV tau_days
## [1,] 1750 0.1022824 0.1047194 25.57059
## [2,] 1750 0.1789947 0.1130030 44.74868
##
## $rates
##             rf      years days
## 1m 0.001279932 0.08333333   30
## 3m 0.002239373 0.25000000   90
```

```
# Fourth 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
```

Some Theoretical Workings

We present some important theory from which we based the implementation of a variety of the functions we use in this project.

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:

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

Since this is shared by all the subsequent parts, we compute the log-returns for both of the indexes.

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

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

# remove first return
sp500_rets <- sp500_rets[-1]
vix_rets <- vix_rets[-1]

# remove nas
sp500_rets[is.na(sp500_rets)] <- 0
vix_rets[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
```

Computing Prices from Returns

In order to produce forecasts, we crate a function to forecast the 5 day ahead prices from the returns. Since:

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

This logic is implemented in the `f_next_Pt()` and `f_logret_to_price()`, lkocated under the `code/Utils.R` folder.

Value at Risk (VaR) and Expected Shortfall (ES)

VaR: For a random variable X, the **Value-at-Risk (VaR)** at level α is defined as the α-lower quantile of the distribution of X, thus:

$$VaR_X(\alpha) = F_X^{-1}(1-\alpha)$$

ES: Expected shortfall is calculated by averaging all of the returns in the distribution that are worse than the VAR of the portfolio at a given level of confidence.

Profit of an European Option

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

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

, where:

- S: Spot price (current)
- S_0: Spot price at the begining 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

One risk driver and Gaussian Model

Steps:

1. Compute the daily log-returns of the underlying stock.
2. Assume they are iid normally distributed.
3. Generate 10 000 scenarios for the one-week ahead (five days) underlying price using the normal distribution fitted to the past invariants.
4. Determine the P&L distribution of the book of options, using the simulated underlying values. Assume the implied volatility stays the same. Take interpolated rates for the term structure.
5. Compute the VaR95 and the ES95.

Gaussian fit to underlying and simulation

```r
# simulation parameters
n_ahead = 5  # number of days ahead
n_sim = 10000 # number of simulations

# Obtain MLE Gaussian parameters from the log-returns
mean_sp500 = mean(sp500_rets) #mean of sp500
sd_sp500 = sd(sp500_rets) #standard deviation of sp500

# examine parameters
mean_sp500
```

```
## [1] 0.00004283042
```

```r
sd_sp500
```

```
## [1] 0.01332592
```

```r
#initialize matrix of returns forecasted until T+5
sp500_rets_forecast = matrix(NA,n_sim, n_ahead)

# simulate for each day-ahead
for(t in 1:n_ahead)
{
  # sample 10k times from the Gaussian with MLE fitted parameters
  sp500_rets_forecast[, t] = rnorm(n_sim,
                                   mean = mean_sp500,
                                   sd = sd_sp500)
}

# assign column names to simulations
colnames(sp500_rets_forecast) = c("T+1", "T+2", "T+3", "T+4", "T+5")

# display
head(sp500_rets_forecast)
```

```
##               T+1           T+2           T+3          T+4          T+5
## [1,]  0.017005823  0.0073089051 -0.0115141102 -0.003427709 -0.001716596
## [2,]  0.002078350 -0.0115245462  0.0252682790 -0.011453261 -0.000712799
## [3,]  0.008794631  0.0008169275 -0.0099349616  0.026960675 -0.008663354
## [4,]  0.003454860  0.0219946008  0.0126413691 -0.014854845  0.013330971
## [5,] -0.020716646 -0.0023375453 -0.0007802683  0.012998556 -0.009341018
## [6,] -0.025337923  0.0091787516  0.0268056074  0.012639150 -0.008375990
```

Computing Prices from Returns

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

# calculate the price and values from the simulated log-returns
sim_val_mats_one_driver <- f_logret_to_price(sp_init = spT,
                                             sim_rets_sp500 = sp500_rets_forecast,
                                             n_ahead = n_ahead
                                             )

# unpack matrices
sim_price_sp500_one_driver <- sim_val_mats_one_driver$sp500

# compare simulated returns with the price
head(sim_price_sp500_one_driver)
```

```
##        T+1      T+2      T+3      T+4      T+5
## 1 1712.873 1725.438 1705.685 1699.848 1696.933
## 2 1687.494 1668.158 1710.846 1691.363 1690.158
## 3 1698.865 1700.254 1683.445 1729.450 1714.532
## 4 1689.818 1727.397 1749.372 1723.577 1746.708
## 5 1649.462 1645.611 1644.328 1665.841 1650.353
## 6 1641.857 1656.997 1702.014 1723.663 1709.286
```

```r
# save data from the simulated values
save(sim_price_sp500_one_driver, file=here("data_out", "sim_price_sp500_one_driver.rda"))
```

Pricing the simulations and Profit/Loss Distribution

Option Pricing of Simulated Values

Next, we calculate the price of the book of options for the simulated values using the `f_opt_price_simulation()` function under `code/OptionPricing.R`:

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

# obtain number of strike prices
n_K <- length(option_book$K)

# generate option names for the list
optnames <- as.vector(mapply(paste0, rep("opt", n_K), seq(1:n_K)))

# Initialize Profit/Loss mats for each option in the book
call_price_matrices <- initialize_sim_mats(sim_price_sp500_one_driver,# copy mat dims
                                           num_mats = length(K_vec),
                                           lnames=optnames
                                           )

# Initialize Profit/Loss mats for each option in the book
PL_matrices <- initialize_sim_mats(sim_price_sp500_one_driver,
                                   num_mats = length(K_vec),
                                   lnames=optnames
                                   )
```

```r
#Loop to calculate the P&L of each option from our book of options
for(i in 1:n_K) # each of the options
{
  for(j in 1:n_ahead) # for each of the days
  {
    # compute the call price for the i-th option at the j-th day
    price_call = prc_opt(option_book$T[i]-j, # shifted maturity
                         option_book$K[i], # strike price
                         calls, # matrix of calls values
                         rf_mat, # structure of risk-free rates
                         sim_price_sp500_one_driver[,j],
                         vix[[length(vix)]]) # use the last day of the vix for all options prices

    # Assign the Call price to matrix
    call_price_matrices[[i]][,j] = price_call

    # Compute and assign profit loss for opt i at  day j
    PL_matrices[[i]][,j] = option_profit(S = sim_price_sp500_one_driver[,j],
                                         K = option_book$K[i],
                                         c = price_call)$call_profit

  }
}
```

Option Pricing of Simulated Values

```r
# option prices for each of the options
head(call_price_matrices$opt1)
```

```
##         T+1       T+2       T+3       T+4       T+5
## 1 114.21767 126.22299 106.98233 101.21853  98.23697
## 2  90.43462  72.90027 111.93714  93.19913  91.82121
## 3 100.94658 102.02470  86.14394 129.94478 115.22097
## 4  92.56070 128.13462 149.69751 124.17947 146.95159
## 5  57.94608  54.42143  52.87861  70.07581  56.55800
## 6  52.12214  63.49563 103.48176 124.26322 110.11896
```

```r
head(call_price_matrices$opt2)
```

```
##        T+1      T+2       T+3      T+4      T+5
## 1 69.33862 79.66383  62.44676 57.20520 54.31956
## 2 49.58669 35.95667  66.70890 50.61226 49.07656
## 3 58.08873 58.62728  45.44190 82.50573 68.90977
## 4 51.27445 81.39128 101.19869 77.24932 98.21505
## 5 26.14329 23.53733  22.20305 33.10327 23.64521
## 6 22.47264 29.42042  59.48231 77.32512 64.42816
```

```r
head(call_price_matrices$opt3)
```

```
##         T+1       T+2       T+3       T+4       T+5
## 1 23.885204 28.294059 20.416415 18.070070 16.728045
## 2 15.784722 10.757085 22.194057 15.576105 14.808636
## 3 19.123248 19.111481 13.893310 28.954003 22.532567
## 4 16.430026 29.112682 38.826862 26.514976 36.405384
## 5  7.672210  6.798688  6.337572  9.583467  6.628573
## 6  6.537811  8.622774 19.214432 26.549471 20.675129
```

```
head(call_price_matrices$opt4)
```

```
##         T+1       T+2       T+3       T+4       T+5
## 1 10.822108 13.201655  8.774055  7.481960  6.734105
## 2  6.548224  4.083224  9.716726  6.242978  5.803149
## 3  8.258955  8.174861  5.495288 13.351098  9.705570
## 4  6.872876 13.672221 19.358868 11.976821 17.591216
## 5  2.759012  2.359847  2.141981  3.468266  2.197043
## 6  2.281465  3.133040  8.147893 11.996036  8.730768
```

Distribution of Options P/L

```
#showing the values of the P&L distribution for the first option where T = 20 and K = 1600
head(PL_matrices$opt1)
```

```
##         T+1       T+2       T+3      T+4      T+5
## 1  50.08208  75.64441 117.03784 144.7478 165.4473
## 2  73.86512 128.96712 112.08303 152.7672 171.8631
## 3  63.35316  99.84269 137.87622 116.0215 148.4633
## 4  71.73904  73.73277  74.32266 121.7868 116.7327
## 5 106.35366 147.44597 171.14156 175.8905 207.1263
## 6 112.17761 138.37176 120.53840 121.7031 153.5653
```

```
head(PL_matrices$opt2)
```

```
##        T+1       T+2       T+3      T+4      T+5
## 1 44.96112  72.20357 111.57341 138.7611 159.3647
## 2 64.71305 115.91072 107.31127 145.3540 164.6078
## 3 56.21101  93.24012 128.57827 113.4606 144.7745
## 4 63.02529  70.47611  72.82148 118.7170 115.4693
## 5 88.15645 128.33006 151.81712 162.8630 190.0391
## 6 91.82711 122.44697 114.53785 118.6412 149.2562
```

```
head(PL_matrices$opt3)
```

```
##         T+1      T+2      T+3      T+4       T+5
## 1 -9.585462 23.57333 53.60375 77.89622  96.95626
## 2 -1.484980 41.11031 51.82611 80.39019  98.87567
## 3 -4.823506 32.75591 60.12686 67.01229  91.15174
## 4 -2.130284 22.75471 35.19331 69.45131  77.27893
## 5  6.627532 45.06870 67.68260 86.38282 107.05574
## 6  7.761931 43.24462 54.80574 69.41682  93.00918
```

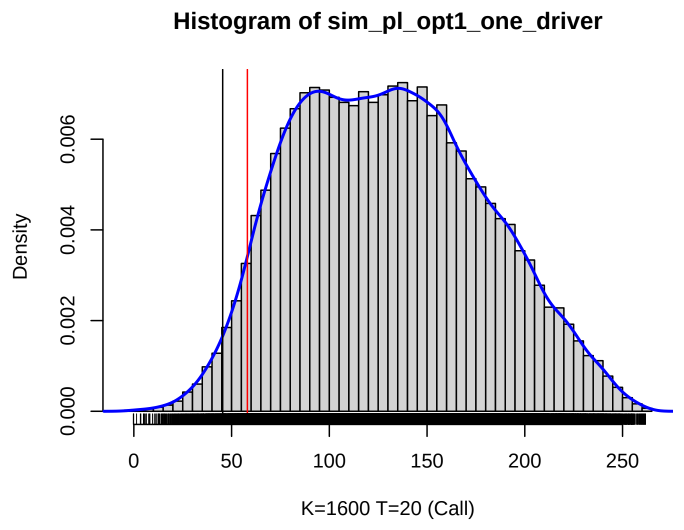
```
head(PL_matrices$opt4)
```

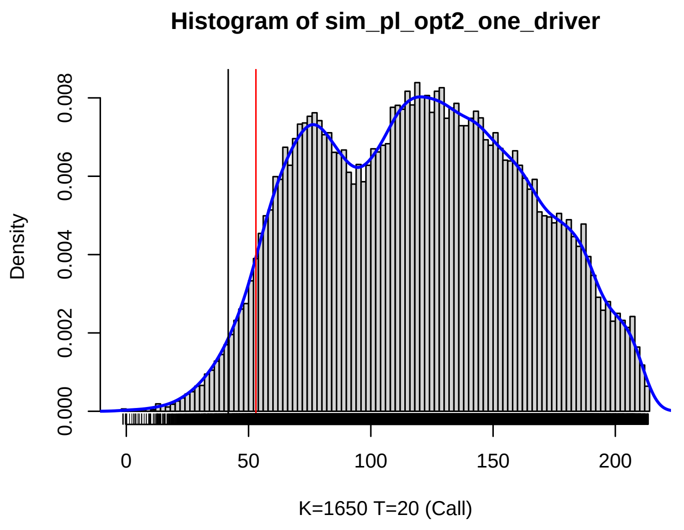
```
##          T+1        T+2       T+3      T+4      T+5
## 1 -10.822108 -11.3342628 15.246114 38.48433 56.95020
## 2  -6.548224  -2.2158318 14.303443 39.72331 57.88116
## 3  -8.258955  -6.3074688 18.524881 32.61519 53.97874
## 4  -6.872876 -11.8048288  4.661301 33.98947 46.09309
## 5  -2.759012  -0.4924548 21.878188 42.49802 61.48727
## 6  -2.281465  -1.2656478 15.872276 33.97025 54.95354
```

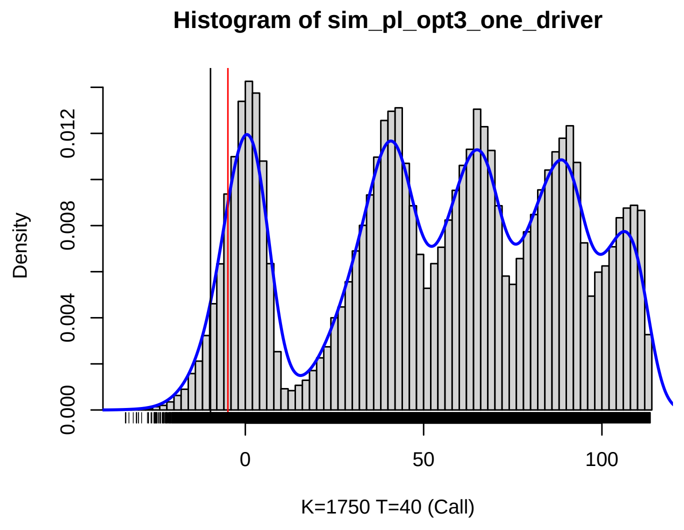
Distribution of Options P/L

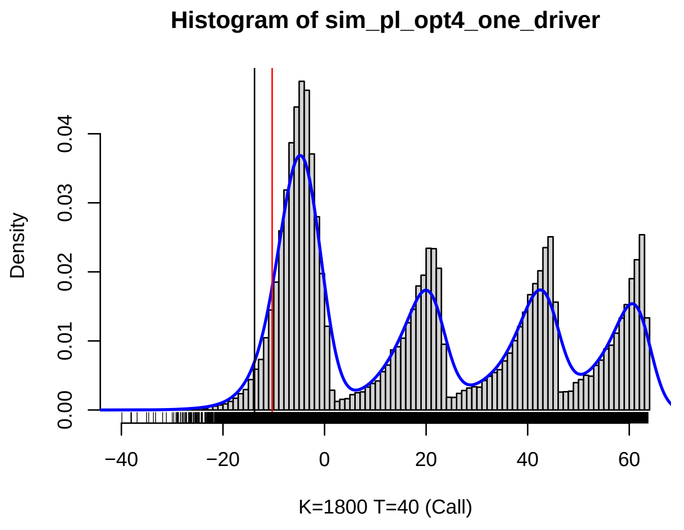
Next, using all the simulated profits and losses for each of the options, we display a histogram for the distribution for each of the options, for the aggregated 5 days of simulation:

```r
# flatten the matrices 5-days ahead simulated P/L for the three options
sim_pl_opt1_one_driver <- as.vector(PL_matrices$opt1)
sim_pl_opt2_one_driver <- as.vector(PL_matrices$opt2)
sim_pl_opt3_one_driver <- as.vector(PL_matrices$opt3)
sim_pl_opt4_one_driver <- as.vector(PL_matrices$opt4)

# Compute the 95% VaR and 95% ES
opt1_one_driver_VaR_ES <- f_VaR_ES(sim_pl_opt1_one_driver, alpha = 0.05)
opt2_one_driver_VaR_ES <- f_VaR_ES(sim_pl_opt2_one_driver, alpha = 0.05)
opt3_one_driver_VaR_ES <- f_VaR_ES(sim_pl_opt3_one_driver, alpha = 0.05)
opt4_one_driver_VaR_ES <- f_VaR_ES(sim_pl_opt4_one_driver, alpha = 0.05)

# plot the distribution for each of the options
par(mfrow = c(2,2))

# distribution of first option
hist(sim_pl_opt1_one_driver, nclass = round(10 * log(n_sim)),
     probability = TRUE, xlab=paste0("K=", K_vec[1], " T=", T_vec[1], " (Call)"))
lines(density(sim_pl_opt1_one_driver), lwd=2, col="blue")
abline(v=opt1_one_driver_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt1_one_driver_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt1_one_driver)


# distribution of second option
hist(sim_pl_opt2_one_driver, nclass = round(10 * log(n_sim)),
     probability = TRUE, xlab=paste0("K=", K_vec[2], " T=", T_vec[2], " (Call)"))
lines(density(sim_pl_opt2_one_driver), lwd=2, col="blue")
abline(v=opt2_one_driver_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt2_one_driver_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt2_one_driver)


# distribution of third option
hist(sim_pl_opt3_one_driver, nclass = round(10 * log(n_sim)),
     probability = TRUE, xlab=paste0("K=", K_vec[3], " T=", T_vec[3], " (Call)"))
lines(density(sim_pl_opt3_one_driver), lwd=2, col="blue")
abline(v=opt3_one_driver_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt3_one_driver_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt3_one_driver)


# distribution of fourth option
hist(sim_pl_opt4_one_driver, nclass = round(10 * log(n_sim)),
     probability = TRUE, xlab=paste0("K=", K_vec[4], " T=", T_vec[4], " (Call)"))
lines(density(sim_pl_opt4_one_driver), lwd=2, col="blue")
abline(v=opt4_one_driver_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt4_one_driver_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt4_one_driver)
```









VaR and ES

In what follows, we compute the 95% VaR and 95% ES for the P/L for the book of options (drawn as a red and black vertical lines in the previous plots).

95% VaR

```
opt1_one_driver_VaR_ES$VaR # first option
```

```
## [1] 58.08371
```

```
opt2_one_driver_VaR_ES$VaR # second option
```

```
## [1] 53.00309
```

```
opt3_one_driver_VaR_ES$VaR # third option
```

```
## [1] -4.885842
```

```
opt4_one_driver_VaR_ES$VaR # fourth option
```

```
## [1] -10.31715
```

95% ES

```
opt1_one_driver_VaR_ES$ES # first option
```

```
## [1] 45.44666
```

```
opt2_one_driver_VaR_ES$ES # second option
```

```
## [1] 41.67759
```

```
opt3_one_driver_VaR_ES$ES # third option
```

```
## [1] -9.762817
```

```
opt4_one_driver_VaR_ES$ES # fourth option
```

```
## [1] -13.78571
```

Two risk drivers and Gaussian Model

1. Compute the daily log-returns of the underlying stock.
2. Compute the daily log-returns of the VIX.
3. Assume they are invariants normally distributed.
4. Generate 10 000 scenarios for the one-week ahead underlying price and the one week ahead VIX value using the normal distribution fitted to the past risk drivers.
5. Determine the P&L distribution of the book of options, using the simulated values. Take interpolated rates for the term structure.

Multivariate Gaussian Distribution

A random vector with a multivariate Gaussian dsitribution has pdf given by:

$$f(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{-1}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right),$$

where the mean vector μ and covariance matrix Σ are given by:

$$\mu := \mathbb{E}[X] \quad , \quad \Sigma := Cov(X) = \mathbb{E}[(X-\mu)(X-\mu)^T],$$

Generating the simulation scenarios

```r
# random seed for replication
set.seed(1234)

# Simulation parameters
n_sim = 10000 # set number of simulations
n_ahead = 5 # days ahead to produce samples

# MLE parameters to fit Bivariate Gaussian
rets <- rets <- cbind(sp500_rets, vix_rets)
mu <- apply(rets, 2, mean)
Sigma <- cov(rets)

# preallocate matrices to store simulations
sim_rets_sp500_two_drivers <- matrix(NA, nrow = n_sim, ncol=n_ahead)
sim_rets_vix_two_drivers <- matrix(NA, nrow = n_sim, ncol=n_ahead)

# assign days ahead
colnames(sim_rets_sp500_two_drivers) <- c("T+1", "T+2", "T+3", "T+4", "T+5")
colnames(sim_rets_vix_two_drivers) <- c("T+1", "T+2", "T+3", "T+4", "T+5")

# perfom n_ahead days of n_sim scenarios
for(t in 1:n_ahead){

  # Sample n_sim times from Bivariate Gaussian
  U_sim <- rmvnorm(mean = mu, sigma = Sigma, n = n_sim)

  # store simulation of log return in matrix
  sim_rets_sp500_two_drivers[ ,t] <- U_sim[, 1]
  sim_rets_vix_two_drivers[ ,t] <- U_sim[, 2]
}

# preview of simulated log returns
head(sim_rets_sp500_two_drivers)
```

```
##                T+1          T+2          T+3          T+4          T+5
## [1,] -0.01452047884 -0.008575627 -0.008985105 -0.015451475  0.008642492
## [2,]  0.03146009812 -0.008756203 -0.008519667  0.003134699 -0.016605277
## [3,] -0.00006771544  0.001335024 -0.014045626 -0.012531388  0.018494012
## [4,] -0.00095596747 -0.007538540  0.012621275  0.012048710  0.009144804
## [5,]  0.00215055720  0.025399617  0.036646435  0.019970952  0.020314311
## [6,]  0.00397546667 -0.019472596 -0.008878132 -0.009106073 -0.012019799
```

```
head(sim_rets_vix_two_drivers)
```

```
##             T+1         T+2          T+3          T+4         T+5
## [1,]  0.02790344 -0.04563389  0.007363624  0.020320459 -0.01486904
## [2,] -0.15757613  0.02198088  0.003944074  0.005411475  0.07385687
## [3,]  0.02801445 -0.08725384  0.031677422  0.025219017 -0.10499518
## [4,] -0.02959970  0.10395645 -0.088927866 -0.020812520 -0.10768910
## [5,] -0.05134825 -0.09259495 -0.103083070 -0.048474265 -0.05330125
## [6,] -0.05894487  0.05974810  0.028461994  0.051213103  0.05162827
```

Computing Prices from Returns

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

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

# calculate the price and values from the simulated log-returns
sim_val_mats_two_drivers <- f_logret_to_price(sp_init = spT,
                                              vix_init = vixT,
                                              sim_rets_sp500 = sim_rets_sp500_two_drivers,
                                              sim_rets_vix = sim_rets_vix_two_drivers
                                              )

# unpack matrices
sim_price_sp500_two_drivers <- sim_val_mats_two_drivers$sp500
sim_vol_vix_two_drivers <- sim_val_mats_two_drivers$vix
```

```
# compare simulated returns with the price
head(sim_rets_sp500_two_drivers)
```

```
##                T+1          T+2          T+3          T+4          T+5
## [1,] -0.01452047884 -0.008575627 -0.008985105 -0.015451475  0.008642492
## [2,]  0.03146009812 -0.008756203 -0.008519667  0.003134699 -0.016605277
## [3,] -0.00006771544  0.001335024 -0.014045626 -0.012531388  0.018494012
## [4,] -0.00095596747 -0.007538540  0.012621275  0.012048710  0.009144804
## [5,]  0.00215055720  0.025399617  0.036646435  0.019970952  0.020314311
## [6,]  0.00397546667 -0.019472596 -0.008878132 -0.009106073 -0.012019799
```

```
head(sim_price_sp500_two_drivers)
```

```
##        T+1      T+2      T+3      T+4      T+5
## 1 1659.714 1645.542 1630.823 1605.818 1619.756
## 2 1737.811 1722.660 1708.046 1713.409 1685.192
## 3 1683.876 1686.125 1662.608 1641.904 1672.551
## 4 1682.381 1669.746 1690.954 1711.451 1727.174
## 5 1687.615 1731.029 1795.642 1831.863 1869.457
## 6 1690.698 1658.094 1643.439 1628.541 1609.084
```

```
# compare simulatedlog rets with volatility
head(sim_rets_vix_two_drivers)
```

```
##              T+1         T+2          T+3          T+4         T+5
## [1,]  0.02790344 -0.04563389  0.007363624  0.020320459 -0.01486904
## [2,] -0.15757613  0.02198088  0.003944074  0.005411475  0.07385687
## [3,]  0.02801445 -0.08725384  0.031677422  0.025219017 -0.10499518
## [4,] -0.02959970  0.10395645 -0.088927866 -0.020812520 -0.10768910
## [5,] -0.05134825 -0.09259495 -0.103083070 -0.048474265 -0.05330125
## [6,] -0.05894487  0.05974810  0.028461994  0.051213103  0.05162827
```

```
head(sim_vol_vix_two_drivers)
```

```
##         T+1       T+2       T+3       T+4       T+5
## 1 0.1494115 0.1427465 0.1438015 0.1467535 0.1445875
## 2 0.1241170 0.1268754 0.1273768 0.1280679 0.1378847
## 3 0.1494281 0.1369425 0.1413499 0.1449600 0.1305116
## 4 0.1410622 0.1565159 0.1431982 0.1402487 0.1259302
## 5 0.1380274 0.1258206 0.1134968 0.1081263 0.1025139
## 6 0.1369828 0.1454168 0.1496151 0.1574769 0.1658207
```

```
# save data from the simulated values
save(sim_price_sp500_two_drivers, file=here("data_out", "sim_vol_sp500_student_two_driver.rda"))
save(sim_vol_vix_two_drivers, file=here("data_out", "sim_vol_vix_student_two_driver.rda"))
```

Pricing the simulation scenarios

Recall the initial (call) options:

1. **1x** strike $K = 1600$ with maturity $T = 20d$
2. **1x** strike $K = 1650$ with maturity $T = 20d$
3. **1x** strike $K = 1750$ with maturity $T = 40d$
4. **1x** strike $K = 1800$ with maturity $T = 40d$

Option Pricing of Simulated Values

Next, we calculate the price of the book of options for the simulated values using the `f_opt_price_simulation()` function under `code/OptionPricing.R`:

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

# Obtain the obtion prices from simulation values
opt_price_mats_two_drivers <- f_opt_price_simulation(sim_price_sp500 = sim_price_sp500_two_drivers,
                                                     sim_vol_vix = sim_vol_vix_two_drivers,
                                                     K_vec = option_book$K,
                                                     T_vec = option_book$T,
                                                     put=FALSE)
```

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

```
##         T+1       T+2       T+3       T+4       T+5
## 1  66.68728  54.02917  42.71021  26.80658  34.02905
## 2 138.10830 123.09877 108.68059 113.86093  86.78888
## 3  87.48674  88.33310  67.33352  50.40038  74.62107
## 4  85.46622  75.39346  92.94920 112.21830 127.39399
## 5  90.03235 131.35543 195.77394 231.98490 269.56908
## 6  92.82222  64.41266  52.76334  42.46920  30.82871
```

```
head(opt_price_mats_two_drivers$opt2)
```

```
##        T+1      T+2       T+3       T+4       T+5
## 1 32.38157 23.05594  16.16312   8.20146  11.07612
## 2 89.58968 75.59114  62.44611  66.80010  44.38387
## 3 47.67859 46.72819  31.23345  20.28261  34.40737
## 4 45.28501 38.84878  50.62105  66.22985  78.86678
## 5 48.54975 83.15830 145.80883 181.98943 219.57259
## 6 50.67050 30.05014  22.51934  16.73860  10.84459
```

```
head(opt_price_mats_two_drivers$opt3)
```

```
##        T+1       T+2       T+3       T+4        T+5
## 1 10.16496  6.420129  4.494681  2.474435   3.131890
## 2 28.49501 22.417872 16.959402 18.480885  12.036857
## 3 15.71079 13.230025  8.642336  5.696024   8.014769
## 4 13.53483 13.250368 15.439190 20.710010  22.625763
## 5 14.22807 25.531258 59.182806 87.095500 121.028158
## 6 14.82393  8.836440  6.821049  5.617638   4.223514
```

```
head(opt_price_mats_two_drivers$opt4)
```

```
##         T+1       T+2       T+3       T+4       T+5
## 1  3.953736  2.165600  1.408310  0.701712  0.894689
## 2 12.091303  8.985618  6.280185  6.945608  4.301397
## 3  6.652558  4.979943  3.022374  1.855286  2.469318
## 4  5.309532  5.568502  6.171834  8.610549  8.747185
## 5  5.539641 10.522942 29.343085 48.571779 75.779449
## 6  5.786782  3.228457  2.406129  1.984229  1.475170
```

```
# Compute the profits and loses for the simulation  from the simulated option premiums
PL_mats_two_drivers <- f_pl_simulation(sim_price_sp500 = sim_price_sp500_two_drivers,
                                       opt_price_mats = opt_price_mats_two_drivers,
                                       K_vec =option_book$K)
```

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

```
##         T+1       T+2       T+3       T+4       T+5
## 1 102.44520 165.07191 184.59617 221.14125 256.77849
## 2  31.02417  96.00231 118.62579 134.08690 204.01866
## 3  81.64574 130.76798 159.97286 197.54745 216.18648
## 4  83.66625 143.70762 134.35718 135.72952 163.41355
## 5  79.10013  87.74564  31.53244  15.96292  21.23846
## 6  76.31026 154.68842 174.54304 205.47863 259.97883
```

```
head(PL_mats_two_drivers$PL2)
```

```
##        T+1       T+2       T+3      T+4       T+5
## 1 86.75090 146.04513 161.14326 189.7464 229.73142
## 2 29.54280  93.50993 114.86027 131.1477 196.42367
## 3 71.45389 122.37288 146.07293 177.6652 206.40017
## 4 73.84747 130.25230 126.68533 131.7180 161.94076
## 5 70.58273  85.94278  31.49755  15.9584  21.23495
## 6 68.46197 139.05093 154.78704 181.2092 229.96296
```

```
head(PL_mats_two_drivers$PL3)
```

```
##          T+1      T+2      T+3      T+4       T+5
## 1  8.967514 62.68095 72.81170 95.47339 137.67565
## 2 -9.362533 46.68320 60.34698 79.46694 128.77068
## 3  3.421682 55.87105 68.66404 92.25180 132.79277
## 4  5.597646 55.85071 61.86719 77.23782 118.18178
## 5  4.904408 43.56982 18.12357 10.85233  19.77938
## 6  4.308544 60.26464 70.48533 92.33019 136.58403
```

```
head(PL_mats_two_drivers$PL4)
```

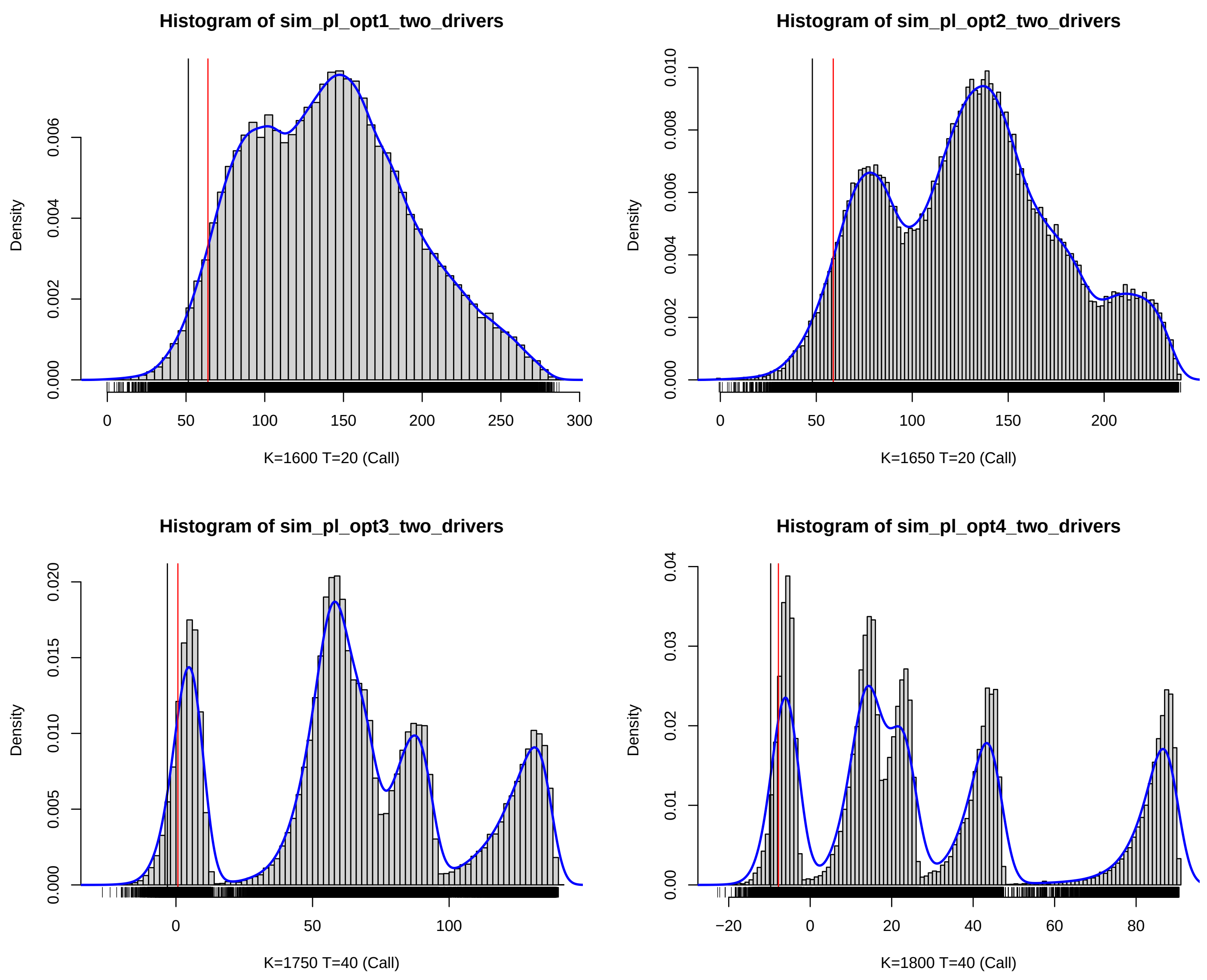
```
##           T+1       T+2       T+3        T+4      T+5
## 1  -3.953736 16.935475 25.898071 47.2461139 89.91285
## 2 -12.091303 10.115457 21.026196 41.0022179 86.50614
## 3  -6.652558 14.121132 24.284007 46.0925399 88.33822
## 4  -5.309532 13.532573 21.134547 39.3372769 82.06036
## 5  -5.539641  8.578133 -2.036704 -0.6239531 15.02809
## 6  -5.786782 15.872618 24.900252 45.9635969 89.33237
```

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_two_drivers <- as.vector(PL_mats_two_drivers$PL1)
sim_pl_opt2_two_drivers <- as.vector(PL_mats_two_drivers$PL2)
sim_pl_opt3_two_drivers <- as.vector(PL_mats_two_drivers$PL3)
sim_pl_opt4_two_drivers <- as.vector(PL_mats_two_drivers$PL4)

# Compute the 95% VaR and 95% ES
opt1_two_drivers_VaR_ES <- f_VaR_ES(sim_pl_opt1_two_drivers, alpha = 0.05)
opt2_two_drivers_VaR_ES <- f_VaR_ES(sim_pl_opt2_two_drivers, alpha = 0.05)
opt3_two_drivers_VaR_ES <- f_VaR_ES(sim_pl_opt3_two_drivers, alpha = 0.05)
opt4_two_drivers_VaR_ES <- f_VaR_ES(sim_pl_opt4_two_drivers, alpha = 0.05)

# plot the distribution for each of the options
par(mfrow = c(2,2))

# distribution of first option
hist(sim_pl_opt1_two_drivers, nclass = round(10 * log(n_sim)),
     probability = TRUE, xlab=paste0("K=", K_vec[1], " T=", T_vec[1], " (Call)"))
lines(density(sim_pl_opt1_two_drivers), lwd=2, col="blue")
abline(v=opt1_two_drivers_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt1_two_drivers_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt1_two_drivers)

# distribution of second option
hist(sim_pl_opt2_two_drivers, nclass = round(10 * log(n_sim)),
     probability = TRUE, xlab=paste0("K=", K_vec[2], " T=", T_vec[2], " (Call)"))
lines(density(sim_pl_opt2_two_drivers), lwd=2, col="blue")
abline(v=opt2_two_drivers_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt2_two_drivers_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt2_two_drivers)

# distribution of third option
hist(sim_pl_opt3_two_drivers, nclass = round(10 * log(n_sim)),
```

```r
     probability = TRUE, xlab=paste0("K=", K_vec[3], " T=", T_vec[3], " (Call)"))
lines(density(sim_pl_opt3_two_drivers), lwd=2, col="blue")
abline(v=opt3_two_drivers_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt3_two_drivers_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt3_two_drivers)

# distribution of fourth option
hist(sim_pl_opt4_two_drivers, nclass = round(10 * log(n_sim)),
     probability = TRUE, xlab=paste0("K=", K_vec[4], " T=", T_vec[4], " (Call)"))
lines(density(sim_pl_opt4_two_drivers), lwd=2, col="blue")
abline(v=opt4_two_drivers_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt4_two_drivers_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt4_two_drivers)
```



Two risk drivers and copula-marginal model

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:

$$C_\rho^{\text{Gauss}}(u, v) = \Phi_\rho(F_{\nu_1}^{-1}(u), F_{\nu_2}^{-1}(v)),$$

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

Generating the simulation scenarios

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

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

# convert to vector since fitting without dependence
sp500_rets_vec <- as.vector(sp500_rets)
vix_rets_vec <- as.vector(vix_rets)

# calculate means and sds for both indices
mu <- c(mean(sp500_rets_vec), mean(vix_rets_vec))
sigma <- c(sd(sp500_rets_vec), sd(vix_rets_vec))

# display
mu
```

```
## [1]  0.00004283042 -0.00014976541
```

```r
sigma
```

```
## [1] 0.01332592 0.06367330
```

Fitting Student-t to the marginals

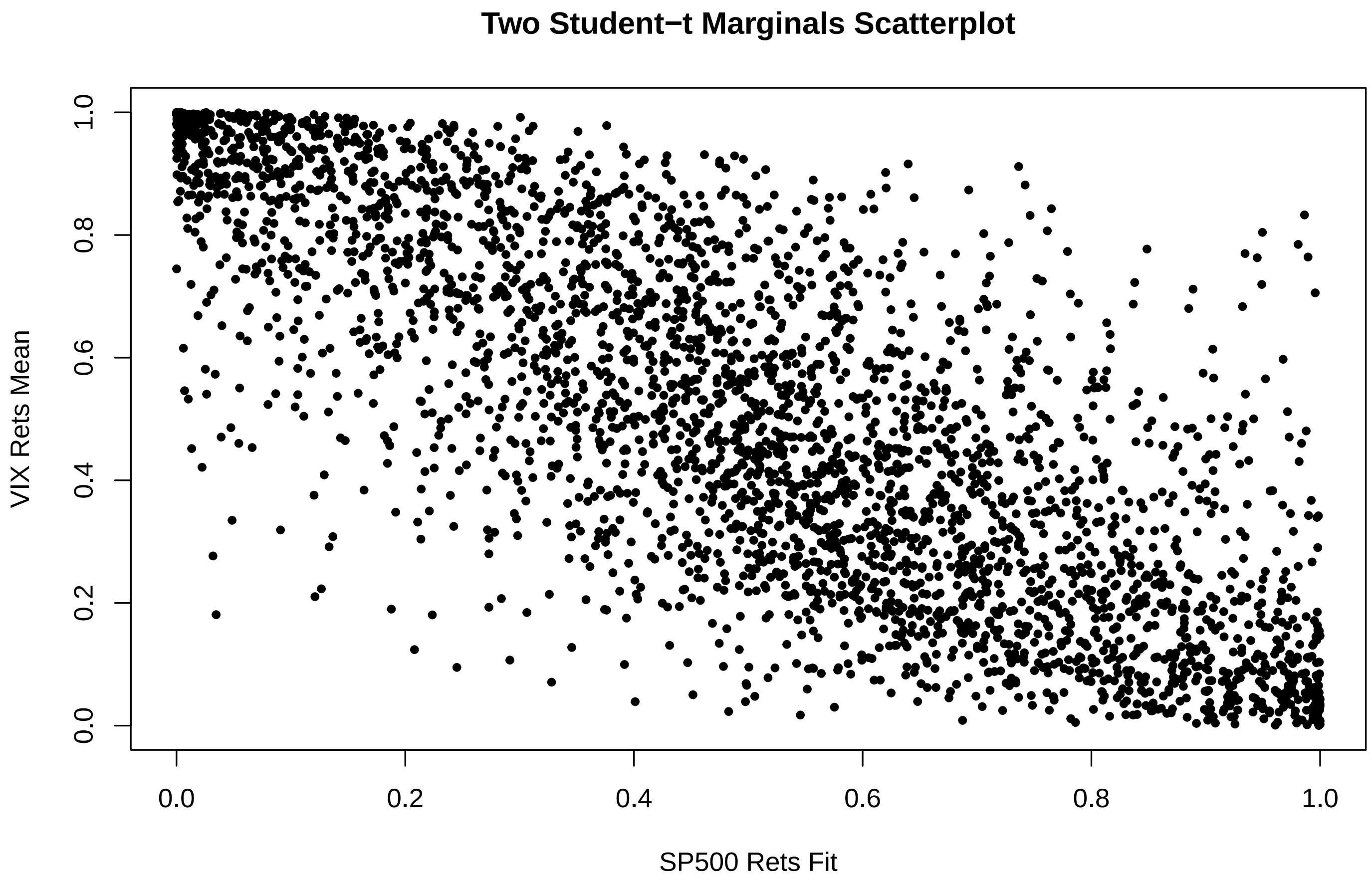
```
## Functions to optimize marginals fits by MLE, but fixing the degrees of freedom
f_t_optim1 <- function(x, mean, sd){
  dstd(x, mean=mean, sd=sd, nu=10) # nu is fixed!
}
f_t_optim2 <- function(x, mean, sd){
  dstd(x, mean=mean, sd=sd, nu=5)
}

# Student-t for sp500
fit1 <- suppressWarnings(
  fitdistr(x = sp500_rets_vec,
           densfun = f_t_optim1,
           start = list(mean = 0, sd = 1))
  )
theta1 <- fit1$estimate #extract fitted parameters

# Student-t for vix
fit2 <- suppressWarnings(
  fitdistr(x = vix_rets_vec,
           densfun = f_t_optim2,
           start = list(mean = 0, sd = 1))
  )
theta2 <- fit2$estimate # extract fitted parameters

# display parameters
theta1
```

```
##         mean           sd
## 0.0001985145 0.0118642050
```

```
theta2
```

```
##         mean           sd
## -0.003220748  0.062088272
```

```
# Fit Student-t to the marginals
U1 <- pstd(sp500_rets_vec, mean = theta1[1], sd = theta1[2], nu = 10) # sp500
U2 <- pstd(vix_rets_vec,mean = theta2[1], sd = theta2[2], nu = 5) # vix
U <- cbind(U1, U2) # join into one matrix
plot(U, pch = 20, cex = 0.9,
     main= "Two Student-t Marginals Scatterplot",
     xlab="SP500 Rets Fit",
     ylab="VIX Rets Mean")
```



Fitting the copula

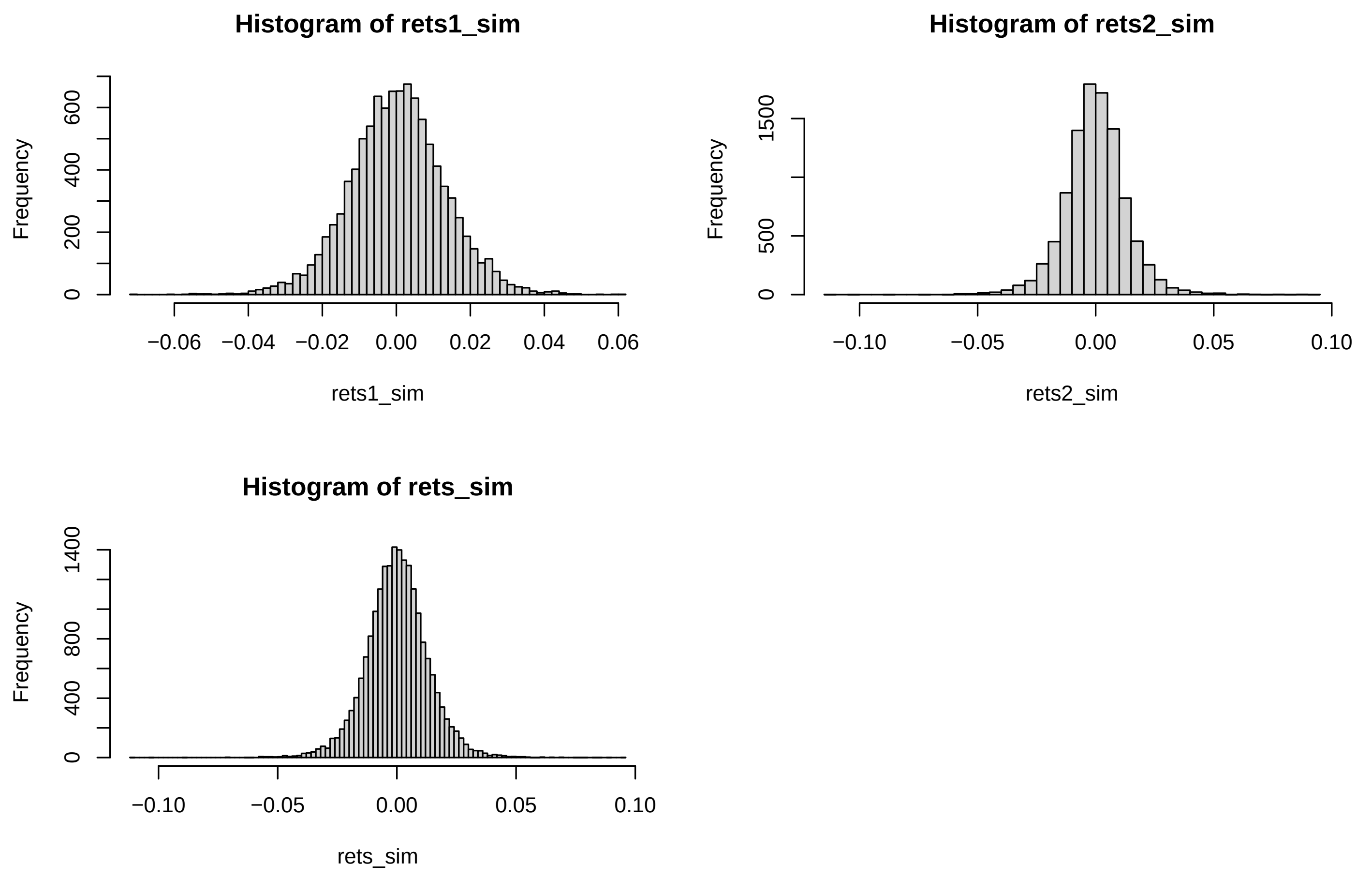
```
# Obtain the best rho for the Gaussian Copula
C <- copula::normalCopula(dim = 2)
fit <- copula::fitCopula(C, data = U, method = "ml")
fit
```

```
## Call: copula::fitCopula(C, data = U, ... = pairlist(method = "ml"))
## Fit based on "maximum likelihood" and 3409 2-dimensional observations.
## Copula: normalCopula
##   rho.1
## -0.7681
## The maximized loglikelihood is 1577
## Optimization converged
```

Sampling from the copula

```
## TEST: Sampling from copula n_sim times for one day

# seed for replication
set.seed(420)

# Simulation parameters
n_sim = 10000 # set number of simulations
```

```
# produce simulations from copula
U_sim <- rCopula(n_sim, fit@copula)

# use copula U_sim to reproduce the marginals with student-t distr
rets1_sim <- qstd(U_sim[,1], mean = mu[1], sd = sigma[1], nu = 10) # sp500
rets2_sim <- qstd(U_sim[,2], mean = mu[1], sd = sigma[1], nu = 5) # vix
rets_sim <- cbind(rets1_sim, rets2_sim)

# visualize
par(mfrow = c(2,2))
hist(rets1_sim, nclass=50)
hist(rets2_sim, nclass=50)
hist(rets_sim, nclass = round(10 * log(n_sim)))
```



We can now sample for the five days of interest using the fitted copula with marginal Student-t for the invariants:

```
# random seed for replication
set.seed(69)

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

# Simulation parameters
n_sim = 10000 # set number of simulations
n_ahead = 5 # days ahead to produce samples

# preallocate matrices to store simulations
sim_rets_sp500_copula <- matrix(NA, nrow = n_sim, ncol=5)
```

```r
sim_rets_vix_copula <- matrix(NA, nrow = n_sim, ncol=5)

# assign days ahead
colnames(sim_rets_sp500_copula) <- c("T+1", "T+2", "T+3", "T+4", "T+5")
colnames(sim_rets_vix_copula) <- c("T+1", "T+2", "T+3", "T+4", "T+5")

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

# perform n_head days of n_sim scenarios
for(t in 1:n_ahead){

  # Sample n_sim times from Gaussian Copula
  U_sim <- rCopula(n_sim, fit@copula)

  # use copula U_sim to reproduce the marginals quantiles F^{-1}(u) with student-t distr
  rets1_sim <- qstd(U_sim[,1], mean = theta1[1], sd = theta1[2], nu = 10) # sp500
  rets2_sim <- qstd(U_sim[,2], mean = theta2[1], sd = theta2[2], nu = 5) # vix
  # rets1_sim <- qt(U_sim[,1], df = 10) # sp500
  # rets2_sim <- qt(U_sim[,2], df = 5) # vix

  # store simulation of log return in matrix
  sim_rets_sp500_copula[ ,t] <- rets1_sim
  sim_rets_vix_copula[ ,t] <- rets2_sim
}

# preview of simulated log returns
head(sim_rets_sp500_copula)
```

```
##                T+1           T+2          T+3           T+4           T+5
## [1,] -0.0007685304  0.0123646766  0.011883305 -0.0136580981  0.0116412057
## [2,]  0.0012828306  0.0132485442  0.002254894  0.0007951660 -0.0002286700
## [3,] -0.0152897173  0.0051491029  0.007970122 -0.0120968034  0.0025567609
## [4,]  0.0200566243  0.0145248803 -0.003288650  0.0233829262  0.0000780408
## [5,] -0.0040661956  0.0004293485 -0.004066821  0.0065378840 -0.0094186044
## [6,] -0.0030992731 -0.0005730132 -0.002523800 -0.0001025489  0.0280233780
```

```r
head(sim_rets_vix_copula)
```

```
##              T+1          T+2          T+3         T+4         T+5
## [1,]  0.01233761  0.009027113 -0.003028593  0.01030640 -0.04496119
## [2,] -0.04056158 -0.070165762 -0.019413450 -0.03163761 -0.06287902
## [3,]  0.08110267 -0.029403877 -0.070004779  0.10688557 -0.05067782
## [4,] -0.08435415 -0.029576430  0.020095418 -0.11508302 -0.02100565
## [5,] -0.05239563 -0.017163781  0.021388893 -0.04255361  0.02882294
## [6,]  0.01667030 -0.034105252  0.032095127  0.04118773 -0.10460650
```

Computing Prices from Returns

See: `code/Utils.R`.

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

# calculate the price and values from the simulated log-returns
```

```
sim_val_mats_copula <- f_logret_to_price(sp_init = spT,
                                         vix_init = vixT,
                                         sim_rets_sp500 = sim_rets_sp500_copula,
                                         sim_rets_vix = sim_rets_vix_copula
                                         )

# unpack matrices
sim_price_sp500_copula <- sim_val_mats_copula$sp500
sim_vol_vix_copula <- sim_val_mats_copula$vix
```

```
# compare simulated returns with the price
head(sim_rets_sp500_copula)
```

```
##                T+1           T+2          T+3           T+4           T+5
## [1,] -0.0007685304  0.0123646766  0.011883305 -0.0136580981  0.0116412057
## [2,]  0.0012828306  0.0132485442  0.002254894  0.0007951660 -0.0002286700
## [3,] -0.0152897173  0.0051491029  0.007970122 -0.0120968034  0.0025567609
## [4,]  0.0200566243  0.0145248803 -0.003288650  0.0233829262  0.0000780408
## [5,] -0.0040661956  0.0004293485 -0.004066821  0.0065378840 -0.0094186044
## [6,] -0.0030992731 -0.0005730132 -0.002523800 -0.0001025489  0.0280233780
```

```
head(sim_price_sp500_copula)
```

```
##        T+1      T+2      T+3      T+4      T+5
## 1 1682.696 1703.631 1723.997 1700.611 1720.523
## 2 1686.152 1708.639 1712.496 1713.859 1713.467
## 3 1658.438 1667.000 1680.339 1660.135 1664.385
## 4 1718.106 1743.244 1737.520 1778.627 1778.766
## 5 1677.156 1677.877 1671.067 1682.028 1666.260
## 6 1678.779 1677.817 1673.588 1673.416 1720.975
```

```
# compare simualted log rets with volatility
head(sim_rets_vix_copula)
```

```
##              T+1          T+2          T+3         T+4         T+5
## [1,]  0.01233761  0.009027113 -0.003028593  0.01030640 -0.04496119
## [2,] -0.04056158 -0.070165762 -0.019413450 -0.03163761 -0.06287902
## [3,]  0.08110267 -0.029403877 -0.070004779  0.10688557 -0.05067782
## [4,] -0.08435415 -0.029576430  0.020095418 -0.11508302 -0.02100565
## [5,] -0.05239563 -0.017163781  0.021388893 -0.04255361  0.02882294
## [6,]  0.01667030 -0.034105252  0.032095127  0.04118773 -0.10460650
```

```
head(sim_vol_vix_copula)
```

```
##         T+1       T+2       T+3       T+4       T+5
## 1 0.1471038 0.1484377 0.1479888 0.1495219 0.1429481
## 2 0.1395243 0.1300701 0.1275693 0.1235965 0.1160642
## 3 0.1575753 0.1530094 0.1426643 0.1587579 0.1509128
## 4 0.1335461 0.1296541 0.1322859 0.1179054 0.1154545
## 5 0.1378829 0.1355365 0.1384667 0.1326981 0.1365785
## 6 0.1477425 0.1427887 0.1474458 0.1536456 0.1383853
```

```
# save data from the simulated values
save(sim_price_sp500_copula, file=here("data_out", "sim_vol_sp500_student_copula.rda"))
save(sim_vol_vix_copula, file=here("data_out", "sim_vol_vix_student_copula.rda"))
```

Pricing the simulation scenarios

Recall the initial (call) options:

1. **1x** strike $K = 1600$ with maturity $T = 20d$
2. **1x** strike $K = 1650$ with maturity $T = 20d$
3. **1x** strike $K = 1750$ with maturity $T = 40d$
4. **1x** strike $K = 1800$ with maturity $T = 40d$

Option Pricing of Simulated Values

Next, we calculate the price of the book of options for the simulated values using the `f_opt_price_simulation()` function under `code/OptionPricing.R`:

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

# Obtain the obtion prices from simulation values
opt_price_mats_copula <- f_opt_price_simulation(sim_price_sp500 = sim_price_sp500_copula,
                                                 sim_vol_vix = sim_vol_vix_copula,
                                                 K_vec = option_book$K,
                                                 T_vec = option_book$T,
                                                 put=FALSE)
```

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

```
##         T+1       T+2       T+3       T+4       T+5
## 1  86.23574 105.37464 124.76889 102.12821 121.03486
## 2  88.78893 109.44086 113.03982 114.23159 113.70201
## 3  66.65944  72.69544  83.12697  66.56864  68.88293
## 4 118.86658 143.51277 137.81035 178.75124 178.87927
## 5  80.50277  80.66897  74.46093  83.78932  69.33574
## 6  82.79798  81.17742  77.44717  77.42645 121.39973
```

```
head(opt_price_mats_copula$opt2)
```

```
##        T+1      T+2      T+3       T+4       T+5
## 1 46.48155 61.70920 78.25204  58.32579  73.95801
## 2 47.71885 63.62631 66.31017  66.82666  65.61893
## 3 33.13371 36.56380 42.86045  31.93575  32.34860
## 4 72.49288 94.81189 89.35858 128.84859 128.94002
## 5 41.17638 40.67486 35.98004  41.96190  31.25854
## 6 43.90957 41.79820 39.06444  39.20650  74.00504
```

```
head(opt_price_mats_copula$opt3)
```

```
##        T+1      T+2       T+3      T+4       T+5
## 1 14.89728 20.95350 27.885097 19.27076 24.212867
## 2 14.16838 18.20225 18.474134 17.57226 15.287382
## 3 11.39332 11.95749 12.593098 10.80279  9.903929
## 4 22.83133 32.08418 29.631045 48.06577 47.138302
## 5 11.64210 10.99219  9.756198 10.67051  7.872894
## 6 14.03689 12.40472 11.964642 12.71741 23.268344
```

```
head(opt_price_mats_copula$opt4)
```

```
##        T+1       T+2       T+3       T+4       T+5
## 1 6.166137  9.282383 13.009175  8.267717 10.484668
## 2 5.561280  7.062379  7.000344  6.317630  4.913499
## 3 4.728855  4.840530  4.803842  4.324934  3.687453
## 4 9.650471 14.351591 13.037869 22.497231 21.488011
## 5 4.337938  3.936011  3.430825  3.624611  2.536209
## 6 5.760424  4.774975  4.645414  5.114203  9.747642
```

Distribution of the Profit and Loss for the Book Of Options

Calculating the profits

For each of the simulated prices and resulting premiums, we want to calculate the profit generated at each simulation timestep. The function used is `f_pl_simulation()`, found under `code/OptionPricing.R`.

```
# Compute the profits and loses for the simulation  from the simulated option premiums
PL_mats_copula <- f_pl_simulation(sim_price_sp500 = sim_price_sp500_copula,
                                  opt_price_mats = opt_price_mats_copula,
                                  K_vec = option_book$K)
```

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

```
##         T+1       T+2      T+3      T+4      T+5
## 1 116.97618 136.90907 149.7426 207.6944 196.4563
## 2 114.42298 132.84285 161.4716 195.5910 203.7892
## 3 136.55247 169.58827 191.3845 243.2539 248.6082
## 4  84.34534  98.77094 136.7011 131.0713 138.6119
## 5 122.70914 161.61474 200.0505 226.0333 248.1554
## 6 120.41393 161.10629 197.0643 232.3961 196.0914
```

```
head(PL_mats_copula$PL2)
```

```
##         T+1       T+2      T+3      T+4      T+5
## 1 106.73036 130.57451 146.2594 201.4968 193.5332
## 2 105.49306 128.65740 158.2013 192.9959 201.8722
## 3 120.07820 155.71991 181.6510 227.8868 235.1426
## 4  80.71903  97.47182 135.1529 130.9740 138.5512
## 5 112.03553 151.60885 188.5314 217.8607 236.2326
## 6 109.30234 150.48551 185.4470 220.6161 193.4861
```

```
head(PL_mats_copula$PL3)
```

```
##        T+1      T+2       T+3      T+4      T+5
## 1 38.31463 71.33021  96.62635 140.5518 143.2783
## 2 39.04353 74.08146 106.03731 142.2503 152.2038
## 3 41.81859 80.32622 111.91835 149.0198 157.5872
## 4 30.38058 60.19953  94.88040 111.7568 120.3529
## 5 41.56981 81.29152 114.75525 149.1521 159.6183
## 6 39.17503 79.87899 112.54680 147.1052 144.2228
```

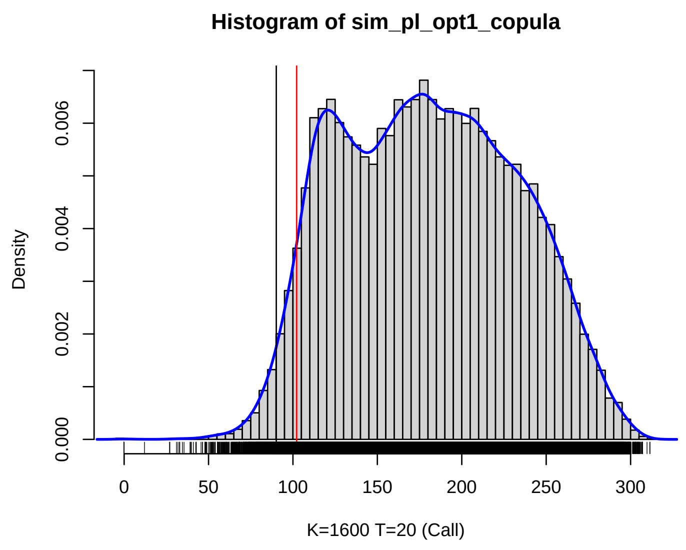
```
head(PL_mats_copula$PL4)
```

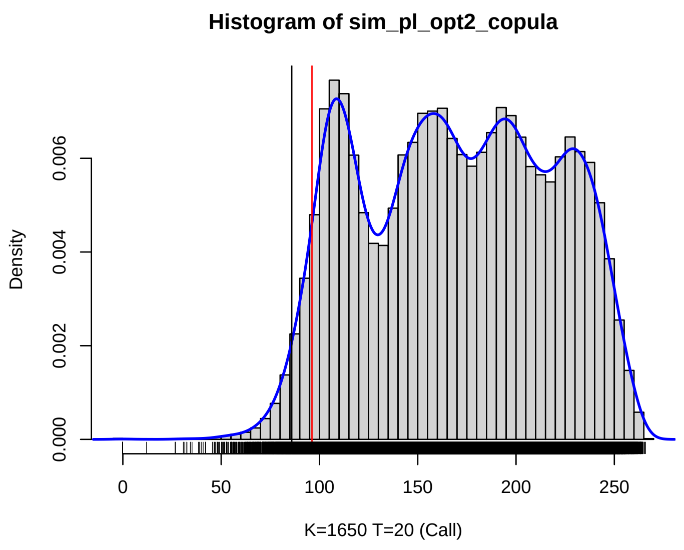
```
##         T+1      T+2      T+3       T+4       T+5
## 1 -2.954223 33.00133 61.50227 101.55487 107.00651
## 2 -2.349366 35.22133 67.51110 103.50495 112.57768
## 3 -1.516941 37.44318 69.70760 105.49765 113.80372
## 4 -6.438557 27.93212 61.47358  87.32535  96.00316
## 5 -1.126024 38.34770 71.08062 106.19797 114.95497
## 6 -2.548510 37.50873 69.86603 104.70838 107.74353
```

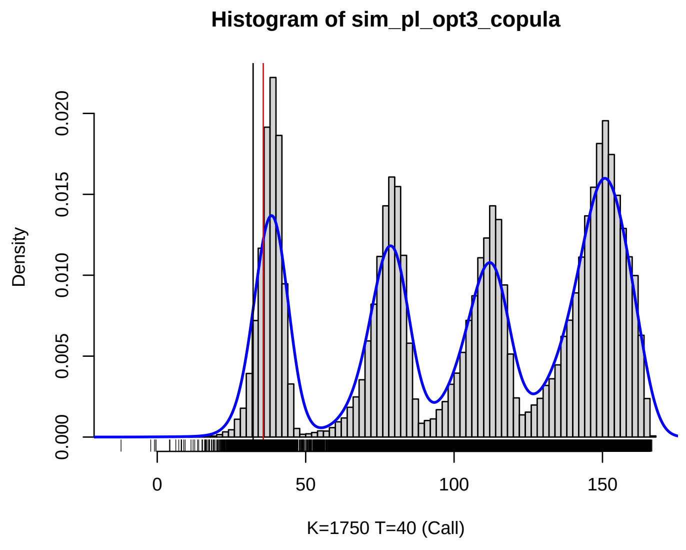
Distribution of Options P/L

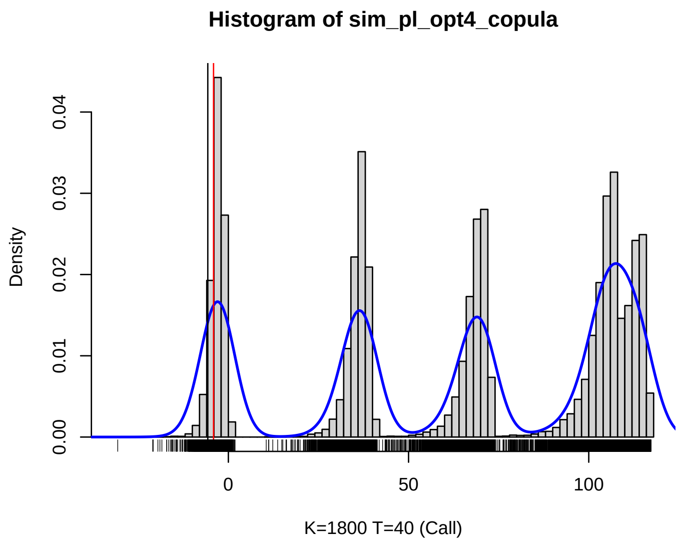
Next, using all the simulated profits and losses for each of the options, we display a histogram for the distribution for each of the options, for the aggregated 5 days of simulation:

```
# flatten the matrices 5-days ahead simulated P/L for the three options
sim_pl_opt1_copula <- as.vector(PL_mats_copula$PL1)
sim_pl_opt2_copula <- as.vector(PL_mats_copula$PL2)
sim_pl_opt3_copula <- as.vector(PL_mats_copula$PL3)
sim_pl_opt4_copula <- as.vector(PL_mats_copula$PL4)

# Compute the 95% VaR and 95% ES
opt1_copula_VaR_ES <- f_VaR_ES(sim_pl_opt1_copula, alpha = 0.05)
opt2_copula_VaR_ES <- f_VaR_ES(sim_pl_opt2_copula, alpha = 0.05)
opt3_copula_VaR_ES <- f_VaR_ES(sim_pl_opt3_copula, alpha = 0.05)
opt4_copula_VaR_ES <- f_VaR_ES(sim_pl_opt4_copula, alpha = 0.05)

# plotting grid
par(mfrow = c(2,2))

# plot the distribution for each of the options
hist(sim_pl_opt1_copula, nclass = round(10 * log(n_sim)),
     probability = TRUE, xlab=paste0("K=", K_vec[1], " T=", T_vec[1], " (Call)"))
lines(density(sim_pl_opt1_copula), lwd=2, col="blue")
abline(v=opt1_copula_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt1_copula_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt1_copula)

# plot the distribution for each of the options
hist(sim_pl_opt2_copula, nclass = round(10 * log(n_sim)),
     probability = TRUE, xlab=paste0("K=", K_vec[2], " T=", T_vec[2], " (Call)"))
lines(density(sim_pl_opt2_copula), lwd=2, col="blue")
abline(v=opt2_copula_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt2_copula_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt2_copula)

# plot the distribution for each of the options
hist(sim_pl_opt3_copula, nclass = round(10 * log(n_sim)),
     probability = TRUE, xlab=paste0("K=", K_vec[3], " T=", T_vec[3], " (Call)"))
lines(density(sim_pl_opt3_copula), lwd=2, col="blue")
abline(v=opt3_copula_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt3_copula_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt3_copula)

# plot the distribution for each of the options
hist(sim_pl_opt4_copula, nclass = round(10 * log(n_sim)),
     probability = TRUE, xlab=paste0("K=", K_vec[4], " T=", T_vec[4], " (Call)"))
lines(density(sim_pl_opt4_copula), lwd=2, col="blue")
abline(v=opt4_copula_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt4_copula_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt4_copula)
```









These all look like multimodal distributions. The last one, particularly shows a different mode for each of the fives days computed. The 95% VaR (red) and ES (black) are all displayed in the plots.

VaR95

```
opt1_copula_VaR_ES$VaR # first option
```

```
## [1] 102.1821
```

```
opt2_copula_VaR_ES$VaR # second doption
```

```
## [1] 96.20371
```

```
opt3_copula_VaR_ES$VaR # third option
```

```
## [1] 35.71444
```

```
opt4_copula_VaR_ES$VaR # fourth option
```

```
## [1] -4.111671
```

ES95

```
# display
opt1_copula_VaR_ES$ES
```

```
## [1] 90.12398
```

```
opt2_copula_VaR_ES$ES
```

```
## [1] 85.90898
```

```
opt3_copula_VaR_ES$ES
```

```
## [1] 32.29534
```

```
opt4_copula_VaR_ES$ES
```

```
## [1] -5.708562
```

Volatility Surface

Steps:

1.Fit a volatility surface to the implied volatilities observed on the market (traded call and put options).Minimize the absolute distance between the market implied volatilities and the model implied volatilities. The parametric surface is given by:

$$\sigma(m,\tau) = \alpha_1 + \alpha_2(m-1)^2 + \alpha_3(m-1)^3 + \alpha_4\sqrt{\tau},$$

where: - $m = K/S$ is the **monyness**. - τ is the time to maturity of the option in years. - $\alpha_1, \dots, \alpha_4$ are model parameters.

2. Re-price the portfolio in one week assuming the same parametric model but shifted by the one-year ATM implied volatility difference.

Note: ATM means at-the-money, which means that $m = 1$. Assume that the one-year ATM implied volatility given by the VIX is $(\alpha_1 + \alpha_4)$.

Traded Options data

First, we do some data preparation with the traded options available. Since for $K > S$ a put option has zero profit (don't want to exercise at a higher price), we discard values with $m > 1$, and similarly for call options with $m < 1$.

```
#initialize the last price of the underlying
S <- sp500[length(sp500)][[1]] #3410
VIX <- as.numeric(vix[length(vix)])

# convert to draframe for easier manipulation
calls_df <- as.data.frame(calls)
puts_df <- as.data.frame(puts)

# assign extra column to puts (1) and calls (0)
calls_df["type"] <- "call"
puts_df["type"] <- "put"

# check dimensions
dim(calls_df)
```

```
## [1] 422   5
```

```
dim(puts_df)
```

```
## [1] 750   5
```

```
# stack both of these matrices together
puts_calls <- rbind(calls_df, puts_df)

# integrate the price
puts_calls["S"] <- rep(S, nrow(puts_calls))
puts_calls["m"] <- puts_calls["K"]/puts_calls["S"]

# filter the calls that have moniness over one
calls_m_over <- puts_calls[(puts_calls["type"] == "call") & (puts_calls["m"] >= 1), ]

# filter the puts that have moniness below one
puts_m_under <- puts_calls[(puts_calls["type"] == "put") & (puts_calls["m"] < 1), ]

# combine these results into putcalls again
call_put_data <- rbind(calls_m_over, puts_m_under)
```

```
head(call_put_data)
```

```
##       K        tau        IV tau_days type       S        m
## 40 1685 0.02557005 0.1163882 6.392513 call 1683.99 1.000600
## 41 1690 0.02557005 0.1152727 6.392513 call 1683.99 1.003569
## 42 1695 0.02557005 0.1133776 6.392513 call 1683.99 1.006538
## 43 1700 0.02557005 0.1114214 6.392513 call 1683.99 1.009507
## 44 1705 0.02557005 0.1054823 6.392513 call 1683.99 1.012476
## 45 1710 0.02557005 0.1105213 6.392513 call 1683.99 1.015445
```

```
tail(call_put_data)
```

```
##         K      tau        IV tau_days type       S         m
## 1159 1550 2.269406 0.1923240 567.3514  put 1683.99 0.9204330
## 1160 1575 2.269406 0.2100174 567.3514  put 1683.99 0.9352787
## 1161 1600 2.269406 0.2033535 567.3514  put 1683.99 0.9501244
## 1162 1625 2.269406 0.1883611 567.3514  put 1683.99 0.9649701
## 1163 1650 2.269406 0.1876401 567.3514  put 1683.99 0.9798158
## 1164 1675 2.269406 0.1799079 567.3514  put 1683.99 0.9946615
```

Fitting the volatility surface

The functions that we will use are implemented under `code/VolatilitySurface.R`. The optimization problem can be written as:

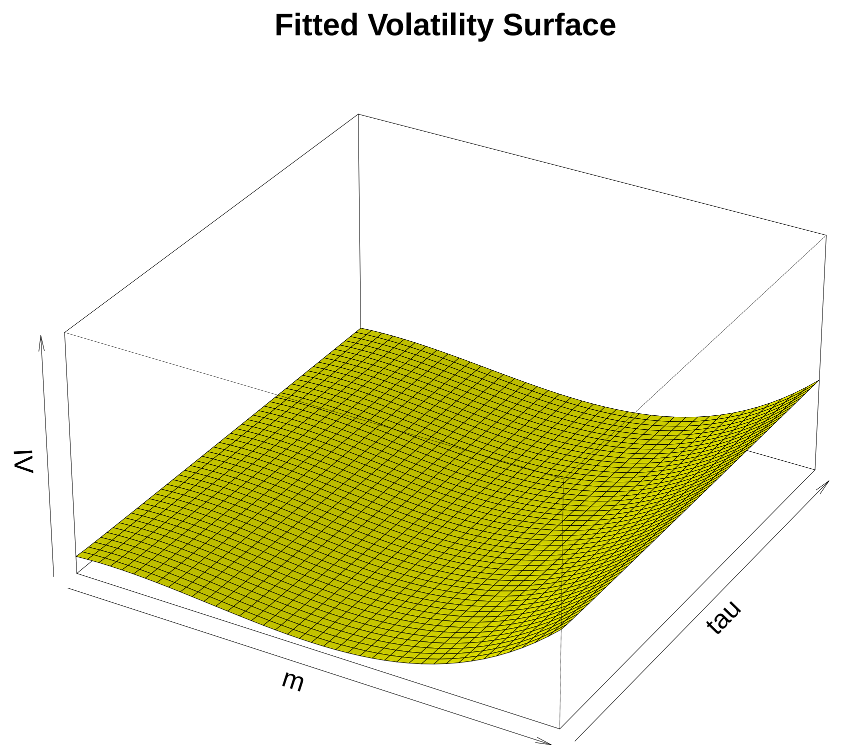
$$\begin{aligned}\vec{\alpha}^* &= \underset{\vec{\alpha}}{\arg\min} \sum_{t=1}^{T} \left|\sigma_t^{observed} - \sigma(m.\tau)\right| \\ &= \underset{\alpha_1,\alpha_2,\alpha_3,\alpha_4}{\arg\min} \sum_{t=1}^{T} \left|\sigma_t^{observed} - \left(\alpha_1 + \alpha_2(m-1)^2 + \alpha_3(m-1)^3 + \alpha_4\sqrt{\tau}\right)\right|\end{aligned}$$

```
# Optimize the objective using available data
alpha <- f_sig_optim(call_put_data)
alpha # best fitted parameters
```

```
## [1]  0.21914594  1.69482614  1.33353414 -0.07907473
```

Volatility Surface Plot

```
#Summary of data
Moneyness <- call_put_data$m
Tau <- call_put_data$tau

#Initialize x and y axis
maturity <- seq(from =0, to = 2.5, by = 0.05)
moneyness <-  seq(from =0, to = 2, by = 0.05)

# memory allocation
IV <- matrix(NA, nrow = length(moneyness), ncol = length(maturity))

# Creating (x,y,z) values for the surface
for(i in 1:length(moneyness)){
  for(j in 1:length(maturity)){
    IV[i,j] <- f_sig_IV(alpha, moneyness[i], maturity[j])
  }
}


# perspective plot
persp(x = moneyness, y = maturity, z = IV,
      xlab = "m",
      ylab = "tau",
      zlab = "IV",
      main = "Fitted Volatility Surface",
      ylim = c(0,2.5), xlim = c(0, 2), zlim = c(0,8),
      theta = 30, phi = 30, expand = 0.5,
      shade=0.4, lwd=0.1, col = "yellow",
      tck = 1, r = 15, d = 0.5
      )
```



Repricing the Portfolio in on week

We are assuming the same parametric model but shifted by the VIX difference. There is a *distance* between the model and the data, we need to keep the same distance and project it forward when repricing. For this we will compute a ratio in change of this distance, and adjust the parameters accordingly.

```
#Assuming we take the mean of the 5th simulated day
VIX_sim <- colMeans(sim_vol_vix_copula)[5]

#Finding the initial difference between model and ATM
#delta of the VIX T = 0
```

```
VIX_model <- alpha[1] + alpha[4]
delta <- VIX - VIX_model # distance between market and model

# Shift the simulated by the distance
VIX_model2 <- VIX_sim - delta # VIX_sim - (VIX - VIX_model)

# Compute ratio chane
delta_ratio <- as.numeric(VIX_model2/VIX_model)
alpha.adjusted <- alpha*c(delta_ratio, 1, 1, delta_ratio)

# display values
print(paste("VIX (last day): ", VIX))
```

```
## [1] "VIX (last day):  0.1453"
```

```
print(paste("VIX model (a1 + a4): ", VIX_model))
```

```
## [1] "VIX model (a1 + a4):  0.140071209012717"
```

```
print(paste("delta (VIX - VIX_model): ", delta))
```

```
## [1] "delta (VIX - VIX_model):  0.00522879098728288"
```

```
print(paste("Shifted VIX (VIX_sim - delta): ", VIX_model2))
```

```
## [1] "Shifted VIX (VIX_sim - delta):  0.139073589258003"
```

```
print(paste("delta_ratio (Shifted VIX / VIX_model): ", delta_ratio))
```

```
## [1] "delta_ratio (Shifted VIX / VIX_model):  0.992877767231782"
```

```
print("alpha (orginal):")
```

```
## [1] "alpha (orginal):"
```

```
print(alpha)
```

```
## [1]  0.21914594  1.69482614  1.33353414 -0.07907473
```

```
print("alpha.adjusted:")
```

```
## [1] "alpha.adjusted:"
```

```
print(alpha.adjusted)
```

```
## [1]  0.21758513  1.69482614  1.33353414 -0.07851154
```

Next, recall that we had the following Call options in our book:

```
option_book
```

```
## $T
## [1] 20 20 40 40
##
## $K
## [1] 1600 1650 1750 1800
```

Using the corresponding volatility, we will reprice them accordingly:

```
# MLE estimator for the last spot price (day 5)
S_T5 = mean(sim_price_sp500_copula[, n_ahead])
S_vec <- rep(S_t, length(option_book$K)) # repeat the value

# pack the underlying price , strikes and maturities
option_book_df <- data.frame(S=S_T5, option_book)
option_book_df$m <- option_book_df$K / option_book_df$S
option_book_df$tau_days <- option_book_df$T - 5 # annualized time to maturity
option_book_df$tau <- option_book_df$tau_days/250 # annualized time to maturity

# compute the IV from the volatility surface (with adjusted alpha)
f_sig_IV_shifted <- function(m, t){f_sig_IV(alpha.adjusted, m, t)}
option_book_df$IV <- mapply(f_sig_IV_shifted, option_book_df$m, option_book_df$tau)
option_book_df
```

```
##          S  T    K         m tau_days  tau        IV
## 1 1685.826 20 1600 0.9490899       15 0.06 0.2025706
## 2 1685.826 20 1650 0.9787490       15 0.06 0.1991064
## 3 1685.826 40 1750 1.0380671       35 0.14 0.1907383
## 4 1685.826 40 1800 1.0677262       35 0.14 0.1963969
```

```
# create wrapper for prc_opt function
prc_opt_wrapper <- function(tau, K, S, IV){
  prc_opt(T = tau, # prc_opt takes maturity in days -> see doc
          K = K, # strike prices
          calls=calls,
          rf_mat = rf_mat,
          price_vec = S, # estimated spot price in 5 days
          vol_vec = IV # recomputed/adjusted volatility in 5 days
          )
}

# reprice the porftolio
option_book_df$repriced_opt <-  mapply(prc_opt_wrapper,
                                   option_book_df$tau_days,
                                   option_book_df$K,
                                   option_book_df$S,
                                   option_book_df$IV)

# display
option_book_df
```

```
##          S  T    K         m tau_days  tau        IV repriced_opt
## 1 1685.826 20 1600 0.9490899       15 0.06 0.2025706     92.02401
## 2 1685.826 20 1650 0.9787490       15 0.06 0.1991064     53.49511
## 3 1685.826 40 1750 1.0380671       35 0.14 0.1907383     23.35043
## 4 1685.826 40 1800 1.0677262       35 0.14 0.1963969     13.01379
```

The `repliced` column provides the reprices portfolio in one week assuming the same fitted parametric model shifted by the one-year ATM implied volatility difference.

Full Approach

1. Filter the volatility clustering of the log-returns of the underlying using a GARCH(1,1) model with Normal innovations. Use the residuals as invariants.
2. Take and AR(1) model for the log-returns of the VIX. Use the residuals as invariants.
3. Use normal marginals for the invariants and a normal copula.
4. Generate draws for the invariants, compute next week (five days) values and reprice the portfolio.
5. Compute the VaR95 and ES95.

Log returns of the underlying

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

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

# remove first return
sp500_rets <- sp500_rets[-1]
vix_rets <- vix_rets[-1]

# remove nas
sp500_rets[is.na(sp500_rets)] <- 0
vix_rets[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
```

GARCH(1,1) Model

Model specification

$$
\begin{aligned}
y_t &= \epsilon_t \sigma_t, \\
\sigma_t^2 &= \omega \ + \ \alpha y_{t-1}^2 + \beta \sigma_{t-1}^2 \\
\epsilon_t &\overset{i.i.d.}{\sim} \mathcal{N}(0,1),
\end{aligned}
$$

Mean and variance

$$\mathbb{E}[Y_t] \approx 0$$
$$\mathbb{V}ar[Y_t] = \mathbb{E}[\epsilon_t^2] = \mathbb{E}[\sigma_t^2] = \frac{\omega}{(1-\alpha-\beta)}$$

Stationarity Conditions

$$\omega \geq 0$$
$$\alpha\ ,\ \beta > 0$$
$$\alpha + \beta < 1 \quad \text{(Covariance-Stationary)}$$

VaR

$$VaR_Y(\alpha) = \Phi^{-1}(1-\gamma)\sigma_t,$$

Log-likelihood

$$\ln L(\theta|\mathbf{y}) = -\frac{T}{2}\ln(2\pi) - \sum_{t=1}^{T}\ln\sigma_t^2 - \frac{1}{2}\sum_{t=1}^{T}\frac{y_t^2}{\sigma_t^2}.$$

Volatility clustering of the log-returns of the underlying with GARCH(1,1)

Which indicates a high level of autocorrelation in the returns.

Fitting the GARCH(1,1)

```
# source code for garch
source(here("code", "GARCH.R")) # GARCH model implementation

# Estimate the GARCH(1,1) model
fit_garch <- f_optim_garch(sp500_rets)
```

```
## Aside: If we had used the MSGARCH package

# load MSGARCH
library("MSGARCH")
```

```
## Warning: package 'MSGARCH' was built under R version 4.2.3
```

```
# GARCH with NOrmal innovations
garch_n <- MSGARCH::CreateSpec(variance.spec = list(model = c("sGARCH")),
                               distribution.spec = list(distribution = c("norm")))
fit_garch_n <- MSGARCH::FitML(spec = garch_n, data = sp500_rets)

#check the fit
summary(fit_garch_n)
```

```
## Specification type: Single-regime
## Specification name: sGARCH_norm
## Number of parameters in variance model: 3
## Number of parameters in distribution: 0
## ------------------------------------------
## Fitted parameters:
##          Estimate Std. Error  t value  Pr(>|t|)
## alpha0_1   0.0000     0.0000   4.7392 1.073e-06
## alpha1_1   0.0859     0.0294   2.9253 1.721e-03
## beta_1     0.9035     0.0038 240.8889    <1e-16
## ---------------------------------------------
## LL: 10660.12
## AIC: -21314.24
## BIC: -21295.8374
## ---------------------------------------------
```

Inpect the parameters

```
# extract parameters (omega, alpha, beta)
theta_hat_garch <- fit_garch$theta_hat
theta_hat_garch
```

```
## [1] 0.000001461511 0.090037405420 0.903175089840
```

Verify stationarity

```
# make sure stationarity is satisfied
sum(theta_hat_garch[2:3])
```

```
## [1] 0.9932125
```

Mean Squared Error

```
# MSE ?
sqrt(theta_hat_garch[1] / (1 - sum(theta_hat_garch[2:3]))) * sqrt(250)
```

```
## [1] 0.2320149
```

```
# sd of returns annualized?
sd(sp500_rets) * sqrt(250)
```

```
## [1] 0.2107013
```

Residuals

The residuals are given by:

$$\hat{\epsilon}_t = \frac{y_t}{\hat{\sigma}_t}$$

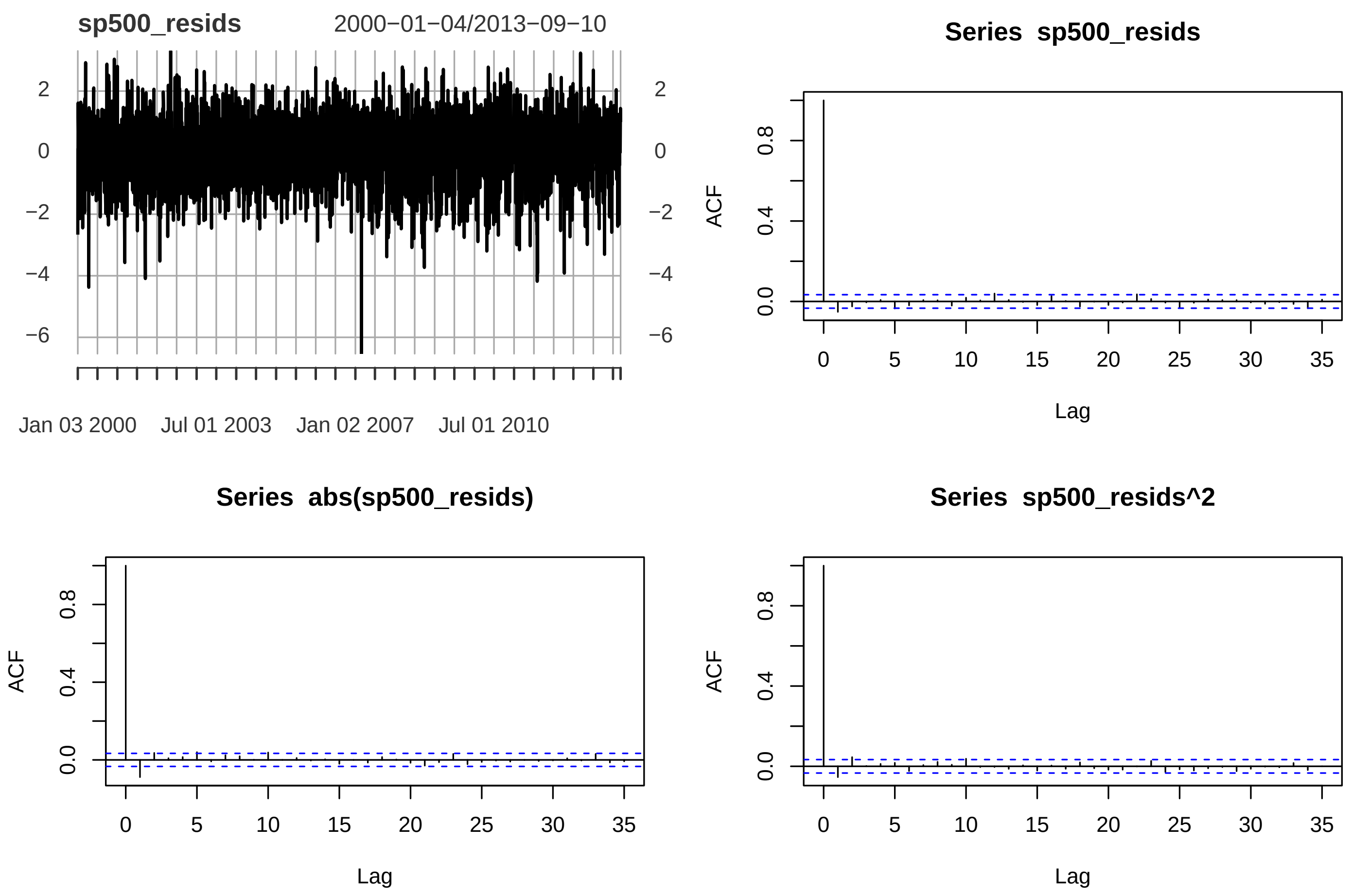
```
# extrct the residuals
sp500_resids <- fit_garch$eps_hat

# inspect their mean and variance
mean(sp500_resids)
```

```
## [1] 0.005801314
```

```
sd(sp500_resids)
```

```
## [1] 0.9908629
```

```r
# Look at dependence in the residuals
par(mfrow = c(2,2))

# Eps_hat = Innovations Series
plot(sp500_resids, pch = 20)

# autocorr of innovations
acf(sp500_resids)

# autocorr of the absolute values
acf(abs(sp500_resids))

# autocorr of the variance of the innovations
acf(sp500_resids^2)
```



Fitting GARCH(1,1) with mean

AR(1) for the log-returns of the VIX

First-order Autoregressive Process AR(1)

- Let $\{\varepsilon_t\}$ be a mean-zero white noise process with variance σ^2.
- Consider a process $\{X_t\}$, independent of $\{\varepsilon_t\}$.
- Let ϕ be constant.

The **AR(1) process** satisfies:

$$X_t = \phi X_{t-1} + \varepsilon_t$$

It can be shown that:

$$\mu_X(t) = \mathbb{E}[X_t] = \phi \mu_X(t-1) = 0 \quad , \forall t$$

when the process is stationary, and the autocovariance function $\gamma_X(h)$ with alg h and autocorrelation $\rho_X(h)$ are given by

$$\gamma_X(h) = \frac{\phi^{|h|}\sigma^2}{1-\phi^2} \quad \text{and} \quad \rho_X(h) = \phi^{|h|}$$

VIX log-returns

```
library("forecast")
# Construct an AR(1) model to the vix
vix_ar1 <- ar(vix_rets, order.max = 1)
vix_ar1$ar # phi coefficient
```

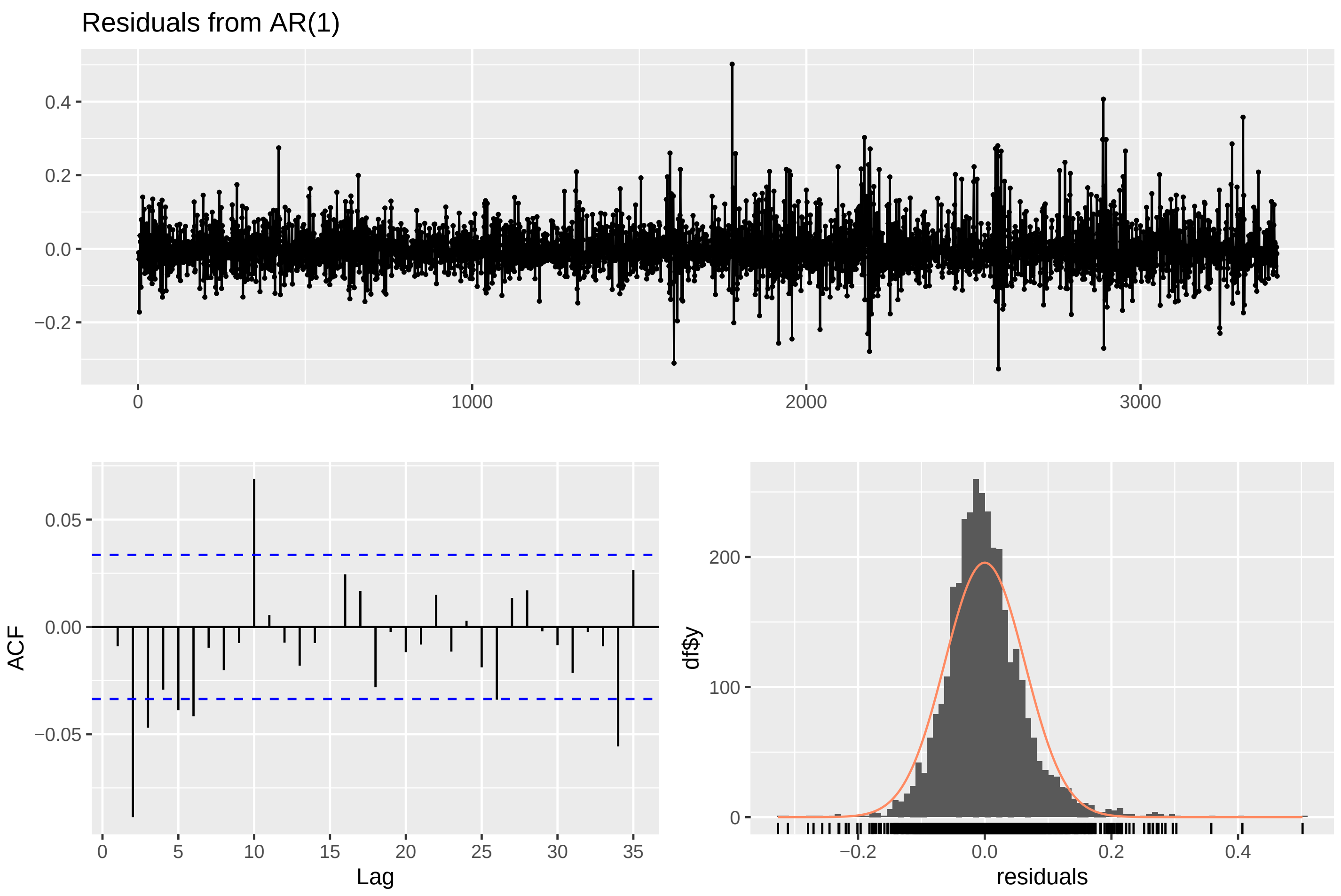
```
## [1] -0.1074941
```

Stationarity of the residuals & underlying normality

```
# extract the residuals
vix_resids <- vix_ar1$resid
vix_resids[1] <- 0 # first residual is NA
head(vix_resids)
```

```
## [1]  0.00000000 -0.01053427 -0.02833403 -0.17206226 -0.01850675  0.03585869
```

```
# comes from the forecast package
checkresiduals(vix_ar1, main="Residuals for AR(1) Model")
```



```
##
##  Ljung-Box test
##
## data:  Residuals from AR(1)
## Q* = 66.683, df = 10, p-value = 0.0000000001929
##
## Model df: 0.   Total lags used: 10
```

Normal Copula with Normal Marginals for the Invariants

Bivariate Gaussian Copula

Recall that the bivariate Gaussian copula is given by:

$$C_\rho^{\text{Gauss}}(u,v) = \Phi_\rho(\Phi^{-1}(u), \Phi^{-1}(v)). \quad \Longleftarrow \quad H(x,y) = C(F(x), G(y))$$

$$C_\rho^{\text{Gauss}}(u,v) = \int_{-\infty}^{\Phi^{-1}(u)} \int_{-\infty}^{\Phi^{-1}(v)} \frac{1}{2\pi\sqrt{1-\rho^2}} \exp\left(-\frac{x^2 - 2\rho xy + y^2}{2(1-\rho^2)}\right) dxdy$$

Gaussian marginals to the invariants

```r
# invariants are the residuals
sp500_resids <- as.vector(sp500_resids)
vix_resids <- as.vector(vix_resids)
```

```
# display some values
head(sp500_resids, 10)
```

```
##  [1] -2.66453978  0.10514445  0.05487059  1.61107285  0.62756665 -0.76350023
##  [7] -0.26044462  0.74944189  0.67144427 -0.44500488
```

```
head(vix_resids, 10)
```

```
##  [1]  0.00000000 -0.01053427 -0.02833403 -0.17206226 -0.01850675  0.03585869
##  [7]  0.01900603 -0.04896233 -0.10447530  0.07897068
```

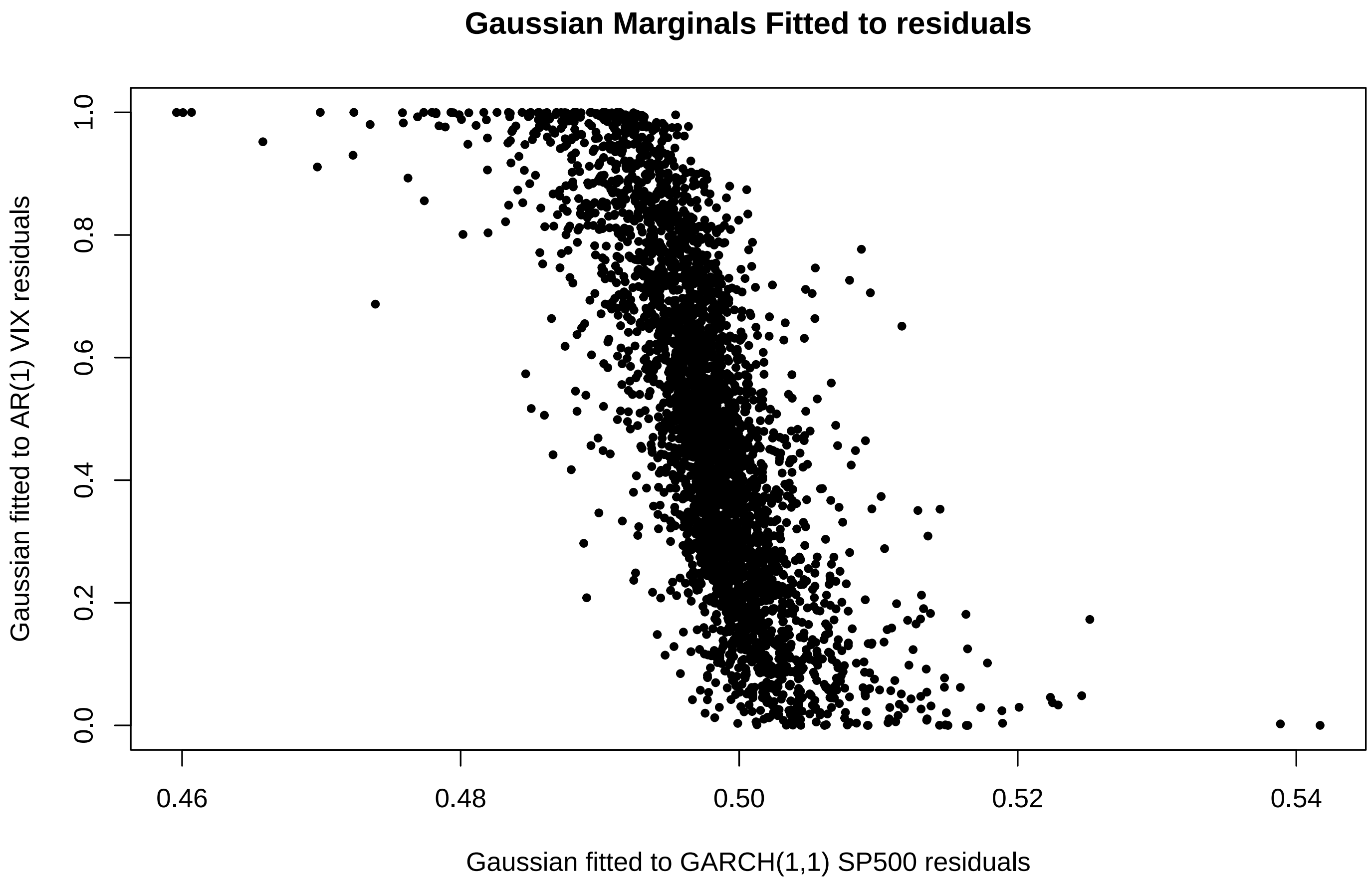
```
library("MASS")
## Fit marginals by MLE

# Gaussian for sp500 invariants (from the GARCH(1,1))
fit1 <- suppressWarnings(
  fitdistr(x = sp500_resids,
           densfun = dnorm,
           start = list(mean = 0, sd = 1))
  )
theta1 <- fit1$estimate #extract fitted parameters

# Gaussian for vix invariants (from the AR(1))
fit2 <- suppressWarnings(
  fitdistr(x = vix_resids,
           densfun = dnorm,
           start = list(mean = 0, sd = 1))
  )
theta2 <- fit2$estimate # extract fitted parameters

# display parameters
theta1
```

```
##        mean          sd
## 0.005801451 0.990717432
```

```
theta2
```

```
##           mean             sd
## -0.00004052605  0.06327442562
```

```
# Fit a Gaussian to the marginals
U1 <- pnorm(sp500_rets_vec, mean = theta1[1], sd = theta1[2]) # sp500
U2 <- pnorm(vix_rets_vec,mean = theta2[1], sd = theta2[2]) # vix
U <- cbind(U1, U2) # join into one matrix
plot(U,
     pch = 20, cex = 0.9,
     main="Gaussian Marginals Fitted to residuals",
     xlab="Gaussian fitted to GARCH(1,1) SP500 residuals",
     ylab="Gaussian fitted to AR(1) VIX residuals"
     )
```



Fitting the Gaussian Copula

```
# Obtain the best rho for the Gaussian Copula
C <- normalCopula(dim = 2)
fit <- fitCopula(C, data = U, method = "ml")
fit
```

```
## Call: fitCopula(C, data = U, ... = pairlist(method = "ml"))
## Fit based on "maximum likelihood" and 3409 2-dimensional observations.
## Copula: normalCopula
##   rho.1
## -0.2006
## The maximized loglikelihood is 4.903
## Optimization converged
```

Simulating the invariants with the Copula

```
# random seed for replication
set.seed(69)

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

# Simulation parameters
```

```
n_sim = 10000 # set number of simulations
n_ahead = 5 # days ahead to produce samples

# preallocate matrices to store simulations
sim_inv_sp500 <- matrix(NA, nrow = n_sim, ncol=5)
sim_inv_vix <- matrix(NA, nrow = n_sim, ncol=5)

# assign days ahead
colnames(sim_inv_sp500) <- c("T+1", "T+2", "T+3", "T+4", "T+5")
colnames(sim_inv_vix) <- c("T+1", "T+2", "T+3", "T+4", "T+5")

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

# perform n_head days of n_sim scenarios
for(t in 1:n_ahead){

  # Sample n_sim scenarios from Gaussian Copula
  U_sim <- rCopula(n_sim, fit@copula)

  # use copula U_sim to reproduce the marginals quantiles F^{-1}(u) with Gaussian distr
  inv1_sim <- qnorm(U_sim[,1], mean = theta1[1], sd = theta1[2]) # sp500
  inv2_sim <- qnorm(U_sim[,2], mean = theta2[1], sd = theta2[2]) # vix
  invs_sim <- cbind(rets1_sim, rets2_sim)

  # store simulation of log return in matrix
  sim_inv_sp500[ ,t] <- inv1_sim
  sim_inv_vix[ ,t] <- inv2_sim
}

# preview of simulated invariants
head(sim_inv_sp500)
```

```
##              T+1        T+2         T+3        T+4        T+5
## [1,]  0.04447154  1.5691517  1.39371665 -1.4835644  0.9565560
## [2,] -0.22933227  0.9195288  0.09356549 -0.2042606 -0.6108582
## [3,] -1.03976298  0.3455542  0.31955037 -0.5236770 -0.1662329
## [4,]  1.51949269  1.4194440 -0.18535447  1.6314713 -0.1877315
## [5,] -0.98740133 -0.1065783 -0.26676884  0.3869440 -0.8275658
## [6,] -0.19608631 -0.3949083  0.02257609  0.4012918  2.1015336
```

```
head(sim_inv_vix)
```

```
##              T+1          T+2         T+3         T+4         T+5
## [1,]  0.02303111  0.055872097  0.03438314 -0.01742967 -0.03425186
## [2,] -0.05770198 -0.065635901 -0.02089352 -0.04514607 -0.09491143
## [3,]  0.08084674 -0.028569376 -0.08021689  0.11747472 -0.06914887
## [4,] -0.06615843 -0.002383062  0.02820027 -0.09207113 -0.03004906
## [5,] -0.09149873 -0.022648918  0.02798077 -0.04514606  0.02429271
## [6,]  0.02318232 -0.053178235  0.04957536  0.07071302 -0.07137518
```

Transforming back the invariants to returns

From GARCH(1,1) residuals to SP500 returns

$$\hat{\epsilon}_t = \frac{y_t}{\hat{\sigma}_t} \implies \hat{y}_t = \hat{\epsilon}_t \hat{\sigma}_t$$

and

$$\begin{cases} \sigma_t^2 = \omega \ + \ \alpha y_{t-1}^2 + \beta \sigma_{t-1}^2 \\ \hat{y_t} = \hat{\epsilon}_t \hat{\sigma}_t \end{cases}$$

$$\begin{cases} \sigma_{T+1}^2 = \omega \ + \ \alpha y_T^2 + \beta \sigma_T^2 \\ y_{\hat{T}+1} = \hat{\epsilon}_{T+1} \hat{\sigma}_{T+1} \\ \vdots \\ \sigma_{T+t}^2 = \omega \ + \ \alpha y_{T+t-1}^2 + \beta \sigma_{T+t-1}^2 \\ y_{\hat{T}+t} = \hat{\epsilon}_{T+t} \cdot \hat{\sigma}_{T+t} \end{cases}$$

First, obtain last conditional variance available (up to time T):

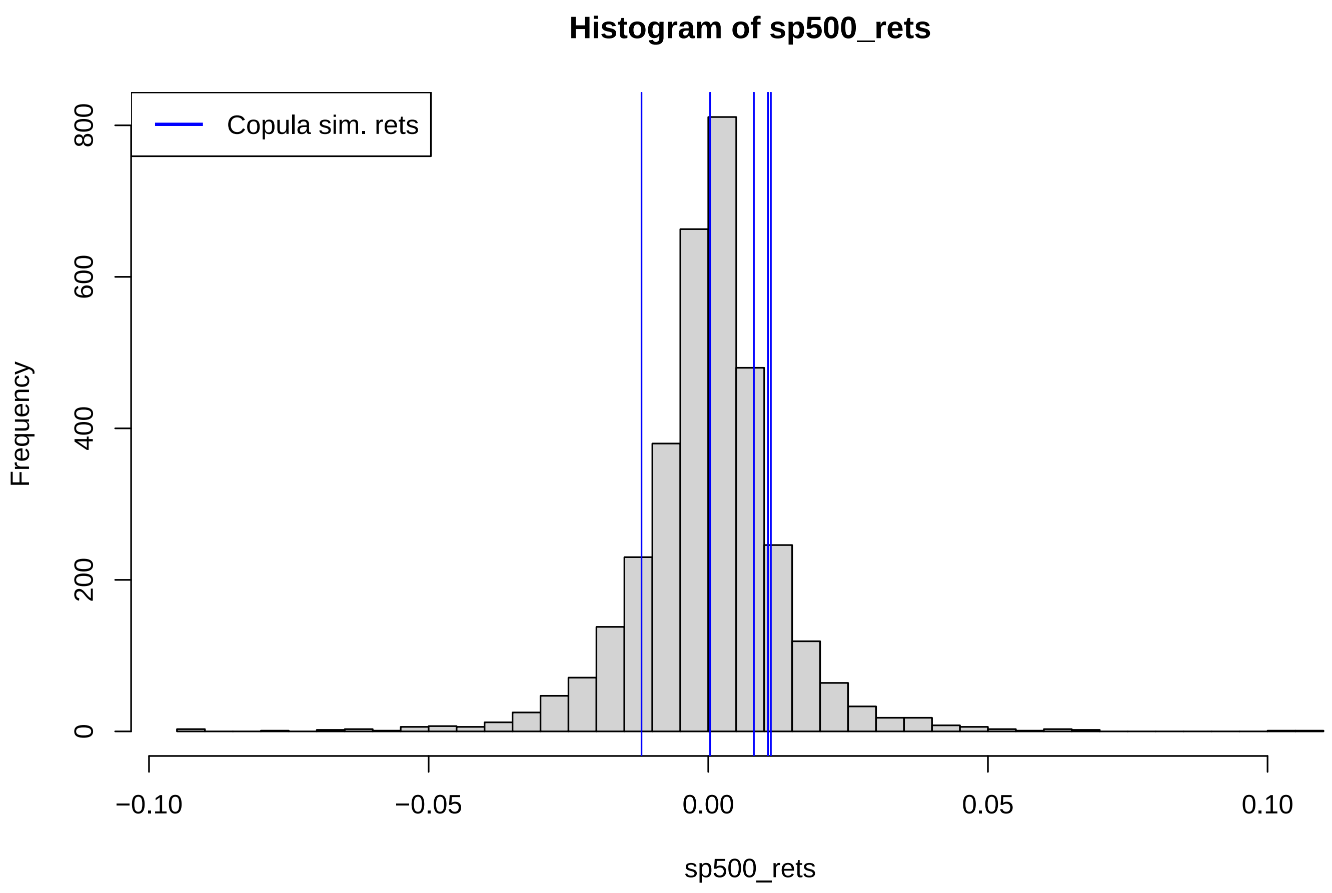
```r
# load source code with GARCh custom functions
source(here("code", "GARCH.R")) # display the pdf through a 3-d chart

# data from up to T
y <- sp500_rets_vec
sig2 <- fit_garch$sig2_hat # vector of sig2 from GARCH
theta <- fit_garch$theta_hat # GARCH parameters

# initial parameters
y_prev <- y[length(y)] # last sp500 observation
sig2_prev <- sig2[length(sig2)] # last sig2_T

# residuals forecasted from copula (invariants)
garch_resids_next <- sim_inv_sp500
resids_next <- garch_resids_next[1, ] # example vector of residuals for prediction

# obtain 5days-ahead prediction for variance
sig2_forecast <- f_forecast_y(theta = theta,
                              sig2_prev = sig2_prev,
                              y_prev = y_prev,
                              resids_next = resids_next)

sig2_forecast
```

```
## $resids_next
##         T+1         T+2         T+3         T+4         T+5
##  0.04447154  1.56915167  1.39371665 -1.48356435  0.95655596
##
## $sig2_next
## [1] 0.00005469191 0.00005086762 0.00005868089 0.00006472350 0.00007274435
##
## $y_next
## [1]  0.0003288847  0.0111914311  0.0106763511 -0.0119354110  0.0081584945
```

```r
# apply to all rows and pack into a matrix
sp500_sim_rets_full <- t(apply(sim_inv_sp500, 1, function(x){f_forecast_y(theta=theta,
                                                                          sig2_prev = sig2_prev,
                                                                          y_prev = y_prev,
                                                                          resids_next = x)$y_next}))
colnames(sp500_sim_rets_full) <- c("T+1", "T+2", "T+3", "T+4", "T+5")
head(sp500_sim_rets_full)
```

```
##                  T+1           T+2           T+3          T+4          T+5
## [1,]  0.0003288847  0.0111914311  0.0106763511 -0.011935411  0.008158495
## [2,] -0.0016960033  0.0065742685  0.0006715923 -0.001415663 -0.004098909
## [3,] -0.0076894607  0.0025900801  0.0023221298 -0.003689649 -0.001145947
## [4,]  0.0112372528  0.0111971934 -0.0015391388  0.013046763 -0.001620878
## [5,] -0.0073022255 -0.0007951261 -0.0019197751  0.002696617 -0.005611657
## [6,] -0.0014501362 -0.0028215149  0.0001568720  0.002694088  0.013752217
```

```
# example 5-days ahead simulation vs actual values:
hist(sp500_rets, nclass=30)
abline(v=sig2_forecast$y_next, col="blue")
legend(x="topleft",
       legend = c("Copula sim. rets"),
       col = c("blue"),
       lwd=rep(2, time=2))
```



From AR(1) residuals to VIX observations

The AR(1) model specifies

$$X_t = \phi X_{t-1} + \varepsilon_t$$

therefore for the step ahead predictions

$$\begin{cases} x_{T+1} = \phi x_T + \varepsilon_T \\ x_{T+2} = \phi x_{T+1} + \varepsilon_{T+1} \\ \vdots \\ x_{T+t} = \phi x_{T+t-1} + \varepsilon_{T+t-1} \end{cases}$$

Transforming the simulated returns into SP500 prices and VIX values

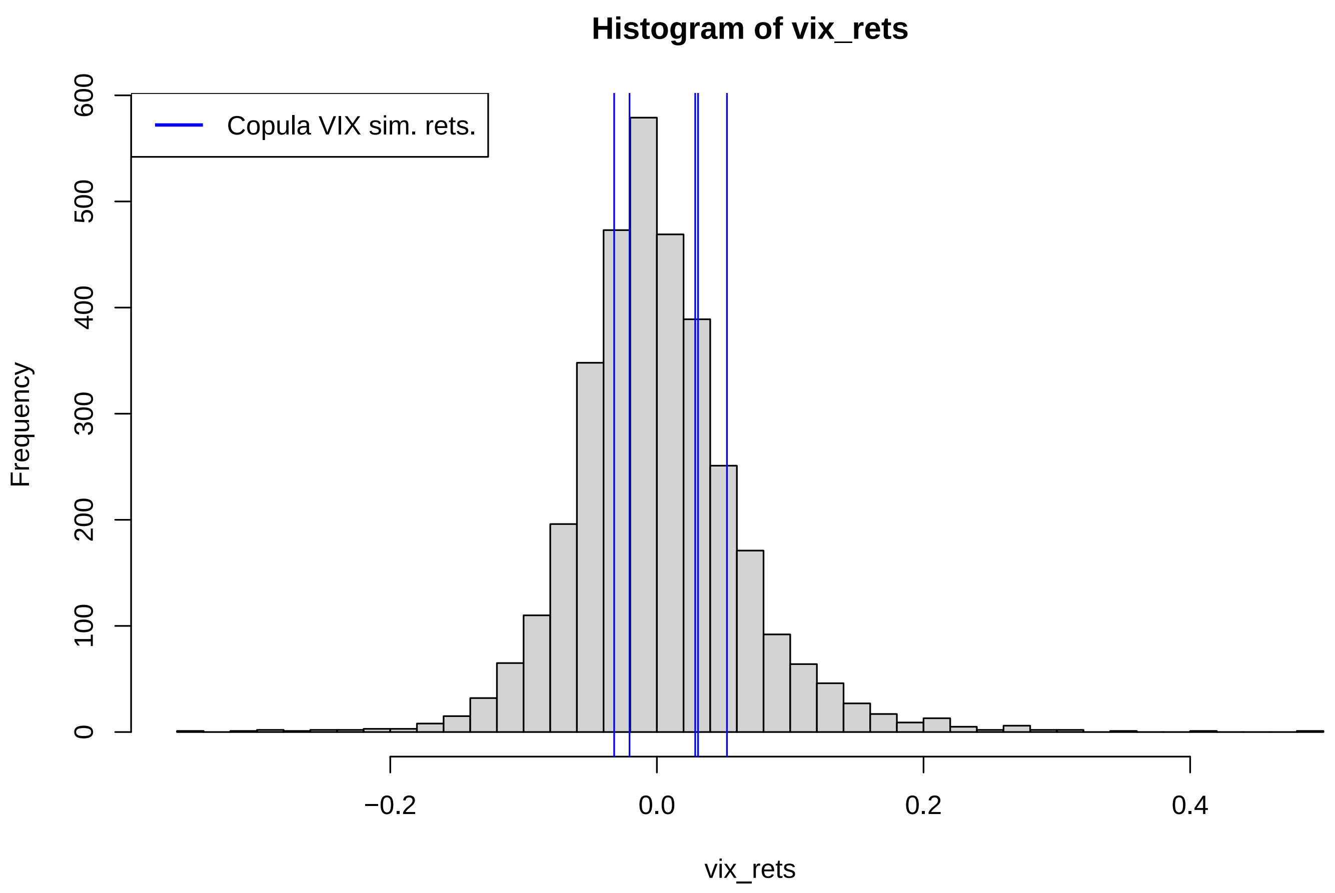
```
# data from up to T (VIX)
x <- vix_rets_vec

# initial parameters
phi <- vix_ar1$ar
x_prev <- x[length(x)] # last vix observation

# residuals forecasted from copula (vix invariants)
ar1_resids_next <- sim_inv_vix
ar1_res_next <- ar1_resids_next[1, ] # example vector

# forecast the vix values using the copula simulated residuals
ex_vix_forecast <- f_forecast_x(phi=phi, x_prev = x_prev, resids_next = ar1_res_next)
ex_vix_forecast
```

```
## [1]  0.03087567  0.05255314  0.02873398 -0.02051841 -0.03204625
```

```
# apply to all rows and pack into a matrix
vix_sim_rets_full <- t(apply(sim_inv_vix, 1, function(x){f_forecast_x(phi=phi,
                                                          x_prev = x_prev,
                                                          resids_next = x)}))
colnames(vix_sim_rets_full) <- c("T+1", "T+2", "T+3", "T+4", "T+5")
head(vix_sim_rets_full)
```

```
##              T+1          T+2         T+3         T+4         T+5
## [1,]  0.03087567  0.052553143  0.02873398 -0.02051841 -0.03204625
## [2,] -0.04985742 -0.060276522 -0.01441415 -0.04359663 -0.09022505
## [3,]  0.08869130 -0.038103171 -0.07612103  0.12565729 -0.08265629
## [4,] -0.05831387  0.003885337  0.02778262 -0.09505760 -0.01983093
## [5,] -0.08365417 -0.013656585  0.02944877 -0.04831163  0.02948593
## [6,]  0.03102688 -0.056513443  0.05565022  0.06473095 -0.07833338
```

```
# example 5-days ahead simulation vs actual values:
hist(vix_rets, nclass=40)
abline(v=ex_vix_forecast, col="blue") # one simulation
legend(x="topleft",
       legend = c("Copula VIX sim. rets."),
       col = c("blue"),
       lwd=rep(2, time=2))
```



Transforming returns back to SP500 prices and VIX values

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

# calculate the price and values from the simulated log-returns
sim_val_mats_full <- f_logret_to_price(sp_init = spT,
                                       vix_init = vixT,
                                       sim_rets_sp500 = sp500_sim_rets_full,
                                       sim_rets_vix = vix_sim_rets_full
                                       )

# unpack matrices
sp500_sim_price_full <- sim_val_mats_full$sp500
vix_sim_vol_full <- sim_val_mats_full$vix
```

```
# compare simulated returns with the price
head(sp500_sim_rets_full)
```

```
##                T+1           T+2           T+3          T+4          T+5
## [1,]  0.0003288847  0.0111914311  0.0106763511 -0.011935411  0.008158495
## [2,] -0.0016960033  0.0065742685  0.0006715923 -0.001415663 -0.004098909
## [3,] -0.0076894607  0.0025900801  0.0023221298 -0.003689649 -0.001145947
## [4,]  0.0112372528  0.0111971934 -0.0015391388  0.013046763 -0.001620878
## [5,] -0.0073022255 -0.0007951261 -0.0019197751  0.002696617 -0.005611657
## [6,] -0.0014501362 -0.0028215149  0.0001568720  0.002694088  0.013752217
```

```
head(sp500_sim_price_full)
```

```
##        T+1      T+2      T+3      T+4      T+5
## 1 1684.544 1703.502 1721.787 1701.359 1715.296
## 2 1681.136 1692.225 1693.362 1690.966 1684.049
## 3 1671.091 1675.425 1679.320 1673.135 1671.219
## 4 1703.020 1722.196 1719.548 1742.129 1739.308
## 5 1671.738 1670.409 1667.205 1671.707 1662.353
## 6 1681.550 1676.812 1677.075 1681.599 1704.885
```

```
# compare simualted log rets with volatility
head(vix_sim_rets_full)
```

```
##              T+1          T+2         T+3         T+4         T+5
## [1,]  0.03087567  0.052553143  0.02873398 -0.02051841 -0.03204625
## [2,] -0.04985742 -0.060276522 -0.01441415 -0.04359663 -0.09022505
## [3,]  0.08869130 -0.038103171 -0.07612103  0.12565729 -0.08265629
## [4,] -0.05831387  0.003885337  0.02778262 -0.09505760 -0.01983093
## [5,] -0.08365417 -0.013656585  0.02944877 -0.04831163  0.02948593
## [6,]  0.03102688 -0.056513443  0.05565022  0.06473095 -0.07833338
```

```
head(vix_sim_vol_full)
```

```
##         T+1       T+2       T+3       T+4       T+5
## 1 0.1498562 0.1579422 0.1625464 0.1592452 0.1542229
## 2 0.1382333 0.1301473 0.1282848 0.1228121 0.1122166
## 3 0.1587756 0.1528396 0.1416370 0.1606013 0.1478604
## 4 0.1370693 0.1376029 0.1414795 0.1286502 0.1261241
## 5 0.1336396 0.1318269 0.1357668 0.1293636 0.1332348
## 6 0.1498789 0.1416436 0.1497496 0.1597636 0.1477264
```

Pricing the simulation scenarios

Recall the initial (call) options:

1. **1x** strike $K = 1600$ with maturity $T = 20d$
2. **1x** strike $K = 1650$ with maturity $T = 20d$
3. **1x** strike $K = 1750$ with maturity $T = 40d$
4. **1x** strike $K = 1800$ with maturity $T = 40d$

Option Pricing of Simulated Values

Same as before, we calculate the price of the book of options for the simulated values using the `f_opt_price_simulation()` function under `code/OptionPricing.R`:

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

# Initialize strikes and maturities the options
T_vec <- c(20, 20, 40,40) #  maturities
K_vec <- c(1600, 1650, 1750, 1800) # Strikes

# Obtain the obtion prices from simulation values
opt_price_mats_full <- f_opt_price_simulation(sim_price_sp500 = sp500_sim_price_full,
                                              sim_vol_vix = vix_sim_vol_full,
                                              K_vec = K_vec,
                                              T_vec = T_vec,
                                              put=FALSE)
```

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

```
##           T+1       T+2       T+3       T+4       T+5
## 1  88.12166 105.77293 123.09876 103.30486 116.20379
## 2  84.11933  93.67377  94.50170  91.84806  84.70578
## 3  77.23641  79.93141  82.12692  77.78522  74.61420
## 4 104.44075 122.86101 120.26090 142.32576 139.47357
## 5  75.30684  73.65459  70.81417  74.04991  65.60863
## 6  85.44634  80.18381  80.73205  85.18549 105.94356
```

```
head(opt_price_mats_full$opt2)
```

```
##         T+1      T+2      T+3      T+4      T+5
## 1 48.21438 62.86143 77.77960 60.12340 70.40484
## 2 43.96031 50.33030 50.47168 47.41020 40.21253
## 3 40.81414 41.84976 41.99636 40.11689 36.13479
## 4 60.25694 76.10114 73.75878 93.33145 90.35781
## 5 36.88126 35.10595 33.09037 34.31082 28.35230
## 6 46.14419 40.93505 41.73635 45.62296 61.01254
```

```
head(opt_price_mats_full$opt3)
```

```
##           T+1       T+2       T+3       T+4       T+5
## 1 15.983934 23.204893 30.708491 21.761400 24.967726
## 2 12.642155 13.364089 12.895659 10.818594  7.086030
## 3 14.435450 13.858234 12.148520 14.003073 10.756745
## 4 18.479334 24.989384 24.464055 30.341565 27.894229
## 5  9.669601  8.768552  8.539126  7.976019  6.702534
## 6 15.193991 11.945160 13.225169 15.976066 19.757453
```

```
head(opt_price_mats_full$opt4)
```

```
##        T+1       T+2       T+3       T+4       T+5
## 1 6.806268 10.943184 15.468670 10.007180 11.464947
## 2 4.804589  4.817726  4.492091  3.407689  1.767350
## 3 6.292417  5.766854  4.570096  5.941050  3.998499
## 4 7.581813 10.925431 10.749666 13.060487 11.447923
## 5 3.364087  2.908896  2.862845  2.468675  2.029383
## 6 6.405588  4.530340  5.307950  6.924876  8.360180
```

Distribution of the Profit and Loss for the Book Of Options

Calculating the profits

For each of the simulated prices and resulting premiums, we want to calculate the profit generated at each simulation timestep. The function used is `f_pl_simulation()`, found under `code/OptionPricing.R`.

```
# Initialize strikes and maturities the options
T_vec <- c(20, 20, 40,40) #  maturities
K_vec <- c(1600, 1650, 1750, 1800) # Strikes

# Compute the profits and loses for the simulation  from the simulated option premiums
PL_mats_full <- f_pl_simulation(sim_price_sp500 = sp500_sim_price_full,
                                opt_price_mats = opt_price_mats_full,
                                K_vec = K_vec)
```

```r
# display profit matrices
head(PL_mats_full$PL1)
```

```
##        T+1      T+2       T+3      T+4      T+5
## 1 43.50442 54.94830  65.58802 152.9507 139.1810
## 2 47.50675 67.04747  94.18508 164.4075 170.6790
## 3 54.38967 80.78982 106.55986 178.4704 180.7705
## 4 27.18534 37.86022  68.42588 113.9298 115.9112
## 5 56.31925 87.06665 117.87261 182.2057 189.7761
## 6 46.17975 80.53742 107.95473 171.0701 149.4412
```

```r
head(PL_mats_full$PL2)
```

```
##        T+1      T+2       T+3      T+4      T+5
## 1 33.41170 47.85980  60.90718 146.1322 134.9799
## 2 37.66578 60.39093  88.21509 158.8454 165.1722
## 3 40.81194 68.87147  96.69042 166.1387 169.2500
## 4 21.36914 34.62009  64.92800 112.9241 115.0269
## 5 44.74483 75.61528 105.59641 171.9448 177.0324
## 6 35.48189 69.78618  96.95043 160.6326 144.3722
```

```r
head(PL_mats_full$PL3)
```

```
##          T+1        T+2       T+3      T+4      T+5
## 1 -15.983934 -12.483660  7.978288 84.49419 80.41702
## 2 -12.642155  -2.642856 25.791120 95.43699 98.29872
## 3 -14.435450  -3.137001 26.538259 92.25251 94.62800
## 4 -18.479334 -14.268151 14.222724 75.91402 77.49052
## 5  -9.669601   1.952681 30.147653 98.27957 98.68221
## 6 -15.193991  -1.223927 25.461610 90.27952 85.62729
```

```r
head(PL_mats_full$PL4)
```

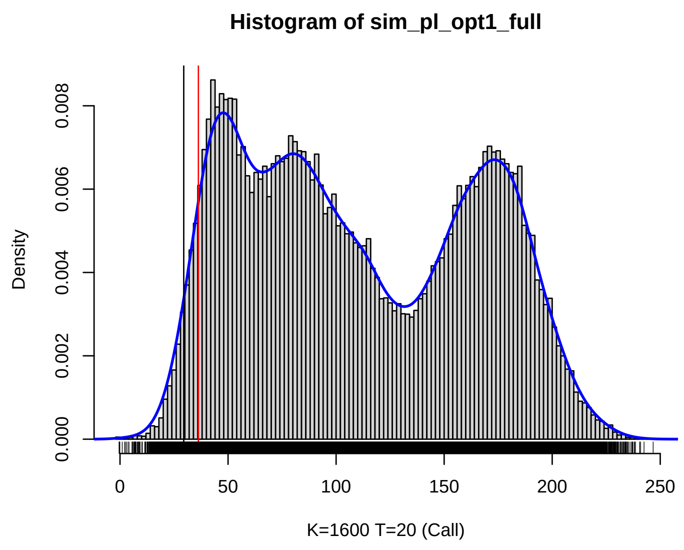
```
##         T+1        T+2        T+3      T+4      T+5
## 1 -6.806268 -10.943184 -15.468670 46.24841 43.91980
## 2 -4.804589  -4.817726  -4.492091 52.84790 53.61740
## 3 -6.292417  -5.766854  -4.570096 50.31454 51.38625
## 4 -7.581813 -10.925431 -10.749666 43.19510 43.93682
## 5 -3.364087  -2.908896  -2.862845 53.78691 53.35536
## 6 -6.405588  -4.530340  -5.307950 49.33071 47.02457
```

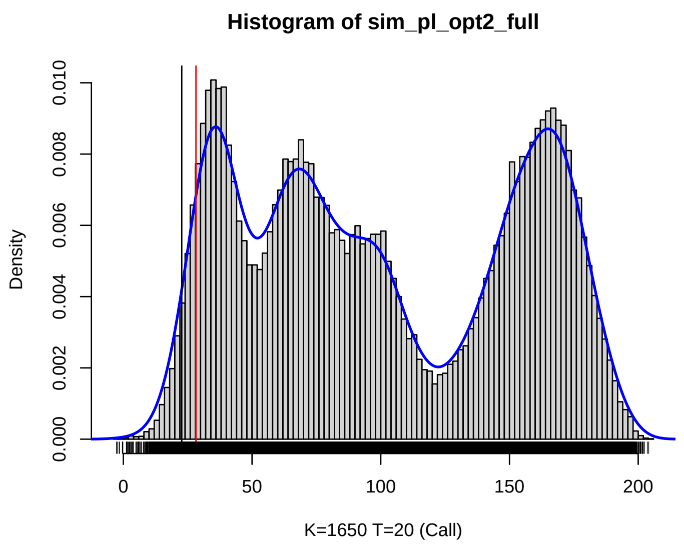
Distribution of Options P/L

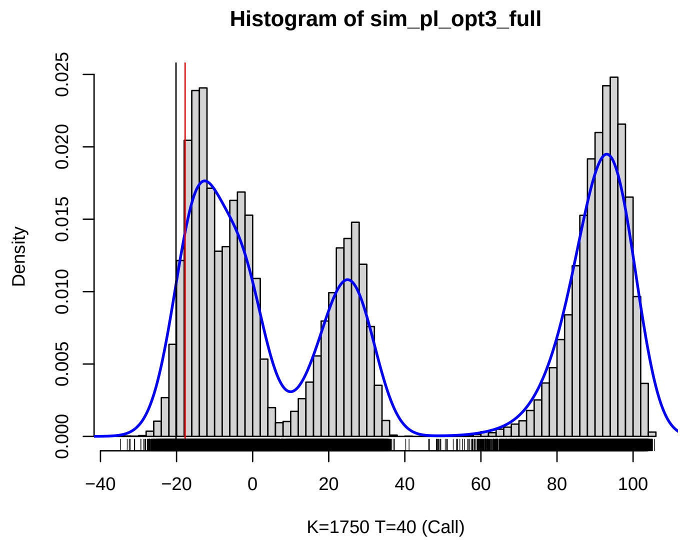
Next, using all the simulated profits and losses for each of the options, we display a histogram for the distribution for each of the options, for the aggregated 5 days of simulation:

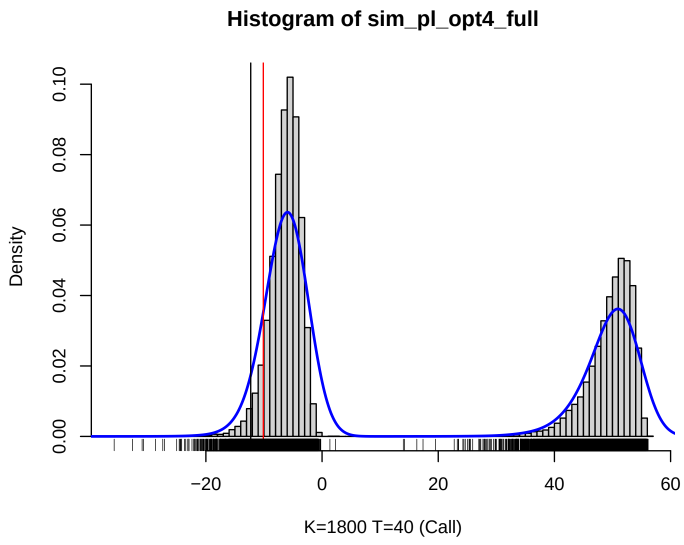
```r
# flatten the matrices 5-days ahead simulated P/L for the three options
sim_pl_opt1_full <- as.vector(PL_mats_full$PL1)
sim_pl_opt2_full <- as.vector(PL_mats_full$PL2)
sim_pl_opt3_full <- as.vector(PL_mats_full$PL3)
sim_pl_opt4_full <- as.vector(PL_mats_full$PL4)

# Compute the 95% VaR and 95% ES
opt1_full_VaR_ES <- f_VaR_ES(sim_pl_opt1_full, alpha = 0.05)
opt2_full_VaR_ES <- f_VaR_ES(sim_pl_opt2_full, alpha = 0.05)
opt3_full_VaR_ES <- f_VaR_ES(sim_pl_opt3_full, alpha = 0.05)
opt4_full_VaR_ES <- f_VaR_ES(sim_pl_opt4_full, alpha = 0.05)
```

```r
# plot the distribution for each of the options
par(mfrow = c(2,2))

# distribution of first option
hist(sim_pl_opt1_full, nclass = round(10 * log(n_sim)),
     probability = TRUE, xlab=paste0("K=", K_vec[1], " T=", T_vec[1], " (Call)"))
lines(density(sim_pl_opt1_full), lwd=2, col="blue")
abline(v=opt1_full_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt1_full_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt1_full)

# distribution of second option
hist(sim_pl_opt2_full, nclass = round(10 * log(n_sim)),
     probability = TRUE, xlab=paste0("K=", K_vec[2], " T=", T_vec[2], " (Call)"))
lines(density(sim_pl_opt2_full), lwd=2, col="blue")
abline(v=opt2_full_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt2_full_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt2_full)

# distribution of third option
hist(sim_pl_opt3_full, nclass = round(10 * log(n_sim)),
     probability = TRUE, xlab=paste0("K=", K_vec[3], " T=", T_vec[3], " (Call)"))
lines(density(sim_pl_opt3_full), lwd=2, col="blue")
abline(v=opt3_full_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt3_full_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt3_full)

# distribution of fourth option
hist(sim_pl_opt4_full, nclass = round(10 * log(n_sim)),
     probability = TRUE, xlab=paste0("K=", K_vec[4], " T=", T_vec[4], " (Call)"))
lines(density(sim_pl_opt4_full), lwd=2, col="blue")
abline(v=opt4_full_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt4_full_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt4_full)
```









VaR95

```
opt1_full_VaR_ES$VaR # first option
```

```
## [1] 36.26838
```

```
opt2_full_VaR_ES$VaR # second doption
```

```
## [1] 28.21012
```

```
opt3_full_VaR_ES$VaR # third option
```

```
## [1] -17.74786
```

```
opt4_full_VaR_ES$VaR # fourth option
```

```
## [1] -10.11307
```

ES95

Expected shortfall is calculated by averaging all of the returns in the distribution that are worse than the VAR of the portfolio at a given level of confidence.

```r
# display
opt1_full_VaR_ES$ES
```

```
## [1] 29.49193
```

```r
opt2_full_VaR_ES$ES
```

```
## [1] 22.69627
```

```r
opt3_full_VaR_ES$ES
```

```
## [1] -20.14902
```

```r
opt4_full_VaR_ES$ES
```

```
## [1] -12.28245
```