Statistical Methods in Financial Engineering - Risk Management Project

Chloe Morin-Leclerc, Felix-Antoine Groulx, Denis Genest, Thien Duy Tran

December 12, 2019

# Part I: Project Guidelines

## Context

You work as a quantitative analyst for a large investment bank. You and your team are responsible for challenging the models used by traders and risk managers. You work with R and love reproducible research. All your files are written in Rmarkdown or R notebook.

You can watch the introductory video on Rmarkdown to help you build properly the R file. Moreover, you use GitHub with your team. You’ll build a dedicated project for the tasks below and use RStudio with GitHub extensively. You also use Loom to share your findings with other teams in the investment bank.

## Learning Objectives

1. Content (scientific rigor, concepts, creativity).
2. Choose the right tools.
3. Implement the steps correctly.
4. Come up with innovative solutions.

## Form: Coding, Collaboration and Presentation

1. Build RStudio project with proper folder structure and Rmarkdown/nootebook file to reproduce your results.
2. Program with state-of-the-art coding standards.
3. Use GitHub repository for collaborative research.
4. Use Loom video for presenting your results.

## Objective

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

# Part II: Data

## Loading the Data

The first step is to load the database ‘Market’. ‘Market’ is a list of 5 elements: S&P500 index prices, VIX values, the term structure of interest rates, and traded call and put options information. To make sure this code can run on any platform, we use the library ‘here’.

# install.packages("here")  
library("here")

## here() starts at C:/Users/TTD/Documents/GitHub/Method\_Stat

# Load the data  
load(file = here("Data", "Market.rda"))  
  
# Load the functions  
source(file = here("Functions", "price\_call.r")) # Prices calls using the Black-Scholes formula  
source(file = here("Functions", "lin\_inter.r")) # Linear interpolation of the interest rates  
  
# Assign data to different variables  
vix <- as.vector(Market$vix)  
sp\_500 <- as.vector(Market$sp500)  
calls <- as.vector(Market$calls)  
puts <- as.vector(Market$puts)  
  
# Create a matrix 'rates' with two columns: maturities and risk-free rates  
rates <- matrix(data = NA, nrow = length(Market$rf), ncol = 2)  
rates[,1] <- as.numeric((attributes(Market$rf))[[2]])  
rates[,2] <- Market$rf  
  
# Assign column names to 'rates'  
colnames(rates) <- c("Maturities", "Risk-free Rates")

# Part III: Pricing a Portfolio of Options

The portfolio under consideration contains four options: K = 1600 and T-t = 20 days, K = 1650 and T-t = 20 days, K = 1750 and T-t = 40 days, and K = 1800 and T-t = 40 days. We assume that there is 250 days in a year and thus convert times to expiry in years by dividing by 250. We first create a matrix ‘book’ that contains information regarding the options in the portfolio.

# Create matrix  
book\_1 <- matrix(data = NA, nrow = 4, ncol = 4)  
  
# Assign names to columns  
colnames(book\_1) <- c("Quantity", "Call", "StrikePrice", "Maturity")  
  
# Store initial values  
book\_1[1,] <- c(1, 1, 1600, 20 / 250)  
book\_1[2,] <- c(1, 1, 1650, 20 / 250)  
book\_1[3,] <- c(1, 1, 1750, 40 / 250)  
book\_1[4,] <- c(1, 1, 1800, 40 / 250)

The next step is to use the most recent underlying asset price and VIX value as well as the interpolated risk-free rates to compute the value of each option in the portfolio that we store in the variable ‘call\_price’. The value of the portfolio of options is simply the sum of each option’s value. To do this, we use two functions described below.

1. ‘lin\_inter’: Takes as input a maturity in years (360-day basis) and outputs the associated rate. Uses linear interpolation with the given term structure.
2. ‘price\_call’: Takes as inputs the underlying asset price, the option strike price, the risk-free rate, the volatility, and the maturity. Uses Black-Scholes model to price call options.

# Number of observations  
n\_obs <- length(sp\_500)  
  
# Convert 250-day basis year in 360-day basis year  
m\_1 <- unique(book\_1[,4])[1] \* (250 / 360)  
m\_2 <- unique(book\_1[,4])[2] \* (250 / 360)  
  
# Interpolated interest rates  
r\_1 <- lin\_inter(m\_1)  
r\_2 <- lin\_inter(m\_2)  
  
# Latest underlying asset price (spot price)  
S\_1 <- sp\_500[n\_obs]  
  
# Latest VIX value  
vol\_1 <- vix[n\_obs]  
  
# Strike prices  
K <- book\_1[,3]  
  
# Maturities  
M <- book\_1[,4]  
  
# Initialize a vector to store call prices  
call\_price <- c(rep(NA, 4))  
  
# Compute the option values and store them in 'call\_price'  
call\_price[1] <- price\_call(S\_1, K[1], r\_1, vol\_1, M[1])  
call\_price[2] <- price\_call(S\_1, K[2], r\_1, vol\_1, M[2])  
call\_price[3] <- price\_call(S\_1, K[3], r\_2, vol\_1, M[3])  
call\_price[4] <- price\_call(S\_1, K[4], r\_2, vol\_1, M[4])  
  
# Compute the price of the portfolio and store it in 'PF\_price\_1'  
PF\_price\_1 <- sum(call\_price)

## The value of the portfolio of European call options is 156.21$.

## The options are worth respectively 87.31$, 47.52$, 15.10$, 6.28$.

It is not surprising to see that the value of the first option is higher than the value of the second option. Indeed, since the strike price is lower for the first option, the first option is deeper ITM than the second option and it should therefore have a higher price. The same reasoning applies to the third and fourth options.

# Part IV: One Risk Driver and Gaussian Model

We now want to estimate the value-at-risk (VaR) and the expected shortfall (ES) of this portfolio of options over the course of the following week. To do so, we must first compute the underlying asset log returns.

# Daily underlying asset log returns  
log\_return <- diff(log(sp\_500))

Next, we assume that the underlying asset log returns follow a normal distribution. The normal distribution parameters can be found by computing the empirical mean and standard deviation of the underlying asset daily log returns.

# Number of log returns  
n\_ret <- length(log\_return)  
  
# Calibration  
mu\_hat <- mean(log\_return)  
s2\_hat <- mean((log\_return - mu\_hat)^2) \* (n\_ret / (n\_ret - 1))  
sd\_hat <- s2\_hat^0.5  
  
# Store parameters in 'theta\_1'  
theta\_1 <- c(mu\_hat, sd\_hat)

Now that we have estimated the parameters of the underlying asset return log returns, we can run a simulation of the underlying asset price one week ahead by generating normally distributed IID shocks with mean ‘mu\_hat’ and standard deviation ‘sd\_hat’. Since we found the distribution parameters using daily log returns, we generate five shocks per simulation and sum these shocks to obtain the overall shock over one week. We fix the size of the simulation to 10 000.

# Number of simulation  
H <- 10000  
  
# Number of days between now and the forecast horizon  
t <- 5  
  
# Set seed for generating pseudo-random numbers  
set.seed(4321)  
  
# Store in 'sim\_ret\_1' normally distributed IID shocks with mean 'mu\_hat' and standard deviation 'sd\_hat'  
sim\_ret\_1 <- matrix(rnorm(t \* H, mean = theta\_1[1], sd = theta\_1[2]), nrow = H, ncol = t)

In order to obtain the underlying asset price one week from now, we simply have to compute the exponential sum of the five daily normally distributed IID shocks per trajectory and multiply by the latest underlying asset price. The last thing we need to modify is the risk-free rate for the remaining life of the option. The strike price does not change over the option life and the volatility is assumed to be constant.

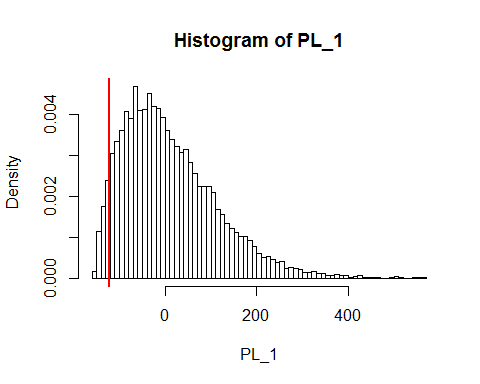
# Compute the price of the underlying asset one week from now  
S\_2 <- S\_1 \* exp(rowSums(sim\_ret\_1))  
  
# Update the risk-free rate (360-day basis year)  
r\_3 <- lin\_inter(m\_1 - t / 360)  
r\_4 <- lin\_inter(m\_2 - t / 360)  
  
# Initialize a matrix to store call prices (10 000 rows (1 per simulation), 4 columns (1 per option))  
call\_price\_2 <- matrix(NA, nrow = H, ncol = 4)  
  
# Loop through 10 000 simulations and price each call option  
for (i in 1:10000){  
 call\_price\_2[i,1] <- price\_call(S\_2[i], K[1], r\_3, vol\_1, M[1] - t / 250)  
 call\_price\_2[i,2] <- price\_call(S\_2[i], K[2], r\_3, vol\_1, M[2] - t / 250)  
 call\_price\_2[i,3] <- price\_call(S\_2[i], K[3], r\_4, vol\_1, M[3] - t / 250)  
 call\_price\_2[i,4] <- price\_call(S\_2[i], K[4], r\_4, vol\_1, M[4] - t / 250)  
}

For each replication, we can compute the value of the portfolio of options by summing the value of the four call options. Recall that we want to estimate the VaR and the ES of this portfolio of options over the course of the following week. Therefore, for each replication, we must compute a P&L by discounting at the risk-free rate the value of the portfolio of options one week from now and subtracting the value of the portfolio observed today.

# Compute the price of the portfolio for each replication and store it in 'PF\_price\_2'  
PF\_price\_2 <- rowSums(call\_price\_2)  
  
# Compute the risk-free rate for a period of 5 days (360-day basis year)  
r\_PL <- lin\_inter(t / 360)  
  
# Compute the P&L  
PL\_1 <- PF\_price\_2 \* exp(-(t / 360) \* r\_PL) - PF\_price\_1

The last step is to compute the VaR and the ES of the portfolio of options P&L distribution. In order to do so, we sort the P&L values in ascending order and identify the (1-alpha)-quartile of the distribution for the VaR, where alpha is the risk level (in our case, 0.95). The ES is simply the mean of the P&L values smaller than the VaR.

# Set alpha to a desired significance level  
alpha <- 0.95  
  
# Compute the VaR and the ES of the P&L distribution  
VaR\_1 <- sort(PL\_1)[(1 - alpha) \* H]  
ES\_1 <- mean(sort(PL\_1)[1:((1 - alpha) \* H)])  
  
# Plot an histogram  
hist(PL\_1, nclass = round(10 \* log(length(PL\_1))), probability = TRUE)  
  
# Add a vertical line to show the VaR  
abline(v = quantile(PL\_1, probs = (1 - alpha)),  
 lty = 1,  
 lwd = 2.5,  
 col = "red")



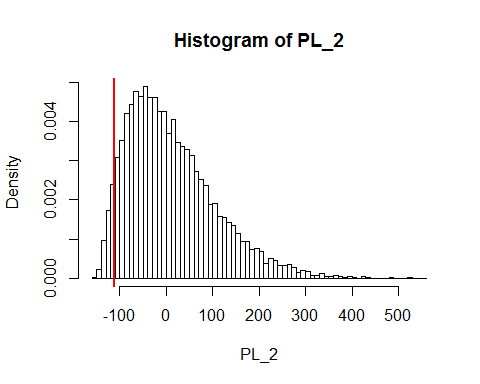
## The value at risk at alpha = 0.95 is -121.67$.

## The expected shortfall at alpha = 0.95 is -134.17$.

[Discussion on the result (talk about asymmetry of call option payoffs) and the assumption (normal distribution, volatility is constant)]

# Part V: Two risk drivers and Gaussian model

# install.packages("MASS")  
library("MASS")  
  
# Daily VIX log returns  
vix\_return <- diff(log(vix))  
  
# Store underlying asset log returns and VIX log returns in 'rets'  
rets <- matrix(NA, nrow = n\_ret, ncol = 2)  
rets[,1] <- log\_return  
rets[,2] <- vix\_return  
  
# Calibration  
mu\_hat <- colMeans(rets)  
sg\_hat <- ((n\_ret - 1) / n\_ret) \* cov(rets)  
  
# Store parameters in 'theta\_2'  
theta\_2 <- list(mu = mu\_hat, sigma = sg\_hat)  
  
# Initialize the array 'sim\_ret\_2'  
sim\_ret\_2 <- array(data = NA, dim = c(H, 2, t))  
  
# Set seed for generating pseudo-random numbers  
set.seed(4321)  
  
# Store in 'sim\_ret\_2' daily shocks from a multivariate normal distribution with parameters 'theta\_2'  
for (i in 1:t) {  
 sim\_ret\_2[,,i] <- mvrnorm(n = H, mu = theta\_2$mu, Sigma = theta\_2$sigma)  
}  
  
# Initialize two vectors that contain the underlying asset price and the VIX value one week from now  
S\_3 <- rep(NA, H)  
vol\_3 <- rep(NA, H)  
  
# Compute the underlying asset price and the VIX value one week from now  
for (i in 1:H) {  
 S\_3[i] <- S\_1 \* exp(sum(sim\_ret\_2[i,1,]))  
 vol\_3[i] <- vol\_1 \* exp(sum(sim\_ret\_2[i,2,]))  
}  
  
# Initialize a matrix to store call prices (10 000 rows (1 per simulation), 4 columns (1 per option))  
call\_price\_3 <- matrix(NA, nrow = H, ncol = 4)  
  
# Loop through 10 000 simulations and price each call option  
for (i in 1:10000){  
 call\_price\_3[i,1] <- price\_call(S\_3[i], K[1], r\_3, vol\_3[i], M[1] - t / 250)  
 call\_price\_3[i,2] <- price\_call(S\_3[i], K[2], r\_3, vol\_3[i], M[2] - t / 250)  
 call\_price\_3[i,3] <- price\_call(S\_3[i], K[3], r\_4, vol\_3[i], M[3] - t / 250)  
 call\_price\_3[i,4] <- price\_call(S\_3[i], K[4], r\_4, vol\_3[i], M[4] - t / 250)  
}  
  
# Compute the price of the portfolio for each replication and store it in 'PF\_price\_3'  
PF\_price\_3 <- rowSums(call\_price\_3)  
  
# Compute the P&L  
PL\_2 <- PF\_price\_3 \* exp(-(t / 360) \* r\_PL) - PF\_price\_1  
  
# Compute the VaR and the ES of the P&L distribution  
VaR\_2 <- sort(PL\_2)[(1 - alpha) \* H]  
ES\_2 <- mean(sort(PL\_2)[1:((1 - alpha) \* H)])  
  
# Plot an histogram  
hist(PL\_2, nclass = round(10 \* log(length(PL\_2))), probability = TRUE)  
  
# Add a vertical line to show the VaR  
abline(v = quantile(PL\_2, probs = (1 - alpha)),  
 lty = 1,  
 lwd = 2.5,  
 col = "red")



## The value at risk at alpha = 0.95 is -111.15$.

## The expected shortfall at alpha = 0.95 is -123.77$.

[Discussion on the result (VaR is smaller because the underlying asset and the VIX are negatively correlated; when the value of the underlying asset drops, the volatility tends to rise and offsets the effect of the underlying asset price on the call option price) and the assumption (normal distribution for log returns)]

# Part VI: Two risk drivers and copula-marginal model

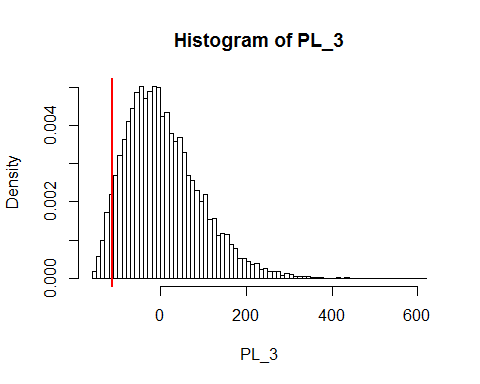
# install.packages("copula")  
library("copula")  
  
# install.packages("fGarch")  
library("fGarch")

## Loading required package: timeDate

## Loading required package: timeSeries

## Loading required package: fBasics

library("timeDate")  
library("timeSeries")  
library("fBasics")  
  
# Load the function  
source(file = here("Functions", "nll\_student.r")) # Computes the negative log-likelihood function  
  
# Define an initial vector of parameters for the underlying asset log returns  
theta\_0 <- c(mean(log\_return), sd(log\_return), 10)  
  
# Calibrate the student-t distribution on the underlying asset log returns  
tmp <- optim(par = theta\_0,  
 fn = nll\_student,  
 method = "L-BFGS-B",  
 lower = c(-Inf, 1e-5, 10),  
 x = log\_return)  
  
# Store parameters in 'theta\_3'  
theta\_3 <- tmp$par  
  
# Define an initial vector of parameters for the VIX log returns  
theta\_0 <- c(mean(vix\_return), sd(vix\_return), 5)  
  
# Calibrate the student-t distribution on the VIX log returns  
tmp <- optim(par = theta\_0,  
 fn = nll\_student,  
 method = "L-BFGS-B",  
 lower = c(-Inf, 1e-5, 5),  
 x = vix\_return)  
  
# Store parameters in 'theta\_3'  
theta\_4 <- tmp$par  
  
# Compute 'U\_1' and 'U\_2' and combine these two variables in 'U'  
U\_1 <- pstd(log\_return, mean = theta\_3[1], sd = theta\_3[2], nu = theta\_3[3])  
U\_2 <- pstd(log\_return, mean = theta\_4[1], sd = theta\_4[2], nu = theta\_4[3])  
U <- cbind(U\_1, U\_2)  
  
# Calibrate a Gaussian copula  
C <- normalCopula(dim = 2)  
fit <- fitCopula(C, data = U, method = "ml")  
  
# Set seed for generating pseudo-random numbers  
set.seed(4321)  
  
sim\_U <- rCopula(H \* t, fit@copula)  
sim\_log\_return <- qstd(sim\_U[,1], mean = theta\_3[1], sd = theta\_3[2], nu = theta\_3[3])  
sim\_vix\_return <- qstd(sim\_U[,2], mean = theta\_4[1], sd = theta\_4[2], nu = theta\_4[3])  
  
# Initialize the array 'sim\_ret\_3'  
sim\_ret\_3 <- array(data = NA, dim = c(H, 2, t))  
  
# Store in 'sim\_ret\_3' daily log returns for the underlying asset and the VIX  
for (i in 1:t) {  
 sim\_ret\_3[,,i] <- c(sim\_log\_return[(H \* (i - 1) + 1):(H \* i)], sim\_vix\_return[(H \* (i - 1) + 1):(H \* i)])  
}  
  
# Initialize two vectors that contain the underlying asset price and the VIX value one week from now  
S\_4 <- rep(NA, H)  
vol\_4 <- rep(NA, H)  
  
# Compute the underlying asset price and the VIX value one week from now  
for (i in 1:H) {  
 S\_4[i] <- S\_1 \* exp(sum(sim\_ret\_3[i,1,]))  
 vol\_4[i] <- vol\_1 \* exp(sum(sim\_ret\_3[i,2,]))  
}  
  
# Initialize a matrix to store call prices (10 000 rows (1 per simulation), 4 columns (1 per option))  
call\_price\_4 <- matrix(NA, nrow = H, ncol = 4)  
  
# Loop through 10 000 simulations and price each call option  
for (i in 1:10000){  
 call\_price\_4[i,1] <- price\_call(S\_4[i], K[1], r\_3, vol\_4[i], M[1] - t / 250)  
 call\_price\_4[i,2] <- price\_call(S\_4[i], K[2], r\_3, vol\_4[i], M[2] - t / 250)  
 call\_price\_4[i,3] <- price\_call(S\_4[i], K[3], r\_4, vol\_4[i], M[3] - t / 250)  
 call\_price\_4[i,4] <- price\_call(S\_4[i], K[4], r\_4, vol\_4[i], M[4] - t / 250)  
}  
  
# Compute the price of the portfolio for each replication and store it in 'PF\_price\_4'  
PF\_price\_4 <- rowSums(call\_price\_4)  
  
# Compute the P&L  
PL\_3 <- PF\_price\_4 \* exp(-(t / 360) \* r\_PL) - PF\_price\_1  
  
# Compute the VaR and the ES of the P&L distribution  
VaR\_3 <- sort(PL\_3)[(1 - alpha) \* H]  
ES\_3 <- mean(sort(PL\_3)[1:((1 - alpha) \* H)])  
  
# Plot an histogram  
hist(PL\_3, nclass = round(10 \* log(length(PL\_3))), probability = TRUE)  
  
# Add a vertical line to show the VaR  
abline(v = quantile(PL\_3, probs = (1 - alpha)),  
 lty = 1,  
 lwd = 2.5,  
 col = "red")



## The value at risk at alpha = 0.95 is -112.92$.

## The expected shortfall at alpha = 0.95 is -127.30$.

# Part VII: Volatility Surface

#install.packages("rgl")  
library("rgl")  
  
# Load the functions  
source(file = here("Functions", "vol\_surface.R")) # Implied volatility of an option  
source(file = here("Functions", "vol\_calibrate.r")) # Sum of absolute deviations of implied volalities  
  
# Count the number of traded options  
nb\_opts <- nrow(Market$calls) + nrow(Market$puts)  
  
# Build a matrix that contains the information relevant to traded call and put options  
mkt\_vol <- matrix(NA, nrow = nb\_opts, ncol = 4)  
  
# Assign names to columns  
colnames(mkt\_vol) <- c("S", "K", "tau", "IV")  
  
# Latest underlying asset price (spot price)  
mkt\_vol[,1] <- t(t(rep(sp\_500[n\_obs], nb\_opts)))  
  
# Strike price of call options  
mkt\_vol[1:nrow(Market$calls),2] <- Market$calls[,1]  
  
# Strike price of put options  
mkt\_vol[(nrow(Market$calls)+1):nb\_opts,2] <- Market$puts[,1]  
  
# Time to expiry of call options  
mkt\_vol[1:nrow(Market$calls),3] <- Market$calls[,2]  
  
# Time to expiry of put options  
mkt\_vol[(nrow(Market$calls)+1):nb\_opts,3] <- Market$puts[,2]  
  
# Implied volatility of call options  
mkt\_vol[1:nrow(Market$calls),4] <- Market$calls[,3]  
  
# Implied volatility of put options  
mkt\_vol[(nrow(Market$calls)+1):nb\_opts,4] <- Market$puts[,3]  
  
# Set a vector of initial values of a1, a2, a3, and a4  
x0 <- c(0.2, 1, 1, 0.1)  
  
# Calibrate the volatility surface on traded call and put options  
tmp <- optim(par = x0,  
 fn = vol\_calibrate)  
  
# Store parameters in 'theta\_vol'  
theta\_vol <- tmp$par  
  
# Generate a sequence of strike price and time to expiry  
x1 <- sp\_500[n\_obs] \* seq(0.5, 1.5, (1.5 - 0.5) / 1000)  
x2 <- seq(0.01, 2, (2 - 0.01) / 1000)  
  
# Generate al possible combinations of 'x1' and 'x2'  
x3 <- expand.grid(x1,x2)  
  
# Compute the implied volatility for each combination  
y <- vol\_surface(sp\_500[n\_obs], x3[,1], x3[,2], theta\_vol[1], theta\_vol[2], theta\_vol[3], theta\_vol[4])  
  
# Create a 3D-plot of the fitted volatility surface  
plot3d(x3[,1], x3[,2], y)

# Part VIII: Full approach

#install.packages("rugarch")  
library("rugarch")

## Loading required package: parallel

## Registered S3 method overwritten by 'xts':  
## method from  
## as.zoo.xts zoo

##   
## Attaching package: 'rugarch'

## The following object is masked from 'package:stats':  
##   
## sigma

# Residuals of the log-returns of the underlying using a Garch(1,1) with Normal innovations  
spec <- ugarchspec(variance.model = list(model = "sGARCH"),  
 mean.model = list(armaOrder = c(0,0),  
 include.mean = FALSE))  
  
fit <- ugarchfit(spec = spec, data = log\_return)  
Resid\_returns <- fit@fit$residuals  
  
# Residuals of the log-returns of the Vix using an AR(1) model   
ar1\_vix <- arima(vix\_return,order = c(1,0,0))  
Resid\_vix <- ar1\_vix$residuals  
  
# Fit normal marginals by MLE  
fit1 <- suppressWarnings(fitdistr(x = Resid\_returns,  
 densfun = dnorm,  
 start = list(mean = 0, sd = 1)))  
theta1 <- fit1$estimate  
  
fit2 <- suppressWarnings(fitdistr(x = Resid\_vix,  
 densfun = dnorm,  
 start = list(mean = 0, sd = 1)))  
theta2 <- fit2$estimate  
  
# Compute 'U\_1' and 'U\_2' and combine these two variables in 'U'  
U1 <- pnorm(Resid\_returns, mean = theta1[1], sd = theta1[2])  
U2 <- pnorm(Resid\_vix, mean = theta2[1], sd = theta2[2])  
U <- cbind(U1, U2)  
  
# Calibrate a Gaussian copula  
C <- normalCopula(dim = 2)  
fit <- fitCopula(C, data = U, method = "ml")  
  
# Set seed for generating pseudo-random numbers  
set.seed(4321)  
sim\_U <- rCopula(H \* t, fit@copula)  
sim\_Resid\_returns <- qnorm(sim\_U[,1], mean = theta1[1], sd = theta1[2])  
sim\_Resid\_vix <- qnorm(sim\_U[,2], mean = theta2[1], sd = theta2[2])  
  
# Initialize the array 'sim\_ret\_4'  
sim\_ret\_4 <- array(data = NA, dim = c(H, 2, t))  
  
# Store in 'sim\_ret\_4' daily residuals of log returns for the underlying asset and the VIX  
for (i in 1:t) {  
 sim\_ret\_4[,,i] <- c(sim\_Resid\_returns[(H \* (i - 1) + 1):(H \* i)], sim\_Resid\_vix[(H \* (i - 1) + 1):(H \* i)])  
}  
  
# Initialize two vectors that contain the residuals of the underlying asset price and the VIX value one week from now  
S\_5 <- rep(NA, H)  
vol\_5 <- rep(NA, H)  
  
# Compute the underlying asset price and the VIX value one week from now  
for (i in 1:H) {  
 S\_5[i] <- S\_1 \* exp(sum(sim\_ret\_4[i,1,]))  
 vol\_5[i] <- vol\_1 \* exp(sum(sim\_ret\_4[i,2,]))  
}  
  
# Initialize a matrix to store call prices (10 000 rows (1 per simulation), 4 columns (1 per option))  
call\_price\_5 <- matrix(NA, nrow = H, ncol = 4)  
  
# Loop through 10 000 simulations and price each call option  
for (i in 1:10000){  
 call\_price\_5[i,1] <- price\_call(S\_5[i], K[1], r\_3, vol\_5[i], M[1] - t / 250)  
 call\_price\_5[i,2] <- price\_call(S\_5[i], K[2], r\_3, vol\_5[i], M[2] - t / 250)  
 call\_price\_5[i,3] <- price\_call(S\_5[i], K[3], r\_4, vol\_5[i], M[3] - t / 250)  
 call\_price\_5[i,4] <- price\_call(S\_5[i], K[4], r\_4, vol\_5[i], M[4] - t / 250)  
}  
  
# Compute the price of the portfolio for each replication and store it in 'PF\_price\_5'  
PF\_price\_5 <- rowSums(call\_price\_5)  
  
# Compute the P&L  
PL\_4 <- PF\_price\_5 \* exp(-(t / 360) \* r\_PL) - PF\_price\_1  
  
# Compute the VaR and the ES of the P&L distribution  
VaR\_4 <- sort(PL\_4)[(1 - alpha) \* H]  
ES\_4 <- mean(sort(PL\_4)[1:((1 - alpha) \* H)])  
  
# Plot an histogram  
hist(PL\_4, nclass = round(10 \* log(length(PL\_4))), probability = TRUE)  
  
# Add a vertical line to show the VaR  
abline(v = quantile(PL\_4, probs = (1 - alpha)),  
 lty = 1,  
 lwd = 2.5,  
 col = "red")

