Statistical Methods in Financial Engineering - Risk Management Project

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

December 14, 2019

# LOOM video link:

# 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’. Also, we load 2 functions, the first one is use to price options using Black-Scholes and the second one is use to interpolate the term structure of interest rates. These functions will be detailled further in this script.

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

## here() starts at C:/Users/tdtra/Desktop/Method Stat/New Project/Method\_Stat

# Load the data  
load(file = here("Data", "Market.rda"))  
  
# Load the functions  
source(file = here("Functions", "price\_BS.r")) # Option prices 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. K is the strike price and T-t is days before maturity. 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 <- matrix(data = NA, nrow = 4, ncol = 6)  
  
# Assign names to columns  
# Option Type: Call = 1, Put = 0  
colnames(book) <- c("Quantity", "Option Type", "Strike Price", "Maturity", "Option price", "Position Value")  
  
# Store initial values  
book[1,1:4] <- c(1, 1, 1600, 20 / 250)  
book[2,1:4] <- c(1, 1, 1650, 20 / 250)  
book[3,1:4] <- c(1, 1, 1750, 40 / 250)  
book[4,1:4] <- 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 price of each option in the portfolio. The value of the portfolio of options is simply the sum of each option’s price multiplied by its quantity. 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\_BS’: Takes as inputs the underlying asset price, the option strike price, the risk-free rate, the volatility, the maturity and the option type. Uses Black-Scholes model to price options.

# Number of observations  
n\_obs <- length(sp\_500)  
  
# Convert 250-day basis year in 360-day basis year  
m <- book[,4]\*(250 / 360)  
  
# Interpolated interest rates  
rf <- mapply(lin\_inter, m)  
  
# Latest underlying asset price (spot price)  
S0 <- sp\_500[n\_obs]  
S <- matrix(data = S0, nrow = 4)  
  
# Latest VIX value  
Vol0 <- vix[n\_obs]  
Vol <- matrix(data = Vol0, nrow = 4)  
  
# Strike prices  
K <- book[,3]  
  
# Maturities  
M <- book[,4]  
  
# Option type  
Type <- book[,2]  
  
# Compute the options values  
book[,5] <- mapply(price\_BS, S, K, rf, Vol, M, Type)  
  
# Compute the value of the portfolio  
book[,6] <- book[,1]\*book[,5]  
PF\_val <- sum(book[,6])

## The total value of the Options Portfolio is 156.98$.

##   
## Portfolio detail:

## Quantity Option Type Strike Price Maturity Option price  
## [1,] 1.00 1.00 1600.00 0.08 87.57   
## [2,] 1.00 1.00 1650.00 0.08 47.72   
## [3,] 1.00 1.00 1750.00 0.16 15.30   
## [4,] 1.00 1.00 1800.00 0.16 6.38   
## Position Value  
## [1,] 87.57   
## [2,] 47.72   
## [3,] 15.30   
## [4,] 6.38

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: Risk Model 1: One Risk Driver and Gaussian Distribution

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.

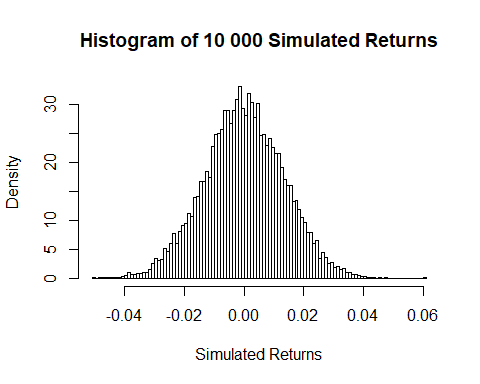
# Daily underlying asset log returns  
log\_return <- diff(log(sp\_500))

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\_1 <- mean(log\_return)  
s2\_hat\_1 <- var(log\_return)  
sd\_hat\_1 <- s2\_hat\_1^0.5  
  
# Store parameters in 'theta\_1'  
theta\_1 <- c(mu\_hat\_1, sd\_hat\_1)

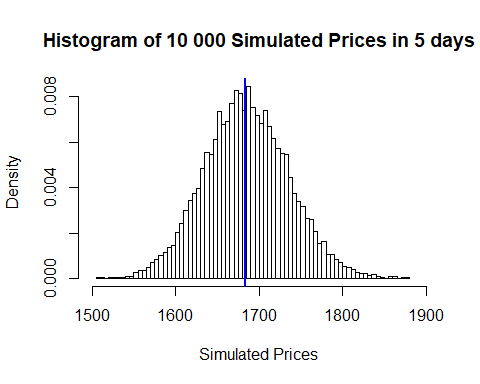
Now that we have estimated the parameters of the underlying asset return log returns, we can run a simulation of the underlying asset price one week ahead by generating normally distributed IID shocks with mean ‘mu\_hat’ and standard deviation ‘sd\_hat’. Since we found the distribution parameters using 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  
  
# Use set seed for testing purpose? Yes (1), No (0).  
source(file = here("Functions", "seed.r"))  
use\_set\_seed <- 1  
seed(use\_set\_seed)  
  
# 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)  
  
# Histogram of the simulated returns  
hist(sim\_ret\_1[,5], nclass = round(10 \* log(length(sim\_ret\_1))),  
 probability = TRUE,  
 main = "Histogram of 10 000 Simulated Returns",  
 xlab = "Simulated Returns")



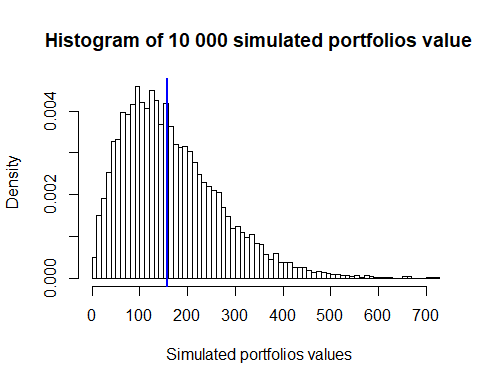
In order to obtain the underlying asset price one week from now, we simply have to compute the exponential sum of the five daily normally distributed IID shocks per trajectory and multiply by the latest underlying asset price. The last things we need to modify are the risk-free rate for the remaining life of the option and the remaining days 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  
sim\_S\_1 <- S0 \* exp(rowSums(sim\_ret\_1))  
  
# Update the risk-free rate (360-day basis year)  
rf\_m\_t <- mapply(lin\_inter, (m - t / 360))  
  
# Initialize a matrix to store call prices (10 000 rows (1 per simulation), 4 columns (1 per option))  
sim\_price\_1 <- matrix(NA, nrow = H, ncol = 4)  
  
# Loop through H simulations and price the options  
for (i in 1:H){  
 sim\_price\_1[i,] <- mapply(price\_BS, sim\_S\_1[i], K, rf\_m\_t, Vol, M - t / 250, Type)  
}  
  
# Histogram of the simulated prices  
hist(sim\_S\_1, nclass = round(10 \* log(length(sim\_S\_1))),  
 probability = TRUE, xlim = c(1500,1900),  
 main = "Histogram of 10 000 Simulated Prices in 5 days",  
 xlab = "Simulated Prices")  
  
# Add a verticale line that represents the spot price  
abline(v = S0,  
 lty = 1,  
 lwd = 2.5,  
 col = "blue")



For each replication, we can compute the value of the portfolio of 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\_val\_1'  
PF\_val\_1 <- rowSums(sim\_price\_1 \* book[,1])  
  
# Histogram of portfolio's value  
hist(PF\_val\_1, nclass = round(10 \* log(length(PF\_val\_1))),  
 probability = TRUE, xlim = c(0,700),  
 main = "Histogram of 10 000 simulated portfolios value",  
 xlab = "Simulated portfolios values")  
  
# Add a vertical that represent last portfolio's value  
abline(v = PF\_val,  
 lty = 1,  
 lwd = 2.5,  
 col = "blue")

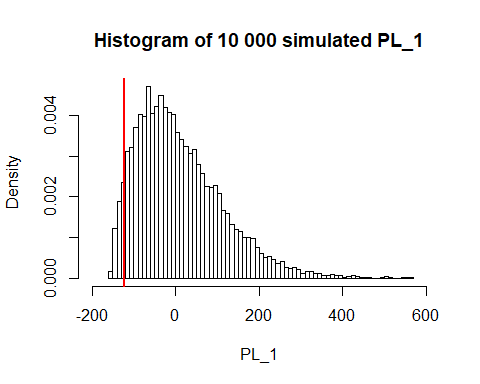


Based on the histogram of the simulated portfolio values, we notice that the porfolio of options as a function of underlying asset price at t = 0 is right skewed. Call option prices are therefore more sensitive to a rise in the price of the underlying asset rather than a decline.

# Compute the risk-free rate for a period of 5 days (360-day basis year)  
rf\_t <- lin\_inter(t / 360)  
  
# Compute the P&L  
PL\_1 <- PF\_val\_1 \* exp(-(t / 360) \* rf\_t) - PF\_val

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 empirical 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, xlim = c(-200,600),  
 main = "Histogram of 10 000 simulated PL\_1")  
  
# 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 -122.24$.

## The expected shortfall at alpha 0.95 is -134.80$.

## Analysis portfolio drivers

This section explain what drive the portfolio price: What drive portfolio value is the inputs of our models (BlackShcole formula). Strike price and volatility are constants, risk-free rate is stable. So distribution of the portfolio value can be explained by price of underlying asset and days remaining until maturity.

Porfolio value is right skewed due to the relationship between calls value and underlying price at t = 0.

Here we observe the value of the calls and the portfolio of the calls 5 days later. We isolated the change in portfolio value due to days until maturity change only i.e. without change in the underlying asset price, volatility and risk-free rate.

day\_s = 5/250  
book\_5day\_later <- book[,1:6]  
book\_5day\_later[,4] <- book[,4] - day\_s  
  
book\_5day\_later[,5] <- mapply(price\_BS, S, K, rf, Vol, M - day\_s, Type)  
book\_5day\_later[,6] <- book\_5day\_later[,1]\*book\_5day\_later[,5]  
  
PF\_val\_M\_choc <- sum(book\_5day\_later[,6])  
  
M\_choc <- PF\_val\_M\_choc - PF\_val

## Calls value and maturity usually have positive relationship, and we can observe that 5 days in less until maturity lower

## portfolio value by 8.06$.

Summary of P&L distribution and it’s drivers

1. Portfolio price and underlying price have positive relationship
2. P&L is right skewed because of the relationship between Calls value and underlying asset price at t = 0
3. Portfolio value is more sensible to a positive change in uderlying asset price than a negative one
4. 5 days in less until maturity make the portolio loose 8.06$ in value (for a same underlying asset price)

# Part V: Risk Model 2: Two Risk Drivers and Gaussian Distribution

The biggest assumption of the Risk Model 1 is that the volatility is assumed constant between now and the following week. In practice, volatility is not constant as evidenced by the time series of the VIX index as a function of time. To account for this, we allow volatility to vary in Risk Model 2. More specifically, we model the dynamics of the underlying asset price and volatility with a multivariate normal distribution that we calibrate using the S&P500 and VIX log returns.

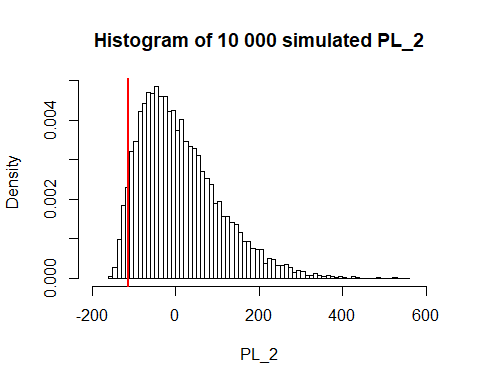
# install.packages("MASS")  
library("MASS")  
  
# Daily VIX log returns  
vix\_return <- diff(log(vix))  
  
# Store underlying asset log returns and VIX log returns in 'rets'  
rets\_2 <- matrix(NA, nrow = n\_ret, ncol = 2)  
rets\_2[,1] <- log\_return  
rets\_2[,2] <- vix\_return  
  
# Calibration  
mu\_hat\_2 <- colMeans(rets\_2)  
sg\_hat\_2 <- ((n\_ret - 1) / n\_ret) \* cov(rets\_2)  
  
# Store parameters in 'theta\_2'  
theta\_2 <- list(mu = mu\_hat\_2, sigma = sg\_hat\_2)

We run a simulation of the underlying asset price and volatility index value one week from now and re-price our portfolio of options in each scenario.

# Initialize the array 'sim\_ret\_2'  
sim\_ret\_2 <- array(data = NA, dim = c(H, 2, t))  
  
# Set seed for generating pseudo-random numbers  
seed(use\_set\_seed)  
  
# 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  
sim\_S\_2 <- rep(NA, H)  
sim\_vol\_2 <- rep(NA, H)  
  
# Compute the underlying asset price and the VIX value one week from now  
for (i in 1:H) {  
 sim\_S\_2[i] <- S0 \* exp(sum(sim\_ret\_2[i,1,]))  
 sim\_vol\_2[i] <- Vol0 \* exp(sum(sim\_ret\_2[i,2,]))  
}  
  
# Initialize a matrix to store call prices (H rows (1 per simulation), 4 columns (1 per option))  
sim\_price\_2 <- matrix(NA, nrow = H, ncol = 4)  
  
# Loop through H simulations and price each option  
for (i in 1:H){  
 sim\_price\_2[i,] <- mapply(price\_BS, sim\_S\_2[i], K, rf\_m\_t, sim\_vol\_2[i], M - t / 250, Type)  
}  
  
# Compute the price of the portfolio for each replication and store it in 'PF\_val\_2'  
PF\_val\_2 <- rowSums(sim\_price\_2 \* book[,1])

Finally, we plot the P&L distribution.

# Compute the P&L  
PL\_2 <- PF\_val\_2 \* exp(-(t / 360) \* rf\_t) - PF\_val  
  
# 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, xlim = c(-200,600),  
 main = "Histogram of 10 000 simulated PL\_2")  
  
# 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 -111.69$.

## The expected shortfall at alpha 0.95 is -124.36$.

Compared to Risk Model 1, Risk Model 2 has a higher (better) VaR and a higher (better) ES for alpha = 0.95. This is not surprising knowing that the correlation between the underlying asset log returns and the VIX log returns is negative:

## The correlation between the S&P 500 and the VIX is -0.7537

Since the value of a call option increases as the underlying asset and volatility rise, and because these two variables are negatively correlated, they have an offsetting effect on the value of a call option. As a result, our portfolio of call options is less risky and has a higher VaR.

# Part VI: Risk Model 3: Two Risk Drivers and Copula-Marginal Model (Student-t and Gaussian Copula)

We add another layer on top of what we built up to now by introducing a Gaussian copula in the mix. The copula models the dependence structure of the two marginal distributions. In this case, we assume that the marginals are student-t distributions with 10 degrees of freedom for the underlying asset price and 5 degrees of freedom for the volatility.

The first step in implementing Risk Model 3 is to calibrate the two marginal distributions using a similar methodology than previously discussed.

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

## Loading required package: timeDate

## Loading required package: timeSeries

## Loading required package: 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\_S\_3'  
theta\_S\_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\_vol\_3'  
theta\_vol\_3 <- tmp$par

The second step in implementing Risk Model 3 is to use the probability integral transformation theorem to obtain random variables that follow a uniform distribution for both the underlying asset and the volatility from their respective cumulative distribution function. We call these variables ‘U\_1’ and ‘U\_2’ and use them to calibrate the Gaussian copula.

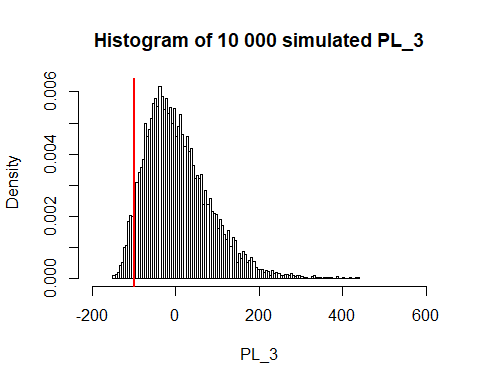
# Compute 'U\_1' and 'U\_2' and combine these two variables in 'U'  
U\_1 <- pstd(log\_return, mean = theta\_S\_3[1], sd = theta\_S\_3[2], nu = theta\_S\_3[3])  
U\_2 <- pstd(vix\_return, mean = theta\_vol\_3[1], sd = theta\_vol\_3[2], nu = theta\_vol\_3[3])  
U <- cbind(U\_1, U\_2)  
  
# Calibrate a Gaussian copula  
C <- normalCopula(dim = 2)  
fit <- fitCopula(C, data = U, method = "ml")  
  
# Set seed for generating pseudo-random numbers  
seed(use\_set\_seed)  
  
sim\_U <- rCopula(H \* t, fit@copula)  
sim\_log\_ret <- qstd(sim\_U[,1], mean = theta\_S\_3[1], sd = theta\_S\_3[2], nu = theta\_S\_3[3])  
sim\_vix\_ret <- qstd(sim\_U[,2], mean = theta\_vol\_3[1], sd = theta\_vol\_3[2], nu = theta\_vol\_3[3])

We run a simulation of the underlying asset price and volatility index value one week from now and re-price our portfolio of options in each scenario.

# 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\_ret[(H \* (i - 1) + 1):(H \* i)], sim\_vix\_ret[(H \* (i - 1) + 1):(H \* i)])  
}  
  
# Initialize two vectors that contain the underlying asset price and the VIX value one week from now  
sim\_S\_3 <- rep(NA, H)  
sim\_vol\_3 <- rep(NA, H)  
  
# Compute the underlying asset price and the VIX value one week from now  
for (i in 1:H) {  
 sim\_S\_3[i] <- S0 \* exp(sum(sim\_ret\_3[i,1,]))  
 sim\_vol\_3[i] <- Vol0 \* exp(sum(sim\_ret\_3[i,2,]))  
}  
  
# Initialize a matrix to store call prices (H rows (1 per simulation), 4 columns (1 per option))  
sim\_price\_3 <- matrix(NA, nrow = H, ncol = 4)  
  
# Loop through H simulations and price each option  
for (i in 1:H){  
 sim\_price\_3[i,] <- mapply(price\_BS, sim\_S\_3[i], K, rf\_m\_t, sim\_vol\_3[i], M - t / 250, Type)  
}  
  
# Compute the price of the portfolio for each replication and store it in 'PF\_val\_3'  
PF\_val\_3 <- rowSums(sim\_price\_3 \* book[,1])

We plot the P&L distribution, as we did in previous parts.

# Compute the P&L  
PL\_3 <- PF\_val\_3 \* exp(-(t / 360) \* rf\_t) - PF\_val  
  
# 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, xlim = c(-200,600),  
 main = "Histogram of 10 000 simulated PL\_3")  
  
# 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 -98.67$.

## The expected shortfall at alpha 0.95 is -112.77$.

Compared to Risk Model 1 and Risk Model 2, Risk Model 3 has the smallest VaR and the smallest ES for alpha = 0.95. As it was the case with Risk Model 2, we account for the fact that the underlying asset price and the volatility are negatively correlated. On top of that, we use the Gaussian copula to model the dependence structure of the two marginal distributions. The Gaussian copula underestimates the likelihood of extreme negative events in the tail of the distributions.

# Part VII: Risk Model 4: Volatility Surface

In this section, we take a slightly different approach than with previous risk models. Indeed, we fit a volatility surface on implied volatilities of traded call and put options using the parametric model described in the assignment.

# 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  
  
calls <- as.matrix(Market$calls)  
puts <- as.matrix(Market$puts)  
   
# Count the number of traded options  
nb\_opts <- nrow(calls) + nrow(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] <- matrix(data = S0, nb\_opts)  
  
# Strike price of options  
mkt\_vol[,2] <- c(calls[,1], puts[,1])  
  
# Time to expiry of options  
mkt\_vol[,3] <- c(calls[,2], puts[,2])  
  
# Implied volatility of options  
mkt\_vol[,4] <- c(calls[,3], 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 options  
tmp <- optim(par = x0, fn = vol\_calibrate)  
  
# Store parameters in 'theta\_vol'  
theta\_vol <- tmp$par

We have a plot for the volatility surface. To reduce computation time, we put it in comments.

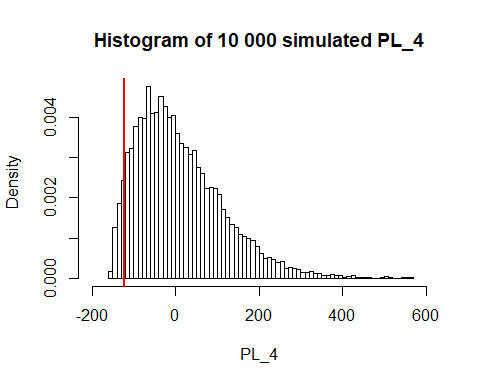
#install.packages("rgl")  
#library("rgl")  
  
# Generate a sequence of strike price and time to expiry  
#x1 <- S0 \* seq(0.5, 1.5, (1.5 - 0.5) / 1000)  
#x2 <- seq(0.01, 2, (2 - 0.01) / 1000)  
  
# Generate al possible combinations of 'x1' and 'x2'  
#x3 <- expand.grid(x1,x2)  
  
# Compute the implied volatility for each combination  
#y <- vol\_surface(S0, 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)

Now that we have estimated the parameters that describe the shape of the volatility surface observed at t = 0, we can compute the ATM implied volatility given by the VIX. To estimate the ATM implied volatility in five days, we assume that the ATM implied volatility difference stays the same. Therefore, we use linear interpolation to project the implied volatility in five days.

# Compute the ATM implied volatility  
vix\_1y\_atm\_IV <- theta\_vol[1] + theta\_vol[4]  
vix\_atm\_IV <- vol\_surface(S0, S0, M, theta\_vol[1], theta\_vol[2], theta\_vol[3], theta\_vol[4])  
  
# Compute the ATM implied volatility difference  
vol\_shift <- ((vix\_1y\_atm\_IV - vix\_atm\_IV) / 250) \* t  
  
# Set seed for generating pseudo-random numbers  
seed(use\_set\_seed)  
  
# Store in 'sim\_ret\_4' normally distributed IID shocks   
sim\_ret\_4 <- matrix(rnorm(t \* H, mean = mean(log\_return), sd = var(log\_return)^0.5), nrow = H, ncol = t)  
  
# Compute the underlying asset price one week from now  
sim\_S\_4 <- S0 \* exp(rowSums(sim\_ret\_4))  
  
# Initialize a matrix to store call prices (H rows (1 per simulation), 4 columns (1 per option))  
sim\_price\_4 <- matrix(NA, nrow = H, ncol = 4)  
  
# Loop through H simulations and price each option  
for (i in 1:H){  
 sim\_price\_4[i,] <- mapply(price\_BS, sim\_S\_4[i], K, rf\_m\_t, Vol + vol\_shift, M - t / 250, Type)  
}  
  
# Compute the price of the portfolio for each replication and store it in 'PF\_val\_4'  
PF\_val\_4 <- rowSums(sim\_price\_4 \* book[,1])

We plot the P&L distribution, as we did in previous parts.

# Compute the P&L  
PL\_4 <- PF\_val\_4 \* exp(-(t / 360) \* rf\_t) - PF\_val  
  
# 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, xlim = c(-200,600),  
 main = "Histogram of 10 000 simulated PL\_4")  
  
# Add a vertical line to show the VaR  
abline(v = quantile(PL\_4, probs = (1 - alpha)),  
 lty = 1,  
 lwd = 2.5,  
 col = "red")



## The value at risk at alpha 0.95 is -122.51$.

## The expected shortfall at alpha 0.95 is -135.02$.

Compared to previous risk models, Risk Model 4 exhibits similar results to the first risk model. This can be explained by the fact that by isolating the effect of the shift of the volatility surface on option prices, we did not take into account the negative correlation that exists between the underlying asset price and the volatility. In this case, we only generated stock returns and shifted the current implied volatility by the one-year ATM implied volatility difference. The shift reduced slightly the volatility of the options and therefore the VAR and ES of P&L is a bit lower than Risk Model 1.

# Part VIII: Risk Model 5: Full approach

In this section, instead of using the time series of our market data, we use the residuals. We use a Garch(1,1) model for the log-returns of the underlying asset and an AR(1) model for the volatiliy. We assume that the residuals are normally distributed and use a gaussian copula to model the dependence structure.

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

## Loading required package: parallel

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

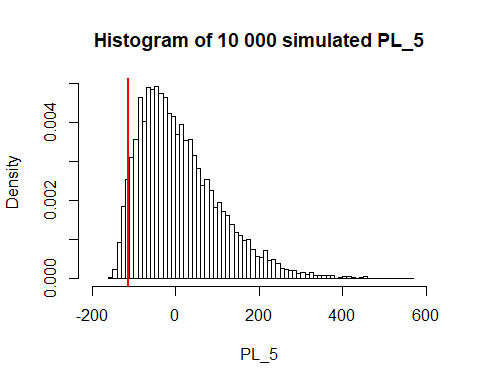
##   
## Attaching package: 'rugarch'

## 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", garchOrder = c(1,1)),  
 mean.model = list(armaOrder = c(0,0),  
 include.mean = FALSE),  
 distribution.model = "norm")  
  
garch\_fit <- ugarchfit(spec = spec, data = log\_return)  
Resid\_returns <- garch\_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  
  
# Set seed for generating pseudo-random numbers  
seed(use\_set\_seed)  
  
# 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)  
  
# U1 <- pnorm(log\_return, mean = mean(log\_return), sd = var(log\_return)^0.5)  
# U2 <- pnorm(vix\_return, mean = mean(vix\_return), sd = var(vix\_return)^0.5)  
# U <- cbind(U1, U2)  
  
# Calibrate a Gaussian copula  
C <- normalCopula(dim = 2)  
fit <- fitCopula(C, data = U, method = "ml")  
  
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\_5'  
sim\_ret\_5 <- array(data = NA, dim = c(H, 2, t))  
  
# Store in 'sim\_ret\_5' daily residuals of log returns for the underlying asset and the VIX  
for (i in 1:t) {  
 sim\_ret\_5[,,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  
sim\_S\_5 <- rep(NA, H)  
sim\_vol\_5 <- rep(NA, H)  
  
# Compute the underlying asset price and the VIX value one week from now  
for (i in 1:H) {  
 sim\_S\_5[i] <- S0 \* exp(sum(sim\_ret\_5[i,1,]))  
 sim\_vol\_5[i] <- Vol0 \* exp(sum(sim\_ret\_5[i,2,]))  
}  
  
# Initialize a matrix to store call prices (H rows (1 per simulation), 4 columns (1 per option))  
sim\_price\_5 <- matrix(NA, nrow = H, ncol = 4)  
  
# Loop through H simulations and price each option  
for (i in 1:H){  
 sim\_price\_5[i,] <- mapply(price\_BS, sim\_S\_5[i], K, rf\_m\_t, sim\_vol\_5[i], M - t / 250, Type)  
}

We plot the P&L distribution, as we did in previous parts.

# Compute the price of the portfolio for each replication and store it in 'PF\_price\_5'  
PF\_val\_5 <- rowSums(sim\_price\_5 \* book[,1])  
  
# Compute the P&L  
PL\_5 <- PF\_val\_5 \* exp(-(t / 360) \* rf\_t) - PF\_val  
  
# Compute the VaR and the ES of the P&L distribution  
VaR\_5 <- sort(PL\_5)[(1 - alpha) \* H]  
ES\_5 <- mean(sort(PL\_5)[1:((1 - alpha) \* H)])  
  
# Plot an histogram  
hist(PL\_5, nclass = round(10 \* log(length(PL\_5))),   
 probability = TRUE, xlim = c(-200,600),  
 main = "Histogram of 10 000 simulated PL\_5")  
  
# Add a vertical line to show the VaR  
abline(v = quantile(PL\_5, probs = (1 - alpha)),  
 lty = 1,  
 lwd = 2.5,  
 col = "red")



## The value at risk at alpha 0.95 is -112.03$.

## The expected shortfall at alpha 0.95 is -123.96$.

The VAR and ES are similar to the second model. We also tried this model versus using regular vix returns and underlying asset returns and end up with similar results. This means that the Garch(1,1) and AR(1) models doesn’t really improve the results. Using the principle of Occam’s razor, between two model giving similar results, using the simpler model is better.

# Part IX: Results and Conclusion

Here is a table summarizing the results of each model.

## Portfolio value 156.98$.

##   
## Portfolio detail:

## Quantity Option Type Strike Price Maturity Option price  
## [1,] 1.00 1.00 1600.00 0.08 87.57   
## [2,] 1.00 1.00 1650.00 0.08 47.72   
## [3,] 1.00 1.00 1750.00 0.16 15.30   
## [4,] 1.00 1.00 1800.00 0.16 6.38   
## Position Value  
## [1,] 87.57   
## [2,] 47.72   
## [3,] 15.30   
## [4,] 6.38

##   
## Portfolio 5-days risk:

## M1: 1 risk/Gauss M2: 2 risks/Gauss M3: 2 risk/Copula  
## Value At Risk -122.239 -111.686 -98.673   
## Expected Shortfall -134.800 -124.359 -112.774   
## M4: Vol Surface M5: Full Approach  
## Value At Risk -122.515 -112.025   
## Expected Shortfall -135.022 -123.956