Project 2 / Risk Management

2024-04-16

In this report, we implement part of the risk management framework for estimating the risk of a book of European call options on the S&P500. We take into account two risk drivers: the underlying and implied volatility.

For ease of interpretation and for efficiency, we proceed by vectorizing, cleaning and organizing our data on the S&P500, the VIX index, term structure of interest rates, and traded options.

```r
library("PerformanceAnalytics")
library("magrittr")
library("dplyr")
library("here")
library("zoo")
library("magrittr")
library("docstring")
library("qrmdata")
library("plotly")
library("akima")

set.seed(777)
source(here("Functions/Functions.R"))
load(here("Data/Market.rda"))

# Vectorize, categorize & store data
variables <- list(

  index_prices = Market$sp500 %>%
    zoo::fortify.zoo() %>%
    dplyr::as_tibble() %>%
    dplyr::rename(Date = Index, SP500 = ".") %>%
    dplyr::arrange(Date) %>%
    dplyr::left_join(
      Market$vix %>%
        zoo::fortify.zoo() %>%
        dplyr::as_tibble() %>%
        dplyr::rename(Date = Index, VIX = ".") %>%
        dplyr::arrange(Date), by = "Date"),

  calls = Market$calls %>% dplyr::as_tibble(),

  puts = Market$puts %>% dplyr::as_tibble(),

  rf = dplyr::tibble(
    rf_rate = unlist(Market$rf[, 1]),
    maturity = unlist(as.numeric(names(Market$rf)))),

  options = dplyr::tibble(
    strike_price = c(1600, 1650, 1750, 1800),
    maturity = c(20, 20, 40, 40))
)

head(variables)
```

```
## $index_prices
## # A tibble: 3,410 × 3
##    Date       SP500   VIX
##    <date>     <dbl> <dbl>
##  1 2000-01-03 1455. 0.242
##  2 2000-01-04 1399. 0.270
##  3 2000-01-05 1402. 0.264
##  4 2000-01-06 1403. 0.257
##  5 2000-01-07 1441. 0.217
##  6 2000-01-10 1458. 0.217
##  7 2000-01-11 1439. 0.225
##  8 2000-01-12 1432. 0.228
##  9 2000-01-13 1450. 0.217
## 10 2000-01-14 1465. 0.197
## # … with 3,400 more rows
##
## $calls
## # A tibble: 422 × 3
##        K    tau    IV
##    <dbl>  <dbl> <dbl>
##  1  1280 0.0256 0.737
##  2  1370 0.0256 0.969
##  3  1380 0.0256 0.945
##  4  1400 0.0256 0.527
##  5  1415 0.0256 0.508
##  6  1425 0.0256 0.482
##  7  1430 0.0256 0.273
##  8  1450 0.0256 0.379
##  9  1455 0.0256 0.429
## 10  1470 0.0256 0.713
## # … with 412 more rows
##
## $puts
## # A tibble: 750 × 3
##        K    tau    IV
##    <dbl>  <dbl> <dbl>
##  1  1000 0.0256 1.01
##  2  1025 0.0256 1.01
##  3  1050 0.0256 0.962
##  4  1075 0.0256 0.917
##  5  1100 0.0256 0.873
##  6  1120 0.0256 0.838
##  7  1125 0.0256 0.830
##  8  1130 0.0256 0.821
##  9  1140 0.0256 0.804
## 10  1150 0.0256 0.753
## # … with 740 more rows
##
## $rf
## # A tibble: 14 × 2
##     rf_rate maturity
##       <dbl>    <dbl>
##  1 0.000710  0.00274
##  2 0.000980  0.0192
##  3 0.00128   0.0833
##  4 0.00224   0.25
##  5 0.00342   0.5
##  6 0.00451   0.75
##  7 0.00432   1
##  8 0.00643   2
##  9 0.00906   3
## 10 0.0117    4
## 11 0.0141    5
## 12 0.0176    7
## 13 0.0208   10
## 14 0.0204   30
##
## $options
## # A tibble: 4 × 2
##   strike_price maturity
##          <dbl>    <dbl>
## 1         1600       20
## 2         1650       20
## 3         1750       40
## 4         1800       40
```

Pricing of a portfolio of options

We begin by assuming the book of European call options: 1x strike K = 1600 maturity 20 days, 1x strike K = 1650 maturity 20 days, 1x strike K = 1750 maturity 40 days and 1x strike K = 1800 maturity 40 days.

To price our book of options, we apply Black-Scholes with the latest price of the underlying, the latest implied volatility (i.e.VIX for all options) and we linearly interpolate the term structure of interest rates to find the corresponding rate.

We assume 360 days/year for the term structure and 250 days/year for the time to maturity of options.

```r
# price options using Black-Scholes
variables$options$prices <- f_pfPricing(
  S = dplyr::last(variables$index_prices$SP500),
  sd = dplyr::last(variables$index_prices$VIX),
  K = variables$options$strike_price,
  t = variables$options$maturity,
  rf = variables$rf$rf_rate,
  rf_t = variables$rf$maturity)
```

One risk driver and univariate Gaussian model

To begin, we're interested in the one-week ahead P&L distribution of our book of options. Here, we strictly consider one risk factor (the underlying), and allow the implied volatilities to remain fixed in a five-days ahead period.

We proceed firstly by assuming log-returns of the underlying are iid normally distributed and stimulate one-week ahead returns. Then, we retrieve the one-week ahead prices of the underlying, re-interpolate to find the new rate and apply Black-Scholes.

```r
message("Do not overwrite original variable because slice(-1) loses data.")
```

```
## Do not overwrite original variable because slice(-1) loses data.
```

```r
#compute log rets & store
variables$index_prices <- variables$index_prices %>%
  dplyr::mutate(
    dplyr::across(
      .cols = c(SP500, VIX),
      .fns = ~log(.) - dplyr::lag(log(.)),
      .names = "{.col}_log_return"
    )) %>%
  dplyr::slice(-1)

# fit normal distribution to rets
y <- diff(variables$index_prices$SP500_log_return)
sig_hat <- f_fitGaussian(y)

# simulate 5 days ahead prices
n_sim <- 10000
sim_prices <- f_forecastPrices(
  n_sim = n_sim,
  days = 5,
  mu = sig_hat$mean,
  sd = sig_hat$sd,
  S = dplyr::last(variables$index_prices$SP500))


# initialize
sim_optionPrices <- matrix(ncol = length(sim_prices),
                           nrow = length(variables$options$strike_price))

# find options price for simulated scenarios
for (i in seq_along(sim_prices)) {

  sim_optionPrices[, i] <- f_pfPricing(
    S = sim_prices[i],
    sd = dplyr::last(variables$index_prices$VIX),
    K = variables$options$strike_price,
    t = variables$options$maturity - 5,
    rf = variables$rf$rf_rate,
    rf_t = variables$rf$maturity)
}

# subtract option price at t_0 from simulated scenario prices at t_1
profitLoss <- sweep(sim_optionPrices, MARGIN = 1, variables$options$prices, `-`)

# sum the P&Ls of all options for each scenario
book_pnl <- colSums(profitLoss)

# determine distribution of P&L book of options
theta_hat2 <- f_fitGaussian(book_pnl)
theta_hat2
```

```
## $mean
## [1] -40.6824
##
## $sd
## [1] 209.7386
```

We now plot the P&L distribution of our book of options.

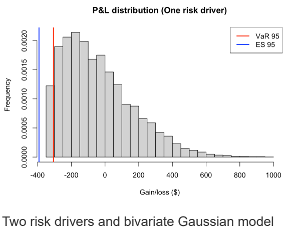
```r
f_ES <- function(p, mu = 0, sd = 1){mu + sd * dnorm(qnorm(p)) / (1 - p)} #quick ES helper function

# VaR95 and ES95
a <- 0.05
VaR95 <- quantile(book_pnl,a)[[1]]
VaR95 <- -VaR95
VaR95
```

```
## [1] 306.2875
```

```r
ES95 <- -f_ES(0.95,theta_hat2$mean,theta_hat2$sd)
ES95
```

```
## [1] -391.948
```

```r
# plot pnl
hist(book_pnl,
     main = "P&L distribution (One risk driver)",
     xlab = "Gain/loss ($)",
     ylab = "Frequency",
     breaks = 25,
     border = "black",
     freq=FALSE)

abline(v = -VaR95, col = "red",lwd = 2)
abline(v = ES95, col = "blue", lwd = 2)
legend("topright", legend = c("VaR 95","ES 95"), col = c("red","blue"), lwd = 2)
```

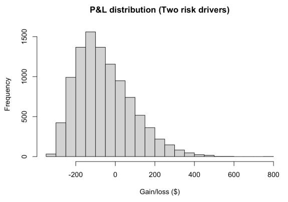


Two risk drivers and bivariate Gaussian model

We now consider two risk factors: the underlying and the volatility. Similar to the previous section, we assume log-returns of the underlying and the log difference of volatility are iid normally distributed. Then, we stimulate one-week ahead prices and volatilities, re-interpolate to find the new rate and apply Black-Scholes.

```r
library("mvtnorm")

x1 <- variables$index_prices$SP500_log_return
x2 <- variables$index_prices$VIX_log_return
rets <- cbind(x1, x2)

## fit univariate Gaussian model to log rets
# risk driver 1
sig_hat1 <- f_fitGaussian(x1)
# risk driver 2
sig_hat2 <- f_fitGaussian(x2)

# simulate 10000 log rets with multivariate Gaussian model
n_sim <- 10000
mu <- c(sig_hat1$mean, sig_hat2$mean)
sigma <- cov(rets)
S <- dplyr::last(variables$index_prices$SP500)
VIX <- dplyr::last(variables$index_prices$VIX)

#get prices form simulated log rets using the bivariate model
sample_prices <- matrix(nrow = n_sim, ncol = 2)
for (i in 1:n_sim) {
  sample_rets <- rmvnorm(5, mean = mu, sigma = sigma)
  sample_prices[i, 1] <- last(S * exp(cumsum(sample_rets[, 1])))
  sample_prices[i, 2] <- last(VIX * exp(cumsum(sample_rets[, 2])))
}

# Value options using simulated values. Take interpolated rates for the term structure
sim_optionPrices2 <- matrix(ncol = n_sim,
                            nrow = length(variables$options$strike_price))

for (i in seq_along(sim_prices)) {
  sim_optionPrices2[, i] <- f_pfPricing(
    S = sample_prices[i, 1],
    sd = sample_prices[i, 2],
    K = variables$options$strike_price,
    t = variables$options$maturity - 5,
    rf = variables$rf$rf_rate,
    rf_t = variables$rf$maturity)
}
```

```r
# Find pnl by subtracting initial option price at t_0 from each scenario price at t_1
profitLoss2 <- sweep(sim_optionPrices2, MARGIN = 1, variables$options$prices, `-`)
book_pnl2 <- colSums(profitLoss2)
hist(book_pnl2,
     main = "P&L distribution (Two risk drivers)",
     xlab = "Gain/loss ($)",
     ylab = "Frequency",
     breaks = 20,
     border = "black")
```



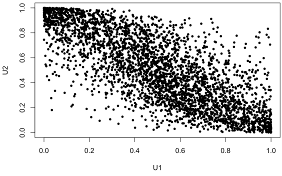
Two risk drivers and copula-marginal model

Considering multiple risk factors marks an improvement from the last method, although it still suffers from a major drawback when the distribution of the underlying returns or the log difference of the VIX are not actually normally distributed. In other words, the stimulated observations through the multivariate normal distribution are poor when the marginals are not necessarily normal. We thus turn towards Copulas to solve this.

In order to strictly isolate any dependence structure between our risk factors, we turn to Copulas.

We now assume the returns of the underlying to follow a Student-t distribution with v = 10 degrees of freedom and the log-difference of the VIX to follow a Student-t distribution with v = 5 degrees of freedom. W will use the normal copula to merge the marginals, recompute the P&L and the risk figures.

```r
library("copula")
library("MASS")
library("fGarch")

x1 <- variables$index_prices$SP500_log_return
x2 <- variables$index_prices$VIX_log_return
set.seed(123)

# fit student-t marginals to log rets of SP500 & VIX by MLE
fit1 <- suppressWarnings(
  fitdistr(x = x1, densfun = dstd, nu = 10, start = list(mean = 0, sd = 1)))
fit2 <- suppressWarnings(
  fitdistr(x = x2, densfun = dstd, nu = 5, start = list(mean = 0, sd = 1)))

theta1 <- fit1$estimate
theta2 <- fit2$estimate

# fit Gaussian copula
U1 <- pstd(x1, mean = theta1[1], sd = theta1[2], nu = 10) #fix degrees of freedom
U2 <- pstd(x2, mean = theta2[1], sd = theta2[2], nu = 5)
U <- cbind(U1, U2)

plot(U, pch = 20, cex = 0.9, main = "Gaussian Copula")
```

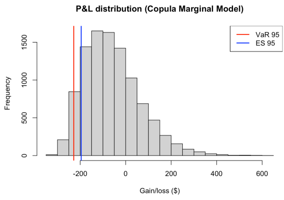


```
C <- normalCopula(dim = 2)
fit <- fitCopula(C, data = U, method = "ml")
```

```
## Warning in fitCopula.ml(copula, u = data, method = method, start = start, :
## possible convergence problem: optim() gave code=52
```

```
# Risk by simulation
n_sim <- 10000
w <- c(.5, .5)

marginal_sim_prices <- matrix(nrow = n_sim, ncol = 2)

for (i in 1:n_sim) {
  # simulate pairwise log-returns (SP500, VIX) using copula
  U_sim <- rCopula(5, fit@copula)
  copula_retsSP500 <- qstd(U_sim[, 1], mean = theta1[1], sd = theta1[2], nu = 10)
  copula_retsVIX <- qstd(U_sim[, 2], mean = theta2[1], sd = theta2[2], nu = 5)

  #from pairwise simulated returns, extract individual prices for SP500
  # & volatility for VIX
  marginal_sim_prices[i, 1] <- last(S * exp(cumsum(copula_retsSP500))) #for SP500
  marginal_sim_prices[i, 2] <- last(VIX * exp(cumsum(copula_retsVIX))) #for VIX
}

# value book of options using Copula-simulated prices & VIX
# take interpolated rates for the term structure
sim_optionPrices3 <- matrix(ncol = length(sim_prices),
                            nrow = length(variables$options$strike_price))

for (i in 1:n_sim) {
  sim_optionPrices3[, i] <- f_pfPricing(
    S = marginal_sim_prices[i, 1],
    sd = marginal_sim_prices[i, 2],
    K = variables$options$strike_price,
    t = variables$options$maturity - 5,
    rf = variables$rf$rf_rate,
    rf_t = variables$rf$maturity)
}

# Find P&L by substracting initial option price from each scenario price
profitLoss3 <- sweep(sim_optionPrices3, MARGIN = 1, variables$options$prices, `-`)
book_pnl3 <- colSums(profitLoss3)
```

```
# Compute VaR and ES
alpha <- 0.05
theta_hat3 <- f_fitGaussian(book_pnl3) # det P&L distribution of book of options

VaR95_copula <- -quantile(book_pnl3, alpha)
ES95_copula <- -f_ES(1-alpha, theta_hat3$mean,theta_hat3$sd)

hist(book_pnl3,
     main = "P&L distribution (Copula Marginal Model)",
     xlab = "Gain/loss ($)",
     ylab = "Frequency",
     breaks = 25,
     border = "black")
abline(v = -VaR95_copula, col = "red",lwd = 2)
abline(v = ES95_copula, col = "blue", lwd = 2)
legend("topright", legend = c("VaR 95","ES 95"), col = c("red","blue"), lwd = 2)
```



Volatility surface

The Black-and-Scholes model assumes volatility to be constant and returns to be log-normally distributed, which is not particularly what the market exhibits. To compensate for this, we now make use of the volatility surface.

For the code below, we will work with the parametric surface σ(m, τ) = α1 + α2(m - 1)^2 + α3(m - 1)^3 + α4√τ, where $m = \frac{K}{S}$ is the moneyness and τ is the time to maturity of the option in years.

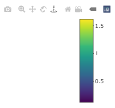
We now fit a volatility surface to the implied volatilities observed on the market (traded call and put options). Then, we minimize the absolute distance between the market implied volatilities and the model implied volatilities.

```
knitr::opts_chunk$set(warning = FALSE, message = FALSE)

# calculate moneyness for calls
calls_moneyness <- variables$calls %>%
  dplyr::rename(strike = K, maturity = tau, market_vol = IV) %>%
  dplyr::mutate(moneyness = strike / dplyr::last(variables$index_prices$SP500))

# calculate moneyness for puts
puts_moneyness <- variables$puts %>%
  dplyr::rename(strike = K, maturity = tau, market_vol = IV) %>%
  dplyr::mutate(moneyness = strike / dplyr::last(variables$index_prices$SP500))

# concat both moneyness (for puts and calls)
implied_vols <- dplyr::bind_rows(
  calls_moneyness %>% dplyr::select(strike, maturity, moneyness, market_vol),
  puts_moneyness %>% dplyr::select(strike, maturity, moneyness, market_vol)
)

# fit volatility surface model against puts and calls
# find optimal parameters
fitted_surface_params <- f_fitVolSurface( m = implied_vols$moneyness,
                                          tau = implied_vols$maturity,
                                          market_vols = implied_vols$market_vol)

# compute volatilities with parameter estimates
implied_vols$adjusted_vol <- f_VolatilitySurface( m = implied_vols$moneyness,
                                                  tau = implied_vols$maturity,
                                                  params = as.numeric(fitted_surface_params))

# plot the volatility surface (see viewer tab)
interp_data <- akima::interp(
  x = implied_vols$moneyness,
  y = implied_vols$maturity,
  z = implied_vols$adjusted_vol,
  duplicate = "mean",
  xo = seq(min(implied_vols$moneyness), max(implied_vols$moneyness), length = 100),
  yo = seq(min(implied_vols$maturity), max(implied_vols$maturity), length = 100)
)
```

```
## Warning in akima::interp(x = implied_vols$moneyness, y =
## implied_vols$maturity, : collinear points, trying to add some jitter to avoid
## colinearities!
```

```
## Warning in akima::interp(x = implied_vols$moneyness, y =
## implied_vols$maturity, : success: collinearities reduced through jitter
```

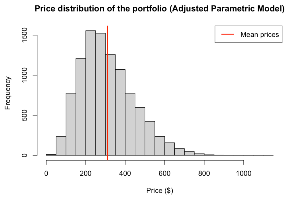
```
fig_3d <- plotly::plot_ly(x = interp_data$x, y = interp_data$y, z = interp_data$z, type = "surface")
fig_3d <- fig_3d %>% layout(scene = list(xaxis = list(title = "Moneyness"),
                                         yaxis = list(title = "Time to Maturity"),
                                         zaxis = list(title = "Implied Volatility")))
fig_3d
```



The volatility surface and volatility “smile” are displayed in the above plots. Implied volatility increases for deep ITM options and deep OTM options, whereas it decreases for ATM options as demonstrated by the local minimum. Moreover, implied volatility increases with time-to-maturity and with moneyness.

We now re-price our book of options in one week assuming the same parametric model but shifted by the one-year ATM implied volatility difference (i.e. we assume the VIX and the one-year ATM implied volatility difference remain when we simulate the VIX forward).

```
# Re-price the portfolio in one week assuming the same parametric model but shifted by the one-year ATM IV differ
ence
# take simulated forward VIX and underlying price

portfolio_price_vol_surface <- matrix(
  ncol = n_sim,
  nrow = length(variables$options$strike_price))

for (i in seq_along(sim_prices)) {
  # find the Implied volatility of our PF options in 5 days using our parametric model and the simulated underlyi
ng price in 5 days
  variables$options$IV_surface <- f_VolatilitySurface(
    m = variables$options$strike_price / sample_prices[i, 1],
    tau = (variables$options$maturity - 5) / 250,
    params = as.numeric(fitted_surface_params))

  # find portfolio option values in 5 days such that the volatility surface level match the VIX in 5 days
  portfolio_price_vol_surface[, i] <- suppressWarnings(f_pfPricing(
    S = sample_prices[i, 1],
    sd = variables$options$IV_surface - (as.numeric(fitted_surface_params[1]) + as.numeric(fitted_surface_params
[4])) + sample_prices[i, 2],
    K = variables$options$strike_price,
    t = variables$options$maturity - 5,
    rf = variables$rf$rf_rate,
    rf_t = variables$rf$maturity))
}

# Plot the price of the portfolio for each simulation and the mean using the adjusted parametric model
hist(colSums(portfolio_price_vol_surface),
     main = "Price distribution of the portfolio (Adjusted Parametric Model)",
     xlab = "Price ($)",
     ylab = "Frequency",
     breaks =25,
     border = "black")
abline(v = mean(colSums(portfolio_price_vol_surface)), col = "red", lwd = 2)
legend("topright", legend = "Mean prices", col = "red", lwd = 2)
```



```
# Calculate P&L book value of portfolio for each simulation (recompute the prices of each option)
profitLoss4 <- sweep(portfolio_price_vol_surface, MARGIN = 1, variables$options$prices, `-`)
book_pnl4 <- colSums(profitLoss4)

hist(book_pnl4,
     main = "P&L distribution (Adjusted Parametric Model)",
     xlab = "Gain/loss ($)",
     ylab = "Frequency",
     breaks = 25,
     border = "black")
```