

TP2 Risk Management

TP2: Hair Parra , Alessio Bressan, Ioan Catalin

2023-04-12

Libraries

Risk Management: European Options Portfolio

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

Data

Load the database Market. Identify the price of the **SP500**, the **VIX index**, the term structure of interest rates (current and past), and the traded options (calls and puts).

```
# load dataset into environment
load(file = here("data_raw", "Market.rda"))

# reassign name and inspect structure of loaded data
mkt <- Market
summary(mkt)
```

```
##           Length Class  Mode
## sp500 3410    xts    numeric
## vix   3410    xts    numeric
## rf      14 -none- numeric
## calls 1266 -none- numeric
## puts  2250 -none- numeric
```

```
str(mkt)
```

```
## List of 5
## $ sp500:An xts object on 2000-01-03 / 2013-09-10 containing:
##   Data:    double [3410, 1]
##   Index:   Date [3410] (TZ: "UTC")
## $ vix :An xts object on 2000-01-03 / 2013-09-10 containing:
##   Data:    double [3410, 1]
##   Index:   Date [3410] (TZ: "UTC")
## $ rf : num [1:14, 1] 0.00071 0.00098 0.00128 0.00224 0.00342 ...
##   ..- attr(*, "names")= chr [1:14] "0.00273972602739726" "0.0192307692307692" "0.0833333333333333" "0.25" .
## $ calls: num [1:422, 1:3] 1280 1370 1380 1400 1415 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:3] "K" "tau" "IV"
## $ puts : num [1:750, 1:3] 1000 1025 1050 1075 1100 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:3] "K" "tau" "IV"
```

Let's unpack these into the env. individually:

```
# unpack each of the elements in the mkt list
sp500 <- mkt$sp500
vix <- mkt$vix
Rf <- mkt$rf # risk-free rates
calls <- mkt$calls
puts <- mkt$puts

# assign colname for aesthetic
colnames(sp500) <- "sp500"
colnames(vix) <- "vix"
```

SP500 and VIX

By inspection, we observe that we the SP500 and VIX indices are contained in the `sp500` and `vix` xts objects respectively.

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

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

```
head(vix)
```

```
##              vix
## 2000-01-03 0.2421
## 2000-01-04 0.2701
## 2000-01-05 0.2641
## 2000-01-06 0.2573
## 2000-01-07 0.2172
## 2000-01-10 0.2171
```

```
par(mfrow = c(2,1))
```

```
# plot both series on top of each other
plot(sp500)
plot(vix)
```

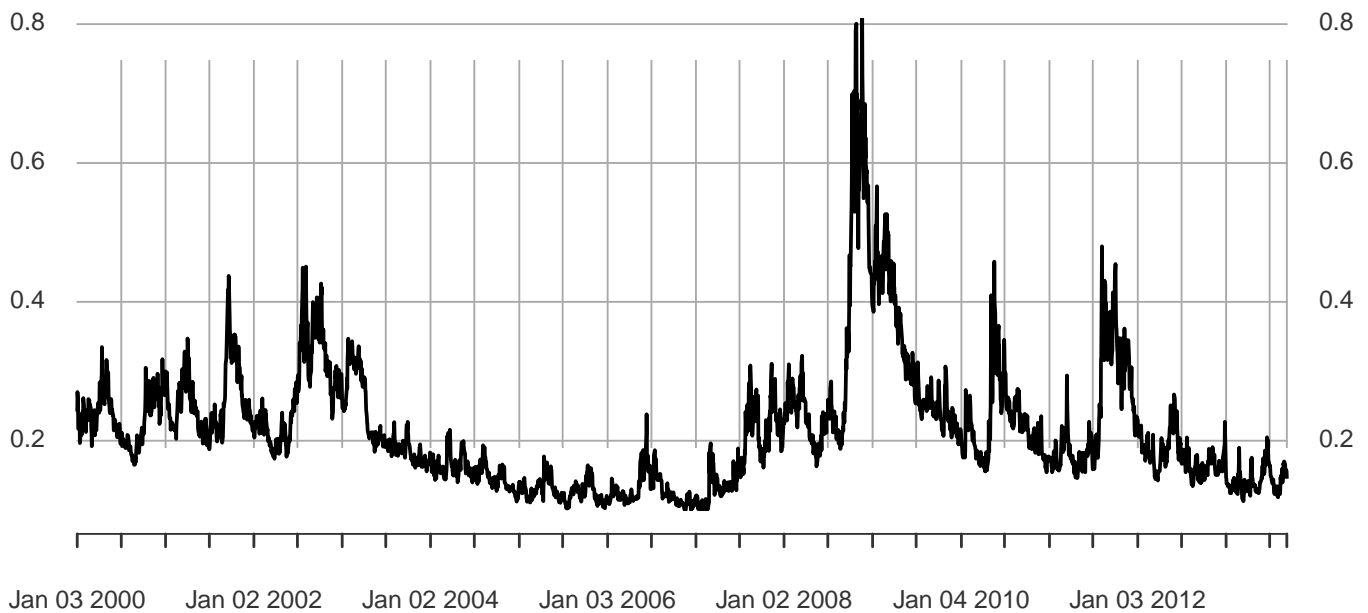
sp500

2000-01-03 / 2013-09-10



vix

2000-01-03 / 2013-09-10



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
```

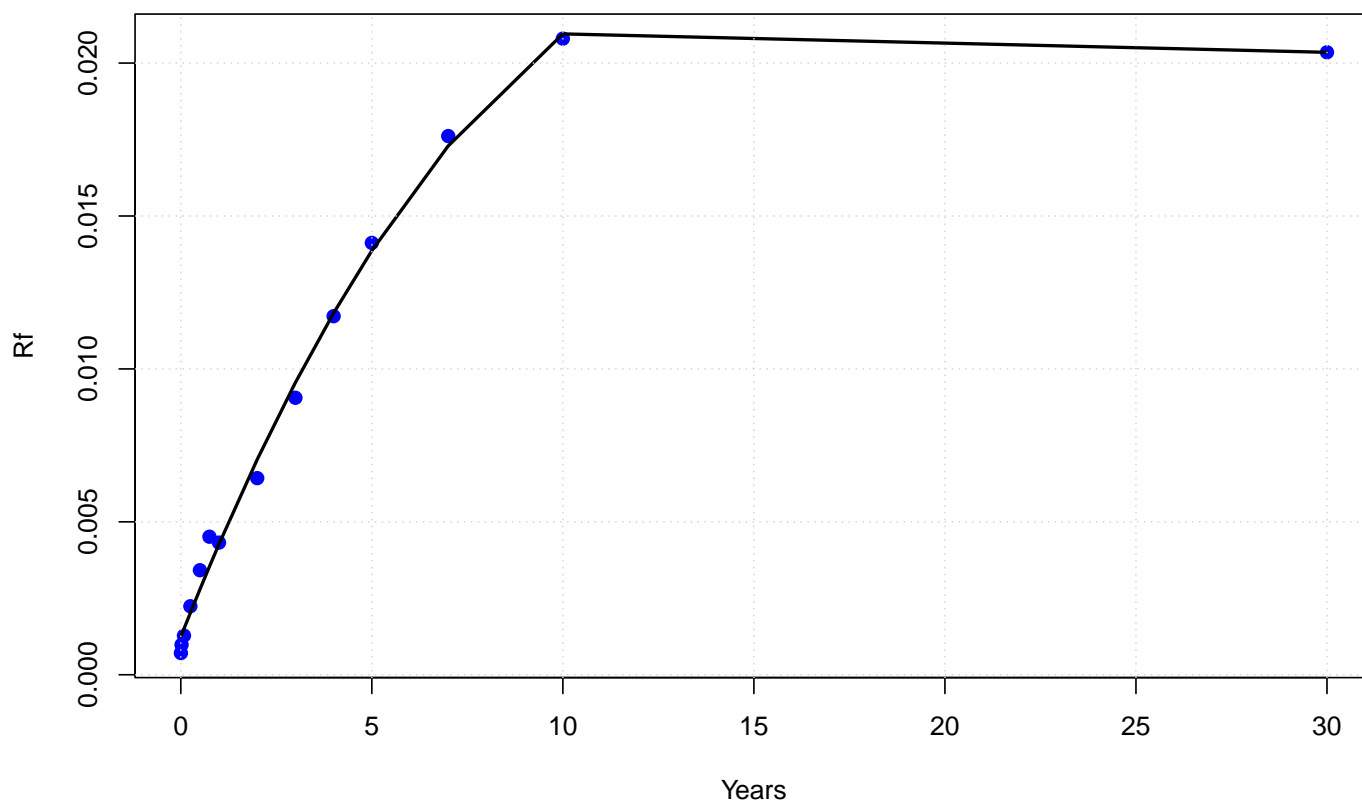
```
##          1d          1w          1m          3m          6m          9m
## 0.0007099993 0.0009799908 0.0012799317 0.0022393730 0.0034170792 0.0045123559
##          1y          2y          3y          4y          5y          7y
## 0.0043206525 0.0064284968 0.0090558654 0.0117237591 0.0141196498 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
```

Term Structure of Risk-Free Rates



Calls

The `calls` object displays the different values of K (**Strike Price**), τ (**time to maturity**) and $\sigma = IV$ (**Implied Volatility**)

```
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:

$$\begin{cases} C(S_t, t) = S e^{yT} N(d_1) - K e^{-r \times \tau} N(d_2), \\ P(S_t, t) = K e^{-r \times \tau} (1 - N(d_2)) - S e^{y \times T} (1 - N(d_1)), \end{cases}$$

where:

$$\begin{cases} d_1 = \frac{\ln\left(\frac{S_t}{K}\right) + \tau\left(r + \frac{\sigma^2}{2}\right)}{\sigma\sqrt{\tau}} \\ d_2 = d_1 - \sigma\sqrt{\tau} \end{cases}$$

, further the Put Option price corresponds to the ****Put-Call parity****, given by:

$$C(S_t, t) + K e^{-r \times \tau} = P(S_t, t) + S_t$$

Note As here we don't have dividends, then $y = 0$, and so

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

BlackScholes function

```
# source code with options pricign
source(here("code", "OptionPricing.R")) # BlackScholes and Option pricing

# 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**:

- 1x strike $K = 1600$ with maturity $T = 20d$
- 1x strike $K = 1605$ with maturity $T = 40d$
- 1x strike $K = 1800$ with maturity $T = 40d$

Find the price of this book given **the last underlying price** and the **last implied volatility** (take the VIX for all options). Use **Black-Scholes** to price the options. Take the current term structure and **linearly interpolate** to find the corresponding rates. Use 360 days/year for the term structure and **250 days/year** for the maturity of the options.

Nearest values

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

```
# 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:

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

```
# Function to interpolate y given two points
interpolate <- function(x, x1=1, y1=1, x2=2, y2=2){
  y1 + (x-x1)*(y2-y1)/(x2-x1)
}
```

Finding the rates through interpolation

The **yield curve** for the given structure of interest rates can be modeled a function $r_f = f(x)$, where x is the number of years. Then, we can interpolate the values as follows:

```
# Interest rates
rf_mat
```

```
##           rf           years           days
## 1d  0.0007099993  0.002739726   0.9863014
## 1w  0.0009799908  0.019230769   6.9230769
## 1m  0.0012799317  0.083333333  30.0000000
## 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
```



```
head(calls)
```

```
##           K           tau           IV tau_days
## [1,] 1280 0.02557005 0.7370370 6.392513
## [2,] 1370 0.02557005 0.9691616 6.392513
## [3,] 1380 0.02557005 0.9451401 6.392513
## [4,] 1400 0.02557005 0.5274481 6.392513
## [5,] 1415 0.02557005 0.5083375 6.392513
## [6,] 1425 0.02557005 0.4820041 6.392513
```

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

```
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
```

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

1. 1x strike $K = 1600$ with maturity $T = 20d$
2. 1x strike $K = 1605$ with maturity $T = 40d$
3. 1x strike $K = 1800$ with maturity $T = 40d$

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

```
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=40, K=1605, calls=calls, rf_mat=rf_mat, S_t = S_t, IV = IV)
```

```
## $Call
## [1] 90.22871
##
## $Put
## [1] NA
##
## $S
## [1] 1683.99
##
## $K
## [1] 1605
##
## $r_interp
## [1] 0.001721275
##
## $calls
##      K      tau      IV tau_days
## 1605.0000000 0.1022824 0.1676923 25.5705949
##
## $rates
##      rf      years      days
## 1m 0.001279932 0.08333333 30
## 3m 0.002239373 0.25000000 90
```

Third 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
```

Two risk drivers and copula-marginal model (Student-t and Gaussian Copula)

1. Compute the daily log-returns of the underlying stock
2. Assume the first invariant is generated using a Student-t distribution with $\nu = 10$ df and the second invariant is generated using a Student-t distribution with $\nu = 5$ df.
3. Assume the **normal copula** to merge the marginals.
4. Generate 10000 scenarios for the one-week ahead price 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:

$$H(x, y) = C(F(x), G(y)) = C(F^{-1}(u), G^{-1}(v))$$

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.

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:

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

```
# 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
```

Generating the simulation scenarios

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

```
# Load required libraries
library("fGarch")

## NOTE: Packages 'fBasics', 'timeDate', and 'timeSeries' are no longer
## attached to the search() path when 'fGarch' is attached.
##
## If needed attach them yourself in your R script by e.g.,
##       require("timeSeries")
```

```
##
## Attaching package: 'fGarch'

## The following object is masked from 'package:TTR':
##
## volatility
```

```
library("MASS")
library("copula")
library("Matrix")

# 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
```

```
sigma
```

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

```
## Fit marginals by MLE
```

```
# Student-t for sp500
fit1 <- suppressWarnings(
  fitdistr(x = sp500_rets_vec,
    densfun = dstd,
    start = list(mean = 0, sd = 1, nu = 10))
)
theta1 <- fit1$estimate #extract fitted parameters
```

```
# Student-t for vix
fit2 <- suppressWarnings(
  fitdistr(x = vix_rets_vec,
    densfun = dstd,
    start = list(mean = 0, sd = 1, nu = 5))
)
theta2 <- fit2$estimate # extract fitted parameters
```

```
# display parameters
theta1
```

```
##          mean          sd          nu
## 0.0004414879 0.0156603739 2.6953920404
```

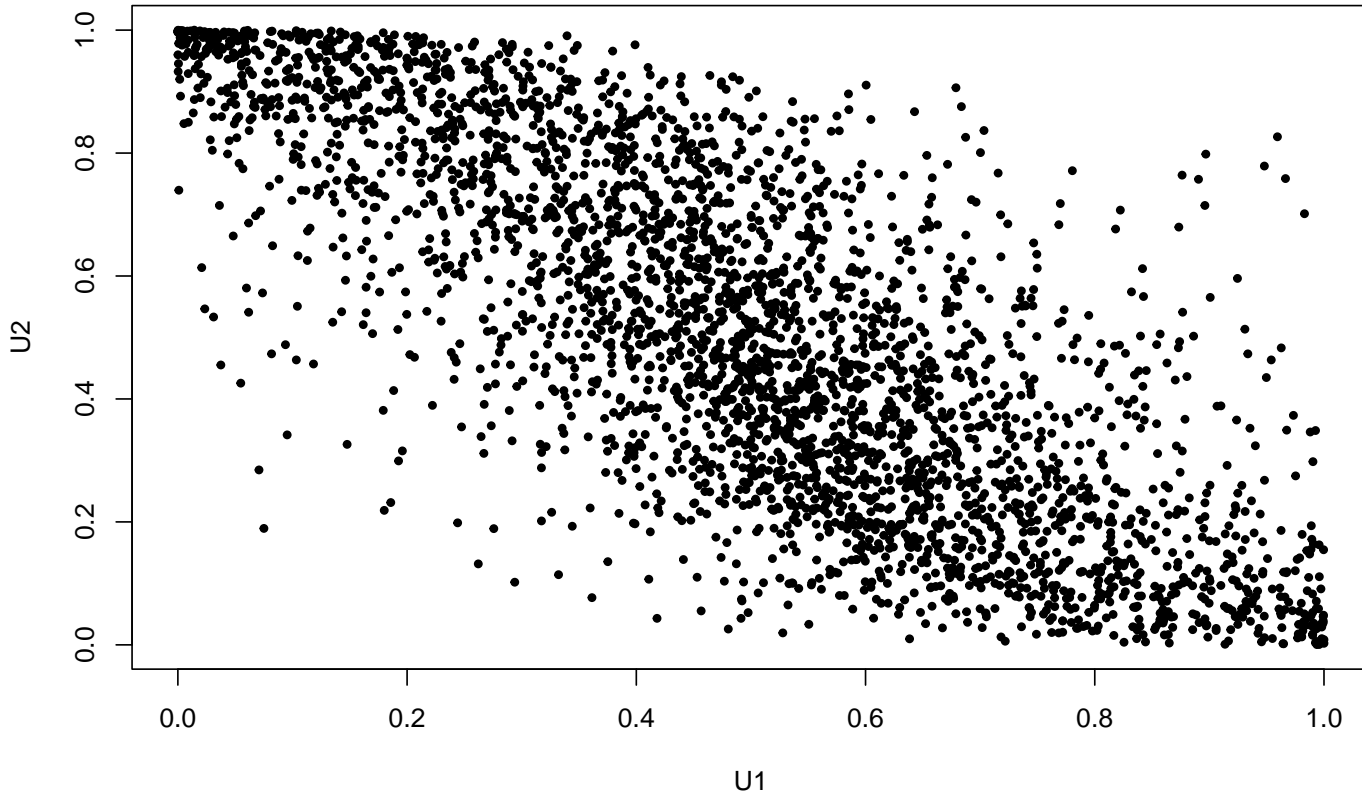
```
theta2
```

```
##          mean          sd          nu
## -0.003475206 0.064192681 4.230323432
```

```

# Fit Student-t to the marginals
# U1 <- pstd(sp500_rets_vec, mean = theta1[1], sd = theta1[2], nu = theta1[3]) # sp500
# U2 <- pstd(vix_rets_vec, mean = theta2[1], sd = theta2[2], nu = theta2[3]) # vix
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
# U1 <- pt(sp500_rets_vec, df = 5) # sp500
# U2 <- pt(vix_rets_vec, df = 10) # vix
U <- cbind(U1, U2) # join into one matrix
plot(U, pch = 20, cex = 0.9)

```



```

# 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
## -0.7984
## The maximized loglikelihood is 1494
## Optimization converged

```

```

# seed for replication
set.seed(420)

# Simulation parameters
n_sim = 10000 # set number of simulations

```

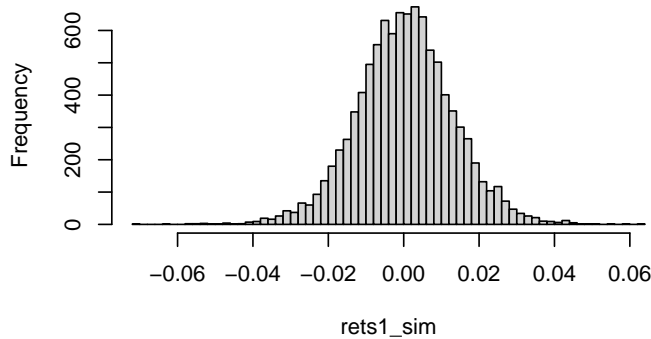
```
# n_ahead = 5 # days ahead to produce samples

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

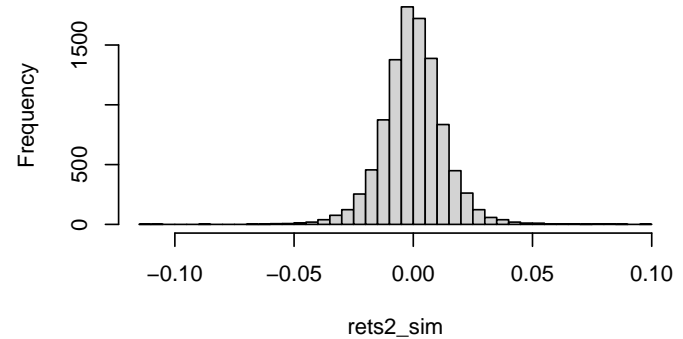
# use copula U_sim to reproduce the marginals with student-t distr
# rets1_sim <- qstd(U_sim[,1], mean = mu[1], sd = sigma[1], nu = theta1[3]) # sp500
# rets2_sim <- qstd(U_sim[,2], mean = mu[1], sd = sigma[1], nu = theta2[3]) # vix
rets1_sim <- qstd(U_sim[,1], mean = mu[1], sd = sigma[1], nu = 10) # 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)))
```

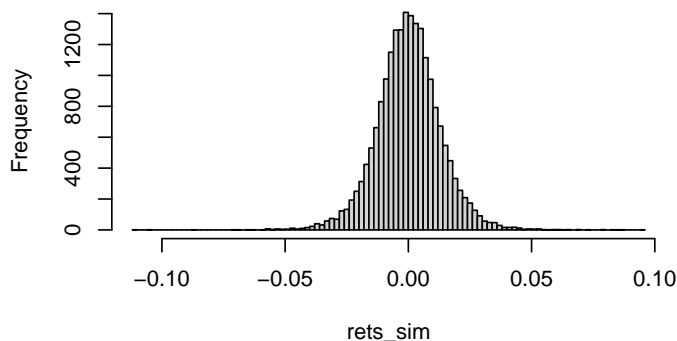
Histogram of rets1_sim



Histogram of rets2_sim



Histogram of rets_sim



```
# 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 <- matrix(NA, nrow = n_sim, ncol=5)
sim_rets_vix <- matrix(NA, nrow = n_sim, ncol=5)

# assign days ahead
colnames(sim_rets_sp500) <- c("T+1", "T+2", "T+3", "T+4", "T+5")
colnames(sim_rets_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 5-days ahead 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
  rets_sim <- cbind(rets1_sim, rets2_sim)

  # store simulation of log return in matrix
  sim_rets_sp500[,t] <- rets1_sim
  sim_rets_vix[,t] <- rets2_sim
}

# preview of simulated log returns
head(sim_rets_sp500)

```

```

##           T+1           T+2           T+3           T+4           T+5
## [1,] -0.0009645126  0.0158554079  0.015383149 -0.0174310226  0.0155476192
## [2,]  0.0022227010  0.0178616966  0.003249986  0.0015075435  0.0004961551
## [3,] -0.0202696762  0.0070645564  0.011080134 -0.0163632659  0.0039542793
## [4,]  0.0267344996  0.0190648399 -0.004275895  0.0312856876  0.0004778219
## [5,] -0.0045092785  0.0008870700 -0.005286801  0.0089807462 -0.0122229007
## [6,] -0.0039970206 -0.0002199501 -0.003419139 -0.0004051185  0.0371441443

```

```
head(sim_rets_vix)
```

```

##           T+1           T+2           T+3           T+4           T+5
## [1,]  0.01231074  0.006644294 -0.005354024  0.01255679 -0.04752175
## [2,] -0.04109607 -0.073223553 -0.020098934 -0.03207569 -0.06300583
## [3,]  0.08429964 -0.030662396 -0.071921523  0.10934242 -0.05145715
## [4,] -0.08896620 -0.032518583  0.020560914 -0.12085679 -0.02129170
## [5,] -0.05179948 -0.017505235  0.022004416 -0.04412445  0.03046923
## [6,]  0.01708910 -0.034281364  0.032441799  0.04104414 -0.11119617

```

Computing Prices from Returns

Next, we crate a function to forecast the 5 day ahead prices from the returns. Since:

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

This logic is implemented in the `f_logret_to_price()` function through the `f_next_Pt()` function, under the `code/Utils.R` script.

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

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

# unpack matrices
sim_price_sp500 <- sim_val_mats$sp500
sim_vol_vix <- sim_val_mats$vix
```

```
# compare simulated returns with the price
head(sim_rets_sp500)
```

```
##           T+1           T+2           T+3           T+4           T+5
## [1,] -0.0009645126  0.0158554079  0.015383149 -0.0174310226  0.0155476192
## [2,]  0.0022227010  0.0178616966  0.003249986  0.0015075435  0.0004961551
## [3,] -0.0202696762  0.0070645564  0.011080134 -0.0163632659  0.0039542793
## [4,]  0.0267344996  0.0190648399 -0.004275895  0.0312856876  0.0004778219
## [5,] -0.0045092785  0.0008870700 -0.005286801  0.0089807462 -0.0122229007
## [6,] -0.0039970206 -0.0002199501 -0.003419139 -0.0004051185  0.0371441443
```

```
head(sim_price_sp500)
```

```
##           T+1           T+2           T+3           T+4           T+5
## 1 1682.367 1709.254 1735.751 1705.757 1732.485
## 2 1687.737 1718.154 1723.747 1726.347 1727.204
## 3 1650.200 1661.899 1680.415 1653.142 1659.692
## 4 1729.618 1762.909 1755.387 1811.174 1812.039
## 5 1676.414 1677.901 1669.054 1684.111 1663.651
## 6 1677.272 1676.904 1671.180 1670.503 1733.719
```

```
# compare simulated log rets with volatility
head(sim_rets_vix)
```

```
##           T+1           T+2           T+3           T+4           T+5
## [1,]  0.01231074  0.006644294 -0.005354024  0.01255679 -0.04752175
## [2,] -0.04109607 -0.073223553 -0.020098934 -0.03207569 -0.06300583
```

```
## [3,]  0.08429964 -0.030662396 -0.071921523  0.10934242 -0.05145715
## [4,] -0.08896620 -0.032518583  0.020560914 -0.12085679 -0.02129170
## [5,] -0.05179948 -0.017505235  0.022004416 -0.04412445  0.03046923
## [6,]  0.01708910 -0.034281364  0.032441799  0.04104414 -0.11119617
```

```
head(sim_vol_vix)
```

```
##           T+1           T+2           T+3           T+4           T+5
## 1 0.1470998 0.1480804 0.1472897 0.1491509 0.1422287
## 2 0.1394498 0.1296037 0.1270248 0.1230150 0.1155035
## 3 0.1580798 0.1533063 0.1426674 0.1591518 0.1511695
## 4 0.1329316 0.1286783 0.1313515 0.1163985 0.1139464
## 5 0.1379651 0.1355711 0.1385873 0.1326051 0.1367077
## 6 0.1478044 0.1428233 0.1475327 0.1537141 0.1375377
```

Pricing the simulation scenarios

Recall the initial (call) options:

- 1x strike $K = 1600$ with maturity $T = 20d$
- 1x strike $K = 1605$ with maturity $T = 40d$
- 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)

# Initialize strikes and maturities the options
T_vec <- c(20,40,40) # maturities
K_vec <- c(1600, 1605, 1800) # Strikes

# Obtain the option prices from simulation values
opt_price_mats <- f_opt_price_simulation(sim_price_sp500 = sim_price_sp500,
                                         sim_vol_vix = sim_vol_vix,
                                         K_vec = K_vec,
                                         T_vec = T_vec,
                                         put=FALSE)

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

```
##           T+1           T+2           T+3           T+4           T+5
## 1  85.93966 110.69740 136.26860 107.01790 132.81728
## 2  90.24098 118.71906 124.10892 126.59144 127.36517
## 3  60.25326  68.45593  83.19676  60.93087  64.90429
## 4 130.13155 163.09257 155.57730 211.29567 212.15179
## 5  79.84446  80.69383  72.68668  85.73559  67.03312
## 6  81.47264  80.36234  75.33654  74.88091 133.99041
```

```
head(opt_price_mats$opt2)
```

```
head(opt_price_mats$opt3)
```

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

```
head(opt_price_mats$opt2)
```

```
head(opt_price_mats$opt3)
```

Distribution of the Profit and Loss for the Book Of Options

Parameters

Hair Parra, Alessio Bessan, Ioan Catalin

Profit at Maturity

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

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

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

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(20,40,40) # maturities
K_vec <- c(1600, 1605, 1800) # Strikes

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

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

```
##           T+1          T+2          T+3          T+4          T+5
## 1 156.9949 187.2824 204.0350 281.2273 266.0588
## 2 152.6936 179.2607 216.1946 261.6537 271.5109
## 3 182.6813 229.5238 257.1068 327.3143 333.9718
## 4 112.8030 134.8872 184.7263 176.9495 186.7242
## 5 163.0901 217.2859 267.6169 302.5096 331.8429
## 6 161.4620 217.6174 264.9670 313.3643 264.8856
```

```
head(PL_mats$PL2)
```

```
##           T+1          T+2          T+3          T+4          T+5
## 1 149.0321 181.7354 200.6217 275.4742 263.0822
## 2 146.0100 176.0373 213.6103 259.5831 270.0157
## 3 170.6694 218.9439 249.4797 315.1517 323.2311
## 4 109.9124 133.6802 183.2563 176.6265 186.4300
## 5 155.4809 210.1011 259.2408 296.3761 323.0949
## 6 152.9400 209.5853 255.8789 303.5546 262.3220
```

```
head(PL_mats$PL3)
```

```
##           T+1          T+2          T+3          T+4          T+5
## 1 36.81158 87.71407 124.4798 179.1517 185.8252
## 2 37.18441 89.35728 131.3917 179.8644 191.9283
## 3 38.91867 93.61705 135.4909 184.5052 195.5563
## 4 30.93695 77.75529 122.3706 150.3484 161.5843
## 5 38.66255 94.03754 137.0303 184.4314 196.5044
## 6 37.34788 93.29921 135.8956 183.4389 186.4738
```

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 <- as.vector(PL_mats$PL1)
sim_pl_opt2 <- as.vector(PL_mats$PL2)
sim_pl_opt3 <- as.vector(PL_mats$PL3)

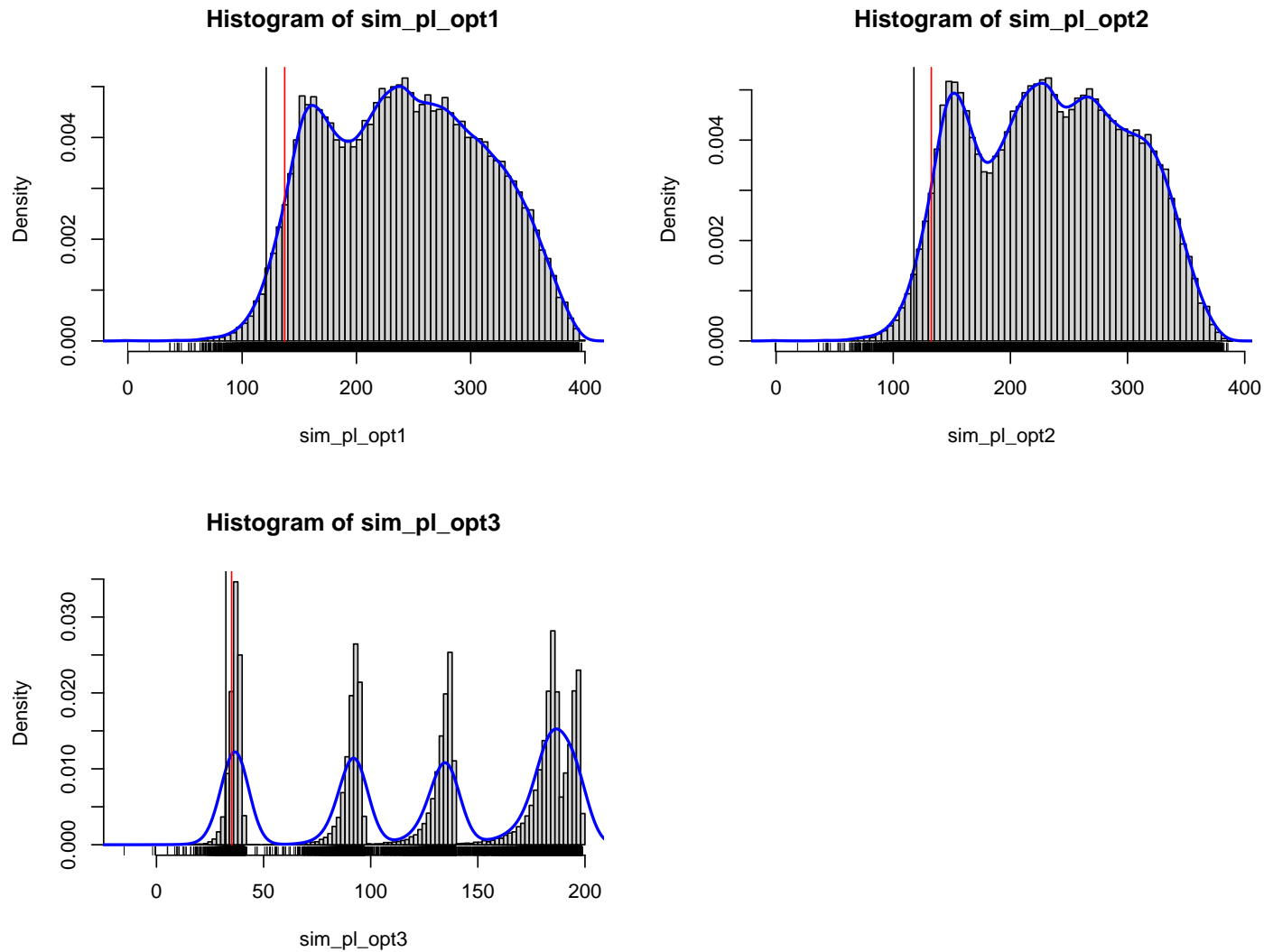
# Compute the 95% VaR and 95% ES
opt1_VaR_ES <- f_VaR_ES(sim_pl_opt1, alpha = 0.05)
opt2_VaR_ES <- f_VaR_ES(sim_pl_opt2, alpha = 0.05)
opt3_VaR_ES <- f_VaR_ES(sim_pl_opt3, alpha = 0.05)

# plot the distribution for each of the options
par(mfrow = c(2,2))
hist(sim_pl_opt1, nclass = round(10 * log(n_sim)), probability = TRUE)
lines(density(sim_pl_opt1), lwd=2, col="blue")
abline(v=opt1_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt1_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt1)

hist(sim_pl_opt2, nclass = round(10 * log(n_sim)), probability = TRUE)
lines(density(sim_pl_opt2), lwd=2, col="blue")
abline(v=opt2_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt2_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt2)

hist(sim_pl_opt3, nclass = round(10 * log(n_sim)), probability = TRUE)
lines(density(sim_pl_opt3), lwd=2, col="blue")
abline(v=opt3_VaR_ES$VaR, col="red") # 95% VaR
abline(v=opt3_VaR_ES$ES, col="black") # expected shortfall
rug(sim_pl_opt3)

```



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

VaR95

Definition

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:

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

First Option

```
opt1_VaR_ES$VaR # first option
```

```
## [1] 137.2302
```

```
opt2_VaR_ES$VaR # second option
```

```
## [1] 132.5263
```

```
opt3_VaR_ES$VaR # third option
```

```
## [1] 35.0578
```

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.

```
# display
opt1_VaR_ES$ES
```

```
## [1] 121.0392
```

```
opt2_VaR_ES$ES
```

```
## [1] 117.6614
```

```
opt3_VaR_ES$ES
```

```
## [1] 32.40935
```

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 requiured 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

$$y_t = \epsilon_t \sigma_t,$$

$$\sigma_t^2 = \omega + \alpha y_{t-1}^2 + \beta \sigma_{t-1}^2$$

$$\epsilon_t \stackrel{i.i.d.}{\sim} \mathcal{N}(0, 1),$$

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
## -----
```

Inspect 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:

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

```
# extract 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
```

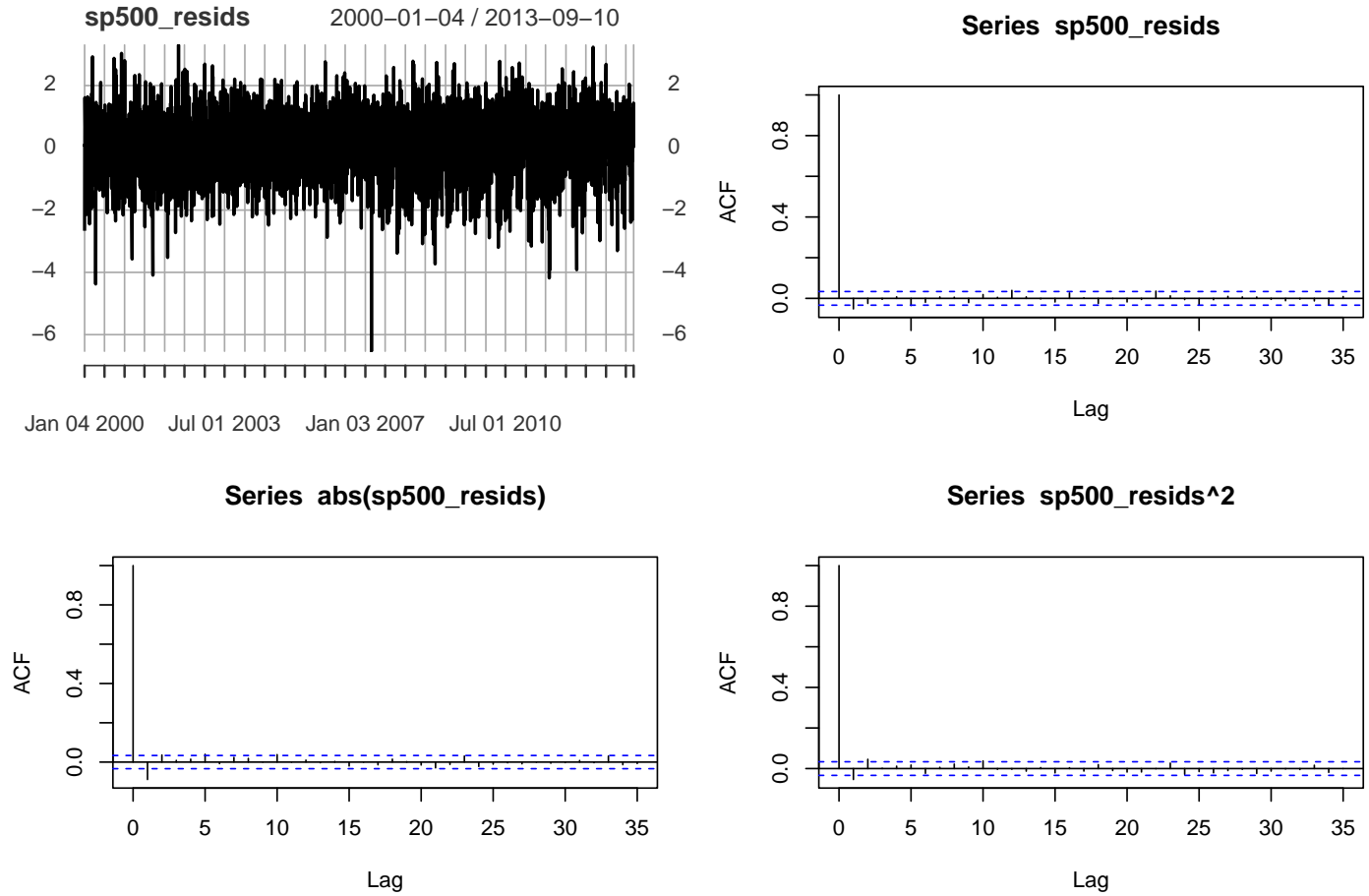
```
# Look at dependence in the residuals
par(mfrow = c(2,2))
```

```
# 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 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
```

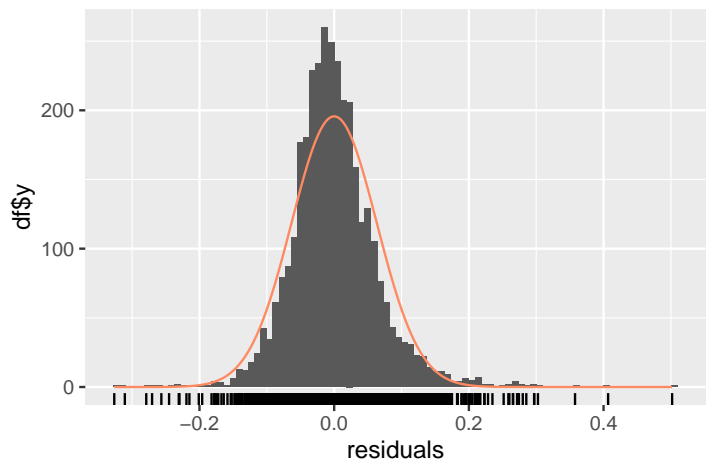
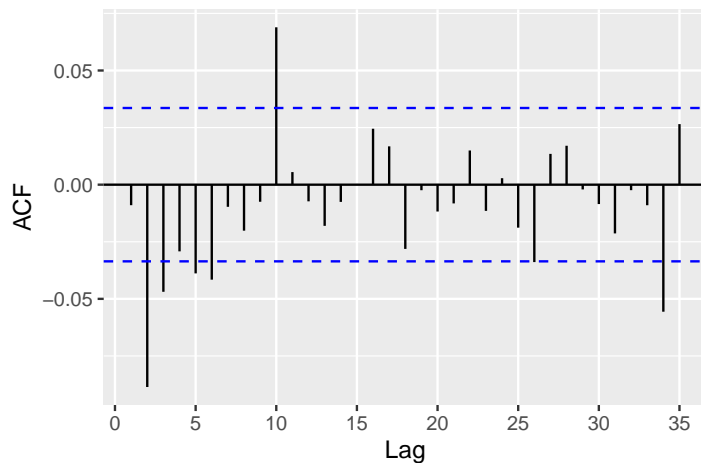
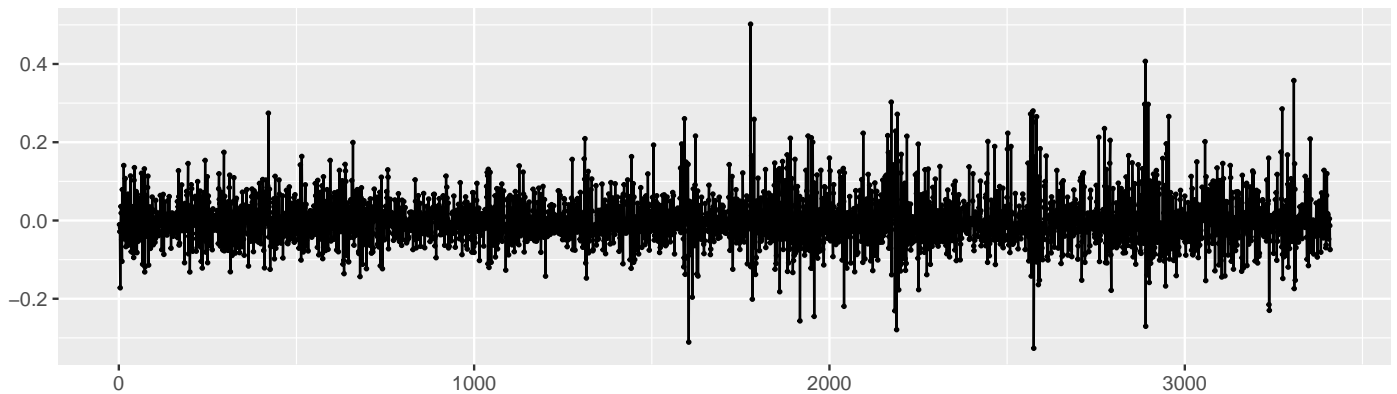
Stationarity of the residuals & underlying normality

```
# 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")
```

Residuals from AR(1)



```
##
## 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:

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

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

Gaussian marginals to the invariants

```
# 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

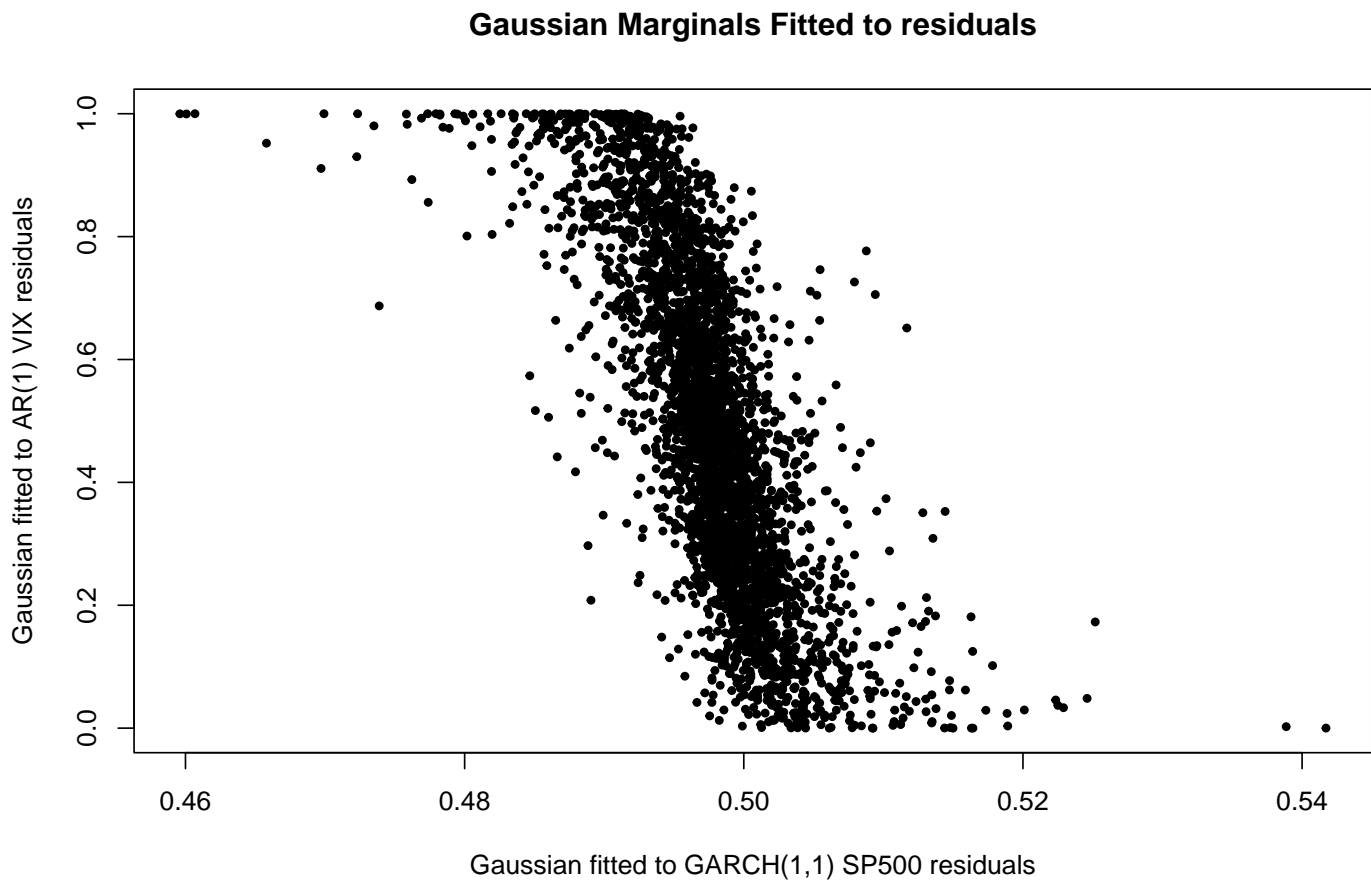
# display parameters
theta1

##          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

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

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

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

```
# 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
```



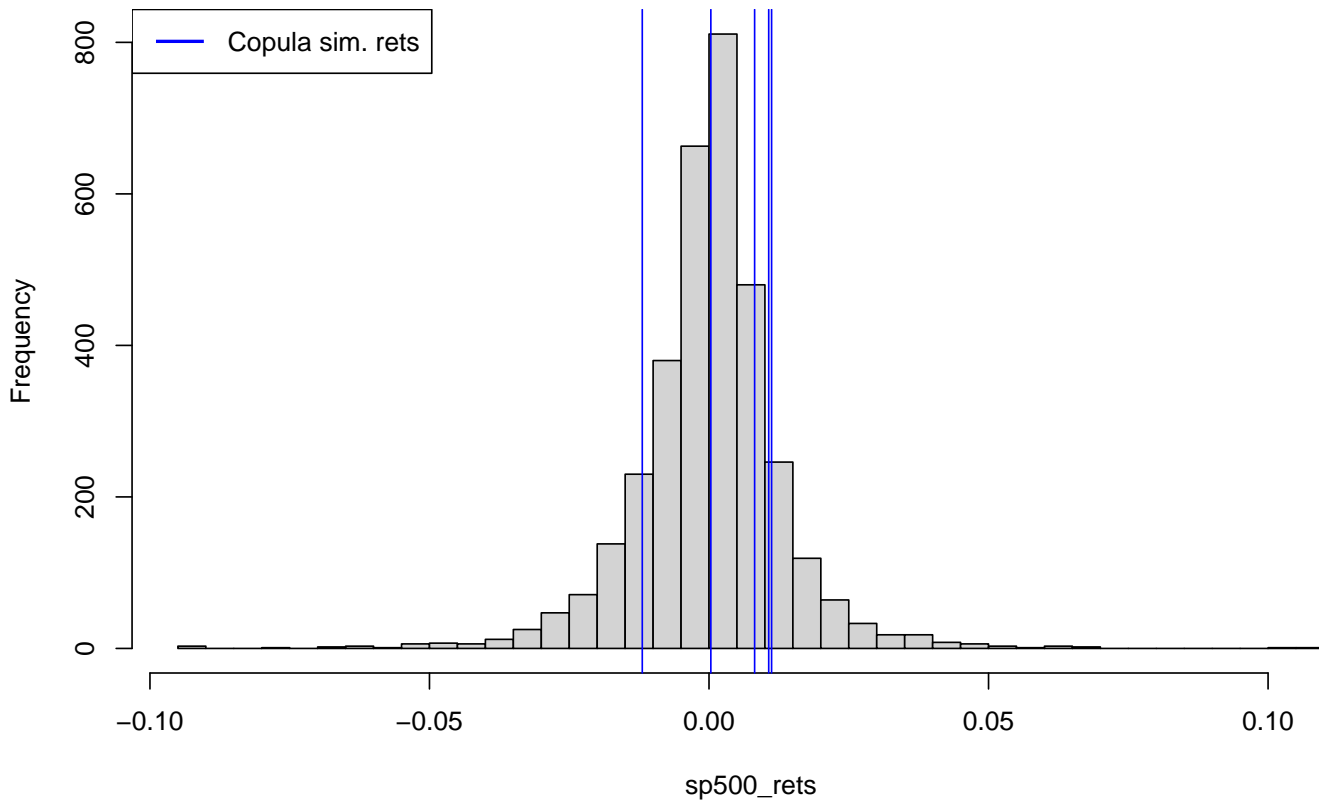
```
# apply to all rows and pack into a matrix
sp500_sim_rets_full <- t(apply(sim_inv_sp500, 1, function(x){f_forecast_y(theta=theta,
                                                                    sig2_prev = sig2_prev,
                                                                    y_prev = 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))
```

Histogram of sp500_rets



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
```

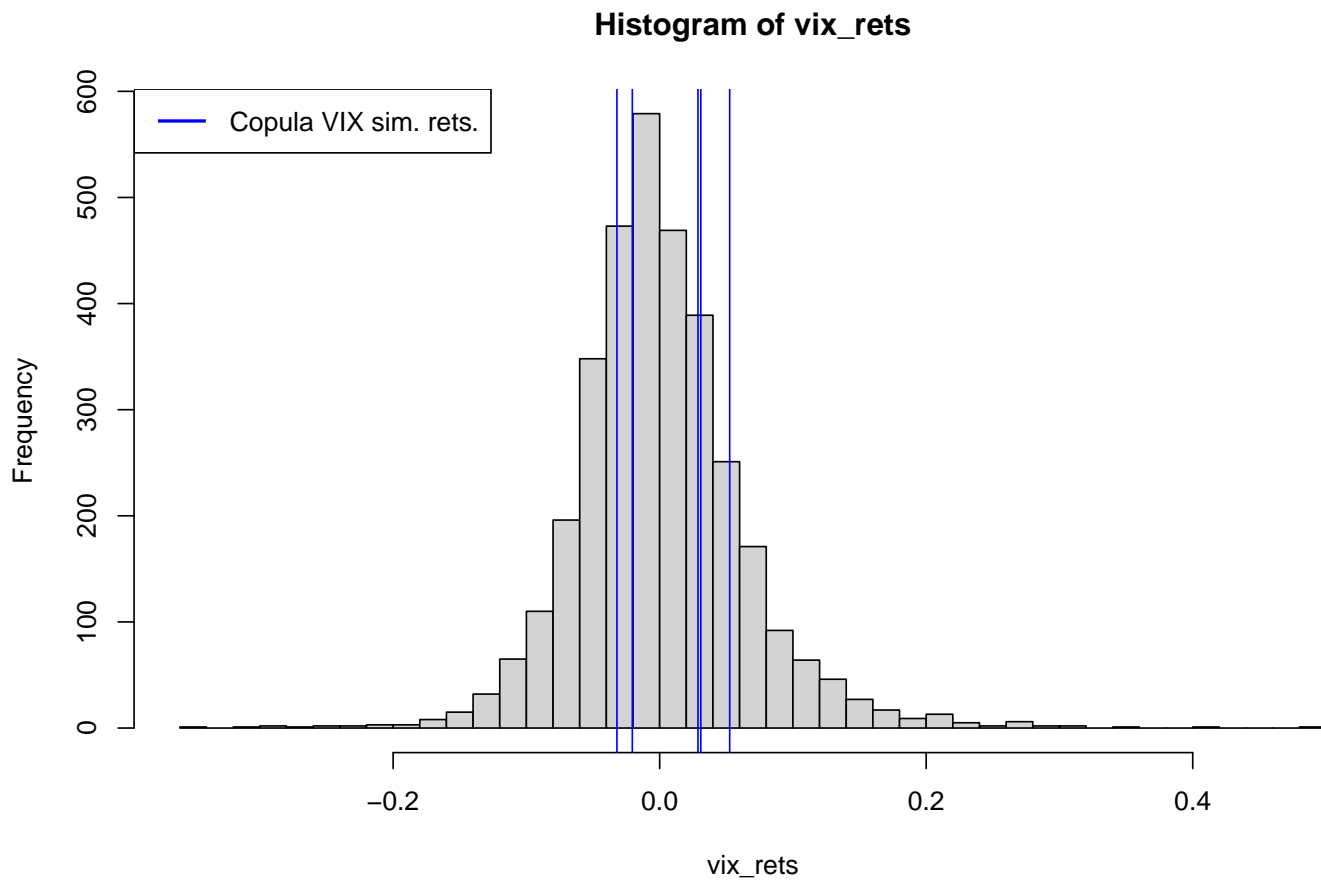
```
## [1] 0.03087567 0.05255314 0.02873398 -0.02051841 -0.03204625
```

```
# 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 simulation)
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 simulated 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
```