The Coca-Cola Company: Stock Analysis and Option Pricing

Applied Statistics for Finance

Andrea Hüscher

April 2020

Contents

1 Introduction **4**

2 Stock analysis **5**
- 2.1 Stock introduction and recent history . . . 5
- 2.2 Charts . . . 6
- 2.3 Risk indexes . . . 7
- 2.4 Check the normality of our distribution . . . 8

3 Parameters estimation **10**
- 3.1 Historical mean and volatility . . . 10
- 3.2 Implied volatility . . . 10
- 3.3 Maximum Likelihood Estimator . . . 11
- 3.4 Final decision on parameter . . . 12

4 European Option Pricing **12**
- 4.1 Black Scholes Formula . . . 12
- 4.2 Monte Carlo Methods . . . 13
- 4.3 Fast Fourier Transform . . . 14
- 4.4 Put-Call parity . . . 15
- 4.5 Greek letters . . . 16
  - 4.5.1 Delta . . . 16
  - 4.5.2 Theta . . . 16
  - 4.5.3 Gamma . . . 17
  - 4.5.4 Rho . . . 17
  - 4.5.5 Kappa . . . 17
  - 4.5.6 Vega . . . 17

5 American Option Pricing **18**
- 5.1 Broadie and Glasserman simulation method . . . 18
- 5.2 Logstaff and Schwartz Least Squares Method . . . 18
- 5.3 Explicit finite-difference method . . . 19
- 5.4 Implicit Finite Difference Method . . . 21

6 Levy Processes **22**
- 6.1 Variance Gamma process . . . 23
  - 6.1.1 Fitting . . . 24
  - 6.1.2 Option Pricing . . . 26
- 6.2 Meixner process . . . 27
  - 6.2.1 Fitting . . . 27
  - 6.2.2 Option pricing . . . 29

7 Multi Asset Options **30**
- 7.1 GBMs correlated through a linear correlation coefficient RHO . . . 31
- 7.2 Copulae . . . 31

8 Multi Asset Options under Lévy Processes **33**
8.0.1 Ballotta Bonfiglioli Model 34
8.1 Lévy Copulae 35

9 My contributions **37**
9.1 Best copula 37
9.2 Relative Strength Index 40

10 Conclusion **41**

1 Introduction

In this project I worked on Coca-Cola stock, starting from analyzing it, pricing options on it and studying its dependence with other stocks. I priced call and put options. In the first part I made a basic stock analysis, then parameters estimations, European and American Option pricing with different methods. In the end I faced the topic of multi asset options, refine the usage of copula and introduce briefly a simple trading indicator.
The data came all from Yahoo Finance.

2 Stock analysis

2.1 Stock introduction and recent history

The Coca-Cola Company (KO) manufactures, markets, and distributes nonalcoholic beverages worldwide. The company primarily produces soft drink concentrates and syrups to be used in carbonated beverages, but selectively produces finished products as well. KO's portfolio also consists of juices, juice drinks, ready-to-drink teas and coffees, and energy and sports drinks. Its diversifed portfolio includes notable brands such as Coca-Cola, Sprite, Fanta, Schweppes, Powerade, Dasani, Costa Coffee, Fuze Tea and many others. The company operates globally in 200+ countries through a network of company-controlled and independent bottling/distribution partners.

In January the company declared a growth of 6% of sales worldwide. It's the first increasing in sales from three years. Strength in 2019 has come from both developed and emerging markets, driven by sparkling sales in North America, including no-sugar versions of the company's popular soft drink brands and smaller package sizes, and pricing in Latin America. Management's value over volume strategy is still in its early innings and continues to drive better alignment across the system. Risks include foreign exchange headwinds, shifting consumer preferences and a slowdown in consumer spending. Furthermore the company plans to cut over 1200 jobs for an estimated cost savings of 5 billions.

Since Coca-Cola sales depends heavily on social events such as concert, cinemas and disco-club, my idea is to analyze this stock always seen as "difensive", and look how COVID-19 pandemic hit its performances and options traded on it.

2.2 Charts

Figure 1: Coca-Cola adjusted price evolution from 2010

In Figure 1 the trend that the Coca-Cola has in the last decade has been displayed. Please note that I used adjusted price, because Coca-Cola is known for always delivering dividends to his shareholders. The red dotted lines are the Bollinger bands, in particular they are the standard deviation of a 20 days Systematic Moving Average.
Then I looked for points where the distribution of the price changes. In particular it happened on 9th October of 2018. You can check it visually, it is the grey vertical line.

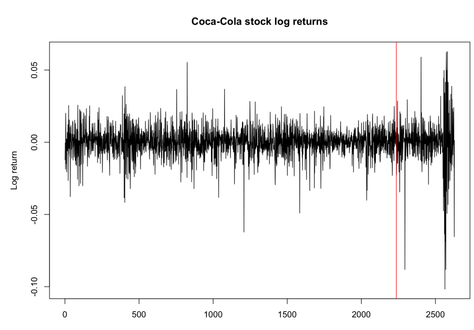

Figure 2: Coca-Cola Log Returns from 2010

In figure 2 we can see the logarithmic returns from 2010 and the same chang-

ing point (now the red vertical line).
Why we use logarithmic returns? Because it is mathematically convenient, in fact we will perform computations for derivatives pricing and this transformation makes the computation easier. They are widely used in financial application because they are time additive (or time consistent). Furthermore there exist a property which states that if log returns are normally distributed then adding this normally distributed variables produces an n period log return which is also normally distributed.

Now let's focus from the smaller time period indicated by the changing point detection.

Figure 3: Coca-Cola October 2018

Here we have a candlestick chart with volumes, a widely used visualization method among traders. It shows the opening, the closing and the interval in which the price traded for every day. In this case if the candle is orange it means the price decreased, if green it means the price increased.

2.3 Risk indexes

I computed the daily Beta of Coca-Cola in relation to the Dow Jones and it resulted about 0.69. It means that if the DJI grows of 100, Coca-Cola grows of 69. It can't be considered a quite low value of beta, so this don't confirm the fact that KO can be considered a "defensive stock". But we have to consider that this value is computed using only the last two years of data more or less.

The VAR at 99% of Coca Cola is about -0.067. This means that historically only on the 1% of cases the daily loss of KO stock has been major than 6.7%.

The Sharpe Ratio during this time span is unfortunately quite low: -0.39. The Sharpe Ratio compares the returns of our stock to the risk-free return (in our case 0.7%, because the 10 years US treasury bond has 0.69% return), and divide this difference to the standard deviation of our stock.

$$S = \left(\frac{R_p - R_f}{\sigma_p} \right) \tag{1}$$

Beta	0.6890417
VAR 99%	-0.0672904
Sharpe Ratio	-0.39538211

2.4 Check the normality of our distribution

Unfortunately the distribution of log returns in our time span, we can see it in many ways:
-Density in figure 4
-QQ-plot in figure 5
-Descriptive statistics

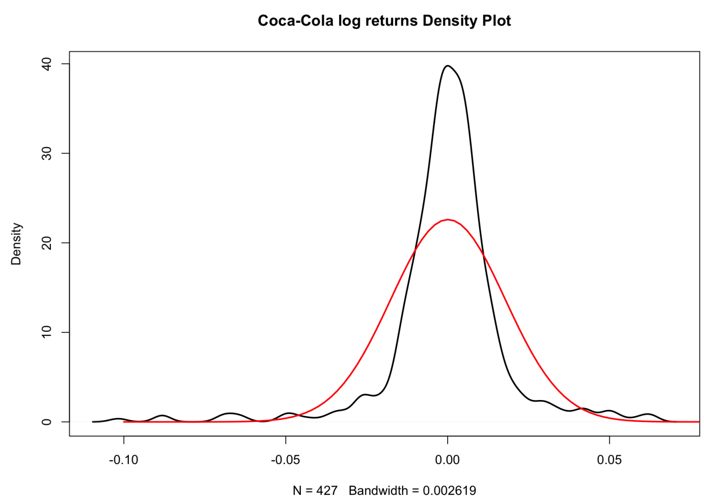

Figure 4: Coca-Cola log returns density function, compared with the normal distribution with same parameters (in red)

In fact here you can see that the actual distribution is steeper and has bumpier tails, w.r.t. the normal case.

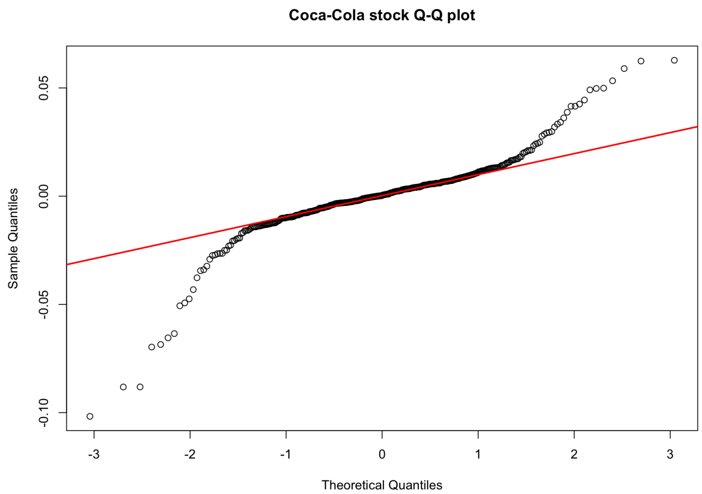

Figure 5: Coca-Cola log returns QQ-plot

In figure 5 the QQ-plot clearly signals the presence of fat tails.

Number of observations	441
Mean	0.0000617
Median	0.0004174
Variance	0.0003079
Standard Deviation	0.0175483
Skewness	-1.0527258
Kurtosis	7.5592460
Min	-0.1017280
Max	0.0627830

In these table we can find some interesting statistics that help us to confirm that the distribution isn't normal.
In fact we have an high kurtosis, note that in the normal case the kurtosis should be about 3. Furthermore we have a negative skewness which tells us that the distribution is more oriented to the losses.

The fact that our distribution isn't Gaussian is very important, because all methods for option pricing based on the concept of Brownian Motion don't suit well our data. This is very important to keep in mind.

3 Parameters estimation

3.1 Historical mean and volatility

First of all let's simply compute the historical volatility and mean both annualized. The results are:

Historical	
Mean	0.05434958
Volatility	0.2785712

3.2 Implied volatility

Implied volatility is a measure of market expectations regarding the asset's future volatility.
In particular it is computed starting from the Black and Scholes formula for determine the price of an European option, in this specific case a call:

$$P_t = C\left(t, S_t\right) = S_t \Phi\left(d_1\right) - e^{-r(T-t)} K \Phi\left(d_2\right) \tag{2}$$

with

$$d_1 = d_2 + \sigma\sqrt{T-t}, \quad d_2 = \frac{\ln\frac{S_t}{K} + \left(r - \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}}$$

where
Φ = cumulative distribution function for a Standard Gaussian
S_t = stock price at price t
r = risk-free rate
T = expiration date
k = strike price
σ = standard deviation of log returns
Now instead of deriving the price of the option, we will compute the implied volatility of KO stock. How? Taking all the parameters we need (except, of course, the standard deviation), and the price for which an option on KO is traded on the market and then derive its implied volatility.

Applying this methods I got interesting results. The first try I did is with the following parameters:
$S_0 = 44.82$
$k = 35$
$T = 14/252$
$P_t = 9.4$
$r = 0.07$
and the value I got was:

$$\sigma = 9.903487e - 17$$

A very low result. It can be caused from various reasons. The most important one is the fact that nowadays the Federal Reserve, pushed by the Trump administration, is taking all action possible sustaining the market in order to avoid further significant downturn in the American stock market.

Other important consideration have to be done on the parameters set for the option. In fact such a low value of volatility can be imputed to the fact that the option is deep in the money. If I change the strike price to 44.5, and T to 8/252 I get:

$$\sigma = 0.1653224$$

Again a too much low value w.r.t. the historical volatility, but much more acceptable.

3.3 Maximum Likelihood Estimator

The likelihood is the probability mass function, or density, for the observed data x (identically independently distributed) viewed as a function of θ:

$$L_n(\theta) = L_n(\theta \mid x_1, \ldots, x_n) = \prod_{i=1}^{n} f(x_i; \theta) \tag{3}$$

In order to compute the maximum likelihood estimator (MLE) it is more convenient to consider the log-likelihood, simply the logarithm of the likelihood:

$$l_x(\theta) = \log \{L_n(\theta)\} \tag{4}$$

The MLE is nothing else than the value of θ in the parameter space Ω that maximizes $l_x(\theta)$, namely:

$$\hat{\theta} = \arg\max_{\theta \in \Omega} \{l_x(\theta)\} \tag{5}$$

Fitting the MLE to my data I applied different optimization algorithms.
First I tried the Limited Memory Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS) which unfortunately doesn't seem to reach convergence. In fact for the estimation of μ it returns the starting point, for the standard error it returns a smaller value w.r.t. the historical, but acceptable:

MLE "L-BFGS-B"	
Mean	0.010 (exactly the starting point)
Volatility	0.02018396

In the following tables you can find the results I got from the other algorithms:

MLE "BFGS" (default algorithm)	
Mean	0.0000581358
Volatility	0.0175459411

MLE conjugate-gradient "CG" algorithm	
Mean	0.00004380472
Volatility	0.01756829

MLE Simulated-annealing "SANN" algorithm	
Mean	0.0003103185
Volatility	0.0171594057

3.4 Final decision on parameter

Given the different results got in the volatility estimation I decided to rely on the historical one.

4 European Option Pricing

4.1 Black Scholes Formula

As I already explained in paragraph 3.2 the BS formula is used to price an European option. In our case I wanted to compute a call option so the formula is (2).
It is the solution of:

$$C(t,x) = e^{-r(T-t)} Ef\left(Z_T^{t,x}\right) \tag{6}$$

with:

$$Z_T^{t,x} = x \exp\left\{\left(r - \frac{1}{2}\sigma^2\right)(T-t) + \sigma\left(B_T - B_t\right)\right\}, \quad T > t \tag{7}$$

where B is a Brownian Motion.
By inspecting the formula (2) we see that the value of the option is positively correlated with the standard deviation, so the bigger the volatility of the underlying asset, the higher the value of the option will be. Furthermore it is positively correlated with the initial stock price, but negatively correlated with the strike price.
In fact the BS formula can be divided in two parts:
The first one:

$$S_t \Phi\left(d_1\right) \tag{8}$$

which is what an investor gets from exercising the call option.
the second one:

$$-e^{-r(T-t)} K \Phi\left(d_2\right) \tag{9}$$

which is what an investor has to pay to exercise the call. In fact $e^{-r(T-t)}$ is the factor that discounts back the strike price.
Please recall that the payoff of an European call option is $f(x) = \max(0, x-K)$ where x is the price of the underlying asset at the expiration date T.

Setting: $S_0 = 44.82$; $K = 42$; $T = 8/252$; $r = 0.07$; volatility = historical; the price is:

$$p = 3.000418$$

Not a bad result given that the option is priced on the market at 3.1.

4.2 Monte Carlo Methods

Monte Carlo simulation is a method of estimating the values of an unknown quantity using inferential statistics.
It consists in drawing pseudo random numbers from a given distribution. More specifically we are interested in computing the expected value of a given function ($Eg(X)$) on a random variable (X). Given the pseudo random numbers simulated $(x_1, \ldots, x_n)$ from our distribution we can approximate the expected value with the sample mean of the function($g(x_i)$), namely:

$$Eg(X) \simeq \frac{1}{n} \sum_{i=1}^{n} g(x_i) = \bar{g}_n \tag{10}$$

This equation (10) holds by the Law of Large Numbers.
Furthermore for the CLT(Central Limit Theorem):

$$\bar{g}_n \xrightarrow{d} N\left(Eg(X), \frac{1}{n} \operatorname{Var}(g(X))\right) \tag{11}$$

We use Monte Carlo simulations when explicit formula for option pricing aren't available. In fact now estimating the random variable $Z_T^{t,x}$ we use the following equation:

$$Z_T^{t,x} = x \exp\left\{\left(r - \frac{1}{2}\sigma^2\right)(T-t) + \sigma\sqrt{T-t}u\right\} \tag{12}$$

with $u \sim$ standard Normal.
The difference between BS is that here the std is multiplied by $\sqrt{T-t}u$.
Now we only need to simulate M observation starting from the standard Normal. Then for each simulated value of the random variable, apply the payoff function, average these values (according to LLN) and discount them.

As said above MC methods rely on the Law of Large Numbers, so when we increase the number of simulation we should obtain a value which tends to the one estimated via BS, because the confidence interval tends to reduce.

Number of simulation	1000	50000	1000000
Monte Carlo	2.990866	3.000928	3.000779

Black Scholes	3.000418

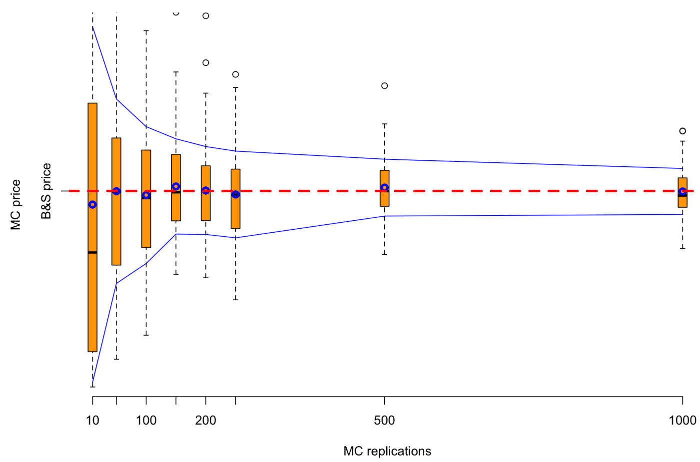

Figure 6: Speed of convergence MC Methods

One problem emerging in MC methods is that to have significant results we have to reduce the variability (MC intervals aren't relevant if the variance is too high), and reduce the number of replications of simulations. How to do this? By Variance Reduction Techniques, such as: preferential sampling, control variables, and antithetic sampling. In our case we applied the antithetic sampling.

4.3 Fast Fourier Transform

This method is based on the concept of the characteristic function, which for a r.v. X is:

$$\varphi(t) = E\left\{e^{itX}\right\} \tag{13}$$

The formula for pricing an European Call option is:

$$C(K,T) = S_0\Pi_1 - Ke^{-rT}\Pi_2 \tag{14}$$

$\prod_1$ is the probability of the option to end in the money, and $\prod_2$ is the probability to finish out of the money. These two values can be computed via the inversion of the characteristic function:

$$\begin{aligned}
\Pi_1 &= \frac{1}{2}\frac{1}{\pi}\int_0^\infty \mathrm{Re}\left(\frac{\exp(-iu\log K)E\left(\exp\left(i(u-i)\log S_T\right)\right)}{\mathrm{iuE}\left(S_T\right)}\right)\mathrm{d}u \\
&= \frac{1}{2}\frac{1}{\pi}\int_0^\infty \mathrm{Re}\left(\frac{\exp(-iu\log K)\varphi(u-i)}{\mathrm{iu}\,\varphi(-i)}\right)\mathrm{d}u \\
\Pi_2 &= \frac{1}{2}\frac{1}{\pi}\int_0^\infty \mathrm{Re}\left(\frac{\exp(-iu\log K)E\left(\exp\left(iu\log S_T\right)\right)}{iu}\right)\mathrm{d}u \\
&= \frac{1}{2}\frac{1}{\pi}\int_0^\infty \mathrm{Re}\left(\frac{\exp(-iu\log K)\varphi(u)}{iu}\right)\mathrm{d}u
\end{aligned} \tag{15}$$

Supposing S_t is a Geometric Brownian Motion, the characteristic function of this GBM becomes:

$$\varphi(u) = \exp\left(iu\left(\mu - \frac{1}{2}\sigma^2\right) - \frac{\sigma^2 u^2}{2}\right) \tag{16}$$

Then the Fast Fourier Transform approximation is:

$$C_T(k) \approx \frac{e^{\alpha k}}{\pi}\sum_{j=1}^{N} e^{-iv_j k}\psi_T\left(v_j\right)\eta \tag{17}$$

It approximates the price of the call.

In my case I got:

$$p = 2.931343$$

It slightly underestimates the market price.

4.4 Put-Call parity

Now we return for a moment to the framework of Black Scholes. The Put-Call parity is a relationship between the prices of European put and call options that have the same strike price and time to maturity. In this model let's consider the payoff of a Put Option:

$$f(x) = \max(0, K - x) \tag{18}$$

then the price of a put is:

$$P_t = C\left(t, S_t\right) = e^{-r(T-t)}K\Phi\left(-d_2\right) - S_t\Phi\left(-d_1\right) \tag{19}$$

If I keep the same values (please look at the bottom part of page 8) of the last call options I traded, I expect that the price of a put in this framework would be close to 0. Because it is slightly in the money considering it a call, but if you consider it as a put is out of the money at this time.

$$p_{put} = 0.08718775$$

The put and call parity essentially shows that the value of a put with same parameters can be deduced starting from a call, and vice versa of course.

$$c + Ke^{-rT} = p + S_0 \tag{20}$$

Which in our case is:

$$\begin{gathered} 3.000417 + 42e^{-0.07*8/252} = 0.08718775 + 44.82 \\ 44.90719 = 44.90719 \end{gathered} \tag{21}$$

4.5 Greek letters

4.5.1 Delta

The Delta of an option is the rate of change of the option price w.r.t. the price of the underlying asset. Substantially it is the slope of the curve that relates the option price to the underlying asset price, namely:

$$\Delta = \frac{\partial P_t}{\partial S_t} \tag{22}$$

In our call option it is:

$$\Delta = 0.91$$

It means that if the underlying asset (the Coca-Cola share) slightly varies, the value of the option changes of 90%, so our option is really sensitive.

4.5.2 Theta

The theta of an option is the rate of change of the value of the option w.r.t. the time. Intuitively it is the derivative of the price w.r.t. time. In our case of a call option:

$$\theta = C_t(t, x) = -rKe^{-r(T-t)}\Phi\left(d_2\right) - \frac{\sigma x}{2\sqrt{T-t}}\Phi\left(d_1\right) < 0 \tag{23}$$

Note that it is always negative, since as time passes, the value of the call tends to decrease.

In our case it is:

$$\theta = -8.066349$$

If time is measured in days, theta is the change in the option value when each day passes.

4.5.3 Gamma

The Gamma represents the rate of change of the Delta of a specific option w.r.t. the price of the underlying asset. Essentially it is the sensitivity of the hedge ration w.r.t. the underlying asset, namely:

$$\gamma = C_{xx}(t,x) = \frac{\partial}{\partial x} C_x(t,x) = \frac{1}{\sigma x \sqrt{T-t}} \Phi\left(d_1\right) > 0 \tag{24}$$

In our case:

$$\Gamma = 0.06931098$$

So if the price of KO shares changes, the hedge ratio varies of about 7%.

4.5.4 Rho

The Rho of an option is the rate of change of the value of the option w.r.t. the interest rate, for an European call:

$$\rho = \frac{\partial C(t,x)}{\partial r} = K(T-t)\Phi\left(d_2\right) e^{-r(T-t)} > 0 \tag{25}$$

In our specific case it is:

$$\rho = 1.208131$$

This means that a 1% change in the risk free rate (in our case, the risk free rate is determined by a 10-y US Treasury Bond) reflects a change of about $0.01 * 1.208131 = 0.01208131$ on the price of our option.

4.5.5 Kappa

The Kappa of an option is the sensitivity of the option price w.r.t. the variation of the exercise price, namely it is the derivative of the option price w.r.t. the exercise price. For an European call:

$$\kappa = \frac{\partial C(t,x)}{\partial K} = e^{-r(T-t)}\left(\Phi\left(-d_2\right) - 1\right) < 0 \tag{26}$$

4.5.6 Vega

The Vega is the change of the price of an option w.r.t. the volatility of the underlying asset:

$$vega = \frac{\partial C(t,x)}{\partial \sigma} = x\sqrt{T-t}\Phi\left(d_1\right) > 0 \tag{27}$$

In our case it is:

$$vega = 1.231322$$

5 American Option Pricing

Pricing an American option is more difficult since this kind of contract can be exercised any time the investor wants, not only on the expiry date as the European one.

In the following parameters we will face different methods for pricing these derivatives, starting from techniques that rely on Monte Carlo simulations, to finite difference methods.

5.1 Broadie and Glasserman simulation method

In their paper of 1997 Broadie and Glasserman stated that it isn't possible to compute an unbiased estimator for the price of an American call option by simulation.
They started with the objective of estimate:

$$C = \max_{\tau} E\left\{e^{-r\tau}\max(S-k,0)\right\} \tag{28}$$

Having a grid of times $0 = t_0 < t_1 < \cdots < t_d = T$ we can simulate $S_1 = S_{t1}, S_2 = S_{t2}, \ldots, S_T = S_{td}$ starting from S_0 and estimate the option price with the formula (28) but discretized:

$$C = \max_{i=0,\ldots,d} E\left\{e^{-rt_i}\max\left(S_i-k,0\right)\right\} \tag{29}$$

With this simulation a tree has been created since for each step b new branches are created.

In our specific case we will create a tree with 3 branches for each node and with a total of 3 time step ($b=3$, $d=3$). After having simulated the tree we can introduce two estimators for our confidence interval. They are both biased but asymptotically unbiased.

The results I got are:

$$\Theta = 3.23075; upper$$

$$\theta = 2.855966; lower$$

This confidence interval is quite good since it contains the real price of the call option at the time (3.1).

5.2 Logstaff and Schwartz Least Squares Method

In this method a key component is the estimation of the continuation value by regression via Least Squares. Instead of construct an entire tree, this method requires the simulation of one path on a grid of times t_i, with $i = 0, 1, ..., d$.
Cosider $V_i(x)$ the value of the option and $f_i(x)$ the payoff function at time t_i given $S_{ti} = x$ The continuation value at time t_i given $S_{ti} = x$ is

$$C_i(x) = E\left\{V_{i+1}\left(S_{ti+1}\right) \mid S_{ti} = x\right\} = \sum_{r=1}^{M} \beta_{ir}\psi_r(x)$$

for some basis functions ψ_r and coefficients $i_r, r = 1, \ldots, M$

Those β coefficients are estimated via Least Squares method using values $(S_{ti}, V_{i+1}(S_{ti+1}))$. Thus, the continuation value is estimated as

$$\hat{C}_i(x) = \hat{\beta}_i' \psi(x) = \sum_{r=1}^{M} \beta_{ir} \psi_r(x) \tag{30}$$

with

$$\hat{\beta}_i' = \left(\hat{\beta}_{i1}, \ldots, \hat{\beta}_{iM}\right), \quad \psi(x) = (\psi_1(x), \ldots, \psi_M(x))'$$

Basically the algorithm consists in:
(i) simulate n independent paths on the grid of times
(ii) at $t = T$ set $\hat{V}_{tj} = f_d(S_{tj}), j = 1, \ldots, n$
(iii) for $i = d-1, \ldots, 1$
specify the l of paths in the money
discount the value $\hat{V}_{i+1,j}, j \in I$ to input in the regression
given the estimates $\hat{V}_{i+1,j}$, run regression to get $\hat{\beta}_j$
estimate the continuation value as in (30)
if $f_i\left(S_i^j\right) \geq \hat{C}_i\left(S_i^j\right)$, set $\hat{V}_{ij}\left(S_i^j\right) = f_i\left(S_i^j\right)$ else $\hat{V}_{ij} = \hat{V}_{i+1,j}$
(iv) calculate $\left(\hat{V}_{11} + \cdots + \hat{V}_{1n}\right)/n$ and discount it to get $\hat{V}_0$

Then the discounted value can be expressed in terms of an expanztion in a proper L^2 space. In fact Longastaff and Schwartz propose this approximation formula:

$$F(t_{k-1}) = \sum_{i=0}^{M} a_j L_j(X) \tag{31}$$

Please note that in this case we are going to compute the value of our put option. The result is:

$$p_{put} = 0.08739784$$

Which we consider not a bad result, since it is slightly higher than the value for the European put. In fact American options usually are more expensive due to the fact that can be exercised any time.

5.3 Explicit finite-difference method

Finite difference methods are based on the Black and Scholes inequality for an American put option:

$$rC(t,x) \leq C_t(t,x) + rxC_x(t,x) + \frac{1}{2}\sigma^2 x^2 C_{xx}(t,x) \tag{32}$$

It is essentially an optimal stopping-time problem, understand if the current price of the underlying asset (in our case KO shares), is below the efficient frontier (f_t):

$$S_t < f_t$$

This kind of techniques tries to solve (32) using numerical arguments.
They consist in taking the time and divide it in N parts, and take the price of the underlying asset and divide it in M parts. Then construct a Cartesian plan composed by $(N+1)*(M+1)$ elements, with time on the x axis and underlying asset on the y axis. Each point on the grid represents a specific value for the option given that time and that value of the underlying asset.
It is important to establish the right minimum and maximum possible values of the underlying asset. In our example considering that the last value of Coca-Cola was 44.82 and the option expire in 8 days we didn't choose a too wide interval: minimum 38 and maximum 50 (because in 8 days we don't expect a big correction, consider that the VAR at 99% was about 6% for the losses).

More specifically explicit finite-difference methods estimate each point of this grid ($C_{i,j}$, with $i = 0, 1, ..., N$ because it counts the time, and $j = 0, 1, ..., M$ because it counts the underlying asset price) backward, so starting from the furthest point w.r.t. time and then come back. In fact the last point on the right of the grid are the price of the European option given the value of the underlying asset.

$$C_{i,j} = a_j^* C_{i+1,j-1} + b_j^* C_{i+1,j} + c_j^* C_{i+1,j+1}$$

with

$$\begin{aligned} a_j^* &= \tfrac{1}{1+r\Delta t}\left(-\tfrac{1}{2}rj\Delta t + \tfrac{1}{2}\sigma^2 j^2 \Delta t\right) \\ b_j^* &= \tfrac{1}{1+r\Delta t}\left(1 - \sigma^2 j^2 \Delta t\right) \\ c_j^* &= \tfrac{1}{1+r\Delta t}\left(\tfrac{1}{2}rj\Delta t + \tfrac{1}{2}\sigma^2 j^2 \Delta t\right) \end{aligned}$$

Our goal is to estimate the price of the American put option at time 0:

$$C(0, S_0)$$

The uppermost row of our grid is composed by the minimum payoff: 0, because there there is the maximum value of our underlying assets. Given that $S_{max} > K$ then the payoff is always 0:

$$C_{i,M} = 0$$

On the other hand, the lowest row of the grid represents the highest payoff possible since in that row lay the minimum value (S_{min}) of the underlying asset:

$$C_{i,0} = K - S_{min}$$

Let's check our grid:

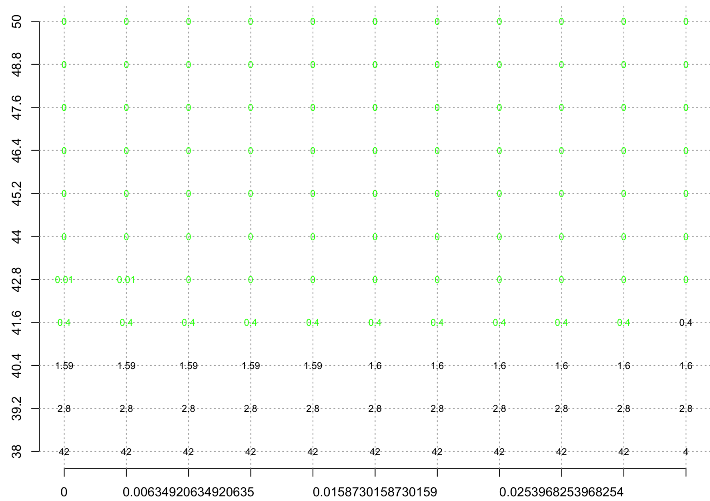

Figure 7: Explicit Finite Difference Method grid for KO American put option

The positive green values represent the parts when it is convenient to exercise before the expiry date the option. On the contrary the black ones are the ones for which the value of the American option is less than the European, so it isn't convenient to exercise those.
Please note that there is an error which I couldn't be able to find in the code, for which the lowest row has value the strike price and in the end the highest payoff, all components should be equal to 4, because, as I already have explained, it is the highest payoff possible given this framework.
In evaluating the option I got a strange value: "numeric(0)" which is some kind of error, probably given to the fact that in reality the right price of the option is close to 0 (as seen previously). Sincerely on this justification I am not dead sure, because even changing the value of the strike price, putting the put option a lot in the money, it returns the same issue.
To notice the fact that usually this method lead to instability and strange value due to the fact that it doesn't fully respect the Markovianity property (it works backward). In fact sometimes it can also return negative values of the option, which is completely not explainable.

5.4 Implicit Finite Difference Method

This method is the only one we face that respects the Markovianity property which, in simply words, states that the future values of a stochastic process depends only on the last value observed of the process itself (for example the Martingale).

For this reason it should overcome to the instability problem of the explicit finite difference method, because it estimates the value of the option starting from t to estimate the value for $t+1$. This doesn't lead to the possibility to have negative values, so it is consider a much better approach.

Checking our grid we see that we only have a slightly change of some values inside the grid, more specifically in the left part of the third row starting from the bottom:

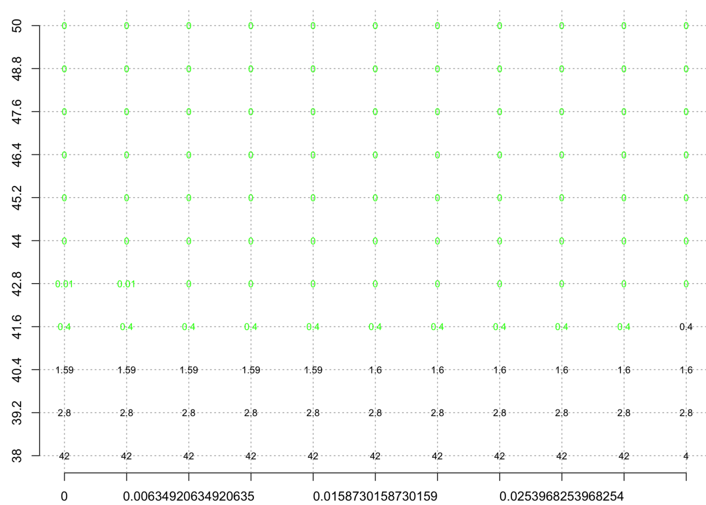

Figure 8: Implicit Finite Difference Method grid for KO American put option

6 Levy Processes

As explained in paragraph 2.4 the GBM isn't a good representation of the reality since the distribution of our data isn't Gaussian. We need to look for a process that suits better our data: a possible solution can be the class of Lévy processes.

You can think of them as GBM with a drift and a "jump" component. This last component is the one that characterizes this kind of process. It is a composed Poisson process and is called "Lévy measure", it is consituted by two factors: one component of "uncertainty", the Poisson distribution, which describes the number of jumps; the other is the dimension of each jump, and this is has to be set, and determines which kind of Lévy process is.
A stylized version of a Lèvy process can be:

$$Z_t = X_t + M_t \tag{33}$$

with

$$X_t = \mu t + \sigma B_t, \quad \mu \in R, \quad \sigma \geq 0$$

where B_t is a Brownian motion and

$$M_t = \sum_{i=0}^{N_t} Y_{\tau i} - \lambda t E\left(Y_{\tau i}\right), \quad \lambda \geq 0$$

with M_t a Poisson process and $Y_{\tau i}$ a sequence of i.i.d. random variables which tells how big are the jumps.
A Lévy process (Z_t) is defined if these properties are satisfied:

(i) Z_t has independent increments, i.e. $Z_t - Z_s$ is independent of $\mathcal{F}_s$, for any $0 \leq s < t \leq T$
(ii) Z_t has stationary increments, i.e. for any $0 \leq s, t \leq T$ the distribution of $Z_{t+s} - Z_t$ does not depend on t
(iii) Z_t is stochastically continuous, i.e. for every $0 \leq t \leq T$ and $\epsilon > 0$, we have that

$$\lim_{s \to t} P\left(|Z_t - Z_s| > \epsilon\right) = 0$$

with $\mathcal{F}_s$ a filtration up to time s

Why this kind of process is better than the GBM in our case? Because now we have the drift component which is deterministic, the diffusive component has always followed the BM, but the jump component is a distribution that is never Gaussian. It can have asymmetry, kurtosis. So it could compensate the problem that we had with GBM, which doesn't take into account asymmetry and kurtosis.
All these information can be synthetized in the Lévy triplet:
1. drift term b
2. diffusion coefficient c
3. Lévy measure (the jump component) ν

6.1 Variance Gamma process

As previously said the jump's size is established by a density function that we decide. In the case of Variance Gamma process it is the Gamma distribution.

$$\tilde{Z}(t) = \tilde{X}(t; \sigma, \nu, \theta) = \tilde{B}(\gamma(t; 1, \nu); \theta, \sigma) \tag{34}$$

It can be considered a pure jump process, because it doesn't have a diffusion component. In fact it can be represented via the Lévy triplet:

$$(b, 0, \nu)$$

Basically it is controlled by three parameters:
a) σ which is the volatility of BM;
b) ν which is the variance rate of the gamma time change;
c) θ which is the drift in the BM.

So the skewness is controlled by θ, while the kurtosis by ν.
In order to estimate the stock price, we have to adapt the original Black Scholes model to the VG process:

$$\tilde{S}(t) = S(0)\exp\left(mt + \tilde{X}\left(t;\sigma_S,\nu_S,\theta_S\right) + \omega_S t\right) \tag{35}$$

Where m is the mean rate stock return under the statistical probability measure. And ω_S guarantees the martingale property:

$$\omega_S = \frac{1}{\nu_S}\ln\left(1 - \theta_S\nu_S - \frac{\sigma_S^2\nu_S}{2}\right) \tag{36}$$

Then to extrapolate the right price we have to perform some transformations that we will see later.

6.1.1 Fitting

Now we have to check if VG fits good or not our data. For convenience I only used the Broyden–Fletcher–Goldfarb–Shanno algorithm because it is the default one. I got the following results:

Parameters	
C	0.0001918398
σ	0.01661524
θ	0.0003649192
ν	1.374867

Monte Carlo simulation for the returns using the parameters I have just estimated:

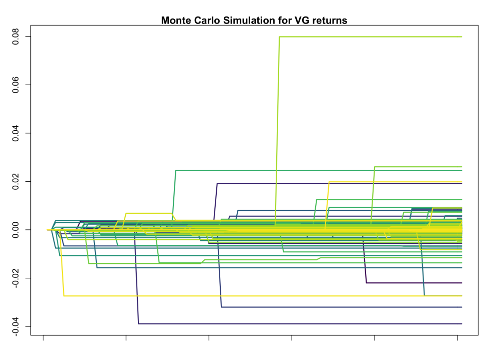

Figure 9: Monte Carlo simulation for a Variance Gamma process

Then after sorting the data let's visually check the Q-Q plot and see how well this process suits our data:

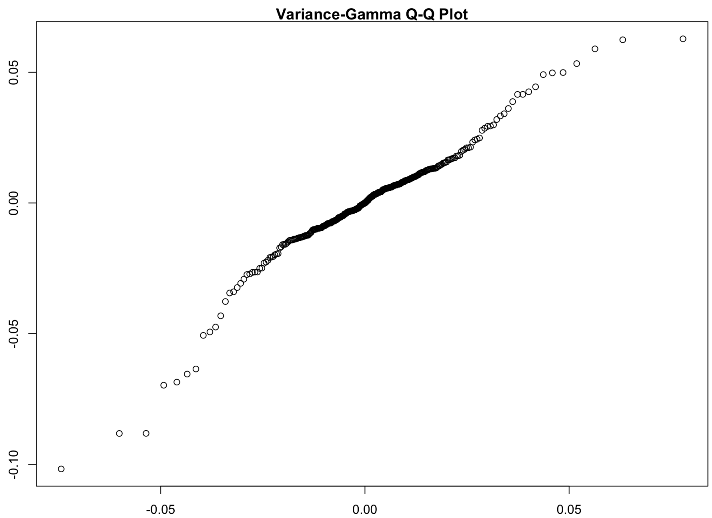

Figure 10: Q-Q plot for VG

It is way better than the Q-Q plot following normal assumption (Figure 5, pag. 6), in fact it is more linear. This fact indicates that the process suits quite well our data.

We have a confirmation of this even after looking at the density and log-density plots (Figure 11)

Now let's check it with hypothesis testing by checking the p-value of a Chi squared test:

$$pvalue = 0.2451$$

It is far bigger than 5%, so we can't reject the H_0 which states that the Variance Gamma is a good fit.

Let's double check this result with Kolmogorov-Smirnov test:

$$pvalue = 0.226$$

Still high p-value so the result is confirmed.

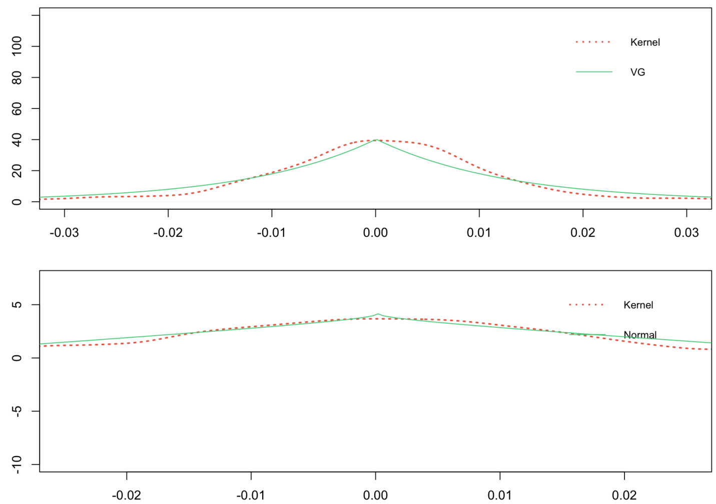

Figure 11: Density and log-density comparison

6.1.2 Option Pricing

As I anticipated before, in order to give financial sense to Lévy processes the first thing we have to do is esponentially transform them:

$$S(t) = S(0)e^{Z(t)}$$

so:

$$S(0) = S(0)e^{Z(0)} = S(0)e^{0}$$

Then, according to the Girsanov Theorem, what we have to do is passing from the physical parameters of our process to the risk neutral ones in two possible ways:
1)Esscher transform

$$f_\theta(x) = \frac{\exp(\theta x) f(x)}{\int_R \exp(\theta x) f(x) dx} \tag{37}$$

2)Mean corrcting martingale (which is the only one we use in the VG).

$$S_t^{RN} = S_0 \exp\left(X_t\right) \frac{\exp(rt)}{E\left[\exp\left(X_t\right)\right]} \tag{38}$$

The price of our European Call option on Coca-Cola shares, via Esscher transform, is:

$$c = 2.91323$$

Another method of pricing with a VG process is via Fast Fourier Transform:

$$c = 2.929595$$

In conclusion I personally think that this process did a quite good job in estimating the real price because it is smaller than the American we estimated before. Another confirmation is that I took the data of this option from Yahoo, and at the time this American option traded at 3.1, so it make absolutely sense that the European has a lower value. Because it is a change of about 5% which is acceptable since the expiry date is at 8 days, so not so long.

6.2 Meixner process

In this case the size of the jump is determined by a Meixner distribution. Its characteristic function is:

$$\phi_M(x; a, b, d, m) = E\left\{e^{itX}\right\} = \left(\frac{\cos(b/2)}{\cosh\frac{au-ib}{2}}\right)^{2d} \exp(imu) \tag{39}$$

with *cosh* which is the hyperbolic cosine:

$$\cosh = \frac{e^x + e^{-x}}{2} = \frac{e^{2x}+1}{2e^x} = \frac{1+e^{-2x}}{2e^{-x}} \tag{40}$$

This process is infinitely divisible, so is proved that it is a Lévy process:

$$\phi_M(x; a, b, d, m) = \phi_M\left(x; a, b, \frac{d}{n}, \frac{m}{n}\right)^n \tag{41}$$

The Meixner process has the following properties:
1) $M_0 = 0$;
2) Independent and stationary increments;
3) Distribution of M_t given by the Meixner distribution.

6.2.1 Fitting

Firstly we have to estimate the parameters of our process. We do it using the Method of Moments. The idea is to match the moments of the population with the empirical moments in order to get estimators of the parameters:

$$\begin{cases} m + ad\tan\left(\frac{b}{2}\right) = x = E(X) \\ \frac{a^2d}{2}\left(\cos^{-2}(b/2)\right) = y = \mathrm{Var}(X) \\ \sin(b/2)\sqrt{2/d} = z = \gamma_1(X) \\ 3 + \frac{3-2\cos^2(b/2)}{d} = w = \gamma_2(X) \end{cases} \tag{42}$$

After a bit of algebra and rearranging we can compute our parameters:

$$\begin{aligned}
d &= \frac{1}{w - z^2 - 3} \\
a &= \sqrt{y\left(2w - 3z^2 - 6\right)} \\
b &= 2arctan\sqrt{\frac{z^2}{2w - 3z^2 - 6}} \\
m &= x - ad \quad \tan\left(\frac{b}{2}\right) = x - \frac{z\sqrt{y}}{w - z^2 - 3}
\end{aligned} \tag{43}$$

Statistics	
$E(X)$	0.0000617
$Var(X)$	0.01754834
$\gamma_1(X)(Skewness)$	-1.056317
$\gamma_2(X)(Kurtosis)$	7.607297

Please note that these values coincide with the ones computed in the introduction of Coca-Cola returns, because they are the same (pag. 6).

Parameters	
m	0.04013924
a	0.3208726
b	-0.8224583
d	0.2864105

Let's analyize the Q-Q plot to check how well this Meixner process fits our data: We see that is more linear than the Normal one (pag.6), so it suits our data better.

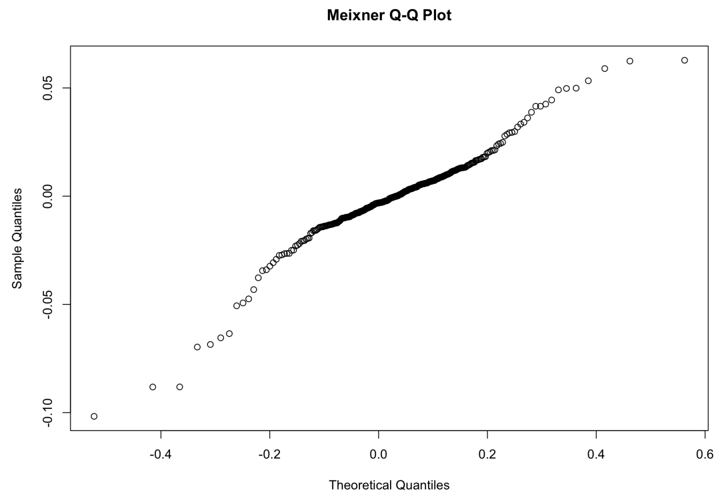

Figure 12: QQ plot for a Meixner process

Applying an Esscher transform we have these simulations:

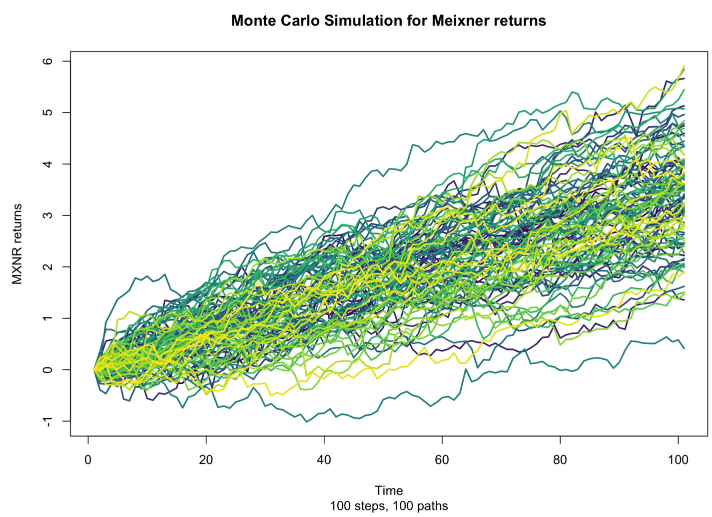

Figure 13: Paths simulation for Meixner process with Esscher transform

6.2.2 Option pricing

Pricing our option with a Meixner process and Esscher transform doesn't perform very well, in fact it overestimates the price of our call too largely, making the result impossible to accept:

$$c_{Esscher} = 2097.285$$

Pricing the option with an mean correcting transform performs poorly as well:

$$c_{Meancorrecting} = 19.07413$$

This process results are too bad to be considered reliable.

7 Multi Asset Options

Multi asset options are derivatives which depend on more than one underlying asset. For instance considering call options:
a) Basket of options: which payoff is given by the weighted sum of the value of its underlying assets minus a strike price, namely:

$$\begin{gathered} c = \max\left[w_1 S_1(T) + w_2 S_2(T) + \ldots + w_k S_k(T) - K, 0\right] \\ \text{with } \sum_{i=1}^{k} w_i = 1 \end{gathered} \tag{44}$$

b)Best-of-options: in this case the payoff is determined by the maximum of the underlying assets minus the strike price:

$$c = \max\left[\max\left(S_1(T), S_2(T), \ldots, S_k(T)\right) - K, 0\right] \tag{45}$$

c)Worst-of-options: it's the vice versa of the previous category, we consider the minimum this time:

$$c = \max\left[\min\left(S_1(T), S_2(T), \ldots, S_k(T)\right) - K, 0\right] \tag{46}$$

d)Options to switch: here the payoff is given by the difference of the values of the underlying assets:

$$c = \max\left[S_2(T) - S_1(T), 0\right] \tag{47}$$

We have just seen that the payoff is determined by a combination of stochastic processes. One huge issue with this kind of contract is that the underlying assets are usually correlated in some way, so we have to "measure" this dependence.
Keep in mind that even if we assume independence between this distributions this problem can have problems to lead to a closed solution. In fact, because of the convolution rule, it isn't sure that two independent r.v. combined return a density function in closed form. Consider two Laplace distribution, their combination doesn't necessarily lead to a Laplace distribution. In this part

we will study the correlation of Coca-Cola stock with the American Express (AXP) one. Why AXP? Firstly it is traded on the same index of KO (NYSE), in addition is a classic "Warren Buffett stock" like KO and considered quite defensive as well, with an almost 2% dividend yield. I wanted to stress their correlation especially during this recent downturn and see how much correlated they are.

7.1 GBMs correlated through a linear correlation coefficient RHO

A very common way to measure correlation is via the Pearson linear coefficient:

$$\rho = \frac{\mathrm{cov}(X, Y)}{\sigma_X \sigma_Y} \tag{48}$$

It is widely used because of its easy usage and interpretation. In our case, looking at the correlation matrix we have a coefficient of 0.4412159. They are positively correlated but not heavily.

Here is displayed a simulation of their GBMs according to their correlation: By inspecting this chart we see that the KO is way more stable than AXP, even

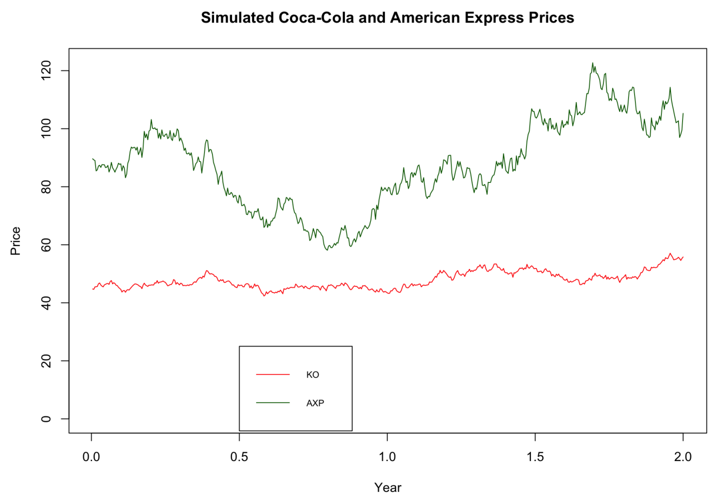

Figure 14: Share price simulation of KO and AXP

taking into account the fact that it is traded on a smaller price than the other. This can be imputed to many factors such as: the position that the companies occupy in their respective industry, another justification can be that the beta of AXP is 2 times the one of KO, so the stock can be considered way more volatile. One last consideration can be the fact that on the long run they both increased, in this sense their positive correlation can be justifiable.

7.2 Copulae

One big issue with the linear coefficient is that it remains constant over the distribution of the rando variable. But the correlation can vary according to the position on the distribution of each r.v., for example in the tails (big losses or big gains) the correlation can be higher than in the center.

A possible solution to overcome to this problem can be the implementation of the concept of Copula. Copulae are mathematical functions that calculate the connections between two variables, in other words, how they "copulate." When X happens (such as a company defaulting), there's a Y chance that Z happens (another company defaults). More specifically, copulae, using the cumulative distribution of the marginals (the r.v. that we want to analyze, KO and AXP), construct a joint cumulative distribution. This is the crucial part, because it takes into account that the dependence could vary on the position of the distribution.
In our case, where we have two marginals, the joint cumulative distribution is:

$$F(x_1, x_2) = c\left(F_{x_1}(x_1), F_{x_2}(x_2)\right) \tag{49}$$

the conditional distribution conditioning on x_2 is:

$$F(x_2 \mid x_1) = \frac{\partial c\left(F_{x_1}(x_1), F_{x_2}(x_2)\right)}{\partial F_{x_1}(x_1)} \tag{50}$$

with density function:

$$f(x_2 \mid x_1) = \frac{\partial^2 c\left(F_{x_1}(x_1), F_{x_2}(x_2)\right)}{\partial F_{x_1}(x_1), \partial F_{x_2}(x_2)} \tag{51}$$

In this part we will focus on a family of copulae: the Archimedean that have a parameter (θ) which manages the dependence between the marginals:

$$c\left(F_{x_1}(x_1), F_{x_2}(x_2)\right) = \phi^{[-1]}\left(\phi\left(F_{x_1}(x_1)\right), \phi\left(F_{x_2}(x_2)\right)\right) \tag{52}$$

The Archimedean copulae we use are:
-Clayton, it stress more the dependence on extreme losses
-Gumbel, it stress more the dependence on extreme gains
-Frank, it stress more the dependence on the center, because it stress quit good both tails but not as strong as the other two copulae
They are differentiated by the value of θ which is a function of Kendall's τ, a measure of rank correlation.
In our case:

$$\varnothing = 0.3577591$$

I fitted these three kinds of copula to our data and compared the likelihood. In this case the best copula is the Frank since its alpha-estimate is the largest and the s.d. the lowest.

The next step is to compare the different copulae on two simulated GBM derived from out τ and θ.
Then on the values we will get, we compute the VAR and ES at 99% on the losses, because we will assume we are long on these two stocks and we are interested in computing the worst case scenario.
The procedure is:
1) Estimate θ and τ starting from KO and AXP time series;

2) Simulate the first GBM ($F_{x_1}(x_1)$) independently;
3) Simulate the second GBM starting from the first just simulated, in order to take into account the dependence between the two. We do this via the inverse transform or via the accept-reject method which takes much more time. In fact there is a snippet in the code when we compare the time consuming of these three copulae. The one which is the quickest is the Clayton because it can perform an inverse transform; while the other two use the acceptance-rejection method
4) Compute the risk measure for each copula.

Copula	VAR	ES
Gumbel	0.07528373	1.01881352
Clayton	0.03009343	1.01927844
Frank	0.08084731	1.01865034

These are quite strange results, in fact we have a bigger ES with is impossible, because if we consider the left tail, as in this case, the ES is always smaller than the VAR at the same confidence level. The VAR can makes sense because we see that the highest decrease possible is in the Clayton (a potential decrease of 97%), which stress the left tail more than the other two, so in this way it could make sense. The Gumbel in fact stress more the right tail, while the Frank stress more the center, and they have similar value of VAR for this reason.
I don't comment the ES because is a really strange value in all three copulae, because in this case if bigger than 1 means that it is an increment, which in our case makes absolutely no sense.
The code should include even a Student's t copula, but I wasn't able to find the fitting of it inside the code. That's the reason why you can't see the comparison with it.
I will use the Student's t copula in the chapter regarding my contributions.

I wouldn't rely on the result of this part for many reasons. Firstly because the marginals are simulated with uniforms r.v., not considering their true distribution. Secondly because they rely on the concept of GBM associated with normal distribution that account for increments, so we don't have negative value (losses) if you look inside every matrix generated.

8 Multi Asset Options under Lévy Processes

In the previous section the assumed that the underlyign assets follow a GMB. Let's now suppose that they follow a Lévy process instead.
We will see two methods to measure the dependence of the underlying Lèvy processes:
1) Ballotta Bonfiglioli Model;
2) Lévy Copula.
I skipped the marginal Lévy processes correlated through a linear correlation coefficient rho because of the limits it presents.

8.0.1 Ballotta Bonfiglioli Model

It can be considered as equal as the Multidimensional GBM. But here the two marginals are:

$$\begin{aligned} X_1(t) &= Y_1(t) + b_1 Z(t) \\ X_2(t) &= Y_2(t) + b_2 Z(t) \end{aligned} \tag{53}$$

With X_1 and X_2 the two idiosyncratic parts, $Z(t)$ is the factor which control the correlation between the marginals Lévy processes and b_1, b_2 are two constants. Note that all the components are independent Lévy processes.
Two stock prices follow:

$$\begin{aligned} S_1(t) = S_1(0) \exp\left(Y_1(t) + b_1 Z(t)\right) \\ S_2(t) = S_2(0) \exp\left(Y_2(t) + b_2 Z(t)\right) \end{aligned} \tag{54}$$

One key fact is that this approach allows a big flexibility. In fact they can be inclused as many Lévy processes as anyone wants in order to take into account all the dependencies. Suppose we have three marginals:

$$\begin{aligned} S_1(t) = S_1(0) \exp\left(Y_1(t) + b_1 Z_4(t) + b_4 Z_6(t) + b_7 Z_7(t)\right) \\ S_2(t) = S_2(0) \exp\left(Y_2(t) + b_2 Z_4(t) + b_5 Z_5(t) + b_8 Z_7(t)\right) \\ S_3(t) = S_3(0) \exp\left(Y_3(t) + b_3 Z_5(t) + b_6 Z_6(t) + b_9 Z_7(t)\right) \end{aligned} \tag{55}$$

Where the Ys are the idiosyncratic components. Z_4 which represents the dependence between the first two marginals; Z_5 represents the dependence between the second and the third marginals; Z_6 dependence between the first and the third marginals. In the end we have Z_7 which takes into account the dependence between all thre marginals.

Here we see the simulations for two marginals:

You can clearly see the jumps which characterizes the Lévy processes. These simulations are run on the same parameters present in the code of lectures.

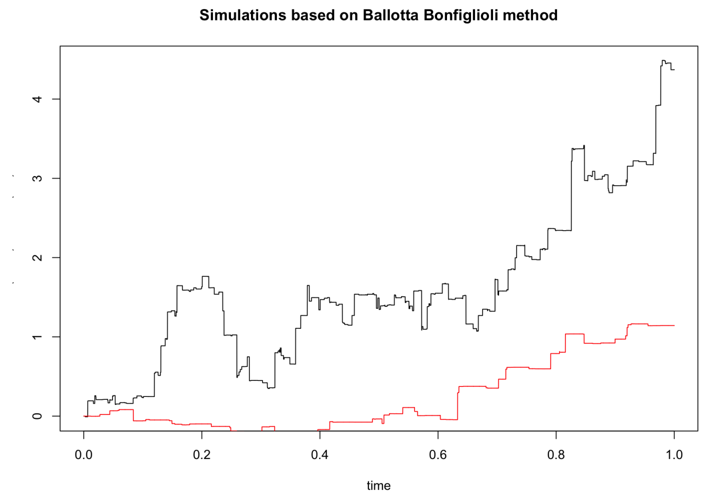

Figure 15: Two marginals simulation based on BB method

8.1 Lévy Copulae

They do exactly what normal copulae do, but they describe dependence on Lévy processes. They use tail integral instead of distribution functions:

$$U(x) = \begin{cases} \nu([x,\infty)), & \text{for } x \in (0,\infty) \\ -\nu((-\infty,0]), & \text{for } x \in (-\infty,0) \\ 0, & \text{for } x = \infty, -\infty \end{cases} \tag{56}$$

Now we have to pose our attention on the Lévy measure, because it capture the jump component, which is crucial in the Lévy processes. We study especially the dependence between jumps.
The θ parameter is responsible for the dependence in the absolute values of the jumps of the marginal Lévy processes. It goes from 0 (independence) to ∞ (total dependence). The η determines the dependence of the sign of the jumps and has an existing domain between 0 and 1: when $\eta = 1$, the two components always jump in the same direction, and when $\eta = 0$, positive jumps in one component are accompanied by negative jumps in the other and viceversa.
These two parameters are difficult to estimate because we would firstly transform the marginals time series in marginal jump series and then estimate the parameters.

Now we will simulate two Variance Gamma process via a Clayton Lévy copula, the procedure is the following:
1)Estimate the parameters of the marginals Variance Gamma processes;
2)Estimate the dependence between the parameters;
3)Simulate the first marginal independently;

4)Simulate the second marginal conditionally on the first one.

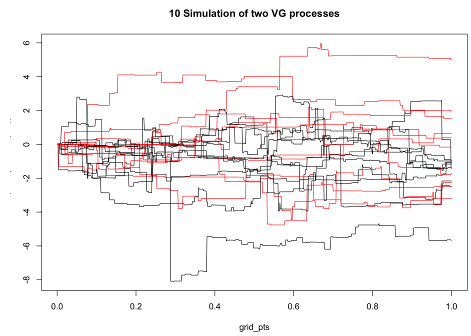

Figure 16: Two Variance Gamma marginals simulation based on Clayton Lévy copula

9 My contributions

I already added some contribution from myself, for more details check the paragraph 2.3 Risk Measure, where I compute and explain the Beta, VAR and Sharpe Ratio of KO. Then via a user-defined function I displayed some interesting statistics of our data, which can be found at page 6. And If we want to consider a contribution of mine I plotted a candlestick chart at page 4, which is the simplest command in quantmod for plotting time series, it can be made nicer and interactive via plotly package.

The core of my contribution is based on the usage of best copula on our data and compare the results of different copulae (Normal and Independent). I thought at the multi asset options and how the concept of copula could help. I suppose it is useful to compute the probability of how many times, on simulations based on a specific copula, when an asset is at its VAR at 99% the other asset exceed that value. I think this could be useful in the Best-of-options and in the case of Worst-of-options scenarios.

9.1 Best copula

The correlation structure could be firstly analysed by looking at the ranks scatter plot in figure 15.

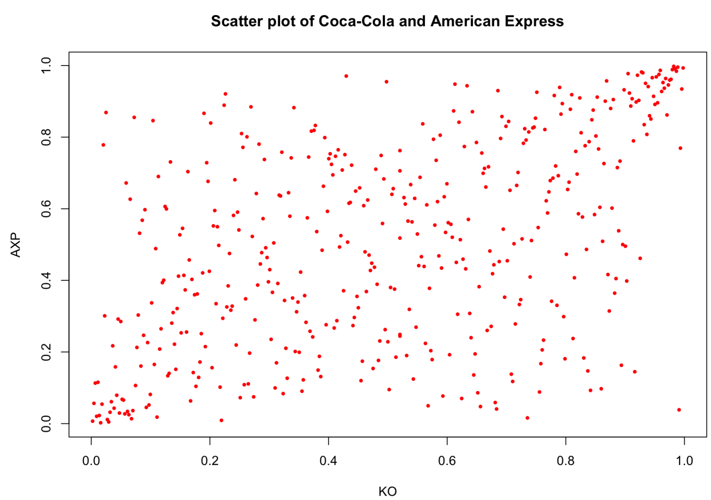

Figure 17: Scatter plot of KO and AXP

It shows that there is a sort of correlation, especially in the two tails.

Then I fitted different distribution with MLE and look for the one that have the smallest AIC (Aikaike Information Criterion, a score for model selection). The result is that for KO the best distribution is the Student's t, and Johnson's

SU-distribution for AXP, but the difference in the AIC with Students't was so light that I decide to keep the t for simplicity.

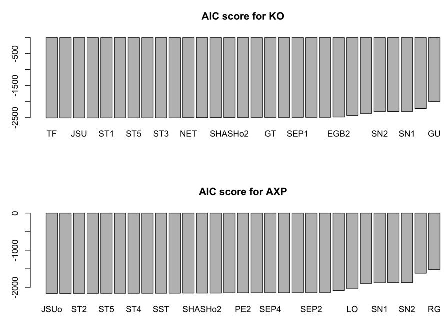

Figure 18: AIC score for the various distribution fitted on our data

Let's quickly check how well Students't estimates our data:

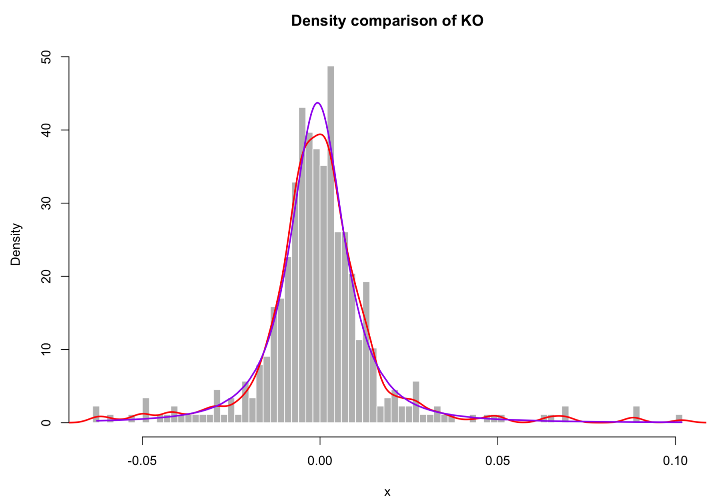

Figure 19: In red the real density of KO, in purple the Student's t with the estimated parameters

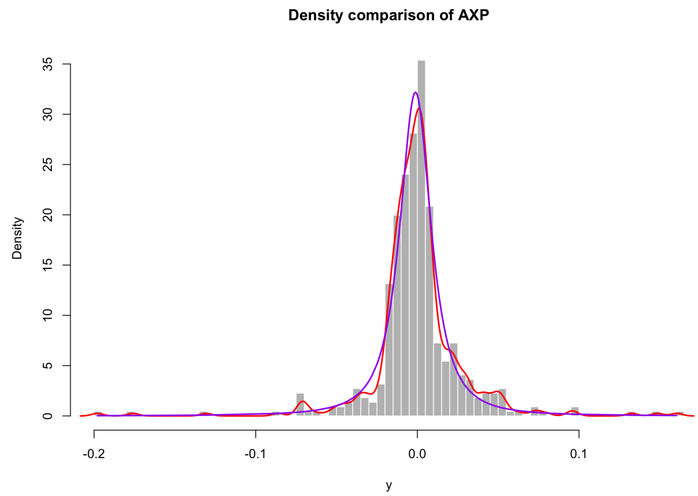

Figure 20: In red the real density of AXP, in purple the Student's t with the estimated parameters

Given these marginals I fit different copulae and check which is the best one. The result is the Students't.
Sincerely in the function that checks which copula is the best I restricted the families of copulae to compare.

Now my goal is to simulate 1000 scenarios with the Student's t copula, the Normal copula and the Independent one. Compute the probability when one asset (Coca-Cola) is at its VAR the other (American Express) exceeds that value. Then compare the probability and see how it changes. Namely compute the probability of $P(X >= k|Y = k)$ where k is the VAR at 99% using the Students't copula and compare it with the independent case and the Gaussian (with Spearman ρ). This should indicate how well the different copulae behave on our data.

	Probability
Students't	0.434
Gaussian	0.185
Independent	0.02

These results make sense, because the independent means no correlation at all between the assets, in fact presents the smallest probability. Gaussian have some correlation but mostly in the center, and take into account that in this case I fitted it with linear Spearman's ρ but the probability increased a lot.
In the end we have the highest probability, actually very high (0.4%) with Students't. This means that if KO suffers a huge loss there's a 0.4% of chance that AXP suffer an equal or higher loss as well. It makes sense because the two

are companies from the same country, listed on the same index, and AXP, as we have seen previously, is much more volatile than KO.

9.2 Relative Strength Index

RSI is an oscilatting indicator, it takes values between to 0 to 100. Its goal is to use a short lookback to predict when the price has temporarely decreased and it is predicted to increase in the next several days. Essentially RSI generates a signal of when it may be a good time to enter in short term position.

$$RSI = 100 * U/(U + D) \tag{57}$$

Where U is the average of differences in increasing closing of three days in our case. D is the average in absolute price of differences in decreasing closing of three days.
I computed it via the RSI function in TTR package. Here is the result: When

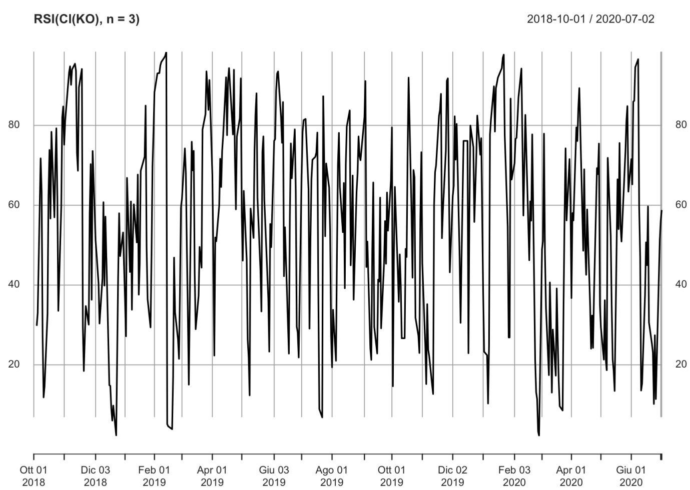

Figure 21: Three days RSI on KO

it is under 30 it can happen an increase in the following days, so if an investor want to take a long position for short term on KO it can be a good time to enter. On the contrary, when it is over 70 it can happen a downturn in the following days, so take a short position could be a good idea.

Please refer to the following chunks of code: (90, 101-112, 1318-1480)

10 Conclusion

Personally in pricing short term options on KO I would choose a Lévy process, in particular the Variance Gamma, because given the data of Coca-Cola share price it is the one that priced it more accurately.
I understand that GBM are easier to model, especially for big institutions where timing is really important. But it is important to take into account that models based esclusively on GBM could lead to devastating situation. Think about the Black Monday on the 19th October 1987: many models of statistical arbitrage and option pricing of first quantitative hedge found were based on GBM, and they could be the reason of that incredible downturn, because they consider large stock market fluctuations extremely rare. For more information on this topic please refer to the book "The Black Swan" of Nassim Taleb.
Further improvement for this project can be the usage of Machine Learning methodologies to price options. It couldn't be possible for me because I didn't find a complete dataset for training the methods I wanted to apply.

References

[1] URL: `https://stats.stackexchange.com/questions/132652/how-to-determine-which-distribution-fits-my-data-best`.

[2] Bradley Efron and Trevor Hastie. *Computer Age Statistical Inference*. Cambridge, 2016.

[3] Marius Hofert et al. *Elements of Copula Modelling with R*. Springer, 2017.

[4] Marius Hofert et al. *Quantitative Risk Management Tools*. URL: `https://qrmtutorial.org/`.

[5] John Hull. *Options, Futures and other derivatives*. Pearson, 1989.

[6] Stafano M. Iacus. *Option Pricing and Estimation of Financial Models with R*. Wiley, 2011.

[7] Stafano M. Iacus. *Simulation and Inference for Stochastic Differential Equations: with R Examples*. Springer, 2008.

[8] Peter Kempthorne et al. *Topics in Mathematics with Applications in Finance*. URL: `https://ocw.mit.edu/courses/mathematics/18-s096-topics-in-mathematics-with-applications-in-finance-fall-2013/index.htm`.

[9] Ubbo F. Wiersema. *Brownian Motion Calculus*. Wiley, 2008.

```
library(quantmod)
library(tseries)
library(sde)
library(fOptions)
library(stats4)
library(ggplot2)
library(VarianceGamma)
library(viridis)
library(moments)
library(PerformanceAnalytics)
library(fBasics)
library(Runuran)
library(zoo)
library(Quandl)
library(gsl)
library(copula)
library(qrng)
library(fitdistrplus)
library(gamlss)
library(gamlss.dist)
library(gamlss.add)
library(VineCopula)
library(TTR)
set.seed(123)

###Stock Introduction###

#User defined funtion made by me to have an overview of our return
report <- function(vec){

  count = length(vec)
  #central tendency
  mean = mean(vec)
  median = median(vec)

  #variability
  variance = var(vec)
  stdev = sd(vec)
  stdev_over_mean = stdev / mean
  skewness = e1071::skewness(vec)
  kurtosis = e1071::kurtosis(vec)

  #rank statistics
  min = min(vec)
  max = max(vec)
  range = max - min

  table = round(t(data.frame(count,
                             mean,
                             median,
                             variance,
                             stdev,
                             stdev_over_mean,
                             skewness,
                             kurtosis,
```

```
                              min,
                              max,
                              range)), 7)

  return(table)
}

getSymbols("KO", from="2010-01-01", to="2020-07-05")
lineChart(KO$KO.Adjusted,name="Coca-Cola",theme="black")

COKE <- get.hist.quote("KO", start = "2010-01-01", end = "2020-07-05")
chartSeries(COKE, TA=c(addVo(), addBBands()), theme="black")
COKE <- COKE$Close
cpoint(COKE)
chartSeries(KO$KO.Adjusted, TA=c(addVo(), addBBands()), theme="black")
addVLine = function(dtlist) plot(addTA(xts(rep(TRUE,NROW(dtlist)),dtlist),on=1, col="red"))

#User defined function to plot the line where the cpoint begins
addVLine(cpoint(COKE)$tau0)


#all log returns

COKE <- as.numeric(COKE)
n <- length(COKE)
X <- log(COKE[-1]/COKE[-n])
plot(X, type = "l", main = "Coca-Cola␣stock␣log␣returns")
abline(v = cpoint(X)$tau0, col = "red")


#Now let's␣focus␣from␣the␣changing␣point

getSymbols("KO",␣from="2018-10-019",␣to="2020-07-05")
lineChart(KO$KO.Adjusted,name="Coca-Cola",theme="black")
chartSeries(KO)␣#candlestick␣chart,␣can␣be␣optimized␣via␣plotly␣package
Coca␣<-␣KO$KO.Adjusted
X␣<-␣diff(log(Coca))
plot(X)
head(X)
sum(is.na(X))␣#There␣should␣be␣only␣the␣first␣observation␣as␣NA
X␣<-␣na.omit(X)
head(X)
report(X)


#Beta␣of␣Cocacola␣during␣the␣period␣considered,␣benchmark␣DJI

getSymbols("^DJI",␣from="2018-01-01",␣to="2020-07-05")
Dow␣=␣DJI$DJI.Adjusted
D␣=␣na.omit(diff(log(Dow)))
CAPM.beta(X,␣D)

#VAR␣99%
VaR(R␣=␣X$KO.Adjusted,␣p␣=␣0.99,␣method␣=␣"historical")

#Sharpe␣Ratio
SharpeRatio(X$KO.Adjusted,␣Rf␣=␣0.007,␣p␣=␣0.99)
```

```
#Cumulative returns
#I didn't put this part in the project
{
  X$KO.Adjusted[1,1] = 0 #-0.006198556
  head(X)
  cum1 = sum(X$KO.Adjusted)
  cum = exp(cum1) - 1purple!40!black;purple!40!black purple!40!blackcum

  XX = Delt(Coca)
  head(XX)
  XX[1,1] = 0
  XX$gross = 1+ XX$Delt.1.arithmetic
  XX$cump = cumprod(XX$gross)
  y.range = range(XX$cump)
  plot(XX$cump, type= "l", auto.grid = FALSE, xlab = "Date", ylab = "Value of investment ($
      ylim = y.range, main = "Coca-Cola performance based on total returns")
  ggplot(XX, aes(x = index(XX), y = XX$cump)) + geom_line(color = "red") + ggtitle("Coca-Co

  X$KO.Adjusted[1,1] = -0.006198556
}

#Let's plot the density function and the qqplot in order to check if it looks
#like a Gaussian distribution

#density plot:
plot(density(X), lwd=2, main="Coca-Cola stock log returns density plot")
f <- function(u) dnorm(u, mean=mean(X), sd=sd(X))
curve( f, -0.1, 0.1, add=TRUE, col="red",lwd=2)

#qq plot:
qqnorm(X, main="Coca-Cola stock Q-Q plot")
qqline(X, col="red",lwd=2)


####PARAMETER ESTIMATION####

#Historical volatility
Delta<- 1/252
alpha.hat <- mean(X,na.rm=TRUE)/Delta
sigma.hat <- sqrt(var(X,na.rm=TRUE)/Delta)
mu.hat<- alpha.hat + 0.5*sigma.hat^2
mu.hat
sigma.hat

#Implied volatility
dim(X)
S0 <- as.numeric(Coca[441])
K <- 35 #deep in the money
T <- 14 * Delta
r <- 0.07
p <- 9.4 #KO200710C00035000      (yahoo)
sigma.imp1 <- GBSVolatility(p, "c", S = S0, X = K, Time = T, r = r,  b = r)
sigma.imp1 #result too much low

K <- 44.5 #moderately in the money
T <- 8 * Delta
```

```
r <- 0.07
p <- 0.76 #KO200710C00044500 (yahoo)
sigma.imp2 <- GBSVolatility(p, "c", S = S0, X = K, Time = T, r = r,  b = r)
sigma.imp2 #"more acceptable" result

##THIS WILL BE THE OPTION I WILL USE IN THE ENTIRE PROJECT
K <- 42 #moderately in the money
T <- 8 * Delta
r <- 0.07
p <- 3.1 #KO200710C00042000 (yahoo)
sigma.imp3 <- GBSVolatility(p, "c", S = S0, X = K, Time = T, r = r,  b = r)
sigma.imp3

#Other attempts of pricing these options, you can skip this part, used only to study the di
K <- 49 #out of the money
T <- 8 * Delta
r <- 0.07
p <- 0.02 #KO200710C00042000 (yahoo)
sigma.impout <- GBSVolatility(p, "c", S = S0, X = K, Time = T, r = r,  b = r)
sigma.impout

K <- 42.5
T <- 167 * Delta #let's try to expand the time horizon
r <- 0.07
p <- 4.5 #KO201218C00042500 (yahoo)
sigma.imp4 <- GBSVolatility(p, "c", S = S0, X = K, Time = T, r = r,
b = r)
sigma.imp4

K <- 20 #deep in the money
T <- 183 * Delta #let's try to expand the time horizon
r <- 0.07
p <- 26.8 #KO201218C00020000 (yahoo)
sigma.imp5 <- GBSVolatility(p, "c", S = S0, X = K, Time = T, r = r,  b = r)
sigma.imp5


#MLE estimation
set.seed(123)
log.lik <- function(mu = mu.hat, sigma = sigma.hat) -sum(dnorm(X, mean = mu,
                                                                    sd = sigma, log = TRUE))
fit <- mle(log.lik, lower = c(0.01,0.01), method = "L-BFGS-B") #"L-BFGS-B"
fit
fit2 <- mle(log.lik,  method = "Brent")  #Brent, gives error
fit2
fit3 <- mle(log.lik)
fit3
fit4 <- mle(log.lik,  method = "BFGS") #same as fit4, in fact it is the default algorithm f
fit4
fit5 <- mle(log.lik,  method = "CG"); fit5
set.seed(123)
fit6 = mle (log.lik, method = "SANN"); fit6
#MLE didn't reach convergence
#Furthermore for the other algo it returns some warnings

### European Option pricing ###
#With Black & Scholes formula
```

```
S0 <- as.numeric(Coca[441])
sigma.hat <- as.numeric(sigma.hat)
K <- 42 #moderately in the money
T <- 8 * Delta
r <- 0.07
p <- 3.1 #KO200710C00042000 (yahoo)
p0 <- GBSOption("c", S = S0, X = K, Time = T, r = r, b = r, sigma = sigma.hat)@price
p0

#with MC
MCPrice <- function(x = 1, t = 0, T = 1, r = 1, sigma = 1,
                    M = 1000, f) {
  h <- function(m) {
    u <- rnorm(m/2)
    tmp <- c(x * exp((r - 0.5 * sigma^2) * (T - t) + sigma *
                       sqrt(T - t) * u), x * exp((r - 0.5 * sigma^2) * (T -
                                                                         t) + sigma * sqrt
    mean(sapply(tmp, function(xx) f(xx)))
  }
  p <- h(M)
  p * exp(-r * (T - t))
}

f <- function(x) max(0, x - K)

set.seed(123)
M <- 1000
MCPrice(x = S0, t = 0, T = T, r = r, sigma.hat, M = M, f = f)
M <- 50000
MCPrice(x = S0, t = 0, T = T, r = r, sigma.hat, M = M, f = f)
M <- 1e+06
MCPrice(x = S0, t = 0, T = T, r = r, sigma.hat, M = M, f = f)

#Speed of convergence
set.seed(123)
m <- c(10, 50, 100, 150, 200, 250, 500, 1000)
p1 <- NULL
err <- NULL
nM <- length(m)
repl <- 100
mat <- matrix(, repl, nM)
for (k in 1:nM) {
  tmp <- numeric(repl)
  for (i in 1:repl) tmp[i] <- MCPrice(x = S0, t = 0, T = T,
                                      r = r, sigma.hat, M = m[k], f = f)
  mat[, k] <- tmp
  p1 <- c(p1, mean(tmp))
  err <- c(err, sd(tmp))
}
colnames(mat) <- m

p0 <- GBSOption(TypeFlag = "c", S = S0, X = K, Time = T, r = r, b = r, sigma = sigma.hat)@pr
minP <- min(p1 - err)
maxP <- max(p1 + err)
plot(m, p1, type = "n", ylim = c(minP, maxP), axes = F, ylab = "MC␣price",
xlab = "MC␣replications")
lines(m, p1 + err, col = "blue")
```

```
lines(m, p1 - err, col = "blue")
axis(2, p0, "B&S␣price")
axis(1, m)
boxplot(mat, add = TRUE, at = m, boxwex = 15, col = "orange",  axes = F)
points(m, p1, col = "blue", lwd = 3, lty = 3)
abline(h = p0, lty = 2, col = "red", lwd = 3)

#FFT
FFTcall.price <- function(phi, S0, K, r, T, alpha = 1, N = 2^12, eta = 0.25) {
  m <- r - log(phi(-(0+1i)))
  phi.tilde <- function(u) (phi(u) * exp((0+1i) * u * m))^T
  psi <- function(v) exp(-r * T) * phi.tilde((v - (alpha +
                                                       1) * (0+1i)))/(alpha^2 + alpha - v^2 +

  lambda <- (2 * pi)/(N * eta)
  b <- 1/2 * N * lambda
  ku <- -b + lambda * (0:(N - 1))
  v <- eta * (0:(N - 1))
  tmp <- exp((0+1i) * b * v) * psi(v) * eta * (3 + (-1)^(1:N) -
                                                   ((1:N) - 1 == 0))/3
  ft <- fft(tmp)
  res <- exp(-alpha * ku) * ft/pi
  inter <- spline(ku, Re(res), xout = log(K/S0))
  return(inter$y * S0)
}
mu=1
sigma <- 0.25
phiBS <- function(u) exp((0+1i) * u * (mu - 0.5 * sigma^2) - 0.5 * sigma^2 * u^2)
FFTcall.price(phiBS, S0 = S0, K = K, r = r, T = T)

#put-call parity
call.price <- function(x = 1, t = 0, T = 1, r = 1, sigma = 1,
                       K = 1) {
  d2 <- (log(x/K) + (r - 0.5 * sigma^2) * (T - t))/(sigma *
                                                        sqrt(T - t))
  d1 <- d2 + sigma * sqrt(T - t)
  x * pnorm(d1) - K * exp(-r * (T - t)) * pnorm(d2)
}
put.price <- function(x = 1, t = 0, T = 1, r = 1, sigma = 1,
                      K = 1) {
  d2 <- (log(x/K) + (r - 0.5 * sigma^2) * (T - t))/(sigma *
                                                        sqrt(T - t))
  d1 <- d2 + sigma * sqrt(T - t)
  K * exp(-r * (T - t)) * pnorm(-d2) - x * pnorm(-d1)
}
C <- call.price(x = S0, t = 0, T = T, r = r, K = K, sigma = sigma.hat)
P <- put.price(x = S0, t = 0, T = T, r = r, K = K, sigma = sigma.hat)
P
C = as.numeric(C)
C - S0 + K * exp(-r * T)

#Greeks
GBSCharacteristics(TypeFlag = "c", S = S0, X = K, Time = T,
                   r = r, b = r, sigma = sigma.hat)


###AMerican Option pricing ###
```

```
# Broadie and Glasserman Monte Carlo method

simTree <- function(b,d, S0, sigma, T, r){
  tot <- sum(b^(1:(d-1)))
  S <- numeric(tot+1)
  S[1] <- S0
  dt <- T/d
  for(i in 0:(tot - b^(d-1))){
    for(j in 1:b){
      S[i*b+j +1] <- S[i+1]*exp((r - 0.5*sigma^2)*dt + sigma*sqrt(dt)*rnorm(1))
    }
  }
  S
}



upperBG <- function(S, b, d, f){
  tot <- sum(b^(1:(d-1)))
  start <- tot - b^(d-1) +1
  end <- tot +1
  P <- S
  P[start:end] <- f(S[start:end])
  tot1 <- sum(b^(1:(d-2)))
  for(i in tot1:0){
    m <- mean(P[i*b+1:b+1])
    v <- f(S[i+1])
    P[i+1] <- max(v,m)
  }
  P
}

lowerBG <- function(S, b, d, f){
  tot <- sum(b^(1:(d-1)))
  start <- tot - b^(d-1) +1
  end <- tot +1
  p <- S
  p[start:end] <- f(S[start:end])
  tot1 <- sum(b^(1:(d-2)))

  m <- numeric(b)
  for(i in tot1:0){
    v <- f(S[i+1])
    for(j in 1:b){
      m[j] <- mean(p[i*b+(1:b)[-j]+1])
      m[j] <- ifelse( v>m[j], v, p[i*b+(1:b)[j]+1])
    }
    p[i+1] <- mean(m)
  }
  p
}


b <- 6
d <- 4
M <- 10000
low <- 0
```

```
upp <- 0
f <- function(x) sapply(x, function(x) max(x-K,0))
sigma = sigma.hat
set.seed(123)
for(i in 1:M){
  S <- simTree(b,d, S0, sigma=sigma.hat , T, r)
  low <- low + lowerBG(S, b,d, f)[1]
  upp <- upp + upperBG(S, b,d, f)[1]
}
low/M
upp/M

#via regression
LSM <- function(n, d, S0, K, sigma, r, T){
  s0 <- S0/K
  dt <- T/d
  z <- rnorm(n)
  s.t <- s0*exp((r-1/2*sigma^2)*T+sigma*z*(T^0.5))
  s.t[(n+1):(2*n)] <- s0*exp((r-1/2*sigma^2)*T-sigma*z*(T^0.5))
  CC <- pmax(1-s.t, 0)
  payoffeu <- exp(-r*T)*(CC[1:n]+CC[(n+1):(2*n)])/2*K
  euprice <- mean(payoffeu)

  for(k in (d-1):1){
    z <- rnorm(n)
    mean <- (log(s0) + k*log(s.t[1:n]))/(k+1)
    vol <- (k*dt/(k+1))^0.5*z
    s.t.1 <- exp(mean+sigma*vol)
    mean <- (log(s0) + k*log( s.t[(n+1):(2*n)] )) / ( k + 1 )
    s.t.1[(n+1):(2*n)] <- exp(mean-sigma*vol)
    CE <- pmax(1-s.t.1,0)
    idx<-(1:(2*n))[CE>0]
    discountedCC<- CC[idx]*exp(-r*dt)
    basis1 <- exp(-s.t.1[idx]/2)
    basis2 <- basis1*(1-s.t.1[idx])
    basis3 <- basis1*(1-2*s.t.1[idx]+(s.t.1[idx]^2)/2)

    p <- lm(discountedCC ~ basis1+basis2+basis3)$coefficients
    estimatedCC <- p[1]+p[2]*basis1+p[3]*basis2+p[4]*basis3
    EF <- rep(0, 2*n)
    EF[idx] <- (CE[idx]>estimatedCC)
    CC <- (EF == 0)*CC*exp(-r*dt)+(EF == 1)*CE
    s.t <- s.t.1
  }

  payoff <- exp(-r*dt)*(CC[1:n]+CC[(n+1):(2*n)])/2
  usprice <- mean(payoff*K)
  error <- 1.96*sd(payoff*K)/sqrt(n)
  earlyex <- usprice-euprice
  data.frame(usprice, error, euprice)
}
sigma = sigma.hat
set.seed(123)
LSM(100000, 5, S0, K, sigma, r, T)

#explict finite difference method
AmericanPutExp <- function(Smin=38, Smax,  T=1, N=10, M=10, K, r=0.07, sigma=0.01){ #note t
```

```r
  Dt = T/N
  DS = (Smax-Smin)/M
  t <- seq(0, T, by =Dt)
  S <- seq(Smin, Smax, by=DS)
  A <- function(j) (-0.5*r*j*Dt + 0.5*sigma^2*j^2*Dt)/(1+r*Dt)
  B <- function(j) (1-sigma^2*j^2*Dt)/(1+r*Dt)
  C <- function(j) (0.5*r*j*Dt + 0.5*sigma^2*j^2*Dt)/(1+r*Dt)
  P <- matrix(, M+1, N+1)
  colnames(P) <- round(t,2)
  rownames(P) <- round(rev(S),2)
  P[M+1, ] <- K   # C(,j=0) = K
  P[1,] <- 0   # C(,j=M) = 0
  P[,0:N+1] <- sapply(rev(S), function(x) max(K-x,0))
  optTime <- matrix(FALSE, M+1, N+1)
  optTime[M+1,] <- TRUE
  optTime[which(P[,N+1]>0),N+1] <- TRUE

  for(i in (N-1):0){
    for(j in 1:(M-1)){
      J <- M+1-j
      I <- i+1
      P[J,I] <- A(j)*P[J+1,I+1] + B(j)*P[J,I+1] + C(j)*P[J-1,I+1]
      if(P[J,I] < P[J,N+1])
        optTime[J,I] <- TRUE
    }
  }
  colnames(optTime) <- colnames(P)
  rownames(optTime) <- rownames(P)
  ans <- list(P=P, t=t, S=S, optTime=optTime,N=N,M=M)
  class(ans) <- "AmericanPut"
  return(invisible(ans))
}

plot.AmericanPut <- function( obj ){
  plot(range(obj$t),range(obj$S),type="n",axes=F,xlab="t", ylab="S")
  axis(1,obj$t,obj$t)
  axis(2,obj$S,obj$S)
  abline(v = obj$t, h = obj$S, col = "darkgray", lty = "dotted")
  for(i in 0:obj$N){
    for(j in 0:obj$M){
      J <- obj$M+1-j
      I <- i+1
      cl <- "green"purple!40!black;
      if(obj$optTime[J,I])
        cl <- "black"
      text(obj$t[i+1],obj$S[j+1], round(obj$P[J,I],2),cex=0.75, col=cl)
    }
  }
  DS <- mean(obj$S[1:2])
  y <- as.numeric(apply(obj$optTime,2, function(x) which(x)[1]))
  lines(obj$t, obj$S[obj$M+2-y]+DS, lty=2)
}
put <- AmericanPutExp(Smax =50, sigma = sigma.hat, K = 42, T=T)
round(put$P,2)
par(mar=c(3,3,1,1))
plot(put)
```

```
myval <- round(put$P[which(rownames(put$P)==S0),1],2)


#Implicit finite difference method

AmericanPutImp <- function( Smin=38, Smax,  T=1, N=10, M=10, K, r=0.07, sigma=0.01){
  Dt = T/N
  DS = (Smax-Smin)/M
  t <- seq(0, T, by =Dt)
  S <- seq(Smin, Smax, by=DS)

  A <- function(j) 0.5*r*j*Dt - 0.5*sigma^2*j^2*Dt
  B <- function(j) 1+sigma^2*j^2*Dt+r*Dt
  C <- function(j) -0.5*r*j*Dt - 0.5*sigma^2*j^2*Dt

  a <- sapply(0:M, A)
  b <- sapply(0:M, B)
  c <- sapply(0:M, C)

  P <- matrix(, M+1, N+1)
  colnames(P) <- round(t,2)
  rownames(P) <- round(rev(S),2)

  P[M+1, ] <- K   # C(,j=0) = K
  P[1,] <- 0   # C(,j=M) = 0
  P[,0:N+1] <- sapply(rev(S), function(x) max(K-x,0))

  AA <- matrix(0, M-1, M-1)
  for(j in 1:(M-1)){
    if(j>1) AA[j,j-1] <- A(j)
    if(j<M) AA[j,j] <- B(j)
    if(j<M-1) AA[j,j+1] <- C(j)
  }

  optTime <- matrix(FALSE, M+1, N+1)
  for(i in (N-1):0){
    I <- i+1
    bb <- P[M:2,I+1]
    bb[1] <- bb[1]-A(1)*P[M+1-0,I+1]
    bb[M-1] <- bb[M-1]-C(M-1)*P[M+1-M,I+1]
    P[M:2,I] <- solve(AA,bb)
    idx <- which(P[,I] < P[,N+1])
    P[idx,I] <- P[idx,N+1]
    optTime[idx, I] <- TRUE
  }
  optTime[M+1,] <- TRUE
  optTime[which(P[,N+1]>0),N+1] <- TRUE
  colnames(optTime) <- colnames(P)
  rownames(optTime) <- rownames(P)
  ans <- list(P=P, t=t, S=S, optTime=optTime,N=N,M=M)
  class(ans) <- "AmericanPut"
  return(invisible(ans))
}



put <- AmericanPutImp(Smax = 50, sigma = sigma.hat, K= 42, T=T)
```

```
round(put$P,2)


par(mar=c(3,3,1,1))
plot(put)


###L vy processes###

#Variance Gamma
vgFit(X) #esitmate VG parameters on the sample
str(vgFit(X))
vg_param <- vgFit(X)$param

c <- as.numeric(vg_param[1])
sigma <- as.numeric(vg_param[2])
theta <- as.numeric(vg_param[3])
nu <- as.numeric(vg_param[4])

N <- 100
nsim <- 1000

#Variance Gamma function
VG=function(sigma, nu, theta, T, N, r) {
  a=1/nu
  b=1/nu
  h=T/N
  t=(0:N)*T/N
  X=rep(0, N+1)
  I=rep(0,N)
  X[1]=0
  for(i in 1:N) {
    I[i]=rgamma(1,a*h,b)
    X[i+1]=X[i] + theta*I[i]+sigma*sqrt(I[i])*rnorm(1)
  }
  return((X)) }


set.seed(123)
VG_paths<-matrix(nrow = nsim, ncol=N+1)
for(i in 1:nsim){
  VG_paths[i,]<-VG(sigma, nu, theta, T, N, r)
}

VG_paths


colori=viridis(nsim)
plot(VG_paths[1,], col=0, type="l", ylim = c(min(VG_paths),max(VG_paths)),
     main = "Monte Carlo Simulation for VG returns", sub = "100 steps, 1000 paths",
     xlab = "Time", ylab = "VG returns")
for(i in 2:nsim){
  lines(VG_paths[i,], col=colori[i], lwd = 2)purple!40!black;
}

l_ret.s <- sort(as.numeric(X))  #sort the log returns

```

```
p <- ppoints(length(l_ret.s)) #plotting position

VG.q <- qvg(p, vgC=c, sigma=sigma, theta=theta, nu=nu) #compute the quantile

plot(VG.q, l_ret.s, main = "Variance-Gamma Q-Q Plot",
     xlab = "Theoretical Quantiles", ylab = "Sample Quantiles")

par(mfrow = c(2,1))

plot(density(X[-1,]), type = "l", lwd = 2, lty = 3, col = "coral2",
     xlim= c(-0.03,0.03), ylim=c(0,120), main ="", xlab ="", ylab = "")
legend ("topright", inset = .02, c("Kernel", "VG"),
        col=c("coral2","seagreen3"), lwd=c(2,1), lty=c(3,1), cex = 0.8, bty = "n")
points(seq(min(X[-1,]), max(X[-1,]), length.out=500),
       dvg(seq(min(X[-1,]), max(X[-1,]), length.out=500),
           mean(X[-1,]), sd(X[-1,])), type="l", col="seagreen3")

#Log-density comparison
gridplot <- seq(min(X[-1,]), max(X[-1,]), length.out=length(X[-1,]))
plot(density(X[-1,])$x, log(density(X[-1,])$y),
     type = "l", lwd = 2, lty = 3, col = "coral2",
     xlim= c(-0.025,0.025), ylim=c(-10,7.5),
     main ="", xlab ="", ylab = "")
legend ("topright", inset = .02, c("Kernel", "Normal"),
        col=c("coral2","seagreen3"), lwd=c(2,1), lty=c(3,1), cex = 0.8, bty = "n")
points(gridplot, log(dvg(gridplot, param = c(c, sigma, theta, nu))),
       type="l", col="seagreen3")

par(mfrow=c(1,1))

#Hypotesis testing
#Chisquared test
chisq.test(l_ret.s, VG.q)
#K-S test
ks.test(as.numeric(X), rvg(length(as.numeric(X)),
                           param = c(c, sigma, theta, nu)))
#summary statistics
final_retVG<-VG_paths[,N]
basicStats(final_retVG)
hist(final_retVG)


VGexp=function(sigma, nu, theta, T, N, r, S0) {
  a=1/nu
  b=1/nu
  h=T/N
  t=(0:N)*T/N
  X=rep(0, N+1)
  I=rep(0,N)
  X[1]=S0
  for(i in 1:N) {
    I[i]=rgamma(1,a*h,b)
    X[i+1]=X[i]*exp(r*t+theta*I[i]+sigma*sqrt(I[i])*rnorm(1))
  }
  return(X)}
```

```
set.seed(123)
VGexp_paths<-matrix(nrow = nsim, ncol=N+1)
for(i in 1:nsim){
  VGexp_paths[i,]<-VGexp(sigma, nu, theta, T, N, r, S0)
}

VGexp_paths

plot(VGexp_paths[1,], col=0, type="l", ylim = c(min(VGexp_paths),max(VGexp_paths)),
     main = "MC Simlation for VG stock prices", sub = "100 steps, 10 paths",
     xlab = "Time", ylab = "Coke")
for(i in 2:nsim){
  lines(VGexp_paths[i,], col=colori[i], lwd = 2)purple!40!black;

}
final_pricesVG<-VGexp_paths[,N]

#mean correcting martingale method on vg
rn_final_pricesVG<-S0*(final_pricesVG)*(exp(r*T)/(mean(final_pricesVG)))
rn_final_pricesVG

basicStats(rn_final_pricesVG)
hist(rn_final_pricesVG)


payoff_VG <- pmax(rn_final_pricesVG - K, 0)
optprice_VG <- mean(payoff_VG)*exp(-r*T)
optprice_VG

#FFT
alpha <- 1.65
phiVG <- function(u) {
  omega <- (1/nu) * (log(1 - theta * nu - sigma^2 * nu/2))
  tmp <- 1 - (0+1i) * theta * nu * u + 0.5 * sigma^2 * u^2 * nu
  tmp <- tmp^(-1/nu)
  exp((0+1i) * u * log(S0) + u * (r + omega) * (0+1i)) * tmp
}

FFTcall.price <- function(phi, S0, K, r, T, alpha = 1, N = 2^12, eta = 0.25) {
  m <- r - log(phi(-(0+1i)))
  phi.tilde <- function(u) (phi(u) * exp((0+1i) * u * m))^T
  psi <- function(v) exp(-r * T) * phi.tilde((v - (alpha +
                                                          1) * (0+1i)))/(alpha^2 + alpha - v^2 +

  lambda <- (2 * pi)/(N * eta)
  b <- 1/2 * N * lambda
  ku <- -b + lambda * (0:(N - 1))
  v <- eta * (0:(N - 1))
  tmp <- exp((0+1i) * b * v) * psi(v) * eta * (3 + (-1)^(1:N) -
                                                   ((1:N) - 1 == 0))/3
  ft <- fft(tmp)
  res <- exp(-alpha * ku) * ft/pi
  inter <- spline(ku, Re(res), xout = log(K/S0))
  return(inter$y * S0)
}


```

```
FFTcall.price(phiVG, S0 = S0, K = K, r = r, T = T)



#Meixner model: it worked very bad, maybe I did some mistakes


x   <-mean(X, na.rm = TRUE)
y   <-sd(X, na.rm = TRUE)
z <-as.numeric(skewness(X, na.rm = TRUE))
w <-as.numeric(kurtosis(X, na.rm = TRUE))

m <- x-((z*sqrt(y))/(w-(z^2)-3))
a <- sqrt(y*(2*w-3*(z^2)-6))
b <- 2*atan(-sqrt((z^2)/(2*w-3*(z^2)-6)))
d <- 1/(w-(z^2)-3)
nsim = 100
N <- 100


#qq plot
MX.q <- uq(pinvd.new(udmeixner(a, b, d, m)), p) #compute the quantile

plot(MX.q, l_ret.s, main = "Meixner␣Q-Q␣Plot",
     xlab = "Theoretical␣Quantiles", ylab = "Sample␣Quantiles")

#esscher transform
theta <- -1/a * (b + 2 * atan((-cos(a/2)+ exp((m-r)/2*d))/sin(a/2)))
b <- a*theta+b #it doesn’t␣perform␣well

MX=function(a,␣b,␣d,␣m,␣N)␣{
␣␣distr␣<-␣udmeixner(a,␣b,␣d,␣m)␣#meixner␣distribution
␣␣gen␣<-␣pinvd.new(distr)␣#Polynomial␣interpolation␣of␣INVerse␣CDF
␣␣rdmMXgen␣<-␣ur(gen,N)␣#randomly␣draws␣N␣objects␣from␣gen␣(from␣a␣Meixner␣distr)
␣␣h=T/N
␣␣X=rep(0,N+1)
␣␣for␣(i␣in␣1:N){
␣␣␣␣X[i+1]=X[i]+rdmMXgen[i]
␣␣}
␣␣return(X)
}

MX(a,␣b,␣d,␣m,␣N)


set.seed(123)
MX_paths<-matrix(nrow␣=␣nsim,␣ncol=N+1)␣#fill␣the␣matrix␣with␣random␣paths␣that␣follow
for(i␣in␣1:nsim){␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣#the␣function␣MX␣just␣created
␣␣MX_paths[i,]<-MX(a,b,d,m,N)
}

MX_paths

plot(MX_paths[1,],␣col=0,␣type="l",␣ylim␣=␣c(min(MX_paths),max(MX_paths)),
␣␣␣␣␣main␣=␣"Monte␣Carlo␣Simulation␣for␣Meixner␣returns",␣sub␣=␣"100␣steps,␣100␣paths",
␣␣␣␣␣xlab␣=␣"Time",␣ylab␣=␣"MXNR␣returns")
for(i␣in␣2:nsim){
```

```
  lines(MX_paths[i,], col=colori[i], lwd = 2);
}



final_retMX<-MX_paths[,N]
basicStats(final_retMX)
hist(final_retMX)

#function for stock price with Meixner returns
MXexp=function(a, b, d, m, N, T, r, S0) {
  distr <- udmeixner(a, b, d, m) #meixner distribution
  gen <- pinvd.new(distr) #Polynomial interpolation of INVerse CDF
  generazioni <- ur(gen,N) #randomly draws N objects from gen (from a Meixner distr)
  h=T/N
  t=(0:N)*T/N
  X=rep(0,N+1)
  X[1]=S0
  for (i in 1:N){
    X[i+1]=X[i]*exp(r*t+generazioni[i])
  }
  return(X)
}


MXexp(a, b, d, m, N, T, r, S0)


set.seed(123)
MXexp_paths<-matrix(nrow = nsim, ncol=N+1)
for(i in 1:nsim){
  MXexp_paths[i,]<-MXexp(a,b,d,m,100,T,r,S0) #vengono tutte le linee uguali perch? MX non v
}

MXexp_paths

final_pricesMX<-MXexp_paths[,N]

payoff_MX <- pmax(final_pricesMX - K, 0)

optprice_MX <- mean(payoff_MX)*exp(-r*T)

optprice_MX



payoff_MXess <- pmax(rn_final_pricesMX - K, 0)
optprice_MXess <- mean(payoff_MXess)*exp(-r*T)
optprice_MXess


#mean correcting
#m <- r -2 *d*log(cos(b/2)/cos((a+b)/2)) is a mess
x   <-mean(X, na.rm = TRUE)
y   <-sd(X, na.rm = TRUE)
z <-as.numeric(skewness(X, na.rm = TRUE))
w <-as.numeric(kurtosis(X, na.rm = TRUE))
```

```
m <- x-((z*sqrt(y))/(w-(z^2)-3))
a <- sqrt(y*(2*w-3*(z^2)-6))
b <- 2*atan(-sqrt((z^2)/(2*w-3*(z^2)-6)))
d <- 1/(w-(z^2)-3)
nsim = 100
N <- 100

MX=function(a, b, d, m, N) {
  distr <- udmeixner(a, b, d, m) #meixner distribution
  gen <- pinvd.new(distr) #Polynomial interpolation of INVerse CDF
  rdmMXgen <- ur(gen,N) #randomly draws N objects from gen (from a Meixner distr)
  h=T/N
  X=rep(0,N+1)
  for (i in 1:N){
    X[i+1]=X[i]+rdmMXgen[i]
  }
  return(X)
}

MX(a, b, d, m, N)


set.seed(123)
MX_paths<-matrix(nrow = nsim, ncol=N+1) #fill the matrix with random paths that follow
for(i in 1:nsim){                          #the function MX just created
  MX_paths[i,]<-MX(a,b,d,m,N)
}

MX_paths
plot(MX_paths[1,], col=0, type="l", ylim = c(min(MX_paths),max(MX_paths)),
     main = "Monte Carlo Simulation for Meixner returns", sub = "100 steps, 100 paths",
     xlab = "Time", ylab = "MXNR returns")
for(i in 2:nsim){
  lines(MX_paths[i,], col=colori[i], lwd = 2);
}



final_retMX<-MX_paths[,N]
basicStats(final_retMX)
hist(final_retMX)

#function for stock price with Meixner returns
MXexp=function(a, b, d, m, N, T, r, S0) {
  distr <- udmeixner(a, b, d, m) #meixner distribution
  gen <- pinvd.new(distr) #Polynomial interpolation of INVerse CDF
  generazioni <- ur(gen,N) #randomly draws N objects from gen (from a Meixner distr)
  h=T/N
  t=(0:N)*T/N
  X=rep(0,N+1)
  X[1]=S0
  for (i in 1:N){
    X[i+1]=X[i]*exp(r*t+generazioni[i])
  }
  return(X)
}
```

```
MXexp(a, b, d, m, N, T, r, S0)


set.seed(123)
MXexp_paths<-matrix(nrow = nsim, ncol=N+1)
for(i in 1:nsim){
  MXexp_paths[i,]<-MXexp(a,b,d,m,100,T,r,S0) #vengono tutte le linee uguali perch? MX non v
}

MXexp_paths

final_pricesMX<-MXexp_paths[,N]

rn_final_pricesMX<-S0*(final_pricesMX)*(exp(r*T)/(mean(final_pricesMX)))

rn_final_pricesMX

payoff_MXmean <- pmax(rn_final_pricesMX - K, 0)

optprice_MXmean <- mean(payoff_MXmean)*exp(-r*T)

optprice_MXmean


### Multi asset options ###
#Rho correlation
GBM <- function(N, sigma, mu, S0, Wt = NULL) {
  if (is.null(Wt)) {
    Wt <- cumsum(rnorm(N, 0, 1))
  }
  t <- (1:N)/252
  p1 <- (mu - 0.5*(sigma*sigma)) * t
  p2 <- sigma * Wt
  St = S0 * exp(p1 + p2)
  return(St)
}

CorrelatedGBM <- function(N, S0, mu, sigma, cor.mat) {
  mu <- as.matrix(mu)
  sigma <- as.matrix(sigma)
  GBMs <- matrix(nrow = N, ncol = nrow(mu))
  Wt <- matrix(rnorm(N * nrow(mu), 0, 1), ncol = nrow(mu))
  Wt <- apply(Wt, 2, cumsum)
  chol.mat <- chol(cor.mat) # upper triangular cholesky decomposition
  Wt <- Wt %*% chol.mat   # key trick for creating correlated paths
  for (i in 1:nrow(mu)) {
    GBMs[,i] <- GBM(N, sigma[i], mu[i] , S0[i], Wt[, i])
  }
  return(GBMs)
}

GetPrices <- function(tickers, startDate='1992-01-02') {
  prices <- get.hist.quote(instrument = tickers[1], start = startDate,
                            quote = 'AdjClose')
  for (tik in 2:length(tickers)) {
```

```
    tmp <- get.hist.quote(instrument = tickers[tik],
                          start = start, quote = 'AdjClose')
    prices <- merge(prices, tmp)
  }
  return(prices)
}

set.seed(123)
N <- 2 * 252
t <- (1:N)/252
start <- '2002-1-1'
tickers <- c('KO', 'AXP')
prices <- GetPrices(tickers, start)

returns.mat <- as.matrix(na.omit(diff(log(prices))))
mean.vec <- as.numeric(colMeans(returns.mat))
sigma.vec <- as.numeric(sqrt(apply(returns.mat, 2, var)))
prices.vec <- as.numeric(prices[nrow(prices)])
cor.mat <-as.matrix(cor(returns.mat))

paths <- CorrelatedGBM(N, prices.vec , mean.vec, sigma.vec, cor.mat)

colors <- c('red', 'darkgreen')
plot(t, paths[,1], type = 'l', ylim = c(0, max(paths)), xlab = "Year",
     ylab = "Price", main = "Simulated Coca-Cola and American Express Prices", col = colors
for (i in 2:ncol(paths)) {
  lines(t, paths[, i], col = colors[i])
}
legend(x = 0.5, y = 25, c('KO', 'AXP'), lty = c(1,1), col = colors, cex = 0.7)

cor.mat


#Copula

#loglikCopula(th.C,returns.mat, claytonCopula())
clay <- fitCopula(claytonCopula(dim = 2), returns.mat, method ="itau")
gumb <- fitCopula(gumbelCopula(dim = 2), returns.mat, method ="itau")
frank <- fitCopula(frankCopula(dim = 2), returns.mat, method ="itau")
summary(clay)
summary(gumb)
summary(frank)


getSymbols("KO", from="2018-10-019", to="2020-07-05")
Coca = KO$KO.Adjusted
B = na.omit(diff(log(Coca)))


getSymbols("AXP", from="2018-10-019", to="2020-07-05")
amex = AXP$AXP.Adjusted
C = na.omit(diff(log(amex)))


M = cbind(B,C)
corKendall(as.matrix(M))

```

```
risk.measures <- function(x, alpha)
{
  if(!is.matrix(x)) x <- rbind(x)
  n <- nrow(x)
  d <- ncol(x)

  aloss  <- rowSums(x) # n-vector of aggregated losses
  VaR <- quantile(aloss, probs=alpha, names=FALSE) # VaR estimate
  l <- sum(xcd <- aloss > VaR)
  ES <- mean(aloss[xcd]) / (1-alpha) # ES estimate

  Alloc.first <- x[aloss>=VaR, 1] %*% rep(1/n, l) / (l/n) # capital allocated to X_1
  Alloc.mid   <- x[aloss>=VaR, floor(d/2)] %*% rep(1/n, l) / (l/n) # capital allocated to X
  Alloc.last  <- x[aloss>=VaR,d] %*% rep(1/n, l) / (l/n) # Capital Allocated to X_d

  ## return estimated risk measures
  c(VaR = VaR, ES = ES, Alloc.first = Alloc.first, Alloc.mid = Alloc.mid,
    Alloc.last = Alloc.last)
}

#Gumbel KO AXP
family.G <- "Gumbel"
tau = 0.3577591
alpha = 0.01
d = 2
n <- 1e5
nu = 3 #degress of freedom of the Student's, but can't find its copula in the code
th.G <- iTau(getAcop(family.G), tau) #theta, the corresponding parameter
gumbel.cop <- onacopulaL(family.G, nacList=list(th.G, 1:d))
set.seed(123)
U.CDM  <- matrix(runif(n*d), ncol=d) # pseudo
U.C.CDM  <- cCopula(U.CDM,  cop=gumbel.cop, inverse=TRUE)

#erT <- exp(-r*T)
rm.C.CDM <- risk.measures(U.C.CDM, alpha)

res <- array(dim=c(2,2,1), dimnames=list(type=c(paste0("VaR.", alpha), paste0("ES.", alpha)
                                          copula=c("Gumbel", paste0("t", nu)),
                                          method=c("CDM")))
res[paste0("VaR.", alpha),,] <- matrix(c(rm.C.CDM[1]), ncol=1)
res[paste0("ES.", alpha),,]  <- matrix(c(rm.C.CDM[2]), ncol=1)
res



#Clayton
family.C <- "Clayton"
th.C <- iTau(getAcop(family.C), tau) # corresponding parameter
clayton.cop <- onacopulaL(family.C, nacList=list(th.C, 1:d))

set.seed(123)
U.CDM  <- matrix(runif(n*d), ncol=d) # pseudo
ClayU.C.CDM  <- cCopula(U.CDM,  cop=clayton.cop, inverse=TRUE)
#erT <- exp(-r*T)
Clayrm.C.CDM <- risk.measures(ClayU.C.CDM, alpha)
res <- array(dim=c(2,2,1), dimnames=list(type=c(paste0("VaR.", alpha), paste0("ES.", alpha)
                                          copula=c("Clayton", paste0("t", nu)),
```

```
                                           method=c("CDM")))
res[paste0("VaR.", alpha),,] <- matrix(c(Clayrm.C.CDM[1]), ncol=1)
res[paste0("ES.", alpha),,]  <- matrix(c(Clayrm.C.CDM[2]), ncol=1)
res

#Frank
family.F <- "Frank"
th.F <- iTau(getAcop(family.F), tau) # corresponding parameter
frank.cop <- onacopulaL(family.F, nacList=list(th.F, 1:d)) # Frank copula
### 2.2 Sampling
set.seed(123)
U.CDM  <- matrix(runif(n*d), ncol=d) # pseudo
FrankU.C.CDM  <- cCopula(U.CDM,  cop=frank.cop, inverse=TRUE) # pseudo
### 2.3 Functional Calculation
#erT <- exp(-r*T)
Frankrm.C.CDM <- risk.measures(FrankU.C.CDM, alpha)
### 2.4 Results
res <- array(dim=c(2,2,1), dimnames=list(type=c(paste0("VaR.", alpha), paste0("ES.", alpha)
                                           copula=c("Frank", paste0("t", nu)),
                                           method=c("CDM")))
res[paste0("VaR.", alpha),,] <- matrix(c(Frankrm.C.CDM[1]), ncol=1)
res[paste0("ES.", alpha),,]  <- matrix(c(Frankrm.C.CDM[2]), ncol=1)
res

#to compare the time consuming
system.time(cCopula(U.CDM,  cop=frank.cop, inverse=TRUE))
system.time(cCopula(U.CDM,  cop=gumbel.cop, inverse=TRUE))
system.time(cCopula(U.CDM,  cop=clayton.cop, inverse=TRUE))


### Multi-Asset Options based on L vy Processes ###

#Ballotta-Bonfiglioli model

VG=function(sigma, nu, mu, T, N) {
  a=1/nu
  b=1/nu
  h=T/N
  t=(0:T)/N
  X=rep(0, N+1)
  I=rep(0,N)
  X[1]=0
  for(i in 1:N) {
    I[i]=rgamma(1,a*h,b)
    X[i+1]=X[i] + mu*I[i]+sigma*sqrt(I[i])*rnorm(1)
  }
  return(X)
}
time <- (0:10000) / 10000

sigma1 <- 0.75
nu1 <- 0.5
mu1 <- 0.1

sigma2 <- 0.70
nu2 <- 0.3
mu2 <- 0.2
```

```
sigma3 <- 0.60
nu3 <- 0.2
mu3 <- 0.3

VG(sigma1, nu1, mu1, 1, 10000)
VG(sigma2, nu2, mu2, 1, 10000)
VG(sigma3, nu3, mu3, 1, 10000)
set.seed(1)
plot(x=time, y=VG(sigma1, nu1, mu1, 10, 10000)+VG(sigma3, nu3, mu3, 1, 10000), type="l",  y
lines(x =time, y = VG(sigma2, nu2, mu2, 1, 10000)+VG(sigma3, nu3, mu3, 1, 10000), col="red"

# Simulation of a bivariate Levy process with finite variation and
# Clayton Levy copula.
#
# Author: Alice Pignatelli di Cerchiara

# maximum number of jumps?
m <- 2000
n <- 10
d <- 1000
# parameters of Clayton Levy copula (chosen as in Figure 1)

# Marginal (1st variance gamma) parameters
c1 <- 10
lambda1_plus <- 1
lambda1_minus <- 1
c2 <- 10
lambda2_plus <- 1
lambda2_minus <- 1

eta <- 0.9
theta <- 0.3

## NOTE: In Tankov they use a different parameterisation?
## [cf. Example 2.1; especially eq. (2.4)]

# t in grid_pts
grid_pts <- (1:d) / d # has to be subset of [0,1]

# define the inverse of the conditional distribution function of U|xi
F_inv <- function(xi, u) {

  # define the two functions B and C internally
  B <- function(xi, u) {
    res <- sign(u - 1 + eta) * (xi >= 0) + sign(u - eta) * (xi < 0)
    return(res)
  }

  C <- function(xi, u) {
    res1 <- (((u - 1 + eta)/eta) * (u >= 1 - eta) + ((1 - eta - u)/(1-eta)) * (u < 1 - eta)
    res2 <- (((u - eta)/(1-eta)) * (u >= eta) + ((eta-u)/eta) * (u < eta)) * (xi < 0)
    return(res1 + res2)
  }

  res <- B(xi,u) * abs(xi) * (C(xi,u)^(-theta/(theta+1)) - 1)^(-1/theta)
  return(res)
```

```
}
#FROM https://cran.r-project.org/web/packages/copula/vignettes/NALC.html
# tail integral for variance gamma
nu_bar_vargamma <- function(x, th, kap, sig, c, lambda_plus, lambda_minus) {
  if (!hasArg(lambda_plus)) {
    lambda_plus <- (sqrt(th^2+2*sig^2/kap)-th)/sig^2
  }
  if (!hasArg(lambda_minus)) {
    lambda_minus <- (sqrt(th^2+2*sig^2/kap)+th)/sig^2
  }
  if (!hasArg(c)) {
    c <- 1/kap
  }
  lambda <- lambda_plus * (x > 0) + lambda_minus * (x < 0)
  -c*expint_Ei(-lambda*abs(x), give=FALSE)
}

## Inverse of the tail integral of a variance gamma Levy process
nu_bar_inv_vargamma <- function(Gamma, th, kap, sig, c, lambda_plus, lambda_minus, ...)
{
  if (!hasArg(lambda_plus)) {
    lambda_plus <- (sqrt(th^2+2*sig^2/kap)-th)/sig^2
  }
  if (!hasArg(lambda_minus)) {
    lambda_minus <- (sqrt(th^2+2*sig^2/kap)+th)/sig^2
  }
  if (!hasArg(c)) {
    c <- 1/kap
  }


  max.val <- nu_bar_vargamma(.Machine$double.xmin, th=th, kap=kap, sig=sig, c=c, lambda_plu
  res <- numeric(length(Gamma))
  large <- (Gamma >= max.val | Gamma <= -1*max.val)
  res[large] <- 0 # de facto indistinguishable from 0 anyways
  if(any(!large)) {
    #lambda <- (sqrt(th^2+2*sig^2/kap)-th)/sig^2
    nu_bar_vargamma_minus <- function(x, z) {
      lambda <- lambda_plus * (x > 0) + lambda_minus * (x < 0)
      -c*expint_Ei(-lambda*abs(x), give=FALSE) - abs(z)
    }

    if (any(Gamma > 0)) {
      res[!large & Gamma > 0] <- vapply(Gamma[!large & Gamma > 0], function(Gamma.)
        uniroot(nu_bar_vargamma_minus, z=Gamma.,
                interval=c(.Machine$double.xmin, 29))$root, NA_real_)
    }

    if (any(Gamma < 0)) {
      res[!large & Gamma < 0] <- vapply(Gamma[!large & Gamma < 0], function(Gamma.)
        uniroot(nu_bar_vargamma_minus, z=Gamma.,
                interval=c(-29, -1*.Machine$double.xmin))$root, NA_real_)
    }
  }
  res
}

```

```
Z1␣<-␣matrix(nrow␣=␣n,␣ncol␣=␣d)

Z2␣<-␣matrix(nrow␣=␣n,␣ncol␣=␣d)


##␣[END]␣COPIED␣(and␣modified)␣FROM␣https://cran.r-project.org/web/packages/copula/vignettes
for␣(r␣in␣1:n)␣{
␣␣#␣now␣let's simulate the two levy processes?
  # along the lines of Theorem 4.3 in the Tankov-paper

  # define X: jump times of Poisson process with lambda = 2
  inter_arrival_times <- rexp(m, 2)
  X <- cumsum(inter_arrival_times)

  # define sequence {Gamma_i^1 : i \in 1, ..., m} as in Remark 4.3
  Gamma1 <- (-1)^(1:m) * X

  # define the sequence {Gamma_i^2 : i \in 1, ..., m} as on page 14
  W <- runif(m) # W_i \sim U[0,1]

  # THIS IS BAD STYLE AND WE WILL IMPROVE IT LATER!
  Gamma2 <- c()
  for (i in 1:m) {
    Gamma2[i] <- F_inv(Gamma1[i], W[i])
  }

  U1_inv <- function(Gamma) {
    res <- nu_bar_inv_vargamma(Gamma, c = c1, lambda_plus = lambda1_plus, lambda_minus = la
  }
  U1_inv <- Vectorize(U1_inv)


  U2_inv <- function(Gamma) {
    res <- nu_bar_inv_vargamma(Gamma, c = c2, lambda_plus = lambda2_plus, lambda_minus = la
  }
  U2_inv <- Vectorize(U2_inv)

  # The next lines are according to eq. (4.9) in the Tankov-paper

  # use a common sequence {V_i : i \in 1, ..., m} as in Theorem 4.3
  V <- runif(m)

  # the function aux computes (4.9) for one t in [0,1]
  U_inv_of_Gamma1 <- U1_inv(Gamma1) # otherwise it will be computed length(grid_pts) times
  aux1 <- function(time_t) {
    sum(U_inv_of_Gamma1 * (V <= time_t))
  }
  aux1 <- Vectorize(aux1)
  Z1[r,] <- sapply(grid_pts, aux1)

  U_inv_of_Gamma2 <- U2_inv(Gamma2) # otherwise it will be computed length(grid_pts) times
  aux2 <- function(time_t) {
    sum(U_inv_of_Gamma2 * (V <= time_t))
  }
  aux2 <- Vectorize(aux2)
  Z2[r,] <- sapply(grid_pts, aux2)
}
```

```
plot(x = grid_pts, y = Z1[1,], type="l", ylim=c(min(Z1, Z2), max(Z1, Z2)), ylab="Z1␣(black),
lines(x = grid_pts, y = Z2[1,], col="red")

for (r in 2:n) {
  lines(x = grid_pts, y = Z1[r,], col="black")
  lines(x = grid_pts, y = Z2[r,], col="red")
}



###My contribution: Best copula###
{
#rank scatter plot
scatco = -coredata(B) #I change the sign of the return, I find it more convenient in comput
scatcoca = pobs(scatco)
scatam = -coredata(C)
scatamex = pobs(scatam)
plot(scatcoca,scatamex, pch=19, col='red', cex=0.5, xlab='KO', ylab='AXP', main = "Scatter␣

M = cbind(scatco,scatamex)
corKendall(as.matrix(M)) #same as before even if I changed the sign

#add sigma to the t distribution
dt_ls = function(x, df=1, mu=0, sigma=1) 1/sigma * dt((x - mu)/sigma, df)
pt_ls = function(q, df=1, mu=0, sigma=1)  pt((q - mu)/sigma, df)
qt_ls = function(p, df=1, mu=0, sigma=1)  qt(p, df)*sigma + mu
rt_ls = function(n, df=1, mu=0, sigma=1)  rt(n,df)*sigma + mu

par(mfrow=c(2,1))

#selection of best marginals: https://stats.stackexchange.com/questions/132652/how-to-deter
#for KO
x = unmatrix(scatco, byrow = TRUE)
x_fit <= fitDist(x, k = 2, type = "realline", trace = TRUE, try.gamlss = TRUE)
summary(x_fit)
x_fit$fits #look for the smallest AIC
barplot(x_fit$fits, main = "AIC␣score␣for␣KO") #TF the best: t family distribution
x_mod.t = fitdistrplus::fitdist(x, 't_ls', start = list(df=1,mu=mean(x),sigma=sd(x)))
summary(x_mod.t)

#for AXP
y = unmatrix(scatam, byrow = TRUE)
y_fit = fitDist(y, k = 2, type = "realline", trace = TRUE, try.gamlss = TRUE)
summary(y_fit)
y_fit$fits #look for the smallest AIC
barplot(y_fit$fits, main = "AIC␣score␣for␣AXP") #Johnson's␣SU-distribution␣the␣best,␣but␣the
y_mod.t = fitdistrplus::fitdist(y, 't_ls', start = list(df=1,mu=mean(x),sigma=sd(x)))
summary(y_mod.t)

#params
x_parameters =list('df'=x_mod.t$estimate[[1]],
                    'mu'=x_mod.t$estimate[[2]],
                    'sigma'=x_mod.t$estimate[[3]])
x_type <-'t_ls'
y_parameters =list('df'=y_mod.t$estimate[[1]],
                    'mu'=y_mod.t$estimate[[2]],
                    'sigma'=y_mod.t$estimate[[3]])
```

```
y_type <-'t_ls'

par(mfrow=c(1,1))

#plots
#x
hist(x, breaks = 100, probability = T,
     col='grey', border = 'white', main = "Density␣comparison␣of␣KO")
points(density(x),
       type='l', col='red', lwd=2)
points(sort(x), dt_ls(sort(x),
                      x_parameters[[1]],
                      x_parameters[[2]],
                      x_parameters[[3]]),
       type='l', col='purple', lwd=2)

#y
hist(y, breaks = 100, probability = T,
     col='grey', border = 'white', main = "Density␣comparison␣of␣AXP")
points(density(y),
       type='l', col='red', lwd=2)
points(sort(y), dt_ls(sort(y),
                      y_parameters[[1]],
                      y_parameters[[2]],
                      y_parameters[[3]]),
       type='l', col='purple', lwd=2)

#best copula selection
x_probs =pt_ls(x,
                x_parameters[[1]],
                x_parameters[[2]],
                x_parameters[[3]])

y_probs =pt_ls(y,
                y_parameters[[1]],
                y_parameters[[2]],
                y_parameters[[3]])

selectedCopula =BiCopSelect(x_probs, y_probs, familyset = c(0,1,2,3,4,5,6), se=T) #in theor
                                                                              #But for the sake of
selectedCopula␣#best␣copula:␣t␣(par␣=␣0.52,␣par2␣=␣2.03,␣tau␣=␣0.34)

#best␣copula
copula_fit␣<-fitCopula(tCopula(dim=2,␣df=2,␣df.fixed␣=␣T),␣cbind(x_probs,␣y_probs),␣method=
copula_fit
summary(copula_fit)
t_copula␣<-copula_fit@copula

#normal␣copula␣(will␣be␣useful␣later)
copula_fit␣<-fitCopula(normalCopula(dim=2),␣cbind(x_probs,␣y_probs),␣method='irho')
copula_fit
summary(copula_fit)
normal_copula␣<-copula_fit@copula

#independent␣copula
independent_copula␣<-indepCopula(dim=2)
```

```
#User defined function to count how many times when the simulated KO var at 99%, AXP exceed
  MonteCarlo <-function(copula, M, verbose){
    print(copula)
    counting <-matrix(0, M, 1)
    for (i in 1:M) {

      #simulation
      biv_probs = rCopula(10000, copula)
      x = qt_ls(biv_probs[,1], x_parameters[[1]], x_parameters[[2]], x_parameters[[3]])
      y = qt_ls(biv_probs[,2], y_parameters[[1]], y_parameters[[2]], y_parameters[[3]])
      data_sim = cbind(x, y)

      #find the index of the observations when Y=VaR_99(Y) (Empirical)
      data_sim_ord = data_sim[order(data_sim[,1]), ]
      idx = nrow(data_sim)*0.99

      #compare X and Y (if X>=VaR_99(Y) count 1, otherwise count 0)
      counting[i,1] = data_sim_ord[idx,2]>=data_sim_ord[idx,1]

      if (verbose) {
        cat(paste(i, ''))
      }
    }
    return(sum(counting)/nrow(counting))
  }

set.seed(123)
M = 1000
verbose = T

#normal
prob_normal_copula = MonteCarlo(normal_copula, M, verbose)
cat('\n\n')

#tcopula
prob_t_copula = MonteCarlo(t_copula, M, verbose)
cat('\n\n')

#independent copula
prob_independent = MonteCarlo(independent_copula, M, verbose)
cat('\n\n')

Prob <- list(
  't-copula'=prob_t_copula,
  'Gaussian-copula'=prob_normal_copula,
  'Independent-copula'=prob_independent)
Prob
}
#RSI indicator
{
# Create an RSI with a 3-day lookback period
spy_rsi <- RSI(price = Cl(KO), n = 3)

# Plot the closing price of KO
plot(Cl(KO))

# Plot the RSI 2
```

```
plot(RSI(Cl(KO), n = 3))
}
```