Politecnico di Milano
School of Industrial and Information Engineering
Master of Science in Mathematical Engineering

BITCOIN AS A DIGITAL ASSET: Correlation and Optimal Portfolio Allocation

Supervisors: Prof. Daniele Marazzina
Prof. Ferdinando M. Ametrano

Master thesis by:
Samuele Vianello
ID: 875534

Academic year 2017-2018

Contents

List of Tables

List of Figures

Abstract

Bitcoin is increasingly establishing itself as a digital investment asset other than a mere speculative instrument. In this work, we explore this subject by studying the correlation that Bitcoin has with other standard assets such as stock and bond indexes, currencies and commodities. We first study the empirical correlation and its statistical significance, then calibrate more sophisticated asset models including jump diffusion and stochastic volatility in their multivariate generalizations to obtain the correlation matrices also under those frameworks. Results are closely related and Bitcoin correlation with any other asset is confirmed to be very low. Given this result, we perform optimal portfolio allocation analyses to investigate its diversification properties, both through Markowitz mean-variance optimization and using the CVaR as the portfolio risk measure. Evidence shows that allocating a small percentage of wealth in the digital assets proves to be extremely beneficial in terms of lowering the risk and increasing the expected returns.

Chapter 1

Introduction

Bitcoin was presented in (Nakamoto, 2009) as a peer-to-peer protocol for electronic cash that would allow online payments to be sent directly from one party to another without going through a financial institution. It started off in 2009 as system that was only used by a niche of people in online cryptology forums to experiment with the transaction protocol. It took a few years for Bitcoin to gain public notoriety, while the price kept rising and having quick crashes.

In particular, Bitcoin generated a lot of buzz in 2017, a year that registered the price increase from 998 $ to 13,412 $ in January 2018 with an all-time high of 19,666 $ on the 17th of December. Like during any of history's gold rushes, many people joined the trend and tried to jump on the train of quick and easy money and were let down when the price deflated to a level of six thousand dollars in 2018 and has lately stabilized around three thousands.

Setting aside the mere numerical value of the price, Bitcoin was the first protocol to solve the problem of double-spending without the need for a centralized party: bitcoins can be transferred but not duplicated, as they only exist as validated transaction in the distributed blockchain. These features allow Bitcoin to have a chance at becoming a global, instantaneous and free payment network that would make wealth transfer as easy as online data sharing. In the same way that e-mail substituted post mail, Wikipedia and other knowledge-based website outdated paper encyclopaedias, music and film streaming services are becoming the new user friendly experience for the two industries, Bitcoin presents itself as a system to exchange wealth between users without the need for banks or other trusted third parties.

Furthermore, Bitcoin is the first digital currency to achieve scarcity in the digital realm: its monetary policy based on deterministic supply mimics

the progressive scarcity of gold. It is this *deterministic* supply that make Bitcoin a suitable long term investment, since it prevents arbitrary increase in its total available amount, unlike what happens with *fiat* money.

For these reasons we believe that Bitcoin can be *digital gold* with an embedded secure network and its characteristics make it resemble more closely a crypto-asset rather than a crypto-currency. Even though there are a number of supporters of this idea, for instance in (Dyhrberg, 2016) it is shown that Bitcoin possesses the same hedging abilities of gold, there is yet no general consensus on the matter and different studies get to the opposite result: see for example (Klein et al., 2018).

Bitcoin has been called many names: a bubble, a Ponzi scheme, but also defined as sound money and store of value. We agree with the latter and believe that if Bitcoin is true digital gold, then its value will express the huge potential that has been so far limited by scepticism and misunderstandings.

In the present work, we intend to first study the correlation that exists between Bitcoin and other types of standard assets, both by analysing the empirical correlation of the returns with its significance and by calibrating more sophisticated models such as *jump diffusion* and *stochastic volatility* models. Secondly, we want to explore the diversification property of adding Bitcoin to a portfolio of asset, by computing the optimal allocation for different levels of risk and expected return in a Markowitz mean-variance framework and by optimizing on the CVaR as the portfolio risk measure.

The inclusion of Bitcoin in an investment portfolio is strongly suggested in (Bouri et al., 2017) where the authors arrive to the conclusion that the digital currency is indeed a great diversifier, while not performing as well as a hedge or safe haven. In (Andrianto, 2017) the diversification properties of Bitcoin are studied by inserting it into a portfolio composed of 8 assets including stocks, currencies and commodities and performing a Markowitz mean-variance analysis. Finally in (Eisl et al., 2015) the authors study the performance of a portfolio that includes Bitcoin adopting a CVaR framework: the result is that a small percentage of wealth should be invested in the digital asset. In the last part of our thesis, we aim to update the results of the papers we just referenced by including the most recent price data, considering a total of 16 assets (15 standard assets plus Bitcoin) and performing a comparison between the two asset allocation frameworks. Our final results match the outcomes of those articles.

The present thesis has been written during the author's fellowship at the Digital Gold Institute(https://www.dgi.io/), a research and development center

focused on teaching, consulting, and advising about scarcity in the digital domain (Bitcoin and crypto-assets) and the underlying blockchain technology.

1.1 Thesis structure

Our work is structured into 5 chapters.

In Chapter 2 we study the empirical correlation between the assets taken into considerations (Section 2.1). We also take a look at the significance of each correlation value by performing two statistical tests, Pearson's t-test and a permutation test, in Section 2.2. Finally, Section 2.3 investigates the rolling correlation between Bitcoin and the other assets.

In Chapter 3 we present each asset model that we are going to adopt in our work. We first give a brief overview of the building blocks necessary to understand the mathematical framework of the models: definitions of geometric Brownian motions, compound Poisson processes and CIR processes are all presented in Section 3.1. In the following sections we consider the models themselves. We present the univariate formulations of Merton (Section 3.2), Heston (Section 3.3) and Bates (Section 3.4) and provide a possible generalization to the multi-asset case following the parsimonious approach as reported in (Dimitroff et al., 2011)

In Chapter 4 we explain how to calibrate each different model: first we present a general overview of the maximum likelihood estimation (Section 4.1) and then we proceed to specify this approach for our three models in Sections 4.2 , 4.3 and 4.4. The last section of the chapter is devoted to the details of our implementation and to the presentation of the numerical results (Section 4.5).

Chapter 5 is where we study the advantages of including Bitcoin in our portfolio. In Section 5.1 we present the general framework for the *modern portfolio theory* by Markowitz. The following Section 5.2 shows how to plot the efficient frontier and the results for the optimal allocation problem. In Section 5.3 we change the portfolio risk measure to the daily CVaR and in Section 5.4 the efficient frontier and the allocations with the CVaR are presented with a comparison to those of Markowitz.

Chapter 6 concludes our work and sums up the main results. It also includes some final remarks to this study.

1.2 Dataset Presentation

The dataset on which we focus our study contains 2163 observations of the prices of 16 assets valued daily (excluding holidays and weekends) from the 19th of July 2010 till the 2nd of November 2018 (all data provided by Bloomberg).

The assets we included in our analysis are grouped into five different classes , as explained in the following list (in brackets we indicate the shortened name that is used in the tables and graphs):

1. Bitcoin (btc): Value of a single bitcoin, quoted in dollars.

2. Stock indexes:

   - S&P500 (sp500): American stock market index based on 500 large company with stock listed either on the NYSE or NASDAQ.
   - EURO STOXX 50 (eurostoxx): equity index of eurozone stocks, covering 50 stocks from 11 eurozone countries.
   - MSCI BRIC (bric): market cap weighted index designed to measure the equity market performance across the emerging country indexes of Brazil, Russia, India and China.
   - NASDAQ(nasdaq): market cap weighted index including all NASDAQ tiers: Global Select, Global Market and Capital Market.

3. Bond indexes:

   - BBG Pan European (bond_europe): Bloomberg Barclays Pan-European Aggregate Index that tracks fixed-rate, investment-grade securities issued in different European currencies.
   - BBG Pan US (bond_us): Bloomberg Barclays US Aggregate Bond Index, a benchmark that measures investment grade, US dollar-denominated, fixed-rate taxable bond market.
   - BBG Pan EurAgg (bond_eur): similar to the Pan European but it only considers securities issued in Euros.

4. Currencies:

   - EUR/USD (eur): spot value of one Euro in US dollars.
   - GBP/USD (gbp): spot value of one British Pound in US dollars.

- CHF/USD (chf): spot value of one Swiss Franc in US dollars.
- JPY/USD (jpy): spot value of one Japanese Yen in US dollars.

5. Commodities:

   - Gold (gold): price of gold measured in USD/Oz.
   - WTI (wti): price of crude oil used as benchmark in oil pricing and as the underlying commodity in the NYMEX oil future contracts.
   - Grain (grain): S&P GPSCI index that measures the performance of the grain commodity market.
   - Metals (metal): S&P GSCI Industrial Metals index that measures the movements of industrial metal prices including aluminium, copper, zinc, nickel and lead.

Chapter 2

Correlation Analysis

In order to get an initial insight on how Bitcoin is correlated with other assets, we will perform a correlation analysis based on the empirical time series of our data. We will focus our attention on the logarithmic returns as it is the standard practice. We will often refer to logarithmic returns simply as returns, only specifying their nature when it is necessary to avoid confusion.

2.1 Empirical Correlation of Returns

We first start by performing some statistical analysis on the data in order to estimate the distribution from which they are sampled. For this part, we will consider our data as successive samples of a N-dimensional vector in $\mathbb{R}^N$, where N is the number of assets:

$$\mathbf{x}_j = \begin{pmatrix} x_{1,j} \\ x_{2,j} \\ \vdots \\ x_{N,j} \end{pmatrix}, j = 1 \dots N_{sample}.$$

Each element i of the vector $\mathbf{x}_j$ represents the j^{th} realization of the returns for asset i.

Following basic statistics, we can now compute the *sample mean* of our vectors of returns as:

$$\bar{\mathbf{x}} = \frac{1}{N_{sample}} \sum_{j=1}^{N_{sample}} \mathbf{x}_j = \begin{pmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_N \end{pmatrix},$$

Table 2.1: Annualized returns and volatilities obtained from sample. We use $T = 255$ as the number of trading days in a year.

	Annualized Mean Return	Annualized Volatility
btc	1.331	1.065
bric	-0.010	0.174
sp500	0.111	0.142
eurostoxx	0.011	0.223
nasdaq	0.143	0.163
bond_europe	0.023	0.082
bond_us	0.024	0.032
bond_eur	0.022	0.088
eur	-0.013	0.086
gbp	-0.018	0.084
chf	0.006	0.107
jpy	-0.030	0.091
gold	0.004	0.156
wti	-0.022	0.320
grain	-0.062	0.218
metal	-0.028	0.184

where $\bar{x}_i = \frac{1}{N_{sample}} \sum_{j=1}^{N_{sample}} x_{i,j}$ is the sample mean of component i.

We then compute the *sample covariance matrix* through the following formula:

$$\bar{\Sigma} = \frac{1}{N_{sample} - 1} \sum_{j=1}^{N_{sample}} (\mathbf{x}_j - \bar{\mathbf{x}})(\mathbf{x}_j - \bar{\mathbf{x}})^T,$$

where $\bar{\mathbf{x}}$ represent the sample mean of the returns just introduced.

All the information needed to obtain the *correlation matrix* C are already included in $\bar{\Sigma}$, we only need to perform some further calculations:

$$C_{i,j} = \frac{\bar{\Sigma}_{i,j}}{\sqrt{\bar{\Sigma}_{i,i}\bar{\Sigma}_{j,j}}}. \tag{2.1}$$

We have thus obtained an empirical estimate of the correlation between our assets returns. The formula in (2.1) is often referred to as *Pearson correlation coefficient*, from the name of the English mathematician Karl Pearson who first formulated it.

Results for the mean returns and the volatilities are inserted in Table 2.1, while in Table 2.2 is reported the correlation matrix.

We are mainly interested in the correlation between Bitcoin and other assets returns, so we will now focus on the first row (or equivalently column,

Table 2.2: Empirical correlation matrix obtained through Pearson's estimation. The values are in percentages and the colour goes from red for $\rho = 100\%$, to white for $\rho = 0\%$ and to blue for $\rho = -100\%$.

	btc	bric	sp500	eurostoxx	nasdaq	bond_europe	bond_us	bond_eur	eur	gbp	chf	jpy	gold	wti	grain	metal
btc	100.0	1.4	4.4	4.1	3.6	1.4	-1.8	1.9	2.3	0.7	2.5	-1.1	-0.2	0.8	3.5	2.7
bric	1.4	100.0	48.4	57.1	47.4	19.6	-15.1	19.8	20.1	24.2	8.1	-16.2	12.9	30.3	15.2	43.2
sp500	4.4	48.4	100.0	62.0	94.9	13.3	-34.1	15.0	18.4	21.1	0.1	-22.2	-0.6	35.1	15.2	34.9
eurostoxx	4.1	57.1	62.0	100.0	56.2	42.0	-27.8	44.7	48.6	41.9	21.2	-16.5	9.3	32.7	15.2	46.5
nasdaq	3.6	47.4	94.9	56.2	100.0	10.9	-31.4	12.3	15.1	18.6	-1.8	-21.4	-1.0	29.3	14.1	32.4
bond_europe	1.4	19.6	13.3	42.0	10.9	100.0	19.4	98.4	91.9	61.8	60.1	39.9	42.8	14.5	11.6	26.8
bond_us	-1.8	-15.1	-34.1	-27.8	-31.4	19.4	100.0	14.5	-0.4	-5.0	14.0	38.6	21.3	-21.1	-6.8	-17.1
bond_eur	1.9	19.8	15.0	44.7	12.3	98.4	14.5	100.0	94.5	53.6	58.4	37.0	40.7	14.3	11.3	28.0
eur	2.3	20.1	18.4	48.6	15.1	91.9	-0.4	94.5	100.0	57.1	59.4	31.2	36.7	18.2	13.5	31.0
gbp	0.7	24.2	21.1	41.9	18.6	61.8	-5.0	53.6	57.1	100.0	35.5	14.2	24.7	21.9	11.9	26.1
chf	2.5	8.1	0.1	21.2	-1.8	60.1	14.0	58.4	59.4	35.5	100.0	36.7	37.1	6.7	7.5	20.8
jpy	-1.1	-16.2	-22.2	-16.5	-21.4	39.9	38.6	37.0	31.2	14.2	36.7	100.0	39.5	-6.5	2.1	-3.1
gold	-0.2	12.9	-0.6	9.3	-1.0	42.8	21.3	40.7	36.7	24.7	37.1	39.5	100.0	14.7	13.7	32.0
wti	0.8	30.3	35.1	32.7	29.3	14.5	-21.1	14.3	18.2	21.9	6.7	-6.5	14.7	100.0	17.8	36.0
grain	3.5	15.2	15.2	15.2	14.1	11.6	-6.8	11.3	13.5	11.9	7.5	2.1	13.7	17.8	100.0	20.7
metal	2.7	43.2	34.9	46.5	32.4	26.8	-17.1	28.0	31.0	26.1	20.8	-3.1	32.0	36.0	20.7	100.0

by symmetry) of the correlation matrix. All these values are fairly close to zero, never even exceeding 5% towards the positive or the negative side. One may thus wonder whether these correlations are *statistically significantly* different from zero. To answer this question, we introduce two statistical tests to check the correlation significance.

2.2 Correlation Significance

The core of Inferential Statistics, the branch of statistics that allows to draw conclusions from the information contained in a set of data, is hypothesis testing.

In our case, we are specifically interested in testing if the sample correlation coefficients are significantly different from zero or not. Both of the following tests are presented in the most general form for a sample of two variables, with their distribution correlation ρ and their sample correlation $\hat{\rho}$.

Following standard testing procedure, we specify the *null hypothesis* and the *alternative hypothesis*:

$$\mathbf{H_0}: \quad \rho = 0 \quad vs. \quad \mathbf{H_1}: \quad \rho \neq 0.$$

These will be common to both presented tests.

2.2.1 Pearson's t-test

Our first test is based on Student's t-distribution and the following t-statistic:

$$t = \hat{\rho}\sqrt{\frac{n-2}{1-\hat{\rho}^2}}, \tag{2.2}$$

which under the null hypothesis is distributed as a Student's t with $n-2$ degrees of freedom, where n stands for the cardinality of the sample. We can thus proceed by computing the relative p-value and compare it to a given level of confidence α (usually $\alpha = 95\%$). The result of the test will be deduced as follows:

- $p - value < 1 - \alpha$: we have statistical evidence to state that the correlation is *significantly* different from zero;
- $p - value \geq 1 - \alpha$: there is *no statistical evidence* to state that the correlation is different from zero.

2.2.2 Permutation test

The permutation test is based on building an empirical distribution of values for the correlation by sampling different pairs of X and Y variable and then computing Pearson's correlation. Let (x_i, y_i) be the original pairs for $i = 1, \ldots, N_{sample}$. Create a new dataset (x_i, y_{i^*}) by replacing y_i with one of the possible $N_{sample}!$ permutations[1] and then compute the new sample correlation for the new dataset. If this is done a large enough number of times, we obtain an empirical distribution of possible values for the correlation of x and y. From this distribution we can then obtain the p-value of the test and thus get the final result in the same way as in the previous case.

2.2.3 Significance results

The values that we obtained for the correlation of Bitcoin with the other assets are reported in Table 2.3, including the resulting p-values for both of the tests that were introduced in the paragraph above.

Looking at the first line alone, we can see that the asset-Bitcoin correlation never surpasses 5% in absolute value. This is exactly what we would expect given that Bitcoin price seems to move on its and not really care about what is happening on the market (at least to some degree).

[1]The ! represents the factorial of a number: $n! = 1 \times 2 \times ... \times (n-1) \times n$.

Table 2.3: Values of the correlation between Bitcoin and the other assets and their p-value using both Pearson's and the permutation test.

	bric	sp500	eurostoxx	nasdaq	bond_europe	bond_us	bond_eur
Correlation	1,41%	4,37%	4,12%	3,59%	1,41%	-1,84%	1,92%
Pearson	52,70%	4,10%	5,40%	9,35%	50,35%	39,90%	37,95%
Permutation	51,21%	4,24%	5,55%	9,48%	51,29%	39,28%	37,20%

	eur	gbp	chf	jpy	gold	wti	grain	metal
Correlation	2,29%	0,73%	2,46%	-1,09%	-0,24%	0,77%	3,50%	2,69%
Pearson	29,00%	72,75%	24,75%	61,45%	91,00%	71,55%	10,10%	20,90%
Permutation	28,64%	73,33%	25,34%	61,32%	91,08%	72,00%	10,34%	21,13%

Moreover, if we also study the significance of the correlation level through Pearson's or the permutation test, we can see that the only asset that has a correlation that is *significantly different from zero*[2] is the Standard&Poor's 500 Index. Still, the correlation that we experience between S&P500 and Bitcoin returns is of 4,37 %, which is considerably low.

Thus, our results show that Bitcoin is fundamentally incorrelated to any of the asset that we are taking into consideration in our analysis.

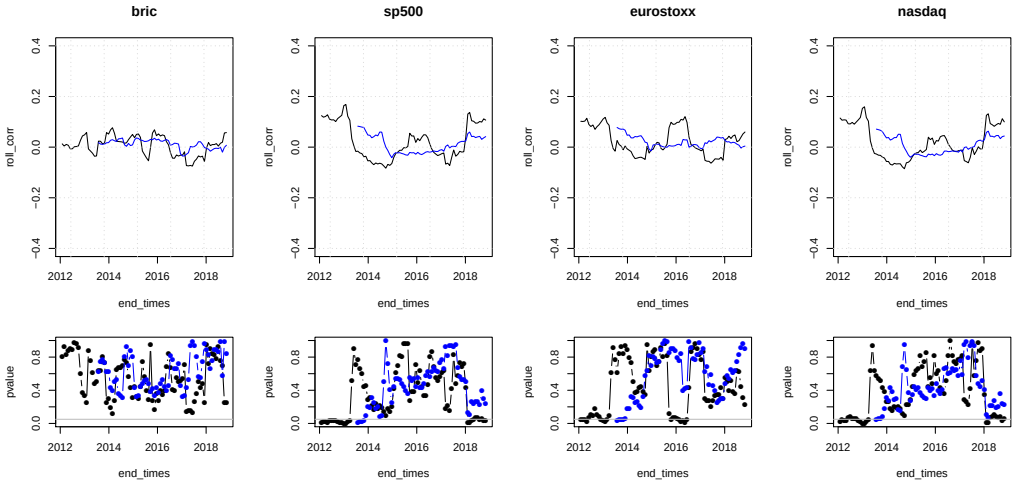
This result is what induced us to consider the possible diversification benefits of introducing Bitcoin in an investor's portfolio. We will see in Chapter 5 what great improvements in terms of increased return and decreased risk this addition brings to our reference portfolio.

2.3 Rolling Correlation

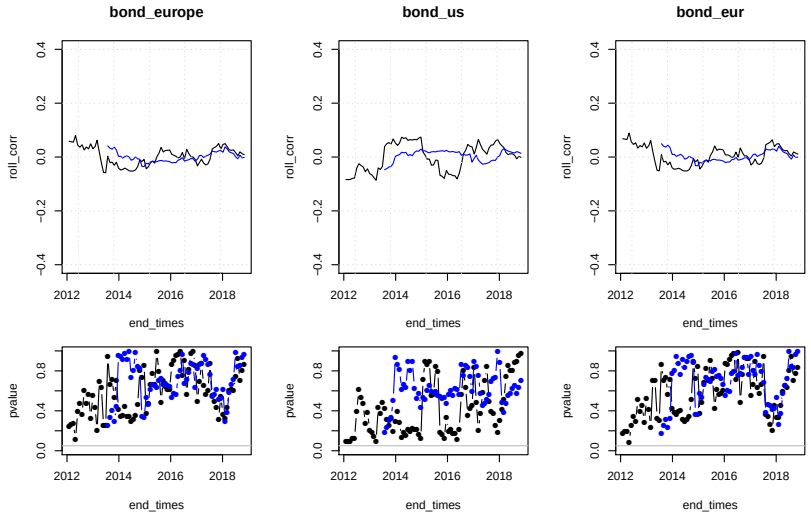
Our study so far has focused on the analysis of the dataset as a whole, with values spanning from July 2010 to early November 2018. This is clearly important if we want to obtain a general overview of the period, but it is also interesting to see how the correlations between the assets has evolved through time. Therefore, we present in Figures 2.1 and the results obtained from calculating the correlation between Bitcoin and the other assets using rolling windows of 36 and 18 months, updated monthly.

There are two graphs for each asset: in the top plots levels of the rolling correlations are represented using two different colours, blue for the 3-year

[2]Considering a confidence level of 5%.

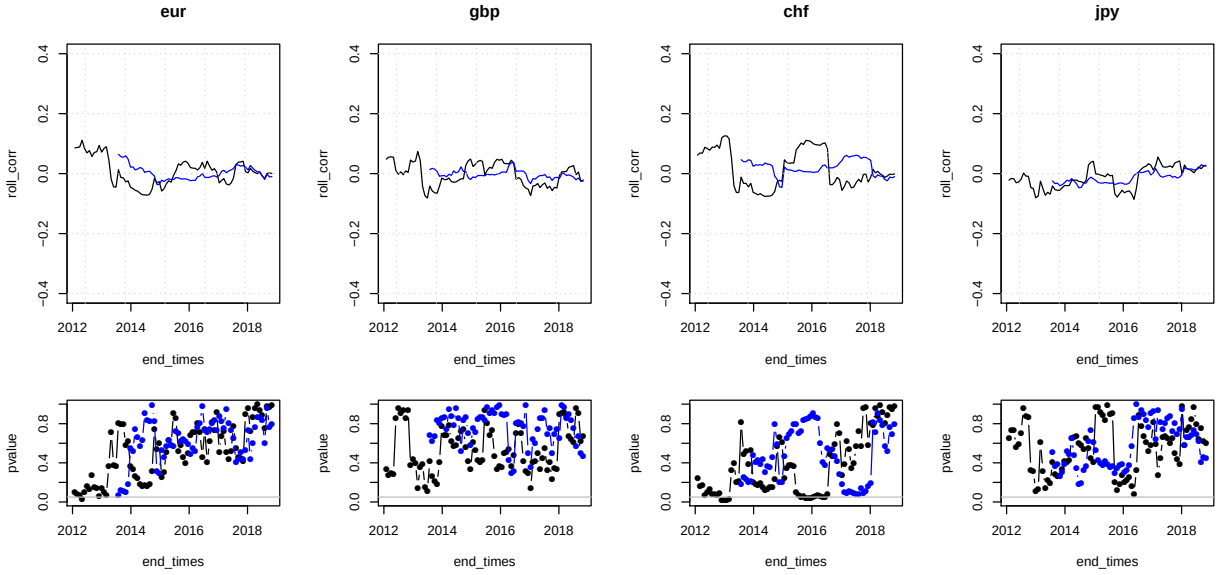


(a) Stocks

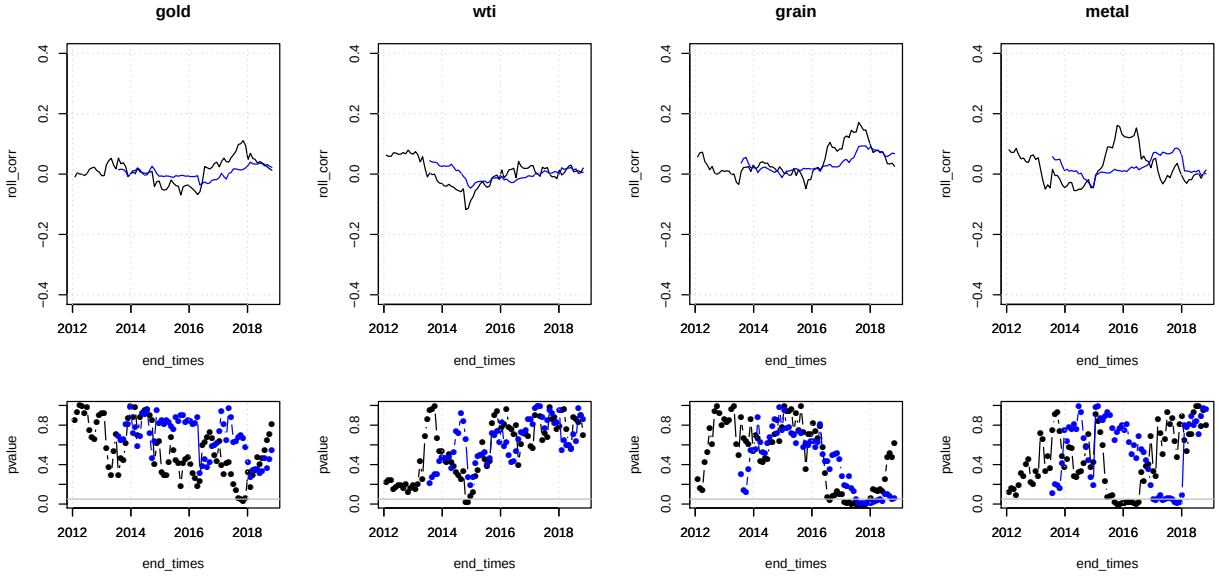


(b) Bonds

Figure 2.1: Plots of rolling correlation for the different asset classes (on top) and significance for each value (on bottom). Blue lines are the 3-year rolling correlations, while the black ones have a window of 18 months. Both computations are updated monthly.[1/2]



(a) Currency exchange



(b) Commodities

Figure 2.2: Plots of rolling correlation for the different asset classes (on top) and significance for each value (on bottom). Blue lines are the 3-year rolling correlations, while the black ones have a window of 18 months. Both computations are updated monthly.[2/2]

and black for the 18-month windows; in the bottom plots we included the significance of each rolling correlation through its p-value. The grey horizontal line represents the 5% level of significance[3].

The main conclusion we can draw from these images is that the correlation of any asset with Bitcoin is hardly ever significantly different from zero, and when it is, its absolute level is never greater than 15% unless for a small period of time.

To confirm the fact that Bitcoin is not correlated with any asset, we can also take a look at the path of the rolling correlations: there is no line that is always above zero, nor below. This indicates that there is no underlying trend, whether positive or negative, and the correlation one might happen to find is only temporary.

[3]As we explained in the previous section, to check that a sample correlation is *significantly* non zero, we compare the p-value of the test to a given level, here $1 - \alpha = 5\%$. Graphically, whenever the dots are above the grey line in Figures 2.1 or 2.2, the corresponding correlation is *not* significantly different from zero.

Chapter 3

Presentation of the Models

In this chapter we will present the stochastic frameworks in which we developed our analysis. We first introduce a *jump diffusion* (JD) model presented in 1976 by R.C. Merton: he added log-normal jumps to the simple Black&Scholes dynamics of the asset price. Then we move to the *stochastic volatility* (SV) model of Heston 1993. Heston introduced a new stochastic process that accounts for the variance of the underlying price which evolves as in B&S but with a stochastic volatility term. The last model we will present was introduced by Bates in 1996 and it is the combination of the former two: an asset dynamics which includes jumps and is driven by a stochastic volatility. All models are first introduced in the one dimensional case and then generalised to the n dimensional case which was then implemented in our code.

3.1 Preliminary Notions

In this section we will briefly present the equation of a geometric brownian motion, introduce the notion of Poisson process and present the CIR process. All of these building blocks will be required to fully understand the models to follow. For more details please refer to (Tankov and Cont, 2015).

3.1.1 Geometric Brownian Motion

The simplest continuous dynamics to describe the price of an asset is that of a geometric Brownian motion:

$$\frac{dS_t}{S_t} = \mu dt + \sigma dW_t, \tag{3.1}$$

where S_t represents the price of the asset at time t, μ is the (constant) drift and σ is the (constant) volatility. W_t is a Wiener process (or Brownian motion). This is the standard and most widespread stochastic differential equation to model asset dynamics, so we will only present those results that will be later used in our study.

Applying Ito's lemma to the previous equation, we can also explicitly express the dynamics of the log-returns $X_t = log(S_t)$, obtaining:

$$dX_t = (\mu - \frac{\sigma^2}{2})dt + \sigma dW_t. \tag{3.2}$$

This stochastic differential has a simple solution which can be computed via stochastic integrals and allows us to describe the dynamics of the log-returns at each instant t starting from $t = 0$:

$$X_t = X_0 + (\mu - \frac{\sigma^2}{2})t + \sigma W_t. \tag{3.3}$$

Thanks to (3.3) we can now express the price dynamics of the asset by inverting the relation with the returns: $S_t = e^{X_t}$. We thus obtain the solution to (3.1):

$$S_t = S_0 \, e^{(\mu - \frac{\sigma^2}{2})t + \sigma W_t}. \tag{3.4}$$

Given that $S_t = e^{X_t}$ and that X_t by its equation is a generalized brownian motion and hence we have $X_t - X_0 \sim \mathcal{N}(\mu - \frac{\sigma^2}{2}, \sigma^2)$, the resulting distribution of prices at time t as units of the initial value is distibuted as a log-normal.

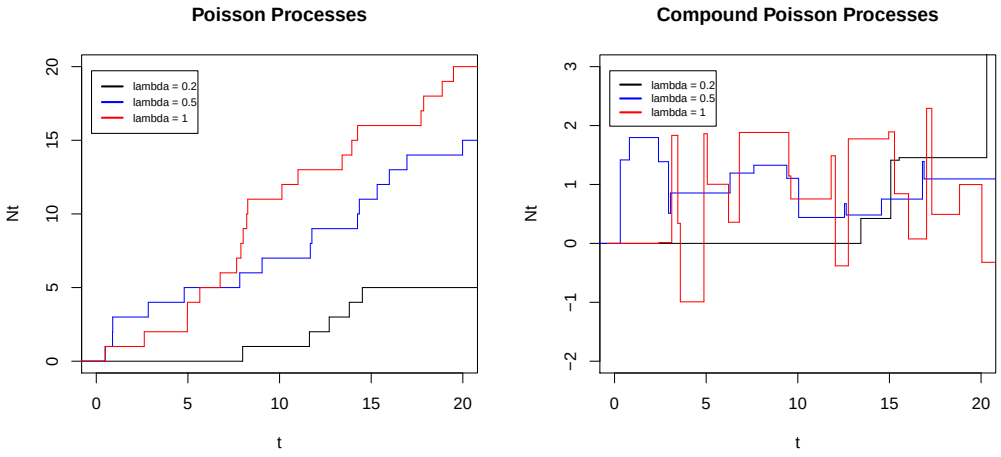
The great success of these framework comes from the simplicity of its dynamics. In particular, since the log-returns follow a Gaussian distribution, μ and σ are easy to calibrate from data and the formulas for pricing options are often explicit. As one can imagine, a simple model can only explain simple phenomena: that's why we have such a great deal of *new and improved* version of equation (3.1).

3.1.2 Poisson Process and Compound Poisson Process

Consider a sequence of *independent* exponential random variables $\{\tau_i\}_{i \geq 1}$ with parameter λ[1] and let $T_n = \sum_{i=1}^{n} \tau_i$. Then we can define the *Poisson process* N_t as

$$N_t = \sum_{n \geq 1} \mathbb{1}_{t \geq T_n}, \tag{3.5}$$

[1] An exponential random variable τ of parameter λ has a cumulative distibution function of the form: $\mathbb{P}(\tau \geq y) = e^{-\lambda y}$



(a) Poisson processes.

(b) Compound with Gaussian jumps.

Figure 3.1: Poisson processes and compound Poisson processes with different λ.

where $\mathbb{1}_{condition}$ is 1 if the *condition* is true, 0 if it is false.

N_t is thus a piece-wise constant RCLL[2] process with jumps that happen at times T_n and are all of size 1, as we can see from Figure 3.1a. An important property of Posson processes is that they have independent and stationary increments, meaning that the increment of $N_t - N_s$ (with $s \leq t$) is independent from the history of the process up to N_s and has the same law of N_{t-s}. At any time t, N_t is distributed as a Poisson of parameter λt, which means it is a discrete random variable on the integer set with

$$\mathbb{P}(N_t = n) = e^{-\lambda t} \frac{(\lambda t)^n}{n!}. \tag{3.6}$$

When working with jump diffusion processes, it is often the case that there is no explicit formula for its density, thus we usually resort to characteristic functions. The characteristic function of N_t is given by

$$\phi_{N_t}(u) = e^{\lambda t(e^{iu}-1)}. \tag{3.7}$$

The computations to get (3.7) from (3.6) are carried out in Appendix A.

For financial applications, it is of little interest to have a process with a single possible jump size. The *compound* Poisson processes are a generalization of Poisson processes where the jump sizes can have an arbitrary

[2]RCLL is shorthand for right continuous with left limit.

distribution. More precisely, consider a Poisson process N_t with parameter λ and a sequence of i.i.d[3] variables $\{Y_i\}_{i\geq 1}$ with law $f_Y(y)$. Then the process defined by

$$X_t = \sum_{n=1}^{N_t} Y_i \tag{3.8}$$

is a compound Poisson process. Examples of this kind of process are plotted in Figure 3.1b.

As before, we developed the computations to obtain an expression for the characteristic function of X_t in Appendix A. The resulting expression depends on the distribution of Y, specifically from its characteristic function $\phi_Y(u)$:

$$\phi_{X_t}(u) = e^{\lambda t(\phi_Y(u)-1)}. \tag{3.9}$$

Both in Merton's and in the Heston's models there will be a jump component driven by a compound Poisson process with Gaussian jump sizes, as we will see in the following paragraphs.

3.1.3 CIR Process

The CIR process was introduced in (Cox et al., 1985) in 1985 by Cox, Ingersoll and Ross (hence the name CIR) as a generalization of a Vasicek process (Vasicek, 1977) to model the mean reverting dynamics of interest rates. Following their notation, the differential equation for the evolution of the rate is given by:

$$dr_t = \kappa(\theta - r_t)dt + \sigma\sqrt{r_t}dW_t, \tag{3.10}$$

where we have three parameters that characterize it: θ is the *long-term value* of the rate, the asymptotic level which it tends to settle at in the long run;κ is the *mean-reversion rate*, the speed at which the rate is pulled back to the θ value; finally σ accounts for the *volatility* of the stochastic component. When $\kappa, \theta > 0$, equation (3.10) represents a first order mean-reverting autoregressive process. Moreover, as reported in (Albrecher et al., 2007), we know that the process will not hit zero if the following condition is satisfied:

$$2\kappa\theta > \sigma^2. \tag{3.11}$$

This condition is usually referred to as *Feller* condition, as stated in the referenced paper.

An example of a CIR process is shown in Figure 3.2, where the mean-reverting effect is clearly visible.

[3]i.i.d stands for independent and identically distributed.

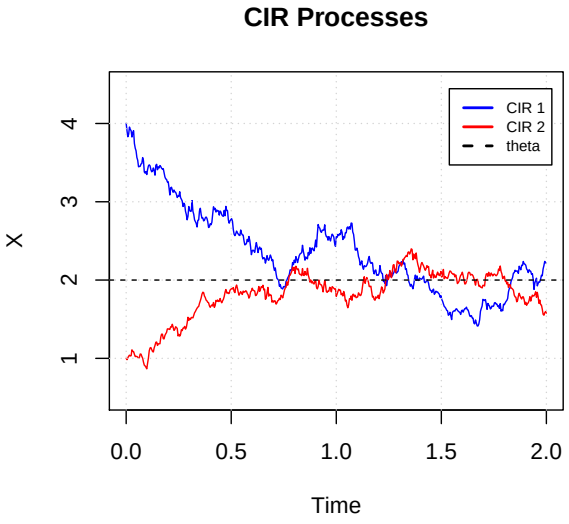


Figure 3.2: Two trajectories of the same CIR process: $dr_t = 3(2 - r_t)dt + 0.5\sqrt{r_t}dW_t$ with different starting point. We can see the mean-reverting effect that attracts both trajectories to the value $\theta = 2$.

When modelling interest rate in continuous time, having r_t hit zero is not an issue since when $r_t = 0$ equation (3.10) reduces to $dr_t = \kappa\theta dt$ which immediately brings the level back to positive values and hence the square root in the dynamics never loses meaning. Conversely, considering a *discretized* version of (3.10), as is the case in a simulation framework, one needs to pay attention on how he models the increments since a simple Euler discretization scheme may cause r_t to reach *negative* values[4] and invalidate the whole model representation.

The non negativity of the CIR process will become fundamental when we introduce Heston and Bates SV models, where the (stochastic) variance of the process driving the asset price will be model as a CIR process.

Since it will be useful later on in the paper, we also present the *stationary* distribution of r_t. Due to the mean reversion effect, r_t will approach a *gamma* distribution with density:

$$f_{r_\infty}(x) = \frac{\omega^\nu}{\Gamma(\nu)} x^{\nu-1} e^{-\omega x}, \tag{3.12}$$

[4]This model was introduced having in mind the certainty that interest rates could never be negative, hence the introduction of a square root in the dynamics. Given recent years interest rates levels, this in no more the case.

where

$$\omega = \frac{2\kappa}{\sigma^2}\ ,\ \nu = \frac{2\kappa\theta}{\sigma^2}.$$

Its *moment generating function*, which will be useful later as well, is defined as follows:

$$M_{r_\infty}(z) = \Big(\frac{\omega}{\omega - z}\Big)^{\nu}. \tag{3.13}$$

3.2 Merton Model

Let us see now how the different building blocks that we just presented can be combined together by first taking a look at the jump diffusion model by Merton.

3.2.1 Original Univariate Model

The jump diffusion model we are interested in was originally introduced in (Merton, 1976) in order to account for the leptokurtic distribution of real market returns and to model sudden falls (or rises) in prices due to the arrival of new information. The asset price dynamics S_t is modelled as a GBM to which a jump component driven by a compound Poisson process is added:

$$\frac{dS_t}{S_t} = (\mu - \lambda\mu_J)dt + \sigma dW_t + Y_t dN, \tag{3.14}$$

where μ and σ are respectively the drift and the diffusion of the continuous part, Y_t is a process modelling the intensity of the jumps and $N(t)$ is the Poisson process driving the arrival of the jumps and has parameter λ. The jump intensity Y_t is distributed as a log-normal with parameters (μ_J, σ_J^2). This means that $y_t = \log(Y_t) \sim \mathcal{N}(\mu_J, \sigma_J^2)$

We can rewrite (3.14) in terms of the log-returns $X_t = log(S_t)$ and obtain, following the computations in (Martin, 2007) and using theory from (Tankov and Cont, 2015):

$$dX_t = (\mu - \frac{\sigma^2}{2} - \lambda\mu_J)dt + \sigma dW_t + \log(Y_t), \tag{3.15}$$

that has as solution:

$$X_t = X_0 + \mu t + \sigma W_t + \sum_{k=1}^{N(t)} y_k, \tag{3.16}$$

where X_0 is the initial value of the log-returns, $y_k = \log(Y_k) = \log(Y_{t_k})$ and t_k is the time when the k^{th} Poisson shock from $N(t)$ happens. Following Merton in (Merton, 1976), we take y_k *i.i.d.* (independent and identically distributed) and Gaussian, as we already stated. Another choice for the distribution of y is given in (Kou, 2002), where an asymmetric distribution is used to account for positive and negative jump in two different ways.

It is often useful when dealing with market data that are by nature discrete, to consider a *discretized* version of (3.16) in which the values are sampled at intervals of Δt in $[0, T]$. We thus get that for $X_i = \log(\frac{S_{i+1}}{S_i})$:

$$X_i = \mu\Delta t + \sigma\sqrt{\Delta t}\, z + \sum_{k=1}^{N_{i+1}-N_i} Y_k \tag{3.17}$$

where we denote $X_i = X_{t_i}$, $N_i = N(t_i)$ and $t_i = i\Delta t$ with $i = 0 \dots N$, $t_N = N\Delta t = T$, z is distributed as a standard Gaussian $z \sim \mathcal{N}(0, 1)$.

The Poisson process $N(t)$ in (3.17) is computed at times t_{i+1} and t_i and these quantities are subtracted. Following the results we presented in Section 3.1.2, $N_{i+1} - N_i$ is distributed as a Poisson random variable N of parameter $\lambda\Delta t$. This allows us to provide an explicit formulation for the transition density of the log-returns using the theorem of total probability:

$$f_{\Delta x}(x) = \sum_{k=0}^{\infty} \mathbb{P}(N = k) f_{\Delta x|N=k}(x). \tag{3.18}$$

This equation is a good analytical result but it is not so practical for calibration purposes, as we will explain more in depth in Chapter 4.

3.2.2 Multivariate Model

Starting from the univariate model introduced in (Merton, 1976), we can develop a generalization to n assets including only idiosyncratic jumps:

$$\frac{dS_t^{(j)}}{S_t^{(j)}} = (\mu_j - \lambda_j \mu_{J_j})dt + \sigma_j dW_t^{(j)} + Y_t^{(j)} dN_t^{(j)} \tag{3.19}$$

where $\mathbf{S}_t$ are the prices of the assets, $j = 1, ..., n$ represents the asset, μ_j are the drifts, σ_j are the diffusion coefficients, $W_t^{(j)}$ are the components of an n-dimensional Wiener process $\mathbf{W}_t$ with $dW^{(i)}dW^{(j)} = \rho_{i,j}$, Y_j represent the intensities of the jumps and $y^{(j)} = \log(Y^{(j)})$ are distributed as Gaussian: $y^{(j)} \sim \mathcal{N}(\mu_{J_j}, \sigma_{J_j}^2)$. Finally, $N^{(j)}(t)$ are Poisson processes with parameters λ_j, which are independent of $\mathbf{W}_t$ and of one another.

3.3 Heston Model

The Heston model was presented in 1993 in (Heston, 1993) as a new framework to model stochastic volatility in the asset price dynamics, which allows to better fit the skewness and kurtosis of the log-return distribution and account for the changes in the volatility of the price process through time.

3.3.1 Univariate Heston Model

The Heston model belongs to the family of SV processes: generalizations of B&S model in which the volatility is no more constant but is itself stochastic.

Starting with a GBM as in the B&S framework, we obtain the dynamics of the price process simply by allowing the volatility in (3.1) to evolve over time and specifying how this evolution takes place. In particular, the dynamics of the *instantaneous* variance $V_t = \sigma_t^2$ for the Heston model is described by a CIR process :

$$\frac{dS_t}{S_t} = \mu dt + \sqrt{V_t} dW_t^S, \tag{3.20}$$

$$dV_t = \kappa(\theta - V_t)dt + \sigma_V \sqrt{V_t} dW_t^V, \tag{3.21}$$

where μ represents the drift in the asset prices, $\kappa > 0$ and $\theta > 0$ are respectively the *mean-reversion* rate and the long-run level for the V_t process, $\sigma_V > 0$ is often referred to as the *volatility of volatility* parameter, in short vol-of-vol. The two brownian motions W_t^S and W_t^V are correlated with correlation coefficient equal to ρ. It can be proven that the variance process is always non-negative if the *Feller* condition

$$2\kappa\theta \geq \sigma_V \tag{3.22}$$

is satisfied. If the same inequalities holds strictly, we have that the variance process will always be strictly positive.

Let us now consider the dynamics of the log-return $x_t = \log(S_t)$, as we did in the Merton case. Unfortunately, given the increased complexity of the model due to the SV part, an explicit formula for the density of the log-return is not available and it has to be computed from the Fourier transform of the characteristic function:

$$f_{x_t}(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \phi_{x_t}(iu) e^{iux} du. \tag{3.23}$$

as explained in the reference paper by Heston.

In (3.23) we omitted the dependence of $f_{x_t}(x)$ and $\phi_{x_t}(u)$ on model parameters and on the initial values of the log-returns x_0 and of the variance process V_0.

To derive the expression of the characteristic function $\phi_{x_t}(u)$, one has to solve a couple of *Fokker-Planck* partial differential equation as is shown in the Appendix of the reference paper by Heston (Heston, 1993). This procedure is beyond the scope of this thesis and thus we will only report the final result, which is a log-affine equation conditioned on the information available at time $t = 0$:

$$\begin{aligned}
\phi_{x_t}(u|x_0, V_0) &= \exp\{A(t,u) + B(t,u)x_0 + C(t,u)V_0\}, \\
A(t,u) &= \mu u i t + \frac{\kappa\theta}{\sigma_V^2}\left((\kappa - \rho\sigma_V u i + d)t - 2\log\left[\frac{1 - ge^{dt}}{1-g}\right]\right), \quad (3.24a)\\
B(t,u) &= iu, \quad (3.24b)\\
C(t,u) &= \frac{\kappa - \rho\sigma_V u i + d}{\sigma_V^2}\left[\frac{1-e^{dt}}{1-ge^{dt}}\right], \quad (3.24c)
\end{aligned}$$

where :

$$\begin{aligned}
d &= \sqrt{(\rho\sigma_V u i - \kappa)^2 + \sigma_V^2(ui + u^2)}, \\
g &= \frac{\kappa - \rho\sigma_V u i + d}{\kappa - \rho\sigma_V u i - d}.
\end{aligned}$$

This formulation is the one proposed by Heston in his original paper (Heston, 1993) but it is shown in (Albrecher et al., 2007) that it has numerical issues when pricing Vanilla options using Fourier methods. We will not be pricing any instrument, however we may incur in the same errors when calibrating our model through the techniques explained in Chapter 4. For this reason, we will be using the alternative formulation presented in (Albrecher et al., 2007):

$$\begin{aligned}
\phi^*_{x_t}(u|x_0, V_0) &= \exp\{A(t,u) + B(t,u)x_0 + C(t,u)V_0\}. \\
A(t,u) &= \mu u i t + \frac{\kappa\theta}{\sigma_V^2}\left((\kappa - \rho\sigma_V u i - d)t - 2\log\left[\frac{1 - g^*e^{-dt}}{1-g^*}\right]\right), \\
& \quad (3.25a)\\
B(t,u) &= iu, \quad (3.25b)\\
C(t,u) &= \frac{\kappa - \rho\sigma_V u i - d}{\sigma_V^2}\left[\frac{1-e^{-dt}}{1-g^*e^{-dt}}\right], \quad (3.25c)
\end{aligned}$$

where :

$$d = \sqrt{(\rho\sigma_V ui - \kappa)^2 + \sigma_V^2(ui + u^2)},$$
$$g^* = \frac{\kappa - \rho\sigma_V ui - d}{\kappa - \rho\sigma_V ui + d} = \frac{1}{g}.$$

The only difference is that the signs of the d terms are all flipped: the origin of the two representations for the characteristic function lies in the fact that the complex root d has two possible values and the second value is exactly minus the first value. A uniform choice between the two possibilities cannot be made over all the complex plane to obtain a continuous function due to the presence of a branch cut in the graph of $\sqrt{z}$. Selecting the sign of d as in $\phi^*_{x_t}$ and g^* avoids the discontinuity affecting our future computations.

For this reason, from now on we will be using the second formulation while leaving out the star * from our notation.

Our characteristic function still depends on the initial values x_0 and V_0 and is thus often called *conditional* characteristic function on x_0 and V_0. We can easily remove the dependence on x_0 by considering the distribution of incremental returns, namely $\Delta x_t = \log(S_t/S_0) = x_t - x_0$

Applying the definition of characteristic function:

$$\begin{aligned}
\phi_{\Delta x_t}(u|V_0) &= \mathbb{E}\big[e^{iu\Delta x_t}|V_0\big] \\
&= \mathbb{E}\big[e^{iu(x_t - x_0)}|V_0\big] \\
&= \mathbb{E}\big[e^{iux_t}|V_0\big]e^{-iux_0} \\
&= \phi_{x_t}(u)e^{-iux_0} \\
&= \exp\{A(t,u) + (B(t,u) - iu)x_0 + C(t,u)V_0\}
\end{aligned}$$

and since $B(t,u) = iu$, the second term in the exponential is equal to zero and we are left with:

$$\phi_{\Delta x_t}(u|V_0) = \exp\{A(t,u) + C(t,u)V_0\}. \tag{3.26}$$

This equation is however still dependent on the initial value of the variance process. In a simulation framework, this would not be an issue, since we can define the level of V_0 ourselves and then generate all the different scenarios. However, if we need to calibrate the model parameters from asset prices, the market data for the variance process are not available. We will address more in depth how to solve this problem later on in Chapter 4. As for now, we show that we can obtain an *unconditional* expression for the characteristic function of a Heston process by approximating the distribution of V_t with its stationary distribution.

The idea of integrating out the variance was presented in (Dragulescu and Yakovenko, 2002) and here we follow a similar approach. The computations to obtain an unconditional characteristic function are the following:

$$\begin{aligned}
\phi_{\Delta x_t}(u) &= \mathbb{E}\big[e^{iu\Delta x_t}\big] \\
&= \mathbb{E}\big[\mathbb{E}\big[e^{iu\Delta x_t}|V_0\big]\big] \\
&= \int_0^\infty \mathbb{E}\big[e^{iu(x_t-x_0)}|V_0=v\big] f_{V_0}(v)dv \\
&= \int_0^\infty \exp\{A(t,u)+C(t,u)v\} f_{V_0}(v)dv \\
&= \exp\{A(t,u)\}\int_0^\infty \exp\{C(t,u)v\} f_{V_0}(v)dv \\
&= \exp\{A(t,u)\} M_{V_0}(C(t,u))
\end{aligned}$$

where we have used the Law of Total Expectation and $M_{V_0}(z)$ indicates the *moment generating function* of V_0 as considered stationary, so that of a Gamma distribution. In particular, it will have the same expression as in (3.13).

3.3.2 Parsimonious Multi-asset Heston Model

The problem with generalizing the Heston framework to include more than one underlying is that the model becomes quickly very complex: we need 5 parameters for each asset ($\mu, \kappa, \theta, \sigma_V$ and ρ), and we can also model all types of correlation between the different stochastic drivers. In particular, we have 4 types of correlation:

- ρ^{S_i,V_i}: correlation that we already have in the unidimensional case, that models the way the price and the variance processes of the same asset are linked ;
- ρ^{S_i,S_j}: correlation between the movements in prices of two different assets ;
- ρ^{V_i,V_j}: correlation between the variance processes of two different assets. One might expect that asset of similar nature have a higher correlation both in the price and in the variance processes;
- ρ^{S_i,V_j}: correlation between the price process of an asset and the variance process of a different one.

To overcome this increase in both mathematical and computational complexity, we will use the *parsimonious* model introduced by Szimayer, Dimitroff and Lorenz in (Dimitroff et al., 2011). Their framework has two main characteristics: each single-asset sub-model forms a traditional Heston model and the parameters are composed by n single-asset Heston parameter sets and $n(n-1)/2$ *asset-asset* correlations.

The n-dimensional model hence becomes:

$$\begin{pmatrix} dS_i(t)/S_i(t) \\ dV_i(t) \end{pmatrix} = \begin{pmatrix} \mu_i \\ \kappa_i(\theta_i - V_i(t)) \end{pmatrix} dt + \begin{pmatrix} \sqrt{V_i(t)} & 0 \\ 0 & \sigma_{V_i}\sqrt{V_i(t)} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \rho_i & \sqrt{1-\rho_i^2} \end{pmatrix} \begin{pmatrix} dW^{S_i}(t) \\ dW^{V_i}(t) \end{pmatrix}, \quad (3.27)$$

where $dW^{S_i}(t)$ and $dW^{V_i}(t)$ are independent processes and we have written explicitly the dependence between the price and the variance processes.

To fully characterize the model, we need to describe how the different W^{S_i} and W^{V_i} are correlated. In accordance with what was stated earlier, the correlation structure is the following:

$$\Sigma^{(S,V)} = cor(\boldsymbol{W}^S, \boldsymbol{W}^V) = \begin{pmatrix} \Sigma^S & 0 \\ 0 & I_n \end{pmatrix}, \quad (3.28)$$

in which $\Sigma^S = cor(\boldsymbol{W}^S)$. Equation (3.28) mathematically represents what we defined as the correlation structure for our model: we only explicitly define the *asset-asset* correlations through matrix Σ^S, the dependence of every variance on other processes is carried over via Σ^S and the correlations ρ_i. More clearly, let $(\Sigma^S)_{i,j} = \rho_{i,j}$[5]:

- $dW^{S_i}(t)dW^{S_j}(t) = \rho_{i,j}dt$
- $dW^{S_i}(t)dW^{V_j}(t) = \rho_{i,j}\rho_j dt$
- $dW^{V_i}(t)dW^{V_j}(t) = \rho_i\rho_{i,j}\rho_j dt$ if $i \neq j$, dt otherwise

A detailed representation of a 2-asset *parsimonious* Heston model can be found in the reference paper (Dimitroff et al., 2011).

[5]Of course we will have $\rho_{i,j} = 1$ whenever $i = j$.

3.4 Bates Model

Bates model is a way of combining both of the characteristics of Merton (price jumps) and Heston (stochastic volatility) in a single framework. It was introduced in 1996 by David Bates, an American professor at University of Iowa.

3.4.1 Univariate Model

In his paper (Bates, 1996), Bates proposes his model as a way of capturing the leptokurtosis in the distribution of log-differences of the USD/DeutscheMark exchange rate. He suggested to improve the versatility of Heston model by including *log-normal, Poisson driven* jumps in the price process, borrowing this addition from Merton model.

Thus, we will have to deal with a total of 8 parameters: $\mu, \kappa, \theta, \sigma_V$ and ρ for the stochastic volatility part, and μ_J, σ_J and λ for the jump component.

$$\frac{dS_t}{S_t} = (\mu - \lambda\mu_J)dt + \sqrt{V_t}dW_t^S + Y_t dN, \tag{3.29}$$

$$dV_t = \kappa(\theta - V_t)dt + \sigma_V\sqrt{V_t}dW_t^V. \tag{3.30}$$

Of course, as these equations are directly obtained from (3.20) and (3.21), we have that the variance process is strictly positive whenever the Feller condition $2\kappa\theta \geq \sigma_V$ is satisfied. As in (3.14), Y_t is the jump process and is such that $y_t = \log(Y_t) \sim \mathcal{N}(\mu_J, \sigma_J^2)$, while $N(t)$ is a Poisson process with parameter λ.

Unfortunately, for the same reason as in previous section, an explicit formula for the distribution of the log-returns $x_t = log(S_t)$ is not available and thus we have to make use of their characteristic function. The silver lining however is that to obtain the expression of $\phi_{x_t}(u)$ we only have to include an extra additive term to the exponential in (3.25a) to account for

the jumps:

$$\phi^*_{x_t}(u|x_0, V_0) = \exp\{A(t,u) + B(t,u)x_0 + C(t,u)V_0 + D(t,u)\},$$

$$A(t,u) = \mu uit + \frac{\kappa\theta}{\sigma_V^2}\left((\kappa - \rho\sigma_V ui - d)t - 2\log\left[\frac{1-g^*e^{-dt}}{1-g^*}\right]\right), \tag{3.31a}$$

$$B(t,u) = iu, \tag{3.31b}$$

$$C(t,u) = \frac{\kappa - \rho\sigma_V ui - d}{\sigma_V^2}\left[\frac{1-e^{-dt}}{1-g^*e^{-dt}}\right], \tag{3.31c}$$

$$D(t,u) = -\lambda\mu_J uit + \lambda t\left[(1+\mu_J)^{ui}e^{\sigma_J^2(ui/2)(ui-1)} - 1\right], \tag{3.31d}$$

where :

$$d = \sqrt{(\rho\sigma_V ui - \kappa)^2 + \sigma_V^2(ui + u^2)},$$
$$g^* = \frac{\kappa - \rho\sigma_V ui - d}{\kappa - \rho\sigma_V ui + d} = \frac{1}{g}.$$

We use the * formulation for the same desirable numerical properties as already stated, while leaving out the * symbol from our notation from now on.

We can perform the same computations as in the Heston case to obtain the *unconditional* characteristic function of the log-returns to obtain:

$$\phi_{\Delta x_t}(u) = \exp\{A(t,u) + D(t,u)\}M_{V_0}(C(t,u)) \tag{3.32}$$

where $M_{V_0}(z)$ indicates the *moment generating function* of V_0 as considered stationary and is distributed as a Gamma distribution. In particular, it will have the same expression as in (3.13) with $\omega = 2\kappa/\sigma_V^2$ and $\nu = \frac{2\kappa\theta}{\sigma_V^2}$.

3.4.2 Parsimonious Multi-asset Bates Model

We generalize the single-asset Bates model to a multi-asset model in the same way that we did with the Heston: we only model the asset-asset correlation ρ^{S_i,S_j} for each pair of asset.

The model definition is similar to (3.27), with the addition of the jump part for the price processes:

$$\begin{pmatrix} dS_i(t)/S_i(t) \\ dV_i(t) \end{pmatrix} = \begin{pmatrix} \mu_i - \lambda_i \mu_{J_i} \\ \kappa_i(\theta_i - V_i(t)) \end{pmatrix} dt + \begin{pmatrix} \sqrt{V_i(t)} & 0 \\ 0 & \sigma_{V_i}\sqrt{V_i(t)} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \rho_i & \sqrt{1-\rho_i^2} \end{pmatrix} \begin{pmatrix} dW^{S_i}(t) \\ dW^{V_i}(t) \end{pmatrix} + \begin{pmatrix} Y_i(t) dN_t^i \\ 0 \end{pmatrix}. \quad (3.33)$$

$Y_i(t)$ are the jump intensities for each asset, such that $\log Y_i(t) \sim \mathcal{N}(\mu_{J_i}, \sigma_{J_i})$ and $N_i(t)$ are the Poisson processes that drive the arrival of the jumps with parameters λ_i. We have explicitly expressed the correlation between the two Brownian motions for each asset in the matrix that multiplies the stochastic differential, so that $W^{S_i}(t)$ and $W^{V_i}(t)$ are independent from each other for all i.

The only correlations that are left to specify are the ones that exist between W^{S_i} and W^{S_j}. Following the same notation as in the previous section, the complete correlation structure is the following:

$$\Sigma^{(S,V)} = cor(\boldsymbol{W}^S, \boldsymbol{W}^V) = \begin{pmatrix} \Sigma^S & 0 \\ 0 & I_n \end{pmatrix}. \quad (3.34)$$

Chapter 4

Calibration of the Models

In this chapter we will explain how the different models were calibrated and what difficulties were overcome. Empirical results are included for each section. In particular, we will calibrate the parameters under the physical measure though maximum likelihood estimation since we only have data for the time-series of the asset prices.

4.1 Maximum Likelihood Calibration

The calibration method that we implemented is the *maximum likelihood* approach, which is a statistical method to obtain the parameters of a target distribution family that best approximate the unknown distribution of the observed data.

Let us call $X = \{x_1, x_2, ..., x_N\}$ the set of available observations and let $\psi = \{\alpha_1, \alpha_2, ...\}$ the set of parameters of the distribution $\mathcal{D}$ that we want to calibrate. Moreover, define $f_{\mathcal{D}}(x; \psi)$ to be the probability density function (pdf) of distribution $\mathcal{D}$ with parameters ψ computed at $x \in \mathbb{D}$, where $\mathbb{D}$ is the domain of the density function.

Our aim is to find the best set of parameters ψ such that we can describe X as samples taken from $\mathcal{D}$:

$$x_i \sim \mathcal{D}(\psi),\ i = 1, \ldots, N. \tag{4.1}$$

The estimation of the parameter set ψ can then be obtained by:

$$\hat{\psi} = \arg\max_{\psi \in \Psi}\ \mathcal{L}(\psi|X), \tag{4.2}$$

where $\mathcal{L}(\psi|X)$ is the likelihood function for distribution $\mathcal{D}(\psi)$ given the observed data X.

In the case that the data in X are i.i.d. we have that the likelihood function can be computed as the product of the probability density function computed at each observation x_i:

$$\mathcal{L}(\psi|X) = \prod_{i=1}^{N} f_{\mathcal{D}}(x_i;\psi). \tag{4.3}$$

In practice, it is often common to consider the *log-likelihood* function $\ell(\psi|X) = \log \mathcal{L}(\psi|X)$. The natural logarithm is a monotonic function so the maximum of the log-likelihood function is achieved in the same point as the basic likelihood function.[1] Taking the logarithm of (4.4) also simplifies the expression in that the product now becomes a sum:

$$\ell(\psi|X) = \mathcal{L}(\psi|X) = \sum_{i=1}^{N} \log f_{\mathcal{D}}(x_i;\psi). \tag{4.4}$$

The issues that arise from the definition of a maximum likelihood estimator are mainly two. The first problem is that to use equation (4.4) we need to know the pdf of the distribution: this might not always be the case with complicated models that only have an explicit expression for their characteristic function. This is what happens for Heston and Bates model, and, as we will see, we are going to need to invert the chf via Fourier inversion and obtain the pdf numerically.

The second issue is that we need to solve a maximization problem in order to obtain an estimate for the parameters of the distribution: it is well known how optimization routines might perform well for some special circumstance and badly under other, especially when the number of parameters increases. The trade-off is, as usual, between fast computations and robustness of the optimization. To solve this, we will opt for a combination of global and local optimizers.

4.2 Calibration of Merton Model

Let us start by considering how we calibrated the jump diffusion model by Merton. We will first present how to obtain the parameters for a single asset model and then move to a multi-asset one.

[1] By how the likelihood function is defined, it can attain only positive values so that the logarithm of $\mathcal{L}(\psi|X)$ is always well defined.

4.2.1 Single Asset Merton Calibration

As we have already introduced, the calibration method that we use is the maximum likelihood approach.

We are going to calibrate the values of the 5 parameters $\psi = \{\mu, \sigma, \mu_J, \sigma_J, \lambda\}$ of the jump diffusion model based on the observations of the log-returns $\Delta x_i = \log \frac{S_i}{S_{i-1}}$ for each time interval Δt. In our analysis, since we are using *daily* log-returns, $\Delta t = 1/255$.

Considering daily log-returns allows us to assume that the different samples of Δx are independent and identically distributed according to the distribution of log-returns in Merton model given in (3.18). For ease of reading, we present it again here:

$$f_{\Delta x}(x;\psi) = \sum_{k=0}^{\infty} \mathbb{P}(N=k) f_{\Delta x|N=k}(x;\psi). \tag{4.5}$$

The formula represents an infinite mixture of Gaussian distributions, due to the infinite possible realization of the Poisson variable that accounts for the arrival of jumps. This fact has a great downside in terms of maximum likelihood estimation. As discussed in (Honoré, 1998), the infinite Gaussian mixture causes the problem of maximizing the (log-)likelihood to be unbounded and thus intractable.

To solve this issue we can introduce a first order approximation, as it has been proposed in (Ball and Torous, 1983). Since we are considering a small time interval Δt, only the first terms of the series in (4.5) become significant. In particular, we have that the probabilities to have $k = 0, 1, 2$ jumps in a single time step, as presented in (3.6), are:

$$\mathbb{P}(N=0) = e^{-\lambda\Delta t}, \tag{4.6a}$$
$$\mathbb{P}(N=1) = \lambda\Delta t\, e^{-\lambda\Delta t}, \tag{4.6b}$$
$$\mathbb{P}(N=2) = \frac{(\lambda\Delta t)^2}{2}\, e^{-\lambda\Delta t}, \tag{4.6c}$$

since $N \sim Pois(\lambda\Delta t)$, which is the Poisson process counting jumps that happen in a Δt interval. Considering $\lambda\Delta t$ as small, we can approximate to the first order $e^{-\lambda\Delta t} = 1 - \lambda\Delta t + o((\lambda\Delta t)^2)$. We thus obtain that:

$$\mathbb{P}(N=0) = 1 - \lambda\Delta t, \tag{4.7a}$$
$$\mathbb{P}(N=1) = \lambda\Delta t\,(1 - \lambda\Delta t) = \lambda\Delta t + o((\lambda\Delta t)^2), \tag{4.7b}$$
$$\mathbb{P}(N=2) = \frac{(\lambda\Delta t)^2}{2}\,(1 - \lambda\Delta t) = o((\lambda\Delta t)^2). \tag{4.7c}$$

The result is that the only relevant terms in (4.5) are the ones for $k = 0, 1$. The formula for the transition density thus becomes:

$$f_{\Delta x}(x;\psi) = \mathbb{P}(N=0) f_{\Delta X|N=0}(x;\psi) + \mathbb{P}(N=1) f_{\Delta x|N=1}(x;\psi). \tag{4.8}$$

We can then write this equation explicitly by considering the different distributions of the log-returns in the case of no jumps and a single jump:

$$f_{\Delta x}(x;\psi) = (1-\lambda\Delta t)\ f_{\mathcal{N}}\Big(x;\widetilde{\mu}\,\Delta t, \sigma^2\Delta t\Big) + (\lambda\Delta t)\ f_{\mathcal{N}}(x;\widetilde{\mu}\,\Delta t + \theta, \sigma^2\Delta t + \delta^2) \tag{4.9}$$

where $f_{\mathcal{N}}(x;\mu,\sigma^2)$ indicates the pdf of a Gaussian with parameters $\mathcal{N}(\mu,\sigma^2)$ compute at x. We used $\widetilde{\mu} = \mu - \sigma^2/2 - \lambda\mu_J$ as a simpler notation to indicate the drift term in the return dynamics.

The distribution of the sample log-returns is thus given by equation (4.9) and we can plug it into (4.3) to obtain the log-likelihood function that we are going to maximize:

$$\ell(\psi|\Delta x) = \sum_{i=1}^{N} \log f_{\Delta x}(\Delta x_i;\psi). \tag{4.10}$$

One can then proceed to maximize equation (4.10) to obtain the optimal parameter set ψ:

$$\hat{\psi} = \arg\max_{\psi\in\Psi}\ \ell(\psi|\Delta x), \tag{4.11}$$

$$\Psi = \{(\mu,\sigma,\mu_J,\sigma_J,\lambda) \in \mathbb{R}^5 \mid \sigma,\sigma_J,\lambda > 0\}. \tag{4.12}$$

4.2.2 Multi-asset Merton Calibration

We now know how to calibrate the Merton parameters when the underlying asset is one. In the case of multiple assets, we decided to adapt (Dimitroff et al., 2011) both for the model definition and the calibration procedure. In the referenced paper, the authors present a *parsimonious approach* to the extension of Heston Model to a multi-asset framework. Since this is the approach that we will follow for the multivariate Heston calibration and for Bates as well, we decided to also perform in the same way the calibration of the parameters for Merton. Using a common approach will allow us to better compare the results, especially in terms of the correlation structure that is what we are ultimately interested in.

The first step in the calibration algorithm is to obtain the parameters of all the single asset models that we are studying. The general way to do this was explained in the previous section.

The next step involves computing the correlation matrix between the different Brownian motions that drive each single-asset model. The way this computation is carried out in (Dimitroff et al., 2011) is by asset pairs: we consider each possible couple of assets and try to obtain the model correlation that best approximates the observed correlation. Let us see how this is achieved more in detail in the case of only two assets.

The problem we are trying to solve is, given the observed sample correlation ρ_{obs} between two assets' returns, find the best model correlation ρ_{model} that solves $\bar{\rho}(\rho_{model}) = \rho_{obs}$. $\bar{\rho}$ is the expectation of the correlation that we find when computing the empirical correlation of the returns simulated using ρ_{model}.

More precisely, we have to simulate N_{sim} different realisations of the 2-asset model in a given time period $[0, T]$ with the same Δt that we have in the observed data, using ρ_{model} as the correlation between the two driving Brownian motions. We then proceed to compute the correlation of the simulated log-returns ρ_{scen} in each scenario and then we take the average of this values. What we obtain is $\bar{\rho} = \mathbb{E}[\rho_{scen}]$.

In order to simulate a two-asset Merton model, we used the following discretization for the log-returns:

$$x_i^{(1)} = x_{i-1}^{(1)} + (\mu_1 - \sigma_1^2/2 - \lambda_1 \mu_{J_1})\Delta t + \sigma_1 \sqrt{\Delta t}\, z_1 + N_1 y_1, \tag{4.13a}$$

$$x_i^{(2)} = x_{i-1}^{(2)} + (\mu_2 - \sigma_2^2/2 - \lambda_2 \mu_{J_2})\Delta t + \sigma_2 \sqrt{\Delta t}\, z_2 + N_2 y_2, \tag{4.13b}$$

where:

- $i = 1, \ldots, N_{step}$ represents the i-th time-step in our simulation: $x_i = x_{t_i}$ and $N_{step} = T/\Delta t$ so that $t_i = i\Delta t$. The value of the initial return x_{t_0} does not affects the final correlation so we can simply set it to zero.

- z_1 and z_2 are the first and second component of a Gaussian vector $\mathbf{z} = (z_1, z_2)^T$ that is distributed as $\mathbf{z} \sim \mathcal{N}_2(\mathbf{0}, Corr)$. $Corr$ is the 2×2 matrix composed of ones in the main diagonal and ρ_{model} in the remaining two spots. More simply, $\mathbf{z}$ is a two dimensional Gaussian with zero mean with covariance matrix equal to the correlation matrix of the model.

- N_1 and N_2 are in general the realizations of two Poisson random variables with parameters $\lambda_1 \Delta t$ and $\lambda_2 \Delta t$ respectively. Given our first order approximation in the density and considering small Δt, N_1 and N_2 are actually two Bernoulli with probability of success (i.e. a jump happens) equal to $\lambda_1 \Delta t$ and $\lambda_2 \Delta t$.

- Finally, y_1 and y_2 are the jump intensities and are realisations of Gaussian variables with parameters $\mathcal{N}(\mu_{J_1}, \sigma_{J_1}^2)$ and $\mathcal{N}(\mu_{J_2}, \sigma_{J_2}^2)$.

Through (4.13) we can thus simulate $x_i^{(1,k)}$ and $x_i^{(2,k)}$ for $k = 1, ..., N_{sim}$ and compute for each scenario the correlation observed from the simulated returns $\rho_{scen}^{(k)} = corr(x_i^{(1,k)}, x_i^{(2,k)})$. From these values we can easily compute $\bar{\rho}$:

$$\bar{\rho} = \mathbb{E}[\rho_{scen}] = \frac{1}{N_{sim}} \sum_{k=1}^{N_{sim}} \rho_{scen}^{(k)}. \tag{4.14}$$

We have that the value of the *expected model correlation* $\bar{\rho}$ is implicitly a function of the model correlation ρ_{model}. The equation we need to solve is then represented by:

$$\bar{\rho}(\rho_{model}) = \rho_{obs}, \tag{4.15}$$

as we have already stated earlier in this section.

In the case of an n-asset model, we have to repeat the previous passages for all possible $n(n-1)/2$ different pairs of assets. We then can build a $n \times n$ matrix M that has ones in the main diagonal and in each element $(l, m), l \neq m$ stores the corresponding model correlation $\rho_{model}^{(l,m)}$. Since we are calibrating each value individually, this matrix may not be a well defined correlation matrix: it may happen that M is not positive definite. The last step to obtain a proper correlation matrix for our n-asset model is to perform some form of *regularization* on M that transforms it into a well defined correlation matrix: positive definite, with elements in the range $[-1, 1]$ and ones in the main diagonal.

Through empirical study performed in the reference paper (Dimitroff et al., 2011), it turns out that the best algorithm between the one proposed by the authors is the regularization by Jäckel, that we briefly present here.

Regularization algorithm by Jäckel:

Input Model correlation matrix M (not necessarily positive definite but with ones on the main diagonal).

1. Perform an eigenvalue decomposition on $M = S\Lambda S^T$, where $\Lambda = diag(\lambda_l)$ and λ_l are the eigenvalues of matrix M.

2. Define $\Lambda' = diag(\lambda_l'$ with $\lambda_l' = \max(\lambda_l, 0))$ as the diagonal matrix that contains the positive part of each eigenvalue.

3. Create the diagonal matrix $T = diag(t_l)$ where $t_l = \left(\sum_m (S_{l,m})^2 \lambda_l'\right)^{-1}$.

4. Define $B = \sqrt{T}S\sqrt{\Lambda'}$.

5. Set $\widehat{M} = BB^T$.

Output Positive definite correlation matrix $\widehat{M}$.

Hence, $\widehat{M}$ is the model correlation matrix that best represents the observed correlation values and that forms the structure of dependence between the Brownian motions that drive each asset in a multivariate Merton model.

4.3 Calibration of Heston Model

Calibrating the Heston model on time-series data presents an additional difficulty since we only have data on the price process, while the model also takes into consideration the dynamics of the variance as a stochastic process.

The problem of deducing the parameter of a *non-observable* process from the observable states of a system is called *filtering* and is in general a complicated numerical procedure. There are nonetheless some studies in the literature on the subject of calibrating the Heston model through filtering, for example (Papi et al., 2015).

In our analysis we preferred to keep the same approach to the calibration for all the three models. We will thus proceed to explain how we can perform a maximum likelihood calibration in the case of stochastic volatility and, in the next section, of both jumps and stochastic volatility.

4.3.1 Single Asset Heston Calibration

The first issue that we have to overcome is that an explicit formula for the probability density function of the log-returns is not available and we have to resort to performing a Fourier inversion on the characteristic function, as we already showed in equation (3.23)

This allows us to have a probability density function (pdf) for the log-return $x_t = \log S_t$ but in order to be able to perform a maximum likelihood calibration in the same way that we did in the previous section, we need to have the distribution for $\Delta x_t = \log(S_{t+\Delta t}/S_t)$ and make sure that they are i.i.d. so that we can apply (4.3).

In order to obtain the pdf for the incremental log-returns Δx_t, we first need the distribution of $\Delta x_{[0,t]} = \log(S_t/S_0)$. We can simply proceed as we explained in Chapter 3 in the section about Heston model to get the characteristic function for $\Delta x_{[0,t]}$ conditioned on the values of V_0:

$$\phi_{\Delta x_{[0,t]}}(u|V_0) = \exp\{A(t,u) + C(t,u)V_0\}. \tag{4.16}$$

The full expressions for $A(t,u)$ and $C(t,u)$ are given in (3.24a). Notice that we omitted the dependence on the model parameters $\psi = \{\mu, \kappa, \theta, \sigma_V, \rho\}$ for ease of notation.

We are now left with the distribution of $\Delta x_{[0,t]}$ as a function of V_0: this represents an issue since moving from $\Delta x_{[0,t]}$ to $\Delta x_t = \Delta x_{[t,t+\Delta t]}$ we have a dependence on the value of V_t which is not available. Moreover, even if data for V_t was indeed available, considering different time-steps t_i for Δx_t, the data samples of Δx_{t_i} would not be identically distributed as they depend on levels of the variance V_{t_i} which are different.

Hence the need for an *unconditional* characteristic function to get rid of the dependence on initial variance. We followed the approach that was introduced in (Dragulescu and Yakovenko, 2002) of integrating out the dependence on V_0. The computations to obtain it were presented in Chapter 3. The unconditional chf for the Heston model has the following expression:

$$\phi_{\Delta x_t}(u) = \exp\{A(\Delta t, u)\} M_\Gamma(C(\Delta t, u)) \tag{4.17}$$

with M_Γ as the moment generating function of a Gamma distribution:

$$\begin{aligned} M_\Gamma(z) &= \left(\frac{\omega}{\omega - z}\right)^\nu, \\ \omega &= \frac{2\kappa}{\sigma_V^2}, \\ \nu &= \frac{2\kappa\theta}{\sigma_V^2}. \end{aligned}$$

We now have a specific expression for the unconditional characteristic function of Δx_t and we can obtain the corresponding density by Fourier inversion:

$$f_{\Delta x_t}(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \phi_{\Delta x_t}(iu) e^{iux} du \tag{4.19}$$

Another advantage of using the uncoditional characteristic function (chf) is that since all the observed log-returns $\Delta x_i = \Delta x_{t_i}$ are considered to be sampled from the same distribution, we can numerically perform the inversion in (4.19) using the Fast Fourier Transform (FFT). This allows for a great increase in the speed of the numerical computations since we can obtain the value of the pdf for all different Δx_i with a single inversion and one

interpolation. The main ideas behind the FFT algorithm are explained in Appendix C.

Now that we have the density function of Δx with parameters $\psi = \{\mu, \kappa, \theta, \sigma_V, \rho\}$ we are able to compute the log-likelihood function by substituting (4.19) in (4.4).

In order to perform the maximization of the log-lokelihood function to calibrate the parameters, we have to make sure that the Feller condition (3.22) is satisfied.

The MLE procedure will thus need to be performed including an additional constraint in the optimization given by Feller condition:

$$\widehat{\psi} = \arg\max_{\psi \in \Psi} \ell(\psi | \Delta x), \tag{4.20}$$

$$\Psi = \{(\mu, \kappa, \theta, \sigma_V, \rho) \in \mathbb{R}^5 \mid \kappa, \theta, \sigma_V > 0, \rho \in [-1, 1], 2\kappa\theta \geq \sigma_V\}. \tag{4.21}$$

4.3.2 Multi-asset Heston Calibration

As stated earlier in this paper, the approach to extending the Heston model to N assets that we are taking into consideration is the parsimonious multi-asset model introduced by Dimitroff et al. in (Dimitroff et al., 2011). It's called *parsimonious* since, as we explained in chapter 3, we are only modelling the asset-asset correlations.

The procedure to calibrating an N asset model is the same that we presented in the multi-asset Merton section and consists of three parts.

Firstly, we have to calibrate all the single asset parameters by themselves. Then we can proceed to compute each asset-asset correlation by solving (4.15) through the simulation of the realization of the log-returns in different scenarios. Finally, we perform Jäckel regularization algorithm to make sure that the final output matrix is indeed a correlation matrix.

The main steps were all presented in section 4.2.2, here we will only give the details for the simulation of a two-asset Heston model.

Since we now have two processes for each asset, we have to simulate the path for both, in each time-step. The discretized dynamics of the log-returns and the variance for a single asset are the following:

$$x_i = x_{i-1} + (\mu - \frac{V_{i-1}}{2})\Delta t + \sqrt{V_{i-1}\Delta t}\, z_x, \tag{4.22a}$$

$$V_i = V_{i-1} + \kappa(\theta - V_{i-1})\Delta t + \sigma_V \sqrt{V_{i-1}\Delta t}\, z_V, \tag{4.22b}$$

with z_x and z_V that are the two components of a Gaussian vector with mean zero and correlation ρ.[2]

The issue that arises when moving from a continuous dynamics to a discrete one as in (4.22) is that the variance V_i might assume negative values even if the Feller condition is verified. This happens because the second and third term in the equation of the variance process may be so negative that combined with the first term V_{i-1} the result is less than zero. Having $V_i < 0$ causes the next update at $i+1$ to be undefined since we would have to compute the square root of a negative value in a real framework.

To solve this issue, a number of possible solutions have been proposed and are collected in (Lord et al., 2010). Among those we have: *absorption*, which consists in setting $V_i = 0$ when it is negative, *reflection*, taking the absolute value $V_i = |V_i|$, and finally *full truncation*, which is the solution proposed by the authors in (Lord et al., 2010) and the one we implement. Other efficient simulation schemes that solve this problem can be found in (Andersen, 2007). It is obtained modifying (4.22) as follows:

$$x_i = x_{i-1} + (\mu - \frac{V_{i-1}^+}{2})\Delta t + \sqrt{V_{i-1}^+ \Delta t}\, z_x, \tag{4.23a}$$

$$V_i = V_{i-1} + \kappa(\theta - V_{i-1}^+)\Delta t + \sigma_V \sqrt{V_{i-1}^+ \Delta t}\, z_V. \tag{4.23b}$$

The notation $y^+ = \max(y, 0)$ indicates the positive part of y.

Equations in (4.23) represent the single asset simulation scheme but in order to calibrate the asset-asset correlations we need to simulate the time-series for a pair of assets.

The updating computations only amount to repeating the single scheme twice:

$$x_i^{(1)} = x_{i-1}^{(1)} + (\mu_1 - \frac{\left(V_{i-1}^{(1)}\right)^+}{2})\Delta t + \sqrt{\left(V_{i-1}^{(1)}\right)^+ \Delta t}\, z_{x^{(1)}}, \tag{4.24a}$$

$$V_i^{(1)} = V_{i-1}^{(1)} + \kappa_1(\theta_1 - \left(V_{i-1}^{(1)}\right)^+)\Delta t + \sigma_{V^{(1)}} \sqrt{\left(V_{i-1}^{(1)}\right)^+ \Delta t}\, z_{V^{(1)}}, \tag{4.24b}$$

$$x_i^{(2)} = x_{i-1}^{(2)} + (\mu_2 - \frac{\left(V_{i-1}^{(2)}\right)^+}{2})\Delta t + \sqrt{\left(V_{i-1}^{(2)}\right)^+ \Delta t}\, z_{x^{(2)}}, \tag{4.24c}$$

$$V_i^{(2)} = V_{i-1}^{(2)} + \kappa_2(\theta_2 - \left(V_{i-1}^{(2)}\right)^+)\Delta t + \sigma_{V^{(2)}} \sqrt{\left(V_{i-1}^{(2)}\right)^+ \Delta t}\, z_{V^{(2)}}. \tag{4.24d}$$

[2]In a more extensive form: $\mathbf{z} = \begin{pmatrix} z_x \\ z_V \end{pmatrix} \sim \mathcal{N}_2\Big(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \Big)$

The important difference with the single asset case is the correlation structure. In order to simulate the random vector $\mathbf{z} = (z_{x^{(1)}}, z_{V^{(1)}}, z_{x^{(2)}}, z_{V^{(2)}})^T$ we have to extract it from a 4-dimensional Gaussian with zero mean and covariance given by Σ where:

$$\Sigma = \begin{pmatrix} 1 & \rho_1 & \rho_{1,2} & \rho_{1,2}\rho_2 \\ \rho_1 & 1 & \rho_1\rho_{1,2} & \rho_1\rho_{1,2}\rho_2 \\ \rho_{1,2} & \rho_1\rho_{1,2} & 1 & \rho_2 \\ \rho_{1,2}\rho_2 & \rho_1\rho_{1,2}\rho_2 & \rho_2 & 1 \end{pmatrix}. \tag{4.25}$$

We could also use the Cholesky decomposition of $\Sigma = LL^T$ in order to obtain $\mathbf{z} = L\tilde{\mathbf{z}}$ from a standard 4-dimensional Gaussian vector[3] $\tilde{\mathbf{z}}$. In this case matrix L would be lower triangular and defined as:

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \rho_1 & \sqrt{1-\rho_1^2} & 0 & 0 \\ \rho_{1,2} & 0 & \sqrt{1-\rho_{1,2}^2} & 0 \\ \rho_{1,2}\rho_2 & 0 & \rho_2\sqrt{1-\rho_{1,2}^2} & \sqrt{1-\rho_2^2} \end{pmatrix}. \tag{4.26}$$

From here on, the procedure is the same as for Merton: we need to solve (4.15) for every possible asset pair to obtain the model correlation M.

The final step is to apply Jäckel regularization algorithm to obtain a valid correlation matrix $\widehat{M}$.

4.4 Calibration of Bates Model

The algorithm to compute the value for the parameters of the Bates model is the same as the two previous ones, so in the following sections we will only focus on the main differences in the functions and equations.

4.4.1 Single Asset Bates Calibration

As it was the case for the Heston model, there is no explicit formula for the density of the log-returns Δx_t. Moreover, the characteristic function is dependent on the initial value of the variance process, unless we use the approximation that we presented in Chapter 3 to obtain (3.32).

[3]The standard Gaussian vector of dimension n is a normal random vector with mean vector zero and the identity matrix as covariance matrix: $\mathcal{N}_n(\mathbf{0}, \mathbf{I}_n)$. This means that all the components are independent from one another.

We can then go ahead and compute the corresponding density function by Fourier inversion: we can perform this computation using the FFT algorithm that allows for an increase in the performance speed. Arguing in the same way as in previous sections, by having the pdf of Δx we can compute it for all the observations of Δx_i with a single inversion.

Finally, to calibrate the model we need to perform the maximization of the log-likelihood function, making sure the parameters satisfy the Feller condition:

$$\widehat{\psi} = \arg\max_{\psi \in \Psi} \ \ell(\psi|\Delta x), \tag{4.27}$$

$$\Psi = \{(\mu, \kappa, \theta, \sigma_V, \rho, \mu_J, \sigma_J, \lambda) \in \mathbb{R}^8 \mid \kappa, \theta, \sigma_V, \sigma_J, \lambda > 0, \rho \in [-1, 1], 2\kappa\theta \geq \sigma_V\}. \tag{4.28}$$

4.4.2 Multi-asset Bates Calibration

Again, the generalization to the multivariate case we present is an adaptation from (Dimitroff et al., 2011). The model is be the same as the one in section 4.3.2 with the addition of the jump component.

The calibration algorithm is composed of the three parts concerning the calibration of all the single asset models, the computation of the asset-asset correlations by simulating the processes, and the regularization of the model correlation matrix in order to obtain an acceptable correlation matrix, with ones in the main diagonal, all other elements in the $[-1, 1]$ range and that is positive definite.

The simulation for a single asset is performed through the following scheme, implementing the *full truncation* approach to solving the issue with the variance being negative:

$$x_i = x_{i-1} + (\mu - \frac{V_{i-1}^+}{2} - \lambda\mu_J)\Delta t + \sqrt{V_{i-1}^+ \Delta t}\, z_x + Ny, \tag{4.29a}$$

$$V_i = V_{i-1} + \kappa(\theta - V_{i-1}^+)\Delta t + \sigma_V \sqrt{V_{i-1}^+ \Delta t}\, z_V, \tag{4.29b}$$

where y is the jump intensity and is distributed as a Gaussian $\mathcal{N}(\mu_J, \sigma_J^2)$ and N is the Poisson variable with parameter $\lambda\Delta t$ which, given our approximation for small Δt, is equivalent to a Bernoulli with probability of success equal to $\lambda\Delta t$. z_x and z_V are the two components of a Gaussian vector with mean zero and correlation ρ.

The same scheme applied to a 2-asset model is the following:

$$x_i^{(1)} = x_{i-1}^{(1)} + (\mu_1 - \frac{\left(V_{i-1}^{(1)}\right)^+}{2}\lambda_1\mu_{J_1})\Delta t + \sqrt{\left(V_{i-1}^{(1)}\right)^+\Delta t}\, z_{x^{(1)}} + N_1 y_1, \tag{4.30a}$$

$$V_i^{(1)} = V_{i-1}^{(1)} + \kappa_1(\theta_1 - \left(V_{i-1}^{(1)}\right)^+)\Delta t + \sigma_{V^{(1)}}\sqrt{\left(V_{i-1}^{(1)}\right)^+\Delta t}\, z_{V^{(1)}}, \tag{4.30b}$$

$$x_i^{(2)} = x_{i-1}^{(2)} + (\mu_2 - \frac{\left(V_{i-1}^{(2)}\right)^+}{2}\lambda_2\mu_{J_2})\Delta t + \sqrt{\left(V_{i-1}^{(2)}\right)^+\Delta t}\, z_{x^{(2)}} + N_2 y_2, \tag{4.30c}$$

$$V_i^{(2)} = V_{i-1}^{(2)} + \kappa_2(\theta_2 - \left(V_{i-1}^{(2)}\right)^+)\Delta t + \sigma_{V^{(2)}}\sqrt{\left(V_{i-1}^{(2)}\right)^+\Delta t}\, z_{V^{(2)}}, \tag{4.30d}$$

where the parameters and variables have the same meaning as in the single asset framework. The random vector $\mathbf{z} = (z_{x^{(1)}}, z_{V^{(1)}}, z_{x^{(2)}}, z_{V^{(2)}})^T$ is a 4-dimensional Gaussian with zero mean and covariance Σ expressed in (4.25). We can also consider the Cholesky decomposition $\Sigma = LL^T$ to sample the vector $\mathbf{z}$ from the independent Gaussian vector $\tilde{\mathbf{z}}$ by doing $\mathbf{z} = L\tilde{\mathbf{z}}$.

After we have calibrated each asset-asset correlation by solving (4.15), we need to perform Jäckel regularization algorithm to obtain a valid model correlation matrix.

4.5 Empirical results

We now present the specifics of how the calibration procedure was implemented in our study, including the numerical results that it yielded. All the code to perform the computations was developed in R for greater portability across platforms. As explained in the Introduction, the dataset is composed by a total of 2163 daily observations of the prices of 16 assets for the time period ranging from July 19, 2010 to November 2, 2018.

4.5.1 Merton Calibration Results

One of the core aims of this study is to compare the values of the empirical correlation of Bitcoin with other main assets as presented in Chapter 2 with the same correlations computed using more sophisticated models, in our case Merton, Heston and Bates.

The calibration algorithm to compute the parameters of the Merton model was implemented step by step as presented in the previous sections.

In order to perform the single-asset calibration we need to specify the boundaries for the domain of the parameter set $\psi = \{\mu, \sigma, \mu_J, \sigma_J, \lambda\}$ and which optimizer we are going to use. In our analysis, the optimization is carried out by first taking advantage of a global optimizer and then performing a local maximization[4]

The global optimizer we adopted is implemented in the R package `DEoptim` and performs *Differential Evolution* optimization: the DE algorithm starts with a population of points randomly sampled from the search space and at each iterations creates new candidate solutions and only keeps the one that best improve the objective function. More information on the package can be found in (Ardia et al., 2016) and an example of its usage in calibrating a JD process in (Ardia et al., 2011). The general DE algorithm was first introduced in (Price et al., 2006).

The local optimizer we used is instead implemented in the function `nlminb` and it performs a quasi-Newton optimization with box constraints. The function is included in the `stats` package which is native in most versions of R.

The maximization of the log-likelihood function is notoriously a difficult task since its shape has many different maxima and most of them have similar values for the function. It is thus necessary to restrict the search domain to values that make both numerical and financial sense. For the calibration of the parameters in the Merton model, we used the boundaries reported in the following table:

	μ	σ	μ_J	σ_J	λ
lower bound	-5	10^{-5}	$2 * q_{0.1\%}(r_{sample})$	10^{-4}	10^{-5}
upper bound	5	2	$q_{0.5\%}(r_{sample})$	0.1	10

We choose to use a different boundary on each asset for the jump mean μ_J because we aim to consider as jumps only true shocks: that is why we base our boundaries on the quantiles $q_\alpha(r_{sample})$ of the daily log-return distribution of our sampled data. Moreover, we limit the volatility of the jumps to 0.1 in order the have a jump distribution that is more concentrated around the value of μ_J. This way, we also avoid having extremely high values of λ,

[4]Since most optimization packages implemented in R usually perform minimization as default, we coded the minimization of the *negative* log-likelihood instead of the maximization of the (positive) log-lkelihood.

Table 4.1: Upper and lower bounds for the parameter μ_J.

	lower bound	upper bound
btc	-0.9316	-0.2541
bric	-0.1085	-0.0342
sp500	-0.0911	-0.0324
eurostoxx	-0.1350	-0.0448
nasdaq	-0.1020	-0.0363
bond_europe	-0.0413	-0.0148
bond_us	-0.0163	-0.0058
bond_eur	-0.0380	-0.0160
eur	-0.0372	-0.0166
gbp	-0.0402	-0.0148
chf	-0.0445	-0.0163
jpy	-0.0576	-0.0196
gold	-0.1108	-0.0343
wti	-0.1776	-0.0608
grain	-0.1143	-0.0462
metal	-0.1076	-0.0344

which instead is the case in (Honoré, 1998). For our assets, the values of the boundaries for μ_J are reported in Table 4.1.

The results for all the single asset calibration are presented in Table 4.2. We can see that for some assets the calibration procedure yields values of μ_J and σ_J that are on the boundaries: this is because the model is trying to explain most of the daily variations as jumps and setting boundaries avoids this phenomenon.

The resulting distribution for the log-returns obtained from the calibration is graphically compared to the empirical distribution of the samples for each asset in Appendix B.

The next step is to perform the calibration of the model correlation matrix by solving equation (4.15) numerically, for each possible asset pair. To do so, we used the function `uniroot` which is included in the `stats` package. `uniroot` implements a bisection method to find the zero of a univariate function $f(x)$ in a given interval $[a, b]$. The only thing we have to make sure of is that $f(a)f(b) < 0$ otherwise the algorithm will return an error. With highly positive or negative values of the observed correlation, there might not be a corresponding value in $[-1, 1]$ for the model correlation that solves the equation. For this reason, whenever we encounter such an issue we set ρ_{model} to be 1 or -1, depending on the sign of ρ_{obs}.

Table 4.2: Numerical results for the calibration of the parameters of the Merton model for each asset.

	μ	σ	μ_J	σ_J	λ
btc	1.791	0.9591	-0.315	0.1000	2.152
bric	0.001	0.1628	-0.034	0.1000	3.106
sp500	0.109	0.1245	-0.032	0.0928	5.196
eurostoxx	0.025	0.1985	-0.045	0.1000	5.577
nasdaq	0.146	0.1455	-0.036	0.0865	4.935
bond_europe	0.026	0.0784	-0.015	0.0015	3.027
bond_us	0.024	0.0300	-0.006	0.0001	3.778
bond_eur	0.025	0.0836	-0.016	0.0012	3.085
eur	-0.010	0.0819	-0.017	0.0001	2.766
gbp	-0.013	0.0783	-0.040	0.1000	0.308
chf	-0.018	0.0877	-0.016	0.1000	3.055
jpy	-0.027	0.0838	-0.020	0.0659	3.094
gold	0.009	0.140	-0.034	0.1000	4.100
wti	0.023	0.299	-0.061	0.0002	3.953
grain	-0.041	0.206	-0.046	0.0017	2.505
metal	-0.013	0.176	-0.034	0.1000	2.215

Finally, the different values of ρ_{model} are stored in a matrix for which we perform Jäckel's regularization. The resulting correlation structure is shown in Table 4.3, with the usual colour notation.

The entire procedure of calibration for the Merton model is fairly fast and only takes about 10 minutes when executed in our computer, a MacBook Pro with a 2,6 GHz Intel Core i5 processor and an 8 GB 1600 MHz DDR3 RAM.

4.5.2 Heston Calibration Results

The implementation of the calibration of Heston model for each asset is very similar to the one that we presented in the previous section for Merton. We use two optimizer, one global and one local.

The main difference in the maximization of the log-likelihood function is that now we have an extra constraint given by the Feller condition: the same will be true for the calibration of the parameters in Bates model.

To include this constraint in the global optimizer we used the package `DEoptimR` and the function `JDEoptim`: this performs the same differential evolution optimization but allows for the specification of extra constraints. The relative manual can be found in (Conceicao, 2016).

For the local optimization we still used the `nlminb` function: to include

Table 4.3: Resulting model correlation matrix for Merton. The values are in percentages and the colour goes from red for $\rho = 100\%$, to white for $\rho = 0\%$ and to blue for $\rho = -100\%$.

	btc	bric	sp500	eurostoxx	nasdaq	bond_europe	bond_us	bond_eur	eur	gbp	chf	jpy	gold	wti	grain	metal
btc	100.0	2.7	7.7	6.3	6.4	1.9	-2.4	2.7	2.7	1.2	4.4	-1.8	-0.6	1.1	4.2	4.2
bric	2.7	100.0	84.9	83.1	92.8	30.5	-24.8	31.1	31.0	39.8	19.0	-33.9	24.7	47.5	23.4	79.7
sp500	7.7	84.9	100.0	88.3	94.1	21.1	-62.5	24.3	32.2	40.1	6.0	-54.6	0.6	64.5	28.1	72.2
eurostoxx	6.3	83.1	88.3	100.0	83.6	59.2	-40.6	62.1	67.9	65.5	45.1	-23.2	26.4	47.1	22.3	75.6
nasdaq	6.4	92.8	94.1	83.6	100.0	18.4	-53.7	20.9	25.4	33.7	-0.2	-55.0	-1.4	51.0	24.5	65.6
bond_europe	1.9	30.5	21.1	59.2	18.4	100.0	20.1	98.6	96.2	72.5	92.1	58.5	69.9	17.6	13.3	36.7
bond_us	-2.4	-24.8	-62.5	-40.6	-53.7	20.1	100.0	15.0	-0.7	-6.5	28.6	61.9	38.0	-25.2	-8.0	-24.4
bond_eur	2.7	31.1	24.3	62.1	20.9	98.6	15.0	100.0	98.4	64.5	90.0	54.2	66.9	17.5	13.0	39.3
eur	2.7	31.0	32.2	67.9	25.4	96.2	-0.7	98.4	100.0	68.1	87.9	46.3	61.2	20.9	15.0	42.3
gbp	1.2	39.8	40.1	65.5	33.7	72.5	-6.5	64.5	68.1	100.0	69.1	23.4	43.6	27.2	14.6	38.5
chf	4.4	19.0	6.0	45.1	-0.2	92.1	28.6	90.0	87.9	69.1	100.0	74.8	86.7	12.9	14.4	44.9
jpy	-1.8	-33.9	-54.6	-23.2	-55.0	58.5	61.9	54.2	46.3	23.4	74.8	100.0	79.1	-12.2	3.0	-4.6
gold	-0.6	24.7	0.6	26.4	-1.4	69.9	38.0	66.9	61.2	43.6	86.7	79.1	100.0	24.2	23.0	56.3
wti	1.1	47.5	64.5	47.1	51.0	17.6	-25.2	17.5	20.9	27.2	12.9	-12.2	24.2	100.0	20.6	49.8
grain	4.2	23.4	28.1	22.3	24.5	13.3	-8.0	13.0	15.0	14.6	14.4	3.0	23.0	20.6	100.0	28.1
metal	4.2	79.7	72.2	75.6	65.6	36.7	-24.4	39.3	42.3	38.5	44.9	-4.6	56.3	49.8	28.1	100.0

the constraint we added a penalty to the negative log-likelihood function that would set its value to a high negative number (e.g. -10^8).

The box constraints for the Heston parameters $\psi = \{\mu, \kappa, \theta, \sigma_V, \rho\}$ are given in the following table:

	μ	κ	θ	σ_V	ρ
lower bound	$\mu_{sample} - 0.05$	10^{-3}	10^{-3}	10^{-5}	10^{-4} or -1
upper bound	$\mu_{sample} + 0.05$	2	3	2	1 or -10^{-4}

By running the calibration a few times we noticed that the resulting values of μ coming from the calibration procedure with the same bounds as in Merton were nonsensical both in magnitude and in sign. Thus, we resort to impose that the boundary for the drift is the expected return obtained from the sample with the possibility to move up or down of 0.05.

For the parameter ρ that models the asymmetry of the return distribution we restrict the boundary to positive values when the skewness of the return distribution is positive and to negative values if the skewness is negative. The calibrated parameters for each asset are shown in Table 4.4.

The resulting density is graphically compared to the empirical distribution of the log-returns in Appendix B.

Next we perform the calibration of the ρ_{model} for each pair of assets, by using the same algorithm coded in `uniroot` that we presented in the previous

Table 4.4: Numerical results for the calibration of the parameters of the Heston model for each asset.

	μ	κ	θ	σ_V	ρ
btc	1.377	0.677	0.738	0.9998	-0.0002
bric	-0.005	2.000	0.030	0.2588	-1.0000
sp500	0.100	0.460	0.017	0.1267	-0.0040
eurostoxx	0.014	0.964	0.047	0.2977	-0.3063
nasdaq	0.153	1.286	0.025	0.2554	-0.3667
bond_europe	0.028	0.621	0.007	0.0562	0.0031
bond_us	0.031	1.726	0.001	0.0378	-0.9606
bond_eur	0.030	0.568	0.008	0.0638	0.0059
eur	-0.004	1.498	0.007	0.1077	0.0522
gbp	-0.023	1.291	0.006	0.0901	-0.0175
chf	0.011	0.001	0.008	0.0034	0.0001
jpy	-0.030	0.985	0.008	0.1045	-0.9874
gold	0.008	0.364	0.023	0.1234	-1.0000
wti	-0.009	0.293	0.103	0.2460	0.0001
grain	-0.069	2.000	0.046	0.3348	0.7014
metal	-0.032	0.001	0.035	0.0079	0.0001

section. Again, in case that the observed sample correlation is too high in absolute value, we set ρ_{model} equal to 1 or -1.

Lastly, we perform the usual regularization to obtain a valid model correlation matrix. The final result is shown in Table 4.5.

The increase in the complexity in terms of the formulation of the model generates a great increase in the computation time for the calibration. Each time we have to compute the (negative) log-likelihood function we need to invert the characteristic function: even when taking advantage of the FFT algorithm there still is a great increase with respect to the calibration time in Merton's case. The total time amounts to about three hour, with a single asset calibration taking up to fifteen minutes. aggiungere caratteristiche computer

4.5.3 Bates Calibration Results

The procedure to calibrate the parameters $\psi = \{\mu, \kappa, \theta, \sigma_V, \rho, \mu_J, \sigma_J, \lambda\}$ for Bates model we proceed exactly in the same way as we did for Heston. We first perform a global optimization using `JDEoptim` and then a local one through `nlminb` adding a penalty to the neg-log-likelihood function.

Table 4.5: Resulting correlation matrix for Heston. The values are percentages and the colour scheme is the usual.

	btc	bric	sp500	eurostoxx	nasdaq	bond_europe	bond_us	bond_eur	eur	gbp	chf	jpy	gold	wti	grain	metal
btc	100.0	1.8	6.3	5.9	5.4	1.9	-2.2	2.6	2.8	0.9	3.2	-1.8	-0.5	0.9	4.5	3.3
bric	1.8	100.0	59.5	67.3	58.0	22.0	-17.0	22.0	22.0	26.8	9.2	-19.7	15.5	33.8	17.1	46.3
sp500	6.3	59.5	100.0	78.3	99.0	15.8	-40.5	17.8	21.5	25.3	0.4	-27.9	-0.9	41.8	18.3	39.8
eurostoxx	5.9	67.3	78.3	100.0	71.2	48.5	-32.2	51.0	54.9	48.3	24.9	-20.4	11.3	37.6	17.5	51.5
nasdaq	5.4	58.0	99.0	71.2	100.0	12.9	-37.0	14.4	17.6	22.2	-2.0	-26.5	-1.5	34.7	16.9	36.9
bond_europe	1.9	22.0	15.8	48.5	12.9	100.0	20.7	98.9	96.1	67.0	63.7	43.4	49.8	15.6	12.4	27.9
bond_us	-2.2	-17.0	-40.5	-32.2	-37.0	20.7	100.0	15.4	-0.6	-5.7	15.1	43.0	24.4	-22.6	-7.3	-17.6
bond_eur	2.6	22.0	17.8	51.0	14.4	98.9	15.4	100.0	97.9	57.7	61.6	40.2	46.9	15.3	11.9	28.8
eur	2.8	22.0	21.5	54.9	17.6	96.1	-0.6	97.9	100.0	60.8	62.6	34.2	41.7	19.1	14.1	31.5
gbp	0.9	26.8	25.3	48.3	22.2	67.0	-5.7	57.7	60.8	100.0	39.3	16.3	28.7	23.9	12.7	27.4
chf	3.2	9.2	0.4	24.9	-2.0	63.7	15.1	61.6	62.6	39.3	100.0	40.3	43.5	6.9	7.8	21.7
jpy	-1.8	-19.7	-27.9	-20.4	-26.5	43.4	43.0	40.2	34.2	16.3	40.3	100.0	48.4	-7.4	2.3	-3.3
gold	-0.5	15.5	-0.9	11.3	-1.5	49.8	24.4	46.9	41.7	28.7	43.5	48.4	100.0	17.1	15.9	35.7
wti	0.9	33.8	41.8	37.6	34.7	15.6	-22.6	15.3	19.1	23.9	6.9	-7.4	17.1	100.0	19.1	37.4
grain	4.5	17.1	18.3	17.5	16.9	12.4	-7.3	11.9	14.1	12.7	7.8	2.3	15.9	19.1	100.0	21.4
metal	3.3	46.3	39.8	51.5	36.9	27.9	-17.6	28.8	31.5	27.4	21.7	-3.3	35.7	37.4	21.4	100.0

The boundaries for each parameter are the combination of the intervals that we use for Merton and Heston and are reported in the following table:

	μ	κ	θ	σ_V	ρ	μ_J	σ_J	λ
lower bound	$\mu_{sample} - 0.05$	10^{-3}	10^{-3}	10^{-5}	10^{-4} or -1	$q_{\alpha_{low}}(r_{sample})$	10^{-4}	10^{-5}
upper bound	$\mu_{sample} - 0.05$	2	3	2	1 or -10^{-4}	$q_{\alpha_{up}}(r_{sample})$	0.1	10

We also introduce conditional bounds on the skewness for the μ_J variable: we set $\alpha_{low} = 0.1\%$ and $\alpha_{up} = 0.5\%$ if the skewness is negative and $\alpha_{low} = 99.5\%$ and $\alpha_{up} = 99.9\%$. The parameters for each single asset in the Bates case are shown in Table 4.6.

The increased number of parameters causes an increase complexity in the shape of the log-likelihood function and hence it is harder for the optimizers to find the exact global optimum.

As for the calibration of the model correlation, we maintain the same approach of using `uniroot` to solve equation (4.15) and setting ρ_{model} to 1 or -1 in case the solver cannot come up with a solution. The regularization using Jäckel algorithm represents as usual the final step to obtain the full model correlation. The resulting matrix is reported in Table 4.7.

The time it takes to calibrate the entire Bates model is of about 5 hours: the same argumentation holds here about the need to invert the characteristic function, but the additional computational time with respect to Heston is due to the increasing number of parameters that expand the dimensionality of the calibration problem.

Table 4.6: Numerical results for the calibration of the parameters of the Bates model for each asset.

	μ	κ	θ	σ_V	ρ	μ_J	σ_J	λ
btc	1.380	0.879	0.6703	1.0855	-0.0036	-0.2388	0.09973	0.928
bric	-0.005	2.000	0.0290	0.2443	-0.9381	-0.0600	0.00001	0.327
sp500	0.100	1.636	0.0168	0.2344	-0.0019	-0.0340	0.00020	1.342
eurostoxx	0.002	1.303	0.0456	0.3311	-0.1511	-0.0506	0.00072	0.812
nasdaq	0.135	1.289	0.0242	0.2498	-0.0001	-0.0368	0.00012	1.478
bond_europe	0.031	2.000	0.0066	0.1005	0.0001	0.0197	0.00001	0.196
bond_us	0.023	1.951	0.0010	0.0424	-0.9963	-0.0134	0.01021	0.001
bond_eur	0.028	1.389	0.0076	0.0960	0.0018	0.0186	0.01057	0.006
eur	-0.004	0.318	0.0071	0.0477	0.1526	0.0250	0.00001	0.163
gbp	-0.021	2.000	0.0060	0.0938	-1.0000	-0.0149	0.00119	0.805
chf	0.005	0.932	0.0079	0.0925	0.7125	0.0215	0.00497	0.731
jpy	-0.034	2.000	0.0075	0.1360	-0.0001	-0.0250	0.00001	1.026
gold	0.008	2.000	0.0220	0.2668	-0.0467	-0.0343	0.00089	1.166
wti	-0.009	0.278	0.1035	0.2400	0.0001	0.0654	0.00347	0.000
grain	-0.069	0.235	0.0443	0.1080	1.0000	0.0628	0.00001	0.571
metal	-0.032	0.001	0.0349	0.0079	0.0001	0.0369	0.00044	0.000

Table 4.7: Resulting correlation matrix for Bates. Values are percentages and the colour scheme is the same used for the other correlation matrices.

	btc	bric	sp500	eurostoxx	nasdaq	bond_europe	bond_us	bond_eur	eur	gbp	chf	jpy	gold	wti	grain	metal
btc	100.0	1.5	5.2	4.7	4.2	1.4	-2.3	2.2	3.1	0.5	2.5	-1.2	-0.1	1.0	3.3	2.8
bric	1.5	100.0	51.8	61.8	50.7	20.6	-16.3	21.1	21.6	25.9	8.2	-17.3	13.8	32.3	16.4	44.8
sp500	5.2	51.8	100.0	67.2	99.6	13.8	-35.9	15.4	19.5	22.0	0.2	-23.0	-0.5	36.9	16.4	35.8
eurostoxx	4.7	61.8	67.2	100.0	61.9	44.4	-29.7	47.4	52.4	45.3	22.4	-17.5	9.8	34.9	16.9	49.1
nasdaq	4.2	50.7	99.6	61.9	100.0	11.4	-34.1	12.7	16.5	20.0	-1.9	-23.2	-0.7	31.7	15.4	34.1
bond_europe	1.4	20.6	13.8	44.4	11.4	100.0	20.1	98.9	94.2	63.5	60.6	40.6	44.0	15.0	12.0	27.0
bond_us	-2.3	-16.3	-35.9	-29.7	-34.1	20.1	100.0	15.1	-0.7	-5.1	14.3	39.5	22.3	-22.2	-7.3	-17.6
bond_eur	2.2	21.1	15.4	47.4	12.7	98.9	15.1	100.0	96.6	55.3	58.9	37.7	41.6	14.9	11.8	28.0
eur	3.1	21.6	19.5	52.4	16.5	94.2	-0.7	96.6	100.0	59.7	60.8	32.3	38.6	19.0	13.9	31.6
gbp	0.5	25.9	22.0	45.3	20.0	63.5	-5.1	55.3	59.7	100.0	36.1	14.7	26.1	22.9	12.9	26.9
chf	2.5	8.2	0.2	22.4	-1.9	60.6	14.3	58.9	60.8	36.1	100.0	37.3	37.9	6.7	7.9	21.0
jpy	-1.2	-17.3	-23.0	-17.5	-23.2	40.6	39.5	37.7	32.3	14.7	37.3	100.0	40.5	-7.1	2.5	-3.3
gold	-0.1	13.8	-0.5	9.8	-0.7	44.0	22.3	41.6	38.6	26.1	37.9	40.5	100.0	15.7	14.4	32.6
wti	1.0	32.3	36.9	34.9	31.7	15.0	-22.2	14.9	19.0	22.9	6.7	-7.1	15.7	100.0	18.9	36.6
grain	3.3	16.4	16.4	16.9	15.4	12.0	-7.3	11.8	13.9	12.9	7.9	2.5	14.4	18.9	100.0	21.5
metal	2.8	44.8	35.8	49.1	34.1	27.0	-17.6	28.0	31.6	26.9	21.0	-3.3	32.6	36.6	21.5	100.0

Chapter 5

Optimal Portfolio Allocation

In this chapter we will explore what the optimal allocation is for our portfolio of assets. We will study the *efficient frontier* using two different risk measures, volatility and CVaR or expected shortfall. In all our analyses, we will compare the effects that including Bitcoin in our portfolio has on the optimal allocation.

5.1 Markowitz Mean-Variance Portfolio Optimization

Modern Portfolio Theory (MPT) is a mathematical framework for creating a portfolio of asset by maximizing the expected return for a given level of risk or by minizing the risk while maintaining the same expected gain. Before the article (Markowitz, 1952) by Harry Markowitz in 1952, the concept of *diversification* (the old warning *not to put all your eggs in one basket*) was only driven by the experience of how markets behaved. Moreover, investors used to base their decisions on expected return alone and thus when given a choice between two assets with different expected returns, they would simply put all their money on the top performing one.

With his article, that would later grant him the Nobel Prize in Economics, Markowitz introduced a more rigorous and mathematically sound framework to assembly a portfolio of assets. His key insight is that an asset's return and risk should not be assessed by itself, but rather by how it affects the overall portfolio risk and return. To do so, the *variance* is used as a proxy for risk. Hence the name *mean-variance* analysis that is often used as a substitute for MPT.

Let us introduce the assumptions underlying the MPT:

1. Investors are *risk averse*: they will always choose the less risky asset, when two assets offer the same return. At the same time, an investor wanting a higher return has to be willing to accept a higher risk. This equally holds for portfolios as a whole: given two portfolio with different risk profiles, he will choose the less risky in case of same return and the most remunerating in case of same risk.

2. Portfolio return is the weighted sum of the single assets' returns: in general $\mathbb{E}[R_{ptf}] = \sum_{i=1}^{N} w_i \mathbb{E}[R_i]$.

3. Portfolio variance is a function of both the assets variances and their correlations: $V_{ptf} = \sum_{i=1}^{N} w_i \sigma_i^2 + \sum_{i=1}^{N} \sum_{j \neq i, j=1}^{N} w_i w_j \rho_{i,j} \sigma_i \sigma_j$.

Items 2 and 3 above can be more compactly stated using matrix notation, which will come in handy later on in our analysis:

$$r_{ptf}(\mathbf{w}) = \mathbf{w}^T \mathbf{r}, \tag{5.1}$$

$$\sigma_{ptf}^2(\mathbf{w}) = \mathbf{w}^T \Sigma \mathbf{w}, \tag{5.2}$$

where we have the weights vector $\mathbf{w} = [w_1, w_2, ..., w_N]^T$, $\mathbf{r} = [r_1, r_2, ..., r_N]^T$, using the shorthand $r_i = \mathbb{E}[R_i]$ and finally Σ is the $N \times N$ covariance matrix of the assets.

We can now state the *optimization problem* involving the minimization of the portfolio risk for a specified expected portfolio return in terms of the variable we have just introduced.

$$\begin{aligned}
&\min_{\mathbf{w} \in \mathbb{R}^N} && \sigma_{ptf}^2(\mathbf{w}) && \text{(5.3a)}\\
&\text{subject to} && \mathbf{e}^T \mathbf{w} = 1, && \text{(5.3b)}\\
& && \mathbf{r}^T \mathbf{w} = r_{target}, && \text{(5.3c)}\\
& && w_i \geq 0, \text{for } i = 1 \dots N, && \text{(5.3d)}
\end{aligned}$$

where $\mathbf{e}$ indicates a vector of ones and the first constraint makes sure that the sum of the weights always equals to one. This is to represent a portfolio in which all the money available is allocated in the assets we are taking into consideration. The second constraint ensures that the portfolio allocation $\mathbf{w}$ produces the target expected return r_{target}. Finally, the last constraint is in fact optional and is only used to exclude the possibility to go short on any asset.

The optimization problem in (5.3) has a quadratic objective function given by (5.2) and only has linear constraints[1]. Thanks to this property, the optimization can be carried out numerically by any of the quadratic/linear optimizers that are available for most programming languages.

As we are going to explain in the following sections, we will be mainly focusing on the case where there is no short selling, as indeed so far there are no instruments on the market that allow an investor to go short on Bitcoin and our analysis shows that the main diversification advantage comes from including Bitcoin in our portfolio. Allowing short-selling improves our diversification capability only very slightly.

5.2 Markowitz Efficient Frontier

It is interesting to study the set of optimal allocation as a whole, rather than simply focus on one target return and minimizing the portfolio risk. To do so, we can consider a set of target returns and compute for each of them the respective minimum variance. We thus get a set of pairs (σ^2, r) that represents the best allocation in terms of the minimum risk.

We can thus plot those pairs on an X-Y graph and obtain a curve, the *portfolio frontier*, that intrinsically represents our portfolio of N assets. As a usual practice in finance, we will be plotting on the X-axis the volatility σ instead of the variance σ^2.

In Figure 5.1 we can see what the portfolio frontier looks like for our portfolio of assets. It is interesting to notice how the curve divides the plane in two region: the area to the left of the line includes all those pairs (σ, r) that are not attainable with our assets, since they have a volatility that is too low for that level of expected return. On the other hand, the region to the right of the portfolio frontier is made of all the pairs that are possible to obtain with a specific allocation $\mathbf{w}$ but that will never be chosen by an investor: moving to the left on the same level of return we eventually reach a point on the frontier. The portfolio represented by this point will dominate the one we started from in terms of risk, so it will always be a better choice.

We can proceed with the same argument arguing in terms of best return for a given level of risk: we can thus introduce the *efficient* frontier. For every level of volatility that has two corresponding points on the portfolio frontier, only the one with the higher expected return will be chosen by an investor in our reference framework: hence only the top half of the curve (from the vertex and up) will form the *efficient portfolio frontier*.

[1]The last positivity constraint can be easily expressed in matrix form by writing $\mathbf{I}_N\mathbf{w} \geq \mathbf{0}_N$ where $\mathbf{I}_N$ is the identity matrix of order N and $\mathbf{0}_N$ is N-dimensional vector of zeros.

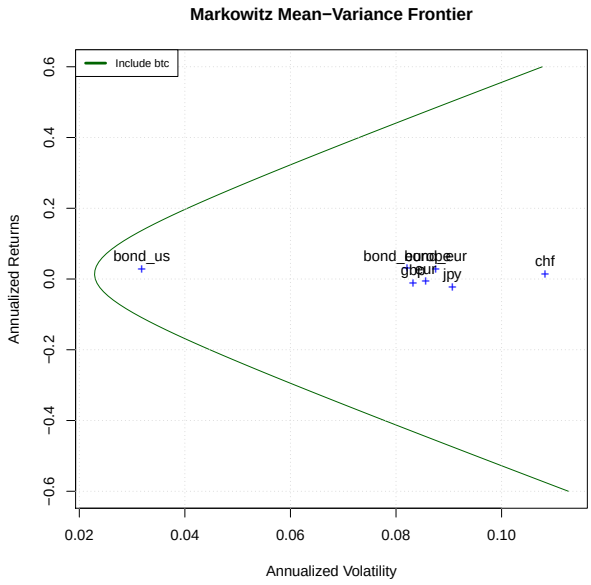


Figure 5.1: The Markowitz Mean-Variance frontier obtained from our portfolio of assets, that includes Bitcoin and allows short-selling.

As a key result, let us keep in mind that any point in the volatility-expected return plane *dominates* all the other portfolio allocations that are represented by points situated below and to the right of it. Conversely, that same point will be dominated by all other allocations above and to the left of it.

Another mathematical perk of considering only the efficient top section is that it renders the frontier a *bijective* function. Hence, for each (attainable) level of risk there is one and only one corresponding expected portfolio return, and vice-versa.

5.2.1 Efficient Frontier with and without Bitcoin

Let us now study how including Bitcoin in our portfolio of assets can help increase the diversification and obtain a higher (expected) return with a lower risk.

In Figure 5.2 we have plotted the efficient Markowitz frontier in all possible cases: including and excluding Bitcoin while allowing short-selling, and the same two curves when short-selling is not allowed.

Comparing the two green lines, we can clearly see that the exclusion of short-selling does not penalize our efficient allocation by much. The same

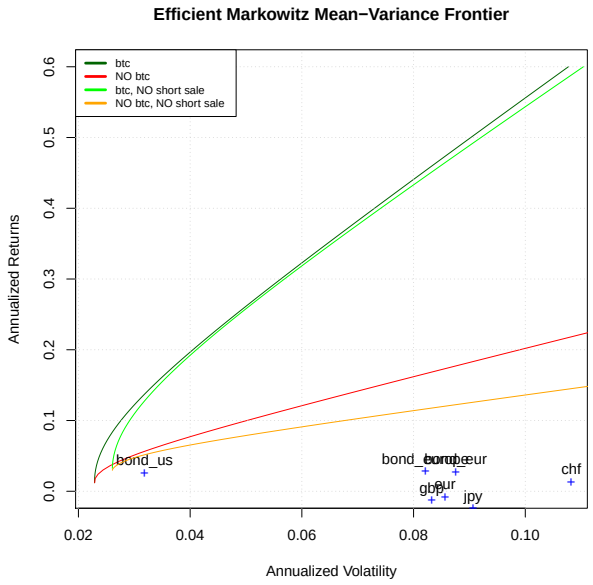


Figure 5.2: The *Efficient* Markowitz Mean-Variance frontier obtained from our portfolio of assets, both including and excluding Bitcoin and with short-selling or not.

can be stated for the red and orange curves, which represent our portfolio when excluding the digital asset. Thus, given our particular set of assets, allowing for short-selling does very little to improve the diversification of our portfolio.

Let us now take a look of what happens when we include Bitcoin in the reference portfolio: as we can see from Figure 5.2, we get a significant improvement in the expected return when considering each level of risk . Equivalently, for the same level of return we have a noticeable decrease in the volatility of our portfolio.

We can see some effects of the diversification properties of adding Bitcoin to our portfolio in Table 5.1 and Table 5.2, where the numerical results for the portfolios that do not allow short-selling are presented.

5.2.2 Portfolio Allocation

We have so far seen the implications of introducing the digital asset in our portfolio in terms of improvement in the expected return and of lowering the overall portfolio risk. Let us now take a look at how Markowitz MPT allocates the wealth into the different assets.

Table 5.1: Expected annualized return for different levels of volatility (also annualized), both including and excluding Bitcoin.

Volatility Level	Return without Bitcoin	Return including Bitcoin
2.61%	3.00%	3.00%
2.75%	3.89%	7.70%
3.00%	4.59%	10.94%
3.25%	5.05%	13.37%
3.50%	5.43%	15.48%
3.75%	5.76%	17.41%
4.00%	6.06%	19.21%
4.25%	6.34%	20.94%
4.50%	6.61%	22.60%
4.75%	6.87%	24.21%
5.00%	7.12%	25.79%
5.25%	7.37%	27.34%
5.50%	7.61%	28.86%
5.75%	7.85%	30.37%
6.00%	8.08%	31.85%

To do so, we can plot the values of $\mathbf{w}$ as resulting from (5.3) for different levels of returns (and hence volatilities): resulting allocations are plotted in Figure 5.3.

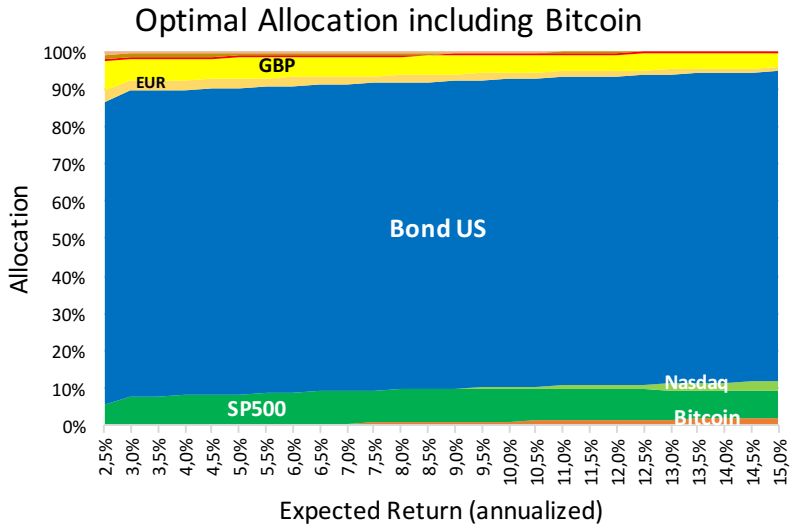
5.3 Portfolio Optimization with CVaR as a Risk Measure

We have so far studied the problem of optimal portfolio allocation through Markowitz MPT, considering the portfolio volatility as a proxy for its risk. This is only one of the possible choices in a more general framework in which the optimization problem to be solved maintains the same constraints but has a different objective function. We can reformulate (5.3) as follows:

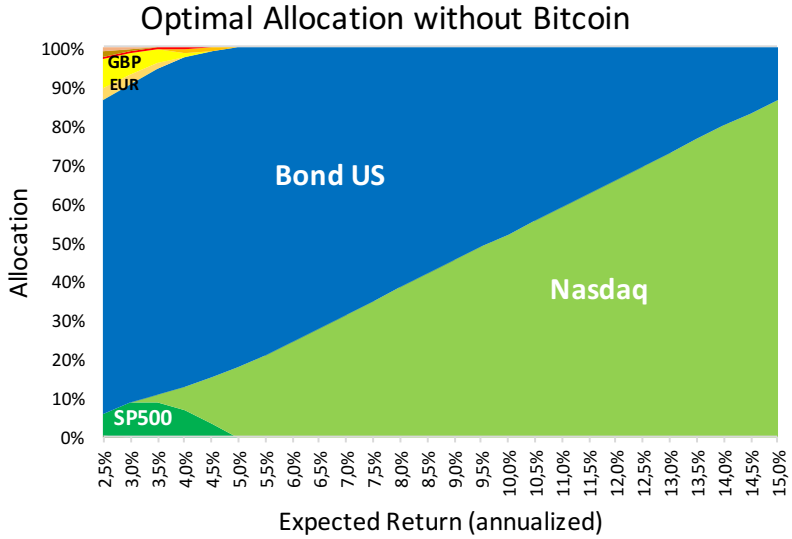
$$\begin{aligned}
\min_{\mathbf{w}\in\mathbb{R}^N} \quad & PtfRisk(\mathbf{w}) && \text{(5.4a)}\\
\text{subject to} \quad & \mathbf{e}^T\mathbf{w} = 1, && \text{(5.4b)}\\
& \mathbf{r}^T\mathbf{w} = r_{target}, && \text{(5.4c)}\\
& w_i \geq 0, \text{for } i = 1 \dots N, && \text{(5.4d)}
\end{aligned}$$

Table 5.2: Annualized volatility for different levels of expected portfolio return (also annualized), both including and excluding Bitcoin.

Return Level	Volatility without Bitcoin	Volatility including Bitcoin
3,00%	2,61%	2,61%
3,50%	2,67%	2,61%
4,00%	2,78%	2,61%
4,50%	2,96%	2,62%
5,00%	3,22%	2,63%
5,50%	3,55%	2,65%
6,00%	3,95%	2,67%
6,50%	4,40%	2,69%
7,00%	4,88%	2,71%
7,50%	5,39%	2,74%
8,00%	5,91%	2,77%
8,50%	6,45%	2,80%
9,00%	7,00%	2,84%
9,50%	7,56%	2,88%
10,00%	8,12%	2,92%
10,50%	8,69%	2,96%
11,00%	9,27%	3,01%
11,50%	9,85%	3,05%
12,00%	10,43%	3,10%
12,50%	11,01%	3,16%
13,00%	11,60%	3,21%
13,50%	12,19%	3,26%
14,00%	12,78%	3,32%
14,50%	13,37%	3,38%
15,00%	13,96%	3,44%



(a) Allocation including Bitcoin.



(b) Allocation without Bitcoin.

Figure 5.3: The figures represent the composition of the portfolio for different levels of expected returns, both including and excluding Bitcoin. The percentage of wealth invested in Bitcoin is about 1.5% for 10% expected return and 2% for 15%. Respective values of portfolio volatility can be found in Table 5.2.

where we have substituted the portfolio volatility with a general measure of the portfolio risk as a function of the weights $\mathbf{w}$.

Using the portfolio volatility as risk measure has its perks and downsides: it is an intuitive and simple way to evaluate how the possible returns vary from the expected value but it takes into account both the positive and the negative deviation from the mean. Thus, a high volatility may be caused by a few extremely high returns that are anything but something to avoid and penalize. For this reason, the notion of *semi-volatility* was introduced to only consider variations towards lower returns than the one expected. This solution however is not very popular.

A more common approach is to measure risk based on the quantile of the loss distribution [2]. The *Value-at-Risk* of level α for a loss distribution is precisely defined as the quantile of order α [3]. In formulas, VaR_α is the value such that:

$$\mathbb{P}(Loss \leq VaR_\alpha) = \alpha \tag{5.5}$$

VaR is a vastly popular and common way to measure the so called "tail-risk": the intrinsic risk contained in loss events that happen very rarely. Its advantage is that it considers only the downside of the expected return as opposed to what the volatility does.

However, VaR has drawbacks as its mathematical definition does not make it a *coherent measure of risk*. Specifically, it lacks the property of sub-additivity: the VaR of two different portfolios considered as one may be greater than the sum of the two single VaRs. This is in direct contradiction with the principle of diversification. For the complete definition of *coherent* risk measure, please refer to Appendix D.

An improved version of VaR is the *Conditional Value-at-Risk* (CVaR), also referred to as *expected shortfall* since indeed it is defined as the average loss among the values of the loss distribution that exceed the corresponding VaR level. Thus, for a continuous loss distribution is defined as:

$$CVaR_\alpha = \frac{1}{1-\alpha}\int_\alpha^1 VaR_\gamma d\gamma \tag{5.6}$$

[2]Depending on how the returns are expressed, we have different ways to compute the loss. In particular, considering the time interval $[0, T]$, for log-returns $r_{log} = log(S_T/S_0)$ the loss is simply the opposite $Loss_{log} = -r_{log}$ while for $r_\% = S_T/S_0$ the loss is $Loss_\% = 1 - r_\%$. As usual we indicate by S_t the price of the asset or the value of the portfolio at time t.

[3] There are two different notations when it comes to what the number α indicates. In this work, α is considered as the percentage of losses that are lower or equal to the value of VaR_α. The other notation is to consider α as the percentage of losses that exceed the VaR_α.

There are two advantages in the usage of CVaR as opposed to VaR. Firstly, it can be proven that CVaR is a *coherent* measure of risk, so we gain an important mathematical property that before was lacking. Secondly, we now take into account also the very extreme values that a loss distribution might have. The latter is especially true for discrete loss distributions (e.g. losses obtained from daily assets returns, which will is our main focus) since there might be a very few truly high losses that nonetheless will be considered in the computation of CVaR by taking the mean of all losses that are higher than the VaR of the same level. On the other hand, the VaR for discrete distributions just amounts to sorting all the possible losses in increasing order and taking the element in position $100 * \alpha / N_{sample}$ as our VaR_α.

These are the reasons that induced us to consider the CVaR as an alternative measure of portfolio risk to be inserted as the objective function to be minimized in the optimization problem (5.4). Considering the literature, there are already a few studies in which the conditional value at risk is used in the optimal portfolio allocation problem, above all the paper by Rockafellar and Uryasev in (Rockafellar et al., 2000). Following studies include for instance (DiClemente, 2002) and (Quaranta and Zaffaroni, 2008). Most of these approaches are based on continuous loss functions, and as a further study beyond this work it could be of interest to study the diversification effect of the inclusion of Bitcoin in the assets portfolio in those frameworks. As we are about to see, our approach focuses on optimizing the empirical historical CVaR from daily return data.

5.4 CVaR Efficient Frontier

Following a similar approach to the one we developed in previous sections regarding Markowitz optimization, we now present the CVaR efficient frontier.

5.4.1 Efficient Frontier with and without Bitcoin

In order to perform an analysis similar to what we have done using Markowitz approach in the case of the daily CVaR as the portfolio risk measure, we have to solve the following optimization problem:

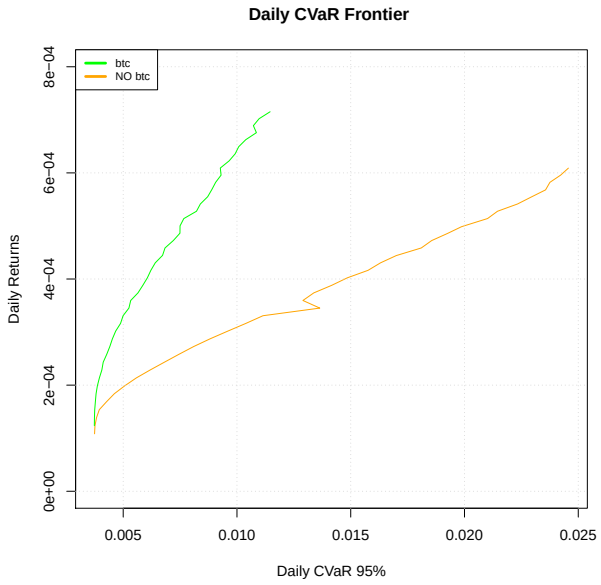


Figure 5.4: The *Efficient* daily $CVaR_{95\%}$ frontier obtained from our portfolio of assets, both including and excluding Bitcoin and without short-selling.

$$\min_{\mathbf{w}\in\mathbb{R}^N} \quad PtfCVaR_{\alpha}^{daily}(\mathbf{w}) \tag{5.7a}$$

$$\text{subject to} \quad \mathbf{e}^T\mathbf{w} = 1, \tag{5.7b}$$

$$\mathbf{r}^T\mathbf{w} = r_{target}, \tag{5.7c}$$

$$w_i \geq 0, \text{for } i = 1 \dots N. \tag{5.7d}$$

To compute the daily CVaR of the portfolio, we consider the daily percentage returns for all the assets and all the time-steps available, and store them in matrix R. R has dimension $N_{timesteps} \times N_{assets}$. We then obtain the empirical distribution of returns for a portfolio with weights $\mathbf{w}$ by doing $R_{ptf}(\mathbf{w}) = R\mathbf{w}$. The CVaR is finally computed as explained in the previous section.

We computed the efficient frontier for the two usual portfolios: one that includes Bitcoin and the other that only contains *standard* assets, both without short-selling. The graphs for both frontiers are plotted in Figure 5.4.

At first glance, we notice that both curves are not perfectly smooth: this is due to the fact that we are using the *discrete* historical distribution for the returns of the assets. On the contrary, when we set our analysis in the MPT framework it can be shown that the optimal frontier has the shape of

a hyperbole when allowing short-selling. Without short sales, the frontier is no longer a hyperbole but it is still a smooth function.

The improvement in portfolio diversification when including Bitcoin in our basket is clearly visible in how the two efficient frontiers part from each other, in the same way that they did in Figure 5.2. We can easily tell from the picture that for the same level of return an investor has to be willing to risk losing more money (higher CVaR) when he does not include Bitcoin in his portfolio of asset. Similarly, for a given level of risk, including the digital asset in our basket allows for a significant increase in the daily expected return, hence also for a higher annual profit.

Numerical results for this section are given in Table 5.3.

5.4.2 Portfolio Allocation

We have seen again the implications of introducing the digital asset in our portfolio in terms of improvement in the expected return and of lowering the daily CVaR of the portfolio. Let us now take a look at how this second type of optimization allocates the money in the different assets.

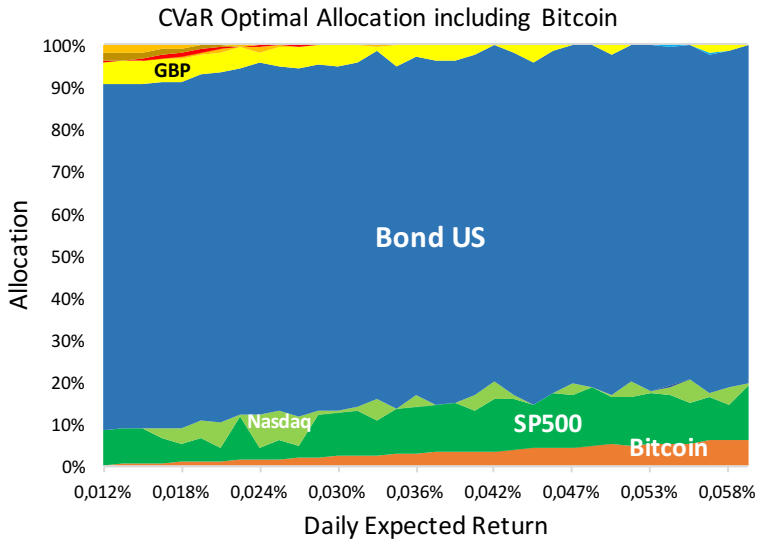
To do so, we plot the values of $\mathbf{w}$ as resulting from (5.7) for different levels of daily returns (and hence CVaRs). The optimal allocations are shown in Figure 5.5.

By comparing the allocations obtained using the volatility as a risk measure and those using the CVaR, we can see that the allocations are quite similar: when including Bitcoin, most of the wealth is allocated in the US bond index, some smaller part in the American stock indexes and a about 3% in Bitcoin. When there is no digital asset, the wealth is initially allocated in the US bond index as before but when its return is no longer enough to reach the target, the investor has to start investing more and more in the Nasdaq and S&P500 indexes.

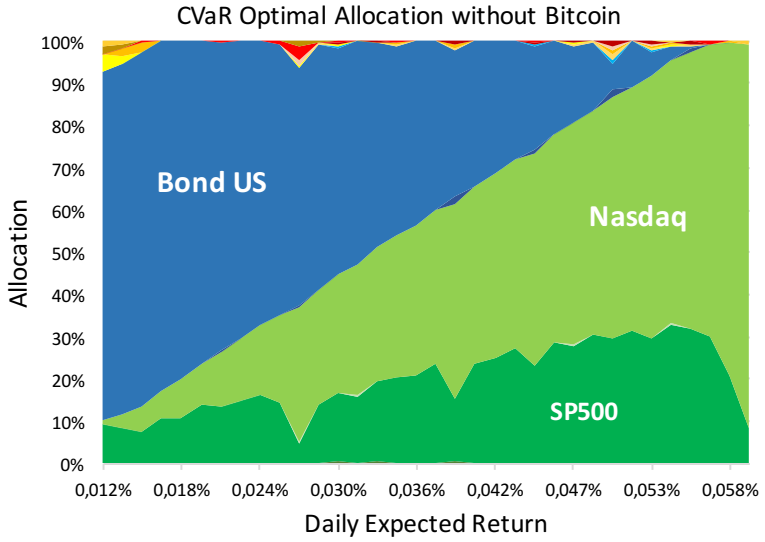
The only difference is in the fact that there is a higher percentage allocated in the S&P500 index when optimizing using the CVaR: other than that, both approaches to the optimal allocation problem suggest the same strategies.

Table 5.3: Results for the optimal daily portfolio CVaR of level 95% for a number of daily expected returns, both including and excluding Bitcoin from our basket.

Daily Return	$DailyCVaR_{95\%}$ without Bitcoin	$DailyCVaR_{95\%}$ including Bitcoin
0.012%	0.376%	0.373%
0.014%	0.383%	0.374%
0.015%	0.396%	0.375%
0.017%	0.425%	0.377%
0.018%	0.457%	0.381%
0.020%	0.505%	0.387%
0.021%	0.560%	0.394%
0.023%	0.617%	0.406%
0.024%	0.684%	0.411%
0.026%	0.753%	0.423%
0.027%	0.870%	0.436%
0.029%	0.883%	0.455%
0.030%	0.973%	0.472%
0.032%	1.026%	0.487%
0.033%	1.121%	0.495%
0.035%	1.190%	0.528%
0.036%	1.256%	0.536%
0.037%	1.334%	0.565%
0.039%	1.403%	0.585%
0.040%	1.478%	0.595%
0.042%	1.557%	0.612%
0.043%	1.637%	0.643%
0.044%	1.700%	0.679%
0.046%	1.787%	0.689%
0.047%	1.870%	0.699%
0.049%	1.936%	0.729%
0.050%	2.036%	0.763%
0.051%	2.077%	0.767%
0.053%	2.152%	0.802%
0.054%	2.241%	0.820%
0.056%	2.295%	0.831%
0.057%	2.361%	0.882%
0.058%	2.386%	0.892%
0.060%	2.421%	0.918%
0.061%	2.448%	0.950%



(a) Allocation including Bitcoin.



(b) Allocation without Bitcoin.

Figure 5.5: The figures represent the composition of the portfolio for different levels of expected returns, both including and excluding Bitcoin. The percentage of wealth invested in Bitcoin is about 3% for 0.036% expected daily return and about 6% for 0.058%. Respective values of portfolio volatility can be found in Table 5.3.

Chapter 6

Conclusions

The aim of this work was to study the properties of Bitcoin as a digital asset.

In Chapter 2 we studied the empirical correlation between the returns of Bitcoin and of 16 other assets, grouped in 4 different classes (stock indexes, bonds indexes, currencies and commodities). We found out that the correlation not only is low, but it also is not significantly different from zero, from a statistical point of view. The same holds for the rolling correlation computed in the time span of our data (mid July 2010 to early November 2018).

Then, in Chapter 4 we obtained the same correlation matrix through the calibration of three continuous asset price models: Merton's jump diffusion, Heston's stochastic volatility and Bates's stochastic volatility with jumps in the price process. For all three, the resulting correlation structure closely resembles the one computed in Chapter 2: this indicates that there is no real need for a sophisticated model to obtain a feel for what the level of the correlation of Bitcoin with other assets is.

The fact that returns of Bitcoin are not correlated with those of more standard assets lead us to consider the possibility of including the digital asset in an investment portfolio. In Chapter 5 we compare the performances of two portfolios, in a Markowitz framework: the first only includes the standard assets while the second also contains Bitcoin. The difference in terms of increased returns or lowered risk is dramatic. We can see this both by looking at the graph of the efficient frontiers and at the numerical results. We performed the same analysis implementing the daily CVaR at 95% as the portfolio risk measure to specifically account for the tails of the historical distribution, obtaining very similar results to the previous case. In general, investing 3% to 5% of our wealth in Bitcoin proved to be extremely beneficial.

A possible further study can be to analyse the correlation between the

most relevant crypto-currencies in the same way that we did here with standard assets. We expect to find that the correlation levels are pretty high and thus it would not prove to be beneficial to include more than one in an investment portfolio.

Another interesting subject of study for a deeper investigation of the asset allocation problem would be to implement a Black and Litterman approach, which allows to also take into account the investor's assumptions on the market returns of the assets.

Even more than the results of the whole numerical analysis that we performed in this work, we believe that the properties, the infrastructure and the community behind Bitcoin make it a cut above other crypto-currencies and stand tall against scepticism and misunderstandings, rendering it the only resilient and durable crypto-investment.

Bibliography

Albrecher, H., Mayer, P., Schoutens, W., and Tistaert, J. (2007). The little heston trap. *Wilmott*, pages 83–92.

Andersen, L. B. (2007). Efficient simulation of the heston stochastic volatility model. *Available at SSRN 946405*.

Andrianto, Y. (2017). The effect of cryptocurrency on investment portfolio effectiveness. *Journal of Finance and Accounting*, 5:229.

Ardia, D., Arango, J. O., and Gomez, N. G. (2011). Jump-diffusion calibration using Differential Evolution. *Wilmott Magazine*, 55:76–79.

Ardia, D., Mullen, K. M., Peterson, B. G., and Ulrich, J. (2016). *DEoptim: Differential Evolution in R.* version 2.2-4.

Ball, C. A. and Torous, W. N. (1983). A simplified jump process for common stock returns. *Journal of Financial and Quantitative Analysis*, 18(01):53–65.

Bates, D. S. (1996). Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options. *Review of Financial Studies*, 9(1):69–107.

Bouri, E., Molnár, P., Azzi, G., Roubaud, D., and Hagfors, L. I. (2017). On the hedge and safe haven properties of bitcoin: Is it really more than a diversifier? *Finance Research Letters*, 20.

Conceicao, E. L. T. (2016). *DEoptimR: Differential Evolution Optimization in Pure R.* R package version 1.0-8.

Cooley, J. W. and Tukey, J. W. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301.

Cox, J. C., Ingersoll, J. E., and Ross, S. A. (1985). A theory of the term structure of interest rates. *Econometrica*, 53(2):385–407.

DiClemente, A. (2002). The empirical value at risk-expected return frontier: A useful tool of market risk managing. *Centro Interdipartimentale sul Diritto e l?Economia dei Mercati*, (11).

Dimitroff, G., Lorenz, S., and Szimayer, A. (2011). A parsimonious multi-asset heston model: Calibration and derivative pricing. *International Journal of Theoretical and Applied Finance (IJTAF)*, 14(08):1299–1333.

Dragulescu, A. A. and Yakovenko, V. M. (2002). Probability distribution of returns in the heston model with stochastic volatility. *Quantitative Finance*, 2(6):443–453.

Dyhrberg, A. H. (2016). Hedging capabilities of bitcoin. is it the virtual gold? *Finance Research Letters*, 16:139–144.

Eisl, A., Gasser, S., and Weinmayer, K. (2015). Caveat emptor: Does bitcoin improve portfolio diversification? *SSRN Electronic Journal*.

Heston, S. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies*, 6:327–43.

Honoré, P. (1998). Pitfalls in estimating jump-diffusion models. *SSRN Electronic Journal*.

Klein, T., Thu, H. P., and Walther, T. (2018). Bitcoin is not the new gold: a comparison of volatility, correlation, and portfolio performance. *International Review of Financial Analysis*, 59:105 – 116.

Kou, S. G. (2002). A jump-diffusion model for option pricing. *Management Science*, 48(8):1086–1101.

Lord, R., Koekkoek, R., and Dijk, D. V. (2010). A comparison of biased simulation schemes for stochastic volatility models. *Quantitative Finance*, 10(2):177–194.

Markowitz, H. (1952). Portfolio selection. *The Journal of Finance*, 7(1):77–91.

Martin, M. (2007). A two-asset jump diffusion model with correlation. Master's thesis, University of Oxford.

Merton, R. (1976). Option prices when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3:125–144.

Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system.

Papi, M., Cacace, F., and Germani, A. (2015). On parameter estimation of heston's stochastic volatility model: a polynomial filtering method.

Price, K. V., Storn, R. M., and Lampinen, J. A. (2006). *Differential Evolution - A Practical Approach to Global Optimization*. Natural Computing. Springer-Verlag. ISBN 540209506.

Quaranta, A. G. and Zaffaroni, A. (2008). Robust optimization of conditional value at risk and portfolio selection. *Journal of Banking & Finance*, 32(10):2046–2056.

Rockafellar, R. T., Uryasev, S., et al. (2000). Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42.

Tankov, P. and Cont, R. (2015). *Financial Modelling with Jump Processes, Second Edition*. Chapman and Hall/CRC Financial Mathematics Series. Taylor & Francis.

Vasicek, O. (1977). An equilibrium characterization of the term structure. *Journal of Financial Economics*, 5(2):177 – 188.

Appendix A

Characteristic Function of a Compound Poisson Process

By definition, the characteristic function of a random variable X is given by:

$$\phi_X(u) = \mathbb{E}[e^{iuX}]. \tag{A.1}$$

Let us first consider a simple Poisson process N_t of parameter λ. At each time $t > 0$, N_t has a discrete distribution that follows:

$$\mathbb{P}(N_t = n) = e^{-\lambda t}\frac{(\lambda t)^n}{n!},\ n = 0, 1, 2, ... \tag{A.2}$$

Since N_t is a *discrete* random variable, the expectation in (A.1) amounts to a sum over all the possible values of N_t:

$$\begin{aligned}\phi_{N_t}(u) = \mathbb{E}[e^{iuN_t}] &= \sum_{n=0}^{\infty} e^{iun}\mathbb{P}(N_t = n) \\ &= \sum_{n=0}^{\infty} e^{iun}e^{-\lambda t}\frac{(\lambda t)^n}{n!} \\ &= e^{-\lambda t}\sum_{n=0}^{\infty}\frac{(\lambda t e^{iu})^n}{n!}.\end{aligned}$$

Using the definition of exponential $e^x = \sum_{n=0}^{\infty}\frac{x^n}{n!}$ we then get the final result:

$$\begin{aligned}\phi_{N_t}(u) &= e^{-\lambda t}e^{\lambda t e^{iu})} \\ &= e^{\lambda t(e^{iu}-1)}.\end{aligned}$$

Let us consider now a compound Poisson process defined by

$$X_t = \sum_{i=1}^{N_t} Y_i \tag{A.3}$$

where Y_i are i.i.d. and have density expressed by the function $f_Y(y)$.

To compute the characteristic function of X_t we can follow the same steps as in the simple Poisson case, but in addition we use the Theorem of Total Expectation to first simplify the expression and then proceed exploiting the i.i.d property:

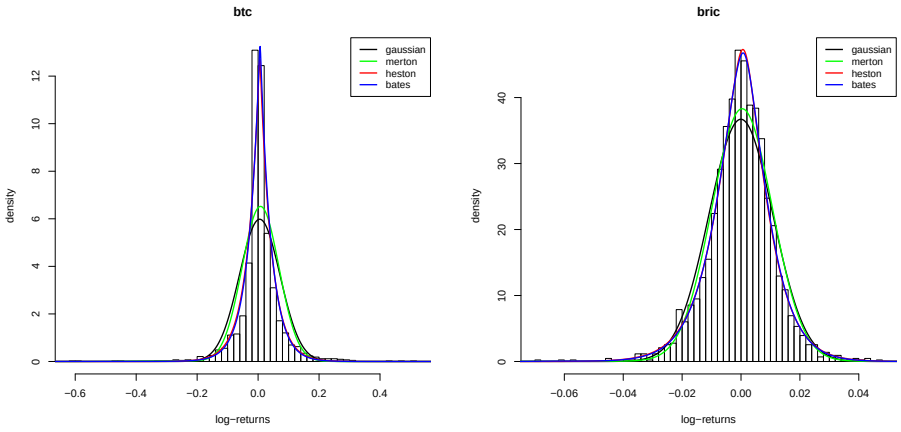
$$\begin{aligned}
\phi_{X_t}(u) &= \mathbb{E}[e^{iuX_t}] \\
&= \sum_{n=0}^{\infty} \mathbb{E}[e^{iuX_t}|N_t = n]\mathbb{P}(N_t = n) \\
&= \sum_{n=0}^{\infty} \mathbb{E}[\prod_{i=1}^{N_t} e^{iuY_i}|N_t = n]\mathbb{P}(N_t = n) \\
&= \sum_{n=0}^{\infty} \prod_{i=1}^{n} \mathbb{E}[e^{iuY_i}|N_t = n]\mathbb{P}(N_t = n) \\
&= \sum_{n=0}^{\infty} (\mathbb{E}[e^{iuY}])^n \mathbb{P}(N_t = n) \\
&= \sum_{n=0}^{\infty} (\phi_Y(u))^n e^{-\lambda t} \frac{(\lambda t)^n}{n!} \\
&= e^{-\lambda t} \sum_{n=0}^{\infty} \frac{(\lambda t\, \phi_Y(u))^n}{n!} \\
&= e^{\lambda t(\phi_Y(u)-1)}.
\end{aligned}$$

Appendix B

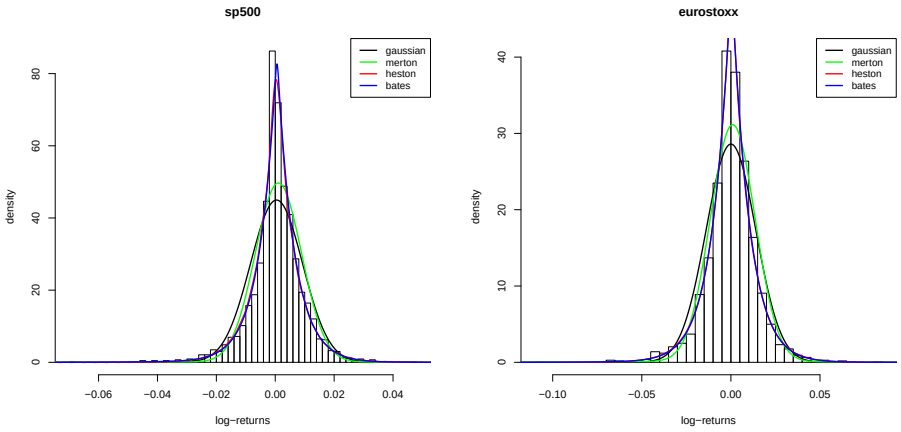
Histograms

In the following pages we include the histograms of the distributions of the log-returns for each of the assets in our dataset. We superimpose to the histograms the graphs of the density functions obtained using the three models of Merton, Heston and Bates with the calibrated parameters shown in Tables 4.2, 4.4 and 4.6.

We can see that given our results from the calibration, Heston and Bates better approximate the empirical distribution of the returns in all cases.

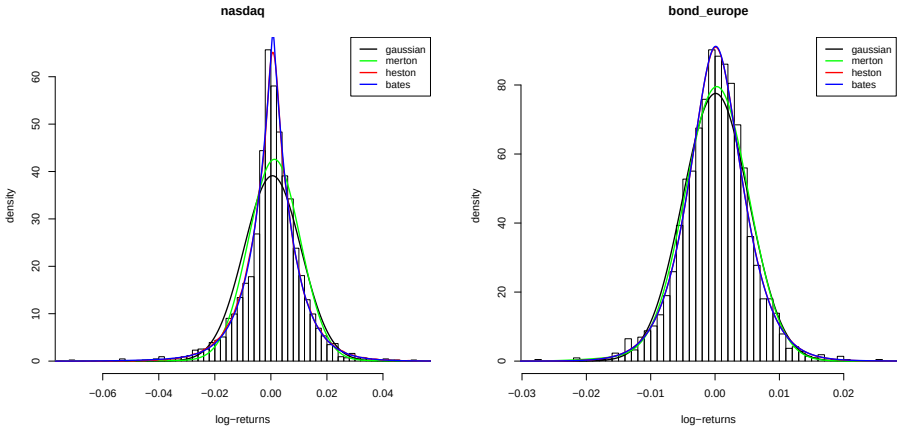


(a) Bitcoin (b) Bric index



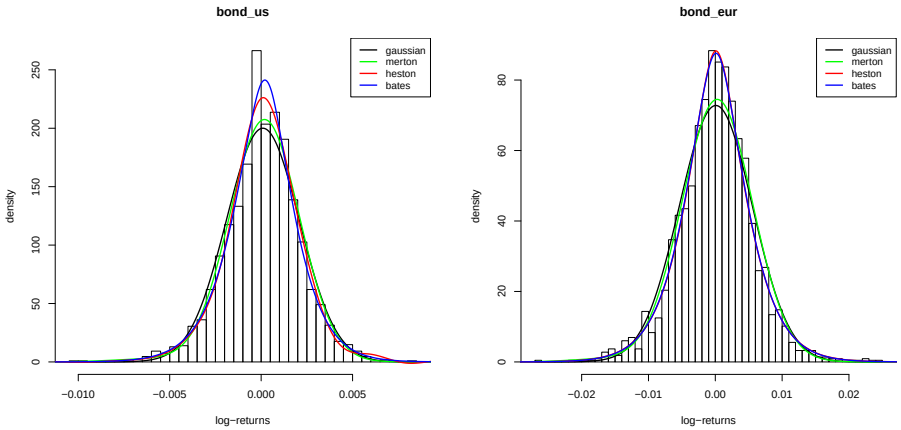
(c) S&P500 (d) Eurostoxx50

Figure B.1: Histogram of the log-returns and pdf for the three models we calibrated and the Gaussian line as reference.[1/3]



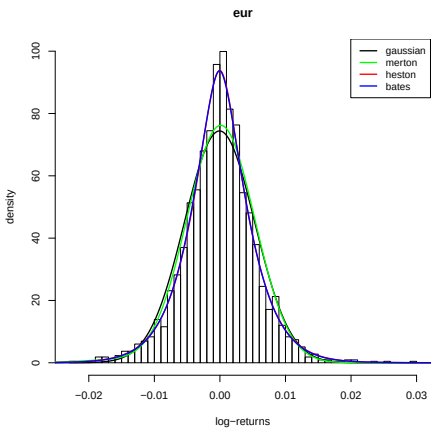
(a) Nasdaq

(b) Bond Europe

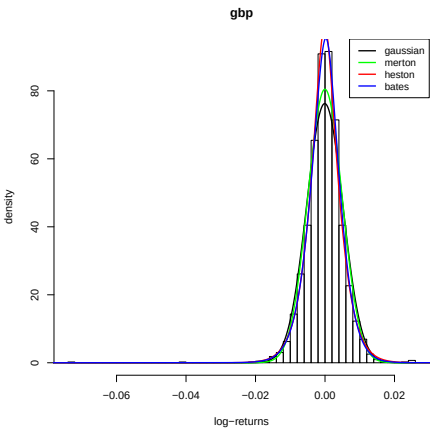


(c) Bond US

(d) Bond EUR

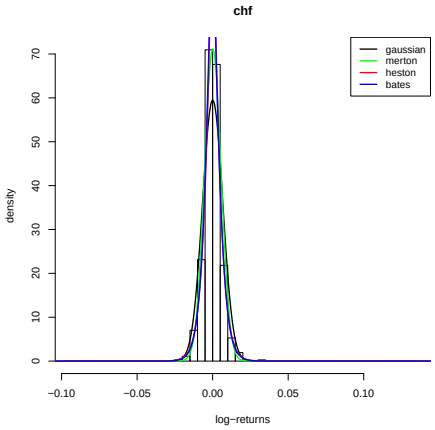


(e) EUR/USD

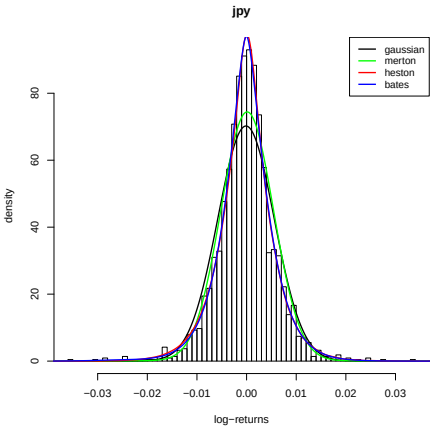


(f) GBP/USD

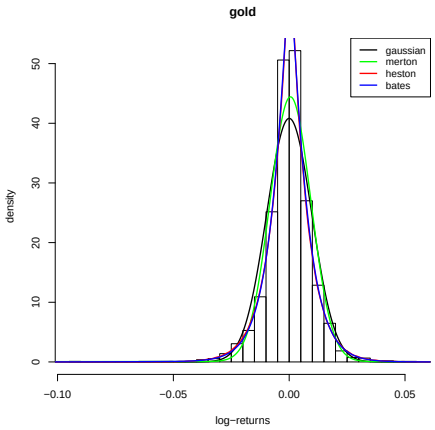
Figure B.2: Histogram of the log-returns and pdf for the three models we calibrated and the Gaussian line as reference.[2/3]



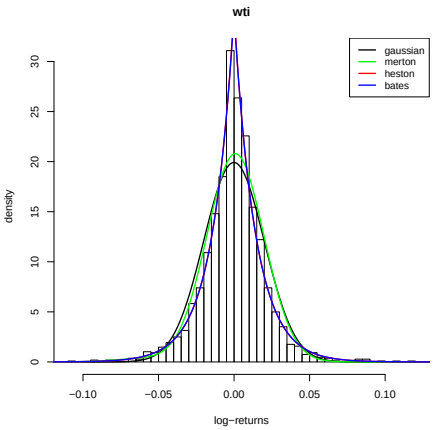
(a) CHF/USD



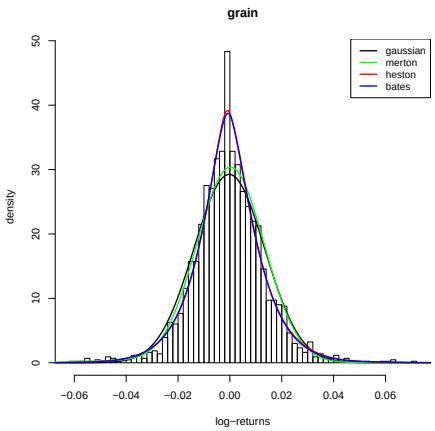
(b) JPY/USD



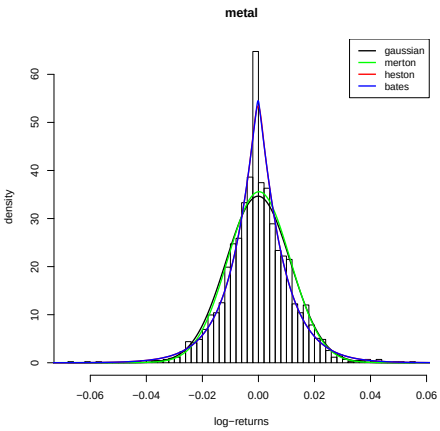
(c) Gold



(d) Wti oil



(e) Grain



(f) Metals

Figure B.3: Histogram of the log-returns and pdf for the three models we calibrated and the Gaussian line as reference. [3/3]

Appendix C

Discrete Fourier Transform and FFT

The Fast Fourier Transform is an algorithm that enables us to compute the Discrete Fourier Transform faster. In particular, a *naïve* implementation of the DFT requires a number of operations on the order of $\mathcal{O}(N^2)$, where N is the number of points that we use for the discrete transform. An FFT algorithm performs the same computation using $\mathcal{O}(N \log N)$ operations.

The discrete Fourier transform for a set $u = u_0, ..., u_{N-1}$ is represented by N values x_k, $k = 0, ..., N-1$:

$$x_k = \sum_{n=0}^{N-1} u_n \, e^{-ikn2\pi/N}, \; k = 0, \ldots, N-1. \tag{C.1}$$

As we can see, we have to perform $\mathcal{O}(N)$ computations for each x_k, for a total of $\mathcal{O}(N^2)$ operations. The FFT is a smart way to compute the same quantities by dividing $N = N_1 N_2$ into two factors N_1 and N_2, computing the DFT for the two smaller samples and then aggregating the results to return the final N values of x_k.

The factorization of N can be performed recursively on N_1 and N_2 as long as they are not primes. The usual approach is thus to take $N = 2^m$ as the m-th power of 2. The most common algorithm is the one by Cooley and Tuckey in Cooley and Tukey (1965) and is the one that is usually implemented in most computational tools.

In our case, we want to compute the density function $f(x)$ starting from the characteristic function $\phi(u)$:

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \phi(iu) e^{iux} du \tag{C.2}$$

using the formula in (C.1). To do so we need first to define for which points we want the density to be computed. Let us say we want to obtain values of $f(x)$ for $x \in [a, b)$ using a discretization of $N = 2^m$ equidistant points. Thus we will have $x_k = a + k\Delta x$ where $\Delta x = (b - a)/N$.

The grid for the u_n is computed using $\Delta u = 2\pi/(N\Delta x)$ and using an interval centred on zero: $u_n = -N/2 + n\Delta u$. This allows us to obtain a formulation that is similar to (C.1) and to take advantage of the FFT algorithm, as we will now show.

We first obtain a discretized version of (C.2) by rectangular approximation and then perform some manipulations to arrive at an expression that looks like that of the DFT:

$$
\begin{aligned}
f(x_k) &= \frac{1}{2\pi} \sum_{n=0}^{N-1} \phi(iu_n)\, e^{iu_n x_k} \Delta u \\
&= \frac{\Delta u}{2\pi} \sum_{n=0}^{N-1} \phi\big(i(-N/2 + n\Delta u)\big)\, e^{i(-N/2+n\Delta u)x_k} \\
&= \frac{\Delta u}{2\pi} \sum_{n=0}^{N-1} \phi\big(i(-N/2 + n\Delta u)\big)\, e^{in\Delta u x_k} e^{i(-N/2)x_k} \\
&= \frac{\Delta u}{2\pi} \sum_{n=0}^{N-1} \phi\big(i(-N/2 + n\Delta u)\big)\, e^{in\Delta u(a+k\Delta x)} e^{i(-N/2)x_k} \\
&= \frac{\Delta u}{2\pi} \left[\sum_{n=0}^{N-1} \phi\big(i(-N/2 + n\Delta u)\big)\, e^{in\Delta u\, a}\, e^{in\Delta u\, k\Delta x} \right] e^{i(-N/2)x_k} \\
&= \frac{\Delta u}{2\pi} \left[\sum_{n=0}^{N-1} \left(\phi\big(i(-N/2 + n\Delta u)\big)\, e^{in\Delta u\, a} \right) e^{ink2\pi/N} \right] e^{i(-N/2)x_k}.
\end{aligned}
$$

We can see that in the last line the part that is limited by the square brackets has the same form as (C.1) and thus it can be quickly computed using any FFT algorithm.

Appendix D

Coherent Risk Measure

Let us define $\mathcal{L}$ the set of general loss distributions (whether they are discrete or continuous it does not matter), and let ρ be a functional defined from $\mathcal{L}$ to $\mathbb{R} \cup \{\infty\}$.

We define ρ to be a *coherent risk measure* if the following four properties hold:

- *subadditivity*: $\forall L_1, L_2 \in \mathcal{L}, \quad \rho(L_1 + L_2) \leq \rho(L_1) + \rho(L_2)$.
- *positive homogeneity*: $\forall \alpha \in \mathbb{R}, \forall L \in \mathcal{L}, \quad \rho(\alpha L) = \alpha \rho(L)$.
- *translation invariance*: $\forall \alpha \in \mathbb{R}, \forall L \in \mathcal{L}, \quad \rho(L + \alpha) = \rho(L) + \alpha$.
- *monotonicity*: $\forall L_1, L_2 \in \mathcal{L}$ such that $L_1 \leq L_2, \quad \rho(L_1) \leq \rho(L_2)$.

The interpretation of the last three properties is immediate and it makes sense that a *coherent* risk measure would satisfy them. The subadditivity is instead a little trickier but it is fundamental since negating it would mean that there is no advantage in diversification, which is known not to be true both by everyday experiences and numerical evidences.

The VaR as a measure of risk is not coherent because it lacks the fundamental property of subadditivity.

Appendix E

Code excerpts

E.1 Correlation analysis

Function to perform the Permutation test:

```
1
2  PermutationTestCorr = function(x,y=0, N=2000){
3    # Two sided permutation test for correlation
4    # H0: rho = 0   vs    H1: rho!=0
5    # Input
6    #   x: first variable values
7    #   y: second variable values
8
9    if (!is.null(dim(x)[2])){
10     if ( dim(x)[2] == 2){
11     y=x[,2]
12     x=x[,1]
13     }
14   }
15   else if (length(x)!=length(y)){
16     stop("Error: samples should have same size.")
17   }
18
19   n = length(x)
20
21   r_sample= cor(x,y)
22
23   larger = 0
24   for(i in 1:N){
25     y_perm = y[sample(n,n)]
26     r_perm = cor(x,y_perm)
27     if (abs(r_sample)<abs(r_perm))
28       larger = larger+1
29   }
30   # p-value is the percentage of r_perm absolutely greater than r_sample
31   p = larger/N
32   return(p)
33 }
```

E.2 Merton, Heston and Bates calibration

E.2.1 General structure for full calibration

Script to calibrate the `N_assets` Merton model, similar structure also for the other models, just different functions for parameters and model correlation

calibration:

```
### Loading daily log-returns dataset
load("returns.Rda")
attach(my_returns)


N = dim(my_returns)[1] #total number of observations
N_assets= 16
dt = 1/255

# matrix to store the parameters for each asset
full_results = matrix(0, nrow = N_assets, ncol = 5)
rownames(full_results) = colnames(my_returns)[2*(1:N_assets)]
colnames(full_results) = c("mu", "sigma","theta","delta","lambda")

### Calibrating each single asset model
for (i in 1:N_assets){
  asset = my_returns[,2*i] #contains single asset daily log-returns
  asset_name = colnames(my_returns)[2*i]

  # actual calibration function
  calibrated_params = CalibrateMerton(x=matrix(asset[1:N], ncol=1),dt = dt, custom_jump_bounds = T)

  full_results[i,]=c(calibrated_params$mu, sqrt(calibrated_params$S), calibrated_params$theta,
     calibrated_params$delta, calibrated_params$lambda)
}


#### Calibrating model Correlation matrix

n=16
# sample corr matrix for comparison
corr_matrix = cor(my_returns[,2*(1:n)])

beg = Sys.time()
# model correlation matrix
model_corr_mat = calibrate_full_correlation_merton(sample_corr_matrix = corr_matrix, Nasset = n,
            mu =full_results[1:n,1] , vol = full_results[1:n,2], mu_j = full_results[1:n,3],
            sigma_j = full_results[1:n,4], lambda = full_results[1:n,5], dt = 1/255,final_t = 2,
            Nsim = 1000)
# show computation time
Sys.time() - beg

# print both matrices to screen for comparison
model_corr_mat
corr_matrix[1:n,1:n]
```

E.2.2 Single asset calibration

Function that performs the calibration for a single asset Merton, only differences in the Heston and Bates cases is the objective function in the optimizers that are respectively `heston_negloglik` and `bates_negloglik`

```
CalibrateMerton=function(x, n, dt, trace = 10, custom_jump_bounds =T){

  library(DEoptim)
  source("MultivariateMertonModel.R")

  if(custom_jump_bounds){
    min_jump = rep(0,n)
    max_jump = rep(0,n)

    alpha_max = 0.995 # quantile for max jump
    alpha_min = 0.999 # quantile for min jump
    for (i in 1:n) {
      min_jump[i] = 2*quantile(x= x[,i], probs = 1-alpha_min)
      max_jump[i] = quantile(x= x[,i], probs = 1-alpha_max)
    }
  }
  else{
    min_jump = rep(-0.1,n) # default jumps at -10%
    max_jump = rep(-1,n)
  }
```

```
21
22    # Obtaining bounds from function BoundsCreator
23    bounds_nocommon = BoundsCreator(custom_jump_mean = custom_jump_bounds,
24    max_jump_mean = max_jump, min_jump_mean = min_jump)
25
26    print(rbind(bounds_nocommon$lower, bounds_nocommon$upper))
27
28    # First optimization using deoptim
29    print("Starting calibration using DEoptim...")
30    control_list_deoptim = list(itermax = 50, NP = 10*length(bounds_nocommon$lower), strategy = 6,trace
          =trace)
31
32
33    start_time_deoptim <- Sys.time()
34    outDE <- DEoptim(merton_negloglik,  # objective function is the negative log likelihood for Merton
35        lower = bounds_nocommon$lower, upper = bounds_nocommon$upper, control = control_list_deoptim,
          dt = dt, x = x)
36    end_time_deoptim <- Sys.time()
37
38    # Second and final optimization using nlminb
39    print("Starting calibration using nlminb...")
40
41    initial=outDE$optim$bestmem
42    start_time_nlminb <- Sys.time()
43    out_nlminb = nlminb(initial,objective = merton_negloglik,lower = bounds_nocommon$lower,
44    upper = bounds_nocommon$upper,dt=dt, x=x,
45    control=list(eval.max = 10000,iter.max = 1000, trace = trace))
46    print(out_nlminb)
47    end_time_nlminb <- Sys.time()
48
49    print(paste("DEoptim time:",end_time_deoptim - start_time_deoptim))
50    print(paste("nlminb time:", end_time_nlminb - start_time_nlminb))
51    print(paste("TOTAL time:", end_time_nlminb - start_time_deoptim))
52
53    res = ParametersReconstruction(out_nlminb$par,n=n)
54    res[["message"]] = out_nlminb$message
55    res[["objective_function"]] = out_nlminb$objective
56    res[["total_time"]] = end_time_nlminb - start_time_deoptim
57    return(res)
58  }
```

Pdf and log-likelihood function for Merton:

```
1   #Pdf for Merton model
2   merton_pdf = function(x, dt, mu, sigma, mu_J, sigma_J, lambda){
3     # Computes the density of a multivariate merton model returns with idiosyncratic and common jumps
4     # ASSUMPTION: in dt time we can only have 0 or 1 jumps in each jump process, so lambda*dt<=1
5     #
6     # INPUT
7     # x:      vector representing at which point to compute the density
8     # mu:     drift of the continuos part
9     # sigma:      covariance of the continuous part
10    # mu_J:  means of the idiosyncratic jump intensity
11    # sigma_J:  volatility of the idiosyncratic jump intensity
12    # lambda:  poisson parameters of the idiosyncratic jump part
13
14    # check on lambdas:
15    ldt =lambda*dt
16    if(ldt>=1){
17      stop("Error: lambda*dt should be lower than 1 (ideally close to 0).")
18    }
19
20    mu_adj= mu - sigma^2/2 -lambda*mu_J
21    pdf=(1-ldt)*dnorm(x, mean = mu_adj*dt, sd = sqrt(sigma^2*dt))+ldt *dnorm(x, mean = mu_adj*dt + mu_J
          , sd = sqrt(sigma^2*dt + sigma_J^2))
22
23    return(pdf)
24  }
25
26
27  merton_negloglik= function(params, x, dt) {
28    # x is a matrix [Npoints * n] of all the points for which we compute the likelihood
29
30    # reconstruction of parameters:
31    mu=params[1]
32    sigma = params[2]
33    mu_J = params[3]
34    sigma_J = params[4]
35    lambda = params[5]
36
37    # computing pdf on each point and adding
```

```
  partial = merton_pdf(x, dt, mu, sigma, mu_J, sigma_J, lambda)
  nll = -sum(log(partial))

  # last check on result
  if (is.nan(nll) | is.na(nll) | is.infinite(nll)) {
    nll = 1e10
  }

  return(nll)
}
```

Pdf, unconditional chf and negative log-likelihood functions for Heston:

```
# Pdf for Heston usinf FFT
pdfHeston_fft = function(x, dt, r, k, theta, sigma_V, rho, N = 2^10){
  # Auxiliary function
  aux_chf = function(u,tau = dt, r_cf = r, k_cf = k,vT = theta, sigma = sigma_V, rho_cf = rho ){
    uncond_cfHeston(u,tau ,r_cf ,k_cf,vT,sigma,rho_cf)
  }

  min_x = min(x, -1)
  max_x = max(x, 1)
  eps = 0.01

  # Function that performs the inversion by using the FFT algorithm
  pdf = characteristic_function_to_density(aux_chf,N, min_x-eps, max_x+eps)

  pdf_xx = interp1(pdf$x, pdf$density, x,method = 'spline')
  return(pdf_xx)
}


# Neg logLikelihood function for Heston
heston_negloglik= function(params, x, dt, check_feller=TRUE){
  r = params[1]
  k =params[2]
  theta = params[3]
  sigma_V = params[4]
  rho = params[5]

  pdfs= pdfHeston_fft(x=x, dt=dt,
  r = r, k=k, theta=theta, sigma_V = sigma_V, rho = rho)

  to_sum = log(pdfs)
  nll = -sum(to_sum)

  if (is.nan(nll) | is.na(nll) | is.infinite(nll)) {
    nll = 1e10
  }

  return(nll)
}



# Unconditional chf for Heston
uncond_cfHeston = function(om, tau, r, k, vT, sigma, rho){

  if (sigma < 1e-08)
    sigma <- 1e-08

  om[which(om==0)]= 1e-8
  d <- sqrt((rho * sigma * (0+1i) * om - k)^2 + sigma^2 * ((0+1i) *om + om^2))
  g <- (k - rho * sigma * (0+1i) * om - d)/(k - rho * sigma * (0+1i) * om + d)

  # to avoid nan due to 0/0
  g[which((k - rho * sigma * (0+1i) * om - d)==0)]=0

  A <- 1i*om *r * tau + vT * k/(sigma^2) * ((k - rho * sigma * (0+1i) * om - d) * tau - 2 * log((1 -
      g * exp(-d * tau))/(1 - g)))
  B <- (k - rho * sigma * (0+1i) * om - d)/sigma^2 * (1 - exp(-d * tau))/(1 - g * exp(-d * tau))

  w = 2*k/sigma^2
  nu = 2*k*vT/sigma^2

  res = exp(A)*(w/(w-B))^nu
  return(res)
}
```

Pdf, unconditional chf and negative log-likelihood functions for Bates:

```
# pdf for Bates using FFT
pdfBates_fft = function(x,  dt,  r, k,theta, sigma_V, rho, lambda, mu_j, sigma_j,N=2^12){
  # Auxiliary function
  aux_bates = function( u, tau=dt, r_cf=r, vT=theta, rho_cf=rho, k_cf=k, sigma=sigma_V, lambda_cf=
      lambda, muJ=mu_j, vJ=sigma_j){
    uncond_cfBates(u,tau, r_cf, vT, rho_cf, k_cf, sigma, lambda_cf, muJ, vJ)
  }

  min_x = min(x, -1)
  max_x = max(x, 1)
  eps = 0.1

  # Function that performs the inversion by using the FFT algorithm
  pdf = characteristic_function_to_density(aux_bates,N, min_x-eps, max_x+eps)

  pdf_xx = interp1( x = pdf$x, y = pdf$density, xi = x, method = 'spline')
  return(pdf_xx)
}


# Neg logLikelihood function for Bates
bates_negloglik = function(params, x, dt, model="bates", check_feller){
  r = params[1]
  k=params[2]
  theta=params[3]
  sigma_V = params[4]
  rho = params[5]
  mu_j= params[6]
  sigma_j=params[7]
  lambda=params[8]

  pdfs= pdfBates_fft(x=x,dt=dt, r=r,k=k, theta=theta,sigma_V=sigma_V,rho =rho,
  mu_j = mu_j, sigma_j = sigma_j, lambda=lambda)

  to_sum = log(pdfs)
  nll = -sum(to_sum)

  if (is.nan(nll) | is.na(nll) | is.infinite(nll)) {
    nll = 1e10
  }

  return(nll)
}

# Unconditional chf for Bates
uncond_cfBates= function (om,tau, r, vT, rho, k, sigma, lambda, muJ, vJ)
  {
  if (sigma < 1e-08)
    sigma <- 1e-08
  #sigma <- max(sigma, 1e-04)

  om[which(om==0)]= 1e-9
  om1i <- om * (0+1i)

  d <- sqrt((rho * sigma * om1i - k)^2 + sigma^2 * (om1i + om^2))
  g <- (k - rho * sigma * om1i - d)/(k - rho * sigma * om1i + d)

  # to avoid nan due to 0/0
  g[which((k - rho * sigma * (0+1i) * om - d)==0)]=0

  cf1 <- om1i * ( r * tau)
  cf2 <- vT * k/(sigma^2) * ((k - rho * sigma * om1i - d) * tau - 2 * log((1 - g * exp(-d * tau))/(1
      - g)))
  cf_jump <- -lambda * muJ * om1i * tau + lambda * tau * ((1 + muJ)^(om1i) * exp(vJ * (om1i/2) * (
      om1i - 1)) - 1)

  A = cf1+cf2+cf_jump
  B = 1/sigma^2 * (k - rho * sigma * om1i - d) * (1 - exp(-d * tau))/(1 - g * exp(-d * tau))

  w = 2*k/sigma^2
  nu = 2*k*vT/sigma^2

  res = exp(A)*(w/(w-B))^nu
  return(res)
}
```

E.2.3 Model correlation calibration

Functions to perform the model correlation matrix calibration in the Merton case:

```
library(MASS)

# Performs the simulation of Merton model given the parameters in input
simulate_mv_merton = function(Nasset=length(mu), mu, vol=NA, mu_j, sigma_j, lambda, CorrMatrix=matrix
     (NA), CovMatrix = matrix(NA),
x0=rep(0,Nasset), dt=1/255, final_t=1, Nsim=100){

if(is.na(CorrMatrix[1])& Nasset==1){
CorrMatrix=matrix(1)
}
if(is.na(CovMatrix[1])){
if (is.na(vol[1]) | is.na(CorrMatrix[1])){
stop("Need to specify volatilities and Correlation Matrix, or only Covariance Matrix")
}
}
if(is.na(vol[1]) & is.na(CorrMatrix[1])){
CorrMatrix = cov2cor(CovMatrix)
vol = sqrt(diag(CovMatrix))
}

if (Nasset!=length(mu) | Nasset!=length(x0) |
Nasset!=length(mu_j) | Nasset!=length(sigma_j) | Nasset!=length(lambda)){
#print(paste(Nasset,length(mu),length(x0),length(mu_j),length(sigma_j),length(lambda)))
stop("Wrong dimension of parameters or initial values.")
}

Nstep = ceil(final_t / dt)
sim_x =array(0, dim = c(Nsim, Nstep+1, Nasset), dimnames = list(NULL,NULL,paste0("X_",1:Nasset)))
for (m in 1:Nasset) {
sim_x[,1,m] = x0[m]
}

for (i in 1:(Nstep)) {
z = mvrnorm(n = Nsim,mu=rep(0,Nasset) ,Sigma =CorrMatrix)
for (m in 1:Nasset) {
dq = rpois(Nsim, lambda = lambda[m]*dt)
# jump = rnorm(Nsim, mean = log(1+mu_j[m]) - 0.5*sigma_j[m]^2, sd = sigma_j[m])
jump = rnorm(Nsim, mean = mu_j[m], sd = sigma_j[m])
sim_x[, i+1,m] = sim_x[,i,m] + (mu[m] - vol[m]^2*0.5 -lambda[m]*mu_j[m] )* dt + vol[m]*sqrt(dt)*z[,m]
      + dq * (jump)
}
}

res_x = lapply(seq(dim(sim_x)[3]), function(t) sim_x[ , , t])
names(res_x) = paste0("x_",1:Nasset)
return( res_x)
}



# Computes correlation matrix from given simulation result
correlation_MC_estimation = function( simulation_result){
# simulation result should be a n-asset simulation output of simulate_mv_merton
# NOTE: correlation is computed on the log returns in [t,t+dt]

Nasset = length(simulation_result)
Nsim = dim(simulation_result[[1]])[1]

if( Nasset == 2 ){
sumcorr= 0
for (k in 1:Nsim) {
# correlation of log-returns in path k
sumcorr = sumcorr + cor(diff(simulation_result[[1]][k,]), diff(simulation_result[[2]][k,]))
}
res = sumcorr/Nsim      # average correlation on different scenarios
}
else if (Nasset > 2){
corr_matrix = diag(nrow = Nasset) # Identity matrix
for ( i in 1:(Nasset-1)) {
for (j in (i+1):Nasset) {
sumcorr = 0
for (k in 1:Nsim) {
sumcorr = sumcorr + cor(diff(simulation_result[[i]][k,]), diff(simulation_result[[j]][k,]))
}
corr_matrix[i,j] = sumcorr/Nsim
```

```
corr_matrix[j,i] = sumcorr/Nsim
}
}
res = corr_matrix
}
else{
stop("Need at least 2 assets to compute correlation.")
}
return(res)
}

# Computes correlation for 2 assets from initial and model parameters [ used in model correlation calibration]
expected_model_correlation_merton = function(model_corr, mu, vol, mu_j, sigma_j, lambda,
x0, dt=1/255, final_t=1, Nsim){
simulated = simulate_mv_merton(Nasset = 2, mu, vol, mu_j, sigma_j, lambda, matrix(c(1,model_corr,
     model_corr,1),ncol=2),
x0=x0, dt=dt, final_t=final_t, Nsim = Nsim)
rho_MC = correlation_MC_estimation(simulated)
rho_MC
}

# Calibrates the model correlation of a 2-asset Merton given the sample corr
calibrate_correlation_merton = function(sample_corr,Nasset=length(mu), mu, vol, mu_j, sigma_j, lambda
     , x0=c(0,0), dt=1/255, final_t=1, Nsim=100){

if (Nasset!=length(mu) | Nasset!=length(x0) |
Nasset!=length(mu_j) | Nasset!=length(sigma_j) | Nasset!=length(lambda)){
stop("Wrong dimension of parameters or initial values.")
}

f_to_be_solved = function(model_corr,  sample_corr, ...){
expected_model_correlation_merton(model_corr=model_corr, ...) - sample_corr
}

res = uniroot(f=f_to_be_solved, interval =c(-1,1),
sample_corr = sample_corr,
mu = mu, vol=vol, mu_j=mu_j, sigma_j=sigma_j, lambda = lambda,
x0=x0, dt=dt, final_t=final_t, Nsim=Nsim,
trace = 3)

return(res$root)
}



# Calibrates the n by n correlation matrix for a n-asset model by iterating on all pairs of assets
calibrate_full_correlation_merton = function(sample_corr_matrix, Nasset=length(mu), mu, vol, mu_j,
     sigma_j, lambda, x0=rep(0,Nasset), dt=1/255, final_t=1, Nsim=500){

if (Nasset!=length(mu) | Nasset!=length(x0) |
Nasset!=length(mu_j) | Nasset!=length(sigma_j) | Nasset!=length(lambda)){
#print(paste(Nasset,length(mu),length(x0),length(mu_j),length(sigma_j),length(lambda)))
stop("Wrong dimension of parameters or initial values.")
}

# create identity matrix
model_corr_matrix = diag(Nasset)

for (i in 1:(Nasset-1)) {
for (j in (i+1):Nasset) {
print(paste0("Estimating element [", i, ',', j,']'))
upper_val = expected_model_correlation_merton(model_corr = 1, mu=mu[c(i,j)], vol = vol[c(i,j)],
mu_j=mu_j[c(i,j)], sigma_j=sigma_j[c(i,j)], lambda=lambda[c(i,j)],
x0=x0[c(i,j)], dt=dt, final_t=final_t, Nsim=Nsim) -sample_corr_matrix[i,j]
lower_val = expected_model_correlation_merton(model_corr = -1, mu=mu[c(i,j)], vol = vol[c(i,j)],
mu_j=mu_j[c(i,j)], sigma_j=sigma_j[c(i,j)], lambda=lambda[c(i,j)],
x0=x0[c(i,j)], dt=dt, final_t=final_t, Nsim=Nsim) -sample_corr_matrix[i,j]
if(upper_val*lower_val < 0){
model_corr_matrix[i,j] =  calibrate_correlation_merton(sample_corr_matrix[i,j], mu=mu[c(i,j)], vol =
     vol[c(i,j)],
mu_j=mu_j[c(i,j)], sigma_j=sigma_j[c(i,j)], lambda=lambda[c(i,j)],
x0=x0[c(i,j)], dt=dt, final_t=final_t, Nsim=Nsim)
}
else{
print(paste("Cannot obtain a correlation of", sample_corr_matrix[i,j] , "with given parameters." ))
if(abs(upper_val)>abs(lower_val)){
model_corr_matrix[i,j] = -1
}
else{
model_corr_matrix[i,j] = 1
}
}
```

```
model_corr_matrix[j,i] = model_corr_matrix[i,j]
}
}

model_corr_matrix = regularization(model_corr_matrix, method = "jackel")

return(model_corr_matrix)
}

# Performs matrix regularization
regularization = function(mat, method =  'jackel'){
decomp = eigen(mat,symmetric = TRUE)
S = decomp$vectors
eigval = decomp$values

# print("Eigenvalues: ")
# print(eigval)

if(method =="jackel"){
eigval[which(eigval<0)]=0

Lambda = diag(eigval)

t = rep(x=NA, length(eigval))
for (i in 1:length(eigval)) {
t[i] = 1/sum(S[i,]^2 * eigval)
}
B = sqrt(diag(t)) %*% S %*% sqrt(Lambda)

res = B %*% t(B)
}
else{
eigval[which(eigval<0)]=1e-5
res = S %*% diag(eigval)%*% t(S)
}
res
}
```

Simulation function for the Bates model (for Heston one only has to set `lambda` to zero):

```

  simulate_mv_bates = function(Nasset=length(mu), mu, k,theta, sigma_V, rho,mu_j=1, sigma_j=0, lambda
     =0, CorrMatrix=matrix(1), S0=rep(1,Nasset), V0, dt=1/255, final_t=1, Nsim=100){

  if (Nasset!=length(mu) | Nasset!=length(k) | Nasset!=length(theta) | Nasset!=length(sigma_V) |
     Nasset!=length(rho) | Nasset!=length(S0) | Nasset!=length(V0) | Nasset!=length(mu_j) | Nasset!=
     length(sigma_j) | Nasset!=length(lambda)){
    stop("Wrong dimension of parameters or initial values.")
  }

  # correlation for the brownian motions driving the 2*Nasset vector (S1,V1,S2,V2, ... ,Sn,Vn)
  full_corr = matrix(ncol = 2*Nasset, nrow = 2*Nasset)
  for (i in 1:Nasset){
  for (j in 1:Nasset) {
  full_corr[2*(i-1)+ c(1,2), 2*(j-1)+ c(1,2)] = correlation_block(rho,CorrMatrix, i,j)
  }
  }

  full_corr=regularization(full_corr)

  Nstep = ceil(final_t / dt)

  sim_x =array(0, dim = c(Nsim, Nstep+1, Nasset), dimnames = list(NULL,NULL,paste0("X_",1:Nasset)))
  sim_V = array(0, dim = c(Nsim, Nstep+1, Nasset),dimnames = list(NULL,NULL,paste0("V_",1:Nasset)))

  for (m in 1:Nasset) {
  sim_x[,1,m] = log(S0[m])
  sim_V[,1,m] = V0[m]
  }

  for (i in 1:(Nstep)) {
  z = mvrnorm(n = Nsim,mu=rep(0,Nasset*2) ,Sigma =full_corr)
  for (m in 1:Nasset) {
  dq = rbinom(Nsim,size = 1, prob= lambda[m]*dt)
  jump = rnorm(Nsim, mean = log(1+mu_j[m]) - 0.5*sigma_j[m]^2, sd = sigma_j[m])
  V_plus= sim_V[,i,m]*(sim_V[,i,m]>0) # positive part for full truncation
  sim_V[,i+1, m] = sim_V[,i,m] + k[m]*(theta[m] - V_plus)*dt + sigma_V[m]*sqrt(V_plus *dt)*z[,2*m]
  sim_x[, i+1,m] = sim_x[,i,m] + (mu[m] - V_plus*0.5 -lambda[m]*mu_j[m] )* dt + sqrt(V_plus * dt)*z
     [,2*m-1] + dq * (jump)
```

```
  }
  }

  res_V = lapply(seq(dim(sim_V)[3]), function(t) sim_V[ , , t])
  names(res_V) = paste0("V_",1:Nasset)
  res_x = lapply(seq(dim(sim_V)[3]), function(t) sim_x[ , , t])
  names(res_x) = paste0("x_",1:Nasset)
  return( list( return = res_x, variance = res_V ))
}
```

E.3 Optimal allocation

E.3.1 Markowitz efficient frontier

Function to compute the efficient frontier without short-selling (just remove the constraint to obtain the frontier to add the possibility to go short):

```
# Function to compute the efficient frontier without shortselling
EfficientFrontier_constr = function(r,S,full=FALSE, N=100, no_short_sales, max_r=NA, min_r =NA){
  # r: expected returns of the assets
  # S: covariance matrix of the asset returns
  # full: computes only upper section of frontier if FALSE
  # N: how many expected returns to take into consideration
  # no_short_sales: vector containing the indexes of the asset for which it is not possible to go
     short

  require(quadprog)

  if (is.na(max_r)){
    max_r = max(r)
  }
  if (is.na(min_r)){
    min_r = min(r)
  }

  # number of assets
  n= length(r)

  D = 2*S       #times 2 because there is a 1/2 in the implicit formulation
  d = rep(0,n)  # zeros

  # Constraint on returns
  A = t(r)

  b = 0.0 # expected return that will be iterated to compute frontier

  # Constraint on weights: sum(w)=1
  A = rbind(A,rep(1,n))
  b= rbind(b,1)

  # Constraint on short selling on given assets
  for (i in no_short_sales){
  A_i = matrix(0,nrow=1,ncol = n)
  A_i[1,i]=1
  A = rbind(A,A_i)
  b = rbind(b,0)
  }

  yy= seq(from = min(r), to = max_r,length.out = N+1)
  yy[1] = yy[1]+ 1e-5
  yy[N+1] = yy[N+1] -1e-5

  b[1]= yy[1]

  xx = rep(0,length(yy))
  for (i in 1:(N+1)) {
    b[1] = yy[i]
    sol = solve.QP(Dmat = D, dvec = (d), Amat = t(A), bvec = t(b), meq = 2)
    xx[i] = sqrt(sol$value)
  }

  if (!full){
    min_sigma = min(xx)
    idx = which(yy>=yy[which(xx==min_sigma)])
```

```
    xx= xx[idx]
    yy= yy[idx]
  }

  res = list(sigma = xx, expected_return=yy)
  return(res)
}
```

E.3.2 CVaR allocation

Function to compute the optimal allocation using CVaR as the portfolio risk measure:

```
# Function to compute the optimal allocation using CVaR
OptimalAllocationDailyCVaR= function(daily_return, alpha, target_return, N_rep=1){
  # daily_return: daily return matrix(N_days * N_assets)
  # alpha: level for the CVaR
  # target_return: target return to be reached
  # N_rep: number of times to compute the result in order to then take an average

  require(alabama)

  N_assets = dim(daily_return)[2]

  expected_asset_return = colMeans(daily_return)

  solution = list(obj = rep(0,N_rep), params = matrix(rep(0,N_rep*N_assets),ncol = N_assets),
     expected_ret=rep(0,N_rep))

  for (i in 1:N_rep) {
  # creating initial random weights summing to one
  sampled =  runif(N_assets)
  rand_initial = sampled/sum(sampled)

  res = auglag(par = rand_initial, fn=daily_ptf_cvar, # objective function
  hin = f_ineq_daily, heq =f_eq_daily, # constraints
  sims=daily_return, alpha = alpha, target_return = target_return, control.outer = list(method = "
     nlminb", trace = FALSE))
  solution$obj[i]=res$value
  solution$params[i,]=matrix(res$par,ncol = N_assets)
  solution$expected_ret[i]=(sum(res$par*expected_asset_return))

  print(paste("Target return:", target_return, "Iteration:", i))
  }

  idx = which.min(solution$obj)

  return(list(objective = solution$obj[idx],
  allocation = solution$params[idx,],
  expected_return = solution$expected_ret[idx]))
}

daily_ptf_cvar = function(w,  sims, alpha, target_return){
  daily_loss = 1 - sims%*%w
  sorted_loss = sort(daily_loss,decreasing = FALSE)
  VaR = quantile(x = daily_loss, probs = 1-alpha)
  idx_var = min(which(sorted_loss >= VaR))
  # print(idx_var)
  CVaR = mean(sorted_loss[idx_var:length(sorted_loss)])
  return(CVaR)
}

# Equality constraints
f_eq_daily = function(w, sims, alpha,target_return){
  # weights should sum to 1
  c1 = sum(w)-1
  N_s = dim(sims)[1]
  # expected return should be the same as target_return
  c3 =  sum(t(sims)%*% matrix(rep(1, N_s),ncol = 1) *w) /N_s - target_return
  return(c(c1, c3))
}

# Inequality constraint
f_ineq_daily = function(w, sims, alpha,target_return){
  # no short-selling, so all weigths should be positive
  return(w)
}
```