LUISS

Department of Economics and Finance *Chair: Econometric Theory*

Stochastic volatility with High-Frequency Data

Analysis of the EuroStoxx index and applications with Julia language

Supervisor

Prof. Giuseppe Ragusa

Candidate

Raffaele Balzano

666241

Co-supervisor

Prof. Pierpaolo Benigno

Academic year 2015-2016

ai miei genitori, a mia sorella

a Marta

ai miei amici e a tutte le persone

che ho incontrato lungo il mio cammino

che mi hanno dato la forza di guardare sempre avanti

CONTENTS

1. INTRODUCTION

This paper's objective is to compare two different volatility frameworks, that are stochastic and non-stochastic (traditional) volatility. Volatility is an essential factor of modern finance theory and modelling, but it cannot be observed. Stochastic volatility provides a framework for the estimation of the time varying volatility. It responds to the need of more complex models in a rapidly changing financial environment. It requires, also, higher computational efforts and solid theory behind. Sometimes the coefficients' estimation is not feasible, and "non-traditional" estimation methods are required. The diffusion of stochastic volatility rose with the improvement of CPUs' computational power and the availability of high-frequency data. High-Frequency Data (HFD) can be used to build more efficient estimators. Data involved is observed at very high-frequencies, usually from few seconds to 30 minutes. It implies that specific methods to handle this huge quantity of data are necessary in order to avoid excessive computational efforts and waste of time. In this thesis the Julia programming language is used, which is very useful in big data analysis. The dataset contains high-frequency (less than one-minute frequency) observations of the EUROSTOXX index. Intra-daily data have been collected to build a daily estimator for the daily unobservable variance process. The aim is to use these estimators to produce (and forecast) a proxy for the volatility of the underlying asset, and use it for trading and risk management purposes.

Stochastic volatility models are typically computationally intensive, still there are several reasons to prefer them to the traditional models (constant volatility or GARCH class). They better fit empirical data, or, as it will be shown, they can be used to standardize daily returns, to obtain a Normal (0,1) distribution. 4 main models are analysed, that are Realized Variance computed with 1-minute frequency (RV1), Realized Variance computed at 5-minutes frequency (RV5), BiPower Variation (BPV) and Two Scales Realized Variance (TSRV). These four estimators are the best representatives for the stochastic variance class of estimators. The first two (RV1, RV5) are the *plain* estimators for the Quadratic Variation (QV) process, that is, the asymptotic estimator of the unobservable volatility process. Two main bias usually affect RV, which are intraday jumps and market microstructure bias. BPV and TSRV try to clean these biases out.

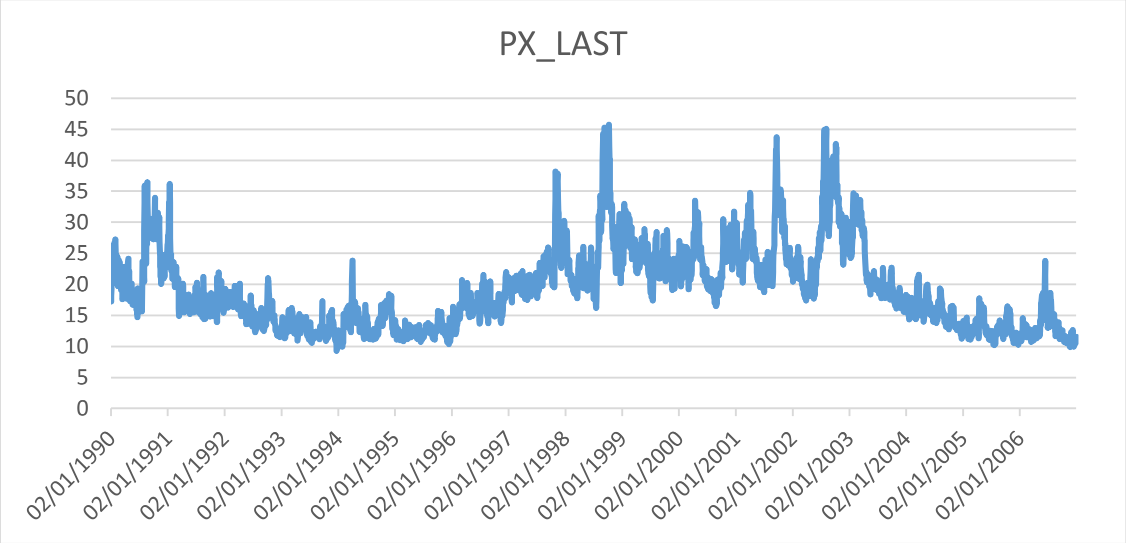
A statistical analysis is then conducted on these variables. Since the aim is to find a good proxy for volatility, which should be able to be "self-forecastable", the analysis is focused on the assessment of the forecasting ability of these variables. Forecasts are implemented with a rolling window procedure. The underlying stochastic process is supposed to follow an AR specification, and, at each forecast horizon, previous one-year observations are used to infer the parameters of the AR model. These parameters are applied to the latest available observations in order to produce a one-step-ahead forecast. Repeating the procedure at all times, a time series array of forecast is produced, and it is then compared with the realized value in order to assess the degree of error. It is introduced also the Heterogeneous AR (HAR) model, which is built with the RV1 moving averages at 1-day, 5-days, 20-days. These horizons correspond respectively to 1-day, 1-week and 1-month trading period. HAR estimator is useful to disentangle the short, medium and long term influence of the variance process. Empirically, this model has higher self-forecasting capacity.

A first application of stochastic volatility is a trading strategy involving the VSTOXX index, that expresses the implied volatility of the EUROSTOXX index. The square root processes of stochastic volatility variables are very similar to that of the VSTOXX. Since they can be used as good proxies for VSTOXX due to the relevant adjusted R-squared, and since they have a significant self-predicting power, it should be possible to make forecasts about the future level of VSTOXX. These models are then compared with a benchmark momentum strategy, that consists into buying the VSTOXX if the index level has risen, and selling otherwise. The preliminary results show that negative returns are achieved using both kind of variables (but RV performs better), due to the unpredictable peaks which frequently occur through the time series. A second strategy consists into adding a mean-reversion momentum signal, to be followed regardless to the value forecasted using the model. With this second method it is possible to achieve positive returns, which only in the case of RV1 estimators are greater than the benchmark strategy.

The last application is a VaR back-testing analysis, comparing a traditional EWMA and stochastic volatility models. The comparison is conducted for a parametric and a Monte Carlo simulation framework, both at 1-day and 10-days horizons. VaR loss level is set to 1%. Models are compared by looking at the exceptions percentage, that is the percentage of times where realized loss is greater than the forecasted VaR. The more this ratio is close to the selected quantile, the better the model is. The parametric model assumes a certain distribution for the standardized time series, and then a certain quantile loss of returns. Theory says that standardizing future daily returns with stochastic variables should account for the leptokurtic bias, and a Normal distribution should be obtained. Empirical evidence shows that the quantile level is not reached, but RV performs better than EWMA. Monte Carlo simulations demonstrate that it is possible to achieve better results with stochastic volatility, if higher-kurtosis distributions for the innovations terms are specified. Using Student-t and the Laplace distributions, indeed, allows to obtain exceptions level very close to the prudential quantile, and much more efficient than traditional methods.

2. LITERATURE OVERVIEW

Over the last decades, literature attempted to develop models to describe assets' returns. A relevant field of research is the identification of the main drivers of returns. It is a widely diffused belief that assets' volatility plays a crucial role in returns composition. At time, it does not exist a theoretical standard model. There are, still, some empirical and theoretical features that are widely accepted. Volatility is time varying, and it manifests *clustering* phenomena. Even if the most of economic models deal with the assumption of a constant volatility, empirical results show that actually large changes are followed by large changes, and small changes tend to be followed by small changes (Mandelbrot, 1963), meaning that volatility is time varying and for long periods it stays almost at the same level. This concept is related to the *mean reversion* phenomenon, namely the attitude of an observed process to come back to its mean level, after a shock has occurred. Figure (1) show how volatility empirically follows these rules.



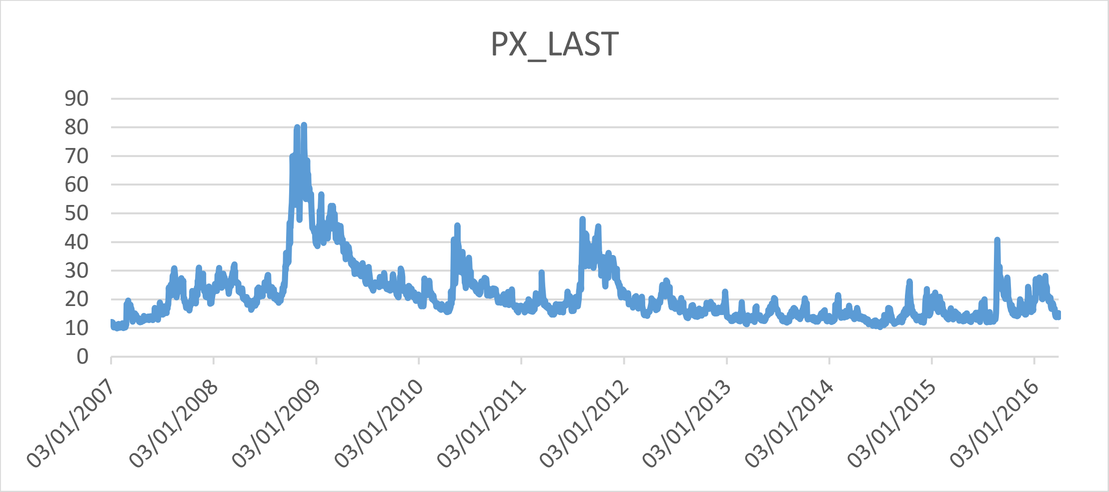


Figure 1a,1b. VIX index between 1990-2006, and between 2007-2016. Source: Bloomberg.

The charts show the level of the VIX (Chicago Board of Exchange Volatility Index), which indicates the 30-day market's expectation of volatility, measured through the implied volatility of plain options on the S&P 100 index. It shows clearly that there are times where volatility stays high for long periods, probably due to shocks that occurred on financial or real market, after which it slowly decreases to its long-term level. The next paragraphs will show the difference between the most common used classes of models: stochastic and non-stochastic.

2.1 Non-stochastic volatility models.

2.1.1 ARCH class models.

The evidence of the phenomena discussed above inspired the development of classes of models that allowed for changes in volatility levels through time. Engle (1982) and Bollerslev (1986) gave important contributions to the time-varying volatility framework with the introduction, respectively, of ARCH and GARCH models, which impose a heteroskedastic (non-constant) structure to volatility. Variance is supposed to be a function of all variables available (observable) until time of analysis: $v_t = f(I_{t-1})$, with I_{t-1} denoting the information set available until then.

Let the return of an asset at time *t* be R_t (it can be considered as the *excess return*, that is the return in excess to risk-free rate). It is possible to define the return process as follows:

2.1

$$R_t = m_t + \xi_t \, , \qquad with\, \xi_t = \sqrt{v_t} \cdot \varepsilon_t$$

where ε_t is a Gaussian White Noise with mean zero and unit variance, m_t and v_t denote the first and the second conditional moments, respectively:

$$m_t = E_{t-1}[R_t] \, , \quad v_t = E_{t-1}[R_t - m_t]^2$$

and ξ_t hence represents the error term of the mean process. Moreover, using this notation, it follows that:

2.2

$$E_{t-1}[R_t] = E_{t-1}\left[m_t + \sqrt{v_t}\varepsilon_t\right] = m_t + \sqrt{v_t}\, E_{t-1}[\varepsilon_t] = m_t$$

2.3

$$Var_{t-1}[R_t] = Var_{t-1}\left[m_t + \sqrt{v_t}\varepsilon_t\right] = v_t\, Var_{t-1}[\varepsilon_t] = v_t$$

where the term v_t goes outside both operators since it is a function of information available at time *t*, thus deterministic.

It is not worth to precise that the "conditional" property of the moments lies in the fact that they are computed using the information available until time *t*. This is pointed out by the subscript of the expectation operator, which can be also written, as $E_{t-1}[X_t] = E[X_{t-1}|I_{t-1}]$. Unconditional moments can be expressed as: $\mu = E[R_t]$ and $\sigma^2 = E[R_t - \mu]^2$. Therewith, ARCH class of models have by construction heteroskedastic conditional volatility, even if this may not exclude a constant unconditional volatility.

Sometimes it is possible to define the variance as the expectation of the squared returns, namely $E_t[R_t^2]$, assuming that returns have zero mean, or re-scaling the returns vector to have a new zero-mean variable: $\tilde{R}_t = R_t - m_t$. This assumption has more support with higher frequency data, where the expected change in returns is negligible.

Coming back to return process, equation (1) states that the realization of the asset's return is made-up by a deterministic component, that is its mean, plus a risky component linked to the unobservable volatility of the asset, which is stochastic since the ε_t is a random variable. It is possible to impose the conditional variance to follow an AutoRegressive (AR) process:

2.4

$$v_t = a_0 + a_1 R_{t-1}^2 + \cdots + a_q R_{t-q}^2$$

It is clear that v_t is a function of past data. Actually, in Engle's paper, the logarithm of v_t is used, but the formulation (4) will be analysed more in detail in the stochastic volatility paragraphs.

Bollerslev (1986) and Taylor (1986) proposed (independently) an extension of the conditional variance that accounted also for lagged values of itself, namely the Generalized ARCH (GARCH):

2.5

$$v_t = a_0 + a_1 R_{t-1}^2 + \cdots + a_q R_{t-q}^2 + +b_1 v_{t-1} + \cdots + b_p v_{t-p}$$

$$with\; a_0 > 0;\; a_i, b_i > 0\; for\; i = 1, \dots\, , \max(q, p)$$

A feature of the GARCH model is that it is able to capture the *clustering* effect, by testing how much the lag-variables parameters are close to 1.

Literature is abundant of different version of ARCH-like models, there are so many that it may be confusing to analyse all of them. Hull (2012) suggested an interesting and simple formulation to monitor the daily volatility, widely used for risk management purpose:

2.6

$$v_t = \lambda\, V_\lambda + a\, r_{t-1}^2 + b\, v_{t-1}^2$$

where r_t denotes the log-return at time t: $r_t = \ln(R_t/R_{t-1})$, and V_λ is the long term volatility (the one that the process is supposed to converge toward). The particularity is that, since λ, a, b are weights given to those variables, it must be true that $\lambda + a + b = 1$. Now, calling $w = \lambda\, V_\lambda$, then it is possible to re-write the model as:

2.7

$$v_t = w + a\, r_{t-1}^2 + b\, v_{t-1}^2$$

that is the formulation of a GARCH (1,1). Then, it is possible to estimate the parameters w, a, b and use the property discussed above to find the long-term variance:

$$\begin{cases} \hat{w} = \hat{l}\, V_l \\ \hat{l} = 1 - \hat{a} - \hat{b} \end{cases} \rightarrow \; V_t = \frac{\hat{w}}{1 - \hat{a} - \hat{b}}$$

Another important reason of the wide use of ARCH-class models, is that, by assuming a distribution for the error terms ξ_t, it is possible to estimate the parameters by maximum likelihood procedure.

ARCH class of models tend to fit quite well data when lower frequencies are analysed. When high frequency data are used these models show some limits, both because volatility may follow intraday patterns and because there may be *noise* within the trades due to market microstructure.

2.1.2 Implied Standard Deviation Models.

From Black-Scholes formula, the price of plain call and put options is a function of stock prices, strike prices, risk-free rate, time and volatility. Since prices of exchanged options are available, it is possible to reverse engineering the Black-Scholes equation to find the volatility value in line with the option prices. Volatility obtained through this method is called *implied volatility*, and represents the market’s expectation about future volatility. Assuming that markets are price-efficient, prices must reflect all information and discounted expectations about future variables. This is also the principle at the base of VIX index calculation. Due to its immediacy, it is very common to use implied volatility as a proxy for near-term volatility. There are several drawbacks in using implied volatility. There may exist risk premia embedded in implied volatility that could deviate from the actual valuation of the option’s price, which lead to biases in the measurement of expected future level of volatility. There may exist unknown variables that are not priced, whose price is erroneously embedded in implied volatility.

2.2 Stochastic volatility models.

The models so far described were characterized by time-varying but deterministic variance. Stochastic-volatility models, instead, introduce “randomness” elements in modelling the variance. Variance is supposed to follow a general stochastic process, $\{V\}$, that is a series of stochastic variables indexed by time: $V_1, V_2, \dots V_t, \dots V_T$. Each random variable V_t, characterized by a probability density function, and, together with the other “members” of the process, by a joint probability density function. In real world just a realization of the process is observed.

Since 80’s, many authors proposed models where variance was imposed to follow a stochastic process. Stochastic volatility models allow for the presence of shocks in both prices and volatility. Volatility is a function of a certain set of variables $\sigma_t = f(K_t)$, where K_t is an unobserved latent process, and $f(\cdot)$ is an increasing function whose codomain is the set of non-negative real numbers. K_t may follow a particular process, such as an ARMA(p,q), a Random Walk, or even a continuous time process, such a Brownian Motion. One of the first adopter of these models structure was P. K. Clark, who analysed the price of traded securities and introduced the concept of “subordinated process” (Clark 1973). Instead of referring to a price process as the sequence of random variables $P_1, P_2, \dots, P_T$, with $t = 1,2,\dots,T$ representing the discrete time at which the realizations of the price comes, Clark assumes that time itself follows a particular stochastic process, whose t is a realization. Hence, t is the realization of a stochastic process, call it $\{\tau\}$, with positive increments: $\tau_i > \tau_j$ for $i > j$. The process $P(\tau_t)$ can be thought as the realization of the quoted price on a trading platform: trades do not happen at periodic intervals of time, but their frequency depends on the activity of buying and selling of operator at market microstructure level. Literature developed several techniques for handling stochastic volatility. The following paragraphs will provide a first insight towards these models, while in the fourth chapter high-frequency stochastic volatility models will be discussed.

2.2.1 *Discrete time models.*

A first sub-class of models are discrete time ones. Taylor (1982) provided oe of the first formulation of discrete time model, namely the *product process*. He started from an expression of returns similar to (1):

2.8

$$r_t = E[r_t] + \sigma_t u_t$$

where r_t is the log-return $ln(R_t/R_{t-1})$, σ_t and u_t are two independent stochastic processes such that: σ_t is strictly positive, $\{u\}$ follows an ARMA(1,1) process with mean zero and unit variance, σ_t and u_s independent for each t and s. If the assumption on the stationarity of $\{\sigma\}$ $\{u\}$ holds, then also $\{r\}$ is stationary. These properties imply *in primis* that:

$$Cov(\sigma_t, u_s) = E[\sigma_t u_t] - E[\sigma_t]E[u_s] = 0 \ \rightarrow \ E[\sigma_t u_t] = E[\sigma_t]E[u_s]$$

which implies:

2.9

$$E[r_t - E[r_t]] = E[\sigma_t u_t] = E[\sigma_t]E[u_s] = 0$$

2.10

$$E[r_t - E[r_t]]^2 = E[\sigma_t u_t]^2 = E[\sigma_t]^2 E[u_s]^2 = E[\sigma_t]^2$$

The variance of returns is the expected value of the squared term σ_t, and thus the expected variance. The process $\{\sigma\}$ is assumed to be modelled as:

2.11

$$\sigma_t = e^{h_t/2}$$

where h_t can be a generic non-zero mean Gaussian linear process. In this particular case it is an AR (1) process:

2.12

$$h_t = a_0 + a_1 h_{t-1} + \varepsilon_t$$

where ε_t is *Gaussian White Noise* with mean zero and variance $\sigma_\varepsilon{}^2$. The term ε_t is exactly the element that distinguish stochastic and non-stochastic models (for example, the one described in equation (4)). This implies that the logarithm of variance is modelled as an AR (1):

$$\ln(\sigma_t) = h_t/2 \ \rightarrow \ln(\sigma_t^2) = h_t$$

2.13

$$\ln(\sigma_t^2) = a_0 + a_1 \ln(\sigma_{t-1}^2) + \varepsilon_t$$

In case u_t is also normal, the process (8) take the name of *log-Normal stochastic volatility model*. The main advantage of using a log-model is that it ensures non-negative values for variance.

Let now analyse some statistical properties of the variance process (13). Its expected value is:

$$E[\ln \sigma_t^2] = E[a_0 + a_1 \ln \sigma_{t-1}^2 + \varepsilon_t]$$

$$= a_0 + a_1 E[\ln \sigma_{t-1}^2]$$

that, by recursively substituting, becomes:

$$= (\sum_{i=0}^{n-1} a_0 a_1^i) + a_1^n E[\ln \sigma_{t-n}^2]$$

Assuming that $|a_1|<1$, which means that the series is stationary, when $n \rightarrow \infty$ the last part of the equation goes to zero, while the first part converges to the sum of a geometric series:

$$= \frac{a_0}{1 - a_1} = \alpha$$

Literature often do not agree with the assumption that *n* goes to infinity, but rather prefer to bound the process within a finite time space. It is possible to specify an initial time t_0, at which the process starts, and then the above expression becomes

$$= (\sum_{i=0}^{N=t-t_0} a_0 a_1^i) + a_1^N E[\ln \sigma_0^2]$$

Supposing that σ_0 is a known value, then the expected value of (13), conditioned on σ_0, becomes

$$= a_0 \frac{1 - a_1^{N+1}}{1 - a_1} + a_1^N \ln \sigma_0^2$$

The variance of (13), knowing that σ_t^2 and ε_t are uncorrelated, is:

$$Var[\ln \sigma_t^2] = Var[a_0 + a_1 \ln \sigma_{t-1}^2 + \varepsilon_t]$$

$$= Var[a_1 \ln \sigma_{t-1}^2 + \varepsilon_t]$$

$$= a_1^2 Var[\ln \sigma_{t-1}^2] + Var[\varepsilon_t]$$

$$= a_1^2 Var[\ln \sigma_{t-1}^2] + \sigma_\varepsilon^2$$

Then, as above, by iteratively substituting:

$$= \frac{\sigma_\varepsilon^2}{1 - a_1^2} = \beta^2$$

Since the series is stationary, these moments are the same at each lag. The covariance and the autocorrelation of the series are:

$$Cov(\ln \sigma_t^2 , \ln \sigma_{t-h}^2) = Cov\left(\sum_{i=0}^{h-1} a_1^i \, (a_0 + \varepsilon_{t-i}) + a_1^h \ln \sigma_{t-h}^2 , \ln \sigma_{t-h}^2 \right)$$

$$= Cov(a_1^h \ln \sigma_{t-h}^2 , \ln \sigma_{t-h}^2)$$

$$= a_1^h Var(\ln \sigma_{t-h}^2) = a_1^h \beta^2$$

$$\rho(\ln \sigma_t^2 , \ln \sigma_{t-h}^2) = \frac{Cov(\ln \sigma_t^2 , \ln \sigma_{t-h}^2)}{Var(\ln \sigma_t^2)}$$

$$= \frac{a_1^h \beta^2}{\beta^2} = a_1^h$$

From equation (13) since ε_t is normally distributed, also $\ln \sigma_t^2$ is normally distributed, with mean α and variance β^2:

$$\ln \sigma_t^2 \sim N(\alpha, \beta^2)$$

and then:

$$\ln\sigma_t = \frac{1}{2}\ln\sigma_t^2 \sim N(\frac{\alpha}{2},\frac{\beta^2}{4})$$

There are, now, sufficient elements to calculate moments of r_t. Re-writing $r_t = u_t e^{\ln\sigma_t}$ and combining equation (9) with the properties of the Normal distribution u_t, it follows that even moments of r_t are all zeros. In general, the *n-th* moment is:

$$E\big[r_t - E[r_t]\big]^n = E[u_t]^n E[\sigma_t]^n = E[u_t]^n E[e^{n\ln\sigma_t}]$$

$$= E[u_t]^n e^{n\alpha/2+\frac{1}{2}n^2\beta^2/4}$$

thus, variance equals:

$$= e^{\alpha+\beta^2/2}$$

and, kurtosis is:

$$= kurt(u_t)\frac{E[(\sigma_t)^4]}{E[(\sigma_t)^2]^2}$$

$$= 3\frac{e^{2\alpha+2\beta^2}}{e^{(\alpha+\beta^2/2)2}} = 3e^{\beta^2}$$

which, for β^2 positive, is greater than the kurtosis of a standard Normal distribution. The log-model accounts also for fatter tails of the returns distribution, as empirical results confirm.

Inference with common statistic tools, *i.e.* by maximum likelihood, is infeasible. There is not a closed-form solution for the likelihood function. The density of r_t is:

$$CDF(r_t) = \int_{-\infty}^{+\infty} PDF(r_t|\sigma_t^2) * PDF(\sigma_t^2|\alpha,\beta^2)d\sigma_t^2$$

where *PDF(r_t)* is a Normal density function, and *PDF(σ_t^2)* is a LogNormal density function. Therefore, the integral has no closed-form, and it can be evaluated only numerically, through simulations. There are alternative tools used to make inference in stochastic volatility framework, and they will be the subject of the next chapter.

2.2.2 Continuous time models.

Continuous time models define the diffusion law both for asset's price and for asset's volatility. The most common approach is to use a Brownian Motion as law of diffusion. In the simplest case, the risky part of return of equation (1) is given by:

2.14

$$M_t = \int_0^t \sigma_s \, dw_s$$

that is, a stochastic integral where w_t is a Brownian Motion process. A Brownian Motion $\{w\}$, or Wiener Process, is a stochastic process defined on a continuous time space with the properties:

- $w(t + \Delta t) - w(t)$ is Normal distributed with mean zero and variance Δt;
- Its increments are independent;
- It is a *martingale*, in the sense that future realization does not depend at all on past observations, but only on current one. M_t is a martingale if also the square root of the integrated variance, $E[\sqrt{\int_0^t \sigma_s^2 \, ds}]$, is a finite quantity.

The Brownian Motion can be thought as a Random Walk model in a continuous time space. A widely used formulation for the Brownian motion is: $dw = \varepsilon\sqrt{dt}$. Equation (14) has a particular appeal in financial modelling, since, taking the square of the M_t process, knowing that $dw \cdot dw = dt$, it yields:

$$IV = \int_0^t \sigma_s^2 \, ds$$

which takes the name of *integrated volatility*. The integrated volatility can be thought as the sum all the spot, *i.e.* instantaneous, variances over time. As pointed out by Barndorff-Nielsen and Shephard (2004), the integrated volatility can be estimated through the quadratic process of returns, which will be deepened in the following chapters. This estimate, in theory, is much more accurate when the time intervals of observations tend to zero, that is with high-frequency data.

Hull and White (1987) were pioneers of continuous time stochastic volatility. They started from Black-Scholes-Merton diffusion process, used in option pricing, where variance process was the solution to the differential equations:

2.15

$$dS_t = \varphi(S, \sigma, t)\ S_t\ dt + \sigma_t\ S_t\ dw$$

$$d\sigma_t^2 = \mu(\sigma, t)\ \sigma_t^2\ dt + \omega\ \sigma_t^2\ dz$$

where *dw* and *dz* are Brownian Motion processes with a correlation *ρ*, and the drift rate *φ* becomes the risk-free rate in a risk-neutral framework. The correlation between shocks in variance and shocks in prices, makes the model closer to reality. Empirically, financial markets, especially equity markets, show high volatility when high change in price occurs (*e.g.* after a dividend announcement). The parameter *μ* can be set such to take into account mean-reverting phenomenon. It is the case of Heston (1993) or Melino and Turnbull (1990), who proposed a continuous-time version of the Taylor logarithmic stochastic volatility. The second equation of (15) is, indeed, replaced by:

2.16

$$d \ln \sigma_t = \mu \cdot (v - \ln \sigma_t)\ dt + \omega\ dz$$

such that volatility is ensured to be positive, and tends to its long-term level *v*. The solution to this stochastic differential equation, expresses the value assumed by *ln(σ$_t$)*. It is equal to:

2.17

$$\ln \sigma_t = \int_0^t \mu(v - \ln \sigma_s)\ ds\ + \int_0^t \omega\ dz_s$$

$$\ln \sigma_t = v + (\ln \sigma_0 - v) e^{-\mu t} + \omega \int_0^t e^{-\mu(t-s)}\ dz_s$$

Conditional moments are:

$$E[\ln \sigma_t \mid \ln \sigma_0 = k_0] = E[v + (\ln \sigma_0 - v)e^{-\mu t} + \omega \int_0^t e^{-\mu(t-s)}\ dz_s]$$

$$= v + (\ln \sigma_0 - v)e^{-\mu t}$$

$$Var[\ln \sigma_t \mid \ln \sigma_0 = k_0] = Var[\omega \int_0^t e^{-\mu(t-s)}\ dz_s]$$

$$= E[\omega \int_0^t e^{-\mu(t-s)}\ dz_s]^2$$

By Itô's isometry, which states that $E[\int_0^t X_s\ dz_s]^2 = E[\int_0^t X_s^2\ ds]$, it yields:

$$= \omega^2 E[\int_0^t e^{-2\mu(t-s)}\ ds] = \frac{\omega^2}{2\mu}(1 - e^{-2\mu t})$$

Equation (17) is also denoted as *Itô Process* or *stochastic integral* form. The volatility process shown in (16) and (17) follows an *Ornstein-Uhlenbeck* process, also called *Gauss-Markov* process, which has the following properties:

- it is stationary, meaning that the multivariate distribution of the process at different lags do not change:

  $$P(\ln \sigma_t , \ln \sigma_{t-1} , \dots , \ln \sigma_{t-s}) = P(\ln \sigma_{t-h} , \ln \sigma_{t-h-1} , \dots , \ln \sigma_{t-h-s}\ , \forall h > 0$$
- it is Gaussian, meaning that the multivariate distribution of the process is normally distributed;
- it is Markovian, meaning that the distribution of future variables depends just on the most recent distribution and not by the older ones:

  $$P(\ln \sigma_{t+1} \mid \ln \sigma_t , \dots , \ln \sigma_{t-s}) = P(\ln \sigma_{t+1} \mid \ln \sigma_t)\ \forall s > 0$$
- it is continuous in probability, meaning that the distribution of adjacent variables in time is almost the same:

  $$P(\ |\ln \sigma_t\ - \ln \sigma_{t-\Delta}| > \varepsilon) \rightarrow 0, as\ \Delta \rightarrow 0, \forall \varepsilon > 0$$
- it is mean-reverting toward the long-term average *v*, with a rate given by μ.

Other widely used continuous time models are the Heston (1993) model, where the processes followed by the volatility is:

$$d\sigma_t^2 = \mu(v - \sigma_t^2)\ dt + \omega\ \sigma_t\ dz$$

and the *jump diffusion*, introduced by Bates (1996). The last model accounts for discontinuous change in diffusion process both of the asset and the volatility structure. It reflects the impact of news or significant shocks in financial market, whose intensity cannot be explained just by Brownian Motion. Bates added to the Brownian Motion diffusion process, another diffusion term modelled on a Poisson distribution, $k_t dq_t$, which is function of the annual frequency of jumps and the percentage jump, k_t, given a jump has occurred. Moreover, q_t is a counting process with intensity λ_t, such that $P(dq_t = 1) = \lambda_t dt$. The asset's stochastic differential equation become:

$$dS_t = \varphi_t \, S_t \, dt + \sigma_t \, S_t \, dw + k_t \, S_t \, dq_t$$

This, in practice, makes computations harder, but empirical tests have shown that jump diffusion models make slight improvement to the time series analysis of prices.

3. HIGH FREQUENCY DATA

This chapter will be focused on the main features of high-frequency data (HFD). HFD can enhance the standard tools of estimation, since they provide a better approximation of continuous-time processes. As Clark (1973) states, the use of HFD necessarily implies to deal with stochastic time space of observation. There are also some practical issues to take into account in analysing HFD.

3.1 Data handling

The current development of infrastructure technology system and the increase of computing power, has made possible for trading venues to collect data at the minimum time interval, *i.e.* at every *tick*. Such a huge availability of data presents issues regarding the process of gathering information, and raw data cannot be used *as are*.

3.1.1 Data Cleaning

Falkenberry (2002) indicated that HFD collecting process may present errors of "transcription", and the frequency of these inaccuracies rises as the frequency of data becomes higher. There may exist at least two kind of errors. There are "human-driven" errors, which can be unintentional, *e.g.* typos, or intentional, *e.g.* an algorithmic trading strategies which post and immediately cancel massive amounts of orders, usually at non-reasonable prices, thus creating noise and false quotes. There are non-human errors, those created by the electronic infrastructure and by the gathering data process. Typical examples are errors of transposition or the loss of some part of the data, such as the decimal part. However, in practice it may be difficult to determine whether a suspicious observation is an error or not, or to identify the cause of the outliers. For instance, it may happen, analysing some minute-by-minute data, to find a minute return of 1%, which may sound relatively high. There could be several reasons for this sudden change in price, it may be due to errors as much as an unexpected announcement that has been absorbed by market. It is important to analyse the outliers, and possibly to look for the cause.

A first *naïve* technique of outliers' recognition is based on return magnitude. This involves to define a certain threshold of returns. If exceeded, the observation is labelled as "suspicious" and eventually discharged. The problem is the right choice of the threshold. It has to depend on the frequency of data, but not in a linear manner. It is foregone that the threshold for daily return should be bigger than an hourly return, but not in a linear way. There may be intraday patterns during a trading day where the price may vary more than the resulting daily return, which is computed just on the opening and closing prices. For instance, the probability of an hourly return exceeding the 10% is higher than the probability of a daily return exceeding the 80% (10% times 8 hours). In formulas: $P(r > threshold) < P(r/t > threshold/t)$. A rapid way to determine thresholds is to choose an appropriate quantile of returns. It can be set the threshold such that the probability of encountering a greater observation is, for example, 99.99%. It ca be added a fixed amount of basis points, whose amount depends on the frequency of data, and then discharge all observation greater than the obtained value.

Another possible solution, that does not involve the use of thresholds, is to find adjacent sequences of anomalous returns with opposite sign. Supposing there is a mistake in the price sequence, it is reasonable to expect that the next observation will turn back to the correct level. The return sequence should display an anomalous value followed by another anomaly of almost the same intensity but different sign. Problems may arise when these errors come in sequences, since it becomes hard to distinguish errors from a temporarily jump of price.

Brownlees and Gallo (2006) propose a detection technique, which relies on the statistical properties of neighbouring prices. An outlier is identified if it exceeds the distance from the trimmed mean of a neighbourhood of k prices by three standard deviations, plus a parameter γ which accounts for a lower bound in case of non-changing quotes. *Trimmed* moments are computed by selecting the k previous and the k following prices. Observation are outliers if:

$$|p_t - \overline{p_t}(k)| > 3\sigma_t(k) + \gamma$$

with $\overline{p_t}(k)$ and $\sigma_t(k)$ denoting the trigged mean and standard deviation. The choice of k must be inducted by the frequency of data. Authors conclude their paper stating that it is necessarily a graphical analysis of the suspicious data. Using the standard deviation in presence of outliers may bias the results, since measure like mean and standard deviation suffer for the presence of outliers. Sometimes it is better to recur to the median value. In the formula above, the standard deviation can be replaced by the median, with an opportunely calibrated parameter. A similar approach is to use the *median absolute deviation* (MAD), which is the median of the absolute deviation from the daily median. Hellerstein (2008) proposes to consider as outliers those observations which, standardized, exceed 2.9652 x MAD, that roughly corresponds to two standard deviations of a Normal distribution.

Other studies, such as Chung, Van Ness and Van Ness (2004), show how the price level has an important effect in the "mis-classification" of non-outliers. For low priced securities even a relative small change in price is able to produce a significant return. For example, a 1$ security which rise to 1.5$ had a 50% return, and it is not unlikely to happen. The threshold of returns has to take into account also the effect on low priced securities.

Finally, it is possible to use machine-learning techniques to combine all these algorithms for the search of outliers. For example, the AdaBoost algorithm, may be efficient in presence of many "weak learners", *i.e.* not efficient classifiers, where a unique "strong learner" is built up by giving appropriate weights to the former ones.

3.1.2 Missing data

It is very frequent to collect HFD with some missing value, especially if collected by small providers or on illiquid markets. Usually these missing data are labelled with a "NA" value, which may cause significant issue in data analysis. Missing data may be caused also by low trading activity. A first immediate approach is to fill up the missing price between two available ones, by interpolating the missing values with an average of available prices, weighted by time. The price at time *t*, with observations available at t_i and t_{i+1} is:

$$\dot{p}_t = (1-\omega)p_i + \omega p_{t+i} \quad \rightarrow \quad \dot{p}_t = p_i + \omega(p_{t+1} - p_t)$$

$$with\, t_i < t < t_{i+1}, \quad and \quad \omega = \frac{t - t_i}{t_{i+1} - t_i}$$

This method is invariant in mean, since produces information that are linear combination of available data, but not in variance. Adding observations with value within the range already measured, a new set of variable is created, but with less “dispersion”. The longer the interval of data is, the more intra-pattern informations are excluded, and the more variance is underestimated. Forgetting for a while of Brownian Motions and price patterns, if a time interval *[0, T]* is divided into *n* subset of equal length Δt, it is true that

$$var_{\Delta t}(P) = \Delta t^2 var_T(P) = \frac{1}{n^2} var_T(P)$$

where, $var_{\Delta t}$ is the variance over the small interval of time Δt, while var_T is the variance over the whole period with just observation at time zero and time *T*. It means that the variance computed adding interpolated observation is smaller than the variance computed with just the initial values. A possible solution could be to impose that the price between two observations move proportionally to the square of the time elapsed:

$$\dot{p}_t = p_i + \sqrt{\omega}(p_{t+1} - p_t)$$

This interpolation method implies to renounce to the mean “insensibility” of the interpolation process. According to the object of the analysis, if the mean or the variance, it is preferable to use one interpolation method or the other. The interpolation method necessarily increases the serial correlation between price sequences. Possible drawbacks are the risk that the autocorrelation series tends to the unit, with problems of stationarity and linear estimation.

Another common issue is the data aggregation when passing to lower frequencies. This procedure necessarily implies a loss in data dispersion (loss of intra-period patterns), which translates into lower variance. This is consistent with the inequality *P(r > threshold) < P(r/t > threshold/t)* presented above. Data aggregation necessarily means loss of information.

In conclusion, there is the trade-off between reducing the frequency of observations and have more treatable data in computational terms, at the cost of losing intra-period information, or maintain high frequency data that needs higher computational efforts and may present noise or biases.

3.1.3 Data synchronization

In the previous chapter it has been discussed how, at higher frequencies, observations happen at irregular interval of time. It becomes hard to analyse together two price processes whose realizations happen at different timing. A first solution is to take just observation with same timing: $\{t\} = \{t_i\} \cap \{t_j\}$. Problems arise with more processes at time, where there is the risk of losing a significant number of information. This is the case of illiquid markets or high frequencies observations. In the latter case, the time set can be seen as a continuous space, thus it becomes hard to take just the common values without the risk of creating gaps. The best scenario would be to create a minimum common discrete interval of time, and then fill each point with an observation. Missing points may be filled with procedures described in the previous paragraph.

3.1.4 Market microstructure noise

When dealing with HFD a common issue is the noise embedded in market microstructure. Each trading platform consist in a *trading book*, which collects all the orders entered into the system by traders, divided into *bid orders* (orders to buy) and *ask orders* (orders to sell). Each trader can submit a *limit order*, that is an order of buying or selling at a specified price; if there is a counterparty, then the order is immediately executed, otherwise it stays in the book, alongside the other unfulfilled orders. When an order arrives, it is executed at first the “best-quoted” order, which is the highest bid within the trading book for a sell order (hit), and the lowest ask for a buy order (lift). If more traders put the same quote, it is executed firstly the oldest one (price time priority). When a trader puts a limit order that can immediately be executed, it is filled until the price is still favourable and the quantity is satisfied. The non-executed part of the order stays in the book until an opposite order comes. Traders can submit also *market orders*, which are executed immediately, whatever the price. This is the main reason of the presence of noise in data. If the trading book is not very liquid, that is when quantities are low and the difference of adjacent quotes is significant relatively to orders arrival, then a sufficient high quantity market-order may significantly shift the last observed (last traded) price. Then, market participants may fill again the resulting gap of quotes with new quotes, restoring the “fair” price of the security. This phenomenon was labelled by Roll (1984) as the *bid-ask bounce*. This situation may happen for several reasons: perhaps who made the market order does not care to hit less favourable quotes, or perhaps there may be tricky algorithms that may cause this situation. It is emblematic the case of the *flash crash*, where on 6th May 2010, within a couple of hours, a misleading algorithm caused the S&P500 to lose about 9%, which soon recovered almost all the losses.

Market microstructure noise emphasize the trade-off about the optimal frequency of data to use. If higher frequencies data structure contains irrational patterns, the results of estimation may be biased, depending on the impact of this noise. On the other side, only by mean of HFD, the estimation of continuous-time processes is possible. In conclusion, if possible, market microstructure noise should not be taken into account, since it does not reflect the *fair* price of the traded security. Some data provider, usually, provide information of the *mid* quote, which may be reasonably better proxy for data analysis.

Market microstructure noise is one of the main drawback of using HFD. Literature proposes different solutions. Aït-Sahalia, Mykland and Zhang (2005a) propose to not care about the noise at very high frequency. Other authors, instead, propose to use alternative robust estimators or techniques, such as the pre-averaging (Jacod et al., 2009), multiscale (Zhang, 2006; Aït-Sahalia *et al.*, 2005b) and the realized kernel estimator (Barndorff-Nielsen et al., 2008).

3.1.5 Ticks frequency effect

Falkenberry (2002) notes how parameters of filtration methods need to be adapted to the several securities, according to the ticks' frequency. He showed that stocks with higher ticks' frequency, which reveal to be those with higher market capitalization and volume, are more subject to errors. This implies that securities with higher tick frequency need a filtering algorithm that focuses on speed of calculation, since the number of observation is much higher. Securities exchanged at lower frequency, on the other side, allow for more tolerance in price movement, due to the greater time between adjacent ticks.

3.1.6 Intraday patterns

Typically, the daily volume and the ticks' frequency, for exchanged securities in regulated markets, show a U-shaped pattern, meaning that the most of transaction happens at the beginning and at the end of the trading day. This effect is more marked in securities with higher ticks' frequency, probably because, at opening, traders "discount" all overnight information received during the non-trading hours, while at closing, they prefer to close some open positions. This may explain why this effect is more marked with higher market cap firms, since they are probably multinational company and thus they are affected to news and shocks from other part of the world. This implies higher level of volatility at the beginning and at the end of a trading day, which requires setting the filtering algorithm such to take into consideration this possibility of higher change in price during these hours.

3.2 Econometrics of High Frequency Data

Having HFD in econometrics is a big advantage, since it is possible to use sample observations to produce accurate estimations of true parameters, through asymptotic theory.

At higher frequencies, the structure and the behaviour of data may be significantly different. Engle and Russel (2004) showed how, at higher frequencies, correlation structure assumes more relevance. Analysing data at microstructure level makes possible to notice a substantial negative autocorrelation between quotes, due to the *bid-ask bounce* effect. Moreover, positive correlation is found at higher lags, due to traders behaviour that prefer to split the order in small quantities, to have a lower impact on the price.

An important application of HFD involves the volatility estimation. In models presented in the first chapter, volatility is treated as a hidden variable to be modelled as a particular stochastic processes. When time between observations tend to zero, it is almost sure that the instantaneous volatility can be captured, or, at least, a close proxy may be computed.

4. VOLATILITY IN HIGH FREQUENCY DATA FRAMEWORK

The model that will be studied is the continuous time process:

4.1

$$dp_{\tau_t} = \mu_{\tau_t} dt + \sigma_{\tau_t} dW_{\tau_t}$$

whose solution is:

$$p_{\tau_t} = \mathrm{M}_{\tau_t} + \int_{\tau_0}^{\tau_t} \sigma_s dW_s$$

with *{p}* the log-price, *{μ}* a generic function denoting the drift rate, *{M}* its integrated value, and *{σ}* the spot volatility of the log-price. The time space *{τ}* is itself a random variable. Assuming that *{μ}* and *{σ}* are independent from the Brownian Motion *{W}*, the log-price instantaneous difference, *i.e.* the instantaneous return, distributes as:

$$dp_{\tau_t} = r_{\delta_t} \sim N(\mu_{\delta_t}, \sigma_{\delta_t}^2)$$

$$with \ \ \mu_{\delta_t} = \int_{\tau_{t-1}}^{\tau_t} \mu_t ds, \quad \sigma_{\delta_t}^2 = \int_{\tau_{t-1}}^{\tau_t} \sigma_t^2 ds$$

The daily drift component *μ* can be consistently estimated with just opening and closing day quotes. It can be also assumed, without loss of generality, to be constant or even zero. The daily *σ* component is unobservable.

4.1 Some definitions

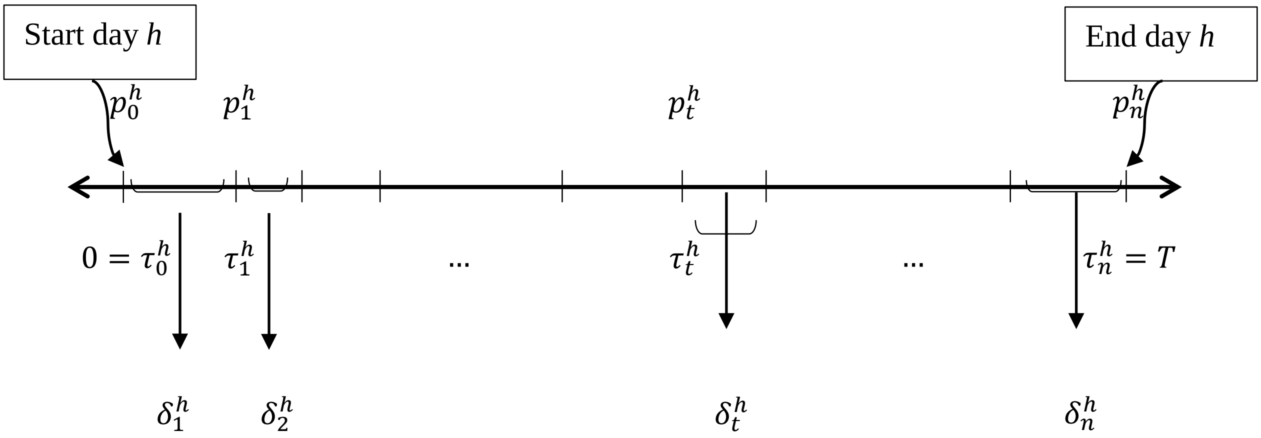
4.1.1 Terminology.

To avoid confusion and abuse of terminology, the basic framework will now be provided once for all. Since the time space analysed is irregular, it will be denoted as the process *{τ}*. The analysis will be conducted on intraday data, on the *h-th* trading day. Each day will be characterized by a starting time *0* and an ending time *T*. The trading day will be divided into the smallest possible time interval, coincident with the time-stamp of the dataset, *i.e.* the time of available observations. Since the time space is a random process, the interval of time *{δ}* is also random. The time space within the trading day is divided into *n* "irregular" sub-interval. Loosely speaking, if the time intervals were all equal, then it would be true that *δ=T/n*. Putting all together:

$$time\ space\ on\ the\ h^{th}\ day: \{\tau(h,t)\} = \{0 = \tau_0^h, \tau_1^h, \dots, \tau_n^h = T^h\}$$

$$= \{\tau_t^h\}, \qquad t = 0, 1, \dots, n$$

$$T^h = \sum_{t=1}^{n} \delta_t^h$$



For the following treatment the index *h* will be dropped, unless confusing. Moreover, to avoid too many subscripts, variables occurring at a specific time will be denoted as follows:

$$p_{\tau_t}^h \overset{\text{def}}{=} p_{h+\tau_t} : \rightarrow p_t$$

4.1.2 Martingales, local martingales, semi-martingales

A martingale M_t is a stochastic process such that the expected value of the future outcomes, given the current set of observation, is equal to the current value:

$$E[M_{t+k}|I_t] = M_t, \quad k \geq 0$$

The random walk theory is a common example of martingale. The Brownian Motion is another example of a martingale. From stochastic calculus, the integral of a bounded process *{X}* whose integrand term is a martingale, *e.g.* a Brownian Motion, is itself a martingale:

$$M_t = \int_0^t X_s dW_s$$

A local martingale, loosely speaking, is a stochastic process that is locally a martingale, meaning that there exist a series of time, called *stopping time*, subset of the whole time space, where the process behaves as a martingale. Finally, a semi-martingale is a composition of a local martingale *{M}* and a finite variation process *{A}*:

$$SM_t = M_t + A_t$$

The classic Itô's formulation of log-price process (4.1) is an example of semi-martingale, with *{A}*=μ *dt* and *{M}*=σ *dW*.

4.1.3 Quadratic variation (QV)

By Doob's decomposition theorem, given a martingale process *{M}*, its quadratic variation ⟨M⟩ is the unique increasing process such that $\langle M \rangle_0=0$ and the process $\{M^2-\langle M \rangle\}$ is still a martingale. For a continuous-time semi-martingale process *{X}*, such as log-price processes, the quadratic variation assumes the form of:

$$\langle X \rangle_t = X_t^2 - 2\int_0^t X_s dX_s$$

The QV process may also be defined on the discrete time space τ, and, in this case, this variable takes the name of *realized variance*. Thanks to Protter (2004) developments, it can be shown that:

$$RV_t \stackrel{\text{def}}{=} \langle X \rangle_t^\tau = \sum_{0<s<t} (X_s - X_{s-1})^2$$

$$\langle X\rangle_t = \mathrm{p}-\lim_{\sup\{\delta\}\to 0} \langle X\rangle_t^\tau$$

$$\Rightarrow QV = p - \lim RV$$

Loosely speaking, the QV process can be considered as the squared process of a variable. Recalling the equation (4.1) for a continuous time stochastic process for the security log-price:

$$dp_t = \mu_t dt + \sigma_t dW_t$$

by squaring both side of equation, it yields to:

$$(dp_t)^2 = \mu_t^2 dt^2 + \sigma_t^2 dW^2 + 2\mu_t \sigma_t dt dW_t$$

and recalling by Itô’s calculus that dt^2 and $dtdW$ tend to zero faster than dW^2, which is of order *dt*, the first and the third term of the equation can be dropped as tend to zero, and it is left that:

$$(dp_t)^2 = \sigma_t^2 dt$$

Integrating both part, it becomes true that:

$$\int_0^t (dp_s)^2 = \int_0^t \sigma_s^2 ds$$

and, recalling that the integral is the limit of the sum as the integrand term tend to zero:

$$\lim_{\sup\{\delta\}\to 0} \sum_{t\in\{\tau\}} \left(p_{t+\delta_t} - p_t\right)^2 = \int_0^t \sigma_s^2 ds$$

$$\lim_{\sup\{\delta\}\to 0} \langle p\rangle_t^\tau = \int_0^t \sigma_s^2 ds$$

$$\mathrm{QV_t} = \mathrm{IV_t}$$

which implies that the integrated volatility is equal to the quadratic variation, and therefore, it can be estimated consistently with RV and HFD. The power of QV is that it provides a consistent estimator of true (integrated) variance without knowing the behaviour of μ or σ. The IV can be seen as a part of the return process. Indeed, the solution to the differential equation can be written as:

$$p_T - p_0 = r_T = \int_0^T \mu_t dt + \int_0^T \sigma_t \, dW_t$$

Recalling that *dp_t* is equal to the *t-th* return, QV process (and also IV) can be estimated by the daily sum of the squared high-frequency returns.

Barndorff-Nielsen and Shephard (2002) defined an asymptotic distribution for the daily QV process, as the time interval δ tend to zero, *i.e.* the number of subsamples *n* goes to infinity:

$$\sqrt{n}\left(\sum_t^T r_t^2 - \int_0^T \sigma_s^2 ds\right) \xrightarrow{d} N\left(0, 2\delta \int_0^t \sigma_s^4 ds\right)$$

$$\rightarrow \sqrt{n}\frac{RV^h - IV^h}{\sqrt{2IQ^h}} \xrightarrow{d} N(0,1)$$

The integral $\int_{t-1}^{t} \sigma_s^4 ds$ is the *integrated quarticity* (IQ), which is not observable. The authors showed that the *realized quarticity* (RQ) estimator is consistent for the IQ:

$$RQ_t = \frac{1}{3} n\delta^{-1} \sum_t^T r_t^4 \xrightarrow{d} IQ^h$$

and the sampled asymptotic distribution becomes:

$$\frac{\sum_t^T r_{\delta_t}^2 - \int_0^T \sigma_s^2 ds}{\sqrt{\frac{2}{3}\sum_t^T r_{\delta_t}^4}} = \sqrt{n}\frac{RV^h - IV^h}{\sqrt{2RQ^h}} \xrightarrow{d} N(0,1)$$

Better results in terms of efficiency, even in small samples, may be obtained by using the approximated estimator:

$$\frac{\ln RV^h - \ln IV^h}{\sqrt{\frac{2RQ^h}{(RV^h)^2}}}$$

4.2 Models for volatility

4.2.1 Equally spaced time interval

The availability of HFD improved substantially the analysis of econometrics models. If the assumption of a regular time space holds, then it is possible to divide the price path in interval of equal length $\Delta t=T/n$. Assuming that the log-price follows the process (4.1):

$$dp_t = \mu\, dt + \sigma\, dW_t$$

then the log difference is normally distributed:

$$\Delta p_t \sim N(\mu \Delta t, \sigma^2 \Delta t)$$

and, by *maximum likelihood estimation*, it yields:

$$\hat{\mu} = \frac{\sum \Delta p_t}{n\Delta t} = \frac{\sum \Delta p_t}{T} = \frac{p_t - p_0}{T}$$

$$\hat{\sigma}^2 = \frac{\sum(\Delta p_t - \overline{\Delta p})^2}{n\Delta t}$$

or, alternatively, the unbiased estimator for variance is:

$$\hat{\sigma}^2 = \frac{\sum(\Delta p_t - \overline{\Delta p})^2}{(n-1)\Delta t}$$

If the log-price is standardized:

$$z_t = \frac{\Delta p_t - \overline{\Delta p}}{\sigma\sqrt{\Delta t}} \sim N(0,1)$$

$$\sum z_t^2 \sim \chi^2_{n-1}$$

the unbiased estimator for the variance can be written as:

$$\hat{\sigma}^2 = \frac{\sum z_t^2}{(n-1)\Delta t} \cdot \frac{\sigma^2 \Delta t}{\sigma^2 \Delta t} = \frac{\sigma^2}{n-1} \cdot \left(\frac{\Delta p_t - \overline{\Delta p}}{\sigma\sqrt{\Delta t}}\right)^2 = \frac{\sigma^2}{n-1}\chi^2_{n-1}$$

and the moments of this estimator are

$$E[\hat{\sigma}^2] = \frac{\sigma^2}{n-1} E[\chi^2_{n-1}] = \frac{\sigma^2}{n-1}(n-1) = \sigma^2$$

$$Var[\hat{\sigma}^2] = \frac{\sigma^4}{(n-1)^2} Var[\chi^2_{n-1}] = \frac{\sigma^4}{(n-1)^2} 2(n-1) = \frac{2\sigma^4}{n-1}$$

which shows that the estimator is consistent and unbiased as *n* tend to infinity. The asymptotic distribution for the variance estimator is:

$$\sqrt{n-1}(\hat{\sigma}^2 - \sigma^2) \overset{d}{\rightarrow} N(0, 2\sigma^4)$$

It is possible, without loss of generality, to express the variance as sum of squared observations, as a centred distribution. Indeed:

$$\hat{\sigma}^2 = \frac{\sum(\Delta p_t - \overline{\Delta p})^2}{n\Delta t} = \frac{\sum(\Delta p_t^2 + \overline{\Delta p}^2 - 2\Delta p_t \overline{\Delta p})}{n\Delta t} = \frac{\sum(\Delta p_t^2) - n\overline{\Delta p}^2}{n\Delta t} = \sigma^2_{centred} - \frac{\overline{\Delta p}^2}{\Delta t}$$

where the last term tends to zero as the interval become smaller. At high frequency, the centred asymptotic distribution is the same as a non-centred one. It can be argued also that tick returns are so small that the mean is almost zero, or negligible.

4.2.2 Irregular time space

Daley and Vere-Jones (1988) defined the *point processes* as those processes where the time of trades is a sequence of non-decreasing random variable, and, at any time point, the number of trades behave as a random variable. This process seems to fit reasonably well the continuous time price process. Engle and Russel (1998) gave an example on how to model the stochastic process of time. They introduced the *autoregressive conditional duration* model (ACD), with the *duration*, δ_t, being the interval of time between two consecutives orders arrival. The duration is a stochastic process, supposed to follow a GARCH-like model. This found application in Engle (2000), who proposed the Ultra High-Frequency GARCH model, where the variance over this small time interval is:

$$v^2_{\delta t} = Var_t(r_t | \delta_t)$$

and then the variance of the unit period, that is what really matters, is:

$$\sigma_t^2 \overset{\text{def}}{=} Var_t\left(\frac{r_t}{\sqrt{\delta_t}}\middle|\delta_t\right) = \frac{v_{\delta t}^2}{\delta_t} \quad \Rightarrow \quad v_{\delta t}^2 = \delta_t \sigma_t^2$$

which is supposed to follow a GARCH (1,1) process. The interval of time between the observations may be modelled as follows:

$$\delta_t = \psi_t \varepsilon_t$$

with *{ψ}* denoting a stochastic process following a generic GARCH distribution. If the ACD process for the arrival times is supposed to be exogenous from price-volatility process, it is possible to estimate duration at first, and then estimate through MLE the GARCH volatility model, conditional on the previous results. This procedure may still result inefficient. Having defined another stochastic process implies to deal with more complex likelihood formulation.

4.2.3 Microstructure noise effect

In the presence of market microstructure noise, according to Hansen and Lunde (2006) the RV estimator for IV is biased, depending on the degree of noise. The observed price is:

$$p_t^* = p_t + \varepsilon_t, \quad \varepsilon_t \sim N(0, \sigma_\varepsilon^2)$$

and, thus, is expected that:

$$\begin{aligned} E[RV_t^*] &= E[\textstyle\sum r_s^{*2}] \\ &= E[\textstyle\sum (r_s + \varepsilon_s - \varepsilon_{s-1})^2] \\ &= IV_t + 2n\sigma_\varepsilon^2 \end{aligned}$$

This result implies that, as *n* approaches to infinity (or the interval tend to zero) the estimator for IV diverges linearly in *n*. This is the reason why in literature is preferred the use of *sparse sampling*, that is the use of 1-30 minutes data frequency, unless there exists an efficient estimator that takes into account the presence of microstructure noise (as, for example, in Andersen, Bollerslev and Diebold, 2008). The sparse sampling procedure reduce the information set available and thus a less precise estimate of instantaneous volatility may result.

4.3 Parameters estimation

As discussed, stochastic volatility models introduce a distribution process also for variance. There are at least two multivariate processes characterized by their particular distribution, that are the price process and the variance process. In high frequency framework also time can be stochastic. This means that the classical method of estimation by maximum likelihood is infeasible, since it would be necessary to deal with several "layers" of distributions and conditional distributions, which do not allow for a closed-form of the likelihood function.

In literature there are several alternative to the SV parameter estimation procedures, for instance the GMM approach. Andersen and Sorensen (1996) gave the first approach toward this issue, by expressing the parameters in terms of population conditional moments, and then substituting the expectation operator with the sample moments. Ruiz (1994), proposed the estimation through Quasi-MLE, which uses a simplified formulation of the likelihood function in order to produce an estimation of the latent variables. Empirical evidence shows that the distribution of innovations if far from normality, in the better case it has fatter tails, if not even asymmetric. The QMLE impose a Gaussian distribution to innovation, which makes computations lighter. This simplification may be applied only if fourth moment of the innovations' distribution is finite, but works very well even if the true distribution is not Normal. The QMLE, under the latter assumption, is a consistent estimator, although not efficient.

With the development of computing power, there have been developed estimation procedure that rely on simulations. For instance, Monte Carlo (MC) simulation methods hae been widely adopted. Since it is not feasible to have a closed-form for the moments of complex distributions, it is possible to simulate the patterns of these distributions and then compute the sample moments on the generated observations. Jacquier, Polson and Rossi (1994), were early adopters of simulations methods using Bayesian analysis. Their studies were used by Chib, Nardari and Shephard (2002) to develop the Markov Chain

Monte Carlo (MCMC) simulation methods for volatility estimation. MCMC approach consists into the application of particular algorithms (Gibbs sampling, Metropolis-Hasting) that generate independent samples of a stochastic process, *i.e.* Markov Chains, and then use simulation method to estimate the parameters of interest. This allows independent draws from complicated posterior distributions of the interested variables, for example volatility and parameters in a stochastic volatility framework.

4.3.1 Maximum likelihood estimator with microstructure bias

The markovian property of log-prices allows certain simplification in likelihood function. Using Bayesian probability and Markov distributions properties, the probability of the observation set *{p_t}* occurring is:

$$\begin{aligned} P(p_n, \dots, p_0; \theta) &= P(p_n | p_{n-1}, \dots, p_0; \theta) \cdot P(p_{n-1}, \dots, p_0; \theta) \\ &= P(p_n | p_{n-1}; \theta) \cdot P(p_{n-1}, \dots, p_0; \theta) \\ &= P(p_n | p_{n-1}; \theta) \cdot \dots \cdot P(p_0; \theta) \end{aligned}$$

and the likelihood function is:

$$\mathcal{L}(\theta) = \ln P(p_0; \theta) + \sum_{i=1}^{n} \ln P(p_i | p_{i-1}; \theta)$$

which does not have a closed-form solution.

Aït-Sahalia, Mykland and Zhang (2005a) gave a practical approach of estimation through MLE in presence of market microstructure noise. Let the noisy log-price be the correct price plus some error term:

$$p_t^* = p_t + \varepsilon_t, \quad \varepsilon_t \sim N(0, \sigma_\varepsilon^2)\ iid$$

Then returns are defined as the difference of log-prices:

$$r_t = p_t - p_{t-\Delta t}, \quad r_t \sim N(0, \sigma_r^2 \Delta t)$$

with r_t independent from the noise ε_t. Substituting the previous equality the observed return is:

$$\begin{aligned} r_t^* &= p_t^* - p_{t-\Delta t}^* \\ &= p_t + \varepsilon_t - p_{t-\Delta t} - \varepsilon_{t-\Delta t} \\ &= r_t + \varepsilon_t - \varepsilon_{t-\Delta t} \end{aligned}$$

whose moments of interest are:

$$Var(r_t^*) = \sigma_r^2 \Delta t + 2\sigma_\varepsilon^2$$

$$Cov(r_t^*, r_{t-\Delta t}^*) = -\sigma_\varepsilon^2$$

Observed returns can be rewritten as a MA (1) process:

$$r_t^* = u_t + \theta u_{t-\Delta t}, \quad u_t \sim N(0, \sigma_u^2)$$

whose moments are:

$$Var(r_t^*) = \sigma_u^2 (1 + \theta^2)$$

$$Cov(r_t^*, r_{t-\Delta t}^*) = \theta \sigma_u^2$$

which can be easily estimated through MLE procedure. Now, equating the previous equations, it is possible to use the latter estimated parameters to find an estimation for the variance of returns and innovations:

$$\widehat{\sigma_\varepsilon^2} = -\hat{\theta}\widehat{\sigma_u^2}$$

$$\widehat{\sigma_r^2} = \frac{\widehat{\sigma_u^2}(1 + \hat{\theta}^2) - 2\widehat{\sigma_\varepsilon^2}}{\Delta t} = \frac{\widehat{\sigma_u^2}(1 - \hat{\theta})^2}{\Delta t}$$

4.3.2 Realized kernel estimator

Kernel estimators are non-parametric class of functions, which allow to fit a distribution starting from observed data. The simplest case consists into fitting the distribution as a sum of sinusoidal curves with equal height and width. Barndorff-Nielsen, Hansen, Lunde and Shephard (2008) proposed a kernel estimator ("flat-top") as an alternative to the MLE. The latter is not consistent and not unbiased in high-frequency framework. This is true in particular when is imposed a long-memory autocorrelation structure, both in the noisy errors and returns. The estimator proposed by the authors is:

$$RK = \gamma_0 + \sum_{h=1}^{H} k\left(\frac{h-1}{H}\right)(\gamma_h + \gamma_{-h})$$

$$with \quad h = -H, \dots, -1, 0, 1, \dots, H$$

with

$$\begin{aligned} \gamma_h &= \sum (p_t - p_{t-\Delta t})(p_{t-h} - p_{t-h-1}) \\ &= \sum r_t r_{t-h} \end{aligned}$$

denoting the realized auto-covariance, and *k(x)* representing a weighting function defined on a domain space *x∈ [0,1]* with *k(0)=1, k(1)=0.* The authors chose the Tuckey-Hanning kernel function:

$$k(x) = \sin^2(\pi/2(1-x)^2\ )$$

The ideal *bandwidth H** is a function both of realized variance and realized quarticity. The authors showed also that this kernel estimator is robust to market microstructure noise and irregularly spaced observations.

4.3.3 Bipower variation

Barndorff-Nielsen and Shephard (2003) introduced the Realized Bipower Variation (BPV) estimator as a robust estimator in case of jump processes in volatility structure. If the underlying model is characterized by jumps, that are discrepancy in price due to news or announcements, the QV process do not converge to the IV. Let the process be:

$$dp_t = \mu_t dt + \sigma_t dW_t + \epsilon_t dq_t$$

whose solution is

$$p_t - p_0 = r_{0\to t} = \int_0^t \mu_s ds + \int_0^t \sigma_s dW_s + k_t N_{0\to t}$$

with *{k} iid* random process, *{q}* a Poisson process and *N* its integral, denoting the number of jumps occurred during the interval of time. As the time interval tend to zero, it is possible to demonstrate that the QV process equals:

$$QV_t = IV_t + JV_t$$

with

$$JV_t = \sum_{j<t} k_j^2$$

Authors proposed to estimate IV through the BPV, defined as

$$BPV_t(p,q) = \mathrm{p} - \lim_{\sup\{\delta\}\to 0} \delta^{1-\frac{p+q}{2}} \sum_{i=2}^{t} \left|r_{\tau_i}\right|^r \left|r_{\tau_{i-1}}\right|^s$$

In general, it is expected that

$$BPV_t(p,q) = \mu_p \mu_q \int_0^t \sigma_u^{p+q} du$$

$$\mu_x = E[|u|^x] = 2^{x/2} \frac{\Gamma\left(\frac{\mathrm{x}+1}{2}\right)}{\Gamma(1/2)}, \quad u \sim N(0,1)$$

which, in the particular case of *p=q=1*, it yields to the following result:

$$\frac{\pi}{2} \sum_{i=2}^{t} \left|r_{\tau_i}\right| \left|r_{\tau_{i-1}}\right| = \frac{\pi}{2} BPV_t \xrightarrow{p} IV_t$$

which is a consistent estimator for IV and robust to the presence of jumps. The reason is that jumps occur just a limited number of times during the observed period. The number of contiguous jumps tends to zero in probability as the time interval goes to zero, and consequently these terms have negligible impact on the limit probability. BPV is a robust estimator, but not efficient, since RV has still less variance. Similarly, it is possible to estimate the IQ with the *quadpower variation*:

$$QPV_t = \delta^{-1} \sum_{i=4}^{t} \prod_{j=0}^{3} \left|r_{i-j}\right| \xrightarrow{p} \frac{4}{\pi^2} IQ_t$$

This is useful to make inference about the continuous property of prices. The standardized bipower variation is:

$$Z_{BPV} = \frac{\delta^{-\frac{1}{2}}(\mu_1^{-2}BPV_t - RV_t)}{\sqrt{\mu_1^{-4}QVP_t}} \xrightarrow{d} N(0,\upsilon)$$

which, for significantly negative values, rejects the null hypothesis of a continuous sample path, in favour of a discrete-jump process.

4.3.4 Two Stage Realized Variance

In presence of market microstructure bias, the QV process of observed "dirty" prices is consistent and asymptotically normal estimator also for the quantity $2nE[\varepsilon^2]$, rather than the only volatility process. Aït-Sahalia, Mykland and Zhang (2005b) proposed the Two Stage Realized Variance (TSRV) estimator to overcome this problem. They started from the fact that the biased estimator yields to:

$$\langle r^* \rangle_t \rightarrow \langle r \rangle_t + 2nE[\varepsilon^2] + O(4)$$

Using a sparse sampling procedure allows to reduce the magnitude of the second element of this equation (the microstructure noise). If, for example, observations are sampled at 1-second interval, then a better estimator can be obtained by sampling at 5-seconds interval, for each of the possible 5-seconds windows, and then averaging those estimators. In formulas:

$$\langle r^* \rangle_t^{avg} = \frac{1}{K}\sum_{k=1}^{K} \langle r^* \rangle_t^k$$

with *K* denoting the sampling interval, *e.g.* 5, and the term into summation is the RV estimator computed on the grid of length *K* starting at *k*. Since the bias term can be consistently estimated through:

$$\widehat{E[\varepsilon^2]} = \frac{1}{2n}\langle r^* \rangle_t$$

this implies that, denoting with $\bar{n}$ the average length of the grids,:

$$\langle r^{*}\rangle_{t}^{avg} = \langle r\rangle_{t} + 2\bar{n}E[\varepsilon^{2}] + O(4)$$

$$\langle r\rangle_{t}^{TSRV} = \langle r^{*}\rangle_{t}^{avg} - \frac{\bar{n}}{n}\langle r^{*}\rangle_{t}$$

which is the unbiased estimator for the QV process.

4.3.5 Realized range-based variance

Methods of estimation that made use of ranges, rather than returns, were pioneered by Parkinson (1980). The very contribution came from Alizadeh, Brandt and Diebold (2002), who showed that these estimators are more efficient than RV. The advantages of the range estimators are that they have less variability (since, by construction, is a measure of data aggregation), are more robust to microstructure noise, and their logarithm is normally distributed. They are built using just the variation of the interested variable, *i.e.* log-prices, over a certain period of analysis, *e.g.* a trading day, ignoring all the intraday observations. Formally the range estimator for the *h-th* day proposed by the authors is:

$$RgV^{h} = \sup_{\mathrm{h-1\leq t\leq h}} p_{t}^{h} - \inf_{\mathrm{h-1\leq t\leq h}} p_{t}^{h}$$

It is the difference between the high and low quote for the selected interval. Those data are also available on the main financial newspapers. A property of this estimator is that it is more efficient than RV and makes computations lighter, but there is a substantial loss of information. There is always the trade-off between sampling at higher frequencies and have more information, or have less information with less noise. Sometimes in literature is widely used the log of this estimator. The estimator proposed by Parkinson (1980), in absence of the drift component in price diffusion is:

$$\widehat{\sigma_{h}^{2}} = \frac{1}{4\ln 2}\sum_{\Delta t\in(h-1;h]}^{h}(RgV^{\Delta t})^{2}$$

and then it is possible to use this estimator in an AR(p) model to forecast the future realization of variance. Rogers and Satchell (1991) improved this formulation introducing both the drift diffusion term and information about opening and closing prices. They found that this estimator is more efficient.

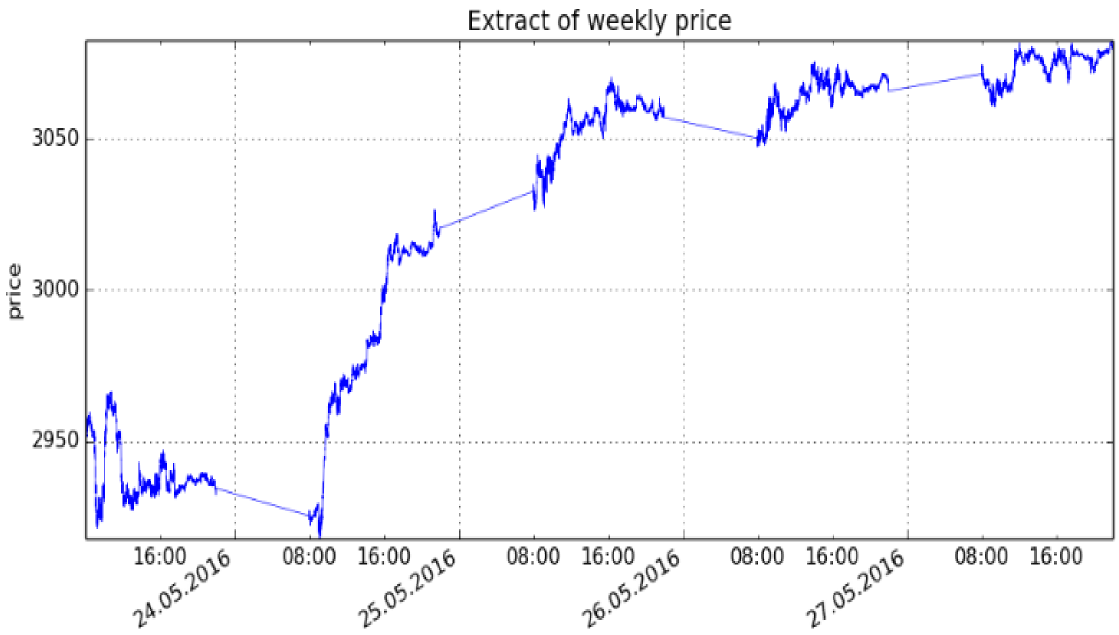
A possible drawback is that these estimators are sensible to outliers. To overcome this issue, a feasible solution is to use the range quantiles, that is to consider determined quantiles of the observation set instead of the lowest and highest observation during the time period.

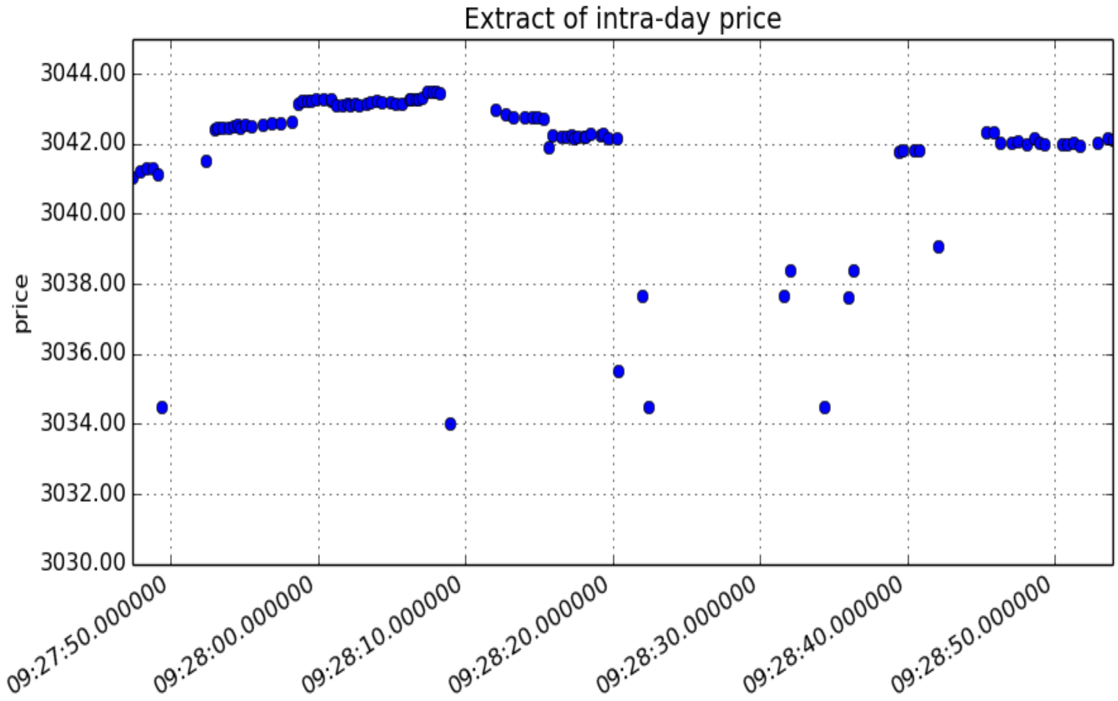
5. APPLICATIONS

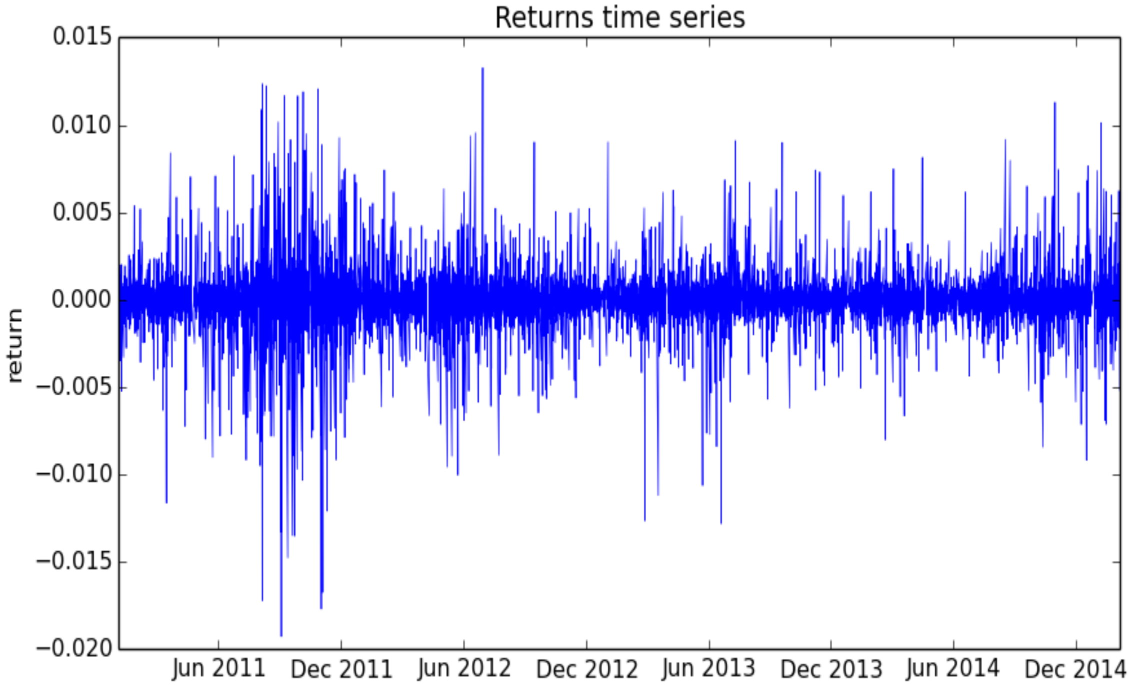
This chapter will be focused on the application of stochastic volatility models to a high-frequency database containing the observations for the EUROSTOXX index levels. The analysis covers the period from 2011 to 2015, at irregular intra-minutes frequencies, for a total of 4.598.132 rows. All computations have been done with Julia language. Figures (2) show some extracts of the database.

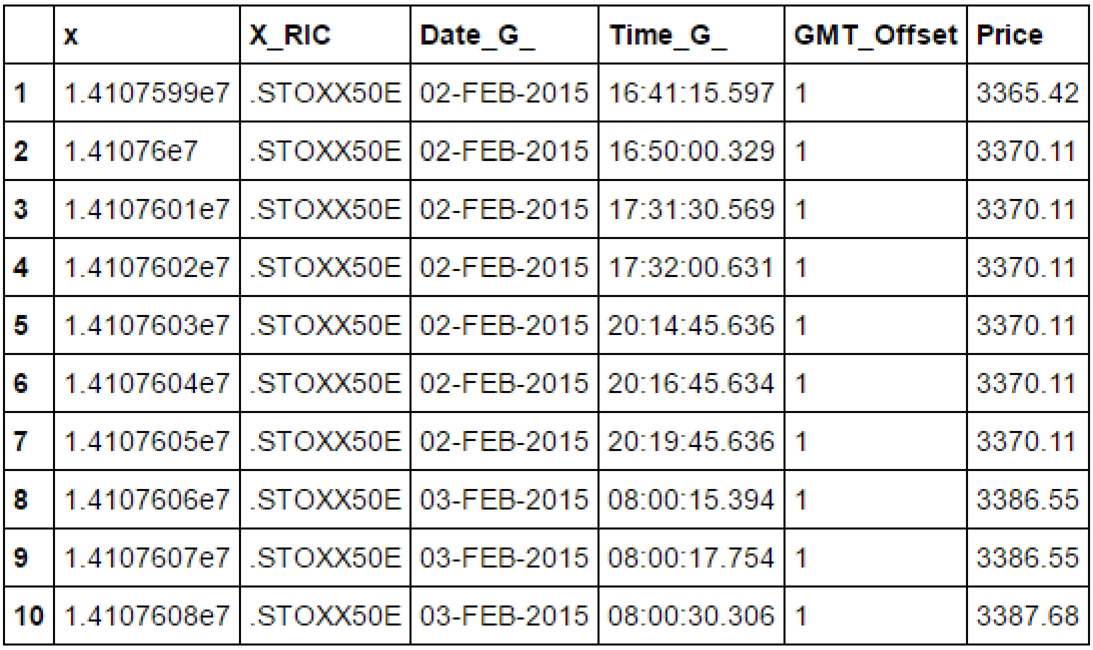
5.1 Preliminary adjustments

Some adjustments were necessary to handle this huge amount of data. Some extracts of the database are shown if figures (2). Figure (2a) is an extract of weekly prices, while figure (2b) is a one-minute extract where possible outliers can be noticed. Figure (2c) plots the time series of observed high-frequency returns. Figure (2d) shows a table of the data analysed.




Extract of intra-day price
price
3044.00
3042.00
3040.00
3038.00
3036.00
3034.00
3032.00
3030.00
09:27:50.000000
09:28:00.000000
09:28:10.000000
09:28:20.000000
09:28:30.000000
09:28:40.000000
09:28:50.000000


Returns time series
return
0.015
0.010
0.005
0.000
−0.005
−0.010
−0.015
−0.020
Jun 2011
Dec 2011
Jun 2012
Dec 2012
Jun 2013
Dec 2013
Jun 2014
Dec 2014



	x	X_RIC	Date_G_	Time_G_	GMT_Offset	Price
1	1.4107599e7	.STOXX50E	02-FEB-2015	16:41:15.597	1	3365.42
2	1.41076e7	.STOXX50E	02-FEB-2015	16:50:00.329	1	3370.11
3	1.4107601e7	.STOXX50E	02-FEB-2015	17:31:30.569	1	3370.11
4	1.4107602e7	.STOXX50E	02-FEB-2015	17:32:00.631	1	3370.11
5	1.4107603e7	.STOXX50E	02-FEB-2015	20:14:45.636	1	3370.11
6	1.4107604e7	.STOXX50E	02-FEB-2015	20:16:45.634	1	3370.11
7	1.4107605e7	.STOXX50E	02-FEB-2015	20:19:45.636	1	3370.11
8	1.4107606e7	.STOXX50E	03-FEB-2015	08:00:15.394	1	3386.55
9	1.4107607e7	.STOXX50E	03-FEB-2015	08:00:17.754	1	3386.55
10	1.4107608e7	.STOXX50E	03-FEB-2015	08:00:30.306	1	3387.68

Figure 2. Extracts of the database. Respectively, weekly sample pattern, one-minute sample pattern, time series of high-frequency returns, sample table of the dataset.

5.1.1 Dates handling

It is necessary to collect the date and time array into the language-specific type. In Julia it is sufficient to merge the *Date* string and the *Time* string of the dataset, and the convert them into a *DateTime* type, specifying the correct formatting. Since intraday time is expressed in GMT time, which means that observed trading hours vary depending on winter or summer time period, dates are converted into local time. It is sufficient to add the GMT offset (hours offset) to the intraday time. This allows to save time for cleaning operations.

Dates operations require much more computing power, respect real number operations. Julia language has few and basic functions which may be applied to *Date* and *DateTime* type, compared to other programming languages. It does not account for *time serialization*, *i.e.* it does not provide the identifier number to dates. The code written for this analysis accounts also for data conversion. Since the final objective is to produce a minute-by-minute time series of prices, dates are converted in number such that each point of this date-time array measures the minutes elapsed from a given starting time. The first observation is selected as the *time-zero* point. It is then possible to save the information of this serialized-time array, along with the identity of the first starting point. Passing from *String* type to *real* type (precisely, to *Float64* or even *UInt32*) allows to reduce the loading time necessary to open the file for future analysis, and it requires less memory usage, either in terms of storage requirements and local memory.

5.1.2 Data cleaning

Cleaning operations need to be applied either on Time array and on Prices. As figure (2) shows, there are some observations which are reported but they do not belong to normal trading hours. Only observations between 9:00 am and 17:30 pm are considered in this analysis, the other will be discarded. It is rather simply to make comparison operations in Julia. The only problem is that, since it does not provide an intraday-time type (time without the day), it is necessary to create an array of *DateTime*, starting from intraday observations, pretending they happen on a same fixed day. Once outliers are detected, the corresponding prices also are deleted from the dataset.

Regarding price cleaning, since the dataset is composed of index observations, which is less sensible to all those errors described in the previous chapters, no further operations need to be done. Index data are given by the weighted sum of its component, which means that whenever an error occurs on one of the underlying stocks, it is simply averaged by the other components' price, thus reducing drastically the entity of the bias and the probability of encountering an error in the index price. The same reasoning applies to market microstructure bias.

5.1.3 Data modelling

In order to compute stochastic volatility measures, it is preferable to create an equally spaced time grid. RV can be computed using heterogeneously-spaced observations, but, as also confirmed by literature, *sparse sampling* is more efficient. The algorithm used in this analysis is built on the following steps. A *DateTime* array of the equally-spaced times is created, and then a price is assigned to each one of this point. The price assigned in correspondence of a given point is the price occurring at the nearest previous observation, thus the last observed price at that point. The position of the wanted observation is determined by comparing each point of the homogeneous-spaced time grid with the whole heterogeneous dates array. The corresponding price is simply the price on the whole price array, at the positions given by the previous step. Appling the "normal" algorithm which finds for the last *DateTime* less than a given point on the time grid, for all elements of this time grid array, would require a huge computation effort and amount of time. The *@time* macro built in Julia allows to understand the time elapsed for the computation of a routine, and the allocated memory. The standard procedure requires, in the best of the cases, about 0,86 seconds each 100 points of the time grid, with memory allocation of more than 63 MB. Since there are 536.550 points on the time grid, the time necessary to run the code would be about 4.614 seconds, that is 1 hour and 20 minutes. Even using the serialized time array, the benefits in reducing the time are very small, elapsed time passes to 0,683 per 100 points (1 hour for whole array). The algorithm developed in this thesis, instead, is able to perform the same computations, on first 100 points) in about 0,0015 seconds and just 17 KB allocated, for a total of 8 seconds on the total array. It is possible to reduce further the elapsed time by using sparse matrices, to a total of 0,19 seconds for the whole array, and a total of 84 MB allocated. Once positions are found, there are still "missing slots" to fill, which are the one that are left blank when non-adjacent minutes observations are encountered. These spaces are filled with the most recent price at that time.

Creating a uniform time grid allows to make computations easier, both because number of observations is reduced, and because the position of a given observation in time is well defined. The latter result implies computational efforts savings, since it is possible to make operations without the need of intraday times. The resulting price matrix has the following form:

$$pricemat = \begin{bmatrix} p_{03/01/11|9:00} & p_{04/01/11|9:00} & \cdots & p_{05/02/15|9:00} \\ p_{03/01/11|9:01} & p_{04/01/11|9:01} & \cdots & p_{05/02/15|9:01} \\ \vdots & \vdots & \ddots & \vdots \\ p_{03/01/11|17:30} & p_{04/01/11|17:30} & \cdots & p_{05/02/15|17:30} \end{bmatrix} \Bigg\downarrow \; \mathit{Intraday\ time}$$

$$\xrightarrow[\mathit{Dates}]{\hspace{6cm}}$$

It is possible to work on this matrix to compute variables of interest. RV at 1-minute frequency (RV1) is computed by summing the squared returns, column-by-column. For purpose of analysis, also 5-minute RV (RV5) is computed, which is derived from the price matrix, where only rows which are multiple of 5 are considered. TSRV is obtained by applying the previous computation to the five 5-minutes rolling windows series of RV, then averaged and cleaned from the bias term. BPV is obtained simply by summing the adjacent product series of returns.

5.2 Data analysis

5.2.1 Returns series

As largely discussed in literature, return distribution is not normal, but shows fatter tails. The distribution has more kurtosis the higher the frequencies are.

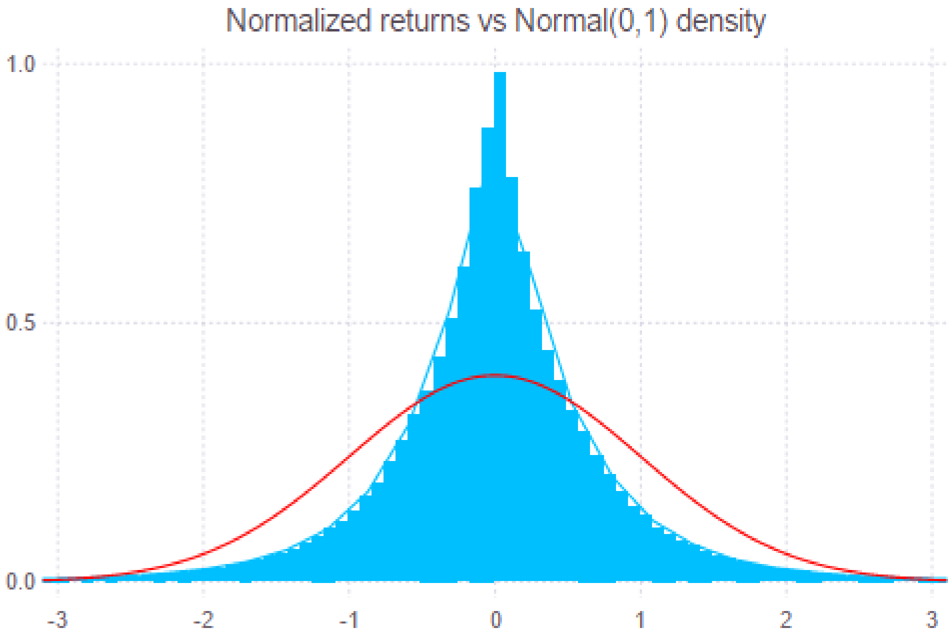


Figure 3: Normalized returns density.

Figure (3) shows the density of returns normalized using the sample standard deviation, compared with a Normal (0,1) distribution. It is clear that returns are distributed almost totally around the mean, with very few, but significant extremes (there are 396 normalized observations which are greater than 10 in absolute value). With daily frequency, normalized return has almost normal distribution. One of the peculiarity of RV is that:

$$\frac{r_{t+1}}{\sqrt{RV_{t+1}}} \overset{d}{\rightarrow} N(0,1)$$

the resulting density assumes the form seen in figure (4), that is very close to a Normal (0,1), at least it is closer than the other one.

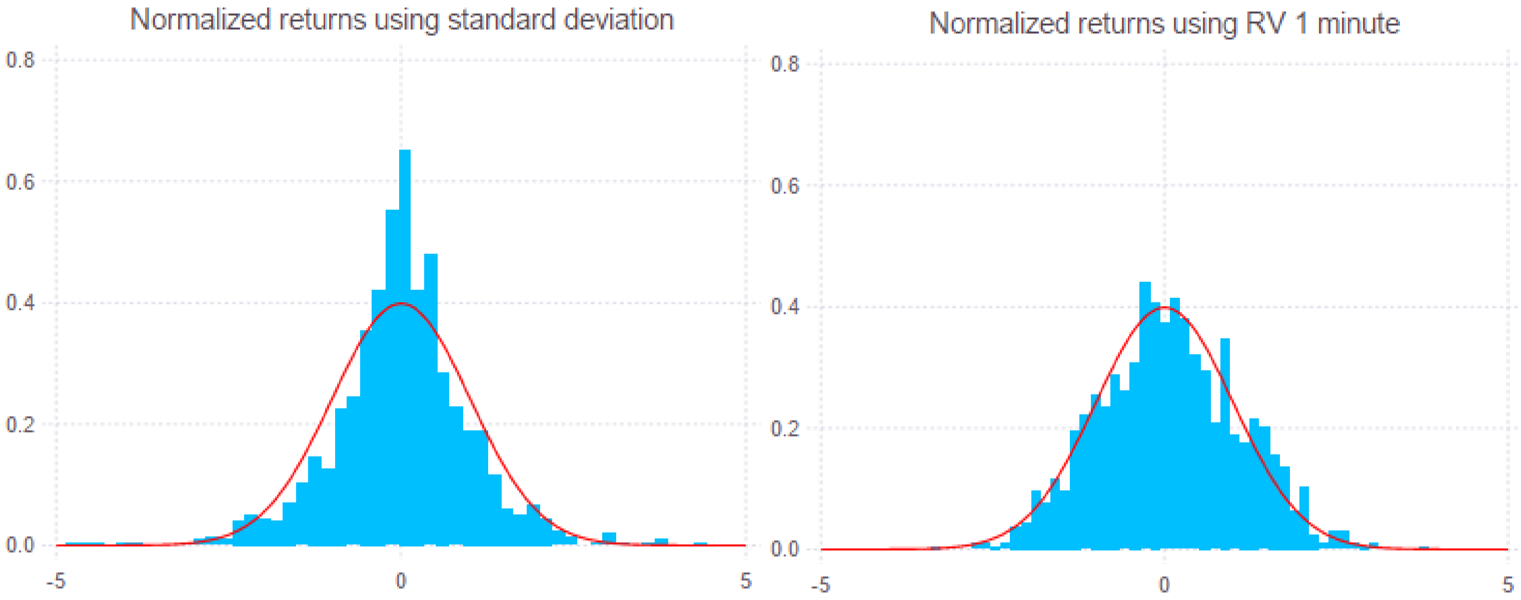


Figure 4: Comparison between normalized return with, respectively, sample standard deviation and RV. In red, Normal (0,1) is displayed.

This is a very important result, because it allows to make more accurate inference. It can be useful, for example, for risk management purposes, in evaluating the quantiles of future realizations of returns.

5.2.2 Volatility measures

In the following section statistical properties of some selected volatility measures will be analysed. The variables considered are RV1, RV5, BPV and TSRV. These last two in particular are chosen since they respond to specific issues of realized variance, respectively jumps and microstructure noise.

Before comparing these four variable, a premise is necessary. RV computed using all available prices has not been included, since it is "too noisy". As stated in previous chapters, *sparse sampling* procedure yields to better results, as shown in following tables and figures. The choice of one and five minutes is inducted mainly by literature, and because these frequencies are the most representative of the category of high-frequency sampled observations. Looking at figure (5), it is clear that RV1 has the highest explanatory power, in terms of adjusted R-squared of a regression of the variable on its lags. The number of lags are determined by BIC procedure. RV5 seems to be a good representative of lower sampling frequencies, since adjusted R-squared is similar to that of the following sampling minutes.

Sampling frequency (minutes)	*Adj. R-squared of* $\boldsymbol{RV_t \sim [1\ RV_{t-lag}]}$	*Adj. R-squared of* $\boldsymbol{\sqrt{RV_t} \sim [1\ \sqrt{RV_{t-lag}}\ ]}$
0	0,091	0,192
1	0,609	0,642
2	0,585	0,624
3	0,540	0,587
4	0,487	0,554
5	0,476	0,546
6	0,468	0,535
7	0,441	0,521
8	0,441	0,526
9	0,467	0,534
10	0,479	0,543

Figure 5: Sparse sampling results for RV. RV_{t-lag} is a matrix whose columns are the lags of RV. The number of lags is given by BIC procedure.

From figure (6) it is possible to notice that BPV has the least number and magnitude of "peaks", confirming the fact that this estimator aims to reduce jump bias in return series. Figure (7) shows the relation between each stochastic variable with its first lag. Combined with results obtained in figure (8), which shows the autocorrelation function, it is interesting to notice how strong and persistent is the relation between two adjacent observations, and it seems to stay stable after about 10 lags. Considered the results here analysed, RV1 and BPV seem to be the variables with the highest relation with their lags. These results translate into good chance of obtaining reasonable forecasts, with a simple AR(p) model of the daily stochastic variables. Figure (8) and (9) illustrates how this explanatory/predicting power is enhanced if square root process is used. Both the autocorrelation of square root process stays always higher than the normal process, and adjusted R-squared increases.

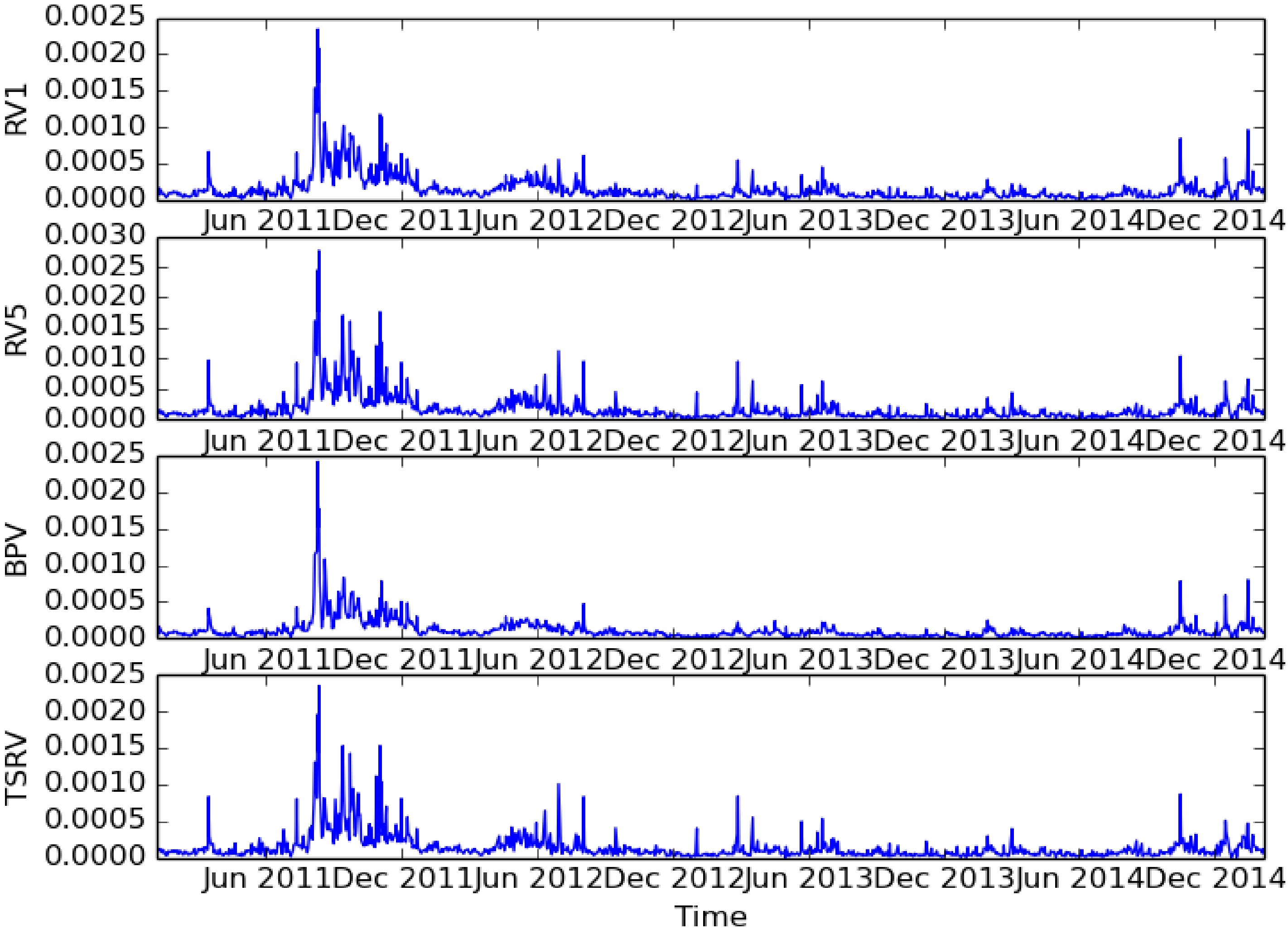


Figure 6: Plot of the stochastic variables time series.

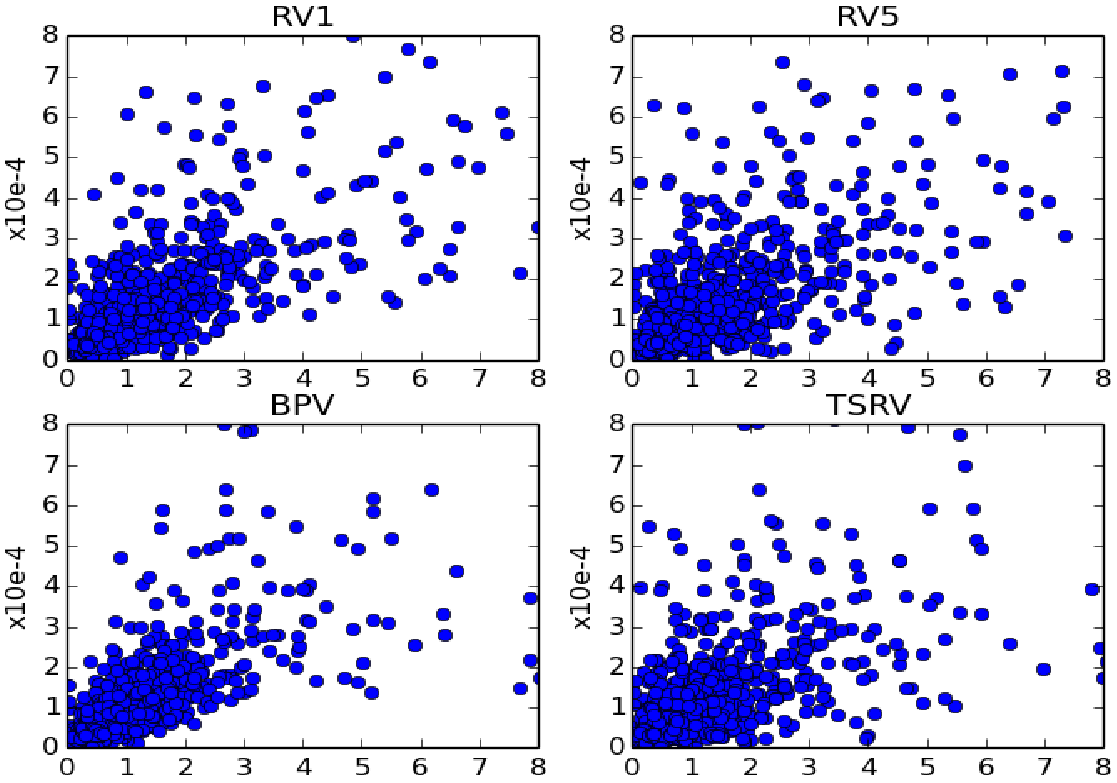


Figure 7: Scatter plot of stochastic variables on their first lag.

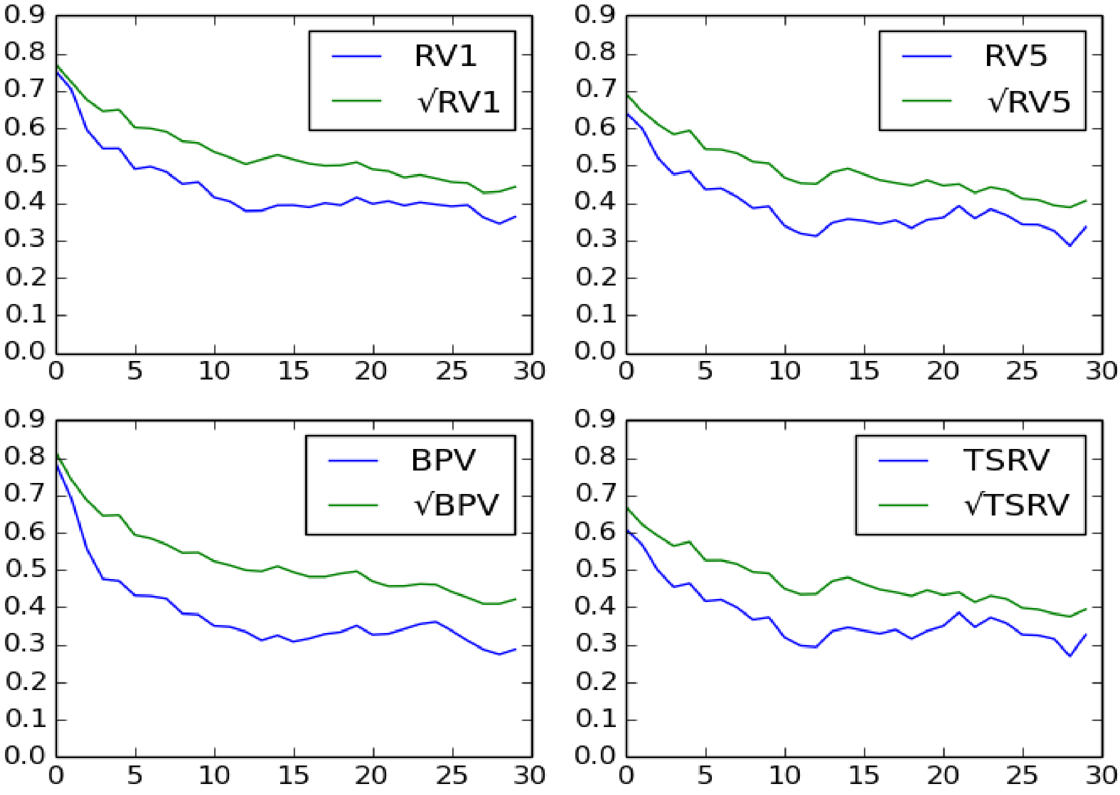


Figure 8: Autocorrelation function at first 30 lags both for normal process and its square root.

Variable	*Adj. R-squared of* $\boldsymbol{RV_t \sim [1\ RV_{t-lag}]}$	*Adj. R-squared of* $\boldsymbol{\sqrt{RV_t} \sim [1\ \sqrt{RV_{t-lag}}\ ]}$
RV1	0,6092	0,6417
RV5	0,4756	0,5463
BPV	0,6341	0,6848
TSRV	0,4395	0,5190

Figure 9: Adjusted R-squared of a regression of the stochastic variables into a constant and its lag. On the left column the analysis is conducted on simple variables, on the right column it is on the square root of those variables.

5.2.3 Assessing self-predictive power

Since the autocorrelations are significant, it is reasonably to assess the predicting ability of the variables on future realizations of themselves. The predicted variables are computed using rolling windows of one year. On each of these rolling windows, the coefficients of a regression of an AR (3) model with constant are computed, and then they are applied to the most recent observations in order to produce the one-step-ahead forecasts. To compare results across the different models, the rule of the squared error is applied. It consists into evaluating the sum of the squared deviations of the forecasts from the realized values. The analysis is conducted both on the normal series and on their square root. Figure (10) and (11) represent the time series of forecasted values. The first set of series is the normal one, while the second series is the square root process. It appears that forecasted values act as smoothing operators, since they are not able to forecast the unpredictable peaks of the realized variables. The most efficient, indeed, is the BPV, since peaks are more difficult to encounter. Figure (12) summarize those results. In both cases, it appears as BPV and RV1 are the most efficient "self-estimators".

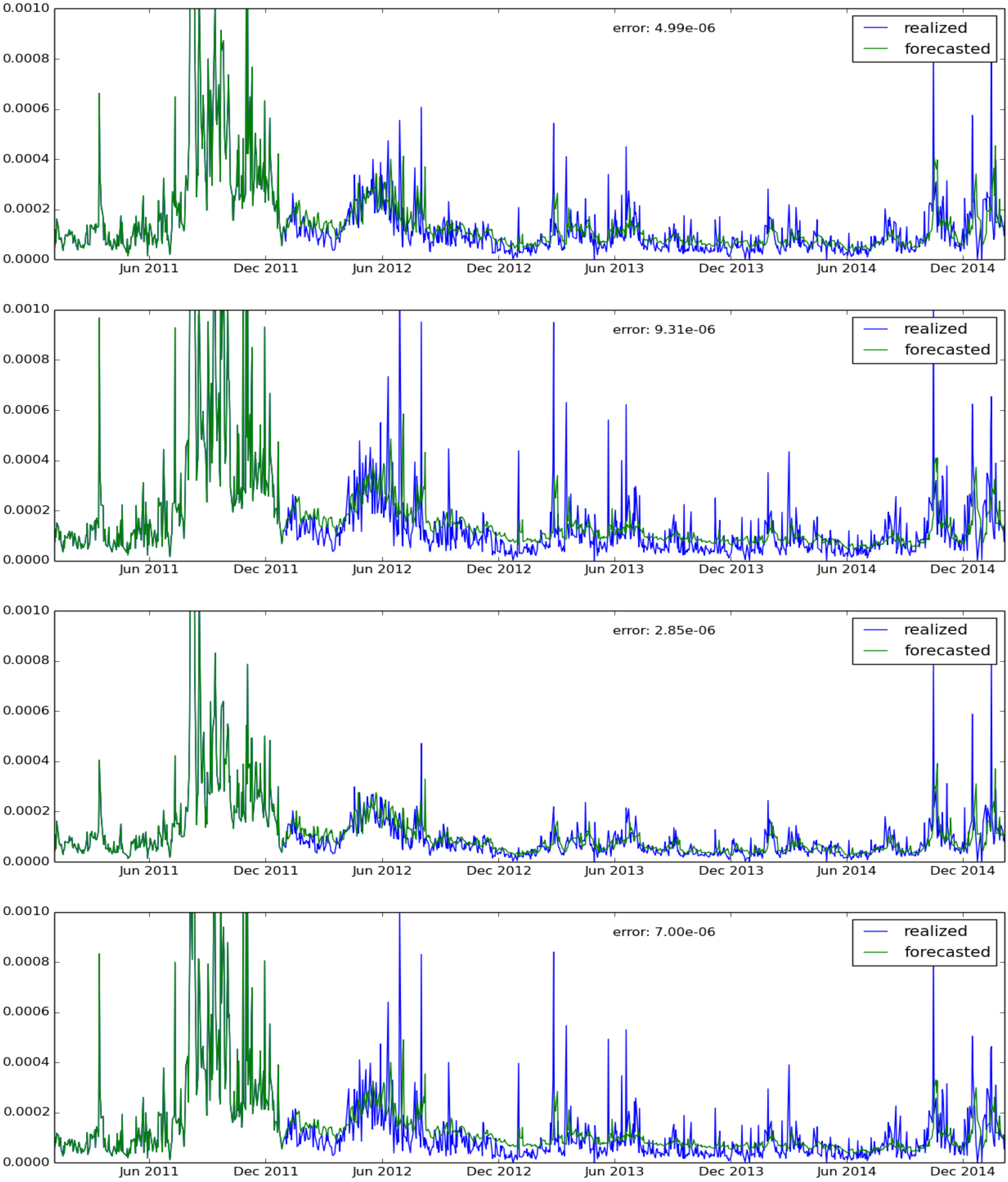


Figure 10: Rolling window forecasts of the stochastic variables vs actual realized observations.

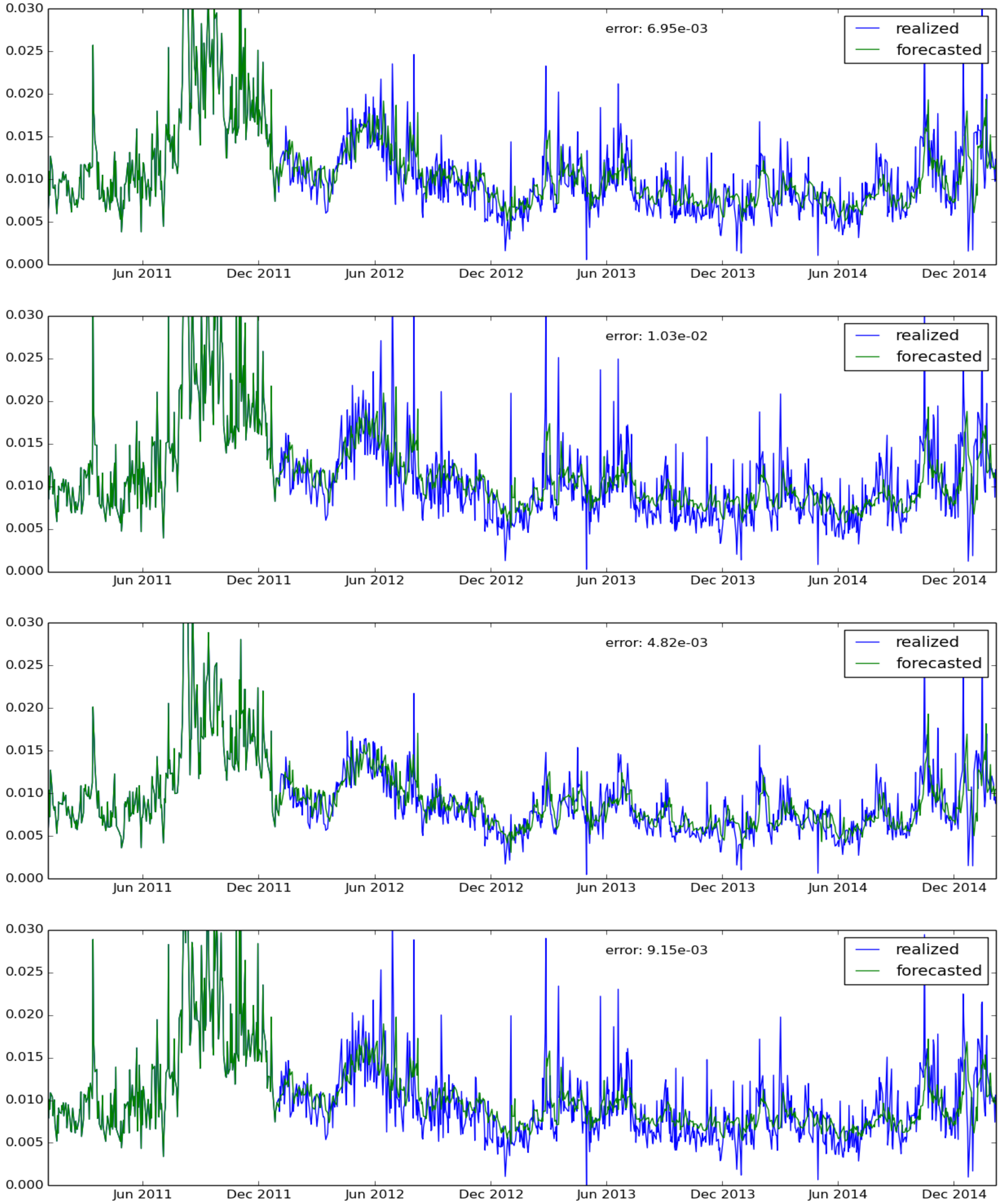


Figure 11: Rolling window forecasts of the square root stochastic variables vs actual realized observations.

Variable	*SSE*	*SSE(n)*	*Sign*
RV1	4,99E-06	0,137935	57,8%
RV5	9,31E-06	0,189187	56,2%
BPV	2,85E-06	0,110936	58,4%
TSRV	7,00E-06	0,164934	56,6%
√*RV1*	0,006945	4,38471	57,6%
√*RV5*	0,010317	5,39894	55,7%
√*BPV*	0,004821	3,66814	59,0%
√*TSRV*	0,009152	5,08072	56,6%

Figure 12: Explanatory power of stochastic variables. The analysis is conducted with a rolling window forecast. SSE stands for sum of squared errors, that are the difference between the forecasted and the observed variables. SSE(n) are the normalized errors, that are SSE divided by the mean of the absolute errors. Sign is the percentage of correct predicted movements, in terms of up or down movement.

The last words of this paragraph are worth to introduce a useful model which uses RV variables to construct a proxy for the stochastic variance. It is the Heterogeneous AutoRegressive (HAR) estimator, suggested by Corsi (2009), defined as follows:

$$HAR - RV_t = \beta_d RV_t^1 + \beta_w RV_t^5 + \beta_m RV_t^{20}$$

with:

$$RV_t^h = \frac{1}{h}\sum_{i=1}^{h} RV_{t-i}$$

The letters *d*, *w* and *m* are often used instead of the number indexes, since they represent, respectively, the daily, weekly and monthly moving average of RV. The HAR-RV may be a very powerful proxy for the variance process. Its utility emerges if it is considered that in a simple AR (3) process, an adjusted R-squared of 80% is obtained. This model seems to better capture the different horizons effects, by disentangling RV process into its weekly and monthly moving averages. It accounts in part also for mean reversion, if the monthly mean is close to this value.

The following section will show how to use stochastic volatility measures for financial purposes.

5.3 Application – trading the V2X

The first application shown in this thesis is a trading application. The object is to trade the VSTOXX index, whose ticker is *V2X*, that is the volatility index for the EUROSTOXX index. It works similarly to VIX index for the S&P500, thus it shows the volatility of the Euro-area implied from hedged at-the-money options. Figure (13) plots the level of V2X index versus the RV1. It is clear that RV is a good proxy for volatility perceived on the financial market. This is more accentuated with the square root process. Figure (14) shows that a linear relation between the index and the stochastic volatility variables, at contemporaneous time, may exist. These variables may be used as good proxies for the true volatility. Combining this with the results obtained in the previous chapters, about the autocorrelation strength, it may be possible to forecast the level, or at least the direction of movement, of the V2X index. Figure (15) and (16) summarizes the results as far discussed. This led to the confirmation that square root process best fits the forecast model.

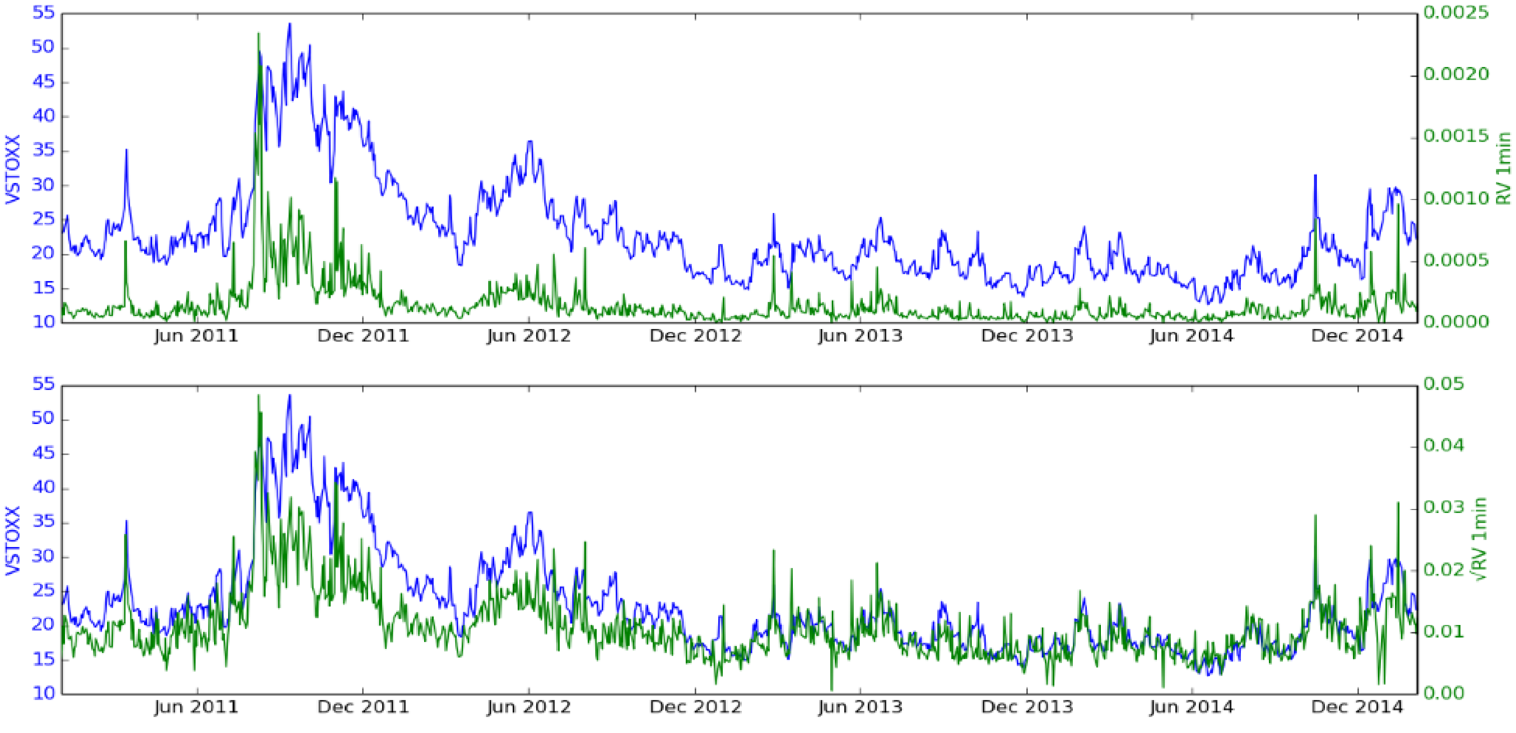


Figure 13: V2X index vs RV1 and square root process for RV1.

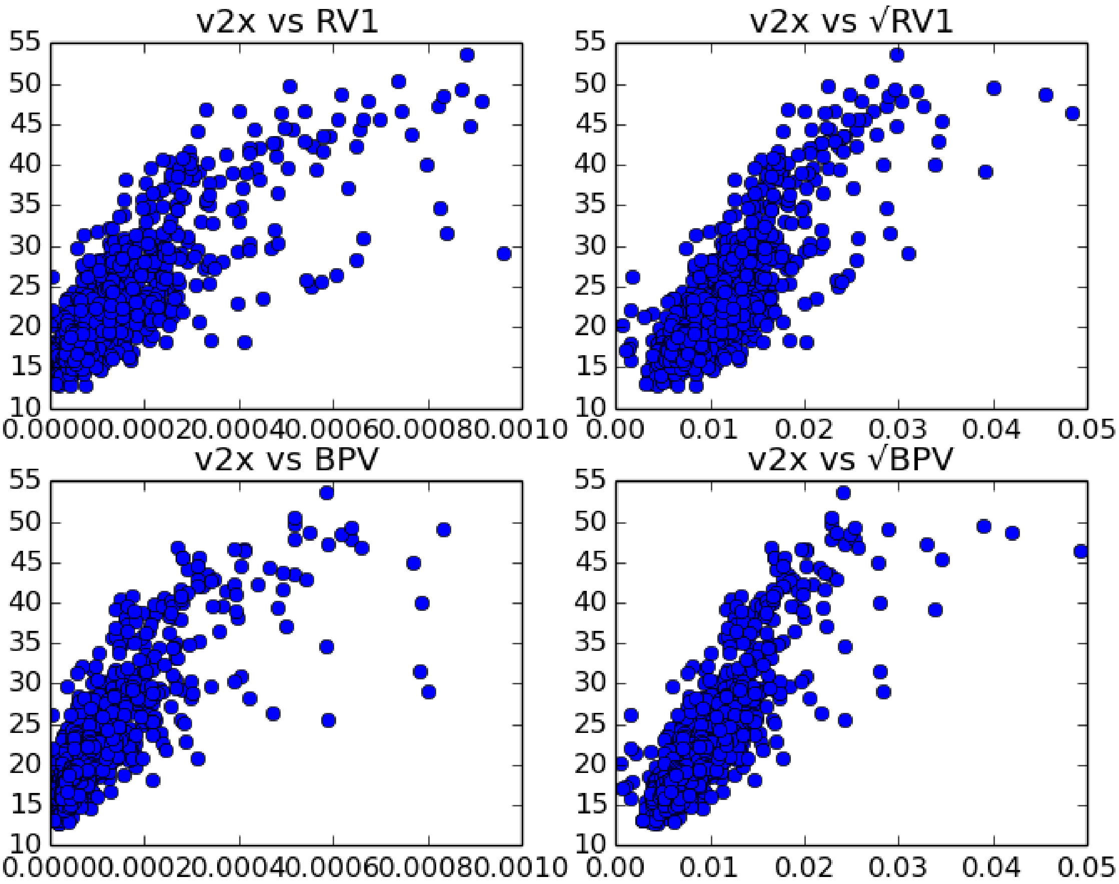


Figure 14: Scatter plots of V2X vs RV1 and BPV, and their square root process. It is possible to infer that a linear relation between these variables may exist.

Explanatory variable	*V2X ~ X_t*	*V2X ~ ln(X_t)*	*V2X ~ √X_t*
RV1	0,5610	0,5812	0,6851
BPV	0,5024	0,5937	0,6868
RV5	0,4939	0,5335	0,6227
TSRV	0,4733	0,4728	0,6034
HAR	0,8248	0,8258	0,8746

Figure 15: Same-time relation between the V2X and the selected stochastic variables. On the rows, the explanatory variables are pointed out, while the column indicates the regression specific function applied to those variables. Cells indicate the adjusted R-squared of these regressions.

Explanatory variable	$V2X_t \sim X_{t-1}$	$V2X_t \sim \sqrt{X_{t-1}}$	$V2X_t \sim \sqrt{X_{t-1}}, \sqrt{X_{t-2}}$
RV1	0,5430	0,6603	0,7284
BPV	0,4848	0,6595	0,7081
RV5	0,4759	0,5998	0,6940
TSRV	0,4554	0,5810	0,6819
HAR	0,8042	0,8501	-

Figure 16: Same analysis of the previous table, but on the first lag of the explanatory variables. HAR model has been analyzed only on the first lag and not also the second, since it already contains previous lag information.

The trading strategy followed is to forecast through an AR model the level of RV1, which was shown to have the higher adjusted R-squared with the V2X index, and then use the sign of this forecast as a trading signal (buy or sell) for the V2X. Results are computed assuming that is possible to buy or sell one contract of V2X at the price given by its level, without considering any transaction costs. Returns are computed as if on each day 1$ is invested, and at the end of the day the position is closed. The next day another contract of 1$ is traded (the proceeds are not reinvested to avoid timing biases). Figure (17) displays the result of this strategy. The benchmark strategy consists into trading the V2X depending on its previous behaviour, thus buy if it has risen, sell otherwise. The strategies involving stochastic volatility consist into forecasting the index with √RV, and then buy if a growth is foreseen, sell otherwise. The last strategy adds to the previous a momentum component, which accounts for mean reversion. If the level of the V2X has risen too much in percentage terms, it is possible to sell the index, in order to benefits from its mean-reverting behaviour. More in detail, if in the previous two days the level of the index rose by a certain threshold, regardless from the forecast, the strategy is to sell the index. The choice of a threshold is not so trivial. Selecting different levels, even close each other, leads to very different results. Thus the results displayed here are the mean of these strategies with threshold set to all values between 10% and 30% with 1% increment. It appears that using a rule to select the threshold, which consist into selecting only thresholds with the highest percentage of adjacent opposite sign, the subset of found thresholds is able to give always a positive return, with a mean (across thresholds) of 38,42% during the overall period.

Strategy 1	*same sign ratio*	*overall return*
V2X	46,21%	-223,14%
RV1	48,86%	-31,41%
$\sqrt{RV1}$	48,86%	-45,13%
HAR	48,23%	-100,32%

Strategy 2 (momentum)	*overall return*	*annualized return*
V2X	1,75%	0,56%
RV1	32,78%	9,58%
$\sqrt{RV1}$	19,26%	5,85%
HAR	14,60%	4,50%
BPV	-68,41%	-31,06%
$\sqrt{BPV}$	-76,99%	-37,76%

Figure 17: These two tables shows the results of the strategy. In the first table only the simple forecasting signal is used, while in the second table is applied also the mean-reverting momentum. "Same sign ratio" is the percentage of time of correct direction (up, down) forecast.

5.4 Application – VaR

The second application consist into computing the VaR capital requirement, comparing stochastic variance measures and a traditional volatility model, the EWMA. Parametric models and Monte Carlo simulation, both at 1-day and 10-days horizons are applied. The benchmark model is the EWMA, which, as stated in Hull (2012), is widely used for risk management purposes, due to its efficiency and light computational efforts. The EWMA model is described recursively, as follows:

$$\sigma_t^2 = \lambda\sigma_{t-1}^2 + (1-\lambda)r_{t-1}^2$$

with parameter λ set equal to 0.94, as discusses in Hull (2012). The investigation is conducted as an *ex-post* analysis, to back-test the different methods used.

For the parametric model, it is assumed that the standardized returns distribute as a Normal (0,1). As discussed in the previous paragraphs, the empirical distribution of returns divided by stochastic volatility is roughly similar to a Standard Normal, thus it is reasonable to expect that the confidence interval for future returns is $[r_{t+1} - \alpha\hat{\sigma}_{t+1}$, $r_{t+1} + \alpha\hat{\sigma}_{t+1}$], with α as the prudential quantile of a Normal (0,1) distribution. Since the analysis is conducted for a 99% VaR, the α parameter is set to 2.326. The next-period volatility, in the case of stochastic variables, is computed through a rolling window AR process forecast.

For the Monte Carlo simulations, the model simulated is the following:

$$r_{t+1} = dp_{t+1} = \varepsilon_{t+1}\hat{\sigma}_{t+1}$$

with an implied drift rate of zero, and the volatility proxy forecasted through a rolling window AR model of the stochastic variables, as explained above.

5.4.1 1-day horizon

Figure (18) shows the results for the parametric model. Exceptions and capital requirement are displayed. Exceptions are the ratio of days where the prudential capital was not sufficient to cover the daily loss occurred. Since the VaR is set to a 99% level, a correctly specified model should observe about 1% of exceptions. If exceptions are too much, then the model is underestimating the risk. If exceptions are less the 1%, then it overstates the risk, and the capital required is excessive. The average capital required is simply the average capital that should be set as reserve by prudential regulation. It is equal to the average daily VaR estimated by the model, and is indicated as a percentage of the total capital invested. It can be thought as the average daily return loss with 99% confidence. It emerges that none of the models is able to catch the required quantile. RV1 performs as the EWMA, but requires less capital.

Variable	*exceptions*	*average capital required*
EWMA	1,89%	2,50%
RV1	1,89%	2,44%
BPV	3,40%	2,11%
HAR	3,03%	2,32%

Figure 18: Results for the parametric model at 1-day horizon.

Figure (19) shows the results for Monte Carlo simulations for RV1. The simulations are run by drawing ε from different distributions, and then compared. In the first simulation ε is drawn from a Normal (0,1) distribution. Results show how this specification does not improve the analysis. The Normal distribution seems to not catch the probability of extreme bad events to happen. Modelling ε as a Normal distribution, underestimates the probability of extremes, which empirically happens more frequently. A solution is to model ε as a Student-t distribution or as a Laplace distribution, since they have fatter tails. The choice of the Student-t degrees of freedom, *v*, is derived in two ways: the first method derives it by maximum likelihood, the second method finds the parameter such that the kurtosis of the resulting Student-t distribution matches the empirical kurtosis of returns. Empirical excess kurtosis is observed to be about 2.51, which implies a degree of freedom parameter of 6.39. The Laplace distribution has been selected due to its similarity with empirical return distribution, including the excess kurtosis, which is of 3 units. The results are clearly in favour of the matched Student-t and Laplace distributions for the ε distribution, since provide always exceptions ratio closer to the 1% level. It can be noticed as results are better than parametric models, clearly in favour of stochastic models. Better results are obtained if the forecast is conducted on the normal (non "square root") process, that is forecasting RV and then taking the square root, rather than applying directly the forecast analysis to the square root process.

Variable	*ε Normal*	*ε Student-t (fitted)*	*ε Student-t (kurtosis matched)*	*ε Laplace*
RV1 (forecast of RV)	1,89%	1,39%	0,63%	0,76%
RV1 (forecast of √RV)	2,40%	1,64%	1,13%	1,39%
BPV (forecast of BPV)	3,28%	2,14%	1,13%	1,51%
BPV (forecast of √BPV)	4,29%	2,52%	1,39%	1,64%
HAR (forecast of HAR)	2,90%	1,89%	0,76%	1,39%
HAR (forecast of √HAR)	3,15%	2,02%	1,01%	1,13%

Figure 19: Results for the Monte Carlo simulations at 1-day horizon.

5.4.2 10-days horizon

This last paragraph, shows results at a 10-days horizon. For the parametric model, the 10-days EWMA volatility is simply the daily volatility multiplied by square root of 10, since it is supposed to remain constant. For the stochastic variables, the volatility over the 10 days is found by forecasting the volatility for each day of the 10-day period. In formulas:

$$\sigma_{t,10d} = \sqrt{\sum_{i=1}^{10} RV1_{t+i}}$$

with $RV1_{t+i}$ given by an AR rolling window forecast of:

$$RV1_{t+i} \sim [1\ RV_t\ RV_{t-1}]$$

The daily RV process is estimated through each 10s of the following days, and then is summed. Figure (20) displays the results. EWMA, on average, is closer to the 1% level, and stochastic volatility measures requires less capital. RV1 is both correct in terms of exceptions, and requires an average capital very close to EWMA level. The (average) capital here required is the capital over a window of 10 days, that is the average loss expected within 10 days with 99% confidence.

Variable	*exceptions*	*average capital required*
EWMA	1,79%	7,88%
RV1	1,28%	7,89%
BPV	2,68%	6,85%
HAR	2,04%	7,40%

Figure 20: Results for the parametric model at 10-days horizon.

For the Monte Carlo simulations, the procedure consists, in each simulation, into drawing 10 different returns for each of the following 10 days, and then summing results over this window. Results with Monte Carlo simulation are better than parametric model, especially for ε drawn from matched Student-t and Laplace distribution.

Variable	*ε Normal*	*ε Student-t (fitted)*	*ε Student-t (kurtosis matched)*	*ε Laplace*
RV1 (forecast of RV)	1,28%	0,77%	0,64%	0,77%
RV1 (forecast of $\sqrt{RV}$)	2,04%	0,89%	0,64%	1,15%
BPV (forecast of BPV)	2,55%	1,66%	1,02%	1,66%
BPV (forecast of $\sqrt{BPV}$)	3,57%	2,42%	1,40%	2,42%
HAR (forecast of HAR)	2,04%	1,53%	0,51%	1,79%
HAR (forecast of $\sqrt{HAR}$)	2,17%	1,79%	0,51%	2,04%

Figure 21: Results for the Monte Carlo simulations at 10-days horizon.

6. CONCLUSIONS

The analysis conducted so far, allows to understand the benefits of using stochastic volatility models, over traditional models. As shown, these models seem to fit better the unobservable volatility process, which drives returns of financial assets. They provide good proxies to use in volatility models.

It was shown how stochastic models have long memory processes. The AR model seem to fit quite well the empirical returns, with significant coefficients for the first lags. The autocorrelation function seems to slowly decay until about the 10th lag, over which is remain stable, still different from zero. This effect is more pronounced as the square root process is analysed. The square root process for the AR model, provides better adjusted R-squared, and better forecasts. It does not imply that the square root process is a better proxy for the true volatility process, especially for empirical application purposes. It may emphasize the magnitude of peaks. This effect can be noticed in the VaR application, where normal processes behave always better than their square root counterpart.

The best proxies appeared to be RV1 and BPV. The former is the best representative of RV category, compared at different sampling horizons, both in terms of self-predicting ability and explanatory power for the volatility process. The latter reduces the magnitude of observed peaks, which translates into better self-predicting power. The HAR-RV model, finally, seems to be able to enhance self-forecasting ability of RV1.

In assessing the ability of fitting the true variance process, these models have been tested with the V2X index, which designate the short term implied volatility for EUROSTOXX. It emerged that RV1, BPV and HAR produced good adjusted R-squared both at same-time and at one-lag analysis. This should mean that those variables may be used to forecast the future level of the index, and thus a trading strategy may be built using stochastic volatility. A rolling window forecast was used to predict the one-step-ahead V2X level, and the signs of these movements were used as signal to buy or to sell the index. Preliminary results were against any kind of strategy, but adding a momentum component to this strategy allowed to produce positive returns. Stochastic volatility models, in particular RV1, are able to beat the benchmark model, built upon the previous

V2X level. Even if square root processes seem to better catch the movements of the V2X, the highest returns from this strategy are obtained through forecasting the "normal" level of stochastic variables, and then applying the square root to those results (instead of conducting the forecasting process directly on the square root variables).

The last application has been a VaR at 1% level back-testing. Models compared were the traditional EWMA and stochastic volatility models RV1, BPV, HAR. The EWMA is the model that performs better on average, since exceptions are closer to the 1% level. The RV1 is the only stochastic model that achieved better results than EWMA, both at 1-day and 10-days horizons. It would be expected that, since standardized (with stochastic volatility variables) returns distributes close to a Standard Normal distribution, the number of exceptions was close to 1%. The level, instead, remains close but higher in all cases, maybe because the sample size was not big enough, or because it has been underestimated the probability of extreme events to happen. The Monte Carlo simulation method was established in order to account for this issue. Results have been clearly improved by generating the returns patterns with higher kurtosis. In the specific case, a matched-kurtosis Student-t and a Laplace distributions were used. These generating processes for returns allow to obtain closer level for exceptions, both at 1-day and 10-days horizons.

It can be finally confirmed the efficiency of stochastic volatility models over traditional ones. It is true that they require higher computational efforts, but, with an efficient software and an efficient code, the computing time can be drastically reduced and compared to traditional methods. The use of Julia language improves significantly the management of these big data. It can be a powerful tool, but it need to be developed and integrated with more advanced functions and packages. Stochastic volatility models seem to provide more accurate estimates of the true volatility process, principally because they try to exploit all the possible information available through HFD. This confirms the theory that big data are useful to better understand the behaviour and the true nature of certain stochastic processes, and maybe forecast their probable future realizations.

A. CODE

Loading data

In [1]:

```
#Pkg.update()
using DataFrames, Distributions, Optim, JuMP, PyPlot, Gadfly;
```

In [2]:

```
#loading cleaned data of eurostoxx.
data=readtable("/home/juser/julia-gdrive/eustoxxclean.csv.gz",eltypes=
[Float64,UTF8String]);

#data of serialized time (number format). It was created in a previous
 code and the saved.
data1=readtable("/home/juser/julia-gdrive/date_min_num.csv.gz");
```

In [3]:

```
price_all=Array{Float64}(data[:price]);

datetimem_num=Array{Int64}(data1[:time_minute]);
datetimem_once=union(datetimem_num,datetimem_num);
```

In [4]:

```
#this was the code used to create the serialized time vector. Once cre
ated that vector there is no
#need to re-run this old code


#datetime_str=Array{UTF8String}(data[:datetime]);
#datetimem_str=map(x->x[1:16],datetime_str); #just minutes

#datetime=DateTime(datetime_str,"y-m-dTHH:MM:SS.sss");
#datetimem=DateTime(datetimem_str,"y-m-dTHH:MM");
#date=Date(datetime);

#date_once=union(date,date);
#datetimem_once=union(datetimem,datetimem);

#date_0=datetimem[1]; #2011-01-03 09:00
#datetimem_num=map(x->(x-date_0)/60000,datetimem); #this is the distan
ce (in minutes) from the observation 0
#datetimem_num_int=map(Int,datetimem_num);
#datetimem_num_once=union(datetimem_num_int,datetimem_num_int);
```

In [5]:

```
#re-creating the old date array (in dates type)
date_0=DateTime(2011,01,03,09);
```

```
date_all=date_0+map(Dates.Minute,datetimem_once); #use datetimem_num for all dates
date_once=union(Date(date_all),Date(date_all));
```

In [ ]:

Creating the grid

- time-grid

In [6]:

```
#this function creates a time grid of equally spaced 1 minutes time
#(just for 1 day, just for trading hours 9:00-17:30)

function create_timegrid(time::Date,start::AbstractString="T09:00:00",finish::AbstractString="T17:30:00")
    st=DateTime(string(time,start));
    fi=DateTime(string(time,finish));
    return collect(st:Dates.Minute(1):fi)
end
```

Out[6]:

```
create_timegrid (generic function with 3 methods)
```

In [7]:

```
grid_length=(17-9)*60+30+1;
tgrid=Array{DateTime}(grid_length*length(date_once));
```

In [8]:

```
#filling the vector
for i=1:length(date_once)
    tgrid[(1:grid_length)+grid_length*(i-1)]=create_timegrid(date_once[i]);
end
```

In [9]:

```
#time grid in minutes
tgrid_num=map(Int,map(x->Int(x-date_0)/60000,tgrid));
```

- finding position

In [10]:

```
#this code does the following:
# - takes the difference of datetimes (in numeric form)
# - non-zero elements are where time has changed, thus we need the position of time before changement
# - using sparse matrix to identify non-zero elements improve speed
# - rowvals takes the position of non-zero element of the sparse matrix
```

```
@time pos=rowvals(sparsevec(diff(datetimem_num)));
```

```
  0.197149 seconds (134.33 k allocations: 83.849 MB, 12.97% gc time)
```

In [11]:

```
#"pos" points the position of the observed_last_price for each date-ti
me element in "datetimem_once"
pos=[1;pos]; #adding the first element because there is nothing before
 9:00 on the first day

#double check:
#length(pos)==length(datetimem_once);
```

- filling the missing points

In [12]:

```
#which are missing observations?
#"there are $(length(tgrid_num)-length(datetimem_once)) missing observ
ation to fill"
```

In [13]:

```
#missing minutes
missing_min=setdiff(tgrid_num,datetimem_once);
#missing minutes position
missing_min_pos=map(x->find(tgrid_num.==x)[1],missing_min); # the "[1]
" is to access to the array given by find();
```

In [14]:

```
#inserting the previous observation
for miss_pos in missing_min_pos
    insert!(pos,miss_pos,pos[miss_pos-1]);
end
```

In [15]:

```
#double check:
#length(pos)==length(tgrid_num)
```

- creating price grid

In [16]:

```
price_grid=Array{Float64}(price_all[pos]);
```

- dividing in days

In [17]:

```
ndays=length(date_once);
price_mat=Array{Float64}(grid_length,ndays);
```

```
price_mat=reshape(price_grid,grid_length,ndays);
```

- return grid

In [18]:

```
#this function is an apply-like function that applies a function to th
e columns of a matrix
function map_colwise(func::Function,data::Array{Float64,2})
    final_nrow=length(func(data[:,1]));
    ncol=size(data,2);
    func_data=Array{Float64,2}(final_nrow,ncol)
    for col=1:ncol
        func_data[:,col]=func(data[:,col]);
    end
    return func_data
end
```

Out[18]:

```
map_colwise (generic function with 1 method)
```

In [19]:

```
#1-minute return over the price mat
ret_1m=map_colwise(x->diff(log(x)),price_mat);
```

In [20]:

```
#return over x minutes interval, given a matrix of prices
function ret_xmin(price::Array{Float64,2},min::Int)
    grid_length=size(price,1);
    price_xmin=price[1:min:grid_length,:];
    ret_mat=map_colwise(x->diff(log(x)),price_xmin);
    return ret_mat
end
```

Out[20]:

```
ret_xmin (generic function with 1 method)
```

In [21]:

```
#1-minute return over the price mat
ret_5m=ret_xmin(price_mat,5);
```

In [ ]:

Stochastic variance variables

preliminary functions

In [22]:

```
#function for the AR(p) coefficients calculation
function x_lag(x::Array{Float64},p::Real,constant::Bool=true)
    #the function picks the time series and divide it into its lag
    #it assumes that more recent observations are at the end
    #it returns X_0, the array of latest observations, and X_lag, the
matrix of lags
    p+=1;
    l=length(x);
    X=Array{Float64}(l-p+1,p);
    for i=1:p
        X[:,i]=x[(p:end)-i+1];
    end
    X_0=X[:,1];
    if constant==true
        X_lag=[ones(l-p+1,1) X[:,2:end]];
    else
        X_lag=X[:,2:end];
    end
    return X_0, X_lag
end
```

Out[22]:

```
x_lag (generic function with 2 methods)
```

In [23]:

```
function loglike(y::Array{Float64},x::Array{Float64},beta::Array{Float
64};sigma2::Real=1.0,result::ASCIIString="llbic")
    #this function computes loglikelihood and bic
    n,p=size(x);
    u=y-x*beta;
    dist=Normal(0,√sigma2);
    contributions=logpdf(dist,u);
    loglikelihood=sum(contributions);
    bic=-2*loglikelihood+log(n)*p;
    if result=="ll"
        out=loglikelihood
    elseif result=="bic"
        out=bic
    else
        out=loglikelihood,bic
    end
    return out
end
```

Out[23]:

```
loglike (generic function with 1 method)
```

In [24]:

```
#this function return the best lag, chosen by BIC.
function best_p(X::Array{Float64})
    l=length(X);
    p_opt=floor(0.02*l);
    P=Int(max(3,min(10,p_opt))); #minimum 3 lags, maximum 10 lags
    bic=Array{Float64,1}(P);
    for p=1:P
        y,x=x_lag(X,p);
        β=x\y;
        bic[p]=loglike(y,x,β,result="bic");
    end
    pp=find(bic.==minimum(bic));
    return pp
end
```

Out[24]:

```
best_p (generic function with 1 method)
```

In [25]:

```
function R²(y::Array{Float64},x::Array{Float64},β::Array{Float64};adju
sted::Bool=true)
    n,p=size(x);
    res=y-x*β;
    ydev=y-mean(y);
    R2=1-res'res/ydev'ydev;
    adjR2=1-(1-R2)*(n-1)/(n-p-1);
    if adjusted==true
        return adjR2[:]
    else
        return R2[:]
    end
end
```

Out[25]:

```
R² (generic function with 1 method)
```

In [26]:

```
#creating the type linReg to avoid excessive creation of variables.
#for now, it is sufficient to access to the beta coefficients and R sq
uared
type linReg
    β::Array{Float64}
    R2::Float64
    R2adj::Float64
end

function linearOLS(y::Array{Float64}, x=[]; constant::Bool=true, p::Re
al=4)
    #if just the y is specified, then linearOLS constructs an AR(p) mo
del
    #if also x is provided, is the case of simple OLS
    if x==[]
        y,x=x_lag(y,p,false)
    end

    sy=length(y);

    if typeof(x)==Array{Float64,1}
        n=length(x);
        p=1;
    else
        n,p=size(x);
    end

    if n!=sy
        error("vectors must have same length")
    end

    if constant==true
        x=[ones(sy,1) x];
    end

    β=x\y;

    #standard errors of betas => o=homosk. e=heterosk.
    #u=y-xβ;
    #σ2=u'u/(n-length(β)-1);
    #VCo=σ2*inv(x'x);
    #ux=u'x;
    #VCe=(x'x)\ux'ux/(x'x);
    #seo=diag(VCo);
    #see=diag(VCe);

    u=y-x*β;
    ydev=y-mean(y);
    R2=1-u'u/ydev'ydev;
    R2adj=1-(1-R2)*(n-1)/(n-p-1);

    return linReg(β,R2[1],R2adj[1])
end
```

Out[26]:

```
linearOLS (generic function with 2 methods)
```

stochastic variables

In [27]:

```
#squared returns
ret_sq=ret_1m.^2;
#absolute returns
ret_abs=abs(ret_1m);


#RV from grid
RV_1m=sum(ret_sq,1)';

RV_5m=sum(ret_5m.^2,1)';


#BPV
ret_bpv=ret_abs[1:end-1,:].*ret_abs[2:end,:];
BPV=π/2*sum(ret_bpv,1)';


#TSRV
#it needs n n-minutes-return RVs. Choose 5 minutes
ret5mRV=zeros(5,size(price_mat,2)); #this will be the array containing on each row a time series of RV on a possible 5 min interval
grid5m_length=Array{Int8}(5); #this is the size of the grid for each possible 5 min interval
for i=1:5
    ret5m=ret_xmin(price_mat[1:end-i+1,:],5);
    ret5mRV[i,:]=sum(ret5m.^2,1);
    grid5m_length[i]=size(ret5m,1);
end
RV_tsrv=mean(ret5mRV,1)'; #this is still biased
avg_length=mean(grid5m_length);
microstr_bias=avg_length/grid_length*RV_1m; #it should be RV of sparse data, but also 1 minute grid should work well;
TSRV=RV_tsrv-microstr_bias; #bias is removed;
```

In [28]:

```
#RV using all prices available
function RV_daily(date_day::Union{Int64,Date},dates_all::Union{Array{Int64},Array{Date,1}},price_all::Array{Float64,1})
    #this function calculates the RV for the specific day, indicated by date_day
    #it starts from the whole array of dates and prices
    pos=find(dates_all.==date_day);
    price_day=price_all[pos];
    ret_day=diff(log(price_day)); #no matters if they need to be changed of sign, since they will be squared
    RV=ret_day'ret_day; #sum of squares
    return RV[1]    #[1] is used to convert Array->scalar
end

RV=map(x->RV_daily(x,Date(date_all),price_all),date_once);
```

In [29]:

```
# HAR-RV
function period_mean(data::Array{Float64},period::Int64)
    #it's a simply moving average

    l=length(data);
    data_out=Array{Float64}(l-period+1);
    [data_out[i]=mean(data[(1:period)+i-1]) for i=1:l-period+1];
    return data_out
end

RVw=period_mean(RV_1m,5); #weekly mean
RVm=period_mean(RV_1m,20); #monthly mean
maxl=length(RVm);
HAR=[RV_1m[end-maxl+1:end] RVw[end-maxl+1:end] RVm];
```

In [ ]:

Statistical properties

In [30]:

```
#whole time series
ret_vec=ret_1m[:]; #all the returns on the same vector, no matter of the day
ret_norm=(ret_vec-mean(ret_vec))./√var(ret_vec); #standardized returns using standard deviation;
```

In [31]:

```
#daily time series of returns
ret_day=(log(price_mat[end,:])-log(price_mat[1,:]))[:];
ret_day_norm=(ret_day-mean(ret_day))/√var(ret_day); #standardized returns using standard deviation;
ret_day_norm_rv=ret_day./√RV_1m; #standardized returns using RV;
```

In [ ]:

Forecast analysis

In [32]:

```
#obsolete function
function forecast_AR_rolling(data_all::Array{Float64},date_all::Array{Date}=date_once;starting_date::Date=Date(2012,01,01))
```

```
    #this function creates a rolling window of 1 year (256 observation)
    #then compute β_ols and makes the forecast for the next trading day of the variable

    ld=length(data_all);
    idx=findlast(date_all.<starting_date); #from here, 1 step ahead forecasts will start

    forecasts=Array{Float64,1}(ld-idx);

    for i=idx:ld-1

        data_window=data_all[i-255:i];
        β=linearOLS(data_window).β
        lb=length(β)-1;
        forecasts[i-idx+1]=([1;data_window[end-lb+1:end]]'β)[1];

    end

    return forecasts

end
```

Out[32]:

```
forecast_AR_rolling (generic function with 2 methods)
```

In [33]:

```
function forecast_rolling(y::Array{Float64},x::Array{Float64};
    date_begin::Date=Date(2012,01,01),date_all::Array{Date}=date_once,window_length::Int64=256)

    #this function does the following:
    # it returns the array forecasts which cointains the forecasts at time t
    # the forecast for time t is made as follows:
    # -  take window of 256 observations from (t-255-2:t-2) for x and window (t-255-1:t-1) for y
    # -  makes a regression between those two variables, catch the beta
    # -  uses this beta on x_t-1 to have a forecast for y_t

    idx=length(date_all)-findlast(date_all.<date_begin); #out of sample array length

    forecasts=Array{Float64,1}(idx);
    for i=1:idx

        #out of sample window goes from (end-idx+1) to end

        x_wind=[ones(window_length) x[(end-window_length+1:end)-idx+i-2,:]]; #x is taken one lag before y
        y_wind=y[(end-window_length+1:end)-idx+i-1];
        β=x_wind\y_wind;
        forecasts[i]=([1 x[end-idx+i-1,:]]*β)[1];

    end

    return forecasts
end
```

Out[33]:

```
forecast_rolling (generic function with 1 method)
```

In [ ]:

example of analysis

In [34]:

```
#self-predicting ability
p=max(best_p(HAR),3);p=p[1];
HAR_lm=linearOLS(HAR,p=p);
HAR_lm.R2adj;
```

In [ ]:

Predicting the VSTOXX

In [35]:

```
datav=readtable("/home/juser/julia-gdrive/sx5e.csv"
                ,eltypes=[UTF8String,Float64,Int64,Float64,Float64,Float64,Float64,Int64]
                );
```

In [36]:

```
date_v=reverse(Date(datav[:Date],"d/m/y"));
volume_ind=reverse(datav[:SX5E_PX_VOLUME]); #volume field on bloomberg of the EU STOXX index;
volume_fut=reverse(datav[:VG1_PX_VOLUME]); #volume of the nearest future;
v2x=reverse(datav[:V2X]); #VIX index on the EU STOXX;
datav2=Array{Float64}([volume_ind volume_fut v2x]);
```

In [37]:

```
first=findfirst(date_v.==date_once[1])[1];
last=findlast(date_v.==date_once[end])[1];
```

```
datav2=datav2[first:last,:];
date_v=date_v[first:last,:];
```

In [38]:

```
#finding missing dates
missing_date=setdiff(date_once,date_v);
missing_date_pos=map(x->findlast(date_once.<x),missing_date);
```

In [39]:

```
function insertrow(data,idx::Int64,ins)
    #add row after idx
    d=[data[1:idx,:]; ins; data[idx+1:end,:]];
end
```

Out[39]:

```
insertrow (generic function with 1 method)
```

In [40]:

```
for pos in missing_date_pos
    datav2=insertrow(datav2,pos,datav2[pos,:]);
    date_v=insertrow(date_v,pos,date_once[pos+1]);
end
```

In [41]:

```
volume_ind=datav2[:,1];
volume_fut=datav2[:,2];
v2x=datav2[:,3];
v2x_ret=diff(log(v2x));
```

In [ ]:

regression - same time

In [42]:

```
#regression analysis
#res vector contains the adj R2 of regression of v2x on x, log(x), √x
x=RV_1m; #<----change this variables for other results;
res=[
    linearOLS(v2x,x).R2adj;
    linearOLS(v2x,log(x)).R2adj;
    linearOLS(v2x,√x).R2adj
];
#res;
```

In [43]:

```
#HAR framework - same time
lhar=size(HAR)[1];
reg_v_har=linearOLS(v2x[end-lhar+1:end],√(HAR)); #<----change the func
tion for other results
#reg_v_har.R2adj;
```

In [ ]:

predicting power - regression on first lag

In [44]:

```
x=RV_1m; a=[]; #<----change the x for other results;
#tested regressions: v2x_t -> x_(t-1) ; v2x_t -> √x_(t-1) ; v2x_t -> √
x_(t-1) √x_(t-2)
#reg_lag cointains adj. R2 of these regressions

yy=v2x[2:end];
xx=x[1:end-1];
reg_lag=linearOLS(yy,xx);
push!(a,reg_lag.R2adj);

xx=√x[1:end-1];
reg_lag=linearOLS(yy,xx);
push!(a,reg_lag.R2adj);

yy=v2x[3:end];
xx=√[x[2:end-1] x[1:end-2]];
reg_lag=linearOLS(yy,xx);
push!(a,reg_lag.R2adj);

#reg_lag;
```

In [45]:

```
#HAR
yy=v2x[end-lhar+2:end];
xx=√HAR[1:end-1,:]; #<----change the function for other results
reg_lag=linearOLS(yy,xx);
#reg_lag.R2adj;
```

In [46]:

```
#AR of just v2x
reg_AR=linearOLS(v2x,p=1);
#reg_AR.R2adj;
```

In [ ]:

assessing predicting power

In [47]:

```
function forecast_analysis(y::Array{Float64},x::Array{Float64};window_
length::Int64=256,date_begin::Date=Date(2012,01,01))

    #this function takes as input:
    # - y      -> the array o be forecasted
    # - x      -> the explanatory variable
    #and it produces a rolling window forecast for y, from x matrix. resulting output are:
    # - yf     -> the realized price array
    # - xf     -> the forecasted price array (using rolling window forecast)
    # - ret    -> the return time series of a strategy which consist into buying if the predicted price should rise
    # - sign   -> the percentage of time of same sign of movement (realized vs forecasted price)
    # - ssdiff -> sum of squared difference between forecast and realized price


    forecasted_price=forecast_rolling(y,x,window_length=window_length,date_begin=date_begin);
    realized_price=y[end-length(forecasted_price)+1:end];
    forecast_difference=realized_price-forecasted_price;
    ssq_difference=sum(forecast_difference.^2);

    forecasted_ret=diff(log(forecasted_price));
    realized_ret=diff(log(realized_price));
    caught_sign=sum(sign(realized_ret).==sign(forecasted_ret))/length(realized_ret);

    strategy_ret=realized_ret.*sign(forecasted_ret);

    dict=Dict(:xf=>forecasted_price,:yf=>realized_price,:ret=>strategy_ret,:sign=>caught_sign,:ssdiff=>ssq_difference);

    return dict
end
```

Out[47]:

```
forecast_analysis (generic function with 1 method)
```

In [48]:

```
f=forecast_analysis(v2x[2:end],v2x[1:end-1],window_length=255) #need to reduce the window length or to start from a later date;
```

In [49]:

```
#to compute the overall return of the strategy use the following codes:

# - 1$ invested at beginning and proceeds reinvested
ov_ret=cumprod(f[:ret]+1);

# - 1$ invested on each day and no proceeds reinvested
```

```
ov_ret=sum(f[:ret]);
```

In [50]:

```
#return with 10% threshold

a=f[:yf];
b=f[:xf];
ra=diff(log(a));
rb=diff(log(b));
tdret=((ra[1:end-1]+ra[2:end]).>0.1)*1; #two days return trigger: true if previous two days returns sum is above +10%

pos=find(tdret)+1; #this is the position on the ret array of the day whose same day + previous day return is >10%

sra=sign(ra);
srb=sign(rb);
srb[pos+1]=-1.0; #sell the next day the trigger occurred
sum(ra.*srb);
```

In [51]:

```
#choose length=236 for HAR, 255 for other, or change date
f=forecast_analysis(v2x,RV_1m,window_length=236); #<----change here variable
predicted=f[:yf];
realized=f[:xf];
ret_p=diff(log(predicted));
ret_r=diff(log(realized));
sra=sign(ret_p);
srb=sign(ret_r);

#this is the average return of choosing as thresholds all levels between 0.1:0.01:0.3
tot_ret=[];
for i=0.1:0.01:0.3
    tdret=((ret_p[1:end-1]+ret_p[2:end]).>i)*1;
    pos=find(tdret)+1;
    srb=sign(ret_r);
    srb[pos+1]=-1.0;
    tot_ret=push!(tot_ret,sum(ret_p.*srb))
end

#mean(tot_ret)
```

In [52]:

```
#choosing specific thresholds
```

```
#choose only those which have higher % of adjacent negative return after peaks

yret=yret=diff(log(forecast_analysis(v2x,RV_1m,window_length=236)[:yf]));
prev_ret_sum=yret[1:end-2]+yret[2:end-1];
next_ret=yret[3:end];
seq=[prev_ret_sum next_ret];
seq70=[]; #70 since only if % is > then 70% are selected
for i in collect(0.08:0.01:0.29)
    pos=find(prev_ret_sum.>i);
    a=seq[pos,:];
    pneg=sum(a[:,2].<0)/length(pos);
    #@printf("%0.2f => %0.2f\n",i,pneg)
    if pneg>0.7
        push!(seq70,i)
    end

end

#seq70
```

In [53]:

```
#testing
f=forecast_analysis(v2x,RV_1m);
a=f[:yf];
b=f[:xf];
ra=diff(log(a));
rb=diff(log(b));
sra=sign(ra);
srb=sign(rb);
s=[];
for i in seq70
    tdret=((ra[1:end-1]+ra[2:end]).>i)*1;
    pos=find(tdret)+1;
    srb=sign(rb);
    srb[pos+1]=-1.0;
    s=push!(s,sum(ra.*srb))
end

#mean(s)
```

In [ ]:

VaR computation

In [54]:

```
#analysis will start on 01/01/2012
idx_beg=findfirst(date_once.>=Date(2012,01,01));
idx_end=length(date_once)-idx_beg;
```

In [55]:

```
#benchmark: constant volatility
price_day=price_mat[end,:][:]; #closing price
ret_day=[0;diff(log(price_day))];
cvol=√var(ret_day); #(constant volatility) daily standard deviation of
 returns;
```

In [56]:

```
#normailization with SV
nretRV=ret_day./√RV_1m;
#quantile(nretRV[:],0.01)
#kurtosis(nretRV) # excess kurtosis is near zero, in effect distribute
s as normal
#skewness(nretRV)
#mean(nretRV)
#;
```

In [57]:

```
#quantile(ret_day_norm,0.01)
#quantile(nretRV[:],0.01)
```

In [ ]:

variables

In [58]:

```
#EWMA

var_is=var(ret_day[1:idx_beg-1]); #variance until 01/01/2012
λ=0.94;
var_ewma=Array{Float64}(idx_end+1); var_ewma[1]=λ*var_is+(1-λ)*ret_day
[idx_beg-1]^2;
[var_ewma[i]=λ*var_ewma[i-1]+(1-λ)*ret_day[idx_beg+i-2]^2 for i=2:idx_
end+1];
var_ewma;
```

In [59]:

```
#SV variables

#RV
y,x=x_lag(RV_1m,2,false);
var_rv=forecast_rolling(y,x,window_length=252);

#BPV
y,x=x_lag(BPV,2,false);
```

```
var_bpv=forecast_rolling(y,x,window_length=252);

#HAR
y=HAR[2:end,:]; x=HAR[1:end-1,:];
var_har=forecast_rolling(y,x,window_length=234); #maybe change the dat
e of beginning;
```

In [ ]:

parametric model 1-day

In [60]:

```
sd=√[var_ewma var_rv var_bpv var_har];  #<--- it is possible to not us
e the square root
realized_ret=ret_day[idx_beg:end];
q=quantile(Normal(),0.01);
var_loss=sd*q;                          #<--- if square root was not u
sed in line 1, then square root "sd"
exceptions=repmat(realized_ret,1,4).<var_loss;

exc=sum(exceptions,1)/length(realized_ret);
```

In [61]:

```
#var_loss is the daily capital put as reserve
mean(-var_loss,1);
```

In [ ]:

MC simulations 1-day

In [62]:

```
#simulated path: r_t = dp_t = ϵ_t ˆσ_t

#this code computes MC simulation with ϵ distributed as Normal, Studen
t-t, Laplace.
#to change variable change the first line of code
#to use the square root process use second line and not the first, and
 viceversa
#the matrix tot_paths contains all possible simulations (on its rows),
 on a given day (on columns)
#it is possible to change also the number of lags included for forecas
t (the second input of x_lag())

y,x=x_lag(RV_1m,2,false); var_rv=forecast_rolling(y,x,window_length=25
2); sd_rv=√var_rv;
#y,x=x_lag(√RV_1m,2,false); sd_rv=forecast_rolling(y,x,window_length=2
52);

realized_price=price_day[idx_beg:end];
```

```
realized_price_diff=diff(price_day)[end-idx_end:end];
npaths=10000;

# ϵ~Normal(0,1)
tot_paths=repmat(sd_rv',npaths).*randn(npaths,length(sd_rv));
var_loss=map_colwise(x->quantile(x,0.01),tot_paths)[:];
exceptions=realized_ret.<var_loss;
r1=sum(exceptions)/length(realized_ret);

# ϵ~Student-t, maximum likelihood
opt=optimize(v->-sum(logpdf(TDist(v),ret_day_norm)),1.0,10.0); v_est=o
pt.minimum;
ϵ=rand(TDist(v_est),npaths,length(sd_rv));
tot_paths=repmat(sd_rv',npaths).*ϵ;
var_loss=map_colwise(x->quantile(x,0.01),tot_paths)[:];
exceptions=realized_ret.<var_loss;
r2=sum(exceptions)/length(realized_ret);

# ϵ~Student-t, matched kurtosis
ret_k=kurtosis(ret_day); v=6/ret_k+4;
ϵ=rand(TDist(v),npaths,length(sd_rv));
tot_paths=repmat(sd_rv',npaths).*ϵ;
var_loss=map_colwise(x->quantile(x,0.01),tot_paths)[:];
exceptions=realized_ret.<var_loss;
r3=sum(exceptions)/length(realized_ret);

# ϵ~Laplace
lap_par=fit(Laplace,ret_day_norm); b_lap=params(lap_par)[2];
ϵ=rand(Laplace(lap_par),npaths,length(sd_rv));
tot_paths=repmat(sd_rv',npaths).*ϵ;
var_loss=map_colwise(x->quantile(x,0.01),tot_paths)[:];
exceptions=realized_ret.<var_loss;
r4=sum(exceptions)/length(realized_ret);

#exceptions results
[r1;r2;r3;r4];
```

In [ ]:

parametric model 10-days

In [63]:

```
#expected ewma: today's ewma
sd_ewma_10=√var_ewma*√10;
```

In [64]:

```
#RV
```

```
var_rv_1_10=zeros(length(sd_rv),10);
for i=1:10 #forecast horizons
    y,x=x_lag(RV_1m,1+i,false);
    x=x[:,i:end];
    var_rv_1_10[:,i]=forecast_rolling(y,x,window_length=245); #it coul
d be also used window_length=256-i
end
sd_rv_10=sqrt(sum(var_rv_1_10,2));

#BPV
var_bpv_1_10=zeros(length(sd_rv),10);
for i=1:10
    y,x=x_lag(BPV,1+i,false);
    x=x[:,i:end];
    var_bpv_1_10[:,i]=forecast_rolling(y,x,window_length=245);
end
sd_bpv_10=sqrt(sum(var_bpv_1_10,2));

#HAR
var_har_1_10=zeros(length(sd_rv),10);
for i=1:10
    y,x=x_lag(HAR,1+i,false);
    x=x[:,i:end];
    var_har_1_10[:,i]=forecast_rolling(y,x,window_length=235);
end
var_har_1_10[var_har_1_10.<0]=0; #sometimes forecasted value are negat
ive;
sd_har_10=sqrt(sum(var_har_1_10,2));
```

In [ ]:

In [65]:

```
sds=[sd_ewma_10 sd_rv_10 sd_bpv_10 sd_har_10][1:end-9,:]; #erasing las
t 10 obs since we don't have realized ones
realized_ret_10=[sum(realized_ret[(1:10)+i-1]) for i=1:length(realized
_ret)-10+1];
var_loss=sds*q;
exceptions=repmat(realized_ret_10,1,4).<var_loss;

exc=sum(exceptions,1)/length(realized_ret_10);
```

In [66]:

```
mean(-var_loss,1);
```

MC simulations 10-days

In [67]:

```
#this simulation uses values taken from previous computation, that is the forecasted values
#for each 10s window of days. Then, in each window, is derived the stochastic variable for each day,
#which is multiplied by a 10000x10 matrix of random errors. In order to simulate returns on each day
#these returns are then summed, and the quantile for each of these windows is taken


l10d=size(var_rv_1_10[1:end-9,:],1); #latest 10 observations don't have the observed counterparty
x=√var_rv_1_10; #<----change this value for other results

var_loss=zeros(l10d)
for i=1:l10d
    next10d=x[i,:];
    tot_paths=sum(repmat(next10d,npaths).*randn(npaths,10),2);
    var_loss[i]=quantile(tot_paths[:],0.01);
end
exceptions=realized_ret_10.<var_loss;
r1=sum(exceptions)/length(realized_ret_10);

var_loss=zeros(l10d)
for i=1:l10d
    next10d=x[i,:];
    tot_paths=sum(repmat(next10d,npaths).*rand(TDist(v_est),npaths,10),2);
    var_loss[i]=quantile(tot_paths[:],0.01);
end
exceptions=realized_ret_10.<var_loss;
r2=sum(exceptions)/length(realized_ret_10);

var_loss=zeros(l10d)
for i=1:l10d
    next10d=x[i,:];
    tot_paths=sum(repmat(next10d,npaths).*rand(TDist(v),npaths,10),2);
    var_loss[i]=quantile(tot_paths[:],0.01);
end
exceptions=realized_ret_10.<var_loss;
r3=sum(exceptions)/length(realized_ret_10);

var_loss=zeros(l10d)
for i=1:l10d
```

```
    next10d=x[i,:];
    tot_paths=sum(repmat(next10d,npaths).*rand(Laplace(lap_par),npath
s,10),2);
    var_loss[i]=quantile(tot_paths[:],0.01);
end
exceptions=realized_ret_10.<var_loss;
r4=sum(exceptions)/length(realized_ret_10);

[r1;r2;r3;r4];
```

In [ ]:

B. BIBLIOGRAPHY

Aït-Sahalia, Y., Mykland, P., Zhang, L., 2005a. *How often to sample a continuous-time process in the presence of market microstructure noise*. Review of Financial Studies, 18, 351–416.

Aït-Sahalia Y., Mykland P., Zhang L., 2005b. *A tale of two time scales: determining integrated volatility with noisy high-frequency data*. Journal of the American Statistical Association, 100, 1394-1411.

Alizadeh, S., Brandt, M., Diebold, F., 2002. *Range-based estimation of stochastic volatility models.* Journal of Finance, 57, 1047-1091.

Andersen, T. G, Bollerslev, T., Diebold, F., 2008. *Parametric and nonparametric measurement of volatility*. Cambridge, MA.: National Bureau of Economic Research.

Andersen, T. G., Sørensen, B.E., 1996. *GMM estimation of a stochastic volatility model: A Monte Carlo study*. Journal of Business & Economic Statistics, 14, 328-352.

Barndorff-Nielsen, O. E., Hansen, P. R., Lunde, A., Shephard, N., 2004. *Regular and modified kernel-based estimators of integrated variance: the case with independent noise*. OFRC Working Papers Series 2004fe20, Oxford Financial Research Centre.

Barndorff-Nielsen, O. E., Hansen, P. R., Lunde, A., Shephard, N., 2008. *Designing realized kernels to measure the ex-post variation of equity prices in the presence of noise*. Econometrica, 76, 1481-1536.

Barndorff-Nielsen, O. E., Shephard, N., 2002. *Econometric analysis of realised volatility and its use in estimating stochastic volatility models*. Journal of the Royal Statistical Society, Series B 64, 253-280.

Barndorff-Nielsen, O. E., Shephard, N., 2003. *Power and bipower variation with stochastic volatility and jumps*. Economics Series Working Papers 2003-W18, University of Oxford, Department of Economics.

Barndorff-Nielsen, O. E. and Shephard, N., 2004. *Econometric analysis of realized covariance: high frequency based covariance, regression and correlation in financial economics*. Econometrica, 72 885-925.

Bates, D. S., 1996. *Jumps and stochastic volatility: exchange rate processes implicit in deutsche mark options*. Review of Financial Studies, Society for Financial Studies, 9, 69-107.

Bauwens, L., Hautsch, N., 2006. *Modelling financial high frequency data using point processes*. Discussion Papers 2006080, Université catholique de Louvain, Centre for Operations Research and Econometrics (CORE).

Blair, B. J., Poon, S.-H., Taylor, S. J., 2001. *Forecasting S&P 100 volatility: the incremental information content of implied volatilities and high frequency index returns*. Journal of Econometrics, 105, 5-26.

Bollerslev, T., 1986. *Generalized Autoregressive Conditional Heteroskedasticity*. Journal of Econometrics, 31, 307-327.

Brownlees, C. T., Gallo, G. M., 2006. *Financial econometric analysis at ultra-high frequency: data handling concerns*. Universita` degli Studi di Firenze Dipartimento di Statistica ‘Giuseppe Parenti’. Econometrics Working Papers Archive (2006/03).

Campbell, J. Y., Lo, A. W., MacKinlay, A.C., Whitelaw, R. F., 1998. *The econometrics of financial markets*. Cambridge University Press, 2(04), 559-562.

Chib, S., Nardari, F., Shephard, N., 2002. *Markov chain Monte Carlo methods for stochastic volatility models*. Journal of Econometrics, 108, 281-316.

Christensen, K., Podolskij, M., 2007. *Realized range-based estimation of integrated variance*. Journal of Econometrics, 141, 323-349.

Chung, K., Van Ness, B., Van Ness, R., 2004. *Trading costs and quote clustering on the NYSE and NASDAQ after decimalization*. Journal of Financial Research, 27, 309-328.

Clark, P. K., 1973. *A subordinated stochastic process model with finite variance for speculative prices*. Econometrica, 41, 135-155.

Corsi, F., 2009. *A simple approximate long memory model of realized volatility*. Journal of Financial Econometrics, 7, 174-196.

Daley, D.J, Vere-Jones, D. (1988). *An Introduction to the theory of point processes*. Springer, New York.

Doob, J. L., 1942. *The Brownian movement and stochastic equations*. Annals of Math, 43, 351-369.

Engle, R. F., 1982. *Autoregressive Conditional Heteroskedasticity with estimates of variance of United Kingdom inflation*. Econometrica, 50, 987-1008.

Engle, R. F., 2000. *The econometrics of ultra-high frequency data*, Econometrica, 68, 1-22.

Engle, R. F., Russell, J.R., 1998. *Autoregressive Conditional Duration: a new model for irregularly spaced transaction data*. Econometrica, 66, 1127-1162.

Engle, R. F., Patton, A. J., 2001. *What good is a volatility model?* Quantitative Finance, Taylor & Francis Journals, 1, 237-245.

Falkenberry, T.N., 2002. *High frequency data filtering: a review of the issues associated with maintaining and cleaning a high frequency financial database*. Technical report, Tick Data, Inc.

Granger, C.W.J., Poon, S.-H., 2003. *Forecasting volatility in finnancial markets*. Journal of Economic Literature, 41, 478-539.

ap Gwilym, O., Sutcliffe, C., 1999. *High-frequency financial market data: sources, applications and market microstructure*. Risk Books, London.

Hamilton, J. D., 1994. *Times series analysis*. Princeton University Press.

Hansen, P. R., Lunde, A., 2006. *Realized variance and IID market microstructure noise*. Econometric Society 2004 North American Summer Meetings 526, Econometric Society.

Harvey, A. C., Ruiz, E., Shephard, N., 1994. *Multivariate stochastic variance models*. The Review of Economic Studies, 61, 247-264.

Hellerstein, J. M., 2008. *Quantitative data cleaning for large databases*. Report for United Nations Economic Commission for Europe. Berkley, CA: EECS Computer Science Division.

Heston, S. L., 1993. *A closed-form solution for options with stochastic volatility with applications to bond and currency options*. The Review of Financial Studies, 6, 327-343.

Hull, J. C., 2012. *Risk management and financial institutions*. Wiley finance

Hull, J. C., White, A. D., 1987. *The pricing of options on assets with stochastic volatilities*. Journal of Finance, 42, 281-300.

Jacod, J., Shiryaev, A. N., 2003. *Limit theorems for stochastic processes*. Springer-Verlag, New York.

Jacod, J., Li, Y., Mykland, P., Podolskij, M., Vetter, M., 2009. *Microstructure noise in the continuous case: the pre-averaging approach*. Stochastic Process and their Applications, 119, 2249-2276.

Jacquier, E., Polson, N. G., Rossi, P. E., 1994. *Bayesian analysis of stochastic volatility models*. Journal of Business and Economic Statistics, 12, 371-417.

Mandelbrot, B. B., 1963. *The variation of certain speculative prices*, The Journal of Business 36, 394-419.

Melino, A., Turnbull, S.M., 1990. *Pricing foreign currency options with stochastic volatility*. Journal of Econometrics, 45, 239-265.

Øksendal, B., 2000. *Stochastic Differential Equations: An Introduction with Applications*. Springer-Verlag.

Parkinson, M., 1980. *The extreme value method for estimating the variance of the rate of return*. Journal of Business, 53, 61-65.

Protter, P., 2004. *Stochastic Integration and Differential Equations*. Springer-Verlag.

Rogers, L., Satchell, S., 1991. *Estimating variance from high, low and closing prices*. Annals of Applied Probability, 1, 504-512.

Roll, R., 1984. *A simple implicit measure of the effective bid-ask spread in an efficient market*. Journal of Finance, 39, 1127-1139.

Ruiz, E., 1994. *Quasi-maximum likelihood estimation of stochastic volatility models*. Journal of Econometrics, 63, 289-306.

Taylor, S. J., 1982. *Financial returns modelled by the product of two stochastic processes - a study of daily sugar prices 1961–79*. Time series analysis: theory and practice,1, 203-226.

Taylor, S. J., 1986. *Modelling financial time series*. Wiley.

SUMMARY

Volatility estimation has a crucial role in modern finance theory, since is an essential input in many models (option pricing, risk management, returns modelling). It is a well-known fact that variance is time varying, but the volatility process cannot be identified, since what is observed is just a realization of this latent variable. In order to estimate volatility, two kinds of models have been developed in literature, that are stochastic and non-stochastic (traditional) models. The formers attempt to describe volatility as a stochastic function of its lags, while the latters provide a deterministic specification for this process. Obviously, stochastic volatility (SV) models require higher computational efforts, but they allow for more complex models building. They can be used, for example, to model more accurate option hedging strategies, or to simulate more realistic return patterns. The availability of high-frequency data (HFD) has made possible more accurate procedures for parameters' estimation. It is possible to estimate the latent daily volatility process through the observed high-frequency returns, and use this time series as a proxy of volatility into financial models. HFD still need particular handling procedure, such as outliers cleaning or sparse sampling, *i.e.* sampling prices at fixed time intervals. In this paper, lastly, two applications are discussed, using data on EUROSTOXX index, a broad index for the EURO zone. The first is a trading strategy on the VSTOOXX (V2X), that is the volatility index of the EUROSTOXX. The second is a VaR analysis on a portfolio exclusively composed of the EUROSTOXX index.

Let the price of a security at a given time be P_t and its natural logarithm expressed as p_t, then return over the previous interval of time, computed as the difference of log-prices is $r_t = dp_t$. A widely diffused belief is that returns are function of their long term mean μ and their (unobserved) variance process σ_t. In differential terms it means that:

$$r_t = dp_t = \mu\, dt + \sigma_t dz_t \tag{1}$$

where z_t is a Brownian Motion, that is a continuous stochastic process such that its increments are *iid* normally distributed with mean zero and variance *dt*. The discrete-time version of equation (1) is $r_t = \mu + \sigma_t z_t$. The first models that accounted for time varying volatility were introduced by Engle (1982) and Bollerslev (1986) with, respectively, ARCH (AutoRegressive Conditional Heteroskedasticity) and GARCH (Generalised ARCH). In ARCH models, the variance, conditional on the available information set and

on an initial value σ_0, is imposed to be a linear function of lagged squared returns, thus is heteroskedastic. In GARCH models variance is function also of its lags. A tipical GARCH formulation is the following:

$$\sigma_t^2 = \omega + \sum_{p=1}^{P} \alpha_p r_{t-p}^2 + \sum_{q=1}^{Q} \beta_q \sigma_{t-q}^2 \qquad (2)$$

where ω, α, β are constant parameters that can be estimated through maximum likelihood. In ARCH class of models, the variance is still a deterministic function of known variables.

With the availability of HFD and the improvement of computational power, SV models has assumed an important role in finance modelling. They allow to estimate more accurately the evolution of the true volatility process. In SV models, also the variance is a stochastic process, *e.g.*:

$$d\sigma_t^2 = \alpha_0 \, \sigma_t^2 \, dt + \alpha_1 \, \sigma_t^2 \, dw_t \qquad (3)$$

where α_0, α_1 are constant parameters, while w_t is a Brownian Motion that may be correlated with the z_t process described in equation (1). Non-stochastic volatility models can be estimated through maximum likelihood estimator (MLE) procedure, where the parameters of the model are those values which maximize the *likelihood* function, *i.e.* the probability of observing that specific sample. In SV models, since innovations terms are not normally distributed, the MLE may result not robust or even not consistent. Sometimes it is infeasible or even impossible to find a closed-form solution for the MLE problem, since, because of the multivariate distribution's complexity, the likelihood function may result too difficult to evaluate. Some authors have developed approximations for the likelihood function that lead to reasonable results. The Quasi-MLE procedure, assumes that innovations are *iid* Normally distributed, such that a simplified version of the likelihood function can be obtained. Another widely used method consists into generating the distribution by repeated random sampling, and then computing moments of these simulated patterns. Monte Carlo (MC) methods, for example, are algorithms where random errors from a specific probability distribution are generated, obtaining several possible outcomes for the variable being simulated. With the application of Bayesian statistics, Markov Chain MC (MCMC) methods can be used to simulate the posterior density for parameters of stochastic volatility models (Chib, Greenberg, 1995). They allow to find an *invariant*, *i.e.* constant, density of the *transition kernel*, that is the conditional distribution function representing the evolution process through time of the simulated variable. The transition kernel is, indeed, iterated a large number of times, until the distribution of the simulated observations reaches a stationary (invariant) state. The resulting invariant density is the posterior distribution from which samples are desired.

HFD can be used to enhance estimations of the volatility process. The solution of the differential equation (1), assuming, without loss of generality, that the mean return is zero, can be expressed as:

$$r_{0,T} = p_T - p_0 = \int_0^T \sigma_t dz_t \tag{4}$$

These processes are also called *Ornstein-Uhlenbeck*, or *Gauss-Markov*. They have Normal and stationary (multivariate) distribution, meaning that the multivariate distribution does not depend on time, and they are Markovian, which means that the density function of future realizations does not depend on past values. Now, let τ be the unequally-spaced time set of observations, δ the stochastic process of time intervals, and $r_{\delta_t}$the return over the *t-th* interval, then the Quadratic Variation process of *p* is defined as:

$$\langle p \rangle_t = \int_0^t (dp_s)^2 = \lim_{\sup\{\delta\}\to 0} \sum_{t\in\tau} \left(p_{t+\delta_t} - p_t\right)^2 = \lim_{\sup\{\delta\}\to 0} \sum_{t\in\tau} r_{\delta_t}^2.$$

Since:

- the Brownian Motion term z_t is a *martingale*, (which implies that the log-price process is a *semi-martingale*);
- according to stochastic calculus, if M_t is a *semi-martingale*, and X an integrable variable, then $\langle \int X\, dM \rangle = \int X^2 d\langle M \rangle$;
- the QV of a Brownian Motion is equal to the elapsed time: $\langle z \rangle_t = t$;

the QV of the price process is equal to:

$$\langle p \rangle_t = \langle \int_0^t dp_s \rangle = \langle \int_0^t \sigma_s dz_s \rangle = \int_0^t \sigma_s^2 ds$$

where the last term is called Integrated Variance (IV), and is exactly the latent variance process that stochastic volatility models attempt to estimate. The IV process is still unobservable, but the QV process can be consistently estimated through the Realized Variance (RV) estimator. RV is the sample counterpart of QV, and is the sum of squared observed returns. Since $QV_t = \lim_{\sup\{\delta\}\to 0} RV_t$, and since time interval of HFD is close to zero, RV computed with HFD is a consistent estimator for both the QV and IV process.

The RV estimator has nice properties, such as the asymptotic normality: $\sqrt{n}\frac{RV-IV}{\sqrt{2IQ}} \overset{d}{\to} N(0,1)$, with n denoting the sample size and IQ the Integrated Quarticity, namely $\int_0^t \sigma_s^4 ds$. IQ is also an unobservable variable, but it can be easily estimated with the Realized Quarticity (RQ) estimator, where $RQ_t = \frac{1}{3} n\delta^{-1} \sum_t^T r_t^4$, which yields to $\sqrt{n}\frac{RV-IV}{\sqrt{2RQ}} \overset{d}{\to} N(0,1)$.

RV may still be affected from biases. In presence of microstructure noise, RV is estimating IV plus the variance of the error terms, that are the difference between real and observed price. Aït-Sahalia *et al.* (2005b) used the Two Stage Realized Variance (TSRV) estimator, which is constructed by averaging all the possible RV estimators obtained with a *sparse sampling* procedure on lower-frequency intervals (*e.g.* 5 minutes), and then subtracting the estimator of the microstructure error: $\langle r\rangle_t^{TSRV} = \langle r^*\rangle_t^{avg} - \frac{\bar{n}}{n}\langle r^*\rangle_t$. This estimator is robust to microstructure noise, thus it is suggested for *tick-by-tick* analysis of traded securities. RV may suffer also from the presence of jumps. Jumps can be thought as those relevant variations in price due to news or announcements. It is necessary to add the jump stochastic component to the price process to avoid model mis-specification. This procedure implies that RV is estimating the sum of IV and the Jump Variation process (JV), that is the sum of squared jumps. Barndorff-Nielsen and Shephard (2003) introduced the realized Bipower Variation (BPV) estimator, expressed as the sum of products of adjacent returns, taken in absolute value: $BPV_t = \sum_{i=2}^t |r_i||r_{i-1}|$. The estimator $\frac{\pi}{2} BPV_t \overset{p}{\to} IV_t$ is then consistent for IV, since the probability of two consecutive jumps is about zero. The Range-based Variance (RgV) is another important estimator for IV. It is built with intra-period information (open, close, high and low quotes) and has the advantage of reducing the data sample, still maintaining core statistical properties. This variable is more efficient than the other estimators, in the sense that presents less variance. Parkinson (1980) initially proposed a formulation for it: $\widehat{\sigma_h^2} = \frac{1}{4\ln 2}\sum_{\Delta t\in(h-1;h]}^{h} (RgV^{\Delta t})^2$, with $RgV^{\Delta t}$ denoting the difference between high and low quotes on the interval *Δt*, over the *h-th* day. Since this estimator is sensible to te presence of outliers, careful data cleaning operation are required, or other quantile levels may be considered instead of high and low quotes.

A feature of HFD is the irregular time spacing of observations, since data is gathered as trades occur (which obviously happen at irregular intervals). Therefore, time can be represented, according to Daley and Vere-Jones (1988) as a *point process*, that is a time set where the time interval is a sequence of non-decreasing random variables. Models such as the *autoregressive conditional duration* (ACD) introduced by Engle and Russel (1998), aim to describe this feature by modelling the interval of time as a differential stochastic equation (similar to that of volatility). If the ACD process is supposed to be exogenous from the price-volatility process, it is possible to estimate duration at first, and then, conditional on these results, estimate the volatility parameters. Even if this procedure is theoretically correct, empirically, better results are obtained with the *sparse sampling* method, which implies to sample observations from a lower-frequency equally-spaced time set.

Dealing with HFD is still not trivial. They may present errors deriving from the information gathering process (such as missing quotes), or from market microstructure inefficiencies (such as temporarily lack of liquidity that implies unreasonable quotes displayed). Outliers' detection techniques are involved to clean data from possible wrong unreasonable prices. Browlees and Gallo (2006), for instance, suggest to mark outliers those observations which exceed the trimmed mean of a neighbourhood of k prices by three (trimmed) standard deviations plus a parameter γ (that depends on data frequency). *Trimmed* moments of a neighbourhood of k prices are computed using previous and following k observations: ${}_k m_t(p) = \sum_{i=-k}^{k} p_{t+i}$ and ${}_k\sigma_t(p) = \sum_{i=-k}^{k}\left(p_{t+i} - {}_k m_t(p)\right)^2$. The rule is to evaluate whether $\left|p_t - {}_k m_t(p)\right| > 3\,{}_k\sigma_t(p) + \gamma$. Another issue of HFD is the missing of data, which is typical of less liquid markets. A common solution is to fill the missing position with the latest available observation, which may be technically correct if no trades had occurred during that time window. Or, the missing value can be substituted by a weighted average of the closer (previous and following) available quotes. The choice of the filling method should be a careful operation, since it may have repercussions on the statistical structure of data. Sometimes sparse sampled observations may improve estimations. Data aggregation necessarily implies loss of information, but the gain in terms of robustness of estimators and lightening of computations is relevant. For instance, microstructure noise, which arises when the observed price fluctuates because of bid-ask spread, can be considerably reduced at 1-minute frequency.

This paper's analysis is conducted on the EUROSTOXX index, which is the main stock index of the Euro-zone. Observations starts from 2011, at irregular intra-minutes frequencies (15 seconds), for a total of 4.598.132 rows. All computations have been done with Julia language. Figure (1) shows an extract of tick-by-tick pattern of the EUROSTOXX index, while figure (2) shows a one-minute extract of one minute. As can be seen, the price stays almost to the same level, and eventually has some "jumps", that can be due to effectively price change or due to errors. Figure (3) plots the relative returns over the entire dataset. Stochastic volatility variables have been computed by creating an equally spaced time and price grid of 1-minute length. The algorithm written for this thesis, which makes use of time serialization and sparse matrix operations, allows to complete the sparse sampling operation in just 0,19 seconds (about 40'000x faster than a normal procedure), with just 84 MB of memory allocated (about 3'000x smaller than a normal procedure). The analysis is then conducted on RV at 1-minute frequency (RV1), RV at 5-minutes frequency (RV5), BPV, TSRV. Figure (4) shows the standardized return, compared with a Normal (0,1) distribution.

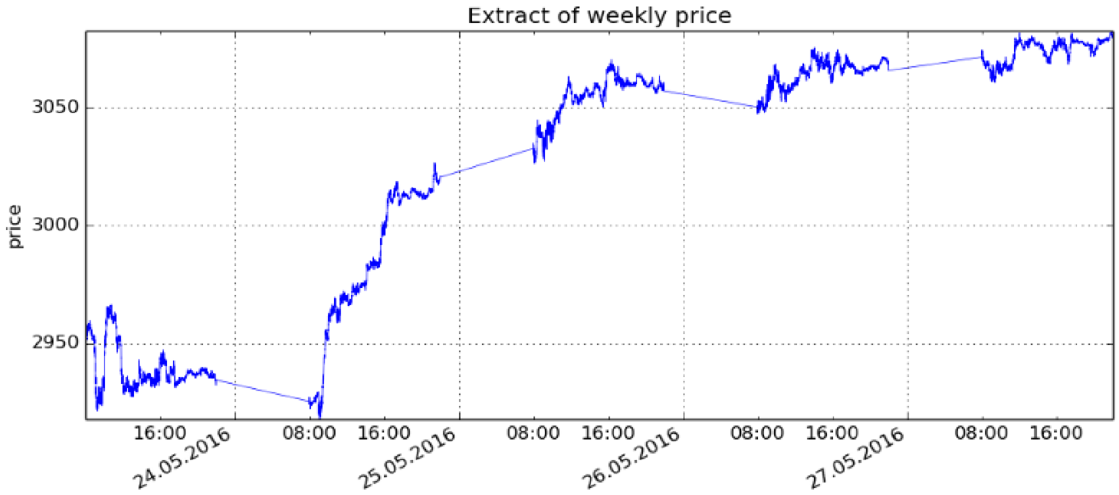


Figure 1. Weekly price pattern.

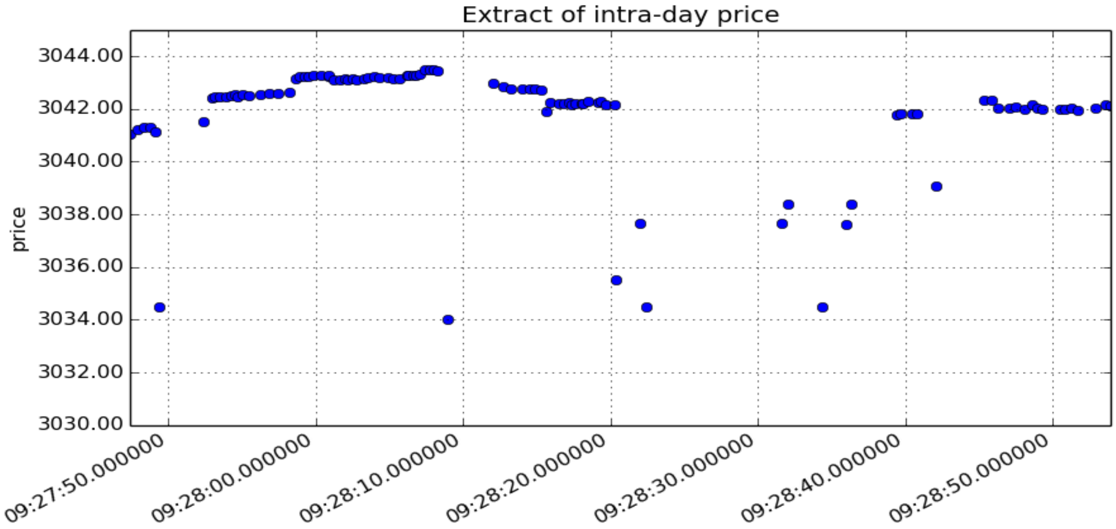


Figure 222. One-minute price pattern.

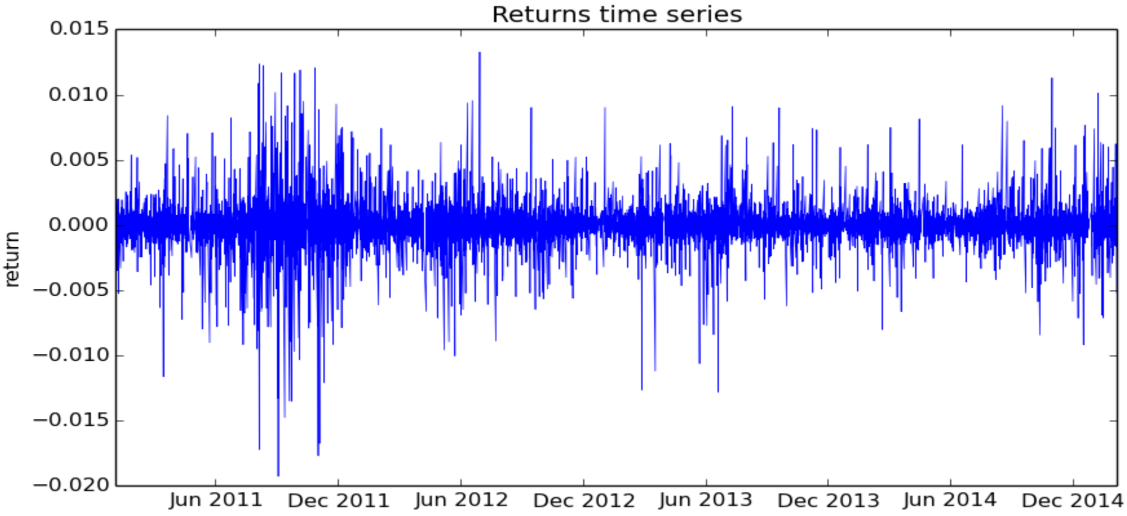


Figure 3. Returns time series.

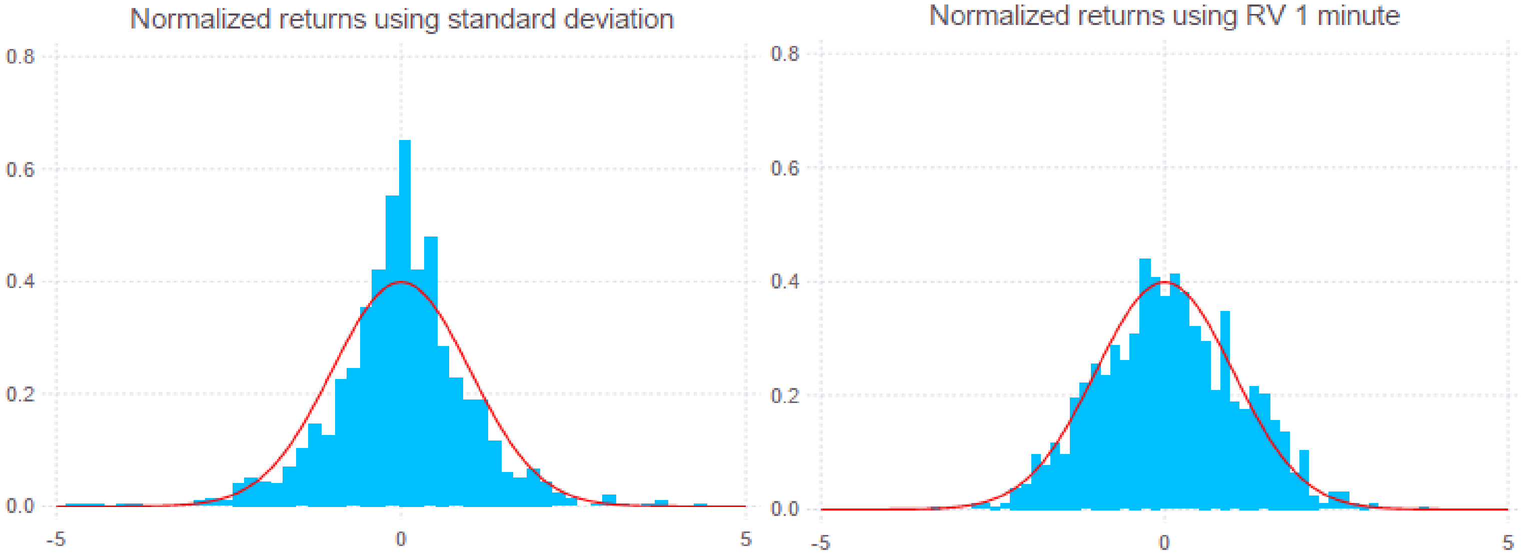


Figure 4. Standardized returns vs Normal (0,1)

The choice of one and five minute for RV is inducted mainly by literature, and because these frequencies are the most representative of the category of high-frequency sampled observations. Figure (5) summarizes the reason of this choice. RV1 has the highest explanatory power, in terms of adjusted R-squared of a regression of the variable on its lags (the number of lags are determined by BIC procedure). RV5 seems to be a good representative of lower sampling frequencies, since adjusted R-squared is similar to that of the following sampling minutes. The aim of this analysis is to assess the predicting power of stochastic volatility, this is the reasons why RV1 and RV5 have been chosen.

Sampling frequency (minutes)	tick	1	2	3	4	5	6	7	8	9	10
Adj. R-squared of $RV_t \sim [1\ RV_{t-lag}]$	0,091	0,609	0,585	0,54	0,487	0,476	0,468	0,441	0,441	0,467	0,479
Adj. R-squared of $\sqrt{RV_t} \sim [1\ \sqrt{RV_{t-lag}}\ ]$	0,192	0,642	0,624	0,587	0,554	0,546	0,535	0,521	0,526	0,534	0,543

Figure 5. RV statistics at different sampling intervals.

Figure (6) shows the time series of RV, RV1 and RV5. Figure (7) shows autocorrelation functions and the time series of some of the analysed variables (RV1, BPV). It is interesting to note how correlation on the first lags is significant, and decays very slowly over time.

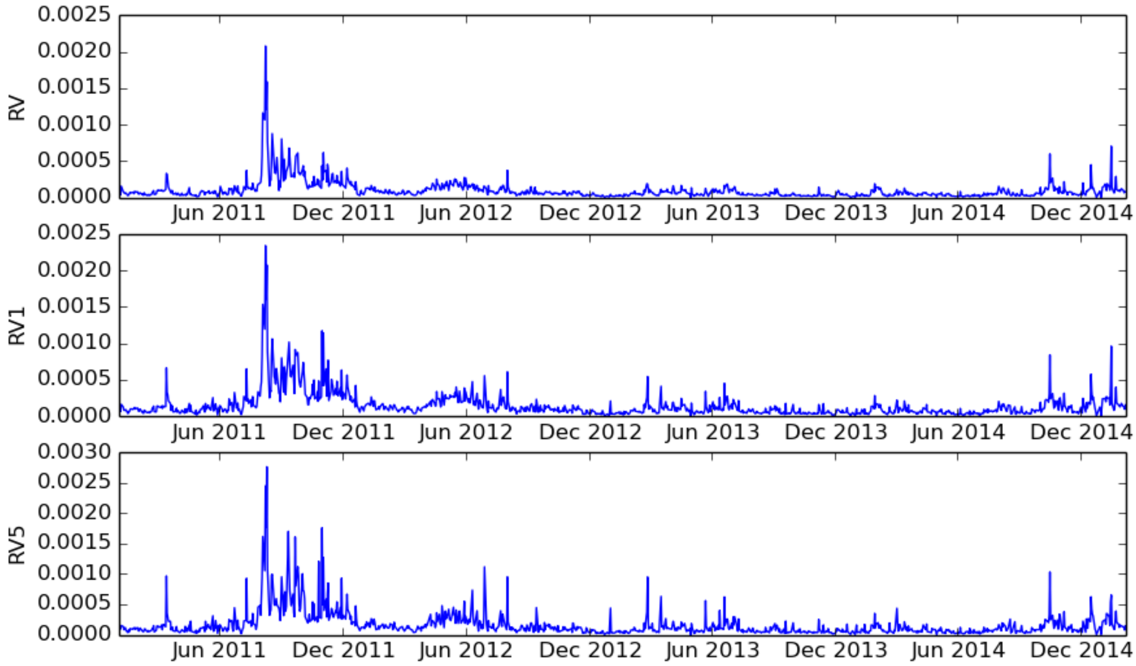


Figure 6. Time series of RV at different sampling horizons (tick-by-tick, one-minute, five-minutes).

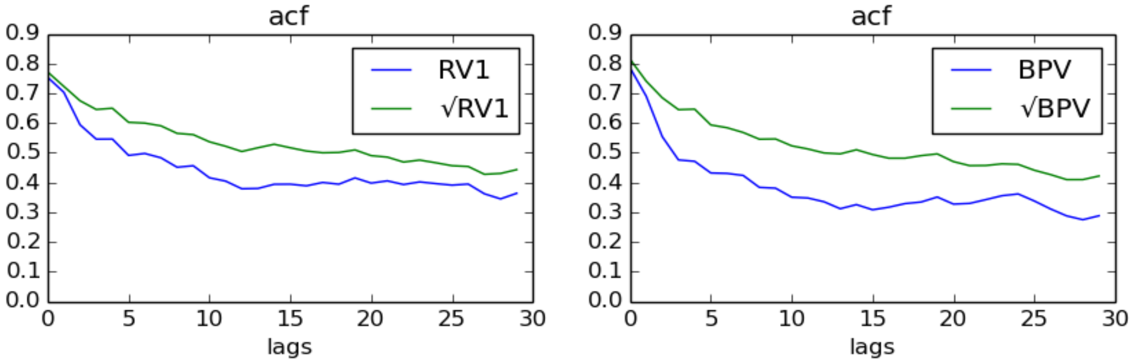


Figure 7. Autocorrelation function of RV1 and BPV, both on the normal and on the square root process.

The analysed stochastic volatility processes show a discrete predicting power. In this paper this predicting power is assessed to forecast future movements of the VSTOXX (V2X) index, to evaluate if it is possible to build a trading strategy that involves stochastic volatility. The V2X shows the level of the implied volatility of hedged at-the-money options on the EUROSTOXX. Another estimator is then introduced, that is the Heterogeneous AR (HAR), defined as the sum of RV's moving averages at one day, one week (5 observations) and one month (20 observations). This model seems to better capture the different horizons effects, by disentangling RV process into its weekly and monthly moving averages. Figure (8) shows results of regressions with V2X index.

Explanatory variable	*RV1*	*BPV*	*RV5*	*TSRV*	*HAR*
$V2X_t \sim X_{t-1}$	0,543	0,4848	0,4759	0,4554	0,8042
$V2X_t \sim \sqrt{X_{t-1}}$	0,6603	0,6595	0,5998	0,581	0,8501
$V2X_t \sim \sqrt{X_{t-1}}, \sqrt{X_{t-2}}$	0,7284	0,7081	0,694	0,6819	-

Figure 8. Adjusted R-square of a regression of V2X and the stochastic volatility variables. The first column indicates the regression run.

There is evidence that stochastic volatility can be used to predict at least the direction of next-day V2X, thus two trading strategies have been built. The first strategy consists into predicting the next-day movement of V2X, whit an AR rolling window forecast of one year, and then buying if an increase is expected, or selling otherwise. It is assumed that the price of one traded contract is equal to the level of the index, with no transaction costs. Only one contract at time is traded, and the position is rolled each day (as if 1$ is invested on each day, without reinvesting). The second strategy adds to the previous a momentum component, which accounts for mean reversion. If the level of the

V2X, over the previous two days, has risen by a certain percentage threshold, then the index is sold, regardless from the forecast. Since the choice of the thresholds leads to different results, figure (9) displays the average return setting the threshold to all values between 10% and 30%, with 1% increment. Results show that, using stochastic volatility, on average, is possible to achieve better returns respect to a benchmark strategy (which involves the use of the only V2X lagged value).

Strategy-1	*same sign ratio*	*overall return*
V2X (lag)	46,21%	-223,14%
RV1	48,86%	-31,41%
$\sqrt{RV1}$	48,86%	-45,13%
HAR	48,23%	-100,32%

Strategy-2 with momentum	*overall return*	*annualized return*
V2X (lag)	1,75%	0,56%
RV1	32,78%	9,58%
$\sqrt{RV1}$	19,26%	5,85%
HAR	14,60%	4,50%

Figure 9. Different strategies to trade the V2X index. The same sign ratio is the ratio of predicted on realized returns' signs. The first column of both tables is the variable used to forecast.

The last application is a VaR backtesting analysis with stochastic volatility. The benchmark model, the EWMA, is compared against RV1, BPV and HAR. EWMA model, described recursively as $\sigma_t^2 = \lambda\sigma_{t-1}^2 + (1-\lambda)r_{t-1}^2$, has been chosen since it is widely used in risk management, due to its efficacy and simplicity. The parameter λ is set to 0.94, following Hull (2012) notation. These variables are compared in parametric and Monte Carlo simulation methods, both at 1-day and 10-days horizons. For the parametric model, it is assumed that standardized returns distribute as a Normal (0,1). Since the standardized (with stochastic volatility) returns empirically distribute as a Normal (0,1), it is reasonable to expect that the confidence interval for the future return is $[r_{t+1} - \alpha\hat{\sigma}_{t+1},\ r_{t+1} + \alpha\hat{\sigma}_{t+1}\,]$, where α, set to 2.326, is the prudential quantile (99-th) of a Normal (0,1) distribution. The next-period volatility, in the case of stochastic variables, is computed through an AR process rolling window forecast. The 10-days EWMA volatility is the daily volatility multiplied by square root of 10 (is assumed to remain constant), while for stochastic variables the volatility over each 10s of days is computed by summing the forecasted variances obtained by direct forecast: $\sigma_{t,10d} = \sqrt{\sum_{i=1}^{10} RV1_{t+i}}$, with RV given by a forecast of $RV1_{t+i} \sim [1\ RV1_t\ RV1_{t-1}]$. Figure (10) shows results of the parametric analysis. The *exceptions* column indicates the percentage of time that the estimated VaR was not sufficient to cover the loss occurred (the more this value is near to 1%, the better the model is), while the *average capital required* column describes the average capital put as reserve, that is the daily expected VaR loss.

	1 day		10 days	
Variable	*exceptions*	*average capital required*	*exceptions*	*average capital required*
EWMA	1,89%	2,50%	1,79%	7,88%
RV1	1,89%	2,44%	1,28%	7,89%
BPV	3,40%	2,11%	2,68%	6,85%
HAR	3,03%	2,32%	2,04%	7,40%

Figure 10. Parametric VaR analysis on 1-day (left) and 10-days (right) horizons.

None of the models is able to perfectly match the required quantile, but RV performs better than the EWMA, since on 1-day it reaches the same exceptions level requiring less capital, while in the 10-days case exceptions occurrence is closer to 1% still requiring same capital than EWMA.

The simulated model for the Monte Carlo methods is: $r_{t+1} = dp_{t+1} = \varepsilon_{t+1}\hat{\sigma}_{t+1}$, with an implied drift rate of zero, and the volatility proxy forecasted through an AR model (rolling window) of the stochastic variables. Simulations are run by drawing ε from a Normal (0,1), Student-t and Laplace distributions. The choice of the latters derives from the fact that, since empirical returns have high kurtosis, drawing from a Normal distribution may underestimate the probability of extreme events to happen. Student-t and Laplace, indeed, have higher kurtosis (Laplace's shape is also closer to that of empirical returns). The choice of the Student-t degrees of freedom is derived in two ways: by maximum likelihood and by finding the degree-of-freedom parameter such that the kurtosis of the resulting Student-t distribution matches the empirical kurtosis of returns. Empirical excess kurtosis is observed to be about 2.51, which implies a degree of freedom parameter of 6.39. Parameters of Laplace distribution are chosen by maximum likelihood. Figure (11) shows exceptions rate at 1-day and 10-days. Results shows how matched Student-t and Laplace distributions for ε simulations, perform better. They are even better than parametric models.

	1 day				10 days			
ε *Variable*	*N(0,1)*	*t (MLE)*	*t (match)*	*Laplace*	*N(0,1)*	*t (MLE)*	*t (match)*	*Laplace*
RV1	1,89%	1,39%	0,63%	0,76%	1,28%	0,77%	0,64%	0,77%
BPV	*3,28%*	2,14%	1,13%	1,51%	2,55%	1,66%	1,02%	1,66%
HAR	*2,90%*	1,89%	0,76%	1,39%	2,04%	1,53%	0,51%	1,79%

Figure 11. Exceptions rate using stochastic variables in Monte Carlo simulations, with ε drawn from different distributions. Parameters of t (MLE) and Laplace distributions are found by MLE. Parameters of t (matched) are found by matching empirical kurtosis.

These results confirm the fact that stochastic variables may provide better insight for the estimation of the volatility process. The higher computation efforts they require are balanced by more accurate estimations. Stochastic volatility models seem to provide more accurate estimates of the true volatility process, principally because they try to exploit all the possible information available through HFD. This confirms the theory that big data are useful to better understand the behaviour and the true nature of certain stochastic processes, and maybe forecast their probable future realizations.

Last few words will be spent on the Julia language. This language is very similar to modern languages such as Matlab or R, but has the advantage that is much faster, comparably to C++. It has been developed relative recently, thus in some cases is still not complete (it does not have an "nice" interface, or some important functions or packages are missing). However, if the aim of the analysis is to deal with huge amount of data, it can be very helpful thanks to its fastness and versatility. If coding is not a pain, this language may give many satisfactions to a person who has to analyse many data and requires a powerful language.