CHARLES UNIVERSITY IN PRAGUE

FACULTY OF SOCIAL SCIENCES

Institute of Economic Studies

Bc. Tomáš Křehlík

Does wavelet decomposition and neural networks help to improve predictability of realized volatility?

Master's thesis

Prague 2013

Author: Bc. Tomáš Křehlík

Supervisor: PhDr. Jozef Baruník, Ph.D.

Academic year: 2012/2013

Bibliographic entry:
KŘEHLÍK, Tomáš. *Does wavelet decomposition and neural networks help to improve predictability of realized volatility?* Prague, 2013. Master's thesis, Charles University, Faculty of Social Sciences, Institute of Economic Studies. Supervisor: PhDr. Jozef Baruník, Ph.D.

Název práce: Does wavelet decomposition and neural networks help realised volatility forecasts?
Autor: Bc. Tomáš Křehlík
Institut: Institut ekonomických studií
Vedoucí bakalářské práce: PhDr. Jozef Baruník, Ph.D
E-mail vedoucího: `barunik@fsv.cuni.cz`

Abstrakt: V této práci zevrubně srovnávám standardní odhady realizované volatility včetně nového waveletového odhadu v časově frekvenční doméně (Barunik and Vacha 2012) na širokém vzorku aktiv: *oleji, zlatu* a indexu *S&P 500*. Waveletový odhad navíc dovoluje rozložit volatilitu do několika investičních horizontů, což má dle literatury přinést další informaci o časové řadě volatility. Dále navrhuji použití neuronových sítí pro předpovídání realizované volatility. V odhadech používám vrstevnatou a rekurzivní topologii. Samotnou realizovanou volatilitu předpovídám kumulativně na 1, 5, 10 a 20 dní dopředu. Předpovědi z neuronových sítí porovnávám oproti ARFIMA modelu a triviálnímu modelu. Potvrzuji pozitivní vlastnosti nového waveletového odhadu v případě *oleje* a *zlata*, ale v případě *S&P 500* se tyto vlastnosti nepotvrzují. Možné vysvětlení je, že metoda nadměrně koriguje data, protože se v těchto datech téměř nevyskytují skoky. Co se týká předpovědí, neuronové sítě překonávají ARFIMA model v objemu informace o dynamické struktuře časové řady.

Klíčová slova: realizovaná volatilita, wavelety, neurální sítě.

Title: Does wavelet decomposition and neural networks help to improve predictability of realized volatility?
Author: Bc. Tomáš Křehlík
Department: Institute of economic studies
Supervisor: PhDr. Jozef Baruník, Ph.D
Supervisor's e-mail address: barunik@fsv.cuni.cz

Abstract: I perform comprehensive comparison of the standard realised volatility estimators including a novel wavelet time-frequency estimator (Barunik and Vacha 2012) on wide variety of assets: *crude oil, gold* and *S&P 500*. The wavelet estimator allows to decompose the realised volatility into several investment horizons which is hypothesised in the literature to bring more information about the volatility time series. Moreover, I propose artificial neural networks (ANN) as a tool for forecasting of the realised volatility. Multi-layer perceptron and recursive neural networks typologies are used in the estimation. I forecast cumulative realised volatility on 1 day, 5 days, 10 days and 20 days ahead horizons. The forecasts from neural networks are benchmarked to a standard autoregressive fractionally integrated moving averages (ARFIMA) model and a mundane model. I confirm favourable features of the novel wavelet realised volatility estimator on *crude oil* and *gold*, and reject them in case of *S&P 500*. Possible explanation is an absence of jumps in this asset and hence over-adjustment of data for jumps by the estimator. In forecasting, the ANN models outperform the ARFIMA in terms of information content about dynamic structure of the time series.

Keywords: realised volatility, wavelets, neural networks.

Length of the thesis: 13070 words.

Firstly, I would like to thank Mr. Baruník for careful guidance and very inspiring collaboration. My deepest thank you goes to my family for their everlasting support and inspiration in life.

I hereby declare, that I have written this thesis using only literature and other sources listed in bibliography. Furthermore, I declare that I have not used this thesis to acquire another academic degree. I acknowledge and agree with lending and publishing of the thesis.
Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Dále prohlašuji, že jsem tuto práci nepoužil k získání jiného akademického titulu. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

In Prague, 15 May 2013 Bc. Tomáš Křehlík

Master Thesis Proposal

Institute of Economic Studies
Faculty of Social Sciences
Charles University in Prague

Author:	**Bc. Tomáš Křehlík**	Supervisor:	PhDr. Jozef Baruník, PhD.
E-mail:	`Tomas.Krehlik@gmail.com`	E-mail:	`barunik@fsv.cuni.cz`
Phone:	+420 724 09 19 26	Phone:	+420 776 25 92 73
Specialization:	Economic Theories	Defense planned:	January 2013

Proposed Topic

Does wavelet decomposition and neural networks help to improve predictability of realized volatility?

Topic Characteristics

This thesis will investigate realised volatility prediction. Volatility is usually modelled by processes with long-term memory. My goal is to test, whether this method can be substituted by neural networks and whether those can surpass results of modelling by long-memory processes. Moreover, predictive models have to be built on an estimate of volatility. In this respect, novel approach is found in wavelet method which allows for a decomposition of volatility into several investment horizons and is robust to jumps. The decomposition is done without loss of any information, hence we can get time series which represent volatility of different time horizons and, added up together, they form the original series. This uncovers potentially new information, which is not part of standard models that are built upon other methods of estimation. Hence, predicting these individual time series could further improve prediction of volatility.

Hypotheses

1. Volatility decomposition into investment horizons improves its predictions.
2. Neural networks are better at predicting volatility than other common methods.

Methodology

In the thesis, I will use a concept of realised volatility, which will be estimated by Jump Wavelet Two-Scale Realized Volatility (JWTSRV) developed by Baruník (2011). The same approach will also be used for the volatility decomposition. For estimation, I will use neural networks. Apart from that standard methods of volatility estimation and auto-regressive fractionally integrated (ARFIMA) model for prediction will be used as benchmark against which the hypotheses will be tested.

Testing will be done in two ways. First, the aforementioned methods will be applied on finance-like time series generated by Monte Carlo method. Second, the same methods will be used on FOREX, commodities and other financial data.

Outline

1. Methodology review

   (a) Realised volatility
   (b) Estimation methods – common methods and Jump Wavelet Two-Scale Realized Volatility
   (c) Prediction methods – ARFIMA and neural networks.

2. Sandbox testing
   (a) Generating finance-like time series
   (b) Estimation results
   (c) Prediction results

3. Empirical analysis on selected real data
   (a) Rationale for choice of the given data, descriptive analysis, expectations
   (b) Estimation results
   (c) Prediction results

Core Bibliography

Andersen, T.G., T. Bollerslev, F.X. Diebold, and H. Ebens. 2001. The distribution of realized stock return volatility. *Journal of Financial Economics* 61 (1): 43–76.

Andersen, T.G., T. Bollerslev, F.X. Diebold, and P. Labys. 2003. Modeling and forecasting realized volatility. *Econometrica* 71 (2): 579–625.

Baruník, Jozef. 2011. Wavelet-based realized variation and covariation theory. PhD diss., Institute of Economic Studies, Faculty of Social Sciences, Charles University in Prague.

Dacorogna, M.M. 2001. *An introduction to high-frequency finance.* Academic Pr.

Gencay, R., F. Selçuk, and B. Whitcher. 2002. *An introduction to wavelets and other filtering methods in finance and economics.* Academic Pr.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2 (5): 359 –366.

McAleer, M., and M.C. Medeiros. 2010. Forecasting realized volatility with linear and nonlinear univariate models. *Journal of Economic Surveys.*

McNelis, P.D. 2005. *Neural networks in finance: gaining predictive edge in the market.* Academic Press.

In Prague; February 13, 2012.

Author Supervisor

Contents

Chapter 1

Introduction

Volatility of a price process is a fundamental parameter to many practical problems on financial markets ranging from hedging to pricing assets and consequently also pricing of derivatives of those assets. It has occupied the field of financial econometrics for approximately last three decades with fruitful results. The current volatility literature is concentrated on *realised volatility* estimators, improving the established estimators and their application in estimation on various assets and forecasting exercises.

Historically, volatility of a price processes was a vivid field of exploration. Various parametric methods which treated the volatility as a latent variable were suggested for estimation, from the famous (Generalized) autoregressive conditional heteroskedasticity (GARCH) models as in Engle (1982) and further developed in Bollerslev (1986), Stochastic volatility methods or exponentially weighted moving averages. These models were overcome by *realised volatility* framework which is non-parametric estimator and makes the conditional volatility ex-post observable.

The realised volatility literature was started by Merton (1980) who noted that with increasing frequency of returns observation the volatility parameter is becoming theoretically observable. Already an orthodox work by Andersen et al. (2003) established a simple estimator which is consistent in absence of noise. The noise issue was then resolved in further literature (Zhang, Mykland, and Ait-Sahalia 2005; Hansen et al. 2008) which precised the estimators to be noise-consistent. Having solved the noise issues the literature then concentrated on jumps, which create a different type of noise in the data. Jump-consistent estimators were developed in Barndorff-Nielsen and Shephard (2004); Baruník (2011). The work of Baruník (2011) develops a most theoretically advanced estimator that is consistent to noise as well as to jumps and uses wavelets to decompose volatility into several investment horizons which potentially adds information to the estimate as market is diverse and different types of investors (long-horizon or short-horizon investors) are present on the markets. Hence, the investment horizon decomposition is hypothesised to increase our knowledge about the markets and improve forecasting.

Furthermore, the applied literature is concentrated on forecasting of the integrated volatility. Historically, the most standard forecasting method was GARCH but currently with the incarnation of *realised volatility* framework, more popularity goes to methods such as heterogeneous autoregressive (HAR) methods in Corsi (2009), which stipulates the current-day volatility to be a combination of several different horizons volatilities. Even though HAR model can replicate the stylized facts about the volatility time series it does not reflect in its specification the long-memory nature of the time series. This long-memory property has been documented in many studies, such as Sowell (1992); Baillie and Bollerslev (1994); Koopman, Jungbacker, and Hol (2005). For that reasons autoregressive fractionally integrated moving average (ARFIMA) method is used in the literature. Despite the fact that artificial neural networks (ANN) have been very successful in predicting various non-linear patterns in other fields, there is only one work to my knowledge that uses artificial neural networks in the domain of realised volatility by McAleer and Medeiros (2011). This work is moreover only an augmentation of HAR model which makes no comparison to pure artificial neural network methods.

In this thesis, I take the state-of-art *realised volatility* estimators that are standard in the literature, including the wavelet estimator developed by Baruník (2011), to investigate the features of the decomposition to investment horizons and whether it improves forecasting performance across different forecasting methods and wide range of assets. Moreover, I use artificial neural networks that have been a successful tool in other fields to see how they compare to other more common forecasting methods in domain of volatility prediction.

Firstly, I take various assets, *gold, crude oil* and *S&P 500* and perform realised volatility estimation with state-of-art methods. Most importantly, I perform the horizon decomposition using the estimator first suggested by Baruník (2011). The thesis shows an important and undocumented results about stable realised volatility decomposition across various assets which further confirms the findings in Baruník (2011) who performed the same exercise on exchange rates for various currencies. Further, the decomposition into investment horizons was hypothesized to bring more information content and hence better predictions of the realised volatility. However, this hypothesis is not supported by my results.

Secondly, I perform forecasting exercise and evaluate artificial neural networks against common method ARFIMA and a mundane approach which is constructed simply as a current observed value scaled by length of forecasting period. This simple estimator is included to benchmark the results and to check that when artificial neural networks approach yields better result than the ARFIMA approach it is also better than the simplistic mundane approach. I add this estimator because several similar *paradoxes* with very simplistic approach often beat very complicated methods. In forecasting, only basic neural network typologies are used to create baseline scenarios for further expansion. The artificial neural networks prove to be consistently better in forecasting the realised volatility dynamics and this thesis serves as a basis for use of neural networks in forecasting the realised volatility.

The rest of the thesis is organised as follows. First, overall methodology for the methods that are used throughout the work is presented. In the second part, the results of the estimation itself are exposed. Last part concludes the thesis. Moreover, significant code-base was developed in R statistical software and the code is an internal part of the work, yet is not presented in print because its amount would at least double the length of the thesis. Many packages created by various authors were used, all of them are credited in the specialised bibliography section. Source code for the empirical part of the thesis will be provided upon request.

Chapter 2

Methodology

From purely methodological point of view, the thesis contains several distinctive bodies of theory that need to be presented because they serve as a basis for the approach utilized in the work. To make the thesis rather self-contained, it is necessary to present all of them. First part is dedicated to the volatility measures of financial series, their derivation from stochastic models of markets, and their measurement and estimation. After estimating a time series, the logical step is to analyse it and try to forecast it. Therefore, the second part of theory is focused on topic of artificial neural networks and some general notes from machine learning domain which elucidate the novel approach to forecasting used in this thesis. More space is dedicated to realised volatility estimation because it is closer to the economic substance of the problem whereas neural networks are more of a tool.

2.1 Realised volatility

Historically, there were many approaches how to estimate volatility of the return process. In contrast to the log-returns themselves, which are easy to obtain from the price process, the biggest problem with the estimation of the volatility of the price process is that it is a latent (unobserved) variable. Therefore, one has to develop a *model* based approaches of measurement. Older approaches usually include parametric models such as Generalised autoregressive conditional heteroskedasticity (GARCH) models or stochastic volatility models. Alternatively, one can approach the estimation from the perspective of asset-pricing and then compute the *price-implied* volatilities. But as suggested, the models are parametric, which means that they are specific to the parametrization.

Most recent approach called realised volatility strives on the other hand to estimate volatility from realised data using non-parametric measures. Historically, this approach dates back to Merton (1980) when he noted that we can estimate the volatility arbitrarily precisely if the number of noiseless observations increases within fixed time interval. The literature was further developed in works Andersen et al. (2003); Barndorff-Nielsen and Shephard (2004); Zhang, Mykland, and Ait-Sahalia (2005); Andersen et al. (2000). There are several brilliant literature reviews on the topic of realised volatility with different accents; these include Andersen and Benzoni (2008), McAleer and Medeiros (2008) or Poon and Granger (2003). Also Baruník (2011) covers the material and I borrow from his treatment heavily. However, I try to be compact for sake of space, so reader interested in deeper treatment of the topic as well as more complete reading lists is delegated to the previously mentioned literature.

In the following part, I will describe basic mathematical preliminaries upon which the estimation theory is based, then I will describe issues with using high-frequency data and approaches that solve it. Finally, I will introduce estimators that deal with the issue of jumps in the time series data.

2.1.1 Continuous no-arbitrage stochastic price process

Presenting the mathematical background, I follow the canonical specification from Andersen et al. (2003) with restriction to univariate case. Let us consider a price process that is defined on a complete probability space $(\Omega, \mathcal{F}, P)$ evolving in time interval $[0, T], T \in \mathbf{N}$. Moreover, there is an information filtration defined as an increasing family of σ-fields $(\mathcal{F}_{t\in[0,T]}) \subset \mathcal{F}$, which satisfies the condition of P-completeness and right continuity.[1] Last assumption is that the prices through time t are included in the information set $\mathcal{F}_t$.

One type of price process that is extensively used in literature is the following jump-diffusion model:

$$dp_t = \mu_t dt + \sigma_t dW_t + \xi_t dq_t, \tag{2.1}$$

where W_t is a Wiener process and q_t is a constant intensity Poisson process with intensity equal to λ. If I exclude the last part with jumps in the equation, I would get typical Wiener

1. Defining with left continuity would be equivalent, but version presented above is more standard.

process

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

with a time-specific drift and a diffusion parameter. So, in principle, the equation (2.1) defines Wiener process which is from time to time accrued by jump which occurs with intensity λ.

The quadratic variation theory is interested in estimating the, so called, integrated variance over the interval $[t-h,t]$, defined as

$$IV_{t,h} = \int_{t-h}^{t} \sigma_s^2 ds,$$

because it is a central variable to many tasks in finance such as pricing or hedging.[2] However, the measure that is more standard in stochastic processes is so called quadratic variation which is defined as

$$QV_{t,h} = \lim_{||P|| \to 0} \sum_{i=0}^{n} \left(p_{z_i} - p_{z_{i-1}}\right)^2,$$

where P is the partition $[z_0, z_1, ..., z_n]$ of the interval $[t-h,t]$, hence $z_0 = t-h$ and $z_n = t$. The equation says that quadratic variation is a limit as the norm of the intervals inside the partition, e.g. $[z_0, z_1]$, $[z_1, z_2]$, ..., goes to zero. This trivially simplifies to

$$QV_{t,h} = \lim_{||P|| \to 0} \sum_{i=1}^{n} \left(r_{z_i}\right)^2.$$

To see how integrated volatility and quadratic variation relate to each other, we use theory of martingales.

Given standard assumptions of frictionless markets: there is no arbitrage in return process, the process has finite instantaneous mean; the process (2.1) belongs to special semimartingale processes as described by Back (1991). Following fundamental results of stochas-

2. It is often a matter of philosophical debate, whether one should be more interested in estimating the integrated volatility or quadratic variation because both measure amount of movement in the price and thus could be used for pricing of the underlying asset. Approach that uses integrated variance for pricing is implicitly assuming that the jumps are exogenous to the system, whereas using quadratic variation assumes endogeneity of jumps.

tic calculus, the return process can be decomposed in the following way.[3]

Proposition 2.1. *(Return decomposition)*

Any arbitrage-free logarithmic price process $(p_t)_{t\in[0,T]}$ subject to regularity condition (as above) may be uniquely represented by

$$r_t \equiv p_t - p_0 = \mu_t + M_t = \mu_t + M_t^C + M_t^J, \tag{2.2}$$

where μ_t is a predictable and finite-variation process, M_t is a local martingale that may be further decomposed to M_t^C, a continuous sample path, infinite variation local martingale component, and M_t^J, a compensated jump martingale. By definition, $\mu_0 \equiv M_0 \equiv M_0^C \equiv M_0^J \equiv 0$, which implies that $r_t \equiv p_t$.

Hence, the return process can be decomposed into three parts: the instantaneous mean that is predictable, and two local martingale innovation parts, of which the second one is due to jumps.

If I denote any semi-martingale process by r_t, the unique quadratic variation of the process over interval t can be written as

$$[r,r]_t = r_t^2 - 2\int_0^t r_{s-}dr_s,$$

where r_{s-} is limit from the left and is well-defined. Then according to Andersen et al. (2003) I define the quadratic return variation of the return process as:

Definition 2.1. *(Quadratic return variation)*

The quadratic return variation of $(r_t)_{t\in[0,T]}$ over $[t-h,t]$, for $0 \leq h \leq t \leq T$, is

$$\begin{aligned} QV_{t,h} = [r,r]_t - [r,r]_{t-h} &= [M^C,M^C]_t - [M^C,M^C]_{t-h} + \sum_{t-h<s\leq t} \delta M_s^2 \\ &= [M^C,M^C]_t - [M^C,M^C]_{t-h} + \sum_{t-h<s\leq t} \delta r_s^2. \end{aligned} \tag{2.3}$$

3. If not stated otherwise, the propositions are stated as in Andersen et al. (2003).

Using martingale representation theorem most logarithmic price processes can be fit into this framework.

As noted above quadratic variation of such process is a special case of the aforementioned decomposition. The quadratic variation of the return process over $[t-h, t], 0 \leq h \leq t \leq T$ is

$$QV_{t,h} = \int_{t-h}^{t} \sigma_s^2 ds + \sum_{t-h \leq s \leq t} J_s^2 = IV_{t,h} + \sum_{t-h \leq s \leq t} J_s^2,$$

where J_s denotes magnitude of jump at time s. Hence, I arrived at the important relation between quadratic variation and integrated variation which is further used in the theory of realised variance.

2.1.2 Realised variance

Realised variance is an approach to measuring return variation based only on the realisation of the return process. The most appealing feature of this approach is that the measure is non-parametric as compared to older approaches to volatility measurement. As noted it dates back to 1980, but only the availability of high-frequency data and computers made this approach feasible.

Now, let us define the original realised variance estimator that is due to Andersen et al. (2003):

Definition 2.2. *(Realised variance)*

The realised variance over $[t-h, t]$, for $0 \leq h \leq t \leq T$ is defined by

$$\widehat{RV}_{t,h} = \sum_{i=1}^{n} r^2_{t-h+\left(\frac{i}{n}\right)h}, \tag{2.4}$$

where n is the number of observations in $[t-h, t]$. Realised volatility is defined as a square root of this measure.

The theory of semi-martingales provides several important results which are accentuated in the literature (Andersen et al. 2003, 2001; Barndorff-Nielsen and Shephard 2001). I state them only in condensed form. First, the realised variance estimator is unbiased and if noise and jumps are not present, the realised variance estimator is consistent non-parametric measure of the integrated variance with $n \to \infty$. Second, the discrete-time returns $r_{t,h}$, given some general restriction on the price process, are distributed as a normal mixture.

Let us now, for this part, drop the jumps in the process and concentrate only on a price diffusion process in the form of

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

When I consider only this simplified process, the above estimator defined in equation (2.4) would measure the realised variance precisely, if there was no micro-structure noise in the returns. There are various sources of micro-structure noise such as price being discrete (quoted usually only to two or three decimal places) or specific rules of trading on the market. Voluminous literature exists on this topic including useful reference book by O'Hara (1995).

To see how micro-structure influences the estimation with different frequency of sampling of the returns during the same interval, imagine a simple price observation model

$$\tilde{p}_t = p_t + \epsilon_t,$$

where $\tilde{p}_t$ is the observed price at times t, p_t is the latent true price and ϵ_t is noise at the same times. Hence, the observed return[4] is

$$\tilde{r}_t = \tilde{p}_{t+1} - \tilde{p}_t = p_{t+1} + \epsilon_{t+1} - p_t - \epsilon_t = r_t + \eta_t.$$

Applying this to the proposed estimator, I get

$$\widehat{RV}_t = \sum_{i=1}^{n_t} \tilde{r}_i = \sum_{i=1}^{n_t} r_i^2 + 2\sum_{i=1}^{n_t} r_i \eta_i + \sum_{i=1}^{n_t} \eta_i^2. \tag{2.5}$$

It is evident (assuming zero mean value for the noise and non-zero variance) that in the presence of noise the naive estimate of the realised volatility is biased. Computing the

4. This definition of return is only used for illustration purposes. Log-return definition or simple return definition yield in spirit similar results with considerably more tiring derivation.

conditional estimate

$$E(\widehat{RV_t}|r_t) = r_t^2 + n_t \operatorname{var}(\eta)$$

Hence, it is burdened with bias which increases as the number of observations in a given day increases.

In this situation, Zhang, Mykland, and Ait-Sahalia (2005) orders five approaches to reduce or completely eliminate bias (from least efficient to most efficient):

1. completely ignoring the noise – the estimator defined above in equation (2.4),
2. sampling sparsely at low frequencies,
3. sampling sparsely at an optimally determined frequency,
4. sub-sampling and averaging,
5. sub-sampling and averaging, and bias-correction.

They overly elaborate on all five possibilities and also present algebraic results for amount of noise present in the various estimators. Here, I describe two of the above approaches that will be used in the empirical part.

First, I will use the estimator that corresponds to sampling sparsely at low frequencies, which means that I use the estimator defined by equation (2.4), however, if I had, for example, one minute returns data, I would transform them into 5 minute returns data and only then perform the estimation. I will denote such estimator as $\widehat{RV}_{t,h}^{(k)}$, where (k) denotes the frequency of the sparse returns. Second, in the end of paper Zhang, Mykland, and Ait-Sahalia (2005), the authors define the estimator which corresponds to the most efficient method – sub-sampling, averaging and bias-correction. It is called Two-scale realised volatility (TSRV) and it is defined as follows:

Definition 2.3. *(Two-scale realised volatility estimator)*

The two-scale realised volatility over $[t-h, t]$ *for* $0 \leq h \leq t \leq T$ *is defined by*

$$\widehat{RV}_{t,h}^{(TSRV)} = \frac{1}{K}\sum_{k=1}^{K} \widehat{RV}_{t,h}^{(k)} - \frac{\bar{n}}{n}\widehat{RV}_{t,h}^{(all)},$$

where $\widehat{RV}^{(k)}_{t,h}$ is the previous estimate on smaller frequency and sample.

Another prominent approach to noise-consistent estimators is due to Hansen et al. (2008). The authors take kernel approach to estimating the volatility. Their estimator takes the naive estimate of the realised volatility and adjusts it by kernel-weighted auto-covariances. To put it more formally:

Definition 2.4. *(Realised kernel estimator)*

The realised kernel volatility estimator over $[t-h,t]$ for $0 \leq h \leq t \leq T$ is defined by

$$\widehat{RV}^{(RK)}_{t,h} = \gamma_{t,h,0} + \sum_{\eta=-H}^{H} k\left(\frac{|\eta|-1}{|H|}\right)\gamma_{t,h,\eta}, \tag{2.6}$$

where

$$\gamma_{t,h,\eta} = \sum_{i=1}^{n} r_{t-h+\left(\frac{i-\eta}{h}\right)h} r_{t-h+\left(\frac{i}{h}\right)h} \tag{2.7}$$

is the realised η-th auto-covariance and $k(\cdot)$ denotes the kernel function.

In the paper Hansen et al. (2008) authors offer several kernel functions with different properties. I will use Parzen kernel which is guaranteed to produce non-negative estimates and comparability with Baruník (2011). The Parzen kernel is defined as follows

$$k(x) = \begin{cases} 0 & \text{if } x > 1 \\ 2(1-x)^3 & \text{if } 1 \geq x \geq \frac{1}{2} \\ 1 - 6x^2 + 6x^3 & \text{if } \frac{1}{2} \geq x \geq 0. \end{cases} \tag{2.8}$$

These two approaches are the most used ones and they efficiently solve the problem of micro-structure noise in the price data. Let us now move to the next problem of jumps in the data.

Before proceeding to deeper treatment, it is important to note that on the theoretical level, an issue of jumps in the stochastic processes is far from resolved. One important problem is distinguishing whether there is finite or infinite activity in the jump part of the process. In the empirical part I work with the traditional, yet simplifying, assumption that the activity is finite. Recently there have been developed statistical tests that distinguish between finite and infinite activity, see Aït-Sahalia and Jacod (2011).

In my thesis, I will use two estimators that are able to deal with jumps. First, I use wavelet approach developed in Baruník (2011). Second, I use estimator developed by Barndorff-Nielsen and Shephard (2004) which is called Bi-power variation. Let us now concentrate on the first approach.

In order to explain the estimator, I have to state some fundamental results from the theory of wavelets to be able to carry on. I achieve this in very simplistic manner with taking the risk of being oversimplifying, but the complete treatment of the theory is beyond the scope of this thesis. The whole treatment of the wavelet theory can be found for example in Gençay, Selçuk, and Whitcher (2001).

2.1.3 Wavelet decomposition and Jump-adjusted wavelet two-scale realised volatility

Wavelets are a relatively modern method of signal-processing similar to Fourier transformations. However, wavelets have the advantage that they decompose the original time series to a series of coefficients that represent strength of the impulse functions of given lengths – scale transformations of so called mother wavelet – whereas the Fourier transform would state that the time series is a linear combination of sinuses and cosines and only would give the coefficients by which magnitude the given sine or cosine is *permanently* present in the whole time series. Hence, wavelets are more suitable for modelling the economic time series because they can account better for structural breaks in the hypothetical underlying model and overall nature of the continuously evolving volatility time series.

Moreover, I use maximal overlap discrete wavelet transform, because due to technical reasons standard discrete wavelet transforms cannot cope with series of different lengths than 2^n. In a very broad sense, maximum overlap transform does something very similar to bootstrap methods in econometrics. It takes the existing data in the time series and uses them in the best way to construct a time series of length 2^n and then to perform *normal* wavelet transformation.

The advantage of wavelet processing is that tests were developed to clean the data from jumps, as I will show later. Moreover, wavelet approach can decompose the realised volatility into several *investment* horizons of lengths 2^i, due to nature of the wavelet transformations. This decomposition can potentially bring about more information and due to that, should be easier to forecast. Let us first look at the wavelet decomposition.

The decomposition is summarised by the following theorem:[5]

PROPOSITION 2.2. *(Energy decomposition in discrete time)*

The energy of the time series $X_i, i = 1, ..., N-1$ can be decomposed on a scale-by-scale basis $J \leq \log_2 N$ so that

$$||X||^2 = \sum_{j=1}^{J} ||\tilde{W}_j||^2 + ||\tilde{V}_J||^2,$$

where $||X||^2 = \sum_{i=0}^{N-1} X_i^2$, $||\tilde{W}_j||^2 = \sum_{i=0}^{N-1} W_{j,i}^2$, $||\tilde{V}_J||^2 = \sum_{i=0}^{N-1} W_{J,i}^2$ and $\tilde{W}_j$ and $\tilde{V}_j$ are N dimensional vectors of the j-th level maximal overlap discrete wavelet transform and scaling coefficients.

Hence, if I define vector

$$\tilde{\mathcal{W}} = \begin{bmatrix} \tilde{W}_1 \\ \tilde{W}_2 \\ \vdots \\ \tilde{W}_J \\ \tilde{V}_J \end{bmatrix}$$

I can restate the equation in the following way

$$||X||^2 = \sum_{j=1}^{J} ||\tilde{W}_j||^2 + ||\tilde{V}_J||^2 = \sum_{j=1}^{J+1} ||\tilde{\mathcal{W}}_j||^2. \tag{2.9}$$

Importantly, note that the norm $||\cdot||^2$ is the same as the common measure of the realised volatility, so if I set $X = r_t$, I can construct estimator, which would decompose the whole volatility into J levels and be equal to the original estimate. That means, I have the following

5. The name *energy decomposition* is due to the fact that after applying the transformation I get magnitude coefficients for the wavelet scales which are in this context called energies.

equality which defines the Wavelet realised volatility (WRV):

$$\widehat{RV}_{t,h} = \sum_{i=1}^{n} r^2_{t-h+\left(\frac{i}{n}\right)h} = ||r|| = \sum_{i=1}^{n}\sum_{j=1}^{J+1} \tilde{\mathcal{W}}^2_{j,t-h+\frac{i}{n}n} = \widehat{RV}^{(WRV)}_{t,h}.$$

By this definition I have arrived to decomposed volatility into investment horizons. The horizons are given by the length of intervals that given levels of wavelet transforms take into account. For example, the first level takes into account 2^1 observations, the second takes 2^2, etc. If I had five minute data, then the first horizon is 5-10 minutes and the second is 10-20 minutes, etc.

The other advantage I lever is data cleaning. Baruník (2011) takes inspiration from Fan and Wang (2007) and introduces the following criterion for detection of the jumps in the data and following adjustment:

Proposition 2.3. *(Jump estimation using wavelets)*

Let $\tilde{\mathcal{W}}_{1,k}$ be the 1^{st} level wavelet coefficients of y_t over $[t-h,t]$ from 6 levels. If for some $\tilde{\mathcal{W}}_{1,k}$

$$|\tilde{\mathcal{W}}_{1,k}| > \frac{median\{|\tilde{\mathcal{W}}_{1,k}|, k = 1, ..., n\}}{0.6754}\sqrt{2\log n},$$

then $\widehat{\tau}_l = \{k\}$ is the estimated jump location with size $\bar{y}_{\widehat{\tau}_{l+}} - \bar{y}_{\widehat{\tau}_{l-}}$ (averages over $[\widehat{\tau}_l, \widehat{\tau}_l + \delta]$ and $[\widehat{\tau}_l, \widehat{\tau}_l - \delta]$, respectively, with $\delta > 0$ being the small neighbourhood of the estimated jump location $\widehat{\tau}_l \pm \delta$; 0.6745 is a robust estimate of the standard deviation).

The jump variation is then estimated by the sum of the squares of all the estimated jump sizes:

$$\widehat{WJV} = \sum_{l=1}^{N_t} \left(\bar{y}_{\widehat{\tau}_{l+}} - \bar{y}_{\widehat{\tau}_{l-}}\right)^2. \tag{2.10}$$

Plainly, Fan and Wang (2007) state that if I look at the first investment horizon, which takes into account two subsequent observations, and if the energy connected with them (there is unusually high movement in magnitude in two observations) is higher than median of the energies in the series adjusted by a constant, there is a jump.

This observation and connected proposition solve the detection of jumps. The next part of the proposition says how to adjust the jump, which is even more important. The magnitude of the jump is, of course, the difference between values just before the jump and just after the jump. The adjusted price series is corrected by this amount at the time of jump. In the paper, he also provides an important result that such an estimation of jumps magnitude is consistent.

Baruník (2011) combines both of the properties and develops so called Jump adjusted wavelet two-scale realised volatility estimator (JWTSRV henceforth) as follows:

Definition 2.5. *(Jump adjusted wavelet TSRV estimator (as in Barunik and Vacha (2012)))*

Let $RV^{(estimator,J)}$ denote an estimator of realised variance over $[t{-}h, t]$, for $0 \leq h \leq t \leq T$, on the jump-adjusted observed data, $y(J) = y_{t,h} - \sum_{l=1}^{N_t} J_l$. The jump-adjusted wavelet two-scale realised variance estimator is defined as:

$$\widehat{RV}_{t,h}^{(JWTSRV)} = \sum_{j=1}^{J+1} \widehat{RV}_{j,t,h}^{(JWTSRV,J)} = \sum_{j=1}^{J+1} \left(\widehat{RV}_{j,t,h}^{(W,J)} - \frac{\bar{n}}{n} \widehat{RV}_{j,t,h}^{(WRV,J)} \right) \tag{2.11}$$

where $\widehat{RV}^{(W,J)} = \frac{1}{G}\sum_{g=1}^{G}\sum_{i=1}^{n} \tilde{\mathcal{W}}^2_{j,t-h+\frac{i}{n}n}$ obtained from wavelet coefficient estimates on a grid of size $\bar{n} = n/G$ and $\widehat{RV}^{(WRV,J)} = \sum_{i=1}^{n} \tilde{\mathcal{W}}^2_{j,t-h+\frac{i}{n}n}$ on the jump-adjusted observed data, $y(J) = y_{t,h} - \sum_{l=1}^{N_t} J_l$.

Moreover, this estimator is proven to be unbiased and consistent so it possesses all the desirable properties. (For statements and proofs, see Baruník (2011).)

There is another popular approach to jump-consistent estimation which was suggested by Barndorff-Nielsen and Shephard (2004). It levers similar observations as the wavelet jump-detection. Andersen, Bollerslev, and Huang (2011) have further developed the estimator and I use their modified definition:

Definition 2.6. *(Bi-power variation estimator)*

The bi-power variation over $[t{-}h, t]$, for $0 \leq h \leq t \leq T$, is defined by

$$\widehat{RV}_{t,h}^{(BPV)} = \mu_1^{-2} \frac{n}{n-2} \sum_{i=3}^{n} \left| r_{t-h+\frac{i}{n}h} \right| \left| r_{t-h+\frac{i-2}{n}h} \right|, \tag{2.12}$$

where $\mu = \pi/2 = E(|Z|^a)$, and $Z \sim N(0,1)$, $a \geq 0$ and $\widehat{RV}_{t,h}^{(BV)} \to \int_{t-h}^{t} \sigma_s^2 ds$.

Moreover, jump-detection in this framework is done by a statistic which is introduced in the following definition:

Definition 2.7. *(Jump detection test, bi-power variation)*

Under the null hypothesis of no within-day jumps,

$$Z_{t,h} = \frac{\frac{\widehat{RV}_{t,h}^{(k)} - \widehat{RV}_{t,h}^{(BPV)}}{\widehat{RV}_{t,h}^{(k)}}}{\sqrt{\left(\left(\frac{\pi}{2}\right)^2 + \pi - 5\right)\frac{1}{n}\max\left(1, \frac{\widehat{TQ}_{t,h}}{\left(\widehat{RV}_{t,h}^{(BPV)}\right)^2}\right)}},$$

where $\widehat{TQ}_{t,h} = n\mu_{4/3}^{-3}\frac{n}{n-3}\sum_{j=5}^{n}|r_{t-h+\frac{i-4}{n}h}|^{4/3}|r_{t-h+\frac{i-3}{n}h}|^{4/3}|r_{t-h+\frac{i-2}{n}h}|^{4/3}$ *is asymptotically standard normal distributed.*

Using this jump-detection, I can then define consistent jump variation by

$$J_{t,h} = I_{Z_{t,h}>\Phi_\alpha}\left(\widehat{RV}_{t,h}^{(k)} - \widehat{RV}_{t,h}^{(BPV)}\right)$$

and the measure of integrated variance as

$$IV_{t,h} = I_{Z_{t,h}>\Phi_\alpha}\widehat{RV}_{t,h}^{(BPV)} + I_{Z_{t,h}\leq\Phi_\alpha}\widehat{RV}_{t,h}^{(k)},$$

where $I_{Z_{t,h}>\Phi_\alpha}$ denotes the indicator function which is equal to one when jump is detected and otherwise zero, the α denotes the significance level of the test.

This estimator is also consistent in estimating the integrated variance, so I can use the approach from the previous estimator and estimate consistently the jump-variation.

I conclude this part with Table 2.1 which summarizes features of various estimators that I have introduced in this section. This overview of the realised variance theory summarises relevant approaches to estimation in the literature and I can move on to next body of literature that I need to present – machine learning and neural networks as the approach will be used for modelling and forecasting of the realised volatility.

Estimator name	Noise consistent	Jump consistent	Decomposed
Naive	no	no	no
Two-scale	yes	no	no
Realised kernel	yes	no	no
Bi-power variation	partially	yes	no
Jump-adjusted wavelet two-scale	yes	yes	yes

Table 2.1: Comparison of realised volatility estimators

2.2 Machine learning

Artificial intelligence is an idea as old as civilisation. Old cultures have dreamt about creating thinking creatures and machines, but only with mathematics and computer sciences was this dream conceptualised in the works of A. Turing who showed that any logical operation can be represented by operations over binary numbers and hence laid foundations for evolution of computers and also learning algorithms often called machine learning. I will use this term often in the forthcoming parts, so it is worth explaining what is meant by it and so that reader does not have to dive into the enormous literature encompassing this topic.

Machine learning, as the name suggests, is a discipline that is closely related to artificial intelligence and its goal is to learn structures from data. Everyone of us is daily using many products that are based on the machine learning algorithms, from spam filters in your email in-box to fuzzy algorithms in your washing machine. Standard machine learning problems can be divided into two types. First, regression (measurement) problem, is a type of problem well-known to all econometricians. In this type of problem I may be trying to exactly predict a value of one (or more) variable. Typically, I might have weight, gender and age of k people and I am trying to predict their height. Second, classification problem, is a bit less frequent in econometrics but nonetheless taught in every undergraduate course. In classification set-up I am trying to infer, based on data, to which category the observation belongs. (In econometrics, methods such as probit, tobit are suited for problems falling in this category.) To use the previous example, classification problem would be to infer the gender of the respondent.

As happens usually in statistics, one has to face trade-offs between the precision of the output and the ease of training, convergence or level of generalisation. Often tasks can be simplified from regression problems to classification problems which usually become *easier* to estimate. In our illustrative case, I could ask to estimate, whether a person will have higher or lower height than some given variable. As reader can notice, I am indeed simplifying our problem and the information that I will gain from the output will also be significantly lower, however, I will usually have better precision. The information reduction means that if I set a threshold 160cm, we cannot discriminate after the estimation, whether the person could also be 200cm. So when designing machine learning algorithm I need to know precisely what I want from the estimation, because problem simplification (demanding less information from the estimated output) would usually make our algorithm much more efficient. For example, if I would be designing trading algorithm which predicts return over the next ten days I could pose the problem to predict the exact return, however, in the end I am mostly interested in knowing whether the return will be above some break-even point.

The machine learning algorithms are also distinguished by approach in which they learn. There are two main types of learning supervised and unsupervised learning which I discuss in here.[6] Supervised learning is the set-up that is more straightforward. We have the explanatory variables as well as the explained variables. Algorithms that fall into this class of learning are, for example, artificial neural networks, support vector machines, or random forest algorithms. In contrast, unsupervised learning is a problem where I do not know the explained variables (usually to which group the observations fall), but I know how many of such groups exist. Typical example of the problem is a spam filter training when the companies do not know which mails are spam and which are not. Based on the input data, the unsupervised learning algorithms can discern clusters in the data.[7] Another type of problem for unsupervised learning are noise filters, which extract voice from background.

6. Usually, about 5 types of learning are distinguished in the literature, interested reader can find more in specialised literature.

7. In practice, the spam filter problems use so called semi-supervised learning which is a combination of supervised learning – user labels mail as spam – and unsupervised learning – the rest of mails. The algorithms are only a bit more complex than in case of the exposed paradigms.

Having introduced notion of machine learning let us move to neural networks which are part of this broad category.

2.3 Neural networks

In this section, I provide brief introduction into neural networks, what they are, why to use them and how to use them. My text draws mainly from book *Neural Networks in Finance: Gaining Predictive Edge in the Market* by McNelis (2005). The book is an excellent introduction to neural networks for people with econometric and statistical background. For more interested readers rather exhausting monograph about neural networks was compounded by Haykin (2007). Now, I start with some examples of neural networks and only then come back to more general statements about neural networks.

Concisely said, neural networks are semi-parametric non-linear forecasting models. They have been used successfully in various supervised machine-learning tasks and have been more or less popular during last fifty years. The very roots date back to 1940s with McCulloch and Pitts. As the name suggests the algorithm is inspired by human brain. Neural network are built from so-called perceptrons – units that take inputs, transform it by a given activation function and provide output to other perceptrons. Every neural network consists of several perceptrons and corresponding connections, this ensemble is called a topology of a network. There are several topologies that are quite traditional and illustrative and I shall discuss them.

2.3.1 Feed-forward neural network

Feed-forward neural network is the simplest example of neural network. Graphical illustration of topology can be found in Figure 2.1. In the Figure 2.1 the simplest version is depicted. In neural-network parlance the perceptrons are gathered in layers. In the Figure, one can see three layers. First, so called input layer, denoted with I in the Figure, takes in the raw data, applies the activation function on them and sends the output to the hidden layer, as suggested by the arrows. Hidden layer (denoted H) aggregates the outputs from the input neurons and again map activation function on them and output the result to the output layer

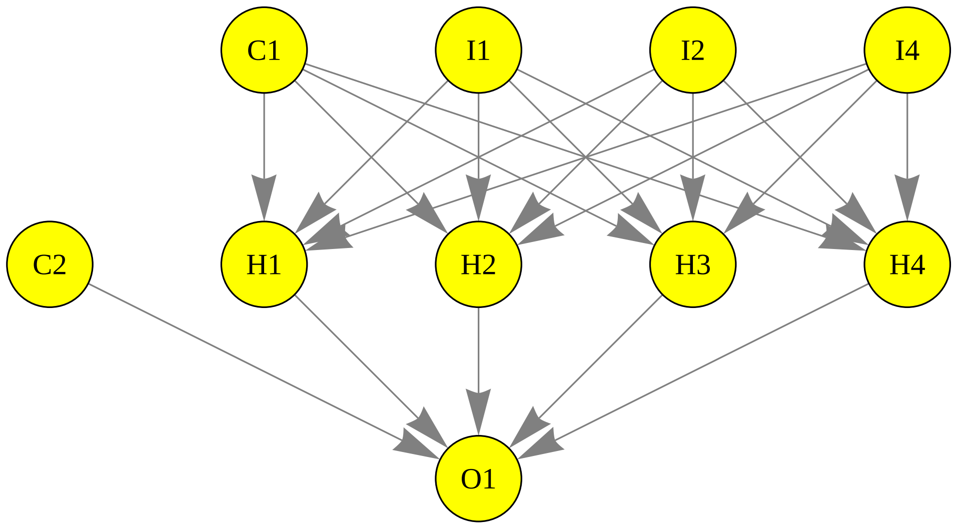

Figure 2.1: Topology of feedforward neural network

(denoted O). The output layer then only aggregates these numbers or can alternatively apply some output function on the number. Moreover, there are special neurons that provide a constant to the estimation to correct for bias. These neurons are denoted C in the Figure.

To be mathematically more precise, let us denote

- i_j – the input j,
- n, n_k – the number of layers in the network, number of neurons in the layer excluding constant neuron,
- a_k^l – the activation of neuron l in layer k, where $k \in \{1, ..., n\}$ and $l \in \{1, ..., n_k\}$,
- y_p – the output,
- $\omega_{k,m}^l$ – the weight going from neuron in layer k, layer m to layer $m + 1$ and neuron l,
- $\alpha(\cdot)$ – the activation function,

- γ_j – the output coefficients.

Usually, the activation function is assumed to be sigmoid function and I will use this convention in the forthcoming explanations. The feed-forward procedure (transforming from inputs to the outputs) of the multilayer perceptron network can then be mathematically described as follows:

$$\alpha(z) = \frac{1}{1+\exp\{-z\}} \tag{2.13}$$

$$a_1^j = \alpha\left(i_j\right) \tag{2.14}$$

$$a_k^j = \alpha\left(c_{k-1} + \sum_{z=1}^{n_k} a_{k-1}^z \omega_{k-1,z}^j\right) \tag{2.15}$$

$$y_p = \gamma_0 + \sum_{j=1}^{n_n} \gamma_j a_n^j. \tag{2.16}$$

Hence, the problem of fitting feed-forward neural network means to find vector of weights ω and γ that minimize error over a given sample. The task of minimizing this error is not trivial and there is significant amount of literature that is concerned with this problem. The first successful algorithm is called back-propagation and was introduced in Werbos (1974). Technical details are not of interest, because the precise algorithms that I will use in the estimation are different from back-propagation which was surpassed by other methods. However, the idea that stands behind the back-propagation is rather elegant and simple and it is worth describing because it illustrates nicely the problem of functional nesting that the learning algorithm is facing.

When performing the task of minimization one tries to numerically evaluate how does some cost function (in our case I could take mean square error of the estimated values to the true values) change with the weights. Hence, I need to compute the derivatives of cost functions with respect to the weights. After that I can hand the problem over to standard optimization algorithms such as (stochastic or batch) gradient descent or BFGS. It is not hard to evaluate on paper, that the derivatives by weights of layers in, let us say, 2^{nd} layer will be influenced by derivative by weights from subsequent layers. The big insight how to compute all the derivatives is that I can back-propagate the error recursively to the network to estimate the derivatives. To state a bit more formally how it works for the weights in the last layer, let $J(y, o) = \frac{1}{2}(o - y)^2$ be the cost function, where y is the estimated value and o is the actual output. Let us denote $s = \sum_{i=1}^{n} a_i \omega_i$ where a_i is input to the neuron i and ω_i is the corresponding weight. Moreover, let $y = \alpha(s)$. I need to compute partial derivatives of J with respect to ω_i

$$\frac{\partial J}{\partial \omega_i} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial s}\frac{\partial s}{\partial \omega_i} \tag{2.17}$$

$$\frac{\partial J}{\partial \omega_i} = \frac{2}{2}(y - o) = y - o \tag{2.18}$$

$$\frac{\partial y}{\partial s} = \alpha'(s) = (\text{assume sigmoid function}) = y(1 - y) \tag{2.19}$$

$$\frac{\partial s}{\partial \omega_i} = a_i \tag{2.20}$$

$$\frac{\partial J}{\partial \omega_i} = (y - o)y(1 - y)a_i \tag{2.21}$$

From this I can see that the a_i term in the final computation is dependent on the subsequent layer. This is itself function of other weights of which I need derivatives. So if I would go deeper into the neural network, the derivative will have to take into account value of the following derivatives in the network.

It is also important to compare this approach to econometric methods. The obvious place to start with is ordinary least squares (OLS). Inquisitive reader already noted that ordinary least-squares method is a subset of neural networks in case where activation function is linear. The OLS is actually a very degenerate case of a neural network without a hidden layer. If I would include hidden layers the network would become more complex than traditional OLS. In case I am using time series as an input to the neural network with inclusion of lags, the model becomes a variant of $AR(n)$ autoregressive process. However, both of these methods, become inherently non-linear when I use the activation functions. We now look at another rather particular topology of the neural networks which can be compared with $ARMA(p, q)$ autoregressive moving average processes.

2.3.2 Recurrent neural networks

Apart from multi-layer perceptron (MLP) networks there is also significant amount of neural network topologies that are called recurrent. The recurrent neural networks are substantially different from the MLP networks, because they contain loops, or otherwise stated, memory. In Figure 2.2 I show one such topology, this exact is called Jordan network.

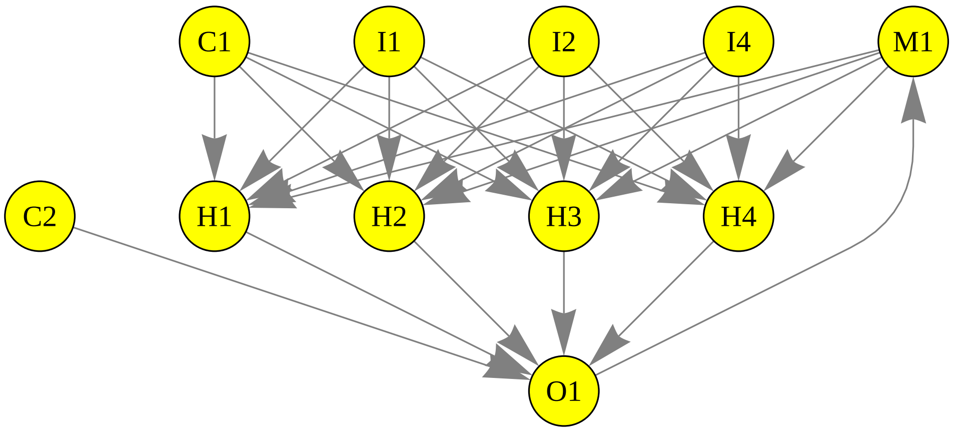

Figure 2.2: Jordan neural network

This particular network is only different from the common MLP by presence of the *context* unit (denoted M), which takes the estimate of the value and propagates it back to the hidden layer. In this particular example, the network has only one context unit, which captures only the last value, but it is straightforward to adjust the topology for cases where it would accommodate more states. There is another prominent example of recurrent neural network, which is called Elman network. Elman network stores in context unit the non-activated values of the neurons from the previous run. Together Elman and Jordan networks constitute the very basics of recurrent neural networks and are called *simple neural networks*. As said before, the Jordan network is generalised $ARMA(p, q)$ to non-linear specification.

2.3.3 General result about neural networks

Let us be a bit more mathematically precise in this part. There are several important results about neural networks that are worth stating in their precise form.[8] As I have illustrated in previous parts neural networks are often generalizations of well-known methods that are encountered in econometrics. We might pose a question to what extent can a function be approximated by a neural network and moreover what kind of network should I use. Answer on this question can be found in universal approximation theorem:

Proposition 2.4. *(Universal approximation theorem (Haykin 2007, pg. 231))*

Let $\phi(\cdot)$ be a non-constant, bounded, and monotone-increasing continuous function. Let I_{m_0} denote the m_0-dimensional unit hypercube $[0,1]^{m_0}$. The space of continuous functions on I_{m_0} is denoted $C(I_{m_0})$. Then, given any function $f \in C(I_{m_0})$ and $\epsilon > 0$, there exists integer M and sets of real constants a_i, b_i, and $\omega_{i,j}$, where $i = 1, ..., m_1$ and $j = 1, ..., m_0$ such that we may define

$$F(x_1, ..., x_{m_0}) = \sum_{i=1}^{m_1} a_i \phi \left(\sum_{j=1}^{m_0} \omega_{i,j} x_j + b_i \right),$$

as an approximate realization of the function $f(x_1, ..., x_{m_0})$; that is

$$|F(x_1, ..., x_{m_0}) - f(x_1, ..., x_{m_0})| < \epsilon,$$

for all $x_1, ..., x_{m_0}$ that lie in the input space.

It is evident from the proposition that any *reasonable* function can be approximated by a neural network with one hidden layer and finite number of neurons in such hidden layer. However, this theorem is only existence proof that generalizes approximations by Fourier series and it is important to concentrate on what the theorem does not say. First, it does not state that such approximation is optimal for training or that such neural network will generalize well. To illustrate practical concerns with topology choice, Barron (1994) derives approximation bounds based on the feed-forward network described in the previous part.

8. I follow more or less closely Haykin (2007) and the propositions are stated in the original form from the text.

First, the approximation bound is defined by

$$\int_{B_r} (f(\mathbf{x}) - F(\mathbf{x})^2 \mu) d\mathbf{x} \leq \frac{2rC_f}{m_1},$$

where C_f is first moment of function $f(\cdot)$ which is assumed to be finite and r is the diameter of the ball B_r. From this result it can be seen that I would like m_1 (number of perceptrons in our case) to be high to get high accuracy of approximation. However, the same moment can be empirically approximated as follows

$$\frac{1}{N} \sum_{i=1}^{N} (f(\mathbf{x}_i) - F(\mathbf{x}_i))^2 \leq \mathcal{O}\left(\frac{C_f^2}{m_1}\right) + \mathcal{O}\left(\frac{m_0 m_1}{N} \log N\right),$$

which implies exactly opposite – I should try to keep m_1 as low as possible to get the best approximation. Hence, I am facing two results that oppose themselves and one has to strike the best compromise.

Another important concern is so called *dimensionality curse*. One can derive that the exact number m_1 depends on m_0, the bigger m_0 is the bigger number of perceptrons one will need to approximate the function. Intuitively, it is clear that functions defined in higher dimension will be harder to approximate. Because neural networks are semi-parametric models, i.e. I have to specify at least order of the model, one would like to have set-up high amount of neurons in the input layer (include all the explaining variables) and let the model train. However, the dimensionality curse stipulates that I anyway have to carefully choose the amount of input I include because the more I include the harder it will be to train the network.

There are various ways how to mitigate these drawbacks. General approach to this problem is dynamic construction of topology during training part or penalizing non-zero size of weights in the network in the sense similar to ridge regression. Dynamic construction of topology means adding or removing perceptrons during training based on some criteria. There is variety of the destructive algorithms – called pruning algorithms in neural networks slang – and several constructive algorithms, one prominent example being cascade correlation. This thesis however uses only the basic neural network topologies that can serve as a baseline for any further improvements and tweaks.

2.3.4 Neural networks training

Once the model is set-up the only remaining step is to train the model, which is, sadly, non trivial. First of all, the training algorithm that I described briefly in previous part, back-propagation, is usually not optimal or slow in training. Moreover, one has to find a way how to discriminate the over-fitted neural networks from models that generalize well.[9] I do not address the issue of choosing the optimal learning function (other option than back-propagation) in here because of complexity and little added value to reader, but I do introduce ways how to asses predictive ability of the model and the overall state of training of function because these can often be useful even in predictive methods in econometrics.

First of all, because I am building predictive models, I need them to generalize well to *unseen* data, otherwise called out-of-sample data. However, as neural networks are universal approximating algorithms, I can always find an algorithm that will fit the in-sample data with very high accuracy, but this model is likely to generalize poorly on the new data, i.e. to over-fit the in-sample data. This problem can be mitigated by using cross-validation paradigm with early stopping.

Cross-validation is based on a simple idea. Because I need to asses the predictive powers of the model, I take part of the data and remove it from the learning process and periodically compute error based on this data and also error of the data on which the model is trained. For example, I compute the error for every 50 epochs[10] while letting the model to train for 1000 epochs. This approach will result into so called learning curves which typically should look like in Figure 2.3.

We can see that the error on the training set is decreasing in every epoch which is secured by the algorithm itself, but the error in cross-validation set starts to grow at some moment. This is the moment when the network is supposed to generalize the best for any other data.

It may seem that such approach will decrease amount of data that I have for training,

9. Generalise loosely means in this context that the model is not over-fitting and having reasonable predictive power.

10. Epoch is a specific name for iteration of learning in machine learning.

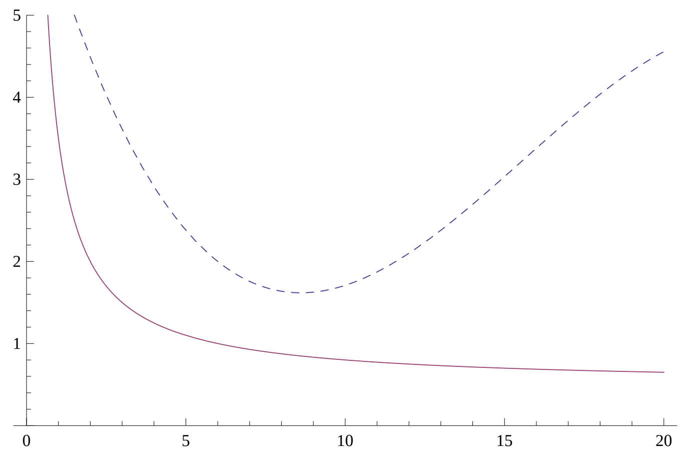

Figure 2.3: Example of learning curves, dashed line: cross-validation error, line: training error. x-axis: epochs, y-axis: error.

however the method can be applied in a way that I split the in-sample data to two equal parts (in number of observations otherwise I select randomly) and first train the neural network on the first half and cross-validate on second and then do the process vice versa. This approach can be generalized into k-fold validation method, where the data are split into k equal parts and then neural network is trained k times on $k-1$ of the folds and cross-validated on the remaining one. The results are averaged afterwards. Moreover, the neural network is trained beginning with random weights and it is possible that the training will get stuck in some local optima so one should try training the model several times in order to get the best solution, preferably global optimum. The cross-validation approach is however problematic while applied to time series data, because it ignores the time dimension of the data which is very important. Nevertheless, cross-validation can be generalised to time series by using time windows.

It is also important to explain the notion of in-sample data and out-of-sample data. One could make a mistake of thinking that I can asses the error of the model based on the cross-validation set. However, that is not true, because as you may have noticed, cross-validation is part of the decision process about the final model. Hence, I cannot asses the predictive accuracy of the model by this set of data. We have to keep some out-of-sample data to get the final error of the forecasting process. To reiterate the whole process of data partitioning, let me suppose I have 100 observations and for simplicity let me assume cross-section data. First, I will form an in-sample and out-of-sample partition in certain proportion that I set. Usual values for the in-sample vs. out-of-sample proportion are 85:15. The observations should be sampled to the sets randomly. The in-sample data are data on which the neural network model is selected and trained. The out-of-sample data should not in any way interact with the model until it is selected as final. Afterwards, I partition the in-sample into cross-validation and training sets. In the simple example of 2-fold validation the in-sample data are divided into equal parts again randomly. On this in-sample data, two estimations are run and then averaged model is produced based on cross-validation method.

In the empirical part I use the least complicated method of cross-validation where I cross-validate just once and then retrain several times on the same sample to get the best optimum.

Chapter 3

Empirical part

3.1 Data description and preprocessing

The empirical analysis performed in the thesis is based on three particular datasets.[1] The datasets were chosen to investigate in different types of items that are traded on the markets and to see whether and how the analysis differs with given type. In particular, I chose one stock index – *S&P 500*, one commodity from metals, namely *gold*, and lastly I chose a particularly interesting commodity – *oil*.

All the data are high-frequency data obtained from Tick Data.[2] The raw dataset consists of one-minute closing prices together with the given time and date. The processing from ticker data to one minute data was done on the side of provider of the data.

Even though the data were obtained in processed form, there are some peculiarities connected with using a particularly long time series and moreover all three assets have gone through a considerable economic development throughout the years. In the next part, I describe problems connected with using long-term time series, how the data for final estimation were selected and then I provide individual description of the data with descriptive statistics.

1. I use interchangeably time series and dataset.
2. Tick Data, GLOBAL HISTORICAL DATA SOLUTIONS 10134-G Colvin Run Road, Great Falls, Virginia 22066 USA

3.1.1 Issues in using long-span time series

All econometric methods are built on simplifying assumptions that real-world data usually do not fulfil. In this part, I describe two most painful departures from the assumptions for this thesis.

The first is assumption of the data being strictly equidistant. Equidistant time series means that all the subsequent observations are realised in intervals that are equally apart from each other. For a financial series to be equidistant I would have to have that for a minute data, there would have to be an observation every minute. The problems that arise from having a non-equidistant observations is primarily heteroskedasticity of the given observations. Let us have the price process which is only reflecting some information process (or latent price process) that does not stop with market trading. Interpreting the time series in this way, it is always only a snapshots of the latent price process. If I would assume the underlying latent process to be standard Wiener process it is known that variance of such process is proportional to time. (The integrated variance from the methodological section.) Hence, if I stop registering the observations, the volatility of the first observation after the pause will be proportional to length of the pause in trading. These observations hence highly bias the estimation.

The most non-equidistant observations in financial datasets arise from exclusion of weekends from trading. Especially in my case, when I would be treating the dataset with wavelets, or even spectral analysis, that creates additional problem, because the *problematic* observations that cumulate a lot of volatility (typically Monday mornings in old times when there was no trading on weekends) are equidistantly spaced and wavelet method could translate this into a special importance of these observations to the overall volatility. This fact has to be kept in mind during preprocessing of the data. In common literature the first observations are usually stripped from the data to acquire unbiased estimate of the volatility of underlying process. To illustrate this effect look at Figure 3.1.

One can see there that in both subsequent days the very first observation is much larger in magnitude than most of the following observations. The magnitude itself is not biggest problem, but the problem is, as described above, that the trading begins always at the same time and hence the spectral methods would *overvalue* this observations.

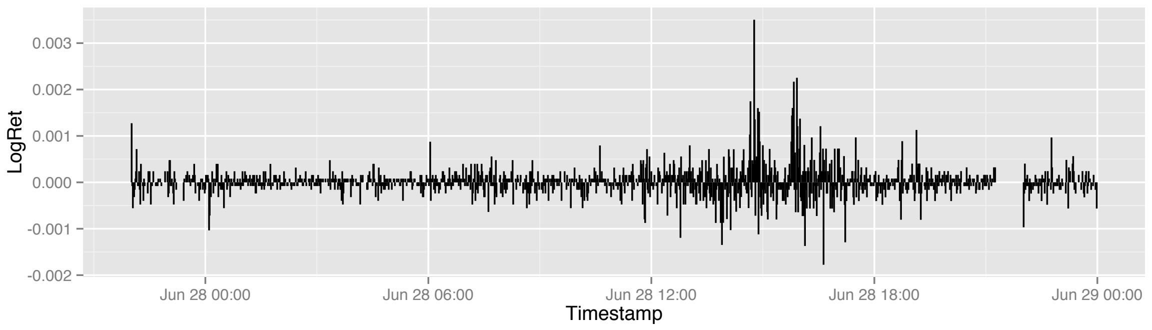

Figure 3.1: Log-returns, illustration of the excessive returns after trading pause, Gold

The second source of non-equidistant observations arise from specificities of working calendar and bank holidays. Fortunately, those can be traced down and disregarded. More problematic days are those where stock exchanges did not work for specific reasons, i.e. after the 9/11 New York Stock Exchange was closed down for four days.[3] In the thesis, I will take a particular approach when I disregard 10 days for each year with fewest number of observations. This rather simplistic approach will remove all exceptionally illiquid days.

The original time series are about 25 years long and hence the revision in the institutional background is understandable and expectable, especially because of the financial markets globalisation and evolution of computer networks. Concretely in the data, I encounter change from 300 observations per day for all three datasets to 1440 for the two commodities and 400 for *S&P 500*. Let us take an example of crude oil. One can see graphically how the number of daily observation evolves in the Figure 3.2.[4] It can be seen from the Figure 3.2 that the market structure – the opening hours and days – are more or less stable from the beginning until 9/11 when there is an instant decrease in trading hours. But more important structural difference comes in 2002, when the market starts to open on Sunday night and the liquidity in market slowly but visibly increases as traders use more and more of the new trading hours.

These changes are more subtle to understand and underpin their effects. However, the

3. The exceptional closing of the New York Stock Exchange can be found at `http://www.nyse.com/pdfs/closings.pdf`.

4. Figures for the rest of the three assets can be found in Appendix A

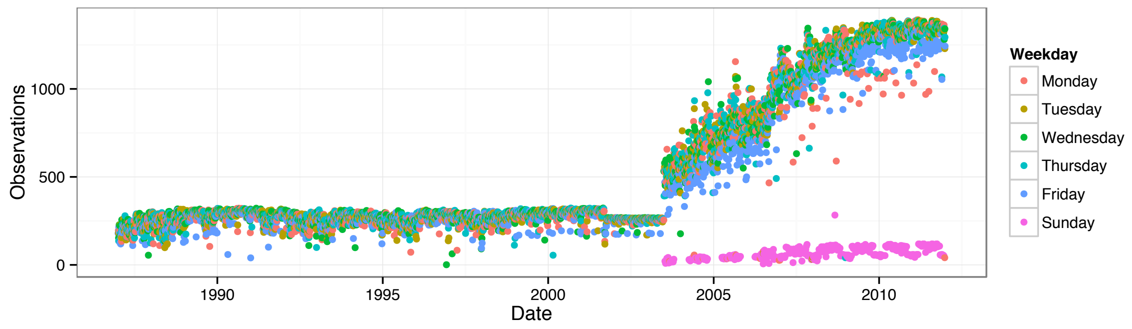

Figure 3.2: Number of observations per day, Crude Oil, whole sample

explanation of the effects of structural changes are not subject of this thesis, on the other hand I concentrate on assessing the forecasting methods and realised volatility measures and hence I would like to have sample that is relatively coherent in structure. Because of these reasons I will limit myself to use of only several last years to capture the current structure of the market. Moreover, all data will be trimmed to exclude illiquid parts of the trading days, more specifically Sundays and late night trading.

Years that I use in the estimation were selected on basis of two main criteria. First, because I will be using machine learning methods for forecasting, I need sufficient amount of data to be able to train the methods efficiently. This number depends on type of learning algorithm that one uses, however, after computing various possible values I should have about 1500 days which translates into 5 years.[5] Second, I have to take into account a historical fact of the crisis of 2008 that it will make part of the sample. To make the algorithm to generalise well I have to include enough *calm* times data.

Hence, the time span selection is a compromise based on goal I want to achieve, methods I am using, and historical reality. I want the model to generalise well for forecasting which means that the model has to underpin the most current structure of the market, so I would like to use relatively short time series. On the other hand I am using a method that is the better the more data I use. Moreover, there is the crisis of 2008 when volatility shoots up so I

5. There is no clear cut approach to tell amount of data necessary for the training process. Several approaches are suggested in the literature and the necessary number of observations is usually dependent on number of weights in the network.

also need to use enough other data to provide sufficient learning material for the algorithm. Combining all the previous requirements I settle in here for the period between years 2007 and 2011.

Precise description of the preprocessing algorithm used can be found in form of flow chart in the Appendix A and source codes. Other quantitative and graphical description of the data is also deferred to the Appendix A.

3.2 Realised volatility estimation

Having prepared the data for the analysis I can get to the important part of the empirical analysis. First, I describe the parameter choice made in the analysis and then I describe and compare the results.

Daily volatilities based on 1 minute intra-day data will be estimated during the exercise. As suggested in the methodology, Section 2.1, I will estimate realised volatility using several estimators that are suggested in the literature. Namely these are: sparse (sometimes called naive), bi-power variation, two-scale realised volatility, jump-adjusted wavelet two-scale realised volatility and kernel estimators. Theoretical construction of the estimators is described in the aforementioned section. However, for the estimation I need to select several parameters.

First, I need to select grid for sub-sampling in the two-scale estimators. Following literature on realised volatility where 5 minute second scale is suggested to perform well for most data I set the parameters accordingly. Second, I need to choose number of decomposition levels for the wavelet two-scale estimator. The number itself is, of course, limited proportionally by number of intra-day observations. More specifically because of the character of wavelet decomposition the maximal number of levels is $\lfloor \log_2 \min(n_d)/s \rfloor$, where n_d corresponds to number of intra-day observations and s is scale parameter which is equal to 5 in my case. So theoretically, if I take for example *gold* where the minimal number of intra-day observations is 454 and I take scale to be 5 minutes I could have 6 levels at maximum. To be coherent across the assets, I find that the lowest number of levels possible is 4 in case of *S&P 500* and hence set the number of levels to 4.

One of the assessing tools whether the estimated realised variance performs well is to look at distributional parameters of the daily log-returns standardised by realised volatility, because these should be, as suggested in Andersen et al. (2001) very close to normal distribution. The descriptive statistics can be found in Tables 3.1, 3.2 and 3.3.

	Mean	Std.dev	Skew	Kurt
bpv	-0.05	1.04	-0.04	-0.28
jwtsrv	-0.05	1.11	-0.04	-0.23
kernel	-0.05	1.03	-0.06	-0.29
naive	-0.04	1.02	-0.04	-0.32
tsrv	-0.05	1.03	-0.06	-0.29

Table 3.1: Distribution description of Crude oil log-returns standardized by $RV^{1/2}$

	Mean	Std.dev	Skew	Kurt
bpv	-0.10	0.98	-0.01	-0.26
jwtsrv	-0.10	1.03	-0.00	-0.20
kernel	-0.10	0.94	-0.00	-0.37
naive	-0.09	0.94	0.01	-0.32
tsrv	-0.10	0.95	-0.00	-0.37

Table 3.2: Distribution description of Gold log-returns standardized by $RV^{1/2}$

	Mean	Std.dev	Skew	Kurt
bpv	-0.13	1.31	-0.14	0.06
jwtsrv	-0.14	1.41	-0.22	0.26
kernel	-0.07	1.04	0.11	-0.48
naive	-0.07	1.05	0.11	-0.43
tsrv	-0.11	1.26	-0.04	-0.25

Table 3.3: Distribution description of S&P 500 log-returns standardized by $RV^{1/2}$

The tables conform to the hypothesis that the unconditional distributions are very close to normal distributions. Interestingly, means of standardised log-returns are consistently negative, though close to zero, which is surprising because one would expect the deviation from normality to be into positive or ambiguous direction. The negative direction suggests a non-substantiated market volatility risk-taking by the agents in the market. This effect is statistically significant if I bootstrap the standard deviation of the mean. In other words, if there is low volatility and high returns, it is predictable that the investors will take the chance and will trade. But in our case the negative mean suggests that investors became risk-taking on average in this period.

Standard deviation is also very close to one as I would expect after performing standardization. This normality evidence is supported by normality tests, that can be found in Tables A.4, A.5, and A.6 in the Appendix A. The tests do not uniformly confirm the hypothesis of normality for all the time series, however it is well-known that with increasing number of observations the normality tests become *too powerful* and even small deviation from normality can cause the tests to reject the hypothesis. That is also the case, because I have about 1200 observations in each time series. Hence, even though some of the tests reject normality, the p-values are not very persuasive in the light of statistical theory especially when I am interested checking approximate normality. I also include a QQ-plot of the results. In Figure 3.4 this plot is produced.[6]

The plot is important primarily due to the reason that I can see more deeply into the working of used methods. For example, BPV estimator should be robust with respect to jumps, but is not robust to micro-structure noise. On the other hand TSRV estimator is robust to micro-structure but not to jumps. And finally JWTSRV should be robust to both. In the Figure 3.4, I can inspect, how different methods treat *outliers* that appear in case of the jump and noise inconsistent estimators. The outliers most probably contain high jump variation and the jump-robust method should have less of them. Indeed, in case of crude oil and gold, it is visually evident that JWTSRV estimate gets rid of the outliers at the end in comparison with the other methods. It is surprising that the BPV estimator does not seem to do as good job in case of these two assets.

Apparently a very different story emerges for *S&P 500* because the jump-consistent estimates seem to do considerably poorer job than other methods. Paradoxically results of normality tests show that in some distributional parameters the *naive* estimate seems to be closest to the normal distribution. Moreover, from the results of normality tests I can claim that TSRV estimate is uniformly better then both JWTSRV and BPV which both should be jump-consistent. It seems that the jump consistent methods have a problem with the *S&P*

6. Another representation in form of histogram with superimposed density can be found in Appendix A.4

500 dataset and most probably the jump-adjustment process is overdone in this case. This specific result is of great interest because the suggested consistency against jumps is worsening results and that is in contradiction with literature which does not emphasise that such result is possible. However, if I look at the construction of the jump-consistent estimators, it is evident that they assume at least some jumps in the data to work. *S&P 500* is a very specific case, which aggregates 500 different titles and probability that there will be a significant jump in the data is very low because it would mean a jump in the whole market. Hence, the methods will have lower threshold for detecting jumps and a lot of jumps will be detected. Jump-adjustment is not constructed for such situation because the theoretical model accounts only for individually occurring jumps, but in case that the jumps would occur sometimes even in pairs, the jump-adjustment could even worsen the situation by adjustment.

Another interesting part of the estimation is the decomposition of the volatility into different investment horizons as allowed by JWTSRV estimator. First of all, the decomposition shows us, how the volatility is created in the time series. It shows how much of the volatility is short-time and what part persists on longer-horizons. Important question that arises is whether the percentual composition of the volatility changes throughout the time and does it change during high volatility times such as crises? Apart from that such a decomposition has an important implication into a risk-management, because the percentage decomposition suggest on what horizon should I concentrate forecasting efforts. For example, if most of the variance is created at the 5-10 minutes scale, as will be the case later.

The decompositions for different assets can be found in Figure 3.3. Even from the first look, there are several interesting observations that I can make. First, the percentage decomposition is surprisingly stable across the assets. The scale 5-10 minutes keeps about 50% of the influence in the observations and scale 10-20 minutes has around 25%. Second, as common sense would suggest most of the volatility is created on the fastest scale which basically means a white-noise. These findings further support Baruník (2011) who found the same composition in the FX markets, namely GBP, USD and CHF. As the market composition is the same in here, and it is reasonable to suspect that it will be approximately same in other cases due to the further evidence in literature.

Having estimated and analysed the time series that will be used for the forecasting ex-

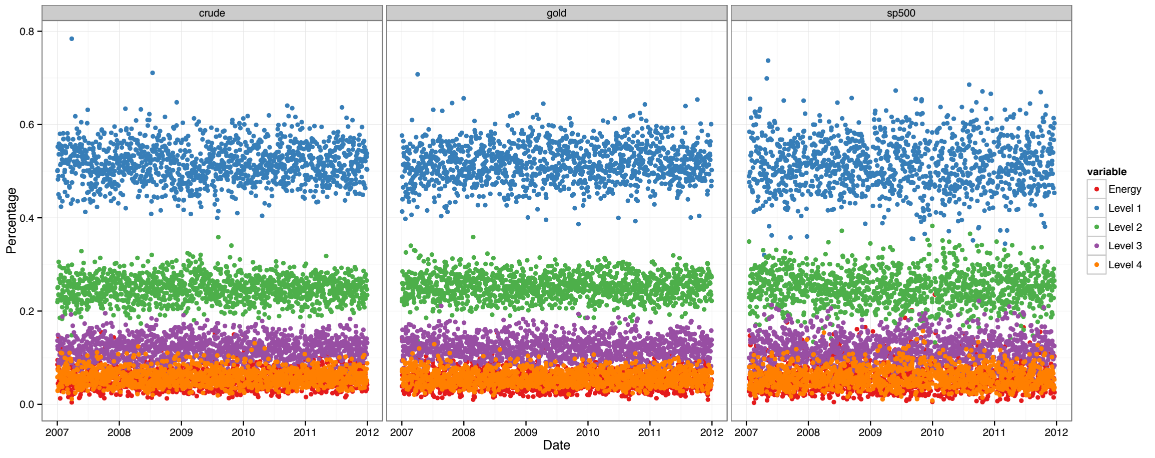

Figure 3.3: Percentage decomposition of $RV^{1/2}$

periment, I can proceed to describe the process of building predictive models for this data.

3.3 Quantitative models estimation and evaluation

Prediction of realised volatility is a popular problem in forecasting literature and there is a substantive literature already written about this topic. As for the main works in the field, the classic paper Andersen et al. (2003) must be mentioned, several important papers have been published investigating the long-memory autoregressive methods, see for example Corsi (2009), others have been interested in using machine learning approaches to the problem, the most prominent McAleer and Medeiros (2011).

As suggested in the introduction, this thesis will assess long-run and short-run predictions of realised volatility by several methods. In case of short-term predictions, I will make predictions for one-day ahead volatilities for 70 days, approximately three months worth of data. I will also use a mundane approach, where I will take the today's observation to be a forecast of tomorrow volatility, further ARFIMA rolling forecast and finally several types of neural networks. For the long-term predictions, I will try to predict volatility 5 days, 10 days and 20 days ahead cumulatively using today's data. Cumulative volatility was chosen merely because it is more useful in practice than only estimating the volatility some certain

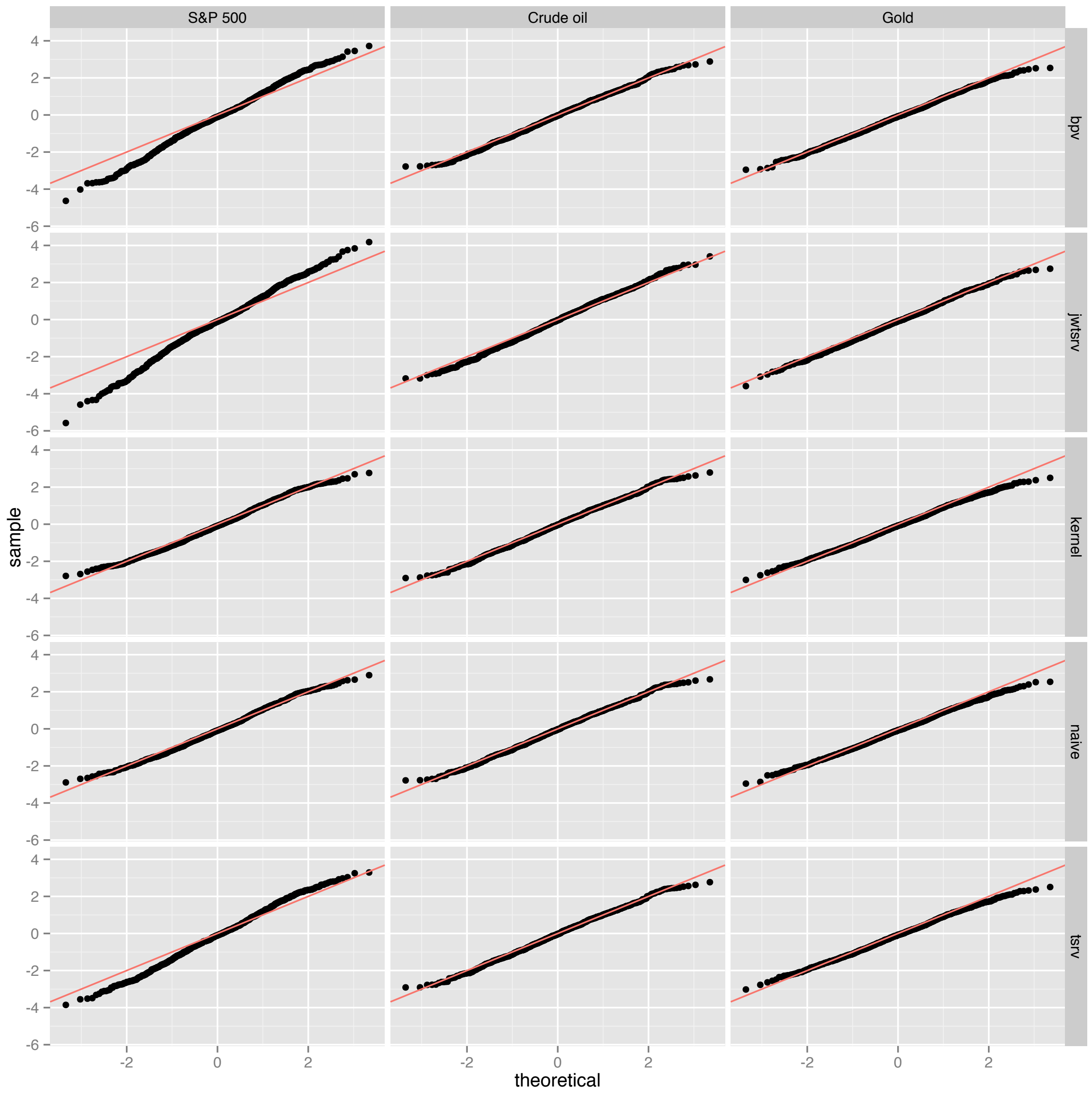

Figure 3.4: Standardised log-returns by $RV^{1/2}$

amount of days ahead. The same methods will be used to predict cumulative volatilities as in the short-term exercise. Mundane, so called lame henceforth, forecast will be constructed as a today's observation multiplied by the period length, ARFIMA will be again rolling forecast and neural networks are set up in a way that the targeted variable are not all the observations but cumulative number for the period. This slight adjustment is an illustration of how to simplify problems for neural networks, as suggested in the part about machine learning. Trying to predict all the ten values would be possible, but unnecessary, because I only need the cumulative number.

The several different time-horizons were chosen because theoretically as one supposes the realised volatility process to be a local martingale, and it seems that the best prediction could indeed be today's value and hence the lame approach might actually work significantly better than other approaches. Already canonical paper Meese and Rogoff (1983) illustrates well this problem. The authors take predictions of exchange rates and they assess predictive power of various models against random walk model. The outcome, at that time quite surprising, is that in short-time horizon one cannot find consistently better method than random walk. However, in the case of long-horizon predictions (week-ahead) the lame prediction should work significantly worse then in the short-horizon case because the *sophisticated* models should learn deeper pattern than the lame approach and thus perform better.

Building ARFIMA and lame model is automatic, however, with neural networks, one has to choose architecture and several parameters that cannot be tuned automatically as in the case of ARFIMA model. Let us look at the process of building the predictive network and specific choices about parameters that I made.

The first step to build a neural network is to choose an architecture. As suggested in theoretical Section 2.3 if I am staying in the realm of relatively common architectures, I can use multi-layer perceptron networks or recursive neural networks. As both type of topologies are important – MLP architecture could be looked at as non-linear version of AR process and recursive networks are non-linear MA or ARMA processes – I will use both and see whether more complex networks increase the accuracy of predictions as intuition would suggest. More specifically, I use feed-forward neural network with one hidden layer that is fed by lagged values of the realised volatility. I choose the number of lags to be 3 as increasing this number does not bring significant improvement in predictive accuracy and I have to be aware of the dimensionality curse – by introducing more lags I am trying to model more complex function which might be unattainable using this type of network. As for the recursive neural networks, I use the Jordan neural network and the Elman neural networks. Jordan neural network has a context layer that stores the last predicted values of specified number of lags whereas Elman neural network has a context layer that stores the values of the hidden neurons from the preceding fit. Learning algorithm that was used to fit the neural networks is resilient propagation developed in Riedmiller and Braun (1993) and its variant for recursive neural networks. It is one of the fastest converging algorithms in use. Moreover, all the networks were fitted several times to get the best fit because the training process always starts from random initialization of weights and on that particular path, the training algorithm might get stuck in local optimum. The reiteration was done 10 times, because of the computational burden.

The predicted values will then be assessed by Mincer-Zarnowitz regression and Diebold-Mariano test which are described in detail in the next section.

3.3.1 Mincer-Zarnowitz regression and Diebold-Mariano test

In the evaluation of results I use two main approaches. Firstly, I use the popular approach by Mincer and Zarnowitz (1969), who suggest running a regression in form

$$V_{t+i}^{(m)} = \alpha + \beta_1 V_{t,i}^{(k,j)} + \epsilon_t,$$

where $V_{t+i}^{(m)}$ is the integrated volatility estimated by estimator m at time $t+i$, $V_{t,i}^{(k,j)}$ is the prediction of the i-step ahead cumulative integrated volatility at time t using method k for estimating the volatility and method j for evaluating the prediction. For example, $V_{t,5}^{(BPV,ELMAN)}$ denotes 5-days ahead cumulative integrated volatility on Bi-power variation data using Elman neural network. Moreover, I have one realised volatility estimator, JWTSRV, that decomposes the volatility into several horizons. The predictions for JWTSRV are constructed by forecasting on all levels separately and then adding up the forecasts to form the final prediction. If we would know the true value of the volatility, we could test the two parameters $\alpha = 0, \beta = 1$ to check for the estimate to be unbiased and consistent respectively. However, not having the true value, we still can use the R^2 of the regression which can be perceived as information content of one estimator vis à vis the second one.

Secondly, to compare the two time series of forecasts I use the widely popular test developed in Diebold and Mariano (1995). The test compares two loss functions and assesses whether the two loss functions differ in statistically significant way. Let us assume time series $\{y_t\}$ and its two different h-step ahead forecasts $\{\hat{y}^A_{t+h}\}$ and $\{\hat{y}^B_{t+h}\}$. The errors of the forecast are:

$$\epsilon^A_{t+h} = y_{t+h} - \hat{y}^A_{t+h}$$
$$\epsilon^B_{t+h} = y_{t+h} - \hat{y}^B_{t+h}$$

The loss function over the time series is defined as a function $L(\cdot)$. The typical examples of loss functions are absolute value $L(x) = |x|$ or quadratic function $L(x) = x^2$. I want to test the hypothesis that the expected loss of the forecast is significantly different, which translates to statistical hypotheses

$$H_0 : E\left(L\left(\epsilon^A_{t+h}\right)\right) = E\left(L\left(\epsilon^B_{t+h}\right)\right)$$
$$H_1 : E\left(L\left(\epsilon^A_{t+h}\right)\right) \neq E\left(L\left(\epsilon^B_{t+h}\right)\right)$$

In the Diebold-Mariano test, loss differential d_t is computed as

$$d_t = E\left(L\left(\epsilon^A_{t+h}\right)\right) - E\left(L\left(\epsilon^B_{t+h}\right)\right)$$

and the test statistic

$$S = \frac{\bar{d}}{\sqrt{\widehat{\sigma}^2_{\bar{d}}/T}},$$

where $\bar{d} = \frac{1}{T}\sum_{t=1}^{T} d_t$ is the mean value of the loss differential and $\widehat{\sigma}^2_{\bar{d}} = \gamma_0 + 2\sum_{j=1}^{\infty}\gamma_j$, $\gamma_i = cov(d_t, d_{t-i})$ is the estimate of the long-run variance. Diebold and Mariano (1995) show that under the null hypothesis $H_0 : E\left(d_t\right) = 0$, the statistic S is asymptotically distributed as $S \sim N(0,1)$. Moreover, we can distinguish the *better* value by the sign of the statistic S.

3.3.2 Results of the forecasting methods application

First of all, it is worth mentioning how and why the results will be represented in the form as I do represent them – mostly in form of figures and diagrams – which is much less standard in econometric and statistical literature. Apart from my belief that figures present results in much faster way than tables, if done properly, another important aspect rises in this work. The thesis deals with rather large amount of data and comparison of several methods and often I encountered a problem how to present about 700 numbers in coherent and understandable way that will, moreover, allow for comparison of the numbers in meaningful way. Even printing this amount of data on one page is next to impossible not speaking about well-structured understandable table.

Let us first look at the Mincer-Zarnowitz regressions and the respective R^2. The statistic measures the information content of the estimator method with respect to the estimated method. In Figures 3.5, 3.6, 3.7, and 3.8 one can find results for in- and out-samples of one day ahead and five days ahead estimations. For the sake of saving space, the rest of the figures for horizons of 10 and 20 days are deferred to the Appendix A, Figures A.5, A.6, A.7, and A.8.

The figures shows R^2 results for the Mincer-Zarnowitz regressions for all the assets and methods that I used. Every part of the facet describes a method and asset combination. The graph itself shows what R^2 was attained for various combinations of explanatory and explained methods of estimating the realised volatility. For example, if on one line a given shape has the highest R^2, i.e. is farthest to the right, I can say that this method is most easily explained by the explaining method. As I will see in the Figure this is usually the case for JWTSRV estimator.

Let us examine the Figures A.5, A.6, A.7, and A.8. First, I look at the short-term one day ahead fitted values and forecasts. Surprisingly, on the short-term, neural networks are performing rather well when compared to the other methods. The amount of explained variance is also, as expected, very high – in the case of in-sample traditional methods attain from 75% to 80% and neural network methods attain about 90%. The neural networks are

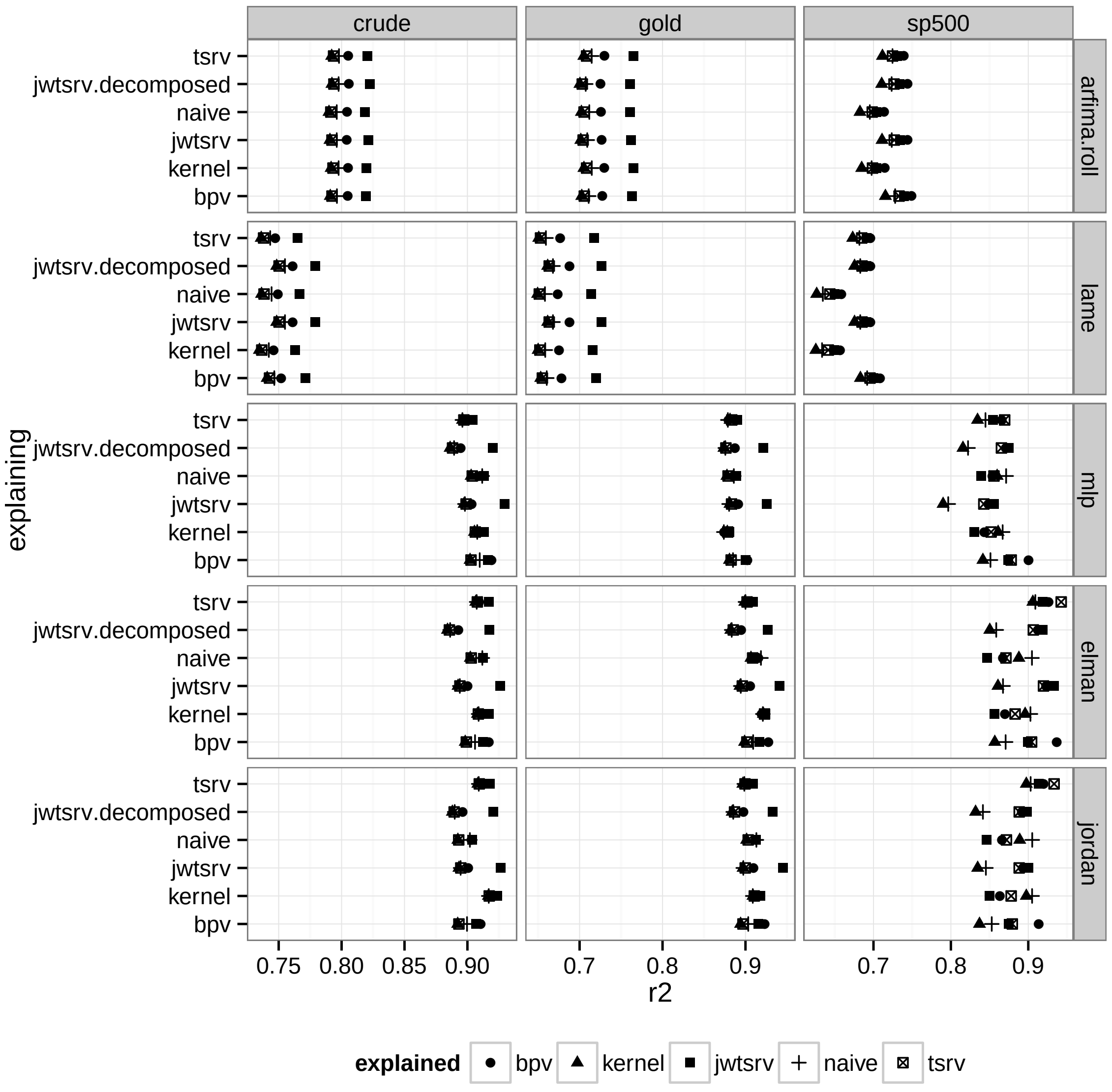

Figure 3.5: R^2 from Mincer-Zarnowitz regressions, one day ahead, in-sample

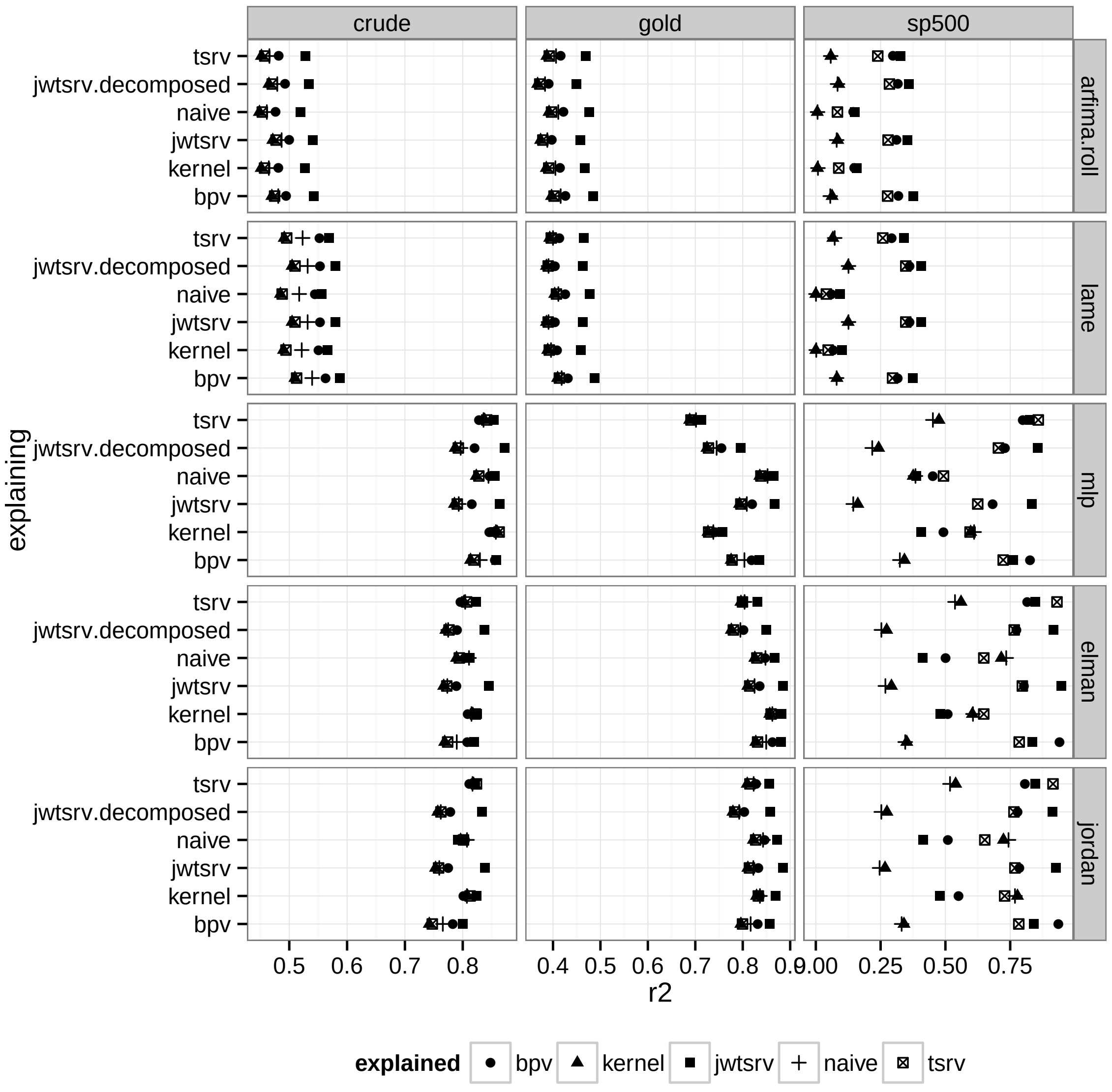

Figure 3.6: R^2 from Mincer-Zarnowitz regressions, one day ahead, out-sample

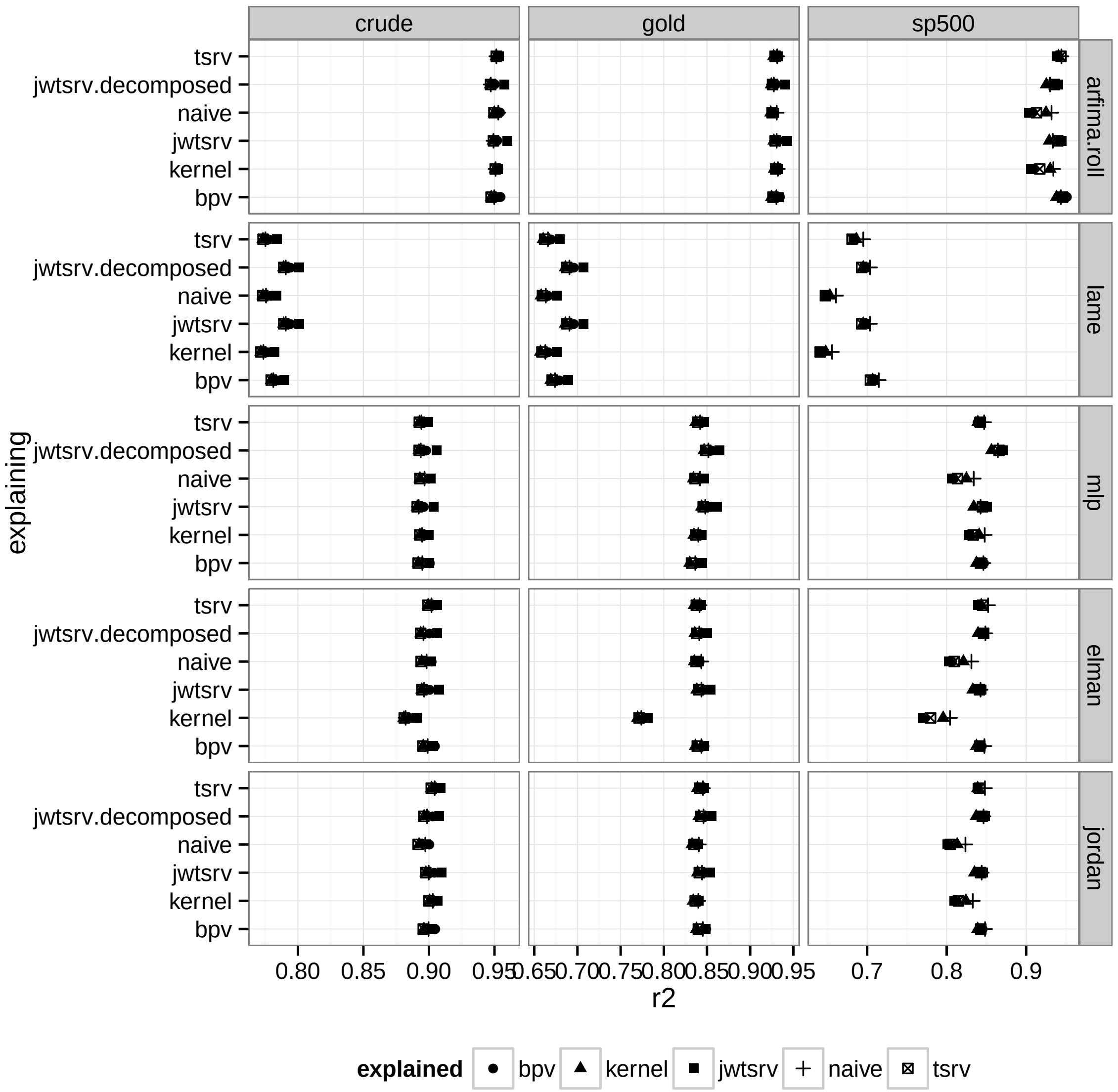

Figure 3.7: R^2 from Mincer-Zarnowitz regressions, five days ahead, in-sample

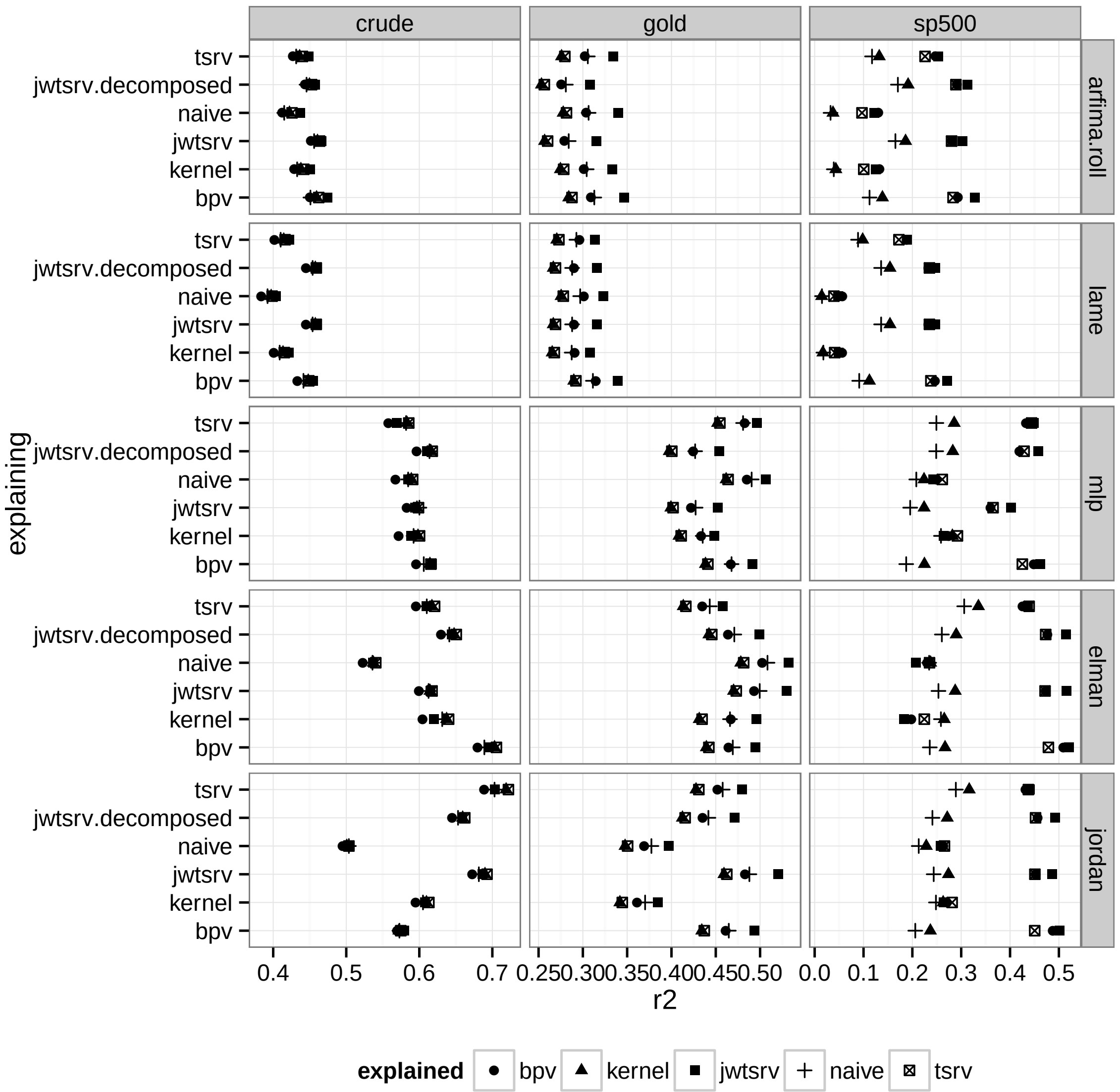

Figure 3.8: R^2 from Mincer-Zarnowitz regressions, five days ahead, out-sample

consistently better and there does not seem to be significant difference between the MLP neural network and the two recursive networks. Hence, the biggest increase in the R^2 seems to be due to the non-linear nature of the neural networks models. Looking at the two *standard* models, the lame estimate and the ARFIMA estimate, in the in-sample, the ARFIMA estimate is slightly better but in the out-sample the ARFIMA estimate performs worse than the lame method. On contrary, in the out-sample the neural network methods outperform the ARFIMA methods by far. This edge in predictive power might be due to the construction of the training process, when the neural-network is stopped to over-fit, when the error rises on the cross-validation set.

Looking at longer-horizons, interestingly, the performance of the lame estimator does not deteriorate much. This result is important because it suggests that the time series moves around its conditional mean and it would probably be even higher if there would not be the data that reflect economic crisis. Another pattern that I find and is worth of attention is that in in-samples, the performance of the ARFIMA method is very good. It seems that the errors are averaging themselves and ARFIMA is becoming more and more precise in predictions on longer horizon cumulative values. However, in the out-samples the performance of ARFIMA is worse and is constantly lower than in case of neural networks. The performance of the networks themselves is quite in line with expectation. In the in-samples the neural networks perform a bit worse than the ARFIMA method because neural networks are restrained from over-fitting by the training process. For the same reasons, they are also better in the out-samples. The exact topology of the neural network starts to matter only on longer horizons. This is sensible because if I suppose a time series with long memory, the autoregressive structure will become more and more pronounced in longer horizons.

If I consider question what estimators of realised volatility I should use for the forecasting and estimation procedure in general, the Figures also can provide valuable insights into this issue. Theoretically, the best estimator should be JWTSRV because it is consistent against noise as well as jumps, so it should be the easiest to forecast. When I look at the figures, I can confirm that JWTSRV usually has the highest R^2 and hence also the information content. Moreover, I can note that selecting an estimator only becomes an issue at longer horizons in out-samples. And also the magnitude how important it is to choose the right estimator is highly dependent on the asset. For example, in case of the crude oil, I can see that selecting whatever estimator will result into fairly similar results on all horizons. On the other hand, in case of S&P 500, different selection might result into 50% R^2 difference in the out-of-sample forecasting. Selecting an explaining method is a case for further investigation and a promising path would most likely be investigation of cross-validation errors in various estimations and their comparison. However, looking into the literature there is one important paper (Liu et al. 2012) which exactly compares if it is possible actually beat the sparse naive estimator in forecasts and there is quite persuasive and discouraging evidence that statistically it is hard to tell the difference.

Moreover, one of the questions was whether the wavelet decomposition caries some additional information or it is only a technical exercise. Theoretically there is a rationale behind the wavelet decomposition, that the variance can be decomposed into several meaningful investment horizons and there might be some interactions between the horizons. This hypothesis is, however, not supported by the results of the forecasting exercise. As can be seen in the Figures A.5, A.6, A.7, and A.8, if the explaining estimator is JWTSRV.decomposed there is no significant and consistent improvement in the performance against other methods. In the case of lame estimator, the Figures suggest equal performance which must be the case due to the construction of the estimators. Approximately same performance is shown in case of all the other methods. Hence, the decomposition does not seem to bring additional information to the forecasting. Further investigation of this hypothesis would have to try predicting all the series at once and levering covariance structure of the ensemble. Even though this approach might turn to be helpful in predictions, it would still lack the interesting economic interpretation that simple decomposition horizon has, i.e. that the market has some deeper structure where different investment horizons have different rules.

Next, I look at the Diebold-Mariano tests for predictive accuracy. I perform the forecast tests on two cases: I test the forecasts errors from the Mincer-Zarnowitz regressions and then I test the pure residuals that are constructed as a difference between the true value of the estimator minus the estimated value. I compare the neural network methods to the ARFIMA estimate and the lame estimation. In the Tables 3.4, 3.5 I present the count of the number of tests that were significant at a given value throughout the whole estimation, on the left in given column is the count of tests that favorizes the neural networks and on the right is count of tests that favorizes the compared method. This can give us an overall picture of the methods.

First, it is important to note, that results from the Mincer-Zarnowitz part from both tables strongly support the results from the previous text and I can see huge difference in numbers of cases when neural-networks outperform both ARFIMA and lame methods. It was not quite evident from the visual inspection of the previous Figures, but from looking at the tables recursive networks seem to be better than the MLP network even for the cost of having more undecided cases of the tests.

Second, as I can see, the results from the part where I compare the true value to the estimated value forecast indicates quite the opposite, or at least gives much more inconsistent picture, from the previous part of the table as a lot of forecasts are evaluated better by ARFIMA and have much higher number of indecisive test results. This issue is likely caused by one feature of training of neural networks. Before the neural network is trained, one has to standardise the inputs[7] and hence the predicted values have to be converted back by an inverse transformation. This inverse transformation in my case is done by taking the same mean and standard deviation as for the training sample. Hence, because the Mincer-Zarnowitz is an equivalent to re-standardization it is logical to argue that the inverse standardization of the output from artificial neural networks is wrong.

This contrasting result implies that the estimation predicts well the dynamics, however the scale and location are not quite right. How to estimate the two parameters remains an open question for further research. A promising way might be using estimate of the values from closer period to the training sample as it better reflects the current state of the market. Or combination of using ARFIMA estimate for mean and standard deviation and applying those to invert the standardization.

To sum the results of the empirical part, I first estimated the realised volatility measure and looked at the standardised log-returns, which should be normal, when standardised by the realised volatility. This theoretical result is confirmed in most of the cases and assets, however, interestingly standardisation of *S&P 500* by jump-consistent estimators of realised volatility, i.e. JWTSRV and BPV, move the returns further away from the normality. The likely cause is that in the theoretical model upon which the jump-adjustment is based, oc-

7. It is not a necessity, but it is considered to be a common practice, because without standardization the training converges only very slowly.

		Minzer-Zarnowitz residuals						True-estimated residuals					
	Significance	MLP		Elman		Jordan		MLP		Elman		Jordan	
Crude	0.025	317	0	298	0	233	0	422	0	461	0	458	0
	0.05	45	0	62	0	56	0	25	0	23	2	38	0
	0.1	89	0	96	0	89	0	42	0	35	1	35	0
	Insignificant	269	0	264	0	342	0	225	6	165	33	177	12
Gold	0.025	353	0	223	0	249	0	37	0	157	4	130	0
	0.05	75	0	2	0	4	0	27	0	64	0	49	0
	0.1	85	0	8	0	7	0	35	0	80	4	91	0
	Insignificant	164	43	453	34	426	34	590	31	389	22	450	0
S&P 500	0.025	178	23	157	16	150	21	302	101	365	79	372	90
	0.05	46	4	31	6	35	4	27	8	34	14	27	9
	0.1	64	1	68	5	61	13	29	10	37	3	28	15
	Insignificant	333	71	355	82	373	63	164	79	113	75	112	67

Table 3.4: Number of given result for Diebold-Mariano test in a group when compared to lame method. Left numbers are favorizing neural networks, right the alternative method.

currence of two jumps that follow each other is close to impossible and in case of *S&P 500* such corrections can occur. Success of this standardization is also confirmed by several normality tests, and this can be considered quite spectacular, because with big amount of data (more than thousand observations in the sample) normality tests are immensely powerful and even small divergence from normality causes refusal of the normality hypothesis. After the estimation exercise, forecasting exercise follows.

Canonical ARFIMA model is fitted accompanied with *lame* estimation. These two represent the more or less standard forecasting methods and the *lame* estimate allows to account for the Meese-Rogoff paradox which established that a most of the forecasting methods only show negligible improvement against unintelligible Brownian motion. Moreover, I fit three standard types of neural networks and compare them with the standard methods. Analysis by means of Mincer-Zarnowitz regressions and Diebold-Mariano tests brings about interesting results. Neural networks have higher informational content concerning the complicated dynamics but fail to be very good predictors because their scale and location parameters are estimated wrongly. This is caused by one of standard techniques used in the training of neural network, the pre-standardization of data and the back-transformation of the forecasts.

		Minzer-Zarnowitz residuals						True-estimated residuals					
	Significance	MLP		Elman		Jordan		MLP		Elman		Jordan	
Crude	0.025	409	56	346	0	300	0	255	165	234	131	268	92
	0.05	9	4	62	0	67	0	8	20	23	14	20	25
	0.1	25	2	51	0	69	0	11	15	27	16	24	28
	Insignificant	185	30	261	0	284	0	118	128	127	148	114	149
Gold	0.025	194	0	333	0	419	0	9	14	81	23	70	0
	0.05	17	0	63	4	53	0	11	7	21	4	15	0
	0.1	52	0	76	13	65	0	11	21	24	14	17	1
	Insignificant	421	36	218	13	162	21	141	506	137	416	173	444
S&P 500	0.025	206	24	194	13	201	9	142	278	186	169	167	226
	0.05	47	0	32	1	36	5	16	15	22	17	16	16
	0.1	52	3	61	8	61	8	17	19	26	23	21	21
	Insignificant	317	71	334	77	332	68	112	121	131	146	131	122

Table 3.5: Number of given result for Diebold-Mariano test in a group when compared to ARFIMA method. Left numbers are favorizing neural networks, right the alternative method.

Moreover, it is visible from the results that the forecasting is highly asset-dependent and also horizon-dependent. When concerned with dynamics, neural networks usually outperform the standard methods in all the cases.

Chapter 4

Conclusion

The volatility literature is an important and significant part of financial econometrics. The question of estimating volatility process is slowly closing with favourable results and prevailing paradigm of realised volatility. One of the most advanced realised volatility estimators in the literature is proposed in Baruník (2011). It uses wavelets and allows for decomposition into investment horizons. Various estimators have various attributes and these should be carefully compared.

In short, my thesis has two significant parts. First the estimation, I estimate on wide range of assets the wavelet realised volatility estimator, benchmark it against the orthodox estimators used in the literature. Moreover, I test the hypothesis that wavelet decomposition brings more information for forecasting. Second the forecasting, I build artificial neural network models for forecasting volatility on several time-horizons. I benchmark these models against common ARFIMA method and provide evidence that ANN should be used more in the empirical literature.

The realised volatility estimation procedure confirms the broad picture of results that can be found in literature about realised volatility estimators. This results seem to be very robust across estimators.

First, the theory suggests that standardised returns by the volatility should be normally distributed and this is strongly confirmed in case of *crude oil* and *gold*. In case of *S&P 500* index the results deserve more attention, because the application of jump-consistent models yields the standardised returns to be less normal than in other cases. However, as suggested in the text, *S&P 500* is a very specific and aggregated case which means that occurrence of jumps in the price of this asset is very unlikely. In this situation, because the methods assume occurrence of jumps they over-identify jumps and hence over-adjust the data.

Second, the results confirm, that noise-consistent and jump-consistent are easier to predict because the behaviour should be less erratic and the series is cleared from disturbing effects. The most prominent in this respect is the JWTSRV estimator.

Third, contrary to the expectation the decomposition of the volatility does not show to improve the explanatory power of various methods in simple settings. By simple setting it is meant that we forecast or fit the series only horizon by horizon and we implicitly assume that they are independent. This result is particularly important because it suggests that there is no easily understandable and interpretable structure of the market. Rather it seems that market cannot be looked at as segmented by investment horizons but we should really look at market as *investment continuum*.

After performing the estimation, I moved to forecasting procedure. The set-up of the exercise was to forecast cumulative volatilities over several time horizons. I used standard rolling ARFIMA method, lame approach that benchmarks against getting results that are inferior to very simple and intuitive method and artificial neural networks. To provide broader insight recursive as well as standard multi-layer perceptron network topologies were used.

First, as one would suppose, it is often very hard to get better results than the very simple method in short-horizon. This result is known for a long time in finance and several paradoxes can be found in literature. Two typical examples are the Meese and Rogoff (1983) paradox when forecasts benchmarked against Brownian motion are not significantly better and the well-known paradox that it is very hard to get better portfolio returns than equal weight portfolio.

Second, this thesis brings new evidence about usability of artificial neural networks in forecasting of realised volatility. Simple MLP network as well as Elman and Jordan neural networks were used. Neural networks proved to have higher information content about dynamics of the volatility series than ARFIMA or lame methods that were used as a benchmark. The typologies performance does not differ which is an interesting result, because MLP is only non-linear variant of AR process. The only short-come is that the networks suffer from post-standardisation of results, which is left as an open question for further research.

The thesis is novel in several aspects. First, I widen the analysis of volatility to three substantially different assets *crude oil, gold* and *S&P 500.* Furthermore, I apply the state-of-art estimation methods of realised volatility on them, including wavelet estimator that is able to decompose volatility into different horizons. Results of estimation mostly support the existing literature on the topic. Secondly, I built neural network models for forecasting and showed that they make better predictions about dynamics than the existing method. I only used simple topologies and deeper investigation in this area is left for further research.

Appendix A

Complementary tables and figures

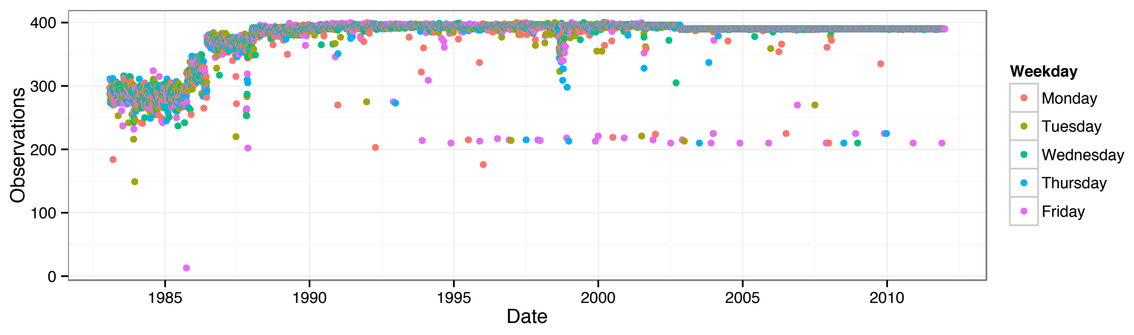

Figure A.1: Number of observations per day, S&P 500, whole sample

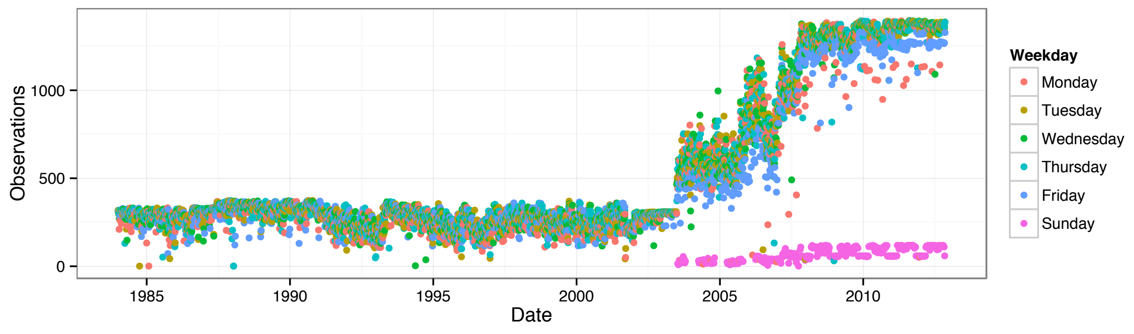

Figure A.2: Number of observations per day, Gold, whole sample

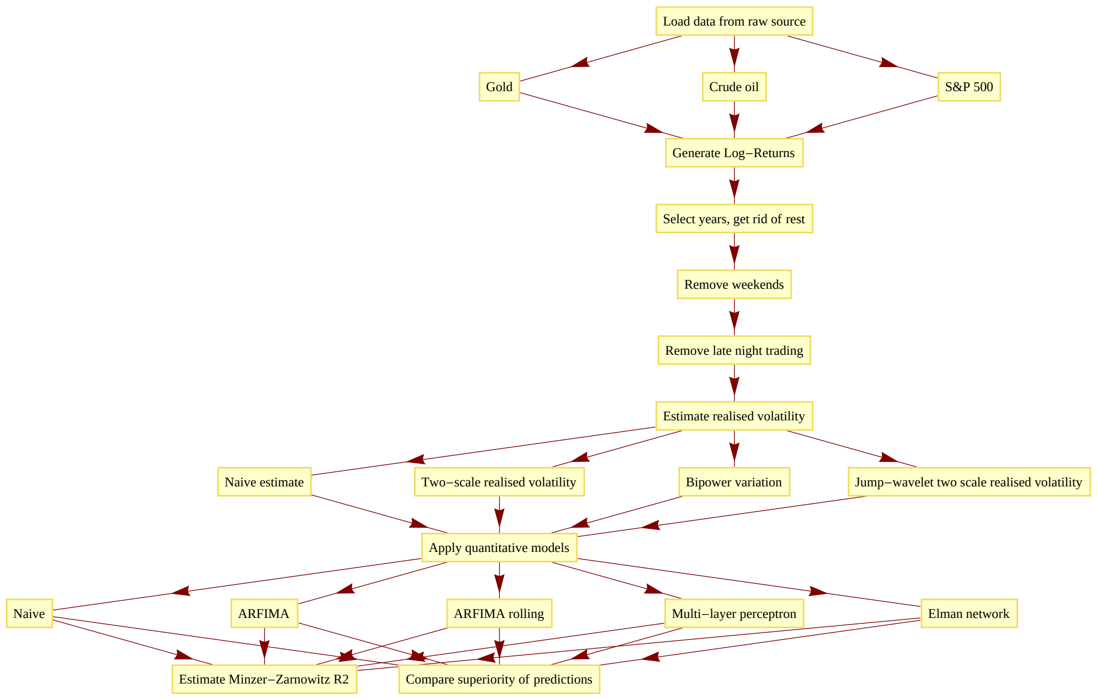
Load data from raw source
Gold
Crude oil
S&P 500
Generate Log−Returns
Select years, get rid of rest
Remove weekends
Remove late night trading
Estimate realised volatility
Naive estimate
Two−scale realised volatility
Bipower variation
Jump−wavelet two scale realised volatility
Apply quantitative models
Naive
ARFIMA
ARFIMA rolling
Multi−layer perceptron
Elman network
Estimate Minzer−Zarnowitz R2
Compare superiority of predictions

Figure A.3: Flowchart description of the program

	Start	End	Days	Observations
Gold	1984-01-03	2012-11-09	7628	3831122
Crude	1987-01-02	2011-12-30	6654	3316592
S&P 500	1983-02-01	2011-12-30	7290	2750092

Table A.1: Summary statistics, original time series

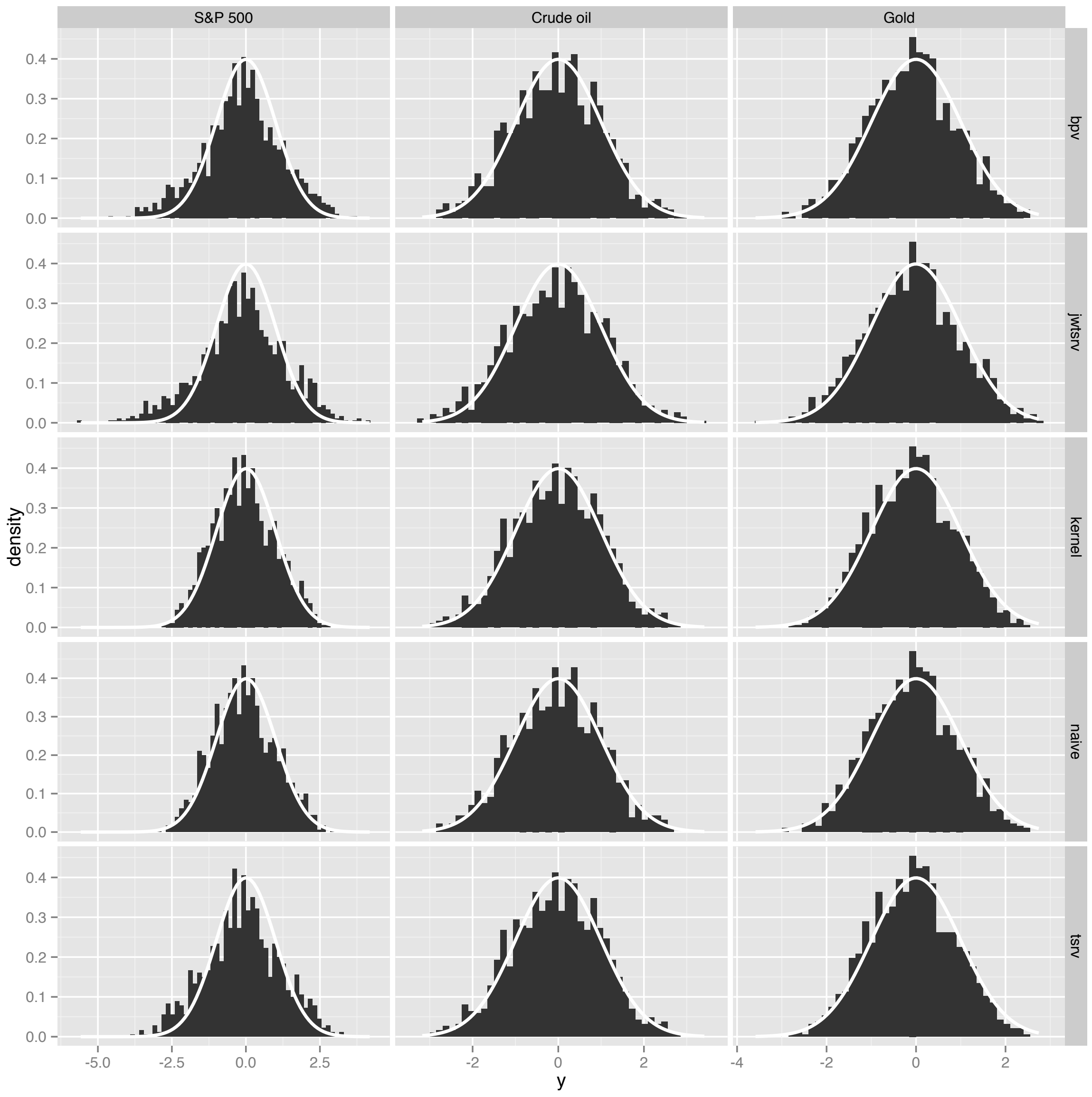

Figure A.4: Histogram of daily log-returns standardized by $RV^{1/2}$, standard normal density superimposed

	Start	End	Days	Observations
Gold	2007-01-02	2011-12-30	1247	1508371
Crude	2007-01-02	2011-12-30	1247	1471701
S&P 500	2007-01-17	2011-12-20	1201	467842

Table A.2: Summary statistics, trimmed time series

	Min	Max	Madian	Average	Kurtosis	Skewness
Gold	-0.02	0.02	0.00	-0.00	36.35	0.37
Crude	-0.04	0.03	0.00	0.00	37.83	0.09
S&P 500	-0.03	0.03	0.00	0.00	78.18	0.48

Table A.3: Summary statistics, log-return series used in estimation

	Shapiro-Wilk	Lilliefors	Anderson-Darling	Pearson chi-square
bpv	0.05	0.30	0.17	0.02
jwtsrv	0.30	0.54	0.26	0.17
kernel	0.04	0.17	0.10	0.04
naive	0.03	0.25	0.13	0.08
tsrv	0.05	0.18	0.10	0.26

Table A.4: P-values of normality tests of Crude oil log-returns standardized by $RV^{1/2}$

	Shapiro-Wilk	Lilliefors	Anderson-Darling	Pearson chi-square
bpv	0.22	0.86	0.60	0.77
jwtsrv	0.41	0.61	0.52	0.43
kernel	0.05	0.31	0.11	0.47
naive	0.16	0.61	0.27	0.66
tsrv	0.06	0.28	0.12	0.64

Table A.5: P-values of normality tests of Gold log-returns standardized by $RV^{1/2}$

	Shapiro-Wilk	Lilliefors	Anderson-Darling	Pearson chi-square
bpv	0.01	0.01	0.00	0.03
jwtsrv	0.00	0.00	0.00	0.00
kernel	0.00	0.10	0.00	0.49
naive	0.00	0.23	0.01	0.54
tsrv	0.03	0.33	0.09	0.08

Table A.6: P-values of normality tests of S&P 500 log-returns standardized by $RV^{1/2}$

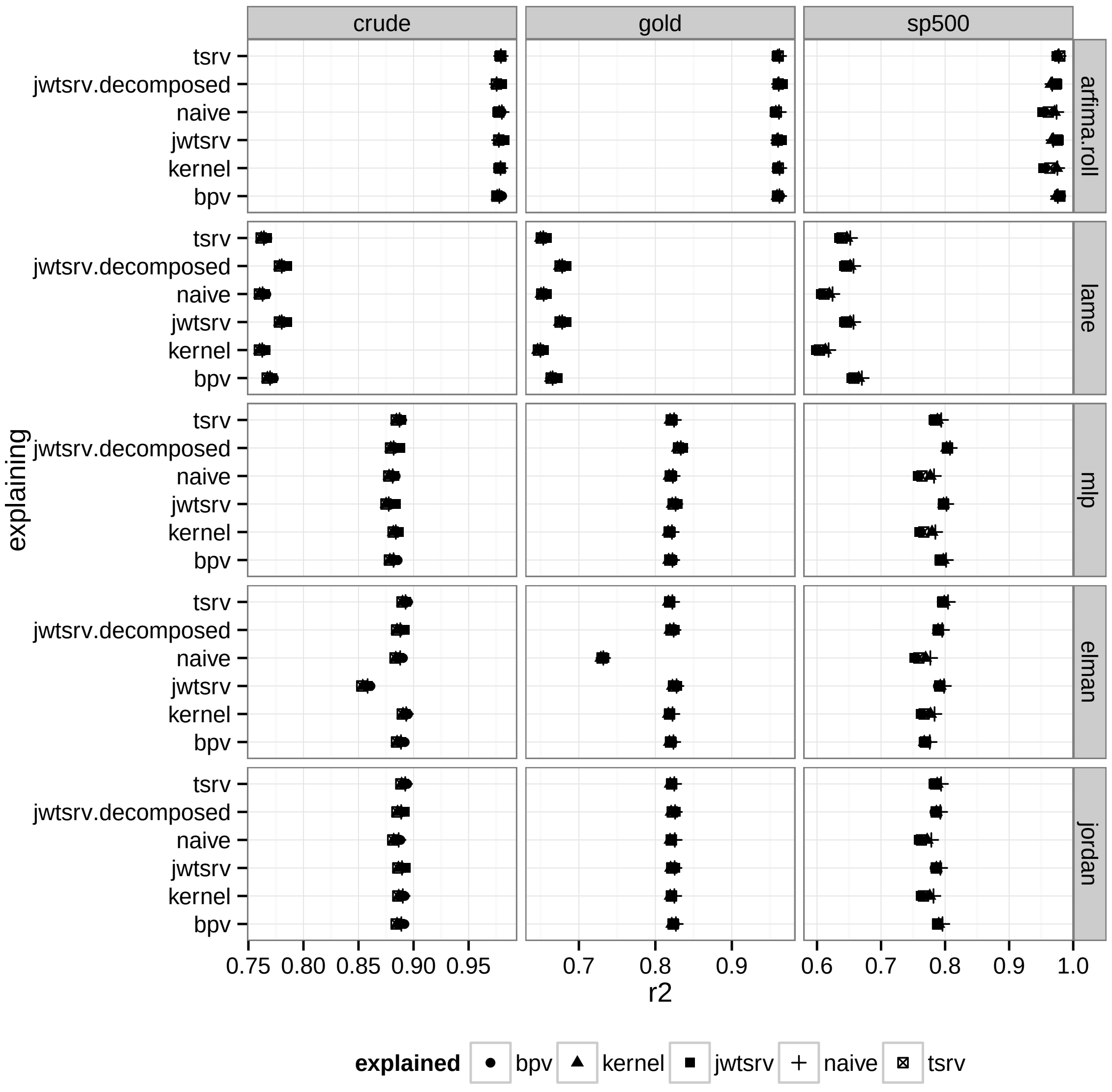

Figure A.5: R^2 from Minzer-Zarnowitz regressions, ten days ahead, in-sample

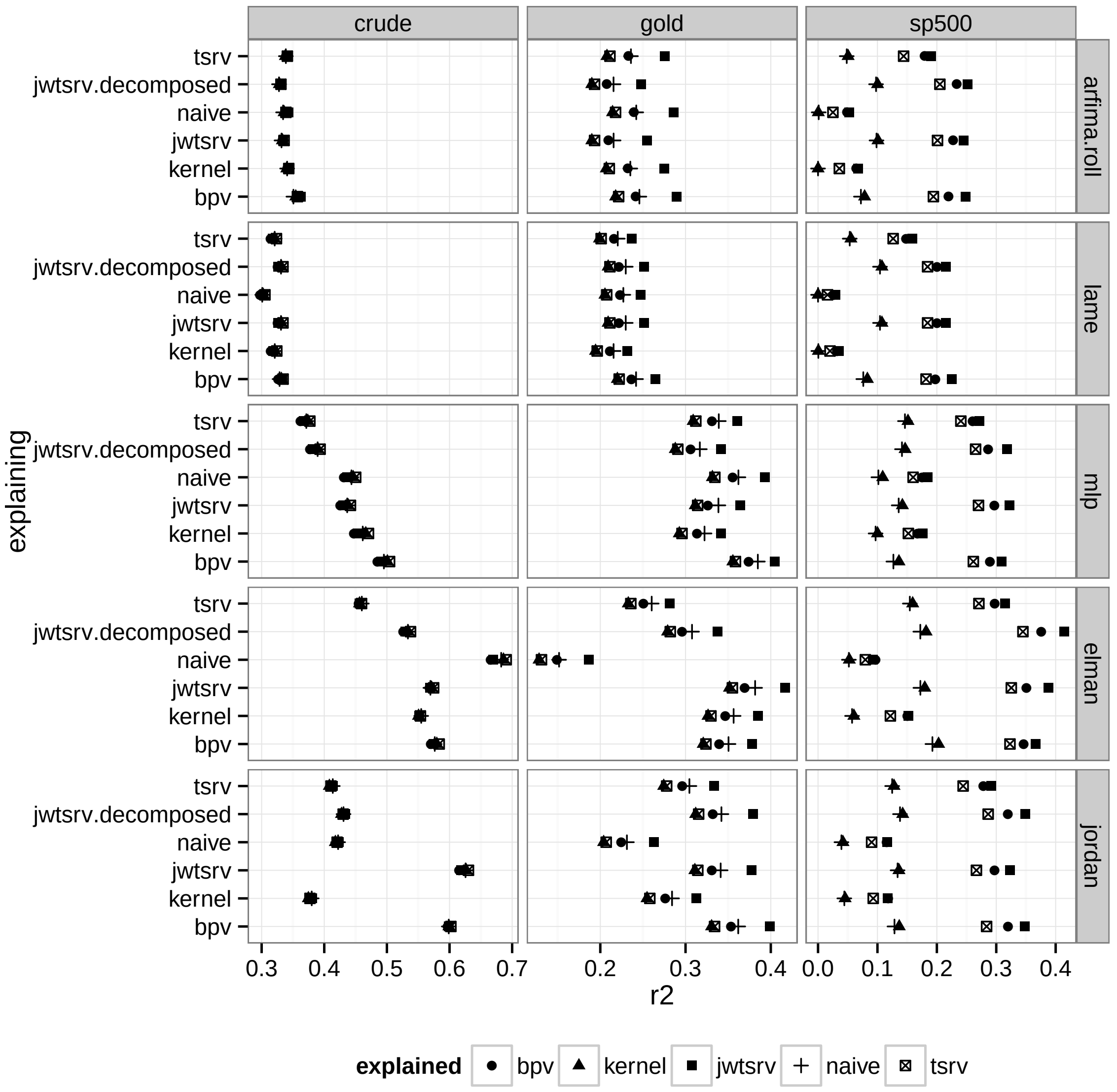

Figure A.6: R^2 from Minzer-Zarnowitz regressions, ten days ahead, out-sample

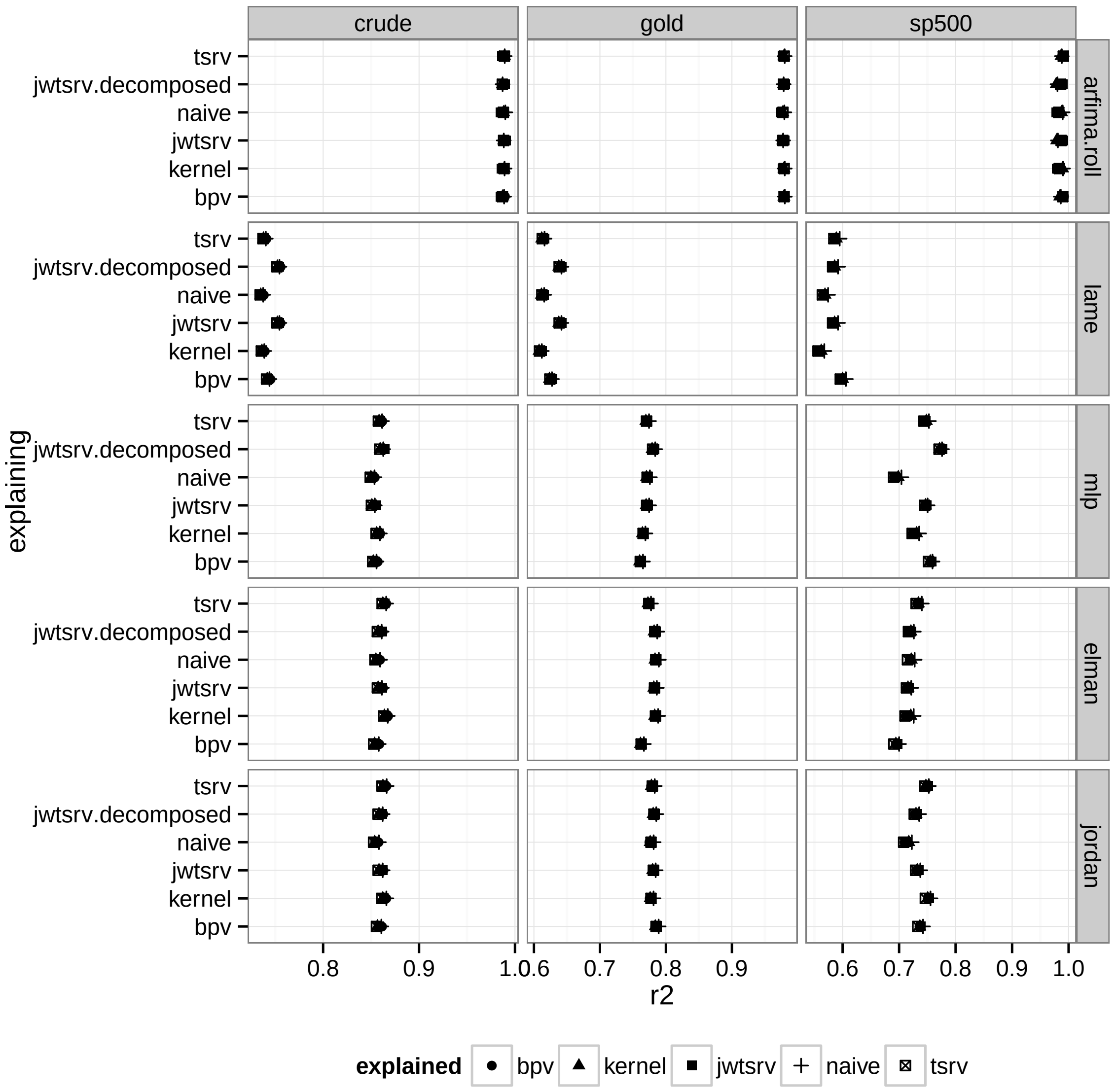

Figure A.7: R^2 from Minzer-Zarnowitz regressions, twenty days ahead, in-sample

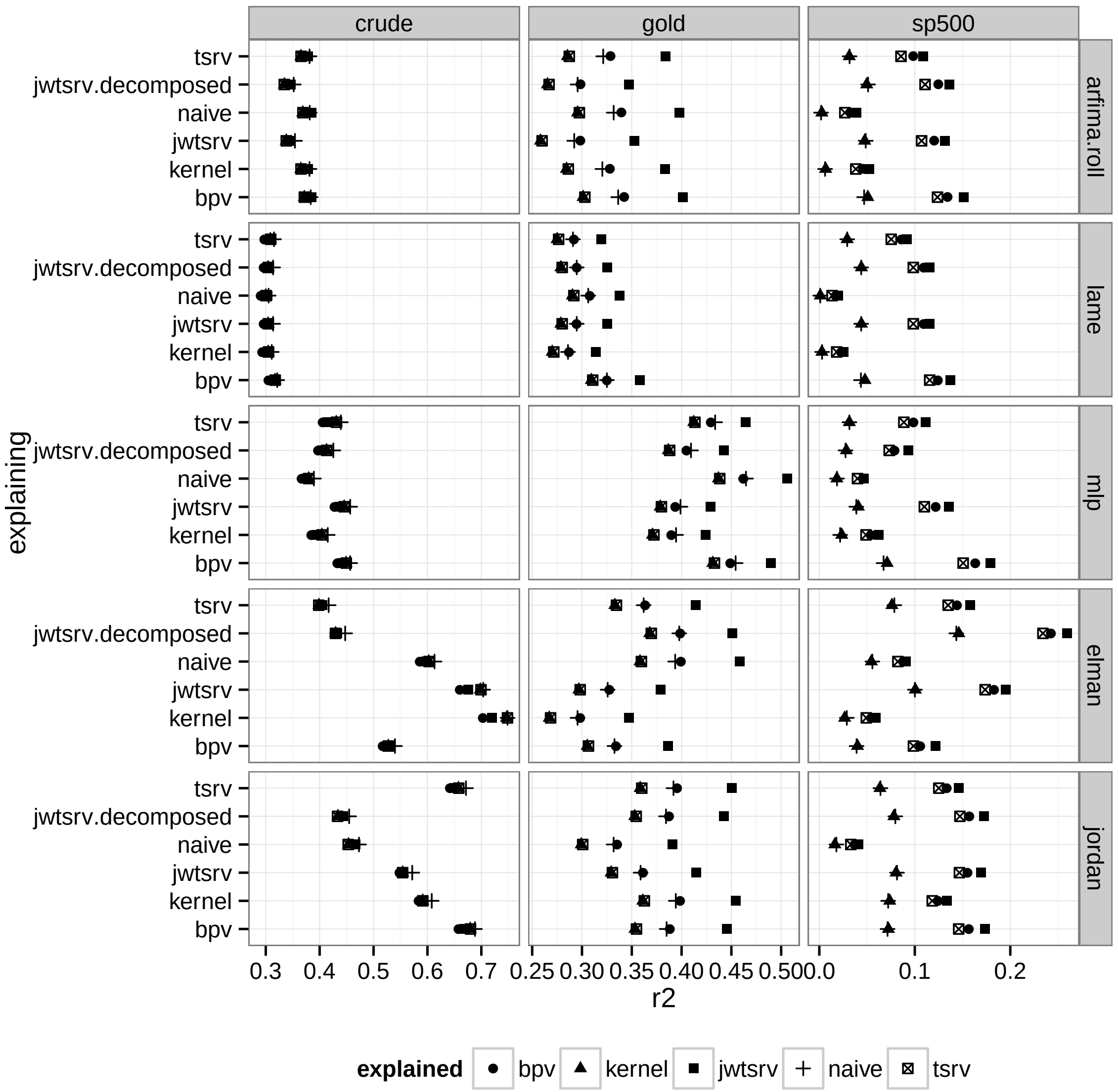

Figure A.8: R^2 from Minzer-Zarnowitz regressions, twenty days ahead, out-sample

Bibliography

Aït-Sahalia, Yacine, and Jean Jacod. 2011. "Testing whether jumps have finite or infinite activity." *The Annals of Statistics* 39 (3): 1689–1719.

Andersen, Torben G, and Luca Benzoni. 2008. "Realized Volatility."

Andersen, Torben G, Tim Bollerslev, Francis X Diebold, and Heiko Ebens. 2001. "The Distribution of Stock Return Volatility." *Journal of Financial Economics* 61 (1): 43–76.

Andersen, Torben G, Tim Bollerslev, Francis X Diebold, and Paul Labys. 2000. "Exchange Rate Returns Standardized by Realized Volatility are (Nearly) Gaussian." 4 (3): 159–179.

———. 2003. "Modeling and forecasting realized volatility." *Econometrica* 71 (2): 579–625.

Andersen, Torben G, Tim Bollerslev, and Xin Huang. 2011. "A reduced form framework for modeling volatility of speculative prices based on realized variation measures." *Journal of Econometrics* 160 (1): 176–189.

Back, Kerry. 1991. "Asset pricing for general processes." *Journal of Mathematical Economics* 20 (4): 371–395.

Baillie, Richard T, and Tim Bollerslev. 1994. "Cointegration, fractional cointegration, and exchange rate dynamics." *The Journal of Finance* 49 (2): 737–745.

Barndorff-Nielsen, Ole E, and Neil Shephard. 2001. "Non-Gaussian Ornstein–Uhlenbeck-Based Models and Some of Their Uses in Financial Economics." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63 (2): 167–241.

———. 2004. "Power and bipower variation with stochastic volatility and jumps." *Journal of Financial Econometrics* 2 (1): 1–37.

Barron, Andrew R. 1994. "Approximation and estimation bounds for artificial neural networks." *Machine Learning* 14 (1): 115–133.

Baruník, Jozef. 2011. "Wavelet-based Realized Variation and Covariation Theory." PhD diss., Charles University in Prague.

Barunik, Jozef, and Lukas Vacha. 2012. "Realized wavelet-based estimation of integrated variance and jumps in the presence of noise." *Available at SSRN.*

Bollerslev, Tim. 1986. "Generalized autoregressive conditional heteroskedasticity." *Journal of econometrics* 31 (3): 307–327.

Corsi, Fulvio. 2009. "A Simple Approximate Long-Memory Model of Realized Volatility." *Journal of Financial Econometrics* 7 (2): 174–196.

Diebold, Francis X, and Roberto S Mariano. 1995. "Comparing Predictive Accuracy." *Journal of Business & Economic Statistics* 20:253–263.

Engle, Robert F. 1982. "Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation." *Econometrica: Journal of the Econometric Society:*987–1007.

Fan, J, and Y Wang. 2007. "Multi-scale jump and volatility analysis for high-frequency financial data." *Journal of the American Statistical Association* 102 (480): 1349–1362.

Gençay, Ramazan, Faruk Selçuk, and Brandon J Whitcher. 2001. *An introduction to wavelets and other filtering methods in finance and economics.* Academic press.

Hansen, Peter Reinhard, Ole Eielsen Barndorff-Nielsen, Neil Shephard, and Asger Lunde. 2008. "Designing Realized Kernels to Measure the ex post Variation of Equity Prices in the Presence of Noise." *Econometrica* 76 (6): 1481–1536. ISSN: 0012-9682. doi:10.3982/ECTA6495.

Haykin, Simon. 2007. *Neural Networks: A Comprehensive Foundation.* Prentice Hall Englewood Cliffs, NJ.

Koopman, Siem Jan, Borus Jungbacker, and Eugenie Hol. 2005. "Forecasting daily variability of the S&P 100 stock index using historical, realised and implied volatility measurements." *Journal of Empirical Finance* 12 (3): 445–475.

Liu, Lily, Andrew J Patton, Kevin Sheppard, and Preliminary Comments Welcome. 2012. *Does anything beat 5-minute rv? a comparison of realized measures across multiple asset classes.* Technical report. Duke University, Working Paper.

McAleer, Michael, and Marcelo C. Medeiros. 2008. "Realized Volatility: A Review." *Econometric Reviews* 27, nos. 1-3 (February): 10–45. ISSN: 0747-4938. doi:10.1080/07474930701853509.

———. 2011. "Forecasting realized volatility with linear and nonlinear univariate models." *Journal of Economic Surveys* 25 (1): 6–18.

McNelis, Paul D. 2005. *Neural networks in finance: gaining predictive edge in the market.* 243. Academic Press.

Meese, Richard A, and Kenneth Rogoff. 1983. “Empirical exchange rate models of the seventies: do they fit out of sample?” *Journal of international economics* 14 (1): 3–24.

Merton, Robert C. 1980. “On estimating the expeted return on the market.” *Journal of Financial Econometrics* 8:323–361.

Mincer, Jacob A, and Victor Zarnowitz. 1969. “The evaluation of economic forecasts.” In *Economic Forecasts and Expectations: Analysis of Forecasting Behavior and Performance,* 1–46. NBER.

O’Hara, Maureen. 1995. *Market microstructure theory.* Blackwell Cambridge, MA.

Poon, Ser-Huang, and Clive WJ Granger. 2003. “Forecasting volatility in financial markets: A review.” *Journal of Economic Literature* 41 (2): 478–539.

Riedmiller, Martin, and Heinrich Braun. 1993. “A direct adaptive method for faster backpropagation learning: The RPROP algorithm.” In *Neural Networks, 1993., IEEE International Conference on,* 586–591. IEEE.

Sowell, Fallaw. 1992. “Modeling long-run behavior with the fractional ARIMA model.” *Journal of Monetary Economics* 29 (2): 277–302.

Werbos, Paul J. 1974. “Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.” PhD diss., Harvard University.

Zhang, Lan, Per A Mykland, and Yacine Ait-Sahalia. 2005. “A tale of two time scales: Determining integrated volatility with noisy high-frequency data.” *Journal of the American Statistical Association* 100 (472): 1394–1411.

R packages

Aldrich, Eric. 2012. *wavelets: A package of functions for computing wavelet filters, wavelet transforms and multiresolution analyses.* R package version 0.2-9. `http://CRAN.R-project.org/package=wavelets`.

Bergmeir, Christoph, and José M. Benítez. 2012. "Neural Networks in R Using the Stuttgart Neural Network Simulator: RSNNS." *Journal of Statistical Software* 46 (7): 1–26. `http://www.jstatsoft.org/v46/i07/`.

Dowle, M, T Short, and S Lianoglou. 2013. *data.table: Extension of data.frame for fast indexing, fast ordered joins, fast assignment, fast grouping and list columns.* R package version 1.8.8. `http://CRAN.R-project.org/package=data.table`.

Eddelbuettel, Dirk. 2013. *Seamless R and C++ Integration with Rcpp.* ISBN 978-1-4614-6867-7. New York: Springer.

Eddelbuettel, Dirk, and Romain François. 2011. "Rcpp: Seamless R and C++ Integration." *Journal of Statistical Software* 40 (8): 1–18. `http://www.jstatsoft.org/v40/i08/`.

George Athanasopoulos, Rob J Hyndman with contributions from, Slava Razbash, Drew Schmidt, Zhenyu Zhou, and Yousaf Khan. 2013. *forecast: Forecasting functions for time series and linear models.* R package version 4.04. `http://CRAN.R-project.org/package=forecast`.

Ghalanos, Alexios. 2013. *rugarch: Univariate GARCH models.* R package version 1.02.

R Core Team. 2013. *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. `http://www.R-project.org/`.

Wickham, Hadley. 2009. *ggplot2: elegant graphics for data analysis.* Springer New York. ISBN: 978-0-387-98140-6. `http://had.co.nz/ggplot2/book`.

———. 2012. *stringr: Make it easier to work with strings.* R package version 0.6.2. `http://CRAN.R-project.org/package=stringr`.

———. 2007. "Reshaping data with the reshape package." *Journal of Statistical Software* 21 (12). `http://www.jstatsoft.org/v21/i12/paper`.