FUNDAÇÃO GETULIO VARGAS
ESCOLA DE ECONOMIA DE SÃO PAULO

BRUNO MORIER

MODELING HIGH FREQUENCY INTRADAY DISCRETE RETURNS

SÃO PAULO
2020

BRUNO MORIER

MODELING HIGH FREQUENCY INTRADAY DISCRETE RETURNS

Tese apresentada à Escola de Economia de São Paulo da Fundação Getulio Vargas como requisito para obtenção do título de Doutor em Economia de Empresas

Campo de Conhecimento:
Finanças

Orientador: Pedro L. Valls Pereira

São Paulo
2020

Morier, Bruno do Nascimento.

Modeling high frequency intraday discrete returns / Bruno do Nascimento Morier. - 2020.

105 f.

Orientador: Pedro L. Valls Pereira.

Tese (doutorado CDEE) – Fundação Getulio Vargas, Escola de Economia de São Paulo.

1. Análise de séries temporais. 2. Volatilidade (Finanças). 3. Aprendizado do computador. 4. Redes neurais (Computação). 5. Ações (Finanças) - Preços - Previsão. I. Pereira, Pedro L. Valls. II. Tese (doutorado) – Escola de Economia de São Paulo. III. Fundação Getulio Vargas. IV. Título.

CDU 519.246.8

Ficha Catalográfica elaborada por: Isabele Oliveira dos Santos Garcia CRB SP-010191/O
Biblioteca Karl A. Boedecker da Fundação Getulio Vargas - SP

BRUNO MORIER

MODELING HIGH FREQUENCY INTRADAY DISCRETE RETURNS

Tese apresentada à Escola de Economia de São Paulo da Fundação Getulio Vargas como requisito para obtenção do título de Doutor em Economia de Empresas.

Campo de Conhecimento:
Finanças

Data de Aprovação:
__/__/_____

Banca examinadora:

Prof. Dr. Pedro L. Valls Pereira
FGV-EESP

Prof. Dr. Marcelo Fernandes
FGV-EESP

Prof. Dr. Luiz Koodi Hotta
Unicamp

Prof. Dr. Marcelo Medeiros
PUC-RJ

Prof. Dr. Christian Zimmer
RoboBanker

AGRADECIMENTOS

Ao Pedro Valls, meu orientador, curso de doutorado. Pelos valiosos ensinamentos ao longo dos diversos cursos que fiz ao durante curso de doutorado e que moldaram os primeiros artigos desta tese e ampliaram meus horizontes em econometria. Pela sua paciência e compreensão no processo dinâmico da tese, em especial no meu momento profissional atual que envolve diversas responsabilidades distintas. Sou tambem especialmente grato pela ajuda que me prestou quando outras oportunidades acadêmicas surgiram, embora não tenhamos seguido por este caminho.

Ao Vladimir Teles, meu orientador no mestrado na EESP, sem o qual não teria a coragem de enfrentar o desafio do doutorado em economia. Me ajudou não apenas no plano para o doutorado, mas tambem enriqueceu de forma importante minha formação em macroeconomia. Foi sempre solicito nos casos onde precisei de alguma ajuda.

Aos demais professores da EESP-FGV pela excepcional formação que obtive no curso de doutorado. Aos amigos do doutorado, os quais tornaram a jornada muito mais leve e descontraída. E não poderia deixar de mencionar à equipe do CMCD, sempre solícita e eficiente em todas as questões para as precisei de auxílio.

À Tatiana Grecco, ao Eduardo Camara Lopes(X) e ao Ricardo Negreiros, meus supervisores no Banco Itaú e posteriormente no Banco Safra, por possibilitarem que eu cursasse o programa de doutorado ao mesmo tempo em que desempenhava funções como pesquisador, gestor de recursos e gestor de pessoas em ambas as instituições. Sem a compreensão destes três diretores jamais teria conseguido equilibrar minhas tarefas academicas e profissionais.

À minha esposa, Viviani, pela paciência e esforço em manter tudo funcionando enquanto eu me dedicava a minha jornada dupla, que durou muitos anos. Sem o seu equilíbrio e especial atenção os nossos filhos não seria possível concluir este curso.

Por fim e não menos importante, aos meus filhos João Pedro e Lara. Pela conpreensão de ambos em relação a rotina de estudos que o doutorado me impôs. Espero que um dia consigam ler e entender o que está neste documento. Em breve teremos mais tempo juntos.

ABSTRACT

This thesis encompasses three papers on the subject of modelling high-frequency intraday discrete price changes. In all papers we consider the task of modelling the discrete conditional distribution of price changes, propose new models and conduct large scale estimation and forecasting exercises in order to compare the models with existing models in the literature. In the first paper we extend the univariate non-linear non-Gaussian state space model of Koopman, Lit and Lucas (2017) by including a specification for the conditional mean. In the second paper we propose a new model for the bi-variate conditional distribution by using Gaussian copulas and by modelling the correlation coefficient dynamics by using a non-linear, non-Gaussian state space model. On the last paper we propose a new model for the univariate conditional distribution where the conditional volatility is predicted by a deep feedforward neural network. We also incorporate three new variables for predicting the discrete price high-frequency volatility: the bid-ask spread, high-low interval spread and the volume traded. In all the three papers the new models outperformed recent literature models considered at the conditional density forecasting exercises conducted.

Keywords: volatility models; high frequency data; discrete price changes; importance sampling; deep learning; neural networks; score driven models; Skellam; non-Gaussian time series models; time-varying copulas, dynamic discrete data, score driven models, NAIS

JEL Classification: C32, C45, G11

RESUMO

Esta tese inclui três artigos sobre o tópico de modelagem de retornos intradiários discretos em alta-frequencia. Em todos os artigos nós conduzimos a tarefa de modelar a distribuição condicional discreta das mudanças de preço, propomos novos modelos de previsão e conduzimos exercícios de estimação e previsão em larga escala para comparar os novos modelos com os modelos existentes na literatura. No primeiro artigo nós estendemos o modelo de espaço de estados univariado, não-linear e não-gaussiano de Koopman, Lit and Lucas (2017) incluindo uma especificação para a média condicional. No segundo artigo nós propomos um novo modelo para a distribuição condicinal bivariada usando cópulas gaussianas dinâmicas e modelando o coeficiente de correlação com um modelo em espaço de estados não-linear e não-gaussiano. No último artigo nós propomos um novo modelo para a distribuição univariada condicional onde a volatilidate condicional é prevista por uma rede neural feedforward. Nós tambem incorporamos três novas variaveis para o modelos de previsão de volatilidade em alta frequência com preços discretos: o spread de compra e venda, o spread entre o preço maximo e preço mínimo e o volume transacionado. Em todos os três artigos os novos modelos mostraram melhor performance nos exercícios de previsão de densidade condicional quando comparados a modelos recentes da literatura.

Palavras-chaves: modelos de volatilidade; dados em alta frequência; retornos discretos; importance sampling; aprendizagem de máquina; redes neurais; modelos de score; Skellam; modelos de série de tempo não gaussianos; NAIS

Classificação JEL: C32, C45, G11

List of Figures

List of Tables

Table of Contents

Introduction

Volatility research in financial time series traditionally have been done by analyzing daily returns datasets. Models like Garch and Stochastic Volatility have been studied for decades in the literature and are popular among practitioners. They are useful for market participants, regulators and exchanges for measuring the risks of financial investments. See Andersen et al. (2006) for a comprehensive survey.

The availability of intraday high-frequency data have moved the literature focus towards the usage of intraday returns for predicting daily volatility. Volatility in asset returns is unobservable and time varying, so that including more timely data in the models (e.g. intraday returns) can increase volatility prediction accuracy. Most of the papers in this literature use non-parametric methods for volatility inference. Papers like Barndorff-Nielsen and Shephard (2002), Andersen et al. (2001) and Hansen and Lunde (2006) follow this approach.

More recently there have been an increased interest in modeling the underlying intraday stochastic process in order to obtain a better description of the intraday return dynamics. The interest in this case is modelling and predicting the intraday volatility and correlation using the intraday returns - in contrast to the existing literature that focused in forecasting daily volatility and correlation. For instance, Shephard and Yang (2016) develop continuous-time stochastic Levy processes and Koopman, Lit and Lucas (2017) estimate a stochastic volatility model using intraday high-frequency data. This thesis follow this recent trend and focus in analyzing high frequency intraday returns to forecast conditional intraday return distribution with emphasis on conditional volatility and correlation.

In the first paper we model the univariate intraday price dynamics by using non-linear non-Gaussian state space models, following an approach close to Koopman, Lit and Lucas (2017). Two different specifications are analyzed. Different from the previous literature, we consider estimation over multiple days and model the mean of the intraday distribution, by using an auto-regressive specification. Comparing the forecasting performance we conclude that all specifications provide forecasts that outperform empirical non-parametric predictors. We also conclude that including the autoregressive specification for the mean enhances the forecasting performance for the log-likelihood out-of-sample prediction loss.

In the second paper we model the bivariate intraday price dynamics by using two different kind of models: State-Space models and Score Driven models. In both cases we consider the case of Gaussian copulas for the conditional distribution. Both types

of models outperform empirical predictors for the bivariate conditional distribution. We also conclude that the newly proposed State Space model have a statistically significant superior forecasting performance for all of our sample.

In the third paper we model the univariate intraday price dynamics by using deep learning. Specifically, we train feedforward neural networks in order to generate predictions for the underlying high frequency volatility. This approach is novel in the literature to the best of the authors knowledge. Four different specifications are tested including different set of features and parameters. All models beat empirical non-parametric forecasting rules considered. Our forecasting procedure for the univariate also provides better out-of-sample forecasts compared to all Space State Models considered in the thesis. We also conclude that new variables have predictive power for the volatility process: the bid-ask spread, high-low interval spread and the volume traded.

The amount of data considered for all papers is huge (more than 2.3 billion prices) and the estimation procedures are computationally expensive. The estimation procedures on this thesis need efficient computational implementations in order to be feasible. The Appendix depicts the computational techniques and machines used for computing the models for this work. They included the use of distributed processing, GPU processing, analytic approximating functions and specific compiled code. We have also used a cluster at Amazon AWS EC2 service for part of the computations.

1 Modeling High Frequency Intraday Returns by Non-Linear State Space Models

Abstract

In this paper we propose a model for discrete intraday high-frequency returns based on non-linear, non-Gaussian state space models, following an approach close to Koopman, Lit and Lucas (2017). Two different specifications are analyzed. Different from the previous literature, we consider estimation over multiple days and model the mean of the intraday distribution by using an auto-regressive specification. We also develop a novel estimation procedure based on the coordinate descent method that minimizes the number of Numerically Accelerated Importance Sampling (NAIS) procedure computations during the optimization. Lastly, we conduct an extensive walk-forward forecasting exercise. Comparing the forecasting performance of all models, we conclude that all state space specifications considered provide forecasts that outperform empirical non-parametric predictors. We also conclude that including the specification for the mean enhances the forecasting performance for the log-likelihood out-of-sample prediction loss.

Keywords: discrete price changes; high frequency data; importance sampling; NAIS; non-Gaussian time series models Skellam; volatility models;

JEL Classification: C32, G11

1.1 Introduction

Understanding and predicting asset return's distribution is important for market participants, regulators and exchanges. Modeling the return's distribution is one of the main research goals in the financial literature. The second moment, volatility, is a primary measure of risk in financial markets and predicting potential losses by using daily volatility has been standard since the 90's. Modeling the first moment is usually pursued by quantitative asset managers.

In recent times, the interest in modeling returns has expanded also to intraday time frame. The ongoing technological advances in financial markets have popularized electronic trading, increasing order management speed and making High Frequency Trading (HFT) possible. Market reactions became faster, including extreme ones, leading to the interest in measuring intraday risk. One extreme example of interest is the 2010 S&P futures market flash crash (see Easley, Prado and O'Hara (2011) and Kirilenko et al. (2017)). The availability of intraday prices have also increased over time and there are now many studies using these prices to forecast daily volatilities and, more recently, to forecast the intraday dynamic itself. See Barndorff-Nielsen and Shephard (2002), Andersen et al. (2001) and Hansen and Lunde (2006) for papers forecasting daily volatilities and refer to Engle and Sokalska (2012) and Andersen, Bollerslev and Lange (1999).

Volatility is unobservable, changes through time and possesses a clustering pattern. All these characteristics also make volatility an interesting target for state space models in finance and in fact this is one of the main approaches for modeling its time dependence. There are many studies considering state space for modelling daily volatility through state space models and also a few papers considering these models for the intraday price changes. See Harvey and Shephard (1996) and Harvey (2007) for daily volatility and Bekierman and Gribisch (2016) for a mixed frequency approach that includes a intraday component. These papers rely on Non-Linear Gaussian State Space models.

But these model are not suitable for modelling the High Frequency Price dynamics, as they rely on a continuous (Gaussian) distribution. Koopman, Lit and Lucas (2017) proposes modelling the underlying price change distribution through a discrete (modified) Skellam distribution. We follow a similar approach in this paper, but we also model the process conditional first moment. It is commonly assumed in finance literature that the expected values of short term returns do not depend on past prices, but this ceases to be the case for high frequency returns, which are affected by the market's microstructure. So we are considering a discrete price, non-linear, non-Gaussian state space model for forecasting the conditional probability distribution for financial assets' returns, including its first and

second moments.

Our dataset consists of trading prices for selected equities for the entire year of 2018, obtained from Thomson Reuters Datascope service. We test our models by using four liquid Brazilian stock. We consider the time frames of 10 seconds, considering only time frames where trades happen. In this regard we follow Koopman et al. (2018) instead of Koopman, Lit and Lucas (2017). This is because we use the same models for the univariate case and for the marginals of the bivariate case in paper 2. As it is common in this literature, we only consider intraday price changes disconsidering the overnight gap.

We estimate the model parameters by maximum likelihood estimation. The log-likelihood function is intractable, so we rely in simulation methods for sampling it. We use the Numerically Accelerated Importance Sampling introduced in Koopman, Lucas and Scharth (2015). This approach is close to he Efficcient Importance Sampling that can be found in Liesenfeld and Richard (2003) and Richard and Zhang (2007). For the forecasting procedure we use the Bootstrap Particle Filter.

On the empirical section we conduct a forecasting exercise, comparing the forecasting performance for different models specifications. We also compare the models performance with empirical non-parametric predictors. In order to make all the computations we parallelized the computation through several GPUs, setting up a cluster on Amazon AWS EC2. The details of the computational aspects are described in the Appendix for the thesis.

The rest of this paper is organized as follows. Section 2 contains the data description, specification and treatment description. Sections 3 and 4 describe the analytical model. Sections 5 and 6 provide details on the numerical estimation procedure. Sections 7 and 8 contain the empirical forecasting exercise along with the results from the estimation procedure. Section 9 concludes.

1.2 Data

Our dataset is formed by intraday prices for four Brazilian Stocks: Petrobras (PETR4), Vale (VALE3), Itau-Unibanco (ITUB4) and Bradesco (BBDC4) for the entire year of 2018. These stocks are among the most liquid Brazilian stocks. The data used in this paper consists of the closing trading prices for the intraday interval of 10 seconds obtained from B3 exchange by using Thomson Reuters Datascope service. Taking all four stocks together, our dataset consists of more than 2.3 billion prices.

In the literature we have studies with approaches similar to this paper that uses both 1 second (Koopman, Lit and Lucas (2017)) and 10 seconds time interval (Koopman et al. (2018)). The first one is concerned with the volatility dynamics, analyzing univariate time series while the second makes a multivariate analysis. The 10 seconds time frame is more convenient for the multivariate analysis as it raises the probability of the joint event of simultaneous trading for the assets considered in the analysis, as argued in Koopman et al. (2018). A reason for using 10 seconds interval even in the univariate case is that although we chose the most liquid Brazilian stocks, these stocks are less liquid than the American counterparts used in Koopman, Lit and Lucas (2017). For these two reasons we chose the 10 seconds interval for the three papers in this thesis. There is a last advantage in this choice: it makes the whole thesis comparable and based in the same dataset, which is described in this section.

It is worth noting that even using the 10 seconds time frame (instead of the 1 second) we still have to deal with a lot of missing values, as the stocks do not necessarily trade every 10 seconds interval. Regarding this issue we also take the same approach of Koopman et al. (2018), by only updating the model when trading activity occurs. So we do not pad missing prices as done in Koopman, Lit and Lucas (2017). Repeating prices when trading activity do not occur has the effect of equating the event of no trade in a time t to the (different) event of a trade happening in time t with the same price as $t-1$, which is not desirable. This procedure would inflate the number of zero price changes.

We consider all prices ranging from the market opening to the market closing. We model the intraday price changes, so we discard the overnight price changes in the cases where we estimate the model through more than one day. We apply a data-cleaning procedure before the analysis, in order to clean for exchange reporting errors, as recommended by Brownlees and Gallo (2006).

In table 1 we show descriptive statistics for the entire sample. We notice that there is high occurrence of no change in price (0's) and changes of magnitude 1 (± 1). The distribution for all stocks is also fat tailed, as the minimum and maximum price changes are more than 30 times the standard deviation. This last fact can also be checked by looking at the 0.1% and 99.9% percentiles that are more than 5 times the standard deviation. These occurrences shows not only a fatter than normal distribution, but also fatter than the Skellam or Modified Skellam distributions.

In figure 1 we plot the histograms for the price changes for the selected stocks in the entire year of 2018. In order to make this plot we discard the most severe price changes, that is, we only consider the percentiles 0.1% to 99.9%, as the minimum and maximum values

Table 1 – Descriptive Statistics for the 2018 Sample

	# prices	%0	%±1	std	min	max	0.1%	99.9%
Itau-Unibanco	585,142	33	36	1.90	-58	38	-9	9
Bradesco	583,730	36	40	1.57	-63	35	-8	7
Petrobras	587,000	43	43	1.24	-33	26	-6	5
Vale	567,534	31	36	2.09	-49	53	-11	11

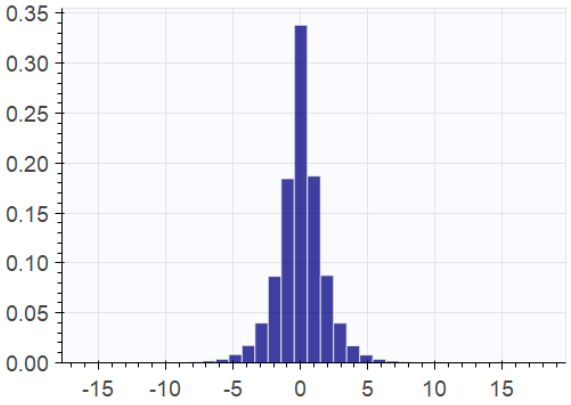
We report prices as the number of 10 seconds closing prices considered in the sample, %0 as the percentage of 0 price changes, $\% \pm 1$ as the percentage of $\% \pm 1$ price changes, std as the standard deviation, min as the minimum price change in ticks, max as the maximum price change in ticks, 0.1% as the 0.1% percentile of price changes in ticks and 99.9% as the 99.9% percentile of price changes in ticks the data, including whatever notes are needed.

exceed the range of the graphs. We can see from this plots that the price changes takes few values most of the time emphasizing the need for a discrete distribution for this data.

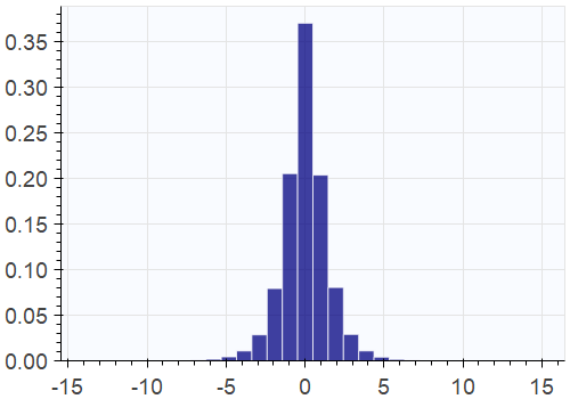
We also note from figure 1 that the least volatile stock in terms of tick volatility, Petrobras, attains less values most of the time for its price changes, while compared with the others. This does not mean that Petrobras stock is less volatile than the others considering usual price log returns . In fact, Petrobras stock is more volatile than the other in this sample. The histogram dispersion for each stock is actually determined by the stock volatility divided by the ratio of the tick size to its price. And for our sample this ratio is higher for Petrobras than for the others. This fact forces the price changes to attain less distinct values most of the time.

Figure 1 – Histogram for price changes in 2018

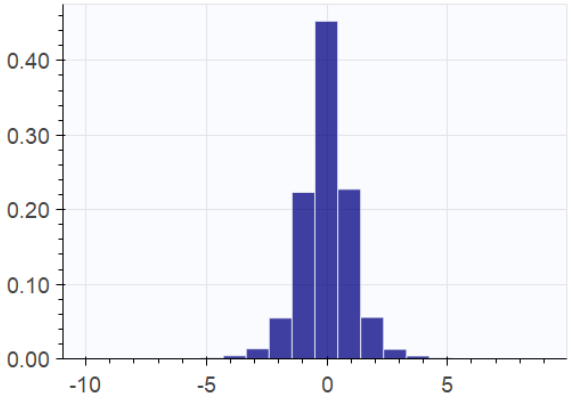
(a) Itau-Unibanco



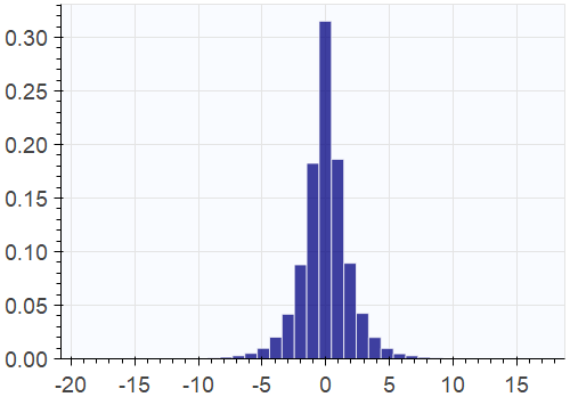
(b) Bradesco



(c) Petrobras



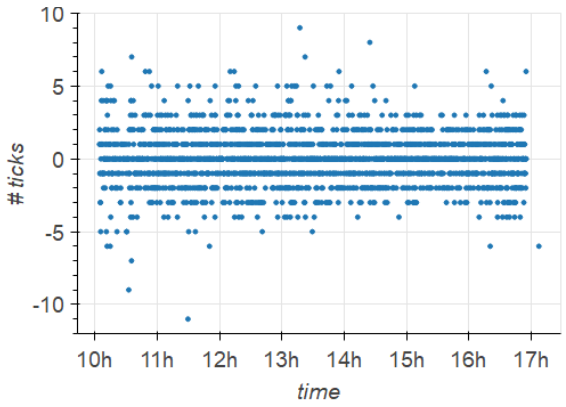
(d) Vale



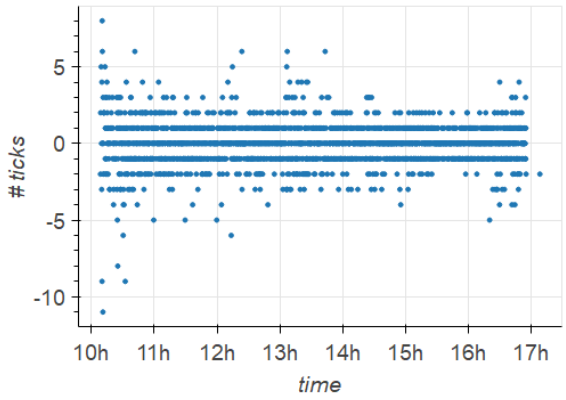
In figure 2 we plot the price changes for the selected stocks in number of ticks for a selected date (June 20, 2018). We can see again that Petrobras price changes attain less distinct values than the other stocks.

Figure 2 – Price changes for assets on June 20, 2018

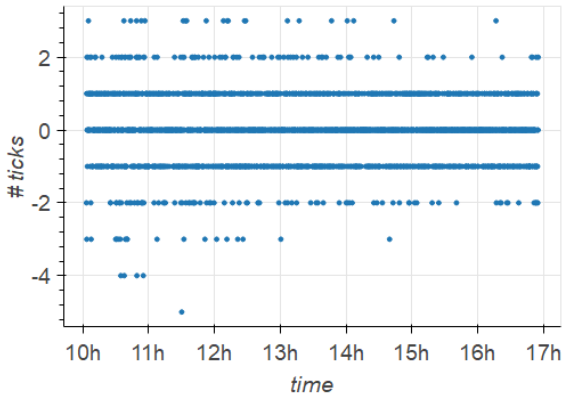
(a) Itau-Unibanco



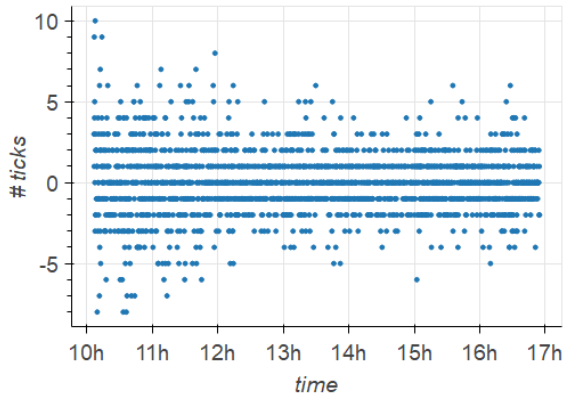
(b) Bradesco



(c) Petrobras



(d) Vale



1.3 Modeling Univariate High Frequency returns

Daily financial market returns are usually modelled by continuous distributions, often by Gaussian distribution. It is common to use continuous distributions even in the case of intraday returns, when the frequency is not high enough (e.g. 30 minutes). This is an useful approximation as market prices and returns are discrete, but at these time frames the wide range of distinct values frequently attained by the underlying market process allows such simplification.

It is clear however from the descriptive analysis conducted in the previous section that discreetness is an important feature at the 10 seconds time frame. The histograms for the price changes show that only a few distinct values are attained most of the time. So the Gaussian distribution is not suitable for modeling this high-frequency data, as any other continuous distribution. So we need another workhorse distribution in order to model our price changes/returns.

Previous works, like Shahtahmassebi (2011) and Shahtahmassebi and Moyeed (2014), have considered the usage of the Skellam distribution for modelling the price change distribution at this time frame. The Skellam distribution was originally derived as the difference between two Poisson distributions. In Koopman, Lit and Lucas (2017) the authors consider a reparameterized version of the Skellam distribution, dependent on the parameters μ and σ, following Alzaid(2010). We follow the same approach. Let $Y_t \in \mathbb{Z}$ be a Skellam distributed variable with $\mathbb{E}(Y_t) = \mu \in \mathbb{R}$ and $\text{Var}(Y_t) = \sigma^2 \in \mathbb{R}^+$ and let $y_t \in \mathbb{Z}$ be a time-indexed realization for Y_t for $t \in 1 \dots T$. It's probability mass function (pmf) is given by $Pr(Y_t = y_t) = p_s(y_t; \mu, \sigma)$:

$$p_s(y_t; \mu, \sigma^2) := \exp(-\sigma^2) \left(\frac{\sigma^2 + \mu}{\sigma^2 - \mu} \right) I_{|y_t|}(\sqrt{\sigma^4 - \mu^2}), \tag{1.1}$$

where $I_{|y_t|}(.)$ is the modified I Bessel function of order $|y_t|$. Comparing with the two original Poisson distributions we have $\mu = \lambda_1 - \lambda_2$ and $\sigma = \lambda_1 + \lambda_2$. The parameter μ not only changes the expected value of Y_t but also its skew. The condition $\mu > 0$ implies a right-skewed distribution while $\mu < 0$ implies a left skewed distribution. In the case where $\mu = 0$, equation (1.1) simplifies to

$$p_s(y_t; \mu, \sigma^2) := \exp(-\sigma^2) I_{|y_t|}(\sigma^2). \tag{1.2}$$

Although the Skellam distribution is a good place to start, it does have some rigidity regarding the shape of its 0 peak. In order to accomodate more flexible patterns in empirical data, both Koopman, Lit and Lucas (2017) and Karlis and Ntzoufras (2008) consider a more flexible version, the Modified Skellam Distribution. In this case a new parameter, γ, is used to transfer mass probability between the peak and its surroundings. The general form of the Modified Skellam distribution transfers mass from $Pr(Y_t = k)$ to $Pr(Y_t = i)$ and $Pr(Y_t = j)$ if $\gamma < 0$, or transfer in the opposite direction if $\gamma > 0$. The Modified Skellam distribution MSKII$(i, j, k, \mu, \sigma^2, \gamma)$ pmf is given by

$$p_{II}(y_t; i, j, k, \mu, \sigma^2, \gamma) := \begin{cases} p_s(y_t; \mu, \sigma^2) & \text{for } y_t \notin \{i, j, k\} \\ p_s(y_t; \mu, \sigma^2) - \gamma\Delta/2 & \text{for } y_t \in \{i, j\} \\ p_s(y_t; \mu, \sigma^2) + \gamma\Delta & \text{for } y_t = k, \end{cases} \tag{1.3}$$

where γ satisfies $2\min(p_s(i, \mu, \sigma^2), p_s(j, \mu, \sigma^2)) > \gamma\Delta > -p_s(k, \mu, \sigma^2)$.

Our case of interest for this paper is given by $k = 0$, $i = -1$ and $j = 1$, in order to transfer between the peak at zero and its surroundings. We also allow the mean and the variance to vary through time. So we simply write the our pmf as

$$p(y_t; \mu_t, \sigma_t^2) := p_{II}(y_t; -1, 1, 0, \mu_t, \sigma_t^2, \gamma).$$

Note that the transformation done at the Modified Skellam Distribution is close to the one done at mixtures. Specifically, the transformation is close to a mixture between the deterministic distribution centered at zero and the Skellam distribution.

A novelty in this paper compared to the existing literature is that we also model the parameter μ, that was previously only considered fixed at zero. We consider two distinct versions of our baseline models with different equations for μ:

1. State Space Model (SS):
$$\mu = 0$$
2. State Space Model with mean (SSM) with
$$\mu = \delta y_{t-1}.$$

The equation for the SSM model suggests a time dependency in the returns similar to an auto-regressive model. To the best of the authors knowledge this structure for the mean is novel for high frequency modeling using the Skellam Distribution and stochastic volatility. In previous works, like Koopman, Lit and Lucas (2017), the mean is assumed to be identical to zero like in the SS model.

1.4 Modelling the Volatility Process

We consider a volatility process close to the one described at Koopman, Lit and Lucas (2017). The serial dependence in $y_1, \cdots, y_n$ second moments is modeled through the parameter σ_t, which follows a dynamic stochastic process. So the model is given by

$$Y_t | \sigma_t^2 \sim p(y_t; \mu_t, \sigma_t^2) \tag{1.4}$$

And there are no other dynamics in Y_t other than the one implied by this structure (i.e parameters μ_t and σ_t).

We model the stochastic process of σ_t by a long term expectation ($\overline{\sigma}$), a seasonality pattern (s_t) and a stochastic component (α_t). Thus σ_t is given by

$$\begin{aligned} \sigma_t^2 &= \overline{\sigma}^2 s_t \exp(\alpha_t) \\ \alpha_{t+1} &= \phi\alpha_t + \eta_t \\ \eta_t &\sim \text{NID}(0, \sigma_\eta(t)^2) \end{aligned} \tag{1.5}$$

We assume $|\phi| < 1$, in order to obtain stationarity, and assume normality of η_t. So the state space equation is Gaussian and linear. We also assume diffuse initialization for α_0. The state space model itself is nevertheless non-Gaussian and nonlinear, as the link function ($\overline{\sigma}^2 s_t \exp(\alpha_t)$) is non linear and the pmf for the observation equation (Y_t) is non-Gaussian.

Observe that in this general model we allow σ_η to change through time. We consider this parameter to vary according to a fixed function. With this specification one can accommodate deterministic sudden volatility changes through the specification of this function, which can be caused by scheduled announcements, for instance. In this paper we consider this function to be constant.

1.4.1 The Seasonality Pattern

A salient feature of intraday volatility is seasonality. This aspect has been documented in many studies (e.g Andersen and Bollerslev (1997)). There are several reasons for deterministic seasonality, including the usual schedule for economic and company news releases, lunch-time, the market opening and market closing. A common pattern is a high volatility on the market opening, which falls through the lunch-time and rises again over day afternoon until the makert close. As shown in Andersen (1996), the intraday volatility is also related to the volume of trading activity, which is itself influenced by the events just described.

We model the seasonality pattern using a parsimonious specification through cubic splines. The treatment is similar to the one used in Harvey and Koopman (1993) and Koopman, Lit and Lucas (2017). Specifically, we write s_t as a zero-sum cubic spline function

$$s_t = \beta' W_t. \tag{1.6}$$

W_t is calculated as described in Poirier (1973). β is a $K \times 1$ vector of parameters associated with $K + 1$ spline knots. In this paper we set $K = 3$ and set the knots for the spline at

the times 10:00, 12:00, 13:30 and 17:00. These times reflect the market opening, start and end of lunch time and market closing.

1.5 Estimation Procedure and state extraction

The model described by equations (1.3),(1.4),(1.5) and (1.6) represents a non-linear non-Gaussian state space. The equation (1.4) represents the non-linear non-Gaussian observation equation while equation (1.5) represents a linear state space equation. We follow Durbin and Koopman (2012) and proceed the parameter estimation by Maximum Likelihood Estimation. Let T be the sample size and denote the model parameters by ψ. In order to proceed with this estimation, we assume that given the realizations $\alpha := (\alpha_0, \alpha_1, \cdots, \alpha_T)$ the observations $y := (y_1, \cdots, y_T)$ are conditionally independent.

1.5.1 Log-Likelihood estimation: Importance Sampling

The joint conditional density can be written as

$$\begin{aligned} p(y|\alpha, \psi) &= \prod_{t=1}^{n} p(y_t|\alpha_t; \psi) \\ p_g(\alpha, \psi) &= \prod_{t=1}^{n} p_g(\alpha_t; \psi). \end{aligned} \tag{1.7}$$

Where p_g is the unconditional distribution of α_t obtained by the state space equation (1.5) with parameters specified in ψ. And one possible expression for the likelihood function is

$$L(y, \psi) = \int p(y, \alpha; \psi) d\alpha = \int p(y|\alpha; \psi) p_g(\alpha; \psi) d\alpha. \tag{1.8}$$

We can use numerical integration techniques to evaluate this last integral, including Monte Carlo. This approach is common in the literature. A naive Monte Carlo procedure can be directly obtained by equation (1.8). We could sample from the unconditional distribution of α_t:

$$\begin{aligned} L(y, \psi) = \int p(y, \alpha; \psi) d\alpha &\simeq \frac{1}{S} \sum_{k=1}^{S} p(y|\alpha^{(k)}; \psi) \\ \alpha^{(k)} &\sim p_g(\alpha; \psi). \end{aligned} \tag{1.9}$$

However, such naive procedure is extremely inefficient as it relies on a simulation of α that does not take the observations y into account. The integral in equation (1.9) is evaluated

efficiently using importance sampling, where we can sample α from a distribution that is closer to the unknown condition distribution $p(\alpha|y;\psi)$.

We consider a Gaussian proxy for $p(\alpha|y;\psi)$. Let $g(\alpha|y;\tilde{\psi})$ be such proxy, with $\tilde{\psi}$ denoting its parameters. This proxy should be obtained by efficient numerical methods, in order to obtain a feasible method of estimating (1.8). Let $\psi^* = (\psi, \tilde{\psi})$. We can write

$$L(y,\psi^*) = \int p(y,\alpha;\psi)d\alpha = \int \frac{p(y,\alpha;\psi)}{g(\alpha|y;\tilde{\psi})} g(\alpha|y;\tilde{\psi})d\alpha \tag{1.10}$$

The importance sampling estimate is thus given by

$$\begin{gathered} \frac{1}{S}\sum_{k=1}^{S} \omega(y,\alpha^{(k)};\psi^*) \\ \omega(y,\alpha^{(k)};\psi^*) = \frac{p(y,\alpha^{(k)};\psi)}{g(\alpha^{(k)}|y;\tilde{\psi})} \\ \alpha^{(k)} \sim g(\alpha|y;\tilde{\psi}). \end{gathered} \tag{1.11}$$

In order to obtain convergence, it suffices to have finite variance in $\omega(y,\alpha^{(k)};\psi)$ to apply the central limit theorem. As done in Durbin and Koopman (2012), we let the Gaussian proxy g be given by $g(\alpha;\psi) = p_g(\alpha;\psi)$. So we can write

$$\omega(y,\alpha;\psi) = \frac{p(y,\alpha;\psi)}{g(\alpha|y;\psi)} = \frac{p(y|\alpha;\psi)p_g(\alpha;\psi)}{g(y|\alpha;\tilde{\psi})g(\alpha;\psi)/g(y;\psi^*)} = g(y;\psi^*)\frac{p(y|\alpha;\psi)}{g(y|\alpha;\psi)} \tag{1.12}$$

1.5.2 Finding the Gaussian Proxy

The closer $g(\alpha|y;\psi^*)$ is to $p(\alpha|y;\psi)$, the more efficient will be the importance sampling procedure. But one should also take into account the computational burden in obtaining $g(\alpha|y;\psi^*)$, as one could also increase the sample as an alternative to obtain less variance in the Monte Carlo estimate.

There are several proposals for $g(\alpha|y;\psi^*)$. One is given in Durbin and Koopman (2012), where $g(\alpha|y;\psi^*)$ is obtained by a 'linearisation' process, by second order Taylor expanding $p(y|\alpha;\psi)$ around the mode of $\alpha|y$. $g(\alpha|y;\psi^*)$ is obtained iterating on the linearised state space model, which can be done efficiently using the Kalman Filter and Smoother. This approach was introduced by Shephard and Pitt (1997) and Durbin and Koopman (1997).

The 'Efficient Importance Sampling'(EIS) method is an alternative and is used in Liesenfeld and Richard (2003) and Richard and Zhang (2007). In this method, the Gaussian importance density parameters are found for each $t = 1, \ldots, T$ through

$$\begin{aligned} \tilde{\psi}_t &= \underset{\tilde{\psi}_t, \lambda_{0t}}{\arg\min} \int \lambda^2(y_t, \alpha_t; \psi_t^*) \omega_t(y_t.\alpha_t; \psi_t^*) g(\alpha_t | y; \psi_t^*) d\alpha_t \\ \lambda(y_t, \alpha_t; \psi_t^*) &:= \log \omega_t(y_t, \alpha_t; \psi_t^*) - \lambda_{0t} := \log p(y_t | \alpha_t; \psi_t) - \log g(y_t | \alpha_t; \tilde{\psi}_t) - \lambda_{0t}, \end{aligned} \tag{1.13}$$

where $\lambda_{0t} \in \mathbb{R}$ is a normalizing constant that sets the mean of $\lambda(y_t, \alpha_t; \psi_t^*)$ to zero.

Richard and Zhang (2007) evaluate (1.13) using importance sampling and calculate the minimization through weighted least squares. Koopman, Lucas and Scharth (2015) use a similar approach, but instead of using importance sampling, they evaluate (1.3) through Gauss-Hermite quadrature Methods, which is highly efficient for low dimensional state spaces. The minimization is also done through weighted least squares and the final log-likelihood estimates can be controlled by control variables to further increase the method efficiency. They call this method as 'Numerically Accelerated Importance Sampling' (NAIS). We use this method to find $g(\alpha | y; \psi^*)$.

For estimating efficiently over higher dimensional state spaces, one can follow Koopman, Lit and Nguyen (2012) and Pinto (2012).

1.5.3 NAIS

In this subsection we derive the NAIS procedure. Our procedure will be for a generic number of dimensions in the state space, so that we do not assume that the dimension is one or two - as previously done in Koopman, Lucas and Scharth (2015) and Koopman, Lit and Lucas (2017). Let n be the dimension of the state space, that is, the dimension of α_t. We first write the Gaussian distribution $g(y_t | \alpha_t; \tilde{\psi}_t)$ as kernel function of α_t:

$$\begin{aligned} g(y | \alpha_t) &= \prod_{t=1}^{T} g(y_t | \alpha_{;}, \tilde{\psi}_t) \\ g(y_t | \alpha_t; \tilde{\psi}_t) &= \exp(a_t + b_t' \alpha_t - \frac{1}{2} \alpha_t' C_t \alpha_t) \\ g(y, \alpha_t; \tilde{\psi}_t) &= \prod_{t=1}^{T} g(y_t | \alpha_{;}, \tilde{\psi}_t) p_g(\alpha_t | \alpha_{t-1}; \psi_t), \end{aligned} \tag{1.14}$$

with scalar a_t, $n \times 1$ vector b_t and $n \times n$ matrix C_t. α_t is also $n \times 1$. a_t is a scalar set so that $g(y_t|\alpha_t; \tilde{\psi}_t)$ integrates to one. So we can set $\tilde{\psi}_t := \{b_t, C_t\}$.

Note that the conditional density in (1.14) can be equivalently written as the density of a linear Gaussian state space model, as done in Shephard and Pitt (1997) and Durbin and Koopman (1997). Let $y_t^* := C_t^{-1} b_t$ and consider the following linear Gaussian state spate model

$$\begin{aligned} y_t^* &= \alpha_t + \varepsilon_t, \quad \varepsilon_t \sim \text{NID}(0, C_t^{-1}) \\ \alpha_{t+1} &= \phi \alpha_t + \eta_t, \quad \eta_t \sim \text{NID}(0, \sigma_\eta^2). \end{aligned} \tag{1.15}$$

Then the conditional density on the observation equation is

$$\begin{aligned} \log g(y_t^*|\alpha_t; \tilde{\psi}_t) &= \frac{1}{2}\left\{ -\log 2\pi + \log|C_t| - (C_t^{-1} b_t - \alpha_t)' C_t (C_t^{-1} b_t - \alpha_t) \right\} \\ &= a_t + b_t' \alpha_t - \frac{1}{2} \alpha_t' C_t \alpha_t, \end{aligned} \tag{1.16}$$

where a_t collects all terms that are constant w.r. to α_t. Therefore $g(y_t^*|\alpha_t; \psi)$ and $g(y_t|\alpha_t; \psi)$ have the same kernel and we conclude that $g(y_t^*|\alpha_t; \psi) \equiv g(y_t|\alpha_t; \psi)$. As the state space equation is also identical, we conclude that $g(y_t^*, \alpha_t; \psi) \equiv g(y_t, \alpha_t; \psi)$. This representation is useful because it provides a way to compute $g(\alpha|y; \psi^{*(k)})$ efficiently. We have just proved that

$$g(\alpha_t|y; \psi^{*(k)}) = N(\hat{\alpha}_t^{(k)}, \hat{P}_t^{(k)}), \tag{1.17}$$

where $\hat{\alpha}_t^{(k)}$ and $\hat{P}_t^{(k)}$ are the mean and covariance matrix for the states α_t conditional on the whole data (y). They can estimated by the Kalman Filter and Smoother applied to (1.15).

Now we turn to NAIS in order to obtain efficient estimates for $\tilde{\psi}$. In NAIS itself, we obtain $\tilde{\psi}$ iteratively. Again, let the parameters for $p(y_t|\alpha)$ be ψ_t and let $\psi_t^{*(k)} := (\psi_t, \tilde{\psi}_t^{(k)})$. Given $\tilde{\psi}_t^{(k)}$ we obtain $\tilde{\psi}_t^{(k+1)}$ by

$$\tilde{\psi}_t^{(k+1)} = \underset{\tilde{\psi}_t^{(k+1)}}{\arg\min} \int \lambda^2(y_t, \alpha_t; \psi^{*(k+1)}) \omega_t(y_t, \alpha_t; \psi^{*(k)}) g(\alpha_t|y; \psi^{*(k)}) d\alpha_t. \tag{1.18}$$

We evaluate this integral numerically, using Gauss-Hermite Quadrature. Let ϕ be the

integrand in (1.18),

$$\phi(y_t, \alpha_t; \tilde{\psi}_t^{(k+1)}, \tilde{\psi}_t^{*(k)}) := \lambda^2(y_t, \alpha_t; \psi^{*(k+1)})\omega_t(y_t, \alpha_t; \psi^{*(k)})g(\alpha_t|y; \psi^{*(k)}). \tag{1.19}$$

To shorten the notation, consider $\phi(\alpha_t) := \phi(y_t, \alpha_t; \tilde{\psi}_t^{(k+1)}, \tilde{\psi}_t^{*(k)})$. Let $F_t^{(k)}$ be the Cholesky decomposition of $\hat{P}_t^{(k)}$, such that $F_t^{(k)} F_t^{(k)\prime} = \hat{P}_t^{(k)}$. Also let $\mathcal{M} = \{1, \cdots, M\}$, where M is the number of desired points in the Gauss-Hermite quadrature per dimension. In each dimension we can generate M points $\mathcal{Z} := \{\tilde{z}_s\}_{s\in\mathcal{M}}$ with Gauss-Hermite weights $h(\tilde{z}_s)$. We then consider the multidimensional version of the Gauss-Hermite quadrature by using $z \in \mathcal{Z}^n \subset \mathbb{R}^n$, whose weights are given by the product of $h(\tilde{z}_s)$ for each coordinate $\tilde{z}_s$ of z. We can index all the $\mathcal{Z}^n$ elements using elements $j \in \mathcal{M}^n$, whose coordinates we denote by $j_s, s \in \{1, \cdots, n\}$.

Then using this multidimensional Gauss-Hermite Quadrature, in all the n state variables, we obtain the points $z \in \mathbb{R}^n$ by approximating equation (1.7) by:

$$\begin{aligned} \int \phi(\alpha_t)d\alpha_t &\simeq \sum_{j\in\mathcal{M}^n} \tilde{w}(z_j)\phi(\tilde{\alpha}_{t,j}^{(k)}), \quad \text{where} \\ \tilde{w}(z_j) &:= \prod_{s=1}^{n} h(z_{j_s})e^{z_{j_s}^2} \\ \tilde{\alpha}_{t,j}^{(k)} &= \hat{\alpha}_t^{(k)} + F_t^{(k)} z_j \\ z_j &:= (z_{j_1}, \cdots, z_{j_n})'. \end{aligned} \tag{1.20}$$

And we just generalized the NAIS method to n dimensions in the most straightforward way. It is worth saying that this method of integration through Gauss-Hermite integration is not the most appropriate for high dimensional state space problems. For this case, using NAIS, we refer to Pinto (2012). He changes the estimation exactly in this step, considering a multi-step Quasi-Monte Carlo method.

Using equation (1.20) the numerical version of (1.18) becomes

$$\arg\min_{\tilde{\psi}_t^{(k+1)}} \sum_{j\in\mathcal{M}^n} \tilde{w}(z_j)\phi(y_t, \tilde{\alpha}_{t,j}^{(k)}; \tilde{\psi}_t^{(k+1)}, \tilde{\psi}_t^{*(k)}) \tag{1.21}$$

Substituting the definition of ϕ and λ, we obtain equation (1.21) as

$$\begin{aligned} \arg\min_{\tilde{\psi}_t^{(k+1)}} \sum_{j\in\mathcal{M}^n} w_j[\log p(y_t|\tilde{\alpha}_{t,j}^{(k)}; \psi_t) - \log g(y_t|\tilde{\alpha}_{t,j}^{(k)}; \tilde{\psi}_t^{(k+1)}) - \lambda_{0t}]^2 \\ w_j := \tilde{w}(z_j)\omega_t(y_t, \tilde{\alpha}_{t,j}^{(k)}; \psi_t^{*(k)})g(\tilde{\alpha}_{t,j}^{(k)}|y; \psi_t^{*(k)}) \end{aligned} \tag{1.22}$$

Then using equation (1.14) one can see that this equation is equivalent to a weighted least squares regression, where $\tilde{\psi}_t = (b_t, \text{vech}(C_t))$ can be obtained as the coefficients for $\tilde{\alpha}_{t,j}^{(k)}$ and $\text{vech}(\tilde{\alpha}_{t,j}^{(k)}\tilde{\alpha}_{t,j}^{(k)'})$, respectively. In other words, we run a weigthed regression with weight w_j and the M^n observations with the equation

$$\log p(y_t|\tilde{\alpha}_{t,j}^{(k)};\psi) = \text{constant} + \kappa'\tilde{\alpha}_{t,j}^{(k)} - \frac{1}{2}\xi'\text{vech}(\tilde{\alpha}_{t,j}^{(k)}\tilde{\alpha}_{t,j}^{(k)'}) + \text{error} \tag{1.23}$$

And store the coefficient results as $b_t^{(k+1)} = \kappa$ and $\text{vech}(C_t^{(k+1)}) = \xi$.

So we have updated $\tilde{\psi}_t^{(k+1)}$ as a function of $\tilde{\psi}_t^{(k)}$. We repeat this procedure until convergence is obtained, as measured by the distance between $\tilde{\psi}_t^{(k+1)}$ and $\tilde{\psi}_t^{(k)}$.

We sumarise the described NAIS algorithm as

1. Find appropriate values for starting the algorithm, setting $\tilde{\psi}_t^{(0)}$. The algorithm usually converges for every starting point. Usually setting b_t to ones and $C_t = I_n$ suffices. It will be faster, of course, if the starting point is closer to the fixed point solution.
2. Given $\tilde{\psi}_t^{(k)}$, Construct the linear state space model defined in equation (1.4). Run Kalman Filter and Smoother in order to obtain $\hat{\alpha}_t^{(k)}$ and $F_t^{(k)}$.
3. Obtain $\tilde{\psi}_t^{(k+1)}$ by weighted least squares using equation (1.22).
4. if $||\tilde{\psi}_t^{(k+1)} - \tilde{\psi}_t^{(k)}|| < \epsilon$, then the algorithm has converged and the solution is $\tilde{\psi}_t^{(k+1)}$. Otherwise set k=k+1 and go to step 2.

It is worth mentioning that after the NAIS convergence, we have an efficient way to generate random sample from $g(\alpha|y;\tilde{\psi})$. We can use the state space (1.15) to generate such sample, using the Kalman Filter and Smoother. See Durbin and Koopman (2012) regarding the Linear State Space Simulation Smoother.

A final essential feature of the NAIS procedure just described is its robustness to the range of values returned by the procedure described by equations (1.13) and its practical implementation, equation (1.23). As these optimizations are unconstrained, one can expect in practice that the C_t matrices obtained by such procedure will not be positive definite in general. But the procedure is robust to that fact and all computations can be executed normally even in that case, as argued in Pinto (2012) and Koopman, Lucas and Scharth (2015).

1.5.4 State extraction: Mode, Smoothing and Filtering

After estimating the model parameters by the method described in the previous section, we need to extract the states from the model in order to make forecasts or just infer the distribution of the instantaneous volatility that happened in the past. In what follows we call smoothed estimates for a function of α_t the estimates that use whole dataset $y_1, \ldots, y_T$ and call filtered estimates the estimates for such function the estimates that only the dataset $y_1, \ldots, y_{t-1}$.

The procedure already gives us a valuable information. The $\hat{\alpha}_t$ and $\hat{F}_t$ that are part of the output of the NAIS procedure are already estimates for the mode of α_t and its covariance matrix, conditional on the whole data. So they are 'smoothed' estimates for the mode.

In order to obtain smoothed estimates for the α_t we can also use the importance sampling procedure described at subsection 1.5.1, analogously to the equation (1.11). Specifically, let h be a function of α we can write (using the same notation as in subsection 1.5.1):

$$E[h(\alpha)] = \int h(\alpha)p(y,\alpha;\psi)d\alpha = \int h(\alpha)\frac{p(y,\alpha;\psi)}{g(\alpha|y;\psi)}g(\alpha|y;\psi)d\alpha \tag{1.24}$$

The importance sampling estimate is thus given by

$$\begin{gathered}\frac{1}{S}\sum_{k=1}^{S} h(\alpha^{(k)})\omega(y,\alpha^{(k)};\psi) \\ \omega(y,\alpha^{(k)};\psi) = \frac{p(y,\alpha;\psi)}{g(\alpha|y;\psi)} \\ \alpha^{(k)} \sim g(\alpha|y;\psi)\end{gathered} \tag{1.25}$$

If we let $h(\alpha) = \alpha$ we can use equation (1.24) to obtain estimates for the mean. We can also let $h(\alpha) = \alpha\alpha'$ to obtain estimates for the second order moments. The $\alpha^{(k)}$ are generated by the same procedure used in estimating the log-likelihood which is the Linear State Space Simulation Smoother applied to the linear state space decribed by equation (1.15). See Durbin and Koopman (2012).

Finally, we need to obtain filtered estimates for the volatility in order to make the forecasting comparisons we sought in this paper. One way to obtain such estimates is to repeatedly obtain smoothed estimates up to $t-1$, either using the mode or the mean and then use state space equation in order to go one step ahead. This approach is time consuming

as one needs to recalculate all previous states again at each interaction. Koopman, Lit and Lucas (2017) follows this approach and use the mode, calculated through NAIS as described above.

We use a different approach in this paper. We calculate the filtered estimates by using the Bootstrap Particle Filter applied to the non-linear non-Gaussian state space described by equations (1.13) - (1.16). A formal description of such algorithm follows. Let N be the number of particles (we use 200), p_g be the Gaussian distribution of the states α_t implied by the state space equation in (1.15) and $h(\alpha)$ be our function of interest.

For all $i \in 1, \ldots, N$:

1. Sample $\tilde{\alpha}_t^{(i)}$ from $p_g(\alpha_t|\alpha_{t-1}^{(i)})$
2. Compute the corresponding weights $\tilde{w}_t^{(i)}$

$$\tilde{w}_t^{(i)} = p_g(y_t|\tilde{\alpha}_t^{(i)}), \quad i = 1, \ldots, N$$

   and normalize:

$$w_t^{(i)} = \tilde{w}_t^{(i)} / \sum_{j=1}^{N} \tilde{w}_t^{(j)}$$

3. Compute expectation of the function of interest

$$\hat{h}_t = \sum_{i=1}^{N} w_t^{(i)} h_t(\tilde{\alpha}_t^{(i)})$$

4. Resample the particles: draw N independent particles $\alpha^{(i)}$ from $\{\tilde{\alpha}_t^{(1)}, \ldots, \tilde{\alpha}_t^{(N)}\}$ with replacement with the corresponding probabilities $\{w_t^{(1)}, \ldots, w_t^{(N)}\}$

And, again, we can use $h(\alpha) = \alpha$ and $h(\alpha) = \alpha\alpha'$ to calculate the first and second filtered moments.

1.6 Optimization Procedure

In this section we describe in greater detail the model optimization procedure used to maximize the log-likelihood. We estimated the models by maximum likelihood. The log-likelihood function is calculated by the equations (1.10) and (1.11). For each interaction the same random numbers are used in order to ensure that the sampling error will not affect the calculus of the derivatives.

To maximize of the log-likelihood function we apply a version of the Coordinate Descent Algorithm. The reason behind this choice is that the parameters that affect the measurement

equation have derivatives that are easier to calculate and the coordinate descent method also diminishes the number of NAIS procedure runs.

Let $\theta = \{\theta_y, \theta_\alpha\}$ represent the model parameters, where θ_y represents the parameters related to the observation equation while θ_α represents the parameters related to the state equation. The algorithm is the following:

1. Start with a initial guess $\theta = \theta_0$.
2. Maximize the log-likelihood function w.r. to θ_α, using the NAIS algorithm to calculate the mode $\hat{\alpha}_t$ and the Gaussian proxy parameters (b_t and C_t) at each iteration.
3. Maximize the log-likelihood function conditional on the values obtained in the previous step with respect to θ_y, while keeping $\hat{\alpha}_t$ and the Gaussian proxy parameters (b_t and C_t) constant.
4. Repeat the procedure until the change in the final parameters (θ) is less than the tolerance ε.

In each iteration of the above algorithm, we use numerical maximization for steps 2 and 3, as the functions are intractable in both cases. We prefer to split the optimization in these two separate optimizations because there is no need to recalculate the NAIS parameters during the optimization in step 3, making it really faster.

We use the BFGS algorithm for step 3. As we keep $\hat{\alpha}_t$ constant at this step, the derivatives for the log-likelihood with respect to θ_y are analytical for this step, making the BFGS algorithm more efficient.

For the step 2 we use the Constrained optimization by linear approximation (Cobyla) algorithm, described in Powell (1994). As we need to update the states $\hat{\alpha}_t$ for each interaction, the log-likelihood function may have too low numerical accuracy at some points for a Quasi-Newton procedure like BFGS. We find that this method is much more robust for this step, as it does not involve derivatives calculation.

The whole algorithm generally converges fast, taking just a few iterations at the outer loop of the Coordinate Descent algorithm. One advantage of the proposed algorithm is that the NAIS procedure is only needed at the step 2, and this step only have two parameters to estimate. Another advantage is the possibility of using analytic expression for the derivative of the log-likelihood function at the step 3.

We estimated the two models described on table 2. We estimated the models for every 5

Table 2 – Estimated Models' Characteristics

Name	estimation sample size	mean equation	forecast sample size
SS	5 days	$\mu_t = 0$	5 days
SSM	5 days	$\mu_t = \delta y_{t-1}$	5 days

consecutive business days periods ('weekly'), for all 'weeks' of 2018. As the models involve seasonality with the period consisting of one day, estimating with more than one day in the sample should help identifying the difference between the seasonality and the state space factors of the volatility process. In this respect we differ from Koopman, Lit and Lucas (2017), that estimated the models with sample size equal to one day.

1.7 Estimation Results

Now we describe the in-sample estimation results for the two models (SS and SSM). The models were estimated for 48 different 5 business days periods in 2018 for all the four equities in our list. In this section we cover the parameter estimates, the estimated seasonality and the filtered volatilities fora sample data.

1.7.1 Parameter Estimates

We report the descriptive statistics for the parameter estimates for model SS and SSM in the tables 3 and 4 respectively. For each equity / parameter we report the sample mean for the parameter estimates on the first row and the standard deviation in parenthesis on the second row.

Table 3 – Descriptive Statistics for SS Model Estimates

	$\overline{\sigma}$	γ	ϕ	${\sigma_\eta}^2$	β_0	β_1	β_2
Bradesco	1.43	0.11	0.96	0.023	0.91	-0.047	-0.12
	(0.10)	(0.12)	(0.038)	(0.029)	(0.28)	(0.085)	(0.075)
Itau-Unibanco	1.58	0.40	0.91	0.062	0.77	-0.00525	-0.12
	(0.15)	(0.27)	(0.061)	(0.054)	(0.32)	(0.087)	(0.093)
Petrobras	1.30	-0.13	0.99	0.0072	0.53	-0.043	-0.0070
	(0.17)	(0.074)	(0.019)	(0.012)	(0.35)	(0.12)	(0.066)
Vale	1.64	0.32	0.87	0.098	0.85	0.042	-0.13
	(0.24)	(0.28)	(0.089)	(0.075)	(0.43)	(0.10)	(0.11)

The descriptive statistics for SS already describe some interesting features on the data that worth mentioning. First, as expected, the ϕ parameter estimates look as close to one as the estimates in Koopman, Lit and Lucas (2017). And this is in line with the expected behavior. The estimates for the σ factor are all at the same order of magnitude.

The betas also have all the same pattern, with $\beta_0 >> (\beta_1, \beta_2)$, suggesting that the seasonality pattern of the 4 stocks should have some similarity. The σ_η parameter, which represents the volatility of the volatility shock, differ for the stocks, being roughly proportional to $1 - \phi$ - showing that the unconditional variance of the α is of the same order of magnitude for all stocks.

The γ parameter shows more variation, with the estimates for the Petrobras being negative while all other are positive. This reflects the marked difference in the histograms shown on section 1.

Observe that the sample standard deviations should not be interpreted as standard errors, and there is no reason to believe that the true model parameters are the same for each estimation. We shall see that some parameters even look to have trended during the year. That said, we see less relative variation for parameters like σ, ϕ and β_0, and more relative variation for γ and σ_η.

Table 4 – Descriptive Statistics for SSM Model Estimates

	σ	γ	δ	ϕ	${\sigma_\eta}^2$	β_0	β_1	β_2
Bradesco	1.39	0.069	-0.22	0.96	0.029	1.00	-0.024	-0.143
	(0.13)	(0.099)	(0.035)	(0.040)	(0.036)	(0.30)	(0.088)	(0.089)
Itau-Unibanco	1.57	0.31	-0.21	0.90	0.075	0.88	0.017	-0.14
	(0.20)	(0.21)	(0.039)	(0.068)	(0.066)	(0.31)	(0.10)	(0.11)
Petrobras	1.25	-0.13	-0.24	0.99	0.0089	0.66	-0.033	-0.020
	(0.16)	(0.059)	(0.062)	(0.018)	(0.015)	(0.34)	(0.13)	(0.080)
Vale	1.64	0.25	-0.20	0.86	0.12	0.95	0.062	-0.14
	(0.28)	(0.24)	(0.048)	(0.088)	(0.086)	(0.44)	(0.12)	(0.13)

Comparing the estimates for SSM parameters on table 4 and the estimates for the SS parameters on table 3, we observe mostly similarities. The estimates for all the common variables in both models have similar means and deviations.

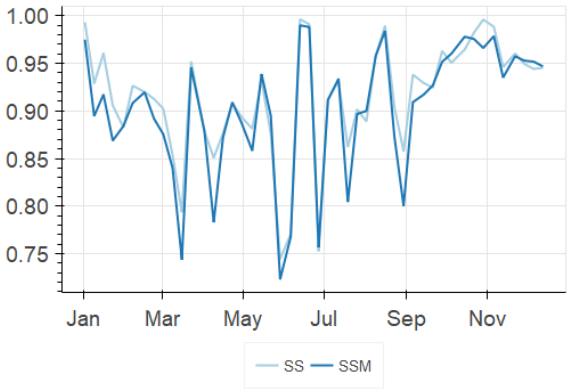
The new parameter, δ, has strikingly similar means across the different stocks. The means are also high when compared the the standard deviations. These estimates already suggest that the model SSM should be preferred to the SS model as δ seem far from zero for most estimates.

We also note that the sign of the mean δ is negative for all stocks. This implies a negative dependency of the mean on previous returns. This effect is expected and has been detected in other works in the literature for intraday data. See Chu, Ding and Pyun (1996) and McInish and Wood (1992). This estimative is new, however, in the context of the model estimated in the present paper. The magnitude of the estimates are also high, as all estimates point to a return reversion that surpass 20% of the previous price change in ticks.

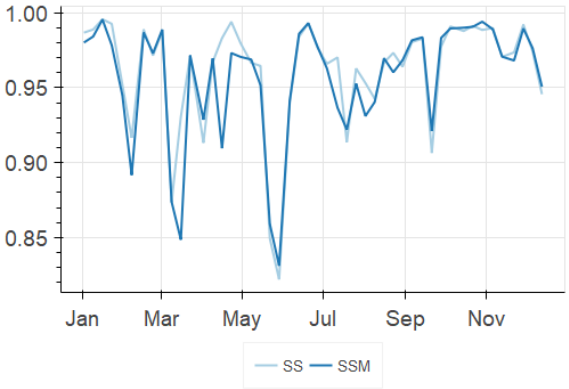
So far we have described the set of estimates as a whole, using descriptive statistics. Now we analyze how the estimates have evolved through time in the year of 2018, describing all coefficient point estimates for both models.

Figure 3 – Estimates for ϕ

(a) Itau-Unibanco

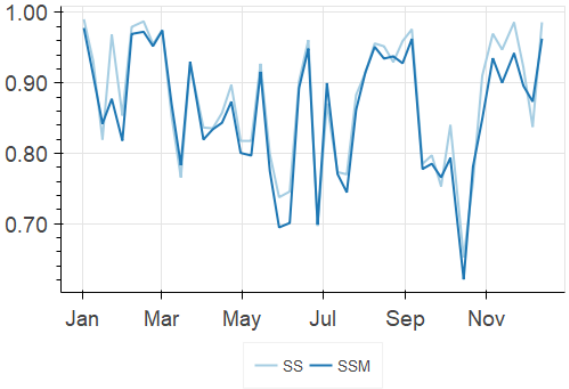


(b) Bradesco



(c) Petrobras

(d) Vale



In figure 3 we see the estimates for ϕ. The estimated show similarities for both models (SS & SSM) for all stocks, for most estimation periods. The estimates vary through time for all stocks, being higher and less volatile for Petrobras, which is the stock with less volatility as measured on a per tick basis (that is, lower σ).

Figure 4 – Estimates for ${\sigma_\eta}^2$

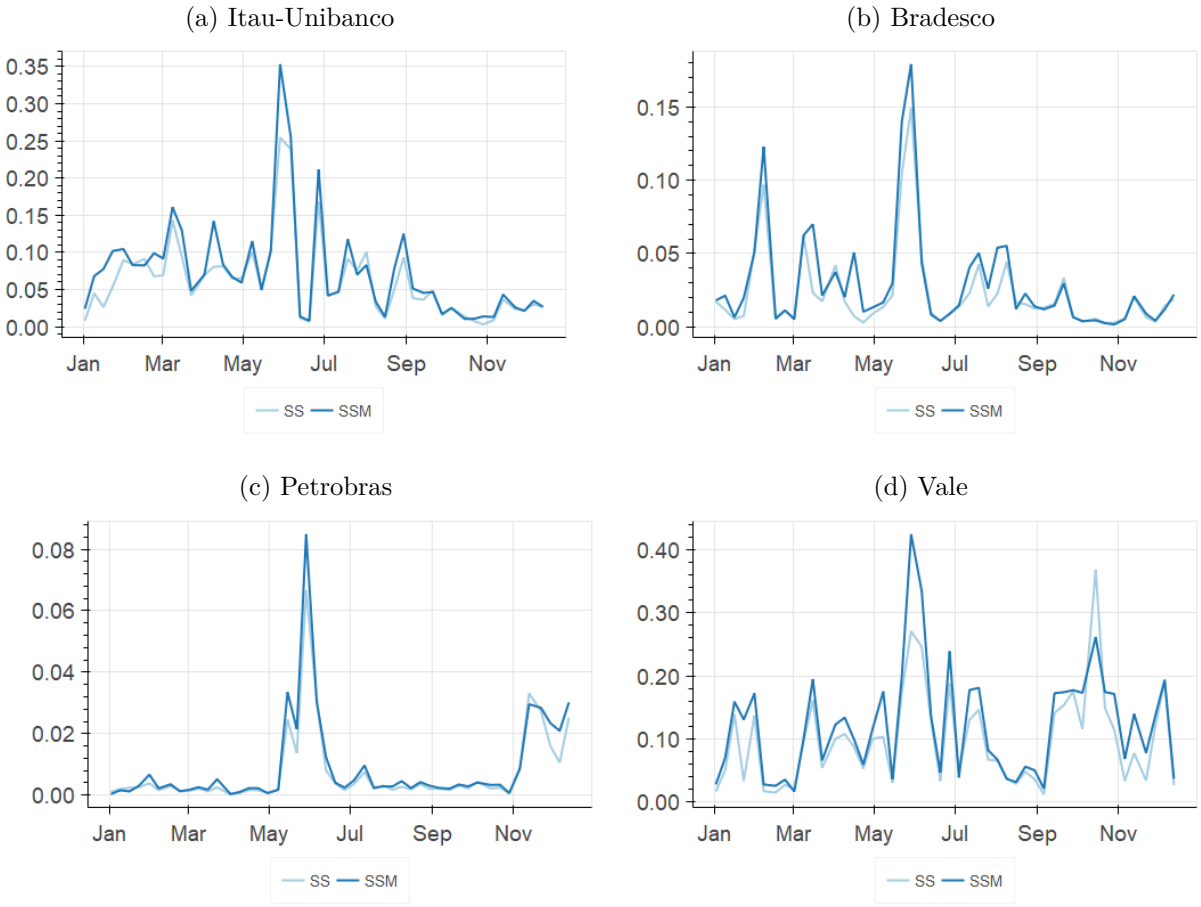


Figure 4 shows estimates for ${\sigma_\eta}^2$. We note again that estimates for both models are similar. As mentioned before, there is a huge level difference between the stocks, mostly in line with the differences with the $1-\phi$ coefficient. Regarding the dynamics, we notice a spike in this coefficient for all stocks between May and July 2018. At this period Brazilian market was hit by three different shocks: (1) A national Truck driver's strike, causing economic turmoil and products' shortage, (2) A temporary deterioration on the elections perception, with a sudden rise in polls of a non market friendly populist candidate (3) an external shock caused by the depreciation of emerging markets currencies.

Figure 5 – Estimates for $\overline{\sigma}$

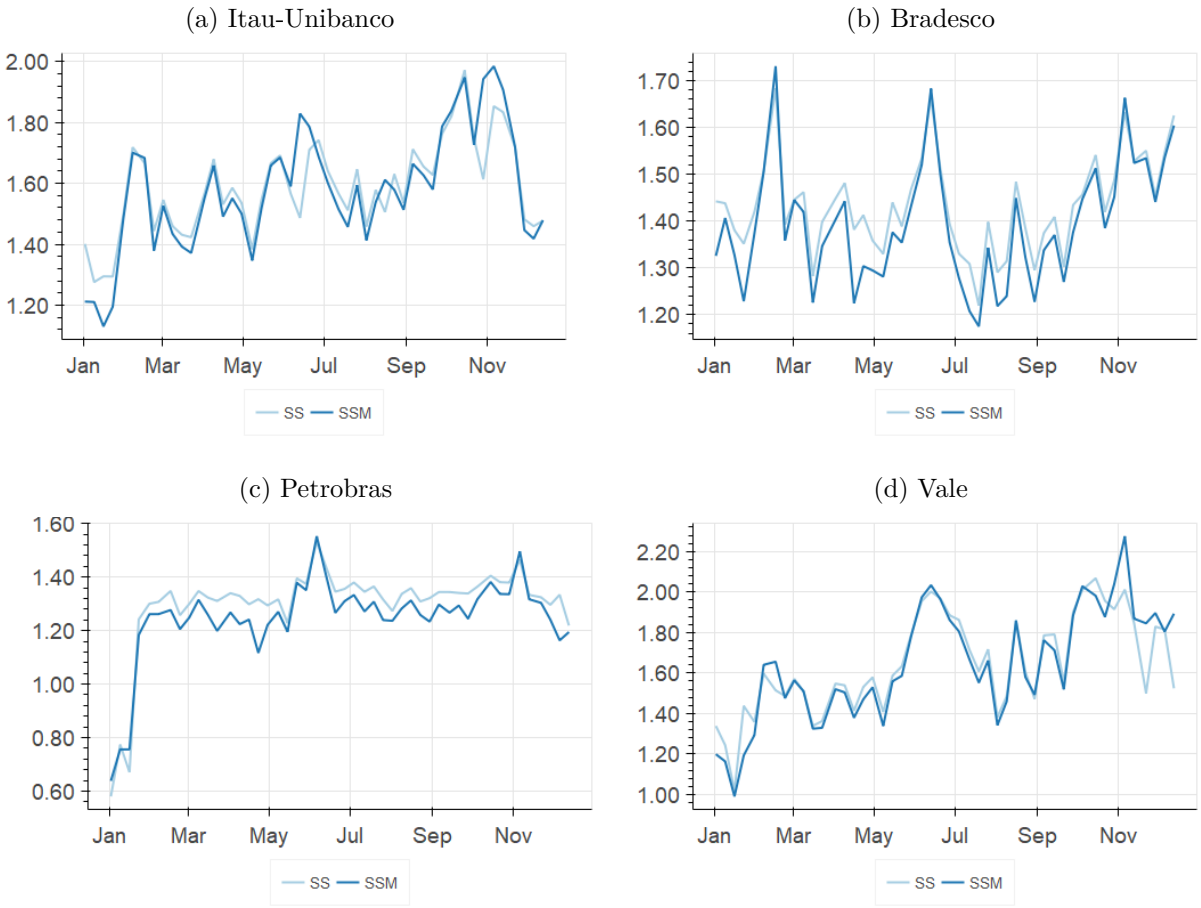


Figure 5 shows estimates for $\overline{\sigma}$. We can see again that the estimates of the two models are similar and that the price return volatility rises between May and July for all stocks, just as in the previous parameter. We also see for this parameter that the volatility rises again on the last quarter, where Brazilian presidential elections happened.

Figure 6 – Estimates for γ

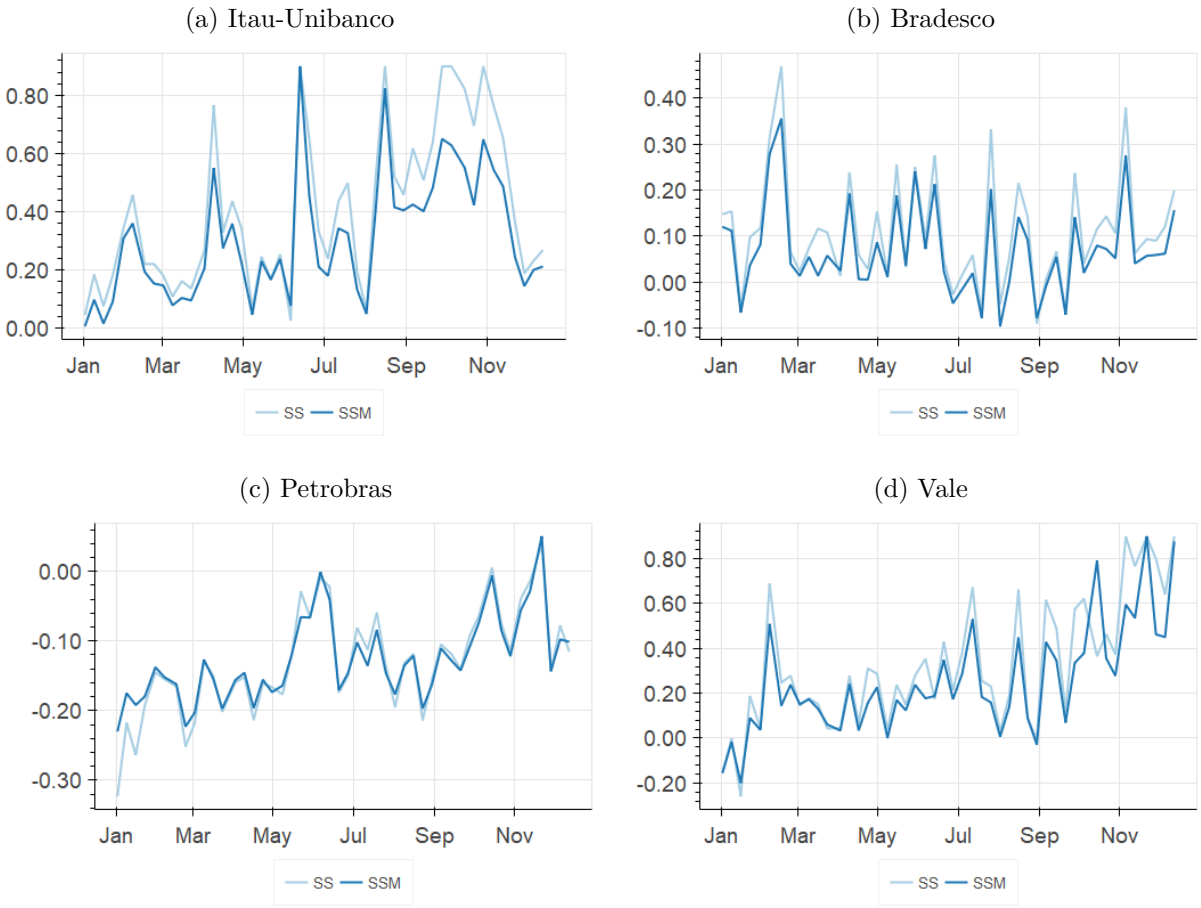


Figure 6 shows estimates for γ. There is a disparity in the levels for the estimates for this parameter across the stocks, notably with Petrobras showing a consistent negative value while all other show mostly positive values. We can also see slightly larger values for this parameter at the last quarter for all stocks except Bradesco.

Figure 7 – Estimates for δ

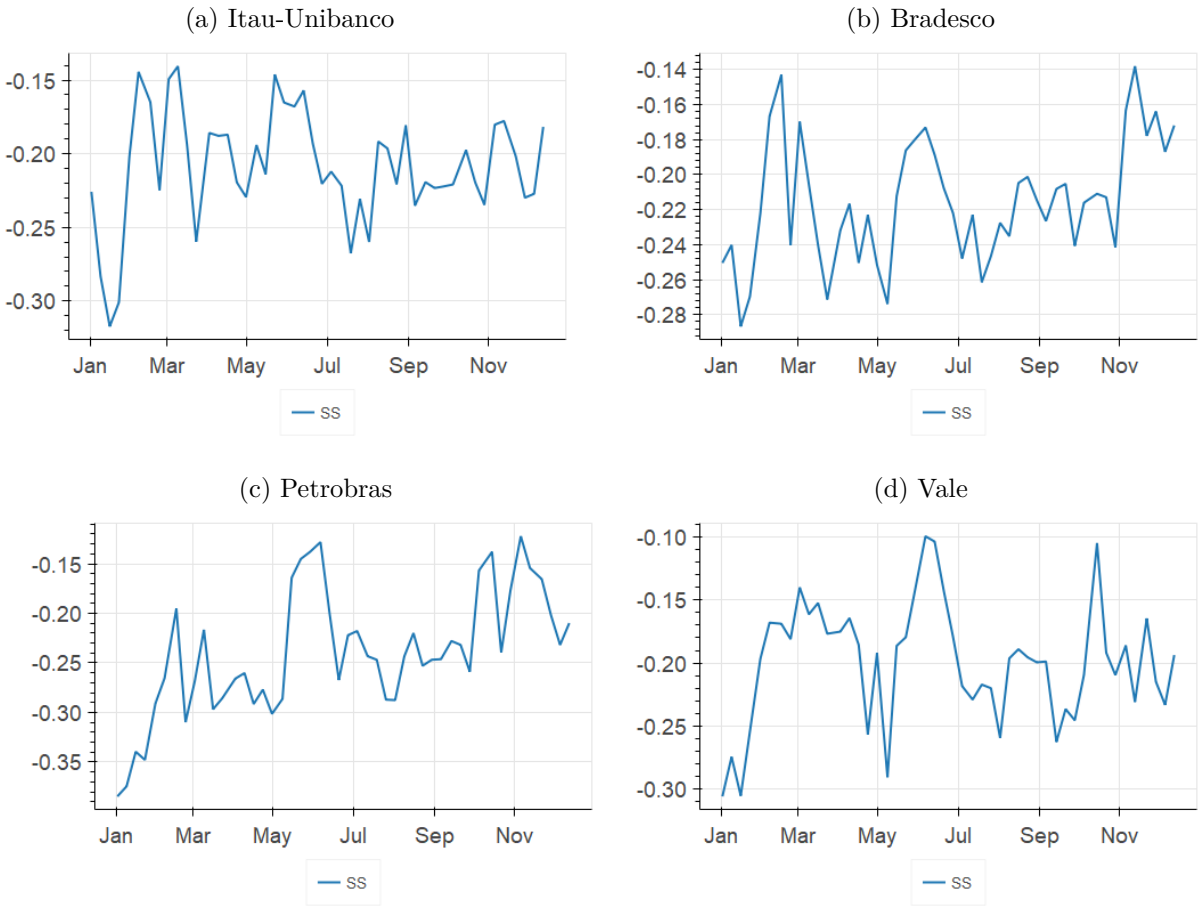
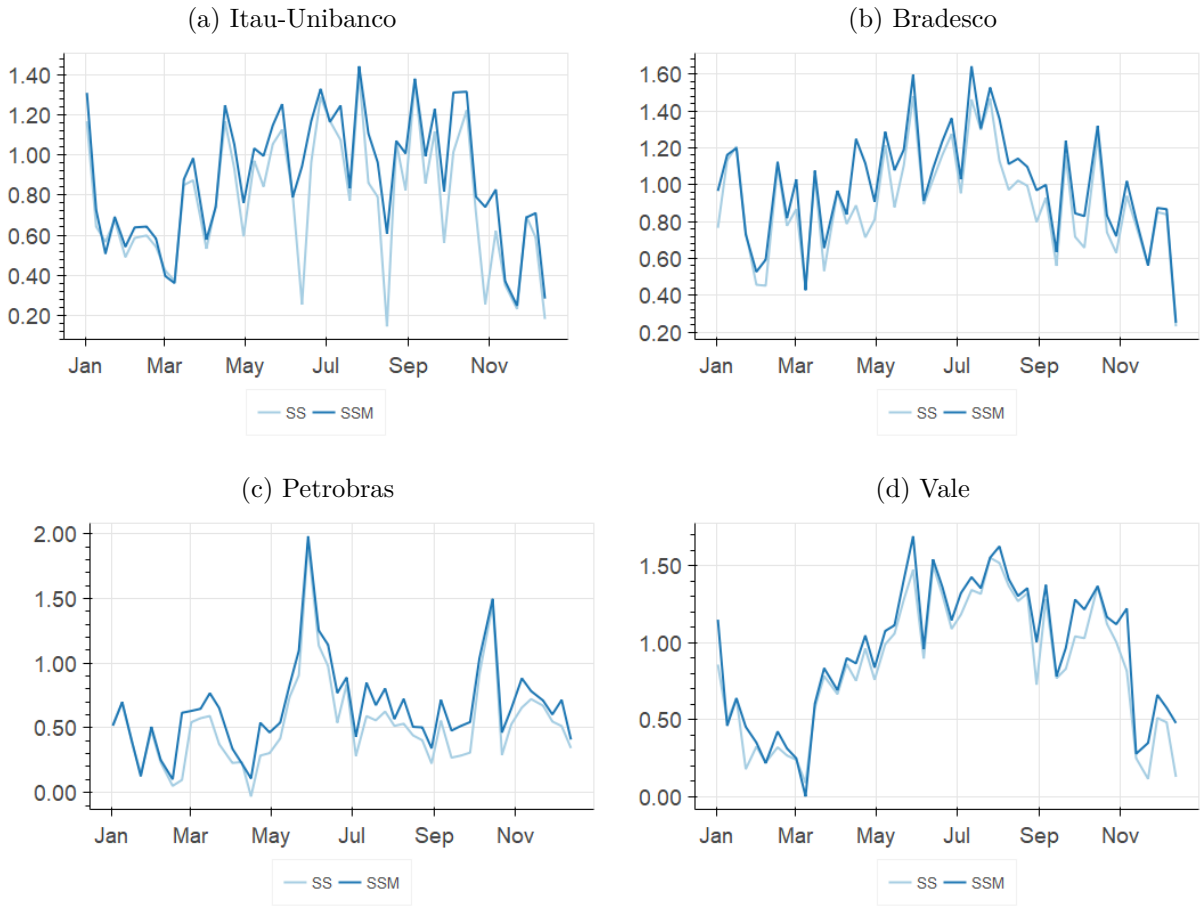


Figure 7 shows the estimates for the new parameter in model SSM. All point estimates for all stocks are negative, being less than -0.10 in all estimates and reaching values below -0.30 for some periods. We note that between May and July all estimates became less negative, implying a lower short term price return reversion for this period, along the higher volatility.

Figure 8 – Estimates for β_0

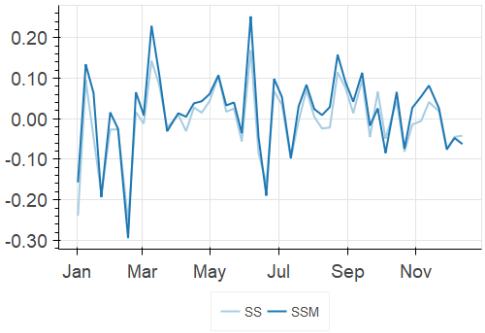


Figures 8, 9 and 10 show the dynamics of the estimates for the parameters related to the seasonality. Looking at the changes in the estimates through the year we can see that β_0 tended to be low at the start and the end of the year, while β_2 showed the opposite dynamic:it was higher at the start and at the end of the year, being lower in between. β_1 had no relevant trends.

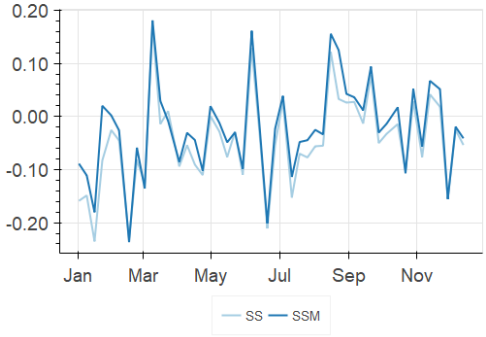
We note that a high β_0 and low β_2 means more volatility at the start of the day. So the changes in parameters β_0 and β_2 show that the seasonality changed through the year with relatively more volatility at the end of the day as compared to the start of the day at the start of the year and at the end of the year.

Figure 9 – Estimates for β_1

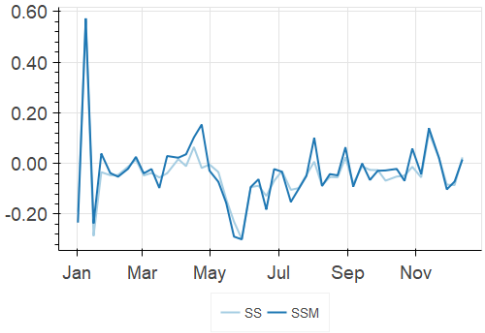
(a) Itau-Unibanco



(b) Bradesco



(c) Petrobras



(d) Vale

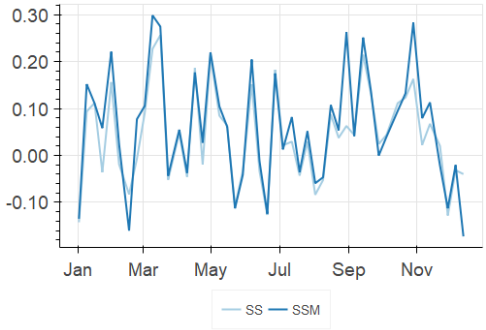
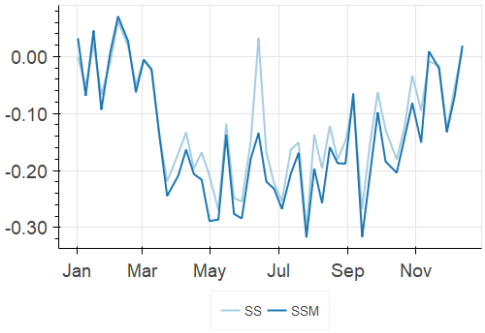


Figure 10 – Estimates for β_2

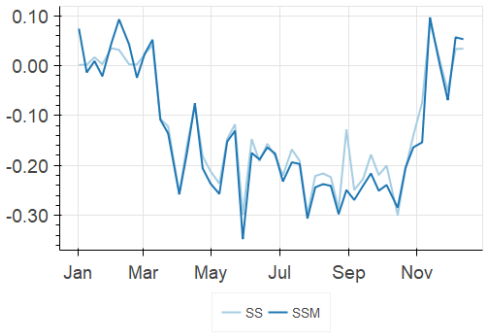
(a) Itau-Unibanco



(b) Bradesco

(c) Petrobras

(d) Vale



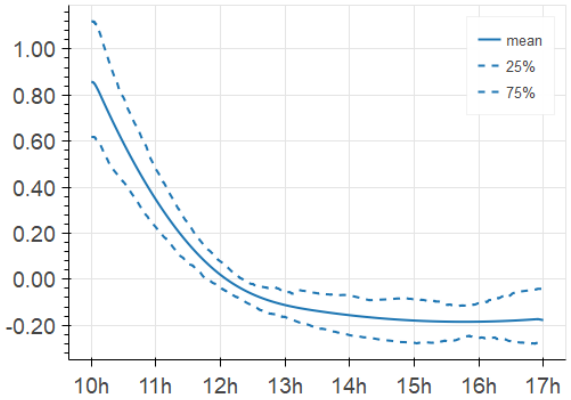
1.7.2 Seasonality

In this sub-section we analyze the descriptive statistics for the volatility seasonality. Figure 11 depicts the seasonality for the SSM model, showing the mean and two percentiles for the log of the seasonality factor for each time of the day.

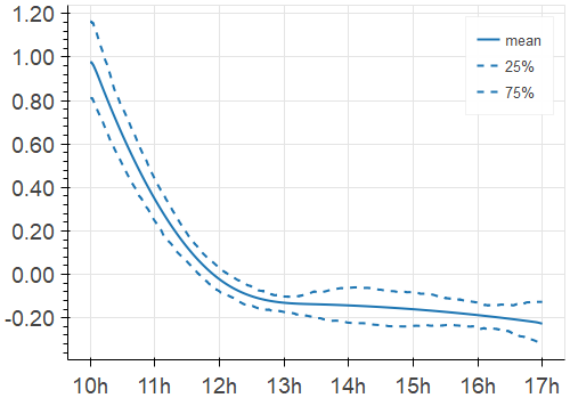
Observe that all stocks show a sharp decrease in the expected volatility from the opening to the end of the day. This effect is also documented in Koopman, Lit and Lucas (2017). One possible reason is that on the starting of the trading session there is more information for the markets to digest, including all the overnight news and economic releases, that are more common in the morning.

Figure 11 – Average Seasonality for 2018

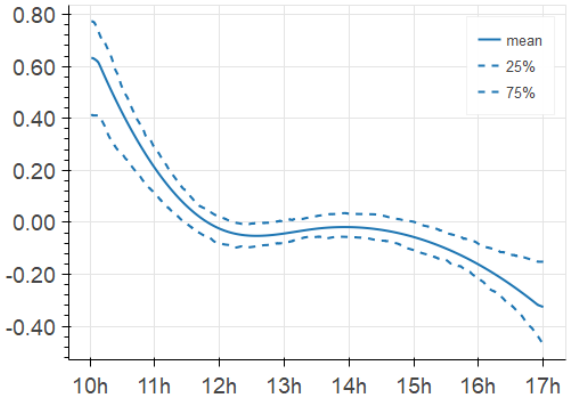
(a) Itau-Unibanco



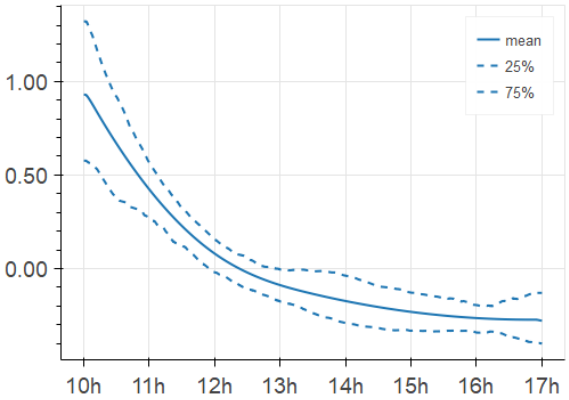
(b) Bradesco



(c) Petrobras



(d) Vale

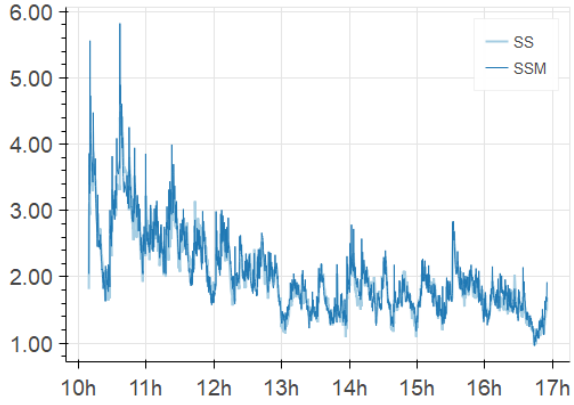


1.7.3 Filtered Volatility Example

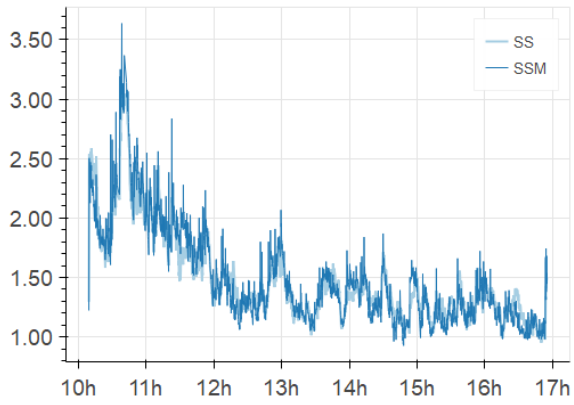
We now show an example of the filtered volatility obtained by the Bootstrap Particle Filter using both SS and SSM models. Figure 12 shows the path of the filtered volatility for the day July 6, 2018. The model was estimated with data from the same 5-days period containing that day. Observe that the volatility decreases during the day as predicted by the mean seasonality in the previous sub-section.

Figure 12 – Filtered Forecast Volatility for July 6, 2018

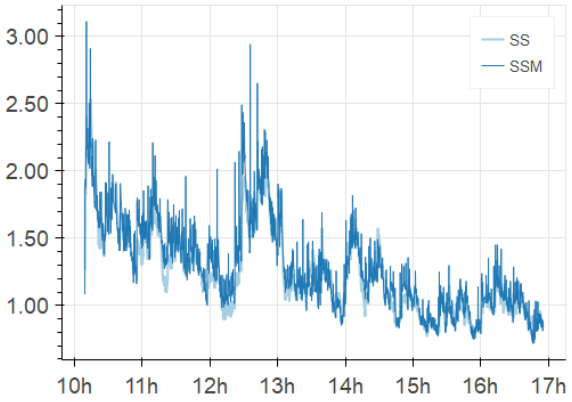
(a) Itau-Unibanco



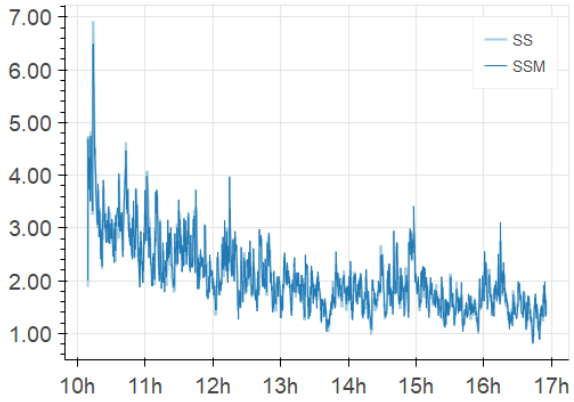
(b) Bradesco



(c) Petrobras



(d) Vale



1.8 Forecasting Performance Comparison

To assess the forecasting performance of the two models we conduct a forecasting exercise for all 5-days periods of 2018. For the forecast of time t and for all models, we use only data up to $t-1$. To compare the performance we estimate the pmf by using empirical forecasting models for the Skellam pmf parameters. The models are:

EW: σ_t is estimated by an EWMA process: $\hat{\sigma}_t^2 = \lambda y_{t-1}^2 + (1-\lambda)\hat{\sigma}_{t-1}^2$. λ is estimated using the fit sample (previous week). μ and γ are set to zero

M_1: σ_t is estimated non-parametrically by a moving average process using a rolling window of the past 90 returns. μ and γ are set to zero

M_2: σ_t is estimated non-parametrically by a moving average process using a rolling window of the past 900 returns. μ and γ are set to zero

E: the pmf of y_t is determined by the empirical probabilities of y_t on the fit sample.

For each 5-day period of 2018, we estimate all the models on the previous 5-day period and calculated the pmf forecasts by iteratively filtering the volatility while keeping the parameters fixed. To assess the forecasting performance we follow Koopman, Lit and Lucas (2017) and used the log-likelihood loss function, which is simply the negative of the sum of the log of the pmf obtained by each model for all the points in the forecasting sample.

In tables 5 and 6 we collect the main statistics for the forecasting performance comparison of the models. For each stock we report the mean log-loglikelihood loss function value for each time on the whole sample (for all periods of 5-day taken together). We also report the Diebold-Mariano statistics comparing all models to the SS Model (DM[SS] column) and comparing all models to the SSM model (DM[SSM] column). In both cases a negative number means that the model in the columns outperforms the model on the row and a positive number means the opposite.

The Diebold-Mariano statistic follow a standard normal distribution. So our exercise suggest that SSM outperforms all other models for forecasting the pmf by a large margin for all stocks. The SS model also outperforms all models for most stocks(except SSM), but with a lower margin. And the EWMA model actually outperformed SS model for the Vale stock.

Table 5 – Forecasting Results - Bradesco and Itau

	Bradesco			Itau		
	Log Loss	DM[SS]	DM[SSM]	Log Loss	DM[SS]	DM[SSM]
SS	1.761	-	-45.97	1.948	-	-42.64
SSM	1.742	45.97	-	1.932	42.64	-
EW	1.764	-9.165	-41.40	1.949	-3.079	-27.54
M_1	1.772	-21.72	-45.63	1.959	-17.74	-36.41
M_2	1.801	-61.97	-76.69	1.987	-51.86	-65.02
E	1.789	-48.61	-64.73	1.968	-27.22	-42.41

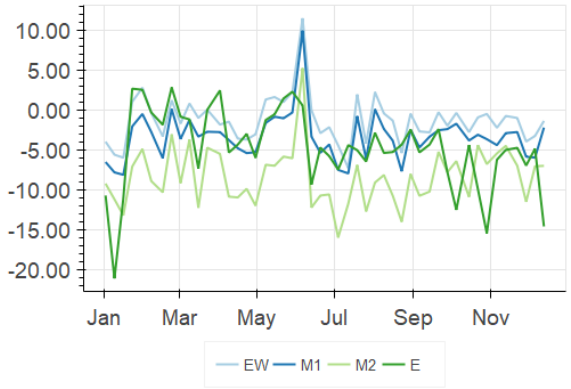
Table 6 – Forecasting Results - Petrobras and Vale

	Petrobras			Vale		
	Log Loss	DM[SS]	DM[SSM]	Log Loss	DM[SS]	DM[SSM]
SS	1.477	-	-45.62	2.027	-	-33.54
SSM	1.453	45.62	-	2.015	33.54	-
EW	1.480	-12.80	-47.44	2.023	7.942	-10.91
M_1	1.486	-28.81	-54.53	2.033	-7.249	-21.30
M_2	1.500	-56.73	-70.70	2.065	-43.57	-52.98
E	1.517	-81.27	-88.45	2.046	-21.21	-31.99

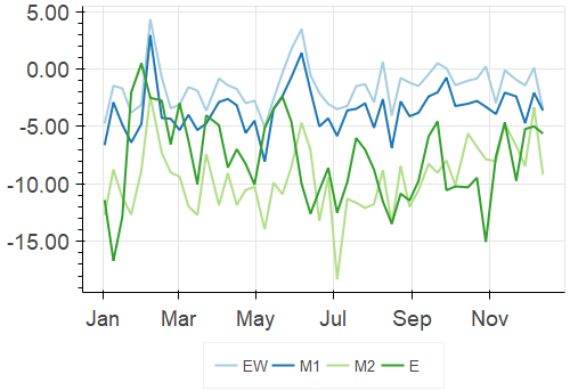
We also compute the DM statistic individually for each 5-day period. We show the results for the comparison with the SS model in figure 13 and the comparison with the SSM model in figure 14. We can see in the figures that all previously reported out-performances are consistent through most of the 5-days samples, with few points of exception.

Figure 13 – Weekly DM Score for SS Model

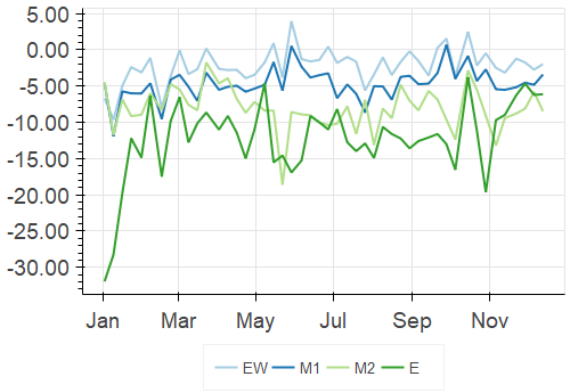
(a) Itau-Unibanco



(b) Bradesco



(c) Petrobras



(d) Vale

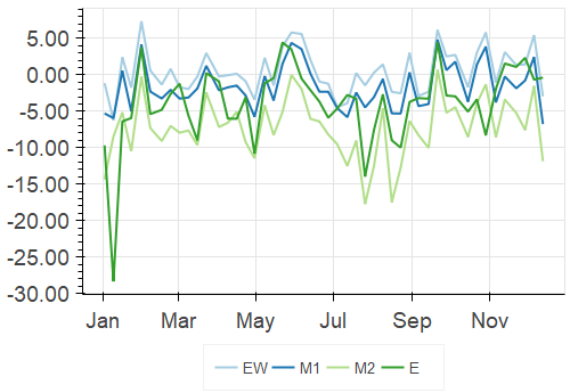
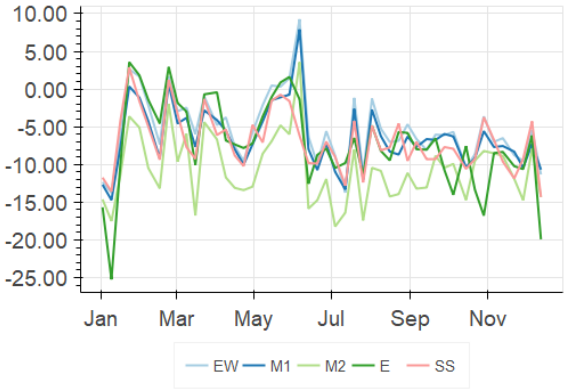
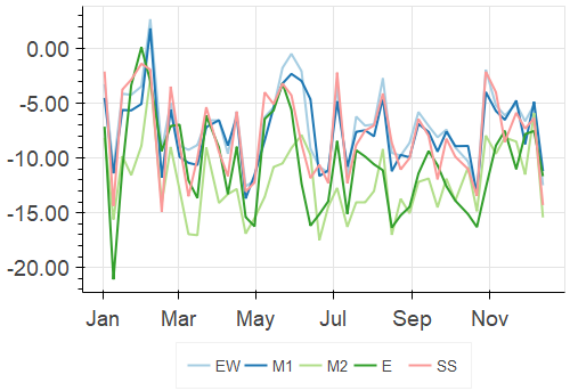


Figure 14 – Weekly DM Score for SSM Model

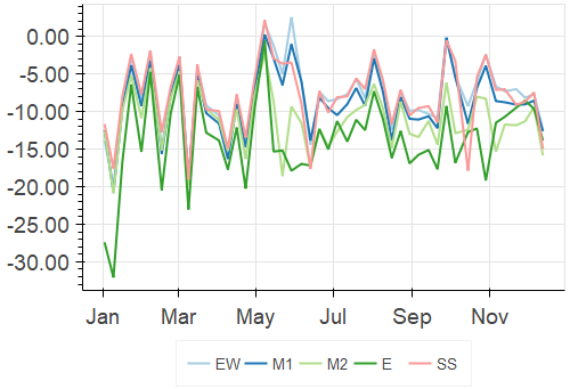
(a) Itau-Unibanco



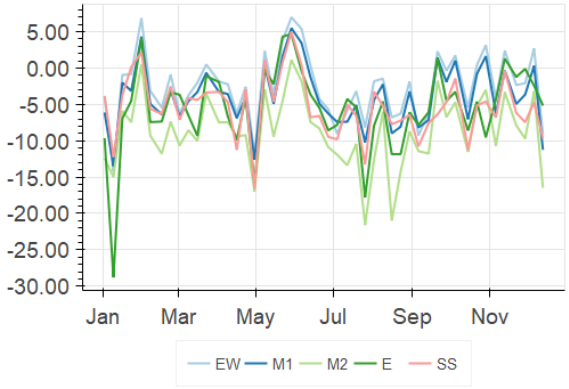
(b) Bradesco



(c) Petrobras



(d) Vale



1.9 Final Remarks

In this paper we extended the non-linear, non-Gaussian State Space model of Koopman, Lit and Lucas (2017) in order to jointly model the mean of the conditional density. We modeled the mean to have an auto-regressive functional form.

We estimated the model in two variants, without the mean (SS) and with mean (SSM). For estimating both models, we used a new estimation procedure, based on the Coordinate Descent algorithm, that minimizes the number of NAIS computations. The models were estimated for four stocks with a 10 seconds time interval, for every 5-days periods along the year of 2018. We found significant negative auto-regressive coefficients for the mean of all stocks and this sign of the estimates was consistent along all the estimates through all periods. Like in Koopman, Lit and Lucas (2017), we found a strong intraday seasonality pattern, although we also showed that this pattern changed slightly through the year. Using the models estimated during the year, we concluded that, on average, the volatility is high at the starting of the day and falls during the day.

We conducted an extensive walk-forward conditional density forecasting exercise. We showed that the SS model outperforms all other models for all stocks with exception of one stock (Vale), in which the EWMA model was superior. The same exercise showed that the new model, the SSM model, outperformed all other the models in forecasting performance. The results show that modeling the mean is important for forecasting the conditional mean and that there is a strong short term reversion for the stocks analyzed in the 10 seconds time frame.

2 Modelling Intraday Covariance

Abstract

In this paper we propose a new model for forecasting discrete high-frequency bi-variate conditional densities and covariance. The model is composed of two marginals using a modified Skellam distribution and a dynamic conditional Gaussian copula. The dynamics of both the volatilities and the correlation are modelled through state space models with a seasonality factor, which permits the measurement of the intraday seasonality for the covariance. We also estimate a Score Driven model following Koopman et al. (2018) and other empirical non-parametric models. By conducting an extensive walk-forward forecasting exercise we conclude that the new model outperforms both the empirical non-parametric predictors and the score-driven model for the forecasting the conditional bivariate distribution. We also conclude that the Score Driven outperforms all the empirical non-parametric models considered.

Keywords: time-varying copulas, dynamic discrete data, high frequency data; discrete price changes; importance sampling, score driven models, Skellam distribution, dynamic dependence

JEL Classification: C32, G11

2.1 Introduction

In the first paper we studied the dynamics of the conditional probability distributions for univariate high frequency intraday price changes. We modelled both the mean and the volatility, showing that it does have an intraday seasonal pattern. We showed that volatility is higher at the start of the day, falling during the day. Other works like Andersen and Bollerslev (1997) and Koopman, Lit and Lucas (2017) arrive at the same conclusion. We argued that one possible reason for such behavior is that information accumulates overnight and the market reacts to it on the starting of the trading session.

A natural extension of this work is to study how bivariate conditional probability distributions evolve during the day. As in the univariate case, the study of such distributions is of interest for exchanges, risk managers and market participants in general. One can question whether the asset correlations do have patterns like the volatility, whether they are higher or lower during the first trading hours.

Koopman et al. (2018) studies this issue, arriving at the conclusion that correlation is lower at the start of the trading session. The authors argue that one possible explanation is that a higher proportion of idiosyncratic information accumulates during the overnight as most firm-specific news are released when the market is closed. Allez and Bouchaud (2011) and Bibinger et al. (2019) also arrives at the same stylized fact, but differently from Koopman et al. (2018) and the present work, they used non parametric realized measures in their studies.

At the present work we propose a new method for estimating the bivariate intraday covariance at high frequency. We extend the model developed in the first paper by adding a dynamic Gaussian copula to the marginals modeled by state space in that paper. This dynamic copula itself relies on a correlation parameter, ρ_t, which is also modeled with a state-space dynamic that has a seasonal component. So we are modeling the high frequency bivariate distribution using modified Skellam distributions marginals, with mean dependency, and a dynamic Gaussian copula.

The new model is the main contribution of this paper. As the correlation has a parametric seasonal component in the model, we can estimate the mean seasonality in the model, contributing to the academic discussion on this seasonality. The model can also be compared with existing models in the literature by its conditional probability distribution predictions. In particular, we can compare it with the Score Driven model developed in Koopman et al. (2018) and simple non parametric predictors to assess its performance.

2.2 Modeling the Dynamic Correlation using Copulas

Let $y_t = (y_{1t}, \ldots, y_{nt}) \in \mathbb{Z}^n$ be a integer-valued n-dimensional vector representing the price changes for the n assets at time t in ticks. We represent the price changes by integers as we are focused in high frequency price changes. At high frequency the price changes are discrete as stocks are traded in multiples of a tick size. So we represent the price changes as integer multiples of the tick size for each stock.

Let $F_i(y_{it}|\mathcal{F}_{t-1}; \theta_{it}^m)$ be the time-varying conditional marginal cdf, where $\mathcal{F}_t := \{y_1, \ldots, y_t\}$ is the filtration (information set) at time t and θ_{it}^m collects all the i marginals parameters. In this paper F_i will always be the cdf of a (modified) Skellam distribution, but the framework of this section is general and could be applied to other marginal distributions.

The dependence structure y_t is modelled by a parametric n-dimensional copula function for each t.

$$C\left[F_1\left(y_{1t}|\mathcal{F}_{t-1}; \theta_{1t}^m\right), \ldots, F_n\left(y_{nt}|\mathcal{F}_{t-1}; \theta_{nt}^m\right)|\mathcal{F}_{t-1}; \theta_t^c\right] \tag{2.1}$$

where θ_t^c collects the parameters of the copula function. Observe the we allow θ_t^c to vary through time, which allow the study of how such parameters vary through the day. In the present paper we will focus on the Gaussian copula, so that the unique parameter is the correlation ρ_t.

Koopman et al. (2018) notes that the discrete copula defined just like above is not unique in the standard representation (see Sklar (1959)). The copula is only uniquely determined in the Cartesian product of the range of the cdf marginals. For example, if the above is applied to two Bernoulli trials with probability p as marginals, the copula would have uniquely determined values at $\{0, p, 1\} \times \{0, p, 1\}$. This problem with discrete copulas contrasts with the continuous copulas, that are uniquely determined at $[0, 1]^n$.

Let $\theta_t := (\theta_{1t}^m, \ldots, \theta_{nt}^m, \theta_t^c)$. With the definitions above we can calculate the joint probability mass function (pmf) by using the 'inclusion-exclusion' formula, obtaining:

$$p(y_t; \theta_t) = \sum_{j_1 \in \{0,1\}} \cdots \sum_{j_n \in \{0,1\}} (-1)^{j_1 + \ldots + j_n} C\left[F_1(y_{1t} - j_1; \theta_{1t}^m), \ldots, F_n(y_{nt} - j_n; \theta_{nt}^m); \theta_t^c\right] \tag{2.2}$$

In the bivariate case the equation (2.2) becomes:

$$\begin{aligned} p(y_t;\theta_t) = C\left[F_1(y_{1t};\theta^m_{1t}), F_2(y_{2t};\theta^m_{2t})\right] - C\left[F_1(y_{1t}-1;\theta^m_{1t}), F_2(y_{2t};\theta^m_{2t})\right] \\ -C\left[F_1(y_{1t};\theta^m_{1t}), F_2(y_{2t}-1;\theta^m_{2t})\right] + C\left[F_1(y_{1t}-1;\theta^m_{1t}), F_2(y_{2t}-1;\theta^m_{2t})\right] \end{aligned} \tag{2.3}$$

The formula in the equation (2.2) is the exact representation of the pmf of the model developed above, but it has two possible short-comings. First, it is cumbersome for high dimensions as the number of terms grow exponentially in the sum. This is not a problem for the bivariate case which is the focus of this paper, but it would be a problem in general. Second, it might not be numerically accurate if $F_i(y_{it}-1;\theta^m_{it})$ is too close to $F_i(y_{it};\theta^m_{it})$ for all i, as the formula makes the difference of such terms. This is an issue especially in calculating $\log(p(y_t;\theta_t))$ and $d\log(p(y_t;\theta_t))/d\theta_t$, and both are needed in what follows.

So we also consider an approximation of equation (2.2). Observe that when $F_i(y_{it}-1;\theta^m_{it})$ is close $F_i(y_{it};\theta^m_{it})$ we can use approximate the copula by its midpoint density in the implied integral and make a first order approximation obtaining the following approximation for equation (2.2).

$$\begin{aligned} p(y_t;\theta_t) \simeq \hat{p}(y_t;\theta_t) &= c(u^*_{1t},\ldots,u^*_{nt};\theta^c_t)\prod_{i=1}^{n}\left[F_i(y_{it};\theta^m_{it}) - F_i(y_{it}-1;\theta^m_{it})\right] \\ &= c(u^*_{1t},\ldots,u^*_{nt};\theta^c_t)\prod_{i=1}^{n} p_i(y_{it};\theta^m_{it}) \end{aligned} \tag{2.4}$$

where u^*_{it} is the midpoint of the marginal i cdf's, that is $u^*_{it} := [F_i(y_{it};\theta^m_{it}) + F_i(y_{it}-1;\theta^m_{it})]/2$, c is the copula density function and p_i is the probability density function for the marginal i.

As mentioned before, we consider only Gaussian copulas in this work. So the copula function is given by:

$$C(u;R) := \Phi_R\left(\Phi^{-1}(u_1),\ldots,\Phi^{-1}(u_n)\right) \tag{2.5}$$

where Φ_R is the multivariate Gaussian cumulative density function with correlation matrix R and Φ^{-1} is the inverse of the Gaussian univariate cumulative density function.

The copula density function is given by:

$$c(u;R) := \frac{1}{\sqrt{\det R}} \exp\left(-\frac{1}{2}\left(\Phi^{-1}(u_1),\ldots,\Phi^{-1}(u_n)\right)\left(R^{-1}-I\right)\left(\Phi^{-1}(u_1),\ldots,\Phi^{-1}(u_n)\right)^T\right) \tag{2.6}$$

In the bivariate case the Gaussian copula parameter θ_t^c is constituted of just one parameter, ρ_t. And the correlation matrix at each time t, R_t, is given by:

$$R_t = \begin{bmatrix} 1 & \rho_t \\ \rho_t & 1 \end{bmatrix} \tag{2.7}$$

2.3 Modeling Bivariate High Frequency returns using State Space Model

Now we develop the main model for this paper. Using the structure developed in the previous section, it suffices to specify the marginals and the dynamics for ρ_t. We will also specify the relevant parameters $\theta_{1t}^m,\ldots,\theta_{nt}^m,\theta_t^c$ that are specific to this model.

The marginal probability density specification follow the same specification of paper 1. So we have for each asset i:

$$y_{it}|\sigma_{it}^2 \sim p_i(y_{it};\mu_{it},\sigma_{it}^2,\gamma) \tag{2.8}$$

We write the marginal pmf i as a Modified Skellam distribution MSKII$(i,j,\mu,\sigma^2,\gamma)$:

$$p_i(y_t;j,l,k,\mu_{it},\sigma_{it}^2,\gamma) := \begin{cases} p_s(y_t;\mu_{it},\sigma_{it}^2) & \text{for } y_t \notin \{j,l,k\} \\ p_s(y_t;\mu_{it},\sigma_{it}^2)-\gamma\Delta/2 & \text{for } y_t \in \{j,l\} \\ p_s(y_t;\mu_{it},\sigma_{it}^2)+\gamma\Delta & \text{for } y_t = k, \end{cases} \tag{2.9}$$

where γ satisfies $2\min(p_s(i,\mu,\sigma^2),p_s(j,\mu,\sigma^2)) > \gamma\Delta > -p_s(k,\mu,\sigma^2)$ and p_s is the Skellam Distribution pmf, which is given by:

$$p_s(y;\mu,\sigma^2) := \exp(-\sigma^2)\left(\frac{\sigma^2+\mu}{\sigma^2-\mu}\right) I_{|y|}(\sqrt{\sigma^4-\mu^2}) \tag{2.10}$$

and $j = -1, l = 1, k = 0$. $I_{|y|}$ denotes the Bessel I Function with $|y|$ degrees of freedom. The equation for the mean chosen to be equal to the SSM model of paper 1,

$$\mu_{it} = \delta_i y_{i,t-1} \tag{2.11}$$

.

For each $i \in \{1, 2\}$ we model the stochastic process of σ_{it} by a long term expectation ($\overline{\sigma_i}$), a seasonality pattern (s_{it}) and a stochastic component. σ_{it} is given by

$$\begin{aligned} \sigma_{it}^2 &= \overline{\sigma_i}^2 s_{it} \exp(\alpha_{it}) \\ \alpha_{i,t+1} &= \phi_i \alpha_{i,t} + \eta_{i,t} \\ \eta_{it} &\sim \text{NID}(0, \sigma_{\eta,i})^2. \end{aligned} \tag{2.12}$$

We assume $|\phi_i| < 1$ in order to obtain stationarity. We also assume that η_{1t} is independent from η_{2t}.

Regarding the seasonality factor s_{it}, we use the spline specification as in the paper 1, using a parsimonious specification through cubic splines. The treatment is similar to the one used in Harvey and Koopman (1993) and Koopman, Lit and Lucas (2017). We write s_{it} as a zero-sum cubic spline function

$$s_{it} = \beta_i' W_t, \tag{2.13}$$

where W_t is calculated as described in Poirier (1973). β_i is a $K \times 1$ vector of parameters associated with $K + 1$ spline knots. In this paper we set $K = 3$ and set the knots for the spline at the times 10:00, 12:00, 13:30 and 17:00. These times reflect the market opening, start and end of lunch time and market closing.

Collecting all parameters for this model for the marginals we obtain that

$$\theta_{it}^m = \{\mu_{it}, \sigma_{it}, \gamma\} \tag{2.14}$$

and the model parameters for each marginal are $\{\overline{\sigma_i}, \delta_i, \phi_i, \sigma_{\eta,i}, \beta_i\}$.

Regarding ρ_t, we model it as a state space dynamic with a linear/Gaussian state space equation and a non-linear link function in order to accommodate for the restriction that

it must lie in the interval $[-1, 1]$. We also include a seasonality factor in the correlation structure. So we write:

$$\begin{aligned} \rho_t &= \tanh(\overline{\rho} + \alpha_{0,t} + s_{0,t}) \\ \alpha_{0,t+1} &= \phi_0 \alpha_{0,t} + \eta_{0,t} \\ \eta_{0t} &\sim \text{NID}(0, \sigma_{\eta,0})^2 \end{aligned} \tag{2.15}$$

where $s_{0,t}$ is given by:

$$s_{0t} = \beta_0' W_t \tag{2.16}$$

The tanh function guarantee that ρ_t lies in $[-1, 1]$. η_{0t_0} is assumed to be independent of η_{1t_1} and η_{2t_2} for all t_0, t_1, t_2. So, collecting all parameters for the copula, we obtain

$$\theta_t^c = (\rho_t) \tag{2.17}$$

and the model parameters related to the copula are given by $(\overline{\rho}, \phi_0, \sigma_{\eta,0}, \beta_0)$.

2.4 Modeling bivariate High Frequency returns using Score Driven Model

Koopman et al. (2018) cites three advantages for the score driven models. First, they possess information theoretic optimality properties (Blasques, Koopman and Lucas (2015)). Second, they have similar forecasting performance as their parameter driven peers, even when the latter are actually the true data generating process (Koopman, Lucas and Scharth (2016)). Third, the model's static parameters can be estimated in a straightforward way using maximum likelihood methods.

Now we derive our score-driven model. The model also builds on the Dynamic Correlation model of section 2.2. We will specify the relevant parameters $\theta_{1t}^m, \ldots, \theta_{nt}^m, \theta_t^c$ directly and map them to the Gaussian copula and Skellam distribution parameters.

We write the score-base update of θ_t as

$$\theta_{t+1} = \omega + A\nabla_t + B\theta_t \tag{2.18}$$

where ω is a constant vector and A and B are constant matrices. ∇_t is given by

$$\nabla_t = \frac{\partial \log p(y_t; \theta_t)}{\partial \theta_t} \tag{2.19}$$

This specification of the score dynamics follows Creal, Koopman and Lucas (2011), Creal, Koopman and Lucas (2013) and Harvey and Luati (2014). We follow Koopman et al. (2018) and consider the case where A and B are diagonal, so that each component for θ_t has its own dynamics. So we have

$$\begin{aligned} A &= \text{diag}(a) \\ B &= \text{diag}(b) \end{aligned} \tag{2.20}$$

for constant vectors a and b.

The analytical formula for the score function ∇_t coordinates is given by equations (2.20) and (2.21):

$$\nabla_{it}^m = \frac{\partial \log p(y_t; \theta_t)}{\partial \theta_{it}^m} = \sum_{j_1 \in \{0,1\}} \cdots \sum_{j_n \in \{0,1\}} \frac{(-1)^{j_1 + \ldots + j_n}}{p(y_t; \theta_t)} \frac{\partial C(u_{1t}, \ldots, u_{nt}; \theta_t^c)}{\partial u_{it}} \frac{\partial u_{it}}{\partial \theta_{it}^m} \tag{2.21}$$

$$\nabla_t^c = \frac{\partial \log p(y_t; \theta_t)}{\partial \theta_t^c} = \sum_{j_1 \in \{0,1\}} \cdots \sum_{j_n \in \{0,1\}} \frac{(-1)^{j_1 + \ldots + j_n}}{p(y_t; \theta_t)} \frac{\partial C(u_{1t}, \ldots, u_{nt}; \theta_t^c)}{\partial \theta_t^c} \tag{2.22}$$

This equation makes more apparent the need for use the approximation of equation (2.4). Both numerator and denominator can be too close to zero when $F_i(y_{it} - 1; \theta_{it}^m)$ is close to $F_i(y_{it}; \theta_{it}^m)$. In that case working with (2.4) is preferable. The corresponding derivatives for that equation are given by equations (2.23) and (2.24):

$$\nabla_{it}^m = \frac{\partial \log p_i(y_{it}; \theta_{it}^m)}{\partial \theta_{it}^m} \tag{2.23}$$

$$\nabla_t^c = \frac{\partial \log c(u_{1t}^*, \ldots, u_{nt}^*; \theta_t^c)}{\partial \theta_t^c} \tag{2.24}$$

These equations are more numerically stable, as they do not suffer from the same problem of dividing too small numbers.

Now that we described the general rule we followed to calculate the score-driven update for θ_t, we can specify the exact equations for the parameters of the score-driven model. We parameterized our final model variables σ_{it} and ρ_t by using equations (2.25) and (2.26):

$$\sigma_{it}^2 = \exp(\theta_{it}^m) \tag{2.25}$$

$$\rho_t = \frac{\theta_t^c}{\sqrt{1 + (\theta_t^c)^2}} \tag{2.26}$$

We calculated the exact likelihood and exact derivatives by using equations (2.3), (2.21) and (2.22) whenever numerically feasible, that is, when:

$$\|F_i(y_{it}; \theta_{it}^m) - F_i(y_{it} - 1; \theta_{it}^m)\|_\infty > \varepsilon_0 \tag{2.27}$$

for a fixed parameter ε_0. Otherwise we use expressions (2.23) and (2.24) whenever the expression in equation (2.27) is less then ε_1. If the value of the expression was in the interval $(\varepsilon_0, \varepsilon_1)$, a linear combination of both expressions was used. This procedure was taken to guarantee both numerical stability and continuity for the score function. The estimation results were robust to changes in ε_0 and ε_1.

Now we show how to calculate the expressions (2.21), (2.22), (2.23) and (2.24) for our specific model.

First, notice that $p(y_t; \theta_t)$ can be calculated using equations (2.3) and (2.5), when calculating in the exact form, and can be calculated by (2.4) and (2.6) in the approximated form. Regarding the derivatives for the copula function, we also note that it can be written as a conditional copula:

$$\frac{\partial C(u_{1t}, u_{2t}; \theta_t^c)}{\partial u_{1t}} = P(U_{2t} \leq u_{2t} | U_{1t} = u_{1t}) \tag{2.28}$$

So in our case of Gaussian copula the equation (2.28) becomes

$$\frac{\partial C(u_{1t}, u_{2t}; \theta_t^c)}{\partial u_{1t}} = \Phi\left(\frac{\Phi^{-1}(u_{2t}) - \rho_t \Phi^{-1}(u_{1t})}{\sqrt{1-\rho_t^2}}\right) \tag{2.29}$$

The derivative of the bivariate Gaussian cdf with respect to the correlation ρ is given by

$$\frac{\partial \Phi_2(x, y; \rho)}{\partial \rho} = \frac{1}{2\pi\sqrt{1-\rho^2}} \exp\left(-\frac{x^2 - 2\rho xy + y^2}{2(1-\rho^2)}\right) \tag{2.30}$$

And now substituting $x = \Phi^{-1}(u_{1t})$, $y = \Phi^{-1}(u_{2t})$ and $\rho = \rho_t$ and applying the chain rule we obtain:

$$\frac{\partial C(u_{1t}, u_{2t}; \theta_t^c)}{\partial \theta_t^c} = (1+\theta_t^c)^{-3/2} \frac{\partial \Phi_2}{\partial \rho}(\Phi^{-1}(u_{1t}), \Phi^{-1}(u_{2t}); \rho_t) \tag{2.31}$$

Lastly, the derivative for the marginal Skellam cdfs can be calculated by cumulative sum the derivatives for the pmfs, which have closed expression. Using the widely known formula for the derivative of the Bessel I function we obtain:

$$\frac{\partial u_{it}}{\partial \sigma_{it}^2} = \exp(-\sigma_{it}^2) \sum_{\nu=-\infty}^{y_{it}} \left[\left(\frac{\nu}{\sigma_{it}^2} - 1\right) I_{|\nu|}(\sigma_{it}^2) + I_{|\nu+1|}(\sigma_{it}^2)\right] \tag{2.32}$$

And by the chain rule we can calculate the last expression we need:

$$\frac{\partial u_{it}}{\partial \theta_{it}^m} = \sigma_{it}^2 \frac{\partial u_{it}}{\partial \sigma_{it}^2} \tag{2.33}$$

2.5 Data

Like in paper 1, our dataset is formed by intraday prices for four Brazilian Stocks: Petrobras (PETR4), Vale (VALE3), Itau-Unibanco (ITUB4) and Bradesco (BBDC4) for the entire year of 2018. These stocks are among the most liquid Brazilian stocks. The data used in this paper consists of the closing trading prices for the intraday interval of 10 seconds obtained from B3 exchange by using Thomson Reuters Datascope service. Taking all four stocks together, our dataset consists of more than 2.3 billion prices.

The 10 seconds time interval was also used in (Koopman et al. (2018)). The 10 seconds time frame is more convenient for the multivariate analysis as it raises the probability of the joint event of simultaneous trading for the assets considered in the analysis, as argued in Koopman et al. (2018). This is the reason we chose the 10 seconds interval.

It worth noting that even using the 10 seconds time frame (instead of the 1 second) we still have to deal with a lot of missing values, as the stocks do not necessarily trade every 10 seconds interval. Regarding this issue we also take the same approach of Koopman et al. (2018), by only updating the model when trading activity occurs on any of the stocks.

We consider all prices ranging from the market opening at 10:00 to the market closing at 17:00. We model the intraday price changes, so we discard the overnight price changes in the cases where we estimate the model through more than one day. We apply a data-cleaning procedure before the analysis, in order to clean for exchange reporting errors, as recommended by Brownlees and Gallo (2006).

For descriptive statistics on the dataset refer to the data section of the paper 1.

2.6 Estimation Procedure

In this section we describe the model estimation procedure. Like in paper 1, we estimated the models by maximum likelihood.

We estimated the two models for every 5 consecutive business days periods ('weekly'), for all 'weeks' of 2018. As the models involve seasonality with the period consisting of one day, estimating with more than one day in the sample should help identifying the difference between the seasonality and the state space factors of the volatility process. In this respect we differ from Koopman, Lit and Lucas (2017), that estimated the models with sample size equal to one day. So we have about 100,000 data points for each estimation.

2.6.1 State Space Covariance Model

For the State Space Model we estimated the parameters for each marginal in separate, using only the data for that marginal, and then estimated the covariance model.

The log-likelihood function is calculated by the equations (2.3) and (2.4). For each interaction the same random numbers are used in order to ensure that the sampling error will not affect the calculus of the derivatives.

To maximize the log-likelihood function we apply a version of the Coordinate Descent Algorithm. The reason behind this choice is that the parameters that affect the measurement equation have derivatives that are easier to calculate and the coordinate descent also makes possible to run the NAIS algorithm less times.

The whole algorithm generally converges fast, taking just a few iterations at the outer loop of the Coordinate Descent algorithm. One advantage of the proposed algorithm is that the NAIS procedure is only needed at the step 2, and this step only have two parameters to estimate. Another advantage is the possibility of using analytic expression for the derivative of the log-likelihood function at the step 3.

2.6.2 Score Driven Covariance Model

All the parameters of the Score Driven model are estimated jointly by maximum likelihood. The log-likelihood function is also calculated by the equations (2.3) and (2.4), but the θ_t is updated according to equation (2.18). The resulting log-likelihood function was then optimized used the BFGS method, with the derivatives estimated numerically.

2.7 Estimation Results

Now we describe the estimation results for both models. In what follows we call C the State Space Covariance Model and CS the Score Driven Covariance Model. Like on paper 1, we start by describing the descriptive statistics of the coefficient estimated obtained by the rolling estimation procedure previously described.

2.7.1 Parameter Estimates

In what follows we write the coefficients for the seasonality for the C model as $\beta_0 = (\beta_0^0, \beta_1^0, \beta_2^0)$.

The statistics for the model C parameter estimates is shown in table 7. We observe that ϕ for both models are close to 1, as we would expect. The coefficient estimates for both pairs are quite different, however. The ρ estimates shows that Bradesco and Itau were much more correlated than Vale and Petrobras on average. This makes sense because Itau and Bradesco are in the same sector, they are both retail banks. Vale and Petrobras are in distinct sectors as Vale is a Mining company and Petrobras is an oil company. As in the case of the estimates on paper 1, the $\sigma_{\eta,0}$ are higher in the cases that the estimates of ϕ are far from one.

The seasonality coefficients show some similarity, as both have a negative β_0^0, a β_2^0 close to zero and the β_1^0 is the highest seasonality coefficient.

Table 7 – Descriptive Statistics for C Parameter Estimates

	ρ	ϕ	$\sigma^2_{\eta,0}$	β_0^0	β_1^0	β_2^0
Bradesco & Itau-Unibanco	0.31	0.96	0.0059	-0.012	0.040	0.0070
	(0.092)	(0.11)	(0.025)	(0.093)	(0.036)	(0.023)
Vale & Petrobras	0.12	1.0	9.6e-05	-0.032	0.025	0.0033
	(0.060)	(0.0072)	(0.00022)	(0.064)	(0.032)	(0.019)

The statistics for the model CS parameter estimates are shown in tables 8 and 9. In this model the mean estimates for both pairs of the correlation AR coefficient (b_2) are really close to one - different from the previous case. This is also the case for the AR coefficients associated with the marginal volatility processes (b_0 and b_1)

Table 8 – Descriptive Statistics for CS Parameter Estimates (w and a)

	w_0	w_1	w_2	a_0	a_1	a_2
Bradesco & Itau-Unibanco	0.0036	0.0053	0.00013	0.073	0.074	0.016
	(0.0020)	(0.0026)	(0.00085)	(0.026)	(0.02)6	(0.010)
Vale & Petrobras	0.0060	0.0018	0.00014	0.075	0.069	0.012
	(0.0031)	(0.0017)	(0.00044)	(0.022)	(0.020)	(0.0044)

Table 9 – Descriptive Statistics for CS Parameter Estimates (b and θ^0)

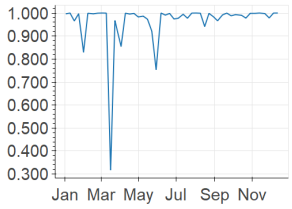
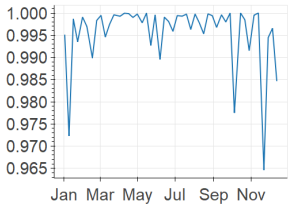
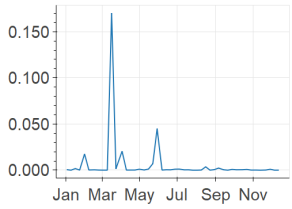
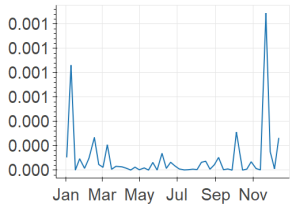
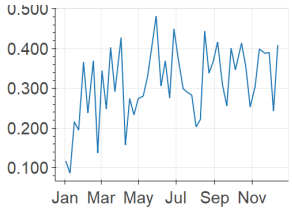
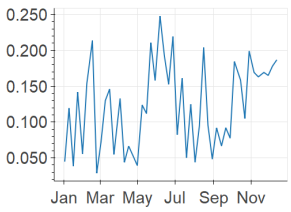
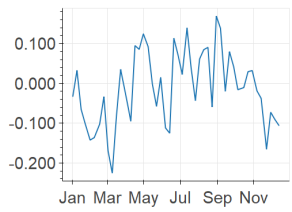
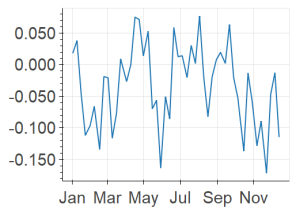
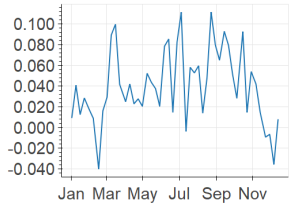
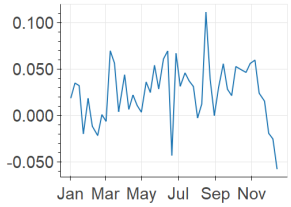
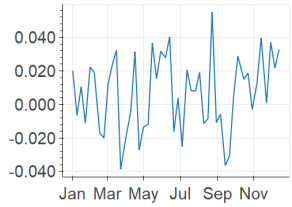
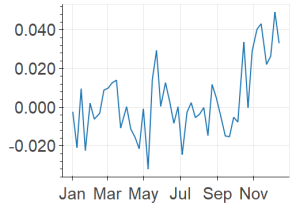
	b_0	b_1	b_2	θ_0^0	θ_1^0	θ_2^0
Bradesco & Itau-Unibanco	1.0	1.0	1.0	1.6	2.0	0.49
	(0.0013)	(0.0013)	(0.0017)	(0.20)	(0.32)	(0.20)
Vale & Petrobras	1.0	1.0	0.99	2.2	1.2	0.42
	(0.0011)	(0.00096)	(0.0080)	(0.48)	(0.30)	(0.28)

In the rest of this subsection we show graphs of all the point estimates for the coefficients for both models. Each point in each graph correspond to the estimate for the variable in one model. Table 10 shows the parameter estimates for the C model.

As the tables with the descriptive statistics already show, the estimate for ϕ is higher for the pair Vale & Petrobras than for Itau-Unibanco & Bradesco. Looking at the graphs actually ϕ is also quite high for Itau-Unibanco & Bradesco most of the time, but at few estimation points the value of ϕ drops significantly. This might have been caused by idiosyncratic events on these specific dates. As happened in the estimates in the paper 1, the estimates for the volatility of the state space innovation ($\sigma^2_{\eta,0}$ here) oscilate jointly with the estimates for ϕ. Whenever the estimates for ϕ falls, the estimates for $\sigma^2_{\eta,0}$ rises.

The estimates for ρ oscilated through the year with no interesting dynamics. Regarding the seasonality, the changes in β_0, β_1 and β_2 suggest that the correlation seasonality changed slightly during the year and at the end of the year the correlation appeared the be relatively higher at the end of the day and lower at the middle of the day.

Table 10 – Estimates for parameters in C model

	Itau & Bradesco	Vale & Petrobras		Itau & Bradesco	Vale & Petrobras
ϕ			$\sigma^2_{\eta,0}$		
ρ			β_0		
β_1			β_2		

Tables 11 and 12 shows the parameter estimates for the CS model.

Table 11 – Estimates for parameters a and b in CS model

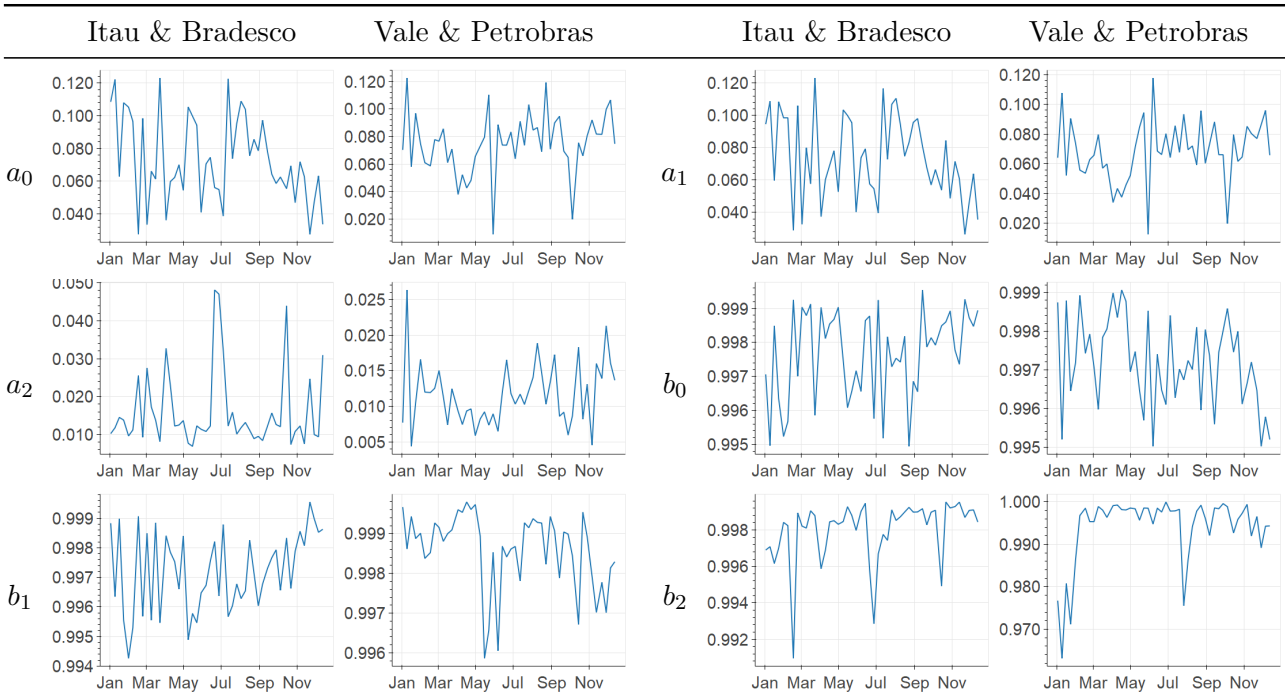
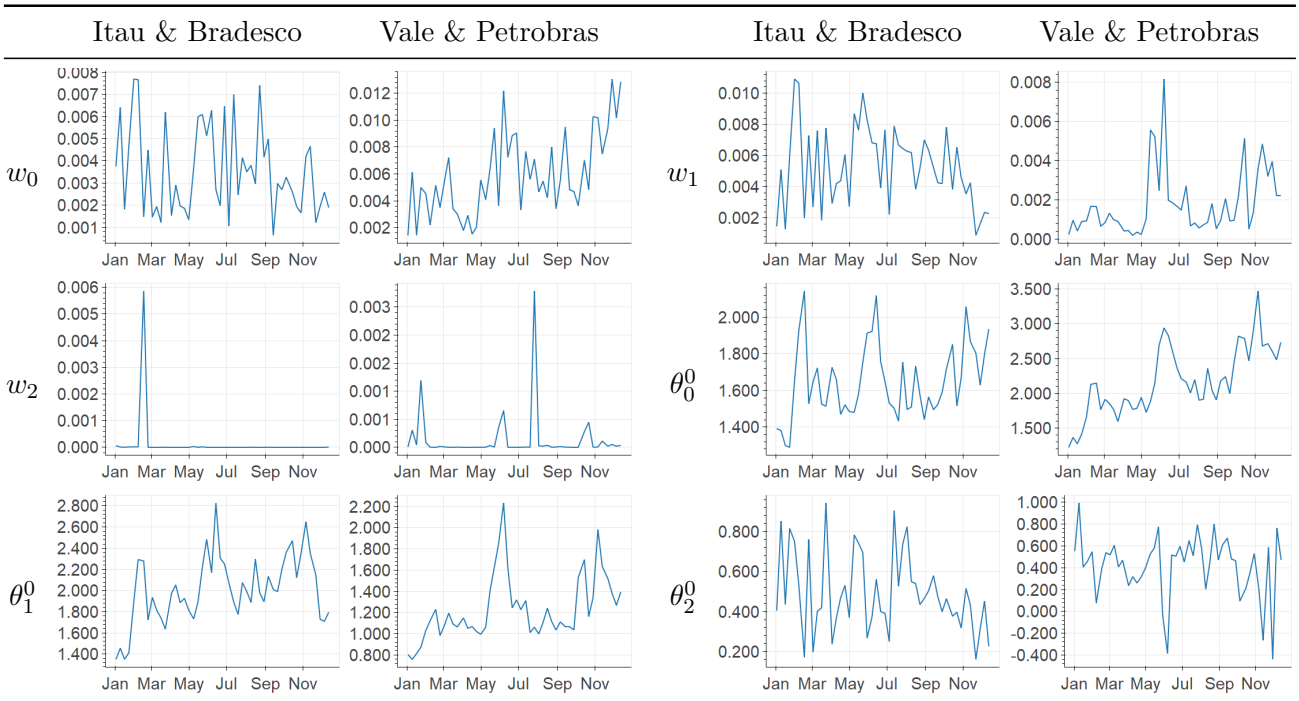


Table 12 – Estimates for parameters w and θ^0 in CS model



2.7.2 Seasonality

In this sub-section we analyze the descriptive statistics for the volatility seasonality.

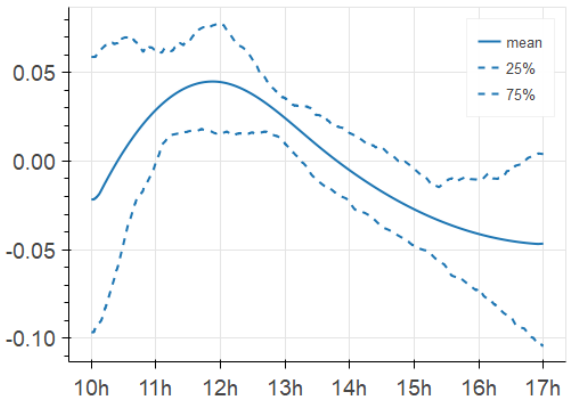
Figure 18 shows the seasonality for the C model, showing the mean and two percentiles

for the log of the seasonality factor for each time of the day. Observe that for both pair of stocks the correlation starts low, rises during the day and falls in the end of the day. This effect is mentioned in Koopman et al. (2018)).

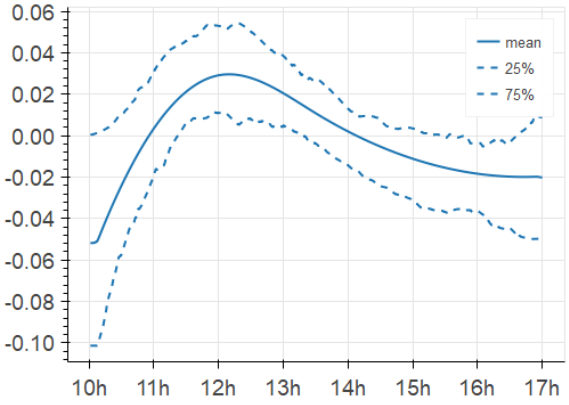
As mentioned before, one possible explanation is that most idiosyncratic news related to stocks are released when the market is closed.

Figure 15 – Average Seasonality for model C in 2018

(a) Itau-Unibanco & Bradesco



(b) Vale & Petrobras



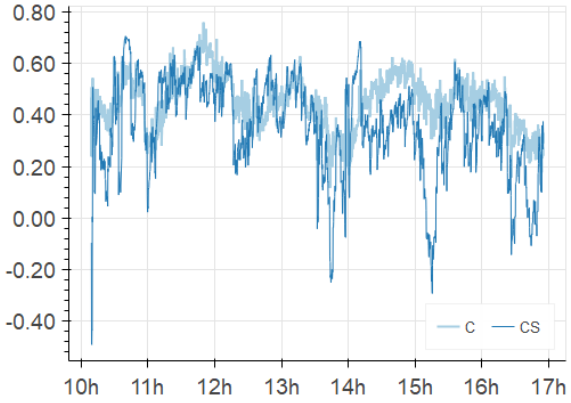
2.7.3 Filtered Correlation Forecasts

We now show an example of the filtered correlation obtained by the Bootstrap Particle Filter applied for the C model and by applying the appropriate transformation to θ_t^c for the CS model. Figure 16 shows the path of the filtered correlation for the day July 6, 2018 for both models. Observe that the correlation starts low, rises during the day and falls in the end of the day, as predicted by the mean seasonality in the previous sub-section. This effect clear on the Vale and Petrobras pair for this day.

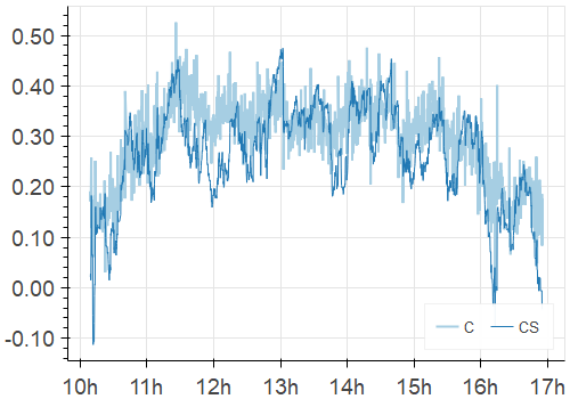
It also worth mentioning that altough the dynamics of both models are different during this day (with the correlation of the C model being smoother), the movements and levels are similar. This is a robustness check on the procedure of both models, as they are totally different.

Figure 16 – Filtered Correlation Forecast for July 6, 2018

(a) Itau-Unibanco & Bradesco



(b) Vale & Petrobras



2.8 Forecasting Performance Comparison

To assess the forecasting performance of the two models we conduct a forecasting exercise for all 5-days periods of 2018. For the forecast of time t and for all models, we use only data up to $t-1$. To compare the performance we estimate the joint bivariate pmf by using empirical forecasting models for the copula and marginal parameters. The models are:

M_1: σ_t is estimated non-parametrically by a moving average process using a rolling window of the past 90 returns. μ and γ are set to zero

M_2: σ_t is estimated non-parametrically by a moving average process using a rolling window of the past 900 returns. μ and γ are set to zero

E: the pmf of y_t is determined by the empirical probabilities of y_t on the fit sample.

We estimate all the models for each 5-day period of 2018 and calculated the pmf forecasts for the next 5-days iteratively by filtering the states while keeping the parameters fixed. To assess the forecasting performance we follow Koopman, Lit and Lucas (2017) and used the log-likelihood loss function, which is simply minus the log of the pmf obtained by each model.

In the table 13 we show the main statistics for the forecasting performance comparison of the models. For each stock pair we report the mean log-loglikelihood loss function value for each time on the whole sample (for all periods of 5-day taken together). We also report the Diebold-Mariano Statistics comparing all models to the C Model (DM[C] column) and comparing all models to the CS model (DM[CS] column). In both cases a negative number means that the model in the columns outperforms the model on the row and a positive number means the opposite.

The Diebold-Mariano statistic follow a standard normal distribution. So our exercise suggest that C outperforms all other models for forecasting the pmf by a large margin for both pairs of stocks. The CS model also outperforms all other models for both pairs of stocks (with the exception of the C model), but with a lower margin.

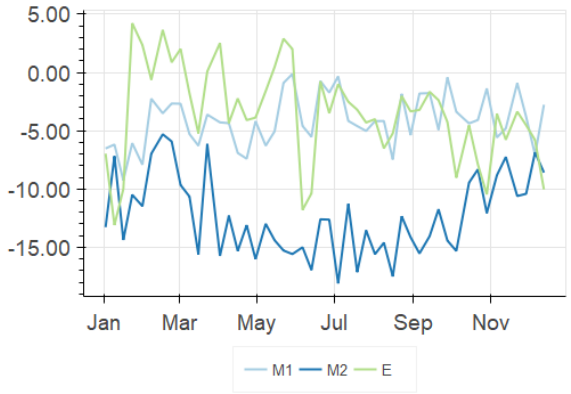
Table 13 – Forecasting Results

	Bradesco - Itau			Vale - Petrobras		
	Log Loss	DM[C]	DM[CS]	Log Loss	DM[C]	DM[CS]
C	3.435	-	217.6	3.269	-	99.47
CS	3.666	-217.6	-	3.496	-99.47	-
M_1	3.680	-225.8	26.10	3.511	-105.7	-33.54
M_2	3.740	-220.7	85.98	3.563	-122.0	-85.24
E	3.690	-297.1	20.33	3.542	-121.8	-46.12

Finally, we also calculate the DM statistic individually for each 5-day period. We show the results for the comparison with the CS model in figure 17 and the comparison with the C model in figure 18. We can see in the figures that all previously reported out-performances are consistent through all the 5-days samples.

Figure 17 – Weekly DM Score comparison for CS Model

(a) Itau-Unibanco & Bradesco



(b) Vale & Petrobras

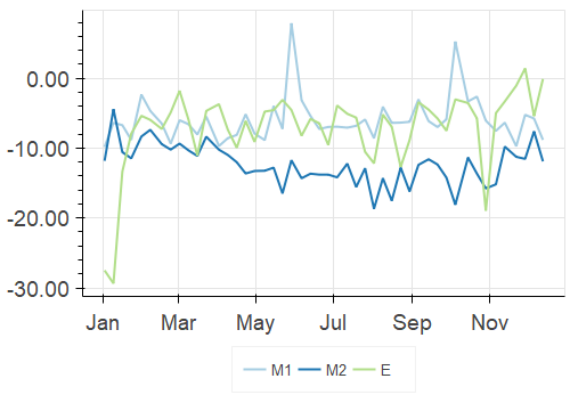
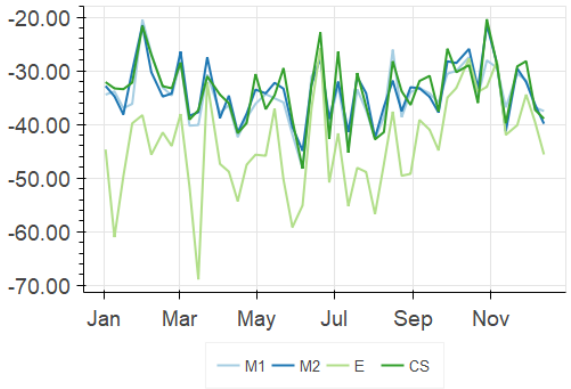
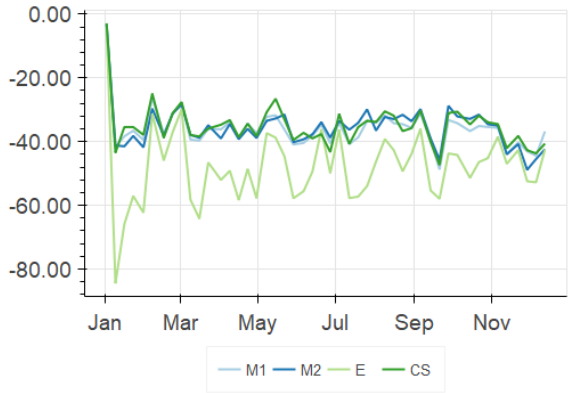


Figure 18 – Weekly DM Score comparison for C Model

(a) Itau-Unibanco & Bradesco



(b) Vale & Petrobras



2.9 Final Remarks

In this paper we developed a new model for forecasting discrete high-frequency bi-variate conditional densities and covariance. The model is composed of two marginals modelled like in the paper 1 (State Space model using a modified Skellam distribution) and a dynamic Gaussian copula with the correlation dynamics also modeled by a state space model. The model was estimated by using the estimation procedure described in paper 1.

The model for the correlation was composed of a state space dynamic factor and a seasonality factor. The model was estimated for two pairs of stocks for the entire year of 2018. Analyzing the estimates for the seasonality factor coefficients through the year, we were able to show that on average the correlation is lower on the starting and on the ending of the day and higher in the middle of the day. This stylized fact was already mentioned in Koopman et al. (2018), but the authors used an entirely different methodology.

We conducted and extensive forecasting exercise comparing the forecasting performance for the proposed model for the conditional bivariate densities with other models, including the Score Driven model proposed in Koopman et al. (2018). The tests were conducted on a walk-forward basis for the entire year of 2018. We found that the proposed model had superior forecasting performance using the log-likelihood loss when compared to all other models, including the score Driven. We also found that the Score Driven was superior to all other models, except for the new proposed one.

3 Forecasting Intraday Volatility and Densities using Deep Learning

Abstract

In this paper we develop a new model for forecasting high-frequency intraday conditional discrete return densities and volatility by using deep learning. Specifically, we model the conditional distribution by a modified Skellam distribution with the mean following an auto-regressive specification and train feedforward neural networks in order to generate predictions for the underlying high-frequency volatility. Four different specifications are tested including different set of features and parameters. Then we conduct a comprehensive walk-forward forecasting experiment in order to compare the forecasting accuracy for the proposed models. All the proposed models beat the empirical non-parametric forecasting rules considered. The new forecasting procedure also provides better out-of-sample forecasts compared to all Space State Models considered in the thesis. We also conclude that new variables have predictive power for the volatility process: the bid-ask spread, high-low interval spread and the volume traded. According to our model estimates, all these variables appear to have a positive non-linear S shaped relation with volatility.

Keywords: volatility models; high frequency data; discrete price changes; deep learning; neural networks; Skellam; non-Gaussian time series models; dynamic discrete data

JEL Classification: C32, C45, G11

3.1 Introduction

As argued in paper 1, modelling and predicting asset returns distributions is a principal research goal in finance. In the last decades this topic received a growing attention, especially regarding the measurement of the second moment - the volatility. And in more recent years the topic expanded to intraday volatility measuring and forecasting. There are many studies using intraday prices to forecast daily volatility (see Andersen et al. (2001),Barndorff-Nielsen and Shephard (2002), and Hansen and Lunde (2006)) and, more recently, papers using this data to analyse and forecast the intraday price dynamics itself (see Engle and Sokalska (2012) and Koopman, Lit and Lucas (2017)).

In the first paper we modelled the intraday price dynamics by using the Modified Skellam distribution for the conditional density of the intraday returns, while using non-linear non-Gaussian state space models for modelling the volatility dynamics, in a similar way to Koopman, Lit and Lucas (2017). The characteristics that motived this choice are that volatility is unobservable, changes through time and possess a clustering pattern. This features are well-known for a long time for daily volatility (see Shephard (2005) for a review).

In this paper we also model intraday conditional return densities with focus on volatility dynamics modelling, but we emphasize the non-linearity of the volatility response to its own past and usual explanatory variables. In order to capture such features we model the volatility dynamics through deep neural networks. Previous works have shown evidence for non-linearities for the volatility process not captured by traditional Garch-Like processes which can be captured by such networks. They also show the ability for such networks in dealing with other non-traditional explanatory variables for predicting volatility. See Donaldson and Kamstra (1997), Hajizadeh et al. (2012), Kristjanpoller, Fadic and Minutolo (2014), Bucci (2019)).

Our work contributes to this literature first by extending the deep neural network modelling to the intraday high-frequency time frame. To the best of authors knowledge there is no previous comprehensive work analyzing the intraday volatility forecast at a time frame lower than 1 minute and considering Skellam-like distributions. We believe that actually this time frame is more suitable for neural networks than the daily time frame, as the distributions are non-Gaussian and the data is abundant and noisy.

Our approach is more flexible than the State-Space and Garch -like approaches, in the sense that is easier to include new explanatory variables as we do not need to specify a 'correct' functional form for the input variable's impact on the volatility process. The

neural network approach is motived by its ability to universally approximate non-linear functions, so we learn the functional form during the estimation. Building on this advantage we include new explanatory variables (bid-ask spread, high-low spread and volume) and test for their impact on the forecasts. We analyze the sensivity of changes of these input variables in the forecasts and analyzing the changes in the predictive power added for the inclusion of these variables.

Lastly, we use the neural network not only to forecast volatility, we actually forecast conditional densities in a similar way to Koopman, Lit and Lucas (2017), Koopman et al. (2018). And we compare the results with models derived from from the approach of such papers, which are tougher benchmarks than the usual Garch-like models. We show the ability for the networks for forecasting the conditional densities and volatilities, testing for the forecasting performance in a comprehensive walk-forward forecasting study.

The rest of this paper is organized as follows. Section 2 gives a brief overview on the Deep Learning approach, describing neural networks, the optimization process and design issues. Section 3 describes the base model and its variants. Section 4 describes the data. Section 5 gives details on the Training (optimization) procedure. Section 6 and 7 describes the estimation results and results related to the forecasting exercise. Section 8 concludes.

3.2 Deep Learning Approach: an Overview

According to Goodfellow, Bengio and Courville (2016), machine learning is essentially a form of applied statistics with increased emphasis on the use of computers to statistically estimate complicated functions and a decreased emphasis on proving confidence intervals around these functions. Machine learning is more concerned about making models to forecast, rather than learning about parameters.

The central challenge in machine learning is that models must perform well on new, previously unseen inputs — not just those on which our model was trained. The ability to perform well on previously unobserved inputs is called generalization. A model has its hyper-parameters chosen in a validation set, is typically trained (fitted) to a training set (in-sample) and tested on the test set (out-sample). The objective is thus designing the model that has the lowest test (generalization) error.

All the emphasis in machine learning is on making models that have low test error. There is no need to attain a maximum likelihood estimate, for instance, and for many machine learning algorithms there are no guaranties that any in-sample estimates are optimal in any sense. The terminology 'optimizing' or 'fitting' is substituted by 'learning' to emphasize

such difference.

3.2.1 Neural Networks and Deep Learning

Deep learning is the part of machine learning that use models that rely on Deep Artificial Neural Networks. We now motivate the use of such procedure.

One might presume that learning a nonlinear function requires designing a specialized model family for the kind of nonlinearity this function presents. Our interest in feedforward networks with hidden layers for this paper is based on the fact that they provide a universal approximation framework. The universal approximation theorem (see Hornik, Stinchcombe and White (1989) and Cybenko (1989)) states that a feedforward network with a linear output layer and at least one hidden layer with any "squashing" activation function (like the logistic sigmoid activation function) can approximate any Borel measurable function between finite-dimensional spaces with any non-zero amount of error, provided that the network has enough hidden units. The derivatives of the feedforward network can also approximate the derivatives of the function arbitrarily well (see Hornik, Stinchcombe and White (1990)).

The existence of such approximating neural network does not mean we can feasibly attain such network by training a network design with just one layer. The approximating network might involve a too large width to be feasible. Goodfellow, Bengio and Courville (2016) argue that for many problems increasing the depth of the network results in better approximating function, reducing the width needed. More complex relations can be modeled this way with less network nodes. The authors compare this structure with a prior that the function being modeled is composed of chain composition of simpler transformations. And in practice deeper modern neural networks have shown a better generalization error.

3.2.2 Optimization

As mentioned, the objective of machine learning is different from classical statistics and this reflects on how optimization is done in machine learning, especially for Deep Learning. Finding the global minimum on the training set is unfeasible for most neural network designs, as the optimization problem involved is complex. And this is not actually the objective, which is in fact generating the model with the lower generalization error possible. So training a deep learning model is significantly reducing a loss function L_1 in the training sample with the hope that this procedure will reduce other loss function L_2 in the test sample.

The typical optimization procedure in Deep Learning is described in the algorithm below.

The typical loss function consists of a sum through samples. At each step a 'mini-batch' of samples is taken from the training set (possibly at random) and the gradient for the loss is calculated using only these samples. If all the training sample is used on the 'mini-batch', we call the optimization method 'batch' gradient descent (otherwise it is called 'minibatch' gradient descent).

Then the model parameters are updated using the gradient times some 'Learning Rate', which might be a fixed parameter (in case of the Stochastic Gradient Descent) or may be adaptive (in the case of Adam or RMSprop, for instance). This procedure is repeated for a number of times, until the training set is passed by the procedure for a maximum number of times. The 'number of epochs' is such number of passages. Some procedures also specify some early stopping criteria, often related to the loss function applied in some other test set.

Algorithm 1: Basic Deep Learning Optimization Algorithm

Result: Trained Network parameters θ
input : learning rate ϵ_k (or adaptive method for generating it)
input : Method for calculating gradients from minibatchs, θ and previous gradients
input : Maximum number of epochs: maxEpochs
input : Starting θ: θ_0
$k \leftarrow 0$;
$\theta \leftarrow \theta_0$;
while $k < maxEpochs$ **do**
  while *can sample from training set* **do**
    Sample m minibatch samples from training set (without replacement);
    Compute gradient g ;
    apply update: $\theta \leftarrow \theta - \epsilon_k g$;
    if *early stopping criteria met* **then**
      return θ;
    end
  end
  $k \leftarrow k + 1$;
  reset training set returning all samples to it;
end

3.3 Model

In this section we develop the base model used in the rest of this paper, its variants, the features and the network design.

Let $y_t \in \mathbb{Z}$ be the integer-valued variable representing the price changes for the asset at time t. We represent the price changes by integers as we are focused in high frequency price changes. At high frequency the price changes are discrete as stocks are traded in multiples of a tick size. So we represent again the price changes as integer multiples of the tick size for each stock.

Let $p(y_t|\mathcal{F}_{t-1};\theta_t)$ be the time-varying conditional pmf's, where $\mathcal{F}_t := \{X_1, \ldots, X_t\}$ is the filtration (information set) at time t and θ_t collects all the pmf's parameters. The X_t is a vector of variables that might include y_t. Our main goal in this paper is to find p, X_t and θ_t that provide best out of sample forecasts for the distribution of y_t. That is, we are searching for the best conditional density forecasts, which are also functions of the volatility forecasts.

3.3.1 Base Model

Like in Koopman, Lit and Lucas (2017) and Koopman et al. (2018), we consider only the case where p is given by the Modified Skellam Distribution. As shown in the first paper, this distribution has a good match for the intraday return's distribution and have a parameterization that turns easier to study volatility. So we have $p(y_t|\mathcal{F}_{t-1};\theta_t) = p_{II}(y_t; i, j, k, \mu_t, \sigma_t^2, \gamma)$, where p_{II} is given by

$$p_{II}(y_t; i, j, k, \mu, \sigma^2, \gamma) := \begin{cases} p_s(y_t; \mu, \sigma^2) & \text{for } y_t \notin \{i, j, k\} \\ p_s(y_t; \mu, \sigma^2) - \gamma\Delta/2 & \text{for } y_t \in \{i, j\} \\ p_s(y_t; \mu, \sigma^2) + \gamma\Delta & \text{for } y_t = k, \end{cases} \tag{3.1}$$

where γ satisfies $2\min(p_s(i, \mu, \sigma^2), p_s(j, \mu, \sigma^2)) > \gamma\Delta > -p_s(k, \mu, \sigma^2)$ and p_s denotes the Skellam pmf. As in the paper 1 we set $(i, j, k) = (-1, 1, 0)$. The Skellam pmf is given by

$$p_s(y; \mu, \sigma^2) := \exp(-\sigma^2) \left(\frac{\sigma^2 + \mu}{\sigma^2 - \mu}\right) I_{|y|}(\sqrt{\sigma^4 - \mu^2}), \tag{3.2}$$

where $I_{|y|}$ is the Bessel I Function with $|y|$ degrees of freedom.

Now we need only to specify the functional forms of σ_t and μ_t. All models considered in this paper, including the ones used for comparison follow the structure described so far. And this includes the models in paper 1. The functional for for μ_t is also the same of that paper. Thus μ_t is given by:

$$\mu_t = \delta y_{t-1} \tag{3.3}$$

We model σ_t by a nonlinear function of X_t that will be approximated by a deep Feed Forward Neural Network. So the functional form of σ_t is simply

$$\sigma_t = F(X_{t-1}) \tag{3.4}$$

where F function represents the Neural Network. In the Machine Learning nomenclature, X_t represents the features we consider in our model. The features themselves might be generated by another model, making a Hybrid model.

3.3.2 The Features

We follow Donaldson and Kamstra (1997), Hajizadeh et al. (2012), Kristjanpoller, Fadic and Minutolo (2014) and use a Hybrid model, using outputs from another model as a feature for the neural network. In particular we used the output of a simple EWMA model, with its λ parameter estimated in-sample. The EWMA model is given by

$$\hat{\sigma}_t^2 = \lambda y_{t-1}^2 + (1-\lambda)\hat{\sigma}_{t-1}^2 \tag{3.5}$$

we also denote EWMA forecast for time t by ew_t, that is, $ew_t := \hat{\sigma}_t^2$.

We also include y_t and y_t^2 as features. The idea for including y_t^2 is that the simple EWMA process might be unable to capture all the (possibly nonlinear) impact of the innovation y_t^2, even if it does already include such input. The y_t is included to capture asymmetric reactions of the volatility process regarding the sign of the return that have long being documented in the daily volatility forecasting literature (see Glosten, Jagannathan and Runkle (1993) and Engle and Ng (1993)).

Following Harvey and Koopman (1993), Koopman, Lit and Lucas (2017) and the discussion on paper 1, we include the seasonality splines as features (W_t). We proceed exactly as in

the paper 1, generating the curves W_t according to Poirier (1973). But instead of using fixed coefficients β like in equation (1.6), the splines join the features of the neural network. So they might interact in a non-linear way with all other features in determining the volatility forecast. As in the case of paper 1 we considered $K = 3$ for generating W_t, and the four resulting spline knots are fixed at the times 10:00, 12:00, 13:30 and 17:00, representing the market open, lunch time and market close. So W_t has dimension 3 and we denote the three resulting features as W_1, W_2, W_3.

Last set of features is composed of 3 new variables not considered in previous papers, which are not traditionally considered in volatility models. The first is the closing bid-ask spread, denoted by ba. When the volatility rises market liquidity providers including market makers, traders and HFT algorithms tend to be less aggressive in attending the market final orders. In the case of high frequency data, a large bid-ask spread also mechanically affect the volatility by increasing the bid-ask bounce effect. In any case this variable is expected to affect the volatility and is included for this reason. See Wyart et al. (2008) for a reference on this effect.

The second variable form this set is the High-Low interval spread for the last 10 seconds, denoted by hl. There is a vast literature relating the daily high-low interval to the volatility, and though our time frame is short (10 seconds), this variable is also expected to have some predictive value for the volatility. The intuition is simple: the more volatile the market, the wider this range should be. This is true for a continuous Brownian Motion (i.e. by the Reflection Principle) and is also expected for the discrete setting we are interested. See Yang and Zhang (2000), for instance, for the usage of this data for volatility forecasting in the daily time frame.

The last variable we consider is the volume of trade for the last 10 seconds, denoted by v. The volume of trade is related to the intensity of the market activity and is reported to rise/fall leading and lagging volatility rises and falls (see Xu, Chen and Wu (2006)). The relation between volume and price changes is so relevant that some authors argue that intraday bar time series (like ones considered in this paper) are better analyzed by making regular intervals in terms of volume traded instead of time in order to make the return distributions more close do iid (see Prado (2018)). There are also data vendors that offer the data formed this way.

All the features above are z-score normalized before entering the Neural Network input. The parameter for such normalization (mean and standard deviation) are estimated in-sample and kept unchanged for the test-sample.

3.3.3 Network Design

As the objective is to approximate an arbitrary non-linear function on the features, and there are just a few of them, we use a fully connected feedforward neural network. This type of network is characterized by the fact that information flows from the input layer, passing to hidden layers and arriving at the output layer without any cycles. Fully connected means that between any two layers all nodes are connected.

Following the modern deep learning practice, we choose a large depth (number of layers) and width (neurons per layer). We use 20 layers, each with 10 neurons. As the other hyper-parameters for the model, these parameters were tested on a validation, set prior to the data used for estimating the models. They are not necessarily the best hyper-parameters for a forecasting network, but in our tests the forecasting capabilities do not change much for large networks.

As the network is fully connected, this design means that we have 2000 weight parameters to estimate and 20 bias parameters. Although it might seem that are too many parameters, each training sample we use for this paper consists of about 20.000 data points.

Networks with a larger depth are usually harder to train, but are able to capture more complex relations between the input variables. Depending on the activation function, the depth can make problems associated with vanishing gradients worse (Goodfellow, Bengio and Courville (2016)) and are more computationally expensive - which was a problem in the past, but not nowadays.

In order to compensate for the difficulty in training these deep networks, we make three design choices:

1. A relatively high width (10 neurons)

2. Use the Leaky ReLU activation function

3. Use a customized initiation procedure

The first choice, picking a high width, alleviates the problem of vanishing gradients by simply adding more terms to the output expression, adding more paths so that the information can flow from the input layer to the output layer. So this choice makes less likely that the dead neurons (neurons with too low gradient) affect the entire network. See Zagoruyko and Komodakis (2016) for a discussion on the effects of network width on

training.

The second choice tackles the vanishing gradients problem directly, by using an activation function whose gradient never vanishes. The LeakyReLU activation function (Maas, Hannun and Ng (2013)) is defined as:

$$\text{LeakyReLU}(x) := \begin{cases} x & x \in (0, \infty), \\ 0.01x & \text{otherwise} \end{cases} \tag{3.6}$$

The gradient of LeakyReLU never vanishes, attaining a minimum value that is strictly greater than zero. This is different than the more standard activation functions tanh, sigmoid and ReLU. Worth mentioning that even tough the gradient never vanishes completely, it can be as low as 1e-40 on the composition of the 20 layers, so that although this choice alleviates the problem, we can still have dead neurons in practice.

The third choice refers to the initiation of the weights for the network, which will be described in greater detail in section 5. In short we initiated an entire path of weights from the input feature *ew* to the output as ones, setting all bias coefficients as zero, except for the last one, which was set to a positive number. This choice of initiation made the starting parameters of the network resemble *ew*, making the gradients not vanishing and the computations for the log-likelihood loss derived from (3.1) more stable.

Let $I = 8$ be the number of features, $L = 20$ the number of layers and $H = 10$ the width of the network. The final definition for the neural network, that is, for the function F is given by the following equations:

$$\begin{aligned} h_t^{(0)} &= \text{leakyReLU}(W^{(in)} X_t + b^{(in)}) \\ h_t^{(i)} &= \text{leakyReLU}(W^{(i)} h_t^{(i-1)} + b^{(i)}), \text{ for } i \in \{1, \dots, L\} \\ \sigma_{t+1}^2 &= F(X_t) = |W^{(out)} h_t^{(L)} + b^{(out)}| \end{aligned} \tag{3.7}$$

where $W^{(in)}$ is $(I \times H)$, $W^{(i)}$ is $(H \times H)$,$W^{(out)}$ is $(H \times 1)$, $b^{(in)}$ is $(H \times 1)$, $b^{(i)}$ is $(H \times 1)$ and $b^{(out)}$ is (1×1).

3.3.4 Model Variants

In order to measure the impact of different components of our model, we estimate a total of 4 versions of the model, including the final model and 3 simplified versions.

The table 14 resumes the characteristics of the models. The most basic is NN_0, in which we force γ and δ to zero and we use the same features used in paper 1 (that is, we exclude *ba*, *hl* and *v*). So this model use the standard Skellam distribution, with no model for the mean ($\mu_t = 0$).

The next is NN_v, which is like NN_0, except that we include the variables *ba*, *hl* and *v*. In $NN_{v,\gamma}$ we allow γ to be different than zero. The final model is $NN_{v,\gamma,\mu}$, in which we also model the mean using equation (3.3).

Table 14 – Model Variant description

model	Features	γ	δ
NN_0	all except *ba*,*hl*, and *v*	$\gamma = 0$	$\delta = 0$
NN_v	all	$\gamma = 0$	$\delta = 0$
$NN_{v,\gamma}$	all	estimated	$\delta = 0$
$NN_{v,\gamma,\mu}$	all	estimated	estimated

3.4 Data

Like in paper 1, our dataset is formed by intraday prices for four Brazilian Stocks: Petrobras (PETR4), Vale (VALE3), Itau-Unibanco (ITUB4) and Bradesco (BBDC4) for the entire year of 2018. These stocks are among the most liquid Brazilian stocks. The data used in this paper consists of the closing trading prices for the intraday interval of 10 seconds obtained from B3 exchange by using Thomson Reuters Datascope service. Taking all four stocks together, our dataset consists of more than 2.3 billion prices.

It is worth noting that even using the 10 seconds time frame (instead of the 1 second) we still have to deal with a lot of missing values, as the stocks do not necessarily trade every 10 seconds interval. Regarding this issue we also take the same approach of Koopman et al. (2018), by only updating the model when trading activity occurs. So we do not pad missing price changes with 0 price changes as done in Koopman, Lit and Lucas (2017). We consider not trading as different event than trading at the last price and we also want to make the analysis in this paper comparable to the analysis on the rest of the thesis.

We consider all prices ranging from the market opening at 10:00 to the market closing at 17:00. We model the intraday price changes, so we discard the overnight price changes in the cases where we estimate the model through more than one day. We apply a data-cleaning procedure before the analysis, in order to clean for exchange reporting errors, as recommended by Brownlees and Gallo (2006).

For descriptive statistics on the dataset refer to the data section of the paper 1.

3.5 Training Procedure

In this section we describe in more detail the training procedure applied for the networks, along with the estimation of other parameters. The network parameters and the parameters γ and δ are estimated together using the procedure described in this section.

3.5.1 Loss Function and Regularization

As the objective is to forecast a conditional density along with the volatility, we choose the base loss function to be the log-likelihood loss for both the training and testing procedures. In order to avoid over-fitting we add L^2 norm penalization to the loss function, commonly know as weight decay. So the loss function expression is given by

$$L(y) = -\sum_t \log\left(p(y_t|\mathcal{F}_{t-1};\theta_t)\right) - \frac{\alpha}{2}\|W\|_2^2 \tag{3.8}$$

where $W = \{W^{(in)}, W^{(out)}\} \cup \{W^{(i)}\}_{i\in\{1,\dots,L\}}$ is the vector containing all weights in the model.

The coefficient for the L^2 penalty is a model hyper-parameter and was obtained using a presample in 2017 by trial and error. We used $\alpha = 100$.

3.5.2 Optimization Algorithm

The Stochastic Gradient Descent (SGD) and its variants are probably the most used optimization algorithms for machine learning in general and for deep learning in particular. It consists in estimating the gradient using a mean of the minibatchs gradient estimates over a batch of data. The parameters are then updated using a hyper-parameter called learning rate times the gradient estimate. So it is a first order gradient descent method where the estimates are updated using just a sub-sample of data.

The main difficulty in using SGD is appropriately setting the learning rate hyper-parameter as its choice greatly determines the predictive power of the trained model (according to Goodfellow, Bengio and Courville (2016)). If the learning rate is too high the algorithm makes the objective function to decay faster but becomes non-robust, failing to keep the objective function decaying. If it is too low, the optimization simply gets too much time to finish. And as one can see by the Newton's Method, it is better to have a lower learning rate when the curvature is higher and conversely.

In order to avoid such difficulties associated with the SGD method, we use the Adamax algorithm. It is an adaptive method introduced in Kingma and Ba (2014) in which the gradients are re-scaled by using the inverse of the weighted L^∞ norm applied to the previous calculated gradients. Adamax is a version of the Adam algorithm introduced in the same paper, which in its usual form uses the L^2 instead of L^∞.

Regarding the batch size, we used all the sample at once for each step of the optimization, so we actually used a 'batch gradient descent' method. As discussed in greater detail in the appendix, we used GPU for training the networks so having a greater batch size did not impact significantly the processing time. By using all data the gradients are calculated with more precision. We used 2000 epochs for optimizing each network, with no early stopping.

This above algorithm performed well in our pre-sample tests and was selected.

3.5.3 Initialization

Initialization is an important and not well understood part of the optimization process in deep learning. The starting values for the neural network heavily affects the results of the optimization process. In our case, the loss function used both for training and testing may not be numerically stable depending on the network outputs, which increases the concern regarding the initialization.

One standard initialization for the weights, which is default in many Deep Learning frameworks is given in equation (3.9)

$$W_{i,j}^{(k)} \sim U\left(-\frac{1}{\sqrt{m_k}}, \frac{1}{\sqrt{m_k}}\right) \tag{3.9}$$

where $k \in \{\text{in}, 1, \ldots, L, \text{out}\}$ is the layer, i, j are matrix coordinates and m_k is the number of columns for the matrix $W^{(k)}$.

We tested this initialization in our procedures and the results were unstable. For concluding the forecasting exercise in the next section we need to estimate more than a thousand different networks, so we need a robust optimization procedure.

So we changed this initialization scheme replacing the first column for matrices $W_{i,j}^{(k)}$ by the vector $(0, \ldots, 0, 1)^T$. This makes the output for the first neuron for all layers to be exactly the first feature, ew. We also set the bias b to zero for all layers except for the first one, which had a positive value in order to make sure that the value is positive. This is

needed because ew is a z-score, and could be negative and we need to keep the value in domain where leakyRelu is the identity function.

With this new initialization scheme the output for the network resembles the output for the EWMA model at the starting of the optimization and the optimization became much more robust, converging for the solutions presented in this paper with just one optimization trial.

Regarding the parameters outside the neural network, we initialized both γ and δ as zero.

3.6 Training and Forecasting Exercise

We applied the procedure in the previous section for training all the model variants in table 14. As done in paper 1, we estimated the models for all 5 consecutive business days periods of 2018, testing the model forecasts for the following 5 business days. The 5 days period was chosen so that the daily volatility seasonality can be better captured by the models.

In order to compare for the forecasts of the neural network models we used estimates for the models used in the paper 1:

SSM: State space model described in paper 1 with $\mu_t = \delta y_{t-1}$

SS: State space model described in paper 1 with $\mu_t = 0$

EW: σ_t is estimated by an EWMA process: $\hat{\sigma}_t^2 = \lambda y_{t-1}^2 + (1-\lambda)\hat{\sigma}_{t-1}^2$. λ is estimated using the fit sample (previous week). μ and γ are set to zero

M_1: σ_t is estimated non-parametrically by a moving average process using a rolling window of the past 90 returns. μ and γ are set to zero

M_2: σ_t is estimated non-parametrically by a moving average process using a rolling window of the past 900 returns. μ and γ are set to zero

E: the pmf of y_t is determined by the empirical probabilities of y_t on the fit sample.

The results for the exercise are described in the next two sections. The computational details of the implementation is described at the appendix.

3.7 Estimation Results

In this section we describe the main in-sample results for the training exercise described in last section. We compare the results obtained in paper 1

3.7.1 Parameter Estimates

Now we describe the estimation results for the most complete model $\text{NN}_{v,\gamma,\mu}$. The model was estimated for 48 different 5 business days periods for all the four equities in our list.

We report the descriptive statistics for the parameter estimates at the table 15. For each equity / parameter we report the sample mean for the parameter estimates on the first row and the standard deviation in parenthesis on the second row. Notice that these statistics refer to estimates in different samples, so the mean and standard deviations are only indicative of common estimated values through the year.They should not be interpreted as estimates and significance test statistics.

Table 15 – Descriptive Statistics for the NN parameter estimates

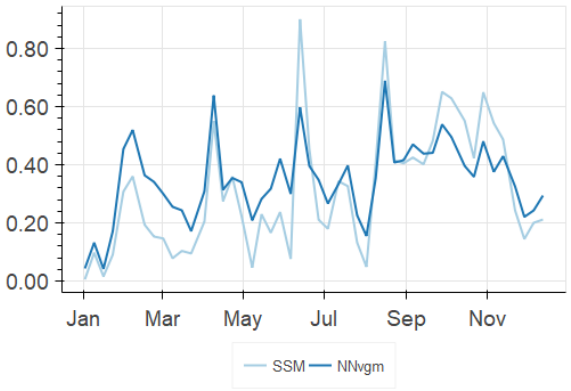
	Bradesco	Itau-Unibanco	Petrobras	Vale
δ	-0.200	-0.186	-0.225	-0.173
	(0.034)	(0.036)	(0.075)	(0.041)
γ	0.125	0.346	-0.103	0.307
	(0.111)	(0.136)	(0.089)	(0.174)

We notice that δ is negative for all stocks, exactly as in the paper 1. This implies a negative dependency of the mean on previous returns, which is the expected value as noted in paper 1. We notice also that γ is only negative for Petrobras, also like in paper 1. In fact the estimates show different but similar values when compared to the estimates for SSM and SS models.

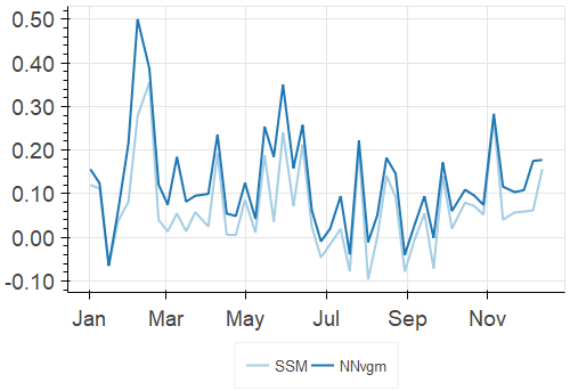
The similarities are more prominent when we compare the point estimates for each sample. Figures 19 and 20 shows the estimates for γ and δ respectively for both $\text{NN}_{v,\gamma,\mu}$ and SSM models.

Figure 19 – Estimates for γ

(a) Itau-Unibanco

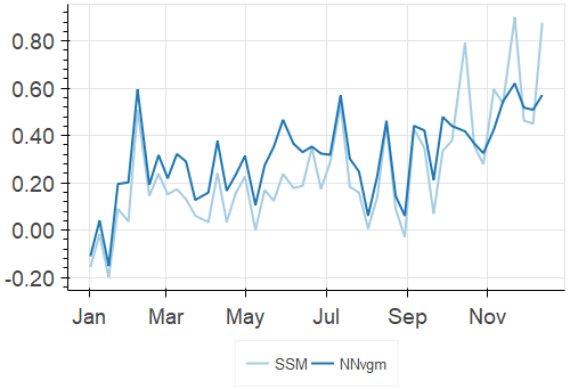


(b) Bradesco



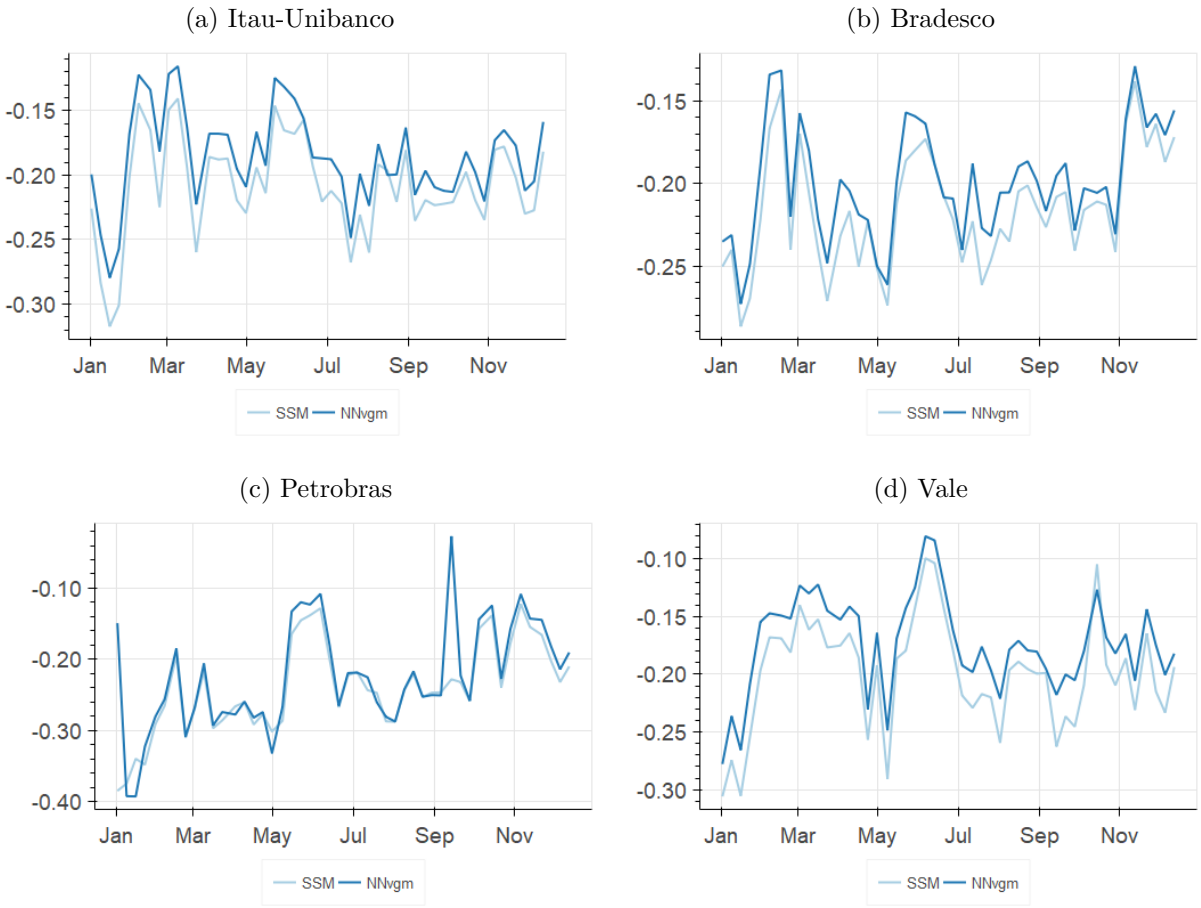
(c) Petrobras

(d) Vale



On both cases the lines plotted are similar for both models, although the level for the SSM model tended to be a little bit lower than for the neural network. This similarity shows the robustness of both estimation procedures, as the estimation procedures are totally different and the models are in fact also different.

Figure 20 – Estimates for δ



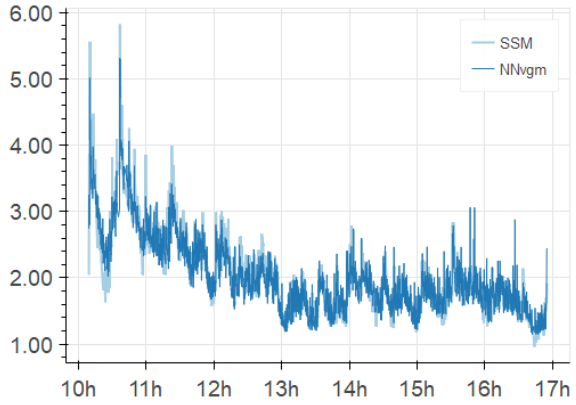
3.7.2 Filtered Volatility Example

We now compare volatility estimates from the Neural Network model $\text{NN}_{v,\gamma,\mu}$ and the SSM model. For the SSM model we used the Bootstrap Particle Filter in order to obtain the filtered forecasts. Figure 23 shows the path of the filtered volatility for the day July 6, 2018 for both models. Observe that the volatility decreases during the day as predicted by the mean seasonality in paper 1.

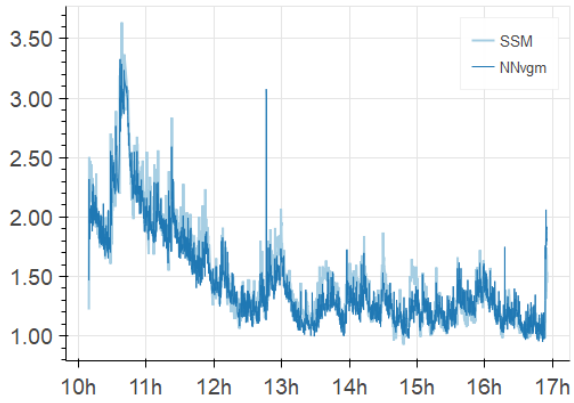
Note also that the volatility estimates for both models are strikingly similar, which is again a robustness check on the procedure of both models - as the models and procedures are totally different in both case. The volatility forecast for the Neural network appears to be little less volatile than the volatility forecast for the SSM model.

Figure 21 – Filtered Forecast Volatility for July 6, 2018

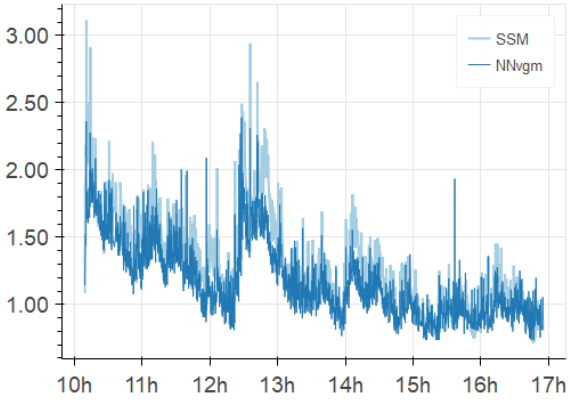
(a) Itau-Unibanco



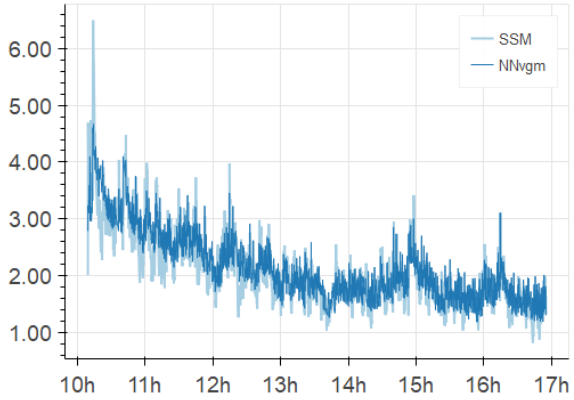
(b) Bradesco



(c) Petrobras



(d) Vale

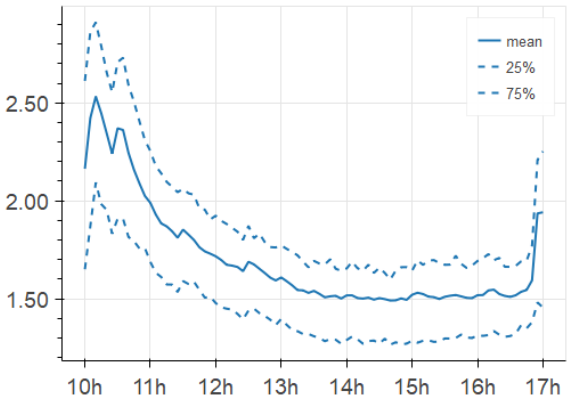


3.7.3 Intraday Seasonality

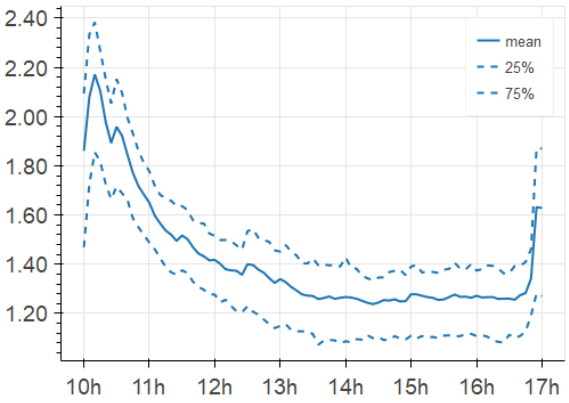
In this sub-section we analyze the descriptive statistics for the volatility seasonality. Figure 22 depicts the seasonality for the $\text{NN}_{v,\gamma,\mu}$ model, showing the mean and two percentiles for the volatility predicted by the model for each time of the day in the entire sample. Observe that for all stocks have a sharp decrease in the mean volatility predicted from the opening to the end of the day. This effect is also documented in Koopman, Lit e Lucas (2017). As stated in paper 1, one possible reason is that on the starting of the trading session there is more information for the markets to digest, including all the overnight news and economic releases, that are more common in the morning.

Figure 22 – Average Seasonality for 2018

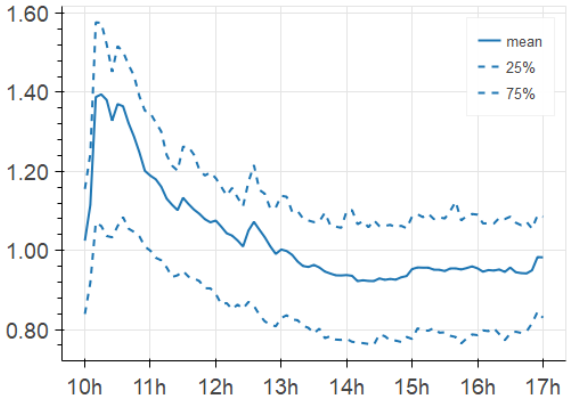
(a) Itau-Unibanco



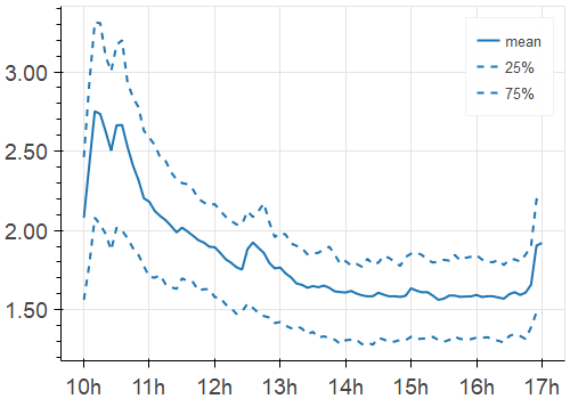
(b) Bradesco



(c) Petrobras



(d) Vale



The concept show in the figures is different however than the one show in paper 1, figure 11. In that paper we plotted the theoretical seasonality arriving from the seasonality splines estimated in that model. Now we show percentiles for the model prediction directly. So we have more variability in the sample now, as the volatility varies not only because of the seasonality - the variability is also related to shocks. The curves are also less smooth beacuse there is no spline structure behind this estimates.

We can see in this figure an interesting pattern not captured by the splines: A spike in the volatility at the very ending of the trading session. This is actually the close auction, that happens on the last 5 minutes of the trading session for Brazilian equities. The splines are not able to capture this because of the smoothness imposed by such structure.

3.7.4 Non-linear Sensivity Analysis

As discussed in section 2, Deep Learning method is focused in building models that are good for forecasting. It is usually hard to get a deeper understanding on how the networks

deal with each input feature, especially because the output of the network might be a highly non-linear function of the inputs. And the inputs interact with themselves inside the network, making interpretations harder.

Besides such difficulties, our objective in this section is to get a better understanding on how the trained networks react to changes in the features, on average. We want to answer questions about what to expect in terms of volatility when the bid-ask spread rises, or if there is any detected relation between volume and volatility.

We make two exercises. First, we shock all input variables by adding one standard deviation, one at a time keeping the other constant, for the whole training sample. Then we compute the changes in the prediction for the volatility for each stock, for each sample time. Table 16 reports the descriptive statistics for these sensibilities, reporting means and standard deviations aggregating through time. The standard deviations are reported below the means, in parenthesis.

Table 16 – Standardized Sensivity to shock in variables

	ew	y	y^2	ba	hl	v	W_0	W_1	W_2
Itau-Unibanco	0.30	-0.0023	0.078	0.13	0.090	0.036	0.082	0.028	-0.012
	(0.12)	(0.021)	(0.036)	(0.044)	(0.028)	(0.033)	(0.060)	(0.033)	(0.027)
Bradesco	0.26	-0.0039	0.060	0.095	0.061	0.022	0.067	0.021	-0.011
	(0.083)	(0.022)	(0.033)	(0.051)	(0.025)	(0.025)	(0.042)	(0.022)	(0.026)
Vale	0.35	-0.0094	0.083	0.11	0.092	0.076	0.12	0.051	-0.0016
	(0.13)	(0.024)	(0.037)	(0.059)	(0.033)	(0.054)	(0.088)	(0.038)	(0.030)
Petrobras	0.20	-0.0092	0.032	0.033	0.049	0.053	0.036	0.0098	-0.0023
	(0.11)	(0.015)	(0.021)	(0.019)	(0.031)	(0.036)	(0.038)	(0.026)	(0.014)

We observe that, on average, the models estimated point to a positive relation between the ew variable and the volatility output of the network, with a one standard deviation shock in that variable changing the volatility forecast between 0.20 and 0.35 depending on the stock.

The relation with y is negative on average, which is the expected sign for this sensibility, but with a low average value and a high dispersion. So the networks do not appear to be highly sensible to this variable, at least for a shock keeping all other variables constant. But as the relations modelled are non-linear, this input feature can interact with others leveraging its effects.

The relation with y^2 is clearly positive for all stocks, but with a impact below ew. The variables ba, hl and v all have positive effects in the output volatility, on average, for all stocks. As discussed before, this is the expected sign for the impact of these variables according to the literature.

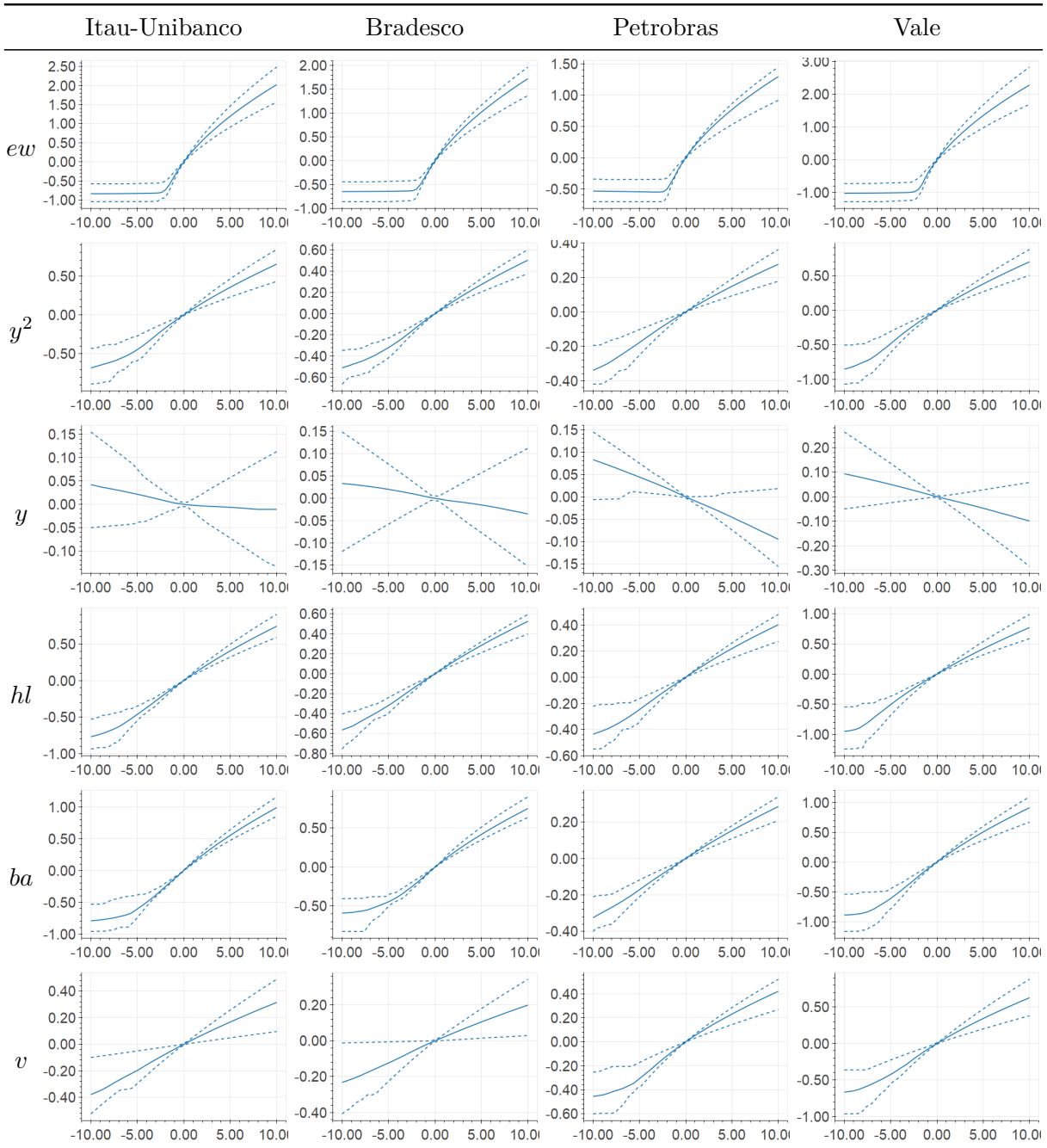
The last three features, W_0, W_1 and W_2 are the seasonality splines. The first two tend to have positive values, with a higher values for the first, while the latter small negative values (with high relative dispersion). This is similar to the estimates of the coefficients β_0, β_1 and β_2 for the model SSM in the table 4, although in that table coefficients for β_1 are small negative on average.

In the second exercise we shock the input features by values ranging from -10 standard deviations and 10 standard deviations, keeping all other features constant. We shock all features, except for the seasonality splines. These shocks were again computed for the whole training sample, like in the previous exercise. The objective is to get an understanding on the non-linearities in the impact of each variable, holding all else constant. Observe that most input variables are fat tailed, so that shocks with 10 standard deviations actually happen frequently in the sample. The results are reported on the figures collected a table 17.

The solid line reflects the mean effect for the shock for each variable, while the dashed lines represent the 25% and 75% percentiles for the shocks. Looking at the figures in table 17 we can readily see that the higher relative dispersion of the impact is attained at the variable y. The other variables have a relatively lower dispersion.

Regarding the non-linearity of the effects, most variables have a S shaped impact format, where the effect for the impact is diminished for high absolute valued shocks. Perhaps the most salient effect is for the ew feature, where the impact seems to have a floor for negative values. The concavity for negative values is also salient for hl, ba and v, for most stocks.

Table 17 – Non-linear sensivity to standardized shocks of selected variables



3.8 Forecasting Exercise Results

We now report the results for the walk-forward forecasting exercise described at section 6. We follow Koopman, Lit and Lucas (2017) and compare the forecasting performance of the models using log-likelihood loss. This measure is used because it takes into account the entire conditional density forecast into account, being able to compare models regarding not only the volatility forecasts but also forecasts for the mean and the format of the conditional distributions, including the tails and the frequency of zeros. So we are evaluating the capacity for the models to forecast not only the volatility, but their ability to forecast

the conditional densities. For the forecast of time t and for all models, we use only data up to $t-1$.

Also following Koopman, Lit and Lucas (2017) and the methodology applied in paper 1, we compare the forecast performance by using Diebold-Mariano (DM) Statistics. In tables 18 and 19 we compute the mean log-likelihood Loss (column Log Loss) for each 10 seconds interval and the Diebold Mariano statistics comparing each model with the models $\text{NN}_{v,\gamma}$ and $\text{NN}_{v,\gamma,\mu}$ (columns $\text{DM}_{v,\gamma}$ and $\text{DM}_{v,\gamma,\mu}$). We used all the 2018 sample for computing this statistic, using the density prediction and the realized return y_t for every 10 seconds period - so this is a huge sample (more than 500,000 points) for each comparison. A positive number means that the model in the row outperforms the model in the column, while a negative number means the opposite.

Table 18 – Forecasting Results - Itau-Unibanco and Bradesco

	Bradesco			Itau-Unibanco		
	Log Loss	$\text{DM}_{v,\gamma}$	$\text{DM}_{v,\gamma,\mu}$	Log Loss	$\text{DM}_{v,\gamma}$	$\text{DM}_{v,\gamma,\mu}$
NN_0	1.761	-30.13	-69.49	1.945	-46.35	-73.06
NN_v	1.755	-6.379	-63.01	1.936	-32.07	-63.85
$\text{NN}_{v,\gamma}$	1.754	-	-62.03	1.933	-	-53.62
$\text{NN}_{v,\gamma,\mu}$	1.737	62.03	-	1.919	53.62	-
EW	1.764	-28.19	-61.14	1.949	-41.11	-64.28
M_1	1.772	-37.02	-63.15	1.959	-49.34	-68.00
M_2	1.801	-78.92	-98.27	1.987	-83.51	-97.81
E	1.789	-53.56	-71.86	1.968	-50.97	-65.34
SS	1.761	-19.09	-53.49	1.948	-37.34	-61.49
SSM	1.742	23.71	-11.68	1.932	1.640	-28.56

Looking at the columns Log Loss and $\text{DM}_{v,\gamma,\mu}$ for both tables 18 and 19, we can see that the model $\text{NN}_{v,\gamma,\mu}$ outperforms all other neural network specifications and also outperform all other comparing models, including the SS and SSM models. This out-performance is highly statistically significant as the DM statistic closest to zero is -11.68 and the this statistic has a standard normal distribution asymptotically.

Looking at the columns Log Loss and $\text{DM}_{v,\gamma}$ we also see that the final NN version with no mean modelled $\text{NN}_{v,\gamma}$ outperforms all models except for the models with mean modelled ($\text{NN}_{v,\gamma,\mu}$ and SSM). This shows the importance of modelling the mean for this kind of models, which is a novelty of the present work - regarding discrete high-frequency densities.

It also worth noting that this consistency in outperforming this set of models were not obtained by the SS model. This model actually did not outperform the EWMA model for Vale at the same sample, as pointed by the table 6 in paper 1. Another interesting point to note is that although $\text{NN}_{v,\gamma}$ did not outperform SSM for all stocks, it did outperform for one of them (Vale).

Table 19 – Forecasting Results - Petrobras and Vale

	Petrobras			Vale		
	Log Loss	$\text{DM}_{v,\gamma}$	$\text{DM}_{v,\gamma,\mu}$	Log Loss	$\text{DM}_{v,\gamma}$	$\text{DM}_{v,\gamma,\mu}$
NN_0	1.476	-32.03	-75.98	2.018	-35.09	-62.28
NN_v	1.473	-17.88	-71.08	2.012	-24.47	-57.65
$\text{NN}_{v,\gamma}$	1.471	-	-68.67	2.010	-	-52.60
$\text{NN}_{v,\gamma,\mu}$	1.448	68.67	-	1.997	52.60	-
EW	1.480	-35.90	-75.36	2.023	-31.29	-53.89
M_1	1.486	-46.21	-80.39	2.033	-41.99	-59.74
M_2	1.500	-71.67	-98.33	2.065	-82.16	-94.76
E	1.517	-98.75	-119.1	2.046	-45.70	-58.24
SS	1.477	-26.17	-71.23	2.027	-37.60	-57.88
SSM	1.453	31.53	-12.58	2.015	-9.123	-36.13

We arrange the comparing DM Statistic for comparing all the models individually in table 20. The calculation is exactly the same as the tables 18 and 19, but now we can compare all pair of models. A positive number means (again) that the model in the row outperforms the model in the columns, where a negative number means the opposite.

We can see that even the simplest neural network model considered, NN_0, outperforms the *EW* model for all stocks - which does not happen even for the SS model, that underperforms *EW* for Vale. NN_0 also outperforms SS for all stocks, altough the outperformance for Bradesco is not statistically significant (DM=-0.5). Recall that this model does not even allow γ to be different from zero and uses the same input variables as SS.

Comparing the performance of NN_v and NN_0 we see that the newly added variables (*ba*, *hl* and *v*) significantly enhances the predictive power for the model. NN_v beats NN_0 by a high margin, with DM statitics above 20 in absolute value terms.

Comparing $\text{NN}_{v,\gamma}$ and NN_v we also see that including the γ the forecasts' loss becomes significantly lower, also by a large margin. Finally, comparing $\text{NN}_{v,\gamma,\mu}$ and $\text{NN}_{v,\gamma}$ we see again the importance of modelling the mean. This appear to be the single most important

feature comparing the forecasting performance, as the DM statistics of this last comparison is even higher in absolute terms then the last comparisons.

Table 20 – DM Statistics Results

Itau-Unibanco

	$NN_{v,\gamma,\mu}$	$NN_{v,\gamma}$	NN_v	NN_0	EW	SS
SSM	-28.6	1.6	7.9	24.8	27.5	42.6
SS	-61.5	-37.3	-27.5	-7.3	3.1	-
EW	-64.3	-41.1	-33.6	-14.7	-	-
NN_0	-73.1	-46.4	-35.0	-	-	-
NN_v	-63.9	-32.1	-	-	-	-
$NN_{v,\gamma}$	-53.6	-	-	-	-	-

Bradesco

	$NN_{v,\gamma,\mu}$	$NN_{v,\gamma}$	NN_v	NN_0	EW	SS
SSM	-11.7	23.7	24.5	37.3	41.4	46.0
SS	-53.5	-19.1	-17.3	-0.5	9.2	-
EW	-61.1	-28.2	-27.0	-12.4	-	-
NN_0	-69.5	-30.1	-28.9	-	-	-
NN_v	-63.0	-6.4	-	-	-	-
$NN_{v,\gamma}$	-62.0	-	-	-	-	-

Petrobras

	$NN_{v,\gamma,\mu}$	$NN_{v,\gamma}$	NN_v	NN_0	EW	SS
SSM	-12.6	31.5	34.7	40.7	47.4	45.6
SS	-71.2	-26.2	-15.3	-4.3	12.8	-
EW	-75.4	-35.9	-29.6	-20.9	-	-
NN_0	-76.0	-32.0	-24.2	-	-	-
NN_v	-71.1	-17.9	-	-	-	-
$NN_{v,\gamma}$	-68.7	-	-	-	-	-

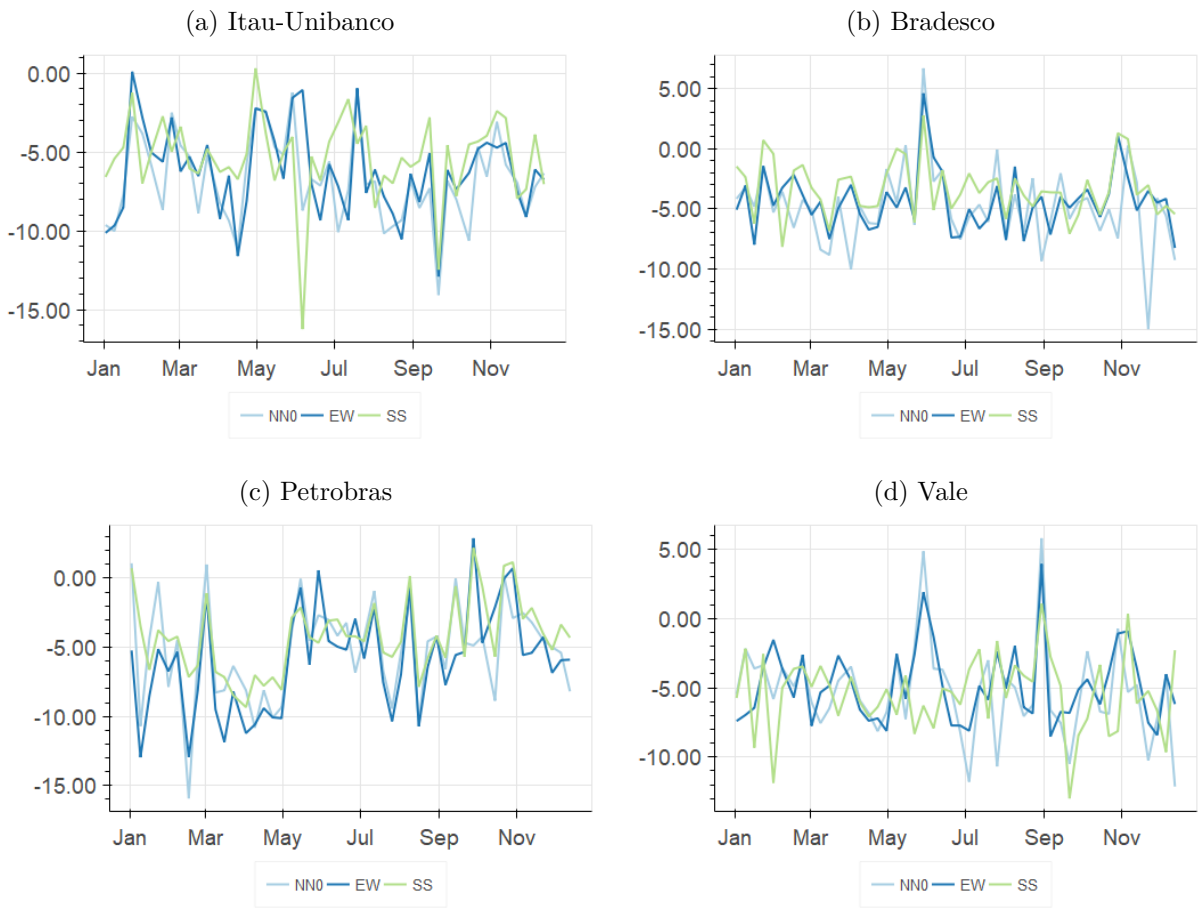
Vale

	$NN_{v,\gamma,\mu}$	$NN_{v,\gamma}$	NN_v	NN_0	EW	SS
SSM	-36.1	-9.1	-5.4	5.8	10.9	33.5
SS	-57.9	-37.6	-32.3	-19.6	-7.9	-
EW	-53.9	-31.3	-26.8	-13.4	-	-
NN_0	-62.3	-35.1	-27.2	-	-	-
NN_v	-57.7	-24.5	-	-	-	-
$NN_{v,\gamma}$	-52.6	-	-	-	-	-

We also computed the DM statistic individually for each 5-day period. We show the results for the comparison with the $\text{NN}_{v,\gamma}$ model in figure 23 and the comparison with the $\text{NN}_{v,\gamma,\mu}$ model in figure 24. We can see in the figures that all previously reported out-performances are consistent through most of the 5-days samples, with few points of exception.

Each point in these figures represent the statistics applied to the 5-days period, which constitutes more than 20.000 data points. In figure 23 we see that the test statistics are similar for all models compared to $\text{NN}_{v,\gamma}$, and they oscillate around the level of -5.

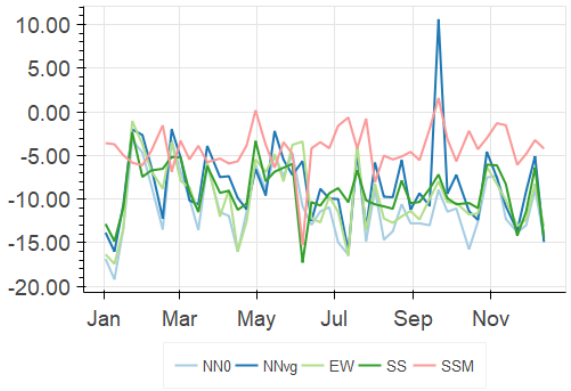
Figure 23 – Weekly DM Score for $\text{NN}_{v,g}$ Model



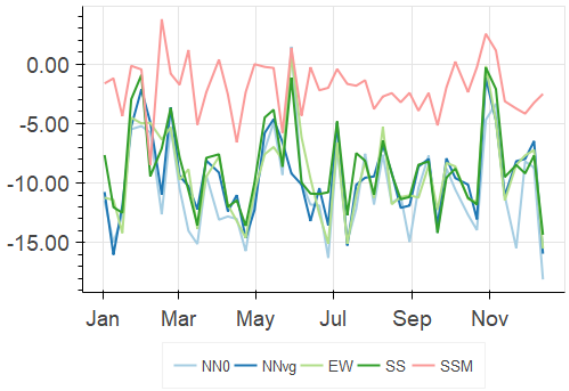
In figure 24 we see that the test statistics are similar for all models compared to $\text{NN}_{v,\gamma,\mu}$, except for the SSM model, and they oscillate around the level of -10. The comparison with the SSM yields a smaller statistic (in absolute value terms), mostly ranging from -2 to -5.

Figure 24 – Weekly DM Score for $NN_{v,g,m}$ Model

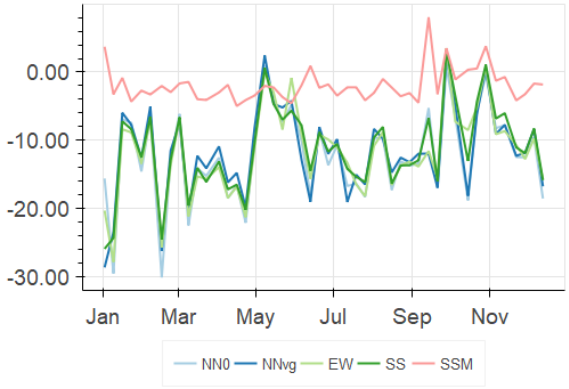
(a) Itau-Unibanco



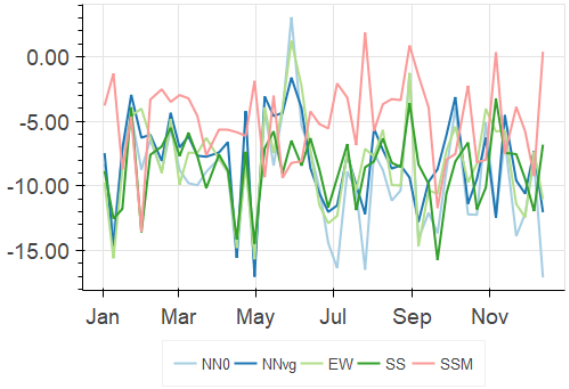
(b) Bradesco



(c) Petrobras



(d) Vale



3.9 Final Remarks

In this paper we developed a new model for forecasting discrete high-frequency densities and volatility. The model is composed of a deep feed-forward neural network to forecast volatility, an AR model for the mean and uses the Modified-Skellam distribution for computing densities. We are unaware of any other works using Deep Learning to predict volatility at this time-scale. Besides the forecasting performance, this model is easier to implement and computationally faster than the State Space counterparts.

We used this model to forecast conditional intraday discrete high-frequency densities and volatilities, contributing to the nascent literature on this subject. In this new model we added new variables not present in previous papers on high frequency volatility forecasting, including the bid-ask spread, the high-low interval spread and the volume traded. We conducted a sensibility analysis of the networks estimated and found that these variables affected the volatility forecasts, finding a positive relation between all of them and the volatility. We also showed that this relation is not linear, appearing to have an S shape

(with more concavity on for negative shocks). We also included the previous price change in the model, but sensibility analysis was not conclusive. We show that the inclusion of these variables increase the prediction power of the model.

We compare the in-sample filtered volatilities, seasonality, and common parameters obtained with the Neural Network Model with the State Space models derived at paper 1 - and find many similarities. But, more interestingly, we conducted an extensive out-sample walk-forward forecasting exercise, involving the entire year of 2018 and about 2 billion data points of price changes of four distinct stocks. The new Neural Network model showed the best forecasting performance for all the stocks when compared to State Space Models and EWMA models. This outperformance was consistent between all the stocks, across time and with and without modeling the mean of the density.

Bibliography

ALLEZ, R.; BOUCHAUD, J.-P. Individual and collective stock dynamics: intra-day seasonalities. *New Journal of Physics*, IOP Publishing, v. 13, n. 2, p. 025010, 2011. 49

ALZAID, A. A.; OMAIR, M. A. et al. On the poisson difference distribution inference and applications. *Bulletin of the Malaysian Mathematical Sciences Society*, Springer Science & Business Media, v. 8, n. 33, p. 17–45, 2010.

ANDERSEN, T. G. Return volatility and trading volume: An information flow interpretation of stochastic volatility. *The Journal of Finance*, Wiley Online Library, v. 51, n. 1, p. 169–204, 1996. 23

ANDERSEN, T. G.; BOLLERSLEV, T. Intraday periodicity and volatility persistence in financial markets. *Journal of empirical finance*, Elsevier, v. 4, n. 2-3, p. 115–158, 1997. 23, 49

ANDERSEN, T. G. et al. Volatility and correlation forecasting. *Handbook of economic forecasting*, Elsevier, v. 1, p. 777–878, 2006. 12

ANDERSEN, T. G. et al. The distribution of realized exchange rate volatility. *Journal of the American statistical association*, Taylor & Francis, v. 96, n. 453, p. 42–55, 2001. 12, 15, 68

ANDERSEN, T. G.; BOLLERSLEV, T.; LANGE, S. Forecasting financial market volatility: Sample frequency vis-a-vis forecast horizon. *Journal of empirical finance*, Elsevier, v. 6, n. 5, p. 457–477, 1999. 15

BARNDORFF-NIELSEN, O. E.; SHEPHARD, N. Econometric analysis of realized volatility and its use in estimating stochastic volatility models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Wiley Online Library, v. 64, n. 2, p. 253–280, 2002. 12, 15, 68

BEKIERMAN, J.; GRIBISCH, B. A mixed frequency stochastic volatility model for intraday stock market returns. 2016. 15

BIBINGER, M. et al. Estimating the spot covariation of asset prices—statistical theory and empirical evidence. *Journal of Business & Economic Statistics*, Taylor & Francis, v. 37, n. 3, p. 419–435, 2019. 49

BLASQUES, F.; KOOPMAN, S. J.; LUCAS, A. Information-theoretic optimality of observation-driven time series models for continuous responses. *Biometrika*, Oxford University Press, v. 102, n. 2, p. 325–343, 2015. 54

BROWNLEES, C. T.; GALLO, G. M. Financial econometric analysis at ultra-high frequency: Data handling concerns. *Computational Statistics & Data Analysis*, Elsevier, v. 51, n. 4, p. 2232–2245, 2006. 17, 58, 77

BUCCI, A. Realized volatility forecasting with neural networks. 2019. 68

CHU, Q. C.; DING, D. K.; PYUN, C. Bid-ask bounce and speads in the foreign exchange futures market. *Review of Quantitative Finance and Accounting*, Springer, v. 6, n. 1, p. 19–37, 1996. 35

CREAL, D.; KOOPMAN, S. J.; LUCAS, A. A dynamic multivariate heavy-tailed model for time-varying volatilities and correlations. *Journal of Business & Economic Statistics*, Taylor & Francis, v. 29, n. 4, p. 552–563, 2011. 55

CREAL, D.; KOOPMAN, S. J.; LUCAS, A. Generalized autoregressive score models with applications. *Journal of Applied Econometrics*, Wiley Online Library, v. 28, n. 5, p. 777–795, 2013. 55

CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, Springer, v. 2, n. 4, p. 303–314, 1989. 70

DONALDSON, R. G.; KAMSTRA, M. An artificial neural network-garch model for international stock return volatility. *Journal of Empirical Finance*, Elsevier, v. 4, n. 1, p. 17–46, 1997. 68, 73

DURBIN, J.; KOOPMAN, S. J. Monte carlo maximum likelihood estimation for non-gaussian state space models. *Biometrika*, Oxford University Press, v. 84, n. 3, p. 669–684, 1997. 25, 27

DURBIN, J.; KOOPMAN, S. J. *Time series analysis by state space methods*. [S.l.]: OUP Oxford, 2012. 24, 25, 29, 30

EASLEY, D.; PRADO, M. M. L. D.; O'HARA, M. The microstructure of the "flash crash": flow toxicity, liquidity crashes, and the probability of informed trading. *The Journal of Portfolio Management*, Institutional Investor Journals Umbrella, v. 37, n. 2, p. 118–128, 2011. 15

ENGLE, R. F.; NG, V. K. Measuring and testing the impact of news on volatility. *The journal of finance*, Wiley Online Library, v. 48, n. 5, p. 1749–1778, 1993. 73

ENGLE, R. F.; SOKALSKA, M. E. Forecasting intraday volatility in the us equity market. multiplicative component garch. *Journal of Financial Econometrics*, Oxford University Press, v. 10, n. 1, p. 54–83, 2012. 15, 68

GLOSTEN, L. R.; JAGANNATHAN, R.; RUNKLE, D. E. On the relation between the expected value and the volatility of the nominal excess return on stocks. *The journal of finance*, Wiley Online Library, v. 48, n. 5, p. 1779–1801, 1993. 73

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. 69, 70, 75, 78

HAJIZADEH, E. et al. A hybrid modeling approach for forecasting the volatility of s&p 500 index return. *Expert Systems with Applications*, Elsevier, v. 39, n. 1, p. 431–436, 2012. 68, 73

HANSEN, P. R.; LUNDE, A. Realized variance and market microstructure noise. *Journal of Business & Economic Statistics*, Taylor & Francis, v. 24, n. 2, p. 127–161, 2006. 12, 15, 68

HARVEY, A.; KOOPMAN, S. J. Forecasting hourly electricity demand using time-varying splines. *Journal of the American Statistical Association*, Taylor & Francis Group, v. 88, n. 424, p. 1228–1236, 1993. 23, 53, 73

HARVEY, A.; LUATI, A. Filtering with heavy tails. *Journal of the American Statistical Association*, Taylor & Francis, v. 109, n. 507, p. 1112–1122, 2014. 55

HARVEY, A. C. Long memory in stochastic volatility. In: *Forecasting volatility in the financial markets*. [S.l.]: Elsevier, 2007. p. 351–363. 15

HARVEY, A. C.; SHEPHARD, N. Estimation of an asymmetric stochastic volatility model for asset returns. *Journal of Business & Economic Statistics*, Taylor & Francis, v. 14, n. 4, p. 429–434, 1996. 15

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. *Neural networks*, Elsevier, v. 2, n. 5, p. 359–366, 1989. 70

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks*, Elsevier, v. 3, n. 5, p. 551–560, 1990. 70

KARLIS, D.; NTZOUFRAS, I. Bayesian modelling of football outcomes: using the skellam's distribution for the goal difference. *IMA Journal of Management Mathematics*, Oxford University Press, v. 20, n. 2, p. 133–145, 2008. 21

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 79

KIRILENKO, A. et al. The flash crash: High-frequency trading in an electronic market. *The Journal of Finance*, Wiley Online Library, v. 72, n. 3, p. 967–998, 2017. 15

KOOPMAN, S.; LIT, R.; NGUYEN, T. *Modified efficient importance sampling using state space methods*. [S.l.], 2012. 26

KOOPMAN, S. J.; LIT, R.; LUCAS, A. intraday stochastic volatility in discrete price changes: the dynamic skellam model. *Journal of the American Statistical Association*, Taylor & Francis, n. just-accepted, 2017. 6, 7, 12, 14, 15, 16, 17, 21, 22, 23, 26, 31, 33, 34, 42, 44, 46, 49, 53, 58, 64, 68, 69, 72, 73, 77, 88, 89

KOOPMAN, S. J. et al. Dynamic discrete copula models for high-frequency stock price changes. *Journal of Applied Econometrics*, Wiley Online Library, v. 33, n. 7, p. 966–985, 2018. 16, 17, 48, 49, 50, 54, 55, 58, 63, 66, 69, 72, 77

KOOPMAN, S. J.; LUCAS, A.; SCHARTH, M. Numerically accelerated importance sampling for nonlinear non-gaussian state-space models. *Journal of Business & Economic Statistics*, Taylor & Francis, v. 33, n. 1, p. 114–127, 2015. 16, 26, 29

KOOPMAN, S. J.; LUCAS, A.; SCHARTH, M. Predicting time-varying parameters with parameter-driven and observation-driven models. *Review of Economics and Statistics*, MIT Press, v. 98, n. 1, p. 97–110, 2016. 54

KRISTJANPOLLER, W.; FADIC, A.; MINUTOLO, M. C. Volatility forecast using hybrid neural network models. *Expert Systems with Applications*, Elsevier, v. 41, n. 5, p. 2437–2442, 2014. 68, 73

LIESENFELD, R.; RICHARD, J.-F. Univariate and multivariate stochastic volatility models: estimation and diagnostics. *Journal of empirical finance*, Elsevier, v. 10, n. 4, p. 505–531, 2003. 16, 26

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. In: *Proc. icml*. [S.l.: s.n.], 2013. v. 30, n. 1, p. 3. 76

MCINISH, T. H.; WOOD, R. A. An analysis of intraday patterns in bid/ask spreads for nyse stocks. *the Journal of Finance*, Wiley Online Library, v. 47, n. 2, p. 753–764, 1992. 35

PINTO, M. S. F. Essays on monte carlo methods for state space models. Amsterdam: Vrije Universiteit, 2012. 26, 28, 29

POIRIER, D. J. Piecewise regression using cubic splines. *Journal of the American Statistical Association*, Taylor & Francis, v. 68, n. 343, p. 515–524, 1973. 23, 53, 74

POWELL, M. J. A direct search optimization method that models the objective and constraint functions by linear interpolation. In: *Advances in optimization and numerical analysis*. [S.l.]: Springer, 1994. p. 51–67. 32

PRADO, M. L. de. *Advances in Financial Machine Learning*. 1st. ed. [S.l.]: Wiley Publishing, 2018. ISBN 1119482089. 74

RICHARD, J.-F.; ZHANG, W. Efficient high-dimensional importance sampling. *Journal of Econometrics*, Elsevier, v. 141, n. 2, p. 1385–1411, 2007. 16, 26

SHAHTAHMASSEBI, G. Bayesian modelling of ultra high-frequency financial data. University of Plymouth, 2011. 21

SHAHTAHMASSEBI, G.; MOYEED, R. Bayesian modelling of integer data using the generalised poisson difference distribution. *International Journal of Statistics and Probability*, v. 3, n. 1, p. 35, 2014. 21

SHEPHARD, N. *Stochastic volatility: selected readings*. [S.l.]: Oxford University Press on Demand, 2005. 68

SHEPHARD, N.; PITT, M. K. Likelihood analysis of non-gaussian measurement time series. *Biometrika*, Oxford University Press, v. 84, n. 3, p. 653–667, 1997. 25, 27

SHEPHARD, N.; YANG, J. J. Continuous time analysis of fleeting discrete price moves. *Journal of the American Statistical Association*, Taylor & Francis, n. just-accepted, 2016. 12

SKLAR, M. Fonctions de repartition an dimensions et leurs marges. *Publ. inst. statist. univ. Paris*, v. 8, p. 229–231, 1959. 50

WYART, M. et al. Relation between bid–ask spread, impact and volatility in order-driven markets. *Quantitative Finance*, Taylor & Francis, v. 8, n. 1, p. 41–57, 2008. 74

XU, X. E.; CHEN, P.; WU, C. Time and dynamic volume–volatility relation. *Journal of Banking & Finance*, Elsevier, v. 30, n. 5, p. 1535–1558, 2006. 74

YANG, D.; ZHANG, Q. Drift-independent volatility estimation based on high, low, open, and close prices. *The Journal of Business*, JSTOR, v. 73, n. 3, p. 477–492, 2000. 74

ZAGORUYKO, S.; KOMODAKIS, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 75

Appendix

APPENDIX A – Computational Aspects

The computations needed for the exercises in this thesis would be very hard to carry a few years ago. Recent advances in IT, both for hardware and software made the tasks in this work less difficult. In this appendix we describe the technologies used in this work.

A.1 Working in Parallel with GPU and CUDA

One of the main advances in hardware for the last decade is the cheap availability of parallel processing. First, we now have access to GPU units, that were popularized in along the 2000's decade. And they were specialized for deep learning tasks over the last decade.

A graphics processing unit (GPU) is a specialized processor unit initially designed to accelerate the creation of images for output to a display device. GPUs are used in embedded systems, mobile phones, personal computers, workstations, and game consoles. Their highly parallel structure makes them more efficient than general-purpose central processing units (CPUs) for algorithms that process large blocks of data in parallel. In a personal computer, a GPU can be present on a video card or embedded on the motherboard.

The chip maker NVIDIA created the CUDA (Compute Unified Device Architecture), which is a parallel computing platform and application programming interface (API) model. It allows software developers and software engineers to use a graphics processing unit (GPU) for general purpose processing. The CUDA platform is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements, for the execution of compute kernels. In top of CUDA, nVidia maintains a set of specialized libraries. Three of them are of great interest for the computations involved in this work: cuBLAS, cuSOLVER and cuDNN.

The NVIDIA cuBLAS library is a fast GPU-accelerated implementation of the standard basic linear algebra subroutines (BLAS). NVIDIA cuSOLVER library provides a collection of dense and sparse direct solvers with the intent of providing useful LAPACK-like features, such as common matrix factorization and triangular solve routines for dense matrices, a sparse least-squares solver and an eigenvalue solver. Lastly, the NVIDIA CUDA Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks. The cuDNN library provides highly efficient implementations for standard

routines such as forward and backward convolution, pooling, normalization, and activation layers.

The CUDA framework became the standard framework for general processing in GPU's. Fortunately end users do not need to deal with this infrastructure directly. The availability of cuBLAS, cuSOLVER and cuDNN made easy for software developers to develop high level packages that build in the CUDA architecture in order to make scientific computing. As in the case of BLAS and LAPACK for the CPU's, most users actually access the libraries transparently, using high level wrappers. This is the case for Pytorch, Tensorflow and MXNet, which are deep learning frameworks backed by Google, Facebook and Apache Foundation, respectively. They provide generic access to classes implementing Deep Learning features that can run either in CPU or GPU, with almost the same code. They also provide generic matrix / tensor libraries that make the use of BLAS, LAPACK, cuBLAS and cuSOLVER transparent to the user, computing the transformations on both CPU and GPU with the same interface.

For the present work we computed all the Neural Network training procedures on the GPU, running four training problems at the same time. We used a gaming notebook with a CUDA enabled NVIDIA GeForce GPU. The networks were coded using Pytorch framework. We trained four models, for four stocks and 48 periods (768 models) for 2000 epochs. The training took less then 1 day to complete. We estimate that using the CPU the computation wold be about 20 times slower.

A.2 Distributed Computing: AWS EC2 Cluster

Parallelizing computations over the GPU is a great and economical way of processing many tasks in a short period of time. However, this approach is no panacea: there are limitations and practical difficulties in using GPU.

First, not all problems are well suited for processing in parallel. A great example is time series analysis. Most time series models contain some time dependency, which likely involves a recurrence that usually must be computed sequentially. The task of computing an AR model for 20.000 timestamps or running a Kalman filter for the same lenght cannot be parallelized. And these applications were ubiquitous in papers 1 and 2 computations. When the task cannot be parallelized, it makes absolutely no sense to use GPU's, as each processor in the GPU has limited capabilities when compared with the CPU. In these cases computing in GPU is actually slower. The GPU main advantage is the number of processors and this advantage is not relevant in these scenarios.

Second, some times it is hard to write a code that runs entirely on the GPU even when the problem can be parallelized. Taking the example of the Kalman filtering, one cannot parallelize the filtering procedure from one time series, but could make the filtering of several time-series at the same time. And this would be the case of the Kalman Simulator Smoothing algorithm used to sample for Gaussian distribution obtained by the NAIS procedure. So that part of the algorithm could be parallelized. But in order to do this on the GPU, one would need to write extremely efficient compiled functions for the Kalman Filtering over the GPU, with custom Kernels, which is hard and time consuming. Another difficulty faced is the possible unavailability of certain specialized mathematical functions over the GPU.

So we opted to compute the models of the first two papers on the CPU. The computations involved the optimization of 2 models for 4 stocks over 48 periods in the first paper, and 2 models for 2 pairs of stocks over 48 periods in the second paper. The forecasting part was also computationally expensive for the State Space models in both papers, as the Bootstrap Particle Filter was used. These two papers involved the computation of the estimation procedure of 960 models and the computation of the particle filtering for 768 models. Computing all these models over a single desktop using one CPU would take many days to complete, possibly some weeks. So we opted to use a cluster on Amazon AWS EC2 to make such computations, distributing the computing process through 5 instances, each with 72 processors each. we used the c5.18xlarge instance type. The computations took about 1 day to complete.

A.3 Approximating Functions

Another difficulty faced is the unavailability of certain specialized mathematical functions over the GPU. This is the case for the modified Bessel I function, needed for the Skellam pmf computation. Actually, even the computation of this function on the CPU is too slow, as it involves the expansion of series for computing each value. The function might not be numerically stable also, even when using the exponential scaled version.

So we opted to approximate this function by splines. The same was done with the 2 dimensional Gaussian CDF function, needed for the Gaussian copula computation. The implementations became faster and more numerically stable, maintaining the differentiability needed for the optimizations.

A.4 Compiling Python Code

A final important topic concern how to generate low level efficient code for implementing specialized algorithms needed for the paper. The language of choice for this paper was python because of its easy of use, it does have large scientific and generic computing libraries and because it is open source. But the language is not particularly fast. This is not a problem if the problem at hand is not computationally intensive or if most of the computation can be done by calling compiled code already available in python libraries. Unfortunately the computations for this work did not fell into any of these two categories.

Using python to write the time-series loops needed for the algorithms in this work would lead to implementations 5 to 10 times slower than low level code implementations, making the computations unfeasible in practice. Writing this code directly in a low level language would be too time consuming for the purposes of the current work. Our solution involved generating low level compiled code using python package Numba. Numba allows on-the-fly/transparent compilation of python code, approaching low level C/Fortran Speed. Only a subset of the Python language is supported, so some effort compatible code is needed. But it involves much less effort than writing C/Fortran specialized functions.