Modelling Intraday Covariance

Bruno Morier[a] and Pedro L. Valls Pereira[b]

[a]Head of Quantitative Strategies - Banco Safra
[b]Sao Paulo School of Economics-FGV and CEQEF-FGV. Rua Itapeva 474, 10o andar
ZIP code: 01332-000 Sao Paulo, SP, Brazil
Corresponding author, E-mail: pedro.valls@fgv.br

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

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 et al. (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 Morier and Valls Pereira (2023b) 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 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.

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

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:

$$p(y_t;\theta_t)=\sum_{j_1\in\{0,1\}}\cdots\sum_{j_n\in\{0,1\}}(-1)^{j_1+\ldots+j_n}C\left[F_1(y_{1t}-j_1;\theta_{1t}^m),\ldots,F_n(y_{nt}-j_n;\theta_{nt}^m);\theta_t^c\right]. \tag{2}$$

In the bivariate case the equation (2) becomes:

$$\begin{aligned}p(y_t;\theta_t)&=C\left[F_1(y_{1t};\theta_{1t}^m),F_2(y_{2t};\theta_{2t}^m)\right]-C\left[F_1(y_{1t}-1;\theta_{1t}^m),F_2(y_{2t};\theta_{2t}^m)\right]\\&-C\left[F_1(y_{1t};\theta_{1t}^m),F_2(y_{2t}-1;\theta_{2t}^m)\right]+C\left[F_1(y_{1t}-1;\theta_{1t}^m),F_2(y_{2t}-1;\theta_{2t}^m)\right].\end{aligned} \tag{3}$$

The formula in the equation (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 $F_i(y_{it}-1;\theta_{it}^m)$ is too close to $F_i(y_{it};\theta_{it}^m)$ 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). Observe that when $F_i(y_{it}-1;\theta_{it}^m)$ is close $F_i(y_{it};\theta_{it}^m)$ 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).

$$\begin{aligned}p(y_t;\theta_t)\simeq\hat{p}(y_t;\theta_t)&=c(u_{1t}^*,\ldots,u_{nt}^*;\theta_t^c)\prod_{i=1}^n\left[F_i(y_{it};\theta_{it}^m)-F_i(y_{it}-1;\theta_{it}^m)\right]\\&=c(u_{1t}^*,\ldots,u_{nt}^*;\theta_t^c)\prod_{i=1}^n p_i(y_{it};\theta_{it}^m),\end{aligned} \tag{4}$$

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:

$$C(u;R) := \Phi_R\left(\Phi^{-1}(u_1),\ldots,\Phi^{-1}(u_n)\right), \tag{5}$$

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:

$$c(u;R) := \frac{1}{\sqrt{\det R}}\exp\left(-\frac{1}{2}\left(\Phi^{-1}(u_1),\ldots,\Phi^{-1}(u_n)\right)\left(R^{-1}-I\right)\left(\Phi^{-1}(u_1),\ldots,\Phi^{-1}(u_n)\right)^T\right). \tag{6}$$

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

$$R_t = \begin{bmatrix} 1 & \rho_t \\ \rho_t & 1 \end{bmatrix}. \tag{7}$$

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 $\theta^m_{1t},\ldots,\theta^m_{nt},\theta^c_t$ 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:

$$y_{it}|\sigma^2_{it} \sim p_i(y_{it};\mu_{it},\sigma^2_{it},\gamma). \tag{8}$$

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

$$p_i(y_t;j,l,k,\mu_{it},\sigma^2_{it},\gamma) := \begin{cases} p_s(y_t;\mu_{it},\sigma^2_{it}) & \text{for } y_t \notin \{j,l,k\} \\ p_s(y_t;\mu_{it},\sigma^2_{it}) - \gamma\frac{\Delta}{2} & \text{for } y_t \in \{j,l\} \\ p_s(y_t;\mu_{it},\sigma^2_{it}) + \gamma\Delta & \text{for } y_t = k, \end{cases} \tag{9}$$

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:

$$p_s(y;\mu,\sigma^2) := \exp(-\sigma^2)\left(\frac{\sigma^2+\mu}{\sigma^2-\mu}\right)I_{|y|}(\sqrt{\sigma^4-\mu^2}), \tag{10}$$

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,

$$\mu_{it} = \delta_i y_{i,t-1}. \tag{11}$$

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

$$\begin{aligned} \sigma_{it}^2 &= \overline{\sigma_i}^2 s_{it} \exp(\alpha_{it}) \\ \alpha_{i,t+1} &= \phi_i \alpha_{i,t} + \eta_{i,t} \\ \eta_{it} &\sim \text{NID}(0, \sigma_{\eta,i})^2. \end{aligned} \tag{12}$$

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 et al. (2017). We write s_{it} as a zero-sum cubic spline function

$$s_{it} = \beta_i' W_t, \tag{13}$$

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

$$\theta_{it}^m = \{\mu_{it}, \sigma_{it}, \gamma\}, \tag{14}$$

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:

$$\begin{aligned} \rho_t &= \tanh(\overline{\rho} + \alpha_{0,t} + s_{0,t}) \\ \alpha_{0,t+1} &= \phi_0 \alpha_{0,t} + \eta_{0,t} \\ \eta_{0t} &\sim \text{NID}(0, \sigma_{\eta,0})^2, \end{aligned} \tag{15}$$

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

$$s_{0t} = \beta_0' W_t. \tag{16}$$

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

$$\theta_t^c = (\rho_t), \tag{17}$$

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

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 et al. (2015)). Second, they have similar forecasting performance as their parameter driven peers, even when the latter are actually the true data generating process (Koopman et al. (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

$$\theta_{t+1} = \omega + A\nabla_t + B\theta_t, \tag{18}$$

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

$$\nabla_t = \frac{\partial \log p(y_t; \theta_t)}{\partial \theta_t}. \tag{19}$$

This specification of the score dynamics follows Creal et al. (2011, 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

$$\begin{aligned} A &= \text{diag}(a) \\ B &= \text{diag}(b), \end{aligned} \tag{20}$$

for constant vectors a and b.

The analytical formula for the score function ∇_t coordinates is given by equations (20) and (21):

$$\nabla_{it}^m = \frac{\partial \log p(y_t; \theta_t)}{\partial \theta_{it}^m} = \sum_{j_1 \in \{0,1\}} \cdots \sum_{j_n \in \{0,1\}} \frac{(-1)^{j_1 + \ldots + j_n}}{p(y_t; \theta_t)} \frac{\partial C(u_{1t}, \ldots, u_{nt}; \theta_t^c)}{\partial u_{it}} \frac{\partial u_{it}}{\partial \theta_{it}^m} \tag{21}$$

$$\nabla_t^c = \frac{\partial \log p(y_t; \theta_t)}{\partial \theta_t^c} = \sum_{j_1 \in \{0,1\}} \cdots \sum_{j_n \in \{0,1\}} \frac{(-1)^{j_1 + \ldots + j_n}}{p(y_t; \theta_t)} \frac{\partial C(u_{1t}, \ldots, u_{nt}; \theta_t^c)}{\partial \theta_t^c}. \tag{22}$$

This equation makes more apparent the need for use the approximation of equation (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 (4) is preferable. The corresponding derivatives for that equation are

given by equations (23) and (24):

$$\nabla_{it}^{m} = \frac{\partial \log p_i(y_{it}; \theta_{it}^{m})}{\partial \theta_{it}^{m}} \tag{23}$$

$$\nabla_{t}^{c} = \frac{\partial \log c(u_{1t}^{*}, \ldots, u_{nt}^{*}; \theta_{t}^{c})}{\partial \theta_{t}^{c}}. \tag{24}$$

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 (25) and (26):

$$\sigma_{it}^{2} = \exp(\theta_{it}^{m}) \tag{25}$$

$$\rho_t = \frac{\theta_t^c}{\sqrt{1 + (\theta_t^c)^2}}. \tag{26}$$

We calculated the exact likelihood and exact derivatives by using equations (3), (21) and (22) whenever numerically feasible, that is, when:

$$\left\| F_i(y_{it}; \theta_{it}^{m}) - F_i(y_{it} - 1; \theta_{it}^{m}) \right\|_{\infty} > \varepsilon_0 \tag{27}$$

for a fixed parameter ε_0. Otherwise we use expressions (23) and (24) whenever the expression in equation (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 (21), (22), (23) and (24) for our specific model.

First, notice that $p(y_t; \theta_t)$ can be calculated using equations (3) and (5), when calculating in the exact form, and can be calculated by (4) and (6) in the approximated form. Regarding the derivatives for the copula function, we also note that it can be written as a conditional copula:

$$\frac{\partial C(u_{1t}, u_{2t}; \theta_t^c)}{\partial u_{1t}} = P(U_{2t} \leq u_{2t} | U_{1t} = u_{1t}) \tag{28}$$

So in our case of Gaussian copula the equation (28) becomes

$$\frac{\partial C(u_{1t}, u_{2t}; \theta_t^c)}{\partial u_{1t}} = \Phi\left(\frac{\Phi^{-1}(u_{2t}) - \rho_t \Phi^{-1}(u_{1t})}{\sqrt{1 - \rho_t^2}} \right). \tag{29}$$

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

$$\frac{\partial \Phi_2(x, y; \rho)}{\partial \rho} = \frac{1}{2\pi\sqrt{1 - \rho^2}} \exp\left(-\frac{x^2 - 2\rho x y + y^2}{2(1 - \rho^2)} \right), \tag{30}$$

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

we obtain:

$$\frac{\partial C(u_{1t}, u_{2t}; \theta_t^c)}{\partial \theta_t^c} = (1+\theta_t^c)^{-3/2} \frac{\partial \Phi_2}{\partial \rho}(\Phi^{-1}(u_{1t}), \Phi^{-1}(u_{2t}); \rho_t). \tag{31}$$

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:

$$\frac{\partial u_{it}}{\partial \sigma_{it}^2} = \exp(-\sigma_{it}^2) \sum_{\nu=-\infty}^{y_{it}} \left[\left(\frac{\nu}{\sigma_{it}^2} - 1 \right) I_{|\nu|}(\sigma_{it}^2) + I_{|\nu+1|}(\sigma_{it}^2) \right]. \tag{32}$$

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

$$\frac{\partial u_{it}}{\partial \theta_{it}^m} = \sigma_{it}^2 \frac{\partial u_{it}}{\partial \sigma_{it}^2}. \tag{33}$$

5 Data

Like in Morier and Valls Pereira (2023b), 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.[1] 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 Morier and Valls Pereira (2023b) and reproduced here in Par I: Data of the supplementary material .

[1] Also this procedure makes the time series equally spaced, as it is the way that we should model time series using the standard techniques.

6 Estimation Procedure

In this section we describe the model estimation procedure. Like in Morier and Valls Pereira (2023b), 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 et al. (2017), that estimated the models with sample size equal to one day. So we have about 100,000 data points for each estimation.

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 (3) and (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.

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 (3) and (4), but the θ_t is updated according to equation (18). The resulting log-likelihood function was then optimized used the BFGS method, with the derivatives estimated numerically.

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 Morier and Valls Pereira (2023b), we start by describing the descriptive statistics of the coefficient estimated obtained by the rolling estimation procedure previously described.

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 1. 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 1: 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 2 and 3. 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 2: 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)

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 4 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

Table 3: 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)

the value of ϕ drops significantly. This might have been caused by idiosyncratic events on these specific dates. As happened in the estimates in Morier and Valls Pereira (2023b), 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 4 shows the parameter estimates for the C model.

Tables 5 and 6 shows the parameter estimates for the CS model.

Table 4: Estimates for parameters in C model

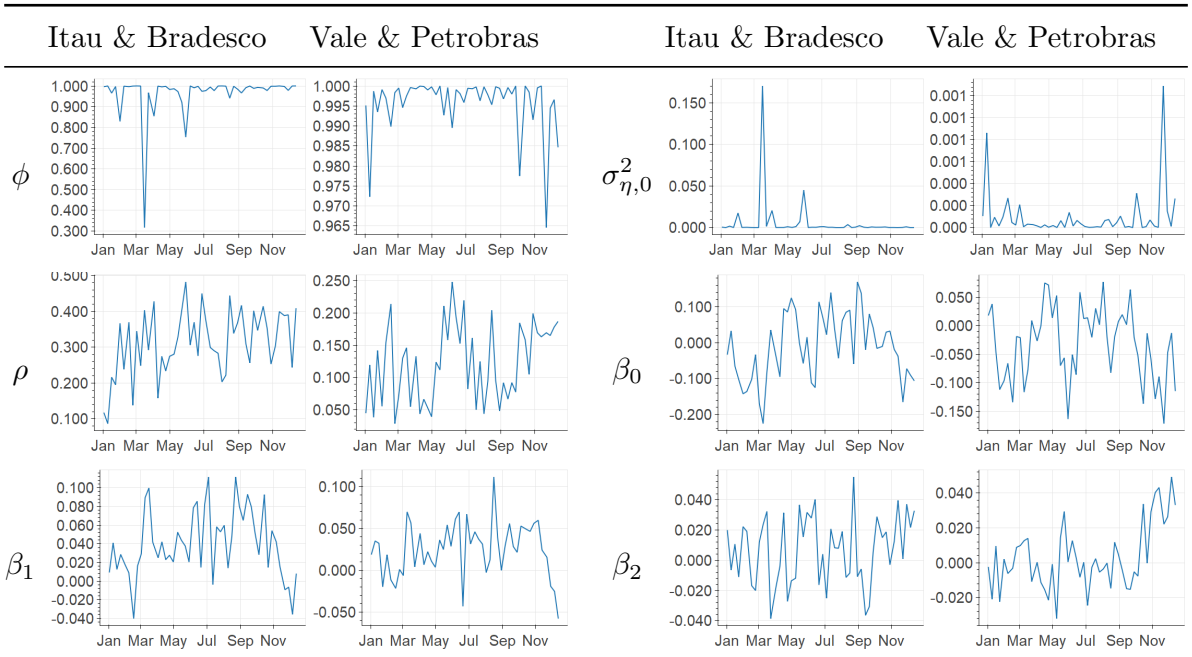


Table 5: Estimates for parameters a and b in CS model

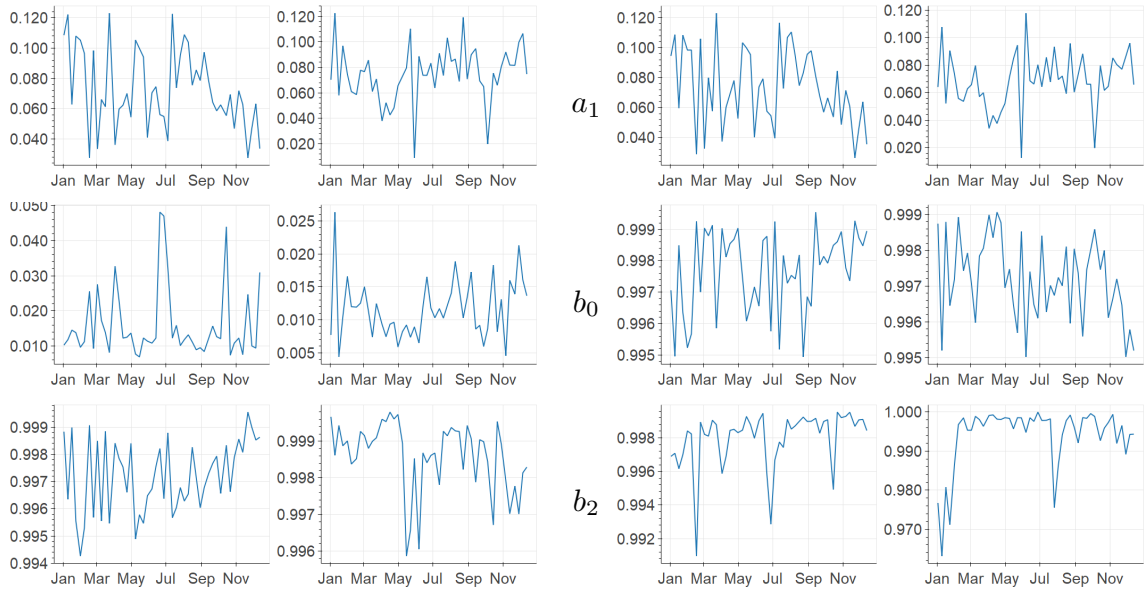
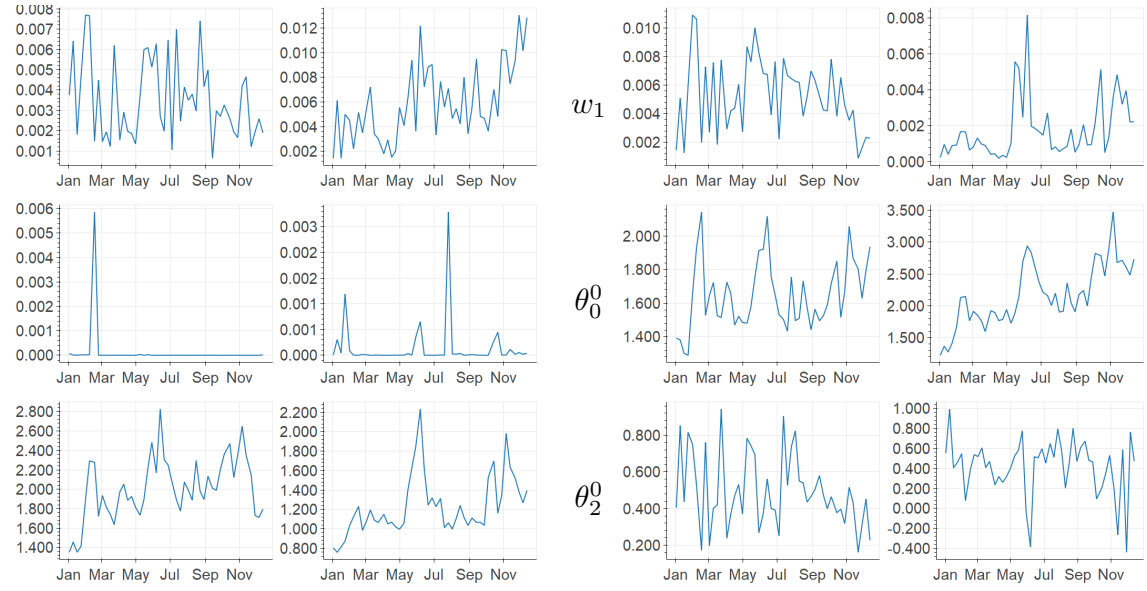
	Itau & Bradesco	Vale & Petrobras	Itau & Bradesco	Vale & Petrobras
				

Table 6: Estimates for parameters w and θ^0 in CS model

	Itau & Bradesco	Vale & Petrobras	Itau & Bradesco	Vale & Petrobras
				

7.2 Seasonality

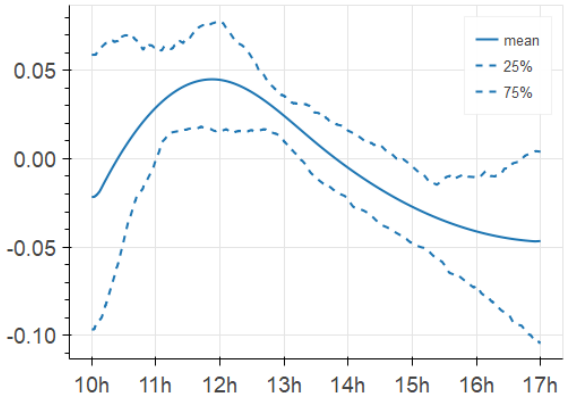
In this sub-section we analyze the descriptive statistics for the volatility seasonality.

Figure 4 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)).

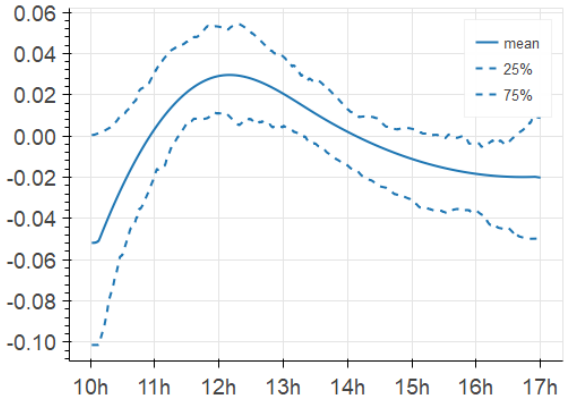
As mentioned before, one possible explanation is that most idiosyncratic news related to stocks are released when the market is closed.

Figure 1: Average Seasonality for model C in 2018

(a) Itau-Unibanco & Bradesco



(b) Vale & Petrobras



7.3 Filtered Correlation Forecasts

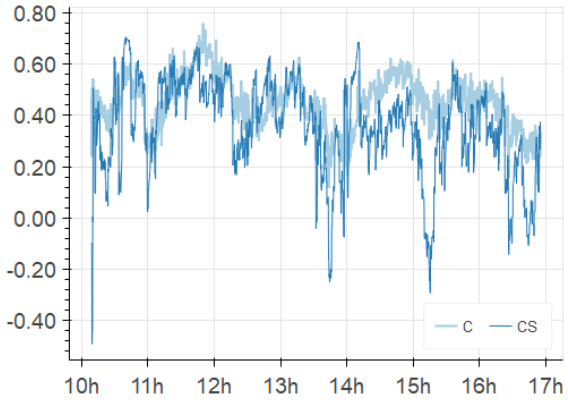
We now show an example of the filtered correlation obtained by the Bootstrap Particle Filter applied for the C model and by applying the appropriate transformation to θ_t^c for the CS model. Figure 2 shows the path of the filtered correlation for the day July 6, 2018 for both models. Observe that the 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.

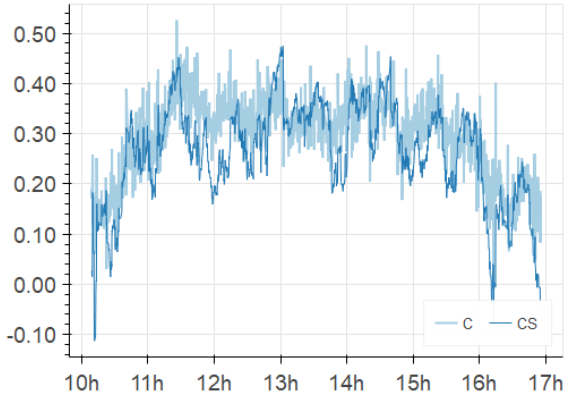
This is a robustness check on the procedure of both models, as they are totally different.

Figure 2: Filtered Correlation Forecast for July 6, 2018

(a) Itau-Unibanco & Bradesco



(b) Vale & Petrobras



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 et al. (2017) and used the log-likelihood loss function, which is simply minus the log of the pmf obtained by each model.

In the table 7 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.

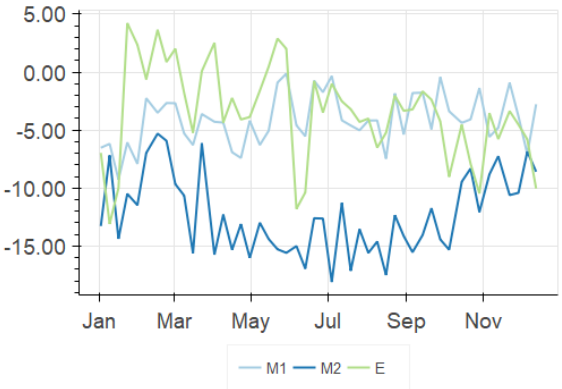
Table 7: Forecasting Results

	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 3 and the comparison with the C model in figure 4. We can see in the figures that all previously reported out-performances are consistent through all the 5-days samples.

Figure 3: Weekly DM Score comparison for CS Model

(a) Itau-Unibanco & Bradesco



(b) Vale & Petrobras

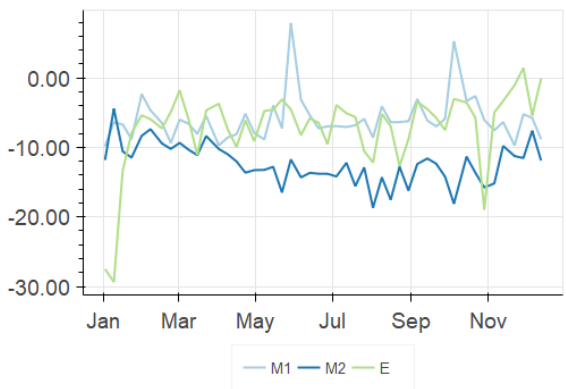
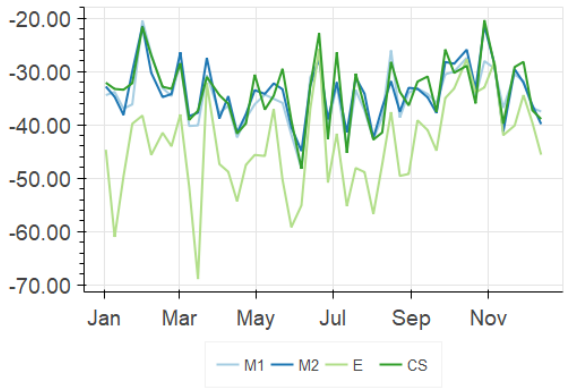
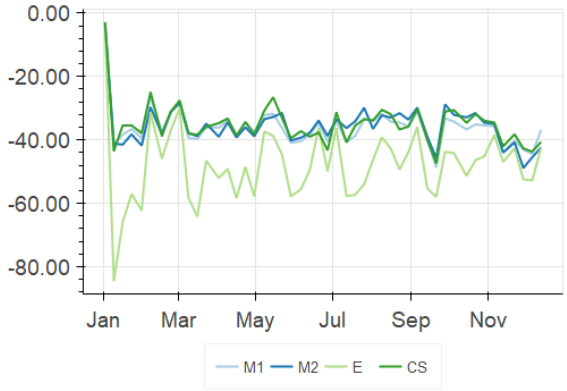


Figure 4: Weekly DM Score comparison for C Model

(a) Itau-Unibanco & Bradesco



(b) Vale & Petrobras



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 Morier and Valls Pereira (2023b).

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.

References

Allez, R. and Bouchaud, J.-P. (2011). Individual and collective stock dynamics: intra-day seasonalities. *New Journal of Physics*, 13(2):025010.

Andersen, T. G. and Bollerslev, T. (1997). Intraday periodicity and volatility persistence in financial markets. *Journal of Empirical Finance*, 4(2-3):115–158.

Bibinger, M., Hautsch, N., Malec, P., and Reiss, M. (2019). Estimating the spot covariation of asset prices—statistical theory and empirical evidence. *Journal of Business & Economic Statistics*, 37(3):419–435.

Blasques, F., Koopman, S. J., and Lucas, A. (2015). Information-theoretic optimality of observation-driven time series models for continuous responses. *Biometrika*, 102(2):325–343.

Brownlees, C. T. and Gallo, G. M. (2006). Financial econometric analysis at ultra-high frequency: Data handling concerns. *Computational Statistics & Data Analysis*, 51(4):2232–2245.

Creal, D., Koopman, S. J., and Lucas, A. (2011). A dynamic multivariate heavy-tailed model for time-varying volatilities and correlations. *Journal of Business & Economic Statistics*, 29(4):552–563.

Creal, D., Koopman, S. J., and Lucas, A. (2013). Generalized autoregressive score models with applications. *Journal of Applied Econometrics*, 28(5):777–795.

Harvey, A. and Koopman, S. J. (1993). Forecasting hourly electricity demand using time-varying splines. *Journal of the American Statistical Association*, 88(424):1228–1236.

Harvey, A. and Luati, A. (2014). Filtering with heavy tails. *Journal of the American Statistical Association*, 109(507):1112–1122.

Koopman, S. J., Lit, R., and Lucas, A. (2017). intraday stochastic volatility in discrete price changes: the dynamic skellam model. *Journal of the American Statistical Association*, 112(520):1490–1503.

Koopman, S. J., Lit, R., Lucas, A., and Opschoor, A. (2018). Dynamic discrete copula models for high-frequency stock price changes. *Journal of Applied Econometrics*, 33(7):966–985.

Koopman, S. J., Lucas, A., and Scharth, M. (2015). Numerically accelerated importance sampling for nonlinear non-gaussian state-space models. *Journal of Business & Economic Statistics*, 33(1):114–127.

Koopman, S. J., Lucas, A., and Scharth, M. (2016). Predicting time-varying parameters with parameter-driven and observation-driven models. *Review of Economics and Statistics*, 98(1):97–110.

Morier, B. and Valls Pereira, P. L. (2023a). Forecasting intraday volatility and densities using deep learning. Technical report, CEQEF Working papers.

Morier, B. and Valls Pereira, P. L. (2023b). Modeling high frequency intraday returns by non-linear statespace models. Technical report, CEQEF Working papers.

Poirier, D. J. (1973). Piecewise regression using cubic splines. *Journal of the American Statistical Association*, 68(343):515–524.

Sklar, M. (1959). Fonctions de repartition an dimensions et leurs marges. *Publ. inst. statist. univ. Paris*, 8:229–231.

Supplementary Material for the paper "Modelling Intraday Covariance" by Bruno Morier and Pedro L. Valls Pereira

In the supplemntary material for the paper Forecasting Intraday Volatility and Densities using Deep Learning it is presented to parts. This first part is a description of the data used in this paper and also in Morier and Valls Pereira (2023b). The second part is the computational aspects for this paper.

Part I: 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 et al. (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 et al. (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 et al. (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 8 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.

Table 8: 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

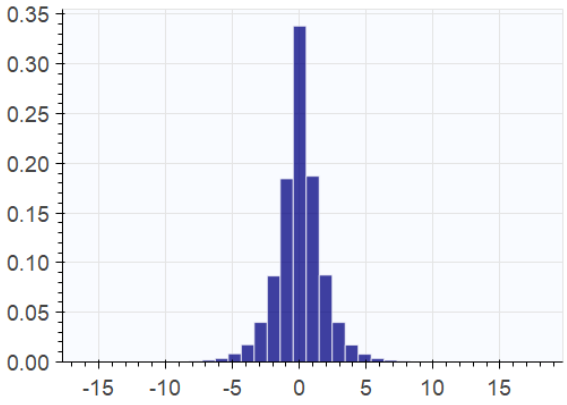
Note: We report prices as the number of 10 seconds closing prices considered in the sample, %0 as the percentage of 0 price changes, $\% \pm 1$ as the percentage of $\% \pm 1$ price changes, std as the standard deviation, min as the minimum price change in ticks, max as the maximum price change in ticks, 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.

In figure 5 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 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.

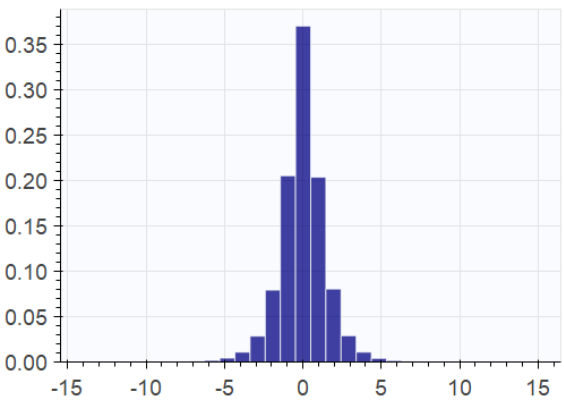
We also note from figure 5 that the least volatile stock in terms of tick volatility, Petrobras, attains less values most of the time for its price changes, while compared with the others. This does not mean that Petrobras stock is less volatile than the others considering usual price log returns . In 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 5: Histogram for price changes in 2018

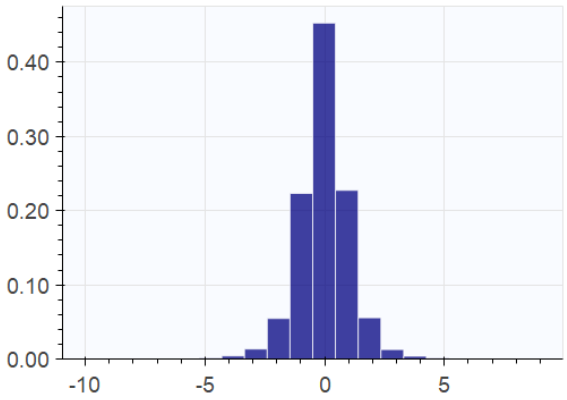
(a) Itau-Unibanco



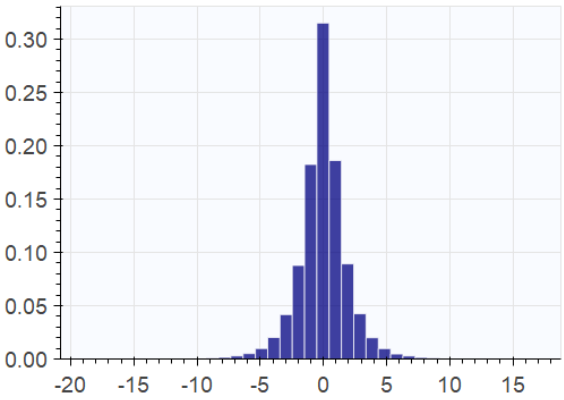
(b) Bradesco



(c) Petrobras

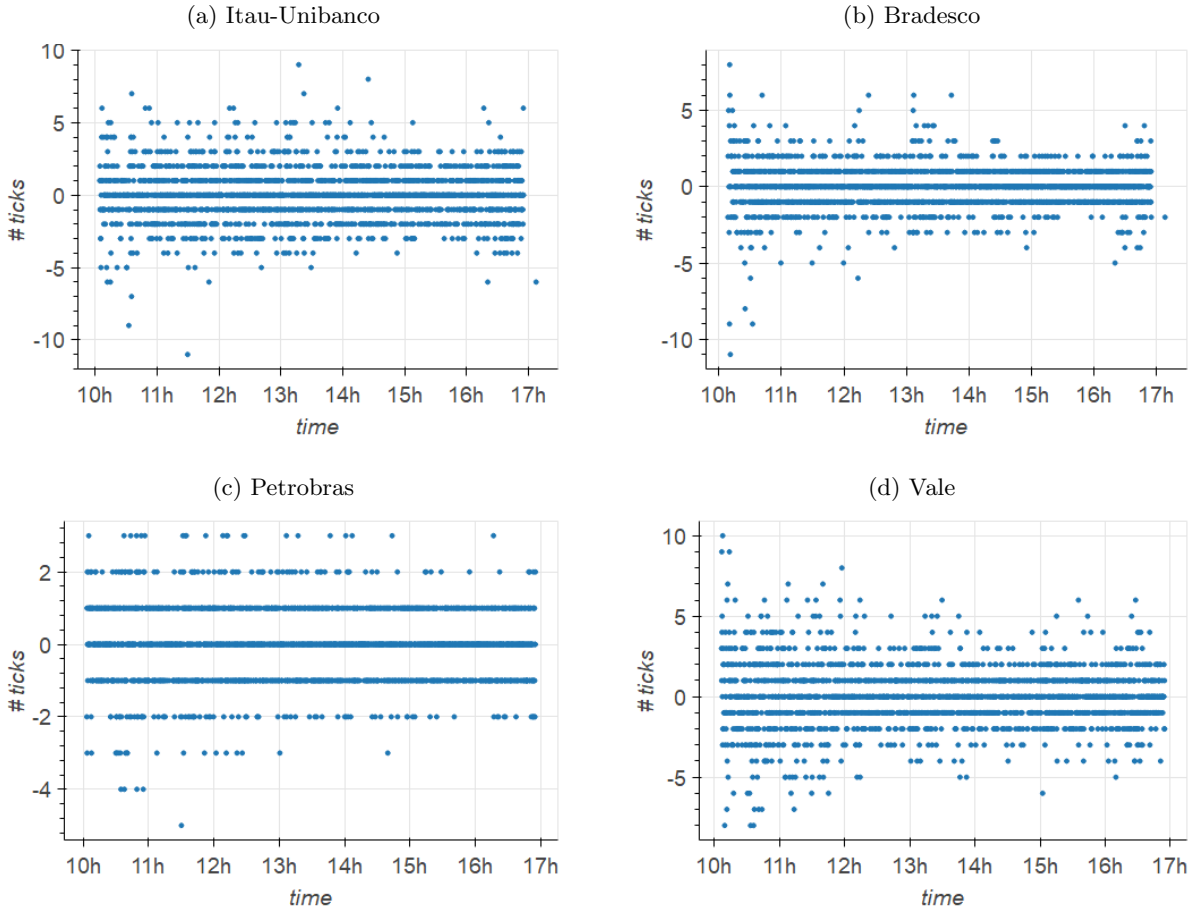


(d) Vale



Note: In figure 6 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 6: Price changes for assets on June 20, 2018



Part II: Computational Aspects

The computations needed for the exercises in this article 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 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.

B 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 this paper and Morier and Valls Pereira (2023b)

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 this paper and Morier and Valls Pereira (2023a) on the CPU. The computations involved the optimization of 2 models for 4 stocks over 48 periods in Morier and Valls Pereira (2023a), and 2 models for 2 pairs of stocks over 48 periods in this 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.

C 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.

D 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.